Lecture 10: Monte Carlo Rendering

Chapters 4, 5 and 7 in Advanced GI

Fall 2004

Kavita Bala
Computer Science
Cornell University

Direct paths

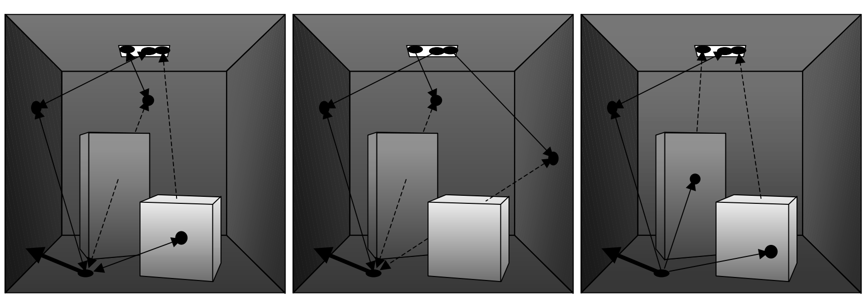
- Different path generators produce different estimators and different error characteristics
- Direct illumination general algorithm:

```
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);
```

Indirect Illumination

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

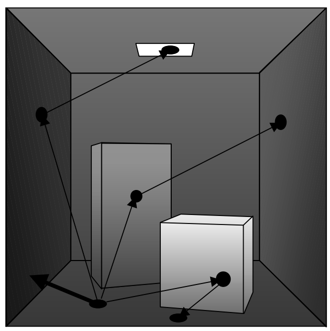
Indirect paths



Surface sampling	Source shooting	Receiver gathering
- 2 visibility terms; can be 0	- 1 visibility term - 1 ray intersection	- 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

Indirect paths

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

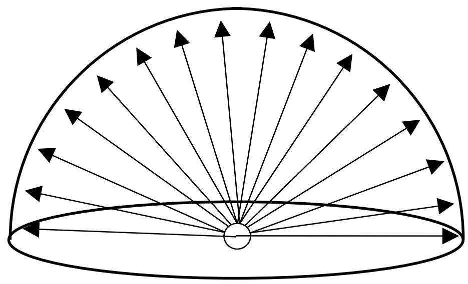

Stochastic Ray Tracing

- Sample 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

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

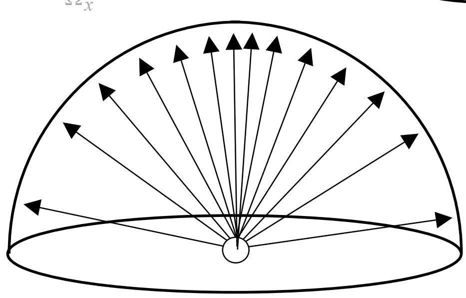


$$p(\Theta) = 1/(2\pi)$$

Sampling strategies

- Sampling according to the cosine factor

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

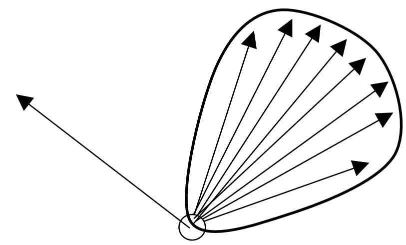


$$p(\Theta) = \cos\theta / \pi$$

Sampling strategies

- Sampling according to the BRDF

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



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

Example: sample according to BRDF

- Discrete pdf q_1, q_2, q_3 $\quad q_1 + q_2 + q_3 = 1$

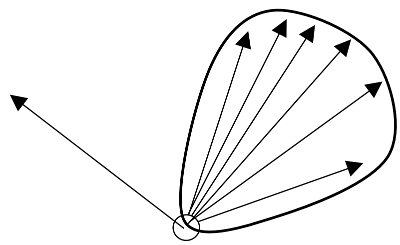
$$L_{indirect} = L_{diffuse} + L_{specular}$$

$$\langle L_{indirect} \rangle = \left\{ \begin{array}{c} \dfrac{L(x \leftarrow \Psi_i) k_d \cos(N, \Psi_i)}{q_1 p_1(\Psi_i)} \mid \xi < q_1 \\ \dfrac{L(x \leftarrow \Psi_i) k_s \cos^n(R, \Psi_i) \cos(N, \Psi_i)}{q_2 p_2(\Psi_i)} \mid q_1 \le \xi < q_1 + q_2 \\ 0 \mid otherwise \end{array} \right\}$$

