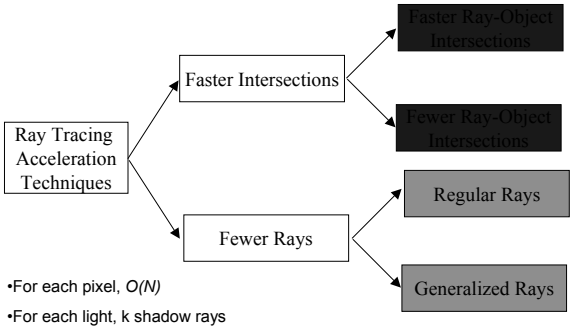
Lecture 13: Acceleration Structures

Fall 2004
Kavita Bala
Computer Science
Cornell University

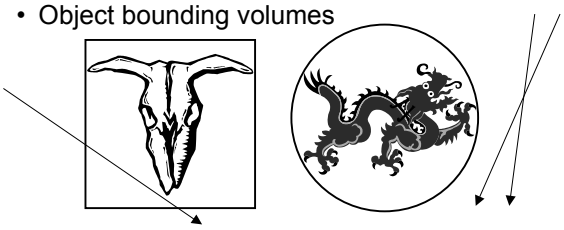
Making RT faster



- For each pixel, *O(N)*
- For each light, k shadow rays
- For GI and antialiasing: many rays per pixel

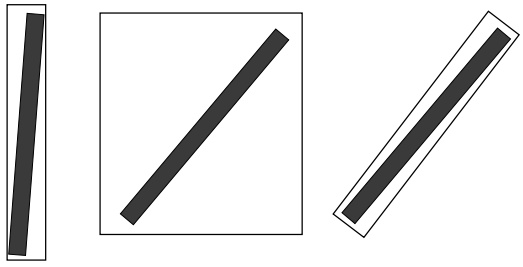
Faster Ray-Object Intersections

- Object bounding volumes



- Avoid intersection tests for expensive objects: e.g., polygon sets, spline surfaces
  - Ray/sphere or ray/cuboid test is fast

Tight Fit to Bounding Volume

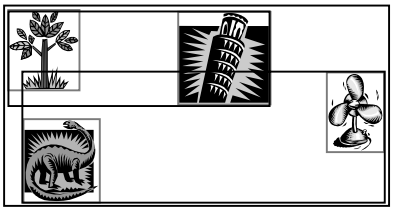


Fewer Ray-Object Intersections

- From O(N) to O(log N)
- How?
  - Apply the idea of bounding boxes hierarchically
  - Cluster objects hierarchically
  - Single intersection might eliminate cluster
- Bounding volume hierarchy
- Space subdivision
  - Octree, Kd-tree, BSP-trees

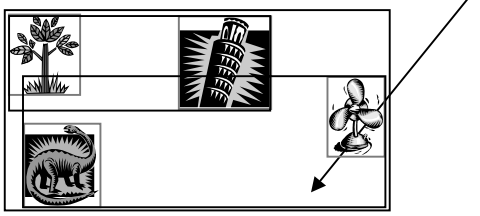
Bounding Volume Hierarchy

- Hierarchical object bounding volumes
- Spheres, axis-aligned bounding boxes (AABB), oriented bounding boxes(OBB): fast



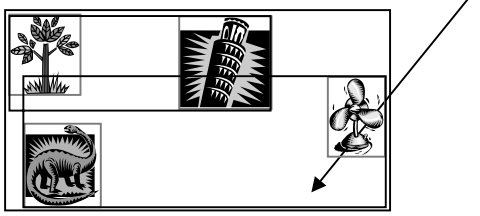
Intersection Acceleration

- Trace ray against root node
- If ray intersects node
  - Trace ray against ALL children (Recurse)



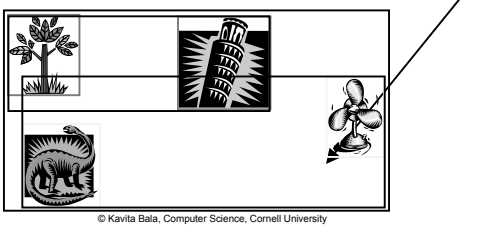
Intersection Acceleration

- Trace ray against root node
- If ray intersects node
  - Trace ray against ALL children (Recurse)



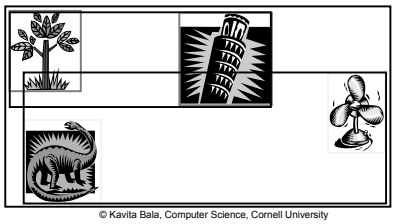
Intersection Acceleration

- If no intersection, eliminate tests with all children!



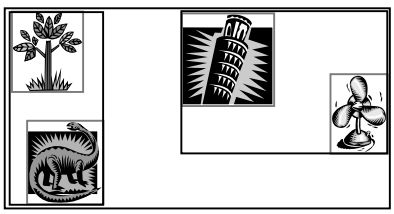
BVH: Construction

- Group objects together
  - Top-down: how to split?
  - Bottom-up: minimize surface area?



BVH: Construction

- Group objects together
  - Top-down: how to split?
  - Bottom-up: minimize surface area?

