



# **OpenGL Startup**

**Computer Graphics**

**Yu-Ting Wu**

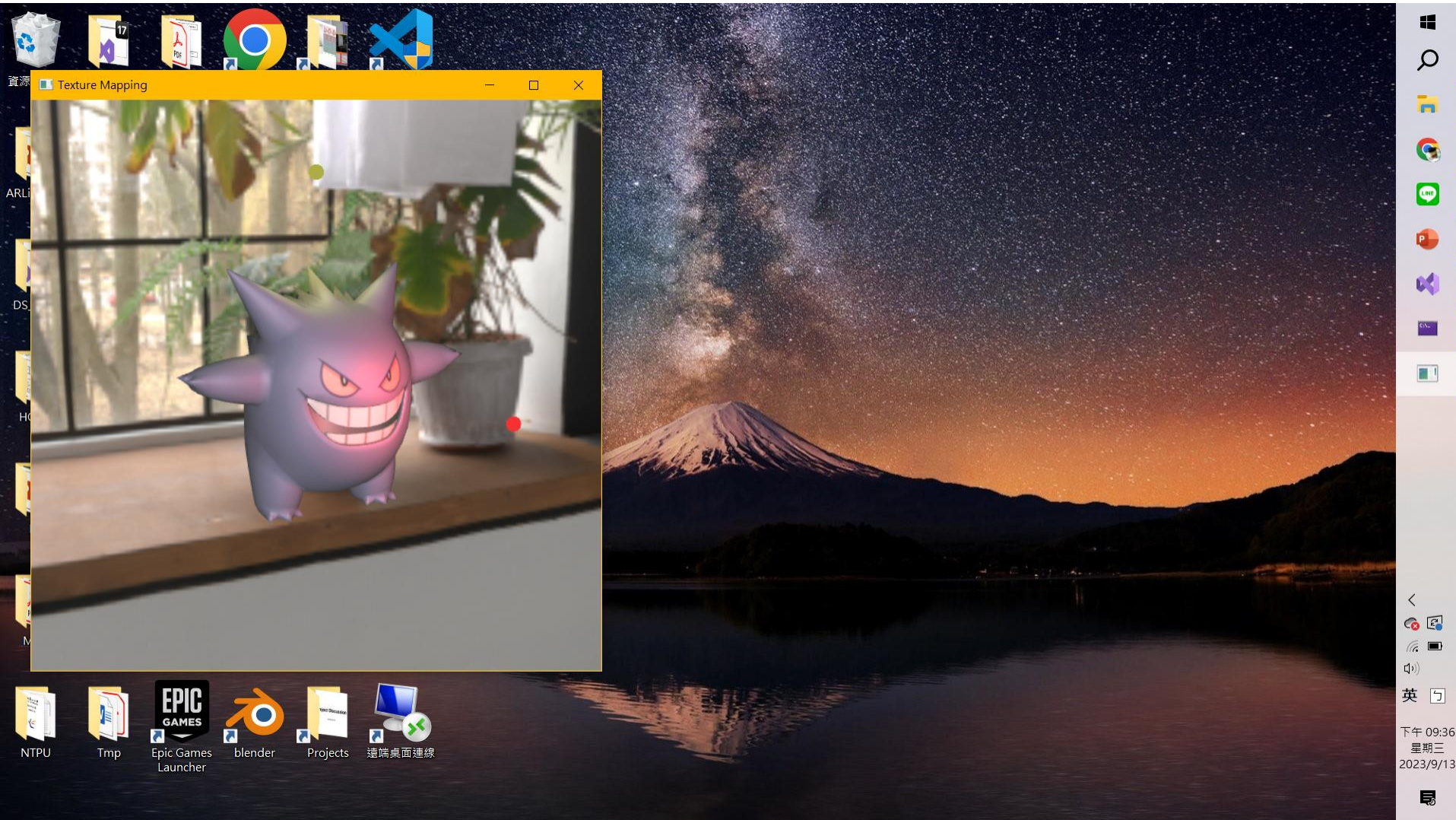
# Outline

- [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



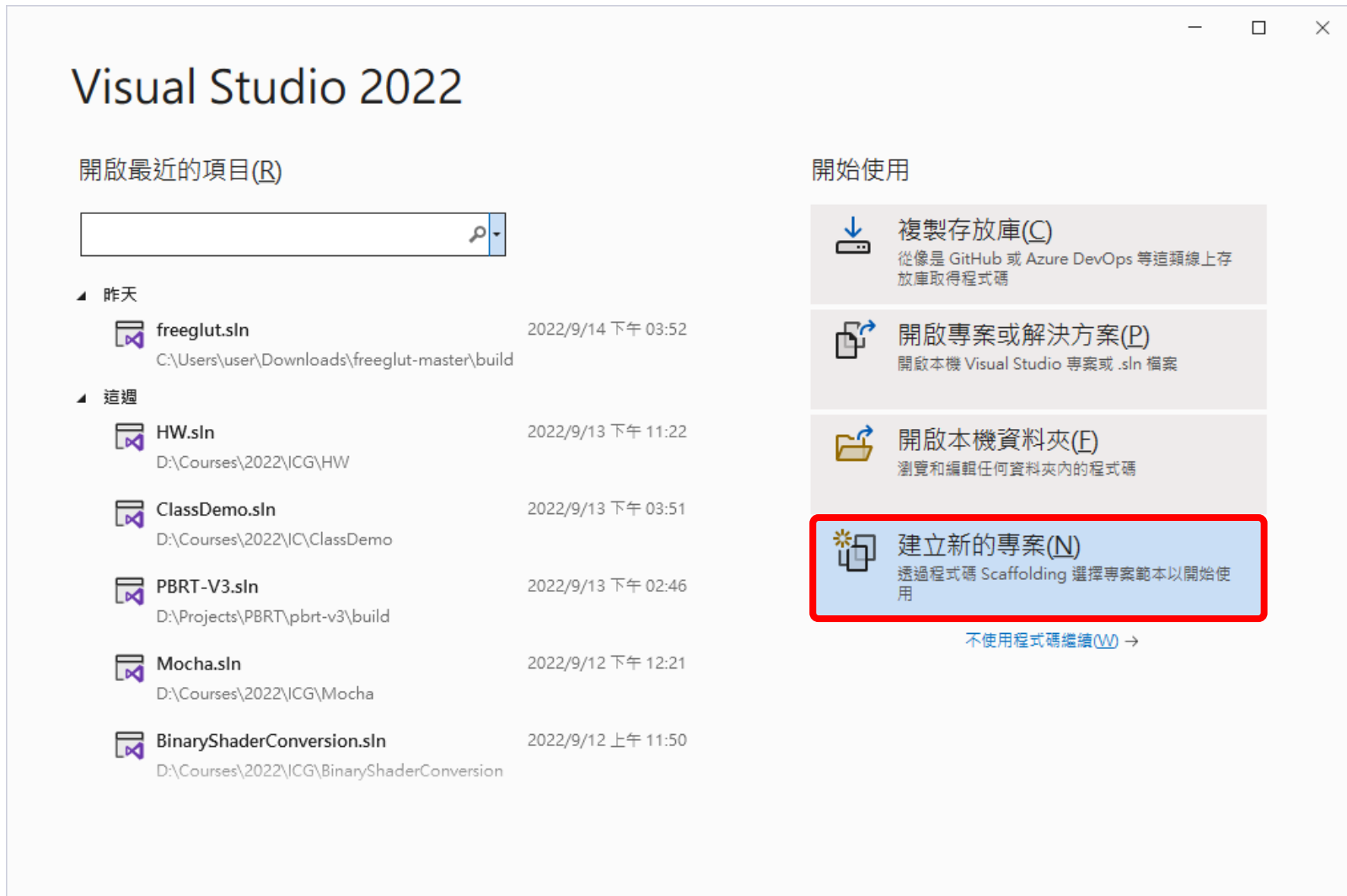
# Library for Handling Screen Rendering

- **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

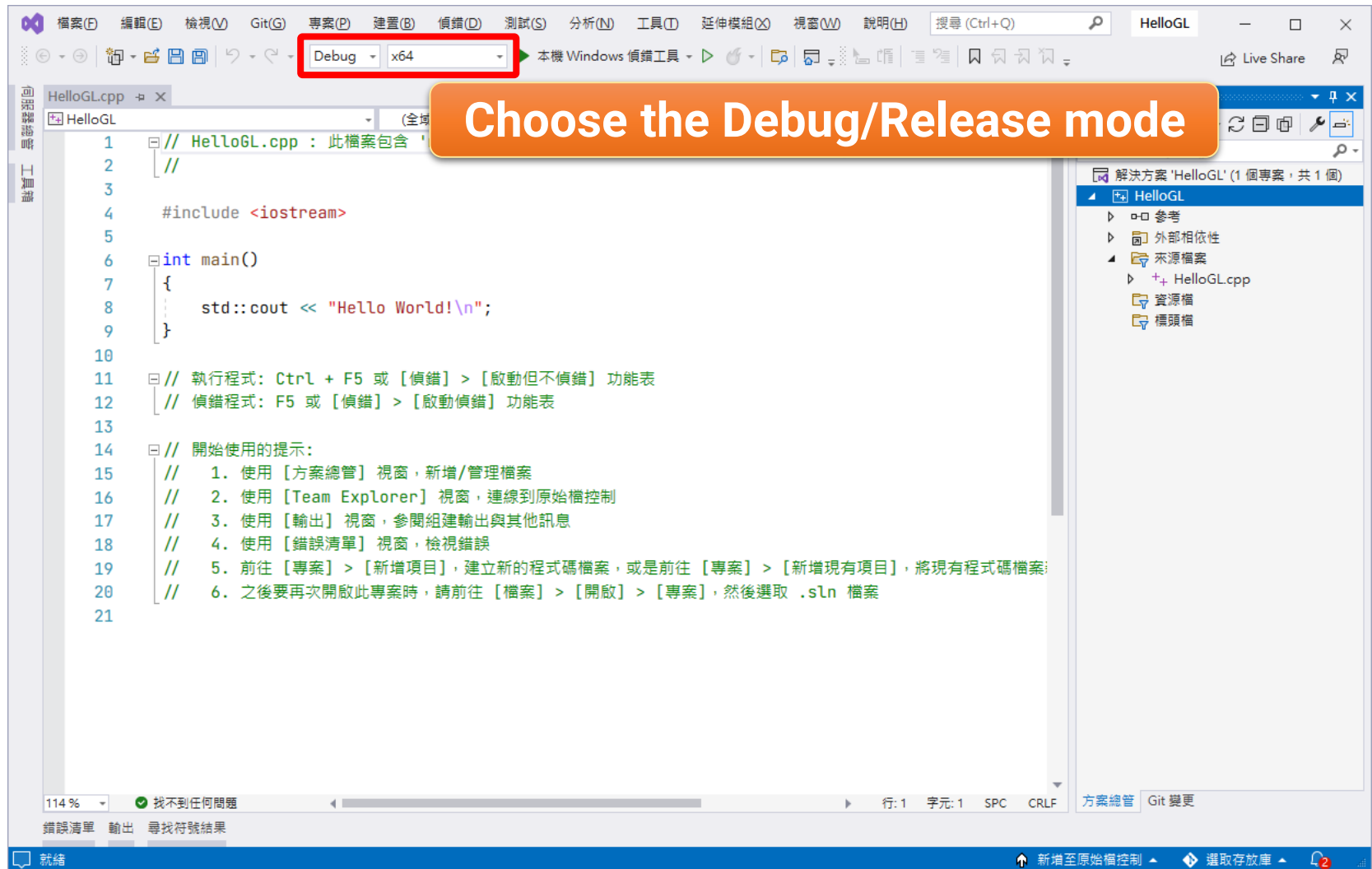
位置 (L)  
C:\Users\user\Desktop\

解決方案名稱 (M) ⓘ  
HelloGL

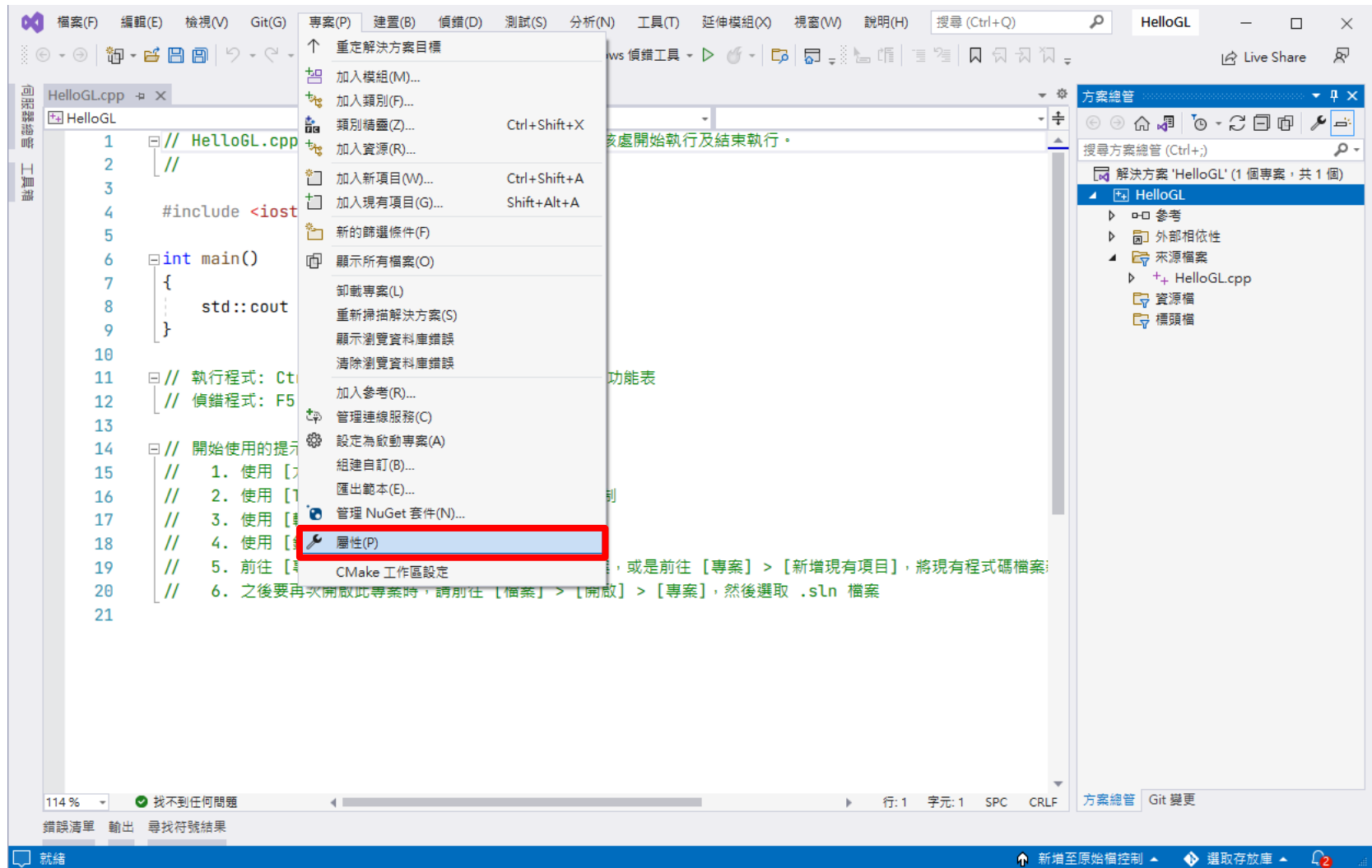
將解決方案與專案置於相同目錄中 (D)

上一步 (B) 建立 (C)

# Setup the Project in VS



# Setup the Project in VS



# Setup the Project in VS (cont.)

For include headers, you can use “all configuration”

