光线追踪渲染实践指南

清华大学 计算机科学与技术系

2021 年 5 月 16 日

目录

1 综述

日常生活中常见的游戏，电影等产品给我们呈现出了许多色彩绚丽，纷繁复杂的精彩画面。这些画面的呈现离不开渲染技术的支持。

在游戏方面，一种被称作做“光栅化”的算法被广泛应用。这种技术的优点在于可以高效地进行3D 图形的绘制，而且由于其高度可并行性，这种方法可以通过 GPU 硬件加速实现在个人终端上实时流畅的渲染。但是，这种技术的缺点在于这种技术难以或无法实现许多光照效果而导致画面失真，设计师往往需要使用大量的技巧使呈现的画面更加真实饱满。

在电影，尤其是动画电影方面，由于电影制作不需要实时动态的画面而更注重于画面的真实感，另一种被称作“光线追踪”的算法及其相关算法被大量地应用，这也是我们今天的实践主题。这类方法的优点在于其计算方法模拟了视觉感知的物理过程，可以很方便地渲染出更加具有真实感的画面。缺点在于其需要进行大量的计算以提高画面绘制的精度。不过，随着计算机硬件的升级，光线追踪技术已开始逐步应用于实时渲染领域，相信不久的将来我们可以看到质量与效率兼得的解决方案。

2 基本思想

如果你对光线追踪的基本思想已经有了一定的了解，你可以选择跳过此段以节约时间。

2.1 从光源到相机

相机成像的物理过程大概可以这样描述：首先光线从光源出发，进入到环境中。当光线在传播路径上遇到阻碍时，其传播的方向会按照一定的规律发生变化，同时其强度也可能发生一定的变化。经过在环境中的传播，如果一束光最终照进了相机的镜头，落在相机的底片上，这束光线就给底片提供了一份色彩信息。当无数条这样的光线所携带的信息汇聚在一起时，底片上便会形成一幅完整的图像。

注意：对于渲染来说，我们仅将光假想为一条条射线即可，对于在物理方面有较深造诣的同学现在请暂时不要考虑诸如“光的本质是一种电磁波，其波动性如何模拟?”一类的问题。

2.2 模拟这个过程

受以上物理过程的启发，我们可以构思出一个模拟这种过程的渲染方法：

在空间中设置若干个光源，并随机产生若干条以这些光源的位置为端点的射线。赋予每一条射线一个光强值，这些射线即是模拟的“光线”。

对每一条光线求解其在行进路径上遇到的阻碍，通常是可表示的几何体（以三角形最为常用）。根据遇到的阻碍的属性对光线的方向和强度进行修改，重复以上过程，直到光线落到虚拟的相机底片上，根据其落点位置计算其贡献，或者光线没有遇到阻碍，则视为该束光线消失在无穷远处，废弃即可。

由于现实中我们所看到的光线数量近乎无穷大，我们不可能完美地还原真实的光照场景。不过，理论上只要我们对环境以及光传播过程的描述足够详细真实，当模拟的从光源发出的光线足够多时，这种方

(a) 正向传播　　(b) 逆向追踪

图 1: 两种光路示意（图片截自迪士尼动画工作室发布的路径追踪技术讲解视频《Disney's Practical Guide to Path Tracing》）

法可以创造出以假乱真的照片级的图像。但是，虽然这种方法理论上是可行的，在实际操作中其需要消耗的计算资源过大，且浪费过多。因为有大量的光线最终并不能进入收集信息，对最终结果产生贡献的“眼睛”。或是被损耗殆尽，或是消失在无穷远的地方。因此，在实际实践中，没有人通过这种方法进行渲染。

2.3 逆向追踪

由于光路的可逆性，我们可以将以上过程反过来，如图1(b)所示。则渲染过程变为：

从相机处出发，对渲染图像（“底片”）的每一个像素产生若干对应的光线，经过在场景中的传播，如果一束光线最终抵达了一个光源，则计算该光线产生的亮度贡献，需要考虑这束光线在传播过程中的折损以及光源的亮度。否则，提供一个亮度为零的贡献。

