Global Illumination

Today

- Direct vs. global illumination
- Energy balance and the rendering equation
- Path tracing
- Russian roulette
- Path guiding

Next

- Bidirectional light transport
- Volume scattering
0 Bounces (visible lights)
1 Bounce (Direct Illumination)
Indirect Illumination
Direct + Indirect
Direct vs. Global Illumination

Fast Separation of Direct and Global Components of a Scene Using High Frequency Illumination, Nayar et al. 2006
Energy Balance

Accountability

- [outgoing] - [incoming] = [emitted] - [absorbed]

Macro level:

- The total light energy put into the system must equal the energy leaving the system (usually, via heat)

\[
\Phi_o - \Phi_i = \Phi_e - \Phi_a
\]
Energy Balance

Accountability

- \([\text{outgoing}] - [\text{incoming}] = [\text{emitted}] - [\text{absorbed}]\)

Micro level:

- The energy flowing into a small region of phase space must equal the energy flowing out:

\[
E_o(p) - E_i(p) = E_e(p) - E_a(p)
\]
Surface Balance Equation

\[
\begin{align*}
\text{[outgoing]} &= \text{[emitted]} + \text{[incoming]} - \text{[absorbed]} \\
\text{[reflected]} &= \text{[incoming]} - \text{[absorbed]} \\
\text{[outgoing]} &= \text{[emitted]} + \text{[reflected]}
\end{align*}
\]

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

Need to know incident radiance.
Incident Radiance Function
The Reflection Equation

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

Need to know incident radiance.

**transport function:** \( tr(p, \omega) \)
Returns first intersection point on surface in the scene
The Rendering Equation

Radiance invariance along rays:

\[ L_i(p, \omega_i) = L_o(tr(p, \omega_i), -\omega_i) \]

Incident radiance in terms of exitant radiance at 1st visible surface:

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

Light scattering     Light transport
The Rendering Equation

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

Partition the rendering equation

■ Direct and indirect illumination

Apply Monte Carlo integration to each

■ One sample for each—no splitting!
■ Assumption: 100s of samples per pixel

Terminate paths with Russian Roulette
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.
Partitioning The Rendering Equation

**Incident direct illumination:** \( L_{i,d}(p, \omega_i) \)

- Sample lights+BRDFs, use MIS

**Incident indirect illumination:** \( L_{i,i}(p, \omega_i) \)

- Handle with recursive evaluation of the rendering equation

\[
L_o(p, \omega_o) = L_e(p, \omega_o) + \int_{H^2} f_r(\omega_i \rightarrow \omega_o) L_{i,d}(p, \omega_i) \cos \theta_i \, d\omega_i + \int_{H^2} f_r(\omega_i \rightarrow \omega_o) L_{i,i}(p, \omega_i) \cos \theta_i \, d\omega_i
\]
Partitioning The Rendering Equation

Incident direct illumination: \( L_{i,d}(p, \omega_i) \)

- Sample lights+BRDFs, use MIS

Incident indirect illumination: \( L_{i,i}(p, \omega_i) \)

- Handle with recursive evaluation of the rendering equation

\[
L_o(p, \omega_o) = L_e(p, \omega_o) + \\
\int_{H^2} f_r(\omega_i \rightarrow \omega_o) L_{i,d}(p, \omega_i) \cos \theta_i \, d\omega_i + \\
\int_{H^2} f_r(\omega_i \rightarrow \omega_o) L_o(tr(p, \omega_i), -\omega_i) \cos \theta_i \, d\omega_i
\]
Path Tracing: Indirect Illumination

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

Sample incoming direction from some distribution (e.g. proportional to BRDF): \( \omega_i \sim p(\omega) \)

Recursively call path tracing function to compute incident indirect radiance

Estimator:
\[
\frac{f_r(\omega_i \rightarrow \omega_o) L_{i,i}(p, \omega_i) \cos \theta_i}{p(\omega_i)}
\]
Path Tracing: Indirect Illumination

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

Sample incoming direction from some distribution (e.g. proportional to BRDF): \( \omega_i \sim p(\omega) \)

Recursively call path tracing function to compute incident indirect radiance

Estimator: \[
\frac{f_r(\omega_i \rightarrow \omega_o) L_o(tr(p, \omega_i), -\omega_i) \cos \theta_i}{p(\omega_i)}
\]
Path Tracing

\[ L_e \]

\( p_1 \)

\( p_2 \)
Path Tracing: Recursive

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

Spectrum PathLo(Ray ray) {
    Intersection isect = scene->Intersect(ray);
    BSDF brdf = isect.GetBSDF();
    Vector3f wo = -ray.d;

    Spectrum Ld = DirectLighting(bsdf, wo);