組態(C): 作用中 (Debug) 平台(P): 作用中 (x64) 組態管理員(O)...

組態屬性

- 一般
- 進階
- 偵錯
- VC++ 目錄
- C/C++**
- 連結器
- 資訊清單工具
- XML 文件產生器
- 瀏覽資訊
- 建置事件
- 自訂建置步驟
- Code Analysis

其他 Include 目錄	
其他 #using 目錄	
其他 BMI 目錄	
其他模組相依性	
其他標頭單位相依性	
掃描來源是否具有模組相依性	否
將 Include 翻譯為 Import	否
偵錯資訊格式	[編輯後繼續] 的程式資料庫 (/ZI)
支援 Just My Code 偵錯	是 (/JMC)
Common Language Runtime 支援	
使用 Windows 執行階段擴充功能	
隱藏程式啟始資訊	是 (/nologo)
警告層級	層級 3 (/W3)
將警告視為錯誤	否 (/WX-)
警告版本	

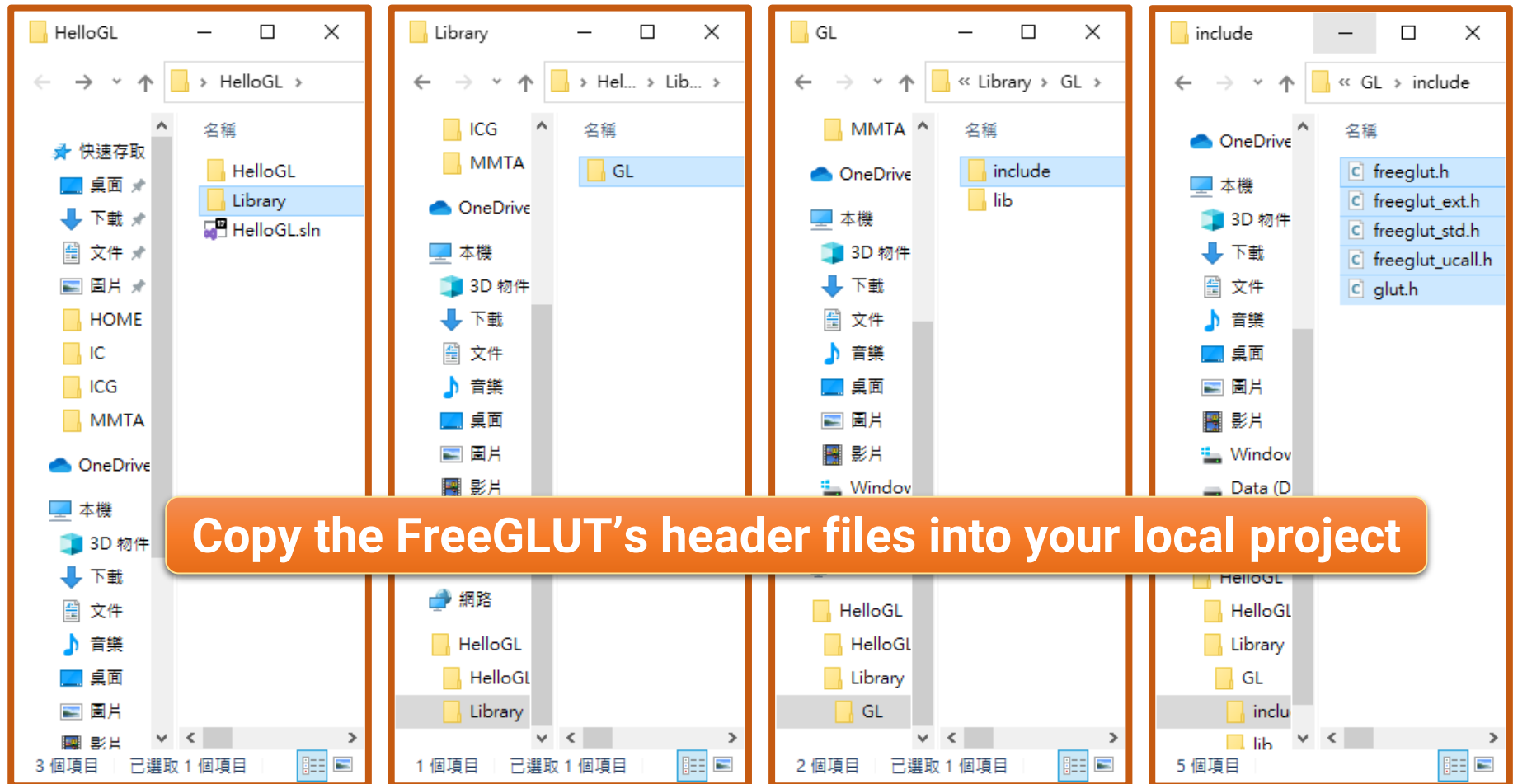
指定一個或多個要新增至 include 路徑的目錄。若有一個以上，請以分號分隔。 [path]

確定 取消 套用(A)

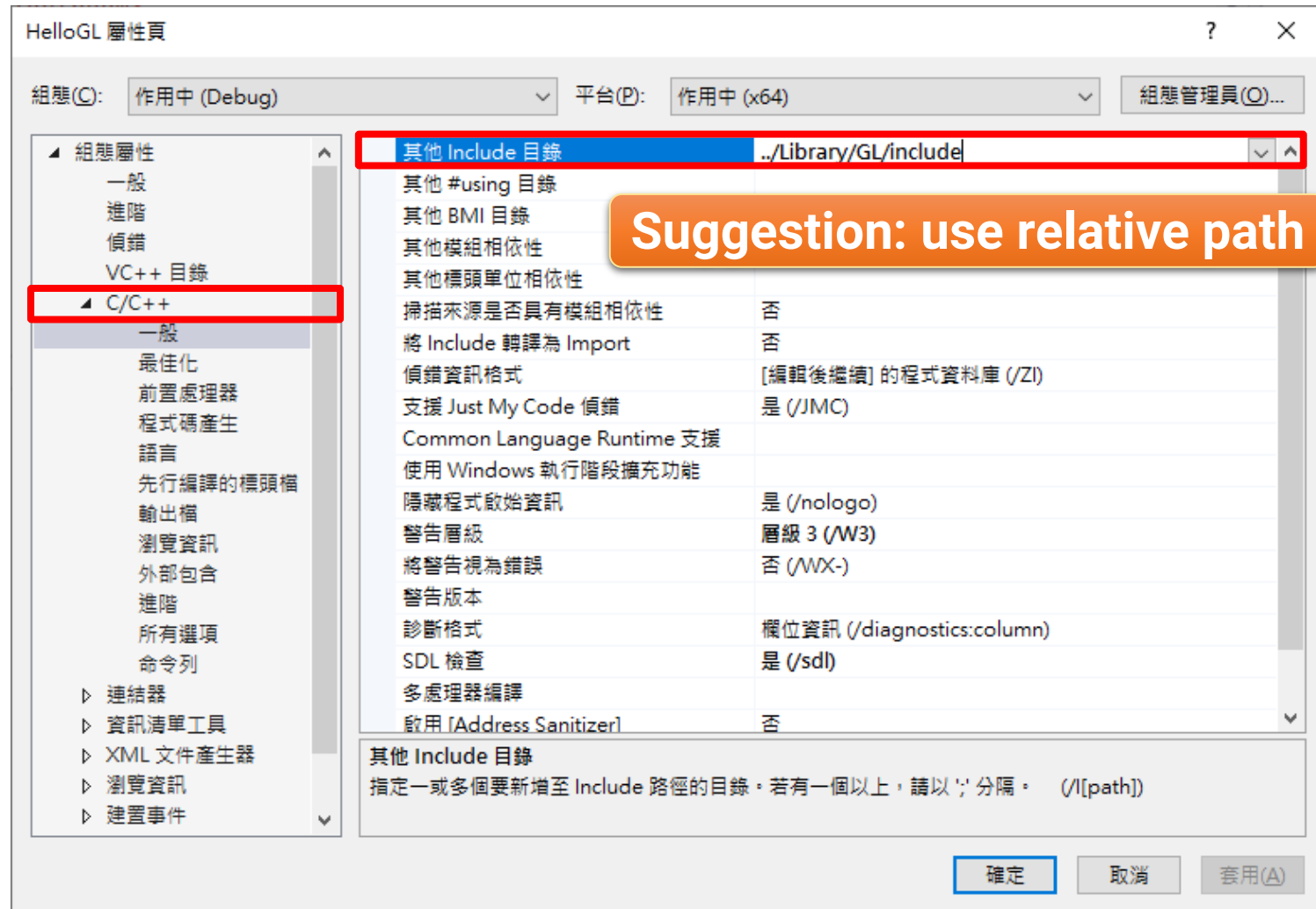
Set the location of all additionally included headers (suggestion: copy the files to your project folder)

