Flocking

- C. W. Reynolds: "Flocks, herds, and schools: A distributed behavioral model" (1987)
- a flock seems to react as autonomous entity although it is a collection of individual beings
- flocking algorithm emulates this phenomenon
- results resemble various natural group movements
- boid = an autonomous agent in a flock

Rules of flocking

1. **Separation**: Do not crowd flockmates.
2. **Alignment**: Move in the same direction as flockmates.
3. **Cohesion**: Stay close to flockmates.
4. **Avoidance**: Avoid obstacles and enemies.

→ boid's behavioural urges

Observations

- stateless algorithm
  - no information needs to be maintained
  - boid re-evaluates the environment on each update cycle
- no centralized control
  - emergent behaviour

Other uses for flocking

- swarm algorithms
  - solution candidate = boid
  - solution space = flying space
  - separation prevents crowding the local optima
- obstacle avoidance in path finding
  - steer away from obstacles along the path

Influence maps

- discrete representation of the synthetic player's knowledge of the world
- strategic and tactical information
  - frontiers, control points, weaknesses
- influence
  - type
  - repulsiveness/alluringness
- recall path finding and terrain generation

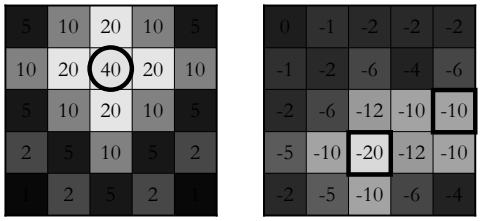
Assumptions

- a regular grid over the game world
- each tile holds numeric information of the corresponding area
  - positive values: alluringness
  - negative values: repulsiveness

Construction

1. initialization
   - assign values to the tiles where the influence exists
2. propagation
   - spread the effect to the neighbouring tiles
   - linear or exponential fall-off
   - cut-off point

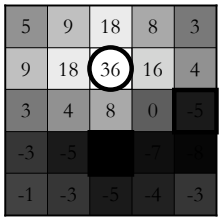
Example: Initialization and propagation



Aggregation

- influence map can be combined
  - the same (or compatible) granularity
- example
  - map 1 = my troops
  - map 2 = enemy's troops
  - map 3 = map 1 + map 2 = battlefield
- aggregation
  - operator: sum, product
  - weights: to balance the effects

Example: Aggregation



Evaluation

- static features: compute beforehand
- periodical updates
  - categorize the maps based on the rate of change
  - lazy evaluation

Key questions for synthetic players

- how to achieve real-time response?
- how to distribute the synthetic players in a network?
- how autonomous the synthetic players should be?
- how to communicate with other synthetic players?