北京航空航天大學

人工智能学院（人工智能研究院）

Computer Graphics
Lecture 17: Ray Tracing IV (Monte Carlo Path Tracing)

潘成伟 (Chengwei Pan)

Email: pancw@buaa.edu.cn

Office: Room B1021, New Main Building

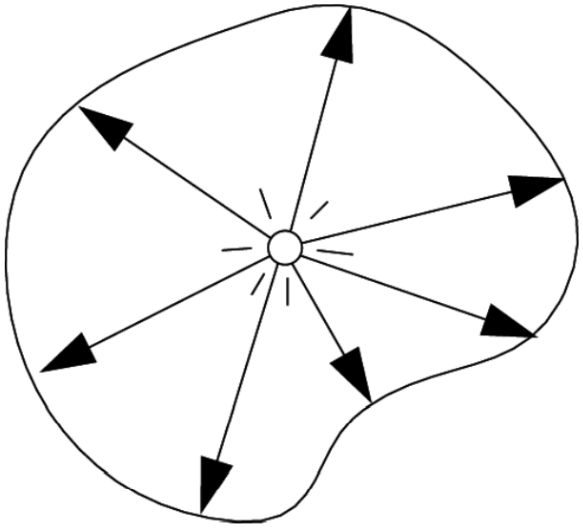
北京航空航天大学，人工智能学院

School of Artificial Intelligence, Beihang University

Last Lecture

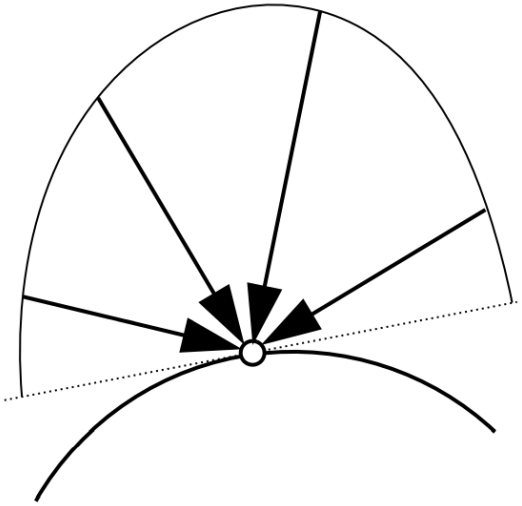
- Radiometry
  - Radiant flux, intensity, irradiance, radiance
- Light transport
  - The reflection equation
  - The rendering equation
- Global illumination

Important Light Measurements of Interest



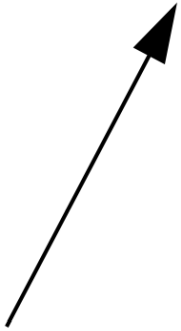
Light Emitted
From A Source

"Radiant Intensity"



Light Falling
On A Surface

"Irradiance"



Light Traveling
Along A Ray

"Radiance"

Radiance

- Definition: power **per unit solid angle** **per projected unit area**.

$$L(\mathrm{p},\omega) \equiv \frac{\mathrm{d}^2\Phi(\mathrm{p},\omega)}{\mathrm{d}\omega\,\mathrm{d}A\cos\theta}$$

- Recall
  - Irradiance: power per projected unit area
  - Intensity: power per solid angle

- So
  - Radiance: Irradiance per solid angle
  - Radiance: Intensity per projected unit area

The Rendering Equation

- Re-write the reflection equation:

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

by adding an Emission term to make it general!

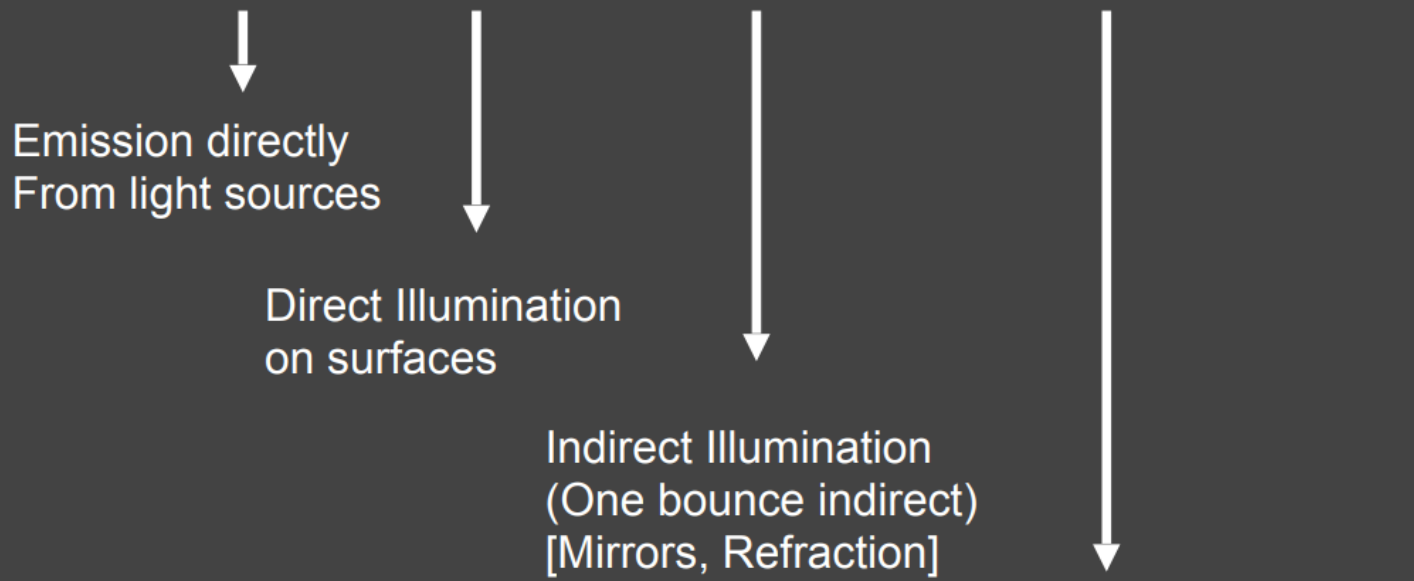
- The rendering equation

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

Note: now, we assume that all directions are pointing outwards!

Ray Tracing

$$L = E + KE + K^2E + K^3E + \ldots$$



This Lecture

- Monte Carlo Integration

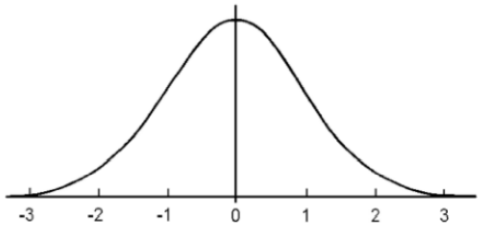
- Path Tracing

Monte Carlo Integration

Probabilities

- Continuous Variable and Probability Density Functions

$$X \sim p(x)$$



  - Conditions on p(x)

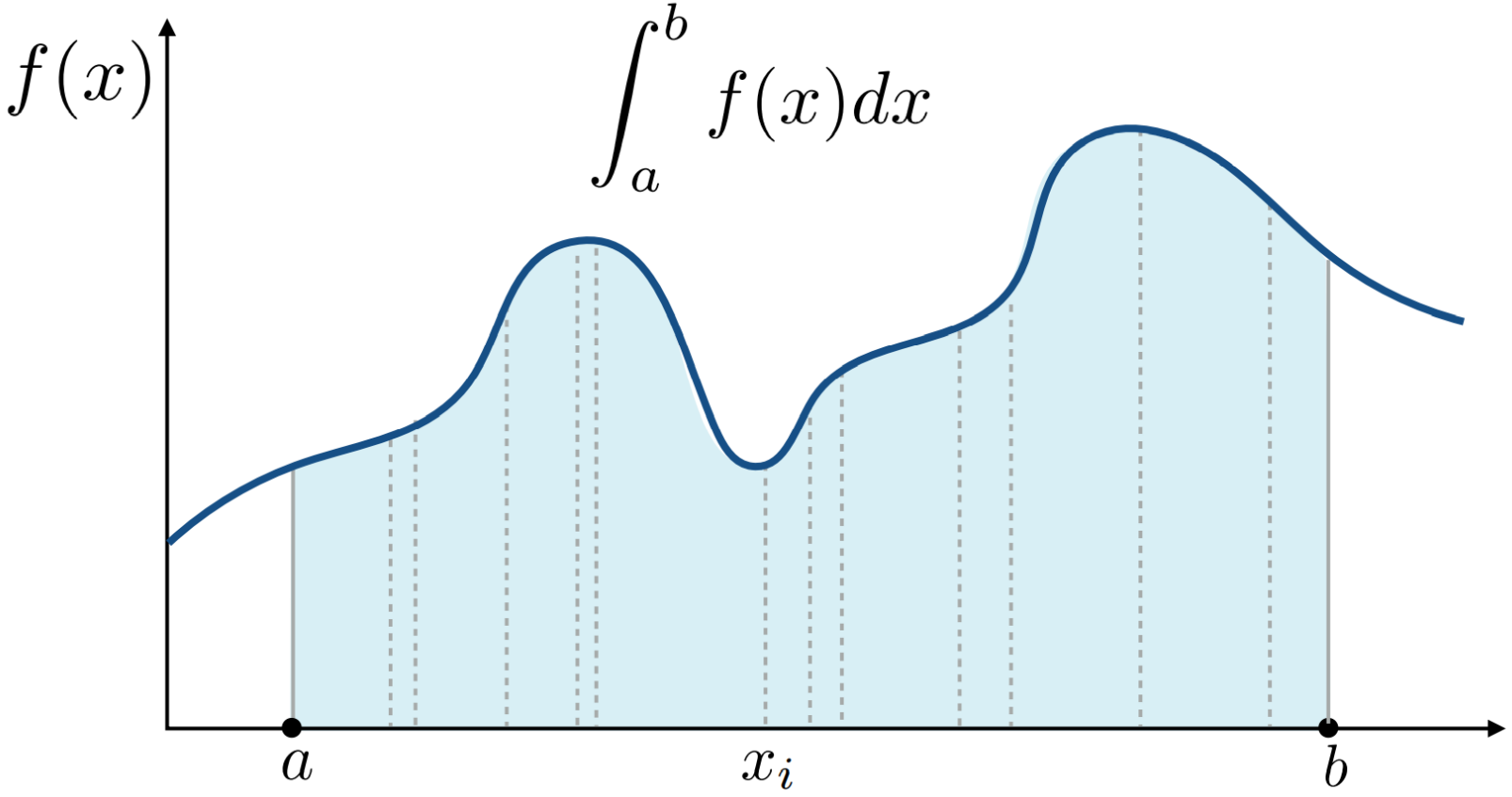
$$p(x) \geq 0 \text{ and } \int p(x)\,dx = 1$$

