Computer Graphics I Tutorial: Hello OpenGL

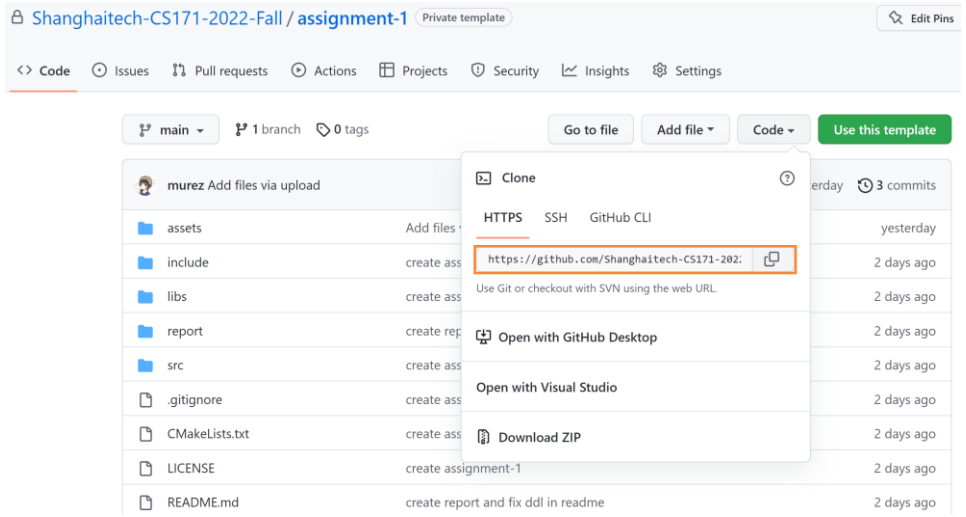
TA: Yiheng Wu, Zhanrui Zhang, Siyuan Zhang

Outline

- CMake
- OpenGL, glfw, glad, and glm
- How to create a window?
- How to draw a triangle?
- How to draw a rectangle?
- How to draw a colorful rectangle?
- How to get time and input?
- How to finish HW1?

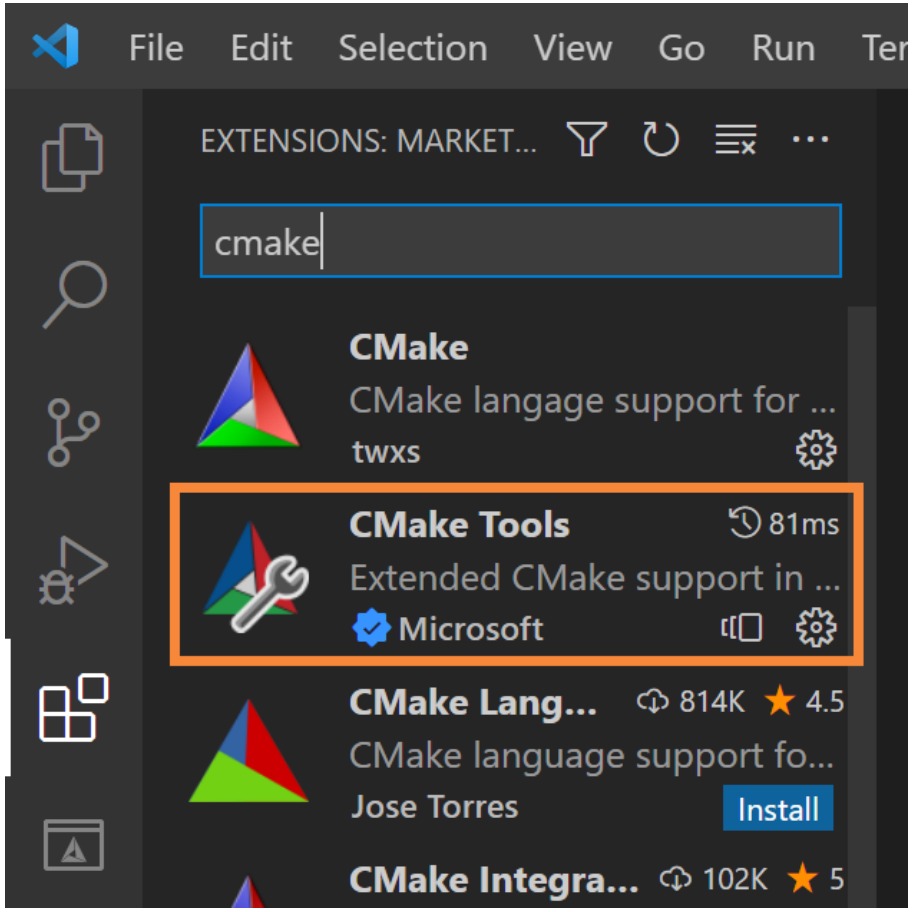
CMake on Windows

- Step1: git clone



CMake on Windows

- Step1: git clone
- Step2: vscode
  - Download this CMake plugin



CMake on Windows

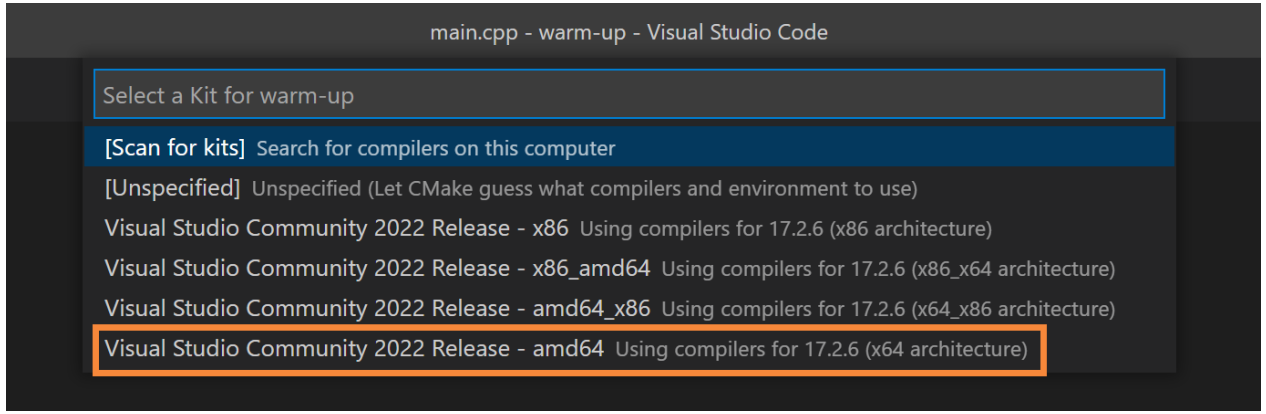
- Step1: git clone
- Step2: vscode
  - Download this CMake plugin
  - Click “File”, “Open Folder”, and select your folder

CMake on Windows

- Step1: git clone
- Step2: vscode
  - Download this CMake plugin
  - Click “File”, “Open Folder”, and select your folder
  - Click “No Kit Selected”

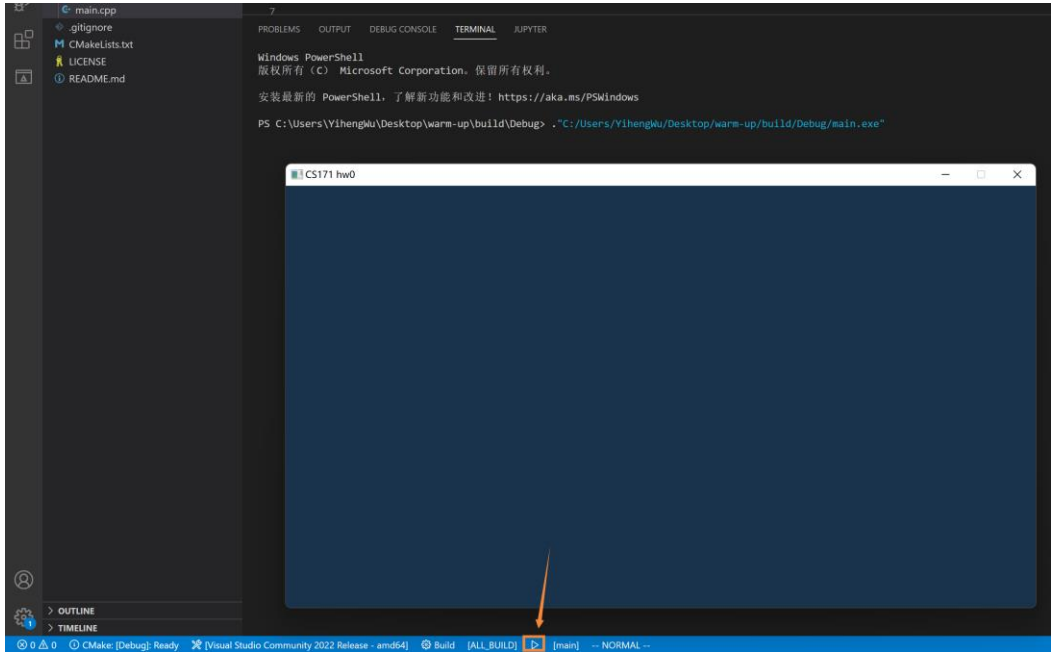
CMake on Windows

- Step1: git clone
- Step2: vscode
  - Download this CMake plugin
  - Click “File”, “Open Folder”, and select your folder
  - Click “No Kit Selected”
  - Select “amd64” here



