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.

Graphs, Search, Pathfinding

(behavior involving **where** to go)

Static, Kinematic, & Dynamic Movement; Steering, Flocking, **Formations**

(behavior involving **how** to go)

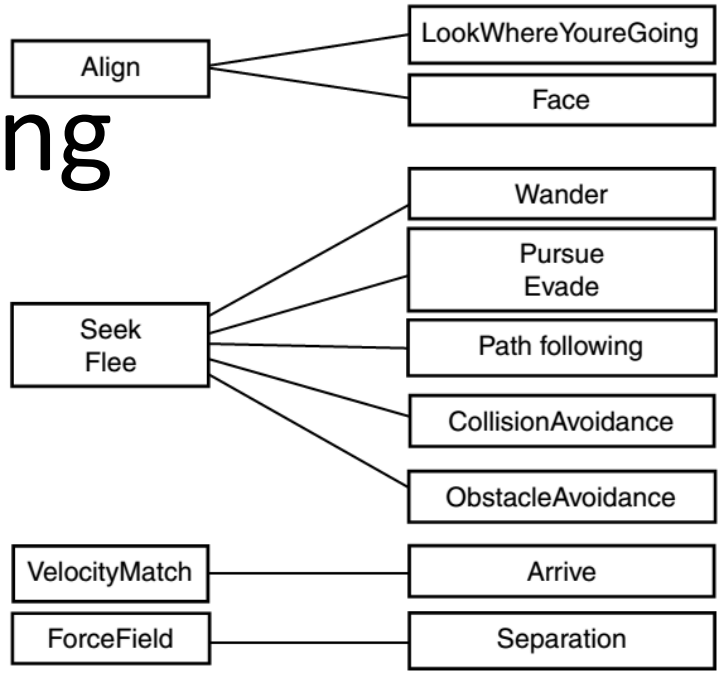
PREVIOUSLY ON…

N-2: Movement & Steering

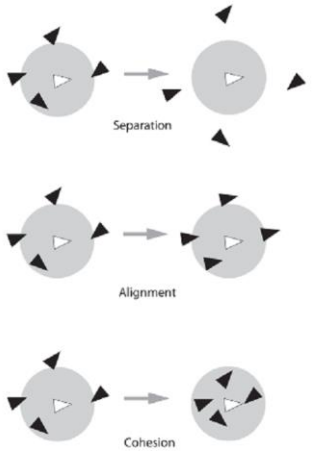
1. What do movement algorithms output in static environ?
2. What do movement algorithms input in kinematic environ?
3. What do movement algorithms output in kinematic environ?
4. What is the deal with time & variable frame rates?
5. What was the insight about updates if time << 1?
6. How are kinematic 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?

N-2, 1: Flocking, Steering

1. Steering vs flocking vs swarming?
2. Steering Family Tree
3. How might we combine behaviors?
4. Can we be sure combinations work?
5. What three steering mechanisms enable flocking?
6. Spatial partitioning w/ special data structures: Why? How?



