Disclaimer: I use these notes as a guide rather than a comprehensive coverage of the topic. They are neither a substitute for attending the lectures nor for reading the assigned material.

Announcements

- HW 3 due Sunday night, February 11

PREVIOUSLY ON…

N-2: Heuristic Search Recap

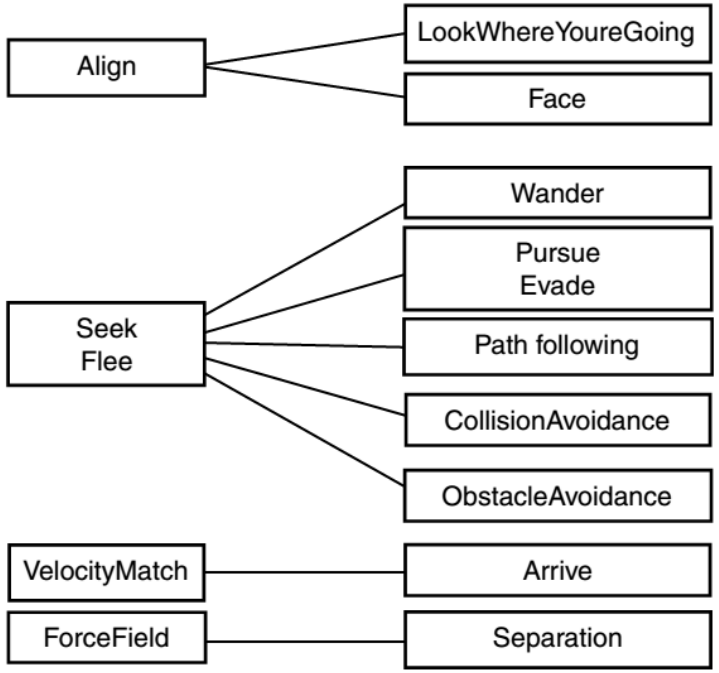
- A*
  - Use when we can't precompute
    - Dynamic environments
    - Memory issues
  - Optimal when heuristic is admissible (and assuming no changes)
  - Replanning can be slow on really big maps
- Hierarchical A* is the ~same, but faster
  - Within 1% of A* optimality but up to 10x faster
- Real-time A*
  - Stumbling in the dark, 1 step lookahead
  - Replan every step, but fast!
  - Realistic? For a blind agent that knows nothing
  - Optimal when completely blind
- Real-time A* with lookahead
  - Good for fog-of-war
  - Replan every step, with fast bounded lookahead to edge of known space
  - Optimality depends on lookahead

N-1: Movement & Steering

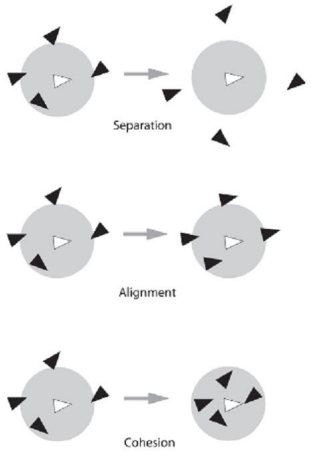
1. What do movement algorithms output in static environ?
2. What do movement algorithms input in kinematic environ?
3. What do movement algorithms output in kinematic environ?
4. What is the deal with time & variable frame rates?
5. What was the insight about updates if time << 1?
6. How are kinematic seek and pursue different?
7. What's the point of kinematic arrival?
8. Kinematic wander varies what randomly?
9. What’s the main difference between kinematic and steering/dynamic movement?

Steering

1. Steering vs flocking?
2. Steering Family Tree
3. How might we combine behaviors?
4. What three steering mechanisms enable flocking?



Millington Fig 3.29



Buckland Fig 3.16

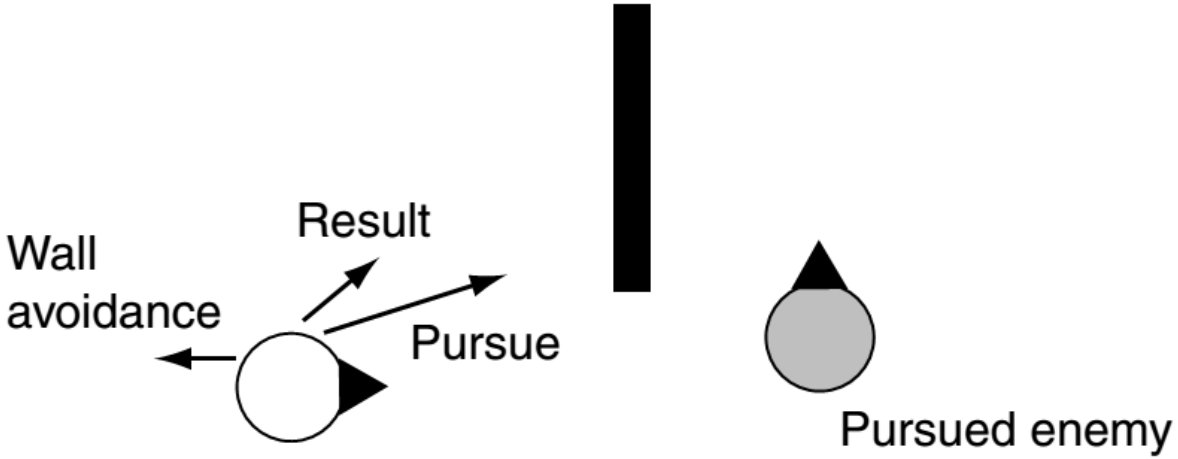
Steering Behaviors (Dynamic)