# Setup the Project in VS (cont.)

- My setting



# Setup the Project in VS (cont.)



# Setup the Project in VS (cont.)

The screenshot shows the Visual Studio IDE with a C++ project named 'HelloGL'. The main source file, 'HelloGL.cpp', contains the following code:

```
1 #include "freeglut.h"
2 #include <iostream>
3
4 int main()
5 {
6     std::cout << "Hello GL!" << std::endl;
7
8     return 0;
9 }
10
11
12
```

An orange callout box points to the first line of code, stating: "Now you can find the included header".

The error list at the bottom shows a single error:

錯誤清單	專案	檔案	行
LNK110: 無法開啟檔案 'freeglutd.lib'	HelloGL	LINK	1

An orange callout box points to this error, stating: "But we still need to set the lib".

# Setup the Project in VS (cont.)

HelloGL 屬性頁

組態(C): 作用中 (Debug) 平台(P): 作用中 (x64) 組態管理員(O)...

組態屬性

- 一般
- 進階
- 偵錯
- VC++ 目錄
- C/C++
- 連結器**
  - 一般**
  - 輸入
  - 資訊清單檔
  - 偵錯
  - 系統
  - 最佳化
  - 內嵌 IDL
- 瀏覽資訊
- 建置事件

輸出檔案	\$(OutDir)\$(TargetName)\$(TargetExt)
顯示進度	未設定
版本	
啟用累加連結	是 (/INCREMENTAL)
增量連結資料庫檔案	\$(IntDir)\$(TargetName).ilk
隱藏程式啟始資訊	是 (/NOLOGO)
忽略匯入程式庫	否
登錄輸出	否
個別使用者重新導向	否
<b>其他程式庫目錄</b>	<b>e.g., ../Library/GL/lib</b>
連結程式庫相依性	是
使用程式庫相依性輸入	否
連結狀態	
防止 DLL 載入	

輸出檔案  
/OUT 選項會覆寫連結器建立的程式預設名稱和位置。

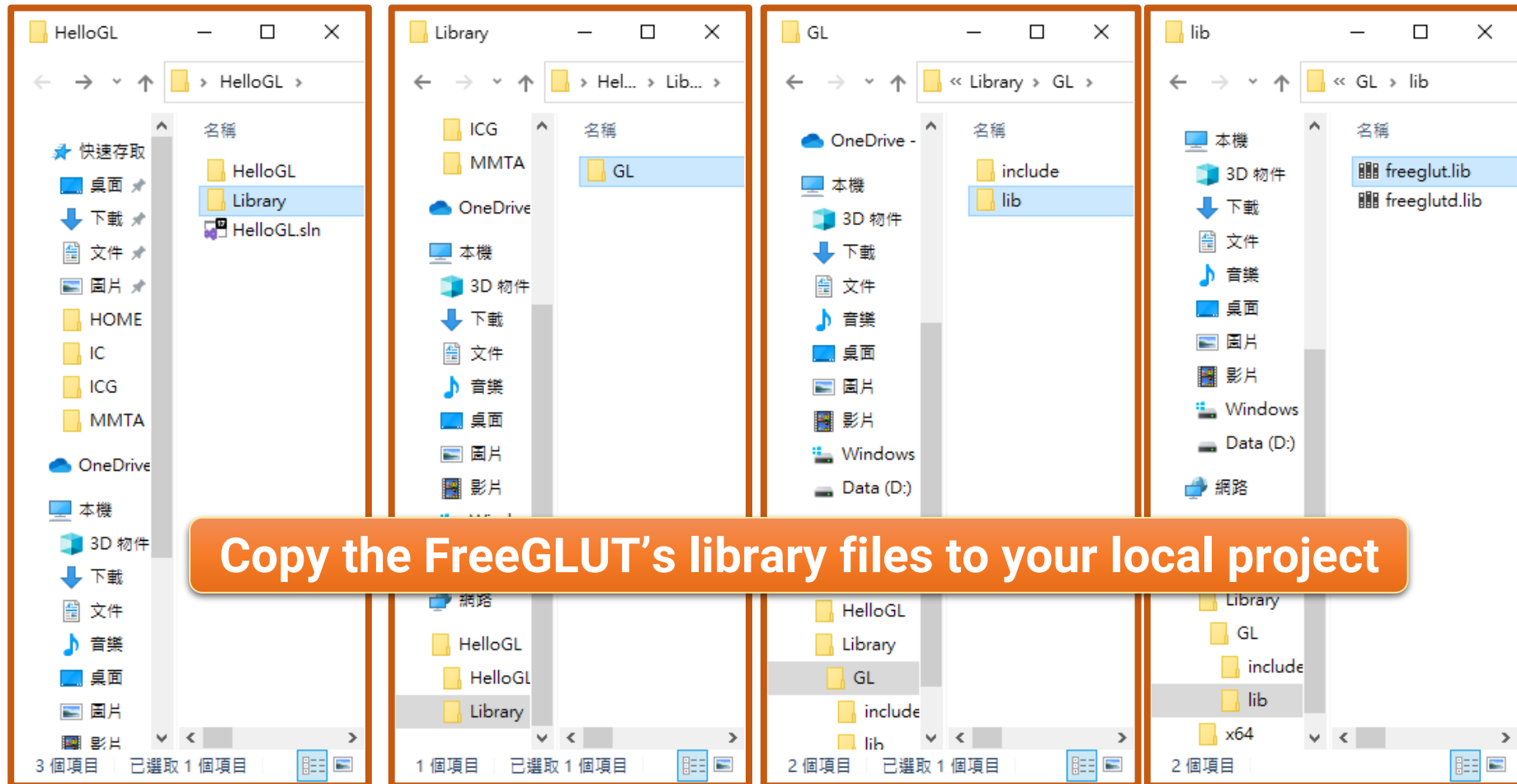
確定 取消 套用(A)

