



OpenGL Startup

Computer Graphics

Yu-Ting Wu

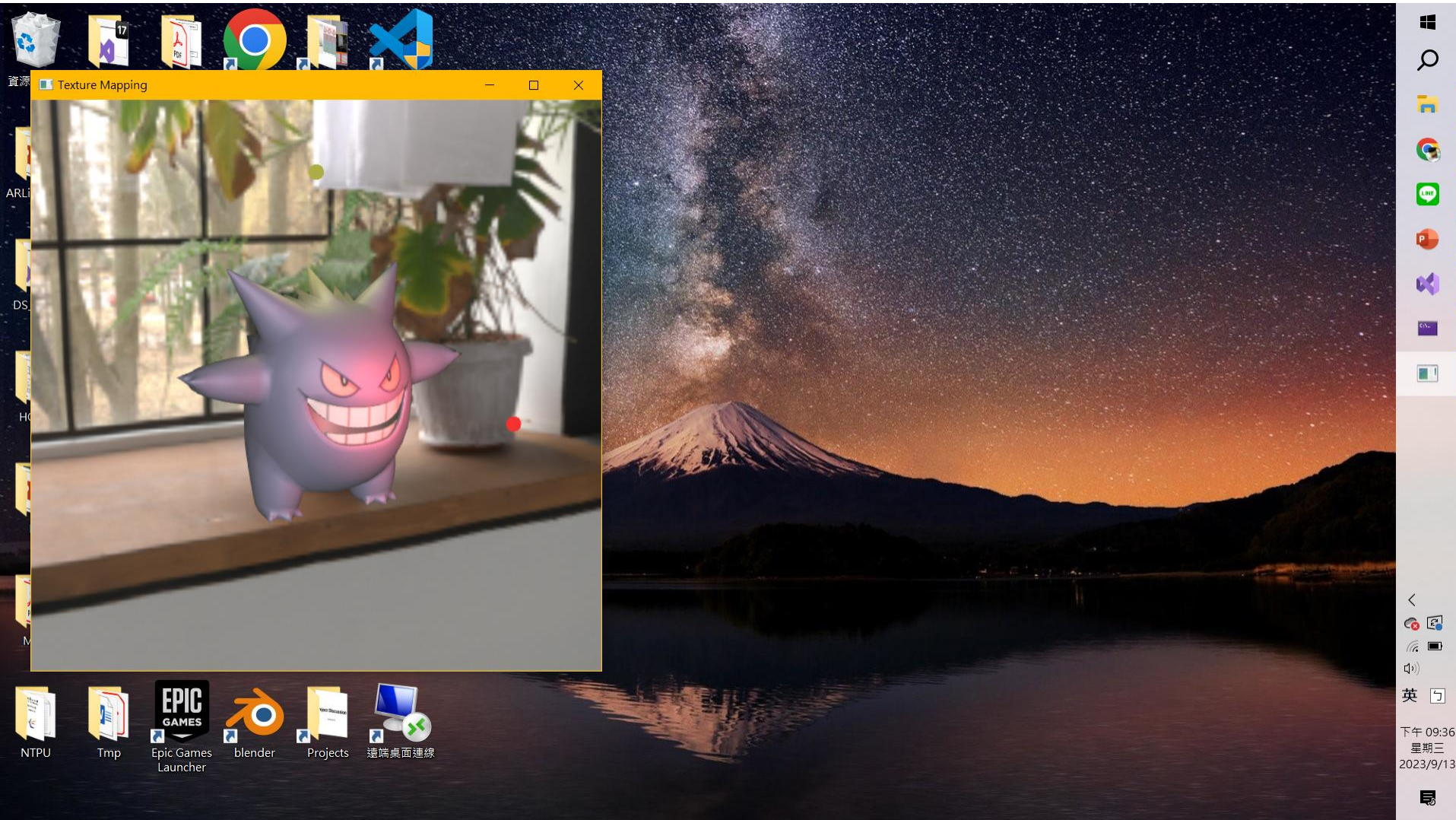
Outline

- [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



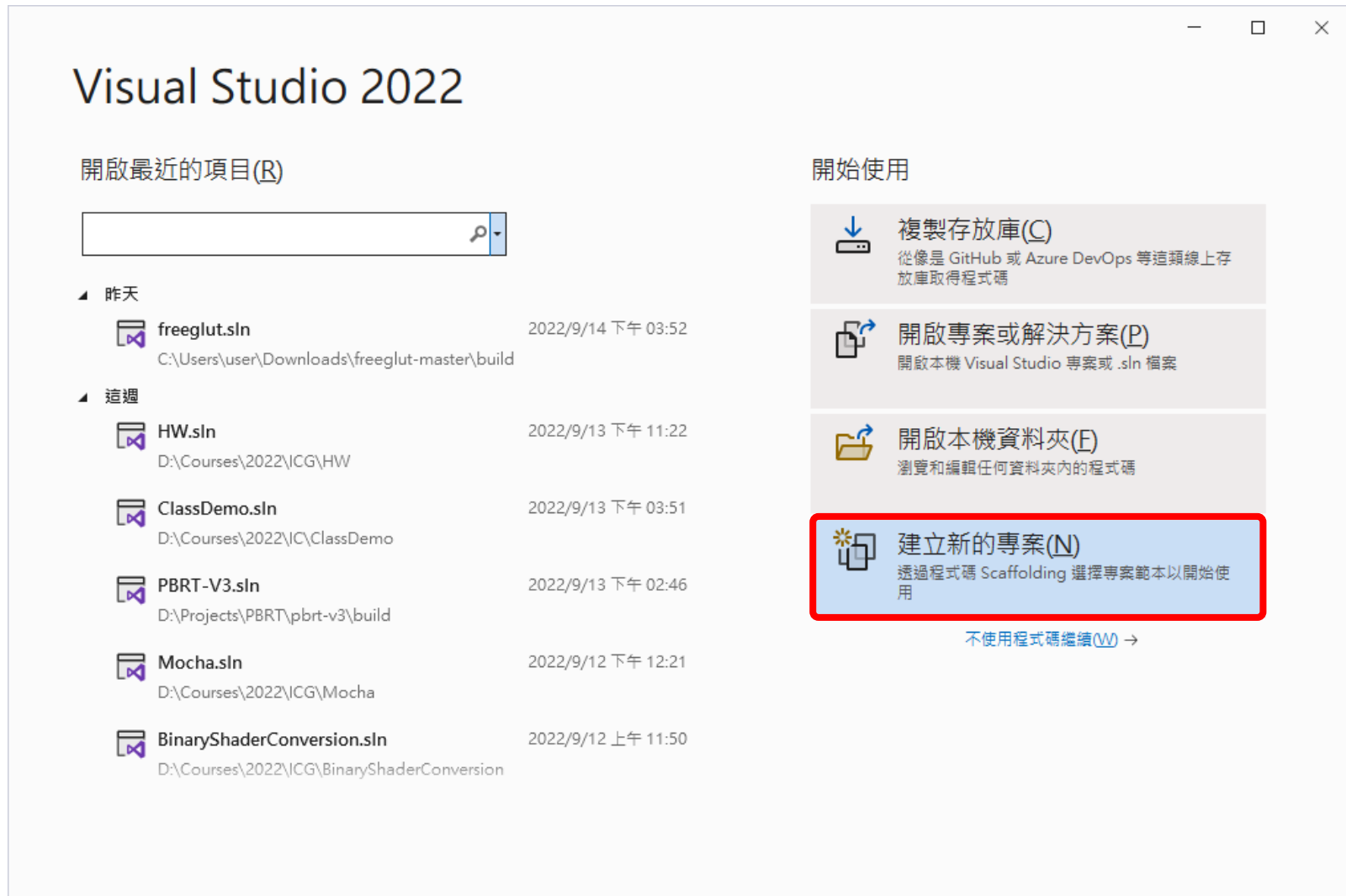
Library for Handling Screen Rendering

- **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

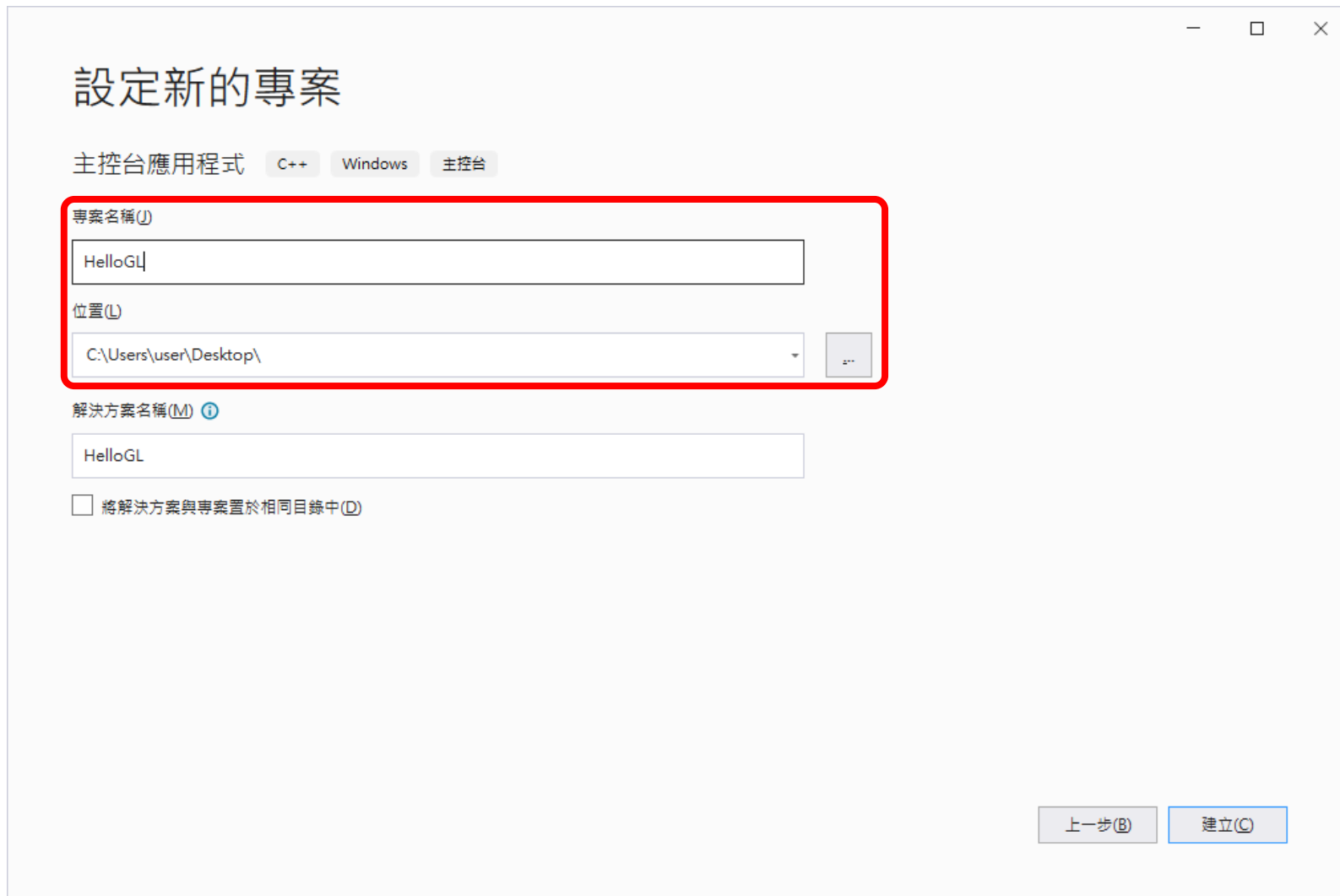
Create a New Project in VS



Create a New Project in VS (cont.)



Create a New Project in VS (cont.)



設定新的專案

主控台應用程式 C++ Windows 主控台

專案名稱(P)

HelloGL

位置(L)

