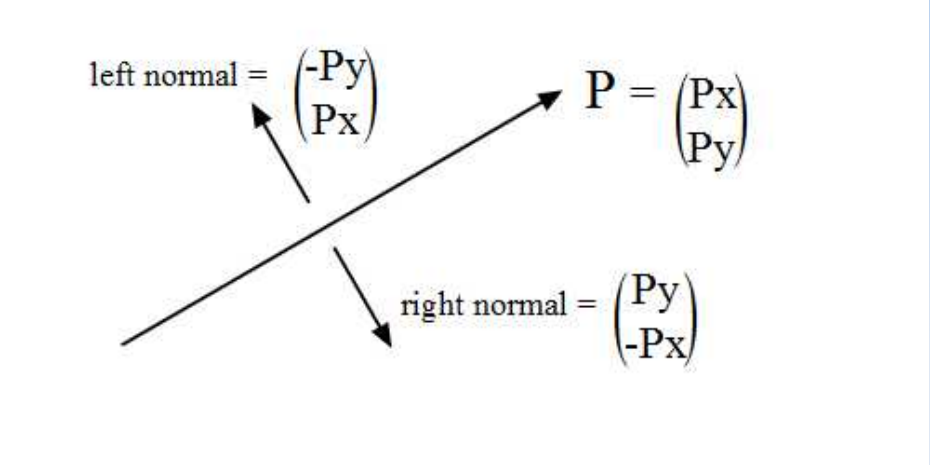
CS 134

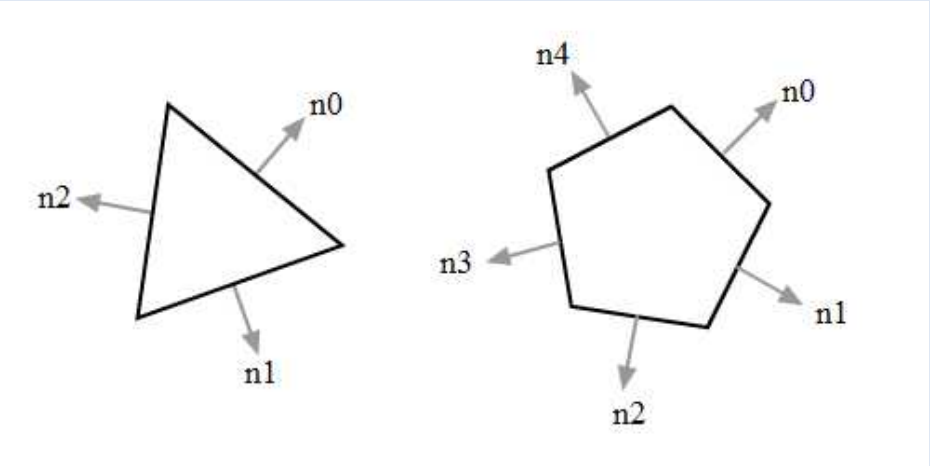
Background Shapes &
Collision Resolution

(Top-down and Side-scrolling)

Polygon / Polygon

- Create vectors for each side, see if all points of one polygon are on the “outside” half of the world.
  - These are called normal vectors





Physics Update Overview

- So far:
- Collision Detection
  - (sprite / sprite)

- To cover:
- Collision Detection
  - (sprite / background)
- Collision Resolution
- Advanced Actor Motion

