Coordinated Agent Movement

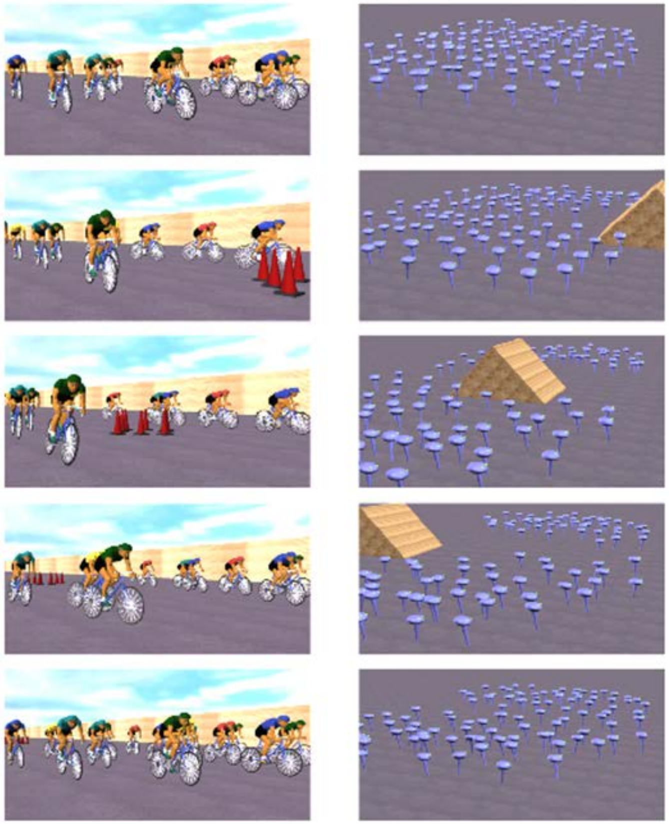
Jeff Wilson

Coordinated Movement

- Somewhat more difficult than moving just one NPC
  - Disappearing goal (what now?)
  - New obstacles in path (re-plan path)
  - Collisions with other NPCs
  - Groups of units (get stuck in tight spaces?)
  - Units in formation

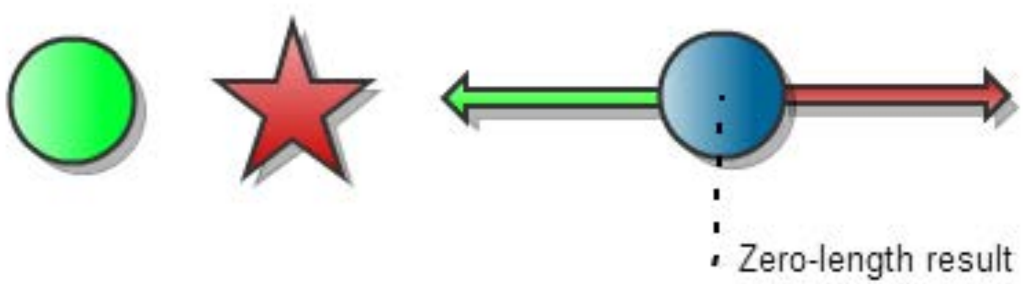
Group Behaviors

- Lots of background characters to create a feeling of motion
- Make area appear interesting, rich, populated

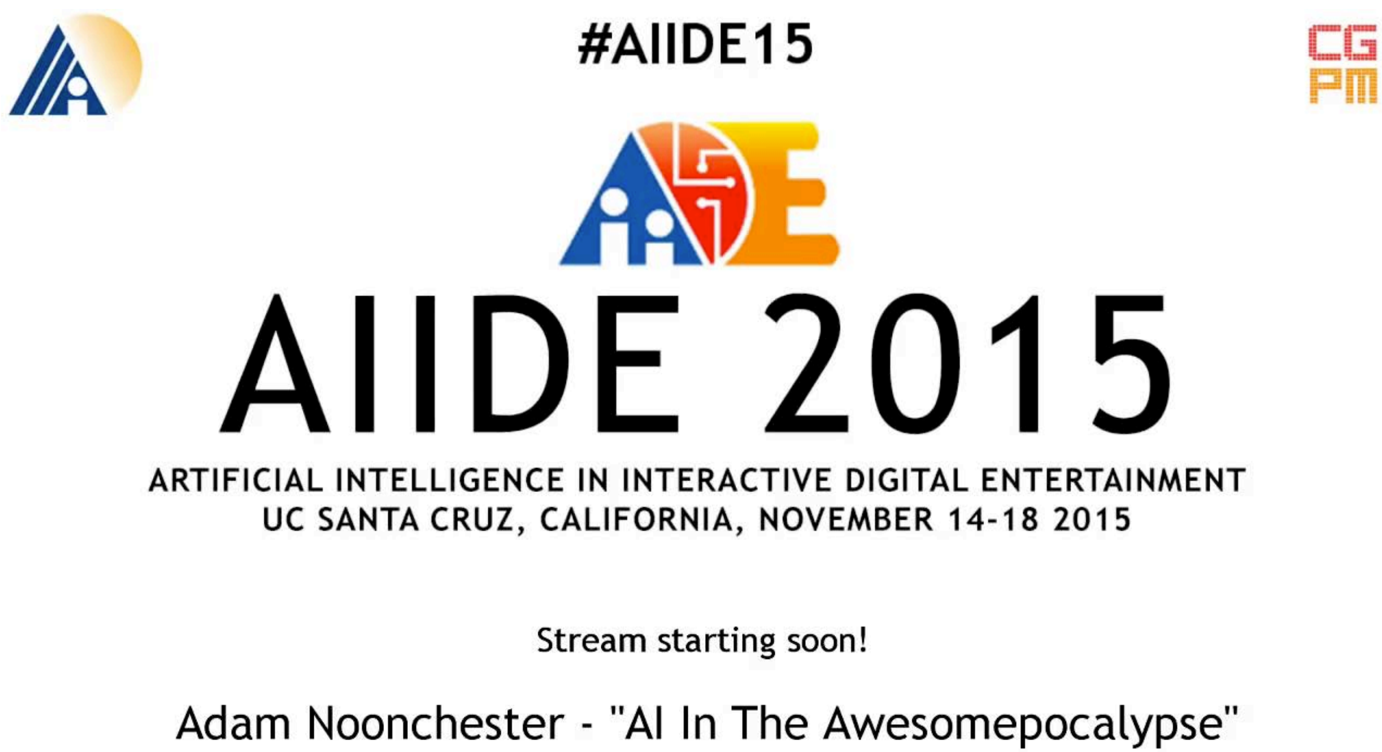


Swarming

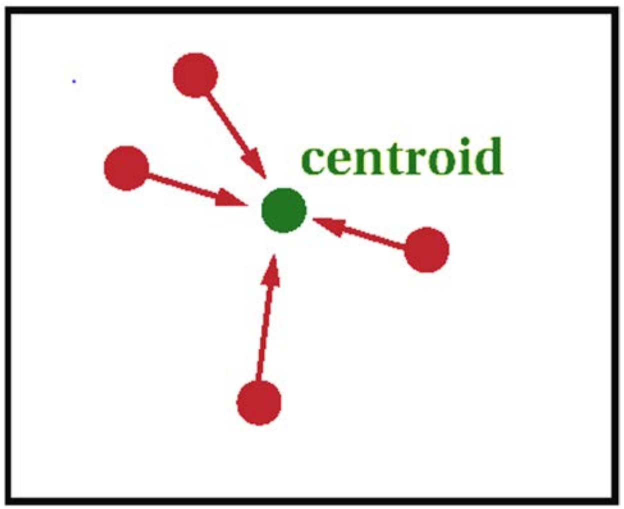
- Goal seeking (common) and obstacle avoidance (other agents with same goal)
- Problems?
  - Bunching up
  - Starting and stopping (vicious cycle)



Swarming


#AIIDE15
AIIDE 2015
ARTIFICIAL INTELLIGENCE IN INTERACTIVE DIGITAL ENTERTAINMENT
UC SANTA CRUZ, CALIFORNIA, NOVEMBER 14-18 2015
Stream starting soon!
Adam Noonchester - "AI In The Awesomepocalypse"

Flocking -- (HalfLife, Unreal)



