Ray Tracing I: Basics

Today

- Basic algorithms
- Overview of pbrt
- Ray-surface intersection

Next lecture

- Techniques to accelerate ray tracing of large numbers of geometric primitives

Light Rays

Three ideas about light rays

1. Light travels in straight lines (mostly)
2. Light rays do not interfere with each other if they cross (light is invisible!)
3. Light rays travel from the light sources to the eye (but the physics is invariant under path reversal - reciprocity).
Ray Tracing in Computer Graphics

Appel 1968 - Ray casting
1. Generate an image by sending one ray per pixel
2. Check for shadows by sending a ray to the light

Ray Tracing in Computer Graphics

“An improved Illumination model for shaded display”
T. Whitted,
CACM 1980

Time:
VAX 11/780 (1979)
74 min.
PC (2006)
6 sec.

Spheres and Checkerboard, T. Whitted, 1979
Ray Tracing Video

Spheres-over-plane.pbrt (mirror depth=1)
Spheres-over-plane.pb (mirror depth=10)

Spheres-over-plane.pb (glass depth=1)
Spheres-over-plane.pbrt (g/m depth=10)

Table 1.1: Main Interface Types. Most of pbrt is implemented in terms of 13 key abstract base classes, listed here. Implementations of each of these can easily be added to the system to extend its functionality.

<table>
<thead>
<tr>
<th>Base class</th>
<th>Directory</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape</td>
<td>shapes/</td>
<td>3.1</td>
</tr>
<tr>
<td>Aggregate</td>
<td>accelerators/</td>
<td>4.2</td>
</tr>
<tr>
<td>Camera</td>
<td>cameras/</td>
<td>6.1</td>
</tr>
<tr>
<td>Sampler</td>
<td>samplers/</td>
<td>7.2</td>
</tr>
<tr>
<td>Filter</td>
<td>filters/</td>
<td>7.7</td>
</tr>
<tr>
<td>Film</td>
<td>film/</td>
<td>7.8</td>
</tr>
<tr>
<td>Material</td>
<td>materials/</td>
<td>9.2</td>
</tr>
<tr>
<td>Texture</td>
<td>textures/</td>
<td>10.3</td>
</tr>
<tr>
<td>VolumeRegion</td>
<td>volumes/</td>
<td>11.3</td>
</tr>
<tr>
<td>Light</td>
<td>lights/</td>
<td>12.1</td>
</tr>
<tr>
<td>Renderer</td>
<td>renderers/</td>
<td>13.3</td>
</tr>
<tr>
<td>SurfaceIntegrator</td>
<td>integrators/</td>
<td>Ch. 15 intro</td>
</tr>
<tr>
<td>VolumeIntegrator</td>
<td>integrators/</td>
<td>16.2</td>
</tr>
</tbody>
</table>
PBRT Architecture

Figure 1.17: Class Relationships for the Main Rendering Loop in the SamplerRenderTask::Render() Method in renderers/sample.cpp. The Sampler provides a sequence of sample values, one for each image sample to be taken. The Camera turns a sample into a corresponding ray from the film plane, and the Integrators compute the radiance along that ray arriving at the film. The sample and its radiance are given to the Film, which stores their contribution in an image. This process repeats until the Sampler has provided as many samples as are necessary to generate the final image.
Ray-Plane Intersection

Ray: \( \vec{P} = \vec{O} + t \vec{D} \)

\[ 0 \leq t < \infty \]

Plane: \( (\vec{P} - \vec{P}') \cdot \vec{N} = 0 \)
\[ ax + by + cz + d = 0 \]

Solve for intersection

Substitute ray equation into plane equation

\[ (\vec{P} - \vec{P}') \cdot \vec{N} = (\vec{O} + t \vec{D} - \vec{P}') \cdot \vec{N} = 0 \]

\[ t = -\frac{(\vec{O} - \vec{P}') \cdot \vec{N}}{\vec{D} \cdot \vec{N}} \]
Optimize Ray-Slab?

Ray-Slab

$\ell_1 \quad \ell_2$

Optimize Ray-Box?

Note: Procedural geometry

Ray-Box
Optimize Ray-Convex Polyhedra?

Polyhedra defined as the intersection of N half-planes

Ray-Triangle Intersection 1: Insidedness

Barycentric coordinates

\[ \mathbf{P} = s_1 \mathbf{P}_1 + s_2 \mathbf{P}_2 + s_3 \mathbf{P}_3 \]

Inside triangle criteria

\[ 0 \leq s_1 \leq 1 \]
\[ 0 \leq s_2 \leq 1 \]
\[ 0 \leq s_3 \leq 1 \]
\[ s_1 + s_2 + s_3 = 1 \]
Ray-Triangle Intersection 2

3 points define a plane \( \vec{P} = s_1 \vec{P}_1 + s_2 \vec{P}_2 + s_3 \vec{P}_3 \)

Find ray-plane intersection point

Test whether that point is inside the triangle

\[
\begin{bmatrix}
\vec{P}_1 & \vec{P}_2 & \vec{P}_3
\end{bmatrix}
\begin{bmatrix}
s_1 \\
s_2 \\
s_3
\end{bmatrix}
= [\vec{P}]
\]

\[
\begin{bmatrix}
s_1 \\
s_2 \\
s_3
\end{bmatrix}
= \left[ \begin{bmatrix}
\vec{P}_1 & \vec{P}_2 & \vec{P}_3
\end{bmatrix} \right]^{-1} [\vec{P}]
\]

