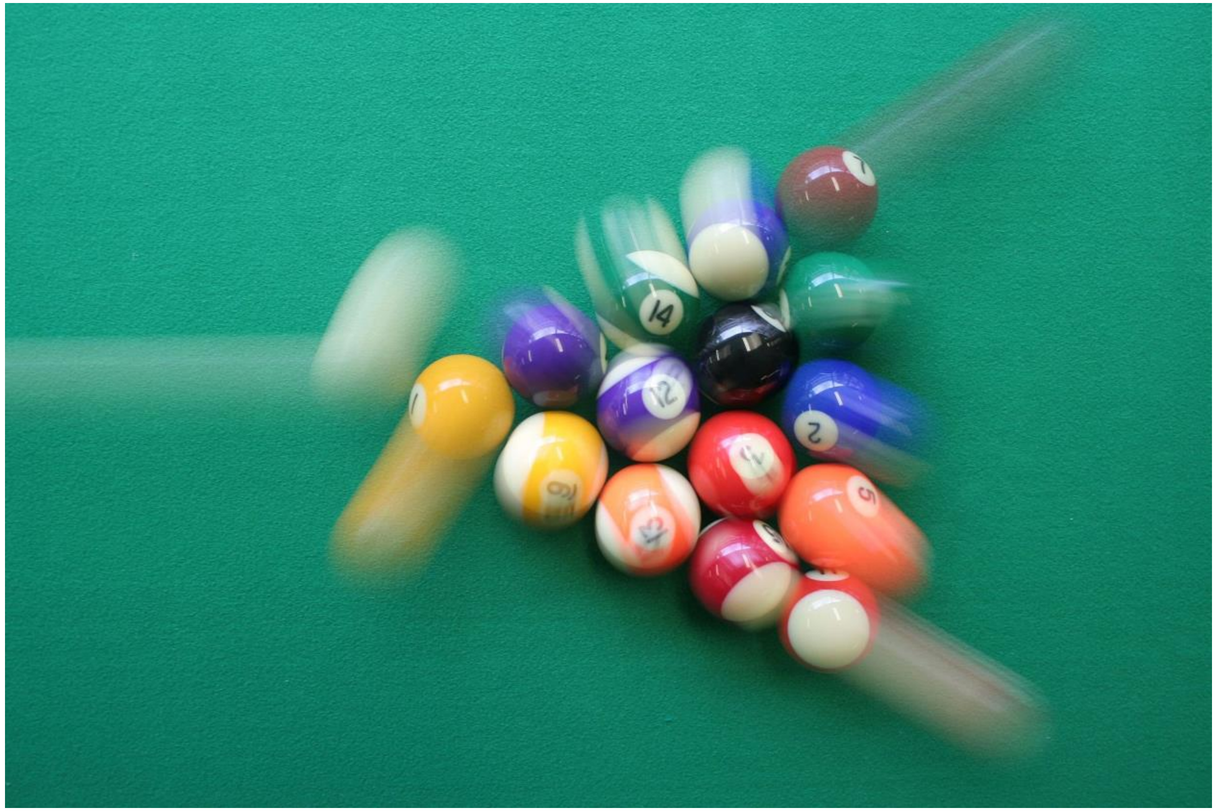
CPSC 427
Video Game Programming

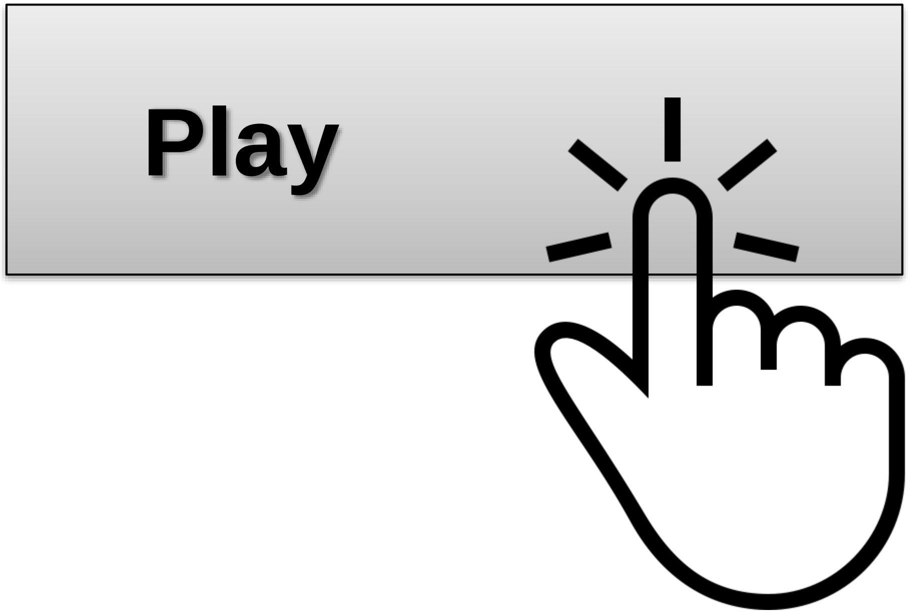
Collisions (and a bit of physics)



Helge Rhodin

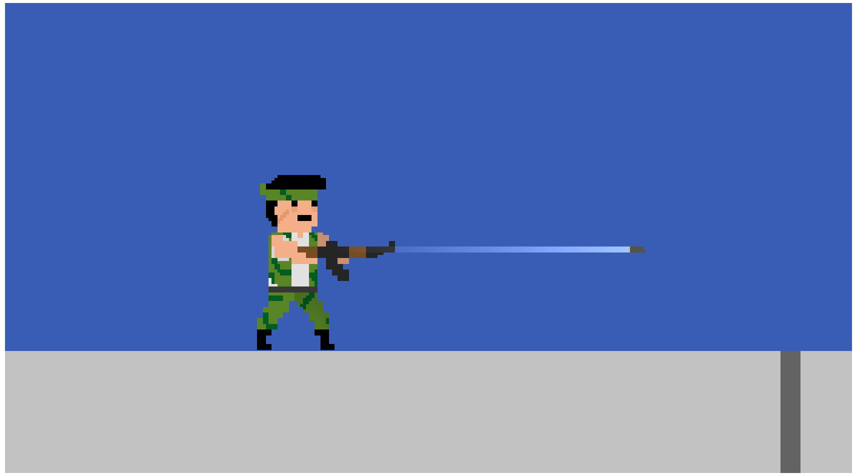
Motivation: Object selection

- *Point inside object boundary?*



Motivation: Bullet trajectories

- ***Line-object or point-object intersection?***

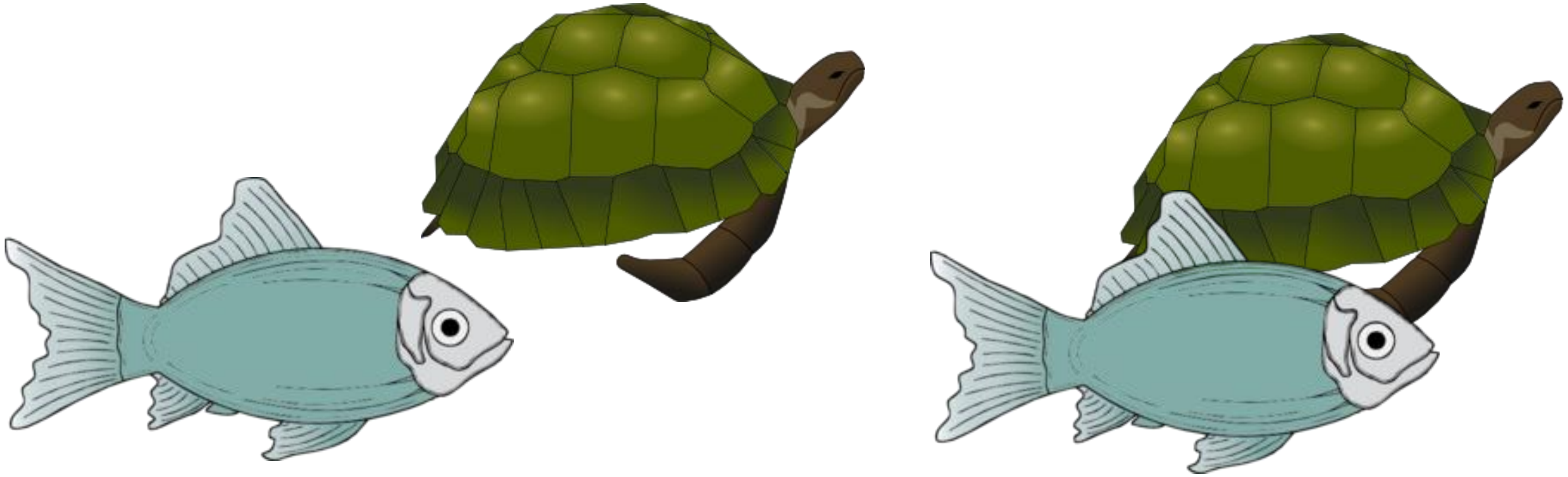


https://forum.unity.com/threads/2d-platformer-shooting.365971/

Motivation: Collision

- ***Prevent object penetration***

- ***How?***



Collision Configurations?

