Colliders and Collisions

CMSC425.01 Fall 2019

Administrivia

- Hw1 due Wednesday
- Old Hw1 with solutions now on web site
- Also notes on Ray-Circle intersection
  - Problem and solution from spring 2019
  - Notes on the problem from this semester
- Final project proposal out

Today’s questions

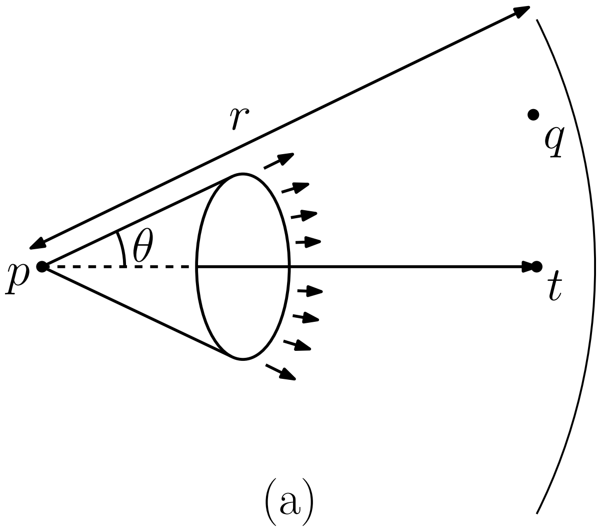
1) Applying geometry to game problems
2) How to detect object collisions

Later: How to put into Unity

Problem 1: Shot gun weapon

- Problem:
- Given weapon defined by
  - Location p
  - Target point t
  - Spread angle θ
- And object defined by
  - Location q

- Return true if q hit

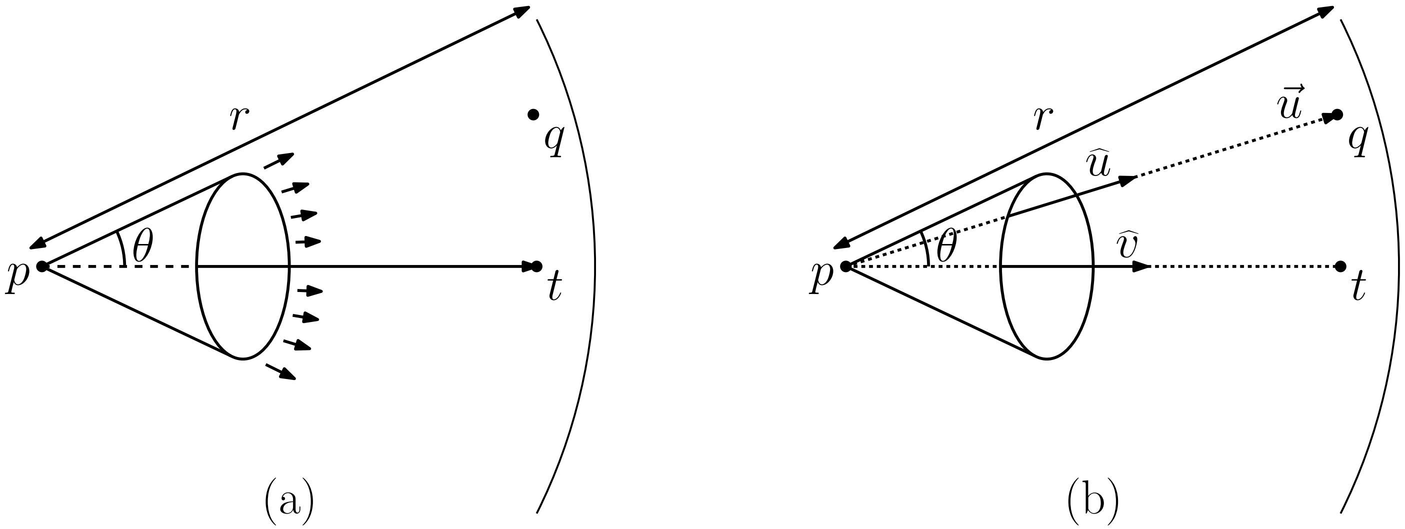


(a)

Problem 1: Shot gun weapon

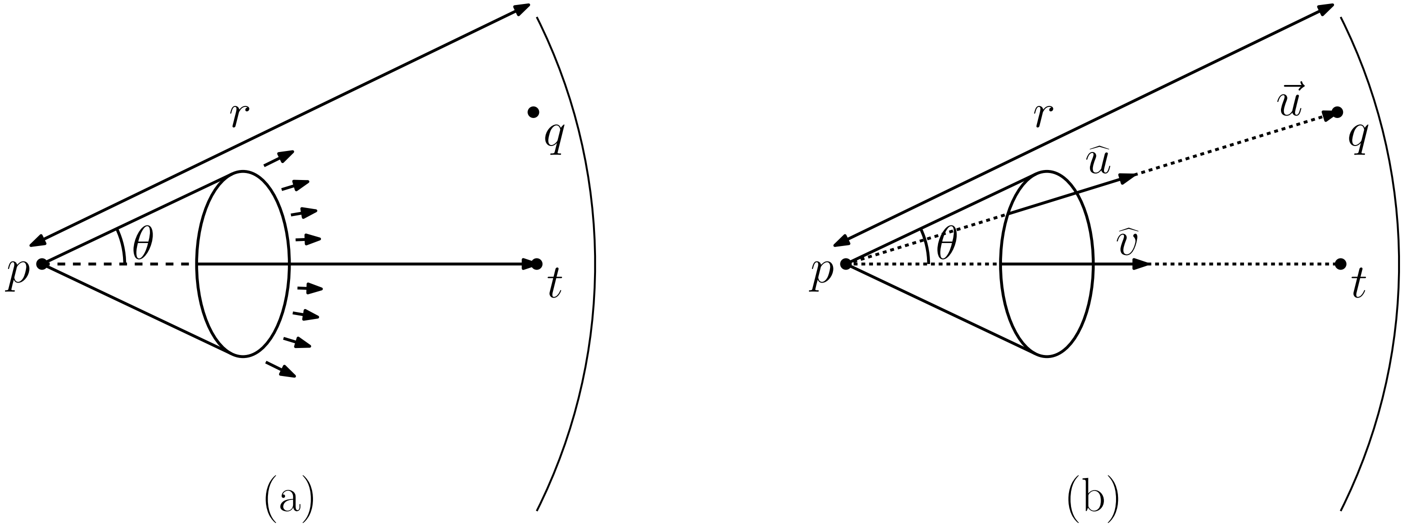
- Problem:
- Given weapon defined by
  - Location p
  - Target point t
  - Spread angle θ
- And object defined by
  - Location q

- Return true if q hit



Problem 1: Shot gun weapon

- Problem:
- Given weapon defined by
  - Location p
  - Target point t
  - Spread angle θ
- And object defined by
  - Location q
- Return true if hit

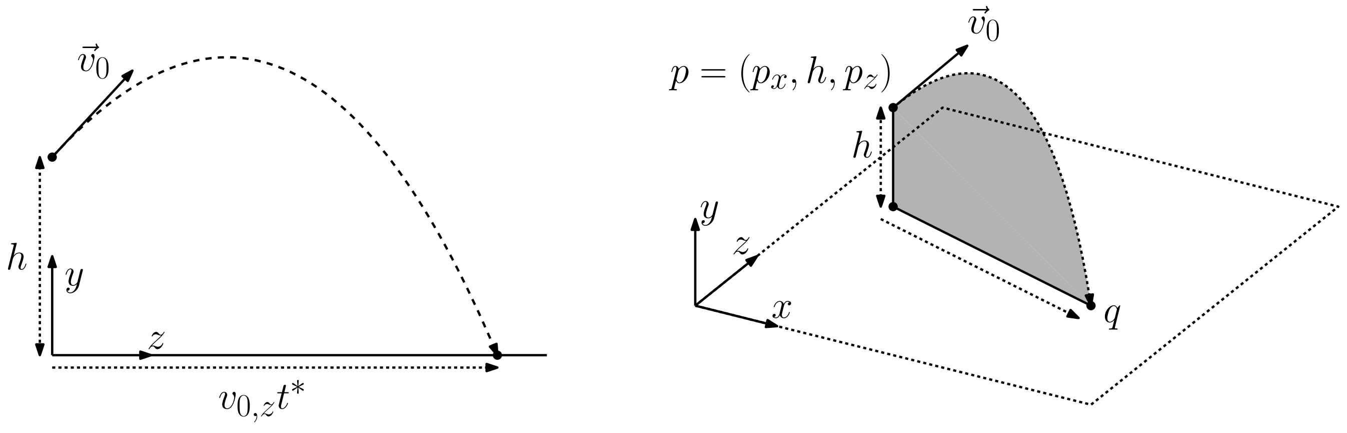


$$
\begin{aligned}
\vec{v} &\leftarrow t - p; \qquad \vec{u} \leftarrow q - p \\
\ell(v) &\leftarrow \|\vec{v}\| = \sqrt{\vec{v}\cdot\vec{v}}; \qquad \ell(u) \leftarrow \|\vec{u}\| = \sqrt{\vec{u}\cdot\vec{u}} \\
\widehat{v} &\leftarrow \text{normalize}(\vec{v}) = \vec{v}/\ell(v); \qquad \widehat{u} \leftarrow \text{normalize}(\vec{u}) = \vec{u}/\ell(u) \\
c_1 &\leftarrow \widehat{u}\cdot\widehat{v} \\
c_2 &\leftarrow \cos\left(\theta\cdot\frac{\pi}{180}\right) \\
\text{return} &\quad \textbf{true} \text{ iff } (c_1 \geq c_2 \text{ and } \ell(u) \leq r).
\end{aligned}
$$

