

# PBR notes

## 前言约定

- 记号
  1. L 光照方向
  2. N 法线方向
  3. V 视角方向
  4. T 切线方向
  5. H 半角向量
- 无论入射光，反射光，其方向向量都约定为从表面点出发向外。

## 1. 基于物理的渲染

### 1.1 能量守恒

$$\forall \omega_i \text{ 有 } \int_{\Omega} f_r(p, \omega_i, \omega_o)(n \cdot \omega_i) d\omega_i \leq 1$$

### 1.2 光路可逆（互异性）

## 2. 迪斯尼原则的 PBR

- 可实时计算
- 参数少且简单直观
- 能涵盖绝大多数材质表现
-

### 3. Direct lighting

#### 3.1 Cook-Torrance 微平面模型

法线分布函数项 Normal distribution function (specular D)

几何遮蔽项 Geometric shadowing (specular G)

菲涅尔项 Fresnel (specular F)

#### 3.2 渲染方程：


$$L_o(p, \omega_o) = k_d \int_{\Omega} \left(\frac{\rho}{\pi}\right) L_i(p, \omega_i) n \cdot \omega_i d\omega_i + k_s \int_{\Omega} \left(\frac{DFG}{4 \cdot (\omega_o \cdot n) \cdot (\omega_i \cdot n)}\right) L_i(p, \omega_i) n \cdot \omega_i d\omega_i$$

kd 对应 diffuse 部分，ks 对应 specular 部分

### 4. Indirect lighting (IBL)

#### 4.1 Indirect diffuse lighting

$$L_o(p, \omega_o) = \left(\frac{\rho}{\pi}\right) \int_{\Omega} L_i(p, \omega_i) n \cdot \omega_i d\omega_i$$

  
预计算

#### 预积分计算 irradiance map

从 radiance map（环境立方体贴图）预积分计算出 irradiance map，环境立方体贴图的每个像素代表一个环境光源，积分计算中，以给定点的法线方向确定的上半球面为积分域，而上式中的积分值仅与法线  $n$  有关，因此我们可以针对每一个法线取值预计算出积分值，然后保存到贴图中以供实时渲染时查询使用，这张贴图称为 irradiance map，这实际上是对原环境立方体贴图做了一个核为  $n \cdot \omega_i$  的一个卷积计算，如下图所示。



Left the radiance map, right the irradiance map (Lambert rendering equation)

## Monte Carlo 计算定积分

要计算定积分：

$$F = \int_{\Omega} f(x) dx$$

可以采用对被积函数的随机采样值的统计平均来近似估算积分

$$F \approx F_N = \frac{1}{N} \sum_{i=1}^N \frac{f(x_i)}{p(x_i)}, \text{ 其中 } p(x_i) \text{ 是取 } x_i \text{ 的概率。}$$

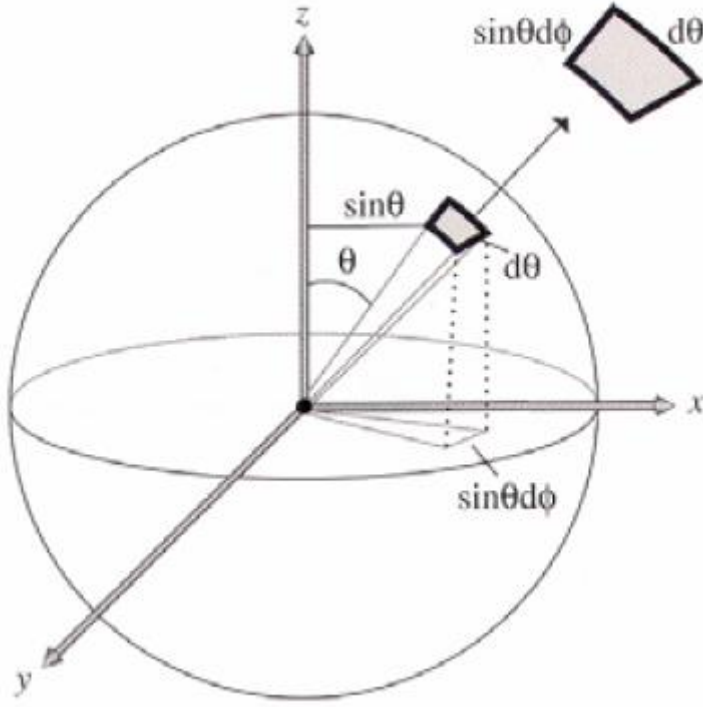
可以证明统计值的数学期望： $E[F_N] = F$ ，也即是  $F$  的一个无偏估计。

