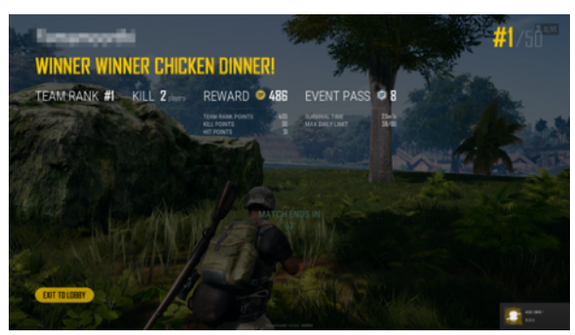
Advertisement

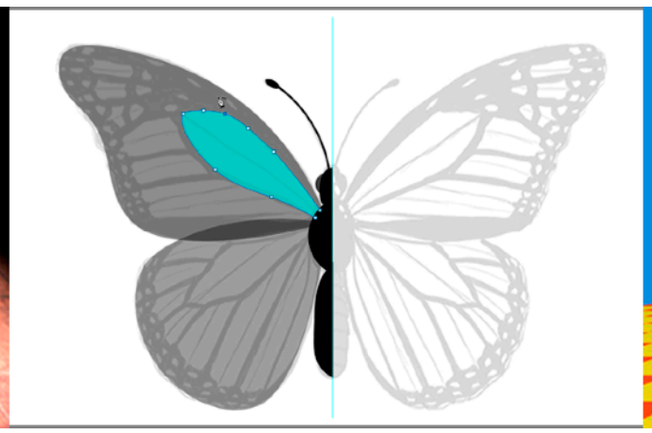
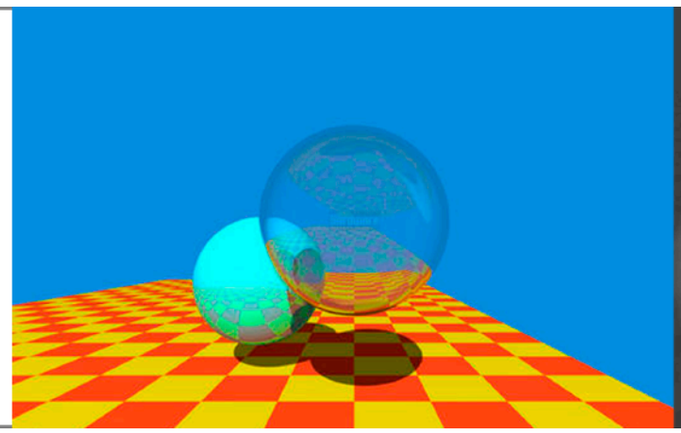
- Prof. Qi Sun from NYU
  - VR / AR / perception / RT graphics, http://qisun.me
- Fall 2020 - Spring 2021 (remote is fine)
  - 2-3 research interns
  - 1 postdoc / visiting scholar
- See details on GAMES website / WeChat group
- Send resume to qisun@nyu.edu now!



Introduction to Computer Graphics

GAMES101, Lingqi Yan, UC Santa Barbara

Lecture 16:
Ray Tracing 4
(Monte Carlo Path Tracing)


http://www.cs.ucsb.edu/~lingqi/teaching/games101.html

Announcements

- Regarding the difficulty of the last lecture
  - Modern Graphics does require it
- We are working on final project ideas
  - But again, welcome to come up with your own
- Today's lecture is ~~easy~~ ~~normal~~ a little bit hard
  (Next lectures will be much easier!)

Last Lecture

- Radiometry cont.
- Light transport
  - The reflection equation
  - The rendering equation
- Global illumination
- Probability review

Today

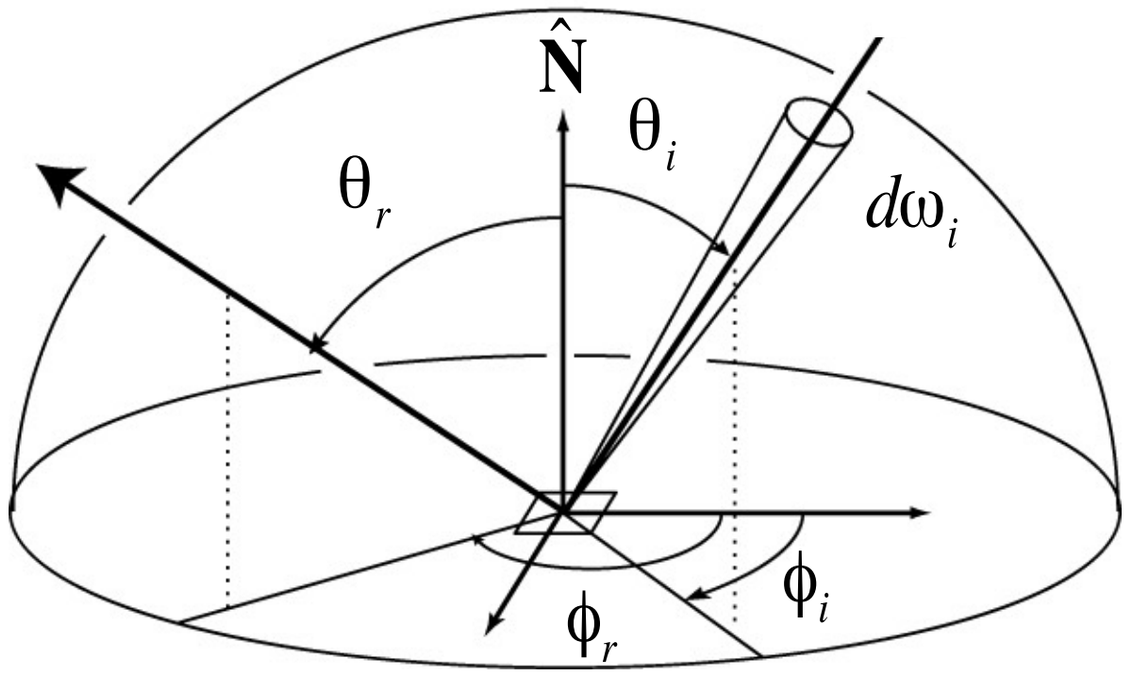
- A Brief Review
- Monte Carlo Integration
- Path Tracing

Review - The Rendering Equation

- Describing the light transport

$$L_o(p,\omega_o) = L_e(p,\omega_o) + \int_{\Omega^+} L_i(p,\omega_i) f_r(p,\omega_i,\omega_o)(n\cdot\omega_i)\,\mathrm{d}\omega_i$$

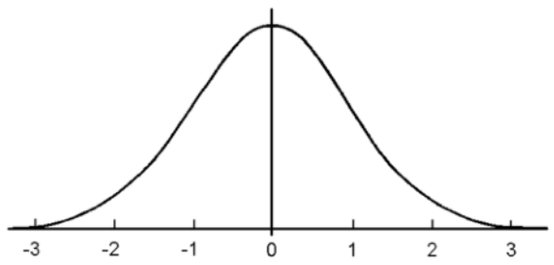
这个方程的积分部分描述了着色点接收到直接和间接光照后这个点反射出去的光，而难点在于两个：一是积分不好求，二是来自其他表面的光照 Li 实际上是另一个渲染方程



Review - Probabilities

- Continuous Variable and Probability Density Functions

$$X \sim p(x)$$



- Understanding: randomly pick an X -> more likely to be a number closer to 0 (in this case)

