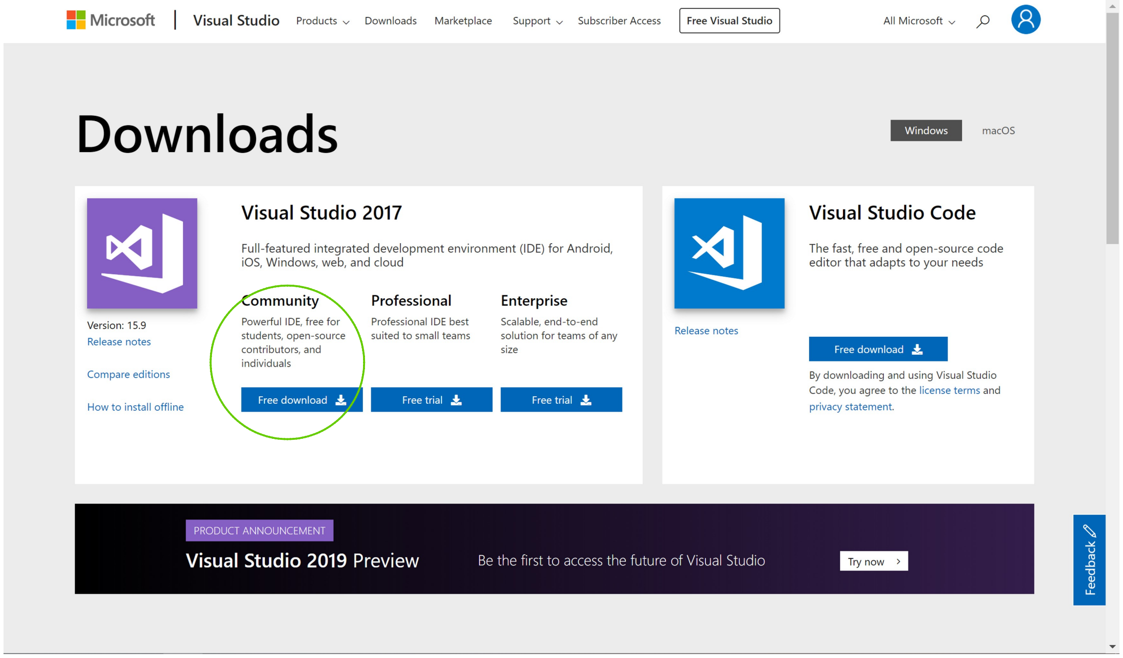
OpenGL 3.0+ and Visual Studio 2017 Setup

Wednesday, February 13, 2019 12:50 PM

By Yijia Liu

Steps:

1. Download and install Visual Studio 2017
   a. Use this link: https://visualstudio.microsoft.com/downloads/
   b. Choose the Community Version. It is free for academic use.


Microsoft | Visual Studio Products Downloads Marketplace Support Subscriber Access Free Visual Studio All Microsoft
Windows macOS
Free download
Free trial
Free trial
Free download
Try now
Feedback

Downloads

Visual Studio 2017

Full-featured integrated development environment (IDE) for Android, iOS, Windows, web, and cloud

Version: 15.9
Release notes
Compare editions
How to install offline

Community
Powerful IDE, free for students, open-source contributors, and individuals

Professional
Professional IDE best suited to small teams

Enterprise
Scalable, end-to-end solution for teams of any size

Visual Studio Code

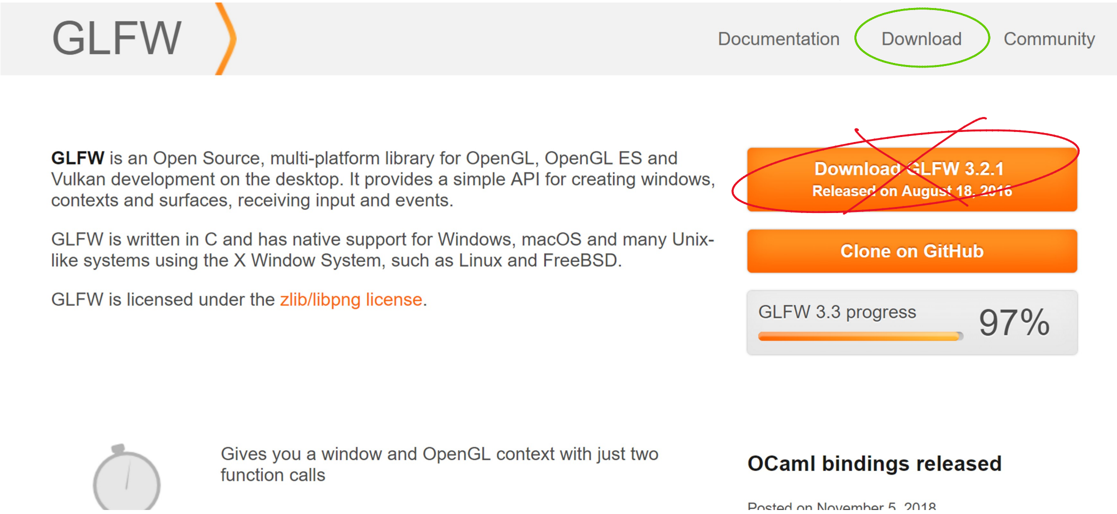
The fast, free and open-source code editor that adapts to your needs

Release notes

By downloading and using Visual Studio Code, you agree to the license terms and privacy statement.

PRODUCT ANNOUNCEMENT
Visual Studio 2019 Preview
Be the first to access the future of Visual Studio

   c. Install Visual Studio 2017.
2. Download GLFW
   a. Use this link: https://www.glfw.org/
   b. Don't click the "Download GLFW 3.x.x", click the "Download" on top right.



GLFW is an Open Source, multi-platform library for OpenGL, OpenGL ES and Vulkan development on the desktop. It provides a simple API for creating windows, contexts and surfaces, receiving input and events.

GLFW is written in C and has native support for Windows, macOS and many Unix-like systems using the X Window System, such as Linux and FreeBSD.

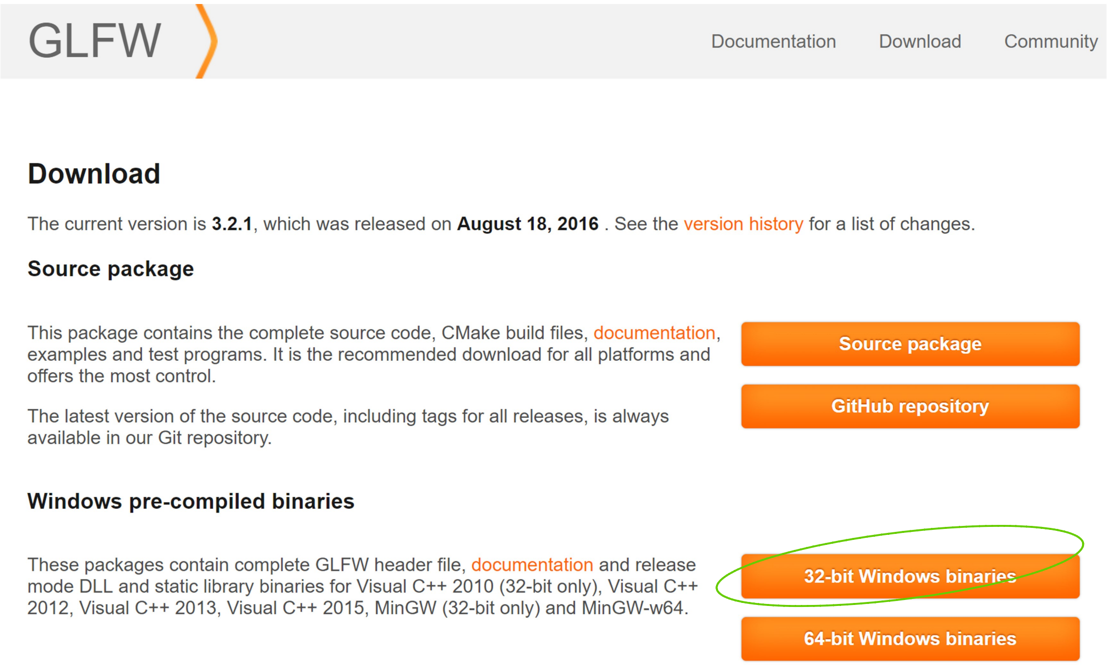
GLFW is licensed under the zlib/libpng license.

Gives you a window and OpenGL context with just two function calls

OCaml bindings released

Posted on November 5, 2018

   c. On next page, scroll to "Windows pre-compiled binaries", you either choose the "32-bit Windows binaries", or the 64-bit one. Most of the time you only need the 32-bit one, especially for beginner proposes.



Download

The current version is **3.2.1**, which was released on **August 18, 2016** . See the version history for a list of changes.

Source package

This package contains the complete source code, CMake build files, documentation, examples and test programs. It is the recommended download for all platforms and offers the most control.

