

Homework 2

Basic

实验截图

算法介绍

1. 画简单三角形
2. 对三个顶点分别改成红绿蓝

实现过程

解释图片颜色出色多色混起来的原因

3. 添加一个GUI，可以选择改变三角形的颜色

实验截图：

实现思路：

Bonus

1. 绘制其他的图元

画点

画线

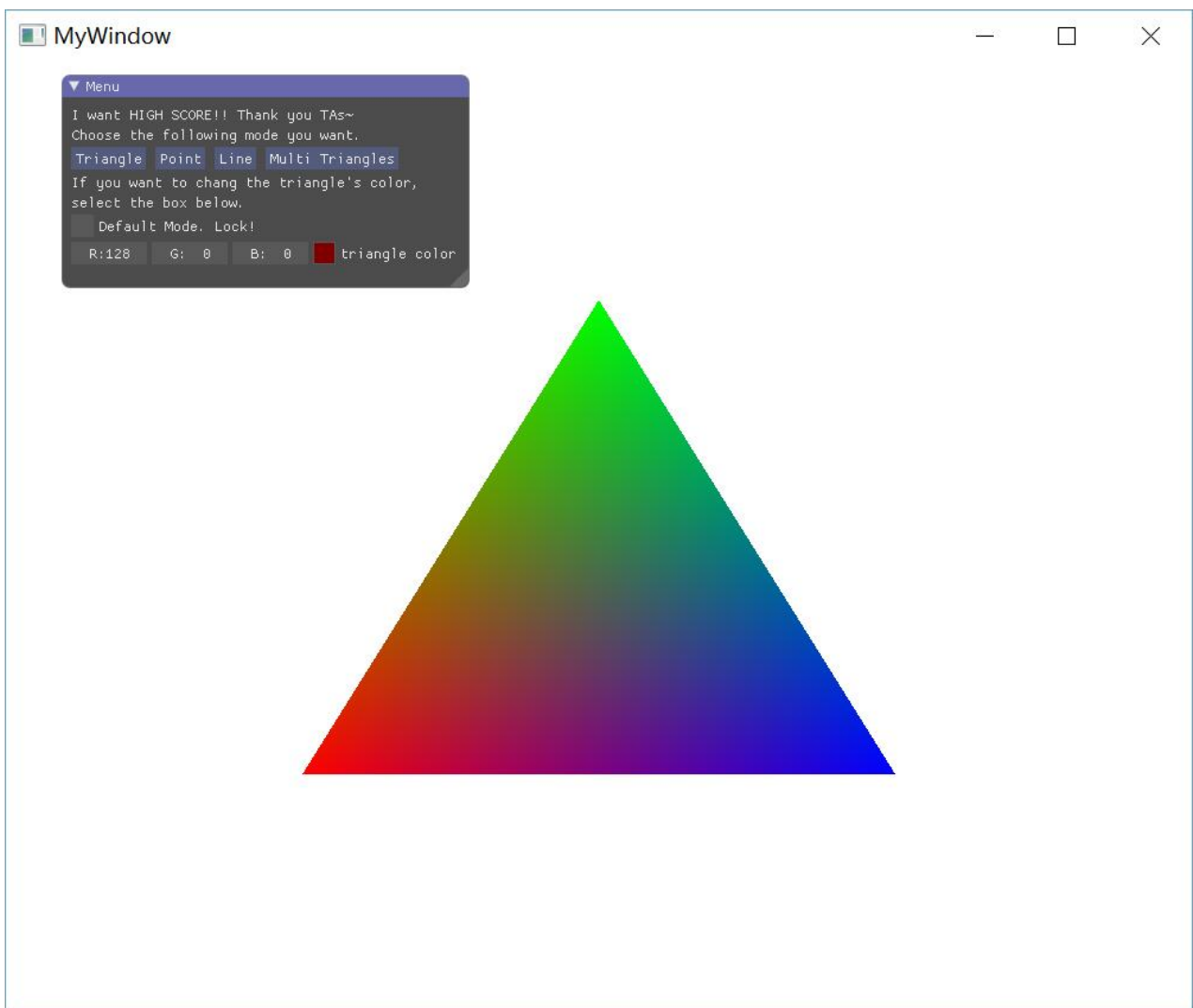
2. 使用EBO来绘制多个三角形

Homework 2

更加具体的实现成果可以参考演示视频

Basic

实验截图



算法介绍

1. 画简单三角形

按照中文教程的过程来做就行，比较简单。

1. 编译vshader和fshader，并且生成着色器程序。
2. 准备好顶点的位置信息。
3. 建立vao，vbo。绑定vao、vbo。将数据刷进vbo。
4. 进入render loop，激活着色器程序，渲染vao中的数据出来。