Conditions on p(x): $p(x) \geq 0 \text{ and } \int p(x)\,dx = 1$

Expected value of X: $E[X] = \int x\,p(x)\,dx$

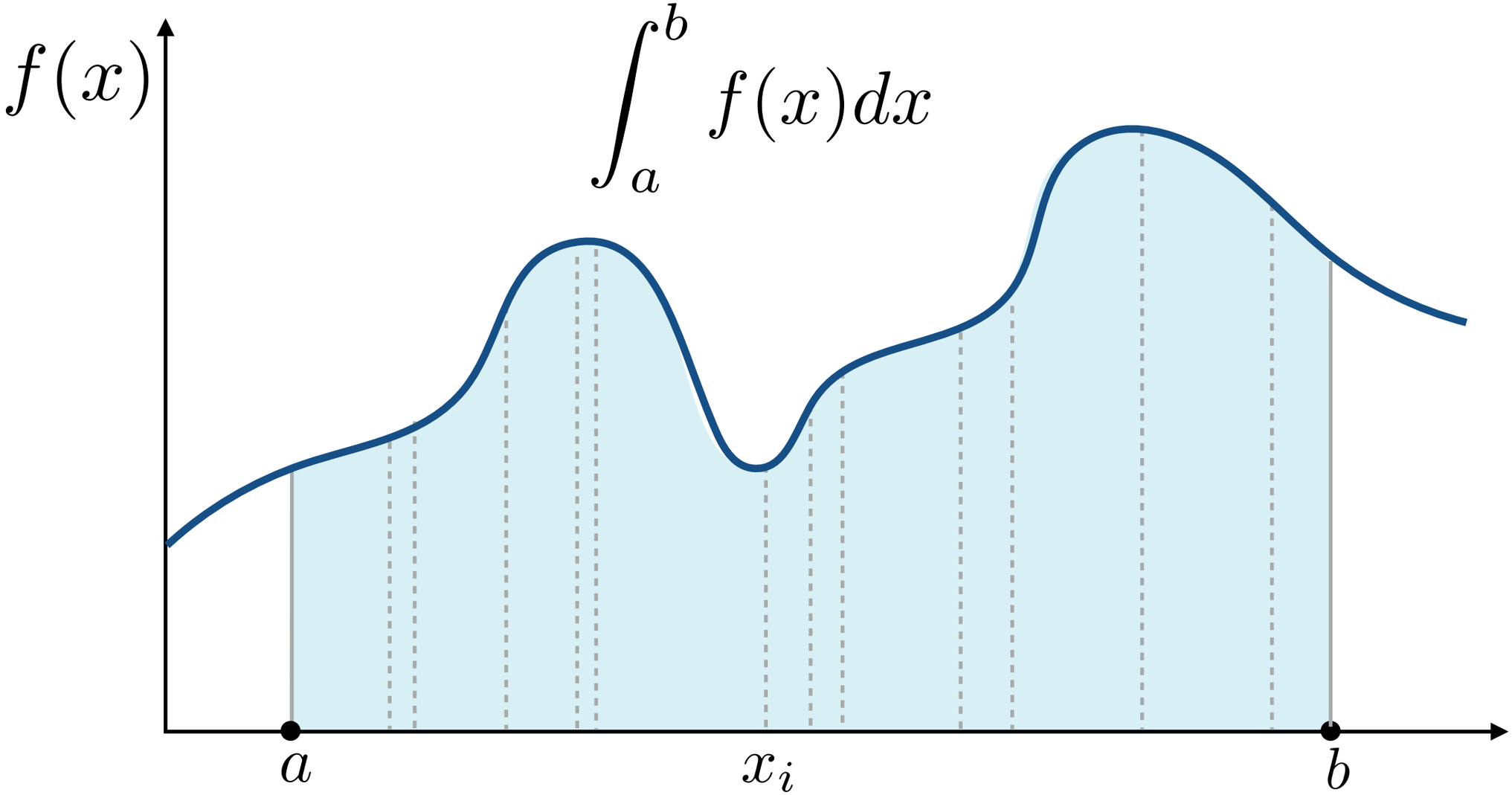
数学期望，是随机变量多次实验趋于的稳定值

Monte Carlo Integration

解决积分难求的问题

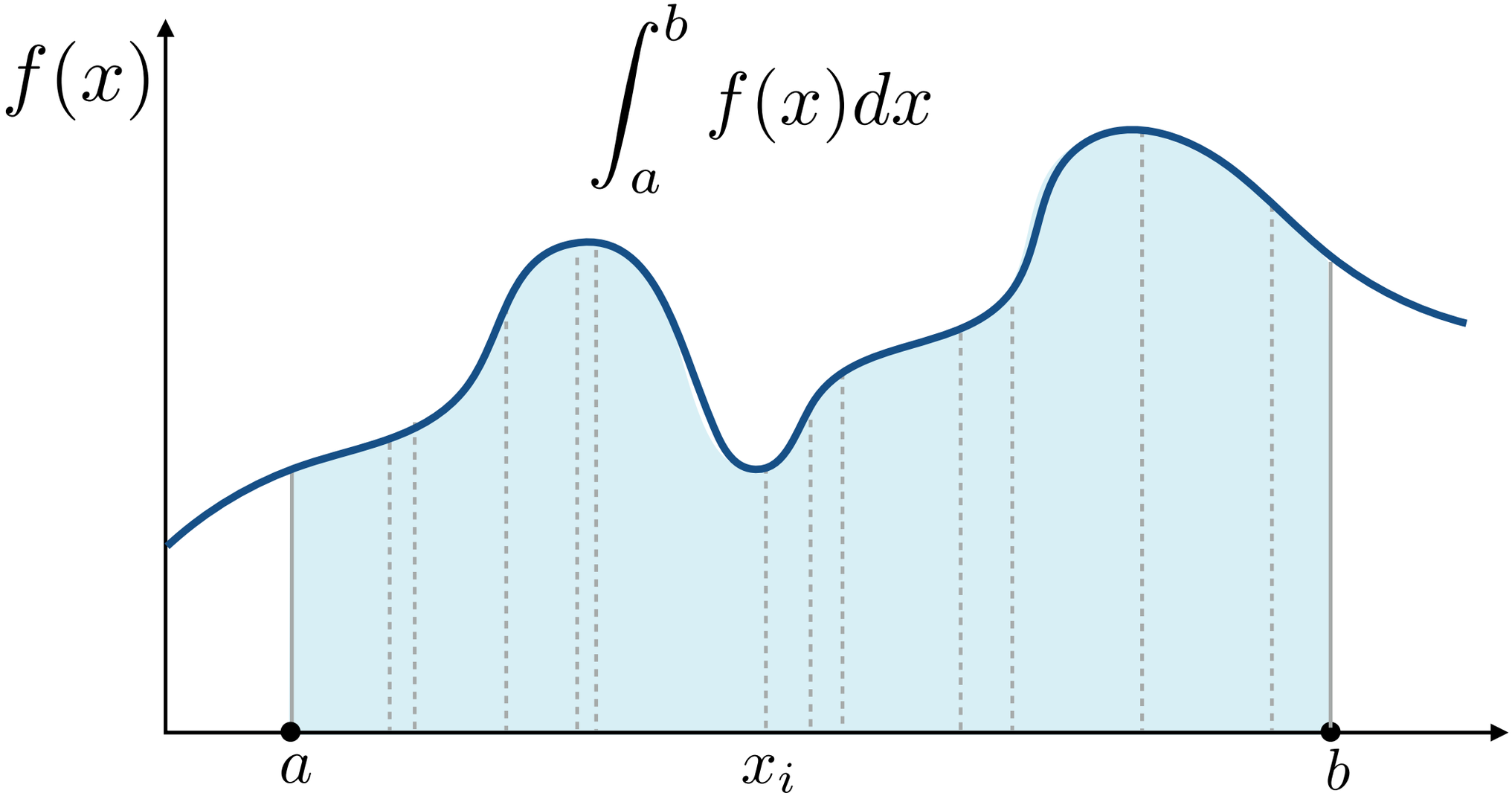
Monte Carlo Integration

Why: we want to solve an integral, but it can be too difficult to solve analytically.



Monte Carlo Integration

What & How: estimate the integral of a function by averaging random samples of the function's value.



Monte Carlo Integration

Let us define the Monte Carlo estimator for the definite integral of given function $f(x)$

Definite integral

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

Random variable

$$X_i \sim p(x)$$

Monte Carlo estimator

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

在区间ab采样N次，这个积分最后的结果约等于这个采样结果的平均值（也就是其期望E(X)）。而1/N表示平均值， p(X) 为这个采样所服从的概率密度函数

Example: Uniform Monte Carlo Estimator

在积分区域上随机采样，若采样方式为均匀采样（采样pdf为均匀分布），则认为样本和采样值的乘积 xf(x) 为单次采样的积分结果，随后多次采样取平均，得到最终近似结果

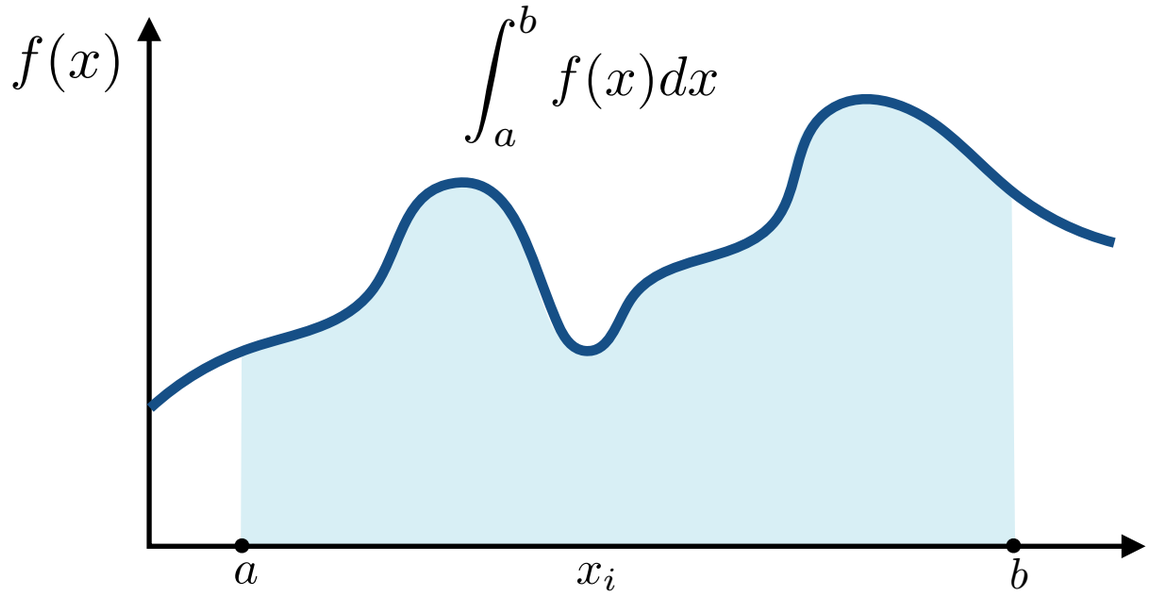
Uniform random variable

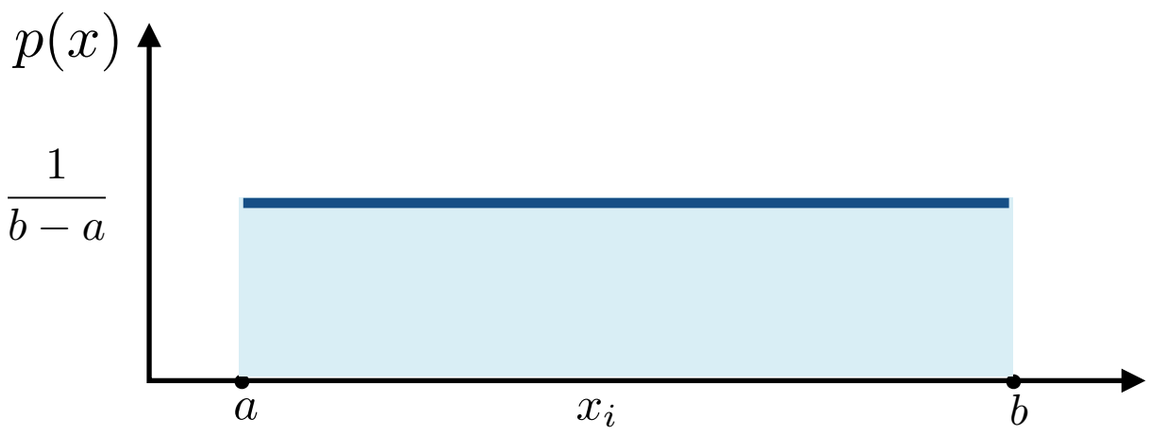
$$X_i \sim p(x) = C \quad \textbf{(constant)}$$

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

$$\Longrightarrow \int_a^b C\,dx = 1$$

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





Example: Uniform Monte Carlo Estimator

Let us define the Monte Carlo estimator for the definite integral of given function $f(x)$

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

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

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

均匀分布采样 蒙特卡洛积分就是求了一个面积的平均值

Monte Carlo Integration

$$\int f(x)\,\mathrm{d}x = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)} \qquad X_i \sim p(x)$$

Some notes:

- The more samples, the less variance.
- Sample on x, integrate on x.

