Building a physics engine - part 4: broad phase of collision detection

Dr. Giuseppe Maggiore

NHTV University of Applied Sciences
Breda, Netherlands

Table of contents

Broad phase of collision detection

Increasing performance, in general

- What is the fastest instruction?

Broad phase of collision detection

Increasing performance, in general

- What is the fastest instruction?
- The one that is not run!

Broad phase of collision detection

Increasing performance, in collision detection

- How do we increase performance in a collision detection system?
- Quickly and cheaply exclude pairs of colliders
- Process known as *collision culling*
  - We *ensure lack* of collisions
  - *Presence is ensured* only during narrow phase
- Akin to frustum/occlusion culling

Collision culling

Bounding spheres

- An obvious choice is bounding spheres
- Identical w.r.t. rotation
- Fast to check against other spheres

Collision culling

Intersection of bounding spheres

- Two spheres, $\langle C_0, r_0 \rangle$ and $\langle C_1, r_1 \rangle$
- Intersection when $|C_1 - C_0| \leq r_1 + r_0$
- Intersection also when $|C_1 - C_0|^2 \leq (r_1 + r_0)^2$

Collision culling

Intersection of bounding spheres

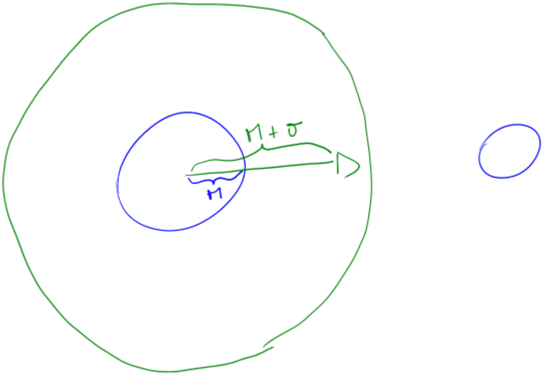
- If the spheres are moving, then we can increase their radii by their speed
- Or we can project their relative speed

Collision culling

Intersection of bounding spheres

- If the spheres are moving, then we can increase their radii by their speed
- Or we can project their relative speed
- $\sigma = |(V_1 - V_2) \cdot \frac{C_1 - C_0}{|C_1 - C_0|}|$
- $|C_1 - C_0| \leq r_1 + r_0 + \sigma$

Moving spheres

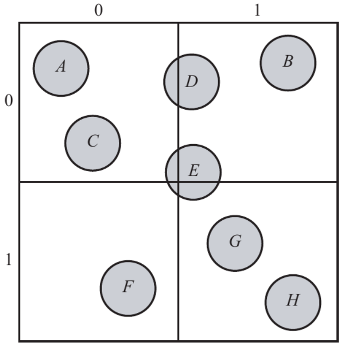


Space partitioning

Space partitioning

- We can also decompose space in axis-aligned-bounding-boxes ("bins")
- Even earlier no-collision determination
- This would reduce the number of sphere-to-sphere checks

SLIDE



Space partitioning

Space partitioning

- We can divide space in *bins*; each bin is an AABB
- Sphere intersection with an AABB bounded by points L and U
- No intersection if $|C_j - L_j| \leq r_j + \sigma_j$ or $C_j - U_j| \leq r_j + \sigma_j$ for all axes $j = x, y, z$

Space partitioning

Space partitioning

- When a sphere moves, it only moves to a neighbouring bin; less checks
- We can find the right bin directly with modulus operations (hashing)

Axis aligned bounding boxes

AABB intersection

- A simple and powerful algorithm exists for determining intersection groups of AABBs
- It is particularly fast, especially if the AABBs do not move too much between frames

Axis aligned bounding boxes

AABB intersection

- Update AABBs (if needed)
- Insertion sort the extremes of each box; one list for every axis (2 for 2D, 3 for 3D, etc.)
  - After the first frame the list is *nearly sorted*
  - $O(n)$ complexity