Problem 2: Projectile aiming tool

- Problem:
- Given projectile with
  - Initial location (0,h,0)
  - Initial velocity $\vec{v}_0 =< v_{0,x}, v_{0,y}, v_{0,z} >$

- Find landing location
  - Location (x,0,z)



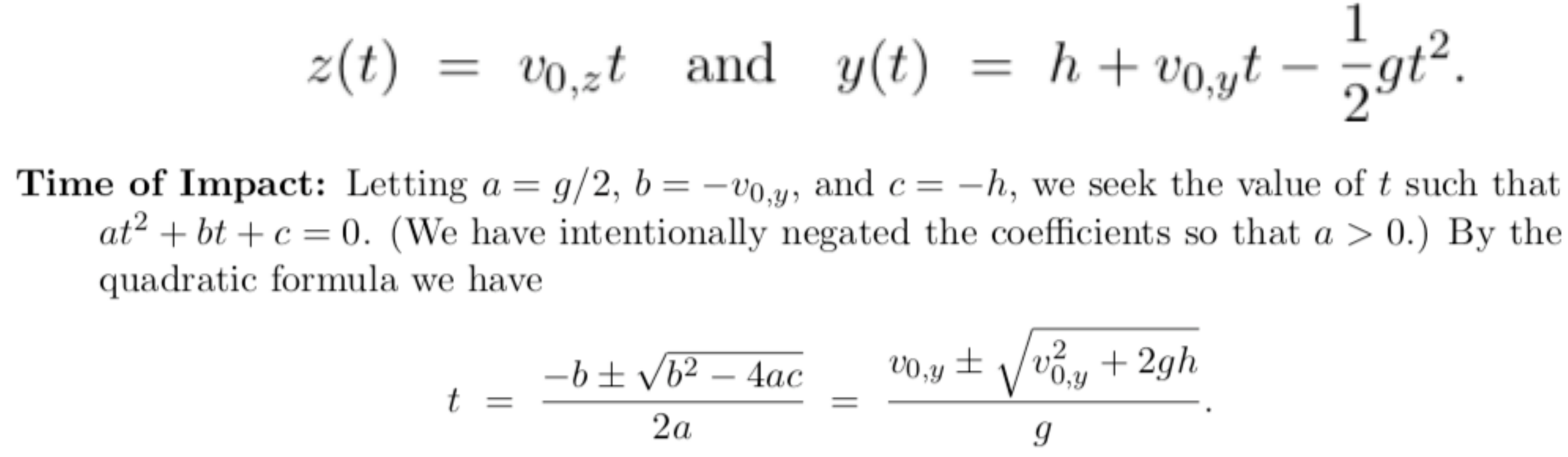
(a)

(b)

Problem 2: Projectile aiming tool

- Problem:
- Given projectile with
  - Initial location (0,h,0)
  - Initial velocity $\vec{v}_0 =< v$

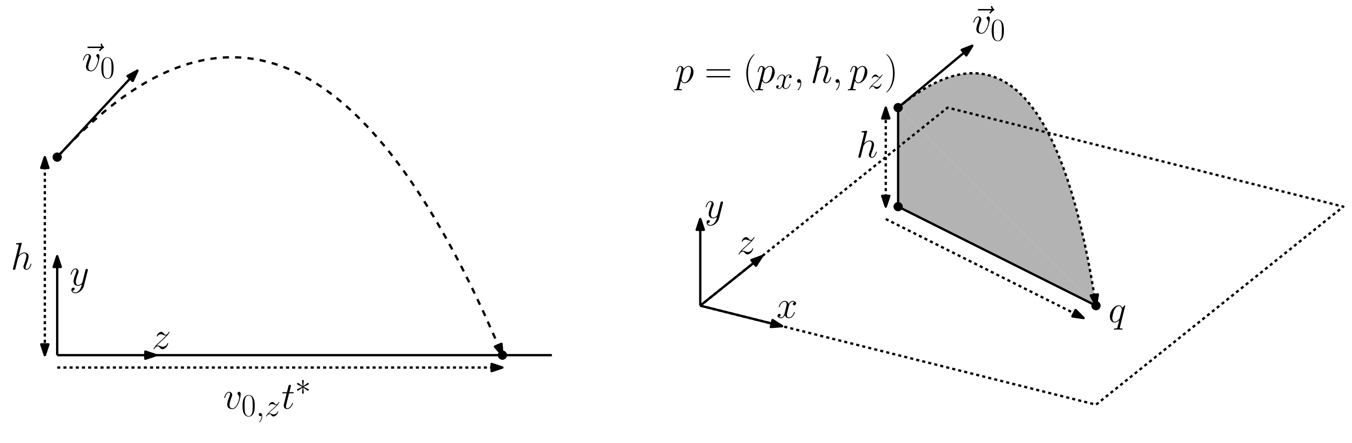
- Find landing location
  - Location (x,0,z)



$$z(t) \ = \ v_{0,z}t \quad \text{and} \quad y(t) \ = \ h + v_{0,y}t - \frac{1}{2}gt^2.$$

Time of Impact: Letting $a = g/2$, $b = -v_{0,y}$, and $c = -h$, we seek the value of t such that $at^2 + bt + c = 0$. (We have intentionally negated the coefficients so that $a > 0$.) By the quadratic formula we have

$$t \ = \ \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \ = \ \frac{v_{0,y} \pm \sqrt{v_{0,y}^2 + 2gh}}{g}.$$

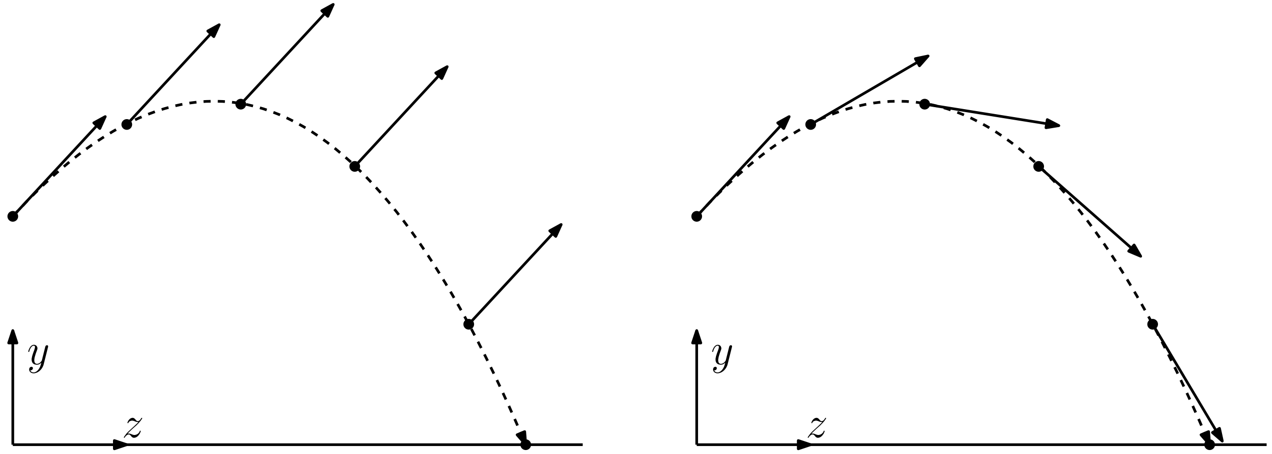


(a)

(b)

Problem 3: Shooting an(d) arrow

- Problem:
- If projectile show direction (eg, arrow)
  - Initial location (0,h,0)
  - Initial velocity $\vec{v}_0 =< v_{0,x}, v_{0,y}, v_{0,z} >$
- Find direction orientation
  - Location (x,0,z)



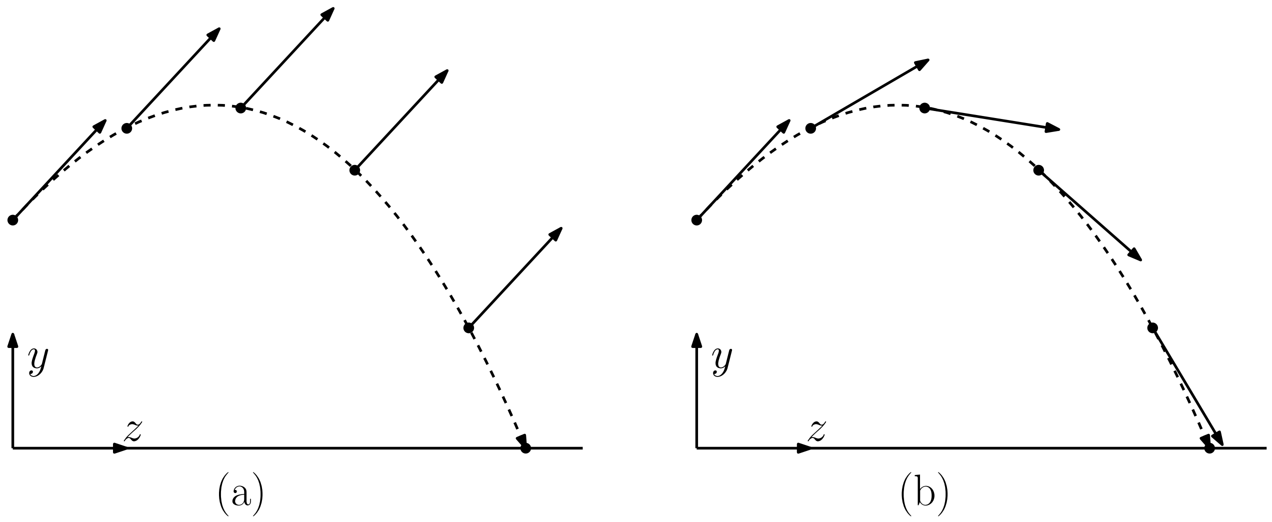
(a)

