Lecture 17:
Global Illumination 2 &
Path Tracing

Computer Graphics and Imaging
UC Berkeley CS184/284A
Direct lighting + recursive specular reflection / refraction
+ all multiple bounces of light = Global Illumination
Recall: Reflection Operator

\[ R(g)(p, \omega_o) \equiv \int_{H^2} \mathcal{f}_r(p, \omega_i \rightarrow \omega_o) g(p, \omega_i) \cos \theta_i \, d\omega_i \]

Incoming radiance (surface light field)

Outgoing radiance (surface light field)
Recall: Transport Operator

Outgoing radiance (surface light field)

Incoming radiance (surface light field)

\[ T(f)(p, \omega_o) \equiv f(tr(p, \omega), -\omega) \]
Recall: Rendering Equation in Operator Notation

\[ L_o(p, \omega_o) = L_e(p, \omega_o) + \int_{H^2} f_r(p, \omega_i \rightarrow \omega_o) L_o(tr(p, \omega_i), -\omega_i) \cos \theta_i \, d\omega_i \]

Define full one-bounce light transport operator: \( K = R \circ T \)

\[ L_o = L_e + K(L_o) \]
Solving the Rendering Equation
Solving the Rendering Equation

• Rendering equation:

\[ L = L_e + K(L) \]
\[ (I - K)(L) = L_e \]

\[ L = (I - K)^{-1}(L_e) \]

• Solution desired:

\[ L = L_e + K(L) \]
\[ (I - K)(L) = L_e \]

\[ L = (I - K)^{-1}(L_e) \]

• How to solve?
Solution Intuition

For scalar functions, recall:

\[
\frac{1}{1 - x} = 1 + x + x^2 + x^3 + \cdots
\]

converges for \(-1 < x < 1\)

Similarly, for operators, it is true that

\[
(I - K)^{-1} = \frac{1}{I - K} = I + K + K^2 + K^3 + \cdots
\]

(Neumann series)

converges for \(\|K\| < 1\)

where \(\|K\| < 1\) means that the “energy” of the radiance function decreases after applying \(K\). This is intuitively true for valid scene models based on energy dissipation (though not trivial to prove, see Veach & Guibas).
Formal Solution

Neumann series:

\[(I - K)^{-1} = \frac{1}{I - K} = I + K + K^2 + K^3 + \cdots\]

Check:

\[(I - K) \circ (I - K)^{-1}\]

\[= (I - K) \circ (I + K + K^2 + K^3 + \cdots)\]

\[= (I + K + K^2 + \cdots) - (K + K^2 + \cdots)\]

\[= I\]

Again, energy dissipation makes it possible to show that the series converges.
Rendering Equation Solution

\[ L = (I - K)^{-1}(L_e) \]

\[ = (I + K + K^2 + K^3 + \cdots)(L_e) \]

\[ = L_e + K(L_e) + K^2(L_e) + K^3(L_e) + \cdots \]

Intuitive: Sum of successive bounces of light

This calculates the steady-state surface light field over the scene.
$L_e$
$K(L_e)$
\( (K \circ K \circ K)(L_e) \)
\((K \circ K \circ K \circ K \circ K)(L_e)\)
\((K \circ K \circ K \circ K \circ K \circ K \circ K)(L_e)\)
\[ \sum_{i=0}^{0} K^i(L_e) \]
\[ \sum_{i=0}^{1} K^i(L_e) \]
\[
\sum_{i=0}^{2} K^i(L_e)
\]
\[ \sum_{i=0}^{3} K^i(L_e) \]
\[ \sum_{i=0}^{4} K^i(L_e) \]
\[ \sum_{i=0}^{5} K^i(L_e) \]
\[ \sum_{i=0}^{6} K^i(L_e) \]
Direct illumination
One-bounce global illumination
Two-bounce global illumination
Four-bounce global illumination
Eight-bounce global illumination
Sixteen-bounce global illumination
Light Paths
1-Bounce Path Connecting Ray to Light

