The Rendering Equation & Monte Carlo Ray Tracing

Last Time?
- Local Illumination
  - BRDF
  - Ideal Diffuse Reflectance
  - Ideal Specular Reflectance
  - The Phong Model
- Radiosity Equation/Matrix
- Calculating the Form Factors

From Last Time
- Computing Form Factors
- Advanced Radiosity
  - Progressive Radiosity
  - Adaptive Subdivision
  - Discontinuity Meshing
  - Hierarchical Radiosity

Form Factor from Ray Casting
- Cast $n$ rays between the two patches
  - Compute visibility (what fraction of rays do not hit an occluder)
  - Integrate the point-to-point form factor
- Permits the computation of the patch-to-patch form factor, as opposed to point-to-patch

Stages in a Radiosity Solution

<table>
<thead>
<tr>
<th>Stage</th>
<th>Percentage</th>
<th>Why So Costly?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Geometry</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Emittance &amp; Reflectance Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Camera Position &amp; Orientation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Form Factor Calculation</td>
<td>&gt; 90%</td>
<td>Calculation &amp; storage of $n^2$ form factors; $(n^3$ for naive visibility calculation)</td>
</tr>
<tr>
<td>Solve the Radiosity Matrix</td>
<td>&lt; 10%</td>
<td></td>
</tr>
<tr>
<td>Radiosity Solution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visualization (Rendering)</td>
<td>- 0%</td>
<td></td>
</tr>
<tr>
<td>Radiosity Image</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Progressive Refinement
- Goal: Provide frequent and timely updates to the user during computation
- Key Idea: Update the entire image at every iteration, rather than a single patch
- How? Instead of summing the light received by one patch, distribute the radiance of the patch with the most undistributed radiance.
Reordering the Solution for PR

**Shooting:** the radiosity of all patches is updated for each iteration:

\[
\begin{bmatrix}
    \mathbf{B}_1 & \mathbf{B}_2 & \cdots & \mathbf{F}_1 & \mathbf{B}_3 & \cdots \\
    \vdots & \vdots & \ddots & \vdots & \vdots & \ddots \\
    \mathbf{B}_n & \mathbf{B}_{n+1} & \cdots & \mathbf{F}_n & \mathbf{B}_{n+2} & \cdots \\
\end{bmatrix}
\]

This method is fundamentally a Southwell relaxation.

Progressive Refinement w/out Ambient Term

Progressive Refinement with Ambient Term

Increasing the Accuracy of the Solution

What’s wrong with this picture?

- Image quality is a function of patch size
- Compute a solution on a uniform initial mesh, then refine the mesh in areas that exceed some error tolerance:
  - shadow boundaries
  - other areas with a high radiosity gradient

Adaptive Subdivision of Patches

- Coarse patch solution (145 patches)
- Improved solution (1021 subpatches)
- Adaptive subdivision (1306 subpatches)

Discontinuity Meshing

- Limits of umbra and penumbra
  - Captures nice shadow boundaries
  - Complex geometric computation to construct mesh
Discontinuity Meshing

“Fast and Accurate Hierarchical Radiosity Using Global Visibility”
Durand, Drettakis, & Puech 1999

Hierarchical Radiosity

- Group elements when the light exchange is not important
  - Breaks the quadratic complexity
  - Control non trivial, memory cost

Practical Problems with Radiosity

- Meshing
  - memory
  - robustness
- Form factors
  - computation
- Diffuse limitation
  - extension to specular takes too much memory

Questions?

Lightscape http://www.lightscape.com

Today

- Does Ray Tracing Simulate Physics?
- The Rendering Equation
- Monte-Carlo Integration
- Sampling
- Monte-Carlo Ray Tracing vs. Path Tracing

Does Ray Tracing Simulate Physics?

- No…. traditional ray tracing is also called “backward” ray tracing
- In reality, photons actually travel from the light to the eye
Forward Ray Tracing

- Start from the light source
  - But very, very low probability to reach the eye
- What can we do about it?
  - Always send a ray to the eye…. still not efficient

Transparent Shadows?

- What to do if the shadow ray sent to the light source intersects a transparent object?
  - Pretend it’s opaque?
  - Multiply by transparency color?
    (ignores refraction & does not produce caustics)
- Unfortunately, ray tracing is full of dirty tricks

Is this Traditional Ray Tracing?

