**Spherical Harmonic Lighting And Light Probe Interpolation**

Author: 练孙鸿

这篇玩意主要参考一篇47页的文献[1]，这是一篇讲了关于球谐光照的很多东西，包括动机、相关基础、相关数学工具、例子、应用等。之后笔者在此基础上介绍了Unity Light Probe，一种基于球谐函数和四面体插值的近似GI模块/系统。

**0 引言(ref. p1-p2)**

球谐光照(Spherical Harmonic Lighting, SH lighting)是一种用于【计算投射到3D模型上、允许我们捕捉(capture)、重新光照(relight)、**实时**展现**全局光照**(Global Illumination)风格的区域光源(area light sources)】的一种技术。这种技术最初是在Siggraph 2002的论文[2]里面提出的，用PRT技术实现极致的写实光照模型。甚至我们如果看一下他论文里面的推导过程，我们可以发现其实它是有着相互联系的技术集合的工具箱。

论文的结果非常的引人注目，用来计算它的代码写起来思路也是很直接的。但是原始的论文对于第一次读的人来说不是那么友好，因为它假定了读者有不少相关的背景知识。所以这篇东西，就给读者介绍更多的背景知识，给出一些原始论文钟“为什么这么做“的一些见解，希望各位能获取到有用信息然后在你自己的游戏里面添加上球谐光照（笔者注：感觉好像挺爽的，似乎是篇科普文。然后我看了一下页数……47页……这怕是有点硬核了）

笔者注：一般来说，球谐光照都用来做低频的环境光照明，反射光和高光比较高频，用球谐函数来编码的话精度不是很够用。然后Light Probe给我的感觉就是**用球谐函数编码的光场(light field)捕捉器**，然后Unity里面的Light Probe系统就是直接进行四面体内光场的插值。

**1 光照计算(Illumination Calculation)(ref. p2-p4)**

最简单的光照模型就是漫反射表面反射模型(diffuse surface reflection model)了，它也被称作是“点积光照”(dot product lighting)，因为光照的强度是要乘个系数的，这系数就是表面法线和当前点到光源位置的单位方向向量。也就是：

这是渲染方程(the rendering equation)的简化，而渲染方程是基于物理建模推算出来的一种“黄金准则”。但问题是，渲染方程并不是那么好解的，最起码在“实时”这个限定条件下是一个很硬核的问题。它是方向（向量）在半球面(hemi-sphere)上的积分：

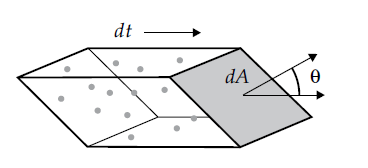
其中：

是从位置出射的的光线强度

是在的自发光

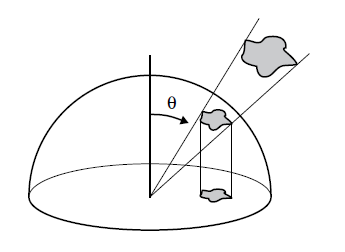
是

想象时间是静止的，继续想象一个充满着光子的空间，每个单位空间都可以定义光子密度(photon density)。我们可以再假设光子是线性运动的（挺合理的，一般情况下光沿直线传播嘛）。渲染的核心就是，找出单位时间里有多少光子撞到静止的表面上，一般用物理量——光通量(light flux)来表示，这个物理量的单位是焦耳/秒(J/s)，或者是说瓦特（W）。



图：投影角(projected angle)的起源——用于计算通量密度(flux density)

要计算通量，我们首先要注意到所有在时间内会撞到表面上的光子其实是分布在一个立体里面的（是一个有着不同朝向的平面的扫掠形状）。很明显当越大的时候，单位面积平面扫掠出来的体积越小，所以就越少光子会撞到平面上。这个撞到单位面积的光子数量其实是跟“表面法线和光流动方向”的夹角余弦值成正比的。



图：投影立体角(projected solid angle)

不过前面都只是随便说了一下关于渲染方程的东西，体会一下就好了。更多的内容还是要自己查阅相关的文献了。RTR，PBR等书都有相关的介绍，然后相关文献也是。而且提出球谐光照的动机之一也是打算要**实时**地去近似求解渲染方程。

**2 蒙特卡洛积分(Monte Carlo Integration)(ref. p4-p7)**

根据渲染方程的启发，我们有了一个目标函数去做积分，这个函数描述了入射光强。但是我们一般情况下没办法知道这个函数的解析式，那看起来只能借助下数值方法求解了。那么我们介绍一种方法：蒙特卡洛积分。（讲着玩意是因为后面求球谐系数的时候需要用蒙特卡洛方法来做数值积分）

接下从一些概率论基础知识开始引入下蒙特卡洛积分：

1. 如果一个变量在它地定义域内都有相同的概率出现，那么这个变量就是均匀随机变量(uniform random variables)
2. 如果一个均匀随机变量取值在之间的话，那么他就是规范随机变量(canonical random variable)。这种变量很适合用来在其他的分布里面进行采样。
3. 概率密度函数(probability density function, PDF)是概率分布函数的导数。服从一定概率密度函数的变量一般记作。
4. 数学期望：
5. 如果用上伯努利的大数定理，我们可以得到：
6. 所以如果我们对积分的话，可以得到：

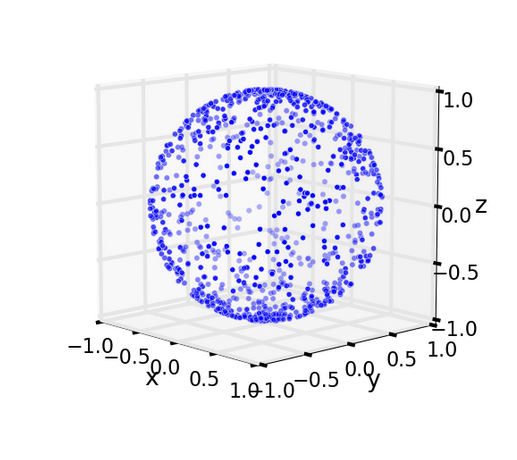
从操作上来讲，我们就需要的大量采样，然后每个采样都要除个概率密度函数值，然后这些scaled过的函数值求和，最后除以采样数，就得到了积分的近似值。

1. 对于Monte-Carlo估计器来说，我们给每个函数值的scale叫做权重，于是：
2. 如果我们可以保证是一个在我们关注的采样空间中的均匀分布，我们就可以再基于7的方法再简化一下：对求和，除以采样数和常量的积，可以省了不少计算。

从渲染方程出发我们知道，我们需要在半球面上进行积分，所以我们就只需要**生成一些在球面上平均分布的点**（更加专业地说应该叫**无偏估计**(unbiased random samples)）。如果我们输入一对独立的变量，那么我们就可以用如下方式把这个正方形区域地随机变量映射到球面坐标：