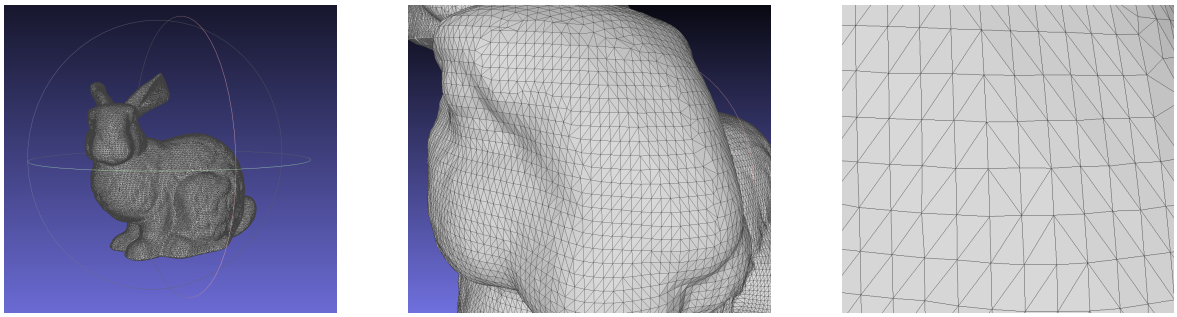
这个改进后的方法即为光线追踪的基本思想，具体的实现方法请继续阅读下文。

3 算法基础

在具体实践方面，你可能会用到一些线性代数以及几何计算相关的知识。如果你对矩阵及向量运算，以及他们和几何之间的关系不太熟悉的话，请先移步 ‘math&geometry.pdf’ 进行学习。

3.1 场景信息的表示与计算

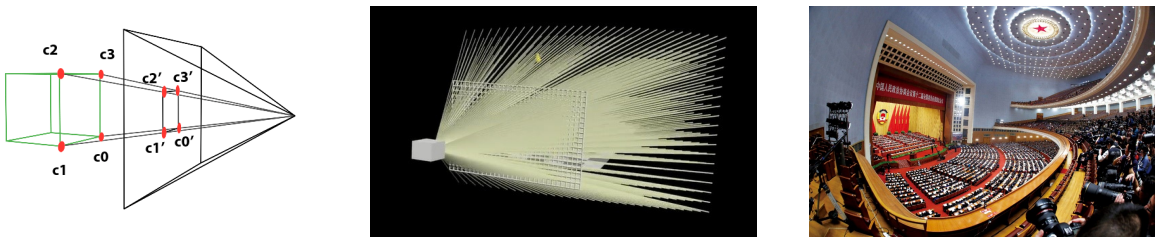
在绝大部分的渲染场景中，表示场景模型的基本单元是三角形。一个三角形只需要三个空间坐标即可表示，并且一个一般三角形一定在一个平面上，这给我们计算表示光线的射线与场景的交点提供了便利，具体的计算方法参见 ‘math&geometry.pdf’。由于三角形是最基本的平面几何单元，它可以很方便地构建出复杂的模型，如图 2，我们可以使用这种简单的平面几何单元近似地表示出曲面效果。



(a) 斯坦福兔子模型　(b) 放大图像　(c) 再次放大

图 2: 模型的三角形细节（图片使用 MeshLab 2020 渲染）

3.2 相机与投影平面



(a) 投影平面示意　(b) 从相机生成光线示意　(c) 人民大会堂（鱼眼镜头示例）

图 3: 不同类型的相机示意图（图片源自网络）

现实中的相机结构复杂，一个镜头可能使用了多层透镜以使底片成像清晰，同时还需要复杂的机械结构实现对焦操作。

对于渲染来说，我们不需要模拟如此复杂的相机模型。将以上模型简化，我们可以用一个相机基准坐标点和一个投影平面来表示相机的空间位置和朝向。此时，我们可以将成像过程想象为求解空间中的点与相机基准坐标点连线与投影平面的交点，如图 3(a) 所示。对于逆向追踪，我们则需要生成以相机基准坐标点为端点，指向投影平面上像素对应点的射线，如图 3(b) 所示。

