Flocking with Boids

Shijia Hu hsjssl97@gmail.com UID: 405351364
Simeng Pang pangsimeng6@gmail.com UID: 604625053
Chenlei Song chillysong@cs.ucla.edu UID: 505322861
Qiyue Yao qiyueyao@cs.ucla.edu UID: 105349443

University of California, Los Angeles
Prof. Demetri Terzopoulos

June 18, 2020

Abstract

Boids was first developed by Craig Reynolds in 1986. Our group developed a program that simulates and demonstrates the flocking behavior of animals. Basically, we follow separation, alignment and cohesion rules, and determine the behavior of each individual according to those rules. We also added more complex rules into the rule set, including obstacle avoidance, target following and predators. This project was developed and visualized in Unity, a real-time development platform for 2D,3D interaction programs.

1 Introduction

In a typical natural environments it is common to find a huge number of animals, plants and small dynamic particles. This is also the case in other densely populated systems, such as ocean, communities of insects, etc. Computer simulations of these systems are very useful and can be helpful in predicting the animals behavior. These kinds of models are usually based on the flocking boids approach, which is the fundamental part of a flock of artificial intelligence.

Boids was first developed by Craig Reynolds in the late 1980s, it is an artificial life program, which simulates the flocking behaviour of birds, it is a kind of swarm intelligence which has the feature of decentralized and self-organized, in which the activity of the flocking behavior are delegated away from the central, and self organized is that the order arises from local interaction between small parts of this initially disordered system. The complexity of Boids arises from the interaction of individual agents (like a single bird in the environment), and these individual agent adhering to a set of simple rules like separation, alignment and cohesion.

In this work, We present an artificial animal construction framework that has been obtained as a generalization of the existing flocking models, but is not limited to them. Like we apply this framework to underwater ecosystem. We are going to provide a formal definition of the framework including basic boids, multiple flocks and obstacle avoidance, and gives an example of its use. Later this framework can also incorporate the effects of fear and the emotion transition. like predators and how agent will behave in their presense, like escape or some other effects. We also implemented the target following, which can be slightly tuned to imitate the behavior of leadership, and the flow field following, to show how the flock of fish will act in the existence of ocean occurence.

2 Boids Model

There are three typical behaviors with a Boids Model-cohesion, alignment, avoidance. These behaviors are appied on neighbors. Figure 1 defines neighbors. For each agent (yellow fish), draw a radius of r. Any agent (blue fish) falling within the radius will be considered as neighbors. Other agents (red fish) outside are non-neighbors and not studied in the behaviors.

2.1 Cohesion

As shown in Figure 2, cohesion keeps the agents move together with its neighbors. It is realized by finding the average position of the neighbors and navigating the agent towards there.

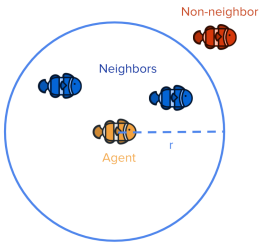

Figure 1: Neighbors in Boids Model

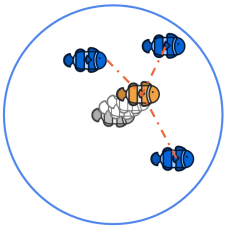

Figure 2: Cohesion behavior

2.2 Alignment

As shown in Figure 3, alignment keeps the flock in roughly the same direction by finding average direction of neighbors and aligning the agent to that direction.

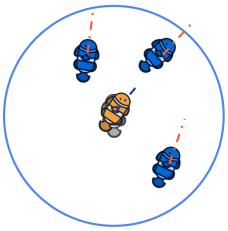

Figure 3: Alignment behavior

2.3 Avoidance

As shown in Figure 4, avoidance prevents agents from overlapping and collision. A new radius is defined smaller than the neighbor radius but larger than the agent itself. If any neighbor is in the red radius, navigate the agent away.

3 Implementation

3.1 Basic Boids

In the implementation, neighbors are found by iterating through all other agents than the target agent and checking radius overlapping. Similar to the theory in section 2, we defined neighbor radius and avoidance radius for the agent. In Figure 5, the fishes are colored based on the number of neighbors. The yellow fish has no neighbors and the red fishes all have many neighbors.

