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

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

Big diffuse light source, 20 minutes

Monte Carlo Path Tracing

1000 paths/pixel
Monte Carlo Path Tracing

Integrate radiance for each pixel by sampling paths randomly.

\[
L_o(x, \omega) = L_e(x, \omega) + \int_{\Omega} f(x, \omega', \omega) L'_o(x, \omega') (\omega' \cdot \hat{n}) d\omega'
\]

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 \( += \frac{1}{n} \times \text{TracePath}(p, d) \)

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 \( 2 \times (L_{o,em}, L_{o,green}, L_{o,blue}) // 2 = 1/50(\%)) \)
  - Reflected:
    - generate ray in random direction \( d' \)
    - return \( 2 \times f(x, d, d') \times (n \cdot d') \times \text{TracePath}(p', d') \)

Path Tracing

CS546B Lecture 14
10 paths / pixel
Pat Hanrahan, Spring 2008
Arnold Renderer (M. Fajardo)
- Works well diffuse surfaces, hemispherical light

From UCB CS 294 a few years ago

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+

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

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

\[ \int f(x) dx = \frac{1}{N} \sum_{i=1}^{N} Y_i \]

\[ Y_i = \frac{f(x_i)}{p(x_i)} \]

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 \( L_e = (0,0,0) \) then \( p_{emit} = 0 \) else \( p_{emit} = 0.9 \) (say)
- If \( \text{random}() < p_{emit} \) then:
  - Emitted: return \( \frac{1}{p_{emit}} \cdot (L_{red}, L_{green}, L_{blue}) \)
  - Else Reflected: generate ray in random direction \( d' \):
    - return \( \frac{1}{p_{emit}} \cdot \frac{f(d \cdot d')}{n \cdot d'} \cdot \text{TracePath}(p', d') \)
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.
$L += \text{weight} \times L_r(\text{light sources})$
weight $= \text{reflectance}(r)$
Choose new ray $r' \sim \text{BRDF pdf}(r)$
Go to Step 2.

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
Monte Carlo Extensions

Unbiased
- Bidirectional path tracing
- Metropolis light transport

Biased, but consistent
- Noise filtering
- Adaptive sampling
- Irradiance caching

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
\[ N = \sum_{i=1}^{n} N_i \]
\[ F_i = \frac{1}{N} \sum_{j=1}^{N_i} F(j) \]

New variance
\[ \sigma_i^2 = \frac{1}{N_i} \sum_{j=1}^{N_i} \sigma_j^2 \]

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
\[ \sigma_i^2 = \frac{1}{N} \sum_{j=1}^{N} \sigma_j^2 = \frac{\sum_{j=1}^{N} \sigma_j^2}{N} \]

D. Mitchell 95, Consequences of stratified sampling in graphics
Comparison of simple patterns

16 samples for area light, 4 samples per pixel, total 64 samples
If interested, see my recent paper “A Theory of Monte Carlo Visibility Sampling”
Figures courtesy Tianyu Liu

Comparison

Path Tracing: From Lights

- Step 1. Choose a light ray
- Step 2. Find ray-surface intersection
- Step 3. Reflect or transmit
  - if \( u < \text{reflectance}(x) \)
  - Choose new direction \( d \sim \text{BRDF}(O|I) \)
  - goto Step 2
- else if \( u < \text{reflectance}(x) + \text{transmittance}(x) \)
  - Choose new direction \( d \sim \text{BTDF}(O|I) \)
  - goto Step 2
- else // absorption=1–reflectance-transmittance
  - terminate on surface; deposit energy

Bidirectional Path Tracing

Path pyramid (\( k = l + e = \text{total number of bounces} \))

Why Photon Map?

- Some visual effects like caustics hard with standard path tracing from eye
- May usually miss light source altogether
- Instead, store “photons” from light in kd-tree
- Look-up into this as needed
- Combines tracing from light source, and eye
- Similar to bidirectional path tracing, but compute photon map only once for all eye rays

Caustics

Path Tracing: 1000 paths/pixel
Note noise in caustics

Slides courtesy Henrik Wean Jensen
Caustics
Photon Mapping: 10000 photons
50 photons in radiance estimate

Reflections Inside a Metal Ring
50000 photons
50 photons to estimate radiance

Caustics on Glossy Surfaces
340000 photons, 100 photons in radiance estimate

HDR Environment Illumination

Global Illumination
global photon map  caustics photon map

Direct Illumination
Specular Reflection

Caustics

Indirect Illumination

Mies House: Swimming Pool