Physics and Collision Detection

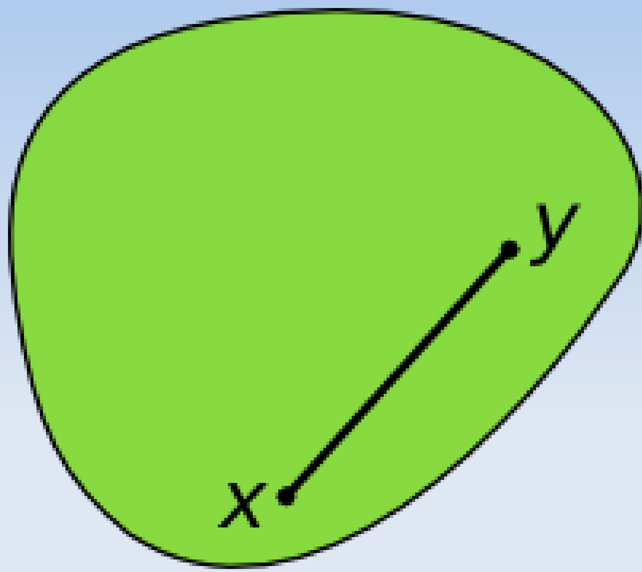
Alan Hazelden

http://www.draknek.org/

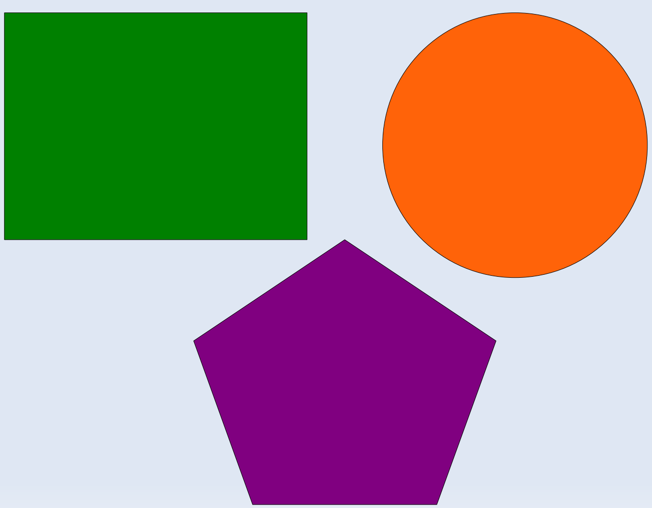
Required knowledge

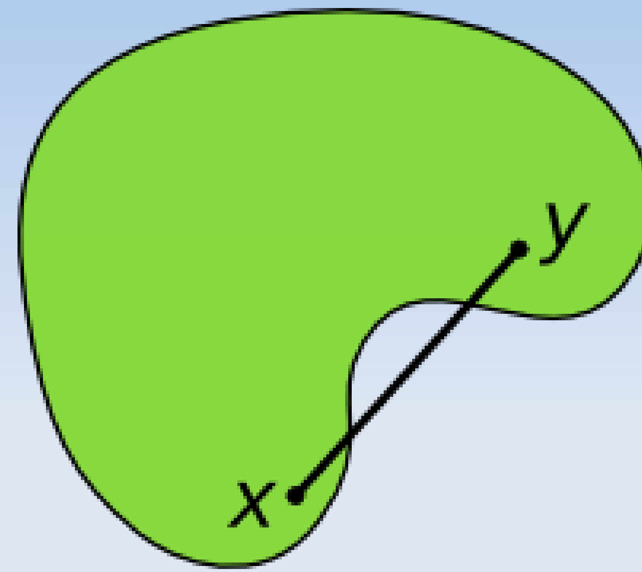
- Basic physical concepts
  - Velocity
  - Acceleration
  - Mass
  - Forces
- Basic geometrical concepts
  - Vectors
  - Matrices

