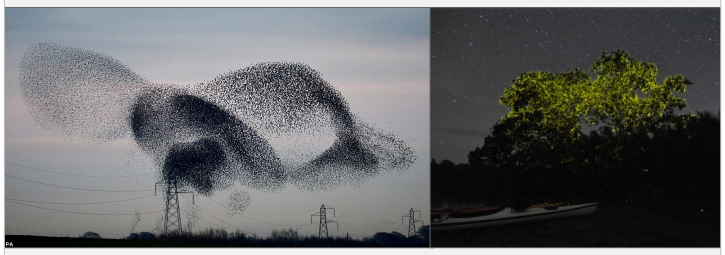
Behavioral Animation

Complex Group Behavior

https://www.youtube.com/watch?v=V4f_1_r80RY



http://www.dailymail.co.uk/news/article-2514252/Incredible-photos-ofhe-moment-TEN-THOUSAND-starlings-fly-formation-Scottish-Borders.html

http://www.firefly.org/synchronous-fireflies.html

Emergent Behavior

Emergent behavior is complex behavior which arises from

- the application of simple rules, and
- the interaction of (only) nearby individuals

→ *global patterns from local behavior*
→ *organization without a leader*

Many natural systems display emergent behavior.





https://en.wikipedia.org/wiki/Emergence
http://www.pbs.org/wgbh/nova/nature/emergence-examples.html

Boids

- due to Craig Reynolds
  - published at SIGGRAPH 1987

- a major advance in computer animation in movies
  - Reynolds won a Scientific & Engineering Academy Award in 1998

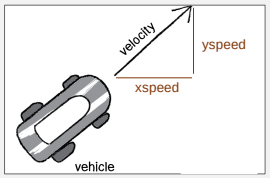
demonstrated in *Stanley and Stella in: Breaking the Ice*, 1987

http://www.youtube.com/watch?v=3bTqWsVqyzE

first feature film use in *Batman Returns*, 1992

A Simple Vehicle Model (Boid)

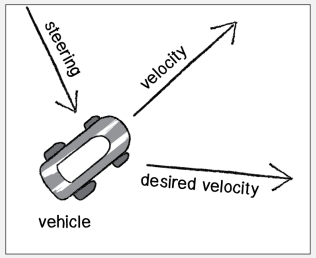
- a basic boid has
  - position
  - velocity (speed and direction)
- additional properties
  - maximum acceleration
  - maximum speed



http://natureofcode.com/book/chapter-6-autonomous-agents/

Steering Vectors

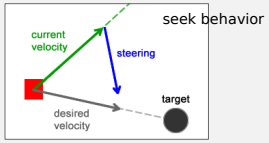
- a **steering vector** indicates how the boid wants to turn
  - direction and effort
- the steering vector is used to update the boid's velocity



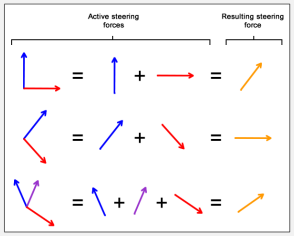
http://natureofcode.com/book/chapter-6-autonomous-agents/

Steering Behaviors

- a **steering behavior** is a method for computing a steering vector



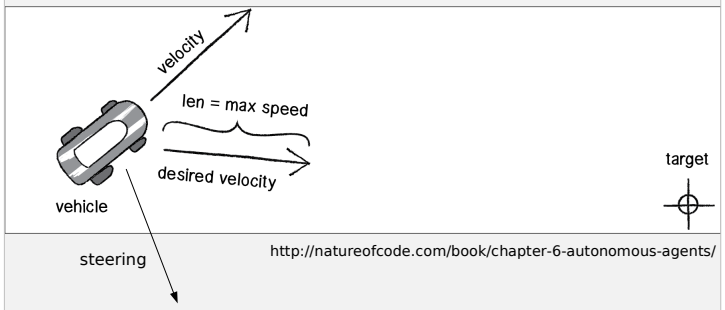
- steering vectors for multiple behaviors can be combined into the **net steering vector** (or **force**)
  - net steering force is used to update the boid's velocity



https://gamedevelopment.tutsplus.com/tutorials/understanding-steering-behaviors-flee-and-arrival--gamedev-1303
https://gamedevelopment.tutsplus.com/tutorials/understanding-steering-behaviors-movement-manager--gamedev-4278

Seek

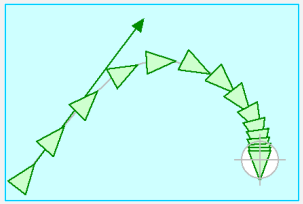
- **seek** steers the boid to head towards the target at its maximum speed

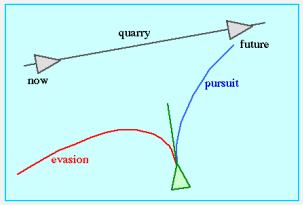


http://natureofcode.com/book/chapter-6-autonomous-agents/

Arrive and Pursue

- **arrive** is similar to seek, but the boid slows once it is within the *stopping radius* near the target
- **pursue** is similar to seek, but uses an estimate of the interception point instead target's current position