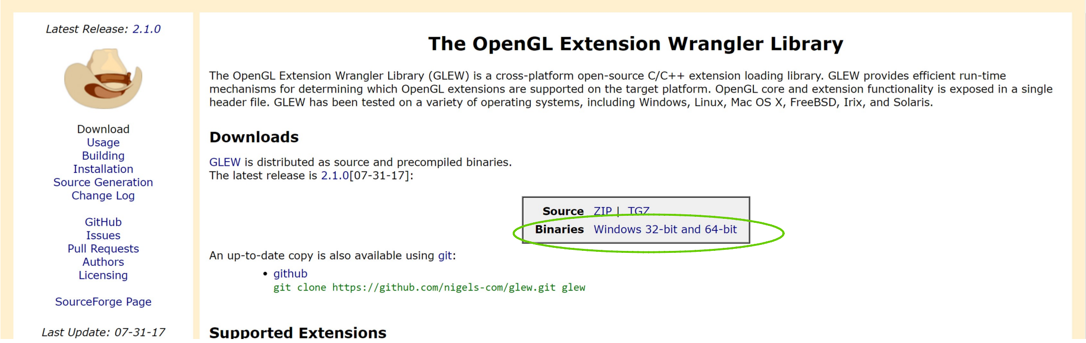
The latest version of the source code, including tags for all releases, is always available in our Git repository.

Windows pre-compiled binaries

These packages contain complete GLFW header file, documentation and release mode DLL and static library binaries for Visual C++ 2010 (32-bit only), Visual C++ 2012, Visual C++ 2013, Visual C++ 2015, MinGW (32-bit only) and MinGW-w64.

   d. Click and download the zip file.

3. Download GLEW.
   a. Use this link: http://glew.sourceforge.net/ (Or Google "glew")
   b. Select "Windows 32-bit and 64-bot"



The OpenGL Extension Wrangler Library

The OpenGL Extension Wrangler Library (GLEW) is a cross-platform open-source C/C++ extension loading library. GLEW provides efficient run-time mechanisms for determining which OpenGL extensions are supported on the target platform. OpenGL core and extension functionality is exposed in a single header file. GLEW has been tested on a variety of operating systems, including Windows, Linux, Mac OS X, FreeBSD, Irix, and Solaris.

Downloads

GLEW is distributed as source and precompiled binaries.
The latest release is 2.1.0[07-31-17]:

Source	ZIP | TGZ
Binaries	Windows 32-bit and 64-bit

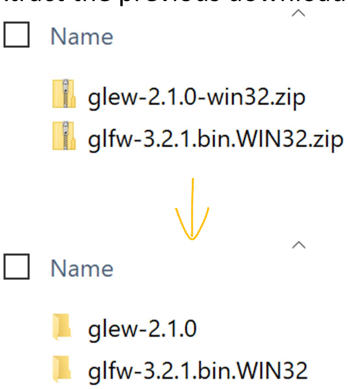
An up-to-date copy is also available using git:

- github
  `git clone https://github.com/nigels-com/glew.git glew`

Supported Extensions

   c. Click and download the zip file.

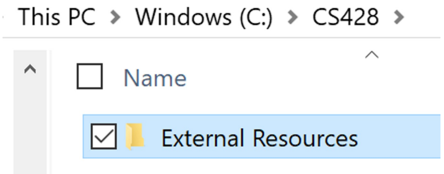
4. Folder Placement (Setup of the OpenGL programming environment)
   Now we have the VS2017, the latest GLFW and the latest GLEW. We can beginning.
   a. Extract the previous downloaded GLFW and GLEW zip file.



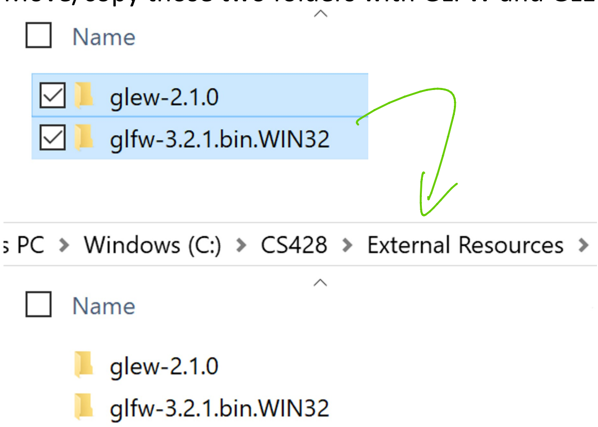
   b. Now we create another folder for our C++ project directory
      Let's name it "CS428".
      You can create this folder anywhere for you convenience.
      For example I created one on "C:\CS428".

You can later rename this folder (the root folder), move it to a different location / computer, and it will still run. Because library files will be link to this folder.

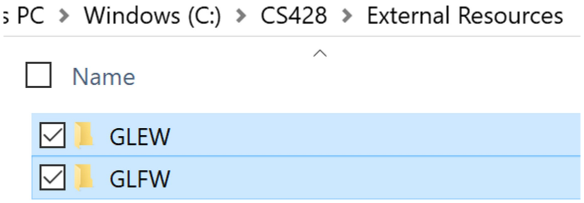
c. Inside "CS428", create a folder where the GLFW and GLEW library will be placed. I name it "External Resources", you can name it whatever you like.



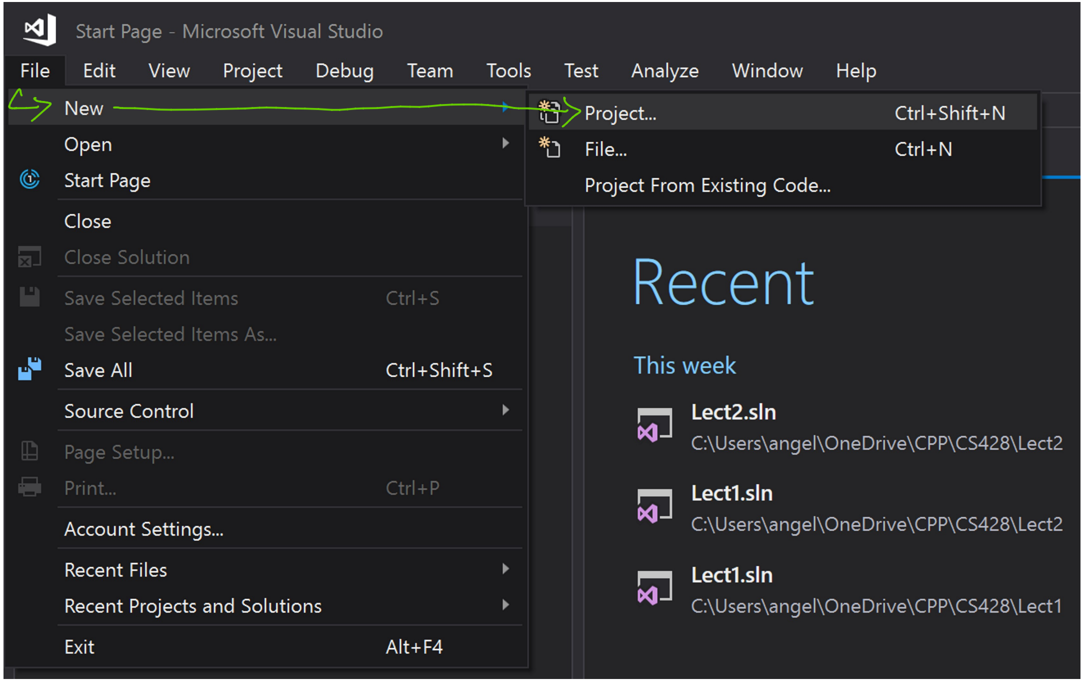
d. Move/copy those two folders with GLFW and GLEW libraries into "External Resources



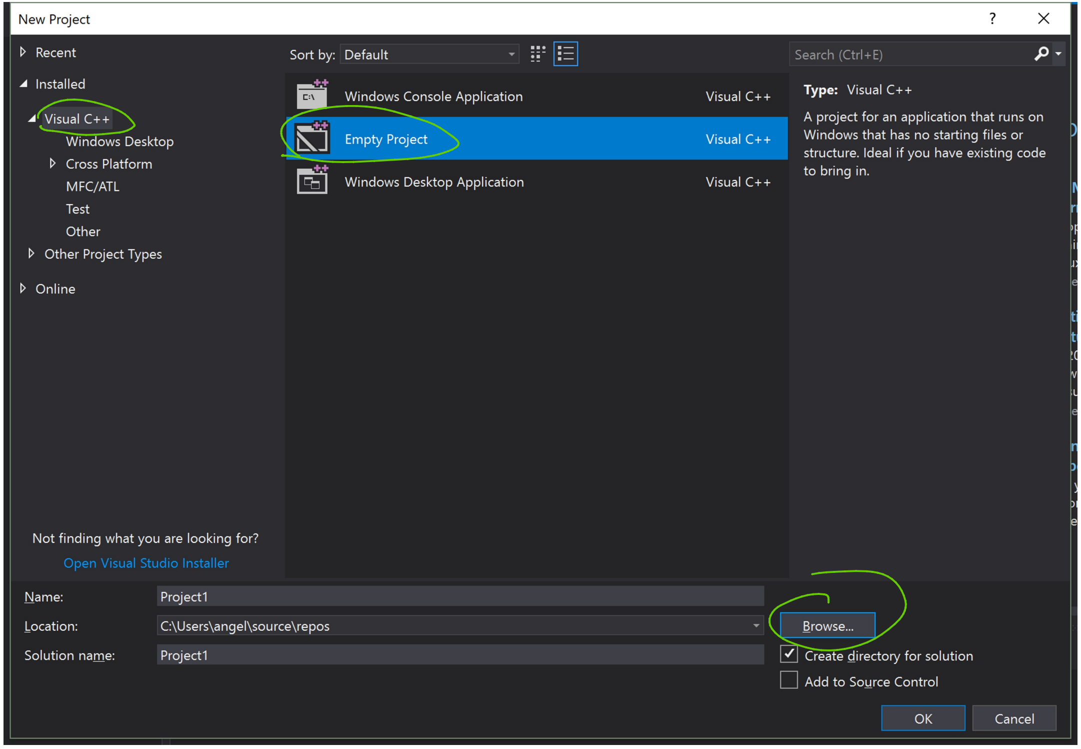
e. I suggest renaming GLFW and GLEW libraries folder for easier to maintain.



