BRAC UNIVERSITY

Undergraduate Thesis

Computer-Generated Artificial Life Model: Algorithm for Hunters Hunting Preys

by

Md. Mahbub-Ul Haque
Surid Imam Khan Sur
Rehnuma Binta Shahriar
Supti Rani Saha

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University

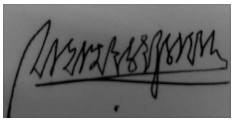
Approval

The thesis titled "Computer-Generated Artificial Life Model:Algorithm for Hunters Hunting Preys" submitted by

1. Surid Imam Khan Sur (17201117)
2. Rehnuma Binta Shahriar (18101421)
3. Md. Mahbub-Ul Haque (18101428)
4. Supti Rani Saha (18101156)

Of Summer, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on September 26, 2021.

Examining Committee:
Supervisor:
(Member)



Dr. M. Kaykobad
Distinguished Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Tanvir Kaykobad
PhD Student
School of Computing
Queen's University, Kingston, Canada

Head of Department:
(Chairperson)

Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

This thesis simulates the behavior of predators and preys observed between a pack of wolves and a deer herd during a hunting scenario. We use the Boids flocking algorithm, an artificial life model developed by Craig Reynolds to simulate the deer herd. We have also enriched the existing algorithm to help flocks recognize their surroundings, like their neighbors and predators, and react accordingly. The pack of wolves has a leader, and they coordinate as a group isolating a deer from the herd and then capturing it. To implement the predator's behavior and intelligence, we have developed some algorithms applying their hunting procedure. A part of the research is to explore simple rules that the wolves can follow to achieve their hunting goals. We have also worked on the concept of the Convex Hull algorithm, that can help the predators define their attack positions. Interruptions in the hunting environment between prey and predators that affects the actions of preys and predators have also been studied. In this experiment, actions by the prey and predators are compared in terms of realism.

Keywords: Simulation and Modelling, Boids, Flocking, Predator, Prey.

Acknowledgement

This project would not have been possible without the help of the Almighty. Also, we would like to thank Dr. M. Kaykobad for introducing us to the problem.

Table of Contents

List of Figures

Chapter 1

Introduction

In nature, predators have to effectively hunt the prey for their survival, whereas preys also like to stay safe from the attacks of predators roaming nearby. The animals stay in a group or create flocks to stay safe from the attacks of predators. On the other hand, predators also stay in a group to increase their possibility of hunting prey together. These behaviors of prey and predators for ensuring their survival can be simulated on top of the work by Craig Reynolds(1986) [4] in the Boids algorithm. Reynolds proposed several rudimentary rules that, when stimulated, can create flocking behaviors as seen in nature by a school of sheep, a flock of birds behavior, or even among humans while escaping a building on fire. Our goal for this thesis is to simulate animal interaction in nature. Specifically, we simulate a deer herd trying to escape from the attack of a group of predators and how predators force some prey to be isolated and captured by them. The dynamics of the movement of prey and predators appear very interesting to us. From this problem, we can see deer as prey, how they move, and the communication between the wolves of a pack while hunting. Since the whole dynamics of movement are governed by intelligence, elements of artificial intelligence have been applied in the program to simulate the behavior of prey and predators applying their intelligence for reaching their goals.

In particular, artificial Boids algorithms have been applied to the movement of predators like wolves in hunting preys like deer. This conception of artificial intelligence and its process of working has inspired us to make the simulation. This, in particular, challenges us to develop an artificial Boids algorithm for the wolves. However, our objective is not limited only to prey-predator behavior simulation. We can think of the behavior of human beings caught in a fire in a multi-storied building and finding a way to escape. If their behavior can be adequately simulated, we should plan for escape routes through which would-be victims of fire hazards can save their lives. Alternatively, for that matter, we can design gates for a concert in a stadium where spectators would like to reach their seating place in a minimum amount of time or chaos. The following section defines a flock of predators and preys and flocking behavior to understand their movement better.

1.1 Flock and Flocking Behavior

A flock gathers a group of the same animals to forage or travel with one another (Figure 1.1). Flocking is a behavior of collective motion by a group of self-propelled entities, which is considered an emergent behavior arising from simple rules that

Figure 1.1: Birds traveling in flock (source:[20]) and Fishes flocking behavior (source:[21])

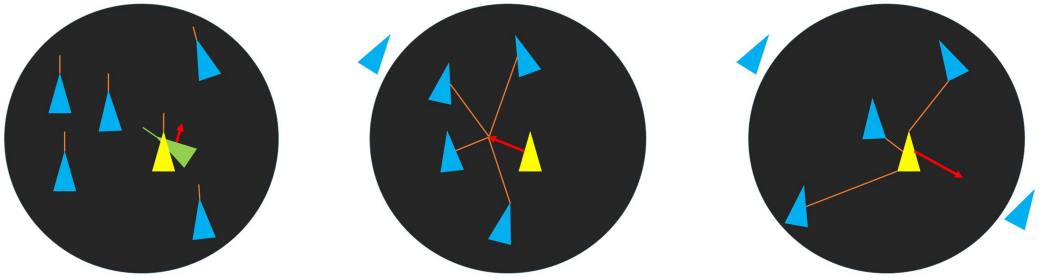


Figure 1.2: Alignment, Cohesion and Separation

are followed by individuals and do not involve any central coordination (Figure 1.1) [27]. By analyzing the behaviors of prey and Predator, we found that prey follows the flocking behavior.

Mathematical models and computer simulations can emulate these Flocking behaviors. Reynolds [4] described the flocking model consisting of three steering behaviors that depend on their nearby flock mates' position and velocity as 'Separation,' 'Cohesion,' and 'Alignment' (Figure 1.2). All this information we studied from the previous works done by others Which we are going discuss in the next section.

1.2 Literature Survey

Flock and flocking behavior has been well studied in the literature. There are so many papers on this topic. Flock and flocking behavior was first introduced by Craig Reynolds in 1987. Reynolds [4] described in his paper 'Flocks, Herds, and Schools: A Distributed Behavioral Model' the schooling, swarming, and herd behavior of a group of animals with flocking behaviors that can be implemented in computer-generated simulation using the Boids algorithm. This paper mainly addresses bird behavior. The model in this paper is based on simulating the behavior of each bird by avoiding collisions and maintaining separation independently, and moving in a particular direction. In this paper, Reynolds mainly presented a model incorporating polarized, non-colliding, and aggregate motion to observe whether the birds try to stick together and move in a particular direction and avoid collisions with one another and other objects in their environment. Hence, it is difficult to measure to what extent these simulation objectives are valid.Then Bajec, Mraz,

and Zimic [7] published a paper named 'Boids with a Fuzzy Way of Thinking' that mainly focuses on implementing one of the Boids steering forces using fuzzy logic. The primary purpose of simulation is to show if the alignment steering function has the most significant influence on the Boids ability to flock. The alignment steering function depends only on the velocities of the observed Boids flock-mates and ignores their position. Hence, the final suggestion is to use fuzzy logic as the basis of the Boids decision about its next step as a tool for constructing artificial animals (animats). However, it is not based on Newton's laws of motion. Again with the idea of fuzzy logic with the other three forces of Boids, they also explained the position and the speed of the Boids. However, the paper 'Autonomous Boids' by Hartman and Benes˘ [9] discusses an exciting observation on Boids which can be taken from the perspective of artificial life. They added two more functions to the Boids algorithm. They added leaders to the Boids who will lead their flock-mates. Also, they calculated the center of density and the position vector to gather the information of the Boids position by using global direction, which is called "Eccentricity." The application is written in C++ and uses OpenGL, GLUT, and GLUI to aim for real-time simulations. It is mainly a simple an extension of idea of Boids. This model provides realistic results that are in tune with the expectations and correspond visually to the flocking behavior, and in the future, it will be implemented in virtual reality hardware.'Flocking Boids with Geometric Vision, Perception, and Recognition' is the paper by Holland and Semwal [10] with the purpose of visual perception in flocking. Light Reception Based and Geometric based visualization are some aspects of 'Boids with Eyes.' However, the other important aspect of the vision process is visual perception. 'Perception Algorithm' is another term that has been proposed for gathering information from Boids environment based upon its visual model. Moreover, there is a phase in the Perception Algorithm in the 'Pattern Recognition' phase to compare each visible neighbor Boids with their own pattern. This paper's ultimate goal is to determine whether a species of birds form flocks with or without visual models.'Simulating Species Interactions and Complex Emergence in Multiple Flocks of Boids with GPUs' is a paper-based on Agent-based modeling (ABM) by using GPU written by Alwyn V. Husselmann and Ken A. Hawick [13]. Here, the authors wanted to explore the use of GPU data parallelism to simulate large numbers of Boids. For example, agents semi-interactively can explore some new parameters and are introduced to simulate multiple and competing distinct flocks of agents. GPU is beneficial for parallel applications and can support multiple distinct flocks or clusters of individual agents. This paper is an extended version of Reynolds's Boids model to show how this model includes different containment geometries in a multi-flocking Boids system. The paper on 'Survey Paper on Swarm Intelligence' is based on ants, bees, and birds' where Keerthi, Ashwini, and Vijaykumar [15] collected the behavior that represents the collective behavior of a decentralized, self-organized system. Moreover, in this paper, they introduced Doctor Kennedy and E Berhart's (1995) [6] particle swarm optimization (PSO), developed from swarm intelligence. By using SI with PSO, the authors describe the behavior and the principles of movements of birds in swarms. Though, PSO mainly concentrated on the math's basic theory where the particle can move stably by analyzing the condition's stability. As a result, the three most popular and successful SI optimization techniques are focused in this paper by using the idea of SI.The paper 'Time-Varying Data Visualization using Information Flocking Boids' is writ-

ten by Andrew Vande Moere* [8], which is based on self-organization and behavior simulation principles that can represent dynamic data evolution by extending the concept of information flocking. Time-varying data-sets contain data objects that are altered in time due to continuously executed, time-dependent data updates. Static State Replacement, Time-Series Plots, Static State Morphing, Control Applications, and Equilibrium Attainment are used to visualize time-varying data-sets. This paper has described the algorithms and principles that drive the information flocking method and how they can be applied to a real-world and almost chaotic data-set.Then Kapi, Sunar, and Zamri's paper 'A Review on Informed Search Algorithms for Video Games Pathfinding' [23] is based on Pathfinding algorithms that have received increasing attention in many applications such as video games and robotics, metabolic pathways in the past few years. The ideal algorithm A* can speed by enhancing it from various perspectives as it reduces memory consumption and no post-processing is needed. By using swarm intelligence in the pathfinding technique, so many other fields are growing. Hence, the recent research progress in the game's navigation emphasizes optimization using four different perspectives.The paper 'Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives' written by Sengupta, Basak, and Alan Peters II [18] is based on Particle Swarm Optimization (PSO), which is the foundations and frontiers of advances in PSO. Here, Fuzzy Adaptive (FA) is used to make fuzzy sets and membership rules. However, The topology of the swarm of particles sets up a degree of a network of its members.According to Artaxo et al., paper 'Autonomous Cooperative Flight Control for Airship Swarms' [19] is a paper investigating the design and implementation of controllers for autonomous cooperative airships flights using two different approaches. Two different swarm strategies have been tested based on the Reynolds Boids algorithm and the Robotic Particle Swarm Optimisation algorithm paper.Then Hahn et al. wrote the paper 'Foraging Swarms using Multi-Agent Reinforcement Learning' [22], where they evaluated the Boids foraging with the help of multi-agent reinforcement learning (MARL). They also showed that each predator moves freely without provoking their flock-mates. Lastly, they evaluated that each predator learns from the flock-mates and can easily select its targeted object. Using the idea of the Boids Algorithm, Mavhemwa and Nyangani wrote the paper 'Uniform spatial subdivision to improve Boids Algorithm in a gaming environment' [16] about creating a video game that simulates the real-life crowd behavior. He also showed several approaches that can improve the flocking simulation. A robotic environment full of fish school and this fish school's fish maintaining the Boids algorithm is written by Connor et al. in their paper ' Analysis of Robotic Fish Using Swarming Rules with Limited Sensory Input' [17]. They also showed the benefits of using Virtual Reality that can make the fish modify itself and can interact with the flock-mates easily.'Simulation of the Flocking Behavior of Birds With the Boids algorithm' written by Erneholm [12], which undergoes the flocking behavior. Moreover, along with the algorithm, he compares two different neighborhoods of Boids.Joselli et al.'s paper 'A Flocking Boids Simulation and Optimization Structure for Mobile Multicore Architectures' [14] is the paper where they use the approach of flocking Boids Algorithm to high-end mobile multicore architectures. They used the flocking Boids algorithm simulation on the Android render script API. For the development of our day-to-day life, people are converting their lives to robotic systems. For this reason, Kasmarik et al. made the evolution of the Boids

Flocking algorithm and Swarm behavior in their paper 'Autonomous Recognition of Collective Behaviour in Robot Swarms' [24]. In their paper, they used several functions to recognize the flocking behavior to simulate a robot. For our thesis implementation we also used some ideas about convex hull algorithm. Jarvis first wrote the paper 'On the Identification of the Convex Hull of a Finite Set of Points in the Plane' [2] that gives the proper definition of the convex hull algorithm. The algorithm shows that we can create a convex hull with some finite number of points by three simple steps.But while calculating the convex hull of a simple polygon, people were failing to use the algorithm. In 1981 Toussaint and Avis wrote the paper, 'On a convex hull algorithm for polygons and its application to triangulation problems' [3], where they showed how using the externally weak visible polygon can easily make a convex hull. We can make internal and external triangulate of the polygon at a particular time. Lastly, we used the simulation plateform 'Godot' to implement our ideas in a visual representation. For this reason, we took the help of the book 'Introducing Godot: Why Migrate?' written by Thorn and Alan [25]. All this previous works inspired us and helped to determine our objectives of the thesis, which we talked about in the next section.

1.3 Objectives of the Thesis

Our research is motivated to simulate a natural phenomenon between a herd of deer and a pack of wolves. We also want to simulate the movements derived by the flocking algorithm and the effects on flocks by their surrendering flockmates. Moreover, another main objective of this thesis is to show the way of hunting and communication between the wolves, which ultimately represents a situation of real-life hunting and the outcome. We want to observe the reaction caused by the artificial intelligence of both deer herds and wolves in some artificial situations. This whole analysis leads us to the understanding of the decision-making capabilities of artificially generated life models.

The objectives of this research can be divided into two main sections, implementation of prey behavior using Boid's algorithm introduced by Reynolds [4] and simulation of the predators' behaviors.

Implementing Flocking Behavior of Prey

Our first objective is to simulate the behavior of prey. Using Boids Algorithm by Reynolds [4], we set the moving rules for each prey individually, which helps to simulate the flocking behavior. We also using fuzzy logic by Bajec, Mraz, and Zimic [7] to get a more accurate and realistic simulation, implying the individual prey's observation and analytical capability within a range. Also, by adding an escape behavior to prey, the whole movement of the prey group can be controlled.

Simulating Predator's Characteristics

Our second objective is to simulate predators. The group of predators will be led by a leader, the movement and diction of other predators in the group will be manipulated by the leader's decision. To make the formation of hunting and communicate with each other, swarm intelligence can be implied. We will introduce simple rules like

detecting the target, chasing the target, isolating the target, and terminating the target, making the simulation of the wolves hunting.

Our other objective is to analyze the outcome from the simulation and compare it with real-life scenarios. These will lead us to make predictions in situations similar to the simulations. Moreover, our research may interest other researchers in working on more real-life scenarios and unleash new scope of this thesis like determining the fire execution path of a building compound, estimating the path of movement of a mob, swarm drones, or robots, war fair simulations and, other natural phenomenons. After we had discussed the objectives, we had to organize our thesis in a order. Which helps the readers to follow the way, also helps for better understanding.

1.4 Organization of the Thesis

In Chapter 1, we introduce the flock and flocking behavior and a brief literature survey. We discussed the objectives, scope, and organization of this thesis. In Chapter 2, we have made a review of literature in the field. We also elaborately covered more important papers discussing Boids flocking algorithms, convex hulls, and video simulations. In Chapter 3, we have discussed the idea of animal behavior and some basic rules of simulating prey. We have also discussed the problems we have faced in simulating behaviors of prey and predators. Chapter 4 contains details of the algorithms and their implementation. We have simulated behaviors of multiple intelligent predators led by a leader and multiple prey. We have used appropriate figures and diagrams to increase comprehension of the process. Chapter 5 contains the results that were expected before the experiment, which was guessed by analyzing the theories. The result we got after the experiments is also in this chapter and compared with the expected results. We concluded our thesis in Chapter 6. This chapter contains the concepts of this thesis's future work, which can motivate other researchers to work further with this topic. In the end, a section of the bibliography contains all the references given in the whole thesis.

Chapter 2

Literature Review

There are many research articles studying behavior of preys and predators and simulating their behavior. We have selected some articles that appear very relevant to our interest. In the following sections we shall summarize the findings of these articles. As there are many research articles, in this chapter, we only focused on the articles we deemed necessary in our research. Reynolds [4] first showed the simulation of a flock of birds or a school of fish using three rudimentary forces: *cohesion, alignment, and avoidance.* Bajec, Mraz, and Zimic [7] worked furthermore with these three forces to improve the flock's moving behavior according to the degree of truth. Combining fuzzy logic with cohesion, alignment, and avoidance helped implement the reason for the flock's moving. The fuzziness in their algorithm added a layer of realism to their simulation. Hartman and Benes [9] combined the steering function with the center of density and position vector to find the initial state and calculate the stable position of the Boids in a particular environment. It adds an autonomous movement ability to the group of the flock. Keerthi, Ashwini, and Vijaykumar [15] provided a survey on swarm intelligence where they talked about the collective behavior of decentralized, self-organized systems of animals living in nature. This survey provides a pattern for implementing Raynold's [4] algorithm. Observing this type of survey, Artaxo et al. [19] implemented the robotic swarm with some self-organizing strategies like robustness, flexibility, stability and, emergence. On the video games platform Kapi, Sunar, and Zamri [23] implemented the flocking algorithm simulation in a new way. Lastly, implementation of Flocking Boids Algorithm is visually eye catchy by the idea of R.A. Jarvis's [2] convex hull algorithm. Now we are going to elaborately describe these papers.

