CS 3642 Programming Assignment 1

Boids with Predator (Shark) in Go + WebAssembly

Course: CS 3642 — Artificial Intelligence
Student: James Widner
ID: 001121770
Date: September 9, 2025

Section 1: Agent Design

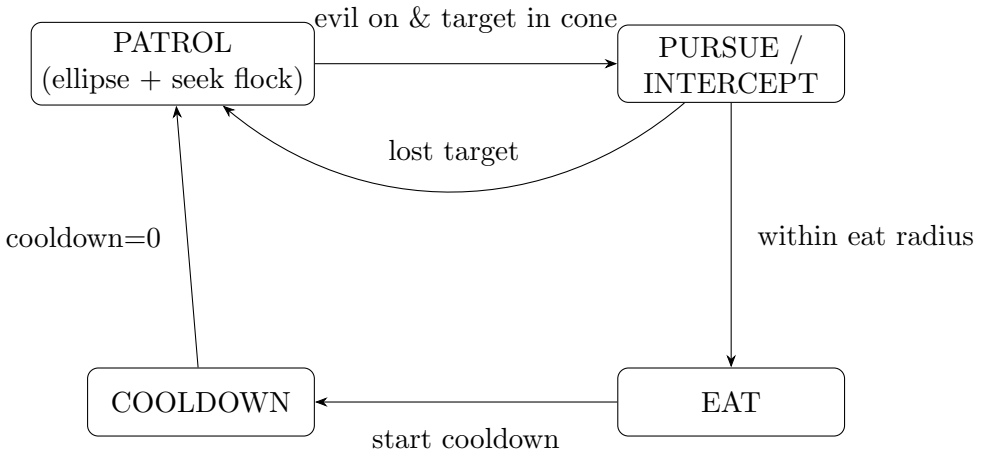
Overview: Why This Is Model-Based

My implementation has two kinds of agents:

- **Boids (fish):** each fish stores position, velocity, tail history, and alive/dead state. They move according to the three classic boids rules: separation, alignment, and cohesion, plus additional models for wall avoidance, a gentle pull to the center, and predator avoidance.
- **Shark (predator):** stores its own position, velocity, patrol phase, smoothed patrol center, and an eating cooldown timer. In patrol mode, it follows an elliptical orbit around the flock using PD control. In evil mode, it performs lead pursuit and can eat boids inside a radius, incrementing a score.

Because both agents use internal state (e.g., flock centroid, wall lookahead, patrol phase, cooldown timer) to choose actions, this is a **model-based agent system**.

Finite-State Diagram (Predator)



Internal Models (Summary)

The three core rules come from Reynolds' boids model (1987), popularized in tutorials such as Sebastian Lague's *Boids* project. A related fish-schooling formulation appears in Inada (2001). I did not derive these equations; I implemented and tuned them for this simulation. In words:

- **Separation:** steer away from nearby boids to avoid collisions.
- **Alignment:** steer to match the average velocity of neighbors.
- **Cohesion:** steer toward the average position of neighbors.

Additional models I added:

- **Wall avoidance:** predictive steering away from world boundaries using a blended wall normal.
- **Home field:** gentle pull back toward the center of the canvas.
- **Predator avoidance:** flee when the shark enters a threat radius.
- **Predator behavior:** elliptical PD patrol, flock-seeking bias, evil-mode interception, eating cooldown, and score tracking.

PEAS for Predator

Performance	Smooth patrol paths, staying near the flock, successful captures in evil mode, natural-looking motion (turn/force bounded), and no wall penetration.
Environment	2D continuous bounded world; 48 boids using Reynolds-style rules (separation, alignment, cohesion); stochastic jitter; predator present.
Actuators	Velocity steering (bounded by max turn and max force); mode toggle (patrol vs evil); cooldown timer for eating.
Sensors	Own position/velocity; flock centroid; nearby boid positions/velocities (with FOV and radius limits); wall proximity (via predictive normal); slider/checkbox inputs from the GUI.

Section 2: Tasks

Task 1 — Cohesive flocking in a bounded 2D world. Boids maintain a stable school using separation, alignment, and cohesion plus predictive wall avoidance and a center pull. *Measure:* average speed stability, cluster compactness, low collisions.

Task 2 — Predator patrol (benign). The shark follows an ellipse around the flock centroid using PD control with a small group-seeking bias. *Measure:* smoothness of path, proximity to flock, natural motion.

Task 3 — Interception and eating (evil mode). In evil mode, the shark selects a forward target, performs lead pursuit with a sweep bias, and eats if within radius, with a cooldown. *Measure:* number of captures (score), time-to-capture, believability of pursuit.