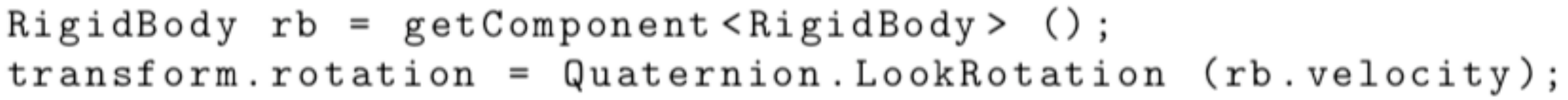
(b)

Problem 3: Shooting an(d) arrow

- Problem:
- If projectile show direction (eg, arrow)
  - Initial location (0,h,0)
  - Initial velocity $\vec{v}_0 =< v_{0,x}, v_{0,y}, v_{0,z} >$

- Find direction orientation
  - Location (x,0,z)

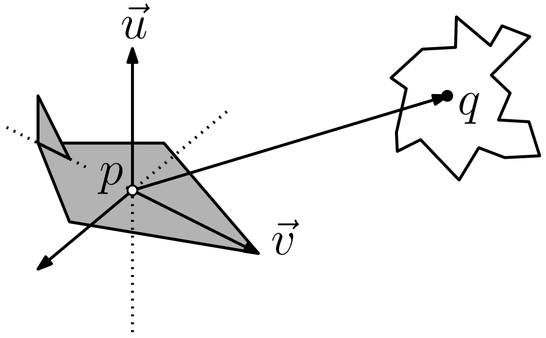




```
RigidBody rb = getComponent<RigidBody> ();
transform.rotation = Quaternion.LookRotation (rb.velocity);
```

Problem 4: Evasive action

- Problem:
- Given ship defined by
  - Location p
  - Forward vector v
  - Up vector u (perpendicular to v?)
- And object defined by
  - Location q



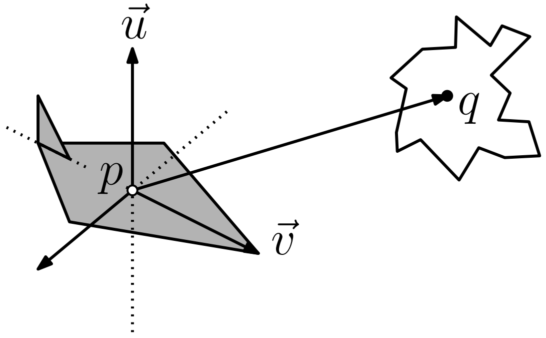
(a)

- Determine if ship should evade
  - Turning up or down
  - Turning left or right

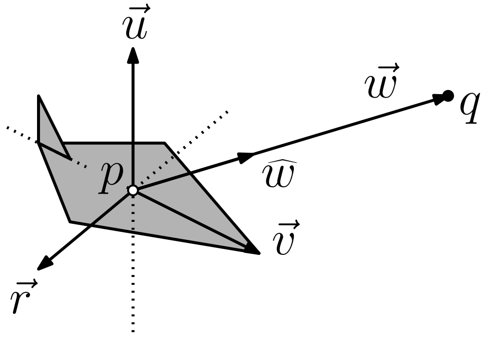
Problem 4: Evasive action

- Problem:
- Given ship defined by
  - Location p
  - Forward vector v
  - Up vector u (perpendicular to v?)
- And object defined by
  - Location q

- Determine if ship should evade
  - Turning up or down
  - Turning left or right



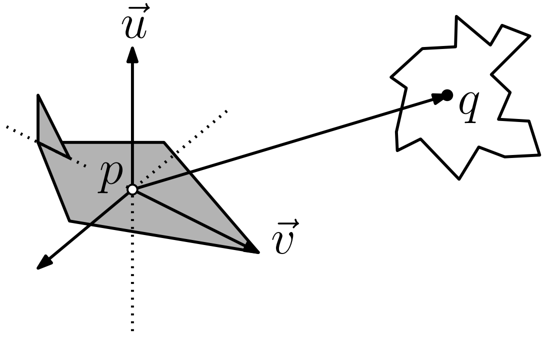
(a)



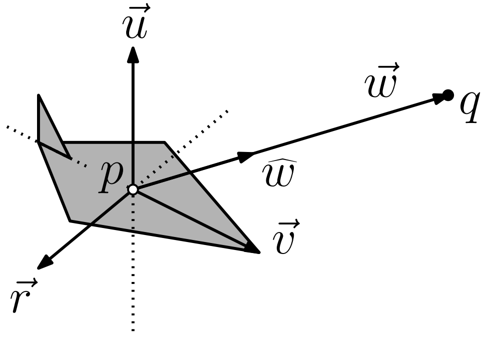
(a)

Problem 4: Evasive action

- Problem:
- Given ship defined by
    - Location p
    - Forward vector v
    - Up vector u (perpendicular to v?)
- And object defined by
    - Location q



(a)



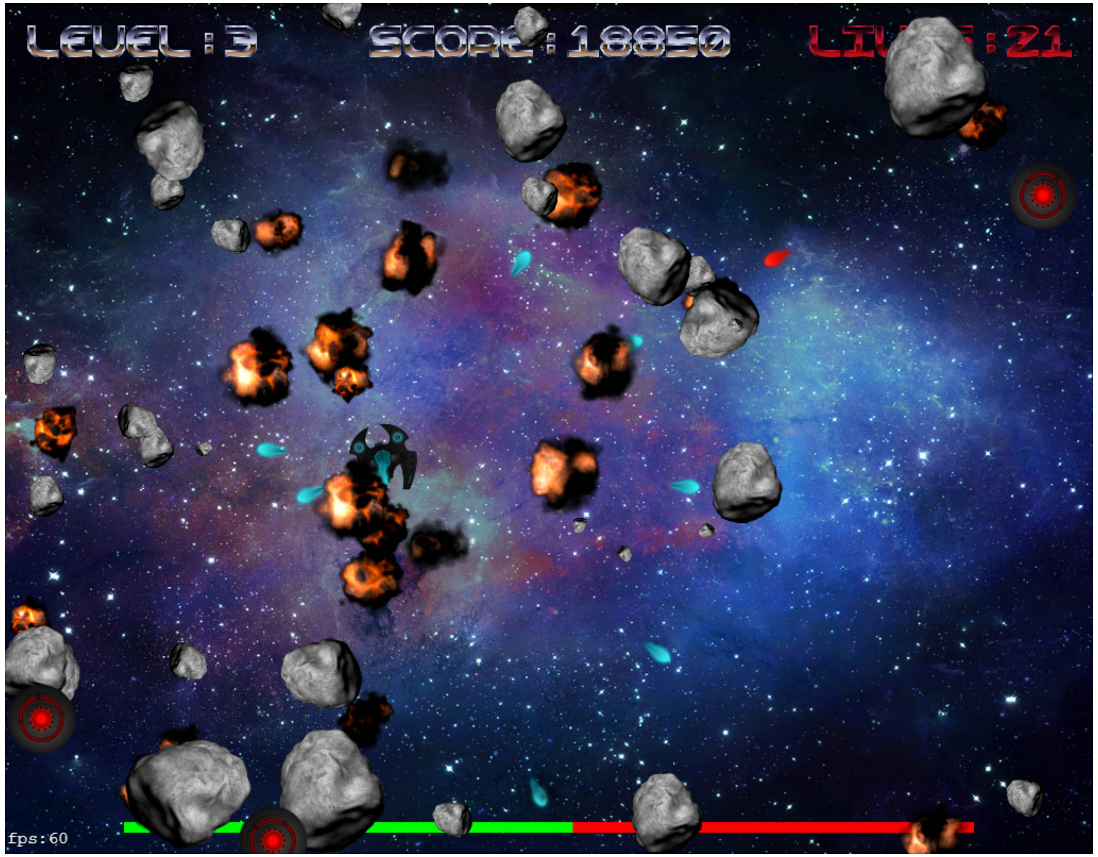
(a)

- Determine if ship should evade
    - Turning up or down
    - Turning left or right

$$\begin{aligned}
\widehat{w} \cdot \vec{u} \;\geq\; 0 \;&\Rightarrow\; \text{(obstacle above) pitch downwards} \\
\widehat{w} \cdot \vec{u} \;<\; 0 \;&\Rightarrow\; \text{(obstacle below) pitch upwards.} \\
\vec{r} \;&\leftarrow\; \vec{v} \times \vec{u} \\
\widehat{w} \cdot \vec{r} \;\geq\; 0 \;&\Rightarrow\; \text{(obstacle to the right) yaw to the left} \\
\widehat{w} \cdot \vec{r} \;<\; 0 \;&\Rightarrow\; \text{(obstacle to the left) yaw to the right.}
\end{aligned}$$

