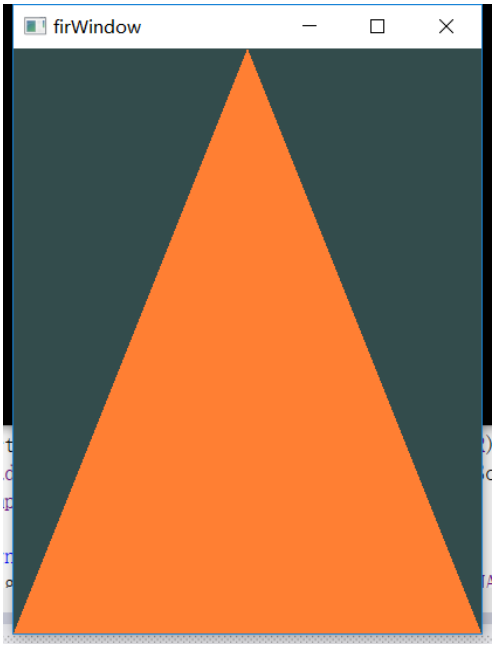
Basic:

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

截图:



实现思路:

首先是按照教程配置好环境，关键在于把相关的 lib 文件和 include 文件要添加到项目中的那个配置路径里，让 IDE 知道去那里找。

然后就是调用函数创建一个窗口，可以设定窗口大小和名字等属性，并设置为当前线程的上下文，再设置视口控制渲染窗口的位置和大小。接着为了在改变窗口大小的时候也可以改变视口的大小需要注册一个回调函数进行调整。最后为了避免窗口一下就结束，还要弄一个渲染循环进行不断的渲染。

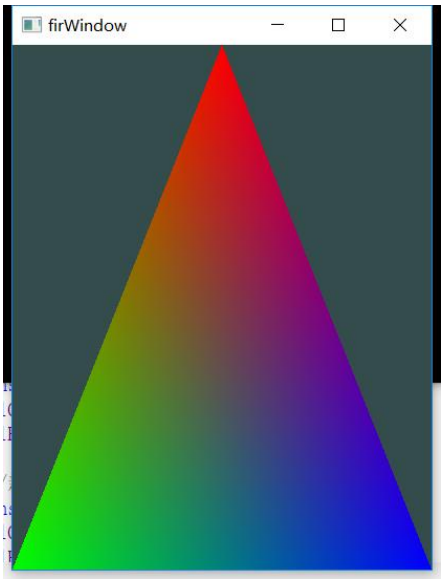
到了画三角形的时候，首先要先创建编译两个着色器，定点着色器和片段着色器。在顶点着色器中我们可以对输入的位置信息等进行处理，例如将坐标进行转换，传到着色器的输出 gl_Position 变量。片段着色器做的是计算像素最后的颜色输出。有 4 个元素的数组：红色、绿色、蓝色和 alpha(透明度)分量，片段着色器只需要一个输出变量，这个变量是一个 4 分量向量，它表示的是最终的输出颜色。然后就是根据我们编写的上面两个着色器源码，先创建着色器对象，将源码附加到着色器对象上，然后进行编译。接着就是创建一个着色器程序对象，将编译后的着色器对象附加到程序对象上，并进行链接。

至于三角形的数据，一开始先设定三角形顶点信息存在数组里，接着就是创建顶点数组对象（VAO），绑定，任何随后的顶点属性调用都会储存在这个 VAO 中，避免每次都要设置一堆属性。在创建一个顶点缓冲对象（VBO），它会在 GPU 内存（通常被称为显存）中储存大量顶点。使用这些缓冲对象的好处是我们可以一次性的发送一大批数据到显卡上，而不是每个顶点发送一次。绑定 VBO 后，就用 glBufferData 将顶点数据传送过去缓冲的内存中。然后还要 glVertexAttribPointer 定义了 OpenGL 该如何解释顶点数据并用 glEnableVertexAttribArray 启用数据。

最后就用上面的程序对象来绘制图元就行了，glDrawArrays 函数，它使用当前激活的着色器，之前定义的顶点属性配置，和 VBO 的顶点数据（通过 VAO 间接绑定）来绘

