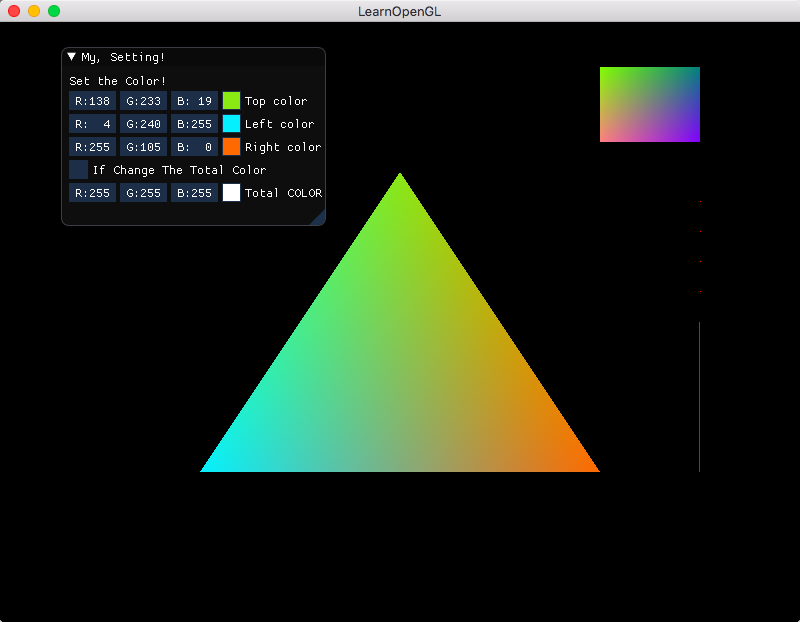
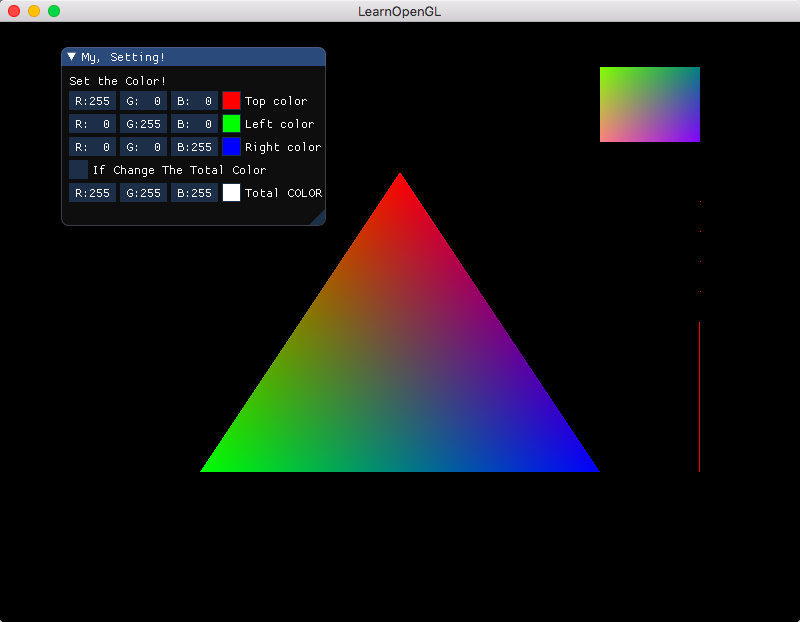
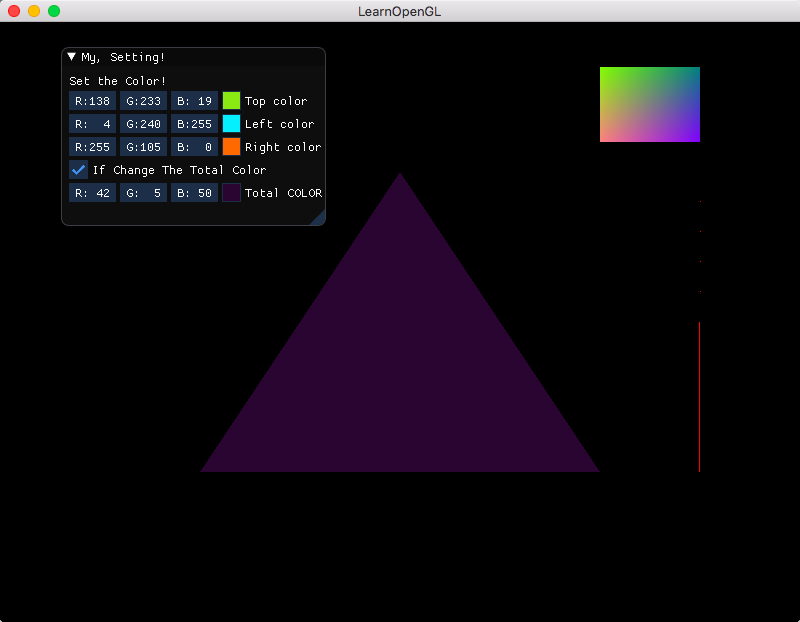
图形学作业HW2