Camera

Light
1-Bounce Paths Connecting Ray to Light
2-Bounce Path Connecting Ray to Light

Camera

Viewpoint

Viewing window

Pixel

Traced ray

Light
2-Bounce Paths Connecting Ray to Light

Camera

Light

Viewpoint

Pixel

Viewing window

Traced ray

CS184/284A

Ren Ng
2-Bounce Paths Connecting Ray to Light
3-Bounce Paths Connecting Ray to Light

Camera

Light

Viewpoint

Pixel

Viewing window

Traced ray

CS184/284A

Ren Ng
3-Bounce Path Connecting Ray to Light
3-Bounce Path Connecting Ray to Light
3-Bounce Path Connecting Ray to Light

Camera

Light

Viewing window
Pixel
Traced ray

CS184/284A
Ren Ng
3-Bounce Path Connecting Ray to Light

Camera

Light
3-Bounce Path Connecting Ray to Light

Camera

Light

Viewpoint

Pixel

Viewing window

Traced ray

CS184/284A

Ren Ng
3-Bounce Path Connecting Ray to Light

Camera

Light

Viewpoint

Pixel

Traced ray

Viewing window
Discussion: Global Illumination Rendering

Sum over all paths of all lengths

Challenges, discuss:

• How to generate all possible paths?
• How to sample space of paths efficiently?
Sum Over Paths
Try 1: Monte Carlo Sum over Paths

```
EstRadianceIn(x, ω)
    p = intersectScene(x, ω);
    L = p.emittedLight(-ω);
    wi, pdf = p.brdf.sampleDirection();
    L += EstRadianceIn(p, wi) * p.brdf(wi, -ω) * costheta / pdf;
    return L;
```

- Note:
  - Importance sampling BRDF
  - Infinite recursion!
Problem: Infinite Bounces of Light

How to integrate over infinite dimensions?

• Note: if energy dissipates, contribution of higher bounces decreases exponentially

Idea: just use N bounces

• Problem: biased! No matter how many Monte Carlo samples, never see light taking N+1 to infinity bounces

Idea: probabilistic termination?

• Non-zero probability of sampling paths of arbitrarily high number of bounces

• Surprisingly, can design this to be unbiased — this is called Russian Roulette
Russian Roulette: Unbiased Random Termination

New estimator: evaluate original estimator with probability $p_{rr}$, reweighted. Otherwise ignore.

Let $X_{rr} = \begin{cases} \frac{X}{p_{rr}}, & \text{with probability } p_{rr} \\ 0, & \text{otherwise} \end{cases}$

Same expected value as original estimator:

$$E[X_{rr}] = p_{rr} E\left[\frac{X}{p_{rr}}\right] + (1 - p_{rr}) E[0] = E[X]$$

Want to choose $p_{rr}$ considering Monte Carlo efficiency

- Terminate if expensive and/or low contribution
- In path tracing, expensive to recursively trace path. Increase termination probability if brdf is low in next bounce direction
An unbiased, finite estimator for infinite dimensional integral!
Try 2: Russian Roulette Monte Carlo over Paths

```python
EstRadianceIn(x, ω)
    p = intersectScene(x, ω);
    L = p.emittedLight(–ω);
    wi, pdf = p.brdf.sampleDirection();
    cpdf = continuationProbability(p.brdf, wi);
    if (random01() < cpdf)  // Russian Roulette
        L += EstRadianceIn(p, wi)  // Recursion
            * p.brdf(wi, –ω) * costheta / pdf / cpdf;
    return L;

// Unbiased, computation terminates, but still extremely noisy!
```
Recall: Importance Sampling

Solid angle sampling

Light area sampling
Path Tracing
Path Tracing Overview

Terminate paths randomly with Russian Roulette

Partition the recursive radiance evaluation. At each point on light path

- Direct lighting – non-recursive, importance sample lights
- Indirect lighting – recursive, importance sample BRDF