  - Expected value of X

$$E[X] = \int x\,p(x)\,dx$$

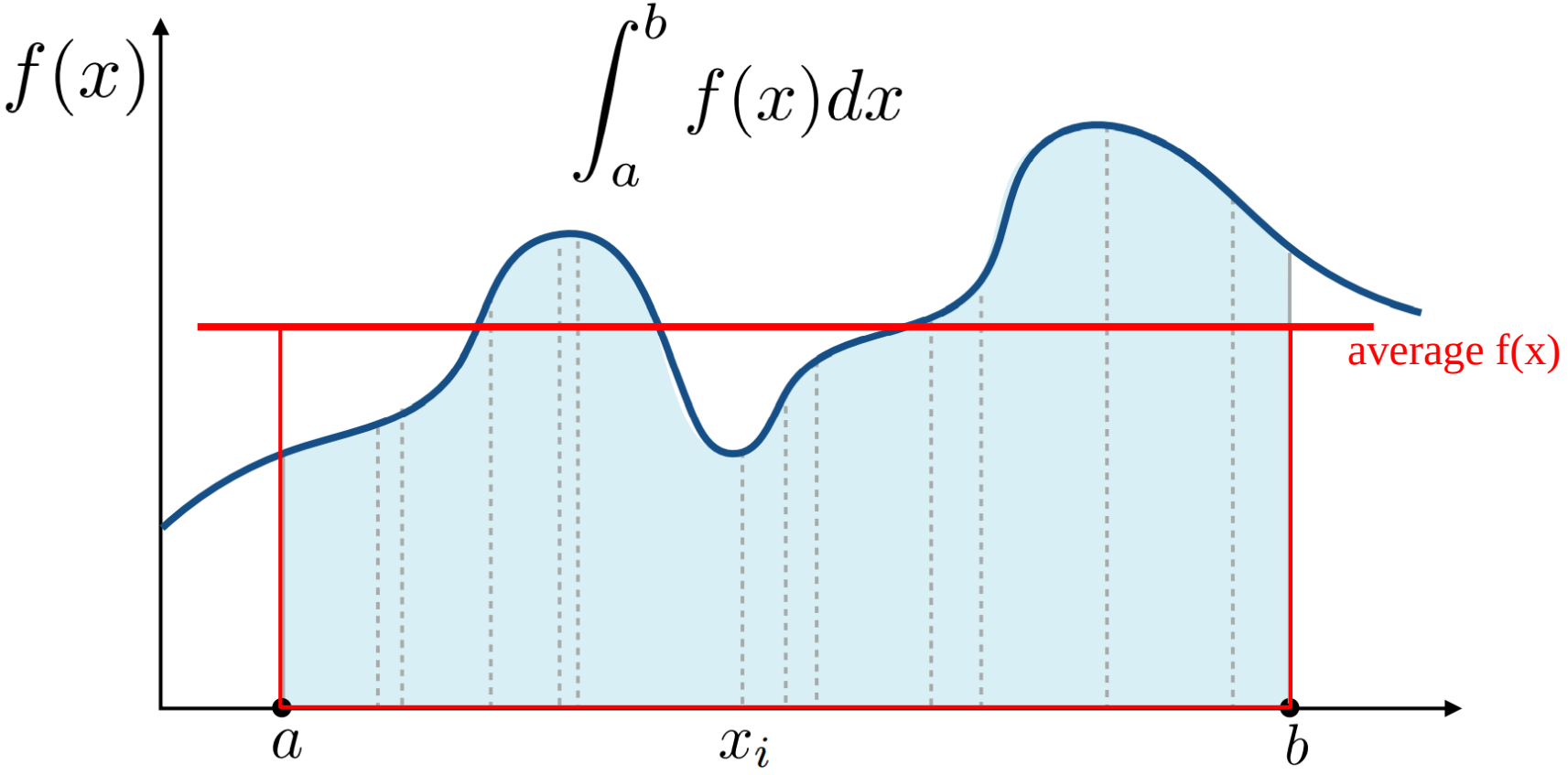
Monte Carlo Integration

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

Monte Carlo Integration

- Proof

$$
\begin{aligned}
E[F_N] &= E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}\right] \\
&= \frac{1}{N}\sum_{i=1}^{N} E\left[\frac{f(X_i)}{p(X_i)}\right] \\
&= \frac{1}{N}\sum_{i=1}^{N}\int_a^b \frac{f(x)}{p(x)}\,p(x)\,\mathrm{d}x \\
&= \frac{1}{N}\sum_{i=1}^{N}\int_a^b f(x)\,\mathrm{d}x \\
&= \int_a^b f(x)\,\mathrm{d}x
\end{aligned}
$$

Example: Uniform Monte Carlo Estimator

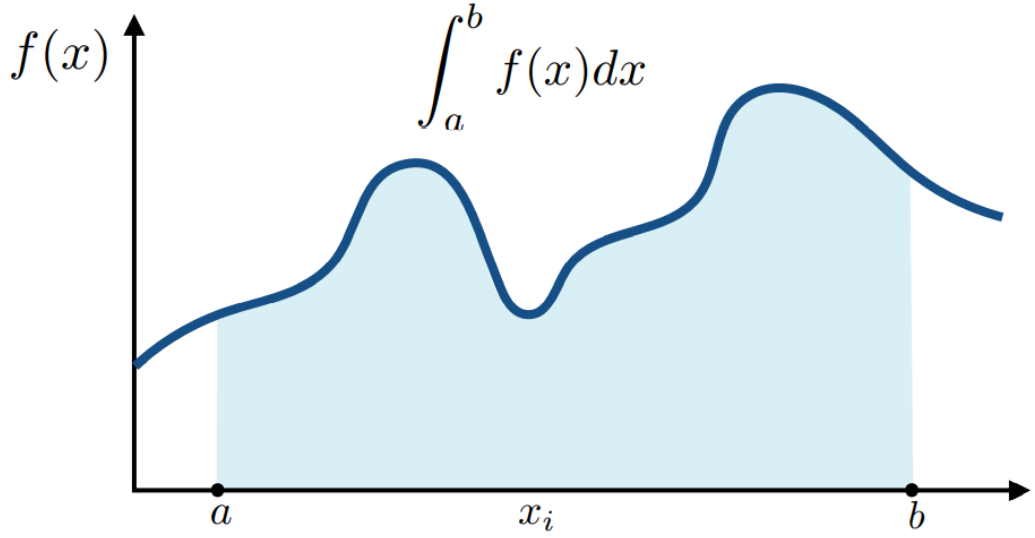
- Uniform random variable

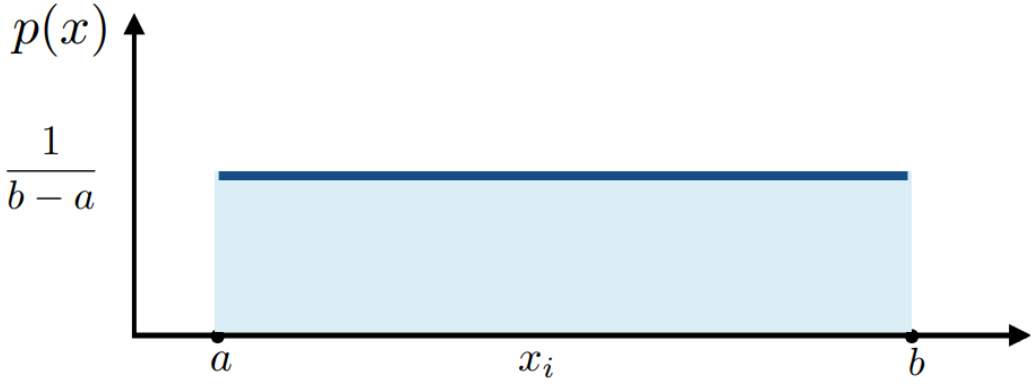
$$X_i \sim p(x) = C \quad \textbf{(constant)}$$

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

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

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





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 $$X_i \sim p(x) = \frac{1}{b-a}$$
  - 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:
  - 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 path tracing algorithm!

Whitted-Style Ray Tracing: Problem 1

- Where should the ray be reflected for glossy materials?

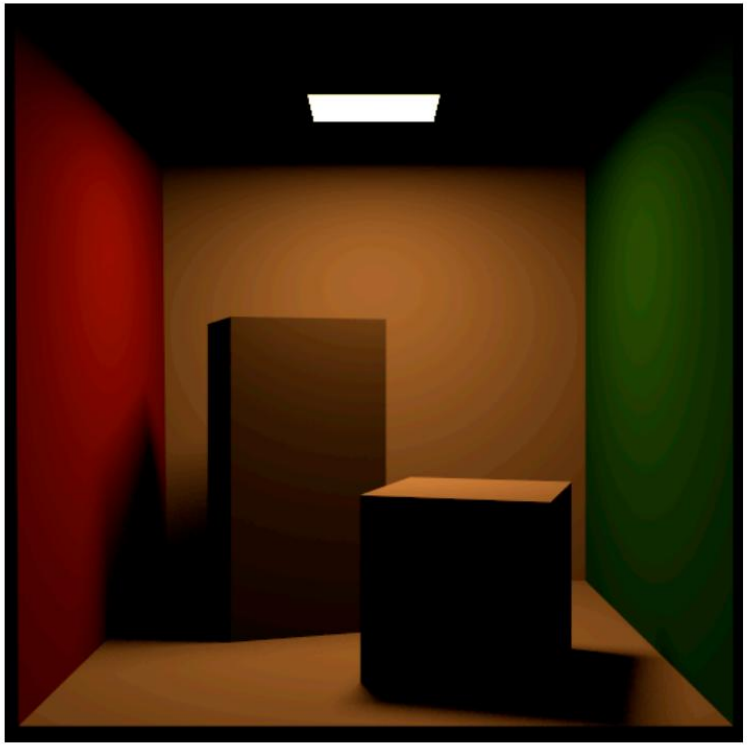
Mirror Reflection

Glossy Reflection

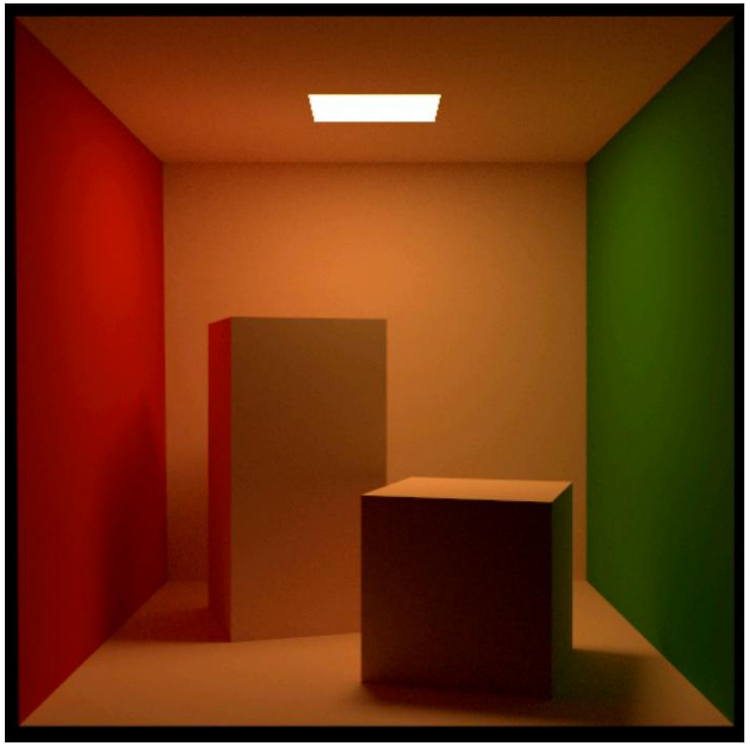
The Utah teapot

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 to solve an integral numerically?

