Ray Tracing

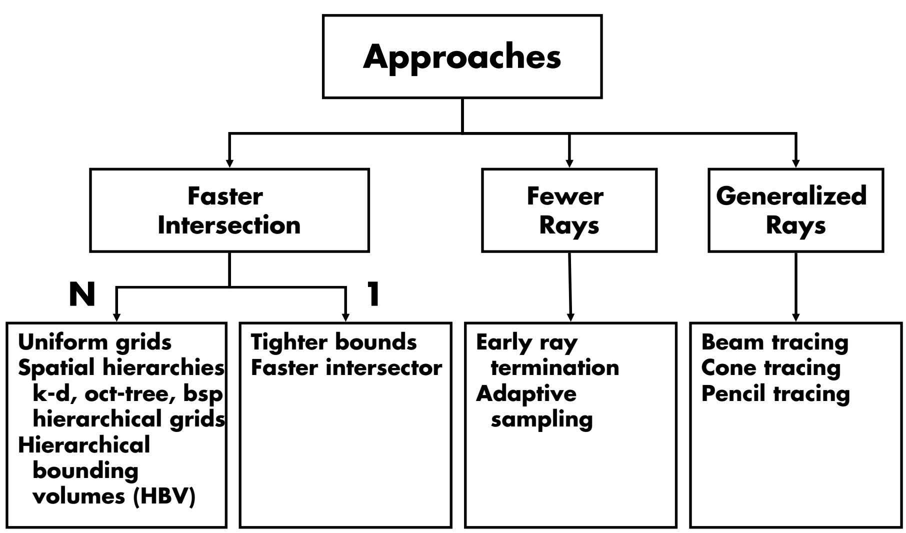
Ray Tracing 1

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

Ray Tracing 2

- Brute force: $|I| \times |O|$
- Acceleration data structures

Ray Tracing Acceleration Techniques



Primitives

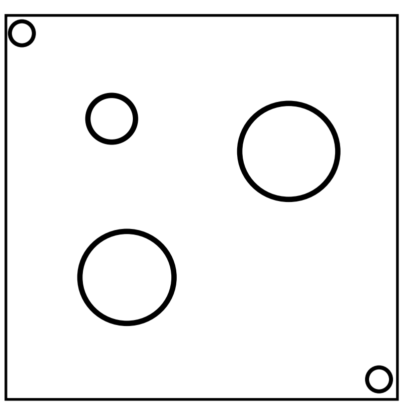
pbrt primitive base class

- 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

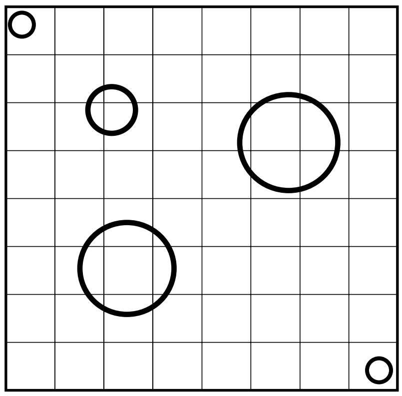
Uniform Grids



Preprocess scene

1. Find bounding box

Uniform Grids



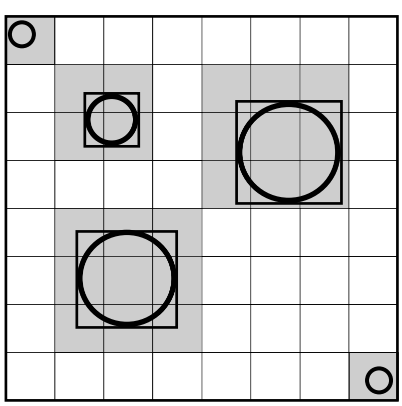
Preprocess scene

1. **Find bounding box**
2. **Determine resolution**

$$n_v = n_x n_y n_z \propto n_o$$

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

Uniform Grids



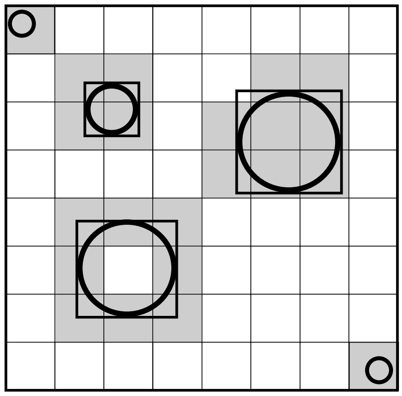
Preprocess scene

1. **Find bounding box**
2. **Determine resolution**

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

2. **Place object in cell, if object overlaps cell**

Uniform Grids



Preprocess scene