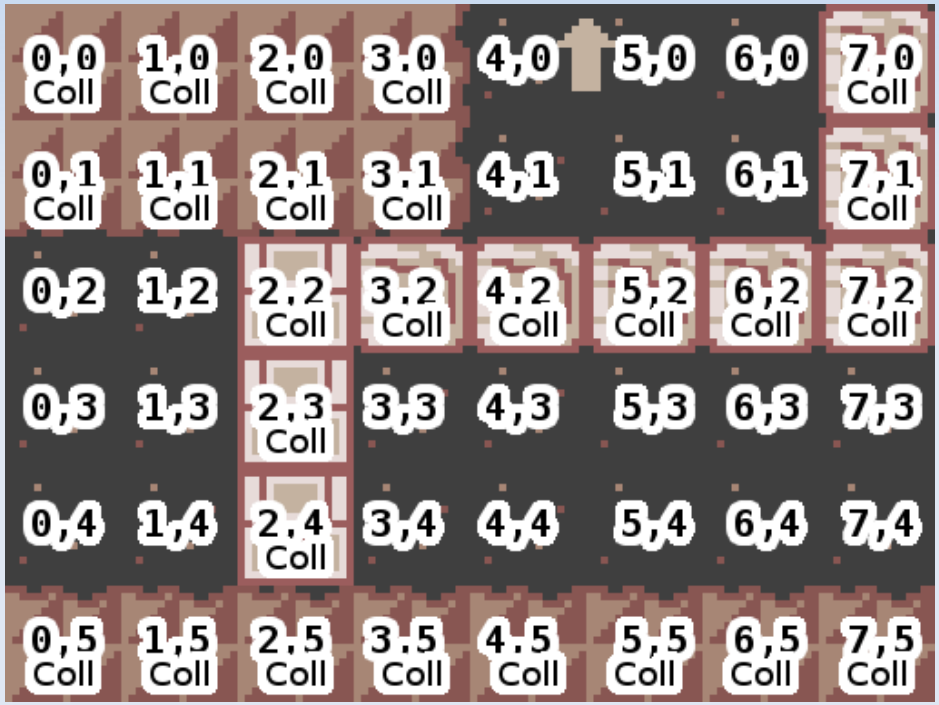
Backgrounds

- Recall how we stored the tile positions in an array?
- int[] bg;
- Collision can go there too!

Backgrounds

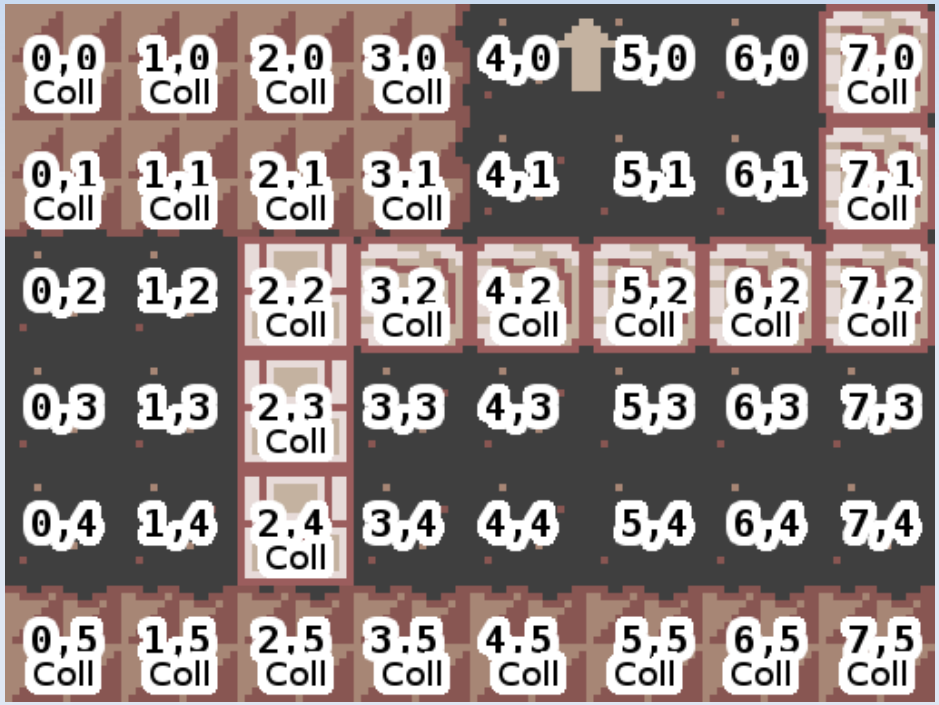
- class Tile {
  int image;
  boolean coll;
  };
  Tile[] bg;

- Other information can go in a tile as well!



