

# Lab01 for CPT 205 Computer Graphics

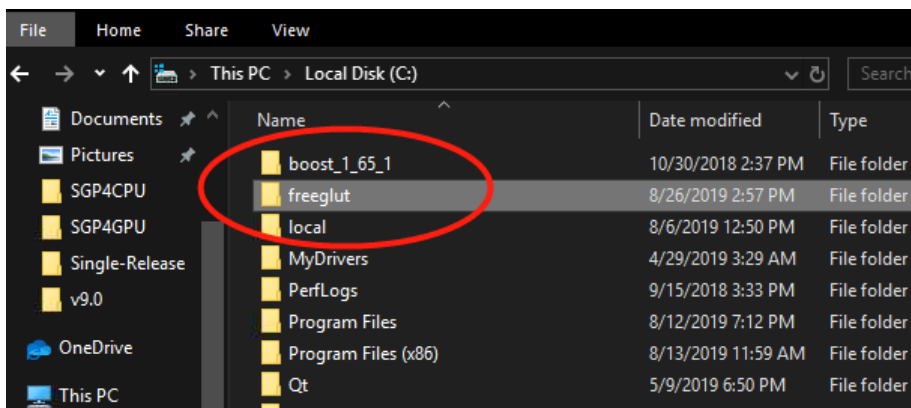
## 1) Introduction

This is the first of a series of practical sheets that are designed to help you understand how to create 2D and 3D graphic images using C++ and OpenGL graphics library. We will be using the Microsoft Visual Studio 2019 as IDE to enter computer programs, save them, compile them, link them and run them.

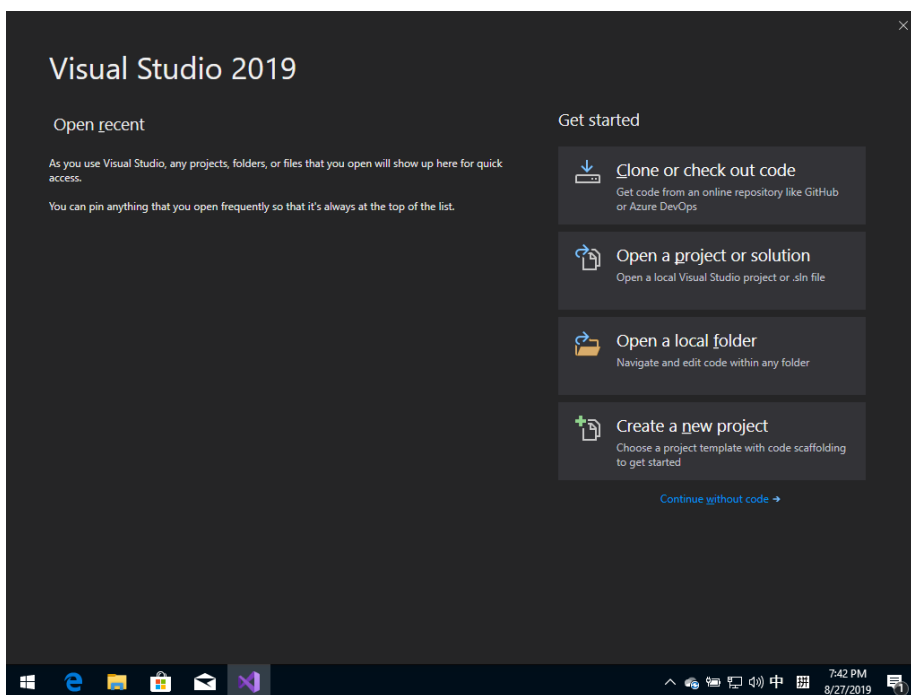
## 2) Getting started with Microsoft Visual Studio 2019 and OpenGL graphics library

The following steps get you started with your very first OpenGL program using Microsoft Visual Studio 2019.

