Lecture 13:
Global Illumination & Path Tracing

Computer Graphics and Imaging
UC Berkeley CS184/284A
Course Roadmap

Rasterization Pipeline
Core Concepts
• Sampling
• Antialiasing
• Transforms

Geometric Modeling
Core Concepts
• Splines, Bezier Curves
• Topological Mesh Representations
• Subdivision, Geometry Processing

Lighting & Materials
Core Concepts
• Measuring Light
• Unbiased Integral Estimation
• Light Transport & Materials

Radiometry & Photometry
Monte Carlo Integration
Global Illumination & Path Tracing
Material Modeling

Intro to Geometry
Curves and Surfaces
Geometry Processing
Ray-Tracing & Acceleration

Rasterization
Transforms & Projection
Texture Mapping
Visibility, Shading, Overall Pipeline

Transforms
Sampling
Antialiasing

Sampling
Antialiasing
Transforms
Hard Shadows

Image credit: Henrik Wann Jensen
Soft Shadows
Caustics
+ Inter-Reflections = Global Illumination
Visual Richness from Indirect Lighting
Visual Richness from Complex Lighting

- Point Light
- Environment Map Lighting
Visual Richness from Complex Materials

Credit: Bertrand Benoit. “Sweet Feast,” 2009. [Blender /VRay]
Photograph (CCD) vs. global illumination rendering
Ray Tracer Samples Radiance Along A Ray

The light entering the pixel is the sum total of the light reflected off the surface into the ray’s (reverse) direction.
Mini-Intro To Material Reflection
Reflection

Definition: reflection is the process by which light incident on a surface interacts with the surface such that it leaves on the incident (same) side without change in frequency
Categories of Reflection Functions

Ideal specular
- Perfect mirror reflection

Ideal diffuse
- Equal reflection in all directions

Glossy specular
- Majority of light reflected near mirror direction

Retro-reflective
- Light reflected back towards light source

Diagrams illustrate how light from incoming direction is reflected in various outgoing directions.
Materials: Mirror
Materials: Diffuse
Materials: Gold
Materials: Plastic
Materials: Red Semi-Gloss Paint
Materials: Ford Mystic Lacquer Paint
Reflection at a Point

Differential irradiance incoming: \[ dE(\omega_i) = L(\omega_i) \cos \theta_i \, d\omega_i \]

Differential radiance exiting (due to \( dE(\omega_i) \)) \[ dL_r(\omega_r) \]
**BRDF**

Definition: The bidirectional reflectance distribution function (BRDF) represents how much light is reflected into each outgoing direction $\omega_r$ from each incoming direction.

\[
dL_r(x, \omega_r) = \frac{dL_r(\omega_r)}{dE_i(\omega_i)} = \frac{dL_r(\omega_r)}{L_i(\omega_i) \cos \theta_i \, d\omega_i} \left[ \frac{1}{\text{sr}} \right]
\]

**NB:** $\omega_i$ points away from surface rather than into surface, by convention.
The Reflection Equation

\[ L_r(p, \omega_r) = \int_{H^2} f_r(p, \omega_i \rightarrow \omega_r) L_i(p, \omega_i) \cos \theta_i \, d\omega_i \]
Solving the Reflection Equation

\[ L_r(p, \omega_r) = \int_{H^2} f_r(p, \omega_i \rightarrow \omega_r) L_i(p, \omega_i) \cos \theta_i \, d\omega_i \]

Monte Carlo estimate:

- Generate directions \( \omega_j \) sampled from some distribution \( p(\omega) \)
- Choices for \( p(\omega) \)
  - Uniformly sample hemisphere
  - Importance sample BRDF (proportional to BRDF)
  - Importance sample lights (sample position on lights)
- Compute the estimator

\[
\frac{1}{N} \sum_{j=1}^{N} \frac{f_r(p, \omega_j \rightarrow \omega_r) L_i(p, \omega_j) \cos \theta_j}{p(\omega_j)}
\]
Recall: Hemisphere vs Light Sampling

Sample hemisphere uniformly

Sample points on light
Direct Lighting Pseudocode (Uniform Random Sampling)

DirectLightingSampleUniform(p, wo)
    wi = hemisphere.sampleUniform();  // uniform random sampling
    pdf = 1.0 / (2 * pi);

    if (scene.shadowIntersection(p, wi))  // Shadow ray
        return 0;
    else
        L = lights.radiance(intersect(p, wi), -wi);
        return L * p.brdf(wi, wo) * costheta / pdf;
Direct Lighting Pseudocode (Importance Sampling of BRDF)

