To Do

- Homework 2 (Direct Lighting) due today!!
- Homework 3 (Path Tracer, Indirect Lighting) May 3
- Assignment is on UCSD Online
- START EARLY
- This lecture goes through details of indirect lighting, Monte Carlo path tracing for the assignment
- Ask re any questions

Indirect Lighting

- Core of path tracing, global illumination
- Supports multiple bounces of light, color bleeding
- General paths, general visual effects

Rendering Equation (Kajiya 86)

Paper introduced rendering equation, path tracing, importance sampling still used today.
Reflection Equation

\[ L_r(x, \omega_i) = L_e(x, \omega_i) + \int \omega \cdot \omega_i \cos \theta \, d\omega \]

Rendering Equation

\[ L_r(x, \omega_o) = L_e(x, \omega_o) + \int L_r(t(x, \omega_i), -\omega_i) f(x, \omega_i, \omega_o)(n \cdot \omega_i) d\omega_i \]

Path Construction

- Single path vs bushy tree
  - Conceptually simplest to render N 1-sample images
  - And then average them

Antialiasing within pixel for "free" (consider pixel having unit area, jitter ray in that, instead of shooting through midpoint)

Sampling Upper Hemisphere

- Uniform directional sampling: how to generate random ray on a hemisphere?
  - Option #1: rejection sampling
    - Generate 3 random numbers \((x, y, z)\), with \(x, y, z \in -1..1\)
    - If \(x^2+y^2+z^2 \leq 1\), reject
    - Normalize \((x, y, z)\)
    - If pointing into surface (ray dot \(n \leq 0\)), flip to -ray

Option #2: inversion method
  - In polar coords, density must be proportional to \(\sin \theta\)
  - Integrate, invert \(\cos^2 \theta\)

Recipe is (start with two random numbers \(\xi_1, \xi_2\) in \(0..1\))
  - \(\phi = \pi - 2\pi \xi_2\)
  - \(z = \xi_1\)
  - \(\theta = \cos^{-1} (z)\)
  - \((x, y, z) = (\sin \theta \cos \phi, \sin \theta \sin \phi, \cos \theta)\)
  - Rotate according to surface normal (\(z\) goes to normal)
    - Normal is \((\alpha, \beta)\) with \(\alpha = \cos(n_z)\) and \(\beta = \sin(n_x) \cos(n_y)\)
    - Rotation matrix \(R = R_y(\beta)R_x(\alpha)\) then do \(R^*(x, y, z)\)
Sampling Upper Hemisphere

- Two random numbers $\xi_1, \xi_2$ in 0..1
  - Generate $\phi$ in 0..$2\pi$: $\phi = 2\pi \xi_2$
  - Generate $z$ in 0..1: $z = \xi_1$
  - Let $\theta = \cos^{-1} z$
  - $\phi, \theta$ in $0 \ldots \pi$ and $z = \sin \theta \cos \phi, \sin \theta \sin \phi, \cos \theta$

- Rotate according to surface normal ($z$ goes to normal)
  - Normal is $(\alpha, \beta)$ with $\alpha = \cos^{-1}(n_z)$ and $\beta = \tan(2(n_x, n_z))$
  - Rotation matrix $R = R_\alpha R_\beta$ then do $R^*(x, y, z)$

Create Local Coordinate Frame

- First, compute $u, v, w$ to create orthonormal frame
  - Vector $a$ is arbitrary (use random or up vector)
  - Be careful when $a$ close to $n$, use alternative vector
    - $w = n$
    - $u = a \times w$
    - $v = w \times u$

- Now, compute ray direction $\omega$
  - $(x, y, z)$ are scalar coordinates; $u, v, w$ are vectors above
  - $\omega = xu + yv + zw$

Or Create Local Coordinate Frame

- Simpler, may be useful for texture etc.
  - Can use any one of 3 methods (rejection, rotation, coordinate frame but assignment spec coord. frame)

Assignment so far (checkpoint 1)

- Sample hemisphere at each bounce
  - Evaluate full MC estimator with $N = 1$ for each ray
  - Upto depth $D = 5$. Final ray $D = 5$ returns emit $L_e$ only
  - Most rays will actually be 0 (do not hit light source)
  - Very inefficient, but render this, will improve on it next