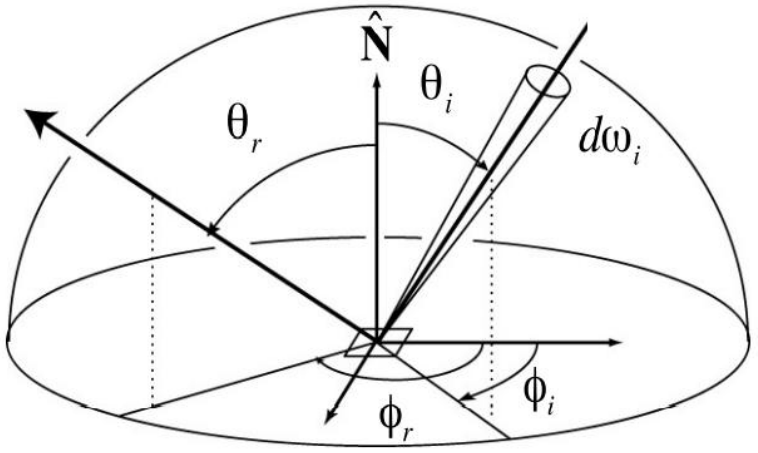
A Simple Monte Carlo Solution

- Abuse the concept of Reflection Equation a little bit

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

- 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

- 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:

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

  - f(x): $L_i(p,\omega_i) f_r(p,\omega_i,\omega_o)(n\cdot\omega_i)$
  - Pdf: $p(\omega_i) = 1/2\pi$

    assume uniformly sampling the hemisphere

A Simple Monte Carlo Solution

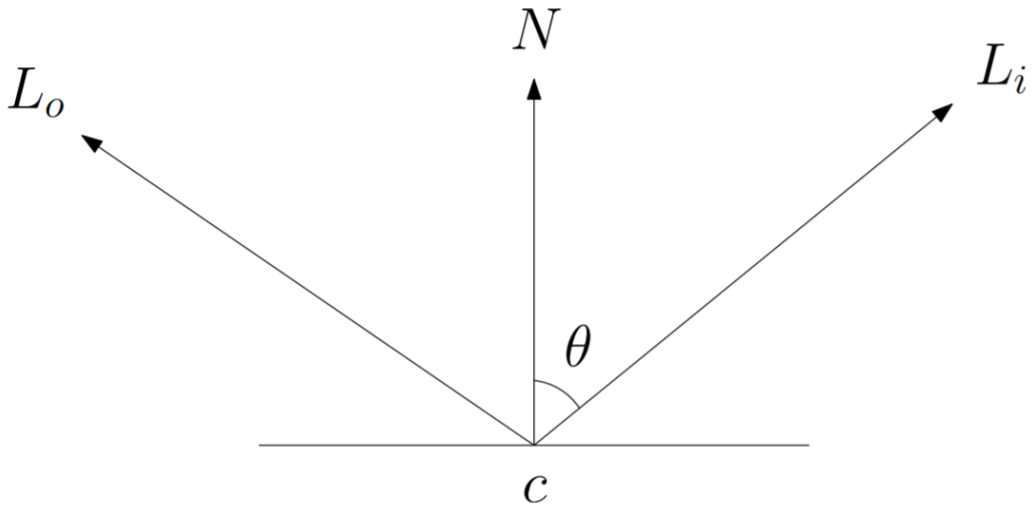
- So, in general

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

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

- What does it mean?
  - A correct shading algorithm for direct illumination!

Computing a Ray Bounce



- L_o is the outgoing radiance
- L_i is the incoming radiance along a randomly reflected ray
- Outgoing radiance for this sample is cL_i , where

$$c = \frac{f_r(p, \omega_i, \omega_0) \cos\theta}{p(\omega_i)}$$

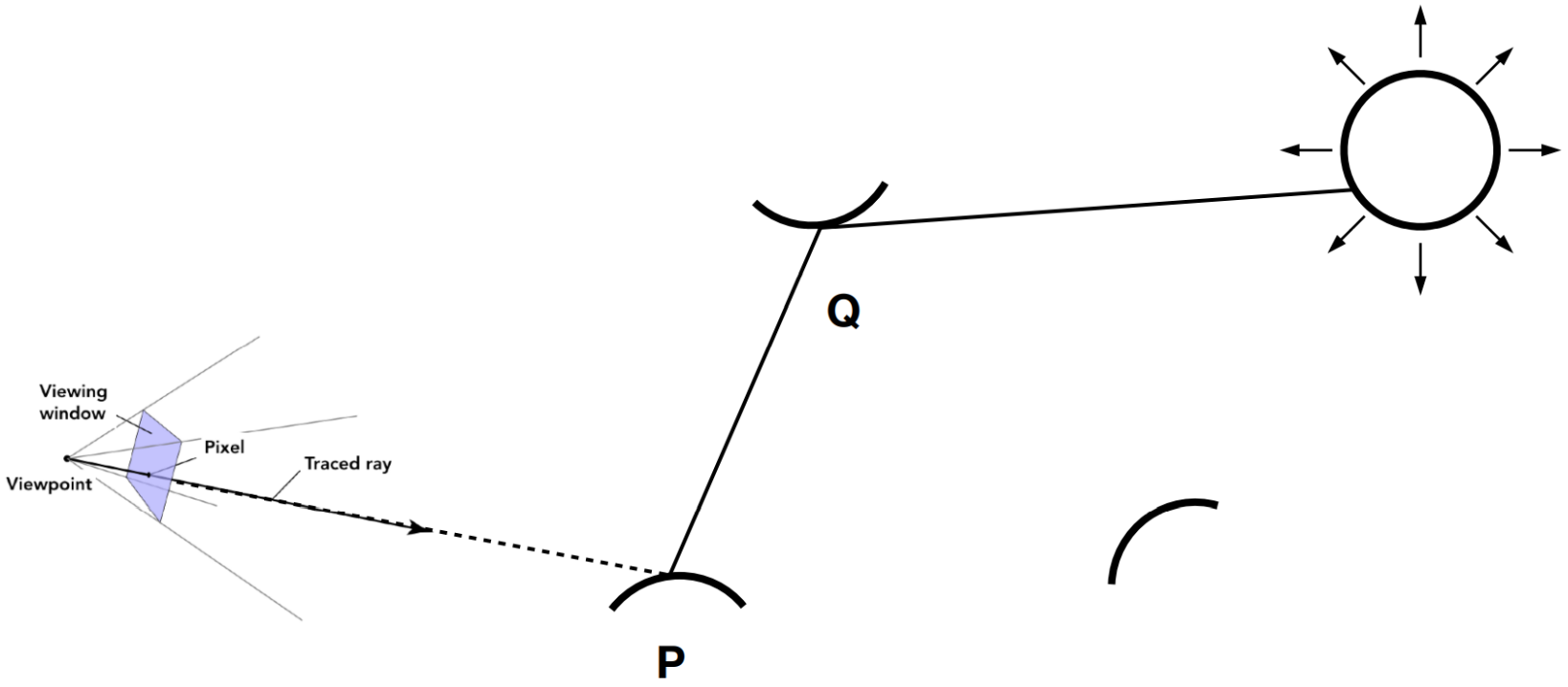
A Simple Monte Carlo Solution

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

```
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
```

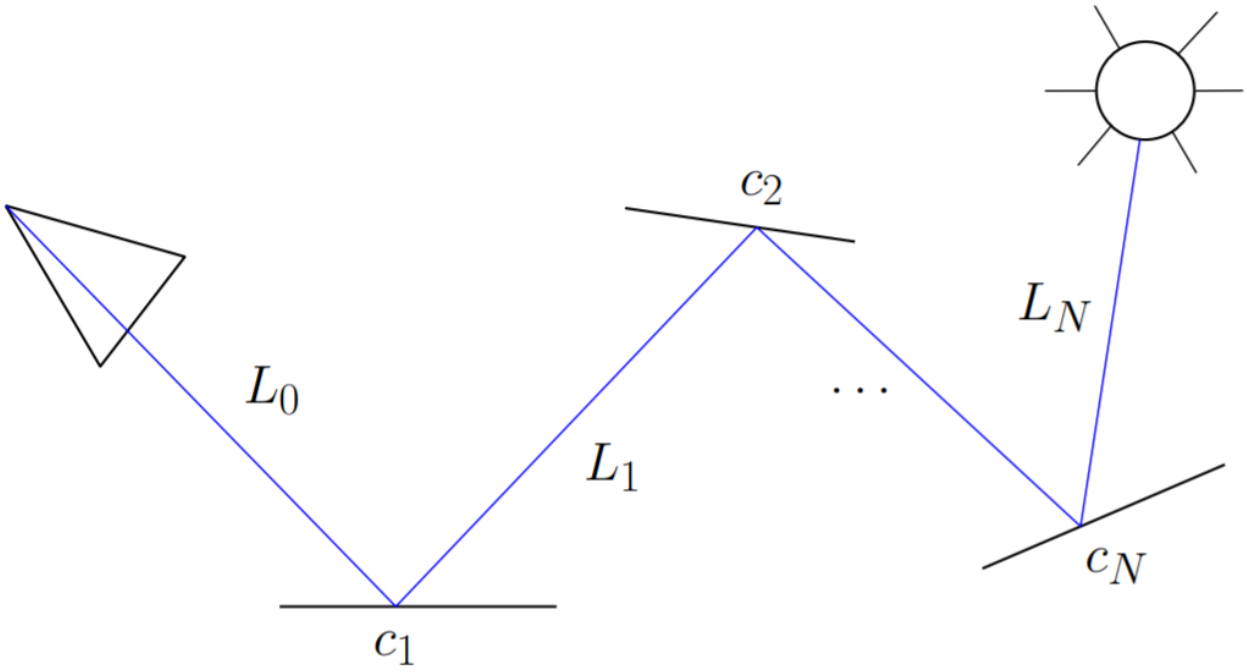
Introducing Global Illumination

- 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



$$L_0 = c_1 L_1 = c_1(c_2 L_2) = c_1(c_2(c_3 L_3))$$
$$= c_1(c_2(c_3(\cdots(c_N L_N)\cdots))) = c_1 c_2 \cdots c_N L_N$$
$$= L_N \prod_{i=1}^{N} c_i$$