Path Tracing

Motivation: Whitted-Style Ray Tracing

Whitted-style ray tracing:

- Always perform specular reflections / refractions
- Stop bouncing at diffuse surfaces

Whitted-Style光线追踪的做法是，光线在镜面反射表面弹射，而在漫反射表面停止，这显然是不符合现实的

Are these simplifications reasonable?

High level: let’s progressively improve upon Whitted-Style Ray Tracing and lead to our path tracing algorithm!

Whitted-Style Ray Tracing: Problem 1

Where should the ray be reflected for glossy materials?

Mirror reflection

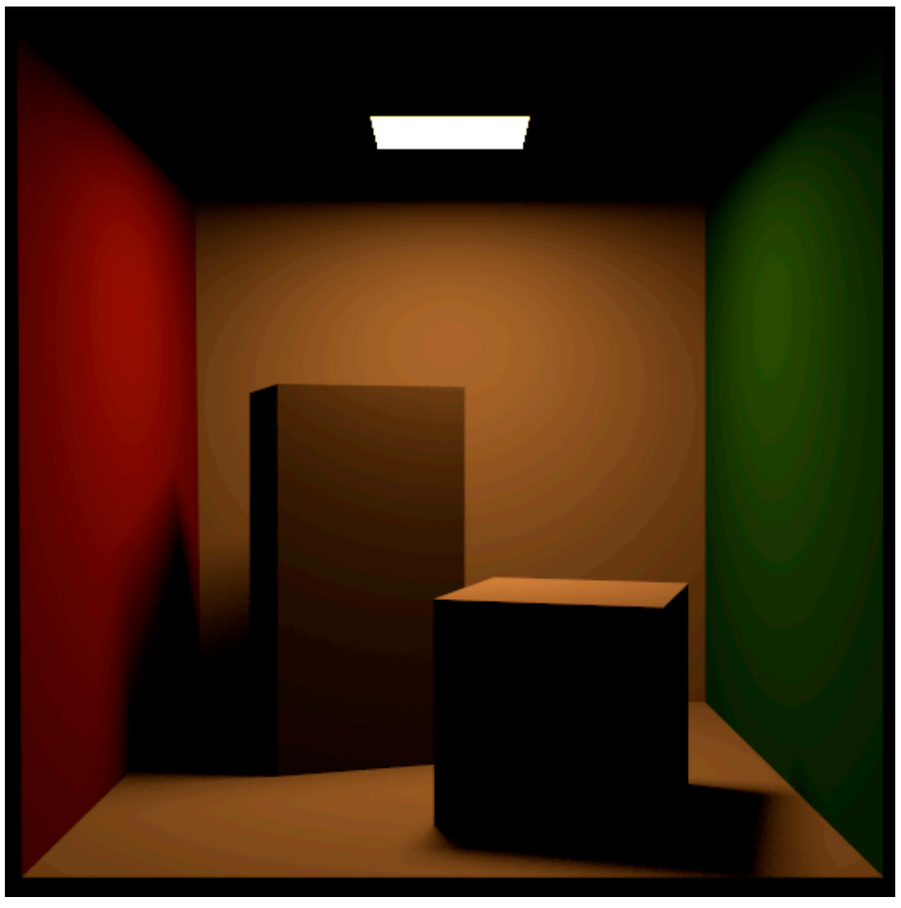
Glossy reflection

The Utah teapot

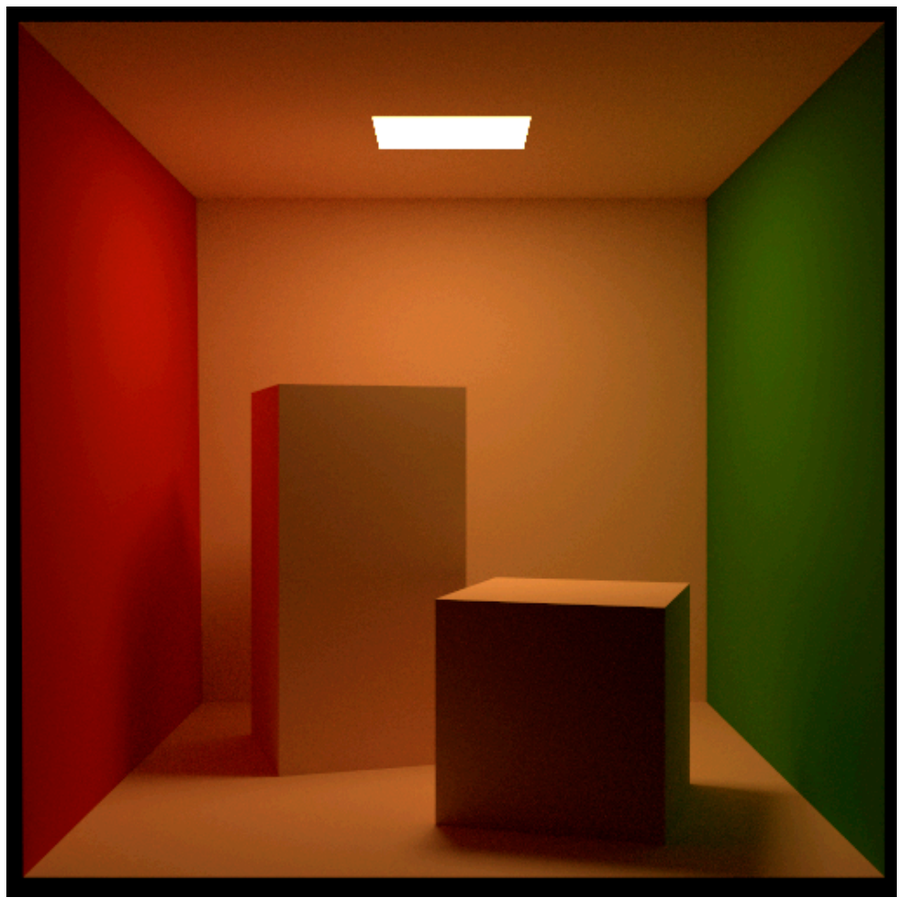
Whitted-Style Ray Tracing: Problem 2

whitted风格光线追踪不考虑漫反射，虽然递归的思想是正确的，但就像下图（康奈尔盒子）所示，whitted的天花板由于接收不到来自环境光照，呈现一个全黑的状态，并且whitted渲染出的长方体并没有表现红墙和绿墙上反射过来的带有色彩的光，相比之下，路径追踪的结果就真实很多

No reflections between diffuse materials?



Path traced:
direct illumination



Path traced:
global illumination

The Cornell box

Whitted-Style Ray Tracing is Wrong

But the rendering equation is correct

$$L_o(p,\omega_o) = L_e(p,\omega_o) + \int_{\Omega^+} L_i(p,\omega_i) f_r(p,\omega_i,\omega_o)(n \cdot \omega_i)\,\mathrm{d}\omega_i$$

