Ray Tracing in Entertainment Industry

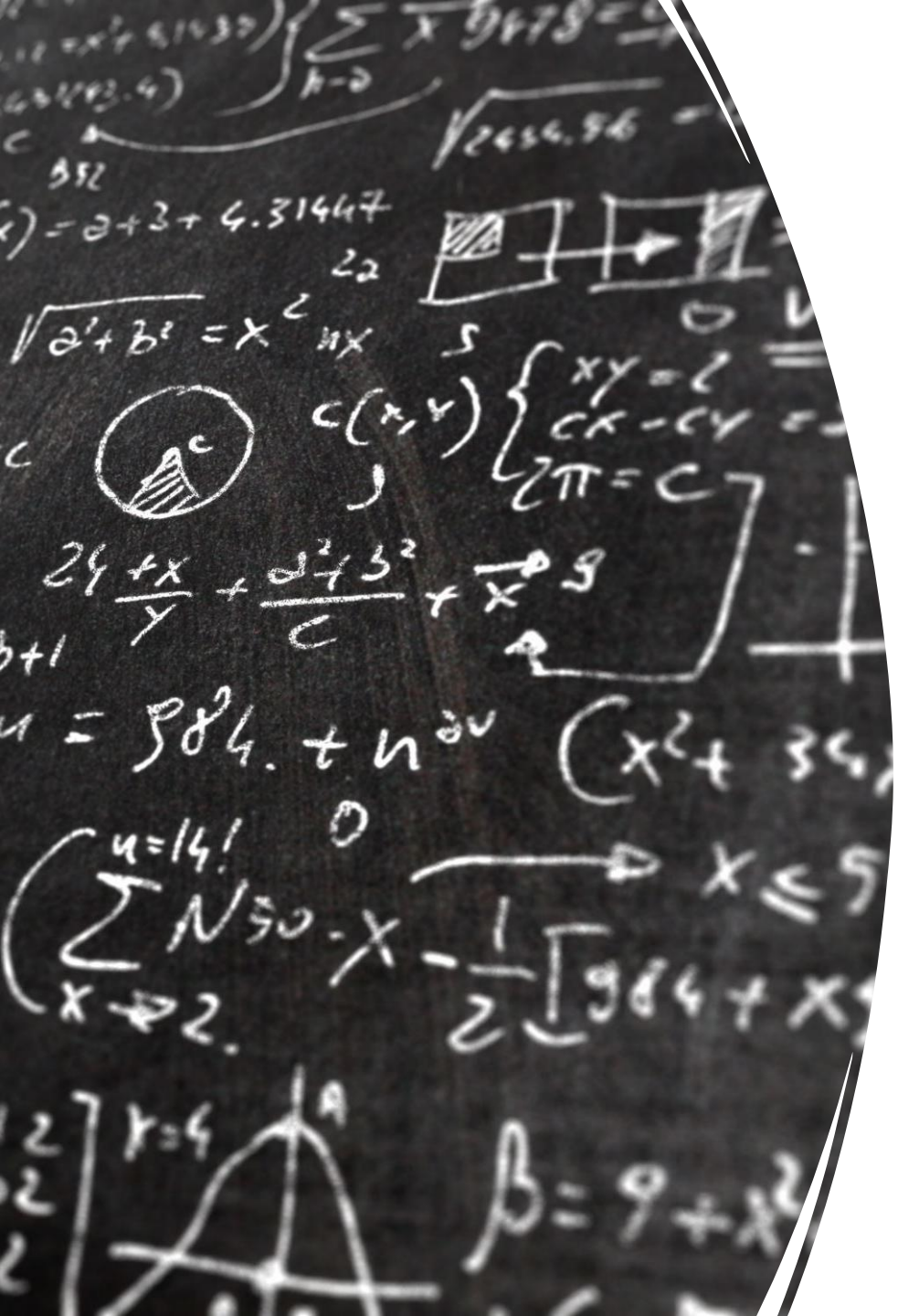
Tanaboon Tongbuasirilai
Dept. Computer Science
Kasetsart University

Week 11
Monte Carlo sampling and Path tracing

Monte Carlo sampling and Path tracing

- **Monte Carlo sampling** is a powerful technique used to randomly sample from a probability distribution. It's like **randomly throwing darts** at a target, where the target represents the probability distribution, and the landing points of the darts represent the samples you draw. By analyzing the distribution of these samples, you can gain valuable insights into the underlying probability distribution.

- **Path tracing** is a rendering technique in computer graphics that produces highly realistic images by simulating the physical behavior of light. It's **a type of Monte Carlo method**, meaning it **uses random sampling** to approximate the solution to a complex problem.



Monte Carlo method

- It can be used to solved
  - Probabilistic problems
    - Particle transport
    - Communication systems
    - Population studies
  - Deterministic problems
    - The evaluation of integrals
    - The systems of algebraic equations
    - Partial differential equations

https://web.tecnico.ulisboa.pt/~mcasquilho/acad/theo/simul/Vujic.pdf

Examples

- **Games and Recreation:**
  - Game strategy evaluation: Analyzing game strategies and potential outcomes in games like chess or Go.
  - Casino simulations: Modeling casino games to estimate odds and payouts.

- **Financial Modeling:**
  - Asset pricing and risk assessment: Modeling the behavior of financial markets and assets to estimate prices, risks, and potential returns.
  - Option pricing: Determining the fair value of options contracts, considering underlying asset price volatility and time to expiration.

Estimating π with randomness

```python
# Estimating Pi with Monte Carlo method
import numpy as np

N = 1000000
inside_circle = 0

for i in range(0, N):
    x = np.random.uniform(-1, 1)
    y = np.random.uniform(-1, 1)
    if x*x + y*y < 1:
        inside_circle = inside_circle + 1

    if i % 10000 == 0:
        print('Estimate of Pi at {} = {}'.format(i, 4 * inside_circle/ N))


print('Estimate of Pi = {}'.format(4 * inside_circle/ N))
```

