Tutorial 1: Warm-up

TA: Hongtu Xu, Mengyun Liu

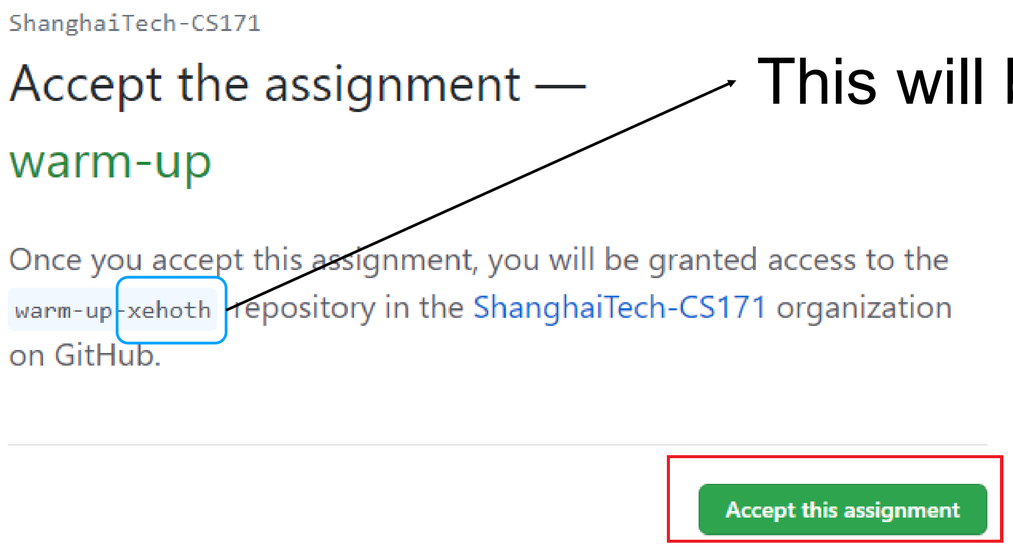
Agenda

- About Homework
- Project Setup
- OpenGL Coding

About Homework

How to get your homework?

- Course Page: https://faculty.sist.shanghaitech.edu.cn/faculty/liuxp/course/cs171.01/
- There's a GitHub Classroom link in the assignment page.
- Click the link & accept the assignment with your GitHub account.



This will be your own account name

How to get your homework?

You accepted the assignment, **warm-up** . We're configuring your repository now. This may take a few minutes to complete. Refresh this page to see updates.

Your assignment is due by **Sep 24, 2021, 22:00**

Note: You may receive an email invitation to join ShanghaiTech-CS171 on your behalf. No further action is necessary.

How to get your homework?

You're ready to go!

You accepted the assignment, **warm-up**.

This will be your own account name

Your assignment repository has been created:

