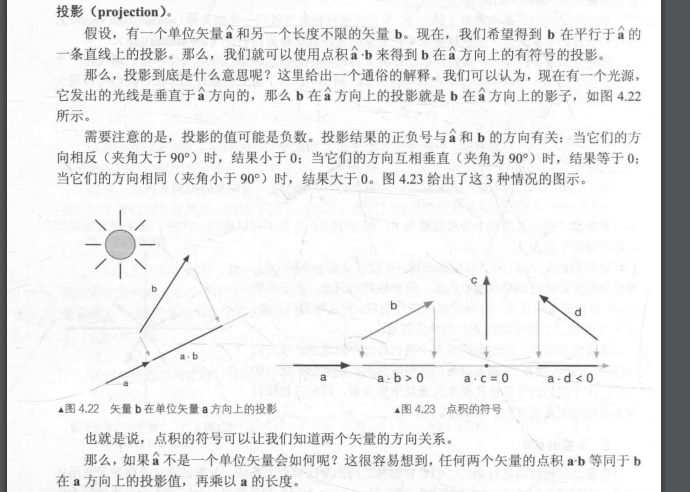
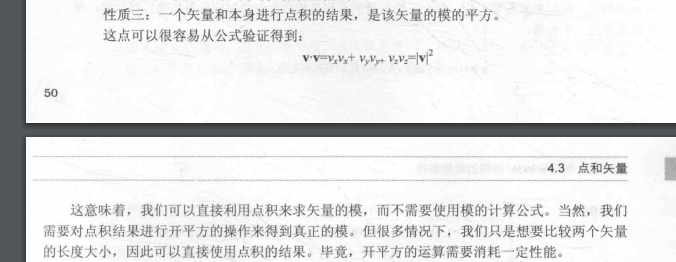
dot(a, b) 向量点积

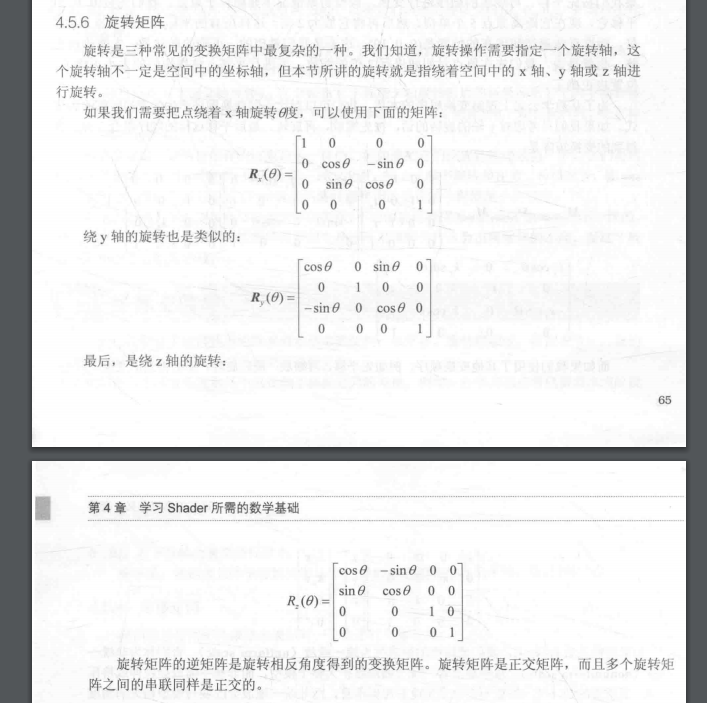
a·b = （ax, ay, az）·(bx, by, bz) = axbx + ayby + azbz;

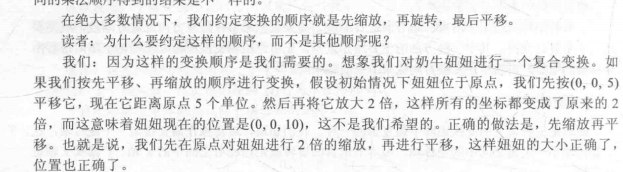
几何意义是投影



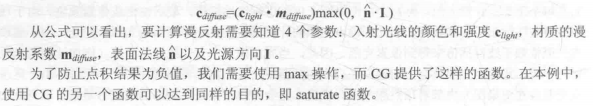


公式2： a·b = |a||b|cos角度

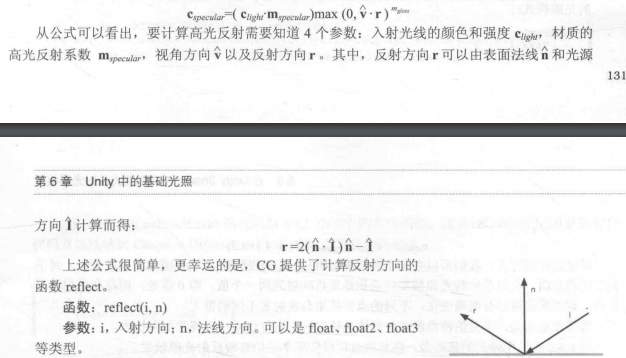




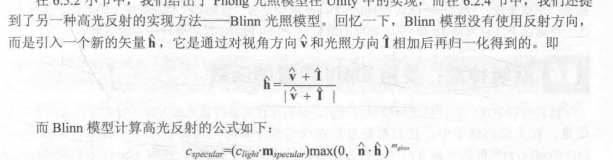
漫反射公式

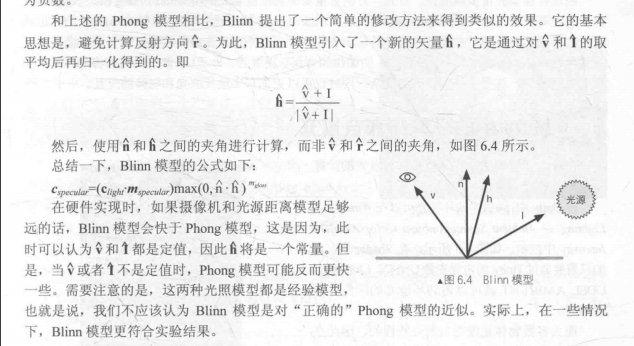


高光反射计算公式



Blinn-Phong光照模型(亮的区域更大)





