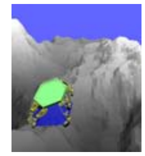
Collision Detection

1

Many Different Situations

Few moving objects, but complex geometry

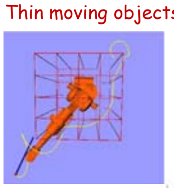
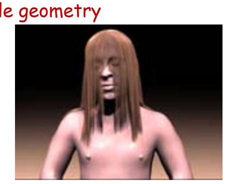
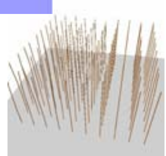
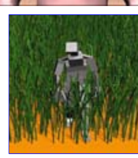




2

Many Different Situations

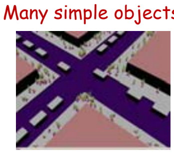
Thin moving objects, with simple geometry

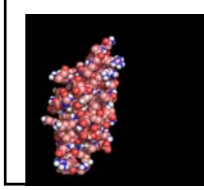
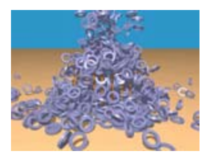





3

Many Different Situations

Many simple objects

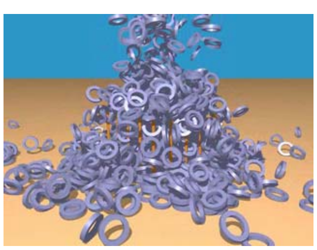


4

Many Different Situations

Many simple objects



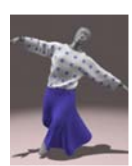
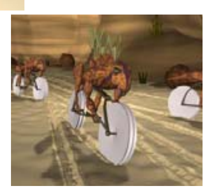
5

Many Different Situations

Many contacts, deformable objects, ...





Many Different Situations

Real-time detection

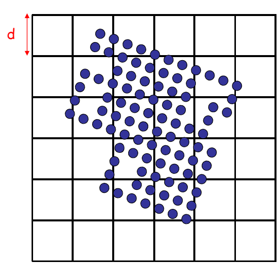
7

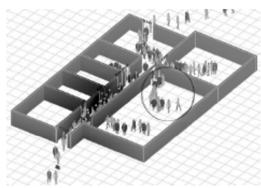
Collision Detection Methods

- Many different methods
- We will focus on two of them:
  - Grid method: good for many simple moving objects of about the same size (e.g., many moving discs with similar radii)
  - Bounding Volume Hierarchy (BVH) method: good for few moving objects with complex geometry

8

Grid Method

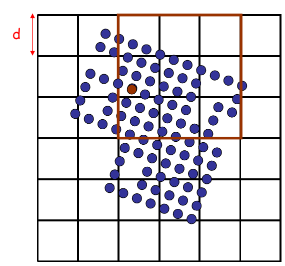


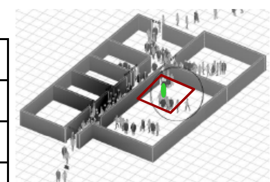


- Subdivide space into a regular grid of cubic or square bins
- Index each object in a bin

9

Grid Method



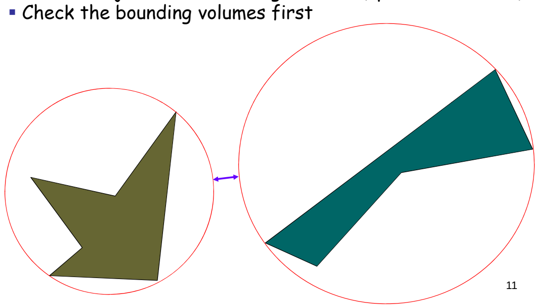


Running time is proportional to number of moving objects

10

Bounding Volume Hierarchy Method

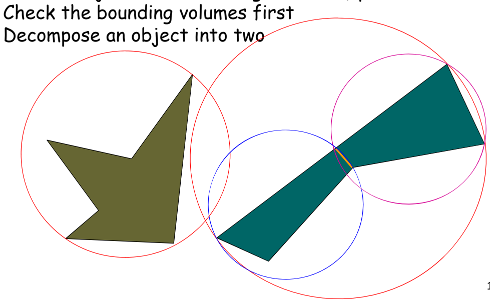
- Enclose objects into bounding volumes (spheres or boxes)
- Check the bounding volumes first