To detect collisions between polygons it is enough to test if their edges intersect

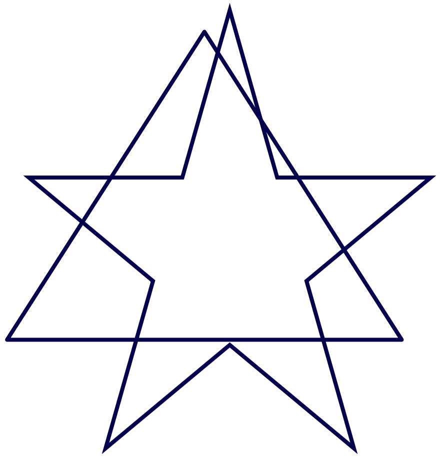
A. True

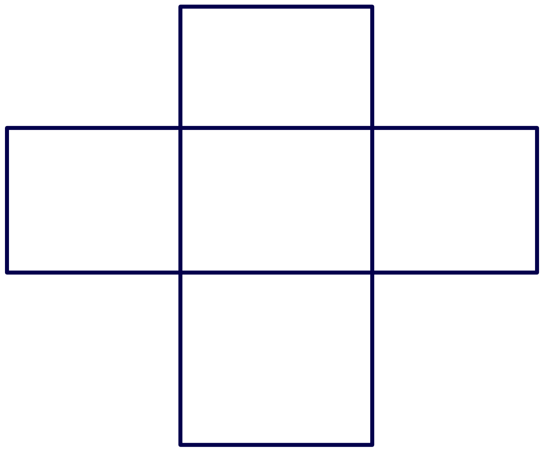
B. False

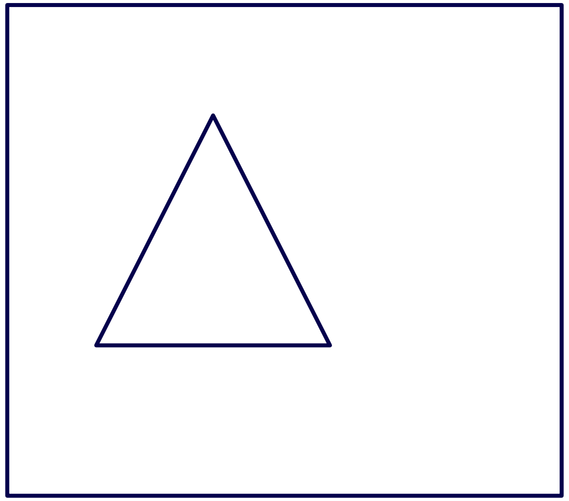
Collision Configurations?

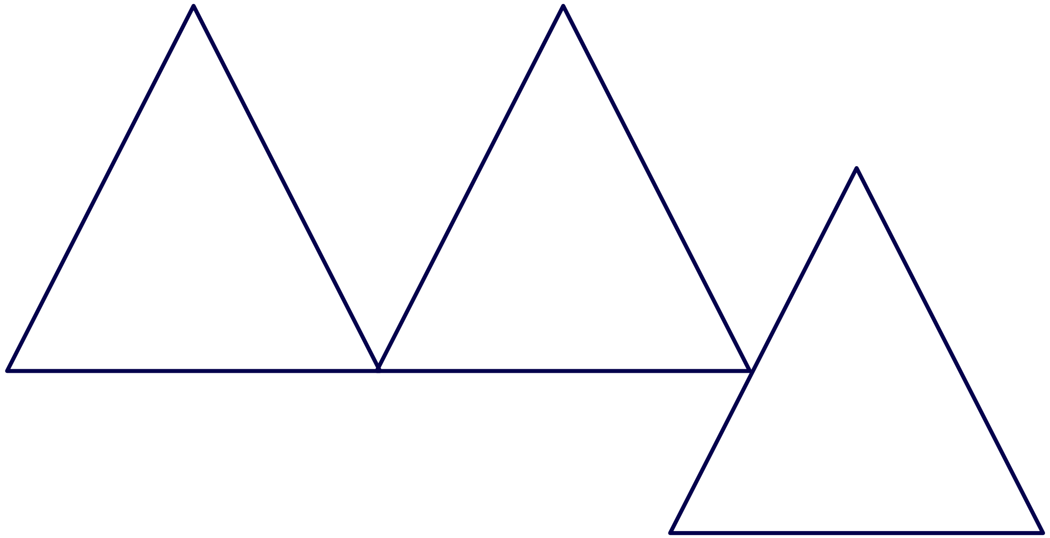
- Segment/Segment Intersection
  - *Point **on** Segment*

- Polygon inside polygon









Inside Test?

- How to test if one poly is inside another?

- Use inside test for point(s)

- How?
  - *Convex Polygon*
    - Same side WRT to line (all sides)
  - *Non-Convex*
    - Subdivide= triangulate
    - How?
    - Shoot rays (beware of corners and special cases)

Resources

http://www.realtimerendering.com/intersections.html

Curves

Mathematical representations:

- Explicit functions:
- Parametric functions
- Implicit functions

Explicit functions

- y = f(x)
- E.g. y = a x + b
- Single y value for each x
- Useful for?
  - *Terrain*
  - *“height field” geometry*

Parametric Functions

- 2D: x and y are functions of a parameter value t
- 3D: x, y, and z are functions of a parameter value t

$$C(t) := \begin{pmatrix} P_y^0 \\ P_x^0 \end{pmatrix} t + \begin{pmatrix} P_y^1 \\ P_x^1 \end{pmatrix} (1-t)$$

Line (segment)

$$C(t) := \begin{pmatrix} \cos t \\ \sin t \end{pmatrix}$$

Circle (arc)

- Depends on parameter range t1 < t < t2

Implicit Function

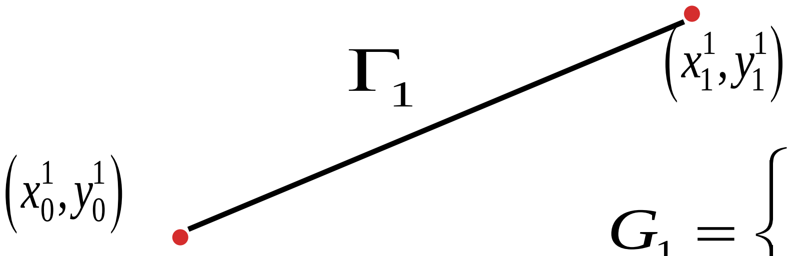
- Curve (2D) or Surface (3D) defined by zero set (roots) of function
- E.g:

$$S(x, y): x^2 + y^2 - 1 = 0$$

$$S(x, y, z): x^2 + y^2 + z^2 - 1 = 0$$

Lines & Segments

Segment Γ_1 *from* $P_0 = (x_0^1, y_0^1)$ *to* $P_1 = (x_1^1, y_1^1)$



$$G_1 = \begin{cases} x^1(t) = x_0^1 + (x_1^1 - x_0^1)t \\ y^1(t) = y_0^1 + (y_1^1 - y_0^1)t \end{cases} \quad t \in [0,1]$$

