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!

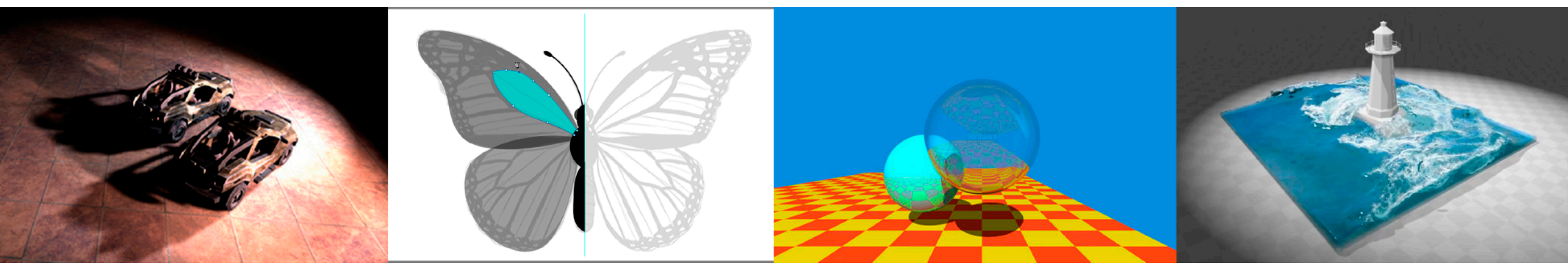
Introduction to Computer Graphics

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

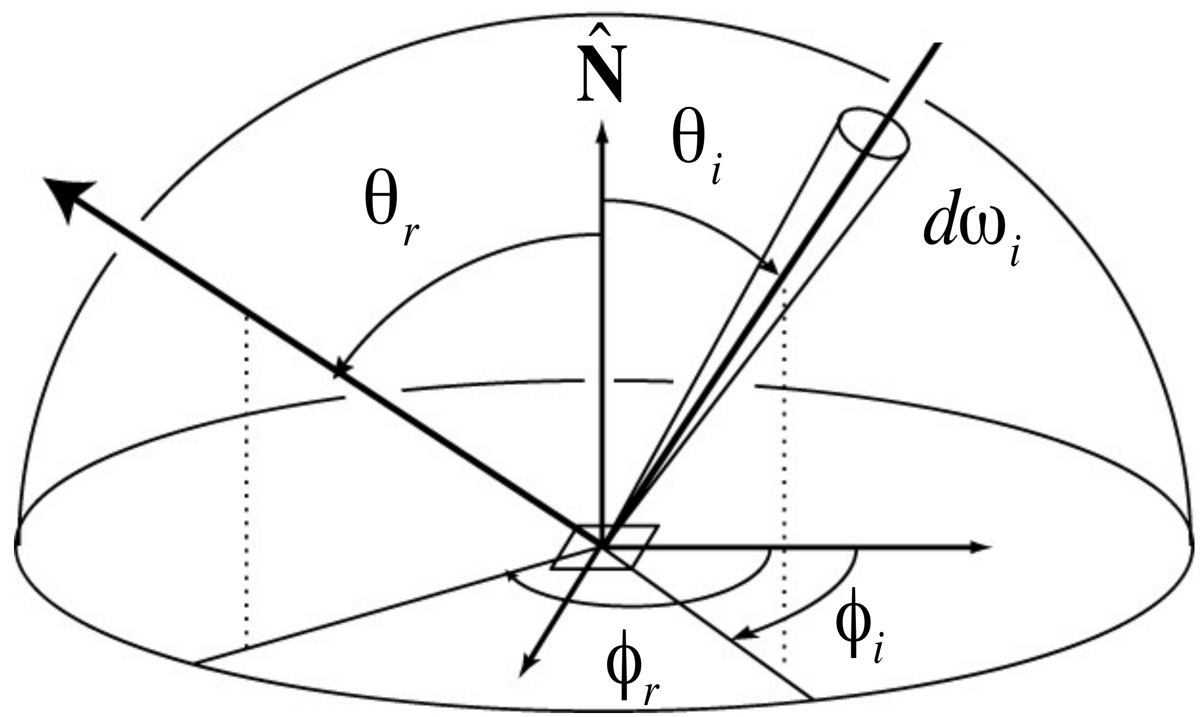
Today

- A Brief Review
- Monte Carlo Integration
- Path Tracing

Review - The Rendering Equation

- Describing the light transport

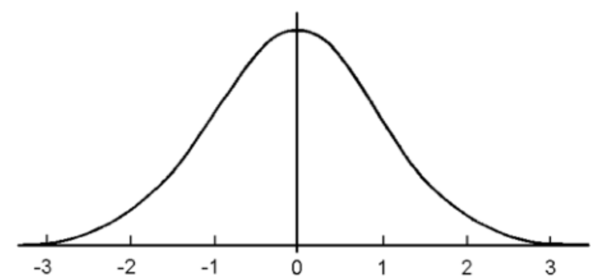
$$L_o(p, \omega_o) = L_e(p, \omega_o) + \int_{\Omega^+} L_i(p, \omega_i) f_r(p, \omega_i, \omega_o)(n \cdot \omega_i)\, \mathrm{d}\omega_i$$



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)

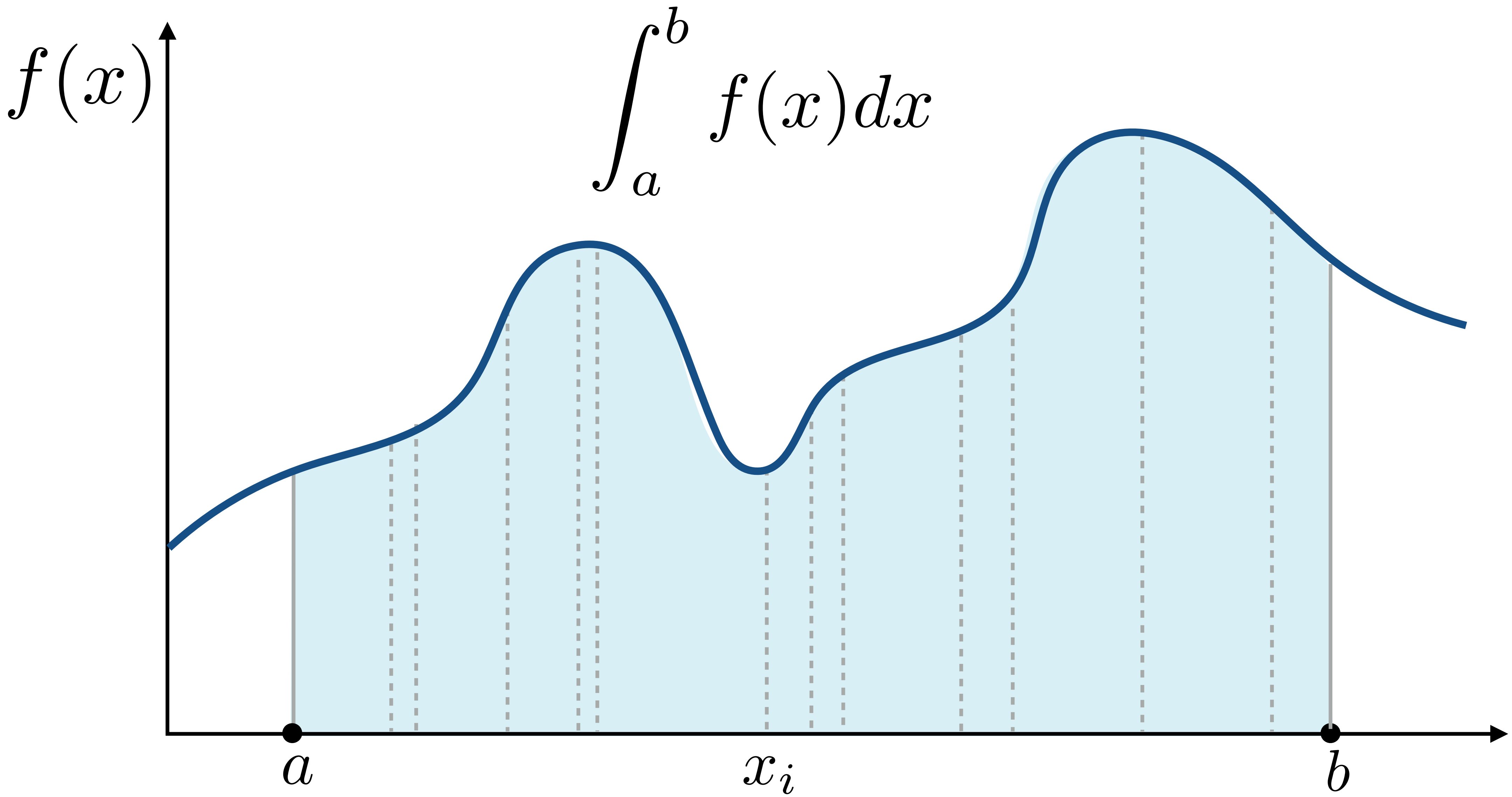
Conditions on p(x): $p(x) \geq 0 \text{ and } \int p(x)\, dx = 1$