5. Setup a new project in Visual Studio 2017
   a. Launch VS2017
   b. File -> New -> Project


Start Page - Microsoft Visual Studio
File Edit View Project Debug Team Tools Test Analyze Window Help
New
Open
Start Page
Close
Close Solution
Save Selected Items Ctrl+S
Save Selected Items As...
Save All Ctrl+Shift+S
Source Control
Page Setup...
Print... Ctrl+P
Account Settings...
Recent Files
Recent Projects and Solutions
Exit Alt+F4
Project... Ctrl+Shift+N
File... Ctrl+N
Project From Existing Code...
Recent
This week
Lect2.sln
C:\Users\angel\OneDrive\CPP\CS428\Lect2
Lect1.sln
C:\Users\angel\OneDrive\CPP\CS428\Lect2
Lect1.sln
C:\Users\angel\OneDrive\CPP\CS428\Lect1

c. Choose: Visual C++ -> Empty Project -> Browse…



Type: Visual C++

A project for an application that runs on Windows that has no starting files or structure. Ideal if you have existing code to bring in.

Name: Project1

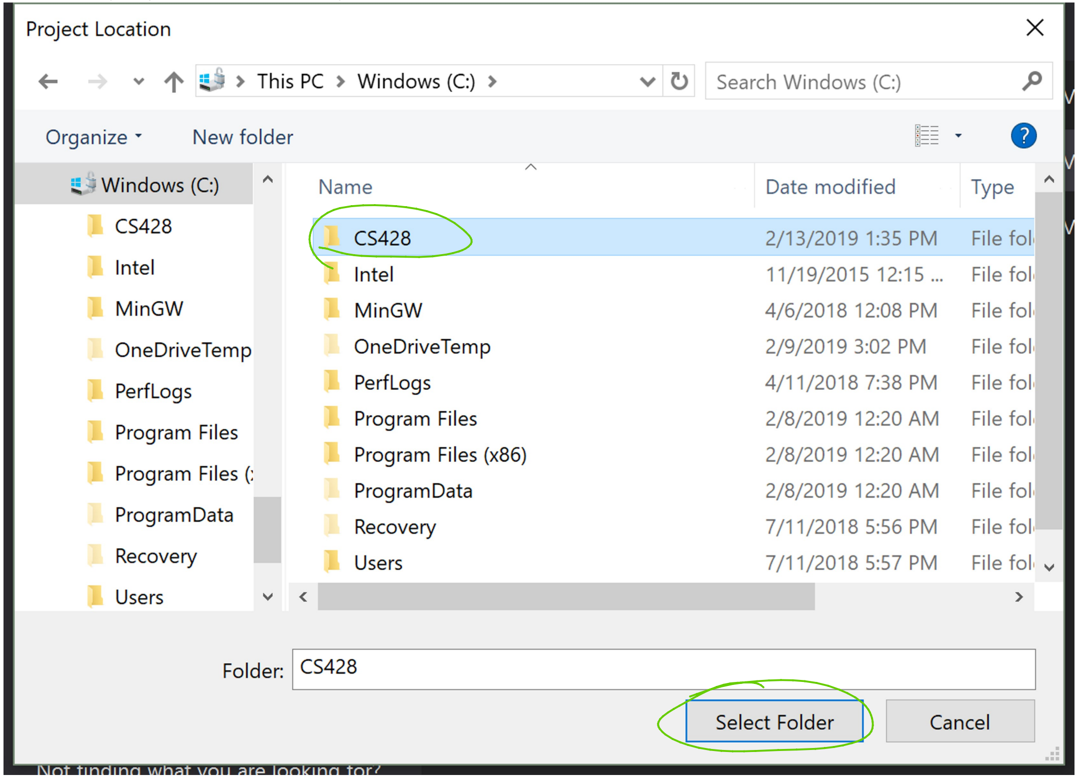
Location: C:\Users\angel\source\repos

Solution name: Project1

☑ Create directory for solution

□ Add to Source Control

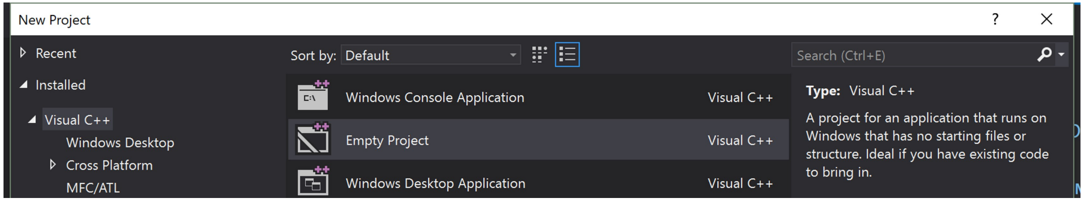
d. Find the project root folder you created. Click "select".



Name	Date modified	Type
CS428	2/13/2019 1:35 PM	File fol
Intel	11/19/2015 12:15 ...	File fol
MinGW	4/6/2018 12:08 PM	File fol
OneDriveTemp	2/9/2019 3:02 PM	File fol
PerfLogs	4/11/2018 7:38 PM	File fol
Program Files	2/8/2019 12:20 AM	File fol
Program Files (x86)	2/8/2019 12:20 AM	File fol
ProgramData	2/8/2019 12:20 AM	File fol
Recovery	7/11/2018 5:56 PM	File fol
Users	7/11/2018 5:57 PM	File fol

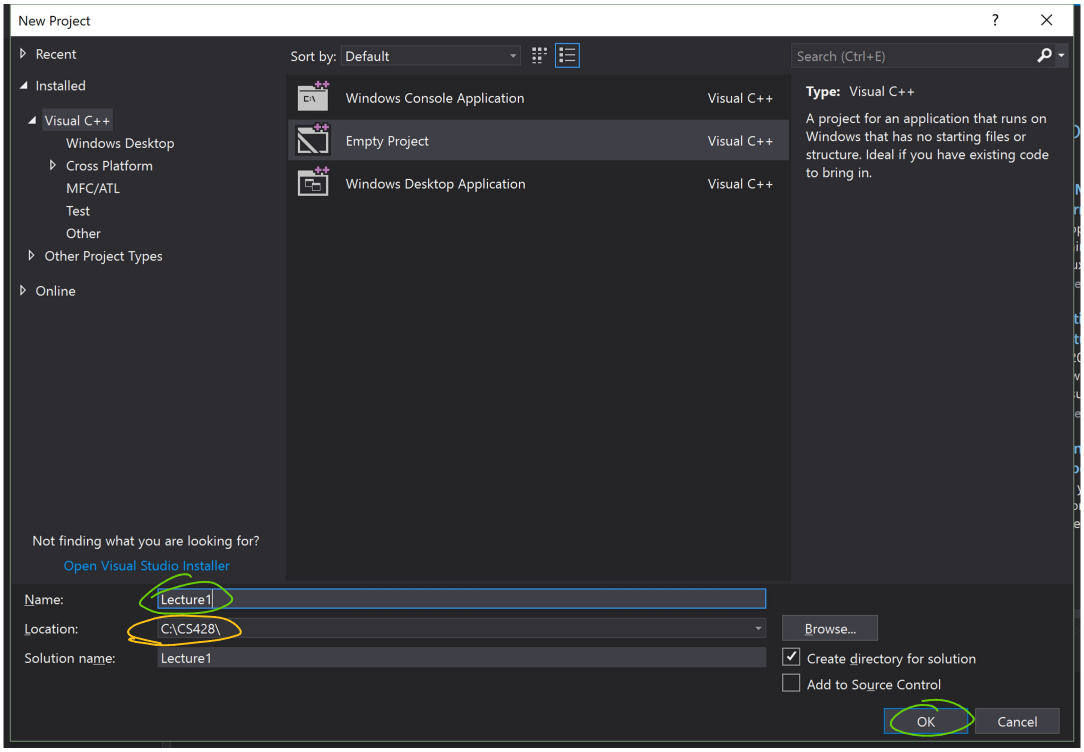
Folder: CS428

e. You see the Location of the Project is changed.
Now we can name the project, for example: Lecture1



Type: Visual C++

A project for an application that runs on Windows that has no starting files or structure. Ideal if you have existing code to bring in.



Type: Visual C++

A project for an application that runs on Windows that has no starting files or structure. Ideal if you have existing code to bring in.