Backgrounds

- As the player moves, you will need to collide against all the background shapes.
  - AABB / AABB
- Use the camera drawing optimization
  - Start: pX / tileW
  - End: (pX + pW) / tileW
  - Similar in Y



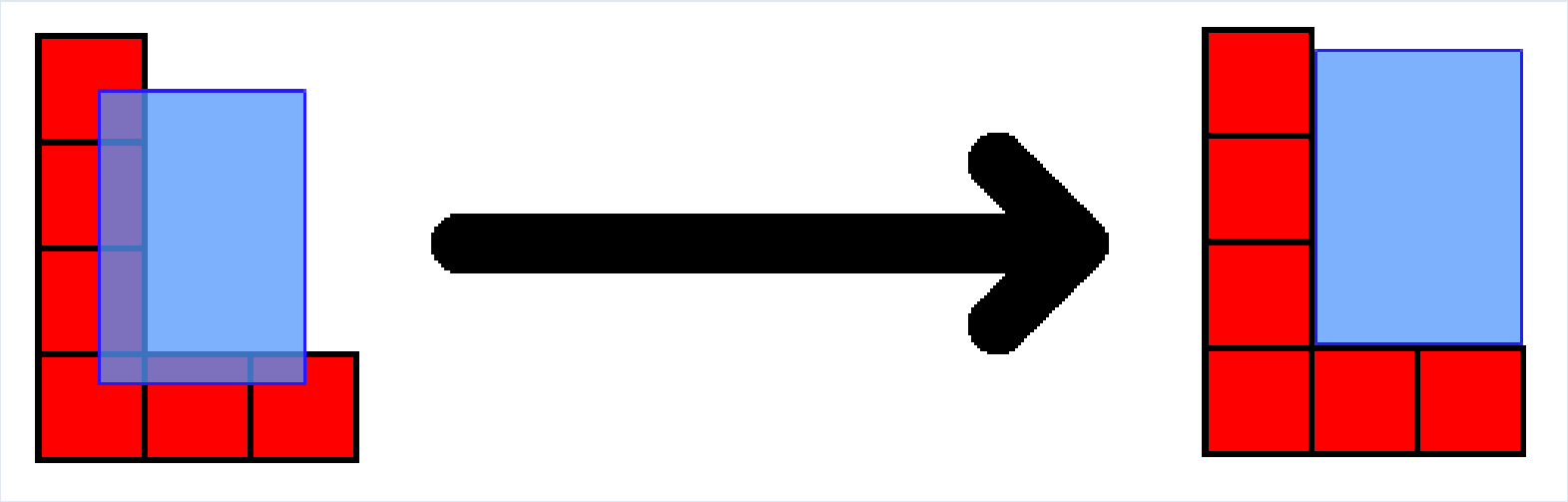
Backgrounds

Questions?

(Remember, this is just for DETECTING collisions)

Collision Resolution

- The job of collision resolution is to “fix” the simulation state so that it is accurate again.



Collision Resolution

- For now, assume a simple top down game
  - Up / Down / Left / Right move player
  - Background is “block shaped”

Collision Resolution

- Simplest possible collision resolution is with bullets
  - Deal damage to player
  - Remove bullet
  - Check if player is dead

Collision Resolution

- For colliding with walls
  - Simple solution: Back up player to prev position
  - More complex: Back up player to when the collision starts
- This may be enough for some games!