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

Monte Carlo Integration

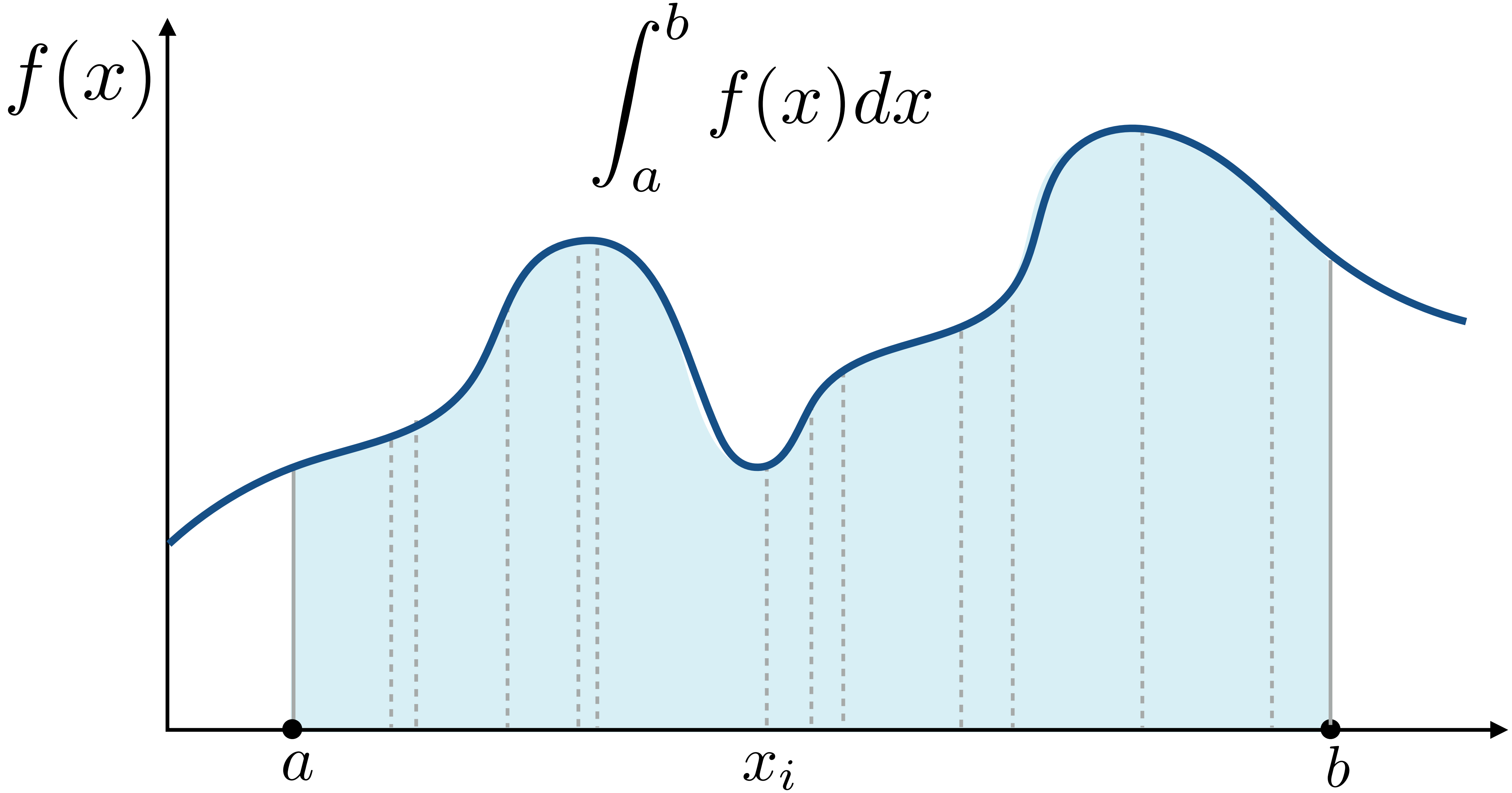
Monte Carlo Integration

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

Example: Uniform Monte Carlo Estimator

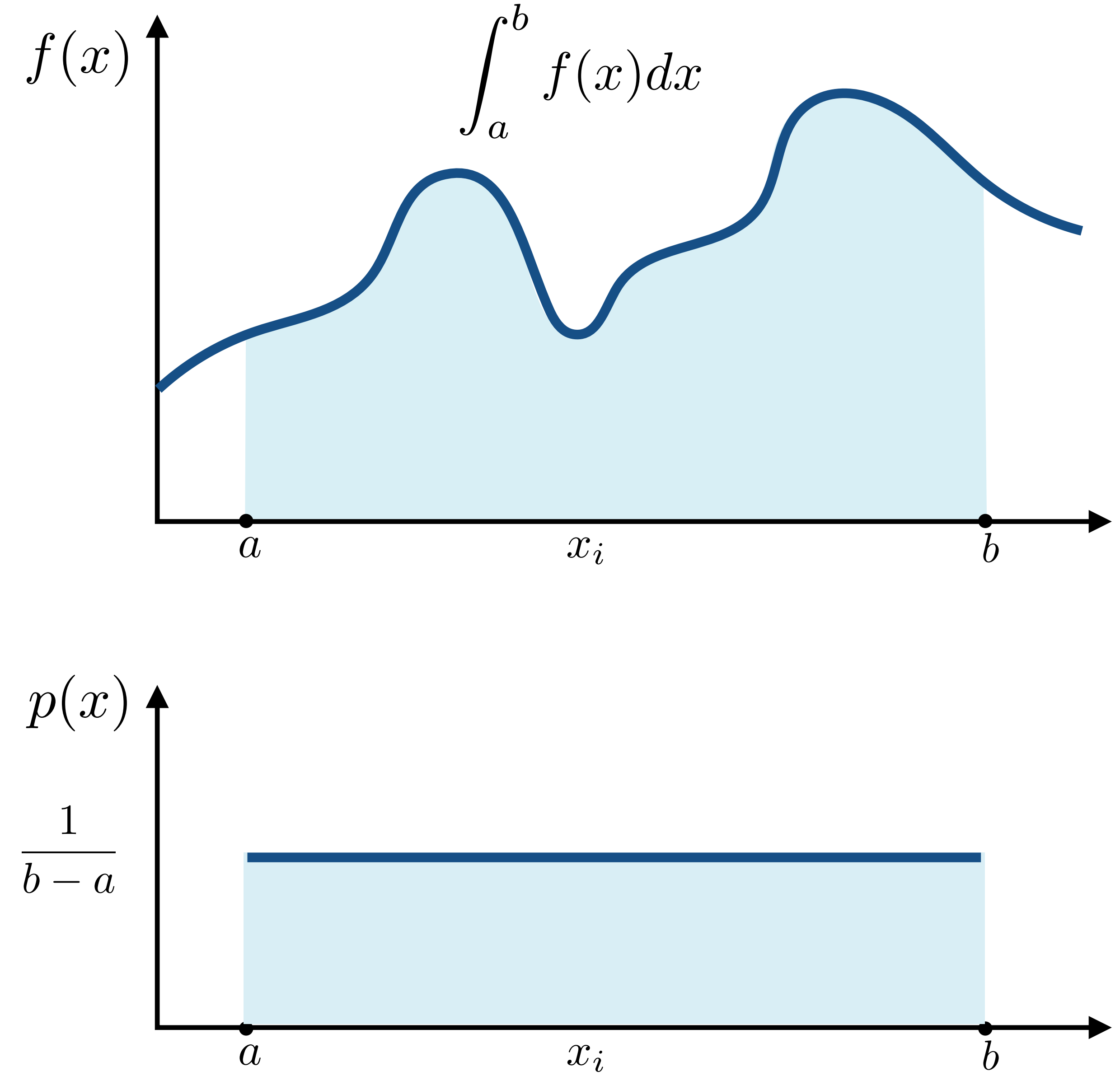
Uniform random variable

$$X_i \sim p(x) = C$$ **(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	$F_N = \frac{b-a}{N}\sum_{i=1}^{N} f(X_i)$

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

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?