Name: Lecture1

Location: C:\CS428\

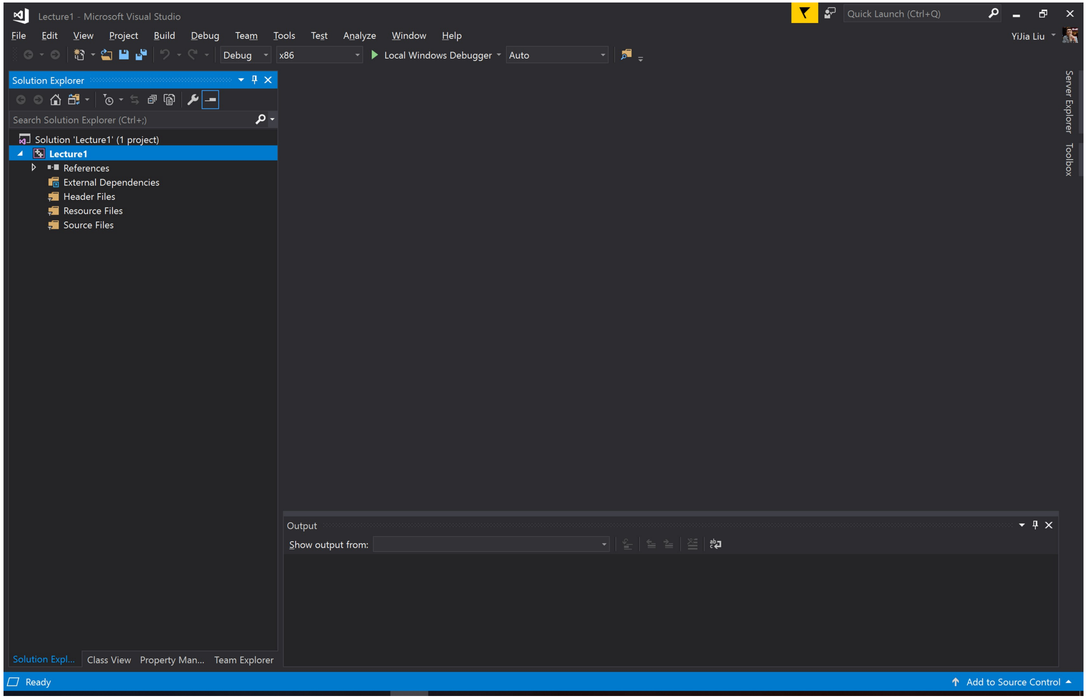
Solution name: Lecture1

☑ Create directory for solution

□ Add to Source Control

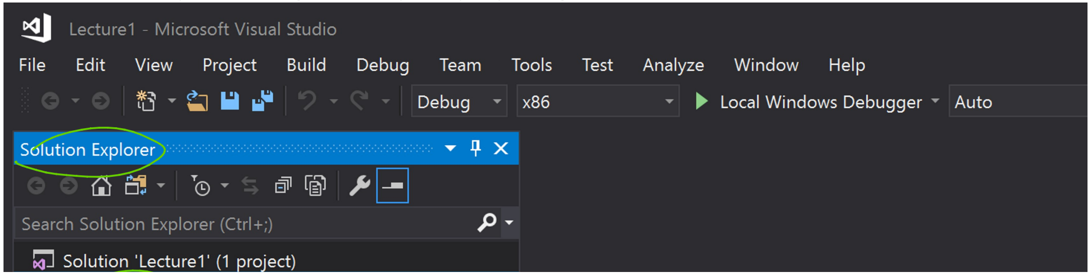
Click "OK".

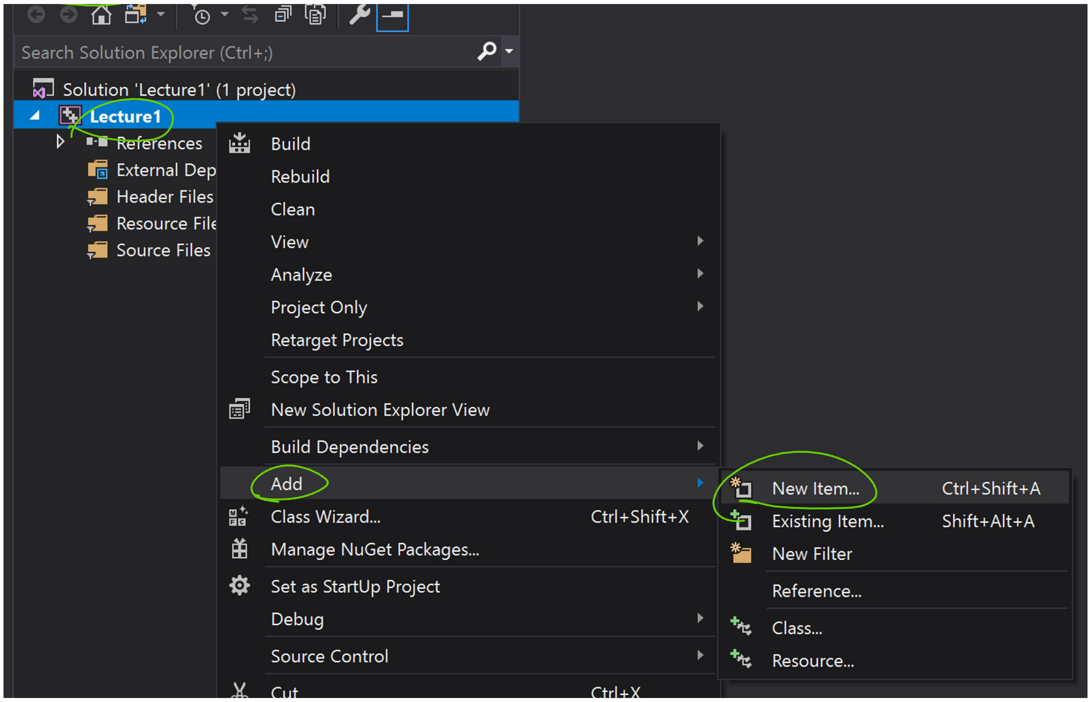
f. You have your new project



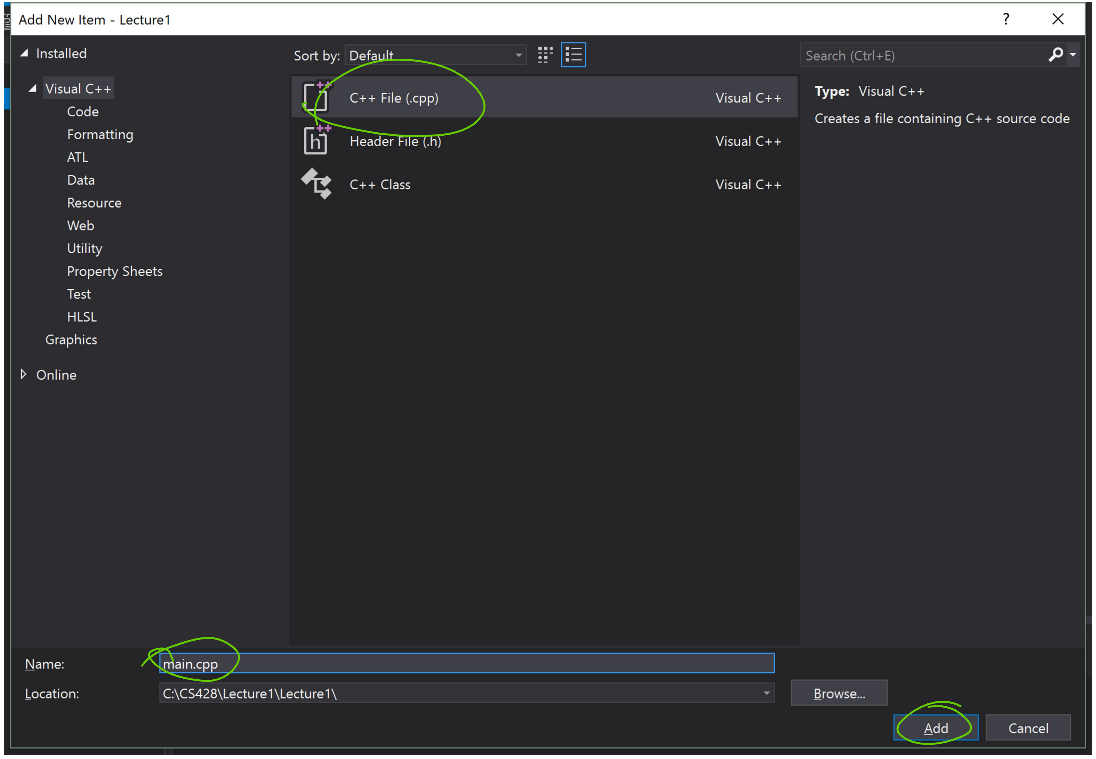
6. Create main.cpp

   a. In "Solution Explorer", right click your project, go to Add -> New Item...





