CS482:
Monte Carlo Ray Tracing:

Sung-Eui Yoon
(윤성의)

http://sglab.kaist.ac.kr/~sungeui/ICG

KAIST

Class Objectives

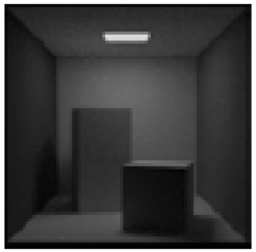
- **Understand a basic structure of Monte Carlo ray tracing**
  - **Russian roulette for its termination**
  - **Path tracing**

Rendering Equation

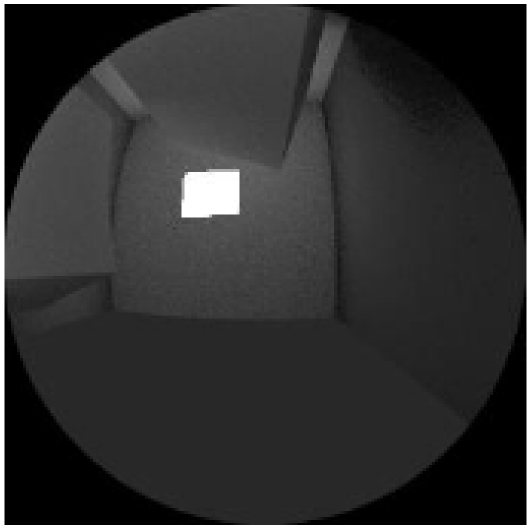
$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_{\Omega_x} f_r(\Psi \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$

function to integrate over all incoming directions over the hemisphere around x

Value we want



$= L_e + \int_{\Omega_x}$



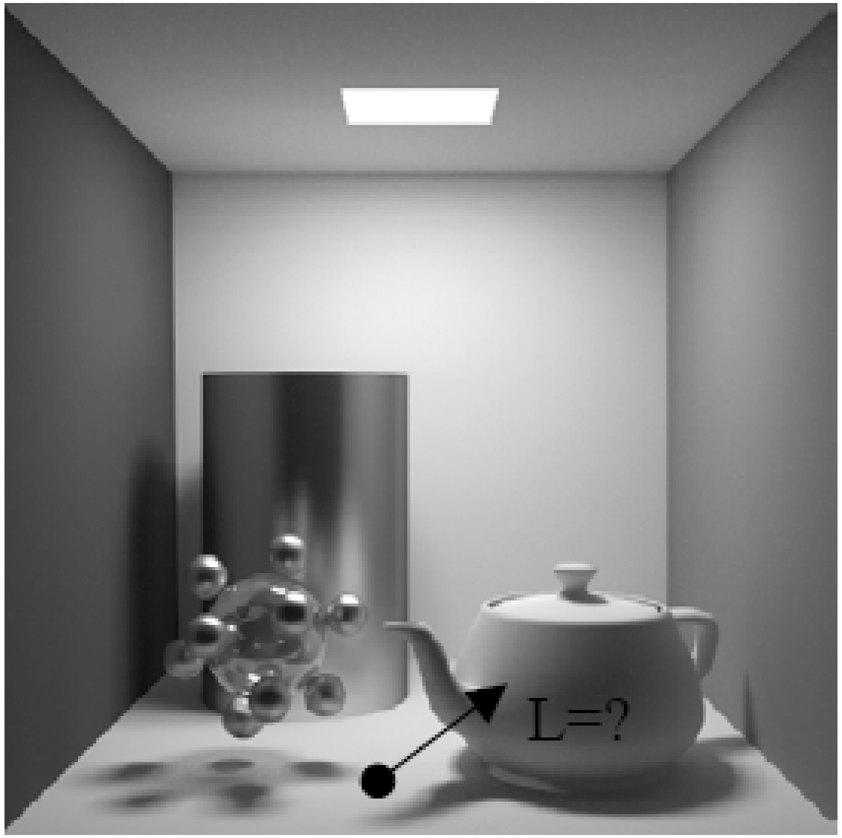
$\cdot f_r \cdot \cos$

How to compute?

$L(x \to \Theta) = ?$

Check for $L_e(x \to \Theta)$

Now add $L_r(x \to \Theta) =$



$$\int_{\Omega_x} f_r(\Psi \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$

How to compute?

- **Use Monte Carlo**

- **Generate random directions on hemisphere Ω_x using pdf $p(\Psi)$**

$$L(x \to \Theta) = \int_{\Omega_x} f_r(\Psi \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$

$$\langle L(x \to \Theta) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(\Psi_i \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi_i) \cdot \cos(\Psi_i, n_x)}{p(\Psi_i)}$$

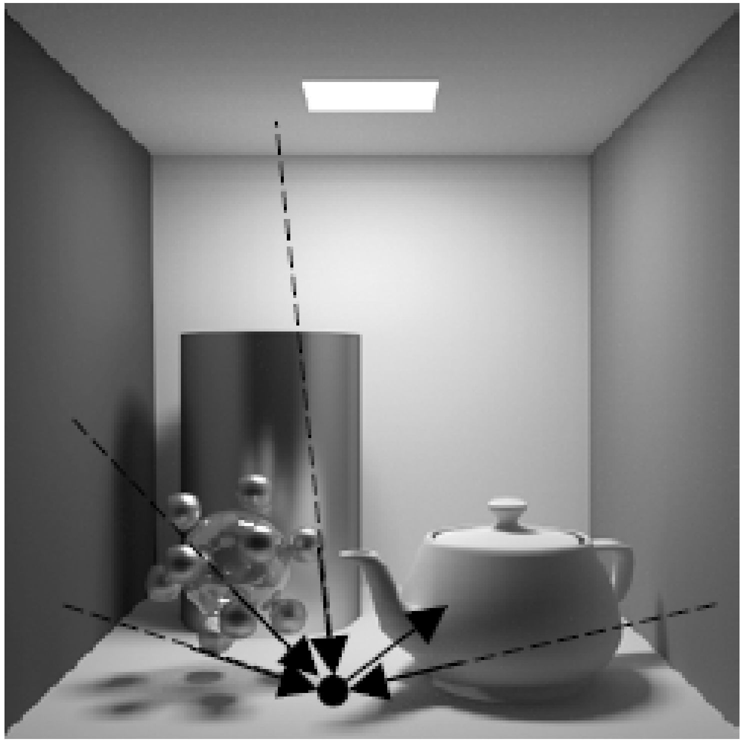
How to compute?

Generate random directions Ψ_i

$$\langle L \rangle = \frac{1}{N}\sum_{i=1}^{N}\frac{f_r(\ldots)\cdot L(x \leftarrow \Psi_i)\cdot \cos(\ldots)}{p(\Psi_i)}$$

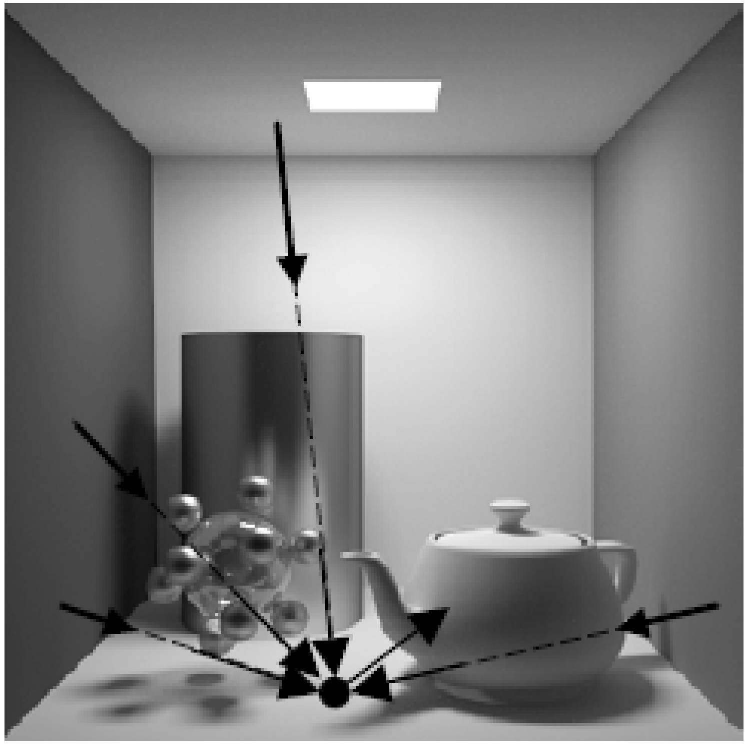
- evaluate brdf
- evaluate cosine term
- evaluate $L(x \leftarrow \Psi_i)$

How to compute?



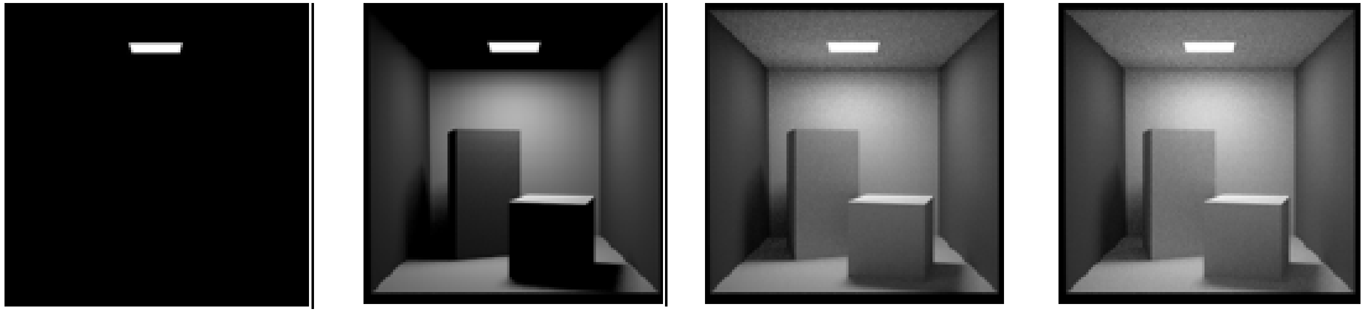
- evaluate $L(x \leftarrow \Psi_i)$?
- Radiance is invariant along straight paths
- $vp(x, \Psi_i)$ = first visible point
- $L(x \leftarrow \Psi_i) = L(vp(x, \Psi_i) \rightarrow \Psi_i)$

How to compute? Recursion ...



- Recursion ….
- Each additional bounce adds one more level of indirect light
- Handles ALL light transport
- "Stochastic Ray Tracing"

When to end recursion?



