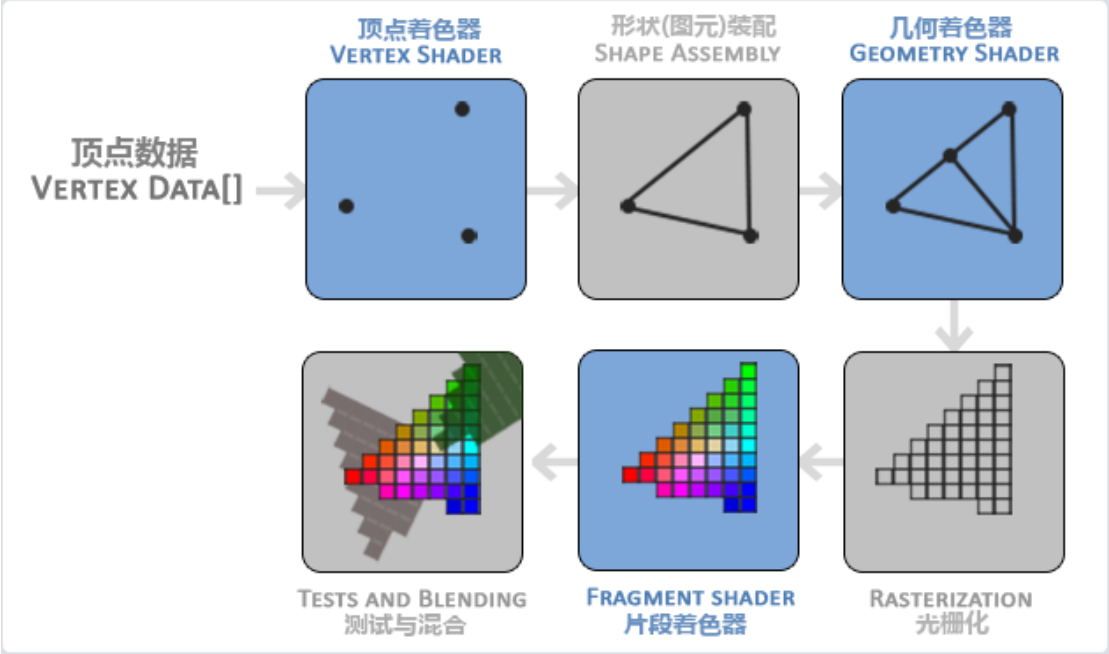
Homework 2 - GUI and Draw simple graphics

Basic:

1. 使用 OpenGL(3.3 及以上)+GLFW 或 freeglut 画一个简单的三角形。

在开始着手画三角形之前，需要做一些准备工作。

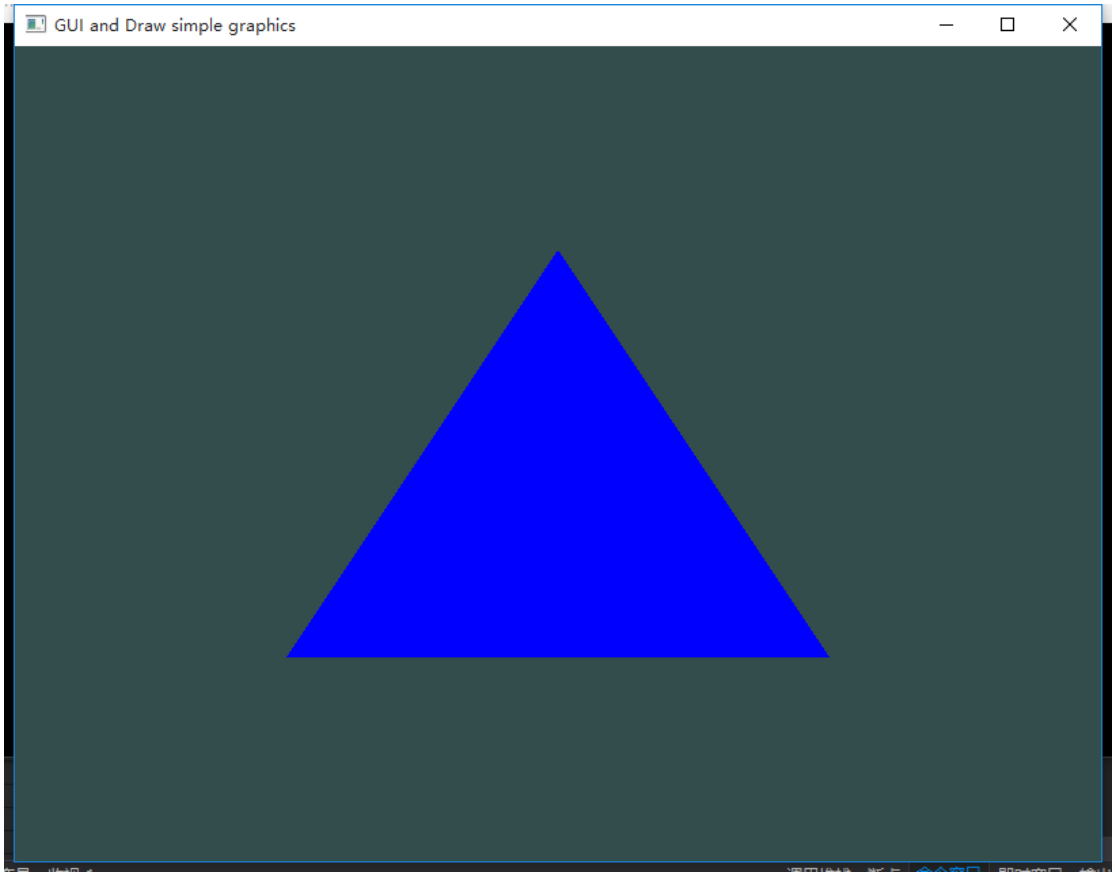
- 配置环境。在 VS2017 上配置好 GLAD 和 GLFW。
- 了解 OpenGL 的图形渲染管线（Graphics Pipeline）。图形渲染管线可以被划分为两个主要部分：第一部分把 3D 坐标转换为 2D 坐标，第二部分是把 2D 坐标转变为实际的有颜色的像素。图形渲染管线的每个阶段的抽象展示如下：



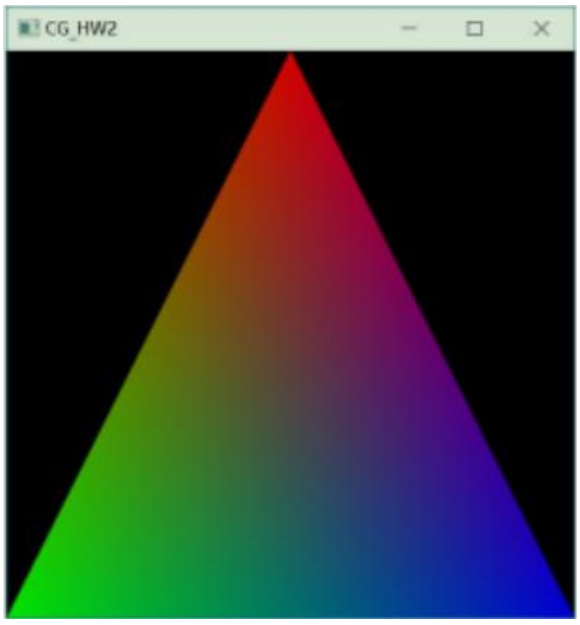
实现思路简述：

- 初始化及配置 GLFW，创建 GLFW 窗口，加载 GLAD。
- 使用 GLSL 编写顶点着色器和片段着色器，编译，链接为一个着色器程序对象。此部分代码在 Shader.h 文件中实现。
- 生成配置 VAO。
- 最后，通过 glDrawArrays 函数来绘制三角形。

