Ubuntu 20.04 LTS software setup guide

Installing the required tools

0. Install the required build tools and a modern C++ compiler (g++ 10 or newer)

```
sudo apt install build-essential cmake g++-8
```

1. Install the OpenGL development libraries

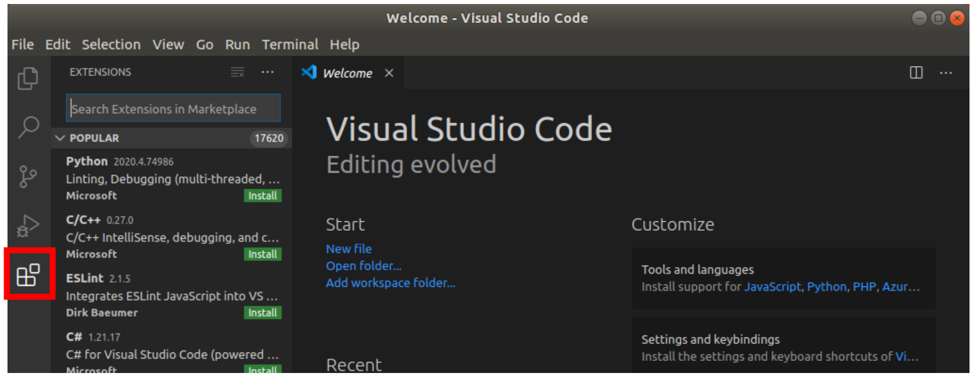
```
sudo apt install libx11-dev libxrandr-dev libxinerama-dev libxcursor-dev libxi-dev libgl1-mesa-dev mesa-utils xlibmesa-glu-dev
```

Running the code

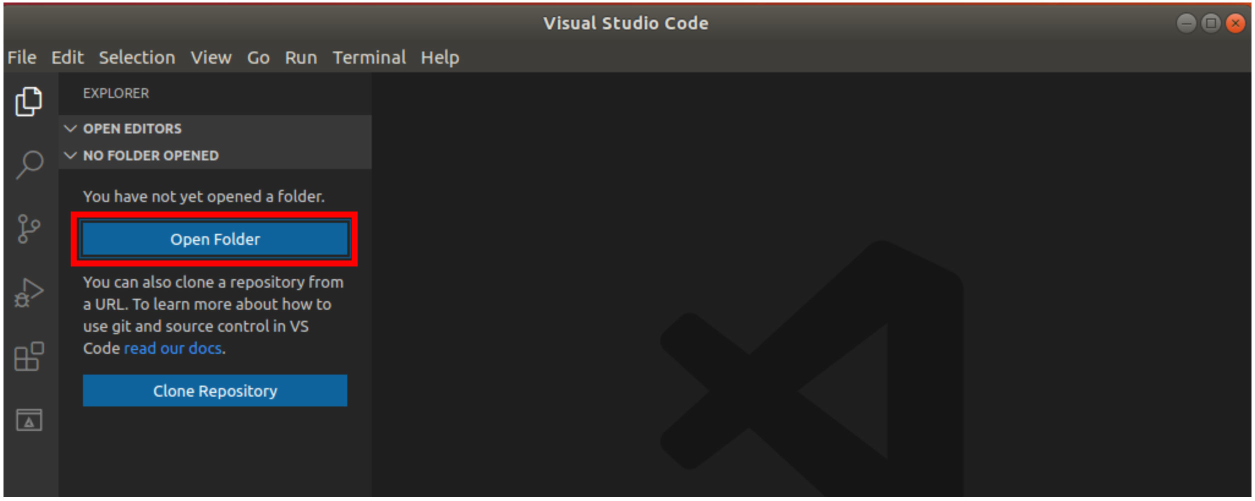
Building & debugging using VS Code (recommended)