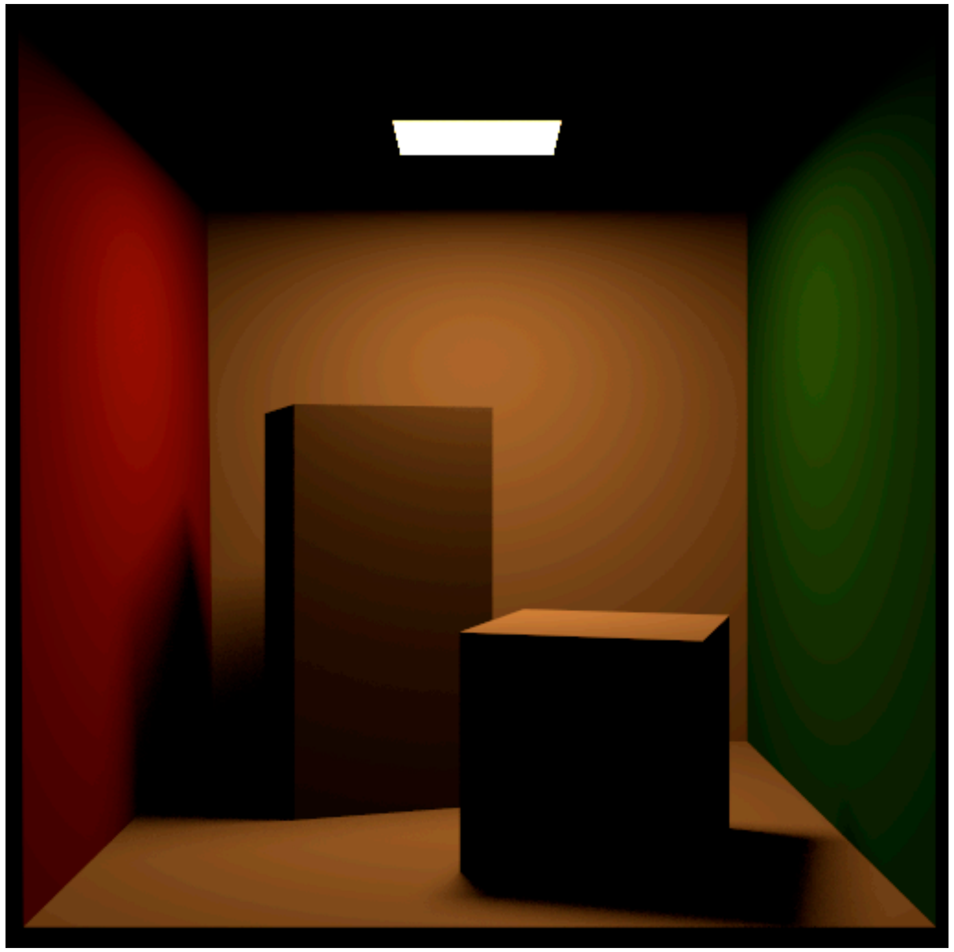
Mirror reflection

Glossy reflection

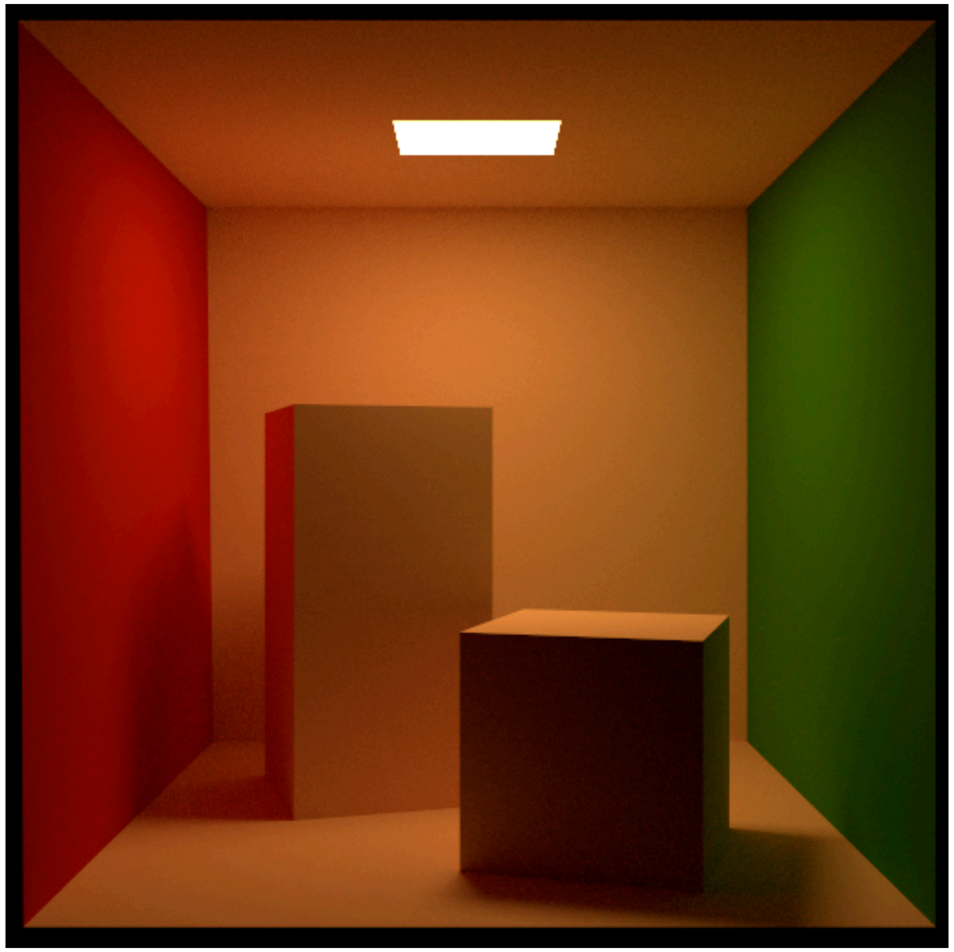
The Utah teapot

Whitted-Style Ray Tracing: Problem 2

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

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

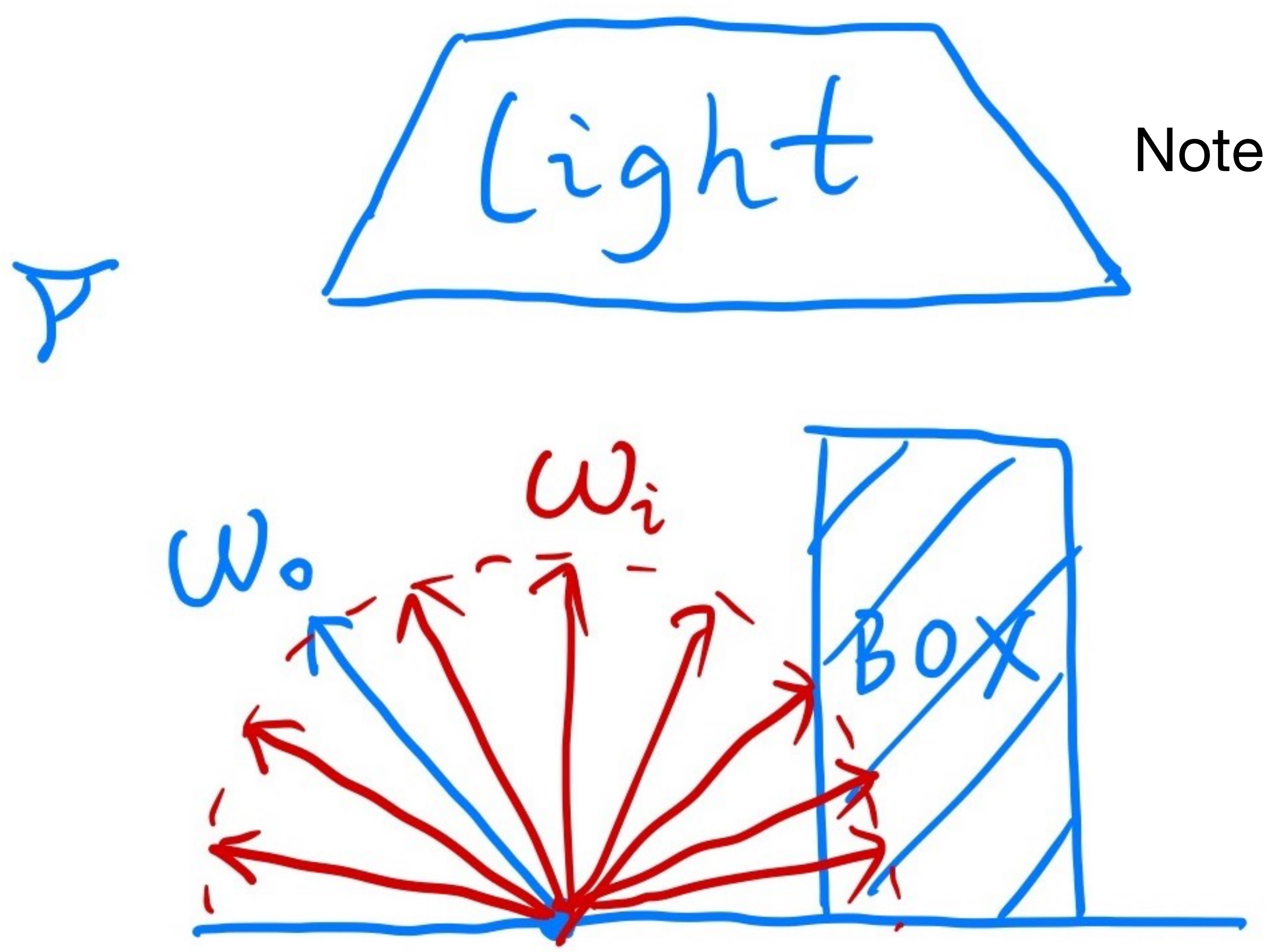
But it involves

- 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

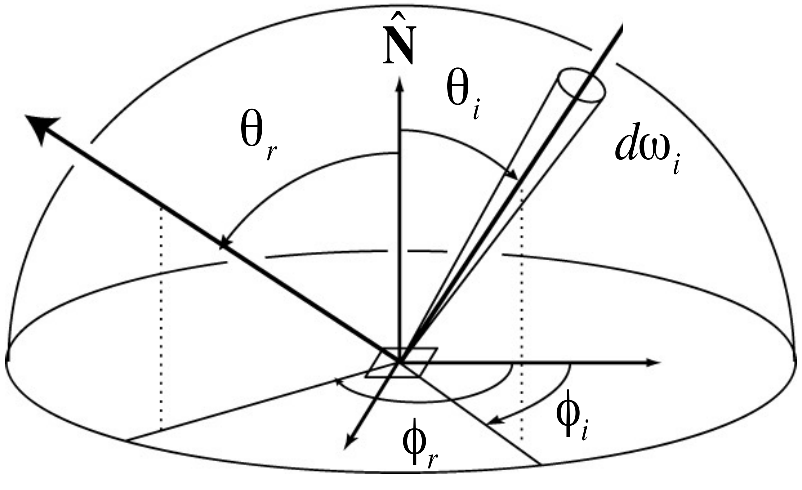
Abuse the concept of Reflection Equation a little bit

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

(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

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? $p(\omega_i) = 1/2\pi$

(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!