Sampling strategies

- Sampling according to the BRDF times the cosine

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

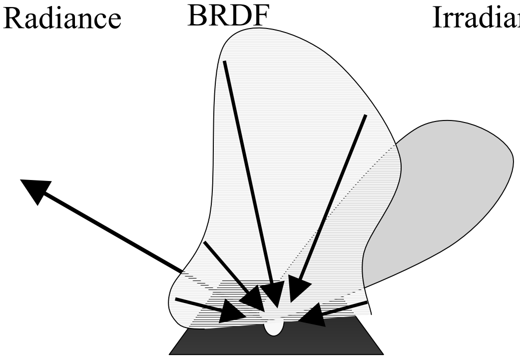


$$p(\Theta) \sim f_r(\Theta \leftrightarrow \Psi) \cos\theta$$

Multi-Importance-Sampling

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

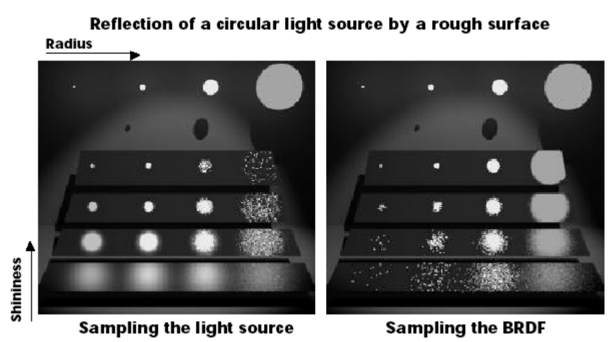
Radiance BRDF Irradiance



Importance Sampling

- Say we want to sample according to cosine term, BRDF, ….
- How do we blend the different sampling algorithms together?

Example



- Want to merge both techniques of sampling
  – How?

Balance Heuristic

- Two sampling techniques: j^{th} sample
  - $X_{1,j}$ with pdf $p_1(x)$, $X_{2,j}$ with pdf $p_2(x)$
  - Estimator Y_j for j^{th} sample

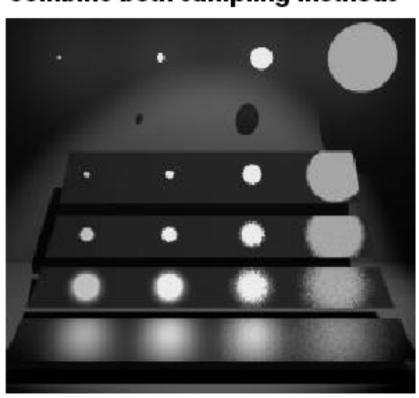
$$Y_{1,j} = \frac{f(X_{1,j})}{p_1(X_{1,j})} \qquad Y_{2,j} = \frac{f(X_{2,j})}{p_2(X_{2,j})}$$

$$Y_j = w_1 Y_{1,j} + w_2 Y_{2,j}$$

$$w_1(x) = \frac{p_1(x)}{p_1(x) + p_2(x)} \qquad w_2(x) = \frac{p_2(x)}{p_1(x) + p_2(x)}$$

Multiple Importance Sampling

Combine both sampling methods



From Veach and Guibas
Read Chapter 9, Veach

Efficiency

$$Efficiency \propto \frac{1}{Variance \bullet Cost}$$

- Some techniques:
  - Importance sampling
  - Sampling patterns
    - Stratified, Quasi-Monte Carlo
  - Many others

General GI algorithm

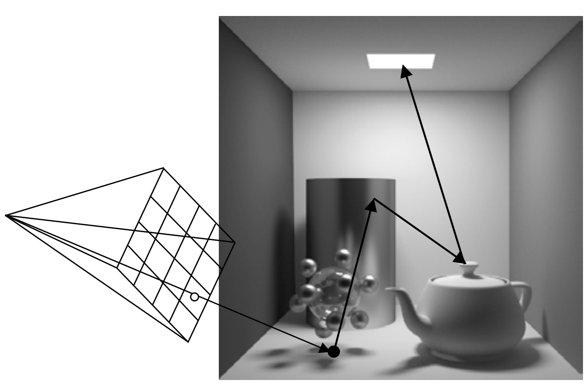
- Design path generators
- Path generators determine efficiency of global illumination algorithm
- Black boxes
  - evaluate brdf, L_e
  - ray intersection
  - visibility evaluation

Other Rendering Techniques

- Bidirectional Path Tracing
- Metropolis
- Biased Techniques
  - Irradiance caching
  - Photon Mapping

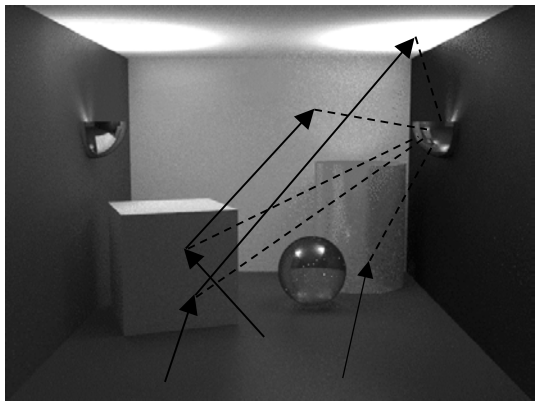
Stochastic ray tracing: limitations

- Generate a path from the eye to the light source



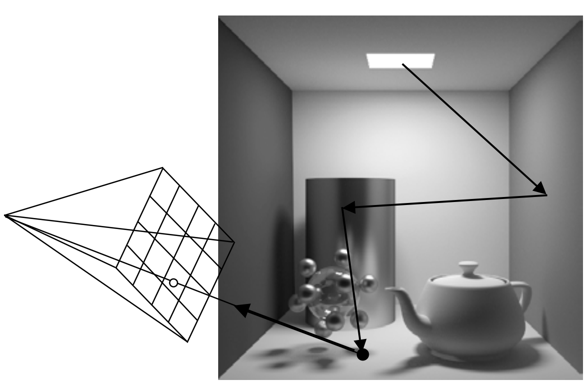
When does it not work?