C:\Users\user\Desktop\

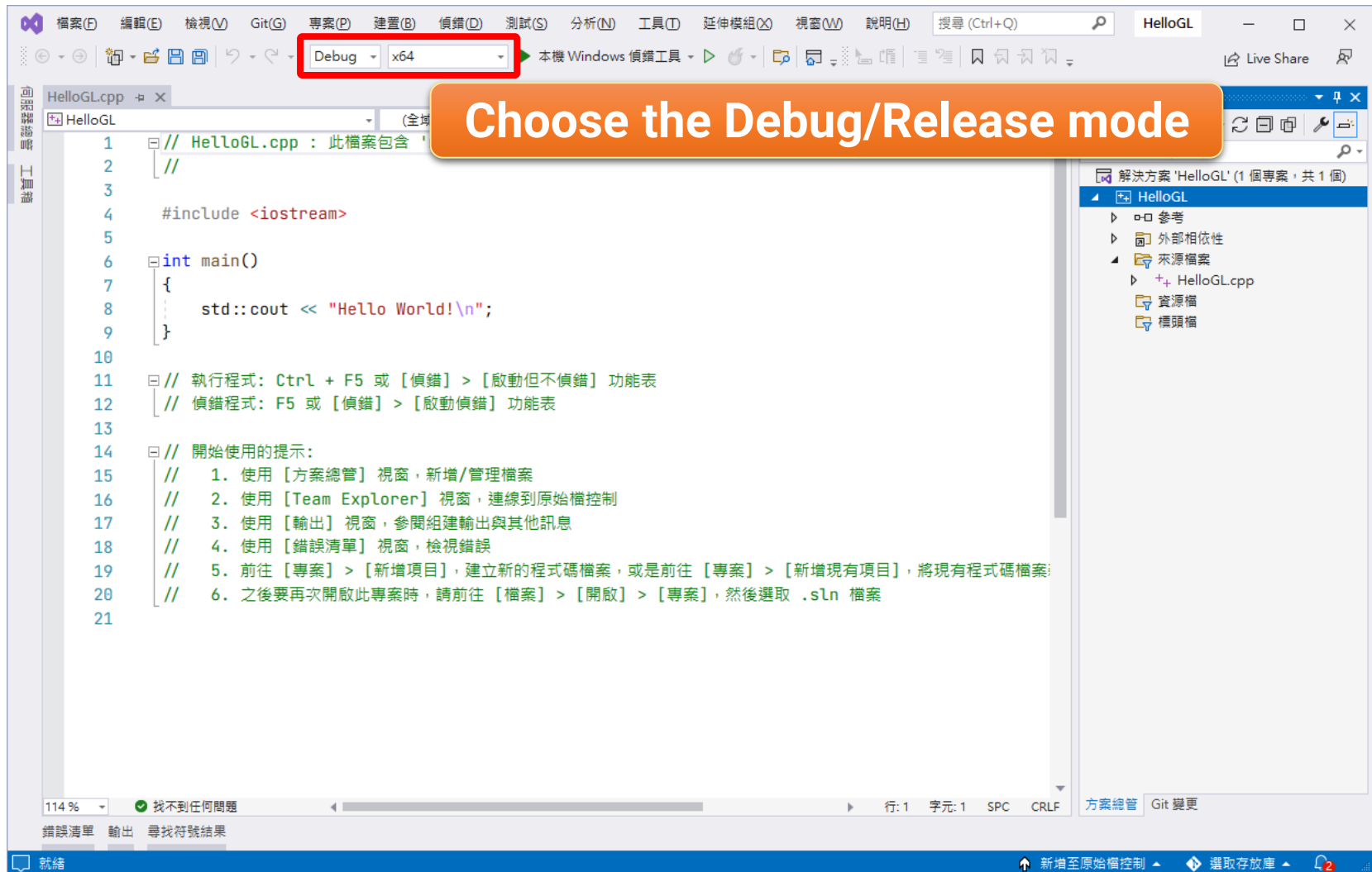
解決方案名稱(M) ⓘ

HelloGL

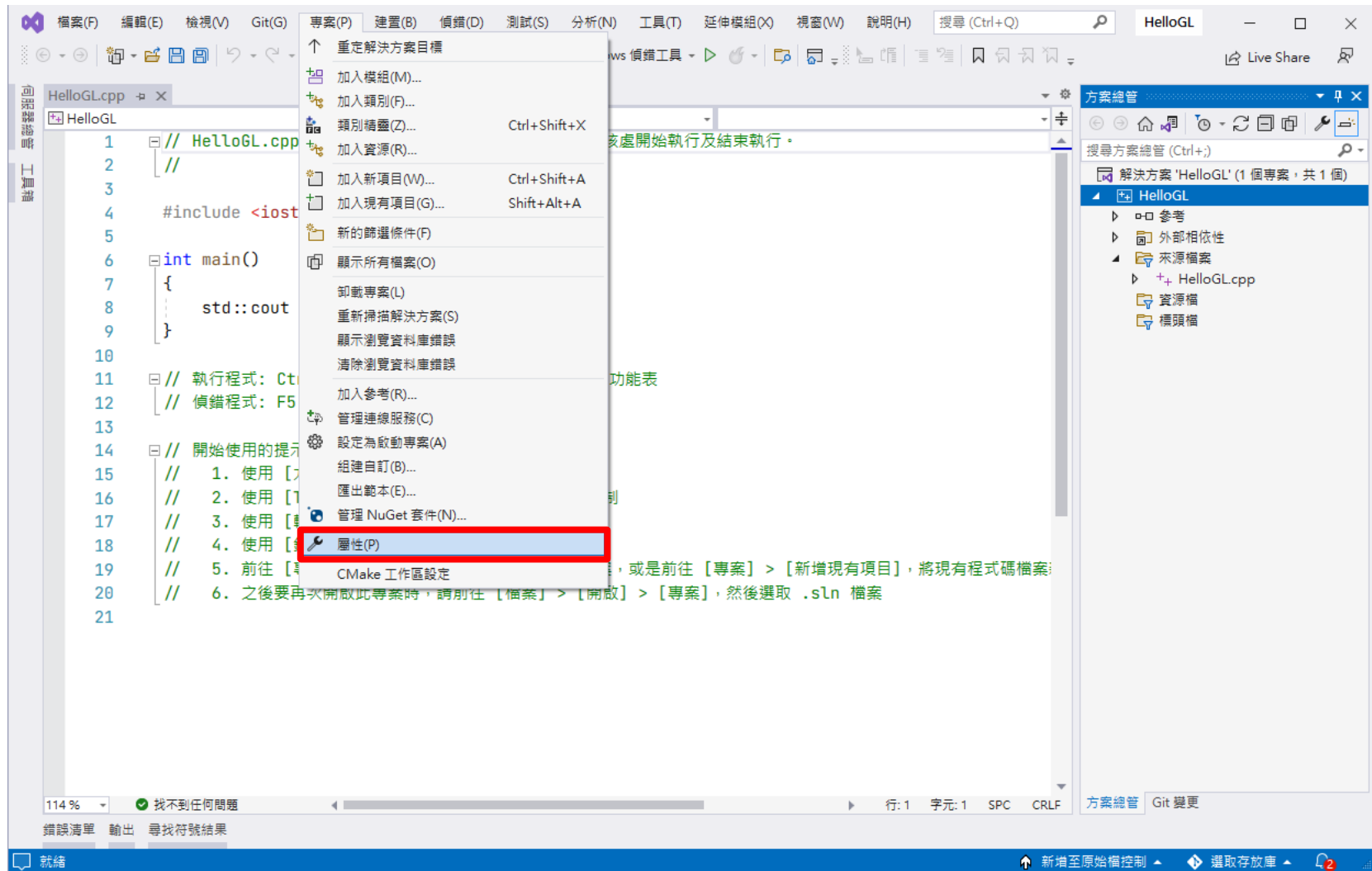
☐ 將解決方案與專案置於相同目錄中(D)

上一步(B) 建立(C)

Setup the Project in VS



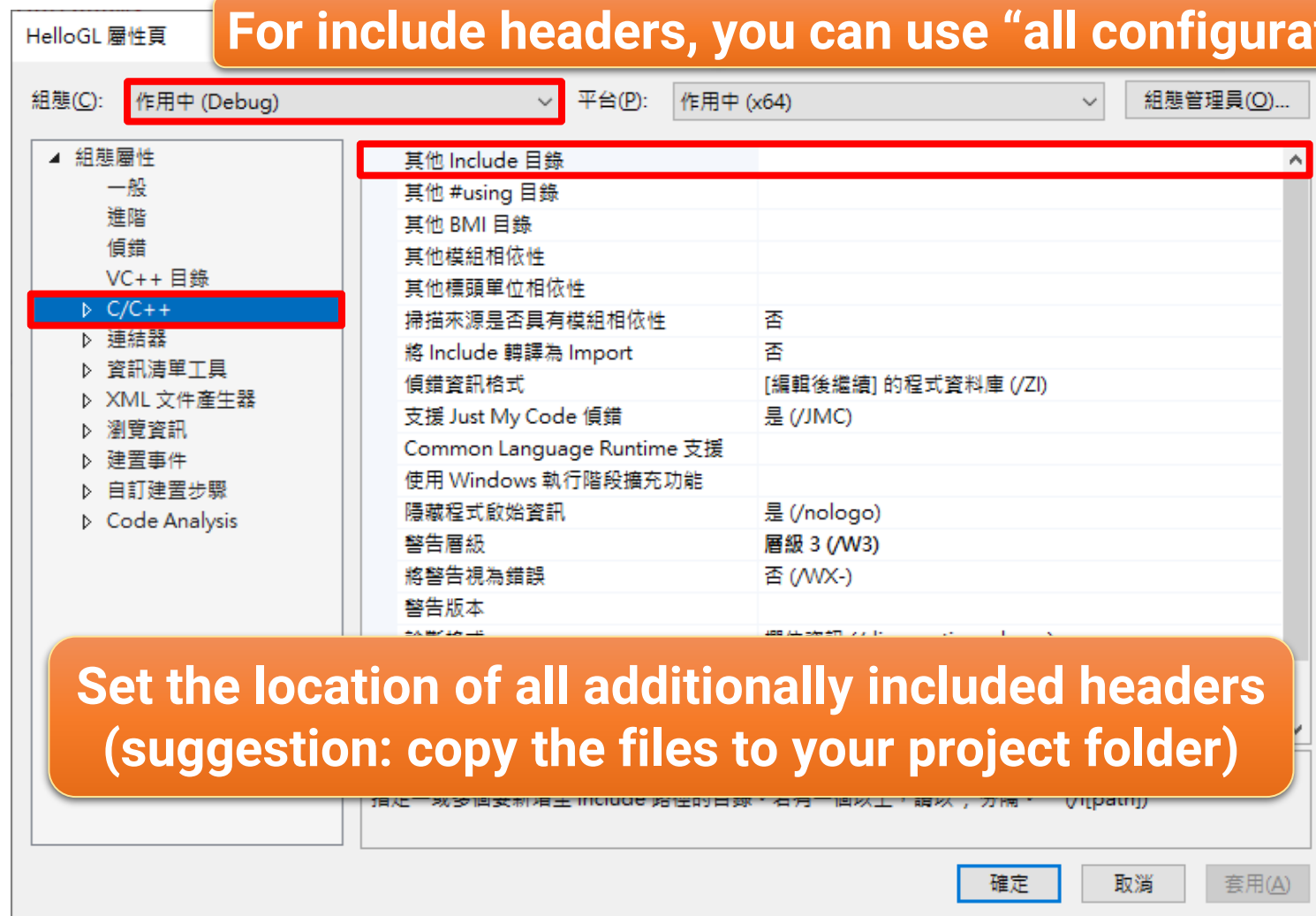
Setup the Project in VS



Setup the Project in VS (cont.)

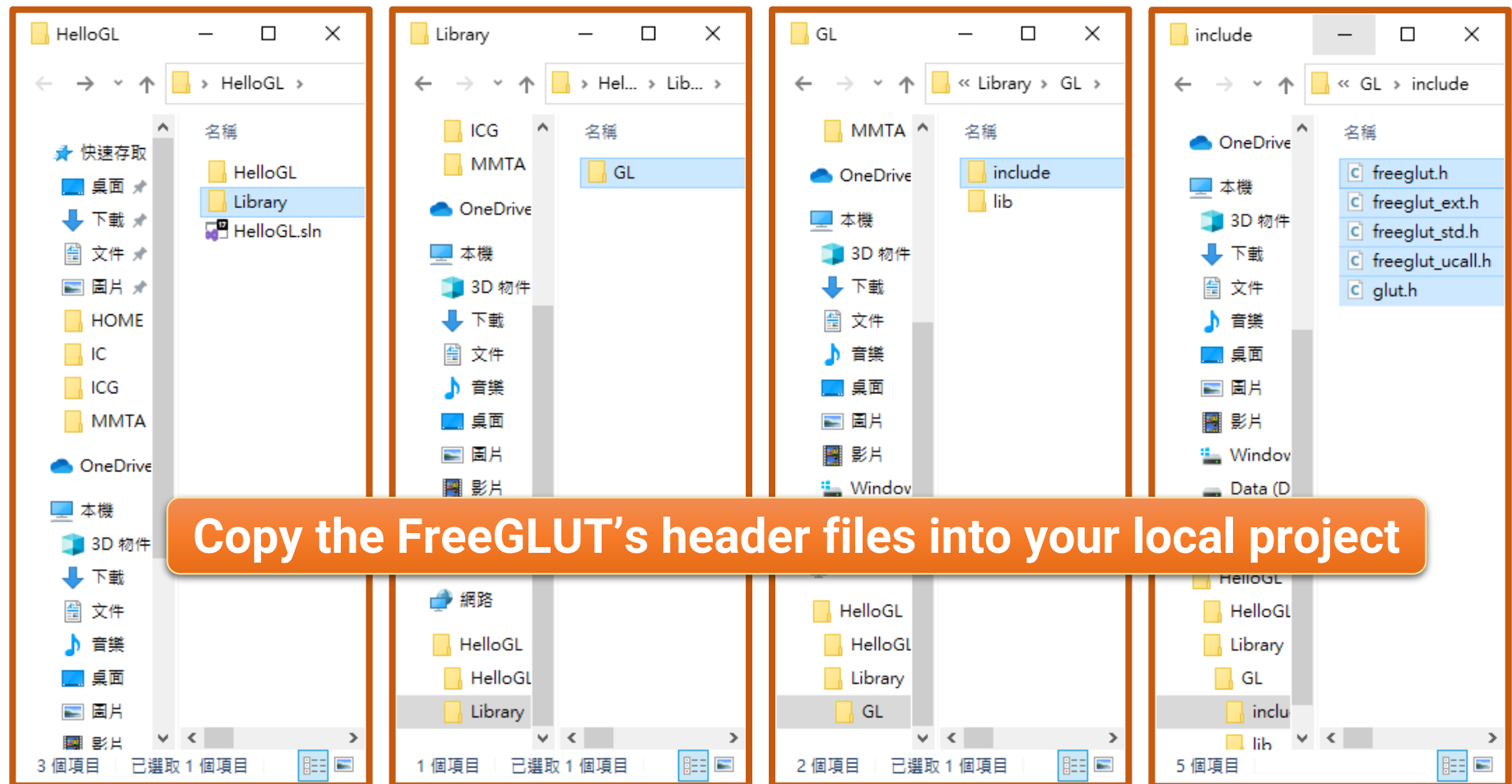
- A library usually consists of three types of files
 - **Header** (*.h, need to include)
 - **Static library** (*.lib, static linking, need to import during compile time)
 - **Dynamic library** (*.dll, dynamic linking, need to access during the run time)