Monte Carlo estimate for each partition separately

- Possible to take just one sample for each
- Assume: 100s - 1000s of paths sampled per pixel
Partitioning the Rendering Equation

\[
\text{EstRadianceIn}(x, \omega) = \text{EstRadianceOut}(p, -\omega)
\]
Partitioning the Rendering Equation

Need to sum paths going through $p$ representing 0, 1, 2, 3, ... bounces of light
Partitioning the Rendering Equation

At p, consider light contributions from paths of varying bounce-length

- 0-bounce: light emitted from p (p is on a light source)
Partitioning the Rendering Equation

At $p$, consider light contributions from paths of varying bounce-length
- 0-bounce: light emitted from $p$ ($p$ is on a light source)
- 1-bounce: from light to $p$ to $x$ ("direct illumination")

1-bounce: from light to $p$ to $x$ ("direct lighting")
Partitioning the Rendering Equation

At \( p \), consider light contributions from paths of varying bounce-length:

- 0-bounce: light emitted from \( p \) (\( p \) is on a light source)
- 1-bounce: from light to \( p \) to \( x \) ("direct illumination")
- >1-bounce: from light to at least one other point to \( p \) to \( x \) ("indirect illumination")
Consider Evaluation of >1 Bounce of Light

At $p$, consider light contributions from paths of varying bounce-length
- 0-bounce: light emitted from $p$ ($p$ is on a light source)
- 1-bounce: from light to $p$ to $x$ ("direct illumination")
- >1-bounce: from light to at least one other point to $p$ to $x$ ("indirect illumination")
Path Tracing Pseudocode

```plaintext
EstRadianceIn(x, ω)       // incoming at x from dir ω
    p = intersectScene(x, ω);
    return ZeroBounceRadiance(p, –ω)
        + AtLeastOneBounceRadiance(p, –ω);

ZeroBounceRadiance(p, wo)  // outgoing at p in dir ω
    return p.emittedLight(wo);
```

Ren Ng
Path Tracing Pseudocode

AtLeastOneBounceRadiance(p, wo)  // out at p, dir wo
    L = OneBounceRadiance(p, wo);  // direct illum

    wi, pdf = p.brdf.sampleDirection();  // Imp. sampling
    p' = intersectScene(p, wi);
    cpdf = continuationProbability(p.brdf, wi, wo);
    if (random01() < cpdf)  // Russ. Roulette
        L += AtLeastOneBounceRadiance(p', -wi)  // Recursive est. of
           * p.brdf(wi, wo) * costheta / pdf / cpdf;  // indirect illum
    return L;

OneBounceRadiance(p, wo)  // out at p, dir wo
    return DirectLightingSampleLights(p, wo);  // direct illum
DirectLightingSamplingLights(p, \omega_o)
    L, \omega_i, pdf = lights.sampleDirection(p);                // Imp. sampling

    if (scene.shadowIntersection(p, \omega_i))                  // Shadow ray
        return 0;
    else
        return L * p.brdf(\omega_i, \omega_o) * \text{costheta} / pdf;

// Note: only one random sample over all lights.
// Assignment 3-A asks you to, alternatively, loop over
// multiple lights and take multiple samples (later slide)
One sample per pixel
32 samples per pixel
1024 samples per pixel
Summary of Intuition on Global Illumination & Path Tracing
Summary of Intuition on G.I. & P.T.

