# Colliders and Collisions

CMSC425.01 Spring 2019

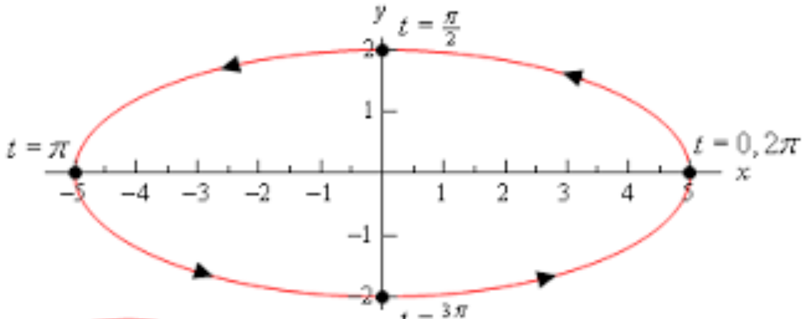
Still at tables …

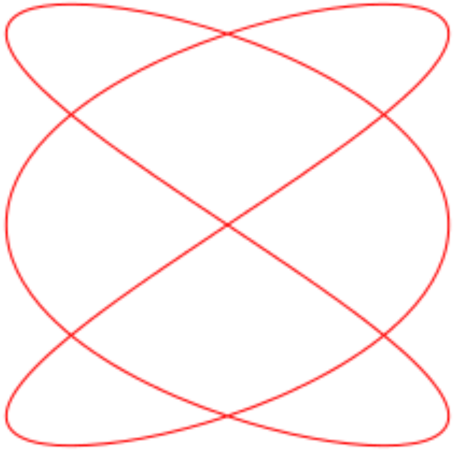
# Administrivia

- Next Hw and Project 2 coming
  - Project 2 – like Project 2 from previous semesters (animated characters, navmesh) but crabs on a beach!
- Mini-lectures coming – videos on single topics (Panopto on Elms)
- The M-word – Midterm.

# Digression 1: Parametric curves (surfaces)

- Types
- Lines
  - Circles
  - Cubic ($x^3$ == human perception)
  - Spheres
  - Bezier curves
- Operations
  - Draw with for loop
  - Tangent is vector derivative
  - Vector representation



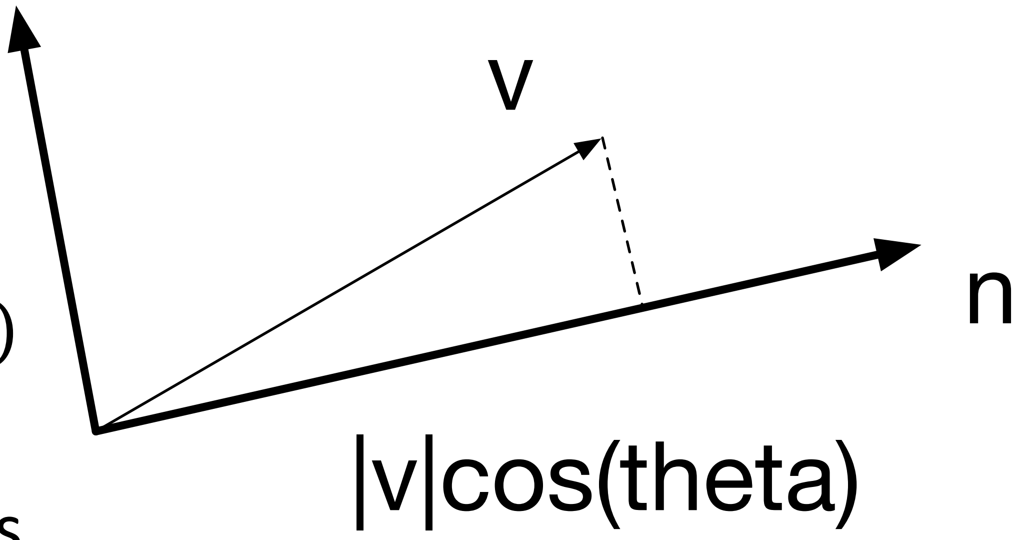


Coordinates are function of parameter t
<x(t),y(t),z(t)>

# Digression 2: Getting projections right

- Projection of v onto n

$$v \cdot n = |v||n|\cos(\theta)$$



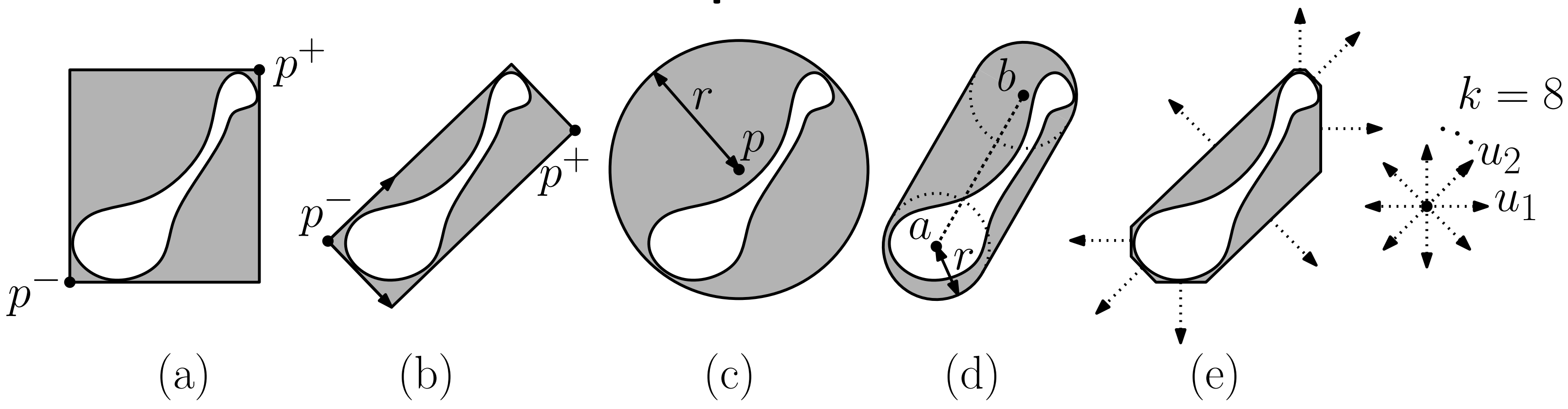
- if $|n| = 1$ then $v \cdot n = |v|\cos(\theta)$
- if $|n| = |v| = 1$ then $v \cdot n = \cos(\theta)$

- When normalize – care about values
- When not – comparisons, signs

# Today’s questions

1) Detecting collisions
2) Organizing spatial data

# Standard collider shapes



(a) Axis-aligned boxes (AABB)

(b) General bounding boxes

(c) Bounding spheres (ellipsoids)

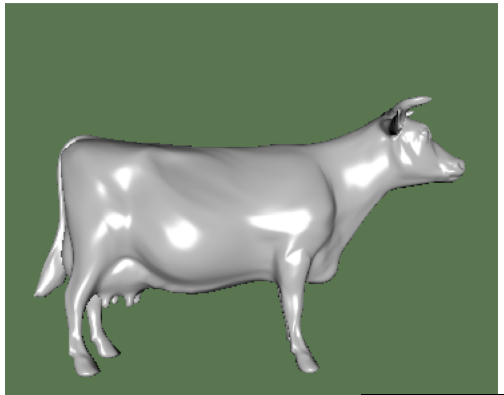
(d) Capsules

(e) k-DOPs (k-discrete oriented polytope)