使用投影平面的假想相机可以有效地保证渲染出的画面不会扭曲变形，准确的说，这种相机模型可以保证场景中的直线渲染出来仍是直线。因为一条直线在投影平面上的投影本质是该直线与相机基准点构成的平面与投影平面的交集，两个不平行平面的交集是一条直线。因此，这种相机模型最为常用。

不过，在一些情况下，我们需要拍摄超广角镜头的图片，或者全景图片。此时我们可以使用角度关系建立映射，得到其他类型的相机模型。如图 3(c) 所示的鱼眼镜头，我们可以明显地感觉到画面发生的严重的变形，但是其收纳内容的广度明显增加了。

在光线追踪的实践过程中，我们可以先生成以原点为相机基准坐标点，朝向正前方（通常是 z 轴正方向）的相机的采样射线，再通过坐标变换矩阵将其变换到任意位置和方向。

3.3 点光源与面光源

光照是渲染过程中需要处理的首要问题，因为相对来说把图像绘制出来是比较容易的。但是如何让绘制出的图片富有真实感就需要依托于准确地光照计算。

在表示光源的时候，我们通常会将光源分为两类：点光源和面光源

顾名思义，点光源即是把光源想象成一个点；而面光源则是一个具有面积的几何图形。他们都具有某种表示亮度的参数。

为了在计算光照时能够准确地表示亮度信息，我们需要引入了光通量、光强度、光亮度、光照度的概念。我们需要讨论的不是十分严谨的物理概念。简单来说，

光通量是一个光源总的能量输出。

光强是单位立体角的光通量（立体角是椎体在以顶点为球心的单位球面上的投影面积，类似于二维中的“弧长”，一个球面的立体角为 4π）

照度是单位面积的光通量

亮度是单位面积的光强

你不需要纠结这些概念之间的区别，我们需要用到的仅仅是光通量和亮度这两个概念。对于点光源来说，我们需要给出这个点光源的光通量，即总的能量输出。而对于面光源，我们需要给出这个面光源的亮度，即直视这个面光源时感受到的刺激程度。我们可以认为，渲染出每个像素的最终颜色就是由无数次对该像素对应视锥范围内的亮度值进行采样决定的。（在我查找到的大多数介绍图形学的文章中，亮度这个概念通常使用“辐射度”一词表达，这些概念极易混淆，区分太过细致容易产生混乱，因此本文档姑且统一使用便于理解的“亮度”一词）

3.4 全局光照

(a) 直接光照　　(b) 全局光照

图 4: 从以上两张图片的对比我们可以看出，当只计算直接光照时，场景中的角落区域是完全黑暗的。采用全局光照可以使无法直接受到光源照射的区域具有亮度，从而使图片显得更加真实

当我们考虑光照问题的时候，一个最简单的思路就是将渲染的位置与光源连线，判断该位置是否被其他场景模型遮挡，从而获知该位置是否处于阴影区，是否需要计算光源的亮度贡献。这种类型的光照

被称为直接光照，即从光源直接照射到物体表面的光照。然而我们知道，由于各种表面反射的存在，光在到达物体表面时不会就此停留，而是会通过反射继续照亮其他区域，因此在许多无法受到光源直接照射的区域，如大楼造成的阴影处依然是存在亮度的，而非一片漆黑。这种经过非自发光物体表面反射后的光照被称作间接光照。直接光照与间接光照的总和被称作全局光照。

相对来讲，直接光照比较容易计算，因为物体表面直接到光源的路径是单一的。而间接光照可能存在从光源出发，经过无数次反射后的才抵达物体表面的路径，因此，全局光照的计算就变得十分困难。直接光照和全局光照的对比可参见图 4。

