Space Partitioning for Broad Sweep Collision Detection
Part 2 - Quadtrees

Game Design Experience
Professor Jim Whitehead
February 13, 2009
Announcements

• Technical Design Document
  ▶ Due today
  ▶ May turn it in to the box by my door by the end of the day
• Have midterm exams
  ▶ See me after class to pick up
  ▶ Answer key is on class website
**Broad vs Narrow Sweep**

- With many small objects in large playfield
  - Each object only has the potential to collide with nearby objects
- Broad sweep
  - Perform a quick pass over n objects to determine which pairs have potential to intersect, p
- Narrow sweep
  - Perform p x p check of object pairs that have potential to intersect
- Dramatically reduces # of checks
Broad sweep approaches

- Grid
  - Divide playfield into a grid of squares
  - Place each object in its square
  - Only need to check contents of square against itself and neighboring squares
  - See http://www.harveycartel.org/metanet/tutorials/tutorialB.html for example

- Space partitioning tree
  - Subdivide space as needed into rectangles
  - Use tree data structure to point to objects in space
  - Only need to check tree leaves
  - Quadtree, Binary Space Partition (BSP) tree

- Application-specific
  - 2D-shooter: only need to check for collision against ship
  - Do quick y check for broad sweep

Point Quadtree (Wikipedia)
Trees

- A tree is a data structure
  - Places contents into nodes
  - Each node has
    - Contents
    - Pointers to 0 or more other levels
  - Children are represented as references

```
// A simple tree node class
Class Node {
  List<Node> children;  // List of references to Nodes
  int contents;
}
```

A tree with 5 nodes (root, a, b, c, d).
Root node has contents 56.
Node a has contents 45, and two children, c and d.
Node b has contents 75, and no children.
Node c has contents 24, and no children.
Node d has contents 51, and no children.
Tree Operations

- Adding a child
  - Create new node
    - n = new Node();
  - Add it to list of children of parent
    - parentNode.children.Add(n);

- Removing a child
  - Find and remove child node from list of children
    - parentNode.children.Remove(childToRemove);
    - (childToRemove is reference to Node of child to be deleted)
Point Quadtree

- A tree where each node has four children
  - Each level represents subdividing space into 4 quadrants
  - Each node holds information about one quadrant
  - A deeper tree indicates more subdivisions

**MX Quadtree Demo**

[http://donar.umiacs.umd.edu/quadtree/points/mxquad.html](http://donar.umiacs.umd.edu/quadtree/points/mxquad.html)

Demo both points and rectangles
C# Node Representation

```csharp
class Node
{
    Point min[4];
    Point max[4];
    int level;
    Node Quad[4];  // holds 4 quadrants
    List<IGameObject> objectList;  // list of game objects in a Node
}
```

- quads is an array with four elements
  - Each element is a reference to a Node
    - Quad[0] – upper left, etc.
  - A recursive data structure
    - Nodes hold pointers to Nodes
- Min[] is an array with four elements
  - Each element is upper left point of quadrant
- Max[] holds lower right point of each quadrant
- Level indicates how many levels down in the tree
  - Useful for bounding the depth of the tree
Adding object to tree

Insert(Node n, IGameObject g)

• Iterate through all 4 quadrants of current node (n)
  ■ If at maximum depth of the tree
    ▷ Add IGameObject to objectList and return
  ■ If AABB (bounding box) of IGameObject lies fully within a quadrant
    ▷ Create child node for that quadrant, if necessary
    ▷ Then call insert on that node (recursion)
  ■ Otherwise, AABB does not lie in just one quadrant
    ▷ Add to contents list at this level

• First call is Insert(root, g)
  ■ That is, start at root node for insertions
Finding, removing object in tree

Find(Node n, IGameObject g)

• Iterate through all 4 quadrants of current node (n)
  ▶ Check if AABB (bounding box) of IGameObject spans multiple quadrants
    • Yes: IGameObject is in this node. Return node.
  ▶ At maximum level of tree?
    • Yes: IGameObject is at this level
  ▶ If AABB (bounding box) of IGameObject lies fully within a quadrant
    • Call find on that node (recursion)

• Removing object from tree
  ▶ Found_node = Find(root, g)
  ▶ Found_node.objectList.Remove(g)