4 times the ratio of (circle/square) = π

1

$$\frac{\pi r^2}{(2r)^2} = \frac{\pi}{4}$$

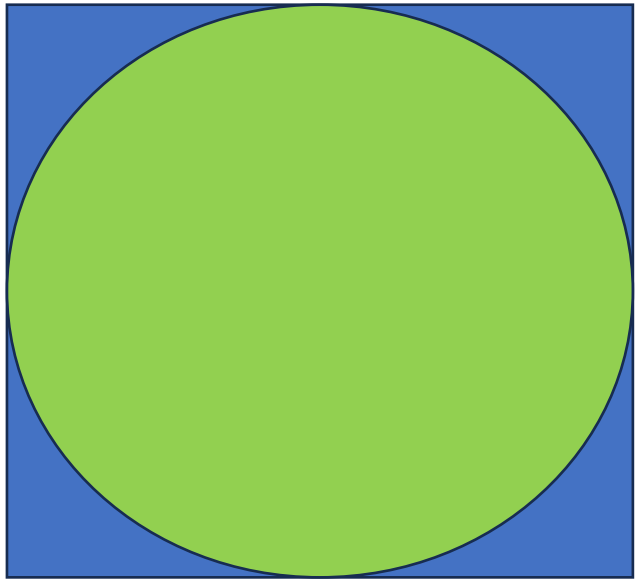
2

$$\frac{\text{area}(circle)}{\text{area}(square)} = \frac{\pi}{4}$$

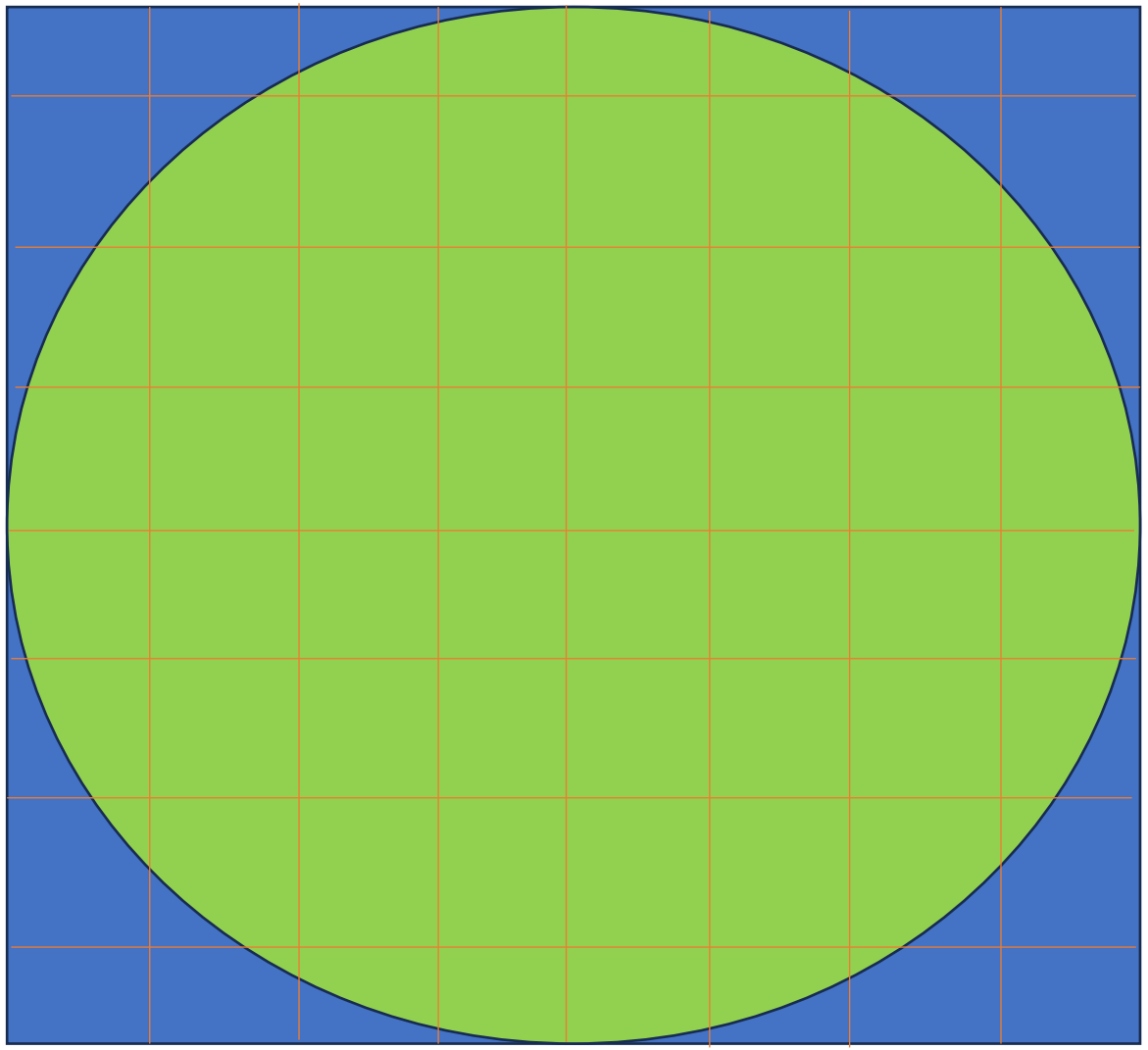
3

Drawing samples that fall inside the circle = the ratio of (circle/square).

This is a Monte Carlo method.



Stratified sampling



```
Estimate of Pi = 3.145268
Estimate of stratified Pi = 3.141372
```

Monte Carlo integration

- Monte Carlo integration is a method for using random sampling to estimate the values of integrals.

- Monte Carlo is that one only needs the ability to evaluate an integrand $f(x)$ at arbitrary points in the domain in order to estimate the value of its integral $\int f(x)dx$.

- Many of the integrals that arise in rendering are difficult or impossible to evaluate directly.

https://pbr-book.org/3ed-2018/Monte_Carlo_Integration

Solving an Integral equation

$$I = \int_0^2 x^2 dx$$

Closed form solution

$$I = \frac{1}{3}x^3\Big|_0^2$$

$$I = \frac{1}{3}(2^3 - 0^3)$$

$$I = \frac{8}{3}$$

Monte Carlo solution

$$E[f(x)|a \leq x \leq b] = \frac{1}{b-a} \cdot \text{area}(f(x), a, b)$$

$$\text{average}(x^2, 0, 2) = \frac{1}{2-0} \cdot \text{area}(x^2, 0, 2)$$

$$\text{average}(x^2, 0, 2) = \frac{1}{2-0} \cdot I$$

$$I = 2 \cdot \text{average}(x^2, 0, 2)$$

Integral estimation with a Monte Carlo method

This function can be arbitrary function.

integral_estimate.py > ...

```python
import numpy as np

N = 10000
a = 0
b = 2.0
sum = 0.0

def func(x):
    return x*x

for i in range(0, N):
    x = np.random.uniform(a, b)
    sum = sum + func(x)

print('I = {}'.format((b-a)*(sum/N)))
```

Integrator ?

- No analytical solution.
- Turning the problem into a numerical method.
- Monte Carlo method is applied.

$$L_o(p, \omega_o) = \int_{S^2} f(p, \omega_o, \omega_i) L_i(p, \omega_i) |\cos \theta_i| \, d\omega_i$$

We would like to solve this integral.

