

## Computer Graphics

CSE 167 [Win 22], Lecture 19: High Quality Rendering  
Ravi Ramamoorthi

<http://viscomp.ucsd.edu/classes/cse167/wi22>

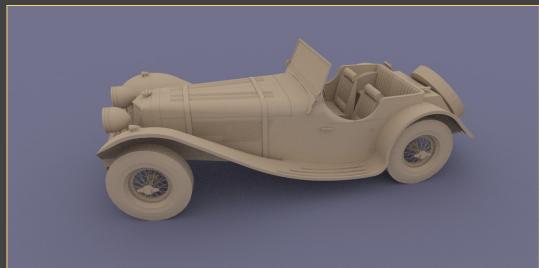
## Summary

- This is the final lecture of CSE 167. (CAPE+TA)
- Good luck on HW 4, written assignment
- Please consider CSE 168 (Rendering), 291 (Physical Simulation) in spring

### Monte Carlo Path Tracing

- 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
- Long history, traces back to rendering eqn Kajiya 86
- (More advanced topic: Slides from CSE 168/274)
- Increasingly, basis for production rendering
- Path tracing today real-time in hardware (for example, using NVIDIA's Optix, Turing RTX)

### Monte Carlo Path Tracing



Big diffuse light source, 20 minutes

Jensen

### Monte Carlo Path Tracing



1000 paths/pixel

Jensen

### Monte Carlo Path Tracing

#### Advantages

- Any type of geometry (procedural, curved, ...)
- Any type of BRDF or reflectance (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

## Monte Carlo Path Tracing

Integrate radiance for each pixel by sampling paths randomly

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

## Simplest Monte Carlo Path Tracer

For each pixel, cast n samples and average

- Choose a ray with  $p$ =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:  $\text{return } 2 * (\text{Le}_{\text{red}}, \text{Le}_{\text{green}}, \text{Le}_{\text{blue}}) // 2 = 1/(50\%)$
  - Reflected:  $\text{generate ray in random direction } d'$   
 $\text{return } 2 * f_r(d \rightarrow d') * (n \cdot d') * \text{TracePath}(p', d')$

## Simplest Monte Carlo Path Tracer

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

- Choose a ray with  $p$ =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:  $\text{return } 2 * (\text{Le}_{\text{red}}, \text{Le}_{\text{green}}, \text{Le}_{\text{blue}}) // 2 = 1/(50\%)$
  - Reflected:  $\text{generate ray in random direction } d'$   
 $\text{return } 2 * f_r(d \rightarrow d') * (n \cdot d') * \text{TracePath}(p', d')$

## Simplest Monte Carlo Path Tracer

For each pixel, cast n samples and average

- Choose a ray with  $p$ =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:  $\text{return } 2 * (\text{Le}_{\text{red}}, \text{Le}_{\text{green}}, \text{Le}_{\text{blue}}) // 2 = 1/(50\%)$
  - Reflected:  $\text{generate ray in random direction } d'$   
 $\text{return } 2 * f_r(d \rightarrow d') * (n \cdot d') * \text{TracePath}(p', d')$

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

## Simplest Monte Carlo Path Tracer

For each pixel, cast n samples and average

- Choose a ray with  $p$ =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:  $\text{return } 2 * (\text{Le}_{\text{red}}, \text{Le}_{\text{green}}, \text{Le}_{\text{blue}}) // 2 = 1/(50\%)$
  - Reflected:  $\text{generate ray in random direction } d'$   
 $\text{return } 2 * f_r(d \rightarrow d') * (n \cdot d') * \text{TracePath}(p', d')$

Path terminated when Emission evaluated

## Path Tracing

CS348B Lecture 14      10 paths / pixel      Pat Hanrahan, Spring 2009

## Arnold Renderer (M. Fajardo)

- Works well diffuse surfaces, hemispherical light



## From UCB CS 294 a few years ago



[Daniel Ritchie and Lila Che](#)

