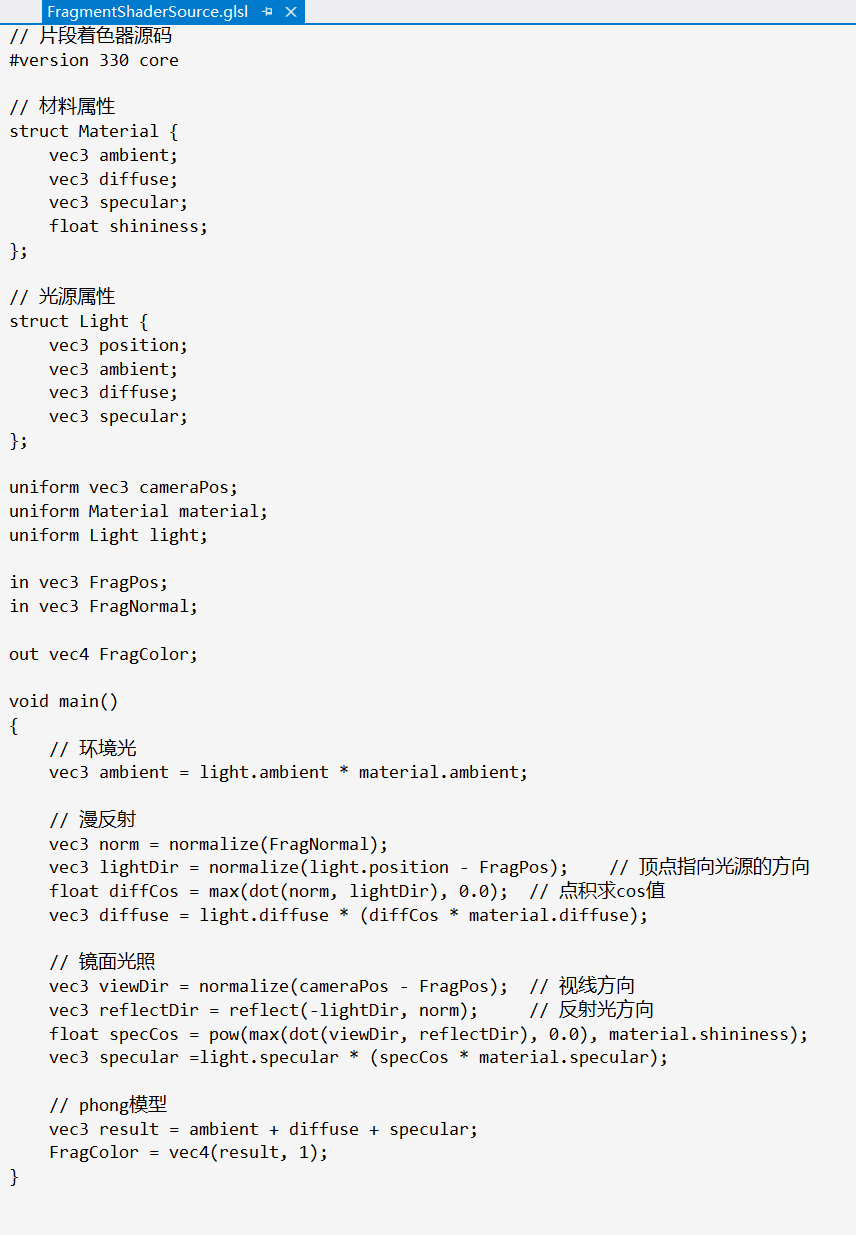
**0 上一期我们创建了一个基于Phong模型的基础光照模型。本期的内容我们将完善该光照模型。**