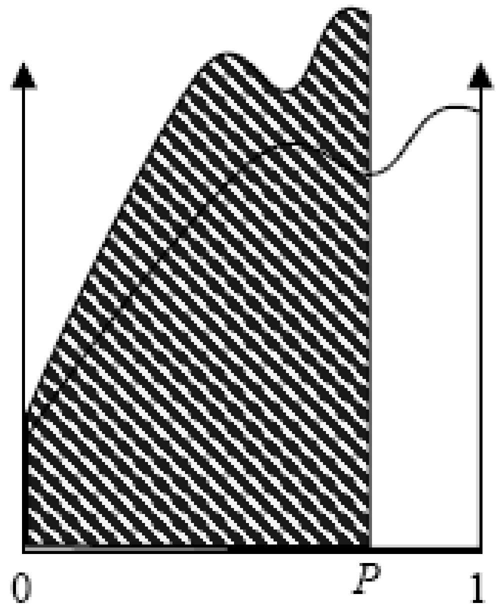
From kavita's slides

- Contributions of further light bounces become less significant
  - Max recursion
  - Some threshold for radiance value
- If we just ignore them, estimators will be biased

Russian Roulette

Integral

$$I = \int_0^1 f(x)dx = \int_0^1 \frac{f(x)}{P} P dx = \int_0^P \frac{f(y/P)}{P} dy$$



Estimator

$$\langle I_{roulette} \rangle = \begin{cases} \frac{f(x_i)}{P} & \textbf{if } x_i \le P, \\ 0 & \textbf{if } x_i > P. \end{cases}$$

Variance $\sigma_{roulette} > \sigma$

Russian Roulette

- **Pick absorption probability, $\alpha = 1 - P$**
  - **Recursion is terminated**

- **1- α, i.e., P, is commonly to be equal to the reflectance of the material of the surface**
  - **Darker surface absorbs more paths**

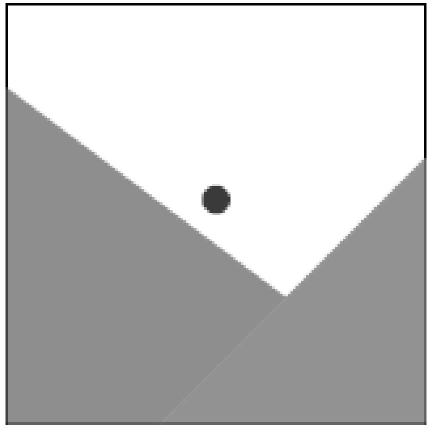
KAIST

Algorithm so far

- Shoot primary rays through each pixel
- Shoot indirect rays, sampled over hemisphere
- Terminate recursion using Russian Roulette

Pixel Anti-Aliasing

- **Compute radiance only at the center of pixel**
  - **Produce jaggies**



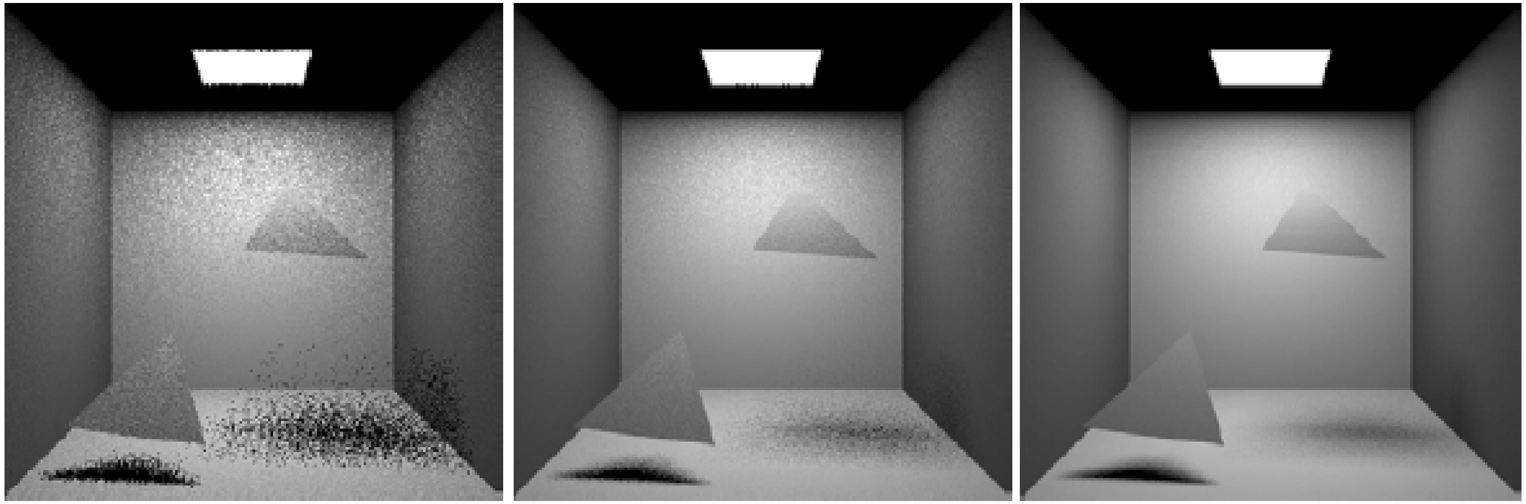
- **We want to evaluate using MC**

- **Simple box filter**
  - **The averaging method**

Stochastic Ray Tracing

- **Parameters**
  - **Num. of starting ray per pixel**
  - **Num. of random rays for each surface point (branching factor)**
- **Path tracing**
  - **Branching factor = 1**

Path Tracing



1 ray / pixel 10 rays / pixel 100 rays / pixel

From kavita's slides

- **Pixel sampling + light source sampling folded into one method**

Algorithm so far

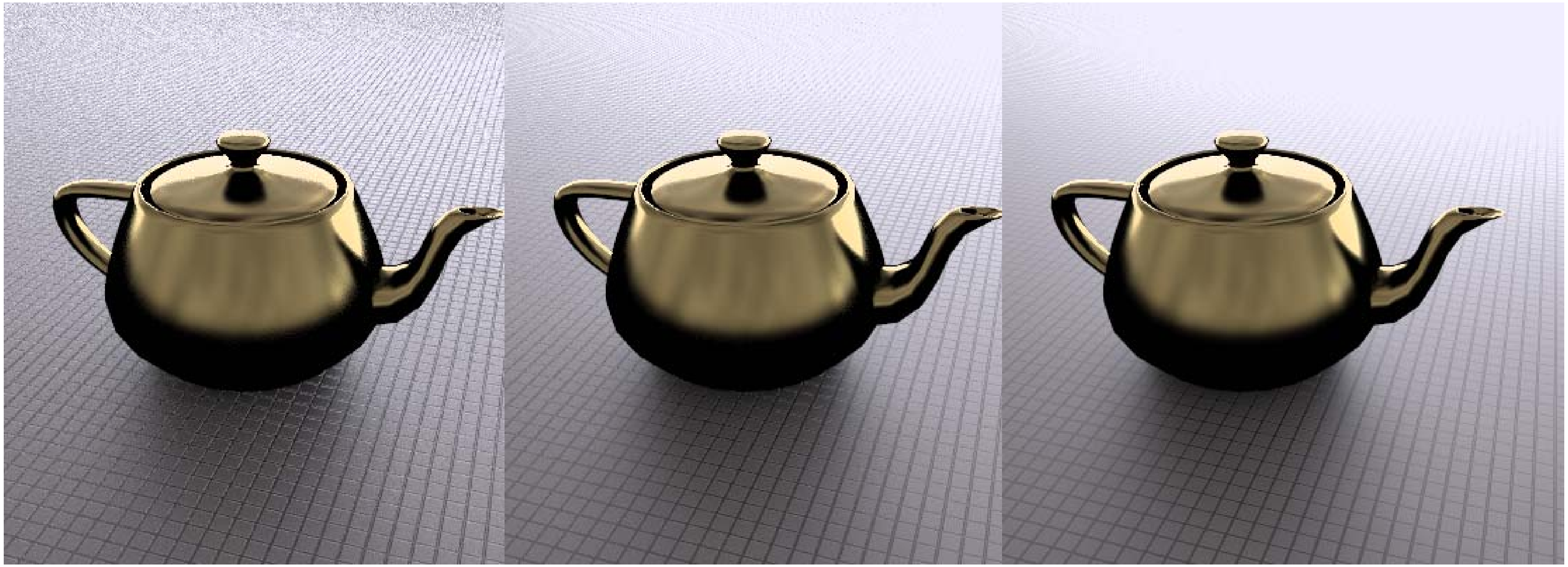
- Shoot primary rays through each pixel
- Shoot indirect rays, sampled over hemisphere
  - Path tracing shoots only 1 indirect ray
- Terminate recursion using Russian Roulette

Performance

- **Want better quality with smaller # of samples**
  - **Fewer samples/better performance**
  - **Quasi Monte Carlo: well-distributed samples**
  - **Adaptive sampling**

KAIST

Some Example

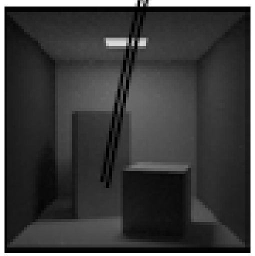


Uniform sampling (64 samples per pixel) | **Adaptive sampling** | **Reference**

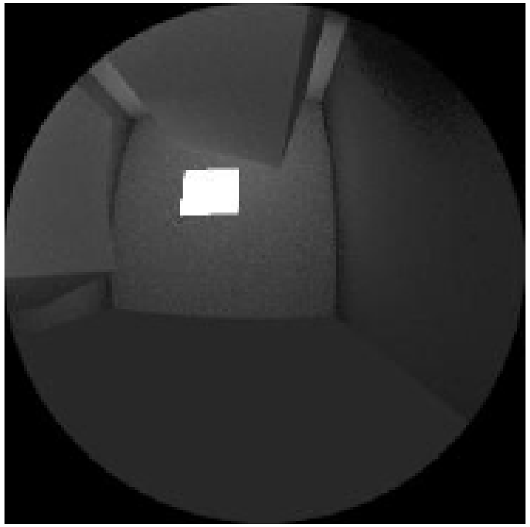
Importance Sampling

$$L(x \to \Theta) = L_e(x \to \Theta) + \int_{\Omega_x} f_r(\Psi \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$

