1. **Shader 基本结构**

Shader “Custom/ShaderName”

{

Properties

{

....

}

SubShader

{

....

Pass

{

CGPROGRAM

....

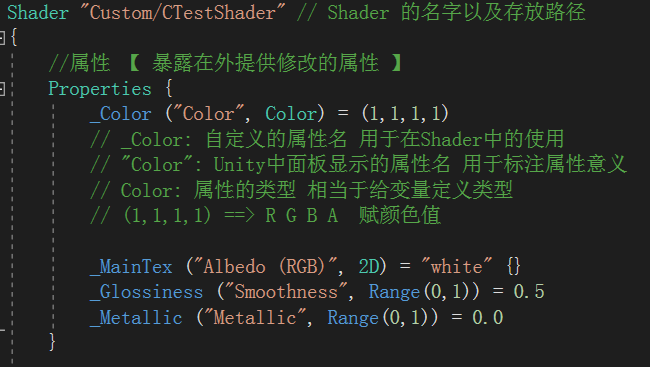
ENDCG

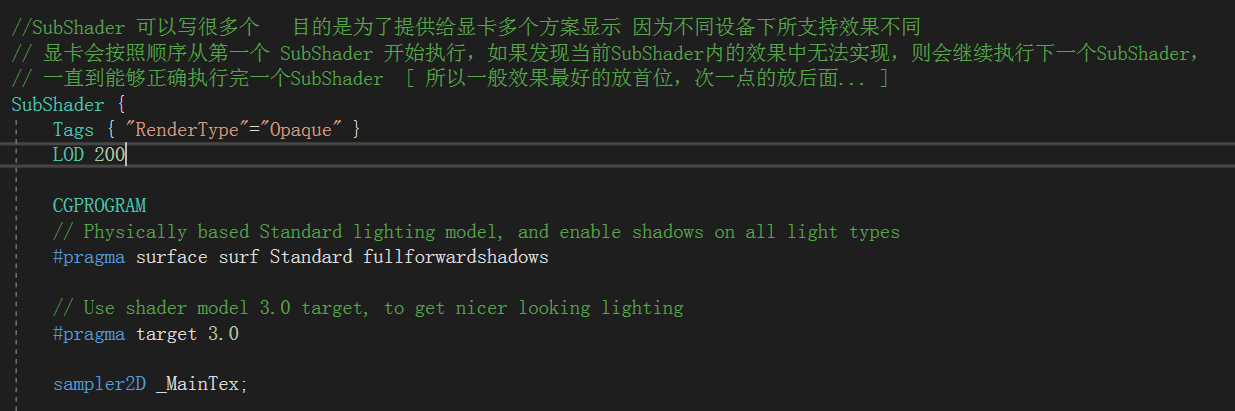
}

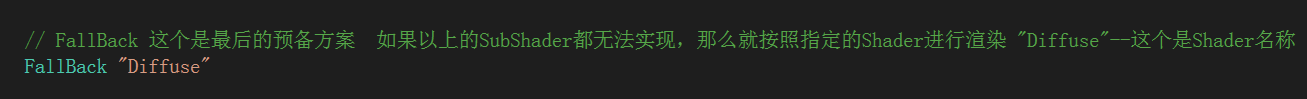
}

FallBack “Diffuse”

