Collision Detection

448430
Spring 2009
4/13/2009
Kyoung Shin Park
Multimedia Engineering
Dankook University

Overview

- Why is collision detection important?
- Types of Collision Detection
- Approaches to Collision Detection and Response in NPSNET
- Static Object Collision Detection and Response
- Dynamic Object Collision Detection and Response
- UNC Collision Detection Algorithms (Ming Lin, etc.)
- Time Critical Algorithm (Philip Hubbard, Cornell University)

Why is collision detection important?

- Especially important in Interactive Virtual Environments.
- Realism suffers dramatically when a vehicle such as a submarine either drives through the pier, or through a ship.
- Needed for our simulations to be viable training tools.
- No matter how good the graphics and textures look, the poor realism resulting from a lack of collision detection breaks the suspension of disbelief.

Types of Collision Detection

- Static(Fixed) Object
  - Moving objects checking for collisions with fixed objects such as terrain, trees, buildings, etc.
- Dynamic
  - Moving objects checking for collisions with other moving objects.

Approaches to Collision Detection and Response

- Ignore altogether (NPSNET I).
- Implement fixed object collision detection and response only (NPSNET-IV).
- Implement both fixed object and dynamic object collision detection and response (NPSNET-II).

Static Objects

- Easy to test for collisions with static (fixed) objects.
- They do not move!!
- Commonly implemented in VR applications since it is relatively easy.
- Good enough if you only have a single moving entity, or entities are restricted in their movement such that collisions between them is not possible.

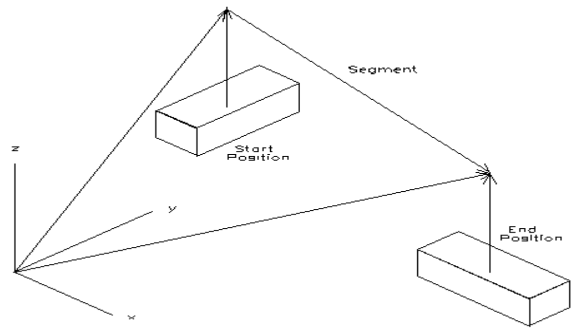
Static Objects in NPSNET

- Added as part of NPSNET-II.
- Still implemented in NPSNET-IV.
- Checks for fixed objects at elevations less than 1,000 meters.
- Reorient a unit vector in the z direction, to the object's heading, pitch, and roll.

Static Objects in NPSNET

- Add to starting and calculated ending positions for the interval to get global coordinate reference.
- Subtract start position from ending position to get segment along path just followed.
- Check to see if we hit any fixed objects along these segments.

Static Objects in NPSNET



Static Objects in NPSNET

- If we did, and the object is a ground vehicle or human entity, reset object to travel in a direction opposite of the previous segment, at a speed of -1.0
- If we did, and the object is something else, the object is DEAD.
- Check to see if we hit the ground or water along these segments.

Static Objects in NPSNET

- If we did, the vehicle is DEAD.
- There are some exceptions to this for other objects on the ground such as bridges and tunnels which are considered to be part of the ground for collision detection purposes.
- Some specific entities such as the ship and submarine are a little different to account for the fact that they operate in the water.

Dynamic Objects in NPSNET

- NPSNET-IV does not support dynamic object collision detection.
- NPSNET-II supported dynamic object collision detection.
- Lost in the development process, probably to improve performance, and lack of interest in improving the algorithm.

UNC Collision Detection

- I-COLLIDE
  - An Interactive and Exact Collision Detection System for Large-Scale Environments.
- OOBTree
  - A Hierarchical Structure for Rapid Interference Detection.

I-COLLIDE

- Avoids "Brute Force" Method which compares the bounding volume of every object in the VE with every other object in the VE.
- Very expensive computationally, $O(n^2)$.
- Acceptable performance if only a small number of objects.
- Objective is to report all geometric contacts between objects.

I-COLLIDE

- In an interactive VE, we do not know the positions and orientation in advance, as the user can change them at any time.
- Therefore, assumes objects motions can not be expressed as a closed form function of time.
- Collision detection is currently on of the major bottlenecks in such environments.

I-COLLIDE

- This algorithm trims the $O(n^2)$ of n simultaneously moving objects using coherence to speed up pairwise interference tests and reduce the number of these tests.
- Complexity is reduced to $O(n + m)$, where m is the number of objects *very close* to each other.

I-Collide

- First, a coarse check is done to see if objects have potentially collided. Checks to see if the overall bounding volumes of objects intersect (Sweep and Prune).
- Second, if a collision has occurred, determine the exact collision position (Exact Collision detection).

Temporal and Geometric Coherence

- Temporal coherence means that the application state does not change significantly between time steps, or frames.
- The objects move only slightly from frame to frame.

Temporal and Geometric Coherence

- The slight movement of the objects translates into geometric coherence, because their geometries (vertex coordinates) change minimally between frames.
- Underlying assumption is that time steps are small enough that objects do not travel large distances between frames.