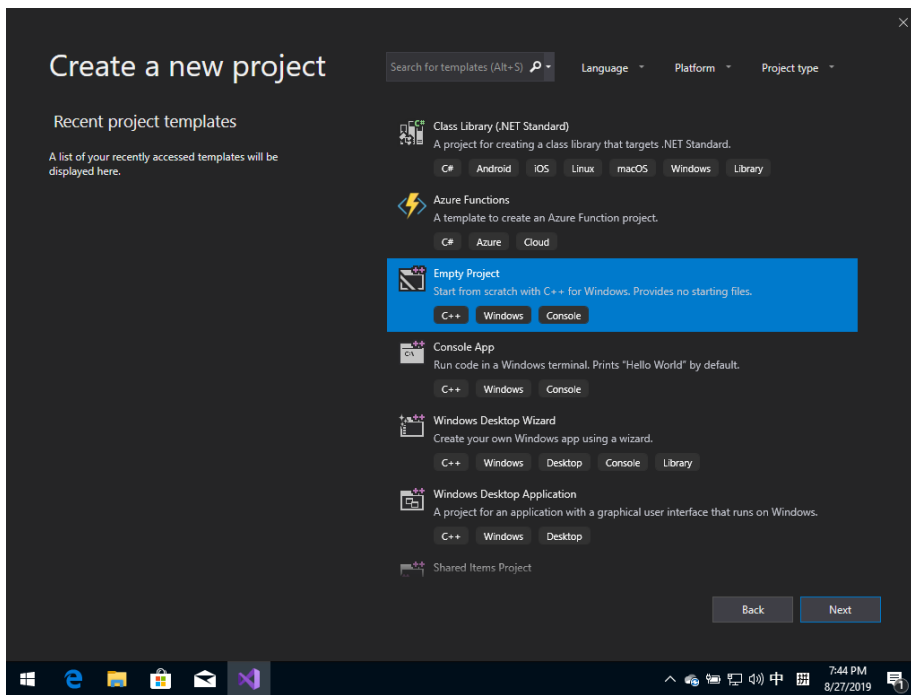
1. Uncompress the file named 'freeglut\_win10\_VS2019.zip' to your local disk C, D and so on. The freeglut graphics library is for Windows 10 x64 and Visual Studio 2019.



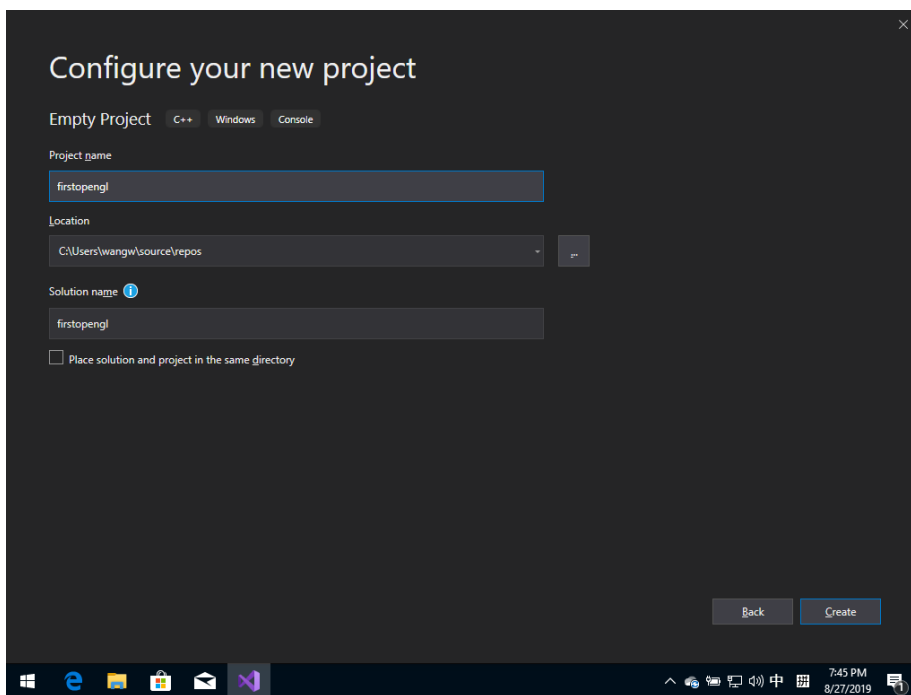
2. Open Microsoft Visual Studio 2019, and you will see the following window (note that this may take a couple of minutes).



