More Natural Paths

Matthew Guzdial

guzdial@ualberta.ca

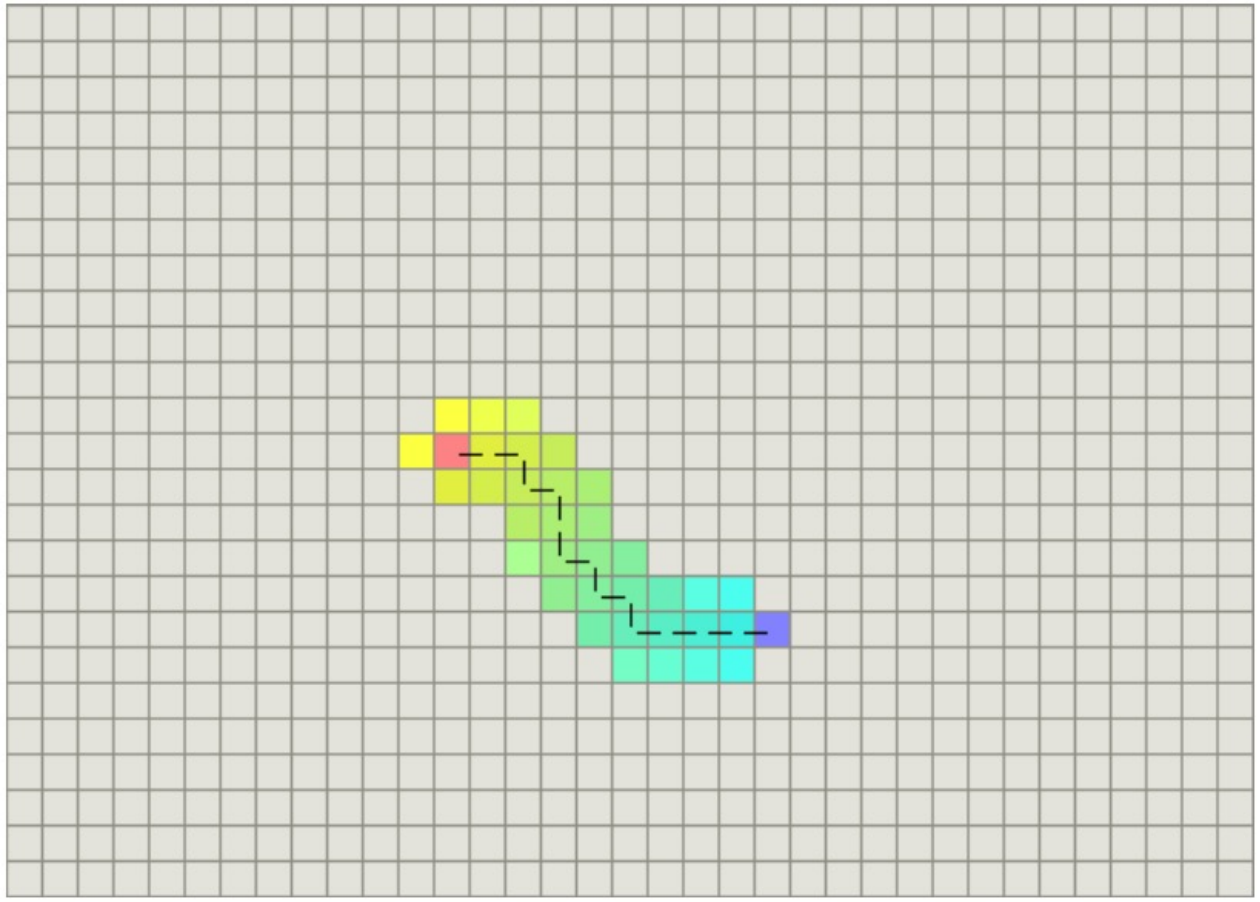
Announcements

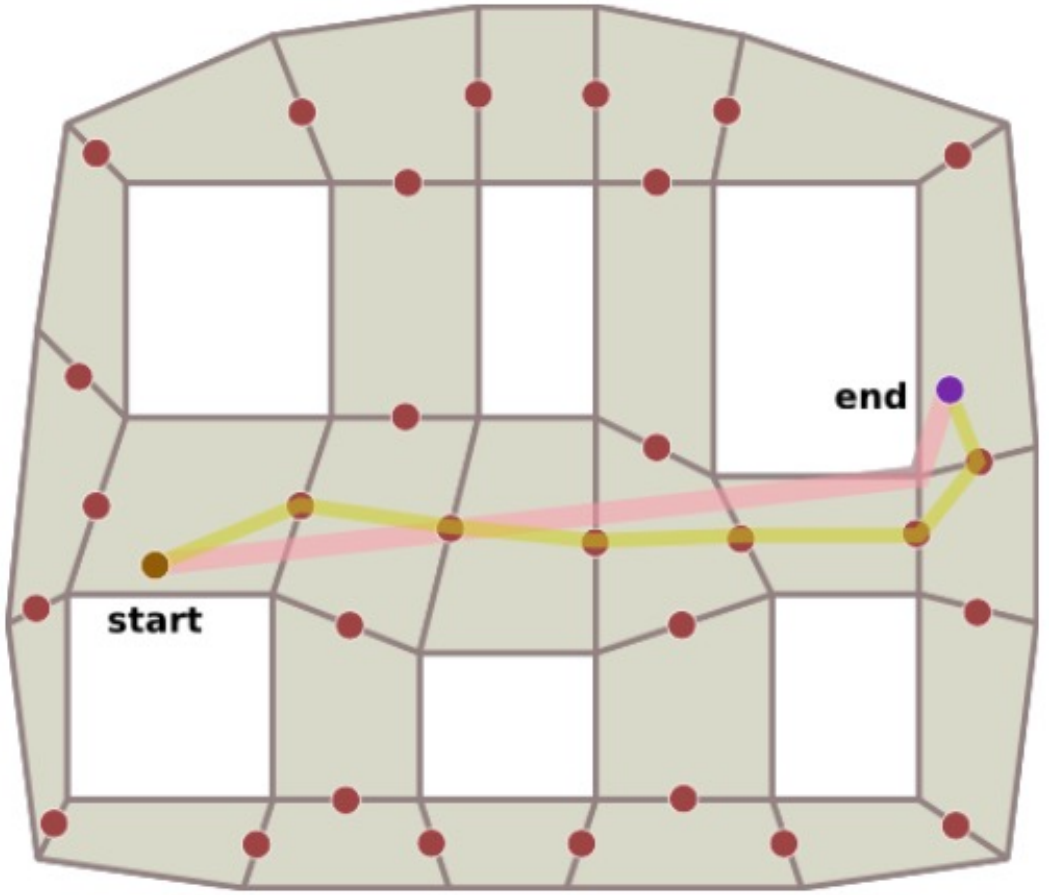
- TA office hours all figured out, on syllabus.
- Lab tomorrow! 5-7:50pm (ETLC E2-002)
- Extra help video for HW1 posted Monday night.
- HW1 due Friday at 11:55pm
  - Watch out for integer division
  - Vector3 should be the final representation type for assignment 1
  - You cannot correctly identify valid grid cells visually
  - **Don’t post your assignment on github.**

Last Class

- Nav Meshes
- Nav Mesh Generation
- Nav Mesh + Path Networks
- Game design can cover for bad AI!