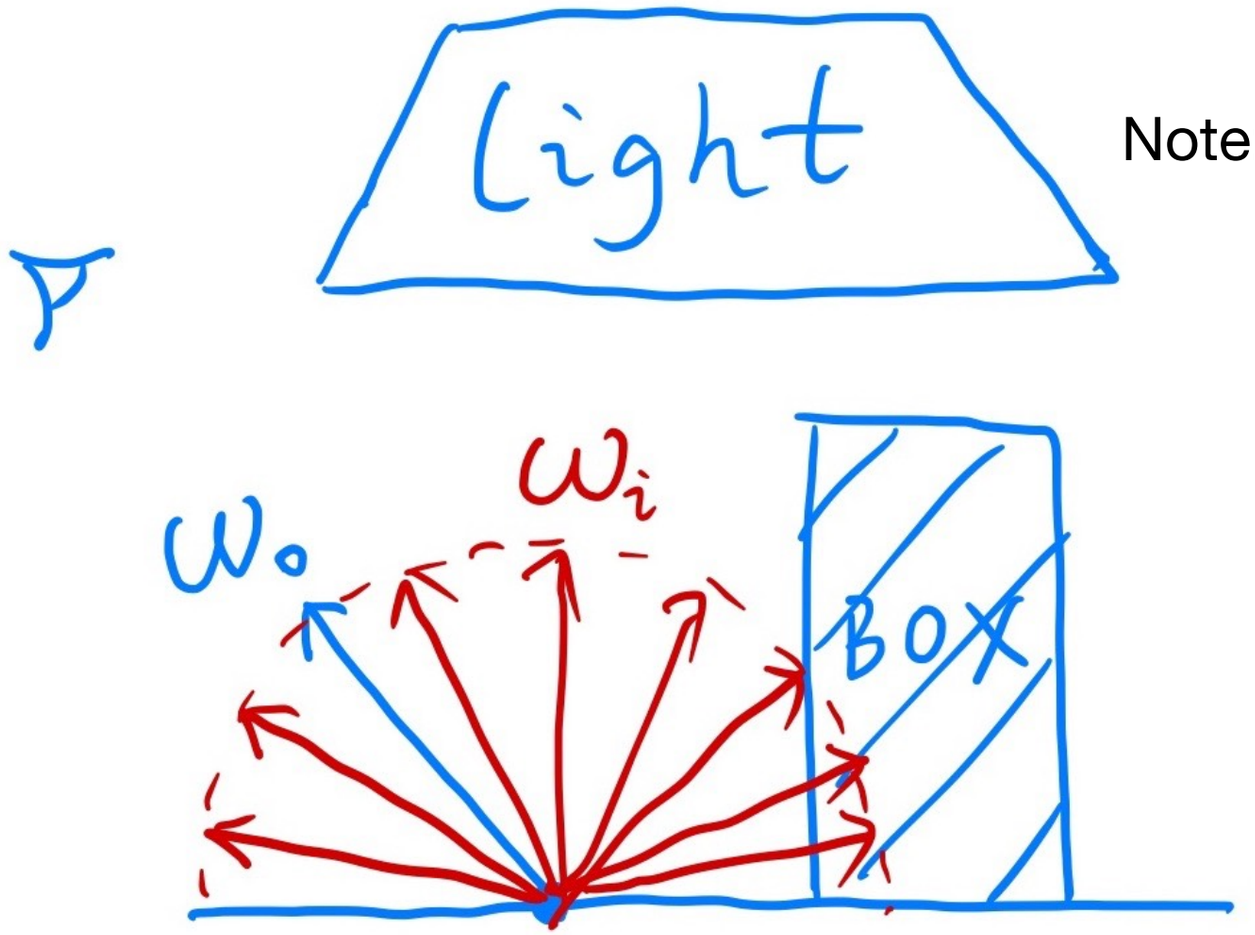
But it involves

- Solving an integral over the hemisphere, and
- Recursive execution

How do you solve an integral numerically?

A Simple Monte Carlo Solution

Suppose we want to render **one pixel (point)** in the following scene for **direct illumination** only



Note: this is an area light

A Simple Monte Carlo Solution

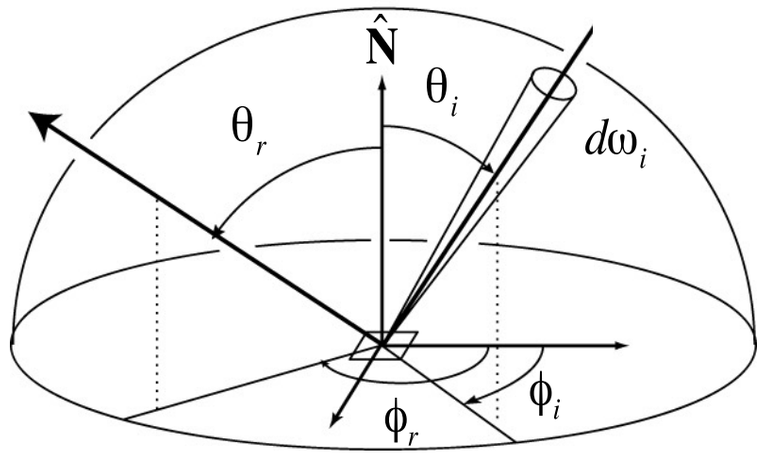
Abuse the concept of Reflection Equation a little bit

$$L_o(p,\omega_o) = \int_{\Omega^+} L_i(p,\omega_i) f_r(p,\omega_i,\omega_o)(n \cdot \omega_i)\,\mathrm{d}\omega_i$$

(again, we assume all directions are pointing outwards)

Fancy as it is, it's still just an integration over directions

So, of course we can solve it using Monte Carlo integration!



A Simple Monte Carlo Solution

对于之前的渲染方程，我们可以用蒙特卡洛积分法进行求解，忽略自发光项（仅计算直接光照），简单考虑均匀采样的情况，半球立体角范围 [0,2π] ，因此 pdf = 1 /2π

We want to compute the radiance at p towards the camera

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

Monte Carlo integration: $\int_a^b f(x)\,\mathrm{d}x \approx \frac{1}{N}\sum_{k=1}^{N}\frac{f(X_k)}{p(X_k)} \qquad X_k \sim p(x)$

What's our "f(x)"? $L_i(p,\omega_i) f_r(p,\omega_i,\omega_o)(n\cdot\omega_i)$

What's our pdf? 单位球的面积是 4pi $p(\omega_i) = 1/2\pi$

(assume uniformly sampling the hemisphere)

A Simple Monte Carlo Solution

So, in general

$$L_o(p,\omega_o) = \int_{\Omega^+} L_i(p,\omega_i) f_r(p,\omega_i,\omega_o)(n\cdot\omega_i)\,\mathrm{d}\omega_i$$

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

(note: abuse notation a little bit for i)

What does it mean?

A correct shading algorithm for direct illumination!

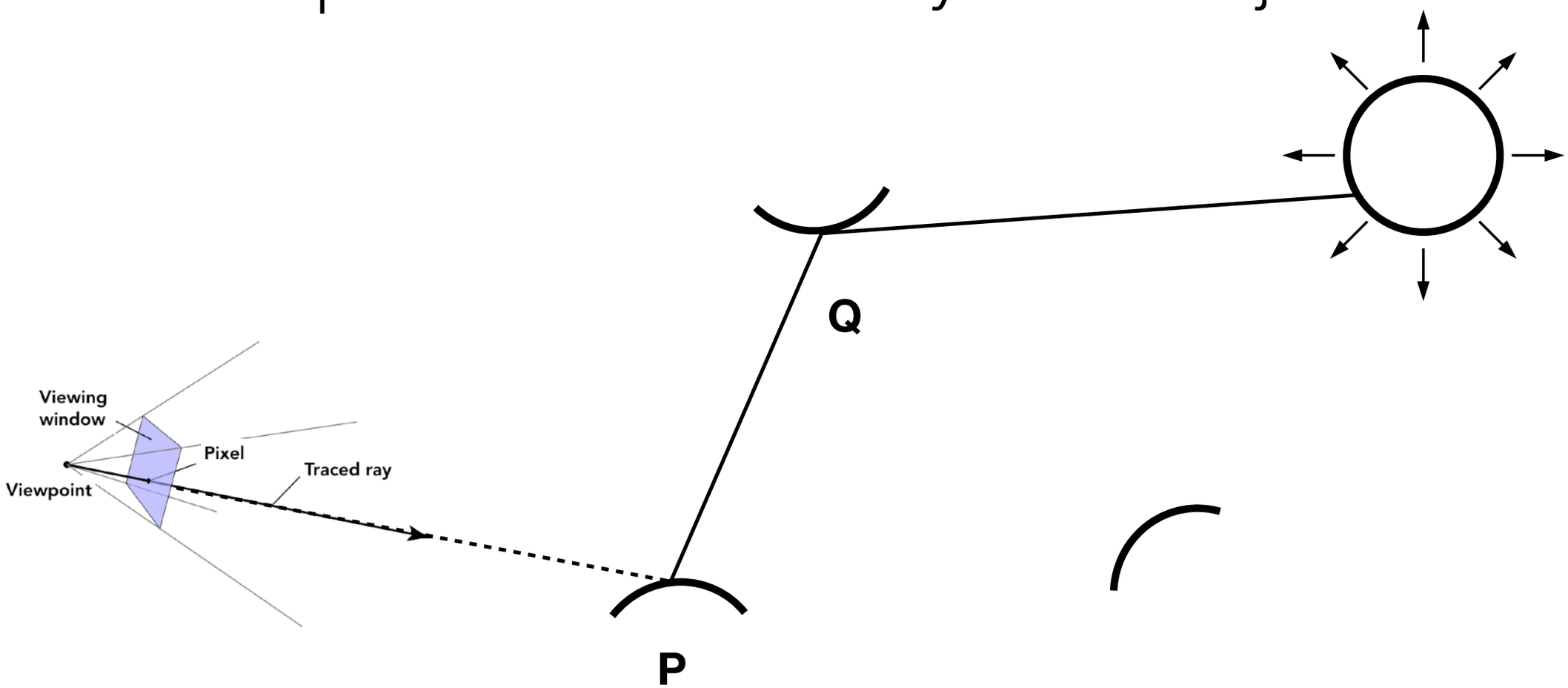
A Simple Monte Carlo Solution

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

```
shade(p, wo)  仅仅考虑 直接光
    Randomly choose N directions wi~pdf  //采样N个方向分别打光
    Lo = 0.0
    For each wi
        Trace a ray r(p, wi)  以 p 点， wi 方向 生成一道光线
        If ray r hit the light  // 若打到光源，则累加
            Lo += (1 / N) * L_i * f_r * cosine / pdf(wi)
    Return Lo
```

Introducing Global Illumination

One more step forward: what if a ray hits an object?



Q also reflects light to P! How much? The dir. illum. at Q!

