Collision Detection

Simulation and Modeling (CSCI 3010U)

Faisal Z. Qureshi

http://vclab.science.ontariotechu.ca

OntarioTech UNIVERSITY

Collision Detection

- Detect collision, and back up time to find the exact time of collision
  - Perform binary search on time to find the exact time of collision
- Too slow for many particle systems
- For many particle systems, predict the time to collide and advance the simulation to that time

List of collisions

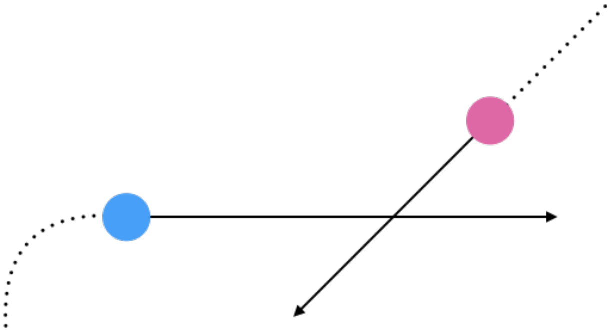
- Maintain a list of collisions, ordered on time to collision
- Don't need to re-compute the collision times in each time step, just look at the first collision on the list and process it
- We then compute a new collision time for this particle and add it to the list
- The new first time on the list becomes our new "next collision"

List of collisions

- This approach can save considerable amount of time
- Predicting "time to collision" usually assumes that the velocity is constant, so it is valid for a few time steps only
- Only track objects that may collide in the near future

Collision detection between moving objects

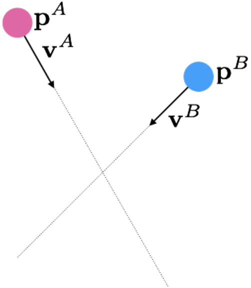
- Consider each pair of objects
- Use their paths to predict whether or not the objects will collide in the near future



Particle-Particle Collision in 2D

Consider two particles A and B with current positions and velocities $(\mathbf{p}^A, \mathbf{v}^A)$ and $(\mathbf{p}^B, \mathbf{v}^B)$ living in a 2D world.

- Position of particle A after time α: $\mathbf{p}^A(\alpha) = \mathbf{p}^A + \alpha\mathbf{v}^A$
- Position of particle B after time β: $\mathbf{p}^B(\beta) = \mathbf{p}^b + \beta\mathbf{v}^B$



- Solve α and β s.t. $\mathbf{p}^A(\alpha) = \mathbf{p}^B(\beta)$ and $\alpha, \beta \geq 0$.
- If $\alpha = \beta$, collision!

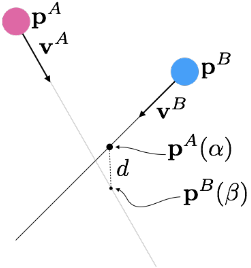
Particle-Particle Collision in 3D

- In the previous slide, we have used line intersection to see whether or not two moving particles will collide. Line intersection is a probability zero event in 3D.

Particle-Particle Collision in 3D

- We solve to following minimization problem to see if two 3D particles will collide

$$d = \min_{\alpha,\beta \geq 0} \|p(\alpha) - p(\beta)\|^2$$



- If d is less than some predefined threshold, estimate time it will take for the two particles to get to the point of intersection to see if the two particles will collide

Intersecting lines in 3d

Lines rarely intersect in 3d, so will reframe this problem to estimating the closest distance between two lines.

Consider the following two lines

$$\begin{aligned}(\text{line } 1) \quad \mathbf{p} &= \mathbf{c}_1 + \alpha\mathbf{d}_1 \\ (\text{line } 2) \quad \mathbf{q} &= \mathbf{c}_2 + \beta\mathbf{d}_2\end{aligned}$$

Line $\mathbf{p} - \mathbf{q}$ must be perpendicular to both line 1 and line 2 when $\mathbf{p}$ and $\mathbf{q}$ are closest to each other. Therefore,

$$\begin{aligned}\mathbf{d}_1^T(\mathbf{p} - \mathbf{q}) &= 0 \\ \mathbf{d}_2^T(\mathbf{p} - \mathbf{q}) &= 0\end{aligned}$$

