Accelerated Raytracing
Why is Acceleration Important?

• Vanilla ray tracing is really slow!
• $m \times m$ pixels, $k \times k$ supersampling, $n$ primitives, average ray path length of $d$, $l$ lights, 2 recursive ray casts per intersection…

• What’s one way to reduce this that we’ve discussed in class?
Reducing These Terms

- Early ray termination
  - Reduce levels of recursion
- Adaptive ray sampling
  - Reduces $k$ supersampling term
- Faster intersections
- Generalized rays
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

- Generalized Rays
  - Early ray termination
  - Adaptive sampling

- Beam tracing
  - Cone tracing
  - Pencil tracing
Faster Intersections

• Optimize the intersection test
• Reduce the number of intersections along $d$ to reduce the number of intersection tests
Spatial Acceleration Structures

Idea: Index data by spatial location

• Geometry’s position in the world affects its likelihood of being seen/intersected

• Fast, approximate queries can eliminate distant or hidden objects
Uniform Spatial Subdivision

- Partition space into cells (voxels)
- Associate primitives with all cells it overlaps
- Trace through voxel array using fast incremental arithmetic
Uniform Grid Preprocessing

- Find world bounding box
- Determine grid resolution
- Place objects into cells that overlap it
Uniform Grid Traversal

3D Digital Differential Analyzer (3DDDA)

- DDA calculates increment in linear interpolation to touch all cells
- Line segment between start and end position within the grid
- Intersect geometry within visited cells
Potential Errors in Overlap

Solve redundancy and accuracy issues with mailboxes

- Each object has a mailbox to store intersection results
- Each ray has a distinct number
- Only intersect if ray does not already have an intersection
- Look for intersection only within current voxel
Spatial Hierarchies

- Cells of parent completely contains all cells of children
- If a query fails in a cell, it will fail for all children
- If a query succeeds, try the children
- Only if query reaches a leaf do we perform the expensive intersection test
Non-Uniform Spatial Subdivision

Partitioning Approach:
   Octrees, k-d trees, BSP trees

Non-partitioning approach:
   Bounding volume hierarchies
Spatial Partitions

kd-tree  oct-tree  bsp-tree
Octree

- Eight children per parent
- Objects in leaf nodes
- Extents of cube can be inferred or stored
Octree Problems

Octrees unbalanced if objects aren’t uniformly distributed

“Teapot in the stadium”
k-d Trees

Octree problem: Cube always split evenly between children

Solution: Allow for splits at variable positions!
**k-d Tree Properties**

- Node represents rectilinear region (axis-aligned)
- Axis-aligned planes split region into two
- Direction of cutting planes alternate with depth
  - Cut planes in different sub-trees at same sub level not necessarily the same
k-d Tree Example
k-d Tree Build
Where to Split Nodes?

How to decide which split plane to use in a given node? What are some possible criteria?
Split Plane Criteria

Balance cost of building k-d tree with cost of traversal (number of steps) and intersection test
Spatial Median Splitting

- Simplest “naive” method
- Dimension of sub-tree’s split plane $p$ chosen in round robin fashion
- Plane positioned at spatial median of voxel
- Terminate recursion when number of triangles in node are below a threshold, or tree depth exceeds maximum depth
Spatial Median Issues

• Does not consider geometry of scene
• Can gain speed ups by maximizing empty space toward root node
Surface Area Metric

Representation of ray traversal cost for a split:

\[ \text{Cost(split)} = C_s + P_L C_L + P_R C_R \]

\(C_s\) is cost of node traversal (constant)
\(C_L\) and \(C_R\) are costs of left and right children (based on number of primitives on that side)
\(P_L\) and \(P_R\) are probabilities of ray hitting that child (assumes uniform ray distribution, so is surface area ratio of child to parent)
Surface Area Heuristic

Greedy, top down approach to tree construction

1. Consider candidate splits for scene partitioning
2. Evaluate candidates using SAM
3. Partition along best candidate and recurse to left and right child
Axis-Aligned Bounding Boxes

Further reduce candidate split planes

Build time is important since k-d tree is rebuilt after every time step!
k-d Tree Traversal

Stack:

Current:
Root
Stack: R

Current: LL
Stack:

Current: R
Stack: RR
Current: RL
Stack:

Current: RR

Diagram showing a tree structure with nodes labeled from L to RR, indicating a path through the nodes.
Stack:

Current: RRR
BSP Trees

Binary Space Partition Trees

- Cutting planes can be any orientation
- Industry standard for spatial subdivision in many game environments
Bounding Volume Hierarchies

- Bounding volume for each object
  - Sphere, AABBs, etc
- Parents bounds bound child’s bounds
- No notion of cells
- Bounding volumes can overlap
BVH Example

 Ideally have:
 Tight bounding volumes
 Balanced trees

Intersect with largest B.V...
...then intersect with children...
...until you reach the leaf nodes - the primitives.
Current Global Illumination
Bidirectional Pathtracing

Shoot rays from light source in addition to scene

Gather rays from pixel based on light sources on surfaces
Naturally Captures:

- Soft shadows
- Caustics
- Depth of field
- Motion blur
- Indirect light
- Glossy and blurred reflections
Brigade

https://www.youtube.com/watch?v=FbGm66DCWok
Cascading Light Propagation Volumes

https://www.youtube.com/watch?v=vPQ3BbuYVh8
VXGI

https://www.youtube.com/watch?v=cH2_RkfStSk
Enlighten

https://www.youtube.com/watch?v=6Fs-4Kvp2nM
Additional Examples

http://realtimeradiosity.com/demos/