0 在上一期的内容中，我们使用着色器（shader）和顶点对象（VAO, VBO）绘制了一个橘色的三角形。本期内容将如下进行展开：

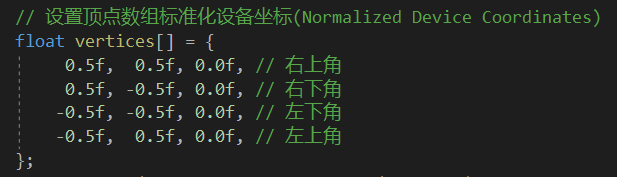
（1）在顶点数组vertices中设置四个矩形角点，并使用元素缓冲数组对象(element buffer object, EBO)来重复使用这四个角点绘制一个矩形。

（2）绘制完单色矩形之后，我们将创建两套渲染对象来实现两种不同颜色的两个矩形的绘制。

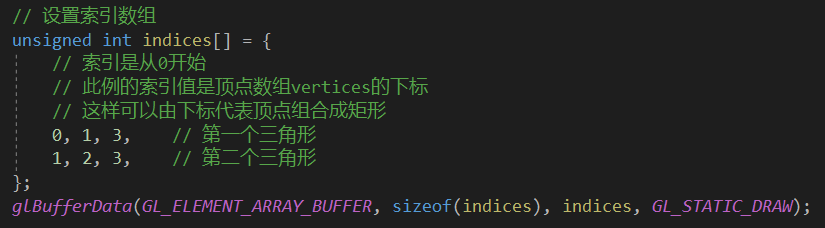
（3）最后我们再绘制一个颜色能随着时间变化的矩形。

1 绘制一个纯色的矩形。

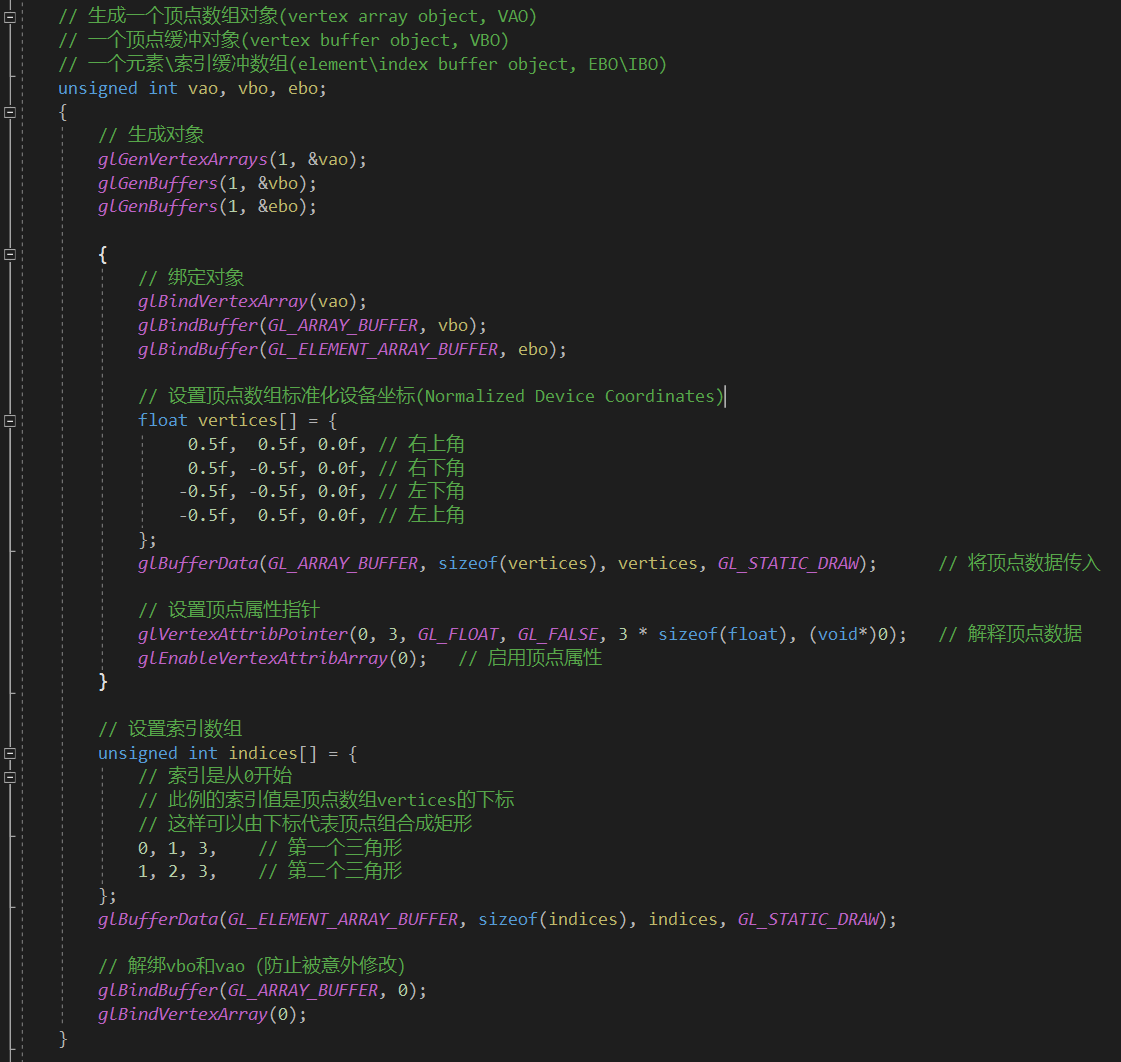
1.0 一个矩形是由两个三角形组成的，所以我们在VBO中的vertices数组中定义矩形的四个角点位置。



