CS680:
Monte Carlo Ray Tracing

Sung-Eui Yoon
(윤성의)

Course URL:
http://jupiter.kaist.ac.kr/~sungeui/SGA/

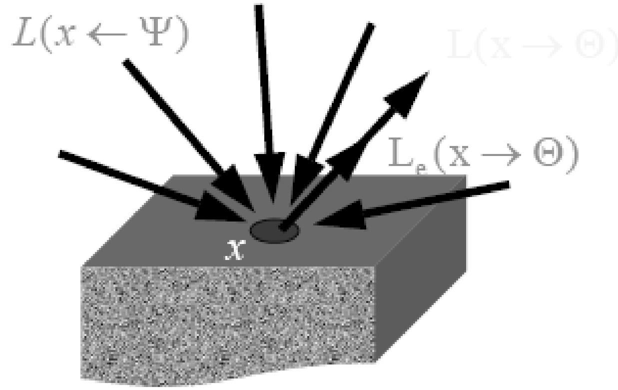
Previous Time

- **Monte Carlo integration**

Why Monte Carlo?

- **Radiace is hard to evaluate**

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



From kavita's slides

- **Sample many paths**
  - **Integrate over all incoming directions**
- **Analytical integration is difficult**
  - **Need numerical techniques**

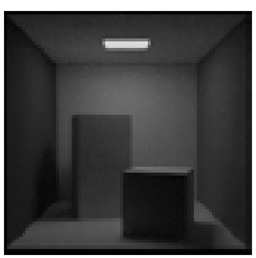
KAIST

Rendering Equation

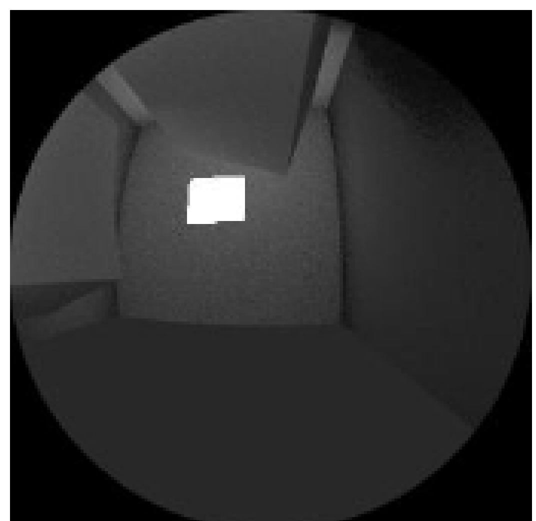
$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_{\Omega_x} \underline{f_r(\Psi \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi}$$

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

Value we want



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



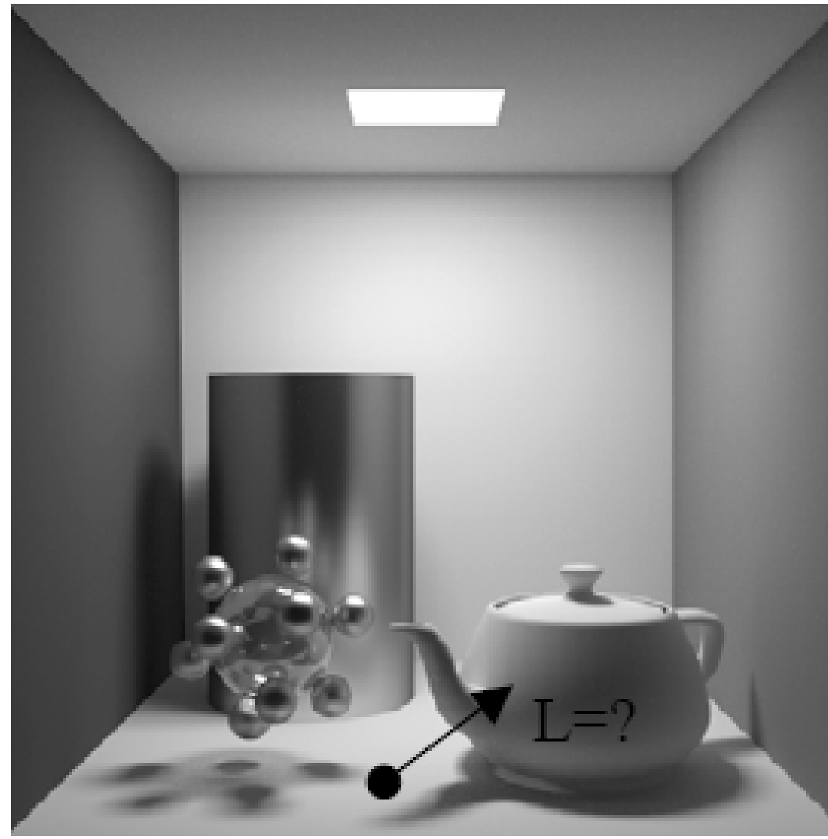
$$\cdot f_r \cdot \cos$$

How to compute?

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

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

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



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

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?

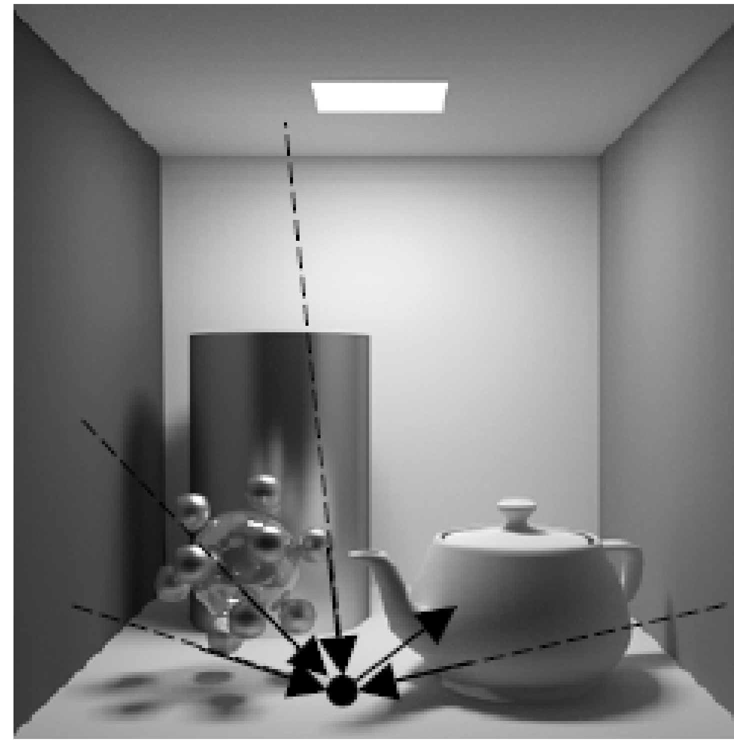
Generate random directions Ψ_i

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

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

How to compute?

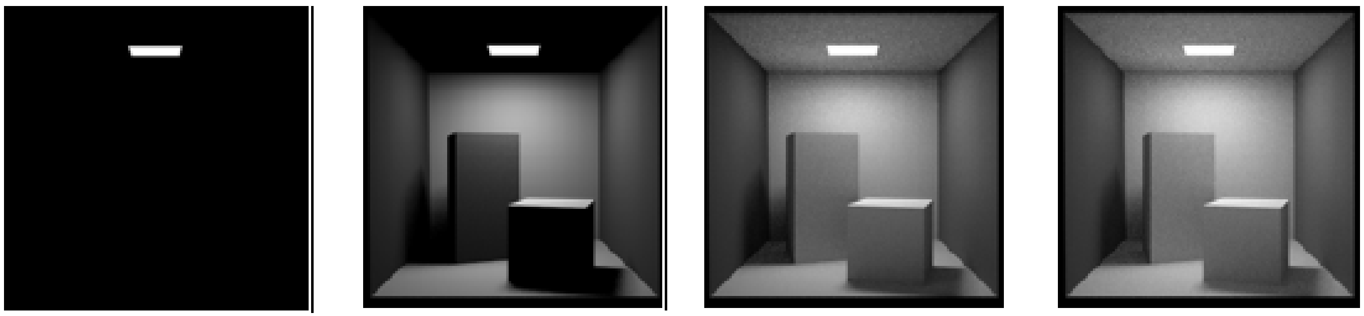
- evaluate $L(x \leftarrow \Psi_i)$?
- Radiance is invariant along straight paths
- $vp(x, \Psi_i)$ = first visible point
- $L(x \leftarrow \Psi_i) = L(vp(x, \Psi_i) \rightarrow \Psi_i)$



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

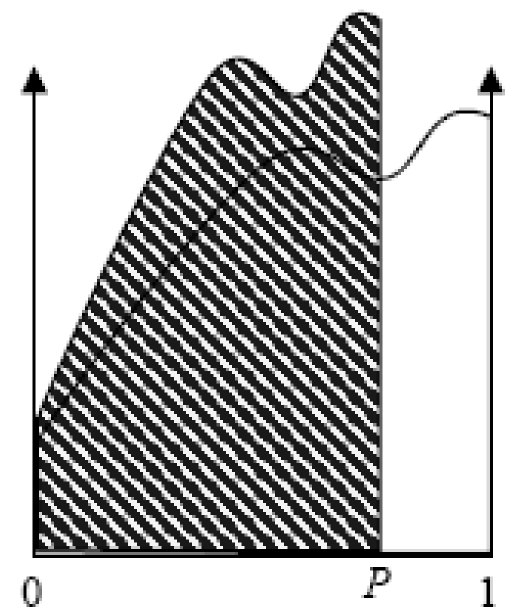
- **Contributions of further light bounces become less significant**
  - **Max recursion**
  - **Some threshold for radiance value**

- **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

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

Variance $\sigma_{roulette} > \sigma$

Russian Roulette

- **Pick absorption probability, α = 1-P**
  - **Recursion is terminated**

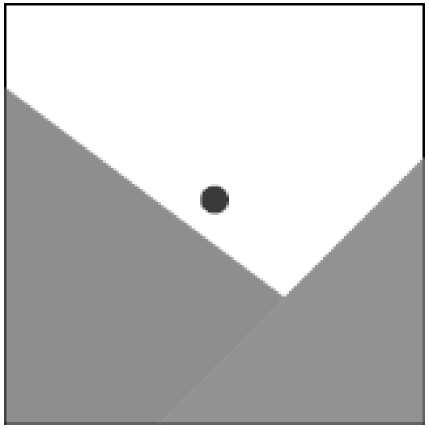
- **1- α is commonly to be equal to the reflectance of the material of the surface**
  - **Darker surface absorbs more paths**

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



- Simple box filter
  - The averaging method

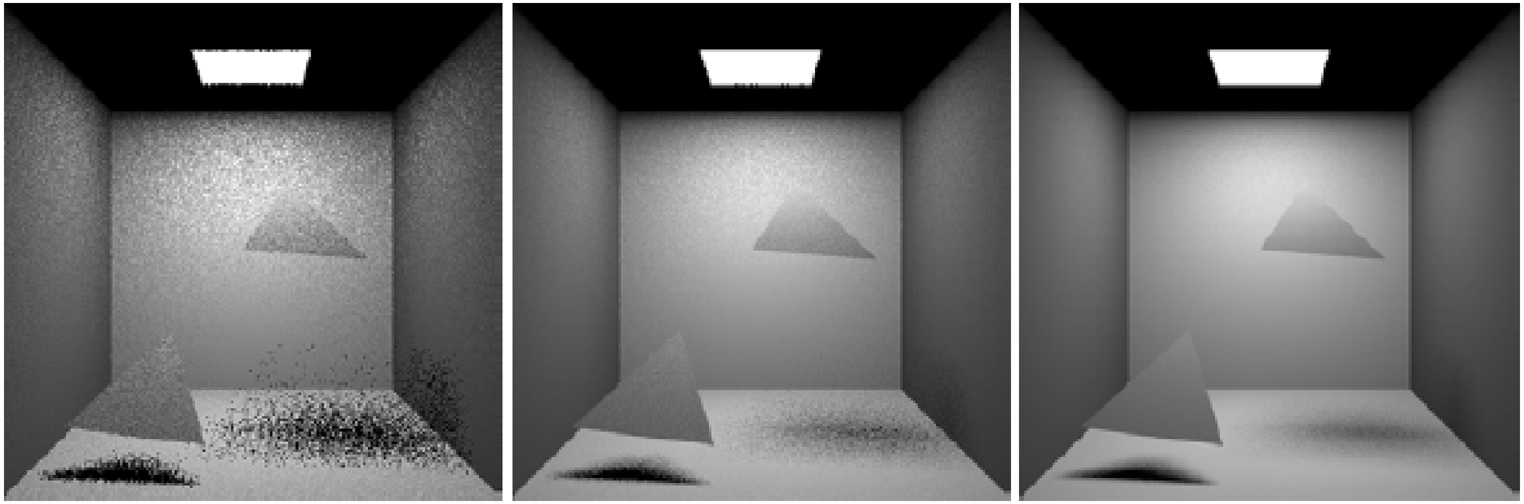
- We want to evaluate using MC

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

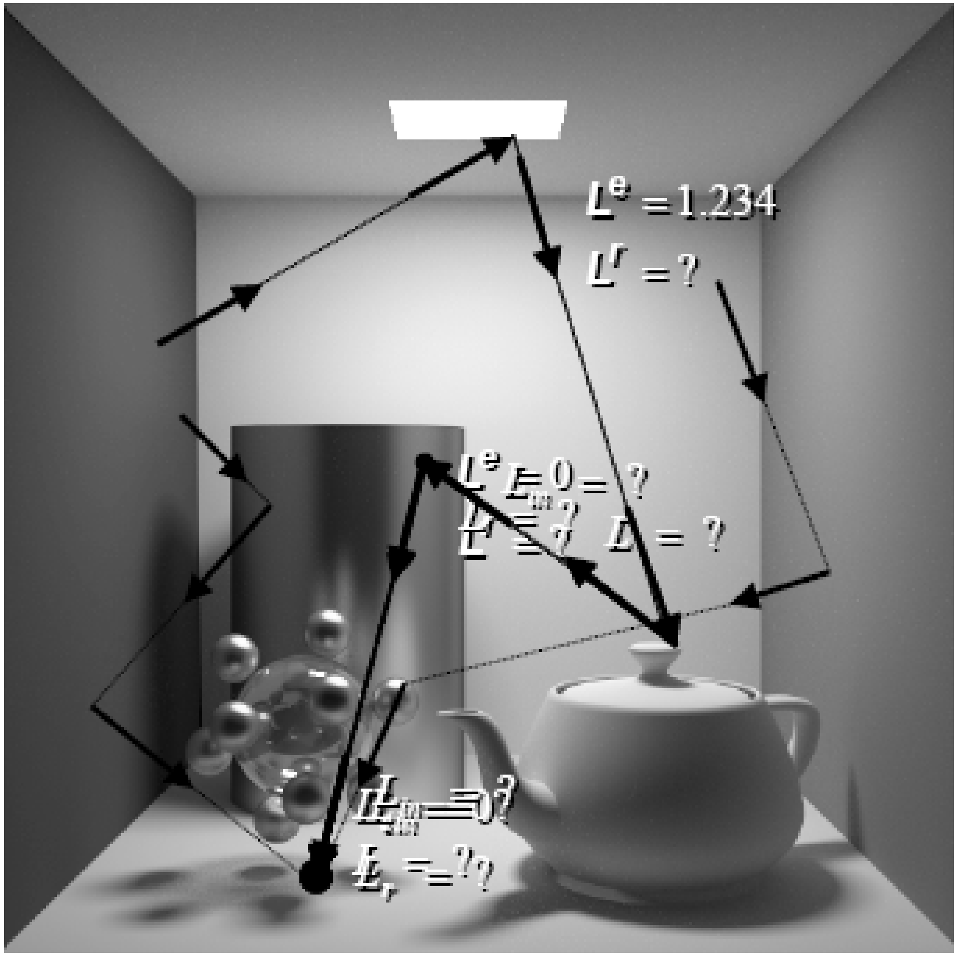
From kavita's slides

- **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**

Algorithm



Performance

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

- **Faster convergence**
  - **Importance sampling**

Stratified Sampling

- Samples could be arbitrarily close

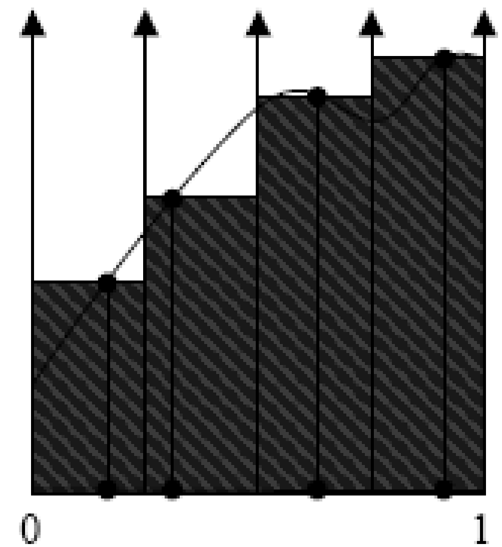
- Split integral in subparts

$$I = \int_{X_1} f(x)dx + \ldots + \int_{X_N} f(x)dx$$

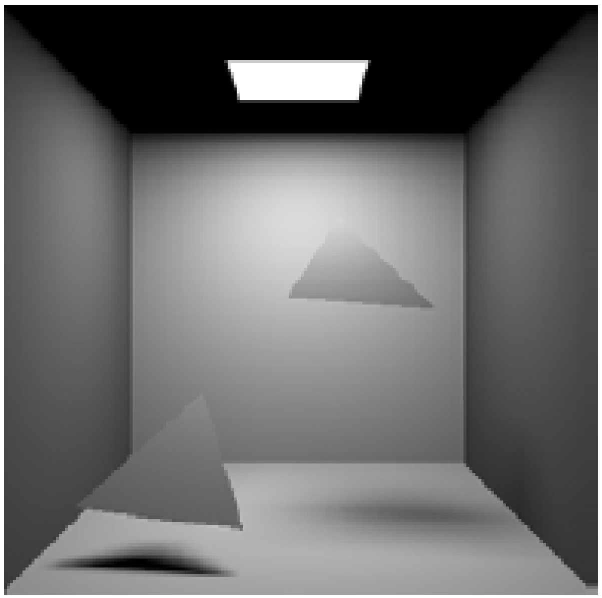
- Estimator

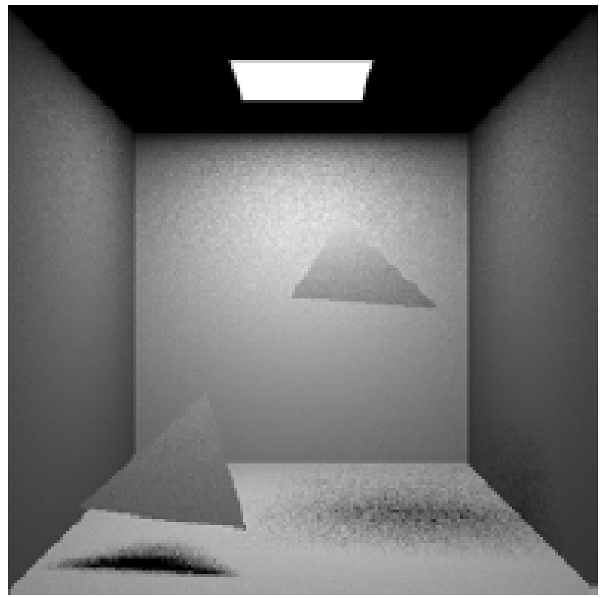
$$\bar{I}_{strat} = \frac{1}{N}\sum_{i=1}^{N}\frac{f(\bar{x}_i)}{p(\bar{x}_i)}$$