https://github.com/ShanghaiTech-CS171/warm-up-xehoth

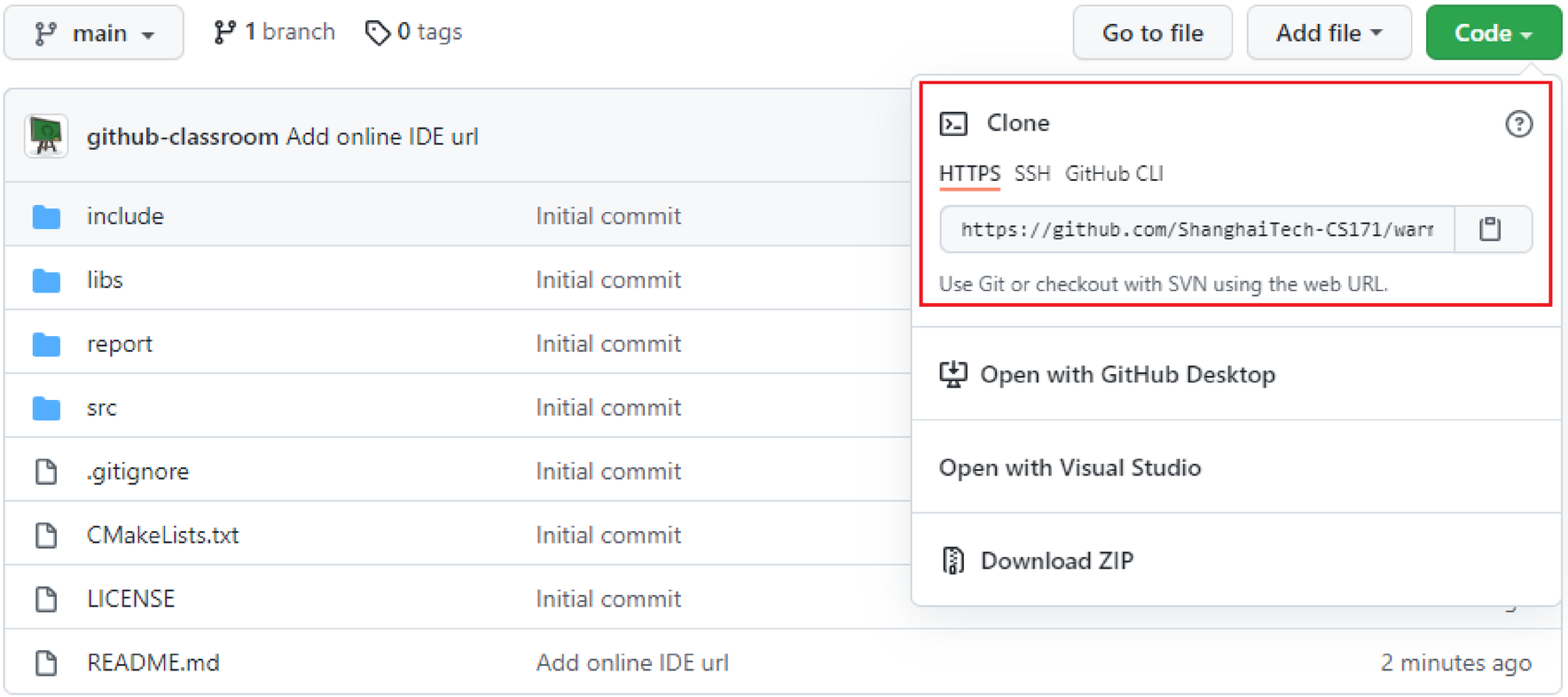
We've configured the repository associated with this assignment (update).

Open in Visual Studio Code

Your assignment is due by **Sep 24, 2021, 22:00**

Note: You may receive an email invitation to join ShanghaiTech-CS171 on your behalf. No further action is necessary.

How to get your homework?



Project Structure

```
|
|–include
|–libs
|–report
|–src
|–.gitignore
|–CMakeLists.txt
|–README.md
```

- include 目录
- 第三方库目录
- 存放你的 report
- 源代码
- 在 git 提交时忽略一些文件(夹)，如 build 等。
  请确保你提交文件的大小小于 30MB
- CMake 配置文件

Assignment Requirement

- You are supposed to draw some objects in the OpenGL window created for you.
- Since this is only a warm-up to help you get familiar with OpenGL, we have no constraints what objects do you need to draw.
- But... We will not expect you to draw a single triangle which will be introduced in this tutorial :D
- We encourage you to explore OpenGL by drawing some interesting scenes.

Grading Rule

- Take it easy! We will not grade your submissions!
- Please just have fun with this assignment!
- DDL: Sep 24, 2021, 22:00

Learning Materials

- Web
  - https://learnopengl-cn.github.io (Strongly Recommended)
  - https:// www.opengl-tutorial.org
  - https://www.khronos.org/opengl/wiki/Tutorials
- Book
  - OpenGL Programming Guide: The Official Guide to Learning OpenGL,
  - Real-Time Rendering, 3rd Edition