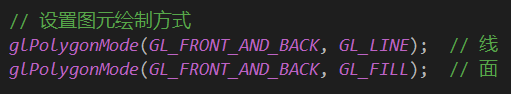
1.1 通过EBO设置以什么顺序去索引VBO中角点的顺序。



1.2 将VBO,VAO,EBO组合起来。



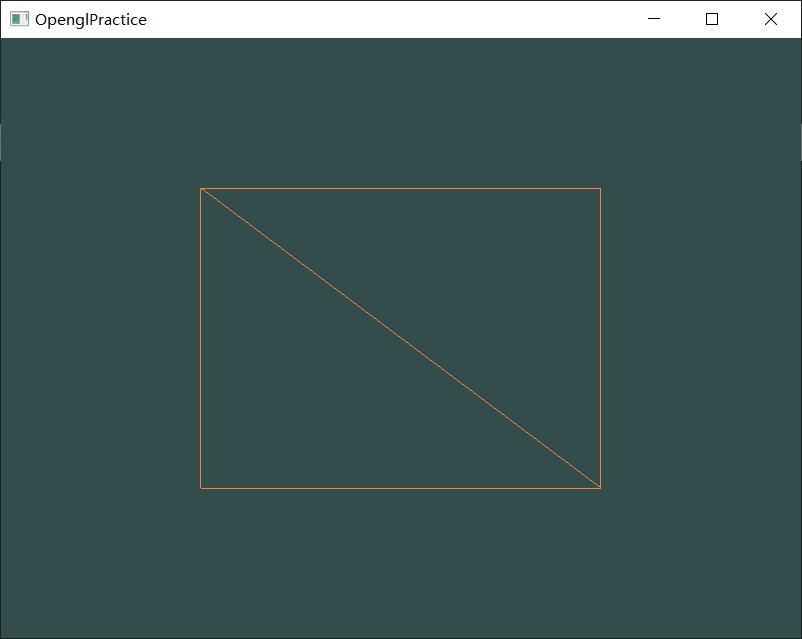
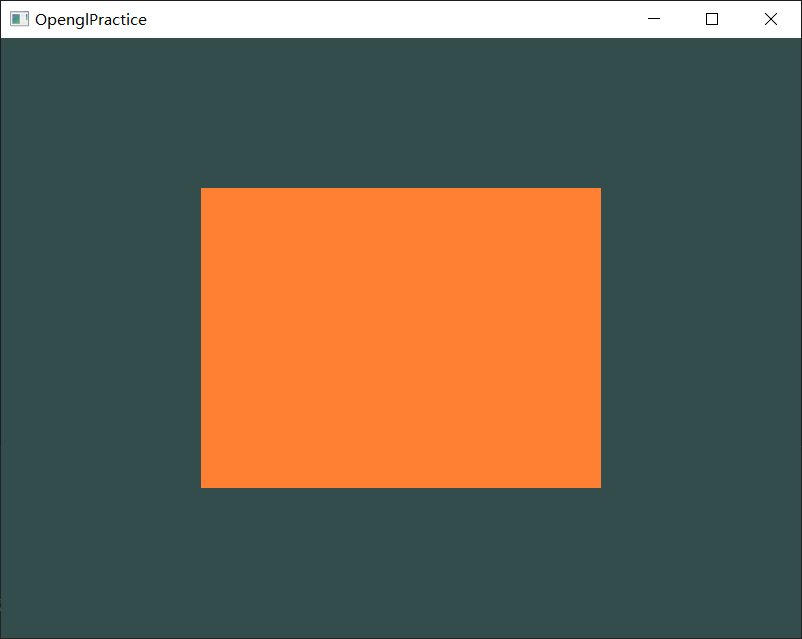
1.3 设置图源绘制的方式是：线还是面。