学号：16340315

姓名：朱俊凯

运行截图：





**Bonus：**

1.三角形右边有一条红色的线的图元，线上方有一些红色的点的图元（比较小，看不太清）。

2.右上的正方形（两个三角形组成）以及主三角，三个三角形是通过一个EBO实现的。

问题：

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。

这是片段插值的结果，首先我们的的片段着色器没有直接赋予颜色，而是从顶点着色器处得来的输入颜色。当渲染一个三角形是，光栅化阶段通常会造成比原指定顶点多的片段，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，它会插值所有片段着色器的输入变量。

实现思路：

imgui部分，一直配置不太理想，所以有部份借鉴了TA提供的imgui库里example的opengl3的例子。

首先是一些**基本的配置，以及对应函数**

1.首先初始化库glfwInit()

2.设置版本号，设置核心模式glfwWindowHint()（我是Mac OS，还要用这条命令去让这些配置生效）

3.创先windows ，glfwCreateWindow(800, 600, "LearnOpenGL", NULL, NULL);

4.得到window的context ，glfwMakeContextCurrent(window);

5.设置缓存刷新时间，使得帧同步 ，glfwSwapInterval(1)

6.设置控制窗口位置和大小 ，glViewport(0, 0, 800, 600);

7.注册窗口大小调整时的回调函数glfwSetFramebufferSizeCallback(window, framebuffer\_size\_callback);（函数是自己实现的，主要是也调用glViewport）

8.初始化那些加载器的库gl3wInit()，glewInit()，gladLoadGL()

9.对ImGui的一些基本配置，ImGui::CreateContext();ImGuiIO& io = ImGui::GetIO(); (void)io; ImGui::StyleColorsDark();

10.初始化两个用于结合imgui和opengl的库ImGui\_ImplGlfw\_InitForOpenGL (window,true);ImGui\_ImplOpenGL3\_Init(glsl\_version);

11.由于要配置着色器，所以前面要先导入两个用GLSL编写的着色器代码字符串

12.编译这些着色器，并且检测错误，然后用连接着色器连接两个着色器，并删除那两个着色器。

三角形基本信息设置

1.在一个vertices数组内记录下各个顶点的位置和颜色。

2.在indices里记录各个三角形所需顶点在数组中对应的位置，很显然这样可以防止一些重复的点在vertices数组中多次出现。

3. ImVec4创建几个颜色变量用于获取后续的ImGui对颜色的一些处理。

渲染循环

1. glfwPollEvents函数检查有没有触发什么事件，processInput(window);一个自己处理键盘输入的函数

2.调整窗口大小，清颜色之类的操作

3.创建IMGUI帧，ImGui\_ImplOpenGL3\_NewFrame(); ImGui\_ImplGlfw\_NewFrame(); ImGui::NewFrame();

4.设置UI内容，ImGui::Begin（），ImGui::ColorEdit3（），……，ImGui::End();

5.根据UI值修改对应变量vertices里的颜色值。

6.创建VAO，VBO，EBO，使用glBindVertexArray绑定VAO，从绑定之后起，我们应该绑定和配置对应的VBO和属性指针，之后解绑VAO供之后使用，VBO，EBO用于管理一部分内存，用于存放一些顶点数据。

7. GL\_ARRAY\_BUFFER形式将顶点信息存到VBO里面，GL\_ELEMENT\_ARRAY\_BUFFER形式将indices信息存到EBO里面。这两种是不同的缓冲类型定义。

8.设置顶点属性指针glVertexAttribPointer（），glEnableVertexAttribArray（）

9.重新绑定VAOglBindVertexArray(VAO);

10.用里面的顶点，通过基本图元的绘制函数进行绘制，因为之前就有不同缓冲区的定义，所以EBO用glDrawElements（），VBO用glDrawArrays（），

11.进行IMGUI的渲染，主要因为想UI层在图形层之上，ImGui::Render(); ImGui\_ImplOpenGL3\_RenderDrawData(ImGui::GetDrawData());

12.交换缓存，glfwSwapBuffers（）函数会交换颜色缓冲（它是一个储存着GLFW窗口每一个像素颜色值的大缓冲），它在这一迭代中被用来绘制，并且将会作为输出显示在屏幕上。

循环结尾

1.关闭摧毁所有UI和窗口，终止进程，ImGui\_ImplOpenGL3\_Shutdown(); ImGui\_ImplGlfw\_Shutdown();ImGui::DestroyContext();glfwDestroyWindow(window);glfwTerminate();

主要的**function/algorithm**上面已经都讲了，下面简短再描述一下：

首先，对环境配置，然后配置着色器，接着，设置顶点颜色和位置的数组，和顶点索引，接着绘制UI，并从中获得颜色信息，修改数组内的颜色信息，绑定到内存指针，将这些信息写进缓冲区里，使用着色器，绘制图像，绘制UI，结束进程。