}



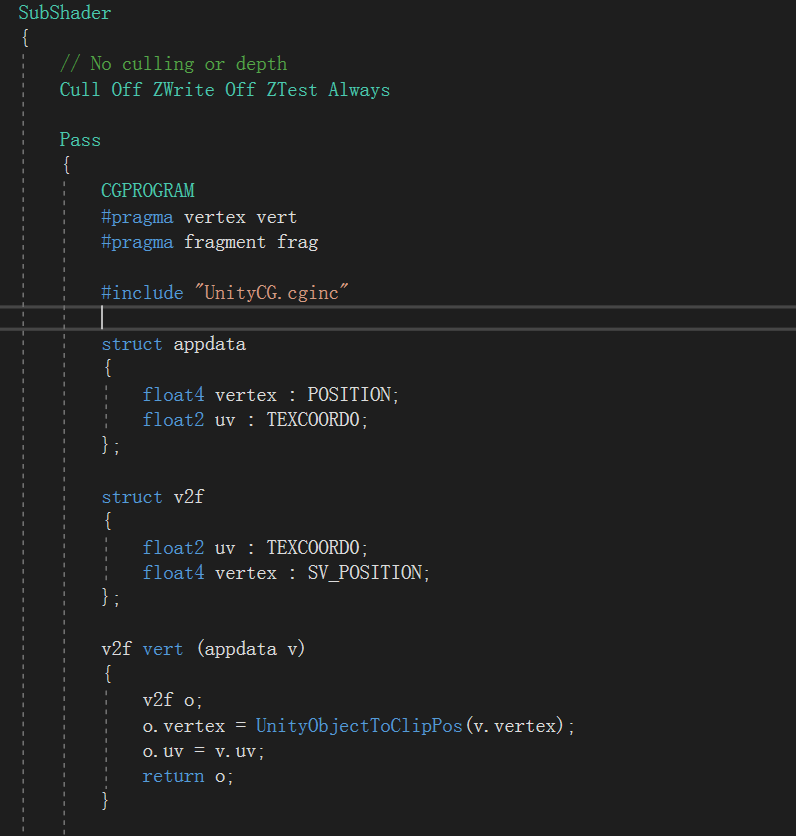




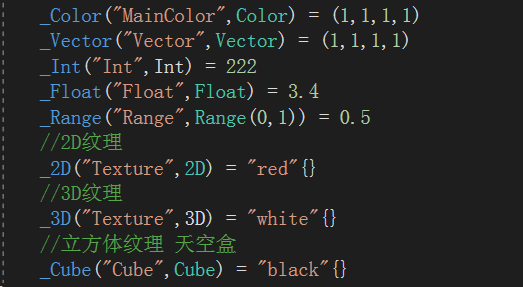
\*\* 这里说一点 在Unity中的标准着色器 Standard Surface Shader 是没有 pass 块的

一般来说 SubShader 中至少有一个 pass 块

Pass 块类似于方法 用来编写shader代码 这样

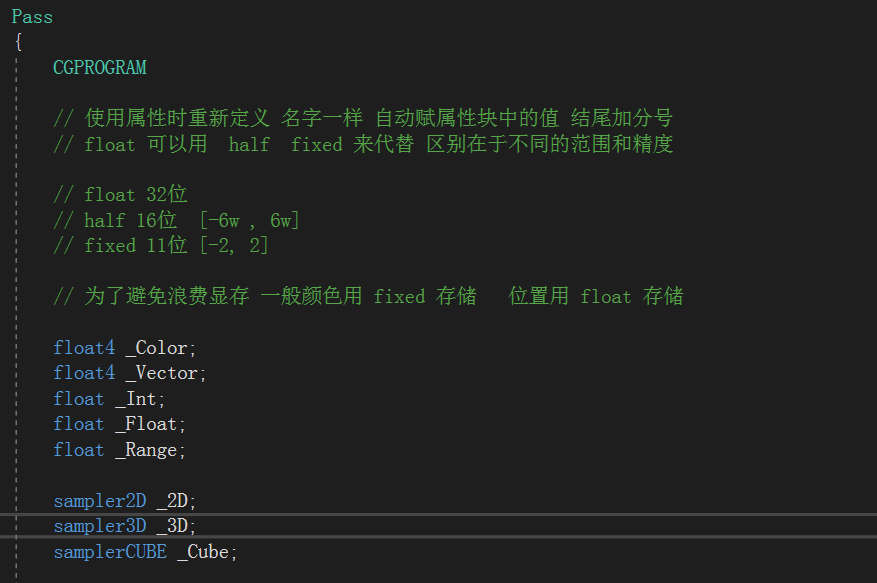


1. **Properties 属性块有哪些属性？**

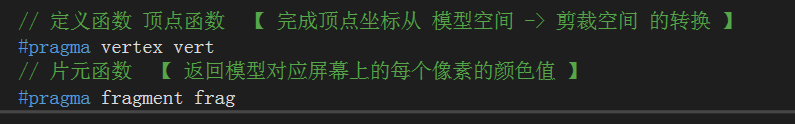


1. **SubShader CG 编码**

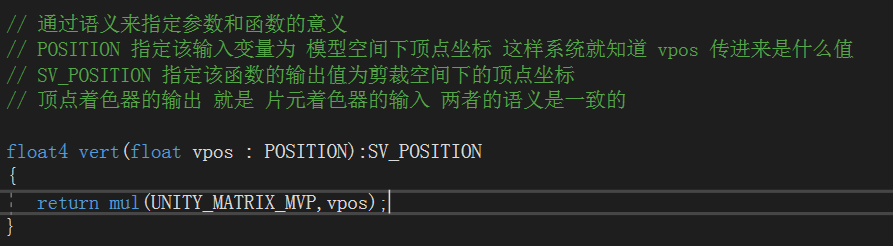
在属性块中定义的属性，在使用的时候需要在 pass 块中重新定义，但是不用重新赋值



定义函数，顶点着色器和片元着色器...