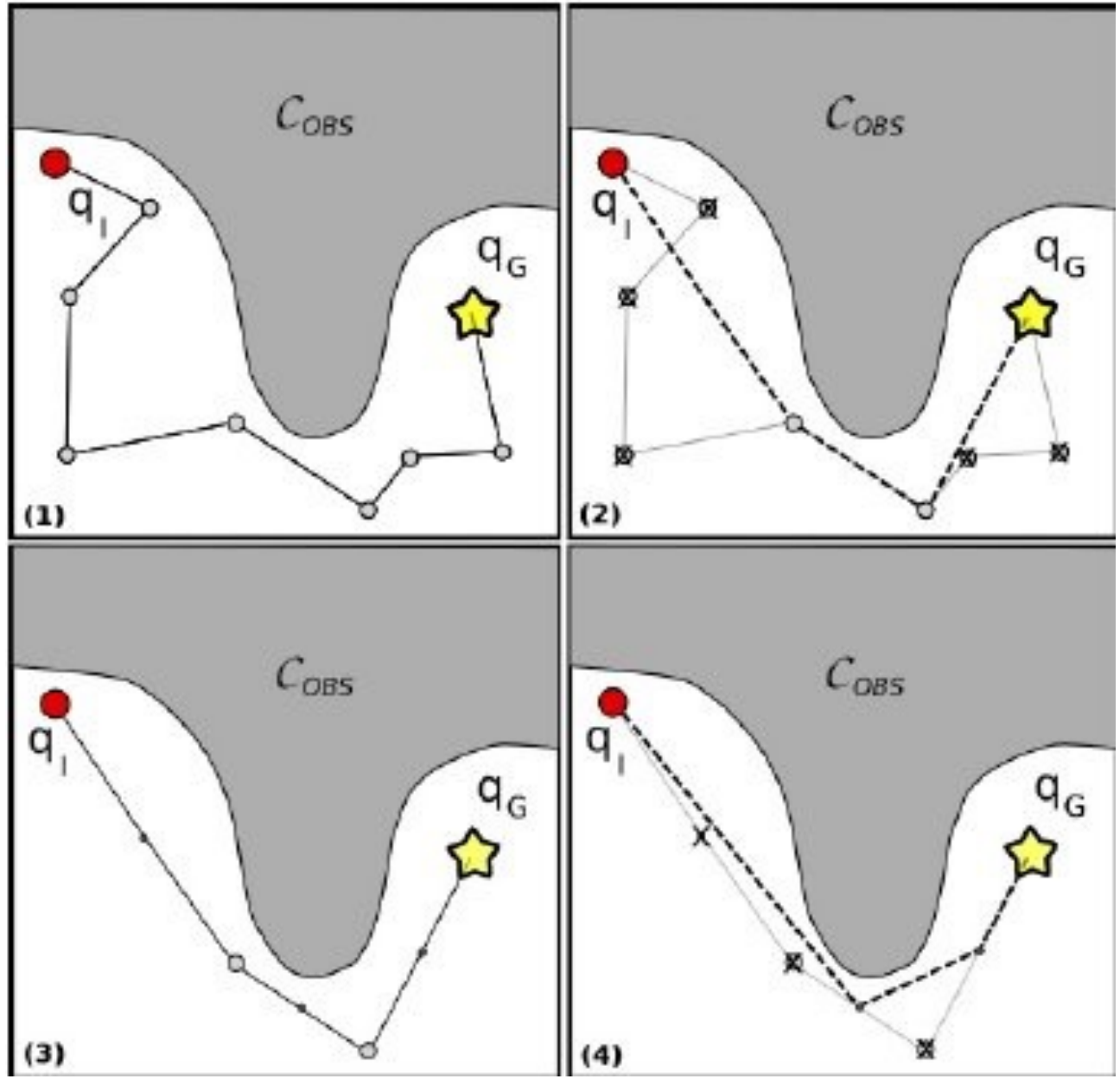
Problem: Greedy Path Planning Looks “Chunky” Even w/ better Representations





Path Smoothing

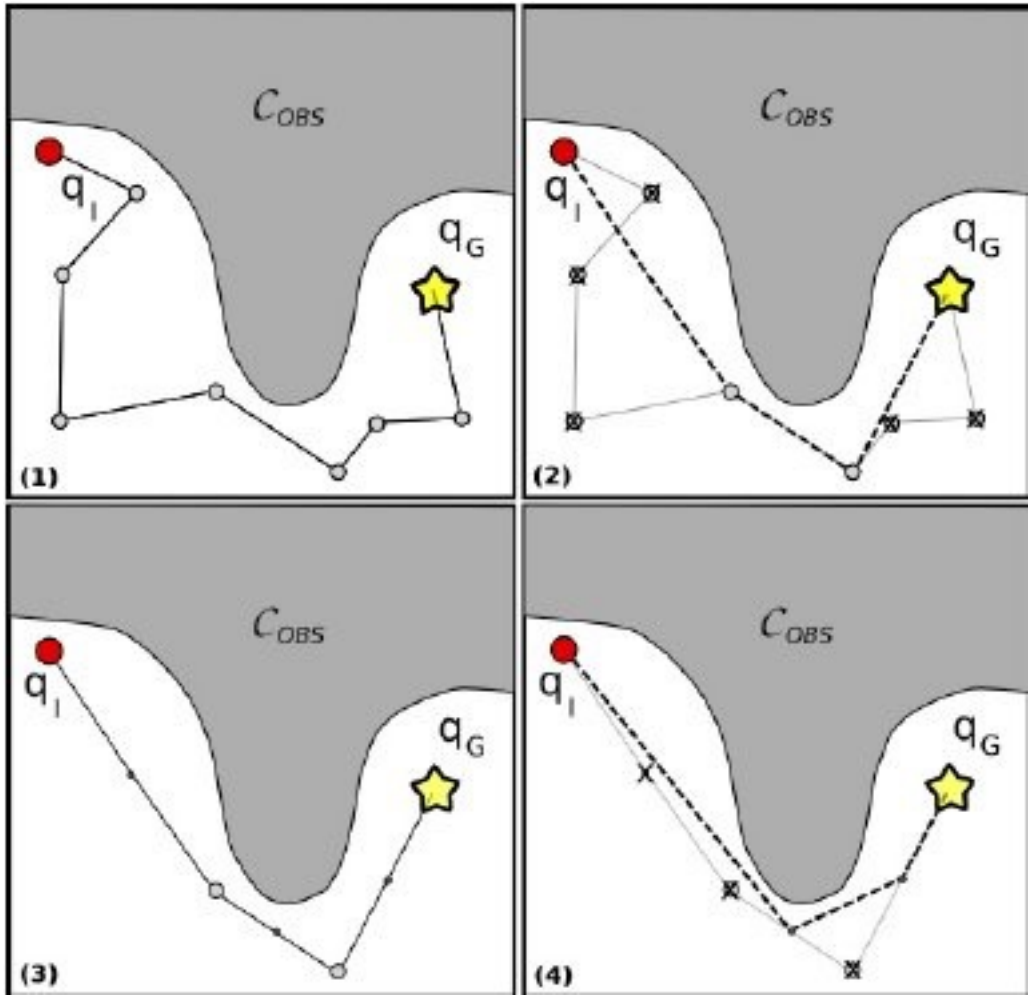
```
Path = CalculatePath()//Path is a list of nodes
currNode = Path[0]
lookAhead = 1
While currNode != goalNode:
    if SafeToTravelBetween(currNode, lookahead):
        Path.remove(lookahead)
        lookahead+=1
    else:
        currNode = Path[lookahead-1]
Return Path
```