1.4 在渲染循环中通过我们设置的EBO规则绘制矩形（也就是两个连起来的三角形）。



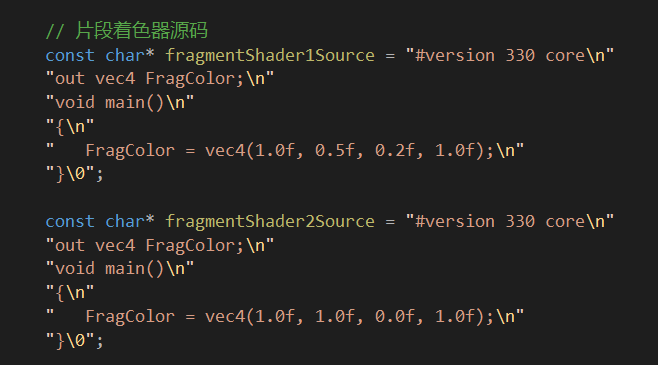
1.5 得到矩形实体和矩形线框。



2 绘制两个颜色不同的矩形。

2.0 在第一节中我们通过设置包含4个顶点坐标位置的buffer和一个橘色的shader程序共同绘制了橘色矩形。接下来我们将通过设置两个不同位置的矩形和两种不同颜色的shader程序，来绘制两个颜色不同的矩形。