**Set the location of all additionally imported libraries (suggestion: copy the files to your project folder)**

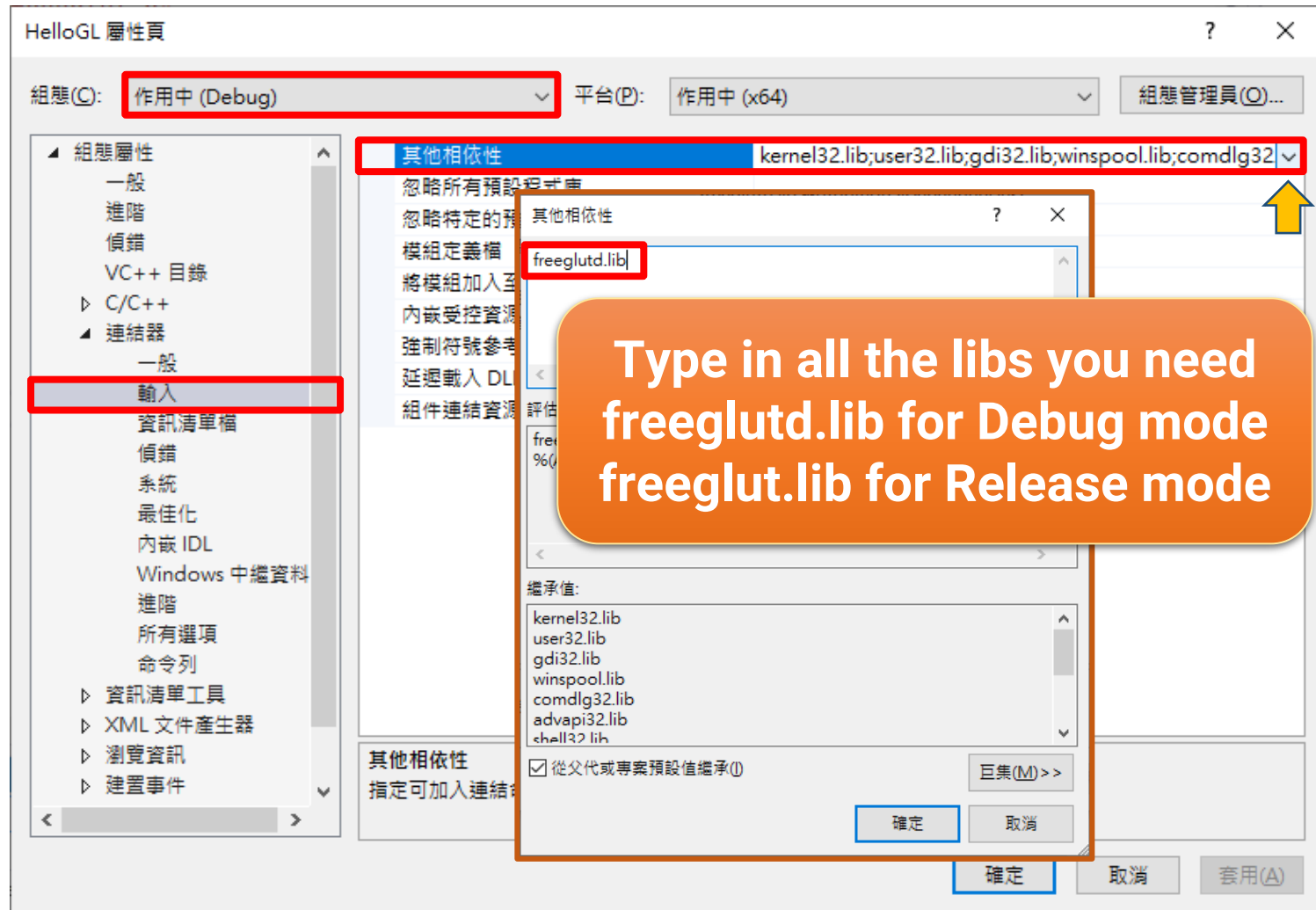


# Setup the Project in VS (cont.)

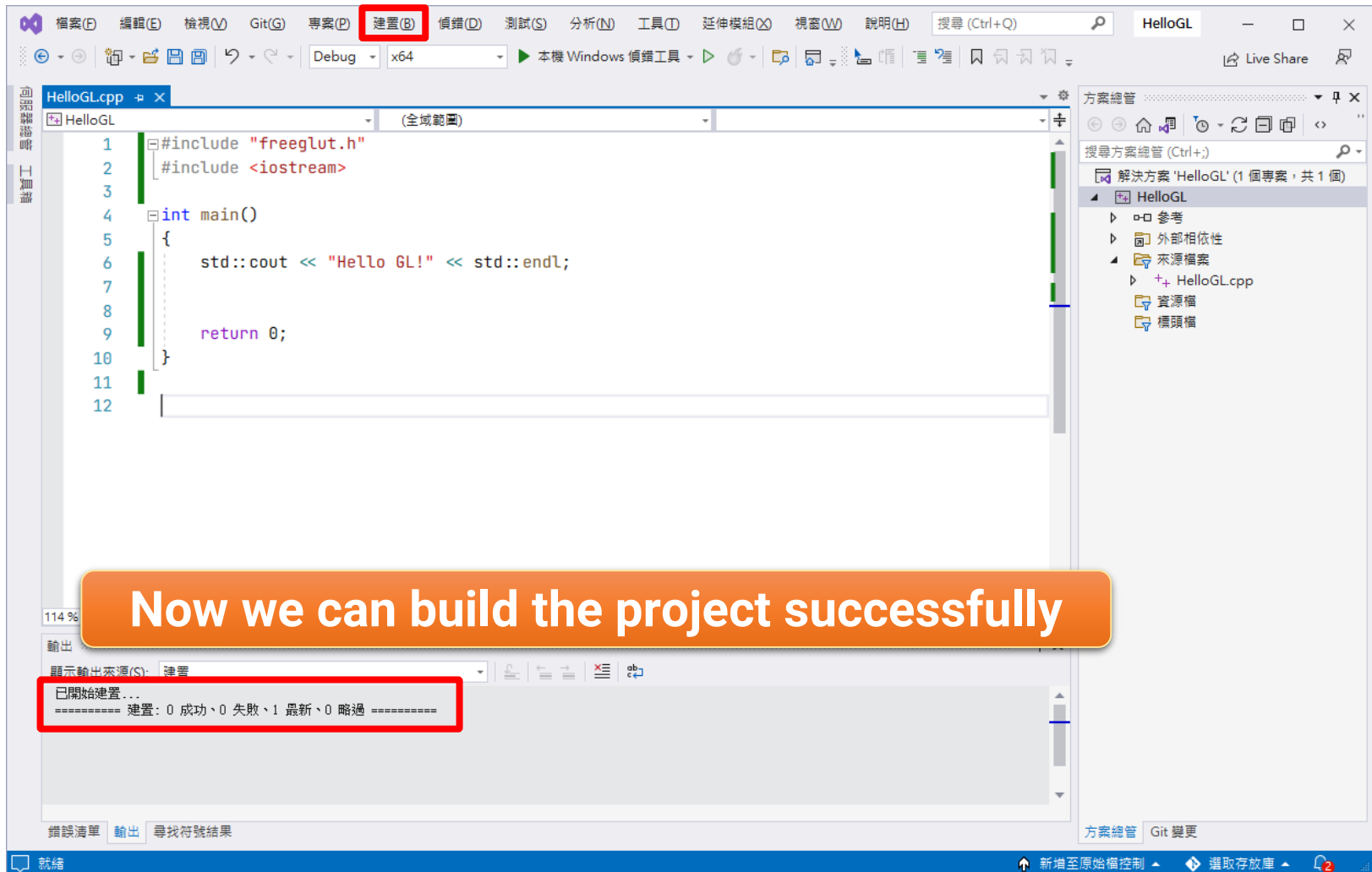
- My setting



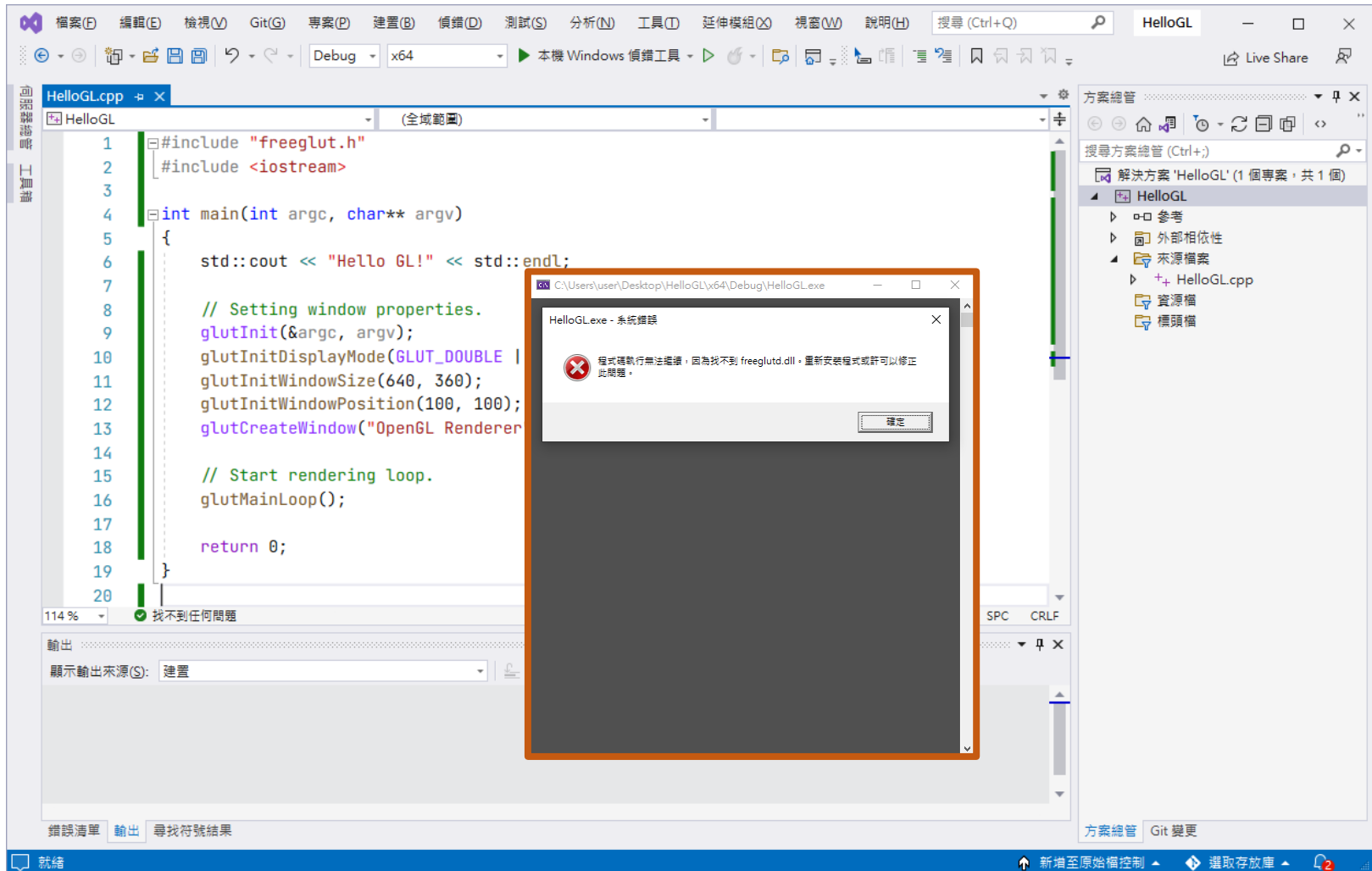
# Setup the Project in VS (cont.)



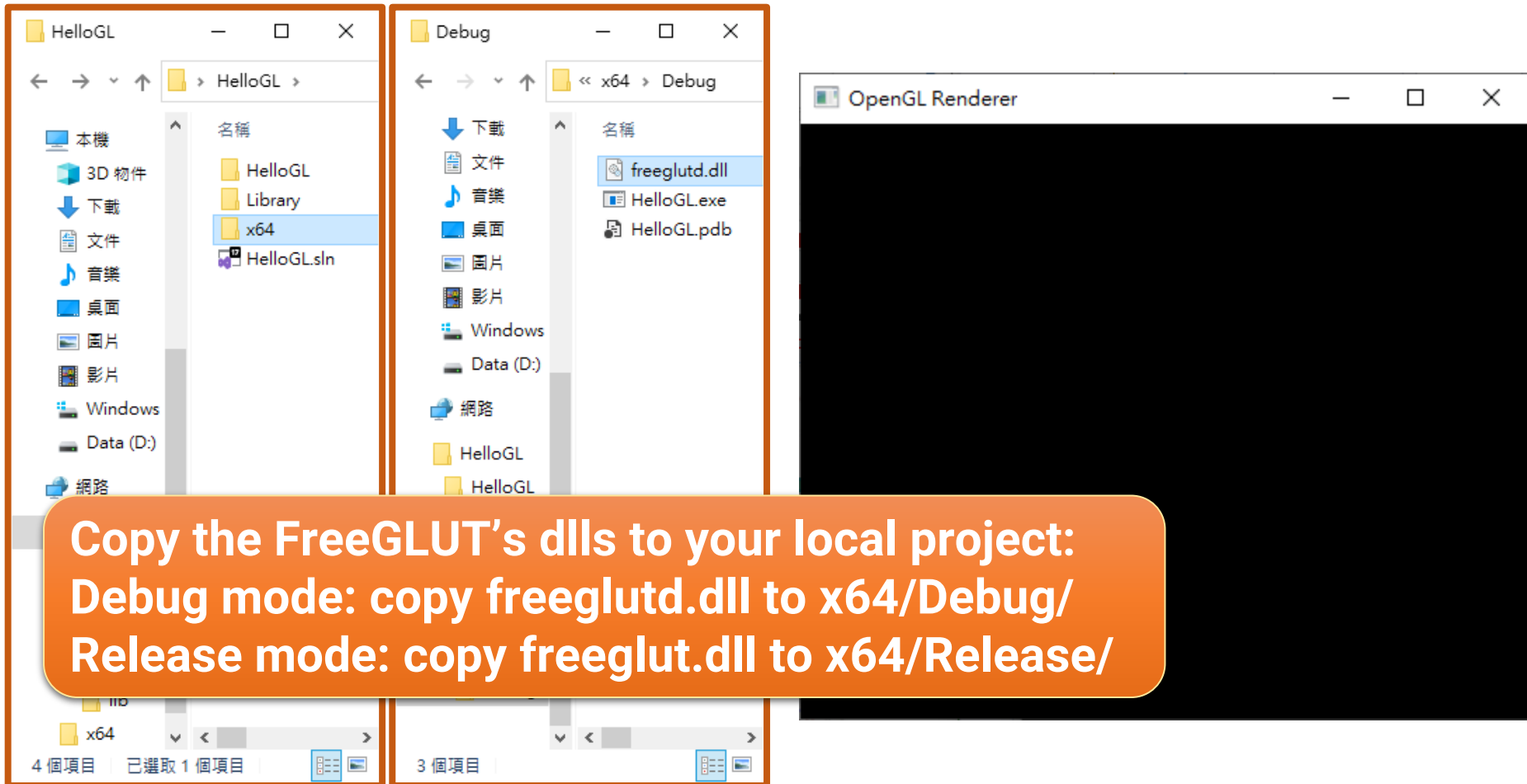
# Setup the Project in VS (cont.)



