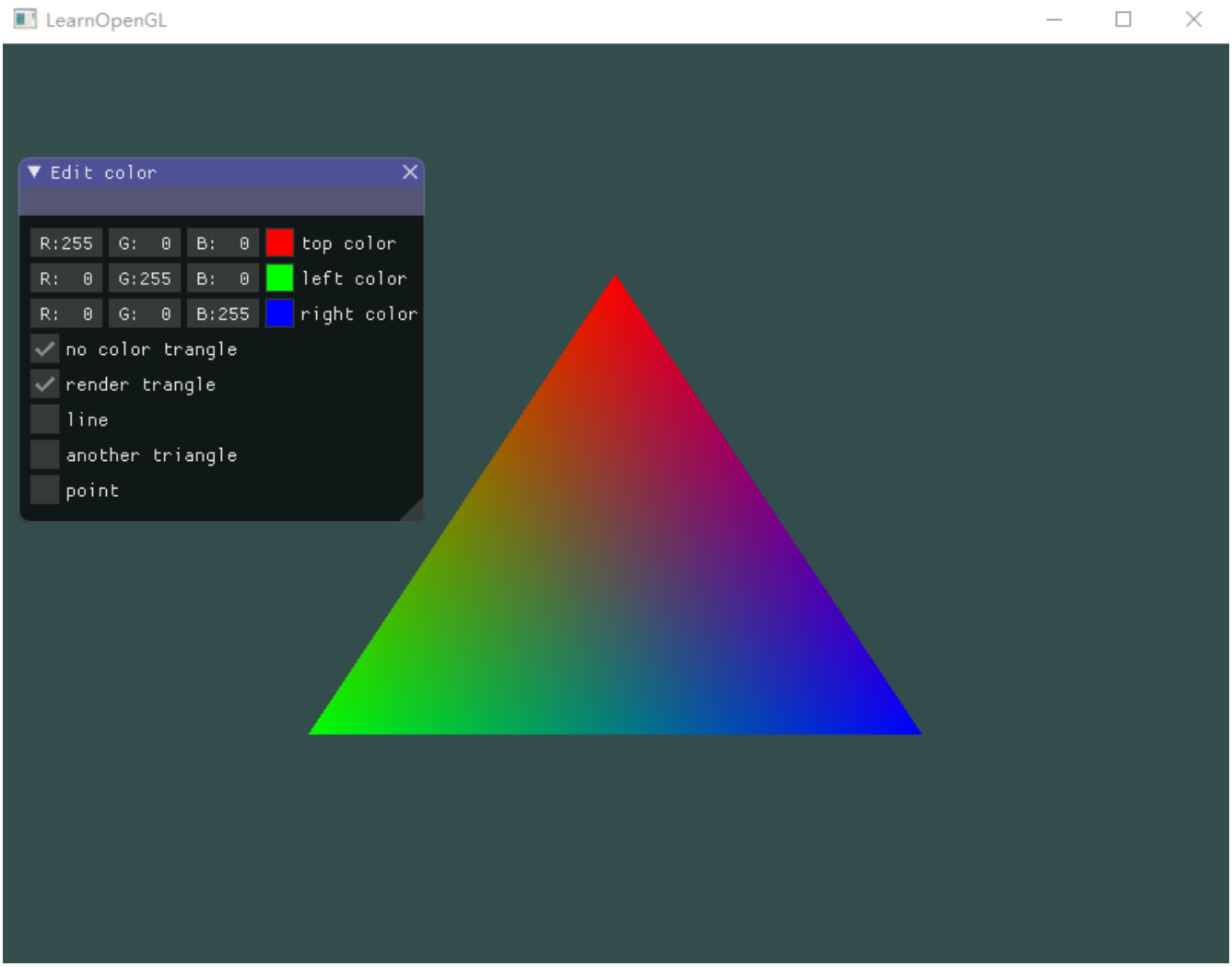
作业2实验报告

实验要求

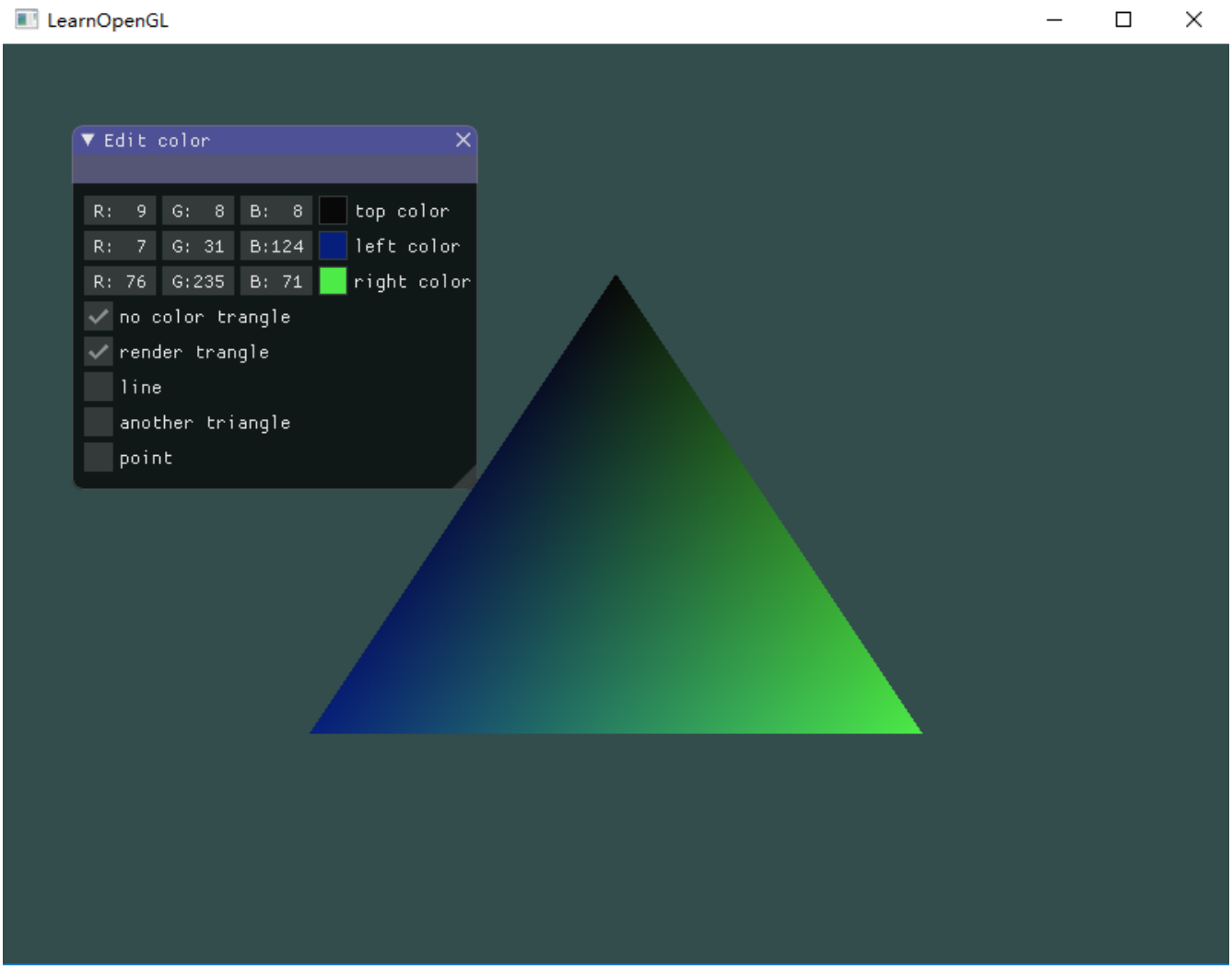
1. 使用OpenGL(3.3及以上)+GLFW或freeglut画一个简单的三角形。
2. 对三角形的三个顶点分别改为红绿蓝，并解释为什么会出现这样的结果。
3. 给上述工作添加一个GUI，里面有一个菜单栏，使得可以选择并改变三角形的颜色。
4. Bonus:
    1. 绘制其他的图元，除了三角形，还有点、线等。
    2. 使用EBO(Element Buffer Object)绘制多个三角形。