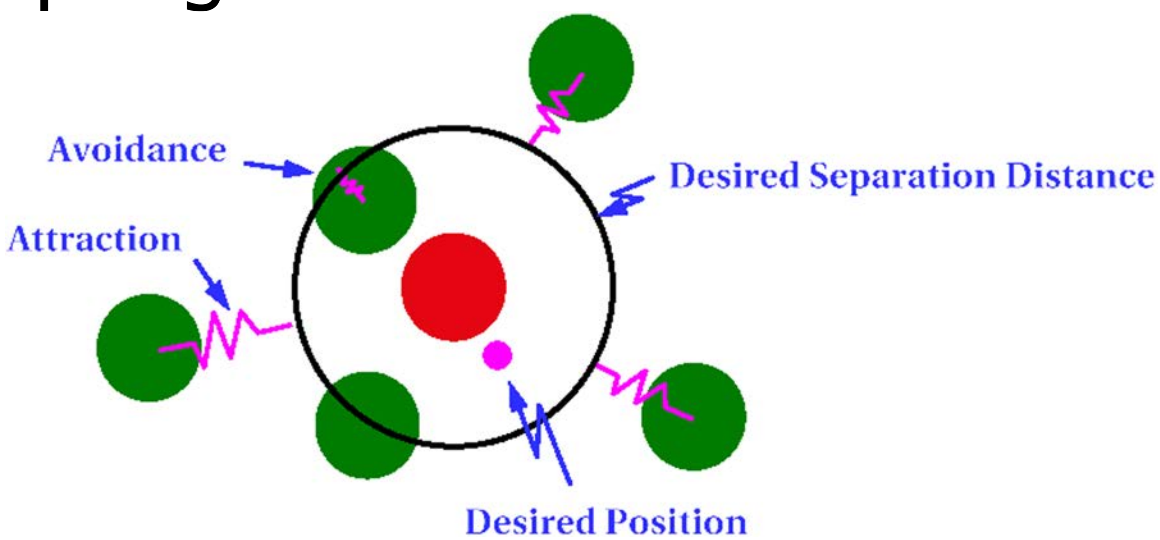
Simple version:
Compute trajectory to head towards centroid

- What might go wrong?

Group Behaviors

Craig Reynolds
SIGGRAPH 1987

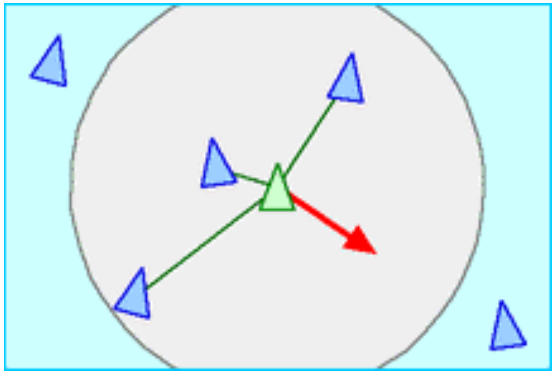
- Reaction to neighbors -- Spring Forces

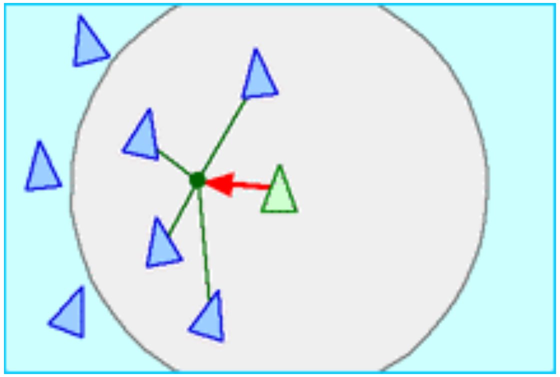


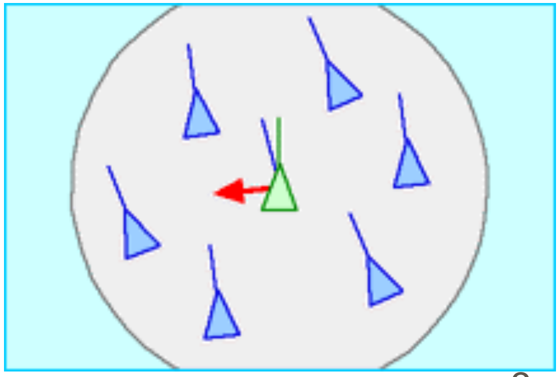
Desired Velocity = current velocity $+ k_p$(error in position)
$+ k_v$(current velocity – nominal velocity)

Boids: Three Steering Mechanisms

- Separation
- Cohesion
- Alignment







Boids: efficiently find neighbors

- Most engines provide a rapid “nearest” function for objects
- Spatial partitioning w/ special data structures:
  - Quad-trees, oct-trees
- Otherwise, comparing all pairs
- Cache neighbor distances as you visit each boid (distances same, just different perspective)

Boids: Brute Force

- Each boid evaluates all other boids to determine neighborhood
- $O(n^2)$ - n is number of boids

Boids – bin-lattice

- Spatial sub-division
- When boid moves, check to see if it is in a new bin (update accordingly)
- *O(n k)* – *k* is number of surrounding bins to consider

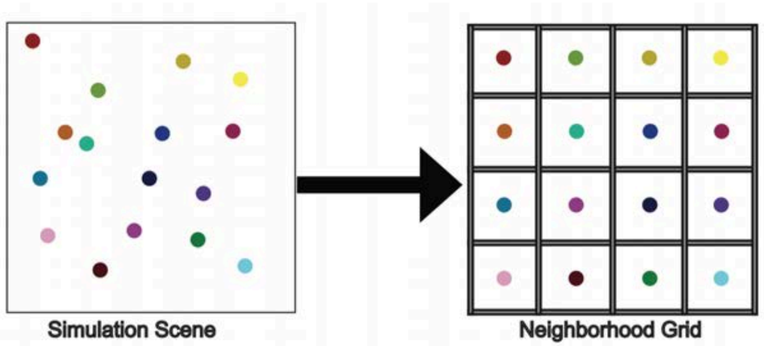


Figure 2: *Construction of the Neighborhood grid in a top-down camera.*

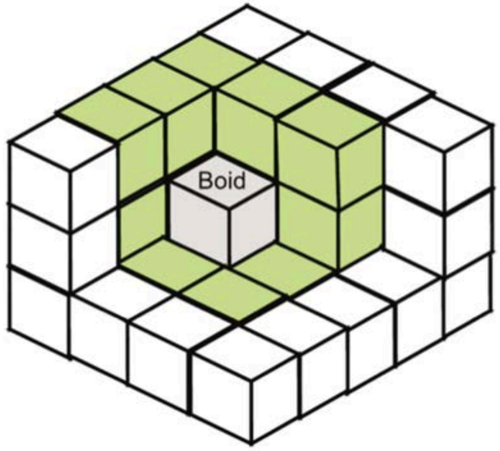
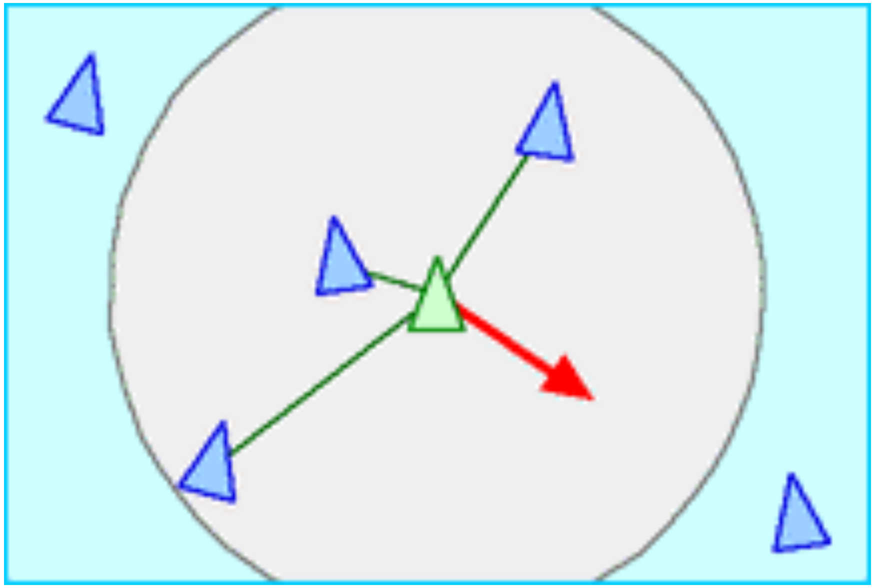
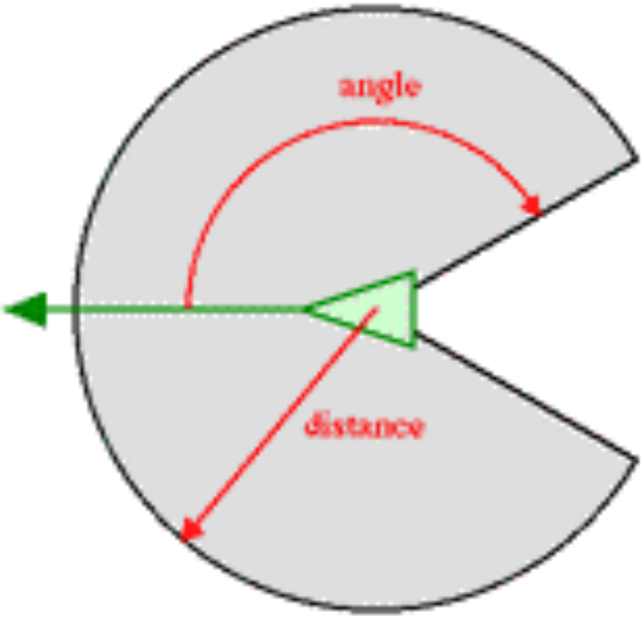