CMake on Windows

- Step1: git clone
- Step2: vscode
  - Download this CMake plugin
  - Click “File”, “Open Folder”, and select your folder
  - Click “No Kit Selected”
  - Select “amd64” here
  - Click the little triangle here

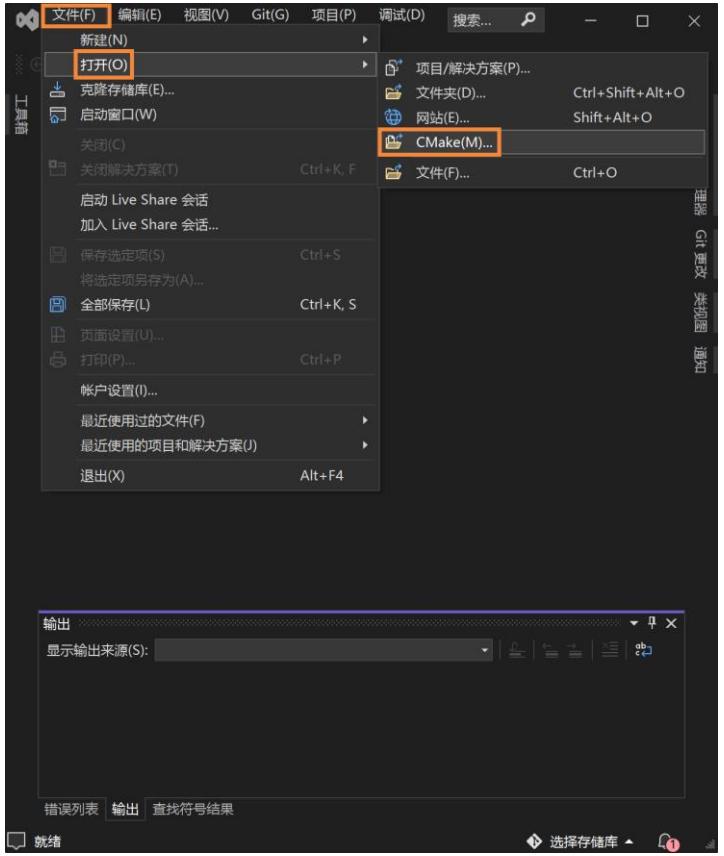


CMake on Windows

- Step1: git clone
- Step2: vs
  - Click here

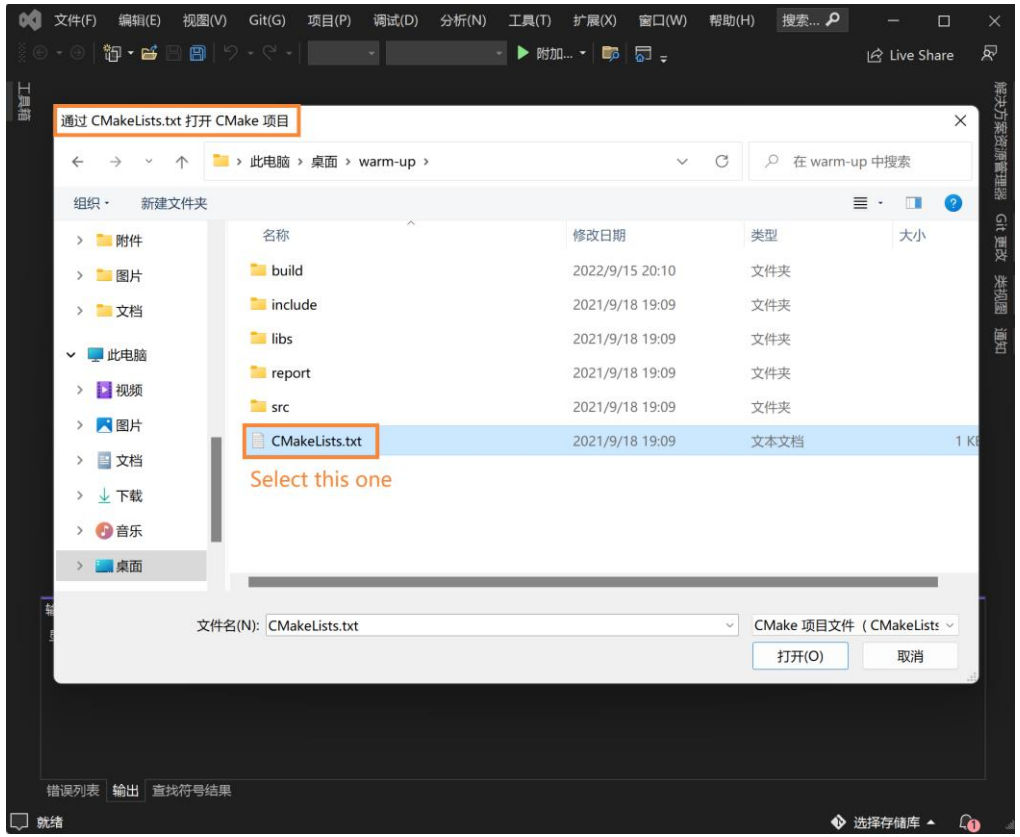
CMake on Windows

- Step1: git clone
- Step2: vs
  - Click here
  - Click “File”, “Open”, “CMake”



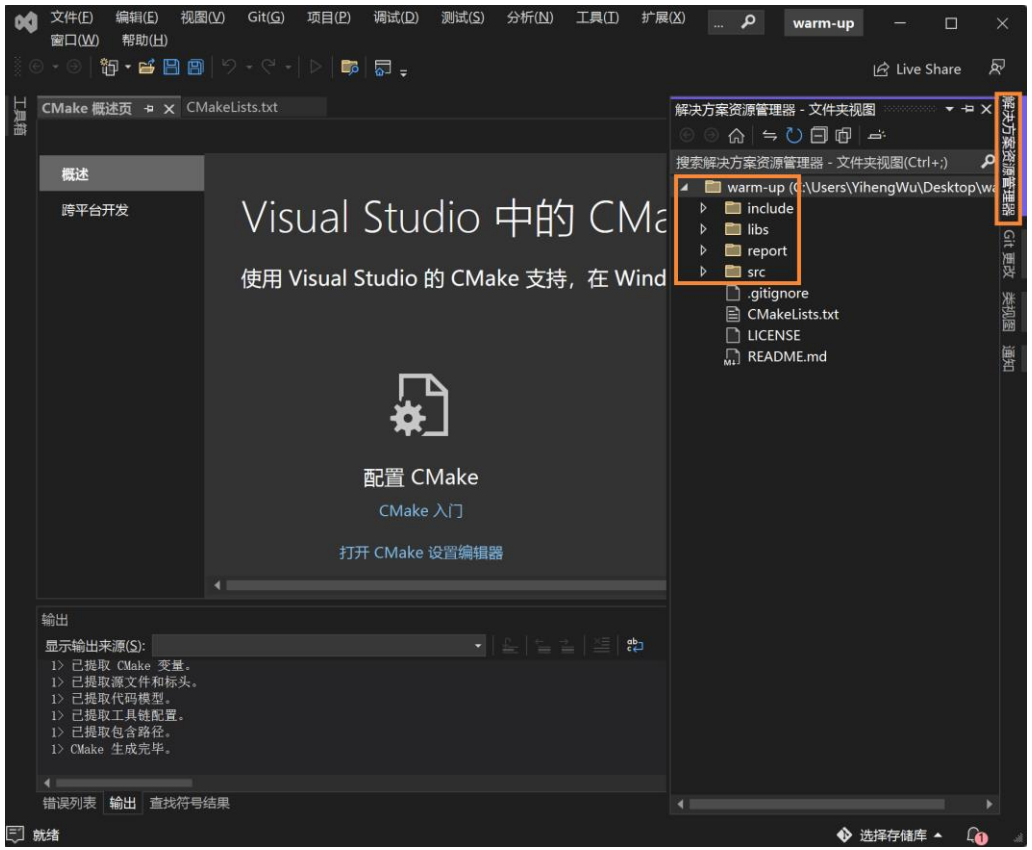
CMake on Windows

- Step1: git clone
- Step2: vs
  - Click here
  - Click “File”, “Open”, “CMake”
  - Select your “CMakeLists.txt” and open