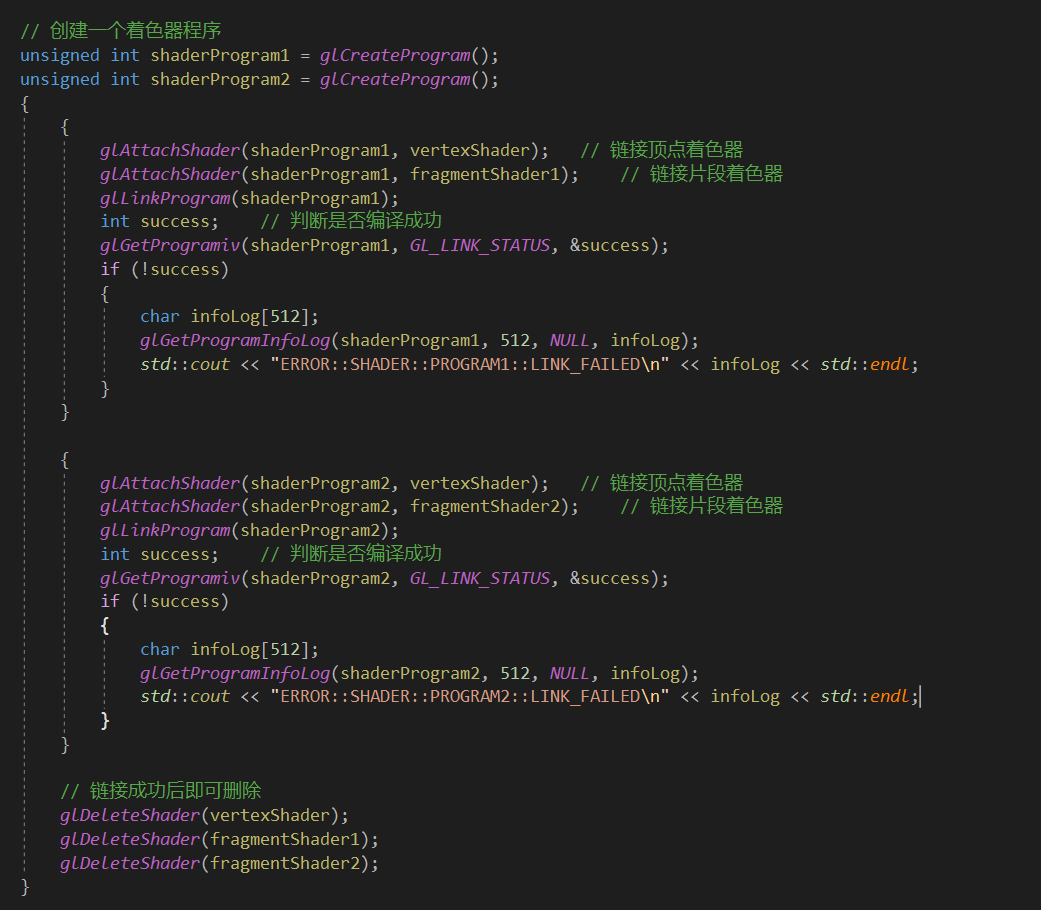
2.1 设置两种颜色的片段着色器源码。一种颜色设为橘黄色、一种颜色设为橙色。



2.2 设置两个片段着色器对象分别编译上面提到的两个片段着色器源码。

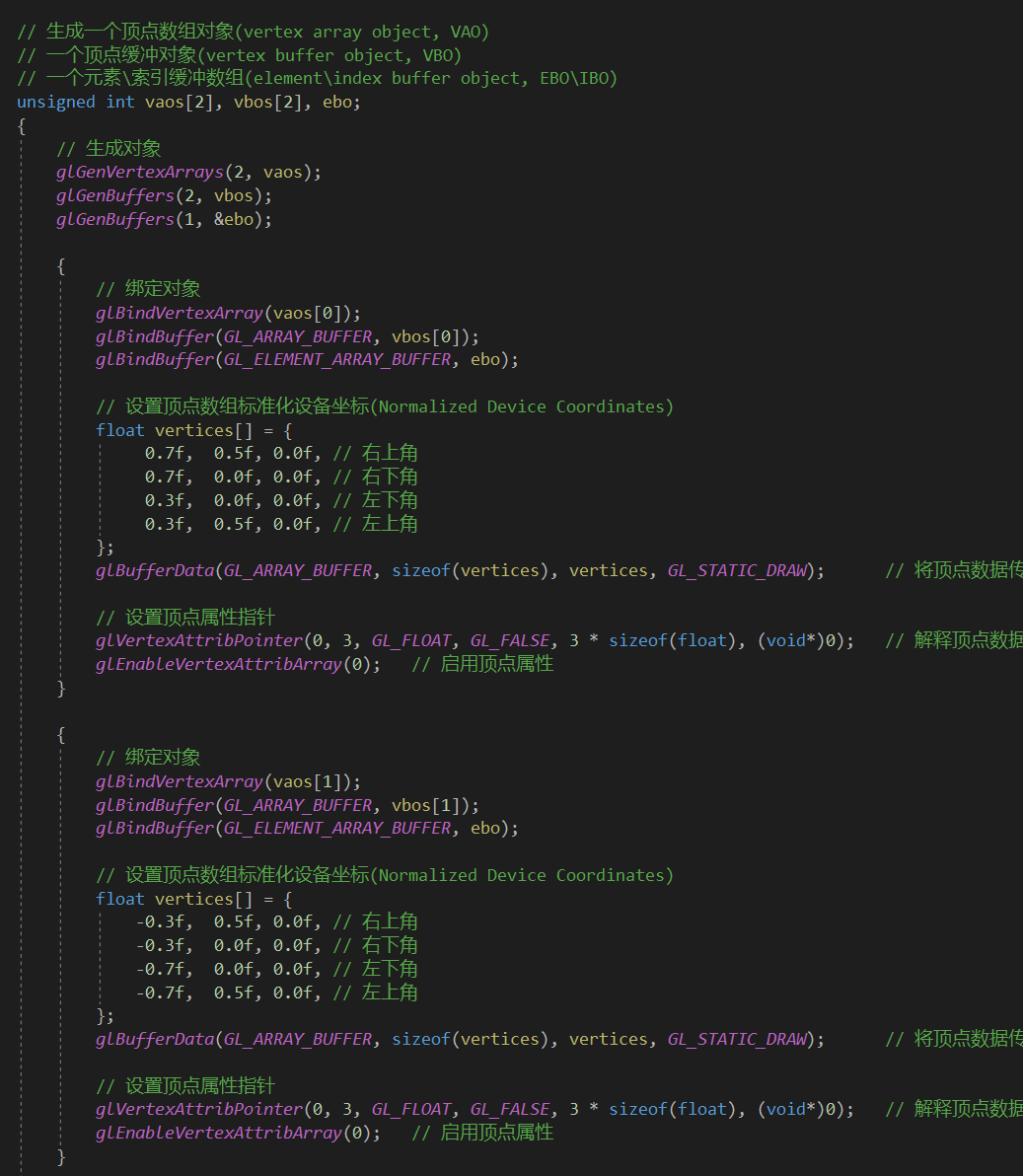


2.3 创建两个着色器程序，分别链接两个片段着色器对象和共用的顶点着色器对象。