Probability review

Random variable X

- Discrete
- Continuous

Probability Density Function (PDF), p(x)

- Discrete random variable → discrete function
- Continuous random variable → continuous function

Cumulative Distribution Function (CDF), P(x)

$$P(x) = Pr\{X \leq x\}$$

$$P(x \in [a, b]) = \int_a^b p(x)\,\mathrm{d}x$$

Expected value and Variance

- The **expected value** of a function is defined as the average value of the function over some distribution of values over its domain.

$$E_p[f(x)] = \int_D f(x)\, p(x)\, \mathrm{d}x$$

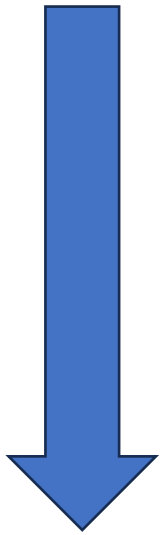
$$E[\cos x] = \int_0^{\pi} \frac{\cos x}{\pi}\, \mathrm{d}x = \frac{1}{\pi}(\sin \pi - \sin 0) = 0$$

- The **variance** of a function is the expected squared deviation of the function from its expected value.

$$V[f(x)] = E\left[(f(x) - E[f(x)])^2\right]$$

Properties of the expected value and variance

$$V[f(x)] = E\left[(f(x) - E[f(x)])^2\right]$$



$$V[f(x)] = E\left[(f(x))^2\right] - E[f(x)]^2$$

$$E[af(x)] = aE[f(x)]$$

$$E\left[\sum_i f(X_i)\right] = \sum_i E[f(X_i)]$$

$$V[af(x)] = a^2 V[f(x)].$$

$$\sum_i V[f(X_i)] = V\left[\sum_i f(X_i)\right]$$

Derivation

$$Var(X) = E[(X - E(X))^2]$$

$$\therefore Var(X) = E[(X - \mu)^2] = E[X^2 - 2\mu X + \mu^2]$$

$$\therefore Var(X) = E[X^2] - E[2\mu X] + E[\mu^2]$$

$$\therefore Var(X) = E[X^2] - 2\mu E[X] + \mu^2$$

$$\therefore Var(X) = E[X^2] - 2\mu \cdot \mu + \mu^2$$

$$\therefore Var(X) = E[X^2] - \mu^2 = E[X^2] - (E[X])^2$$

Estimator of the integral

Previous slides we have this estimator.

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

Proof of the estimator

$$E[F_N] = E\left[\frac{b-a}{N} \sum_{i=1}^{N} f(X_i)\right]$$

$$= \frac{b-a}{N} \sum_{i=1}^{N} E\left[f(X_i)\right]$$

$$= \frac{b-a}{N} \sum_{i=1}^{N} \int_a^b f(x)\, p(x)\, \mathrm{d}x$$

$$= \frac{1}{N} \sum_{i=1}^{N} \int_a^b f(x)\, \mathrm{d}x$$

$$= \int_a^b f(x)\, \mathrm{d}x.$$

Uniform PDF

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

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

The estimator of the integral for arbitrary PDF

Multiple integral and its estimator

$$\int_{z_0}^{z_1} \int_{y_0}^{y_1} \int_{x_0}^{x_1} f(x, y, z)\, \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}z$$

Random sample

$$X_i = (x_i, y_i, z_i)$$

Uniform joint PDF

$$\frac{1}{(x_1 - x_0)} \frac{1}{(y_1 - y_0)} \frac{1}{(z_1 - z_0)}$$

The estimator

$$\frac{(x_1 - x_0)(y_1 - y_0)(z_1 - z_0)}{N} \sum_i f(X_i)$$

How to draw random samples from arbitrary PDF ?

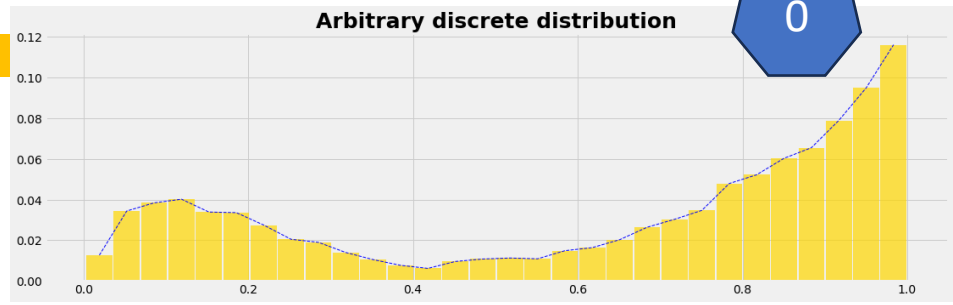
The inversion method

The rejection method

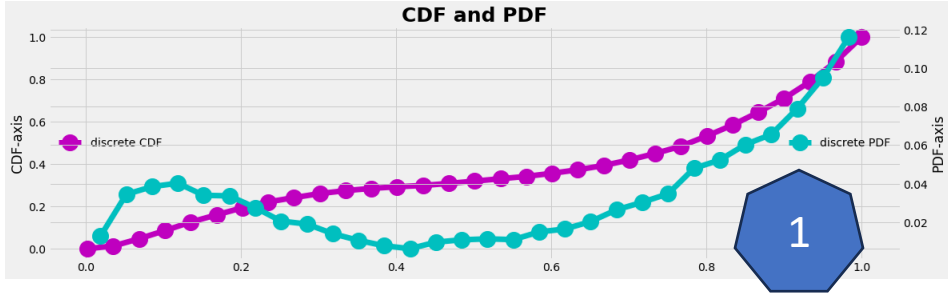
The inversion method

There are 4 steps when do random sampling with the inversion method.

1. Compute the CDF† $P(x) = \int_0^x p(x')\,\mathrm{d}x'$.
2. Compute the inverse $P^{-1}(x)$.
3. Obtain a uniformly distributed random number ξ.
4. Compute $X_i = P^{-1}(\xi)$.

