Today

- Spatial Data Structures
  - Why care?
  - Octrees/Quadtrees
  - Kd-trees

Spatial Data Structures

- Spatial data structures store data indexed in some way by their spatial location
  - For instance, store points according to their location, or polygons, …
  - Before graphics, used for queries like “Where is the nearest McDonalds?” or “Which stars are strong enough to influence the sun?”
- Multitude of uses in computer games
  - Visibility - What can I see?
  - Ray intersections - What did the player just shoot?
  - Collision detection - Did the player just hit a wall?
  - Proximity queries - Where is the nearest power-up?

Spatial Decompositions

- Focus on spatial data structures that partition space into regions, or *cells*, of some type
  - Generally, cut up space with planes that separate regions
  - Always based on tree structures (surprise, huh?)
- *Octrees (Quadtrees)*: Axis aligned, regularly spaced planes cut space into cubes (squares)
- *Kd-trees*: Axis aligned planes, in alternating directions, cut space into rectilinear regions
- *BSP Trees*: Arbitrarily aligned planes cut space into convex regions

Using Decompositions

- Many geometric queries are expensive to answer precisely
  - All of the questions two slides back fall into this category
- The best way to reduce the cost is with fast, **approximate** queries that eliminate most objects quickly
  - Trees with a containment property allow us to do this
  - The cell of a parent completely contains all the cells of its children
  - If a query fails for the cell, we know it will fail for all its children
  - If the query succeeds, we try it for the children
  - If we get to a leaf, we do the expensive query for things in the cell
- Spatial decompositions are most frequently used in this way
  - For example, if we cannot see any part of a cell, we cannot see its children, if we see a leaf, use the Z-buffer to draw the contents

Octree Gems Ch 4.10

- Root node represents a cube containing the entire world
- Then, recursively, the eight children of each node represent the eight sub-cubes of the parent
- Quadtree is for 2D decompositions - root is square and four children are sub-squares
  - What sorts of games might use quadtrees instead of octrees?
- Objects can be assigned to nodes in one of two common ways:
  - All objects are in leaf nodes
  - Each object is in the smallest node that fully contains it
  - What are the benefits and problems with each approach?