    Spectrum fr = bsdf.Sample_f(wo, &wi, &pdf);
    return isect.Le(wo) + Ld +
    fr * PathLo(Ray(isect.P, wi)) * Dot(wi, isect.N) / pdf;
}
Recursive Expansion

\[ L_o(p, \omega_o) = L_e(p, \omega_o) + \frac{f_r(\omega_o, \omega_i) \cos \theta_i}{p(\omega_i)} L_o(p', -\omega_i) \]

\[ = L_e + \frac{f_r \cos \theta_i}{p(\omega_i)} \left[ L_e(p', -\omega_i) + \frac{f_r(-\omega_i, \omega'_i) \cos \theta'_i}{p(\omega'_i)} L_o(p'', \omega''_i) \right] \]

\[ = \ldots \]
Path Contribution

\[ \beta(\bar{p}) = \prod_j \frac{f_r(p_j, \omega_{o,j}, \omega_{i,j}) \cos \theta_{i,j}}{p(\omega_{i,j})} \]

\[ L_i = \beta(\bar{p}) L_e \]
Path Tracing: Iterative

```cpp
Spectrum PathLo(Ray ray) {
    Spectrum Lo = 0, beta = 1;
    while (true) {
        Intersection isect = scene->Intersect(ray);
        BSDF brdf = isect.GetBSDF();
        Vector3f wo = -ray.d;

        Lo += beta * DirectLighting(bsdf, wo);

        Spectrum fr = bsdf.Sample_f(wo, &wi, &pdf);
        beta *= fr * Dot(wi, isect.N) / pdf;
    }
    return Lo;
}
```

Problem?
Russian Roulette

Avoid spending time on samples that make a small contribution

Consider a low-contribution sample of the form:

\[ L = \frac{f_r(\omega_i \rightarrow \omega_o) L_i(\omega_i) V(p, p') \cos \theta_i}{p(\omega_i)} \]
Russian Roulette

\[ L = \frac{f_r(\omega_i \rightarrow \omega_o) L_i(\omega_i) V(p, p') \cos \theta_i}{p(\omega_i)} \]

\[ L = \left[ \frac{f_r(\omega_i \rightarrow \omega_o) L_i(\omega_i) \cos \theta_i}{p(\omega_i)} \right] V(p, p') \]

If tentative contribution (in brackets) is small, total contribution to the image will be small regardless of \( V(p, p') \)
Russian Roulette

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

Same expected value as original estimator:

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

```cpp
Spectrum PathLo(Ray ray) {
    Spectrum Lo = 0, beta = 1;
    while (true) {
        Intersection isect = scene->Intersect(ray);
        BSDF brdf = isect.GetBSDF();
        Vector3f wo = -ray.d;

        Lo += beta * DirectLighting(bsdf, wo);

        Spectrum fr = bsdf.Sample_f(wo, &wi, &pdf);
        beta *= fr * Dot(wi, isect.N) / pdf;

        Float p_rr = 1 - beta.y();
        if (p_rr > UniformFloat()) break;
        beta /= p_rr;
    }
    return Lo;
}
```
No Russian Roulette: 3.9 seconds
Variance 0.00379, MC Efficiency 67.4
Terminate 50% with luminance < 0.25: 3.3 seconds
Variance 0.003808, MC Efficiency 78.59
Terminate 50% with luminance < 0.5: 3.2 seconds
Variance 0.003846, MC Efficiency 80.54
Terminate 90% with luminance < 1: 2.5 seconds
Variance 0.0124, MC Efficiency 32.90
Terminate proportional to path contrib: 2.9 seconds
Variance 0.00413, MC Efficiency 84.76
$p_{rr} = 0.5, \ 612s$
Variance 1.201, MC Efficiency 0.00136
\[ p_{rr} = \text{max RGB, } 3170s \]

Variance 0.0209, Efficiency 0.01512
Path Guiding

Recall the MC estimator:

\[
\frac{f_r(\omega_i \rightarrow \omega_o) \, L_o(tr(p, \omega_i), -\omega_i) \, \cos \theta_i}{p(\omega_i)}
\]

Regular path tracing: sample \( \omega_i \sim f_r(\omega_i \rightarrow \omega_o) \)
(or something similar to it)

But: really want to sample \( \omega_i \sim \propto f_r \, L_o \, \cos \theta \)

Idea: learn the distribution of light in the scene to guide sampling
Path Guiding

(a) Spatial binary tree

(b) Directional quadtree

[Müller et al. 2017]

Hierarchical Sample Warping

[Clarberg et al. 2005]
Path Guiding

<table>
<thead>
<tr>
<th>PT w/ NEE</th>
<th>Ours</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE: 7.949</td>
<td>0.694</td>
<td>—</td>
</tr>
<tr>
<td>Samples per pixel: 3100</td>
<td>1812</td>
<td>—</td>
</tr>
<tr>
<td>Minutes (training + rendering): 0 + 5.1</td>
<td>1.1 + 3.9</td>
<td>—</td>
</tr>
</tbody>
</table>

[Müller et al. 2017]