Colliders and Collisions

CMSC425.01 Spring 2019

Still at tables …

Administrivia

- Hw1 due. Questions?
- Final project proposal. Questions?
- Project 1b submission issues
- Mini-lectures coming – videos on single topics (Panopto on Elms)

Student vs. professional answers

- For classes, just do enough for the grade
- For professional use, need to do more
  - Demonstrate answer to rest of team
  - Verify (test) solution by hand and by computer
  - Make sure it is most efficient (or at least efficient enough)
- This class: get closer to professional answers
- Intellectual property issues
  - Students can't plagiarize but can use assets under fair use (unpublished work)
  - Professionals can plagiarize but can't violate copyright or patent

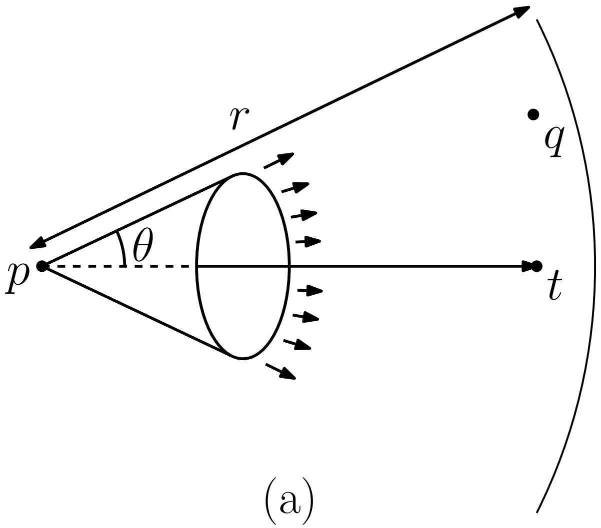
Today’s questions

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

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

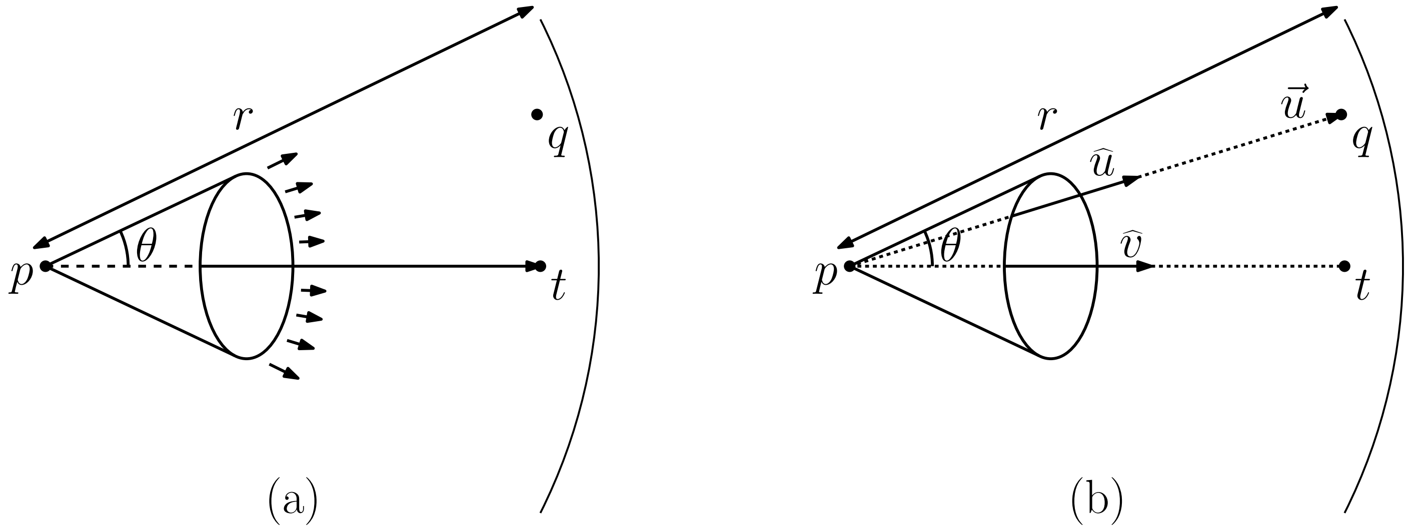


(a)