制图元。

2. 截图



实现思路:

之所以出现这个效果，首先是在顶点信息数组中我们添加了每个点的颜色信息，然后更改了传输数据的代码

```
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)(3 * sizeof(float)));
```

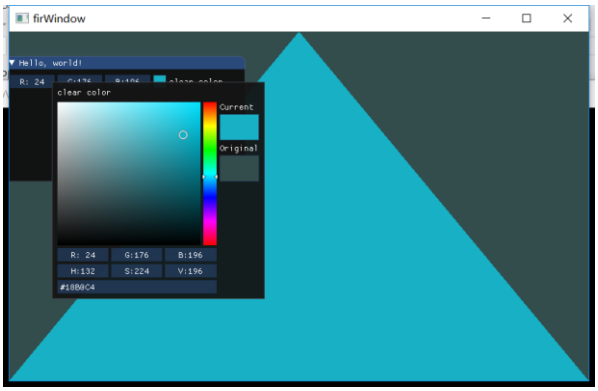
来告诉 openGL 还有不同的信息，再在顶点着色器源码中接收颜色信息再传送给片段着色器中，就完成了不同点的不同颜色。

而出现上述的效果，是因为我们只给了三个颜色，然后片段着色器进行片段插值的结果。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。

3.

截图:



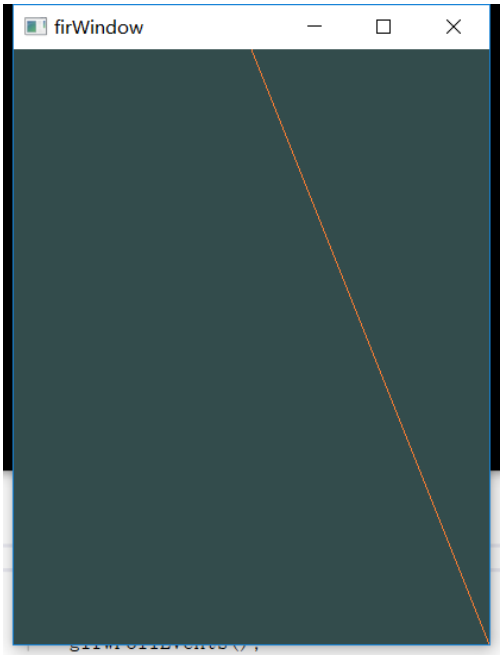
首先是下载 imgui 的源文件，注意还要把 example 里的 imgui_impl_glfw 和 imgui_impl_opengl3 添加进去，然后还要

```
#define IMGUI_IMPL_OPENGL_LOADER_GLAD
```

默认用的是 gl3w,我们用的是 glad。然后就可以用 imgui 的库函数来渲染组件了。参考了一下 example 里的代码，先要创建 imgui 的上下文，然后设定一些像 style 的属性，然后在渲染循环里面先创建 imgui 的帧，然后用 colorEdit3 来渲染出选择颜色板，并且会存到一个颜色 vec4 中，然后我就将 vertices 里面的颜色属性换成这个选择的颜色，再用 glBufferData 函数来传递顶点数据，最后进行 imgui 的 render 渲染和原本三角形的渲染就行了。

Bonus:

1. 截图



实现思路:

就在渲染原来三角形的基础上，修改了一下渲染的函数属性

```
glDrawArrays(GL_LINE_STRIP, 0, 2);
```

2. 截图:

实现思路:

在画三角形的基础上，我们利用索引缓冲对象（EBO），来减少画多个三角形有重叠的顶点时的重复操作，减少额外开销。只存储不同的顶点，并存储这些顶点的绘制顺序作为索引。与 VBO 类似，我们先绑定 EBO 然后用 glBufferData 把索引复制到缓冲里。GL_ELEMENT_ARRAY_BUFFER 当作缓冲目标。最后一件要做的事是用 glDrawElements 来替换 glDrawArrays 函数，来指明我们从索引缓冲渲染。使用 glDrawElements 时，我们会使用当前绑定的索引缓冲对象中的索引进行绘制。顺序是先 VAO，再 EBO，再 VBO。