Find the line through $P_0 = (x_0^1, y_0^1)$ *and* $P_1 = (x_1^1, y_1^1)$

- Parametric $G_1(t), t \in (-\infty, -\infty)$
- Implicit $Ax+By+C=0$
  - *Solve 2 equations in 2 unknowns (set* $A^2+B^2=1$*)*

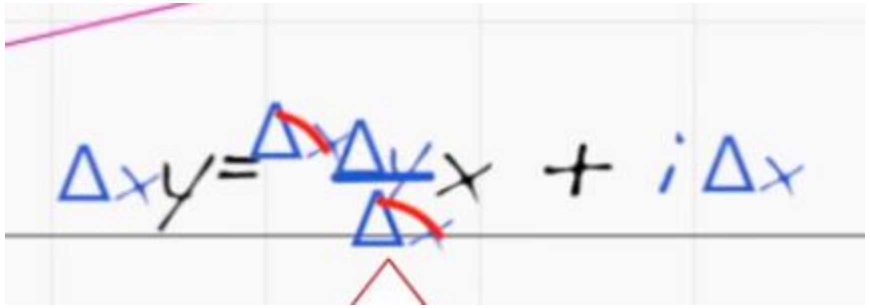
Implicit Line

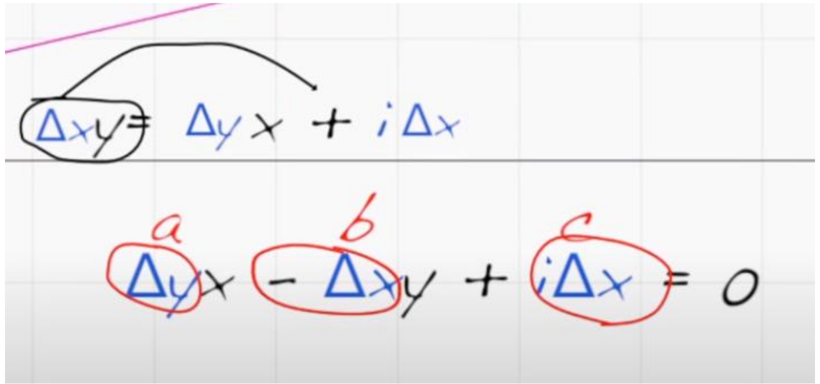
Explicit: $y = m\,x + b$ **Implicit:** $Ax + By + C = 0$

$y = dy/dx\ x + b$
$dx\ y = dy\ x + dx\ b$
$0 = dy\ x - dx\ y + dx\ b$

$\Rightarrow A = dy,\ B = -dx,\ C = dx\ b$

Example
$y = -1/3\ x + 0$
$dx = -3,\ dy = 1,$
$A = 1,\quad B = 3,\quad C = 0$
=> 1 x + 3 y = 0

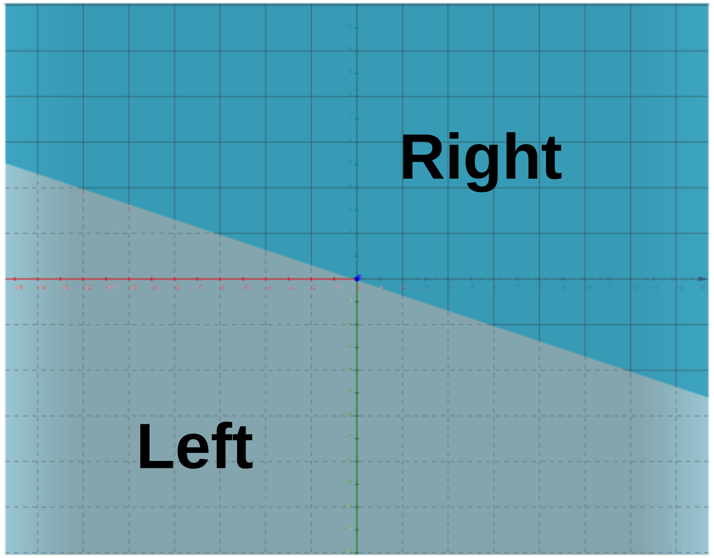


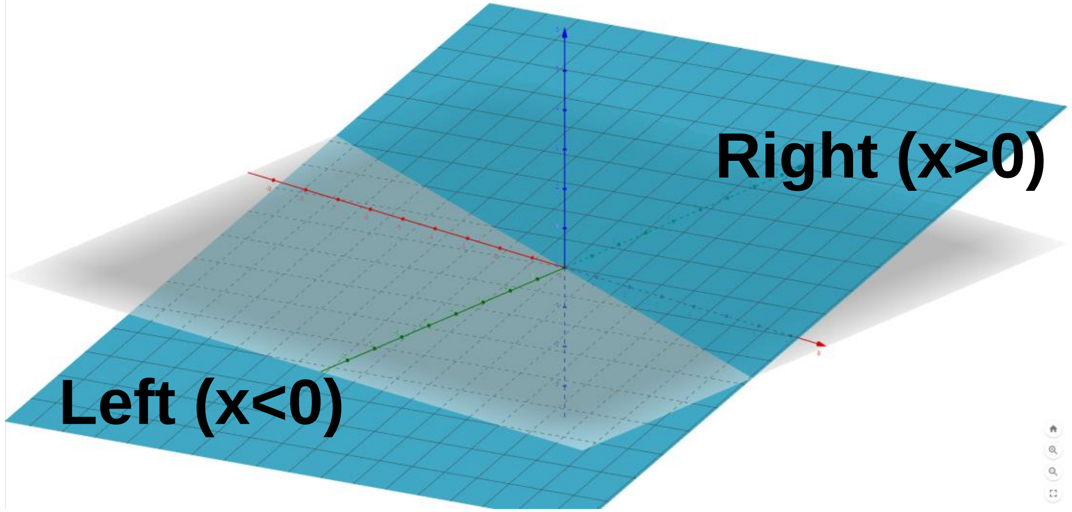


Implicit Line – left or right?

Implicit line in 2D <-> Explicit plane in 3D

$0.1\,x + 0.3\,y = 0$ <-> $f(x,y) = 0.1\,x + 0.3\,y$

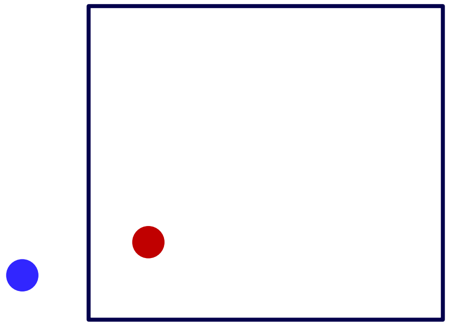




https://www.geogebra.org/3d

Point vs Line (Ray)

- Point $P=(P_x,P_y)$
- Use implicit equation to determine coincidence & side
  - *Implicit* $A x + B y + C = 0$
  - *Solve 2 equations in 2 unknowns (third equation: set* $A^2+B^2=1$*)*
  - *On:* $A P_x + B P_y + C=0$
  - *Use same orientation to get consistent left/right orientation for inside test for lines defining CONVEX polygon*
    - Same sign implies inside
    - *Eg.* ***ALL*** $AP_x + B P_y + C < 0$



Recap: Inside Test?

- How to test if one poly is inside another?
- Use inside test for point(s)

- How?
  - *Convex Polygon*
    - Same side WRT to line equation (all sides)
  - *Non-Convex*
    - Subdivide=triangulate
    - How?
    - Shoot rays (beware of corners and special cases)
    - Other ways?

Self-study:

Winding number algorithm

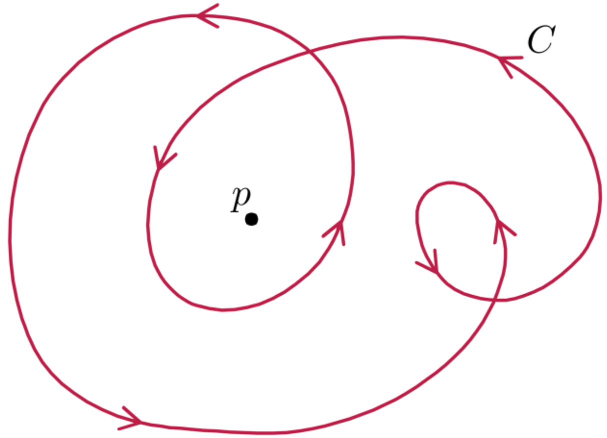
Point in polygon?

