

# Contents

<b>Contents</b> .....	<b>1</b>
0.1 相关工作 .....	1
0.1.1 大量点光源光照 .....	1
0.1.2 实时光照模型 .....	1
0.1.3 Linearly Transformed cosine .....	2
0.2 算法部分 .....	2
0.2.1 反射率方程 .....	2
0.2.2 使用LTC方法近似BRDF积分 .....	4
0.2.3 使用LTC方法来计算单点光源光照 .....	5
0.2.4 使用LTC方法来计算多个点光源光照 .....	7
0.2.5 光源裁剪 .....	8
0.2.6 实验结果 .....	9
<b>Bibliography</b> .....	<b>11</b>



## 0.1 相关工作

### 0.1.1 大量点光源光照

为什么要渲染大量点光源？在对场景做渲染时，由于场景中的光源数量可能很多，甚至是动态的光源时，同时保证渲染效果和渲染速度仍是比较困难的。现如今有很多关于在场景中实时绘制大量点光源的算法，比如Forward+算法[1]里将屏幕分块来进行光源剔除，而为了解决分块中包含的光源太多以及深度不连续等等问题，分簇着色[2]在分块着色的基础上，将分块从二维的屏幕空间延伸到三维的观察空间，在深度上进一步划分。LLL算法[3]为每个像素建立了一个GPU光源链表，在进行光源剔除的同时使得在延迟渲染中，可以在一遍Pass中同时渲染不透明和半透明物体，让半透明物体和粒子的光照渲染变得更加简单。但是这些算法都是在对大量的点光源进行光源剔除，当点光源的光照模型不是Phong模型，而是诸如GGX BRDF等基于物理的复杂光照模型时，渲染效率依然会很低。

### 0.1.2 实时光照模型

用于点光源渲染的光照模型有很多。Bouknight曾提出Lambert模型[4]，但是只适用于粗糙表面。Phong模型加入了镜面反射，而Blinn-Phong模型[5]是对Phong模型的扩展，修正了Phong模型的镜面光效果，计算效率也更高。但是它们都不符合物理规律。在基于物理的渲染中，通常使用一个双向反射分布函数[6]来表示物体表面的反射辐射度(radiance)占入射辐照度(irradiance)的比率。完全遵循物理规律的BRDF函数会非常复杂，实际渲染中通常都是使用近似的方法，不同的近似方法会形成不同的BRDF模型。如今已经有很多比较理想的经验BRDF模型[7]，比如基于分层理论的Granier-Heidrich BRDF模型[8]、基于几何观点的Oren-Nayar模型[9]、基于实验的Schlick BRDF模型[10]等等，不过在实时渲染以及游戏引擎中使用最多的还是Cook等人在Beckman[11]和Torrance-Sparrow[12]的基础上提出的基于微面元理论的Cook-Torrance BRDF模型[13]，该理论认为任何表面都是由非常小的微观面元组成，每个微观面元都有自己的粗糙度。除此之外还有很多Cook-Torrance BRDF模型的变种[14][10][15]，该模型也被实验证明能够很好地模拟真实的表面[16]。对于该模型中的微观面元法线分布函数，比较流行的是Beckmann[11]、Phong[17]以及GGX[18]等分布函数，但是只有GGX分布才会高光中心处衰

减较快，远离高光中心后衰减会变得很慢，使得GGX高光产生拖尾效应，而这一点非常符合现实情况[19]。而微观面元几何遮挡函数(bidirectional shadowing-masking function)可以通过法线分布函数和史密斯阴影函数(Smith Shadowing function)得到，但是史密斯阴影函数是假设入射遮挡(shadowing)和出射遮挡(masking)是不相关的，不过近年来更流行的是Lagarde等人[20]提出的更精确的几何遮挡函数，它考虑了入射遮挡和出射遮挡的相关性。对于漫反射BRDF，曾经比较流行的是lambert BRDF模型[4]，但是由于它失去了方向性，无法有效反映回射反射(retro-reflection)的效果，Disney提出了一个新的漫反射经验模型[21]。

