



中山大學 软件工程学院  
SUN YAT-SEN UNIVERSITY SCHOOL OF SOFTWARE ENGINEERING

# Lecture 17: 全局光照

SSE315: 计算机图形学  
Computer Graphics

---

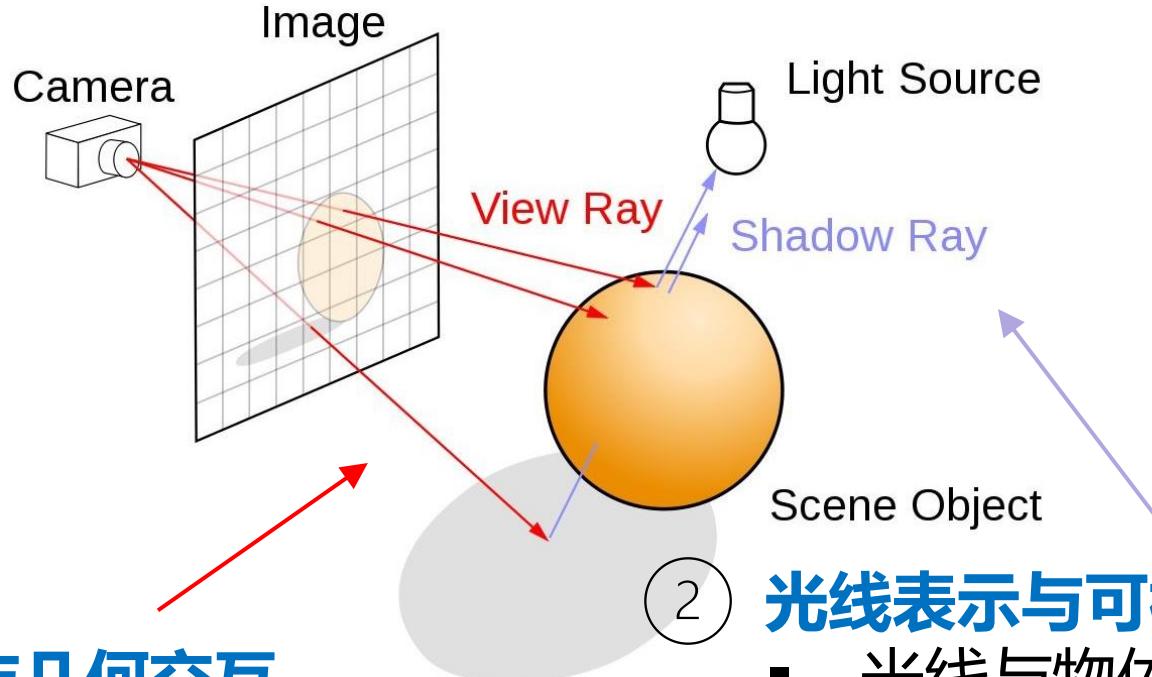
陈壮彬

软件工程学院

chenzhb36@mail.sysu.edu.cn

# 光线追踪 Ray tracing

分成 **光线与几何交互** 和 **光线表示与可视化** 两部分



## ① 光线与几何交互

- 查找光线照射在几何中的位置
- 如何加速上述过程

看到哪里

## ② 光线表示与可视化

- 光线与物体的相互作用，包括反射、折射和散射等
- 如何建模光线
- 全局光照

看到什么

# Today's topics

□ 双向反射分布函数

□ 渲染方程

□ 蒙特卡罗积分法

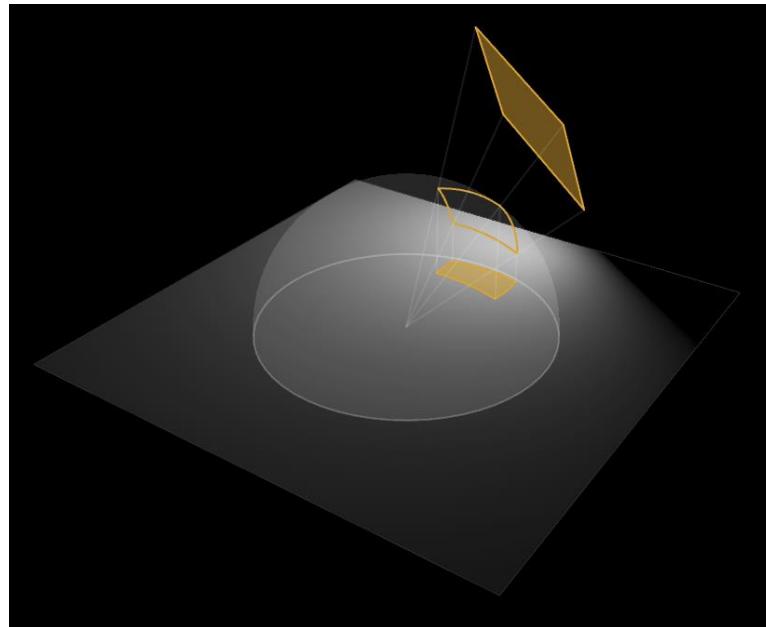
□ 路径追踪

# 辐射度量学 (概念上的) 理解

口上节课学习了很多概念：

- 辐射能量 (Radiant energy)
- 辐照度 (Irradiance)
- 辐亮度 (Radiance)
- ...