Problem 1: Shot gun weapon

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

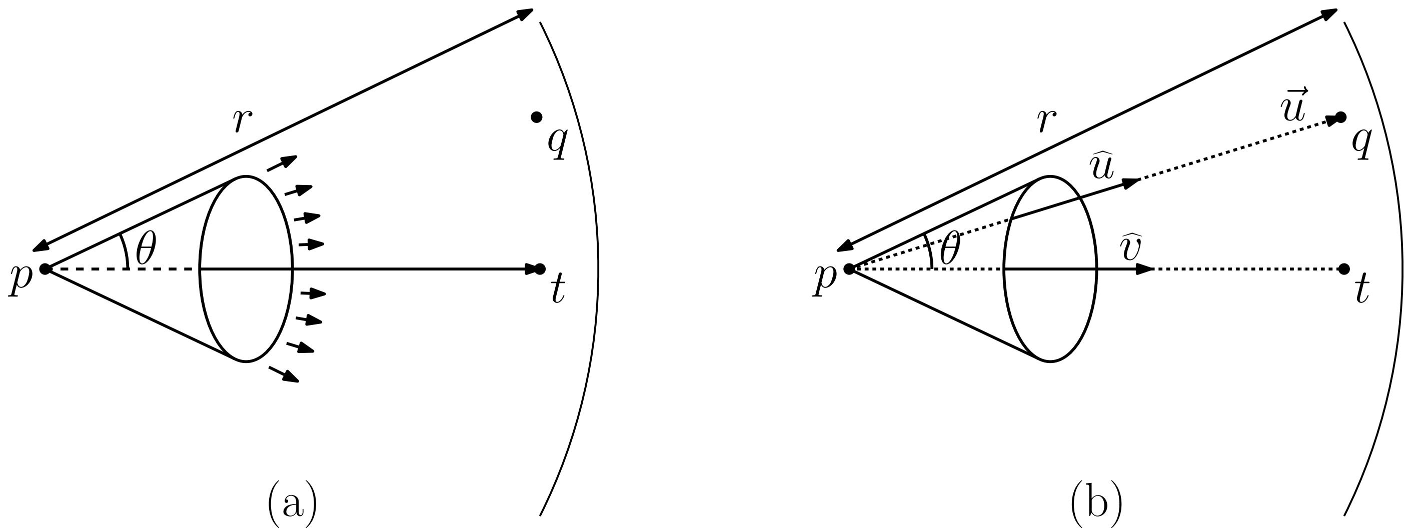
- Return true if q hit



(a) (b)

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

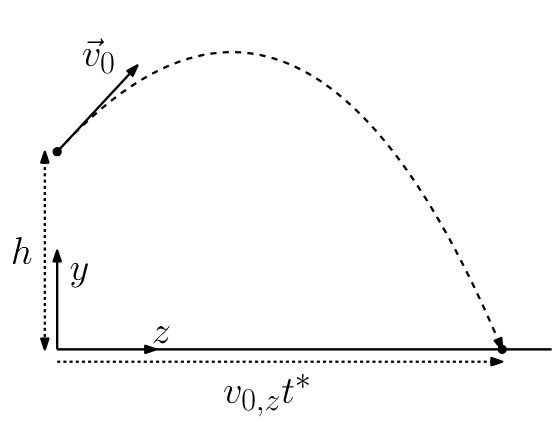


$$
\begin{aligned}
\vec{v} &\leftarrow t - p; \qquad \vec{u} \leftarrow q - p \\
\ell(v) &\leftarrow \|\vec{v}\| = \sqrt{\vec{v}\cdot\vec{v}}; \qquad \ell(u) \leftarrow \|\vec{u}\| = \sqrt{\vec{u}\cdot\vec{u}} \\
\widehat{v} &\leftarrow \text{normalize}(\vec{v}) = \vec{v}/\ell(v); \qquad \widehat{u}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

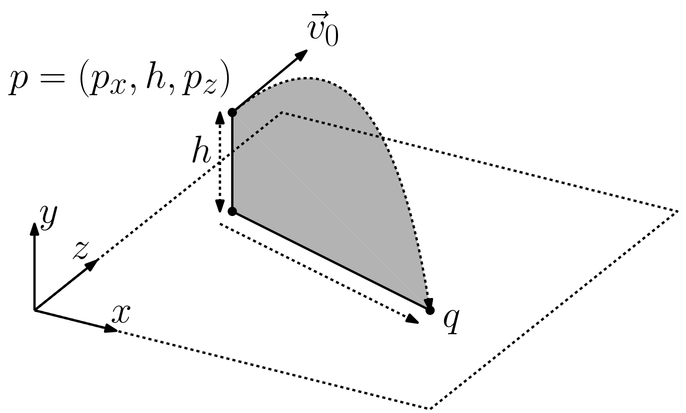
Problem 2: Projectile aiming tool

- 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)