Introducing Global Illumination

```
shade(p, wo)
    Randomly choose N directions wi~pdf
    Lo = 0.0
    For each wi
        Trace a ray r(p, wi)
        If ray r hit the light
            Lo += (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 Lo
```

引入间接光照，对每个间接光，递归进行处理

递归处理间接光

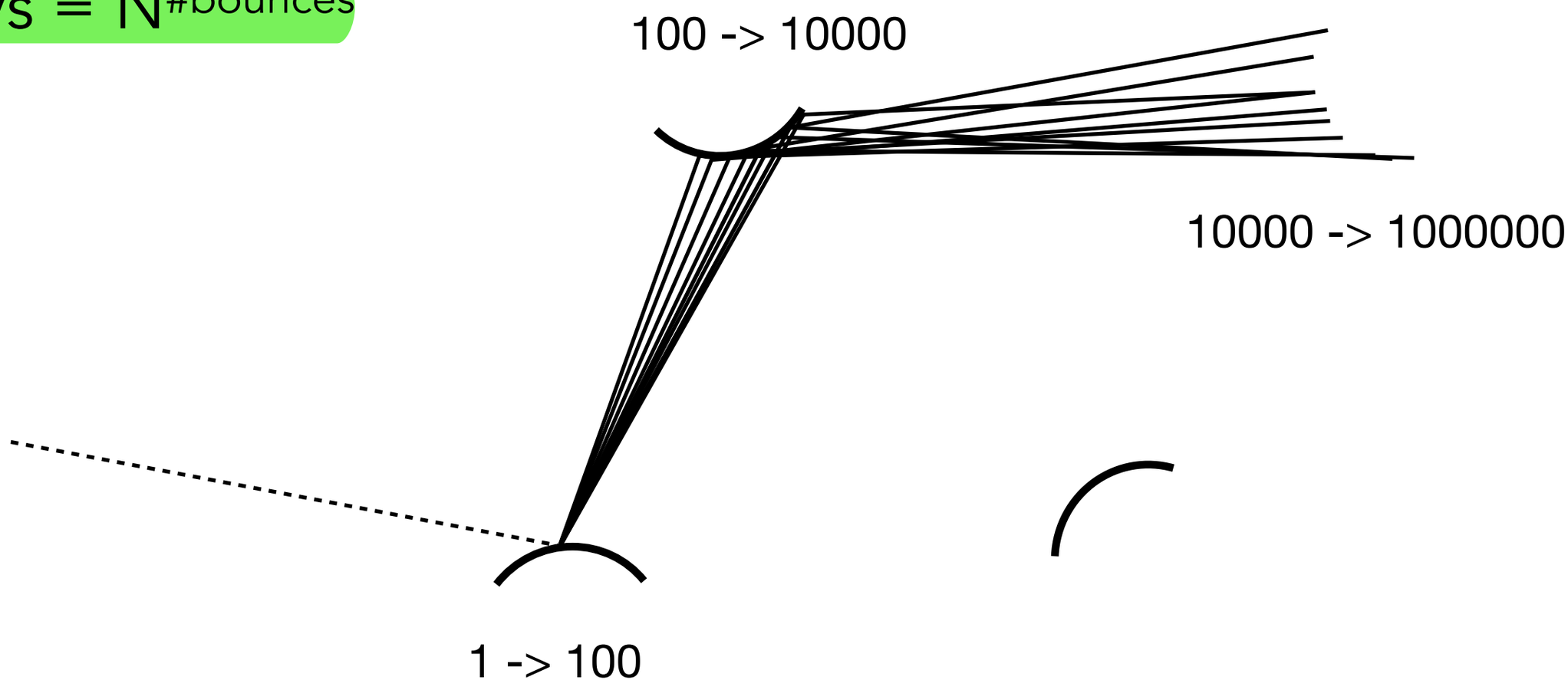
Is it done? **No.**

基本的求解渲染方程已经有了大概的架构，但代码依然存在一些不可忽视的问题

Path Tracing

Problem 1: Explosion of #rays as #bounces go up:

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



p的着色需要q的环境光照信息，而q的环境光照信息又必须包含其他物体的环境光照信息，如此需要追踪的光线数量会呈指数级增长

Key observation: #rays will not explode iff N = ???????

Path Tracing

From now on, we always assume that
only 1 ray is traced at each shading point:

只有当采样数为1的时候，即N=1时，才不会受到这种影响，路径追踪也因此得名（N ≠ 1时称为分布式光线追踪）

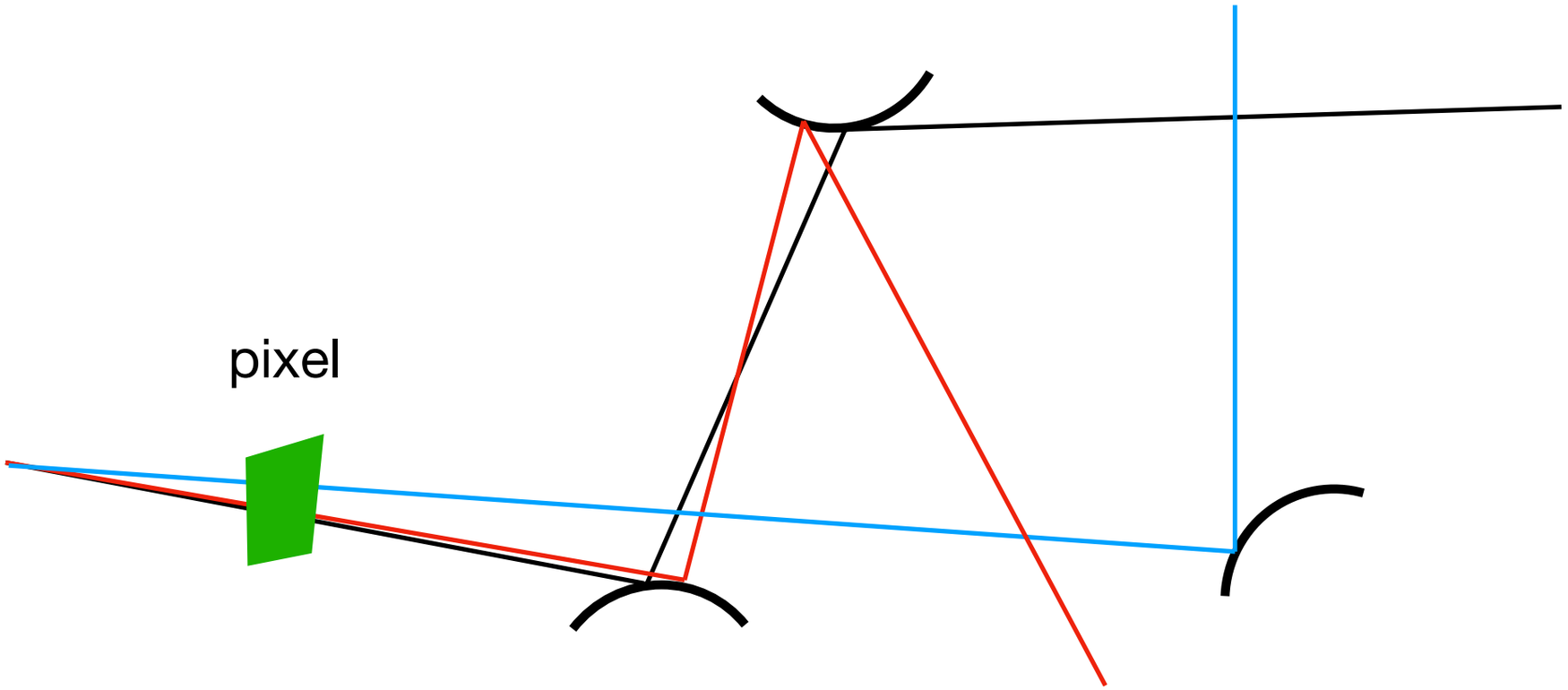
```
shade(p, wo)
    Randomly choose ONE direction wi~pdf(w)
    Trace a ray r(p, wi)
    If ray r hit the light
        Return L_i * f_r * cosine / pdf(wi)
    Else If ray r hit an object at q
        Return shade(q, -wi) * f_r * cosine / pdf(wi)
```

This is **path tracing**! (FYI, Distributed Ray Tracing if N != **1)**

Ray Generation

But this will be noisy!

No problem, just trace more **paths** through each pixel and average their radiance!



这样一来误差就会特别大，渲染结果势必会有非常多的噪点，为了解决这个问题，我们对单个像素计算多次路径追踪结果，随后求平均，这个过程其实也用到了蒙特卡洛方法

Ray Generation

Very similar to ray casting in ray tracing

```
ray_generation(camPos, pixel)                    //对 场景采样 N 个样本点
    Uniformly choose N sample positions within the pixel
    pixel_radiance = 0.0
    For each sample in the pixel                 //对每条路径计算着色后进行累加，注意该处pdf和shade函数中的pdf
                                                 并不相同，这里是对像素采样的pdf
        Shoot a ray r(camPos, cam_to_sample)     从视点到样本点生成一道光线
        If ray r hit the scene at p
            pixel_radiance += 1 / N * shade(p, sample_to_cam)
    Return pixel_radiance
```

Path Tracing

Now are we good? Any other problems in shade()?

```
shade(p, wo)
    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)
    Else If ray r hit an object at q
        Return shade(q, -wi) * f_r * cosine / pdf(wi)
```

很容易看出，上述算法的递归没有终止条件，放到现实中这也是非常合理的，因为现实中的光并不会弹射一定次数后终止弹射，而会一直弹射下去，在算法中强行设置终止次数结束递归不满足现实情况的能量守恒定律，会有一定亮度差异（弹射3次和弹射17次的亮度差异是非常明显的），为了解决这一问题，我们需要用到与俄罗斯轮盘赌类似的思想

Problem 2: The recursive algorithm will never stop!

Path Tracing

Dilemma: the light does not stop bouncing indeed!

Cutting #bounces == cutting energy!

3 bounces

Path Tracing

Dilemma: the light does not stop bouncing indeed!

Cutting #bounces == cutting energy!