口忘记所有细节，只需从概念上知道我们现在能做什么

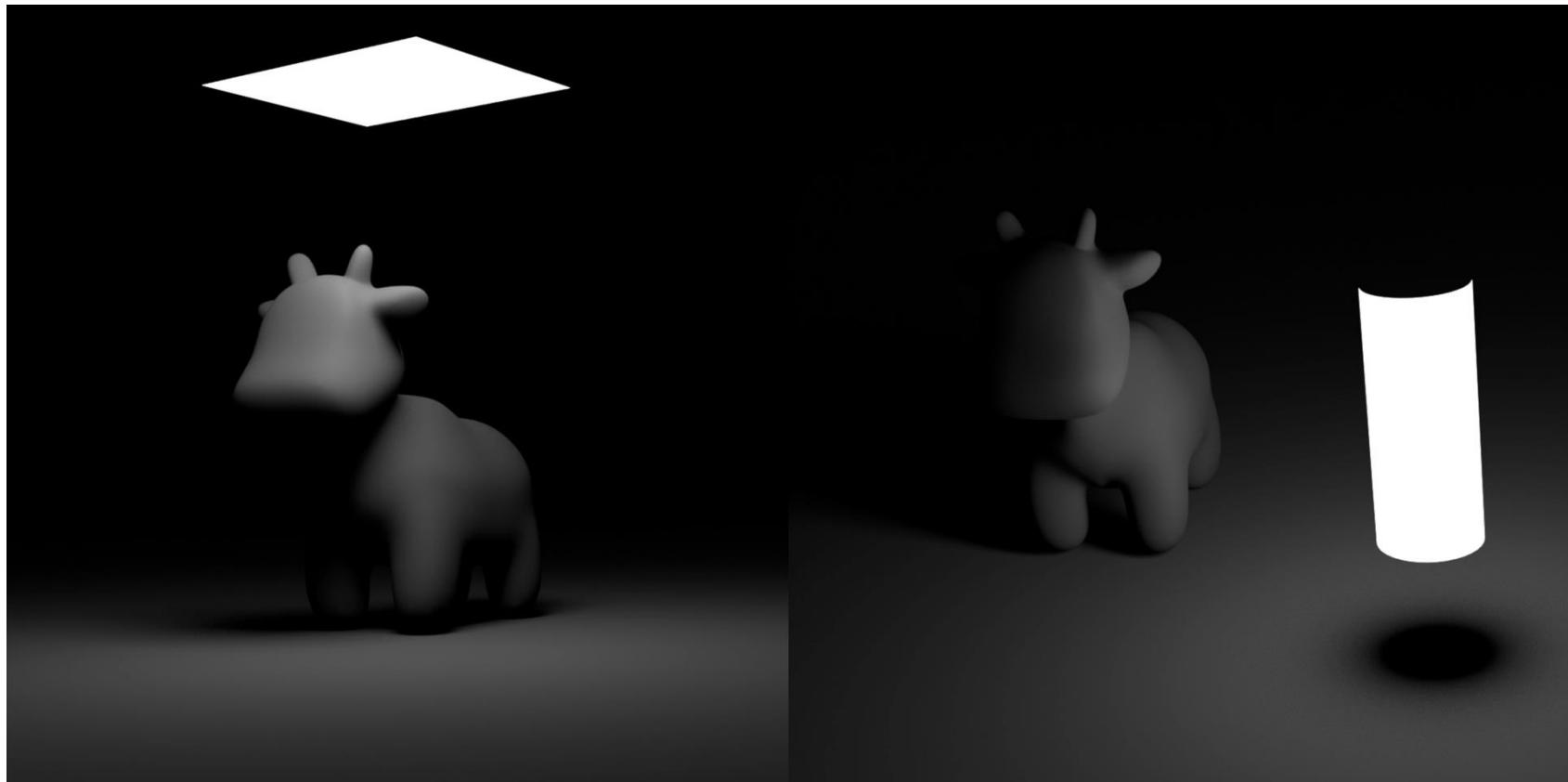


计算一个面光源如何照亮中心一个小区域

计算一个光源从不同方向上照射了多少能量到物体表面的某个小区域

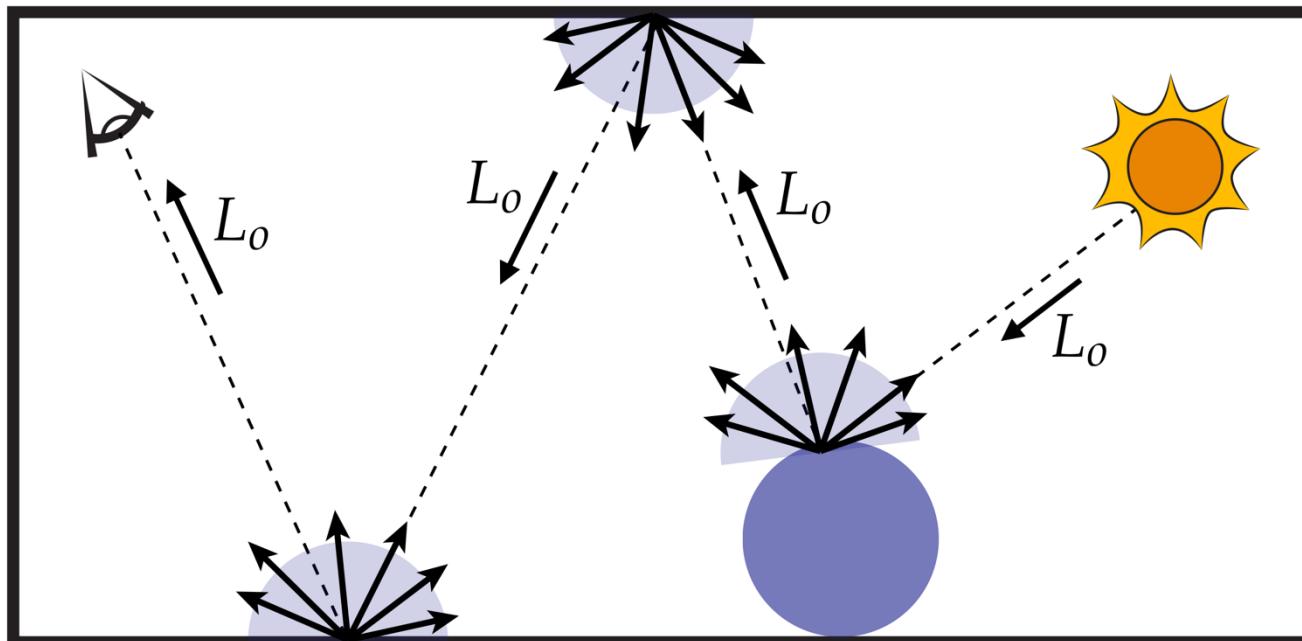
# 辐射度量学 (概念上的) 理解

因此，我们可以准确计算光源到达物体各个部分的能量，进而计算光源如何照亮物体！



# 辐射度量学 (概念上的) 理解

但是，真实的物理世界不仅仅只有光源照亮物体  
还有光线在物体之间形成的**复杂反射**！



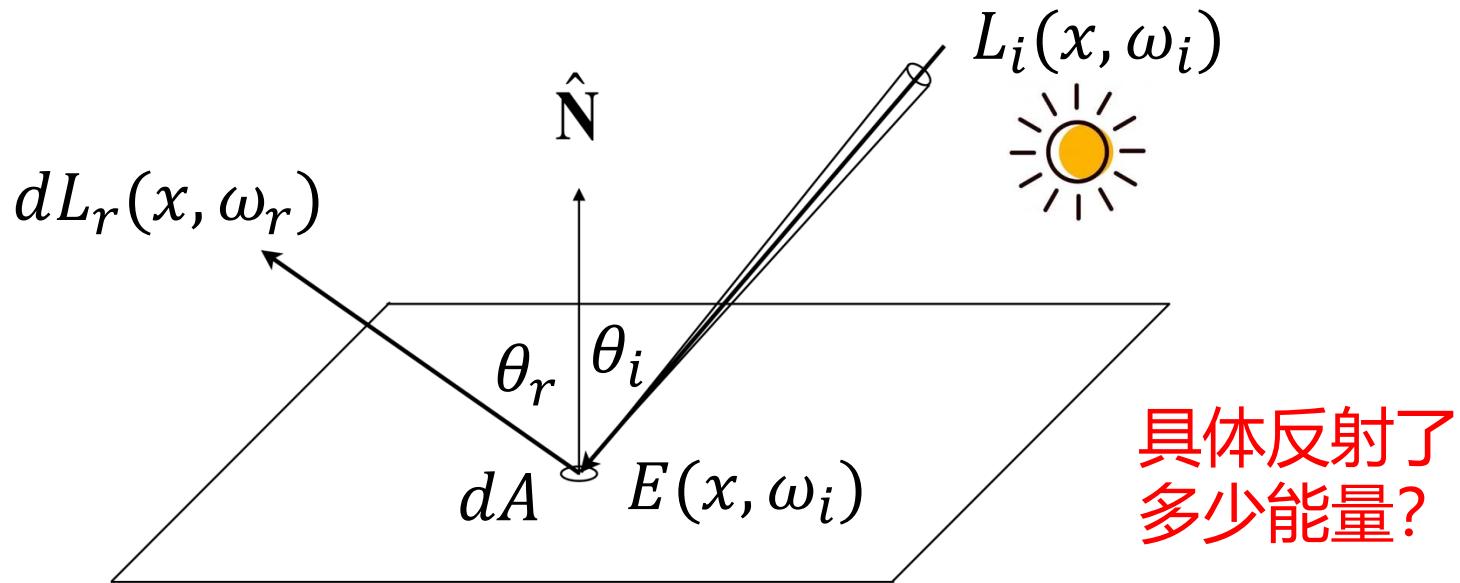
这也是需要光线追踪的原因，光栅化无法建模此过程

对于光在物体表面的**反射**，我们同样可以将物体视作“光源”，计算其如何“照亮”其他物体

# 一个点的反射

□ 通过计算辐照度，我们能知道一个小区域  $dA$  从方向  $\omega_i$  接收的能量  $E$

□ 能量  $E$  随后会变成  $dA$  向另一个方向  $\omega_r$  发射的辐亮度



□ 微分入射辐照度:  $dE(x, \omega_i) = L_i(x, \omega_i) \cos \theta_i d\omega_i$

□ 微分出射辐亮度 (由于  $dE(x, \omega_i)$ ):  $dL_r(x, \omega_r)$

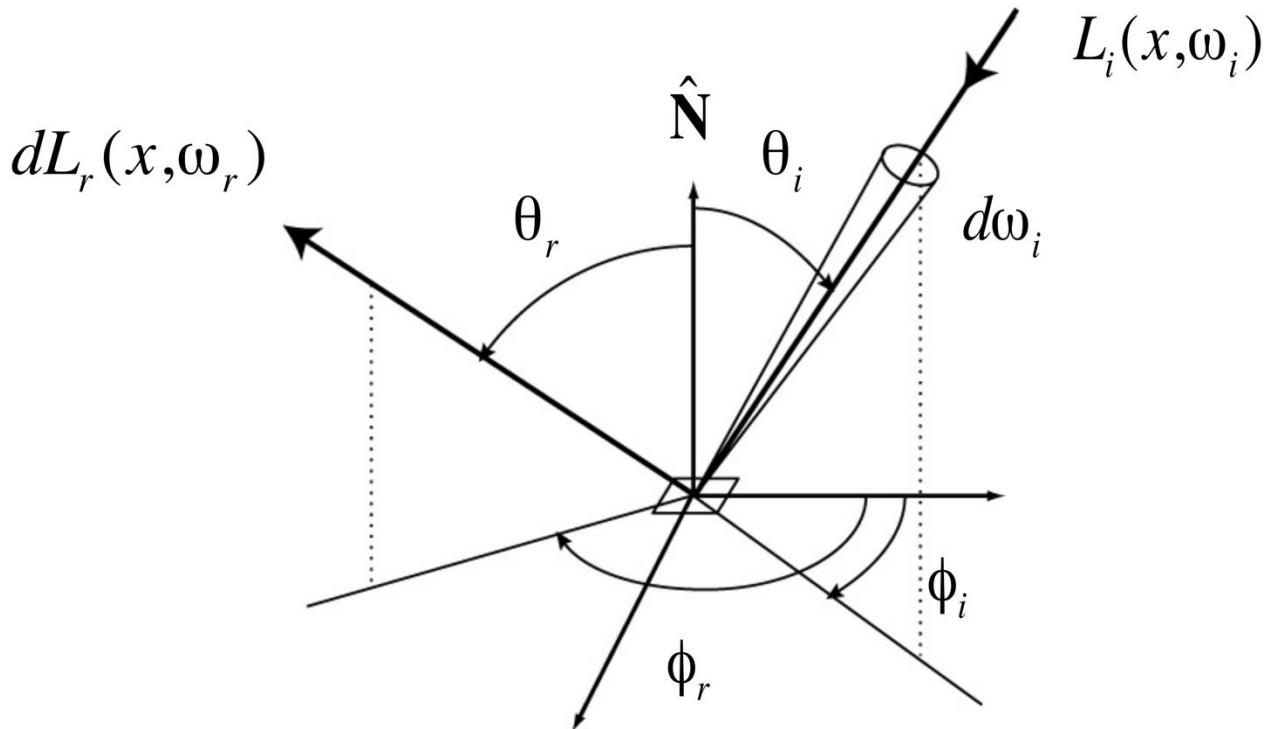
# 双向反射分布函数

# Bidirectional Reflectance Distribution Function

## (BRDF)

# 双向反射分布函数 $f_r(\omega_i \rightarrow \omega_r)$

□代表有多少光从入射方向  $\omega_i$  反射到出射方向  $\omega_r$



$$f_r(\omega_i \rightarrow \omega_r) = \frac{dL_r(\omega_r)}{dE_i(\omega_i)} = \frac{dL_r(\omega_r)}{L_i(\omega_i) \cos \theta_i d\omega_i} \left[ \frac{1}{sr} \right]$$

# 反射模型

口反射是指入射到表面上的光与表面相互作用的过程，使其在频率不变的情况下离开入射侧

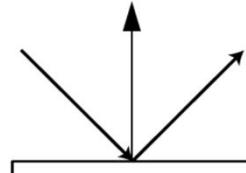
口反射函数的选择决定表面外观 (颜色，光泽度等)



# 一些常见的反射模型

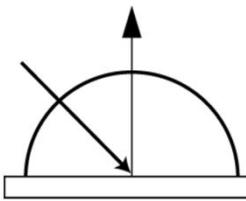
口理想的镜面反射 (Ideal specular)

- 完美的镜子



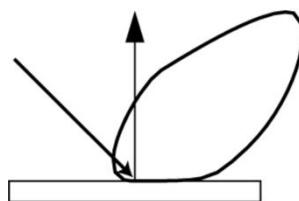
口理想的漫反射 (Ideal diffuse)

- 全方位均匀反射



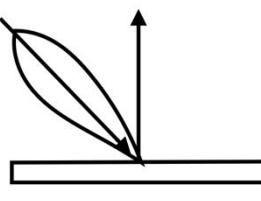
口光泽镜面反射 (Glossy specular)

- 大部分光分布在反射方向



口向后反射 (Retro-reflective)

- 将光线反射回光源



右侧图片展示了来自给定方向的入射光如何在不同方向上反射

# 漫反射



塑料



# 红色半光泽漆



# 福特神秘漆



镜面



黄金

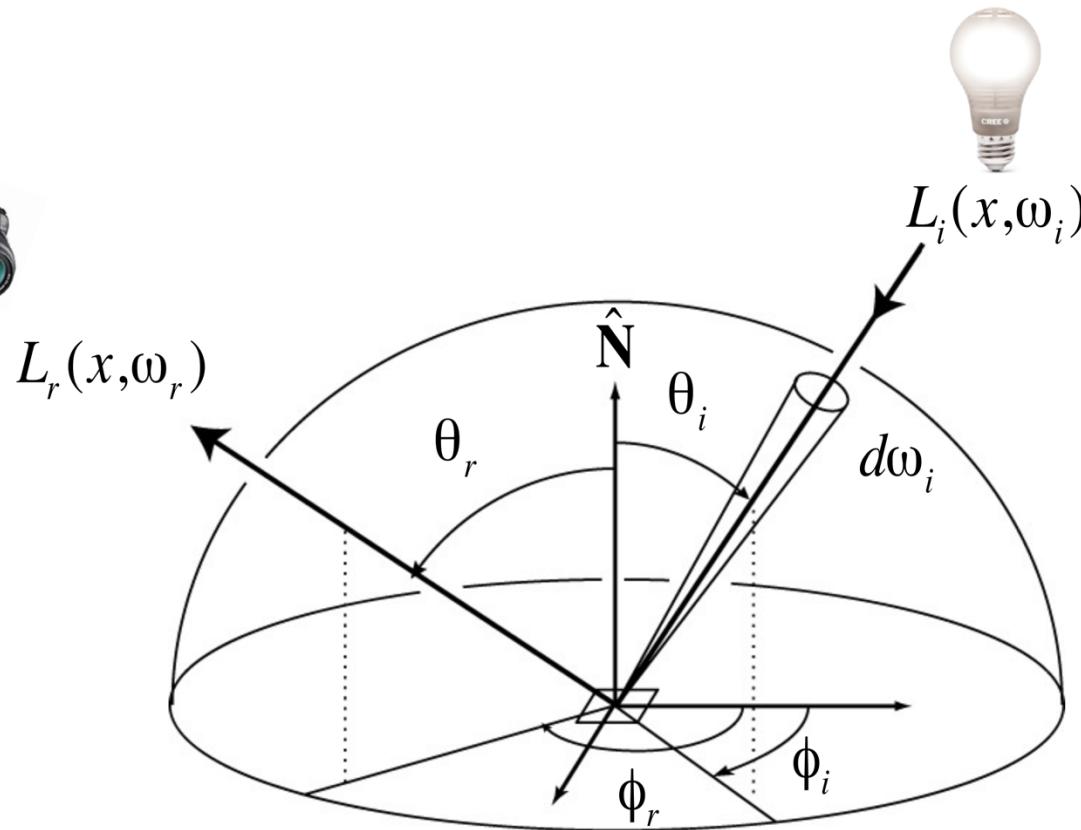


# Materials



基于双向反射分布函数，定义  
光线在物体表面的反射

# 反射方程 The reflection equation



$$L_r(x, \omega_r) = \int_{H^2} f_r(x, \omega_i \rightarrow \omega_r) L_i(x, \omega_i) \cos \theta_i d\omega_i$$

辐亮度 Radiance

BRDF

辐照度 irradiance

# 渲染方程 The rendering equation

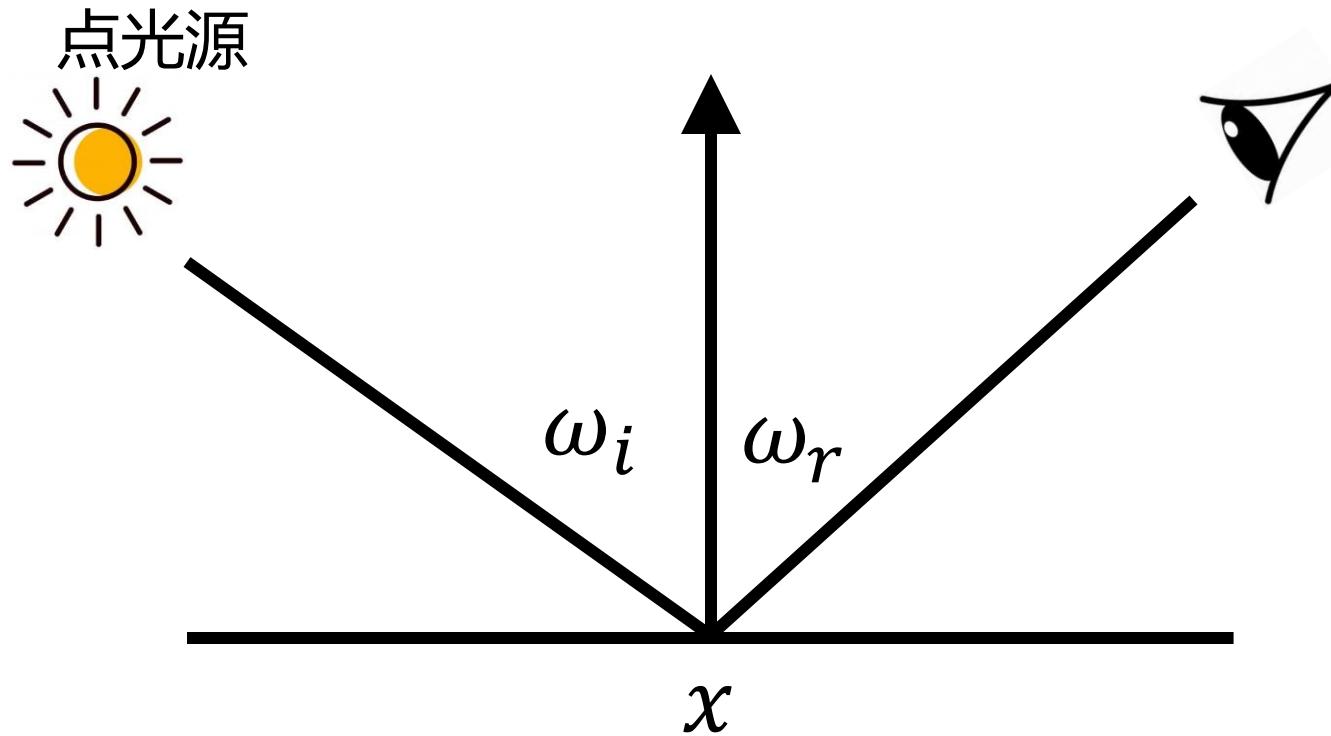
口在反射方程中添加一个 emission term 使其更具一般性

$$L_r(x, \omega_r) = \int_{H^2} f_r(x, \omega_i \rightarrow \omega_r) L_i(x, \omega_i) \cos \theta_i d\omega_i$$



$$L_r(x, \omega_r) = \boxed{L_e(x, \omega_r)} + \int_{\Omega^+} L_i(x, \omega_i) f_r(x, \omega_i, \omega_r) (n \cdot \omega_i) d\omega_i$$

# 渲染方程



$$L_r(x, \omega_r) = L_e(x, \omega_r) + L_i(x, \omega_i)f_r(x, \omega_i, \omega_r)(n \cdot \omega_i)$$

反射的光线  
(即生成的图像)

