0 上一章讲到如何生成一个彩色的矩形。这一章我们将主要引入纹理这个概念。并且编写三件关于纹理图像的例子：

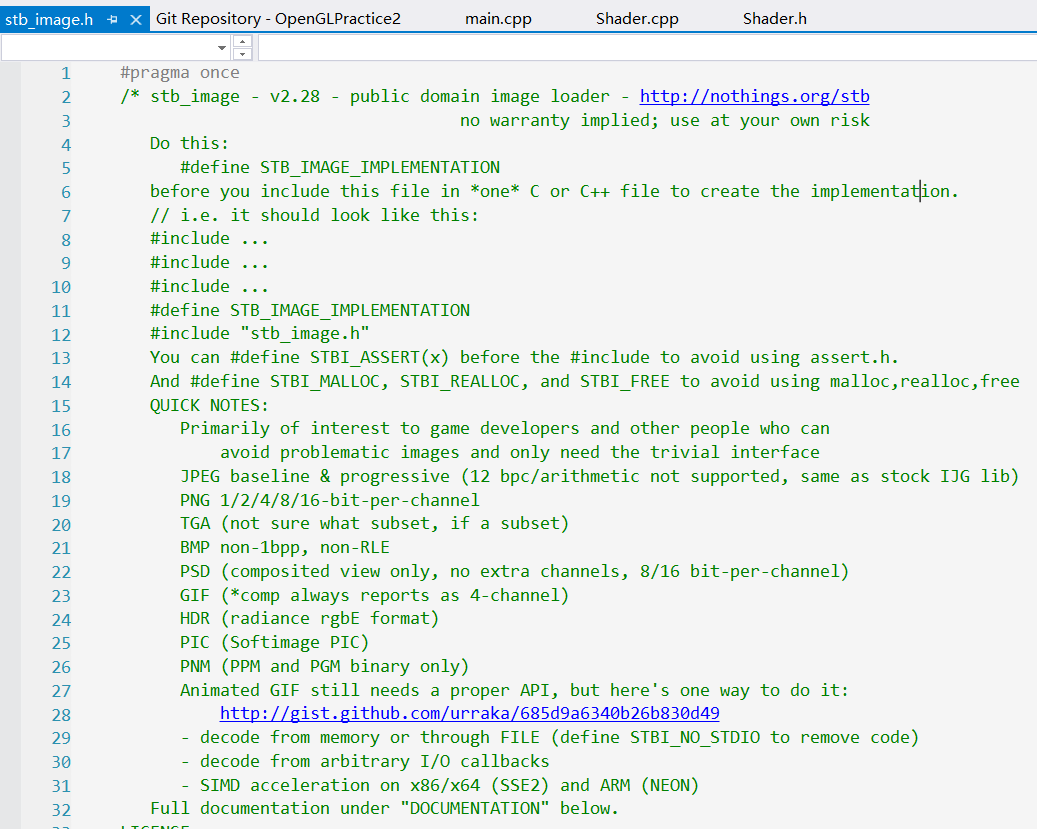
（1）设置单一纹理的矩形。

（2）设置多纹理混合的矩形。

（3）探索放大\缩小时的纹理过滤方式。

**1 设置单一纹理的矩形**

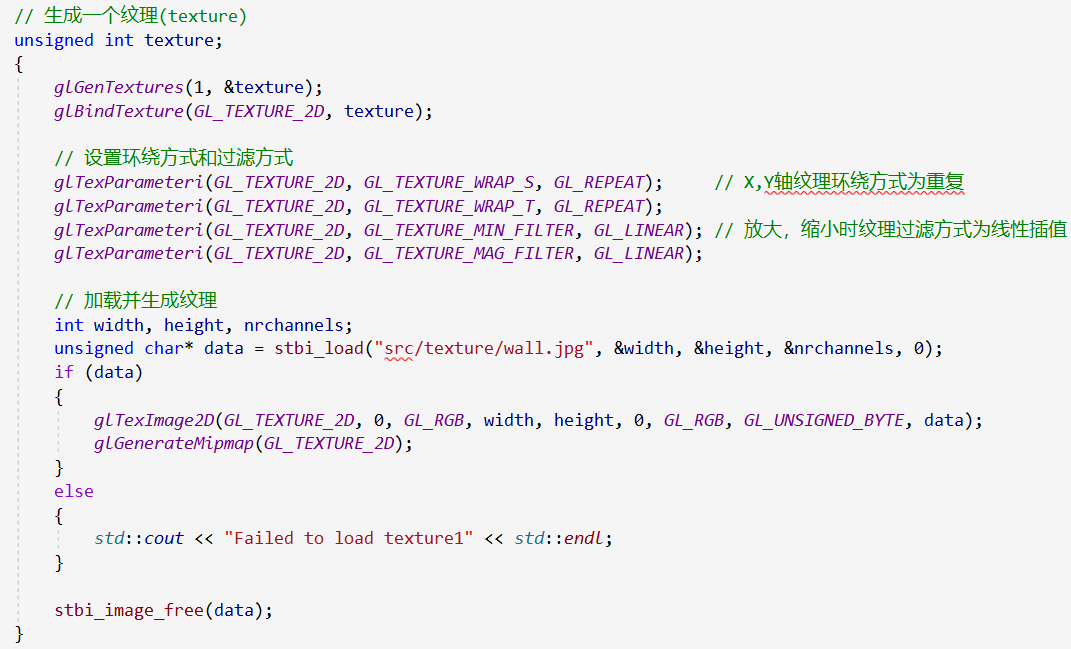
1.0 使用开源库stb\_image（https://github.com/nothings/stb/blob/master/stb\_image.h）。将stb\_image的头文件添加进我们的项目中。