DirectLightingSampleBRDF(p, wo)
  wi, pdf = p.brdf.sampleDirection(wo); // Imp. Sample BRDF

  if (scene.shadowIntersection(p, wi)) // Shadow ray
    return 0;
  else
    L = lights.radiance(intersect(p, wi), -wi);
    return L * p.brdf(wi, wo) * costheta / pdf;
Direct Lighting Pseudocode (Importance Sampling of Lights)

DirectLightingSampleLights(p, wo)
    L, wi, pdf = lights.sampleDirection(p);  // Imp. sampl lights

    if (scene.shadowIntersection(p, wi))  // Shadow ray
        return 0;
    else
        return L * p.brdf(wi, wo) * costheta / pdf;

    // Note: only one random sample over all lights.
    // Assignment 3-1 asks you to, alternatively, loop over
    // multiple lights and take multiple samples
Global Illumination:  
Deriving the Rendering Equation
Again: Reflection Equation

\[ L_r(p, \omega_r) = \int_{H^2} f_r(p, \omega_i \rightarrow \omega_r) L_i(p, \omega_i) \cos \theta_i \, d\omega_i \]
Challenge: This is Actually A Recursive Equation

Reflected radiance depends on incoming radiance

$$L_r(p, \omega_r) = \int_{H^2} f_r(p, \omega_i \rightarrow \omega_r) L_i(p, \omega_i) \cos \theta_i \, d\omega_i$$

But incoming radiance depends on reflected radiance (at another point in the scene)
Transport Function & Radiance Invariance

Definition: the Transport Function, $tr(p, \omega)$, returns the first surface intersection point in the scene along ray $(p, \omega)$

Radiance invariance along rays: $L_o(tr(p, \omega_i), -\omega_i) = L_i(p, \omega_i)$

“Radiance arriving at $p$ from direction $\omega_i$ is equal to the radiance leaving $p'$ in direction $-\omega_i$”
The Rendering Equation

Re-write the reflection equation:

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

Using the transport function: \[ L_i(p, \omega_i) = L_o(tr(p, \omega_i), -\omega_i) \]

The Rendering Equation

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

Note: recursion is now explicit

How to solve?
Light Transport Operators
Operators Are Higher-Order Functions

Functions:

\[ f, g : (x, \omega) \rightarrow \mathbb{R} \]

Operators are higher-order functions:

\[ P : ((x, \omega) \rightarrow \mathbb{R}) \rightarrow ((x, \omega) \rightarrow \mathbb{R}) \]

\[ P(f) = g \]

• Take a function and transform it into another function
Linear Operators

- Linear operators act on functions like matrices act on vectors

\[ h(x) = (L(f))(x) \]

- They are linear in that:

\[ L(af + bg) = aL(f) + bL(g) \]

- Examples of linear operators:

\[ H(f)(x) = \int h(x, x') f(x') \, dx' \]

\[ D(f)(x) = \frac{\delta f}{\delta x}(x) \]
Light Transport Functions & Operators

- Emitted radiance function
  (all surface points & outgoing directions)

- Incoming/outgoing reflected radiance
  (all surface points & in/out directions)

- Transport function - returns the first scene intersection point along given ray

- Reflection operator:
  \[ R(g)(p, \omega_o) \equiv \int_{H^2} f_r(p, \omega_i \rightarrow \omega_o) g(p, \omega_i) \cos \theta_i \, d\omega_i \]

- Transport operator:
  \[ T(f)(p, \omega_o) \equiv f(tr(p, \omega), -\omega) \]

\[ L_e(p, \omega) \]
\[ L_i(p, \omega), \quad L_o(p, \omega) \]
\[ tr(p, \omega) \]
Reflection Operator

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

Incoming radiance (surface light field)

\[ L_i \]

Outgoing radiance (surface light field)

\[ L_o \]
Transport Operator

Outgoing radiance (surface light field)

\[ T(f)(p, \omega_o) \equiv f(tr(p, \omega_o), -\omega_o) \]

\[ T(L_o) = L_i \]

Incoming radiance (surface light field)
Rendering Equation in Operator Notation

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

\[ L_o = L_e + (R \circ T)(L_o) \]

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

• Solution desired:
\[ L = (I - K)^{-1}(L_e) \]

• 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

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

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

(\text{Neumann series})

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

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

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

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

\[ \uparrow \uparrow \uparrow \uparrow \]