Setup the Headers in VS

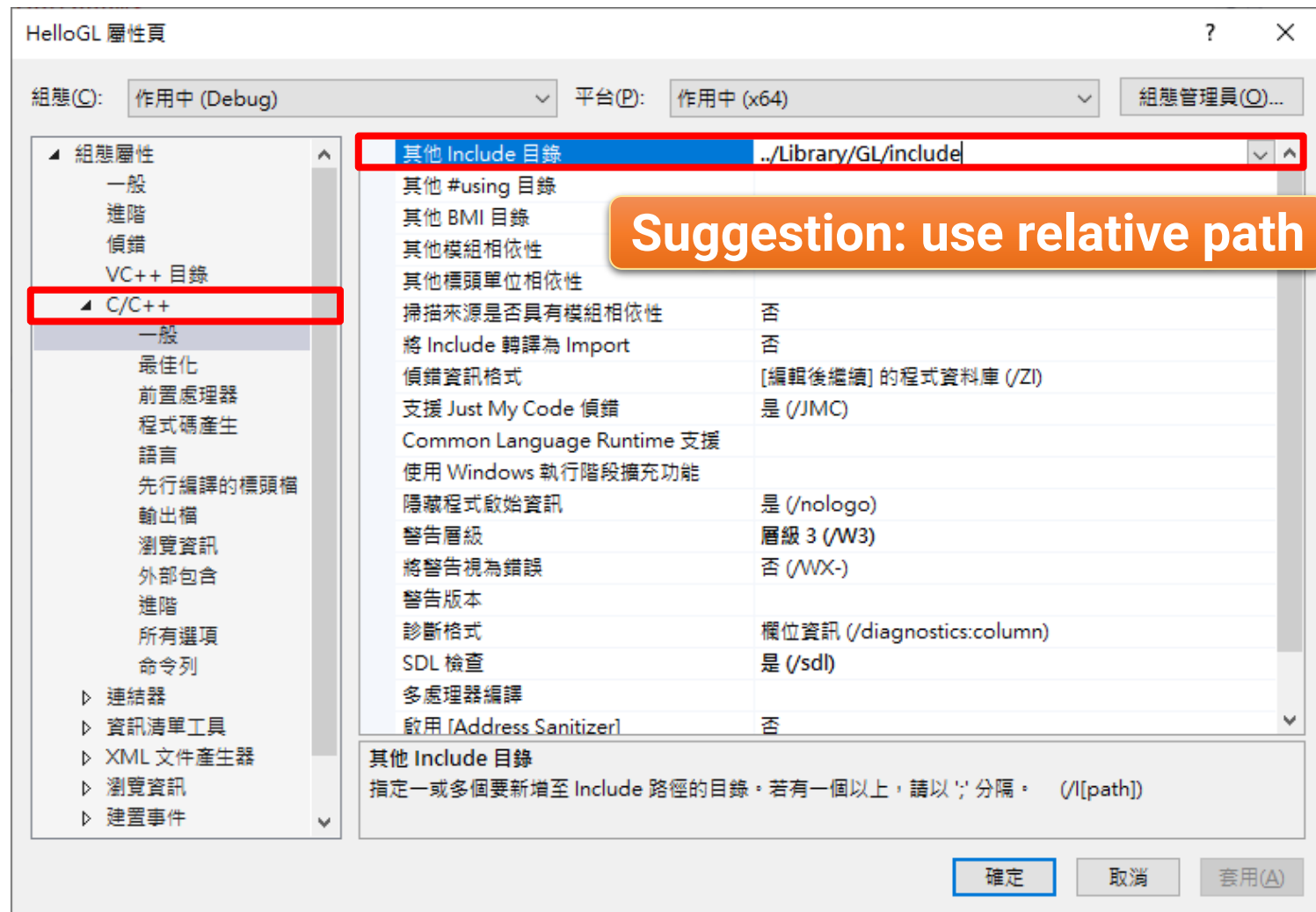


Setup the Headers in VS (cont.)

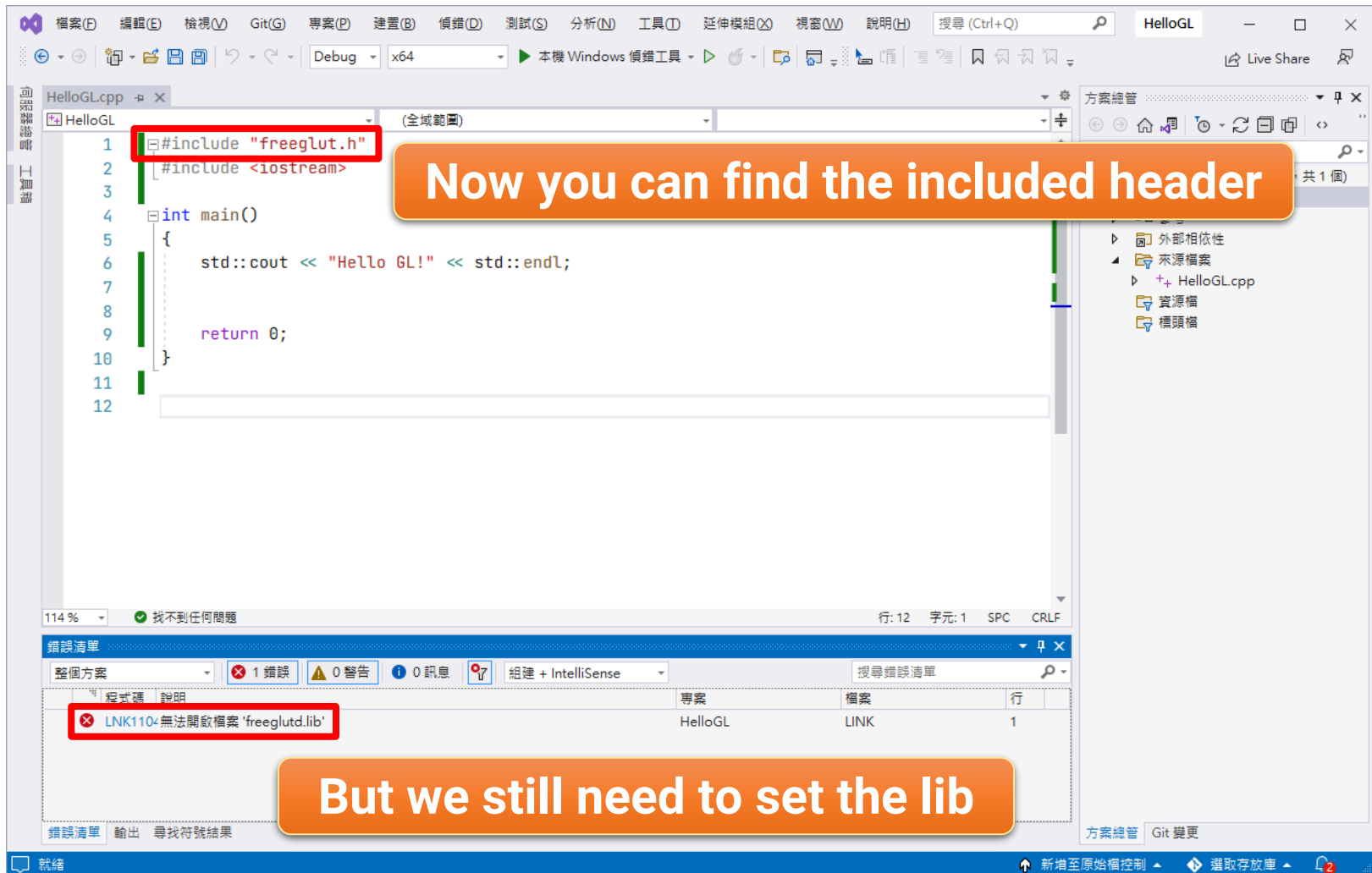
- My setting



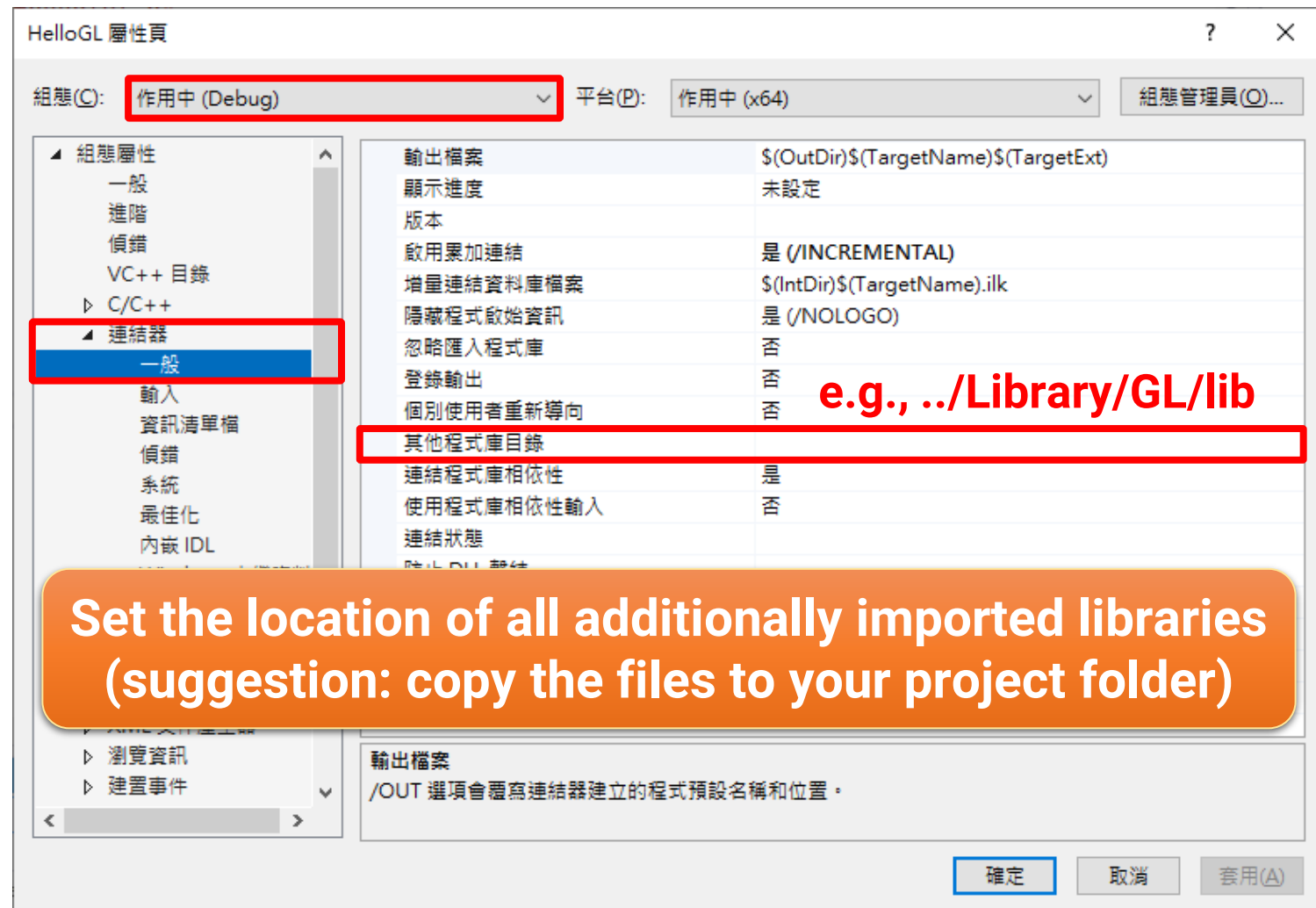
Setup the Headers in VS (cont.)



Setup the Headers in VS (cont.)

