# Boids

Daniel Walther, Jacqueline Ulken

# What are Boids?

Bird-oid Objects

Flocking Rules

# Boids: Bird-oid Objects

- Artificial life program by Craig Reynolds
- Simulates flocking behavior of birds
- Used in movies (*Batman Returns* (1992)), video games (*Half-Life*), teams of robots, swarm robotics, optimizing tasks (Cui, Zhihua; Shi, Zhongzhi (2009). "Boid particle swarm optimisation")

# Flocking Rules

- Separation
  - Boids keep a minimum distance from each other
  - Steer away from each other if they are too close
- Alignment
  - Boids align themselves to each other
  - Steer towards the average direction other local boids move in
- Cohesion
  - Boids stay together
  - Steer towards the average position of other local boids

# Motivation and Goals

Reproduce Craig Reynolds’ results

Observe emergent behavior

# What did we want to achieve?

- Reproduce Craig Reynolds' results
- Implement a way to adjust the flocking rules and observe how they influence the flocking
- Add additional features, and observe how the flocking behavior changes

Exploring emergent behavior:

- Flocks form, move together
- Flocks split and reform after avoiding obstacles

# Implementation

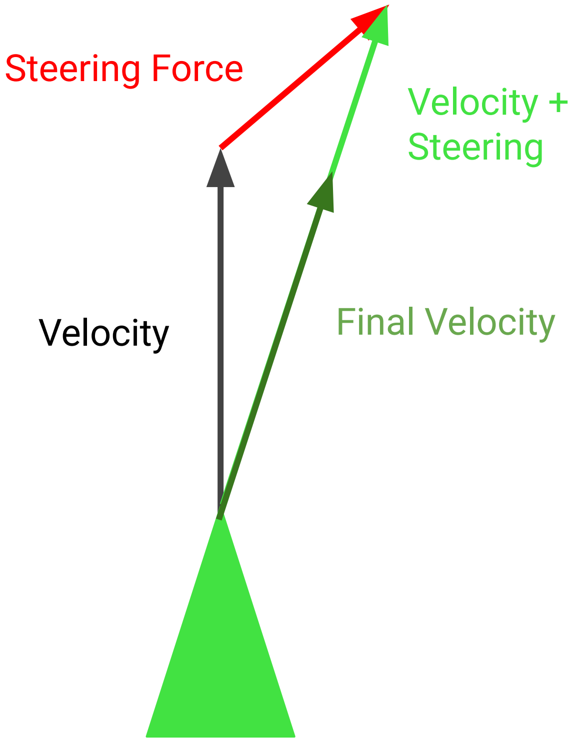
Steering

Pygame

Actor, boid and predator forces

# Steering

- Boids start with random position and velocity
- Outside influences apply “steering forces” to the boid
  - E.g. flocking forces, obstacle avoidance, fleeing from predators
- Sum of all these forces is added to the velocity vector
- Magnitude of the velocity vector is capped to the boid’s maximum speed



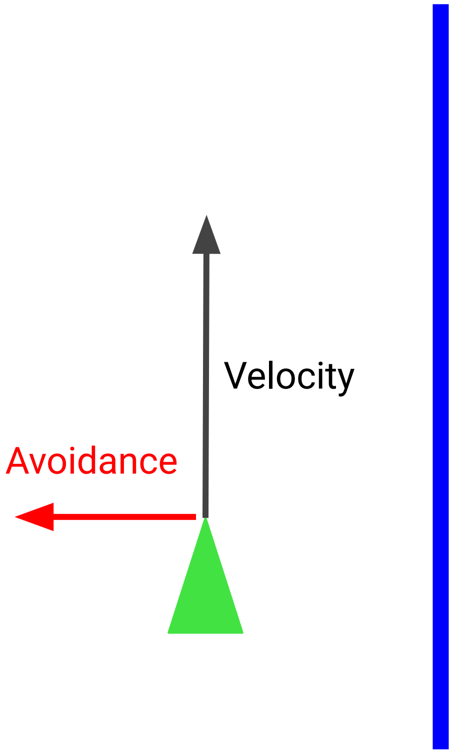
# Pygame: Main Simulation Loop

- Made for Game Development but works great for interactive simulations
- Easy to implement sliders, buttons
- Every frame is one timestep in the simulation. Pygame calculates *dt* for us: the time in milliseconds between frames

1. Pygame events are handled; buttons and sliders
2. Update *dt* and use it in our simulation step:
   - Get boid neighbors
   - Get predators/prey
   - Update forces, velocities and positions
3. Draw visuals on the canvas and update the display

