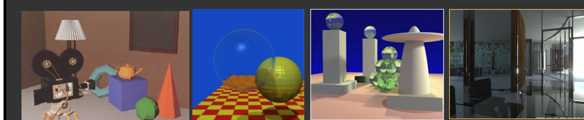
Computer Graphics II: Rendering

CSE 168 [Spr 26], Lecture 8: Indirect Lighting Details
Ravi Ramamoorthi

http://viscomp.ucsd.edu/classes/cse168/sp26



1

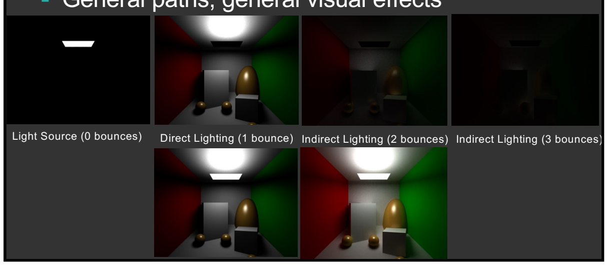
To Do

- Homework 2 (Direct Lighting) due today!!
- Homework 3 (Path Tracer, Indirect Lighting) May 5
- Assignment is on edX edge
- START EARLY
- This lecture goes through details of indirect lighting, Monte Carlo path tracing for the assignment
- Ask re any questions

2

Indirect Lighting

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



3

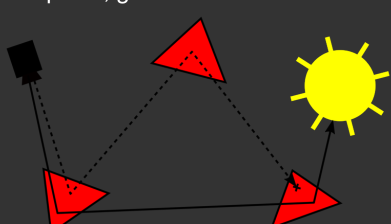
Indirect Lighting

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

4

Indirect Lighting

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



5

Rendering Equation (Kajiya 86)

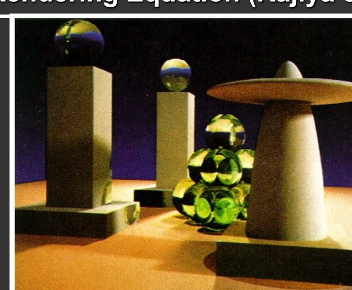
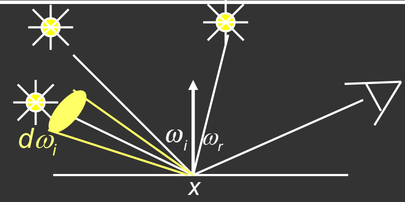


Figure 6. A sample image. All objects are neutral grey. Color on the objects is due to caustics from the green glass balls and color bleeding from the base polygon.

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

6

Reflection Equation



Replace sum with integral

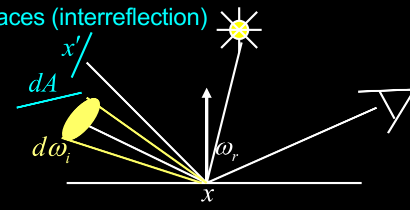
$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_{\Omega} L_i(x,\omega_i) f(x,\omega_i,\omega_r)\cos\theta_i \, d\omega_i$$

Reflected Light (Output Image) | Emission | Incident Light (from light source) | BRDF | Cosine of Incident angle

7

Rendering Equation

Surfaces (interreflection)



$$\omega_i \sim x' - x$$

$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_{\Omega} L_r(x',-\omega_i) f(x,\omega_i,\omega_r)\cos\theta_i \, d\omega_i$$

Reflected Light (Output Image) UNKNOWN | Emission KNOWN | Reflected Light UNKNOWN | BRDF KNOWN | Cosine of Incident angle KNOWN

8

Rendering Equation

- Assignment: slight change in notations

$$L_r(x,\omega_o) = L_e(x,\omega_o) + \int_{\Omega} L_r(t(x,\omega_i),-\omega_i) f(x,\omega_i,\omega_o)(n \bullet \omega_i) d\omega_i$$