## 球面均匀随机采样

- 对给定的方向，以其为中心生成均匀分布的半球面采样点[5]。
- 非差异化随机序列

## 半球面上的积分离散采样

$$L_o(p, \omega_o) = k_d \left( \frac{\rho}{\pi} \right) \int_{\Omega} L_i(p, \omega_i) n \cdot \omega_i d\omega_i, \text{ 转到球面坐标下}$$



$d\omega_i = \sin(\theta)d\theta d\phi$  , 其中  $\theta \in [0, \frac{\pi}{2}]$ ,  $\phi \in [0, 2\pi]$  , 为了球面均匀采样 , 再做积分换元[6] ,

$$\begin{cases} u = 1 - \cos\theta \\ v = \frac{\phi}{2\pi} \end{cases}$$

得到

$$\begin{aligned} L_o(p, \omega_o) &= k_d \left( \frac{\rho}{\pi} \right) \int_0^{2\pi} \int_0^{\frac{\pi}{2}} L_i(p, \omega_i) n \cdot \omega_i \sin\theta d\theta d\phi \\ &= k_d \left( \frac{\rho}{\pi} \right) \int_0^1 \int_0^1 L_i(p, \omega_i) n \cdot \omega_i \sin\theta J du dv \end{aligned}$$

$$J = \begin{vmatrix} \frac{\partial\theta}{\partial u} & \frac{\partial\theta}{\partial v} \\ \frac{\partial\phi}{\partial u} & \frac{\partial\phi}{\partial v} \end{vmatrix} = \frac{2\pi}{\sqrt{1 - (1 - u)^2}} = \frac{2\pi}{\sin\theta}$$

其中  $J$  是多重积分换元的雅可比行列式

代入得： $L_o(p, \omega_o) = k_d(2\rho) \int_0^1 \int_0^1 L_i(p, \omega_i) n \cdot \omega_i du dv$  , 进行 Monte-Carlo 随机采样 ,  $u, v$  服从  $(u, v) \sim U[0, 1]^2$  均匀分布 , 概率密度函数(pdf)为常数 1 , 从  $u, v$  计算出  $\theta, \phi$  , 然后计算方向向量  $\omega_i = (\cos\phi\sin\theta, \sin\phi\sin\theta, \cos\theta)$  , 则 Monte Carlo 积分格式为:

$$L_o(p, \omega_o) \approx k_d \rho \left[ \frac{2}{N} \sum_a^b L_i(p, \omega_i) n \cdot \omega_i \right]$$