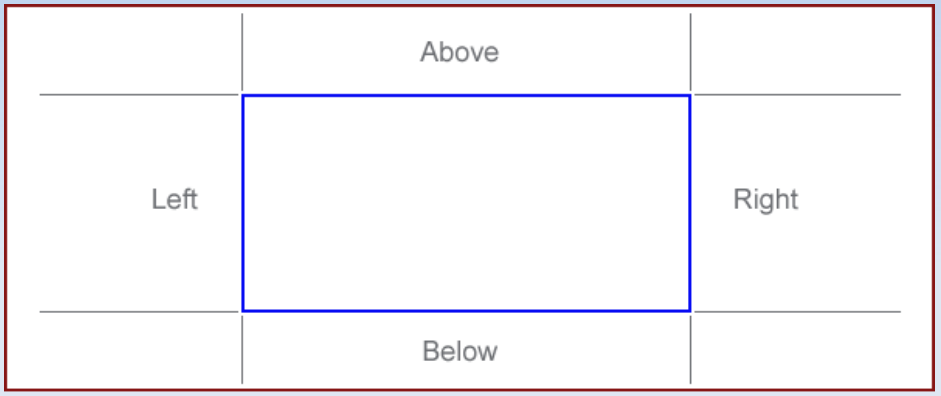
Collision Resolution

- Best feeling solution: Back up on each axis
  - Back up on X axis by overlap with wall tile
  - Back up on Y axis by overlap with ceiling tile
- This allows smooth sliding along walls

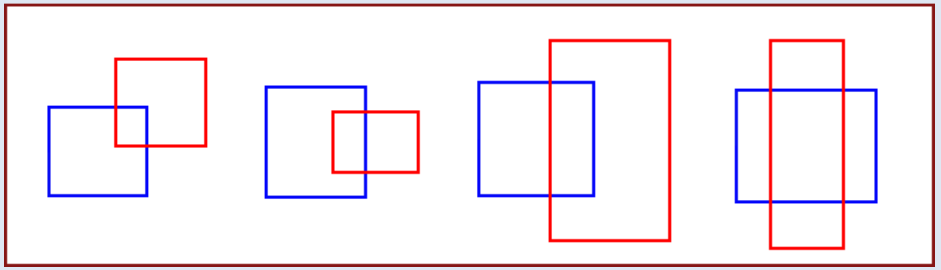
- Also, can do things like wall jumps, pushing, etc. by remembering which axes you collided with prev frame
  - More state for a player!

Collision Resolution

- How do you know the overlap?
  - Collision is just returning true / false!



- right1 < left2 is false, return right1 – left2
  - Similar for other sides



Collision Resolution

Questions?

Raycasting

- Recall: Physics is running at a fixed rate
- Collision is only checked AFTER objects move
- Problem:
  - Fast moving objects can jump past an object
- Raycasts shoot an infinitely long ray to collide with AABB / Circles

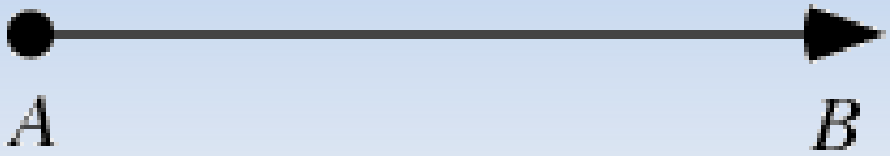
Raycasting

WARNING! WARNING!
INCOMING MATH

Ray / Circle

- Ray formula:

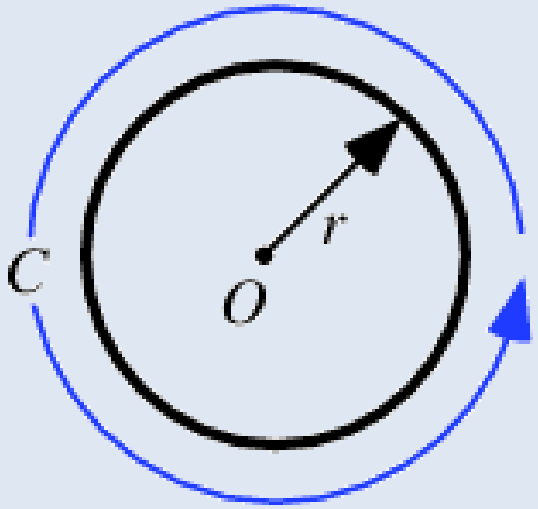
$$p = \boldsymbol{A} + t\,\boldsymbol{B}\,(t \geq 0)$$