1 sample per pixel

64 samples per pixel (may be slow)
Separating Direct/Indirect

- Also called next event estimation (NEE)
- Already know how to do direct (homework 2)
  - By sampling/integrating area light source
  - But vanilla path tracing previously is very inefficient
  - Chance of hitting the light source is very small
- So separate direct and indirect
  - Estimate "next event" on light source for direct
  - Focus energies on "hard" indirect light vs "easy" direct
- Simplest of variance reduction methods
  - Monte Carlo Path tracing always works, is gold standard
  - But challenge is making it fast, removing noise

Formally split incident light at a point:

\[ L(x,\omega_i) = L_d(x,\omega_i) + L_i(x,\omega_i) \]

Reflected light has emission, direct, indirect:

\[ L_r(x,\omega_o) = L_e(x,\omega_o) + L_d(x,\omega_o) + L_I(x,\omega_o) \]

Emission is easy, and we already know direct:

\[ L_e(x,\omega_o) = \frac{2\pi}{N} \sum_{k=1}^{N} L_{o}(t(x,\omega_i(k),-\omega_i(k))(n_i \omega_i(k))) \]

Indirect is now evaluated by path tracing:

\[ L_i(x,\omega_o) = \frac{2\pi}{N} \sum_{k=1}^{N} L_{o}(t(x,\omega_i(k),-\omega_i(k))(n_i \omega_i(k))) \]

\[ L_d(x,\omega_o) \approx \frac{2\pi}{N} \sum_{k=1}^{N} L_{o}(t(x,\omega_i(k),-\omega_i(k))(n_i \omega_i(k))) \]

Implementation: Corner Cases

- Emission from first intersected surface (light sources) should be added, but no emission on subsequent bounces
- Since next event estimation / direct light effectively extends path by a bounce, trace indirect ray to depth D = 1
- Render Cornell box 1 spp, 64 spp D = 5, single unstratified direct light sample per intersection

Separating Direct/Indirect: Notes

\[ L_i(x,\omega_o) = \int L_{\omega_{in}}(x,\omega) f(x,\omega,\omega_o) n_i \omega_i d\omega_i \]

\[ = \frac{2\pi}{N} \sum_{k=1}^{N} L_{o}(t(x,\omega_i(k),-\omega_i(k))(n_i \omega_i(k))) \]

- Note that \( L_i \) above = \( L_d + L_i \) only(not \( L_d \), no emission)

Implementation

- At each intersection in path tracer, execute direct lighting
  - For simplicity, only one (unstratified) ray for each area light
  - Ultimately, we will average many primary samples
- Add in emission where appropriate (light sources only)
- Execute indirect lighting above (randomly sample path)
- To avoid double counting, indirect rays don’t see emission
  - If an indirect ray ever strikes a light source, terminate immediately
  - Without accumulating the light source’s emission

1 sample per pixel (no NEE)

1 sample per pixel (with NEE)
64 samples per pixel (without NEE)

64 samples per pixel (with NEE)

**Russian Roulette**
- Clipping to fixed depth $D$ undesirable
  - Leads to bias, some complex paths need high $D$
  - Continue ray even when throughput is very small
  - In practice, rays may terminate if exit scene, but this can’t formally be guaranteed (hall of mirrors, closed box)
- Russian roulette unbiased at infinite depth
  - Terminate (probabilistically) low throughput paths
  - Increase energy of paths kept alive

**Choosing Probability**
- Choose probability $q$ inversely on throughput
  $$q = 1 - \min\{\max(T_r, T_g, T_b), 1\}$$
- Russian Roulette applied (only) in indirect
  - Determine direct (and emission on first bounce) as usual (no boosting or termination is applied)
  - Then find throughput for ray so far ($BRDF, \cosine, 2\pi$ terms product each bounce), pick random number in $0...1$
    - If number < $q$ terminate (no indirect ray is shot)
    - Otherwise, boost throughput by $1/(1-q)$, shoot indirect

**Russian Roulette Termination**
- Terminate path with some probability $q$
  - If terminated, obviously throughput is 0
  - If left alive, multiply (boost) throughput $T$ by $1/(1-q)$
  - Create fewer higher-energy paths (e.g. if $q = 0.1$, 10 equal paths reduces to 9 (expected) each 10/9 energy. If instead $q = 0.9$, reduce to 1 path with 10 times energy)
  - Keep total energy constant, unbiased $(0q + (1-q)/(1-q))$
  - Probability $q$ controls how aggressive termination (depends on throughput, can increase variance)

**Russian Roulette Images**
- $D = 5$, 16 samples
- $D = \infty$, 16 samples