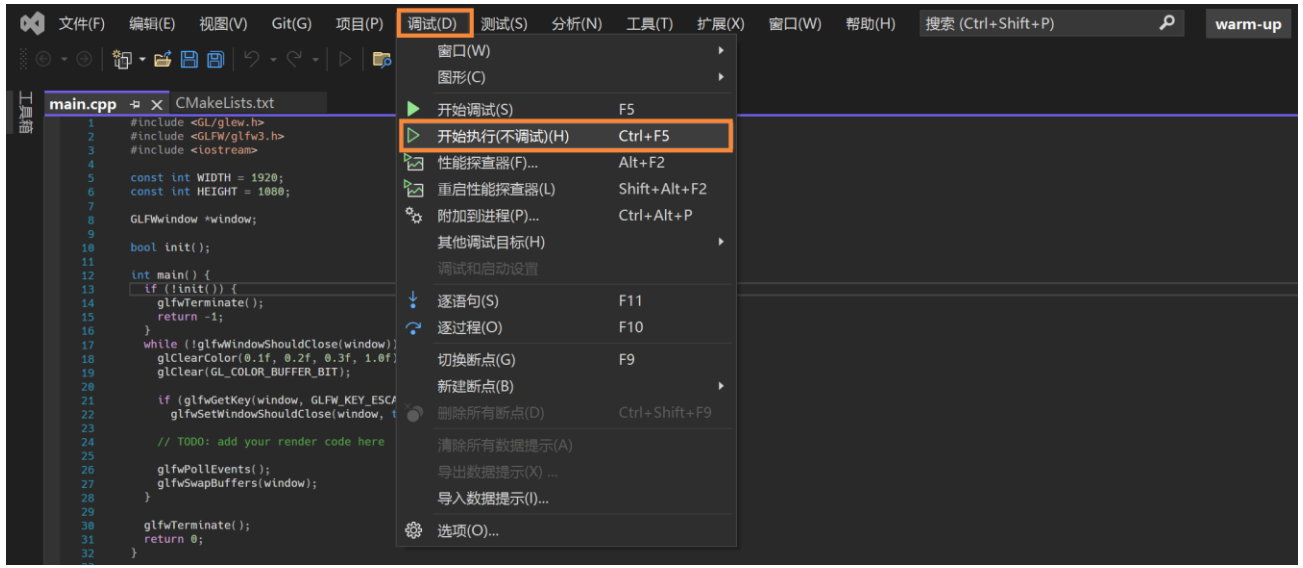
CMake on Windows

- Step1: git clone
- Step2: vs
  - Click here
  - Click “File”, “Open”, “CMake”
  - Select your “CMakeLists.txt” and open
  - You can find all the files here



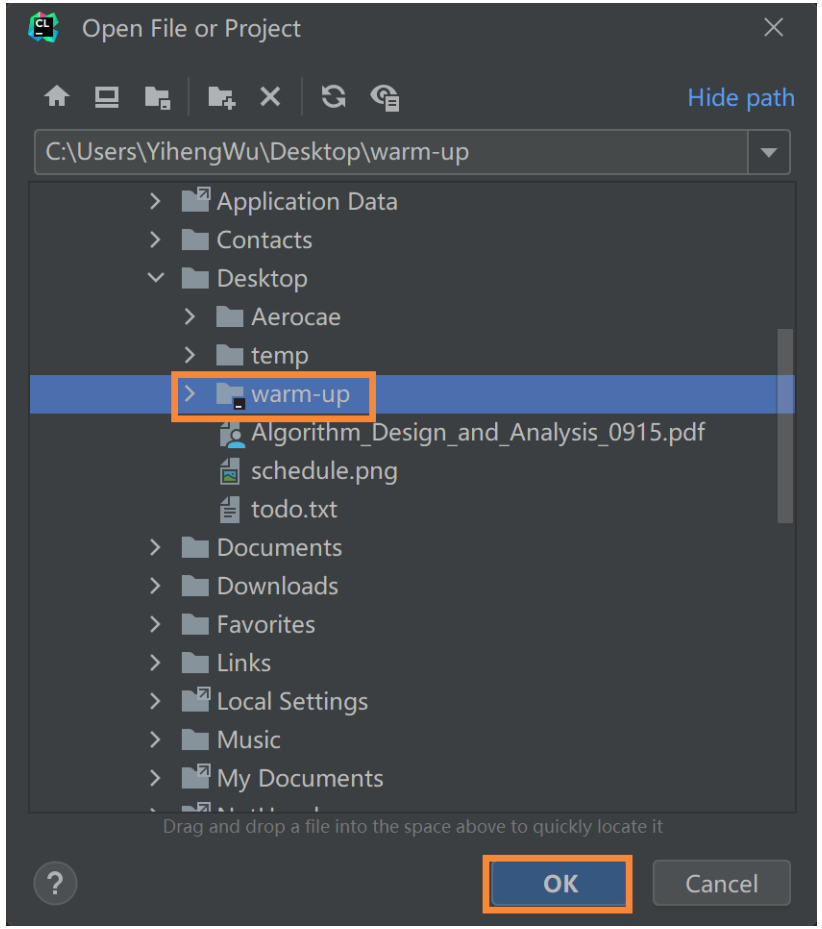
CMake on Windows

- Step1: git clone
- Step2: vs
  - Click here
  - Click “File”, “Open”, “CMake”
  - Select your “CMakeLists.txt” and open
  - You can find all the files here
  - Click here to run



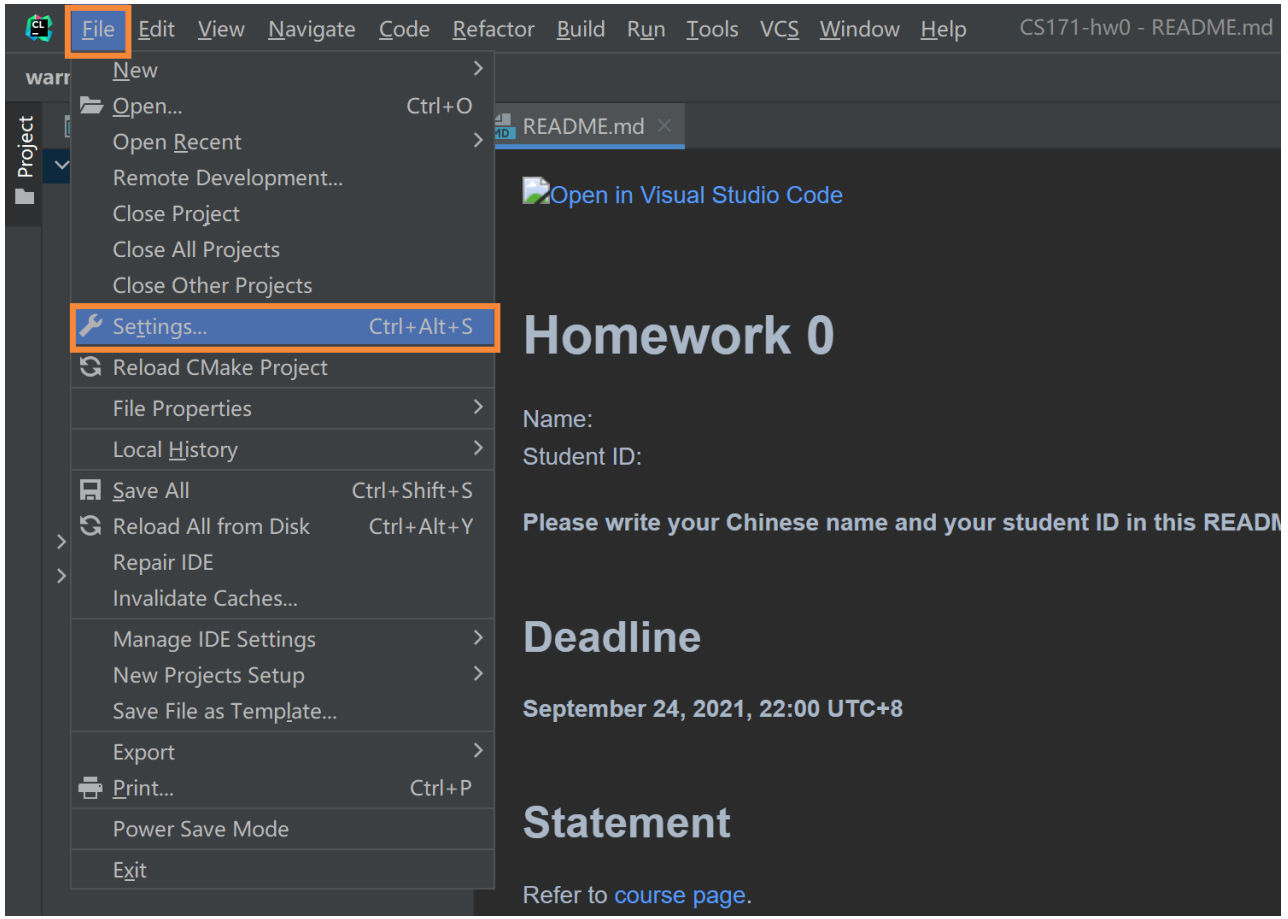
CMake on Windows

- Step1: git clone
- Step2: clion
  - Click “Open File or Project”