细节不赘述。主要记得先开始绑定VAO，之后的VBO和EBO会自动绑定到该VAO上（在该VAO解绑之前）。

2. 对三个顶点分别改成红绿蓝

实现过程

顶点着色器和片段着色器用一个变量 `vColor` 来联系起来。把顶点的颜色属性写进顶点属性数据里面。通过 `glVertexAttribPointer` 来绑定不同的属性。然后着色器的GLSL代码作稍许修改，主要是加上 `vColor` 的连接关系即可。其他步骤和第1小题一样。

解释图片颜色出色多色混起来的原因

根据[参考资料](#)。

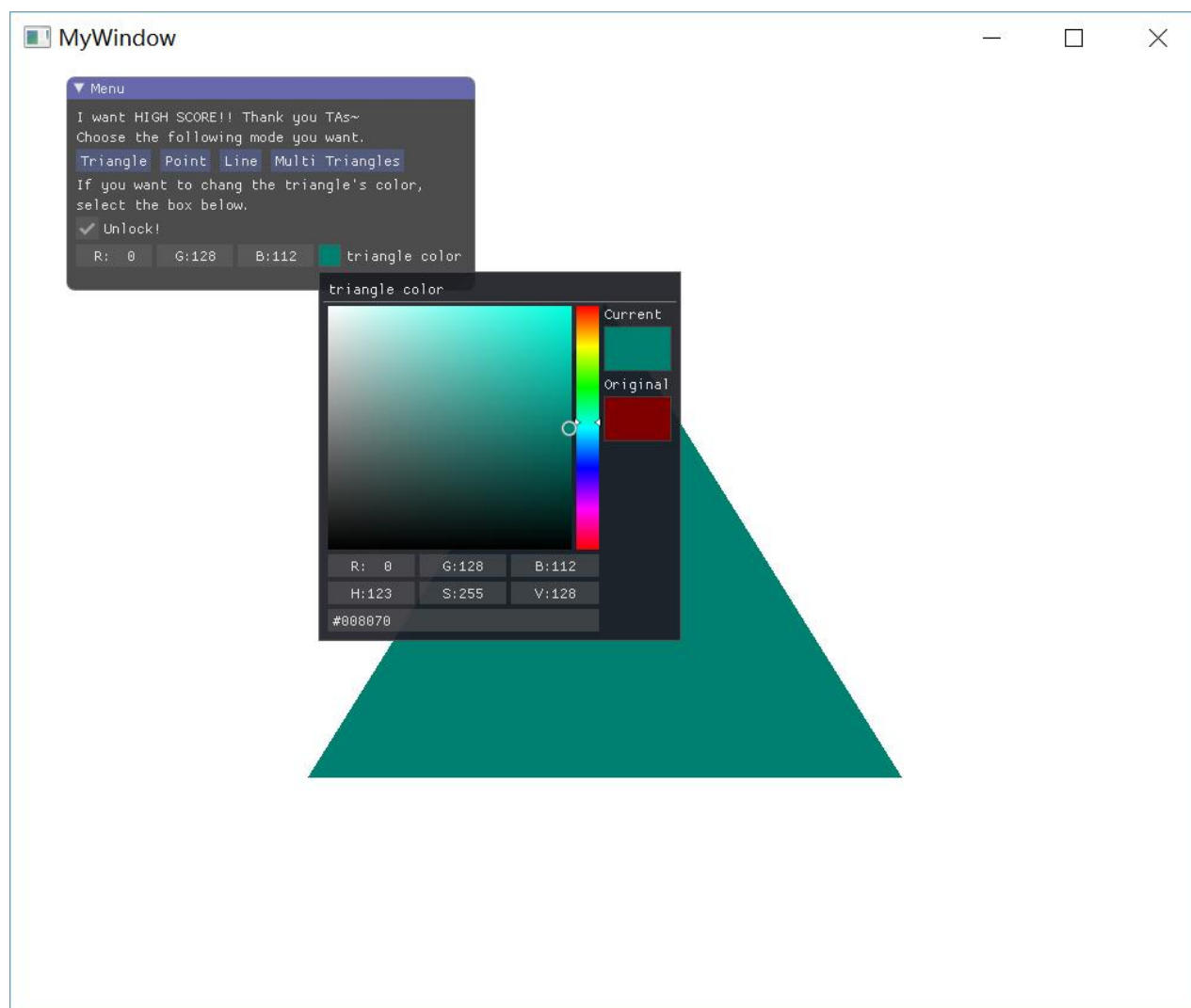
这是在片段着色器中进行的所谓片段插值(Fragment Interpolation)的结果。当渲染一个三角形时，光栅化(Rasterization)阶段通常会造成比原指定顶点更多的片段。光栅会根据每个片段在三角形形状上所处相对位置决定这些片段的位置。