- Scenes in which indirect lighting dominates



Bidirectional Path Tracing

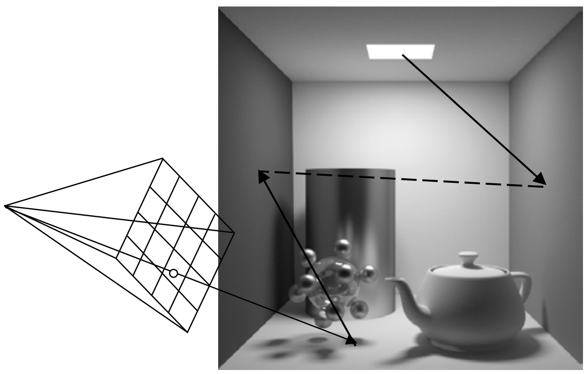
- So … we can generate paths starting from the light sources!



- Shoot ray to camera to see what pixels get contributions

Bidirectional Path Tracing

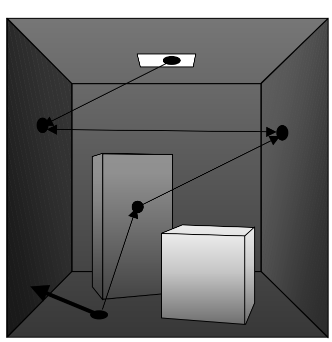
- Or paths generated from both camera and source at the same time ...!



- Connect endpoints to compute final contribution

Complex path generators

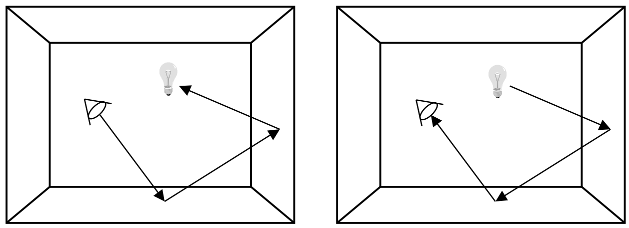
- Bidirectional ray tracing
  - shoot a path from light source
  - shoot a path from receiver
  - connect end points



Why? BRDF - Reciprocity

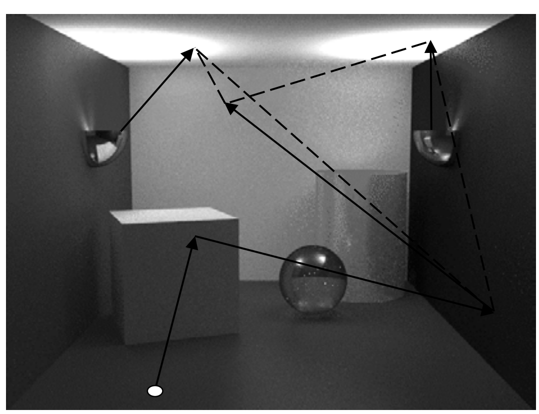
- Direction in which path is generated, is not important: Reciprocity

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



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

Bidirectional path tracing



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)

Classic ray tracing?

- Shoot shadow-rays (direct illumination)
- Shoot perfect specular rays only for indirect
- Ignores many paths
  - Does not solve the rendering equation

Other Rendering Techniques

- Metropolis
- Biased Techniques
  - Irradiance caching
  - Photon Mapping

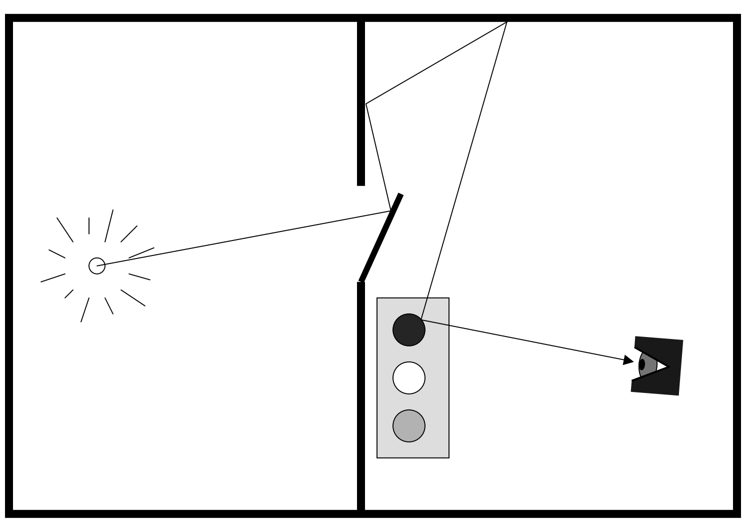
Metropolis

- Based on Metropolis Sampling (1950s)
- 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