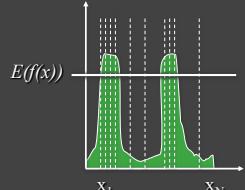
## Importance Sampling

- 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 for movies from 2012+

## 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



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

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

## 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  $Le = (0,0,0)$  then  $p_{emit} = 0$  else  $p_{emit} = 0.9$  (say)
- If  $random() < p_{emit}$  then:
  - Emitted:
    - return  $(1/p_{emit}) * (Le_{red}, Le_{green}, Le_{blue})$
  - Else Reflected:
    - generate ray in random direction  $d'$
    - return  $(1/(1-p_{emit})) * f_r(d \rightarrow d') * (n \cdot d') * TracePath(p', d')$

## 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

## 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  $r' \sim$  BRDF pdf( $r$ )
        Go to Step 2.
```

CS348B Lecture 14

Pat Hanrahan, Spring 2009

## Monte Carlo Extensions

### Unbiased

- Bidirectional path tracing
- Metropolis light transport

### Biased, but consistent

- Noise filtering
- Adaptive sampling
- Irradiance caching

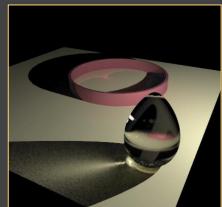
## Monte Carlo Extensions

### Unbiased

- Bidirectional path tracing
- Metropolis light transport

### Biased, but consistent

- Noise filtering
- Adaptive sampling
- Irradiance caching



RenderPark

## Monte Carlo Extensions

### Unbiased

- Bidirectional path tracing
- Metropolis light transport

### Biased, but consistent

- Noise filtering
- Adaptive sampling
- Irradiance caching



Heinrich

## Monte Carlo Extensions

### Unbiased

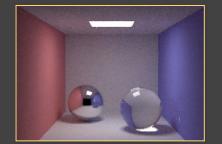
- Bidirectional path tracing
- Metropolis light transport

### Biased, but consistent

- Noise filtering
- Adaptive sampling
- Irradiance caching



Unfiltered Jensen



Filtered Jensen

## Monte Carlo Extensions

### Unbiased

- Bidirectional path tracing
- Metropolis light transport

### Biased, but consistent

- Noise filtering
- Adaptive sampling
- Irradiance caching



Fixed



Adaptive Ohbuchi

## Monte Carlo Extensions

Unbiased

- Bidirectional path tracing