- If the winding number is non-zero

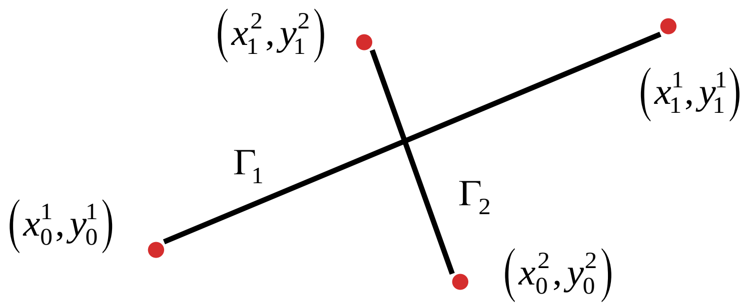
- How to compute the winding number?
  - http://geomalgorithms.com/a03-_inclusion.html

Winding number:

- the number of times that curve travels counterclockwise around the point
- negative if clockwise



Line-Line Intersection



$$G_1 = \begin{cases} x^1(t) = x_0^1 + (x_1^1 - x_0^1)t \\ y^1(t) = y_0^1 + (y_1^1 - y_0^1)t \end{cases} \quad t \in [0,1] \qquad G_2 = \begin{cases} x^2(r) = x_0^2 + (x_1^2 - x_0^2)r \\ y^2(r) = y_0^2 + (y_1^2 - y_0^2)r \end{cases} \quad r \in [0,1]$$

Intersection: *x* & *y* values equal in both representations - two linear equations in two unknowns (*r*,*t*)

$$x_0^1 + (x_1^1 - x_0^1)t = x_0^2 + (x_1^2 - x_0^2)r$$
$$y_0^1 + (y_1^1 - y_0^1)t = y_0^2 + (y_1^2 - y_0^2)r$$

Question: What is the meaning if the solution gives *r*,*t* < 0 or *r*,*t* >1 ?

Question: What is the meaning of $r,t < 0$ or $r,t > 1$?

A. They still collide
B. They do not collide
C. They may or may not collide – need more testing

Efficiency

- Naïve implementation
  - *Test each moving object against ALL other objects at each step*
  - *Horribly expensive*

- How to speed up?

Efficiency

- Naïve implementation
  - *Test each moving object against ALL other objects at each step*
  - *Horribly expensive*

- Speed up
  - *Bounding Volumes*
  - *Hierarchies*

Bounding volumes

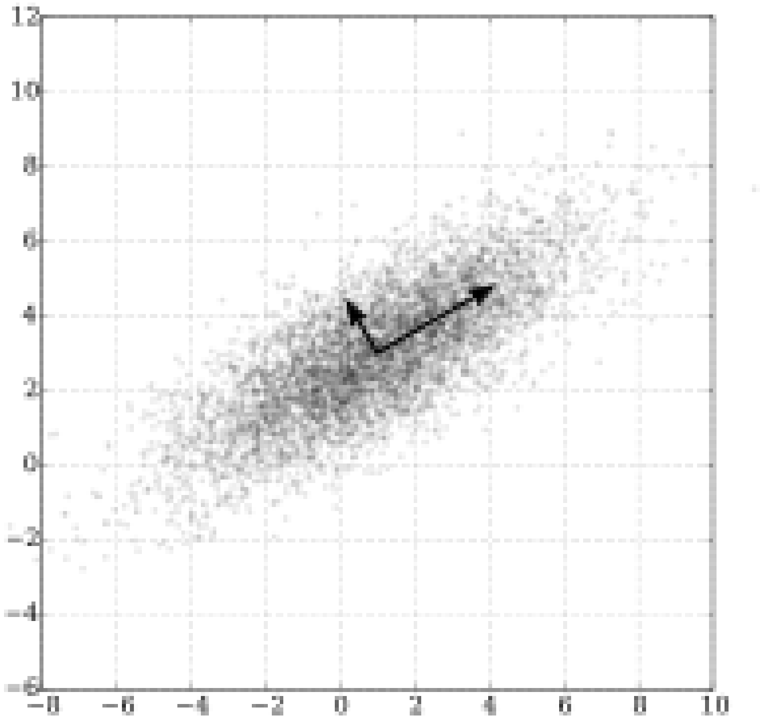
- Axis aligned bounding box (AABB)
  - *+ Trivial to compute*
  - *+ Quick to evaluate*
  - *- May be too big…*

- Tight bounding box
  - *- Harder to compute: Principal Component Analysis (PCA)*
  - *- Slightly slower to evaluate*
  - *- Compact*

Principle Component Analysis (PCA)

Derive the directions of maximum variance

$$\mathbf{w}_{(1)} = \underset{\Vert \mathbf{w} \Vert = 1}{\arg\max} \left\{ \sum_i \left(\mathbf{x}_{(i)} \cdot \mathbf{w} \right)^2 \right\}$$



Wikipedia

Bounding volumes

- Bounding circle
  - *A range of efficient (non-trivial) methods*

- Convex hull
  - *Gift wrapping & other methods…*

Bounding Volume Intersection

- Axis aligned bounding box (AABB)
  - $A.LO <= B.HI \;\&\&\; A.HI >= B.LO$ *(for both X and Y)*

    lower *higher*

- Circles
  - $||A.C - B.C|| < A.R + B.R$

    Center *Radius*

Moving objects

- Sweep – test intersections against before/after segment
  - *Avoid "jumping through" objects*
  - *How to do efficiently?*

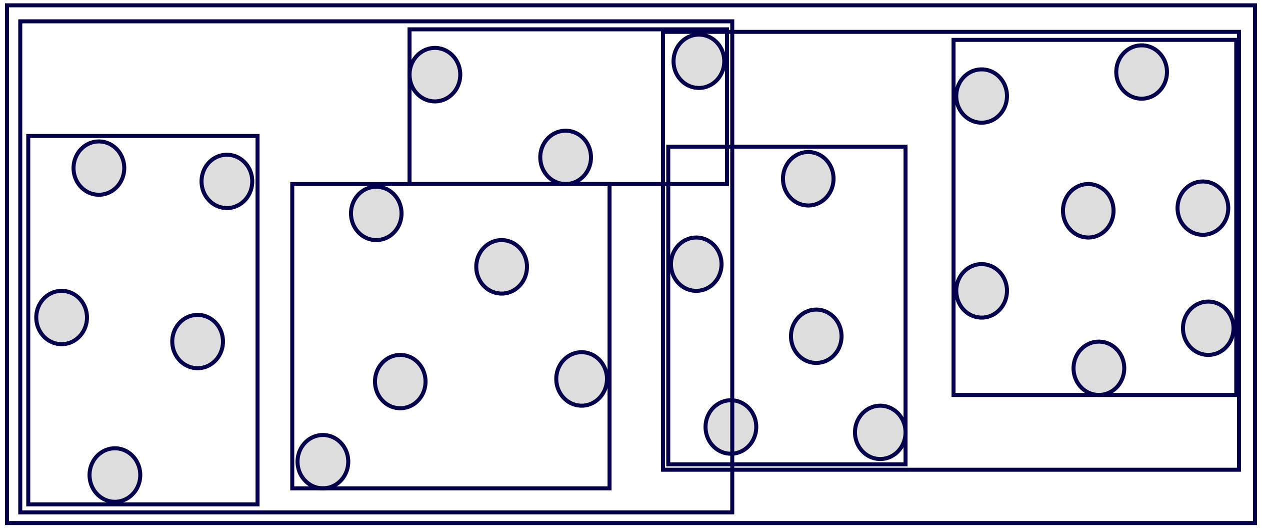
- Boxes?

- Spheres?