- Variance: $\sigma_{strat} \leq \sigma_{sec}$

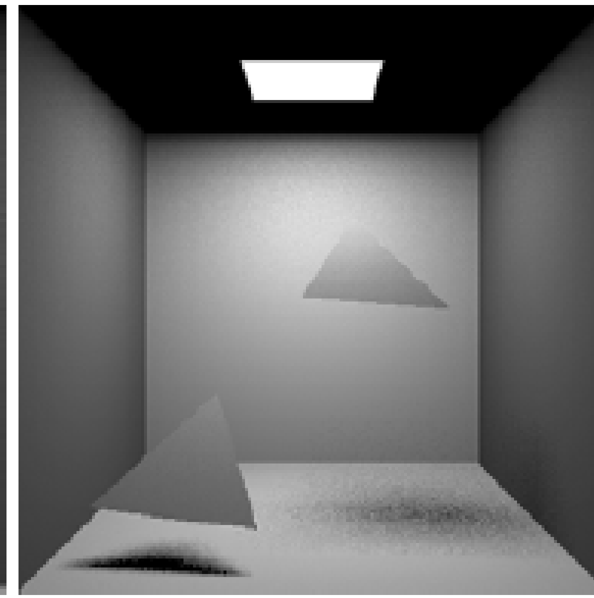


Stratified Sampling



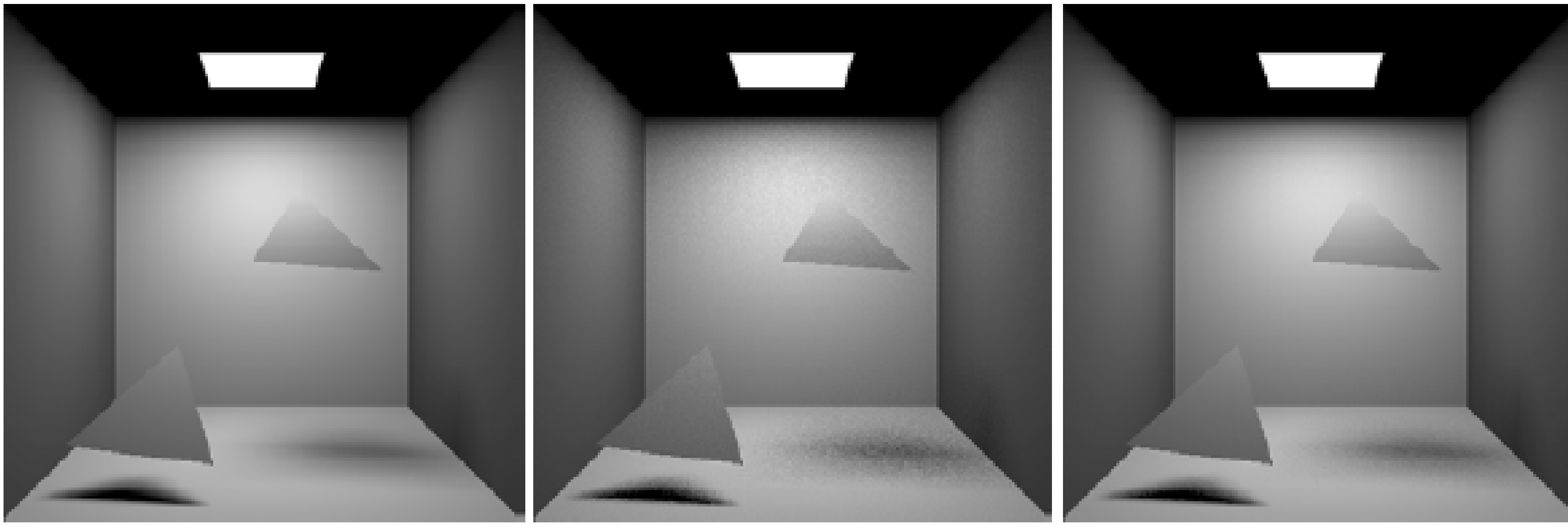


9 shadow rays
not stratified



9 shadow rays
stratified

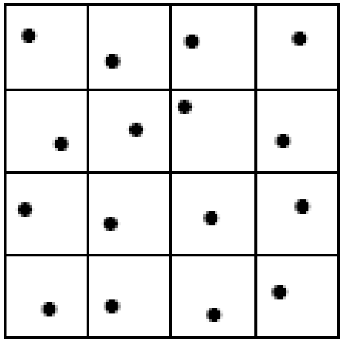
Stratified Sampling



36 shadow rays
not stratified

36 shadow rays
stratified

High Dimensions



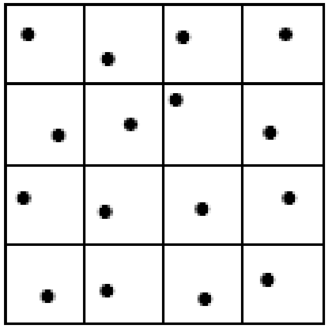
$\rightarrow N^2$ samples

- **Problem for higher dimensions**
- **Sample points can still be arbitrarily close to each other**

KAIST

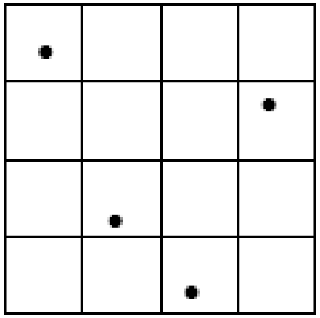
Higher Dimensions

- **Stratified grid sampling**



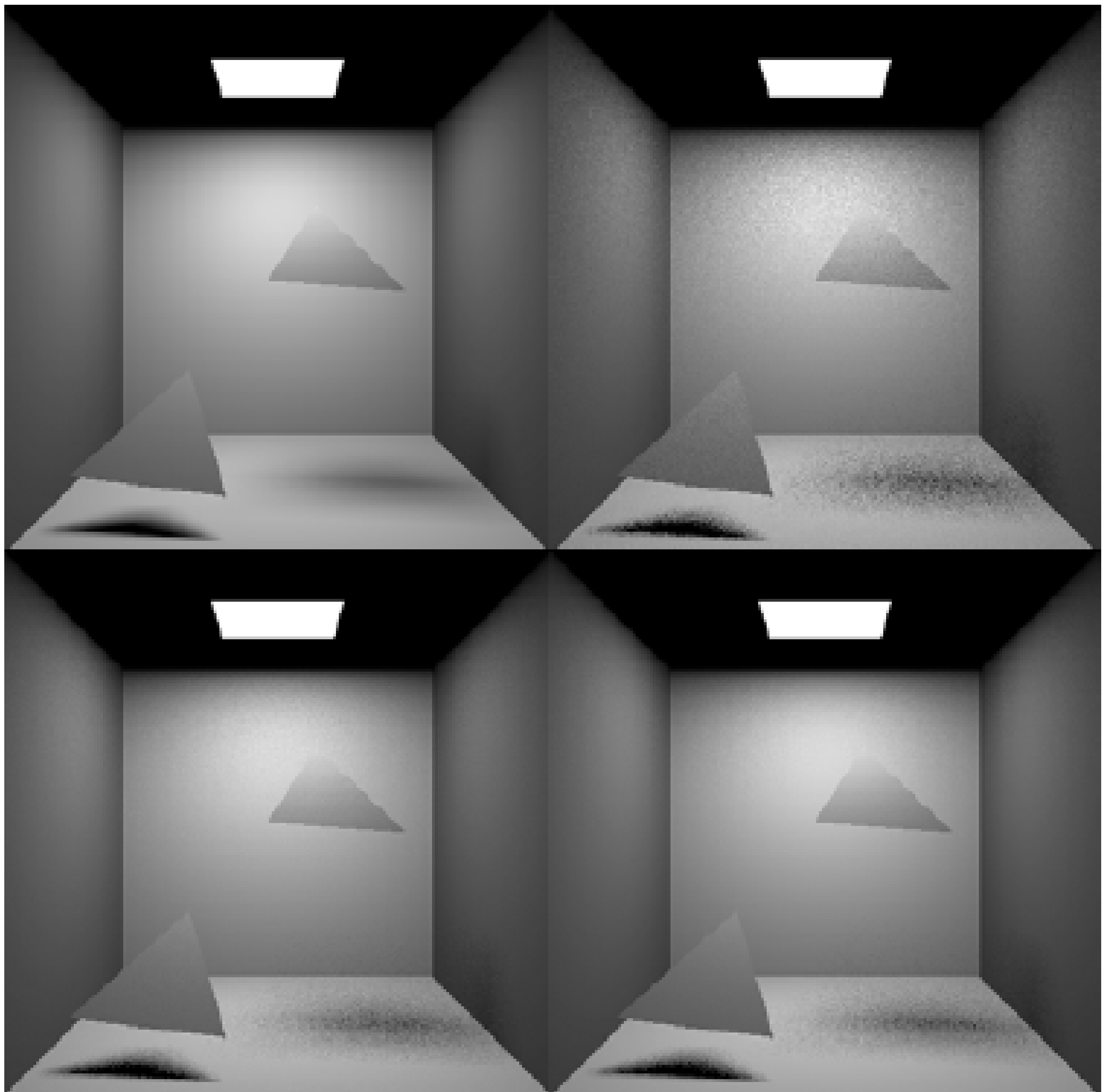
$\rightarrow N^d$ samples

- **N-rooks sampling**



$\rightarrow N$ samples

N-Rooks Sampling - 9 rays

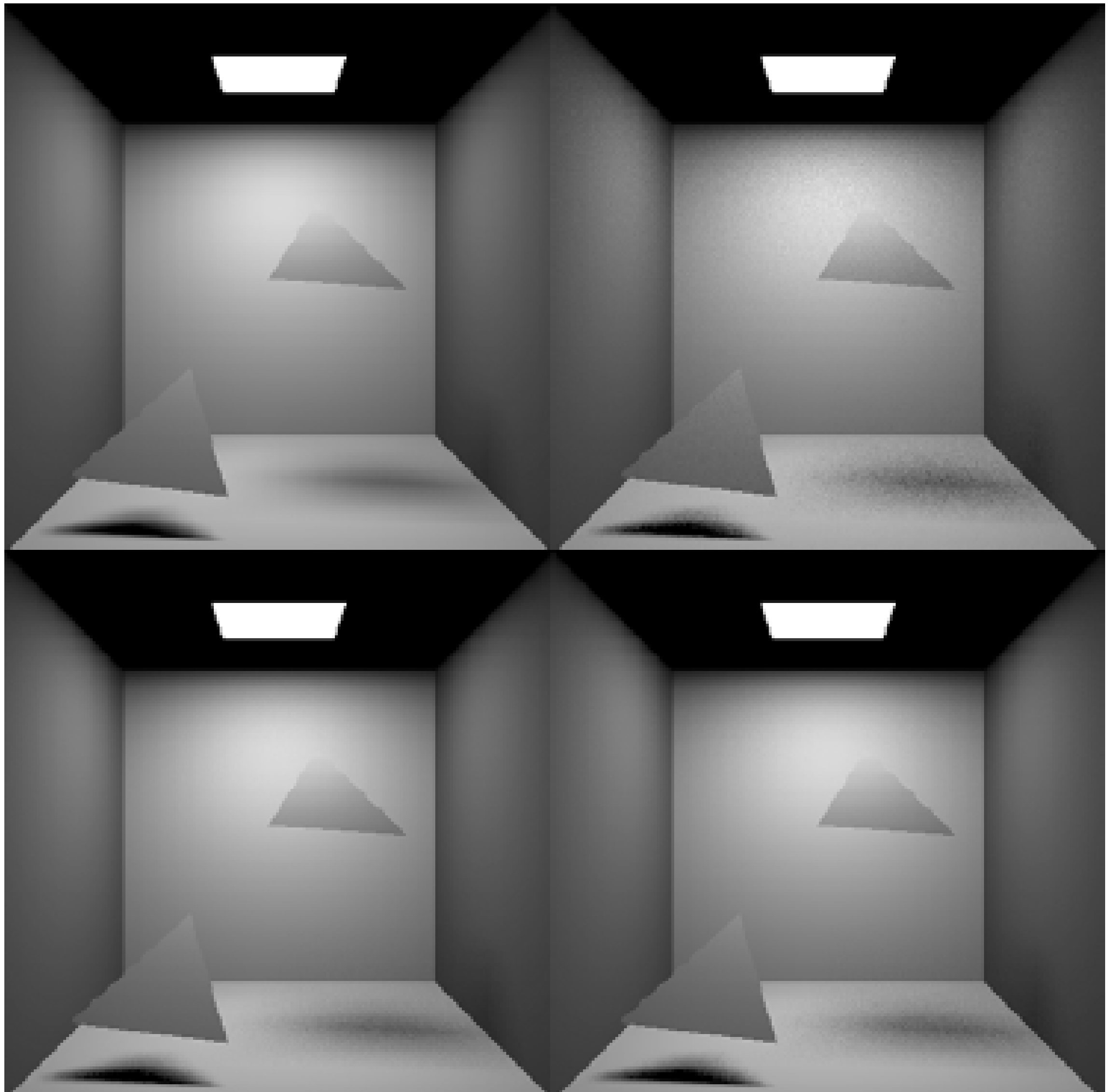


not stratified

stratified

N-Rooks

N-Rooks Sampling - 36 rays



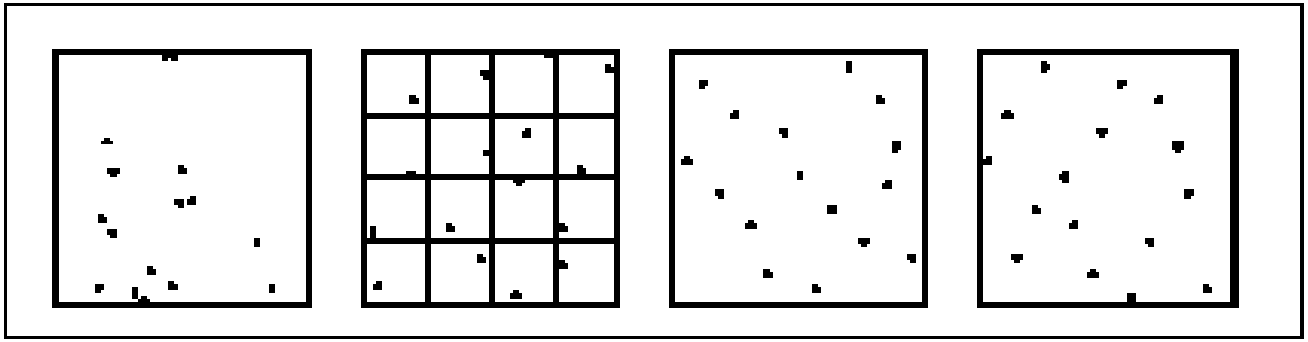
not stratified

stratified

N-Rooks

Quasi Monte Carlo

- Eliminates randomness to find well-distributed samples
- Samples are determinisitic but “appear” random



Quasi-Monte Carlo (QMC)

- Notions of variance, expected value don't apply
- Introduce the notion of discrepancy
  - Discrepancy mimics variance
  - E.g., subset of unit interval [0,x]
    - Of N samples, n are in subset
    - Discrepancy: $|x-n/N|$
  - Mainly: "it looks random"

Example: van der Corput Sequence

- **One of simplest low-discrepancy sequences**

- **Radical inverse function, $\Phi_b(n)$**
  - **Given n =** $\sum_{i=1}^{\infty} d_i b^{i-1}$ **,**
  - $\Phi_b(n) = 0.d_1 d_2 d_3 \ldots d_n$
  - **E.g., $\Phi_2(i)$: $111010_2 \rightarrow 0.010111$**

- **van der Corput sequence, $x_i = \Phi_2(i)$**

Example: van der Corput Sequence

- **One of simplest low-discrepancy sequences**
- $x_i = \Phi_2(i)$

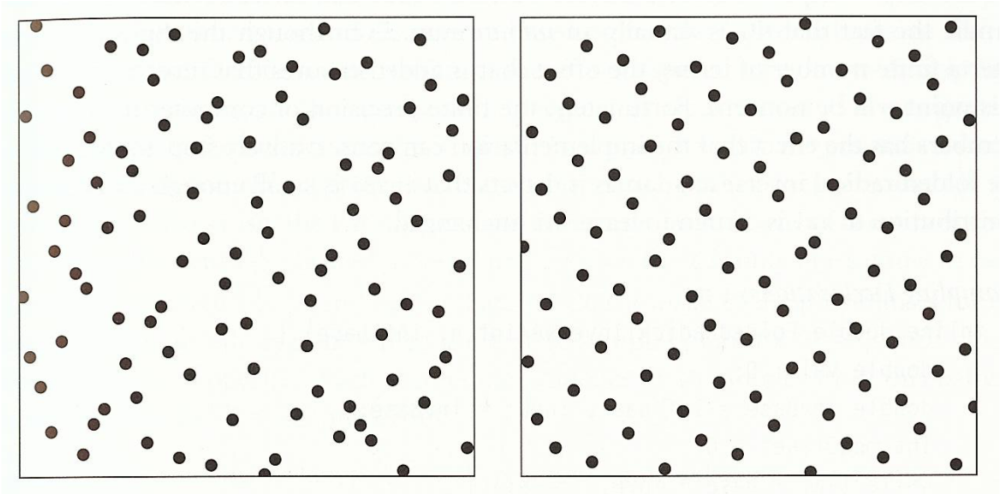
i	Base 2	$\Phi_2(i)$
1	1	.1 = 1/2
2	10	.01 = 1/4
3	11	.11 = 3/4
4	100	.001 = 1/8
5	101	.101 = 5/8
⋮	⋮	⋮

KAIST

Halton and Hammersley

- **Halton**
  - $x_i=(\Phi_2(i), \Phi_3(i), \Phi_5(i), \ldots, \Phi_{prime}(i))$
- **Hammersley**
  - $x_i=(1/N, \Phi_2(i), \Phi_3(i), \Phi_5(i), \ldots, \Phi_{prime}(i))$
  - **Assume we know the number of samples, N**
  - **Has slightly lower discrepancy**

Halton



Hammersley

Why Use Quasi Monte Carlo?

- No randomness
- Much better than pure Monte Carlo method
- Converge as fast as stratified sampling

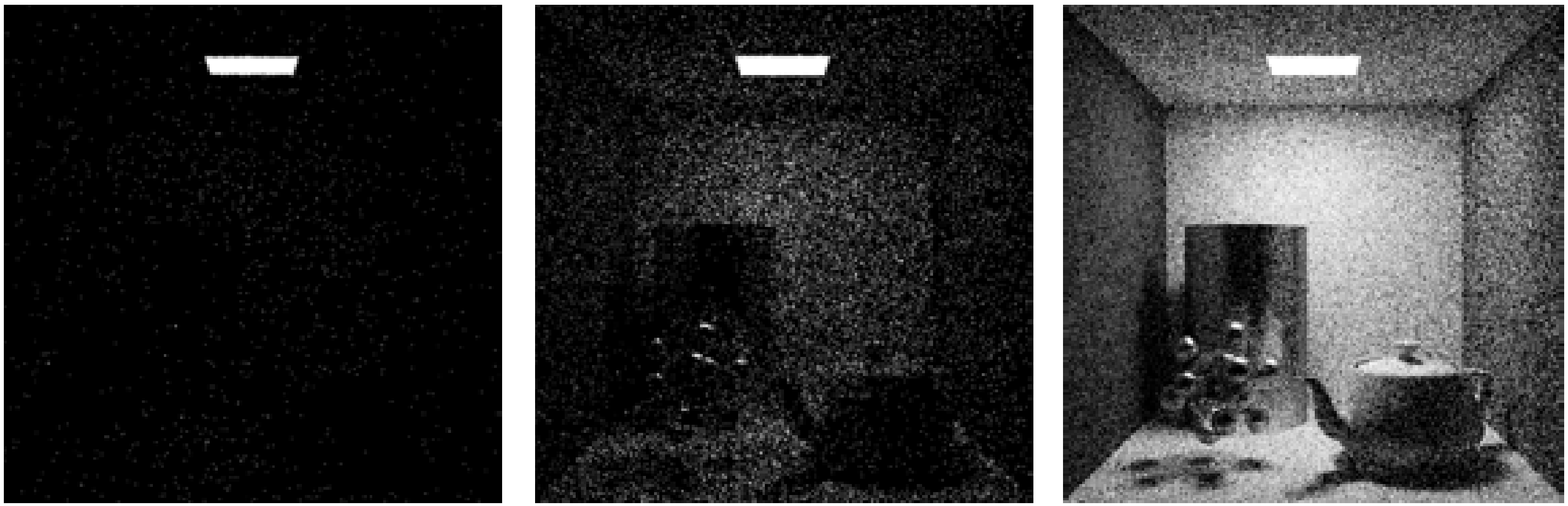
Performance and Error

- **Want better quality with smaller number of samples**
  - **Fewer samples → better performance**
  - **Stratified sampling**
  - **Quasi Monte Carlo: well-distributed samples**

- **Faster convergence**
  - **Importance sampling: next-event estimation**

Path Tracing

Sample hemisphere

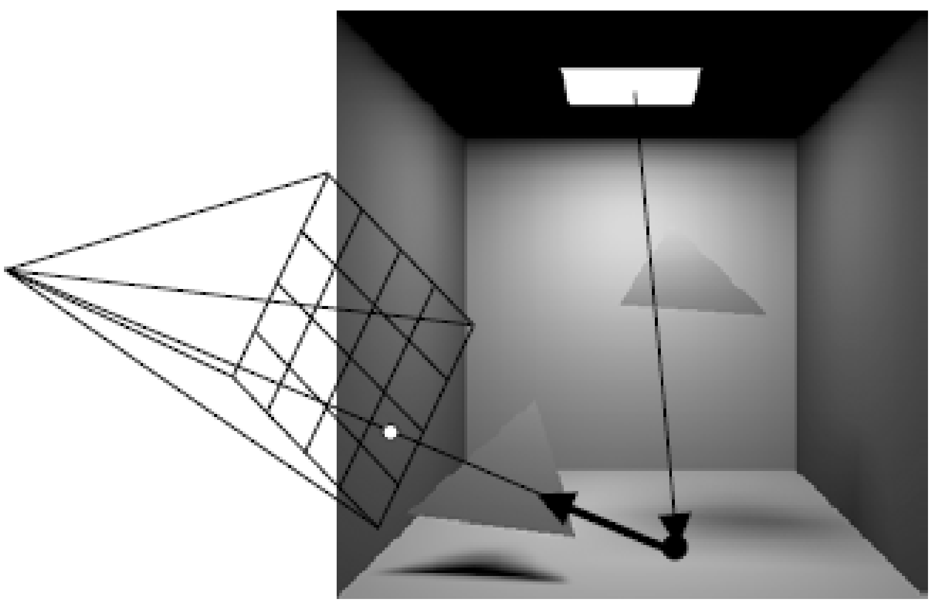


1 sample/pixel   16 samples/pixel   256 samples/pixel

- Importance Sampling: compute direct illumination separately!