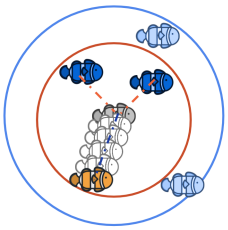

Figure 4: Avoidance behavior

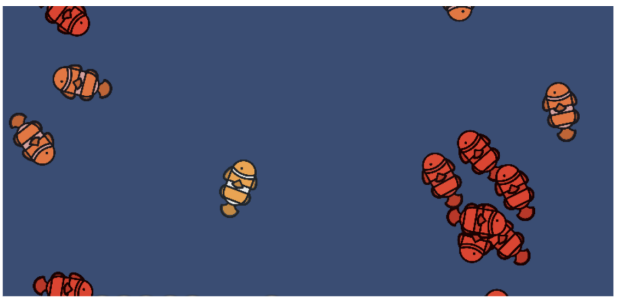

Figure 5: Fish colored based on number of neighbors.

We created a composite of three behaviors and assigned different weights to them for final movement. For instance, we considered alignment more important, we assigned more weights to alignment than cohesion and avoidance.

We also steered cohesion to remove flickering and smoother the movement.

3.2 Target Following

After all the basic Boids behavior such as avoidance, alignment and cohesion, we want to add something more for the fish flock. Normally, fish tend to gather into flocks and move as groups. They like to wander through the water, but sometimes they might want to head to a specific target or led by a head fish to move to some place. Therefore, we need to extend the Boids model for the ability of target following.

As shown in Figure 6 and the video we included, if there is a target at the beginning, all other fish agents would center around the target while following three basic principles. As the target starts to move, fish agents would quickly form a moving flock following the target until the target stops and they would center around the target again.

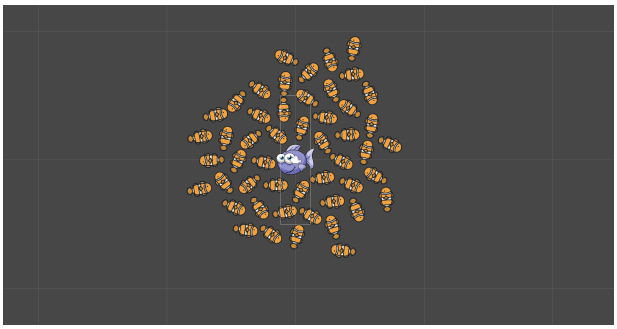

Figure 6: Fish flock center around the target.

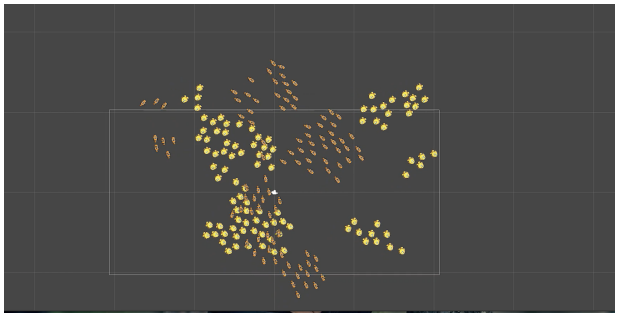

Figure 7: Situation when there are two flocks.

In Unity, we achieve this behavior by always locating the target. If there is a target in the scene, for each fish agent, it always has to look for the target and calculate the distance between target and itself. At the same time, it also has to follow the alignment, avoidance and cohesion principles. Actually, seeking the target is also part of cohesion requirement, since each fish agent wants to remain the connection with the target. After locating the target, each fish agent would swim to the target with a combination force considering target distance and three principles. If the target does not move, the distance between target and fish agent does not change, so the fish flock will not have big movements.

3.3 Multiple Flocks

From the target following example, we see how fish flocking behaves when there is a specific target that is different from fish agent itself. Now that we have seen the basic behavior of a single fish flock, we are also wondering what would happen if there are more than one flocks in the scene. There are some assumption we made for different fish flocks. First of all, we assume all fish flocks follow the Boids model. Secondly, we assume all flocks exhibit the same behavior. Thirdly, we assume each flock acts independently of each other, which means they would not combine into one flock.

The most important technique for separating the flocks is that each fish agent has to know how to filter the neighbors, and there are three steps to achieve this. For each agent, the first step is to identify which flock it belongs to. The second step is to take in a list of transforms, which means looking at all other fish around itself. The last step is to filter these transforms on some criteria.

In Unity, we can achieve first step by setting a new getter method inside the script that creates each individual agent. When the script creates the agent, it can pass itself in for the getter method. For second step, we have to add a filter. Since there are a number of transforms around each agent, it can check transform one by one to see if it is a member of the same flock. If it is, we will add it to the filtered group, otherwise it is not inside the filtered group. Finally, we can add this filter to flock's basic behavior. So now each flock still has the same behavior and each flock has the same filter for neighbors. They now act independently of each other while still swimming together with members of their own flocks.