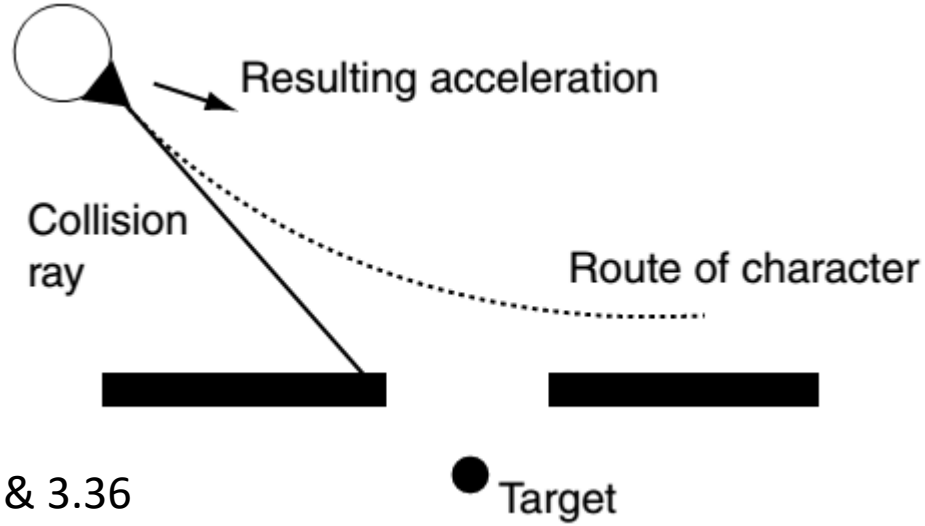
- Kinematic movement
  - Input: use target position and orientation, no velocities.
  - Outputs: desired velocity
- Steering movement (behaviors)
  - Input: target information
    - Velocity and rotation
    - Collision geometry
    - Paths, for path following
    - Average Flock information
  - Output: accelerations
    - Linear acceleration: 2 or 3-D vector
    - Angular acceleration: single float value

- Steering extends kinematic movement by **adding acceleration and rotation**
  - Remember:
    - $\mathbf{p}(t)$: position at time t
    - $\mathbf{v}(t) = \mathbf{p}'(t)$: velocity at time t
    - $\mathbf{a}(t) = \mathbf{v}'(t)$: acceleration at time t
  - Hence:
    - $\Delta\mathbf{p} \approx \mathbf{v}$
    - $\Delta\mathbf{v} \approx \mathbf{a}$

Variable Matching Conflicts

- Match position and orientation? Ok
- Match position and velocity? Conflict
- Moral: have individual matching algorithms, and conflict-resolving combination algorithm