最终绘制结果如下：



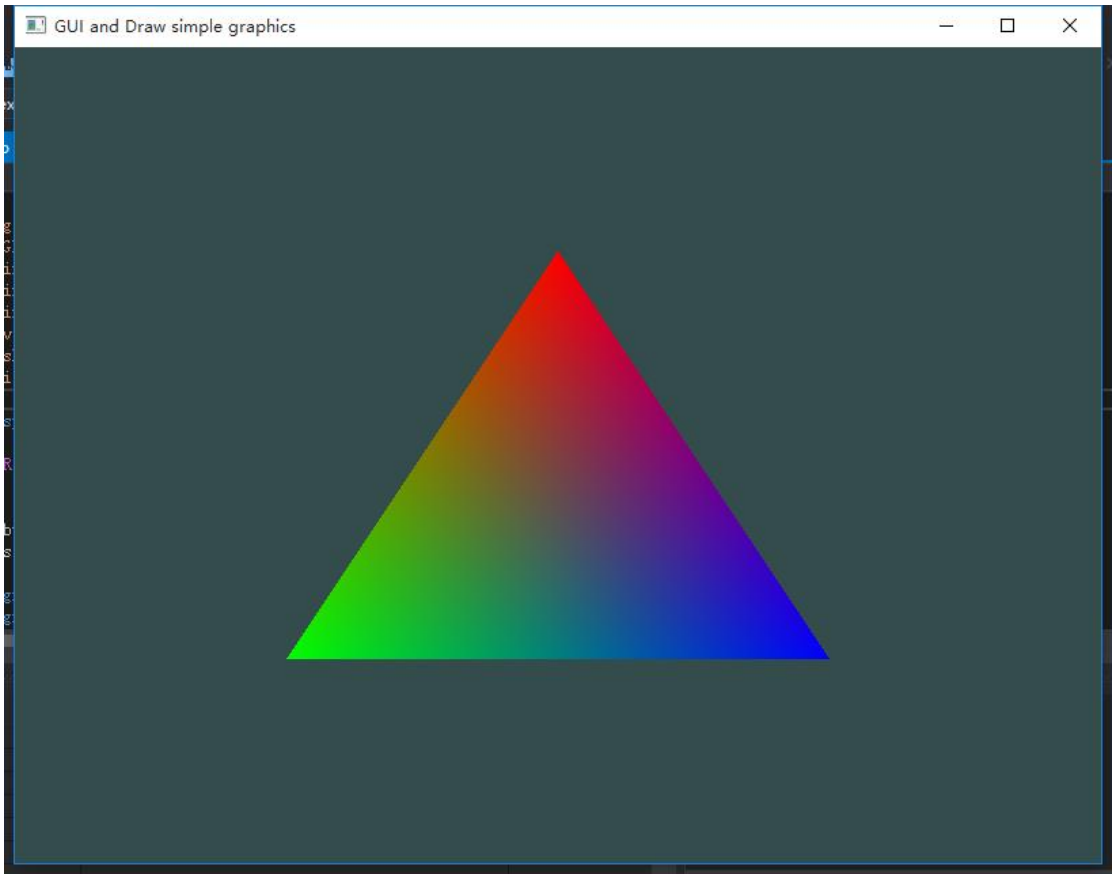
2. **对三角形的三个顶点分别改为红绿蓝，像下面这样。并解释为什么会出现这样的结果。**



在第 1 问的基础上，增加以下步骤：

- 在每个顶点添加颜色属性。
- 重新编写顶点着色器和颜色着色器，编译、链接成程序。
- 重新解释链接顶点属性，即更改 glVertexAttribPointer 函数。需要更改步长以及增加一个 glVertexAttribPointer 函数用于处理颜色属性，需要注意偏移量。