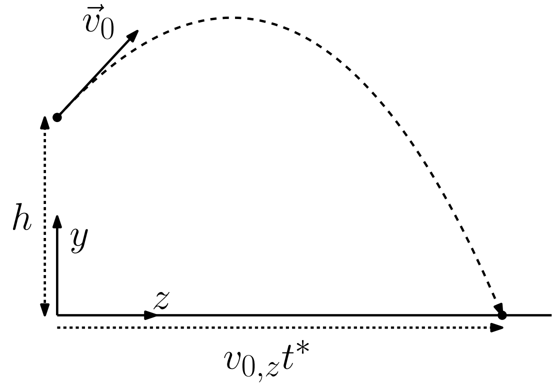
Problem 2: Projectile aiming tool

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

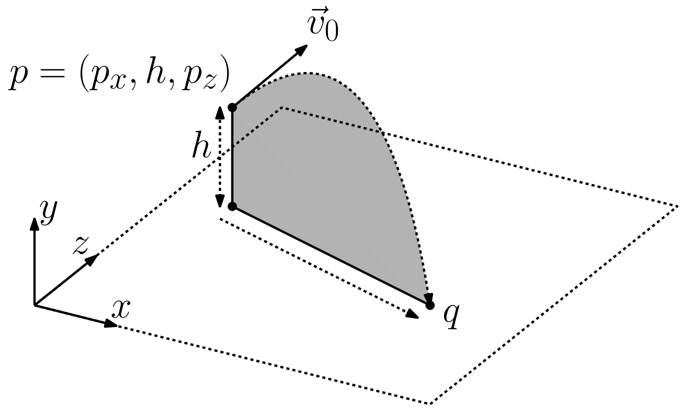
$$z(t) = v_{0,z}t \quad \text{and} \quad y(t) = h + v_{0,y}t - \frac{1}{2}gt^2.$$

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

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



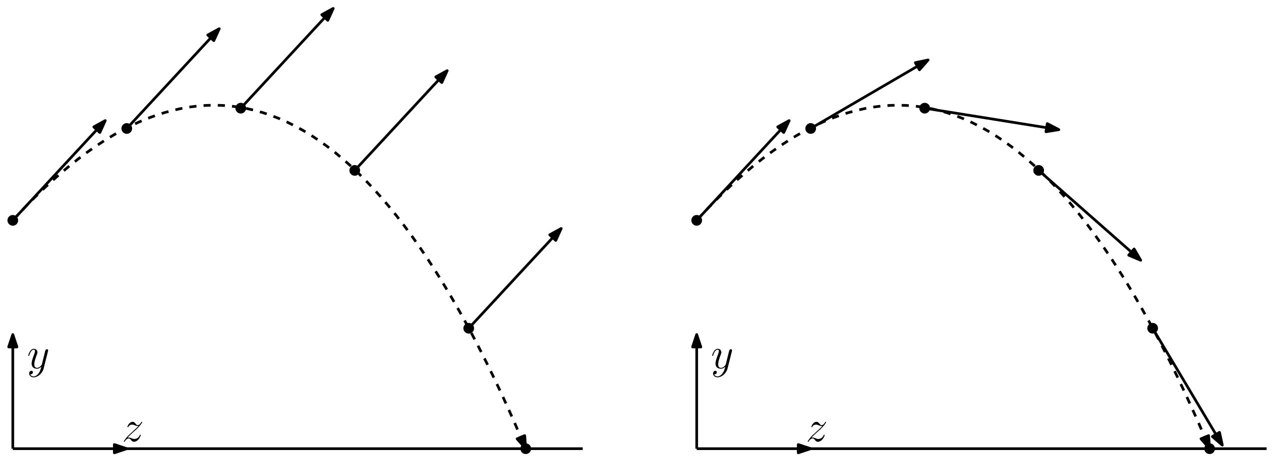
(a)



(b)

Problem 3: Shooting an(d) arrow

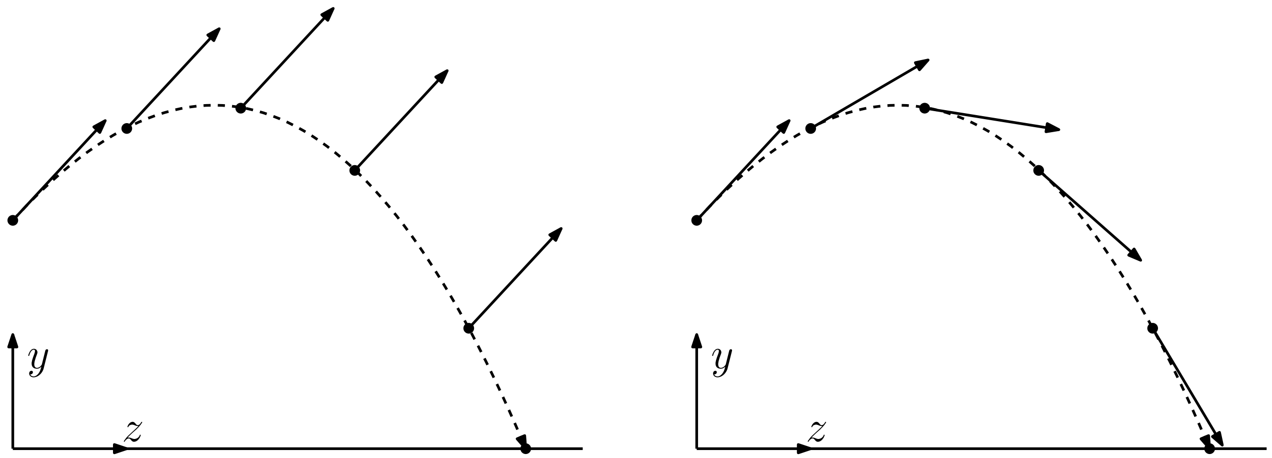
- Problem:
- If projectile show direction (eg, arrow)
  - Initial location (0,h,0)
  - Initial velocity $\vec{v}_0 =< v_{0,x}, v_{0,y}, v_{0,z} >$
- Find direction orientation
  - Location (x,0,z)