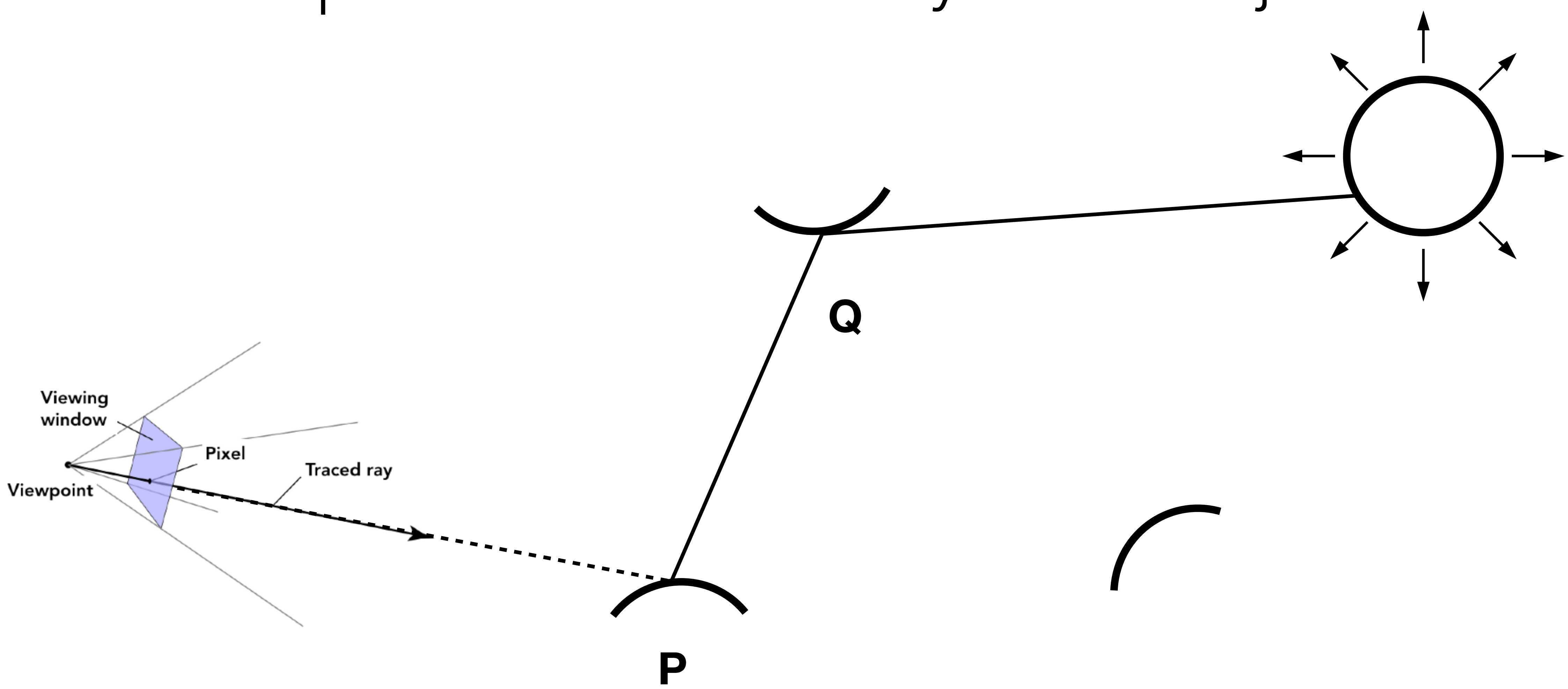
A Simple Monte Carlo Solution

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

```
shade(p, wo)
    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)
    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

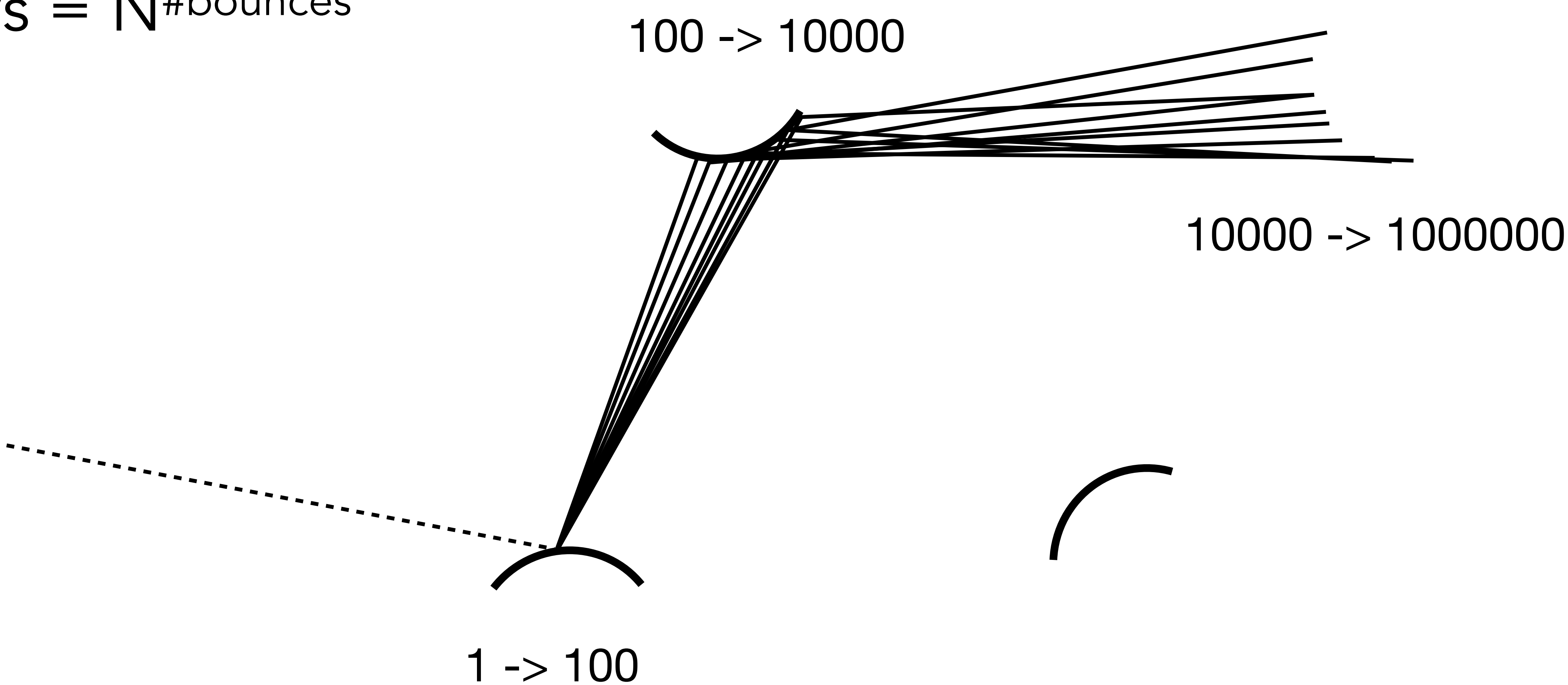
```
shade(p, wo)
    Randomly choose N directions wi~pdf
    Lo = 0.0
    For each wi
        Trace a ray r(p, wi)
        If ray r hit the light
            Lo += (1 / N) * L_i * f_r * cosine / pdf(wi)
        Else If ray r hit an object at q
            Lo += (1 / N) * shade(q, -wi) * f_r * cosine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

Is it done? **No.**

Path Tracing

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

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



