Environment Setup for MacOS

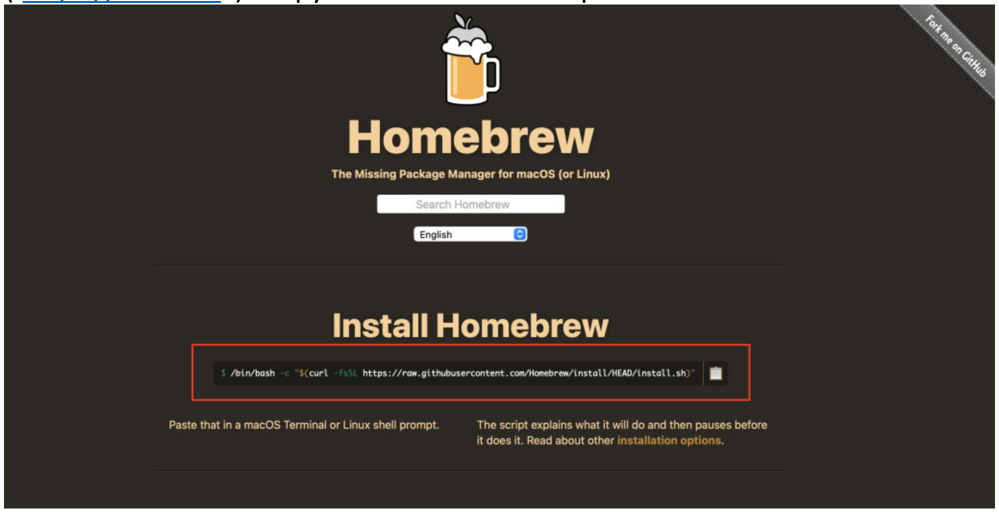
Professor: Dr. Xiaohu Guo

1. Download and install Homebrew

   The requirement for Homebrew:

   a. A 64-bit Intel CPU or Apple Silicon CPU
   b. macOS Mojave (10.14) (or higher)
   c. Command Line Tools (CLT) for Xcode: xcode-select install, developer.apple.com/downloads or Xcode
   d. A Bourne-compatible shell for installation (e.g. bash or zsh)

   Once the said requirements are satisfied, open homepage of Homebrew (https://brew.sh). Copy this command and paste it into Terminal:



```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Paste that in a macOS Terminal or Linux shell prompt.

The script explains what it will do and then pauses before it does it. Read about other installation options.

2. Install required compilers and packages by running following commands

   a. Install gcc and cmake:

```
% brew install gcc cmake
```

   b. Install Eigen and Opencv:

```
% brew install eigen opencv
```

   c. Link Eigen by:

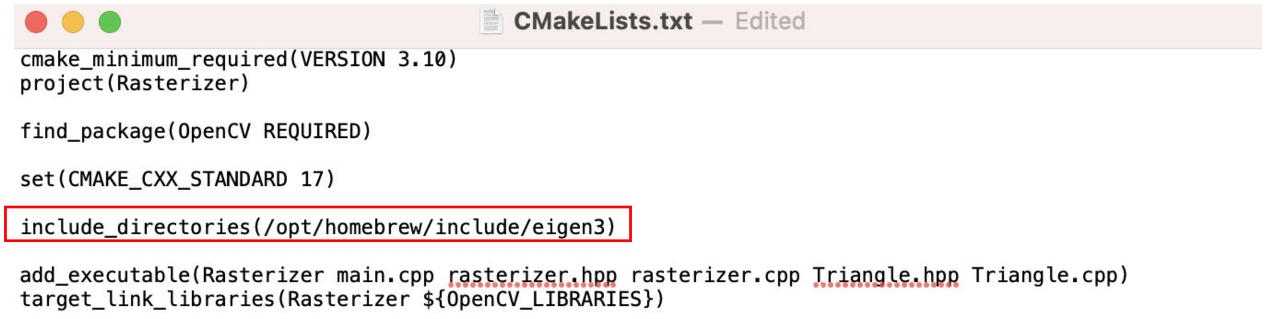
```
% brew link --overwrite  eigen
```

d. Restart Terminal

3. Change configuration (optional)

If you are using Apple Silicon chips (like M1,M2,M3)
Open the CMakeList.txt and change include_directories to
/opt/homebrew/include/eigen3

If you are using Intel chip, the include_directories will be:
/usr/local/include/eigen3

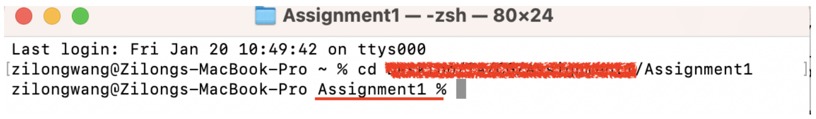