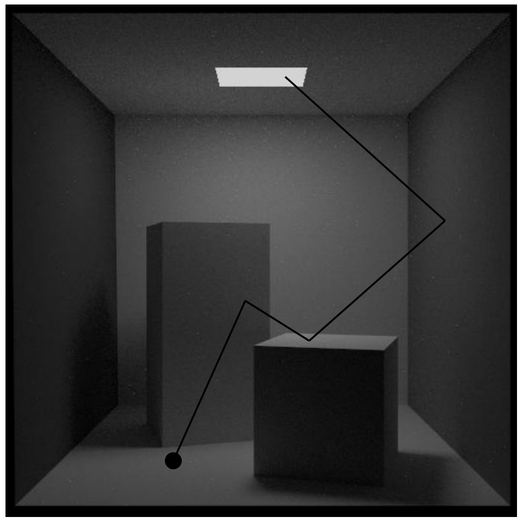
Metropolis

- Generate paths
- Once a valid path is found, mutate it to generate new valid paths
- Advantages:
  - Path re-use
  - Local exploration
    - Insight: found hard-to-find light distribution, mutate to find other such paths

Metropolis

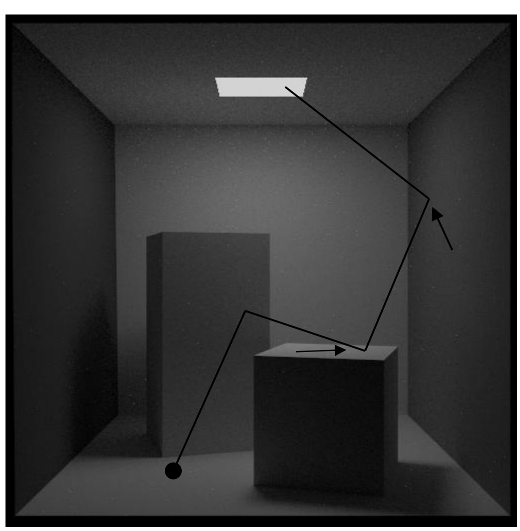


Metropolis



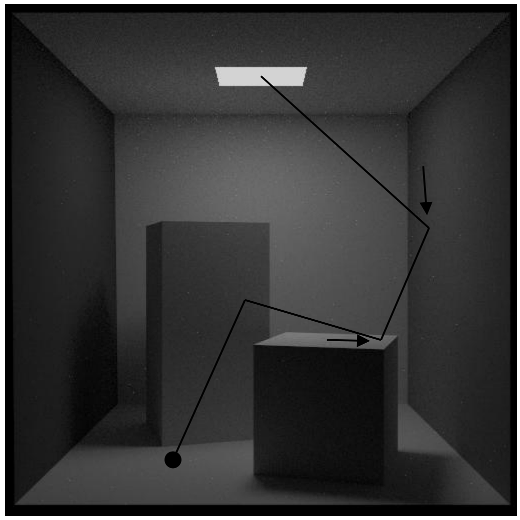
valid path

Metropolis



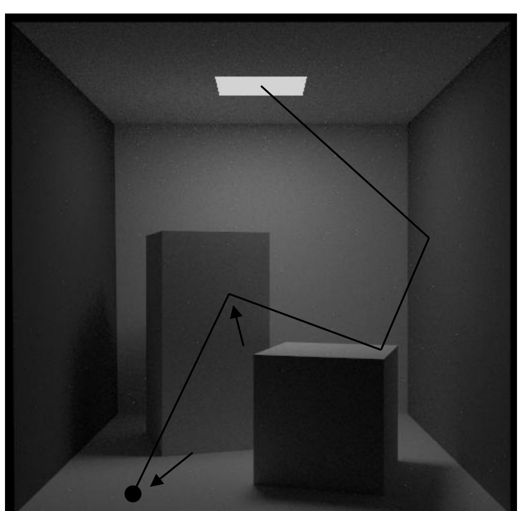
small
perturbations

Metropolis



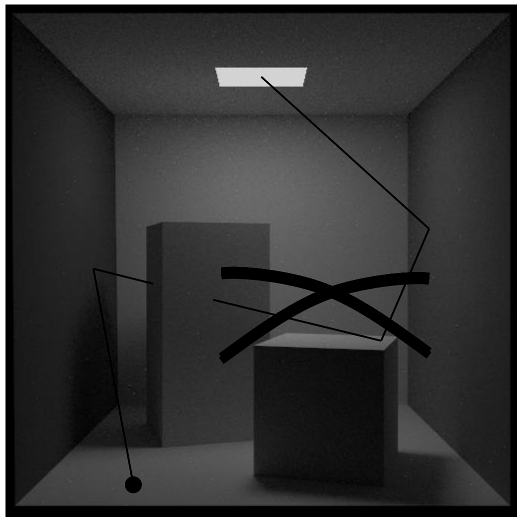
small
perturbations

Metropolis



mutations

Metropolis



Accept mutations based on energy transport

Metropolis

Metropolis

- Advantages
  - Robust
  - Good for hard to find light paths
- Disadvantages
  - Slow
  - 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 result with more samples
  - $E[I_{estimator}] = I + \varepsilon$ where $\lim_{N \to \infty} \varepsilon = 0$