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:

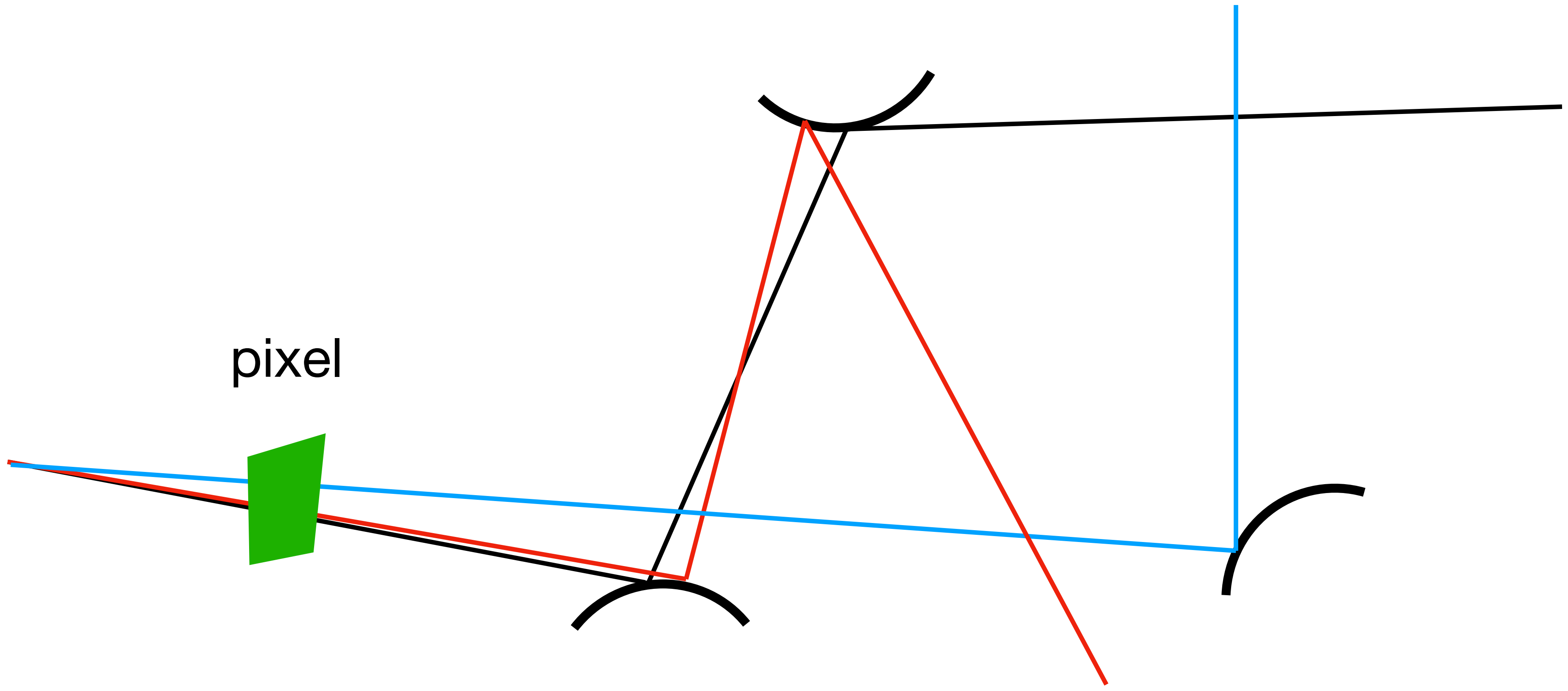
```
shade(p, wo)
    Randomly choose ONE direction wi~pdf(w)
    Trace a ray r(p, wi)
    If ray r hit the light
        Return L_i * f_r * cosine / pdf(wi)
    Else If ray r hit an object at q
        Return shade(q, -wi) * f_r * cosine / pdf(wi)
```

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!



Ray Generation

Very similar to ray casting in ray tracing

```
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
```

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)
```

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)

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**

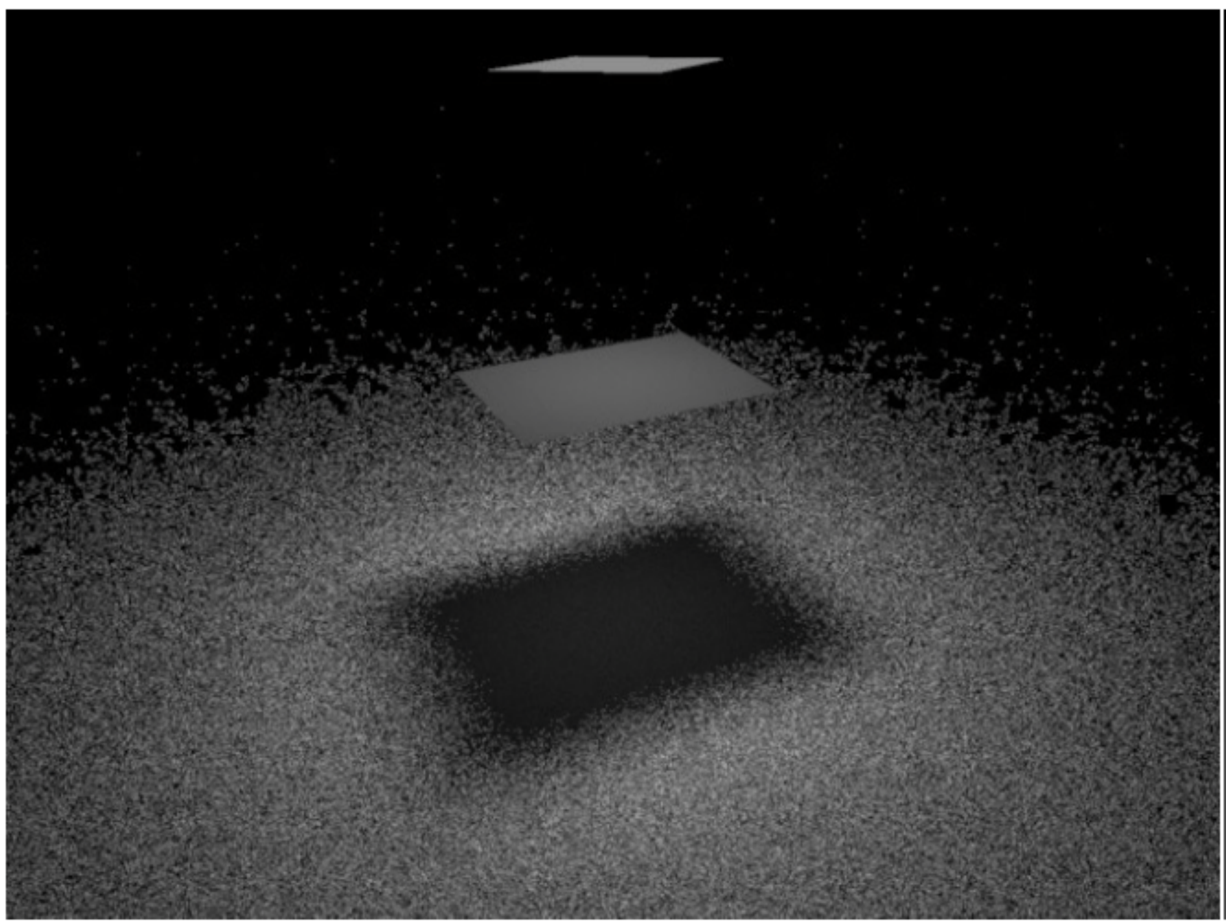
Solution: Russian Roulette (RR)