Hierarchical Bounding Volumes

Bound Bounding Volumes:

- Use (hierarchical) bounding volumes for groups of objects

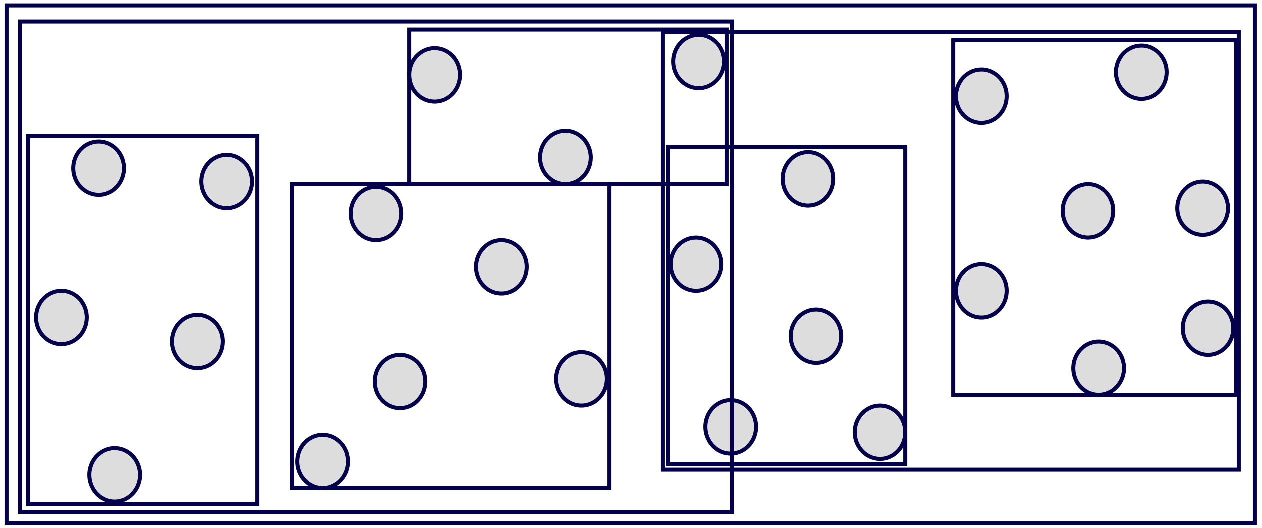


- How to group boxes?
  - *Closest*
  - *Most jointly compact (how?)*

Hierarchical Bounding Volumes

Bound Bounding Volumes:

- Use (hierarchical) bounding volumes for groups of objects



- Challenge: dynamic data…
  - *Need to update hierarchy efficiently*

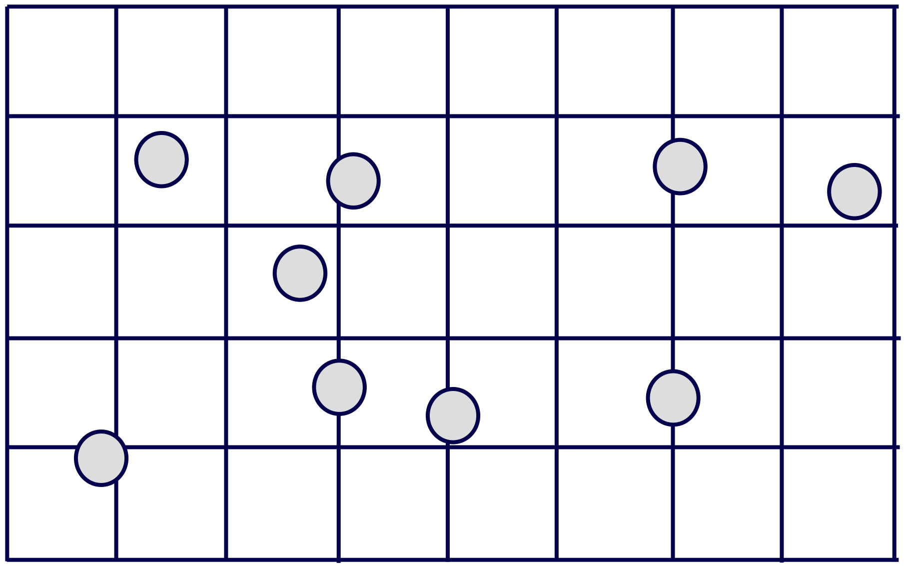
Spatial Subdivision DATA STRUCTURES

- Subdivide space (bounding box of the “world”)
- Hierarchical
  - *Subdivide each sub-space (or only non-empty sub-spaces)*
- Lots of methods
  - *Grid, Octree, k-D tree, (BSP tree)*

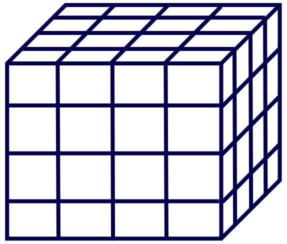
Regular Grid

Subdivide space into rectangular grid:

- Associate every object with the cell(s) that it overlaps with
- Test collisions only if cells overlap



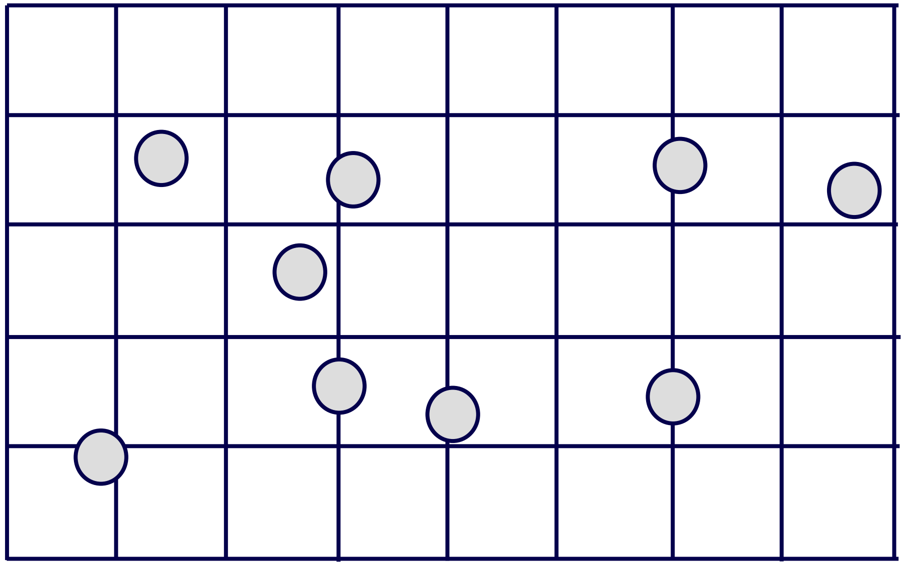
In 3D: regular grid of cubes (voxels):



Creating a Regular Grid

Steps:

- Find bounding box of scene
- Choose grid resolution in x, y, z
- Insert objects
- Objects that overlap multiple cells get referenced by all cells they overlap



Regular Grid Discussion

Advantages?