(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)

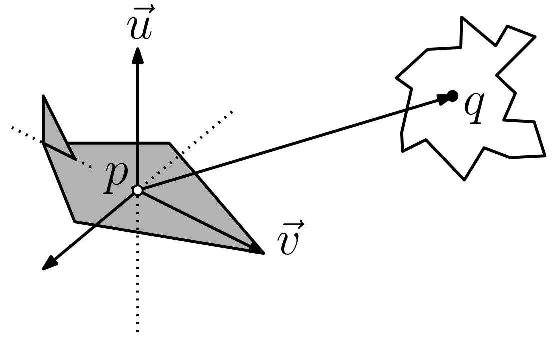


(a) (b)

```
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

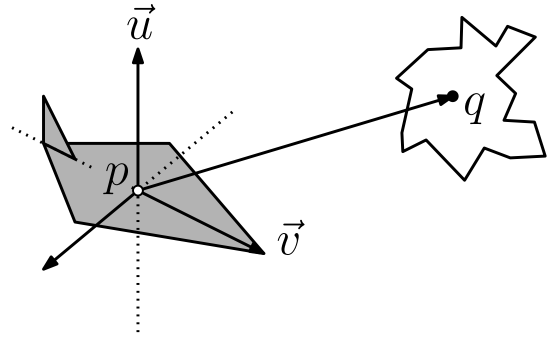


(a)

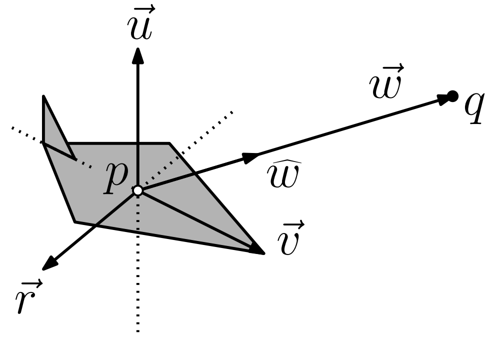
- Determine if ship should evade
  - Turning up or down
  - Turning left or right

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