3.5 物体表面性质的表示与渲染方程

物体表面的性质决定了光线的反射方向的分布情况。其中最为经典的就是理想散射模型，又称朗博（Lambertian）反射。这种模型假设了无论入射光线方向如何，物体表面在任意方向上的亮度相同，即与出射方向无关。这个模型只存在于理想情况中，显示中的大部分漫反射物体表面都不满足这个模型的假设，不过，在我们实现渲染的初期，这个模型可以非常好地帮助我们理解和实现渲染的功能，并且取得不错的效果。

为了能够准确地表示出表面反射光的亮度与入射光的关系，我们需要引入渲染方程：

$$L_o = \int_{\Omega} \rho\left(\omega_o, \omega_i\right) L_i \cos\theta_i d\omega_i$$

这个公式对于学过微积分的同学可以有效地帮助理解渲染这项工作需要解决的根本问题，没有学过积分的同学也不用太过担心，我们今天不会要求具体的积分计算。

在这个公式中，我们要求解的 L_o 表示出射方向上的亮度；

Ω 表示物体表面法向量方向的半球面，即所有可能的入射光的方向的集合；

$\rho\left(\omega_i, \omega_o\right)$ 被称作 双向反射分布函数 (Bidirectional Reflectance Distribution Function, BRDF)，它描述了当入射方向为 ω_i 时，出射方向为 ω_o 的概率。（由于出射方向是连续分布的，此函数的值本质上是概率密度）

L_i 表示入射方向上的亮度。我们不妨这么想象：当我们仰望天空时，如果天空明亮，我们自己会被照的更亮，反之则更暗。如果天空有一部分亮，一部分暗，则我们自己被照亮的程度取决于每一小部分天空的亮度的总和（写成式子就是上述的积分形式）。此项就表示入射方向那一小部分“天空”的亮度。

θ_i 表示入射方向与法线方向的夹角，$\cos\theta_i$ 含义是入射光照明面积与入射光截面面积之比。这一项保证了，当入射角度较大时，物体表面反射光的亮度会更低，因为单位面积分摊到的能量更小了（光照度更低）。这也是符合常识的，例如，傍晚的亮度比正午低。

最后的 $d\omega_i$ 表示 ω_i 的微分，没学过微积分的同学不必过于纠结此项。

一般来说，我们还会给每个表面设置一个反光率，表示光线在该表面反射后的折损程度，只需要在最后乘上即可，不写时默认为 1。

这个公式中最不确定的就是 $\rho\left(\omega_i, \omega_o\right)$ 这一项，对于理想散射模型，由于反射到各个方向上的亮度相等，因而该项为一个常数。又由于在反光率为 1 时我们需要保证入射光通量等于出射光通量，经计算可知理想散射表面的双向反射分布函数 $\rho_{\text{Lambertian}}\left(\omega_i, \omega_o\right) \equiv \frac{1}{\pi}$。以下具体推导过程可以略过：

我们从一个简单的思路出发，当所有方向入射光亮度 L_i 为恒定常数时，L_o 应等于 L_i。设需要求解的常数为 ρ，此时我们有：

$$
\begin{aligned}
L_i &= \int_\Omega \rho L_i \cos\theta_i d\omega_i \\
\frac{1}{\rho} &= \int_\Omega \cos\theta_i d\omega_i \\
&= \int_{-\pi}^{\pi} \int_0^{\frac{\pi}{2}} \sin\theta \cos\theta d\theta d\varphi \\
&= \int_{-\pi}^{\pi} \left(\frac{1}{2} \sin^2\theta \right) \Big|_0^{\frac{\pi}{2}} d\varphi \\
&= \pi \\
\rho &= \frac{1}{\pi}
\end{aligned}
$$

以上介绍的理想散射模型并不能展示出渲染方程的强大，研究者们通过实验和研究得出了更贴近实际物体反射规律的 BRDF，如迪士尼公司开发的“迪士尼原则的 BRDF”，它可以实现通过简单的参数控制即得到不同材质的观感，这大大降低了艺术工作者们的创作难度，见图5。