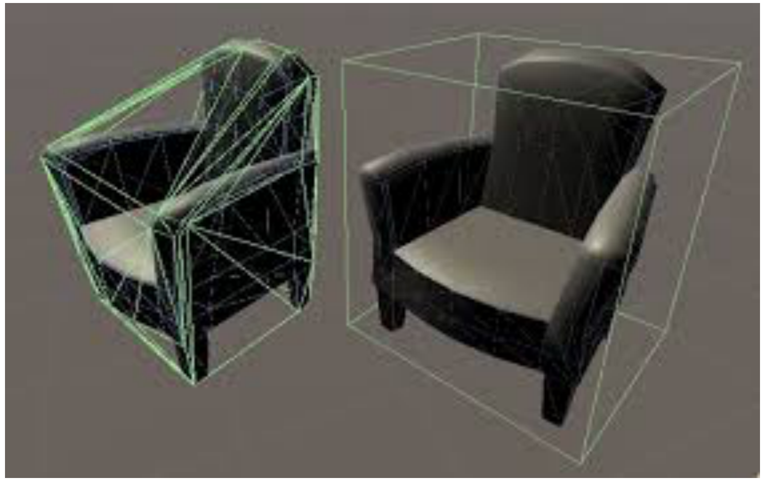
Also – point, mesh,

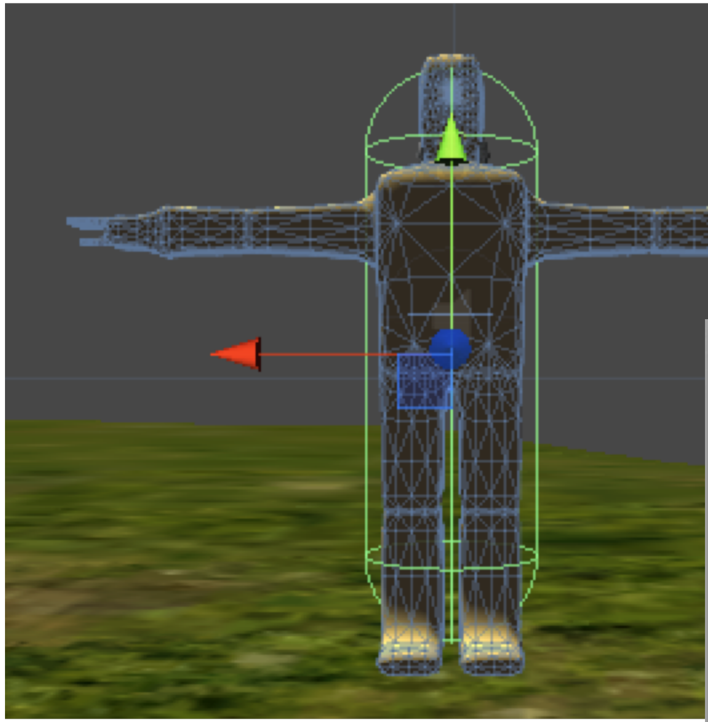
convex hull

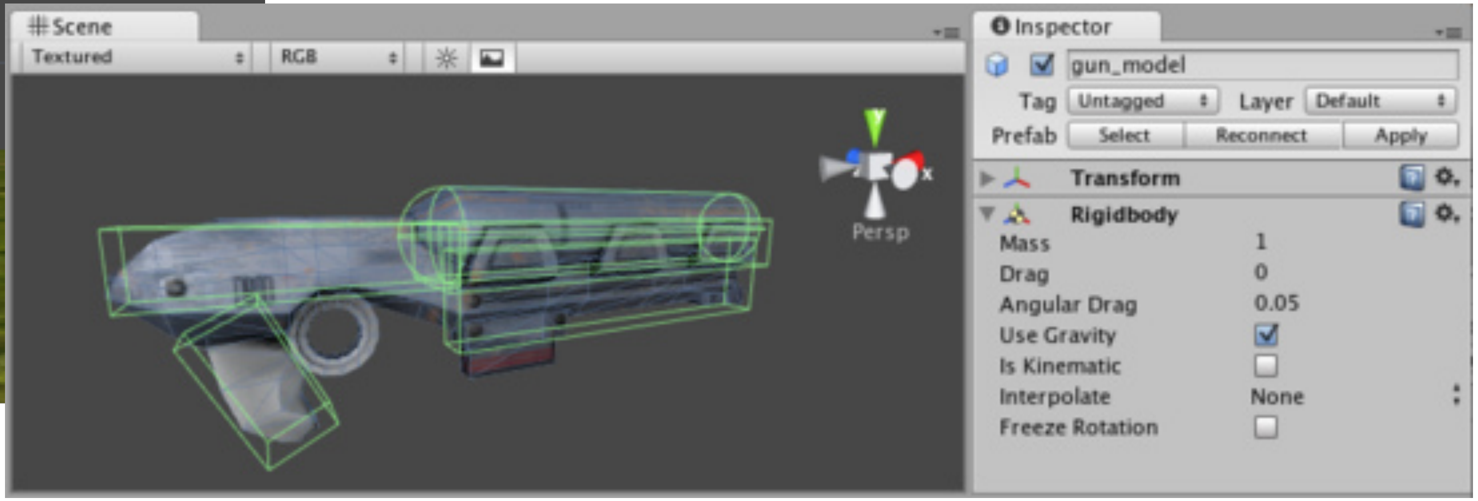
# What would you use?