2.1 Flocks, Herds and Schools: A Distributed Behavioral Model

In nature, we can see a group of fish swims in the same direction in a coordinated manner known as schooling. Also, if some similar animal moves together, that can be identified as swarm behavior or swarming. Again, if a group of animals collaborates without centralized direction, herd behavior can occur in animals' in herds, packs, bird flocks, fish schools, and humans. For example, voting, riots, general strikes, supporting events, religious gatherings, everyday decision-making, judgment, and opinion-forming, this act can be considered as human-based herd

behavior. This schooling, swarming, and herd behavior are flocking behaviors that can be implemented in computer-generated simulation using the Boids algorithm.

Reynolds [4] first reported on the Boids Algorithm. Here he observed an animal's behavior and how the animals stay in a group or create flocks. He mainly focused on three fundamental geometry theories that have been used in animation or computer-aided design. He described the flocking model as consisting of three steering behaviors which are the behaviors that help the autonomous characters to move realistically with the help of some simple forces that depend on the position and the velocity of their nearby flock mates. These three behaviors are — cohesion, alignment, and separation.

Let the current Boid be x. The set of boids in its region are in set S. Each boid has 4 components:

p = position

v = velocity

d = direction facing

m = mass

Separation

At first, Reynolds [4] showed the Boids of animals' steer to avoid the collision of other flock mates. He named it 'Separation'. He showed that collision avoidance and dynamic velocity matching are comparable. These two ensure that the flock mates can move without conflicting with any other flock mates. It mainly depends on the relative position of the flock mates.

Say, there is a Boids b_i in the position of p_i with the velocity of v_i and another Boids b_j visible with the position p_j. So, the $Separation_{Force}$ is,

$$Separation_{Force} = \sum_{\forall b_j \in v_i} (p_i - p_j) \tag{2.1}$$

Alignment

Secondly, he described how the Boids steer towards the most of the Boids is going. It mainly depends on the velocity of the Boids. Most of the Boids steer towards the average Boids vector as the vector says the way and the speed of the Boids. Alignment is a behavior that causes particular Boids to line up with agents close by.

Let there are 2 Boids b_i and b_j along with the velocity v_i and v_j. Here the total visible number of Boids are m.Then the $Alignment_{Force}$ will be,

$$Alignment_{Force} = \sum_{\forall b_j \in v_i} \frac{v_j}{m} \tag{2.2}$$

Cohesion

Thirdly, he showed the Boids movement towards the average position of the steer. Cohesion is a behavior that causes agents to steer towards the "center of mass" - that is, the average position of the agents within a certain radius.

Let at first, in the position p_j of the Boids b_j. now the total number of Boids (m) in the region of s where the center is c.

so, the density at the center(c_i) with all the visible boids:

$$\mathrm{c_i} = \sum_{\forall b_j \in v_i} \frac{p_j}{m} \tag{2.3}$$

Now, at the position p_i of the Boids b_i try to merge towards the dense center with the velocity of v_i. so the $Cohesion_{Force}$ will be:

$$Cohesion_{Force} = c_i - p_i \tag{2.4}$$

Combining the Forces

The flock of Boids work by combining these three forces together. Now let,the total force (F_t) of the Boids:

$$F_t = Separation_{Force} + Alignment_{Force} + Cohesion_{Force} \tag{2.5}$$

The maximum force a Boids is F_m. So, the final force F will be the minimum value of the total force (F_t) and maximum force (F_m):

$F = \min\ (F_t,\ F_m)$

Now, the current Boids x's new velocity(v), position(p) and direction(d) will be,

new velocity, $x.v = \frac{F}{x \cdot m}$

new direction, $x.d = \frac{\overline{F}}{|F|}$

new position, $x.p = x.p + x.v.\Delta t$

Reynolds's paper inspired others to work on this process and some of then extended Reynolds's idea which can be seen in next sections.

2.2 Boids With a Fuzzy Way of Thinking

Bajec, Mraz, and Zimic [7] described Reynolds's [4] Boids algorithm with fuzzy logic in their paper. Fuzzy logic is an approach for computing based on "degrees of truth" rather than the usual "true or false" (1 or 0) Boolean logic on which the modern computer is based. It will help decide on the animats where animats are the artificial animals seen in modern computers. Here they showed Reynolds's [4] Boids algorithm is more helpful with Fuzzy Logic for representing an animal's behavior and decision about their next step. In the paper, they showed that animals form groups for survival and food. So, forming a group depends on the Predator's behavior and the type of food source. Reynolds used three primary rules, collation avoidance, velocity maintenance, flock centering. By these laws, we can say that the Boids tend to be in a particular motion where those cannot exceed a certain velocity, though those were moving in a continuous acceleration, and a finite amount of

energy is considered there. Those Boids are mainly specified by their position, moving direction, and speed, bounded by a maximum speed and force. Bajec, Mraz, and Zimic [7] gave two definitions depending on the boids specification. The first definition is about 'Mapping the Transitional Function' and the second definition is about 'Defining State at a Discrete-Time'. In the following two subsections we discussed this defections briefly.

2.2.1 Mapping of Transition Function

An animat A consists of different types of functions defined by X, Q, Y, δ, λ, P, S, B.

Here X, Q and Y are the non-empty set of finite functions which represent:

X = Input Alphabet

Q = Internal States

Y = Output Alphabet

Perception function is an algorithm which predict values with the combination of different sets or vectors according to it's surroundings.

Now, let P be the vector of perception function. So,

$P = P_1, \ldots\ldots, P_k$

They used a vector of steering function that represents position and velocity (speed and direction) with the location of a Boid in an environment S.

$S = S_1, \ldots\ldots, S_m$

Then they used a mapping function called behavior function B which represents the animat's wish to optimize his personal goals' satisfaction.

Lastly, δ and λ is the mapping functions which is called the output function and transition function accordingly.

A transition function is a map from one coordinate position to another [26]. To determine the mapping of transition function we will need the total number of perception function of a Boid can have and for this, we will be using an equation where,

N_i = Total number of perception function

P_i (x, y) = Perception Function

$i = 1$ to k with k being the number of iterations

and the equation is:

$$\mathrm{N_i} = \mathrm{P_i(x, q)} \tag{2.6}$$

Then we will use the number of perception functions that the Boid is using. We can calculate the number of steering functions a Boid can have, and then we can have the total number of steering functions having all the behavior. And for these, we are using an equation where,

F_j = Total number of Steering function

$S_j((N_1, ..., N_k), q)$ = Steering function

$j = 1, ...l$, Iteration number

and the equation is:

$$F_j = S_j\left((N_1, \ldots, N_k), q\right) \tag{2.7}$$

Finally, we calculate the behavior functions connected to the transition functions, which is the top way to find the transitions of the Boids. If we consider B as the behavior function and f as the function which has all the behavior of the Boids, then B will be the Behavior Function containing all the steering function, $\delta(X, Q)$ will be the Transition function considering m as the Iteration number then the equation where,

$B((F_1, ..., F_m),Q)$ = Behavior function with internal states having all the steering function

$$\delta(\mathrm{X}, \mathrm{Q}) = \mathrm{B}\left((\mathrm{f}_1, \ldots, \mathrm{f}_1), \mathrm{Q}\right) \tag{2.8}$$

After that we are going to learn about the second definition where the state at a discrete-time is defined.

2.2.2 Defining State at a Discrete-Time

By the behavior function, we can have the transition of a Boid, but we cannot determine the state of a Boid where the Boid is in which position and velocity and speed. For this, we need these functions to determine the states,

$p(t)$ = Position in space in respect to time

$v(t)$ = Velocity of the Boids in respect to time

r = Radius of the Boids

f_{ov} = Boids field of view

m = Boids mass

max_s = Boids maximum achievable speed

max_f = Boids available force

so current state,

$$\mathrm{q(t)} = (\mathrm{p(t)}, \mathrm{v(t)}, \mathrm{r},\ f_{ov}\ , \mathrm{m}, \mathrm{max_s}, \mathrm{max_f}) \tag{2.9}$$

In the thesis paper, they describe the steering function, and with its help, they try to find the position of the animats.

The Steering Function of a Boids is:

$$S_a\left(N_f, q\right) = \left(1/\mid N_f \mid \sum_{(S_B, q_B) \in N_f} \left(q_v + s_B\left(q_{Bv} - q_v\right)\right)\right) - q_v \tag{2.10}$$

Which depends on only velocity,

N_f = set of all observed Boids flockmates

q_v = observed Boids velocity

s_B = State of the observed boid

q_{Bv} = velocity vector of B

After talking about the steering function, it is said that the Boids cannot sense crisp value. They understand the relative difference between heading and speed.

Here, H_d = Boids heading difference and shown as,

H_d (Left, Same, Right)

Again, H_d represent that the Boids can head towards left and right. Also, it can stay in the same position.

Moreover, The Boids can have 3 type of speed, which can be slower, faster and same. And by S_d we can represent the speeding factor of Boids. Where, S_d = Boids speed difference and shown as,

S_d (Slower, Same, Faster)

2.3 Autonomous Boids

Hartman and Benes [9] added 'Leader' and 'Eccentricity' with Reynolds [4] Boids Algorithm in their paper and made the flock's movement more efficient. Here, the paper first contains the Boids algorithm, which he first introduced. They used the Separation, Cohesion, and Alignment, and with these, they calculated the center of density and the position vector. By combining the steer function with the center of density and position vector, we can find the initial state and calculate the stable position of the Boids in a particular environment.

Leadership

As per the paper, the Boids without external force (such as a leader) will have an optimal configuration in 3-dimensional space, stay still, oscillate, or form some visual patterns. The local flock-mates (flocks those remains beyond a particular flock radius) are considered leaders evaluated in the visual representation. However, this does not work when the flock becomes too dense. As the leader becomes surrounded by other Boids, the force driving the leader to its destination fails to push through the numerous Boids in front of it.

Eccentricity

It is mainly used for evaluating whether the Boids are inside or outside the flock. At first, we will need to measure the composition of the flock-mates, and then we calculate the eccentricity. Eccentricity is a scalar value, which is the difference between the Boids and the center of the density. In this case, flocks moving densely within the center, so, eccentricity is calculated as the distance of the Boids from the center of the flock and it is denoted by X_i.

Let a Boids b_i in the position of p_i with the state that is the composition of the visible flock-mates V_i and another Boids b_j visible with the position p_j where m is the total number of Boids and e is the visibility range of the Boids.

$$\mathrm{X_i} = \sum_{\forall b_j \in V_i} \frac{p_j}{m \cdot e} \tag{2.11}$$

Here by the divisor e, the value of X_i is normalized. As a result, the X_i will have a value between 0 and 1. It identifies where the Boids are while the flock-mates surround it. At last, they have used global direction to gather the information of the Boids position.

2.4 Survey Paper On Swarm Intelligence

J Wang, G Beni [5] introduced the concept of "Swarm Intelligence" (SI) in their paper. After that, Keerthi, Vijaykumar, and Ashwini [15] published a paper about ants, bees, and birds' collective behavior and wrote a research paper on swarm intelligence that represents the collective behavior of decentralized, self-organized systems. The inspiration of SI originates from biological systems. On that note, ants are considered to be one of the best natural examples of the swarm. Some natural examples of SI include ant colonies, bird flocking, animal herding, bacterial growth, and the school of fish. Doctor Kennedy and E Berhart [6] introduced particle swarm optimization (PSO), which was developed from swarm intelligence.

Typically, SI systems consist of a population of simple agents that interact with one another and with their environment locally. We can use SI in our research as it is based on optimization algorithms and will use the principle to define Predator's hunting position and prey's obstacles avoiding capacity. As PSO focused on analyzing the condition's stability, it is known as a computational method that optimizes a problem by iteratively improving a candidate solution by measuring the quality.

PSO mainly concentrated on the math's basic theory where the particle can move stably by analyzing the condition's stability. Also, particle swarm topology is the research on the topology of a new pattern of particle swarm. Moreover, it combines with other intelligent optimization algorithms, which combines the PSO advantages with practical value with the other intelligent optimization algorithm. Finally, develop the algorithm's application area, which is essential since the PSO algorith PSO has been used widely to explore the developing area.

Algorithm 1 PSO

```
while The end condition is not satisfied do
    for all Particle do
        calculate the objective of the particle
        Update P_best if required
        Update G_best if required
    end for
    Update the inertia weight
    for all Particle do
        calculate the objective of the particle
        Update the velocity (W)
        Update the position (X)
    end for
end whilereturn G_prst as the best estimation of the global optimum
```

Typically, SI systems consist of a population of simple agents that interact with one another and with their environment locally. We can use SI in our research as it is based on optimization algorithms and will use the principle to define Predator's hunting position and prey's obstacles avoiding capacity. As PSO focused on analyzing the stability of the condition and it is known as a computational method. So. on each iteration, PSO modifies the current set of solutions with a new set of slightly modified candidate solutions. PSO uses a fitness function, using which it scores all the solutions, and by keeping the best set of solutions, it iteratively optimizes the solution/reaches its goal.

2.5 Autonomous Cooperative Flight Control for Airship Swarms

According to Artaxo et al. [19], UAVs(Unmanned Aerial Vehicles) is a less expensive, more flexible remotely driven AirCraft that is growing Rapidly in agriculture sectors overall environmental monitoring, inspection, forest fire detection, surveillance, convoy protection and search and rescue. Because of better resolution, the small design and lightweight UAVs can fly close to the ground. It can provide better mapping, monitoring, supporting, and surveillance applications than any satellite-based or full-scale manned aircraft due to these features.

The UAV airships mainly execute two tasks

1. Waypoint path following
2. Moving target tracking

The focus of this study is on two approaches for the design of autonomous cooperative flight controllers. These are,

1. Formation flight
2. Swarm intelligence strategies

Waypoint path following

Used for Surveillance the team of airships receives a set of target coordinates to inspect. They could move from one point to another or stay at a place from the targeted spot in a given radius. For overcoming the limitation of sophisticated mission's multiple UAVs can be used.

Moving target tracking

Used for tracking moving targets like livestock, wild animals, people, or other vehicles on the ground. It surpasses the constraints like unforeseen maneuvers and kinematic of tracking with the sensor readings fusion that comes from the other UAVs.

Let's talk about the two approaches:

Formation flight

This Approach is divided into three layers of communication and coordination of cooperative mission planning.

Layer 1: Task Allocation

Layer 2: Path Planning

Layer 3: Execute the commanded path

The Second layer path planning is mainly related to this approach as it helps the airship to stay coordinated and fixed on a desired distance and orientation angle. The lower Layer track the position and velocity of the moving target. The middle layer uses the State Feedback Kinematic Control (SFKC) approach and the lower-level Nonlinear Sliding Mode Control (SMC) has been adapted.

The Swarm Intelligence Strategies

The Swarm intelligence (SI) behavior has been generated by observing many species of animals and their behavioral patterns. It is a branch of computational intelligence that studies collective behavior emerging within self-organizing societies of agents.

1. **Flocking:** Flocking is a behavior of collective motion by a group of self-propelled entities, which is considered an emergent behavior arising from simple rules that are followed by individuals and do not involve any central coordination. This we have disused in chapter 1.

2. **Foraging:** Foraging is a behavior where animal search for foods in a wild life. Foraging has a direct affect on animal's ability to survive in the wild life and reproduce.

3. **Collaborative Manipulation:** Collaborative manipulation is a behaviour, where entities response to their surrounding environmental changes. In this process the entities continuously adapt new behavior and enable customization. In field of robotics this behaviour is used a lot.

4. **Aggregation:** Animals tendency to group together is known as Aggregation behaviour. This gathering depends on the individual fitness of the animals. This behaviour is more like moving in a group, if the movement is interrupted by other entities the group will continue to move without changing their rules.

The Swarm Robotics

The swarm robotics approach is used to track different species of animals depending on their behaviors. The major characteristics of the algorithm help it to analyze and mimic these behaviors. The robots that create the swarm to communicate with each other

The main characteristics are:

1. **Robustness:** Because of decentralized distributed architecture, the swarms can accomplish the mission despite some individual drones may fail.

2. **Flexibility:** Adaptability to different environments helps them to achieve greater possibilities.

3. **Stability:** Due to a decentralized distributed system each member of the swarm has an independent controller. Therefore, no matter how many members join the swarm the technical complexity will always be less.

4. **Emergence:** As each swarm member works as a group. Simple tasks followed by the individual leads to intelligent behavior.

2.6 A Review on Informed Search Algorithms for Video Games Pathfinding

Kapi, Sunar, and Zamri [23] stated in their paper about numerous modifications to enhance the execution of the informed search algorithm through four classified perspectives:

1. **Modification to the graph representation**

2. **Enhancement of heuristic function**

3. **Hybrid search algorithm**

4. **New data structure**

In the paper, they have shown several optimizations which describe the aspects of optimization.

Optimization by Changing Graph Representation

For representing environments, often graphs or maps are used. So, the first solution to optimize the pathfinding algorithm is to select the best graph representation. Three popular graph representations represent graphs in games: waypoint graphs, grid maps, and navigation meshes.