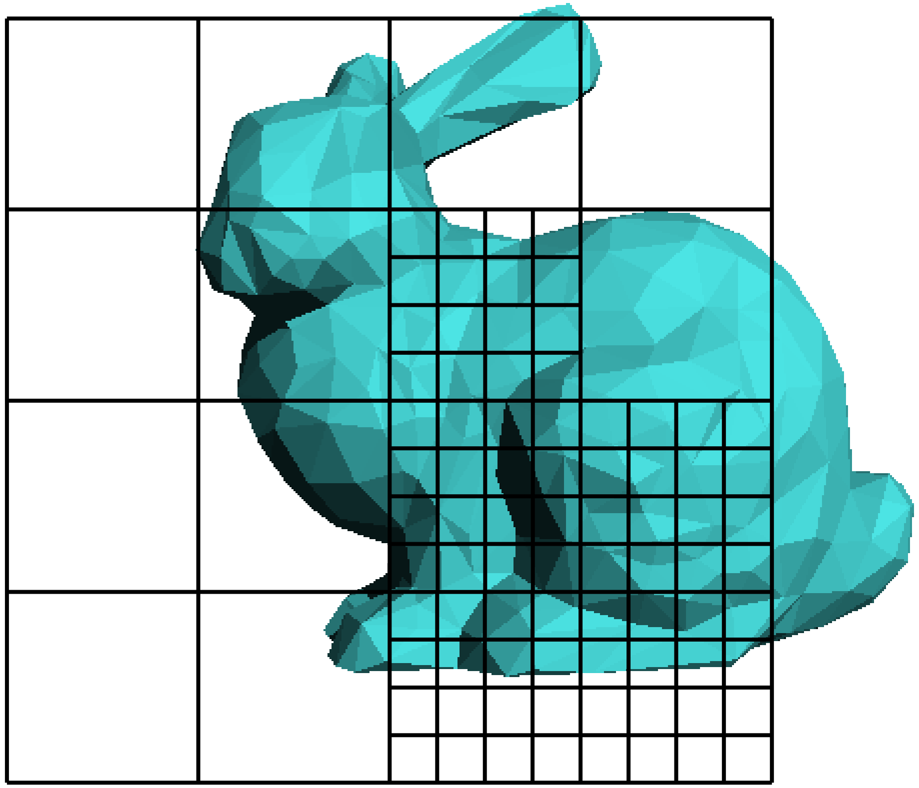
- Easy to construct
- Easy to traverse

Disadvantages?

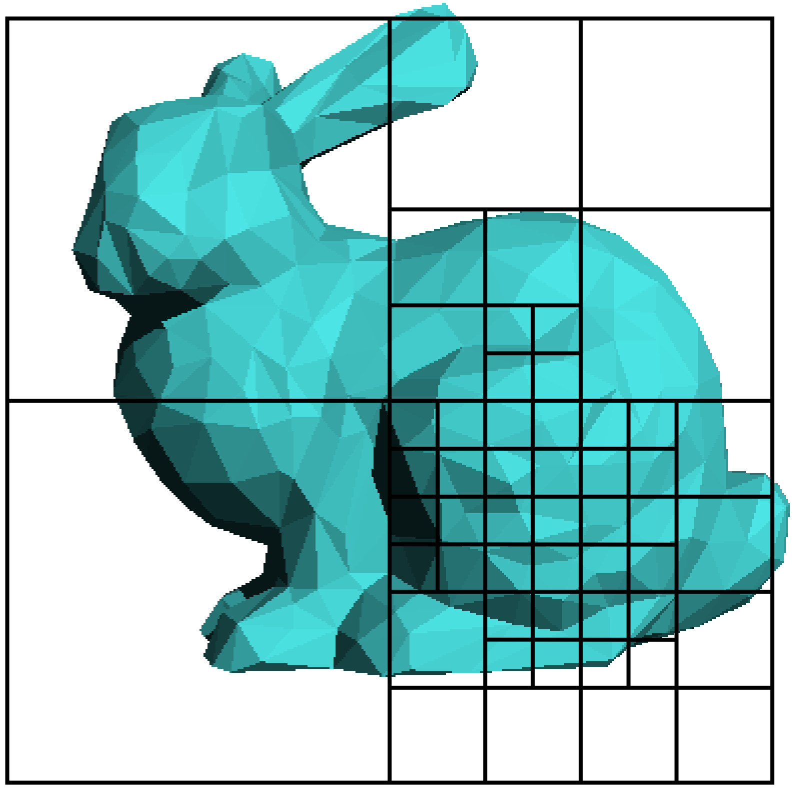
- May be only sparsely filled
- Geometry may still be clumped

Adaptive Grids

- **Subdivide until each cell contains no more than *n* elements, or maximum depth *d* is reached**



Nested Grids



Octree/(Quadtree)

- This slide is curtsey of Fredo Durand at MIT

Physics

Physics-Based Simulation

- **Movement governed by forces**
- **Simple**
  - ***Independent particles***
- **Complex**
  - ***Correct collisions, stacking, sliding 3D rigid bodies***
- **Many many simulators!**
  - ***PhysX (Unity, Unreal), Bullet, Open Dynamics Engine, MuJoCo, Havok, Box2D, Chipmunk, OpenSim, RBDL, Simulink (MATLAB), ADAMS, SD/FAST, DART etc…***

Examples

- Particle systems
  - *Fire, water, smoke, pebbles*
- Rigid-body simulation
  - *Blocks, robots, humans*
- Continuum systems
  - *Deformable solids*
  - *Fluids, cloth, hair*
- Group movement
  - *Flocks, crowds*

Simulation Basics

Simulation loop…

1. ***Equations of Motion***
   - sum forces & torques
   - solve for accelerations: $\vec{\boldsymbol{F}} = \boldsymbol{ma}$
2. ***Numerical integration***
   - update positions, velocities
3. ***Collision detection***
4. ***Collision resolution***

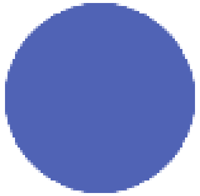
Basic Particle Simulation (first try)

Forces only $\vec{F} = ma$

$$d_t = t_{i+1} - t_i$$

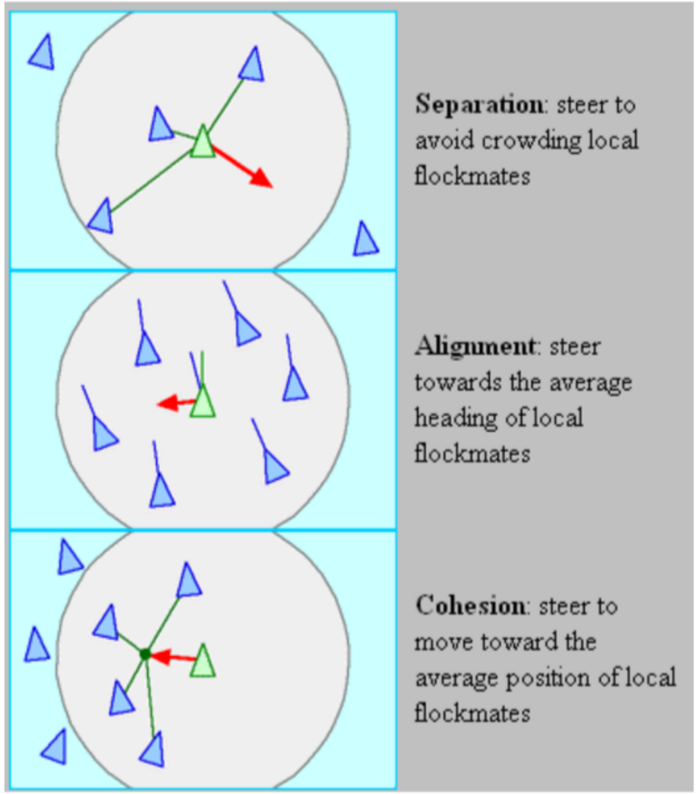
$$\vec{v}_{i+1} = \vec{v}(t_i) + (\vec{F}(t_i)/m)d_t$$

$$\vec{p}_{i+1} = \vec{p}(t_i) + \vec{v}(t_{i+1})d_t$$



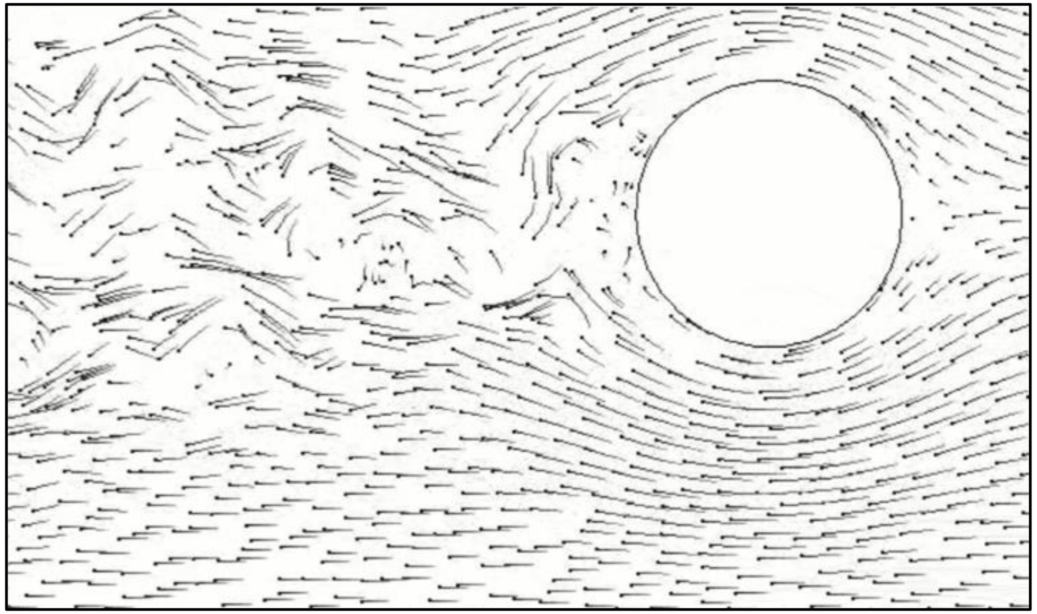
Proxy Forces

- **Behavior forces:**
  flocking birds, schooling fish, etc.
  ["Boids", Craig Reynolds, SIGGRAPH 1987]