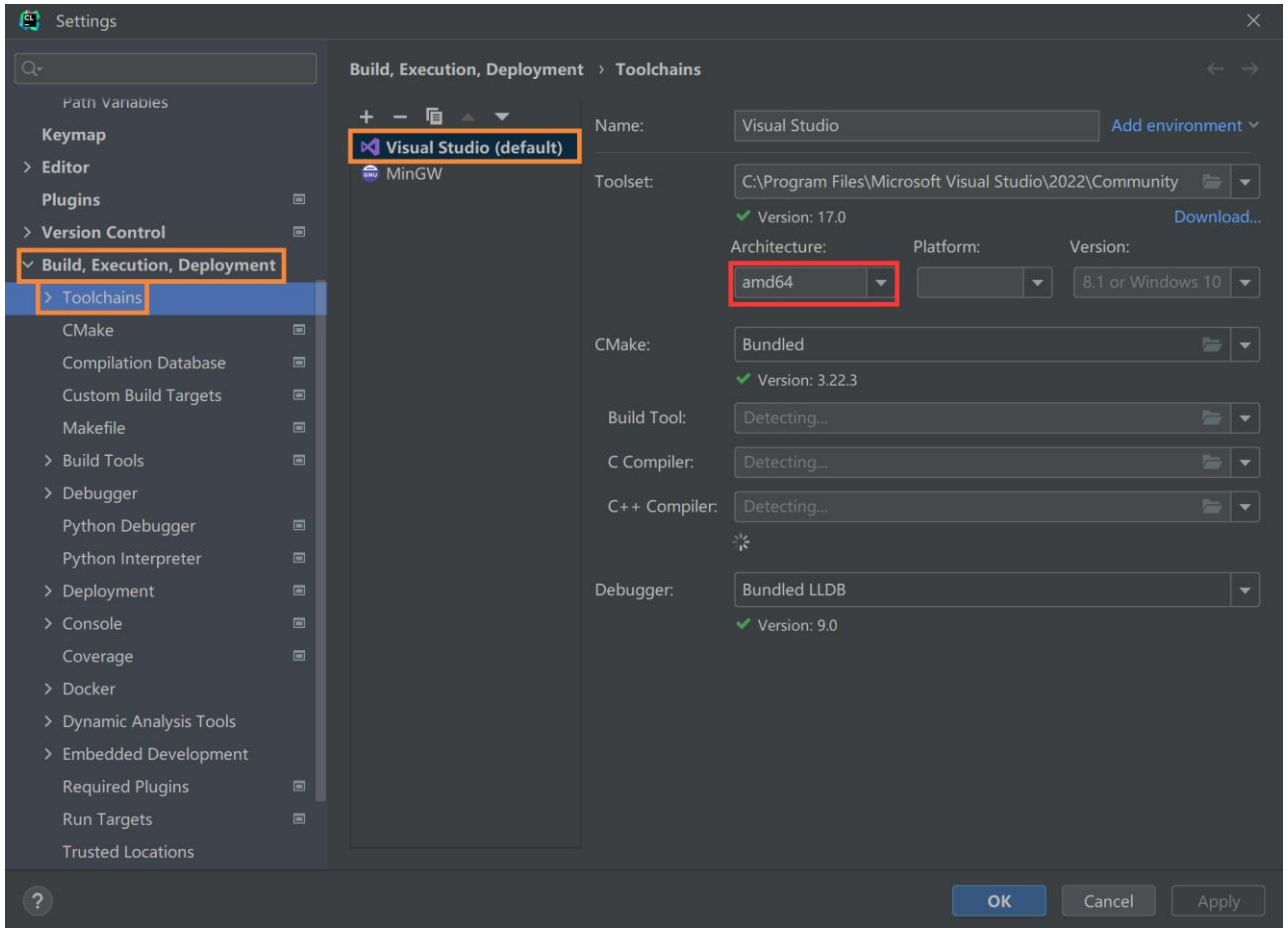
CMake on Windows

- Step1: git clone
- Step2: clion
  - Click “Open File or Project”
  - Click “File”, “Settings”



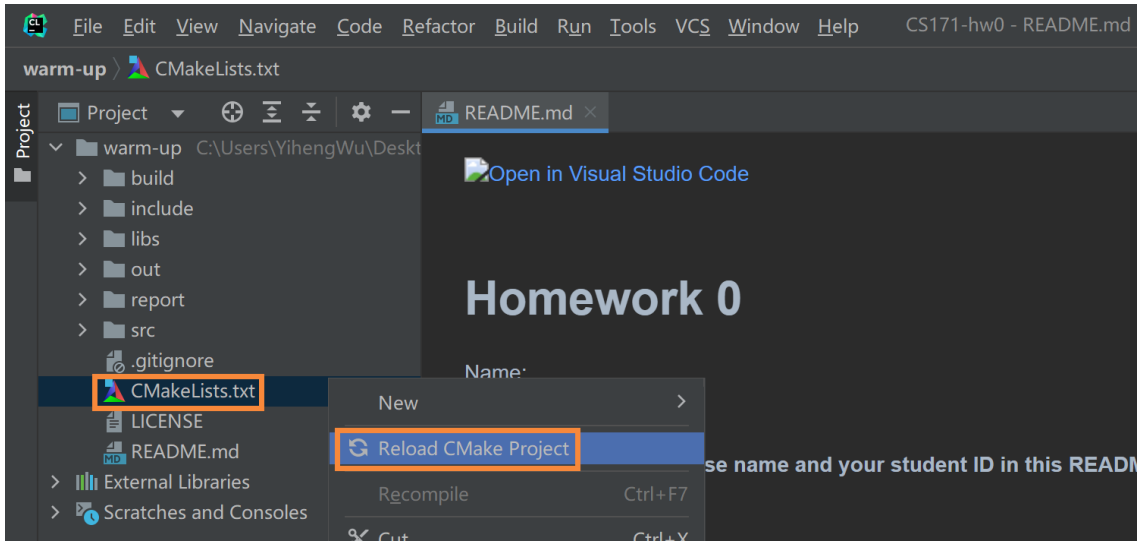
CMake on Windows

- Step1: git clone
- Step2: clion
  - Click “Open File or Project”
  - Click “File”, “Settings”
  - Set “VS amd64” as the compiler



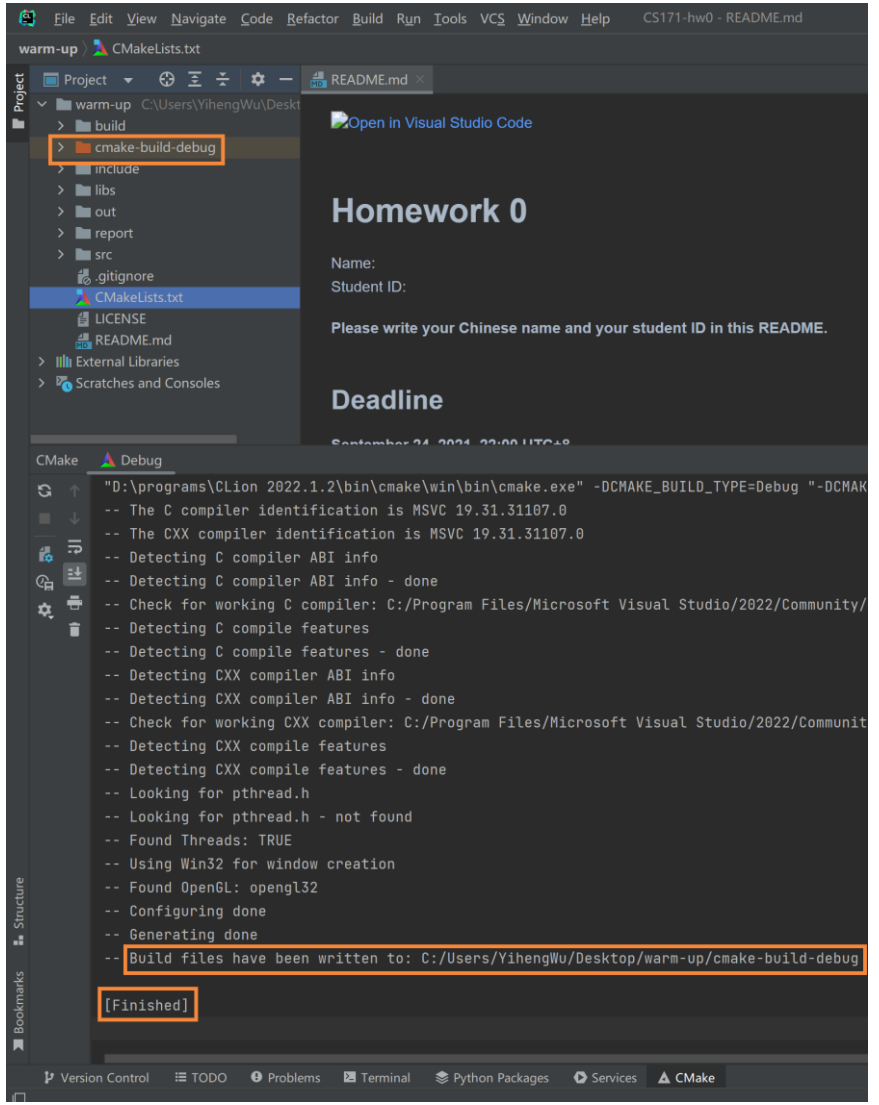
CMake on Windows

- Step1: git clone
- Step2: clion
  - Click “Open File or Project”
  - Click “File”, “Settings”
  - Set “VS amd64” as the compiler
  - Reload CMake project