Optimization by Improving Heuristic Functions

As a critical field of AI, the heuristic search function has been used in game-play, agent control, and path-planning. Moreover, there is two general heuristics function used. However, to help reduce computation, the selection of the heuristic function acts as a substantial part of the overall pathfinding process.

Optimization by Changing the Data Structure

As we know, "abstraction" and "implementation" are two sides of any data structure. However, good area pathfinding is very important because it allows non-player characters (NPCs) to move to specific target positions in computer games by providing navigation for them. However, an algorithm can be ideal as it can speed up by enhancing from various perspectives.

2.7 On the Identification of the Convex Hull of a Finite Set of Points in the Plane

The notion of convex hull is useful in identifying the region that is occupied by a group of prey. Jarvis [2] provided the first algorithm for computing the convex hull of a set of points. He said that the convex hull is the smallest convex polygon covering all the points in a space set. In short, all the points are bounded within that particular polygon, which encloses all of the points. The vertices maximize the area while minimizing the circumference. Convex hull has various applications ranging from image recognition in robotics to finding an animal's home range in ethology. The paper mentions two ways of computing the convex hull of a set of points. The first one is Graham Scan [1], which he explained used to implement Jarvis March's algorithm. The Graham scan algorithm starts by looping over all the points to select the lowest Y coordinate. Then it sorts the points by the angle relative to this bottom-most point with respect to the horizontal axis. Here the angle would range from 0 to 100 degrees. Lastly, the algorithm looks over those points in sorted order, effectively scanning in the counter-clockwise direction. After that, each point is placed on a stack if it makes a line. He used the idea in his Jarvis March algorithm to create a convex hull of a finite set of points. In the convex hull, S is the finite set of points in the X-Y plane. So,

For $i = 1$ to n with n being the number of iterations

X_i is the X coordinate of point i

Y_i is the Y coordinate of point i

S_i=(X_i, Y_i) is the coordinate of the point i

To complete this algorithm, we need to go through 3 steps. The steps are below:

Step-1:

First we select a point whose Y coordinates is smaller than all points present in our point set and define this to be the origin. Then we will define it as S_O where $S_0 = (X_0, Y_0) \in S$. Here,

X_0 = Origin in X-axis

Y_0 = Origin in Y-axis

So the origin will be $S_0(X_0, Y_0)$ if,

$$Y_0 < \min(Y_i) \tag{2.12}$$

In Figure 2.1 we can see that there is a two-axis, X and Y. In the first step, we will mind the minimum position in Y-axis. Here, S_0 has a point (X_0,Y_0), where Y_0 is the smallest point of all the finite set of points. According to Jarvis [2], we need to find the smallest point in the first step.

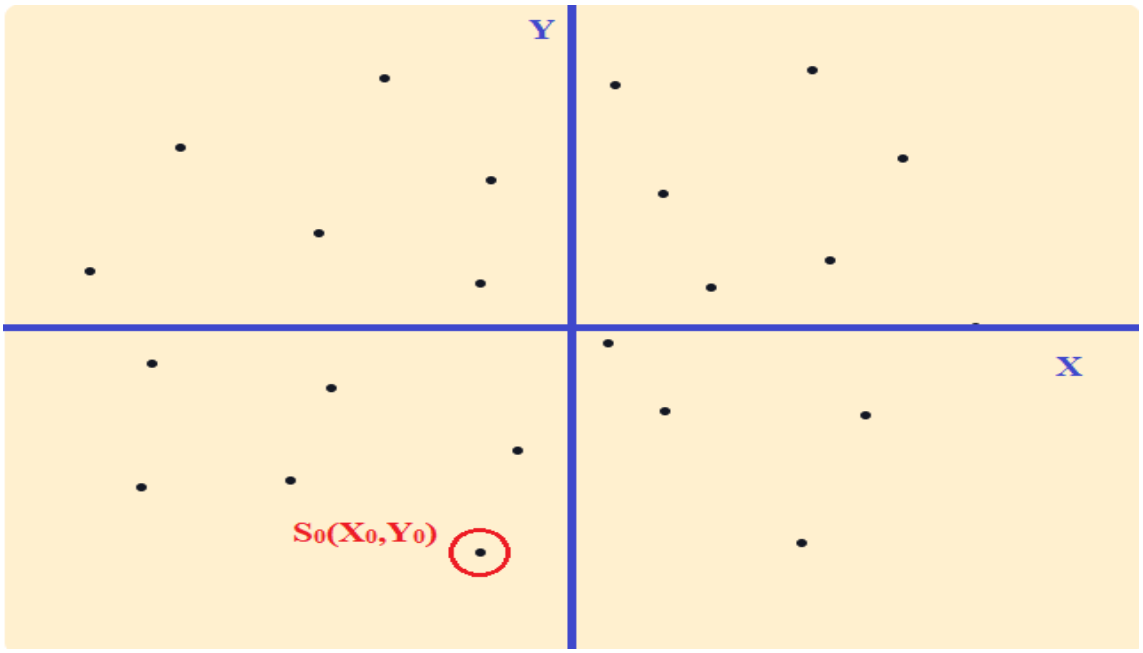


Figure 2.1: Finding origin

Step-2:

Second, we must find the smallest angle with the origin and the other points counter-clockwise. Say, $S_k(X_k, Y_k)$ is the point with which the origin $S_0(X_0, Y_0)$ creates the minimum angle θ_0 counter-clockwise .

Let,

For $i = 1$ to n with n being the number of iterations

θ_0 is the counter-clockwise created smallest angle from the $S_0(X_0, Y_0)$

θ_i is the counter-clockwise angle from $S_0(X_0, Y_0)$ to the point i.

Then the $S_k(X_k, Y_k)$ point will be create with the angle of,

$$\theta_0 < \min(\theta_i) \tag{2.13}$$

To find the following points for creating a convex hull, we need to find the smallest angle calculating counter-clockwise. In Figure 2.2 S_o (X_o,Y_o) is the origin. From this point, we will calculate the minimum angle counter-clockwise. In Figure 2.2 from $S_o(X_o, Y_o)$ we calculated θ_0 as a minimum angle and found the next point $S_k(X_k,Y_k)$.

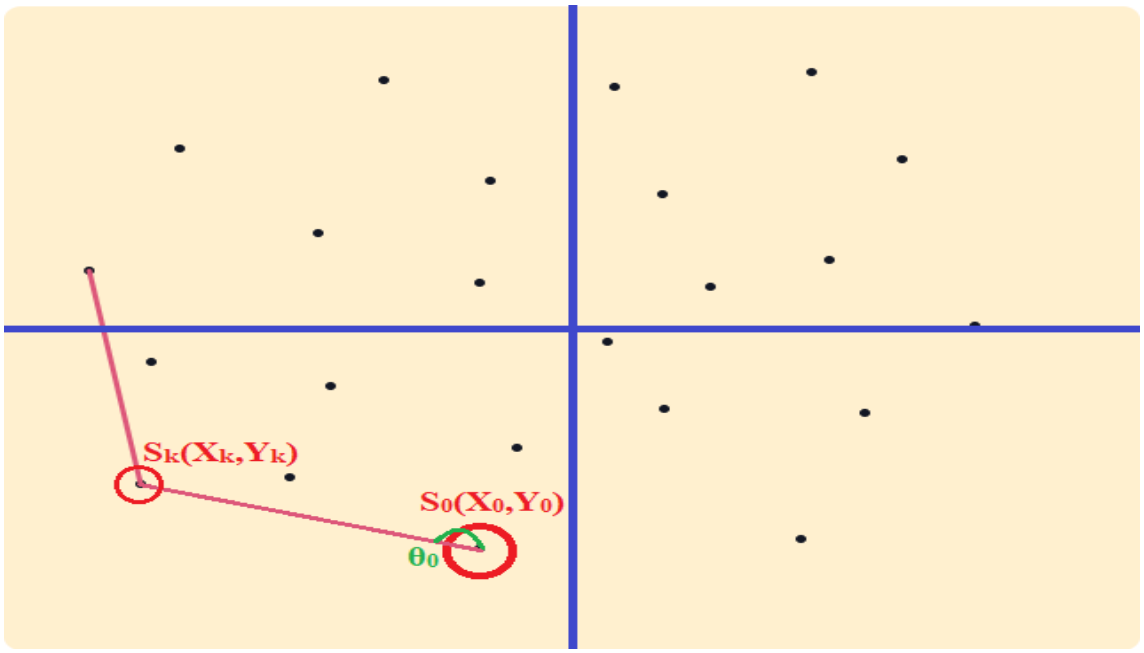


Figure 2.2: Finding the smallest angle

Step-3:

At the end, the origin will shift to the $S_k(X_k, Y_k)$ point and again will create a counter-clockwise angle with the next point and continue the step-2 and 3 until the first origin is found.

So,

$$S_o\left(X_o, Y_o\right) = S_k\left(X_k, Y_k\right) \tag{2.14}$$

In Figure 2.3, we can see previously calculated origin shifted to the next point, and from the new point, it again started to calculate the next point counter-clockwise. Moreover, the process continued.

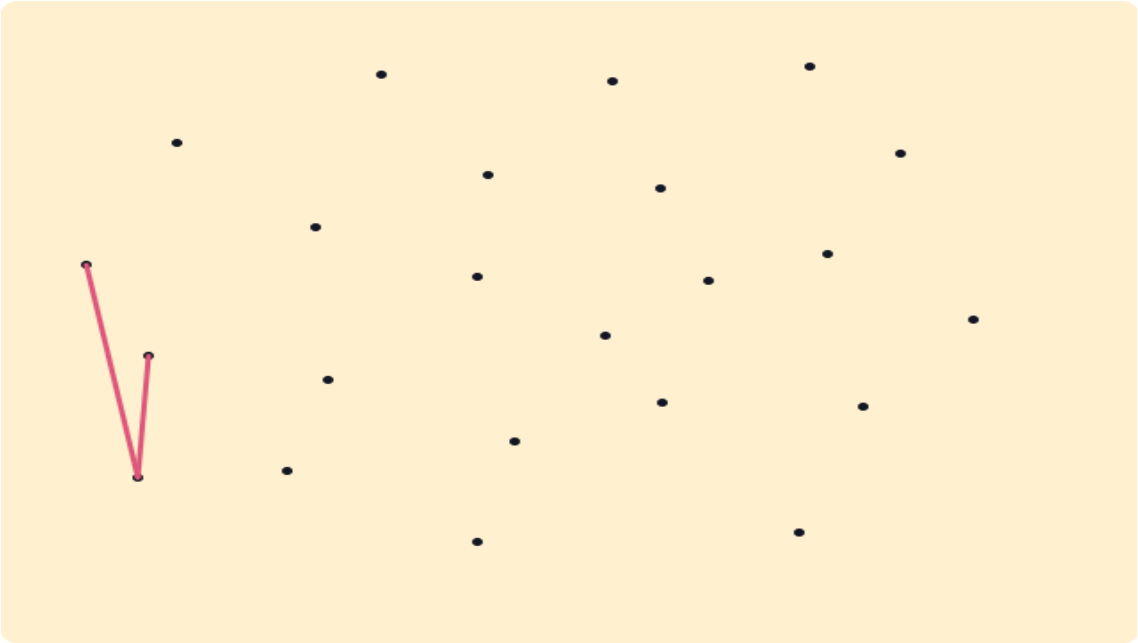


Figure 2.3: Finding the new origin

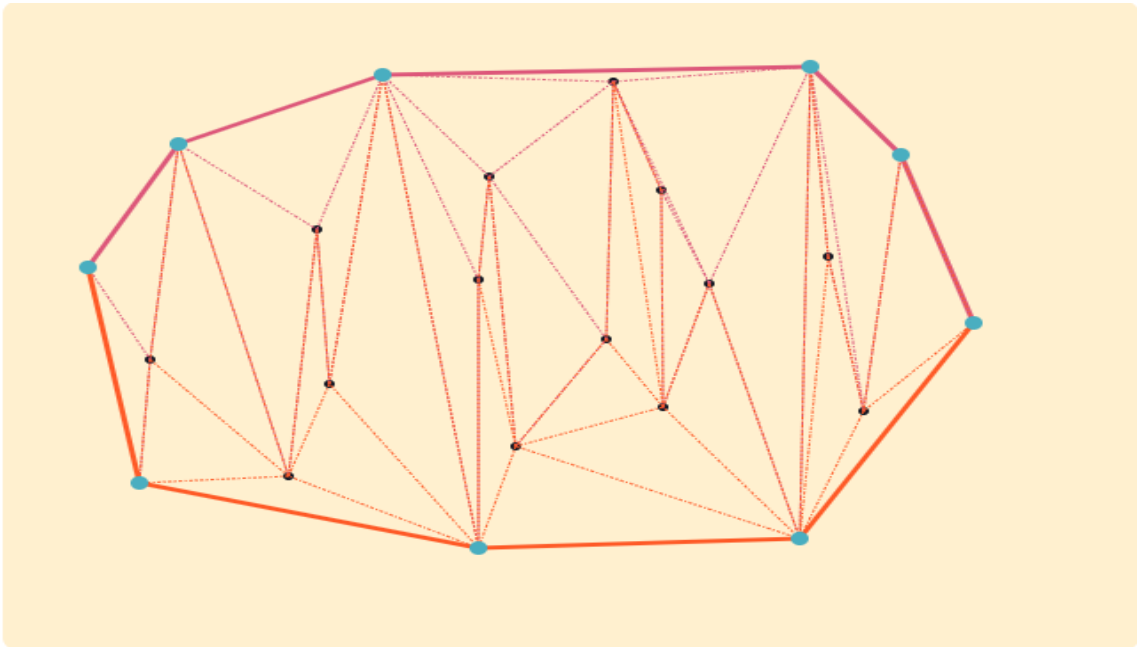


Figure 2.4: Finding the convex hull

In the end, the process continued until it reached the very first point from where the algorithm started to calculate the following points. However, at the end of the three steps, we will get an entire convex hull, including all the points, shown in Figure 2.4.

Chapter 3

Simulation of behavior of animals

Animals are models of complex systems. Therefore, representation should be used to understand their emerging individual, group and social behavior fully. Models can be physical, symbolic, mathematical, or computational, but they are always more superficial than the fauna they represent. In this thesis, we choose computer-generated simulation to represent the behavior of animals. To use models effectively, we need to understand their specific functions and limitations. In this thesis we observed various resources and garter ideas about the behaviour of the animals which we have discussed in the following sections.

3.1 The Idea of the Animals' Behavior

As per stated in our objective, our goal is to make this simulation as natural as possible. To achieve this, we have used a nature documentary of BBC, Yellowstone: Battle for Life [11], as our investigation resource. To simulate a natural phenomenon, we need to understand how the nature works. The documentary shows that:

- A deer herd always moves as a groups where each deer follows its neighbour's movements.
- Usually, the weak deer stays in the middle. As a result, the weaker deer are shielded by the stronger ones from predators from the outside.
- The movement of the deer is food and survival-oriented.
- Normally, deer herd moves toward grasslands and water streams for food. Also, they move in open grounds so that that they can see predators from distance.
- Deer herd also migrate from one place to another for seasonal changes.

On the other the properties of the wolves it shows are:

- When wolves are hunting, they always move toward the deer herd.
- They hunt in groups.
- Generally, they have a leader who decides the target for others by analyzing the speed and direction of movement.

- Predators work more intelligently; first, all the wolves try to disperse the deer herd, and the movement leader selects a target. Then the wolves move towards isolating the targeted deer.
- the chase ends with capturing the targeted deer.
- In case they fail, they do the whole process again, as these acts are for survival.

This ideas helped us to move froward for the implementation process. In the following section we talked about the algorithms which will help to implement the behaviour of deer herd.

3.2 Flocking Algorithm

We use Reynolds's [4] algorithm to implement the deers' flocking behavior. The three main parts of the algorithm, Cohesion, Separation and Alignment is shown below.

3.2.1 Cohesion Algorithm

Cohesion is a type of behavior where Boids try to move toward the average position of the whole group. Cohesion force makes the flocks stay together and do not let the flocks distract apart. Pseudocode Cohesion shows our algorithm for cohesion. Here, $BOID$ an array of all Boids and Boid B is used to iterate it. We store all B's of every neighbour in another array N. Then calculate the summation of all neighbour's positions and divide the summation by the size of array N, and that is how we get the average position of the neighbours. Now, the Boid should go toward the average position. We find out the direction of the average position by the difference between the average position and B's position (Figure 3.1). Finally, the boid B can accelerate toward the direction (Figure 3.2).

Algorithm 2 Cohesion

```
1:  procedure COHESION(ARRAY BOID)
2:      for all Boid B in array BOID do
3:          array N ← Boids in neighbours of B
4:          Sum ← 0
5:          for all Boid b_n in array N do
6:              Sum ← Sum + position of b_n
7:          end for
8:          Avg_pos ← Sum/|N|
9:          Direction d ← (Avg_pos − B_pos) / |Avg_pos − B_pos|
10:         Accelerate B toward d
11:     end for
12: end procedure
```

In the first Figure 3.1 we can see some small triangular shapes, some of which are blue and one is yellow. Moreover, an black circle can be seen around the yellow triangle. This yellow triangular shape indicates the Boids B and the black circle

represents the visual range of Boids B.The blue triangular shape indicates other Boids and which are inside the visual range those can be detected as the neighbours of the Boid B. Here, we can see some orange lines coming out of neighbours indicating towards the average position of the Boids in side the visual range. The green line indicates the difference between Boid B and the average position. In the next Figure 3.2 we can see the working process of cohesion, The Boid B moves to the average position.

