目录

[关于光照得一些名词 1](#_Toc23481)

[常用光照模型 2](#_Toc8695)

[渲染路径 3](#_Toc17217)

[lightMode标签参数 3](#_Toc18312)

[前向渲染路径 3](#_Toc30128)

[常规前向渲染说明 3](#_Toc8176)

[Unity前向渲染路径中有3种处理光照方式 4](#_Toc4601)

[Unity前向渲染路径光源设置 4](#_Toc31918)

[Unity前向渲染处理方式与光源的关系 4](#_Toc148)

[Unity前向渲染路径的两个pass 4](#_Toc21895)

[延迟渲染路径 6](#_Toc28332)

# 关于光照得一些名词

**光源**(light source)

**辐照度**(irradiance) ：光源发出多少光

光照到物体会发生散射和吸收

**散射**(scattering) : 改变光得方向，不改变光得密度颜色

**吸收**(absorption) : 会改变光线得密度和颜色

光在物体表面散射，会有两种方向

**折射**(refraction)或**透射**(transmission) ：将光散射到物体内部

**反射**(reflection) : 将光散射到物体外

为了描述散射，在光照模型中使用不同部分来计算

**高光反射**(specular):描述物体表面如何反射光线

**漫反射**(diffuse):描述光纤被折射，吸收，散射

**出射度**(exitance):描述出射光线得数量和方向，辐照度和出射度之间满足线性关系，他们得比值就是材质漫反射,高光反射得属性

**着色**(shading):根据材质属性(漫反射，高光反射等)，光源信息(光源方向，辐照度等)，使用一个等式计算沿着某个观察方向得出射度得过程，这个等式就是**光照模型**(Lighting Model)

**自发光**(emissive)

**环境光**(ambient)

标准光照模型:自发光+高光+漫反射+环境光

# 常用光照模型

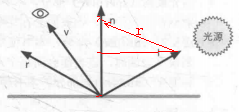
Lambert模型(漫反射)



Phong模型(镜面反射+漫反射)



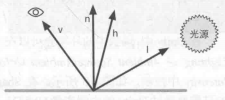
这个公司那个是只是高光部分



Blinn-Phong模型(修正Phong)

 h是v和i取平均后再归一化得到





BRDF（Bi-directional Reflection Distribution Function）双向反射分布函数:

给定入射光线得放射和辐射度后，BRDF可以给出在某个出射方向上得光照能量分布

这里涉及到复杂得物理渲染过程，暂时略过。

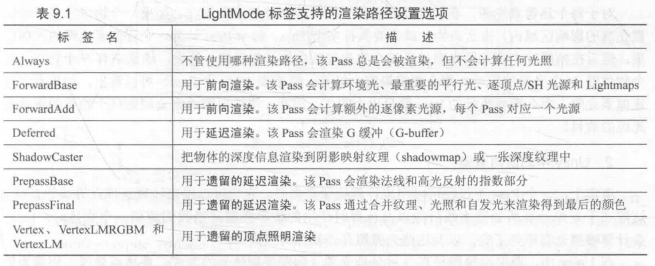
Lambert 模型能够较好地表现粗糙表面上的光照现象，如石灰墙，纸张等等，但是在渲染金属材质制成的物体时，则会显得呆板，表现不出光泽，主要原因是其没有考虑到镜面反射效果，所以Phong模型对其进行了很好的补充。由于Blinn-phng光照模型混合了Lambert的漫射部分和标准的高光，渲染效果有时会比 Phong高光更柔和，有些人认为phong光照模型比blinn-phong更加真实，实际上也是如此，Blinn-phong渲染效果要更加柔和一些，但是由于Blinn-phong的光照模型省去了计算反射光线方向向量的两个乘法运算，速度更快，因此成为许多CG软件中默认的光找渲染方法（gl，dx）

# 渲染路径

修改uinty渲染路径位置，

camera上直接修改，多个相机可以使用不同的渲染路径

## lightMode标签参数



## 前向渲染路径

### 常规前向渲染说明

一个物体对于每个逐像素光源，都要一个单独的pass进行渲染

Unity中的前向路径渲染目的和简要说明

只计算少量重要的光源

假设场景里有一堆光源都在一个物体周围，unity Quality Setting中设置的最大像素光为4

1找最亮的平行光做主光源在ForwardBase中处理。

2然后找3个可用像素光源(光源设置为Important)在ForwardAdd中渲染，因为最大像素光是4减去一个主光源就剩3，如果不够3个，就从光源设置为not Important中找，补足3个放到ForwardAdd中渲染

3 其他光源找出4个最相对重要的作为逐顶点方式渲染(就是把这四个光的信息存到指定变量中，由forwardBase同一渲染)