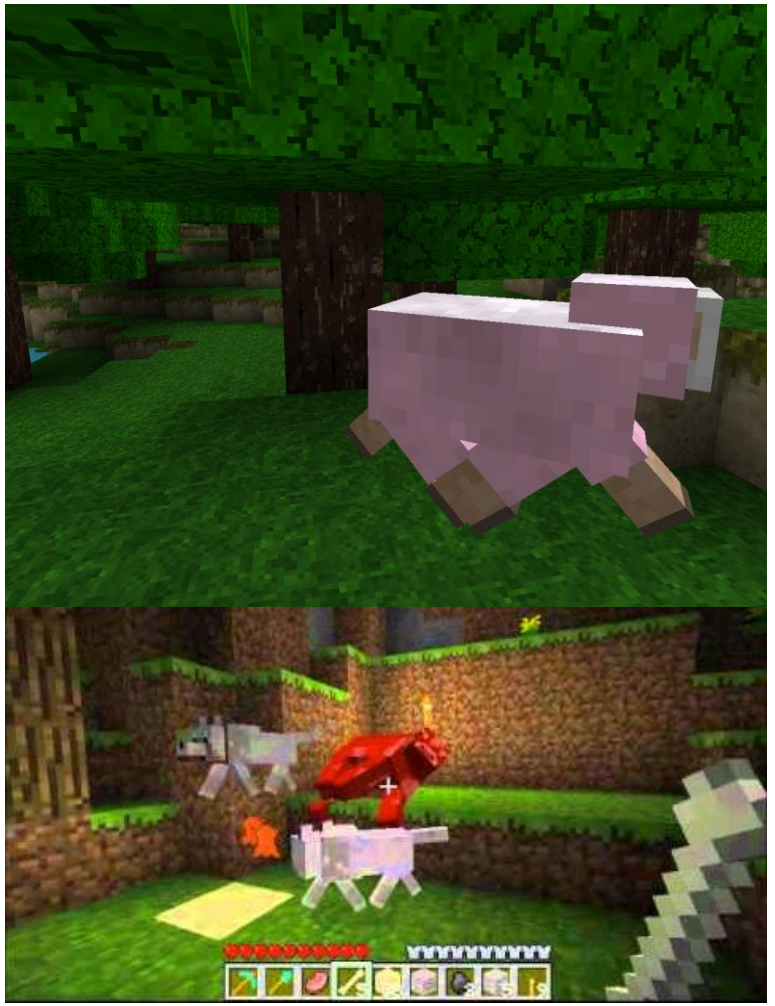
Millington Fig 3.29



Buckland Fig 3.16

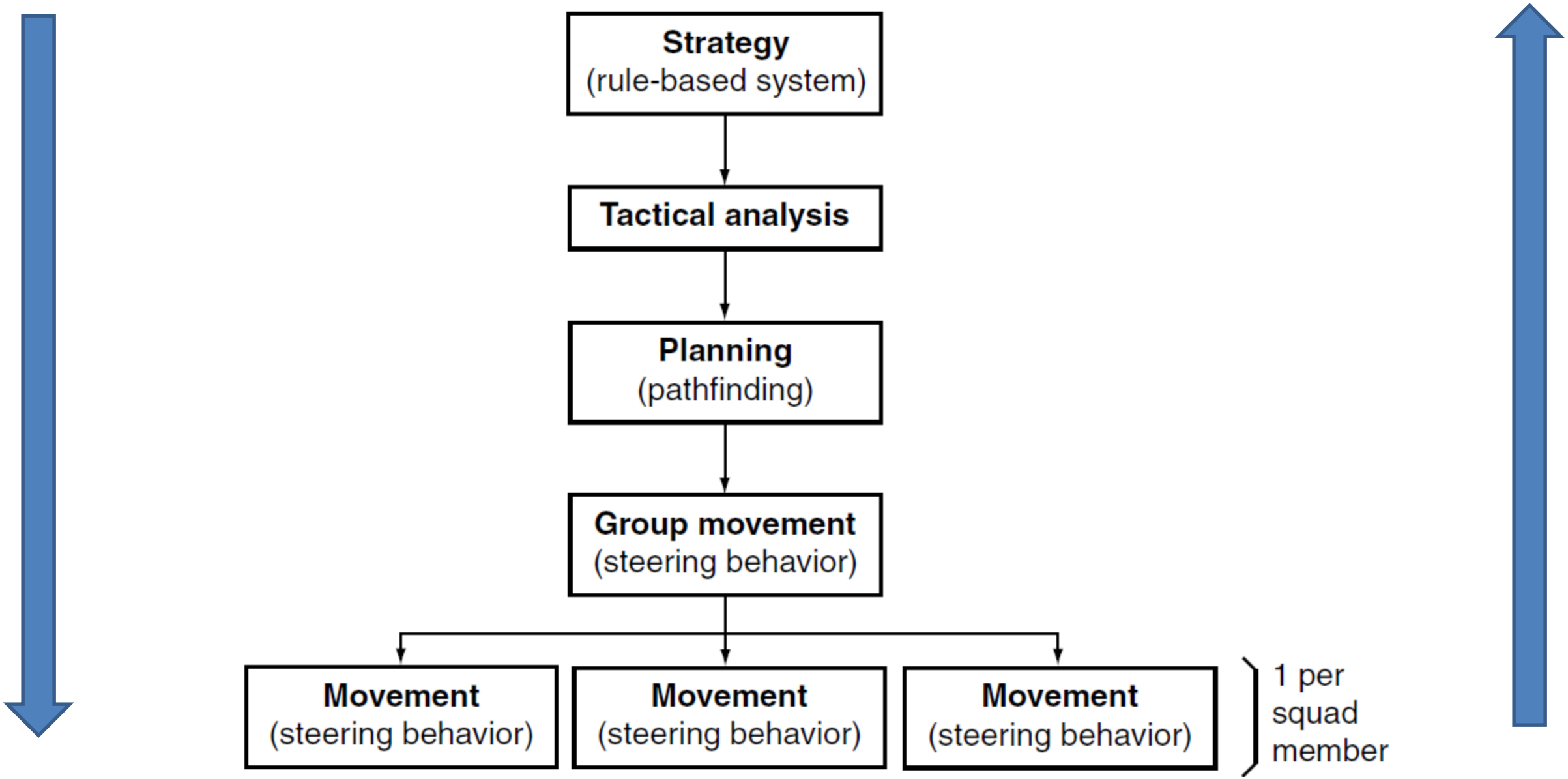
Combining Steering Behavior

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

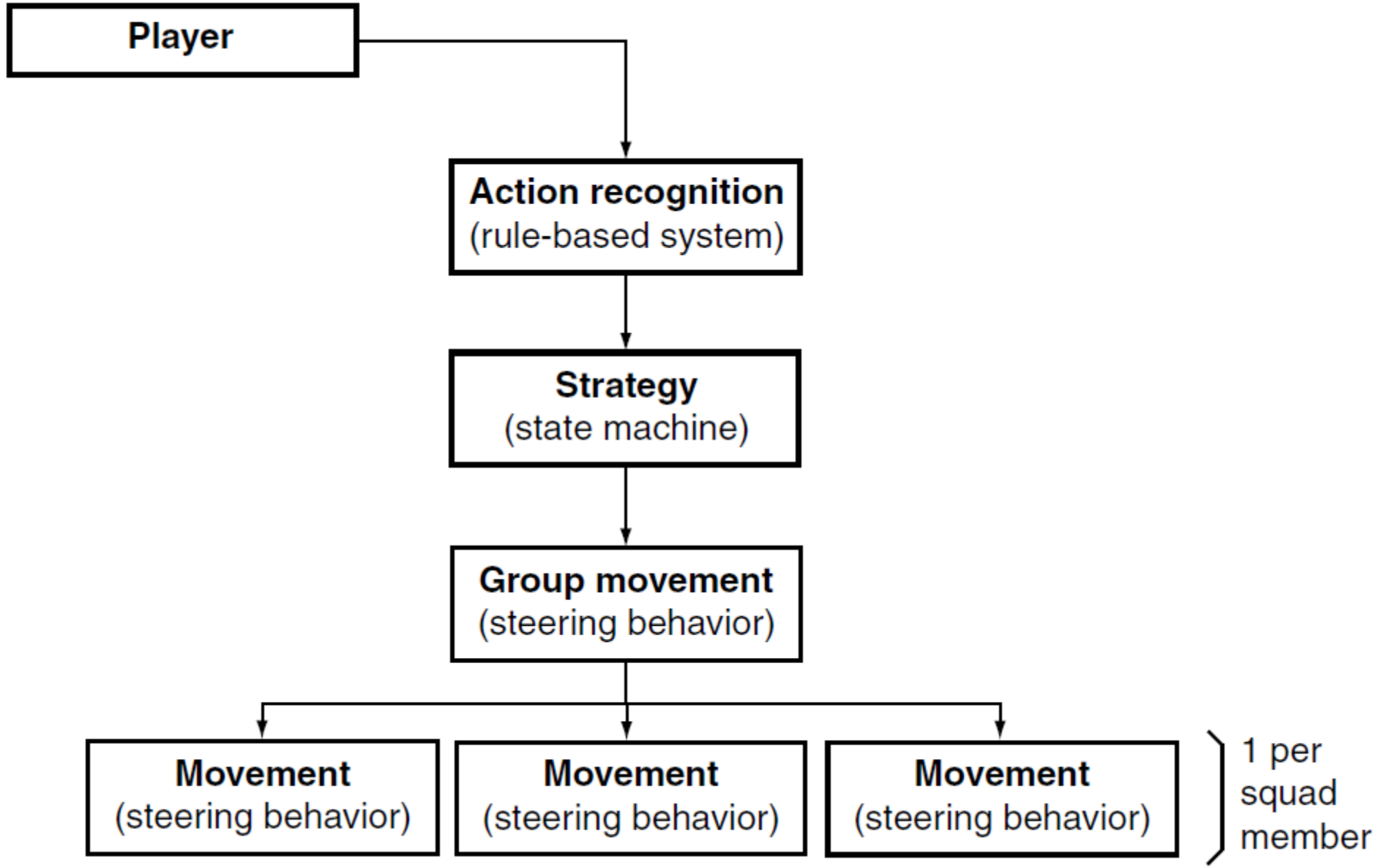


Disclaimer: I use these notes as a guide rather than a comprehensive coverage of the topic. They are neither a substitute for attending the lectures nor for reading the assigned material.

Formations


Strategy
(rule-based system)
Tactical analysis
Planning
(pathfinding)
Group movement
(steering behavior)
Movement
(steering behavior)
Movement
(steering behavior)
Movement
(steering behavior)
1 per
squad
member

M 6.23


Player
Action recognition
(rule-based system)
Strategy
(state machine)
Group movement
(steering behavior)
Movement
(steering behavior)
Movement
(steering behavior)
Movement
(steering behavior)
1 per
squad
member

M 6.25

Coordinated Movement: Formations

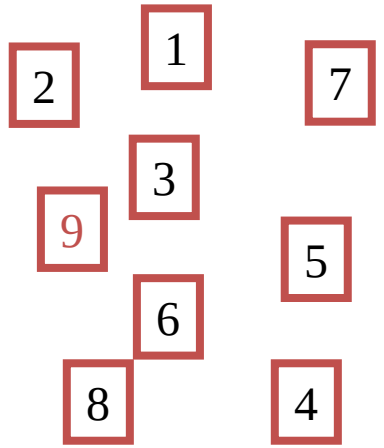
- Coordinated Movement: M Ch 3.7 (B Ch 3, “offset pursuit”)
  - movement of a group of characters so that they retain some group organization
- Two main options:
  - Individuals make complementary decisions (bottom up)
  - Group makes decision as whole and move in prescribed group (top down)
- “**Formation motion**” or “**formation steering**”: Easier to write, more stable, simpler execution than collaborative tactical decision making
  - For now, assume we already made the decision to move together
  - Today we investigate ways to move groups of characters in a cohesive way

Formations: The Idea

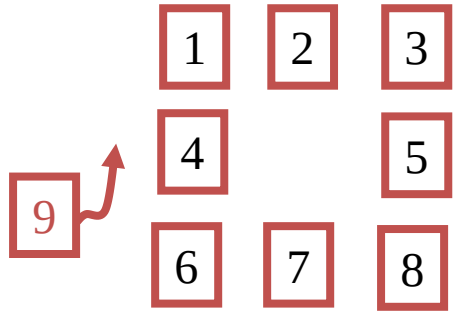
- Groups with unit layouts
  - Layouts designed in advance
- Additional States
  - Forming
  - Formed
  - Broken
- Usually, only formed formations can move

Formations: Forming

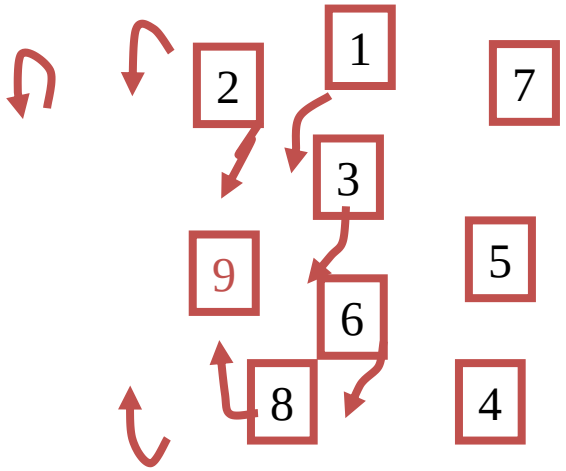
- Schedule arrival into position
  - Start at the middle and work outwards
  - Move one unit at a time into position
  - Pick the next unit with
    - Least collisions
    - Least distance
  - Formed units have highest priority
    - Forming units medium priority
    - Unformed units lowest



Not so good...