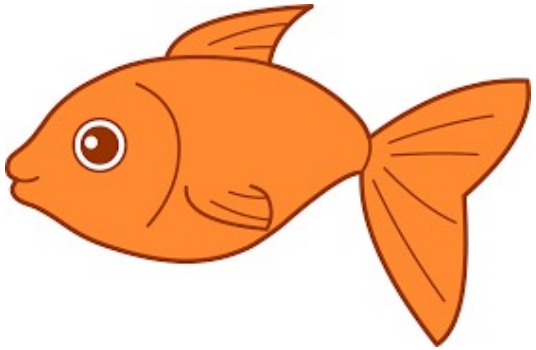


First three give the behavior of the pseudocode on the last page. Fourth goes beyond that!

https://www.researchgate.net/publication/228690192_FootFall_A_ground_based_operations_toolset_enabling_walking_for_the_ATHLETE_rover

Unrealistic Movement, even with Path Smoothing





Participation Question 1

https://forms.gle/M4Y5xx7e8iWngYN48

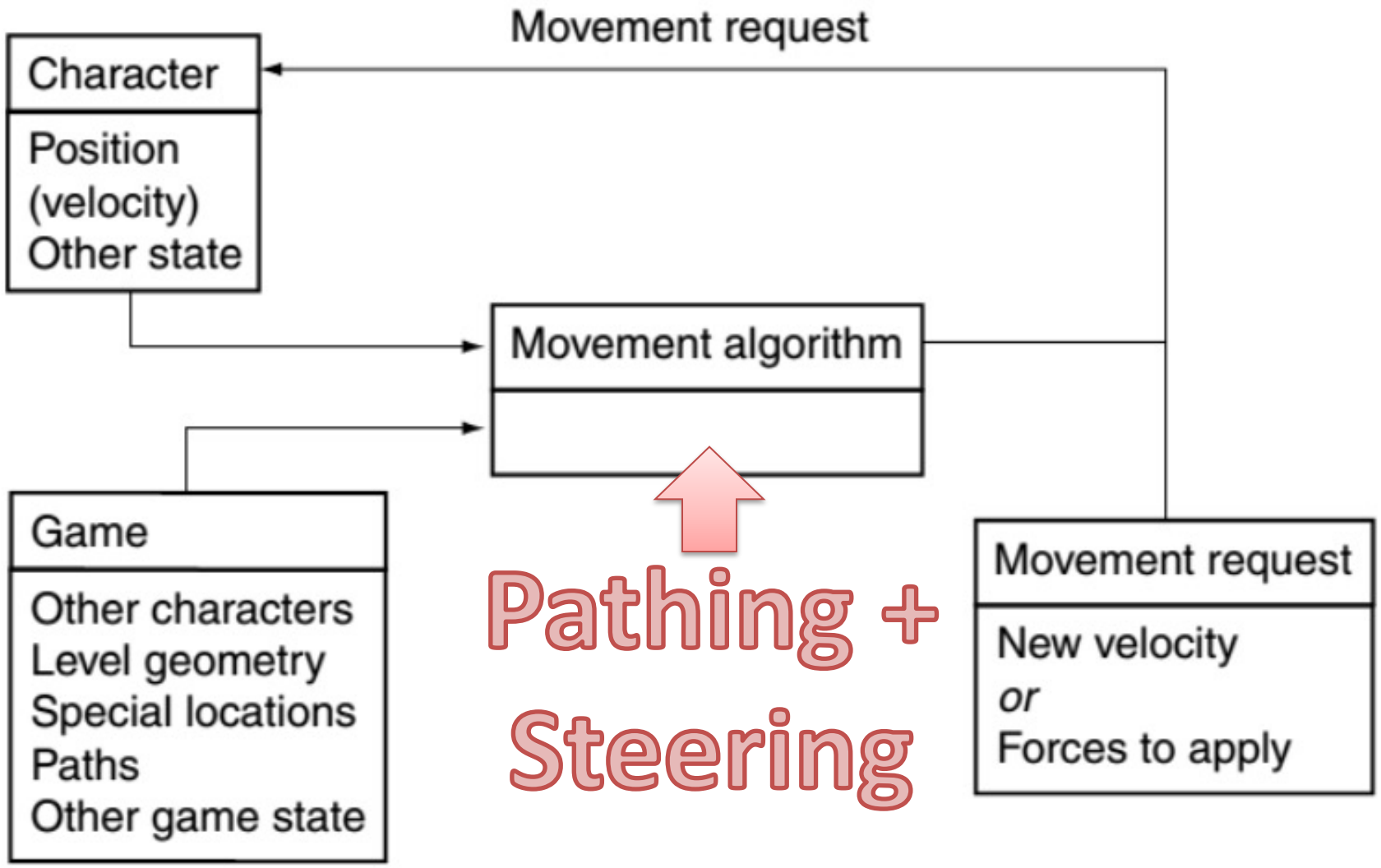
https://tinyurl.com/guz-pq5

What are (other) movement behaviours we'd like to see besides AI agents just moving in a series of straight lines in games?

Goal: Character movement that looks natural with minimal computation.

