Spatial Data Structures and Hierarchies

CS535

Daniel G. Aliaga
Department of Computer Science
Purdue University
Spatial Data Structures

• Store geometric information
• Organize geometric information
• Permit fast access to/of geometric information
• Applications
  – Heightfields
  – Collision detection (core to *many* uses)
  – Simulations (e.g., surgery, games)
  – Rendering (e.g., need to render fast!)
Hierarchical Modeling

• Concept is old but fundamental
  – “Hierarchical geometric models for visible surface algorithms”, James Clark - 1976
Hierarchical Modeling

- **Trees** and Scene Graphs
Hierarchical Modeling

- Trees and Scene Graphs
Hierarchical Modeling

• Trees and Scene Graphs
Bounding Volumes

• Problem:
  – Suppose you need to intersect rays with a scene...
  – Suppose you have a scene divided into objects...

• Solution: bottom-up
  – Wrap complex objects into simple ones
    • Boxes, spheres, other shapes...
  – Organize into a tree
Bounding Volume Hierarchy (BVH)

• How to building an axis aligned bounding box (AABB) BVH?
• How to intersect?
• Complexity? Problem cases?
AABB BVH

• Example construction
  – Given M 2D points, use k-means clustering to determine clusters
  – Then group nearby clusters (e.g., use Voronoi diagram or Delaunay triangulation)
  – And iteratively form a tree from the bottom-up
  – In each node, approximate the contained points using an axis-aligned bounding box
    • e.g., box = [min of all contained pts, max of all contained pts]
Bounding Volume Hierarchy (BVH)

- How to build an oriented bounding box (OBB) BVH?
- How to intersect?
- Complexity? Problem cases? Advantages over axis-aligned?
OBB BVH

• Example construction
  – Similar to AABB BVH but “fit” an oriented box to the points within each cluster/node of the tree
  – Methods:
    • Sample possible rotations and sizes in order to pick the best box
    • Compute distance of points to a line and optimize the line equation parameters until finding the line that best approximates all points
    • Then compute a box width – consider the benefit/cost of the box size
      – e.g., totally containing all points might make the box very large; could also choose to mostly contain the points – however, what does this mean with regards to operations using the BVH?
Space Subdivision

• Binary tree / Quadtree / Octree

• k-D tree

• Binary Space Partitioning (BSP) Tree
A directed edge refers to the link from the parent to the child (the arrows in the picture of the tree).

The root node of a tree is the node with no parents; there is at most one root node in a rooted tree.

A leaf is a node that has no children.

The depth of a node is the length of the path from the root to the node. The root node is at depth zero.

The height of a tree is the depth of its furthest leaf. A tree with only a root node has a height of zero.

Siblings are nodes that share the same parent node.

Size = 9
Height = 3
Root node = 2
Binary Tree

• Operations
  – Search
  – Insert
  – Delete
Quadtree

- Similar to binary-tree, but have 4 children per node
- Each node corresponds to one of four rectangular regions of the current quad
Quadtree

• Similar to binary-tree, but have 4 children per node
• Each node corresponds to one of four rectangular regions of the current quad
• Point quadtree
  – Partitions depend on the data
  – The quad is divided using the previous point within it
Quadtree

• **Point quadtree**
  – Partitions depend on the data
  – The quad is divided using the previous point within it

• **Advantage**
  – Data dependent subdivision reduces (unnecessary) number of quads

• **Disadvantage**
  – Quads do not tightly approximate region surrounding the point
• Matrix (MX) quadtree
  – Location of partition lines independent of the data
  – The occupied nodes are all subdivided until a tight fitting box
Quadtree

• MX quadtree
  – Location of partition lines independent of the data
  – The occupied nodes are all subdivided until a tight fitting box

• Advantage
  – Quads leaf nodes always tightly approximate region surrounding the point

• Disadvantage
  – Potentially lots of levels from root to a point
• **Point Region (PR) quadtree**
  – Location of partition lines independent of the data
  – The nodes are all subdivided until \( p \) or less points per node (e.g., \( p=1 \))
Quadtree

• PR quadtree
  – Location of partition lines independent of the data
  – The nodes are all subdivided until p or less points per node (e.g., p=1)

• Advantage
  – Partition lines are known and paths from root to point is only as long as needs to be

• Disadvantage
  – Quads do not tightly approximate region surrounding the point
Quadtree

• Comparison

Point QT  MX QT  PR QT
Demo

• [http://donar.umiacs.umd.edu/quadtree/](http://donar.umiacs.umd.edu/quadtree/)
Octree

• Analogous to Quadtree but extended to 3D
• Each node is divided into eight subboxes
Octree

• Analogous to Quadtree but extended to 3D
• Each node is divided into eight subboxes
• Similar, there are
  – Point octrees
  – MX octrees
  – PR octrees
K-D tree

• Partition each dimension in a cyclical fashion
  – Thus, can be applied to 2D, 3D, or higher dimensions

• Each node stores a next partitioned “half-space” of data points (or of the data space)
**k-D tree**

- The first split (red) cuts the root cell (white) into two
- Each of which is then split (green) into two subcells
- Each of those four is split (blue) into two subcells
- The final eight called leaf cells
- The yellow spheres represent the tree vertices

A 3-dimensional kd-tree

The resulting kd-tree decomposition

The resulting kd-tree
Demo

- [http://donar.umiacs.umd.edu/quadtree/](http://donar.umiacs.umd.edu/quadtree/)
Binary Space Partitioning (BSP)

• Similar to k-D tree but splitting lines/planes are not necessarily axis-aligned
• Can adapt better to data
• Was algorithm used for visibility sorting...
Binary Space Partitioning (BSP)

- Suitable for any number of dimensions

Separating planes are shown in black and objects in blue.

BSP trees
Demo

• More stuff at
  – http://donar.umiacs.umd.edu/quadtree
  – R-tree?

• See