# Examples







☑ gun_model

Tag Untagged Layer Default

Prefab Select Reconnect Apply

Transform

Rigidbody

Mass 1

Drag 0

Angular Drag 0.05

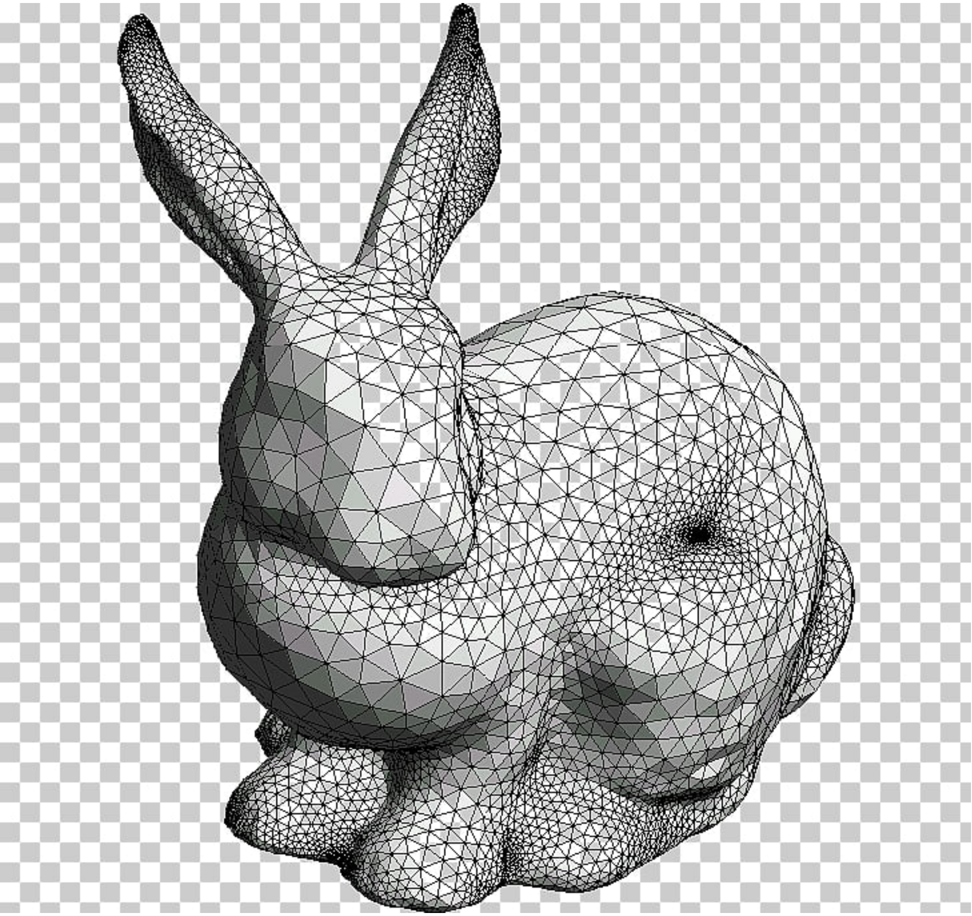
Use Gravity ☑

Is Kinematic □

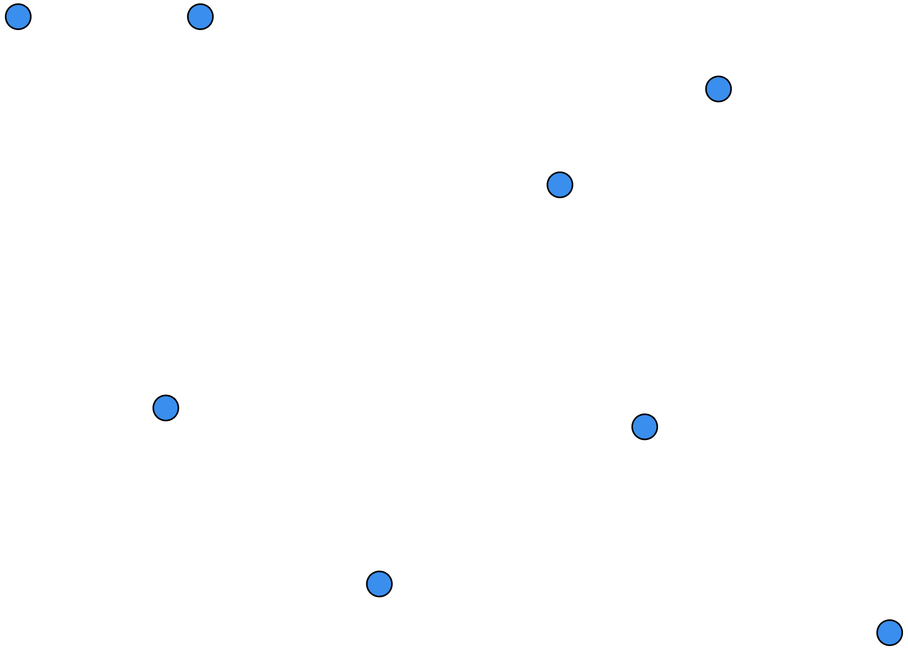
Interpolate None

Freeze Rotation □