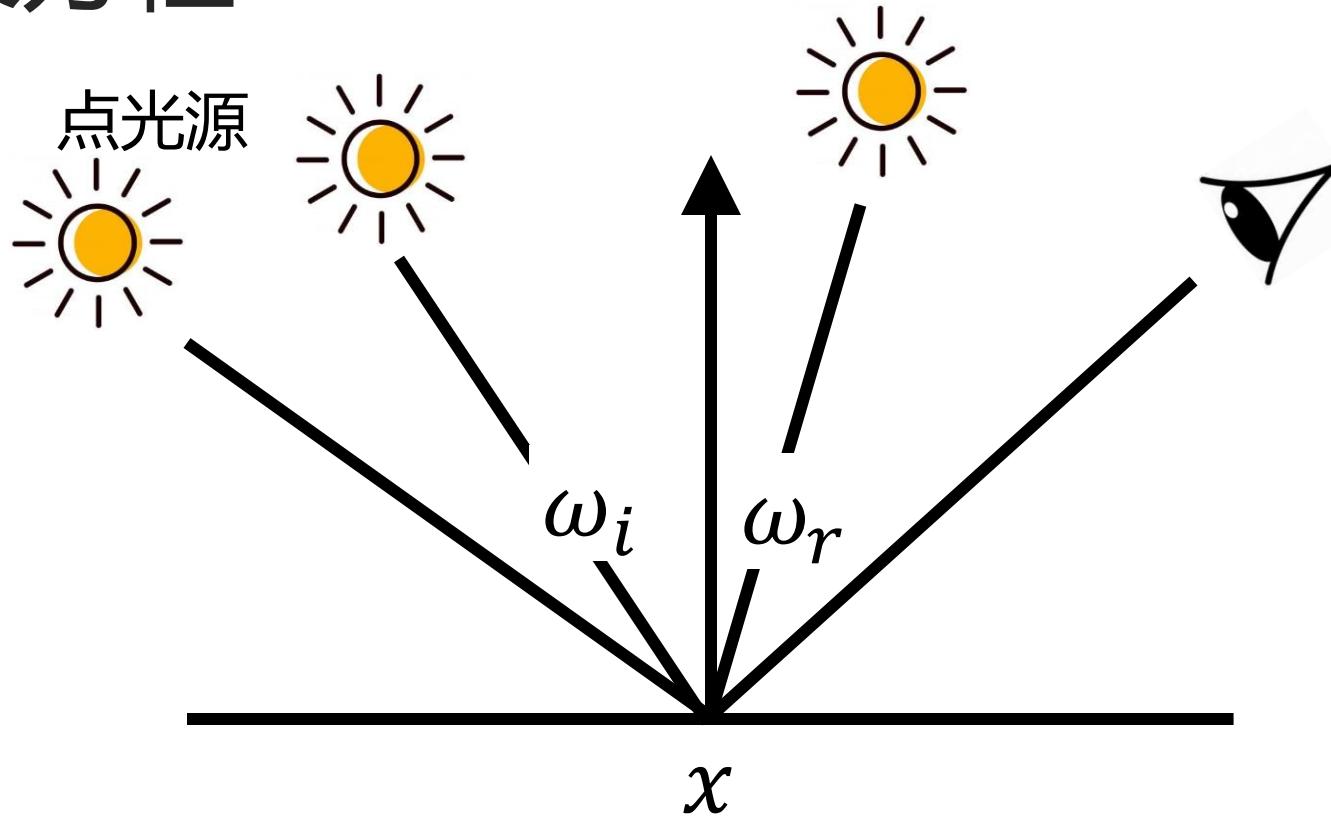
自身光源

来自光源  
的入射光

BRDF

入射光  
的角度

# 渲染方程



$$L_r(x, \omega_r) = L_e(x, \omega_r) + \sum L_i(x, \omega_i) f_r(x, \omega_i, \omega_r) (n \cdot \omega_i)$$

反射的光线  
(即生成的图像)

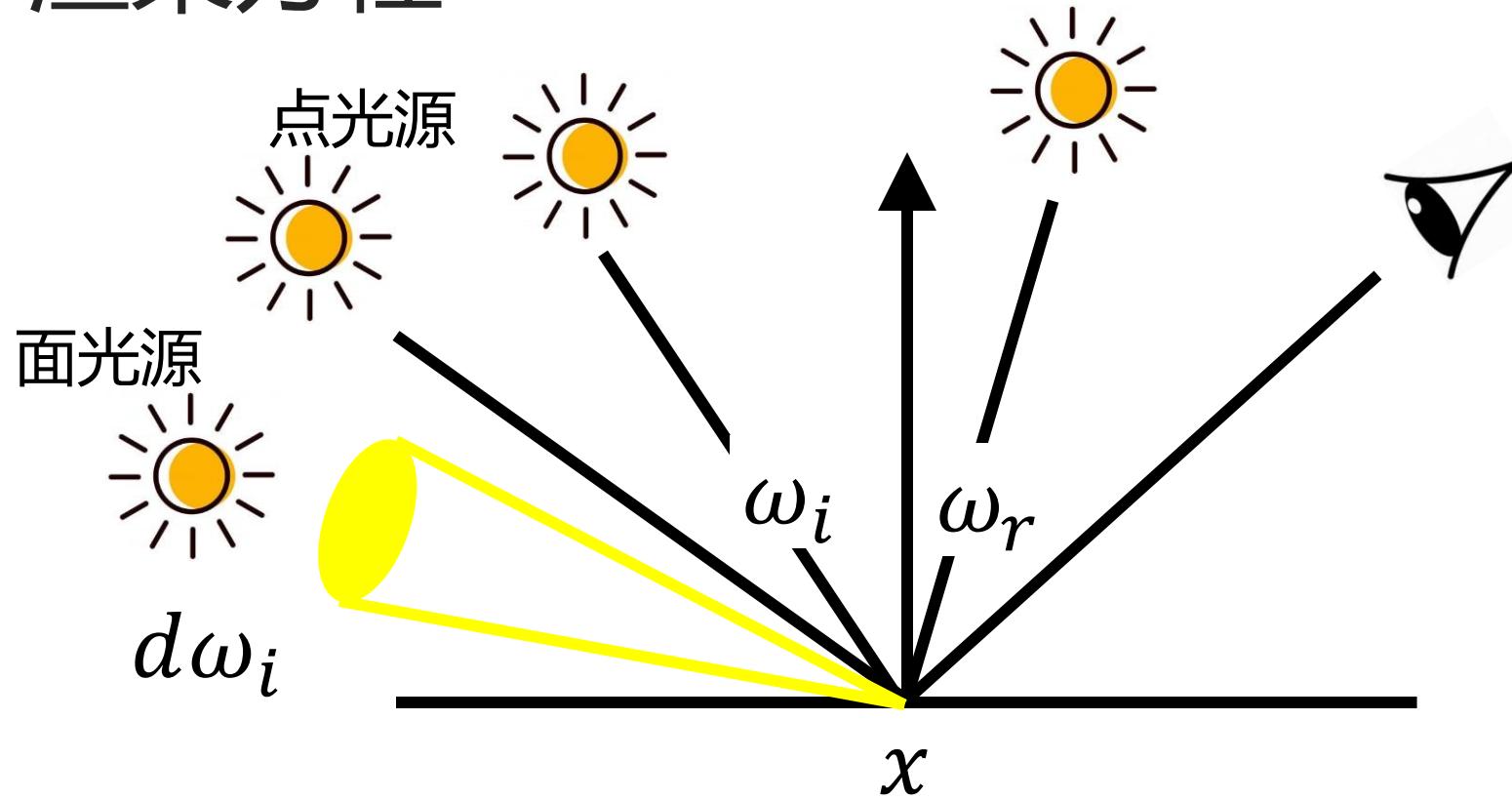
自身光源

来自光源  
的入射光

BRDF

入射光  
的角度

# 渲染方程



$$L_r(x, \omega_r) = L_e(x, \omega_r) + \int_{\Omega} L_i(x, \omega_i) f_r(x, \omega_i, \omega_r) \cos \theta_i d\omega_i$$

反射的光线  
(即生成的图像)

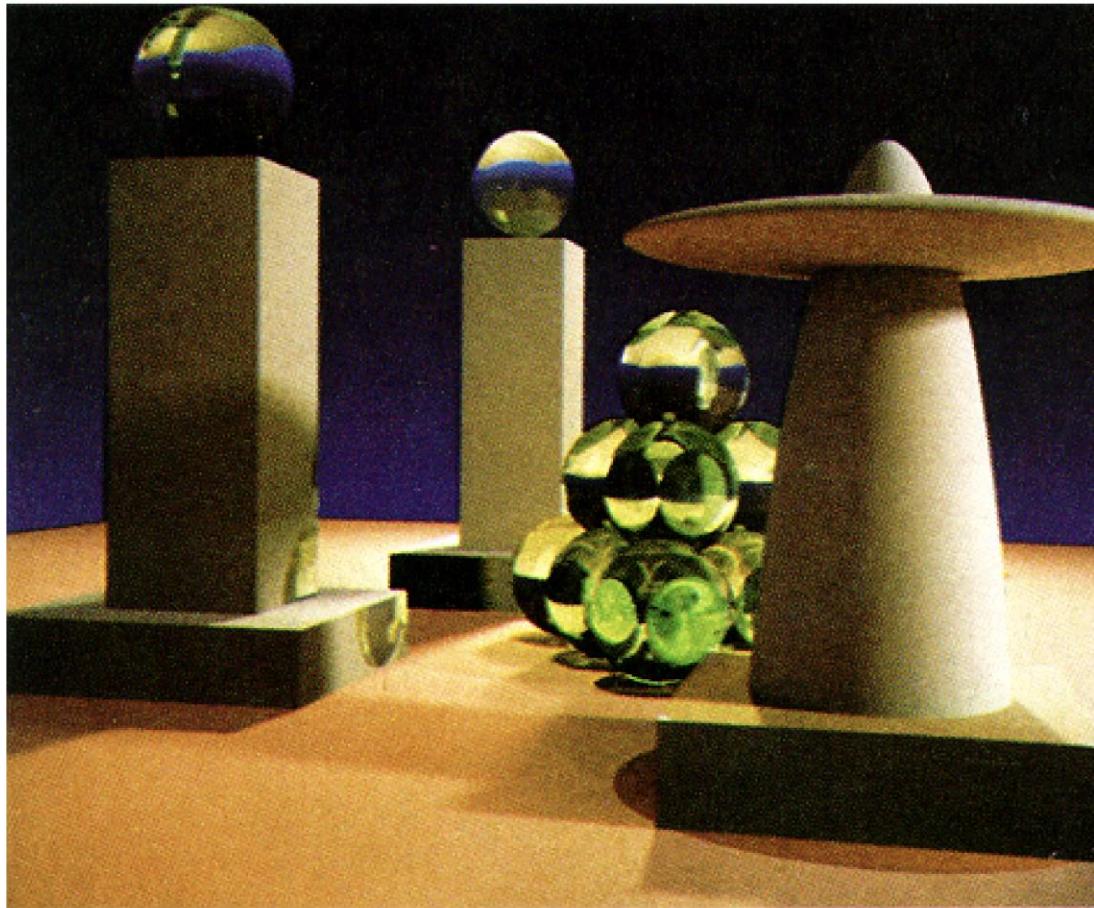
自身光源

来自光源  
的入射光

BRDF

入射光  
的角度

# 渲染方程 (Kajiya 86)

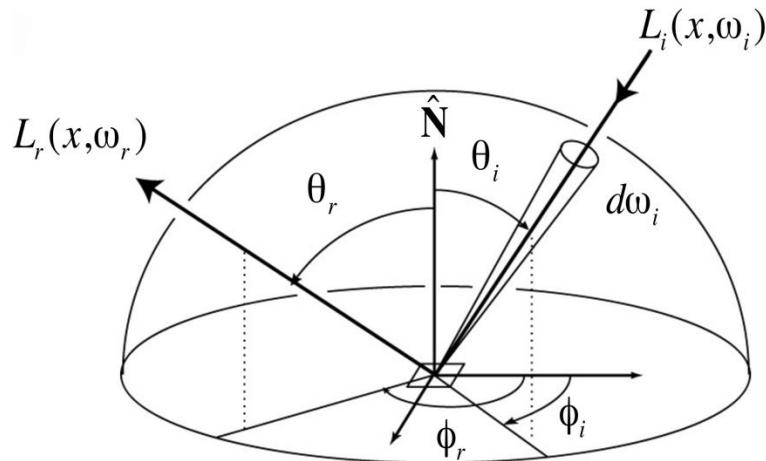


**Figure 6.** A sample image. All objects are neutral grey. Color on the objects is due to caustics from the green glass balls and color bleeding from the base polygon.

# 渲染方程的递归定义

口反射辐亮度 radiance 取决于入射辐亮度

$$L_r(x, \omega_r) = \int_{H^2} f_r(x, \omega_i \rightarrow \omega_r) L_i(x, \omega_i) \cos \theta_i d\omega_i$$



口但入射辐亮度同样取决于场景中另一点的反射辐亮度

口如何求解?