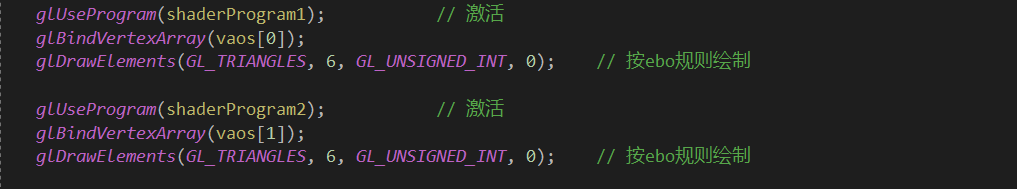


2.4 生成两个顶点数组对象和顶点缓冲对象（将VAO和VBO设置为unsigned int 数组）。

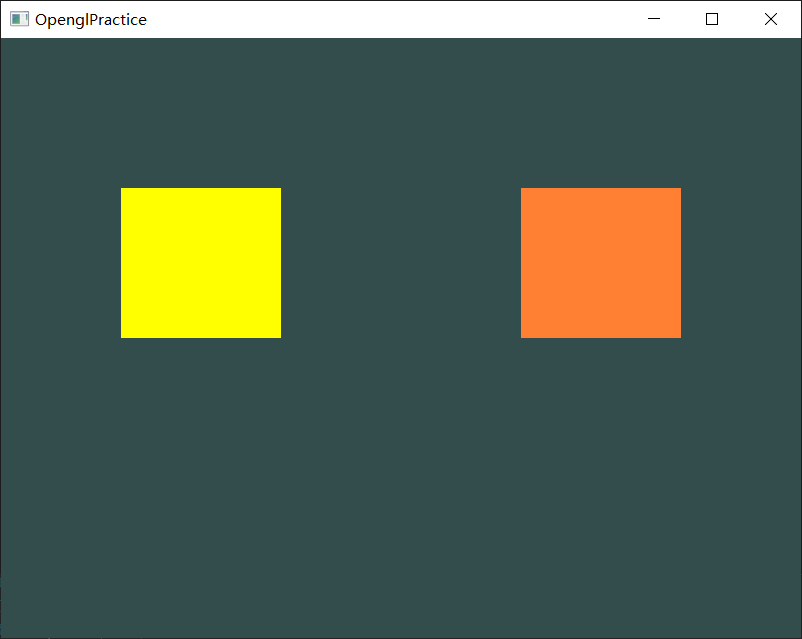
其中第一个VBO的vertices四个顶点坐标对应的是左边的矩形位置，第二个VBO的vertices四个顶点坐标对应的是右边的矩形位置。



