OpenGL Startup

Computer Graphics

Yu-Ting Wu

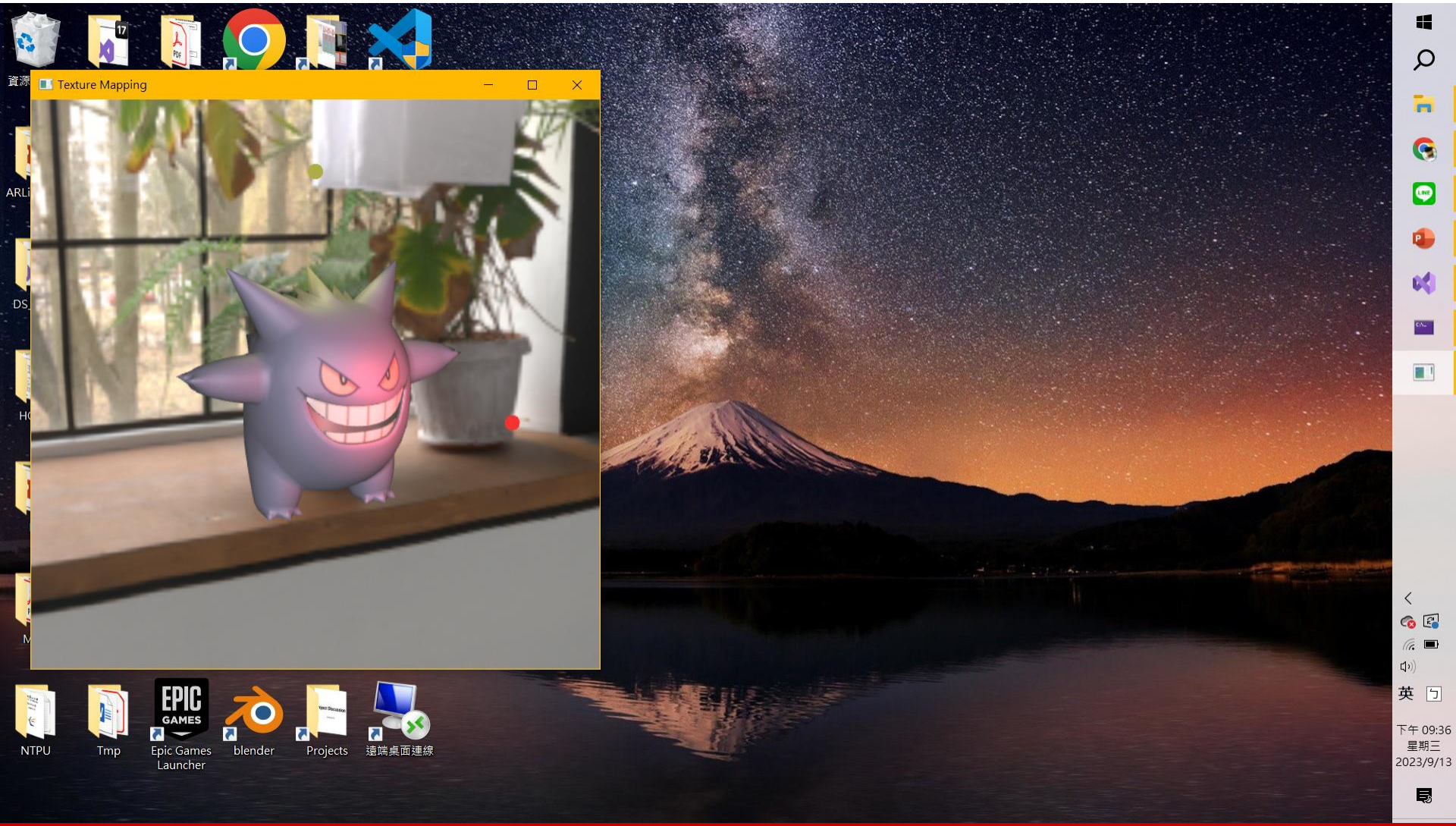
Outline

- Environment setup
- The first OpenGL program
- Appendix: build your own FreeGLUT libraries

Outline

- **Environment setup**
- The first OpenGL program
- Appendix: build your own FreeGLUT libraries

An OpenGL Program



Library for Handling Screen Rendering

- **GLUT: OpenGL Utility Toolkit (link)**
  - Window system independent
  - Implement a simple window application programming interface (API) for OpenGL
  - Designed for constructing small to medium-sized OpenGL programs
    - For large applications, it is suggested to use a native window system toolkit such as Qt for more sophisticated UI
- **FreeGLUT: Free OpenGL Utility Toolkit (link)**
  - GLUT has gone into stagnation and has some issues with licenses
  - FreeGLUT is intended to be a full replacement for GLUT

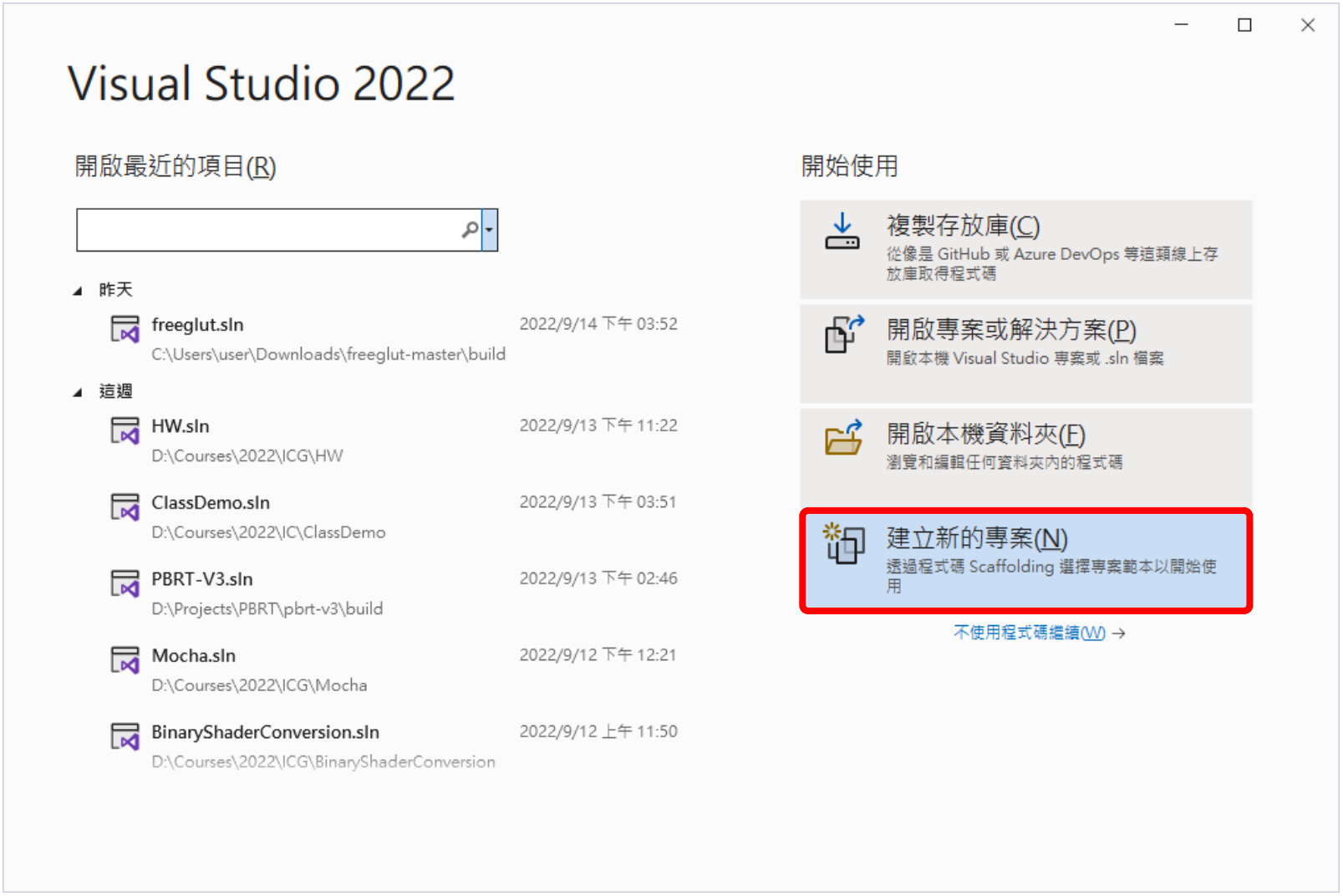
Prepare for FreeGLUT libraries

- Use the files in my sample projects

- Download the binaries from the Internet
  - https://www.transmissionzero.co.uk/software/freeglut-devel/
  - Older version (3.0.0)
  - Not support debug mode

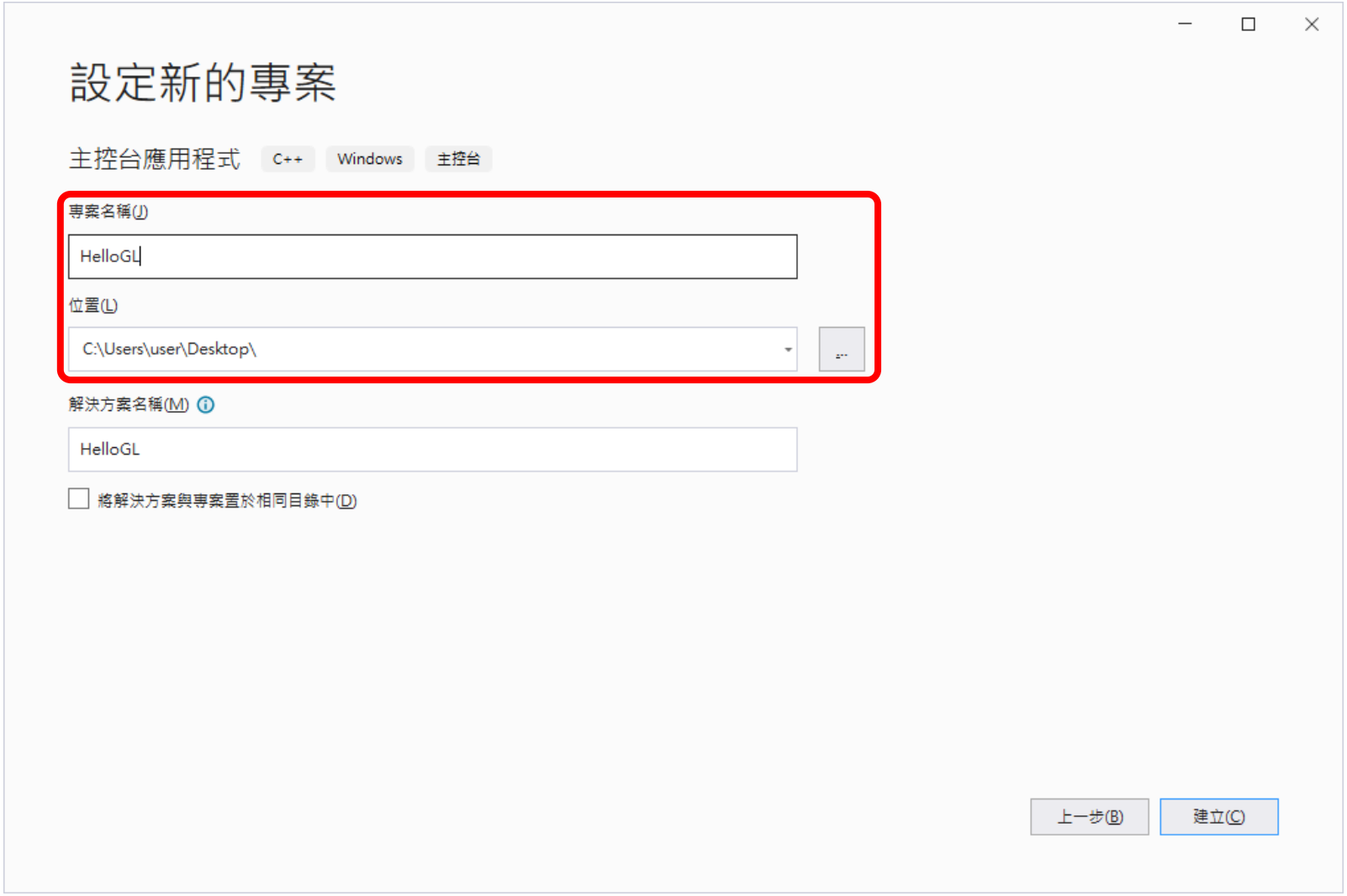
- Build it from scratch with **CMake** by yourself
  - Follow the instructions in the **Appendix section** in this slides

Create a New Project in VS

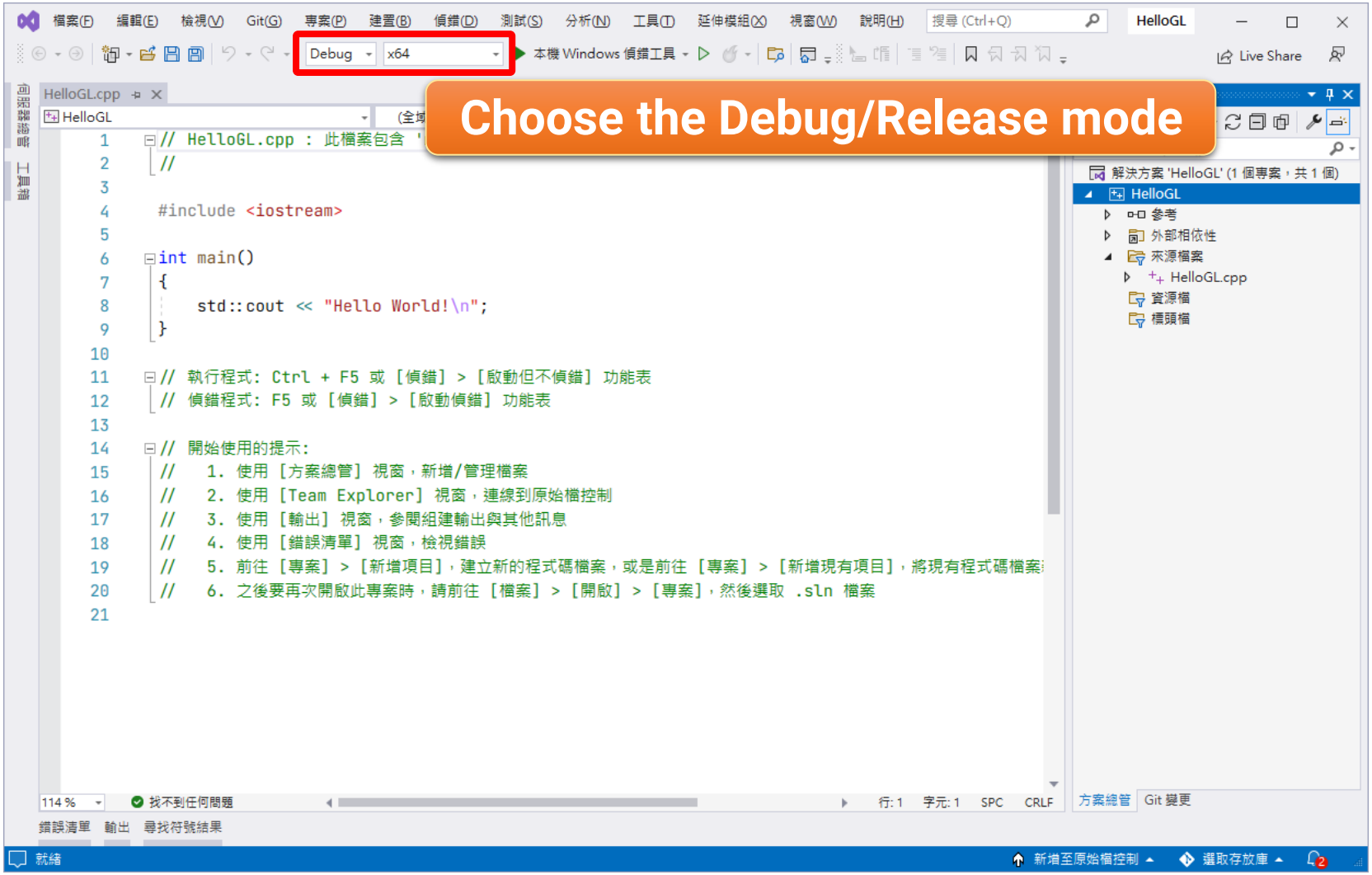


Create a New Project in VS (cont.)