Definitions

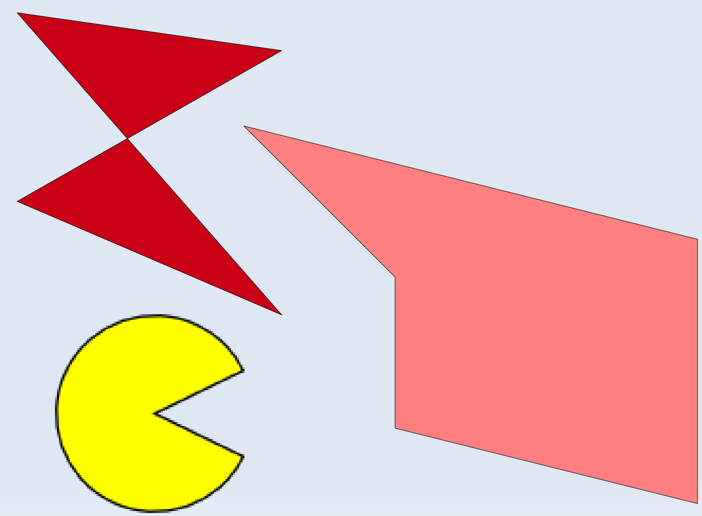


Convex





Concave



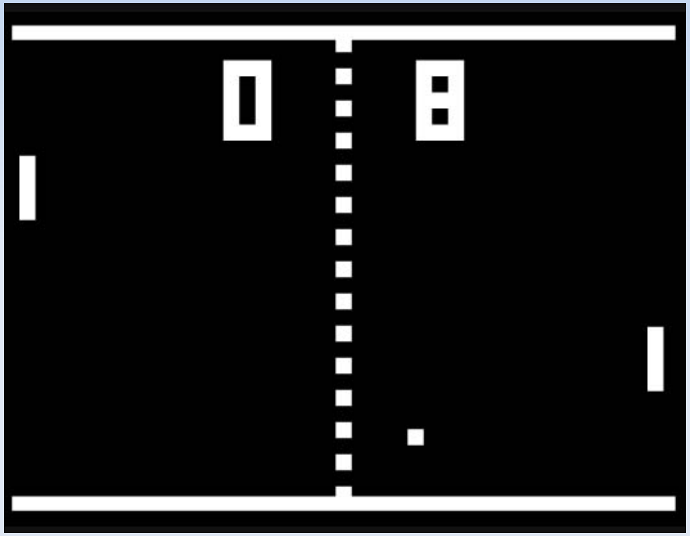
Definitions

- Impulse
  - An instantaneous force
- Moment
  - Component of a force that affects rotation

- Rotation ≠ Orientation
- Rotation = Angular velocity

Games and physics

- How do games use physics?



Games and physics

- How do games use physics?
- Preventing interpenetration of objects

- One good approach:
  - Detect that objects are colliding
  - Do something about it

Collision detection

- Massively important
- Needs to be efficient
- Needs to be accurate
  - But only to a certain extent

Simplification

- We don't perform collision detection on the rendering geometry.

2D primitives

- Circle
- Line
- Rectangle
- Capsule
- Convex polygon

3D primitives

- Sphere
- Line
- Cuboid (aka box)
- Cylinder
- Capsule
- Convex polyhedra

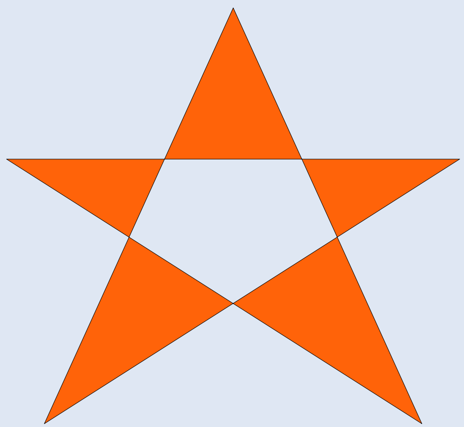
Containment

- Does this shape contain this point?