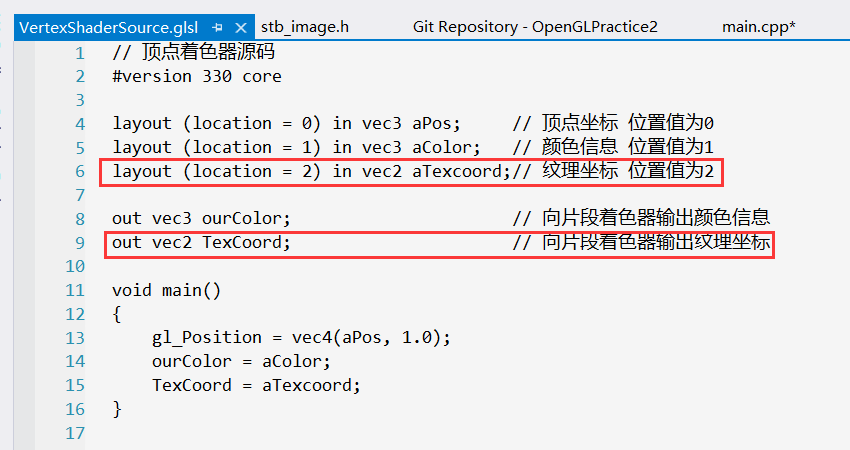
1.1 在main.cpp的顶部添加宏和stb\_image.h的包含。其中宏的作用是将stb\_image.h转为cpp来调用。



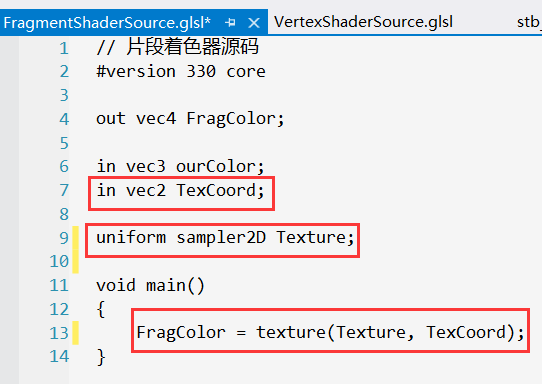
1.2 生成一个纹理对象，设置该纹理对象的环绕方式和过滤方式。使用stb\_image的接口加载纹理图片（一张墙壁的纹理图纸），并将纹理图片的信息附加到纹理对象上。（记得及时释放掉图片信息的资源）