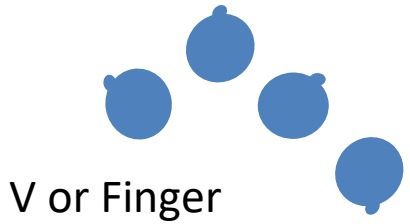


Better...

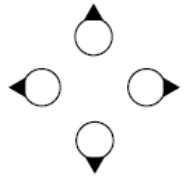


Types of formations

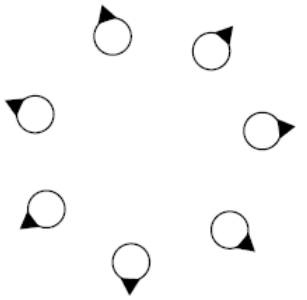
- Fixed
- Scalable
- Emergent
- Multi-level formation w/ anchor points



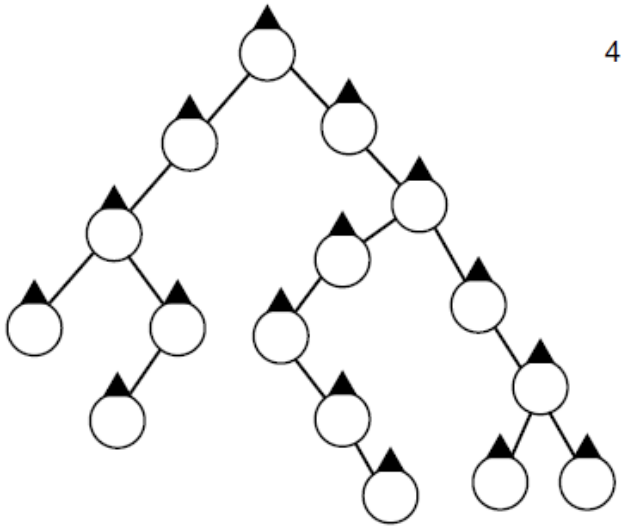
V or Finger
Four



4 characters

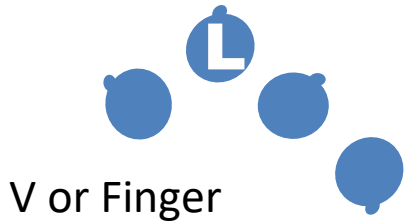


7 characters



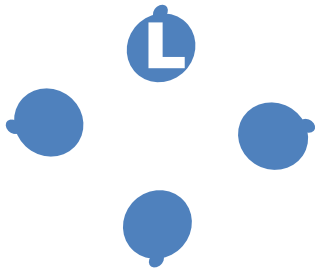
Types of formations: **Fixed**

- Defined by a set of (positioned) slots
- One slot marked as leader
- All other slots defined relative to leader
- Leader moves ignoring that it is in formation
- As leader moves, the pattern/formation moves in unison (we'll see different ways)

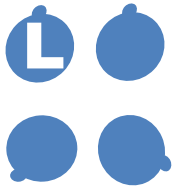


V or Finger Four

Line



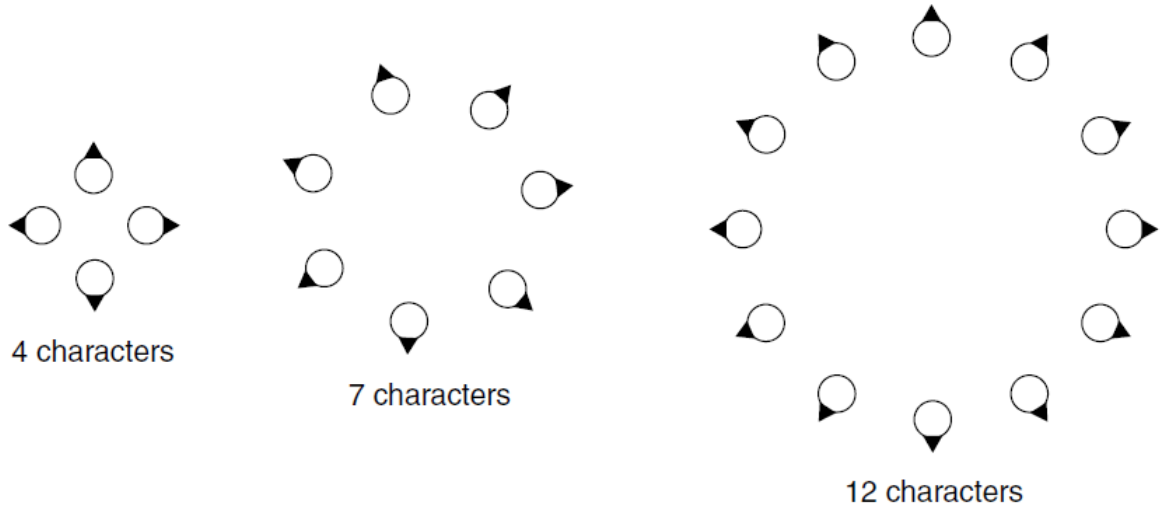
Defensive circle



Two abreast in cover

Types of formations: **Scalable**

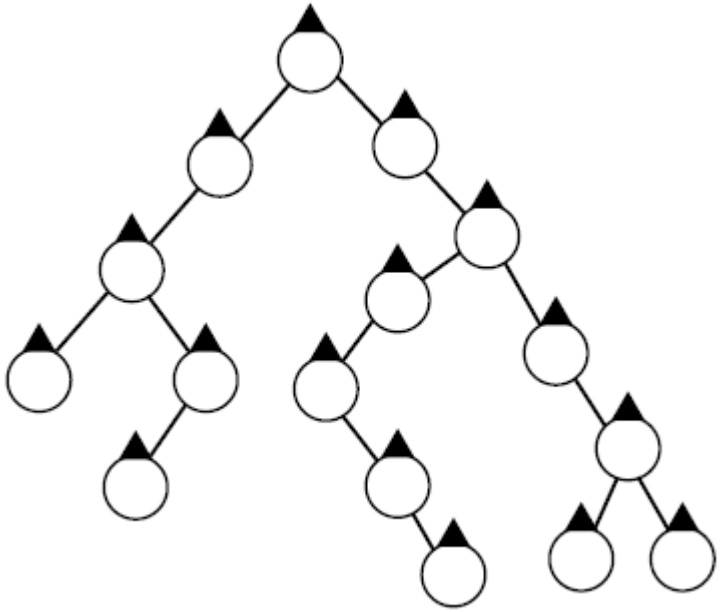
- Structure of formation depends upon number of characters participating
- Implemented without explicit list of slot positions and orientations
  - Function to dynamically determine