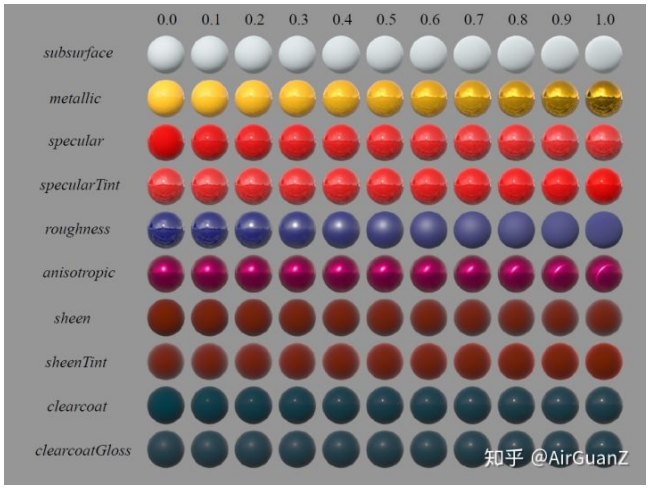


图 5: 迪士尼原则的 BRDF，只需要通过简单的参数控制就可以实现不同材质的效果

4 算法构建

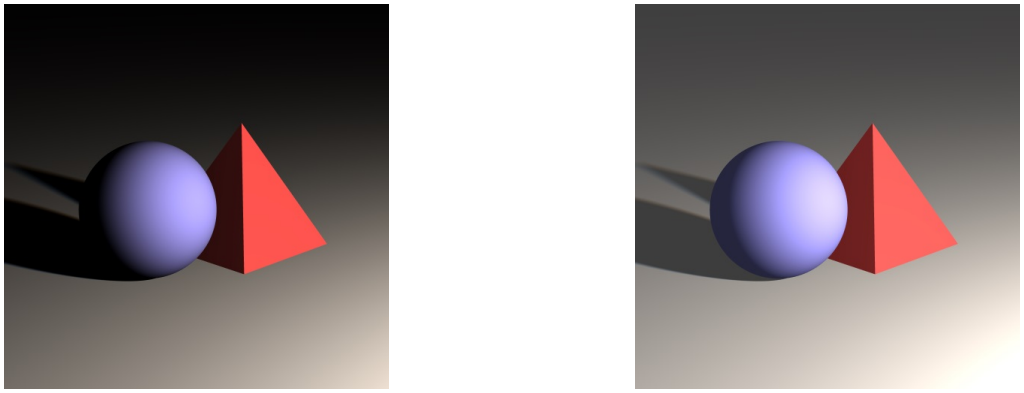
如果你已经理解了以上内容介绍的基本概念，特别是渲染方程的各项含义，那么我们就可以开始构建渲染算法了。渲染本质上就是求渲染方程的解，但是由于渲染方程本身结合了复杂的场景后不太可能手动解出，故而我们需要通过巧妙地方法逼近真实值，这就是我们构建渲染算法的目的。我们称理论上能够逼近渲染方程的真实解的算法为“无偏的”。

4.1 光线追踪

在讨论无偏算法之前，我们先考虑如何实现一个相对简单的渲染器。光线追踪算法可以实现点光源对一次漫反射路径的光照计算。下面我们就来设计一个支持理想散射与理想镜面反射的渲染器，具体来说，步骤如下：

1. 由相机产生光线（射线）
2. 计算光线与场景的交点，得到交点处的表面信息
3. 如果遇到的是漫反射表面，则将交点与光源连线检测是否被阻挡，然后计算光源的亮度贡献。如果是镜面反射表面，则计算出反射后的光线方向递归地继续追踪，直到遇到漫反射表面或不与场景相交为止。

以上这种算法流程只考虑了直接光照，因此不是无偏算法。但是由于其追踪结果相比于之后将要介绍到的路径追踪算法较为确定，且实现了简单的镜面反射等传统光栅化算法难以实现的功能，故而在一些情况下具有其特定的意义。此外，由于该算法没有间接光照，对于光源无法直接照射的区域其结果必然是全黑的（见图6(a)）。为了使画面更加美观，我们可以手动引入一个环境光变量 $L_{ambient}$，将其直接加到漫反射表面计算折损前的亮度总和中，使得不受直接光照的区域依然是可见的（见图6(b)）。