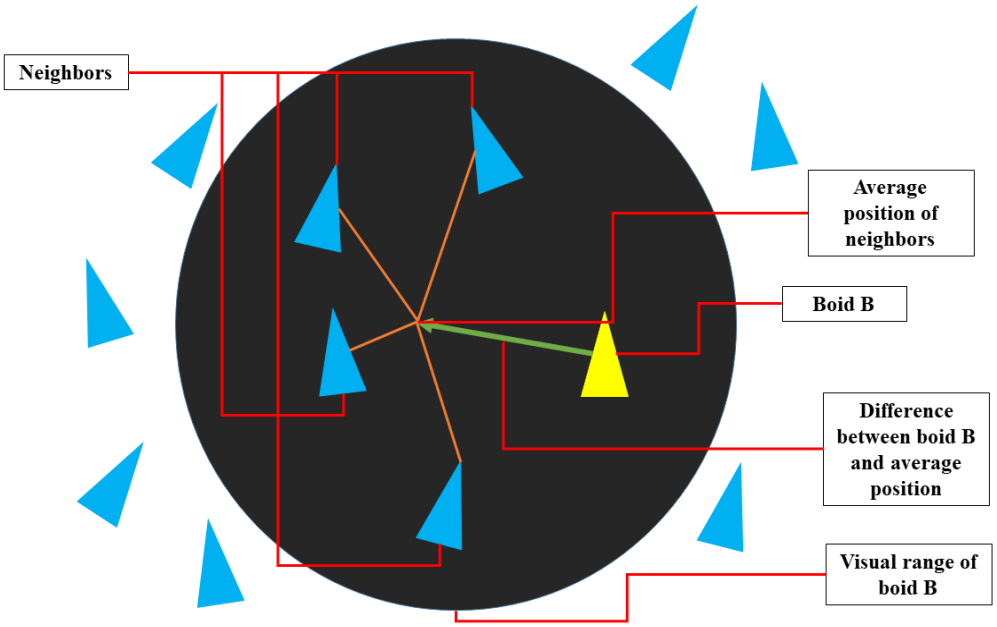


Figure 3.1: Average position detection

Normalizing vectors only changes the magnitude of the vector, whereas the direction will always be the same if we change the magnitude very frequently. In the alignment algorithm, the position of the Boids is changing very frequently. However, applying the normalization, all the position vectors of the Boids are converting into a unit vector. As a result, the direction of the Boids remains the same. Moreover, for the position vectors, we are normalizing because it is a direction vector. Its magnitude is not essential.

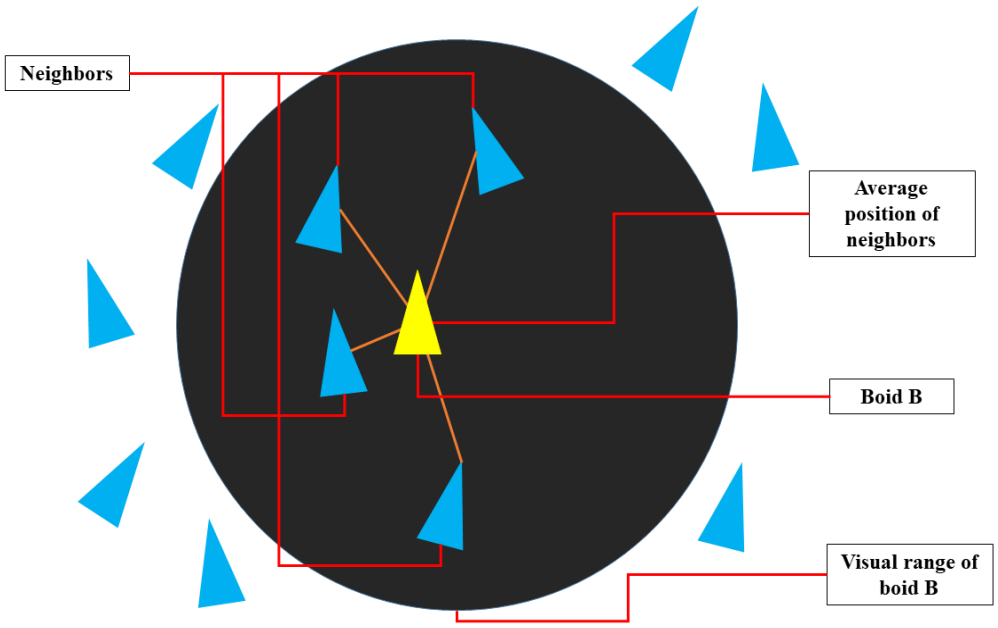


Figure 3.2: Boid moved to the average position

3.2.2 Separation Algorithm

Separation is a type of behavior where Boids try to move with the whole group at a certain distance to avoid a crowd. Because of separation, the flock maintaining a particular and avoid crowding local flock mates. Pseudocode 3 shows our algorithm for the separation algorithm where, $BOID$ an array of all Boids and Boid B is used to iterate it. We store all B's neighbours in another array N. Then we will keep the distance of all the neighbours of B in a variable D. Again, b_n are the characters of the neighbouring array N. Now, we will keep the distance value of the neighbouring Boids in b_n. After that, we will check if the value of D is greater than the given value of the $SeparationDistance$, and if it is true, the procedure will continue. Now, the Boids will go towards the direction which is the difference between the Boids position in B and the neighbours position in b_n, which is illustrated in Figure 3.3. Finally, the boid B can accelerate toward the direction of separation, which can be seen in Figure 3.4.

In the Figure 3.3 the yellow small triangular shape indicate the Boid B and some blue triangular shapes which indicate the neighbours of the Boid B. Moreover, an black circle can be seen around the yellow triangle. The black circle represents the visual range of Boid B. Here, we can see some orange lines coming out of neighbours indicating difference of position of neighbours from Boid B. The green line indicates that the difference between Boid B and neighbour is less then required separation distance. In Figure 3.4 we can see the working process of separation, The Boid B moves from its position to make the difference between its neighbours and itself greater or equal to the separation distance .

Algorithm 3 Separation

```
1:  procedure SEPARATION(ARRAY BOID, SeparationDistance)
2:      for all Boid B in array BOID do
3:          array N ← Boids in neighbours of B
4:          D ← Distance of boids in the neighbour of B
5:          for all Boid b_n in array N do
6:              D ← Distance of boid b_n in N
7:              if D ≥ Separation Distance then
8:                  Continue
9:              end if
10:             Direction ← (b_npos − B_pos) / |b_npos − B_pos|
11:             Accelerate B toward the Direction
12:         end for
13:     end for
14: end procedure
```

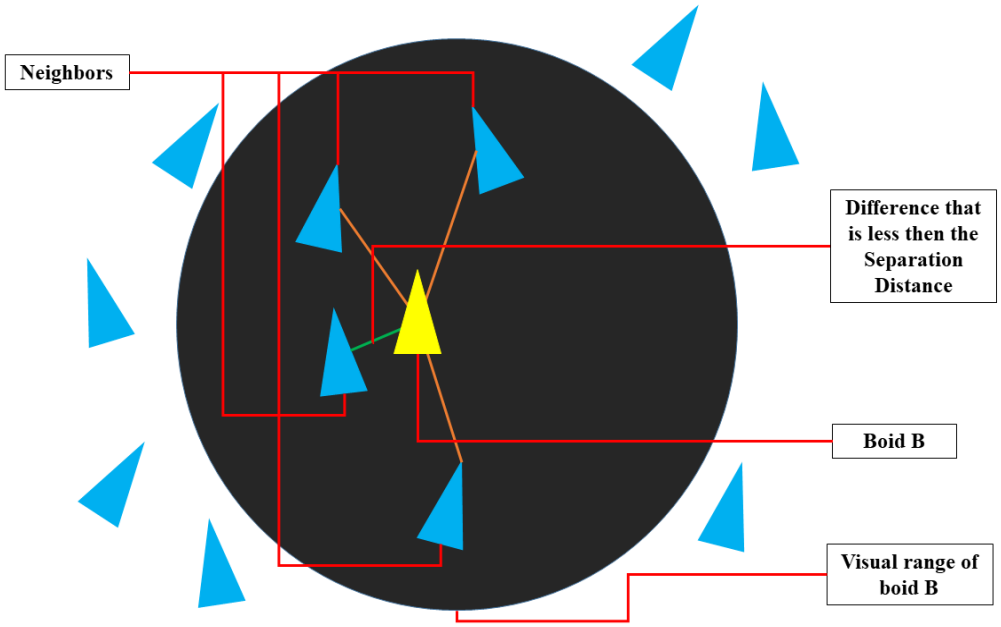


Figure 3.3: Distance detection

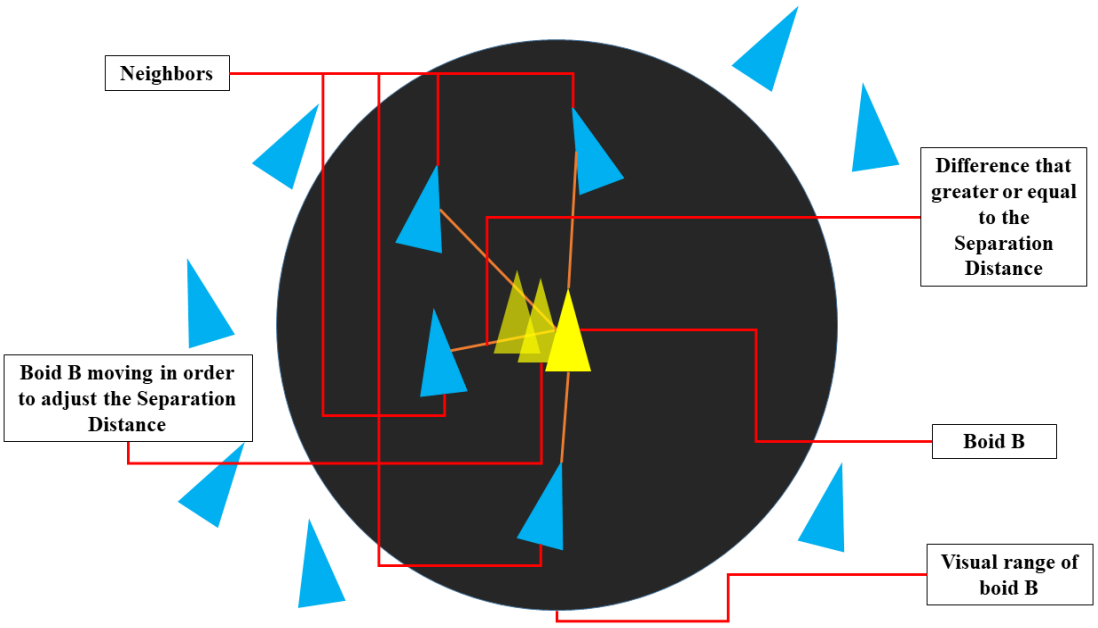


Figure 3.4: Boid moved to maintain Separation distance

3.2.3 Alignment Algorithm

Alignment is a behavior that makes a boid move in the direction that the whole group is moving. It ensures that overall all the boids is the group are moving in the same direction. This process is described in Pseudocode 4. The first loop iterates over all the boids. Then for each of these boids, another loop iterates to get the velocity of its neighbours in array N. After that, the summation of all neighbours' velocity is stored into $Avg_{velocity}$. Then we find the average velocity by dividing the $Avg_{velocity}$ by the size of array N. End of all, B is accelerated toward the average velocity. That is how all the Boids in a particular range aligned to a single direction, which helps show the flocking behavior.

Algorithm 4 Alignment

```
1:  procedure ALIGNMENT(ARRAY BOID)
2:      for all Boid B in array BOID do
3:          array N ← Boids in neighbours of B
4:          for all Boid b_n in array N do
5:              Avg_velocity ← Avg_velocity + velocity of b_n
6:          end for
7:          Avg_velocity ← Avg_velocity / size of N
8:          Accelerate B towards Avg_velocity
9:      end for
10: end procedure
```

In the Figure 3.5 we can see yellow small triangular shapes which indicates the Boid B and some blue triangular shapes which indicates the neighbours of the Boid B. Moreover, a black circle can be seen around the yellow triangle. The black circle represents the visual range of Boid B. Here, we can see some green arrow in top

of the Boids indicating direction of movement of neighbours of Boids B. The white arrow in top of the Boid B indicates the Boid's current direction of movement and the green arrow in top of the Boid B indicates the average direction of Boids in the range. In the next Figure 3.6 we can see the working process of alignment. Here, the Boid B changes it's moving direction toward the average direction. Thus the alignment algorithm works.

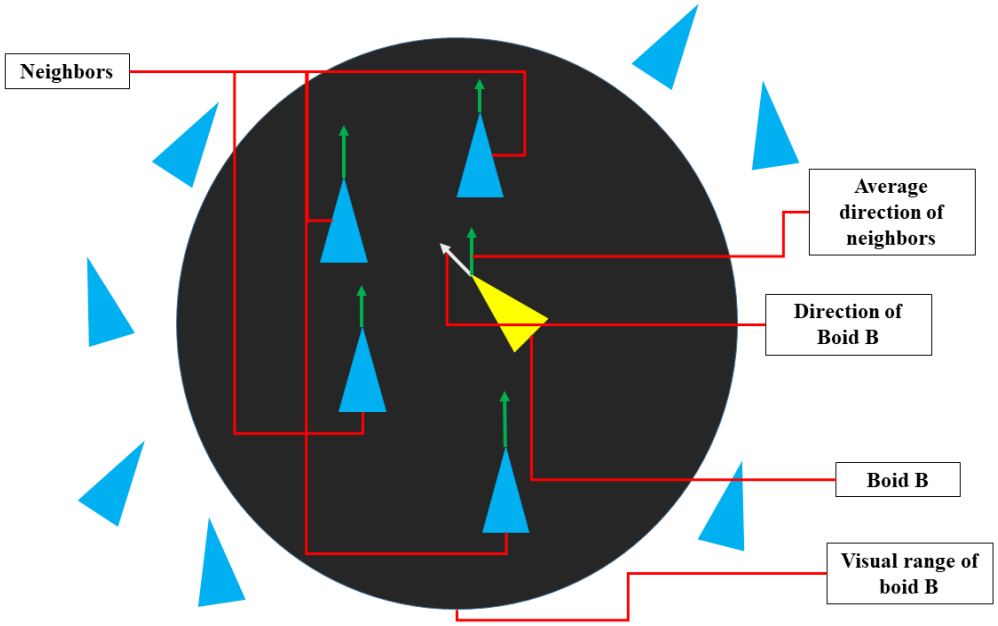


Figure 3.5: Detecting average direction of neighbours

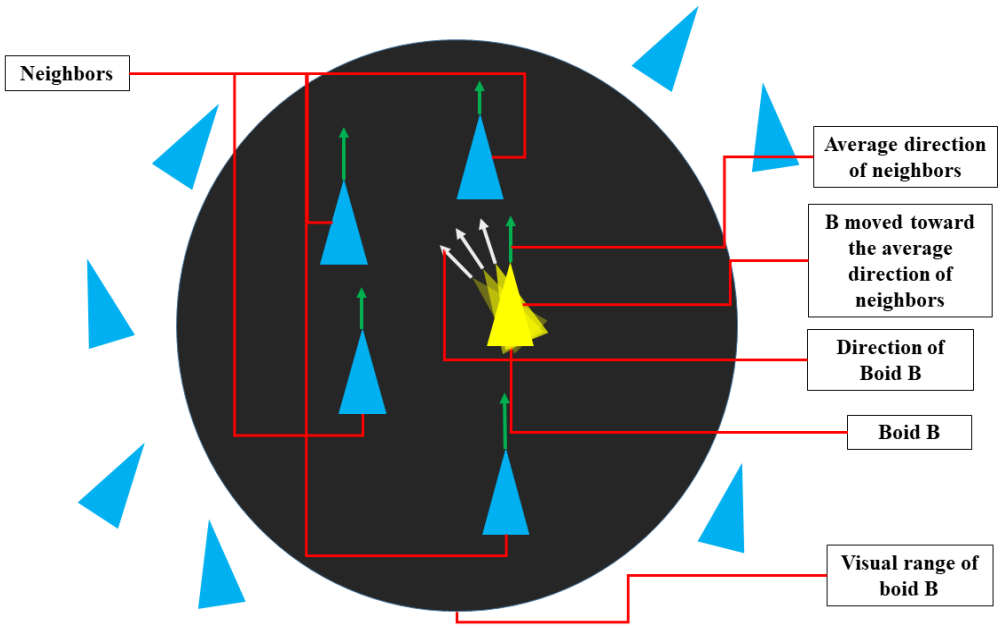


Figure 3.6: Boid moves toward the average direction

3.3 The Problem

As humans, we want to learn about nature and the surroundings that we are living in. However, if we seek information from natural life, we may hamper the natural lifestyle of wildlife and the beauty of nature. Also, some unusual behavior of nature is infrequent. One needs to be in a particular place at a particular time to observe this phenomenon, making it hard to visualize the whole process and even impossible to analyze times.

This motivated us to simulate the scenario artificially, and with that comes the problems. The first problem is selecting a valid application that can support the facilities needed for simulating the behaviors by implementing the algorithms and other logical instructions. Then we had to simulate the deer herd using Reynolds [4] algorithm, which was modified to work perfectly for our scenario. Another problem was to implement predator characteristics. Predators had to be lead by their leaders. Also, the leader needed to select the target and send the pieces of information about the target to other predators. After getting the information, the predators would create formation and tried to isolate the target. Implementing these types of intelligence for every predator was one of the toughest challenges for us.

The last problem we faced was to move the deers with flocking behavior in a path and the termination of the target. Also, the graphical representation of this whole process in a single frame was a big challenge. All this problems took us to the part of solutions which we have discussed in the next section.