M&F 3.55

Types of formations: **Emergent**

- Another solution to scalability
- Each character has own steering, using arrive behavior
- Target for arrive selected based on position of other characters in group. E.g. for V:
  - each char chooses target in front of it, and selects arrive target behind and to side
  - if location is already chosen another target is chosen.
  - Movement updates position and orientation based on this target
  - New target selected when movement cannot be achieved (e.g. obstacle)
- Leader election is optional



M&F 3.56

Emergent wins and losses

Advantages

- Formation emerges from the individual rules of each character, like flocking
- Each char can react individually to obstacles and potential collisions
- No need to factor in the size of the formation when considering turning or wall avoidance

Disadvantages

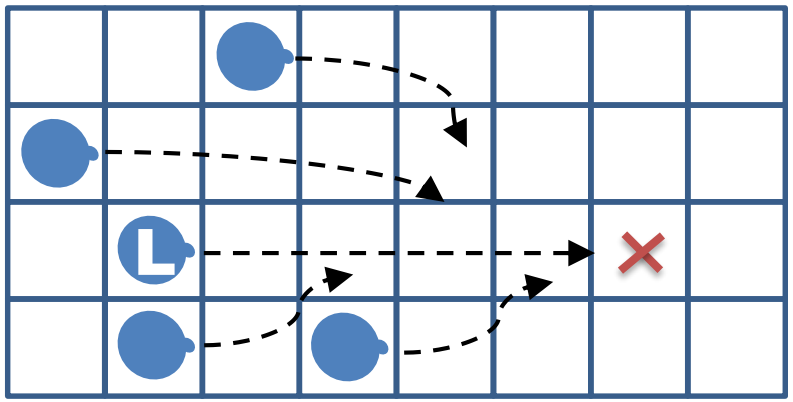
- Can be difficult to set up rules to get just the right shape
  - E.g. characters often end up jostling for position in the center of the V
  - E.g. poor target selection can lead to long diagonal line rather than V
- Debugging is challenging
- Overall effect of controlled disorder

Flavors of “Follow the Leader”

- Once formation is formed, path plan to goal only for “leader”
- All other units move
  - Simple/naïve: all move toward leader
  - More advanced: move to offset position from leader
  - Fancy: rotate/wheel positions relative to leader dir
  - Robust: movement around “anchor point”
- Usually 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

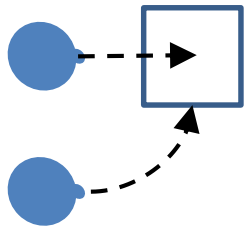
(Fixed Formations) Naïve Follow the Leader

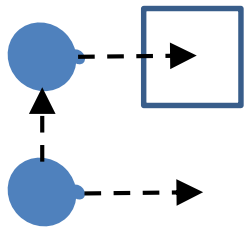
- Select number of units to move to target



- Independent shortest distance navigation: traffic jam
- Traffic jam impacts:
  - Some might stop
  - Some may path plan around teammates
    - Race condition: some teammates will have already moved away, resulting in bot navigating around empty space

Traffic Jams: Replanning vs Waiting

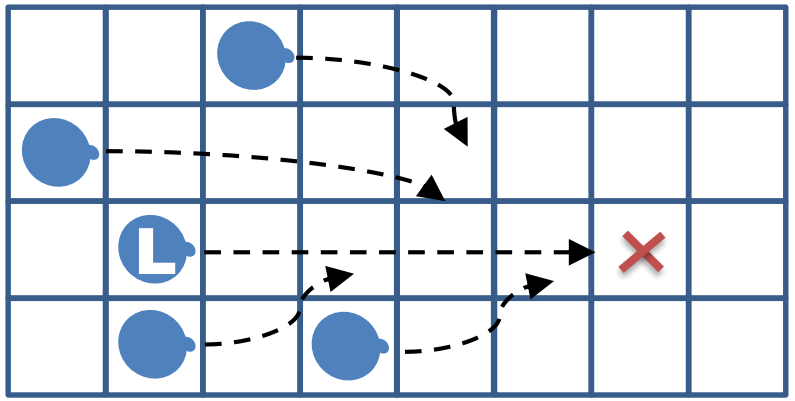


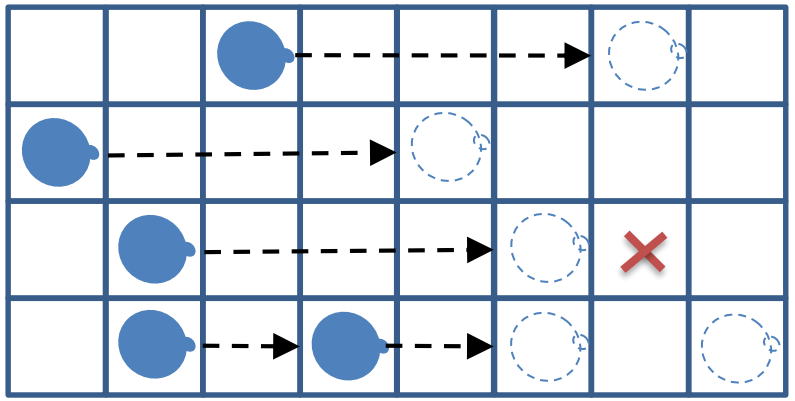


- One NPC “locks” the cell
  - Other agent perceives locked cell as obstacle
- Replanning may cause other agent to move in a different direction later to return to the original cell when it becomes unlocked
- Recognize this, and simply wait.

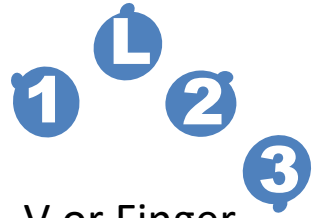
Avoiding Traffic Jams

- Option 1: Position determined directly from formation geometry (often set directly)
- Perform translation / rotation based on relative offsets





Fixed offsets: Rotation & Translation



V or Finger Four

Given relative slot position $\overrightarrow{p_r}$ and relative orientation θ_r calculate final position $\overrightarrow{p_s}$ and orientation θ_s of character in slot s, relative to leader's position $\overrightarrow{p_L}$ and orientation θ_L ($\overrightarrow{\Omega_L}$ in rotation matrix form)

- $\overrightarrow{p_s} = \overrightarrow{p_L} + \overrightarrow{\Omega_L}\overrightarrow{p_r}$
  - $\overrightarrow{p_s} = \overrightarrow{p_L} + [p_{rx}\,cos\theta\ - p_{ry}sin\theta,\ p_{rx}sin\theta + p_{ry}cos\theta]$
- $\theta_s = (\theta_L + \theta_r)\ mod\ 2\pi$

Where rotation $\overrightarrow{\Omega_L} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$

Slot 1:

Relative position $\overrightarrow{p_1} = [-2, -2]$

Relative orientation $\theta_1 = 0.785\text{r}$ // pi/4 or 45deg

Leader position $\overrightarrow{p_L} = [22, 20]$

Leader orientation $\theta_L = 1.5\text{r}$ //85.94deg

$$\overrightarrow{\Omega_L} = \begin{bmatrix} 0.071 & -0.997 \\ 0.997 & 0.071 \end{bmatrix}$$