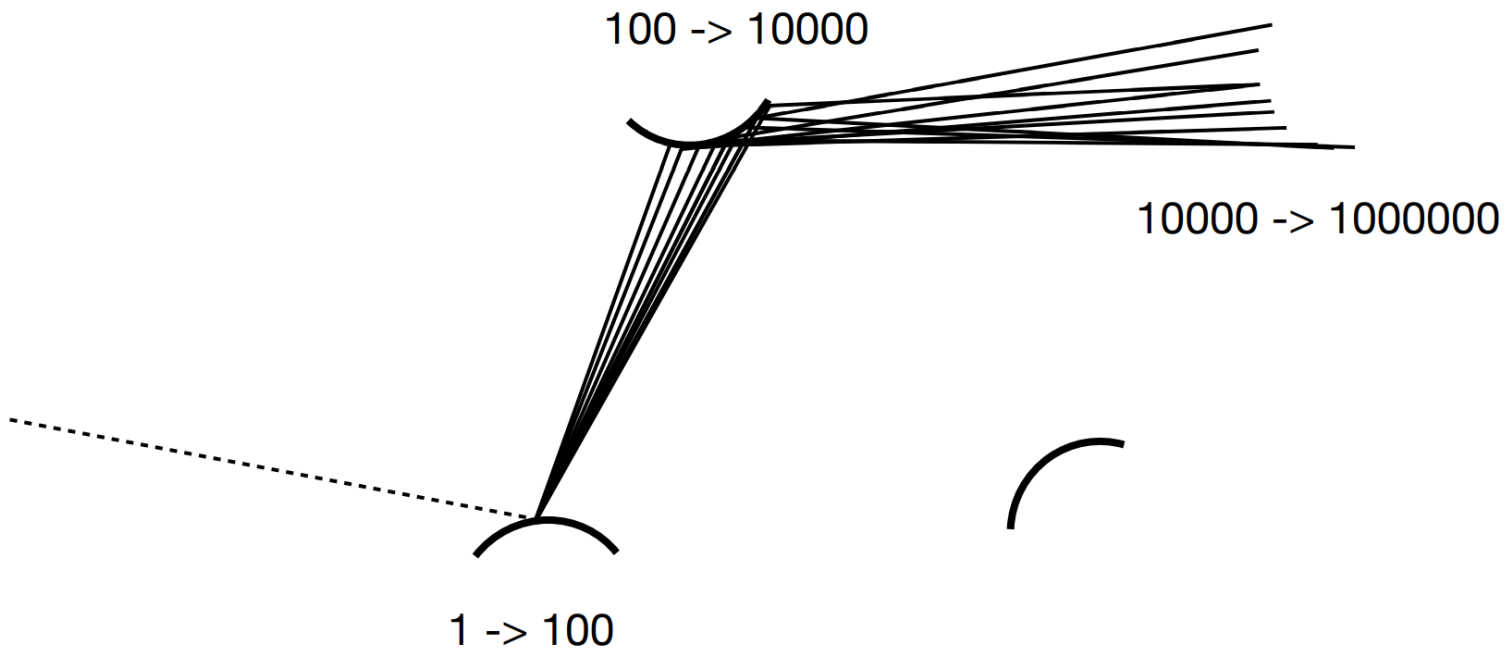
Introducing Global Illumination

```
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

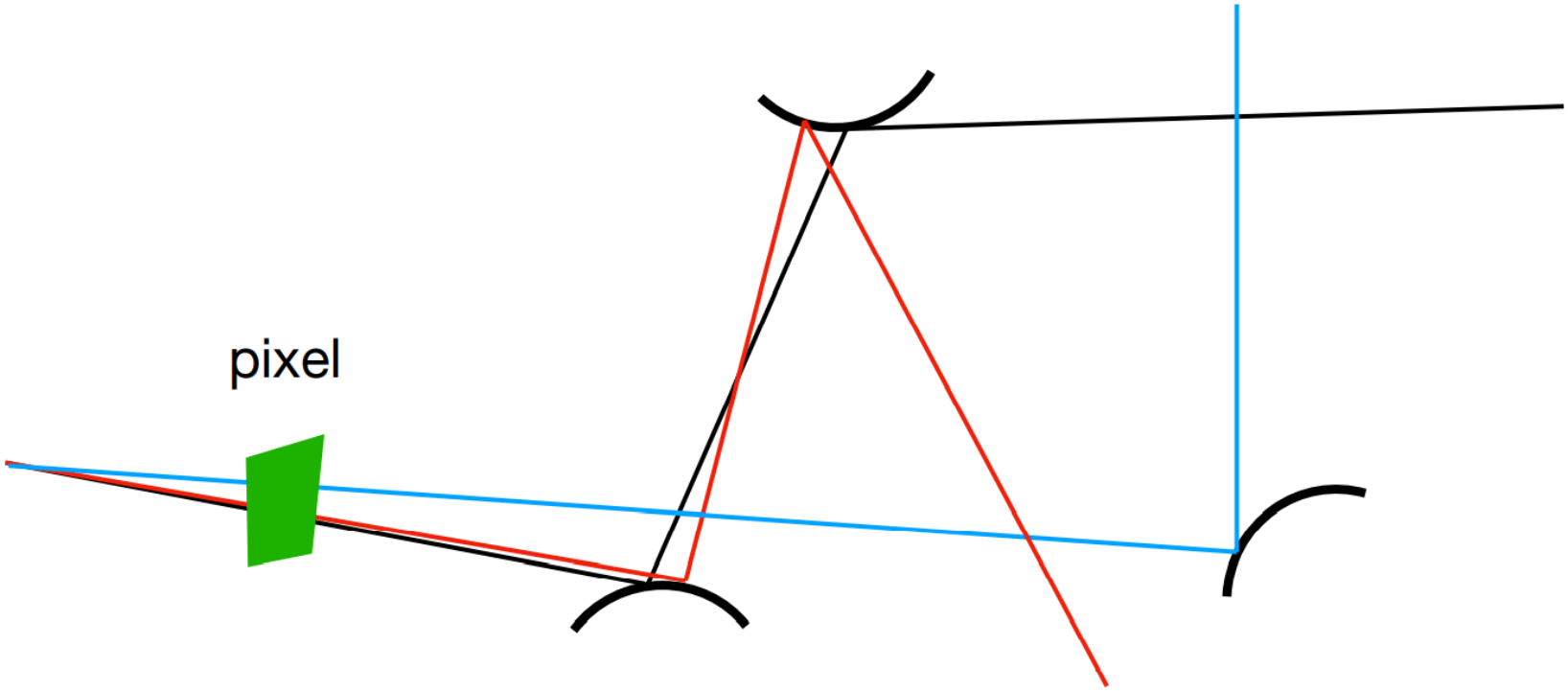
- From now on, we always assume that only 1 ray is traced at each shading point:

```
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

- Previously, we always shoot a ray at a shading point and get the shading result $\boldsymbol{L_o}$

- Suppose we manually set a probability P ($0 < P < 1$)
- With probability P, shoot a ray and return the shading result divided by P: $\boldsymbol{L_o}$ / 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!:

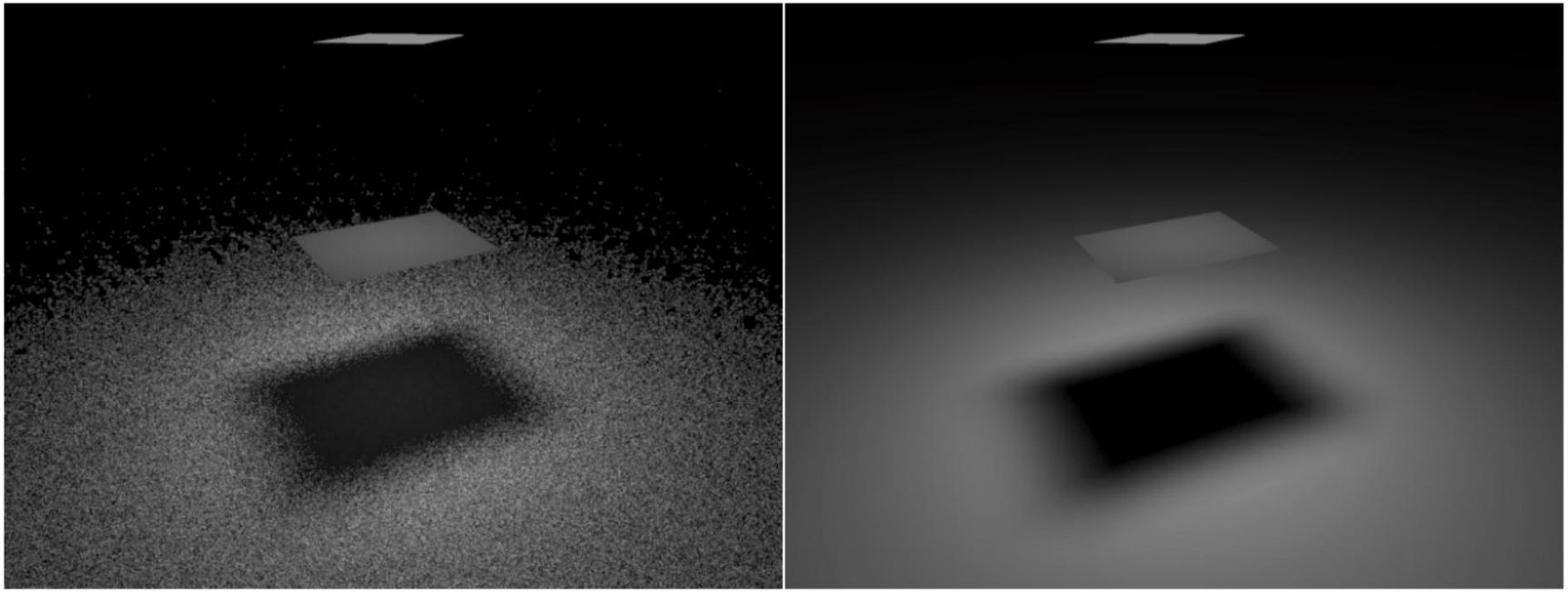
$$E = P * (\boldsymbol{L_o} / P) + (1 - P) * 0 = \boldsymbol{L_o}$$

