Homework2

这次的作业内容是编写一个GUI界面以及使用OpenGL来绘制基本的图元。首先我选择使用ImGUI来实现GUI界面，它是一个C++图形用户界面库，只需要将几个头文件以及源文件包含到项目中就可以使用了，占用很小。

GUI界面编写

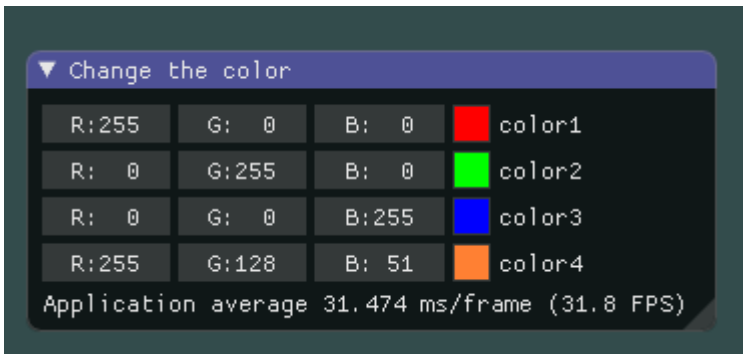
ImGUI项目开发者提供了多个版本的库文件，这里因为我们项目基于GLFW+opengl3.0(以上版本)实现，所以这里需要包含`imgui.h, imgui_impl_glfw.h, imgui_impl_opengl3.h`这三个头文件以使用它。

首先要做的是初始化ImGUI的使用环境，如设置窗口的显示style，有`StyleColorsClassic, StyleColorsDark, StyleColorsLight`三种，这里选择的是第一种，其余还有许多参数可以配置，其中最重要的是将窗口与当前GLFW控制的视窗绑定，通过：

```
ImGui_ImplGlfw_InitForOpenGL(window, true);
ImGui_ImplOpenGL3_Init(glsl_version);
```

其中第一条语句为绑定window，第二条语句指定了glsl着色器语言的版本，这里设置的是`#version 330`。

设置好初始化参数后，ImGUI界面与OpenGL元素一样在渲染循环中进行渲染，此次作业需要绘制三角形并改变其颜色，在GUI界面上加入调色板：



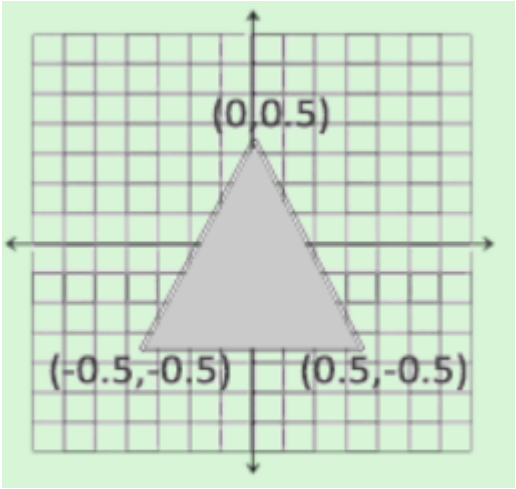
至此，GUI界面编写先告一段落，后面再将它与三角形颜色的变化联系起来。

OpenGL图元绘制

绘制一个简单的三角形

使用OpenGL来绘制三角形，我们需要编写的是与三角形各顶点的数据、存储数据的对象（VAO、VBO、EBO等）、对象的渲染以及顶点着色器、片段着色器相关的代码。其中首先需要设定三角形各顶点的数据，这里可能包括位置与颜色两种数据，首先只关心位置。

我们设置的坐标会经由顶点着色器处理成为标准化设备坐标，其x、y与z值都在-1.0到1.0之间，所以说我们给出来的坐标是相对位置，并不是准确位置。比如将三角形的三个顶点坐标分别设为`(-0.5, -0.5, 0.0), (0, 0.5, 0.0), (0.5, -0.5, 0.0)`，其中第三个坐标代表深度，这样此三角形就处在窗口中央，其大小随窗口大小而变化，如：