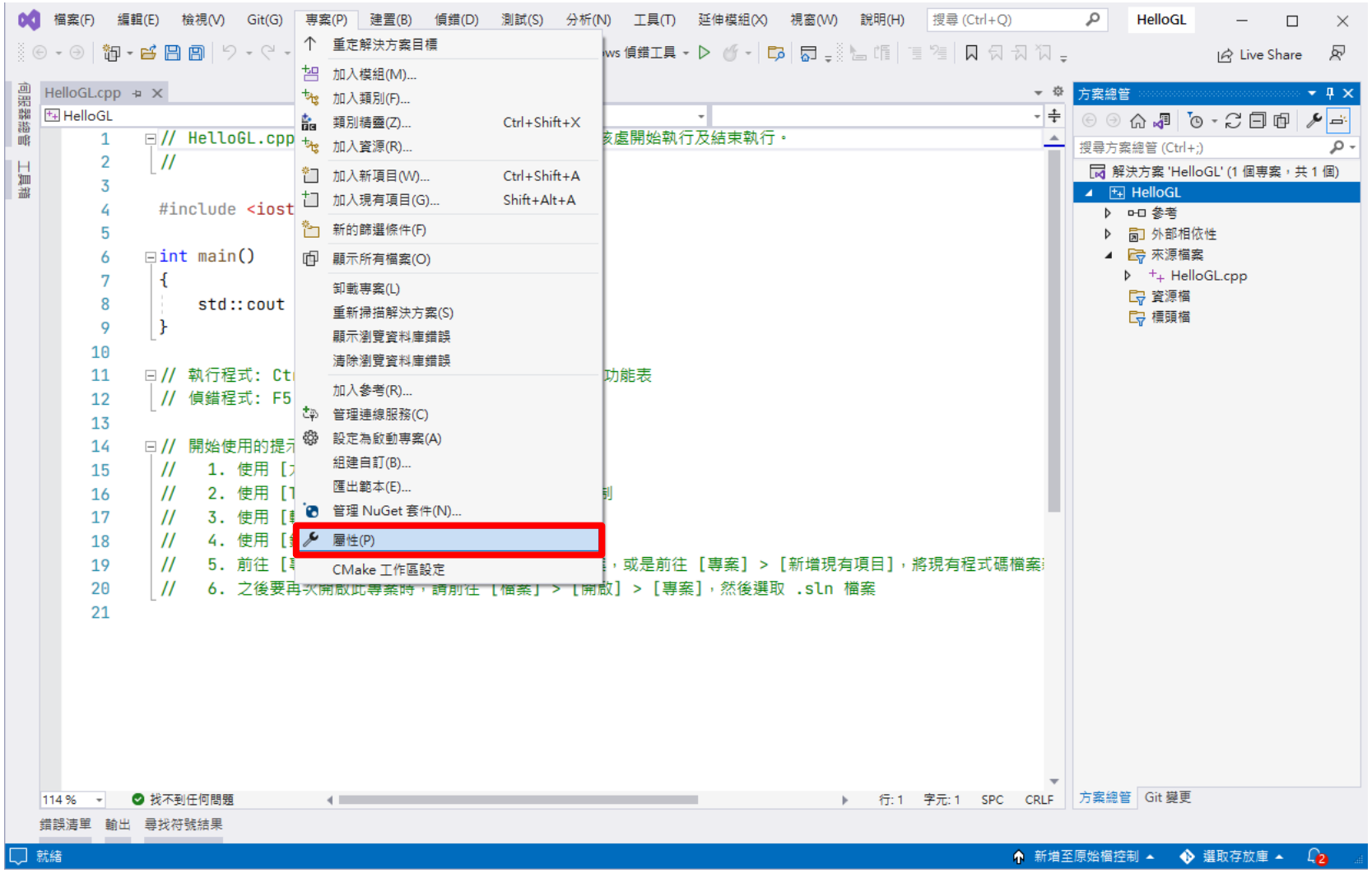
Create a New Project in VS (cont.)



Setup the Project in VS



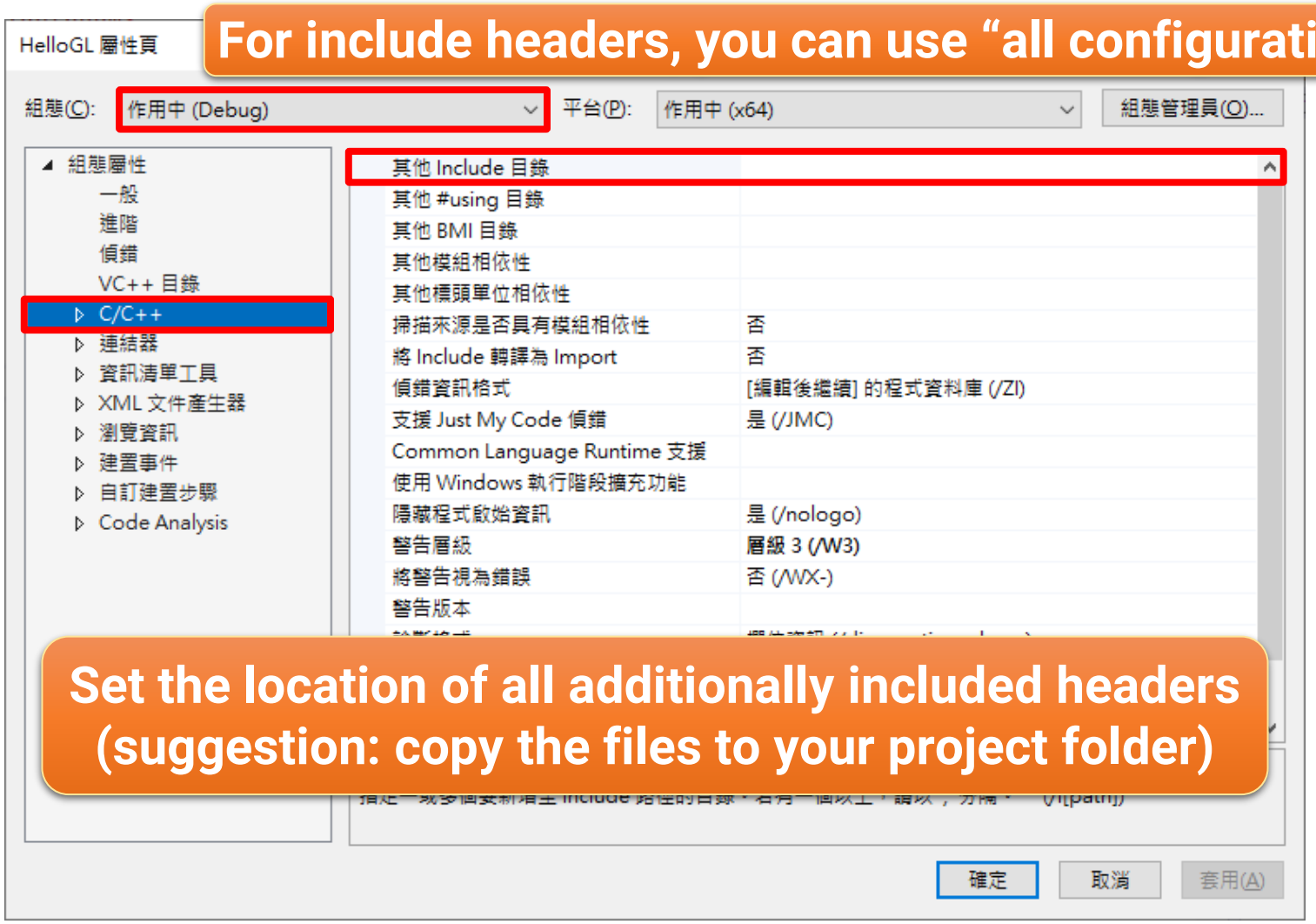
Setup the Project in VS



Setup the Project in VS (cont.)

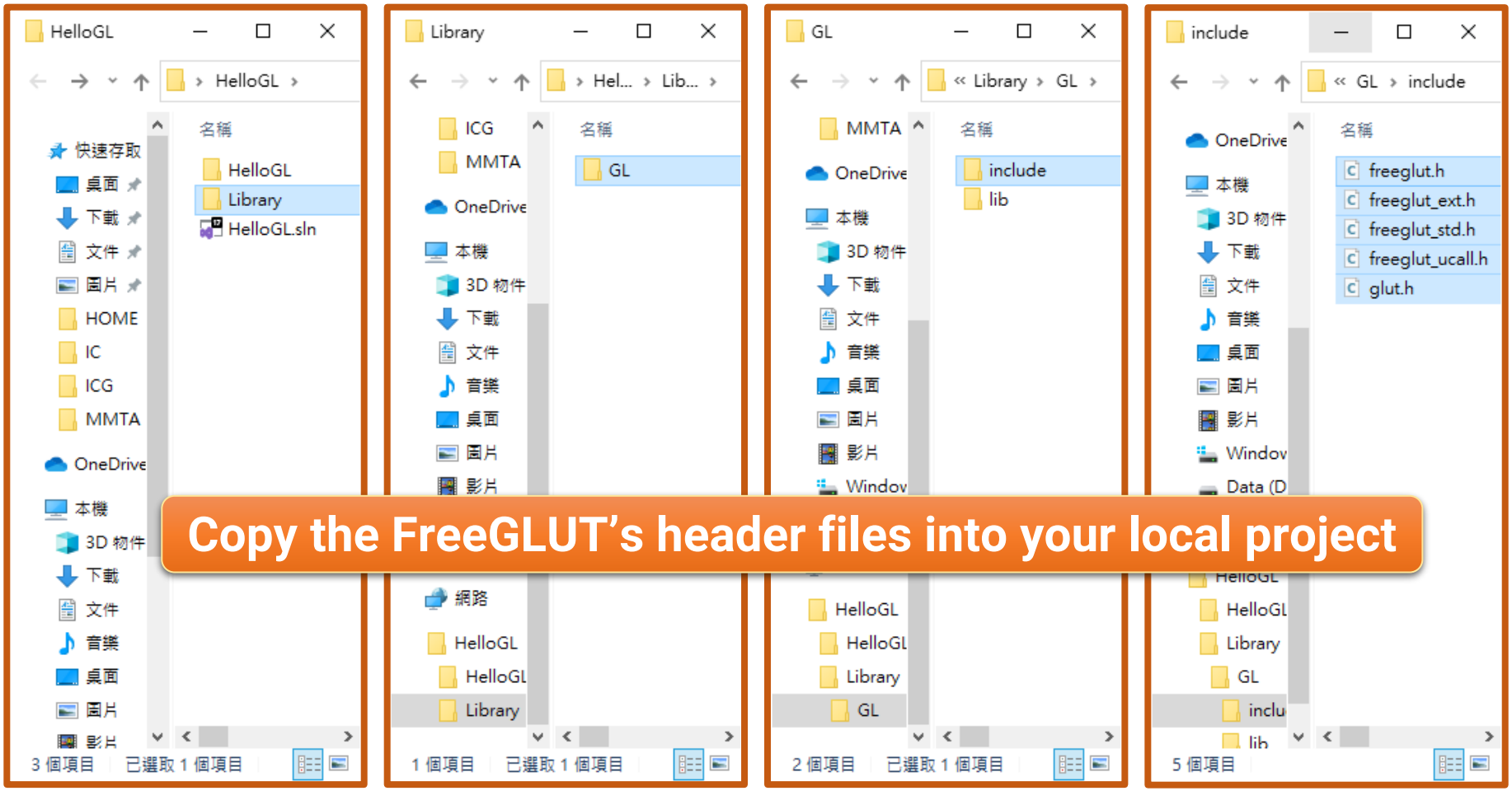
- A library usually consists of three types of files
  - **Header** (***.h**, need to include)
  - **Static library** (***.lib**, static linking, need to import during compile time)
  - **Dynamic library** (***.dll**, dynamic linking, need to access during the run time)

Setup the Headers in VS

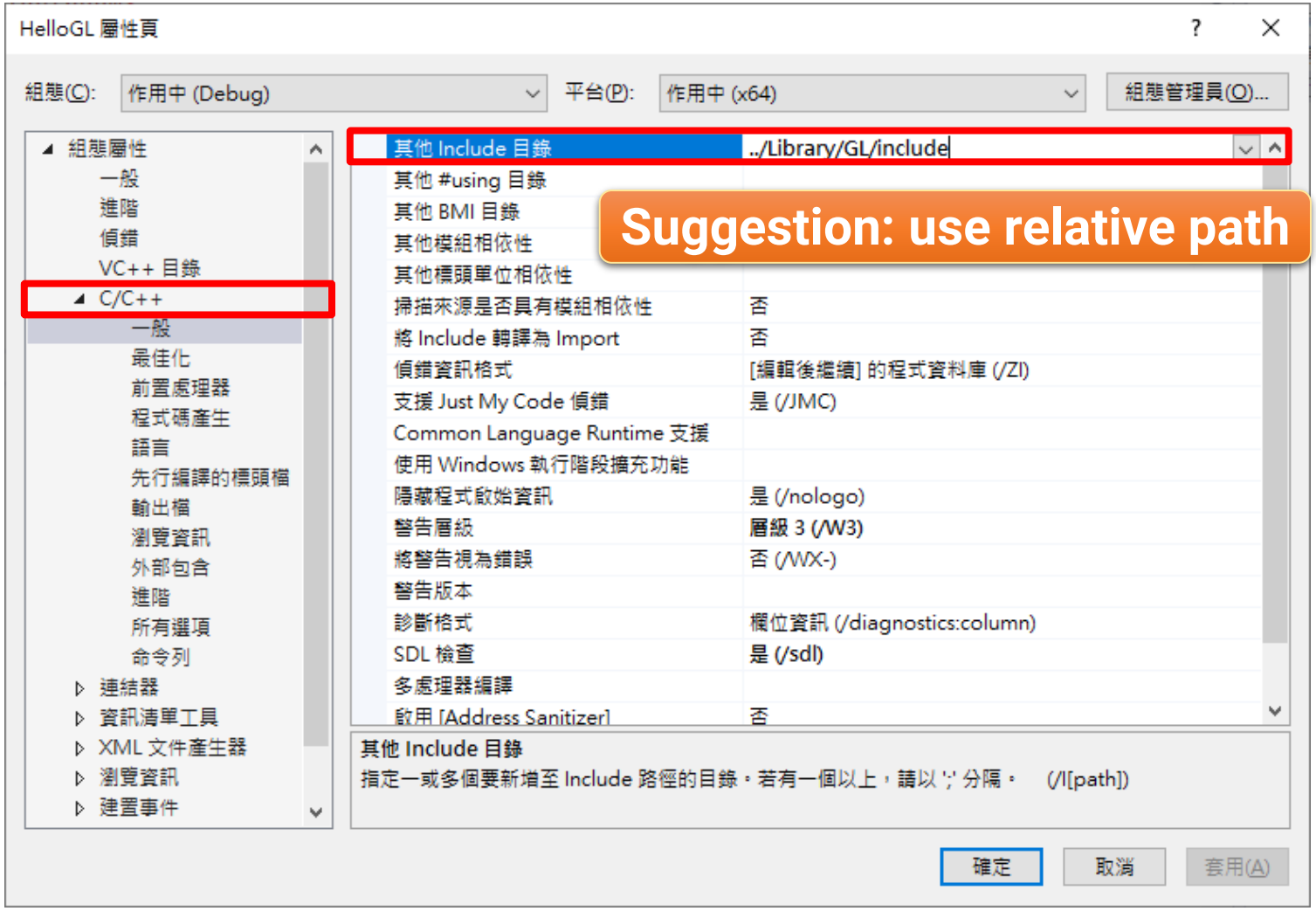


Setup the Headers in VS (cont.)