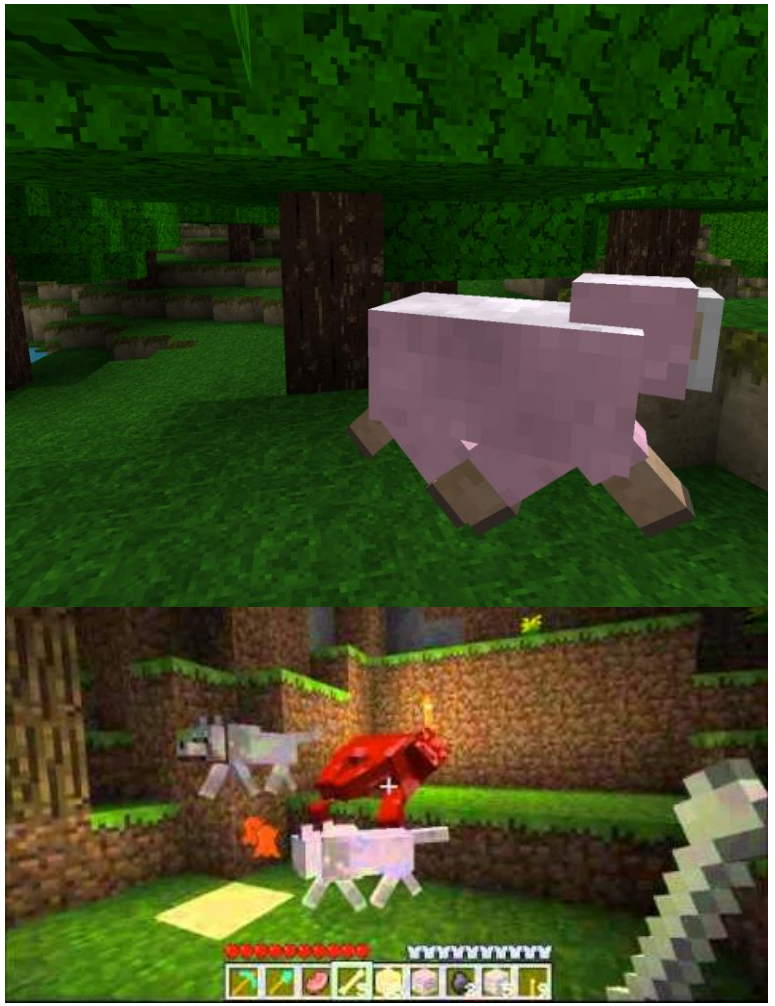
Millington Fig 3.35 & 3.36

can't avoid an obstacle and chase

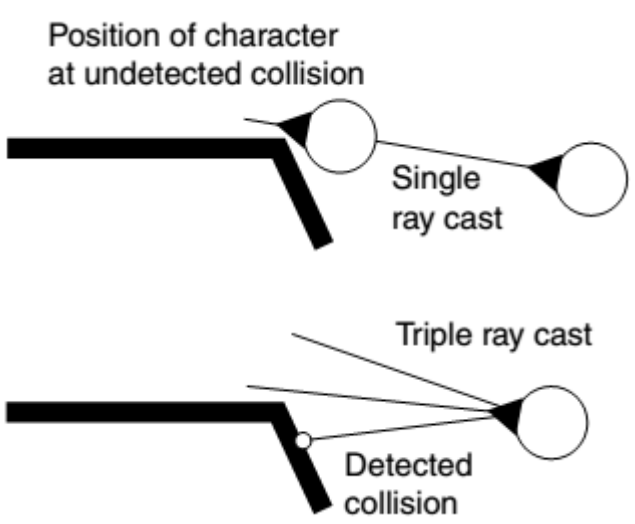
missing a narrow doorway

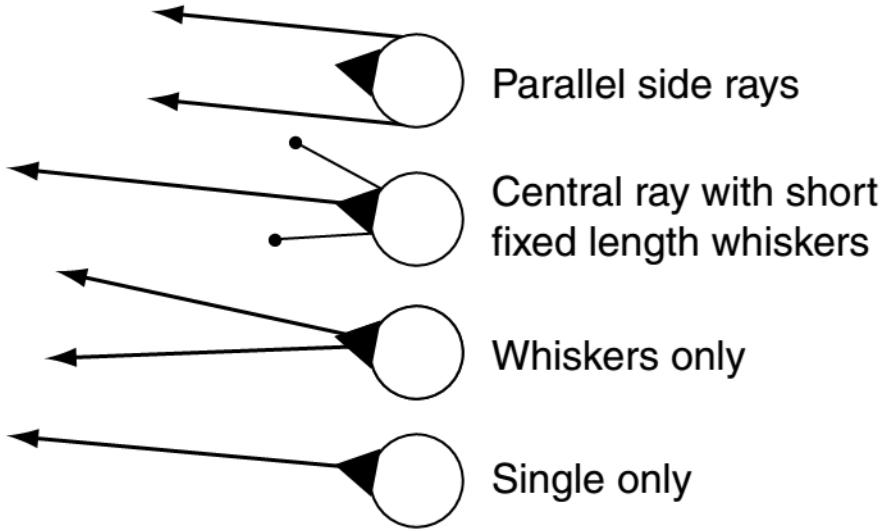
Combining Steering Behavior

- (Weighted) Blending
  - Execute all steering behaviors
  - Combine results by calculating a compromise based on weights
    - Example: Flocking based on separation and cohesion
- Arbitration
  - Selects one proposed steering
- Not mutually exclusive
- Emergent Behavior



One is not enough





Millington Fig 3.25 & 3.26

Weighted Blending

- Simplest way to combine steering behaviors
- Weighted linear sum of accelerations from all involved steering behaviors
- Post-processing velocity threshold
- E.g. rioting crowd may have 1*sep + 1*cohes
- Finding “right” weight can be challenging
  - Characters can get stuck (equilibrium)
  - Constrained environments (conflicts)
  - Jidder

Graphs, Search, Pathfinding

(behavior involving **where** to go)

Steering, Flocking, Formations

(behavior involving **how** to go)