### 0.1.3 Linearly Transformed cosine

由于基于物理的BRDF模型公式复杂并且计算量较大，所以Eric Heitz等人在实现面光源光照时，提出了一种名为Linearly Transformed Cosine的数学方法，用一个线性矩阵和余弦分布来拟合BRDF分布。本文工作受此启发，使用Linearly Transformed Cosine来实现基于物理的大量点光源光照，以达到降低计算量的目的。

## 0.2 算法部分

本文算法要实现的目标是：用LTC的方式来近似实现多点光源下基于物理的光照效果，同时渲染效率要比常用的BRDF方式更高。

### 0.2.1 反射率方程

为了实现更真实的渲染效果，如今业界广泛使用如下的反射率方程来实现基于物理的光照渲染：

$$L_o(p, \omega_o) = \int_{\Omega} f_r(p, \omega_i, \omega_o) L_i(p, \omega_i) n \cdot \omega_i d\omega_i \quad (1)$$

其中 $p$ 着色点， $n$ 是表面法线， $\omega_o$ 为观察方向， $\omega_i$ 为光线入射方向， $L_i$ 是来自 $\omega_i$ 方向上的辐射亮度， $L_o$ 是着色点在上半球空间接受到的辐射度， $f_r$ 是双向反射分布函数(BRDF)，描述了反射方向上的辐射照度与入射方向上的辐射照度的比率。当前主流的游戏引擎(如Unity、Unreal等)都在使用基于微面元

理论的BRDF模型：

$$f_r(\omega_i, \omega_o) = f_d + \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)} \quad (2)$$

其中 $f_d$ 是漫反射系数，D是描述微观面元法线分布的函数，F是描述表面菲涅尔反射的函数，G是描述微观面元之间几何遮挡比率的函数。

所以我们通常使用的反射率方程是：

$$L_o(p, \omega_o) = \int_{\Omega} (f_d + \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)}) L_i(p, \omega_i) n \cdot \omega_i d\omega_i \quad (3)$$

本文采用的D、F、G函数与Unity引擎内部使用的函数一致。

对于法线分布函数D本文采用GGX分布[18]

$$D = \frac{\alpha^2 \chi^+(\theta_h)}{\pi \cos^4 \theta_h (\alpha^2 + \tan^2 \theta_h)^2} \quad (4)$$

其中 $h$ 表示中间向量， $\alpha$ 表示表面粗糙度， $\theta_h$ 是法向量 $n$ 与中间向量 $h$ 的夹角，并且：

$$\chi^+(\theta_h) = \begin{cases} 1, & \theta_h > 0 \\ 0, & \theta_h < 0 \end{cases} \quad (5)$$

对于菲涅尔函数本文采用渲染领域中常用的Schlick近似模型[10]：

$$F = F_0 + (1 - F_0)(1 - (h \cdot \omega_o)^5) \quad (6)$$

其中 $F_0$ 为入射光垂直表面时的菲涅尔反射率的值，即表面的基础反射率。

对于几何遮挡函数G，本文采用了近年来越来越多的引擎开始使用的更精确的一种遮挡函数[20]：

$$G = \frac{\chi^+(\omega_o \cdot h) \chi^+(\omega_i \cdot h)}{1 + \Lambda(\omega_o) + \Lambda(\omega_i)} \quad (7)$$

其中：

$$\Lambda(m) = \frac{-1 + \sqrt{1 + \alpha^2 \tan^2(\theta_m)}}{2} \quad (8)$$

其中 $\theta_m$ 是向量 $m$ 和法线 $n$ 之间的夹角。

对于式-2中的漫反射项 $f_d$ ，本文采用Disney最新使用的漫反射模型[21]：

$$f_d = \frac{c}{\pi}(1 + (F_D - 1)(1 - \cos\theta_l)^5)(1 + (F_D - 1)(1 - \cos\theta_v)^5) \quad (9)$$