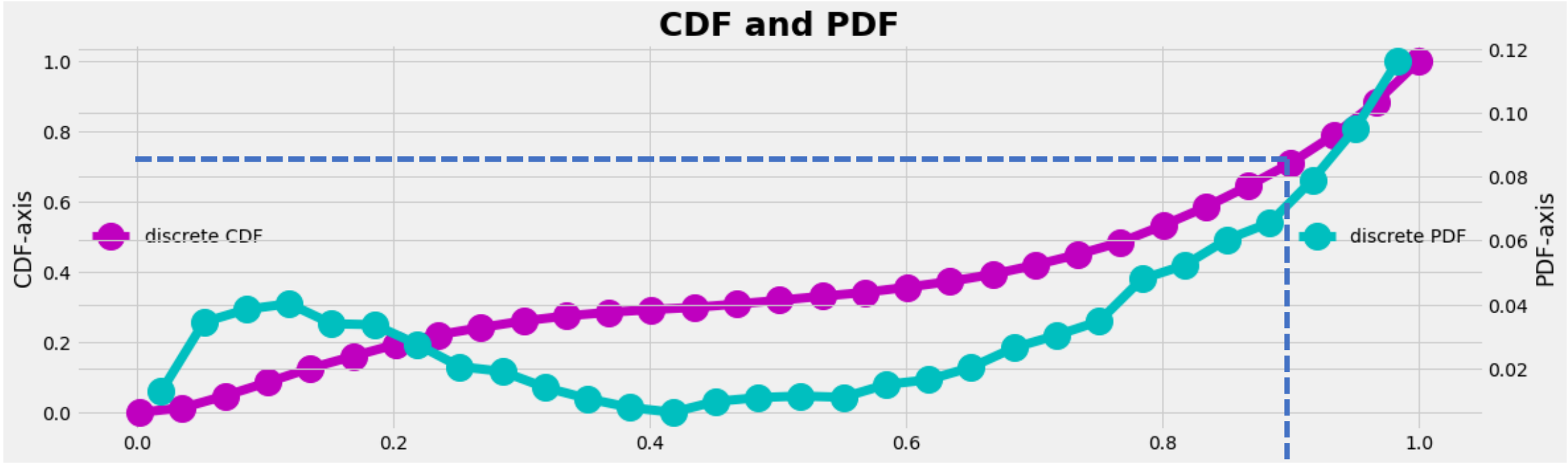


0



1

https://alpynepyano.github.io/healthyNumerics/posts/sampling_arbitrary_distributions_with_python.html



2-3 Sampling with Uniform PDF [0,1]

4 Obtain a random sample according to the arbitrary PDF.

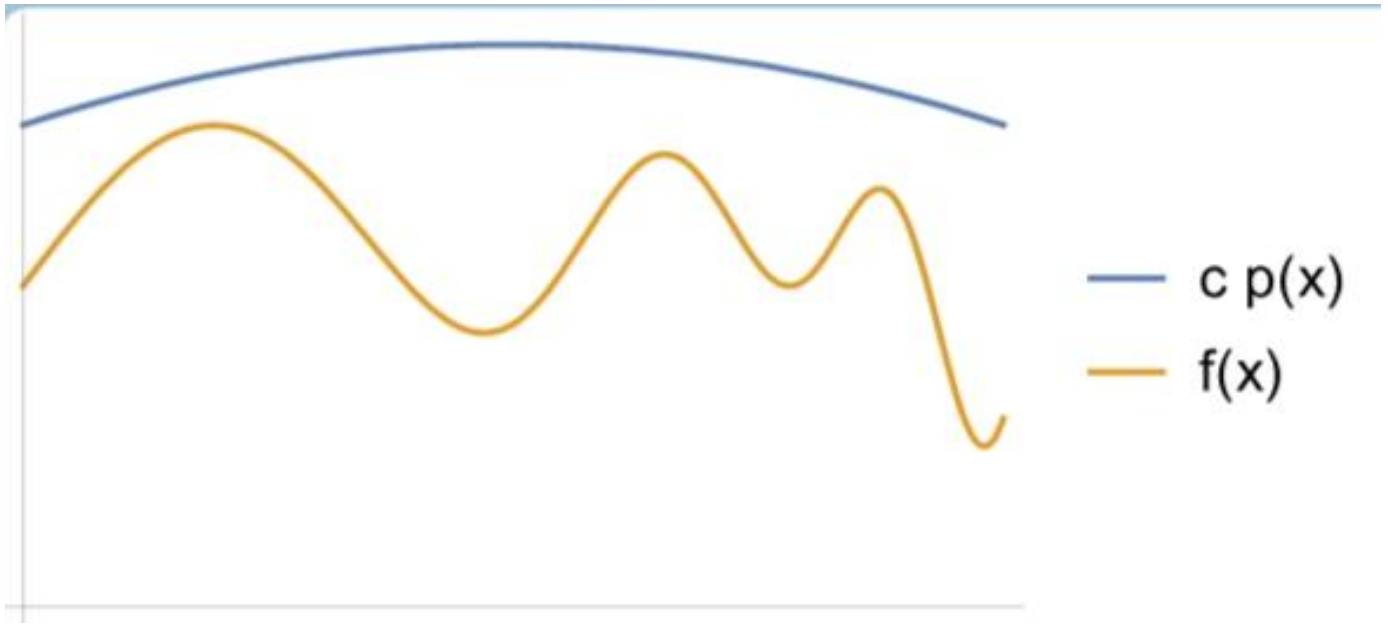
The rejection method

loop forever:

sample X from p's distribution

if $\xi < f(X)/(c\,p(X))$ then

return X



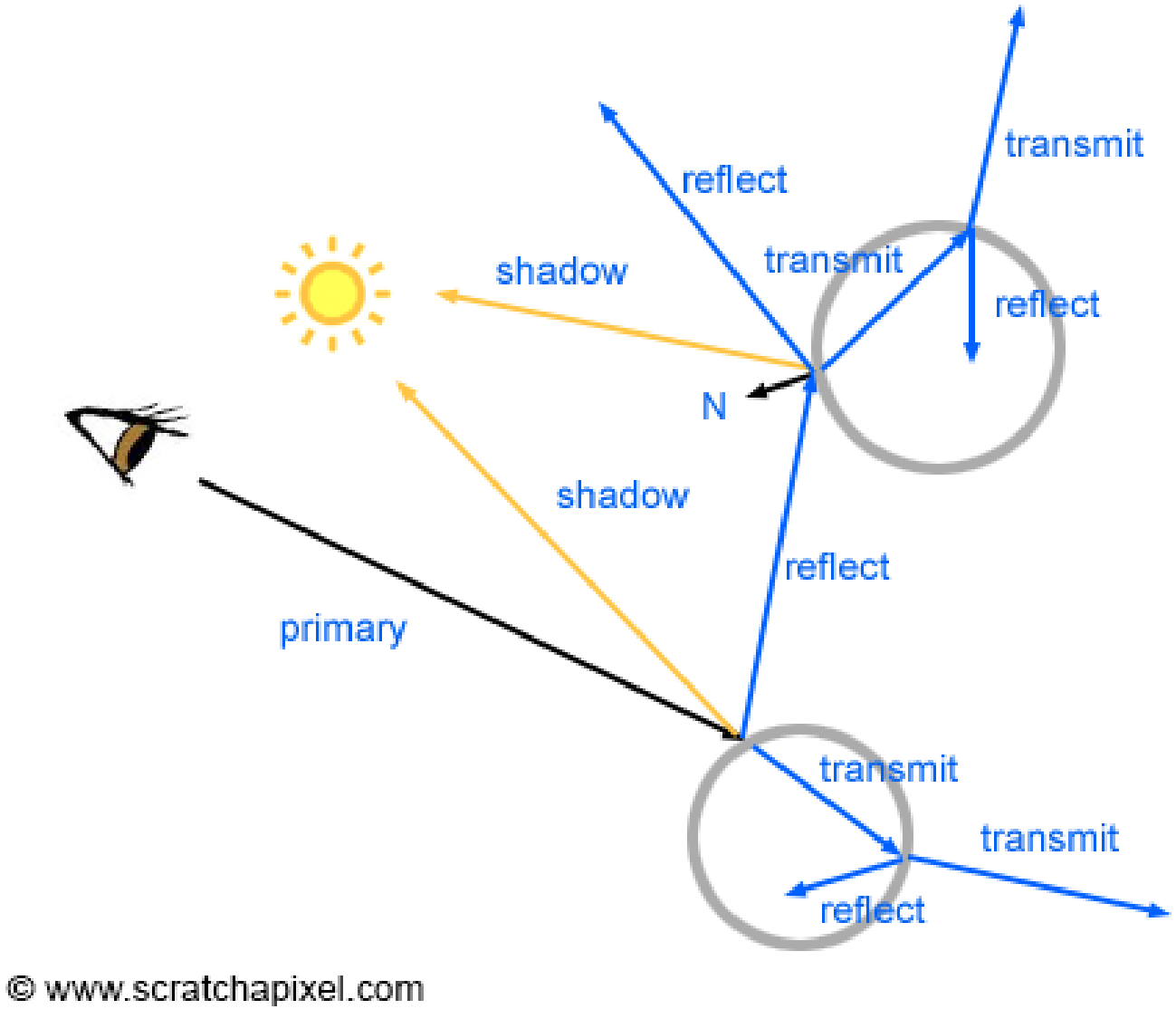
Path tracing

By integrating the Monte Carlo method for solving the rendering equation, the renderer becomes a path tracer.

This means Path Tracing involves in using random sampling techniques and probability density functions to obtain the final render images.

Whitted Ray Tracing

Only 1 ray per pixel !



www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-overview/light-transport-ray-tracing-whitted.html

Path tracing Vs Ray tracing [User perspective]

Ray Tracing vs. Path Tracing: Unraveling the Differences

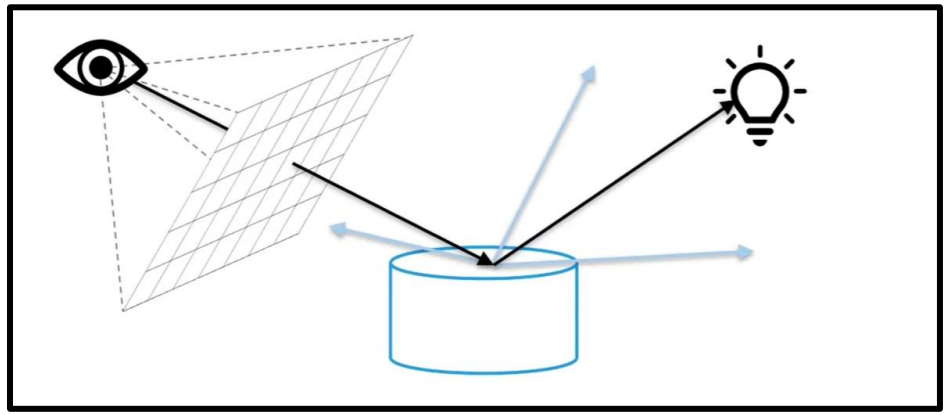
https://www.youtube.com/watch?v=3YIXLLrm7fo

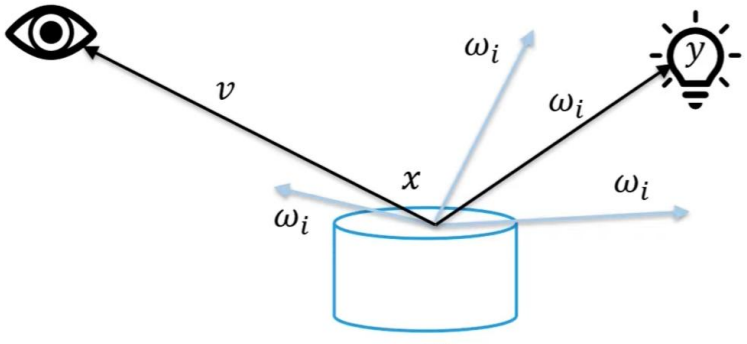
Path Tracing basics

Rendering Lecture 04 - Path Tracing Basics (youtube.com)

https://www.youtube.com/watch?v=w36xgaGQYAY

Path tracing - Direct lighting





$$L(x \to v) = E_x + \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{\pi}E_y \cos(\theta_{\omega_i})\frac{1}{p(\omega_i)}\right)$$

Li

N dot L

Divided by a PDF

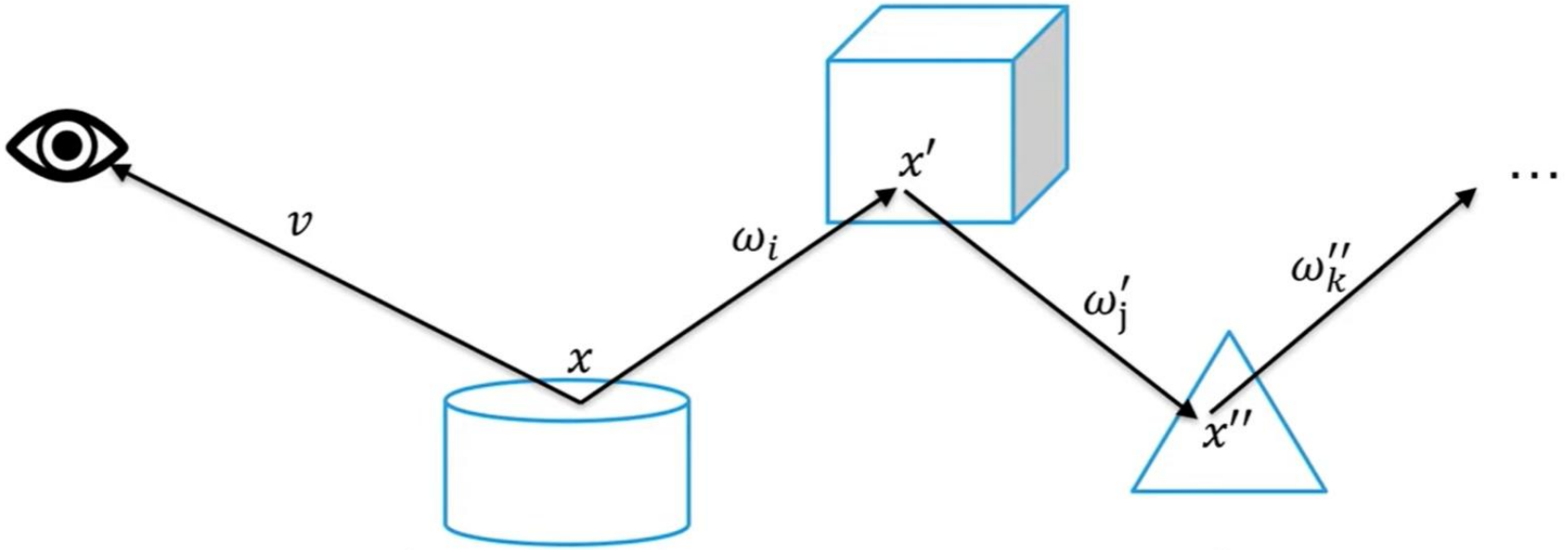
Move E_x term inside the summation

```
for (i = 0; i < N; i++)
      v_inv = camera.gen_ray(px, py)
      x = scene.trace(v_inv)
      f = x.emit
      omega_i, prob_i = hemisphere_uniform_world(x)
      r = make_ray(x, omega_i)
      y = scene.trace(r)
      f = 1/pi * y.emit * dot(x.normal, omega_i) / prob_i
      pixel_color += f
pixel_color /= N
```

$$L(x \to v) = \frac{1}{N}\sum_{i=1}^{N}\left(E_x + \frac{1}{\pi}E_y \cos(\theta_{\omega_i})\frac{1}{p(\omega_i)}\right)$$

