Final Report

Simulating Bird Flocks

By Dragos Secrieru (1739045)
And Andrew Salem (1734820)

Introduction:

The mind of an animal is something really complex and mesmerizing at the same time; our inability to communicate with them and to understand them makes their behavior even more surprising and interesting. Are their actions solely committed for the sake of survival? Do they have their own free will or are they simply acting by instinct?

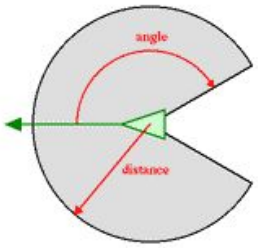
The area in which researchers explore different systems in the real world is called artificial life simulations. One of such simulations is called boids, a program simulating flock behavior in birds. Why is it interesting to analyze this behavior? From an outsider, bird flocks look both unpredictable, complex, and yet ordered in some sense. Is it possible to model such behavior?

For our project, we are basing ourselves on Craig Reynolds' paper: Bird-oid objects. In which he simulated flocking behavior with 3 simple rules. The paper found that emergent behavior arises from the individual boids limited perception.

Description of the model:

Creation of Boids:

Boids are triangular shaped objects. Since they are objects in a code they would normally be able to detect the whole field in which they are created. In order to mimick the real world, we ought to give them a distance of sight and an angle of sight.



This would make the boid more realistic since it will only be able to detect what's in front of it up to a certain distance. The set of rules is based on the fact that every single individual boid has a limited perception.

Set of rules:

1-Separation:

Birds are known for the fragility and frailty of their wings. Therefore when flying together they must always make sure that they never get too close to each other or collision would happen. They must therefore always stray away from each other. In order to do so, we make boids detect all other boids in their lines of sight and then we make them move to a less crowded position.

2-Alignment:

Birds never stray in directions drastically different from each other and they always seem to fly in a similar direction. We must, therefore, make our boids go in similar directions and we do so by making it so that boids update their own velocity vectors in such a way that their new directions will be the average of the directions of all other boids.

3- Cohesion:

Birds want to stay in a group, moving together. In order to make sure that birds don't just leave the pack, cohesion makes sure that every single bird looks inside its range for other birds, and moves in their direction.

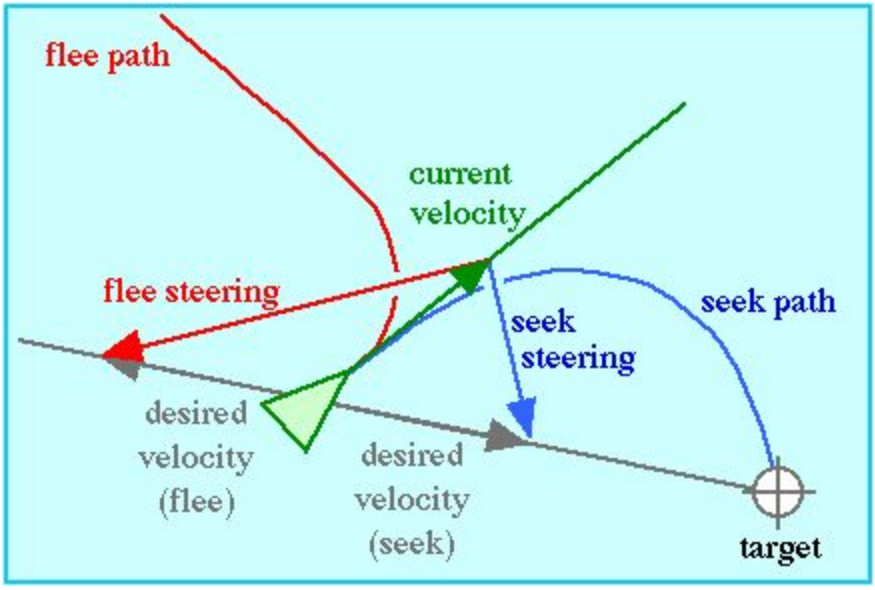
Steering:

It is quite easy to say that the boid will do this and that … But how does it actually do it?

