

Lecture 10: Monte Carlo Rendering

CS 6620, Spring 2009

Kavita Bala

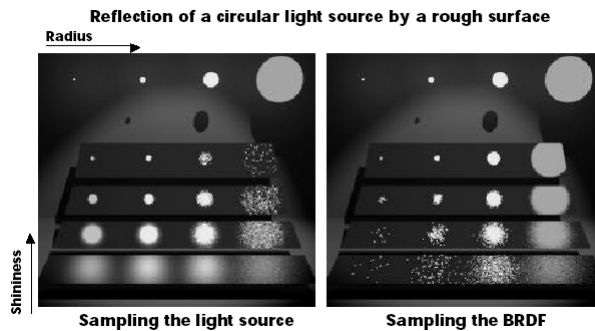
Computer Science

Cornell University

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

Example



- Want to merge both techniques of sampling
 - How?

© Kavita Bala, Computer Science, Cornell University

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

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

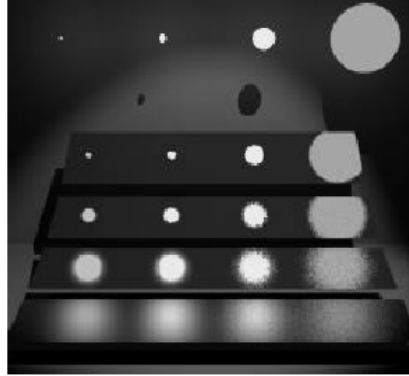
$$Y_j = w_1 Y_{1,j} + w_2 Y_{2,j}$$

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

© Kavita Bala, Computer Science, Cornell University

Multiple Importance Sampling

Combine both sampling methods



From Veach and Guibas
Read Chapter 9, Veach

© Kavita Bala, Computer Science, Cornell University

Indirect paths

```
indirectIllum (x, theta)
    est_rad = 0;
    if (no absorption) {
        for (i=0; i<n; i++)
            sample direction psi on hemisphere;
            y = trace (x, psi);
            est_rad +=(radiance(y,-psi)*BRDF*cos())/p(psi);
    }
    est_rad = est_rad / n;
    return(est_rad/(1-absorption));
```

```
Compute radiance (x, dir){
    estRadiance = Le (x, dir);
    estRadiance += directIllum (x, dir);
    estRadiance += indirectIllum (x, dir);
    return estRadiance;
}
```

© Kavita Bala, Computer Science, Cornell University

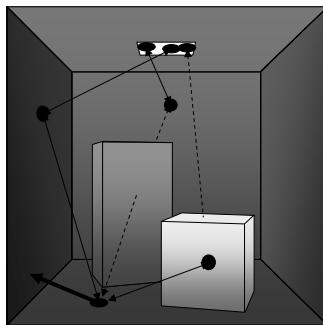
Indirect Illumination

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

© Kavita Bala, Computer Science, Cornell University

Indirect paths - surface sampling

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



- per path:
 - 2 visibility checks

© Kavita Bala, Computer Science, Cornell University

Indirect paths - surface sampling

- Indirect illumination (path length 2):

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

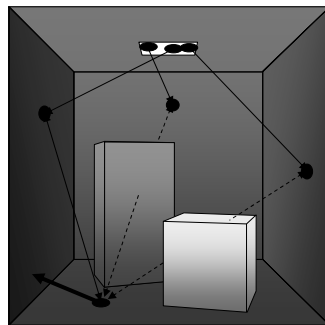
$$\langle L(x \rightarrow \Theta) \rangle = \frac{1}{N} \sum_{i=1}^N \frac{L(y_i \rightarrow \Psi_{1i}) f_r(z_i, -\Psi_{1i} \leftrightarrow \Psi_{2i}) G(z_i, y_i) f_r(z_i, -\Psi_{2i} \leftrightarrow \Theta) G(z_i, x)}{p_y(y_i) p_z(z_i)}$$

- 2 visibility values cause noise
 - which might be 0

© Kavita Bala, Computer Science, Cornell University

Indirect paths - source shooting

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



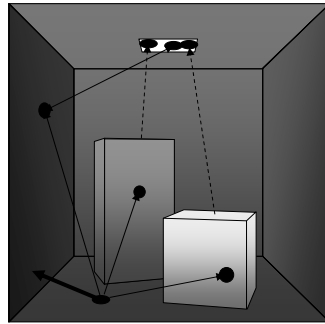
– per path:

- 1 ray intersection
- 1 visibility check

© Kavita Bala, Computer Science, Cornell University

Indirect paths - receiver gathering

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

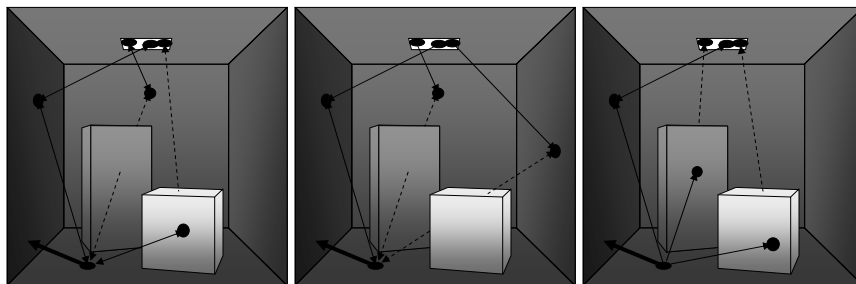


– per path:

- 1 ray intersection
- 1 visibility check

© Kavita Bala, Computer Science, Cornell University

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

© Kavita Bala, Computer Science, Cornell University

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

© Kavita Bala, Computer Science, Cornell University

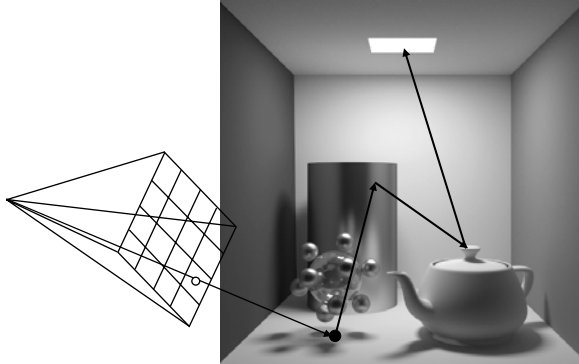
Other Rendering Techniques

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

© Kavita Bala, Computer Science, Cornell University

Stochastic ray tracing: limitations

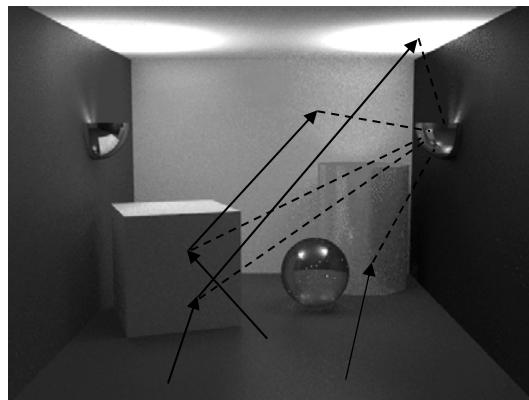
- Generate a path from the eye to the light source



© Kavita Bala, Computer Science, Cornell University

When does it not work?

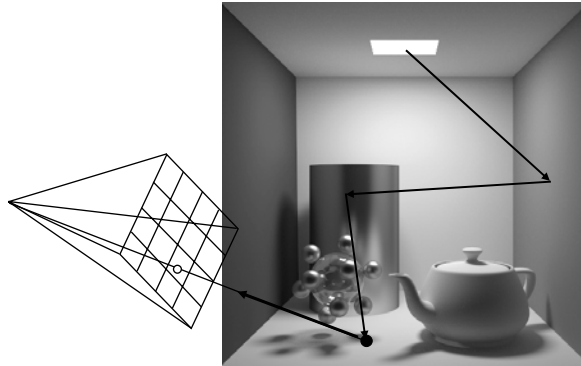
- Scenes in which indirect lighting dominates