Flocking, Formations

2018-01-30

M&F 3.1-3.4

B 3

See also

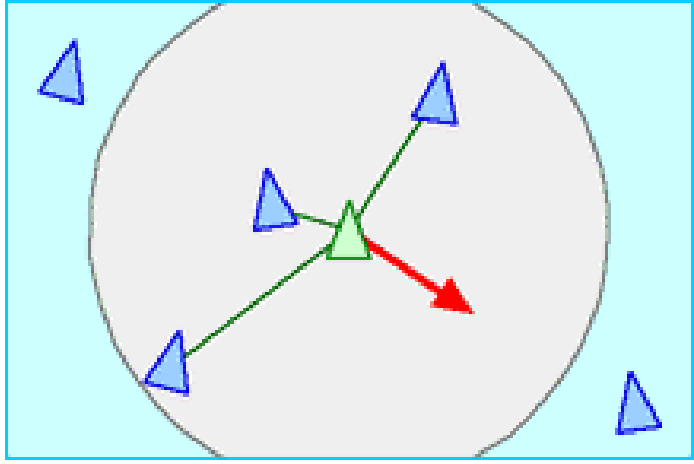
- M Book, Ch 3.1-3.4. Also website: www.ai4g.com
  - Algorithms for K {wander, arrive, seek, flee}
  - https://github.com/idmillington/aicore
- B Ch 3 (B Ch 1)
  - Download sample materials: http://www.jblearning.com/catalog/9781556220784/
- Animations (for simple). Craig Reynolds
  - http://www.red3d.com/cwr/steer/

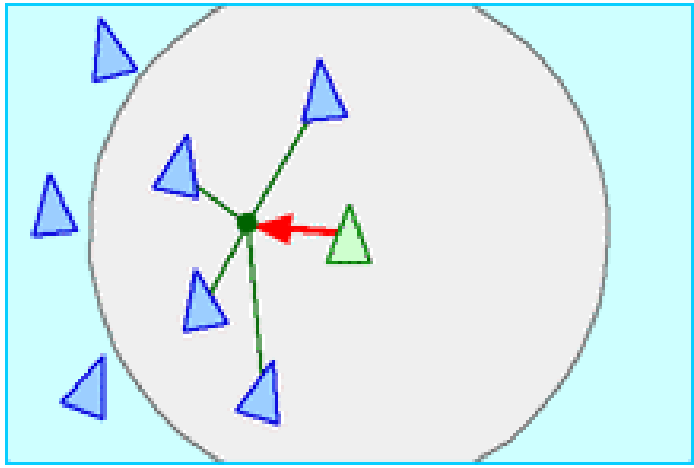
Problems

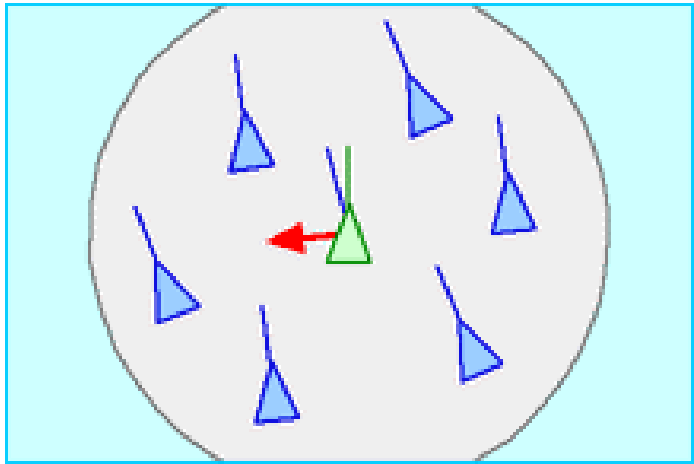
- Bunching:
  - https://www.youtube.com/watch?v=ZIAmoRsu3Z0&feature=youtu.be&list=PLxGbBc3OuMgg7OuyLfvXQLR6HKcoglCfG&t=1833
  - AIIDE 2015 keynote, Adam Noonchester. “AI In The Awesomepocalypse”
  - Sunset Overdrive: IGN 9.0 “Awesome”, Editors’ choice
- Too close:
  - https://youtu.be/ZIAmoRsu3Z0?list=PLxGbBc3OuMgg7OuyLfvXQLR6HKcoglCfG&t=1713