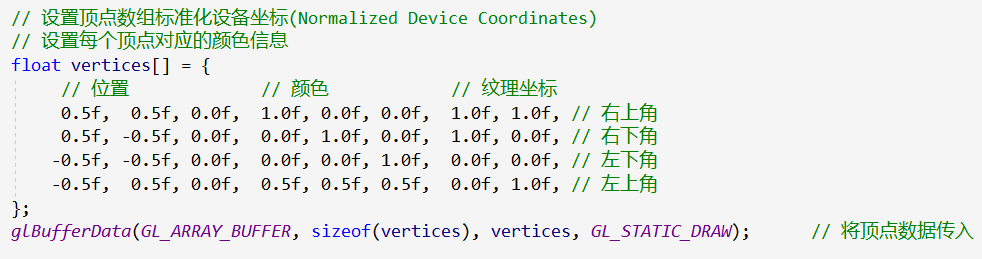
1.3 在顶点着色器源码中添加一个location为2的2维向量来表示纹理坐标信息。并将纹理坐标信息输出给片段着色器。



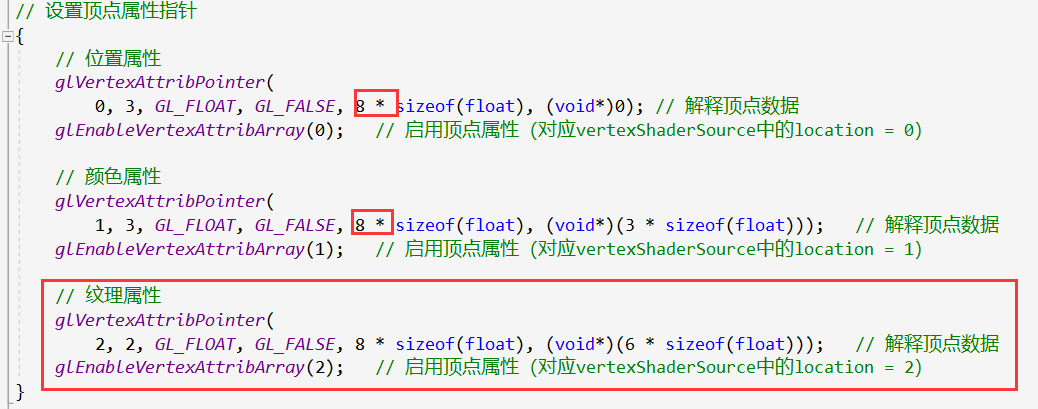
1.4 在片段着色器源码中我们通过传入的纹理坐标和使用uniform设置的采样器纹理变量共同生成一个纹理色彩传给FragColor。



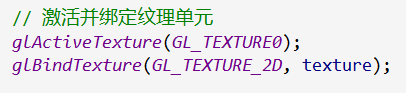
1.5 我们在main()里的vertices后面添加纹理坐标信息。纹理坐标信息的范围在0到1.0之间，坐标原点在左下角，如果超过1.0，则会对纹理图像进行特殊处理（后面会讲到）。