注意: albedo 值  $\rho$  和漫反射系数  $k_d$  不预计算到 irradiance map 中 , 预计算已经除过了  $\pi$ 。

## 漫反射系数的计算

specular 系数就是用  $F$ ，对应的漫反射系数，要考虑金属度

$$K_s = F$$

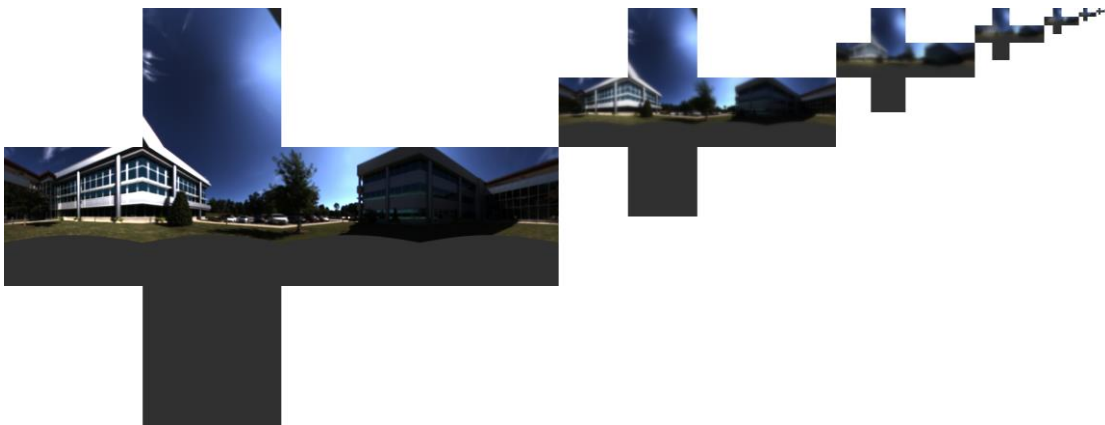
$$K_d = (1 - F)(1 - \text{metallic})$$

## 4.2 Indirect specular lighting

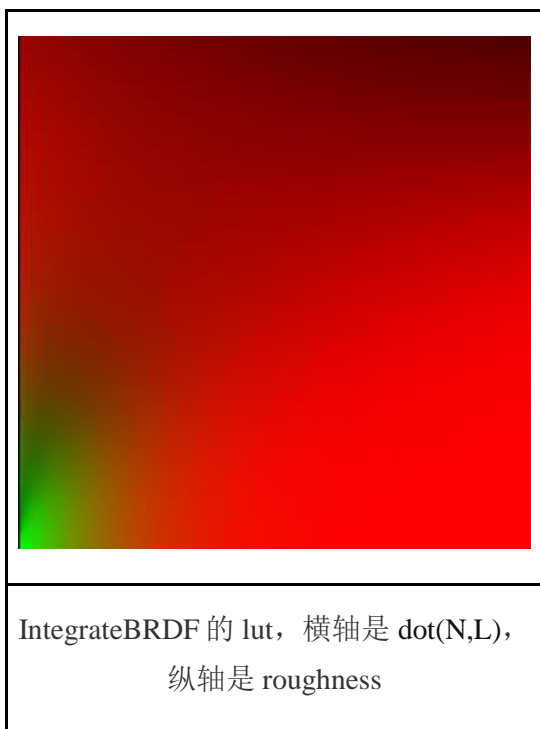
### 利用分离和近似拆分为两部分的乘积：

#### 1. 第一部分是预滤波环境贴图

它类似于辐照度图，是预先计算的环境卷积贴图，但这次考虑了粗糙度。因为随着粗糙度的增加，参与环境贴图卷积的采样向量会更分散，导致反射更模糊，所以对于卷积的每个粗糙度级别，按顺序把模糊后的结果存储在预滤波贴图的 mipmap 中，roughness 越小对应于 mip 层级越高。



#### 2. 第二部分是 brdf-lut，也可以用函数拟合方式计算（使命召唤黑色行动 2）



PartTwo 的结果出了可以预计算出来外，还可以用函数拟合来实时计算。

```
// 这是使命召唤黑色行动 2 的函数拟合 Black Ops II
float2 EnvBRDFApprox_BlackOp2(float Roughness, float NV)
{
    float g = 1 - Roughness;
    float4 t = float4(1/0.96, 0.475, (0.0275 - 0.25*0.04)/0.96, 0.25);
    t *= float4(g, g, g, g);
    t += float4(0, 0, (0.015 - 0.75*0.04)/0.96, 0.75);
    float A = t.x * min(t.y, exp2(-9.28 * NV)) + t.z;
    float B = t.w;
    return float2 ( t.w-A,A);
}
```



```
//UE4 在 黑色行动 2 上的修改版本
float2 EnvBRDFAprox_UE4(float Roughness, float NoV )
{
    // [ Lazarov 2013, "Getting More Physical in Call of Duty: Black Ops II" ]
    // Adaptation to fit our G term.
    const float4 c0 = { -1, -0.0275, -0.572, 0.022 };
    const float4 c1 = { 1, 0.0425, 1.04, -0.04 };
    float4 r = Roughness * c0 + c1;
    float a004 = min( r.x * r.x, exp2( -9.28 * NoV ) ) * r.x + r.y;
    float2 AB = float2( -1.04, 1.04 ) * a004 + r.zw;
    return AB;
}
```