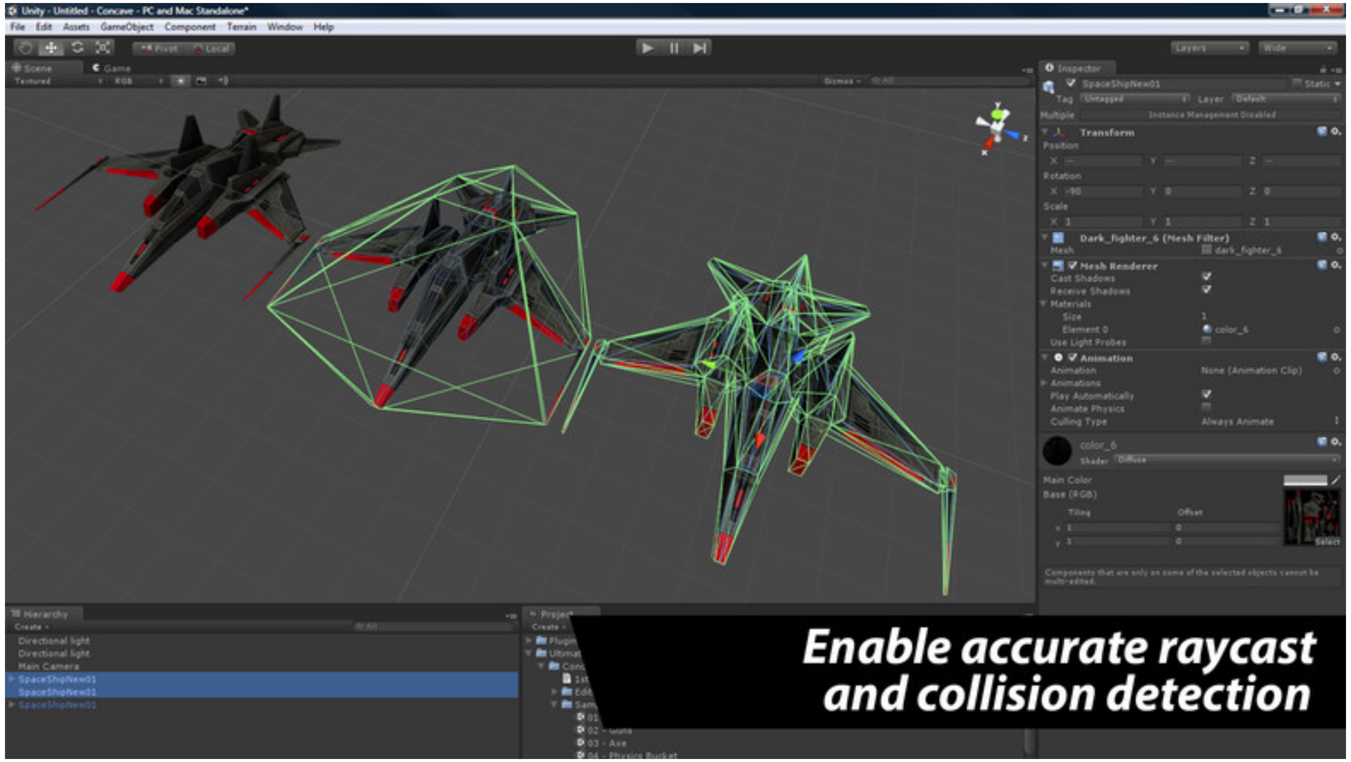
Colliders and Collisions

- How to accurately and efficiently find collisions between game objects?
  - Accurately – account for details of object shape
  - Efficiently – considering both time and space



Collider shapes

- Finding good approximation
  - Accurate enough
  - Fast
- If inaccurate
  - Ghost collisions
    - Bounding shape is too big, signals false collision
  - Bad physics
    - Collision pt at wrong place, angle
- Too accurate then slow

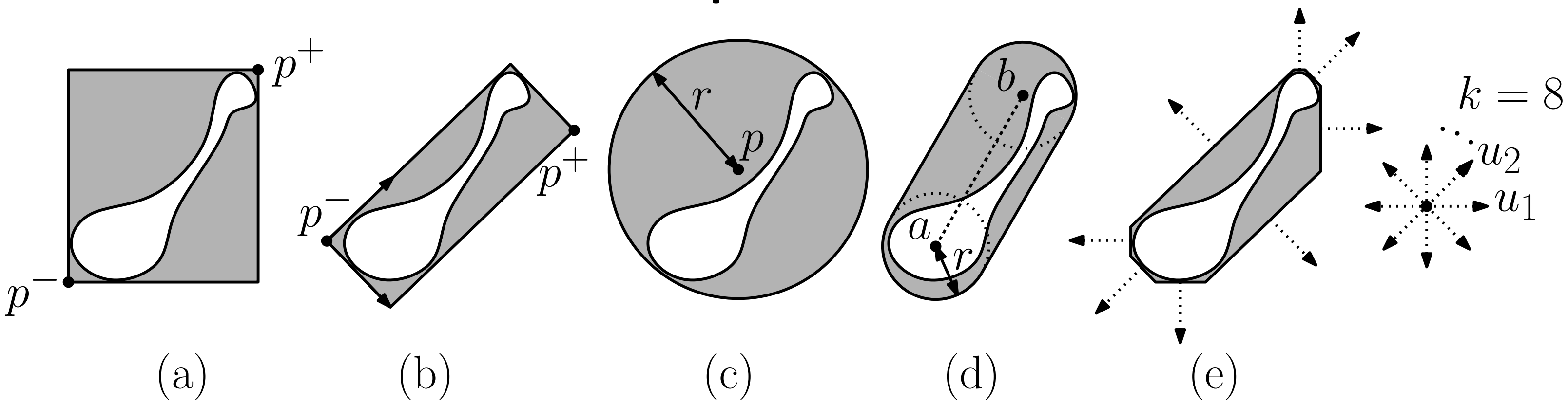


Enable accurate raycast and collision detection

How bound complex shape?

- How would you bound this shape?

Standard collider shapes



(a) Axis-aligned boxes (AABB)

(b) General bounding boxes

(c) Bounding spheres (ellipsoids)

(d) Capsules

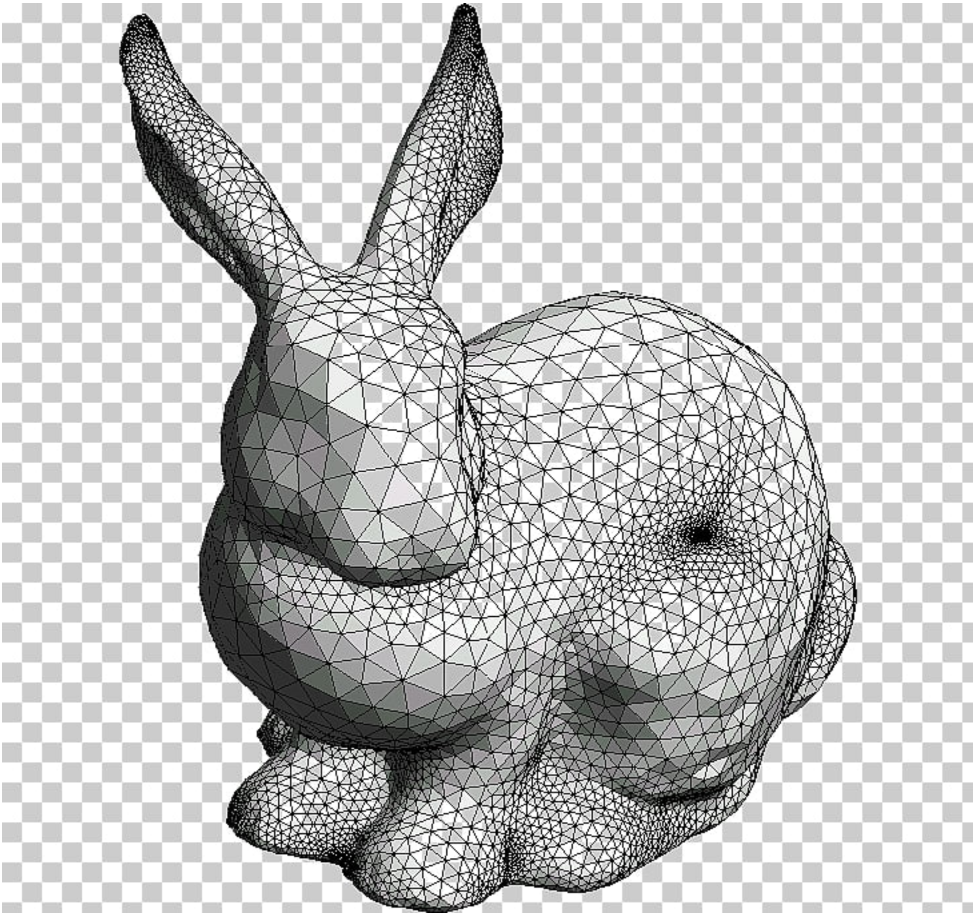
(e) k-DOPs (k-discrete oriented polytope)