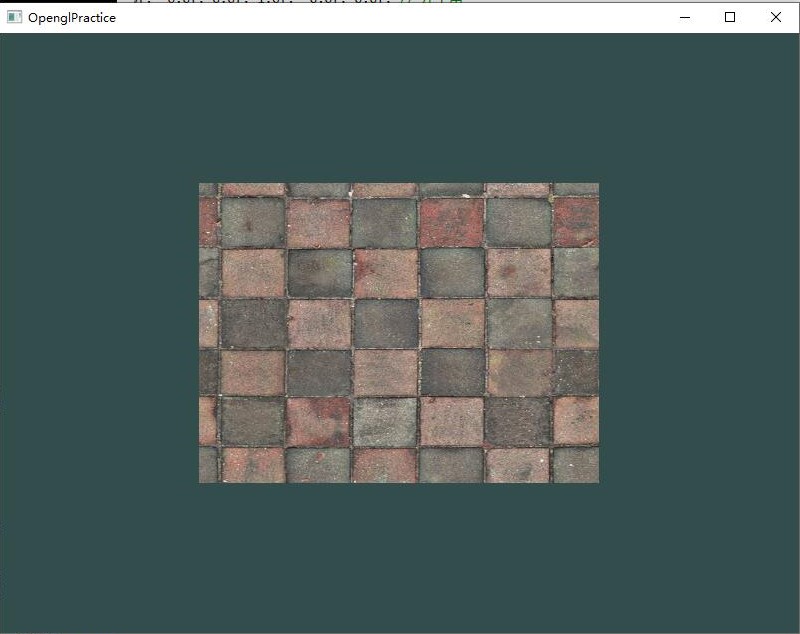
1.6 重新解释vertices数组中的数据排列的步长和偏移量。



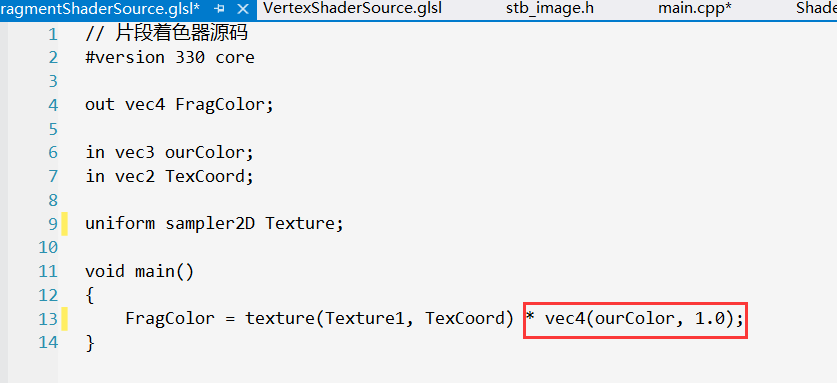
1.7 将纹理单元绑定到0号位。（没有特意的去将片段着色器中的OurTexture设置为0号位，因为此时只有一个纹理信息，OpenGL默认将其设置为了0号位）