b. rename "main.cpp" -> Add



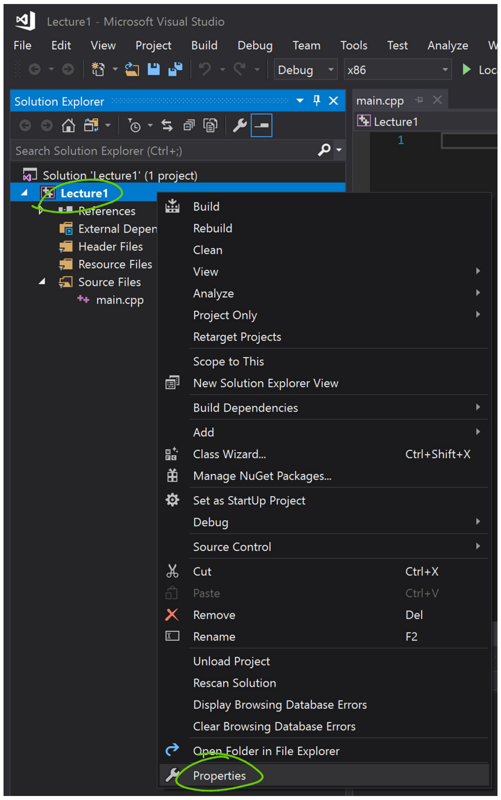
Type: Visual C++

Creates a file containing C++ source code

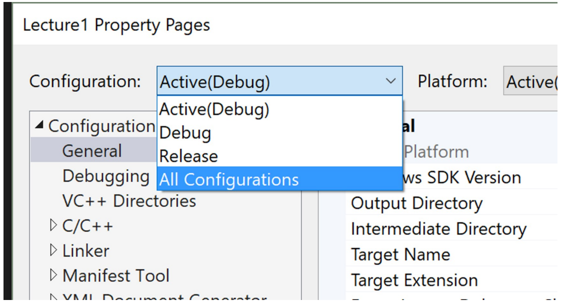
Name: main.cpp

Location: C:\CS428\Lecture1\Lecture1\

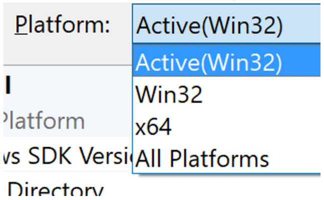
7. Link libraries to project
    a. Right click your project -> Property



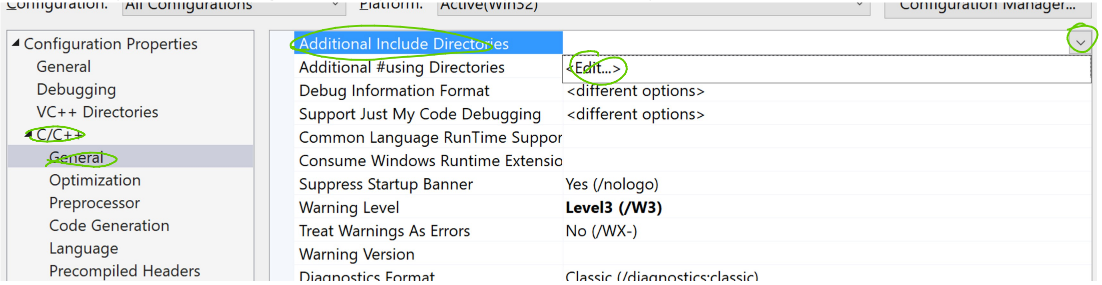
b. On next window, make sure:
Configuration: "All Configuration"



Platform: Win32 if you chose 32-bit

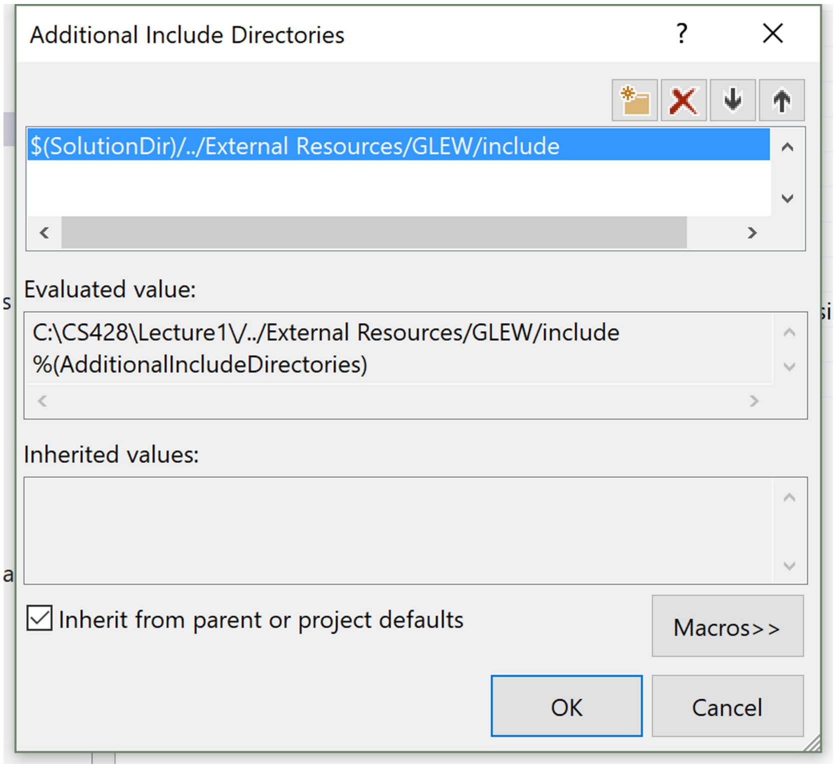


c. Go to: C/C++ -> General -> Additional Include Directories -> Edit
(If you skipped creating "main.cpp" in the previous step", "C/C++" tab can't be found.)



Additional Include Directories
Additional #using Directories <Edit...>
Debug Information Format <different options>
Support Just My Code Debugging <different options>
Common Language RunTime Suppor
Consume Windows Runtime Extensio
Suppress Startup Banner Yes (/nologo)
Warning Level **Level3 (/W3)**
Treat Warnings As Errors No (/WX-)
Warning Version
Diagnostics Format Classic (/diagnostics:classic)

d. In the pop-up window, type in $(SolutionDir)/../External Resources/GLEW/include



Additional Include Directories

$(SolutionDir)/../External Resources/GLEW/include

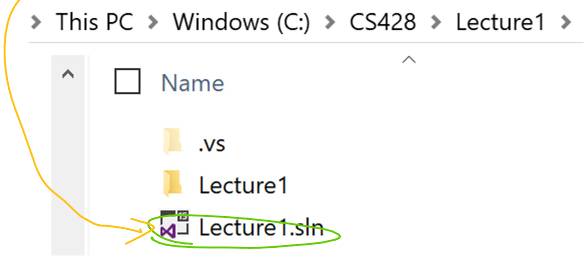
Evaluated value:

C:\CS428\Lecture1\/../External Resources/GLEW/include
%(AdditionalIncludeDirectories)

Inherited values:

☑ Inherit from parent or project defaults

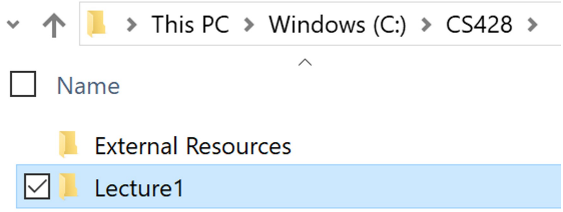
"$(SolutionDir)" is where your project file is:



› This PC › Windows (C:) › CS428 › Lecture1 ›

□ Name
.vs
Lecture1
Lecture1.sln

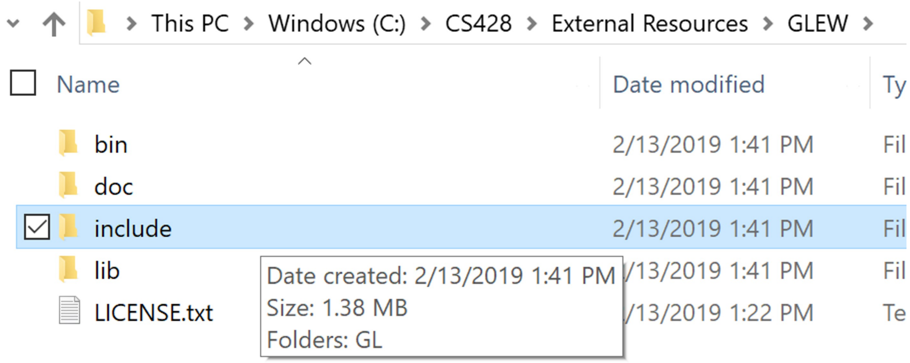
"../" is go back one directory.



› This PC › Windows (C:) › CS428 ›

□ Name
External Resources
☑ Lecture1

"/External Resources/GLEW/include" is what we are linking.