```
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
```

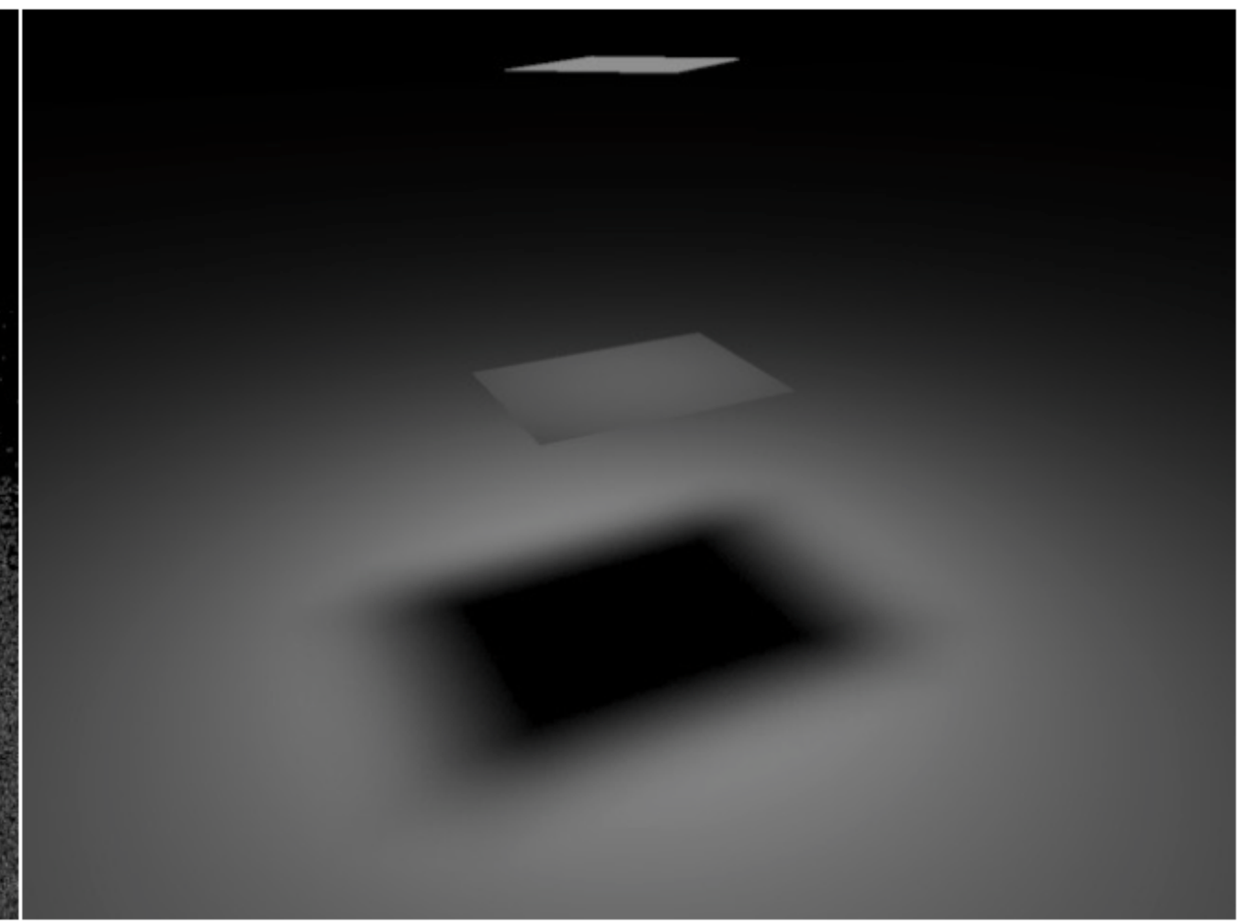
Path Tracing

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

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



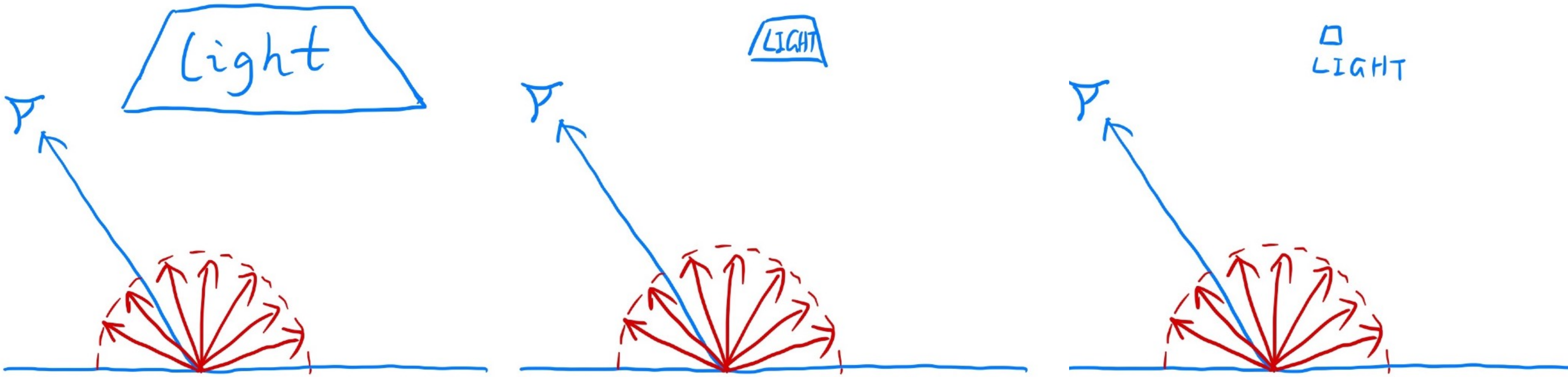
Low SPP (samples per pixel)
noisy results



High SPP

Sampling the Light

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.

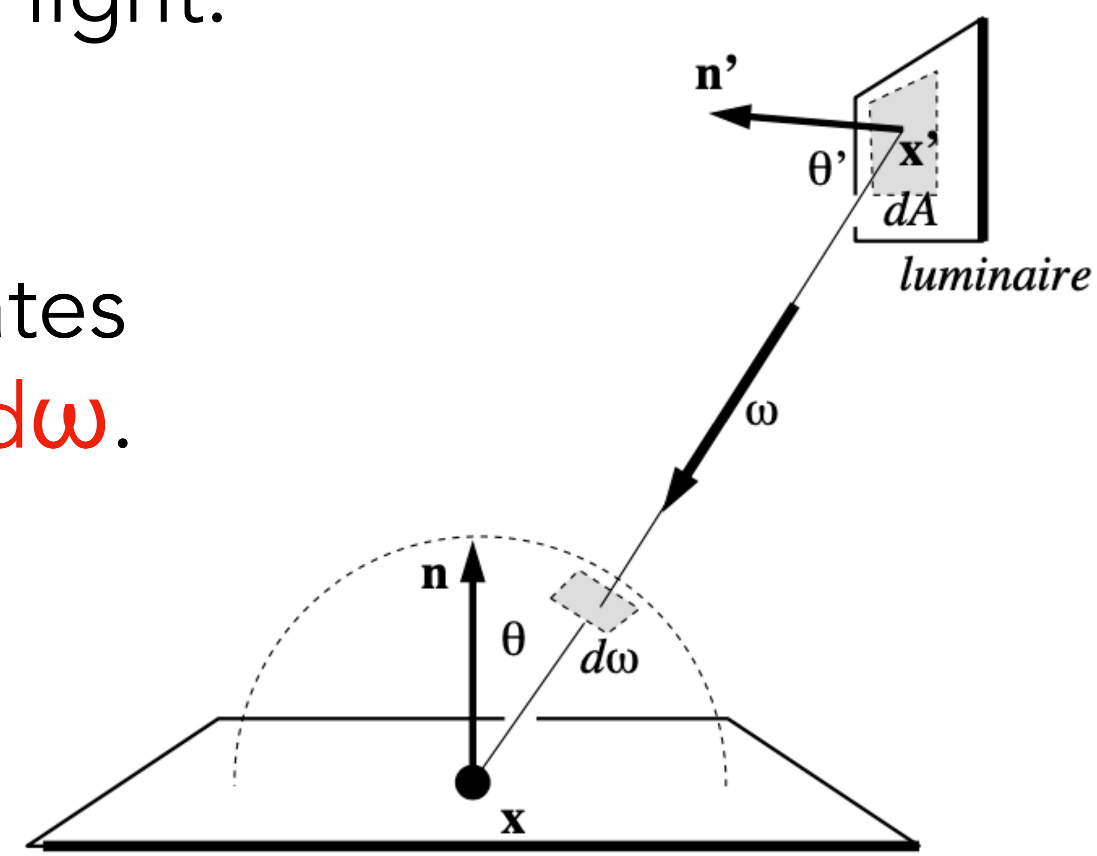