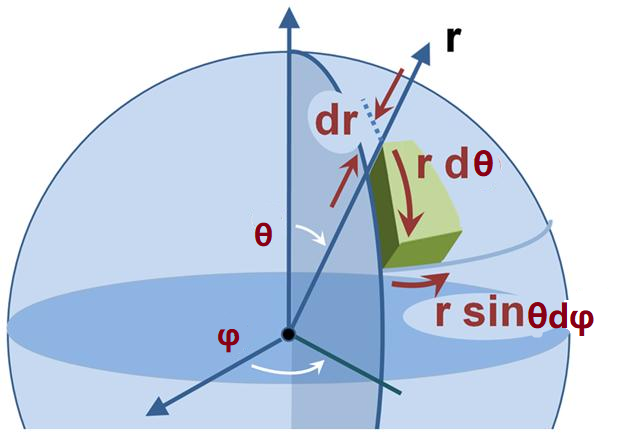
**2\* 球面上的均匀采样(笔者注)**

译者有点疑问：emmm这映射后的能是个均匀随机变量？？？？哦哦好像映射后确实不是笛卡尔坐标系意义的均匀，但是在球面上的分布看起来是均匀，那么这个究竟怎么构造的呢？？？带着疑问google了一下，有一篇文章挺好的[3]。文章前面说了一下，如果直接让当成是一个二维独立均匀随机变量的话，球面两极的点将会很密集：



这是因为单位面积微元不是一个常量，而是跟有关。那么很明显，在靠近处面积微元比较小，然后我们又是想让点集均匀分布在球面上，所以南北极两个极点处的概率密度还是得小一点好。也就是说，如果我们直接：

的话，无论你的纬线的周长是多少，都有同样概率分到同样多的点，所以每条纬线(相同)的密度都是不一样的，这样子会导致越靠近两极，点分布越密集。我们要想办法映射一下，让两极不那么密集，赤道处不要那么疏。



因为球面的表面积，我们又想让概率密度函数在球面上为常数，即，所以我们令：

因为在单位球上r=1，所以：

结合上面两条有：

我们求这个二维独立随机变量的概率密度函数的边缘分布函数：

能看出来，如果我们想要让点在球面上分布均匀，则高纬度的纬线（靠近）的概率密度需要小一点，直观上感受是对的。**我们现在就需要一种数学工具，从一个均匀分布(uniform distributed)变量+一个变换，来生成给定分布的概率分布函数(CDF, Cumulative Distribution Function)**。

文章中给出了一个数学工具，叫**Inverse Transform Sampling**[4]。这玩意的意思是说，可以给定一个均匀分布变量，再给定一个我们想要生成的连续的随机变量以及它理想的分布函数，我们可以推导出：

具体推导直接看维基吧。用这个数学工具我们可以用均匀分布随机变量+给定的CDF的**反函数**来构造一个随机变量，使得这个构造的变量服从我们给定的CDF。中文维基“累积分布函数”下的”反函数”条目[5]也提到这个操作。

我们回到球面均匀采样。因为，的变换就很好构造，先不管了。而是个变量，所以变量的分布函数要搞一下了。

因为：

我们要用上面说的Inverse Transform Sampling来构造出的变换使得

。那么这个数学工具最关键一步就是求出的反函数。那么很明显的：

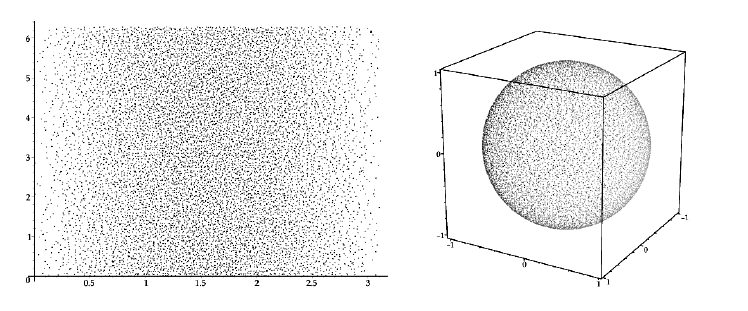
译者又来注了：虽然《SH Lighiting: the Gritty Details》原论文写的变换是：

但是我发现………………在Google上搜索公式并画出图来发现………..**这两东西居然是同一个东西**…………….好吧。因为：

参考反三角函数运算公式[6]，可以推得：

所以论文和讲球面均匀采样的网页说的东西是一样的。所以讲多了那么多东西，只是为了说明下面两个变换都能生成球面均匀分布的点：

所以现在，我们终于做到了在单位球表面上均匀采样了 :) 。

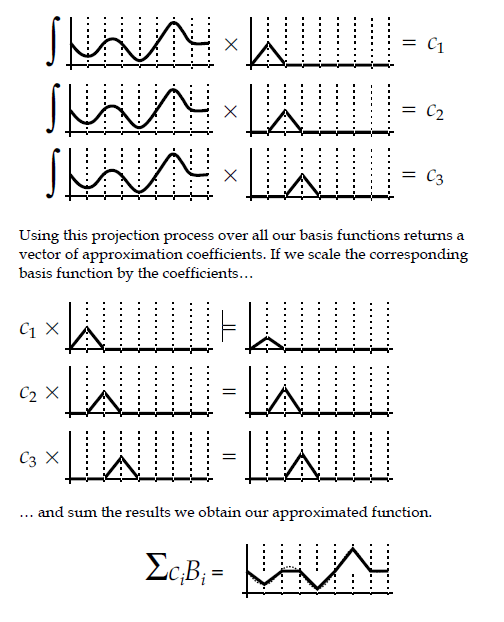


PS:有个方法可以减少Monte-Carlo随机采样的方差，就是[1]**分层采样(stratified sampling)**。意思是把二维随机变量的定义域矩形区域分成个区域，然后每个区域采样一个点。这样做的方差是可以被证明是不大于直接在全定义域里面采样的。

PPS：有一些关于蒙特卡洛光线跟踪的文献[7,8,9]可以参考一下。

**3 正交基函数(Orthogonal Basis Function)(ref. p8-p11 )**

这篇球谐光照的论文用了**基函数(basis function)**的概念。不太严谨地说，基函数是一小段信号（函数），它可以被用来缩放、叠加，然后生成一个对原函数的近似(approximation)。求出每个基函数的缩放系数，或者说scale coefficient，的过程叫做投影(projection)。（笔者注：傅立叶变换其实就是一种”投影”，其中基函数就是谐波）。基函数一般都是**正交函数(orthogonal function)**，正交函数在函数空间里，在给定定义域上内积为0（笔者注：函数的内积定义起来其实很像一个多维向量的内积，就是对应分量相乘然后求和）：



原论文给出的这个例子里面，是基函数，是基函数的系数，原函数经过投影得到系数，后面我们又可以通过来近似重构出。

在上面的例子里面是用了**线性基函数(linear basis function)**（可以分段线性近似出原函数）。其实我们还有很多的基函数可以选择，例如，还有其他可以在给定定义域内满足定义的都是正交函数，例如油管上有个小教程[10]给出的例子是在是正交的。