17 bounces

Solution: Russian Roulette (RR)

（俄罗斯轮盘赌）

Russian Roulette is all about probability

With probability 0 < P < 1, you are fine

With probability 1 - P, otherwise

Example: two bullets,
Survival probability P = 4 / 6

Solution: Russian Roulette (RR)

这个思想其实就是一个伯努利分布（均匀分布）概念，让光线在每个弹射点都有一定概率继续弹射，设这个概率为p，那么光线停止弹射的概率为（1–p）， 为了得到相同的路径追踪结果，意味着这个伯努利实验的期望值必须保持Lo不变，那么我们可以巧妙的认为p概率继续弹射得到的能量是 Lo / p（终止弹射的结果很自然就是0）

Previously, we always shoot a ray at a shading point and get the shading result **Lo**

Suppose we manually set a probability P (0 < P < 1)

With probability P, shoot a ray and return the **shading result divided by P**: **Lo / P**

With probability 1-P, don't shoot a ray and you'll get **0**

In this way, you can still **expect** to get Lo!:

E = P * **(Lo / P)** + (1 - P) * **0** = **Lo**

Solution: Russian Roulette (RR)

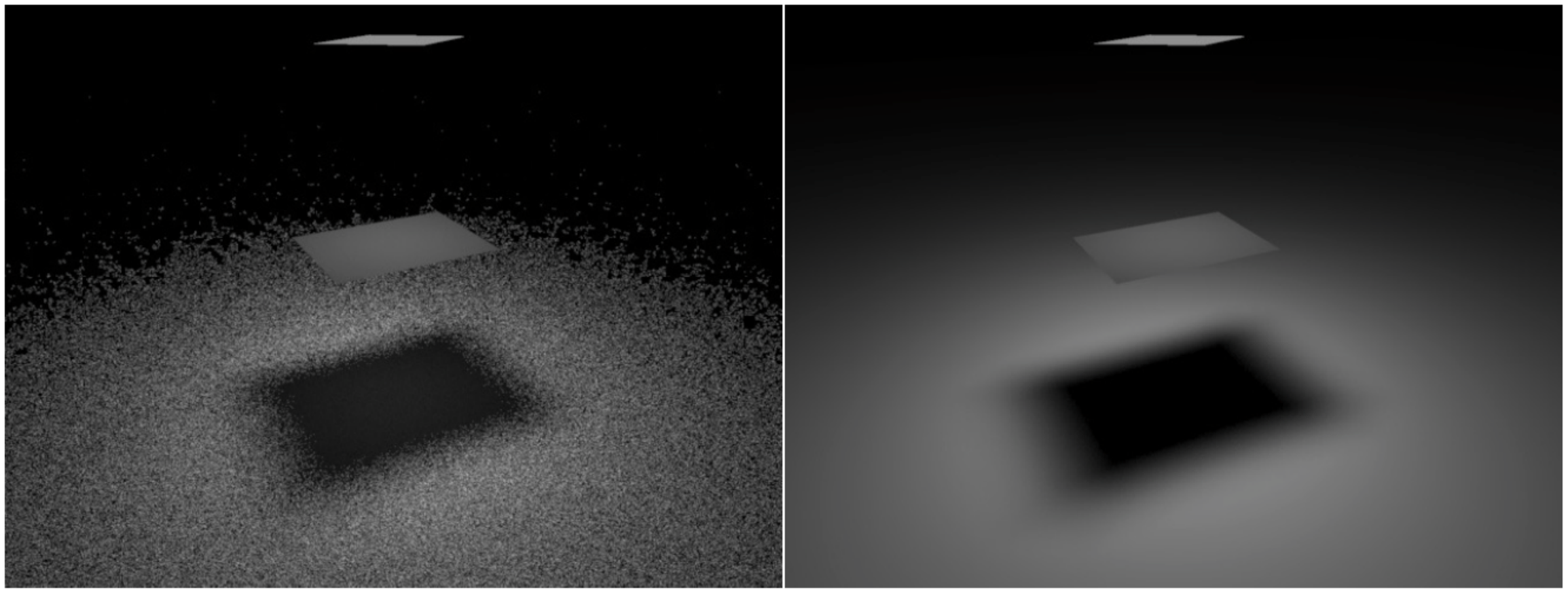
```
shade(p, wo)
    Manually specify a probability P_RR
    Randomly select ksi in a uniform dist. in [0, 1]
    If (ksi > P_RR) return 0.0;

    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
```

Path Tracing

Now we already have a **correct** version of path tracing!

But it's **not really efficient**.



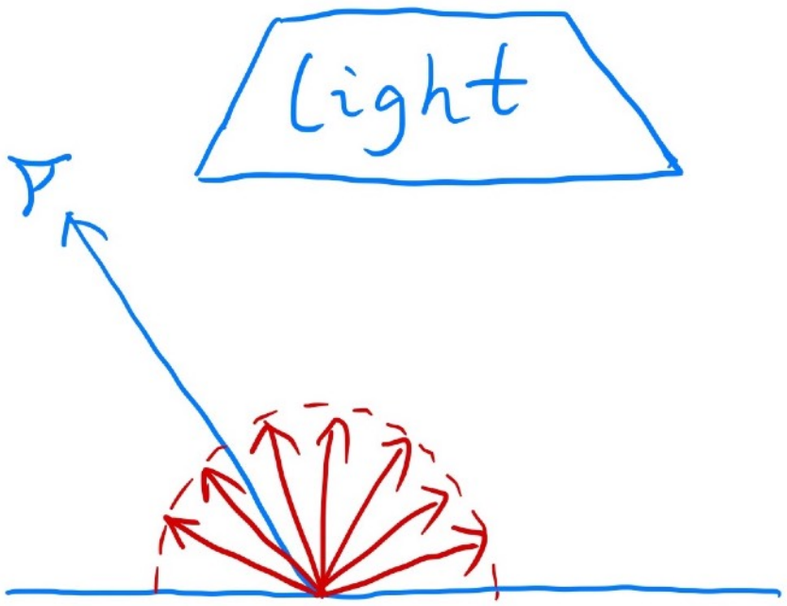
Low SPP (samples per pixel)
noisy results

High SPP

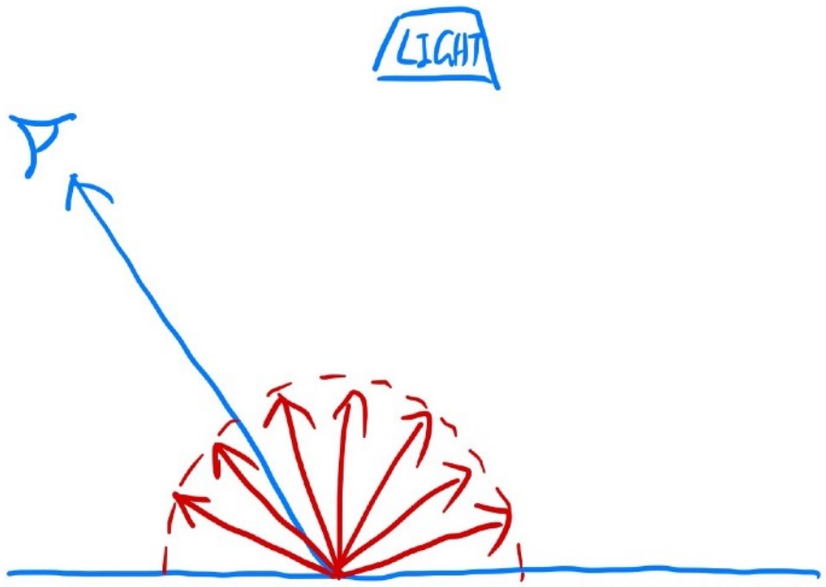
Sampling the Light

如果光源面积很小，则我们的均匀采样就需要很大的N才能保证采样到光源方向。也就是这里问题来自两个第一个是均匀抽样方法浪费了很多光源不存在的角度。

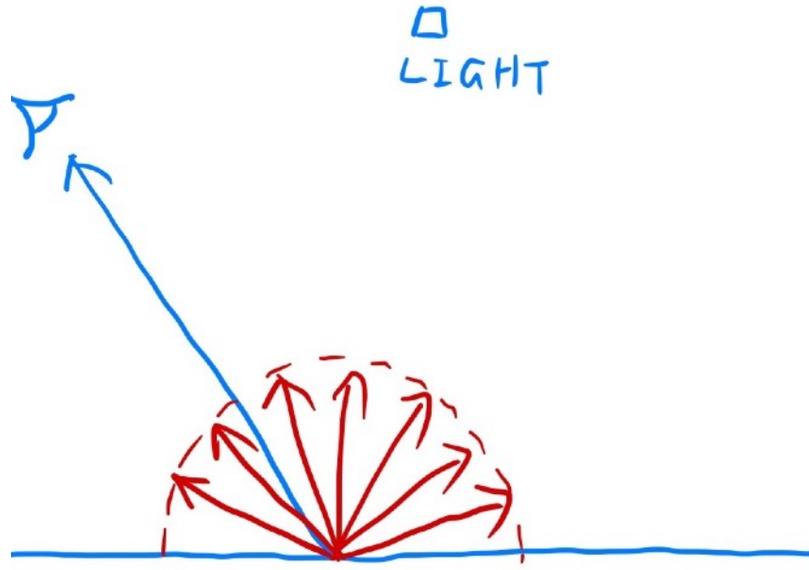
Understanding the reason of being inefficient



Every 5 rays,



500 rays,



50000 rays,

there will be 1 ray hitting the light. So **a lot of rays are "wasted"** if we uniformly sample the hemisphere at the shading point.