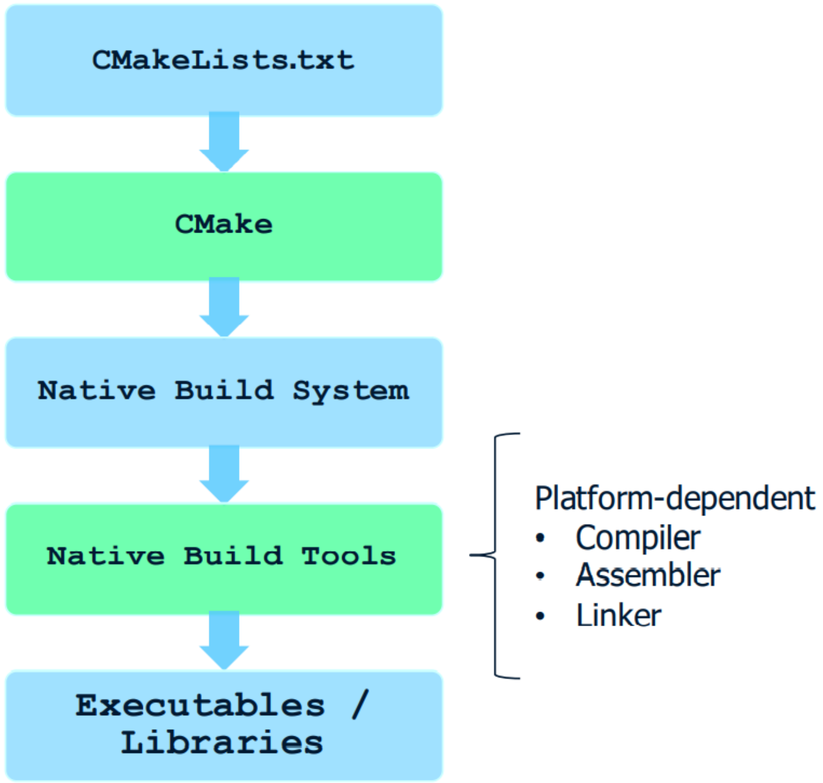
Project Setup

CMake

- Generates native build environments
- Supports multiple platforms
  - Unix/Linux -> makefile
  - Windows -> Some IDE projects (e.g., Visual Studio Solutions)
  - macOS -> makefile/Xcode
- Open-Source
- Cross-Platform
- Manage complex, large build environments
- Integrated Testing & Packaging (Ctest, CPack)

CMake

Build-system generator

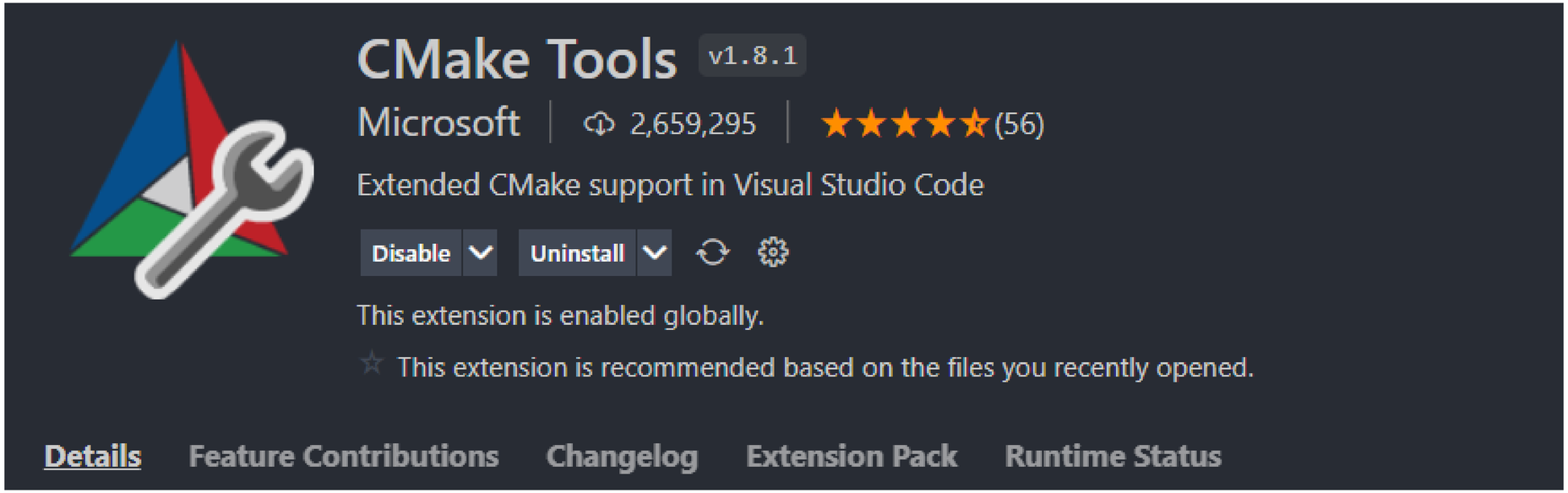


Project Setup (Command Line)