有一族正交函数是数学领域比较关注的，就是**正交多项式(Orthogonal Polynomial)**。正交多项式有一些奇妙(intriguing，这词有点fancy) 的特性（或者说其实这个应该是它的定义吧），这个特性跟正交基函数族类似：

甚至严格一点，如果要求，那么这些正交函数就是**标准正交(orthonormal)**的了。有很多这种正交多项式，例如Chebyshev Polynomial，Jacobi Polynomial，Hermite Polynomial等等。在这篇讲球谐光照的论文里面，我们最关心的就是**勒让德多项式(Legendre Polynomial)**，特别是**伴随勒让德多项式(Associated Legendre Polynomial)**。根据维基百科的说法[11]，勒让德函数是勒让德微分方程(Legendre Differential Equation)的解：

勒让德方程是物理和工程领域里面常常遇到的一类常微分方程，当**试图在球坐标中求解三维拉普拉斯方程**（或者其他偏微分方程的时），问题经常会归结为勒让德方程的求解。当方程满足时，可以得到有界解（解级数收敛）。并且当时，时也有有界解。在这种情况下，方程的解随着值变化而变化，构成的一组由**正交多项式(orthogonal polynomial)**组成的多项式序列，称为勒让德多项式：

伴随勒让德多项式通常记为，有两个参数，，都是整数。伴随勒让德多项式最直观的定义，是基于普通勒让德多项式的导数：

* 注意1:英文维基和中文维基的居然是不一样的，英文维基[12]打了中文维基[13]的脸，说”Thefactor in this formula is known as the [Condon–Shortley phase](https://en.wikipedia.org/wiki/Spherical_harmonics#Condon–Shortley_phase)[14]. Some authors omit it.”。这应该是约定上的一些问题吧。
* 注意2:普通的勒让德多项式可以有复数项，伴随勒让德多项式则都是实数项。
* 注意3:伴随勒让德多项式的定义域是，

伴随勒让德多项式的参数把这一族多项式拆分成为了多个函数“带”(band)，其中是**band index**。取值是。伴随勒让德多项式的每一条带里面的多项式都是正交的（当然不同band的伴随勒让德多项式也是正交的，毕竟是一族正交多项式，但是不同band的函数求内积得到的常量不一样）。个band有个系数：

(\*注：系数的数量有点奇怪，原文是，get不是很到。其实好像时候也是有定义的[12])

虽然从前面看来，公式什么的都有了，但是其实勒让德多项式求解起来还是有一点复杂度的，所以它就很少被用来近似一个1D函数了。求解的时候我们其实不应该直接从定义出发来计算，而是用递推关系来计算。有几条有用的递推式：

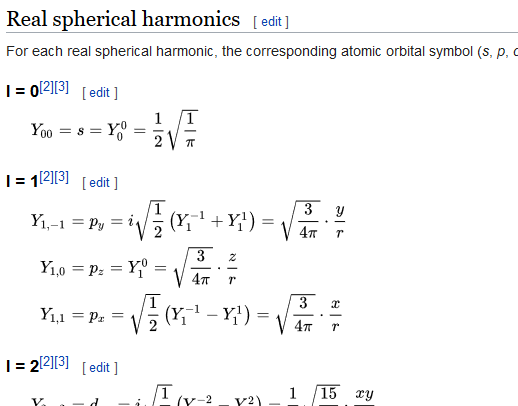
1.

2.

3.

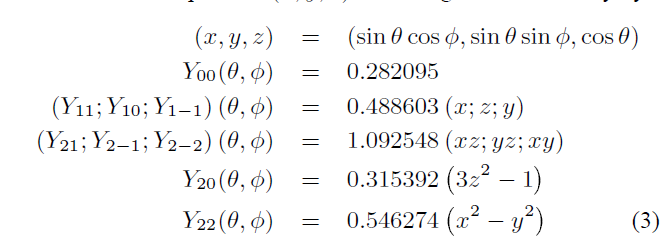
具体的计算还是观摩一下原论文吧。（笔者注：其实维基百科[12]Associated Legendre Polynomial也给出了前面很多项，看到目前为止我感觉如果球谐球谐光照不用太多系数的话，完全可以先预先手算出ALP前n项的值，hhhhh。）

笔者注Again：参考维基百科[17]，里面把实球谐函数的前五个band的表达式全部展开来了，伴随勒让德多项式都已经塞进硬编码的常数里面了。而且一般球谐光照的应用也就用3~4个band，所以**球谐函数完全可以hardcode，没有毛病**。



图：维基百科的实球谐函数展开式

在文献[18]里面，球谐系数的项都已经hardcode成浮点数了：



**4 球谐/球面调和(Spherical Harmonics)(ref. p11-p18)**

前面说的用Associated Legendre Polynomial的正交多项式用来投影和拟合1D函数还行，但是如果要把他用在2D平面甚至球面上就有点麻烦了。Associated Legendre Polynomial是球谐(Spherical Harmonics)的核心，其中**球谐是一种很像傅立叶变换(Fourier Transform)的数学系统**，但是球谐是定义在球面上的。（下面球谐就简写为SH）。SH函数在通用情况下是在复数的基础上定义的，但是我们只关心定义在球面的实函数（在球谐光照里面，这个实函数就是光强场(light intensity field)），所以这篇文章就只关心实球面调和(Real Spherical Harmonics)。