Also – point, mesh,

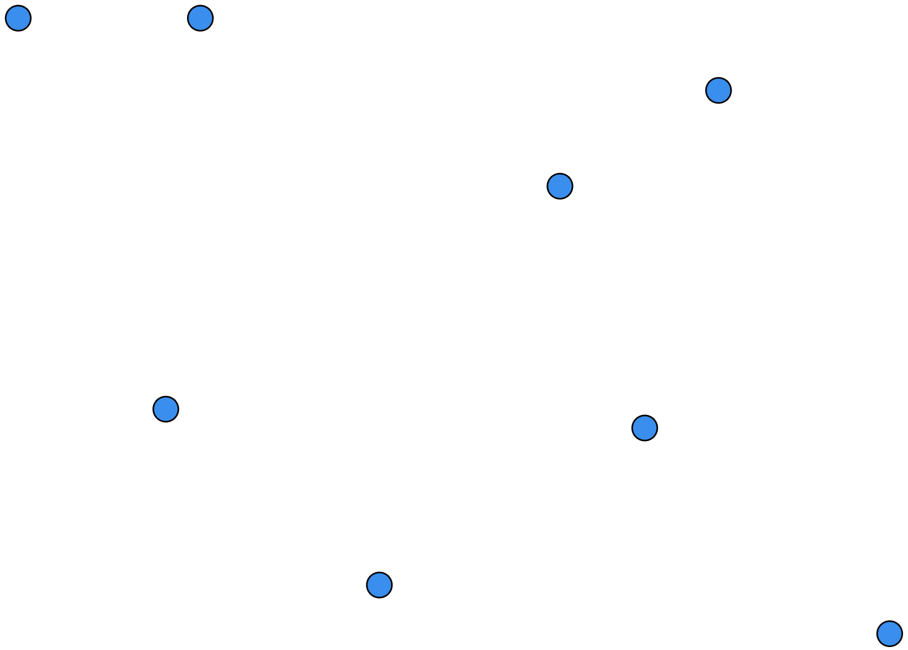
convex hull

What would you use?

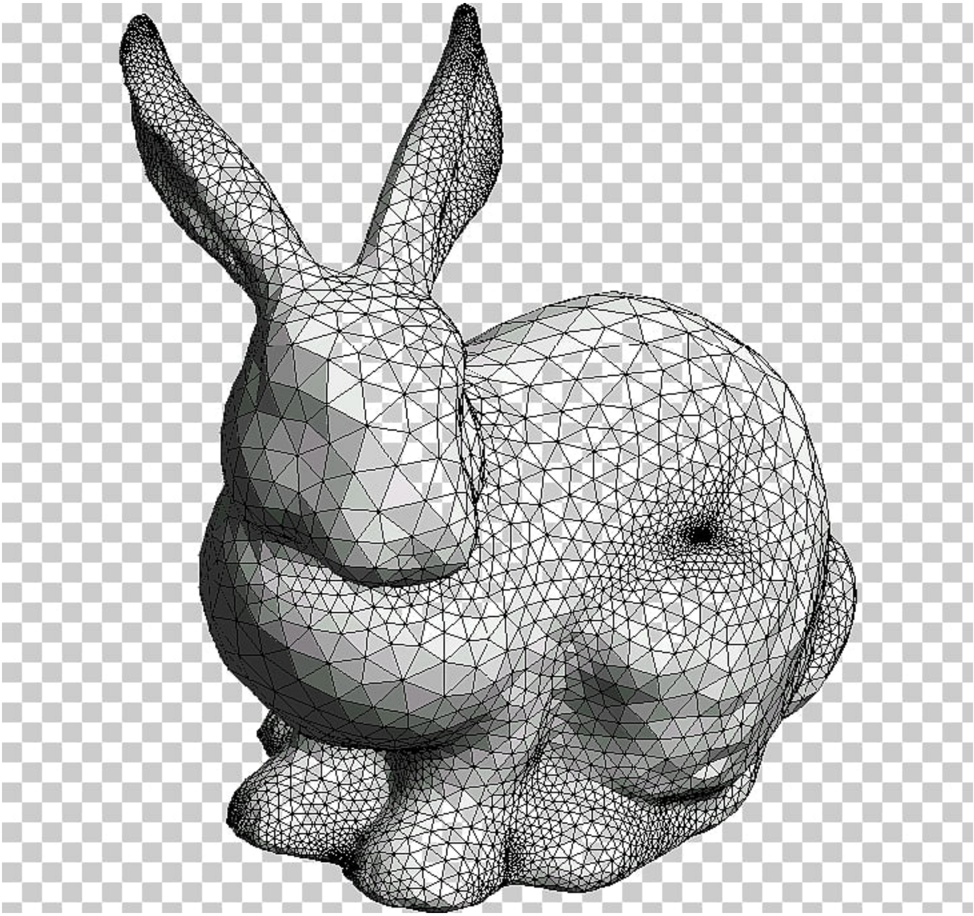
Fitting the collider



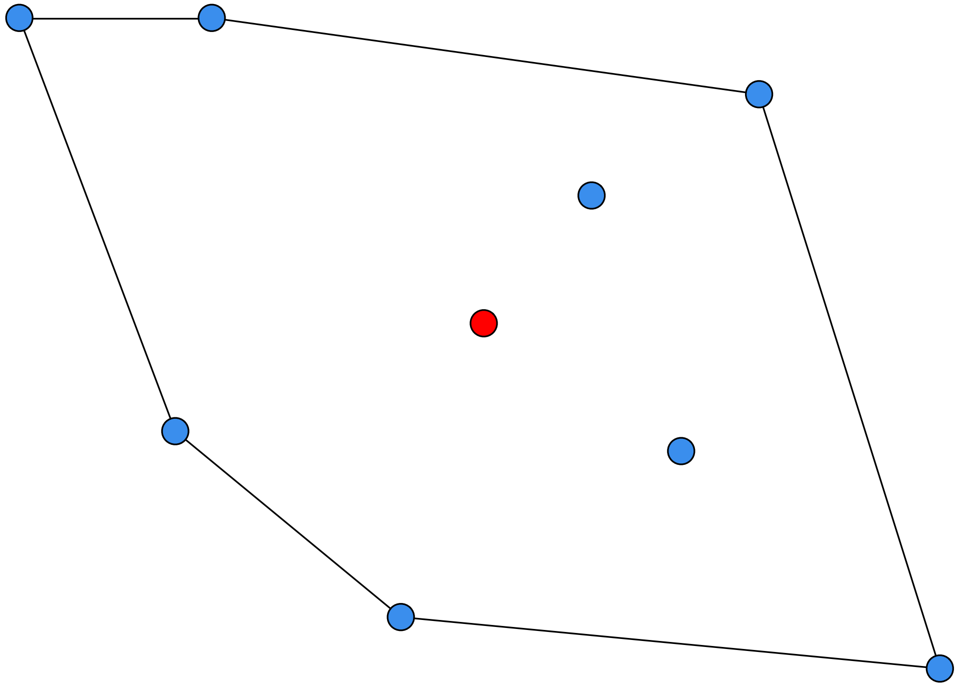
- Data is a set of points



Fitting the collider

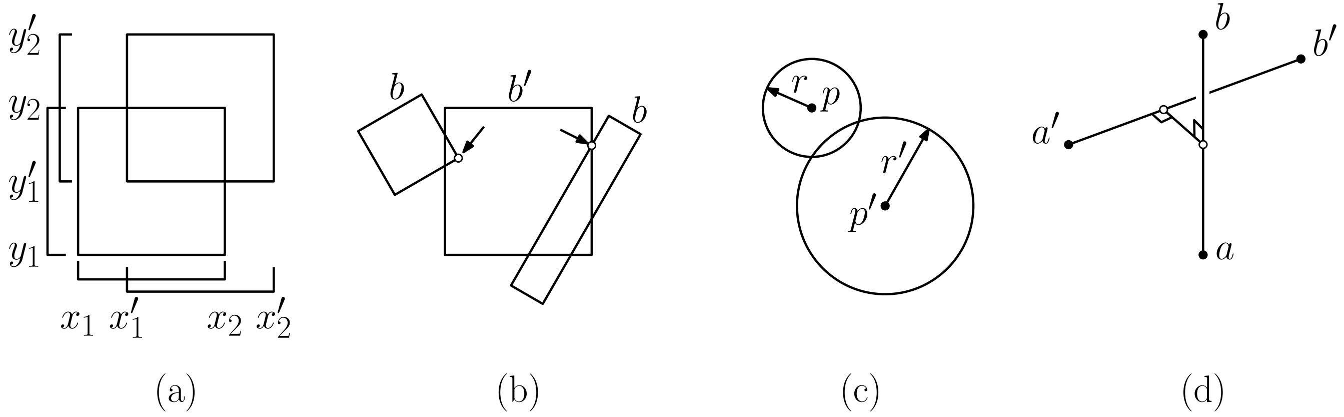


- Centroid and convex hull



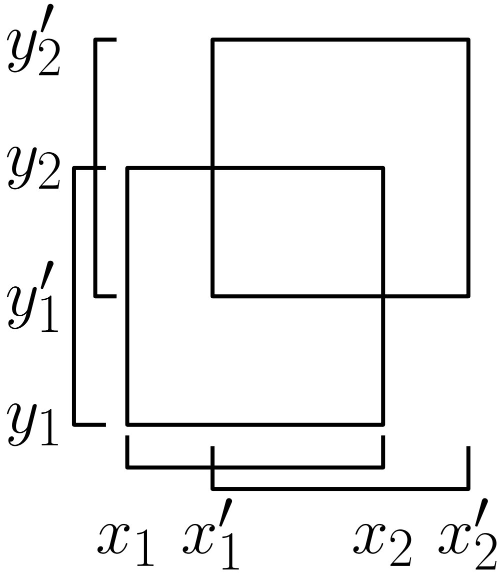
Detecting collisions – how?

