**Spherical Harmonic Lighting:The Gritty Details**

Robin Green

翻译:练孙鸿

**0 引言(p1-p2)**

（注：这篇玩意是47页的，讲的很细的教程式文章，翻译的时候可能跳着来）

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的论文里面提出的(注：Sloan, Kautz, Snyder, *Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments*)，可以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（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）

**1 光照计算(Illumination Calculation)(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：

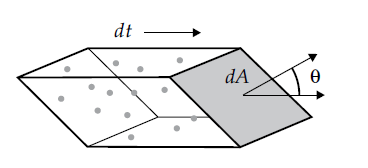
其中：

是从位置出射的的光线强度

是在的自发光

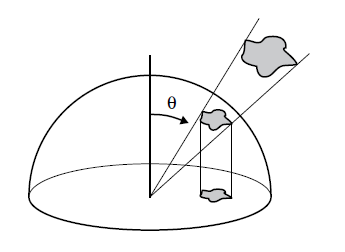
是

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)(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。

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。

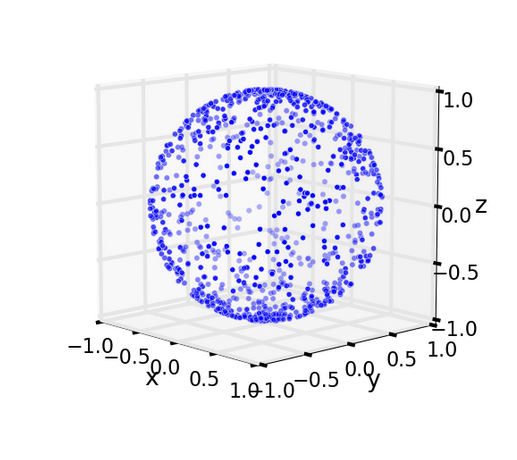
从渲染方程出发我们知道，我们需要在半球面上进行积分，所以我们就只需要**生成一些在球面上平均分布的点**（更加专业地说应该叫**无偏估计**(unbiased random samples)）。如果我们输入一对独立的变量，那么我们就可以用如下方式把这个正方形区域地随机变量映射到球面坐标：

**2\* 球面上的均匀采样(译者注)**

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，下面一篇文挺好的：

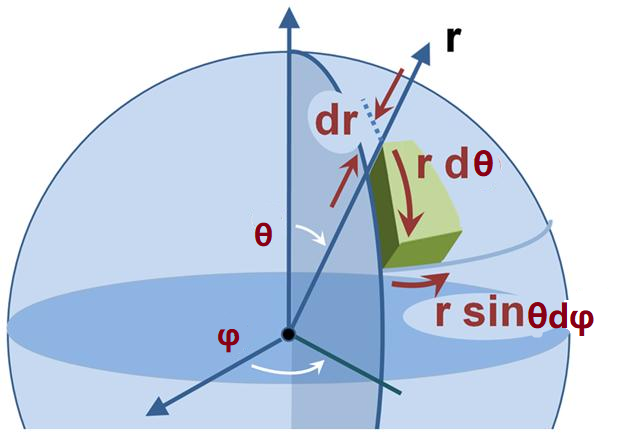
<http://corysimon.github.io/articles/uniformdistn-on-sphere/>

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**。

<https://en.wikipedia.org/wiki/Inverse_transform_sampling>

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。

下面中文维基的链接的”反函数”条目也提到这个操作。

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

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：

又因为：

反三角函数运算公式参考wiki：

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

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 :) 。

另外：有一些关于蒙特卡洛光线跟踪地文献可以参考一下：

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

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

[3] Design of a Realistic Image Synthesis System

**3 正交基函数(Orthogonal Basis Function)(p8-p )**