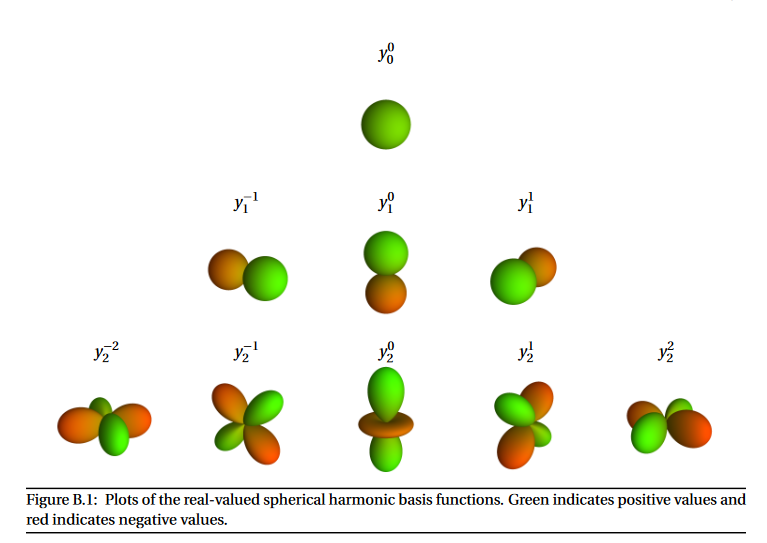
先给出单位球上坐标的标准参数化：

传统上来讲，SH函数记为，表达式是：

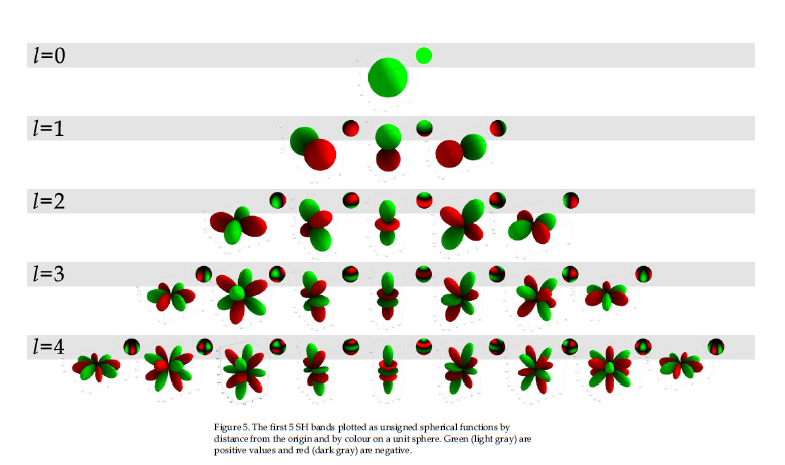
其中还是前文讲了很久的伴随勒让德多项式。这个突然冒出的只是一个缩放系数，用来归一化这个函数：

为了生成所有的SH函数，参数的定义域伴随勒让德多项式有点不一样：

而且有的时候，把SH函数的系数展开成1D向量（变成个线性表）是挺有用的，所以我们就可以定义一个序列：



看上图[15]可知,刚刚好就是中间列的项，下标从0开始。



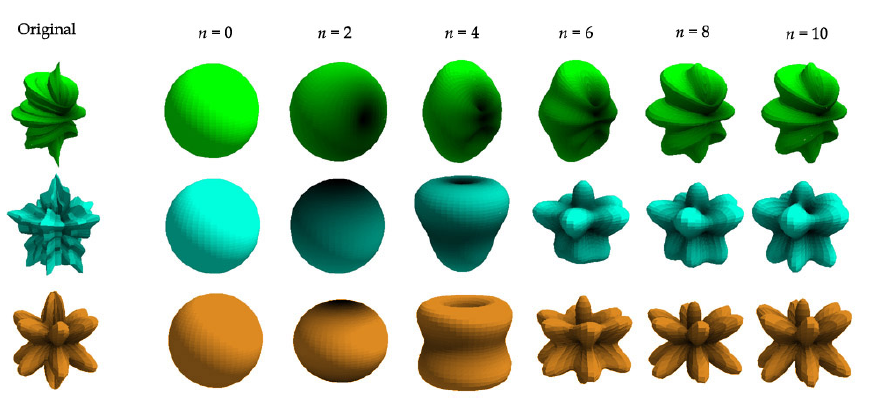
很多论文都会直接把球谐函数的多项式扔给你，没有可视化，非常抽象。上面的图就给出了前五个band的球谐函数的可视化结果。绿色是正值，红色是负值，离中心越远的地方绝对值就越大。（笔者注：那么跟傅立叶变换对比下，傅立叶变换的基底是，可以把1D函数投影上来。而球谐”变换”的基底就是，可以把定义在球面上的2D函数投影上来。实际上我还搜到一个讲得很简介但是好像视角很高的资料[16]，里面说**定义在球面上的函数可以用球谐函数展开成二重广义傅立叶级数**。在球面上的展开式为：

其中用球谐函数进行展开的系数序列计算公式如下（原论文中的未展开版本）：

这个形式跟做1D傅立叶变换时候的卷积贼像哇（或者说是内积会不会好一点）！然后[16]中的系数计算公式直接参数化了：

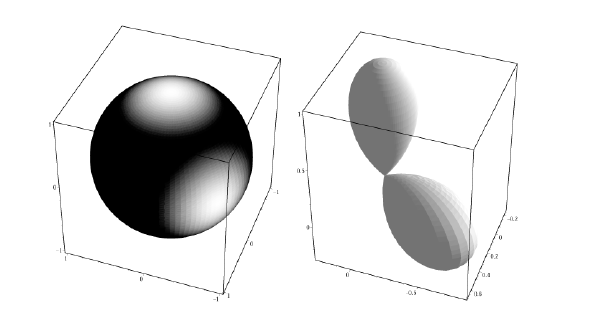
在实际操作里面，按照上面提到的展开式就可以重构出原函数。但是在实际操作中，我们写程序时不可能会有对无穷级数进行储存和卷积的操作，一般展开项只能是有限项，也就是：

其中是球谐基band的数量，显然个band的球谐基数量是个。个band的近似我们就叫做是**n阶近似**(n-th order approximation)吧。因为球谐基的项数是有限的，所以我们只能用球谐基和球谐系数**近似**重构原函数，更准确的说，这个过程是一个**带限(band limited)近似**。在这个语境下，因为频域信号带宽的限制，**大于一定阈值的高频信号就被去掉了**。



从图中可以看出，球谐展开阶数越高，能重构出来的信号就越精确。

之后原论文举了一个简单的例子来说明如何用Monte-Carlo积分来进行基于球谐函数展开和重构的例子。



原文举的例子是，两个大单色光源照到一个点上，那点的光照强度函数是：

然后我们想用球谐基和球谐系数来近似出，所以我们现在要求球谐展开的系数：

但是这种形式化的积分描述比较难实操的，一般写程序的话就要数值地去求解这个投影到球谐基上的系数。回忆一下前文提到的**蒙特卡洛积分估计器**：

在这个语境下，设是预先计算的采样方向向量，那么蒙特卡洛积分的被积函数是：

每个采样的权重是：

所以球谐系数表达式是：

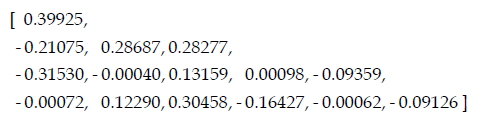
谐基的ht field)

在写代码实现的时候，用蒙特卡洛积分求系数的过程就是一系列的相乘与求和。伪代码：

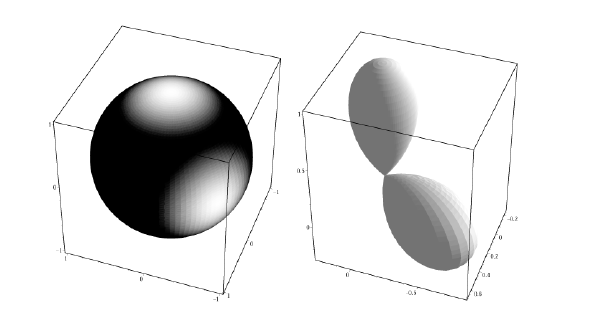
|  |
| --- |
| void SH\_Coefficients()  {  double weight =4.0 \* PI;  //生成n条光线进行采样  for(int i=0; i<n\_samples; ++i)  {  生成带抖动的无偏采样方向(θ,ϕ)  for(int n=0; n<n\_coeff; ++n)  {  //对于某一个light probe，它的每个球谐展开系数c\_i就要累加起所有的【某方向上的irradiance \* 这个方向上SH函数值】  result[n] += light(θ,ϕ)\* samples[i].SH\_basis\_coeff[n];  }  }  // 把蒙特卡洛积分的常数项乘上去（恒定的采样权重，总采样数）  double factor = weight / n\_samples;  for(i=0; i<n\_coeff; ++i)  {  result[i] = result[i] \* factor;  }  } |