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

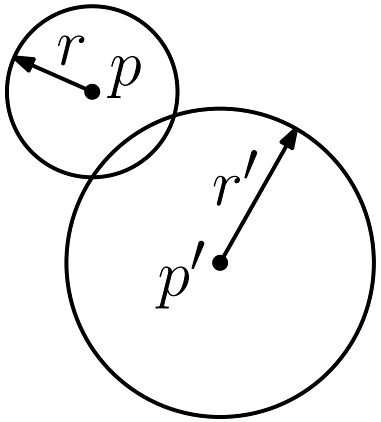


"Easy" cases

- AABB x AABB

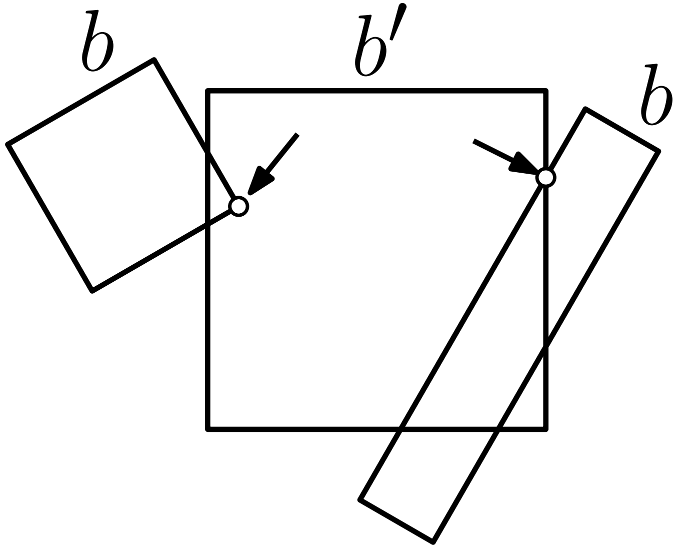


- Sphere x Sphere



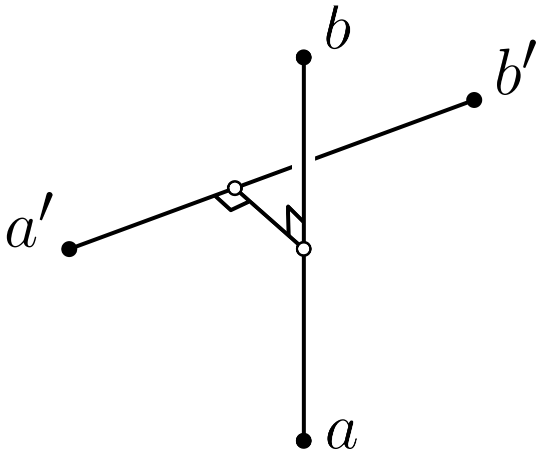
Box to box with rotations

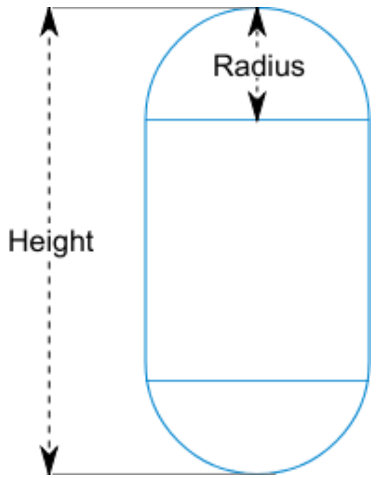
- Rotate one to align with axes

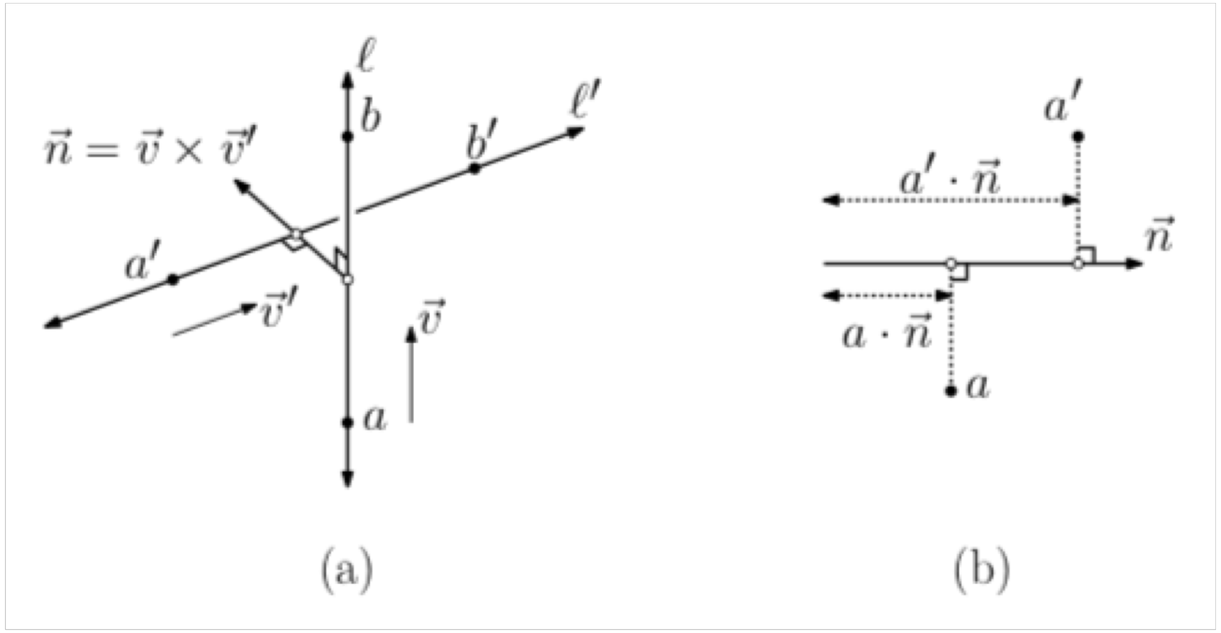


Capsule to capsule

- 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

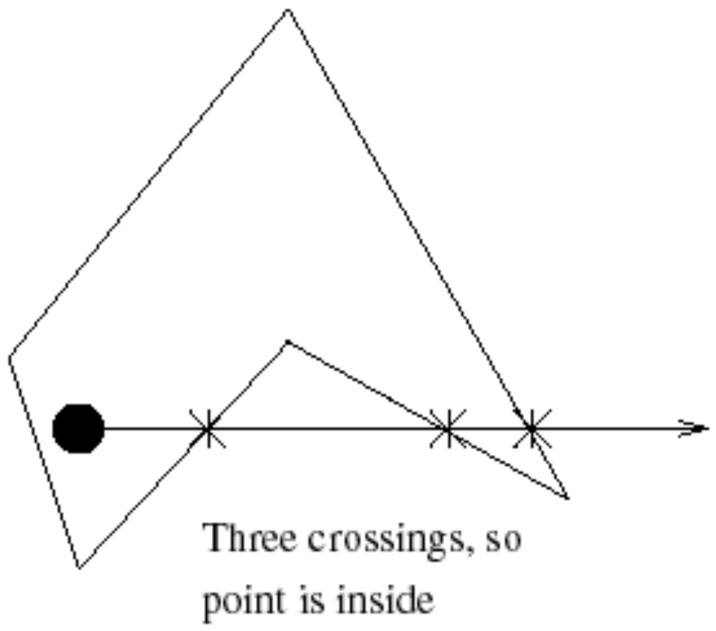
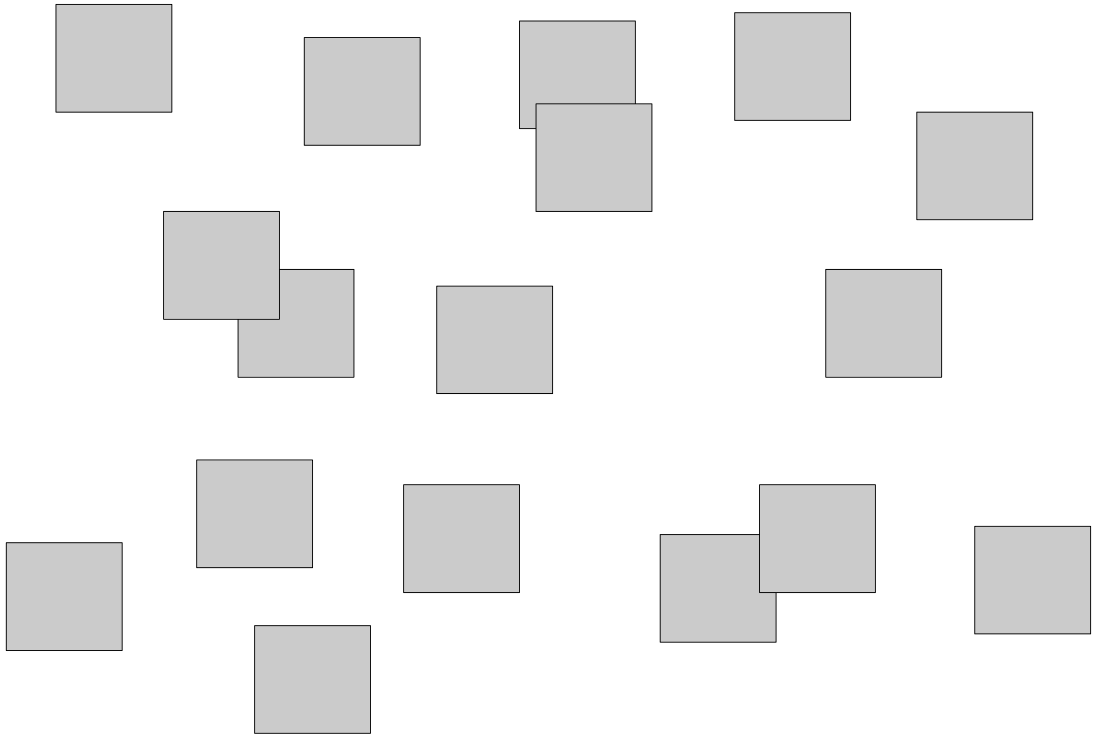


