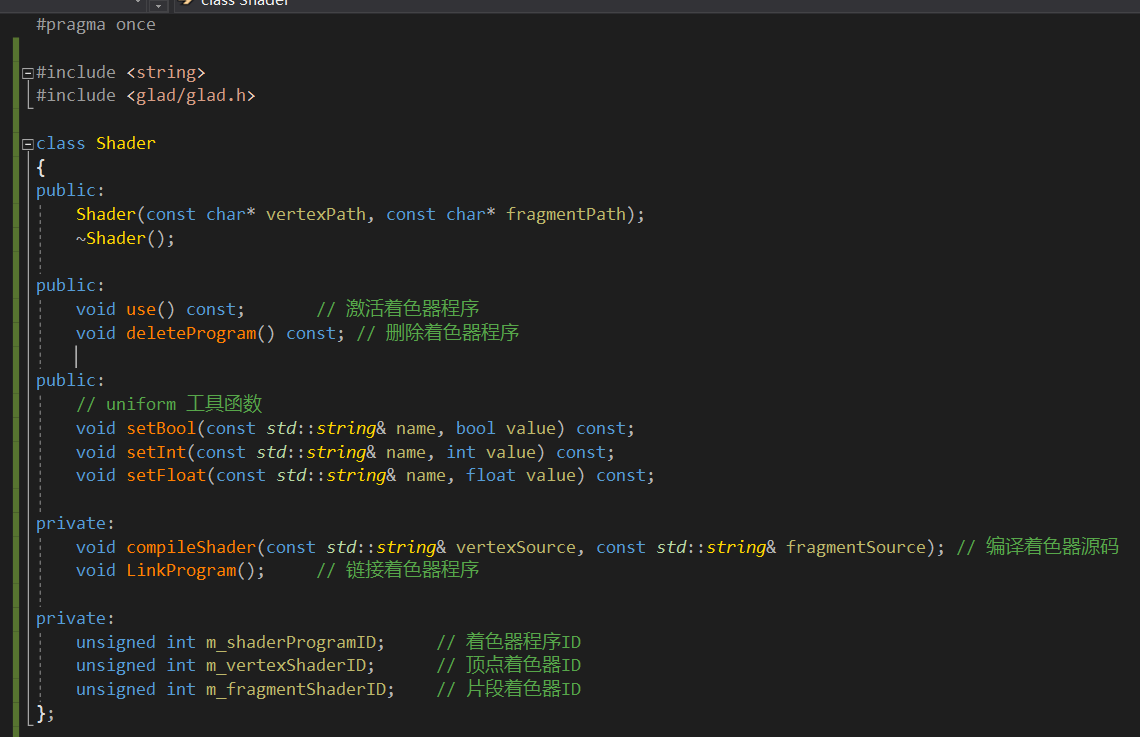
0 上一章我们讲到如何生成不同颜色的矩形，然后延伸了一些内容生成了一个颜色会随时间正弦变化的纯色矩形。本章我们将主要做两件事：