实验结果截图

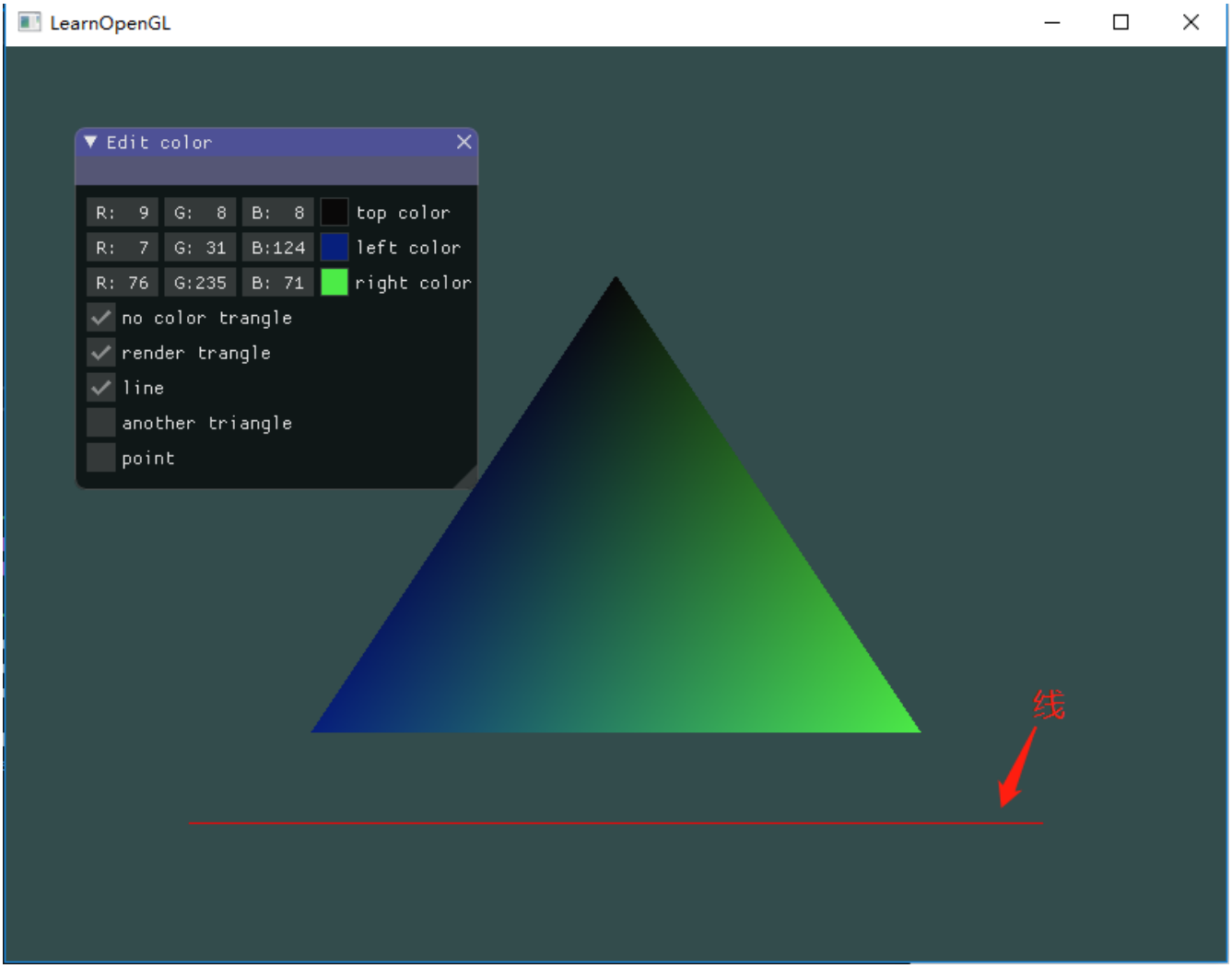
1. 三角形三个顶点为红绿蓝，带有GUI。



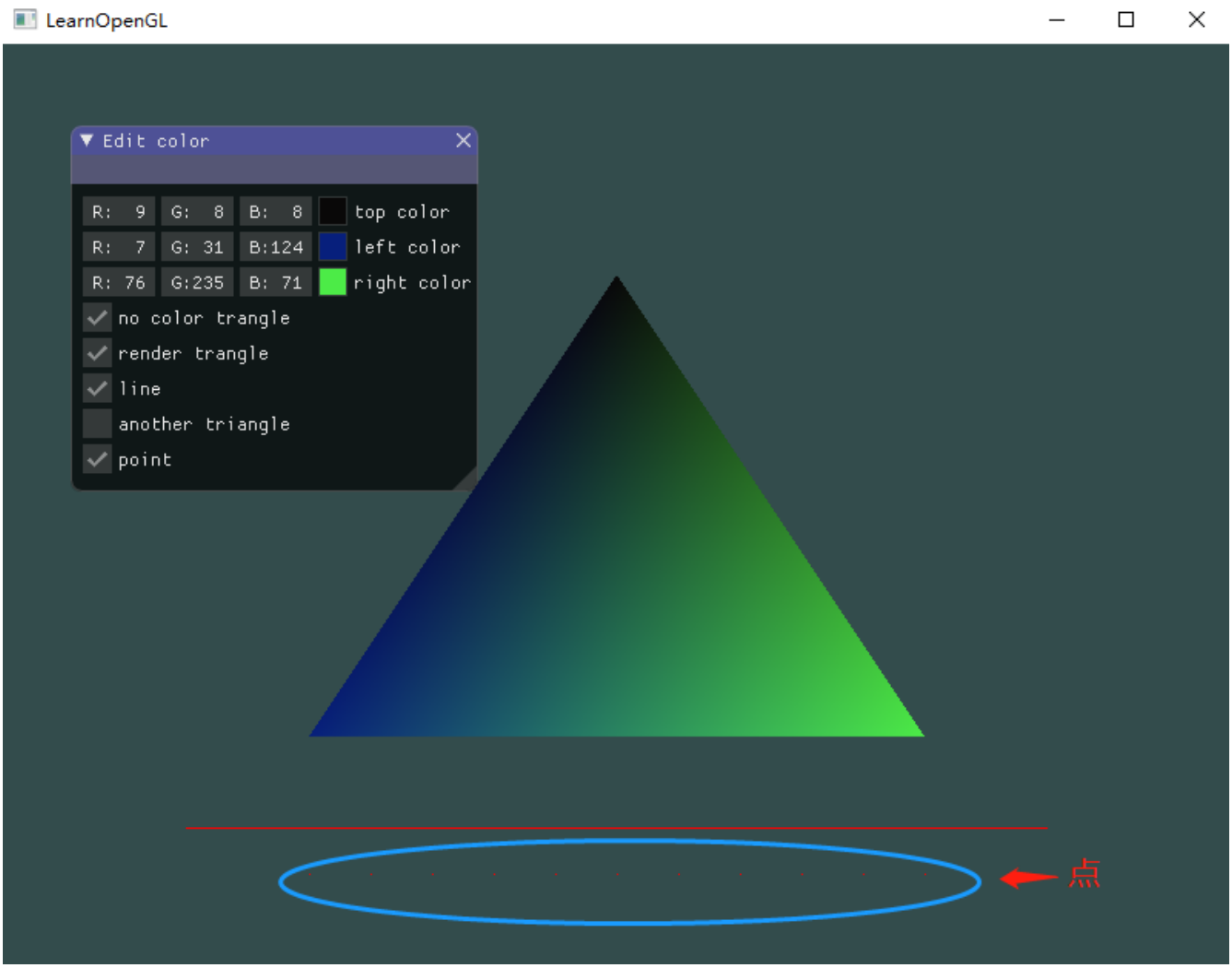
2. 点击GUI颜色表可更改任意顶点的颜色。


LearnOpenGL
Edit color
R: 9
G: 8
B: 8
top color
R: 7
G: 31
B:124
left color
R: 76
G:235
B: 71
right color
no color trangle
render trangle
line
another triangle
point