(a)

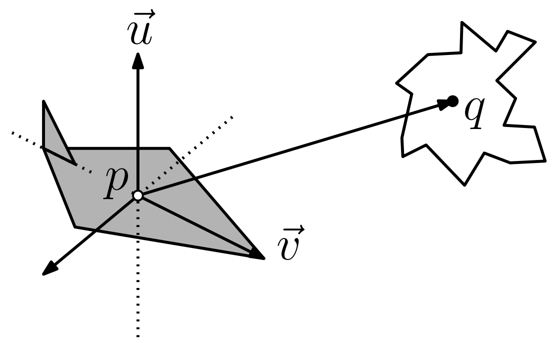


(a)

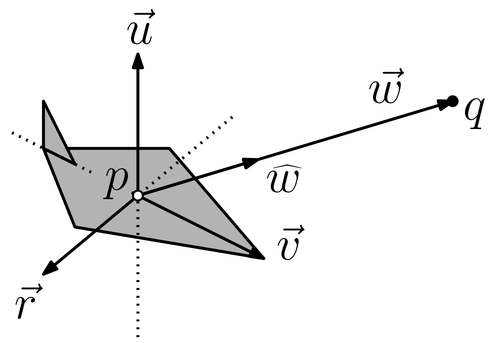
- Determine if ship should evade
  - Turning up or down
  - Turning left or right

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



(a)



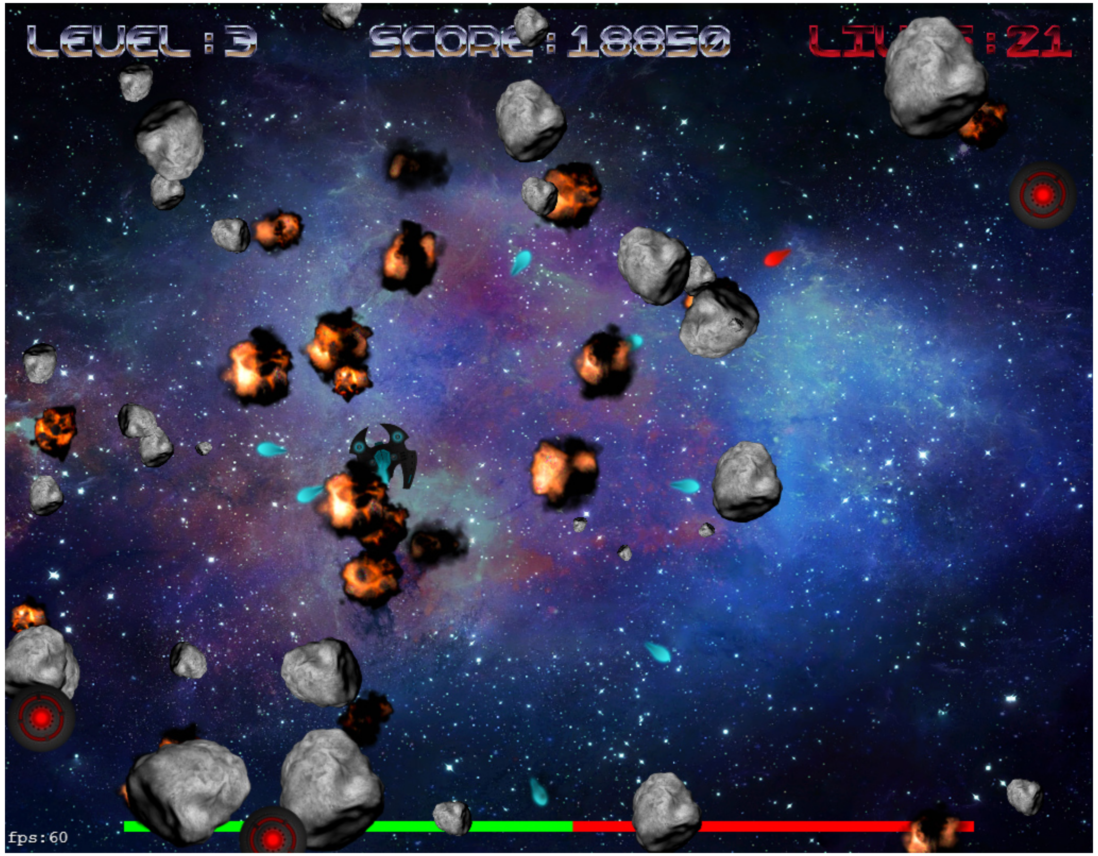
(a)

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

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