Flocking and Swarming

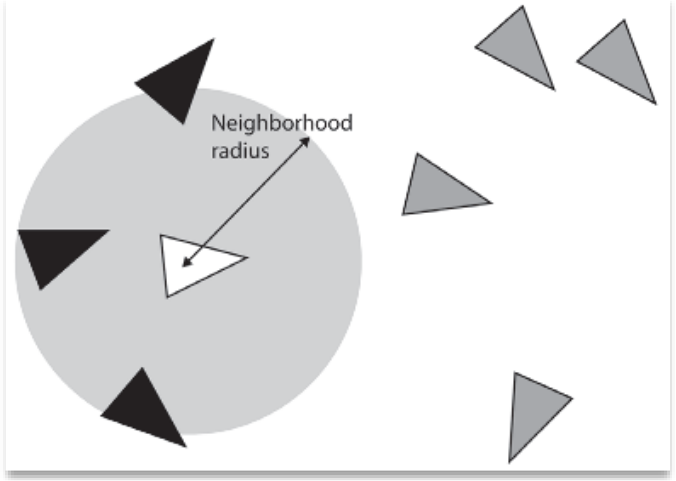
- Craig Reynold's "boids" (Flocking != Swarming)
  - https://www.youtube.com/watch?v=86iQiV3-3IA
  - https://www.youtube.com/watch?v=QbUPfMXXQIY
  - Simulated (apparent behavior of) birds, 1986
  - Blends three steering mechanisms (ordered)
    - Separation: Move away from other birds that are too close
    - Cohesion: Move to center of mass of flock
    - Alignment: Match orientation and velocity of flock
  - Equal Weights for simple flocking behavior



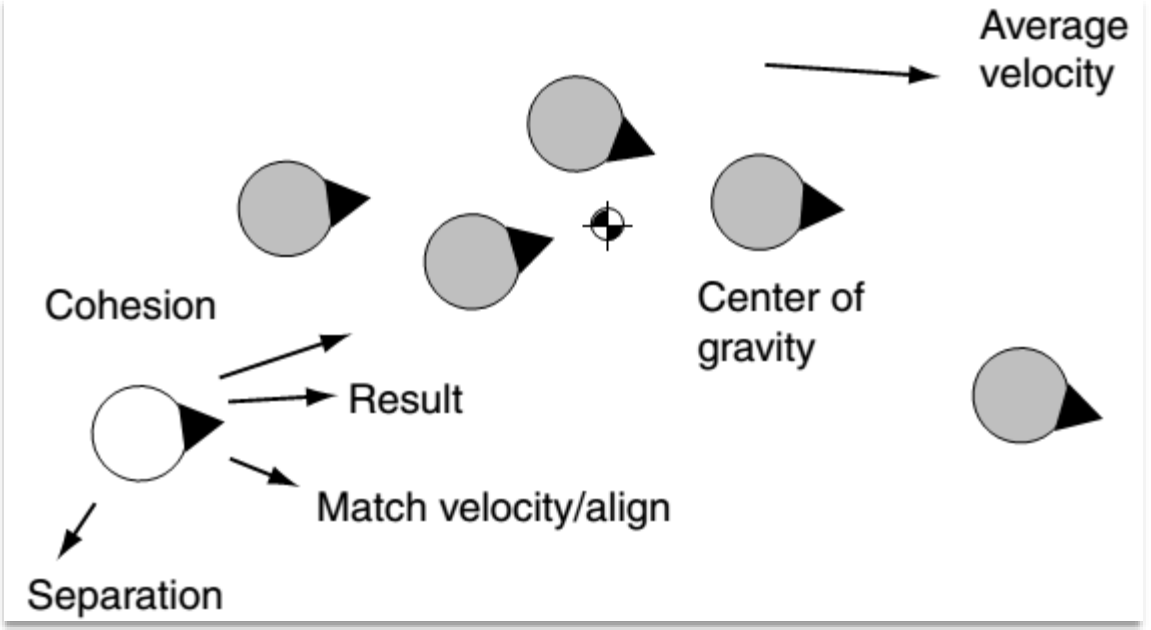




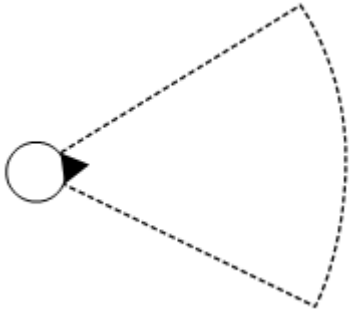
But 1st: won't you be my neighbor



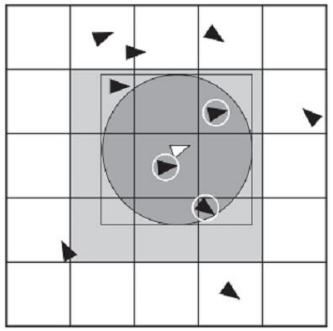
Buckland Fig 3.15



Millington Fig 3.31



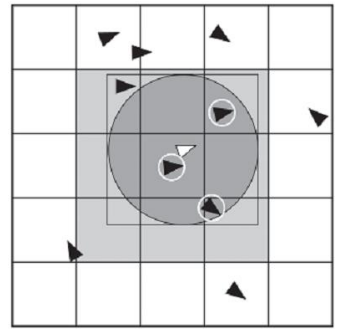
Millington Fig 3.32



Buckland Fig 3.18

Recall findNearestWaypoint()