实现：



Unity 常用语义：

1. 从应用阶段传递模型数据 -----> 顶点着色器

POSITION 模型空间的顶点坐标 float4

NORMAL 顶点法线 float3

TANGENT 顶点切线 float3

TEXCOORD\_ 顶点的纹理坐标 \_ 这里填写索引 如 TEXCOORD0 表示第一组纹理坐标

float2/float4

COLOR 顶点颜色 fixed4/float4

1. 从顶点着色器 ------> 片元着色器

SV\_POSITION 剪裁空间的顶点坐标 必须存在

COLOR0 通常用于修饰第一组顶点颜色 不是必须的

COLOR1 通常用于修饰第二组顶点颜色 不是必须的

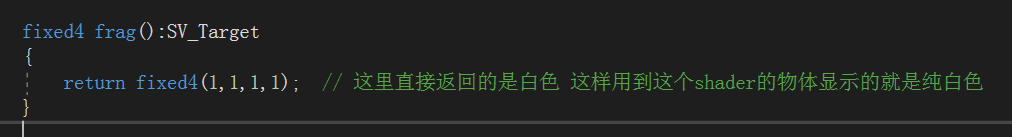
TEXCOORD0 - TEXCOORD7 通常用于修饰纹理坐标 不是必须的

\*\* 除了 SV\_POSITION 之外，其他的语义基本没有明确的需求，可以存储任意数据；

所以通常我们用 TEXCOORD0 来修饰自定义的变量

1. 片元着色器 --------> Unity

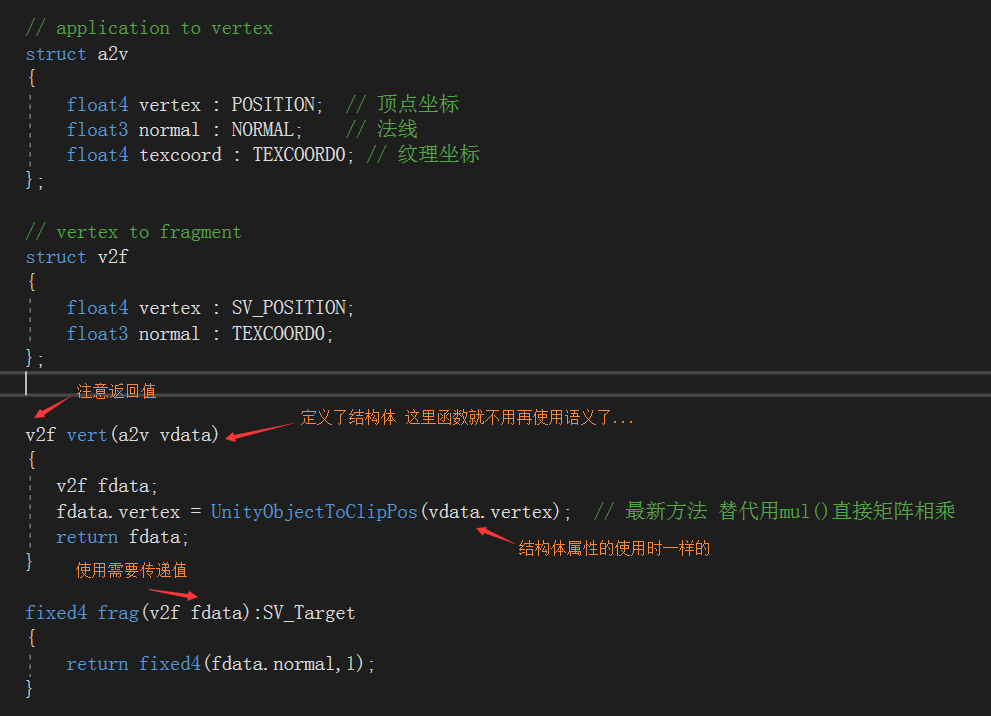
SV\_Target 在片元函数后添加语义 表示将该输出数据传递给 render\_target ( 渲染目标 )



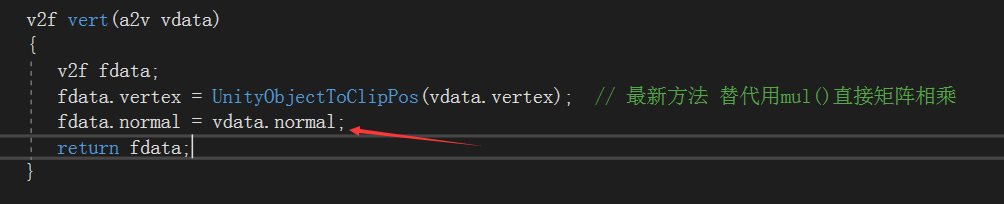
1. **Struct 结构体**

一般使用结构体来封装各种属性 方便在不同着色器之间的属性传递使用...