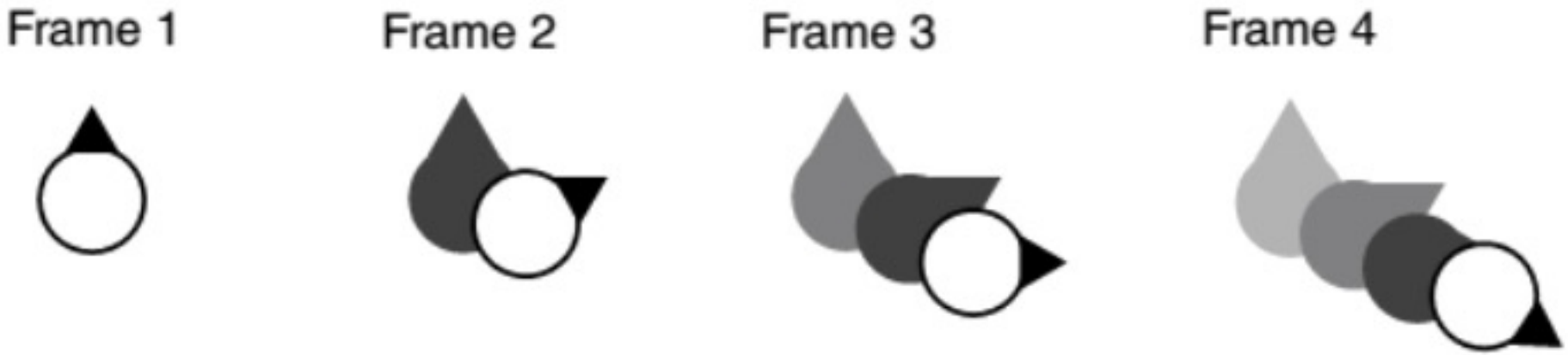
https://youtu.be/J0wa3DX8CZY

Steering: General Framework



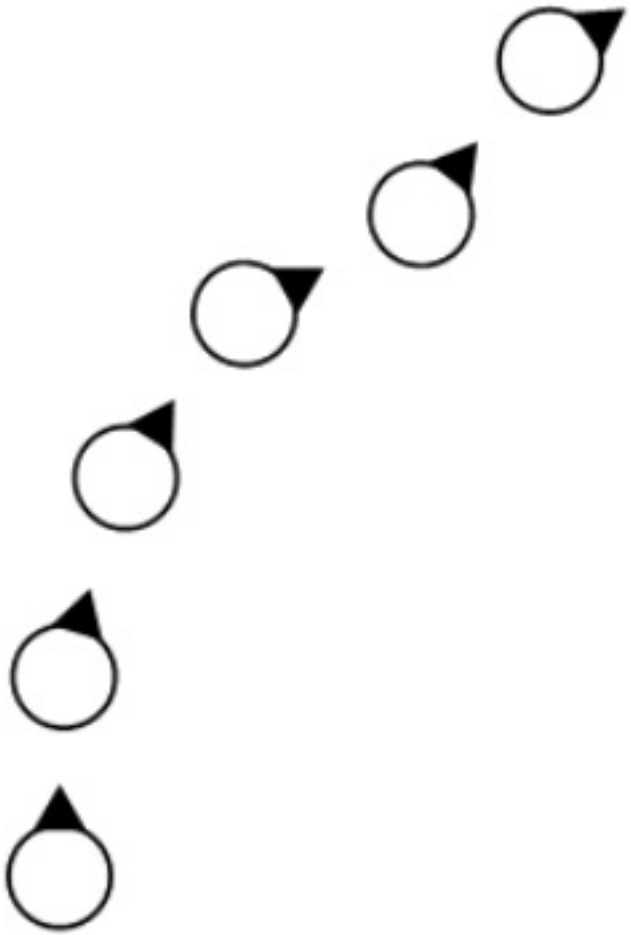
Steering: Facing

- Motion & facing don’t have to be coupled (we don’t have to look where we’re going)
- Many games simplify & force character orientation to be in direction of the velocity
  - Instant (can be awkward)
  - Smoothing



Kinematic Wander

- Move in current direction at max speed
- Vary orientation by some random amount each frame



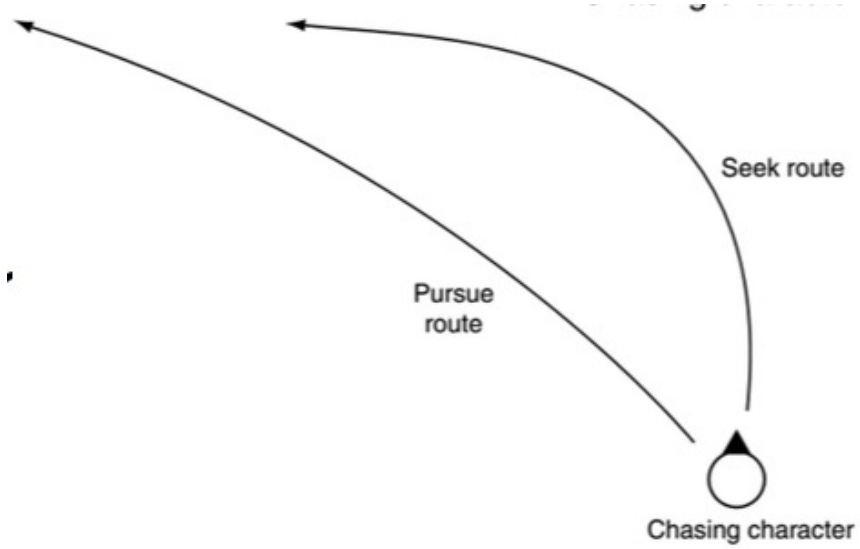
What if we have two characters?