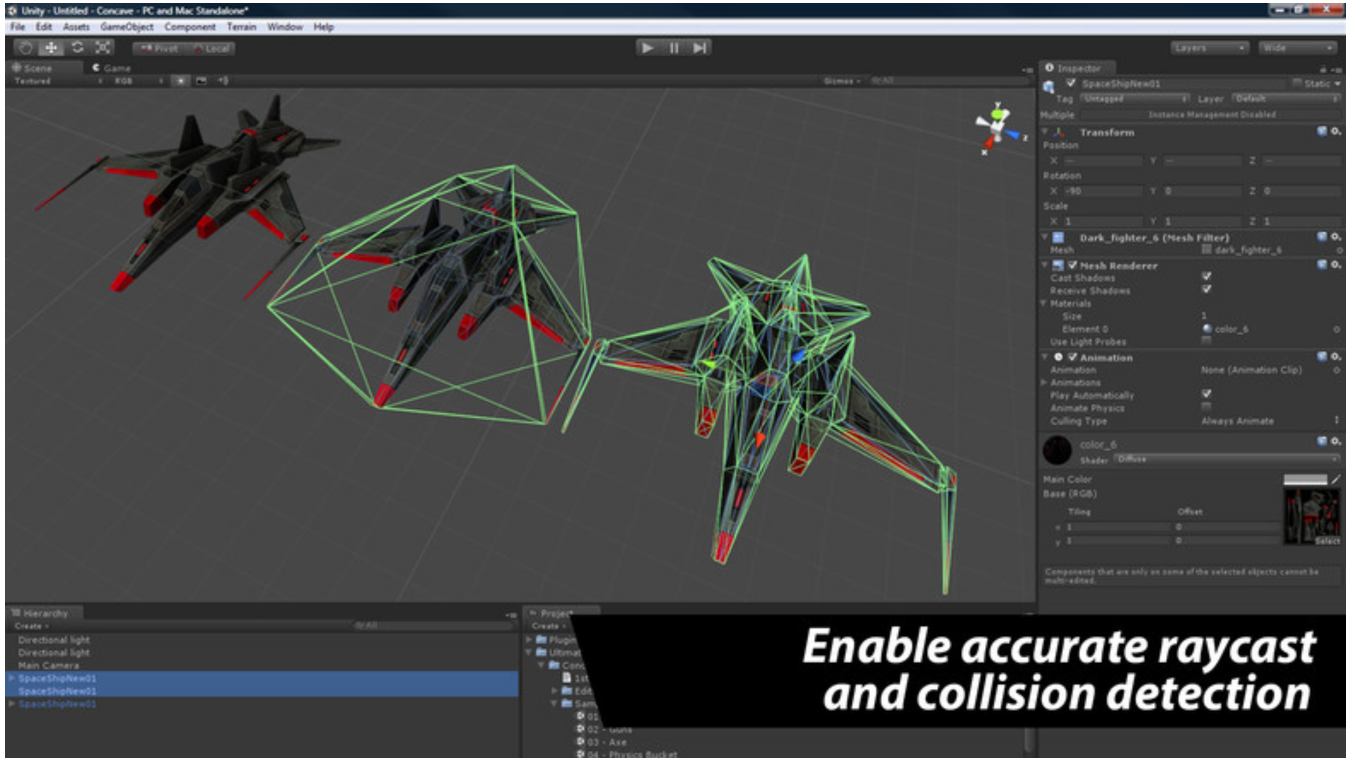
Colliders and Collisions

- 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

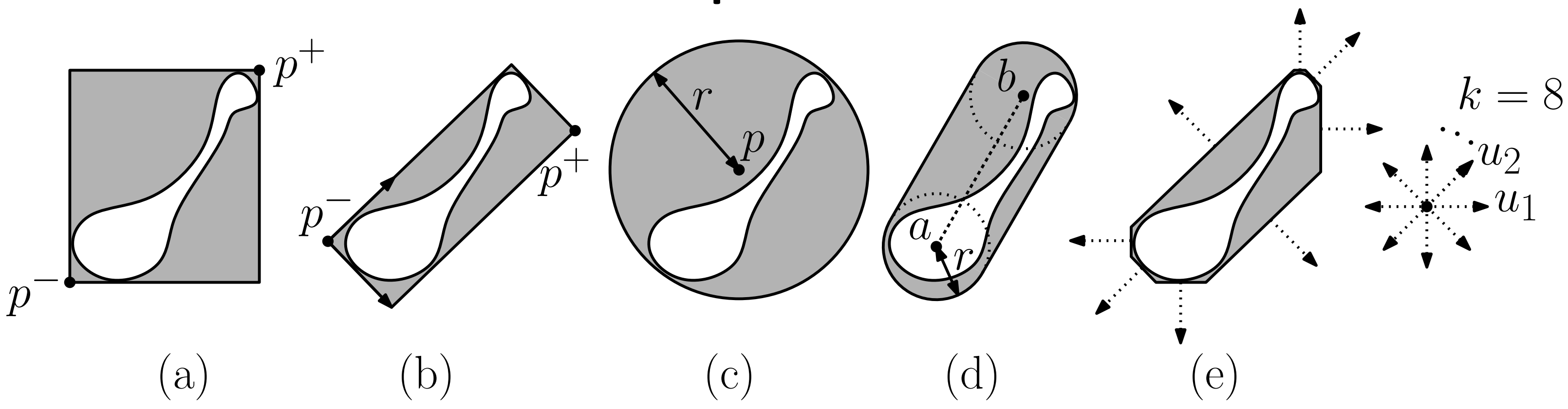
- Finding good approximation
  - Accurate enough
  - Fast
- If inaccurate
  - Ghost collisions
    - Bounding shape is too big, signals false collision
  - Bad physics
    - Collision pt at wrong place, angle
