Realtime Physics Simulation

Alan Hazelden
4th Year CS student

alan@draknek.org
http://www.draknek.org/

What is a physics engine?

- Simulates movement of objects
  - Position; orientation
  - Velocity; rotational velocity
- Models constraints between objects
  - Most common: non-penetration
  - Also: joints, friction

What's the point?

- Games
  - Almost always need non-penetration
  - Almost always need collision detection
  - Almost always need collision resolution
- Physics engines
  - All of the above and more

Accuracy vs. efficiency

- True physics is computationally ridiculous
- So we simplify things
  - Move objects and then resolve problems
  - Simplify collision geometry
  - ”Sleep” non-moving objects
- If we can fake something, we probably should

Structure of a physics engine

- Broadphase
  - Determines which objects could potentially be colliding
- Generate contacts
  - Performs collision detection and finds contacts
- Resolve contacts
  - Find new (valid) positions for all objects

Broadphase (collision culling)

- Brute force collision testing would take $O(n^2)$ comparisons
- We can rule some collisions out very quickly
  - Bounding boxes
  - Exploiting spacial coherence
  - Exploiting temporal coherence

Broadphase (collision culling)

- Many implementations:
  - Bounding boxes for all pairs
  - Regular grid
  - Quadtree/Octree
  - BSP tree (binary space partitioning)
  - Hierarchy of bounding shapes
  - Sort and sweep algorithm

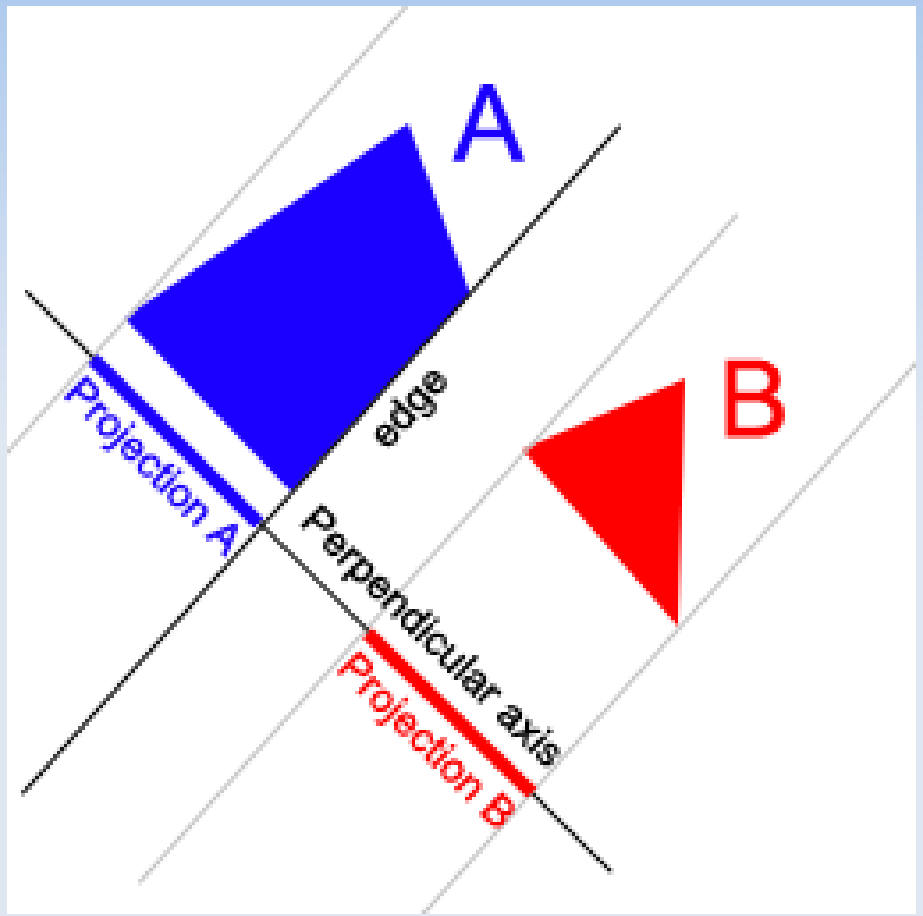
Collision detection

- Bad collision detection means bad physics
- Intersection
  - Are these two shapes touching?
- Collision
  - If these two shapes touch, tell me how and where

Contact generation

- Information generally needed:
  - Contact point
  - Contact normal
  - Amount of penetration
- Available from separating axis test
  - For convex shapes
  - More complicated algorithms exist

Separating axis test



- Try to find a separating direction
- If there is no such direction, then colliding

Collision resolution

- Resolve contacts so there is no penetration
  - In a physically realistic manner!
- Solving one contact may make another worse
- Need to find new positions
- Need to find new velocities
  - Apply impulses so they move apart
  - Take friction into account

Collision resolution

- Calculating the global solution is difficult
  - But apparently you can represent it as a massive LCP (Linear Complementarity Problem)
- Instead, we iterate over contacts repeatedly
  - Converges on global solution (hopefully)

I don't believe it's this easy!