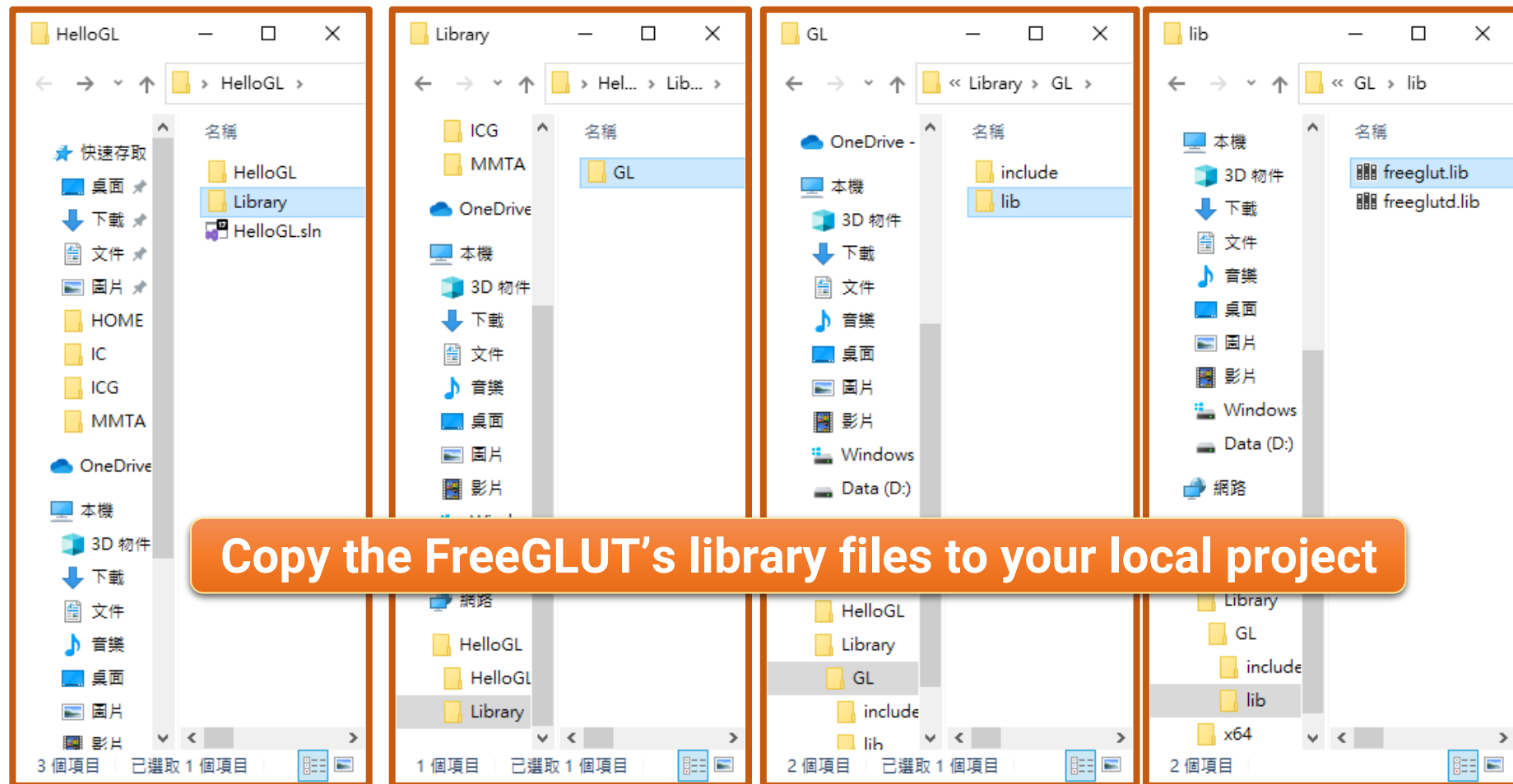


Setup the Static Library in VS

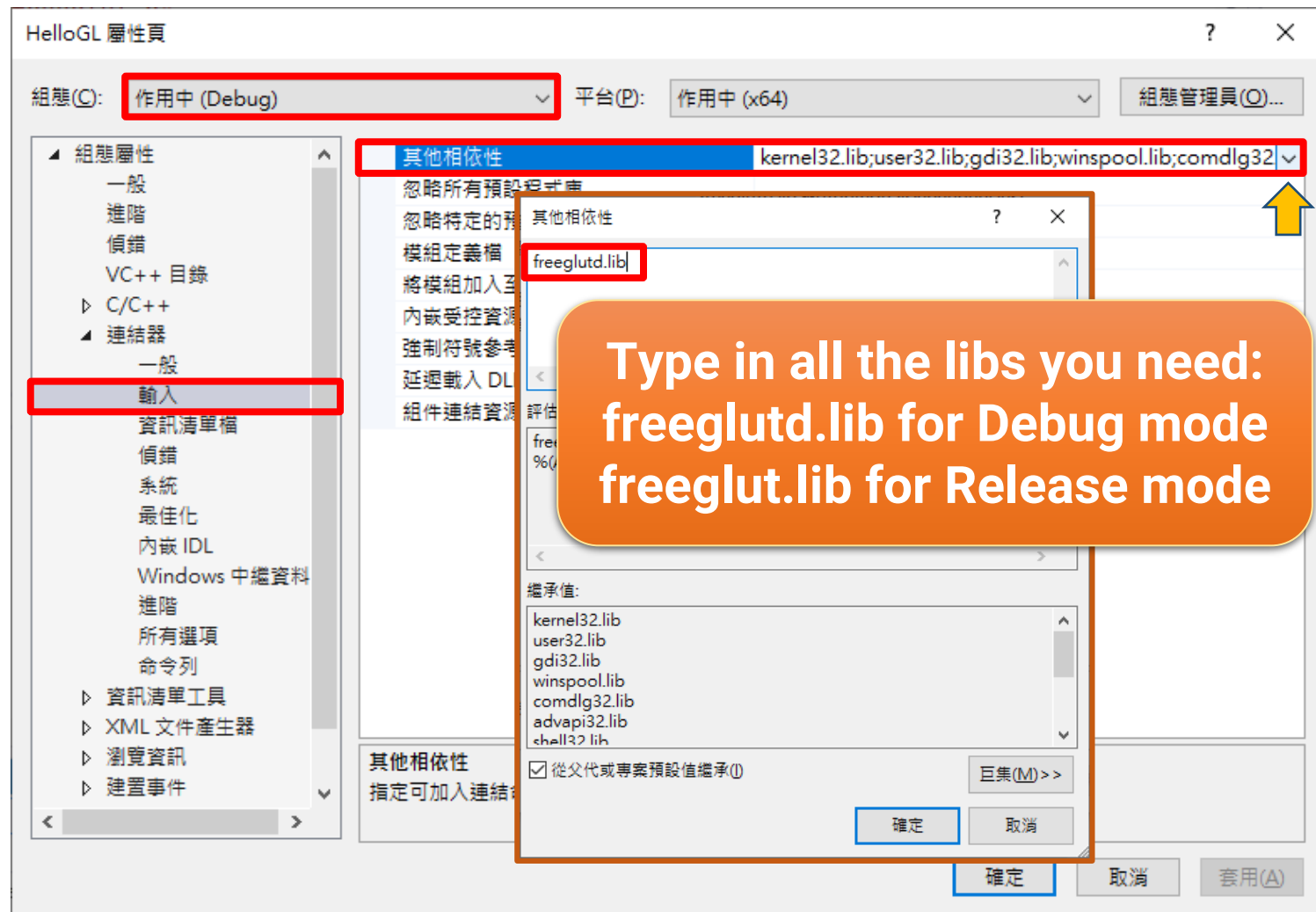


Setup the Static Library in VS (cont.)

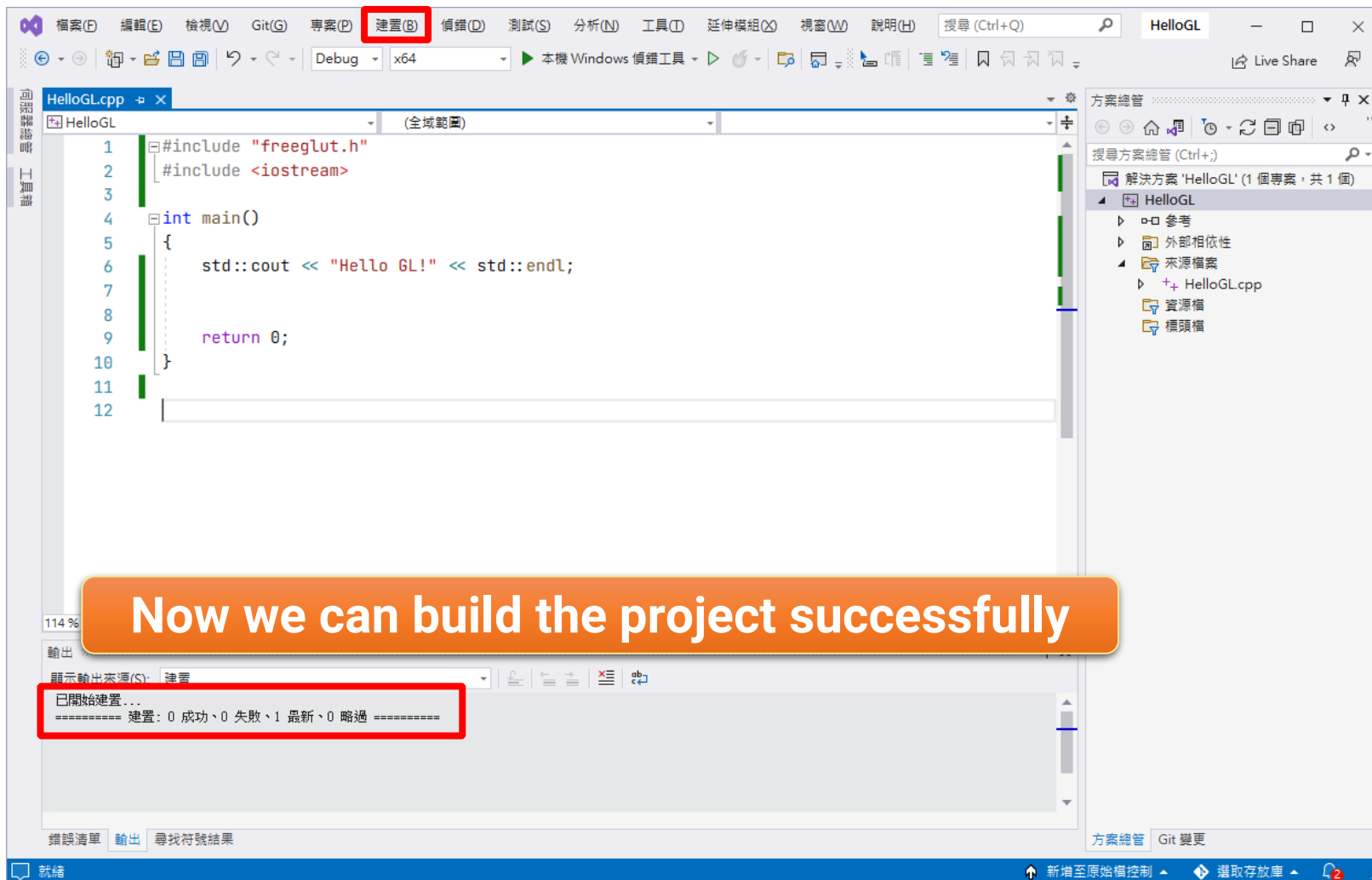
- My setting



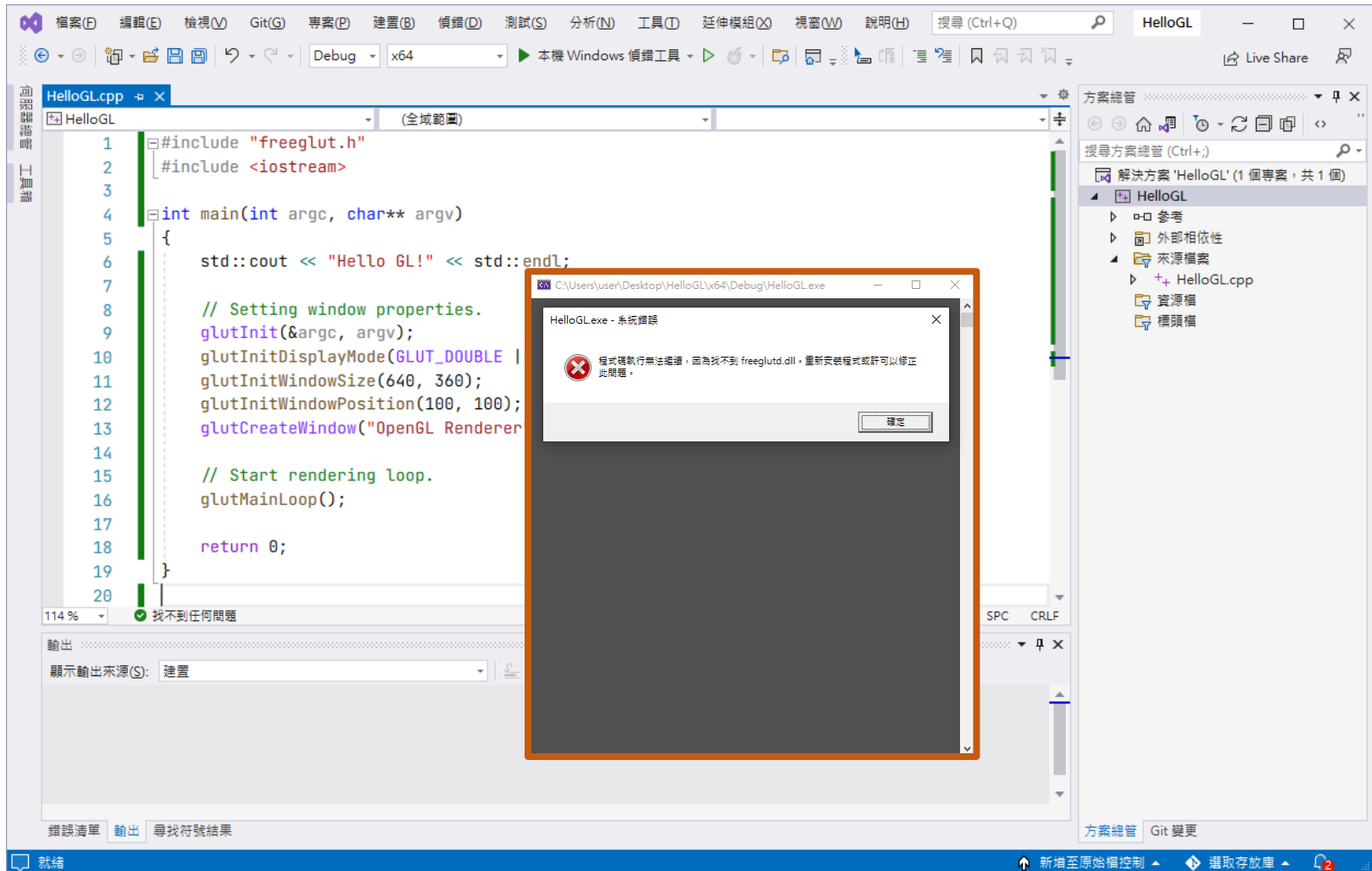
Setup the Static Library in VS (cont.)



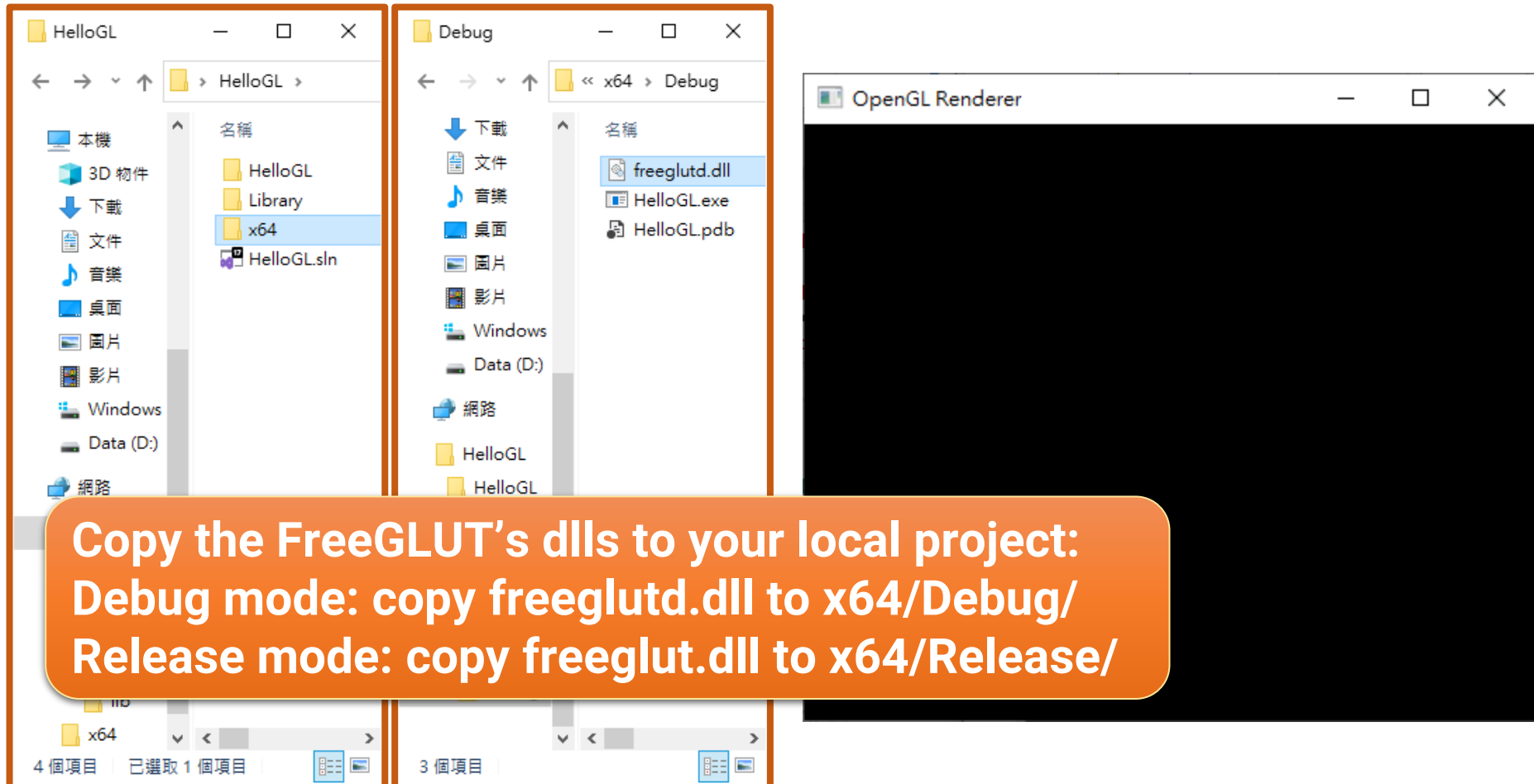
Setup the Static Library in VS (cont.)