Seek: Match position of character with the target

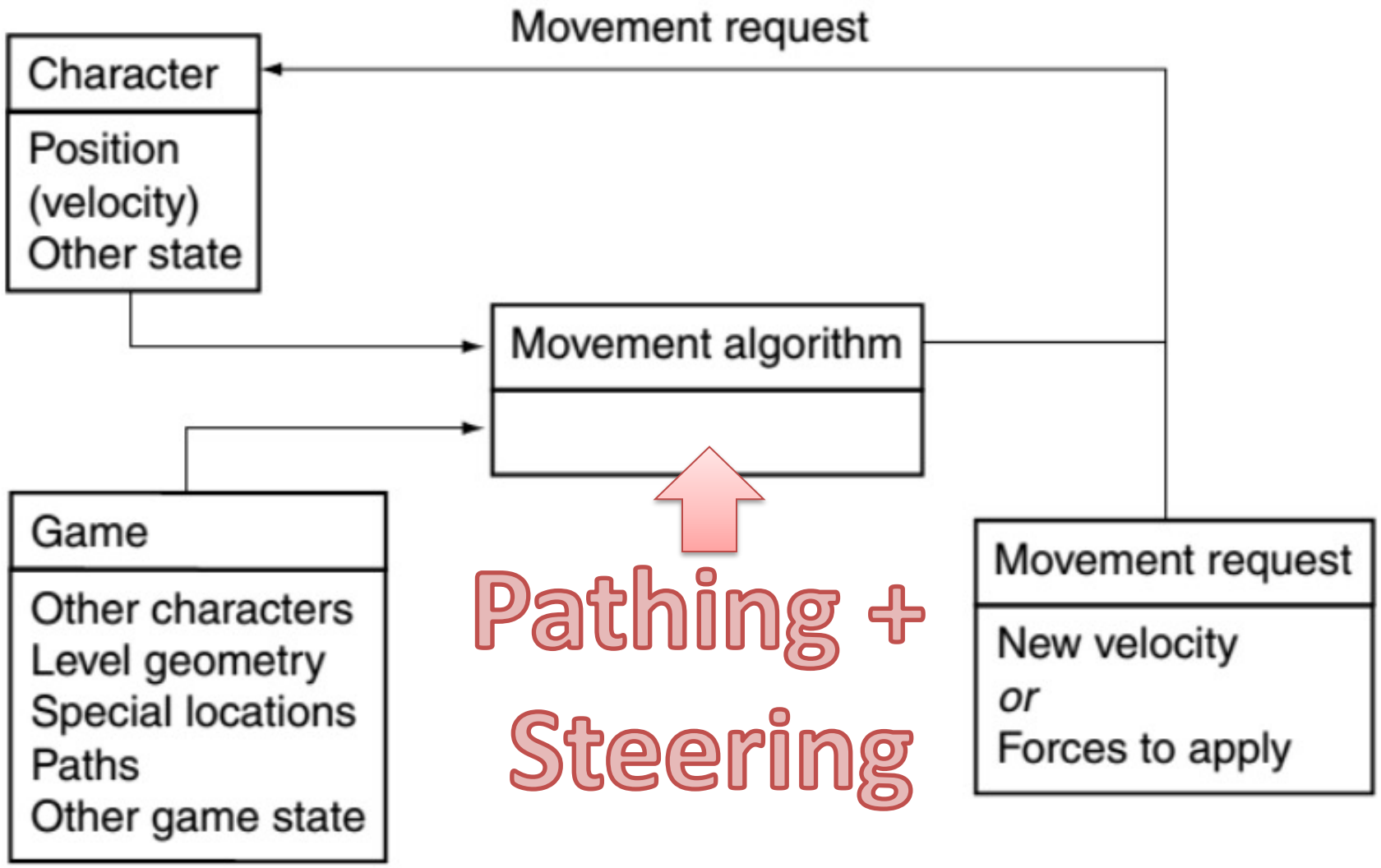
- Find direction to target and go there as fast as possible
  - Kinematic outputs: velocity, rotation
  - Dynamic output: linear and angular acceleration

What can we do beyond seek?

- **Pursue:** Seek based on target motion (instead of position)
- **Evade**
- **Face**
- **Looking where going**
- **Wander**

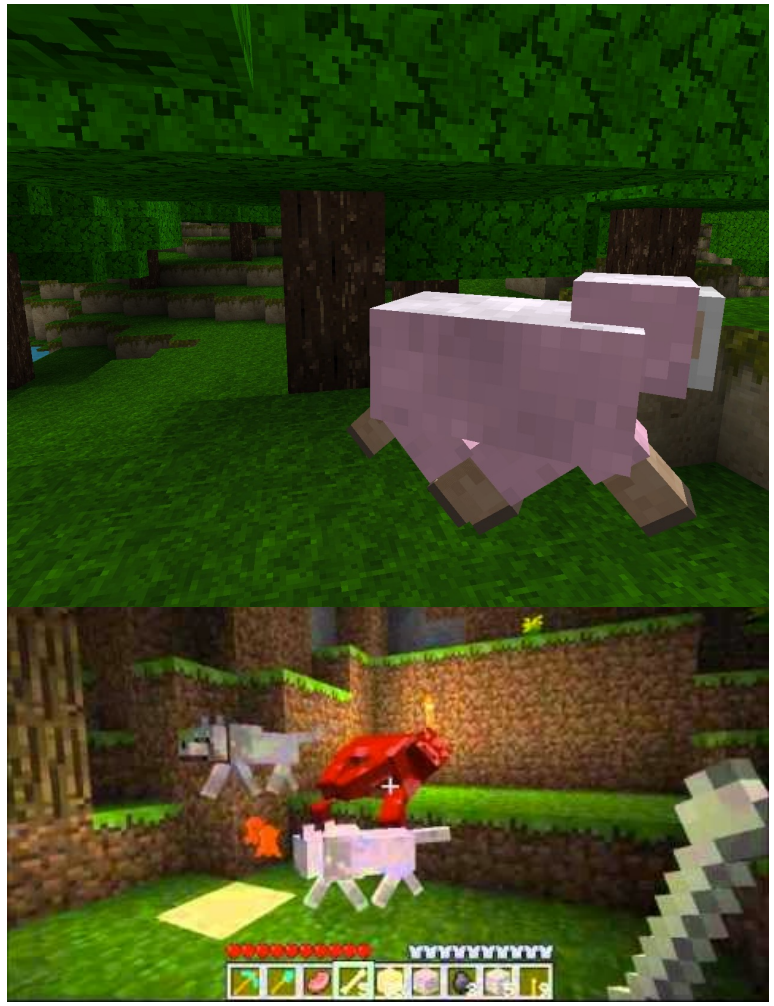


Steering: General Framework



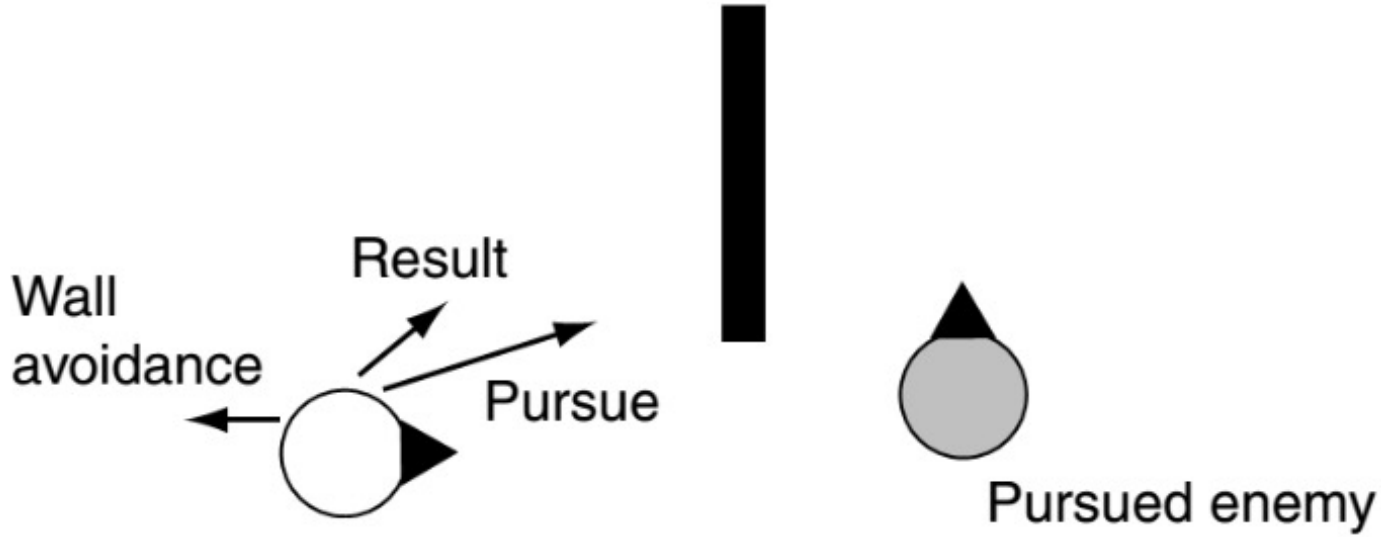
What about combining multiple?