0. Install Visual Studio Code from the Ubuntu Software store

1. Open Visual Studio code and install the “C/C++” and “CMake Tools” extensions by Microsoft


Welcome - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXTENSIONS
Search Extensions in Marketplace
POPULAR 17620
Python 2020.4.74986
Linting, Debugging (multi-threaded, ...
Microsoft
Install
C/C++ 0.27.0
C/C++ IntelliSense, debugging, and c...
Microsoft
Install
ESLint 2.1.5
Integrates ESLint JavaScript into VS ...
Dirk Baeumer
Install
C# 1.21.17
C# for Visual Studio Code (powered ...
Microsoft
Install
Welcome
Visual Studio Code
Editing evolved
Start
New file
Open folder...
Add workspace folder...
Recent
Customize
Tools and languages
Install support for JavaScript, Python, PHP, Azur...
Settings and keybindings
Install the settings and keyboard shortcuts of Vi...

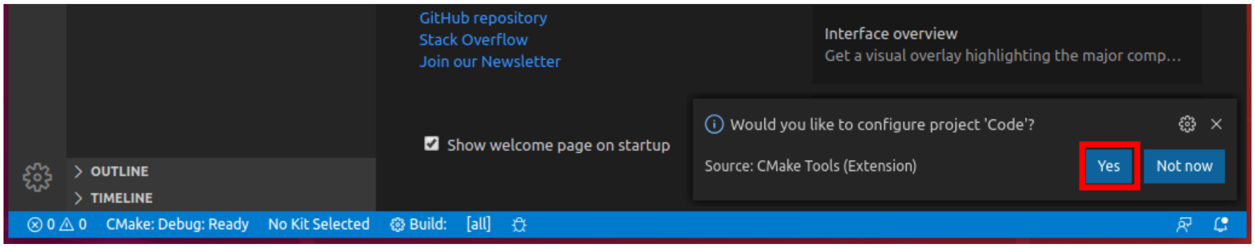
2. In the main view click on “Open Folder” (or “File” => “Open Folder”) and select the root of the project (where your files and the CMakeLists.txt file live).



You have not yet opened a folder.

You can also clone a repository from a URL. To learn more about how to use git and source control in VS Code read our docs.

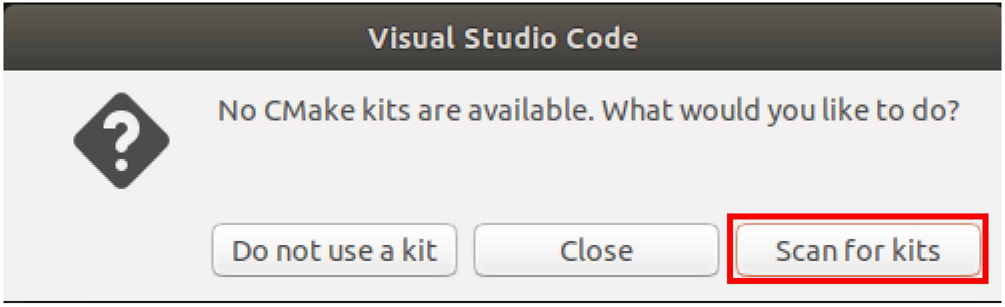
3. After opening the folder a popup should show up asking to configure the project. Click “Yes”.


GitHub repository
Stack Overflow
Join our Newsletter
Interface overview
Get a visual overlay highlighting the major comp...
☑ Show welcome page on startup
OUTLINE
TIMELINE
Yes
Not now
⊗ 0 △ 0 CMake: Debug: Ready No Kit Selected Build: [all]

Would you like to configure project 'Code'?

Source: CMake Tools (Extension)

