

XBox Project Cluster, E2006

Collision Detection

Ken Friis Larsen

kfl@itu.dk

IT University of Copenhagen

Monday October 2, 2006

What Is Collision Detection?

The term **collision detection** is often for any of the following:

- ▶ Detecting the occurrence of a collision between two objects
- ▶ Detecting a collision occurrence and the point of contact (to some degree of accuracy)
- ▶ Detecting the point of contact and modelling a response.

Why Is Collision Detection Important?

Collision Detection is needed/can be used by many different parts of a game engine:

- ▶ The physics engine
- ▶ The render for scene management
- ▶ Camera movement
- ▶ The AI engine

Taxonomy Of Collision Detection Algorithms

- ▶ **Broad phase/narrow phase:** Use two algorithms: one broad for culling away pairs of objects that cannot possibly collide, and one narrow for precise testing.
- ▶ **Single phase:** Use a single partitioning for the entire process. That is, no initial culling.
 - ▶ Binary Space Partition (BSP) Trees are commonly used here

Broad Phase Strategies

- ▶ Temporal coherence
 - ▶ Check 4D space-time objects, estimate next time for collision detection
- ▶ Spatial coherence
 - ▶ divide the world into unit cells
 - ▶ potential collision if two objects are in the same cell
- ▶ Use bounding volumes
 - ▶ Spheres
 - ▶ Axis Aligned Bounding Box (AABB)
 - ▶ Oriented Bounding Box (OBB)
 - ▶ Discrete orientation polytope (k -dop)
- ▶ Hierarchical optimization of the above

Which Kind Of Bounding Volume Should We Use?

- ▶ In general, the tighter the bounding volume, the slower the test.
- ▶ However, the tighter the bounding volume, the fewer unnecessary tests are needed.
- ▶ Cost function:

$$T = N_v C_v + N_p C_p + N_u C_u$$

Where

T the total cost

N_v the number of bounding volume pair overlap tests

C_v the cost of bounding volume test

N_p the number of primitive pair tests

C_p the cost of a primitive pair test

N_u the number of updated bounding volumes due to motion

C_u the cost for updating a bounding volume

Bounding Volume Hierarchies

One way to build bounding volume hierarchies is to build them as so that the leafs are normal bounding volumes whereas the internal nodes are just simple **aggregates** of (the bounding volumes of) their children.

Testing Bounding Volumes Hierarchies For Collision

```
bool FindVolumeHit(BVH A, BVH B) {
    if ( A.IsLeaf && B.IsLeaf ) {
        test intersection of primitive bounding volumes
    } else if ( !A.IsLeaf && !B.IsLeaf ) {
        if ( A.Volume > B.Volume )
            foreach( BVH C in A.Children )
                if ( FindVolumeHit(C, B) ) return true;
        else
            foreach( BVH C in B.Children )
                if ( FindVolumeHit(C, A) ) return true;
    } else if ( A.IsLeaf && !B.IsLeaf ) {
        foreach( BVH C in B.Children )
            if ( FindVolumeHit(C, A) ) return true;
    } else {
        foreach( BVH C in A.Children )
            if ( FindVolumeHit(C, B) ) return true;
    }
    return false;
}
```

Collision Detection With OBBs

Two OBBs are disjoint if:

- ▶ they can be separated by a plane containing a face.
- ▶ or, if they are disjoint and cannot be separated by a face then they can be separated with a plane parallel to an edge from each box.

That gives us a total of 15 tests:

$$3 \text{ faces} + 3 \text{ faces} + 3 \text{ edges} \times 3 \text{ edges}$$

Rather than the 144 test if we had just done the naive intersection test, where we would test each edge on each face.

Narrow Phase

Just use the straightforward test

For two convex polyhedra X and Y (concave can be decomposed into convex):

1. Test: Vertexes of X within Y and vice versa.
2. Test: Edges of X penetrating Y and vice versa.
3. Test: Centroid of faces of X within Y . (Identical and aligned objects)

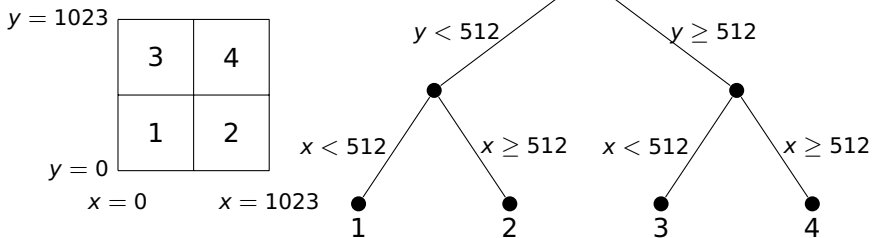
Single Phase

- ▶ Partition the world using some kind of partitioning method, and perform exact collision detection immediately.
- ▶ Popular partition methods
 - ▶ Octrees
 - ▶ Sphere trees
 - ▶ BSP trees

BSP Trees Overview

- ▶ Two different types:
 - ▶ Axis-aligned
 - ▶ Polygon-aligned
- ▶ General idea:
 - ▶ Divide space with a plane
 - ▶ Sort geometry into the space it belongs
 - ▶ Done recursively

A Simple BSP Tree



Using BSP Trees For Collision Detection

- ▶ It is nice to BSP tree for collision detection as well as 3D rendering
- ▶ The BSP for an object can be seen as a “iterative bounding volume” that shrinks the 3D space until it completely encloses the object.
- ▶ To test intersection between two objects, we simply merge their BSP trees, i.e. the first object’s BSP tree is used to partition the second object.

Summary

- ▶ Collision detection is a simplification of physical modeling, focusing on efficient implementations of a pure geometrical problem.
- ▶ Still, we need to provide the physics subsystem with necessary information for computation of collision response.
- ▶ We normally use bounding volumes of different kinds to simplify initial intersection testing.
- ▶ Bounding volumes can be arranged in hierarchies to allow for recursively defined multiscale resolution.
- ▶ Specific collision detection implementations include OBB-trees, as well as making use of the BSP tree of the world for collision detection.

Demo Day

All groups should give a demo showing their halfway status at October 23.