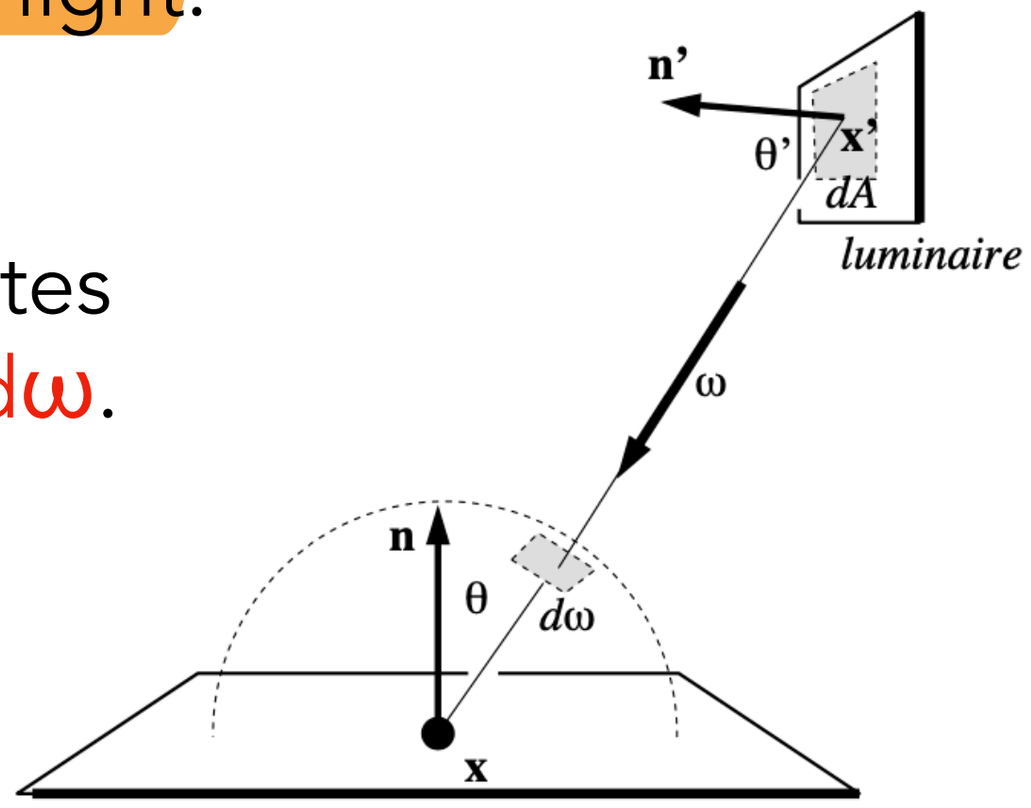
Sampling the Light (pure math)

Monte Carlo methods allows any sampling methods, so we can sample the light (therefore no rays are "wasted")

Assume uniformly sampling on the light:

pdf = 1 / A (because $\int \text{pdf}\, dA = 1$)

But the rendering equation integrates on the solid angle: $L_o = \int L_i f_r \cos d\omega$.



Recall Monte Carlo Integration:
Sample on x & integrate on x

Since we sample on the light, can we integrate on the light?

Sampling the Light

Need to make the rendering equation as an integral of dA

Need the relationship between dω and dA

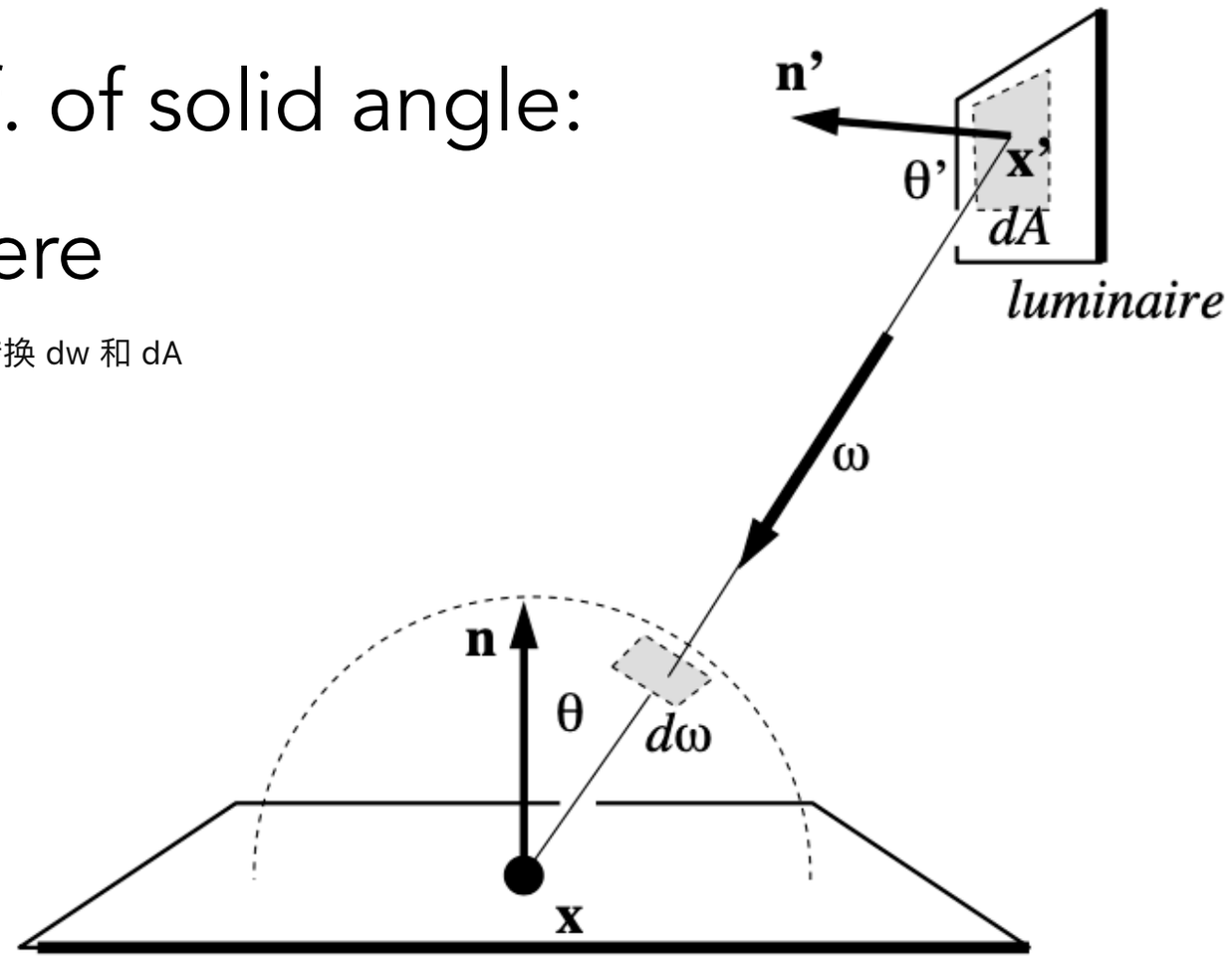
Easy! Recall the alternative def. of solid angle:

Projected area on the unit sphere

这里还是用均匀采样，此时 p 就从 1 / 2pi 变为了 1 / A ，这里关键就在于如何转换 dw 和 dA

$$d\omega = \frac{dA\ \cos\theta'}{\|x' - x\|^2}$$

(Note: θ’, not θ)



Sampling the Light

Then we can rewrite the rendering equation as

$$L_o(x,\omega_o) = \int_{\Omega^+} L_i(x,\omega_i) f_r(x,\omega_i,\omega_o) \cos\theta \,\mathrm{d}\omega_i$$

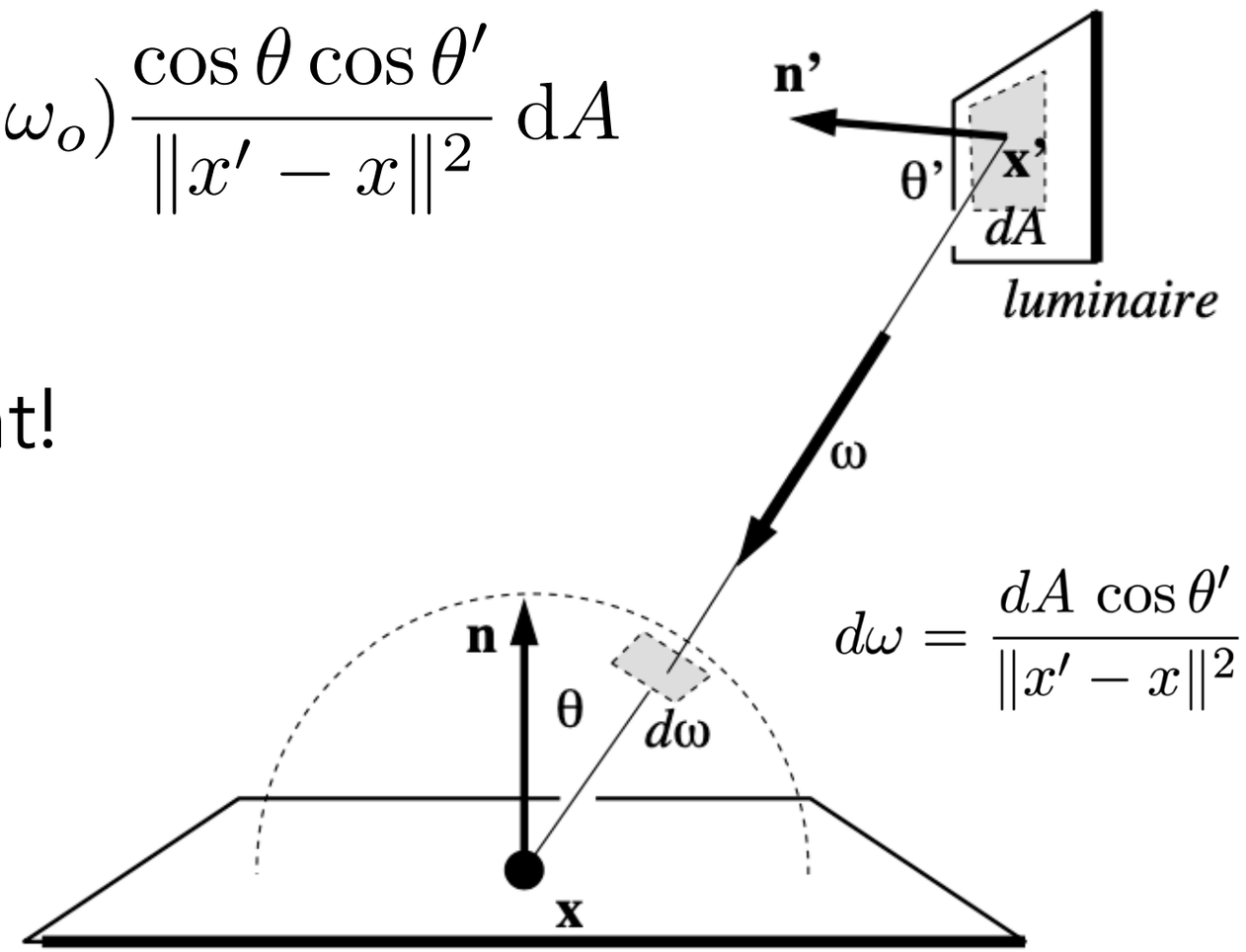
$$= \int_A L_i(x,\omega_i) f_r(x,\omega_i,\omega_o) \frac{\cos\theta \cos\theta'}{\|x'-x\|^2} \,\mathrm{d}A$$

Now an integration on the light!