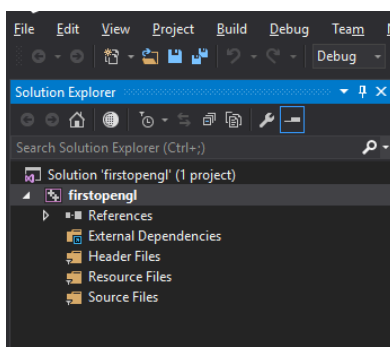
3. Click 'Create a new project', and you will see the following window. Choose 'Empty Project' and click 'Next'.



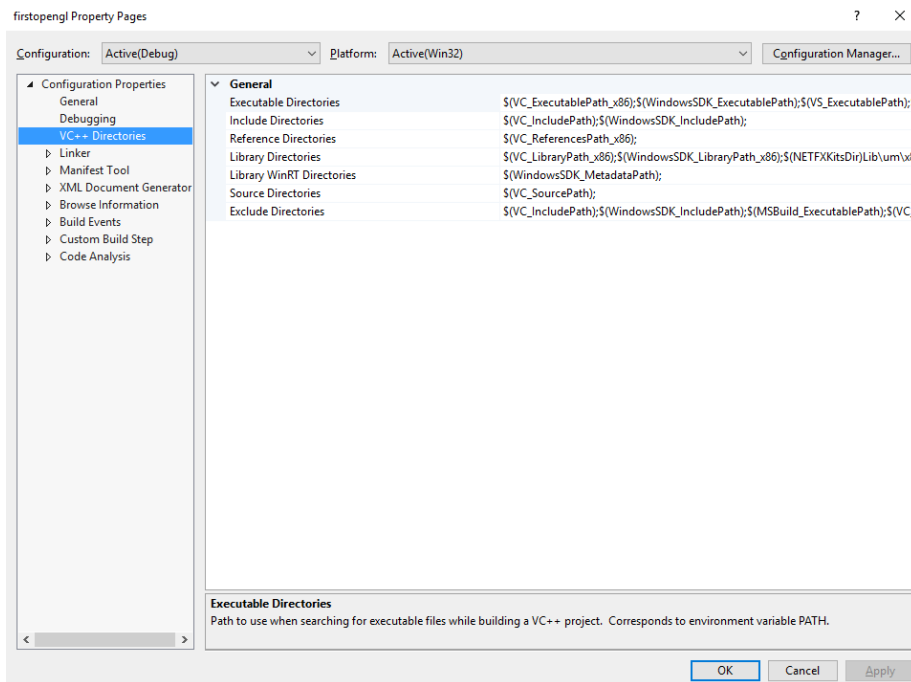
4. Enter a name for the project such as that in the following picture, 'firstopengl'.



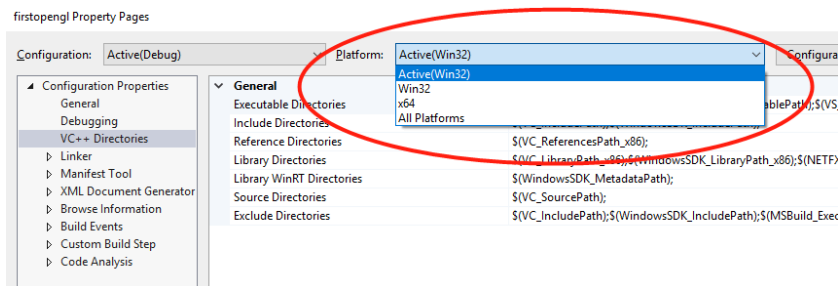
5. Click 'Finish'.