3.4 Details of Solution

Here we describe the solutions to all the problems we faced while creating our simulation. The first problem was to select a valid and feasible application. For simulation, we first selected three applications as candidates. These are Ogre 3D, Unity, and Godot. From the perspective of availability for everyone, we wanted to use Ogre 3D, but it was hard to find resources and supporting works on Ogre. So, we shifted to Unity Game Engine, which gives us a good and smooth simulation, also resources were available. However, implementing code and marge it with the entities was one of the difficult problems. Finally, we started to work with Godot Game Engine. Unity can be stated as a better engine in terms of the quality and complexity of the games. On the other hand, Godot is more helpful for beginner developers for its easy ways of representation. Moreover, our target was to make the simulation 2D, for which Godot stands up as a great option.

The next problem was to implement a flocking algorithm for our scenario. We added some extra characteristics to the existing algorithm. One is for calculating neighbours, by which all the boids can detect and calculate Distance between their neighbours and take actions according to that. We also implemented predator-escaping characteristics for boids, by which boids maintain a distance from the predator and try to escape to a safer place. Moreover, we added a color-changing feature for boids, from which we can understand the density or flock-mates and specify the targeted boid.

One of the most critical tasks of our thesis was implementing multiple predators. At first, we add a single predator and develop its characteristics. One of the characteristics is chasing the boids. In the beginning, we set the predator to go for

the dense area where flocks are gathers, then gradually we change it into a specific boid. After that, we add more predators and the leader of them. The task for the leader is to define the target and transfer the target's data to other predators. Then all the predator moves in order to isolate the target from the group. Sometimes they from some formation to achieve this goal, and after isolating the target, they terminate that.

In the end, we make all of the processes run at a time and in a single frame by using the game engine.

Chapter 4

Algorithms

Algorithm is in the heart of any program. For expressing algorithms we often use structured English that is understandable and concise for representing important features unambiguously, whereas programs themselves might be too cumbersome to grasp the essence. In this thesis, we wanted to add many characteristics for the prey and predators. To implement these characteristics, we develop some ideas for both prey and predators affecting the hunting process. Some of the ideas are: preys escaping characteristics from predators, preys neighbour detecting method, predators states for movement, target selecting capability of the leader of predators, predators characteristics of chasing prey, predators formation creating methods for isolating prey. All these ideas are shown below in algorithmic form and described with examples and figures.

4.1 Predator Escaping Algorithm

One of the main focuses of our thesis is to simulate the prey's behavior against the predator. The Pseudocode 5 shows our algorithm of escaping predator for prey's. According to the real-life scenario, a prey tries to escape the predators when they are close enough. This minimum Distance between prey and predator can be defined by setting a safety range, S, which we can see in Figure 4.1. When a Boid, B detects a predator in its safety range, S, it changes its direction of moving opposite the predator's position and increases its moving speed that represented in Figure 4.2.

Algorithm 5 Escape Predator

```
1: procedure ESCAPE PREDATOR(ARRAY BOID, SafetyRange)
2:     for all Boid B in array BOID do
3:         D ← Distance between boid and predator
4:         S ← Safety Range
5:         if D < Separation Distance then
6:             Q ← difference of position between boid and predator
7:             Accelerate boid according to Q in the opposite direction
8:         end if
9:     end for
10: end procedure
```

In the Figure 4.1 the yellow triangle represents Boid B and the black circle

around it is the visual range of Boid B. Here, the red triangle indicates predator and the orange colour circle around predator, represents the visual range of predator. In Figure 4.1 the minimum safety range can be seen clearly.

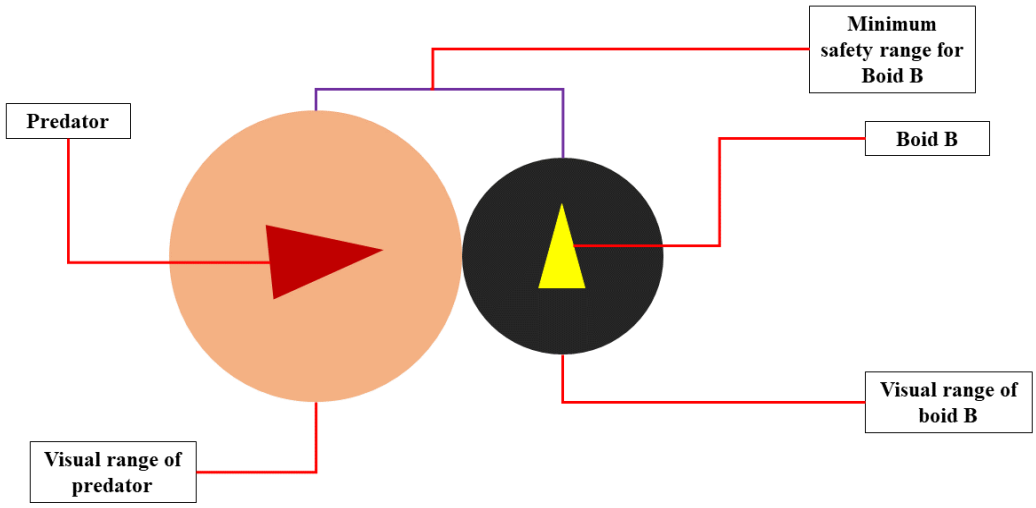


Figure 4.1: Visual safety range of prey and predator

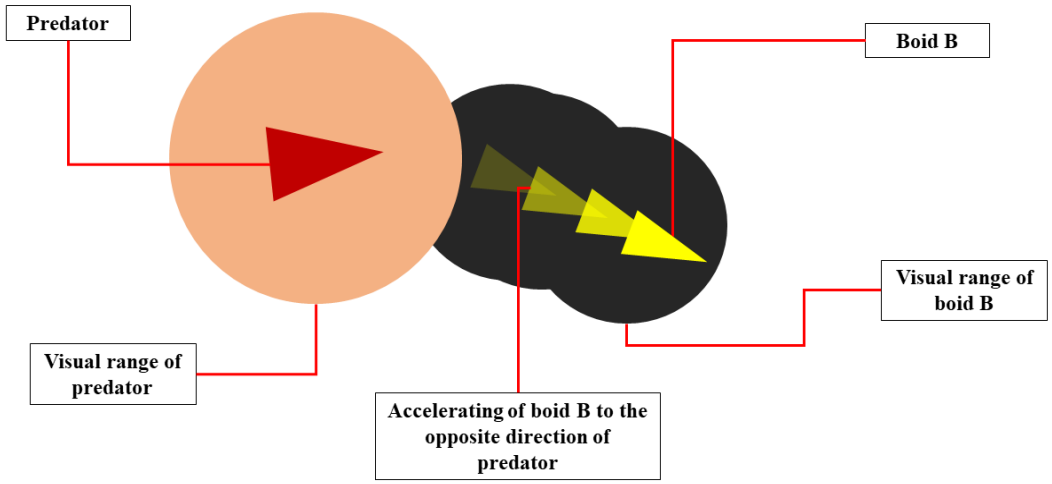


Figure 4.2: Prey trying to escape predator after safety range been compromised

When the Boid B crosses the minimum safety range which can be seen in the Figure 4.2, Boid B accelerates towards the opposite direction of the predator's position. Yellow triangle which are fading shows the continues change of position of Boid B.

4.2 Neighbour Detecting Algorithm

In this thesis preys are implemented using the Boids flocking algorithm. To execute flocking behavior first the prays need to identify their neighbour. Every prey has it's own range, in which other preys are detected as that particular prey's neighbours. In the Pseudocode 6 the array of Boids is iterated and every Boid is denoted by B_i. Inside that loop another array of Boid is iterated and the next Boid is $B_i + 1$ selected. Then the distance of position between B_i and B_i+1 is stored into D. After that the procedure check if the value of D is less then or equal to the $VisualRange$. If yes then, then B_i and $B_i + 1$ is marked as neighbour of each other. Also the distance of B_i and distance of $B_i + 1$ is set to the value of D.

Algorithm 6 Detect neighbour

```
1: procedure DETECT NEIGHBOUR(ARRAY BOID, VisualRange)
2:     for all Boid B_i in array BOID do
3:         for all Boid B_i + 1 in array BOID do
4:             D ← distance of position between B_i and B_i + 1
5:             if D ≤ VisualRange then
6:                 The neighbour of B_i ← B_i + 1
7:                 The neighbour of B_i + 1 ← B_i
8:                 distance of B_i ← D
9:                 distance of B_i + 1 ← D
10:            end if
11:        end for
12:    end for
13: end procedure
```

In Figure 4.3 yellow triangle represents the Boid B and the blue triangles represents the nearby Boids which are denoted by a,b,c,d,e,f,g,h,i. The black cycle around the Boid B indicates the visual range for neighbours of Boid B. Also an array N can be seen which is array of detected neighbour of Boid B. Again another array $N(a)$ is here which is array of detected neighbour of Boid a. Thus, all other arrays of detected neighbour for b,c,d,e,f,g,h,i is shown as dotted lines. As in this sate no neighbour has been detected, all the arrays are empty.

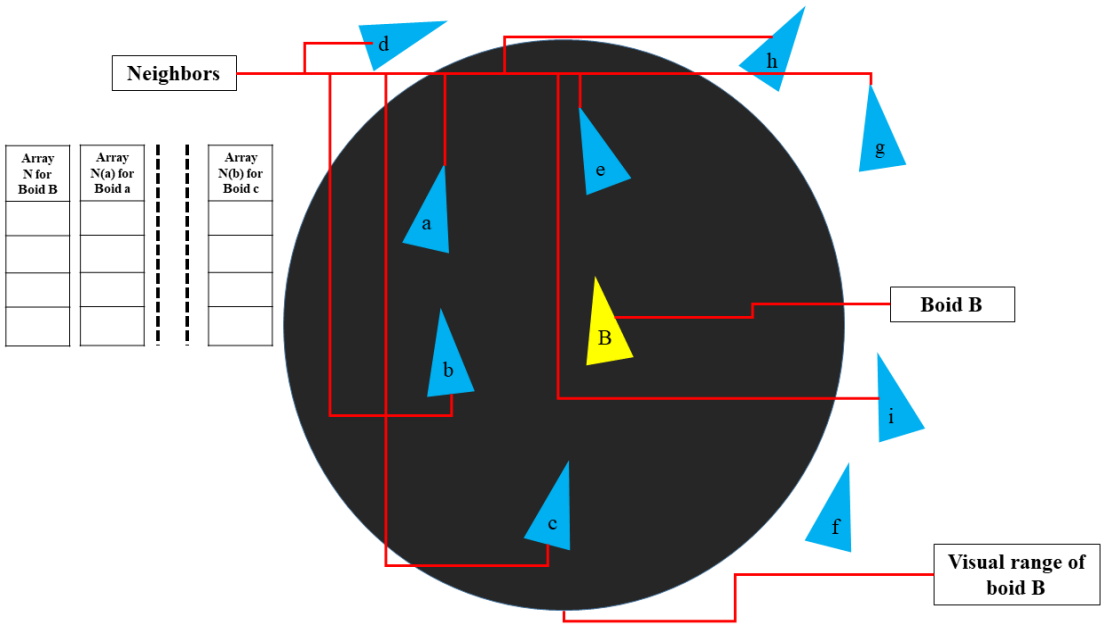


Figure 4.3: Boids before detected as neighbours of boid B and empty array

In Figure 4.4 yellow triangle represents the Boid B and black cycle around the Boid B indicates the visual range for neighbours of Boid B. The blue triangles represents the nearby Boids which are denoted by a,b,c,d,e,f,g,h,i. Here, inside the visual range for neighbours, Boid a,b,c,e has been detected as neighbour of Boid B. Also, the Boid B has become neighbour of Boid a,b,c,e. So, the array N which is array of detected neighbour of Boid B is filled with Boids a,b,c,e. Again another array $N(a)$ which is array of detected neighbour of Boid a is filled with Boid b,e and others. Thus, all other arrays of detected neighbour for Boids are filled with their neighbour Boids.

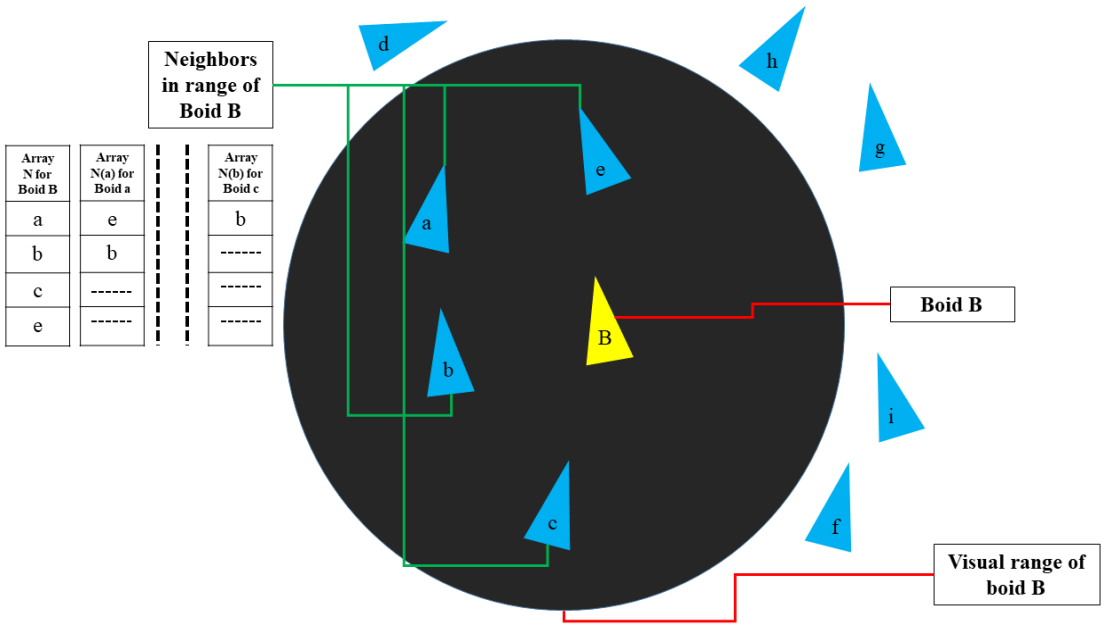


Figure 4.4: Boids after detected as neighbours of boid B and placed in array

4.3 Predator's States

In this thesis, two states for the predator, "Active" and "Inactive" states, have been implemented. In the inactive state, the predators have two types of activity. The leader of the predators stays inactive until it goes inside a specific range around the herd of prey and moves toward the herd of prey with its minimum speed. After entering into the range, the leader selects a target and goes into the active state. Other predators keep inactive until the leader gets active. In the inactive state, other predators move with their minimum speed and go towards the herd of prey to scatter them. When a flag token signals them that the leader is active, they turn active too. After being active, the predators move toward the target that their leader has selected. The predators then make a formation to isolate the target. After successfully isolating the target, they terminate it. If the predator lost the target or moved beyond the specific activation range, they got inactive in this process and again into an active state if the conditions were fulfilled. Also, when they are in an active state, they move at their maximum speed.

There are essential tasks that the leader and predators need to complete in these states, which we have discussed in this section.

4.3.1 Algorithm for Target Selection

Predators one of the most impotent work is to select target wisely. Target selection must be done by analysing preys action, position and movement. As, predator works in group, the responsibility of selecting target is upon the leader. After the target has been selected other members of the group try catch that particular target. This target selection process can be describe with Pseudocode 7.

Algorithm 7 Target Selection

```
1: procedure TargetSelection(array BOID)
2:     for all Boid B in array BOID do
3:         array S ← acceleration speed of B
4:         array D ← distance of Boid B from the predator
5:     end for
6:     Min_S ← minimum value in array S
7:     Min_D ← minimum value in array D
8:     for all Boid B in array BOID do
9:         if acceleration of B ≤ Min_S & distance of B from the predator ≤ Min_D
   then
10:            Target ← B
11:        end if
12:    end for
13:    Return Target
14: end procedure
```

In the Pseudocode at first the array of Boid is iterated and each Boid is denoted as B. Then in an array S the value of all B's acceleration is been stored and in another array D the value of all B's distance from the predator is stored. After that, the minimum value in array S is stored into Min_S and the minimum value in array D is stored into Min_D. Then again, the array of Boids iterates and checks if the acceleration of B is less then Min_S and the distance of B from the predator is less then Min_D. If yes then Boid B is selected as $Target$. Finally, the procedure returns the $Target$.

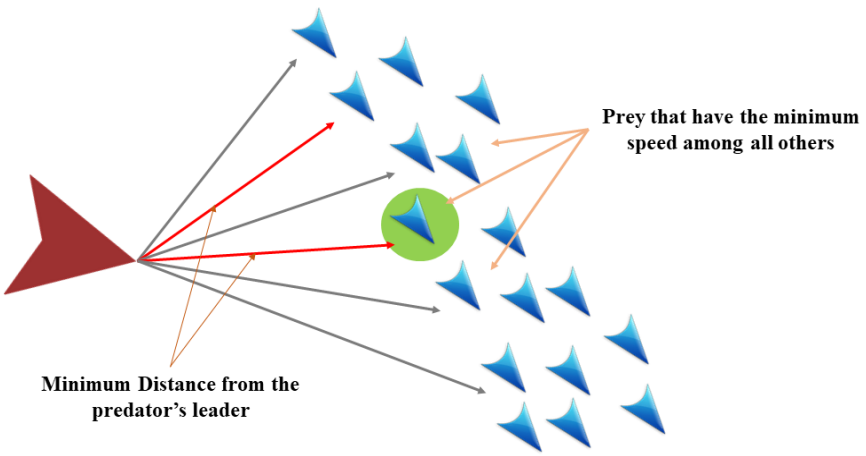


Figure 4.5: Predator selecting target by analysing their speed and distance

In the Figure 4.5 the target selecting process can be seen. Here, the red colored arrowhead is representing the leader of the predators. The blue arrowheads are

considered as the prey. The arrows from the leader shows the distance between the leader and the prey. The red arrow indicates the minimum distance from the leader to the prey. Here, we can see 2 prey has minimum distance from the leader. Again, the yellow arrows shows those prey who has the minimum speed. Three prey can be seen here with minimum speed. Now, the leader selects that prey as target which has both minimum speed and minimum distance. In the Figure 4.5 the prey inside a green circle represents as the target.

