Ray Tracing

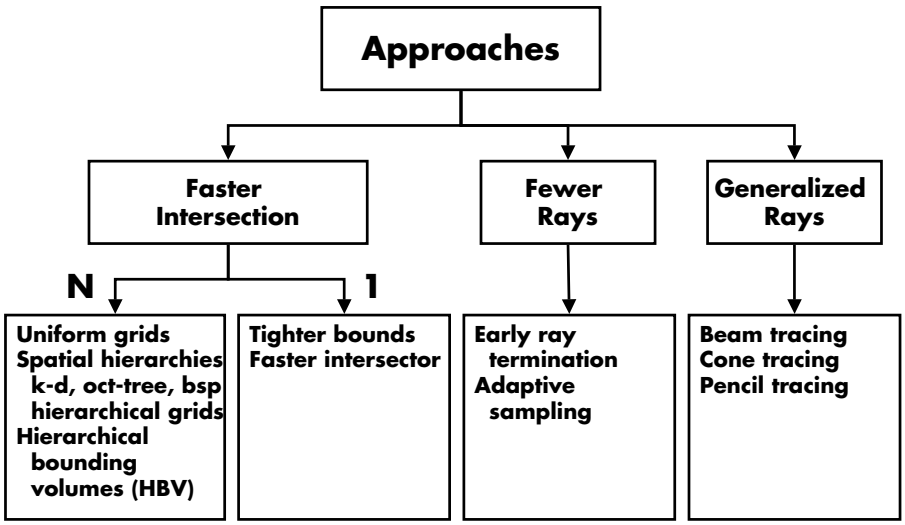
Ray Tracing 1

- **Basic algorithm**
- **Ray-surface intersection (triangles, ...)**

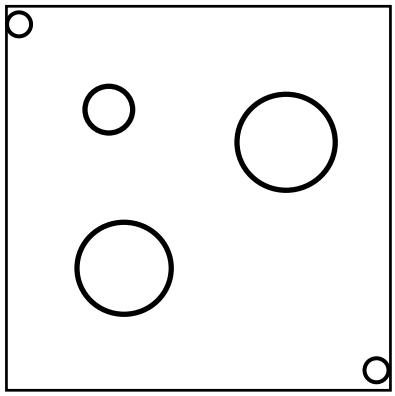
Ray Tracing 2

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

Ray Tracing Acceleration Techniques



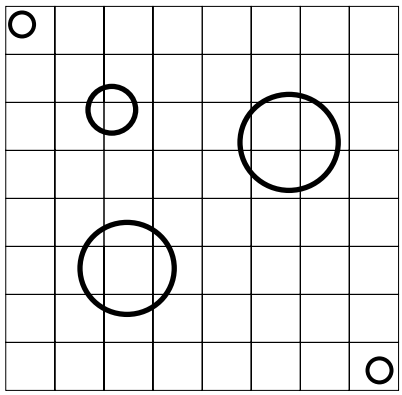
Uniform Grids



Preprocess scene

1. Find bounding box

Uniform Grids



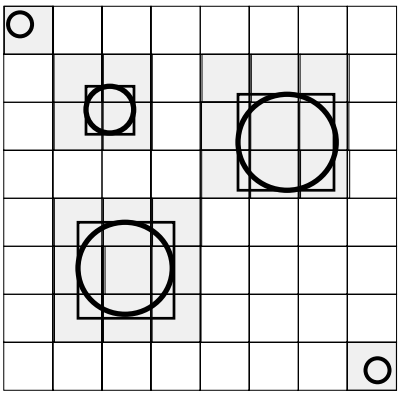
Preprocess scene

1. Find bounding box

2. Determine resolution

$$n^3 = d\,|O|$$

Uniform Grids



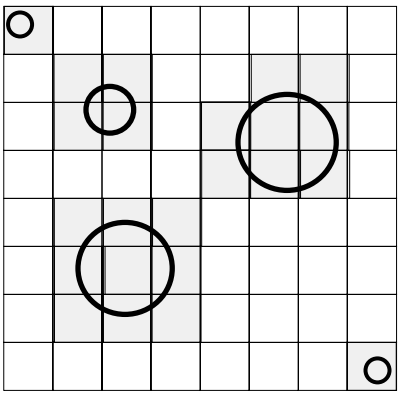
Preprocess scene

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

$$n^3 = d\,|O|$$

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

Uniform Grids



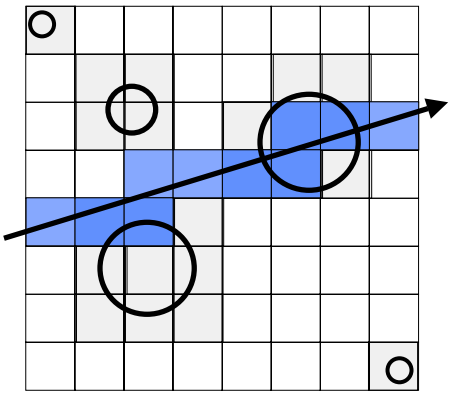
Preprocess scene

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

$$n^3 = d\,|O|$$

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**

`lrt` **algorithm**

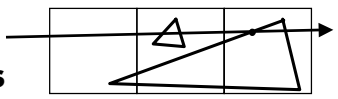
Caveat: Overlap

Optimize for objects that overlap multiple cells

Caveat 1:

Intersection must be within bound

Caveat 2:

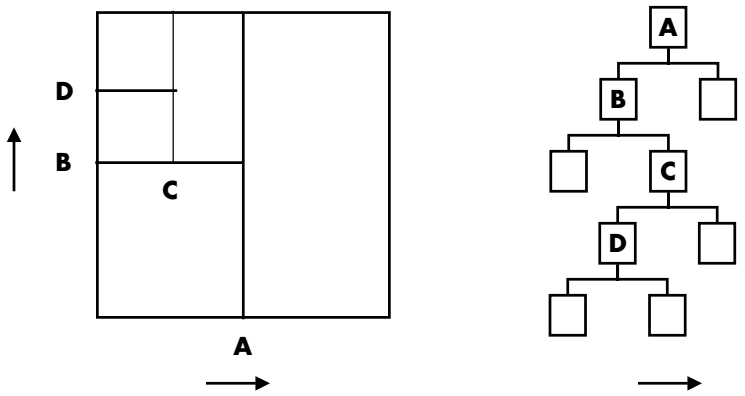


Redundant intersection tests

Mailboxes

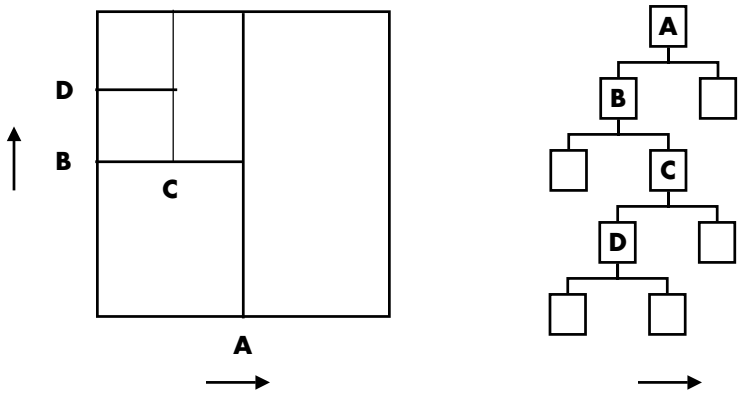
- **Assign each ray a number**
- **Object intersection cache (mailbox)**
  - **Store ray number**
  - **Intersection**

Spatial Hierarchies



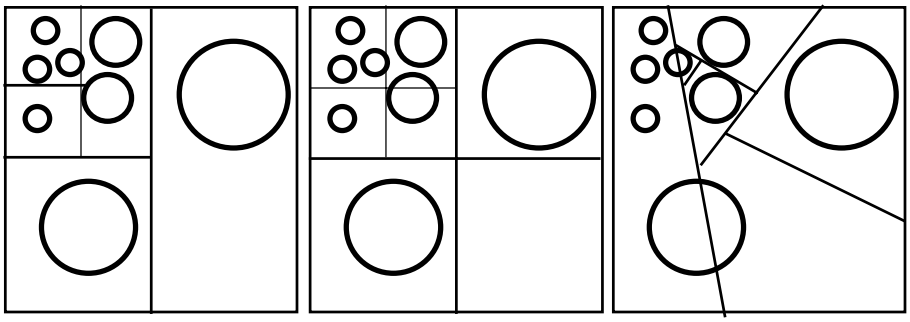
Leaf nodes corresponds unique regions in space