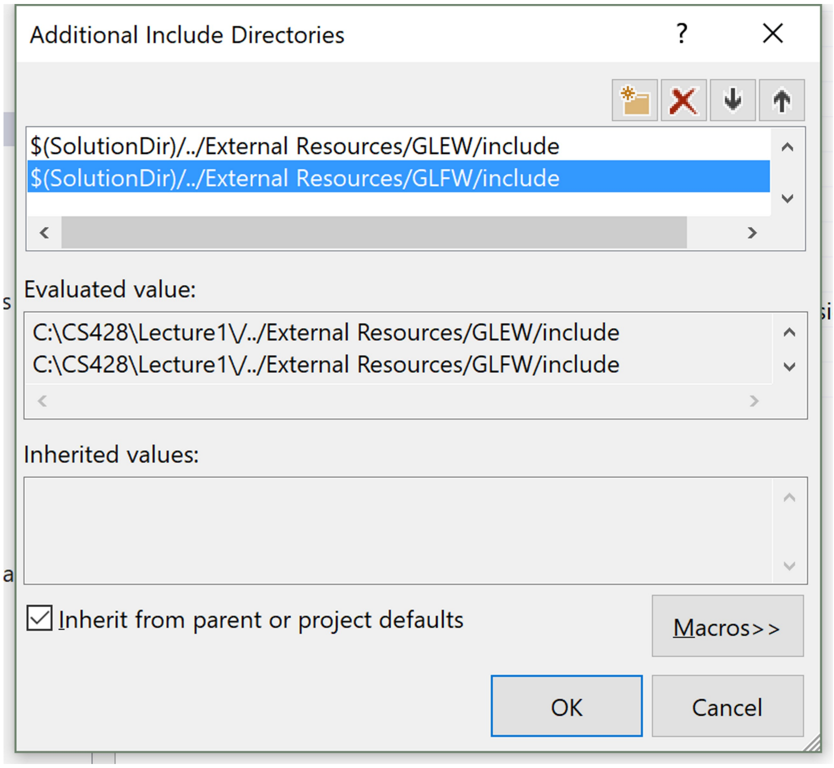


› This PC › Windows (C:) › CS428 › External Resources › GLEW ›

□ Name	Date modified	Ty
bin	2/13/2019 1:41 PM	Fil
doc	2/13/2019 1:41 PM	Fil
☑ include	2/13/2019 1:41 PM	Fil
lib	/13/2019 1:41 PM	Fil
LICENSE.txt	/13/2019 1:22 PM	Te

Date created: 2/13/2019 1:41 PM
Size: 1.38 MB
Folders: GL

e. Same thing goes to the GLFW
You can copy and paste and just change "E" with "F", if you used the naming strategy mentioned in step 4.e.


Additional Include Directories
?
Inherited values:
☑ Inherit from parent or project defaults
Macros>>
OK
Cancel

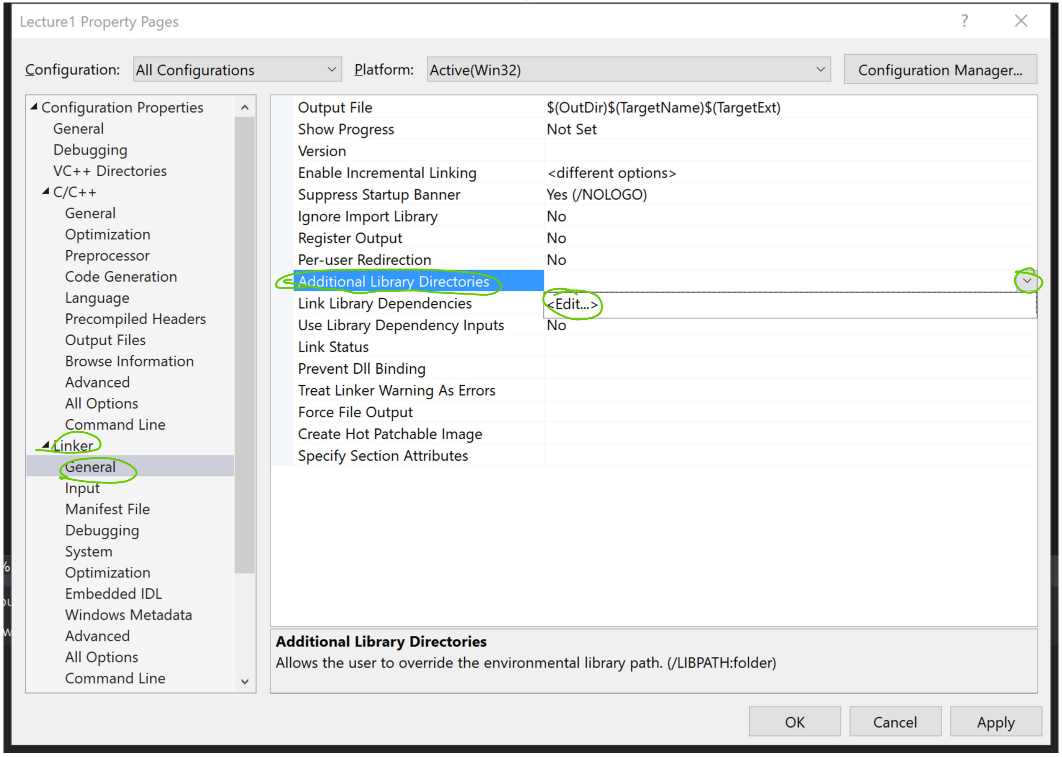
$(SolutionDir)/../External Resources/GLEW/include
$(SolutionDir)/../External Resources/GLFW/include

Evaluated value:

C:\CS428\Lecture1\/../External Resources/GLEW/include
C:\CS428\Lecture1\/../External Resources/GLFW/include

f. When done typing/copying, click OK.

8. Link libraries (part 2)
   a. Go to: Linker -> General -> Additional Library Directories -> Edit

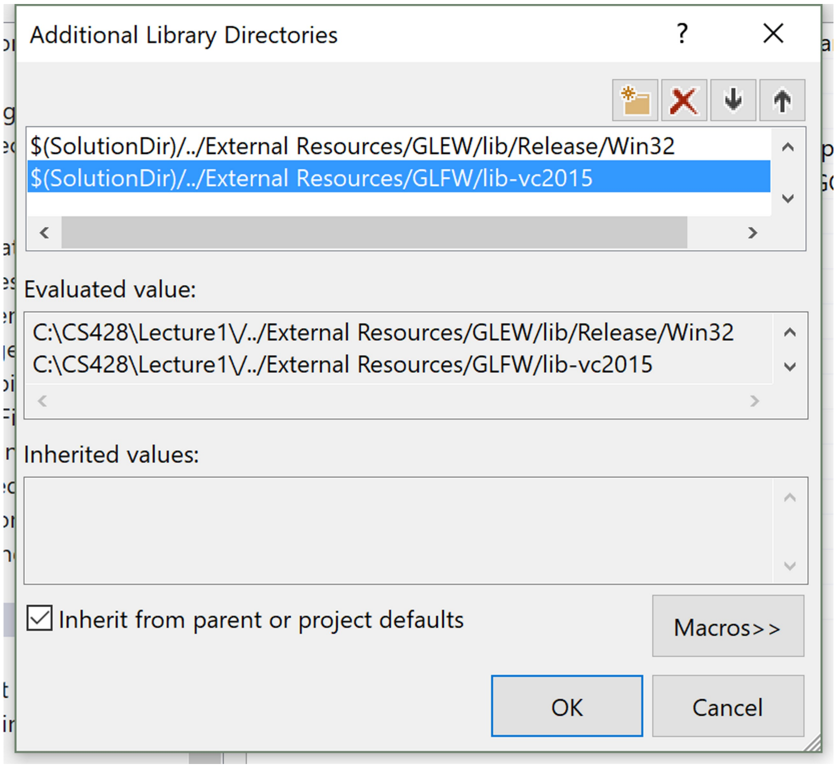


Configuration: All Configurations   Platform: Active(Win32)

Output File	$(OutDir)$(TargetName)$(TargetExt)
Show Progress	Not Set
Version	
Enable Incremental Linking	<different options>
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	
Link Library Dependencies	<Edit...>
Use Library Dependency Inputs	No
Link Status	
Prevent Dll Binding	
Treat Linker Warning As Errors	
Force File Output	
Create Hot Patchable Image	
Specify Section Attributes	

Additional Library Directories
Allows the user to override the environmental library path. (/LIBPATH:folder)

   b. Almost the same with the previous "$(SolutionDir)/" path, but a bit different: You can still copy and paste to save your time.



