Lecture 3 - “Light Transport”

Welcome!

\[ I(x', x'') = g(x, x') \left[ \epsilon(x, x') + \int_s \rho(x, x', x'') I(x', x'') dx'' \right] \]
Today's Agenda:

- Introduction
- The Rendering Equation
- Light Transport
Introduction

Whitted
Introduction

Whitted
Introduction

Whitted

Missing:

- Area lights
- Glossy reflections
- Caustics
- Diffuse interreflections
- Diffraction
- Polarization
- Phosphorescence
- Temporal effects
- Motion blur
- Depth of field
- Anti-aliasing
Introduction

Anti-aliasing

Adding anti-aliasing to a Whitted-style ray tracer:

Send multiple primary rays through each pixel, and average their result.

Problem:

- How do we aim those rays?
- What if all rays return the same color?
Introduction

Anti-aliasing – Sampling Patterns

Adding anti-aliasing to a Whitted-style ray tracer:

Send multiple primary rays through each pixel, and average their result.

Problem:

- How do we aim those rays?
- What if all rays return the same color?
Introduction

Anti-aliasing – Sampling Patterns
Introduction

Anti-aliasing – Sampling Patterns
Introduction

Anti-aliasing – Sampling Patterns
Introduction

Anti-aliasing – Sampling Patterns

Adding anti-aliasing to a Whitted-style ray tracer:

Send multiple primary rays through each pixel, and average their result.

Problem:

- How do we aim those rays?
- What if all rays return the same color?

More info: https://mynameismjp.wordpress.com/2012/10/24/msaa-overview
Introduction

Whitted

Missing:

- Area lights
- Glossy reflections
- Caustics
- Diffuse interreflections
- Diffraction
- Polarization
- Phosphorescence
- Temporal effects
- Motion blur
- Depth of field
✓ Anti-aliasing
Introduction

Distribution Ray Tracing*

*: Distributed Ray Tracing, Cook et al., 1984
Introduction

Distribution Ray Tracing*

Glossy reflections

*: Distributed Ray Tracing, Cook et al., 1984
Introduction

Distribution Ray Tracing*

*: Distributed Ray Tracing, Cook et al., 1984
Introduction

Distribution Ray Tracing

*: Distributed Ray Tracing, Cook et al., 1984
Introduction

Distribution Ray Tracing

Whitted-style ray tracing is a \textit{point sampling} algorithm:

- We may miss small features
- \textbf{We cannot sample areas}

Area sampling:

- Anti-aliasing: one pixel
- Soft shadows: one area light source
- Glossy reflection: directions in a cone
- Diffuse reflection: directions on the hemisphere
Area Lights

Visibility of an area light source:

$$V_A = \int_A V(x, \omega_i) \, d\omega_i$$

Analytical solution case 1:

$$V_A = A_{\text{light}} - A_{\text{light} \cap \text{sphere}}$$

Analytical solution case 2:

$$V_A = ?$$
Approximating Integrals

An integral can be approximated as a Riemann sum:

\[
V_A = \int_A^B f(x) \, dx \approx \sum_{i=1}^{N} f(t_i) \Delta_i, \text{ where } \sum_{i=1}^{N} \Delta_i = B - A
\]

Note that the intervals do not need to be uniform, as long as we sample the full interval. If the intervals are uniform, then

\[
\sum_{i=1}^{N} f(t_i) \Delta_i = \Delta_i \sum_{i=1}^{N} f(t_i) = \frac{B - A}{N} \sum_{i=1}^{N} f(t_i).
\]

Regardless of uniformity, restrictions apply to \( N \) when sampling multi-dimensional functions (ideally, \( N = M^d, M \in \mathbb{N} \)). Also note that aliasing may occur if the intervals are uniform.
Introduction

Monte Carlo Integration

Alternatively, we can approximate an integral by taking random samples:

$$V_A = \int_A^B f(x) \, dx \approx \frac{B - A}{N} \sum_{i=1}^{N} f(X_i)$$

Here, $X_1, \ldots, X_N \in [A, B]$.

As $N$ approaches infinity, $V_A$ approaches the expected value of $f(X)$.

Unlike in Riemann sums, we can use arbitrary $N$ for Monte Carlo integration, regardless of dimension.
Monte Carlo Integration of Area Light Visibility

To estimate the visibility of an area light source, we take $N$ random point samples.