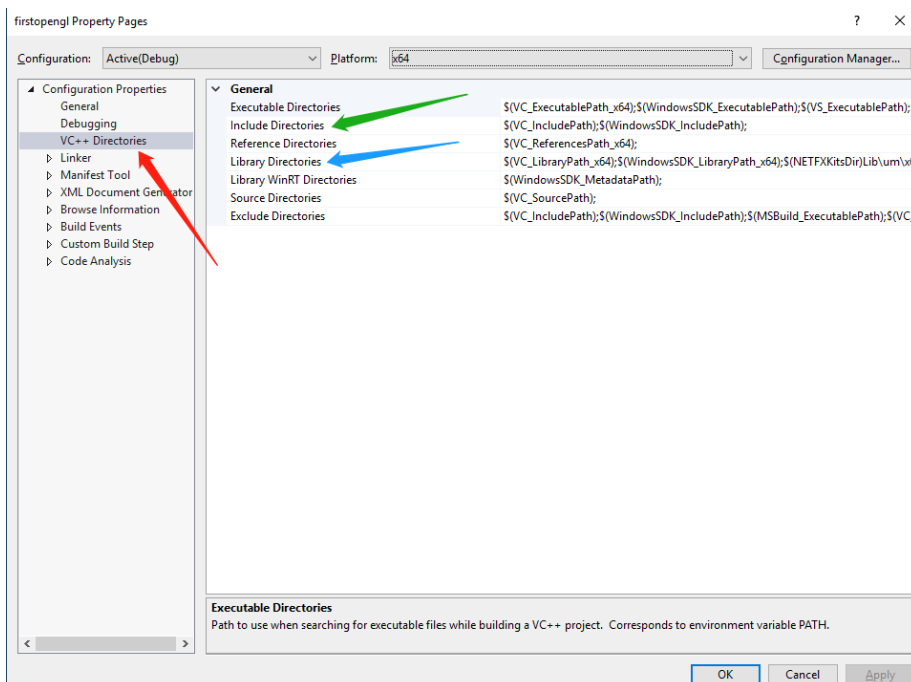
6. Now right click on the project name, choose properties (in the bottom of the list). You will see the following.



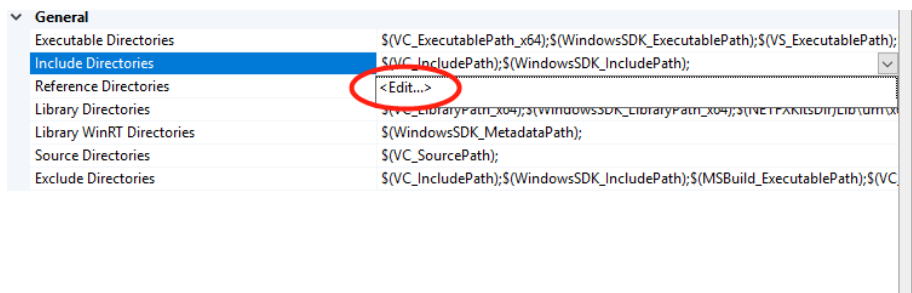
7. Click the symbol 'v' beside the 'Platform', choose 'x64' in the list of Platform.



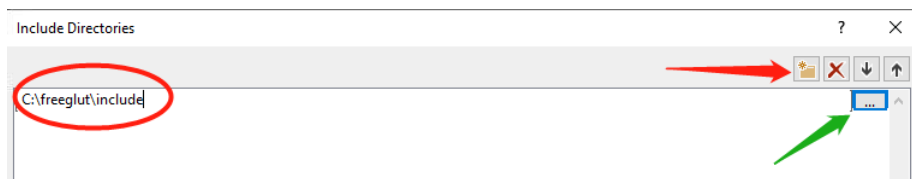
8. Click 'VC++ Directories' on the left side (as shown with the red arrow), and you will see the following picture.



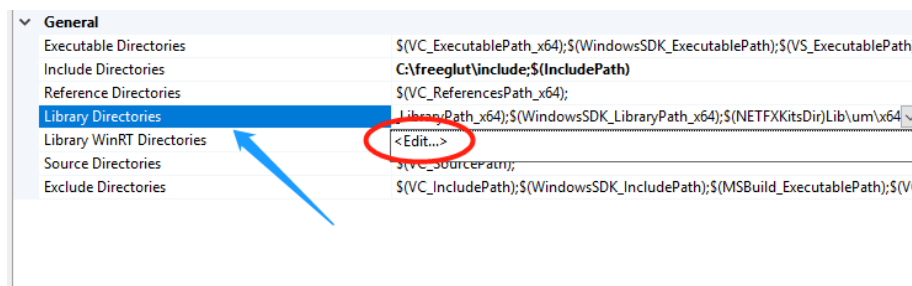
9. Click the symbol 'v' which is on the right side of 'Include Directories', and choose the '<Edit...>' in the list.