Figure 3: *Example of the neighborhood grid with radius = 1.*

Separation

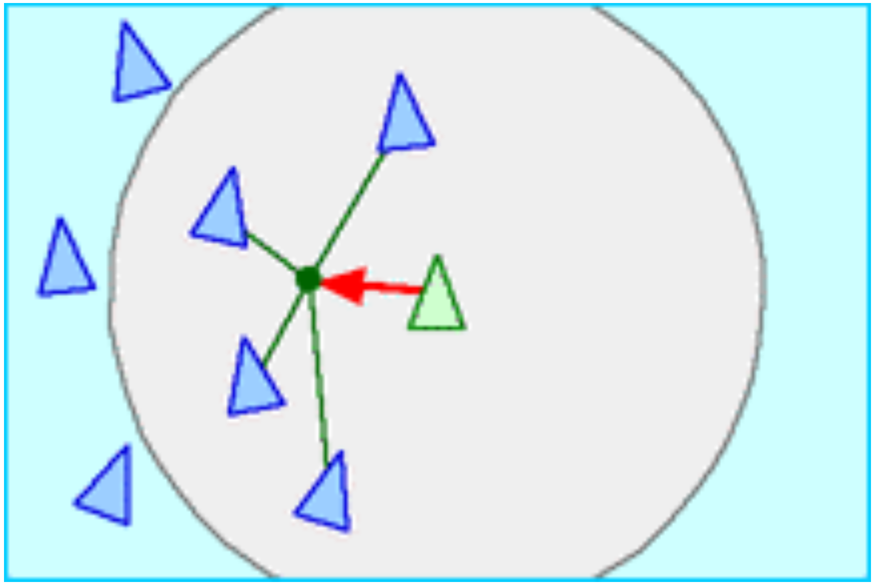
- Steer to avoid crowding local flockmates
  - Force to steer bot away from neighbors
  - Neighborhood is a sphere of a certain radius, or possibly a cone of perception
  - The vector to each bot under consideration is normalized, divided by the distance to the neighbor, and added to the steering force





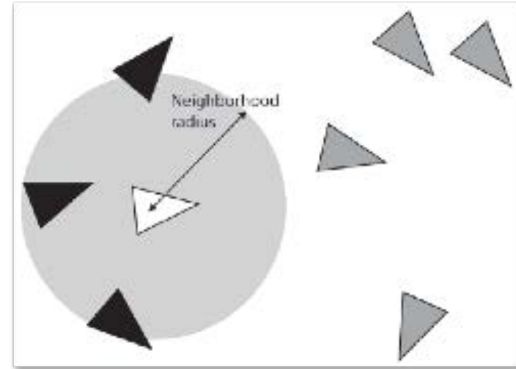
Boids: Cohesion

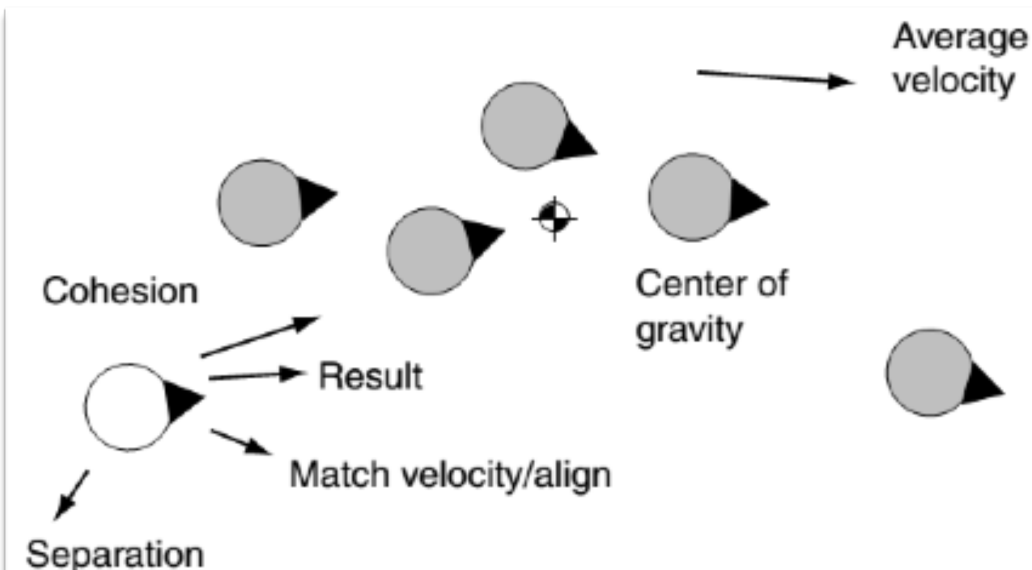
- Steer to average **position** (center of mass) of local flockmates
  - Desired position (center of mass): iterate through all neighbors and average their positions
  - Seek to desired position

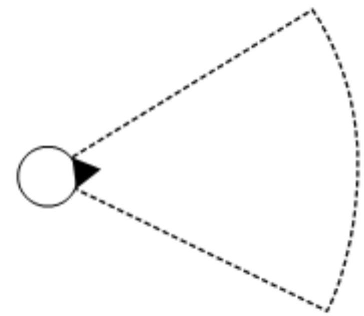


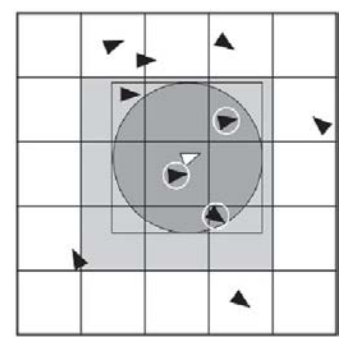
* Center of mass is the average position (X,Y,Z) of boids in neighborhood.