- Circle formula:

$$r = |p - \boldsymbol{O}|$$

$$r = \sqrt{(p - \boldsymbol{O}) \cdot (p - \boldsymbol{O})}$$



Ray / Circle

- Plug ray into circle formula and simplify:

Ray / Circle

- Three possibilities (solutions for t):
  - t has no solutions
    - Ray does not intersect the circle
  - t has one negative, one positive solution
    - Ray starts inside the circle, collides at positive solution
  - t has two positive solutions
    - Ray hits circle, use the lesser solution

Ray / Circle Optimization

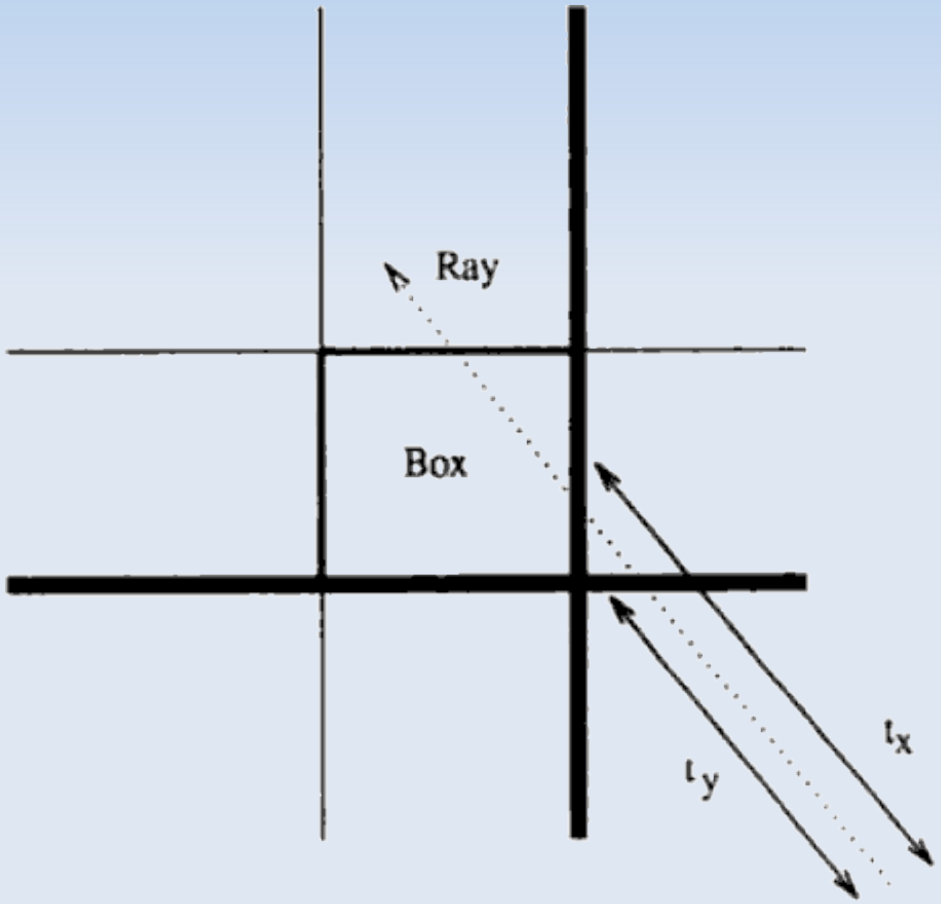
- Remember, we don't care if ray starts inside of circle!
- Calculate t via:

$$t = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

- Collision is only when t is positive.