其中 $\theta_l$ 是 $\omega_i$ 和法线 $n$ 的夹角， $\theta_v$ 是 $\omega_o$ 和 $n$ 的夹角，并且：

$$F_D = 0.5 + 2\alpha \cos^2 \theta_d \quad (10)$$

其中 $\theta_d$ 是入射向量 $\omega_i$ 和中间向量 $h$ 的夹角。

可以初步看到，渲染时因为要按照上述公式去分别计算D、F、G项以及 $f_d$ ，再得到最终的辐射度 $L_o$ ，所以即使是在一个单点光源的环境下，着色点渲染时所需要的计算量也很大。

### 0.2.2 使用LTC方法近似BRDF积分

Eric Heitz等人在解决区域光源光照时提出了一种名为线性转换余弦分布（Linear Transform cosin）的数学方法[22]，来近似计算BRDF积分。如图1所示：图中，上面是在某种粗糙度 $\alpha$ 以及观察向量 $\omega_o$ 下的BRDF分布，分布函数如下：

$$B_s(\omega_i) = \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)} n \cdot \omega_i \quad (11)$$

下面是一个截断余弦（Clamped Cosine）分布，它的分布函数如下：

$$C(\omega_i = (x, y, z)) = \frac{1}{\pi} \max(0, z) \quad (12)$$

其中 $\omega_i$ 是单位向量。

作者提出可以在下面的余弦分布上乘以一个3x3的线性变换矩阵 $M_s$ ，来近似得到上面的BRDF分布。该矩阵 $M_s$ 的形式如下：

$$M_s = \begin{bmatrix} a & 0 & b \\ 0 & c & 0 \\ d & 0 & 1 \end{bmatrix} \quad (13)$$

以二维纹理的纹理坐标 $u$ 作为观察角度 $\theta_o$ ，以纹理坐标 $v$ 作为粗糙度 $\alpha$ ，拟合出不同的矩阵 $M_s$ 。可以将预计算好的 $M_s$ 的逆矩阵存入二维纹理坐标 $(u, v)$ 所对应的纹素中。作者提出在BRDF分布上对一个多边形光源区域 $P$ 的积分，等