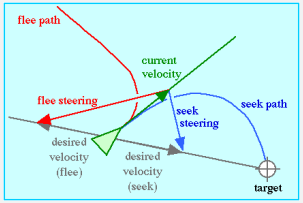


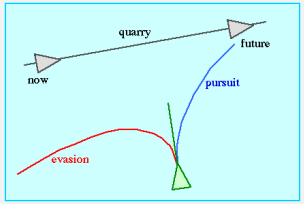
http://www.red3d.com/cwr/steer/gdc99

Flee and Evade

- **flee** is the opposite of seek
- **evade** is the opposite of pursue

In both cases, the goal is to steer so as to be heading away from the target as fast as possible.





http://www.red3d.com/cwr/steer/gdc99

Variations on the Theme

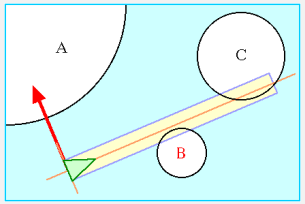
- offset pursuit
  - seek target is some distance *d* to the side of the predicted future position of the target
- interpose
  - seek a target point between two other boids
- hide
  - seek a target on the opposite side of an obstacle from another boid

Forward and Wander

- **forward** steers in the same direction but accelerates as needed to reach maximum speed
- **wander** steers randomly, but not *too* randomly
  - implemented by seeking a point ahead of the boid and not too far to the side

Obstacle Avoidance

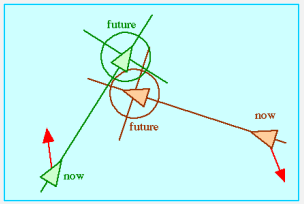
Avoid hitting obstacles.



- identify the most threatening obstacle
  - project region ahead of the boid
    - length depends on boid's speed and agility (longer for faster / less agile)
  - of the obstacles intersecting the region, find the obstacle closest to the boid
- steer away from it

Unaligned Collision Avoidance

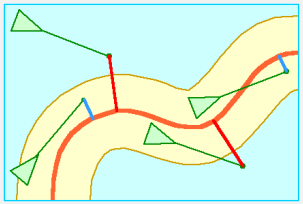
Avoid collisions between boids.



- identify potential collisions
  - determine the point of nearest approach for each other boid
    - assume boids continue with their current velocities
- steer to avoid the location of the nearest such potential collision

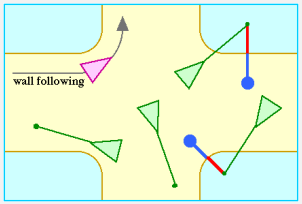
Path Following

Follow a particular corridor.



- estimate boid's future position *p* according to its current velocity
- find the point *q* on the path's spine nearest the estimated future position
- if the distance between *p* and *q* is greater than the path's radius, seek on *q*
  - otherwise no steering is needed

Wall Following and Containment

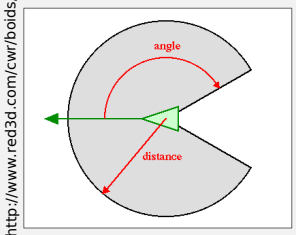


- **wall following**
  - predict future position of boid based on current velocity
  - find closest point on the wall
  - seek target is offset from that closest point
- **containment** – stay within an area
  - if predicted future position of boid is outside the containment region, seek on an inside point

Neighborhoods

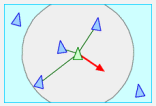
- in groups, a boid only reacts to its neighbors
  - behaviors are local

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



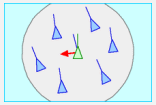
neighborhood is defined by a distance (radius) and an angle

Group Behaviors

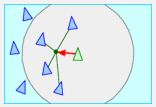


separation: steer away from your neighbors so you don't get too close

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



alignment: steer to match velocity of neighbors

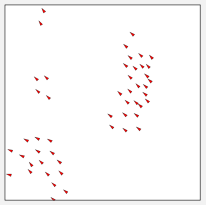


cohesion: steer to move towards the center of your neighbors

Combined Behaviors

Can have multiple behaviors active at once –

- **flocking** combines separation, alignment, and cohesion
  - also forward or wander or another movement element

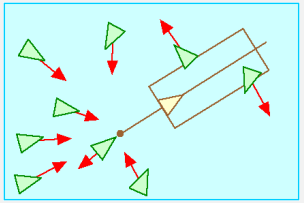


Can choose between different behaviors at different times –

- **shadow**
  - if close to target, use alignment to match velocity at a distance
  - otherwise arrive at target