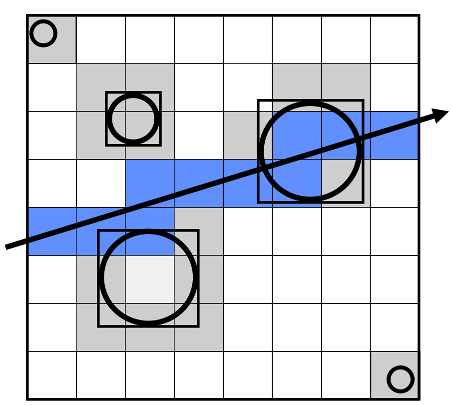
1. **Find bounding box**
2. **Determine resolution**

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

3. **Place object in cell, if object overlaps cell**
4. **Check that object intersects cell**

CS348B Lecture 3 Pat Hanrahan, Spring 2005

Uniform Grids



Preprocess scene

Traverse grid

3D line – 3D-DDA

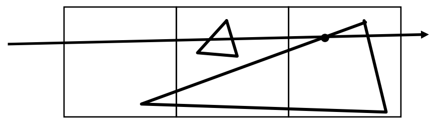
6-connected line

Section 4.3

CS348B Lecture 3 Pat Hanrahan, Spring 2005

Caveat: Overlap

Optimize for objects that overlap multiple cells



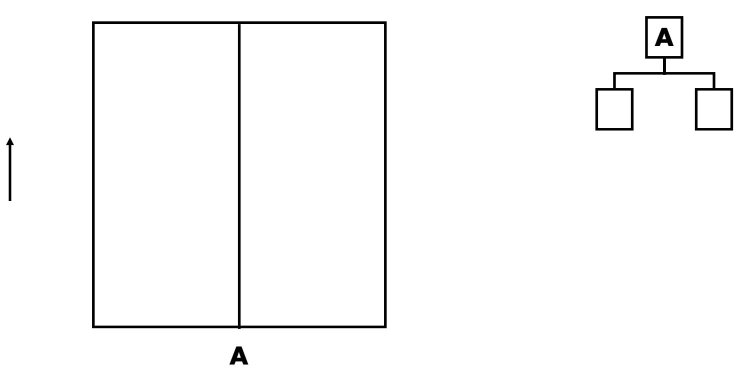
Traverse until tmin(cell) > tmax(ray)

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**

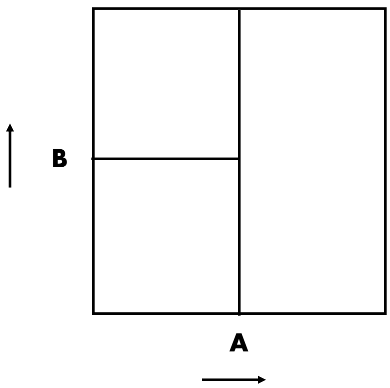
Spatial Hierarchies

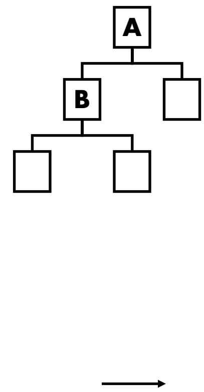


Letters correspond to planes (A)

Point Location by recursive search

Spatial Hierarchies

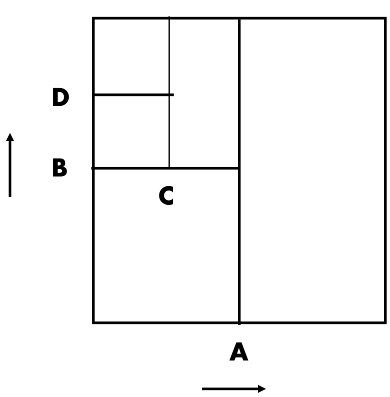


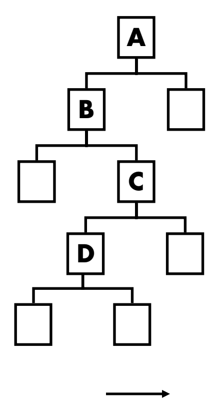


Letters correspond to planes (A, B)