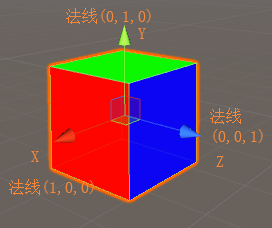
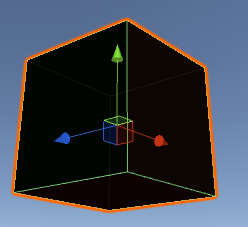
结构体定义大致相同 注意添加每个属性的语义 使用方法也基本类似



上面忘记把法线向量的值赋值给v2f当中了...



这样最后得出来的颜色值是根据这个物体的法线向量来的....

很明显每个面对应的颜色就是这个面上的点对应的法线向量

另外三个面全是黑的....

1. **光照模型**

.................... **漫反射 ( 兰伯特光照模型 )** ....................

公式： **Lum = C \* max(0, cos<L, N>)**

C：到达此点光线的强度和颜色

N：法线

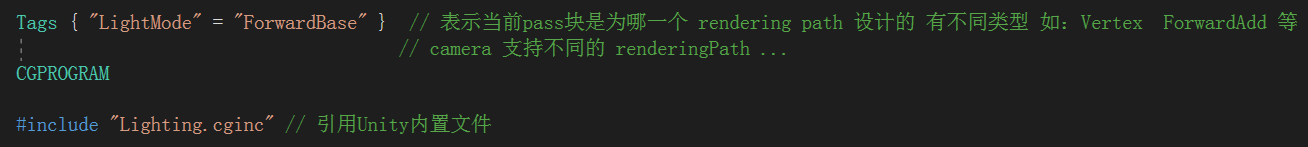
L：单位长度（标准化）的入射光向量

缺陷： 物体背面未受光部分会全黑，导致背面看不见...

\_LightColor0 : 第一个平行光光源

\_WorldSpaceLightPos0 : 第一个平行光光源位置

使用光源需要定义lightMode 和 引用 Lighting.cginc



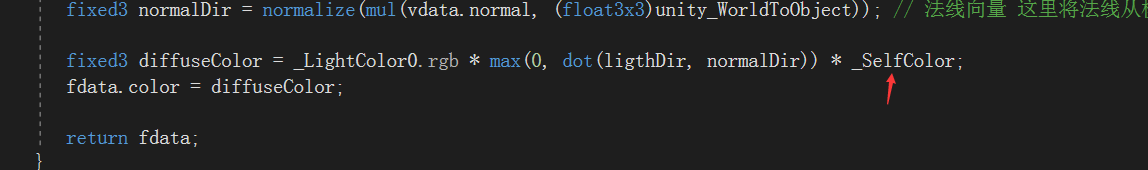


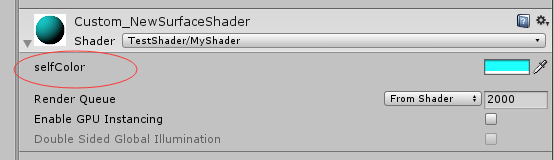
放在顶点函数: 逐顶点光照 根据每个顶点进行一次处理

放在片元函数: 逐片元光照 根据每个片元，也就是像素进行一次处理

因此逐片元光照会比逐顶点光照计算次数多，效率会比逐顶点光照低。 但是不会产生明显的棱角现象，因为逐片元光照对每个像素都进行差值处理，过渡比较平滑...

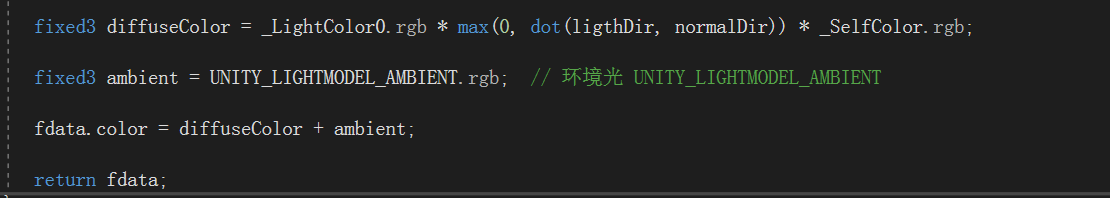
加个颜色属性，让物体可以自己改变自身颜色...