- 创建一个子文件夹用于生成
- 调用 cmake 生成工程并构建

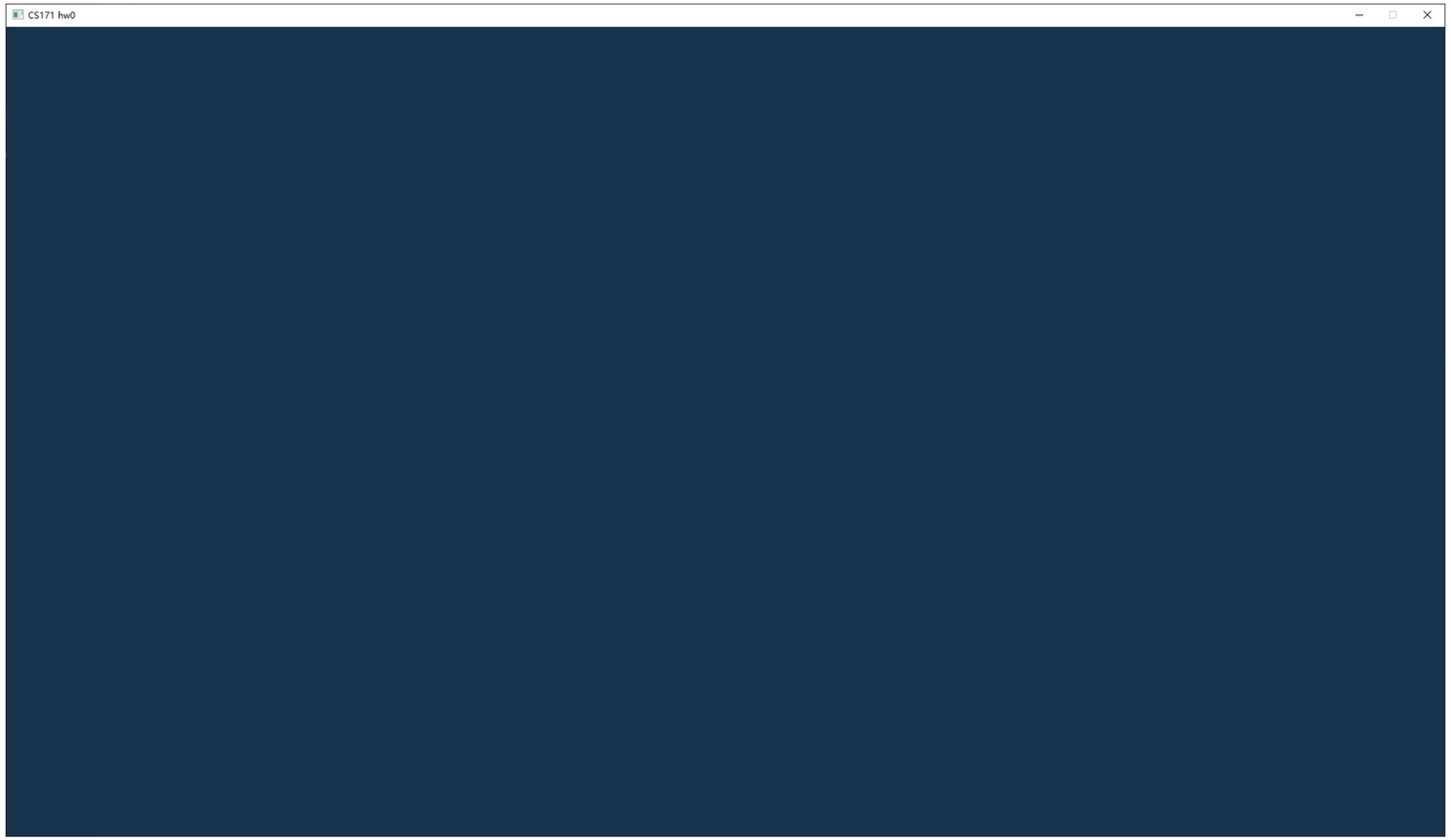
```
mkdir build
cd build
cmake ..
cmake --build .
```

Project Setup (Visual Studio Code)



Project Setup (Visual Studio)

Project Setup (Expected Running Result)



Modern CMake Basics

- Modern CMake: Target and Property
- 把 Target 想象成一个对象 (Object):
- 构造函数:
  - add_executable(<name> [source] …)
  - add_library(<name> [source]…)
- 成员函数:
  - get_target_property
  - set_target_property
  - target_compile_definitions
  - target_compile_features
  - target_compile_options
  - target_link_libraries
  - target_include_directories

