

# Environment Setup Guide

CG 2025

# Outline

- ❖ IDE & Kit
- ❖ Windows
  - C++
  - CMake
- ❖ Linux
  - C++
  - CMake
  - Additional dependencies
- ❖ Mac
  - CMake
- ❖ Demo & test

# IDE & Kit

- ❖ Visual Studio Code (download [here](#))
- ❖ C++
- ❖ CMake
- ❖ GLFW (provided in zip)
  - An Open Source, multi-platform for OpenGL
  - Provide a simple API for creating windows, receiving input and, etc.
- ❖ GLAD (provided in zip)
  - An OpenGL loading library that loads pointers to OpenGL functions at runtime
- ❖ GLM (provided in zip)
  - Math library for OpenGL

# IDE & Kit

## ❖ CMake Tools extension

- Open Visual Studio Code and install the CMake Tools extension.



# Windows

C++

# Step1

- ❖ [download here](#)
- ❖ Scroll down to find 1.20.0 and click Assets

**1.20.0**







- Upgrade to [GCC 13.2](#)
- Upgrade to [busybox-w32 FRP-5007](#)
  - Improved CTRL+c interrupts, esp. in interactive shells like GDB
  - Improved console/terminal handling, esp. in ConEmu

Included software: GCC 13.2.0, busybox-w32 FRP-5007, GDB 13.1, Mingw-w64 11.0.0, GNU Make 4.4, Vim 9.0, Universal Ctags 6.0.0, NASM 2.15.05, binutils 2.40, Cppcheck 2.10.

Notes about builds:

- `-fortran` : includes Fortran ( `gfortran` )
- `-i686` : pure 32-bit, Windows XP compatible, min. req. Pentium 4

**Assets** 11



 44  3  1  12  1  4 57 people reacted

# Step2

- ❖ Download **w64devkit-1.20.0.zip** and extract **w64devkit-1.20.0.zip** to the **C drive**

## ▼ Assets

11

 <a href="#">source.tar</a>	176 MB	Aug 2, 2023
 <a href="#">w64devkit-1.20.0.zip</a>	78.1 MB	Aug 2, 2023
 <a href="#">w64devkit-1.20.0.zip.sig</a>	119 Bytes	Aug 1, 2023
 <a href="#">w64devkit-fortran-1.20.0.zip</a>	88.5 MB	Aug 2, 2023
 <a href="#">w64devkit-fortran-1.20.0.zip.sig</a>	119 Bytes	Aug 1, 2023
 <a href="#">w64devkit-i686-1.20.0.zip</a>	74.1 MB	Aug 2, 2023
 <a href="#">w64devkit-i686-1.20.0.zip.sig</a>	119 Bytes	Aug 1, 2023
 <a href="#">w64devkit-i686-fortran-1.20.0.zip</a>	84.1 MB	Aug 1, 2023
 <a href="#">w64devkit-i686-fortran-1.20.0.zip.sig</a>	119 Bytes	Aug 1, 2023
 <a href="#">Source code (zip)</a>		Aug 1, 2023
 <a href="#">Source code (tar.gz)</a>		Aug 1, 2023