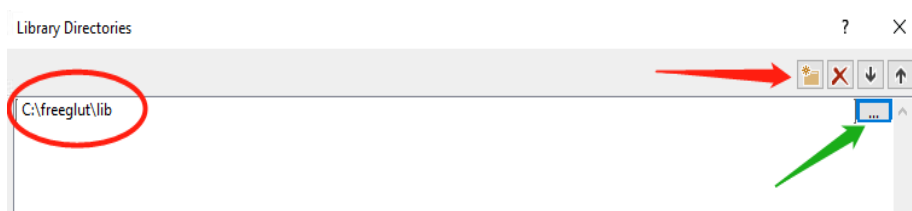
10. Click the 'New Line' button (red arrow), click the '...' button (green arrow), and choose the 'include' folder where you have uncompressed the zip file. Click the 'OK' button, to go back to the previous page.



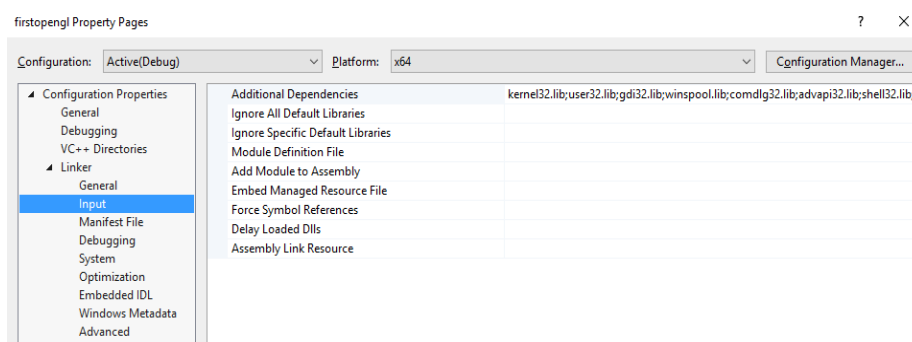
11. Click symbol 'v' on the right side of 'Library Directories' (blue arrow), and choose '<Edit...>' in the list.



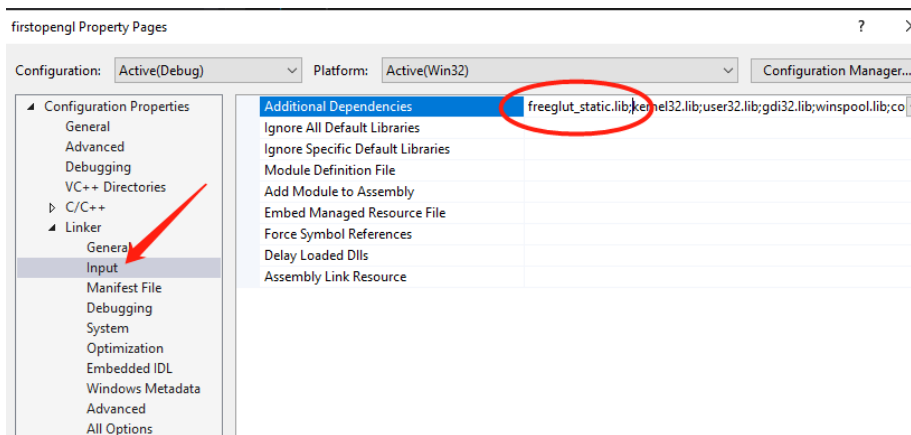
12. Click the 'New Line' button (red arrow), click the '...' button (green arrow), and choose the 'lib' folder where you have uncompressed the zip file. Click the 'OK' button, to go back to the previous page.