Direct Illumination

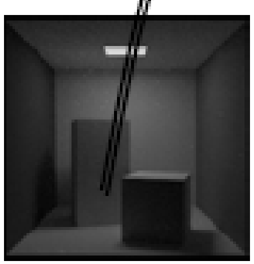
- Paths of length 1 only, between receiver and light source



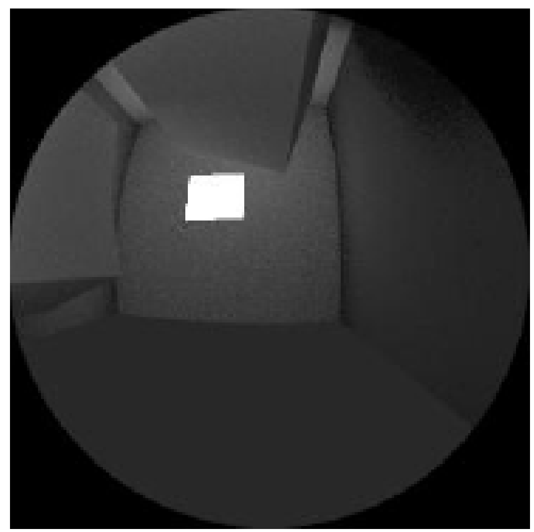
Importance Sampling

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

Radiance from light sources + radiance from other surfaces



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

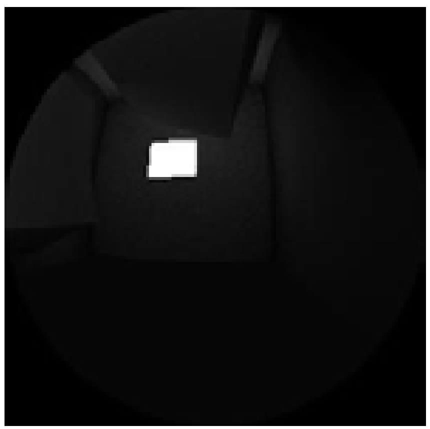


$$\cdot f_r \cdot \cos$$

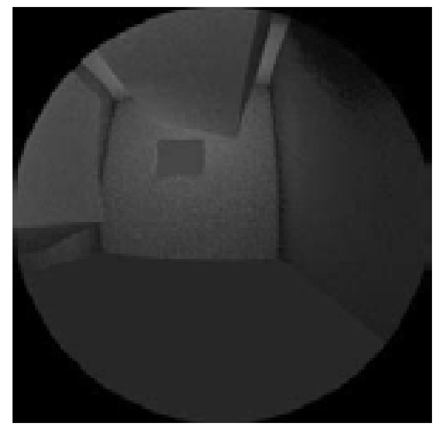
Importance Sampling

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

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



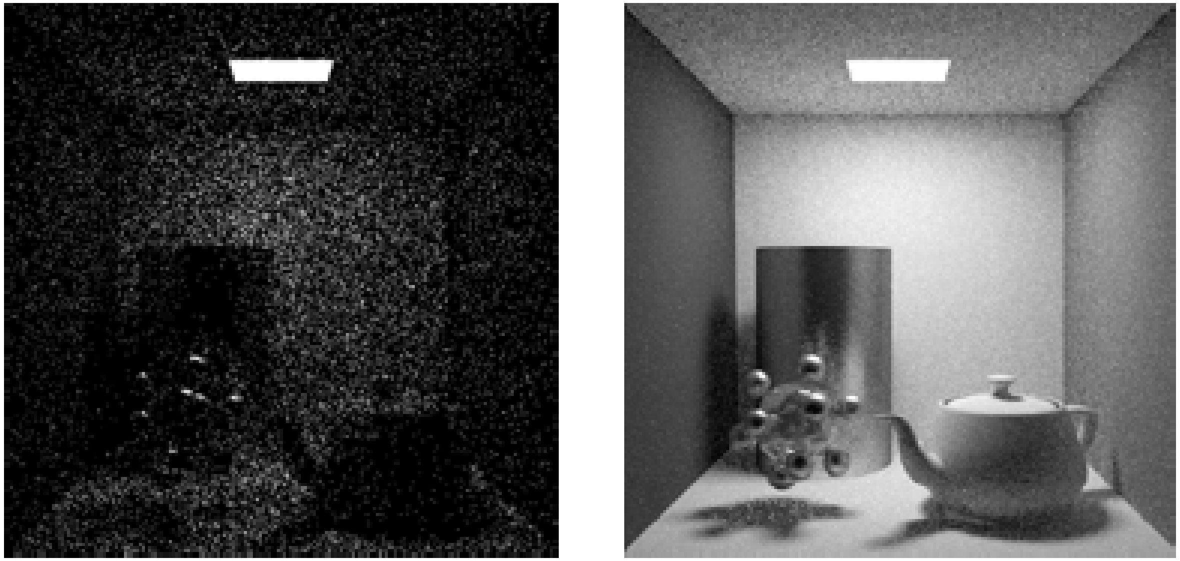
$$\cdot f_r \cdot \cos + \int_{\Omega_x}$$



$$\cdot f_r \cdot \cos$$

- So … sample direct and indirect with separate MC integration

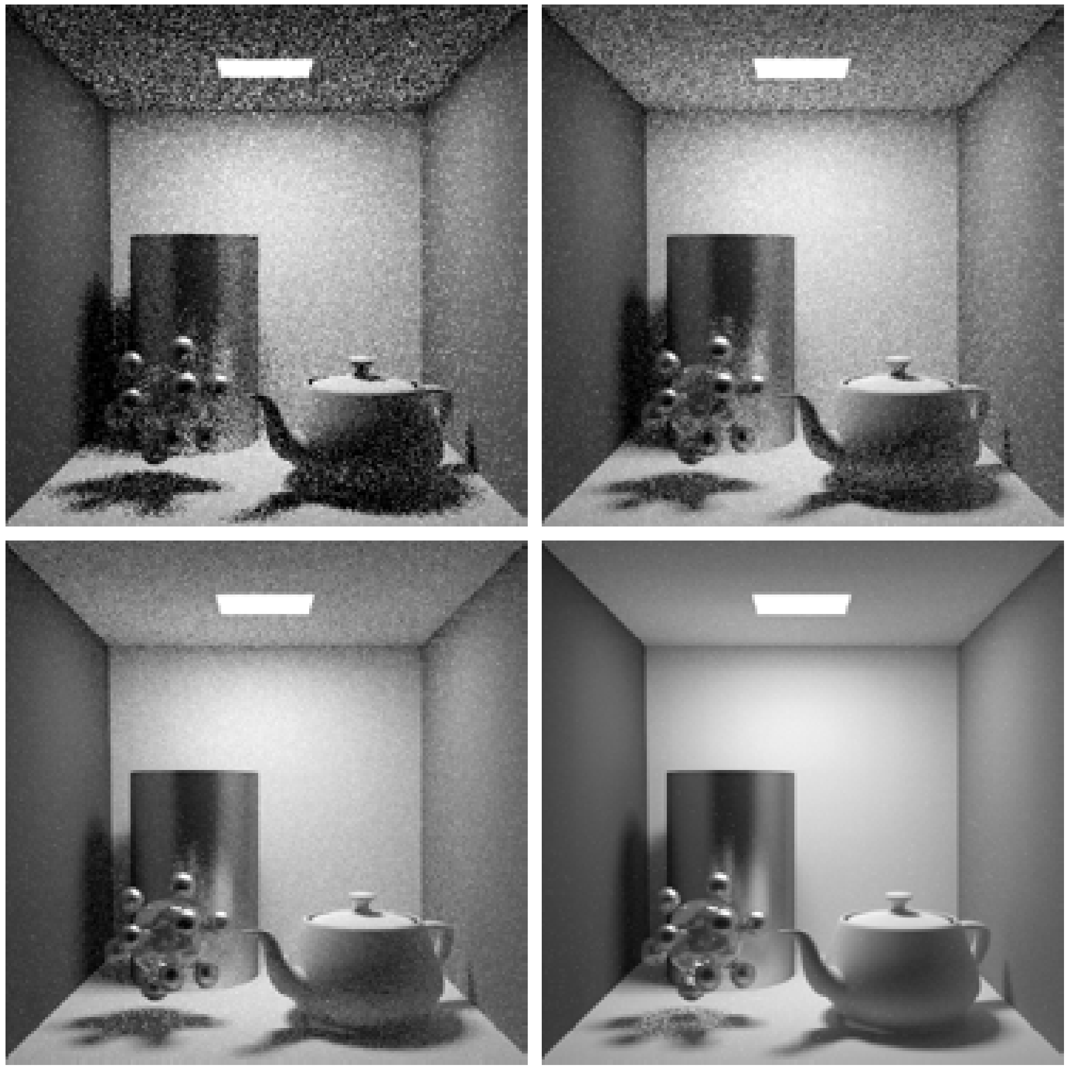
Comparison



From kavita's slides

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

Rays per pixel



1 sample/ pixel

4 samples/ pixel

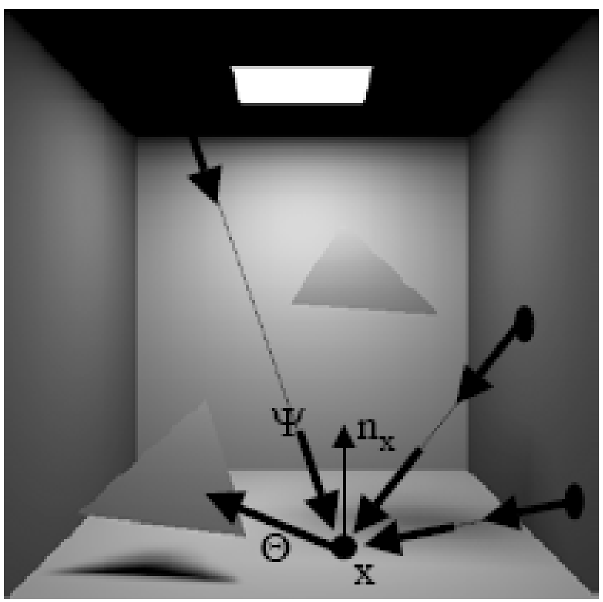
16 samples/ pixel

256 samples/ pixel

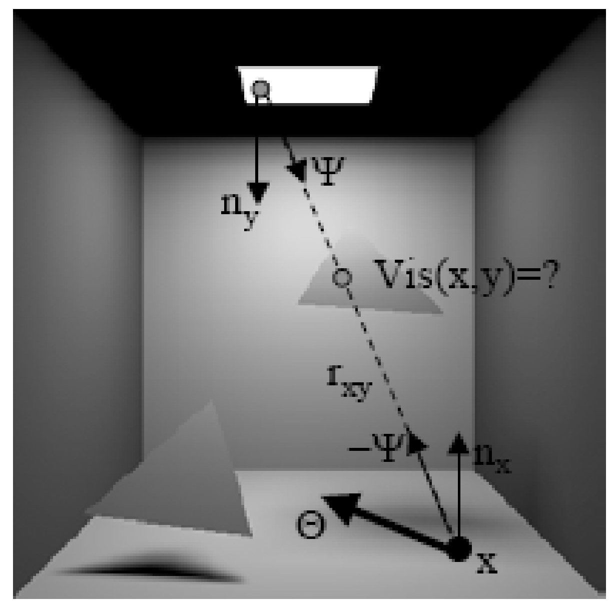
Direct Illumination

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

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



hemisphere integration



area integration

Estimator for direct lighting

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

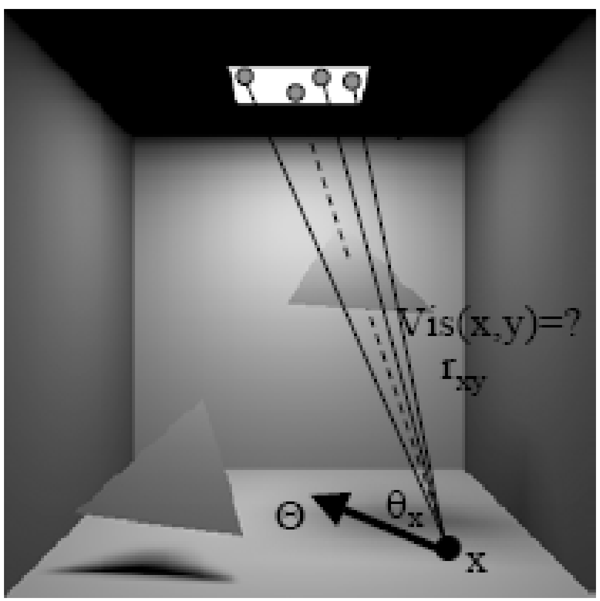
$$p(y)$$

- 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



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

PDF for sampling light

- Uniform

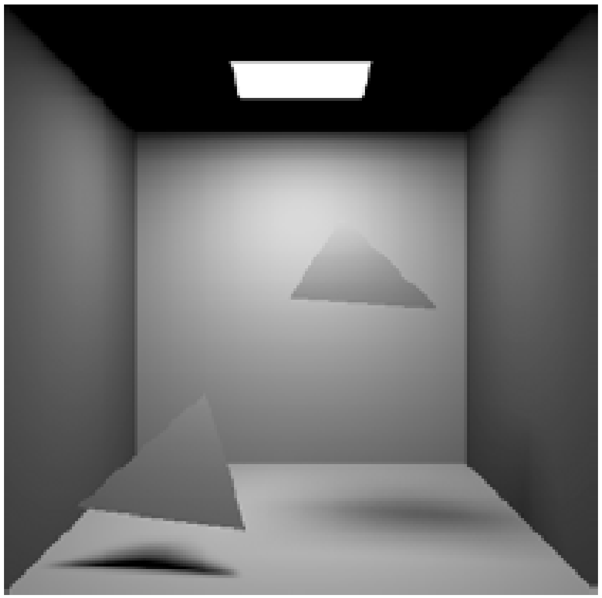
$$p(y) = \frac{1}{Area_{source}}$$

- Pick a point uniformly over light's area
  - Can stratify samples

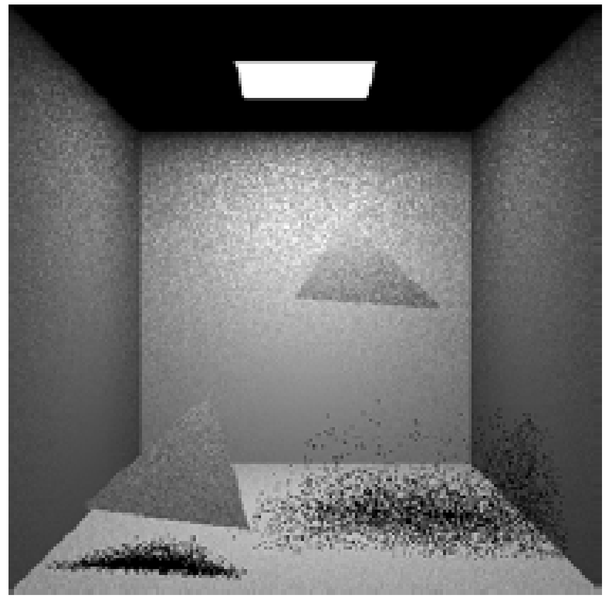
- Estimator:

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

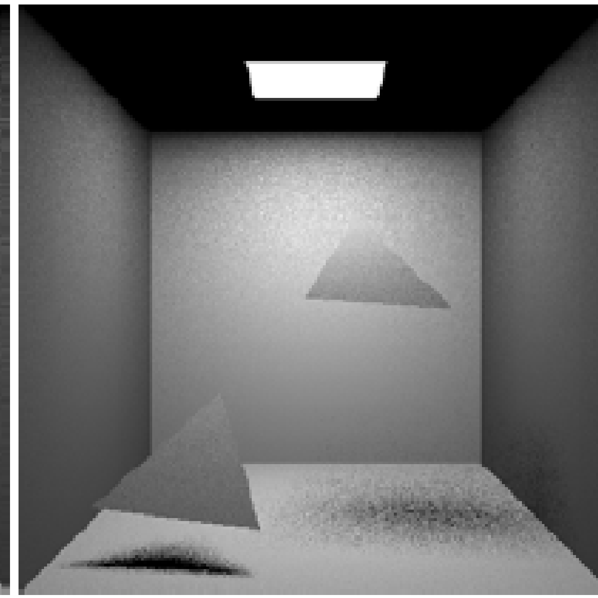
More points ...



