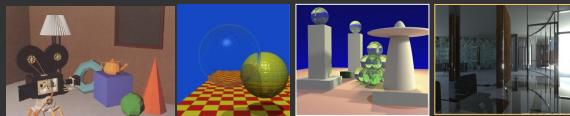


Computer Graphics II: Rendering

CSE 168 [Spr 25], Lecture 7: Monte Carlo Path Tracing
Ravi Ramamoorthi

<http://viscomp.ucsd.edu/classes/cse168/sp25>



1

To Do

- Homework 2 (direct lighting) due in two days
- Next assignment path tracing (on edX edge). This lecture covers much of that material

2

Motivation

- General solution to rendering and global illumination
- Suitable for a variety of general scenes
- Based on Monte Carlo methods
- Enumerate all paths of light transport

3

Monte Carlo Path Tracing



Big diffuse light source, 20 minutes

Jensen

4

Monte Carlo Path Tracing



Jensen

5

Monte Carlo Path Tracing

Advantages

- Any type of geometry (procedural, curved, ...)
- Any type of BRDF (specular, glossy, diffuse, ...)
- Samples all types of paths $(L(SD)^*E)$
- Accuracy controlled at pixel level
- Low memory consumption
- Unbiased - error appears as noise in final image

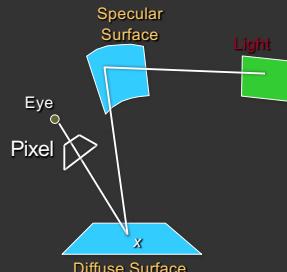
Disadvantages (standard Monte Carlo problems)

- Slow convergence (square root of number of samples)
- Noise in final image

6

Monte Carlo Path Tracing

Integrate radiance for each pixel by sampling paths randomly



$$L_o(x, \bar{w}) = L_e(x, \bar{w}) + \int_{\Omega} f_r(x, \bar{w}', \bar{w}) L_i(x, \bar{w}') (\bar{w}' \cdot \bar{n}) d\bar{w}$$

7

Simple Monte Carlo Path Tracer

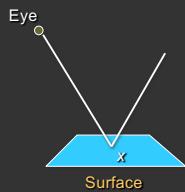
- Step 1: Choose a ray (u, v, θ, ϕ) [per pixel]; assign weight = 1
- Step 2: Trace ray to find intersection with nearest surface
- Step 3: Randomly choose between emitted and reflected light
 - Step 3a: If emitted, return weight' * L_e
 - Step 3b: If reflected, weight' *= reflectance
Generate ray in random direction
Go to step 2

8

Sampling Techniques

Problem: how do we generate random points/directions during path tracing and reduce variance?

- Importance sampling (e.g. by BRDF)
- Stratified sampling



9

Outline

- Motivation and Basic Idea
- *Implementation of simple path tracer*
- Variance Reduction: Importance sampling
- Other variance reduction methods
- Specific 2D sampling techniques

10

Simplest Monte Carlo Path Tracer

For each pixel, cast n samples and average

- Choose a ray with $p=\text{camera}$, $d=(\theta, \phi)$ within pixel
- Pixel color += $(1/n) * \text{TracePath}(p, d)$

TracePath(p, d) returns (r,g,b) [and calls itself recursively]:

- Trace ray (p, d) to find nearest intersection p'
- Select with probability (say) 50%:
 - Emitted:
return $2 * (L_{\text{red}}, L_{\text{green}}, L_{\text{blue}}) // 2 = 1/(50\%)$
 - Reflected:
generate ray in random direction d'
return $2 * f_r(d \rightarrow d') * (n * d') * \text{TracePath}(p', d')$

11

Simplest Monte Carlo Path Tracer

For each pixel, **cast n samples and average over paths**

- Choose a ray with $p=\text{camera}$, $d=(\theta, \phi)$ within pixel
- Pixel color += $(1/n) * \text{TracePath}(p, d)$

TracePath(p, d) returns (r,g,b) [and calls itself recursively]:

- Trace ray (p, d) to find nearest intersection p'
- Select with probability (say) 50%:
 - Emitted:
return $2 * (L_{\text{red}}, L_{\text{green}}, L_{\text{blue}}) // 2 = 1/(50\%)$
 - Reflected:
generate ray in random direction d'
return $2 * f_r(d \rightarrow d') * (n * d') * \text{TracePath}(p', d')$

12

Simplest Monte Carlo Path Tracer

For each pixel, cast n samples and average

- Choose a ray with $p=\text{camera}$, $d=(\theta, \phi)$ within pixel
- Pixel color $+= (1/n) * \text{TracePath}(p, d)$

