Ray Tracing

Ray Tracing 1
- Basic algorithm
- Overview of pbrt
- Ray-surface intersection (triangles, ...)

Ray Tracing 2
- Brute force: \(|I| \times |O|\)
- Acceleration data structures

Ray Tracing Acceleration Techniques

Approaches

- Faster Intersection
  - Uniform grids
  - Spatial hierarchies
  - k-d, oct-tree, bsp
  - Hierarchical grids
  - Hierarchical bounding volumes (HBV)

- Fewer Rays
  - Tighter bounds
  - Faster intersector
  - Early ray termination
  - Adaptive sampling

- Generalized Rays
  - Beam tracing
  - Cone tracing
  - Pencil tracing
Primitives

pbrt primitive base class
  ■ Shape
  ■ Material and emission (area light)
Primitives
  ■ Basic geometric primitive
  ■ Primitive instance
    ■ Transformation and pointer to basic primitive
  ■ Aggregate (collection)
    ■ Treat collections just like basic primitives
    ■ Incorporate acceleration structures into collections
    ■ May nest accelerators of different types
    ■ Types: grid.cpp and kdtree.cpp

Uniform Grids

Preprocess scene
1. Find bounding box
Uniform Grids

Preprocess scene
1. Find bounding box
2. Determine resolution

\[ n_x = n_y = n_z \gg n_o \]
\[ \max(n_x, n_y, n_z) = d \frac{1}{\sqrt[3]{n_o}} \]

Preprocess scene
1. Find bounding box
2. Determine resolution

\[ \max(n_x, n_y, n_z) = d \frac{1}{\sqrt[3]{n_o}} \]
2. Place object in cell, if object overlaps cell
Uniform Grids

Preprocess scene
1. Find bounding box
2. Determine resolution
   \[ \max(n_x, n_y, n_z) = \sqrt[d]{n_o} \]
3. Place object in cell, if object overlaps cell
4. Check that object intersects cell

Uniform Grids

Preprocess scene
Traverse grid
3D line – 3D-DDA
6-connected line

Section 4.3
Caveat: Overlap

*Optimize for objects that overlap multiple cells*

 Traverse until $t_{min}(cell) > t_{max}(ray)$

**Problem:** Redundant intersection tests:

**Solution:** Mailboxes

- Assign each ray an increasing number
- Primitive intersection cache (mailbox)
  - Store last ray number tested in mailbox
  - Only intersect if ray number is greater

Spatial Hierarchies

*Letters correspond to planes (A)*

*Point Location by recursive search*
Spatial Hierarchies

Letters correspond to planes (A, B)
Point Location by recursive search

Spatial Hierarchies

Letters correspond to planes (A, B, C, D)
Point Location by recursive search
Variations

kd-tree  oct-tree  bsp-tree

Ray Traversal Algorithms

Recursive inorder traversal
[Kaplan, Arvo, Jansen]

Intersect (L, tmin, tmax)  Intersect (L, tmin, t*)  Intersect (R, tmin, tmax)
Intersect (R, t*, tmax)
Build Hierarchy Top-Down

Choose splitting plane
- Midpoint
- Median cut
- Surface area heuristic

Surface Area and Rays

Number of rays in a given direction that hit an object is proportional to its projected area

The total number of rays hitting an object is $4\pi \bar{A}$

Crofton’s Theorem:
For a convex body $\bar{A} = \frac{S}{4}$

For example: sphere $S = 4\pi r^2$ $\bar{A} = A = \pi r^2$
**Surface Area and Rays**

The probability of a ray hitting a convex shape that is completely inside a convex cell equals

\[
\text{Pr}[r \cap S_o \mid r \cap S_c] = \frac{S_o}{S_c}
\]

**Surface Area Heuristic**

Intersection time \( t_i \)

Traversal time \( t_t \)

\( t_i = 80t_t \)

\[
C = t_t + p_a N_a t_i + p_b N_b t_i
\]
Surface Area Heuristic

\[ p_a = \frac{S_a}{S} \quad p_b = \frac{S_b}{S} \]

Comparison

<table>
<thead>
<tr>
<th>Time</th>
<th>Spheres</th>
<th>Rings</th>
<th>Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform Grid</td>
<td>d=1</td>
<td>244</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td>d=20</td>
<td>38</td>
<td>83</td>
</tr>
<tr>
<td>Hierarchical Grid</td>
<td></td>
<td>34</td>
<td>116</td>
</tr>
</tbody>
</table>

V. Havran, Best Efficiency Scheme Project
http://sgi.felk.cvut.cz/BES/
Comparison

Univ. Saarland RTRT Engine

Ray-casts per second = FPS @ 1K × 1K

<table>
<thead>
<tr>
<th>RT&amp;Shading Scene</th>
<th>SSE no shd.</th>
<th>SSE simple shd.</th>
<th>No SSE simple shd.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERW6 (static)</td>
<td>7.1</td>
<td>2.3</td>
<td>1.37</td>
</tr>
<tr>
<td>ERW6 (dynamic)</td>
<td>4.8</td>
<td>1.97</td>
<td>1.06</td>
</tr>
<tr>
<td>Conf (static)</td>
<td>4.55</td>
<td>1.93</td>
<td>1.2</td>
</tr>
<tr>
<td>Conf (dynamic)</td>
<td>2.94</td>
<td>1.6</td>
<td>0.82</td>
</tr>
<tr>
<td>Soda Hall</td>
<td>4.12</td>
<td>1.8</td>
<td>1.055</td>
</tr>
</tbody>
</table>

Pentium-IV 2.5GHz laptop
Kd-tree with surface-area heuristic [Havran]

Wald et al. 2003 [http://www.mpi-sb.mpg.de/~wald/]
Interactive Ray Tracing

Highly optimized software ray tracers
- Use vector instructions; Cache optimized
- Clusters and shared memory MPs

Ray tracing hardware
- AR250/350 ray tracing processor
  www.art-render.com
- SaarCOR

Ray tracing on programmable GPUs

Theoretical Nugget 1

Computational geometry of ray shooting

1. Triangles (Pellegrini)
   - Time: $O(\log n)$
   - Space: $O(n^{5+\varepsilon})$

2. Sphere (Guibas and Pellegrini)
   - Time: $O(\log^2 n)$
   - Space: $O(n^{5+\varepsilon})$
Theoretical Nugget 2

Optical computer = Turing machine
Reif, Tygar, Yoshida

Determining if a ray starting at y0 arrives at yn is undecidable

\[ y = y + 1 \]
\[ y = -2 \cdot y \]
\[ \text{if( } y > 0 \text{ )} \]