(a) 无环境光　　(b) 有环境光

图 6: 加上环境光后阴影处不再是纯黑色

4.1.1 镜面反射的计算

在初中阶段的物理课上，同学们应该学习过镜面反射的基本规则，即“入射角等于反射角”，如图7。在给定入射方向和反射面法向量的情况下，如何计算出射方向呢？我们可以发现一个非常有用的性质：入射方向和出射方向的平均值与法向量同向，且这个平均值也是入射方向和出射方向在法向量上的分量。于是，我们可以得到一个非常简单的计算出射方向的公式：

令 $\vec{w_i}$ 表示入射方向（注意，入射方向和入射光行进方向是相反的），$\vec{n}$ 表示法向量，$\vec{w_o}$ 表示需要求的出射方向。有

$$\frac{\vec{w_i}+\vec{w_o}}{2}=(\vec{w_i}\cdot\vec{n})\times\vec{n}$$

$$w_o=2\left(\vec{w_i}\cdot\vec{n}\right)\vec{n}-\vec{w_i}$$

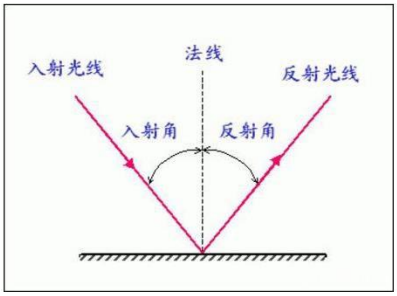


图 7: 镜面反射示意图

4.1.2 点光源对漫反射表面的贡献

对于一个总光通量为 F 的光源，设其与场景内一漫反射点的距离为 d。若该光源与漫反射表面点的连线无场景中其他物体遮挡，则该光源对该点的亮度贡献为 $\frac{F\cos\theta_i}{4\pi^2d^2}$（$\theta_i$ 依旧表示入射方向与法线方向的夹角），这样可以保证总的入射光通量等于出射光通量。多个光源贡献直接相加即可。

4.2 路径追踪

路径追踪算法继承了光线追踪方法的核心思想，并引入蒙特卡洛方法用以逼近渲染方程的积分结果。

4.2.1 蒙特卡洛

当我们遇到难以精确计算的大样本数据时，如围棋随机走子的胜率，难以求解的函数积分等。我们通常可以使用蒙特卡洛方法渐进地逼近真实值。其核心思想是通过随机采样的方式得出近似的采样平均值。

例如我们希望求函数 $f(x)=\sin x^{\cos x}$ 在 $[0,\frac{\pi}{2}]$ 区间上的积分（一元函数积分可以简单地理解为函数图像在对应区间内扫过的有向面积）

我们无法快速的求出这个积分的精确值（这个函数是随便编出来的，至少编者不会解），但是我们可以先生成大量 $[0,1]$ 区间内的均匀随机数 $x_i(1\le i\le N)$，计算出对应位置的函数值 $f(x_i)$，然后求出所有采样出的函数值的平均值 $\frac{1}{N}\sum_{i=1}^{N}f(x_i)$，再乘以 $[0,\frac{\pi}{2}]$ 区间的长度，也就是 $\frac{\pi}{2}$。当样本的数量足够大时，我们可以证明我们的采样结果得到的值与真实值之间的误差大于某个精度值 ϵ 的概率 δ 非常小，即

$$\Pr\left(\left|\frac{\pi}{2N}\sum_{i=1}^{N}\sin x_i^{\cos x_i}-\int_0^{\frac{\pi}{2}}\sin x^{\cos x}dx\right|>\epsilon\right)<\delta$$