- Too accurate then slow



How bound complex shape?

- How would you bound this shape?

Standard collider shapes

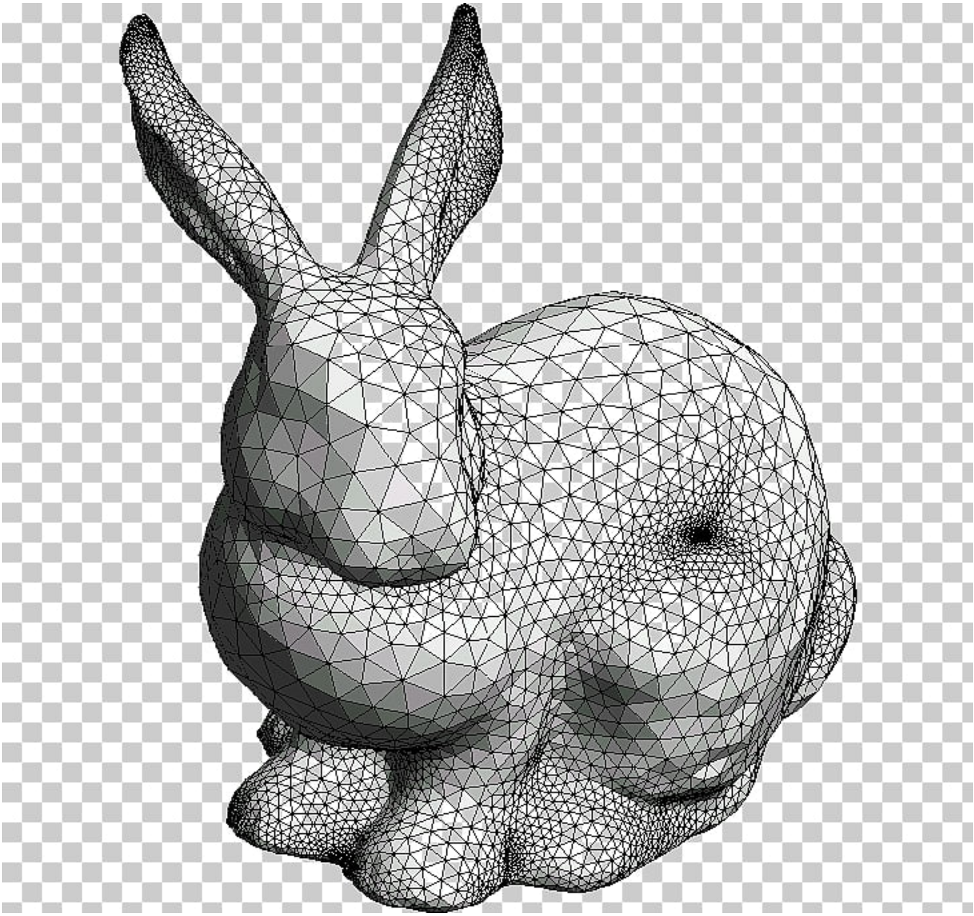


(a) Axis-aligned boxes (AABB)
(b) General bounding boxes
(c) Bounding spheres (ellipsoids)
(d) Capsules
(e) k-DOPs (k-discrete oriented polytope)

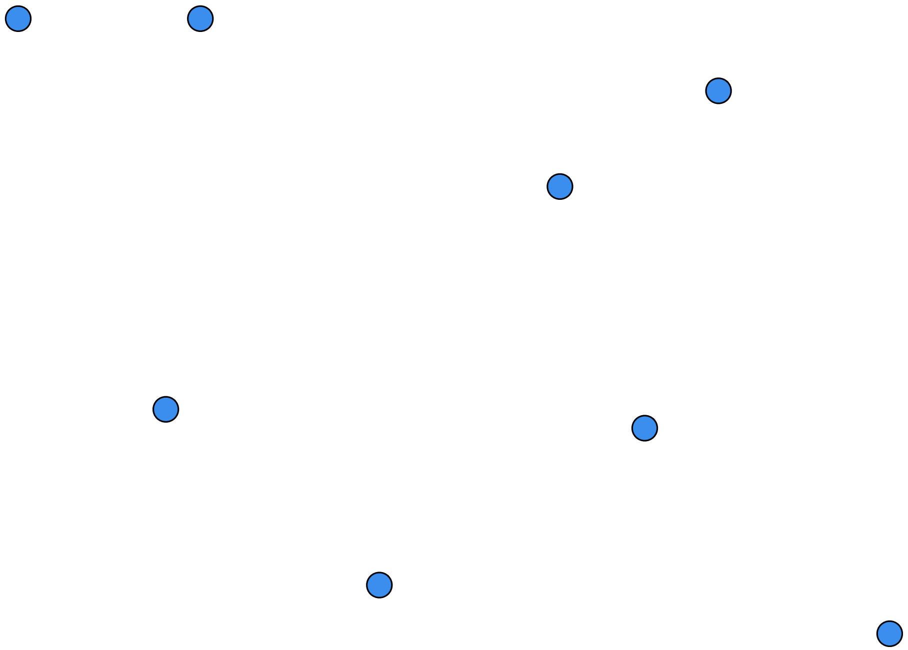
Also – point, mesh, convex hull