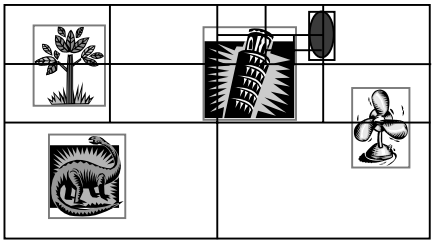


Fewer Ray-Object Intersections

- From O(N) to O(log N)
- Bounding volume hierarchy
- Space subdivision
  - Octree (Quadtree in 2D)
  - Non-uniform (kd-tree)
  - BSP-tree

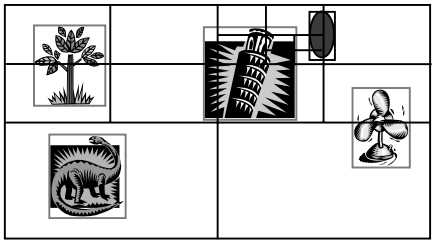
Spatial Hierarchy

- Hierarchical spatial subdivision
  - Divides up space
- Children are distinct and cover parent

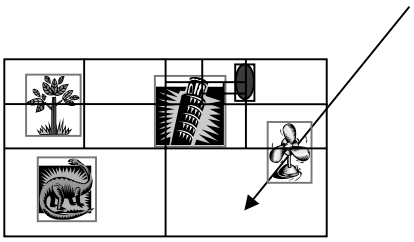


Construction

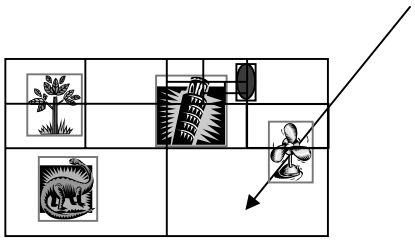
- Maximum depth
- Maximum number of elements in leaf



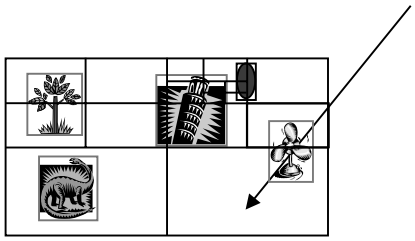
Intersection Acceleration



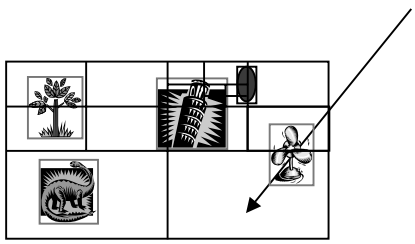
Intersection Acceleration



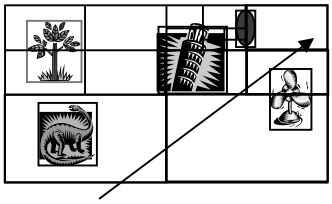
Intersection Acceleration



Intersection Acceleration



Intersection Acceleration

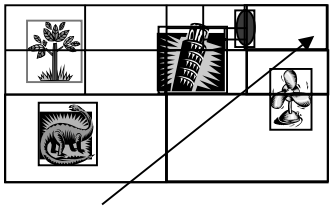


Intersection Acceleration

1. Intersect ray with root: p = root.intersect(ray)
   - If no intersection, done
2. Find p in tree (node j = root.find(p))
3. Test ray against elements in node j
   - If intersection found, done
   - Else find exit point (q) from node j, p = q, goto 2

Octree Properties

- Front to back traversal
- Problem: Same object in multiple cells
  - Split object
  - Could repeatedly intersect: use mailboxes

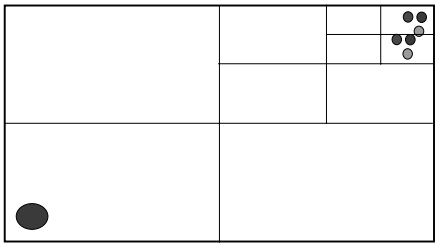


Solutions

- Split object
  - No repeated intersections and correct
  - But, could create lots of little objects
- Use mailboxes
  - Store intersection in the object: avoids repeated intersection
  - What about correctness?
    - Need to check that intersection is in “current” bounding box

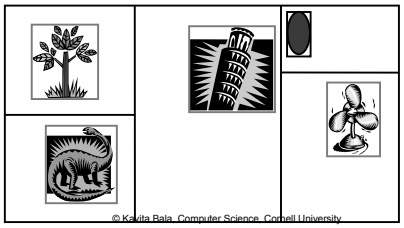
Octree Problems

- Distribution of objects
- Chops up objects



K-dimensional (kd) Tree

- Spatial subdivision
  - Subdivide only 1 dimension
  - Do not subdivide at the center
- Tracing with kd-tree unchanged

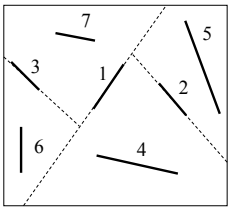


Construction

- Which axis to pick?
- What point on the axis to pick?
- One heuristic:
  - Sort objects on each axis
  - Pick point corresponding to “middle” object
  - Pick axis that has “best” distribution of objects
  - L = n/2, R = n/2 (ideal)
  - Realistically,
    - minimize (L-R) and
    - L approx. n/2, R approx. n/2

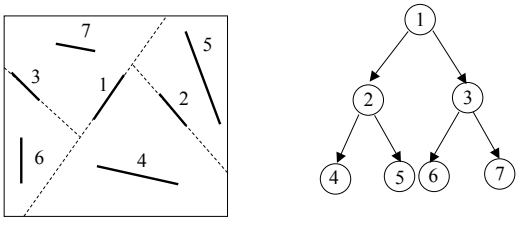
BSP Tree

- Generalization of kd-trees
- Splitting plane is not axis aligned
- Used in games: DOOM



BSP Construction

- Use a polygon to define the splitting plane
- Other objects either split or stored high up



How to construct?

- Least-crossed criterion (random selection of polygons)
  - Do not split many polygons
- Try to make it balanced

BSP Traversal

- Front to back ordering
- Strict occlusion order (not closest object)