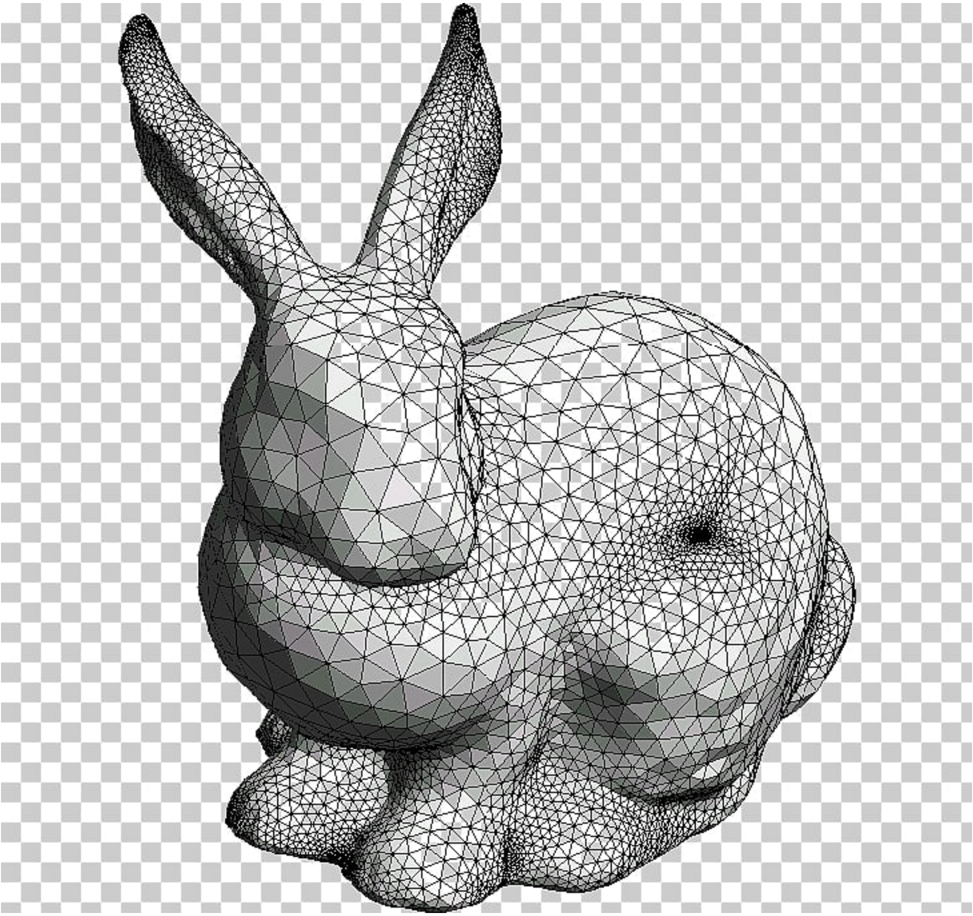
# Fitting the collider



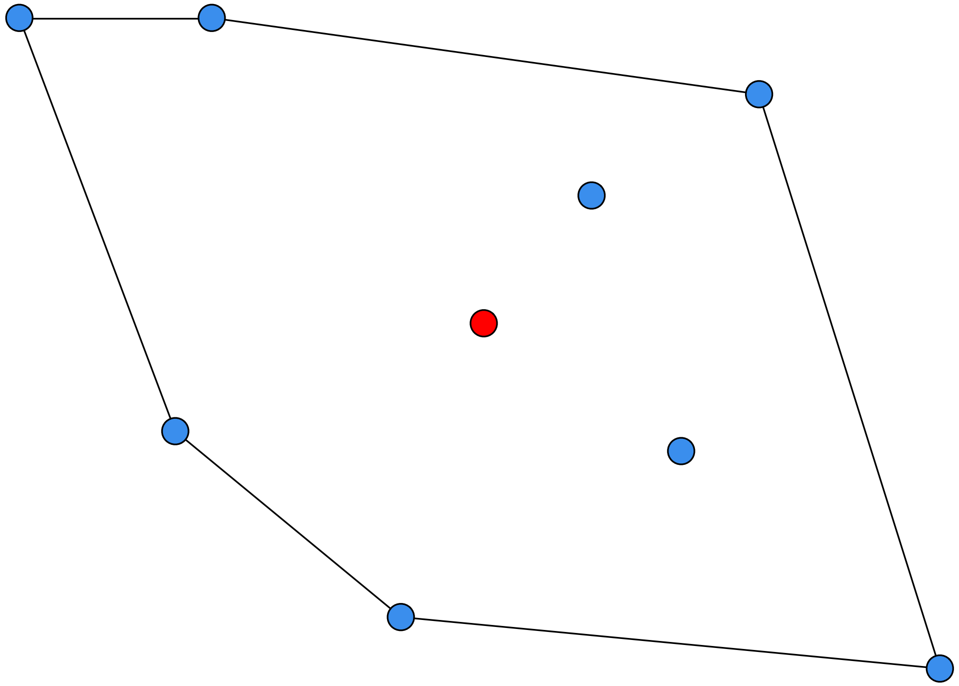
- Data is a set of points



# Fitting the collider

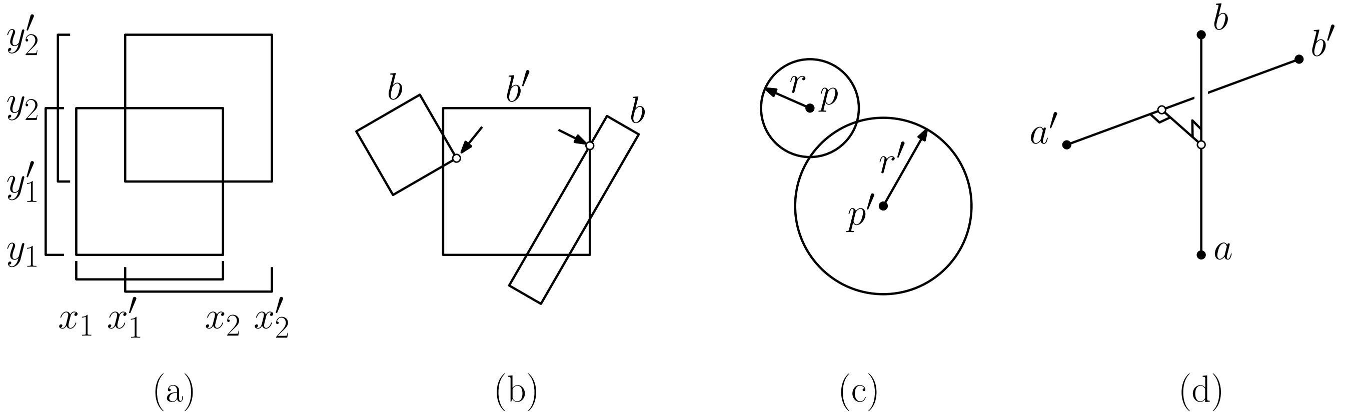


- Centroid and convex hull



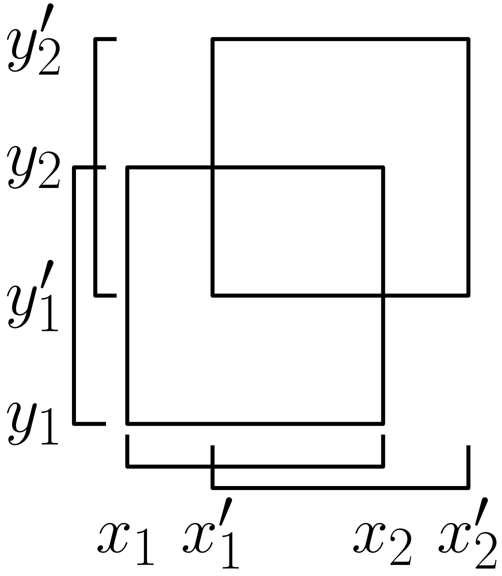
# Detecting collisions – how?