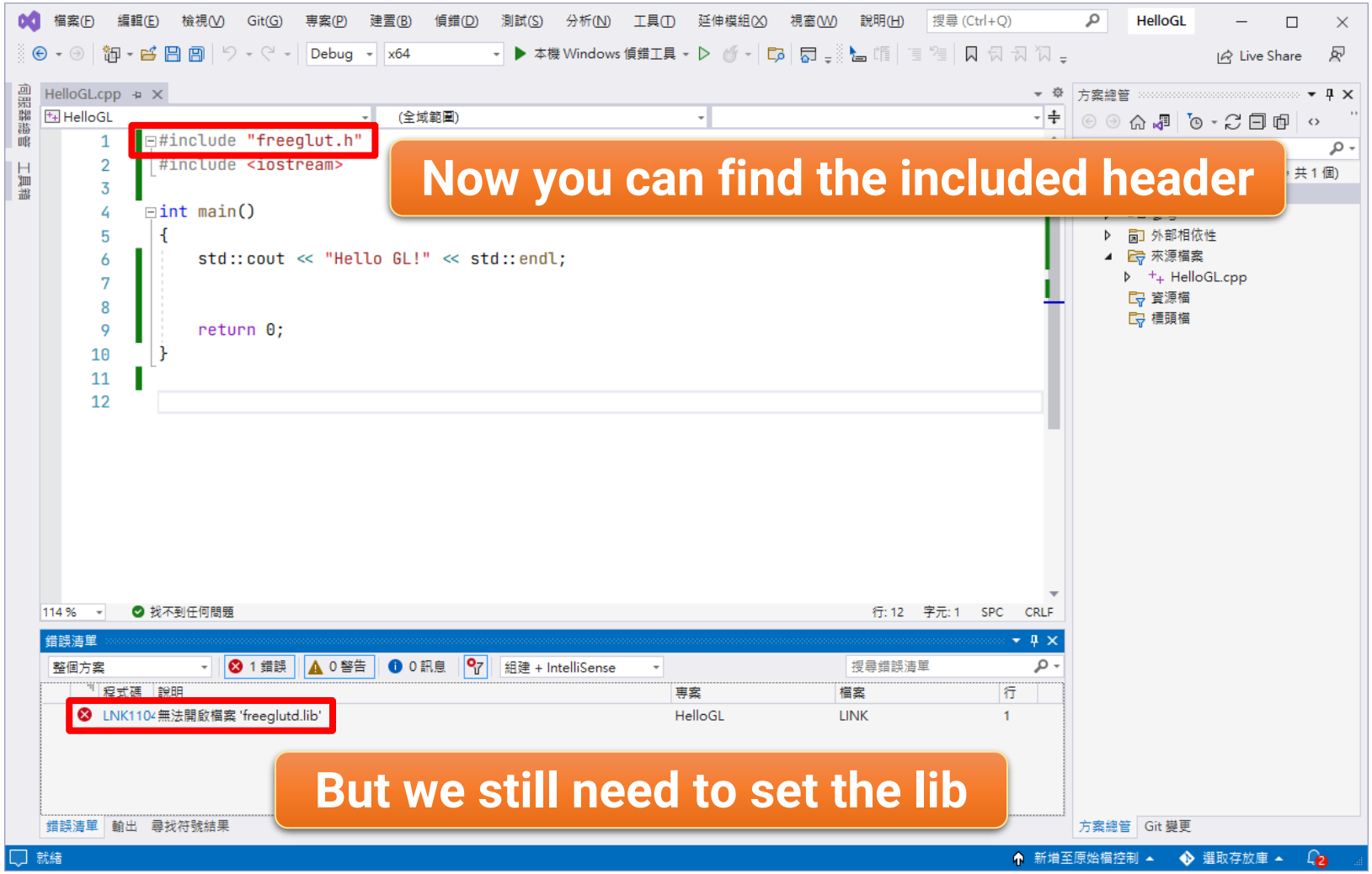
- My setting



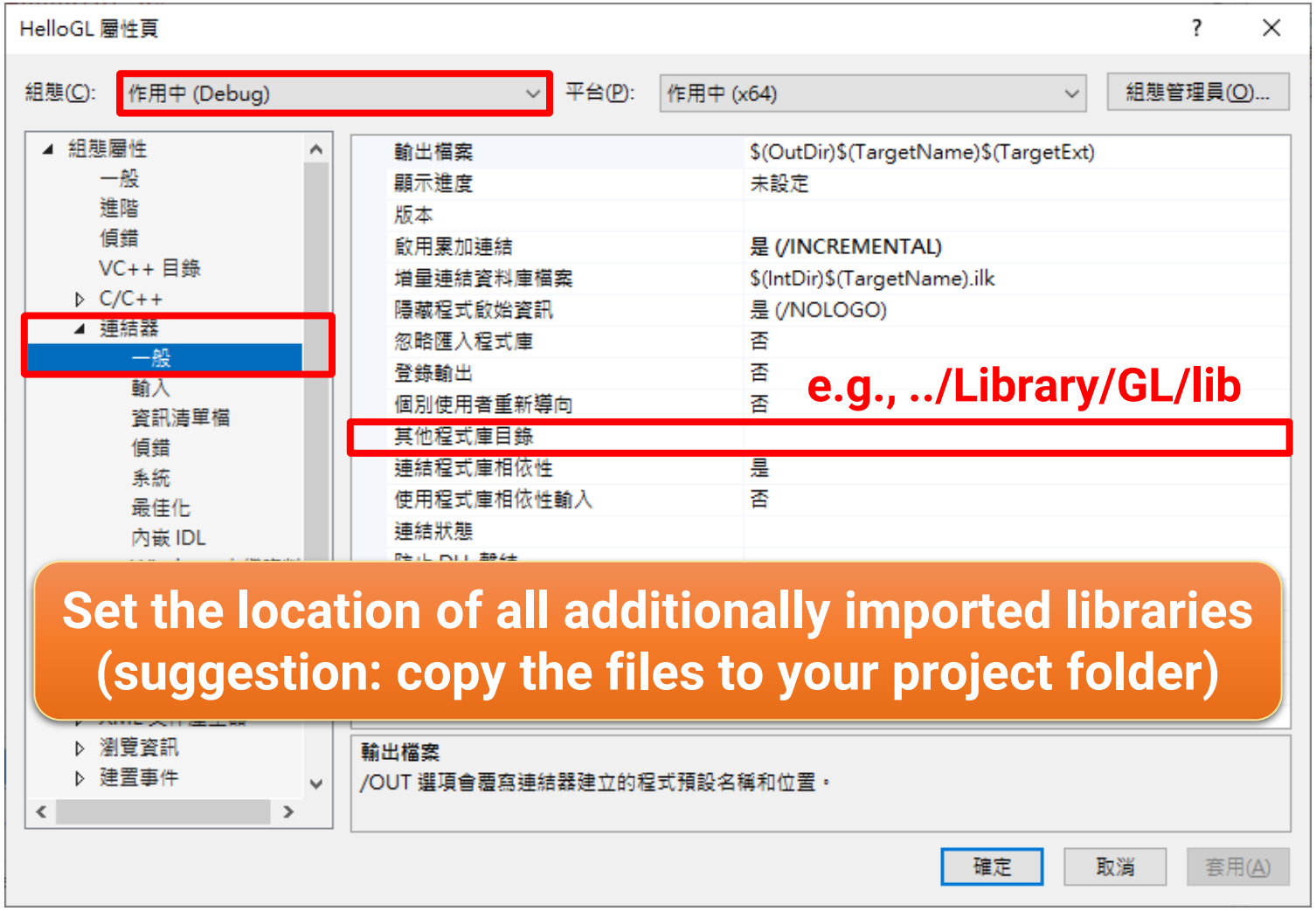
Setup the Headers in VS (cont.)



Setup the Headers in VS (cont.)

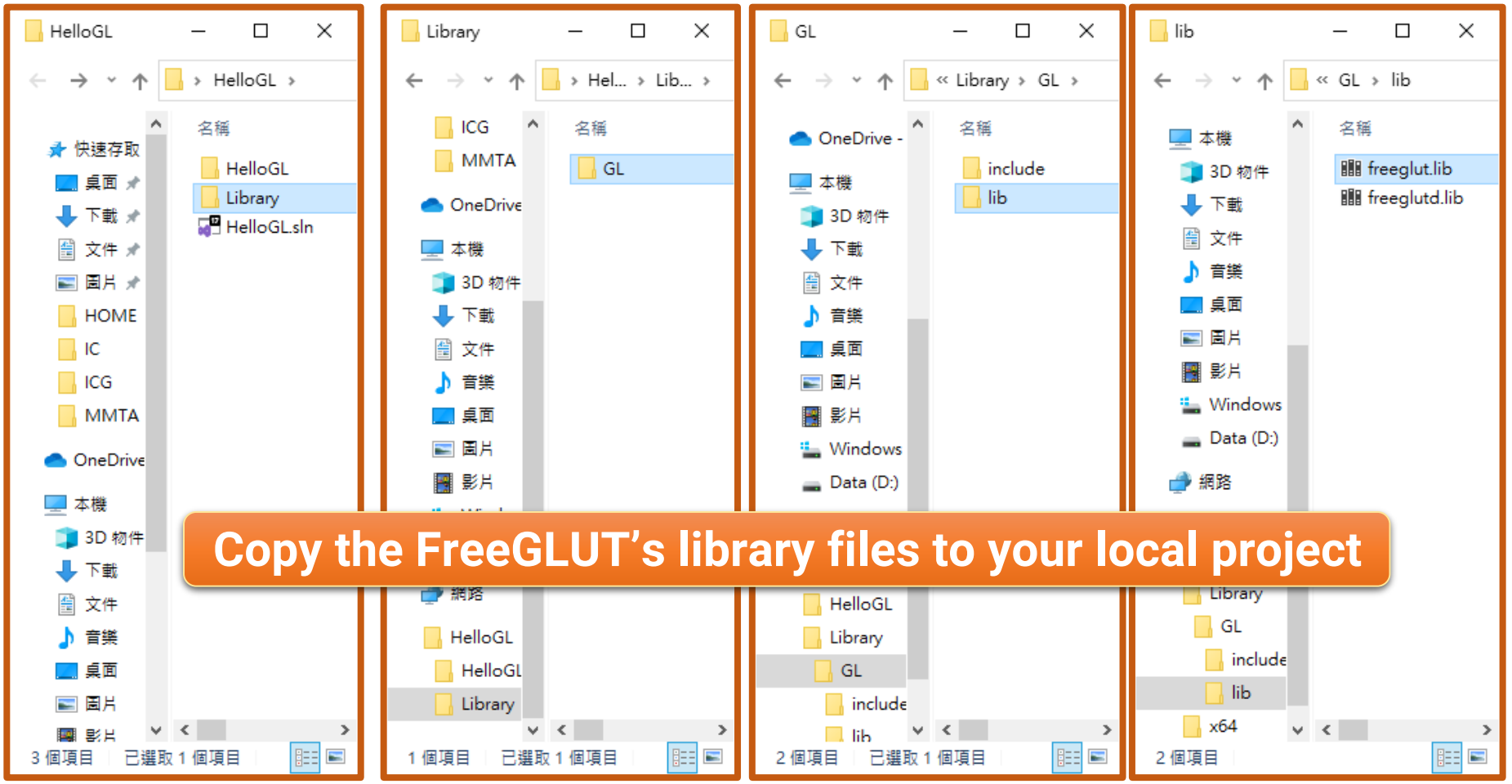


Setup the Static Library in VS

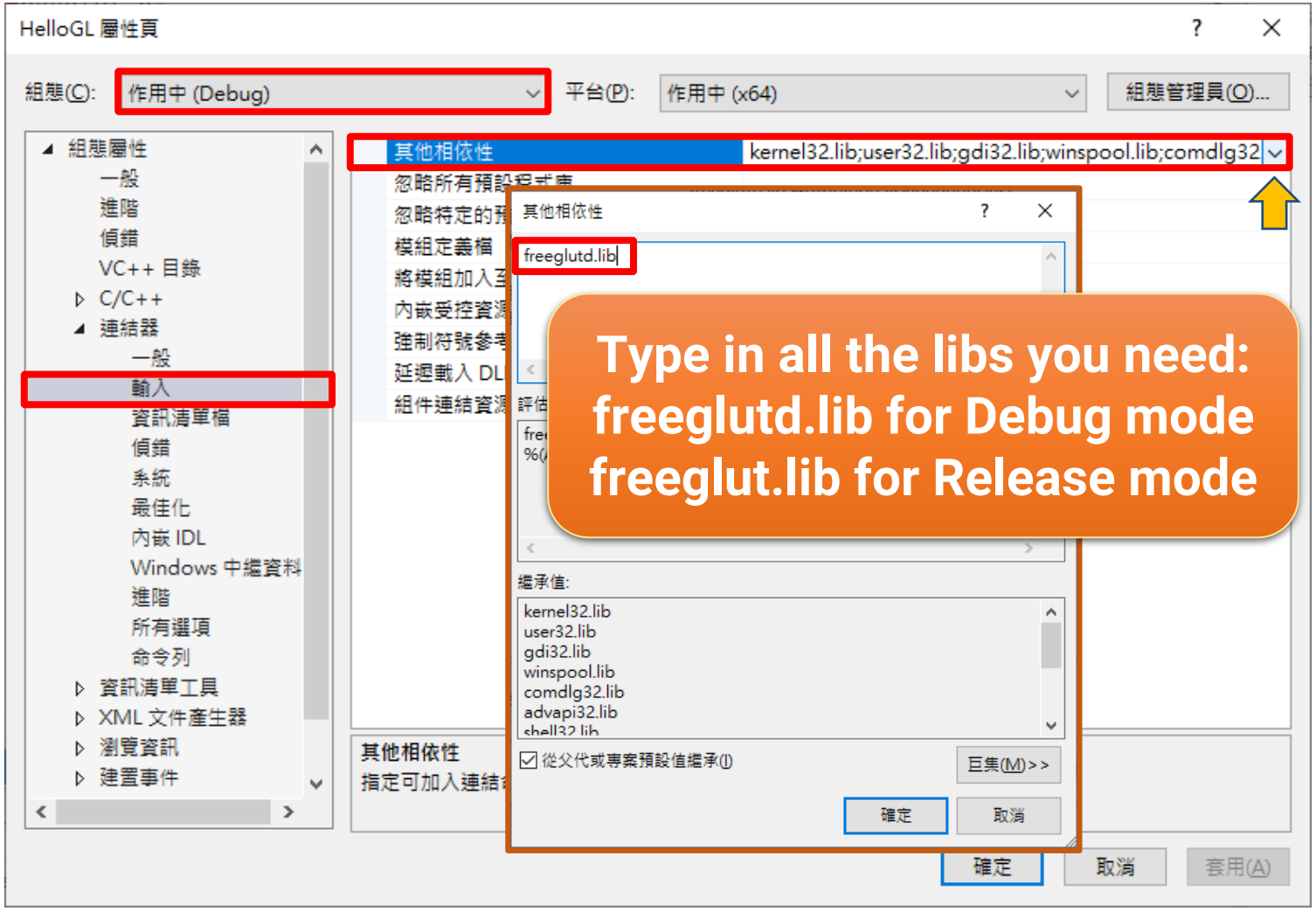


Setup the Static Library in VS (cont.)