$x' = t(x,\omega_i)$ is the raycasting function to first intersection

- Monte Carlo estimator (hemisphere, not area light)
  - Randomly generate sample on hemisphere (total 2π steradians)

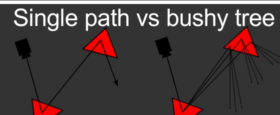
$$L_r(x,\omega_o) \approx L_e(x,\omega_o) + \frac{2\pi}{N}\sum_{k=1}^{N} L_r(t(x,\omega_i(k)),-\omega_i(k)) f(x,\omega_i(k),\omega_o)(n \bullet \omega_i(k))$$

- Not ideal; each L_r call recursively estimated
  - Can lead to exponential growth in samples, termination condition
  - *Set fixed depth D = 5 to guarantee termination for now*
- Instead, consider single path without splitting
  - N = 1 after primary visibility or first bounce (all N for first bounce)
  - Actually render N images, average (Single path vs "bushy tree")

9

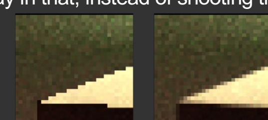
Path Construction

- 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)



10

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 $x^2+y^2+z^2 > 1$, reject
  - Normalize (x,y,z)
  - If pointing into surface (ray dot n < 0), flip to -ray

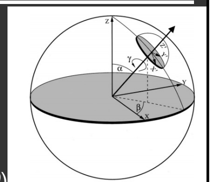
11

Sampling Upper Hemisphere

- Option #2: inversion method
  - In polar coords, density must be proportional to sin θ (remember d(solid angle) = sin θ $d\theta\, d\phi$)
  - Integrate, invert ➔ $\cos^{-1}$
- Recipe is (start with two random numbers ξ_1, ξ_2 in 0…1)
  - Generate ϕ in 0..2π $\phi=2\pi\xi_2$
  - Generate z in 0..1 $z=\xi_1$
  - Let $\theta = \cos^{-1} z$ $\theta=\mathrm{acos}(\xi_1)$
  - $(x,y,z) = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$
- Rotate according to surface normal (z goes to normal)
  - Normal is (α,β) with $\alpha = \mathrm{acos}(n_z)$ and $\beta=\mathrm{atan2}(n_y,n_x)$
  - Rotation matrix $R = R_z(\beta)R_y(\alpha)$ then do $R^*(x,y,z)$

12

Sampling Upper Hemisphere



- Two random numbers ξ_1, ξ_2 in 0…1
  - Generate ϕ in 0..2π $\phi=2\pi\xi_2$
  - Generate z in 0..1 $z=\xi_1$
  - Let $\theta = \cos^{-1} z$ $\theta=\mathrm{acos}(\xi_1)$
  - $(x,y,z) = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$
- Rotate according to surface normal (z goes to normal)
  - Normal is (α,β) with $\alpha = \mathrm{acos}(n_z)$ and $\beta=\mathrm{atan2}(n_y,n_x)$
  - Rotation matrix $R = R_z(\beta)R_y(\alpha)$ then do $R^*(x,y,z)$

$$R=\begin{pmatrix}\cos\beta & -\sin\beta & 0\\ \sin\beta & \cos\beta & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\alpha & 0 & \sin\alpha\\ 0 & 1 & 0\\ -\sin\alpha & 0 & \cos\alpha\end{pmatrix}=\begin{pmatrix}\cos\alpha\cos\beta & -\sin\beta & \sin\alpha\cos\beta\\ \sin\beta\cos\alpha & \cos\beta & \sin\alpha\sin\beta\\ -\sin\alpha & 0 & \cos\alpha\end{pmatrix}$$