What would you use?

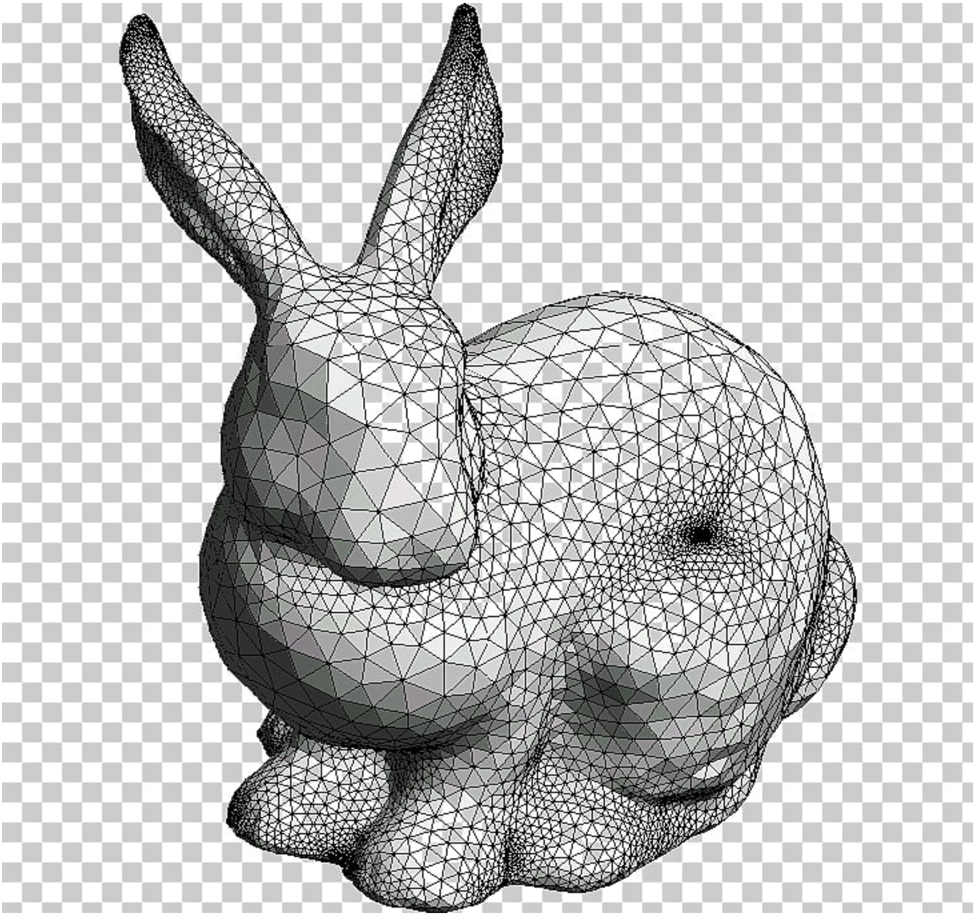
Fitting the collider



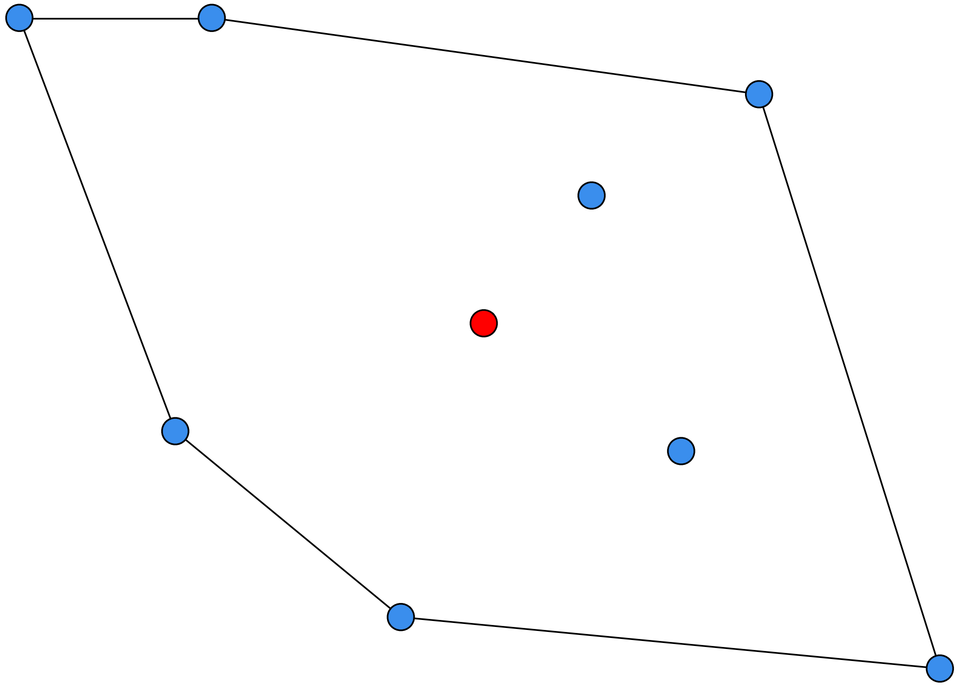
- Data is a set of points



Fitting the collider

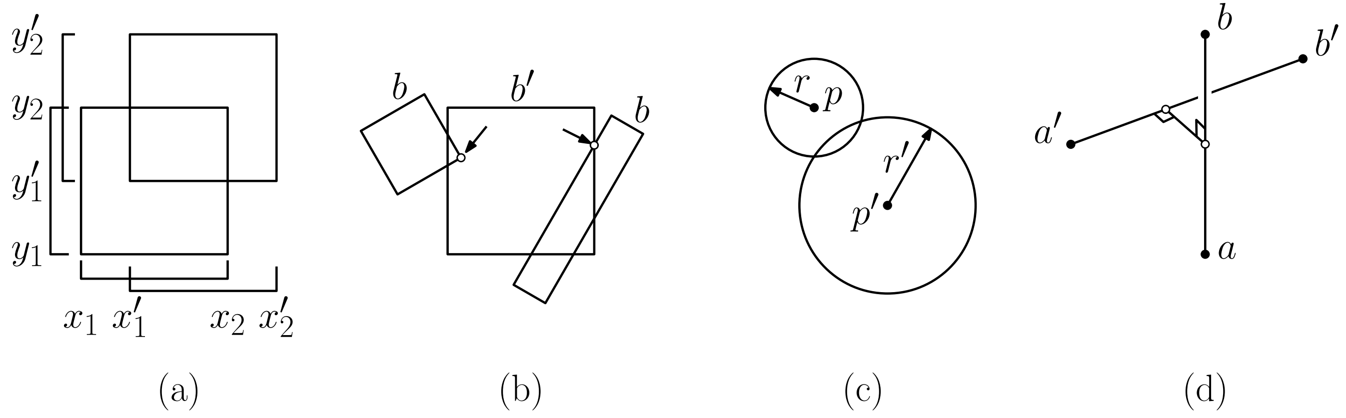


- Centroid and convex hull



Detecting collisions – how?

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



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