•  
*p*

# Direct illumination



•p

One-bounce **global illumination (dir+indir)**

$\bullet p$

# Two-bounce global illumination



$\bullet p$

# Four-bounce global illumination

$\bullet p$

# Eight-bounce global illumination

•  
*p*

# Sixteen-bounce global illumination

# **概率回顾**

# **Probability Review**

# 随机变量

- $X$ : 随机变量, 代表了一系列可能值的分布
- $X \sim p(x)$ : 概率密度函数 (PDF), 用于描述一个随机过程的值为  $x$  的相对概率
- **比如均匀的 PDF**: 所有的值出现的概率相同



- 例如骰子

- $X$  的值为  $1, 2, 3, 4, 5, 6$  的概率相等
- $p(1) = p(2) = p(3) = p(4) = p(5) = p(6)$

# 概率

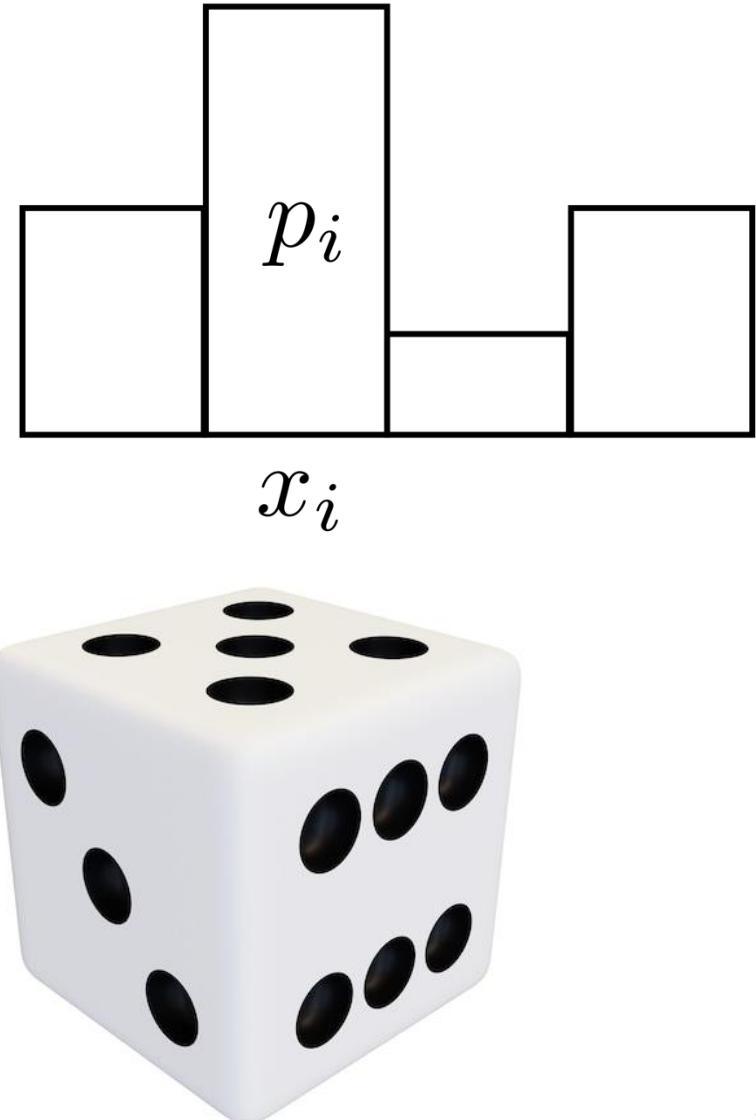
□  $n$  个概率分别为  $p_i$  的离散值  $x_i$

□ 概率函数的性质

$$p_i \geq 0$$

$$\sum_{i=1}^n p_i = 1$$

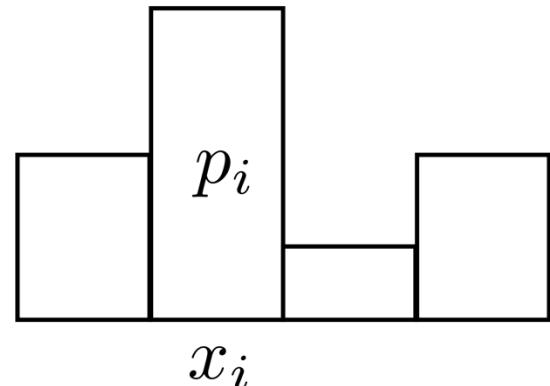
□ 六面骰子的例子： $p_i = \frac{1}{6}$



# 随机变量的期望值

□ 不断地从随机分布中抽样得到的平均值

□ 假设  $X$  是从一个具有  $n$  个离散值  $x_i$  和概率  $p_i$  的分布中抽样



□  $X$  的期望值为:  $E[X] = \sum_i^n x_i p_i$

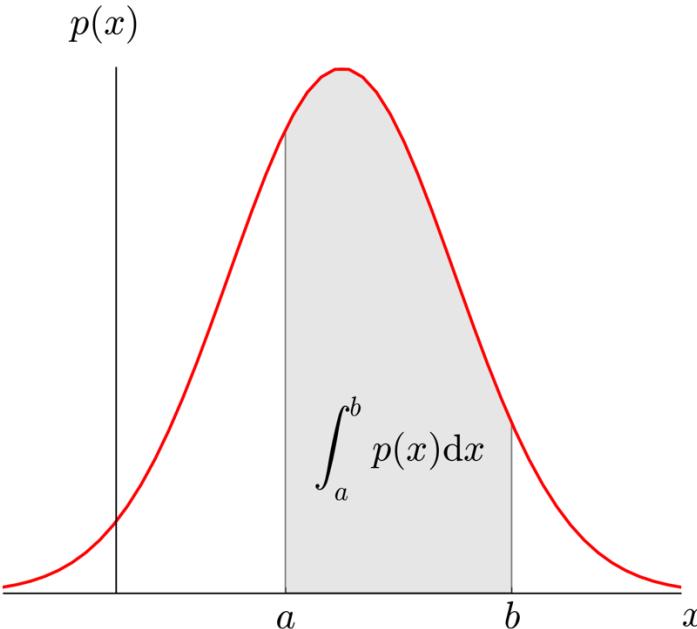
□ 骰子例子:  $E[X] = \sum_{i=1}^n i / 6$

$$= \frac{1 + 2 + 3 + 4 + 5 + 6}{6} = 3.5$$



# 连续分布：概率分布函数 PDF

□ 一个随机变量  $X$  可以取任何连续集合的值，其中特定值的相对概率由连续概率密度函数  $p(X)$  给出



□  $p(X)$  的特征： $p(x) \geq 0$  and  $\int p(x) dx = 1$

□  $X$  的期望值： $E[X] = \int x p(x) dx$

# 随机变量的函数

□ 随机变量  $X$  的函数  $Y$  也是一个随机变量

$$X \sim p(x)$$

$$Y = f(x)$$

□ 随机变量的函数的期望值

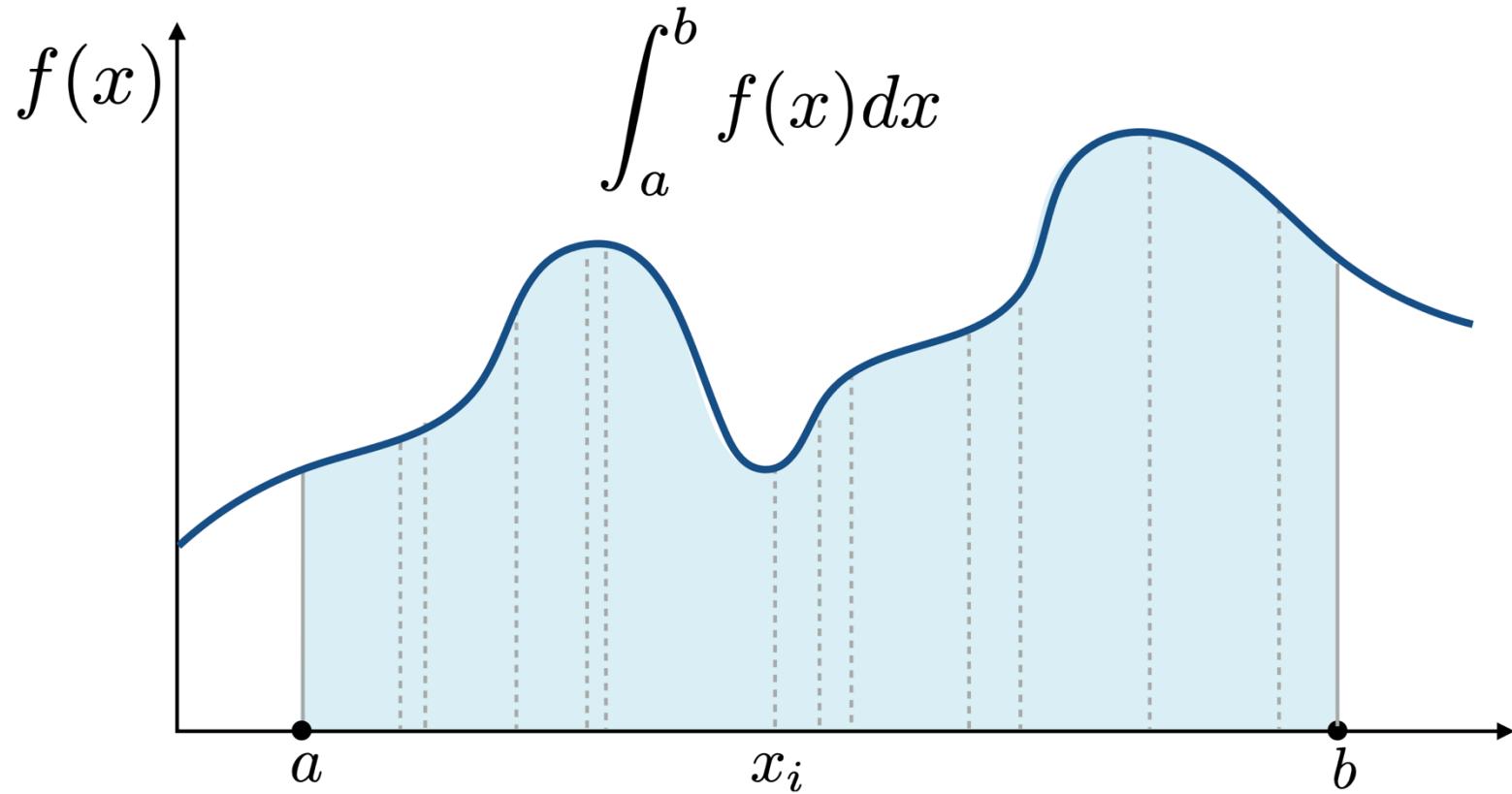
$$E[Y] = E[f(x)] = \int f(x)p(x) dx$$

# 蒙特卡罗积分 Monte Carlo Integration

# 蒙特卡罗积分

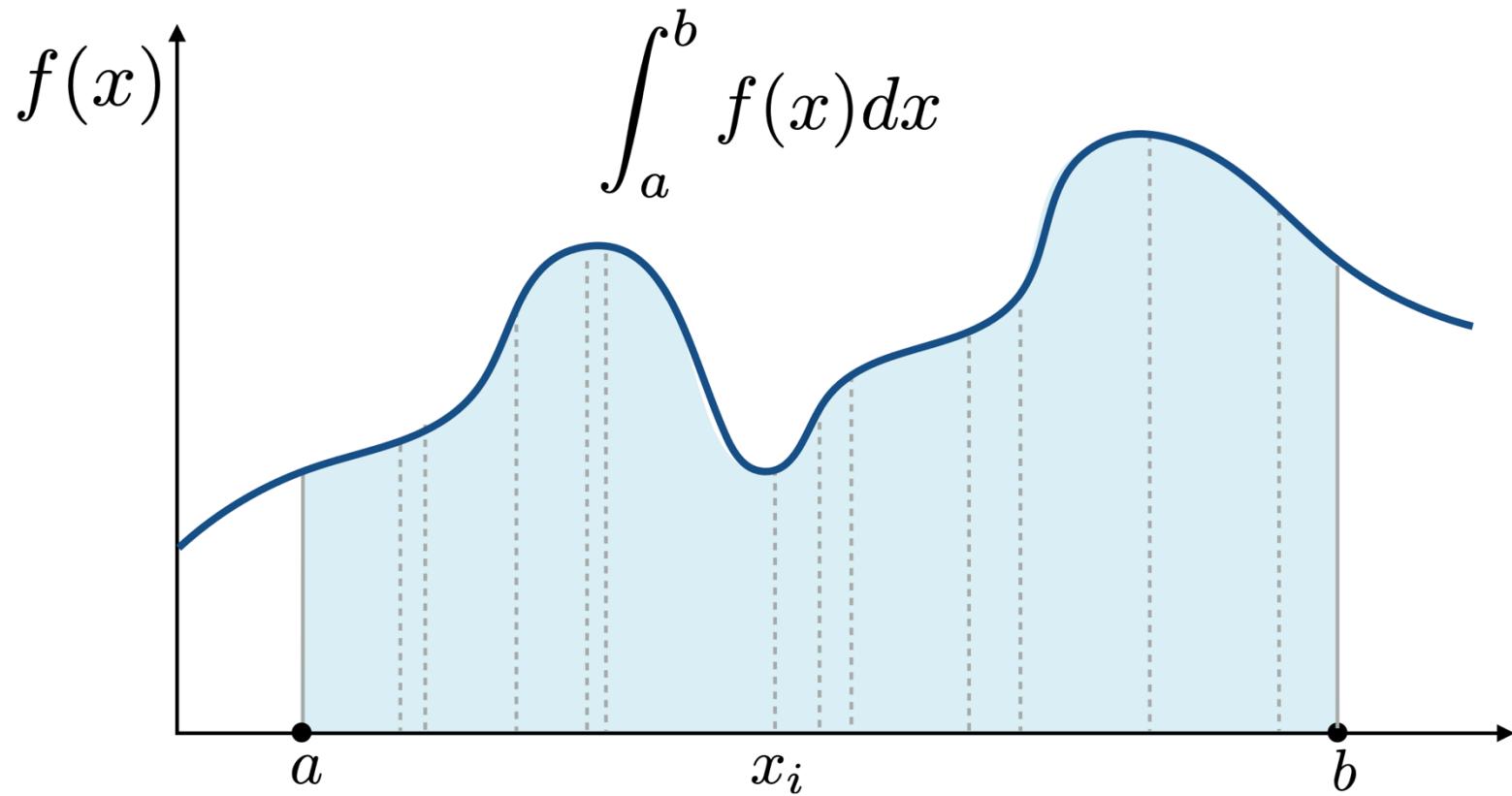
□为什么要使用蒙特卡罗积分？

□我们想求解一个定积分，但解析解太难求了



# 蒙特卡罗积分

- 蒙特卡罗积分怎么使用?
- 通过对函数值的随机样本求平均值来估计函数的积分



# 蒙特卡罗积分

口给定函数  $f(x)$  的定积分的蒙特卡罗估计值

定积分  
Definite integral

$$\int_a^b f(x) dx$$

随机变量  
Random variable

$$X_i \sim p(x)$$

蒙特卡罗估计值  
Monte Carlo estimator