有了顶点数据之后我们通过顶点缓冲对象（VBO）管理这些数据（内存中），指定顶点数据的解释方法（此处只含位置数据），并将数据绑定到一个顶点数组对象（VAO）上，它会储存顶点属性配置与定点缓冲对象。渲染时只需绑定对应的VAO后调用draw函数就可以绘制图像了，此时顶点着色器中只含位置属性，片段着色器将图形渲染成固定颜色，这样就完成了一个简单的三角形：

将三角形三个顶点改成红绿蓝

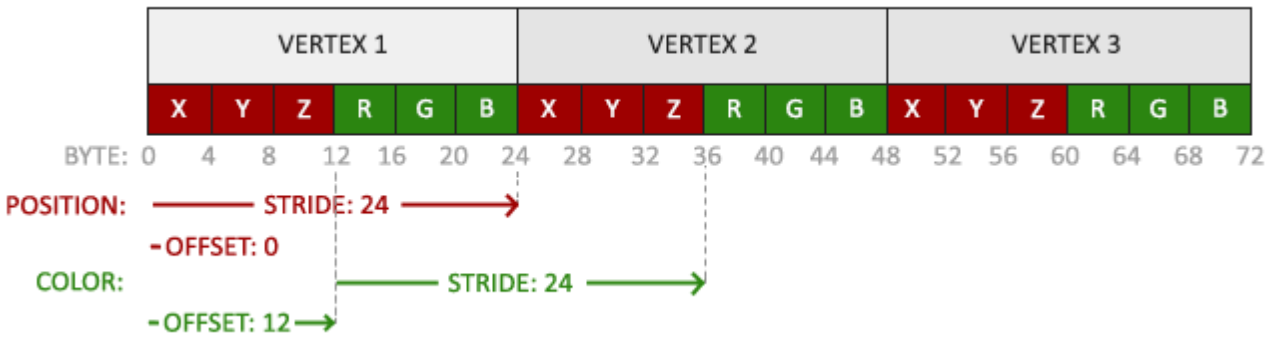
为了做到这一点，需要为顶点数据添加颜色属性，将刚刚的顶点数据：

```
float firstTri[] = {
        //location
        0.0f,  1.0f, 0.0f,
        -0.5f, 0.0f, 0.0f,
        0.5f, 0.0f, 0.0f
};
```

