P6: Evo-Boids

Introduction

In this programming assignment you will need to implement, in Javascript, a genetic algorithm (GA) to drive the behavior and aesthetic expression of flocks of birds in a space covered by several food sources. In order to do that, you will need to implement the three main components of a GA: a chromosome, an operation for altering chromosomes (crossover in this case), and a fitness function. The chromosome represents aesthetic and behavioral information associated with flocks (e.g. color, wing shape, or the tendency to align with other boids in the flock). The crossover operation combines two flocks (represented by chromosomes) to create two new flocks. The fitness function evaluates how "good" a flock is. By default, this function simply calculates the amount of food collected by a flock; if flock *f1* eats more food than flock *f2*, *f1* is considered better than *f2.* The GA you build in this assignment employs a chromosome, crossover operation, and fitness function to evolve a population of flocks towards the optimal value of a fitness function. Your goal is to **refine the behavioral and aesthetic information the chromosome will represent** and to **design two alternative fitness functions to evaluate the quality of the flocks.** You will also **implement the crossover operation.**

Genetic Algorithm

Genetic algorithms (GAs) are stochastic mechanisms for searching large dimensional spaces in order to find optimal or near-optimal solutions. The process mimics natural selection by mixing chromosomes of highly successful solutions as well as applying random mutation in order to perturb solutions into new areas of the search space. GAs are composed by a population of chromosomes that are evolved during a certain amount of generations towards the optimal value of a fitness function. A chromosome represents a solution in the search space and can be seen as an one dimensional array of a given size *n*, as shown in Figure 1. Each element of the chromosome is called a gene and in this example it is being represented by a real number.



Figure 1: Example of chromosome of size *n*, where each gene is represented by a real number.

To evolve a population, the algorithm first initializes it and then repetitively performs four operations to produce a given number of generations: evaluation, selection, crossover and mutation. Initialization instantiates a population of a given size by assigning random values within the valid range. The evaluation stage calculates the quality of each chromosome of the population using a defined fitness function. During the selection phase, the GA probabilistically selects a set (of size *n*) of chromosomes (winners) based on their fitness score. These selected chromosomes are used in the Crossover stage, which recombines them two-by-two, forming new chromosomes (offspring). The last stage of a generation is a Mutation operation, which randomly changes, with a very low probability, some genes in the chromosomes. Figure 2 shows a diagram of the GA generations structure.



Figure 2: Screenshot of the web interface for evolving boids behaviours.

In this assignment you have to implement a GA to evolve the behavior and aesthetic expression of flocks of birds. The Initialization, Selection and Mutation stages are already implemented, so you will implement Crossover. The representation of a chromosome has also been defined and implemented for this problem. The flocks are represented as arrays of real numbers, where these values map to behavioral or other phenotypic expression (e.g. color, wing shape, or responsiveness to alignment forces). In the provided implementation, the last four genes (*behavior0*, *behavior1*, *behavior2*, and *behavior3*) are not linked to any behavior. You will extend this representation by associating three of these genes with additional forces that act upon the flocks.

There is a default fitness function already implemented, but your task is to create two alternative ones. The Crossover operation is not implemented and it is your task to implement it. The basic process for this operation consists of four steps:

|  |  |
| --- | --- |
| 1. Selecting a “Crossover Point” randomly along the chromosome sequence. 2. Create two new arrays, which will hold chromosomes for the offspring of   the two selected parents.   1. From the crossover point to the end of the sequence, swap the gene values   within the offspring.   1. Add the children to the population for the next generation. |  |

Example

In order to run and test your GA you need to open index.html in a web browser (assuming you are in the /src folder).