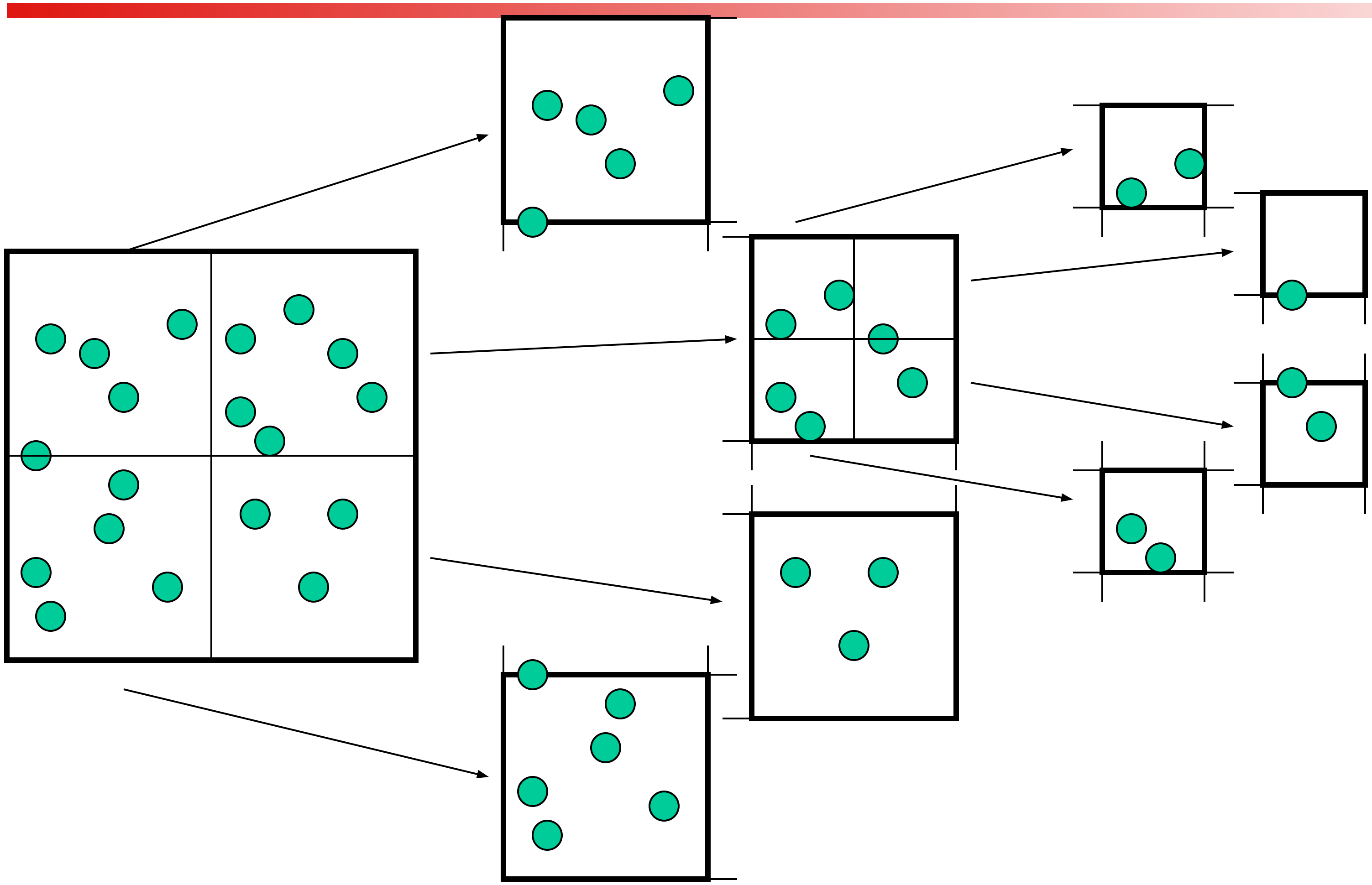
Octree Node Data Structure

- What needs to be stored in a node?
  - Children pointers (at most eight)
  - Parent pointer - useful for moving about the tree
  - Extents of cube - can be inferred from tree structure, but easier to just store it
  - List of pointers to the contents of the cube
    - Contents might be whole objects or individual polygons, or even something else
  - Neighbors are useful in some algorithms (but not all)

Building an Octree

- Define a function, buildNode, that:
  - Takes a node with its cube set and a list of its contents
  - Creates the children nodes, divides the objects among the children, and recurses on the children, or
  - Sets the node to be a leaf node
- Find the root cube (how?), create the root node and call buildNode with all the objects
- When do we choose to stop creating children?
  - Is the tree necessarily balanced?
- What is the hard part in all this? Hint: It depends on how we store objects in the tree

Example Construction



Assignment of Objects to Cells

- Basic operation is to intersect an object with a cell
  - What can we exploit to make it faster for octrees?
- Fast(est?) algorithm for polygons (Graphics Gem V):
  - Test for trivial accept/reject with each cell face plane
    - Look at which side of which planes the polygon vertices lie
    - Note speedups: Vertices outside one plane must be inside the opposite plane
  - Test for trivial reject with edge and vertex planes
    - Planes through edges/vertices with normals like (1,1,1) and (0,1,1)
  - Test polygon edges against cell faces
  - Test a particular cell diagonal for intersection with the polygon
  - Information from one test informs the later tests. Code available online

Polygon-Cell Intersection Tests: Poly-Planes Tests

Images from Möller and Haines

- Planes are chosen because testing for inside outside requires summing coordinates and a comparison
  - Eg. Testing against a plane with normal (1,1,0) only requires checking x+y against a number (2 for a unit cube)
  - What tests for the other planes?

Polygon-Cell Intersection Tests: Edge-Cube Test

Images from Möller and Haines

- Testing an edge against a cube is the same as testing a point (the center of the cube) against a swept volume (the cube swept along the edge)

Polygon-Cell Intersection Tests: Interior-Cube Test

Images from Möller and Haines

- Test for this type of intersection by checking whether a diagonal of the cube intersects the polygon
  - Only one diagonal need to be checked
  - Which one?

Approximate Assignment

- Recall, we typically use spatial decompositions to answer **approximate** queries
  - Conservative approximation: We will sometimes answer yes for something that should be no, but we will never answer no for something that should be yes
- Observation 1: If one polygon of an object is inside a cell, most of its other polygons probably are also
  - Should we store lists of objects or polygons?
- Observation 2: If a bounding volume for an object intersects the cell, the object probably also does
  - Should we test objects or their bounding volumes? (There is more than one answer to this - the reasons are more interesting)