- Metropolis light transport

Biased, but consistent

- Noise filtering
- Adaptive sampling
- Irradiance caching

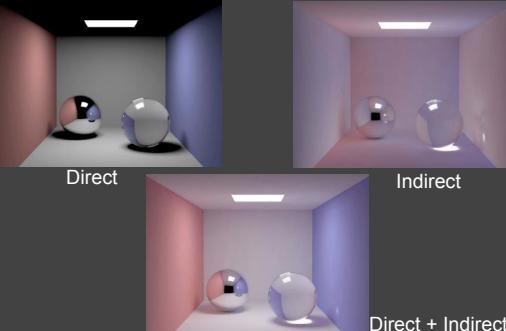


Jensen

## 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 over last 30 years
- Today, hardware for real-time ray, path tracing
- Promising physically-based GPU approach

## Smoothness of Indirect Lighting

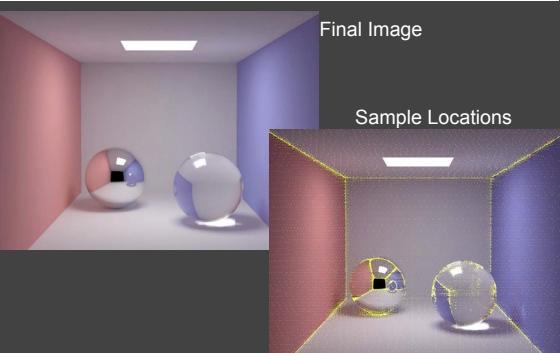


Direct      Indirect      Direct + Indirect

## Irradiance Caching

- Empirically, (diffuse) interreflections low frequency
- Therefore, should be able to sample sparsely
- Irradiance caching samples irradiance at few points on surfaces, and then interpolates
- Ward, Rubinstein, Clear. SIGGRAPH 88, *A ray tracing solution for diffuse interreflection*

## Irradiance Caching Example



Final Image      Sample Locations

## Stratified Sampling

**Stratified sampling like jittered sampling**  
**Allocate samples per region**

$$N = \sum_{i=1}^m N_i \quad F_N = \frac{1}{N} \sum_{i=1}^m N_i F_i$$

**New variance**

$$V[F_N] = \frac{1}{N^2} \sum_{i=1}^m N_i V[F_i]$$