- AABB x AABB
- Box x Box
- Sphere x Sphere
- Capsule x Capsule

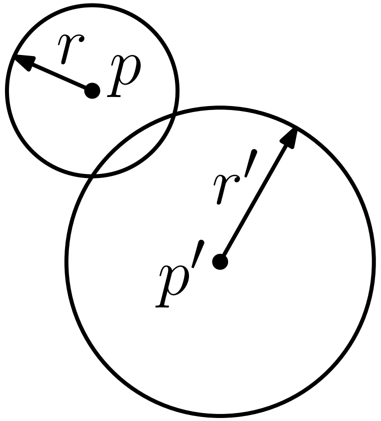


# "Easy" cases

- AABB x AABB

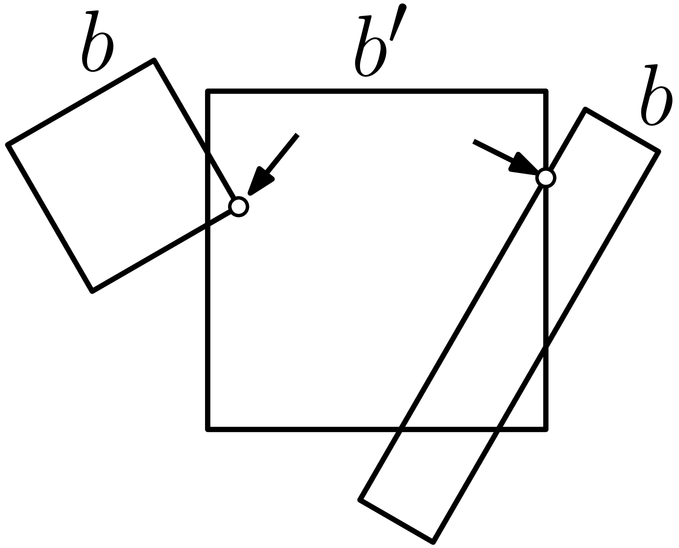


- Sphere x Sphere



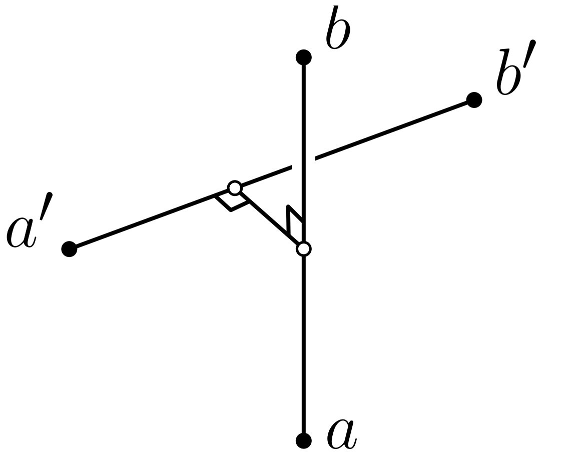
# Box to box with rotations

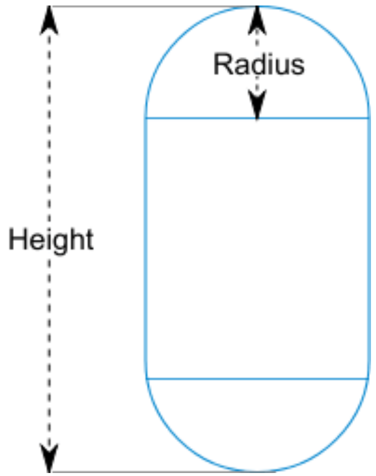
- Rotate one to align with axes

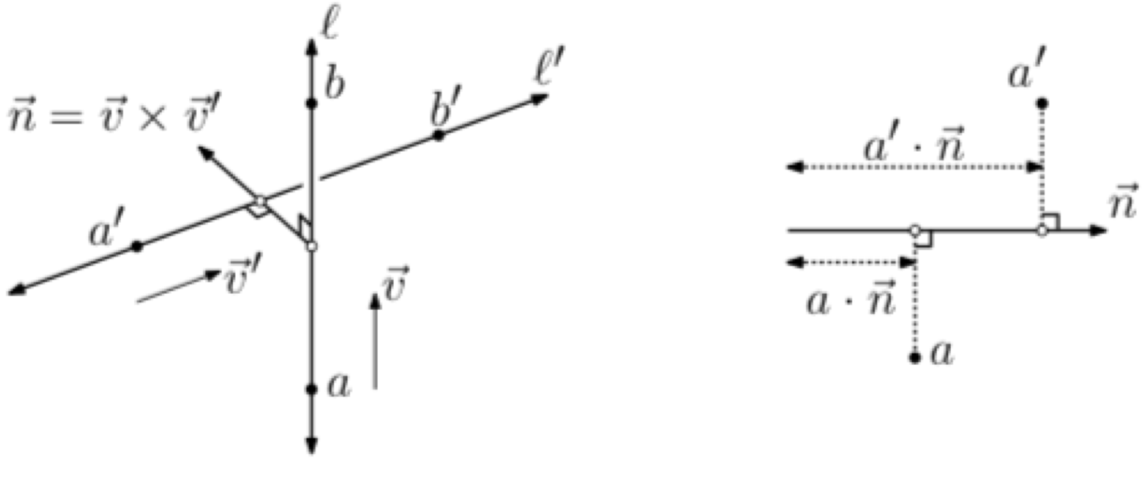


# Capsule to capsule

- Distance between two line segments





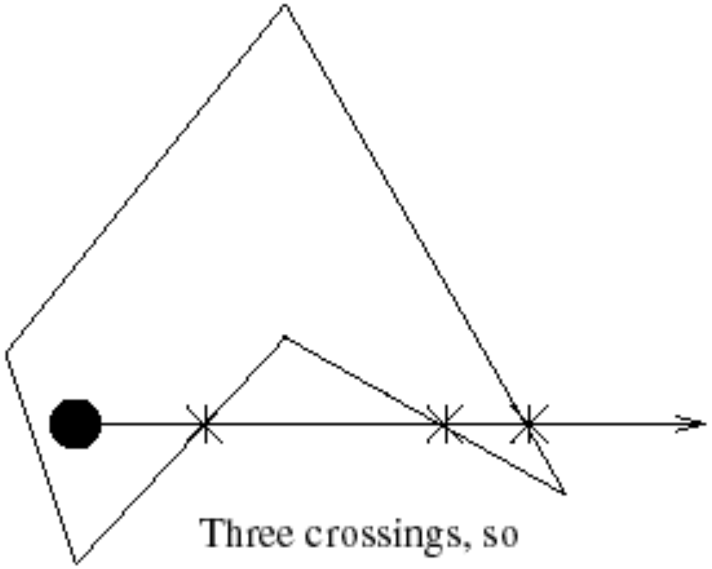


(a)

(b)

# Other collisions

- Cone to point (shot gun)
- Sphere to plane (hw)
- Cylinder to point (practice)
- Point in polygon
- Polygon to polygon