（1）使用物体的材质贴图，更真实的还原物体的光照。

（2）使用放射光贴图模拟黑客帝国的矩阵特效。

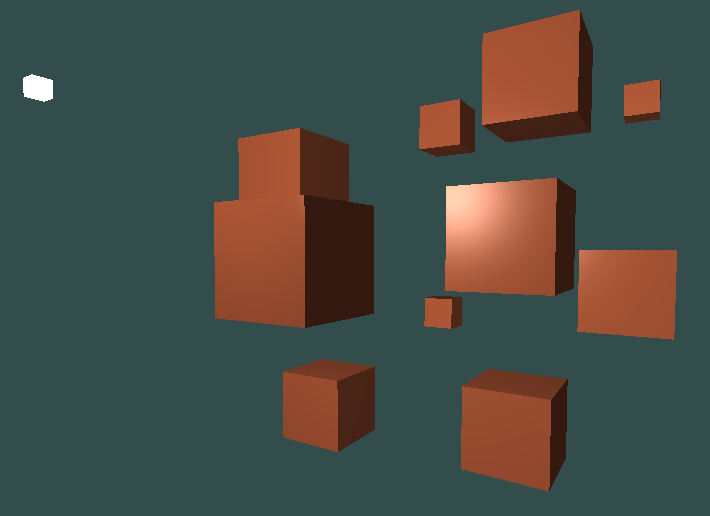
**1 使用物体的材质贴图，更真实的还原物体的光照。**

1.0 我们将片段着色器中的材料属性和光源属性所涉及到的光照信息抽象成数据结构。并使用抽象出来的数据结构来表达Phong模型的光照。

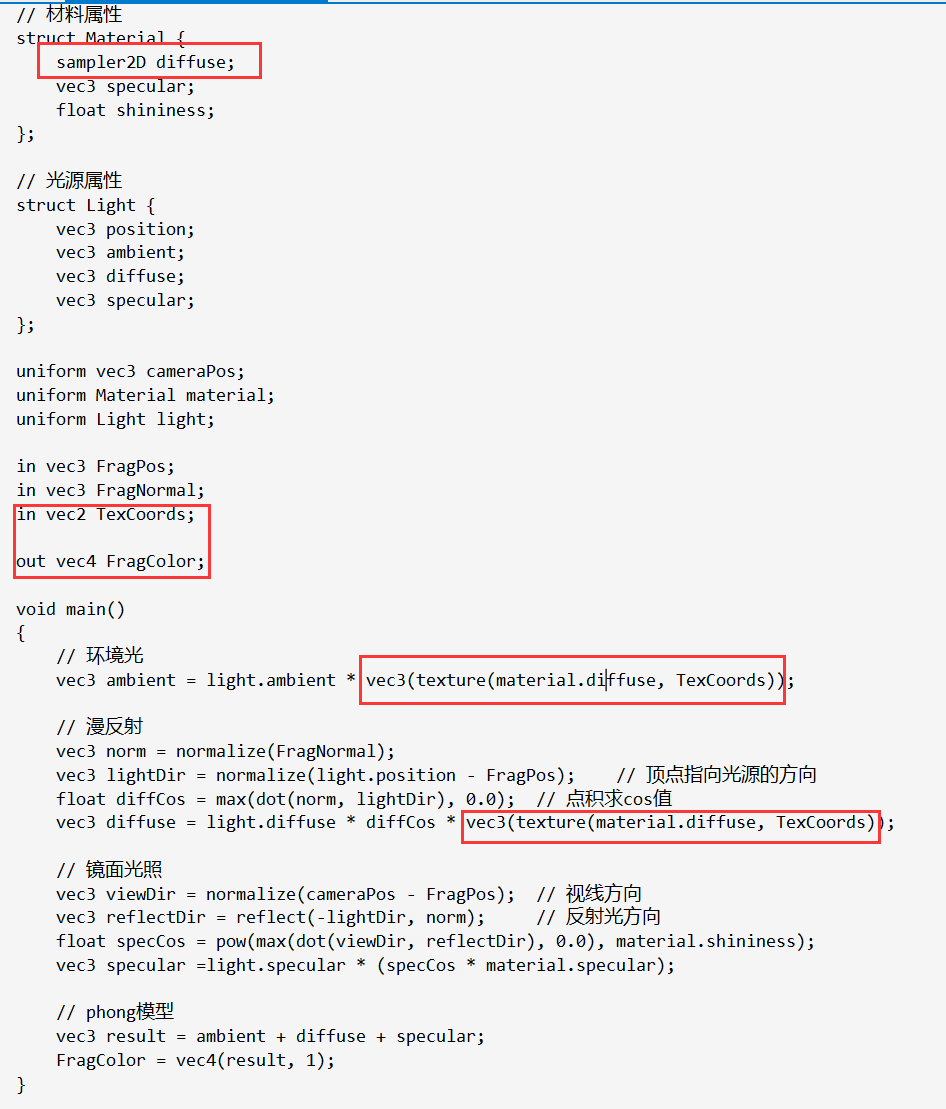