改为：

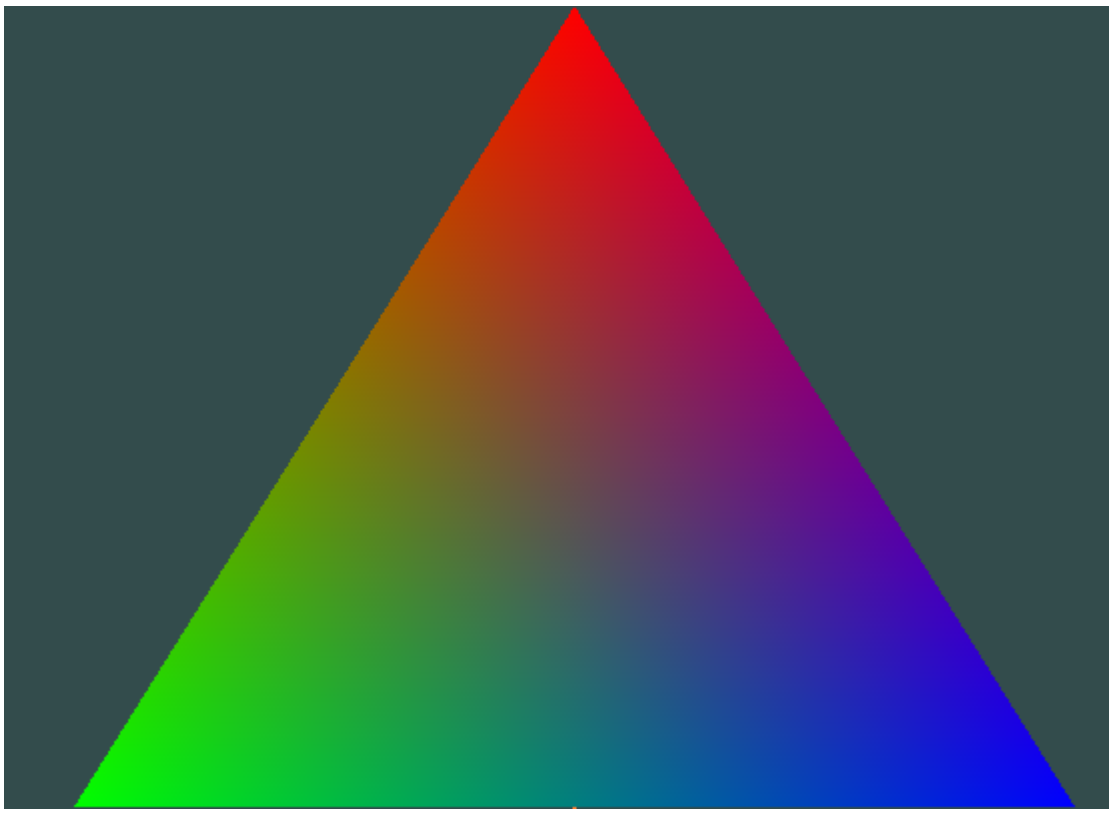
```
float firstTri[] = {
        //location          //color
        0.0f,  1.0f, 0.0f,  1.0f, 0.0f, 0.0f,
        -0.5f, 0.0f, 0.0f,  0.0f, 1.0f, 0.0f,
        0.5f, 0.0f, 0.0f,   0.0f, 0.0f, 1.0f
};
```