- Circles: test distance from centre
- Axis-aligned rectangles: check x/y coordinates
- Oriented rectangle: first convert point to local coordinate system

Containment within a polygon

- Basic idea: draw an infinite line from the point in any direction
  - Count the number of times it crosses an edge
  - Two different ways of counting
    - Give different results for some polygons

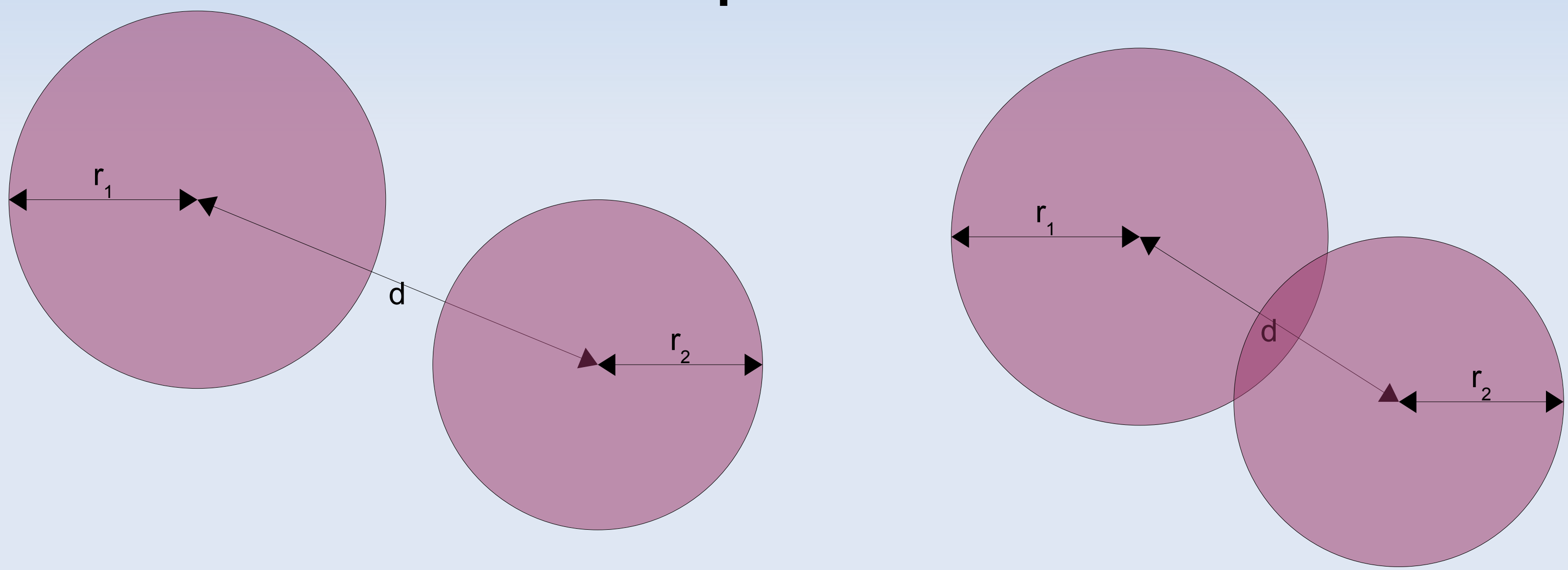


Convex polygons

- Can be represented as the intersection of a set of half-spaces
- To check containment, check against each half-space in turn
  - If outside any, then outside the convex polygon
  - If inside all, then inside the convex polygon

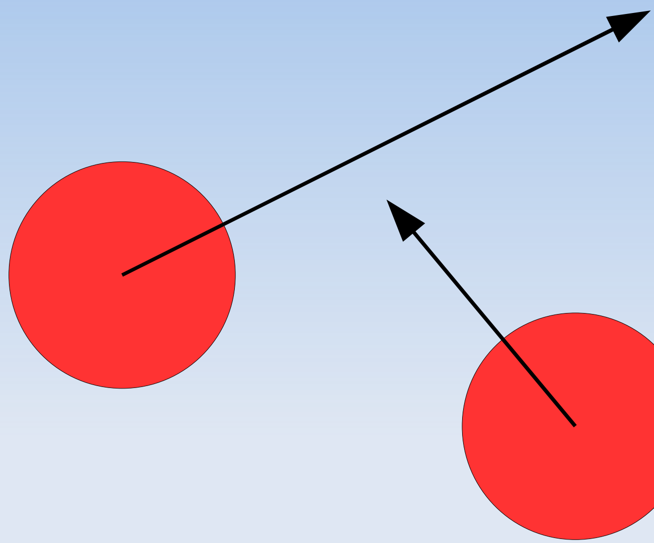
Collision detection

- Are these shapes overlapping?
- Easiest with circles/spheres:

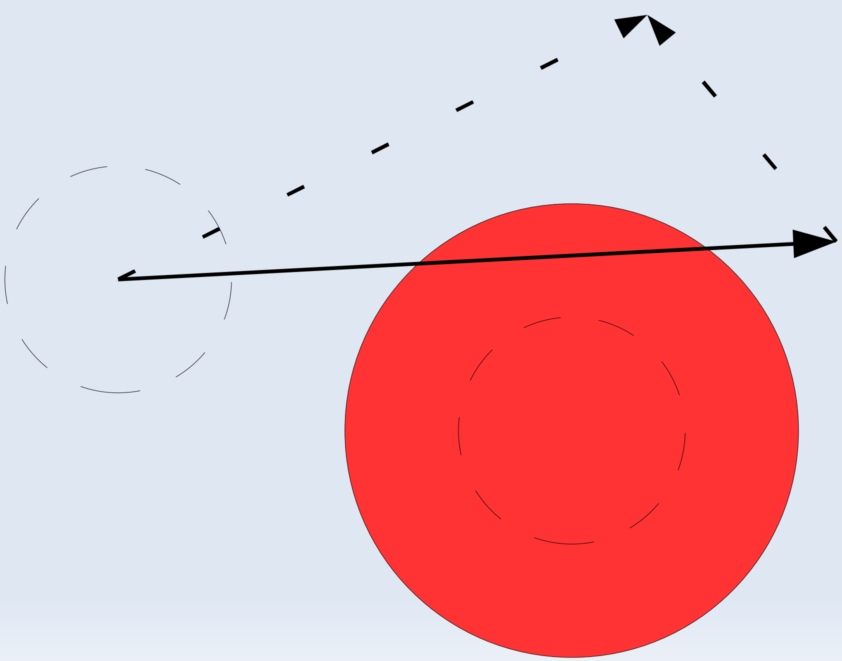


  - Colliding if $r_1 + r_2 > d$
  - Note: Equivalent to containment of a point within a circle of radius $r_1 + r_2$