（1）将着色器程序封装成一个类。

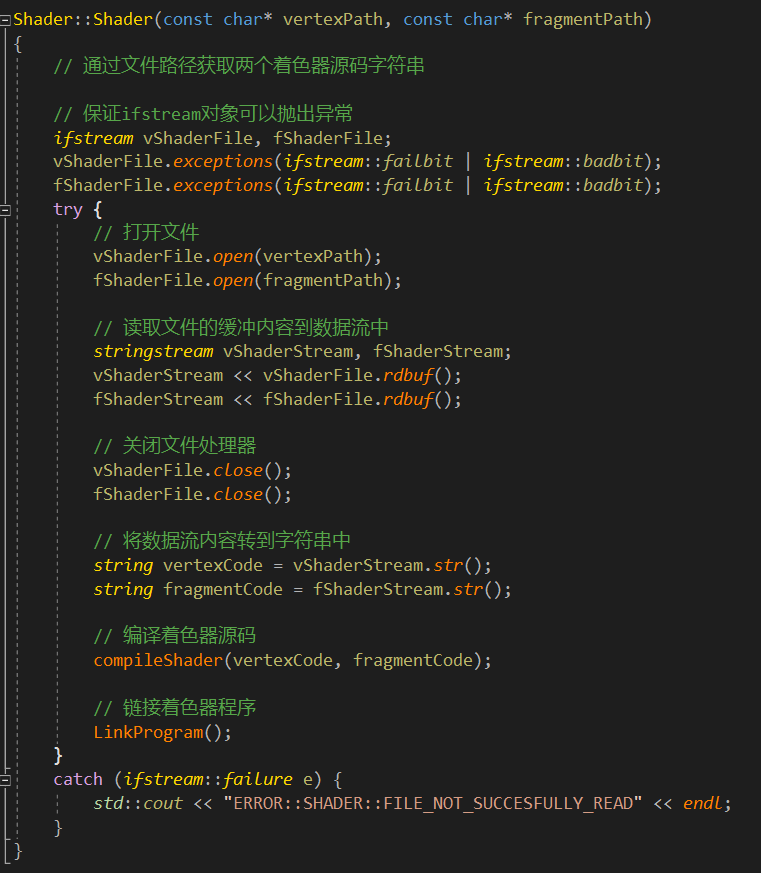
（2）通过在vertices添加矩形四个顶点对应的颜色数据生成一个彩色的矩形。

1 将着色器程序封装成一个类

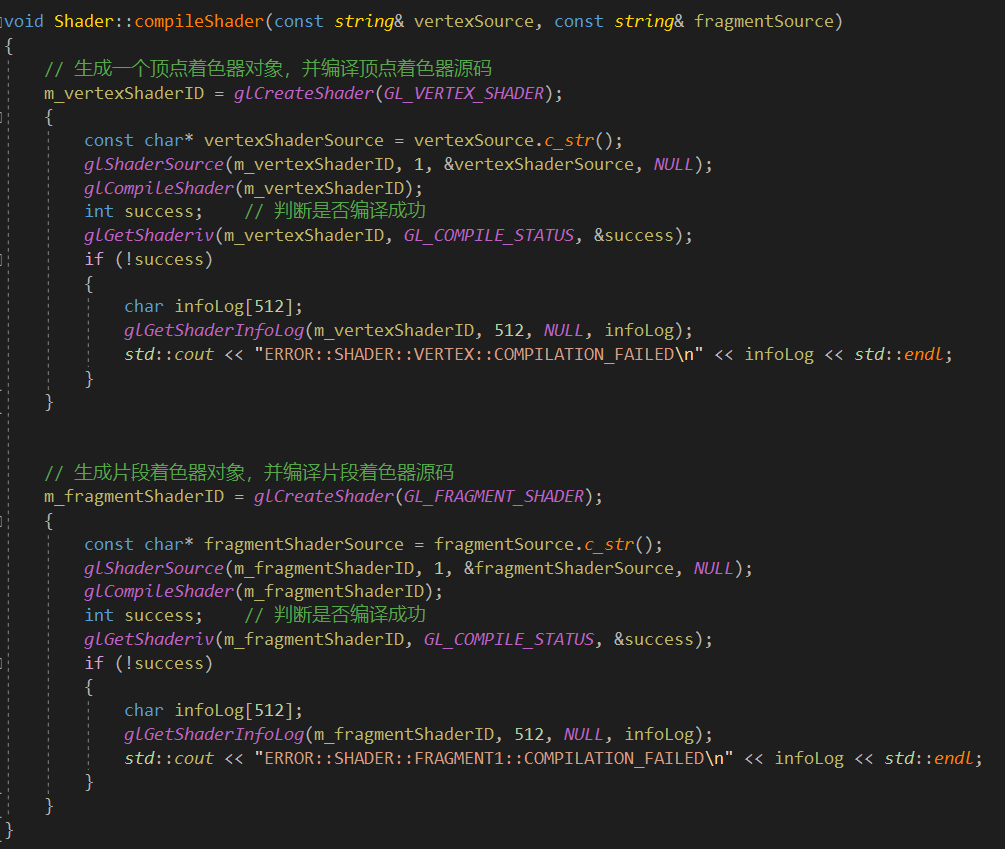
1.1 新建一个Shader头文件和cpp文件。将顶点着色器源码文件路径和片段着色器源码文件路径作为构造函数的参数传入到Shader中。