CMake Example

```
cmake_minimum_required(VERSION 3.16)
project(CS171-hw0 DESCRIPTION "warm up homework" LANGUAGES C CXX)

# add some libs
# ......

list(APPEND INC_DIRS include)

# set source files and add executable
file(GLOB SRC_FILES src/*.cpp)
add_executable(main ${SRC_FILES})
target_compile_features(main PRIVATE cxx_std_17)
target_include_directories(main PRIVATE ${INC_DIRS})
# link libraries
target_link_libraries(main PRIVATE ${LIBRARIES})
```

OpenGL Setup (Window & Extension Wrapper)

- Window (窗口管理库)
  - 古老产品: glut/freeglut
  - 替代品: glfw
- OpenGL Extension Wrapper (函数加载)
  - 古老产品: glew
  - 替代品: glad
- 常见环境配置

a. glfw + glew (考虑到可能存在的立即模式需求，我们提供此模板配置)

b. glfw + glad (推荐配置，针对现代 OpenGL)

c. freeglut + glew

OpenGL Coding

Specifying 2D primitives in OpenGL

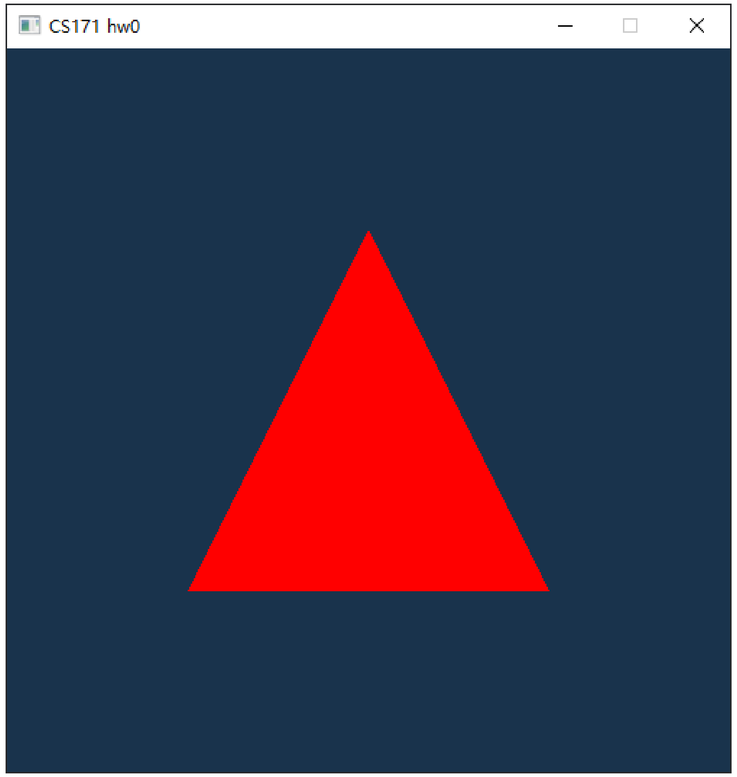
- **Specify 2D triangles**

```
struct Position2D
{
    float x,y;
};

struct Triangle2D
{
   Position2D
        p1,p2,p3;
};

Triangle2D t1,t2,…;
```

```
glBegin(GL_TRIANGLES);

glVertex2f(t1.p1.x, t1.p1.y);
glVertex2f(t1.p2.x, t1.p2.y);
glVertex2f(t1.p3.x, t1.p3.y);

glVertex2f(t2.p1.x, t2.p1.y);
glVertex2f(t2.p2.x, t2.p2.y);
glVertex2f(t2.p3.x, t2.p3.y);

…

glEnd();
```

OpenGL Immediate Mode Example (Triangle)

```
glBegin(GL_TRIANGLES);
  // vertex color (red)
  glColor3f(1.0f, 0.0f, 0.0f);
  // positions
  glVertex2f(-0.5f, -0.5f);
  glVertex2f(0.5f, -0.5f);
  glVertex2f(0.0f, 0.5f);
glEnd();
```



Immediate mode may not be able to use in macOS

Note that this triangle is draw in 500x500 window, it may be different from the triangle you draw.