Collision of moving circles

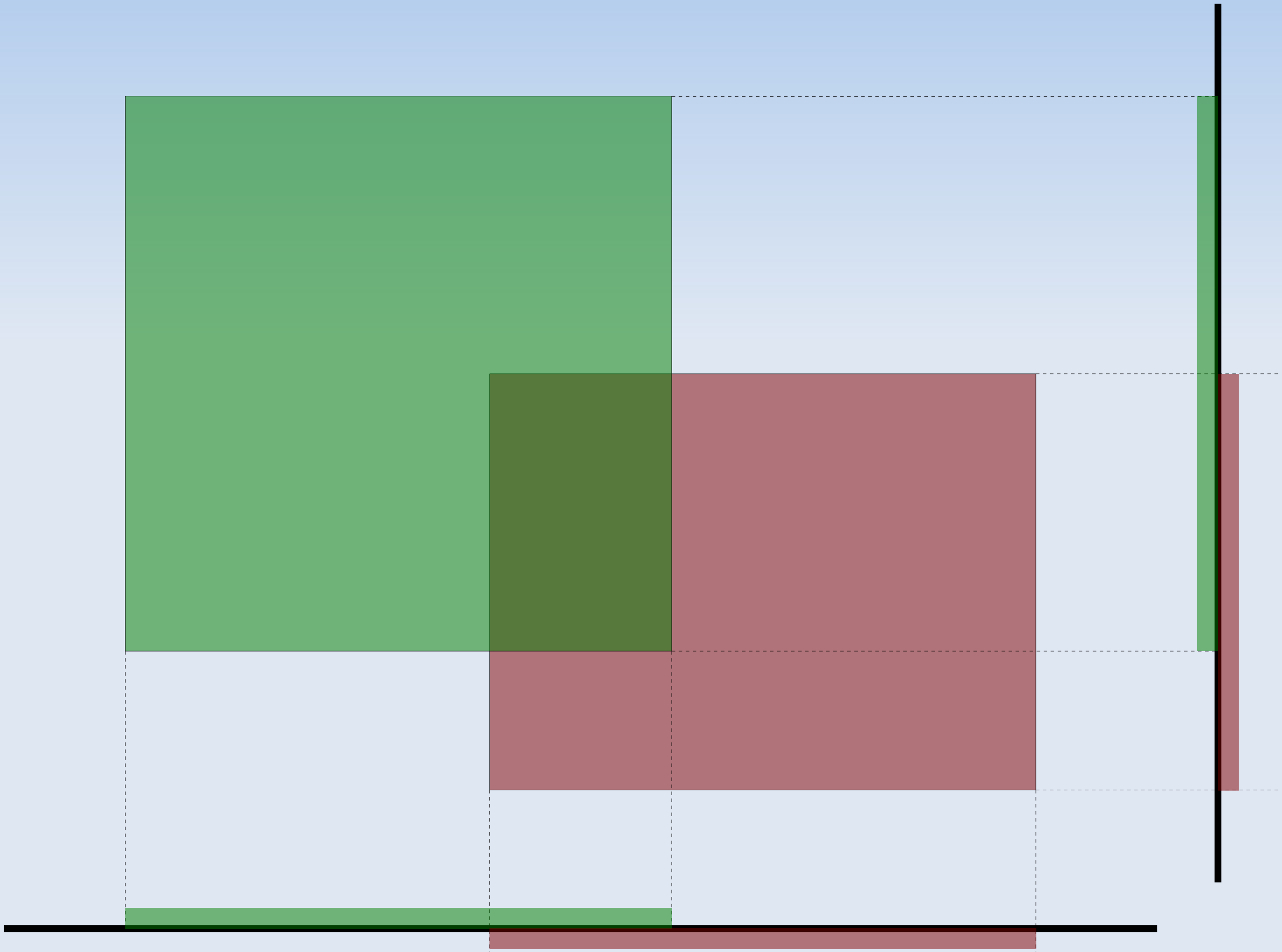


- We can represent this as the intersection of a line and a circle

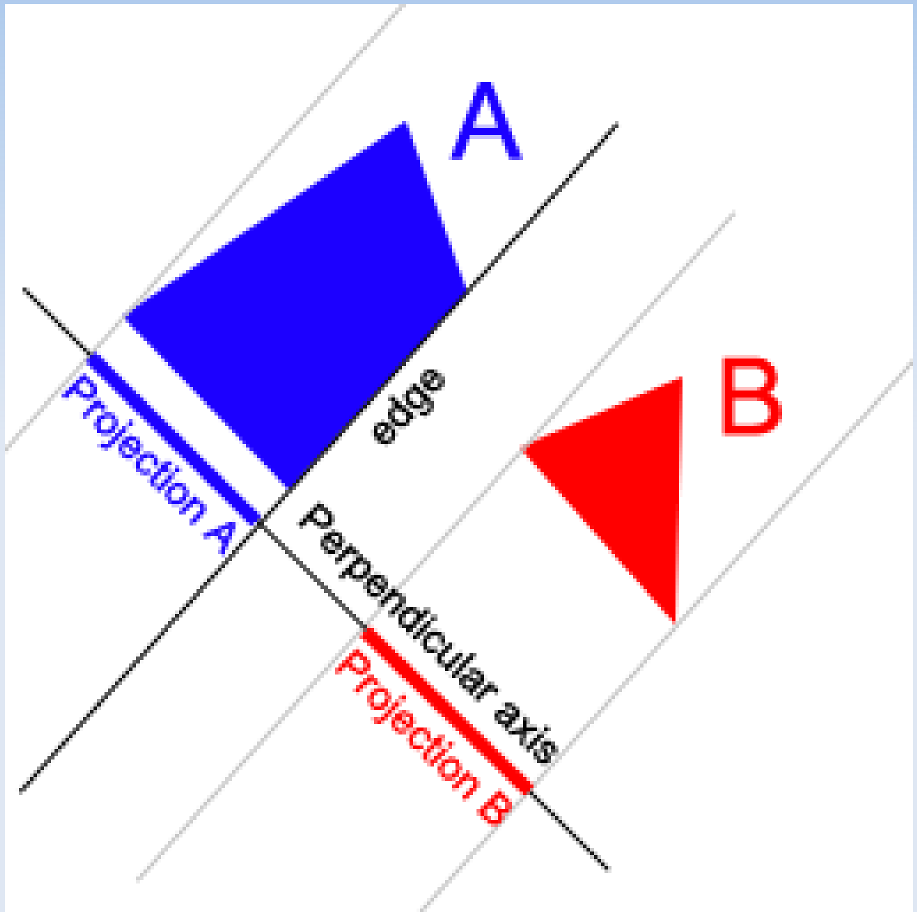


- We can even get the time of collision!
  - Note: usually we don't need this

Axis-aligned boxes



Separating Axis Theorem



- If separated along some axis, not colliding
- If overlapping along all possible axes, colliding

Separating Axis Theorem

- Key observation: can enumerate the possible separating axes
- For axis-aligned 2D boxes, only two (x and y)
- For oriented 2D boxes, only four (two per box)
- What about more complicated shapes?

Collision culling

- Brute force collision testing would take $O(n^2)$ comparisons
- We can rule some collisions out very quickly
  - Bounding boxes
  - Exploiting temporal coherence