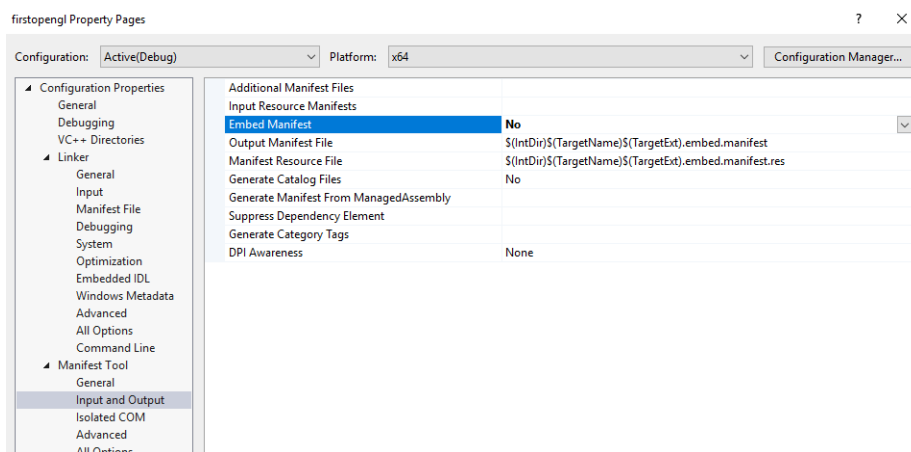
13. Click the symbol '>' which is on the left side of 'Linker', and choose 'Input' in the list of 'Linker'; you will see the following picture.



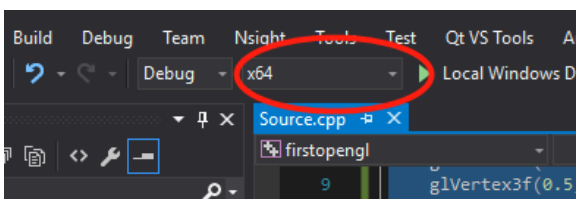
14. Add the words "freeglut\_static.lib;" into 'Additional Dependencies' which is on the top of the right part of the current window. A semicolon ';' is used to separate the libraries.



15. Click the '>' symbol which is on the left side of 'Manifest Tool', and choose 'Input and Output' in the list of 'Manifest Tool'; Set 'Embed Manifest' to "No"; Click 'OK'.



16. Choose 'x64' in the Platform list



17. Now you are ready write and edit your program. We have a sample program on the next page. Right click 'Source Files' and add a 'cpp' file. Copy the following code into the cpp file. Press 'F7' on your keyboard to build the source code.

```

#define FREEGLUT_STATIC // Define a static library for calling functions
#include <GL/freeglut.h> // Include the header file

void renderScene(void) // Function for geometric creation
{
    // Clear the buffer to the predefined color and depth value
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    // Move origin of current user coordinate system to the screen center: similar to a reset operation
    glLoadIdentity();
    glBegin(GL_TRIANGLES); // Start draw TRIANGLE function
    glVertex3f(-0.5, -0.5, 0.0); // Set coordinates of first vertex
    glVertex3f(0.5, 0.0, 0.0); // Set coordinates of second vertex
    glVertex3f(0.0, 0.5, 0.0); // Set coordinates of last vertex
    glEnd(); // End draw TRIANGLE function
    glutSwapBuffers(); // Refresh the screen to display the graphics
}

int main(int argc, char* argv[]) // Standard main function
{
    glutInit(&argc, (char**)argv); // Initialization
    // Define display mode: depth buffer, double buffer and RGBA color
    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
    glutInitWindowPosition(100, 100); // Define the location of the window
    glutInitWindowSize(320, 320); // Define the size of the window
    glutCreateWindow("Hello OpenGL"); // Create a window called "Hello OpenGL"
    glutDisplayFunc(renderScene); // Set the first function
    glutMainLoop(); // Enter the GLUT event processing loop
    return 0; // Return an integer value
}

```

18. Press 'F5' on your keyboard to run the program now. If you have done it correctly, a window with a triangle will come out.