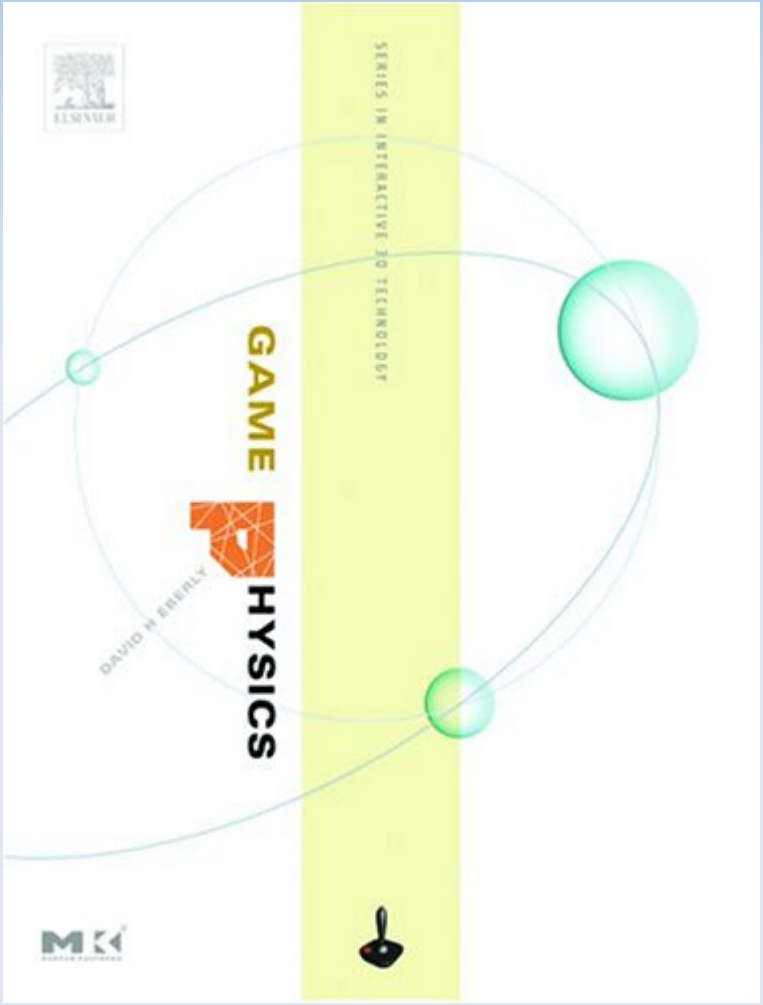
- Of course it isn't.
- Bottlenecks depend on application
- Accuracy required depends on application

- But eventually it works!
  - Just don't try it unless you hate yourself

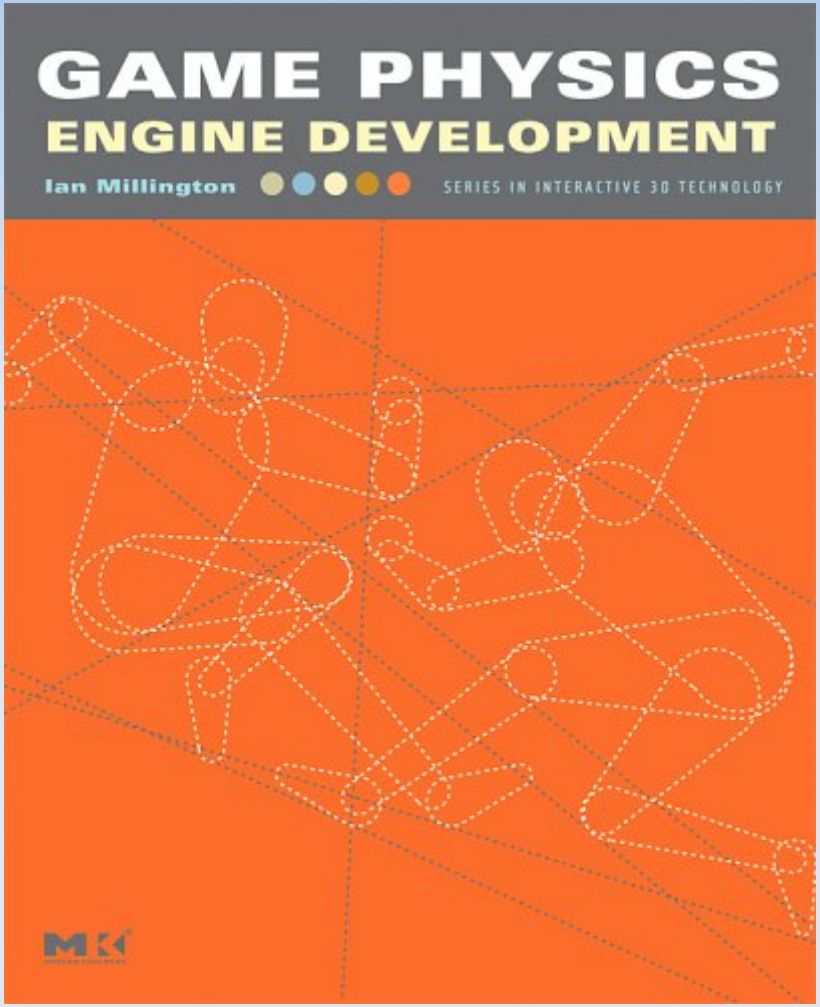
References

Real-Time Collision Detection
Christer Ericson

References



Game Physics
David Eberly



Game Physics Engine Development
Ian Millington

Online resources

- Erin Catto
  - http://www.gphysics.com/
- Glenn Fiedler
  - http://www.gaffer.org/game-physics
- Wikipedia

Existing 2D physics engines

- Box2D
  - http://www.box2d.org/
- Chipmunk
  - http://wiki.slembcke.net/main/published/Chipmunk
- Farseer
  - http://www.codeplex.com/FarseerPhysics
- Large Polygon Collider
  - http://www.draknek.org/physics/ (soon)

Existing 3D physics engines

- Bullet
  - http://www.bulletphysics.com/
- Open Dynamics Engine
  - http://www.ode.org/
- Havok
  - http://www.havok.com/tryhavok
- Large Polygon Collider
  - http://www.draknek.org/physics/ (soon)

Questions?