Radiance from light sources + radiance from other surfaces



$$= L_e + \int_{\Omega_x}$$

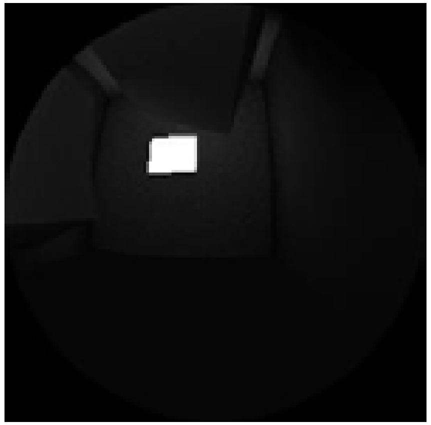


$$\cdot f_r \cdot \cos$$

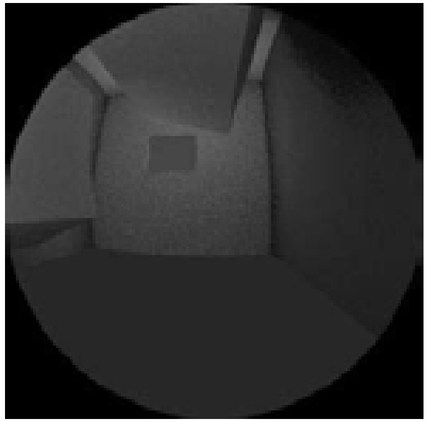
Importance Sampling

$$L(x \rightarrow \Theta) = L_e + L_{direct} + L_{indirect}$$

$$= L_e + \int_{\Omega_x}$$



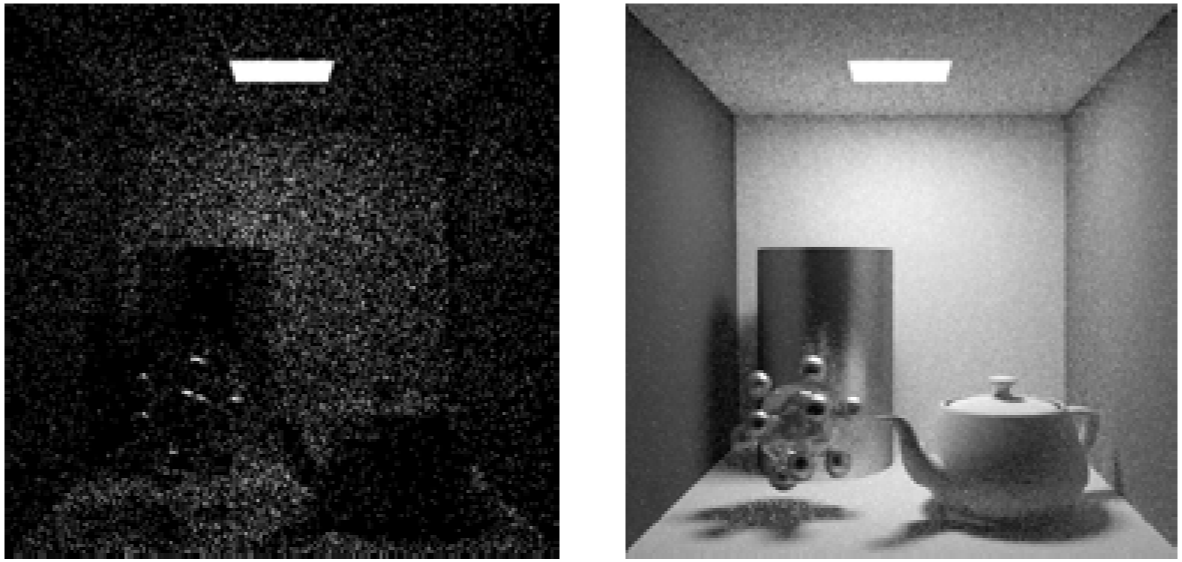
$$\cdot f_r \cdot \cos + \int_{\Omega_x}$$



$$\cdot f_r \cdot \cos$$

- So … sample direct and indirect with separate MC integration

Comparison



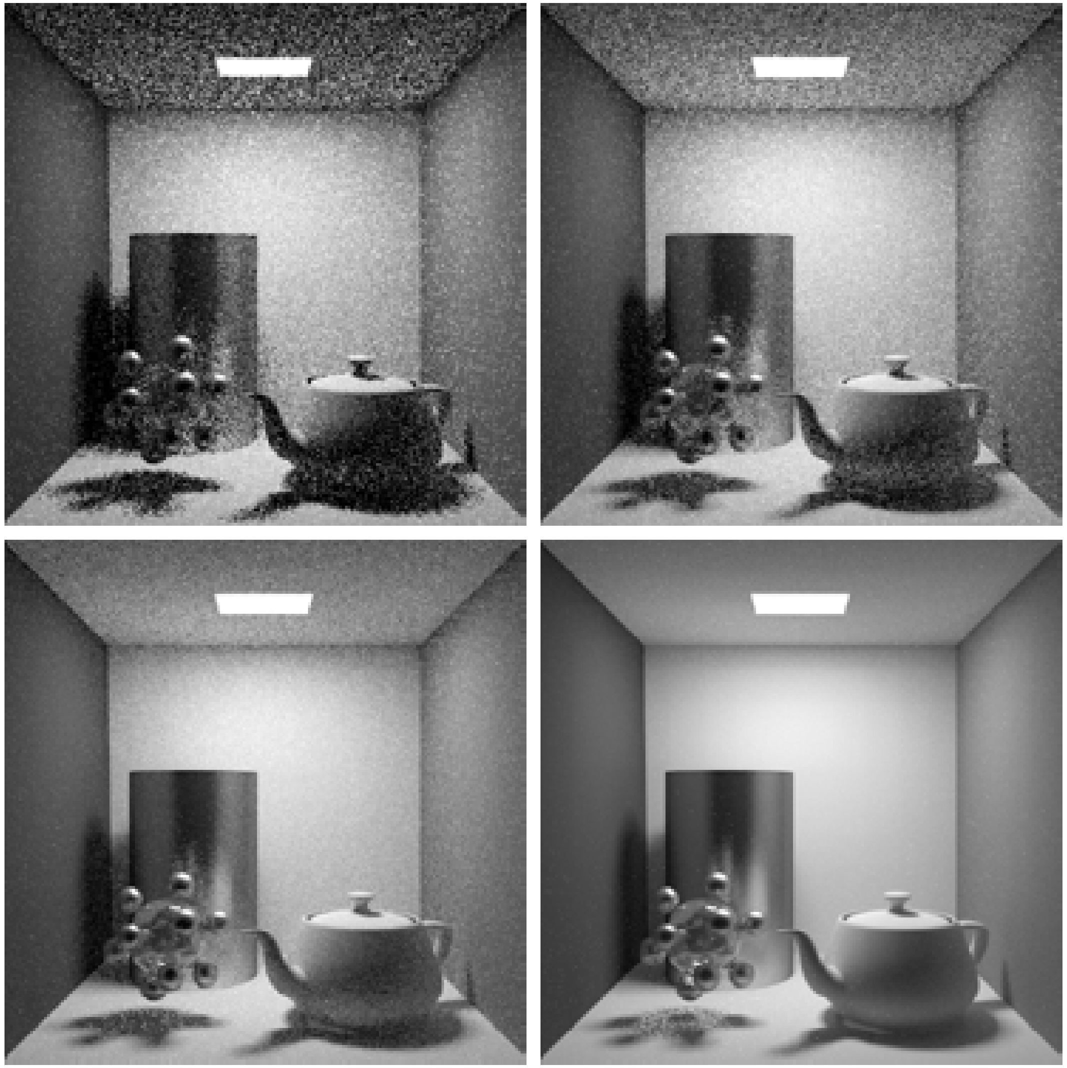
From kavita's slides

- **With and without considering direct illumination**
  - **16 samples / pixel**

Rays per pixel

1 sample/ pixel

4 samples/ pixel



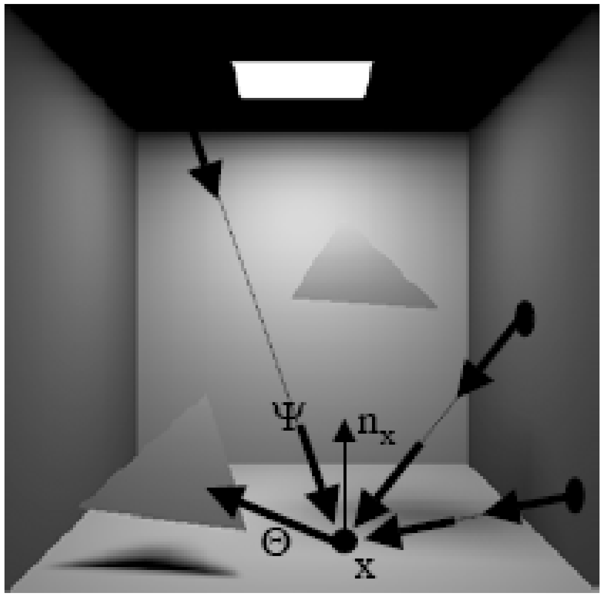
16 samples/ pixel

256 samples/ pixel

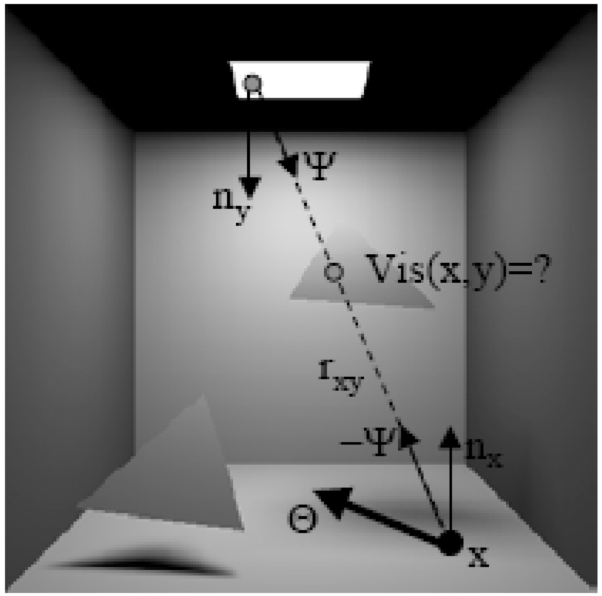
Direct Illumination

$$L(x \to \Theta) = \int_{A_{source}} f_r(x, -\Psi \leftrightarrow \Theta) \cdot L(y \to \Psi) \cdot G(x,y) \cdot dA_y$$