    - If we have a separating axis for two objects, it is likely to still be a separating axis in the next frame

Broad-phase collision detection

- Exploits spatial coherence
  - Whatever that means
- Regular grid
- Quadtree/Octree
- BSP tree (binary space partitioning)
- Hierarchy of bounding shapes

Collision Resolution

- Once we've found a collision, what do we do?
- Information needed:
  - Contact normal
  - Contact point
  - Penetration distance
- This is all found by the Separating Axis test!

Collisions without rotation

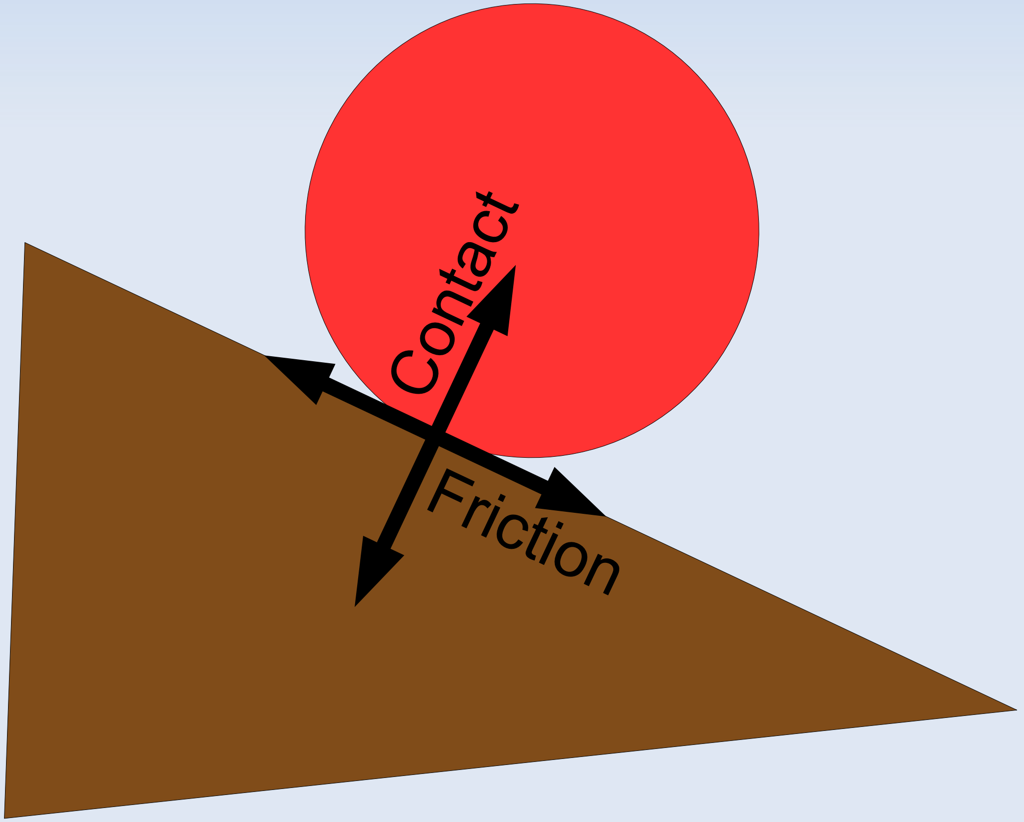
- Impulse applied at the instant of collision
  - In the direction of the collision normal
- Change in velocity determined by:
  - Coefficient of restitution
  - Conservation of momentum
- Better way: calculate magnitude of the impulse
  - Only use coefficient of restitution
  - Conservation of momentum automatically satisfied by balanced impulses