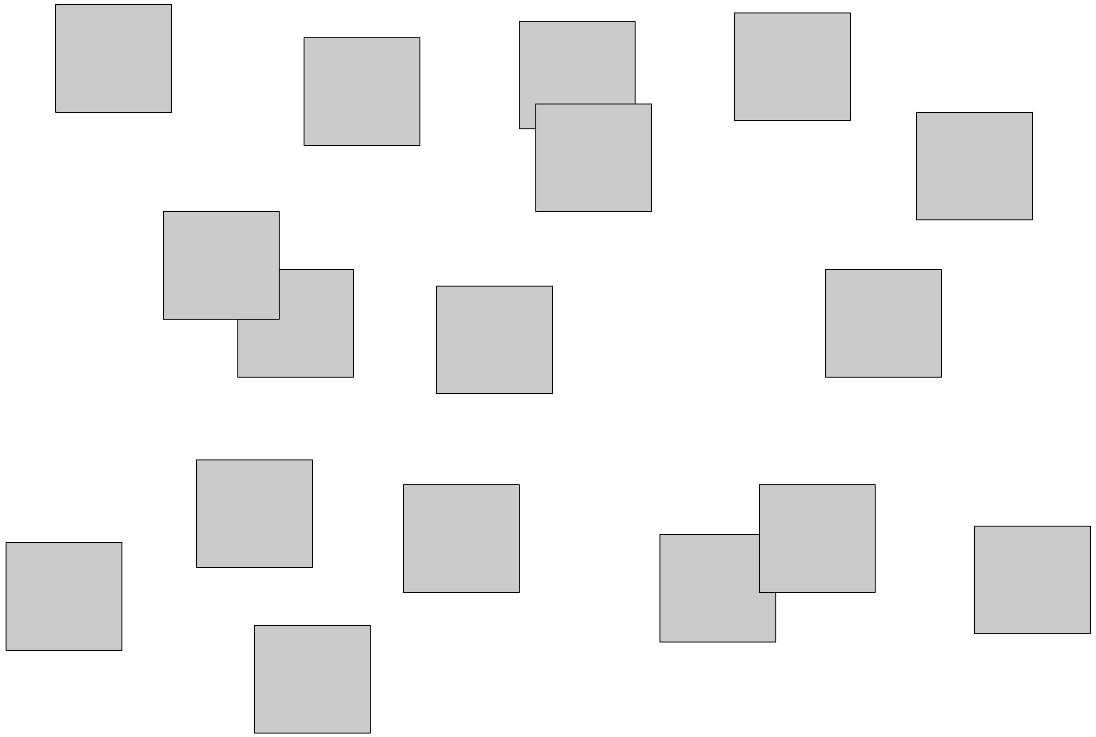


Three crossings, so point is inside

Figure 1 - Crossings Test

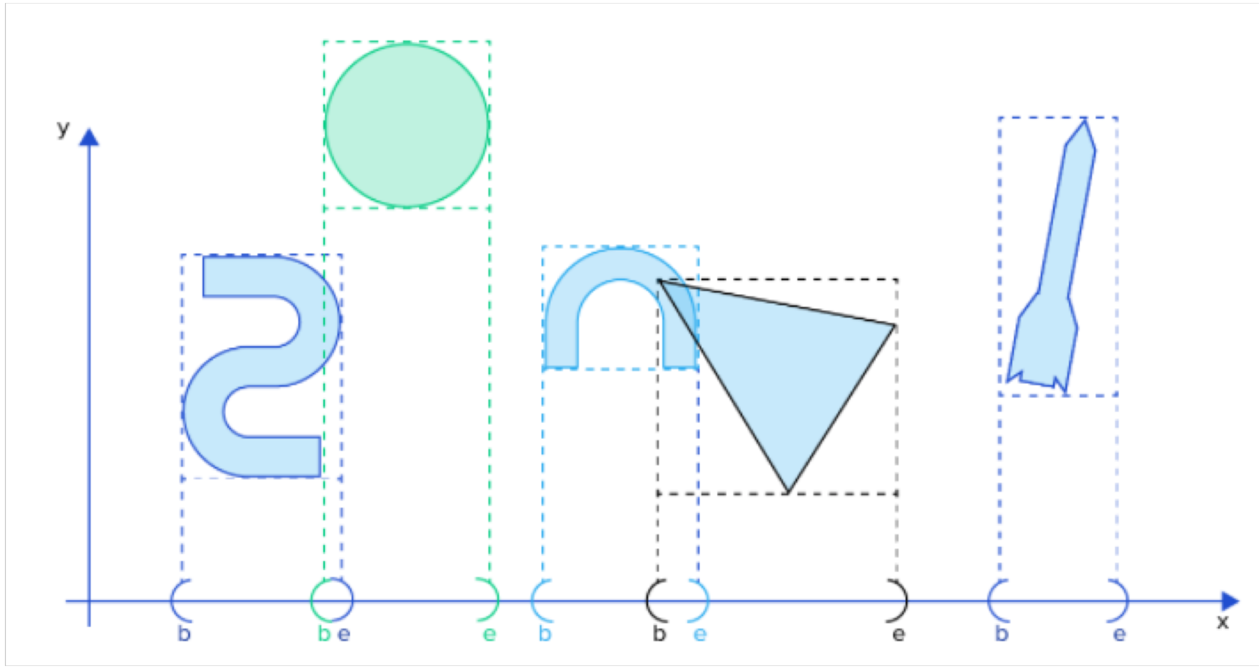
# How to do many efficiently?

- Hierarchical colliders
  - First test bounding box
  - If hit then test better collider
- Problem with many
  - Better than n-squared
  - No obvious sort in 2 or 3D



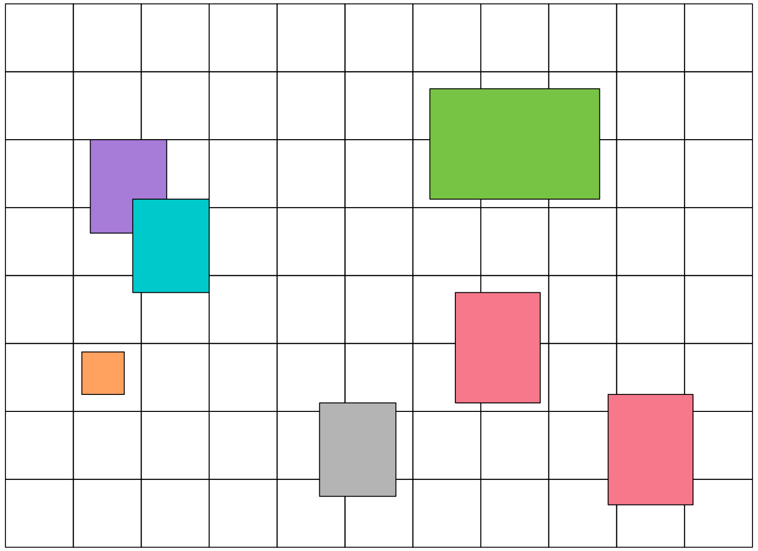
# Sort and sweep algorithm

- Project bounding boxes on one coordinate
- Sort along that coordinate
- Filter tests to overlaps

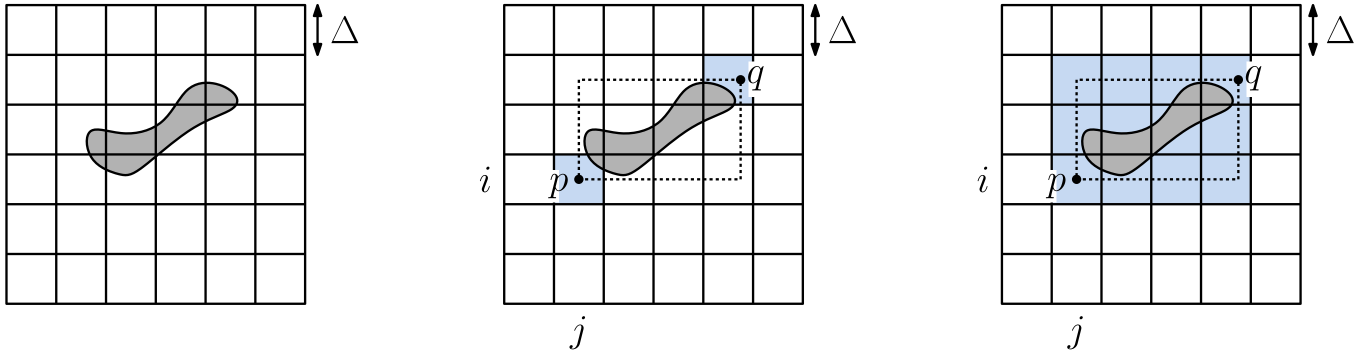


# Grid

- Overlap shapes on grid
- For each cell hit by shape, create ptr to shape



- If two shapes in same cell then need further test

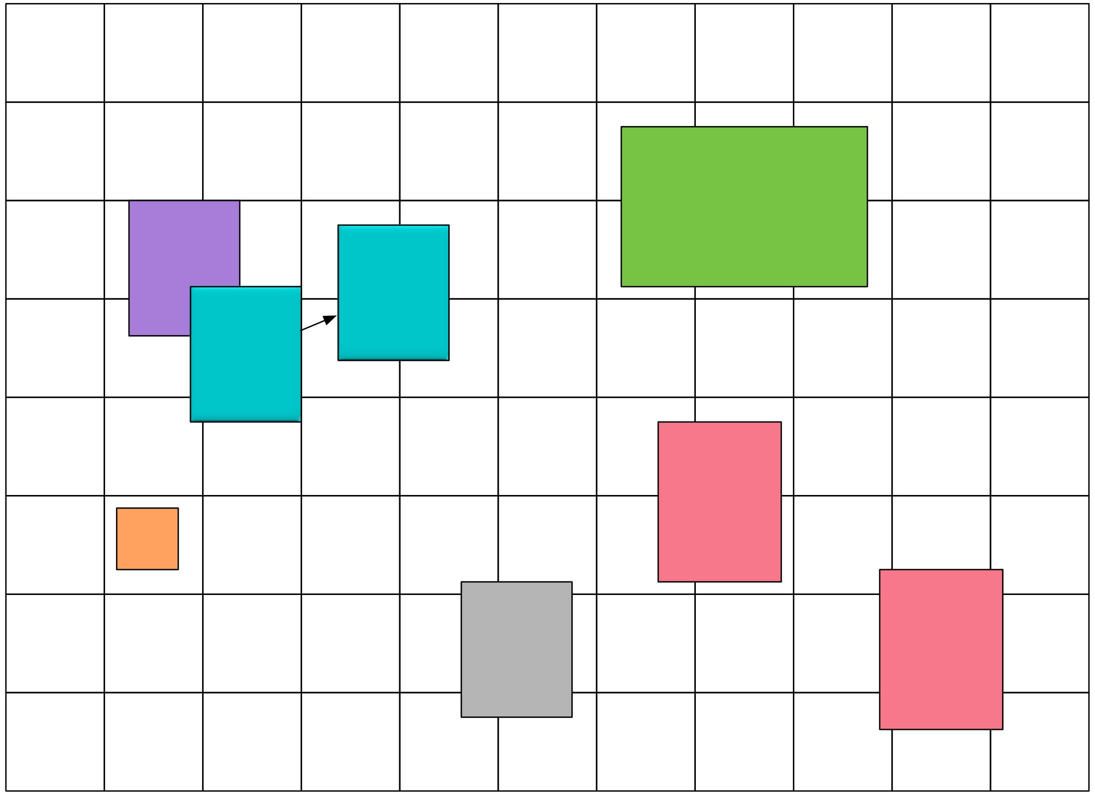


(a) (b) (c)

- What size grid?
- How update grid?