# Step3

❖ Add `C:\w64devkit\bin` to the environment variables



## Step4

- ❖ Open Command Prompt and type `gcc --version` or `g++ --version` to verify that C++ is installed correctly

```
C:\Users\WANG>g++ --version
g++ (GCC) 13.2.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

# CMake

# Step1

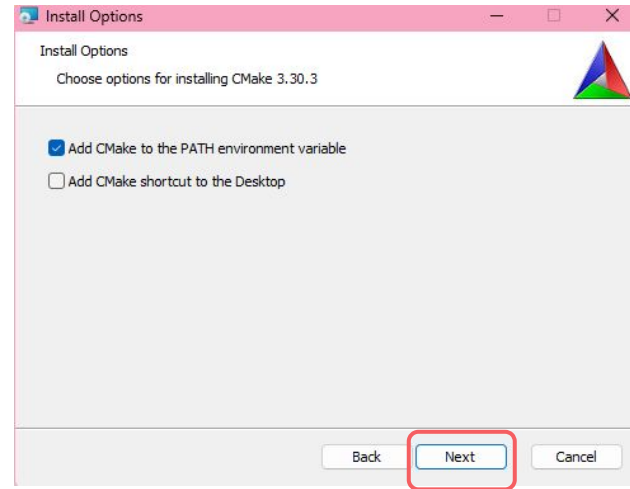
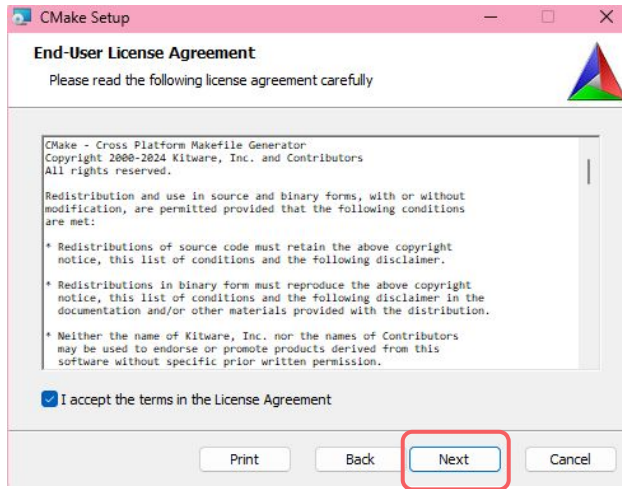
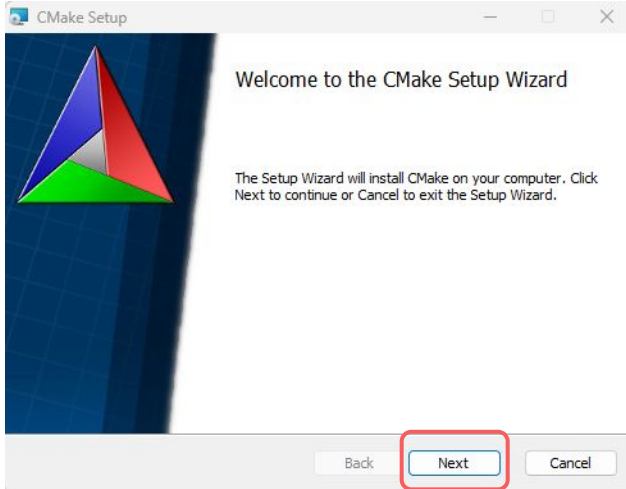
- ❖ [download here](#)
- ❖ Scroll down to find **Windows x64 Installer** and download **cmake-3.30.3-windows-x86\_64.msi**

Binary distributions:

Platform	Files
Windows x64 Installer:	<a href="#">cmake-3.30.3-windows-x86_64.msi</a>
Windows x64 ZIP	<a href="#">cmake-3.30.3-windows-x86_64.zip</a>
Windows i386 Installer:	<a href="#">cmake-3.30.3-windows-i386.msi</a>

# Step2

- ❖ Run **cmake-3.30.3-windows-x86\_64.msi**
  - During the installation, you can keep clicking “Next” to proceed with the default options.



# Linux

## Step1.

### ❖ Install g++

- Refer to your distribution documentation for installation.
- Once you have installed g++, type “g++ –version” in your terminal to check if it’s installed.

## Step2.

### ❖ Install CMake.

- Refer to your distribution documentation for installation.
- Once you have installed CMake, type “cmake –version” in your terminal to check if it’s installed.

## Step3.

- ❖ Install additional dependencies.
  - The additional dependencies would vary base on the windowing system.
- ❖ Check windowing system (wayland / x11)
  - Run the following command in your terminal.  
`echo $XDG_SESSION_TYPE`
- ❖ Base on the windowing system install the need dependencies
  - You can refer to <https://www.glfw.org/docs/3.3/compile.html> for details or follow the bellow guide.



## ❖ For x11 (check the details base on your distribution)

### Dependencies for X11 on Unix-like systems

To compile GLFW for X11, you need to have the X11 development packages installed. They are not needed to build or run programs that use GLFW.

On Debian and derivatives like Ubuntu and Linux Mint the `xorg-dev` meta-package pulls in the development packages for all of X11.

```
sudo apt install xorg-dev
```

On Fedora and derivatives like Red Hat the X11 extension packages `libXcursor-devel`, `libXi-devel`, `libXinerama-devel` and `libXrandr-devel` required by GLFW pull in all its other dependencies.

```
sudo dnf install libXcursor-devel libXi-devel libXinerama-devel libXrandr-devel
```

On FreeBSD the X11 headers are installed along the end-user X11 packages, so if you have an X server running you should have the headers as well. If not, install the `xorgproto` package.

```
pkg install xorgproto
```

On Cygwin the `libXcursor-devel`, `libXi-devel`, `libXinerama-devel`, `libXrandr-devel` and `libXrender-devel` packages in the Libs section of the GUI installer will install all the headers and other development related files GLFW requires for X11.

## ❖ For Wayland (check the details base on your distribution)

### Dependencies for Wayland on Unix-like systems

To compile GLFW for Wayland, you need to have the Wayland and xkbcommon development packages installed. They are not needed to build or run programs that use GLFW.

On Debian and derivatives like Ubuntu and Linux Mint you will need the `libwayland-dev`, `libxkbcommon-dev`, `wayland-protocols` and `extra-cmake-modules` packages. These will pull in all other dependencies.