© Kavita Bala, Computer Science, Cornell University

Bidirectional Path Tracing

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

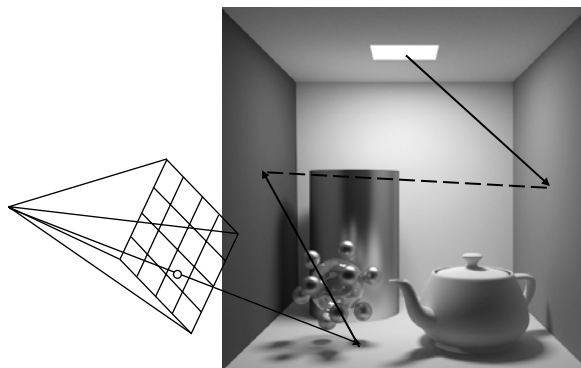


- Shoot ray to camera to see what pixels get contributions

© Kavita Bala, Computer Science, Cornell University

Bidirectional Path Tracing

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



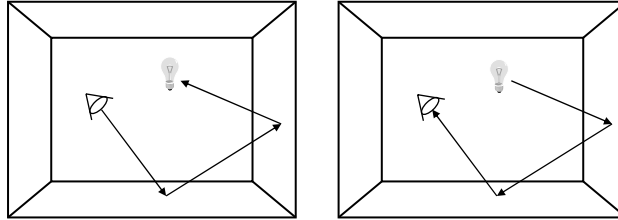
- Connect endpoints to compute final contribution

© Kavita Bala, Computer Science, Cornell University

Why? BRDF - Reciprocity

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

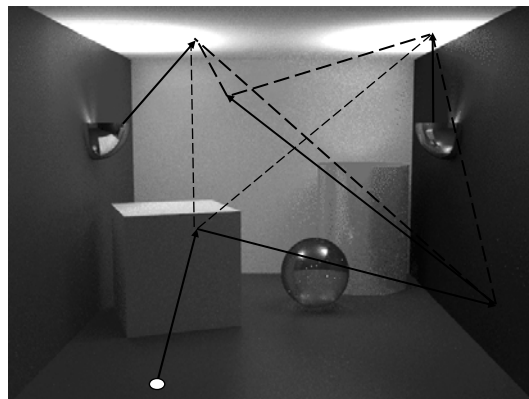
$$f(\Psi \rightarrow \Theta) = f(\Theta \rightarrow \Psi) = f(\Psi \leftrightarrow \Theta)$$



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

© Kavita Bala, Computer Science, Cornell University

Bidirectional path tracing



© Kavita Bala, Computer Science, Cornell University

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)

© Kavita Bala, Computer Science, Cornell University

Classic ray tracing?

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

© Kavita Bala, Computer Science, Cornell University

Other Rendering Techniques

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

© Kavita Bala, Computer Science, Cornell University

Metropolis

- Based on Metropolis Sampling (1950s)
 - Builds on bidirectional path tracing
- 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

© Kavita Bala, Computer Science, Cornell University

Metropolis

- Generate paths (eg, w/ bidirectional pt)
- 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

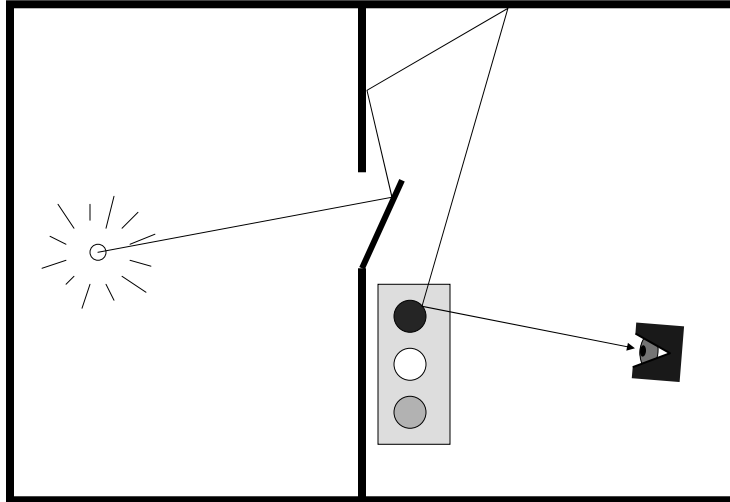
© Kavita Bala, Computer Science, Cornell University

Metropolis



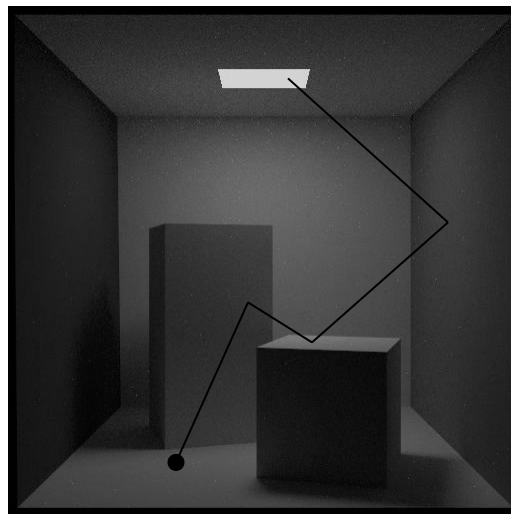
© Kavita Bala, Computer Science, Cornell University