Figure 1 - Crossings Test

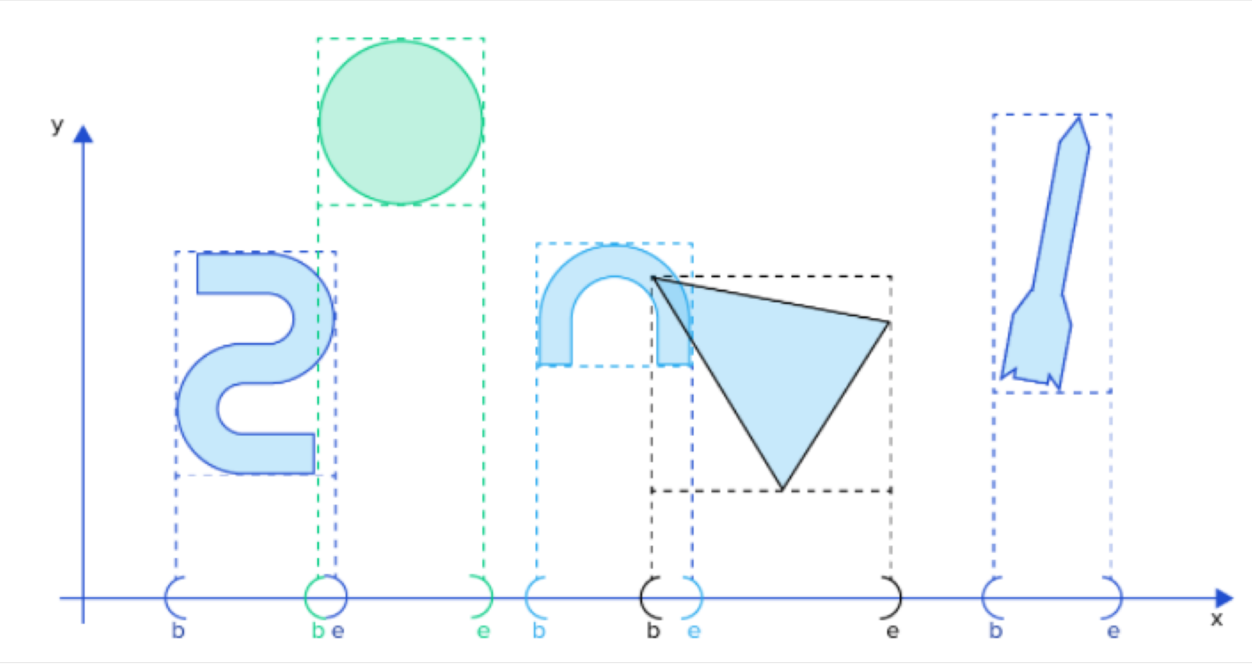
How to do many efficiently?

- 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

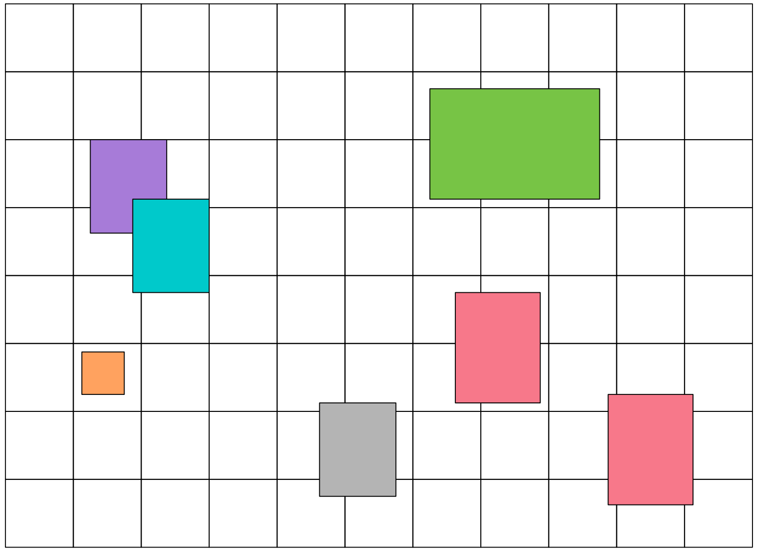


Sort and sweep algorithm

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

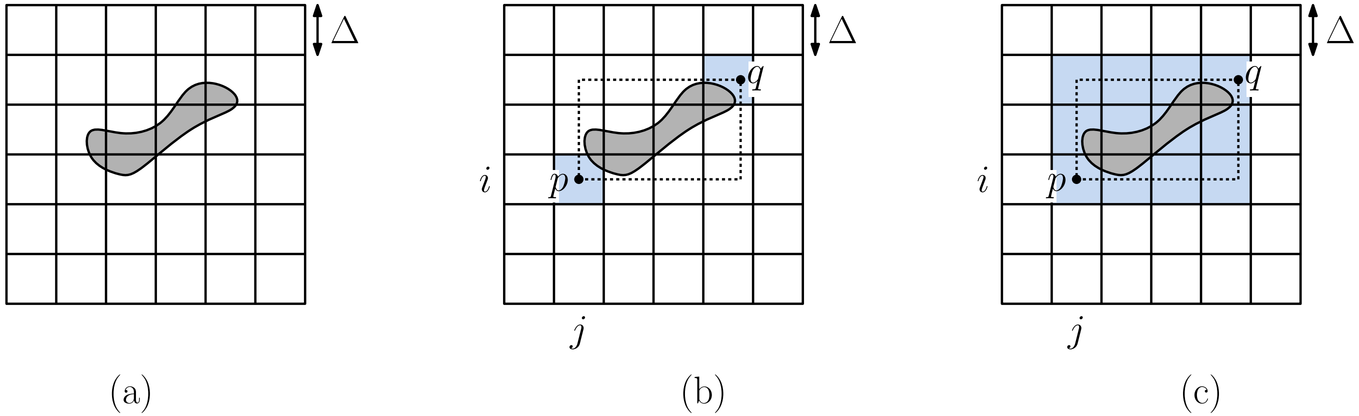


Grid



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

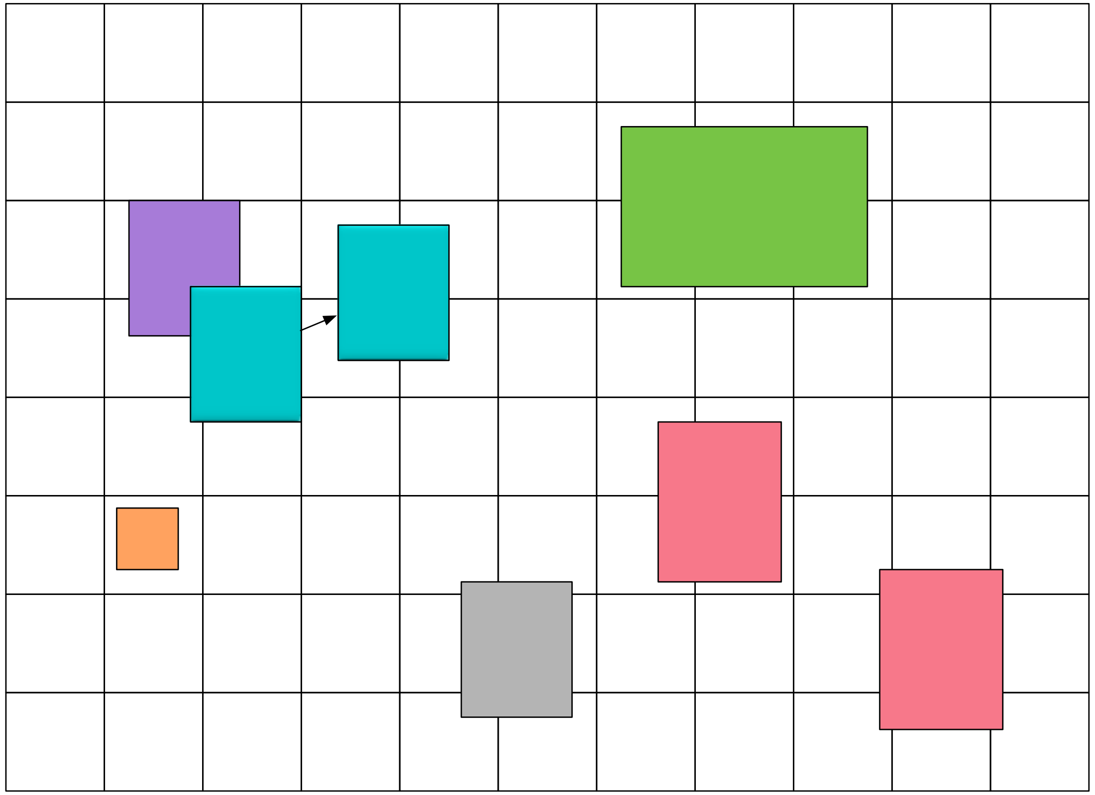
- If two shapes in same cell then need further test



- What size grid?
- How update grid?