- Operator notation leads to insight that solution is adding successive bounces of light
- Trace N paths through a pixel, sample radiance
- Build paths by recursively tracing to next surface point and choosing a random reflection direction. At each surface, sum emitted light and reflected light
- How to terminate paths? We use Russian Roulette to kill probabilistically.
- How to reduce noise? Use importance sampling in choosing random direction. Two ways: importance sample the lights, and importance sample the BRDF.
Implementation Notes
Paths vs Trees

4 eye rays per pixel
16 shadow rays per eye ray
(68 ray traces per pixel)

64 eye rays per pixel
1 shadow ray per eye ray
(128 ray traces per pixel)
64 image samples x 1 light sample, 13.2 seconds
8 image samples x 8 light samples, 7.7 seconds
Multiple Light Sources

Consider multiple lights in direct lighting estimate

One strategy:

• Loop over all N lights, sum Monte-Carlo estimates for each light

• For each light: compute Monte Carlo estimate with M samples taken with importance sampling

Needs N * M samples
This is what the assignment asks you to implement.
Multiple Light Sources (Single Sample)

Consider random sampling of multiple lights with a single sample

- Randomly choose light i, with probability $p_i$
- Randomly sample over that light’s directions, with probability $p_L$
- Probability of choosing sample is $(p_i \times p_L)$
- Weight the lighting calculation by $1/(p_i \times p_L)$
- Is this estimator unbiased? Yes!
- How would you importance sample intelligently?

Can of course average N such samples
Point Lights / Ideal Specular Materials – Issues

Sampling problems

• When sampling directions randomly, we have zero probability of matching exact direction of a point light or mirror reflection / specular refraction

Remedy

• In direct lighting, importance sample point lights by generating a single sample pointing directly at the light (only one sample needed)

• In indirect lighting, importance sample specular BRDFs by generating a sample point directly along the specular refraction / transmission direction
Numerical Precision Issues

Consider calculating ray-intersection with a distant sphere

$C=(1930.420,1973.505), \ R=1$
Numerical Precision Issues

C=(1930.420,1973.505) R=1

True Intersection: (1929.7203..., 1972.7897...)
Computed Intersection: (1930.4196..., 1973.5054...)
Noisy Shadows

- Camera ray
- Shadow ray falsely intersects the same surface
- Computed surface intersection
- Surface
Noisy shadows because of floating point precision problems
Floating-Point Precision Remedies

1. double (fp64) rather than float (fp32)
   - 53-bits of precision instead of 24-bits
   - Increase memory footprint

2. Ignore re-intersection with the last object hit
   - Only works for flat objects (e.g. triangles)
   - No help if model has coincident triangles

3. Offset origin along ray to ignore close intersections
   - Hard to choose offset that isn’t too small or too big
Remedy: Project Intersection Point to Surface

Project intersection point to the closest point on the surface
Good Scenes for Path Tracing (Diffuse, Sky Lighting)

Model by KiKe Oliva

M. Fajardo, Arnold Path Tracer
Good Scenes for Path Tracing (Diffuse, Sky Lighting)

M. Fajardo, Arnold Path Tracer
Good Scenes for Path Tracing (Diffuse, Sky Lighting)

M. Fajardo, Arnold Path Tracer
Good Scenes for Path Tracing (Diffuse, Sky Lighting)

Street scene 1
1536x654, 16 paths/pixel, 2 bounces, 250,000 faces, 18 min., dual PIII-800

M. Fajardo, Arnold Path Tracer
A Challenging Scene for Path Tracing – Why?

1000 paths / pixel

Henrik Wann Jensen
Things to Remember

Global illumination challenge: recursive light transport

Reflection & rendering equations, operator notation

Neumann solution of rendering equation
  • Sum successive bounces of light, infinite series

Path tracing
  • Russian Roulette for unbiased finite estimate of infinite series (infinite dimensional integral)
  • Partition into direct and indirect illumination
  • Importance sampling of lighting and BRDF
Acknowledgments

Thanks to Matt Pharr, Pat Hanrahan and Kayvon Fatahalian for many of these slides. Thanks also to Steve Marschner for the path tracer code progression sequence.

Thanks to Weilun Sun for rendering the Cornell Box with successive bounces of light.

Thanks to Ben Mildenhall for suggestions to improve many slides.