Metropolis



© Kavita Bala, Computer Science, Cornell University

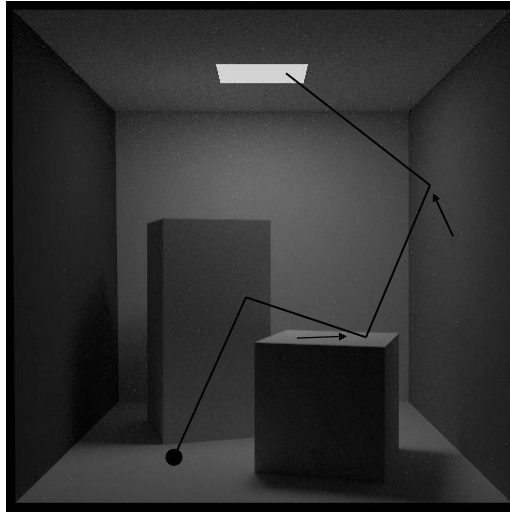
Metropolis



valid path

© Kavita Bala, Computer Science, Cornell University

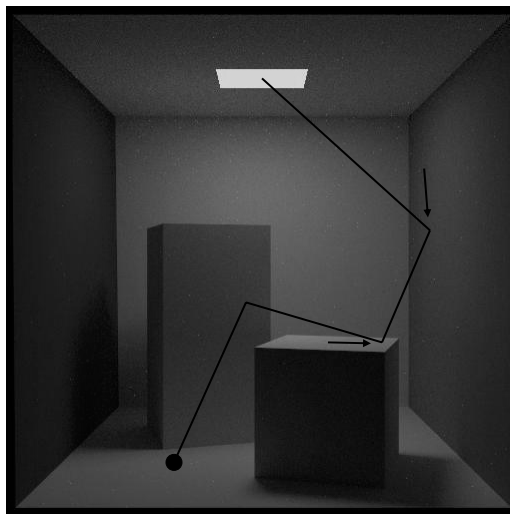
Metropolis



small
perturbations

© Kavita Bala, Computer Science, Cornell University

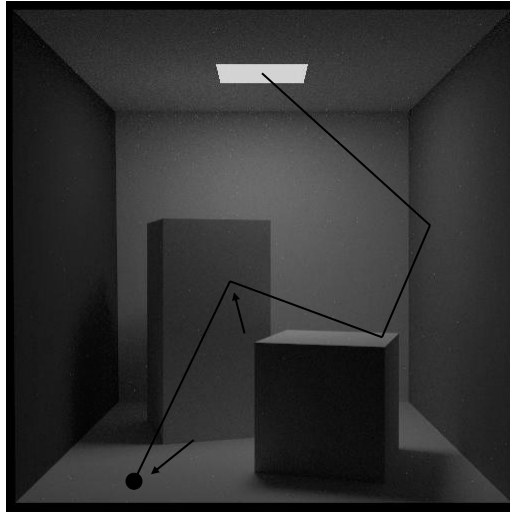
Metropolis



small
perturbations

© Kavita Bala, Computer Science, Cornell University

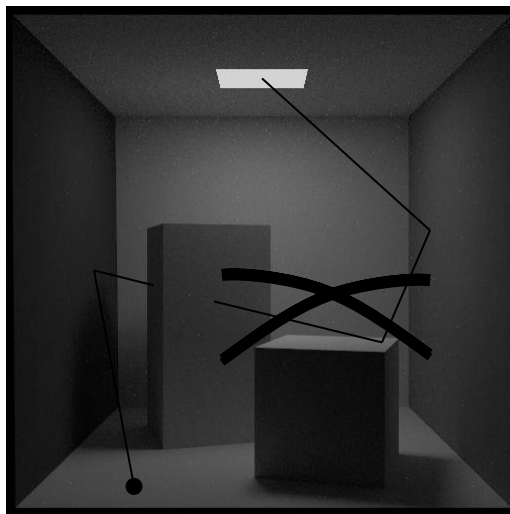
Metropolis



mutations

© Kavita Bala, Computer Science, Cornell University

Metropolis

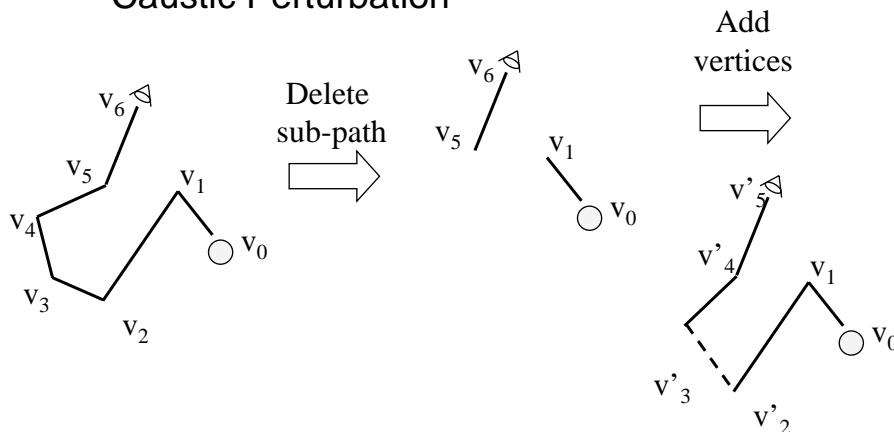


Accept
mutations
based on
energy
transport

© Kavita Bala, Computer Science, Cornell University

Mutations

- Types of mutations:
 - Bidirectional mutation
 - Caustic Perturbation



Theoretical Framework

- Generates a set of samples from f
 - f defined over a state space: set of all paths
- After first sample, each sample is random mutation of previous sample
 - Design mutator
 - Mutation can be accepted (prob. a) or rejected ($1-a$)
- Steady-state pdf is achieved
 - Distribution of sampled paths proportional to contribution to image
 - No explicit use of f (integrate f , normalize f , inverse f etc. not required)

© Kavita Bala, Computer Science, Cornell University

Approach

- Transition probabilities: $T(x \rightarrow x')$
 - Prob. mutator creates mutation from x to x'
- Acceptance probabilities: $a(x \rightarrow x')$