Solution: Russian Roulette

```
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

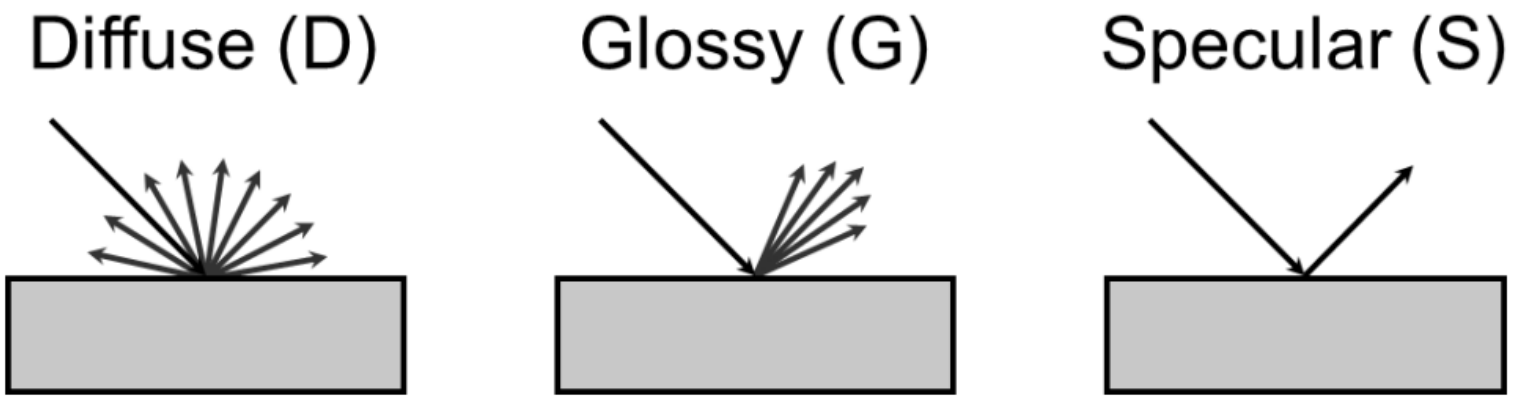
(Multiple) Importance Sampling

- For a basic path tracer to work properly, we should perform two optimizations:
  - BRDF sampling
  - Lighting sampling

These are two forms of importance sampling. For more details, see:

http://www.slideshare.net/takahiroharada/introduction-to-bidirectional-path-tracing-bdpt-implementation-using-opencl-cedec-2015

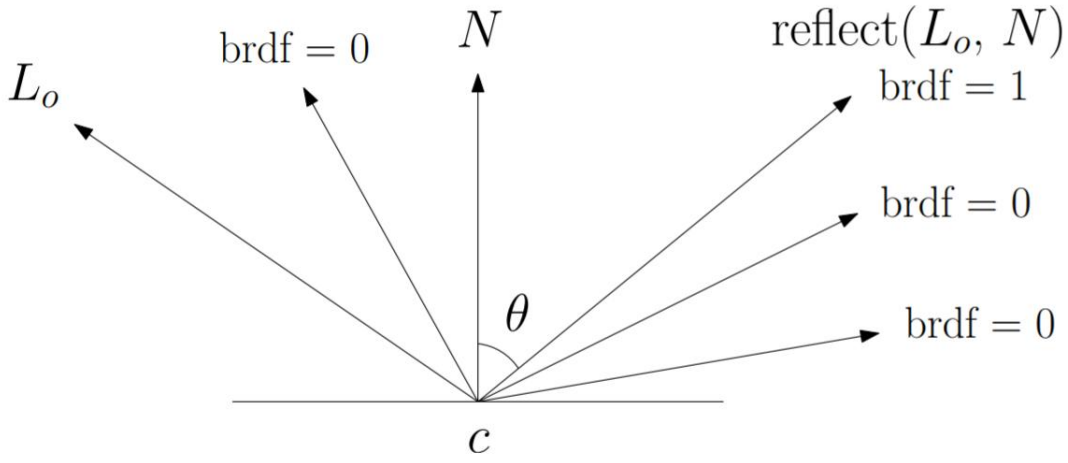
Common BRDFs



- Specular reflects light in a single direction - the direction of mirror reflection.
- Diffuse reflects light equally in all directions.
- Glossy reflects light in a cone centered in the direction of mirror reflection.

BRDF Sampling: Specular Reflection

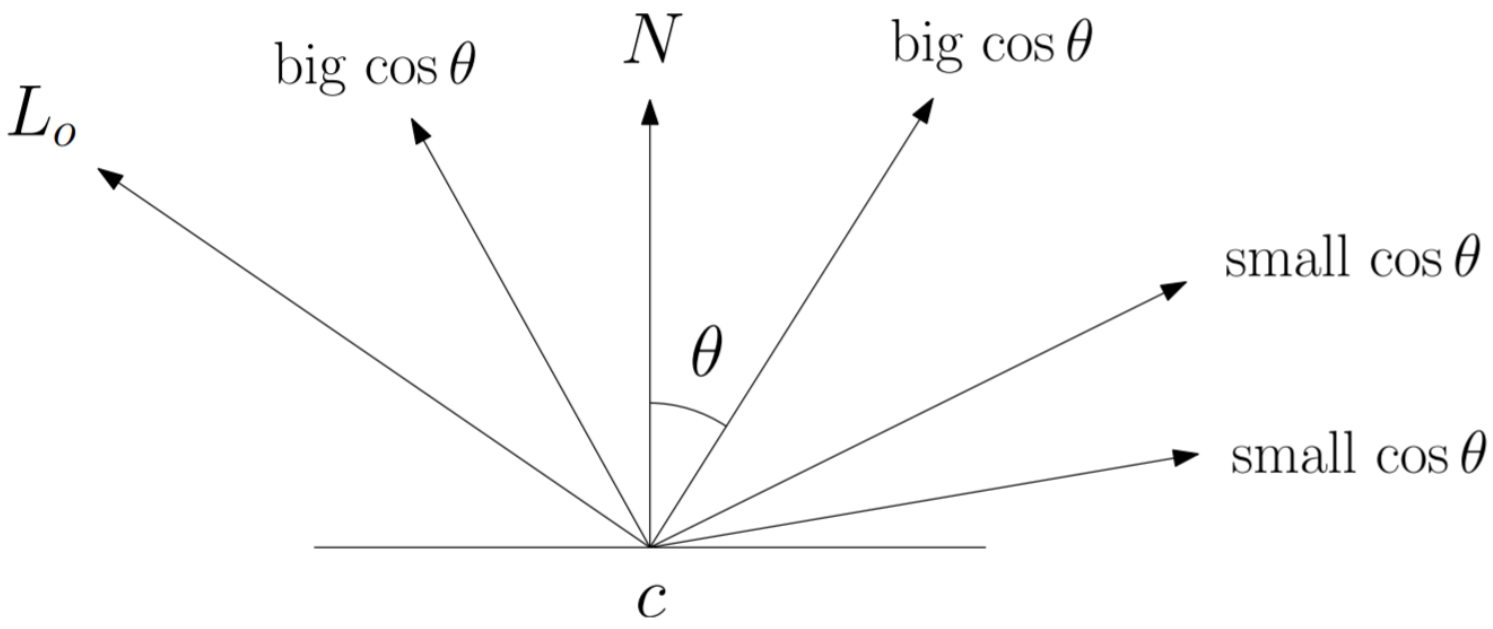
- Consider what happens when we shoot randomly reflected rays from a perfectly specular surface:



- The specular BRDF is 1 in the direction of mirror reflection, and 0 everywhere else.
- We cannot approximate the integral by shooting random rays; we need to sample the direction of mirror reflection explicitly.

BRDF Sampling: Diffuse Reflection

- Similarly, consider shooting random rays from a diffuse surface:



- Because of the cos θ term in the rendering equation, rays at grazing angles have little contribution to the final result.

BRDF Sampling: Diffuse Reflection

- For faster convergence on diffuse surfaces, we should focus more on rays near the normal than at grazing angles.

- To this end, instead of sampling the hemisphere uniformly, we sample it according to a cosine-weighted distribution:

$$p(\omega) \propto \cos\theta$$