© Kavita Bala, Computer Science, Cornell University

Biased Methods

- MC problems
  - Too noisy/slow
  - Noise is objectionable
- Biased methods: store information (caching)
  - Better type of noise: blurring
  - Greg Ward's Radiance
  - Photon Mapping
  - Density Estimation

Irradiance Caching

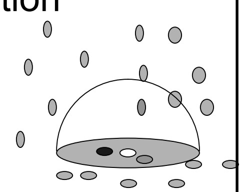
- Introduced by Greg Ward 1988
- Implemented in RADIANCE
  - Public-domain software
- Exploits smoothness of irradiance
  - Cache and interpolate irradiance estimates

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

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

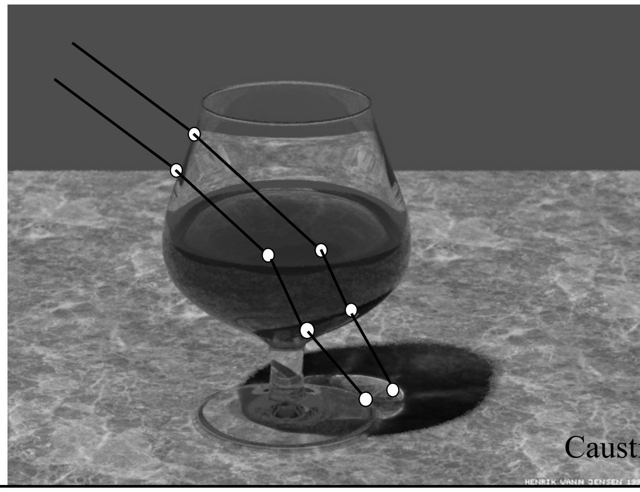
  - Otherwise, compute new sample

Radiance Examples

Radiance: Example

Photon Map

- Build on irradiance caching
- Use bidirectional ray tracing

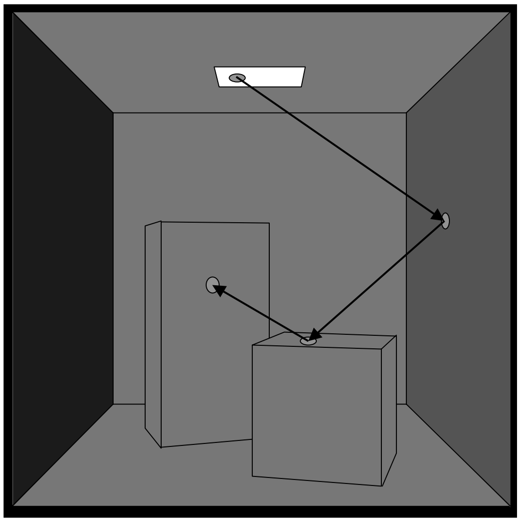


Caustic: LS+ D E

Photon Map

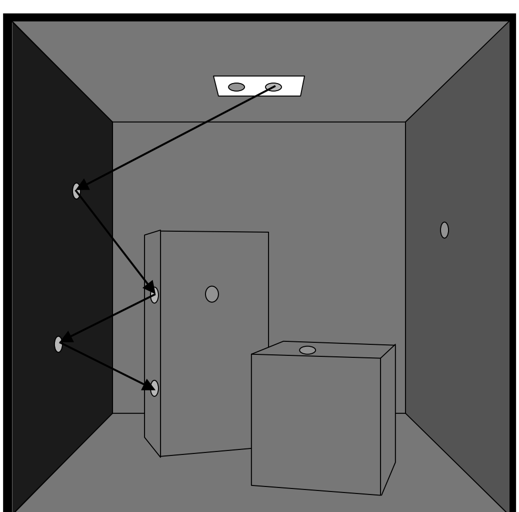
- 2 passes:
  - shoot "photons" (light-rays) and record any hit-points
  - shoot viewing rays, collect information from stored photons

Pass 1: shoot photons



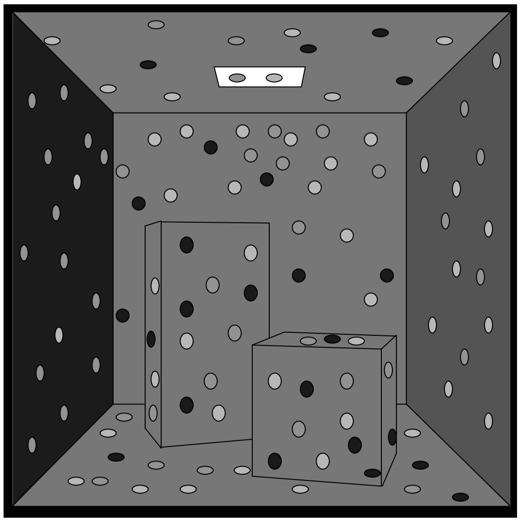
- Light path generated using MC techniques and Russian Roulette
- Store:
  - position
  - incoming direction
  - color
  - ...