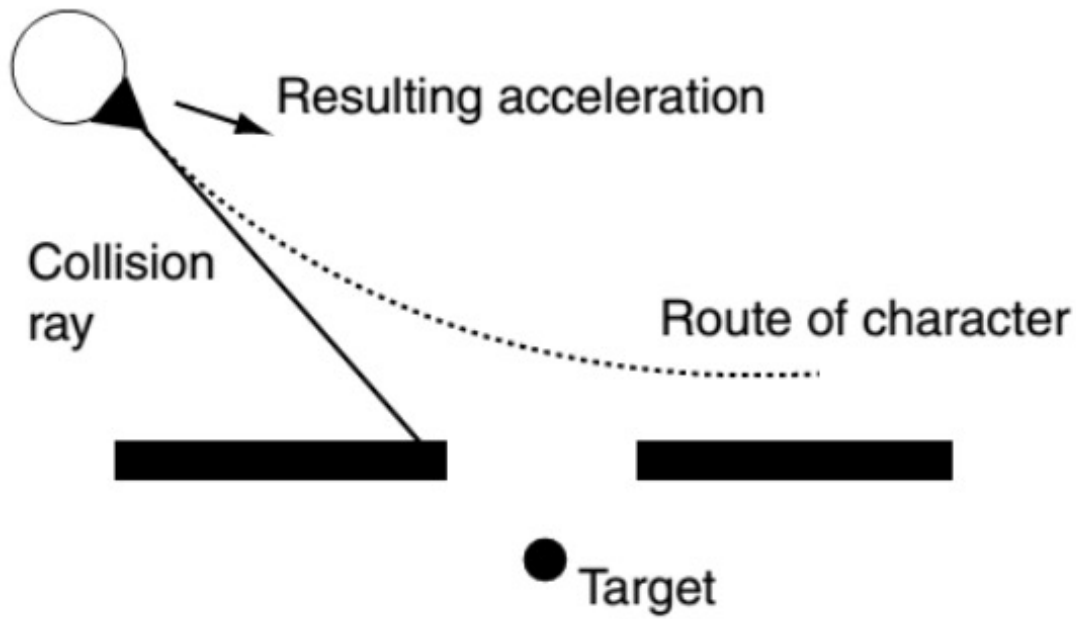
- (Weighted) Blending
  - Execute all steering behaviors
  - Combine results by calculating a next position based on weights.
- Arbitration
  - Selects one proposed steering
  - Not mutually exclusive



Weighted Blending

- Weighted linear sum of accelerations from all involved steering behaviors
- E.g. angry crowd may have 1*separation + 1*cohesion
- Finding “right” weight can be challenging
  - Characters can get stuck (equilibrium)
  - Constrained environments (conflicts)
  - “Jidder”: rapidly move between two positions


Result
Wall
avoidance
Pursue
Pursued enemy


Resulting acceleration
Collision
ray
Route of character
Target

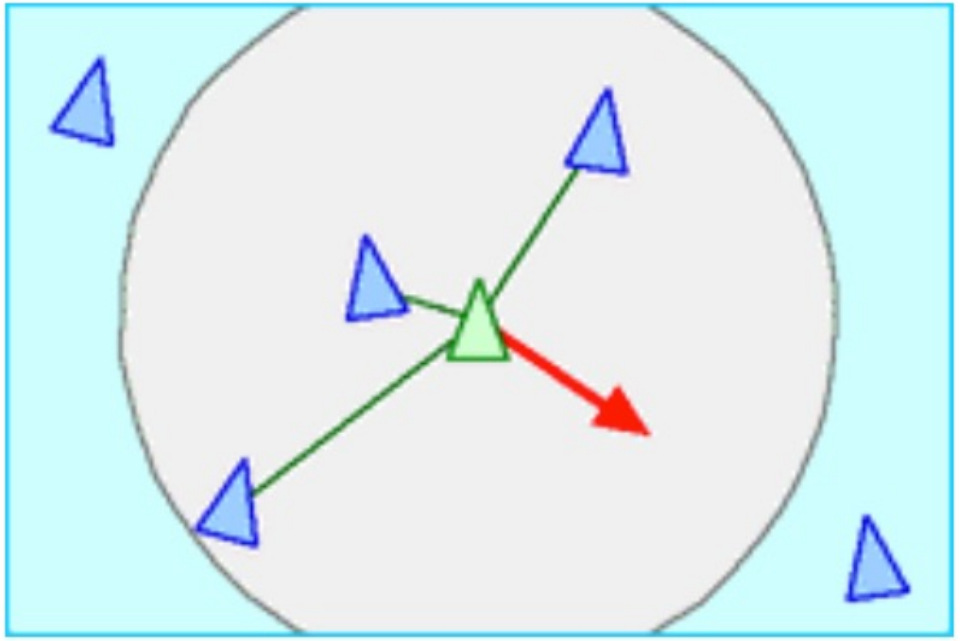
Blending Example: Flocking

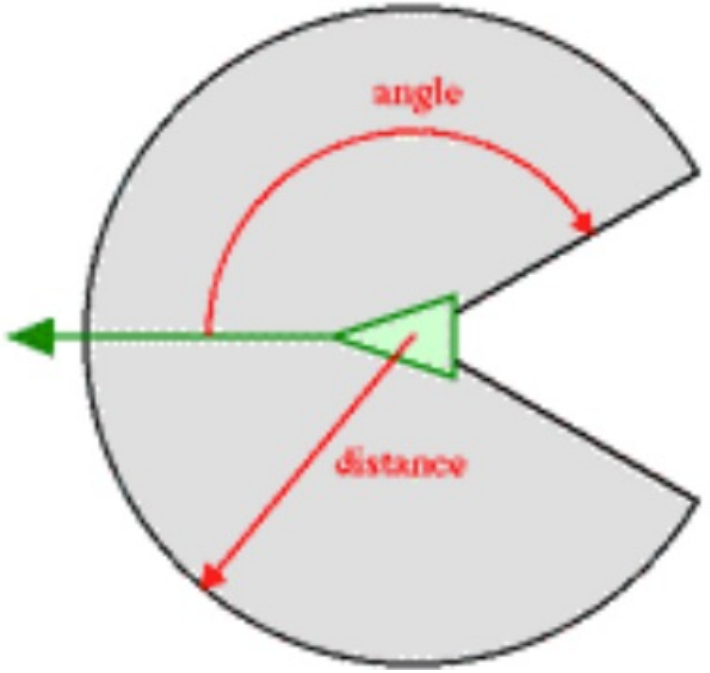
Craig Reynold's "boids" (Flocking)

