MAT 201B Final Project Report
Evolved Boids by Stejara Dinulescu

Project Description:

Evolved Boids is an evolutionary algorithm created in C++, using AlloLib and Gamma. Built for the allosphere, the agents in *Evolved Boids* move in a 3D space, using Separation, Alignment, and Cohesion rules to simulate flocking, as in the *Boids* flocking algorithm created by Craig Reynolds in 1986[1]. However, this algorithm is evolutionary--each agent starts with unique parameters that govern how they move, how they look, and how they sound (i.e. a "genotype" that governs their "phenotype")[2]. These parameters are evaluated over time, and only agents that are deemed viable can reproduce and live[3].

While starting with a basic flocking algorithm that gives each agent a heading and a center value that it moves towards, I wanted to incorporate individuality in the agent movement which could be passed down during reproduction[4]. Each agent thus has a random flocking attribute, added into the average heading calculation (in the alignment rule). Furthermore, each agent has a unique "move rate" and a unique "turn rate," randomly selected at the start of the program and then inherited, which affects the speed at which the agent travels and turns towards its desired position.

Agents also have lifespans, randomly selected upon birth of the agent, that decrement every frame until the agent dies. This lifespan attribute affects the color transparency of the agent (with older, i.e. closer to zero, being more transparent), as well as the size of the agent. Lifespan can be increased if the agent finds food within 10 seconds of existing in the system--otherwise, the agent dies out. The food particles are the circles floating around the environment with the agents, with their size resembling the amount of life added to the agent's lifespan upon consumption. The agent has to be within 0.05 distance of a food particle in order to "eat" it, and the food is removed from the system upon consumption. If there is less than half of the food particles remaining in the system, the environment will regenerate food particles.

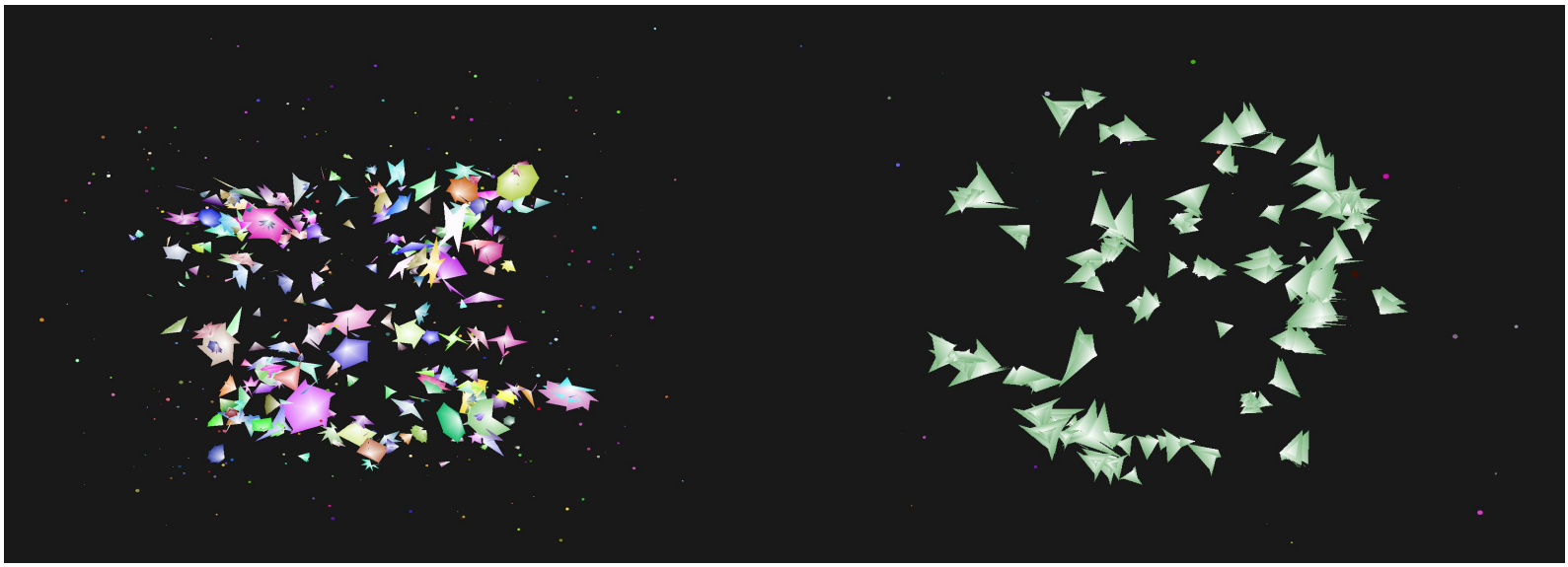
Agent attributes were selected so that the viewer can see how the agent changes over time as it "grows" into adulthood and eventually dies out. Color is randomly selected for each agent, so at the start of the program, each agent has its own unique shade. The evolutionary algorithm starts to come into play upon agent reproduction and agent fitness evaluation[5]. Over the program run time, you can see generations of agents run their life course; upon agent reproduction, most parameters get inherited so that the offspring resemble the average of the parents parameters. Reproduction is only viable, however, if the agent is old enough (to where their reproduction flag turns "on") or if the agent is within a certain distance of another agent who is also able to reproduce. Agents who are most viable (i.e. have the greatest fitness value)

---

[1] Published in *Computer Graphics*, **21**(4), July 1987, pp. 25-34. (ACM SIGGRAPH '87 Conference Proceedings, Anaheim, California, July 1987.)
[2] Chapter on Evolutionary Algorithms in *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies* by Dario Floreano and Claudio Mattiussi.
[3] Ibid.
[4] Ibid.
[5] Ibid.

have a higher chance at reproduction[6], as their fitness value and lifespan are taken into consideration when the algorithm decides whether or not the agent can reproduce.