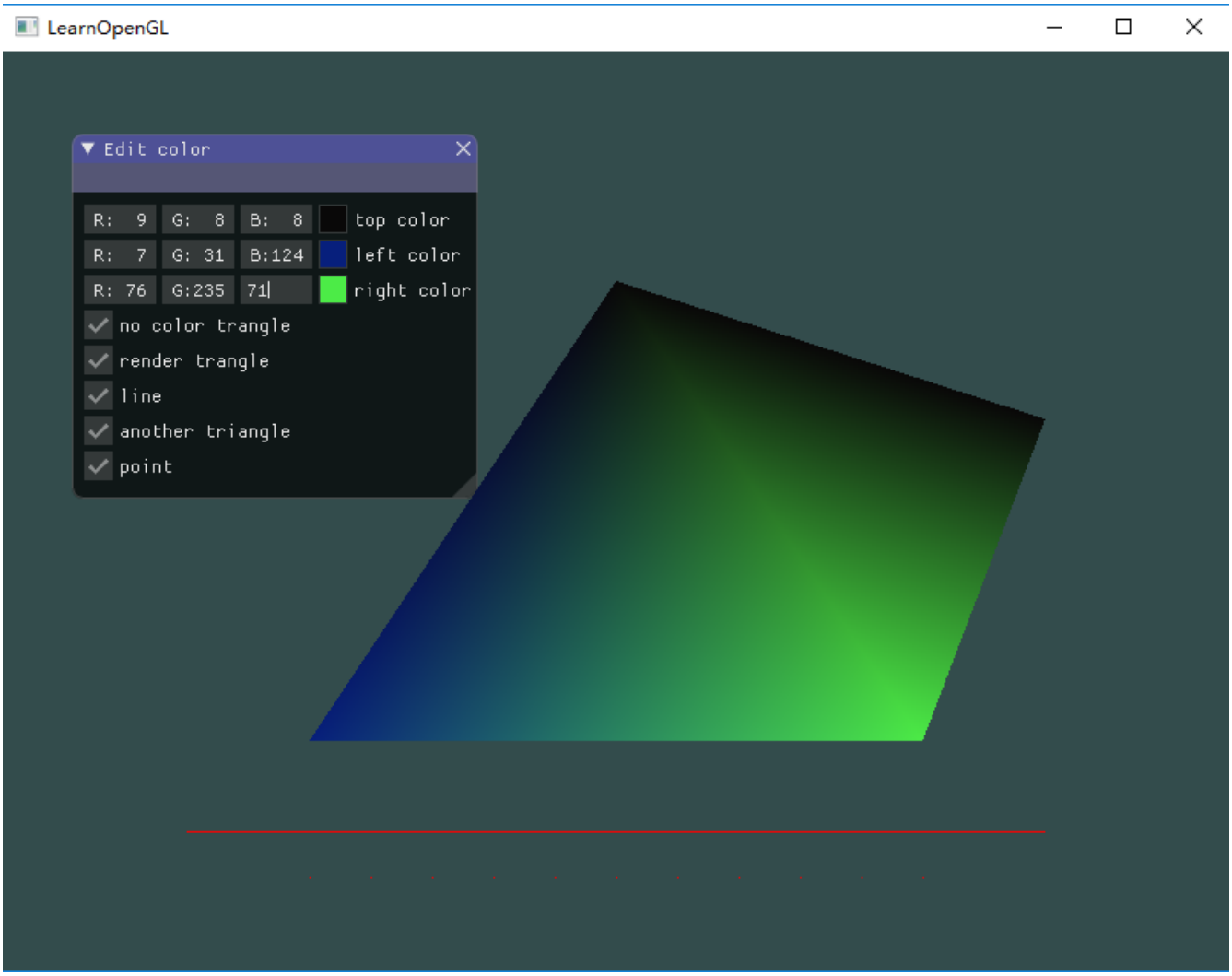
3. 展示线。


LearnOpenGL
Edit color
R: 9
G: 8
B: 8
top color
R: 7
G: 31
B:124
left color
R: 76
G:235
B: 71
right color
no color trangle
render trangle
line
another triangle
point
线