- Simulated (apparent behavior of) birds,1986
- Blends three steering mechanisms (ordered)
  - Separation: move away from others if too close
  - Cohesion: move to center of mass of flock
  - Alignment: match orientation and velocity of flock

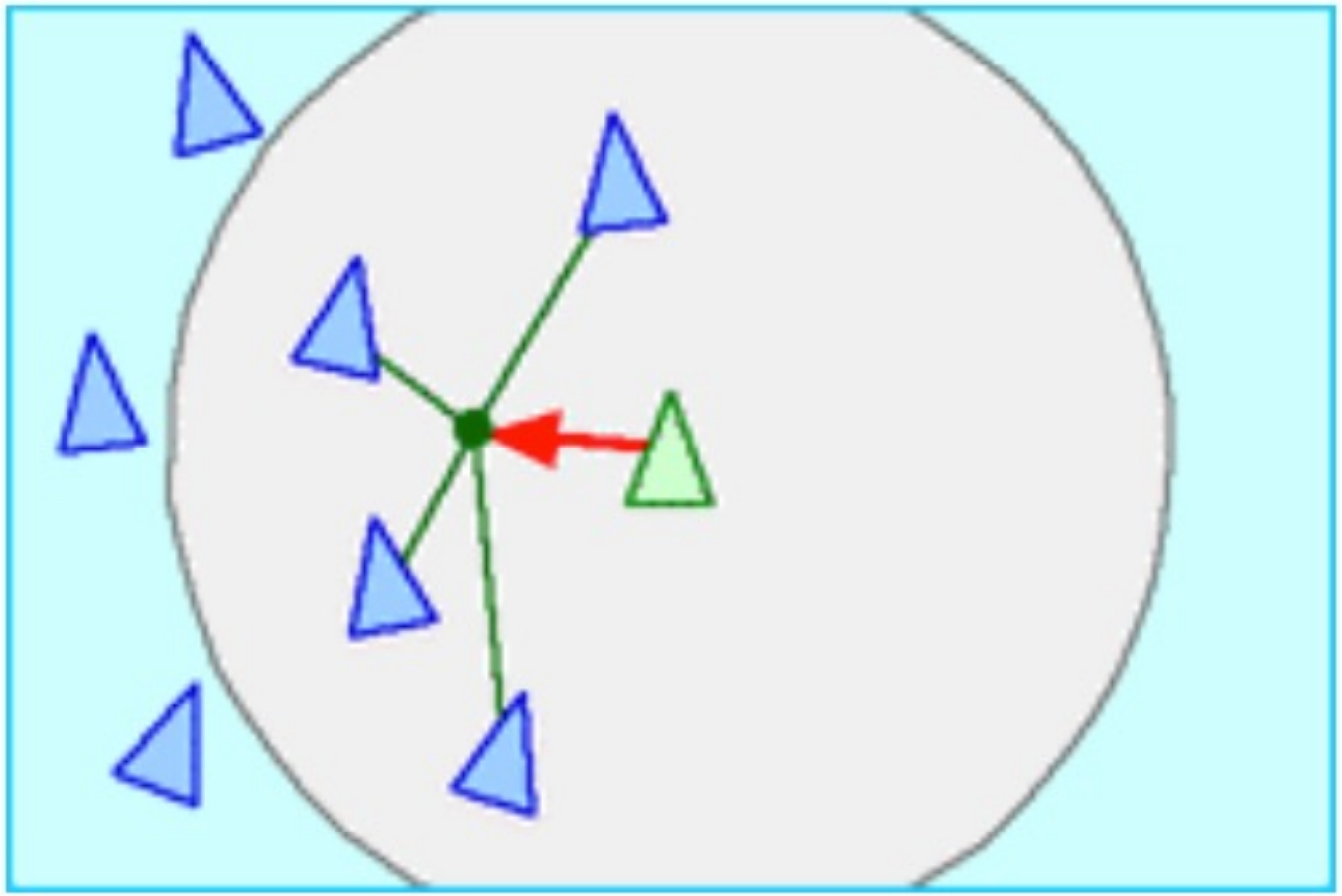
Separation

- Neighborhood is sphere of certain radius, or cone of perception

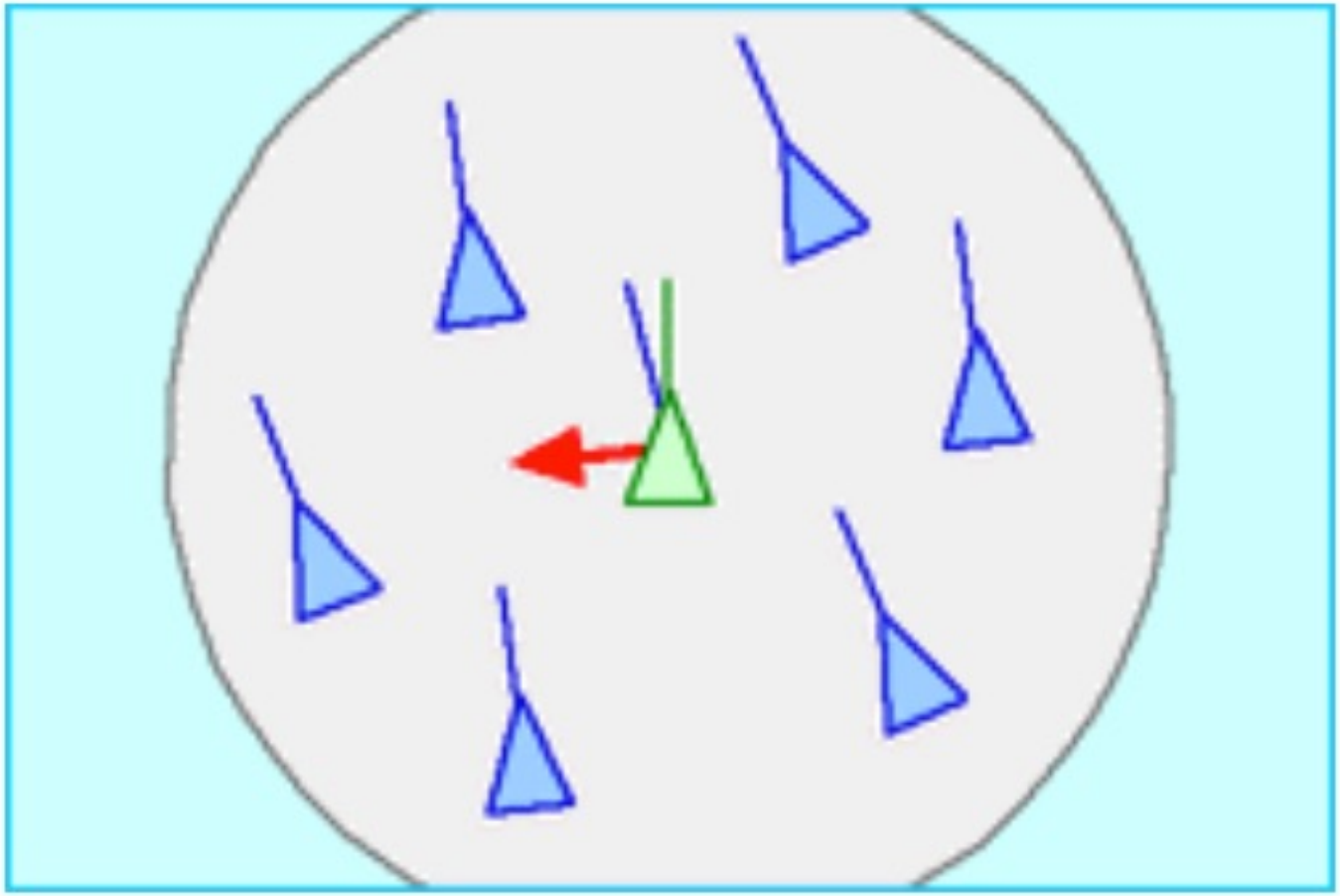




Cohesion



Alignment



Flocking Demos

Technical Example:

https://www.youtube.com/watch?v=WUXq7GYH62Y

Sheep (2000)

Game Example: Townscaper (watch the birds):

https://youtu.be/wp3T8At5644

- We'll be covering the PCG effect used to produce the towns at the end of the semester

Pikmin 3 (2013)

Participation Question 2: Simplified Flocking

https://tinyurl.com/guz-pq52

https://forms.gle/5E7GmxxGUr5zQopV8

Given these three entities and these rules, what would each entities new position be after one update tick? (Assume the entities are updated/checked in alphabetical order)

A. **Entity A** at position (0,0,0)
B. **Entity B** at position (0.5, 1.0, 0.5)
C. **Entity C** at position (1.0, 0.5, 1.0)

1. **Separation**: if the difference to the closest entity in any dimension is <=1, *move 1.0* further away from that entity in that dimension.
2. **Cohesion**: Calculate the midpoint of the other two entities. *Move 0.5* closer to that midpoint in each dimension.

