Real-Time High Quality Rendering
CSE 291 [Winter 2015], Lecture 5
Tour of Modern Offline Rendering

http://www.cs.ucsd.edu/~ravir

To Do
- Project milestone (1-2 pages), final project proposal

Motivation for Lecture
- Must understand modern offline rendering, before considering real-time extensions
- Papers discuss ways to accelerate some effects
- Many more interesting possibilities ahead

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
- Increasingly, basis for production rendering
- Path tracing today real-time in hardware (for example, using NVIDIA’s Optix)

Monte Carlo Path Tracing
Big diffuse light source, 20 minutes

Monte Carlo Path Tracing
1000 paths/pixel
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

Simplest Monte Carlo Path Tracer
For each pixel, cast n samples and average
- Choose a ray with \( p=\text{camera}, \; d=\langle \theta, \phi \rangle \) within pixel
- Pixel color \( = \frac{1}{n} \cdot \text{TracePath}(p, d) \)

\[ L_o(x, \hat{\omega}) = L_e(x, \hat{\omega}) + \int_{\Omega} \hat{f}^e(x, \hat{\omega}', \hat{\omega}) L_e(x, \hat{\omega}') d\hat{\omega} \]

Simplest Monte Carlo Path Tracer
For each pixel, cast n samples and average over paths
- Choose a ray with \( p=\text{camera}, \; d=\langle \theta, \phi \rangle \) within pixel
- Pixel color \( = \frac{1}{n} \cdot \text{TracePath}(p, d) \)

\[ L_o(x, \hat{\omega}) = L_e(x, \hat{\omega}) + \int_{\Omega} \hat{f}^e(x, \hat{\omega}', \hat{\omega}) L_e(x, \hat{\omega}') d\hat{\omega} \]

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:
    \[ \text{return } 2 \cdot (L_{rared}, L_{green}, L_{blue}) \]
    \[ = \frac{1}{50\%} \]
  - Reflected:
    \[ \text{generate ray in random direction } d' \]
    \[ \text{return } 2 \cdot f^r(d \rightarrow d') \cdot (n \cdot d') \cdot \text{TracePath}(p', d') \]
Arnold Renderer (M. Fajardo)
- Works well diffuse surfaces, hemispherical light

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 upcoming movies

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

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

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

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 random() < \( p_{emit} \) then:
  - Emitted:
    - \( (f' \cdot p_{emit}) \cdot (L_{red}, L_{green}, L_{blue}) \)
  - Else:
    - Reflect:
      - generate ray in random direction \( d’ \)
      - return \( (f' \cdot p_{emit}) \cdot f(d \cdot d') \) \( (n \cdot d') \) * TracePath(p', d')

From UCB CS 294 a few years ago
Daniel Ritchie and Lita Cho

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 random() < \( p_{emit} \) then:
  - Emitted:
    - \( (f' \cdot p_{emit}) \cdot (L_{red}, L_{green}, L_{blue}) \)
  - Else:
    - Reflect:
      - generate ray in random direction \( d’ \)
      - return \( (f' \cdot p_{emit}) \cdot f(d \cdot d') \) \( (n \cdot d’) \) * TracePath(p’, d’)

Path Tracing
10 paths / pixel
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.
- \( L += \text{weight} \times L(r) \)
- \( \text{weight} *= \text{reflectance}(r) \)
- 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

Monte Carlo Extensions

Unfiltered
- Noise filtering
- Adaptive sampling
- Irradiance caching

Filtered
- 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 10 years
- Today, hardware for real-time ray, path tracing
- Promising physically-based GPU approach

Smoothness of Indirect Lighting

- 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
**Stratified Sampling**

Stratified sampling like jitted sampling
Allocate samples per region
\[ N = \sum N_i \quad F_i = \frac{1}{N} \sum N_i \]
New variance
\[ \overline{V(F)} = \frac{1}{N} \sum \overline{V(F)} \]
Thus, if the variance in regions is less than the overall variance, there will be a reduction in resulting variance
For examples: An edge through a pixel
\[ \overline{V(F)} = \frac{1}{N} \sum \overline{V(F)} = \frac{\overline{V(F)}}{N} \]

---

**Comparison of simple patterns**

Ground Truth Uniform Random Stratified

Latin Hypercube Quasi Monte Carlo

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

---

**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 \( d \sim \text{BRDF}(O|I) \)
    - go to Step 2
  - else if \( u < \text{reflectance}(x) + \text{transmittance}(x) \)
    - Choose new direction \( d \sim \text{BTDF}(O|I) \)
    - go to Step 2
  - else // absorption = 1 – reflectance – transmittance
terminate on surface; deposit energy

---

**Bidirectional Path Tracing**

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

\[ \begin{align*}
  k = 3 & : (l = 1, e = 2) \\
  k = 4 & : (l = 2, e = 1) \\
  k = 5 & : (l = 3, e = 2) \\
  k = 6 & : (l = 4, e = 1)
\end{align*} \]

---

**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

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
**Direct Illumination**

**Specular Reflection**

**Caustics**

**Indirect Illumination**

**Mies House: Swimming Pool**

**Lightcuts**

- Efficient, accurate complex illumination

Environment map lighting & indirect
Time 111s

Textured area lights & indirect
Time 98s

(640x480, Anti-aliased, Glossy materials)

From Walter et al. SIGGRAPH 05
**Complex Lighting**

- Simulate complex illumination using point lights
  - Area lights
  - HDR environment maps
  - Sun & sky light
  - Indirect illumination

- Unifies illumination
- Enables tradeoffs between components

**Key Concepts**

- Light Cluster
- Light Tree
  - Binary tree of lights and clusters

**Three Example Cuts**

1. #1 #2 #4
2. #1 #3 #4
3. #1 #4

Tableau, 630K polygons, 13,000 lights, (EnvMap + Indirect)
Avg. shadow rays per eye ray 17 (0.13%)