Sampling the Light (pure math)

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 ∫pdf dA = 1)

But the rendering equation integrates on the solid angle: Lo = ∫Li fr cos dω.



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

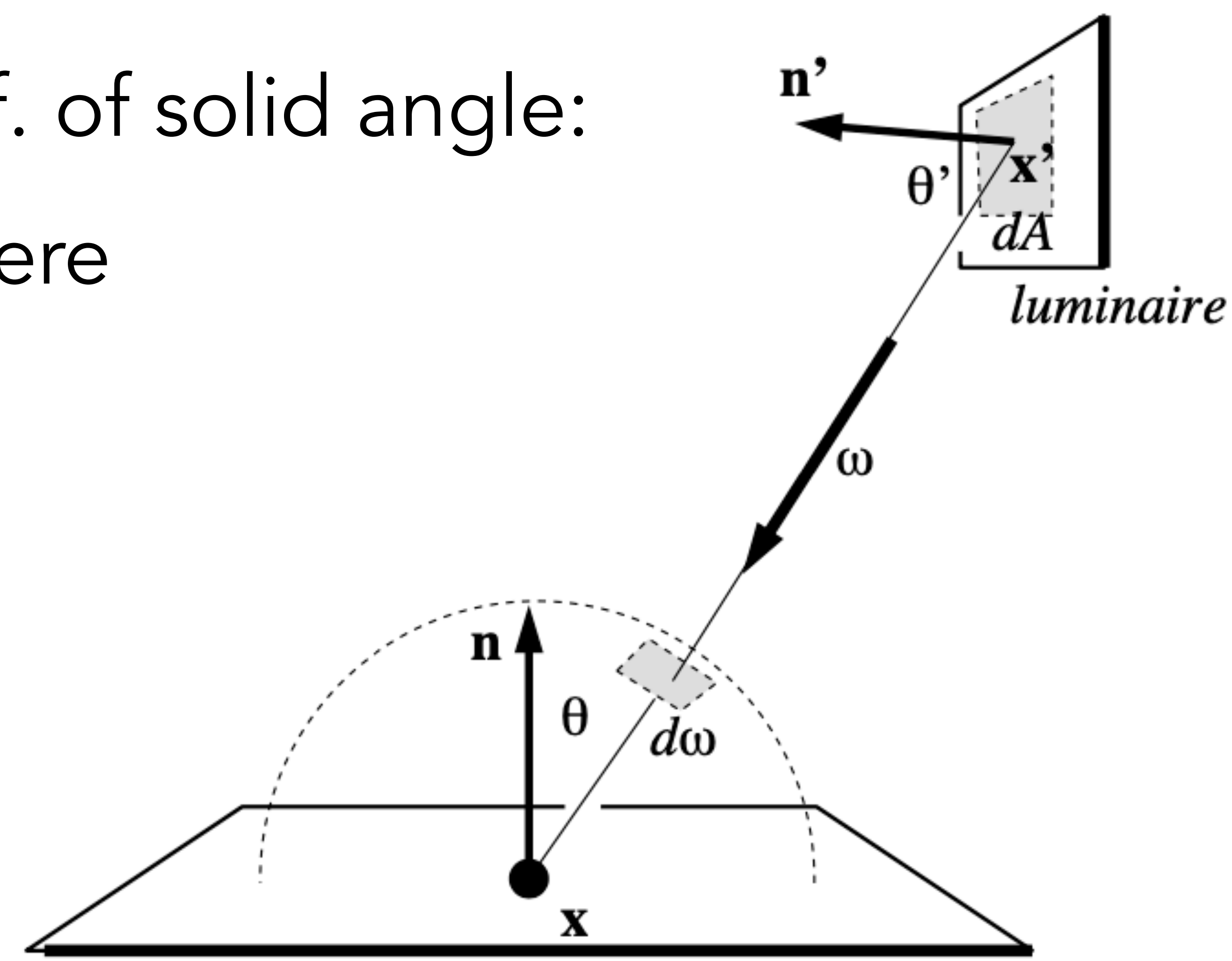
Need the relationship between dω and dA

Easy! Recall the alternative def. of solid angle:

Projected area on the unit sphere

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