Collisions with rotation

- Impulse is applied at the point of contact
  - This will affect rotation
- Also: rotation affects approach speed
  - We want the approach speed of the contact points
- How does an impulse affect rotation?
  - Moment of inertia

Friction

- Perpendicular to the contact normal
- Opposes motion
- $F = \mu R$



Friction

- Friction is a force, not an impulse
  - But I treat it like one anyway
  - Applied at the end of the frame, after contact force

Penetration

- Overlapping objects
- If we resolved contact forces at the instant of collision, this shouldn't happen
  - But we only ever apply these at the end of a frame
- Need to nudge objects so they stop colliding
  - But there could be many objects in a group
  - One approach: keep collisions in a list
    - Sort list by penetration distance

General overview

- Every frame:
  - All objects are moved simultaneously
  - The collision detection system is run
    - Every pair of colliding objects is detected and stored
  - The collision resolver is run
    - Velocities are updated
    - Penetration is removed
  - All objects are drawn at their new positions

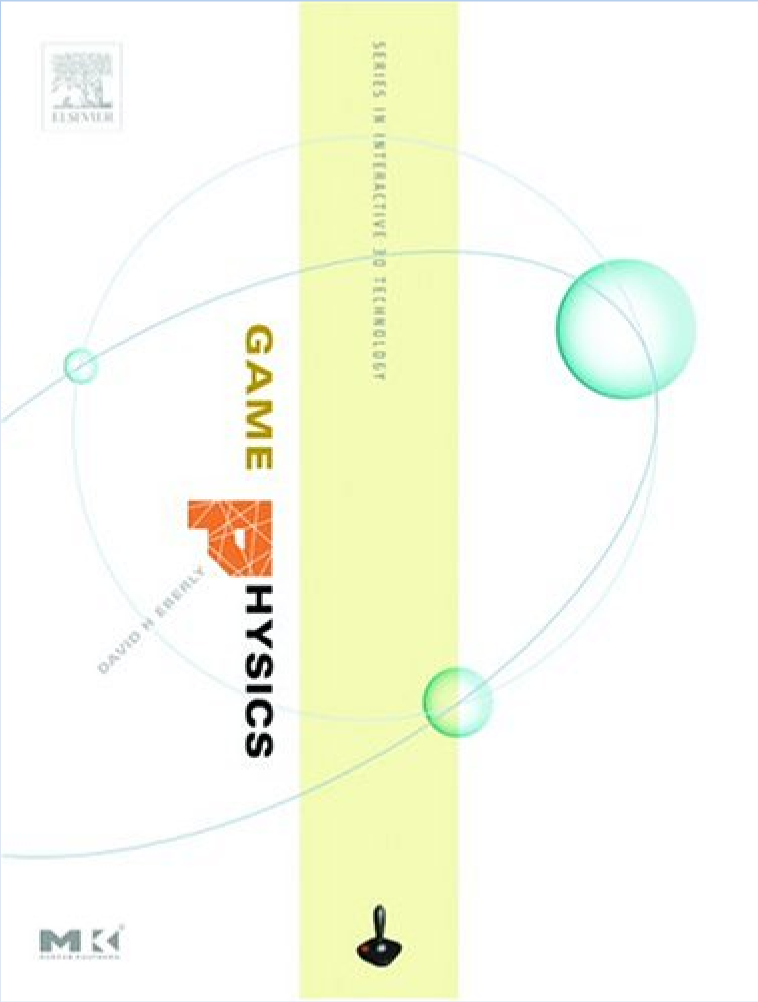
Movement of objects

- First-order Euler integration
  - ```
    x += v * t;
    v += a * t;
    ```
  - Not good
- Second-order Euler integration
  - Based on SUVAT equations
  - ```
    x += v * t + 0.5 * a * t * t;
    v += a * t;
    ```
  - Works, assuming constant acceleration
    - Unfortunately, springs and other constraints don't fit this

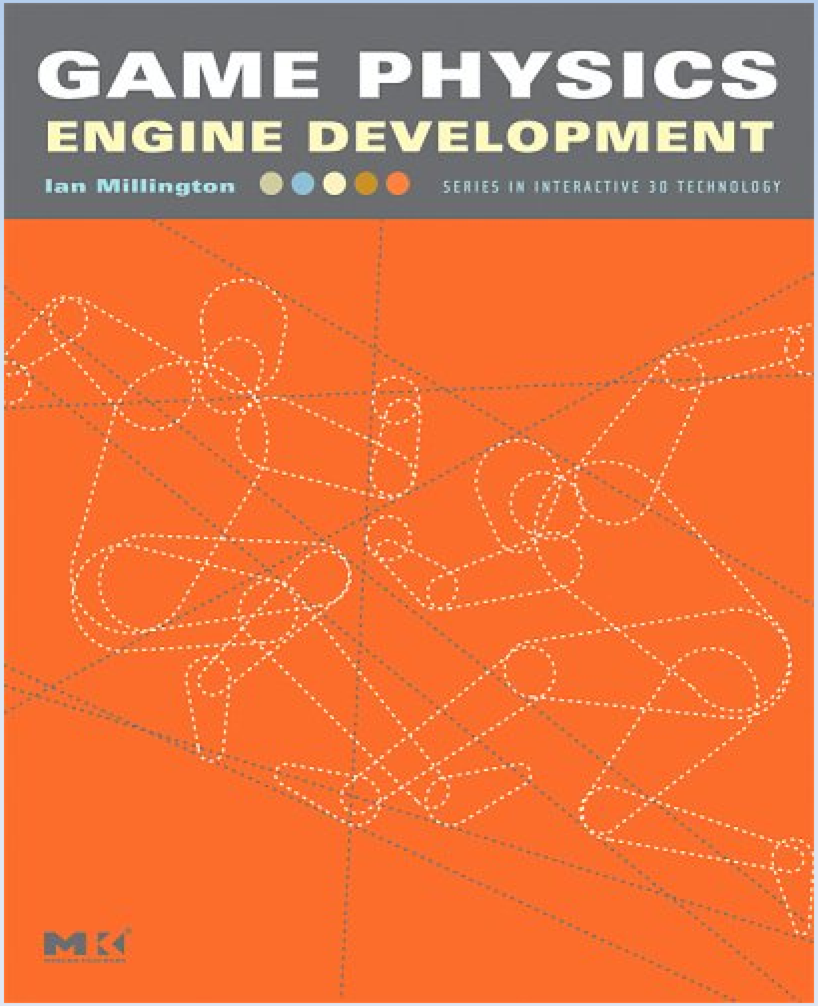
References

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 (C++)
  - http://www.box2d.org/
- Chipmunk (C)
  - http://wiki.slembcke.net/main/published/Chipmunk
- Farseer (XNA)
  - http://www.codeplex.com/FarseerPhysics
- My third year project (C++ and WGD-Lib)
  - http://www.draknek.org/physics/

Existing 3D physics engines

- Bullet
  - http://www.bulletphysics.com/
- Open Dynamics Engine
  - http://www.ode.org/
- Havok
  - http://www.havok.com/tryhavok

Questions?