```
cmake_minimum_required(VERSION 3.10)
project(Rasterizer)

find_package(OpenCV REQUIRED)

set(CMAKE_CXX_STANDARD 17)

include_directories(/opt/homebrew/include/eigen3)

add_executable(Rasterizer main.cpp rasterizer.hpp rasterizer.cpp Triangle.hpp Triangle.cpp)
target_link_libraries(Rasterizer ${OpenCV_LIBRARIES})
```

4. Compiling and Testing

a. Download assignment and open it in the Terminal:

```
% cd ~/* path to your assignment*/assignment1
```

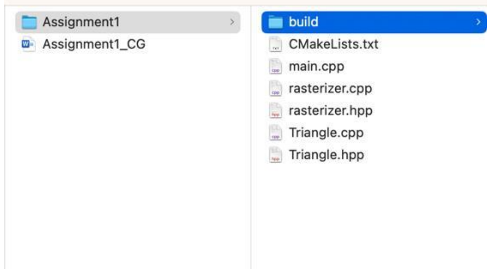


```
Last login: Fri Jan 20 10:49:42 on ttys000
zilongwang@Zilongs-MacBook-Pro ~ % cd [redacted]/Assignment1
zilongwang@Zilongs-MacBook-Pro Assignment1 %
```

b. Create new folder named build and open it

```
% mkdir build
% cd build
```

Note: this fold will located in same folder of all .cpp files

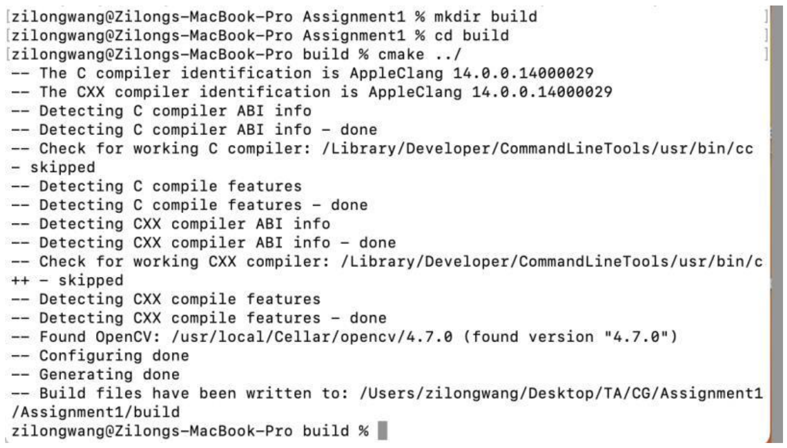


Assignment1
Assignment1_CG

build
CMakeLists.txt
main.cpp
rasterizer.cpp
rasterizer.hpp
Triangle.cpp
Triangle.hpp

c. Create Cmake configurations in build folder

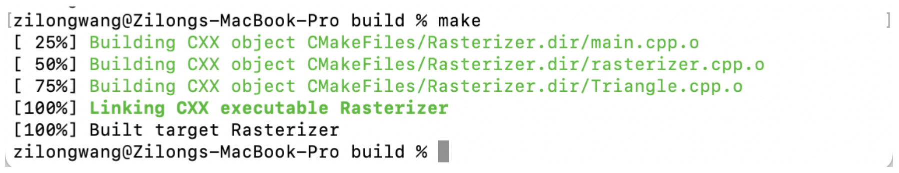
% Cmake ../