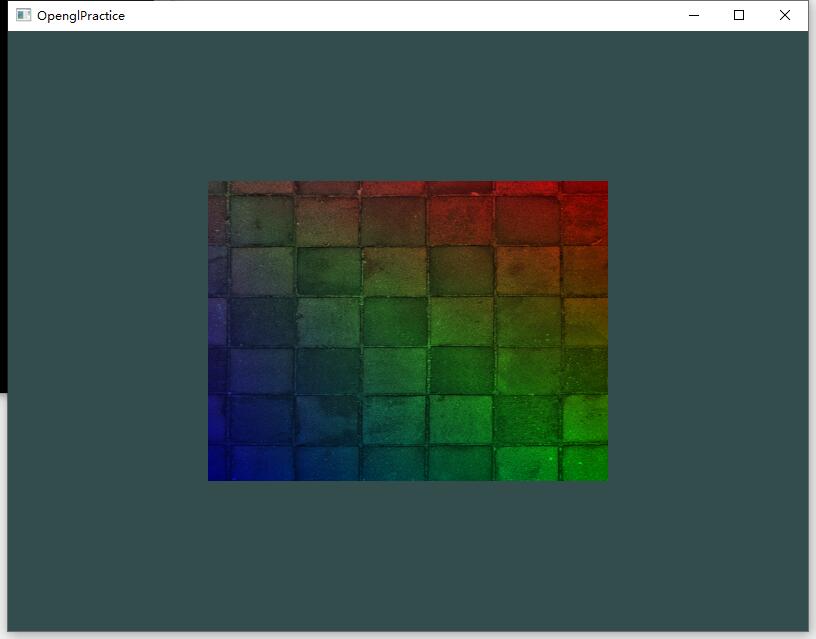
1.8 编译运行后得到一个带有墙壁纹理的矩形图像。



1.9 如果我们不满足单调的纹理，我们可以在纹理的基础上再添加其他的颜色。在片段着色器源码中，在纹理颜色的基础上再乘一个颜色值。

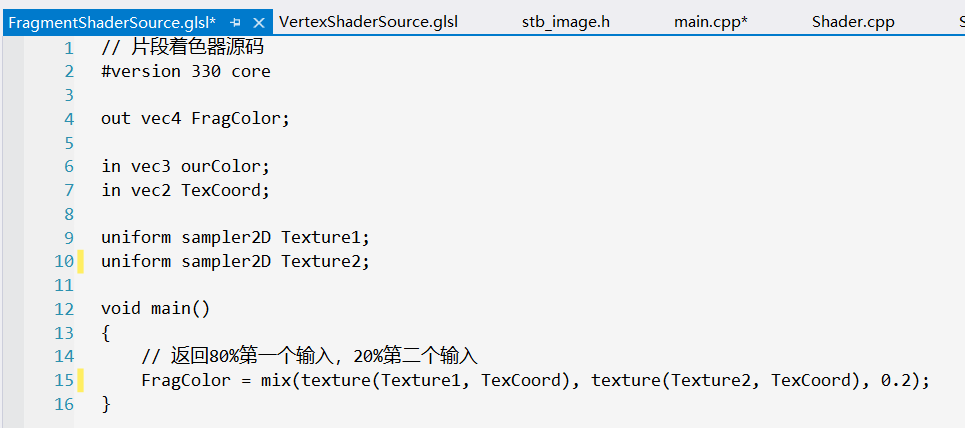


1.10 编译运行后可得到一个带有附加颜色的纹理矩形。



**2 设置多纹理混合的矩形**

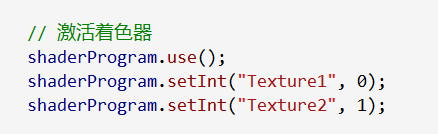
2.0 在片段着色器中添加两个纹理采样变量，将FragColor的颜色设置为由两个纹理图像融合的颜色。（其中0.2这个融合比例值表示纹理1的颜色占用80%，纹理2的颜色占用20%）



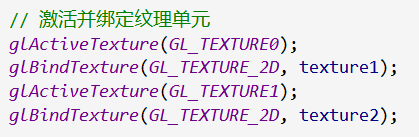
2.1 在main()的第一纹理对象后面再添加第二个纹理对象。将第二张图片（一个贱贱的笑脸）加载到第二个纹理对象中。



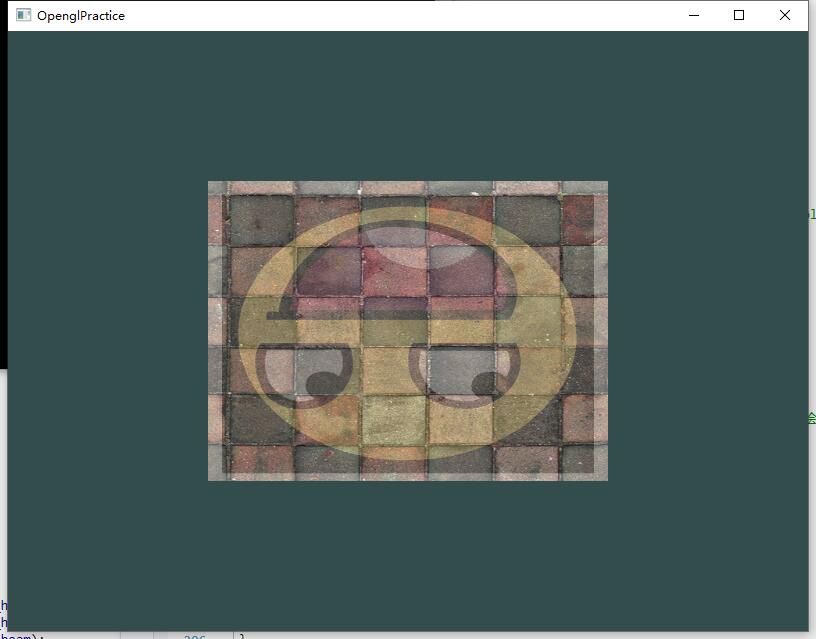
2.2 由于此时程序中有两个纹理对象需要设置，所以不能跟之前一样让OpenGL自己默认设置片段着色器中纹理采样变量的位置值了。需要手动地对纹理采用变量设置其对应的位置值。