Objects in Multiple Cells

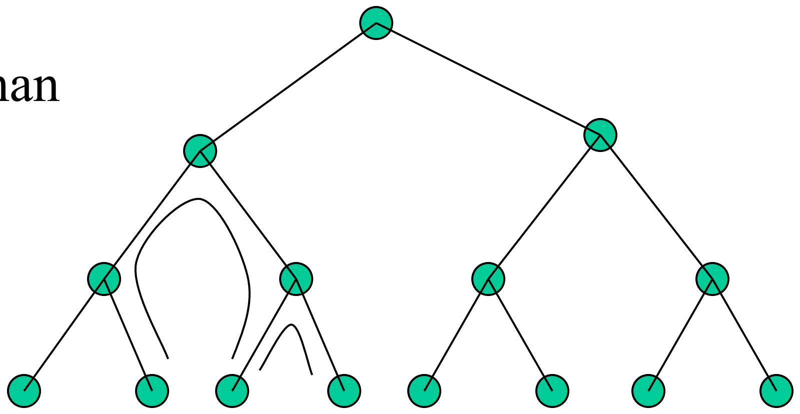
- Assume an object intersects more than one cell
- Typically store pointers to it in all the cells it intersects
  - Why can’t we store it in just one cell? Consider the ray intersection test
- But it might be considered twice for some tests, and this might be a problem
  - One solution is to flag an object when it has been tested, and not consider it again until the next round of testing
    - Why is this inefficient?
  - Better solution is to tag it with the frame number it was last tested
    - Subtle point: How long before the frame counter overflows?
- Also read Gems Ch 4.11 for another solution

Neighboring Cells

- Sometimes it helps if a cell knows it neighbors
  - How far away might they be in the tree? (How many links to reach them?)
  - How does neighbor information help with ray intersection?
- Neighbors of cell A are cells that:
  - Share a face plane with A
  - Have all of A's vertices contained within the neighbor's part of the common plane
  - Have no child with the same property

Finding Neighbors

- Your right neighbor in a binary tree is the leftmost node of the first sub-tree on your right
  - Go up to find first rightmost sub-tree
  - Go down and left to find leftmost node (but don't go down further than you went up)
  - Symmetric case for left neighbor
- Find all neighbors for all nodes with an in-order traversal
- Natural extensions for quadtrees and octrees



Frustum Culling With Octrees

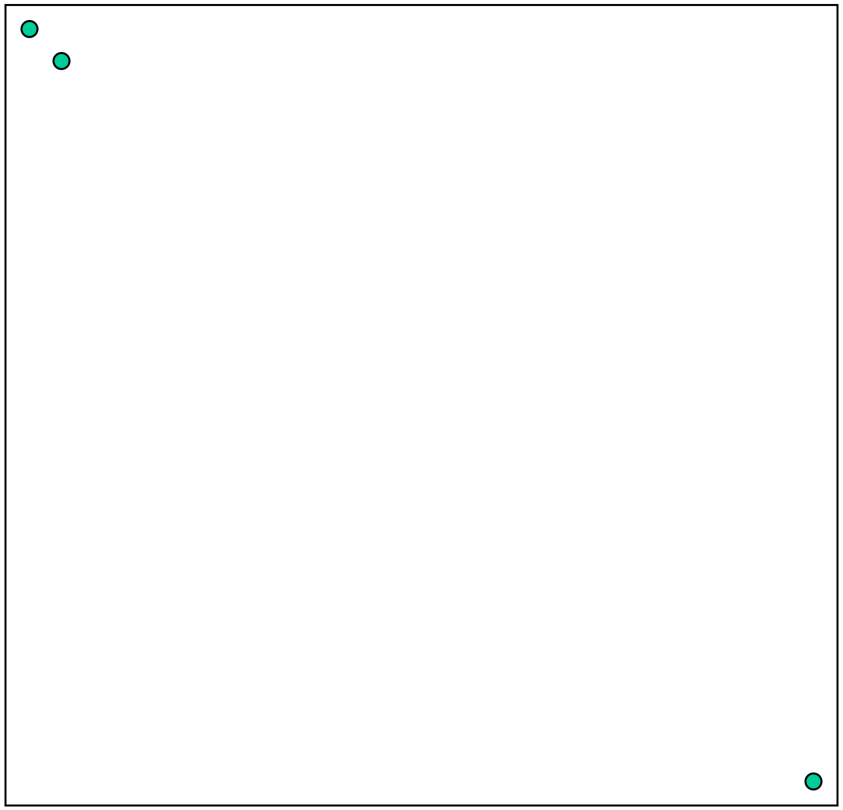
- We wish to eliminate objects that do not intersect the view frustum
- Which node/cell do we test first? What is the test?
- If the test succeeds, what do we know?
- If the test fails, what do we know? What do we do?

Frustum Culling With Octrees

- We wish to eliminate objects that do not intersect the view frustum
- Have a test that succeeds if a cell may be visible
  - Test the corners of the cell against each clip plane. If all the corners are outside one clip plane, the cell is not visible
  - Otherwise, is the cell itself definitely visible?
- Starting with the root node cell, perform the test
  - If it fails, nothing inside the cell is visible
  - If it succeeds, something inside the cell might be visible
  - Recurse for each of the children of a visible cell
- This algorithm with quadtrees is particularly effective for a certain style of game. What style?

Octree Problems

- Octrees become very unbalanced if the objects are far from a uniform distribution
  - Many nodes could contain no objects
- The problem is the requirement that cube always be equally split amongst children


A bad octree case