```
zilongwang@Zilongs-MacBook-Pro Assignment1 % mkdir build
zilongwang@Zilongs-MacBook-Pro Assignment1 % cd build
zilongwang@Zilongs-MacBook-Pro build % cmake ../
-- The C compiler identification is AppleClang 14.0.0.14000029
-- The CXX compiler identification is AppleClang 14.0.0.14000029
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /Library/Developer/CommandLineTools/usr/bin/cc
- skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /Library/Developer/CommandLineTools/usr/bin/c
++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found OpenCV: /usr/local/Cellar/opencv/4.7.0 (found version "4.7.0")
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/zilongwang/Desktop/TA/CG/Assignment1
/Assignment1/build
zilongwang@Zilongs-MacBook-Pro build %
```

d. Build executable file:

% make

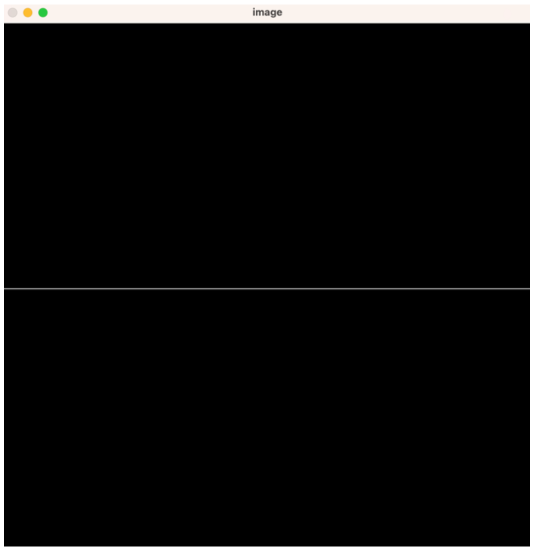