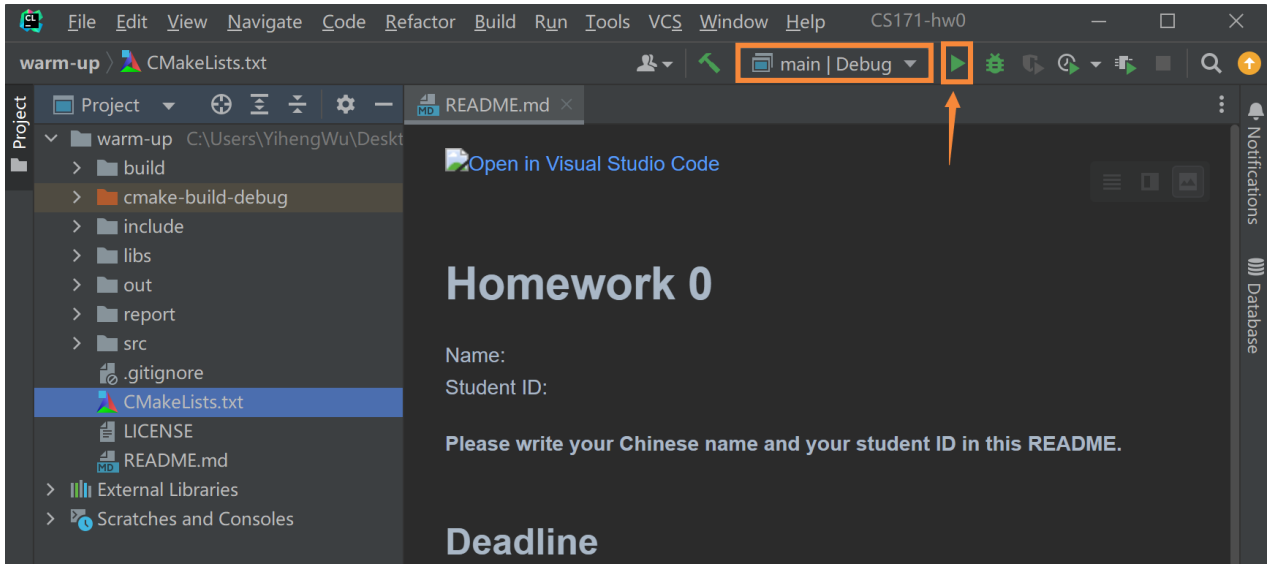
CMake on Windows

- Step1: git clone
- Step2: clion
  - Click “Open File or Project”
  - Click “File”, “Settings”
  - Set “VS amd64” as the compiler
  - Reload CMake project
  - Check success



CMake on Windows

- Step1: git clone
- Step2: clion
  - Click “Open File or Project”
  - Click “File”, “Settings”
  - Set “VS amd64” as the compiler
  - Reload CMake project
  - Check success
  - Click and run



CMake on Linux

- Step1: git clone
- Step2: g++
  - mkdir build
  - cd build
  - cmake ..
  - make –j12
    - Set to “-j12” if your CPU has 12 threads

CMake on Linux

- Step1: git clone
- Step2: g++
  - mkdir build
  - cd build
  - cmake ..
  - make –j12
    - Set to “-j12” if your CPU has 12 threads
  - ./hw0

CMake on Linux

- Step1: git clone
- Step2: g++
  - mkdir build
  - cd build
  - cmake ..
  - make –j12
    - Set to “-j12” if your CPU has 12 threads
  - ./hw0
- Step3: write a script to automatize

```
1   #!/bin/bash
2
3   cd build
4   cmake ..
5   make -j12
6   ./[illegible]
```

Outline

- CMake
- OpenGL, glfw, glad, and glm
- How to create a window?
- How to draw a triangle?
- How to draw a rectangle?
- How to draw a colorful rectangle?
- How to get time and input?
- How to finish HW1?

OpenGL, glfw, glad, and glm

- OpenGL
  - A specification
  - The OpenGL specification specifies exactly what the result/output of each function should be and how it should perform.
- glfw
  - Create and manipulate windows
- glad
  - Get the function pointer
- glm
  - A math library can be used on both CPU and GPU

Outline

- CMake
- OpenGL, glfw, glad, and glm
- How to create a window?
- How to draw a triangle?
- How to draw a rectangle?
- How to draw a colorful rectangle?
- How to get time and input?
- How to finish HW1?

How to create a window?

- Initialize glfw and glad

```
// initialize glfw and glad
bool init();

int main() {
  if (!init()) {      // if (failed to initialize glfw and glad):
    glfwTerminate(); //   terminate
    return -1;       //   return from main()
  }
```

How to create a window?

- Initialize glfw and glad
- The “render loop”

```
// the so-called "render loop"
while (!glfwWindowShouldClose(window)) {
  glClearColor(0.1f, 0.2f, 0.3f, 1.0f); // set the "background" color
  glClear(GL_COLOR_BUFFER_BIT);         // clear the window with this color

  if (glfwGetKey(window, GLFW_KEY_ESCAPE) == GLFW_PRESS) // if ("ESC" is pressed):
    glfwSetWindowShouldClose(window, true);              //   will close the window

  // TODO: add your render code here

  glfwPollEvents();
  glfwSwapBuffers(window); // we will always use double buffers
}
```

How to create a window?

- Initialize glfw and glad
- The “render loop”
- When “ESC” is pressed

```
// the so-called "render loop"
while (!glfwWindowShouldClose(window)) {...}

// terminate
glfwTerminate();
return 0;
```

How to create a window?

- Initialize glfw and glad
- The “render loop”
- When “ESC” is pressed
- Summary

```
// initialize glfw and glad
bool init();

int main() {
  // initialize glfw and glad
  if (!init()) {...}

  // the so-called "render loop"
  while (!glfwWindowShouldClose(window)) {...}

  // terminate
  glfwTerminate();
  return 0;
}

bool init() {...}
```

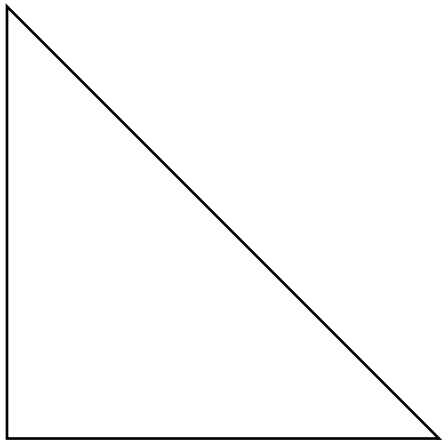
Outline

- CMake
- OpenGL, glfw, glad, and glm
- How to create a window?
- How to draw a triangle?
- How to draw a rectangle?
- How to draw a colorful rectangle?
- How to get time and input?
- How to finish HW1?

How to draw a triangle?

- Vertices

```
const float vertices[] = {
  0, 0, 0,
  1, 0, 0,
  0, 1, 0,
};
```

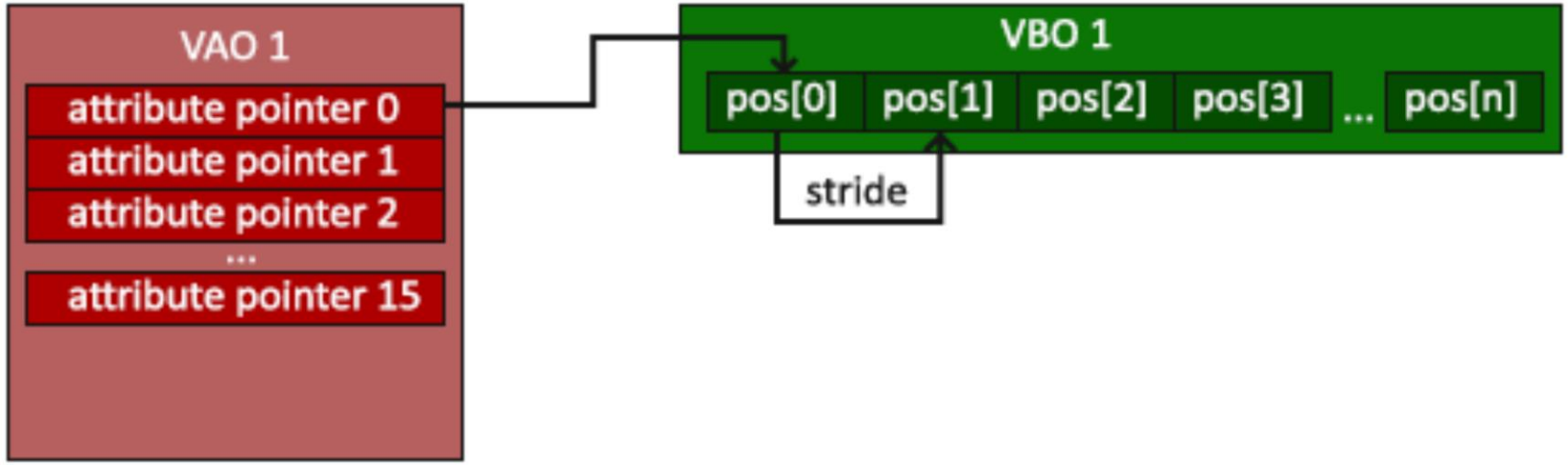


How to draw a triangle?