Final position $\overrightarrow{p_s}$ for slot 1

$$\overrightarrow{p_s} = [22, 20] + [1.85, -2.14]$$

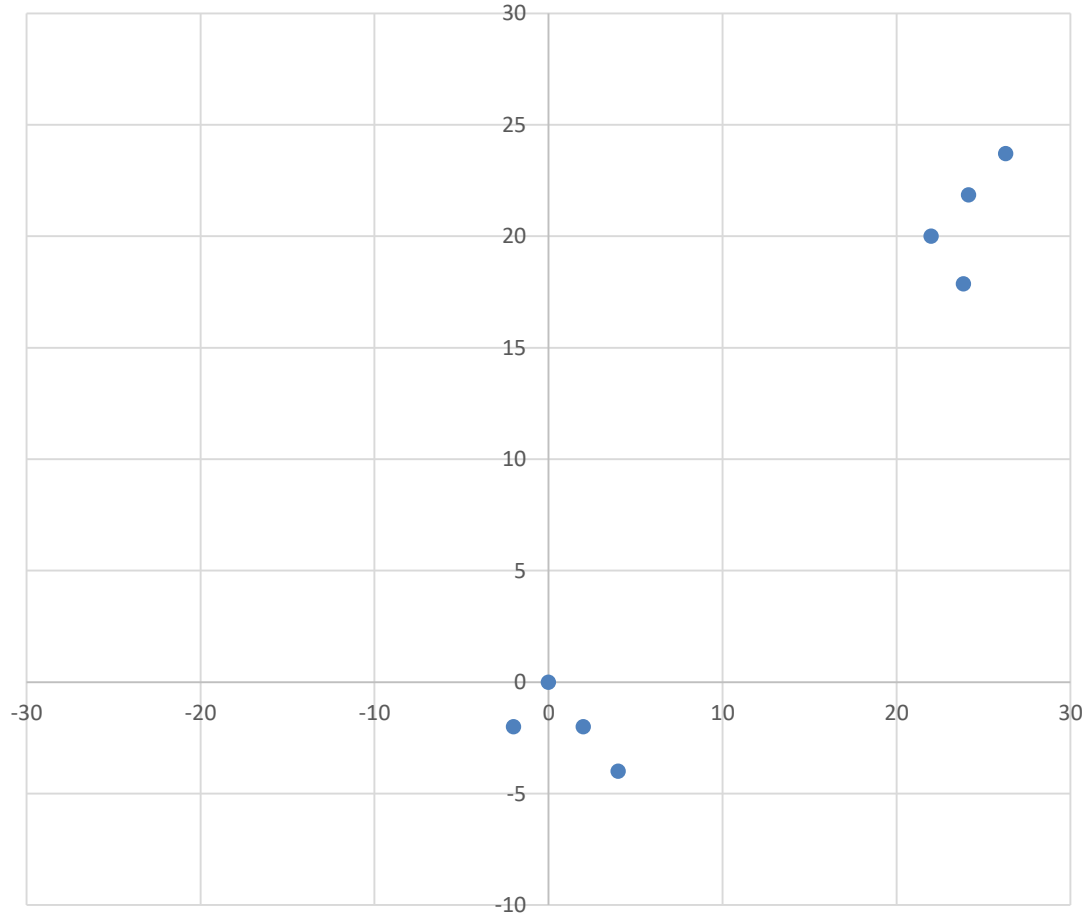
$$\overrightarrow{p_s} = [23.85, 17.86]$$

Final orientation θ_s

$$\theta_s = 1.5r + 0.785r = 2.285r \text{ //130.9deg}$$

https://en.wikipedia.org/wiki/Rotation_(mathematics)

	x	y	rad	deg	sin	cos
	0	0				
leader pos	22	20				
leader orient			1.5	85.94367	0.997495	0.070737
rel pos 1	-2	-2				
rel orient 1			0.785398	45		
rel pos 2	2	-2				
rel orient 2			5.497787	315		
rel pos 3	4	-4				
rel orient 3			4.712389	270		
final pos 1	23.85352	17.86354				
final orient 1			2.285398	130.9437		
final pos 2	24.13646	21.85352				
final orient 2			0.714602	40.94367		
final pos 3	26.27293	23.70703				
final orient 3			6.212389	355.9437		



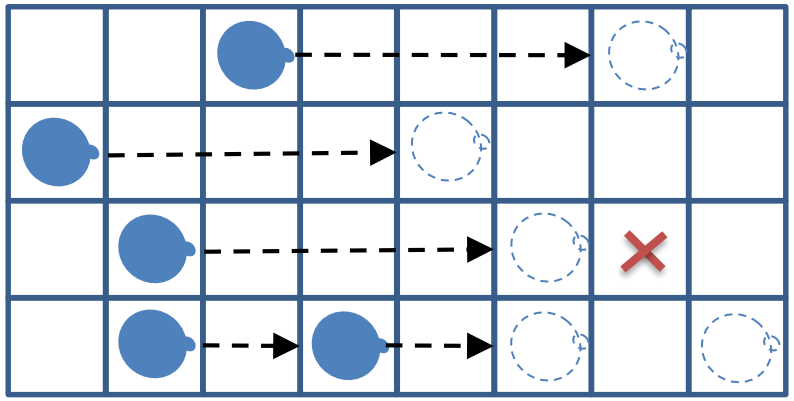
Directly Set Gotchas

- What problems can happen when position and orientation are set directly?

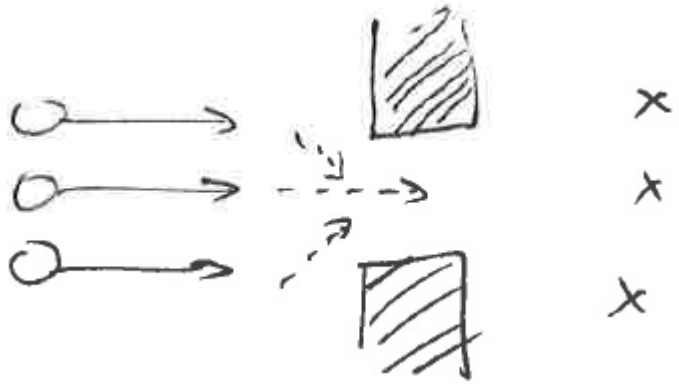
- Leader's move no different to a non-formation character
  - Warp speed for outlying characters: limit turn speed to avoid outlying characters sweeping round at implausible speeds
  - Impediments: collision / obstacle avoidance behaviors should take into account the size of the whole formation