1 shadow ray

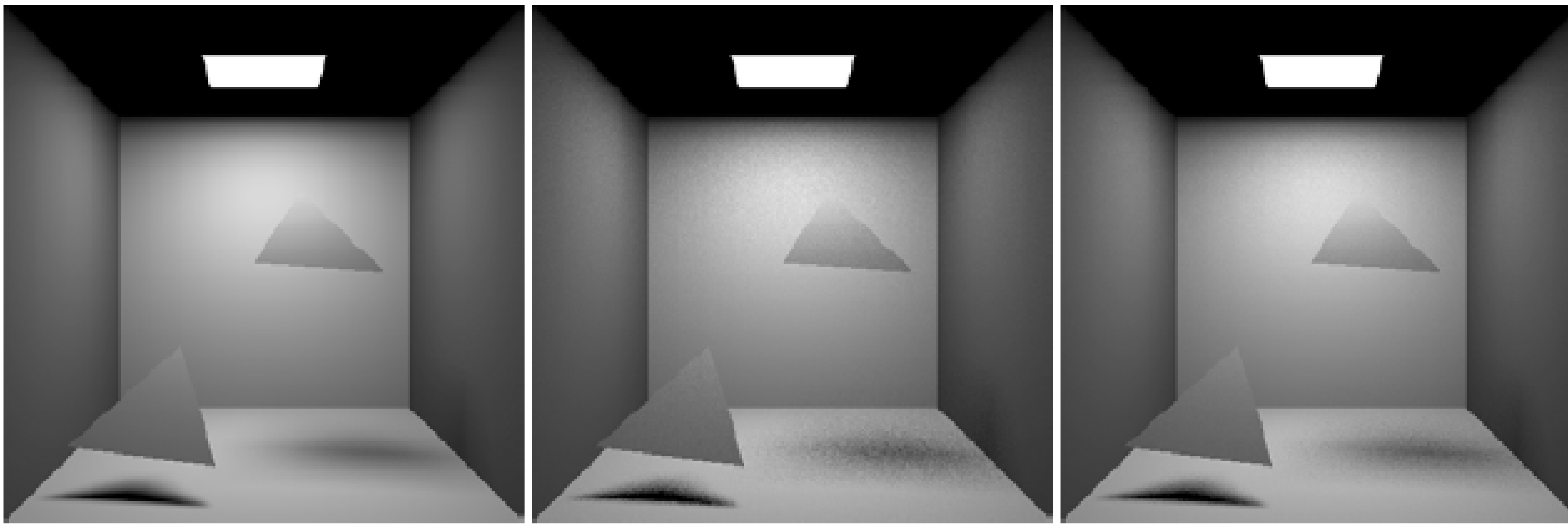


9 shadow rays



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

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

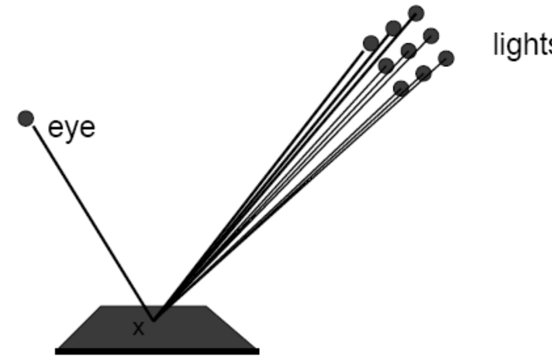
$$E(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r L_{source} \frac{\cos\theta_x \cos\theta_{\bar{y}_i}}{r^2_{x\bar{y}_i}} Vis(x, \bar{y}_i)}{p(\bar{y}_i)}$$

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, …)

Scenes with many lights

- Many lights in scenes: M lights

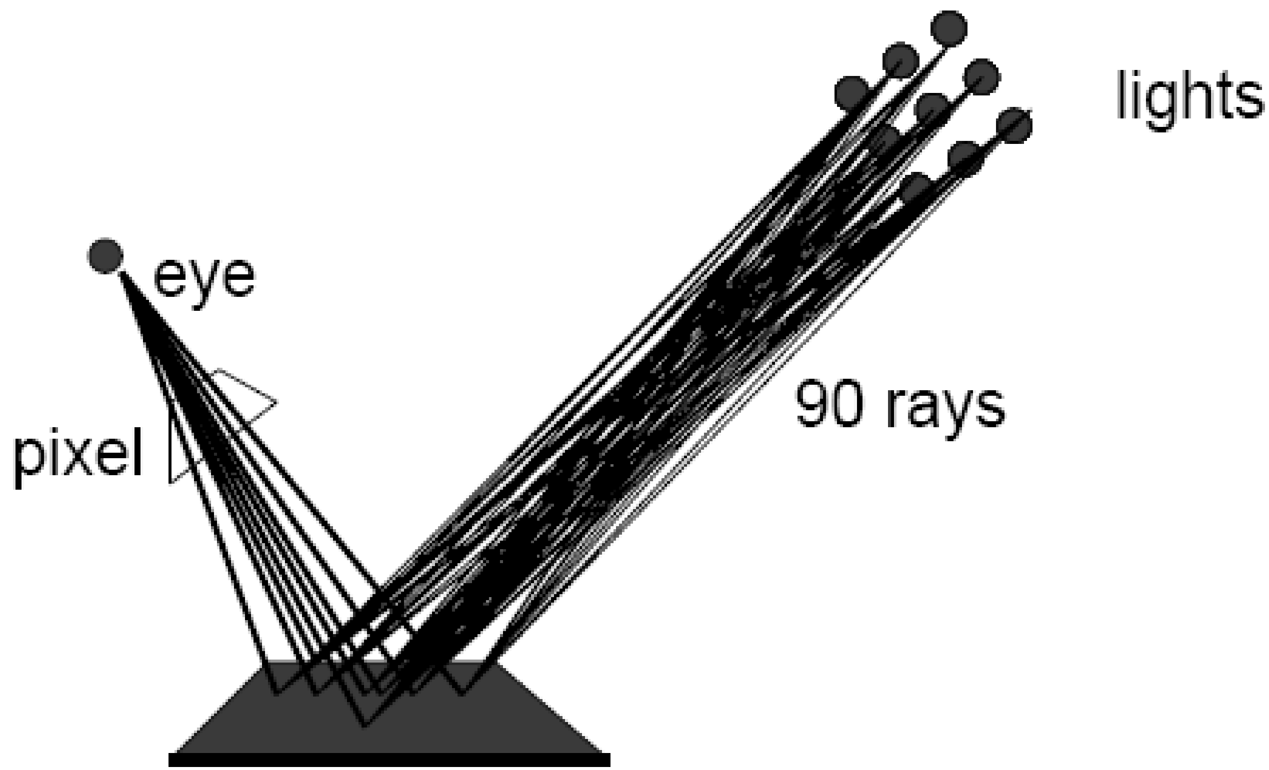


- How to handle many lights?
- Formulation 1: M integrals, one per light
  - Same solution technique as earlier for each light

$$L(x \rightarrow \Theta) = \sum_{i=1}^{M} \int_{A_{source}} f_r(x, -\Psi \leftrightarrow \Theta) \cdot L_{source}(y \rightarrow -\Psi) \cdot G(x, y) \cdot dA_y$$

Antialiasing: pixel

- Anti-aliasing: k M N



Formulation over all lights

- When M is large, each direct lighting sample is very expensive
- We would like to importance sample the lights
- Instead of M integrals

$$L(x \to \Theta) = \sum_{i=1}^{M} \int_{A_{source}} f_r(x, -\Psi \leftrightarrow \Theta) \cdot L_{source}(y \to -\Psi) \cdot G(x, y) \cdot dA_y$$

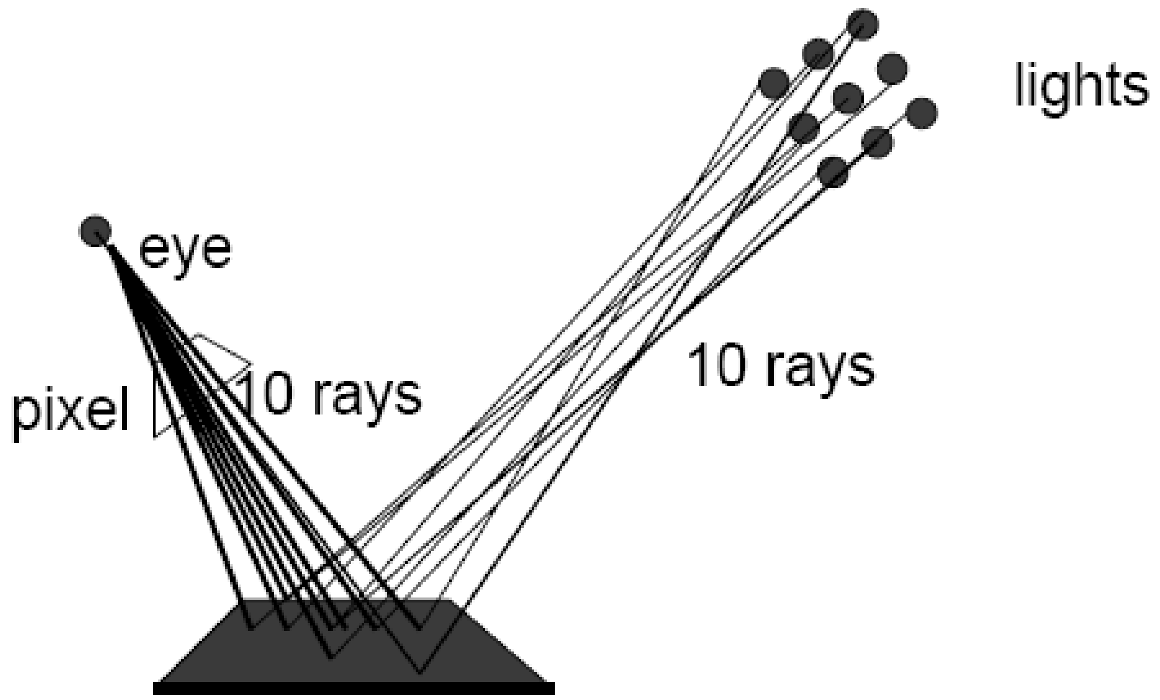
- Formulation over 1 integration domain

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

Why?

- Do not need a minimum of M rays/sample
- Can use only one ray/sample
- Still need N samples, but 1 ray/sample
- Ray is distributed over the whole integration domain
  - Can importance sample the lights

Anti-aliasing



From kavita’s slides

How to sample the lights?

- A discrete pdf $p_L(k_i)$ picks the light k_i
- A surface point is then picked with pdf $p(y_i|k_i)$
- Estimator with N samples:

$$E(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{f_r L_{source} G(x, \bar{y}_i)}{p_L(k_i)p(y_i \mid k_i)}$$

Strategies for picking light

– Uniform $p_L(k) = \frac{1}{M}$

– Area $p_L(k) = \frac{A_k}{\sum A_k}$

– Power $p_L(k) = \frac{P_k}{\sum P_k}$

Do not take visibility into account!

Research on Many Lights

- **Ward 91**
  - **Sort lights based on their maximum contribution**
  - **Pick bright lights based on a threshold**
  - **Do not consider visibility**
- **Many other papers**
- **Look at our reading list**

KAIST

Direct paths

- Different path generators produce different estimators and different error characteristics
- Direct illumination general algorithm:

```
compute_radiance (point, direction)
        est_rad = 0;
        for (i=0; i<n; i++)
                p = generate_path;
                est_rad += energy_transfer(p) / probability(p);
        est_rad = est_rad / n;
        return(est_rad);
```

Stochastic Ray Tracing

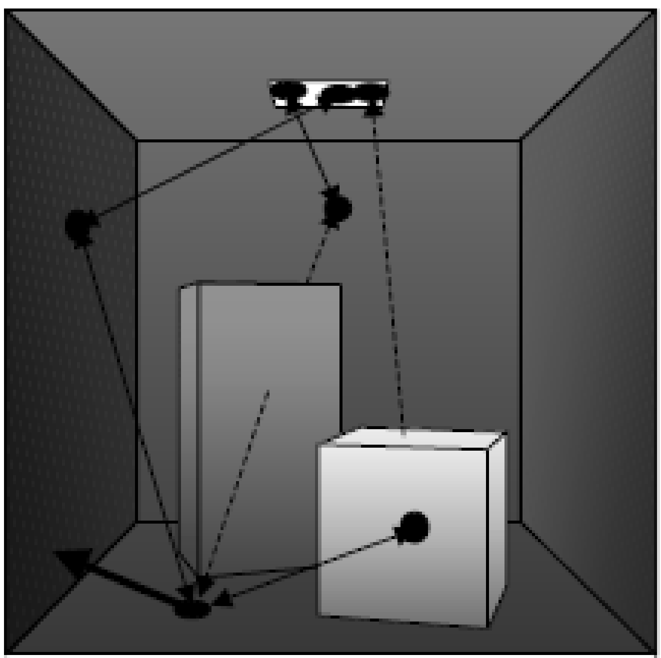
- Sample area of light source for direct term
- Sample hemisphere with random rays for indirect term
- Optimizations:
  - Stratified sampling
  - Importance sampling
  - Combine multiple probability density functions into a single PDF

Indirect Illumination

- Paths of length > 1
- Many different path generators possible
- Efficiency depends on:
  - BRDFs along the path
  - Visibility function
  - ...

Indirect paths - surface sampling

- Simple generator (path length = 2):
  - select point on light source
  - select random point on surfaces



- per path:
  - 2 visibility checks

Indirect paths - surface sampling

- Indirect illumination (path length 2):

y → z → x

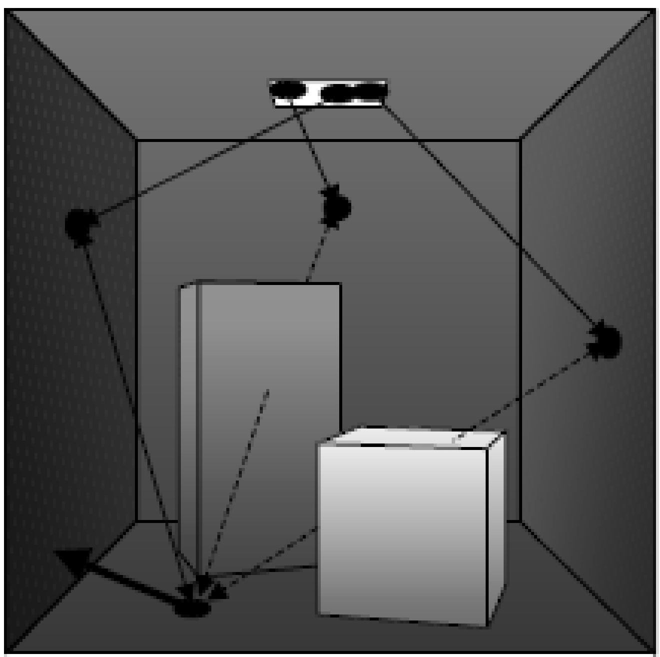
$$L(x \to \Theta) = \int_{A_{source}} \int_{A} L(y \to \Psi_1) f_r(z, -\Psi_1 \leftrightarrow \Psi_2) G(z, y) f_r(x, -\Psi_2 \leftrightarrow \Theta) G(z, x) dA_z dA_y$$

$$\langle L(x \to \Theta) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{L(y_i \to \Psi_{1i}) f_r(z_i, -\Psi_{1i} \leftrightarrow \Psi_{2i}) G(z_i, y_i) f_r(x, -\Psi_{2i} \leftrightarrow \Theta) G(z_i, x)}{p_y(y_i) p_z(z_i)}$$

- 2 visibility values cause noise
  - which might be 0

Indirect paths - source shooting

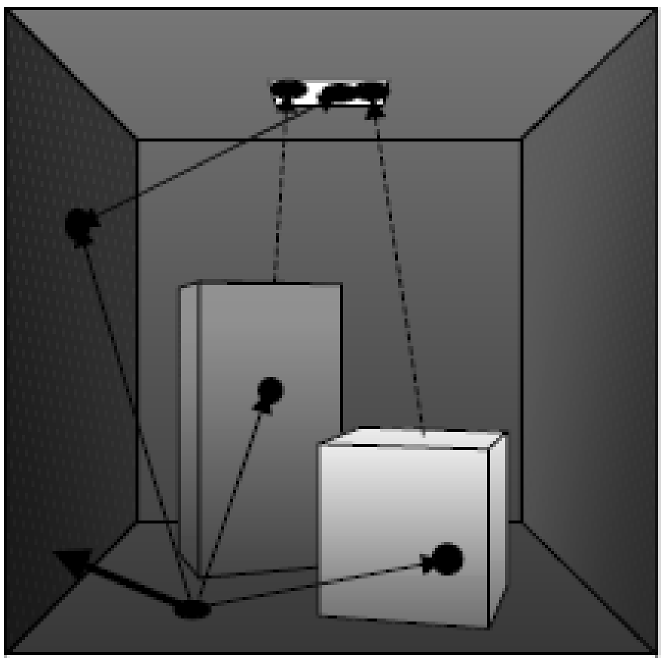
- Shoot ray from light source, find hit location
- Connect hit point to receiver



– per path:
  - 1 ray intersection
  - 1 visibility check

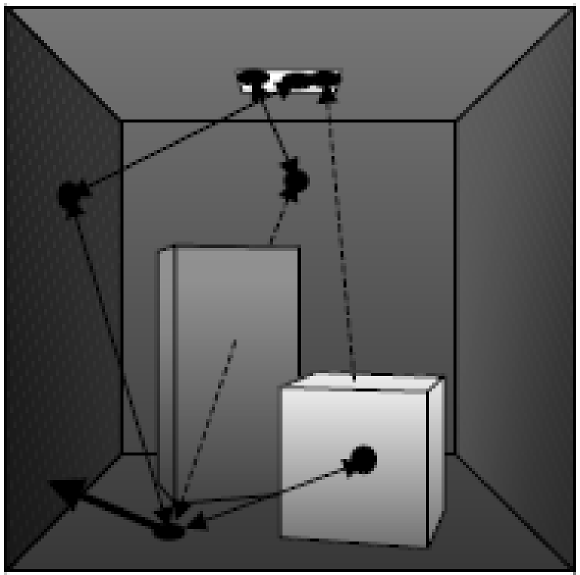
Indirect paths - receiver gathering

- Shoot ray from receiver point, find hit location
- Connect hit point to random point on light source



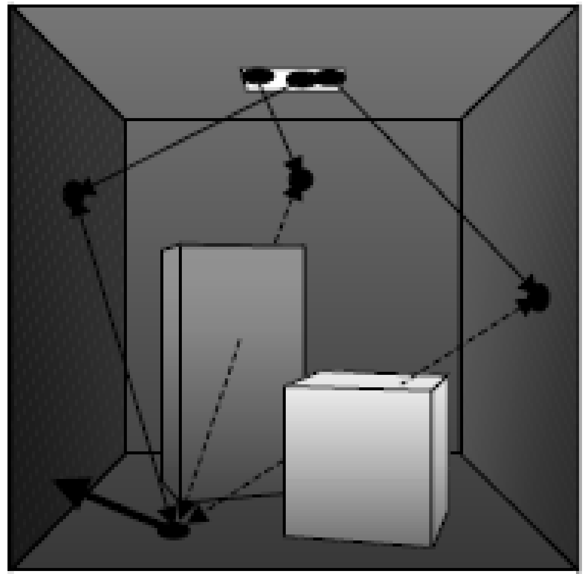
– per path:
  - 1 ray intersection
  - 1 visibility check

Indirect paths



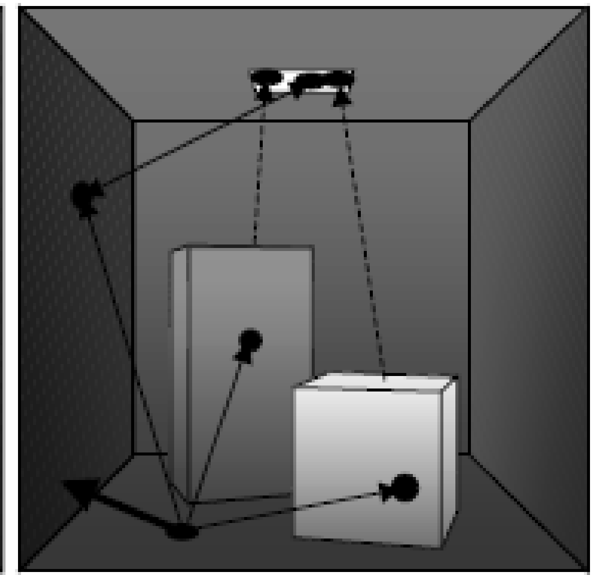
Surface sampling

- 2 visibility terms; can be 0



Source shooting

- 1 visibility term
- 1 ray intersection

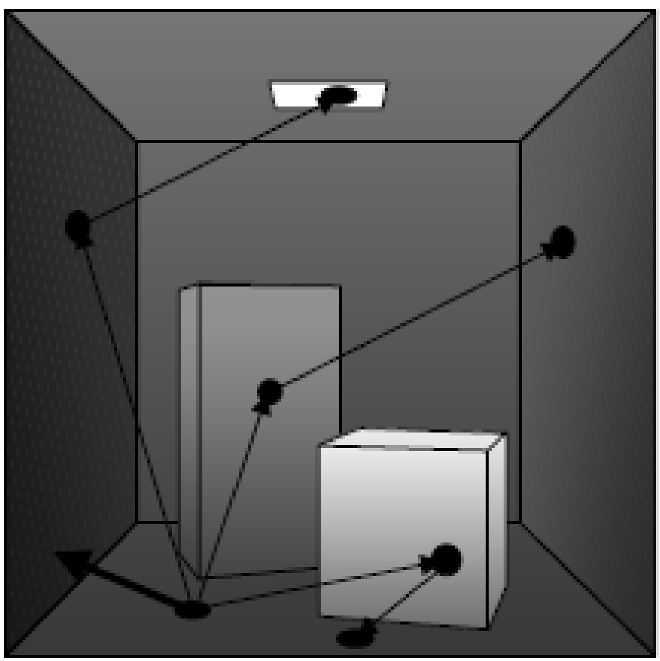


Receiver gathering

- 1 visibility term
- 1 ray intersection

More variants …

- Shoot ray from receiver point, find hit location
- Shoot ray from hit point, check if on light source



– per path:
  - 2 ray intersections
  - L_e might be zero

Indirect paths

- Same principles apply to paths of length > 2
  - generate multiple surface points
  - generate multiple bounces from light sources and connect to receiver
  - generate multiple bounces from receiver and connect to light sources
  - …