 - Prob. mutation from x to x' accepted
- Detailed Balance:

$$f(x)T(x \rightarrow x')a(x \rightarrow x') = f(x')T(x' \rightarrow x)a(x' \rightarrow x)$$

© Kavita Bala, Computer Science, Cornell University

Metropolis



© Kavita Bala, Computer Science, Cornell University

Metropolis

- Advantages
 - Robust
 - Good for hard to find light paths
- Disadvantages
 - Slow
 - Tricky to implement and get right

© Kavita Bala, Computer Science, Cornell University

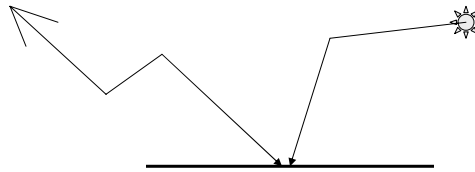
Pure Path Tracing

- Advantages
 - No need for meshing
 - General surfaces – requires ray intersections
 - *Unbiased* estimates
- Disadvantages
 - Noisy! Every point is independent
 - Starts from scratch – does not exploit coherence

© Kavita Bala, Computer Science, Cornell University

Path Re-Use

- What is coherence?
 - Nearby values are similar to what we want



© Kavita Bala, Computer Science, Cornell University

Unbiased vs. Consistent

- Unbiased
 - No systematic error
 - $E[l_{\text{estimator}}] = I$
 - Better results with larger N
- Consistent
 - Converges to correct result with more samples
 - $E[l_{\text{estimator}}] = I + \varepsilon$ where $\lim_{N \rightarrow \infty} \varepsilon = 0$

© Kavita Bala, Computer Science, Cornell University

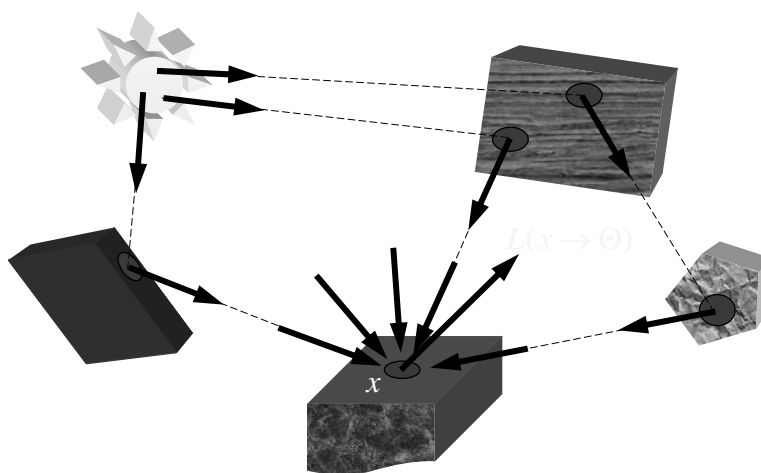
Biased Methods

- Store information (caching)
 - Better type of noise: blurring
- Greg Ward's Radiance
- Photon Mapping