13

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)

$$\|u\| = \|v\| = \|w\| = 1$$
$$u \bullet v = v \bullet w = u \bullet w = 0$$
$$w = u \times v$$

$$p = (p \bullet u)u + (p \bullet v)v + (p \bullet w)w$$

- Associate w with normal (+z = n). Need u, v

14

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

$$w = \frac{n}{\|n\|}$$
$$u = \frac{a \times w}{\|a \times w\|}$$
$$v = w \times u$$

- Now, compute ray direction ω
  - (x,y,z) are scalar coordinates; u,v,w are vectors above

$$\boldsymbol{\omega} = x\mathbf{u} + y\mathbf{v} + z\mathbf{w}$$

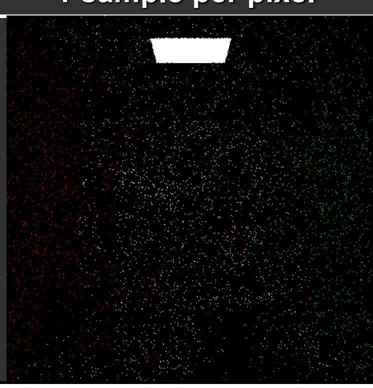
15

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

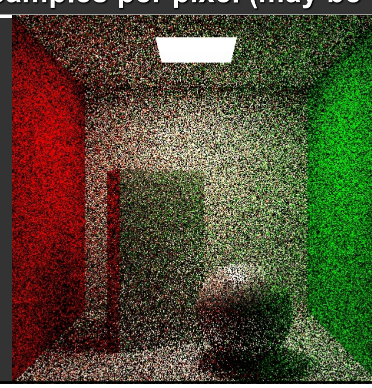
16

1 sample per pixel



17

64 samples per pixel (may be slow)



18

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

19

Separating Direct/Indirect

- Formally split incident light at a point $L_i(x,\omega_i) = L_{dir}(x,\omega_i) + L_{ind}(x,\omega_i)$
- 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

$$L_d(\mathbf{x},\omega_o) \approx L_e \frac{A}{N} \sum_{k=1}^{N} f(\mathbf{x},\omega_i(k),\omega_o)\, G(\mathbf{x},\mathbf{x'}_k)\, V(\mathbf{x},\mathbf{x'}_k)$$

- Indirect is now evaluated by path tracing

$$L_i(x,\omega_o) = \int_\Omega L_{ind}(x,\omega_i) f(x,\omega_i,\omega_o)(n \bullet \omega_i)\, d\omega_i$$

$$\approx \frac{2\pi}{N} \sum_{k=1}^{N} L_o(t(x,\omega_i(k)),-\omega_i(k)) f(x,\omega_i(k),\omega_o)(n \bullet \omega_i(k))$$

20

Separating Direct/Indirect: Notes

$$L_i(x,\omega_o) = \int_\Omega L_{ind}(x,\omega_i) f(x,\omega_i,\omega_o)(n \bullet \omega_i)\, d\omega_i$$

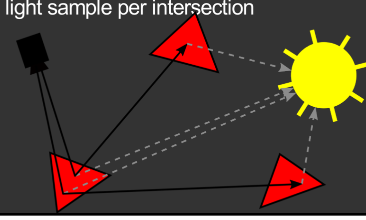
$$\approx \frac{2\pi}{N} \sum_{k=1}^{N} L_o(t(x,\omega_i(k)),-\omega_i(k)) f(x,\omega_i(k),\omega_o)(n \bullet \omega_i(k))$$

- Note that L_o above = L_d + L_I only(not L_r: 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