Pass 1: shoot photons



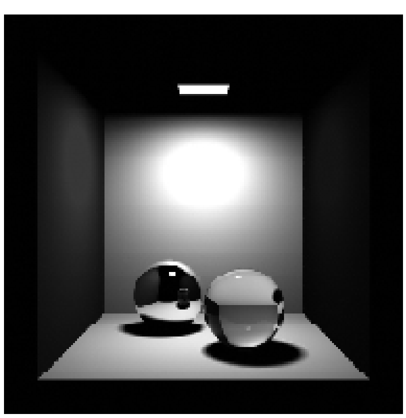
- Light path generated using MC techniques and Russian Roulette
- Store:
  - position
  - incoming direction
  - color
  - ...

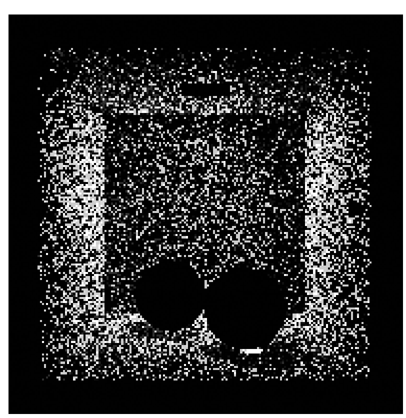
Pass 1: shoot photons



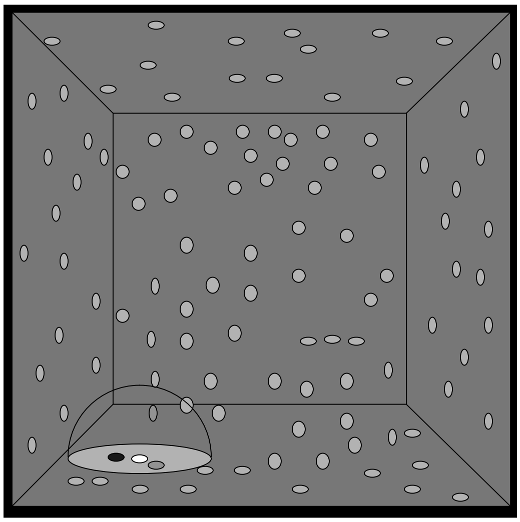
- Light path generated using MC techniques and Russian Roulette
- Store:
  - position
  - incoming direction
  - color
  - ...

Pass 1: shoot photons



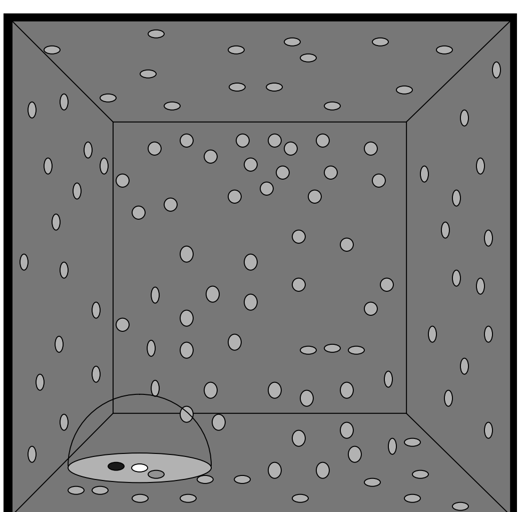


Pass 2: viewing ray (naive)



- Search for N closest photons
- Assume these photons hit the point we're interested in
- Compute average radiance

Pass 2: viewing ray (better)

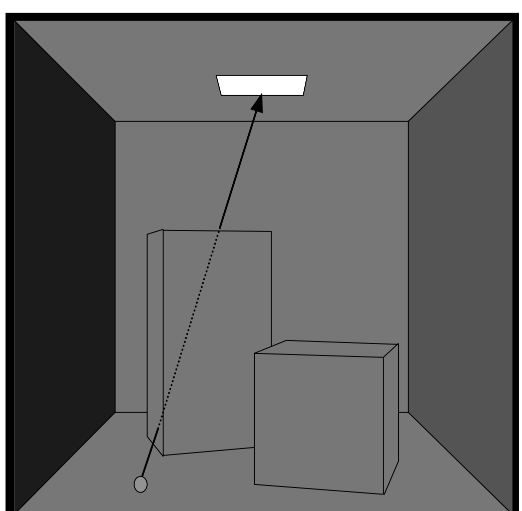


- Search for N closest photons (+check normal)
- Assume these photons hit the point we're interested in
- Compute average radiance

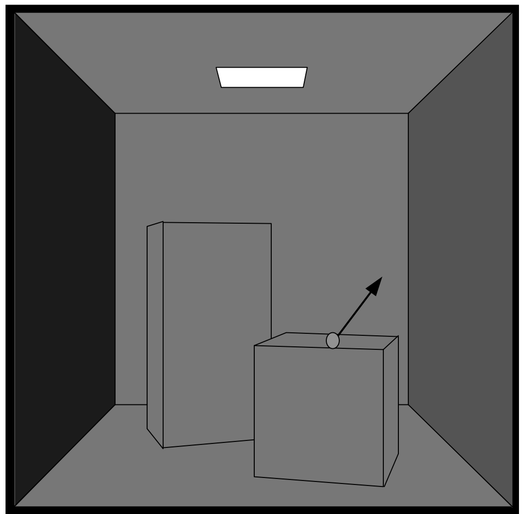
Efficiency

- Want k nearest photons
  – Use *Balanced kd-tree*
- Using photon maps as is create noisy images
  – Need EXTREMELY large amount of photons
- Filtering techniques can be used with different type of kernels
- The filtered results often look too blurry !!!