# Setup the Project in VS (cont.)



# Setup the Project in VS (cont.)



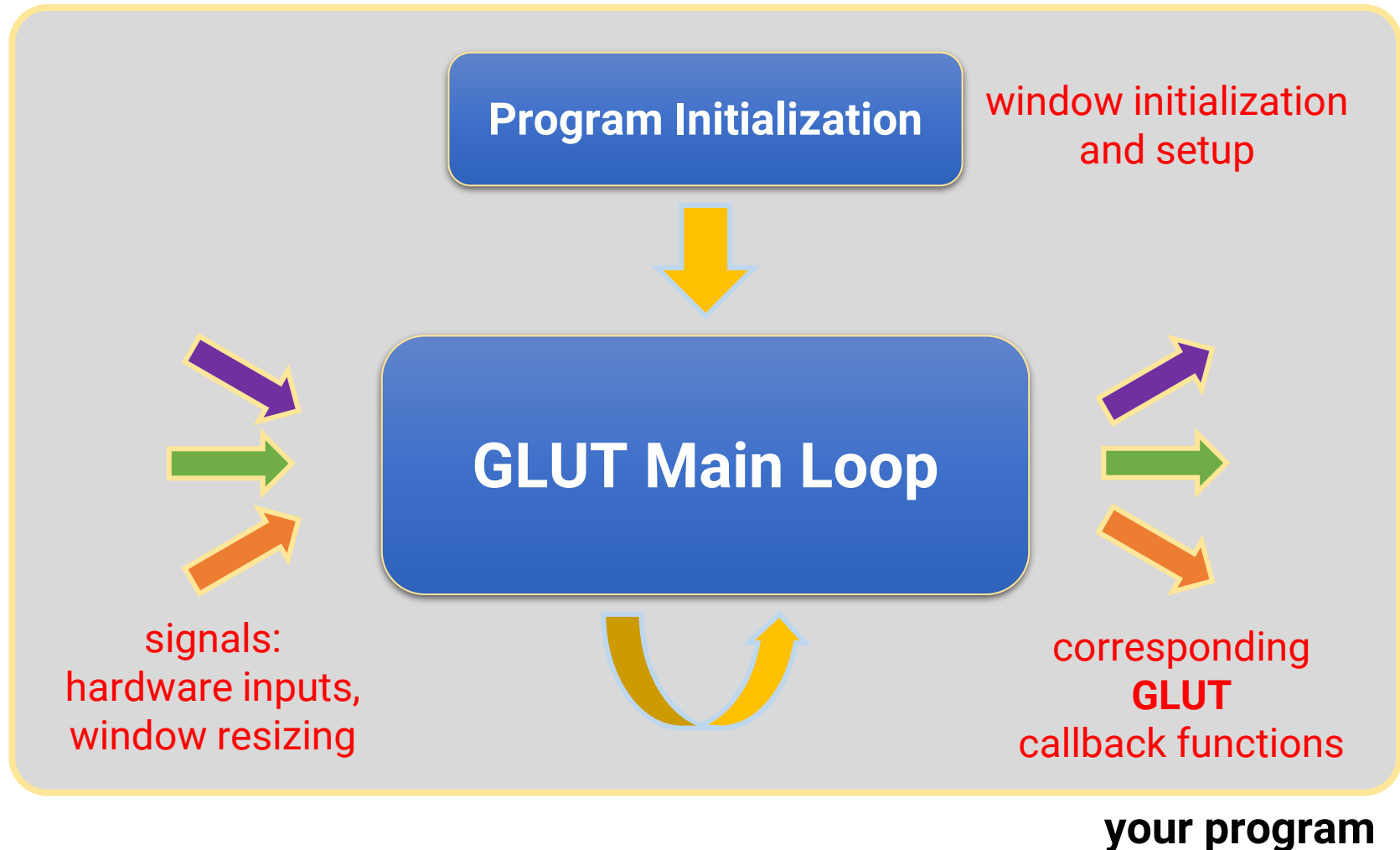
The image shows three windows from Visual Studio. On the left, the 'HelloGL' file explorer shows the project structure with folders 'HelloGL', 'Library', and 'x64', and a file 'HelloGL.sln'. The 'x64' folder is selected. In the middle, the 'Debug' file explorer shows the contents of the 'x64/Debug' folder, including 'freelutd.dll', 'HelloGL.exe', and 'HelloGL.pdb'. The 'freelutd.dll' file is selected. On the right, the 'OpenGL Renderer' window is shown as a solid black rectangle, indicating the application is running but not displaying any content.