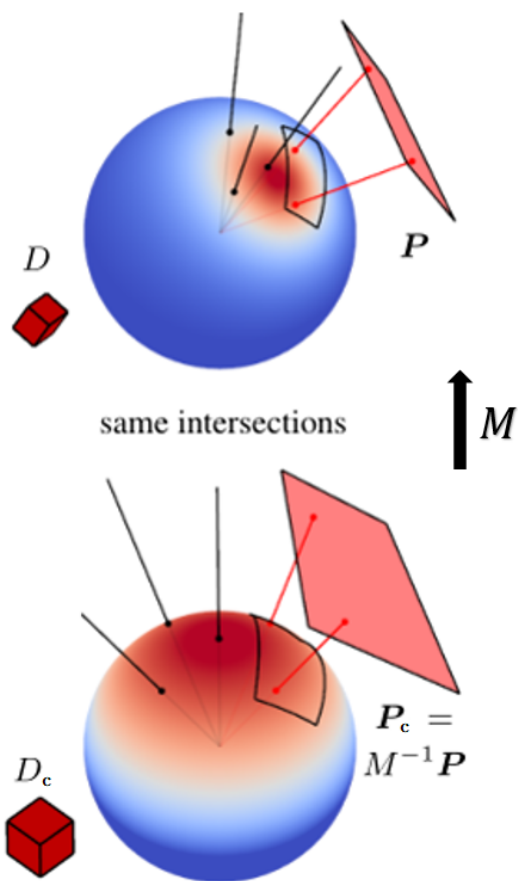


Figure 1: 使用余弦分布乘上矩阵M来近似BRDF分布

效于在余弦分布上对另一个多边形区域 $P_c$ 的积分（多边形 $P_c$ 是对多边形 $P$ 乘上矩阵 $M_s$ 的逆矩阵之后，得到的新的多边形）：

$$\int_P B_s(\omega) d\omega = \int_{P_c} C(\omega_s) d\omega_s \quad (14)$$

其中 $\omega$ 是在BRDF分布下的光线入射向量，而 $\omega_s$ 是 $\omega$ 乘上矩阵 $M_s^{-1}$ 作线性变换之后的新的入射向量。

### 0.2.3 使用LTC方法来计算单点光源光照

将上式-14中的 $\omega$ 看作单位球上的无限小的面积，那么上式中对 $\omega$ 的积分实际上就是在对一个点光源进行光照计算。那么一个着色点接受到来自 $\omega$ 方向上

的点光源的镜面辐射度为：

$$L_s = B_s(\omega) = C(\omega_s) \quad (15)$$

其中 $\omega_s$ 是对 $\omega$ 乘上线性余弦变换矩阵 $M_s$ 的逆矩阵之后得到的新的向量，即：

$$\omega_s = M_s^{-1}\omega \quad (16)$$

但是实际上在式14中无限小的 $d\omega$ 和 $d\omega_s$ 的面积是不一样的（因为 $d\omega_s$ 是 $d\omega$ 作线性变换得到的，线性变换的过程中会改变面积大小）。所以式15是不成立的，还需要乘上两个 $\omega$ 的面积比例：

$$L_s = B_s(\omega) = C(\omega_s) \frac{\partial \omega_s}{\partial \omega} \quad (17)$$

作者在文中已经给出了 $\frac{\partial \omega_s}{\partial \omega}$ 的推导结果，即：

$$B_s(\omega) = C(\omega_s) \frac{\partial \omega_s}{\partial \omega} = C(\omega_s) \frac{|M_s^{-1}|}{||M_s^{-1}\omega||^3} \quad (18)$$

同样的，本文对式9中的Disney漫反射也使用LTC方法拟合出另一个线性余弦变换矩阵 $M_d$ 。那么一个着色点接受来自 $\omega$ 方向上的点光源的漫反射辐射度为：

$$L_d = B_d(\omega) = C(\omega_d) \frac{|M_d^{-1}|}{||M_d^{-1}\omega||^3} \quad (19)$$

其中：

$$B_d(\omega) = f_d n \cdot \omega_i \quad (20)$$

$$\omega_d = M_d^{-1}\omega \quad (21)$$

所以，综合式2、式11、式18、式19以及式20，着色点接受到的来自 $\omega$ 方向上的点光源的辐射度为：

$$L_o = f_r L_i n \cdot \omega = (B_s(\omega) + B_d(\omega)) L_i = (C(\omega_s) \frac{|M_s^{-1}|}{||M_s^{-1}\omega||^3} + C(\omega_d) \frac{|M_d^{-1}|}{||M_d^{-1}\omega||^3}) L_i \quad (22)$$

由式12、式16、式21可以得到：

$$C(\omega_s) = \frac{(M_s^{-1}\omega) \cdot z}{||M_s^{-1}\omega||} \quad (23)$$



$$C(\omega_d) = \frac{(M_d^{-1}\omega) \cdot z}{||M_d^{-1}\omega||} \quad (24)$$

代入式22可得：

$$L_o = \frac{\omega_s \cdot z}{(\omega_s \cdot \omega_s)^2} |M_s^{-1}| L_i + \frac{\omega_d \cdot z}{(\omega_d \cdot \omega_d)^2} |M_d^{-1}| L_i \quad (25)$$

可以看到，借助LTC方法来计算点光源的辐照度，只需要计算几次向量乘法、平方以及矩阵的行列式就可以了，计算量相当简单（不要求矩阵的逆，因为本文在预计算的二维纹理中存储的就是 $M^{-1}$ ，而不是 $M$ ）。