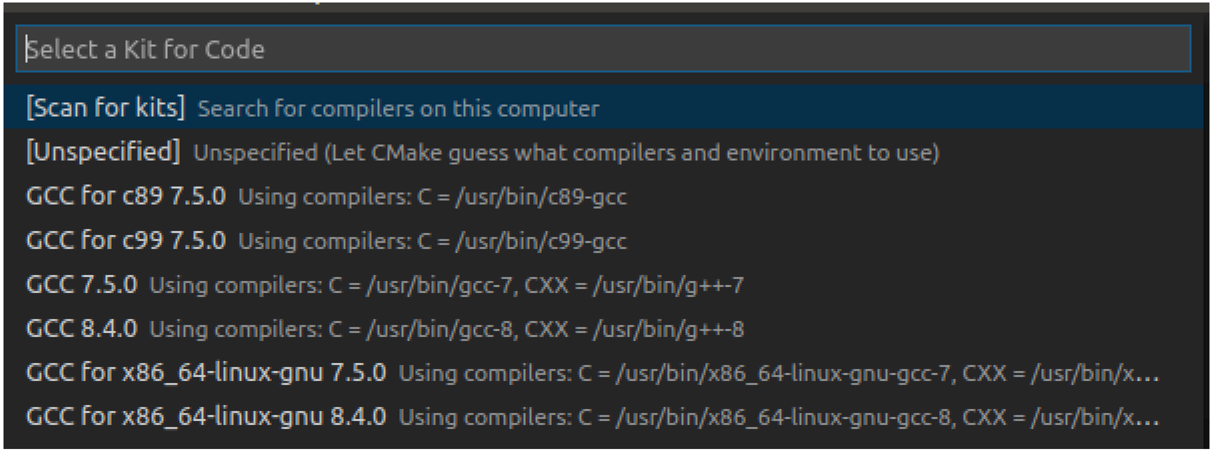
4. The first-time running VS Code, it will ask about CMake kits. Click “Scan for kits”.



Visual Studio Code

No CMake kits are available. What would you like to do?

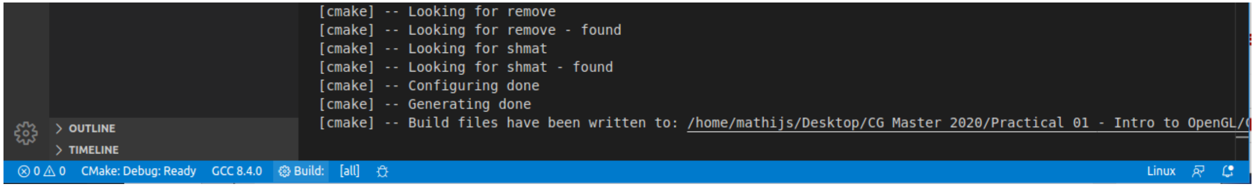
After scanning you will be asked to select a kit. Select the latest version of GCC that is installed (10.0.0 or higher).



Select a Kit for Code

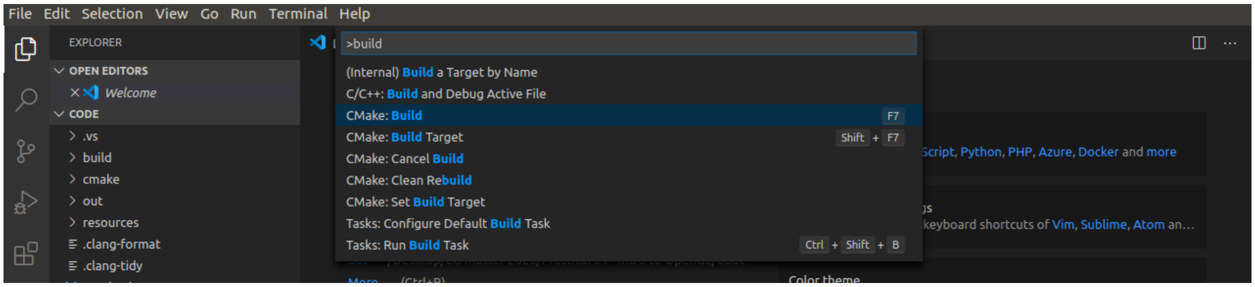
[Scan for kits] Search for compilers on this computer
[Unspecified] Unspecified (Let CMake guess what compilers and environment to use)
GCC for c89 7.5.0 Using compilers: C = /usr/bin/c89-gcc
GCC for c99 7.5.0 Using compilers: C = /usr/bin/c99-gcc
GCC 7.5.0 Using compilers: C = /usr/bin/gcc-7, CXX = /usr/bin/g++-7
GCC 8.4.0 Using compilers: C = /usr/bin/gcc-8, CXX = /usr/bin/g++-8
GCC for x86_64-linux-gnu 7.5.0 Using compilers: C = /usr/bin/x86_64-linux-gnu-gcc-7, CXX = /usr/bin/x...
GCC for x86_64-linux-gnu 8.4.0 Using compilers: C = /usr/bin/x86_64-linux-gnu-gcc-8, CXX = /usr/bin/x...