$$F_N = \frac{1}{N} \sum_{i=1}^N \frac{f(X_i)}{p(X_i)}$$

# 例子：均匀蒙特卡罗估计值

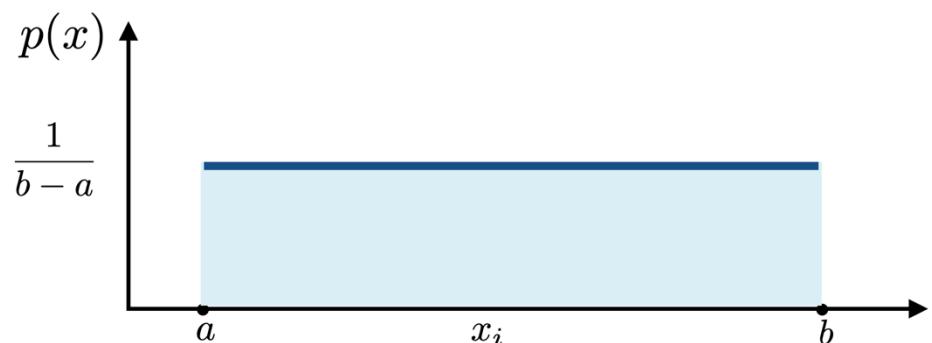
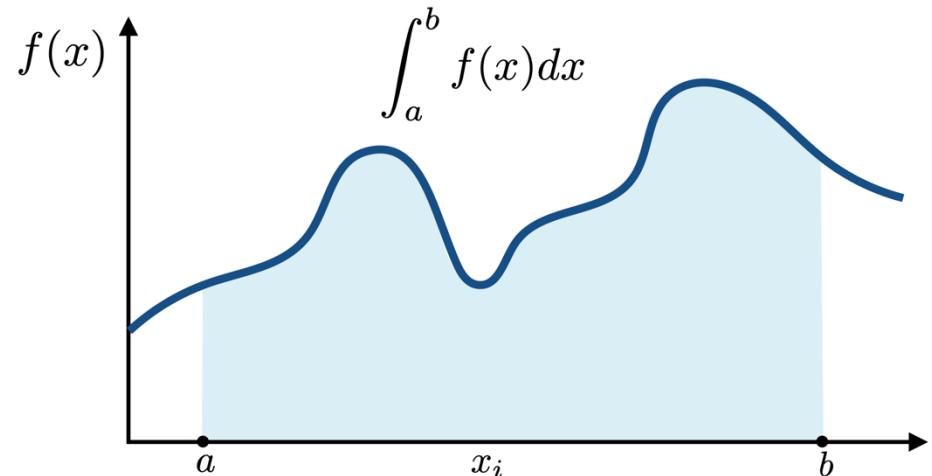
□ 均匀随机变量采样

$$X_i \sim p(x) = C \text{ (constant)}$$

$$\int_a^b p(x) dx = 1$$

$$\Rightarrow \int_a^b C dx = 1$$

$$\Rightarrow C = \frac{1}{b-a}$$



# 例子：均匀蒙特卡罗估计值

口给定函数  $f(x)$  的定积分的蒙特卡罗估计值

定积分  
Definite integral

$$\int_a^b f(x) dx$$

随机变量  
Random variable

$$X_i \sim p(x) = \frac{1}{b - a}$$

蒙特卡罗估计值  
Monte Carlo estimator

$$F_N = \frac{b - a}{N} \sum_{i=1}^N f(X_i)$$

# 蒙特卡罗积分

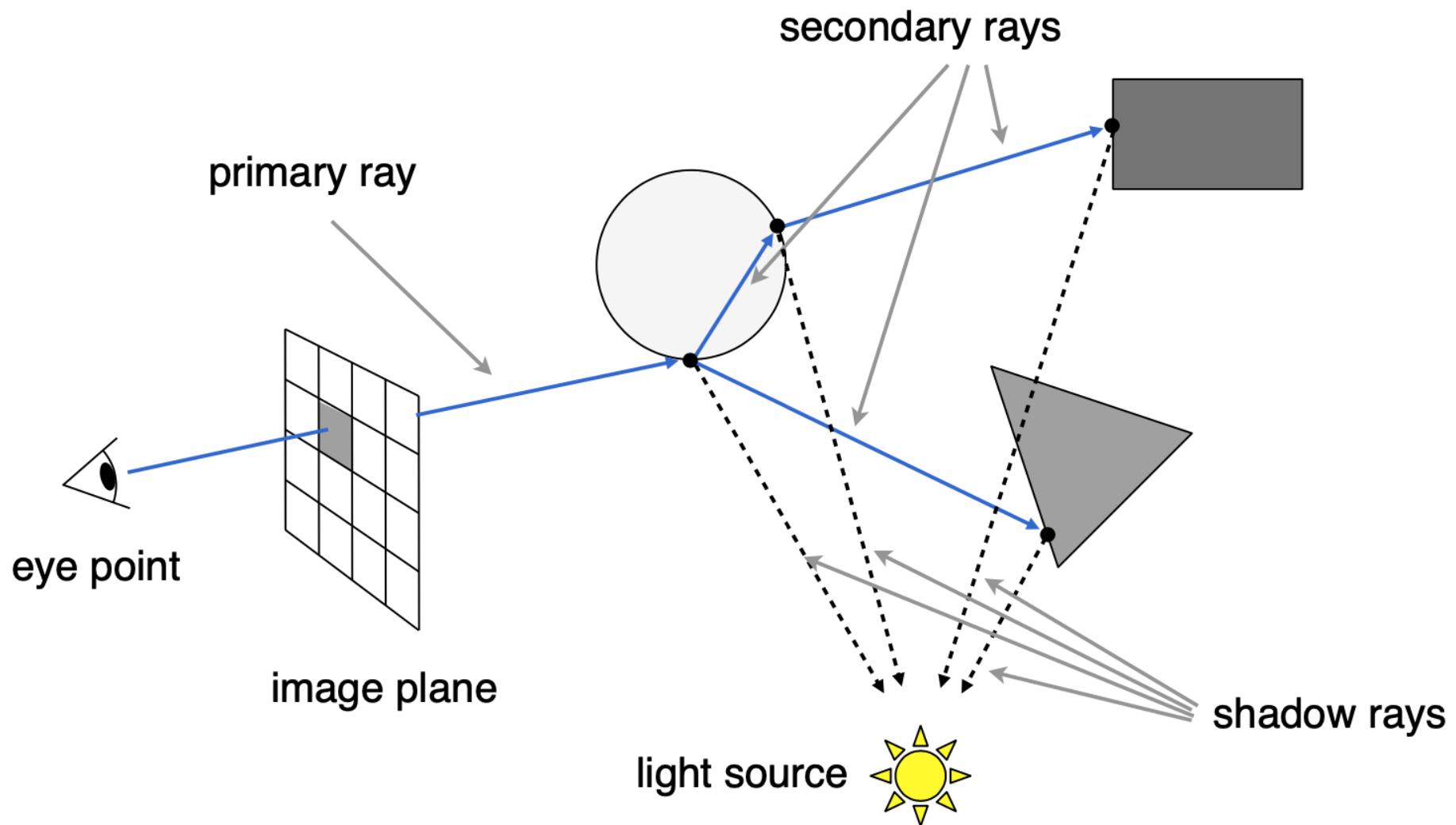
$$\int f(x) dx = \frac{1}{N} \sum_{i=1}^N \frac{f(X_i)}{p(X_i)} \quad X_i \sim p(x)$$

口样本越多，方差越小

口在  $x$  上采样，则在  $x$  上积分（而不是其他变量）

# 求解渲染方程 (路径追踪) Path Tracing

# 回顾：递归光线追踪 (Witted-Style)



# 动机：Whitted-style 的光线追踪

□ Whitted-style 光线追踪的模型简化

- 假设光线反射/折射总是镜像的
- 不处理间接的光照效果 (例如通过折射或反射光照形成的间接光照)

□ 这些简化合理吗？

□ 让我们逐步改进 Whitted-style 的光线跟踪，由此过渡到我们的路径跟踪算法！

# Whitted-style 光线追踪：问题 1

□ Whitted-style 光线追踪假设光线是完美的反射/折射，因此不能处理粗糙有光泽的表面 (比如右下角的磨砂质感)



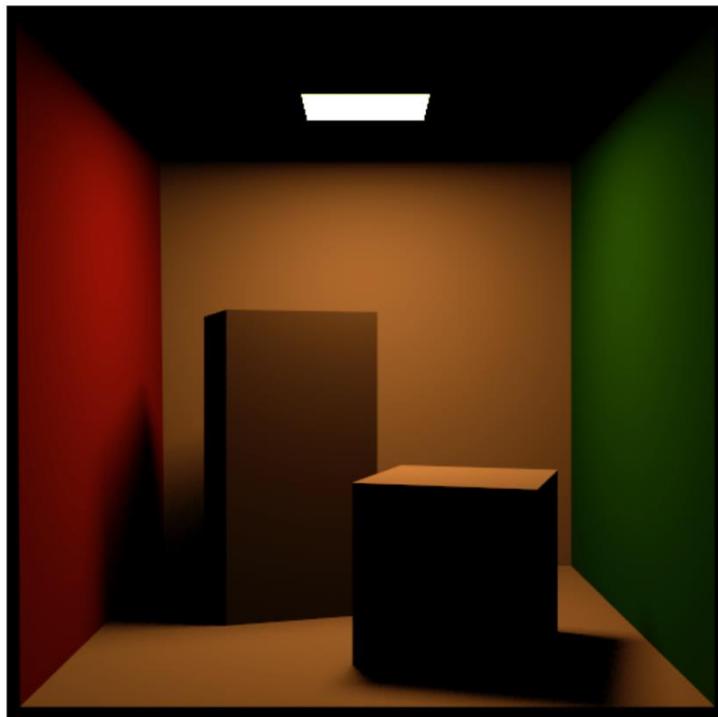
Mirror reflection



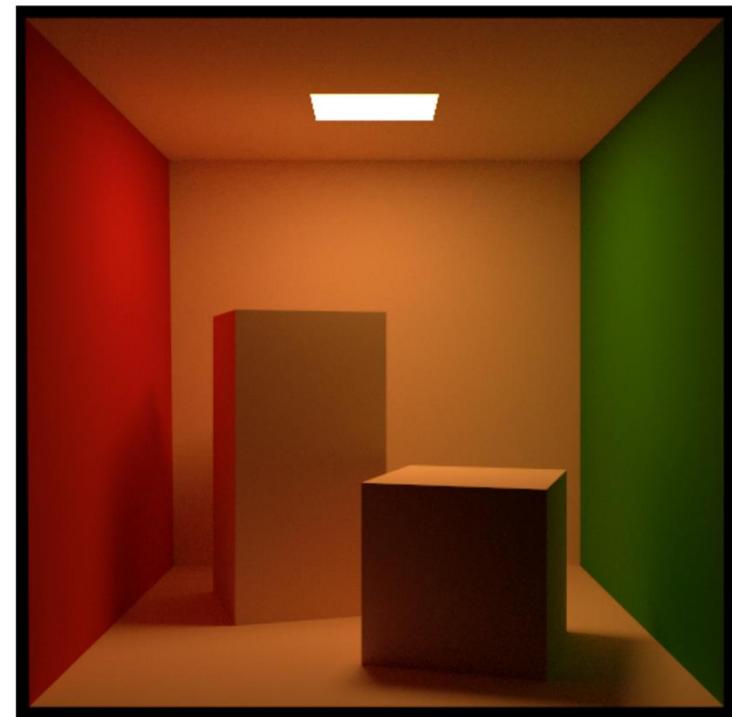
Glossy reflection

# Whitted-style 光线追踪：问题 2

□ Whitted-style 光线追踪主要处理直接光照，未考虑光线在多个物体之间反射或折射后再照射到其他物体上的情况



Path traced:  
direct illumination



Path traced:  
global illumination

# Whitted-style 光线追踪是错误的！

- Whitted-style 光线追踪不能很好地反映真实的光线
- 渲染方程则可以！

$$L_r(x, \omega_r) = L_e(x, \omega_r) + \int_{\Omega^+} L_i(x, \omega_i) f_r(x, \omega_i, \omega_r) (n \cdot \omega_i) d\omega_i$$

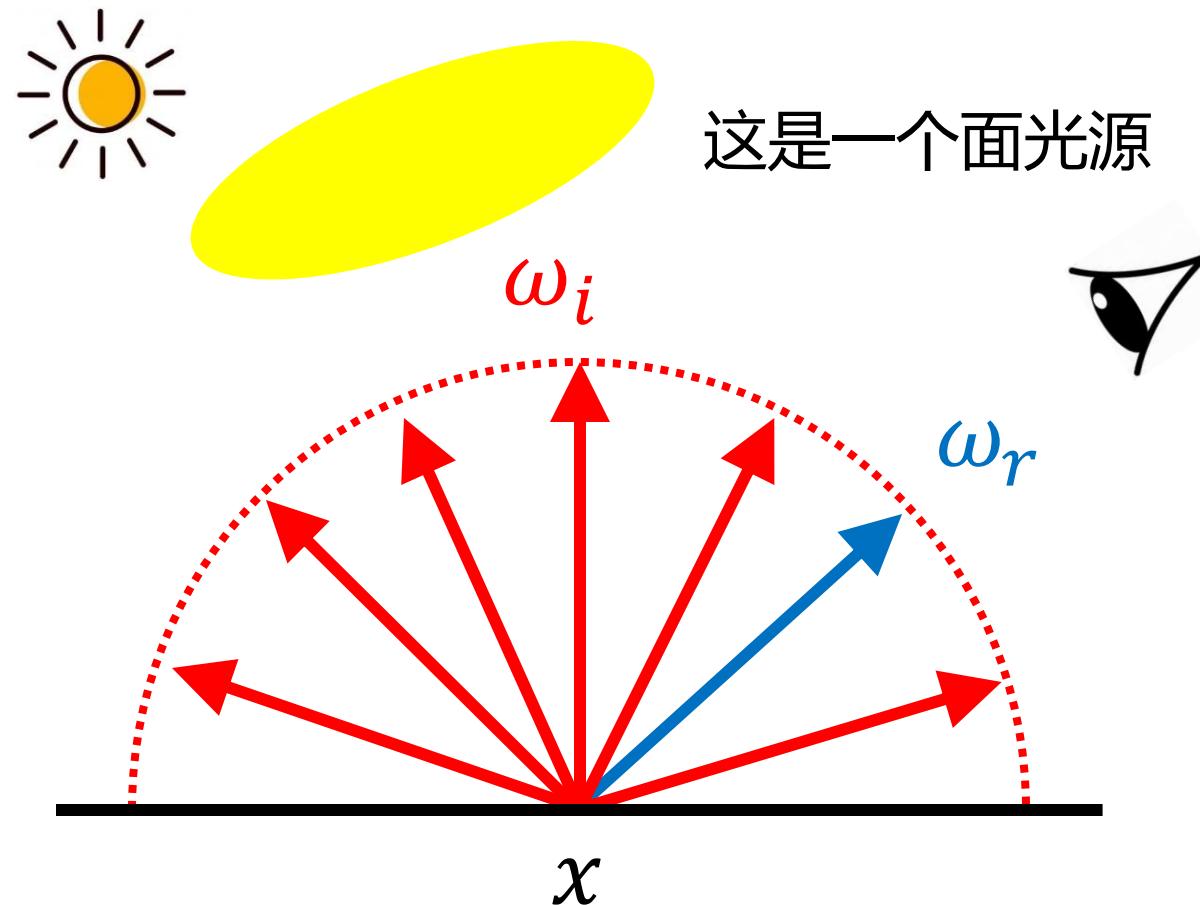
- 但是它包含两个问题：

- 求解半球上的积分
- 递归方程

- 如何数值地求解一个积分？

# 一个简单的蒙特卡罗解法

假设我们想在**直接照明**的场景中渲染一个像素 (点  $x$ )，即只考虑来自光源的辐亮度 (不考虑其他物体表面的反射)