11

Bounding Volume Hierarchy Method

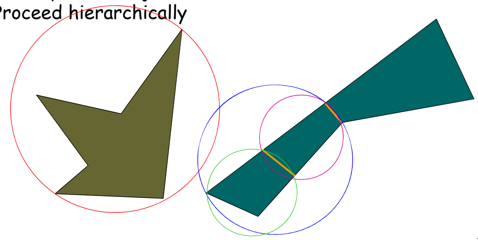
- Enclose objects into bounding volumes (spheres or boxes)
- Check the bounding volumes first
- Decompose an object into two



12

Bounding Volume Hierarchy Method

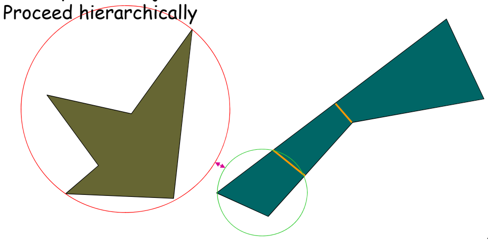
- Enclose objects into bounding volumes (spheres or boxes)
- Check the bounding volumes first
- Decompose an object into two
- Proceed hierarchically



13

Bounding Volume Hierarchy Method

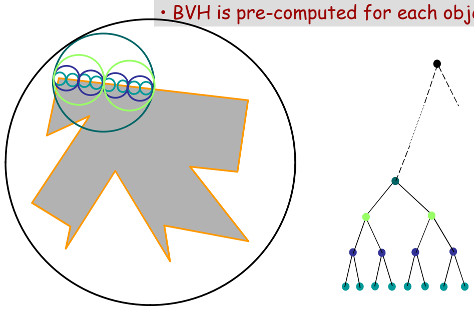
- Enclose objects into bounding volumes (spheres or boxes)
- Check the bounding volumes first
- Decompose an object into two
- Proceed hierarchically