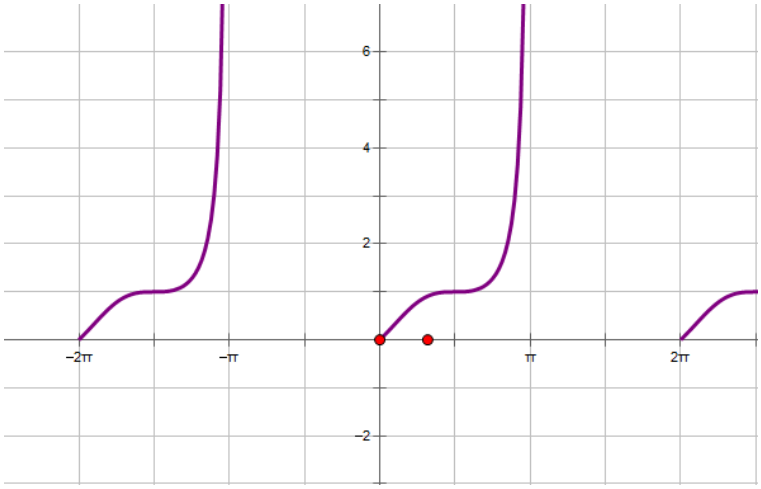


图 8: $\sin x^{\cos x}$ 函数图像

（ϵ, δ 是两个非常小的数，当 $N \to \infty, \epsilon, \delta \to 0$。对这一部分的证明方法有兴趣的同学可以提前学一学概率论）

（概率密度函数简介，此段可以略过）上文提到，在求完采样结果的平均值后，我们需要将结果再乘上采样区间长度 $\frac{\pi}{2}$。这个常数本质上是我们在 $[0, \frac{\pi}{2}]$ 内产生均匀随机数时对应的概率密度函数的倒数。概率密度函数描述了连续随机数的分布情况。其含义是，若一个随机变量在点 x 处的概率密度函数值为 $f(x)$，则该随机数落在 x 周边长度为 d 的邻域内的概率是 $df(x)$。我们需要保证，概率密度函数在其随机域上的积分为 1，即各处随机数总概率为 1。因此，对于均匀随机数，其概率密度函数在随机域内是恒定常数，为随机域大小的倒数，在随机域外为 0。

在路径追踪算法中，我们希望能够求解渲染方程，但是又不可能得到精确解，因此，我们引入了蒙特卡洛方法。需要注意，由于直接采样点光源得到的亮度理论上是无穷大，而采样到点光源的概率又是无穷小，故点光源不再适合于路径追踪算法。我们需要采用面光源渲染。

渲染方程

$$L_o = \int_\Omega \rho\left(\omega_o, \omega_i\right) L_i \cos\theta_i d\omega_i$$

求的是一个半球面区域内的函数积分，我们的采样也要在这个半球面上进行。具体过程如下：

1. 由相机产生光线（射线）

2. 计算光线与场景的交点，得到交点处的表面信息

3. 如果遇到的是漫反射表面，则在法线方向半球均匀随机一个追踪方向继续追踪。如果是镜面反射表面，则计算出反射后的光线方向递归地继续追踪。

4. 在追踪过程中，如果光线遇到或穿过了一个面光源，则在该层追踪返回的亮度中加上该面光源的亮度值。（对于今天我们需要实现的渲染器，面光源本身不阻挡光线的行进，只在路径中提供亮度贡献，故当两个光源重合时，其亮度值翻倍）

5. 每个像素重复多次采样操作以使采样结果逼近真实值

需要注意，我们每次采样返回的都是 L 值，对于漫反射表面，我们需要乘上 BRDF 值 $\frac{1}{\pi}$，光通量分摊系数 $\cos\theta_i$ 以及在半球面上均匀随机采样的概率密度函数的倒数。对于以上过程中如何在半球面上产生均匀随机向量的方法以及追踪过程的终止条件请继续阅读“拒绝采样”与“俄罗斯轮盘”两章内容。