此时数据的存储格式发生了变化，变成了：



所以在指定顶点存储格式时就需要注意到下一个顶点的步长为6。

着色器部分也需要更改，此时给顶点着色器添加颜色属性，并将其输出到片段着色器中进行渲染，得到的结果：

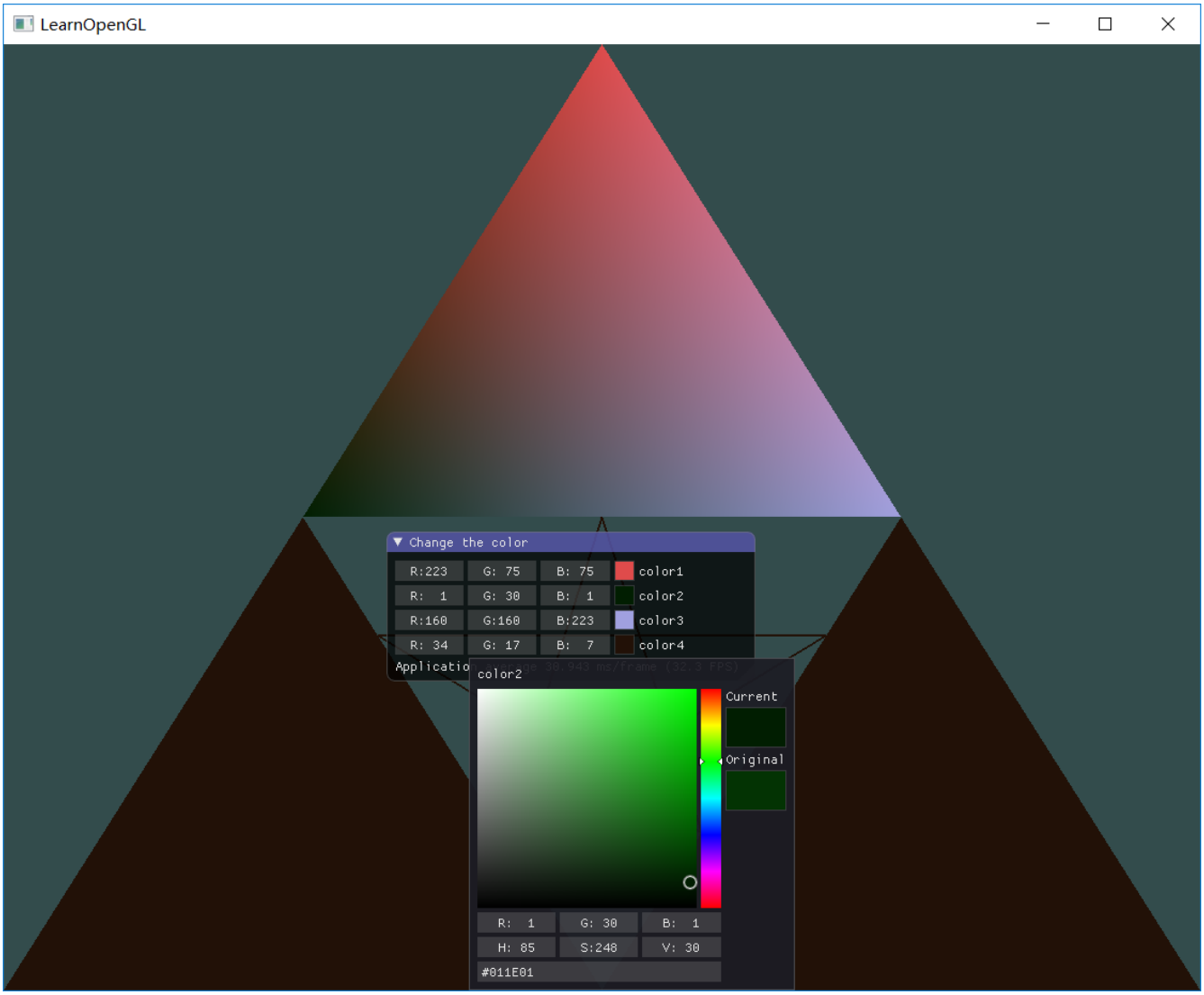


分别设置三个顶点的颜色之后，出现的结果却是渐变形的，像是一个调色板。这是因为在片段着色器中会进行“片段插值”，在光栅化阶段通常会造成比原指定顶点更多的片段，光栅根据每个片段在三角形上所处相对位置决定这些片段的位置。之后对于这些位置会对片段着色器的输入变量（颜色）做插值，其颜色成百分比的形式计算并渲染出来。

改变三角形的颜色

从之前GUI界面的截图可以看到，这里有四个改变颜色的按钮，我将其设置为上面三个分别改变三角形三个顶点的颜色，剩下一个改变另一对三角形的颜色（这一对三角形用EBO绘制）。

统一改变颜色与分别改变顶点的颜色其实现方法是不同的，需要两套独立的顶点/片段着色器来进行渲染。其中分别更改顶点的颜色需要在改变其顶点数据中的颜色属性后，在渲染循环中重新绑定bufferData。而统一更改颜色则需要在渲染循环中将颜色传输给着色器进行渲染。其结果为：