Setup the Dynamic Library in VS



Setup the Dynamic Library in VS (cont.)



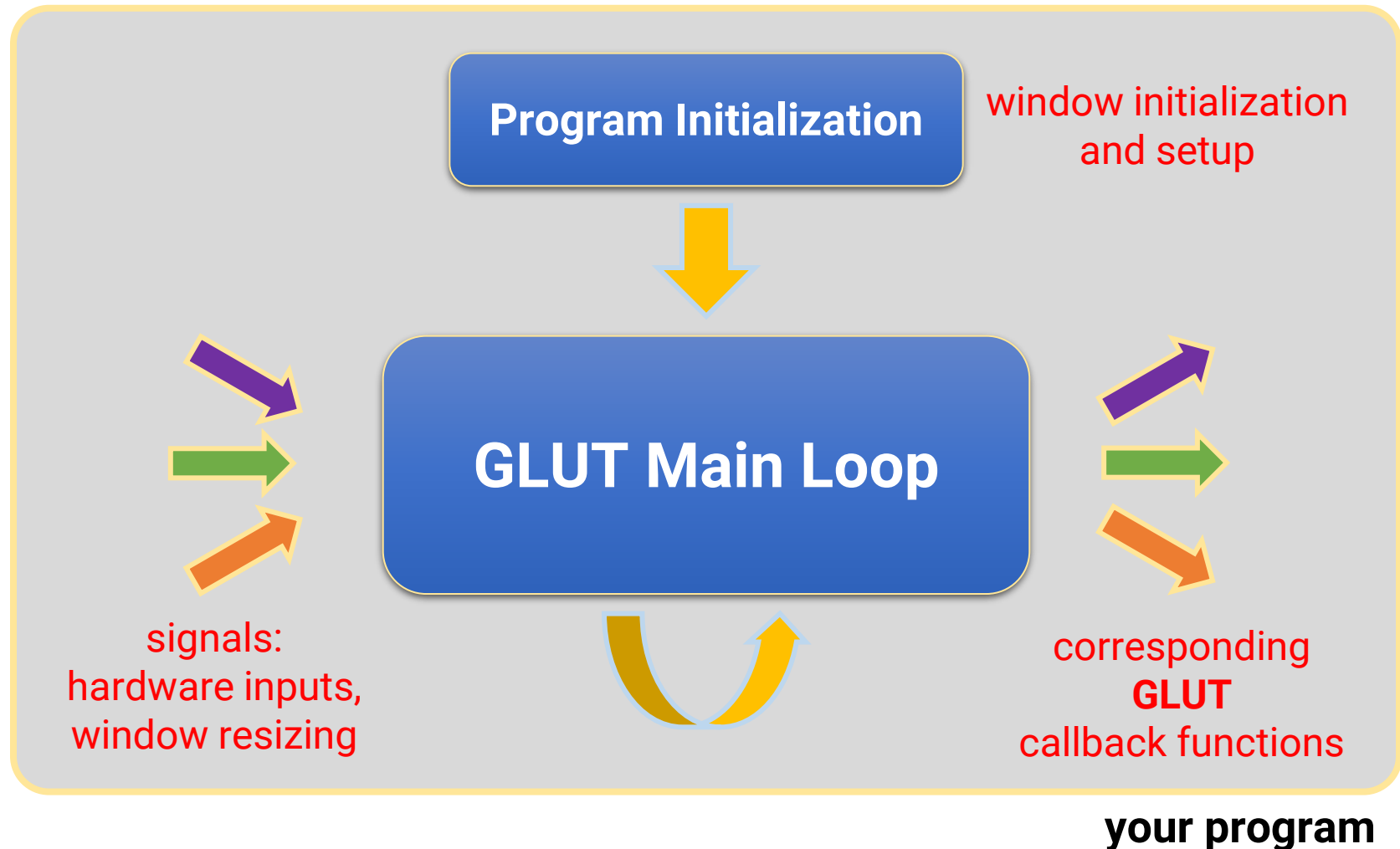
The screenshot displays the Visual Studio interface. On the left, the 'HelloGL' project is open, showing the 'x64' configuration. The 'Debug' folder is selected, and the 'freelutd.dll' file is highlighted. The 'OpenGL Renderer' window is visible on the right, showing a black canvas. An orange callout box provides instructions on where to copy the FreeGLUT DLLs.

Copy the FreeGLUT's dlls to your local project:
Debug mode: copy freelutd.dll to x64/Debug/
Release mode: copy freelut.dll to x64/Release/

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");
```

create the window
and set window
properties

```
    // Initialization.
```

```
    SetupRenderState();
```

do initialization
jobs

```
    // Register callback functions.
```

```
    glutDisplayFunc(RenderSceneCB);
```

```
    glutIdleFunc(RenderSceneCB);
```

```
    glutReshapeFunc(ReshapeCB);
```

```
    glutSpecialFunc(ProcessSpecialKeysCB);
```

```
    glutKeyboardFunc(ProcessKeysCB);
```

register callback
functions

```
    // Start rendering loop.
```

```
    glutMainLoop();
```

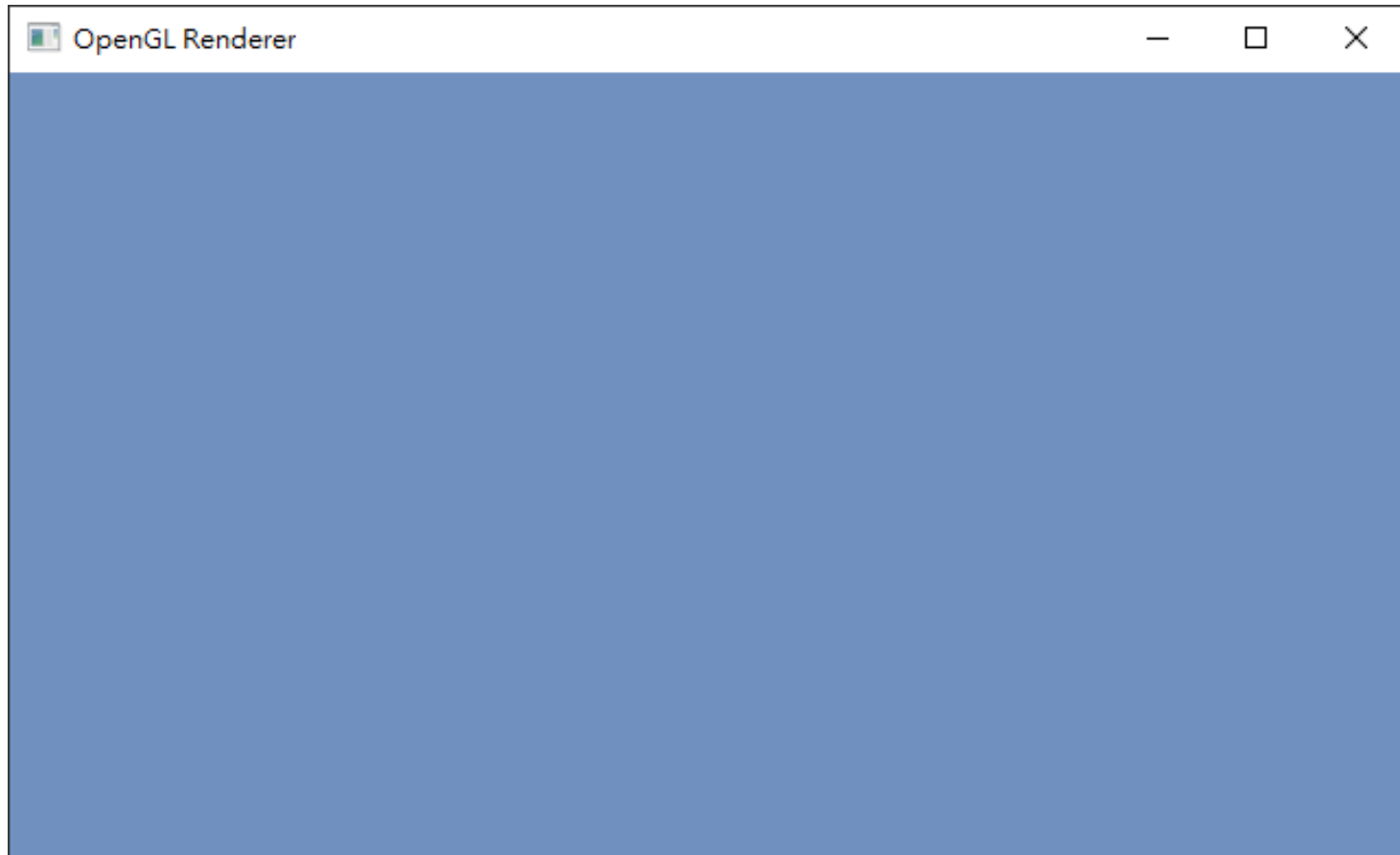
start the
main loop

```
    return 0;
```

```
}
```

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);
```

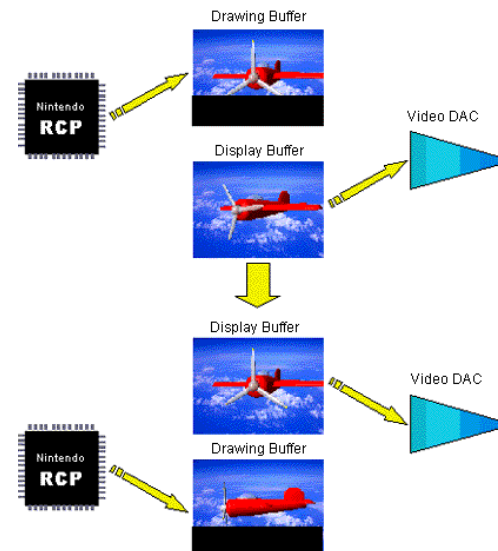
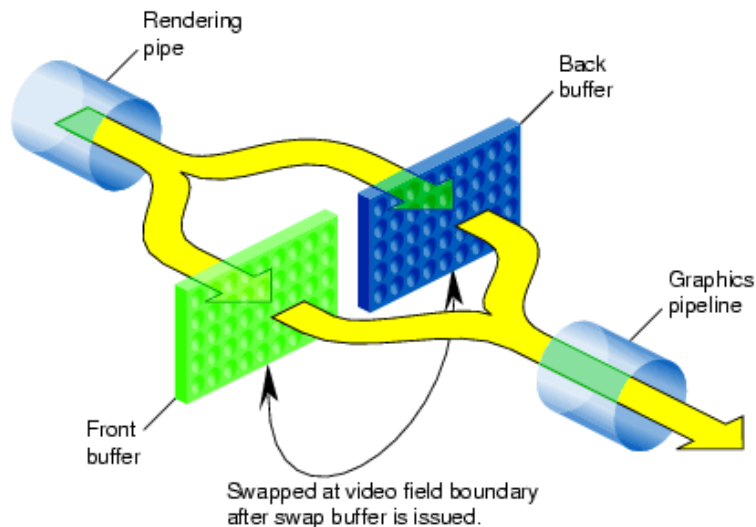
- void **glutInitDisplayMode**(unsigned int mode);
 - Set the initial display mode
 - <https://www.opengl.org/resources/libraries/glut/spec3/node12.html>

```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);  
                  double buffer  color buffer  enable depth buffer  
                           format
```

Double Buffers

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

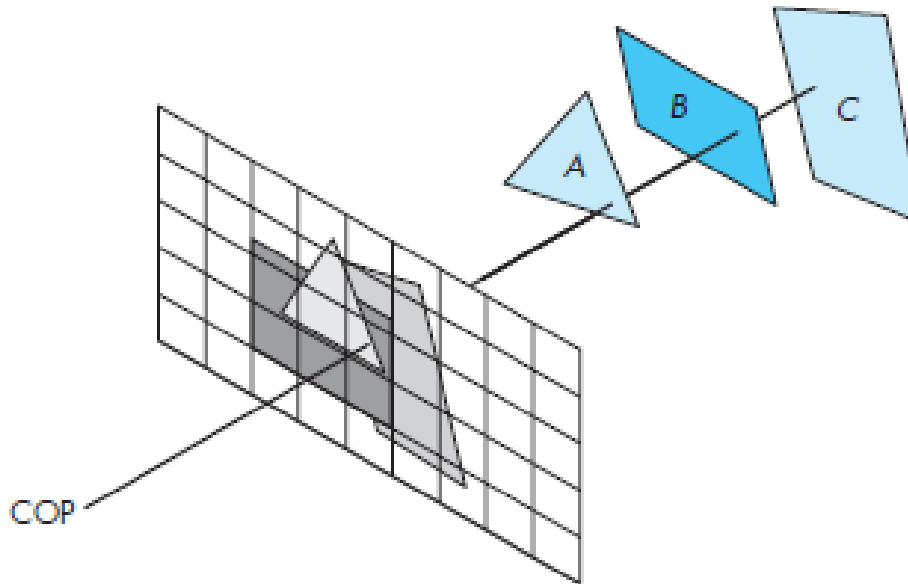
- Prevent artifacts due to potentially seeing parts of an incomplete frame (that is currently drawn)
 - Set the display mode to **GLUT_DOUBLE** in the **glutInitDisplayMode** function
 - Call **glutSwapBuffers** after rendering finished