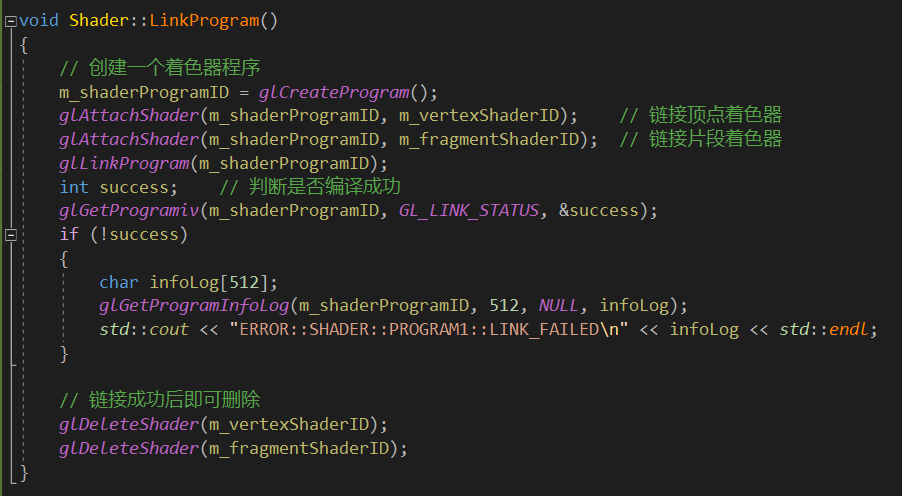
1.2 在Shader的构造函数中读取着色器源码文件内容，并将其编译。最后将其链接到着色器程序中。



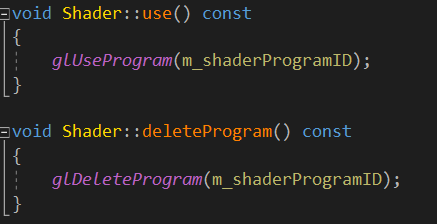
1.3 将之前main函数里的顶点着色器和片段着色器编译实现细节移到编译接口中。