$\text{TracePath}(p, d)$ returns (r, g, b) [and calls itself recursively]:

- Trace ray (p, d) to find nearest intersection p'
- Select with probability (say) 50%:
 - Emitted: $\text{return } 2 * (\text{Le}_{\text{red}}, \text{Le}_{\text{green}}, \text{Le}_{\text{blue}}) // 2 = 1/(50\%)$
 - Reflected: $\text{generate ray in random direction } d'$
 $\text{return } 2 * f_r(d \rightarrow d') * (n * d') * \text{TracePath}(p', d')$

Weight = 1/probability
Remember: unbiased
requires having $f(x) / p(x)$

13

Simplest Monte Carlo Path Tracer

For each pixel, cast n samples and average

- Choose a ray with $p=\text{camera}$, $d=(\theta, \phi)$ within pixel
- Pixel color $+= (1/n) * \text{TracePath}(p, d)$

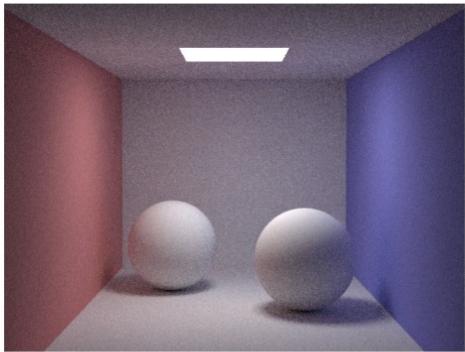
$\text{TracePath}(p, d)$ returns (r, g, b) [and calls itself recursively]:

- Trace ray (p, d) to find nearest intersection p'
- Select with probability (say) 50%:
 - Emitted: $\text{return } 2 * (\text{Le}_{\text{red}}, \text{Le}_{\text{green}}, \text{Le}_{\text{blue}}) // 2 = 1/(50\%)$
 - Reflected: $\text{generate ray in random direction } d'$
 $\text{return } 2 * f_r(d \rightarrow d') * (n * d') * \text{TracePath}(p', d')$

Path terminated when
Emission evaluated

14

Path Tracing



CS348B Lecture 14

10 paths / pixel

Pat Hanrahan, Spring 2009

15

Arnold Renderer (M. Fajardo)

- Works well on diffuse surfaces, hemispherical light



16

From CS 283(294) many years ago



Daniel Ritchie and Lita Cho

17

Advantages and Drawbacks

- Advantage: general scenes, reflectance, so on
 - By contrast, standard recursive ray tracing only mirrors
- This algorithm is *unbiased*, but horribly inefficient
 - Sample “emitted” 50% of the time, even if emitted=0
 - Reflect rays in random directions, even if mirror
 - If light source is small, rarely hit it
- Goal: improve efficiency without introducing bias
 - Variance reduction using many of the methods discussed for Monte Carlo integration last week
 - Subject of much interest in graphics in 90s till today

18

Outline

- Motivation and Basic Idea
- Implementation of simple path tracer
- *Variance Reduction: Importance sampling*
- Other variance reduction methods
- Specific 2D sampling techniques

19

Importance Sampling

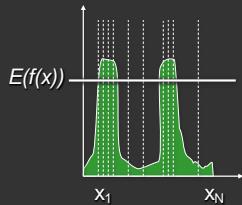
- Pick paths based on energy or expected contribution
 - More samples for high-energy paths
 - Don't pick low-energy paths
- At “macro” level, use to select between reflected vs emitted, or in casting more rays toward light sources
- At “micro” level, importance sample the BRDF to pick ray directions
- Tons of papers in 90s on tricks to reduce variance in Monte Carlo rendering
- Importance sampling now standard in production. I consulted on Pixar's system (~2011)

20

Importance Sampling

Can pick paths however we want, but contribution weighted by 1/probability

- Already seen this division of 1/prob in weights to emission, reflectance



$$\int_{\Omega} f(x) dx = \frac{1}{N} \sum_{i=1}^N Y_i$$

$$Y_i = \frac{f(x_i)}{p(x_i)}$$

21

Simplest Monte Carlo Path Tracer

For each pixel, cast n samples and average

- Choose a ray with $p=\text{camera}$, $d=(\theta, \phi)$ within pixel
- Pixel color $+= (1/n) * \text{TracePath}(p, d)$

TracePath(p, d) returns (r,g,b) [and calls itself recursively]:

- Trace ray (p, d) to find nearest intersection p'
- Select with probability (say) 50%:
 - Emitted:
 - return $2 * (L_{\text{red}}, L_{\text{green}}, L_{\text{blue}}) // 2 = 1/(50\%)$
 - Reflected:
 - generate ray in random direction d'
 - return $2 * f(d \rightarrow d') * (n \cdot d') * \text{TracePath}(p', d')$

22

Importance sample Emit vs Reflect

TracePath(p, d) returns (r,g,b) [and calls itself recursively]:

- Trace ray (p, d) to find nearest intersection p'
- If $L_e = (0,0,0)$ then $p_{\text{emit}} = 0$ else $p_{\text{emit}} = 0.9$ (say)
- If $\text{random}() < p_{\text{emit}}$ then:
 - Emitted:
 - return $(1/p_{\text{emit}}) * (L_{\text{red}}, L_{\text{green}}, L_{\text{blue}})$
 - Else Reflected:
 - generate ray in random direction d'
 - return $(1/(1-p_{\text{emit}})) * f(d \rightarrow d') * (n \cdot d') * \text{TracePath}(p', d')$

23

Importance sample Emit vs Reflect

TracePath(p, d) returns (r,g,b) [and calls itself recursively]:

- Trace ray (p, d) to find nearest intersection p'
- If $L_e = (0,0,0)$ then $p_{\text{emit}} = 0$ else $p_{\text{emit}} = 0.9$ (say)
- If $\text{random}() < p_{\text{emit}}$ then:
 - Emitted:
 - return $(1/p_{\text{emit}}) * (L_{\text{red}}, L_{\text{green}}, L_{\text{blue}})$
 - Else Reflected:
 - generate ray in random direction d'
 - return $(1/(1-p_{\text{emit}})) * f(d \rightarrow d') * (n \cdot d') * \text{TracePath}(p', d')$

24

Outline

- Motivation and Basic Idea
- Implementation of simple path tracer
- Variance Reduction: Importance sampling
- *Other variance reduction methods*
- Specific 2D sampling techniques

25

More variance reduction

- Discussed “macro” importance sampling
 - Emitted vs reflected
- How about “micro” importance sampling
 - Shoot rays towards light sources in scene
 - Distribute rays according to BRDF

26

One Variation for Reflected Ray

- Pick a light source
- Trace a ray towards that light
- Trace a ray anywhere except for that light
 - Rejection sampling
- Divide by probabilities
 - $1/(\text{solid angle of light})$ for ray to light source
 - $1/(2\pi - \text{solid angle})$ for non-light ray
 - So mult by solid angle for ray to light, and $2\pi - \text{solid angle}$ non-light ray
 - Use two rays so no extra factor of 2 needed

27

Russian Roulette

- Maintain current weight along path (need another parameter to TracePath)
- Terminate ray iff $|\text{weight}| < \text{const.}$
- Be sure to weight by $1/\text{probability}$

28

Russian Roulette

Terminate photon with probability p

Adjust weight of the result by $1/(1-p)$

$$E(X) = p \cdot 0 + (1-p) \frac{E(X)}{1-p} = E(X)$$

Intuition:

Reflecting from a surface with $R=0.5$

100 incoming photons with power 2 W

1. Reflect 100 photons with power 1 W
2. Reflect 50 photons with power 2 W

CS348B Lecture 14

Pat Hanrahan, Spring 2009

Path Tracing: Include Direct Lighting

Step 1. Choose a camera ray r given the (x, y, u, v, t) sample
weight = 1;

L = 0

Step 2. Find ray-surface intersection

Step 3.

```
L += weight * Lr(light sources)  
weight *= reflectance(r)  
Choose new ray r' ~ BRDF pdf(r)
```

Go to Step 2.