Depth Buffer

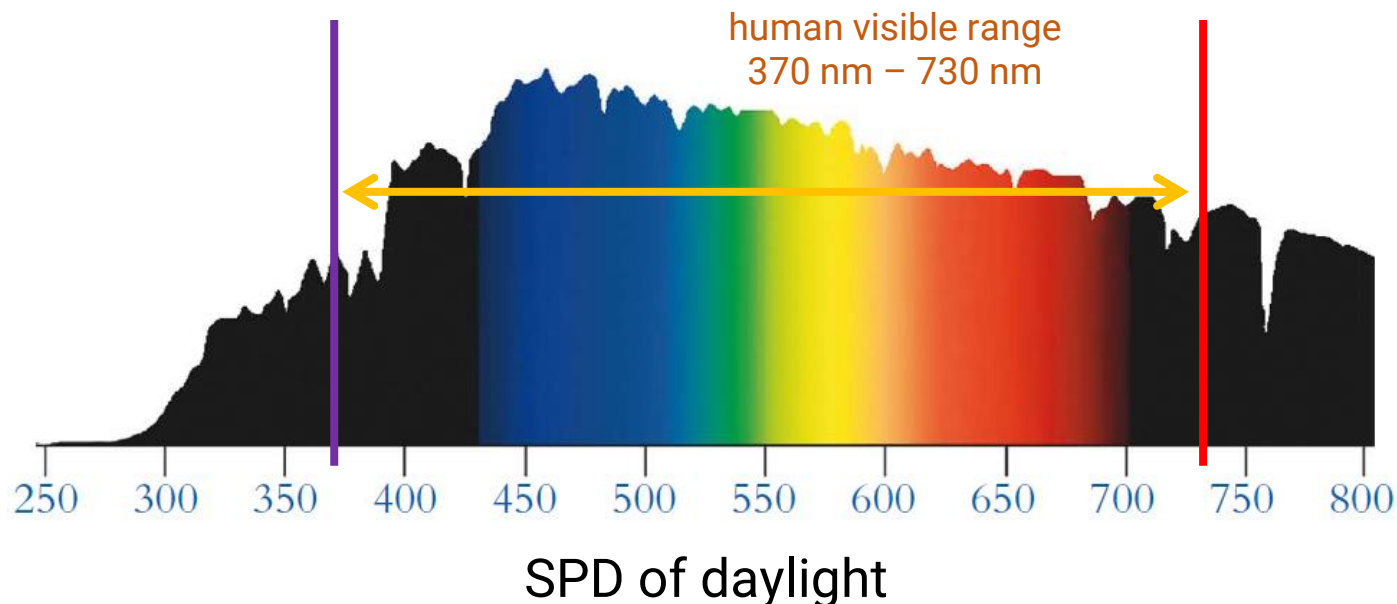
```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
```

- 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



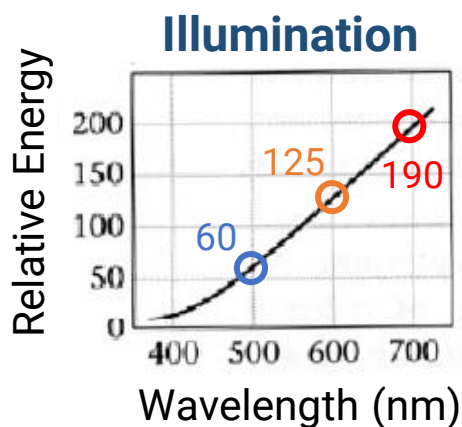
Color: Spectral Power Distribution (cont.)

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

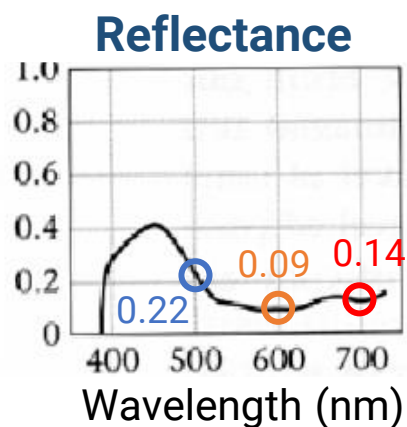


Color: Spectral Power Distribution (cont.)

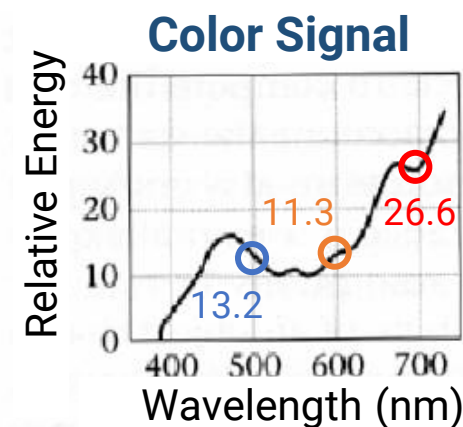
- Reflected color is the result of interaction of **light source spectrum** with **surface reflectance**



*



=



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.)

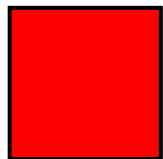
- For more details about tristimulus theory, please refer to the recording of my course, “Multimedia Technology and Application”
 - Course material link:
 - Part 1: <https://reurl.cc/OmQAEX>
 - Part 2: <https://reurl.cc/z5OZr6>
 - Part 3: <https://reurl.cc/axEnaZ>

RGB Color Model

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

(r, g, b),

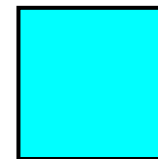
where r, g, b are the **amounts (proportion of the pure light)** of red, green, and blue light making up the color



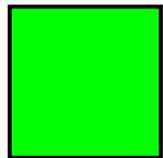
Red
(100%, 0%, 0%)



Black
(0%, 0%, 0%)



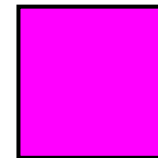
Cyan
(0%, 100%, 100%)



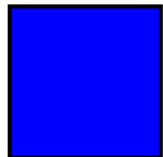
Green
(0%, 100%, 0%)



White
(100%, 100%, 100%)



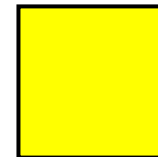
Magenta
(100%, 0%, 100%)



Blue
(0%, 0%, 100%)



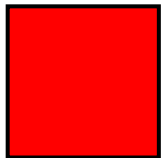
Gray
(50%, 50%, 50%)



Yellow
(100%, 100%, 0%)

RGB Color Model (cont.)

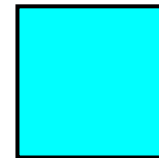
- In most applications, we use **8 bits** (1 byte) for each primary color, making 24 bits (3 bytes) in total
 - The range of each value falls within $[0, 255]$, making a total of $256 \times 256 \times 256 = 16777216$ different colors



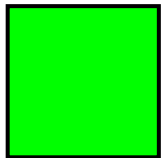
Red
(255, 0, 0)



Black
(0, 0, 0)



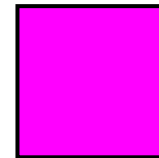
Cyan
(0, 255, 255)



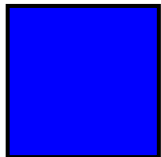
Green
(0, 255, 0)



White
(255, 255, 255)



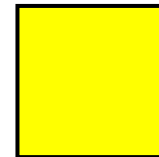
Magenta
(255, 0, 255)



Blue
(0, 0, 255)



Gray
(127, 127, 127)



Yellow
(255, 255, 0)


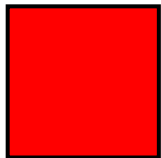
RGB Color Model (cont.)

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

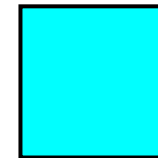
```
float clearColor[4] = {0.44f, 0.57f, 0.75f, 1.00f};
glClearColor(
    (GLclampf)(clearColor[0]),
    (GLclampf)(clearColor[1]),
    (GLclampf)(clearColor[2]),
    (GLclampf)(clearColor[3])
);
```

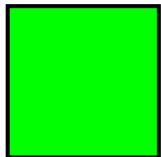
Red
(1.0f, 0.0f, 0.0f)



Black
(0.0f, 0.0f, 0.0f)



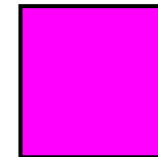
Cyan
(0.0f, 1.0f, 1.0f)



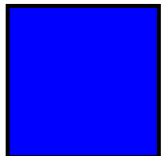
Green
(0.0f, 1.0f, 0.0f)



White
(1.0f, 1.0f, 1.0f)



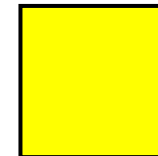
Magenta
(1.0f, 0.0f, 1.0f)



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) { ESC
        // Release memory allocation if needed.
        exit(0);
    }
}
```

clear the canvas (color buffer & depth buffer)

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

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);

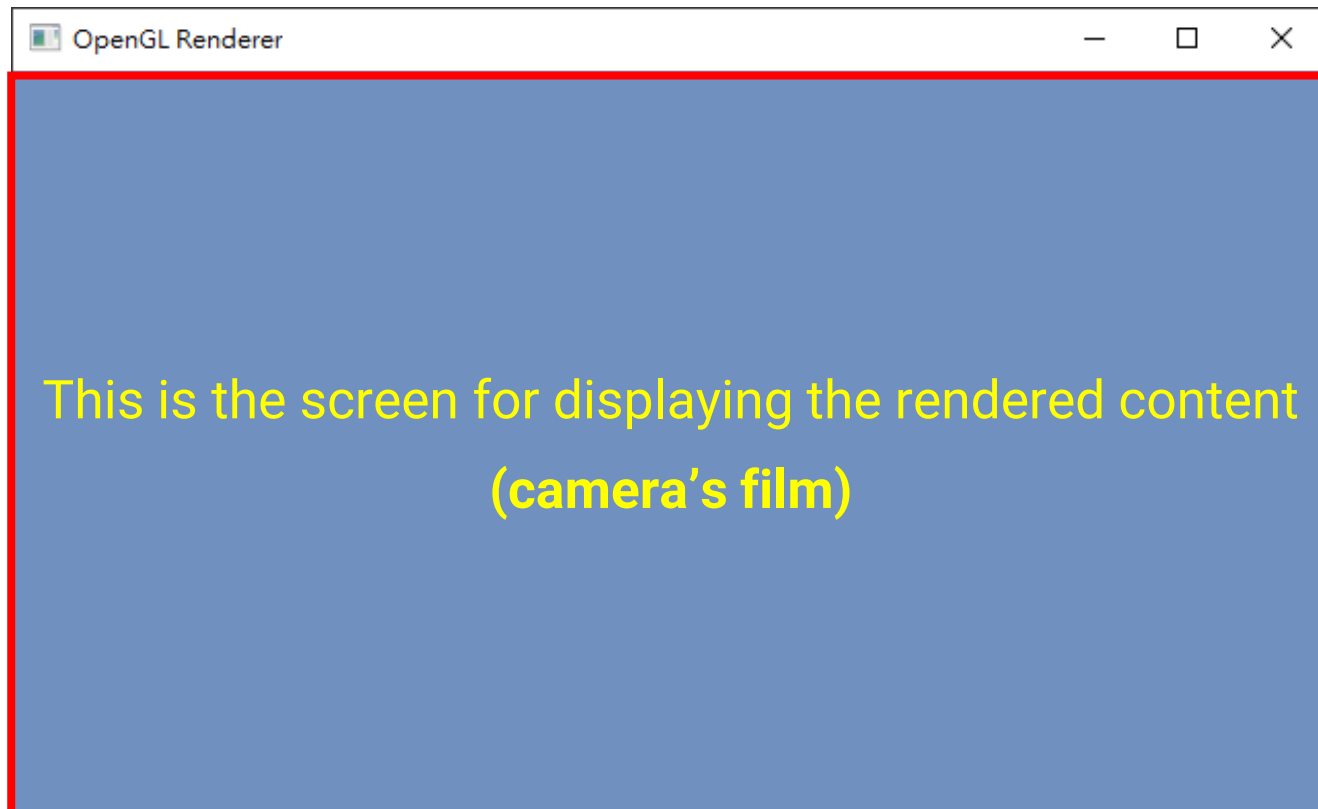
    // Start rendering loop.
    glutMainLoop();

    return 0;
}
```

start the
main loop

API: Start the Main Rendering Loop

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





Outline

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

FreeGLUT

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

The screenshot shows the GitHub repository for FreeGLUT. The repository is public and has 174 forks and 447 stars. The 'Code' dropdown menu is open, showing options to clone the repository using HTTPS or GitHub CLI, to open it with GitHub Desktop, and to download the source code as a ZIP file. The 'Download ZIP' option is highlighted with a red box. The repository's file list is visible on the left, showing folders like .github/workflows, altbuild, android, include/GL, progs, and src, as well as files like .gitignore and AUTHORS. The right sidebar contains information about the repository, including a link to the sourceforge.net page, a README, license, stars, watching, forks, and releases.

Product ▾ Solutions ▾ Open Source ▾ Pricing

Search / Sign in Sign up

FreeGLUTProject / freeglut Public

Notifications Fork 174 Star 447

<> Code Issues 13 Pull requests 7 Actions Projects Security Insights

master 6 branches 31 tags Go to file Code ▾

Clone ?

HTTPS GitHub CLI

<https://github.com/FreeGLUTProject/freeglut>

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

About

Free implementation of the OpenGL Utility Toolkit (GLUT)

freeglut.sourceforge.net

Readme View license 447 stars 43 watching 174 forks

Releases 10

freeglut 3.2.2 Lat on 9 Mar

Continue >>

FreeGLUT (cont.)