From Figure 7 and the corresponding video we included, we can see yellow fish groups with yellow fish and orange fish groups with orange fish. When two flocks meet, they cross over each other's path without interacting with the other flock. When they meet their own members, they behave as one flock.

3.4 Obstacle Avoidance

We also want our fish to be able to avoid obstacles. The obstacle avoidance behavior is pretty similar to the basic avoidance behavior of the flocks, however, we might want to assign different weights to the agent avoidance behavior(i.e. the behavior of avoiding other agents) and obstacle avoidance behavior, we'll have to create two different avoidance behavior objects, and need to distinguish obstacle objects from agent objects.

To accomplish this, we need an obstacle filter. Before we can filter the obstacle objects, we need to set a unique property to obstacle objects thus we can easily figure out whether an object is an obstacle. We put all the obstacle objects in the "Obstacle layer". In unity, every object presented in the screen will be put in a layer. By default, all the objects will be put in the default layer. We create a new layer called "Obstacle layer", and set the layer of the obstacle objects to that layer. In the script, we can figure out an obstacle object by checking the layer of that object.

Inside the filter, we check the layer of each nearby objects(objects inside the agent's collision radius),

and only return objects that belong to the obstacle layer. We then get a list of obstacle objects and pass this list to avoidance behavior script. The avoidance behavior script can be reused here, however this time the objects we need to avoid are a list of obstacles.

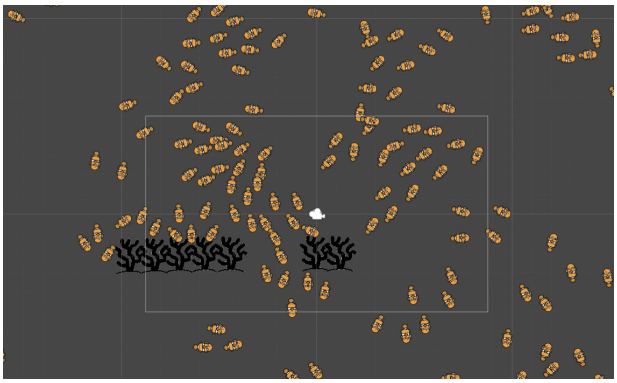

Figure 8: Fish implemented with obstacle avoidance.

As Figure 8 shows, once the flock of fish meets the corals, which are obstacles in our project, agents will try to avoid them by moving towards the opposite direction, and the flock will be divided into two groups.

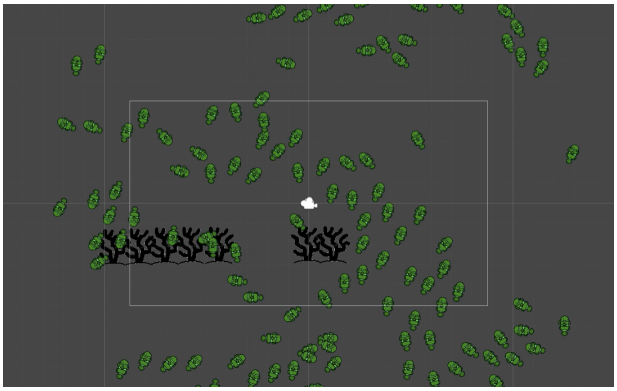

Figure 9: Fish not implemented with obstacle avoidance.

The green fish flock in Figure 9 isn't implemented with obstacle avoidance behavior, so it will ignore all the obstacles and move across the corals.

3.5 Flow Field Following

In reality, there are ocean currents in nature, and fish flocks have the tendency to swim with ocean currents. So we implemented flow field following to simulate this behavior.

We use flow field to simulate the ocean currents. A flow field, also know as a force field or a vector field, defines a mapping from a location in space to a flow vector. The flow vector of a particular location represents the direction of the ocean currents, and an agent has an tendency to move in the direction.

Ocean current should be continuous, but the flow field is discrete. In our implementation, we divided the screen into 1 by 1 grids and each grid has a direction vector, forming a flow field. In order to better simulate the ocean current, the direction vectors of nearby girds shouldn't have sharp difference. Thus we can still see a continuous pattern even if the flow field is not continuous.