1.1 在渲染循环中，根据材料属性和光源属性的数据结构对其相应的值进行设置。编译运行后我们可以看到Phong模型光照渲染正常。

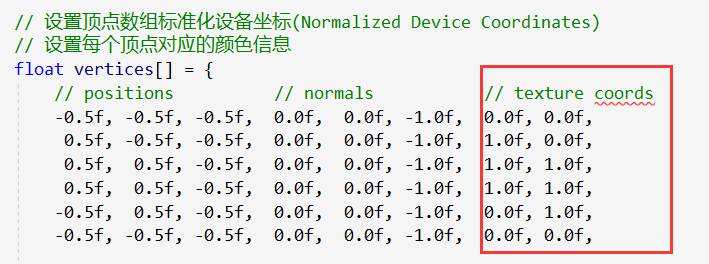


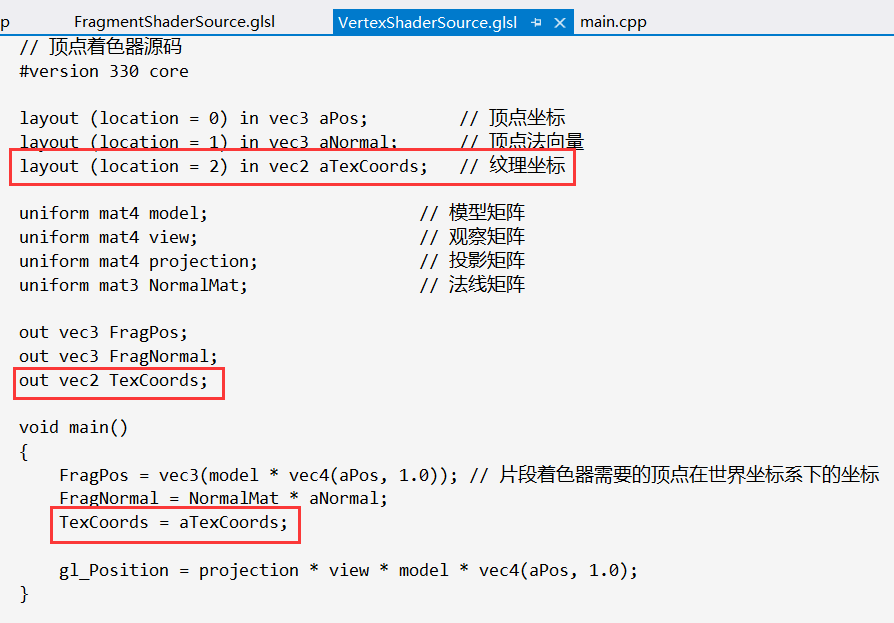


1.2 因为我们接下来要使用材质贴图来设置对象的材质属性。所以我们需要在片段着色器中将Material的漫反射定义为采样值来接收材质贴图中TexCoords坐标值对应的颜色信息。