Additional Library Directories

```
$(SolutionDir)/../External Resources/GLEW/lib/Release/Win32
$(SolutionDir)/../External Resources/GLFW/lib-vc2015
```

Evaluated value:

```
C:\CS428\Lecture1\/../External Resources/GLEW/lib/Release/Win32
C:\CS428\Lecture1\/../External Resources/GLFW/lib-vc2015
```

Inherited values:

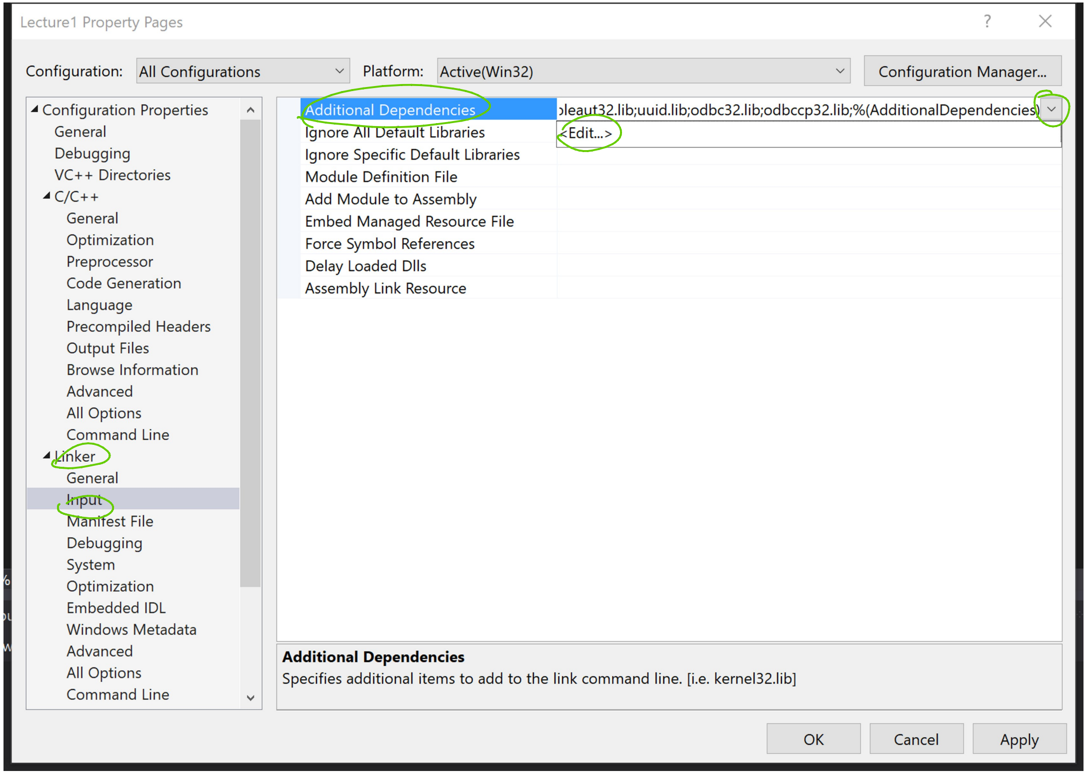
☑ Inherit from parent or project defaults

I am pretty sure you won't have the "lib-vc2017" inside GLFW folder, by now. But "lib-vc2017" works just fine.

e. Click OK.

9. Link libraries (part 3)

a. Go to: Linker -> Input -> Additional Dependencies -> Edit



Lecture1 Property Pages

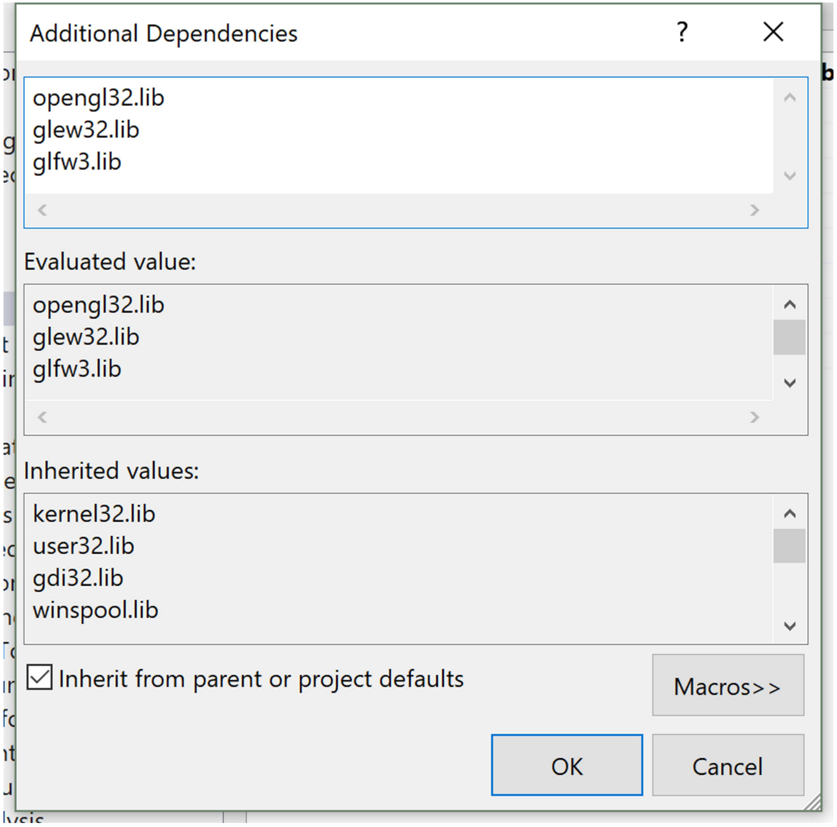
Configuration: All Configurations   Platform: Active(Win32)

Additional Dependencies	oleaut32.lib;uuid.lib;odbc32.lib;odbccp32.lib;%(AdditionalDependencies)
Ignore All Default Libraries	<Edit...>
Ignore Specific Default Libraries	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded Dlls	
Assembly Link Resource	

Additional Dependencies
Specifies additional items to add to the link command line. [i.e. kernel32.lib]

b. Type in:
opengl32.lib

glew32.lib
glfw3.lib



Additional Dependencies ? ✕

opengl32.lib
glew32.lib
glfw3.lib

Evaluated value:

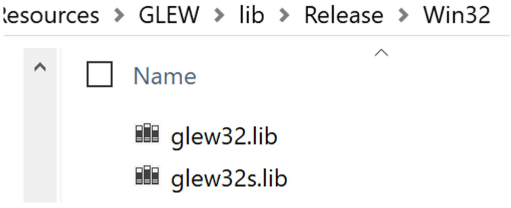
opengl32.lib
glew32.lib
glfw3.lib

Inherited values:

kernel32.lib
user32.lib
gdi32.lib
winspool.lib

☑ Inherit from parent or project defaults

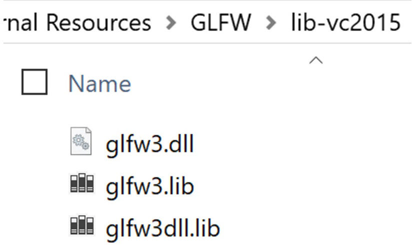
"opengl32.lib" is included in the Windows 10.
For "glew32.lib" and "glfw3.lib" spellings, you can check files in "\GLEW\lib\Release\Win32" and "\GLFW\lib-vc2015" to verify.



esources › GLEW › lib › Release › Win32

□ Name

glew32.lib
glew32s.lib



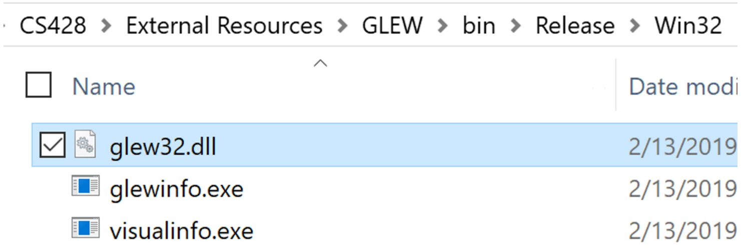
nal Resources › GLFW › lib-vc2015

□ Name

glfw3.dll
glfw3.lib
glfw3dll.lib

c. Click OK.
d. Click Apply.
e. Close Property window.

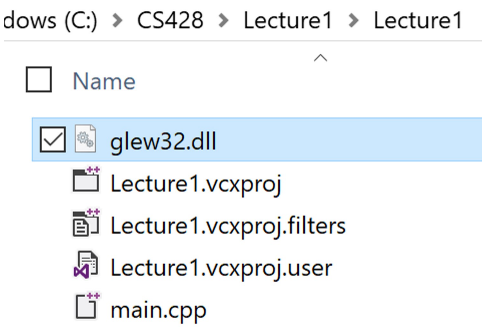
10. Link libraries (Part 4)
    a. Go to your folder and copy the "glew32.dll"



CS428 > External Resources > GLEW > bin > Release > Win32

☐ Name	Date modi
☑ glew32.dll	2/13/2019
glewinfo.exe	2/13/2019
visualinfo.exe	2/13/2019

b. Go to your project folder, where your main.cpp file is. Paste the glew32.dll file to that location.



dows (C:) > CS428 > Lecture1 > Lecture1

☐ Name

☑ glew32.dll

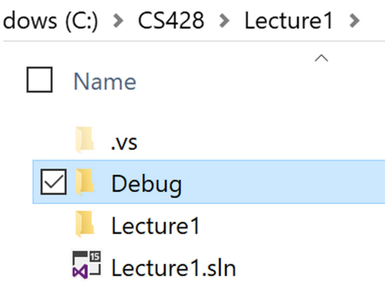
Lecture1.vcxproj

Lecture1.vcxproj.filters

Lecture1.vcxproj.user

main.cpp

c. Go to where your solution file (.sln file) is, create a folder called "Debug"



dows (C:) > CS428 > Lecture1 >

☐ Name

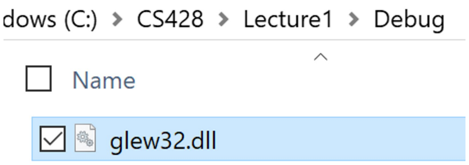
.vs

☑ Debug

Lecture1

Lecture1.sln

Paste the .dll file inside folder "Debug"