Sweep and Prune

- Objective is to reduce the number of pairs of objects that are actually compared, to give O(n + m).
- Assumes each object is surrounded by a 3-D bounding volume.
- Use Dimension Reduction to sort the objects in 3-space.

3-D Bounding Volumes

- Use Fixed-Size Bounding Cubes, or Dynamically-Resized Rectangular bounding bounding boxes.
- Fixed size bounding cubes add less overhead, as their size remains constant.
- It is only necessary to translate them with the object and recompute min and max x,y,z.

3-D Bounding Volumes

- Dynamically resized bounding boxes are the "tightest" axis-aligned box containing the object at a particular orientation.
- More overhead due to recomputation of min, max x,y,z values since they are not a constant distance from the center of the bounding volume.

3-D Bounding volumes

- Dynamic resizing works well with oblong objects, which results in fewer overlaps.
- In walkthrough environments, with few objects moving, the savings gained by fewer pairwise comparisons outweighs the cost of recomputing dynamic bounding box volumes.

Dimension Reduction

- If two bodies collide in 3-D space, their orthogonal projections onto the xy, yz, and xy planes, and x, y, and z axes must overlap.
- This is why the bounding boxes are axis aligned.
- Can effectively project the bounding boxes onto lower dimension, and sort them.

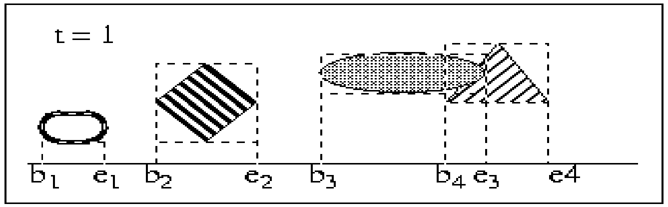
Dimension Reduction

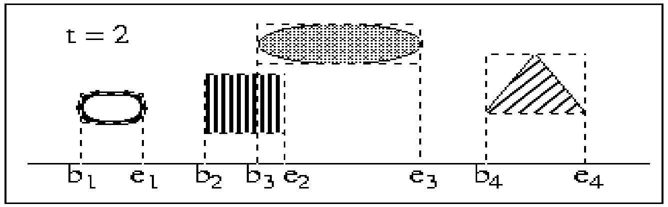
- Works better than space partitioning approaches (NPSNET-II) where it is difficult to choose good partition sizes.

1-D Sweep and Prune

- Bounding Volumes projected onto x, y, and z axis.
- Two objects intersect, if and only if their projections onto all three axis intersect.
- Have one list for each dimension, which is sorted with Bubble sort or insertion sort.
- Maintain Boolean flag which only changes if swaps are made on sorted lists.

1-D Sweep and Prune





1-D Sweep and Prune

- If flags are true for all 3 dimensions, then we pass this pair on to exact collision detection.
- Bubble sort works well if few objects move.
- Insertion sort works well where large numbers of objects move locally.
- Due to temporal coherence, individual lists are likely almost sorted already.

1-D Sweep and Prune

- Both sorts swap only adjacent items, making it convenient to maintain an overlap status for each polytope pair.

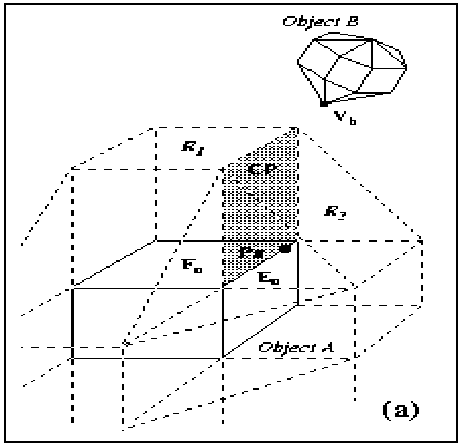
Exact Collision Detection

- Tracks Closest pairs between Convex polytopes.
- Each polytope has Voronoi regions associated with each of its features (faces, edges, vertices)
- Voronoi regions are those points that are closer to the given feature than to any other feature, and are external to the object.

Exact Collision Detection

- In order for two feature to be the closest, each must lie in the Voronoi region of the other.
- If either feature fails the test, step to the next feature which is on the other side of the bounding plane that caused the test to fail.
- When a feature pair fails the test, the next pair is guaranteed to be closer.

Exact Collision Detection



Exact Collision Detection

- Initial features to compare are chosen arbitrarily.
- Successive tests narrow down to closest features.
- Due to coherence, future tests will produce closest features that are near previous closest features.

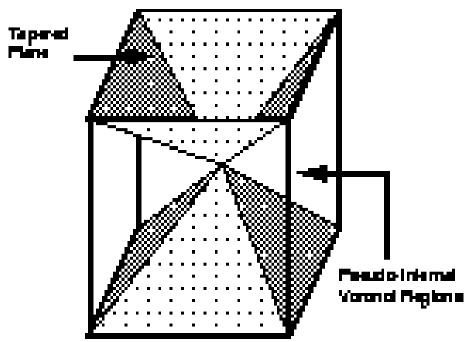
Exact Collision Detection

- Even if objects are moving and changing direction rapidly, reasonable performance is assured by the fact that each feature has a constant number of bounding planes for its Voronoi region.
- If the objects have penetrated one another, a closest pairs feature will not be found. Use an interior pseudo-Voronoi region to solve.