In this case, 5 out of 6 samples are unoccluded:

$$V \approx \frac{1}{6} (1 + 1 + 1 + 0 + 1 + 1) = \frac{5}{6}$$

Properly formulated using a MC integrator:

$$V = \int_{S^2} V(p) \, dp \approx \frac{1}{N} \sum_{i=1}^{N} V(P)$$

With a small number of samples, the variance in the estimate shows up as noise in the image.
Introduction

Distribution Ray Tracing

Key concept of distribution ray tracing:
We estimate integrals using Monte Carlo integration.

Integrals in rendering:

- Area of a pixel
- Lens area (aperture)
- Frame time
- Light source area
- Cones for glossy reflections
- Wavelengths
- Many lights
- ...
Today's Agenda:

- Introduction
- The Rendering Equation
- Light Transport
Whitted, Cook & Beyond

Missing in Whitted:

- Area lights
- Glossy reflections
- Caustics
- Diffuse interreflections
- Diffraction
- Polarization
- Phosphorescence
- Temporal effects
- Motion blur
- Depth of field
- Anti-aliasing

Cook:

- Area lights
- Glossy reflections
- Caustics
- Diffuse interreflections
- Diffraction
- Polarization
- Phosphorescence
- Temporal effects
- Motion blur
- Depth of field
- Anti-aliasing
Rendering Equation

God's Algorithm

1 room
1 bulb
100 watts
$10^{20}$ photons per second

Photon behavior:

- Travel in straight lines
- Get absorbed, or change direction:
  - Bounce (random / deterministic)
  - Get transmitted
- Leave into the void
- Get detected
Rendering Equation
God’s Algorithm - Mathematically

A photon may arrive at a sensor after travelling in a straight line from a light source to the sensor:

\[ L(s \leftarrow x) = L_E(s \leftarrow x) \]

Or, it may be reflected by a surface towards the sensor:

\[ L(s \leftarrow x) = \int_A f_r(s \leftarrow x \leftarrow x') L(x \leftarrow x') G(x \leftrightarrow x') \, dA(x') \]

Those are the options.

Adding direct and indirect illumination together:

\[ L(s \leftarrow x) = L_E(s \leftarrow x) + \int_A f_r(s \leftarrow x \leftarrow x') L(x \leftarrow x') G(x \leftrightarrow x') \, dA(x') \]
Rendering Equation

God’s Algorithm - Mathematically

\[ L(s \leftarrow x) = L_E(s \leftarrow x) + \int_A f_r(s \leftarrow x \leftarrow x') L(x \leftarrow x') G(x \leftarrow x') \, dA(x') \]

- **Emission**
- **Hemisphere**
- **Reflection**
- **Indirect**
- **Geometry factor**
Rendering Equation

\[ L(s \leftarrow x) = L_E(s \leftarrow x) + \int_A f_r(s \leftarrow x \leftarrow x') L(x \leftarrow x') G(x \leftarrow x') \, dA(x') \]

The Rendering Equation*:

- Light transport from lights to sensor
- Recursive
- Physically based

The equation allows us to determine to which extent rendering algorithms approximate real-world light transport.

*: The Rendering Equation, Kajiya, 1986
Today's Agenda:

- Introduction
- The Rendering Equation
- Light Transport
Today's Agenda:

- Introduction
- The Rendering Equation
- Light Transport

```c
if (depth < MAXDEPTH)
    t = random();
else
    t = inside ? 1.0 : 0.0;

if (t < 0.0) return 0.0;
if (t > 1.0) return 1.0;

R = D * n + (1.0 - D) * N;
E = diffuse;
if (rl + refr) &L (depth < MAXDEPTH)
    E = true;

// Radiance = SphereLight(A, B, C, S, N, M, t);
// x = x + radius.y + radius.z) > 0) &L (depth R

// Depth = evaluateDepth( L, N ) * Survive,
// at factor = diffuse = Inv
// at weight = Vis( directdf, bmatrix );
// at cosThetaDF = cos(thetaDF );
// E = ((weight * cosThetaDF / directdf) * (radiance
// radiance walk - done properly, closely following Surface

// diffuse

// Hit bdf = SampleDiffuse( diffuse, N, r1, r2, &b, &pdf );
// Survive;
// pdf;
// E = bdf * (dot(N, r) / pdf);
```
INFOMAGR – Advanced Graphics

Jacco Bikker - November 2022 - February 2023

END of “Light Transport”

next lecture: “Path Tracing”