上面的result[i]就是某个点上光照情况经过”编码”之后得到的球谐系数。在实时渲染的时候，我们想要从离线处理得到的SH系数重新实时计算出辐照度(irradiance)，我们就只需要对【SH系数\*给定入射方向的SH函数值】求个和就好了：

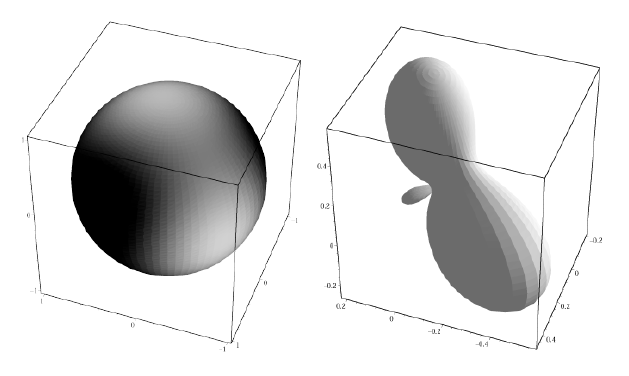
原论文[1]中举得例子求出的SH展开系数的画风是这样的（原文的系数没列齐= =4 band球谐光照应该有16个系数的，但是它只写了15个hhhhh）



图：原文例子的球谐展开系数（band[3]少了一个系数，额）



图：原本的光照信号



图：4-band球谐重构出的光照信号

这个4-band，16系数重构出的光照看起来还勉强ok，毕竟只有16个系数哇（虽然有些实际操作场景里面甚至只用1~3个band）。重构的光照有个突起的小丁丁，是因为前文例子里面用了max()函数，所以有一定的截断和突变，从而造成高频信息。而系数少的Spherical Harmonics只能保存和重构出低频的信号，所以有些失真没办法了。



图：[19]里面从Cube Map生成球谐系数后重建的结果，前面分别是0~5级球谐展开的信号重建结果。

**5 球谐函数的性质(ref. p18-p20)**

[1]球谐函数有很多有趣性质，所以它比其他的基函数更加适合用于编码定义在球面上的光照函数。

* **球谐函数的标准正交性**

首先，球谐函数不只是正交的，它还是标准正交(orthonormal)的。这意味着如果。（函数正交的卷积定义）

* **球谐函数的旋转不变性**

其次，球谐函数还是有**旋转不变**(rotationally invariant)的。这个意思是，如果我们有旋转操作，原函数是，旋转后函数是，即：

那么：

换句话说，旋转函数本身，和旋转输入，输出的结果都应该是一样的。这个性质看起来没什么用，但是看一下JPEG的离散余弦变换(DCT)，DCT不是平移不变的，所以在高压缩率下JPEG图像会产生很明显的方块感。那么我们关注回球谐函数，因为球谐函数有旋转不变性，所以在我们旋转物体、灯光的时候，光强不会产生涨落、抖动、挪移等不良(objectionable)现象。

* **（重要）函数乘积的积分等于其球谐系数向量的点积！**

SH函数的下一个特性是杀手级别的牛逼！（哦）。在我们做光照的时候，通常情况下我们需要某种形式的入射光描述、某种形式描述的表面反射（我们叫这个描述表面反射的函数叫**传输函数**(transfer function)，据说可以是BRDF之类的reflection descriptor function），把它们相乘，来得到结果的反射光。但是我们还是需要对整个半球面的入射光进行积分。也就是我们要计算：

其中是入射光，是传输函数。现在有一个超级流批又神奇的特性：**两个函数的乘积的积分等于两组他们的球谐系数向量的点积** 。设和分别是的球谐近似，分别是的球谐投影系数，则球谐函数的标准正交性保证了如下特性[1][2]：

这个牛逼特性的一种解释角度是，[20]能量在不同域上通过不同的运算来达到守恒。

（笔者注(2018.8.10)：这个虽然按定义来看似乎好像有点自然，但是观感上又很神奇。按理来说求某点的反射光是要做一波半球积分的，然而却不需要用Monte-Carlo积分，而是直接一波球谐系数的点积？？！这个想一想，感觉可能有点道理，因为在把原函数投影到SH basis上的过程已经对整个球面做过Monte-Carlo积分了，然后SH系数已经算是包含了球面积分的信息在里面了）

这个特性是非常关键的，因为我们把一个球面积分坍缩成了维向量的点积，而一般就是3或者4而已。这一步是实现**实时环境光照明**的重要一环。通过把函数投影到球谐空间(Spherical Harmonic Space)我们可以把球面积分转化为很快的操作，以至于可以实时计算 ！！**（笔者注：emmm那么传输函数又怎么定义呢orz真是螺旋爆炸跪）**

其实还有其他的技巧可以用于变换一个球谐函数。假设我们现在有一个未知的、任意的球面光源函数，然后我们又有一个阴影的描述函数。我们现在想要把入射光的SH系数映射到另一组已经被阴影函数遮罩(masked)过的SH系数，结果记为。我们可以构造一个能把”的SH投影映射到的SH投影”的线性操作。这个线性操作可以在不知道的情况下用**转移矩阵(transfer matrix)**来表示。构造出这个矩阵的计算：

**（笔者注:我又跪了orz，）这个积分究竟要怎么求啊？每个元素都要求一波三重积？)**

这个矩阵可以让我们把一组普通的光源SH系 变换成为一组添加了阴影的光源的SH系数，具体操作起来就是：

最终生成的转移矩阵一般都会比较稀疏。

转移矩阵乘上SH系数向量会得到另一个SH系数向量，而这个结果SH系数向量描述的光照情况，就跟投影到SH basis前原光照\*阴影遮罩产生的结果一样！



图：把一个转移矩阵施加到一个球谐系数向量上

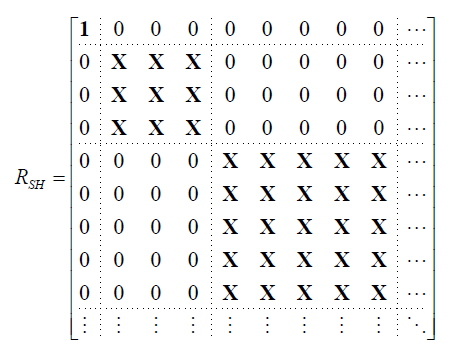
所以转移矩阵算是另外一种储存阴影的方法了。

**6 旋转球谐函数(ref. p20-p26)**

（笔者注:省略了非常多，因为不知道讲这么多关于SH Function的旋转来干嘛。所以后面主要是加了点自己的补充）

我们之前提到球谐函数是旋转不变的，但是我们究竟要怎么旋转一个已经进行了球谐投影的函数呢？答案并不是平凡的。第一个问题：我们究竟要用什么形式的旋转呢？如果是用欧拉角，那么又是用什么顺规呢？XYZ？ZYX？ZYZ？如果如果用四元数(Quaternion)来定义旋转呢？…..