实现的结果如下：

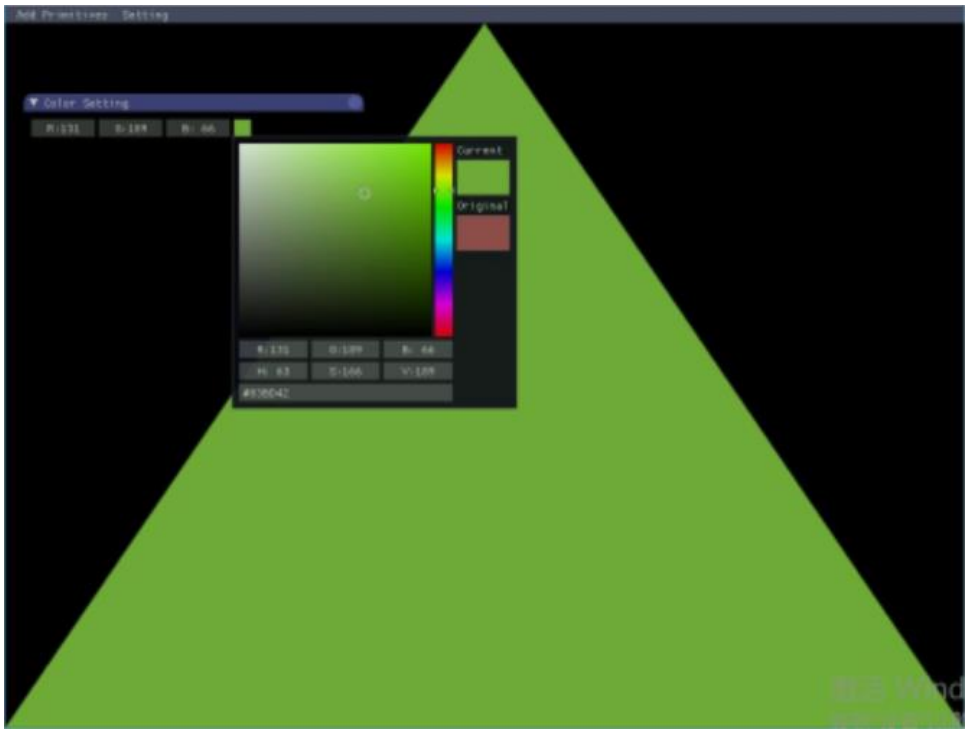


出现这样的结果的原因：

这是在片段着色器中进行片段插值(Fragment Interpolation)的结果。在渲染一个三角形时，光栅化(Rasterization)阶段通常会造成比原指定顶点更多的片段。光栅会根据每个片段在三角形形状上所处相对位置决定这些片段的位置。基于这些位置，它会插值(Interpolate)所有片段着色器的输入变量。

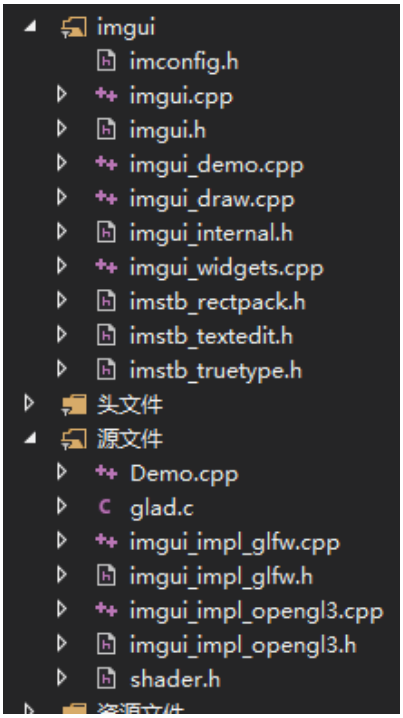
即是说，在渲染三角形的光栅化阶段，OpenGL 会根据提供的 3 个顶点来确定三角形的边和三角形内部的像素点，当确定了三角形所有像素点的位置后，由于只提供了三角形的 3 个顶点的属性，则对于其他像素点的属性，着色器会通过这 3 个顶点的属性，以位置属性为比例参考进行插值处理。

3. 给上述工作添加一个 GUI，里面有一个菜单栏，使得可以选择并改变三角形的颜色。



使用 ImGui 工具添加 GUI，实现流程:

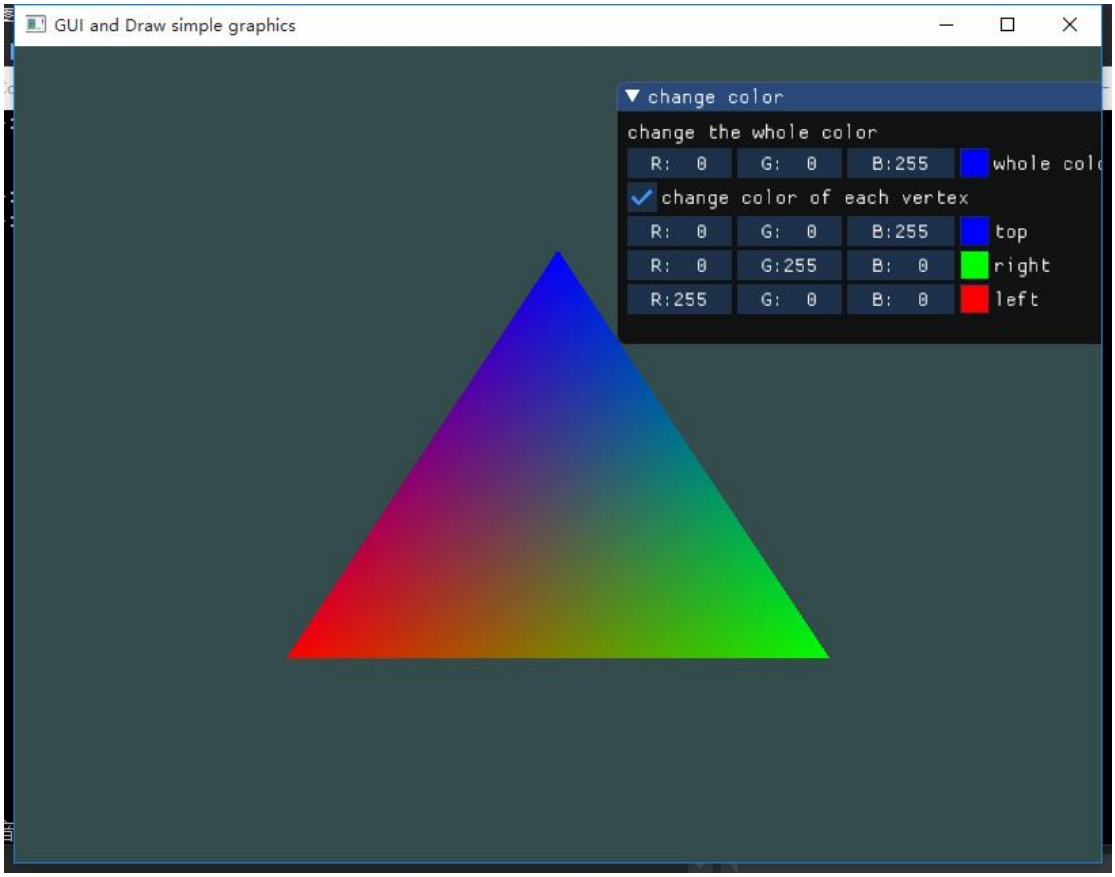
- 现在 ImGui 源代码，将使用 ImGui 需要的依赖文件增加到项目中，如下：



- 由于我使用的是 GLAD，将文件 imgui_impl_opengl3.h 中 OpenGL3 的默认 loader 改为 GLAD。
- 设置 ImGui 的 context，io 和窗口等。增加语句 glfwSwapInterval(1);使得缓存不断更新，达到更改视图的效果。

- 在 while(!glfwWindowShouldClose(window))循环中，设置新的 ImGui 的 Frame，使用 lable、text 等组件构建 GUI 菜单栏。同时将菜单栏中的变化映射到角点属性中。
- 为了使每一次刷新都能重新链接，着色，将 VAO 和 VBO 以及着色器着色等过程放入循环中，使得界面的每一帧都根据当前的缓存情况进行渲染。