Modern OpenGL (Recommended)

OpenGL Mode: Immediate v.s. Core

- Immediate mode (立即模式):
  - 早期的 OpenGL 使用
  - 固定渲染管线
  - 容易使用和理解
  - 绘制图形很方便 (glBegin & glEnd)
  - 大多数功能都被库隐藏起来，不够灵活，效率低
- Core mode (核心模式):
  - 现代 OpenGL
  - 可编程渲染管线
  - 更多的灵活性，更高的效率
  - 更深入的理解图形编程
  - 虽然上手更困难，但这份努力是值得的

State Machine (状态机) v.s. Objects (对象)

- 当我们使用一个对象时，通常看起来像如下一样（把OpenGL上下文看作一个大的结构体）：

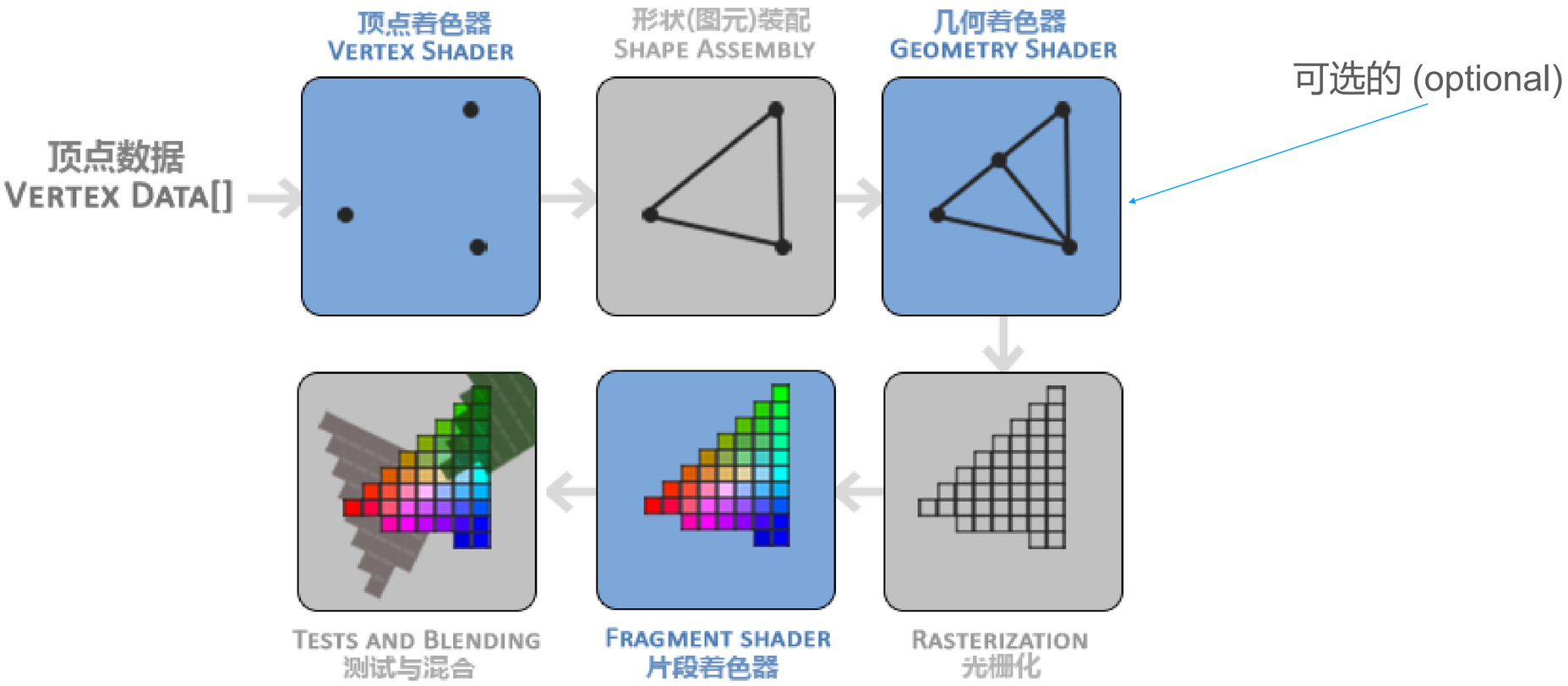
Object 只是一个统称，实际使用的时候可能是 VertexArray, Buffer, …

```
// 创建对象
unsigned int objectId = 0;
glGenObject(1, &objectId);
// 绑定对象至上下文
glBindObject(GL_WINDOW_TARGET, objectId);
// 设置当前绑定到 GL_WINDOW_TARGET 的对象的一些选项
glSetObjectOption(GL_WINDOW_TARGET, GL_OPTION_WINDOW_WIDTH, 800);
glSetObjectOption(GL_WINDOW_TARGET, GL_OPTION_WINDOW_HEIGHT, 600);
// 将上下文对象设回默认
glBindObject(GL_WINDOW_TARGET, 0);
```

```
// OpenGL的状态
struct OpenGL_Context {
  ...
  object* object_Window_Target;
  ...
};
```