Grid

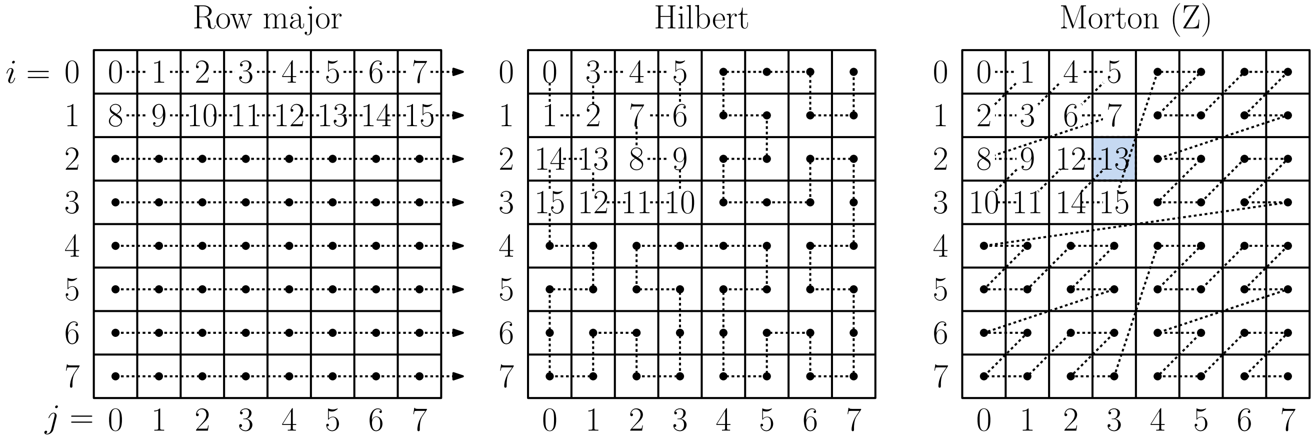
- How treat moving and static objects?
    - One agent in static space?



How store grid?

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

- Bit shuffle for Morton’s
  - See notes



Row major

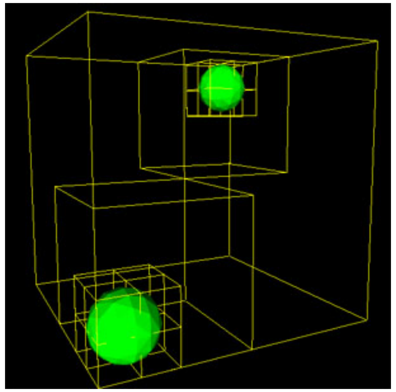
Hilbert

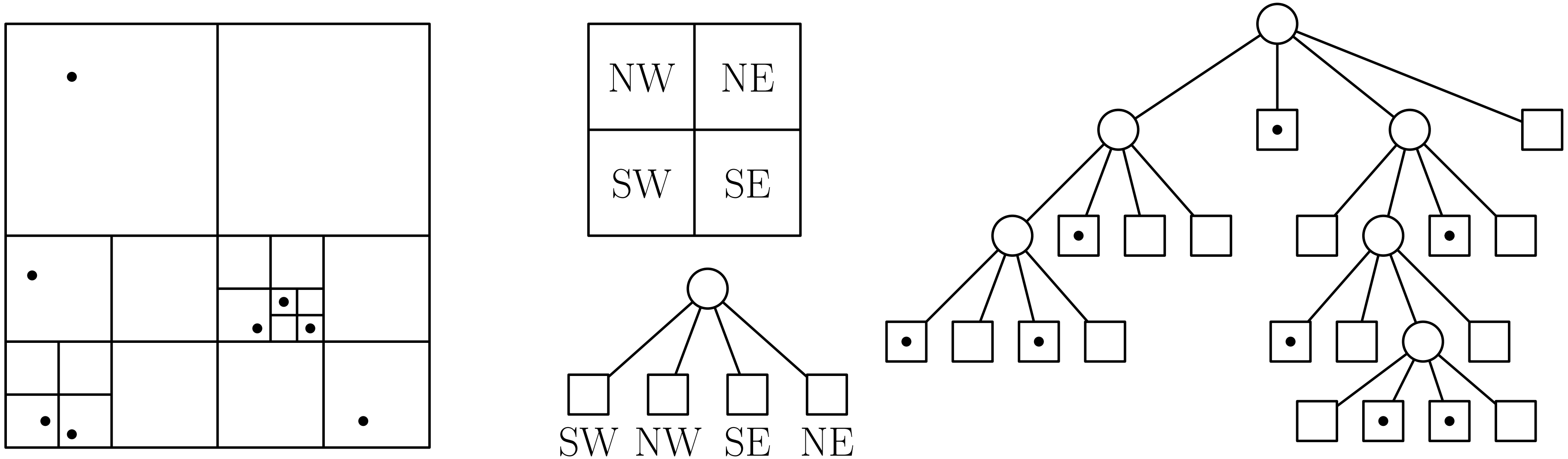
Morton (Z)

(a) (b) (c)

Quadtrees: hierarchical space decomposition

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





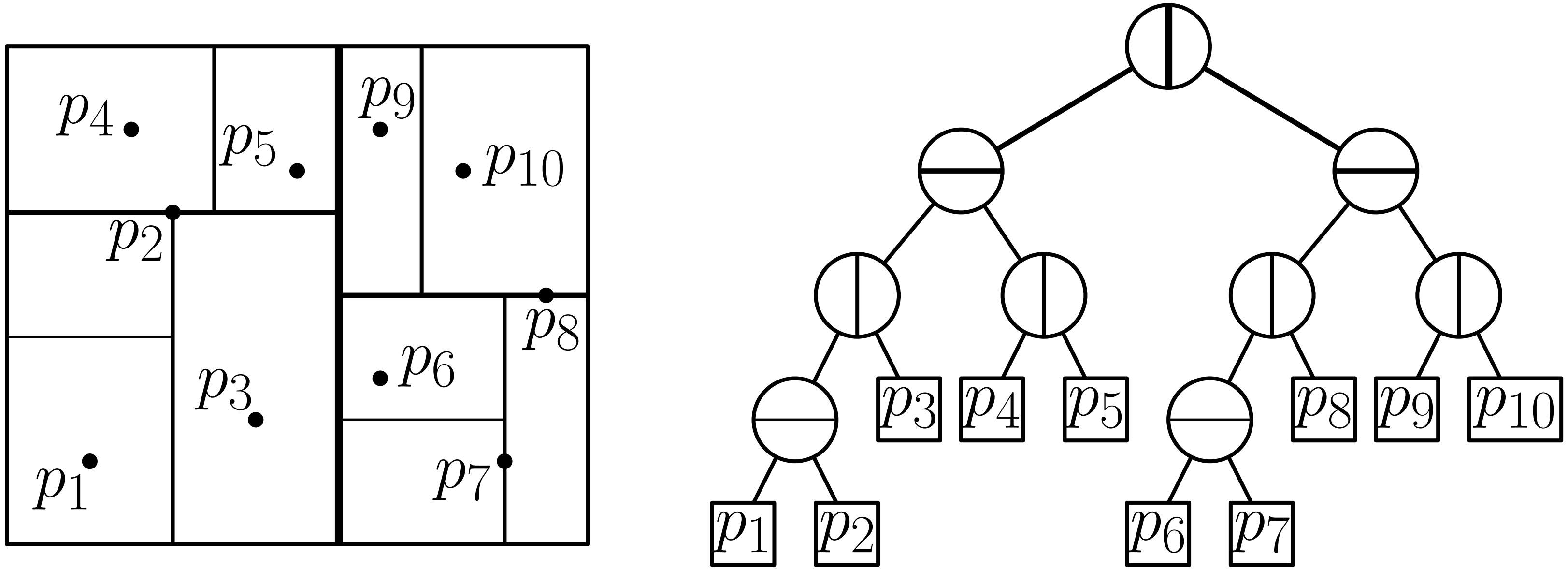
(a)

(b)

(c)

K-d trees

- Alternating coordinates
- Divisions based on data



Readings

- David Mount's lectures on Geometric problems, and on Geometric Data Structures

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