实现结果如下：



演示结果见 task3.gif

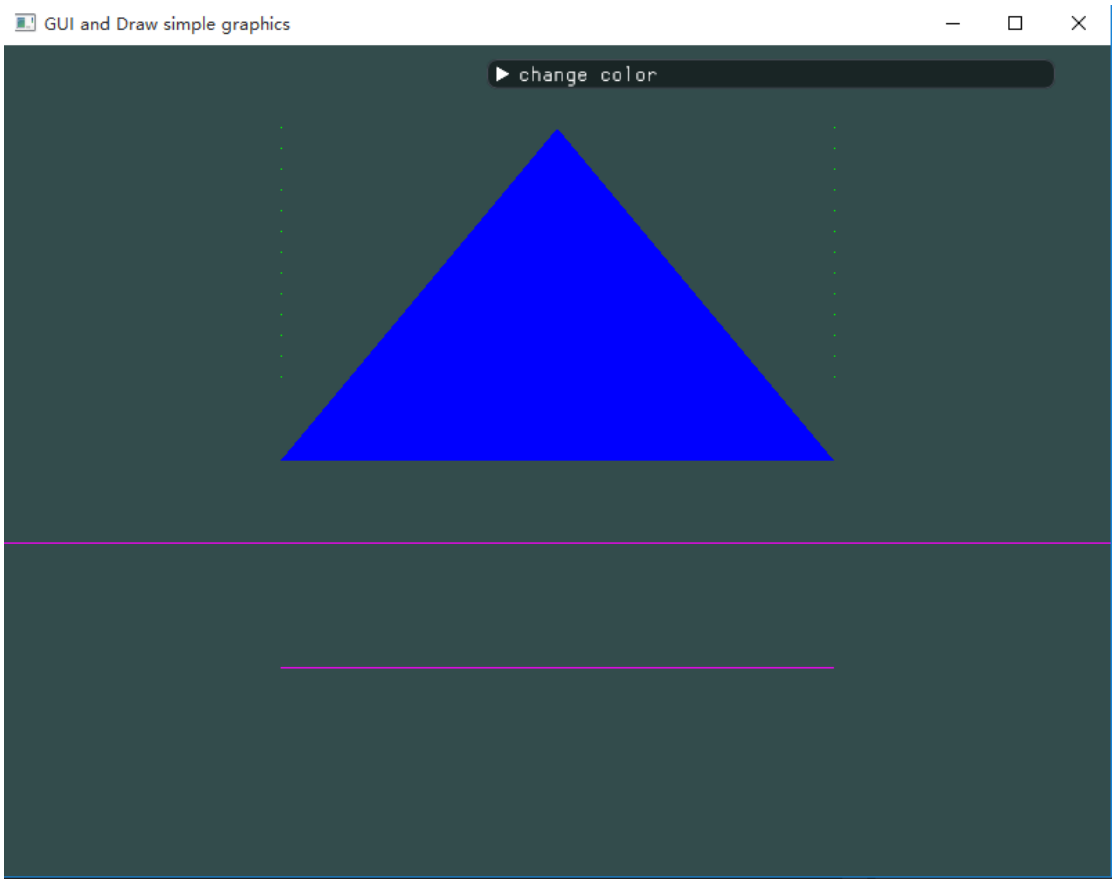
Bonus：

1. 绘制其他的图元，除了三角形，还有点、线等。

绘制点和线的流程和绘制三角形差不多。事实上，绘制点和线的大部分流程都在绘制三角形的时候已经完成了。现在只需在原有代码的基础上作出如下更改：

- 在原本定义三角形属性数组（vertices）中，增加线和点的属性。需要注意，一条线由两个点决定，我们在这里只需定义线的两个端点属性即可。
- 关于线和点的渲染过程，和三角形是一样的。在这里，我们可以直接使用前面画三角形的着色器。
- 最后，则是使用 glDrawArrays 函数画出点和线。glDrawArrays 有三个参数:第一个参数是指渲染类型，分别为 GL_TRIANGLES（三角形）、GL_LINES（线）、GL_POINTS（点）；第二个参数为顶点数组对象的开始坐标（从零开始），即在顶点属性数组（vertices）中，从哪一个元素开始是这次绘制的第一个顶点；第三个参数是顶点数目，即这次要绘制的在顶点属性数组的顶点个数。

最终绘制结果如下：



2. 使用 EBO(Element Buffer Object)绘制多个三角形。

在原本代码的基础上进行更改，步骤如下：

- 在 vertices 数组中增加你要添加的三角形的顶点属性，如果某个顶点由多个三角形共享，则只添加一次即可。
- 添加 indices 数组，作为索引缓冲对象内容。将每个三角形的顶点做 vertices 数组中的下标保存在 indices 中，如下：

```
//索引缓冲对象内容
unsigned int indices[] = {
    0, 1, 2,    //第1个三角形
    0, 4, 3,    //第2个三角形
    5, 6, 7,    //第3个三角形
    8, 9, 10    //第4个三角形
};
```

- 类似添加 VBO 的操作，添加 EBO。即对 EBO 调用 glGenBuffers、glBindBuffer、glBufferData 三个函数进行处理。
- 使用 glDrawElements 函数绘制三角形。最终调用为

  glDrawElements(GL_TRIANGLES, 12, GL_UNSIGNED_INT, 0);

绘制结果如下：