- Unzip the package

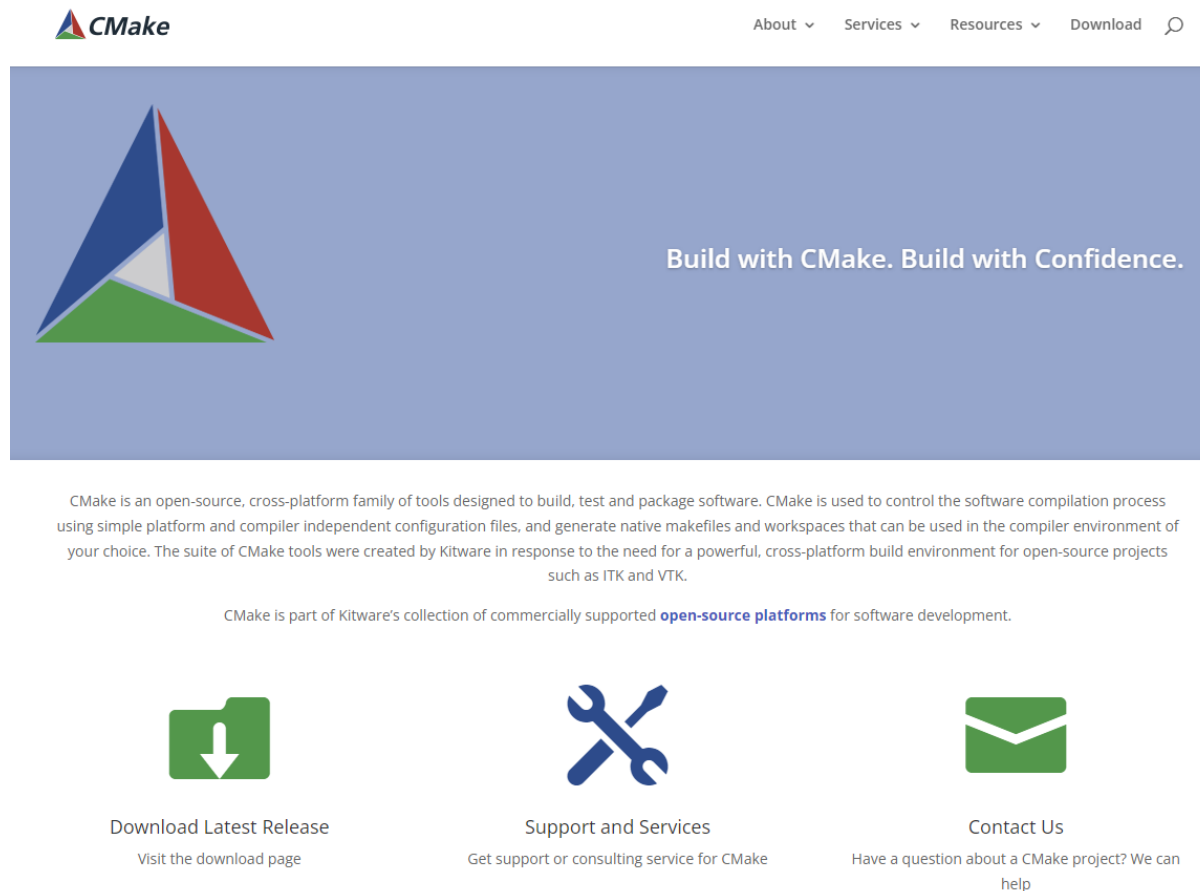
.github	2022/9/11 上午 07:31	檔案資料夾	
altbuild	2022/9/11 上午 07:31	檔案資料夾	
android	2022/9/11 上午 07:31	檔案資料夾	
include	2022/9/11 上午 07:31	檔案資料夾	
progs	2022/9/11 上午 07:31	檔案資料夾	
src	2022/9/11 上午 07:31	檔案資料夾	
.gitignore	2022/9/11 上午 07:31	文字文件	1 KB
android_toolchain.cmake	2022/9/11 上午 07:31	CMake 來源檔案	1 KB
AUTHORS	2022/9/11 上午 07:31	檔案	2 KB
blackberry.toolchain.cmake	2022/9/11 上午 07:31	CMake 來源檔案	10 KB
ChangeLog	2022/9/11 上午 07:31	檔案	163 KB
CMakeLists.txt	2022/9/11 上午 07:31	文字文件	24 KB
config.h.in	2022/9/11 上午 07:31	IN 檔案	1 KB
COPYING	2022/9/11 上午 07:31	檔案	2 KB
freeglut.pc.in	2022/9/11 上午 07:31	IN 檔案	1 KB
freeglut.rc.in	2022/9/11 上午 07:31	IN 檔案	2 KB
FreeGLUTConfig.cmake.in	2022/9/11 上午 07:31	IN 檔案	1 KB
mingw_cross_toolchain.cmake	2022/9/11 上午 07:31	CMake 來源檔案	1 KB
README.android	2022/9/11 上午 07:31	ANDROID 檔案	1 KB
README.blackberry	2022/9/11 上午 07:31	BLACKBERRY 檔案	2 KB
README.cmake	2022/9/11 上午 07:31	CMake 來源檔案	5 KB
README.cygwin_mingw	2022/9/11 上午 07:31	CYGWIN_MINGW...	8 KB
README.macosx	2022/9/11 上午 07:31	MACOSX 檔案	2 KB
README.md	2022/9/11 上午 07:31	Markdown 來源...	4 KB
README.mingw_cross	2022/9/11 上午 07:31	MINGW_CROSS ...	2 KB
README.win32	2022/9/11 上午 07:31	WIN32 檔案	5 KB

Build the source code
using **CMake**

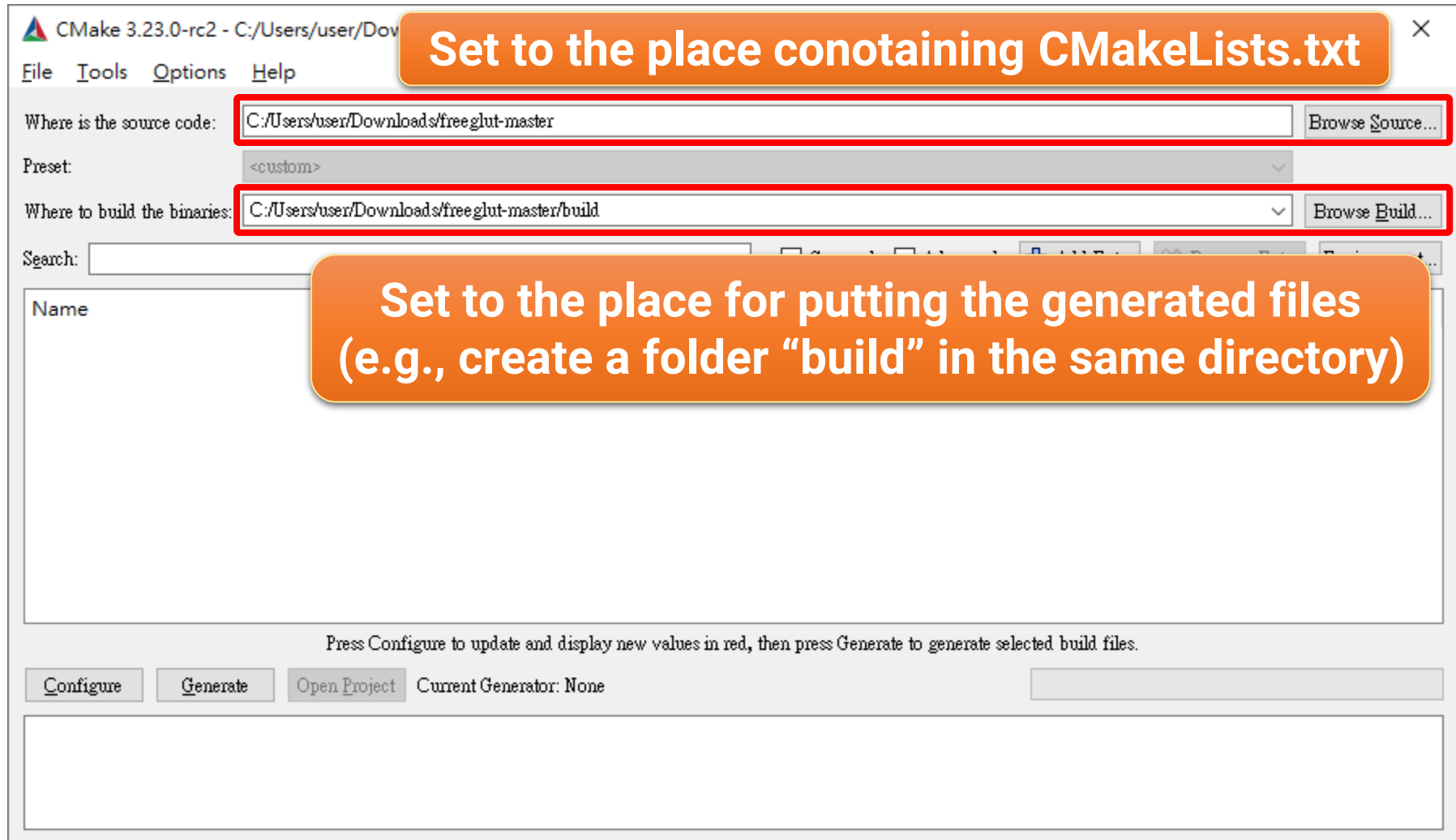


CMake

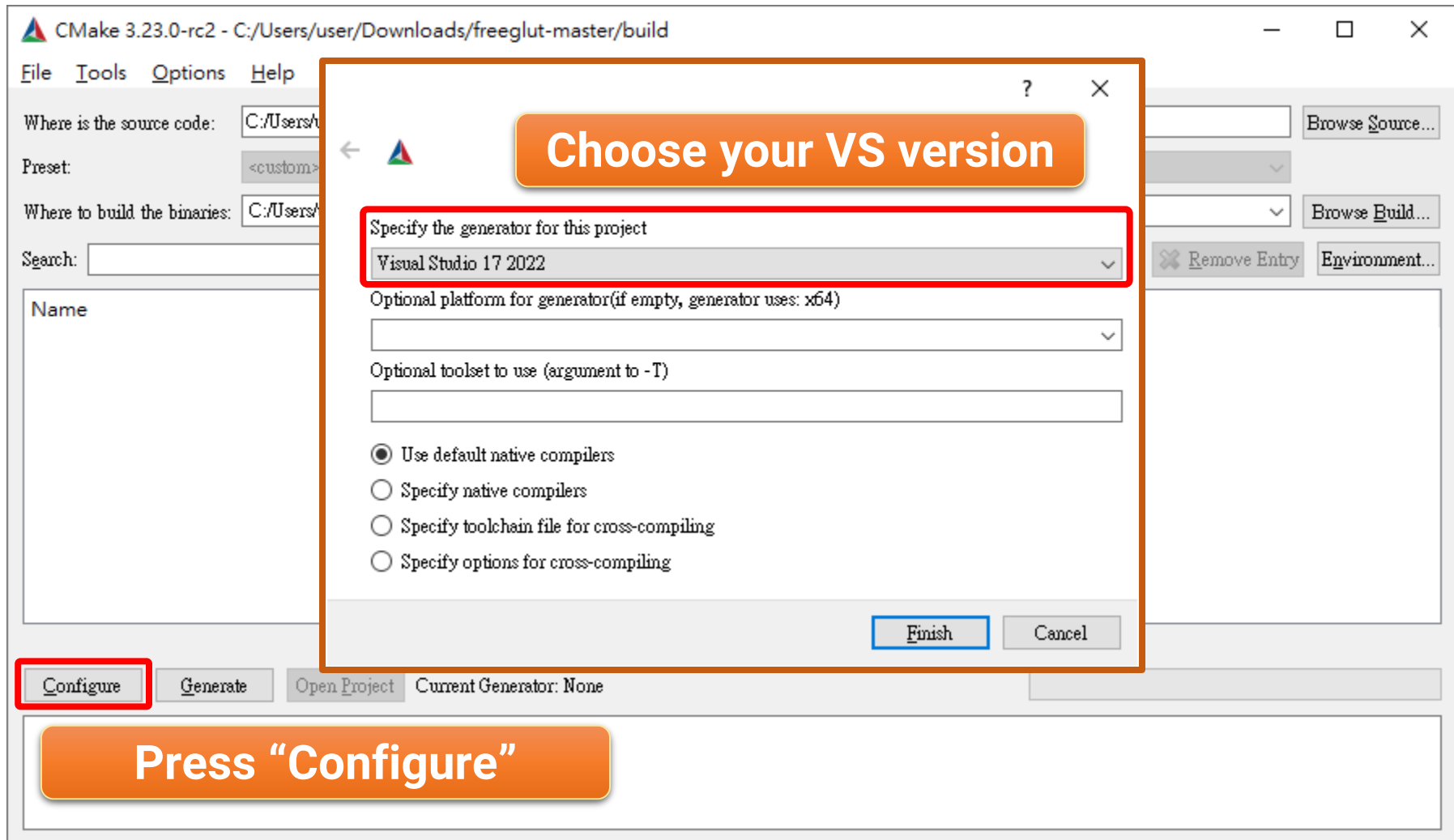
- Download and install CMake: <https://cmake.org/>