Challenges for direct set fixed offsets

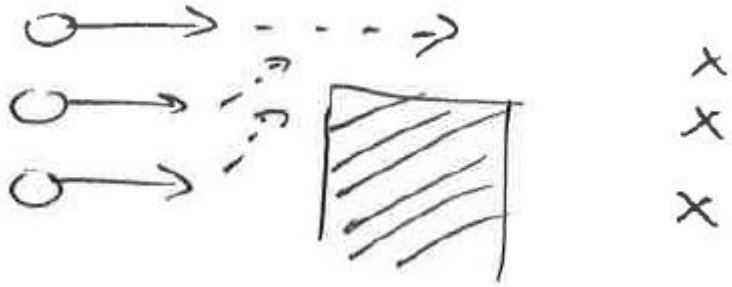
- Offsets



- Choke points

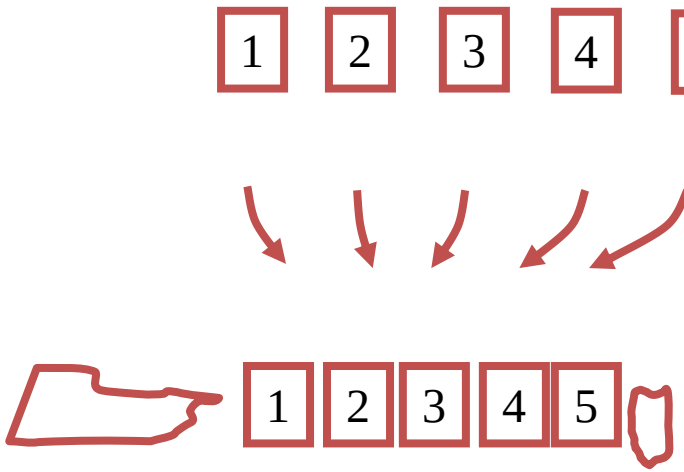


- Obstructions

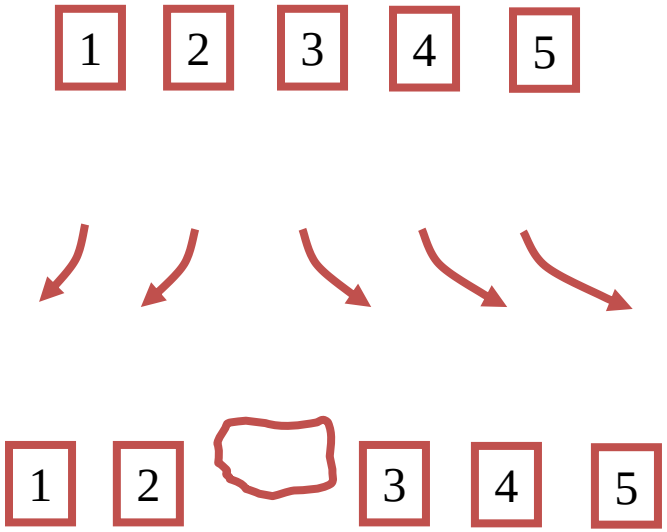


- Occluded destinations

Formations: Obstacles



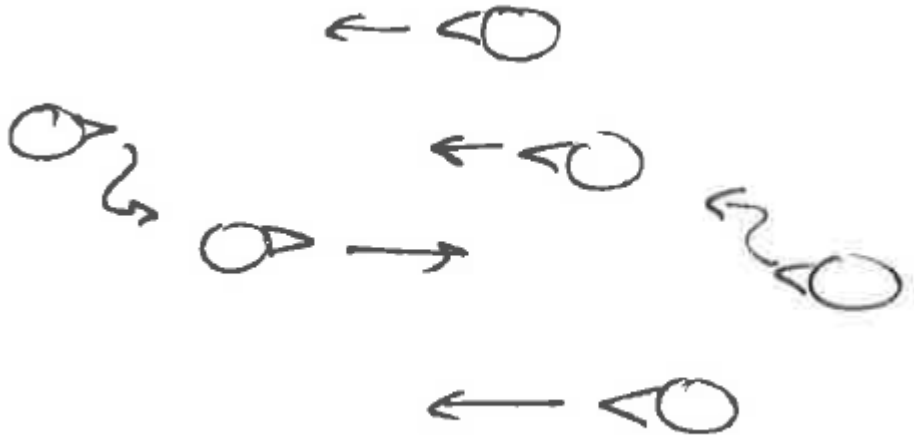
Scale formation layout to fit through gaps

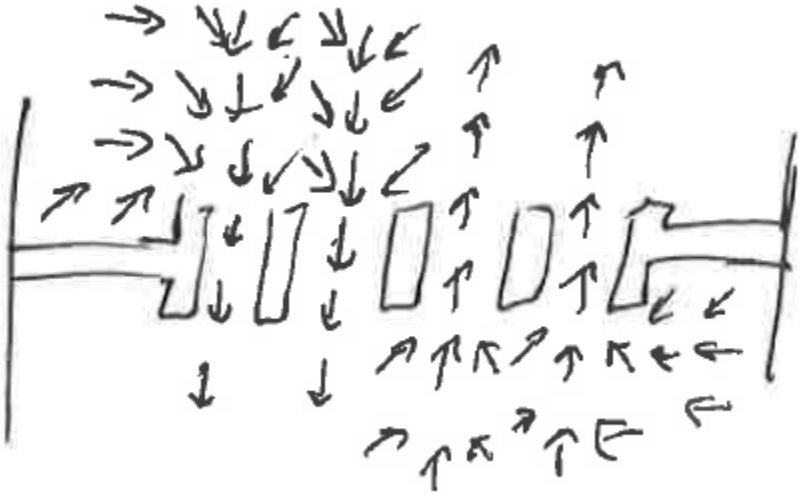


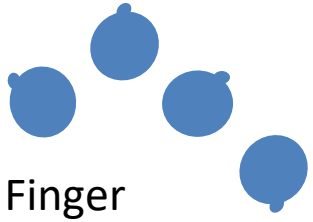
Subdivide formation around small obstacles

Recall other approaches

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



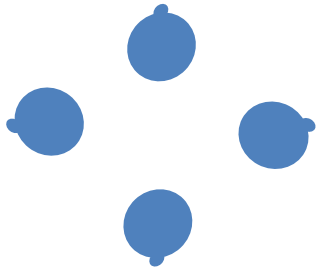


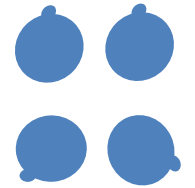


V or Finger Four

Line

Fixed formations

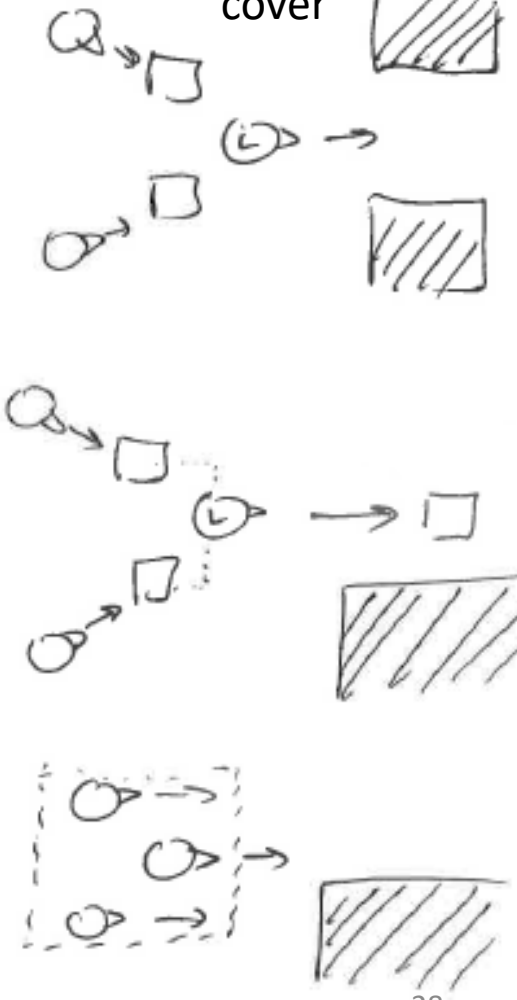




- **Follow the leader**: Path plan for leader (naive)
  - All others move toward leader
- **Leader w/ Offsets**: Path plan for leader (alt). “Two-level”
  - All team members path plan/steer (arrive) to an offset
  - Often position and orientation set directly from formation geometry
  - Only leader does long-range path planning
  - Flow around obstacles and through choke points
  - Make use of waiting when region becomes occupied
  - Lock region about to travel to
- **Virtual Unit Entity**: Replace team with a virtual bot
  - All members controlled by a joint animation
- Pros & Cons of each?

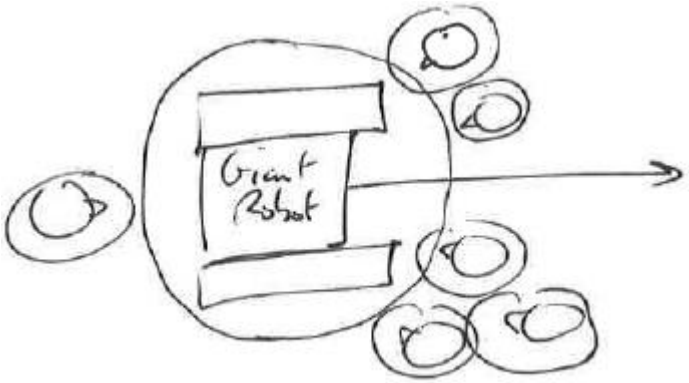
Defensive circle

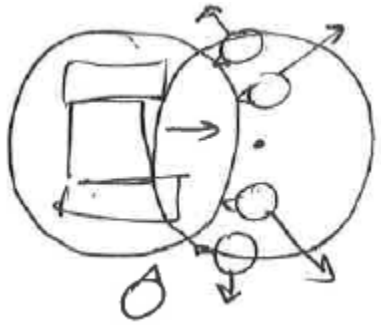
Two abreast in cover



Entities of different sizes: Giant Robots

- Problems:
  - Large units get swarmed/surrounded by smaller entities & cannot move – paralyzed by collision boxes. No space big enough
  - Can try to go around too , but can get blocked in
  - Large units: from powerful to liability