2.3 将两个纹理对应绑定到纹理采用位置中。

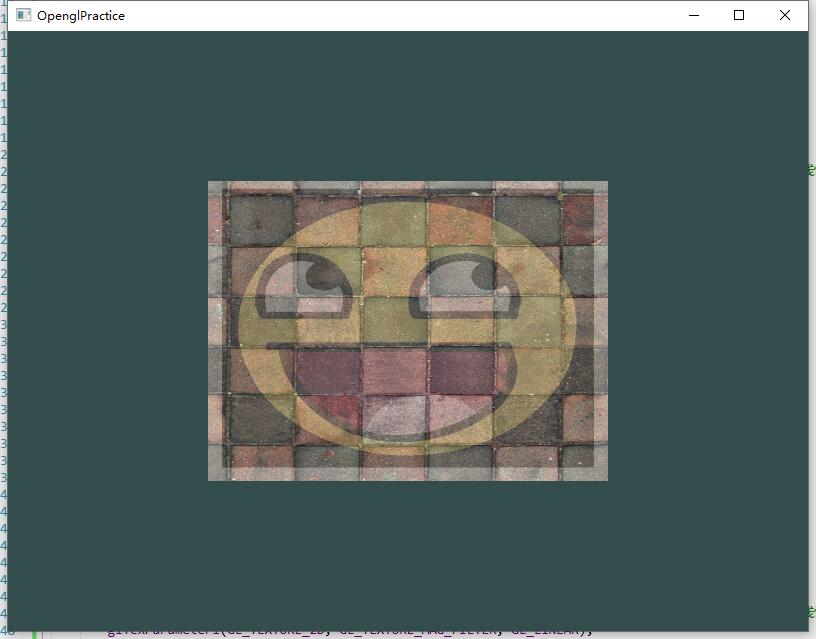


2.4 编译运行后可得到一个倒转的笑脸融合墙壁纹理的矩形。



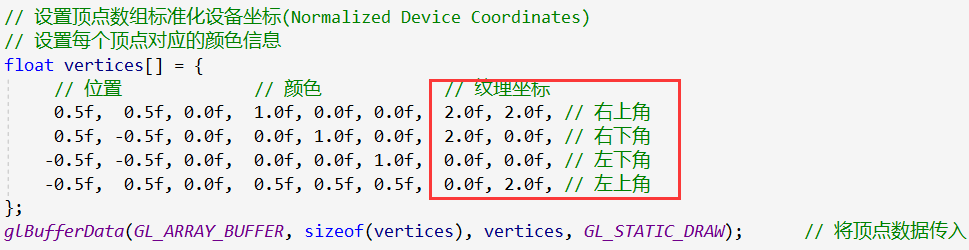
2.5 由于stb\_image传入的图片其y轴与OpenGL的y轴正好相反，所以使用stb\_image的接口反转图片的y轴后，得到正确方向的纹理图像。