4.3.2 Formation Creation of Predators

In order to terminate the target, at first, the predator needs to isolate it. This isolation process starts with the formation creation of predators. Convex hull algorithm could be used to implement these characteristics of predators. In this thesis, we have studied the concept of the convex hull, where the entity can find out the outer line or the endpoint flocks of a flock group. Then by closing down, that outer line would help to format a positioning system for the predators. By maintaining that position, the predators could isolate the target. Also, cohesion and separation behavior would emerge within the predators. Nevertheless, the lack of time and complex implementing orientation of the algorithm led us to follow an alternative way. In an alternative way, the acceleration speed of the predator works as a parameter for the formation. Different predator speeds allow a chain-like formation, and the target chasing algorithm 8 is responsible for going toward the target. Combining all these, a half-circular formation is created. This formation keeps the prey away from its herd, and gradually it gates isolated.

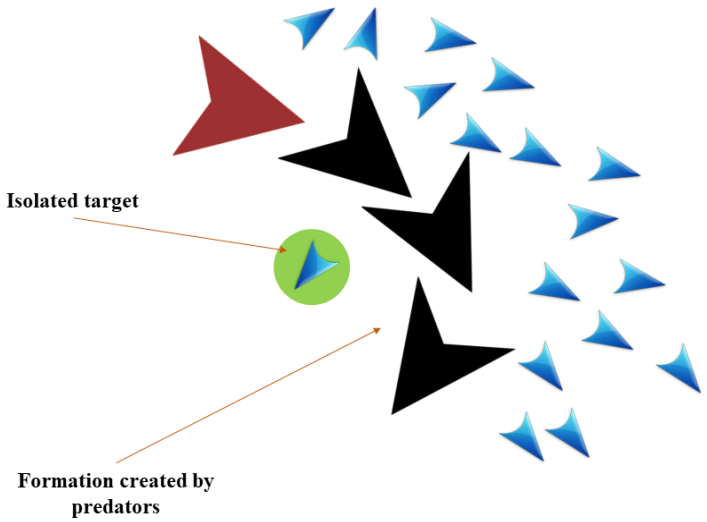


Figure 4.6: Predator creating formation to isolate the target

In the Figure 4.6, we can see predators, indicated by the back arrowheads and their leader with the red arrowhead. Here, the blue arrowhead within the green circle is the target. The predators and their leader made a half-circular formation,

not allowing the target to go back to the herd. Thus, the target gets isolated, and terminating it becomes easier for the predators.

4.3.3 Algorithm for Chasing Target

To simulate the predators chasing their target, we added some chasing characteristics to the predators' behaviour. Here, chasing means following the target with allocated speed. When the predator start to chase a prey it will speed up to it's maximum speed and the predator do not stop until it reach the target position. This process of chasing can be described with the Pseudocode 8. Here, the procedure checks if the predator P has reached to the target position or not. If the predator P has not reached then the speed of P is increased to the maximum speed and accelerate P toward the target position.

Algorithm 8 Chasing Target

1: **procedure** CHASING($Target_{pos}$, $Maximum_{Speed}$)
2: **if** $Target_{pos} \neq$ the position of predator P **then**
3: speed of $P \leftarrow Maximum_{Speed}$
4: accelerate P toward the $Target_{pos}$
5: **end if**
6: **end procedure**

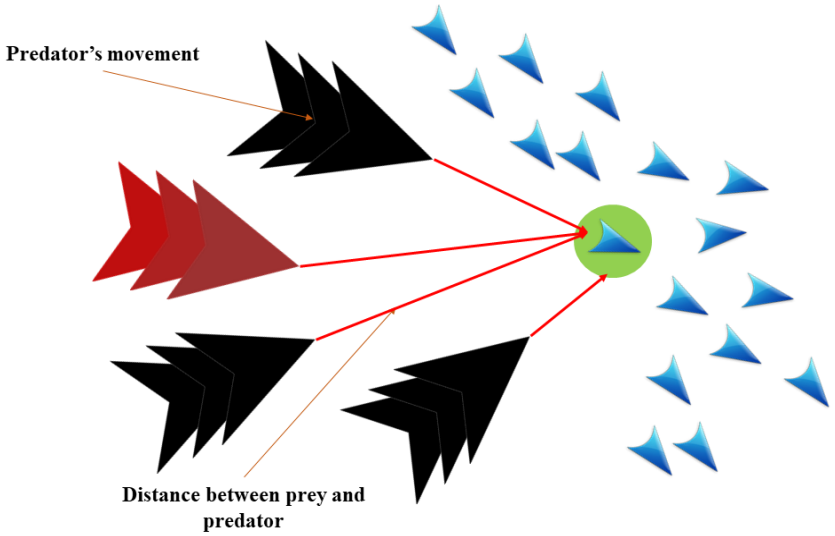


Figure 4.7: Predator chasing target

In the Figure 4.7, we can see predators, indicated by the back arrowheads and their leader with the red arrowhead. Here, the blue arrowhead within the green circle is the target. The read arrows from the predators toward the target represents the distance between predators and target. The predators and the leader is moving toward the target to terminate it. Predator chases the target until the difference of position becomes zero.

Chapter 5

Experimental Results

In this thesis, we implemented models of prey and predator into the simulation environment, and their characteristics with the help of algorithms stated above. The parameters we have used to bring the simulation into action are described in this section of our thesis. Moreover, after the implementation, we looked into the final simulation, which we consider as the result of this experiment. These results are being described and compared with the expectations. In some parts, the result perfectly satisfies our expectations. In some parts, it is good, in others, more work need to be done. The project we have created for this experiment has been uploaded in GitHub, Which anyone can access from https://github.com/MD-Mahbub-Ul-Haque/Computer-Generated-Artificial-Life-Model-Algorithm-of-an-Intelligent-Pack-Hunter-Hunting-Prey.git. In the uploaded file there are also some video links which contains recorded simulation of the project. In this chapter, we discuss these issues detailing both our achievements and shortfalls.

5.1 Parameters of Simulation

At first, parameters that have been used to built and run the simulation will be discussed. Godot has been used to make simulations; most of the parameters are Godot's built-in simulation parameters. Also, specific behaviors like escape predators, target selection, and others were implemented with the help of algorithms. In order to understand easily, these parameters have been discussed by dividing them into some main sections. All of these factors are been described bellow.

5.1.1 Environment of the game engine

There are several reasons behind using Godot in this thesis. The first reason is that Godot is free of cost. Where other game engines like Unity, Lumberyard needs to be licensed to be used. Moreover, Godot is open-sourced, allowing us to download a code and immediately edit, inspect, and compile it. It also allows redistributing any new version that someone makes. These are significant advantages for the student. Creating and editing a project in Godot is easier. Thorn and Alan [25] talked about the project creating process of Godot.To create a new Godot project or run an existing project, one needs to open the Godot engine first. To create a new project, one needs to click on the "New Project" option and name the project, along with the path for the project (Figure 5.1). Again, to open an existing project in the device,

one needs to go for the “Import” option and browse the “Project Path” from the device. After choosing the project path or the zip file, one must click the “Import Edit” button to open the project and edit it (Figure 5.2).

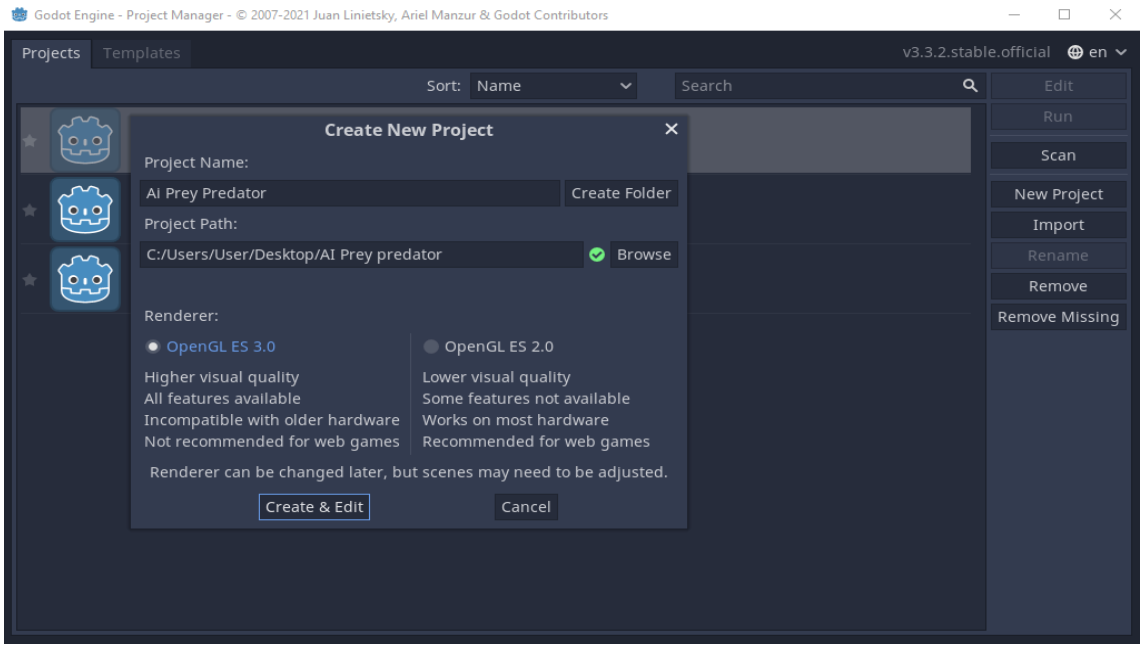


Figure 5.1: New project creating in Godot

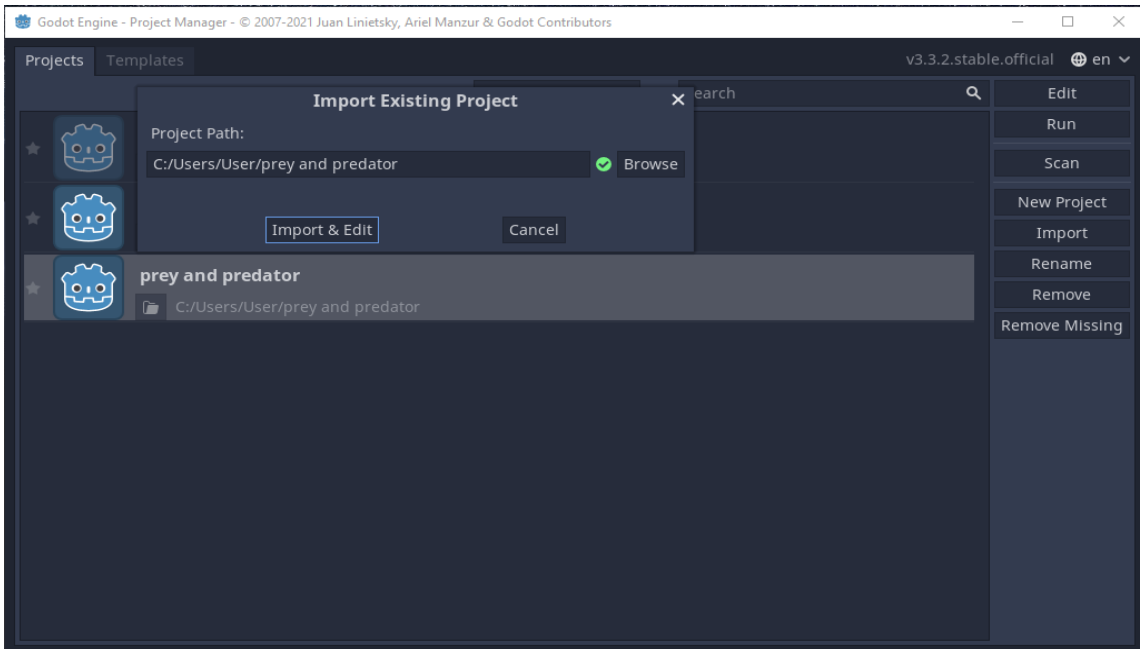


Figure 5.2: Importing project in Godot

The editor’s window has multiple options like Scene, Project, Debug, Editor, Help (Figure 5.3). Also, it allows choosing between 2D and 3D worksheets. From the

"Scene," one can make various characters for the project and add scrips (Figure 5.4). It also contains a play button that allows instant preview of the project.

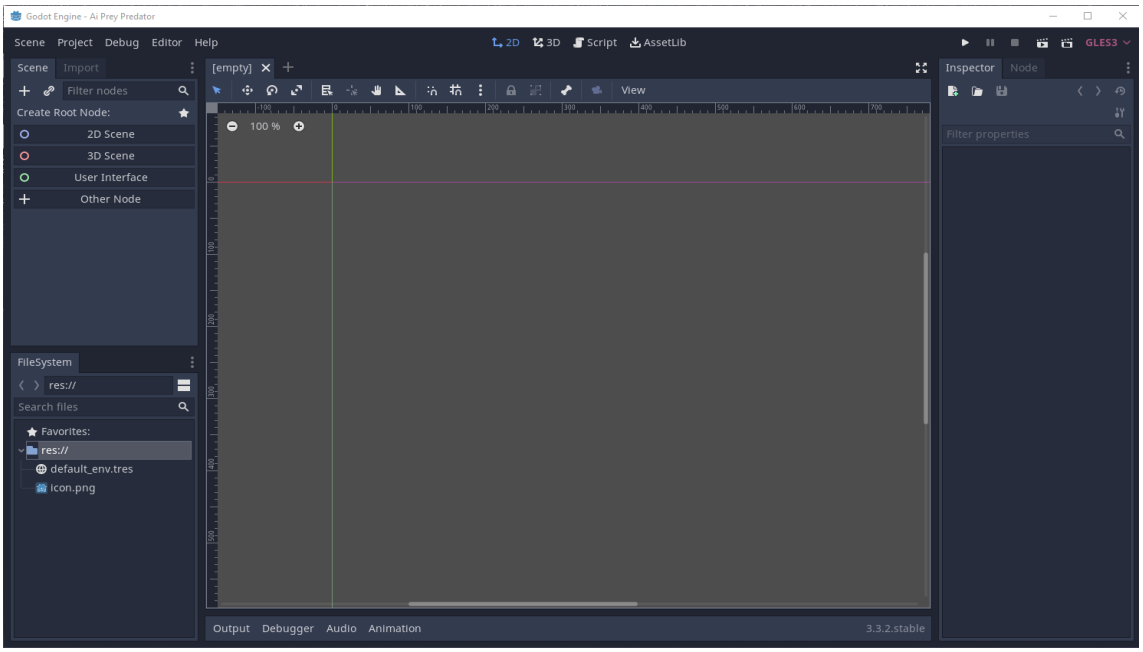


Figure 5.3: Environment of Godot

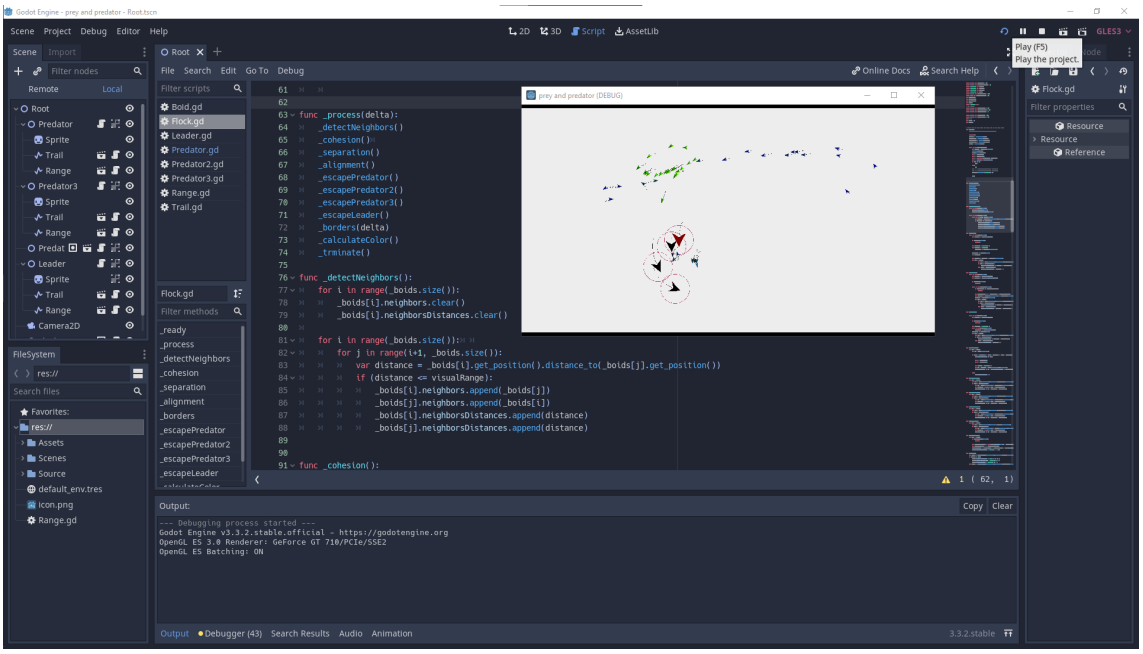


Figure 5.4: Godot project with script

Godot 3.3.2. the official version uses C as its scripting language. As C is a versatile, powerful, and easy-to-learn language, it helps to deliver a good run-time performance. All the features make Godot a user-friendly game developing environment for beginners.