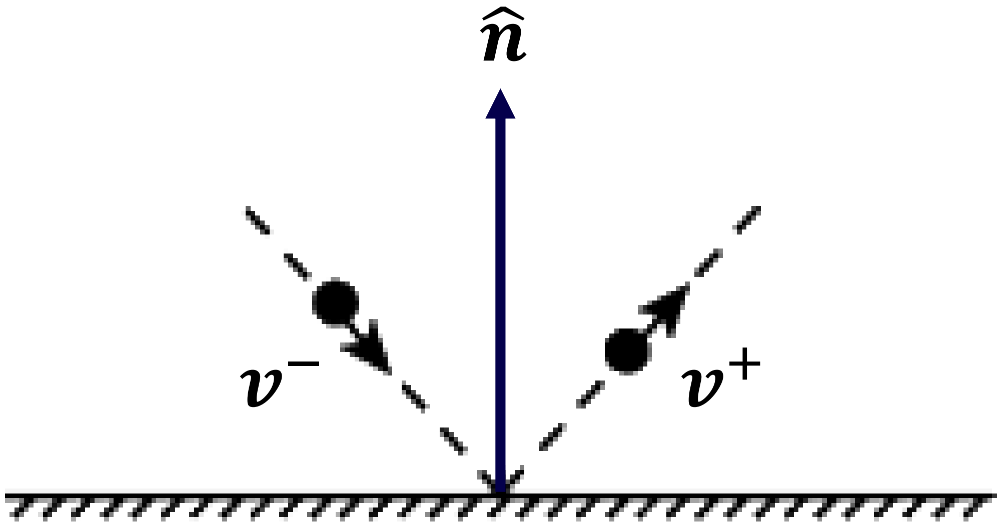
Courtesy of Craig W. Reynolds. Used with permission.

- **Fluids**
  ["Curl Noise for Procedural Fluid Flow" R. Bridson, J. Hourihan, M. Nordenstam, Proc. SIGGRAPH 2007]



Particle-Plane Collisions

- **More formally…**
  - ***Apply an impulse of magnitude j***
    - Inversely proportional to mass of particle
  - ***In direction of normal***



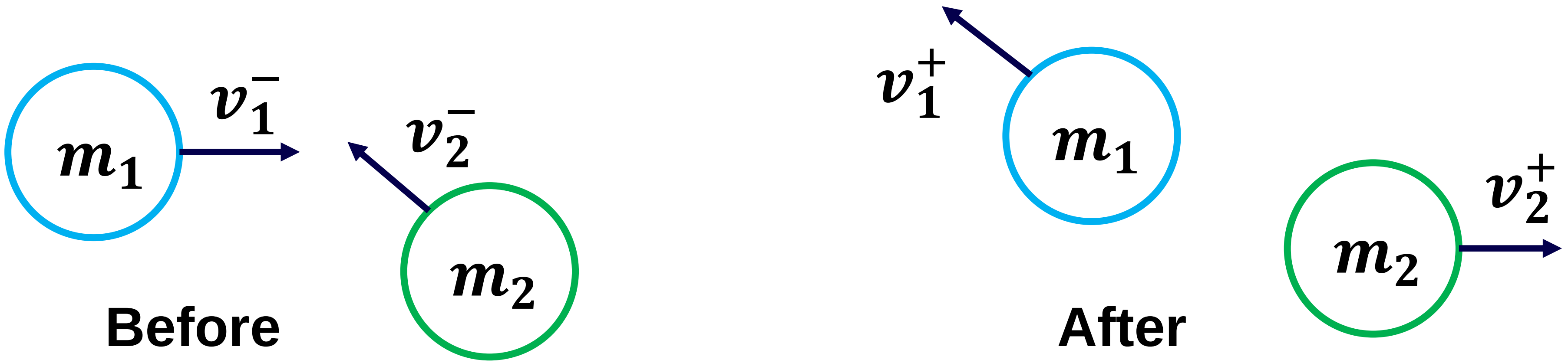
$$j = (1 + \epsilon)m$$

$$\vec{j} = j\,\hat{n}$$

$$v^+ = \frac{\vec{j}}{m} + v^-$$

Particle-Particle Collisions (radius=0)

- Particle-particle **frictionless elastic impulse response**



- Momentum is **preserved**

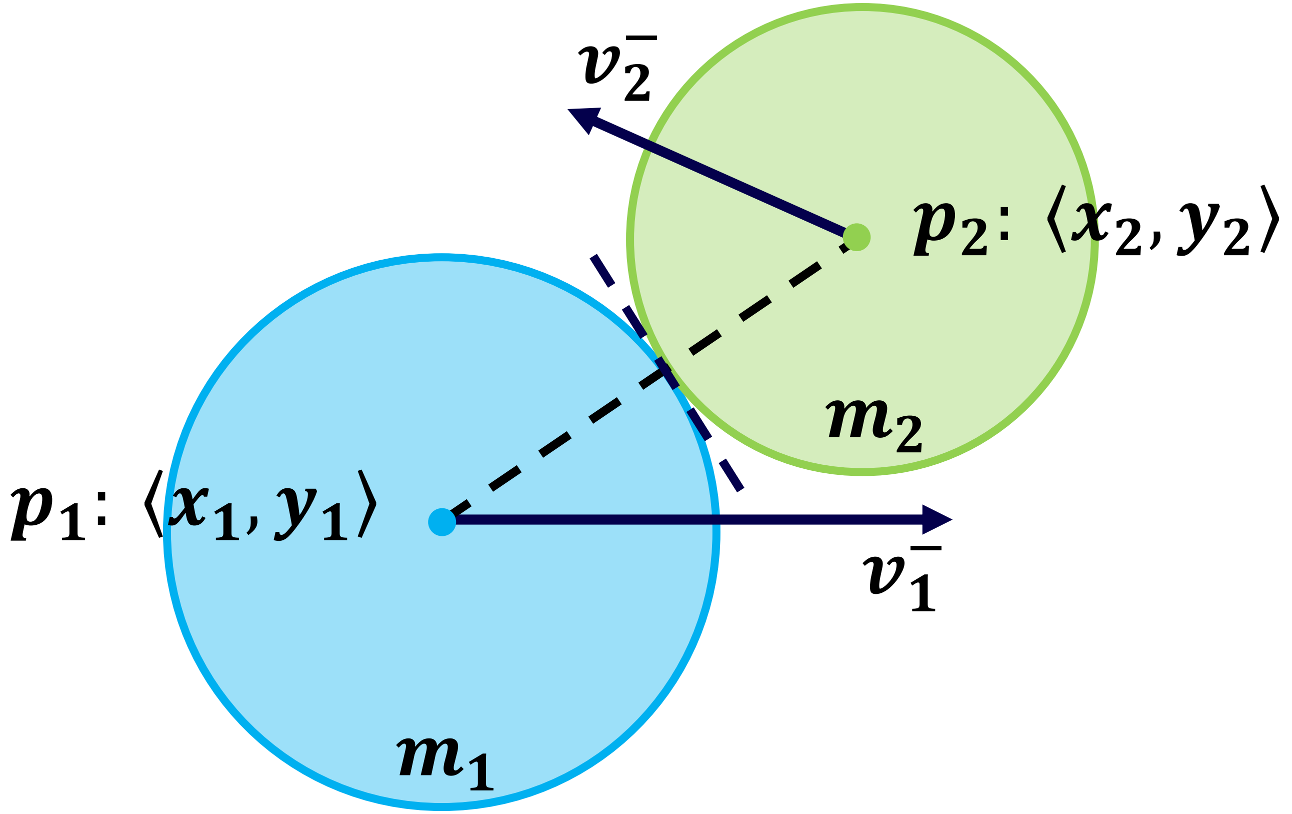
$$m_1 v_1^- + m_2 v_2^- = m_1 v_1^+ + m_2 v_2^+$$