Point Location by recursive search

Spatial Hierarchies

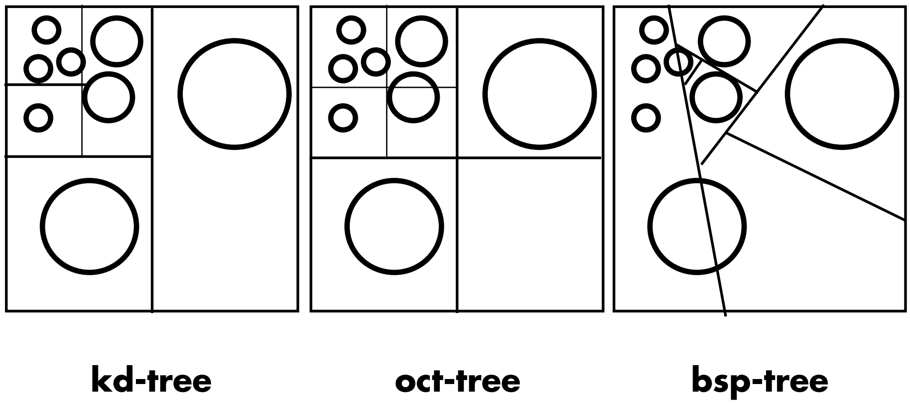




Letters correspond to planes (A, B, C, D)

Point Location by recursive search

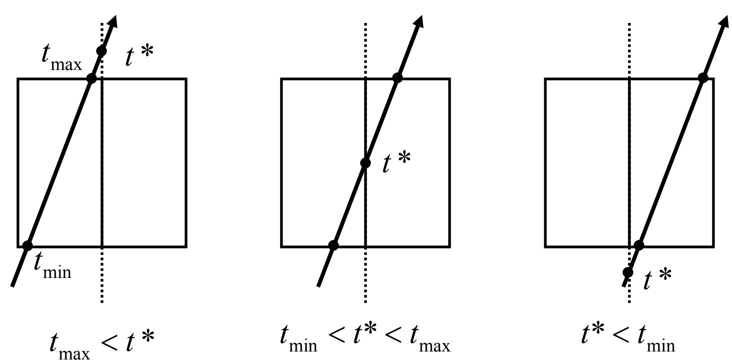
Variations



Ray Traversal Algorithms

Recursive inorder traversal

[Kaplan, Arvo, Jansen]

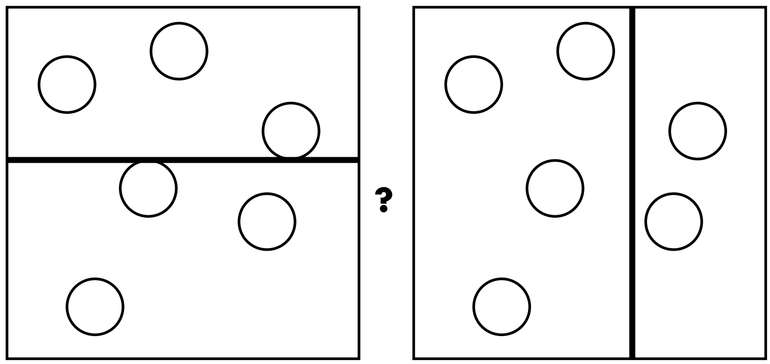



```
Intersect(L,tmin,tmax)
```

```
Intersect(L,tmin,t*)
Intersect(R,t*,tmax)
```

```
Intersect(R,tmin,tmax)
```

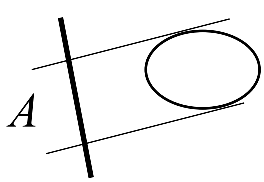
Build Hierarchy Top-Down



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 $4\pi \overline{A}$

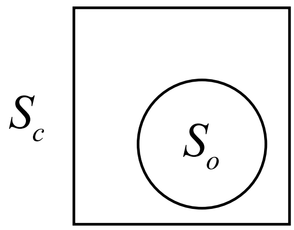
Crofton's Theorem:

For a convex body $\overline{A} = \frac{S}{4}$

For example: sphere $S = 4\pi r^2 \quad \overline{A} = A = \pi r^2$

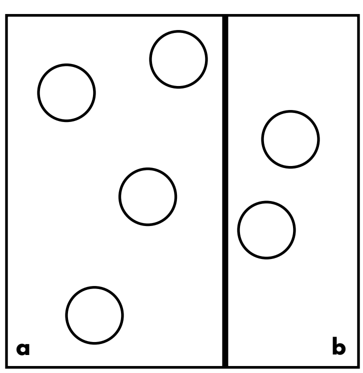
Surface Area and Rays

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



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

Surface Area Heuristic



Intersection time

t_i

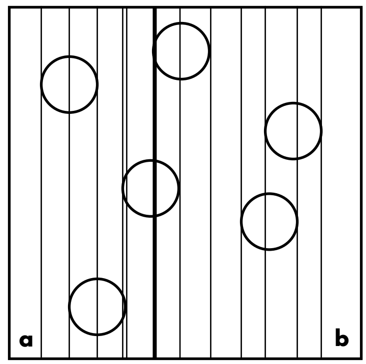
Traversal time

t_t

$t_i = 80 t_t$

$$C = t_t + p_a N_a t_i + p_b N_b t_i$$

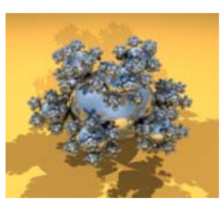
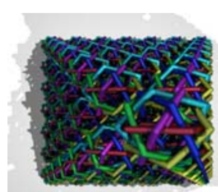
Surface Area Heuristic



2n splits

$$p_a = \frac{S_a}{S} \qquad p_b = \frac{S_b}{S}$$

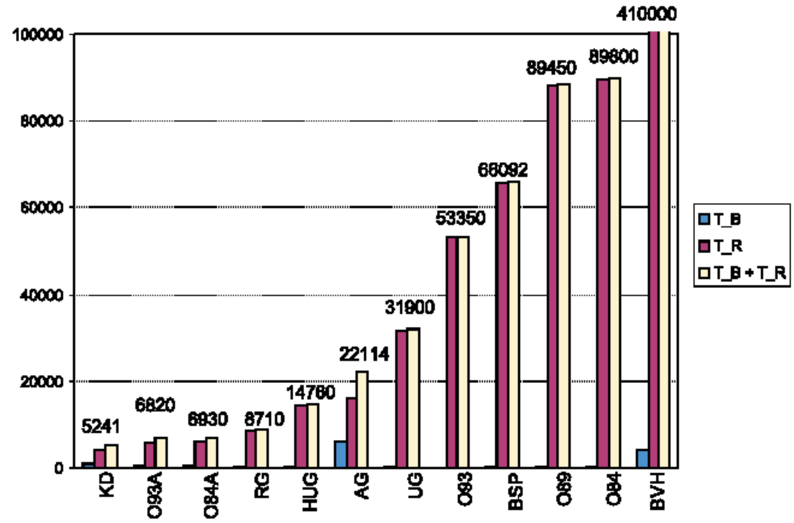
Comparison


Time		Spheres	Rings	Tree
Uniform Grid	d=1	244	129	1517
	d=20	38	83	781
Hierarchical Grid		34	116	34

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

Comparison



Univ. Saarland RTRT Engine

Ray-casts per second = FPS @ 1K × 1K

RT&Shading Scene	SSE no shd.	SSE simple shd.	No SSE simple shd.
ERW6 (static)	7.1	2.3	1.37
ERW6 (dynamic)	4.8	1.97	1.06
Conf (static)	4.55	1.93	1.2
Conf (dynamic)	2.94	1.6	0.82
Soda Hall	4.12	1.8	1.055

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**

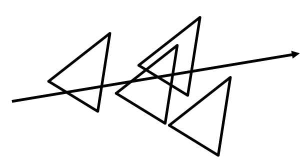
Ray tracing on programmable GPUs

Theoretical Nugget 1

Computational geometry of ray shooting

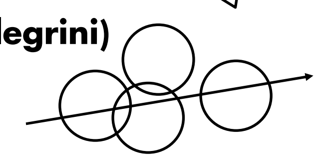
1. Triangles (Pellegrini)

- **Time:** $O(\log n)$
- **Space:** $O(n^{5+\varepsilon})$



2. Sphere (Guibas and Pellegrini)

- **Time:** $O(\log^2 n)$
- **Space:** $O(n^{5+\varepsilon})$

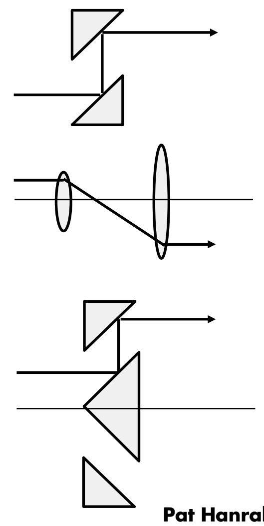


Theoretical Nugget 2

Optical computer = Turing machine

Reif, Tygar, Yoshida

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



y = y+1

y = -2*y

if(y>0)