Moller-Trumbore Algorithm

\[
\vec{O} + t \vec{D} = (1 - b_1 - b_2) \vec{P}_0 + b_1 \vec{P}_1 + b_2 \vec{P}_2
\]

\[
\begin{bmatrix}
t \\
b_1 \\
b_2
\end{bmatrix}
= \frac{1}{\vec{S}_1 \cdot \vec{E}_1}
\begin{bmatrix}
\vec{S}_2 \cdot \vec{E}_2 \\
\vec{S}_1 \cdot \vec{S} \\
\vec{S}_2 \cdot \vec{D}
\end{bmatrix}
\]

Where:

\[
\vec{E}_1 = \vec{P}_1 - \vec{P}_0 \\
\vec{E}_2 = \vec{P}_2 - \vec{P}_0 \\
\vec{S} = \vec{O} - \vec{P}_0 \\
\vec{S}_1 = \vec{D} \times \vec{E}_2 \\
\vec{S}_2 = \vec{S} \times \vec{E}_1
\]

Cost = (1 div, 27 mul, 17 add)
Ray-Sphere Intersection

Ray: \[ \vec{P} = \vec{O} + t\vec{D} \]

Sphere: \[ (\vec{P} - \vec{C})^2 - R^2 = 0 \]
\[ (\vec{O} + t\vec{D} - \vec{C})^2 - R^2 = 0 \]
\[ t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]
\[ a = \vec{D}^2 \]
\[ b = 2(\vec{O} - \vec{C}) \cdot \vec{D} \]
\[ c = (\vec{O} - \vec{C})^2 - R^2 \]

Geometric Methods: Normals

e.g. Sphere
\[ \vec{N} = \vec{P} - \vec{C} \]
\[ \vec{P} = (x, y, z) \]
\[ x = \sin \theta \cos \phi \]
\[ y = \sin \theta \sin \phi \]
\[ z = \cos \theta \]
\[ \frac{\partial \vec{P}}{\partial \theta} = (\cos \theta \cos \phi, \cos \theta \sin \phi, -\sin \theta) \]
\[ \frac{\partial \vec{P}}{\partial \phi} = (-\sin \theta \sin \phi, \sin \theta \cos \phi, 0) \]
\[ \vec{N} = \frac{\partial \vec{P}}{\partial \theta} \times \frac{\partial \vec{P}}{\partial \phi} \]
Geometric Methods: Parameters

e.g. Sphere

\[ x = \sin \theta \cos \phi \]
\[ y = \sin \theta \sin \phi \]
\[ z = \cos \theta \]
\[ \theta = \theta_{\text{max}} + \nu (\theta_{\text{max}} - \theta_{\text{min}}) \]
\[ \phi = u \phi_{\text{max}} \]
\[ \phi = \tan^{-1}(x, y) \]
\[ \theta = \cos^{-1} z \]
\[ \nu = (\theta - \theta_{\text{min}}) / (\theta_{\text{max}} - \theta_{\text{min}}) \]
\[ u = \phi / \phi_{\text{max}} \]

Quadrics
Ray-Implicit Surface Intersection

\[ f(x, y, z) = 0 \]

\[
\begin{align*}
  x &= x_0 + x_1 t \\
  y &= y_0 + y_1 t \\
  z &= z_0 + z_1 t
\end{align*}
\]

\[ f^*(t) = 0 \]

1. Substitute ray equation
2. Find positive, real roots

Univariate root finding
- Newton’s method
- Regula-falsi
- Interval methods
- Heuristics

Ray-Algebraic Surface Intersection

\[ p_n(x, y, z) = 0 \]

\[
\begin{align*}
  x &= x_0 + x_1 t \\
  y &= y_0 + y_1 t \\
  z &= z_0 + z_1 t
\end{align*}
\]

\[ p_n^*(t) = 0 \]

Degree \( n \)
- Linear: Plane
- Quadric: Spheres, …
- Quartic: Tori

Polynomial root finding
- Quadratic, cubic, quartic
- Bezier/Bernoulli basis
- Descartes’ rule of signs
- Sturm sequences
History

<table>
<thead>
<tr>
<th>Type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polygons</td>
<td>Appel ‘68</td>
</tr>
<tr>
<td>Quadrics, CSG</td>
<td>Goldstein &amp; Nagel ‘71</td>
</tr>
<tr>
<td>Tori</td>
<td>Roth ‘82</td>
</tr>
<tr>
<td>Bicubic patches</td>
<td>Whitted ‘80, Kajiya ‘82</td>
</tr>
<tr>
<td>Superquadrics</td>
<td>Edwards &amp; Barr ‘83</td>
</tr>
<tr>
<td>Algebraic surfaces</td>
<td>Hanrahan ‘82</td>
</tr>
<tr>
<td>Swept surfaces</td>
<td>Kajiya ‘83, van Wijk ‘84</td>
</tr>
<tr>
<td>Fractals</td>
<td>Kajiya ‘83</td>
</tr>
<tr>
<td>Height fields</td>
<td>Coquillart &amp; Gangnet ‘84, Musgrave ‘88</td>
</tr>
<tr>
<td>Deformations</td>
<td>Barr ‘86</td>
</tr>
<tr>
<td>Subdivision surfs.</td>
<td>Kobbelt, Daubert, Siedel, ‘98</td>
</tr>
</tbody>
</table>

P. Hanrahan, A survey of ray-surface intersection algorithms