Boids: Cohesion



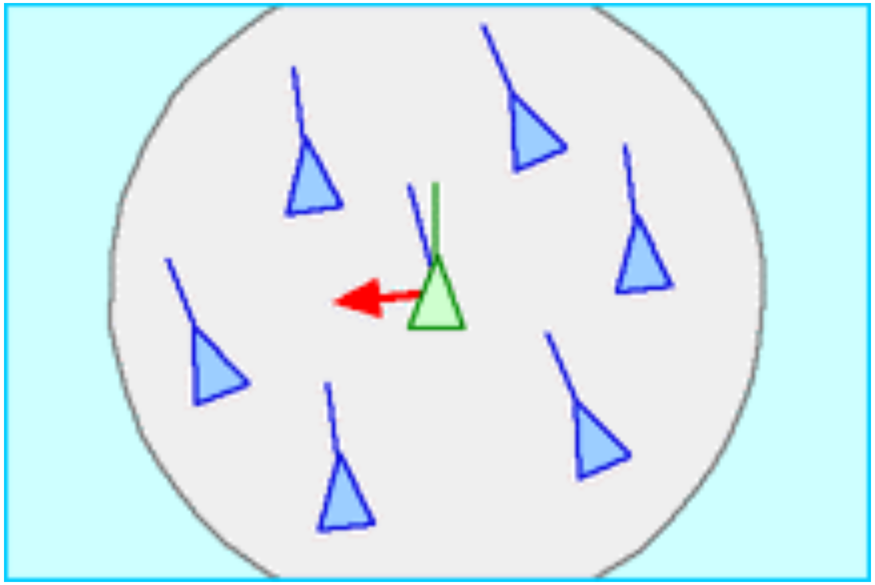






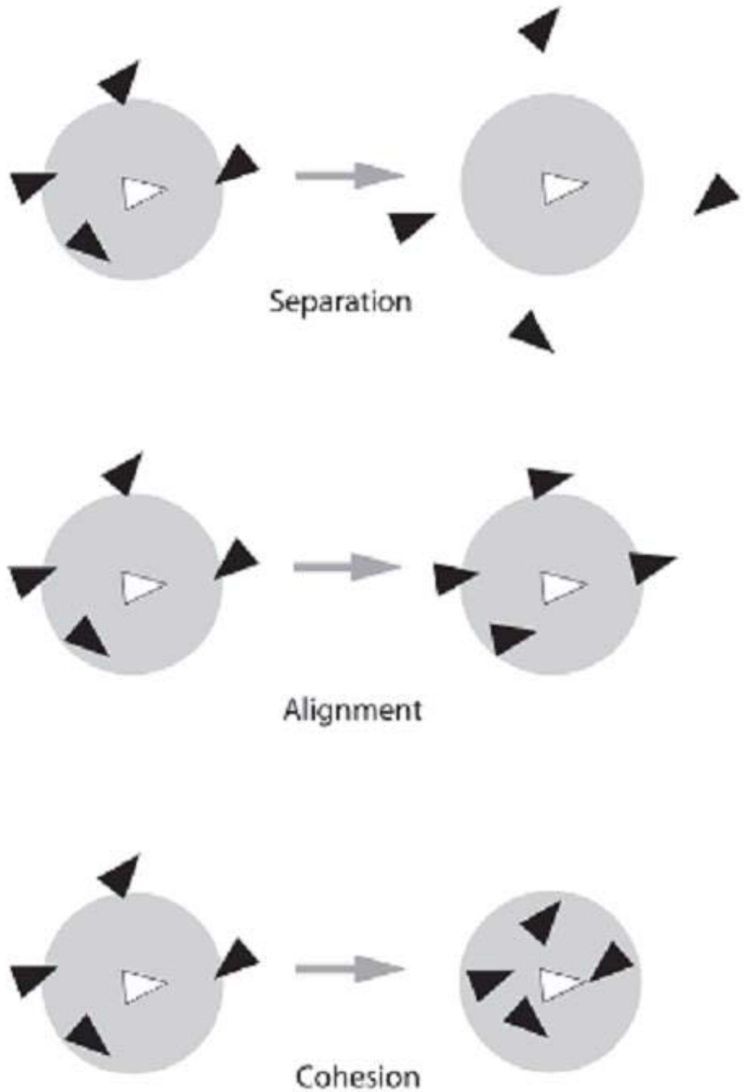
Boids: Alignment

- 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

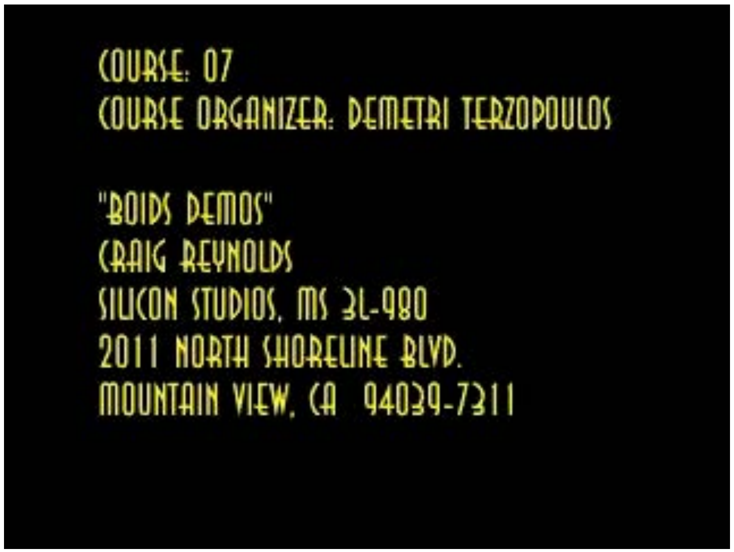


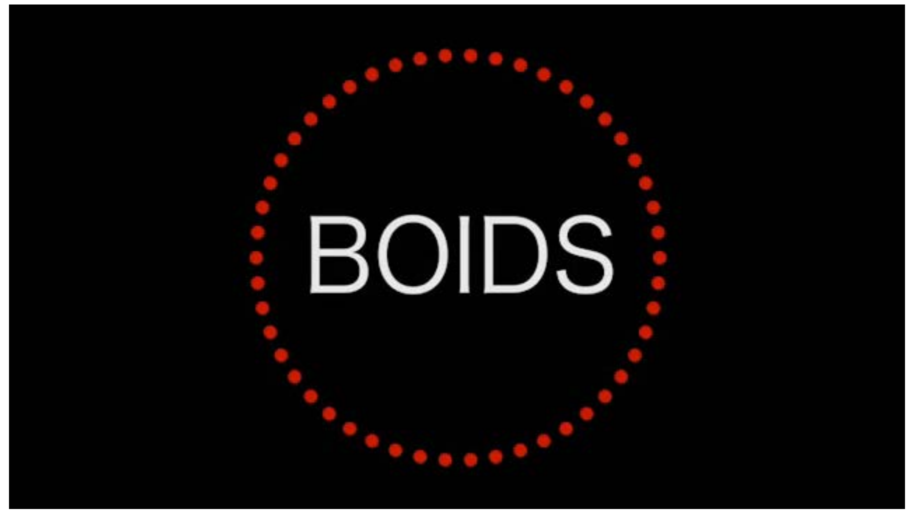
Average heading and velocity of other boids in neighborhood