- Estimator and noise characteristics change with path generator

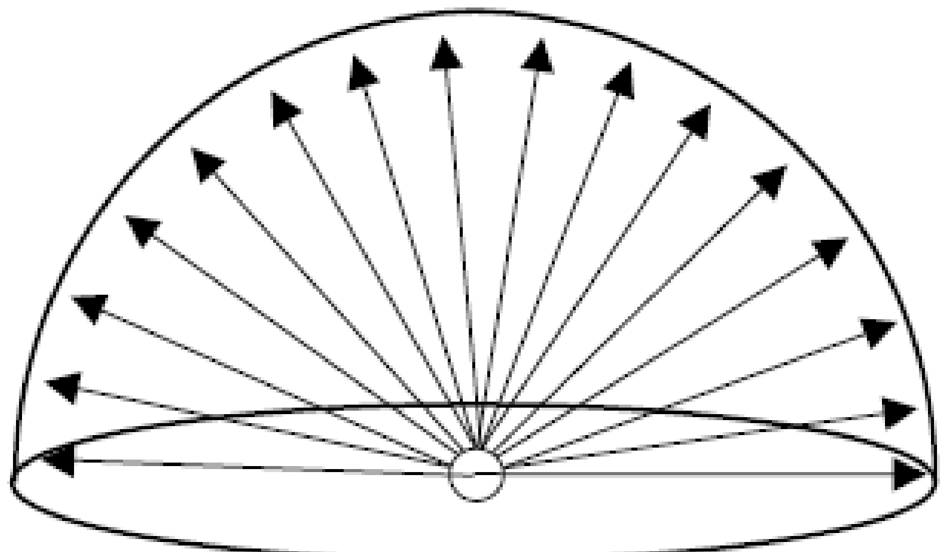
Stochastic Ray Tracing

- Sample area of light source for direct term
- Sample hemisphere with random rays for indirect term
- Optimizations:
  - Stratified sampling
  - Importance sampling
  - Combine multiple probability density functions into a single PDF

Sampling strategies

- Uniform sampling over the hemisphere

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

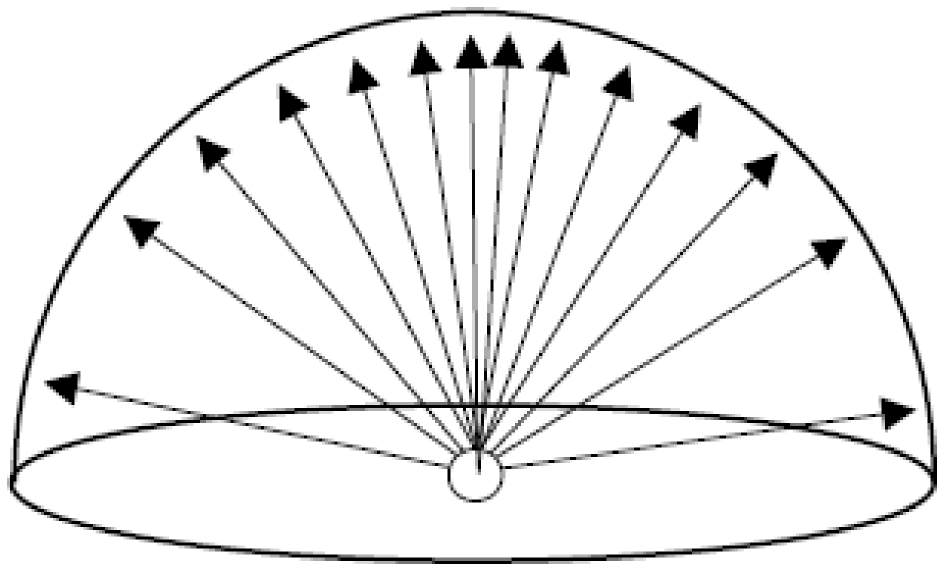


$$p(\Theta) = 1/(2\pi)$$

Sampling strategies

- Sampling according to the cosine factor

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

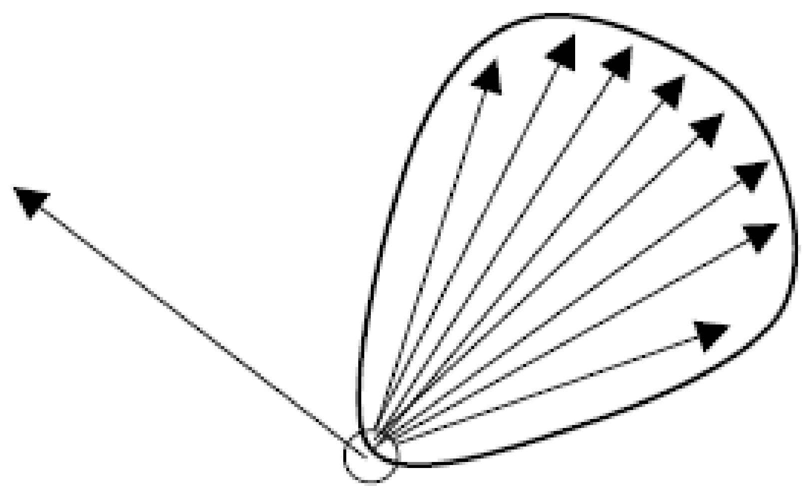


$$p(\Theta) = \cos\theta / \pi$$

Sampling strategies

- Sampling according to the BRDF

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

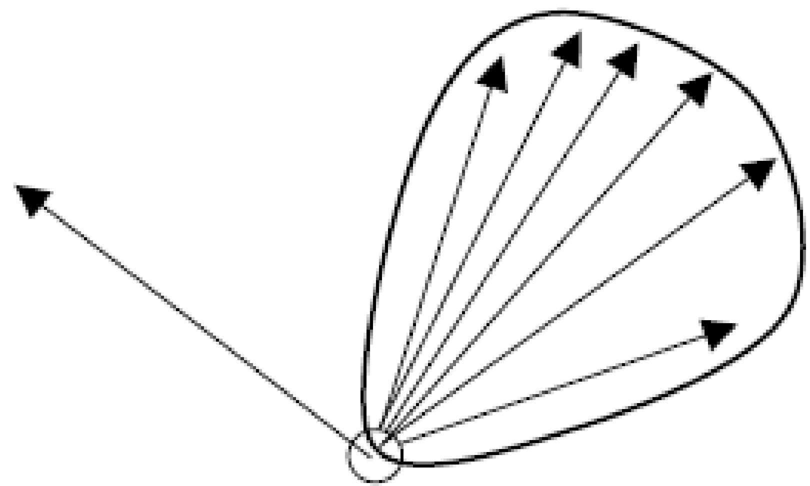


$$p(\Theta) \sim f_r(\Theta \leftrightarrow \Psi)$$

Sampling strategies

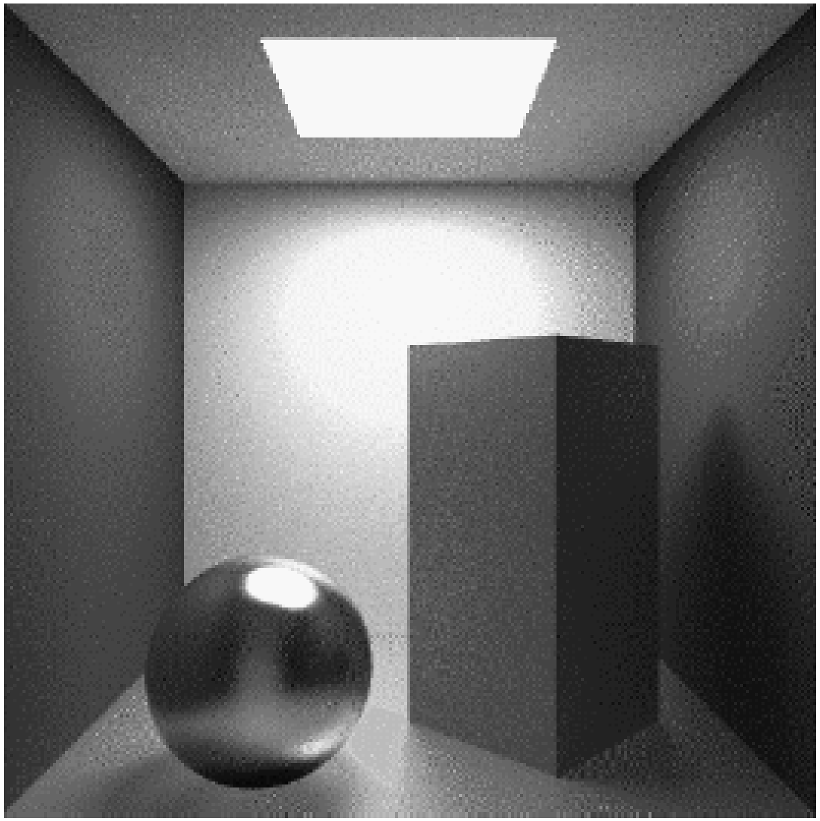
- Sampling according to the BRDF times the cosine

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

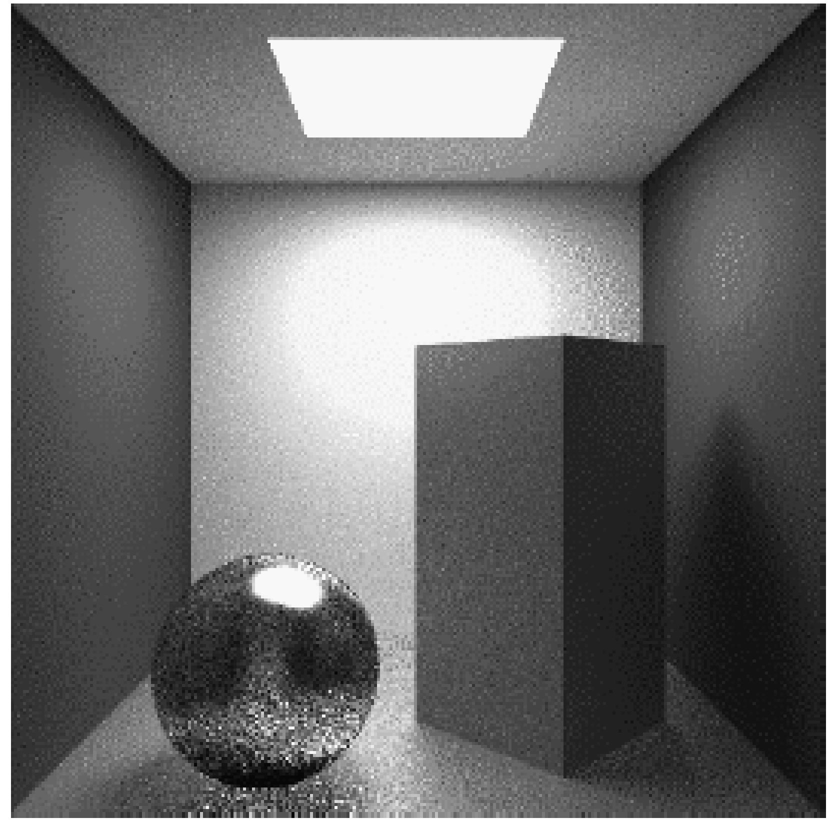


$$p(\Theta) \sim f_r(\Theta \leftrightarrow \Psi) \cos\theta$$

Comparison



With importance sampling
(brdf on sphere)



Without importance sampling

General GI Algorithm

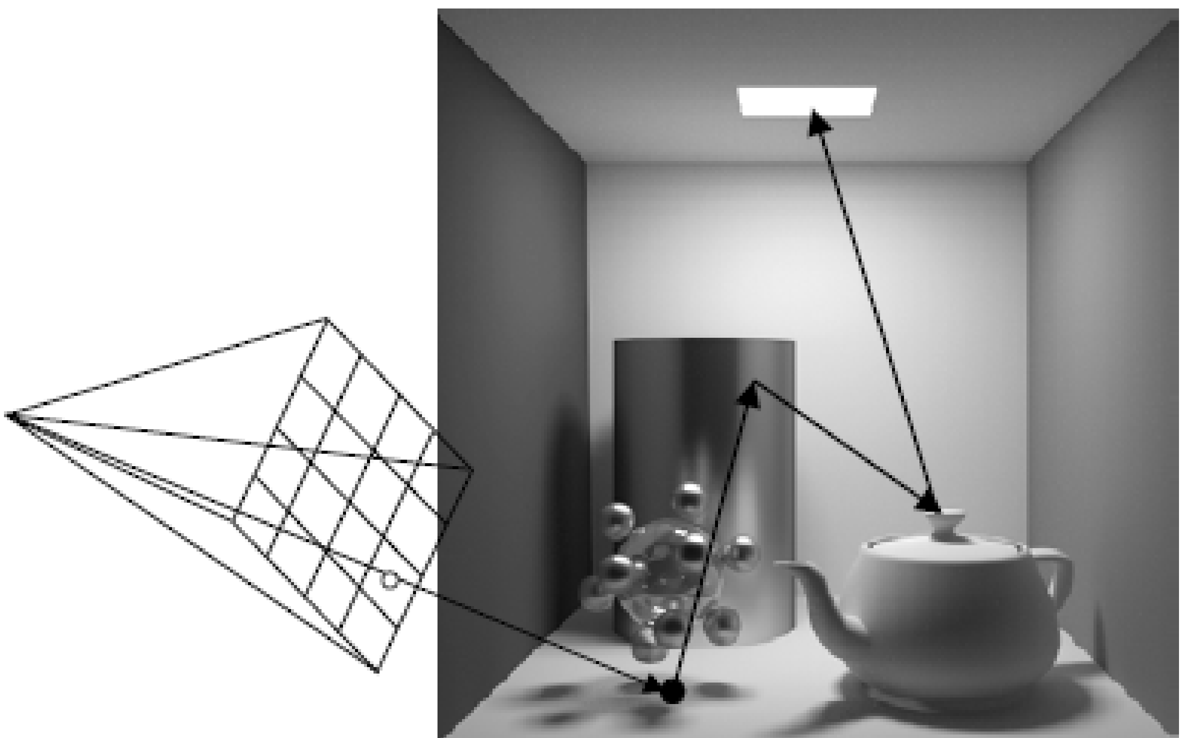
- **Design path generators**
- **Path generators determine efficiency of GI algorithm**
- **Black boxes**
  - **Evaluate BRDF, ray intersection, visibility evaluations, etc**

Other Rendering Techniques

- **Bidirectional path tracing**
- **Metropolis**
- **Biased techniques**
  - **Irradiance caching**
  - **Photon mapping**

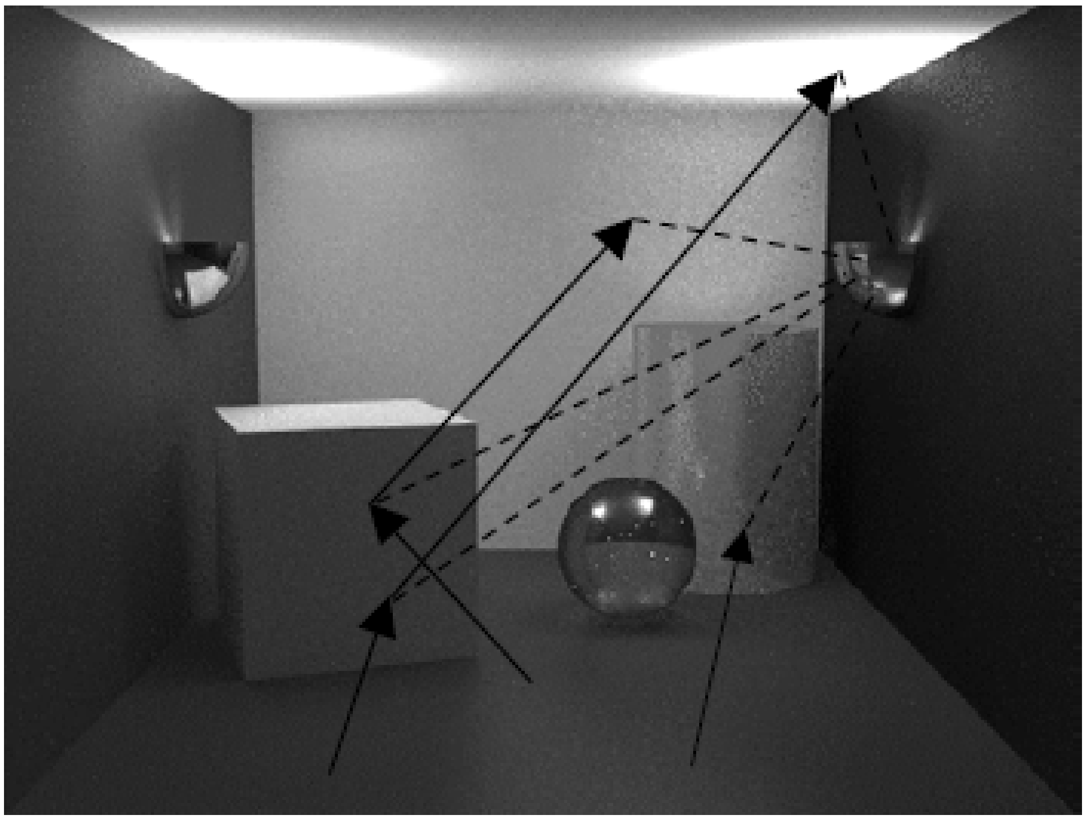
Stochastic ray tracing: limitations

- Generate a path from the eye to the light source



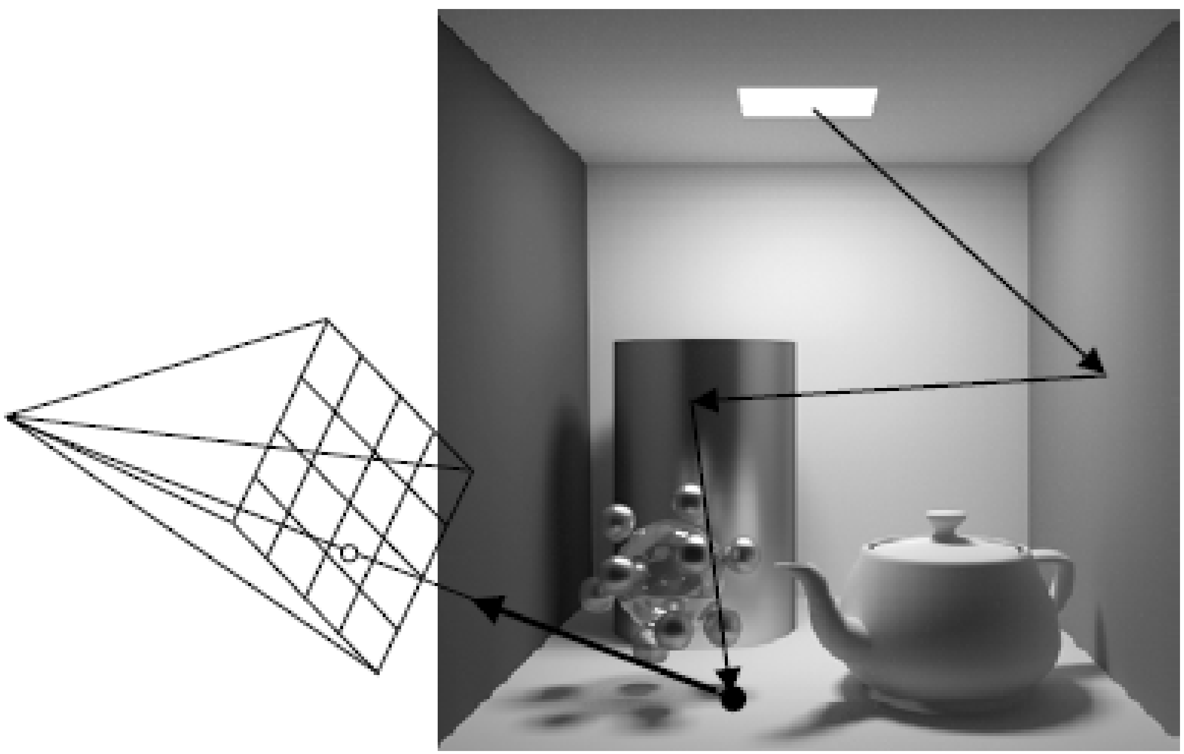
When does it not work?