$$G(x,y) = \frac{\cos(n_x, \Theta)\cos(n_y, \Psi) Vis(x,y)}{r_{xy}^2}$$



hemisphere integration



area integration

Estimator for direct lighting

- Pick a point on the light's surface with pdf

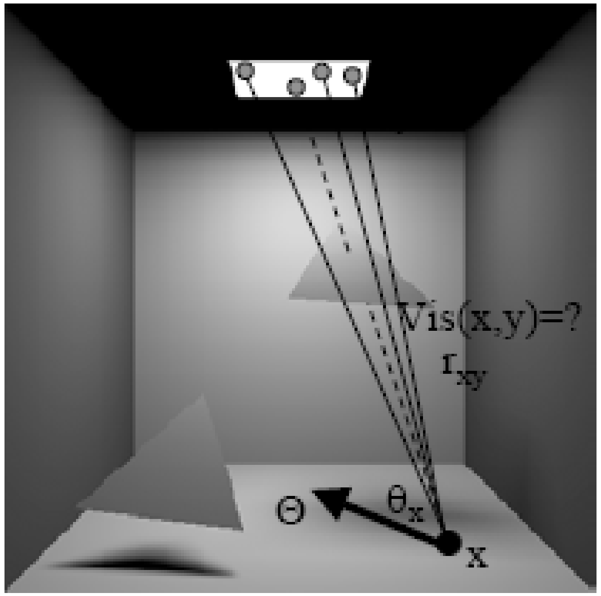
$$p(y)$$

- For N samples, direct light at point x is:

$$E(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{f_r L_{source}\frac{\cos\theta_x \cos\theta_{\bar{y}_i}}{r_{x\bar{y}_i}^2} Vis(x,\bar{y}_i)}{p(\bar{y}_i)}$$

Generating direct paths

- Pick surface points y_i on light source
- Evaluate direct illumination integral



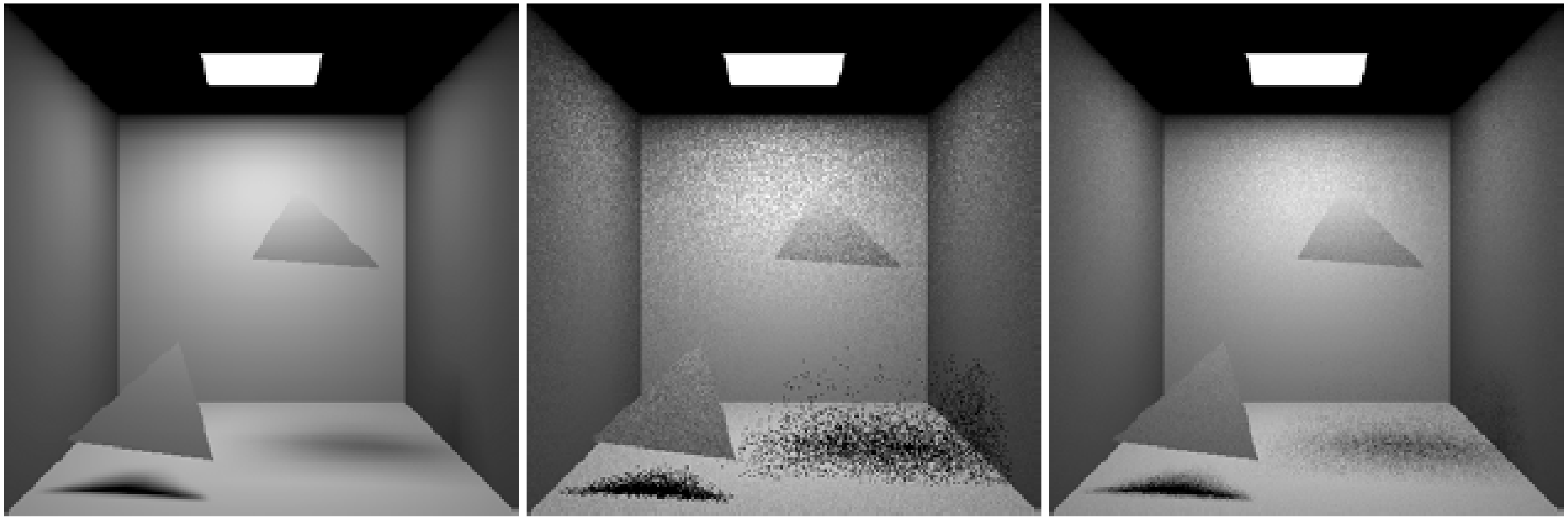
$$\langle L(x \to \Theta) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(\ldots) L(\ldots) G(x, y_i)}{p(y_i)}$$

PDF for sampling light

- Uniform $$p(y) = \frac{1}{Area_{source}}$$
- Pick a point uniformly over light's area
  - Can stratify samples
- Estimator:

$$E(x) = \frac{Area_{source}}{N} \sum_{i=1}^{N} f_r L_{source} \frac{\cos\theta_x \cos\theta_{\bar{y}_i}}{r_{x\bar{y}_i}^2} Vis(x, \bar{y}_i)$$

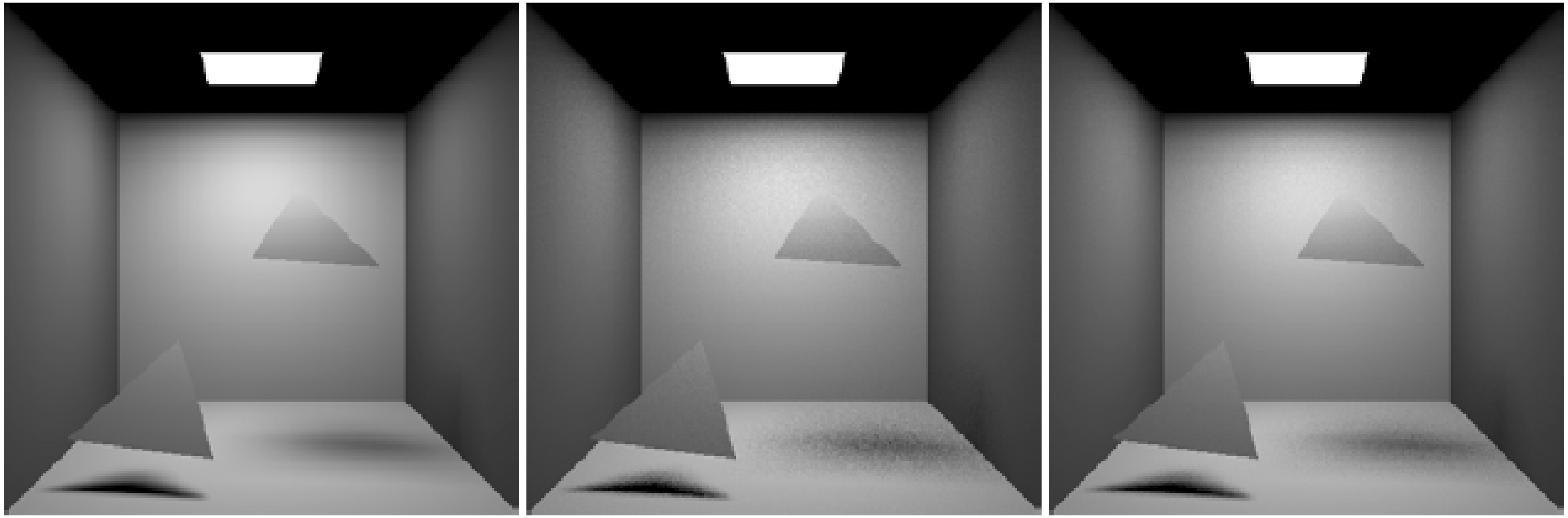
More points ...



1 shadow ray   9 shadow rays

$$E(x) = \frac{Area_{source}}{N} \sum_{i=1}^{N} f_r L_{source} \frac{\cos\theta_x \cos\theta_{\overline{y}_i}}{r^2_{x\overline{y}_i}} Vis(x, \overline{y}_i)$$

Even more points ...



36 shadow rays

100 shadow rays

$$E(x) = \frac{Area_{source}}{N} \sum_{i=1}^{N} f_r L_{source} \frac{\cos\theta_x \cos\theta_{\bar{y}_i}}{r_{x\bar{y}_i}^2} Vis(x, \bar{y}_i)$$

Different pdfs

- Uniform

$$p(y) = \frac{1}{Area_{source}}$$

- Solid angle sampling
  - Removes cosine and distance from integrand
  - Better when significant foreshortening

$$E(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r L_{source} \frac{\cos\theta_x \cos\theta_{\bar{y}_i}}{r_{x\bar{y}_i}^2} Vis(x, \bar{y}_i)}{p(\bar{y}_i)}$$

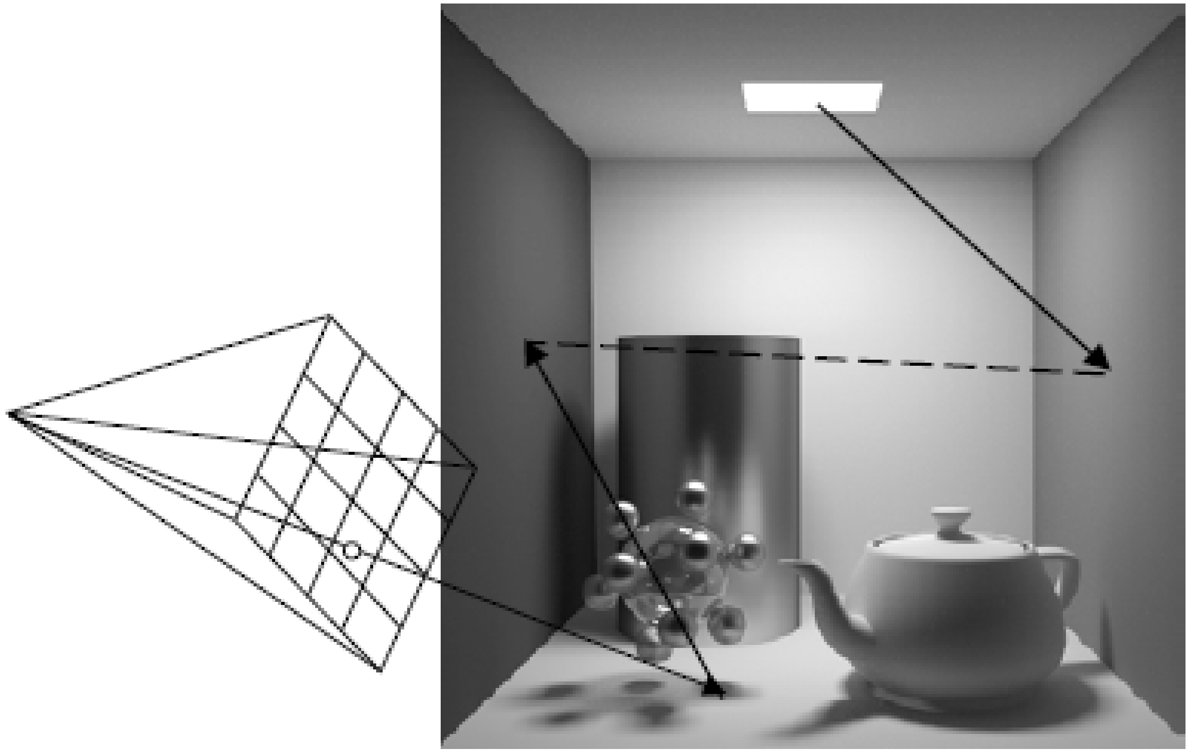
Parameters

- How to distribute paths within light source?
  - Uniform
  - Solid angle
  - What about light distribution?