# Actor Forces

- Actors: both predators and boids are “actors”
  - All actors calculate obstacle avoidance. Two types:
    - Circle obstacle (easy to implement)
    - Wall obstacles (more difficult)



# Boid Forces

- Flocking forces
  - First find all neighbors
  - Separation: find neighbors within separation radius
  - Alignment: find average velocity of flock
  - Cohesion: find average position of flock
- Fleeing force
  - Find closest predator
  - Predict predator's future position using its current velocity
  - Steer away from that position

# Predator Forces

- Pursuit Force
  - Find closest boid
  - Calculate future position using it's current velocity
  - Steer towards that position

# Demonstration

# Difficulties

And how we solved them

Visualization

Version control

Jittering of the boids

# Visuals: Live interactive animation

Matplotlib **+**

- Familiar
- Wide use in science
- Focus on plotting data
- (Static and animated plotting)

=> Video demonstration

Pygame **-**

- Unfamiliar
- Less known for simulations
- Not as comprehensive (smaller team, less features)
- (probably sucks for static plotting)

# Visuals: Live interactive animation

Matplotlib **-**

- No advice on *interactive live* animations
- Complicated to make this work
- IDE problems
  - Python 3.x version matters for *live* animations

Pygame **+**

- Made for live user interactions
- By design lightweight & easy to use, even at low levels
- Easy to learn, good docs
- Even matplotlib backends are easily integrated (any visuals)
- …just works (even Win95)

Resolution: switching to pygame & adapt

# Version control

- git vcs (GitHub)
- Visualization challenges made things worse & delayed progress
- Pycharm:
  - Built-in vcs can be confusing
  - Hidden & environment files caused merge conflicts
  - Failed resolution: CodeWithMe parallel coding & git don’t agree
  - Adapting boids’ locations to pygame’s pixel based system

Resolutions:

multiple ‘main’ files, coordinate merging, swap dev roles

# Jittering of the boids (emergent bug)

- Appearance of glitching mid-air, caused by frame-to-frame change of direction

=> video demonstration

Resolution:

Smooth animation (rotation) over last few frames

# Conclusions

What did we accomplish/learn?

Emergent behavior

Further research

# Emergence: intuitive & qualitative analysis

- Do the boids look like actual swarms? ***Yes***
- Did we reproduce emergent group behavior? Does the entity (flock) have properties its parts (boids) do not have on their own? ***Yes*** *(https://en.wikipedia.org/wiki/Emergence)*
  - Moving ‘through’/around an obstacle makes no sense individually
  - Predator avoidance when in a group looks quite realistic

=> These observations imply locally adaptive behavior on the individual level is *sufficient* for the flocking appearance and seemingly coordinated/orchestrated movement at the swarm level.

# Emergence: quantitative analysis

- How can we quantify this ‘flocking appearance’? What is ‘seemingly coordinated movement’?
- How to measure ‘swarminess’? How to identify ‘emergence’ quantitatively?

=> efforts to answer these questions have begun recently (2016 onwards)

# Further research

- Predator avoidance / evasive maneuvers
- Pack hunting
- How do swarms form?
- How do swarms start and stop motion? Parallels to cell biology (cell to cell communication, oscillatory behavior)?
- How do swarming species differ?

# Sources


- https://www.red3d.com/cwr/boids/ (the basic flocking behavior)
- https://www.cs.toronto.edu/~dt/siggraph97-course/cwr87/ (Reynolds' paper from 1987 "Herds, Flocks, and Schools: A Distributed Behavioral Model"
- https://www.red3d.com/cwr/steer/gdc99/ (Reynolds' paper from 1999 "Steering Behaviors For Autonomous Characters")
- https://www.red3d.com/cwr/ (Reynolds' homepage with portrait)
- https://gamedevelopment.tutsplus.com/series/understanding-steering-behaviors--gamedev-12732 (more recent take on the algorithms with programming examples, written by Fernando Bevilacqua (Homepage))
- https://matplotlib.org/
- https://www.pygame.org/docs/
- https://pygamewidgets.readthedocs.io/en/latest/ (user interactions like sliders and buttons)

# Sources

- https://en.wikipedia.org/wiki/Emergence
  "In philosophy, systems theory, science, and art, **emergence** occurs when an entity is observed to have properties its parts do not have on their own, properties or behaviors which emerge only when the parts interact in a wider whole." [source]:
  - https://plato.stanford.edu/archives/spr2012/entries/properties-emergent/
- https://link.springer.com/chapter/10.1007/978-3-319-41000-5_12
  (quantifying swarming behaviour, 2016, abstract tackles my questions spot-on)
- Other robotics papers start to emerge (2018 onwards)