- Scenes in which indirect lighting dominates



Bidirectional Path Tracing

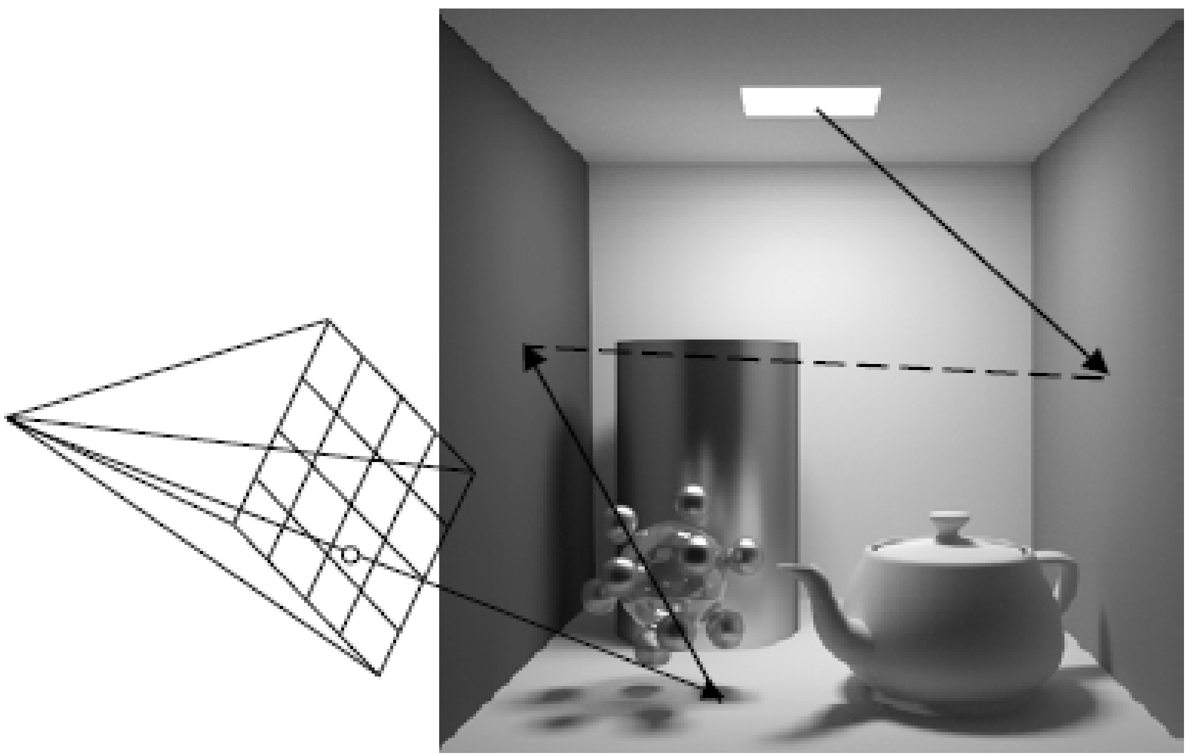
- So … we can generate paths starting from the light sources!



- Shoot ray to camera to see what pixels get contributions

Bidirectional Path Tracing

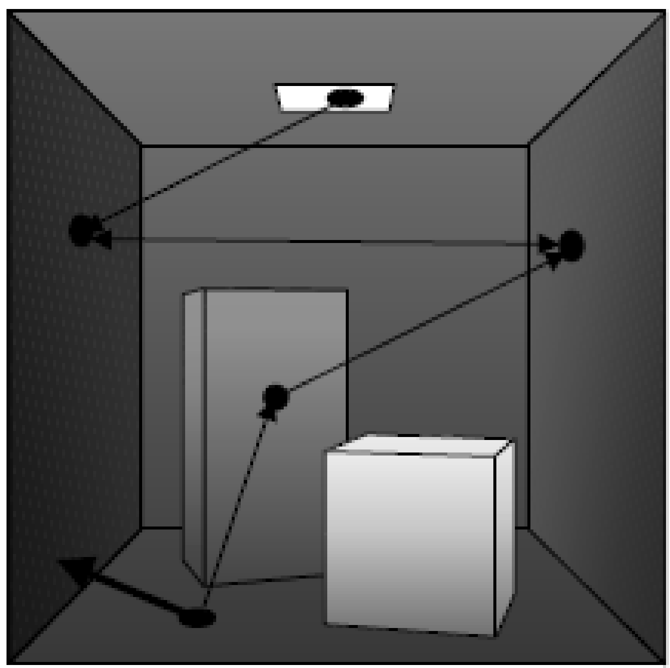
- Or paths generated from both camera and source at the same time ...!



- Connect endpoints to compute final contribution

Complex path generators

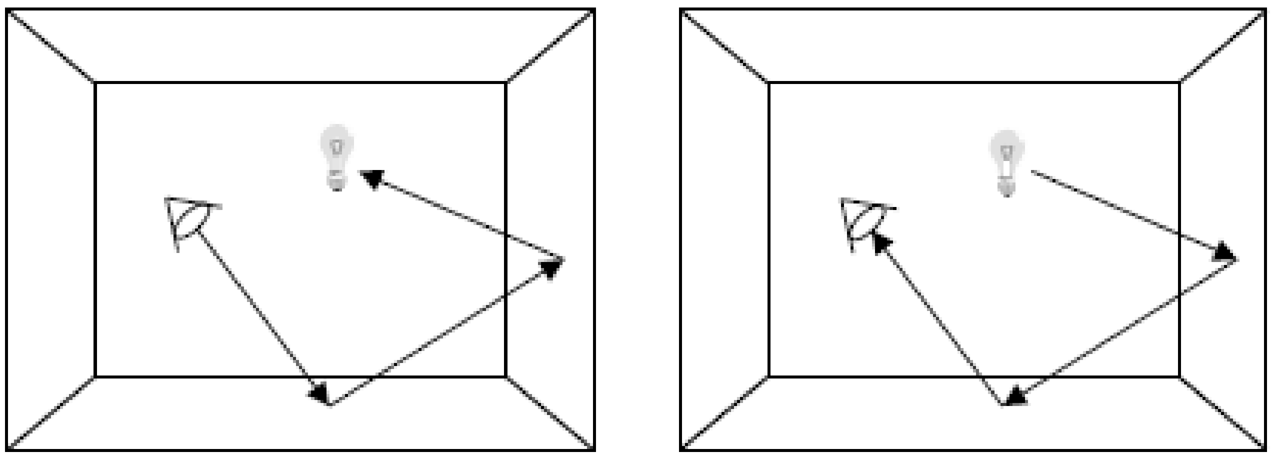
- Bidirectional ray tracing
  - shoot a path from light source
  - shoot a path from receiver
  - connect end points



Why? BRDF - Reciprocity

- Direction in which path is generated, is not important: Reciprocity

$$f(\Psi \rightarrow \Theta) = f(\Theta \rightarrow \Psi) = f(\Psi \leftrightarrow \Theta)$$



- Algorithms:
  - trace rays from the eye to the light source
  - trace rays from light source to eye
  - any combination of the above

Bidirectional ray tracing

- Parameters
  - eye path length = 0: shooting from source
  - light path length = 0: gathering at receiver
- When useful?
  - Light sources difficult to reach
  - Specific brdf evaluations (e.g., caustics)

Other Rendering Techniques

- **Metropolis**

- **Biased techniques**
  - **Irradiance caching**
  - **Photon mapping**

Metropolis

- **Based on Metropolis sampling (1950's)**
  - **Introduced by Veach and Guibas to CG**
- **Deals with hard to find light paths**
  - **Robust**
- **Hairy math, but it works**
  - **Not that easy to implement**

Metropolis

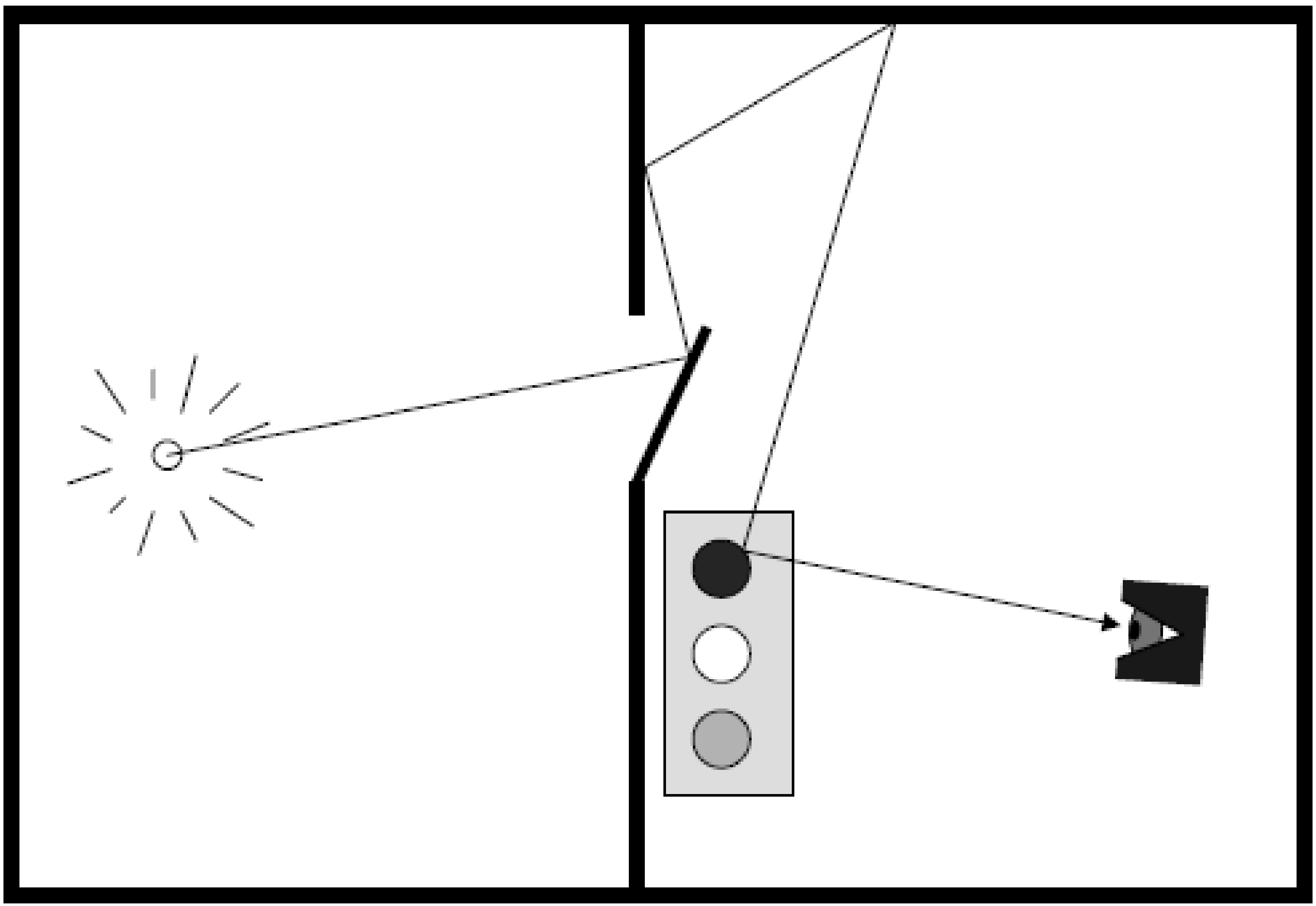
- Generate paths

- Once a valid path is found, mutate it to generate new valid paths

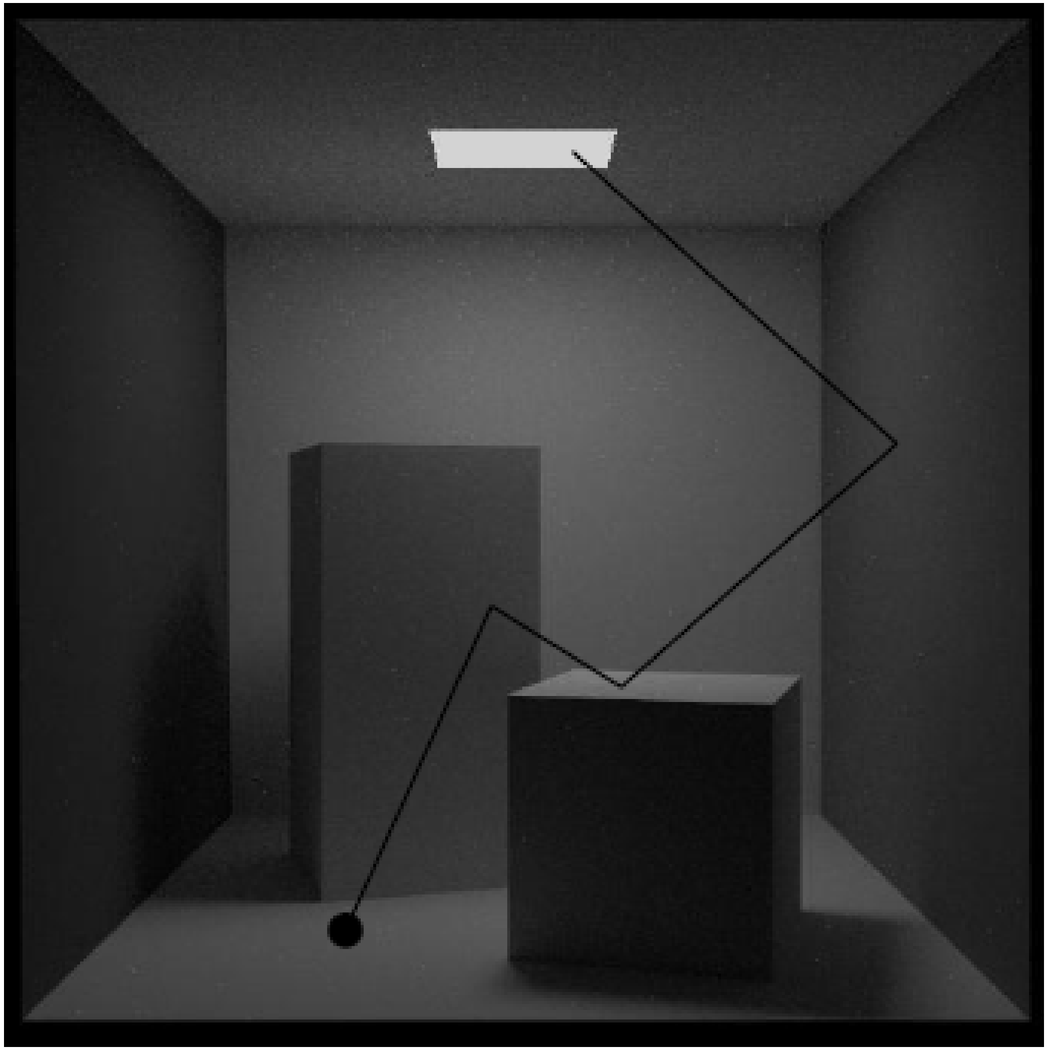
- Advantages:
  - Path re-use
  - Local exploration: found hard-to-find light distribution, mutate to find other such paths

KAIST

Metropolis

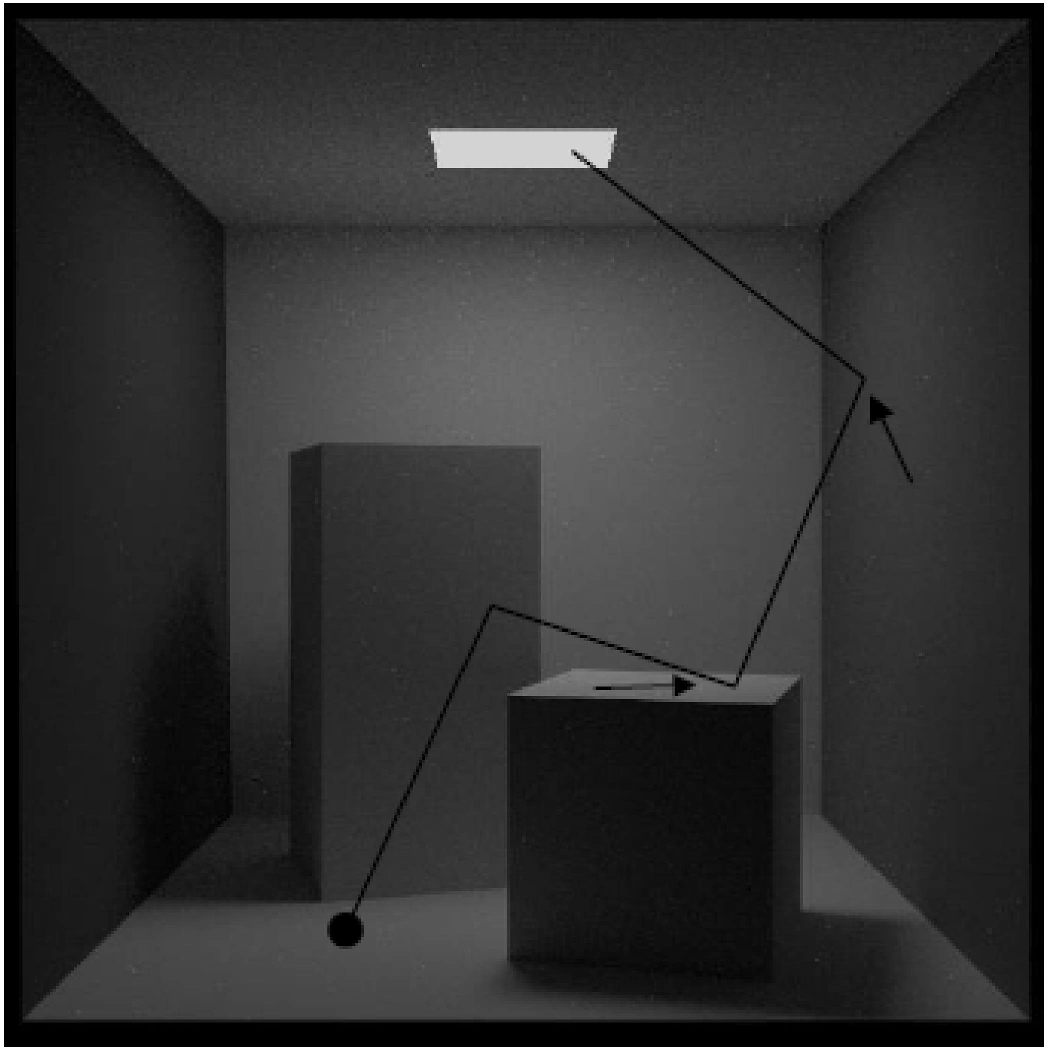


Metropolis



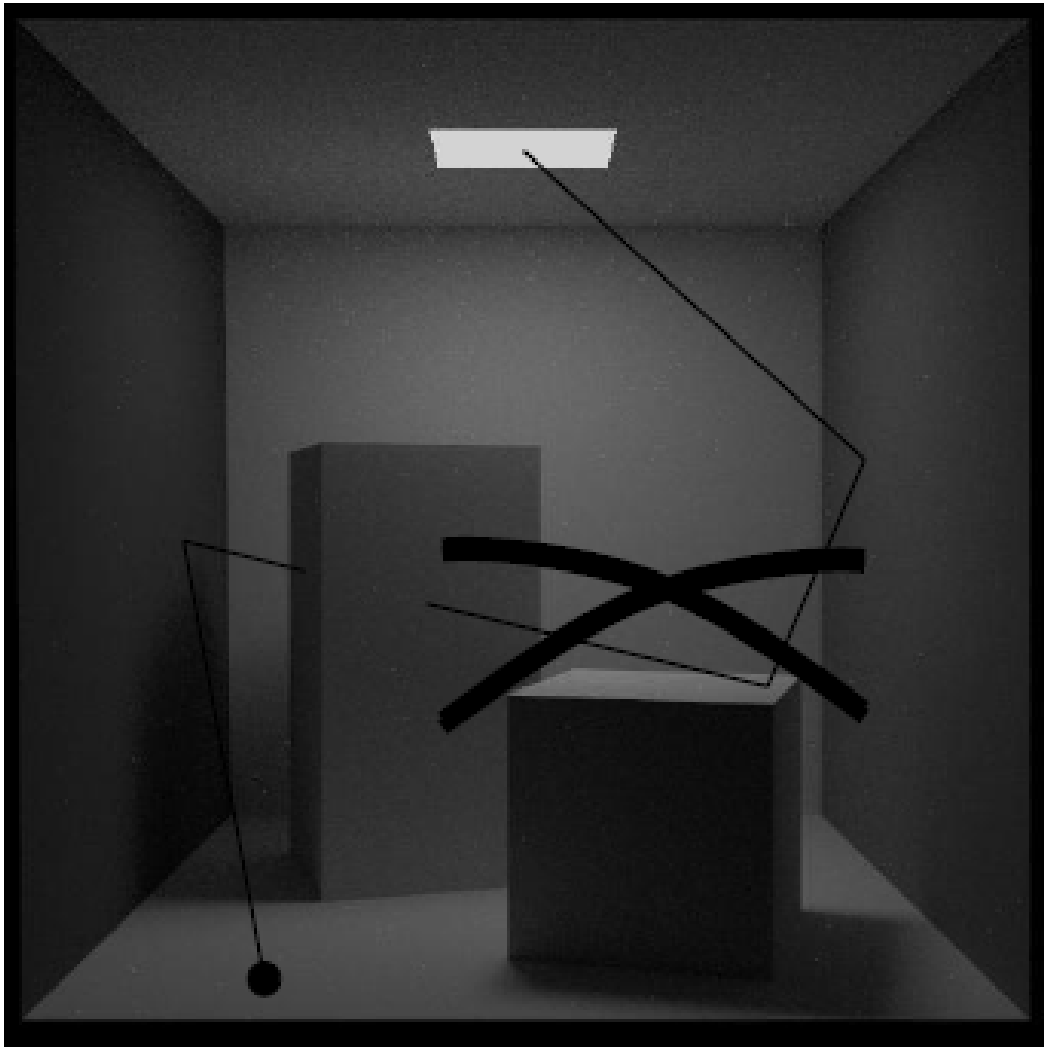
valid path

Metropolis



small perturbations

Metropolis



Accept mutations based on energy transport

Metropolis

- Advantages
  - Robust
  - Good for hard to find light paths
- Disadvantage
  - Slow convergence for many important paths
  - Tricky to implement and get right