Ray / AABB

- Intersect ray with all sides, choose earliest intersection that is inside box
- Four sides of of AABB are:
  - x = left
  - x = right
  - y = top
  - y = bottom

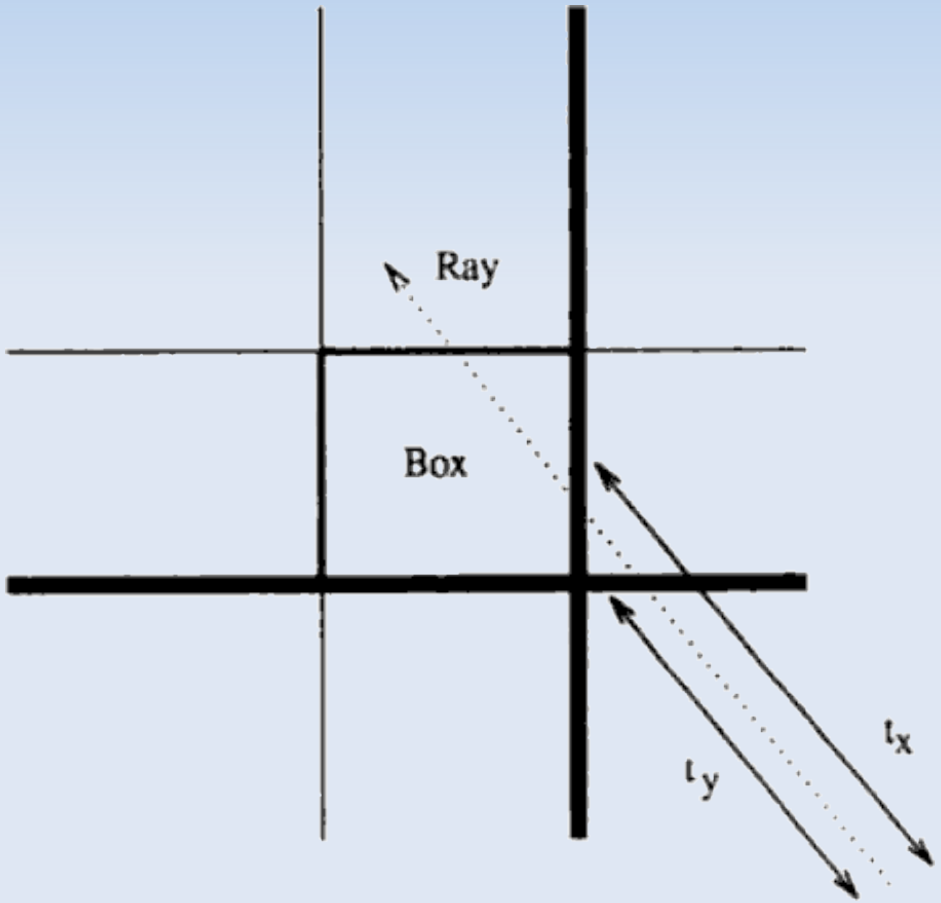


Ray / AABB

- Plug into side formula and simplify:
  - $A_x + t B_x = left \rightarrow t = (left - A_x) / B_x$
  - $A_x + t B_x = right \rightarrow t = (right - A_x) / B_x$
  - Solve for t
- Given t, check if specified point is in box edge
  - $Top < A_y + t B_y < bottom$
- Then do same thing for y, and choose earliest t

Ray / AABB Optimization

- For any particular direction, only two sides could be the earliest
- Unless the ray is exactly parallel to X or Y, only the later of the two times can be inside the box



Homework

- Some sort of projectile with “enemies” to hit.

- When the projectile hits the enemy, the projectile goes away.
- Enemies can have health, so they need to be hit multiple times.
- Extra credit:
  - Enemies can shoot projectiles too!
  - Raycast-style instant shots