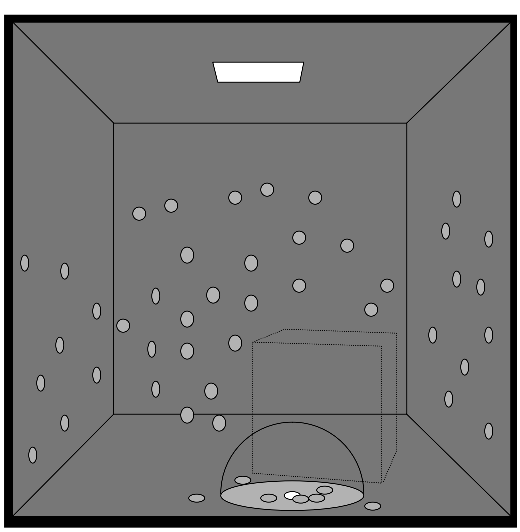
Pass 2: Direct Illumination



Pass 2: Specular reflections

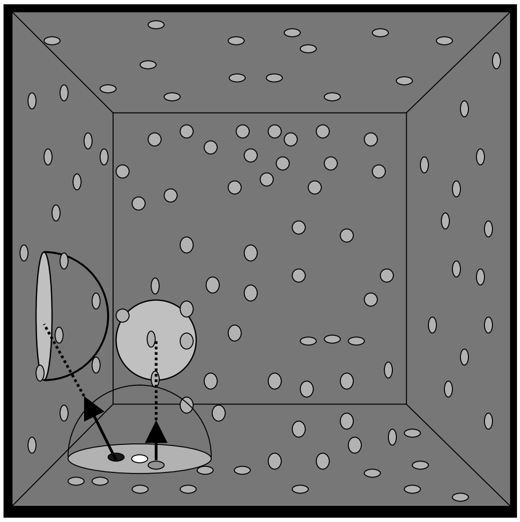


Pass 2: Caustics



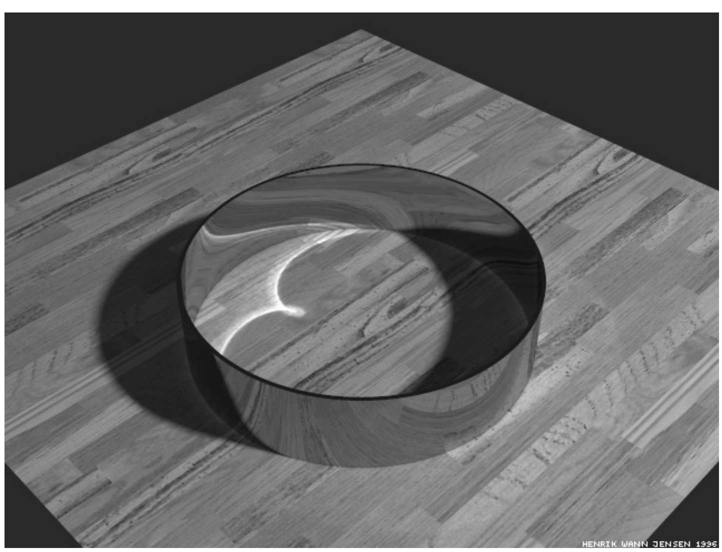
- Direct use of "caustic" maps
- The "caustic" map is similar to a photon map but treats LS*D path
- Density of photons in caustic map usually high enough to use as is

Pass 2:Indirect Diffuse

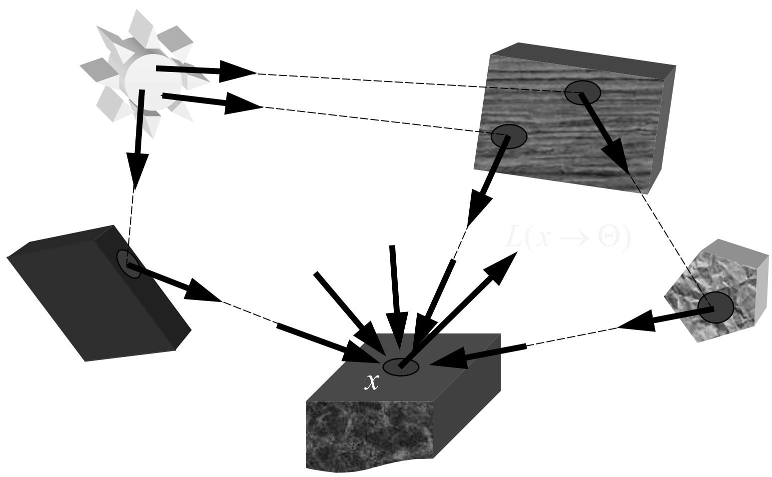


- Search for N closest photons
- Assume these photons hit the point
- Compute average radiance by importance sampling of hemisphere