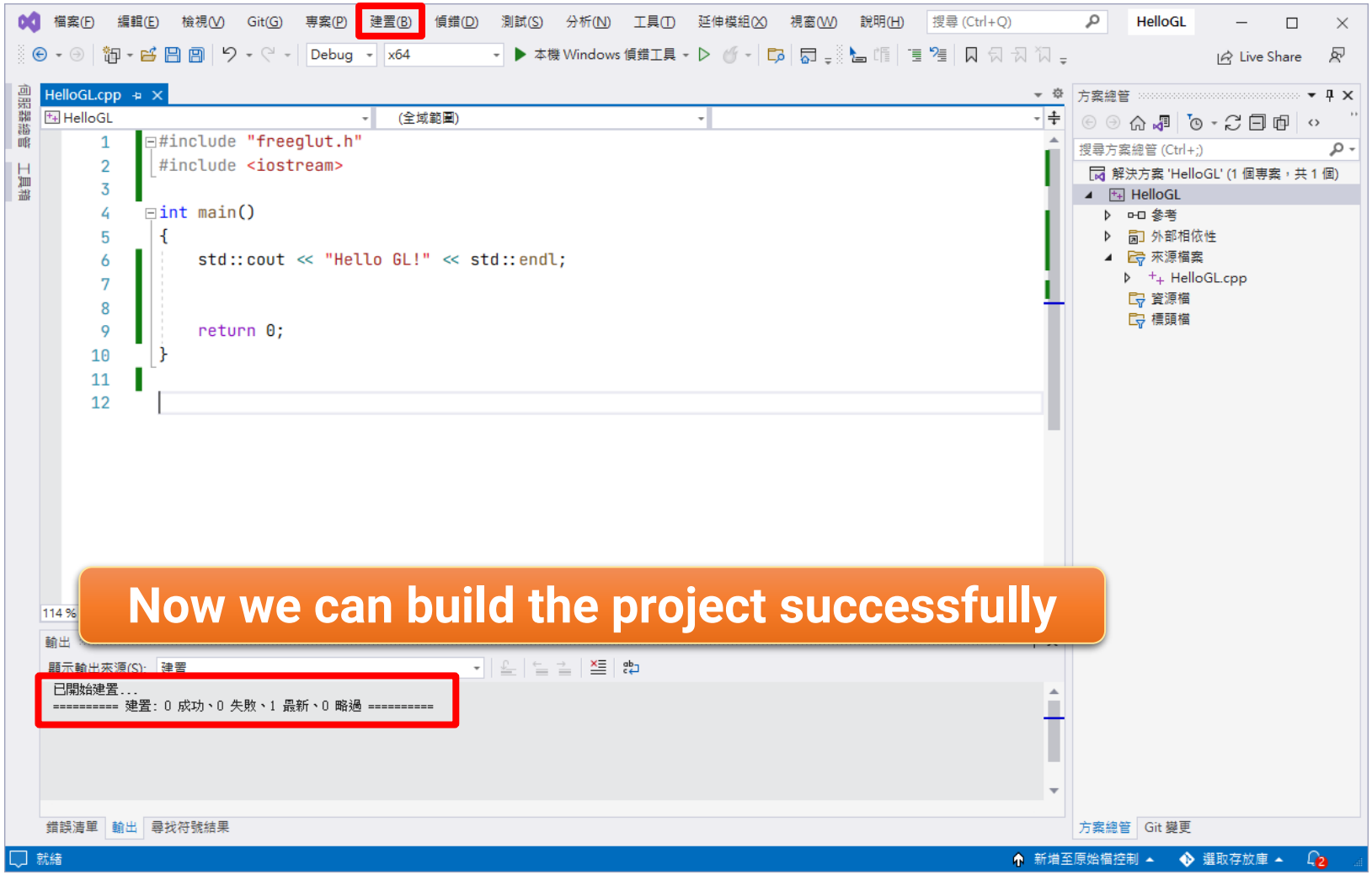
- My setting



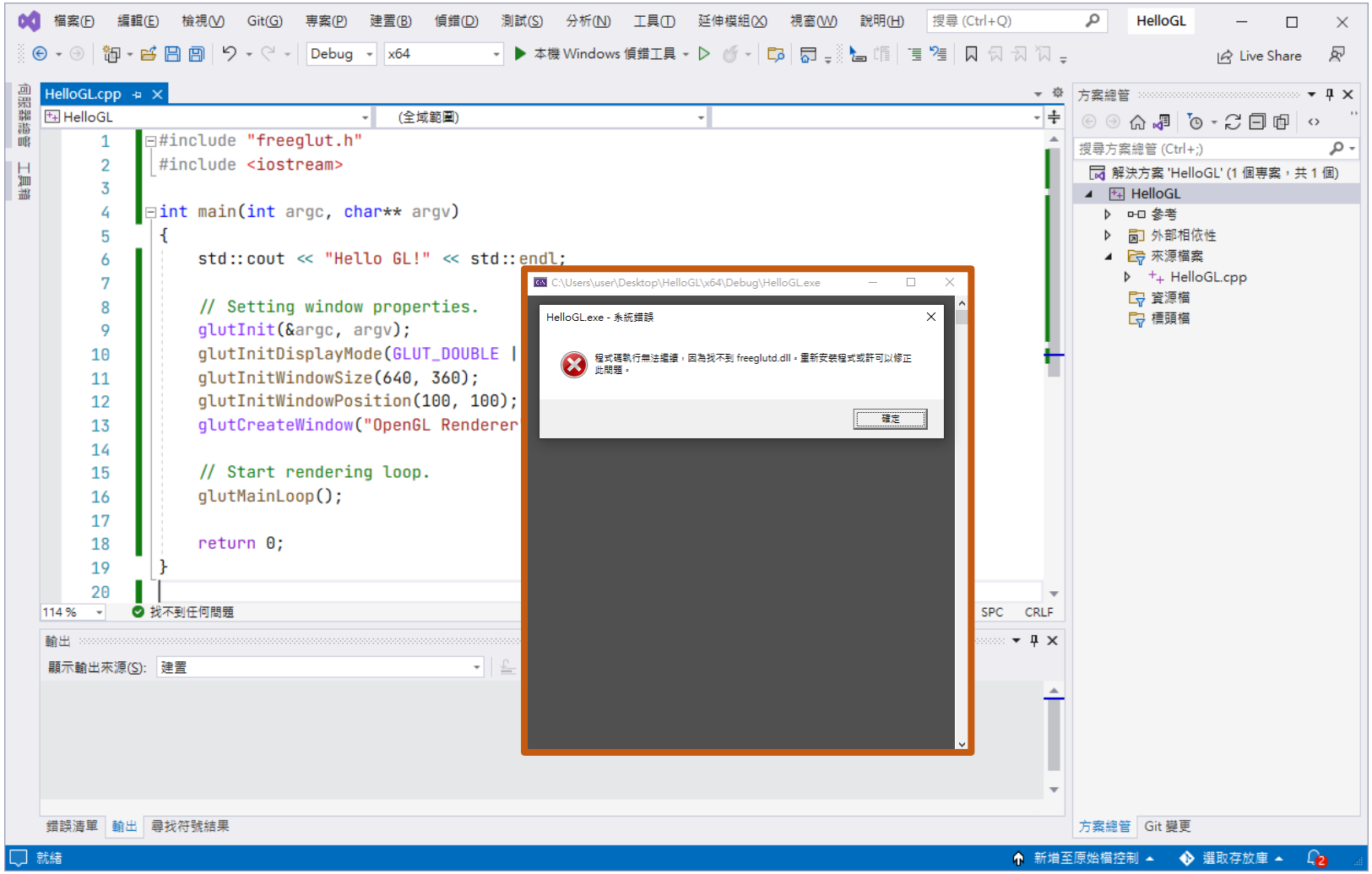
Setup the Static Library in VS (cont.)



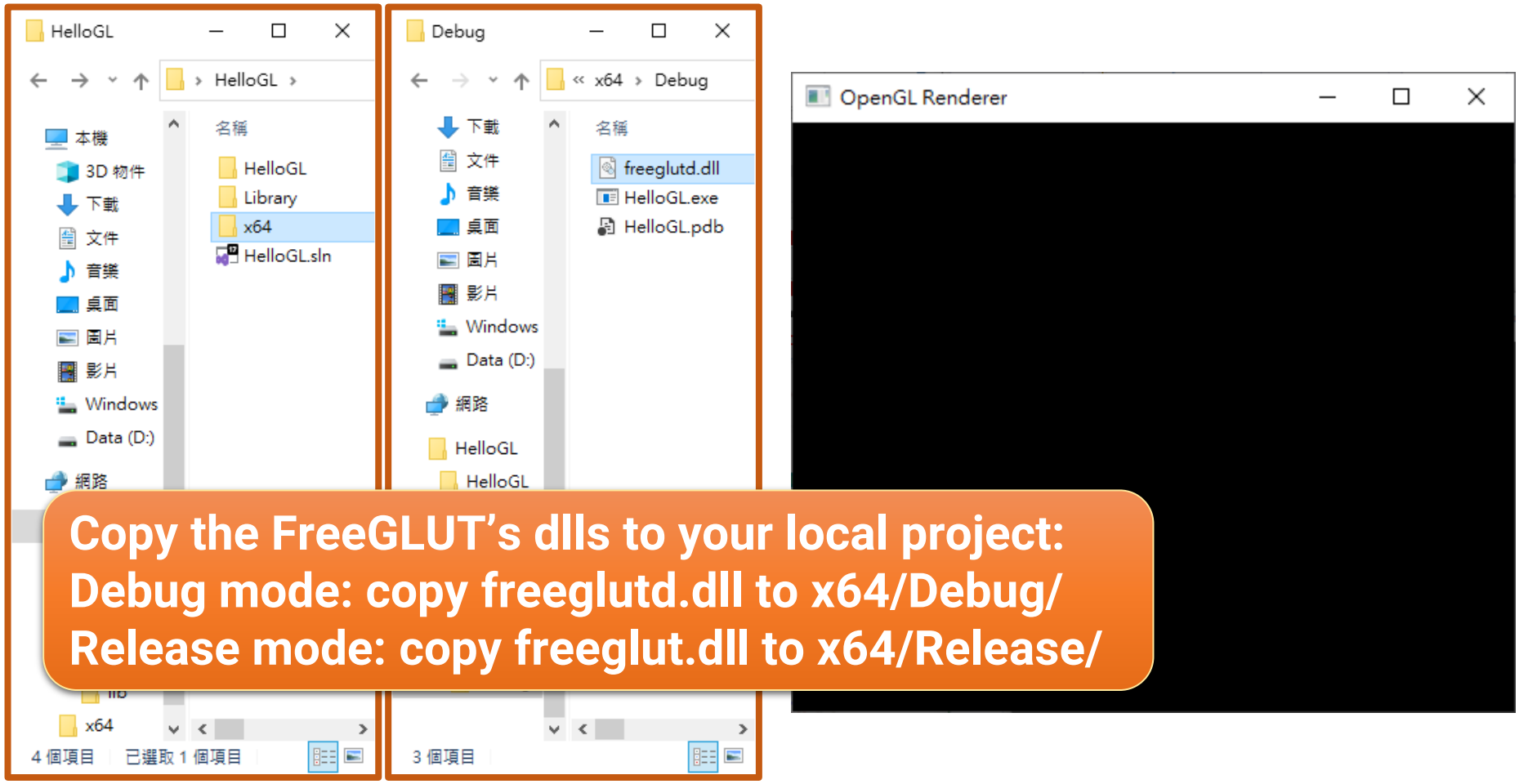
Setup the Static Library in VS (cont.)



Setup the Dynamic Library in VS



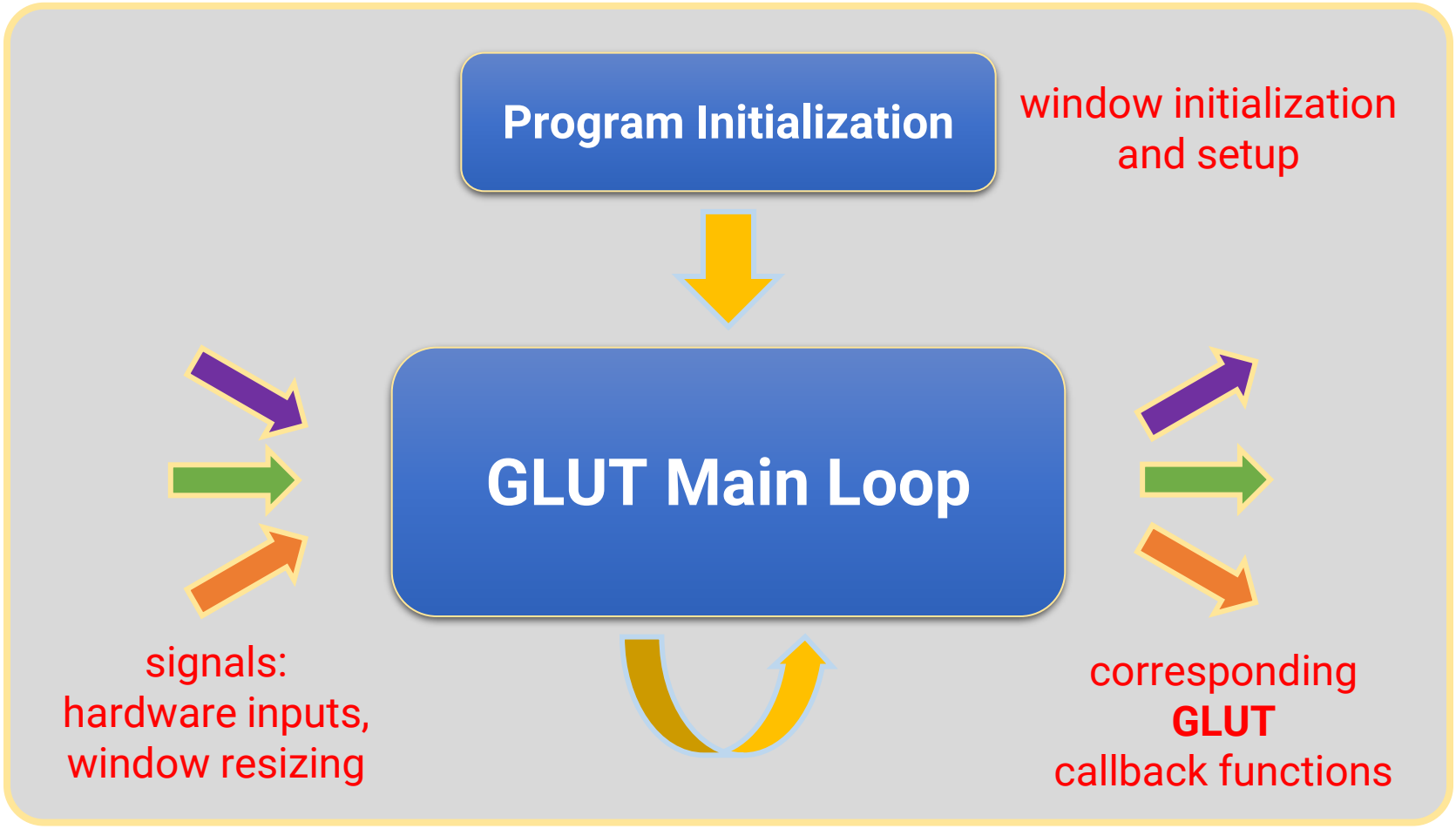
Setup the Dynamic Library in VS (cont.)