- Instant Radiosity
- Lightcuts/Multidimensional lightcuts

© Kavita Bala, Computer Science, Cornell University

Summary of MC



... find paths between sources and surfaces to be shaded

© Kavita Bala, Computer Science, Cornell University

MC Advantages

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

© Kavita Bala, Computer Science, Cornell University

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 $p(x)$?

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

© Kavita Bala, Computer Science, Cornell University

Non-Uniform Samples

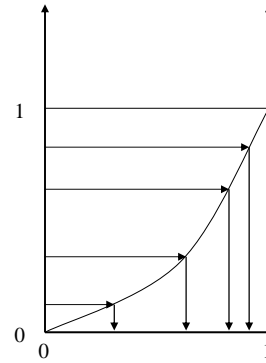
- 1) Choose a normalized probability density function $p(x)$
- 2) Integrate to get a probability distribution function $P(x)$:

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

- 3) Invert P :

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

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



© Kavita Bala, Computer Science, Cornell University

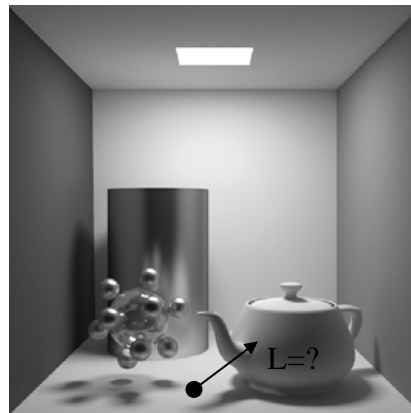
How to compute?

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

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

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

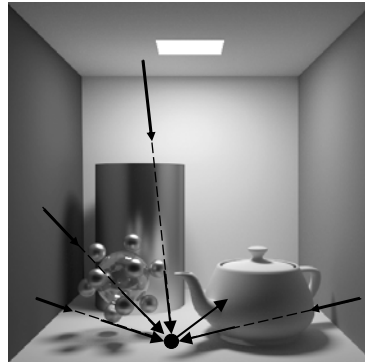
$$\int_{\Omega_x} f_r(\Psi \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$



© Kavita Bala, Computer Science, Cornell University

How to compute? Recursion ...

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



© Kavita Bala, Computer Science, Cornell University

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

© Kavita Bala, Computer Science, Cornell University

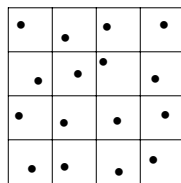
Stochastic Ray Tracing

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

© Kavita Bala, Computer Science, Cornell University

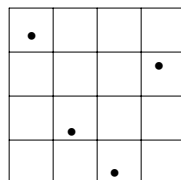
Higher Dimensions

- Stratified grid sampling:



→ N^d samples

- N-rooks sampling:



→ N samples

© Kavita Bala, Computer Science, Cornell University

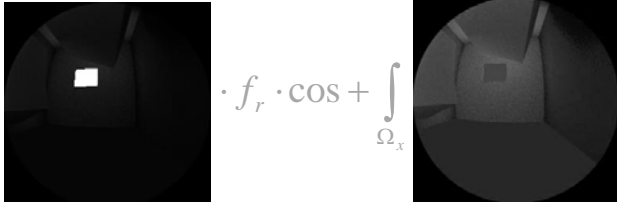
Quasi Monte Carlo

- 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

© Kavita Bala, Computer Science, Cornell University

Next Event Estimation

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

$$= L_e + \int_{\Omega_x} \text{img}_1 \cdot f_r \cdot \cos + \int_{\Omega_x} \text{img}_2 \cdot f_r \cdot \cos$$


- So ... sample direct and indirect with separate MC integration

© 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);
```

© Kavita Bala, Computer Science, Cornell University

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

© Kavita Bala, Computer Science, Cornell University

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

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

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

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

© Kavita Bala, Computer Science, Cornell University

Other Rendering Techniques

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

© Kavita Bala, Computer Science, Cornell University