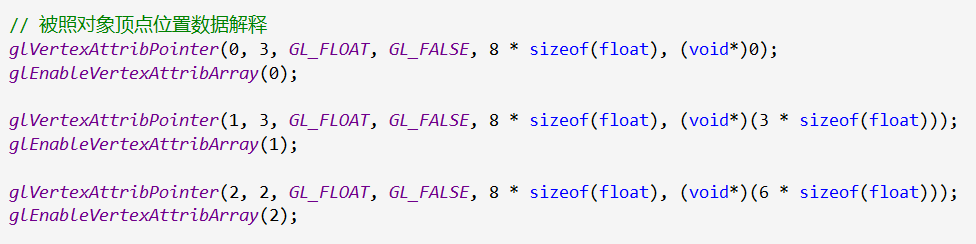


1.3 我们在main()中的vertices数组中设置好材质贴图的坐标值，并在顶点着色器中设置好材质贴图的坐标值信息。

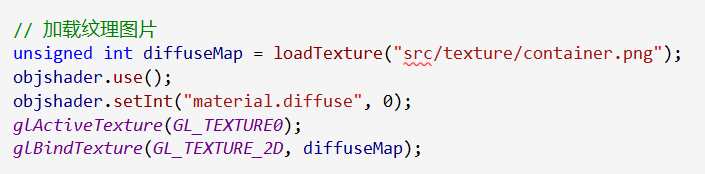


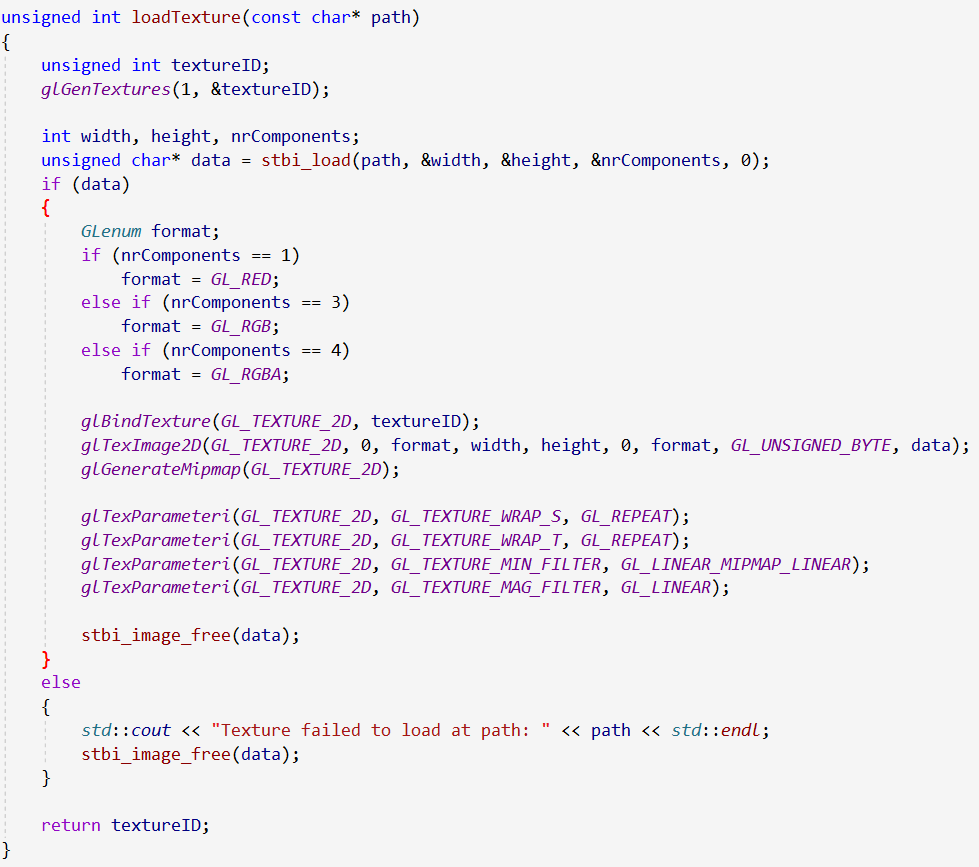


1.4 新添加了材质贴图的坐标信息，需要重新解释数组的含义。

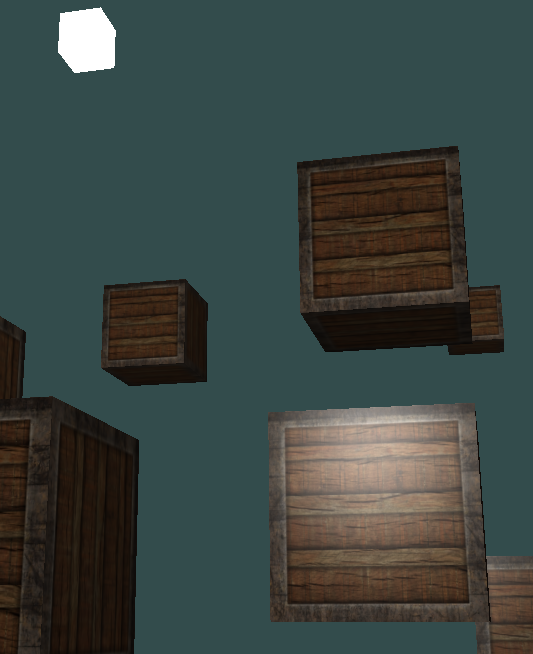


1.5 加载材质图片并将纹理ID绑定到第0个纹理槽中。

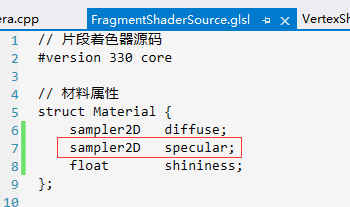




1.6 编译运行后，可得到贴好箱子材质纹理的多个矩形，观察光照情况可发现箱子的金属边框和中间的木质挡板都在反射着相同强度的光照。这显然是不太真实的。

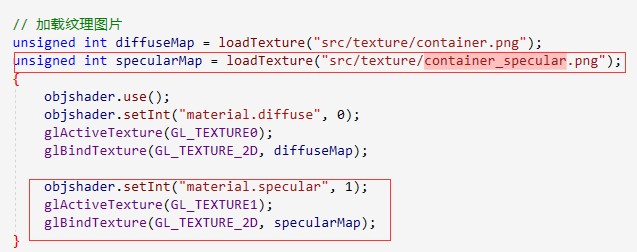


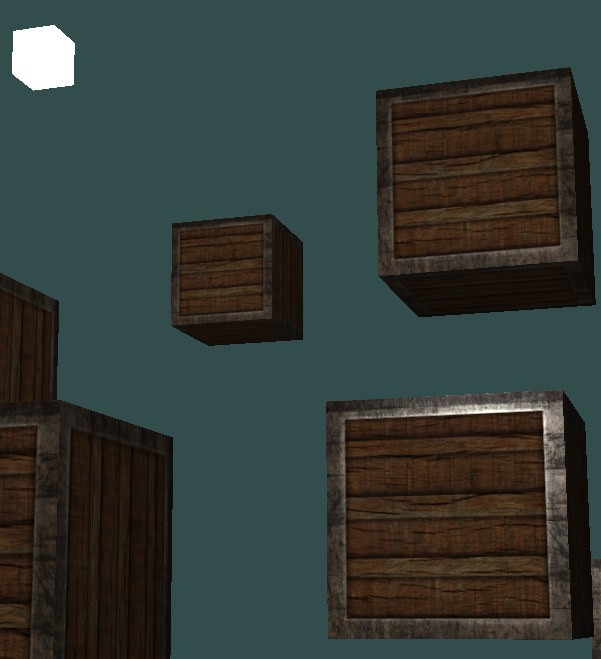