- No, Refraction and complex reflection for illumination are not handled properly in traditional (backward) ray tracing

Refraction and the Lifeguard Problem

- Running is faster than swimming

Today

- Does Ray Tracing Simulate Physics?
- The Rendering Equation
- Monte-Carlo Integration
- Sampling
- Monte-Carlo Ray Tracing vs. Path Tracing

The Rendering Equation

- Clean mathematical framework for light-transport simulation
- At each point, outgoing light in one direction is the integral of incoming light in all directions multiplied by reflectance property
Reading for Today:

- "The Rendering Equation", Kajiya, SIGGRAPH 1986

The Rendering Equation

\[ L(x', \omega') = L(x', \omega') + \int \rho(x', \omega') L(x, \omega) G(x, x') V(x, x') \, dA \]

\( L(x', \omega') \) is the radiance from a point on a surface in a given direction \( \omega' \)

The Rendering Equation

\[ L(x', \omega') = E(x', \omega') + \int \rho(x', \omega') L(x, \omega) G(x, x') V(x, x') \, dA \]

\( E(x', \omega') \) is the emitted radiance from a point: \( E \) is non-zero only if \( x' \) is emissive (a light source)

The Rendering Equation

\[ L(x', \omega') = E(x', \omega') + \int \rho(x', \omega') L(x, \omega) G(x, x') V(x, x') \, dA \]

For each \( x \), compute \( L(x, \omega) \), the radiance at point \( x \) in the direction \( \omega \) (from \( x \) to \( x' \))

The Rendering Equation

\[ L(x', \omega') = E(x', \omega') + \int \rho(x', \omega') L(x, \omega) G(x, x') V(x, x') \, dA \]

Sum the contribution from all of the other surfaces in the scene

The Rendering Equation

\[ L(x', \omega') = E(x', \omega') + \int \rho(x', \omega') L(x, \omega) G(x, x') V(x, x') \, dA \]

scale the contribution by \( \rho(x', \omega') \), the reflectivity (BRDF) of the surface at \( x' \)
The Rendering Equation

For each x, compute V(x,x'), the visibility between x and x':
1 when the surfaces are unobstructed along the direction $\omega$, 0 otherwise.

$L(x',\omega') = E(x',\omega') + \int \rho(x,\omega')L(x,\omega)G(x,x')V(x,x') dA$

Intuition about G(x,x')?

• Which arrangement of two surfaces will yield the greatest transfer of light energy? Why?

Rendering Equation $\Rightarrow$ Radiosity

$L(x',\omega') = E(x',\omega') + \int \rho(x,\omega')L(x,\omega)G(x,x')V(x,x') dA$

B$x' = E_{x'} + \int \rho_{x'}G(x,x')V(x,x') dA$

Discretize:
$B_i = E_i + \rho \sum F_i B_j$

Questions?

Today

• Does Ray Tracing Simulate Physics?
• The Rendering Equation
• Monte-Carlo Integration
  – Probabilities and Variance
  – Analysis of Monte-Carlo Integration
• Sampling
• Monte-Carlo Ray Tracing vs. Path Tracing

Questions?

1 glossy sample per pixel

256 glossy samples per pixel
Monte-Carlo Computation of $\pi$

- Take a random point $(x,y)$ in unit square
- Test if it is inside the $\frac{1}{4}$ disc
  - Is $x^2 + y^2 < 1$?
- Probability of being inside disc?
  - area of $\frac{1}{4}$ unit circle / area of unit square
    $= \frac{\pi}{4}$
- $\pi = 4 \times$ number inside disc / total number
- The error depends on the number of trials

Convergence & Error

- Let’s compute 0.5 by flipping a coin:
  - 1 flip: 0 or 1
    $\rightarrow$ average error $= 0.5$
  - 2 flips: 0, 0.5, 0.5 or 1
    $\rightarrow$ average error $= 0.25$
  - 4 flips: 0(*1), 0.25(*4), 0.5(*6), 0.75(*4), 1(*1)
    $\rightarrow$ average error $= 0.1875$
- Unfortunately, doubling the number of samples does not double accuracy

Review of (Discrete) Probability

- Random variable can take discrete values $x_i$
- Probability $p_i$ for each $x_i$
  $0 < p_i < 1$, $\sum p_i = 1$
- Expected value
  $E(x) = \sum_{i=1}^{n} p_i x_i$
- Expected value of function of random variable
  - $f(x_i)$ is also a random variable
    $E[f(x)] = \sum_{i=1}^{n} p_i f(x_i)$

Variance & Standard Deviation

- Variance $\sigma^2$: deviation from expected value
- Expected value of square difference
  $\sigma^2 = E[(x - E[x])^2] = \sum_{i=1}^{n} (x_i - E[x])^2 p_i$
- Also
  $\sigma^2 = E[x^2] - (E[x])^2$
- Standard deviation $\sigma$:
  square root of variance (notion of error, RMS)

Monte Carlo Integration

- Turn integral into finite sum
- Use $n$ random samples
- As $n$ increases…
  - Expected value remains the same
  - Variance decreases by $n$
  - Standard deviation (error) decreases by $\frac{1}{\sqrt{n}}$
- Thus, converges with $\frac{1}{\sqrt{n}}$

Advantages of MC Integration

- Few restrictions on the integrand
  - Doesn’t need to be continuous, smooth, ...
  - Only need to be able to evaluate at a point
- Extends to high-dimensional problems
  - Same convergence
- Conceptually straightforward
- Efficient for solving at just a few points
Disadvantages of MC Integration

- Noisy
- Slow convergence
- Good implementation is hard
  - Debugging code
  - Debugging math
  - Choosing appropriate techniques
- Punctual technique, no notion of smoothness of function (e.g., between neighboring pixels)

Questions?