Unbiased vs. Consistent

- **Unbiased**
  - **No systematic error**
  - **$E[I_{estimator}] = I$**
  - **Better results with larger N**

- **Consistent**
  - **Converges to correct results with more samples**
  - **$E[I_{estimator}] = I + \varepsilon$, where $\lim_{n \to \infty} \varepsilon = 0$**

Biased Methods

- MC methods
  - Too noisy and slow
  - Nose is objectionable
- Biased methods: store information (caching)
  - Irradiance caching
  - Photon mapping

Irradiance Caching

- **Introduced by Greg Ward 1988**
- **Implemented in RADIANCE**
  - **Public-domain software**

- **Exploits smoothness of irradiance**
  - **Cache and interpolate irradiance estimates**

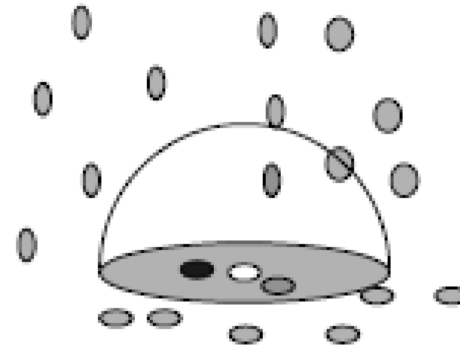
KAIST

Irradiance Caching Approach

- **Irradiance E(x) estimated using MC**
- **Cache irradiance when possible**
  - **Store in octree for fast access**
- **When do we use this cache of irradiance values?**

Smoothness Measure

- When new sample requested
  - Query octree for samples near location
  - Check ε at x, x_i is a nearby sample



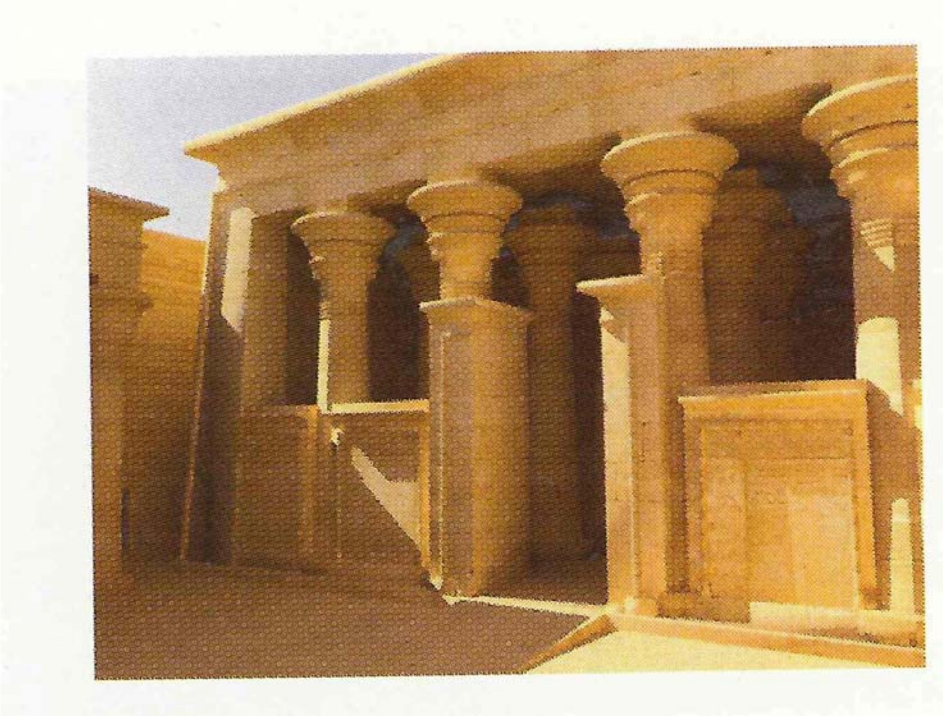
$$\varepsilon_i(x,\vec{n}) = \frac{\| x_i - x \|}{R_i} + \sqrt{1 - \vec{n} \bullet \vec{n}_i}$$

  - Weight samples inversely proportional to ε_i

$$E(x,\vec{n}) = \frac{\sum_{i, w_i > 1/a} w_i(x,\vec{n}) E_i(x_i)}{\sum_{i, w_i > 1/a} w_i(x,\vec{n})}$$

  - Otherwise, compute new sample

Irradiance Caching: Result

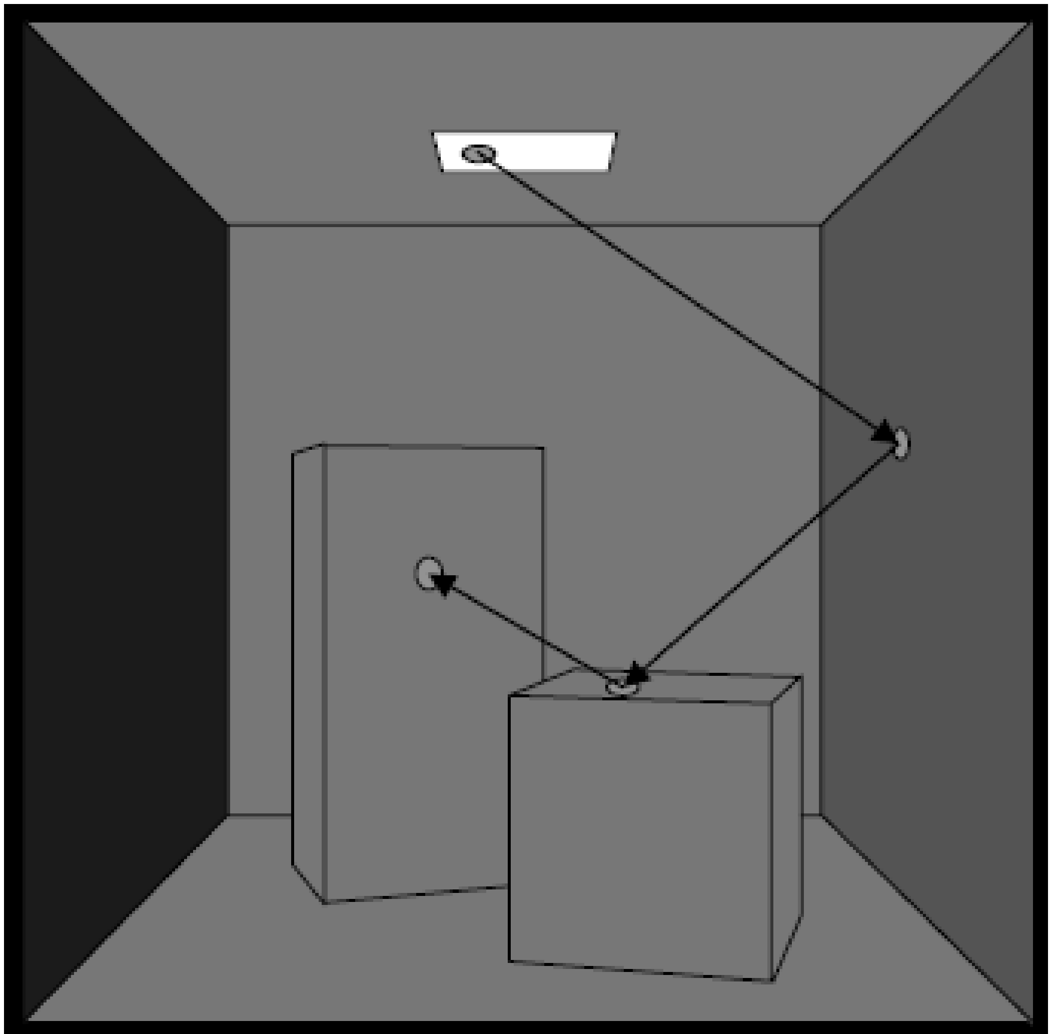


From Dutre et al.

Photon Mapping

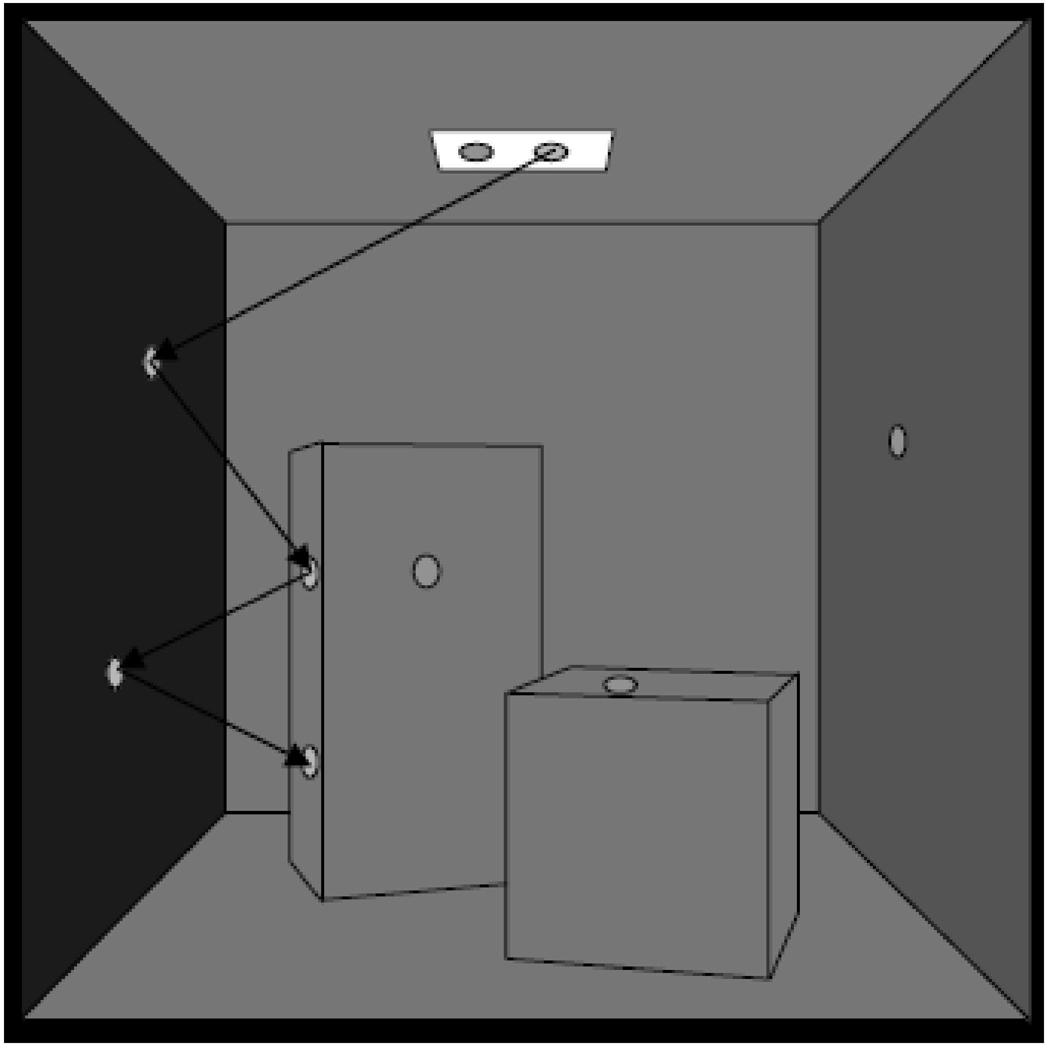
- **2 passes:**
  - **Shoot "photons" (light-rays) and record any hit-points**
  - **Shoot viewing rays and collect information from stored photons**

Pass 1: shoot photons



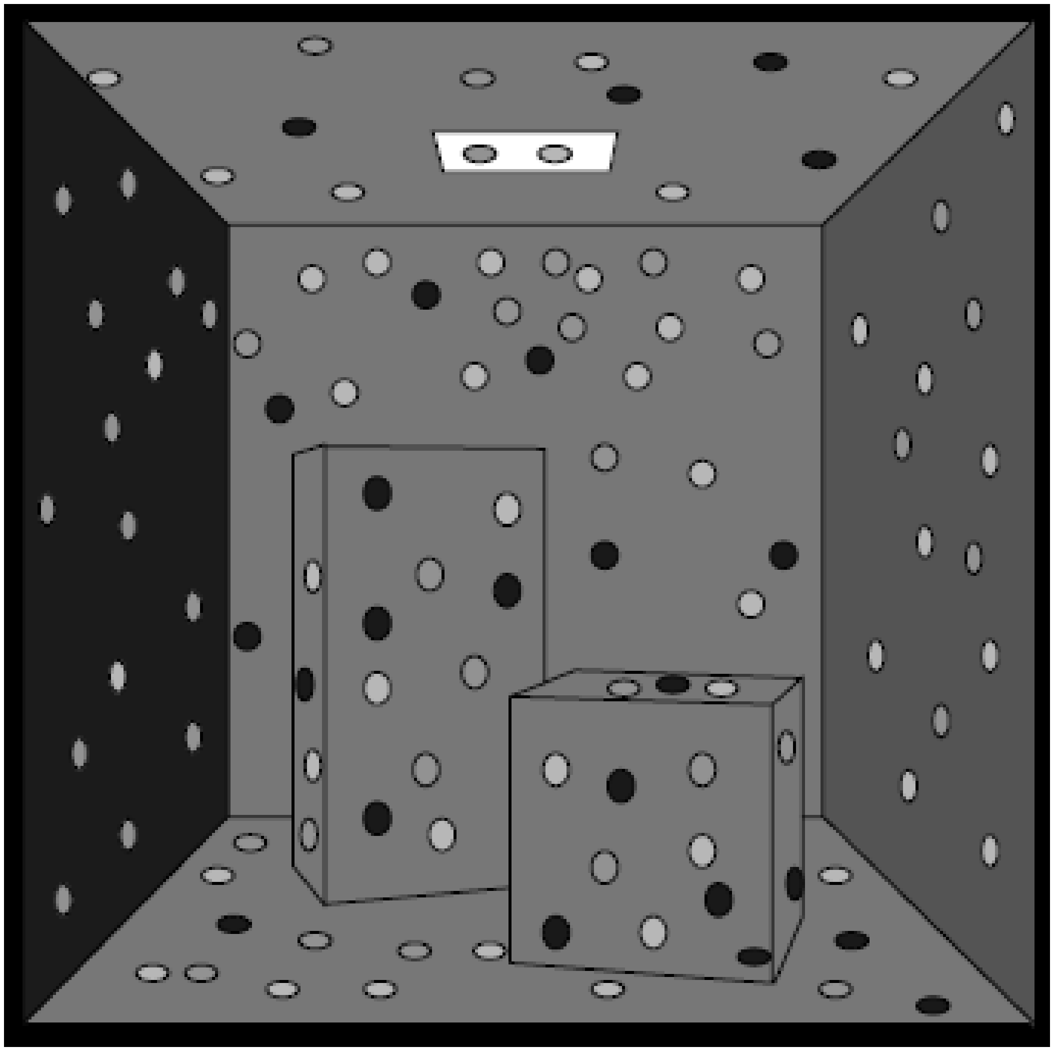
- Light path generated using MC techniques and Russian Roulette
- Store:
  - position
  - incoming direction
  - color
  - ...

Pass 1: shoot photons



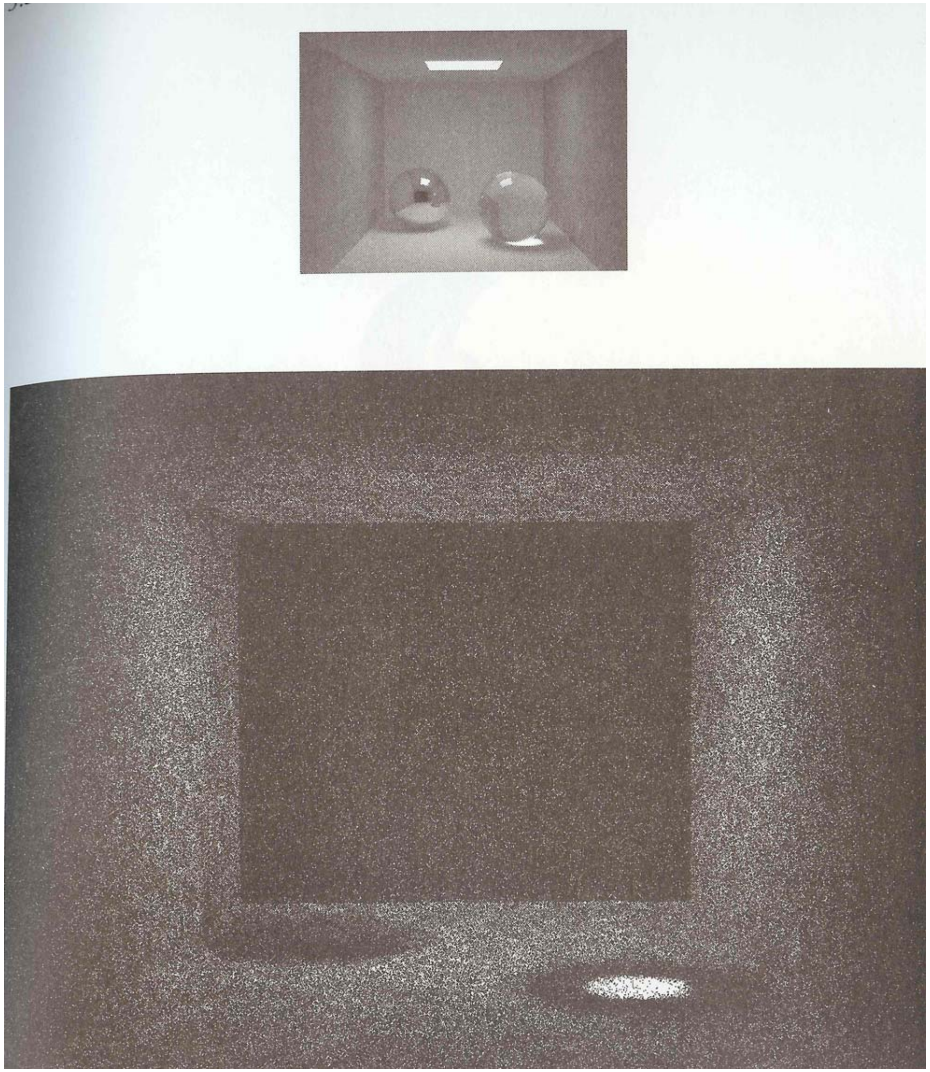
- Light path generated using MC techniques and Russian Roulette
- Store: **Flux for each photon**
  - position
  - incoming direction
  - color
  - ...