Boids: Separation, Alignment, Cohesion



Boids

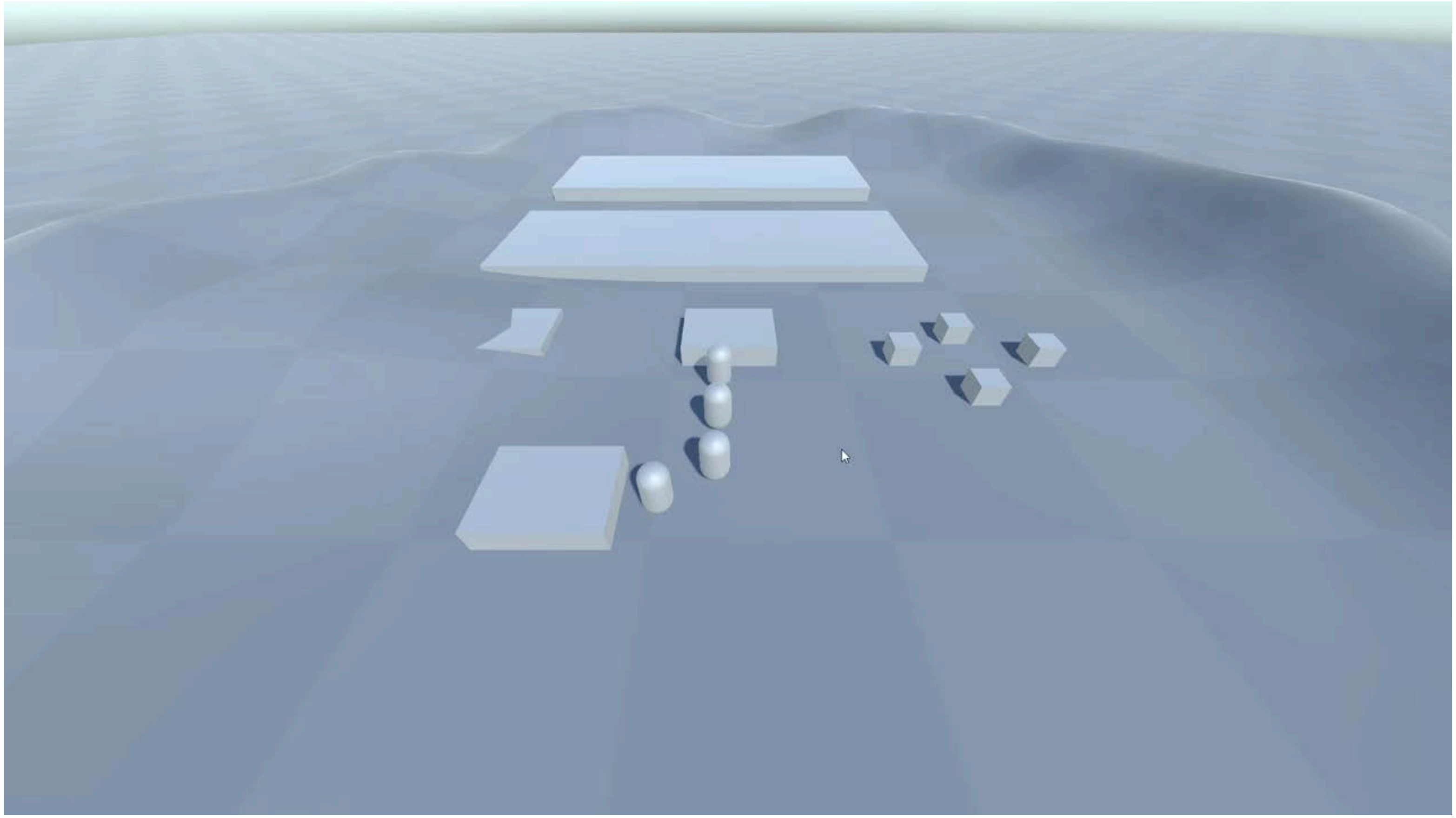

COURSE: 07
COURSE ORGANIZER: DEMETRI TERZOPOULOS
"BOIDS DEMOS"
CRAIG REYNOLDS
SILICON STUDIOS, MS 3L-980
2011 NORTH SHORELINE BLVD.
MOUNTAIN VIEW, CA 94039-7311


BOIDS

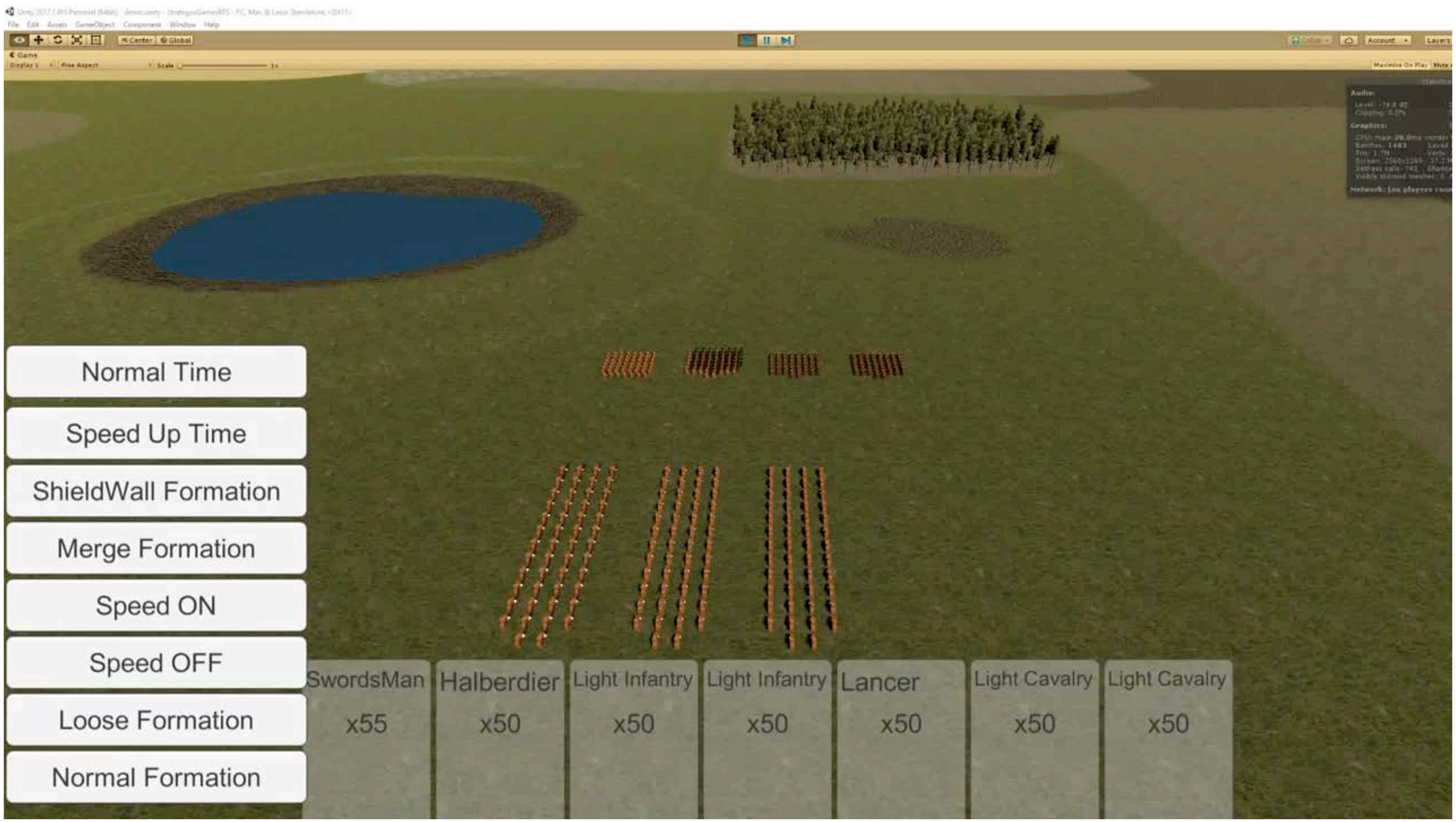
Units, Groups, Formations

- Unit
  - An individual moving NPC
- Group
  - A collection of units
- Formation
  - A group with position assignments per group member

Example



Example



Normal Time
Speed Up Time
ShieldWall Formation
Merge Formation
Speed ON
Speed OFF
Loose Formation
Normal Formation
SwordsMan x55
Halberdier x50
Light Infantry x50
Light Infantry x50
Lancer x50
Light Cavalry x50
Light Cavalry x50

Coordinated Movement

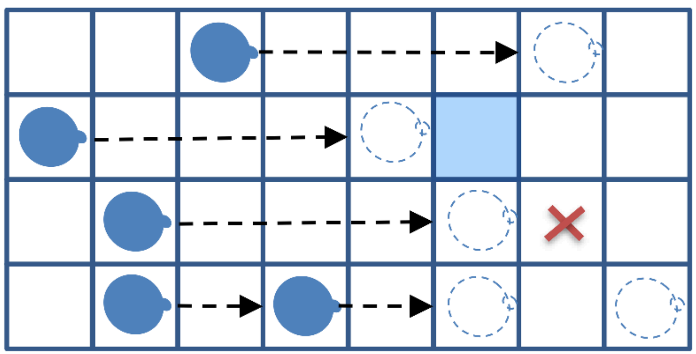
- Options:
  - Individuals plan/move completely independently, but share common goal
  - Individuals make complementary decisions
    - Suggests some shared information/communication
  - Group makes decision as a whole
    - Individual units may work to follow group decision

Moving as a group

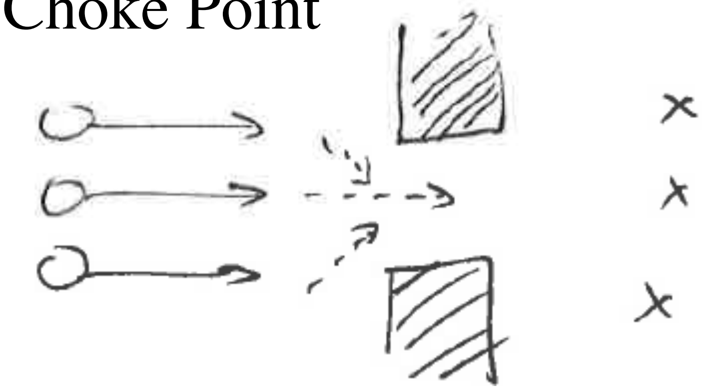
- Independent – shortest distance to target
  - Often results in traffic jam as units attempt to occupy same locations
- Traffic jam impacts:
  - Agents might stop
  - Agents may path plan around teammates
    - Possibly moving away from pending traffic jam leaving empty space that no agent uses!