# Grid

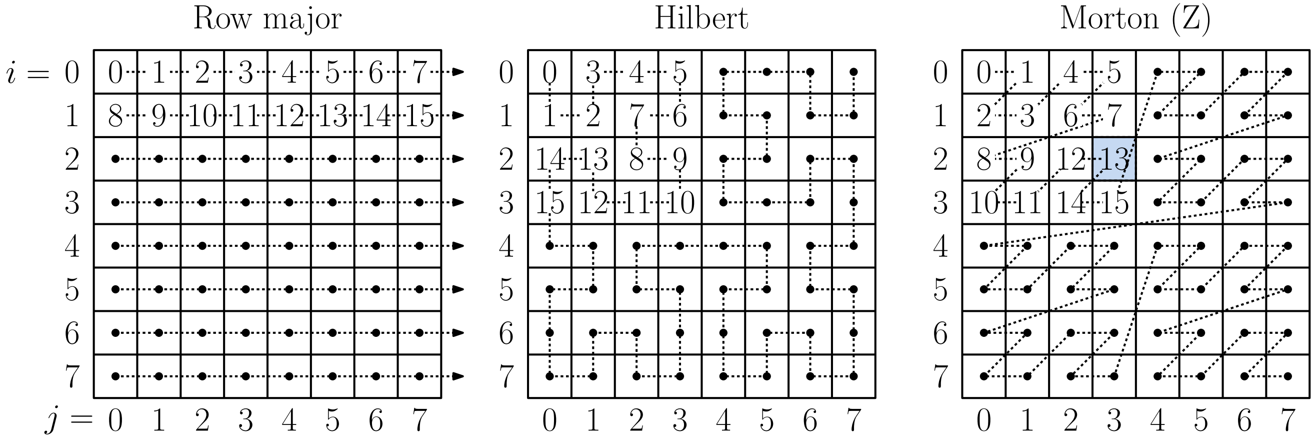
- How treat moving and static objects?
  - One agent in static space?



# How store grid?

- Row-column order (standard)
- Hashmap
- Space filling order
  - Hilbert
  - Morton

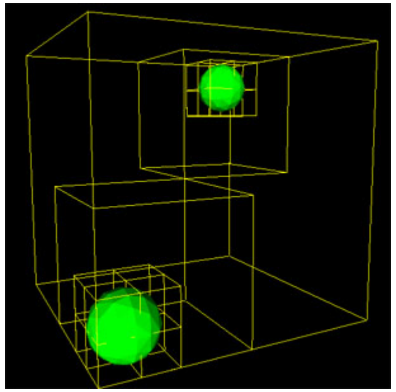
- Bit shuffle for Morton’s
  - See notes

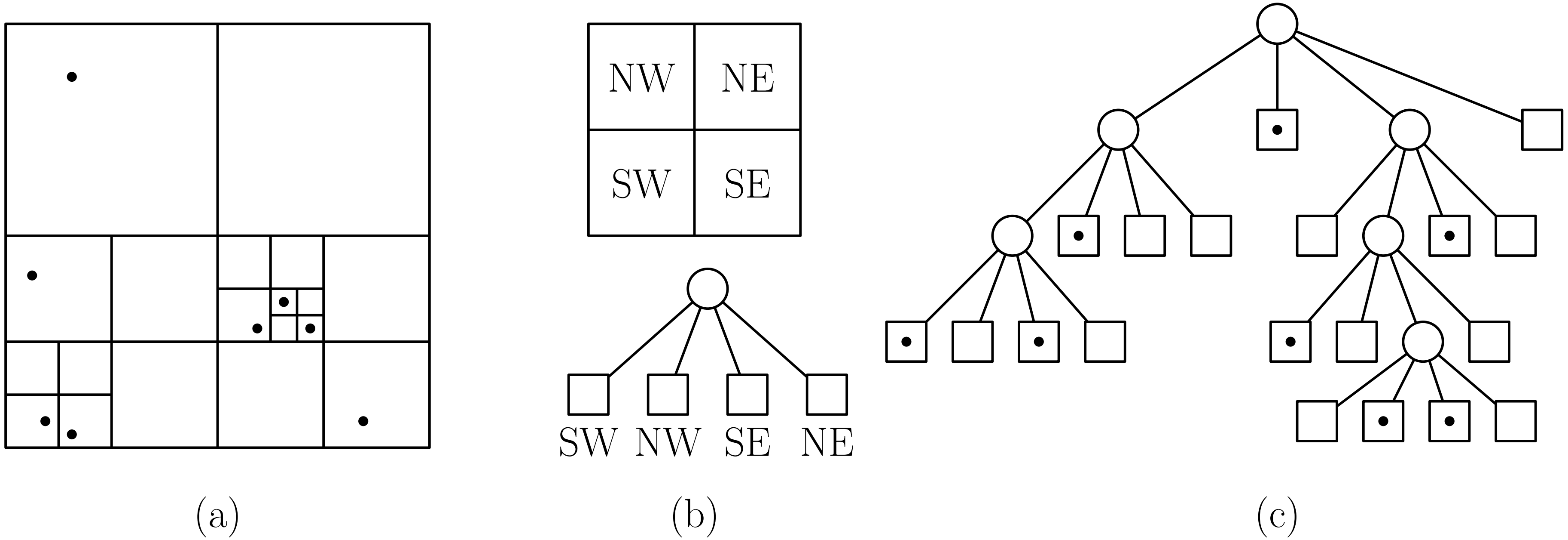


(a) (b) (c)

# Quadtrees: hierarchical space decomposition

- Four way division on midpoint
  - NW, NE, SW, SE
  - Midpt independent of data
- 3D
  - Octrees





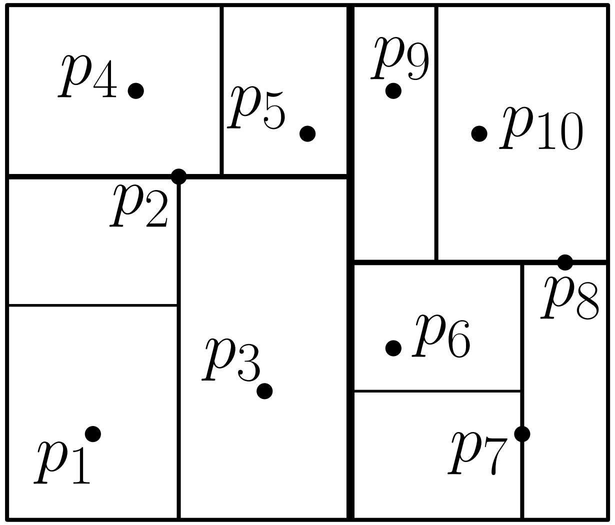
(a)