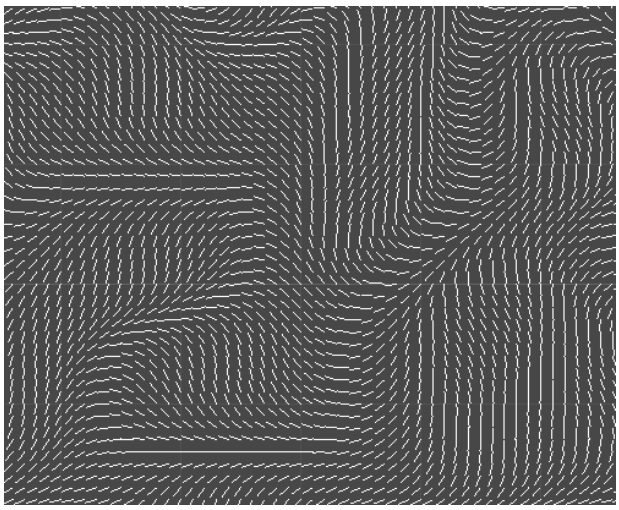

Figure 10: Flow field.

For each agent in the flock, we pass its position to the flow field object. By rounding down its x and y position, we could figure out which grid the fish belongs to. The grid will have a direction vector which represents the direction of the ocean current. We make the fish move towards that direction. Put all agents together, the flock would move with the flow field.

3.6 Predator

To better describe the ocean ecosystem, we need to show the fear the flock is facing, so we implemented predator and prey to simulate the Shark chasing the flock of fish. When a fish discover the appearance of its predator, its fear will be recalculated from the ratio of its comfort distance to the distance between itself and the shark. After doing the summation of the fear based on the existence of all the sharks, a new speed will be add to the fish's original speed. And the fish will move away faster from its original position.

For each shark in the flock of sharks, it will check the fish which is closest to itself, and find out the

prey's status, whether it has been eaten by other shark, and then the shark will chase the closest fish by adjusting its direction towards the fish and improve the speed so that it can chase its target.

Figure 11: Predator.

When the distance between a prey and a predator is too small, which means the fish has already been eaten by the shark. The fish's death time will be calculated and its status will be set to death. At this time we cannot see this fish from the screen. And after a while, which is a certain amount of time, the fish will be respawn by the flock, and it can become the predators' target again.

4 Conclusion

Boids models simulation of flocking accurately by studying individual behaviors in a flock. Cohesion, alignment and avoidance shape the basic flocking, where a flock stays together and move towards a direction. There are many influencing factors in the environment that would affect flock behavior, most of them could be modeled in a similar way by analyzing individual behavior.

Unity provides suitable tools for implementing boids, including generating flock of agents and defining behaviors. We also took advantage of Unity visualization.

5 Future Direction

We chose some typical behaviors to implement. They all map to some real world situations, such as flow field maps to ocean current, obstacle maps to corals, food chain maps to predators, zoology maps to multiple flocks, etc. However, there are many more environmental factors in the real ocean life that we did not include, and the fact that various species exercise different individual behaviors.

Artificial life with Boids could be an extensible interdisciplinary area. For future work, we could discover, analyze and implement more flocking behaviors and wander, containment etc. And it is also promising to try out different combinations of the behaviors to simulate different species.

References

[1] Charlotte Hemelrijk and Hanno Hildenbrandt. "Schools of fish and flocks of birds: Their shape and internal structure by self-organization". In: *Interface focus* 2 (Dec. 2012), pp. 726–737. DOI: 10.1098/rsfs.2012.0025.

[2] Lorenzo Mori. *Flocking Behaviour, a Unity3D AI experiment – Lorenzo Mori*. Jan. 2018. URL: http://www.lorenzomori.com/unity3d/flocking-behaviour-a-unity3d-ai-experiment/.

[3] Craig Reynolds. *Boids*. 1995. URL: https://www.red3d.com/cwr/boids/.

[4] Craig Reynolds. *Steering Behaviors For Autonomous Characters*. 1997. URL: http://www.red3d.com/cwr/steer/.

[5] Craig W. Reynolds. "Flocks, Herds and Schools: A Distributed Behavioral Model". In: *SIGGRAPH Comput. Graph.* 21.4 (Aug. 1987), pp. 25–34. ISSN: 0097-8930. DOI: 10.1145/37402.37406. URL: https://doi.org/10.1145/37402.37406.

[6] John Toner and Yuhai Tu. "Flocks, herds, and schools: A quantitative theory of flocking". In: *Physical Review E* 58.4 (Oct. 1998), pp. 4828–4858. ISSN: 1095-3787. DOI: 10.1103/physreve.58.4828. URL: http://dx.doi.org/10.1103/PhysRevE.58.4828.