**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

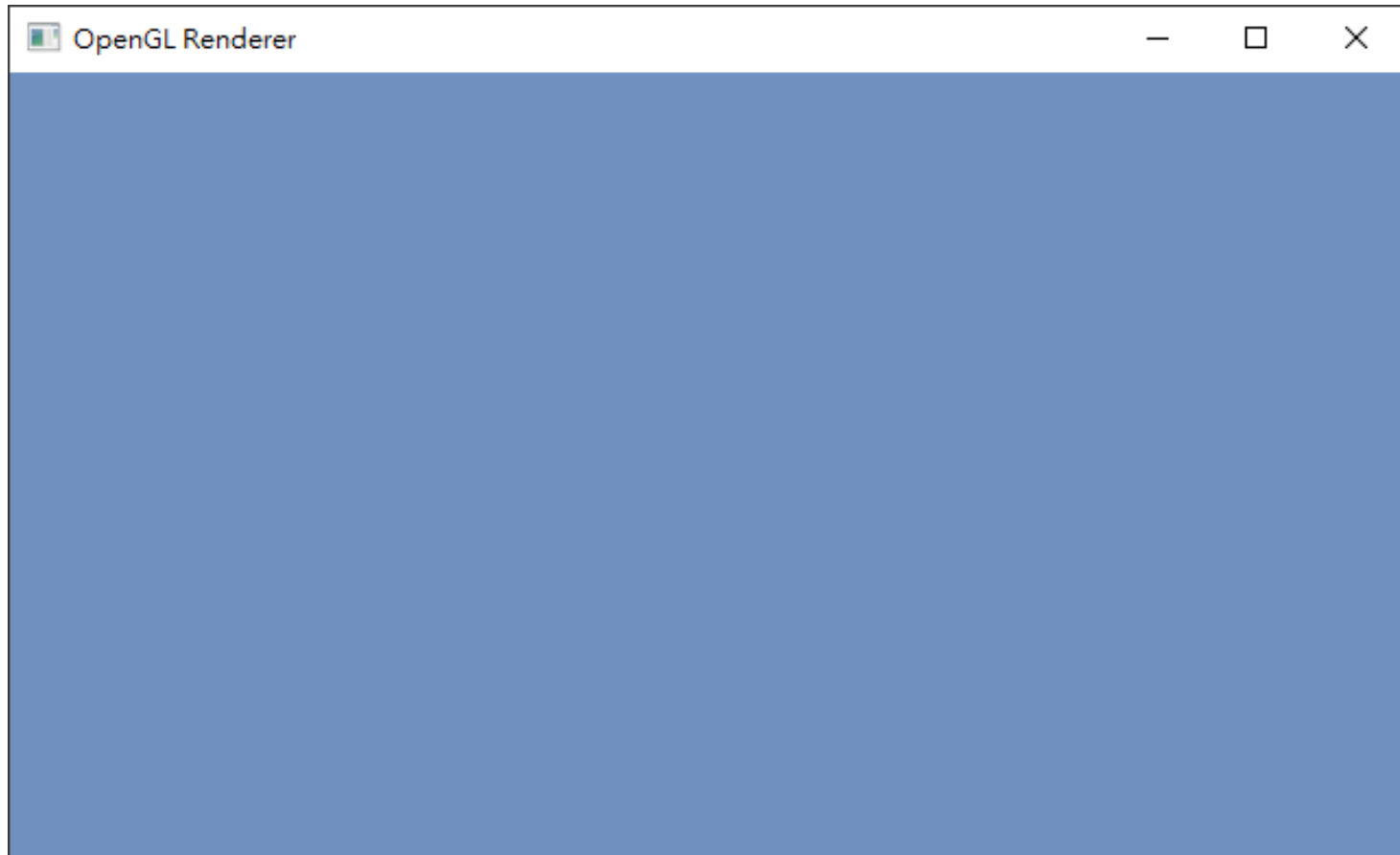
```
return 0;
```

```
}
```



# 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);
```

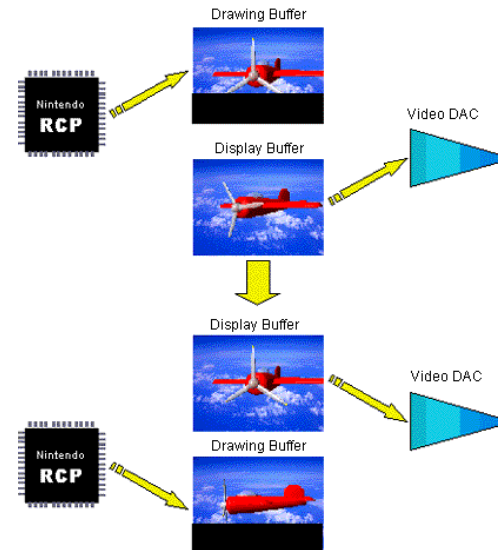
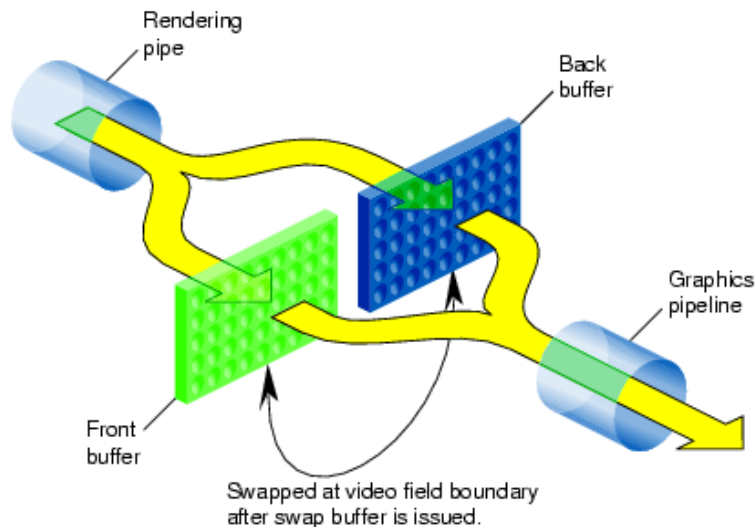
- void **glutInitDisplayMode**(unsigned int mode);
  - Set the initial display mode
  - <https://www.opengl.org/resources/libraries/glut/spec3/node12.html>

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

# Double Buffers

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

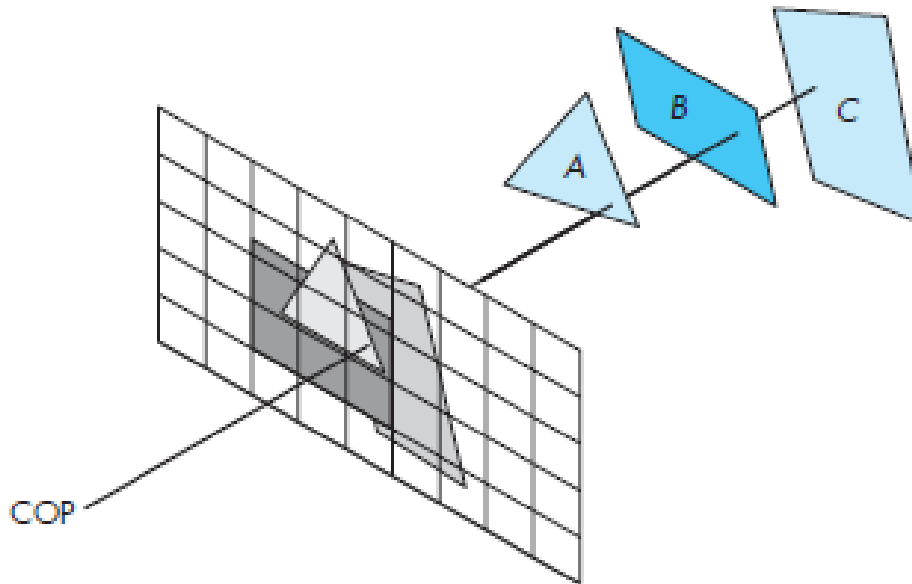
- Prevent artifacts due to potentially seeing parts of an incomplete frame (that is currently drawn)
  - Set the display mode to **GLUT\_DOUBLE** in the **glutInitDisplayMode** function
  - Call **glutSwapBuffers** after rendering finished



# Depth Buffer

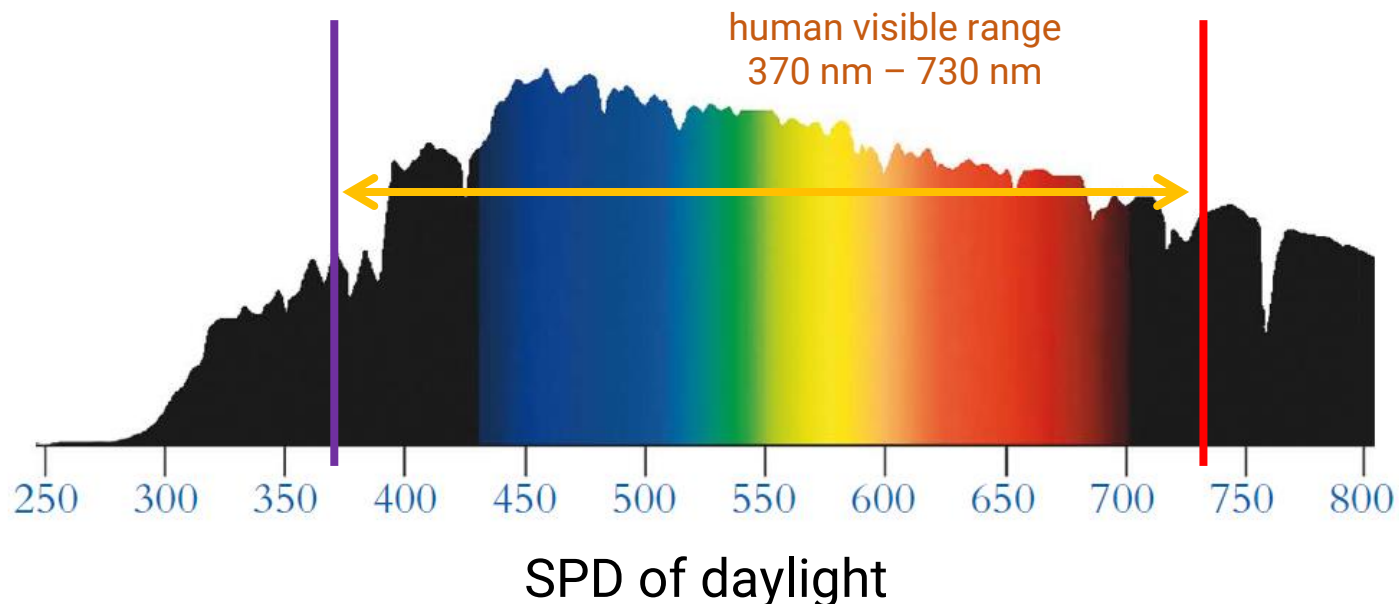
```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
```

- 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.)

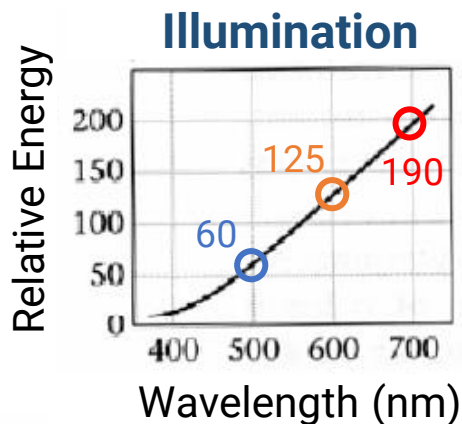
- Reflected color is the result of interaction of **light source spectrum** with **surface reflectance**



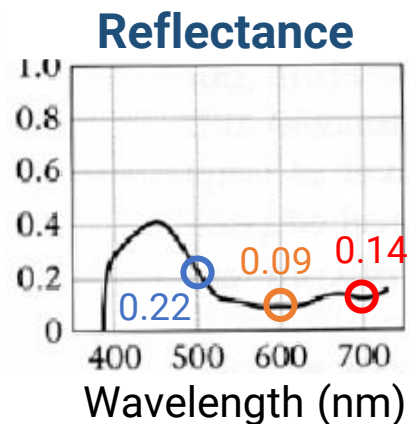


# Color: Spectral Power Distribution (cont.)

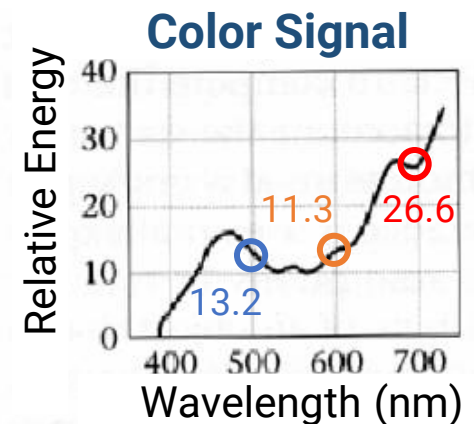
- Reflected color is the result of interaction of **light source spectrum** with **surface reflectance**



\*



=



# 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.)

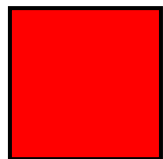
- For more details about tristimulus theory, please refer to my course in “Multimedia Technology and Application”
  - Course material link:
    - Part 1: <https://reurl.cc/11Nmk8>
    - Part 2: <https://reurl.cc/IDKep9>
    - Part 3: <https://reurl.cc/65n0Gb>

# RGB Color Model

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

**(r, g, b),**

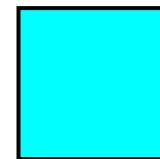
where r, g, b are the **amounts (proportion of the pure light)** of red, green, and blue light making up the color



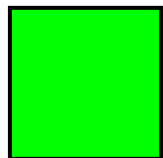
Red  
(100%, 0%, 0%)



Black  
(0%, 0%, 0%)



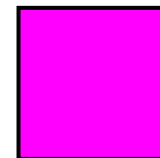
Cyan  
(0%, 100%, 100%)



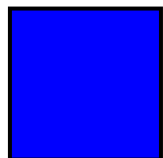
Green  
(0%, 100%, 0%)



White  
(100%, 100%, 100%)



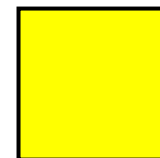
Magenta  
(100%, 0%, 100%)



Blue  
(0%, 0%, 100%)



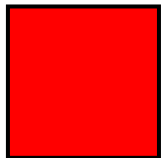
Gray  
(50%, 50%, 50%)



Yellow  
(100%, 100%, 0%)

# RGB Color Model (cont.)

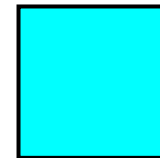
- In most applications, we use **8 bits** (1 byte) for each primary color, making 24 bits (3 bytes) in total
  - The range of each value falls within  $[0, 255]$ , making a total of  $256 \times 256 \times 256 = 16777216$  different colors



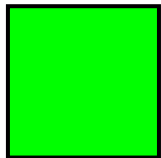
Red  
(255, 0, 0)



Black  
(0, 0, 0)



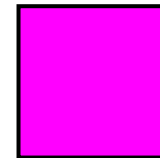
Cyan  
(0, 255, 255)



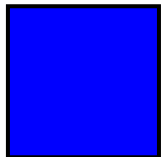
Green  
(0, 255, 0)



White  
(255, 255, 255)



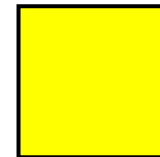
Magenta  
(255, 0, 255)



Blue  