- Solution… Physics?
  - Pushes small units out of way, but they can pile on more.
- Ignore smaller units
  - Plan for big unit without small regard
  - Broadcast where it is going / send messages for smaller entities to move out of way
  - Smaller units navigate to nearest place outside large unit path
  - Appears as if small units are “afraid” of getting squished

Leadership problems

- What if leader needs to avoid an obstacle?
  - leader's actions are mimicked by the other characters, although they are largely free to cope with obstacles in their own way
  - E.g. all the slots in the formation lurch sideways and every other character will lurch sideways to stay with the slot.
- How do we ensure everyone "keeps up"

- Introduce "invisible" leader called the "**anchor point**"
  - separate steering system controlling the whole formation, but no individuals
  - invisible leader has location in the game, used to lay out the formation pattern and determine the slot locations. But leader is not a char
  - ignores small obstacles & bumping into other characters
  - simplifies implementation: no roll tracking and new leader election
  - Anchor point = center of mass of slots = average position and orientation

(Fixed formations) Leader w/ offsets: “keeping up”

- Slow the formation down options:
  - Half of character speed
  - Let slowest character determine formation speed
  - Moderate movement based on distance from slot target
- Moderate movement of formation based on current positions of characters in slot: kinematics of **anchor point**
  - Base position, orientation, velocity of anchor points on the average of characters in the slot
    - Choosing exactly the average means that characters are almost in position, so that they move slowly towards their slot position.
    - Anchor point moves even slower
    - etc.
  - Move anchor point ahead of the average for moving formations
  - Set anchor point to average for stationary formations
  - Ensure that anchor point orientation is average orientation of slots, or formation will spin around

Using the Anchor

- Center of mass (COM)

$$p_c = \frac{1}{n} \sum_{i=1..n} \begin{cases} p_{s_i} & \text{if slot } i \text{ is occupied,} \\ 0 & \text{otherwise,} \end{cases}$$

- Anchor update

$$p_{\text{anchor}} = p_c + k_{\text{offset}} v_c,$$

  - k_{offset} is small offset ahead of COM
  - Vc is velocity of COM

- Slot position update

$$p'_{s_i} = p_{s_i} - p_c.$$

- Average orientation vector

$$\vec{v}_c = \frac{1}{n} \sum_{i=1..n} \begin{cases} \vec{\omega}_{s_i} & \text{if slot } i \text{ is occupied,} \\ 0 & \text{otherwise,} \end{cases}$$

- Unit vector form

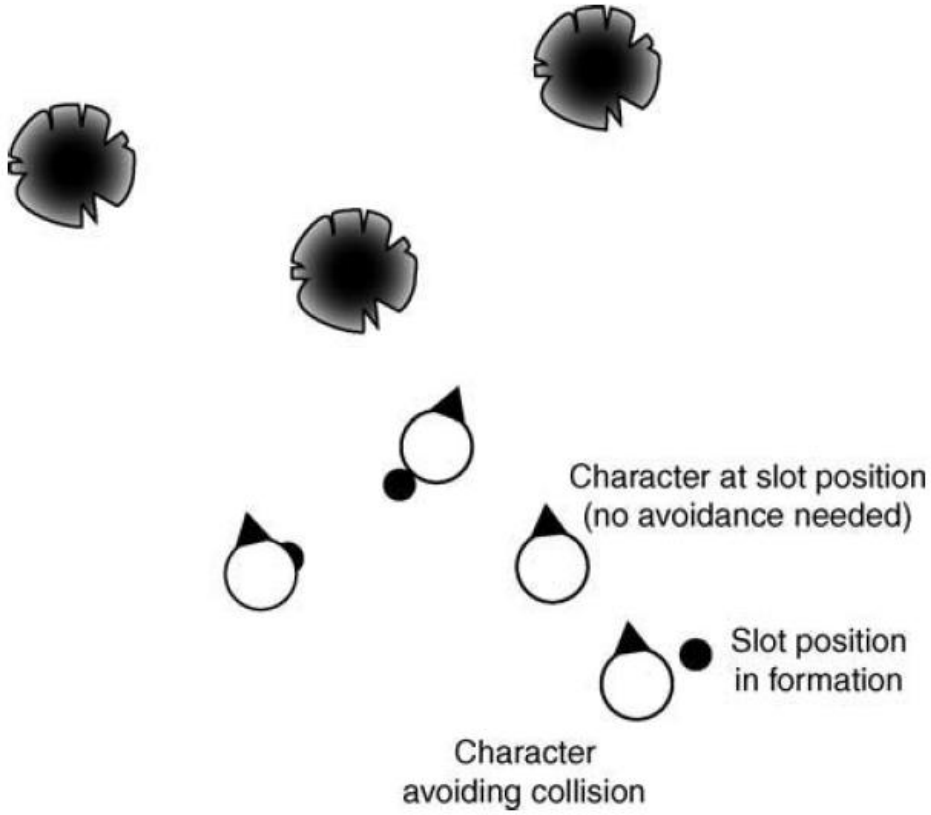
$$\vec{\omega}_c = \frac{\vec{v}_c}{|\vec{v}_c|},$$

- Slot orientation update

$$\omega'_{s_i} = \omega_{s_i} - \omega_c.$$

Types of formations: **Multi-Tier w/ anchor**

- Combines geometric formations with the flexibility of an emergent approach
- Defined/fixed pattern of slots
- Use the slot at a target location for an arrive behavior, with collision avoidance etc
- Two-level steering, in sequence:
  - first the “leader” (anchor) steers the formation pattern
  - then each character in the formation steers to stay in the pattern
- A slot may be briefly impossible to achieve, but chars steering algorithm ensures sensible behavior



M&F 3.57

Hierarchical Movement System

- Group structure
  - Manages its own priorities
  - Resolves its own collisions
  - Elects a *commander* that traces paths, etc.
- 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