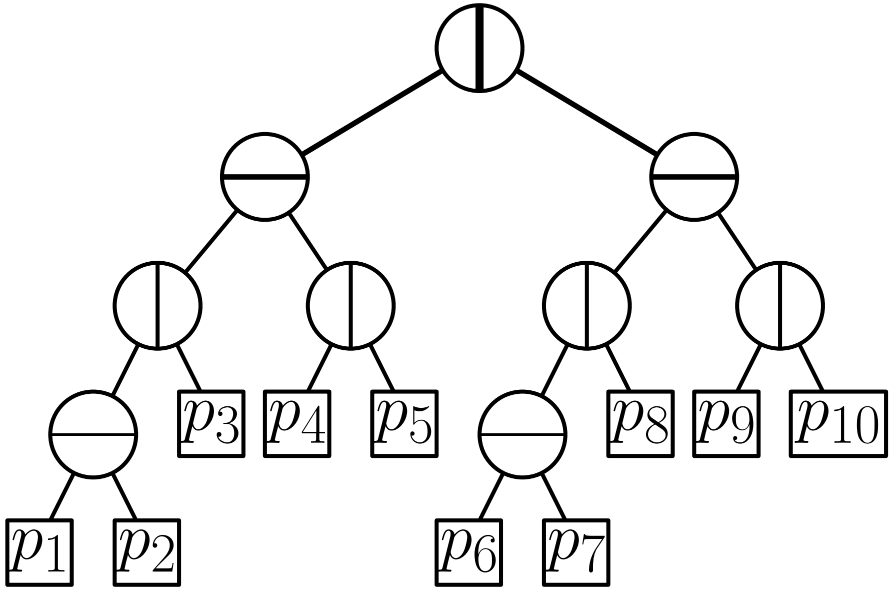
(b)

(c)

# K-d trees

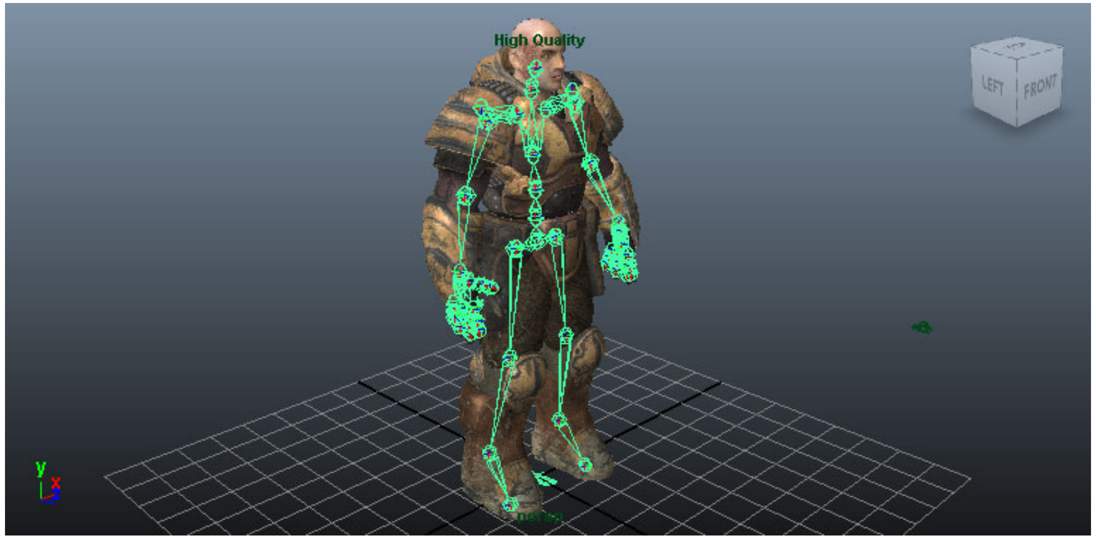
- Alternating coordinates
- Divisions based on data





# Skeletons and rigging

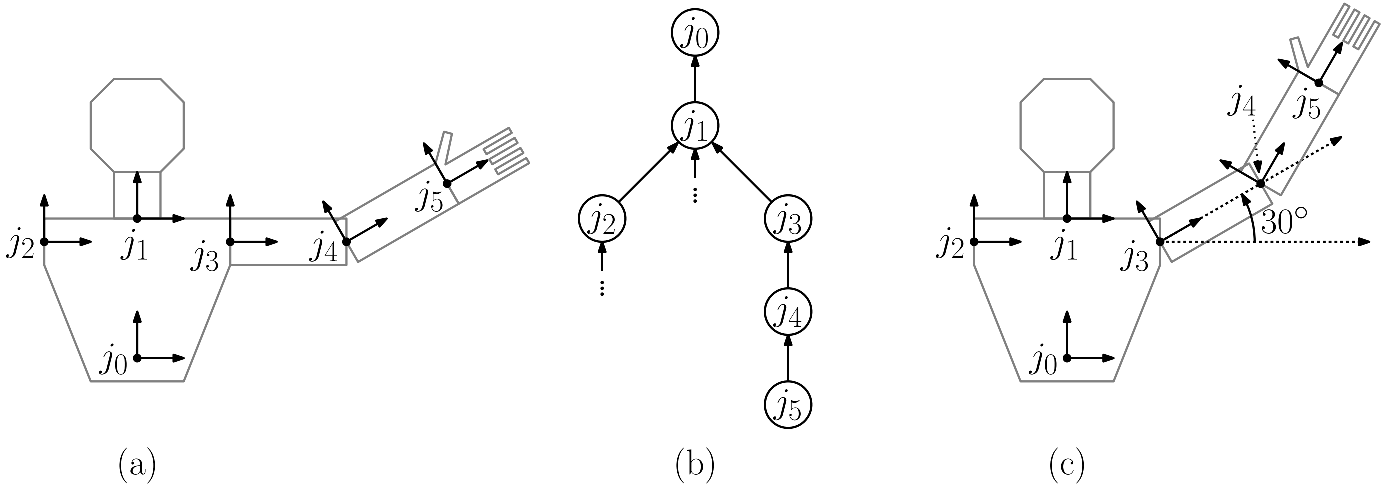
- Character animation
  - Create a skeleton
  - Define transforms between parts
  - Interpolation transforms to move
  - Rig with “flesh”
  - Create behavior animations
  - Blend between animations for smooth actions in game
- Can find as Unity Assets
- Use Mecanim tool
- [https://www.youtube.com/watch?v=HPwu7eIwjV8](https://www.youtube.com/watch?v=HPwu7eIwjV8)



Camera Persp
z
y
x
(0) MinotaurArmature Strap_F_base_L

# Step 1: Skeleton and transformations

- ***Kinematics***
  - Forward – given joints and transformations, estimate end position
  - Reverse – given end position estimate transformations



- Forward – “easy”
- Reverse – hard!

# Readings

- David Mount's lectures on Geometric Data structures and on Skeletal Animation and Kinematics

- Good tutorial on collisions
- https://www.toptal.com/game/video-game-physics-part-ii-collision-detection-for-solid-objects