Intersecting lines in 3d

It follows

$$\begin{aligned} \mathbf{d}_1^T\left((\mathbf{c}_1 - \mathbf{c}_1) + \alpha \mathbf{d}_1 - \beta \mathbf{d}_2\right) = & \quad 0 \\ \mathbf{d}_2^T\left((\mathbf{c}_1 - \mathbf{c}_2) + \alpha \mathbf{d}_1 - \beta \mathbf{d}_2\right) = & \quad 0 \end{aligned}$$

and

$$\begin{bmatrix} \mathbf{d}_1^T \mathbf{d}_1 - \mathbf{d}_1^T \mathbf{d}_2 \\ \mathbf{d}_2^T \mathbf{d}_1 - \mathbf{d}_2^T \mathbf{d}_2 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} -\mathbf{d}_1^T(\mathbf{c}_1 - \mathbf{c}_2) \\ -\mathbf{d}_2^T(\mathbf{c}_1 - \mathbf{c}_2) \end{bmatrix}$$

Solving the above in a least squares fashion should give us $\hat{\alpha}$ and $\hat{\beta}$.

Intersecting lines in 3d

Estimated point from line 1

$$\hat{\mathbf{x}}_1 = \mathbf{c}_1 + \hat{\alpha}\mathbf{d}_1$$

and the estimated point from line 2

$$\hat{\mathbf{x}}_2 = \mathbf{c}_2 + \hat{\beta}\mathbf{d}_2.$$

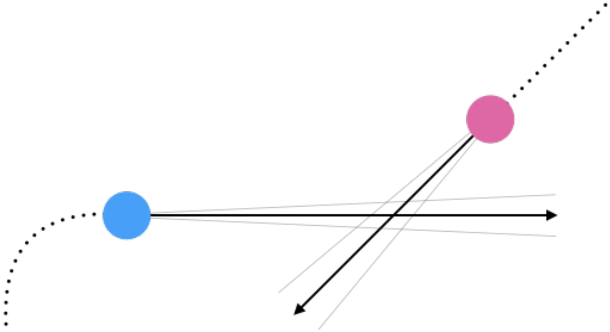
The estimated intersection point is

$$\hat{\mathbf{x}} = \frac{\hat{\mathbf{x}}_1 + \hat{\mathbf{x}}_2}{2}$$

Note that $(\mathbf{c}_1, \mathbf{c}_2, \mathbf{d}_1, \mathbf{d}_2, \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \hat{\mathbf{x}}) \in \mathbb{R}^3$. For rays, check $\hat{\alpha}, \hat{\beta} \geq 0$ and for lines, check $0 \leq \hat{\alpha}, \hat{\beta} \leq 1$

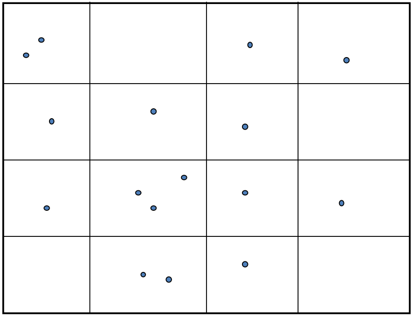
Collision detection between moving objects

- Dealing with uncertainty over time



Efficient collision detection

- Spatial partitioning

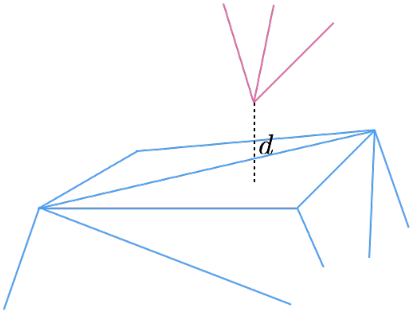


- Size of the grid cells should be several times the maximum distance that a particle can travel in time step
- Each grid cell contains a list of particles
  - Lists
  - Hash tables
  - Arrays

Collision detection for rigid bodies

- Possiblities (Polyhedral objects)
  - Vertex - Face
  - Vertex - Edge
  - Vertex - Vertex
  - Edge - Edge
  - Edge - Face
  - Face - Face
- Which of the the above situations are more likely to occur in practice?
- Complex rigid objects can have thousands of vertices, edges and faces!
  - Many systems only consider Vertex - Face collisions, claiming that other 5 options are too rare to consider

Vertex - Face collision



- Compute signed distance between a vertex location (point) and the plane representing the face
- If distance is less than or equal to zero, collision!

Speeding up Rigid Body Collisions

- Spatial partitioning
- Enclose rigid bodies into simpler shapes
  - If simple shapes don't collide then the rigid bodies won't collide as well

Collision Response

Check out notes on collision response available on the course web.

Summary

- Particle-Particle collision detection in 2D and 3D
- Collision detection between rigid bodies
- Speeding up collision detection