2.5 在渲染循环中，绘制两个矩形。其中shaderProgram1激活之后颜色将被设置为橘黄色，然后再关联第一个VAO（左边位置的矩形对象），通过EBO的规则绘制矩形即可在屏幕左边绘制一个橘黄色矩形。同理可得右边的橙色矩形。

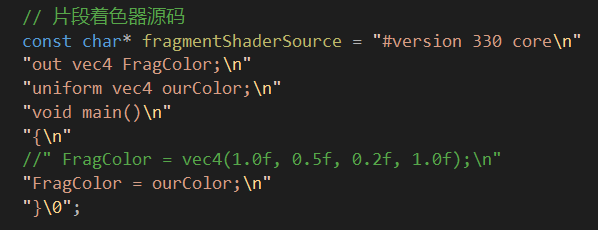


2.6 编译通过后运行程序，可得到两个颜色不同的矩形。

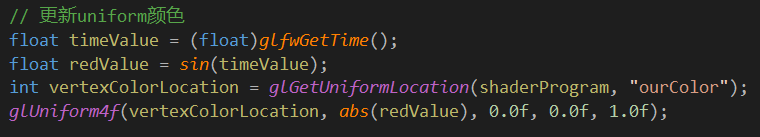


3 绘制完单一纯色的三角形，我们再玩一点更花的。我们来创建一个颜色可以根据时间流逝而变化的矩形。

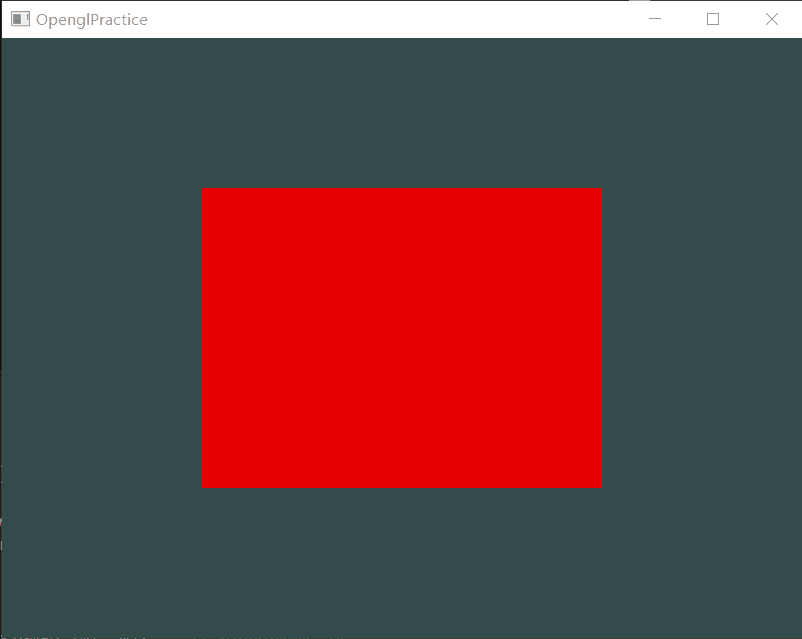
3.0 首先我们要修改一下片段着色器中的颜色。将之前定死的颜色用uniform vec4 ourColor来代替。这里的ourColor是一个在shader程序之外的由我们主程序来控制的vec4变量。



3.1 在渲染循环中我们获取当前的时间，将当前的时间作为变量转成正弦值的绝对值（0~1.0）之间。获取着色器程序中的ourColor颜色变量后，将颜色值赋给它。



3.2 编译运行程序得到一个颜色随时间变化的矩形。



程序源码：