#### 0.2.4 使用LTC方法来计算多个点光源光照

由式25可得，着色点在多个点光源光照下的辐射照度为：

$$L_o = \sum_{\omega_i} \left( \frac{\omega_s \cdot z}{(\omega_s \cdot \omega_s)^2} |M_s^{-1}| L_i + \frac{\omega_d \cdot z}{(\omega_d \cdot \omega_d)^2} |M_d^{-1}| L_i \right) \quad (26)$$

其中 $\omega_s = M_s^{-1}\omega_i$ 、 $\omega_d = M_d^{-1}\omega_i$ 。因为对同一个着色点来说，对任何点光源 $M_s^{-1}$ 和 $M_d^{-1}$ 是定值，所以：

$$L_o = |M_s^{-1}| \sum_{\omega_i} \frac{\omega_s \cdot z}{(\omega_s \cdot \omega_s)^2} L_i + |M_d^{-1}| \sum_{\omega_i} \frac{\omega_d \cdot z}{(\omega_d \cdot \omega_d)^2} L_i \quad (27)$$

而使用BRDF方式来计算多点光源光照的公式如下：

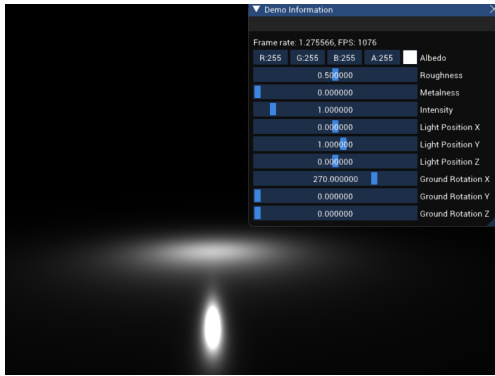
$$L_o = \sum_{\omega_i} \left( f_d + \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)} \right) L_i n \cdot \omega_i \quad (28)$$

其中 $D$ 、 $F$ 、 $G$ 、 $f_d$ 需要分别按照式4、6、7、9来计算。

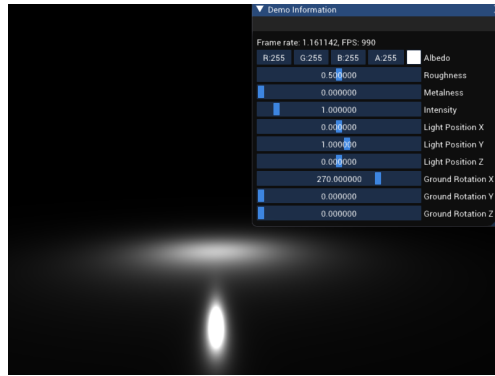
从式27、28可以看出，本文用LTC的方式来实现多点光源光照，计算量比使用BRDF方式要少很多，从而达到本文要实现的目标：以更快的速率实现多点光源下基于物理的光照效果。除此之外，使用本文LTC渲染点光源的方法，不会因为BRDF公式（ $DFG$ 以及 $f_d$ 项）的复杂性而增加渲染时间，也就是说，即使未来BRDF因为要达到更加物理真实的效果而变得更加复杂，使用本文的方法是会降低渲染帧率的。本文算法对于点光源在基于物理的渲染上具有很好的可扩展性。甚至可以让复杂的点光源PBR光照在移动平台上也能快速实现出较好的效果。