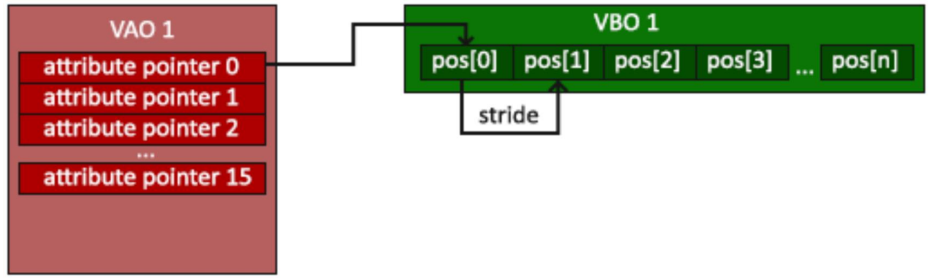
- Vertices
- Copy to a GPU buffer

How to draw a triangle?

- Vertices
- Copy to a GPU buffer
  - Vertex buffer object (vbo) and vertex array object (vao)



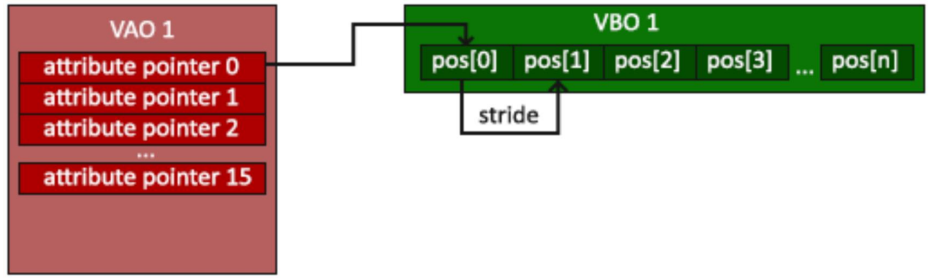
How to draw a triangle?



- Vertices
- Copy to a GPU buffer
  - Vertex buffer object (vbo) and vertex array object (vao)

```
GLuint vao;
glGenVertexArrays(1, &vao); // generate a vertex array object (vao)
glBindVertexArray(vao);     // bind to the state machine

GLuint vbo;
glGenBuffers(1, &vbo);                  // generate a vertex buffer object (vbo)
glBindBuffer(GL_ARRAY_BUFFER, vbo); // bind to the state machine
                                        // actually, bind to the above vao
```

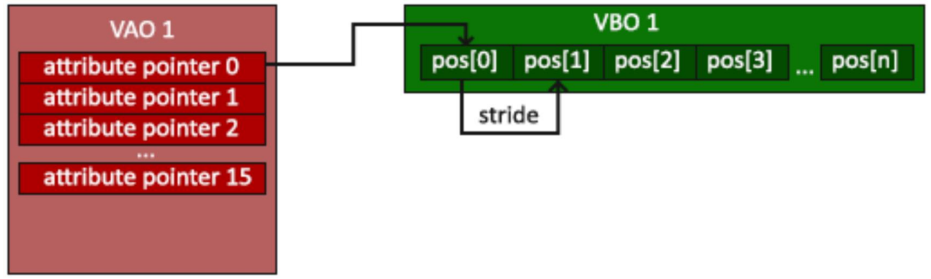
How to draw a triangle?



- Vertices
- Copy to a GPU buffer
  - Vertex buffer object (vbo) and vertex array object (vao)
  - Copy to vbo

```
GLuint vbo;
glGenBuffers(1, &vbo);                  // generate a vertex buffer object (vbo)
glBindBuffer(GL_ARRAY_BUFFER, vbo); // bind to the state machine
                                        // actually, bind to the above vao

glBufferData(GL_ARRAY_BUFFER, // upload to whom? our vbo!
             9 * sizeof(float), vertices,
             GL_STATIC_DRAW); // data never change
```

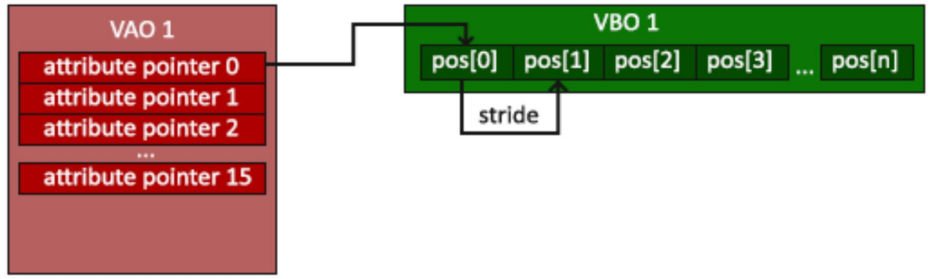
How to draw a triangle?



- Vertices
- Copy to a GPU buffer
  - Vertex buffer object (vbo) and vertex array object (vao)
  - Copy to vbo
  - Set attribute pointer

```
glBufferData(GL_ARRAY_BUFFER, // upload to whom? our vbo!
             9 * sizeof(float), vertices,
             GL_STATIC_DRAW); // data never change


glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), 0);
glEnableVertexAttribArray(0);
```

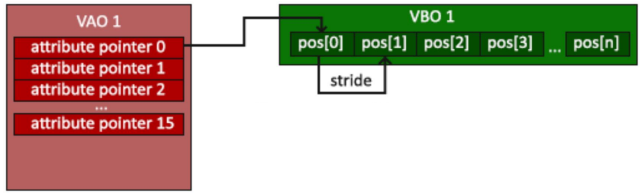
How to draw a triangle?



- Vertices
- Copy to a GPU buffer
  - Vertex buffer object (vbo) and vertex array object (vao)
  - Copy to vbo
  - Set attribute pointer
  - Unbind the vao

```
glBindVertexArray(0); // unbind the vao from the state machine
// you can create and bind another vao and another vbo, ...
```

How to draw a triangle?

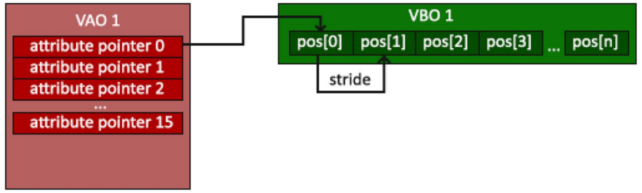


- Vertices
- Copy to a GPU buffer
  - Vao and vbo
  - Copy to vbo
  - Set attribute pointer
  - Unbind the vao
  - Summary

```
// initialize glfw and glad
if (!init()) {...}

GLuint vao;
glGenVertexArrays(1, &vao); // generate a vertex array object (vao)
glBindVertexArray(vao);     // bind to the state machine

GLuint vbo;
glGenBuffers(1, &vbo);              // generate a vertex buffer object (vbo)
glBindBuffer(GL_ARRAY_BUFFER, vbo); // bind to the state machine
                                    // actually, bind to the above vao

glBufferData(GL_ARRAY_BUFFER, // upload to whom? our vbo!
             9 * sizeof(float), vertices,
             GL_STATIC_DRAW); // data never change

glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), 0);
glEnableVertexAttribArray(0);

glBindVertexArray(0); // unbind the vao from the state machine
// you can create and bind another vao and another vbo, ...

// the so-called "render loop"
while (!glfwWindowShouldClose(window)) {...}
```

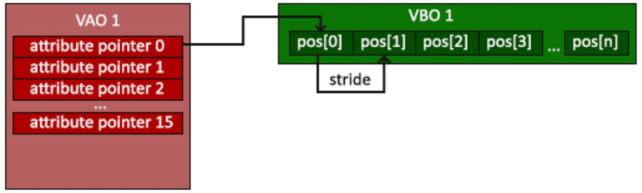
How to draw a triangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader



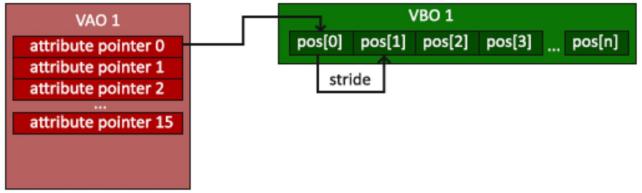
How to draw a triangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
  - Tell the GPU how to draw
  - compiled at runtime



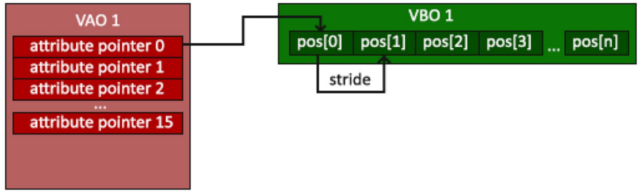
How to draw a triangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
  - Tell the GPU how to draw
  - compiled at runtime
  - Input: position of the 3 vertices of the triangle
  - Output: color of each pixel (fragment) covered by the triangle



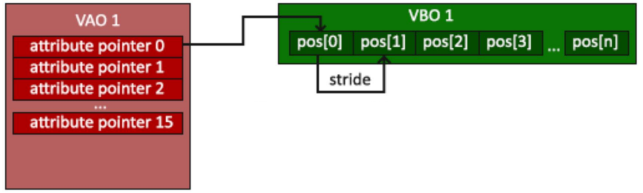
How to draw a triangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
  - Tell the GPU how to draw
  - compiled at runtime
  - Input: vertex data
  - Output: fragment color



How to draw a triangle?