- Most engines provide a rapid “nearest” function for objects
- Spatial partitioning w/ special data structures:
  - Quad-trees (2d), oct-trees (3d), *k*-d trees
  - Binary space partitioning (BSP tree): https://en.wikipedia.org/wiki/Binary_space_partitioning
  - Multi-resolution maps (hierarchical grids)
- The gain over all-pairs techniques depends on number of agents/objects



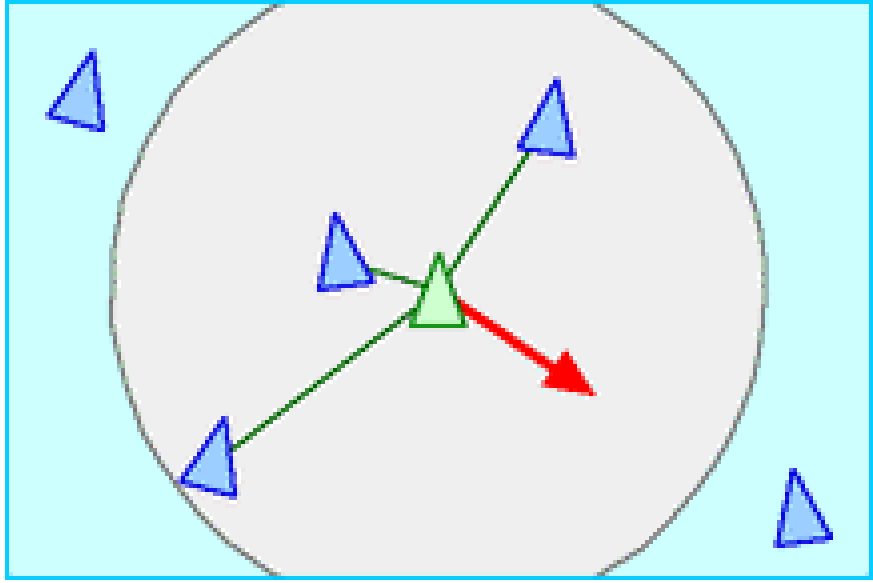
Buckland Fig 3.18

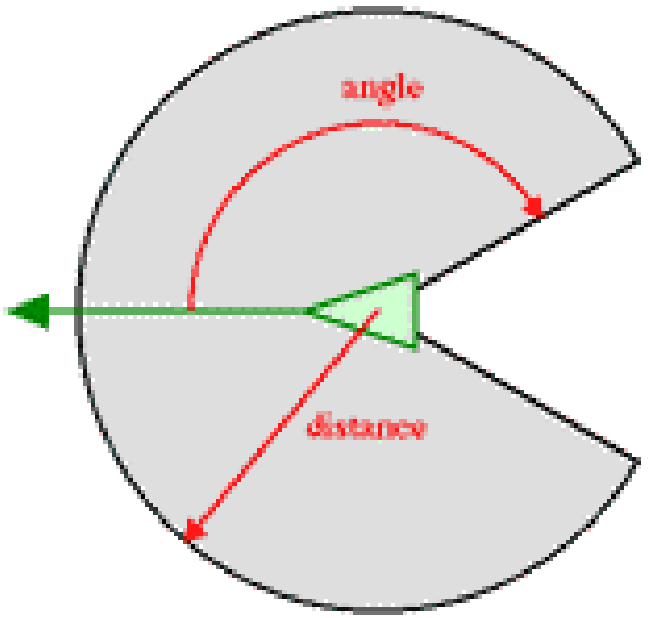
Demo

- Another big shoal (neighbors)

Separation

- Steer to avoid crowding local flockmates
  - Force to steer a bot away from neighbors
  - Neighborhood is a sphere of certain radius, or possibly a cone of perception
  - The vector to each bot under consideration is normalized, divided by the distance to the neighbor, and added to the steering force.