- As usual, the pdf must add up to 1:

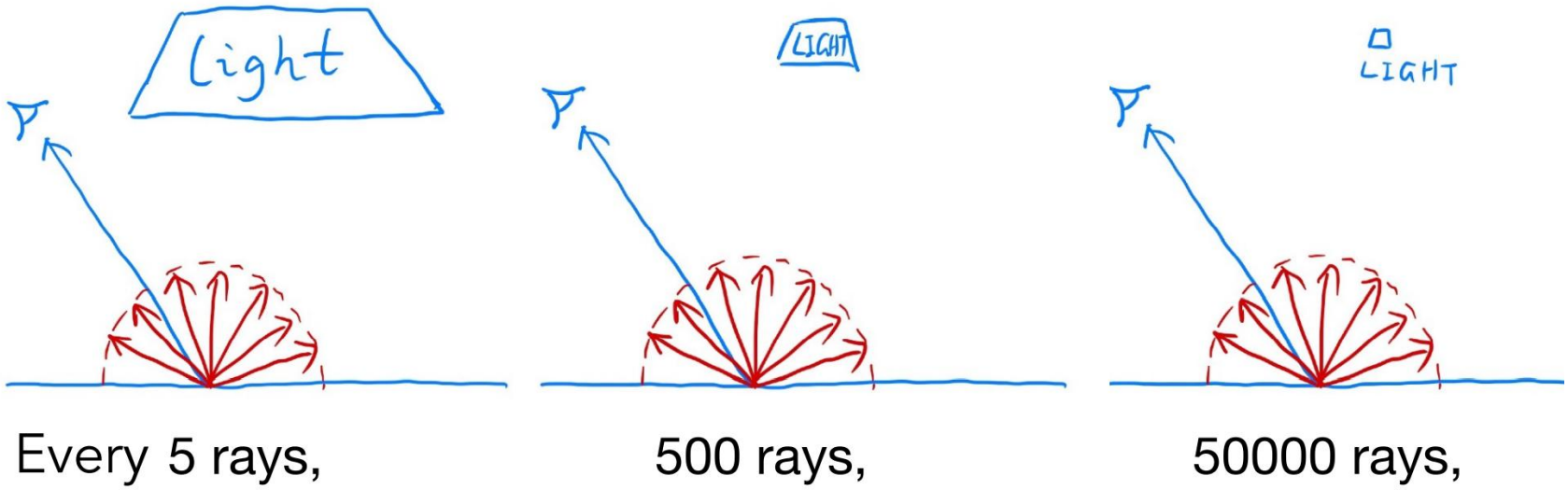
$$\int_\Omega k \cos\theta \, \mathrm{d}\omega = 1 \Leftrightarrow k = \frac{1}{\pi}$$

- Therefore

$$p(\omega) = \frac{\cos\theta}{\pi}$$

Light Sampling

- Understanding the reason of being inefficient

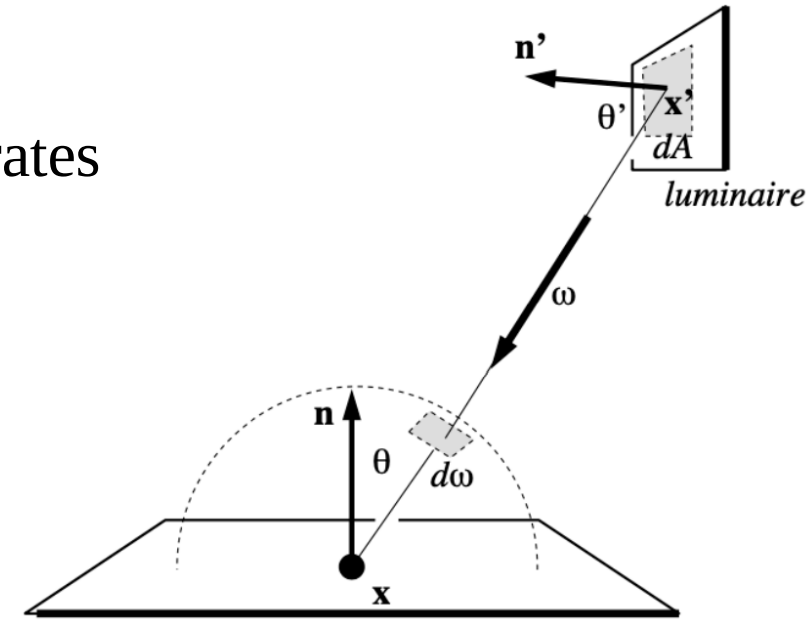


there will be 1 ray hitting the light. So a lot of rays are “wasted” if we uniformly sample the hemisphere at the shading point

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: $L_o = \int L_i\, f_r\, \cos\theta\, d\omega$
- Recall Monte Carlo Integration: Sample on x & integrate on x



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

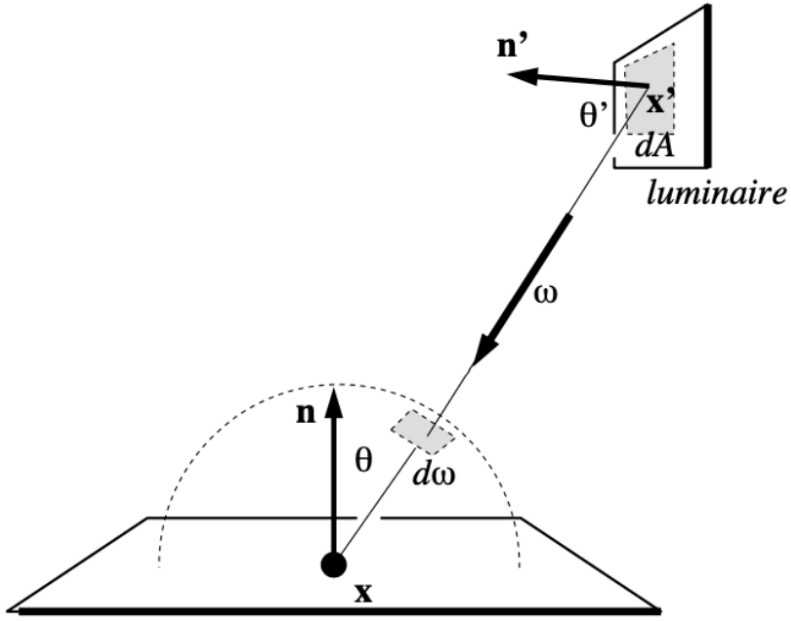
Sampling the Light (pure math)

- 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

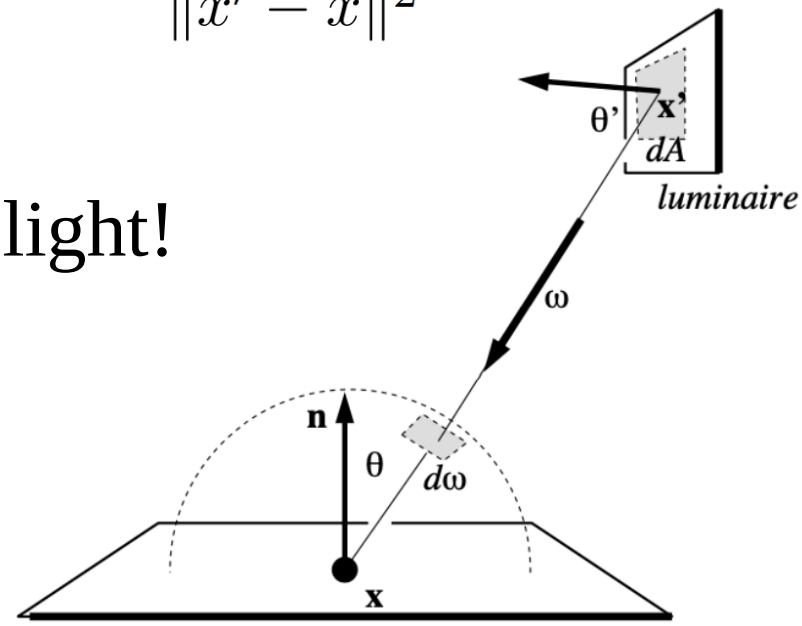


Sampling the Light

- Then we can rewrite the rendering equation as

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

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



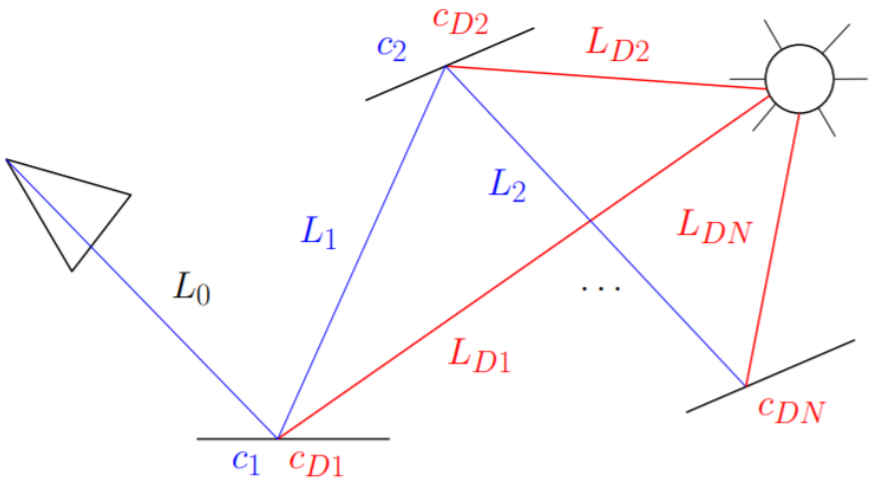
- Now an integration on the light!
- Monte Carlo integration:
  - "f(x)": everything inside
  - Pdf: 1 / A

Sampling the Light

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

- Now we consider the radiance coming from two parts:
  - light source (direct, no need to have RR)
  - other reflectors (indirect, RR)

$$L_0 = c_{D1}L_{D1} + c_1 L_1$$
$$= c_{D1}L_{D1} + c_1(c_{D2}L_{D2} + c_2 L_2)$$



Sampling the Light

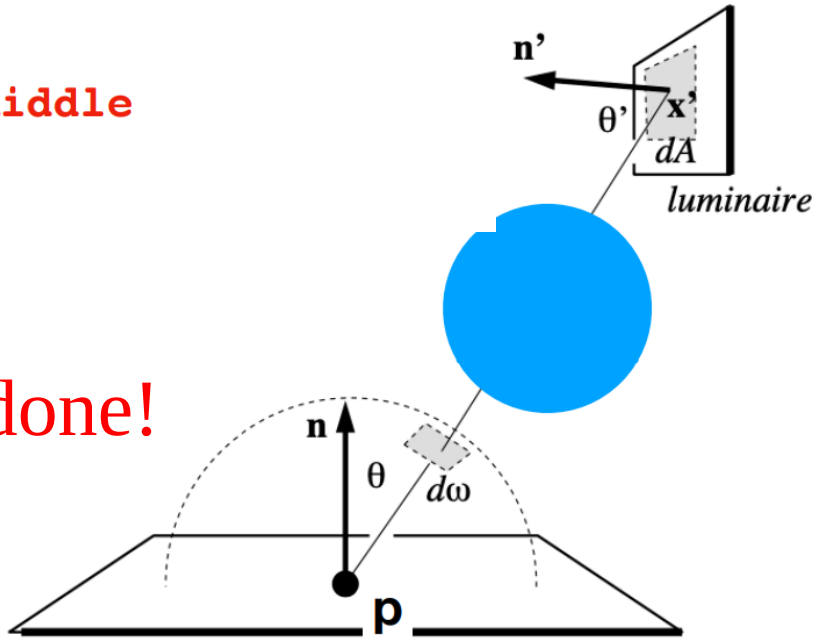
```
shade(p, wo)
    # Contribution from the light source.
    Uniformly sample the light at x' (pdf_light = 1 / A)
    L_dir = L_i * f_r * cos θ * cos θ' / |x' - p|^2 / pdf_light