Axis aligned bounding boxes

AABB intersection

- Run sweep algorithm
    - Active intervals $= \emptyset$
    - When a beginning value is encountered, add it as intersecting all active intervals; add it to active intervals
    - When end value is encountered, remove it from the active intervals
- Intersections must be confirmed across all axes

Oriented bounding boxes

OBB

- An OBB is characterized by a center and three directions (columns of the rotation matrix)
- The vertices are $P = C + \sigma_0 e_0 U_0 + \sigma_1 e_1 U_1 + \sigma_2 e_2 U_2$
  - $\sigma_i = 1$ or $\sigma_i = -1$
  - e_i are the half extents

Oriented bounding boxes

OBB SAT

- With the separating axis test, it may seem that we need to test 6 face normals for one, 6 for the other, and $12^2 = 144$ edge pair cross products
- That's quite a lot!

Oriented bounding boxes

OBB SAT

- The OBB is symmetric, so many tests are redundant
  - Three unique face directions
  - Three unique edge directions
- The minimum number of required tests is 3 face normals for one, 3 for the other, and $3^2 = 9$ edge pair cross products

Oriented bounding boxes

OBB SAT

- We project both OBBs onto one of the unique potential separating directions $Q + tD$
- We look for an extremal vertex such that $\max_P D \cdot (P - Q)$
- $D \cdot (P - Q) = D \cdot (C + \sigma_0 e_0 U_0 + \sigma_1 e_1 U_1 + \sigma_2 e_2 U_2 - Q)$

Oriented bounding boxes

OBB SAT

- We are maximizing, so we do not try all the σ_i combinations

$$
\begin{aligned}
D \cdot (P - Q) &= D \cdot (C + \sigma_0 e_0 U_0 + \dots - Q) && (1)\\
&= D \cdot (C - Q) + \sigma_0 e_0 D \cdot U_0 + \dots && (2)\\
&= D \cdot (C - Q) + \sum_{i=0}^{2} |e_i D \cdot U_i| && (3)
\end{aligned}
$$

Oriented bounding boxes

OBB SAT

- Maximization results in

$$\max_P D \cdot (P - Q) = \underbrace{D \cdot (C - Q)}_{\gamma} + \underbrace{\sum_{i=0}^{2} |e_i D \cdot U_i|}_{r}$$

- Minimization results in
$\min_P D \cdot (P - Q) = D \cdot (C - Q) - \sum_{i=0}^{2} |e_i D \cdot U_i|$
- The interval is thus $[\gamma - r, \gamma + r]$
- **Important:** the separating directions *must be unit length*, and the edges as well

Oriented bounding boxes

OBB SAT

- We project both OBBs onto their intervals $[\gamma_1 - r_1, \gamma_1 + r_1]$ and $[\gamma_2 - r_2, \gamma_2 + r_2]$
- They intersect when $|\gamma_2 - \gamma_1| < r_1 + r_2$

Oriented bounding boxes

OBB SAT - final optimization

- Some separating directions are taken from the cross product of two edge directions: $D = U_i^1 \times U_k^2$
- When we plug those directions D in the above formulas, we get $U_i^1 \times U_k^2 \cdot U_j^1$
- We can rewrite $U_i^1 \times U_k^2 \cdot U_j^1 = U_i^1 \times U_j^1 \cdot U_k^2$
- We can cache the products $U_i^1 \times U_j^1$, so we do not have to recompute them

Moving objects

Moving objects

- We ignore the angular velocity; does not improve much, and is very complex to handle
- We can enlarge the interval radii by projecting the current relative velocity onto the separating direction
- $s = D \cdot (V_2 - V_1)$

Further optimizations

Islands

- Apply collision response system only to groups of objects in contact
- Flood-fill algorithm

Further optimizations

Sparse matrices for collision response

- Avoid multiplying lots of zeroes
- Store matrix row as list of `int * float` entries

Assignment

Assignment

- Before the end of next week
- Group-work archive/video on Natschool or uploaded somewhere else and linked in your report
- Individual report by each of you on Natschool
- Build a broad phase collision detector that supports a combination of bounding spheres, AABBs, bins, and OBBs

That's it

Thank you!