Spatial Hierarchies



Point Location by recursive search

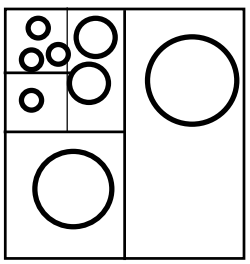
Variations



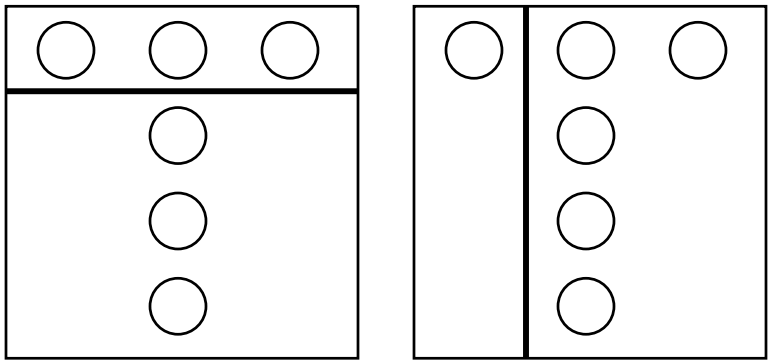
kd-tree oct-tree bsp-tree

Creating Spatial Hierarchies

```
insert(node,prim) {
  if( overlap(node->bound,prim) )
     if( leaf(node) ) {
        if( node->nprims > MAXPRIMS
         && node->depth  < MAXDEPTH ) {
           subdivide(node);
           foreach child in node
             insert(child,prim)
        }
        else
          insertlist(node->prims,prim);
     }
     else
       foreach child in node
         insert(child,prim)
}
// Typically MAXDEPTH=16, MAXPRIMS=2-8
```



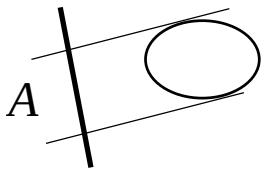
Median Cut



Build hierarchy bottom-up
Choose direction and position carefully
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 $4\pi\overline{A}$

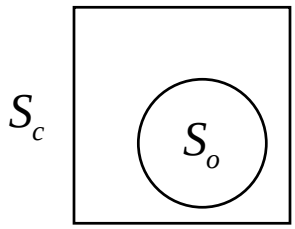
Crofton's Theorem:

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 an object that is completely inside a cell equals

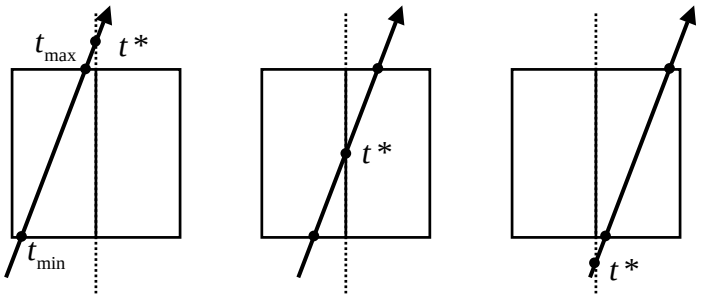


$$\Pr[r \cap O] = \frac{S_o}{S_c}$$

Ray Traversal Algorithms

Recursive inorder traversal

Kaplan, Arvo, Jansen

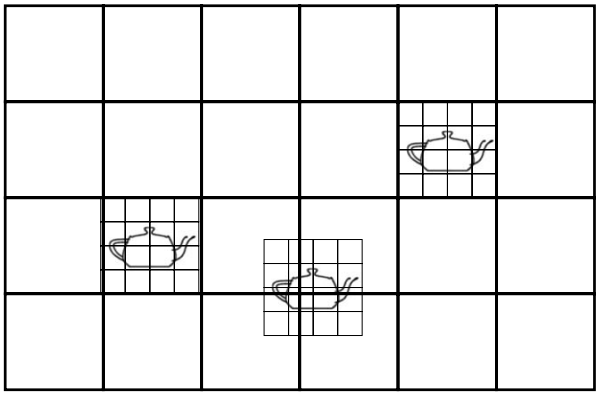


$t_{\max} < t*$ $t_{\min} < t* < t_{\max}$ $t* < t_{\min}$

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

Hierarchical Grids

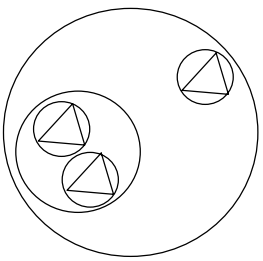
Good compromise preferred by many practitioners



Hierarchical Bounding Volumes

Create of tree of bounding volumes

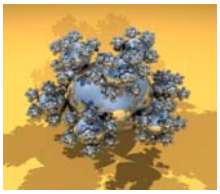
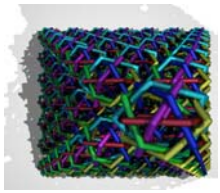
- **Children are contained within parent**
- **Creation preprocess**
  - **From model hierarchy**
  - **Automatic clustering**



Search

```
intersect(node,ray,hits) {
  if( intersectp(node->bound,ray)
    if( leaf(node) )
      intersect(node->prims,ray,hits)
    else
      for each child
        intersect(child,ray,hits)
}
```

Comparison


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

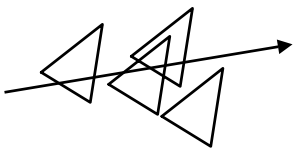
V. Havran, Best Efficiency Scheme Project
`http://sgi.felk.cvut.cz/BES/`

Theoretical Nugget 1

Computational geometry of ray shooting

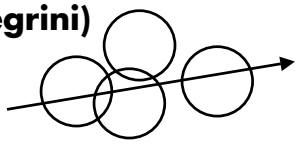
1. Triangles (Pellegrini)

- **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