21

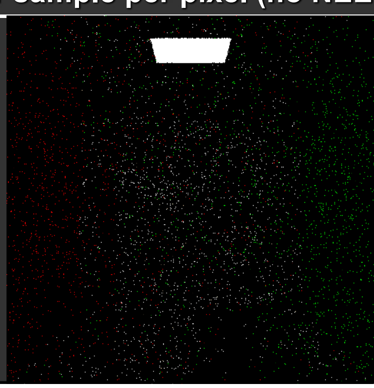
Implementation: Corner Cases

- 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



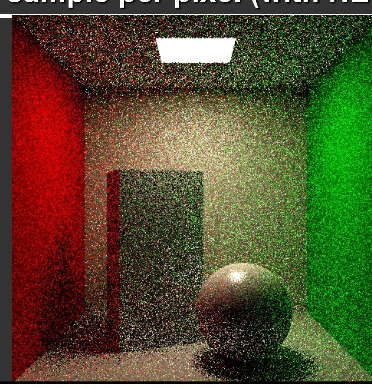
22

1 sample per pixel (no NEE)



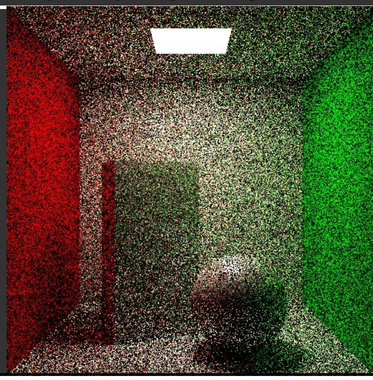
23

1 sample per pixel (with NEE)



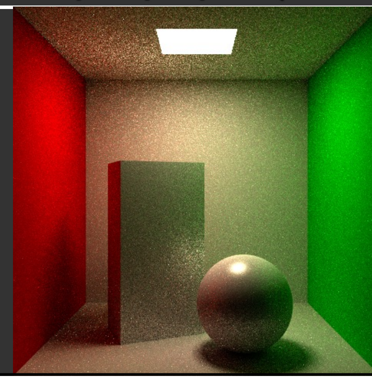
24

64 samples per pixel (without NEE)



25

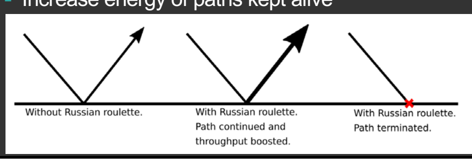
64 samples per pixel (with NEE)



26

Russian Roulette

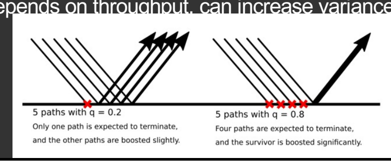
- Clipping to fixed depth D undesirable
  - Leads to bias, some complex paths need high D
  - Continue ray even when throughput is very small
  - In practice, rays may terminate if exit scene, but this can't formally be guaranteed (hall of mirrors, closed box)
- Russian roulette unbiased at infinite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



27

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 (0*q + (1-q)/(1-q))
  - Probability q controls how aggressive termination (depends on throughput, can increase variance)



28

Choosing Probability

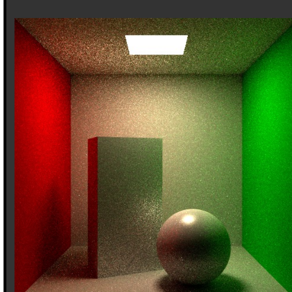
- Choose probability q inversely on throughput

$$q = 1 - \min\left(\max\left(T_r, T_g, T_b\right), 1\right)$$

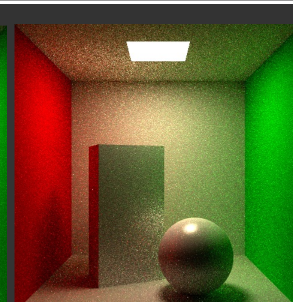
- Russian Roulette applied (only) in indirect
  - Determine direct (and emission on first bounce) as usual (no boosting or termination is applied)
  - Then find throughput for ray so far (BRDF, cosine, 2π terms product each bounce), pick random number in 0…1
    - If number < q terminate (no indirect ray is shot)
    - Otherwise, boost throughput by 1/(1-q), shoot indirect

29

Russian Roulette Images



D = 5, 16 samples



D = infinity, 16 samples

30