1.7 我们在片段着色器中将材料属性的镜面光照参数也设置成采样类型，这样我们便可以通过高光的材质贴图来控制物体材料的镜面反射属性。





1.8 使用同样的方法加载镜面光照的贴图，并将其纹理ID设置到第1个纹理槽中。

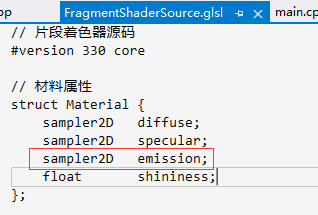


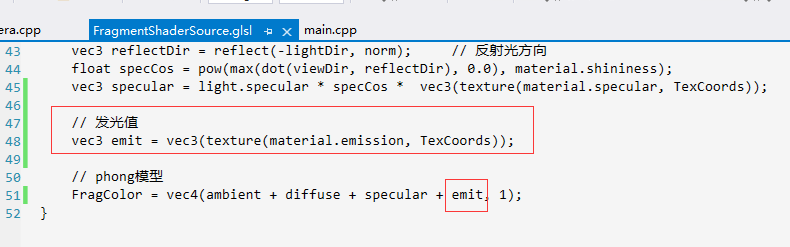
1.9 编译运行程序后可看到此时箱子的光照变得更加真实了。中间的木质挡板几乎不怎么反光，金属条形边框却泛着金属的光泽。

**2 使用放射光贴图模拟黑客帝国的矩阵特效。**

2.0 在游戏中，通常会有一种宝石发光的特效。类似于下面机器人的眼睛。

2.1 我们通过使用放射光贴图来设置发光值。在材质属性中添加emission发光值属性，并将放射光贴图的颜色属性关联到发光值中。





2.2 将类似黑客帝国中的数字矩阵放射光贴图加载到第2个纹理槽中。



2.3 编译运行程序我们将得到一个黑客帝国数字矩阵风格的箱子！