```
zilongwang@Zilongs-MacBook-Pro build % make
[ 25%] Building CXX object CMakeFiles/Rasterizer.dir/main.cpp.o
[ 50%] Building CXX object CMakeFiles/Rasterizer.dir/rasterizer.cpp.o
[ 75%] Building CXX object CMakeFiles/Rasterizer.dir/Triangle.cpp.o
[100%] Linking CXX executable Rasterizer
[100%] Built target Rasterizer
zilongwang@Zilongs-MacBook-Pro build %
```

Then if there is no error, you will find a executable file in the build folder, you can test it by running:

% ./Rasterizer

A window will pop up with rendering result of your assignment. For default assignment 1, it will look like this:

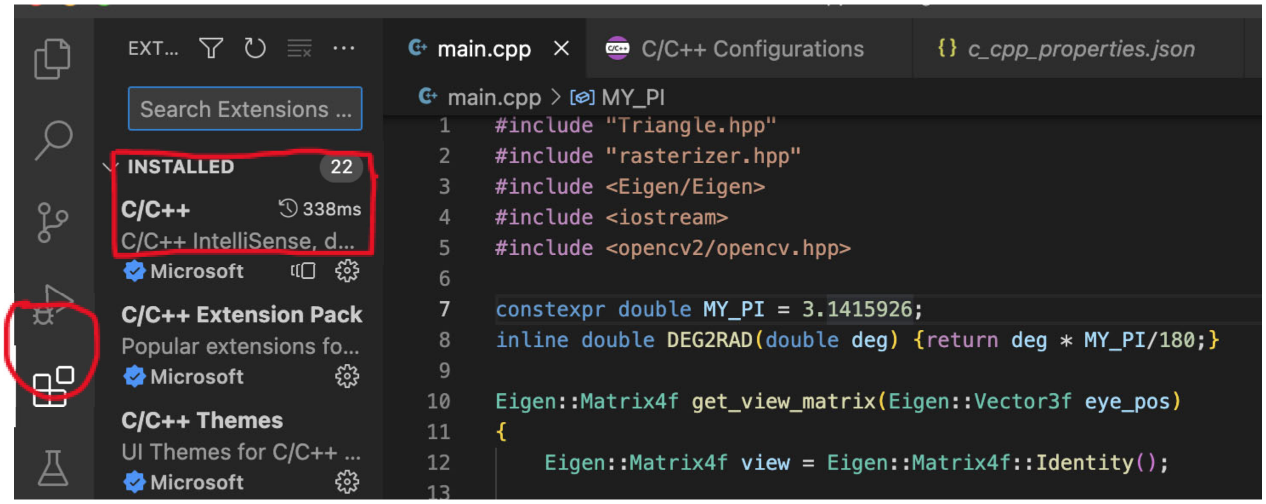


Coding

Now, you can start to code using VScode.

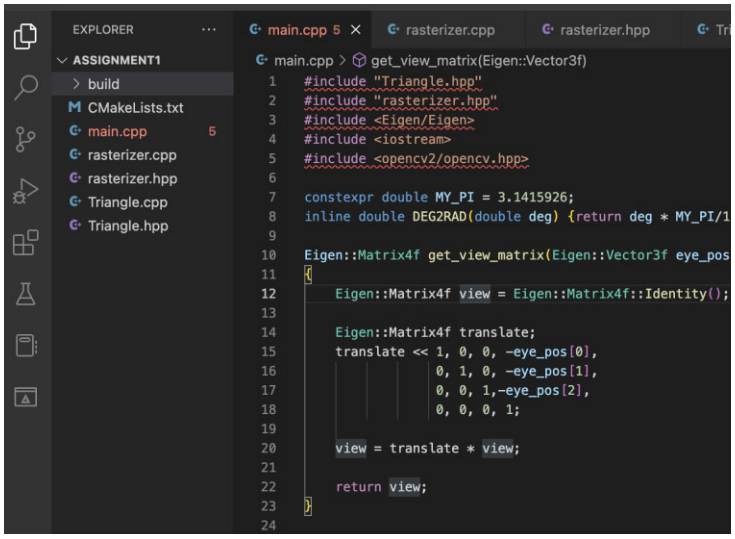
(if you don't have VScode installed, just go to their website and follow the instruction.)

Open up Vscode and open the assignment's folder Check extension, make sure C/C++ has installed



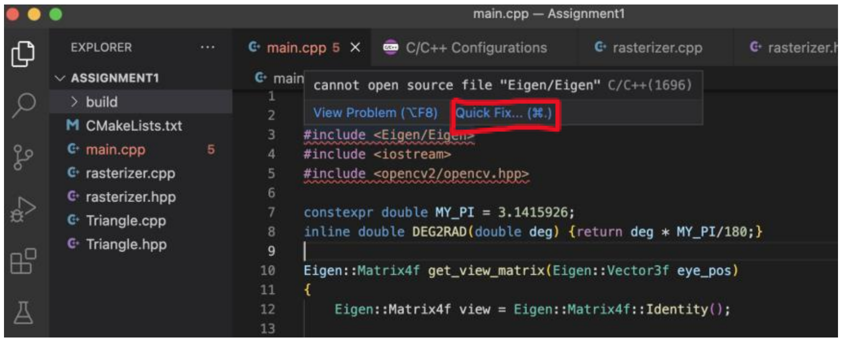
```cpp
#include "Triangle.hpp"
#include "rasterizer.hpp"
#include <Eigen/Eigen>
#include <iostream>
#include <opencv2/opencv.hpp>

constexpr double MY_PI = 3.1415926;
inline double DEG2RAD(double deg) {return deg * MY_PI/180;}