Combined Behaviors

leader following



- followers want to stay near the leader without crowding the leader or each other or getting in the leader's way
  - if follower is in a region in front of the leader, it steers away from the leader's path
  - otherwise, followers arrive at a point just behind the leader
  - separation is used to keep followers out of each other's way

 http://www.red3d.com/cwr/steer/gdc99

Behavioral Animation

- combines steering and other actions with a higher-level "brain" to determine what action(s) to do in what situations

The Lion King (1994)

https://www.youtube.com/watch?v=NofrY8eB3u4

The Lord of the Rings (2001-2003)

https://www.youtube.com/watch?v=rCZ3SN65kIs

Shrek 2 (2004)

Artificial Fish

- Tu and Terzopoulos, 1994

"Imagine a virtual marine world inhabited by a variety of realistic fishes..."

https://www.youtube.com/watch?v=VpZ93n5QQuQ

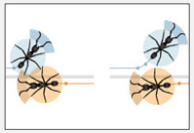
https://www.youtube.com/watch?v=aP1_XkCdAaE

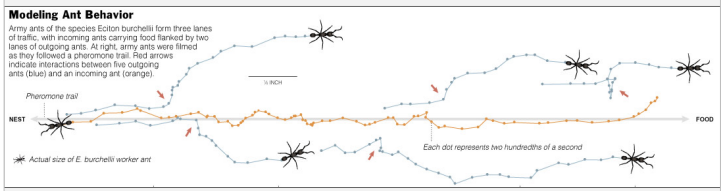
https://www.youtube.com/watch?v=jxR1YwHM5CQ

Tu, Terzopoulos, "Artificial Fishes: Physics, Locomotion, Perception, Behavior", SIGGRAPH '94
http://www.siggraph.org/education/materials/HyperGraph/animation/art_life/fish.htm

Ants

- basic behaviors
  - follow pheromone trail
  - if another ant is met, slow down and turn aside





http://www.nytimes.com/imagepages/2007/11/12/science/2007113_TRAFFIC_GRAPHIC.html

Boids in Processing

- boid's position and velocity are variables of type `PVector`
  - `.x` and `.y` to access the x and y components
- `draw()` contains four steps –
  - draw the scene
  - compute the net steering force –
    - compute the steering force for each behavior
    - combine the forces
    - limit the size of the force
  - update boid's velocity
    - add the net steering force to the velocity
    - limit the max speed
  - update boid's position
    - add the velocity to the position
    - wrap at the edges of the window

```
PVector position;
PVector velocity;

position.x   // x
position.y   // y
velocity.x   // xspeed
velocity.y   // yspeed
```

three patterns for combining forces, depending on whether there is just one behavior, multiple behaviors active at once, or action selection to choose active behavior(s)

```
void draw () {

  // 1 - draw scene
  background(255);
  drawBoid(pos, vel, 255, 0, 0);

  // 2 - compute net steering force
  //  a - compute steering force for each behavior
  PVector wander = computeWander(pos,vel);
  //  b - combine forces (one behavior)
  PVector steer = new PVector(0,0);
  steer.add(wander);
  //  c - limit the size of the force that can be applied
  steer.limit(maxforce);

  // 3 - update boid's velocity
  //  a - add net steering force
  vel.add(steer);
  //  b - limit boid's max speed
  vel.limit(maxspeed);

  // 4 - update boid's position
  //  a - update position by adding velocity
  pos.add(vel);
  //  b - wrap at edges of window
  if ( pos.x > width ) {
    pos.x = 0;
  } else if ( pos.x < 0 ) {
    pos.x = width;
  }
  if ( pos.y > height ) {
    pos.y = 0;
  } else if ( pos.y < 0 ) {
    pos.y = height;
  }
}
```

one behavior


```
  // 2 - compute net steering force
  //  a - compute steering force for each behavior
  PVector wander = computeWander(pos,vel);
  //  b - combine forces (one behavior)
  PVector steer = new PVector(0,0);
  steer.add(wander);
  //  c - limit the size of the force that can be applied
  steer.limit(maxforce);
```

one behavior

```
// 2 - compute net steering force
//  a - compute steering force for each behavior
PVector wander = computeWander(pos,vel);
PVector seek = computeSeek(pos,vel,new PVector(mouseX,mouseY));
//  b - combine forces (weighted sum)
PVector steer = new PVector(0,0);
wander.setMag(1.2);
seek.setMag(1);
steer.add(wander);
steer.add(seek);
//  c - limit the size of the force that can be applied
steer.limit(maxforce);
```

multiple behaviors active at the same time

```
// 2 - compute net steering force
//  a - compute steering force for each behavior
PVector wander = computeWander(pos, vel);
PVector seek = computeSeek(pos, vel, new PVector(mouseX, mouseY));
//  b - combine forces (action selection)
PVector steer = new PVector(0, 0);
if ( pos.x < width/2 ) {  // wander in the left side of the window
  steer.add(wander);
} else {  // seek in the right side of the window
  steer.add(seek);
}
//  c - limit the size of the force that can be applied
steer.limit(maxforce);
```

choosing between behaviors