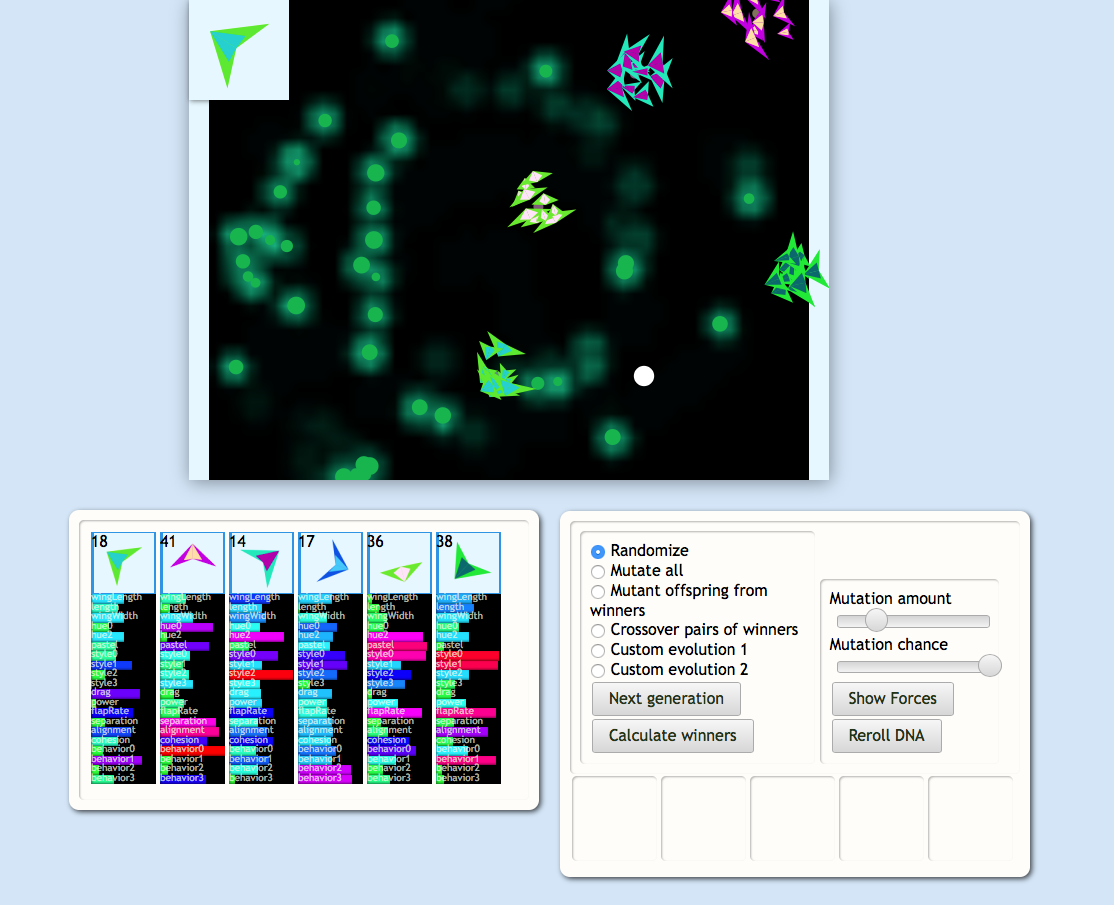


Figure 3: Screenshot of the web interface for evolving boids behaviours.

The interface of this assignment, as shown in Figure 3, has three panels that can be dragged around with the mouse. The center panel is rendering the boids together with some food sources (represented by the green circles). The left panel displays the genomes of the flocks currently displayed. You can adjust the genome values by hand by dragging the corresponding gene. The right panel controls the evolution process and it has the following buttons:

* ***Calculate winners***: selects three winners according to the fitness function. By default, the fitness function simply selects flocks maximizing food collected.
* ***Radio buttons***: provide options for how the next generation of flocks is generated.
  + ***Randomize***: generates new flocks randomly
  + ***Mutate all:*** mutates all existing flocks
    - Mutation chance and magnitude are controlled by the corresponding sliders
  + ***Mutant offspring from winners:*** creates mutated versions from winners
    - Mutation chance and magnitude are controlled by the corresponding sliders
  + ***Crossover pairs of winners:*** creates offspring by performing crossover between pairs of winners
  + ***Custom*** - ignore these
* ***Next generation:*** - uses the selected radio button mechanism to generate the next generation of flocks.

Base Code Overview

**- js/aof/boids/boid.js**

This file contains the behavior and aesthetic expressions of an individual within a flock so **here is where you should extend the chromosome representing the flocks**. In the *preUpdate* function, the boids are moved by a number of forces. Most of them are already implemented, including alignment, cohesion, and separation as well as basic forces to keep them on screen and moving forward. Extra gene values are also added (behavior0 - behavior3) which can be set to influence other types of forces, for example:

* Line 114: A "wander randomly" force
* Line 118: A rudimentary "go after food" force (to show how to query the food map)
* Line 134: A "mouse following" force, turned off (to show how to interact with the mouse).

Try mixing the values for behaviors 0-3 in with these forces, or create your own force.

**- aof/evo-panel.js**

Contains the evolutionary behavior for the flocks. "Mutate all" and "mutate winners" are both implemented as examples of how to access values.

* Line 140 (“breedWinners”): **Here is where you have to implement the crossover operation**, combining the chromosome of two winners.
* Line 177 (“#set-winners”): **Here is where you have to implement your fitness functions**. By default, the fitness function sorts the flocks by how much food they’ve gathered.

Potential ideas to optimize for:

* A specific dispersion level of the flock (not too tight or too spread out)
* Minimizing the distance to the cursor
* A multi-objective function, combining two or more objectives (e. g. evolving successful flocks that are drawn to the cursor ***and***maximize food gathered.)

Requirements

* Extend the flocks representation adding three more behavioral and aesthetic variables.
* Define two alternative fitness functions to evaluate the quality of the flocks.
* Implement the Crossover operation.
* Submit a document answering answering the following questions:

1. What new forces did you add to affect the behavior of a flock? How did you tie in the genetic values?
2. Describe your two new fitness functions. What do they try to optimize? Try evolving flocks for several generations under each. How successfully do the flocks evolve to fit the desired outcomes from these selective pressures? Can you outperform the evolutionary process by adjusting flock genomes by hand?
3. Experiment with evolving flocks using only mutation, only crossover, and a mix of both. Describe your findings in how the evolutionary mechanisms differ over time.

Submission Instructions

Submit a zip file named in the form of “Lastname1-Lastname2-P6.zip” containing:

* A folder named src/ containing your solution using the same file structure as the one provided (Please include all the files).
* A text file named **README** answering the three questions described in the previous section.

[Submission Link](https://goo.gl/0JPAQs)