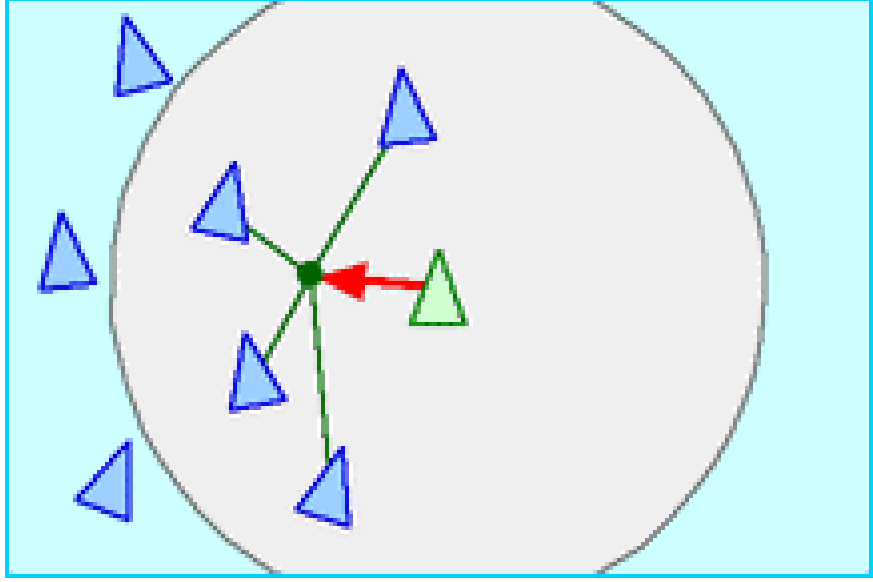




http://www.red3d.com/cwr/boids/

Cohesion

- Steer to average **position** (center of mass) of local flockmates
  - Desired position (center of mass): iterate through all neighbors and average their positions
  - Seek to desired position

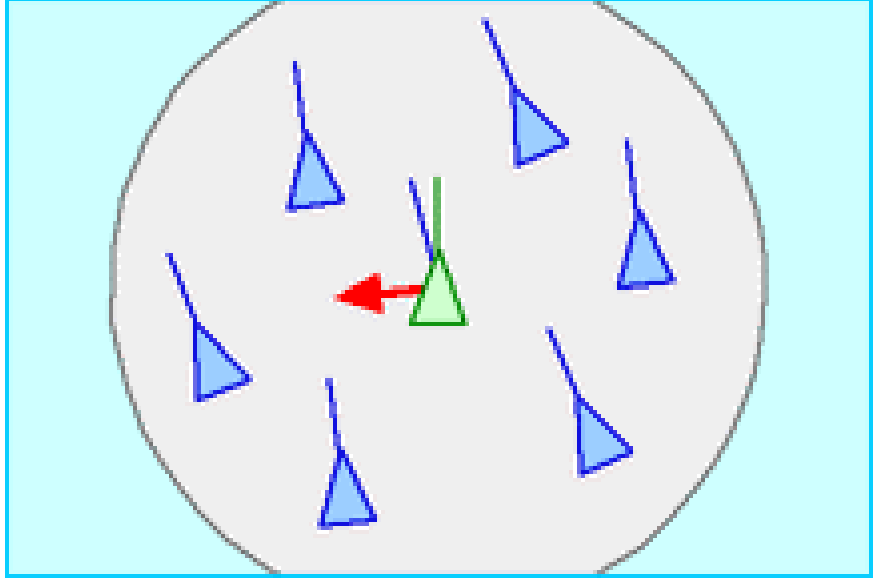


* Center of mass is the average position (X,Y,Z) of boids in neighborhood.

http://www.red3d.com/cwr/boids/

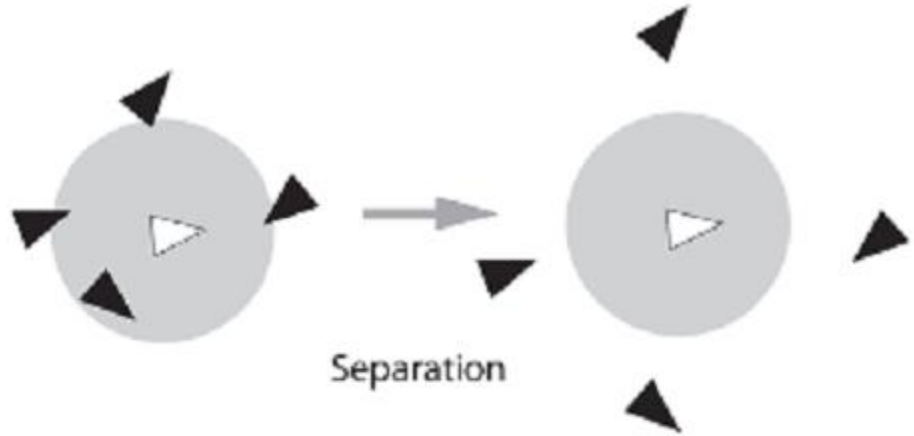
Alignment

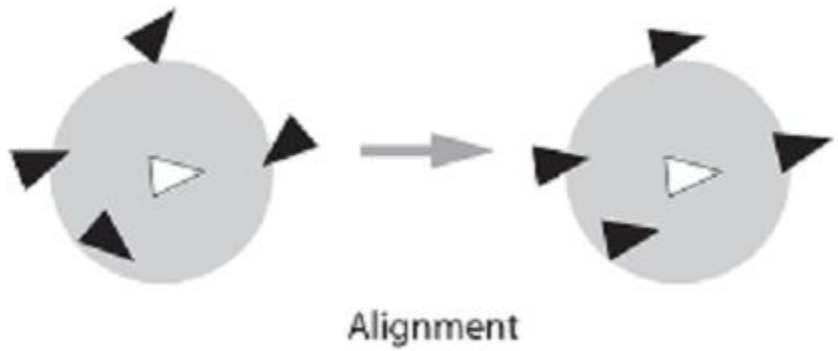
- Steer towards average **heading**
  - Attempts to keep bots aligned with neighbors
  - Desired heading: iterate through all neighbors and average their heading vectors
  - For each neighbor, subtract bot's heading from desired heading