- How many paths (“shadow-rays”)?
  - Total?
  - Per light source? (~intensity, importance, …)

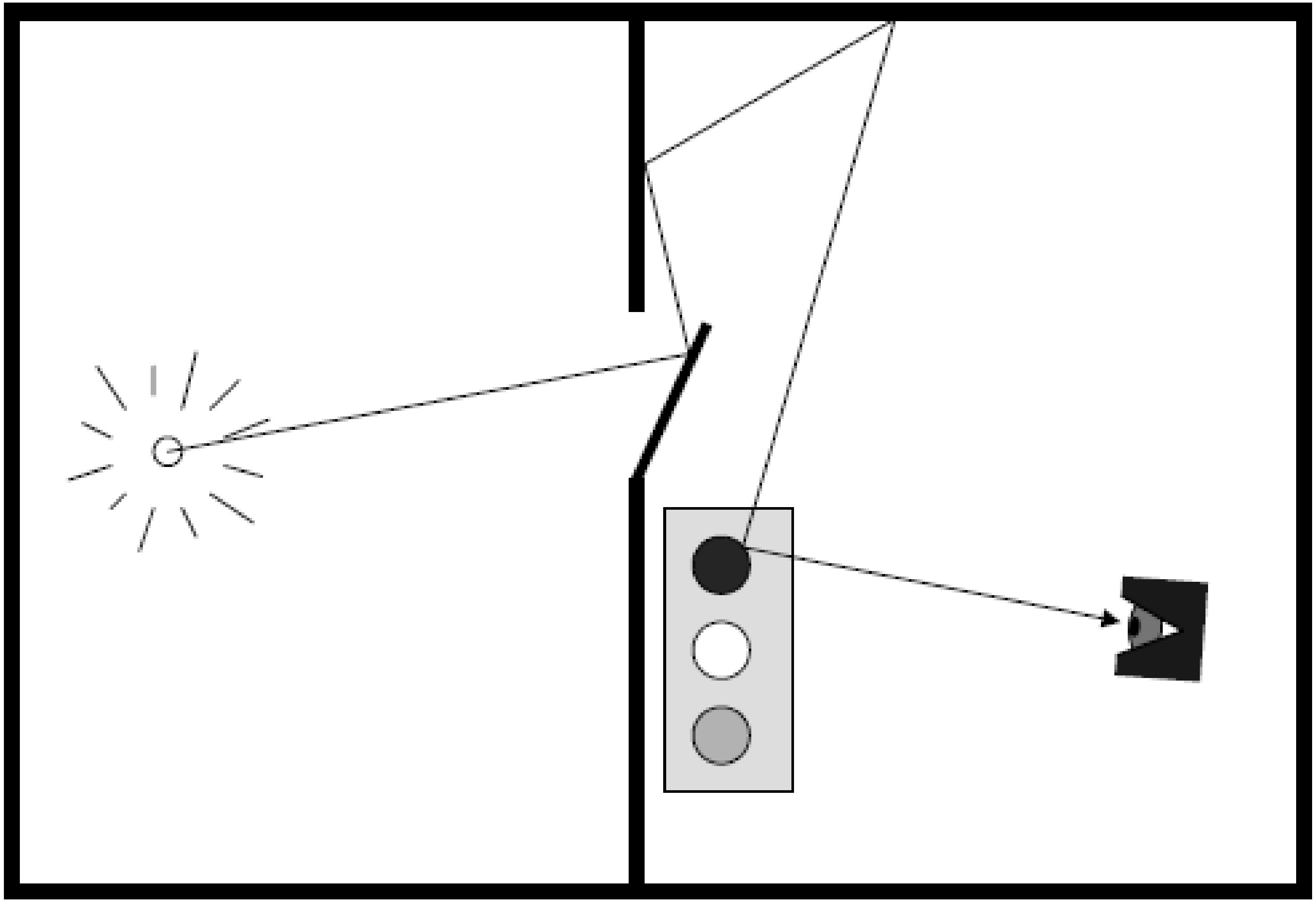
Bidirectional Path Tracing

- Or paths generated from both camera and source at the same time …!