从SH函数的正交性出发，我们可以推出SH旋转过程是一个线性操作，而且不同band之间的系数之间是不会有交互的。原文说用一个的旋转矩阵来做这一个操作。这个矩阵将会是“块对角稀疏”的(block diagonal sparse)。



旋转的过程确实算是一个比较简单的计算，但是构造这个旋转矩阵的系数就没这么trivial了。我们现在先把SH函数的球面坐标转换为笛卡尔三维坐标，然后我们意外的发现，前两个band展开之后的笛卡尔坐标表达式还挺简单的：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

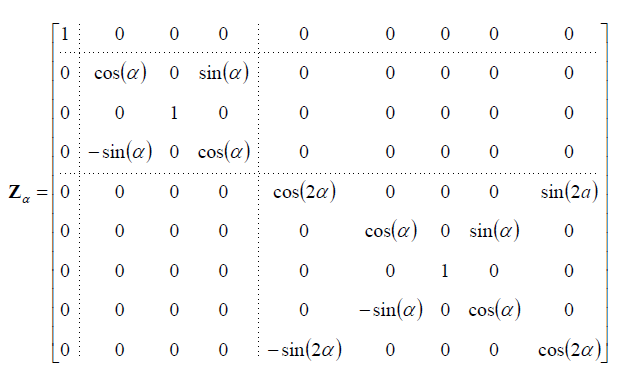
其中一般情况下。

在使用的时候，我们就在单位球上选择一个点，然后代入特定SH的参数求出SH函数的值。（不过其实笛卡尔坐标表达不是很方便做SH投影，反而形式化的表达式对我们更有用）。

我们再回忆一下前面说的转移矩阵，我们在那个基础上再加上旋转的元素。在形式化的表达下，我们先对每个球谐函数采样SH sample进行旋转之后再积分：

上面形式化表达式就可以构造一个的矩阵来把球谐系数映射成旋转后的球谐系数了。如果改用显示参数化的公式的话，举个例子：

例如前三band的绕z轴旋转的9x9矩阵：



当矩阵扩张到更加高的band的时候，第个band对应的矩阵分块就会用到。（笔者注：这这这，这旋转矩阵的构造一点都不trivial啊，居然会有用到倍角这种东西）。

用旋转矩阵的方法在SH函数低阶的时候还好，但是在实际操作中，阶数一高就很让人头疼了。一个是顺规与最少所需旋转数，和由此带来的大计算量。（不说了，看原文吧，因为他讲很多我还没看出来用途orz）

…….

唉还是来讲点科普的吧。球谐函数最开始是设计出来描述单原子在量子尺度的角动量分布（这能一定程度上解释了参数和为什么是整数，因为他们是用来描述**原子状态的不可再分的量子数**）。在科学研究里面球谐函数用的最多是计算化学(Computational Chemistry)，计算化学家们在量子层面对原子进行建模来研究化学反应。直到1999年，有一篇基础研究的论文才提出了用**Wigner D-函数**来更加高效地旋转一个球谐函数。但是对于计算机图形学来说，我们更加需要的是一组递推关系来把low band index的旋转矩阵转换到higher band index的旋转矩阵。因为low index的band的旋转矩阵会更加好算一点。原文给出了几篇关于SH函数旋转的文献[21][22][23]。

**6\* 为什么要旋转球谐函数啊？（笔者注）**

因为对球谐函数的旋转的动机有点疑惑，所以笔者还是再查了一下文献。[24]里面说，球谐函数的旋转在实时光照和全局光照(GI)里面是个很重要的算法。在某些应用里面，需要做per-vertex乃至per-pixel的球谐函数旋转操作，这意味着快速的球谐函数旋转非常重要了。因为在这篇文章(2006)之前的SH function旋转算法都不太能做到实时。所以[24]就给出了球谐函数小角度旋转的近似求解算法。大概就是用了球谐函数旋转矩阵的截断的泰勒展开式。（Emmm感觉不一定好呃，正交性之类的怎么办喔）。所以我就只从这篇文里面抄点球谐函数旋转的意义、动机。

在用球面基函数(spherical basis functions)的时候（例如Spherical Harmonics或者wavelets小波）来实现有环境光照明的实时渲染时，我们需要面对一个问题：把环境光照明（在世界坐标系下）对齐到反射率函数或者BRDF（在TBN切线空间下）。这个对齐就需要用旋转了。（意思是要把BRDF的SH coefficient通过旋转转换到世界坐标系？这样就能跟入射光照的SH coefficient处在同一空间下了？）。[24]说那篇SIGGRAPH 2002的论文[2]里面说，球谐函数的旋转就是环境光照明里面的性能瓶颈。还有篇[25]东西是把BRDF储存多几份，预先把TBN空间的BRDF预旋转几份，不过这样又很费带宽，而且还不能使用高频BRDF和各向异性BRDF。而且如果使用了法线贴图(normal map)来调制(modulate)模型表面法线的话，甚至需要逐像素的SH rotation。总之据说就是球谐函数的旋转就是Realtime Indirect Lighting的一个性能瓶颈。

球谐函数旋转的问题定义是这样的[26]：给定一个球谐系数向量，它表示的球面函数，现在要找出另外一个球谐系数向量来表示旋转后的函数，其中就是那个旋转操作（线性操作的话，旋转自变量和反向旋转函数本身一样）。

于是就跟前文提到的一样，旋转一个SH function是一个线性的操作，所以可以用旋转矩阵来实现。然后[24]还做了矩阵的三阶泰勒展开。至于具体的着色算法，下文会引用[1][26]来讲。

**7 基于球谐的漫反射光照计算(ref. p26-p )**

前面说了这么多，包括函数的球谐分解、球谐函数的旋转、球谐函数的性质等等，我们终于可以给我们的模型计算一点光照了。现在[2]会讲好几种光照技术，然后看看他们分别是怎么实现的。**但是具体的渲染和着色甚至还得留到下一章！**

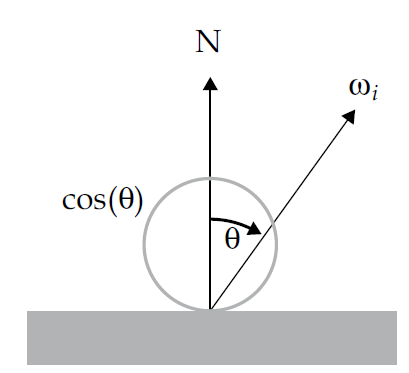
假设我们现在已经加载好了多边形模型到内存了，我们暂时只关注模型的顶点、法线、三角形列表。第一步我们先做一下预处理，把同样位置但是有不同法线的顶点分裂开来，以及焊接顶点(weld vertices)。接下来，我们要**给所有的顶点([1]原文中所谓的”lighting points”)计算传输函数(transfer function)的SH projection**。