Emitted 1-bounce 2-bounce 3-bounce

Intuitive: Sum of successive bounces of light

This calculates the steady-state surface light field over the scene.
$L_e$
$K(L_e)$
$(K \circ K)(L_e)$
\( (K \circ K \circ K)(L_e) \)
\((K \circ K \circ K \circ K)(L_e)\)
\((K \circ K \circ K \circ K \circ K \circ K \circ K)(L_e)\)
\[ \sum_{i=0}^{0} K^i(L_e) \]
\[ \sum_{i=0}^{1} K_i(Le) \]
\[ \sum_{i=0}^{2} K^i(L_e) \]
\[ \sum_{i=0}^{3} K^i(L_e) \]
\[ \sum_{i=0}^{4} K^i(L_e) \]
\[ \sum_{i=0}^{5} K^i(L_e) \]
$\sum_{i=0}^{6} K^i(L_e)$
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

Camera

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

Camera

Viewing window

Pixel

Traced ray

Light
3-Bounce Paths Connecting Ray to Light
3-Bounce Path Connecting Ray to Light
3-Bounce Path Connecting Ray to Light
3-Bounce Path Connecting Ray to Light

Camera

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

Camera

Light
Global Illumination Rendering

Sum over all paths of all lengths

Challenges:
- 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)

\[ p = \text{intersectScene}(x, \omega); \]
\[ L = p.\text{emittedLight}(-\omega); \]
\[ \omega_i, \text{pdf} = p.\text{brdf.sampleDirection}(-\omega); \]
\[ L += \text{EstRadianceIn}(p, \omega_i) * p.\text{brdf}(\omega_i, -\omega) * \text{costheta} / \text{pdf}; \]

return L;

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

How to integrate over infinite dimensions?

• Note: if energy dissipates, contributions from higher bounces decrease exponentially

Idea: just use N bounces

• Problem: biased! No matter how many Monte Carlo samples, never see light taking N+1 to infinity bounces
Russian Roulette
Russian Roulette - Unbiased Random Termination

Idea: probabilistic termination of recursion

• At every recursive step (every bounce of light), probabilistically choose to stop the recursion
  • Specifically, continue with probability $p_{rr}$
• This goes from an infinite recursion, to a finite number of recursive function calls (how many?)
• But won’t this bias our Monte Carlo integral estimate of the infinite bounces of light?
• Surprisingly, no! We can adjust the Monte Carlo estimator so that it remains unbiased (next slide)
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 we expect 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 an infinite dimensional integral!
Try 2: Russian Roulette Monte Carlo over Paths

EstRadianceIn(x, \(\omega\))

\[
\begin{align*}
p &= \text{intersectScene}(x, \omega); \\
L &= p.\text{emittedLight}(-\omega); \\
\omega_i, \text{pdf} &= p.\text{brdf}\.\text{sampleDirection}(-\omega); \\
\text{cpdf} &= \text{continuationProbability}(p.\text{brdf}, \omega_i); \\
\text{if} \ (\text{random01}() < \text{cpdf}) & \quad \text{// Russian Roulette} \\
\quad \text{// Recursion} \\
L &= L + \text{EstRadianceIn}(p, \omega_i) * p.\text{brdf}(\omega_i, -\omega) \* \text{costheta} / \text{pdf} / \text{cpdf}; \\
\text{return} \ L;
\end{align*}
\]

// 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 \(-\omega\) ("direct lighting")

1-bounce: from light to p to x ("direct illumination")
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 p to \(-\omega\) ("indirect lighting")
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

\[
\text{EstRadianceIn}(x, \omega) \quad \text{// incoming at } x \text{ from dir } \omega \\
p = \text{intersectScene}(x, \omega); \\
\text{return ZeroBounceRadiance}(p, -\omega) \\
+ \text{AtLeastOneBounceRadiance}(p, -\omega);
\]

\[
\text{ZeroBounceRadiance}(p, \omega) \quad \text{// outgoing at } p \text{ in dir } \omega \\
\text{return } p.\text{emittedLight(}\omega)\text{;}
\]
Path Tracing Pseudocode

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

    wi, pdf = p.brdf.sampleDirection(wo);  // 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
Direct Lighting Pseudocode (Lights)

DirectLightingSamplingLights(p, wo)
    L, wi, pdf = lights.sampleDirection(p); // Imp. sampling

    if (scene.shadowIntersection(p, wi)) // Shadow ray
        return 0;
    else
        return L * p.brdf(wi, wo) * costheta / pdf;

// Note: only one random sample over all lights.
// Assignment 3-1 asks you to, alternatively, loop over
// multiple lights and take multiple samples (later slide)
Direct Lighting (0 and 1 Bounce Only)
Path Tracing (All Bounces of Light)
One sample (path) per pixel
32 samples (paths) per pixel
1024 samples (paths) 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
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 single 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

Surface

Computed surface intersection

Shadow ray falsely intersects same 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.