CS348B Lecture 14

Pat Hanrahan, Spring 2009

29

30

Monte Carlo Extensions

Unbiased

- Bidirectional path tracing
- Metropolis light transport

Biased, but consistent

- Noise filtering
- Adaptive sampling
- Irradiance caching

31

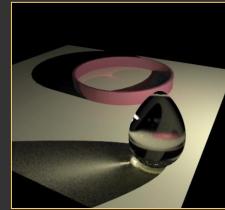
Monte Carlo Extensions

Unbiased

- Bidirectional path tracing
- Metropolis light transport

Biased, but consistent

- Noise filtering
- Adaptive sampling
- Irradiance caching



32

Monte Carlo Extensions

Unbiased

- Bidirectional path tracing
- Metropolis light transport

Biased, but consistent

- Noise filtering
- Adaptive sampling
- Irradiance caching



33

Monte Carlo Extensions

Unbiased

- Bidirectional path tracing
- Metropolis light transport

Biased, but consistent

- Noise filtering
- Adaptive sampling
- Irradiance caching



34

Monte Carlo Extensions

Unbiased

- Bidirectional path tracing
- Metropolis light transport



35

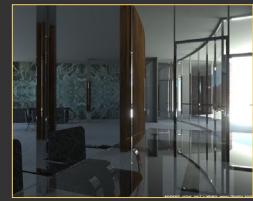
Monte Carlo Extensions

Unbiased

- Bidirectional path tracing
- Metropolis light transport

Biased, but consistent

- Noise filtering
- Adaptive sampling
- Irradiance caching



36

Monte Carlo Path Tracing Image



2000 samples per pixel, 30 computers, 30 hours

Jensen

37

Outline

- Motivation and Basic Idea
- Implementation of simple path tracer
- Variance Reduction: Importance sampling
- Other variance reduction methods
- *Specific 2D sampling techniques*

38

2D Sampling: Motivation

- Final step in sending reflected ray: sample 2D domain
- According to projected solid angle
- Or BRDF
- Or area on light source
- Or sampling of a triangle on geometry
- Etc.

39

Sampling Upper Hemisphere

- Uniform directional sampling: how to generate random ray on a hemisphere?
- Option #1: rejection sampling
 - Generate random numbers (x,y,z) , with x,y,z in $-1..1$
 - If $x^2+y^2+z^2 > 1$, reject
 - Normalize (x,y,z)
 - If pointing into surface ($\text{ray dot } n < 0$), flip

40

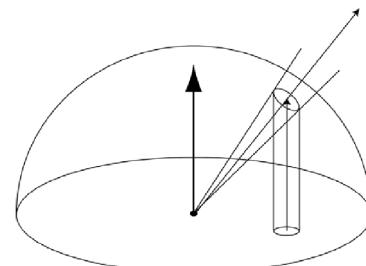
Sampling Upper Hemisphere

- Option #2: inversion method
 - In polar coords, density must be proportional to $\sin \theta$ (remember $d(\text{solid angle}) = \sin \theta \, d\theta \, d\phi$)
 - Integrate, invert $\rightarrow \cos^{-1}$
- So, recipe is
 - Generate ϕ in $0..2\pi$
 - Generate z in $0..1$
 - Let $\theta = \cos^{-1} z$
 - $(x,y,z) = (\sin \theta \cos \phi, \sin \theta \sin \phi, \cos \theta)$
- *This is what you need to do for homework 3 (simple upper hemisphere sampling). Anything more advanced (importance sampling later in lecture) is extra (homework 4).*

41

Sampling Projected Solid Angle

Generate cosine weighted distribution



CS348B Lecture 6

Pat Hanrahan, Spring 2004

42

BRDF Importance Sampling

- Better than uniform sampling: importance sampling
- Because you divide by probability, ideally probability proportional to $f_r * \cos \theta_i$

43

BRDF Importance Sampling

- For cosine-weighted Lambertian:
 - Density = $\cos \theta \sin \theta$
 - Integrate, invert $\rightarrow \cos^{-1}(\sqrt{z})$
- So, recipe is:
 - Generate ϕ in $0..2\pi$
 - Generate z in $0..1$
 - Let $\theta = \cos^{-1}(\sqrt{z})$

44

BRDF Importance Sampling

- Phong BRDF: $f_r \sim \cos^n \alpha$ where α is angle between outgoing ray and ideal mirror direction
- Constant scale = $k_s(n+2)/(2\pi)$
- Can't sample this times $\cos \theta_i$
 - Can only sample BRDF itself, then multiply by $\cos \theta_i$
 - That's OK – still better than random sampling

45

BRDF Importance Sampling

- Recipe for sampling specular term:
 - Generate z in $0..1$
 - Let $\alpha = \cos^{-1}(z^{1/(n+1)})$
 - Generate ϕ_α in $0..2\pi$
 - This gives direction w.r.t. ideal mirror direction
- Convert to (x,y,z) , then rotate such that z points along mirror dir.

46

Summary

- Monte Carlo methods robust and simple (at least until nitty gritty details) for global illumination
- Must handle many variance reduction methods in practice
- Importance sampling, Bidirectional path tracing, Russian roulette etc.
- Rich field with many papers, systems researched even over last 10 years

47