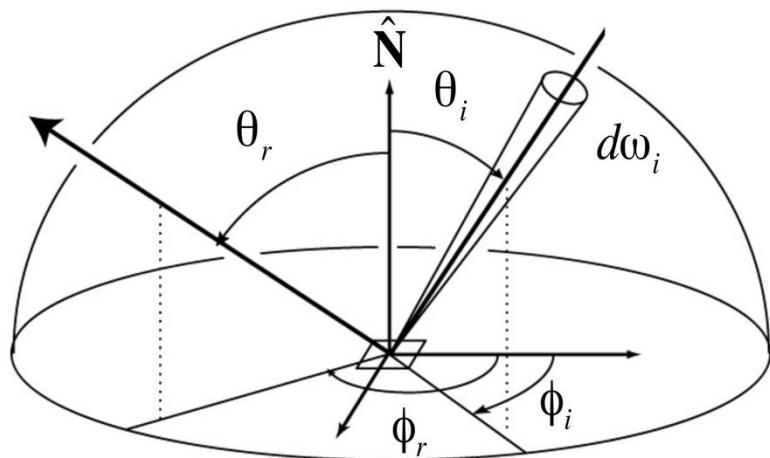
# 一个简单的蒙特卡罗解法

回到我们的渲染方程 (假设物体本身不发光)

$$L_r(x, \omega_r) = \int_{\Omega^+} L_i(x, \omega_i) f_r(x, \omega_i, \omega_r) (\mathbf{n} \cdot \omega_i) d\omega_i$$

虽然看起来很复杂，但也只是对半球区域做积分

因此，我们当然可以用蒙特卡罗积分法对其进行求解



# 一个简单的蒙特卡罗解法

口我们想计算在点  $x$ , 面向相机方向  $\omega_r$  的辐亮度  $L_r(x, \omega_r)$

$$L_r(x, \omega_r) = \int_{\Omega^+} L_i(x, \omega_i) f_r(x, \omega_i, \omega_r) (n \cdot \omega_i) d\omega_i$$

口蒙特卡罗积分法 
$$\int_a^b f(x) dx \approx \frac{1}{N} \sum_{k=1}^N \frac{f(X_k)}{p(X_k)} \quad X_k \sim p(x)$$

口我们的  $f(x)$  是什么?  $L_i(x, \omega_i) f_r(x, \omega_i, \omega_r) (n \cdot \omega_i)$

口我们的是  $p(x)$  什么?  $p(\omega_i) = \frac{1}{2\pi}$

假设我们对半球面进行均匀采样

# 一个简单的模特卡罗解法

□ 模特卡罗积分结果

$$\begin{aligned} L_r(x, \omega_r) &= \int_{\Omega^+} L_i(x, \omega_i) f_r(x, \omega_i, \omega_r) (\mathbf{n} \cdot \omega_i) d\omega_i \\ &\approx \frac{1}{N} \sum_{i=1}^N \frac{L_i(x, \omega_i) f_r(x, \omega_i, \omega_r) (\mathbf{n} \cdot \omega_i)}{p(\omega_i)} \end{aligned}$$

□ 这个公式表示什么？

□ 一种正确的、适用于**直接照明**的着色算法

# 一个简单的蒙特卡罗解法

$$L_r(x, \omega_r) \approx \frac{1}{N} \sum_{i=1}^N \frac{L_i(x, \omega_i) f_r(x, \omega_i, \omega_r) (n \cdot \omega_i)}{p(\omega_i)}$$

shade(p, wr )

Randomly choose N directions wi~pdf

Lr = 0.0

For each wi

Trace a ray r(p, wi)

If ray r hit the light

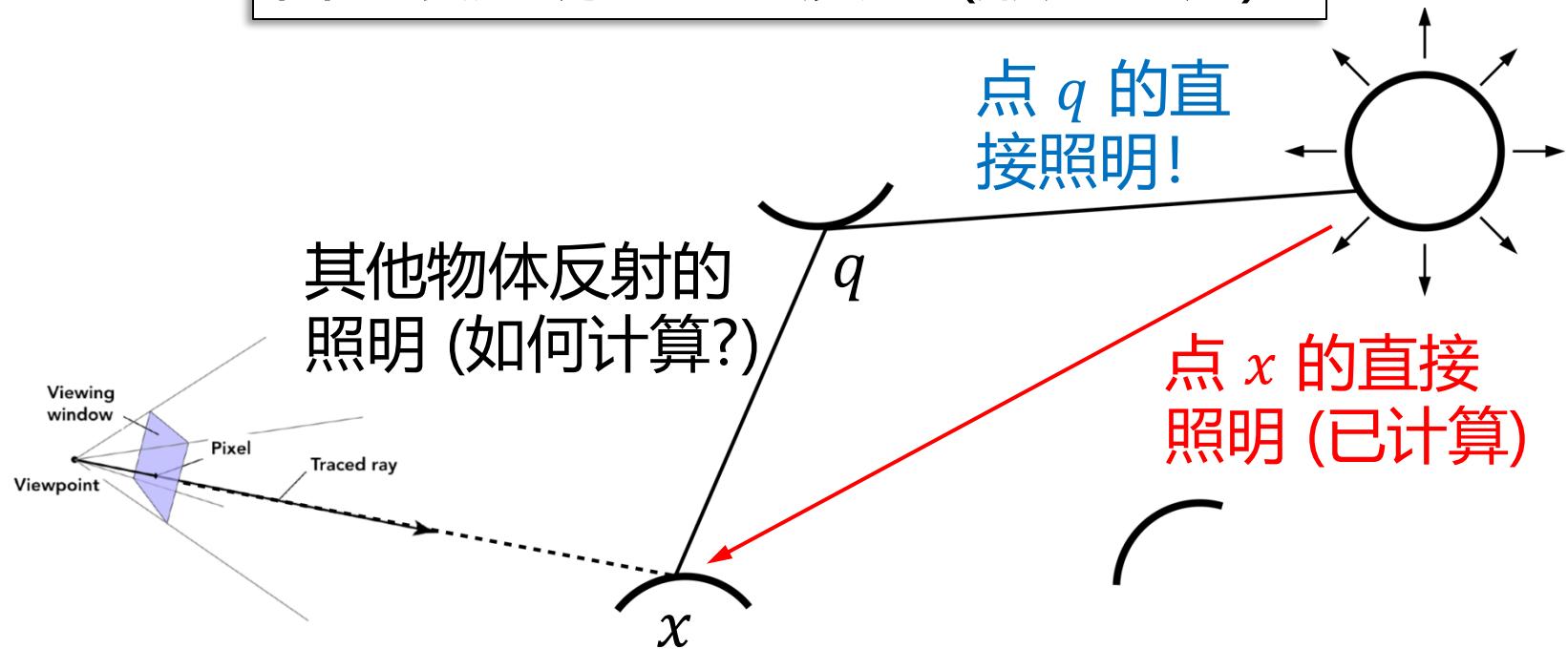
Lr += (1 / N) \* L\_i \* f\_r \* cosine / pdf(wi)

Return Lr

# 全局光照 Global Illumination

- 口向前一步：刚刚只计算了点  $x$  来自直接照明的能量
- 口如果光线击中其他物体（点  $q$ ）并反射到点  $x$ ，如何计算此部分的能量？

点  $x$  来自点  $q$  的能量即为点  $q$  对来自直接照明能量的反射（形成递归）



# 全局光照

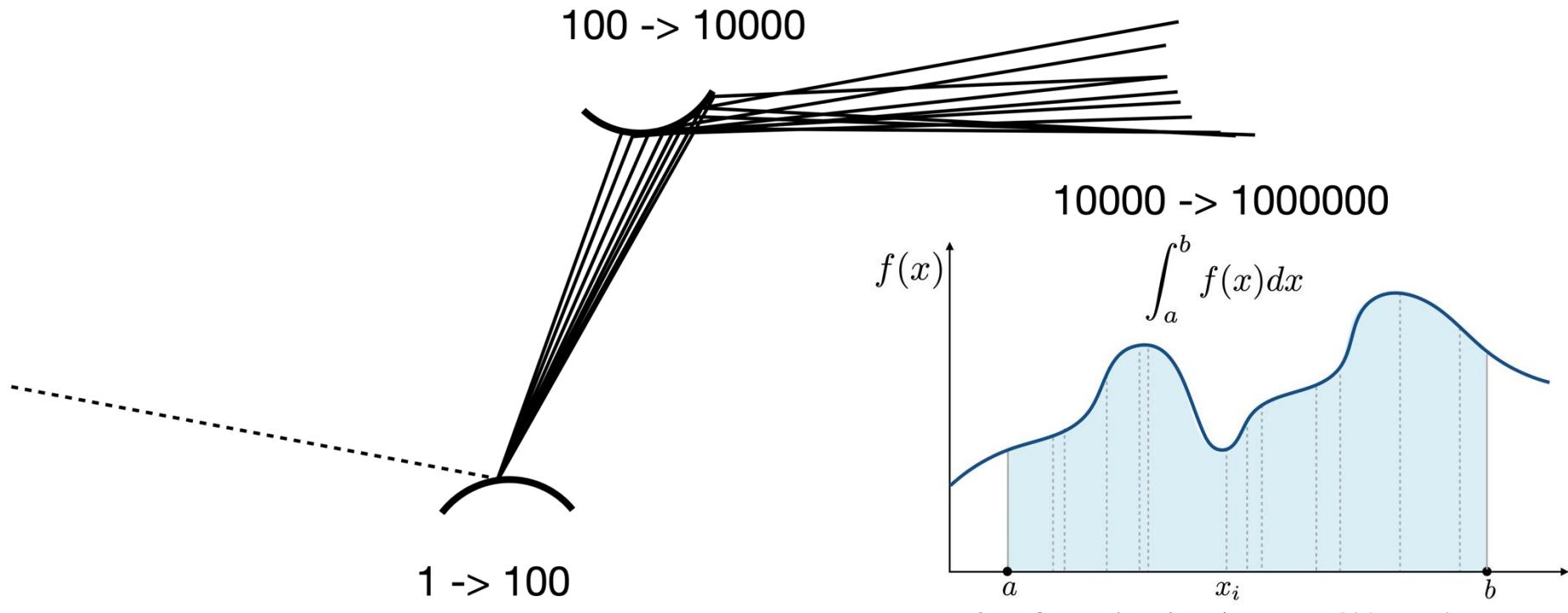
```
shade(p, wr)
    Randomly choose N directions wi~pdf
    Lr = 0.0
    For each wi
        Trace a ray r(p, wi)
        If ray r hit the light
            Lr += (1 / N) * L_i * f_r * cosine / pdf(wi)
        Else If ray r hit an object at q
            Lo += (1 / N) * shade(q, -wi) * f_r * cosine
            / pdf(wi)
    Return Lr
```

这样是否就完成了? No. 至少还有 2 个问题!