Detection of positions and vectors is doable within its distance of sight but for it to reach the calculated position it must “Steer”.

For an object to steer it must take into account 3 factors: its initial velocity, its desired velocity and the magnitude of the steering vector found.

In order to explain how steering works imagine the following example: assume you have a sentient asteroid that wants to hit earth but that is directed towards Mars. Its initial velocity is the one who's direction is leading it to Mars. Its desired velocity is the one that will lead it to earth. Its steering vector is therefore equal to the difference of the initial speed vector by the desired vector. And in order to know at what speed or "how urgently" it wants to go to earth it can multiply the magnitude of the newly found steering vector by any number it wants thus accelerating towards earth or going slowly towards it.



Other than that, they were no extreme numerical methods used. Vector addition was perhaps the most complicated piece of math that there was in the code.

Research Question:

Our question is then the following: What would happen if we were to keep all the variables in the code the same except for cohesion? In other words, we would like to showcase individualism and group thinking in the same experiment by changing the degree at which boids get close to each other.

This would also allow us to showcase the evolutionary importance of flocking since it is also used by marine species and insects in order to destabilize their predators and escape them more easily. (see discussion).

The code:

This section will briefly explain how the code works and how it was written.

The code contains many important classes :

- Vector : which defines what is a vector and the many equation and operations that apply to them
- Boids : which are our objects stored in a list; they have velocity and acceleration as characteristics, their position is updated according to our set of rules which are defined as

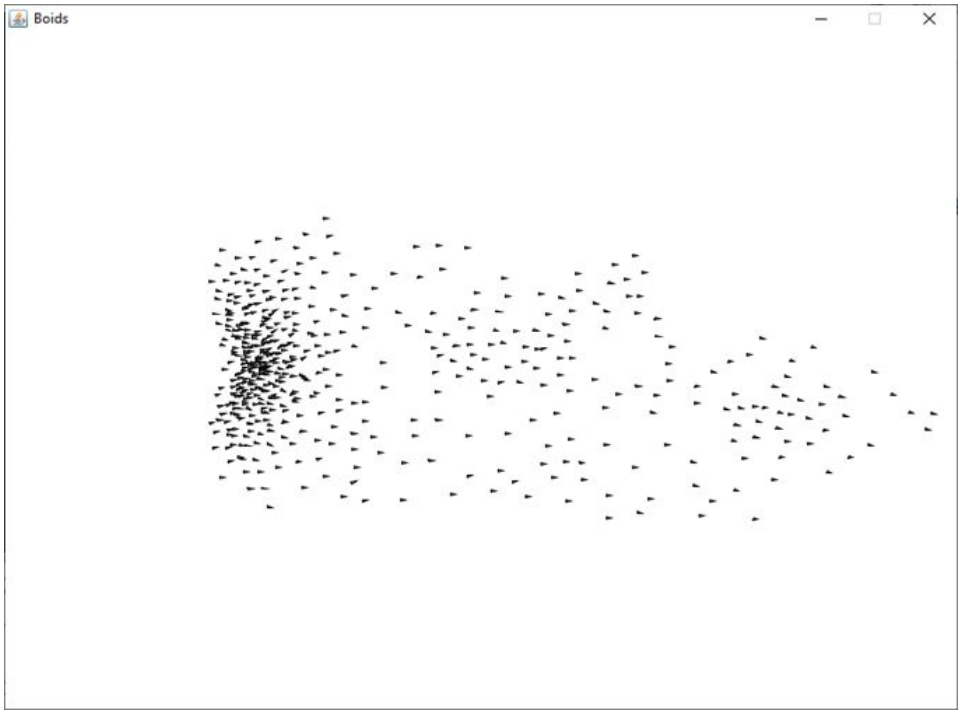
vectors that affect the whole boid population through a set of methods taking their magnitude into account.

