Simulating Realistic Crowds Using Rule-Based Systems

Harshpreet Singh
Taheen Rayhan
Yawen Zhang

Introduction

Our project is a crowd simulation that simulates large crowds of individuals. We use birds to represent individuals, each with a mind of its own. All the birds are constantly moving and accelerating; at the start, they each have more or less random accelerations and velocities. However, our contributions to this project changed how these birds would move and gave them patterns to follow, simulating real-life crowds more effectively.

Contributions

Our project added five significant rules to the crowd simulation: cohesion, alignment, separation, random and mouse.

1. Cohesion

The cohesion rule accounts for the "flocking" of birds or animals travelling in herds. Within a given radius, the "birds" travel towards each other. The cohesion rule makes the birds look at every other nearby, take the average position of those birds, and change the bird's trajectory to accelerate towards the average position.
As the first rule, we can see how the bonds start grouping within a particular area of each other. But from what we know, birds do not stack on top of one another and spin in circles, which is where the different rules come into play.

2. Alignment

The rule of alignment describes the behaviour in which the birds fly in the same direction, which increases flock efficiency and security. It adds to the rule of cohesion, where instead of forming flocks that are not moving, the alignment allows migration of the flocks. The process needed to meet this purpose is similar to the rule of cohesion. We first loop through all the nearby birds within a certain radius; then we calculate the average velocity of all the nearby birds by finding the sum of velocities divided by the total number of birds. After that, we need to normalize the velocity vectors – this step omits the magnitude of the velocity vectors so that only their direction will be considered. This is what we wanted since we were only interested in figuring out the direction in which the birds should be flying. Then we started playing around with the static variables, such as the nearby radius, which is at default set to 70; however, we found that if we increase the radius, the objects started to be more widely spread (enormous flock), and if we decrease the radius, the objects started to me more tightly packed (small flock).

3. Separation

The separation rule was added to ensure that the simulation of our birds is more realistic and visually appealing. By adding separation, our birds would no longer stack together as before. The cohesion rule would result in a few stacks of birds, and even without it, birds would overlap while accelerating. We added separation to ensure that these problems would be solved.
To add separation, we made a function that looped through all the birds on the screen (excluding itself) and checked if they were within a certain radius called the separation radius. Our function then calculated the average displacement of all the birds in that radius and adjusted the acceleration to the opposite direction.
We had two variables that would control the output of this rule: separation radius and separation weight. The separation radius was the radius within which the separation rule would be enacted if other birds were in. The separation weight controlled the actual power of the separation.

4. Mouse

The mouse rule is to add in some interactive elements that may allow more applications of this simulation. For example, we can set the mouse as a predator, which the objects will be flying away from, or we can set the mouse as prey, which the objects will be approaching. The coding process is quite simple since Jonathan left us with two handy functions that allow us to check whether the mouse is down or not and get the mouse's position if it is pressed. Then, to redirect the objects' flying direction, we simply subtract the object's current position from the mouse position to get the vector of the displacement direction. The static variable we could play around for this element is the weighting; similar to what Harsh mentioned in the separation rule, the mouse weighting controls the strength of this element compared to the rest of the rules.

5. Random

The last rule that we made was giving the birds a random deviation. This accounts for birds, or animals that travel in packs, deviating from their group, having their own "minds." Take a herd of sheep, for example. Let's say a farmer is trying to gather a herd of sheep in one place. As they are led, there are many cases where the sheep follow a different trajectory than the rest of the group. This is basically what the random rule accounts for, where, as you can see, a group of boys start diverging from the group and follow their path before joining the main flock.

Applications

Our work in crowd simulation has applications in many areas, including entertainment, urban planning, and crisis training. In entertainment, video games and movies need large crowds as a part of their scenes, which can be simulated realistically through our work. In urban planning, crowd simulation can be used to plan out sidewalks, roads, and other areas where large crowds could interact with the city. Lastly, in crisis management, crowd simulation can show how individuals respond to problems like natural disasters, allowing the user to determine how to keep crowds safe.

Shortcomings

While working on this project, we had many things that could have been improved that affected how long it took for our rules to be added:

1. Our ideas themselves needed to be corrected in some circumstances. For example, when creating the cohesion rule, our first idea was to pair two birds together and have them go towards each other. This idea would not result in the flocks we wanted, as many birds would not go towards each other.
2. Our familiarity with the software we were using also held us back in some respects. Some of us needed to become more familiar with NumPy and Pygame, both of which were used in this project to create the visualizations of crowd simulation. As we had never used much of this software, we were unfamiliar with the functions and, therefore, had to spend more time learning about the logic of Pygame and NumPy.
3. Our code readability affected our collaboration.

We had different ideas for our variable names and how we would use logic, which affected how we could understand each other's code and collaborate.

Discussions

Boid's principles were developed by Craig Reynolds in 1986, describing the flocking behaviour of birds; by setting simple rules for individual birds in a patch, we can output an organized group behaviour. Moreover, this simulation involves some physics since we address the bird's acceleration according to the three principles. Finally, by programming the simulation with the help of mentoring, we could improve our programming skills and become more familiar with loops, if statements and pregame functions.

Future Work
Now, working on the Boids simulation program, we realized that Boids is not limited to a simulation of a flock or herd of birds and animals, but there are countless ways to extend what we have so far.
Predator-prey interactions are another prominent feature that can be implemented. When a school of fish encounters a shark, for example, there is a relation where the predator is attracted to the prey. At the same time, they try their best to get as far as they can, even if it means deviating from the flock, and it would be cool to see it implemented in the program.
We are also seeing how bonds interact with obstacles and avoid a collision. There are obstacles all around us in real life, and extending the program to see how bonds navigate in realistic environments like complex buildings, oceanic systems, or mazes would also be cool.
The extensions to this project are not limited to Predator/Prey, Obstacles, and Realistic Simulations. It goes far beyond these examples, offering endless opportunities for development and simulating complex behaviours.