5. CMake will now start configuring your project. We set it up to download and compile some third-party dependencies. This may take a couple of minutes when running it for the first time. Once CMake is finished it should show something like this:



```
[cmake] -- Looking for remove
[cmake] -- Looking for remove - found
[cmake] -- Looking for shmat
[cmake] -- Looking for shmat - found
[cmake] -- Configuring done
[cmake] -- Generating done
[cmake] -- Build files have been written to: /home/mathijs/Desktop/CG Master 2020/Practical 01 - Intro to OpenGL/(
```

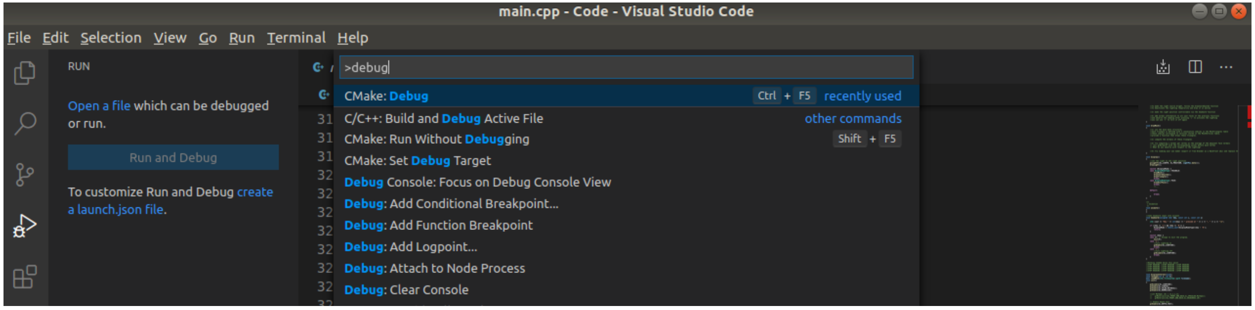
6. To compile the project, open the Command Palette (ctrl + shift + p) and search for “CMake: Build” OR use the shortcut F7.



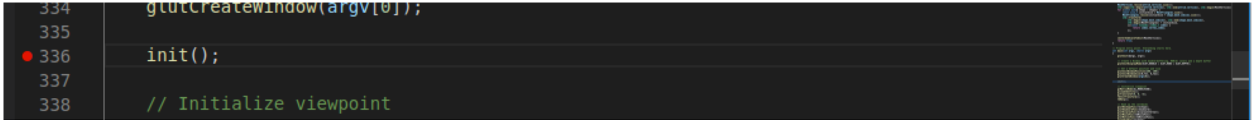
>build

(Internal) Build a Target by Name
C/C++: Build and Debug Active File
CMake: Build F7
CMake: Build Target Shift + F7
CMake: Cancel Build
CMake: Clean Rebuild
CMake: Set Build Target
Tasks: Configure Default Build Task
Tasks: Run Build Task Ctrl + Shift + B

7. To run the project, use the Command Palette (ctrl + shift + p) and search for "CMake: Debug" OR use the shortcut ctrl + F5.


main.cpp - Code - Visual Studio Code
File Edit Selection View Go Run Terminal Help
RUN
Open a file which can be debugged or run.
Run and Debug
To customize Run and Debug create a launch.json file.
>debug
CMake: Debug Ctrl + F5 recently used
C/C++: Build and Debug Active File other commands
CMake: Run Without Debugging Shift + F5
CMake: Set Debug Target
Debug Console: Focus on Debug Console View
Debug: Add Conditional Breakpoint...
Debug: Add Function Breakpoint
Debug: Add Logpoint...
Debug: Attach to Node Process
Debug: Clear Console