14

Bounding Volume Hierarchy Method

- BVH is pre-computed for each object

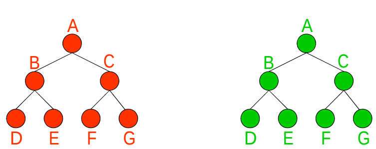


15

BVH in 3D

16

Collision Detection

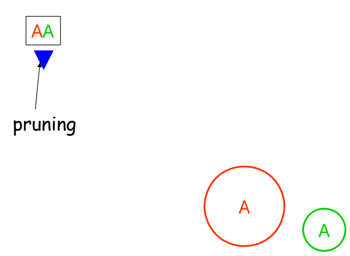


Two objects described by their precomputed BVHs

17

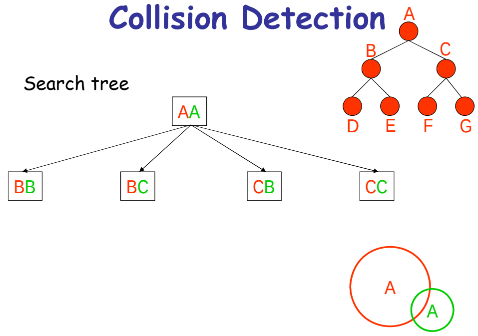
Collision Detection

Search tree

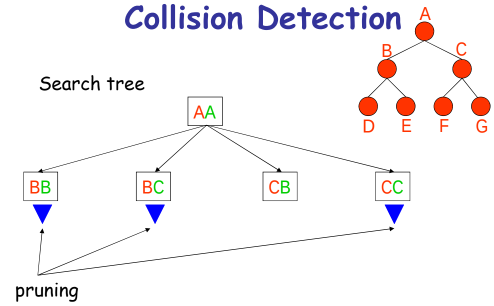


18

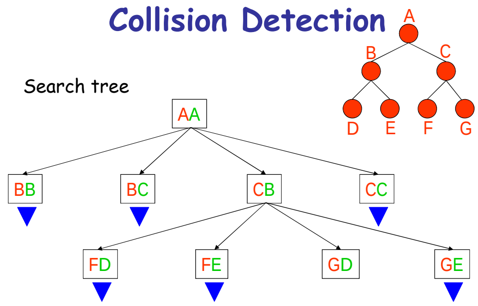
Collision Detection



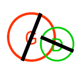
Collision Detection



Collision Detection



If the pieces contained in G and D overlap → collision



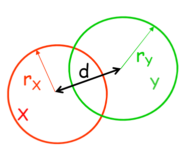
Collision Detection

- Pruning discards subsets of the two objects that are separated by the BVs
- Each path is followed until pruning or until two leaves overlap
- When two leaves overlap, their contents are tested for overlap

Search Strategy

- If there is no collision, all paths must eventually be followed down to pruning or a leaf node
- But if there is collision, it is desirable to detect it as quickly as possible
- → Greedy best-first search strategy with $f(N) = d/(r_X+r_Y)$

[Expand the node XY with largest relative overlap (most likely to contain a collision)]

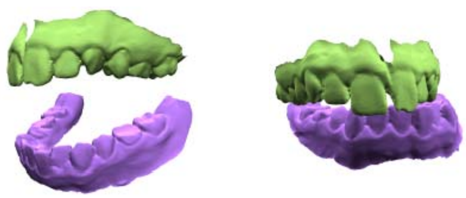


Performance

Several thousand collision checks per second for 2 three-dimensional objects each described by 500,000 triangles, on a 1-GHz PC

Faster when objects are well separated or have much overlap.
Slower when objects barely overlap or are very close.
Why?

Bad Case for BVH



Desirable Properties of BVs and BVHs

BVs:

?

26

Desirable Properties of BVs and BVHs

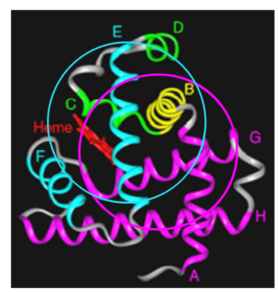
BVs:

- Tightness
- Efficient testing
- Invariance

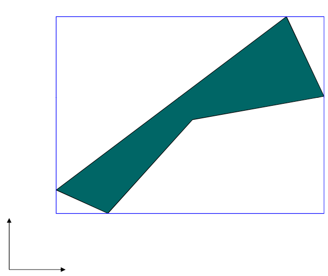
27

Spheres

- Invariant
- Efficient to test
- But tight?



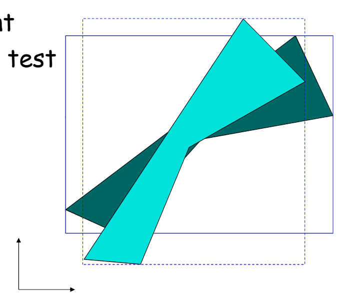
Axis-Aligned Bounding Box (AABB)



29

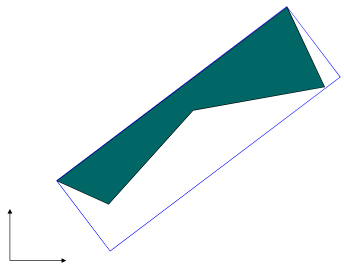
Axis-Aligned Bounding Box (AABB)

- Not invariant
- Efficient to test
- Not tight



30

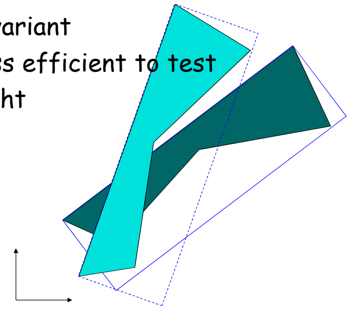
Oriented Bounding Box (OBB)