(0, 0, 255)



Gray  
(127, 127, 127)



Yellow  
(255, 255, 0)

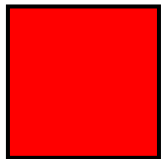
# RGB Color Model (cont.)

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

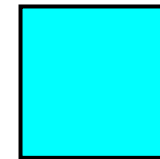
```
float clearColor[4] = {0.44f, 0.57f, 0.75f, 1.00f};
glClearColor(
    (GLclampf)(clearColor[0]),
    (GLclampf)(clearColor[1]),
    (GLclampf)(clearColor[2]),
    (GLclampf)(clearColor[3])
);
```



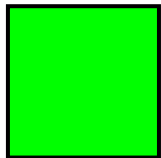
Red  
(1.0f, 0.0f, 0.0f)



Black  
(0.0f, 0.0f, 0.0f)



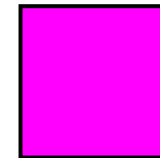
Cyan  
(0.0f, 1.0f, 1.0f)



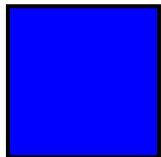
Green  
(0.0f, 1.0f, 0.0f)



White  
(1.0f, 1.0f, 1.0f)



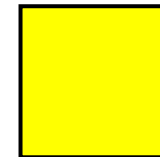
Magenta  
(1.0f, 0.0f, 1.0f)



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);

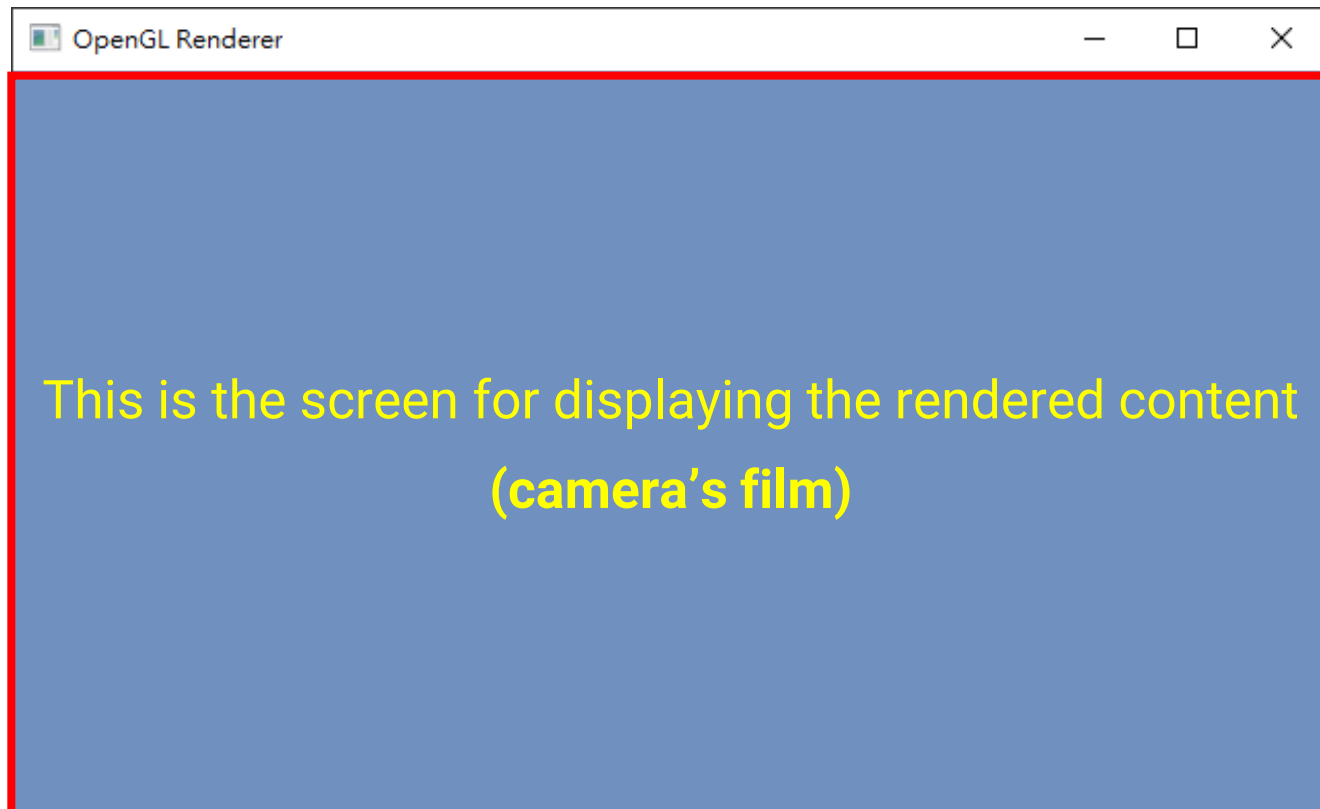
    // Start rendering loop.
    glutMainLoop();

    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>

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](https://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

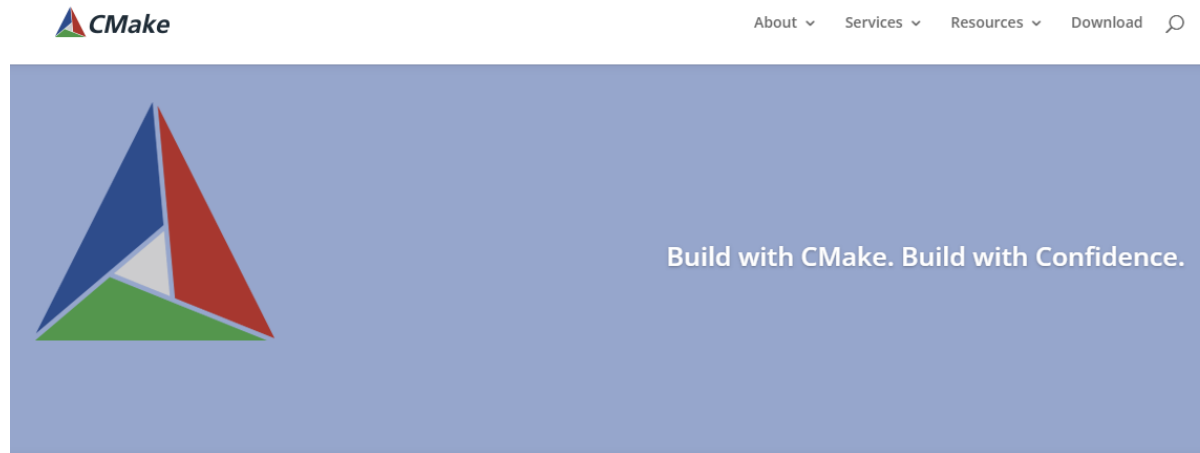
Build the source code  
using **CMake**





# CMake

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



CMake is an open-source, cross-platform family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in the compiler environment of your choice. The suite of CMake tools were created by Kitware in response to the need for a powerful, cross-platform build environment for open-source projects such as ITK and VTK.

CMake is part of Kitware's collection of commercially supported **open-source platforms** for software development.



Download Latest Release

Visit the download page



Support and Services

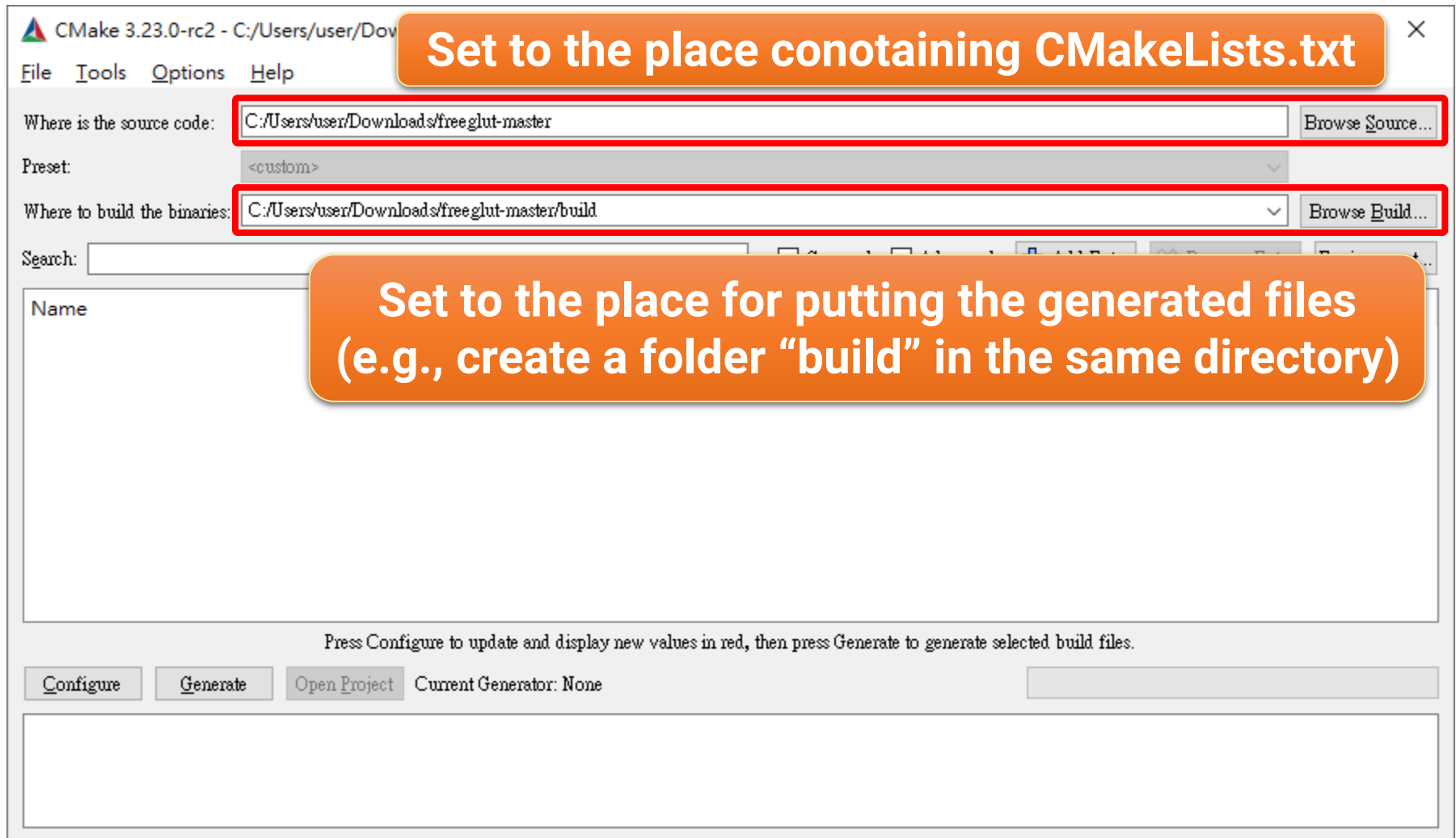
Get support or consulting service for CMake



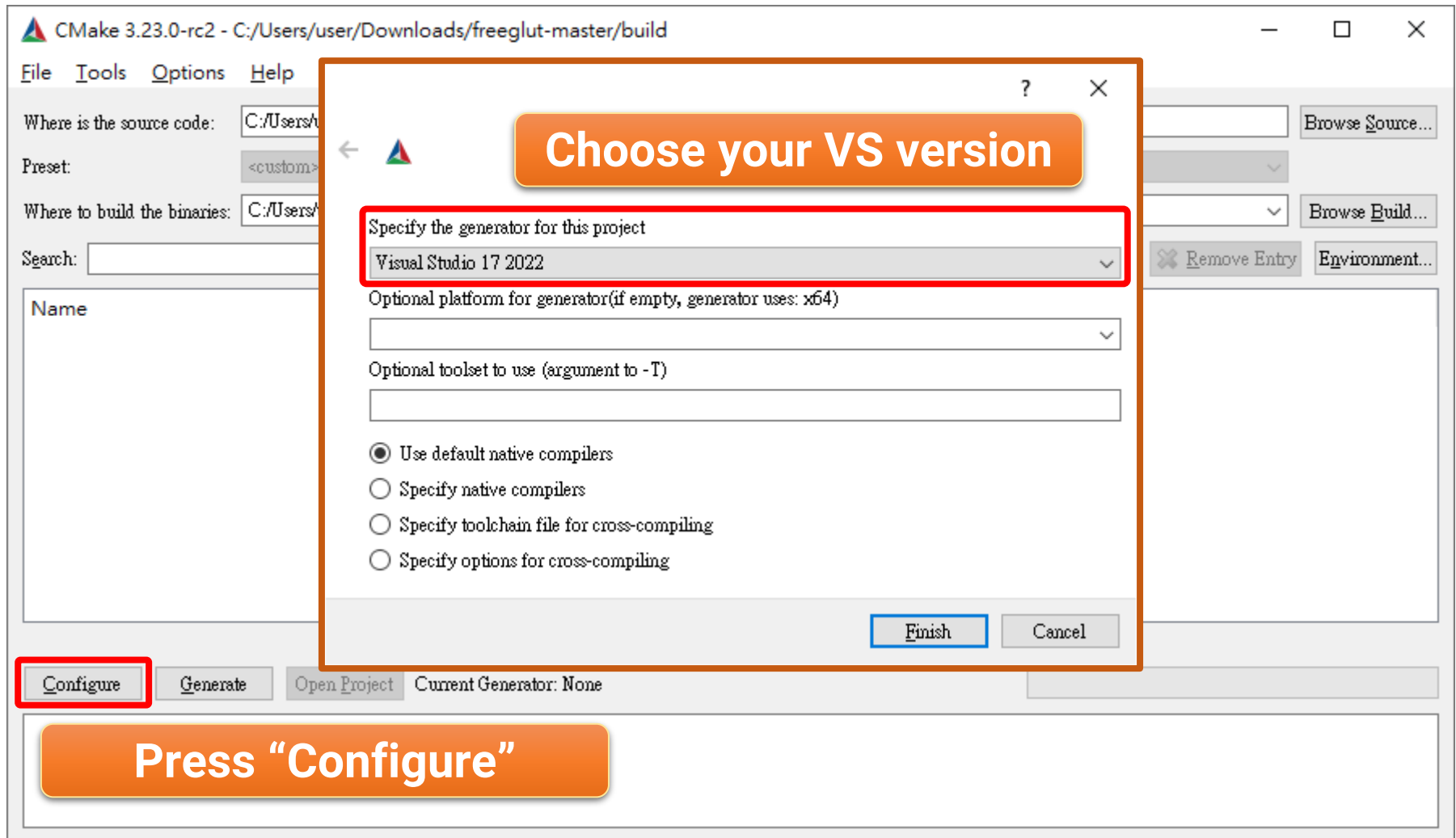
Contact Us

Have a question about a CMake project? We can help

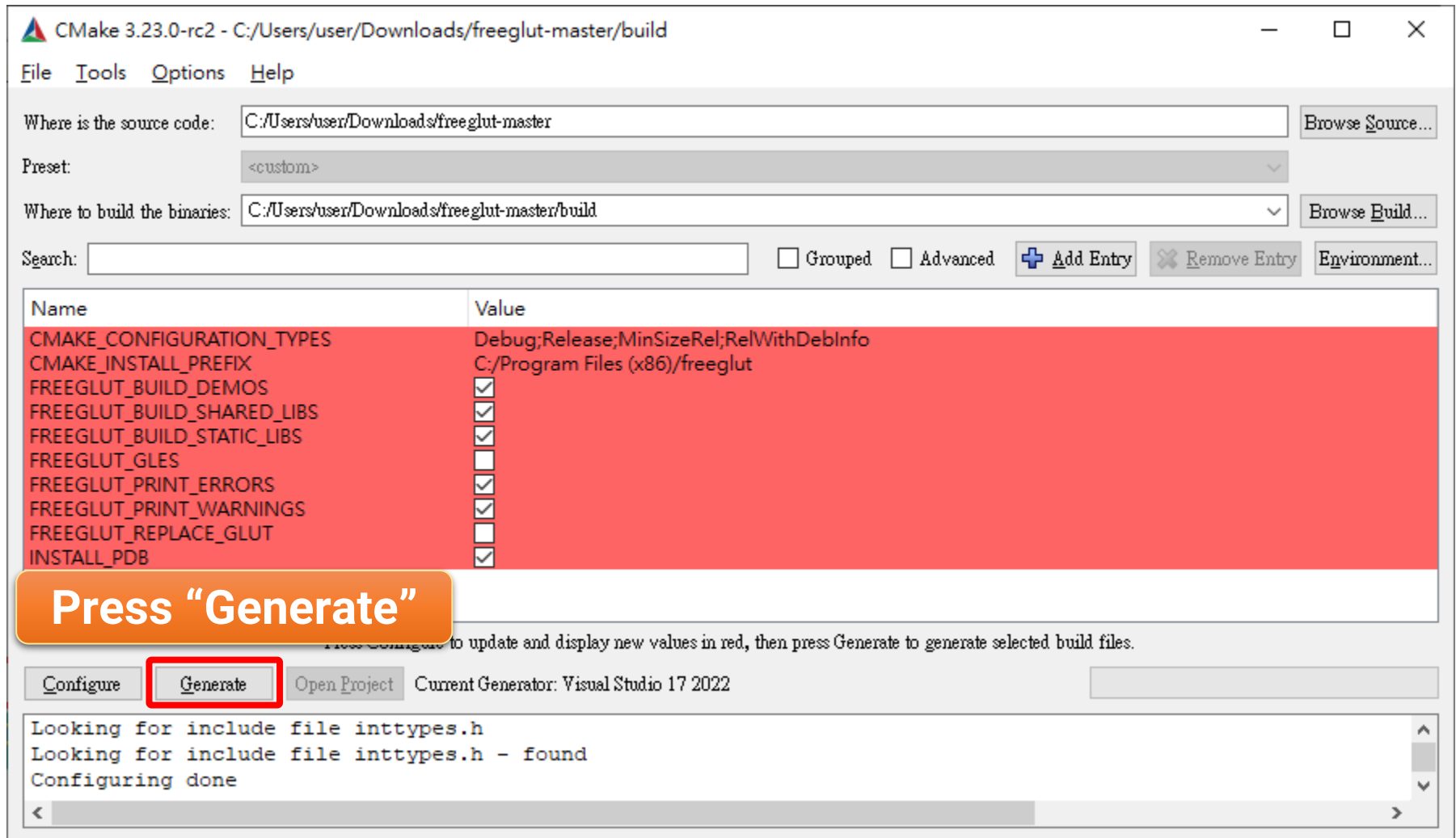
# Setup CMake for Building FreeGLUT



# Configuration



# Generate VS Project



CMake 3.23.0-rc2 - C:/Users/user/Downloads/freetgl-master/build

File Tools Options Help

Where is the source code: C:/Users/user/Downloads/freetgl-master Browse Source...

Preset: <custom>

Where to build the binaries: C:/Users/user/Downloads/freetgl-master/build Browse Build...

Search:   Grouped  Advanced + Add Entry ✖ Remove Entry Environment...

Name	Value
CMAKE_CONFIGURATION_TYPES	Debug;Release;MinSizeRel;RelWithDebInfo
CMAKE_INSTALL_PREFIX	C:/Program Files (x86)/freetgl
FREETGL_BUILD_DEMOS	<input checked="" type="checkbox"/>
FREETGL_BUILD_SHARED_LIBS	<input checked="" type="checkbox"/>
FREETGL_BUILD_STATIC_LIBS	<input checked="" type="checkbox"/>
FREETGL_GLES	<input type="checkbox"/>
FREETGL_PRINT_ERRORS	<input checked="" type="checkbox"/>
FREETGL_PRINT_WARNINGS	<input checked="" type="checkbox"/>
FREETGL_REPLACE_GLUT	<input checked="" type="checkbox"/>
INSTALL_PDB	<input checked="" type="checkbox"/>

**Press "Generate"**

Configure Generate Open Project Current Generator: Visual Studio 17 2022

