Ray-Scene Intersection

Acceleration techniques
- Bounding volume hierarchies
- Spatial partitions
  - Uniform grids
  - Octrees
  - BSP trees

Beyond rays
- Beam tracing
- etc.

Bounding Volumes
- Check for intersection with simple shape first
  - If ray doesn’t intersect bounding volume, then it doesn’t intersect its contents

Bounding Volume Hierarchies
- Build hierarchy of bounding volumes
  - Bounding volume of interior node contains all children

Bounding Volume Hierarchies
- Use hierarchy to accelerate ray intersections
  - Intersect node contents only if hit bounding volume

Intersection
Intersection(Ray ray, Scene scene)

min_t = infinity
min_primitive = NULL

For each primitive in scene
  t = Intersect(ray, primitive);
  if (t < min_t) then
    min_primitive = primitive
    min_t = t
  
return Intersection(min_t, min_primitive)
Bounding Volume Hierarchies III

- Sort hits & detect early termination

```
FindIntersection(Ray ray, Node node)
{
    // Find intersections with child node bounding volumes
    ...
    // Sort intersections front to back
    // Process intersections (checking for early termination)
    min_t = infinity;
    for each intersected child i {
        if (min_t < bv_t[i]) break;
        shape_t = FindIntersection(ray, child);
        if (shape_t < min_t) { min_t = shape_t; }
    }
    return min_t;
}
```

Ray-Scene Intersection

Acceleration techniques
- Spatial partitions
  - Uniform grids
  - Octrees
  - BSP trees

Beyond rays
- Beam tracing
  - etc.

Uniform Grid

- Construct uniform grid over scene
  - Index primitives according to overlaps with grid cells

```
Uniform Grid
• Trace rays through grid cells
  - Fast
  - Incremental

Only check primitives in intersected grid cells
```

Ray-Scene Intersection

Acceleration techniques
- Spatial partitions
  - Uniform grids
  - Octrees
  - BSP trees

Beyond rays
- Beam tracing
  - etc.

Uniform Grid

- Potential problem:
  - How choose suitable grid resolution?

Too little benefit if grid is too coarse
Too much cost if grid is too fine
Octree

- Construct adaptive grid over scene
  - Recursively subdivide box-shaped cells into 8 octants
  - Index primitives by overlaps with cells

Generally fewer cells

Octree

- Trace rays through neighbor cells
  - Fewer cells
  - More complex neighbor finding

Trade-off fewer cells for more expensive traversal

Ray-Scene Intersection

Acceleration techniques
- Bounding volume hierarchies
- Spatial partitions
  - Uniform grids
  - Octrees
  - BSP trees

Beyond rays
- Beam tracing
- etc.

Binary Space Partition (BSP) Tree

- Recursively partition space by planes
  - Every cell is a convex polyhedron

Binary Space Partition (BSP) Tree

- Simple recursive algorithms
  - Example: point finding

Binary Space Partition (BSP) Tree

- Trace rays by recursion on tree
  - BSP construction enables simple front-to-back traversal
Binary Space Partition (BSP) Tree

```c
RayTreeIntersect(Ray ray, Node node, double min, double max) {
    if (node is a leaf)
        return intersection of closest primitive in cell, or NULL if none
    else
        dist = distance of the ray point to split plane of node
        near_child = child of node that contains the origin of Ray
        far_child = other child of node
        if the interval to look is on near side
            return RayTreeIntersect(ray, near_child, min, max)
        else if the interval to look is on far side
            return RayTreeIntersect(ray, far_child, min, max)
        else if the interval to look is on both side
            if (RayTreeIntersect(ray, near_child, min, dist)) return ...;
            else return RayTreeIntersect(ray, far_child, dist, max)
}
```

Other Accelerations

- Screen space coherence
  - Check last hit first
  - Beam tracing
  - Pencil tracing
  - Cone tracing
- Memory coherence
  - Large scenes
- Parallelism
  - Ray casting is "embarassingly parallelizable"
- etc.

Beam Tracing

- Trace “bundle of rays” all at once

Beam Tracing Method

- Specular reflections

3D Environment

Spatial Subdivision

Cell Adjacency Graph

Spatialized Audio

Off-Line Interactive
Beam Tracing Method

- Input is source, receiver, and 3D environment

Step 1: Spatial Subdivision

- Partition space into convex polyhedral cells

Step 2: Beam Tracing

- Trace beams through cell adjacency graph

Step 2: Beam Tracing

- Trace beams through cell adjacency graph

Step 2: Beam Tracing

- Trace beams through cell adjacency graph
Step 2: Beam Tracing

• Trace beams through cell adjacency graph

Step 2: Beam Tracing

• Store all beams in a tree data structure

Beam tree encodes regions reached by different sequences of scattering from source

Step 3: Path Generation

• For each beam containing receiver ...

Step 3: Path Generation

• Lookup propagation sequence in beam tree
Step 3: Path Generation
- Construct shortest path along sequence

Step 4: Auralization
- Apply filter for each propagation path

Beam Tracing Method
- 3D Environment
- Spatial Subdivision
- Cell Adjacency Graph
- Stationary Sources
- Beam Tracing
- Beam Trees
- Moving Receiver
- Path Generation
- Propagation Paths
- Source Audio
- Auditory Display
- Spatialized Audio

Beam Tracing Demo
- Off-Line Interactive
**Experimental Results**

- Test propagation path update rates in large environments with several reflections

**Beam Tracing Results**

- Beam tree does not necessarily grow with global complexity of environment

**Path Generation Results**

- Propagation paths updated interactively ... even for large environments

**Path Generation Video**

- 10,057 input polygons
- 8 specular reflections
- 6 updates per second

**Path Generation Demo**

**Auralization Video**

- Specular reflection only
Diagnostic Results

Summary

- Intersection acceleration techniques are important
  - Bounding volume hierarchies
  - Spatial partitions
- General concepts
  - Sort objects spatially
  - Make trivial rejections quick
  - Utilize coherence when possible
  
  Expected time is sub-linear in number of primitives

- Useful for sound propagation too!