    # 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
```

Sample the Light

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

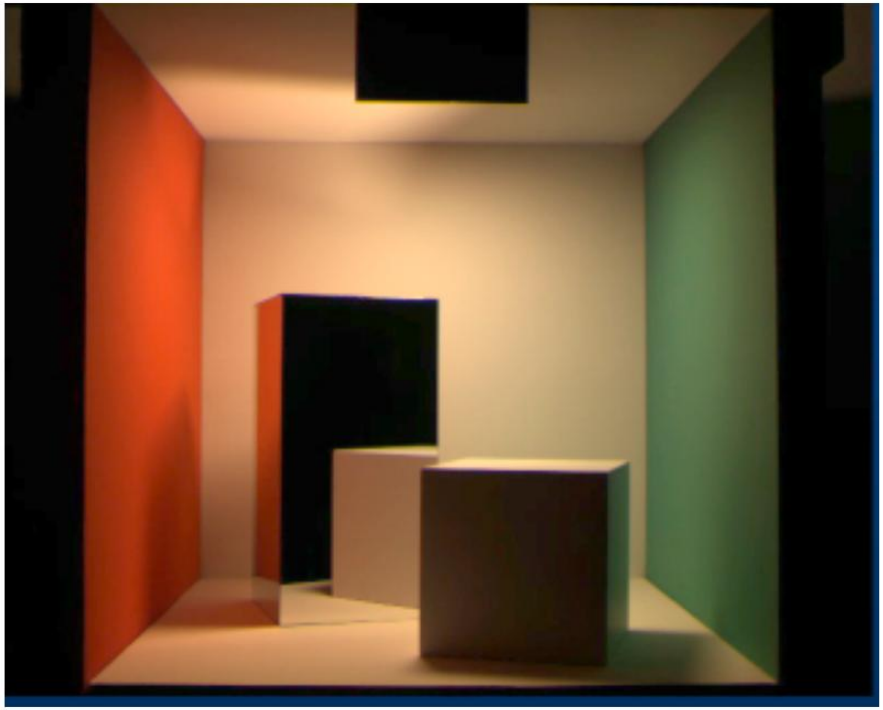
```
# Contribution from the light source.
L_dir = 0.0
Uniformly sample the light at x' (pdf_light = 1 / A)
Shoot a ray from p to x'
If the ray is not blocked in the middle
    L_dir = …
```

- Now path tracing is finally done!

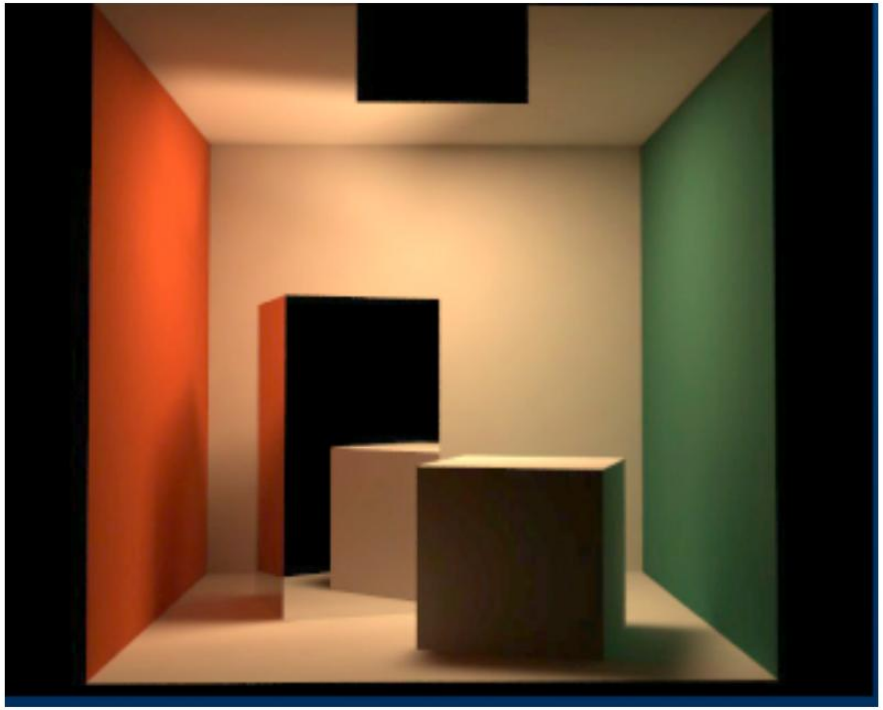


Is Path Tracing Correct?

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



Photo



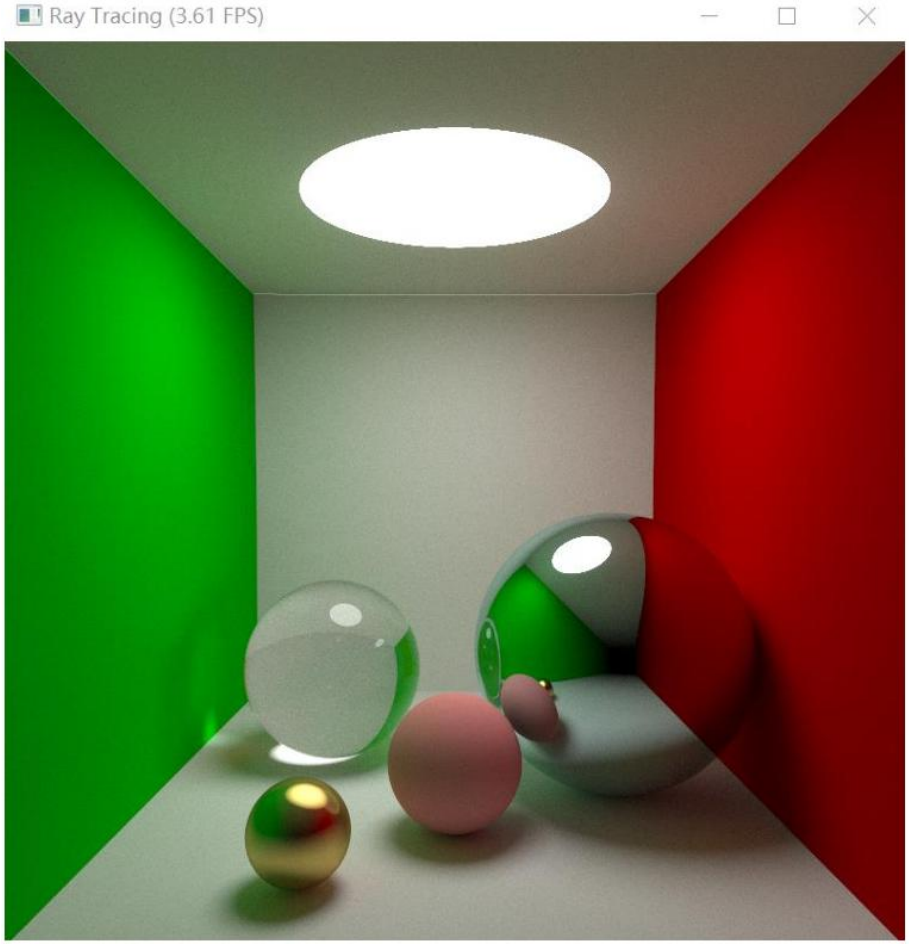
Path tracing:
Global illumination

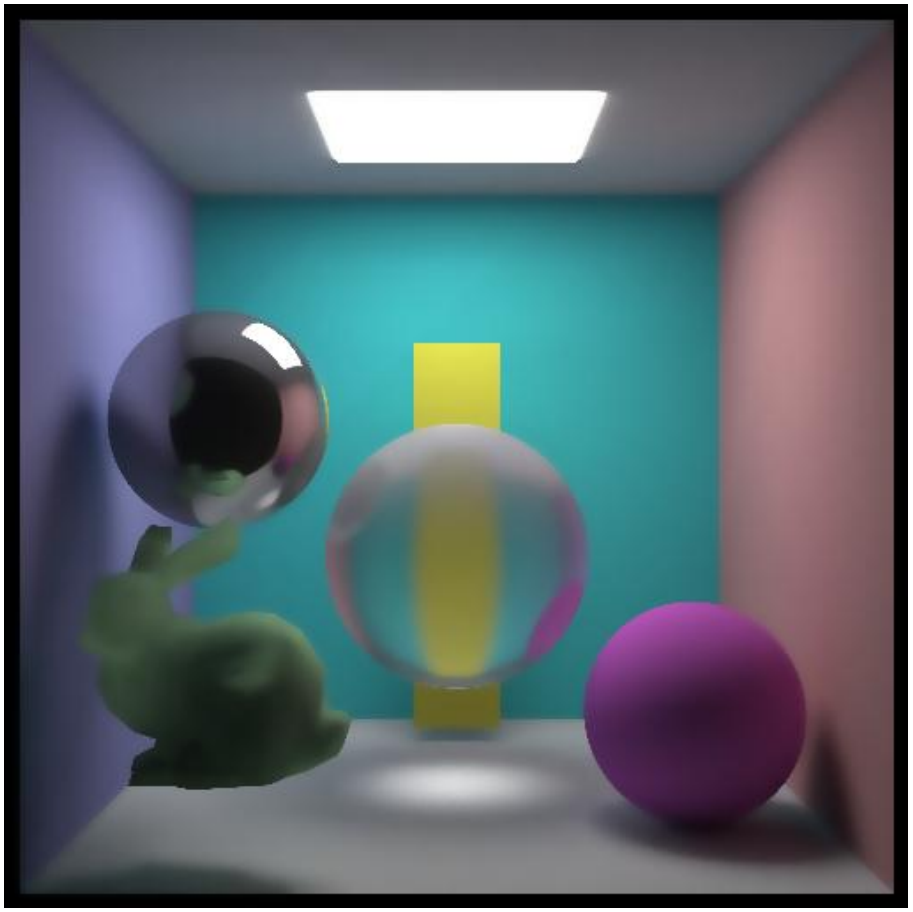
The Cornell box — http://www.graphics.cornell.edu/online/box/compare.html

Assignments

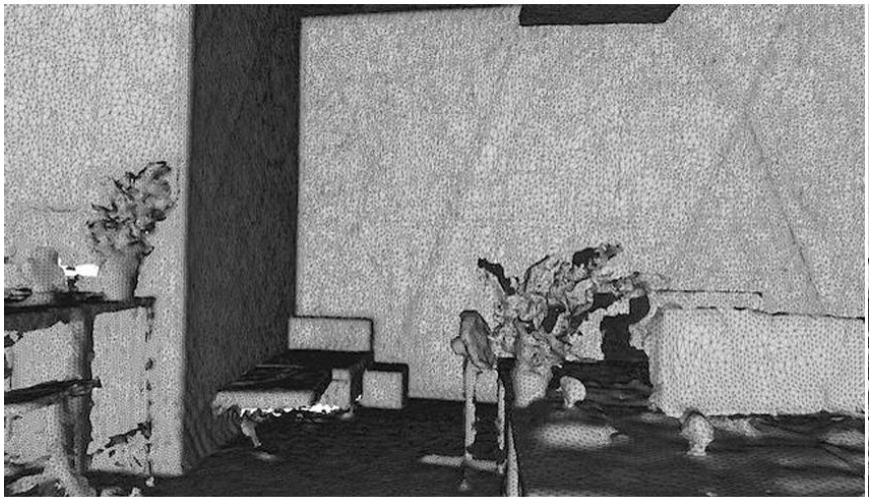