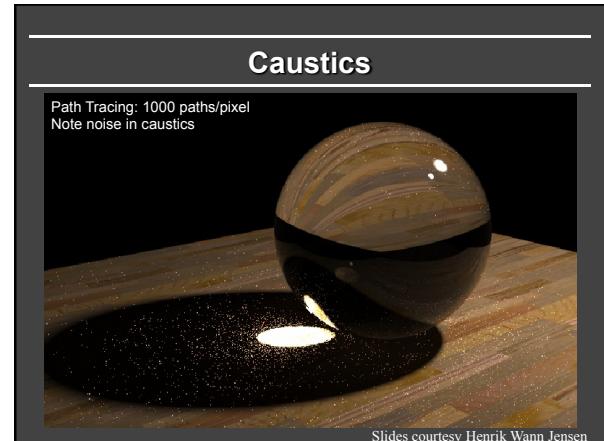
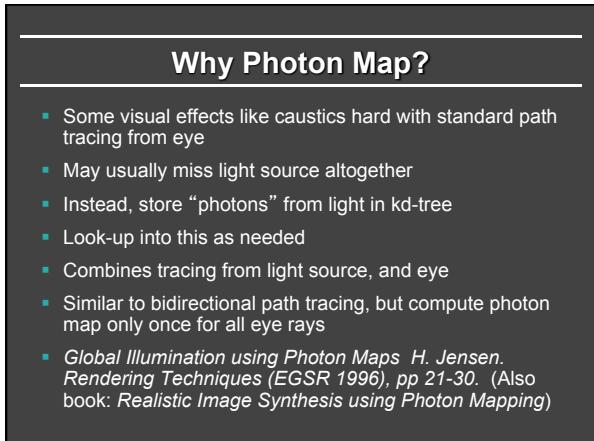
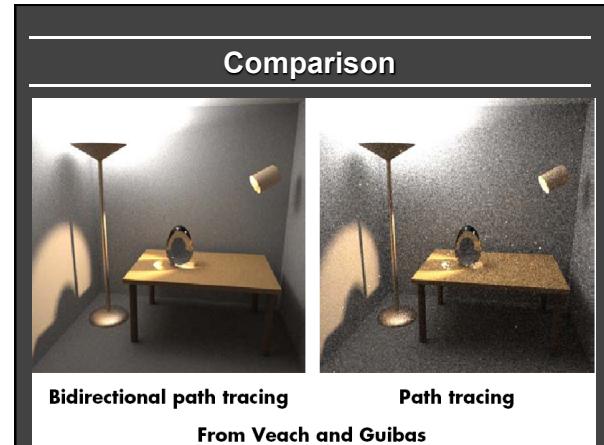
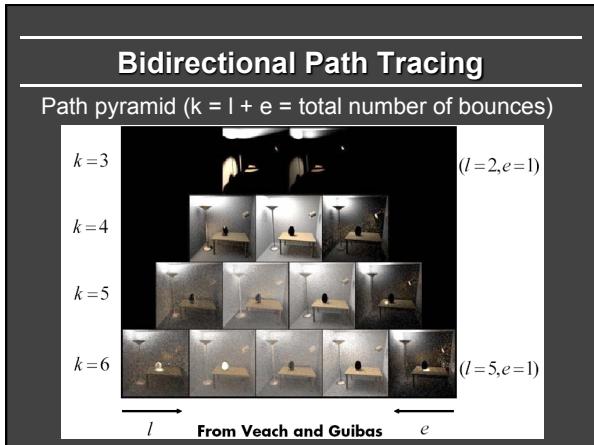
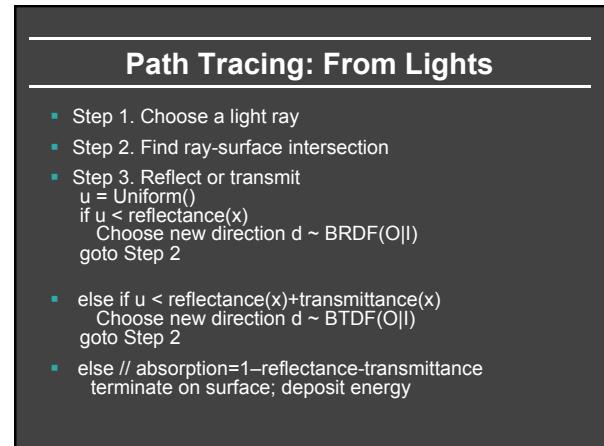
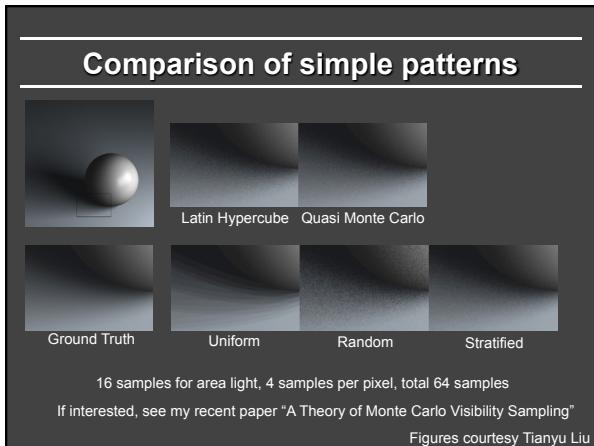
**Thus, if the variance in regions is less than the overall variance, there will be a reduction in resulting variance**

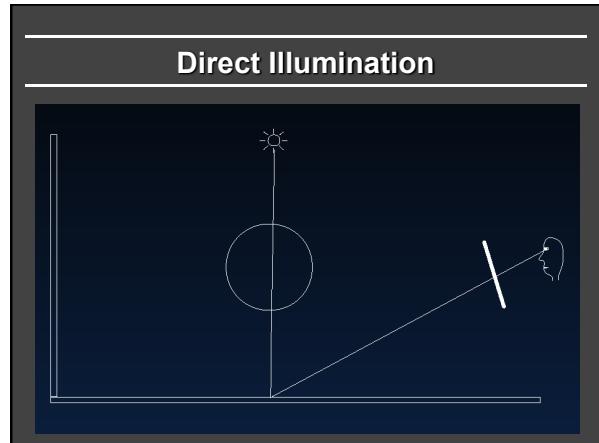
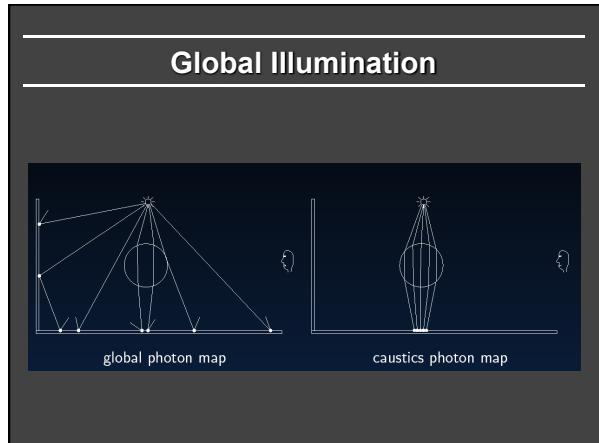
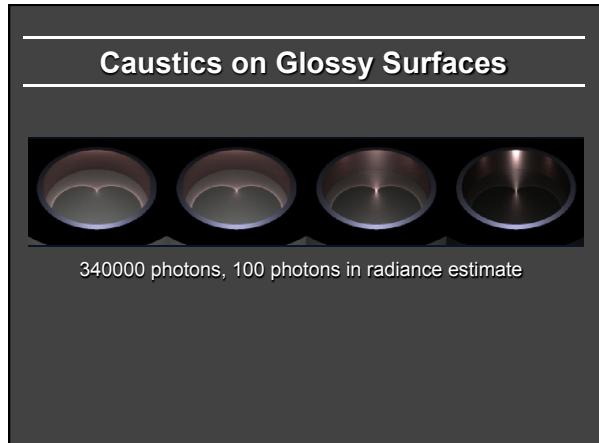
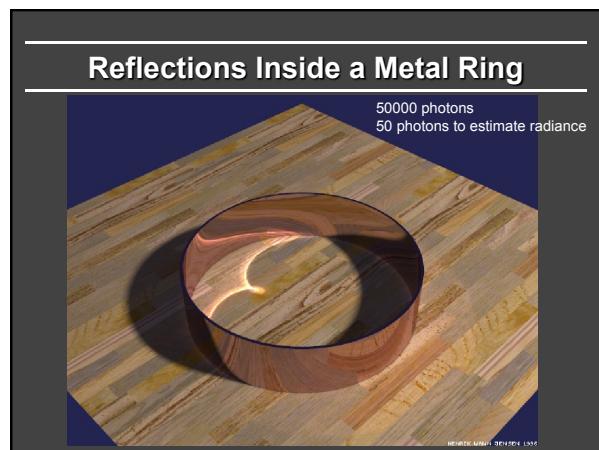
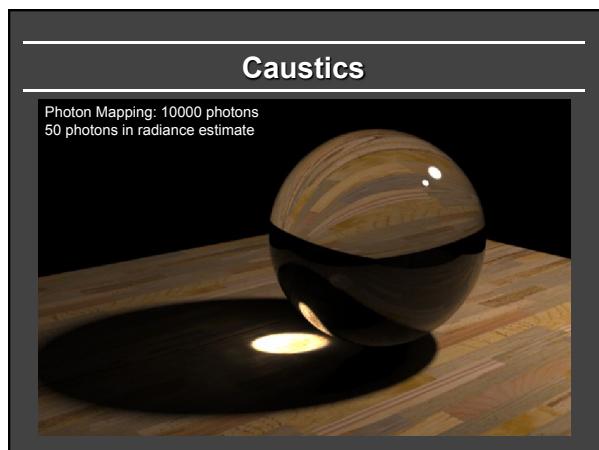
**For example: An edge through a pixel**

$$V[F_N] = \frac{1}{N^2} \sum_{i=1}^m V[F_i] = \frac{V[F_E]}{N^{1.5}}$$

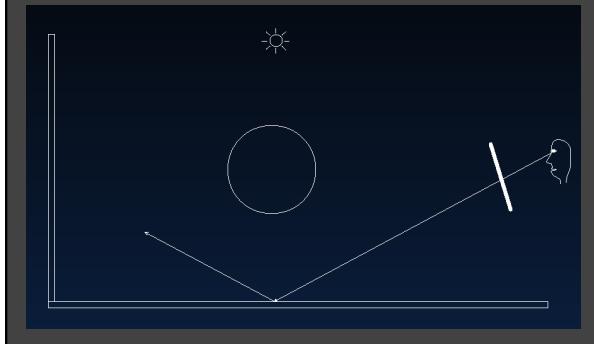
CS348B Lecture 9      Pat Hanrahan, Spring 2002

D. Mitchell 95, Consequences of stratified sampling in graphics

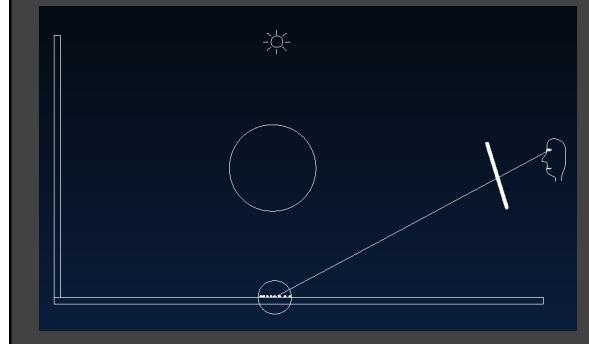




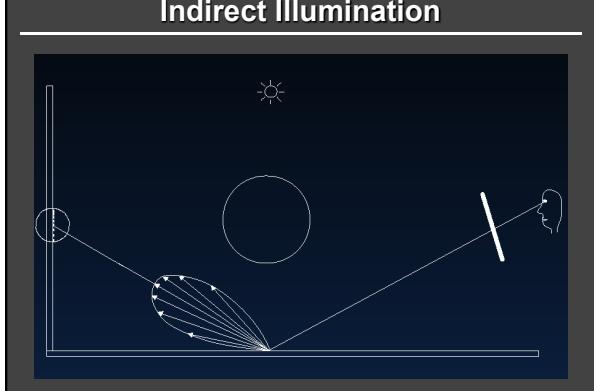
### Specular Reflection



### Caustics



### Indirect Illumination



### Mies House: Swimming Pool