Eigen::Matrix4f get_view_matrix(Eigen::Vector3f eye_pos)
{
    Eigen::Matrix4f view = Eigen::Matrix4f::Identity();
```

Back to files. While you open main.cpp, you will notice that there are errors. It is because we didn't setup environment within VScode.



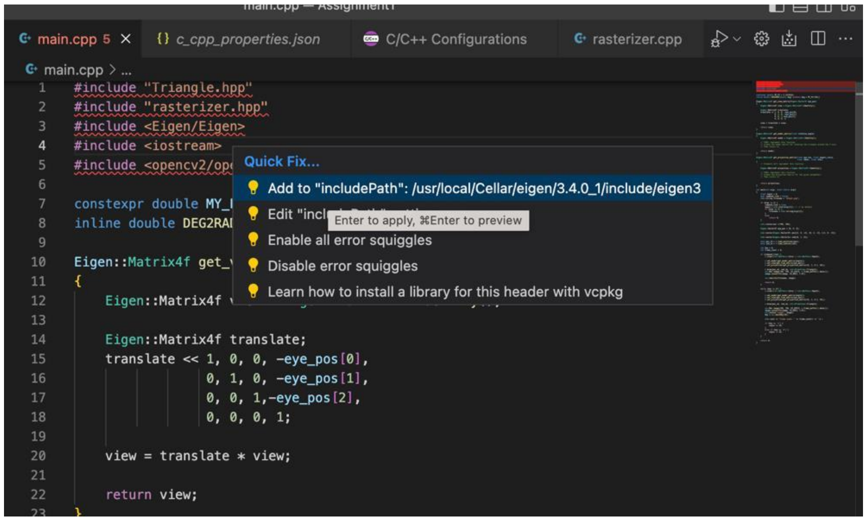
```
1   #include "Triangle.hpp"
2   #include "rasterizer.hpp"
3   #include <Eigen/Eigen>
4   #include <iostream>
5   #include <opencv2/opencv.hpp>
6
7   constexpr double MY_PI = 3.1415926;
8   inline double DEG2RAD(double deg) {return deg * MY_PI/1
9
10  Eigen::Matrix4f get_view_matrix(Eigen::Vector3f eye_pos
11  {
12      Eigen::Matrix4f view = Eigen::Matrix4f::Identity();
13
14      Eigen::Matrix4f translate;
15      translate << 1, 0, 0, -eye_pos[0],
16                   0, 1, 0, -eye_pos[1],
17                   0, 0, 1,-eye_pos[2],
18                   0, 0, 0, 1;
19
20      view = translate * view;
21
22      return view;
23  }
24
```

Then you can simply move cursor on the 3rd and 5th line, and click Quick Fix



```
cannot open source file "Eigen/Eigen" C/C++(1696)

3   #include <Eigen/Eigen>
4   #include <iostream>
5   #include <opencv2/opencv.hpp>
6
7   constexpr double MY_PI = 3.1415926;
8   inline double DEG2RAD(double deg) {return deg * MY_PI/180;}
9
10  Eigen::Matrix4f get_view_matrix(Eigen::Vector3f eye_pos)
11  {
12      Eigen::Matrix4f view = Eigen::Matrix4f::Identity();
13
```

Then select first option, it will automatically create a properties file for C and add installed package in it,


main.cpp — Assignment1
main.cpp 5
c_cpp_properties.json
C/C++ Configurations
rasterizer.cpp
main.cpp > ...
Quick Fix...
Add to "includePath": /usr/local/Cellar/eigen/3.4.0_1/include/eigen3
Edit "incl
Enter to apply, ⌘Enter to preview
Enable all error squiggles
Disable error squiggles
Learn how to install a library for this header with vcpkg

```
1   #include "Triangle.hpp"
2   #include "rasterizer.hpp"
3   #include <Eigen/Eigen>
4   #include <iostream>
5   #include <opencv2/ope
6
7   constexpr double MY_
8   inline double DEG2RAD
9
10  Eigen::Matrix4f get_
11  {
12      Eigen::Matrix4f
13
14      Eigen::Matrix4f translate;
15      translate << 1, 0, 0, -eye_pos[0],
16                   0, 1, 0, -eye_pos[1],
17                   0, 0, 1,-eye_pos[2],
18                   0, 0, 0, 1;
19
20      view = translate * view;
21
22      return view;
```

If it doesn't auto-filled the path, then you need: click includePath -> find include path section in c/c++ configuration -> add path

For M-series chip:
/opt/homebrew/Cellar/eigen/3.4.0_1/include/eigen3
/opt/homebrew/Cellar/opencv/4.7.0/include/opencv4

For intel chip:
/opt/homebrew/Cellar/eigen/3.4.0_1/include/eigen3
/opt/homebrew/Cellar/opencv/4.7.0/include/opencv4

Caution: make sure the OpenCV version is correct
check your OpenCV version by running -> brew info opencv

IMPORTANT

1. Every time you modified codes, you **have to** re-"make" (2-d) a new executable file, which is created with your new set of codes, for getting result.

2. Always running your .exe file by Terminal in build folder, which will make sure output image saved in build folder. Otherwise, it will be saved to root folder of your Mac if you directly open .exe file.