Pass 1: shoot photons

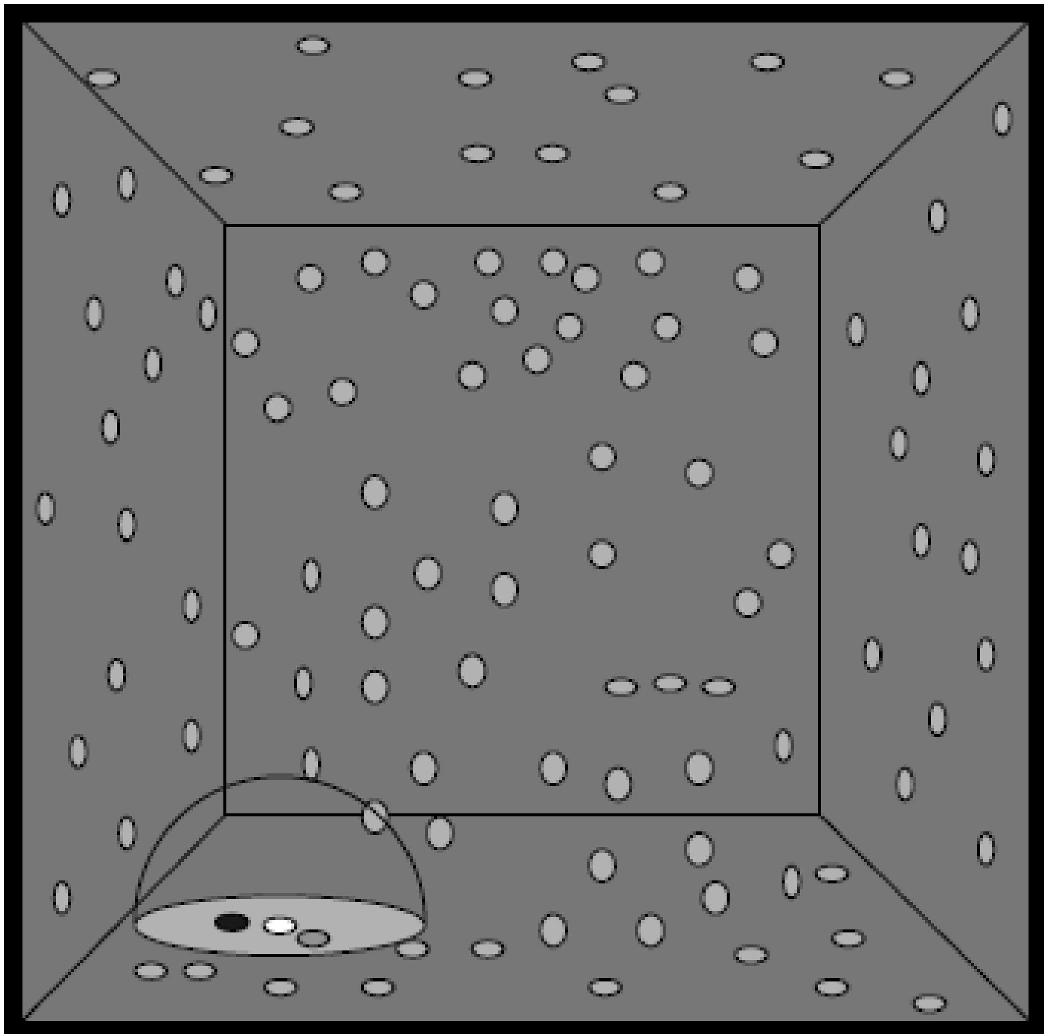


- Light path generated using MC techniques and Russian Roulette
- Store: **for diffuse materials**
  - position
  - incoming direction
  - color
  - ...

Stored Photons



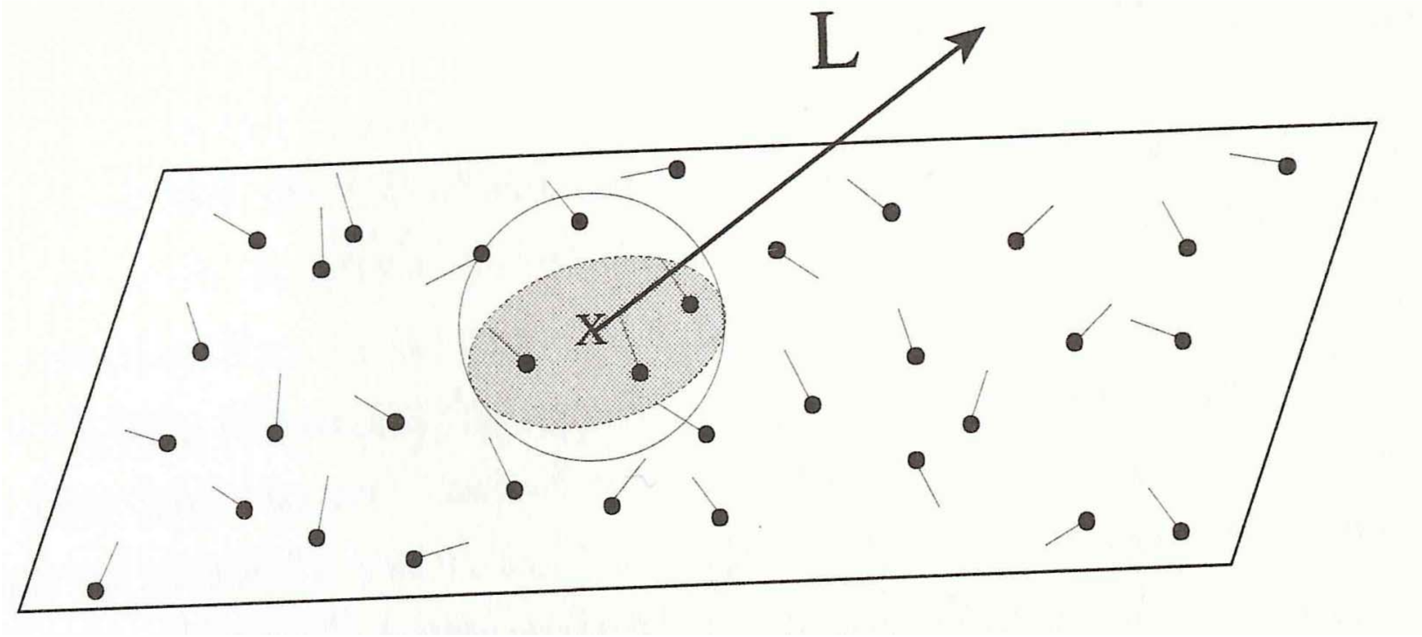
Pass 2: viewing ray



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

Radiance Estimation

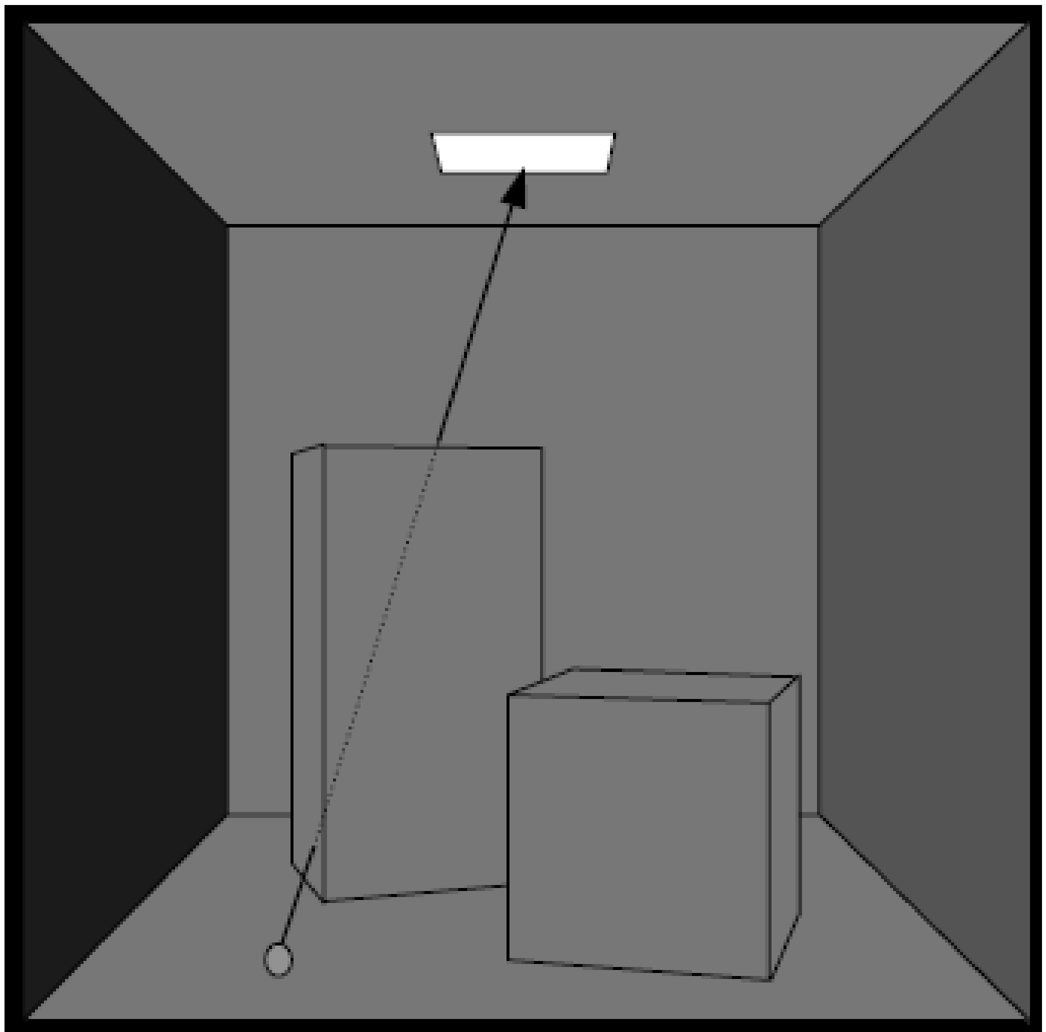
- Compute N nearest photons
  - Compute the radiance for each photon to outgoing direction
  - Consider BRDF
  - Divided by area



Efficiency

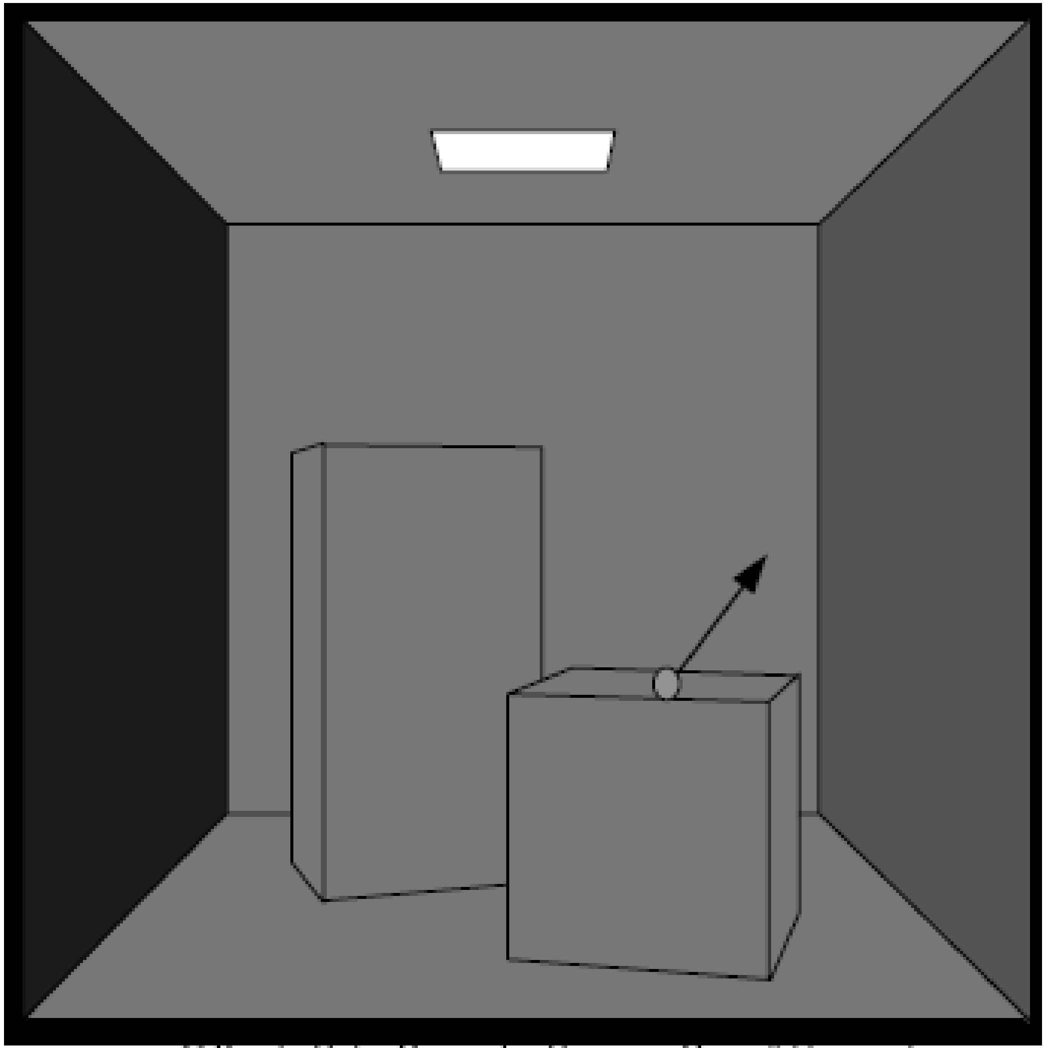
- **Want k nearest photons**
  - **Use kd-tree**
- **Using photon maps as it create noisy images**
  - **Need extremely large amount of photons**

Pass 2: Direct Illumination



Perform direct illumination for visible surface using regular MC sampling

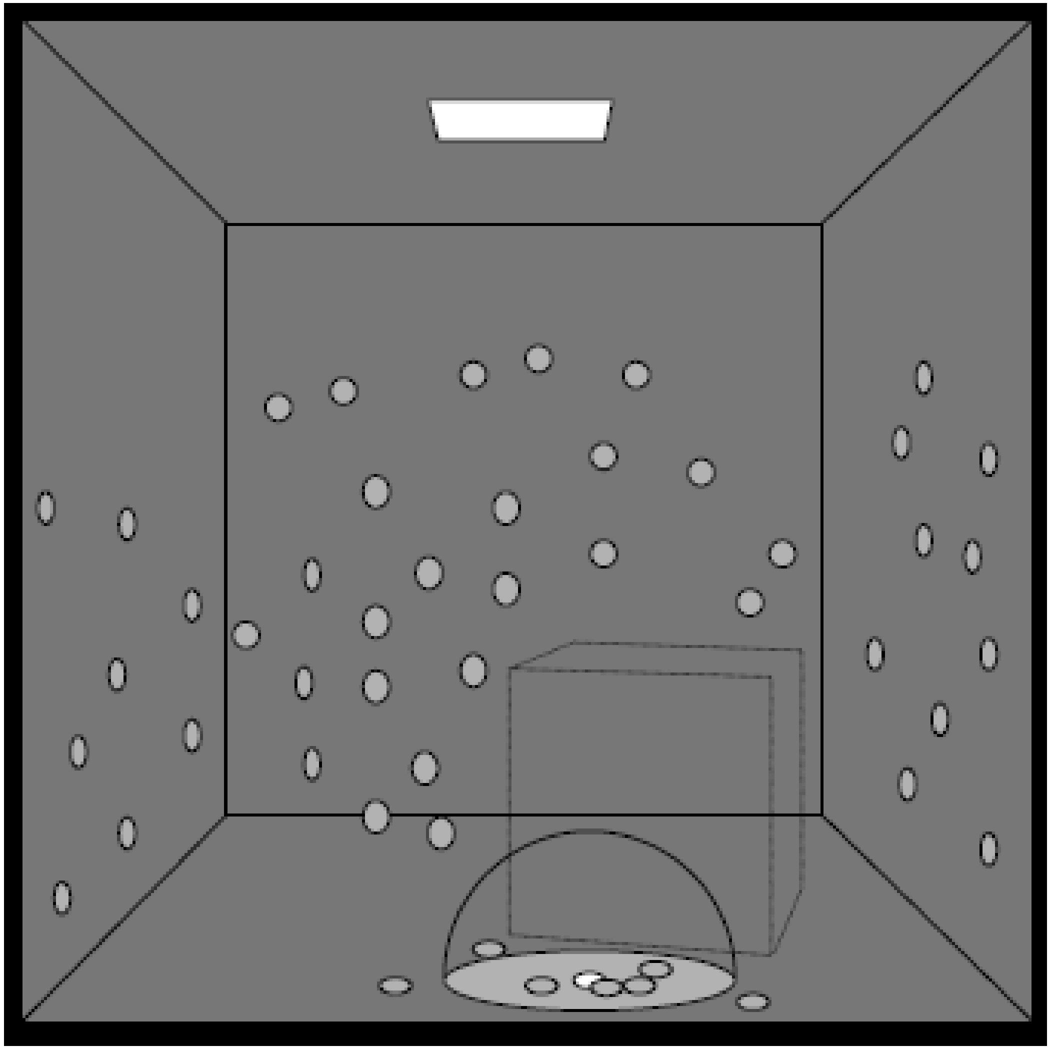
Pass 2: Specular reflections



© Kavita Bala, Computer Science, Cornell University

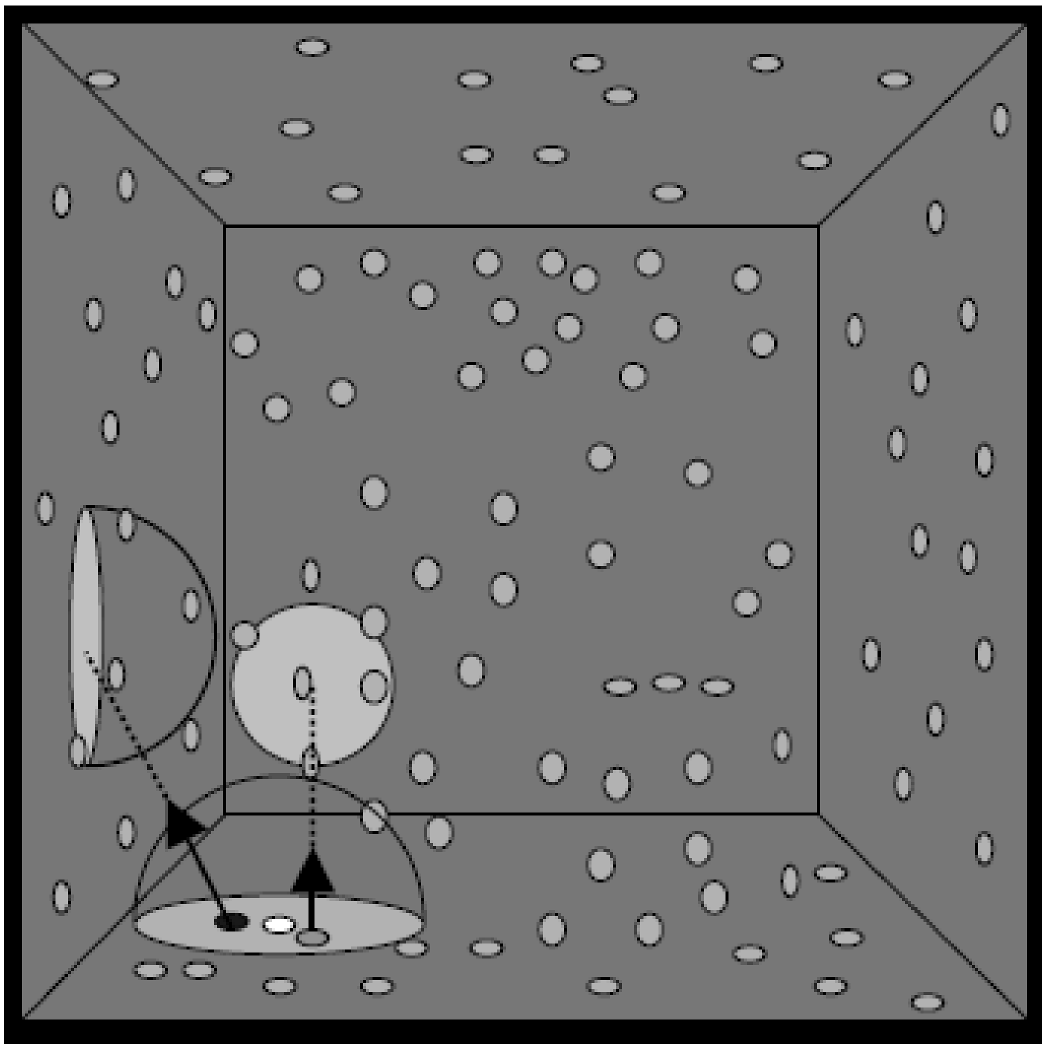
Specular reflection and transmission are ray traced

Pass 2: Caustics



- Direct use of "caustic" maps
- The "caustic" map is similar to a photon map but treats LS*D path
- Density of photons in caustic map usually high enough to use as is

Pass 2:Indirect Diffuse



- Search for N closest photons
- Assume these photons hit the point
- Compute average radiance by importance sampling of hemisphere

Result

Summary

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