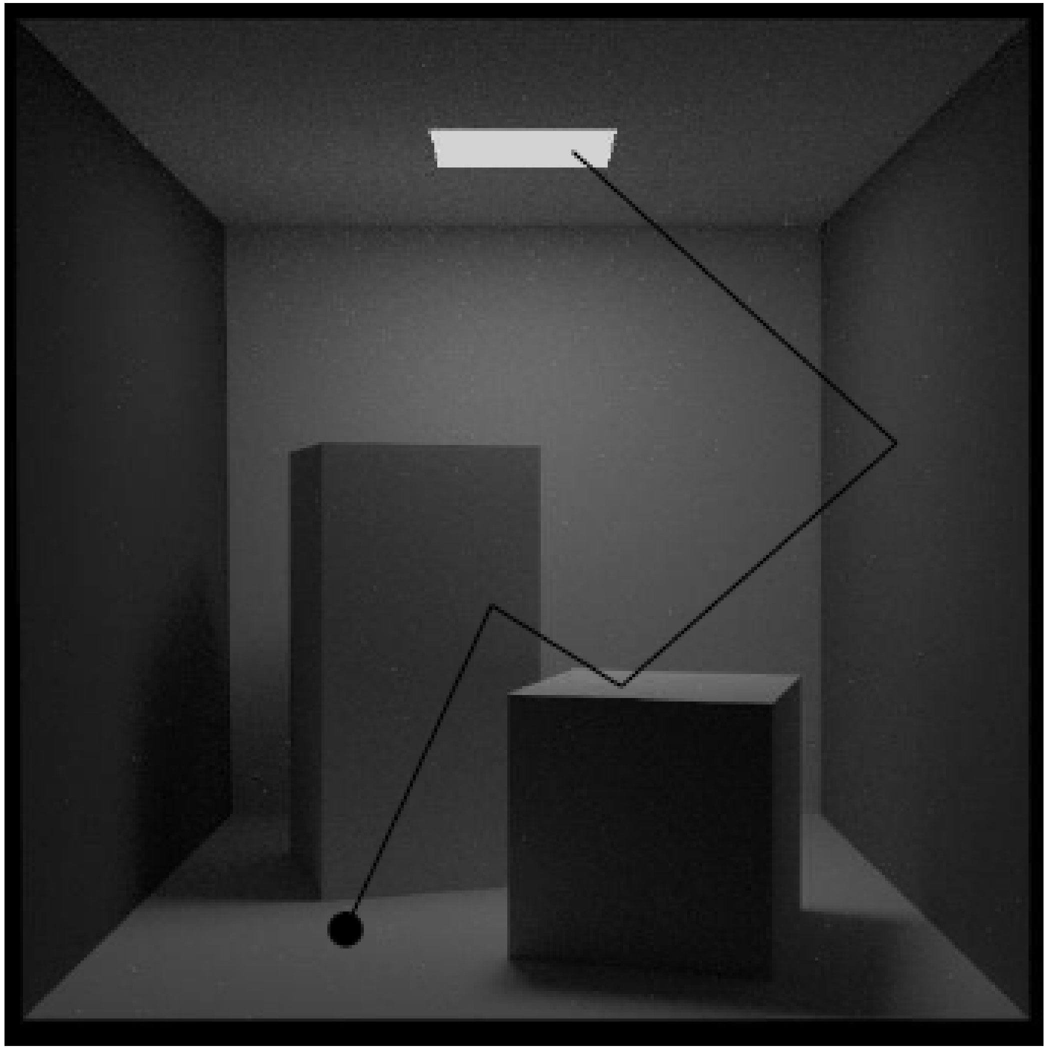


- Connect endpoints to compute final contribution

Metropolis



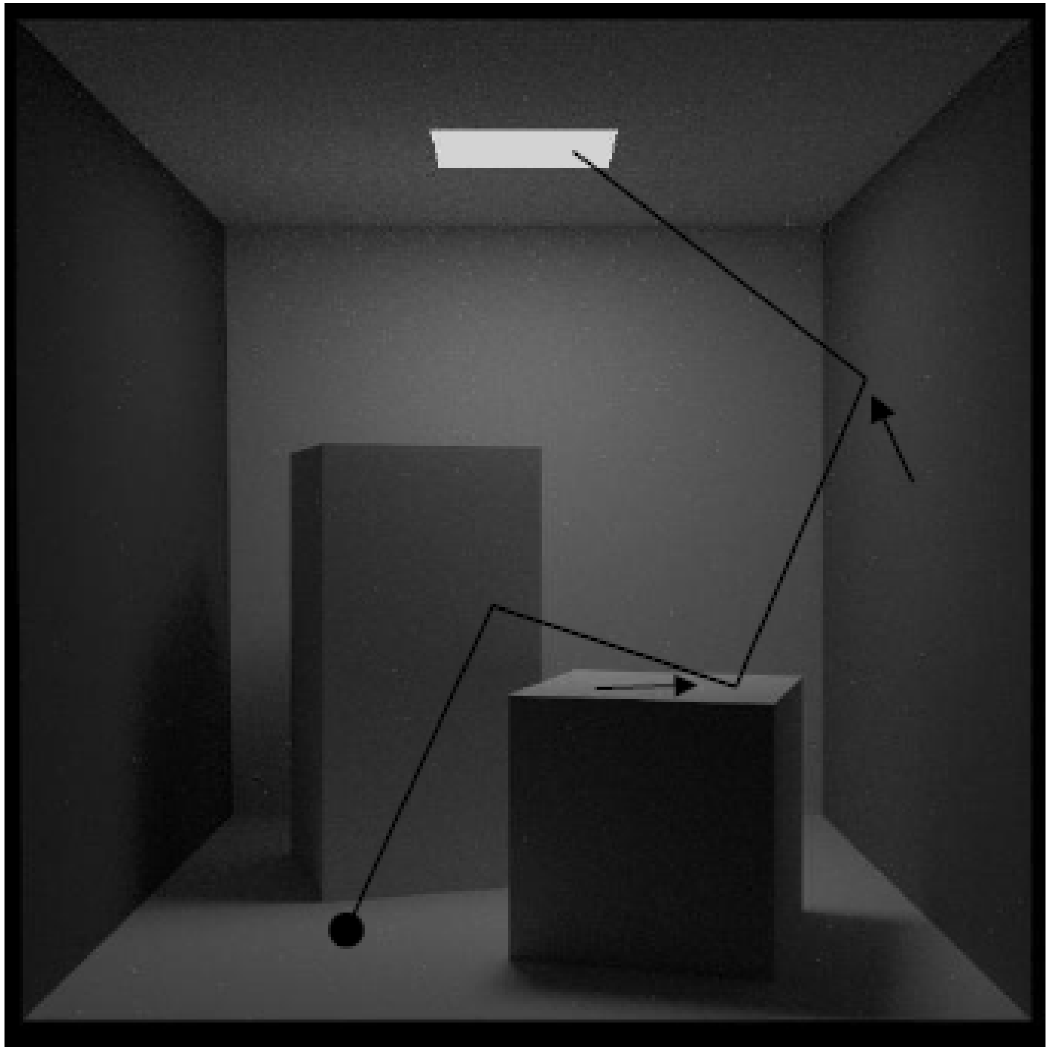
Metropolis



© Kavita Bala, Computer Science, Cornell University

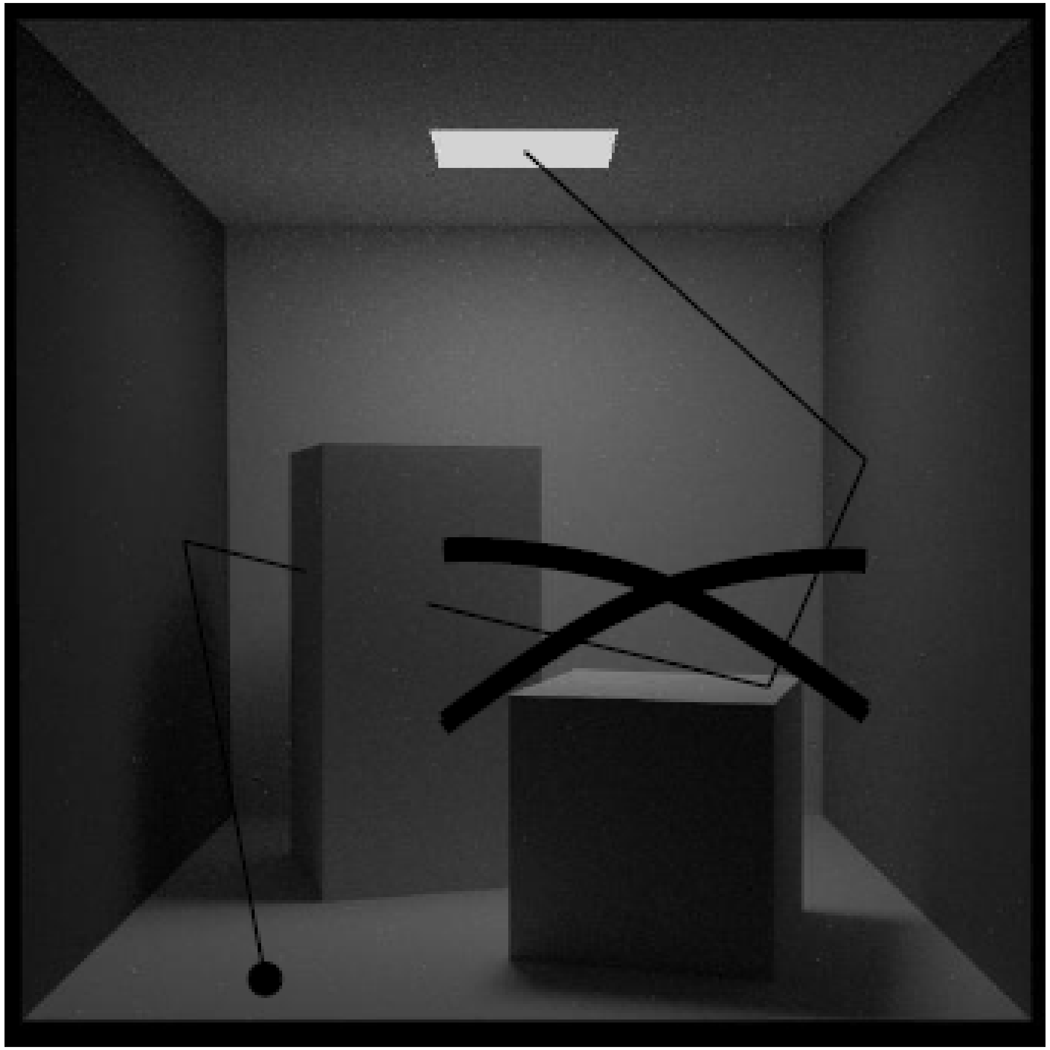
valid path

Metropolis



small
perturbations

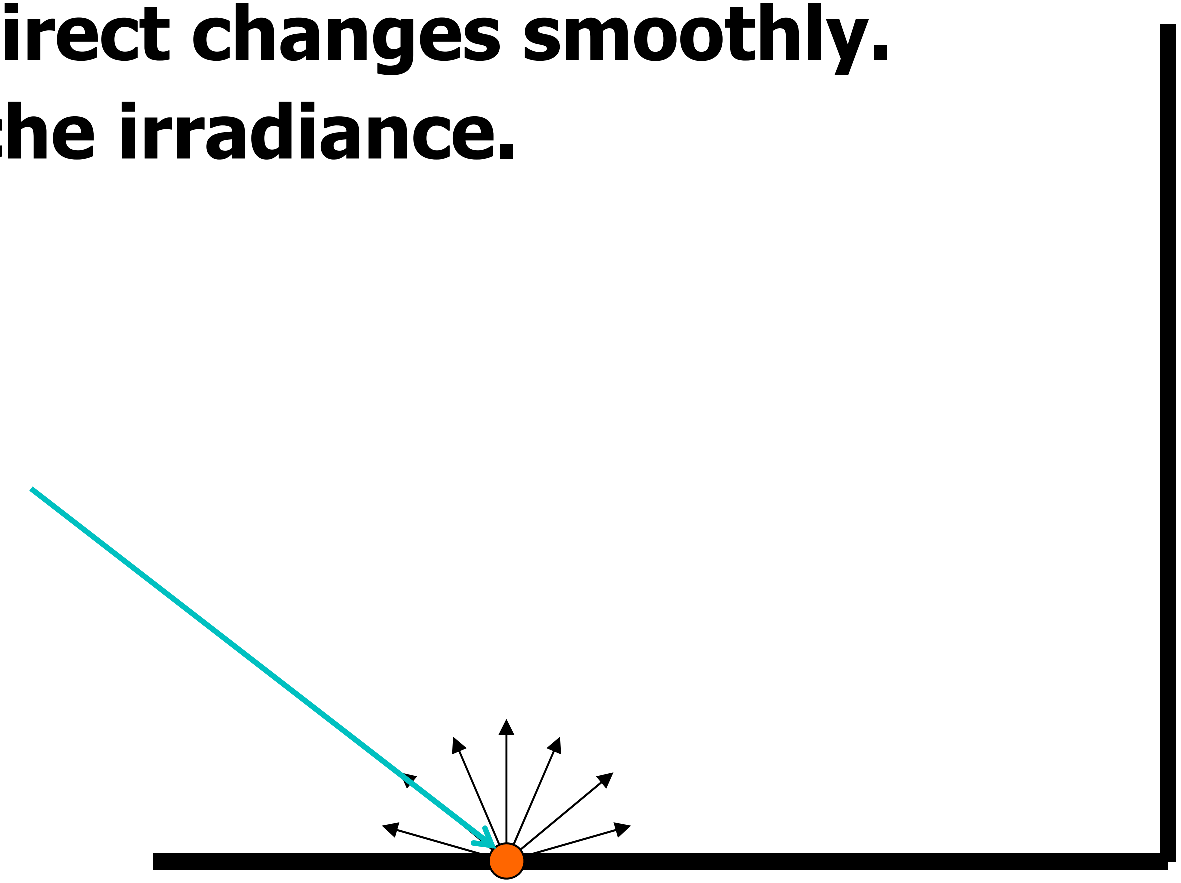
Metropolis



Accept mutations based on energy transport

Biased Methods: Irradiance Caching

- Indirect changes smoothly.
- Cache irradiance.