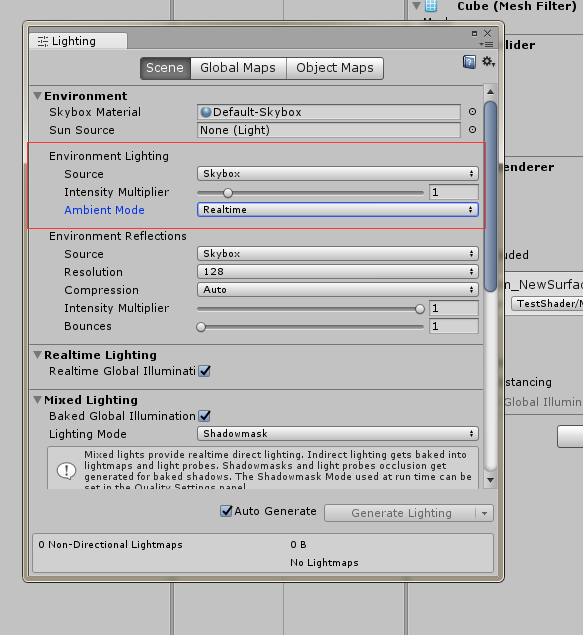
颜色融合 直接相乘

再加个环境光 受到环境光影响会变得更亮...



直接在后面相加 各种光叠加 直接相加

环境光可以在Unity中的 window -> lighting -> settings 中设置



....................  **半兰伯特光照模型** ....................

公式： **Lum = C \* cos<L, N>\*0.5 + 0.5**

C：到达此点光线的强度和颜色

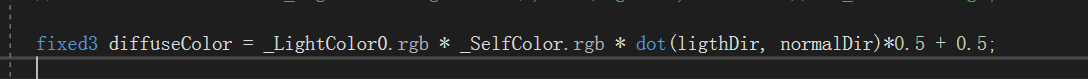
N：法线

L：单位长度（标准化）的入射光向量

经过稍微改造后的漫反射... 乘以一个0.5 再加一个 0.5

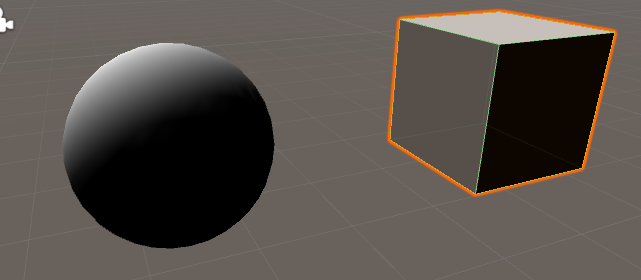
将 cos<> 余弦函数的值范围调整到 [ 0, 1 ]

为什么？ 请查阅 初中数学 余弦函数...

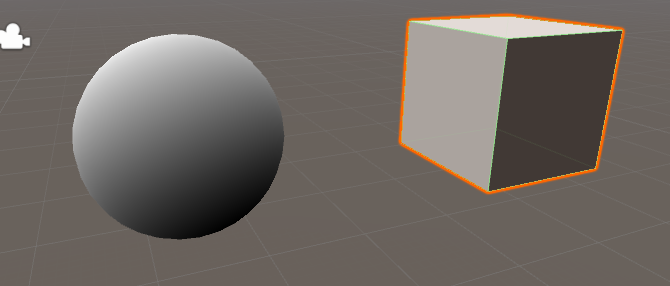


采用该模型，很明显就没那么黑了...

之前：



现在：



....................  **高光反射光照模型** ....................

公式： **Lum = C \* max( cos<R, V>, 0 )^** gloss

C：到达此点光线的强度和颜色

R：反射光方向单位向量

V：视野方向单位向量

Gloss：高光反射系数 ( 好像一般为10 ) 系数越大 亮点越小

\*\* 计算幂次方用函数 pow( x, y ) = x^y

提供 reflect 函数可计算反射光方向 reflect ( L, N ) ; 需传入两个参数 L ：入射光方向， N : 法线 这里注意的是 L 入射光方向得取负数