Task 4 — Human-in-the-loop control. Sliders adjust rule weights, separation radius, shark speed, patrol scale, bias, group-seek, and tail length. Checkbox toggles evil mode. *Measure:* UI responsiveness, smooth live updates at ∼60 FPS.

Section 3: Codes and Outputs

Implementation Notes

- Implemented in Go, compiled to WebAssembly.
- Core types: `Vec2`, `Boid`, `Shark`.
- Main loop: read sliders → update flock centroid → update shark (patrol/pursuit) → boid updates → render.
- Rendering: sprites for boids, tails, shark. Score updated live via JS.

Repository and Demo

Source code: https://github.com/pi128/CS3642
Live WASM demo: https://pi128.github.io/CS3642/
Original plan: port to my *FreeWiLi* device; code is organized to enable that deployment. Unable to flash device for WASM files :(

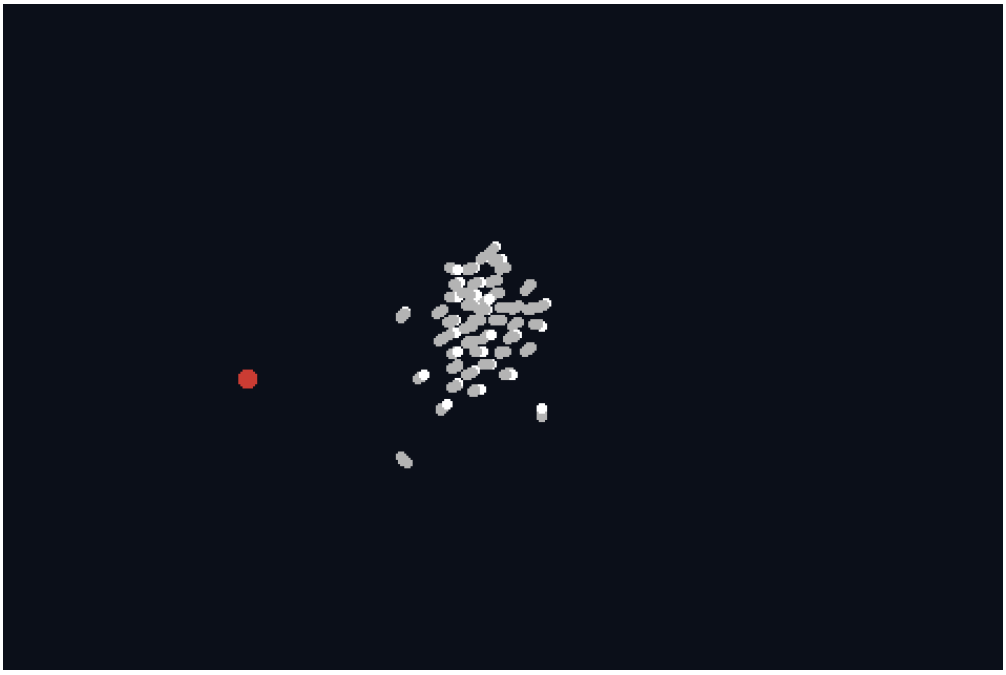


Figure 1: Simulation running with flock and shark.

Discussion, Limitations, and Future Work

The classic boids rules (separation, alignment, cohesion) were adapted from prior work (Reynolds 1987; Lague's Boids repo; Inada 2001). My additions include predictive wall avoidance, a home field, and the predator with two modes.

Limitations: In evil mode, the shark can still move somewhat erratically due to rapid target switching and aggressive lead pursuit. Boids are all identical and lack behavioral states (panic vs cruise).

Future work: Add hysteresis or dwell time for target switching, smooth the shark's desired velocity to reduce jitter, explore multi-predator dynamics, implement spatial partitioning (e.g., grids) for scalability, and complete the FreeWiLi port.

References

- C. W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model," *Computer Graphics* 21(4), 1987.
- Sebastian Lague, *Boids* (GitHub repo). URL: https://github.com/SebLague/Boids
- Y. Inada, "Steering Mechanism of Fish Schools," *Complexity International*, vol. 8, 2001. (PDF available online)

Academic Honesty Statement

I affirm that this report and the associated code were prepared by me, James Widner (ID: 001121770), for CS 3642.

I clearly distinguish between:

- Ideas and rules adapted from prior work: Reynolds' boids model (1987), Sebastian Lague's Boids repository, and Inada's 2001 fish-schooling paper.
- My own contributions: implementation in Go/WASM, predictive wall avoidance, home-field steering, stochastic jitter, and the predator (shark) with elliptical PD patrol, flock-seeking bias, evil-mode interception, cooldown eating, and UI slider integration.

I hope this work complies with the academic integrity policies of CS 3642. I started this project in late August then this assignment came up and I thought it would be a good challenge.