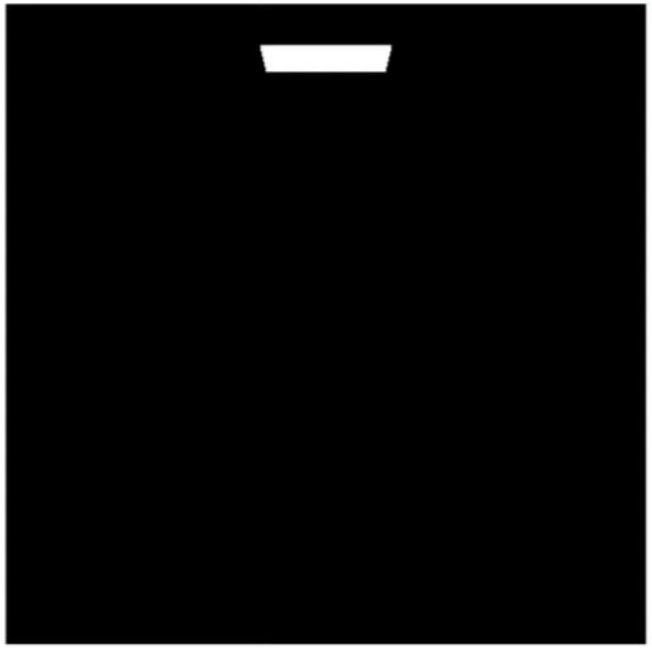
Path tracing – Indirect lighting

- With Monte Carlo integration, we might replace all $\int$ with $\sum$



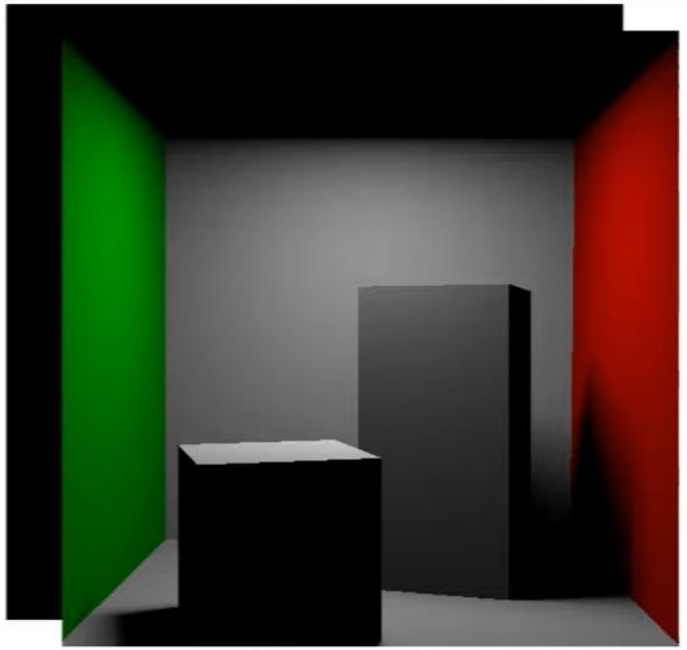
$$L(x \to v) = E_x + \frac{1}{N}\sum_{i=1}^{N} f_r \left(E_{x'} + \frac{1}{N}\sum_{j=1}^{N} f'_r \ \ldots \cos\left(\theta_{\omega'_j}\right) \frac{1}{p(\omega'_j)} \right) \cos(\theta_{\omega_i}) \frac{1}{p(\omega_i)}$$

Reconsidering Sample Distribution



$$L(x \to v) = E_x$$
$$+ \int_{\Omega} f_r \, E_{x'} \cos(\theta_\omega) \, d\omega$$
$$+ \int_{\Omega} f_r \int_{\Omega'} f_r' \, E_{x''} \cos(\theta_{\omega'}) \cos(\theta_\omega) \, d\omega' d\omega$$
$$+ \int_{\Omega} f_r \int_{\Omega'} f_r' \int_{\Omega''} f_r'' \, E_{x'''} \cos(\theta_{\omega''}) \cos(\theta_{\omega'}) \cos(\theta_\omega) \, d\omega'' d\omega' d\omega$$
$$+ \ \ldots$$

Reconsidering Sample Distribution



$$L(x \to v) = E_x + \int_\Omega f_r \, E_{x'} \cos(\theta_\omega) \, d\omega + \int_\Omega f_r \int_{\Omega'} f_r' \, E_{x''} \cos(\theta_{\omega'}) \cos(\theta_\omega) \, d\omega' d\omega + \int_\Omega f_r \int_{\Omega'} f_r' \int_{\Omega''} f_r'' \, E_{x'''} \cos(\theta_{\omega''}) \cos(\theta_{\omega'}) \cos(\theta_\omega) \, d\omega'' d\omega' d\omega + \ldots$$

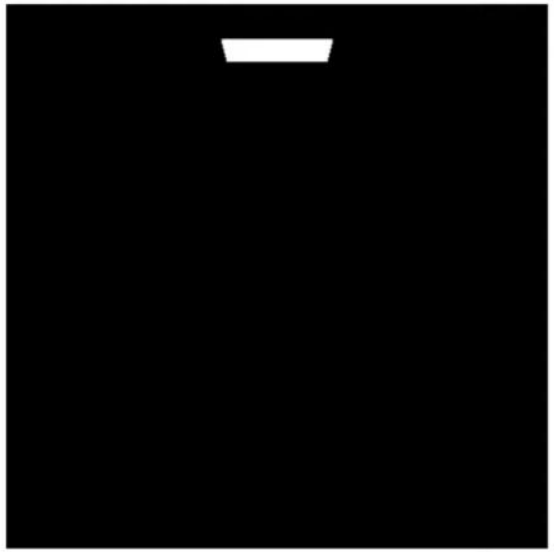
Reconsidering Sample Distribution

$$L(x \to v) = E_x + \int_\Omega f_r \, E_{x'} \cos(\theta_\omega) \, d\omega + \int_\Omega f_r \int_{\Omega'} f_r' \, E_{x''} \cos(\theta_{\omega'}) \cos(\theta_\omega) \, d\omega' d\omega + \int_\Omega f_r \int_{\Omega'} f_r' \int_{\Omega''} f_r'' \, E_{x'''} \cos(\theta_{\omega''}) \cos(\theta_{\omega'}) \cos(\theta_\omega) \, d\omega'' d\omega' d\omega + \ldots$$