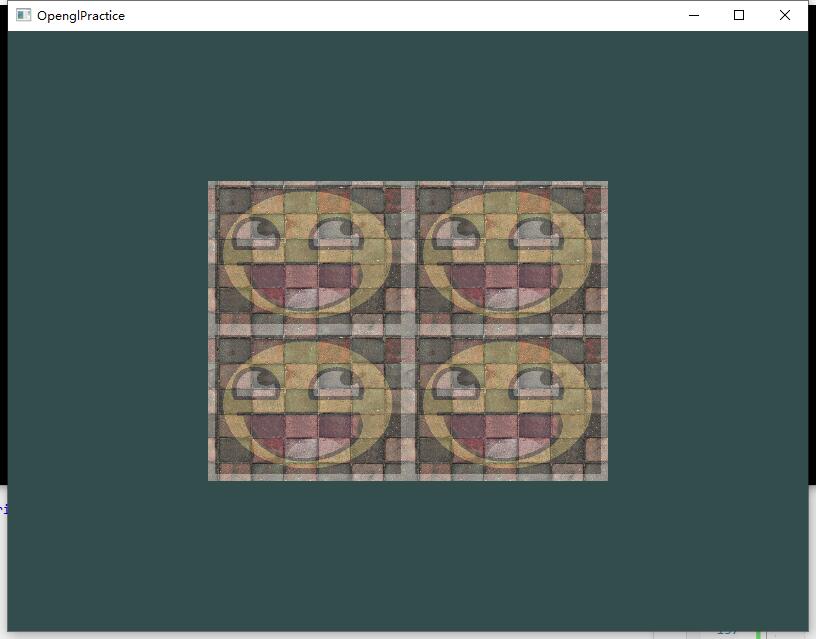


**3 探索放大\缩小时的纹理过滤方式**

3.0 之前在vertices中设置的纹理坐标范围控制在了0到1.0之间。现在我们将纹理坐标值扩大到2.0。

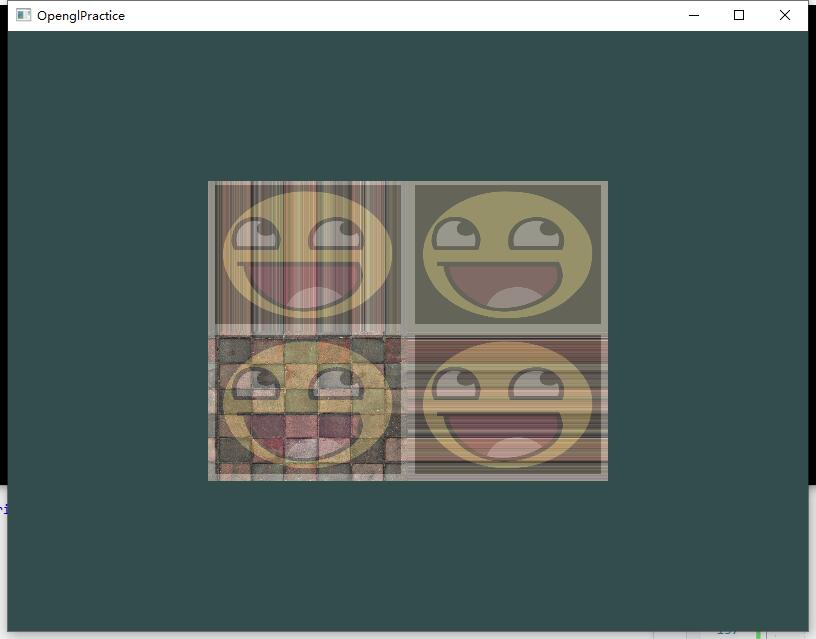


3.1 编译运行程序，此时会出现矩形的面积大于纹理图像的情况。且纹理重复的方式为”重复图像”。

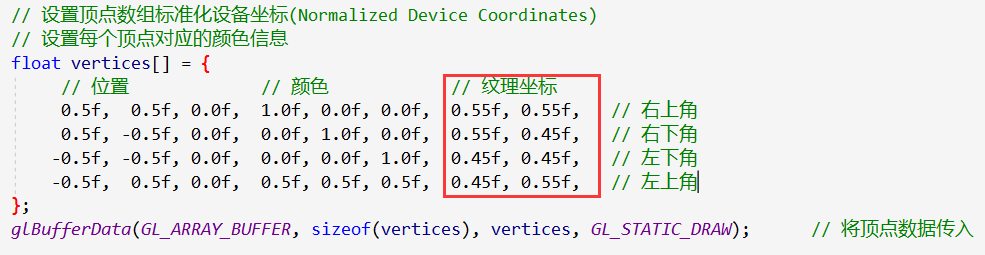


3.2 我们可以将第一个纹理对象的纹理环绕方式改为延伸像素边缘。此时得到的图像将不再是两个纹理均为重复，而是第一个纹理为边缘延伸，第二个纹理为重复。

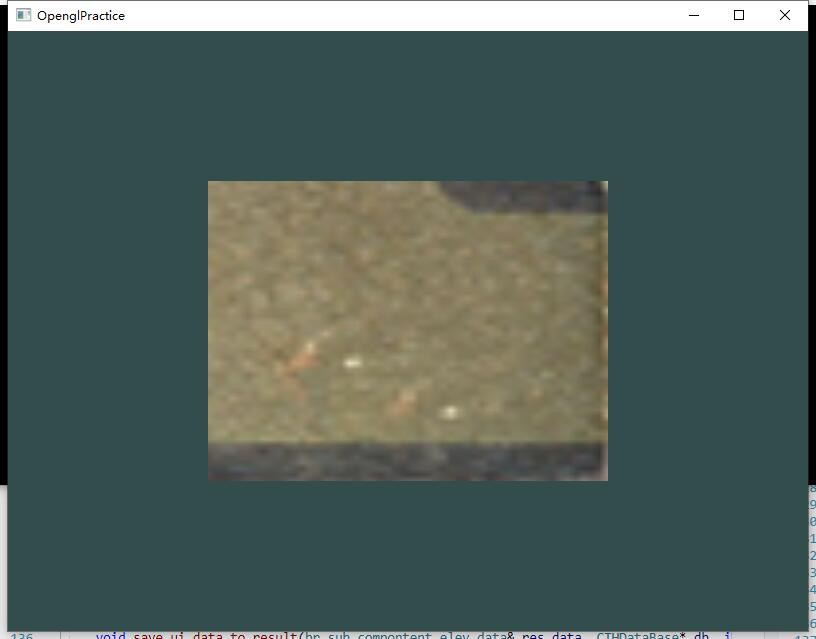




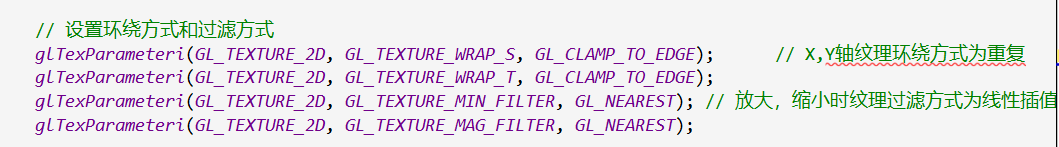
3.3 刚刚我们将纹理坐标扩大到了2.0，现在我们再试试看将纹理坐标缩小到0.45-0.55看看会是什么效果。