- 研究报告或大作业（代码和报告）
  - 选题不限
    - 小型绘制系统
    - 网格简化
    - 光线追踪
    - 动画、特效
    - 神经辐射场
  - 可以组队（人数不超过3人，备注分工情况及总分的分配比例）
  - 文件命名：学号+姓名+大作业名.zip
  - https://bhpan.buaa.edu.cn/link/AAC850F7FDC495485592DC5D906D85309B 文件夹名：大作业 有效期限：2026-01-26 23:59提取码：cg25

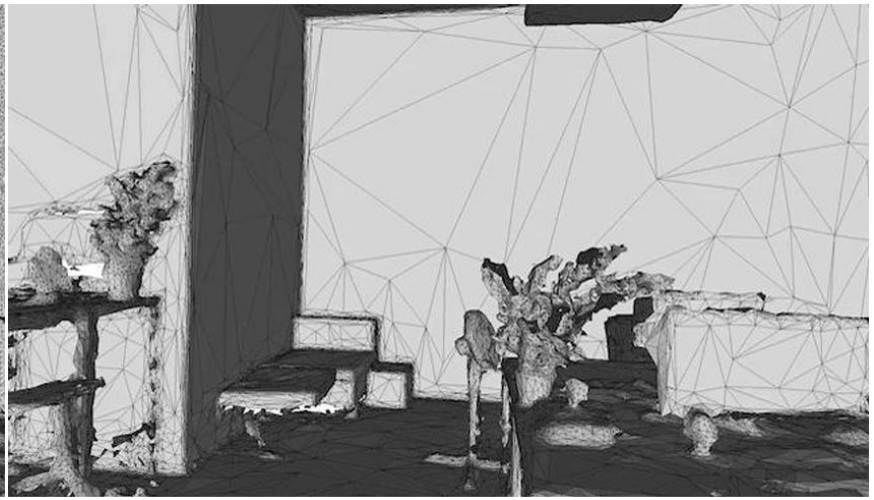
Ray Tracing

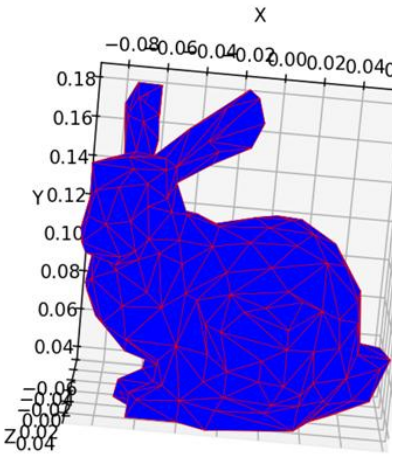


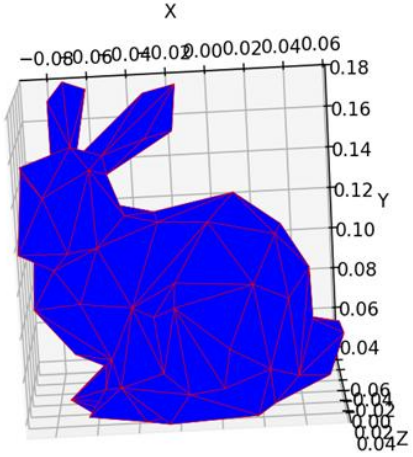


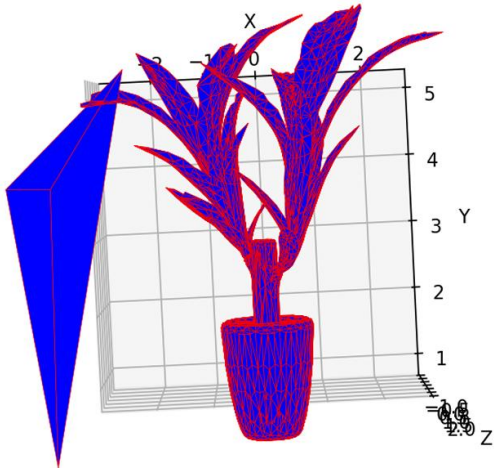
Mesh Simplification

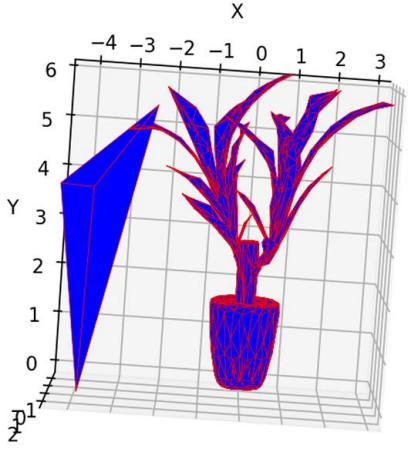




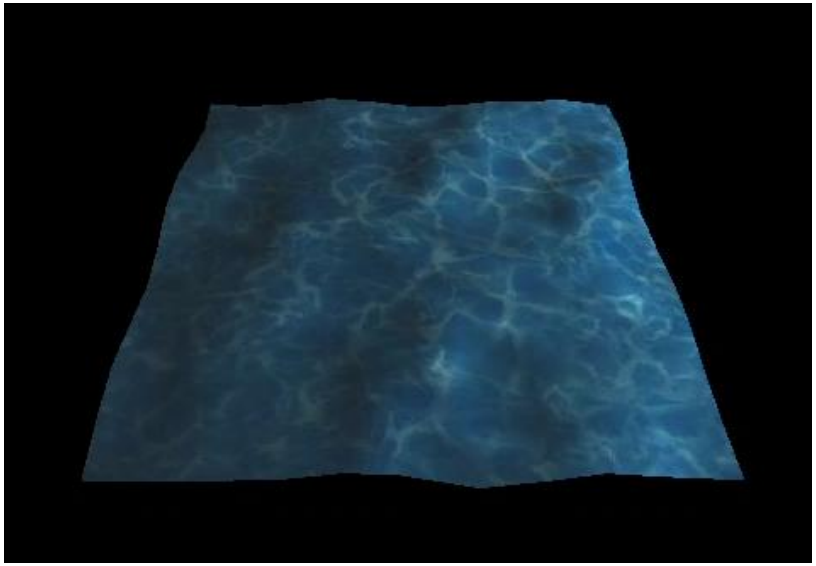


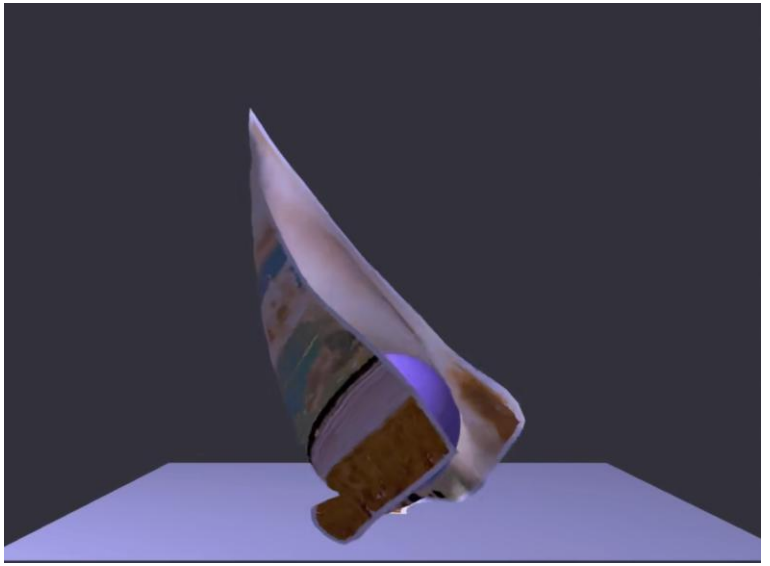


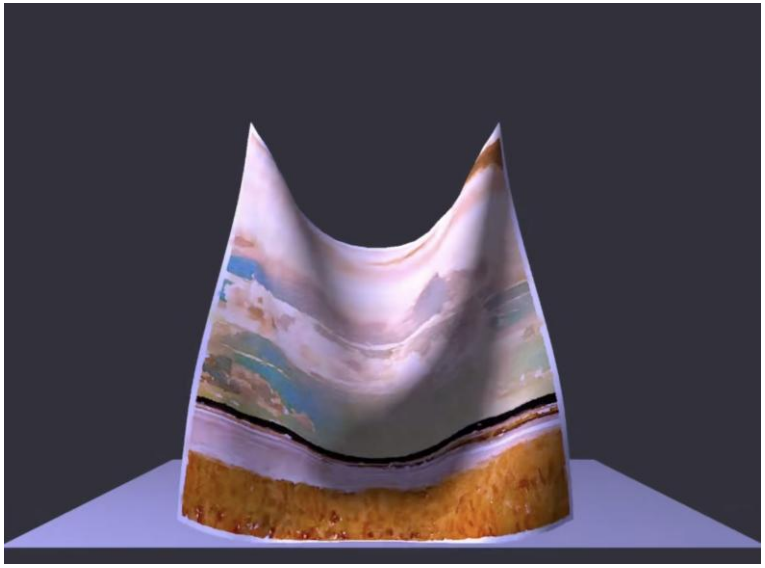


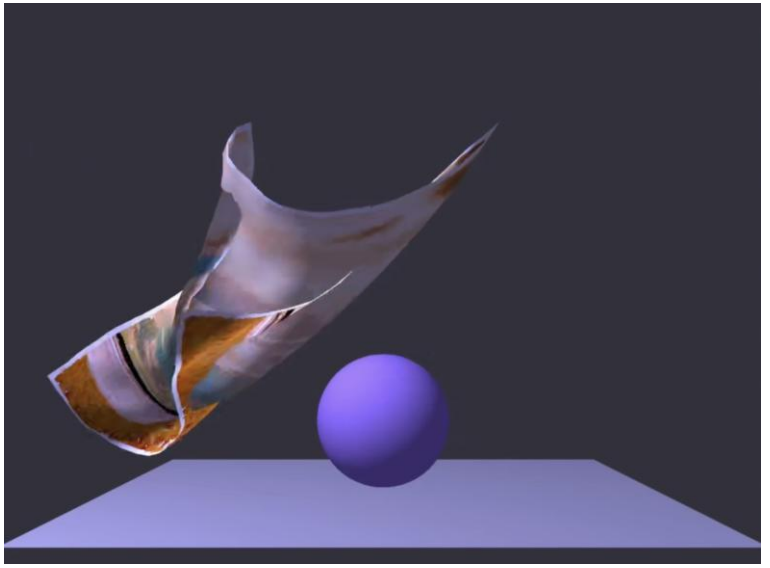


Simulation

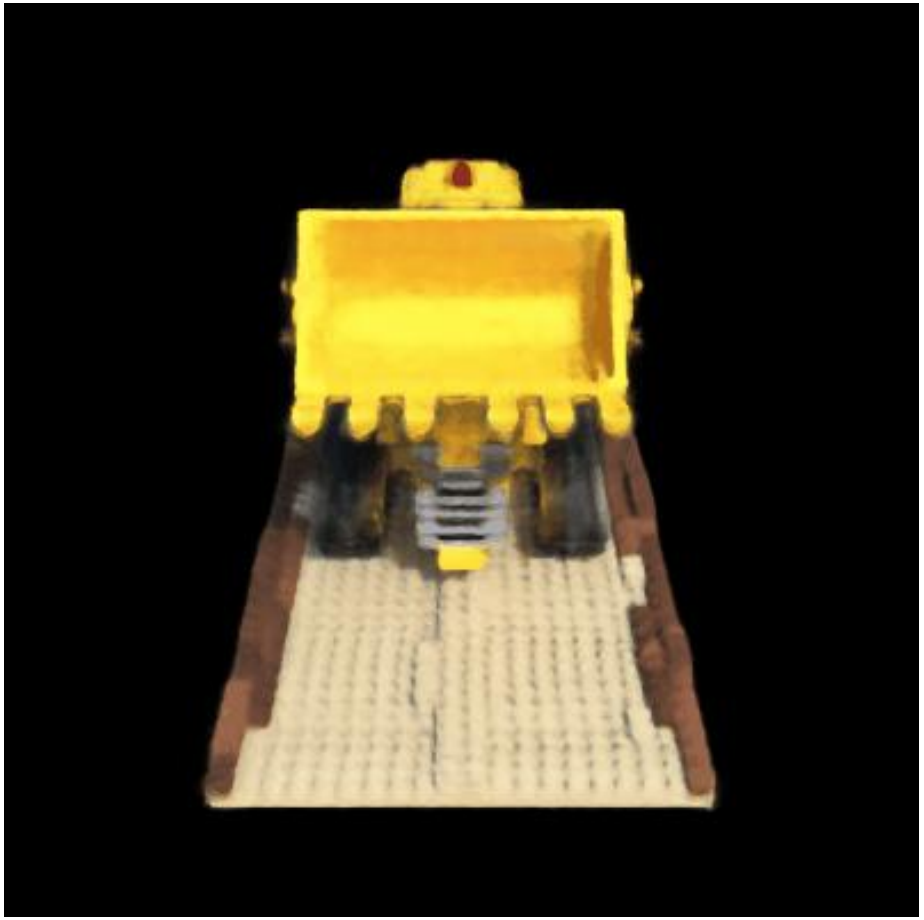








NeRF



谢谢

北京航空航天大學

人工智能研究院