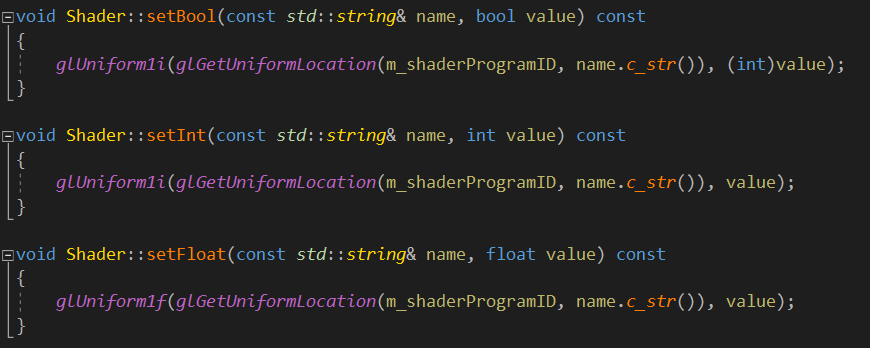
1.4 将main函数里的着色器程序链接实现细节移到链接函数中。



1.5 设置着色器激活函数和删除函数。

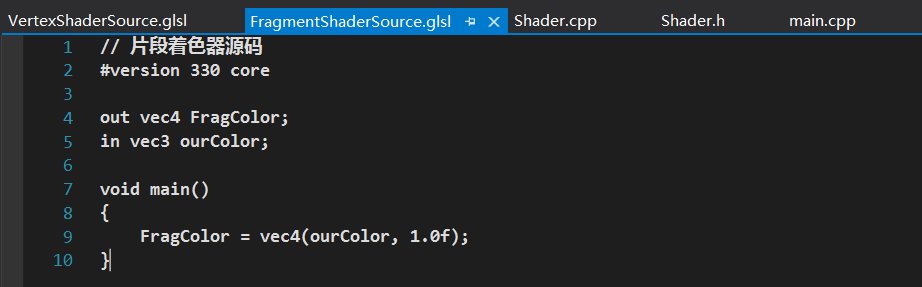


1.6 设置各种数据类型的着色器uniform参数的接口。

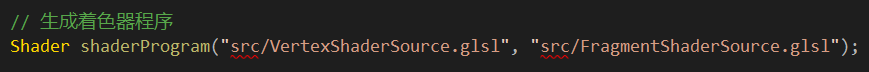


1.7 将顶点着色器源码和片段着色器源码分别存放在两个不同的GLSL后缀的文本文件中。

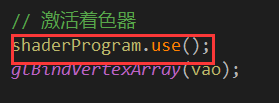




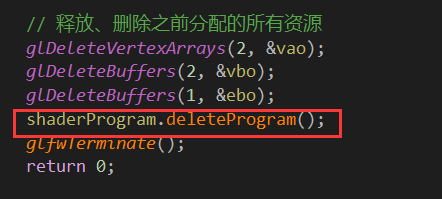
1.8 在main函数中将之前的shader流程都删了，改为直接使用Shader对象的方式。



1.9 在渲染循环中，每次绘制之前，使用Shader对象激活着色器。

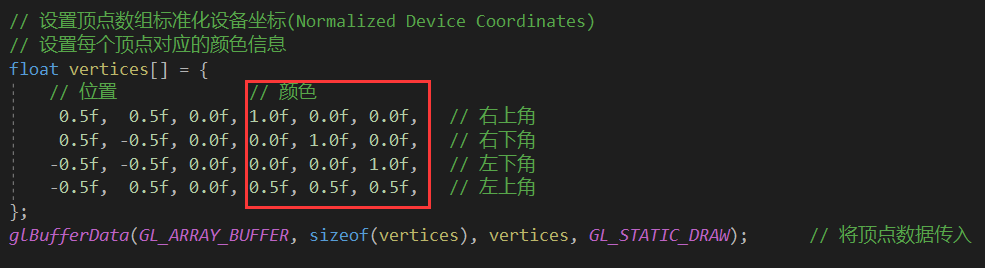


1.10 渲染完毕退出渲染循环后，需删除着色器程序分配的资源。

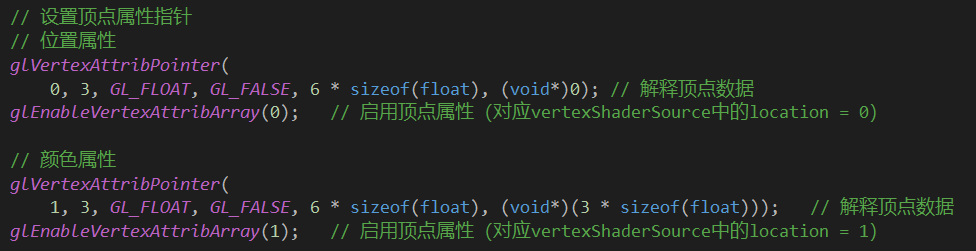


2. 封装完着色器程序之后，我们再实现一个花里胡哨的功能，即通过给vertices数组中添加4个矩形角点对应的颜色信息，来生成一个彩色的矩形。

2.1 在vertices中添加颜色信息。



2.2 分别解释vertices数组中左半部分数据是角点坐标信息，右半部分数据是角点颜色信息。



2.3 编译并运行程序，得到一个彩色的三角形。（设置了四个顶点的颜色，然后OpenGL在光栅化时会根据像素位置对颜色进行插值，才得到的这幅彩色的图象）