Monte Carlo integration:

"f(x)": everything inside

Pdf: 1 / A

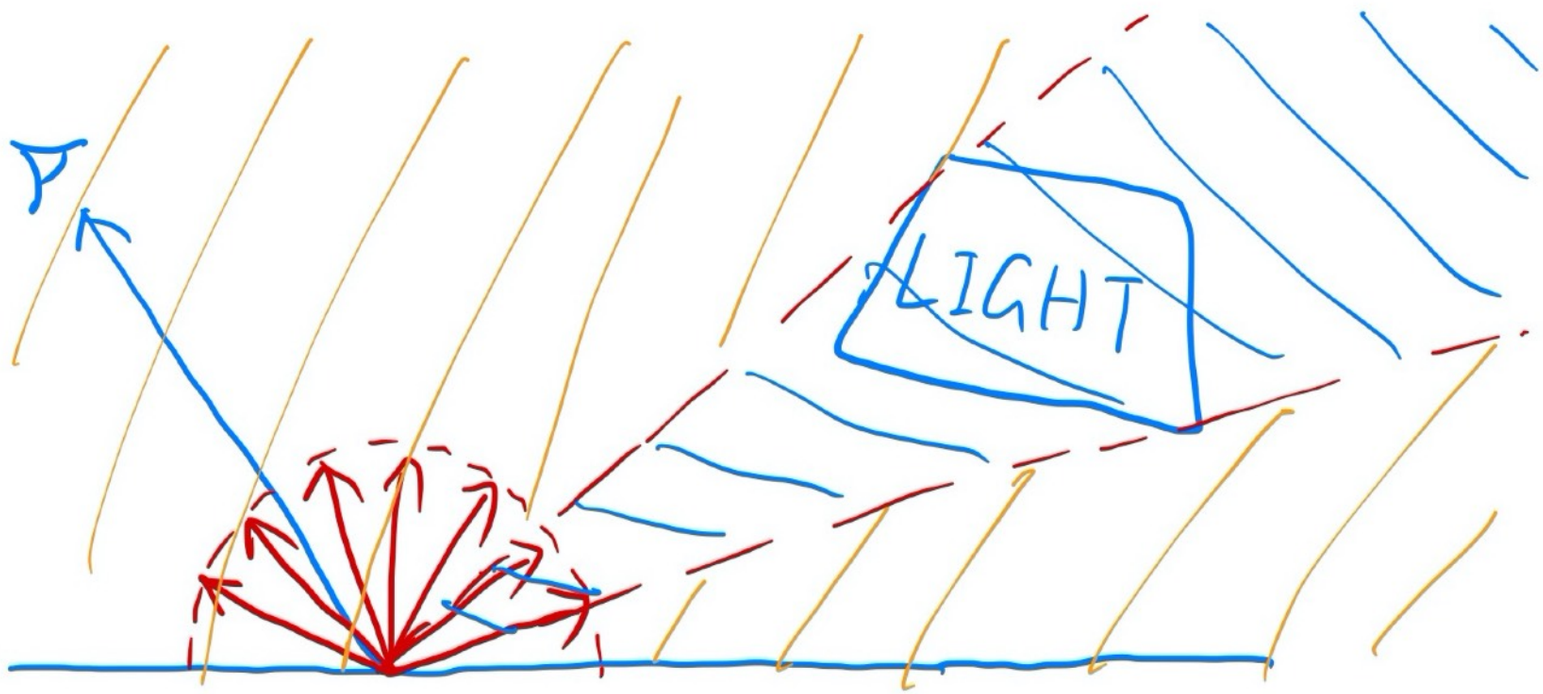


Sampling the Light

Previously, we assume the light is “accidentally” shot by uniform hemisphere sampling

Now we consider the radiance coming from two parts:

1. light source (direct, no need to have RR)
2. other reflectors (indirect, RR)



Sampling the Light

shade(p, wo)　分为两部分计算，一部分是光源直接对着色点的影响，另一部分是光源的间接影响

```
# Contribution from the light source.
Uniformly sample the light at x' (pdf_light = 1 / A)
L_dir = L_i * f_r * cos θ * cos θ' / |x' - p|^2 / pdf_light

# Contribution from other reflectors.
L_indir = 0.0
Test Russian Roulette with probability P_RR
Uniformly sample the hemisphere toward wi (pdf_hemi = 1 / 2pi)
Trace a ray r(p, wi)
If ray r hit a non-emitting object at q 光线击中了不发光物体，相交于 q
    L_indir = shade(q, -wi) * f_r * cos θ / pdf_hemi / P_RR

Return L_dir + L_indir
```

Sampling the Light

One final thing: how do we know if the sample on the light is not blocked or not?

```
# 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!

Some Side Notes

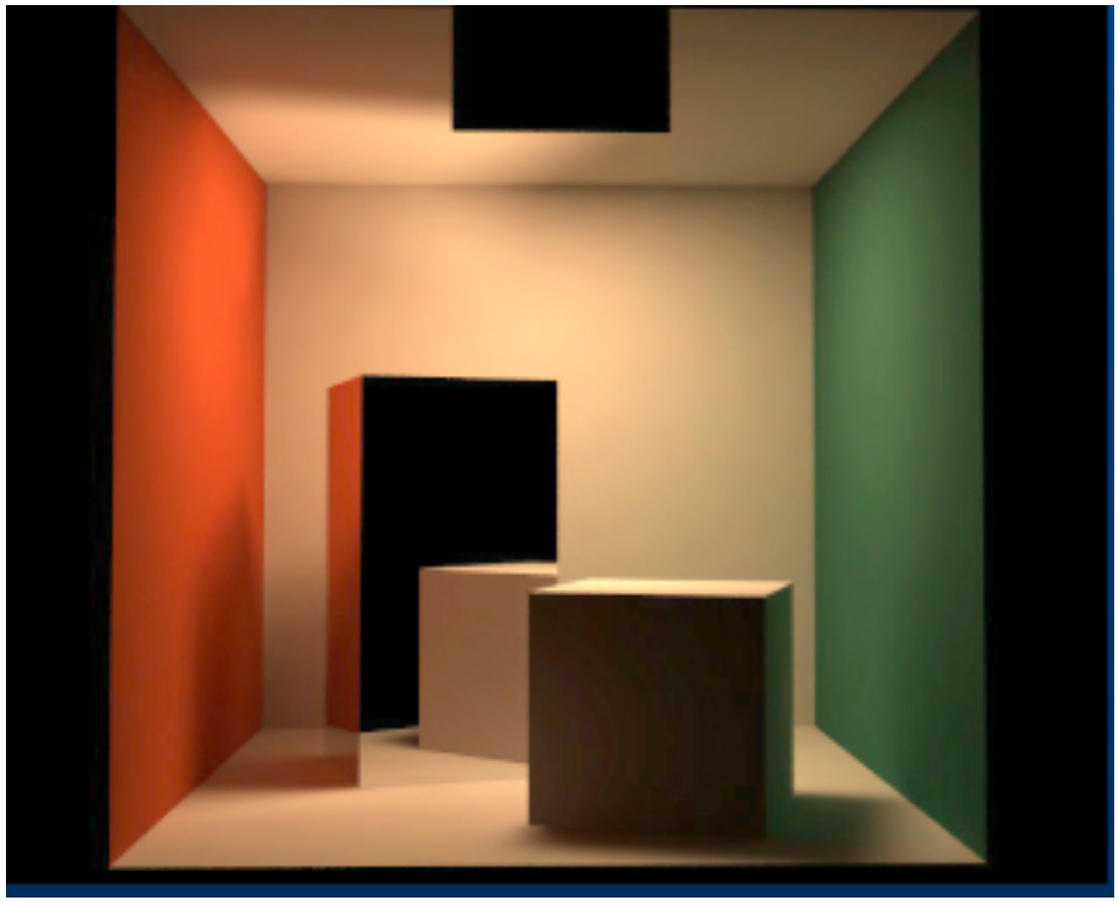
- Path tracing (PT) is indeed difficult
  - Consider it the most challenging in undergrad CS
  - Why: physics, probability, calculus, coding
  - Learning PT will help you understand deeper in these

- Is it still “Introductory”?
  - Not really, but it’s “modern" :)
  - And so learning it will be rewarding also because …

Is Path Tracing Correct?

Yes, almost 100% correct, a.k.a. PHOTO-REALISTIC

Photo



Path traced:
global illumination

The Cornell box — http://www.graphics.cornell.edu/online/box/compare.html

Ray tracing: Previous vs. Modern Concepts

- Previous

  - Ray tracing == Whitted-style ray tracing

- Modern (my own definition)

  - **The general solution of light transport**, including

  - (Unidirectional & bidirectional) path tracing

  - Photon mapping

  - Metropolis light transport

  - VCM / UPBP…

Things we haven’t covered / won’t cover

- Uniformly sampling the hemisphere
  - How? And in general, how to sample any function? (sampling)
- Monte Carlo integration allows arbitrary pdfs
  - What's the best choice? (importance sampling)
- Do random numbers matter?
  - Yes! (low discrepancy sequences)

Things we haven’t covered / won’t cover

- I can sample the hemisphere and the light
  - Can I combine them? Yes! (multiple imp. sampling)
- The radiance of a pixel is the average of radiance on all paths passing through it
  - Why? (pixel reconstruction filter)
- Is the radiance of a pixel the color of a pixel?
  - No. (gamma correction, curves, color space)
- Asking again, is path tracing still “Introductory”?
  - This time, yes. Fear the science, my friends.

Thank you!

(And thank Prof. Ravi Ramamoorthi and Prof. Ren Ng for many of the slides!)