4. 展示点。


LearnOpenGL
Edit color
R: 9
G: 8
B: 8
top color
R: 7
G: 31
B:124
left color
R: 76
G:235
B: 71
right color
no color trangle
render trangle
line
another triangle
point
点

5. 展示另外一个三角形。



实验主要函数算法介绍

1. 三角形三个顶点为什么可以变为红绿蓝。

   可以变色的原因是由于使用了着色器，着色器是运行在GPU上的小程序，这些小程序为图形渲染管线的某个特定部分而运行。着色器分为两种，一种是顶点着色器，一种是片段着色器，这两个着色器可以由我们自己编写并编译，正因为可以自己编写，所以也就可以自己决定想怎么渲染颜色。

   每一个着色器有输入输出，当顶点着色器的输出作为片段着色器的输入时，只要我们给与每个顶点所需要的颜色，则可以渲染出三个顶点分别为红绿蓝的效果。

   那么重点就在于如何给顶点着色器输入位置和颜色，并且如何将顶点着色器的输出作为片段着色器的输入。

   在输入位置和颜色时，从教程中可以知道是计算数组中的位置、颜色属性对应的位置进行输入:

```
// 位置属性
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)0);
glEnableVertexAttribArray(0);
// 颜色属性
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)(3*
sizeof(float)));
glEnableVertexAttribArray(1);
```

而要将顶点着色器的输入作为片段着色器的输出时，则需要将着色器链接为一个着色器程序对象，如何在渲染的时候激活这个着色器程序，当链接着色器至一个程序的时候，它会把每个着色器的输出链接到下个着色器的输入。所以我们按照顺序链接顶点着色器和片段着色器，就可以将顶点着色器的输出作为片段着色器的输入。

```
// 着色器程序链接
ID = glCreateProgram();
glAttachShader(ID, vertexShader);
glAttachShader(ID, fragmentShader);
glLinkProgram(ID);

//检查是否链接成功
glGetProgramiv(ID, GL_LINK_STATUS, &success);
if (!success) {
    glGetProgramInfoLog(ID, 512, NULL, infoLog);
    std::cout << "ERROR::SHADER::PROGRAM::LINKING_FAILED\n" << infoLog << std::endl;
}
// 链接到程序后删除着色器对象
glDeleteShader(vertexShader);
glDeleteShader(fragmentShader);
```

我根据教程的提示，将着色器的内容写成了一个类，直接调用类构造函数即可实现着色器的编译、链接了，然后再渲染时激活其程序即可。

2. imGUI的使用

主要是根据github上的例子进行学习使用的，通过博客和文档了解到gui上有checkbox和ColorEdit3这两个函数。学习这两个函数的用法后，便将原本三角形的颜色静态值改为一个变量，这样当更改颜色时便是更改了变量的值，重新渲染时便可以更改颜色。

```
float vertices[] = {
// 位置              // 颜色
// 三角形
0.5f, -0.5f, 0.0f,  right_color.x, right_color.y, right_color.z,   // 右下
-0.5f, -0.5f, 0.0f,  left_color.x, left_color.y, left_color.z,   // 左下
0.0f,  0.5f, 0.0f,  top_color.x, top_color.y, top_color.z,    // 顶部
0.7f,  0.2f, 0.0f,  top_color.x, top_color.y, top_color.z,
   ...};

ImGui::ColorEdit3("top color", (float*)&top_color);
ImGui::ColorEdit3("left color", (float*)&left_color);
ImGui::ColorEdit3("right color", (float*)&right_color);
```

3. bonus画线、点、另外的三角形

这个则是根据教程上给与的函数变量进行设置的，`GL_POINTS、GL_TRIANGLES、GL_LINE_STRIP`，绘制指令的调用把图元传递给OpenGL，从而绘制出了我们需要的形状。

这里还有函数`glDrawArrays(GL_LINE_STRIP, 4, 2);`的使用，第一个参数是我们需要的形状绘制指令，第二个参数是数组开始的位置，第三个是使用多少个点。通过这个函数进行线和点的绘制。

画另外的三角形则根据教程上来使用便可。