# 路径追踪

□问题 1：随着光线反弹，光线的数量将爆炸

$$\#rays = N^{\#bounces}$$



相当于仅在这里采样一次

□为了避免光线爆炸，当且仅当  $N = 1$ ！

# 路径追踪

从现在起，我们假设在每个着色点仅跟踪一条反射光线：

shade( $p, wr$ )

Randomly choose **ONE** direction  $wi \sim pdf(w)$

Trace a ray  $r(p, wi)$

If ray  $r$  hit the light

    Return  $L_i * f_r * cosine / pdf(wi)$

Else If ray  $r$  hit an object at  $q$

    Return  $shade(q, -wi) * f_r * cosine / pdf(wi)$

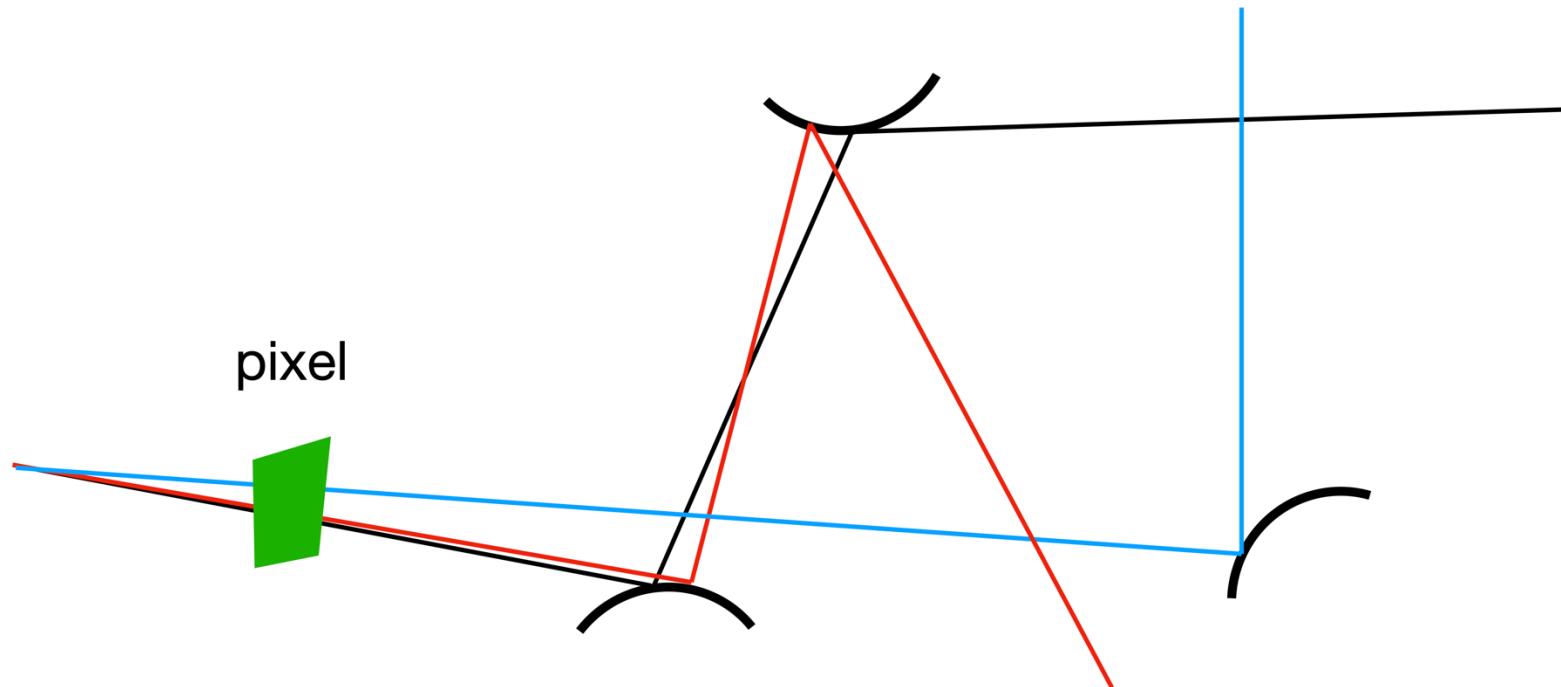
这就是**路径追踪 (Path tracing)**

# 光线生成

□但是这种做法会充满噪声！

□没关系，只需在每个像素中追踪更多光线，并平均不同光线的辐亮度

□这里每条光线在每个着色点仅反射一次，区别于反射多次的情况（会形成光线爆炸）



# 光线生成算法

与光线投射很类似

```
ray_generation(camPos, pixel)
```

Uniformly choose N sample positions within the pixel

```
pixel_radiance = 0.0
```

For each sample in the pixel

```
    Shoot a ray r(camPos, cam_to_sample)
```

```
    If ray r hit the scene at p
```

```
        pixel_radiance += 1 / N * shade(p, sample_to_cam)
```

Return pixel\_radiance

# 光线追踪

□ 将目前的情况结合起来

**shade(p, wr)**

Randomly choose ONE direction  $w_i \sim \text{pdf}(w)$

Trace a ray  $r(p, w_i)$

If ray  $r$  hit the light

    Return  $L_i * f_r * \cosine / \text{pdf}(w_i)$

Else If ray  $r$  hit an object at  $q$

    Return **shade(q, -wi) \* f\_r \* cosine / pdf(wi)**



递归!

□ 这样是否就完成了？

□ No, 这个递归算法永远不会结束！

# 路径追踪

口真实情况是：现实物理世界中，光线确实不会停止反弹  
口限制一条光线的反弹次数，相当于削减了它的能量

3 bounces



# 路径追踪

口真实情况是：现实物理世界中，光线确实不会停止反弹  
口限制一条光线的反弹次数，相当于削减了它的能量

17 bounces



# 解决方法：俄罗斯轮盘赌

- 俄罗斯轮盘赌 (Russian Roulette, RR)
- With probability  $0 < P < 1$ , you can survive
- With probability  $1-P$ , otherwise



假设枪里有两颗子弹，则生存的概率为  $P=4/6$

以一定的概率停止往下追踪

# 解决方法：俄罗斯轮盘赌

口之前我们总是向着色点射出光线，并获得着色结果  $L_r$

口现在我们手动设置一个概率  $p$  ( $0 < p < 1$ )

口在概率  $p$  下，我们射出光线并将着色结果除以  $p$ :  $L_r/p$

口在概率  $1 - p$  下，**不发射光线，着色结果为 0**

口这样，着色结果的期望值仍然为  $L_r$ ！

$$E = p * \left( \frac{L_r}{p} \right) + (1 - p) * 0 = L_r$$

# 解决方法：俄罗斯轮盘赌

**shade(p, wr)**

Manually specify a probability P\_RR

Randomly select ksi in a uniform dist. in [0, 1]

If (ksi > P\_RR) return 0.0;

Randomly choose ONE direction wi~pdf(w)

Trace a ray r(p, wi)

If ray r hit the light

    Return L\_i \* f\_r \* cosine / pdf(wi) / P\_RR

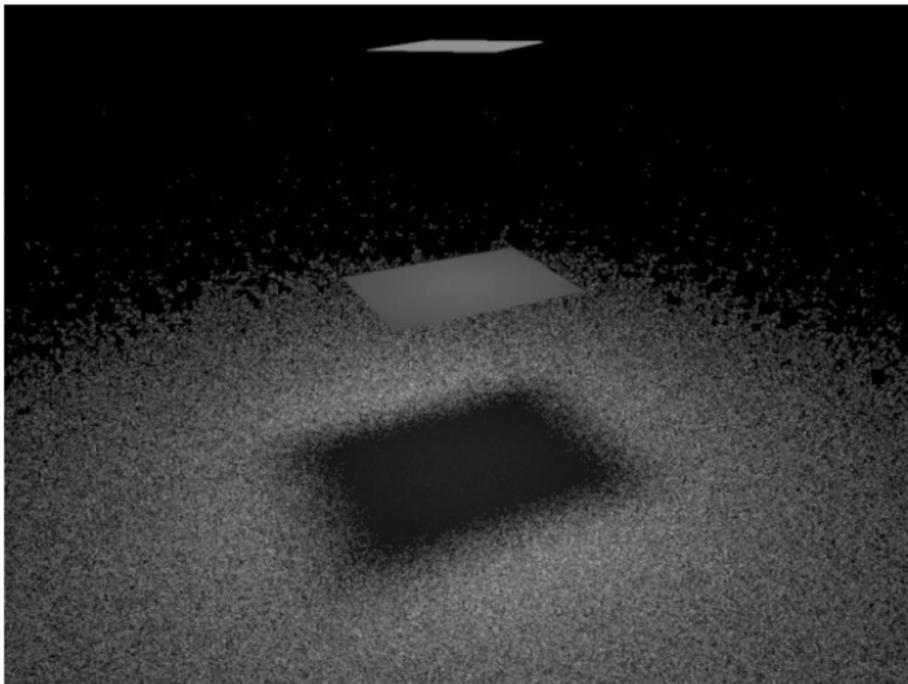
Else If ray r hit an object at q

    Return shade(q, -wi) \* f\_r \* cosine / pdf(wi) / P\_RR

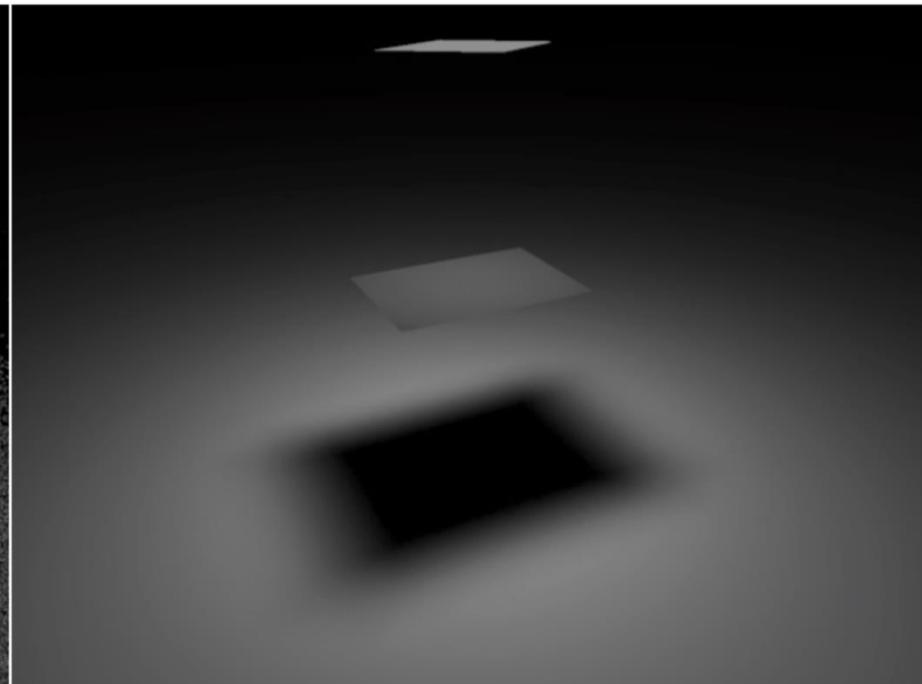
# 路径追踪

□ 现在我们已经有了一个**正确版本**的路径追踪！

□ 但并不是很高效



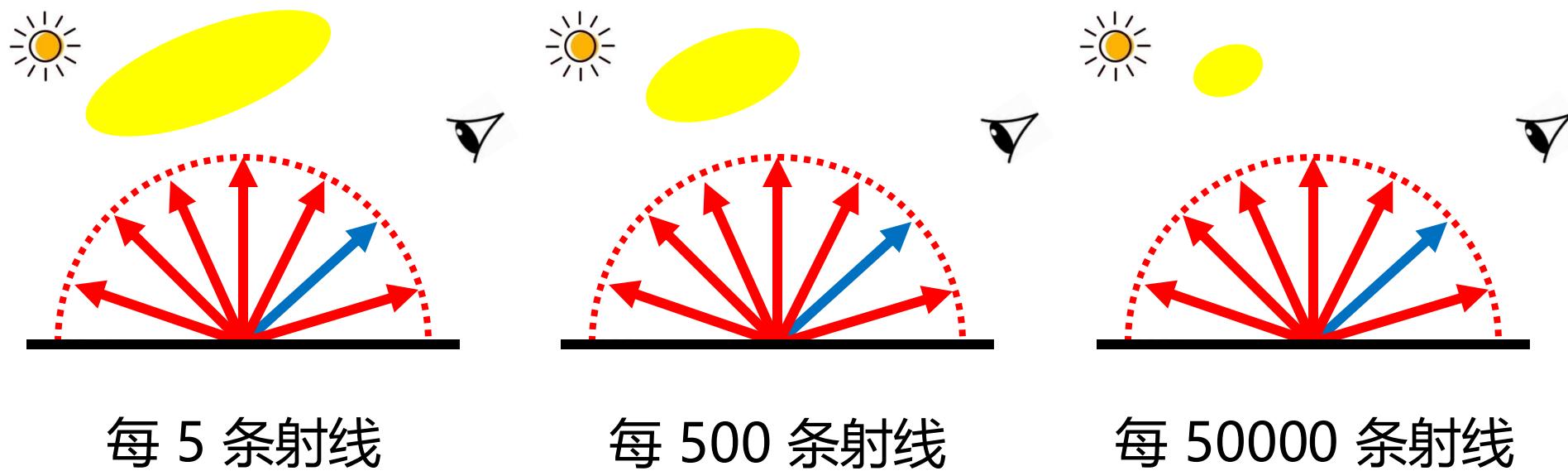
Low SPP (samples per pixel)  
noisy results



High SPP

# 低效的原因

□ 我们使用的采样函数  $p(\omega_i)$  为半球的均匀采样  $p(\omega_i) = \frac{1}{2\pi}$ , 只有当采样的射线打到光源或其他反射物体时才有效



□ 才会有一条射线射中光源!

□ 因此, 如果我们在着色点对半球进行**均匀采样**, 就会“浪费”很多光线

# 对光线进行采样

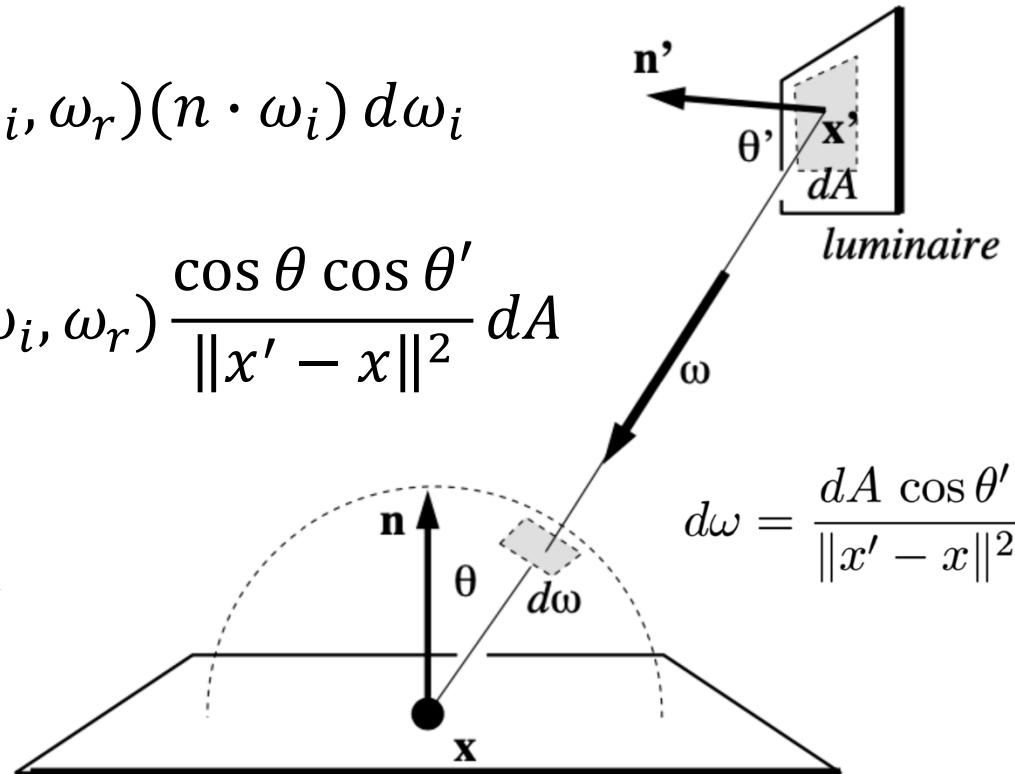
口蒙特卡罗积分法允许对任何变量进行采样，因此我们转而对光线进行采样（而不是半球体），这样便不会浪费光线