- Kinetic energy is **preserved**

$$\tfrac{1}{2} m_1 {v_1^-}^2 + \tfrac{1}{2} m_2 {v_2^-}^2 = \tfrac{1}{2} m_1 {v_1^+}^2 + \tfrac{1}{2} m_2 {v_2^+}^2$$

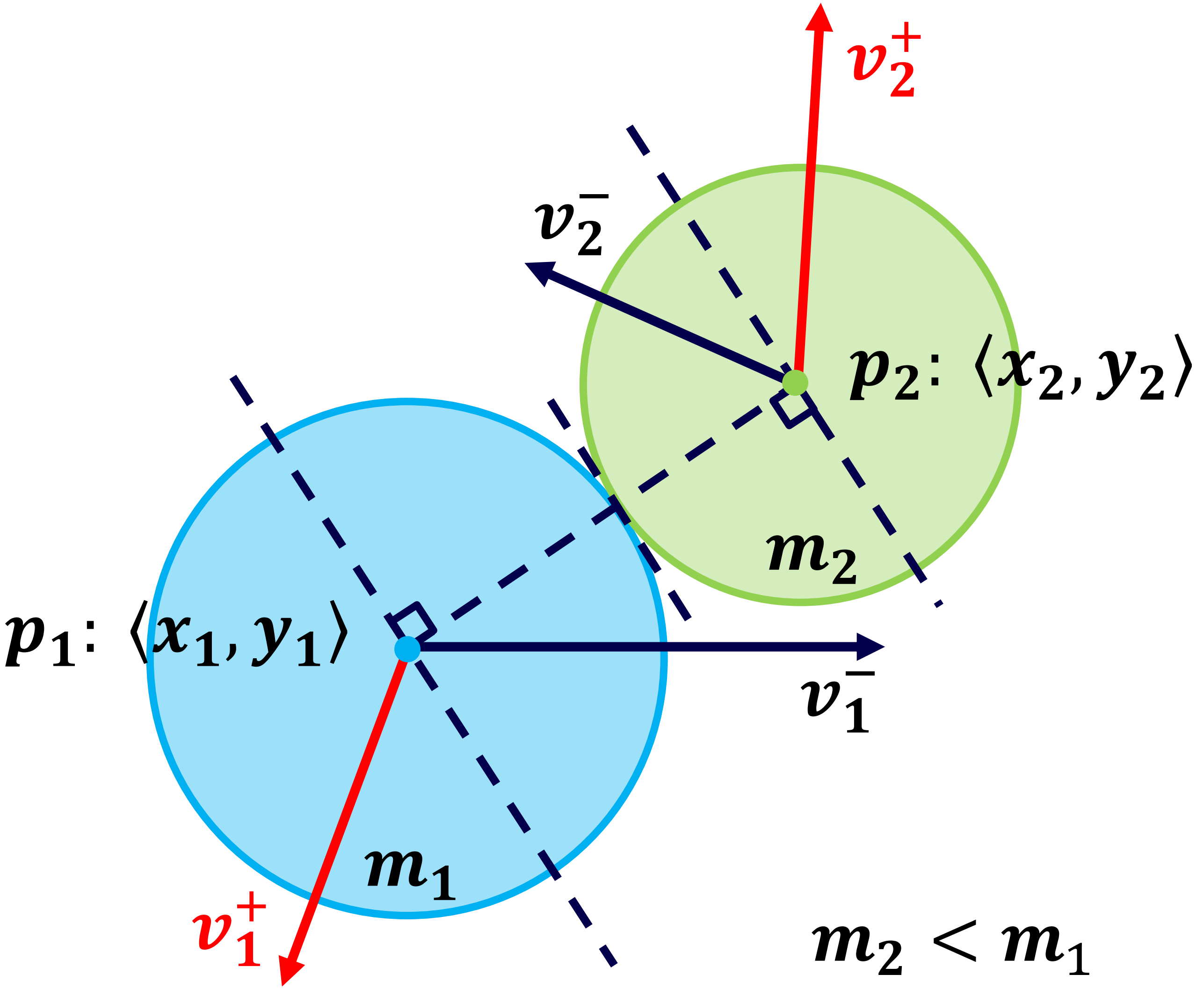
Particle-Particle Collisions (radius >0)

- **What we know…**
  - ***Particle centers***
  - ***Initial velocities***
  - ***Particle Masses***
- **What we can calculate…**
  - ***Contact normal***
  - ***Contact tangent***



Particle-Particle Collisions (radius >0)

- Impulse **direction** reflected across **tangent**
- Impulse **magnitude** proportional to **mass of other particle**



Particle-Particle Collisions (radius >0)

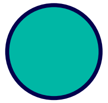
- **More formally…**

$$\boldsymbol{v}_1^+ = \boldsymbol{v}_1^- - \frac{2\boldsymbol{m}_2}{\boldsymbol{m}_1 + \boldsymbol{m}_2} \frac{\langle \boldsymbol{v}_1^- - \boldsymbol{v}_2^- \rangle \cdot \langle \boldsymbol{p}_1 - \boldsymbol{p}_2 \rangle}{\| \boldsymbol{p}_1 - \boldsymbol{p}_2 \|^2} \langle \boldsymbol{p}_1 - \boldsymbol{p}_2 \rangle$$

$$\boldsymbol{v}_2^+ = \boldsymbol{v}_2^- - \frac{2\boldsymbol{m}_1}{\boldsymbol{m}_1 + \boldsymbol{m}_2} \frac{\langle \boldsymbol{v}_2^- - \boldsymbol{v}_1^- \rangle \cdot \langle \boldsymbol{p}_2 - \boldsymbol{p}_1 \rangle}{\| \boldsymbol{p}_2 - \boldsymbol{p}_1 \|^2} \langle \boldsymbol{p}_2 - \boldsymbol{p}_1 \rangle$$

Self Study: Rigid Body Dynamics

- **From particles to rigid bodies…**



Particle

$$state = \begin{cases} \vec{x}\ position \\ \vec{v}\ velocity \end{cases}$$

$\mathbb{R}^4$ in 2D

$\mathbb{R}^6$ in 3D

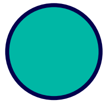
Rigid body

$$state = \begin{cases} \vec{x}\ position \\ \vec{v}\ velocity \\ q, R\ rotation\ matrix\ 3x3 \\ \vec{w}\ angular\ velocity \end{cases}$$

$\mathbb{R}^{12}$ in 3D

Self Study: Rigid Body Dynamics

- From particles to rigid bodies…



Newton's equations of motion

$$\Sigma\vec{F} = m\vec{a}$$

$$\begin{bmatrix} m & & \\ & m & \\ & & m \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} \Sigma\vec{F} \end{bmatrix}$$

$$M\vec{a} = \Sigma\vec{F}$$

Newton-Euler equations of motion

$$\begin{bmatrix} m & & & & & \\ & m & & & & \\ & & m & & & \\ & & & & & \\ & & & & \mathbf{I} & \\ & & & & & \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} \Sigma\vec{F} \\ \\ \end{bmatrix}$$

Inertia tensor

$$\Sigma\vec{\tau} - \vec{w} \times \mathbf{I}\vec{w}$$