Each agent's fitness value starts at zero and is updated every frame based on their flocking parameters. If within a desirable range, the fitness value gets incremented accordingly (i.e. proportional with viability of that specific trait). If not, the fitness value is decreased accordingly. At a certain age (randomly decided upon "birth" of the agent), the fitness value is evaluated. If the agent is "viable", it can live. If their fitness value is below a cutoff value, their lifespan becomes zero and the agent dies. The system thus finds equilibrium over time if the agents are able to succeed in their environment (see images below, which display the system close to the start of runtime and after reaching equilibrium). A specific movement pattern emerges, and the color of the agents also moves towards a homogenous RGB value. The environment also has a "fluid field" simulation, where each cube section of the space contains a vector that applies a force on the agents in that area. The environment furthermore implements a random "culling" procedure, where the agents in a particular space are randomly killed, simulating environmental changes or disasters.



In terms of sound, each agent has a "chirplet" sound, created by me using a Gamma sine oscillator. Properties of the chirplet include a center frequency (i.e. starting frequency) and a range that the frequency can move from the center frequency, which determines the agent's ending frequency. This ending frequency can be higher or lower than the center frequency, giving the agent a chirplet that moves upward or downward in pitch. These "chirps" are generated using an ImpulseGenerator class written by Aaron Anderson[7]. Thus, an agent emits its own unique sound at varied intervals, described by the ImpulseGenerator object. The system at the beginning contains a cacophony of sounds, coming from all of the agents. However, as the agents die and the system tries to stabilize, the individual chirps of the agent become more

---

[6] Ibid.

[7] The ImpulseGenerator source code is taken from the Pedal sound library by Aaron Anderson and Keehong Youn, found at https://github.com/aaronaanderson/Pedal commit number (c922916).

distinguishable from the group. If the population of agents dies out before reaching equilibrium, you can hear each agent die by their distinct sound. These sound parameters (center frequency, range, direction of chirp, and chirp duration) are passed down to offspring during reproduction.

Project Challenges, Successes, Failures, and Questions:

The biggest challenge for me at the beginning was understanding the components and structure of an evolutionary algorithm. I had never implemented one before, but I was very intrigued by the idea after seeing "Evolved Virtual Creatures" by Karl Sims[8]. After reading the chapter on Evolutionary Algorithms[9], I had a greater understanding of the components necessary for such an algorithm, including agent genotype, agent phenotype, population size, fitness functions and values, reproduction, and environmental simulations. I learned that evolutionary algorithms are used in industry to find viable options for a goal or outcome. Through creating my system, I was able to understand the process and how the system changes based on decisions made each step of the way.

Another huge challenge was working with shaders, particularly since I needed multiple shaders for the different components in my system (i.e. the agents and the food). However, with help, I was able to navigate my way around a shader and have a better understanding of how they work together to form the visual output we see on screen.

An additional challenge was the system. I didn't have much technical experience with sound in general, and I came in with very nebulous and weak ideas of my visions for what the sound would be like. I initially wanted each agent to have its own unique sound. However, this wasn't working with the structure of how I handled the agents in memory for a while. I refactored my code, using an array instead of a vector to store the agents, and was able to get the sound to work by structuring my own Chirplet class.

I would say that a success of my system is the algorithm itself. There are many factors involved that evolved over time--as I got one thing to work, it gave me ideas for how I would approach the next milestone. I am content with the fact that I was able to not only implement the algorithm but to also have the changes be visible. I was worried that when a viewer looks at the system, they would not be able to tell what was going on. However, I am fairly confident that one can see the changes happening (maybe not all of the nuances, but one can definitely see the system stabilize over time and certain motions and color patterns emerging). I would say the biggest failure of the project is in the visual aesthetic. Currently, my work is in "debug" mode; everything is visually there, but the aesthetic of the system needs to be pushed. With the help from the "diverse agents" example, I was able to change the look of the agents and give them unique shapes. However, I did not have the time and shader expertise to really push the aesthetics in the way that I intended. I would like to keep this project moving forward so that at some point, it can be displayed in the AlloSphere and look like an aesthetically intentional piece. I am excited for these future possibilities and for the addition of more nuances in the algorithm itself.

---

[8] Published in Computer Graphics (Siggraph '94 Proceedings), July 1994, pp.15-22.

[9] *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies* by Dario Floreano and Claudio Mattiussi.