Photon Map Results



Summary of MC



… find paths between sources and surfaces to be shaded

MC Advantages

- Convergence rate of O($\frac{1}{\sqrt{N}}$)
- Simple
  - Sampling
  - Point evaluation
  - Can use black boxes
- General
  - Works for high dimensions
  - Deals with discontinuities, crazy functions,…

MC integration - Non-Uniform

- Generate samples according to density function p(x)

$$\langle I \rangle = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

- Some parts of the integration domain have higher importance

- What is optimal p(x)?

$$p(x) \approx f(x) / \int f(x)dx$$

Non-Uniform Samples

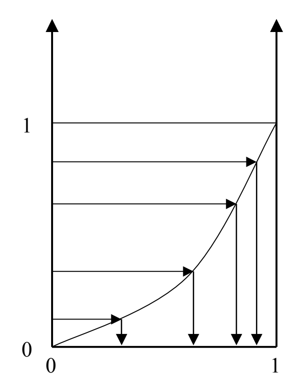
- 1) Choose a normalized probability density function *p(x)*

- 2) Integrate to get a probability distribution function *P(x)*:

$$P(x) = \int_0^x p(t)dt$$

- 3) Invert *P*:

$$x = P^{-1}(\xi)$$



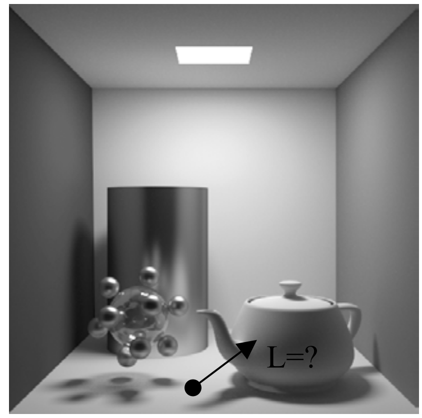
Note this is similar to going from y axis to x in discrete case!

How to compute?

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

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

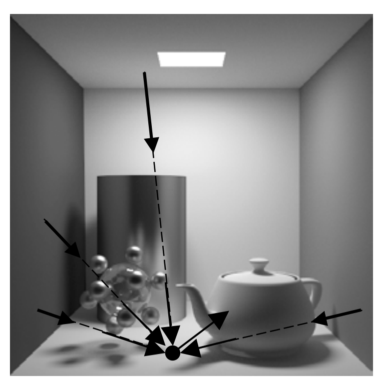
Now add $L_r(x \rightarrow \Theta) =$



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

How to compute? Recursion ...

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



Russian Roulette

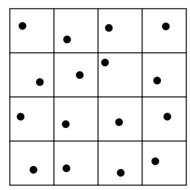
- Terminate recursion using Russian roulette
- Pick some 'absorption probability' α
  - probability $1-\alpha$ that ray will bounce
  - estimated radiance becomes $L/(1-\alpha)$
- E.g. $\alpha = 0.9$
  - only 1 chance in 10 that ray is reflected
  - estimated radiance of that ray is multiplied by 10
- Intuition
  - instead of shooting 10 rays, we shoot only 1, but count the contribution of this one 10 times

Stochastic Ray Tracing

- Parameters?
  - # starting rays per pixel
  - # random rays for each surface point (branching factor)
- Branching factor = 1: path tracing

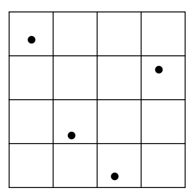
Higher Dimensions

- Stratified grid sampling:



$\rightarrow N^d$ samples

- N-rooks sampling:



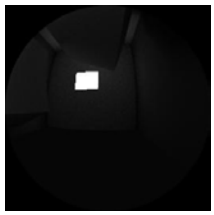
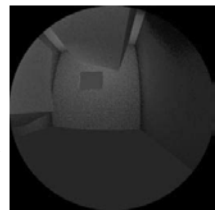
$\rightarrow N$ samples

Quasi Monte Carlo

- Converges as fast as stratified sampling
  - Does not require knowledge about how many samples will be used
- Using QMC directions evenly spaced no matter how many samples are used
- Samples properly stratified-> better than pure MC

Next Event Estimation

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

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

- So … sample direct and indirect with separate MC integration

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

Balance Heuristic

- Two sampling techniques: j^{th} sample
  - $X_{1,j}$ with pdf $p_1(x)$, $X_{2,j}$ with pdf $p_2(x)$
  - Estimator Y_j for j^{th} sample

$$Y_{1,j} = \frac{f(X_{1,j})}{p_1(X_{1,j})} \qquad Y_{2,j} = \frac{f(X_{2,j})}{p_2(X_{2,j})}$$

$$Y_j = w_1 Y_{1,j} + w_2 Y_{2,j}$$

$$w_i(x) = \frac{p_i(x)}{p_1(x) + p_2(x)}$$

Other Rendering Techniques

- Bidirectional Path Tracing
- Metropolis
- Biased Techniques
  - Irradiance caching
  - Photon Mapping