## 5. IBL error source

### 1. 分离和近似引入的误差

### 2. $v=n=r$ 假设引入的误差

## 6. important tricks

- BRDF 的几何遮蔽项（SpecularG）在计算 IBL 和直接光时略有不同：

$$K_{direct} = \frac{(roughness + 1)^2}{8}$$

$$K_{IBL} = \frac{(roughness)^2}{2}$$

- Fresnel 项在计算 IBL 时和计算直接光时的有所区别：

在计算 IBL 时，由于环境光来自半球内围绕法线  $N$  的所有方向，因此没有一个确定的半向量来计算菲涅耳效应。为了模拟菲涅耳效应，我们用法线和视线之间的夹角计算菲涅耳系数。然而，之前我们是以受粗糙度影响的微表面半向量作为菲涅耳公式的输入，但我们目前没有考虑任何粗糙度，表面的反射率总是会相对较高。间接光和直射光遵循相同的属性，因此我们期望较粗糙的表面在边缘反射较弱，引入粗糙度，降低反射。



$$F_{fresnel} = f_0 + (1 - f_0)(1 - h \cdot v)^5$$

$$F_{fresnel\_roughness} = f_0 + (\max(f_0, 1 - roughness) - f_0)(1 - n \cdot v)^5$$

- roughness 为 0，对应着完全的反射，对于非面积光源，会产生无穷大的反射值（能量全部集中到了面积为 0 的一点上），UE4 的 SM5 下 roughness 默认最小值为 0.2
- F：菲涅尔系数，是光线发生反射与折射的比例，当光线垂直进入表面时的菲涅尔系数记为 F0，F0 是可以实际测量出来的，不同的材质，这个 F0 是不同的，F0 越大，感觉是越明亮，金属材质通常 F0 比较大，而且是有颜色的（不同频率的光，对应菲涅尔系数不同），非金属材质最低也有一点点反射，就取(0.04, 0.04, 0.04)，渲染中通常用一个权重系数（金属度）在非金属材质的最小菲涅尔系数和具有最强金属性的材质菲涅尔系数（自身漫反射颜色）之间进行线性插值得到 F0。
- 实际计算中，高光项并没有再乘上 ks。

#### [reference]

[1] <https://learnopengl.com/PBR>

[2] <https://google.github.io/filament/Filament.html>

[3] [http://www.codinglabs.net/article\\_physically\\_based\\_rendering.aspx](http://www.codinglabs.net/article_physically_based_rendering.aspx)

[4] <https://seblagarde.wordpress.com/2011/08/17/hello-world/>

[5] <http://www.sztemple.cc/articles/pbr%E7%90%86%E8%AE%BA%E4%BD%93%E7%B3%BB%E6%95%B4%E7%90%86%Ef%BC%88%E4%B8%89%Ef%BC%89%Ef%BC%9aib>

[6] <http://corysimon.github.io/articles/uniformdistn-on-sphere/>

[Schlick94] Christophe Schlick. 1994. An Inexpensive BRDF Model for Physically-Based Rendering. Computer Graphics Forum, 13 (3), 233-246.

[7] [https://en.wikipedia.org/wiki/Inverse\\_transform\\_sampling](https://en.wikipedia.org/wiki/Inverse_transform_sampling)

[8] [https://dk81.github.io/dkmathstats\\_site/prob-inverse-cdf.html](https://dk81.github.io/dkmathstats_site/prob-inverse-cdf.html)

[9] <https://www.cnblogs.com/timlly/p/10631718.html>