4 其他光源由SH近似在forwardBase中渲染

Unity具体给光源排序的方式是黑盒，大体满足如下条件

1 最亮平行光，2 其他像素光源，3 非像素光 具体每个光源的排序还跟光的影响力有关

### Unity前向渲染路径中有3种处理光照方式

1 逐像素处理:对重要光源的处理，最重要的在forwardBase处理，其他在forwardAdd处理

2 逐顶点处理：意味着光的信息被放到了4个固定的光源信息变量中，因为只需要处理一次所以在forwardBase中处理

3 球谐函数(Spherical Harmonics,SH)( 这个简单说就是一堆不重要的光源直接用一个函数近似处理)处理。因为只需要处理一次所以在forwardBase中处理

### Unity前向渲染路径光源设置

光源可以设置为Important和Not Important

设置成Important也可以叫像素光源

意味着unity建议用逐像素渲染，并把相关光照数据存到固定变量中

注意是建议，在自定义Forwardbase中如果逐顶点也可以，但一般有法线的情况还是在frag中处理，所以可以人为就是逐像素这一种情况

设置成Not Important意味着用逐顶点或SH处理(除非满足前向处理方式与光源关系第4条，这时一些Not Important的光也可以用逐像素方式渲染)

### Unity前向渲染处理方式与光源的关系

1 场景种最亮的平行光总是按像素处理(forwardBase种主要处理的逐像素光)

2 设置成Not Important的光源，会按逐顶点或SH处理

3 设置成Important的光源，会按逐像素处理

4 根据以上规则得到的逐像素光源数小于Quality Setting中设置的(最大)逐像素光源数(Pixel Light Count),会有更多光源以逐像素的方式渲染

### Unity前向渲染路径的两个pass

ForwardBase

1 一个ForwardBase只执行一次，但是可以有多个pass标识为ForwardBase(比如双面材质)

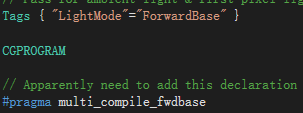
2 ForwardBase种处理逐像素光一定是平行光

3 在ForwardBase中处理最多4个顶点光和SH函数，环境光，自发光也在这里处理

4 ForwardBase中可以读取LightMap

5 ForwardBase中默认开启了阴影

6 注意，要把#pragma multi\_compile\_fwdbase加上，否则拿不全正确光源信息

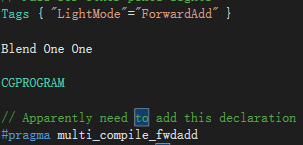


ForwardAdd

用于处理非主光源之外的像素光源，注意前向处理方式与光源关系第4条

ForwardBase中默认不支持阴影，可以用#pragma multi\_compile\_fwadd\_fullshadows指令开启阴影

注意，要把#pragma multi\_compile\_fwdadd加上，否则拿不全正确光源信息





说明\_LightMatrix0已经变更为unity\_WorldToLight



### 光照衰减

光照衰减公式计算量大，unity中从固定贴图\_LightTexture0中取衰减参数

如果光源使用了cookie，那么衰减查找纹理就是\_LightTextureB0(这个还不清除什么情况)

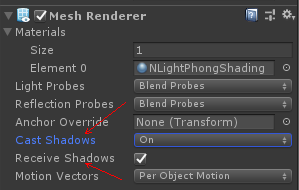
平行光无衰减 attent = 1 .0

点光源衰减 存到LightTexture0中

聚光灯 此时衰减纹理存储到了\_LightTextureB0中，这张纹理和点光源中的\_LightTexture0是等价的，\_LightTexture0存储的不再是基于距离的衰减纹理，而是一张基于张角范围的衰减纹理

## Unity内置阴影的一些注意事项

Unity使用的是屏幕阴影映射，有些平台不能使用这个就使用传统阴影映射



**投射阴影的物体**：这个勾选了，就会执行这个物体的ShadowCaster的pass，把物体深度信息渲染到深度贴图中

**接收阴影的物体**: 这个勾选了，渲染当前物体时会根据当前深度贴图渲染其他物体在这个物体上的影子

使用内置函数渲染影子基本流程可以在NLightFullForward.shader中找到，3步就能完成

关于ShadowCaster这个pass的说明

Forward Render中有base和add两个pass，ShadowCaster这可以是第三个pass用来把物体深度信息渲染到深度贴图中，这个深度贴图任何接收阴影的物体计算阴影时都要用到

可以自己实现这个pass，如果不实现，unity就会使用shader的callback中的ShadowCaster，比如 Fallback Diffuse，就会使用Diffuse的ShadowCaster pass

特殊说明下透明物体alpha blend 和alpha test的处理方式

可以分别在callback中使用

FallBack "Transparent/VertexLit"

FallBack "Transparent/Cutout/VertexLit"

Unity"Transparent/VertexLit"中没有ShadowCaster pass

如果想要产生阴影可以使用Fallback Diffuse

## 延迟渲染路径

### 延迟渲染路径与前向渲染路径比较