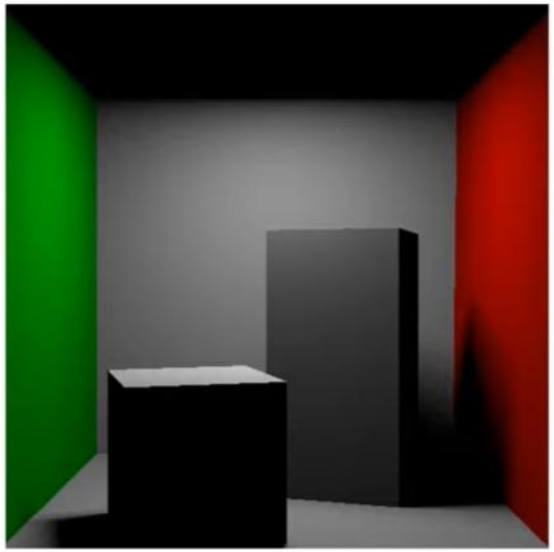
Reconsidering Sample Distribution

$$L(x \to v) = E_x + \int_\Omega f_r \, E_{x'} \cos(\theta_\omega) \, d\omega + \int_\Omega f_r \int_{\Omega'} f_r' \, E_{x''} \cos(\theta_{\omega'}) \cos(\theta_\omega) \, d\omega' d\omega + \int_\Omega f_r \int_{\Omega'} f_r' \int_{\Omega''} f_r'' \, E_{x'''} \cos(\theta_{\omega''}) \cos(\theta_{\omega'}) \cos(\theta_\omega) \, d\omega'' d\omega' d\omega + \ldots$$

- The return-on-investment is not that great!



1 sample



N samples

N^2 samples

N^3 samples

Let's implement this with recursion!

```
v_inv = camera.gen_ray(px, py)
pixel_color = Li(v_inv)

function Li(v_inv)
      x = scene.trace(v_inv)
      f = 0
      for (i = 0; i < N; i++)
            omega_i, prob_i = hemisphere_uniform_world(x)
            ray = make_ray(x, omega_i)
            f += x.alb/pi * Li(ray) * dot(x.normal, omega_i) / prob_i
      return x.emit + f/N
```

$$L(x \to v) = E_x + \frac{1}{N}\sum_{i=1}^{N} f_r \; L(x \leftarrow \omega_i) \; \cos(\theta_{\omega_i}) \frac{1}{p(\omega_i)}$$

Reconsidering Sample Distribution

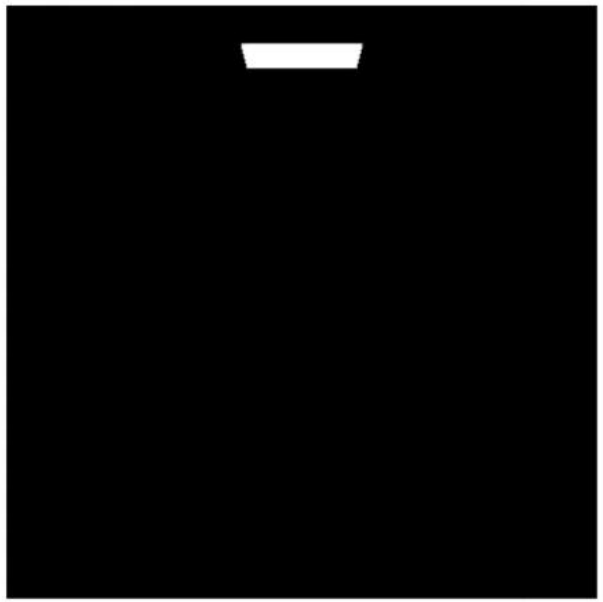
- We have seen a different way of writing these results last time

- The path integral form used a single integral for each bounce!

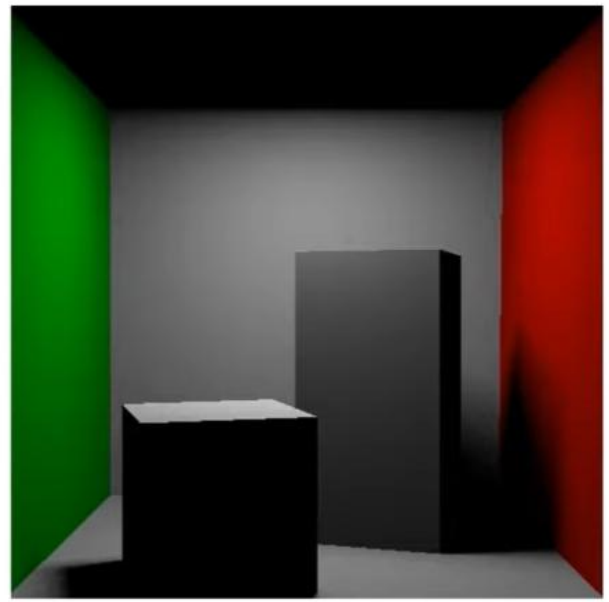
$$\int_{\Omega_0} \mathrm{f_j}(\bar{x})\,\mathrm{d}\mu(\bar{x}) + \int_{\Omega_1} \mathrm{f_j}(\bar{x})\,\mathrm{d}\mu(\bar{x}) + \ldots + \int_{\Omega_\infty} \mathrm{f_j}(\bar{x})\,\mathrm{d}\mu(\bar{x})$$

$$\begin{aligned} L(x \to v) = E_x &\\ &+ \int_{\Omega_1} f_r\, E_{x'} \cos(\theta_\omega)\, d\mu(\bar{x}) \\ &+ \int_{\Omega_2} f_r\, f_r' E_{x''} \cos(\theta_{\omega'}) \cos(\theta_\omega)\, d\mu(\bar{x}) \\ &+ \int_{\Omega_3} \underbrace{f_r f_r' f_r''\, E_{x'''} \cos(\theta_{\omega''}) \cos(\theta_{\omega'}) \cos(\theta_\omega)}_{\mathrm{f_j}(\bar{x})}\, d\mu(\bar{x}) \\ &+ \ldots \end{aligned}$$

- No more exponential sample growth!