### 0.2.5 光源裁剪

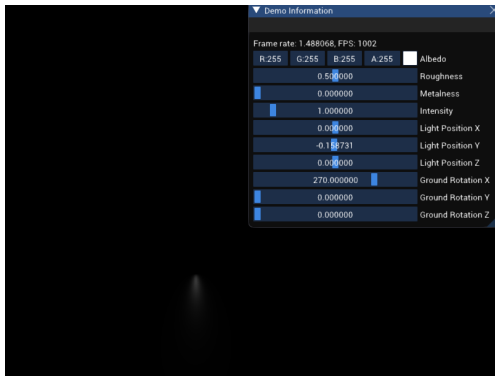
按照上述的方法，当变换前和变换后的光源位置均处于微平面以上时，使用LTC和BRDF方法的效果相似。如图2中(a)和(b)，其中光源初始位置为 $(0, 0, 1)$ ，地面粗糙度为 $\alpha = 0.5$ 。考虑可能会出现的以下两种情况：1) 当点光源 $\omega_1$ 的初始位置于点P所在半球面以下，而在乘上变换矩阵 $M_1^{-1}$ 后，光源位置移动到半球面以上；2) 当点光源 $\omega_2$ 的初始位置于点P所在半球面以上，而在乘上变换矩阵 $M_2^{-1}$ 后，光源位置移动到半球面以下。



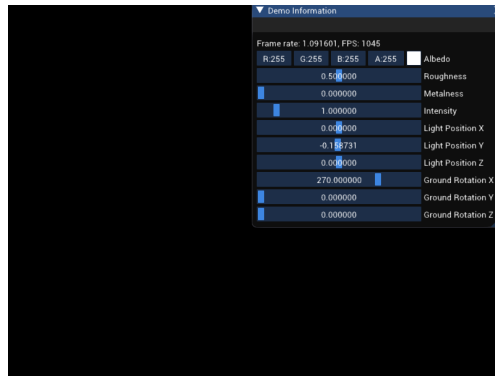
(a) 正常情况下使用LTC方法计算单个点光源



(b) 正常情况下使用BRDF方法计算单个点光源



(c) 情况1下使用LTC方法计算单个点光源



(d) 情况1下使用BRDF方法计算单个点光源

Figure 2: 光源裁剪

按照实际中的真实效果，当点光源位于平面以下时，平面表面应当无任何光照效果。而当出现情况1时，使用两种方法所呈现的效果如图2中(c)和(d)，其中光源初始位置为 $(0, 0, -0.158731)$ ，地面粗糙度为 $\alpha = 0.5$ ，可以看出使用LTC的方法在平面上有明显的“余光”，这是因为虽然 $\omega_1.z < 0$ ，但由式16和21，

$\omega_s.z > 0$ ,  $\omega_d.z > 0$ , 代入式22, 有 $L_o > 0$ 。因此对于这类光源, 我们需要在计算时裁剪去掉。很明显当点光源初始位置在平面以下时, 有

$$\omega_1 \cdot n < 0 \quad (29)$$

对于第2种情况, 当 $\omega_2$ 乘上变换矩阵 $M_2^{-1}$ 后, 光源位置移动到半球面以下, 此时由式16和21, 有 $\omega_s.z < 0$ 和 $\omega_d.z < 0$ , 代入式22, 有 $L_o < 0$ , 不符合真实情况, 因此此类光源也需要进行裁剪。综上, 整合所有情况之后的式25应为:

$$L_o = \begin{cases} 0, & \omega_i \cdot n < 0 \\ \frac{\max(\omega_s.z, 0)}{(\omega_s \cdot \omega_s)^2} |M_s^{-1}| L_i + \frac{\max(\omega_d.z, 0)}{(\omega_d \cdot \omega_d)^2} |M_d^{-1}| L_i, & \omega_i \cdot n > 0 \end{cases} \quad (30)$$

### 0.2.6 实验结果

本文的实验环境是: Intel(R)Xeon(R)CPU E3-1230 V2@3.30GHz, 16GB内存, NVIDIA GeForce GTX 1060 6GB显卡。

在有100个点光源的场景中, 使用BRDF方式以及本文的算法在场景粗糙度 $\alpha = 0.27$ 时所实现的渲染结果如图3所示:

当渲染不同数量的光源时, 使用BRDF方式以及本文算法的帧率和提升的渲染时间如表1所示:

	100个光源	500个光源	1000个光源	5000个光源
BRDF帧率(fps)	118	23	12	2
本文算法帧率(fps)	285	56	28	5
本文算法时间提升(ms)	4.97	25.62	47.62	255.49

Table 1: 本文算法以及BRDF的渲染效率



(a) 使用BRDF方式渲染的结果



(b) 使用本文算法渲染的结果

Figure 3: 使用BRDF和本文算法结果的对比较

## Bibliography

- [1] Takahiro Harada, Jay McKee, and Jason C Yang. Forward+: Bringing deferred lighting to the next level. 2012.
- [2] O. Olsson, M. Billeter, and U. Assarsson. Clustered deferred and forward shading. In *Acm Siggraph*, 2012.
- [3] Abdul Bezrati. Real-time lighting via light linked list. *GPU Pro 360 Guide to Lighting*, page 6, 2018.
- [4] W. Jack Bouknight. A procedure for generation of three-dimensional half-toned computer graphics presentations. *Communications of the Acm*, 13(9):527–536, 1970.
- [5] James F Blinn. Models of light reflection for computer synthesized pictures. In *ACM SIGGRAPH computer graphics*, volume 11, pages 192–198. ACM, 1977.
- [6] Fred E. Nicodemus. Directional reflectance and emissivity of an opaque surface. *Applied Optics*, 4(7):767–773, 1965.
- [7] R. Montes and Carlos Ureña. An overview of brdf models. 2012.
- [8] X Granier and W Heidrich. A simple layered rgb brdf model. *Graphical Models*, 65(4):171–184, 2003.
- [9] S K Nayar and . Oren, M. Visual appearance of matte surfaces. *Science*, 267(5201):1153–1156, 1995.
- [10] Christophe Schlick. An inexpensive brdf model for physically-based rendering. In *Computer graphics forum*, volume 13, pages 233–246. Wiley Online Library, 1994.
- [11] Petr Beckmann and Andre Spizzichino. *The Scattering of Electromagnetic Waves From Rough Surfaces*. 1963.

- 
- [12] Kenneth E Torrance and Ephraim M Sparrow. Theory for off-specular reflection from roughened surfaces. *Josa*, 57(9):1105–1114, 1967.
  - [13] Robert L Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics (TOG)*, 1(1):7–24, 1982.
  - [14] Sylvia C Pont and Jan J Koenderink. Bidirectional reflectance distribution function of specular surfaces with hemispherical pits. *Journal of the Optical Society of America A Optics Image Science and Vision*, 19(12):2456, 2002.
  - [15] Csaba Kelemen and Laszlo Szirmay-Kalos. A microfacet based coupled specular-matte brdf model with importance sampling. In *Eurographics Short Presentations*, volume 2, page 4, 2001.
  - [16] Addy Ngan, Frédo Durand, and Wojciech Matusik. Experimental analysis of brdf models. In *Proc Eurographics Symposium on Rendering*, 2005.
  - [17] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.
  - [18] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Eurographics Symposium on Rendering Techniques, Grenoble, France*, pages 195–206, 2007.
  - [19] Matusik and Wojciech. A data-driven reflectance model. In *Acm Siggraph*, page 759, 2003.
  - [20] Sebastien Lagarde and Charles De Rousiers. Moving frostbite to pbr. *Proc. Physically Based Shading Theory Practice*, 2014.
  - [21] Brent Burley and Walt Disney Animation Studios. Physically-based shading at disney. In *ACM SIGGRAPH*, volume 2012, pages 1–7, 2012.
  - [22] Eric Heitz, Jonathan Dupuy, Stephen Hill, and David Neubelt. Real-time polygonal-light shading with linearly transformed cosines. *Acm Transactions on Graphics*, 35(4):41, 2016.