Simple Offsets

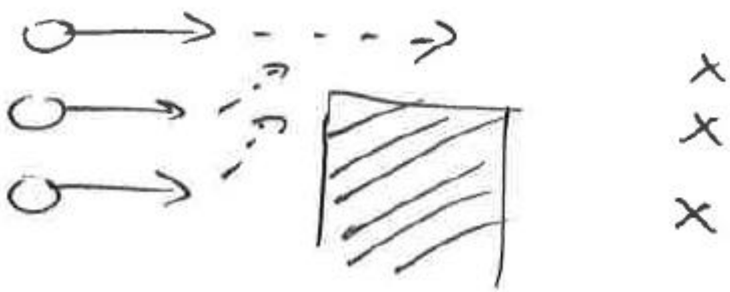
- Individual agents maintain offset to goal relative to current group pos centroid



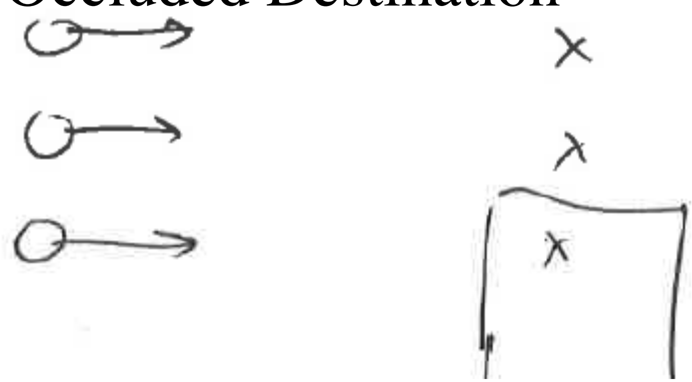
Choke Point



Obstruction

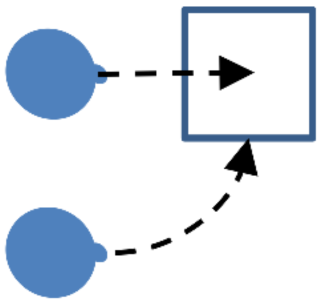


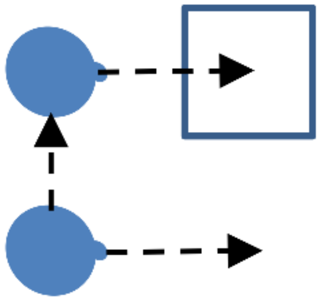
Occluded Destination



Re-planning vs Waiting

- NPC *locks* current cell (best with grid lattice)
  - Other agent perceives locked cell as obstacle
- Re-planning may cause other agent to move in a diff. dir, later to return to orig. cell (assumed re-planning whenever cell states change)
- Use heuristic to recognize this, and simply make blocked agents wait



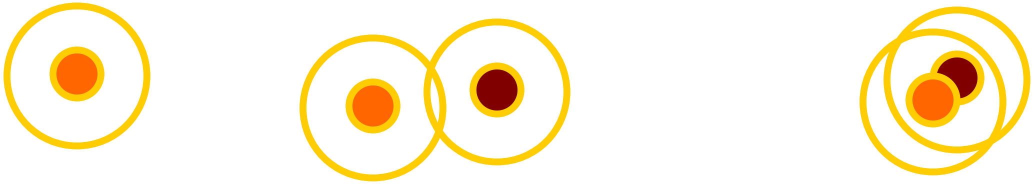


Coordinated Elements

- Collision detection
  - Detection of immediate collisions
  - Near future
  - Linear extrapolation, or higher order
- Perform the usual collision detection optimizations
  - Spatial hierarchies
  - Simplified tests
  - Unit approximations

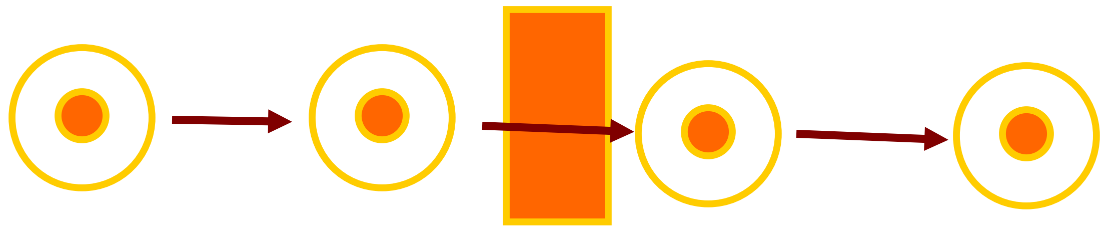
Collision Detection

- Levels of collision
  - Hard radius (small)
    - Must not have 2 units overlap hard radius
  - Soft radius (large)
    - Soft overlap not preferred, but acceptable



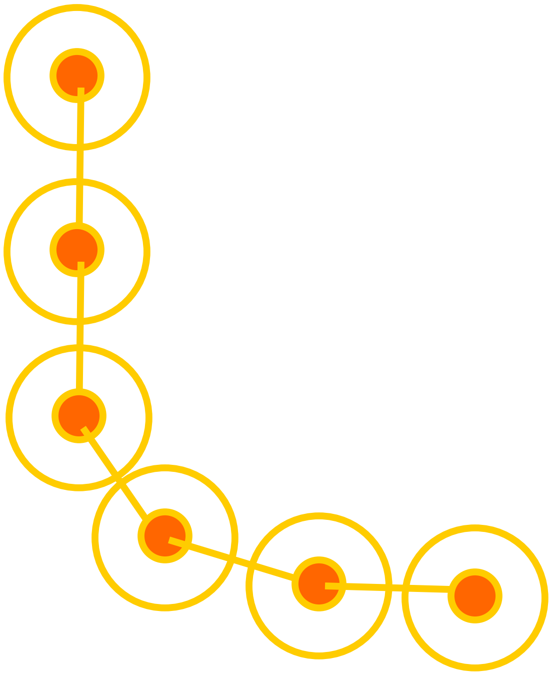
Collision Detection

- With movement, need to avoid problems with bad temporal samples
  - Sample frequently



  - Detect collisions with extruded units
    - Can use raycast(s)
  - Use a movement line
  - Detect distance from Line segment

Unit Line



- Useful if you can't mark map locations as occupied (e.g. not grid lattice)
- Unit line follows path
- Can implement minimum turn radius
- Gives mechanism for position prediction
- Connected line segments
  - Time stamps per segment
  - Orientation per segment
  - Acceleration per segment

Collision Avoidance Planning

- Don’t search a new path at each collision
- Adopt a Priority Structure
  - Higher priority items move
  - Lower priority items wait or get out of the way
- Case-based reasoning to perform local path reordering

Collision Resolution Summary

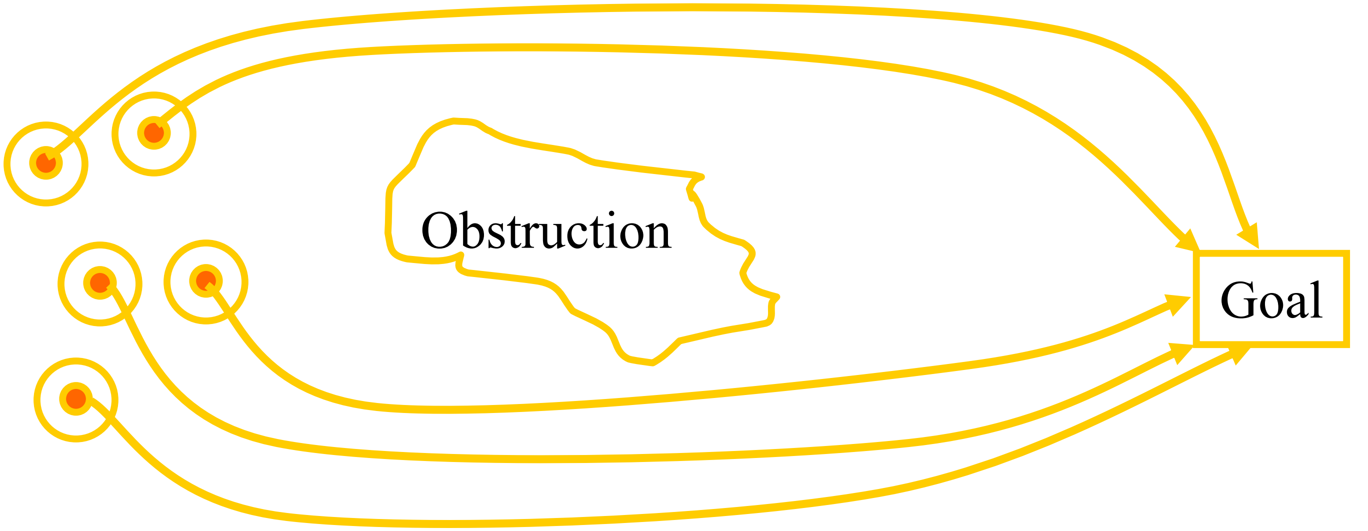
- Favor:
  - High priority NPCs over Low Priority
  - Moving over non-moving
- Lower Priority NPCs
  - Back out of the way
  - Stop to allow others to pass
- General
  - Resolve all high-priority collisions first