- Flock which takes the role of “drawing” the boids and updating their position.

The code was written using the guidelines for the final tpe (final lab test) of the year 2016 of a university in Lebanon we are therefore not allowed to submit it. We have then decided to post the sources used to write those guidelines instead. Note that the backbone of the code (ie: jframe initialization was already given to us by the guidelines and we have then decided to modify them in order to optimize our code).

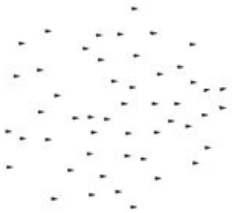
Results:

1. Flocking simulation (200 boids)



The code works “perfectly” well. The variables are well named therefore we can easily change the following variables: eyesight, angle, number of spawned boids, max speed, separation force, alignment force, cohesion force. This improves the reusability of the code and reproducibility of our results.

In order to respond to our cohesion question, we start moving the cohesion variable around.

Cohesion = 0: 

Cohesion = 3:

Cohesion = 1.3:

At c = 0, the birds don't seem to be really affecting each other. The way that they were spawned, is the move that the stay for the duration of their flight. At cohesion c = 3, the birds just clump up together, the cohesion rule completely overpowers the separation variable. The flock acts like one. At cohesion 1.3 we notice flocking behavior similar to flying birds.

At around cohesion = 2.5, something interesting happens. The video of that was shown during the oral. A vortex is created, with the birds moving in and out of the center of the flock.

Discussion:

The results found were to be expected: putting the cohesion at 0 means that the boids will never group back and they will therefore only separate until their direction vectors all point towards the same direction. In that case they don't separate after the first separation. Putting the cohesion at 3 makes it so that the boids are always stuck to each other. There would therefore exist a certain cohesion threshold "after which the boids can't separate".

Finally a notable result is the vortex. Usually used by fish in order to escape marine predators. This means that the model could be used to simulate more than one type of species .

Flocking behaviour is quite interesting, for it is observed in many different species but the change in “rule magnitude” affects the patterns so much that they seem unrelated (see the fish vortex vs a bird flock). As mentioned before it is believed that it is used to destabilize predators : predators have evolved to be perfect targeters; in other words they will target a single prey and keep on following it until it succumbs to them or until it escapes them. A prey flying within a flock is harder to detect since it is always surrounded by its peers, sometimes shrouded and hidden and sometimes visible. The brain of predators, being unable to process the change fast enough , gets too slow and is less efficient.

Finally, its most interesting “façade” comes from the fact that it creates semi chaotic and organized behavior through the application of a simple ruleset. You don’t need more than 3 rules to make an organized and chaotic flock; no need for the detection of everything within the infinity of space or anything as complex. Simply three basic and logical rules.

Note that we would also like to communicate a list of weaknesses that the code has which we will try to solve for another time :

- If the boids were to move in the negative x direction without significant changes in the y direction the simulation could run indefinitely and we wouldn’t be able to see what is actually happening. (that is because of an out of screen method that is only defined in the positive x and both y directions but not on the negative x since the boids spawn there.)

- If too many boids are to be created they spawn badly and create a vortex when they shouldn’t (or they simply overlap).
- The code can sometimes be really slow and it is not well optimized.

Sources:

The way we learned how to code that was from “Coding challenge #124: Flocking simulation” by Daniel Shiffman. This is explained with more detail on his webpage https://natureofcode.com/book/chapter-6-autonomous-agents/. (it’s in javascript but one of us knows how to code in that language)

The video itself uses the original algorithm created by Craig Reynolds in 1986. A simulation of his algorithm can be seen here: https://www.youtube.com/watch?v=86iQiV3-3IA. In this case, the boids were flying in a 3-dimensional space.

Craig Reynolds boids website: https://www.red3d.com/cwr/boids/.

The series of JFrame tutorials by Bryan Chauvin on youtube.
https://javatutorial.net/swing-jframe-basics-create-jframe