[26]里面说，要算的传输函数可以是BRDF，然后需要计算的SH projection是spatially-varied的，这个spatially varied的参数是什么都行，那么模型顶点的index也可以看作是一个spatially varied的参数嘛。然后**BRDF和光照的SH系数都是需要求出来的（但是这些计算究竟是offline还是on-the-fly，还要再斟酌研究一下）**

前文没解释传输函数现在终于说一下了。传输函数是用来跟入射光做点积的向量函数(?)的，点积的结果就是近似的光照强度。对于漫反射表面，我们现在介绍三种不同种类的传输函数。

* **7.1 无阴影的漫反射传输**

我们回到渲染方程，我们把它剥剩最起码的核心元素，就一个光源和接受光源的某个平面上的表面点，然后用没有inter-reflection和没有阴影的直接光照。



其中：

:沿着方向出射的光(flux还是intensity啊)

:处的BRDF

:沿着方向入射的光(flux还是intensity啊)

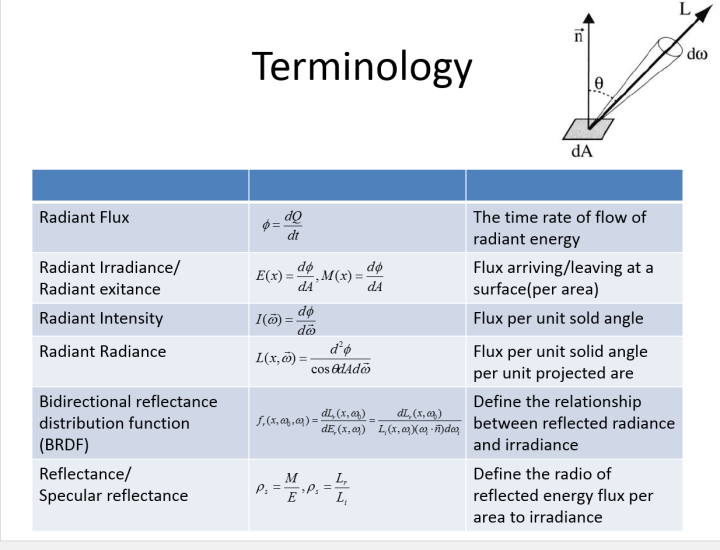
:几何项，或者说是公式里的余弦项

漫反射表面的BRDF在各个方向上的反射光都是相等的，所以这条式子可以大幅度的简化。漫反射表面的光照是view-dependent的，然后因为各个方向的反射光相等，所以也可以消去了。所以我们可以得出Diffuse Unshadowed的光照方程是：

其中是入射光，是在点处的表面反照率(漫反射系数)(surface albedo)，

是在点处的表面法线。（在这里我们的用反照率albedo作为衡量Lambertian漫反射里面(出射radiance/入射irradiance)的BRDF）

笔者注：这BRDF和radiance、irrdiance等等各种玩意的量纲我果然还是没记得清楚，贴个图：



所以这条式子的每个元素的定义是这样的：

**这里笔者有点很傻的小疑问，这个“入射光”“出射光”的量纲究竟是什么？就是”光照强度”吗？**

Anyway，我们把渲染方程的光源和**传输函数**分离开来，记没有阴影影响的渲染方程的传输函数为，这个传输函数就是很之前提到过的，把他投影到球谐基上之后得到的SH系数向量就可以跟入射光的SH系数向量做点乘得到最终光照信息。于是在这个最简单的情形下，传输函数就只是那个几何余弦项（）：

所以为了计算传输函数的SH投影的系数向量，我们又要掏出我们之前生成的在球面上随机均匀分布的射线向量。因为入射光会受表面的反照率(albedo)影响，我们要分开R,G,B三个通道来分别计算光照函数的SH投影。

|  |
| --- |
| //对于所有的pre-calculated的球面均匀分布的随机方向射线  for(int i=0;i<n\_samples;++i)  {  //计算当前射线sample的cosine  double H = Dot(射线采样的vec3，法线);  //如果射线在上半球...  if(H > 0.0)  {  //那么就投影到SH space  // SH project over all bands into the sum vector  for(int j=0; j<n\_coeff; ++j)  {  //这个sample[i].coeff[j]在'gritty detail'p7 Monte-Carlo Integration这一节最后的地方第一次出现（超级坑），储存预计算好的这个方向向量的第i个球谐函数值。value应该是传输函数t(s)乘上了SH函数值，并求和，所以是把传输函数t(s)投影到了SH space了。(话说顺便把albedo乘上了是什么意思？哦哦哦BRDF应该考虑上表面的入射光的吸收率嘛)  value = H \* sample[i].coeff[j];  result[j + red\_offset ] += albedo\_red \* value;  result[j + green\_offset] += albedo\_green \* value;  result[j + blue\_offset ] += albedo\_blue \* value;  }  }  else  {  //不在上半球的射线就不管了  }  }  // 把上面求的和乘以(权重/采样数)，求出BRDF的球谐投影的系数向量  double factor = area / n\_samples;  for(i=0; i<3\*n\_coeff; ++i)  {  coeff[i] = result[i] \* factor;  } |

上面就是计算单点BRDF的SH projection 系数向量的伪代码。看样子，网格的“lighting points”的传输函数的SH projection只能离线预计算了（而且这玩意还是定义在TBN空间里的）。至于要怎么让物体或者光照可以动起来，我们还是要拭目以待= =



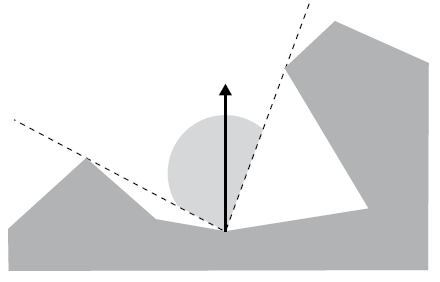
图：Diffuse Unshadowed的着色效果。但是光照用的是任意的基于球谐的区域光源！

* **7.2 有阴影的漫反射传输**

我们基于上一节的Unshadowed Diffuse渲染方程，加多一个可见性项(visibility term)：

其中是可见性测试函数，如果一定长度的射线被物体本身挡住了，就返回0，否则就1。

在给我们简化版的光照方程乘多了一个项之后，物体表面上的点就不再假定周围没有阻碍了，而是会跟当前点周围的几何体有交互，以及入射区域光源的fov。那么在这种情况下，传输函数就是：