Outline

- Environment setup
- **The first OpenGL program**
- Appendix: build your own FreeGLUT libraries

Recap: Life Cycle of a GLUT Program



Structure of a GLUT Program

```
// OpenGL and FreeGlut headers.
#include <freeglut.h>
int main(int argc, char** argv)
{
    // Setting window properties.
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
    glutInitWindowSize(640, 360);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("OpenGL Renderer");

    // Initialization.
    SetupRenderState();

    // Register callback functions.
    glutDisplayFunc(RenderSceneCB);
    glutIdleFunc(RenderSceneCB);
    glutReshapeFunc(ReshapeCB);
    glutSpecialFunc(ProcessSpecialKeysCB);
    glutKeyboardFunc(ProcessKeysCB);

    // Start rendering loop.
    glutMainLoop();

    return 0;
}
```

create the window and set window properties

do initialization jobs

register callback functions

start the main loop

A FreeGLUT Window

- FreeGLUT will create and maintain a window on screen

Structure of a GLUT Program

```
// OpenGL and FreeGlut headers.
#include <freeglut.h>
int main(int argc, char** argv)
{
    // Setting window properties.
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
    glutInitWindowSize(640, 360);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("OpenGL Renderer");

    // Initialization.
    SetupRenderState();

    // Register callback functions.
    glutDisplayFunc(RenderSceneCB);
    glutIdleFunc(RenderSceneCB);
    glutReshapeFunc(ReshapeCB);
    glutSpecialFunc(ProcessSpecialKeysCB);
    glutKeyboardFunc(ProcessKeysCB);

    // Start rendering loop.
    glutMainLoop();

    return 0;
}
```

create the window and set window properties

API: Create an OpenGL (GLUT) Window

- void **glutInit**(int *argc, char **argv);
  - Initialize the GLUT library

```
glutInit(&argc, argv);
```

- int **glutCreateWindow**(char *name);
  - Create a top-level window

```
glutCreateWindow("OpenGL Renderer");
```

API: Setting Window Properties

- void **glutInitWindowSize**(int width, int height);
  - Set the initial window size
- void **glutInitWindowPosition**(int x, int y);
  - Set the initial window position

```
glutInitWindowSize(640, 360);
glutInitWindowPosition(100, 100);
```

- void **glutInitDisplayMode**(unsigned int mode);
  - Set the initial display mode
  - https://www.opengl.org/resources/libraries/glut/spec3/node12.html

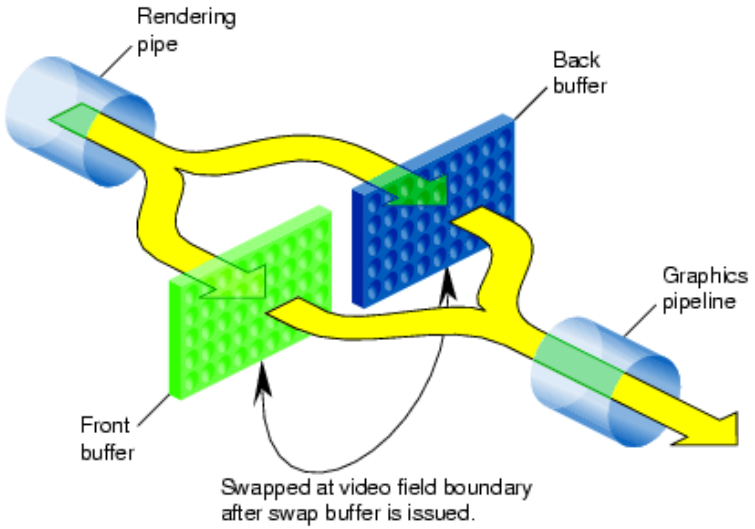
```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
```

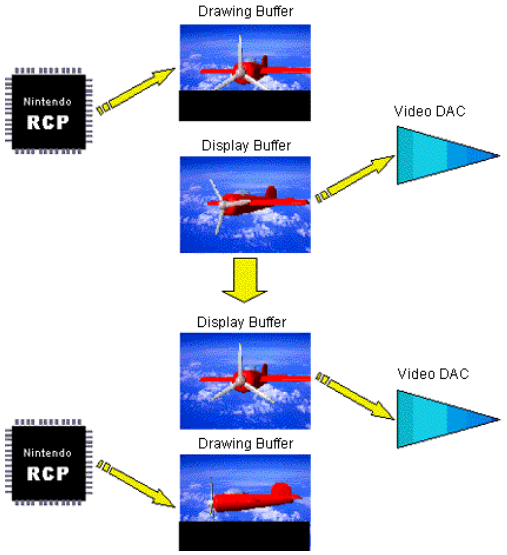
GLUT_DOUBLE: double buffer
GLUT_RGBA: color buffer format
GLUT_DEPTH: enable depth buffer

Double Buffers

```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
```

- Prevent artifacts due to potentially seeing parts of an incomplete frame (that is currently drawn)
  - Set the display mode to **GLUT_DOUBLE** in the **glutInitDisplayMode** function
  - Call **glutSwapBuffers** after rendering finished

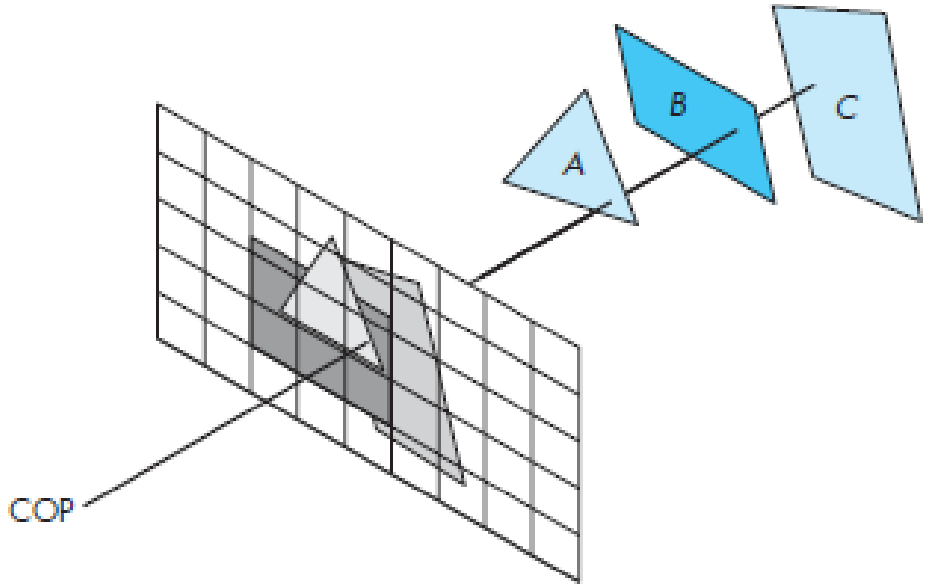




Depth Buffer

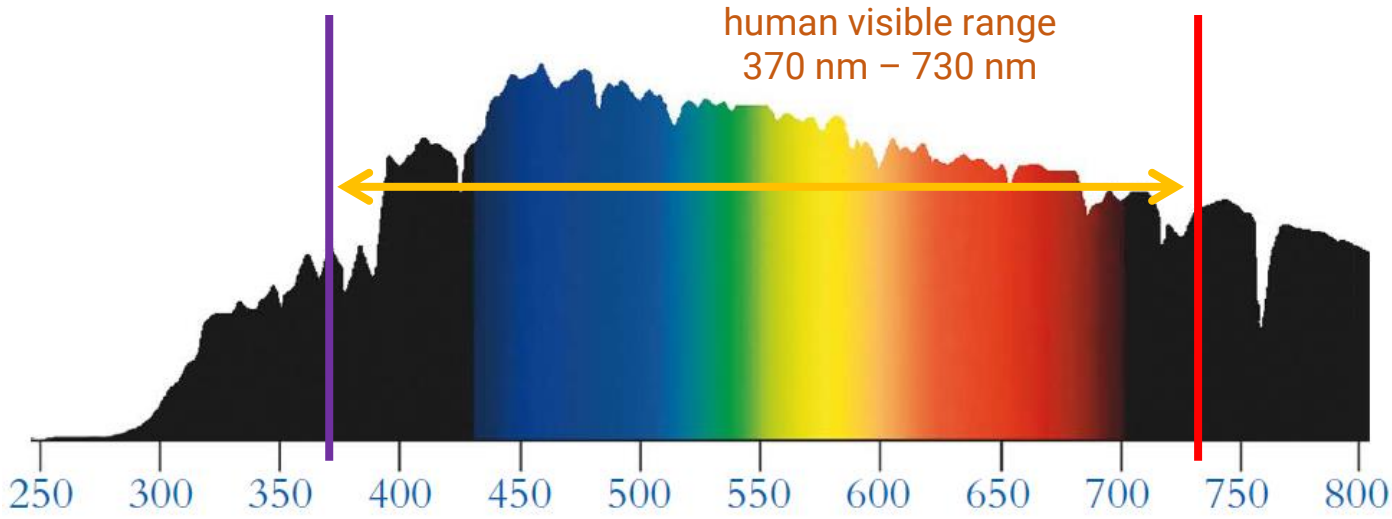
```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
```

- Keep track of the **nearest surface** to **each pixel** during rendering the scene (many surfaces are projected to cover the same pixel)



Color: Spectral Power Distribution

- Light is an electromagnetic wave, and we can measure its wavelength and intensity
- **Spectral power distribution (SPD)** is a description of how the intensity of light varies with its wavelength



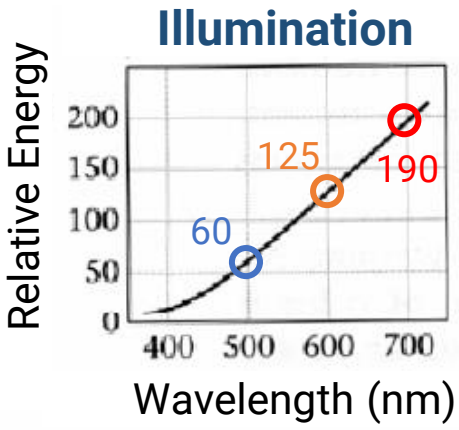
SPD of daylight

Color: Spectral Power Distribution (cont.)

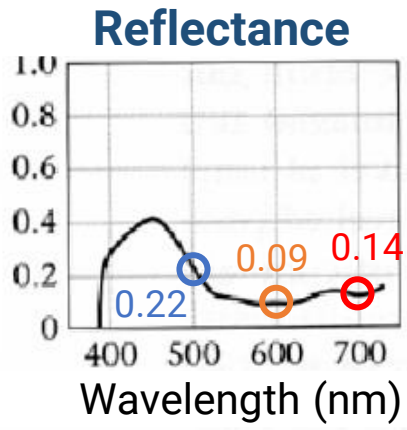
- Reflected color is the result of interaction of **light source spectrum** with **surface reflectance**

Color: Spectral Power Distribution (cont.)

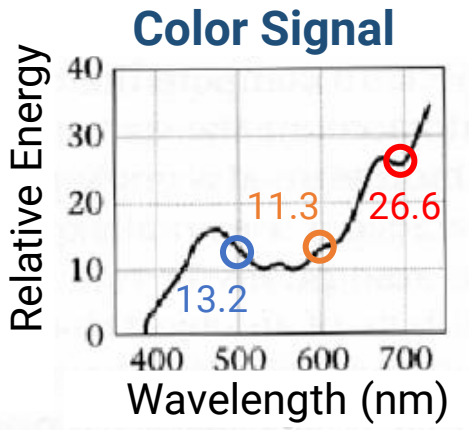
- Reflected color is the result of interaction of **light source spectrum** with **surface reflectance**



*



=



Tristimulus Theory

- SPDs are too cumbersome for representing the color in computer graphics
- Need a more compact, efficient, and accurate way to represent color signals
  - Find proper basis functions to map the infinite-dimensional space of all possible SPDs to a **low-dimensional space of coefficients**
- We use the **tristimulus theory**
  - All visible SPDs can be accurately represented with **three values**

  = Any color can be specified by just three values, giving the weights of each of the three components

Tristimulus Theory (cont.)

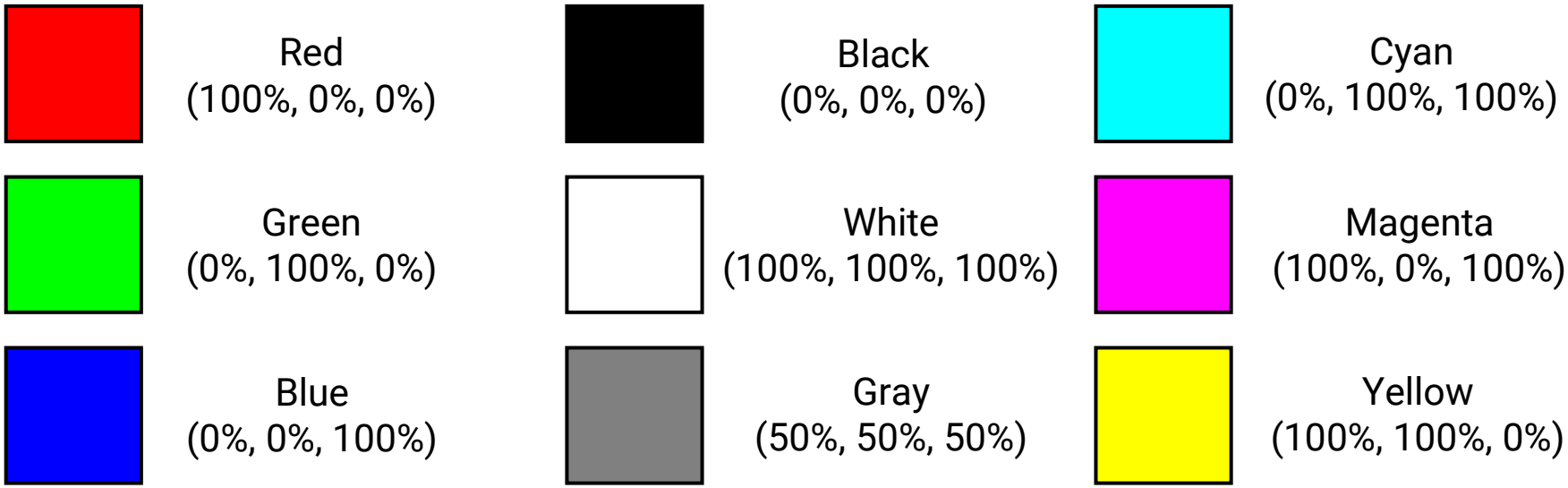
- For more details about tristimulus theory, please refer to the recording of my course, “Multimedia Technology and Application”
  - Course material link:
    - Part 1: https://reurl.cc/93D7x8
    - Part 2: https://reurl.cc/Lla4Dx
    - Part 3: https://reurl.cc/YqYVn4

RGB Color Model

- We can write a color with the RGB model in the form of

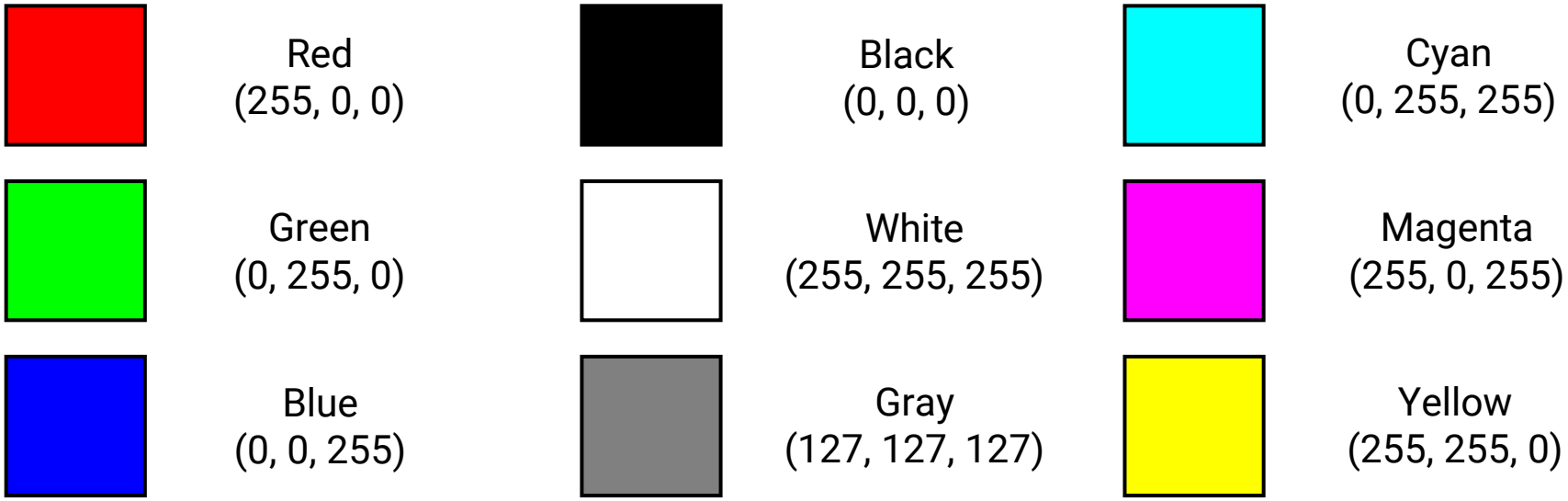
  (r, g, b),

where r, g ,b are the **amounts (proportion of the pure light)** of red, green, and blue light making up the color



RGB Color Model (cont.)