```
Looking for include file inttypes.h
Looking for include file inttypes.h - found
Configuring done
```

# 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
freeglut.pc	2022/9/14 下午 03:46	PC 檔案	1 KB
freeglut.rc	2022/9/14 下午 03:46	Resource Script	2 KB
freeglut.sln	2022/9/14 下午 03:47	Visual Studio Sol...	43 KB
freeglut.vcxproj	2022/9/14 下午 03:47	VC++ Project	64 KB
freeglut.vcxproj.filters	2022/9/14 下午 03:47	VC++ Project Filt...	8 KB
freeglut_static.vcxproj	2022/9/14 下午 03:47	VC++ Project	59 KB
freeglut_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

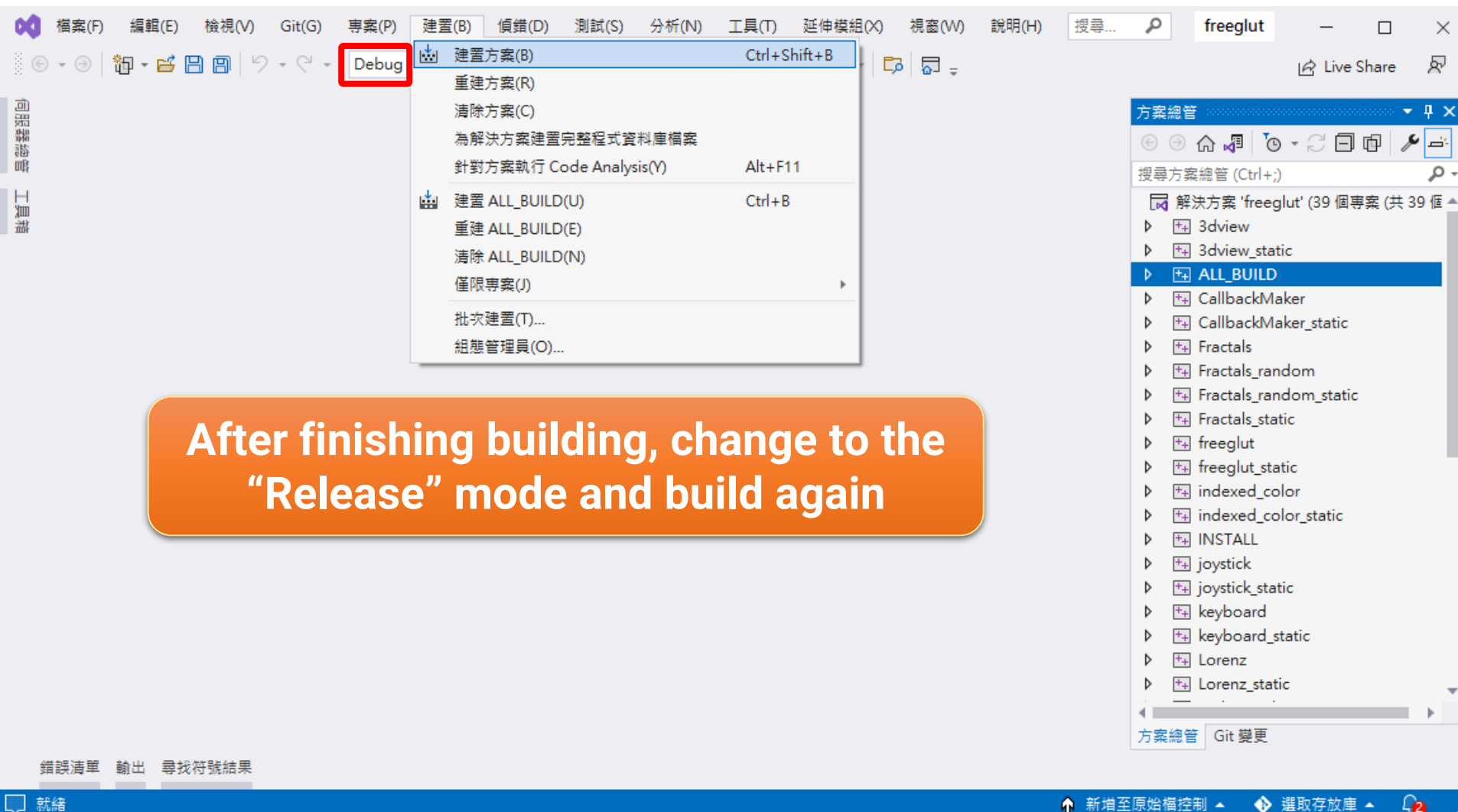
# Open Solution with Visual Studio

The screenshot shows the Visual Studio interface with the 'freeglut' solution open. The 'Debug' configuration is selected, and the 'x64' platform target is highlighted with a red box. The solution explorer on the right shows the project structure, including 'ALL\_BUILD' and various sub-projects like '3dview', 'Fractals', and 'joystick'. An orange callout box is overlaid on the main workspace area.

**Make sure the building version matches your OS**

Visual Studio interface elements visible include the menu bar (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)), the toolbar, the search bar (搜尋...), the Live Share button, the solution explorer (方案總管), and the status bar (鎖誤清單, 輸出, 尋找符號結果, 新增至原始檔控制, 選取存放庫).

# 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 - □ ×

同級目錄 工具欄

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 方案總管 (Ctrl+;) 搜尋方案總管 (Ctrl+;)

- 解決方案 'freelut' (39 個專案 (共 39 個))
- 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 >

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	
<b>bin</b>	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	
freeglut.dir	2022/9/14 下午 03:57	檔案資料夾	
freeglut_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	
<b>lib</b>	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