Unit Priority

- Giant slow moving units
- More powerful
- Important to game objectives/story
- The *leader*
- Already parked versus already moving
- Unit objectives can provide priority context as well

Groups

- Groups stay together
  - All units move at same speed
    - Slowest unit, slows everyone down
  - All units follow the same general path
  - Units arrive at the same time



Groups

- Can use a hierarchical movement system
- Group structure
  - Manages its own priorities
  - Resolves its own collisions
  - Elects a *commander* that traces paths, etc.
    - Commander can be an explicit game feature

Follow the Leader

- Path plan to goal only for leader
- All other units move toward leader (simple)
  - More advanced: or offset position from leader
  - Fancy: rotate/wheel positions relative to leader dir
- Can be simple steering behaviors for followers, unless they fall too far behind or blocked by static collider
- In this case, path plan back in range of leader. Possibly also communicate to leader to stop/slow down for straggler

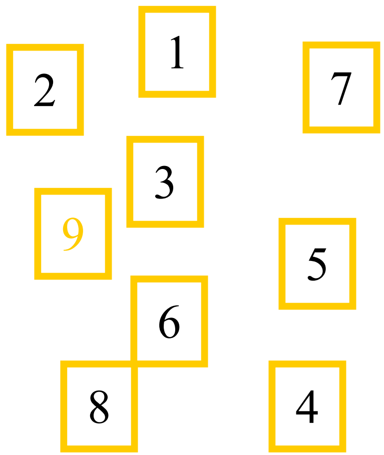
Formations

- Groups with unit layouts
  - Layouts designed in advance
- Additional States
  - Forming
  - Formed
  - Broken
- Only formed formations can move

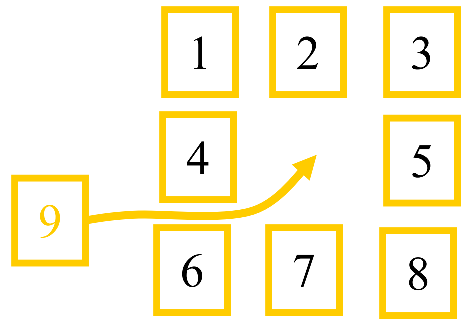
Formations

- Schedule arrival into position
  - Start at the middle and work outwards
  - Move one unit at a time into position
  - Pick the next unit with
    - Least collisions
    - Least distance
  - Formed units have highest priority
    - Forming units medium priority
    - Unformed units lowest

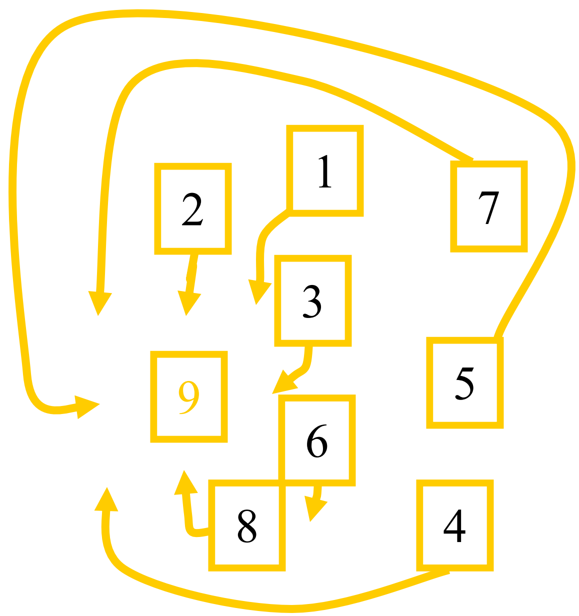
Formations



Not so good…

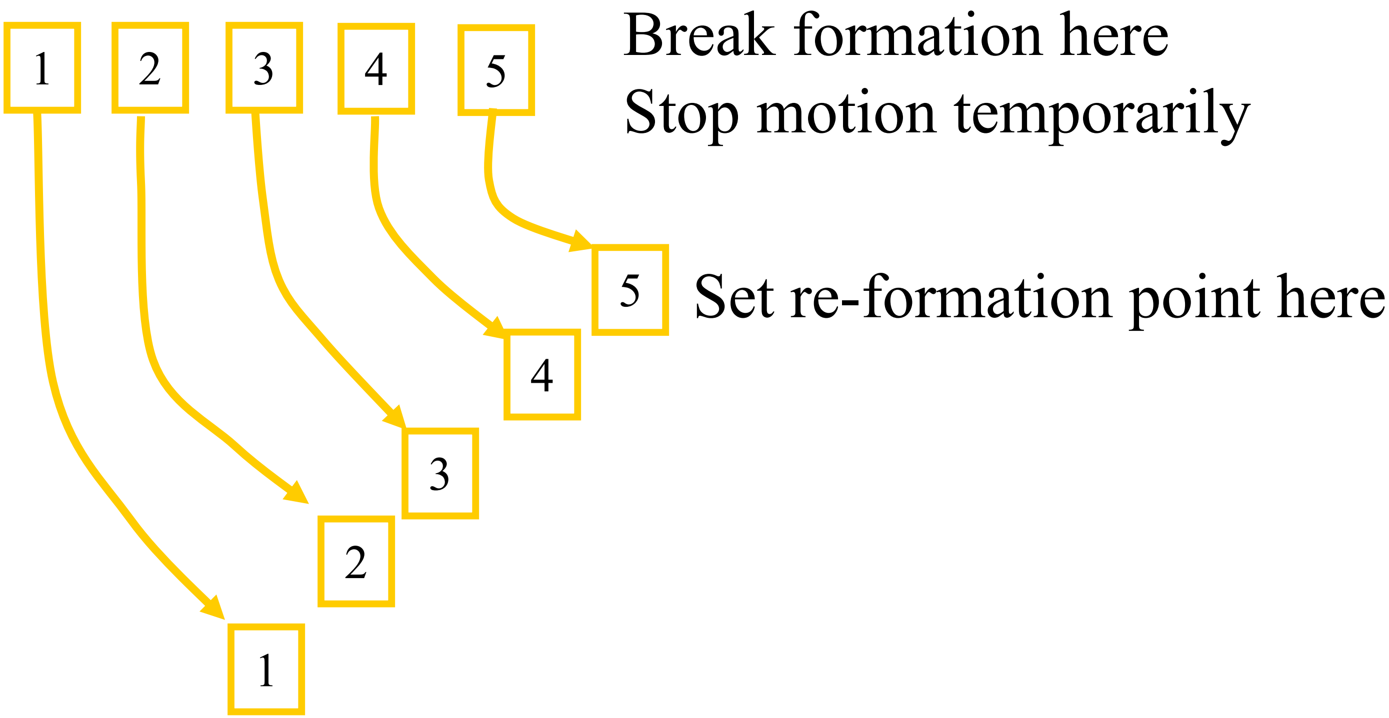


Better…



Formations: Wheeling/Orienting

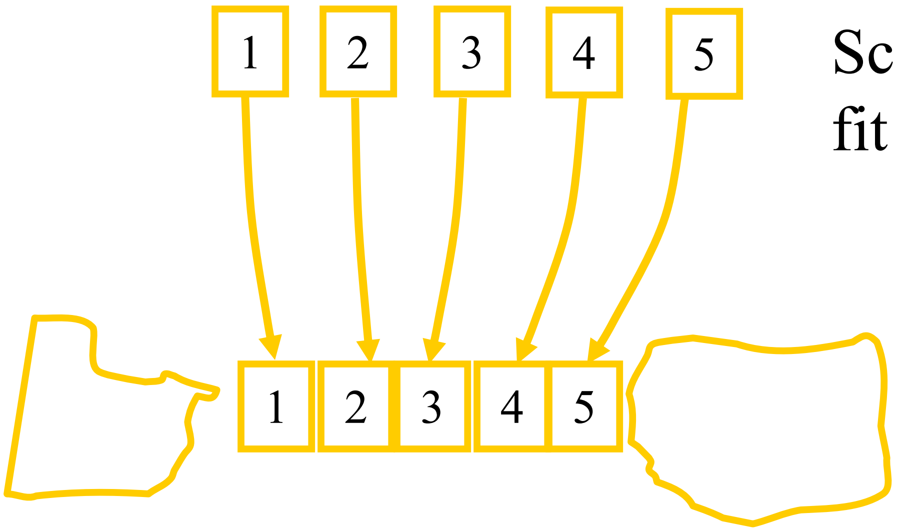
- Only necessary for non-symmetric formations