N samples



N samples

N samples

N samples

Reconsidering Sample Distribution

- We can again pull the sum to the front…

$$L(x \to v) = \frac{1}{N}\sum_{i=1}^{N}(E_x$$
$$+ f_r E_{x'} \cos(\theta_\omega) \frac{1}{p(\omega)}$$
$$+ f_r f_r' E_{x''} \cos(\theta_{\omega'}) \cos(\theta_\omega) \frac{1}{p(\omega)p(\omega')}$$
$$+ f_r f_r' f_r'' E_{x'''} cos(\theta_{\omega''}) \cos(\theta_{\omega'}) \cos(\theta_\omega) \frac{1}{p(\omega)p(\omega')p(\omega'')}$$
$$+ \ldots)$$

Reconsidering Sample Distribution

- …and rewrite to highlight the original recursion

$$L(x \to v) = \frac{1}{N}\sum_{i=1}^{N}\left(E_x + f_r\left(E_{x'} + f_r'(\ \ldots\ )\cos\theta_{\omega'}\frac{1}{p(\omega')}\right)\cos(\theta_\omega)\frac{1}{p(\omega)}\right)$$

- We are back to a single sum for integration with recursion!

- This also fits perfectly with our main loop and interface design

Path Tracing with a Single Sum

```
for (i = 0; i < N; i++)
      v_inv = camera.gen_ray(px, py)
      pixel_color += Li(v_inv, 0)
pixel_color /= N
```

$$L(x \to v) = \boxed{\frac{1}{N}\sum_{i=1}^{N}} \left(E_x + f_r\,(\ldots)\cos(\theta_\omega)\frac{1}{p(\omega)} \right)$$

```
function Li(v_inv, D)
      if (D >= NUM_BOUNCES)
            return 0
      x = scene.trace(v_inv)
      f = x.emit
      omega, prob = hemisphere_uniform_world(x)
      r = make_ray(x, omega)
      f += x.alb/pi * Li(r, D+1) * dot(x.normal, omega)/prob
      return f
```

Recall week 09 : Uniform sampling on …

Sphere

$$x = \cos(2\pi r_1) \cdot 2\sqrt{r_2(1-r_2)}$$

$$y = \sin(2\pi r_1) \cdot 2\sqrt{r_2(1-r_2)}$$

$$z = 1 - 2r_2$$

$r1 \in [0,1]$
$r2 \in [0,1]$

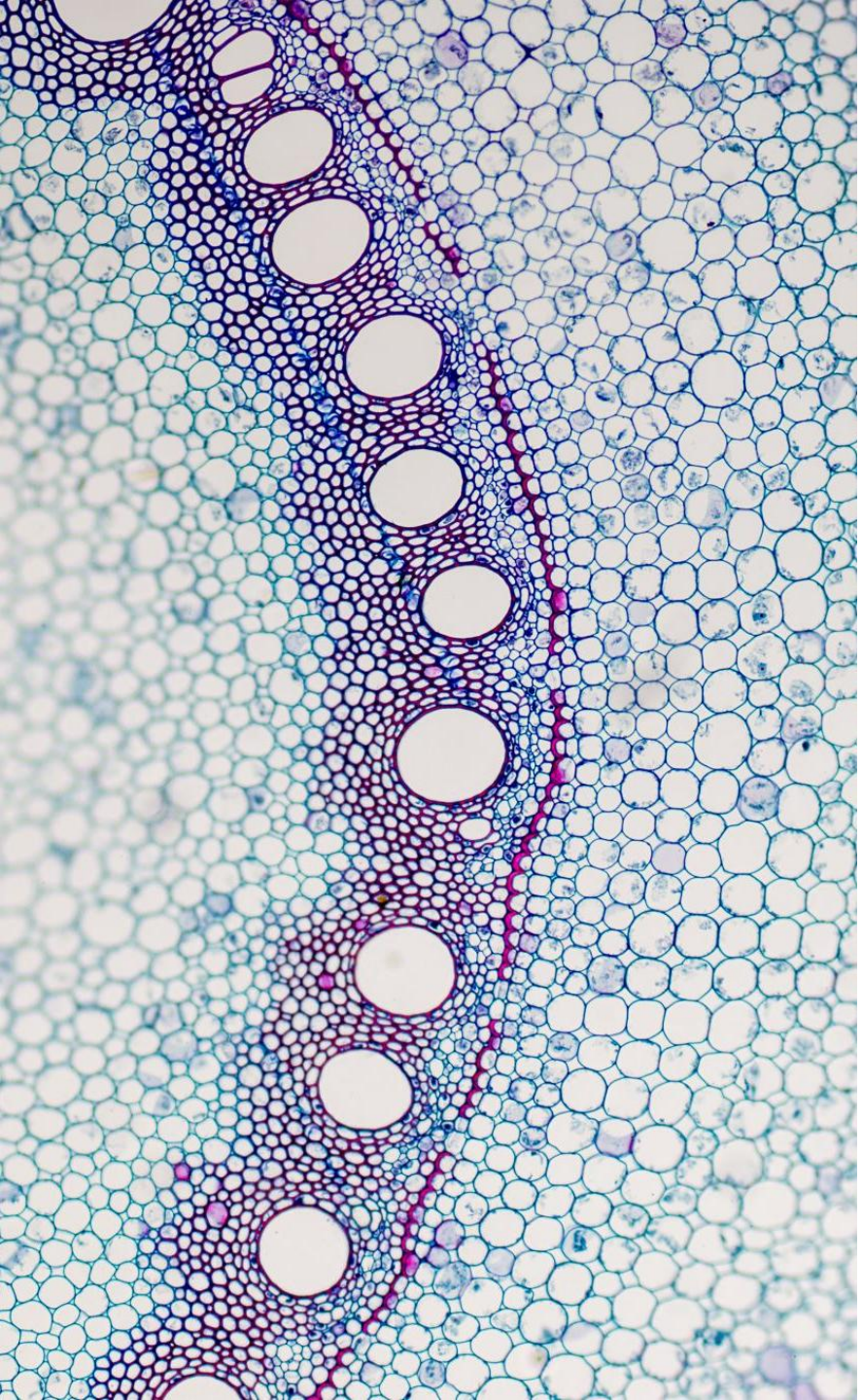
hemisphere

$$z = \cos(\theta) = \sqrt{1-r_2}$$

$$x = \cos(\phi)\sin(\theta) = \cos(2\pi r_1)\sqrt{1-z^2} = \cos(2\pi r_1)\sqrt{r_2}$$

$$y = \sin(\phi)\sin(\theta) = \sin(2\pi r_1)\sqrt{1-z^2} = \sin(2\pi r_1)\sqrt{r_2}$$

$r1 \in [0,1]$
$r2 \in [0,1]$



Uniform sampling on hemisphere

- The probability to derive the sampling on hemisphere is $p(\omega) = \frac{1}{2\pi}$, see the derivation in 13.6.1 PBRT book.
- The uniform sampling on sphere is $p(\omega) = \frac{1}{4\pi}$.
- Note that the probability is subjected to the solid angle ω.

- However, the rendering equation has the cosine weighted term, choosing a PDF close to the integrand will result in faster convergence.
- That means we want to choose $p(\omega) \propto \cos(\theta)$.
- We will talk about this issue in the next week. [Variance reduction]