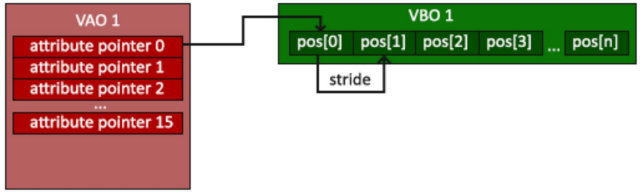
- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
  - Tell the GPU how to draw
  - compiled at runtime
  - Input: vertex data
  - Output: fragment color
  - Vertex shader




```
const char* vertex_shader_source_code =
  "#version 330 core                          \n"
  "layout (location = 0) in vec3 in_pos;\n"
  "void main() {                              \n"
  "  gl_Position = vec4(in_pos.x,             \n"
  "                     in_pos.y,             \n"
  "                     in_pos.z,             \n"
  "                     1.0);                 \n"
  "}                                          \n";
```

How to draw a triangle?

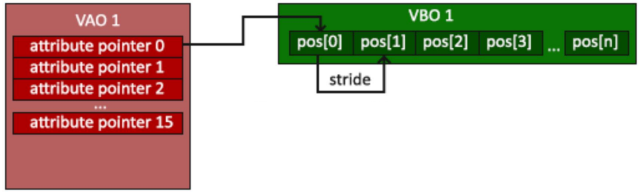
- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
  - Tell the GPU how to draw
  - compiled at runtime
  - Input: vertex data
  - Output: fragment color
  - Vertex shader
  - Fragment shader




```
const char* fragment_shader_source_code =
  "#version 330 core                          \n"
  "out vec4 frag_color;                       \n"
  "void main() {                              \n"
  "  frag_color = vec4(1.0f, 0.5f, 0.2f,\n"
  "                    1.0f);                 \n"
  "}                                          \n";
```

How to draw a triangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
  - Tell the GPU how to draw
  - compiled at runtime
  - Input: vertex data
  - Output: fragment color
  - Vertex shader
  - Fragment shader
  - Compile them




```
// compile the vertex and fragment shader
GLuint compileShaders(const char* vertex_shader_source,
                      const char* fragment_shader_source);
```

```
// compile the vertex and fragment shader
GLuint shader_program = compileShaders(
  vertex_shader_source_code,
  fragment_shader_source_code);
```

How to draw a triangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
  - Tell the GPU how to draw
  - compiled at runtime
  - Input: vertex data
  - Output: fragment color
  - Vertex shader
  - Fragment shader
  - Compile them
  - Summary

```
const char* vertex_shader_source_code =
source code of the vertex shader

const char* fragment_shader_source_code =
source code of the fragment shader

// compile the vertex and fragment shader
GLuint compileShaders(const char* vertex_shader_source,
                      const char* fragment_shader_source) {...}

int main() {
  // initialize glfw and glad
  if (!init()) {...}

  // copy to a GPU buffer
  {...}

  // compile the vertex and fragment shader
  GLuint shader_program = compileShaders(
    vertex_shader_source_code,
    fragment_shader_source_code);

  // the so-called "render loop"
  while (!glfwWindowShouldClose(window)) {...}

  // terminate
  glfwTerminate();
  return 0;
}
```

How to draw a triangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
- Draw with shader and vao

```
// TODO: add your render code here
// our render code here
glUseProgram(shader_program);
glBindVertexArray(vao);
glDrawArrays(GL_TRIANGLES, 0, 3);
```

How to draw a triangle?

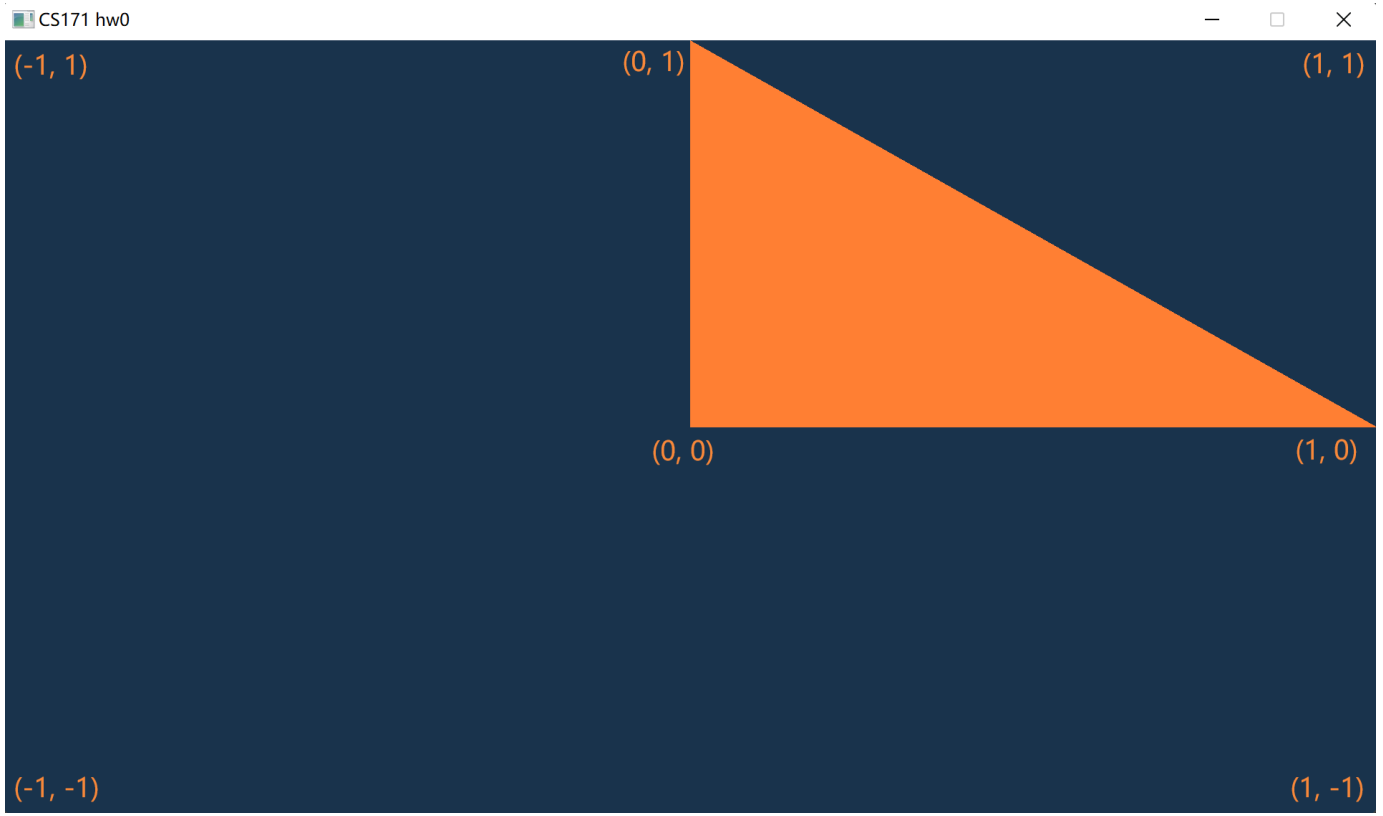
- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
- Draw with shader and vao
- Compile and run

How to draw a triangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
- Draw with shader and vao
- Compile and run
  - why at right-up corner?
  - why 不是等腰三角形?

How to draw a triangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
- Draw with shader and vao
- Compile and run
  - why at right-up corner?
  - why 不是等腰三角形?



Outline

- CMake
- OpenGL, glfw, glad, and glm
- How to create a window?
- How to draw a triangle?
- How to draw a rectangle?
- How to draw a colorful rectangle?
- How to get time and input?
- How to finish HW1?

How to draw a rectangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
- Draw with shader and vao
- Compile and run

How to draw a rectangle?

- Vertices

```
const float vertices[] = {
  0, 0, 0,
  1, 0, 0,
  0, 1, 0,

  1, 0, 0,
  0, 1, 0,
  1, 1, 0,
};
```

How to draw a rectangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)

```
glBufferData(GL_ARRAY_BUFFER, // upload to whom? our vbo!
             18 * sizeof(float), vertices,
             GL_STATIC_DRAW); // data never change
```

How to draw a rectangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
- Draw with shader and vao

```
// TODO: add your render code here
// our render code here
glUseProgram(shader_program);
glBindVertexArray(vao);
glDrawArrays(GL_TRIANGLES, 0, 6);
```

How to draw a rectangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
- Draw with shader and vao
- Compile and run

Outline

- CMake
- OpenGL, glfw, glad, and glm
- How to create a window?
- How to draw a triangle?
- How to draw a rectangle?
- How to draw a colorful rectangle?
- How to get time and input?
- How to finish HW1?

How to draw a colorful rectangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
- Draw with shader and vao
- Compile and run

How to draw a colorful rectangle?