Today

- Does Ray Tracing Simulate Physics?
- The Rendering Equation
- Monte-Carlo Integration
- Sampling
  - Stratified Sampling
  - Importance Sampling
- Monte-Carlo Ray Tracing vs. Path Tracing

Domains of Integration

- Pixel, lens (Euclidean 2D domain)
- Time (1D)
- Hemisphere
  - Work needed to ensure uniform probability

Example: Light Source

- We can integrate over surface or over angle
- But we must be careful to get probabilities and integration measure right!

Stratified Sampling

- With uniform sampling, we can get unlucky
  - E.g. all samples in a corner
- To prevent it, subdivide domain $\Omega$ into non-overlapping regions $\Omega_i$
  - Each region is called a stratum
- Take one random samples per $\Omega_i$
Stratified Sampling Example

<table>
<thead>
<tr>
<th></th>
<th>Unstratified</th>
<th>Stratified</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x) = e^{4\sin(2x^2)}$</td>
<td>$O(1/\sqrt{N})$</td>
<td>$O(1/N)$</td>
</tr>
<tr>
<td>$f(x) = e^{8\sin(3x^2)}$</td>
<td>$O(1/\sqrt{N})$</td>
<td>$O(1/N)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$N$</th>
<th>$I$</th>
<th>$N$</th>
<th>$I$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.75039</td>
<td>1</td>
<td>2.70457</td>
</tr>
<tr>
<td>10</td>
<td>1.9893</td>
<td>10</td>
<td>1.72858</td>
</tr>
<tr>
<td>100</td>
<td>1.79139</td>
<td>100</td>
<td>1.77025</td>
</tr>
<tr>
<td>1000</td>
<td>1.75146</td>
<td>1000</td>
<td>1.77606</td>
</tr>
<tr>
<td>10000</td>
<td>1.77313</td>
<td>10000</td>
<td>1.77610</td>
</tr>
</tbody>
</table>

Importance Sampling

$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}$$

- Choose $p$ wisely to reduce variance
  - Want to use a $p$ that resembles $f$
  - Does not change convergence rate (still sqrt)
  - But decreases the constant

Uniform vs. Importance Sampling

$$U(\omega_i)$$

$P(\omega_i)$

5 Samples/Pixel

Uniform vs. Importance Sampling

$U(\omega_i)$

$P(\omega_i)$

25 Samples/Pixel

Uniform vs. Importance Sampling

$U(\omega_i)$

$P(\omega_i)$

75 Samples/Pixel
Questions?

Naïve sampling strategy

Optimal sampling strategy

Veach & Guibas "Optimally Combining Sampling Techniques for Monte Carlo Rendering" SIGGRAPH 95

Today

• Does Ray Tracing Simulate Physics?
• The Rendering Equation
• Monte-Carlo Integration
• Sampling
• Monte-Carlo Ray Tracing & Path Tracing

Ray Casting

• Cast a ray from the eye through each pixel

Ray Tracing

• Cast a ray from the eye through each pixel
• Trace secondary rays (light, reflection, refraction)

Monte-Carlo Ray Tracing

• Cast a ray from the eye through each pixel
• Cast random rays to accumulate radiance contribution
  – Recurse to solve the Rendering Equation

Importance of Sampling the Light

Without explicit light sampling

With explicit light sampling

1 path per pixel

4 path per pixel

Should also systematically sample the primary light
Monte Carlo Path Tracing
- Trace only one secondary ray per recursion
- But send many primary rays per pixel (performs antialiasing as well)

Ray Tracing vs Path Tracing
2 bounces
5 glossy samples
5 shadow samples
How many rays cast per pixel?
1 main ray + 5 shadow rays +
5 glossy rays + 5x5 shadow rays +
5*5 glossy rays + 5x5x5 shadow rays
= 186 rays

How many 3 bounce paths can we trace per pixel for the same cost?
186 rays / 8 ray casts per path
= ~23 paths

Which will probably have less error?

Questions?

10 paths/pixel
100 paths/pixel

Images from Henrik Wann Jensen

Readings for Friday (3/22) pick one:

Raytracing & Epsilon

intersects sphere
@ t = -0.01
intersects sphere
@ t = 0.01
intersects sphere
@ t = 14.3
intersects light
@ t = 25.2
intersects light
@ t = 26.9

Solution: advance the ray start position epsilon distance along the ray direction OR ignore all intersections < epsilon (rather than < 0)

What's a good value for epsilon? Depends on hardware precision & scene dimensions

Images from Zachary Lynn