基于这些位置，它会插值(Interpolate)所有片段着色器的输入变量。比如说，我们有一个线段，上面的端点是绿色的，下面的端点是蓝色的。如果一个片段着色器在线段的70%的位置运行，它的颜色输入属性就会是一个绿色和蓝色的线性结合；更精确地说就是30%蓝 + 70%绿。

这正是在这个三角形中发生了什么。我们有3个顶点，和相应的3个颜色，从这个三角形的像素来看它可能包含50000左右的片段，片段着色器为这些像素进行插值颜色。如果你仔细看这些颜色就应该能明白了：红首先变成到紫再变为蓝色。片段插值会被应用到片段着色器的所有输入属性上。

3. 添加一个GUI，可以选择改变三角形的颜色

实验截图：



实现思路：

在片段着色器中添加了两个 **uniform** 变量来帮助交互。 **bool isdefault** 和 **vec4 uni_color**

isdefault 是用来条件判断是显示先前任务的三角形还是改变三角形的颜色。

uni_color 是通过GUI修改的颜色值，用来渲染三角形的颜色。

在主函数和GUI中，通过一个checkbox的勾选来改变 `isdefault` 的值。然后调用组件 `ImGui::ColorEdit3` 去改变 `uni_color` 的值从而实现对三角形颜色的改变。

Bonus

构建多个VAO和VBO，分别存储不同模式下的顶点数据。

在GUI中使用 `button` 来选择模式，通过改变一个 `mode` 值，然后用 `switch` 语句来进行不同的渲染方案。

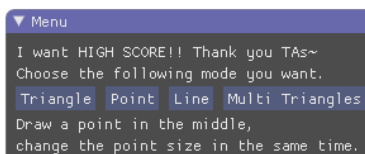
1. 绘制其他的图元

画点

```
1. bonus_shader.use();
2. glPointSize(5.0f);
3. glBindVertexArray(VAO[mode]);
4. glDrawArrays(GL_POINTS, 0, 1);
```