```
sudo apt install libwayland-dev libxkbcommon-dev wayland-protocols extra-cmake-modules
```

On Fedora and derivatives like Red Hat you will need the `wayland-devel`, `libxkbcommon-devel`, `wayland-protocols-devel` and `extra-cmake-modules` packages.

```
sudo dnf install wayland-devel libxkbcommon-devel wayland-protocols-devel extra-cmake-modules
```

On FreeBSD you will need the `wayland`, `libxkbcommon`, `wayland-protocols`, `evdev-proto` and `kf5-extra-cmake-modules` packages.

```
pkg install wayland libxkbcommon wayland-protocols evdev-proto kf5-extra-cmake-modules
```

# Mac

# Step1. (Only need to install Cmake)

## ❖ Install Cmake (installed using Homebrew)

- Run the following command in your terminal.  
`brew install cmake`
- Once you have installed Cmake, type “cmake –version” in your terminal to check if it's installed.

## ❖ If you do not have Homebrew installed

- Run the following command in your terminal.  
`/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"`
- Once you have installed Homebrew, type “brew” in your terminal to check if it's installed.

- ❖ More installation details (if needed)

<https://cse.engineering.nyu.edu/cs653/OpenGLCompilationMacLinux8.pdf>

# Demo & test

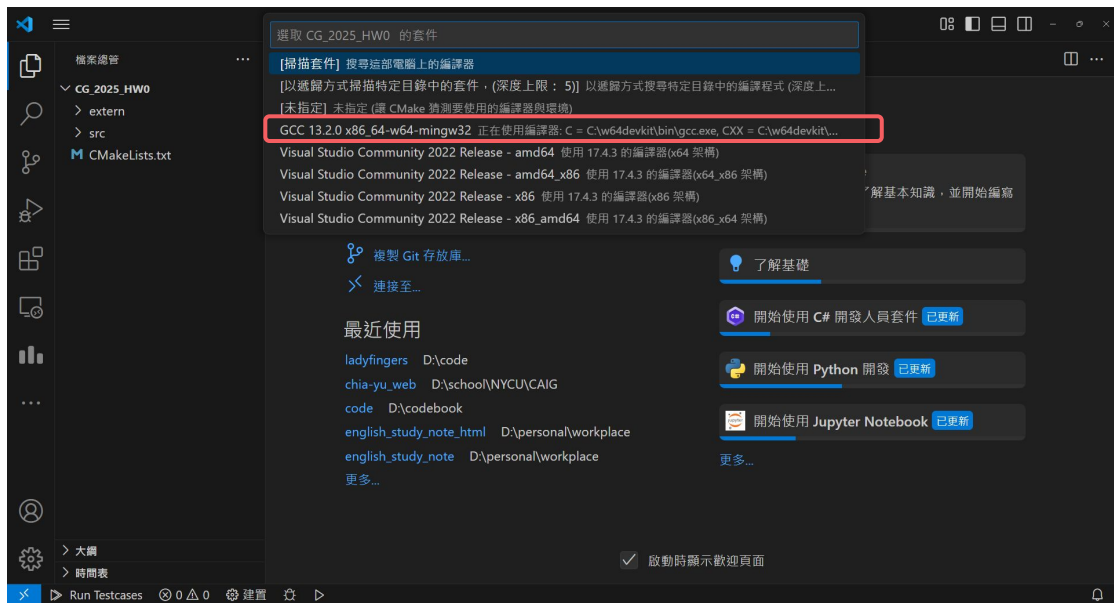
# Step1 & 2

- ❖ Step1 : Extract **CG\_2025\_HW0.zip** to any location
  - Unzip the file to a folder of your choice.
- ❖ Step2 : Open Visual Studio Code and select the **CG\_2025\_HW0** folder.
  - Launch Visual Studio Code, then use the “Open Folder” option to navigate to and select the **CG\_2025\_HW0** folder

# Step3

## ❖ After opening, select the corresponding g++ package

- Once you open the project in Visual Studio Code, it will prompt you to select the appropriate g++ package. Choose the previously installed g++ compiler located at `C:\w64devkit\bin\gcc.exe`.





# Step4

- ❖ Press the **Run button** in the bottom left corner
  - After selecting the g++ compiler, simply press the "Run" button in the bottom left corner of VS Code to launch the project.