Entity A: starts at (0,0,0)

- Separation: B (0.5, 1.0, 0.5) is the closest entity, so move A to (-1, -1, -1)
- Cohesion: Midpoint of B and C is (0.75, 0.75, 0.75), so move A from (-1, -1, -1) to **(-0.5, -0.5, -0.5)**.

Entity B: starts at (0.5, 1.0, 0.5)

- Separation: C (1, 0.5, 1) is the closest Entity, so move B to (-0.5, 2.0, -0.5)
- Cohesion: Midpoint of A and C is (0.25, 0, 0.25) so move B from (-0.5, 2.0, -0.5) to **(0, 1.5, 0)**

Entity C: starts at (1.0, 0.5, 1.0)

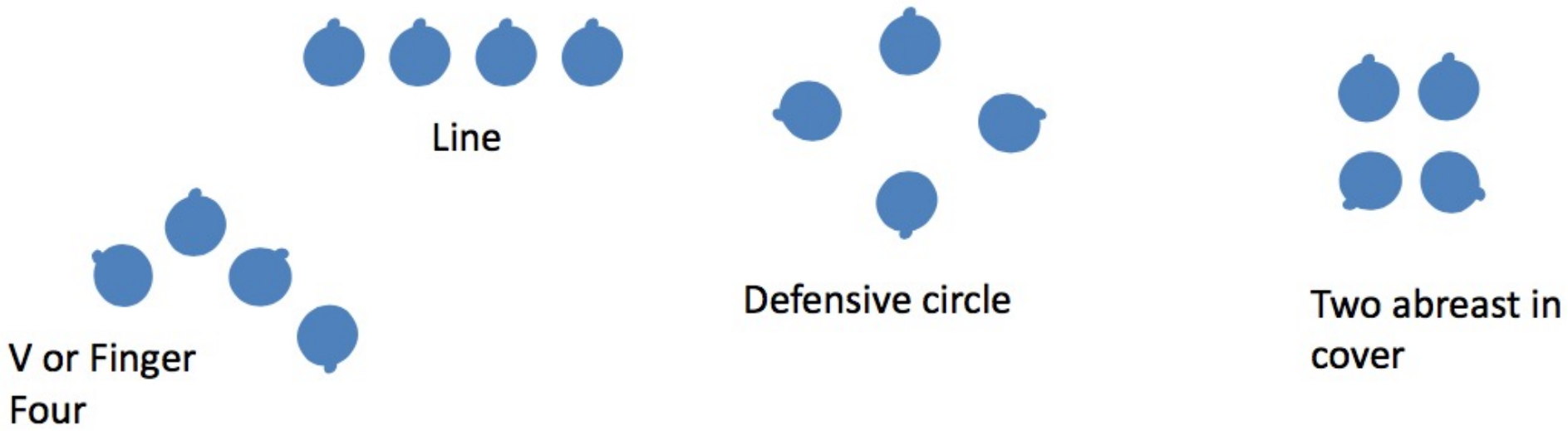
- Separation: B (0, 1.5, 0) is the closest Entity, so move C to (2, -0.5, 2)
- Cohesion: Midpoint of A and B is (-0.25, 0.5, -0.25) so move C from (2, -0.5, 2) to **(1.5, 0, 1.5)**

Formations

Instead of algorithmically determining individual position, can pre-define particular formations.

Two ways to handle:

1. Path plan for leader: All others try to stay at an offset to leader
2. Entire team is a single agent: All as one pathing



Mass Effect: Elevator

Final Fantasy 7 Remake

Kingdom Hearts 3

Fire Emblem: Three Houses

Summary

- An AI agent doesn’t have to just follow the path its given as a set of straight lines.
- We can smooth the path to make it look less “spiky”
- We can include extra “steering” behaviour (lots of options!)

Next Time

- Introduction to A*
- Assignment 2 is Released!
- Practice Quiz 2
- Assignment 1 is Due (that night)