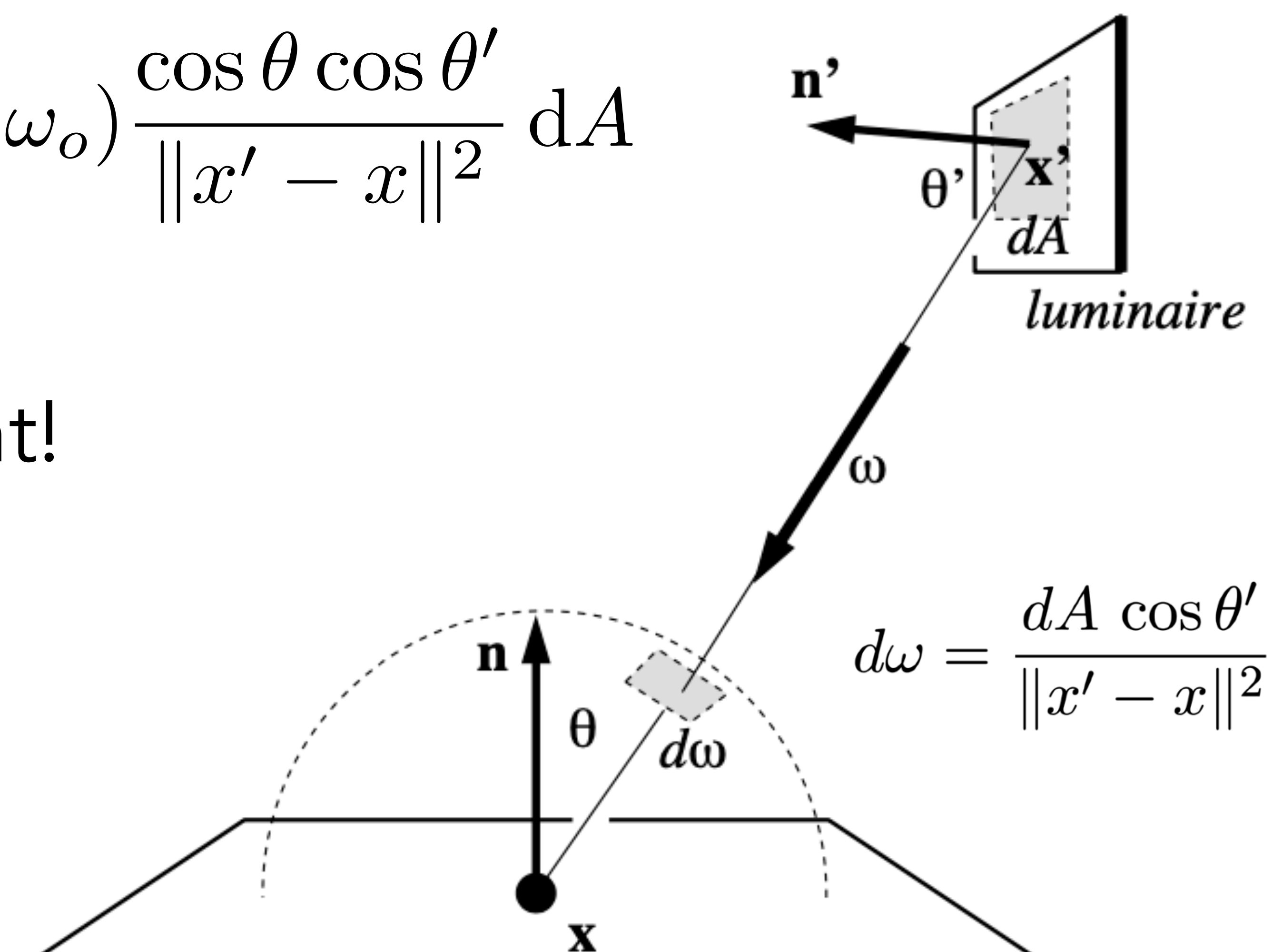
(Note: θ', not θ)



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

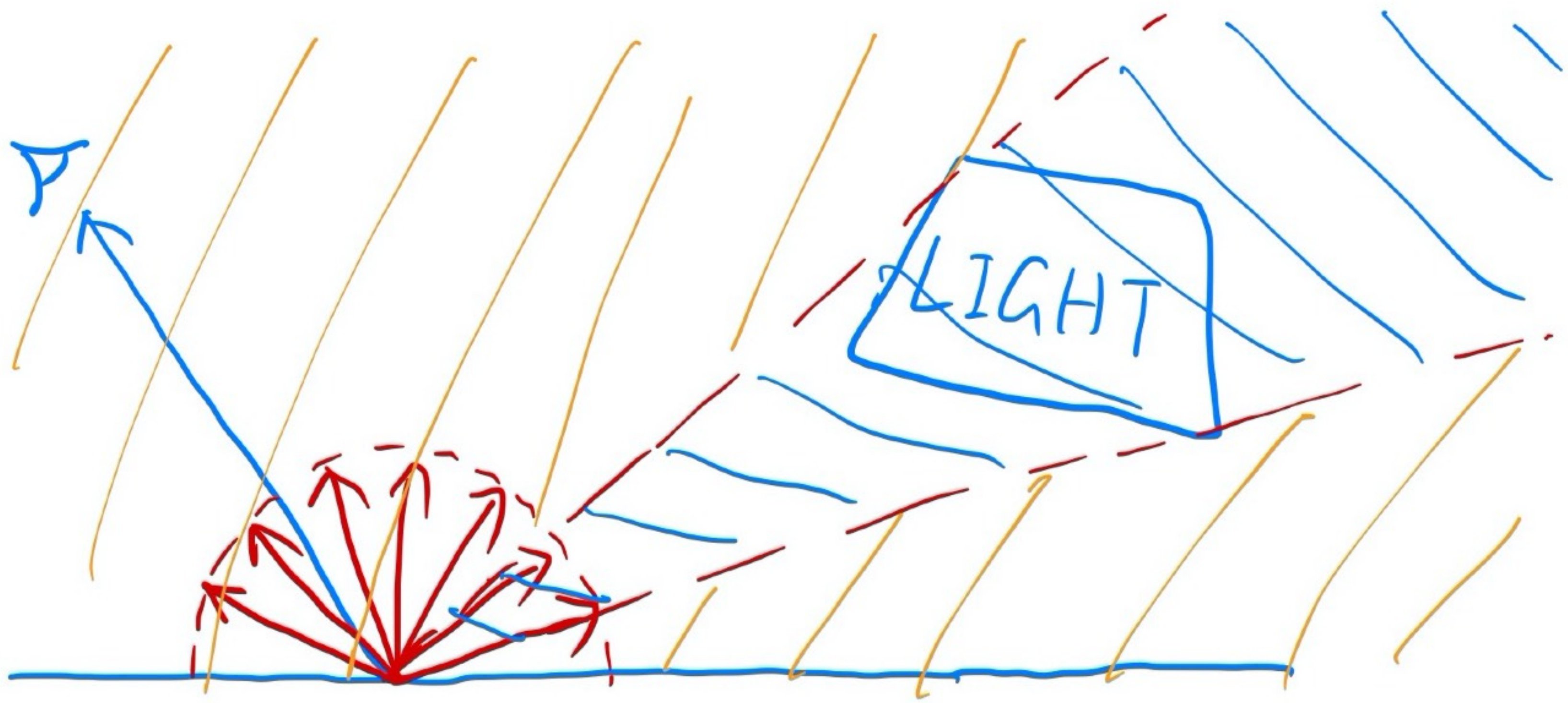
Pdf: 1 / A

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)



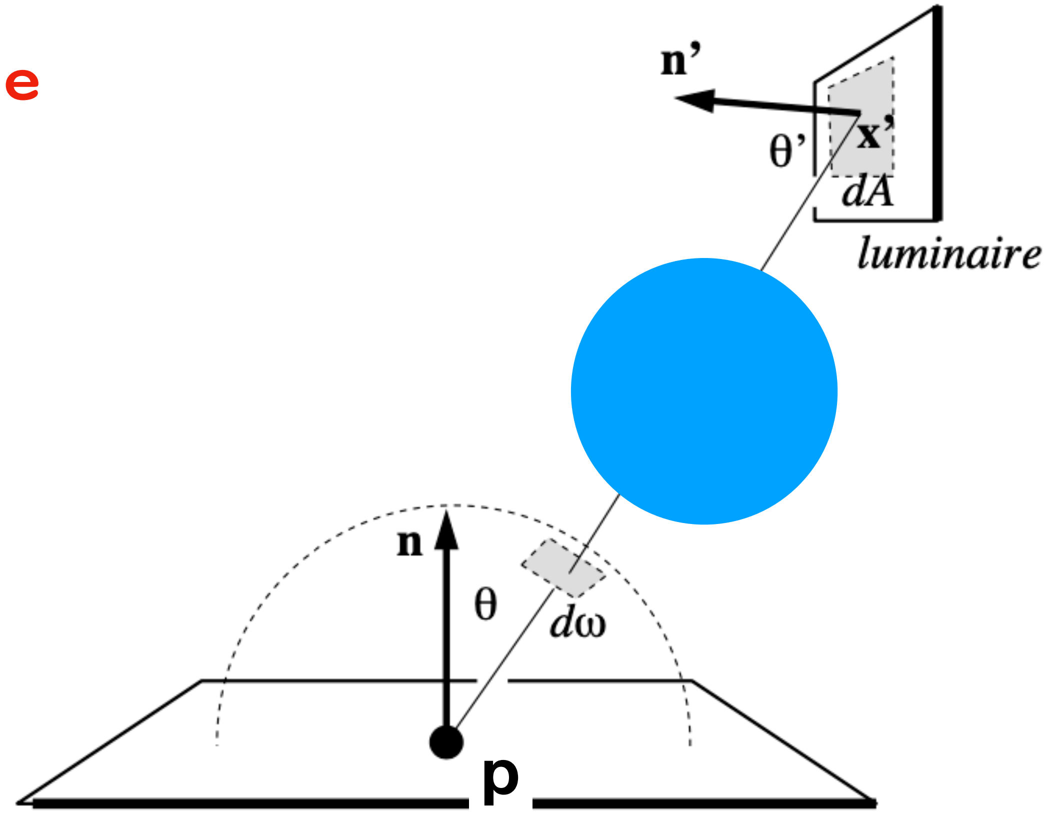
Sampling the Light

```
shade(p, wo)
    # 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
        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!)