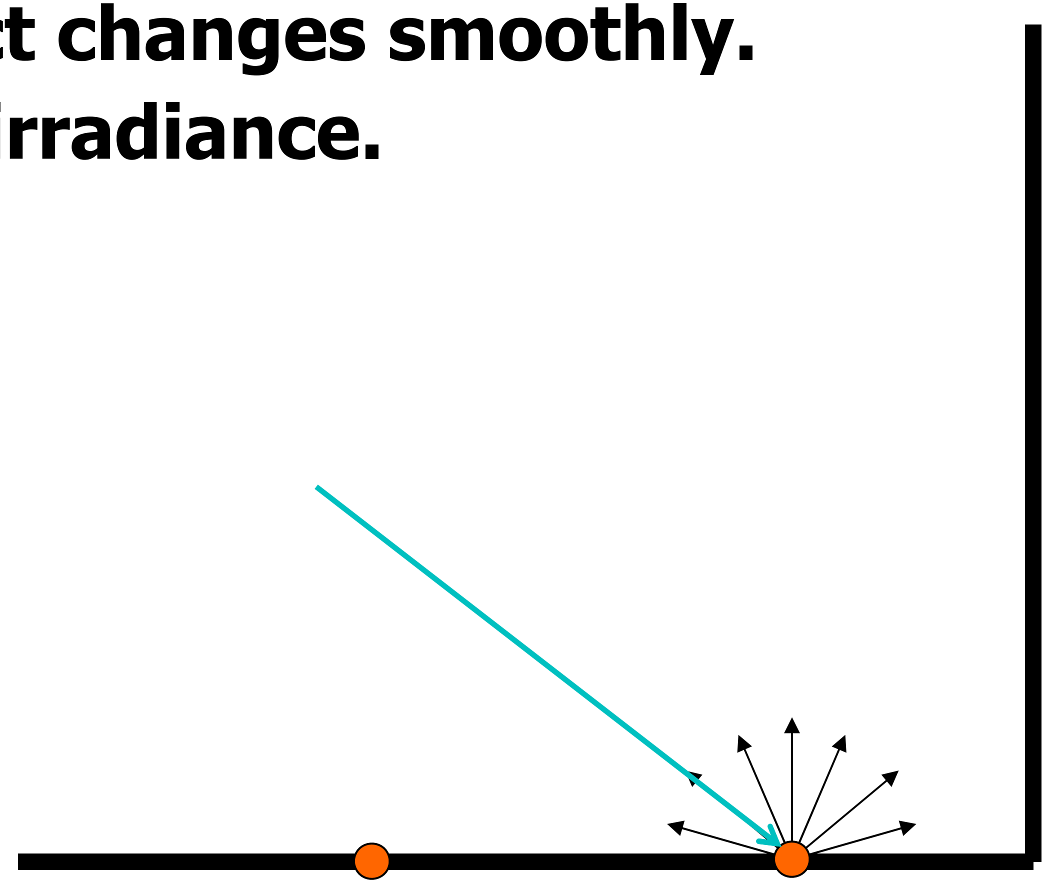


From Wang's slides

KAIST

Irradiance Caching

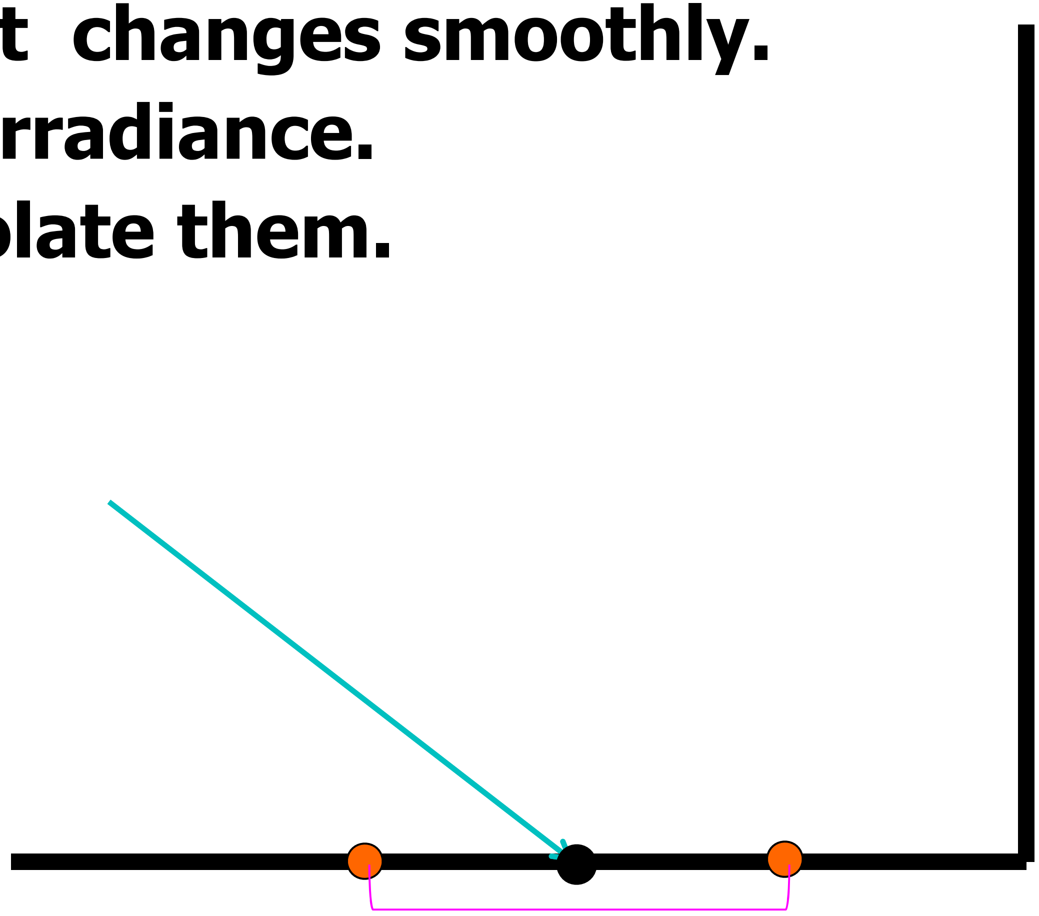
- Indirect changes smoothly.
- Cache irradiance.



From Wang's slides

Irradiance Caching

- Indirect changes smoothly.
- Cache irradiance.
- Interpolate them.

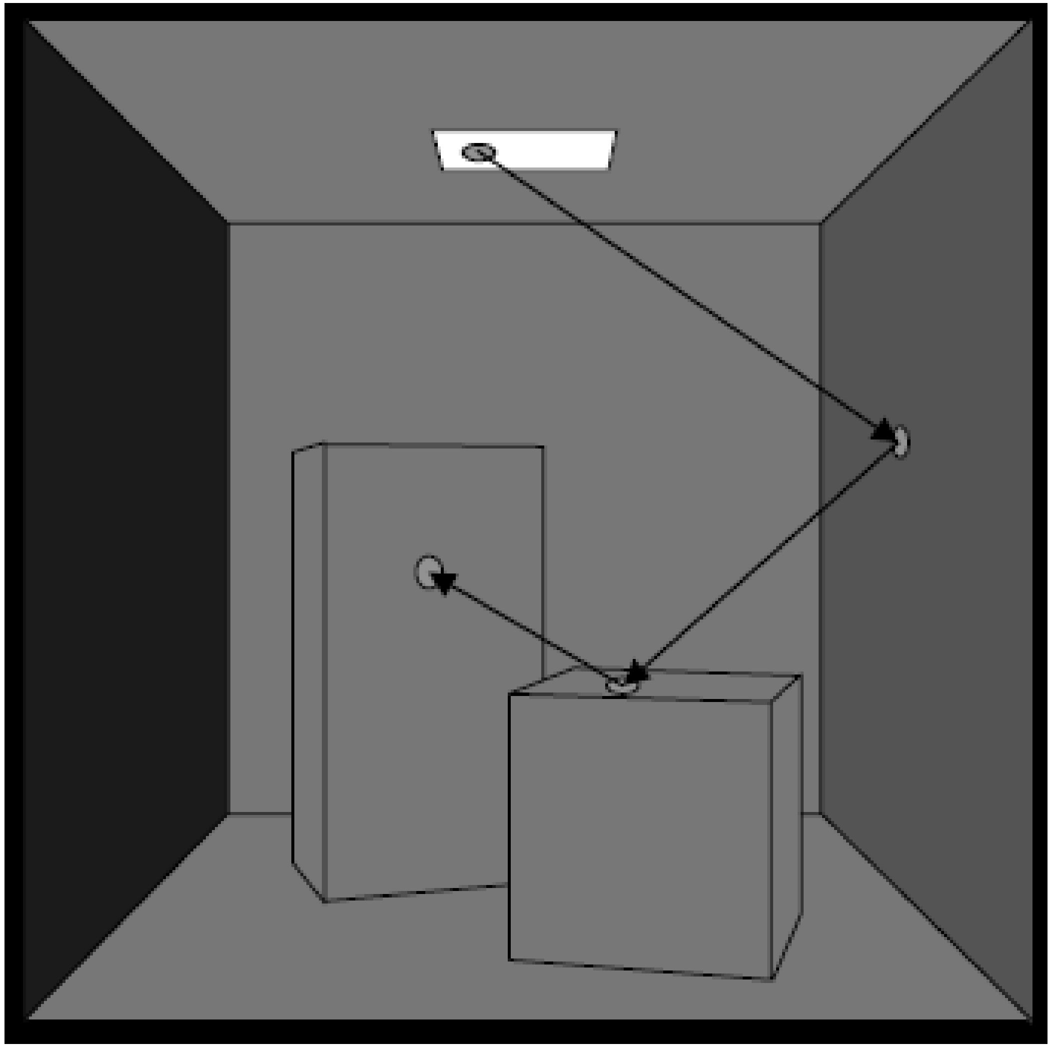


From Wang's slides

Biased Method: Photon Mapping

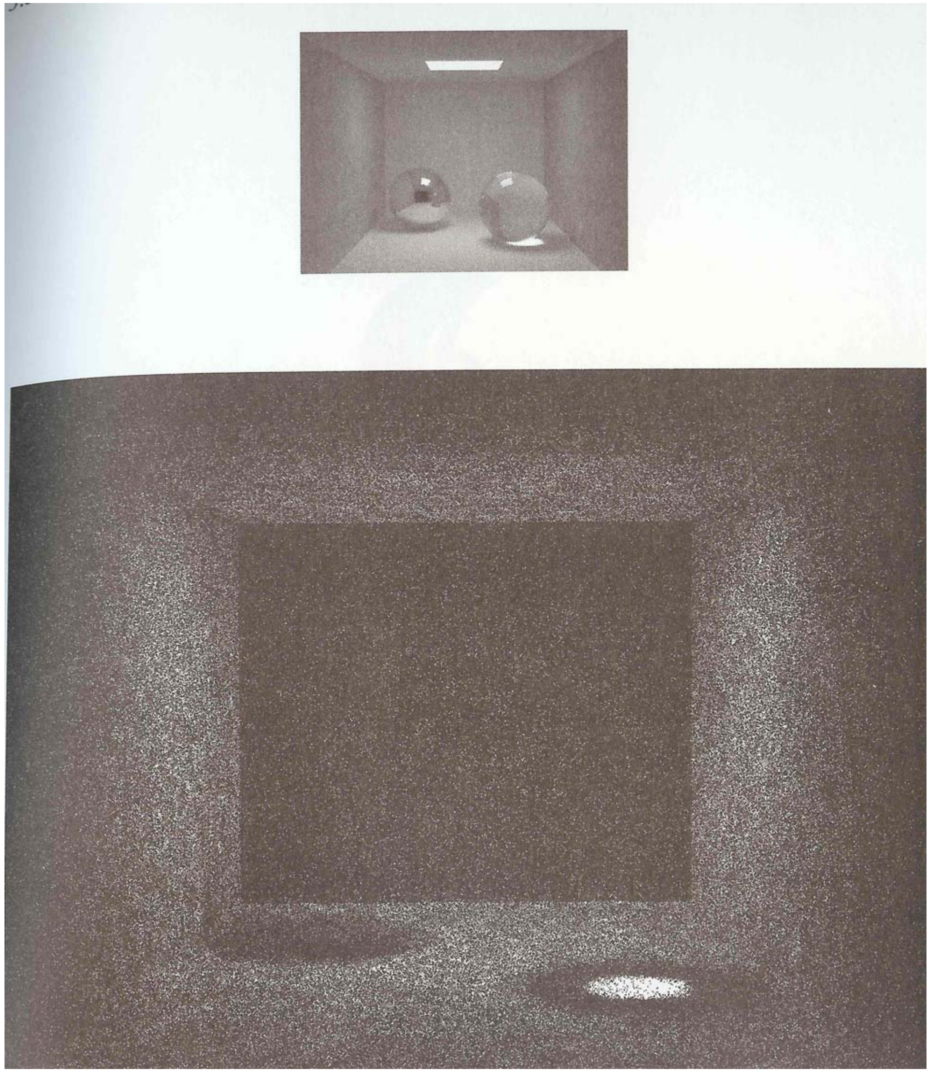
- **2 passes:**
  - **Shoot "photons" (light-rays) and record any hit-points**
  - **Shoot viewing rays and collect information from stored photons**