5.1.2 Visualization factors

As our thesis is simulation-based, visualization of the whole process is more important. In the Godot environment, we have a camera view positioning feature (Figure 5.5). These features allow viewing from different angles and perspectives. It also helps to observe the process in a particular range by maintaining the ratios of the contents.

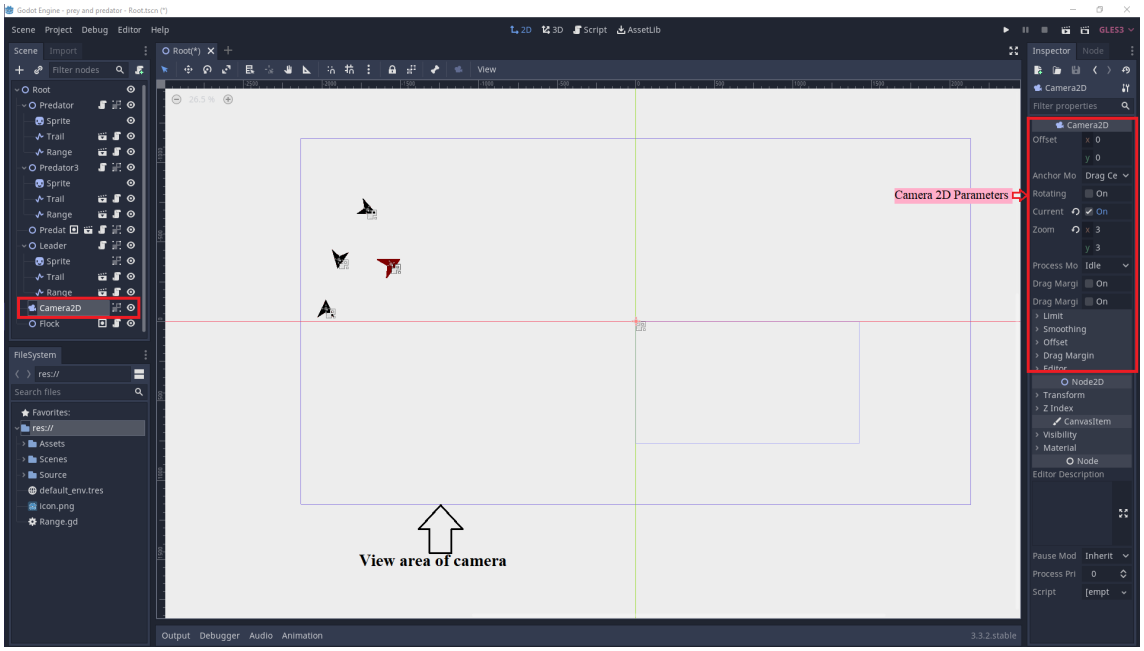


Figure 5.5: Camera positioning feature of Godot

To visualize the prey and predators, we used three types of sprites (Figure 5.6). These sprites are arrowhead-shaped, which helps to understand the heading of the prey and predators. The Figure 5.6 shows that the blue-colored sprite represents the prey, the black-colored sprite represents the predators, and the red-colored sprite represents the leader of the predators. Also, it helps to represent their other characteristics, like rotating, changing direction.

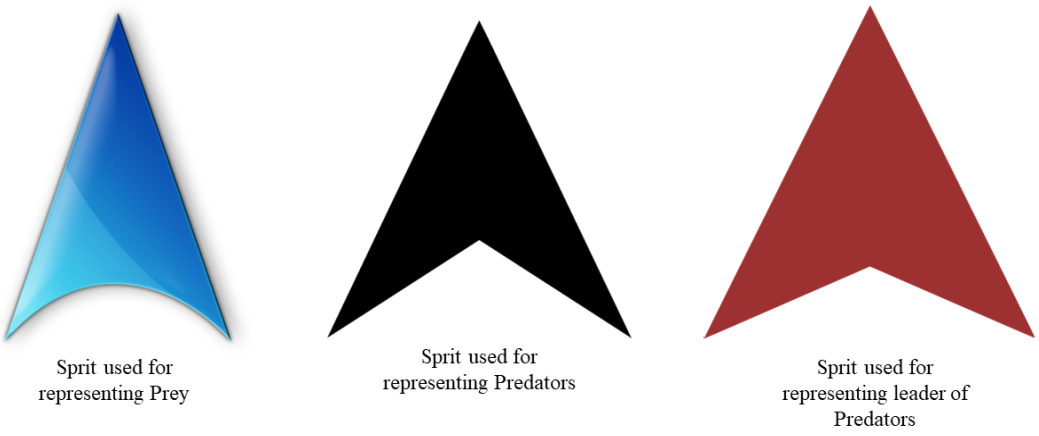


Figure 5.6: Sprites used for representing Prey, Predators and Leader of Predators

Colors have a significant role in this visualization. At first, the whole environment's background color has been chosen white so that the movement of every character can be seen clearly. Then a function has been implemented for the prey's color changing. In this simulation, the prey interchanges into five colors to represent five states. In the beginning, when the prey is inactive and isolated, it turns into blue color. When the prey starts to move gradually, the color turns green from blue. When the concentration of preys increases in a particular area, the color of prey gradually turns yellow from green. Again when prey is separated, it turns from yellow to green to blue (Figure 5.7). The shade of these colors changes according to the concentration of prey. Again, the color of prey turns white when prey is selected as a target by the leader of predators (Figure 5.8). When the target changes, the color also changes with it. A prey becomes black when a predator has terminated that prey (Figure 5.9).

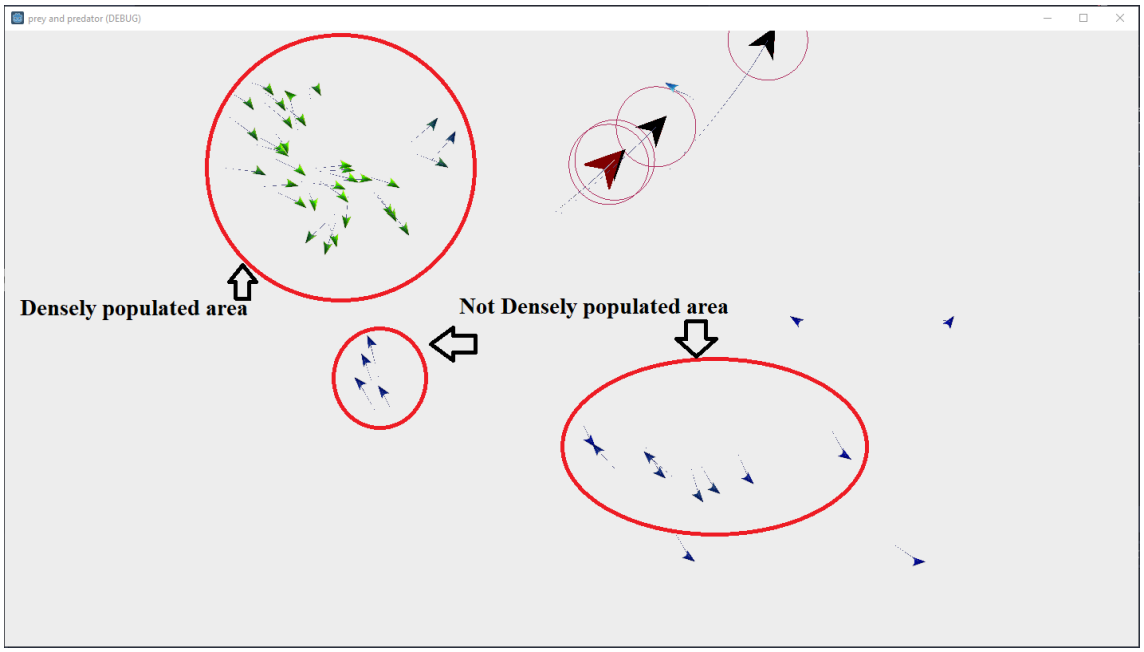


Figure 5.7: Boids colour transformation according to boid's density

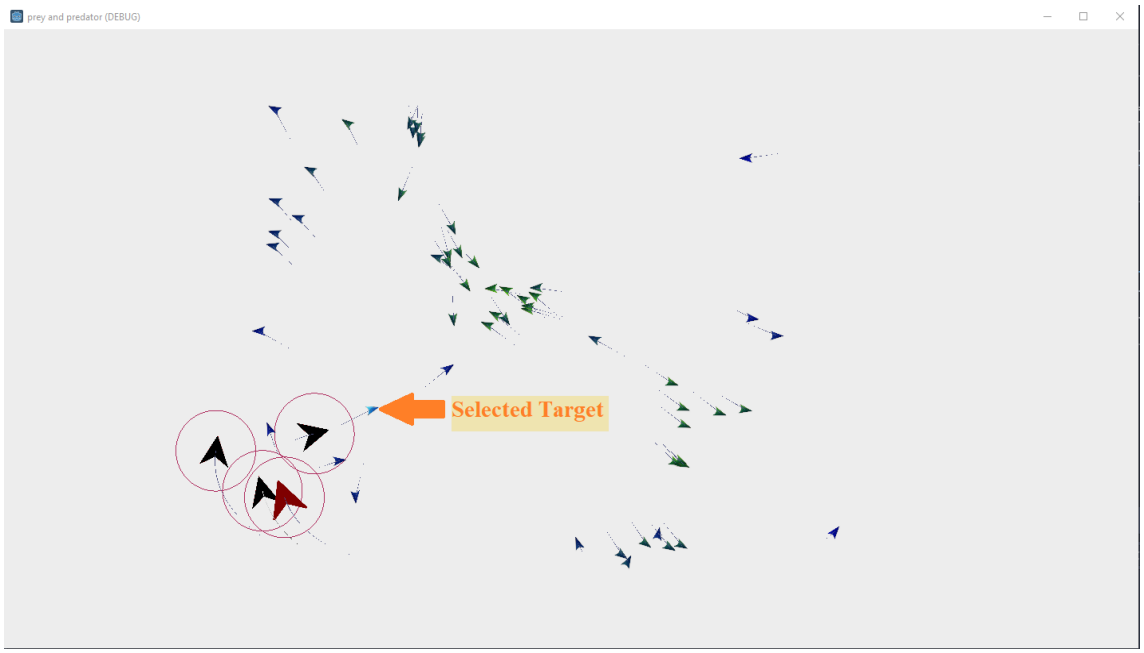


Figure 5.8: Target selected by leader of the predators

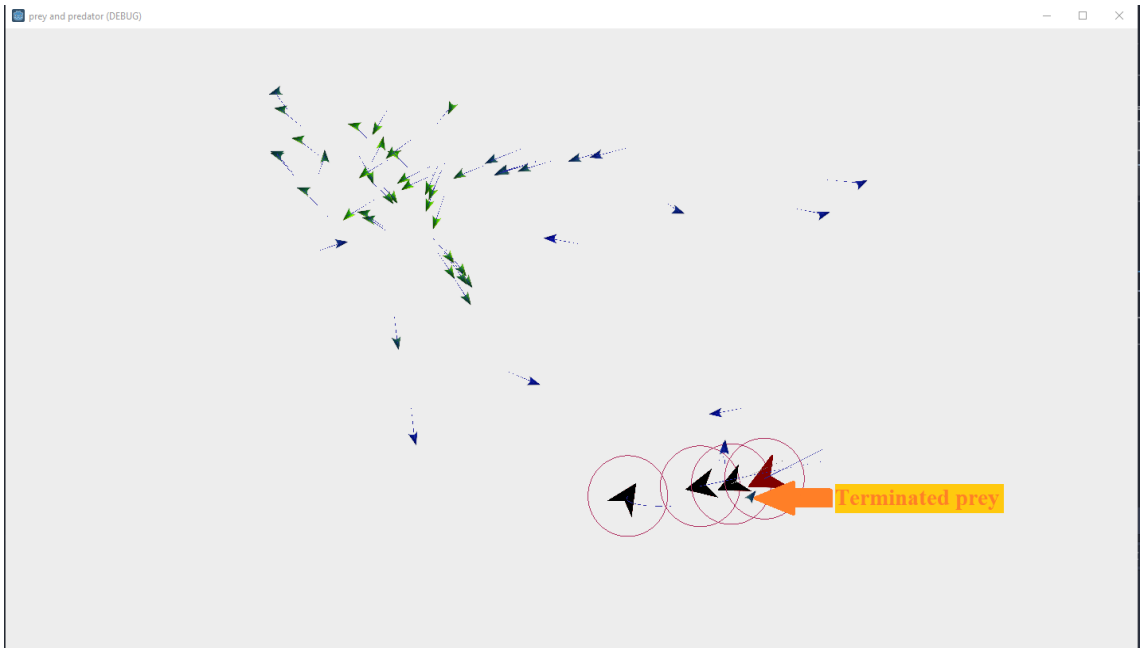


Figure 5.9: Terminated prey

A frame-freezing attribute is used when the target comes inside a specific range of a predator. It slows down the frame rate of the whole process to observe all the movements clearly and slowly. This feature helps a lot for analyzing what does a prey and predator do at that specific moment.

5.1.3 Flocks Local Parameters

As prey needs to maintain their flocking behavior, they use some parameters to change values according to users' preferences. We have implemented these parameters with scrips into the Godot environment (Figure 5.10). The first parameter is for controlling the number of preys. In the script, we have given a default value for predators, and in the interface, we can change the value as we want. Here, if the integer value of the parameter increases, the number of prey increases, and if the integer value decreases, the number of prey decreases. Then comes the visual parameter range of the prey. By increasing the integer value of the visual range, the navigation site of the prey increases, and the prey can detect predators from a distance. Moreover, a prey can follow other prey's movement to maintain the flocking behavior. By decreasing the visual range, the navigation range decreases for prey.

The separation distance is a parameter by which the program can control the minimum distance between the preys. If separation distance is increased, the preys maintain a greater distance between themselves. Again, if separation distance is decreased, the preys maintain less distance between themselves. With the parameter predator's minimum distance, the distance maintained between prey and predator has been controlled. This parameter works more like a safe range for prey. If a prey detects any predator in this range, it gets alert and runs away from the predator. Like other parameters, increasing the value of this parameter increases the range, and decreasing the parameter value decreases the range.

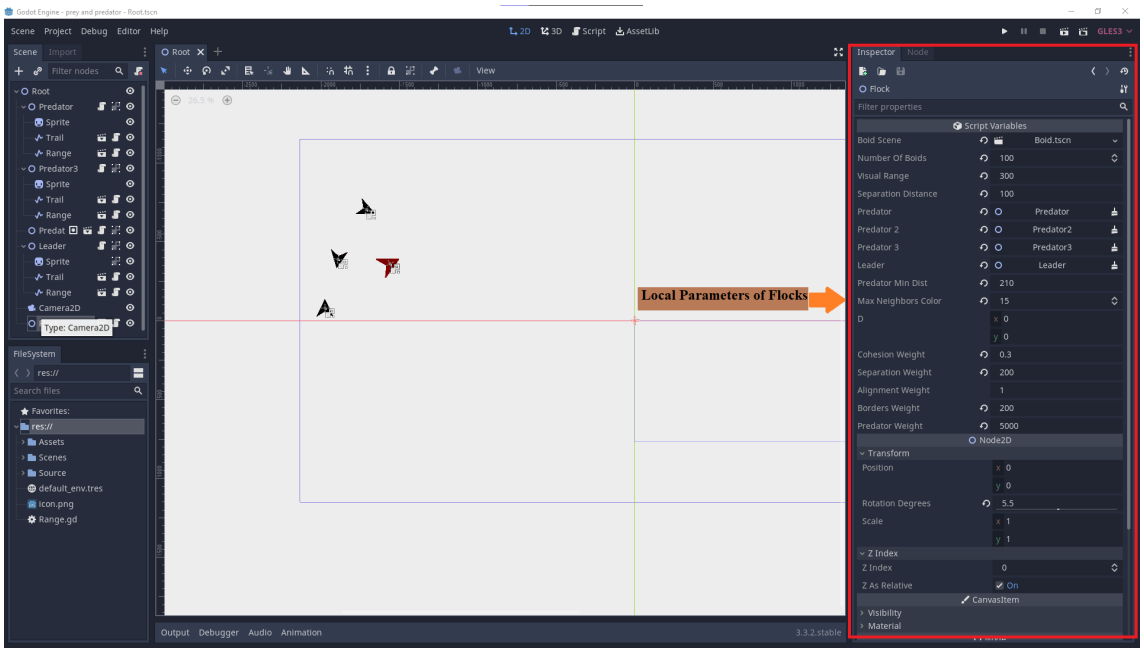


Figure 5.10: Flocks local parameters

The parameter maximum neighbour color is used for maintaining the color shade according to the density. It works like a density indicator range. Changing the value of this parameter can determine if the population is danced or not with those numbers of prey.

The parameters cohesion weight, separation weight, and alignment weight determine the integrity range of this behavior. For example, if the separation weight increases, the Boids maintain the separation distance more strictly. If the cohesion weight increases, the Boids come more closer to maintain the average position. These weight parameters are mainly responsible for Boids flocking behaviors.

Predator's weight determines the repulsion force that implies of prey when it crosses the safety range. If the parameter value decreases, the prey comes too close to the predator as the repulsion force decreases. On the other hand, increasing this parameter increases the repulsion force, and prey does not come close to the predators. Border weight also works like a predator's weight, but the border is fixed in a position. As a result, it just keeps the prey and predators stay in that specific area.

Predator and leader parameters inherit the information of different predators and prey. This information helps the preys to change their states and path of movement. Without this parameter, the prey will not interact with predators.

5.1.4 Local Parameters for Predators

Like prey, predators also have their characteristics, which are controlled by different parameters. We have implemented these parameters with scrips into the Godot environment as we have done for the preys (Figure 5.11). Unlike prey, every predator has its parameters, and the predator leader has a little different set of parameters.