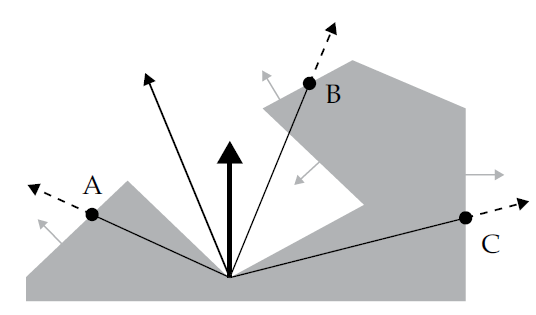
（译者注：看起来这个可见性项只考虑了AO的样子哇- -）

那么加了一个可见性项之后产生新的效果一般叫做Ambient Occlusion，AO这个特性是传统的Local Illumination和Global Illumination的一个重大区别。但是计算这一个传输函数就需要我们从当前点去跟踪光线，看看有没有和场景里的其他的多边形有碰撞。

|  |
| --- |
| //对于所有的pre-calculated的球面均匀分布的随机方向射线  for(int i=0; i<n\_samples; ++i)  {  //如果射线在上半球...  Hs = DotProduct(sample[i].vec, normal);  if(Hs > 0.0)  {  //如果当前pos发出的射线没有跟其他东西相交  if(!self\_shadow(pos,sample[i].vec))  {  //没有hit到其他地方的射线就对最终光照信息有贡献  for(int j=0; j<n\_coeff; ++j)  {  value = Hs \* sample[i].coeff[j];  result[j + red\_offset ] += albedo\_red \* value;  result[j + green\_offset] += albedo\_green \* value;  result[j + blue\_offset ] += albedo\_blue \* value;  }  }  else  {  //发出的射线hit到自己（？）了  }  }  else  {  //不在上半球的射线就不管了  }  }  // 把上面求的和乘以(权重/采样数)，求出BRDF的球谐投影的系数向量  double factor = area / n\_samples;  for(i=0; i<3\*n\_coeff; ++i)  {  coeff[i] = result[i] \* factor;  } |

伪代码里面的self\_shadow()可以根据自己的需求用ray-tracing来实现。然后下面讲一些实现这一个SH projection的一些注意事项。

1. 如果你用了求交的加速数据结构如体素网格(voxel grid)、层级包围盒树(hierarchical bounding box tree)、BSP(binary space partitioning) tree, 那么其实不需要做一次初始的ray-box intersection了，因为射线的原点肯定在包围盒的内部。
2. self\_shadow()函数的目的是找出被遮蔽的光线，所以我们会从模型**上**的某个位置发出各个方向的射线。问题就是，这些射线很有可能会跟临近的多边形共面，那这么一来，如果不谨慎处理的话，求交结果就会直接得到射线起点。注意排除这样的情况。
3. 要记得做单面多边形检测。



图：ray-tracing的时候很可能有这种坑爹的情况



图：用5阶shadowed diffuse球谐传输函数渲染出来的效果（感觉这AO好夸张2333）

* **7.3 有阴影与相互反射(Inter-reflection)的漫反射传输**

最后一个漫反射光照方式是最流批的。比起最trivial的local illumination，渲染方程里面的入射光可以不只是光源的直接入射光，还可以是**从其他物体上二次反射过来的光**。如果加上这种次反射光，我们的渲染方程就变成了：

其中：

是上一节7.2讲的diffuse shadowed函数

是上一节7.2提到的可见性测试函数（返回0和1）

是从其他位置反射过来的光

**引用**

[1] Green R. Spherical harmonic lighting: The gritty details[C]// Game Developers Conference. 2003.

[2] Sloan P P, Kautz J, Snyder J. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments[C]// Conference on Computer Graphics & Interactive Techniques. ACM, 2002:527-536.

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

[4] <https://en.wikipedia.org/wiki/Inverse_transform_sampling>

[5] <https://zh.wikipedia.org/wiki/%E7%B4%AF%E7%A7%AF%E5%88%86%E5%B8%83%E5%87%BD%E6%95%B0>

[6] <https://zh.wikipedia.org/wiki/%E5%8F%8D%E4%B8%89%E8%A7%92%E5%87%BD%E6%95%B0>

[7] Siggraph 2001, “State of the Art in Monte Carlo Ray Tracing”,Course 29

[8] Peter Shirley,”Realistic Ray Tracing”, A. K. Peters, 2001

[9] Design of a Realistic Image Synthesis System

[10] <https://www.youtube.com/watch?v=8ZyeHtgMBjk>

[11] <https://zh.wikipedia.org/wiki/%E5%8B%92%E8%AE%A9%E5%BE%B7%E5%A4%9A%E9%A1%B9%E5%BC%8F>

[12] <https://en.wikipedia.org/wiki/Associated_Legendre_polynomials>

[13] <https://zh.wikipedia.org/wiki/%E4%BC%B4%E9%9A%8F%E5%8B%92%E8%AE%A9%E5%BE%B7%E5%A4%9A%E9%A1%B9%E5%BC%8F>

[14] <https://en.wikipedia.org/wiki/Spherical_harmonics#Condon%E2%80%93Shortley_phase>

[15] <https://cs.dartmouth.edu/~wjarosz/publications/dissertation/appendixB.pdf>

[16] <http://www.wlxt.uestc.edu.cn/wlxt/ncourse/Mathematic/web/content/wlkc/wlkc-19-05.htm>

[17] <https://en.wikipedia.org/wiki/Table_of_spherical_harmonics>

[18] Ramamoorthi R. An efficient representation for irradiance environment maps[C]// Conference on Computer Graphics & Interactive Techniques. 2001:497-500.

[19] Sloan P P. Stupid Spherical Harmonics (SH) Tricks[C]// 2010.

[20] <http://imgtec.eetrend.com/blog/7360>

[21] Ivanic J and Ruedenberg K, “*Rotation Matrices for Real Spherical Harmonics, Direct Determination by Recursion*”, J. Phys Chem. A,Vol. 100, 1996, pp 6342-6347. See also “*Additions and Corrections: Rotation Matrices for Real Spherical Harmonics*”, J. Phys Chem. A,Vol. 102, No.45, 1998, pp 9099-9100

[22] Choi, Cheol Ho et al, “*Rapid and stable determination of rotation matrices between spherical harmonics by direct recursion*”, J. Chem.Phys. Vol 111, No. 19, 1999, pp 8825-8831

[23] Blanco, Miguel A et al, “*Evaluation of the rotation matrices in the basis of real spherical harmonics*”, J. Molecular Structure (Theochem),419, 1997, pp19-27

[24] Konttinen J, Pattanaik S, Bouatouch K. Fast approximation to spherical harmonic rotation[C]// Spring Conference on Computer Graphics. ACM, 2006:PAGE@9.

[25] Ng R, Ramamoorthi R, Hanrahan P. Triple product wavelet integrals for all-frequency relighting[J]. Acm Transactions on Graphics, 2004, 23(3):477-487.

[26] Kautz J, Sloan P P, Snyder J. Fast, Arbitrary BRDF Shading for Low-Frequency Lighting Using Spherical Harmonics[C]// Eurographics Symposium on Rendering/eurographics Workshop on Rendering Techniques. The Eurographics Association, 2002:291-296.