Pass 1: shoot photons



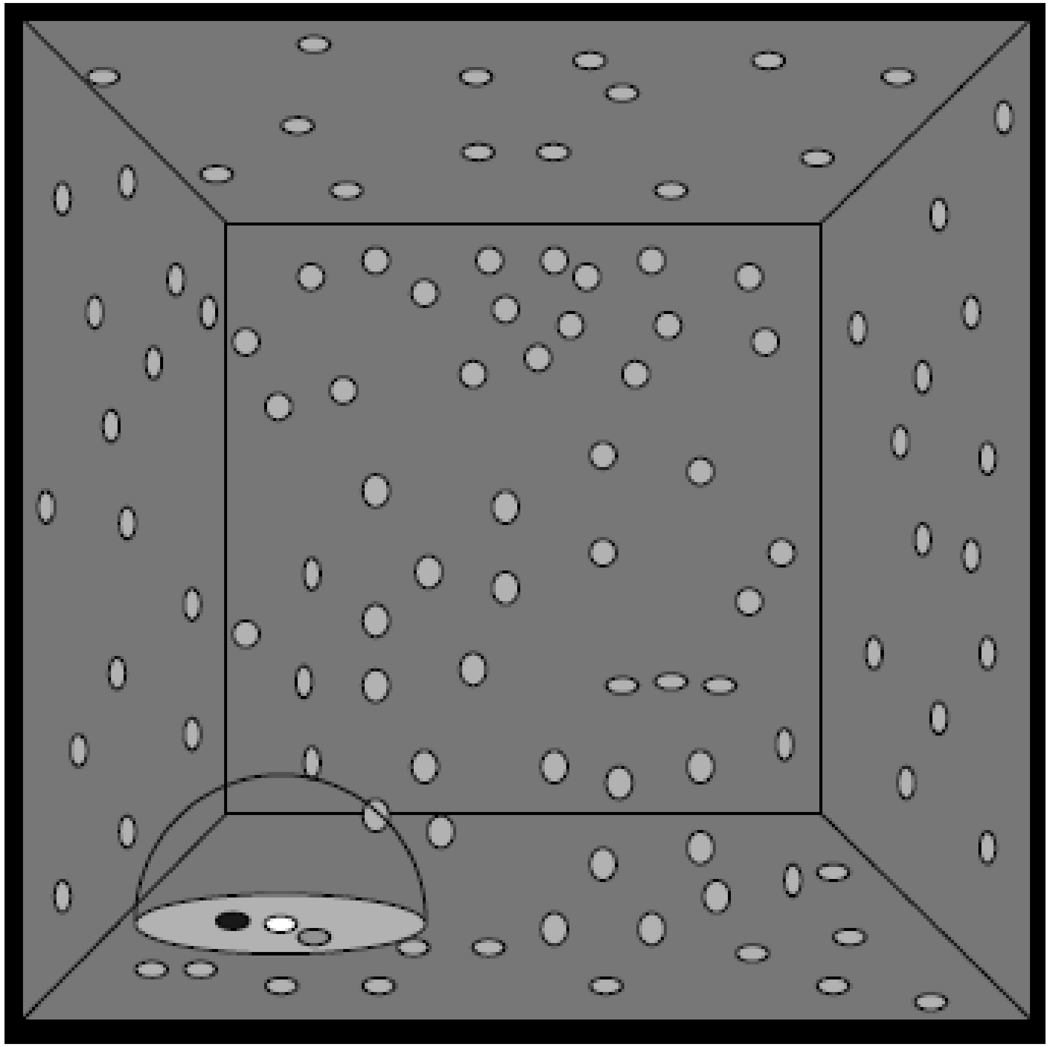
- Light path generated using MC techniques and Russian Roulette
- Store:
  - position
  - incoming direction
  - color
  - ...

Stored Photons



Generate a few hundreds of thousands of photons

Pass 2: viewing ray



- Search for N closest photons (+check normal)
- Assume these photons hit the point we're interested in
- Compute average radiance

Result

350K photons for the caustic map

Result

350K photons for the caustic map

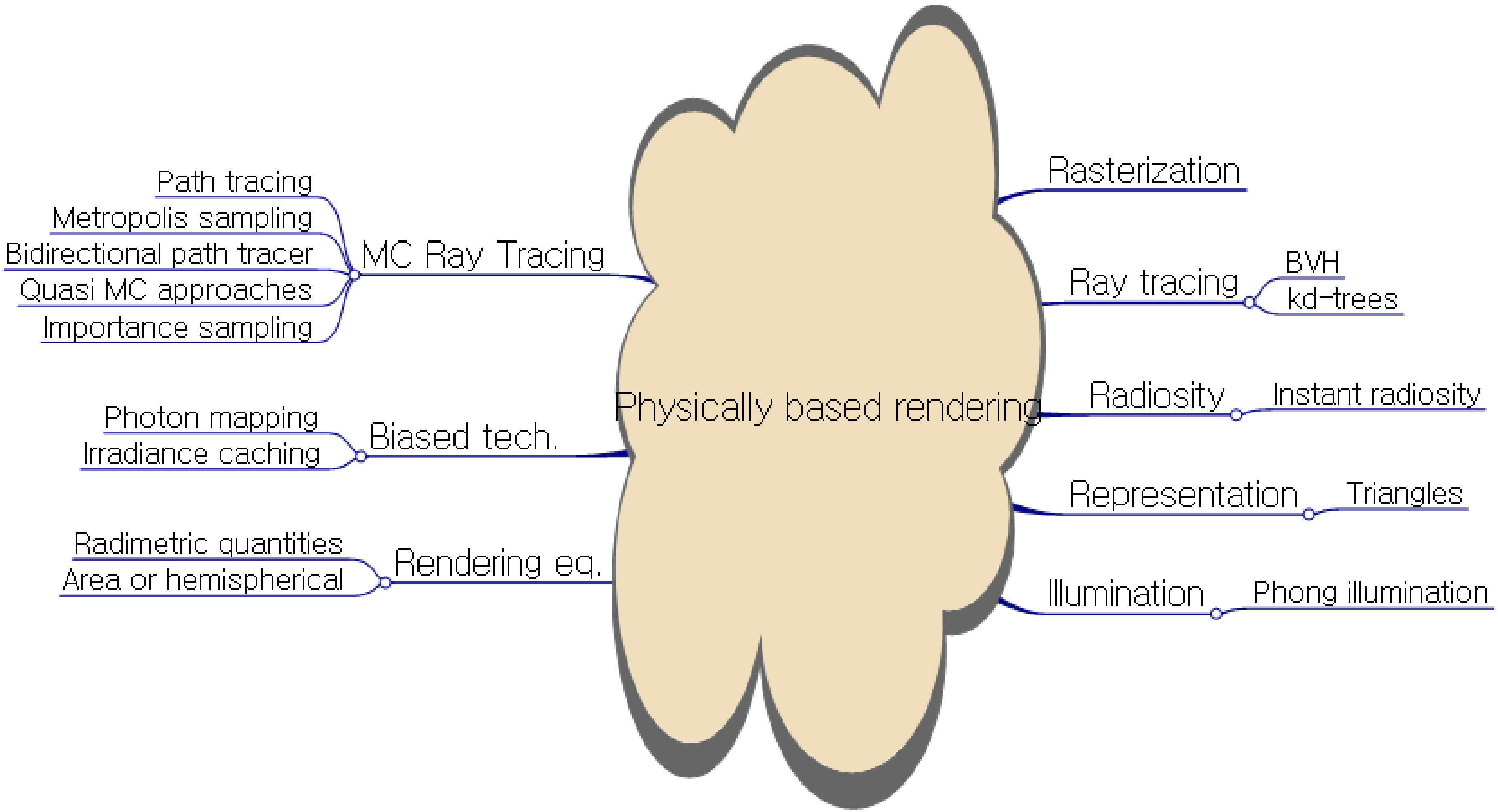
Class Objectives were:

- Understand a basic structure of Monte Carlo ray tracing
  - Russian roulette for its termination
  - Path tracing

Summary

- Two basic building blocks
- Radiometry
- Rendering equation
- MC integration
- MC ray tracing
  - Unbiased methods
  - Biased methods

Summary



Physically based rendering
MC Ray Tracing
Path tracing
Metropolis sampling
Bidirectional path tracer
Quasi MC approaches
Importance sampling
Biased tech.
Photon mapping
Irradiance caching
Rendering eq.
Radimetric quantities
Area or hemispherical
Rasterization
Ray tracing
BVH
kd-trees
Radiosity
Instant radiosity
Representation
Triangles
Illumination
Phong illumination

Next Time…

- Instant radiosity

Homework

- **Go over the next lecture slides before the class**
- **Watch 2 SIG/I3D/HPG videos and submit your summaries every Tue. class**
  - **Just one paragraph for each summary**

Example:

Title: XXX XXXX XXXX

Abstract: this video is about accelerating the performance of ray tracing. To achieve its goal, they design a new technique for reordering rays, since by doing so, they can improve the ray coherence and thus improve the overall performance.

Any Questions?

- **Submit four times in Sep./Oct.**
- **Come up with one question on what we have discussed in the class and submit at the end of the class**
  - **1 for typical questions**
  - **2 for questions that have some thoughts or surprise me**