- In most applications, we use **8 bits** (1 byte) for each primary color, making 24 bits (3 bytes) in total
  - The range of each value falls within [0, 255], making a total of 256 x 256 x 256 = 16777216 different colors



RGB Color Model (cont.)

A? Alpha for transparency

```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
```

- In OpenGL, we use a floating value between **[0, 1]** for each primary color

```
float clearColor[4] = {0.44f, 0.57f, 0.75f, 1.00f};
glClearColor(
    (GLclampf)(clearColor[0]),
    (GLclampf)(clearColor[1]),
    (GLclampf)(clearColor[2]),
    (GLclampf)(clearColor[3])
);
```

Red (1.0f, 0.0f, 0.0f)	Black (0.0f, 0.0f, 0.0f)	Cyan (0.0f, 1.0f, 1.0f)
Green (0.0f, 1.0f, 0.0f)	White (1.0f, 1.0f, 1.0f)	Magenta (1.0f, 0.0f, 1.0f)
Blue (0.0f, 0.0f, 1.0f)	Gray (0.5f, 0.5f, 0.5f)	Yellow (1.0f, 1.0f, 0.0f)

Structure of a GLUT Program

```
// OpenGL and FreeGlut headers.
#include <freeglut.h>
int main(int argc, char** argv)
{
    // Setting window properties.
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
    glutInitWindowSize(640, 360);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("OpenGL Renderer");

    // Initialization.
    SetupRenderState();

    // Register callback functions.
    glutDisplayFunc(RenderSceneCB);
    glutIdleFunc(RenderSceneCB);
    glutReshapeFunc(ReshapeCB);
    glutSpecialFunc(ProcessSpecialKeysCB);
    glutKeyboardFunc(ProcessKeysCB);

    // Start rendering loop.
    glutMainLoop();

    return 0;
}
```

register callback functions

API: Setting Callback Functions

- Register the callback functions when receiving events
- Commonly used
  - glutDisplayFunc
  - glutIdleFunc
  - glutReshapeFunc
  - glutKeyboardFunc / glutSpecialFunc
  - glutMouseFunc
  - glutMenuStatusFunc
- Each callback function has its own input format
- Please refer to the following page for all possible callback functions
  - https://www.opengl.org/resources/libraries/glut/spec3/node45.html

API: Setting Callback Functions (cont.)

```
void RenderSceneCB()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // Render something here.
    // TODO.

    glutSwapBuffers();
}


void ProcessKeysCB(unsigned char key, int x, int y)
{
    // Handle other keyboard inputs those are not defined as special keys.
    if (key == 27) {
        // Release memory allocation if needed.
        exit(0);
    }
}
```

clear the canvas (color buffer & depth buffer)

swap the front (for drawing) and back (for displaying) buffer

ESC

Structure of a GLUT Program

```
// OpenGL and FreeGlut headers.
#include <freeglut.h>
int main(int argc, char** argv)
{
    // Setting window properties.
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
    glutInitWindowSize(640, 360);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("OpenGL Renderer");

    // Initialization.
    SetupRenderState();

    // Register callback functions.
    glutDisplayFunc(RenderSceneCB);
    glutIdleFunc(RenderSceneCB);
    glutReshapeFunc(ReshapeCB);
    glutSpecialFunc(ProcessSpecialKeysCB);
    glutKeyboardFunc(ProcessKeysCB);

    // Start rendering loop.
    glutMainLoop();

    return 0;
}
```

do initialization jobs

API: Initialization

- void **glClearColor**(GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha);
  - Set the color to clear the color buffer

```
void SetupRenderState()
{
    float clearColor[4] = {0.44f, 0.57f, 0.75f, 1.00f};
    glClearColor(
        (GLclampf)(clearColor[0]),
        (GLclampf)(clearColor[1]),
        (GLclampf)(clearColor[2]),
        (GLclampf)(clearColor[3])
    );
}
```

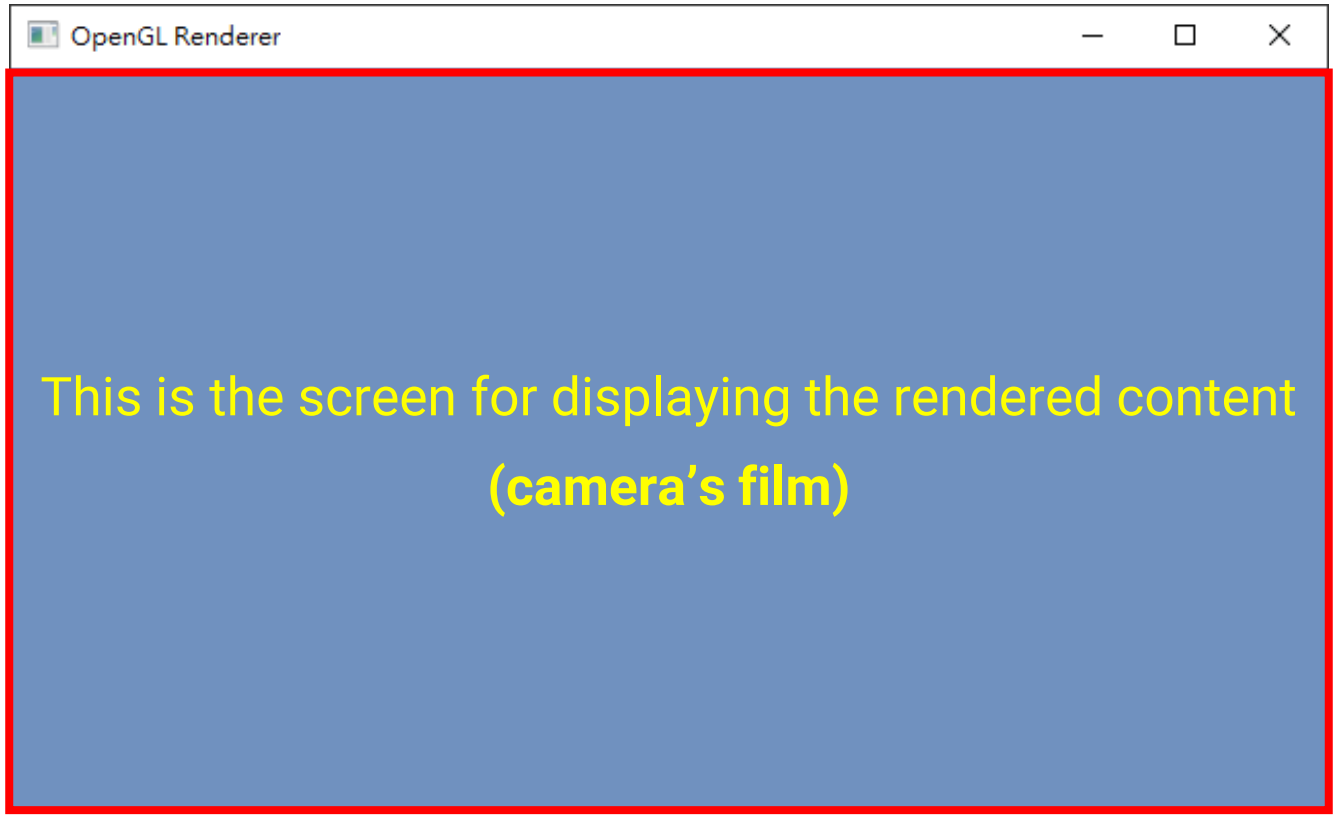
Structure of a GLUT Program

```
// OpenGL and FreeGlut headers.
#include <freeglut.h>
int main(int argc, char** argv)
{
    // Setting window properties.
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
    glutInitWindowSize(640, 360);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("OpenGL Renderer");

    // Initialization.
    SetupRenderState();

    // Register callback functions.
    glutDisplayFunc(RenderSceneCB);
    glutIdleFunc(RenderSceneCB);
    glutReshapeFunc(ReshapeCB);
    glutSpecialFunc(ProcessSpecialKeysCB);
    glutKeyboardFunc(ProcessKeysCB);

    // Start rendering loop.
    glutMainLoop();

    return 0;
}
```

start the main loop

API: Start the Main Rendering Loop

- void **glutMainLoop**(void);
  - Enter the GLUT event processing loop



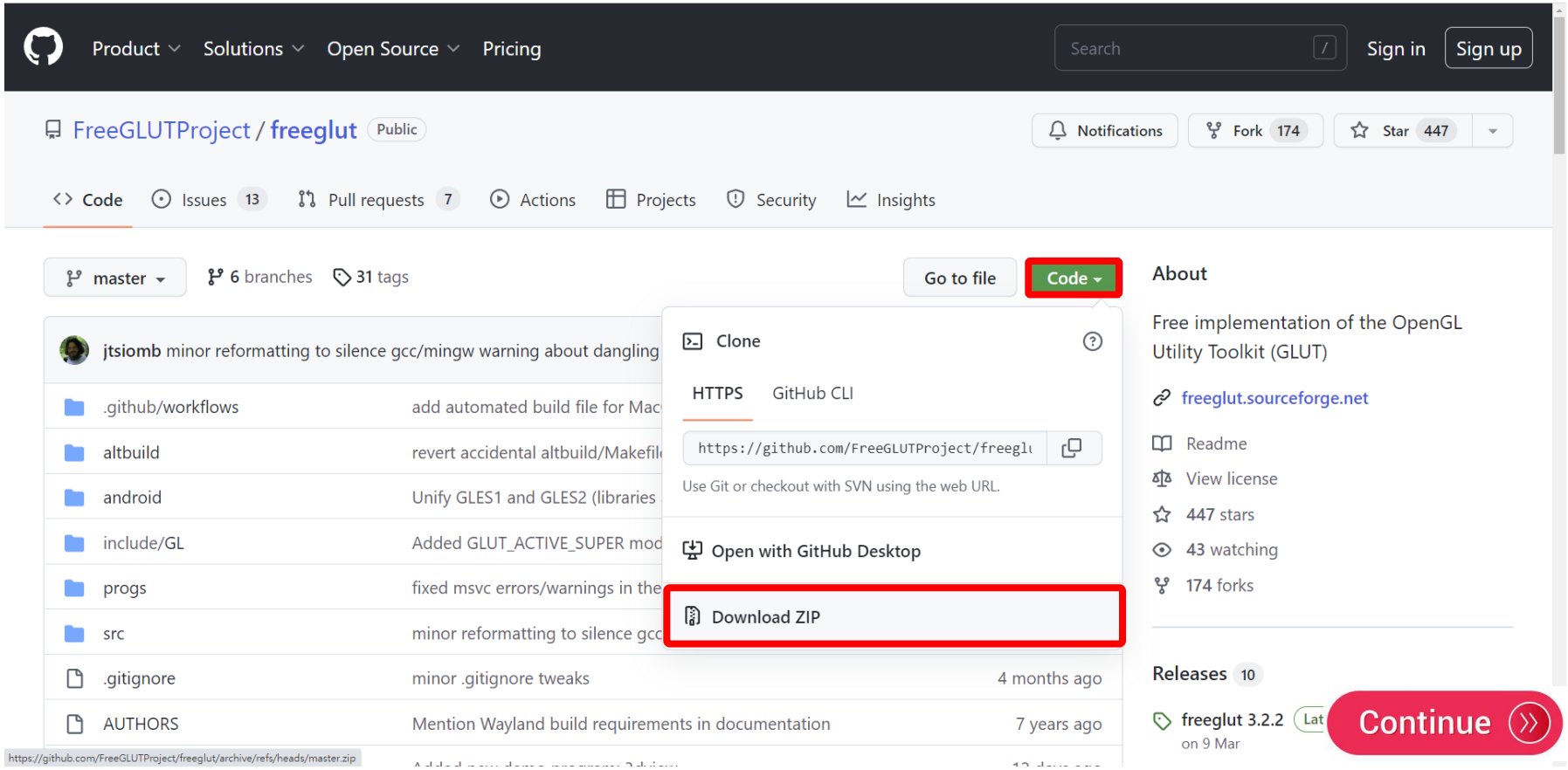
END OF PRESENTATION
ANY QUESTIONS?

Outline

- Environment setup
- The first OpenGL program
- **Appendix: build your own FreeGLUT libraries**

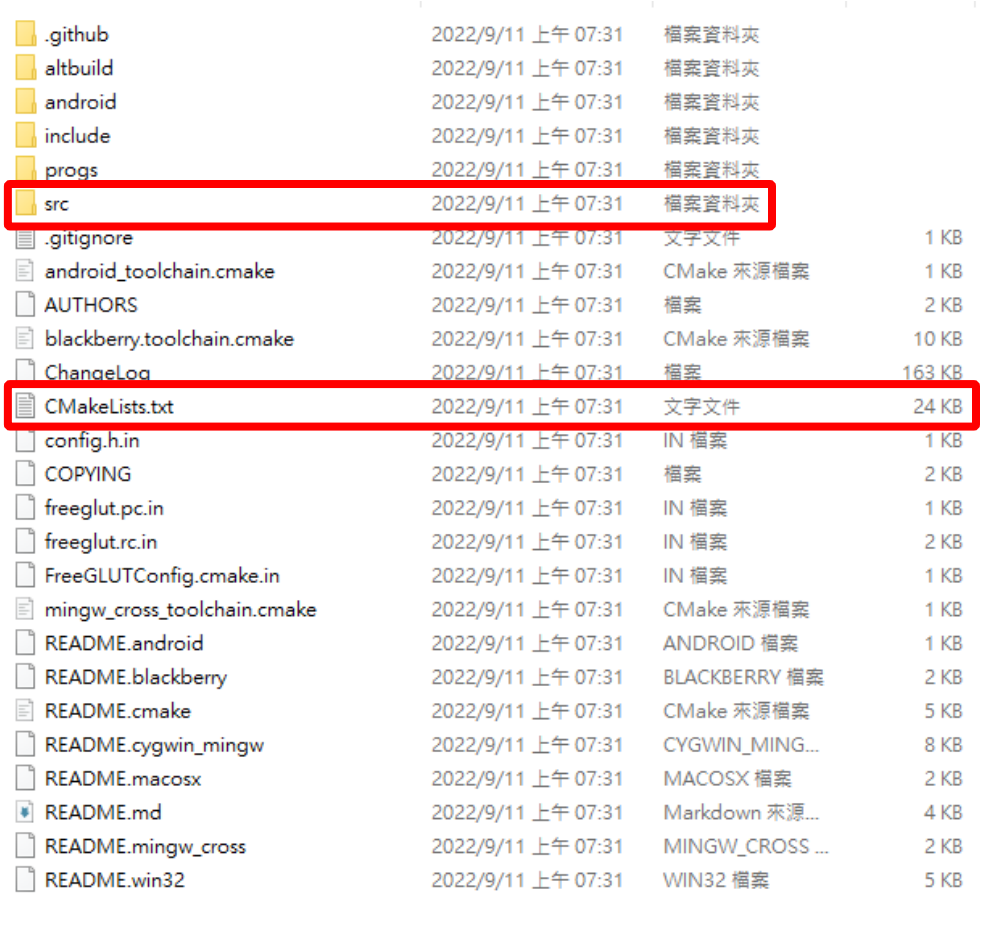
FreeGLUT

- Download the source code from https://github.com/FreeGLUTProject/freeglut



FreeGLUT (cont.)