使用 `glPointSize(5.0f)` 来凸显一下点的位置。

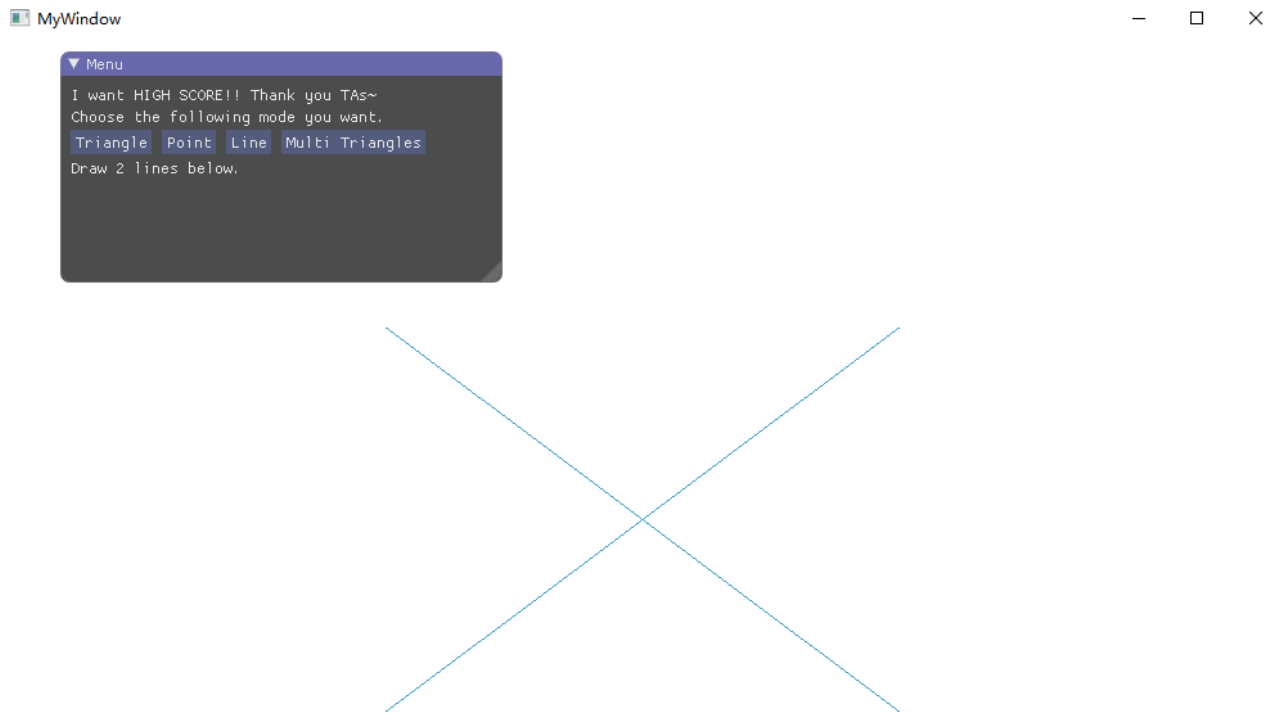
MyWindow



```
▼ Menu
I want HIGH SCORE!! Thank you TAs~
Choose the following mode you want.
Triangle Point Line Multi Triangles
Draw a point in the middle,
change the point size in the same time.
```

画线

为了图像美观画了两条交叉线。这和前面画其他图元的方法类似，只不过指定的图元值变成了 `GL_LINES` 而已。



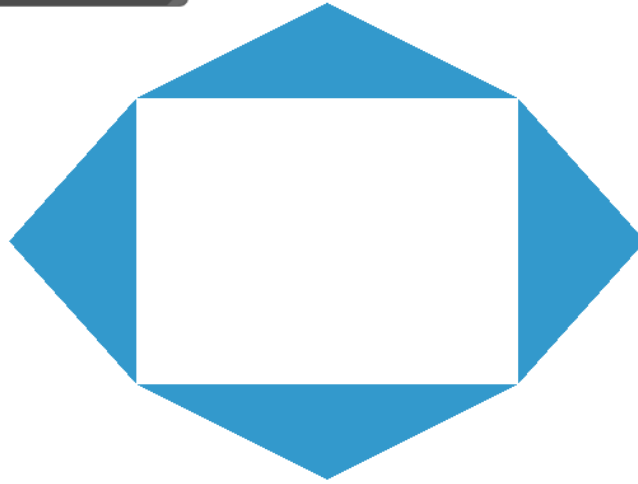
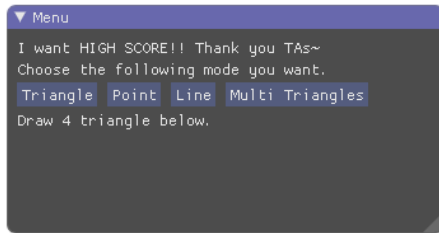
2. 使用EBO来绘制多个三角形

只需要在准备该任务的VAO时，多刷入一个EBO的数据即可

```
glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(i1), i1, GL_STATIC_DRAW);
```

然后在 Render loop 中再使用 `glDrawElements` 函数来进行绘制。

为了图像美观画了4个顶点有相连的三角形形成一个对称图案。



具体的 OpenGL 函数的参数用法参考网上的[文档](#)，没有太多赘述的必要。