4.2.2 拒绝采样

假设我们拥有一个可以生成 $[0,1]$ 区间内均匀随机数的随机数生成器，如何将其改写为半球面上的均匀随机器呢？

第一步，我们可以通过三次 $[0,1]$ 随机生成器的调用获得一个 $[0,1]^3$ 立方体内的随机向量。

第二步，将这个立方体放缩到 $[-1,1]^3$ 范围内

第三步，判断我们随机出的向量是否在以原点为球心，半径为 1 的球体内，如果不在，则返回第一步

第四步，将得到的随机向量放缩到模长为 1 的状态，如果这个向量与物体表面法向量的夹角大于 π，则将其取反。

这种在某个特定区域内产生均匀随机数的思想被称作拒绝采样，对于无法比较有效地直接进行均匀采样的随机域，这种方法非常的简单易用。（当年编者在进行冬令营面试的时候，被问到如何把一个 1 到 7 的离散均匀随机器改成一个 1 到 10 的离散均匀随机器，其核心思想也是如此【这不算透题吧】）

4.2.3 俄罗斯轮盘

俄罗斯轮盘赌是一种非常血腥的游戏，大概是说游戏参与者轮流拿一把只装有一颗子弹的左轮手枪冲自己开枪，活下来的人就是胜利者。对于我们的路径追踪算法来说，如果渲染场景的空间是封闭的，光线永远不会逃逸到无穷远处自动停止，因此我们必须加入一个自动停止追踪过程的机制。其核心思想与“俄罗斯轮盘”非常相似，即每次进入下一层追踪前，以一个概率 p “毙掉”之后的追踪过程，这样就可以保证整个过程追踪的层数期望为 $\frac{1}{p}$。特别注意：因为每层都有 p 的概率不再进行追踪，我们需要给追踪的结果乘上 $\frac{1}{1-p}$ 以保证采样结果是无偏的。在具体实现过程中，p 的取值建议为 0.1 到 0.2。

4.2.4 直接光采样

路径追踪的光照采样本身是一个十分被动的过程，只有当采样光线穿过光源面时才能为采样结果提供贡献。为了提高采样率，一个很自然的想法就是增加光源方向的采样光线数量。直接光采样的主要思想就是，在追踪到一个漫反射表面时，直接对光源上的一个点进行采样，以提高直接光照的采样效率。当然，一直采样光源上的同一个点是不对的，这样我们解得的光照效果就退化成了点光源的情形。这个方法的正确理解是：把面光源想象成一个会在其所在面上均匀随机出现的点光源，其整体呈现的效果就是面光源了。需要注意的是，所谓“随机出现的点光源”，其实是“随机出现的面积微元”，面光源是有方向性的，我们在转化模型的过程中不能丢失这一性质。

这个集中了整个面光源能量的面积微元的等效光强是多少呢？假设这个面光源是一个三角形，我们从这个三角形中心点沿三角形所在平面的法线方向移动至无穷远处，此时，这个三角形光源就退化成了一个点光源，这个近似点光源的光强为面光源亮度乘上该面光源的面积。

于是，我们可以得出，直接光采样，一个面光源上的“近似采样点光源”对漫反射表面上一点的贡献为

$$\frac{La\cos\theta\cos\phi}{\pi d^2}$$

其中 L 表示面光源的亮度，a 表示面光源的面积，θ 表示漫反射表面点到“光源采样点”方向与漫反射表面法向量的夹角，ϕ 表示面光源在“光源采样点”处的法向量与“光源采样点”到漫反射表面点方向的夹角，d 表示“光源采样点”到漫反射表面点的距离。

经过对比不难发现，上述公式与光线追踪算法的漫反射表面贡献计算公式十分相似，主要区别是加入了光源上夹角的余弦一项。直接光采样就是需要在追踪到漫反射表面时，通过均匀随机的方式将面光源近似成点光源，直接依照上述公式计算光照贡献。需要注意的是，计算过直接光的表面需要在下一层追踪中移除简介光照的贡献，否则贡献被重复计算会导致图片出现偏差。

5 后记

至此，有关光线追踪和路径追踪的主要知识点已经讲完了。图形学是一个非常有意思的研究领域，这方面的研究给我们带来了无数精彩的视觉体验。除去真实感渲染外，计算机图形学领域还研究图像处理，动画技术，物理模拟等方面的内容。希望感兴趣的同学积极地在课余时间进行钻研。祝大家一切顺利！