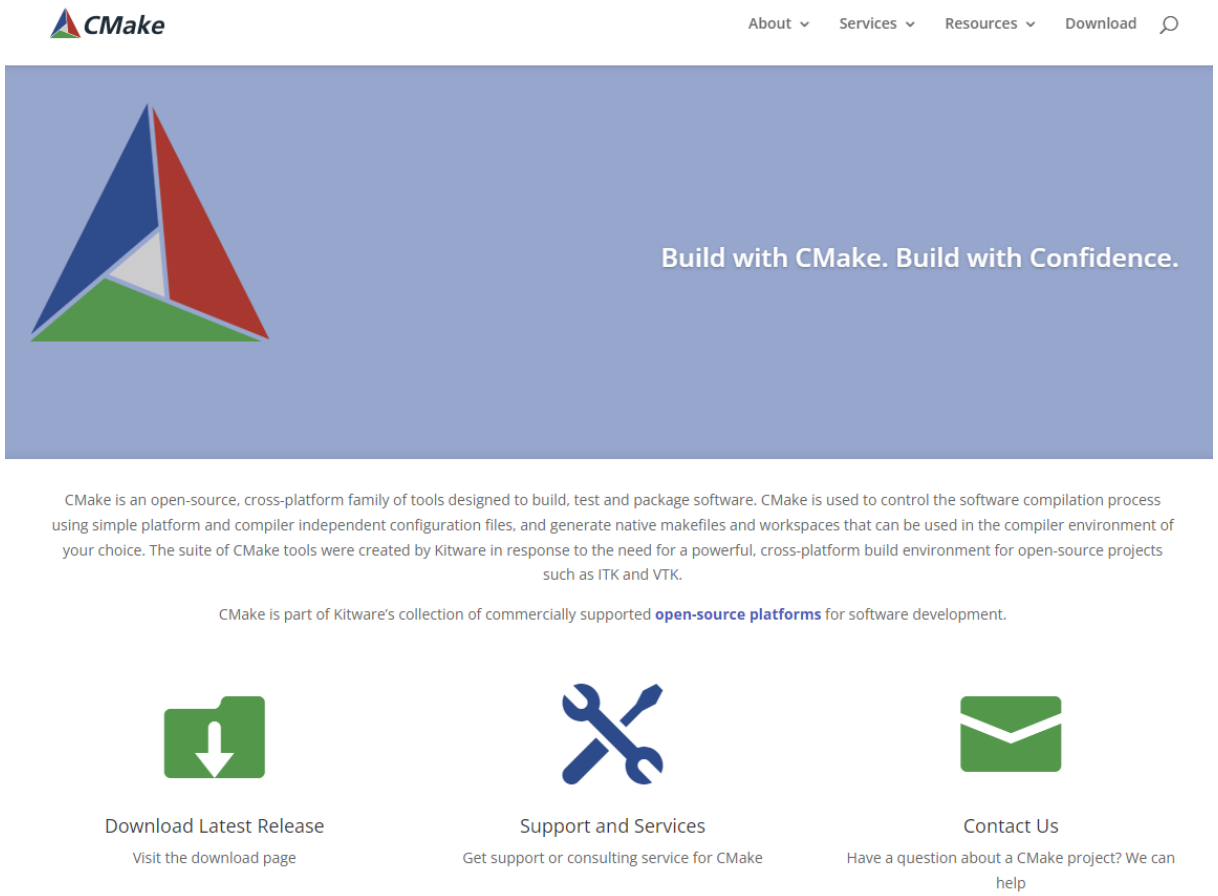
- Unzip the package



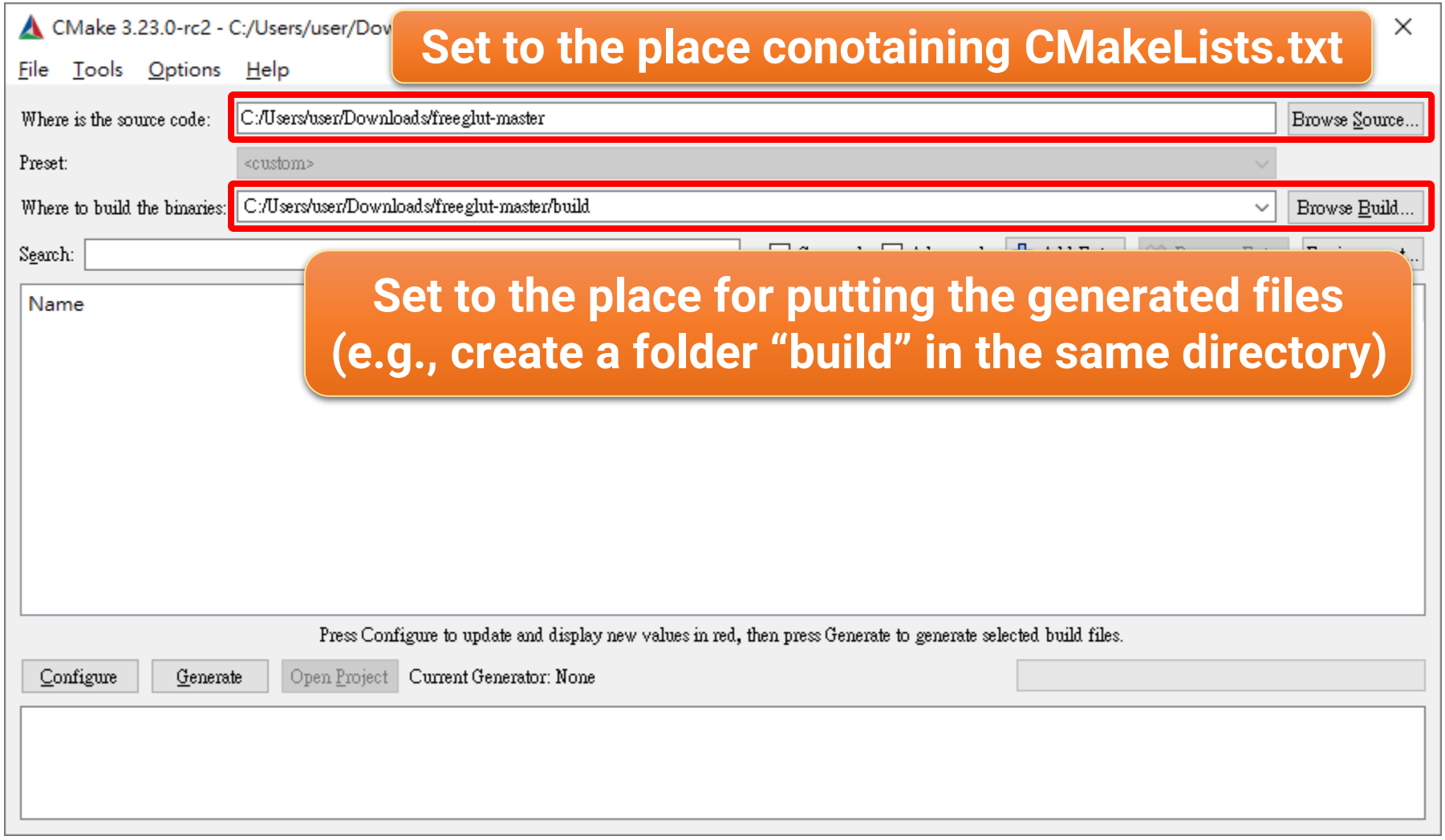
Build the source code using **CMake**

CMake

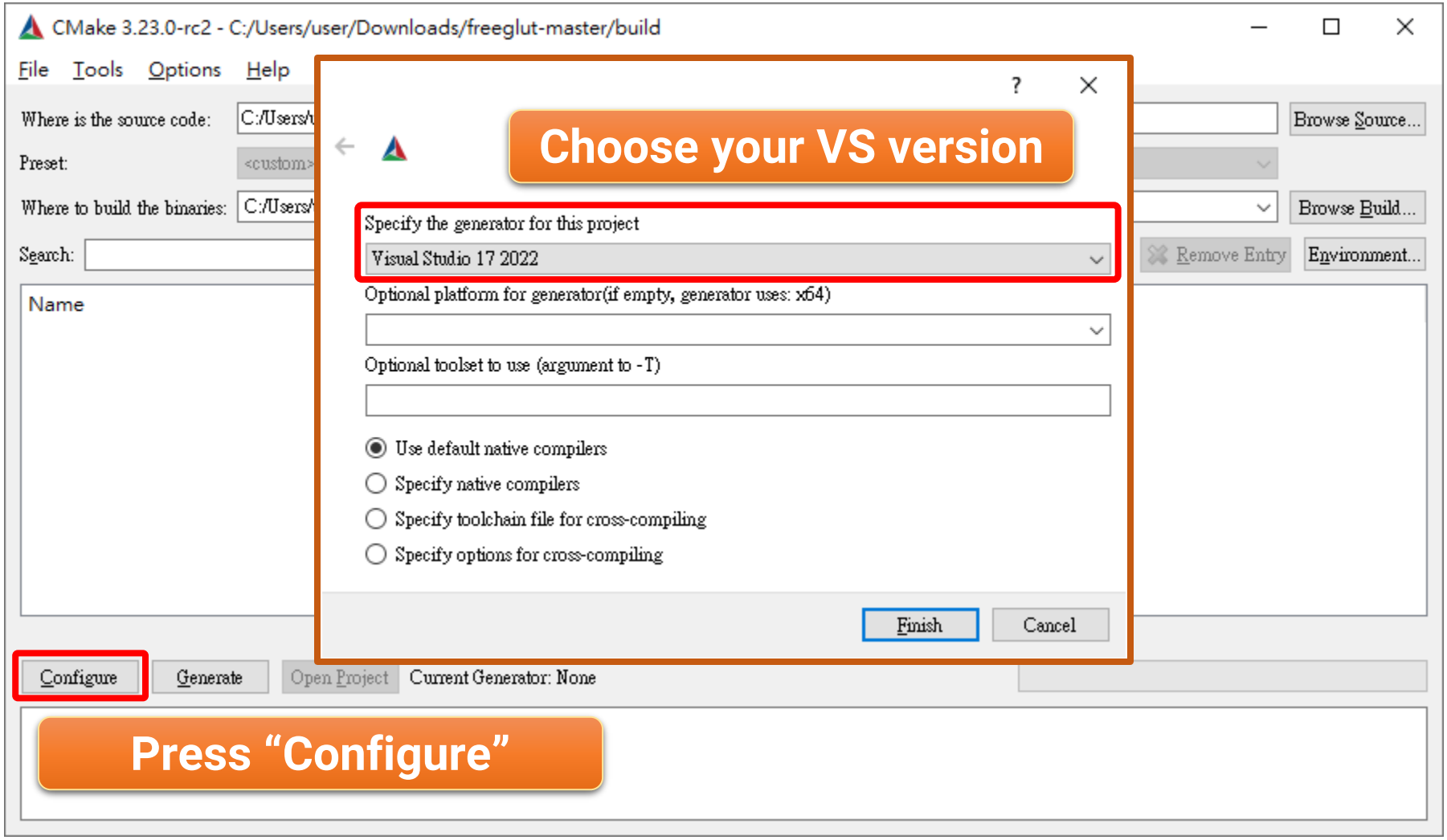
- Download and install CMake: https://cmake.org/