3.4 可以看到在矩形范围内，我们只加载了面积非常小的一块纹理图像被加载了。（此时纹理过滤方式为线性插值）



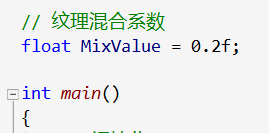
3.5 将纹理过滤方式从线性插值改为取最近点。图像的像素块会更明显。



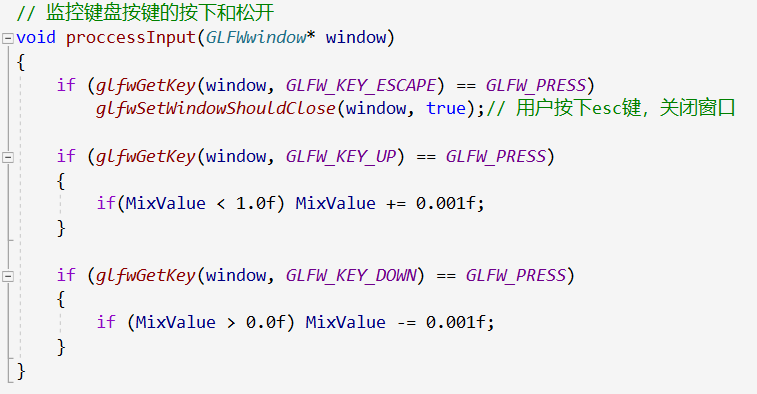


**4 最后我们实现一个通过键盘上下两个键，动态调整两个纹理图像混合比例的功能。**

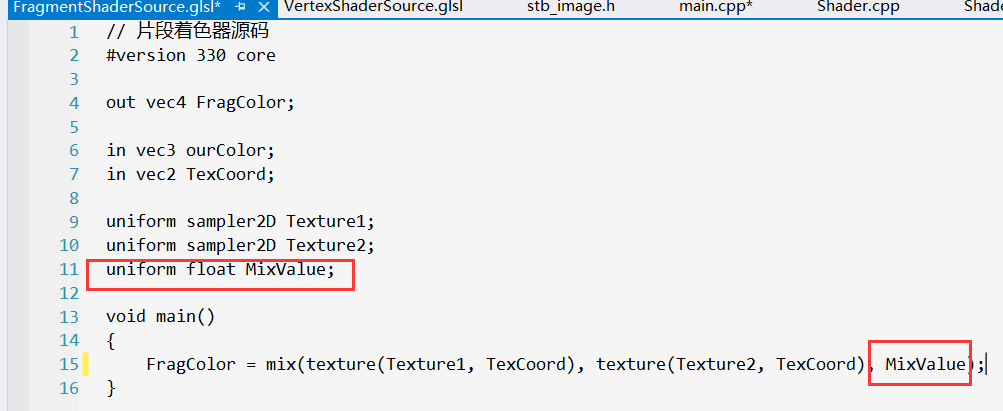
4.0首先在main()前设置一个全局的控制混合比例的参数。



4.1 然后在键盘触发响应的回调函数中设置混合比例参数的调整。



4.2 在片段着色器中添加一个纹理混合比例参数。



4.3 在渲染循环中，动态的设置纹理混合比例参数。



4.4 编译运行程序，可以得到一个可以通过上下键控制纹理混合比例的矩形图像。