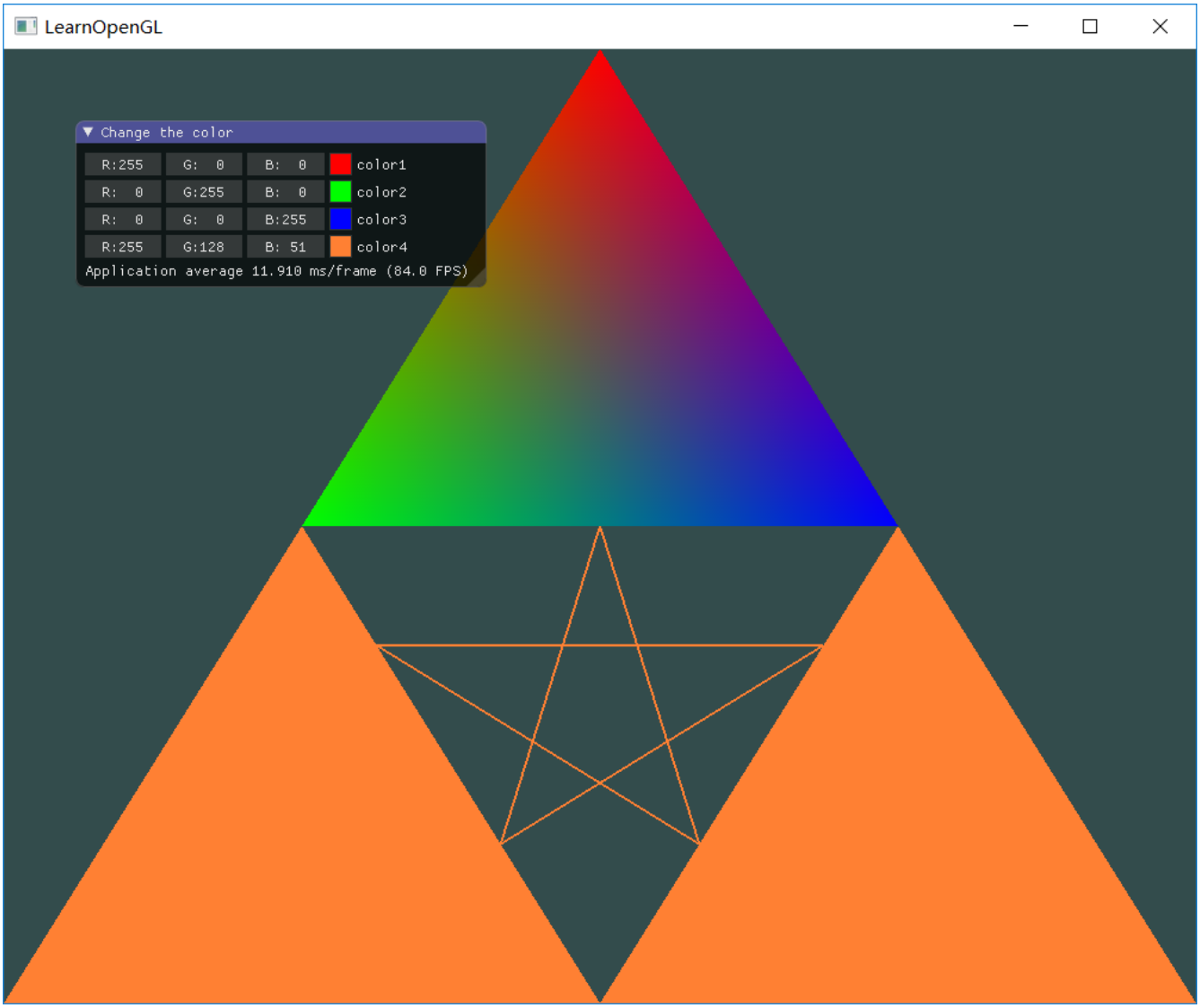


注意其中color1、color2、color3对应的是上方三角形三个顶点的颜色，color4对应的是用EBO一起绘制的两个三角形的颜色。

绘制其他图元

最后，还可以使用OpenGL来绘制线、点等图元。比如绘制线段，一样需要指定其顶点数据，并绑定对象。只需要在draw的时候将绘制类型从`GL_TRIANGLES`改为`GL_LINES`就可以了。

最终效果：


LearnOpenGL
Change the color
R:255
G: 0
B: 0
color1
R: 0
G:255
B: 0
color2
R: 0
G: 0
B:255
color3
R:255
G:128
B: 51
color4
Application average 11.910 ms/frame (84.0 FPS)