More details: [Learn OpenGL]

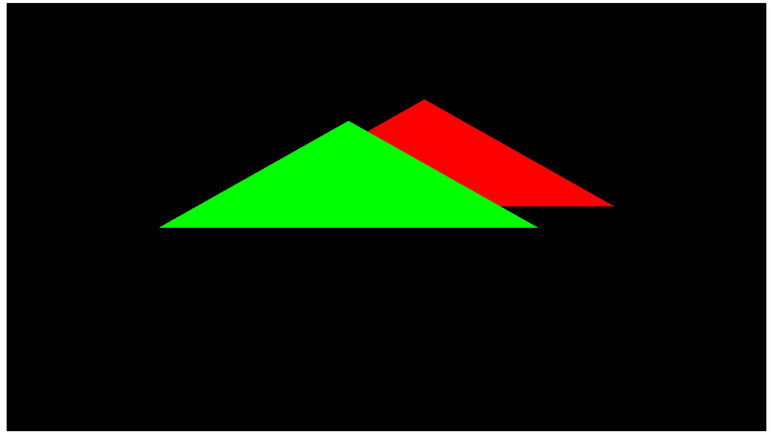
OpenGL Rendering Pipeline (渲染管线)



OpenGL NDC Coordinates

- NDC: Normalized Device Coordinates (标准化设备坐标)
- 在 OpenGL 中，标准化设备坐标的 (x, y, z) 值均在 [−1,1] 中
- NDC 坐标是**左手**坐标系

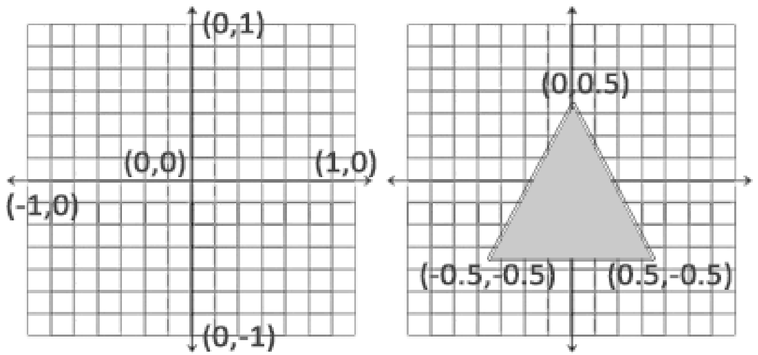
- 绿色三角形 z 为 -0.5，红色为 0.5
- 红的被绿的挡住了说明 z 越大离得越远



VBO (Vertex Buffer Object, 顶点缓冲对象)

- 用于在 GPU 中储存顶点数据
- 如 CPU 端顶点数据如下:

```
float vertices[] = {
 -0.5f, -0.5f, 0.0f,
  0.5f, -0.5f, 0.0f,
  0.0f,  0.5f, 0.0f
};
```




```
// VBO id
unsigned int VBO;
// 生成 VBO
glGenBuffers(1, &VBO);
// 绑定一个 VBO 对象 (why? 状态机)
glBindBuffer(GL_ARRAY_BUFFER, V
BO);
// 将 vertices 数据复制到 VBO
glBufferData(GL_ARRAY_BUFFER,
      sizeof(vertices),
      vertices,
      GL_STATIC_DRAW);
```

More details: [Learn OpenGL]

References

- Learn OpenGL: https://learnopengl.com/
- Modern CMake 简介: https://zhuanlan.zhihu.com/p/76975231
- Lecture 2: https://faculty.sist.shanghaitech.edu.cn/faculty/liuxp/course/cs171.01/pdf/lecture02.pdf