Api：

1. saturate（x），强制x范围0-1
2. UNITY\_LIGHTMODEL\_AMBIENT, 环境光
3. \_WorldSpaceLightPos0, 光线方向（指向光源的）
4. \_LightColor0, 入射光纤颜色
5. 光照需要引用 Lighting.cginc
6. 计算反射：reflect(I, n) l入射方向,这里不能直接用\_WorldSpaceLightPos0，需要取反，因为此公式要的方向是从光源发出的方向

几点容易出错的地方：

1. 法线坐标转换的时候，法线比较特殊，直接用\_Object2World不行，需要用它的逆转置矩阵，所以需要用\_World2Object,这是逆矩阵，而把法线写到前面，就变成逆转置矩阵了，即



1. 获取视角方向时，这个变量要的是从点到摄像机的向量，而不是从摄像机到点，所以用摄像机减点

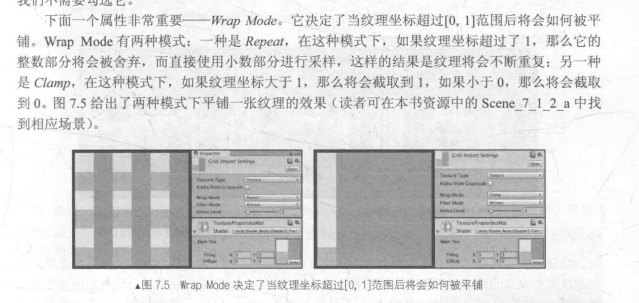


纹理名\_ST，ST是缩放（scale）和平移(translation)的缩写，可以让我们得到该纹理的缩放和平移值， \_MainTex\_ST.xy存储的是缩放值,而\_MainTex\_ST.zw存储的是偏移值

//这个人对于颜色相乘的解释特别好

正片叠底（multiply）：将两个颜色的像素值相乘，然后除以255得到的结果就是最终色的像素值。通常执行正片叠底模式后的颜色比原来两种颜色都深。任何颜色和黑色正片叠底得到的仍然是黑色，任何颜色和白色执行正片叠底则保持原来的颜色不变，而与其他颜色执行此模式会产生暗室中以此种颜色照明的效果。  
　　像素点的像素值是0-255，黑色值是0，白色是255.

也就是当两个颜色相乘的时候表示的是两种颜色的叠加。而两个颜色进行相加是为了给光源制造强光效果。其实从最终的计算效果来看，乘法产生的值一般都是偏小的，会相互抵消原先的一些颜色。而加法的值都是向1这个方向靠近的，也就是向白色方向靠近，约趋近白色也就表示越亮。

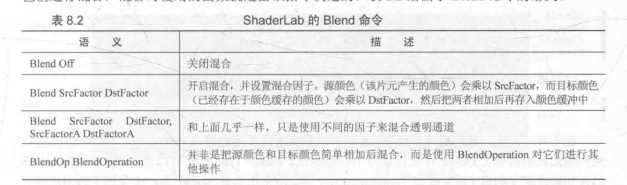


像素和法线的映射公式

Pixel = (normal + 1) / 2

深度测试 ZWrite

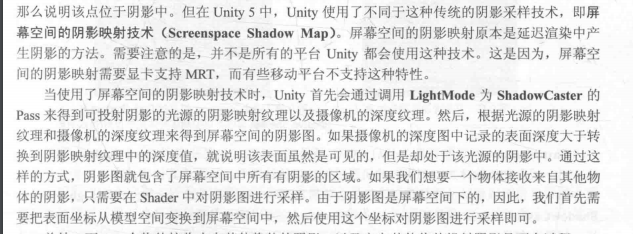
透明度测试：在片元着色器中使用void clip(float x)进行透明度测试



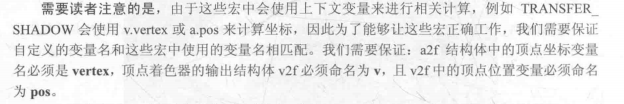
# 阴影

Unity阴影检测原理：Shadow Map，也就是把摄像机放在与光源重合的位置上，那么场景中该光源的阴影区域就是那些摄像机看不到的地方

Unity用一个额外的pass来专门更新光源的阴影映射纹理，即LightMode=ShadowCaster,



这shader真的坑



阴影书写规范：

1. 在v2f中加入 SHADOW\_COORDS(2) 参数就是TEXCOORD当前是几就写几
2. vert中加入TRANSFER\_SHADOW(o);
3. frag中，fixed shadow = SHADOW\_ATTENUATION(i);先算出shadow值，然后和漫反射高光反射相加的结果相乘

UNITY\_LIGHT\_ATTENUATION:又计算光照衰减又计算阴影，在AutoLight.cginc中

用法：UNITY\_LIGHT\_ATTENUATION(atten, i, i.worldPos),注意不用声明atten

如果想在Additional Pass中添加阴影效果，需要使用

#pragma multi\_compile\_fwdadd\_fullshadows

内置的VertexLit没有提供透明度计算，可以把Fallback设置为Transparent/Cutout/VertexLit