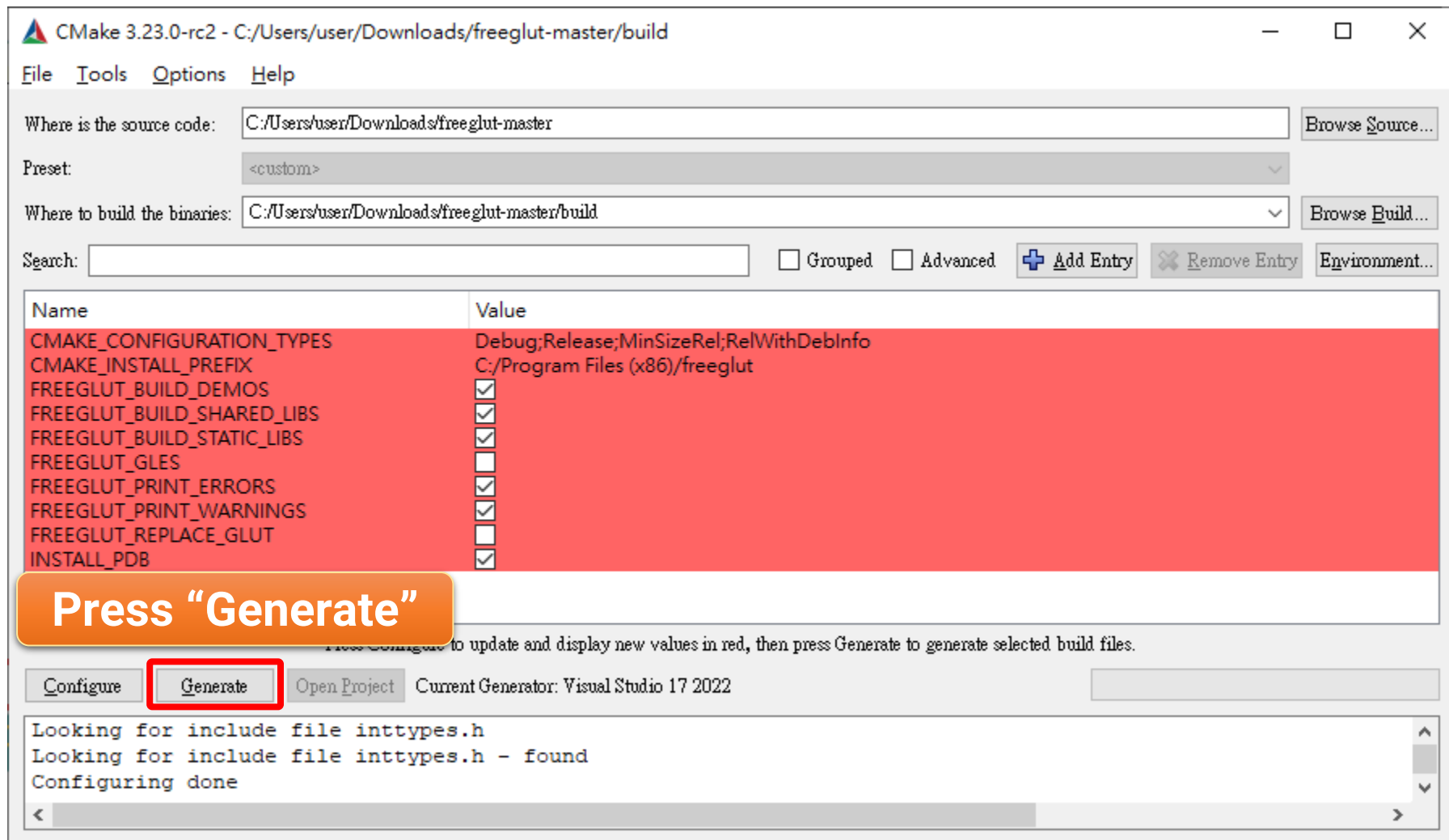
Setup CMake for Building FreeGLUT



Configuration



Generate VS Project



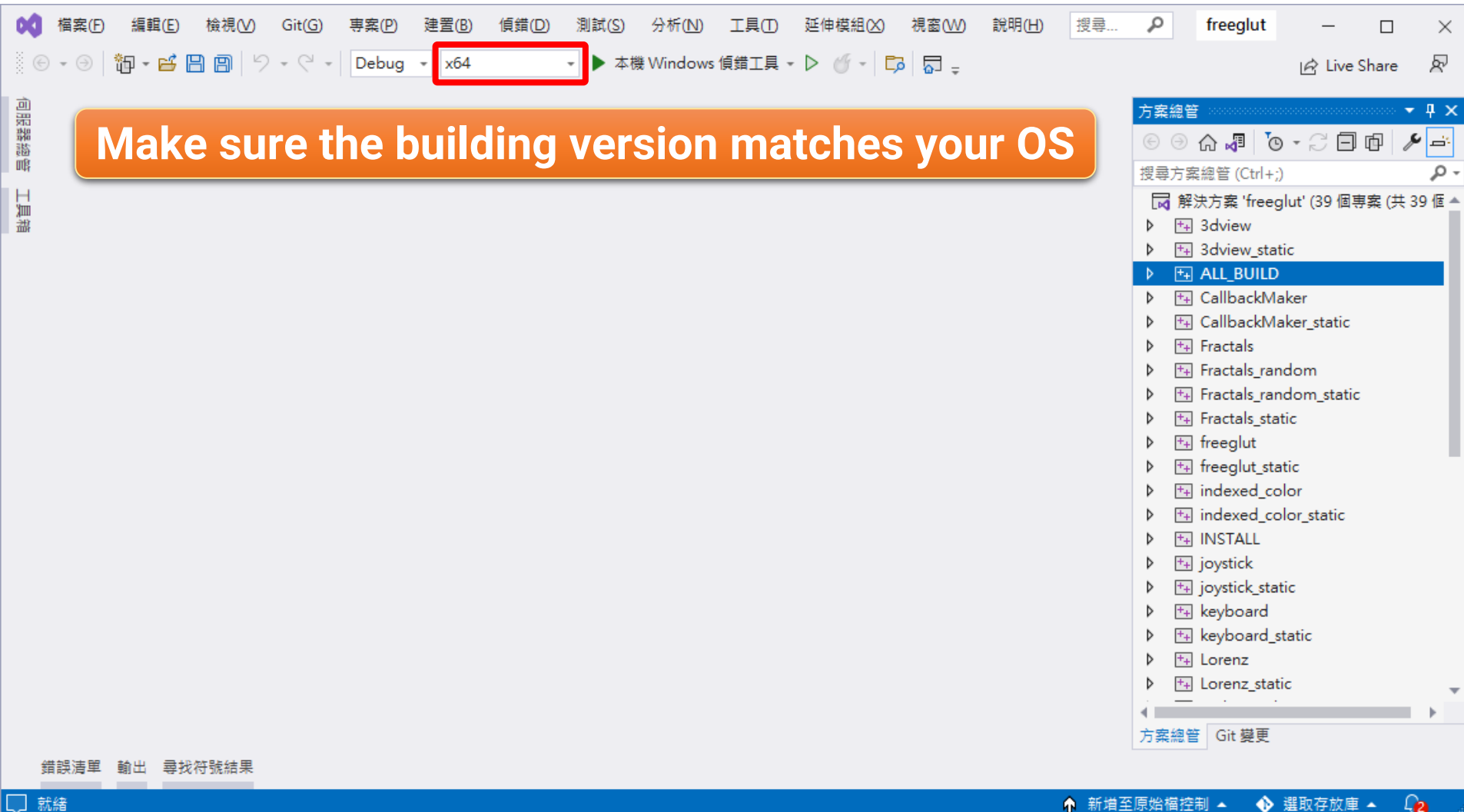
Examine VS Project

-master > build

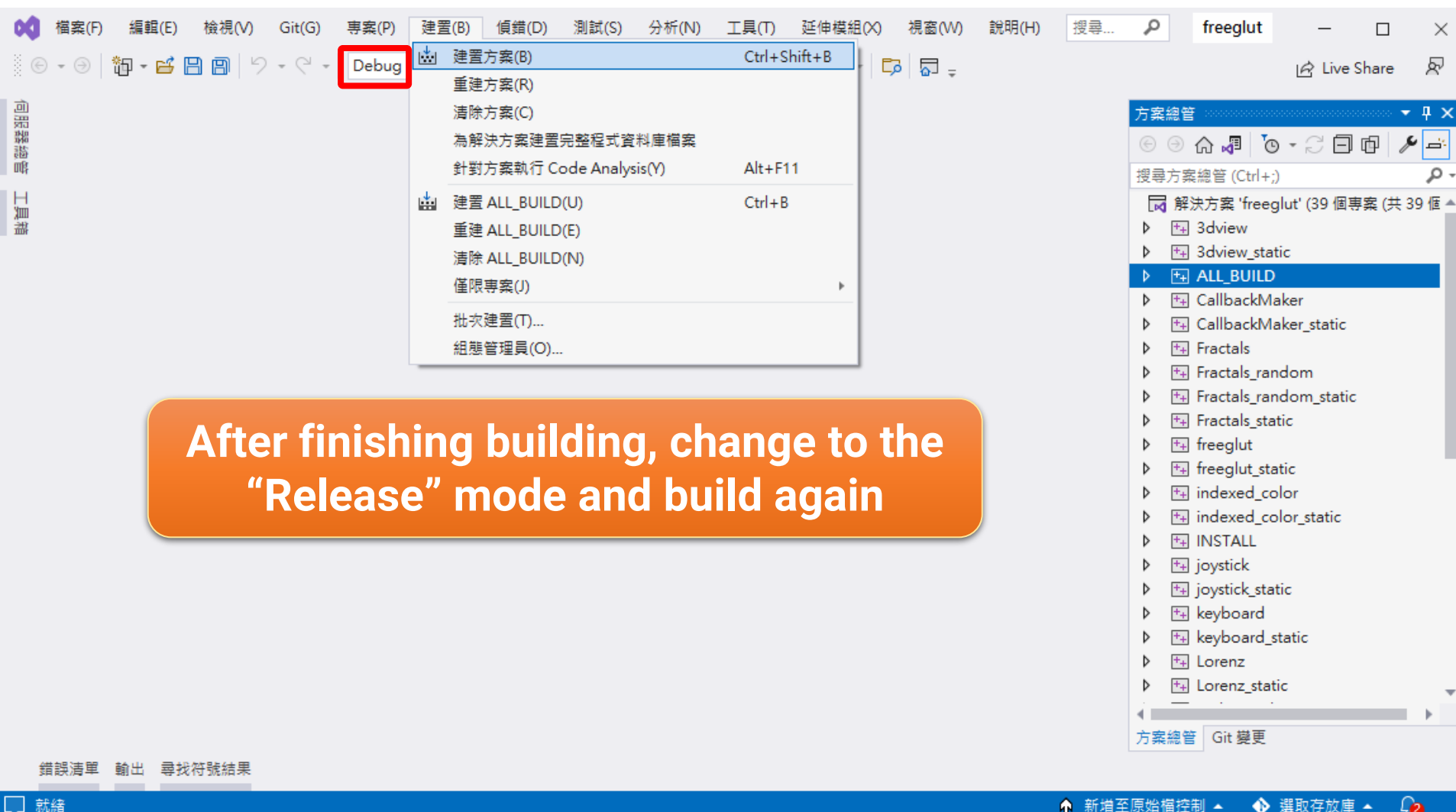
搜尋 build

名稱	修改日期	類型	大小
Fractals_random.vcxproj.filters	2022/9/14 下午 03:47	VC++ Project Filt...	1 KB
Fractals_random_static.vcxproj	2022/9/14 下午 03:47	VC++ Project	60 KB
Fractals_random_static.vcxproj.filters	2022/9/14 下午 03:47	VC++ Project Filt...	1 KB
Fractals_static.vcxproj	2022/9/14 下午 03:47	VC++ Project	60 KB
Fractals_static.vcxproj.filters	2022/9/14 下午 03:47	VC++ Project Filt...	1 KB
freelut.pc	2022/9/14 下午 03:46	PC 檔案	1 KB
freelut.rc	2022/9/14 下午 03:46	Resource Script	2 KB
freelut.sln	2022/9/14 下午 03:47	Visual Studio Sol...	43 KB
freelut.vcxproj	2022/9/14 下午 03:47	VC++ Project	64 KB
freelut.vcxproj.filters	2022/9/14 下午 03:47	VC++ Project Filt...	8 KB
freelut_static.vcxproj	2022/9/14 下午 03:47	VC++ Project	59 KB
freelut_static.vcxproj.filters	2022/9/14 下午 03:47	VC++ Project Filt...	8 KB
indexed_color.vcxproj	2022/9/14 下午 03:47	VC++ Project	59 KB
indexed_color.vcxproj.filters	2022/9/14 下午 03:47	VC++ Project Filt...	1 KB
indexed_color_static.vcxproj	2022/9/14 下午 03:47	VC++ Project	60 KB
indexed_color_static.vcxproj.filters	2022/9/14 下午 03:47	VC++ Project Filt...	1 KB
INSTALL.vcxproj	2022/9/14 下午 03:47	VC++ Project	10 KB
INSTALL.vcxproj.filters	2022/9/14 下午 03:47	VC++ Project Filt...	1 KB
joystick.vcxproj	2022/9/14 下午 03:47	VC++ Project	59 KB
joystick.vcxproj.filters	2022/9/14 下午 03:47	VC++ Project Filt...	1 KB
joystick_static.vcxproj	2022/9/14 下午 03:47	VC++ Project	60 KB
joystick_static.vcxproj.filters	2022/9/14 下午 03:47	VC++ Project Filt...	1 KB
keyboard.vcxproj	2022/9/14 下午 03:47	VC++ Project	59 KB
keyboard.vcxproj.filters	2022/9/14 下午 03:47	VC++ Project Filt...	1 KB
keyboard_static.vcxproj	2022/9/14 下午 03:47	VC++ Project	60 KB
keyboard_static.vcxproj.filters	2022/9/14 下午 03:47	VC++ Project Filt...	1 KB

Open Solution with Visual Studio



Debug/Release Build



The screenshot shows the Visual Studio interface. The 'Build' menu is open, and the 'Debug' option is highlighted with a red box. The 'Solution Explorer' on the right shows the project structure for 'freelut', with 'ALL_BUILD' selected. An orange callout box contains the text: 'After finishing building, change to the "Release" mode and build again'.

檔案(F) 編輯(E) 檢視(V) Git(G) 專案(P) **建置(B)** 偵錯(D) 測試(S) 分析(N) 工具(T) 延伸模組(X) 視窗(W) 說明(H) 搜尋... freelut

Debug 建置方案(B) Ctrl+Shift+B

- 重建方案(R)
- 清除方案(C)
- 為解決方案建置完整程式資料庫檔案
- 針對方案執行 Code Analysis(Y) Alt+F11
- 建置 ALL_BUILD(U) Ctrl+B
- 重建 ALL_BUILD(E)
- 清除 ALL_BUILD(N)
- 僅限專案(J)
- 批次建置(T)...
- 組態管理員(O)...

方案總管

- 3dview
- 3dview_static
- ALL_BUILD**
- CallbackMaker
- CallbackMaker_static
- Fractals
- Fractals_random
- Fractals_random_static
- Fractals_static
- freelut
- freelut_static
- indexed_color
- indexed_color_static
- INSTALL
- joystick
- joystick_static
- keyboard
- keyboard_static
- Lorenz
- Lorenz_static

方案總管 Git 變更

錯誤清單 輸出 尋找符號結果

就緒

新增至原始檔控制 選取存放庫

Examine the Built Binary Files

master > build > 🔍 搜尋 build

名稱	修改日期	類型	大小
3dview.dir	2022/9/14 下午 03:57	檔案資料夾	
3dview_static.dir	2022/9/14 下午 03:57	檔案資料夾	
bin	2022/9/14 下午 03:57	檔案資料夾	
CallbackMaker.dir	2022/9/14 下午 03:57	檔案資料夾	
CallbackMaker_static.dir	2022/9/14 下午 03:57	檔案資料夾	
CMakeFiles	2022/9/14 下午 03:57	檔案資料夾	
Fractals.dir	2022/9/14 下午 03:57	檔案資料夾	
Fractals_random.dir	2022/9/14 下午 03:57	檔案資料夾	
Fractals_random_static.dir	2022/9/14 下午 03:57	檔案資料夾	
Fractals_static.dir	2022/9/14 下午 03:57	檔案資料夾	
FreeGLUT	2022/9/14 下午 03:47	檔案資料夾	
freelut.dir	2022/9/14 下午 03:57	檔案資料夾	
freelut_static.dir	2022/9/14 下午 03:57	檔案資料夾	
indexed_color.dir	2022/9/14 下午 03:57	檔案資料夾	
indexed_color_static.dir	2022/9/14 下午 03:57	檔案資料夾	
joystick.dir	2022/9/14 下午 03:57	檔案資料夾	
joystick_static.dir	2022/9/14 下午 03:57	檔案資料夾	
keyboard.dir	2022/9/14 下午 03:57	檔案資料夾	
keyboard_static.dir	2022/9/14 下午 03:57	檔案資料夾	
lib	2022/9/14 下午 03:57	檔案資料夾	
Lorenz.dir	2022/9/14 下午 03:57	檔案資料夾	
Lorenz_static.dir	2022/9/14 下午 03:57	檔案資料夾	
multi-touch.dir	2022/9/14 下午 03:57	檔案資料夾	
multi-touch_static.dir	2022/9/14 下午 03:57	檔案資料夾	
One.dir	2022/9/14 下午 03:57	檔案資料夾	
One_static.dir	2022/9/14 下午 03:57	檔案資料夾	

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