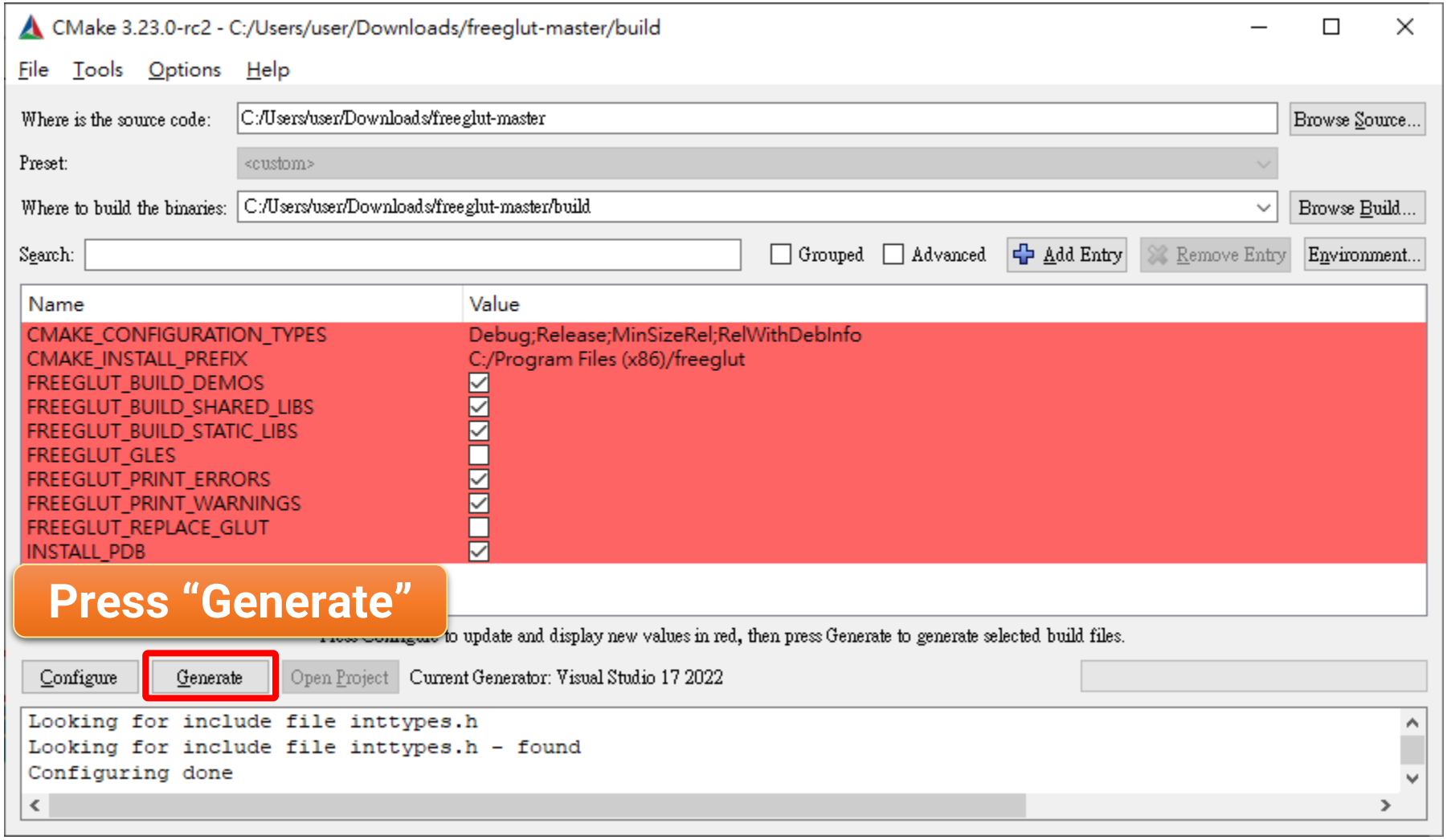
Setup CMake for Building FreeGLUT



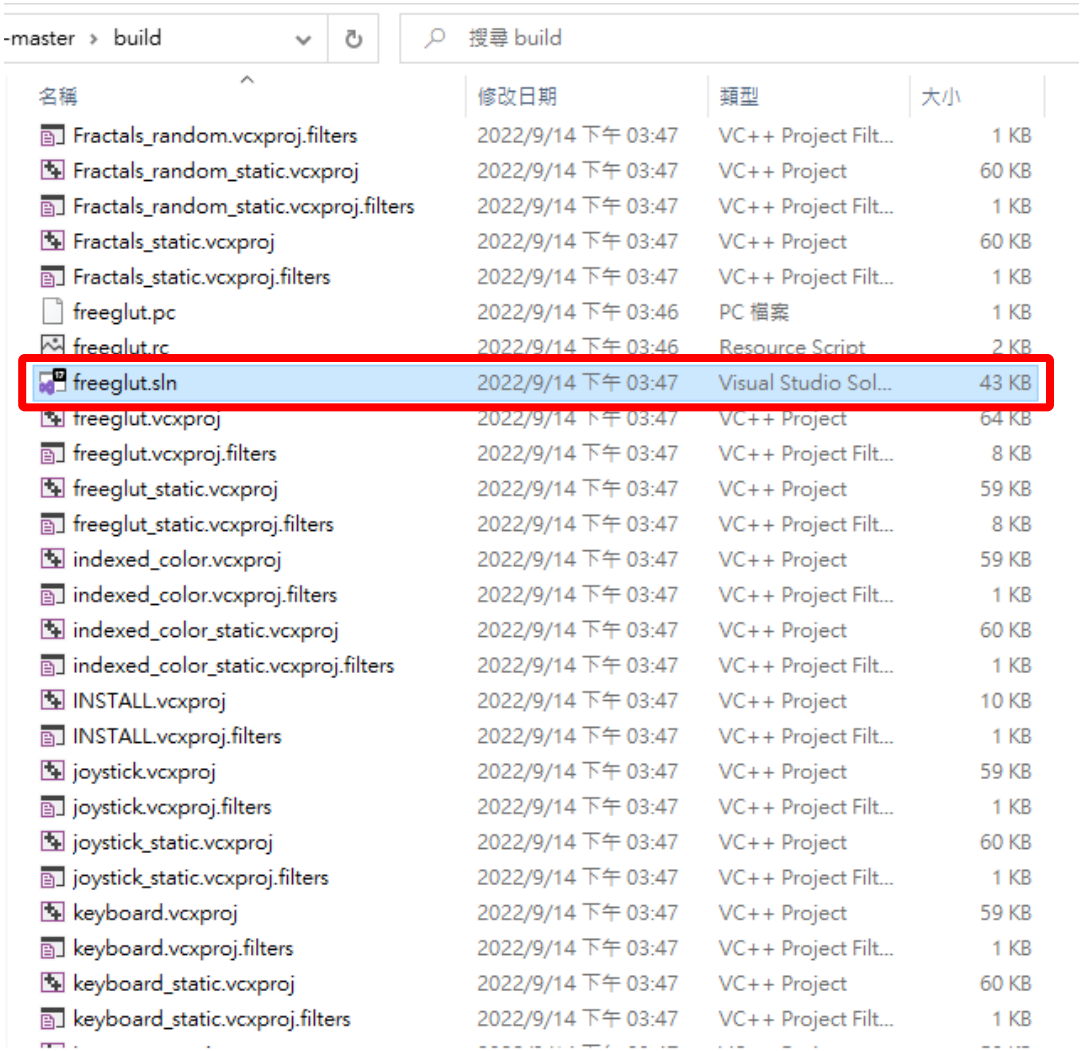
Configuration



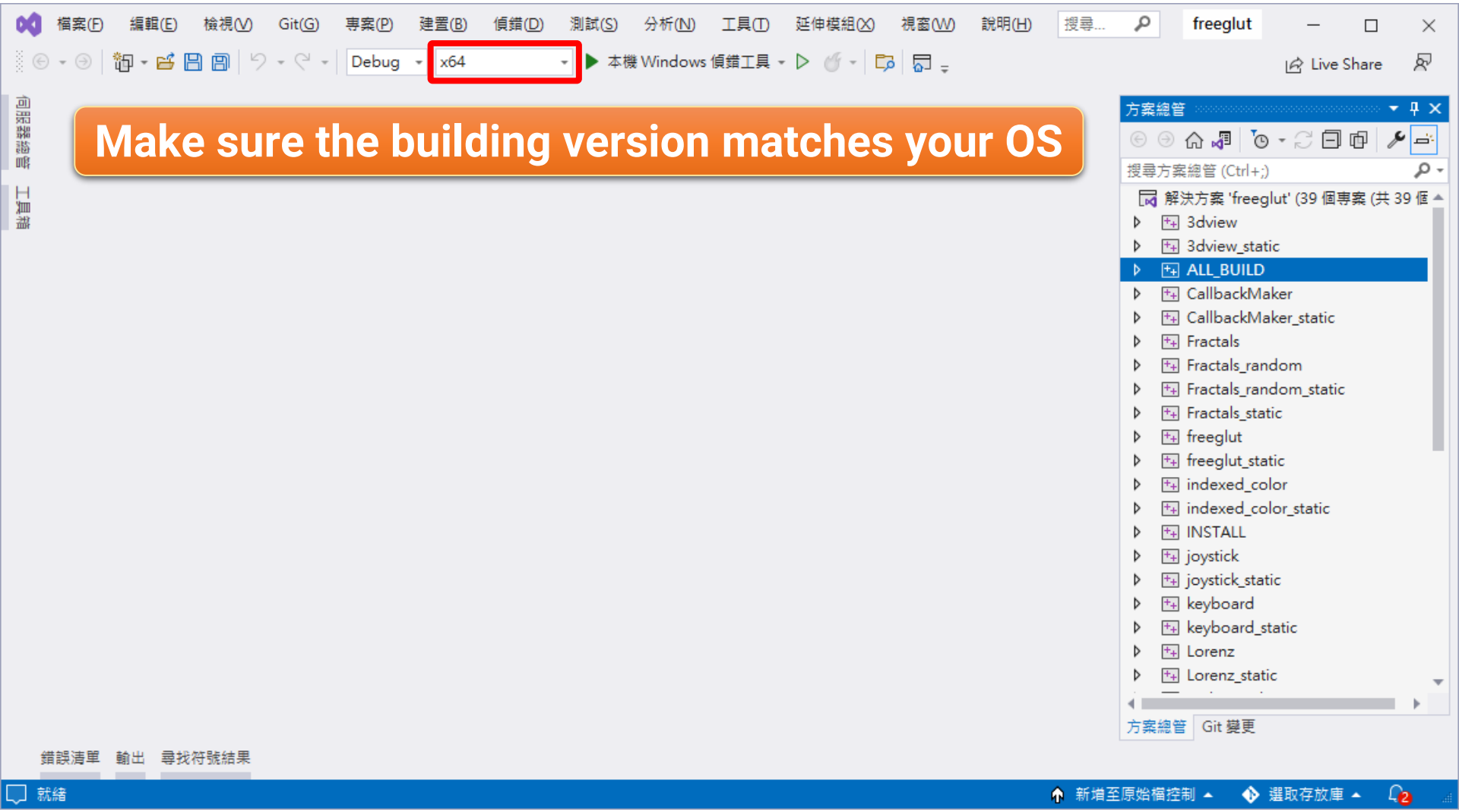
Generate VS Project



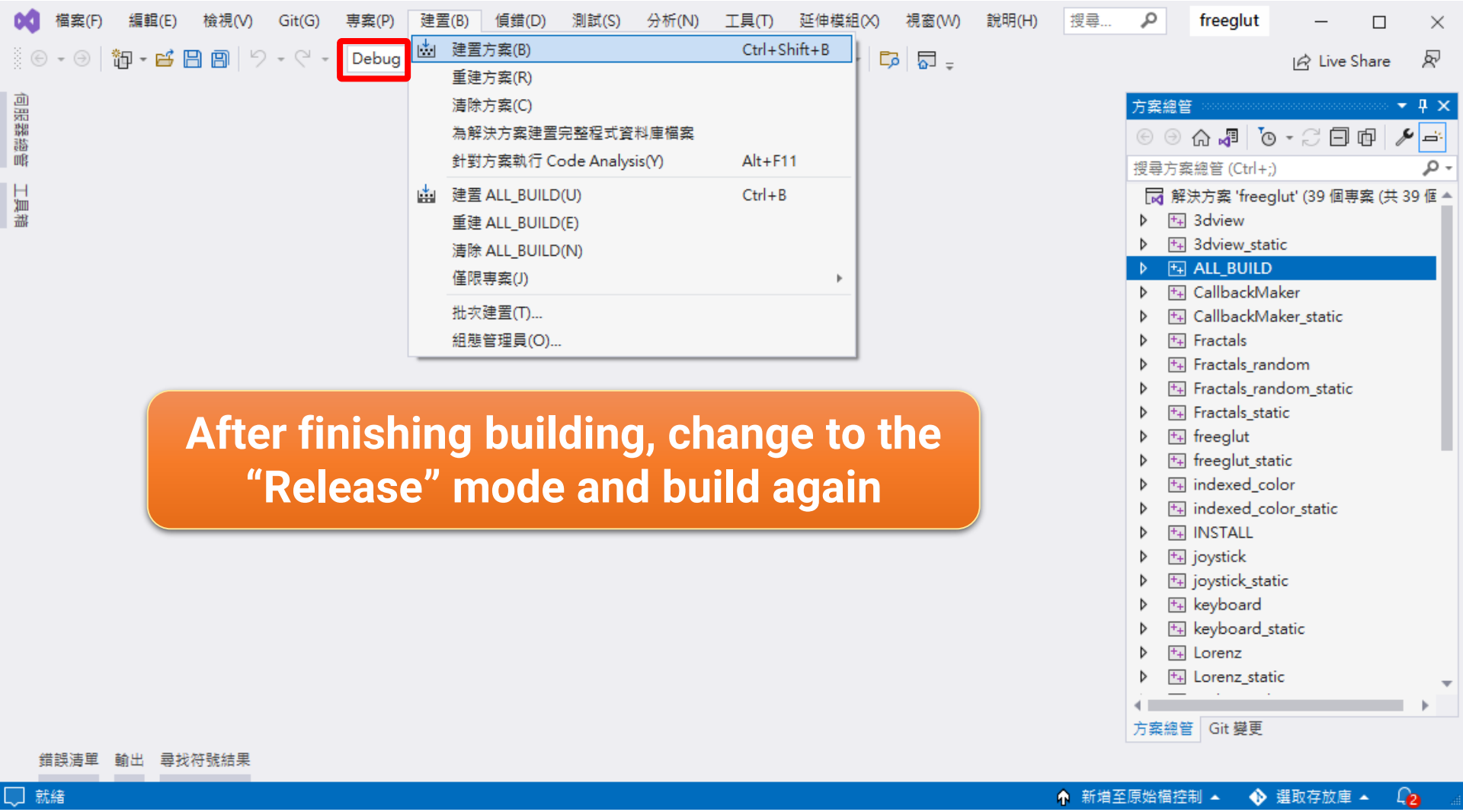
Examine VS Project



Open Solution with Visual Studio

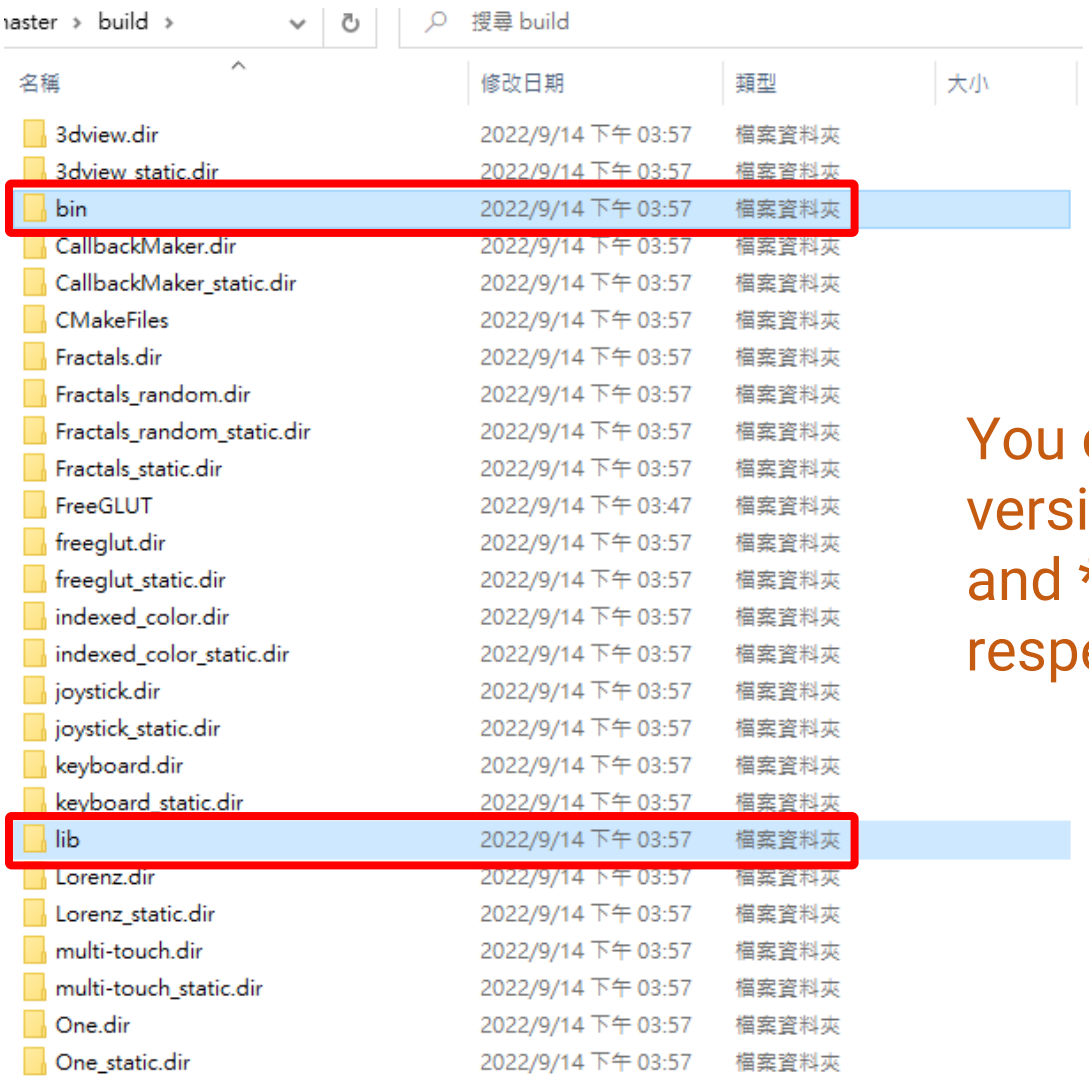


Debug/Release Build



After finishing building, change to the "Release" mode and build again

Examine the Built Binary Files



You can find the Debug/Release versions of ***.lib (in the lib folder)** and ***.dll (in the bin folder)**, respectively