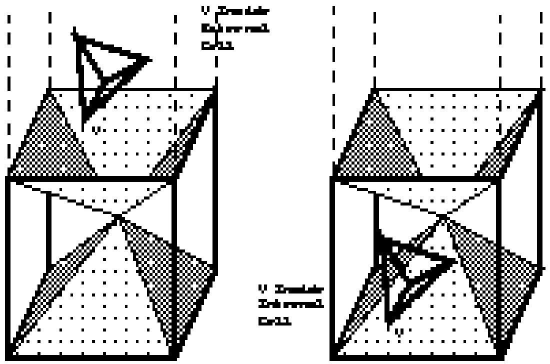
Penetration Detection

- When polytopes interpenetrate, some features may not fall into any Voronoi regions.
- Pseudo-Voronoi regions partition the interior space of the convex polytopes.

Penetration Detection



Penetration Detection



Extension to non-convex objects

- Use hierarchical tree representation of the object where interior nodes may or may not be convex, but all the leaf nodes are convex.
- The bounding volume of each node, is the union of the bounding volume of all its children.
- Recursively traverse tree to determine exact collision.

Extension to non-convex objects

- If there is a collision between a pair of parent nodes, algorithm expands their children.
- If children collide, recursively proceeds down the tree to determine if collision has occurred.
- Finds exact collision point.

Extension to non-convex objects

- Some other algorithms utilize similar techniques, but with different types of tree structures.
- Various techniques are available for subdividing the bounding volume of the object to build the tree structure.

OOBTree

- Applicable to general polygonal models.
- Pre-computes a hierarchical representation of models using tight-fitting oriented bounding box trees (OOBTrees).
- At runtime, traverse the trees, checking for overlaps between oriented bounding boxes based on separating axis theorem.

Separating axis theorem

- Project bounding box onto axis (not necessarily a coordinate axis).
- Each box forms an interval on the axis.
- If the intervals don't overlap, the axis is called the separating axis for the boxes.
- If they do overlap, further tests may be required.

Time Critical Algorithm

- Collision detection algorithms for Interactive Virtual worlds must support real-time performance, and be able to deal with motion that is controlled by a user.
- Few collision detection algorithms support both of these requirements.
- Detects collisions between simplified representations of objects.

Time Critical Algorithm

- These representations support progressive refinement. Having detected a collisions at a given level, proceed to a representation with higher level of detail.
- All bounding volumes are spheres, with more spheres defined at each level.
- This process may be interrupted at any level by the application to improve performance.

Time Critical Algorithm

- May sacrifice accuracy for time.
- Is this a good approach? A question of debate amongst researchers. The UNC papers criticize it.
- It all depends on the application. User's may find it easier to deal with less accuracy, if it overcomes "time-lag" induced simulator sickness.

Time Critical Algorithm

- There are other algorithms to consider. Broad Phase - first step of algorithm which uses upper bounds on the objects' accelerations to build a set of space-time bounds.
- Narrow Phase - second step progressively refines the accuracy of the collision detection.

Broad Phase

- Builds a set of space-time bounds, a 4-D structure.
- Space-time bounds serve as conservative objects as to where the object will be in the future.
- Utilizing these bounds, calculate the time (t_i) at which a collision is possible between two objects.

Broad Phase

- Until t_i is reached, no collisions are possible, and the broad phase need not do any work during the intervening frames.
- When t_i is reached, again enter the broad phase and determine is a collision has occurred utilizing low level-of-detail space-time bounding box.

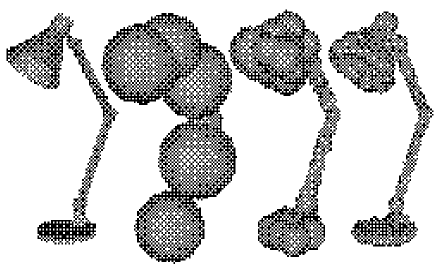
Broad Phase

- If no collision occurs at t_i, build a new set of space-time bounds, compute a new t_i, and continue.
- If bounding boxes intersect at t_i, switch to the narrow phase of refinement.

Narrow Phase

- Progressively refines the approximation of the object surfaces. Bounding spheres are split into two or more spheres to provide a higher level-of-detail bounding volume that looks more and more like the actual object.
- After each repetition, allows itself to be interpreted by the application.

Levels of Detail Refinement



Narrow Phase

- Proceeds through progressive levels of refinement until objects no longer intersect, or the process is interrupted.
- Thus, the accuracy of the collision detection depends on the time that can be allocated for the narrow phase refinement.
- Returns to broad phase for next loop.