

31 OCTOBER 2007
COLLISION DETECTION AND DISTANCE COMPUTATION

Haptic Interaction: Human-computer interaction where the human “feels” forces on the cursor as it encounters physical forces in its virtual world.

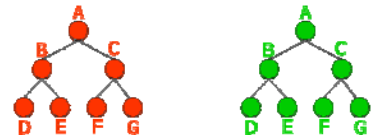
Collision detection methods:

1. Grid method

- a. Good for many simple objects of same size
- b. Normally, checking every object (disc) for collision with all the others is $O(n^2)$
- c. Algorithm:
 - i. Make a grid of the space. Know how many discs can fit in one grid-square
 - ii. Iterate over all discs (objects) and create data structure that maps grid squares to discs inside of them.
 - iii. To check whether a disc collides with any others: Check the distance between its center and all other disc centers within local region of 9 (3x3) grid squares.
- d. Local checking region is a bounded space → there is a maximum number of discs inside → algorithm is $O(n)$.

2. Bounding Volume Hierarchy (BVH)

- a. Good for few, geometrically complex objects
- b. Big idea: enclose objects by simple geometric shape (e.g. sphere) and run radius-based collision check. If they collide, recursively break shapes down into smaller shapes and repeat until either nothing collides or we are convinced of collision.



- c. We like to pre-compute BVH trees for each object (binary tree with largest bound at the root node)

- d. To check collision between two objects:

- i. Search downward on the two binary trees (this makes our search tree).



- ii. In each step in the search tree, we check collision for all combinations of node pairs chosen from both trees (this means every node in the search tree has 4 child nodes, because the two BVH trees are binary ($2^2=4$)).
- iii. We prune the search tree as we go down. That is, we only continue down nodes that have “potential collision” in them.
- iv. If we still believe there is a collision after having reached the bottom of the search tree (smallest bound radius), we run a collision check on the edges of the shape that pass through this last bounded region.

- e. Loophole that we should also consider in BVH: Object entirely inside object

- f. Properties of BV shapes that we want:

- i. Tightness
 - 1. OBB
- ii. Efficient testing
 - 1. Sphere, AABB
- iii. Invariance

1. Sphere
 2. OBB
- g. → We usually use Spheres or OBBs.
- h. Properties of BVHs that we want: i. Separation, ii. Balanced tree.