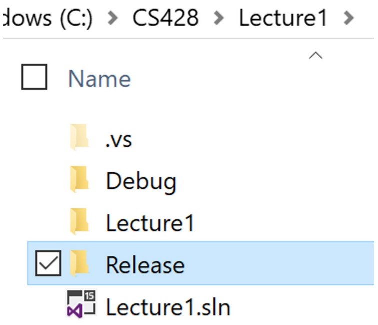


dows (C:) > CS428 > Lecture1 > Debug

☐ Name

☑ glew32.dll

d. Go back the where your solution file (.sln file) is, create a folder called "Release". Paste the .dll file in there as well.



dows (C:) > CS428 > Lecture1 >

☐ Name

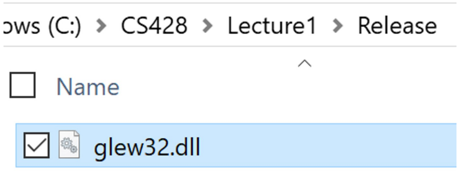
.vs

Debug

Lecture1

☑ Release

Lecture1.sln



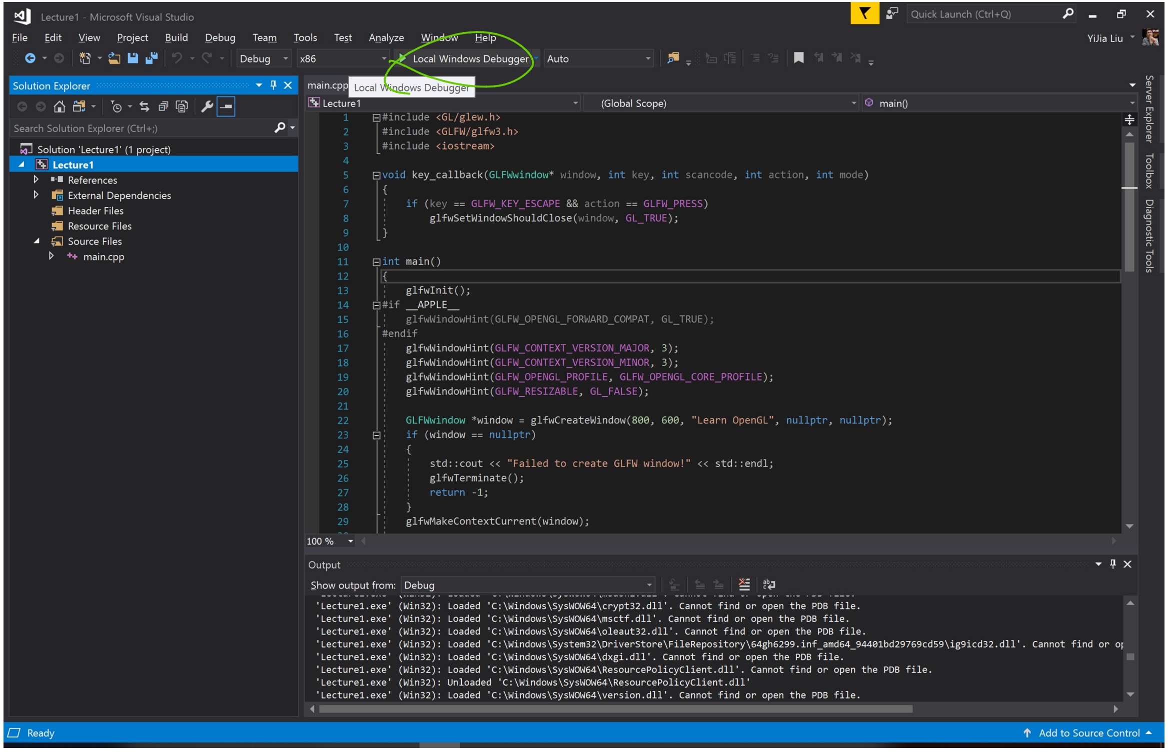
ows (C:) > CS428 > Lecture1 > Release

☐ Name

☑ glew32.dll

11. We are all set!

12. Copy code into main.cpp and test the result. Click green arrow to run.

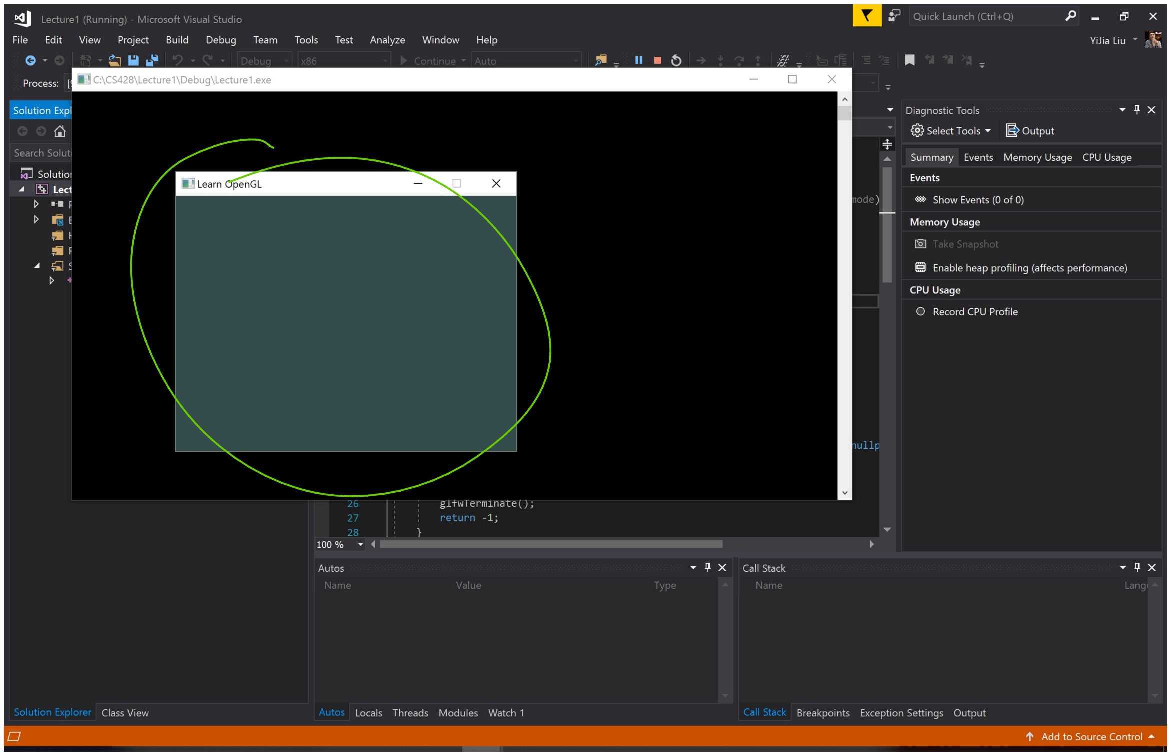


```cpp
#include <GL/glew.h>
#include <GLFW/glfw3.h>
#include <iostream>

void key_callback(GLFWwindow* window, int key, int scancode, int action, int mode)
{
    if (key == GLFW_KEY_ESCAPE && action == GLFW_PRESS)
        glfwSetWindowShouldClose(window, GL_TRUE);
}

int main()
{
    glfwInit();
#if __APPLE__
    glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
#endif
    glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
    glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
    glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
    glfwWindowHint(GLFW_RESIZABLE, GL_FALSE);

    GLFWwindow *window = glfwCreateWindow(800, 600, "Learn OpenGL", nullptr, nullptr);
    if (window == nullptr)
    {
        std::cout << "Failed to create GLFW window!" << std::endl;
        glfwTerminate();
        return -1;
    }
    glfwMakeContextCurrent(window);
```

Output

Show output from: Debug

```
'Lecture1.exe' (Win32): Loaded 'C:\Windows\SysWOW64\crypt32.dll'. Cannot find or open the PDB file.
'Lecture1.exe' (Win32): Loaded 'C:\Windows\SysWOW64\msctf.dll'. Cannot find or open the PDB file.
'Lecture1.exe' (Win32): Loaded 'C:\Windows\SysWOW64\oleaut32.dll'. Cannot find or open the PDB file.
'Lecture1.exe' (Win32): Loaded 'C:\Windows\System32\DriverStore\FileRepository\64gh6299.inf_amd64_94401bd29769cd59\ig9icd32.dll'. Cannot find or op
'Lecture1.exe' (Win32): Loaded 'C:\Windows\SysWOW64\dxgi.dll'. Cannot find or open the PDB file.
'Lecture1.exe' (Win32): Loaded 'C:\Windows\SysWOW64\ResourcePolicyClient.dll'. Cannot find or open the PDB file.
'Lecture1.exe' (Win32): Unloaded 'C:\Windows\SysWOW64\ResourcePolicyClient.dll'
'Lecture1.exe' (Win32): Loaded 'C:\Windows\SysWOW64\version.dll'. Cannot find or open the PDB file.
```


Lecture1 (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Continue
C:\CS428\Lecture1\Debug\Lecture1.exe
Learn OpenGL
Diagnostic Tools
Select Tools
Output
Summary Events Memory Usage CPU Usage
Events
Show Events (0 of 0)
Memory Usage
Take Snapshot
Enable heap profiling (affects performance)
CPU Usage
Record CPU Profile
Autos
Name Value Type
Call Stack
Name Lang
Solution Explorer Class View
Autos Locals Threads Modules Watch 1
Call Stack Breakpoints Exception Settings Output
Add to Source Control

Success!