- Vertices
  - add color to each vertex

```
const float vertices[] = {
/*
  position      color
 */
  0, 0, 0,      1, 0, 0, // red
  1, 0, 0,      0, 1, 0, // green
  0, 1, 0,      0, 0, 1, // blue

  1, 0, 0,      0, 1, 0, // green
  0, 1, 0,      0, 0, 1, // blue
  1, 1, 0,      1, 0, 0, // red
};
```

How to draw a colorful rectangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
  - copy more bytes

```
glBufferData(GL_ARRAY_BUFFER, // upload to whom? our vbo!
             36 * sizeof(float), vertices,
             GL_STATIC_DRAW); // data never change
```

How to draw a colorful rectangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
  - copy more bytes
  - set another attribute pointer for colors

```
glVertexAttribPointer(0,                    // POSITION is the 0th attribute
                      3, GL_FLOAT,          // has 3 floats
                      GL_FALSE,
                      6 * sizeof(float),    // step 6 floats to reach next
                      (void*)0);            // offset is 0
glEnableVertexAttribArray(0);               // activate POSITION

glVertexAttribPointer(1,                    // COLOR is the 1st attribute
                      3, GL_FLOAT,          // has 3 floats
                      GL_FALSE,
                      6 * sizeof(float),    // step 6 floats to reach next
                      (void*)(3 * sizeof(float)));  // offset is 3 floats
glEnableVertexAttribArray(1);               // activate COLOR
```

How to draw a colorful rectangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
  - Vertex shader

```
const char* vertex_shader_source_code =
  "#version 330 core                          \n"
  "layout (location = 0) in vec3 in_pos;      \n"
  "layout (location = 1) in vec3 in_color;    \n"
  "out vec3 color;                            \n"
  "void main() {                              \n"
  "  gl_Position = vec4(in_pos.x,             \n"
  "                     in_pos.y,             \n"
  "                     in_pos.z,             \n"
  "                     1.0);                 \n"
  "  color = in_color;                        \n"
  "}                                          \n";
```

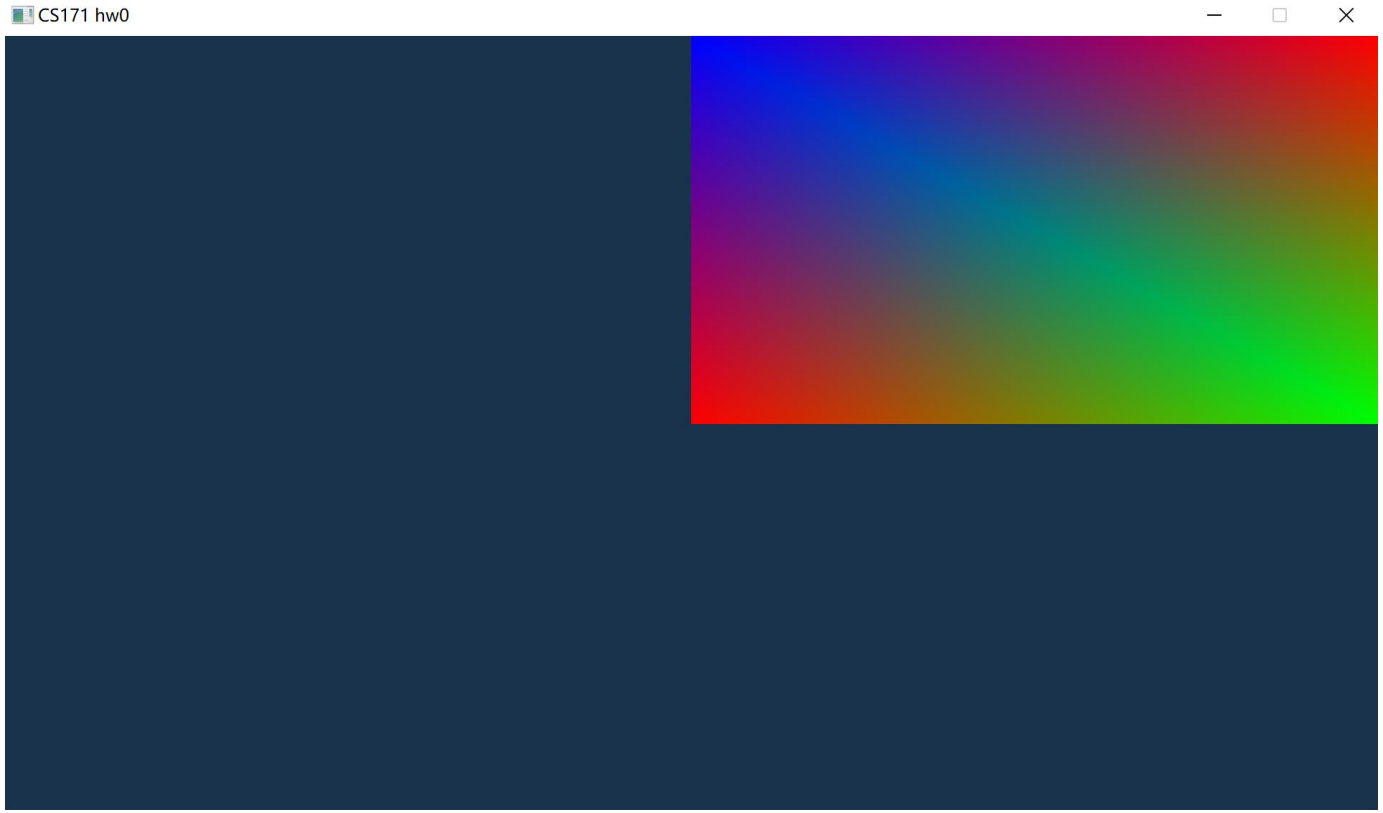
How to draw a colorful rectangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
  - Vertex shader
  - Fragment shader

```
const char* fragment_shader_source_code =
  "#version 330 core                    \n"
  "in vec3 color;                       \n"
  "out vec4 frag_color;                 \n"
  "void main() {                        \n"
  "  frag_color = vec4(color,           \n"
  "                    1.0f);           \n"
  "}                                    \n";
```

How to draw a colorful rectangle?

- Vertices
- Copy to a GPU buffer (vao and vbo)
- Shader
- Draw with shader and vao
- Compile and run



Outline

- CMake
- OpenGL, glfw, glad, and glm
- How to create a window?
- How to draw a triangle?
- How to draw a rectangle?
- How to draw a colorful rectangle?
- How to get time and input?
- How to finish HW1?

How to get time and input?

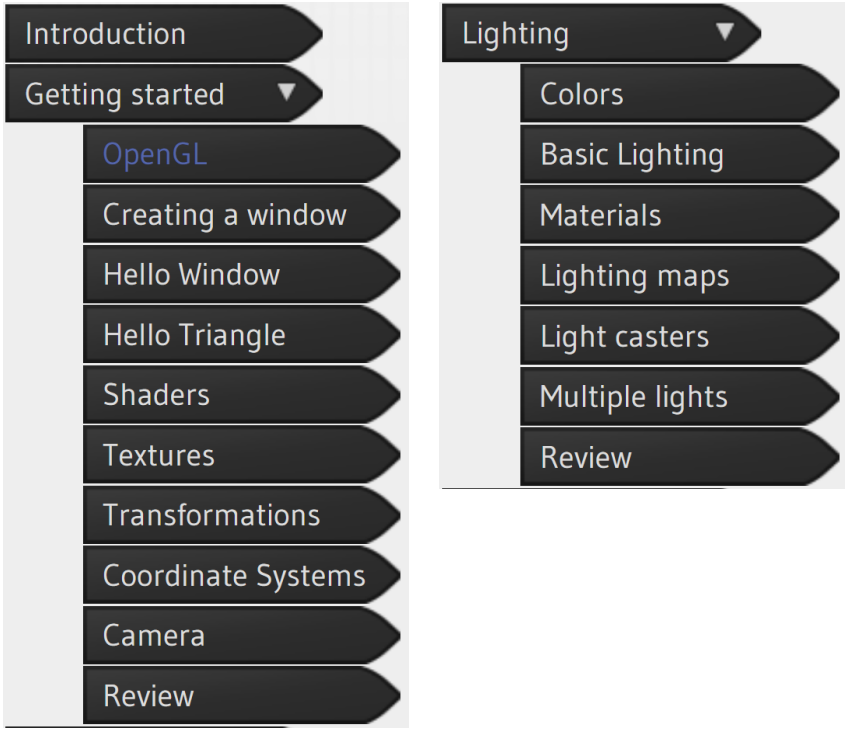
- Time
  - glfwGetTime()
- Input
  - glfwGetMouseButton()
  - glfwGetKey()
  - glfwGetCursorPos()

Outline

- CMake
- OpenGL, glfw, glad, and glm
- How to create a window?
- How to draw a triangle?
- How to draw a rectangle?
- How to draw a colorful rectangle?
- How to get time and input?
- How to finish HW1?

How to finish HW1?

- It is impossible to cover every details in tutorials…
- Learn more by yourself
  - LearnOpenGL - OpenGL



How to finish HW1?

- It is impossible to cover every details in tutorials…
- Learn more by yourself
  - LearnOpenGL - OpenGL
- Write more codes

Thank you