31

Oriented Bounding Box (OBB)

- Invariant
- Less efficient to test
- Tight



32

Comparison of BVs

	Sphere	AABB	OBB
Tightness	-	--	+
Testing	+	+	o
Invariance	yes	no	yes

No type of BV is optimal for all situations

33

Desirable Properties of BVs and BVHs

BVs:
- Tightness
- Efficient testing
- Invariance

BVH:

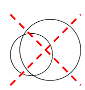
?

34

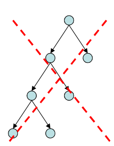
Desirable Properties of BVs and BVHs

BVs:
- Tightness
- Efficient testing
- Invariance

BVH:



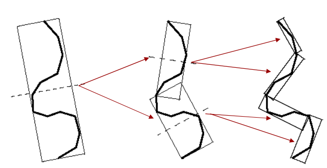
- Separation
- Balanced tree



35

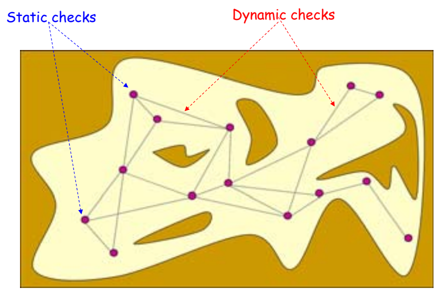
Construction of a BVH

- Top-down construction
- At each step, create the two children of a BV
- Example:
  For OBB, split longest side at midpoint



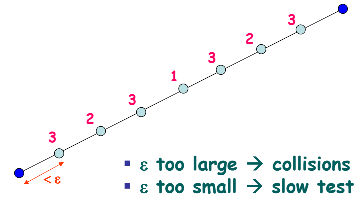
36

Static vs. Dynamic Collision Detection

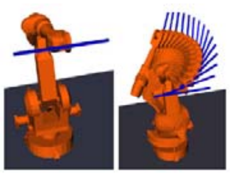


Usual Approach to Dynamic Checking (in PRM Planning)

1) Discretize path at some fine resolution ε
2) Test statically each intermediate configuration

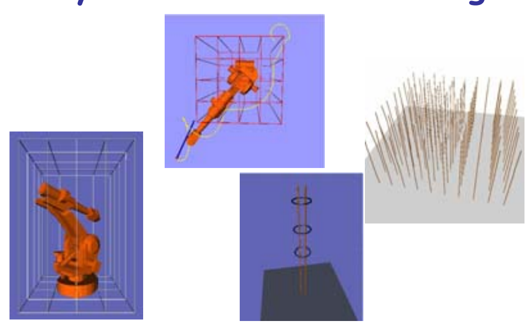


- ε too large → collisions are missed
- ε too small → slow test of local paths



- ε too large → collisions are missed
- ε too small → slow test of local paths

Examples of Difficult Dynamic Collision Checking



Adaptive Bisection

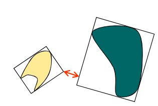
Ideas:

a) Relate configuration changes to path lengths in workspace
b) Use distance computation rather than pure collision checking
c) Bisect adaptively

Greedy Distance Computation
(same recursion as collision detection)

Greedy-Distance(A,B)

1. If dist(A,B) > 0, then return dist(A,B)
2. If A and B are both leaves, then return distance between their contents
3. Switch A and B if A is a leaf, or if B is bigger and not a leaf
4. Set A_1 and A_2 to be A's children
5. $d_1 \leftarrow$ Greedy-Distance(A_1,B)
6. If $d_1 > 0$ then
   a. $d_2 \leftarrow$ Greedy-Distance(A_2,B)
   b. If $d_2 > 0$ then return Min(d_1,d_2)
7. Return 0



What about deformable objects?

43