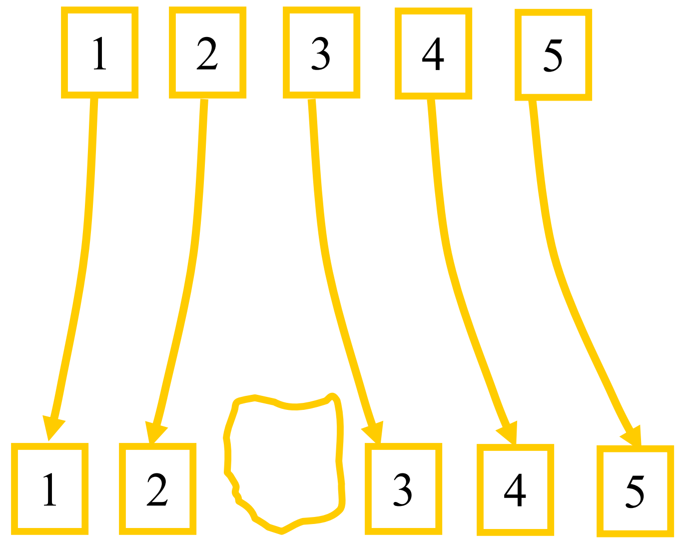
Follow the Leader: Keeping Up

- Slow leader down so that slowest unit can keep up in straight line
- Turning/wheeling can introduce additional demands (outside units of turning formation need to travel faster)
- Throttle translation+rotation speed according to kinematic analysis of formation points

Formations: Obstacles



Scale formation layout to fit through gaps



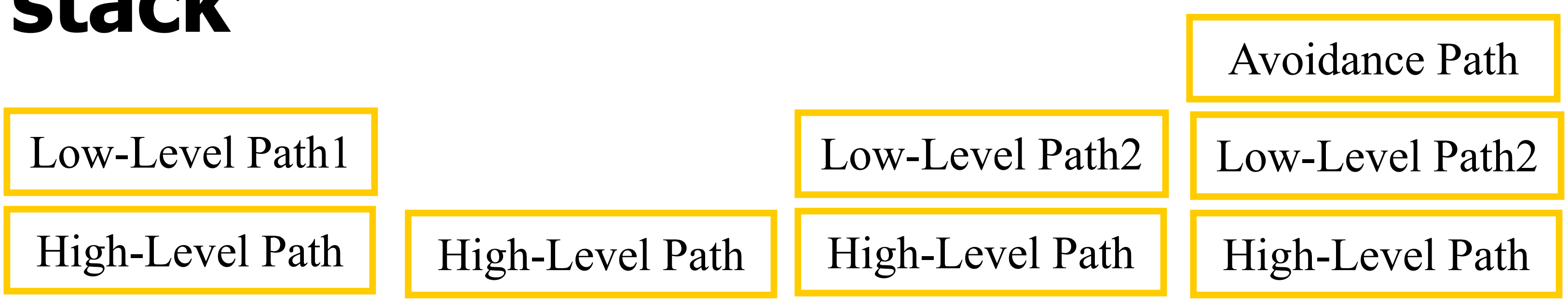
Subdivide formation around small obstacles

Formations

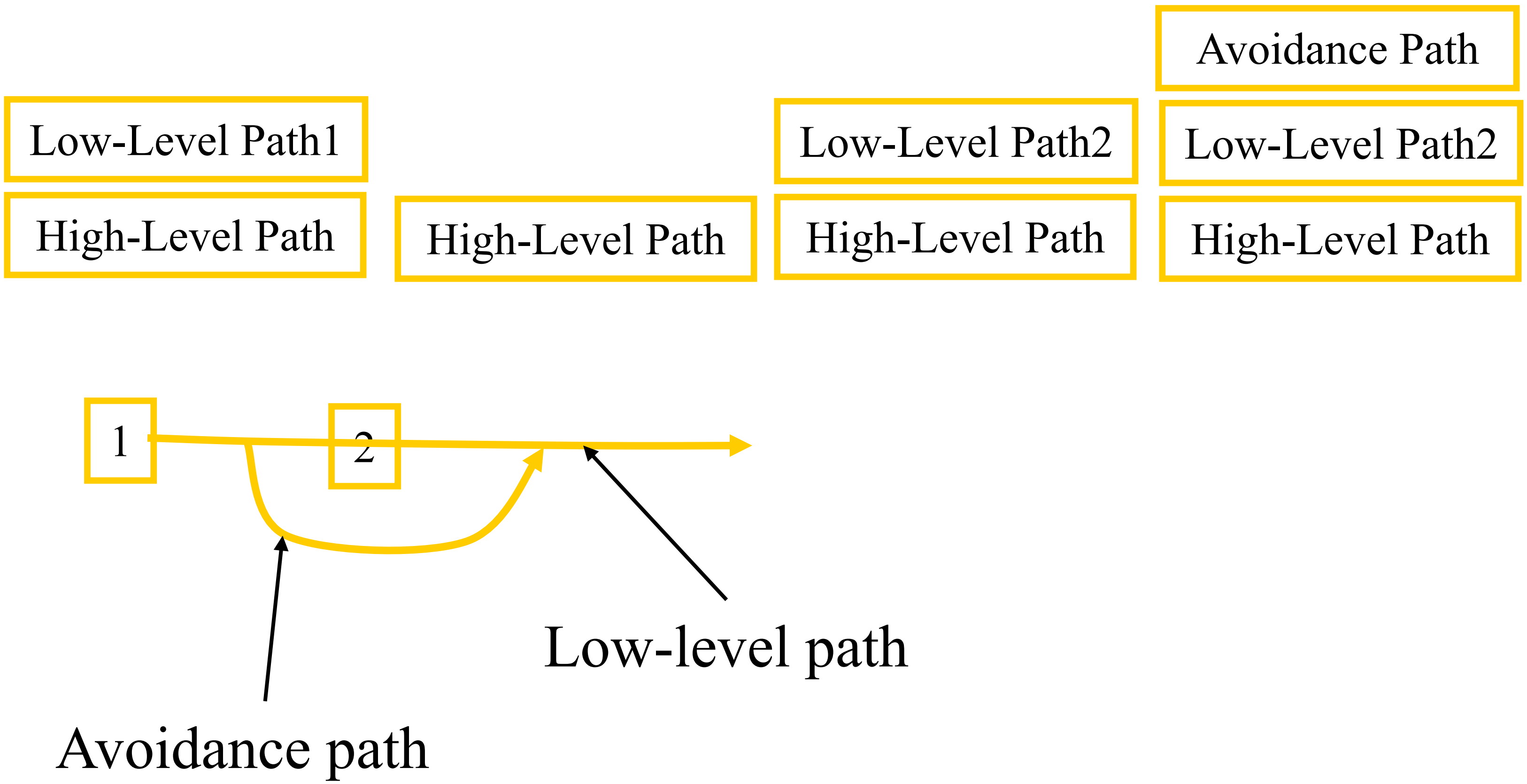
- Adopt a hierarchy of paths to simplify path-planning problems
- High-level path considers only large obstacles
  - Perhaps at lower resolution
  - Solves problem of gross formation movement
  - Paths around major terrain features

Formations

- Low-level path
  - Detailed planning within each segment of high-level path
  - Details of obstacle avoidance
- Implement path hierarchy with **path stack**



Path Stack



Low-Level Path1
High-Level Path
High-Level Path
Low-Level Path2
High-Level Path
Avoidance Path
Low-Level Path2
High-Level Path
1
2
Low-level path
Avoidance path

General

- Use high-level and low-level pathing
- Units will overlap!
- Understand the update loop
  - It affects unit movement