口先对光源采样，然后投影到半球上

$$\begin{aligned} L_r(x, \omega_r) &= \int_{\Omega^+} L_i(x, \omega_i) f_r(x, \omega_i, \omega_r) (\mathbf{n} \cdot \omega_i) d\omega_i \\ &= \int_A L_i(x, \omega_i) f_r(x, \omega_i, \omega_r) \frac{\cos \theta \cos \theta'}{\|x' - x\|^2} dA \end{aligned}$$

口现在我们是对面光源积分

口采样函数  $p(x) = \frac{1}{A}$



# 对光线进行采样

口最后一件事是：我们如何知道着色点是否被遮挡？

```
# Contribution from the light source.
```

```
L_dir = 0.0
```

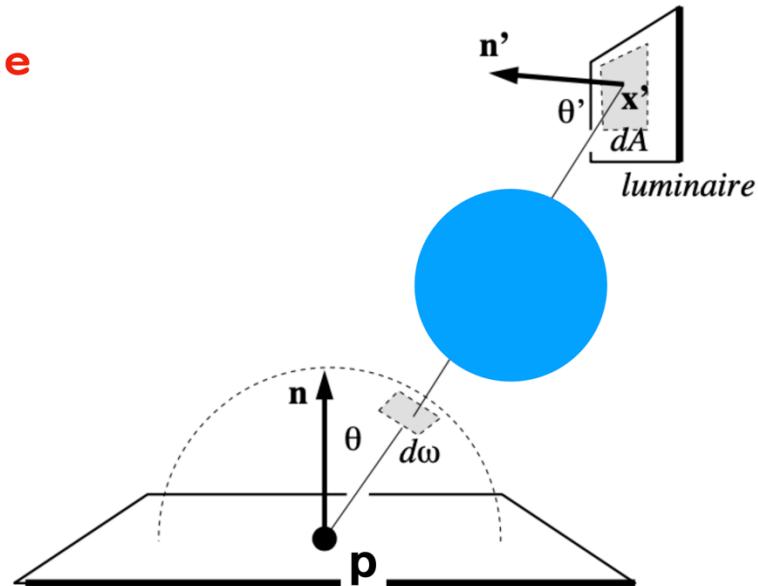
```
Uniformly sample the light at x' (pdf_light = 1 / A)
```

Shoot a ray from p to x'

If the ray is not blocked **in the middle**

```
L_dir = ...
```

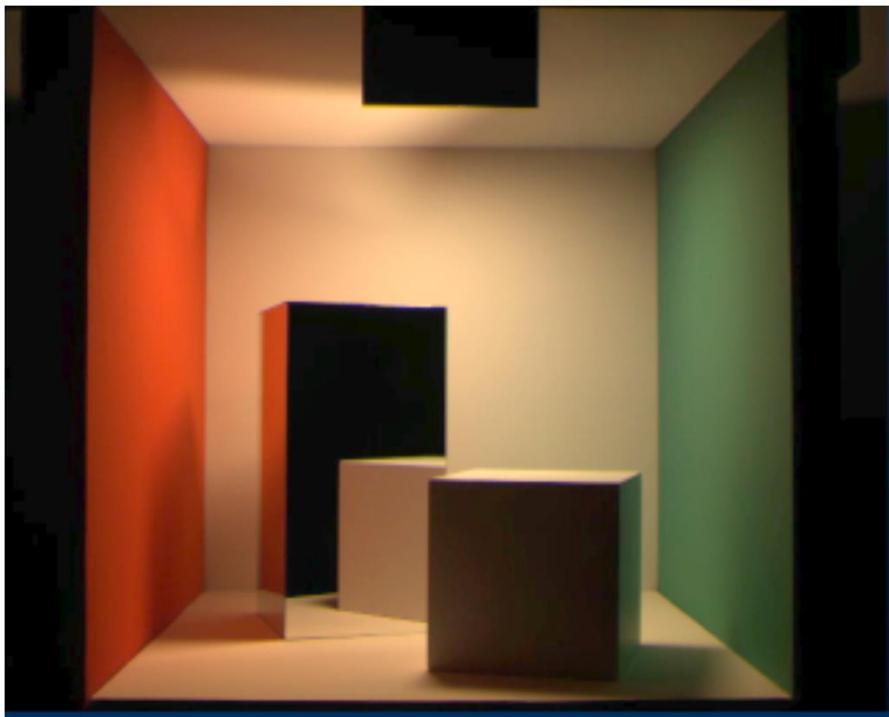
Now path tracing is finally done!



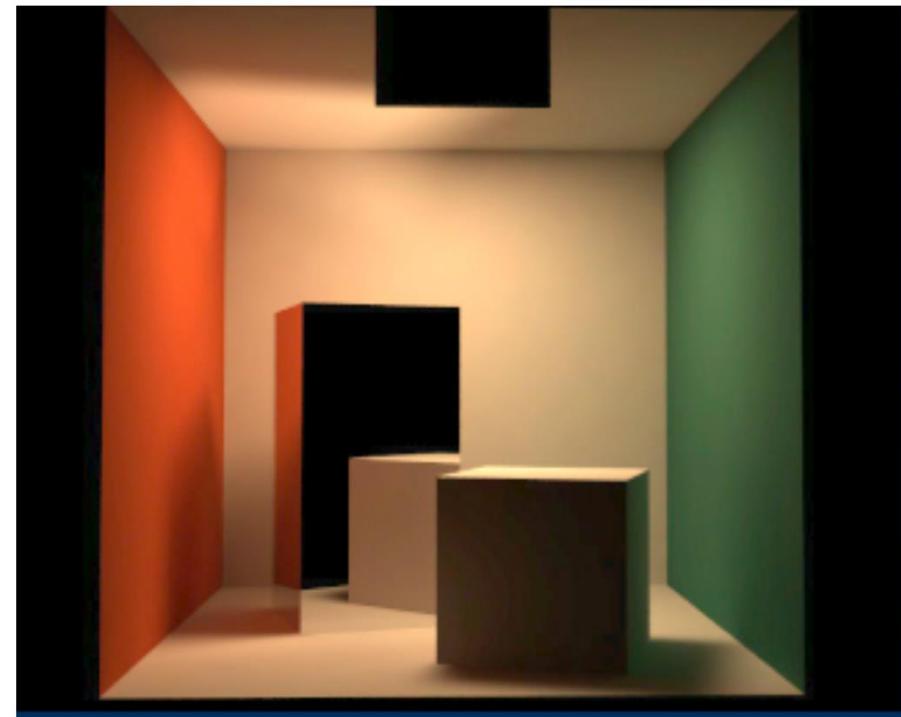
# 路径追踪是否正确？

口是的，几乎 100% 正确，能达到真实照片的水平

口下列哪张图片是照片，哪张是算法生成的？



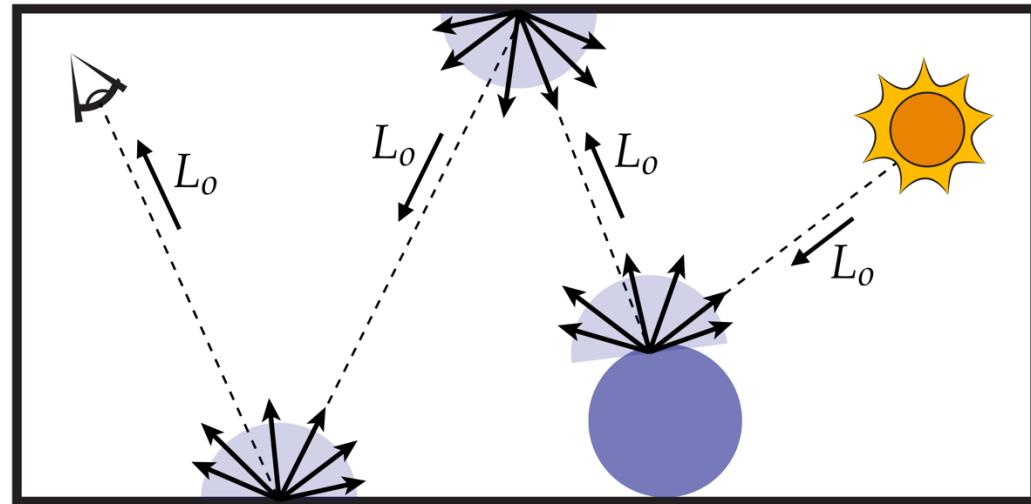
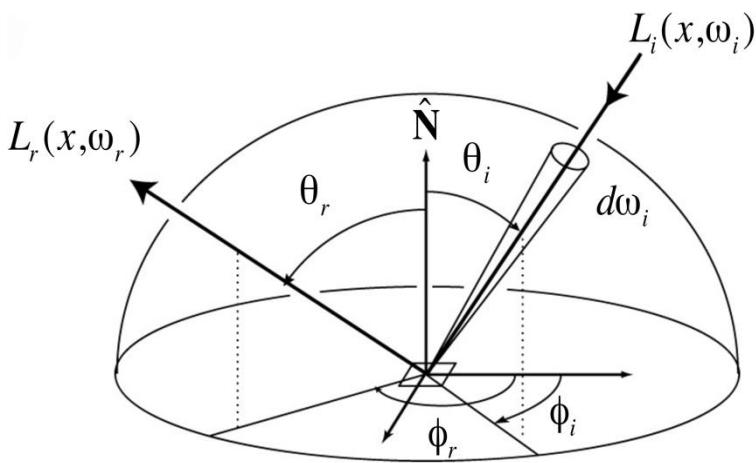
照片



路径追踪算法生成的  
全局光照

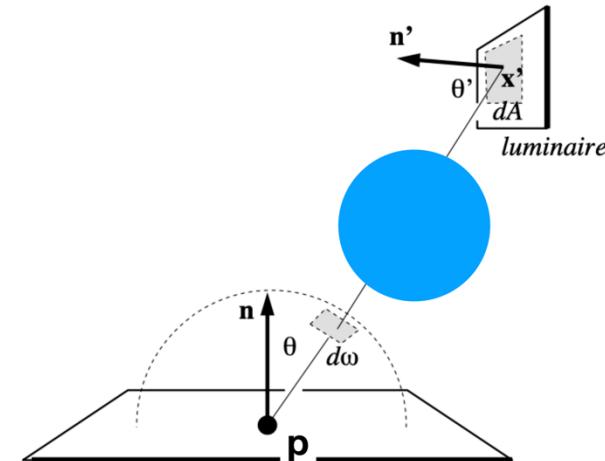
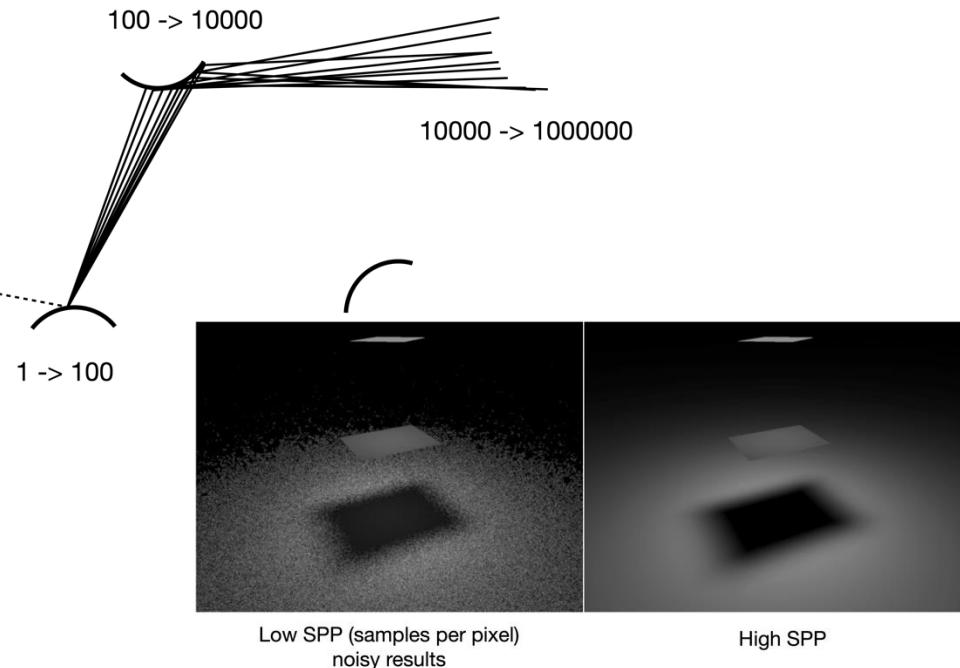
# 总结 – 路径追踪算法 (概念)

- 口通过辐亮度 (radiance) 精准计算光源达到物体的能量
- 口通过双向反射分布函数 (BRDF) 建模物体表面材质 (描述光线如何在物体表面任意方向的反射)
- 口能够考虑光线在多个物体之间的反射和折射



# 总结 – 路径追踪算法 (解法)

- 利用蒙特卡罗积分法近似求解
- 假定光线在物体表面仅反射一次 (即采样函数仅采样一次) 来避免光线爆炸，通过发射多条光线减少噪声
- 通过在光源处 (而不是半球体处) 采样来增加计算效率
- 通过碰撞检测判断着色点与光源之间是否存在遮挡物





中山大學 软件工程学院  
SUN YAT-SEN UNIVERSITY SCHOOL OF SOFTWARE ENGINEERING

谢谢

陈壮彬  
软件工程学院  
[chenzhb36@mail.sysu.edu.cn](mailto:chenzhb36@mail.sysu.edu.cn)