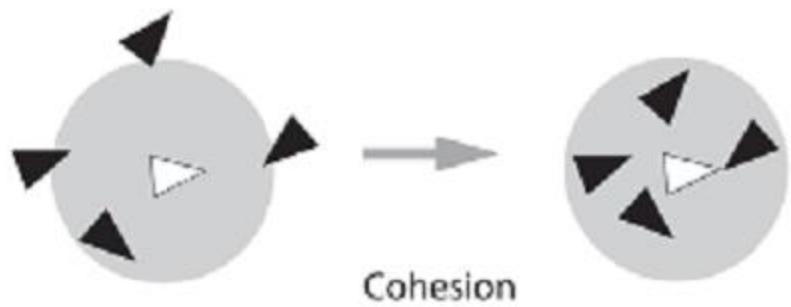


* Average heading and velocity of other boids in neighborhood

http://www.red3d.com/cwr/boids/







Buckland Fig 3.16

Demo

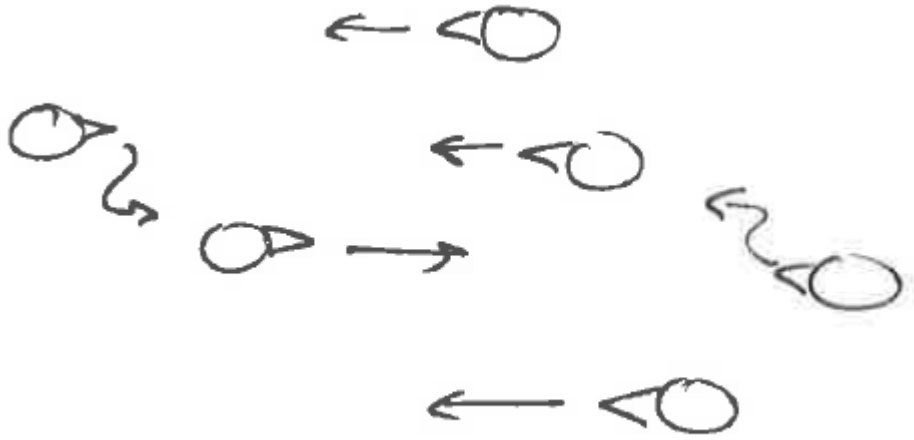
- Flocking (turn on and off)
- Big shoal (blending other behaviors)

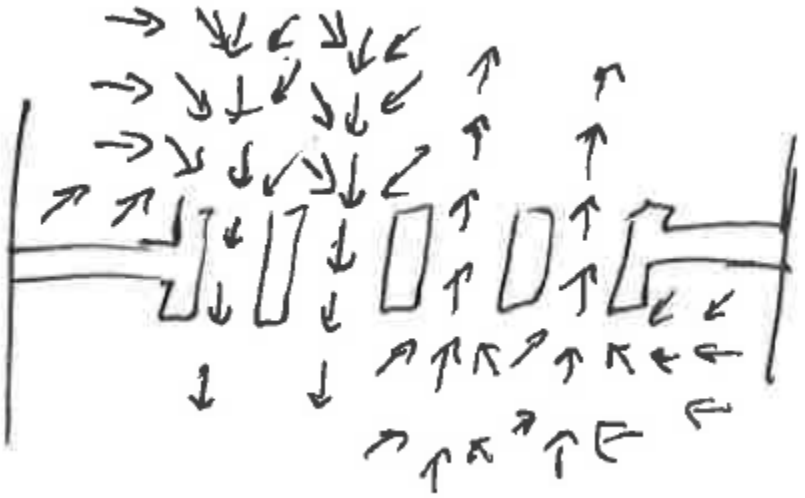
Flocking Demos

- http://www.red3d.com/cwr/boids/
- http://www.red3d.com/cwr/boids/applet/
- Kevin's: https://drive.google.com/file/d/15yiXwCqgCyxQrJvjTf_ymygOtCAKiM7o/view
  - https://piazza.com/class/jc71bf1wrdg6d6?cid=155

Other approaches worth noting

- Lane formation
- Smart maps / smart environments
  - choke points are particularly problematic: kitchen door in restaurant
  - Navigation fields provide authorial control





See Also

- M Ch 3, B Ch 3 (& Ch 1)
- Source from Millington
  - https://github.com/idmillington/aicore
- Source from Buckland
  - http://www.jblearning.com/catalog/9781556220784/
- Java-based animations (combined behaviors)
  - http://www.red3d.com/cwr/steer/
- http://www.cse.scu.edu/~tschwarz/coen266_09/PPT/Movement%20for%20Gaming.ppt