8. To set breakpoints in your C++ code. Click left of the number of the line that you want to set a breakpoint on.

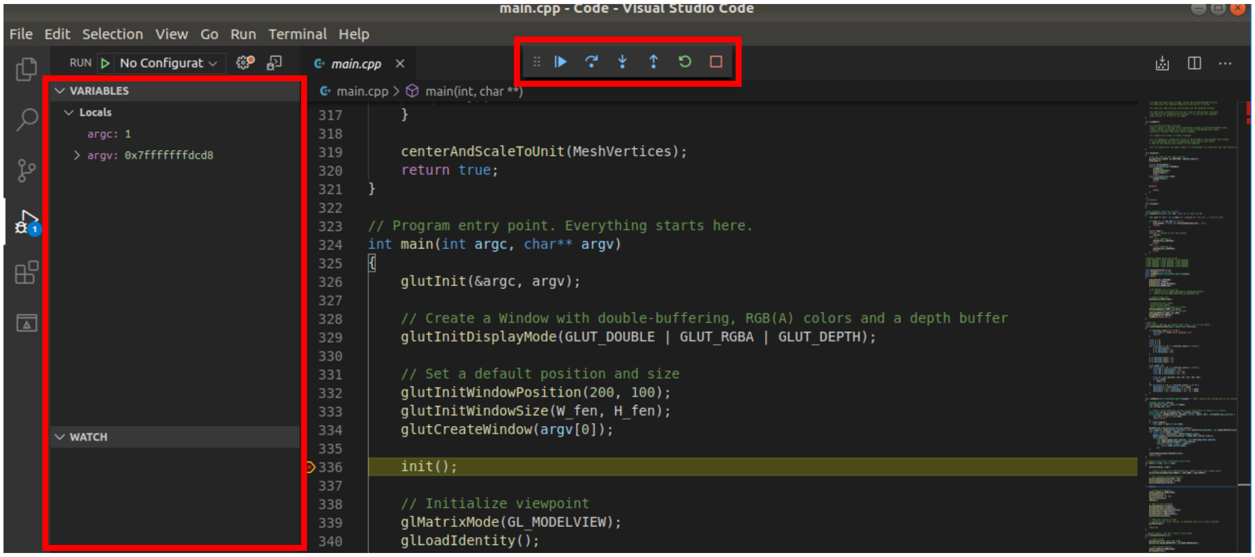


```
    glutCreateWindow(argv[0]);

    init();

    // Initialize viewpoint
```

When you run your code (step 7) it should now stop at the breakpoint and show you the values of local variables. Use the buttons at the top to step into, out of or over a function.


main.cpp - Code - Visual Studio Code
File Edit Selection View Go Run Terminal Help
RUN ▷ No Configurat
main.cpp
main.cpp > main(int, char **)
VARIABLES
Locals
WATCH

```
argc: 1
> argv: 0x7fffffffdcd8
```

```
    }

    centerAndScaleToUnit(MeshVertices);
    return true;
}

// Program entry point. Everything starts here.
int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    // Create a Window with double-buffering, RGB(A) colors and a depth buffer
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);

    // Set a default position and size
    glutInitWindowPosition(200, 100);
    glutInitWindowSize(W_fen, H_fen);
    glutCreateWindow(argv[0]);

    init();

    // Initialize viewpoint
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
```

Building & running using the terminal

0. Download, unzip the code and cd into the root directory of the project (the directory where your code lives)

1. Create a build folder and go into it

```
mkdir build
cd build
```

2. Let CMake configure the project and generate a Makefile (run once)
   ```
   cmake -DCMAKE_BUILD_TYPE=Debug -DCMAKE_CXX_COMPILER=g++-10 ../
   ```

3. Compile the project (run this every time you change your code). Replace 6 by the number of cores in your CPU.
   ```
   make -j6
   ```

4. Run the code
   ```
   ./Practical01
   ```