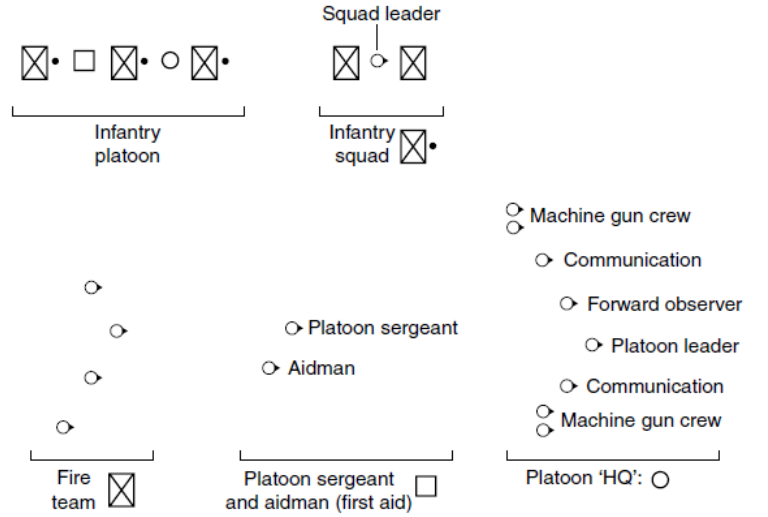
More than two levels & Dynamic slots

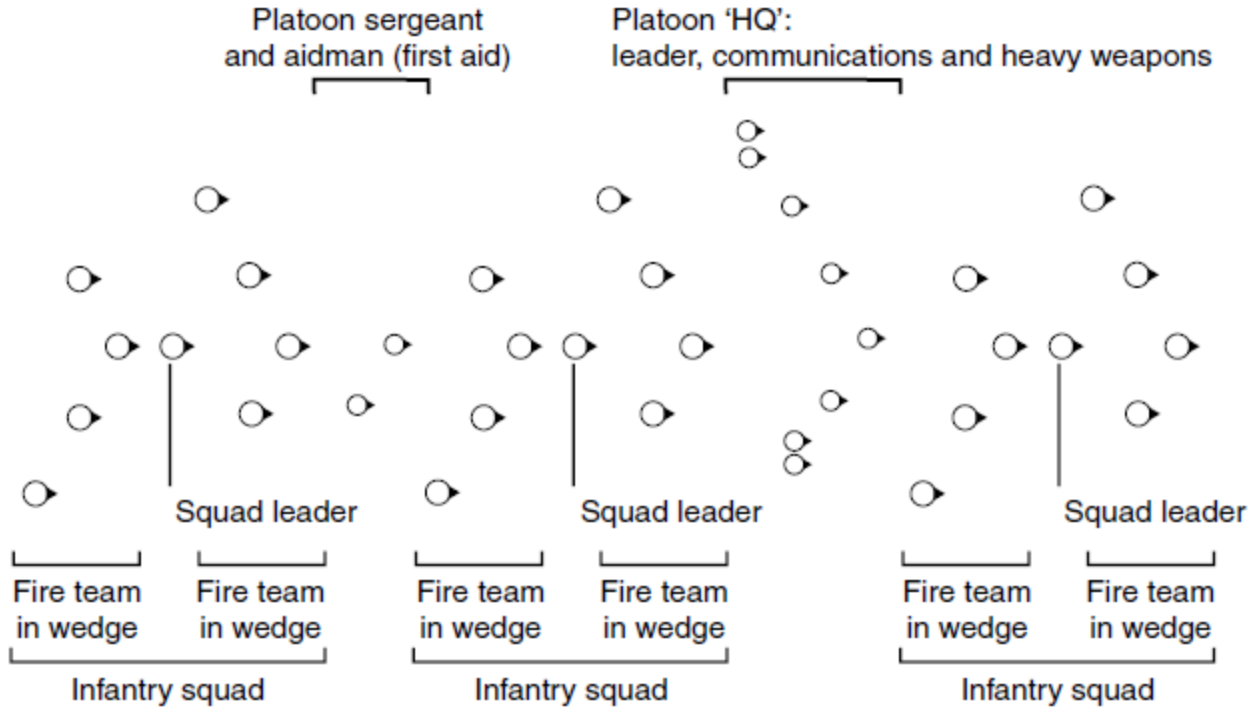
- Needed in military games with lots of units
  - Fire Team (2) → Squad (10) → Platoon (3 to 4 squads) → Company…
- Slot positions now distinguish between roles of characters
  - Squad leader, heavy weapons team, etc
- Also useful for “playbooks”
  - E.g. fielders in a baseball double play, corner kick strategies, etc
- Tactical movement: squads collaborating with others (one moves, other covers)

https://www.youtube.com/watch?reload=9&v=OKRues4Fwrk

Formations[2]

- Each formation has its own steering anchor point
  - This is managed by another formation
  - Anchor point is trying to stay in a slot position of a higher level formation
  - As long as both characters and formations expose the same interface, the formation system can cope with putting either an individual or a whole sub-formation into a single slot





On Slot Assignment

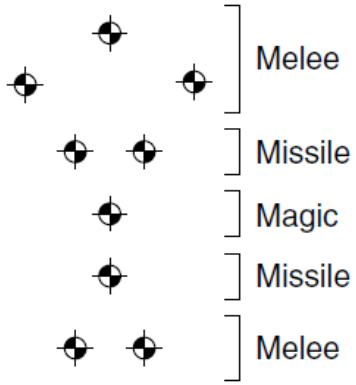
- What if slots were designed for different character roles?
- number of possible assignments of *k* characters to *n* slots

$$ {}_nP_k \equiv \frac{n!}{(n-k)!}. $$

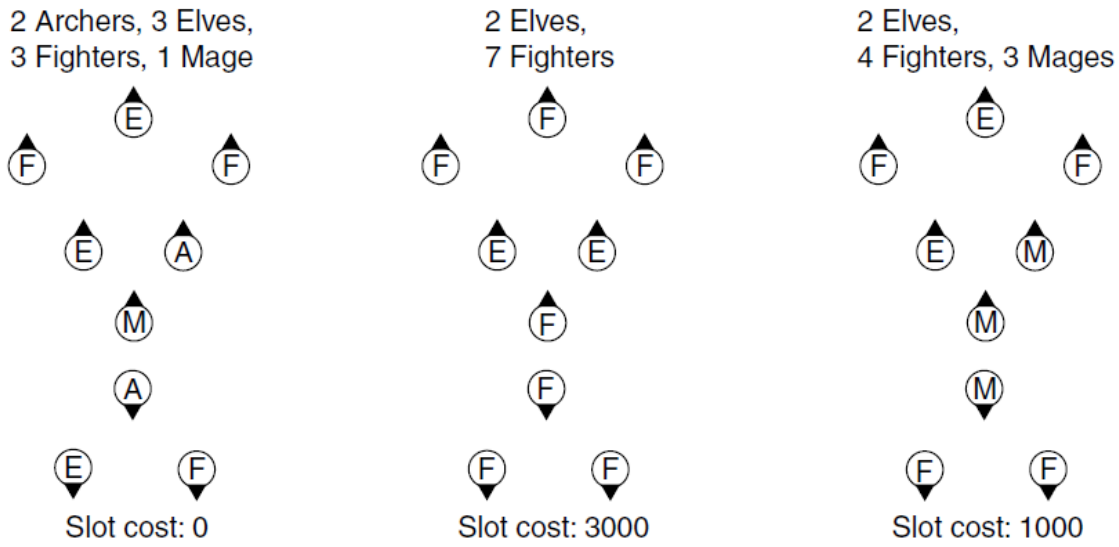
  - 20 slots, 20 chars = 2500 trillion
- Assignment problem is NP-complete

- Options
  - Many different formations for different party compositions
  - Write code to generate formation for characters
  - Role “cost” values (ideal=0, impossible = inf)
    - assign characters to slots in such a way that the total cost is minimized: look at each character in turn and assign it to a slot with the lowest slot cost
    - If there are no ideal slots left for a character, then it can still be placed in a non-suitable slot.

	Magic	Missile	Melee
Archer	1000	0	1500
Elf	1000	0	0
Fighter	2000	1000	0
Mage	0	500	2000



M&F 3.6



M&F 3.61

No chair left when music stops

- Consider highly constrained characters first and flexible characters last
  - Calculate **ease of assignment** value which reflects how difficult it is to find slots for them

$$\sum_{i=1..n} \begin{cases} \frac{1}{1+c_i} & \text{if } c_i < k, \\ 0 & \text{otherwise,} \end{cases}$$

  - C_i is cost of occupying slot i, k is "too difficult" threshold
- Characters that can only occupy a few slots will have lots of high slot costs and therefore a low ease rating

AIIDE 2015 Videos

- Mesh: https://youtu.be/ZIAmoRsu3Z0?list=PLxGbBc3OuMgg7OuyLfvXQLR6HKcoglCfG&t=1014
- Formations: https://youtu.be/ZIAmoRsu3Z0?list=PLxGbBc3OuMgg7OuyLfvXQLR6HKcoglCfG&t=1713
- Nav Mesh Weighting: https://youtu.be/ZIAmoRsu3Z0?list=PLxGbBc3OuMgg7OuyLfvXQLR6HKcoglCfG&t=1833
- Formations + Nav mesh Weighting: https://youtu.be/ZIAmoRsu3Z0?list=PLxGbBc3OuMgg7OuyLfvXQLR6HKcoglCfG&t=2266
- Designed path network points: https://youtu.be/ZIAmoRsu3Z0?list=PLxGbBc3OuMgg7OuyLfvXQLR6HKcoglCfG&t=2407

Next Class

- Finite state machines
- Read: Buckland, CH 2 (M Ch 5.1-4)