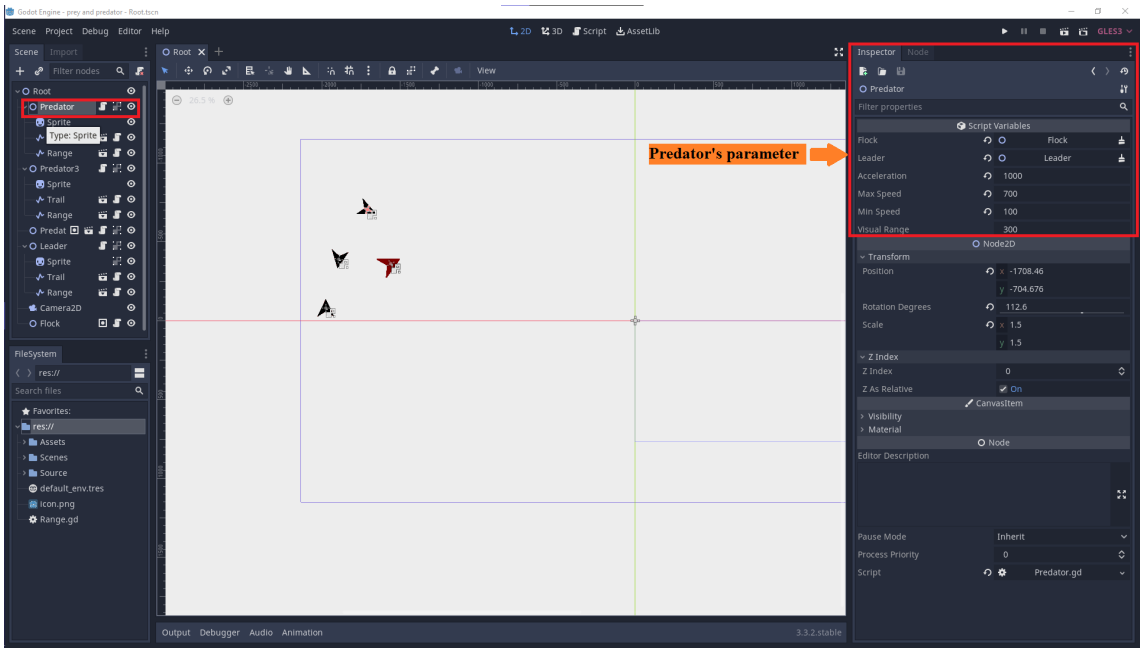


Figure 5.11: Parameters of the predators

First, in the parameter set of the predator leader (Figure 5.12), accelerating speed, maximum speed, minimum speed, and visual range can be seen. Also, a flock parameter is in the parameter set. This flock inherits the characteristics and information of the prey, which helps predators interact with the prey. The value of acceleration parameters determines the accelerating speed of predators. The speed of acceleration is proportional to the value of the parameter.

Maxim speed and minimum speed determine the predator's highest speed and lowest speed. When the predators are in inactive mode, they move with the minimum speed. On the other hand, when the predators are active, they move at maximum speed. These values of parameters are different in every predator.

Visual range is one of the essential parameters for the leader and other predators. For the leader, this range usually is more extensive than the other predators. It helps the leader to select targets by analyzing their speed and positions. For other predators, the visual range helps to hunt and terminate the target. The visual range is proportional to the value of the parameter.

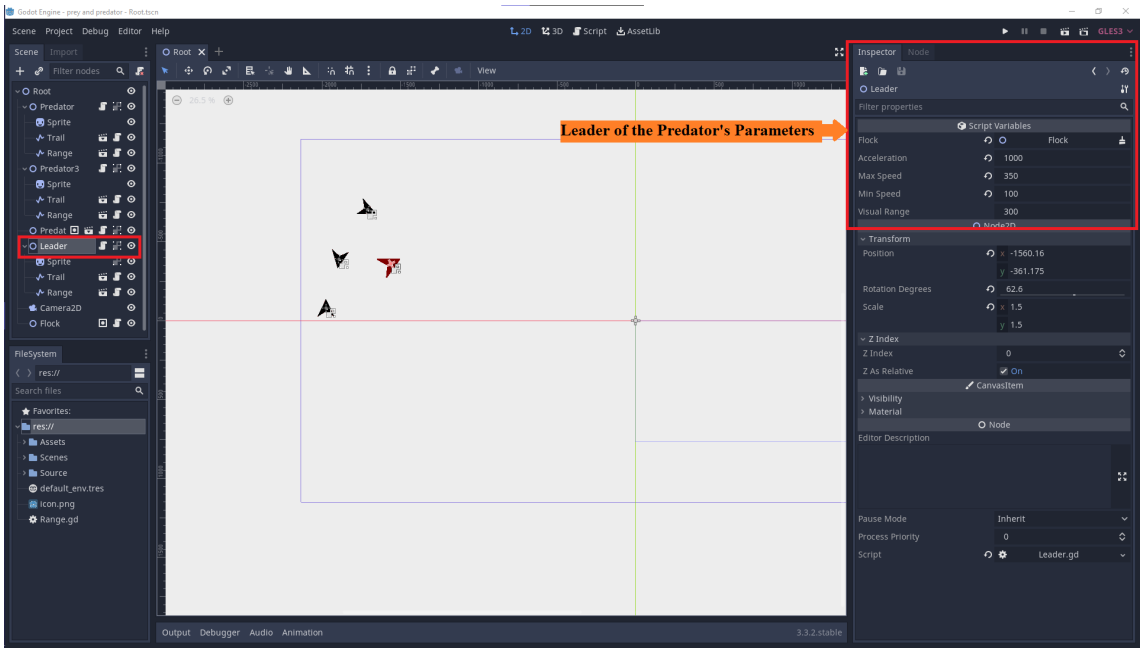


Figure 5.12: Parameters of the predator's leader

Except for the leader, other predators have an extra parameter, which is the leader (Figure 5.12). This parameter inherits the leaders selected target for the other predators. Without this parameter, predators will always be in the inactive state and not go for any target.

5.2 Results

After implementing all the characteristics of prey and predator with their parameters, we ran the simulation and got results. Before implementation, we theoretically assumed a result. After completing the program, we got the result and compared it with our expected one. Here, we discussed these expected results and the actual results we got and compared them for accuracy.

5.2.1 Expected Results

At first, we expected a group of prey was in a path moving with flocking behavior. Another group of predators was moving forward to the group of prey intended to hunt. In the Figure 5.13, we can see this initial state, where blue triangles are considered prey or deer, and orange triangles are assumed to be predators. Furthermore, they are moving in an open field.

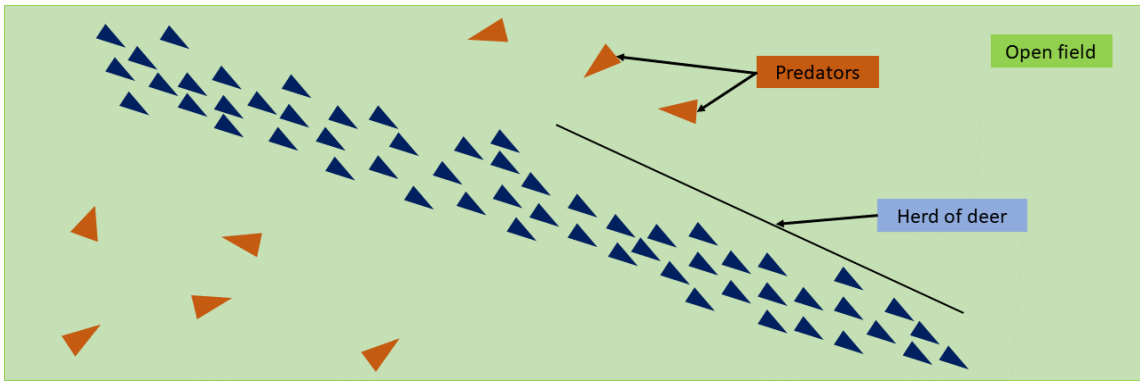


Figure 5.13: Expected initial state of deer herd and wolves

In the next step, we assumed the predators or wolf go near the deer herd to distract the whole group and find the weakest one. In the Figure 5.14, we see the light orange colored circle around the orange triangles has represented the visual range of predators.

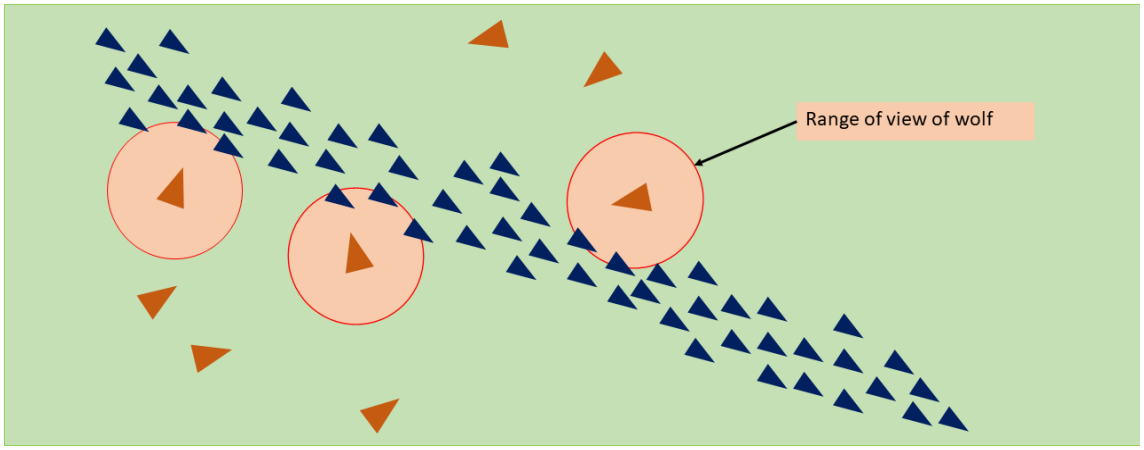


Figure 5.14: Expected wolves movement to distract deer herd

Then, the predators come close to the prey's visual range, represented by the blue circle around the blue triangle (Figure 5.15). When the deer senses the presence inside their range, they move in the opposite direction and get scattered.

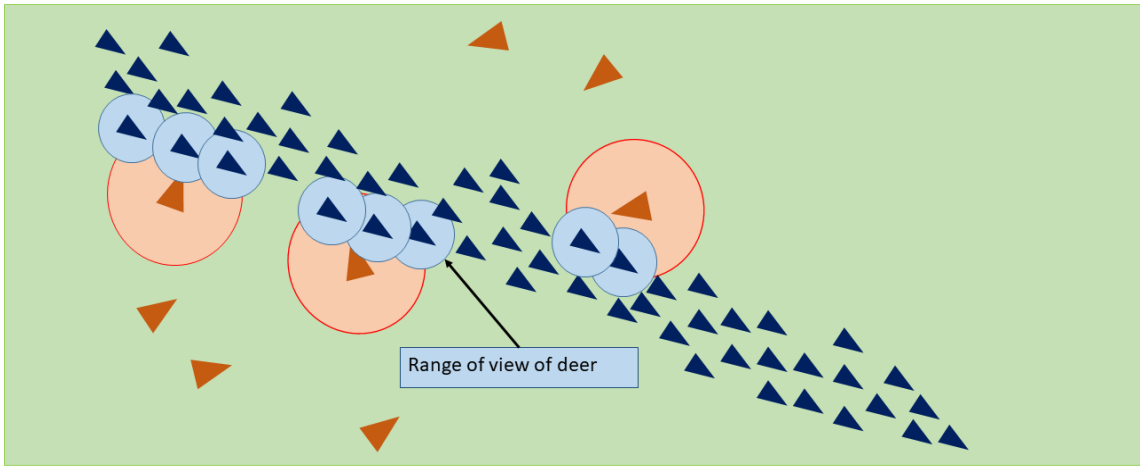


Figure 5.15: wolves are coming inside the visual range of deer

When the preys are scattered, the leader of the predators, denoted by the red triangle (Figure 5.16), starts to analyze the speed and position of preys to find the weakest one and set it as a target for others.

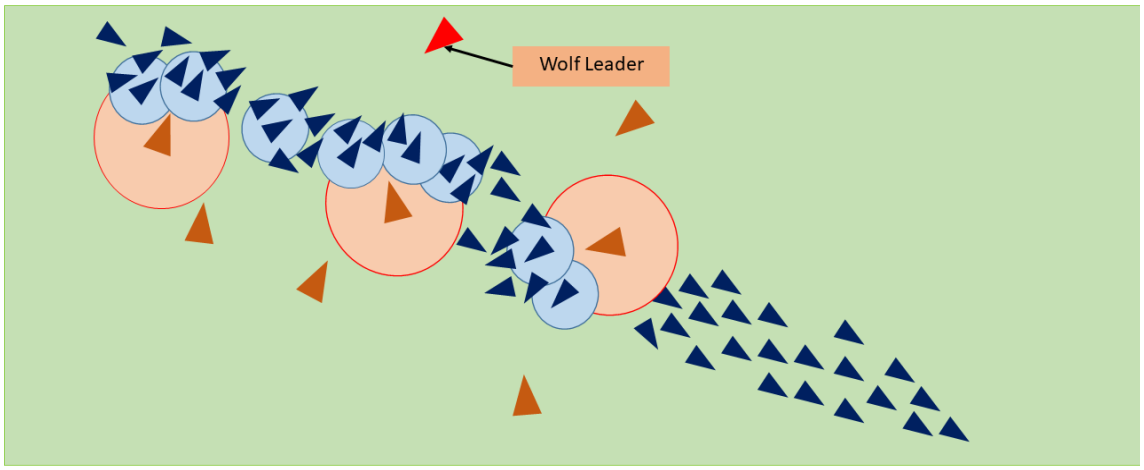


Figure 5.16: Deer heard is scattered by the wolves

The wolf's leader selects a target shown by a black-colored triangle (Figure 5.17). Then the leader passes the position of the target to the other predators.

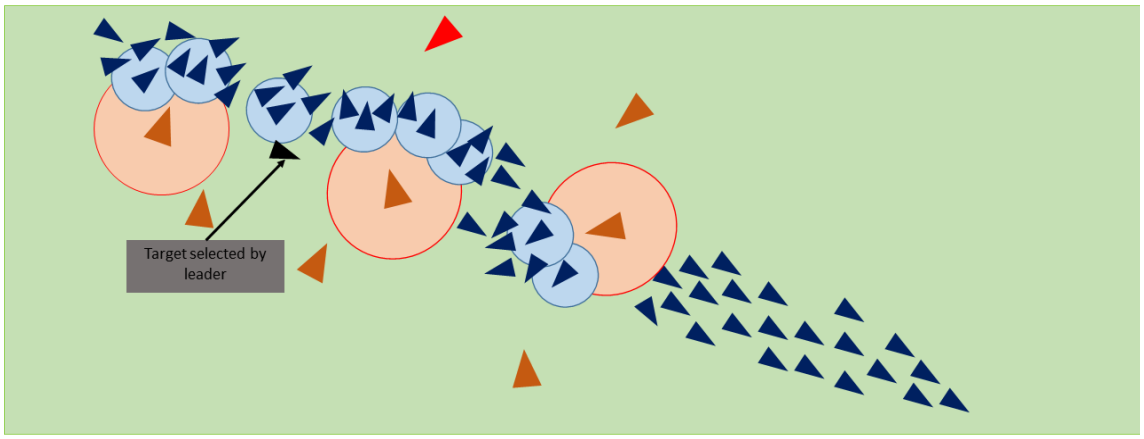


Figure 5.17: Leader of the wolves choosing target

The leader approaches the target to isolate the target from the group of deer (Figure 5.18). Also, the leader's approaches confirm the target for the other wolves.

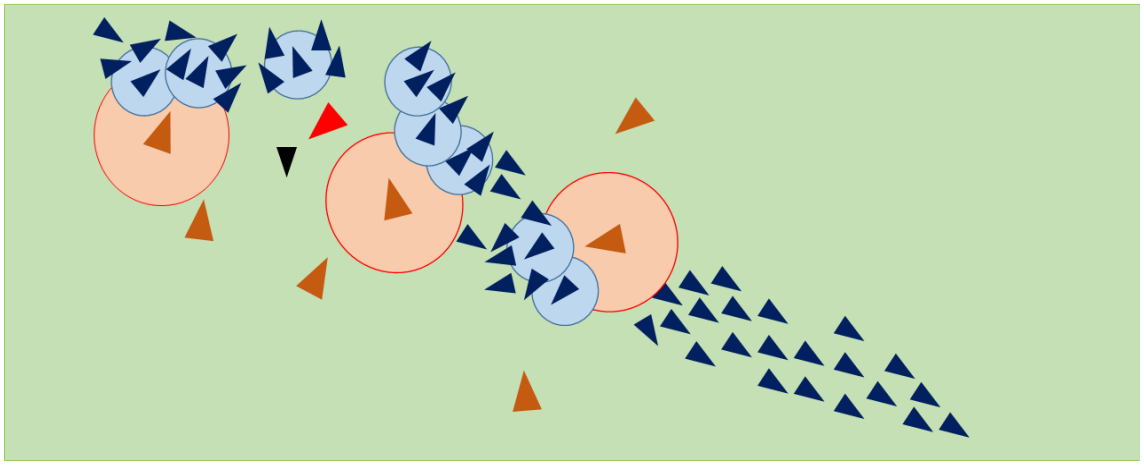


Figure 5.18: Leader of the wolves chasing the target

When all predators confirm the target, they approach it to surround it and isolate it from the group of deer (Figure 5.19).

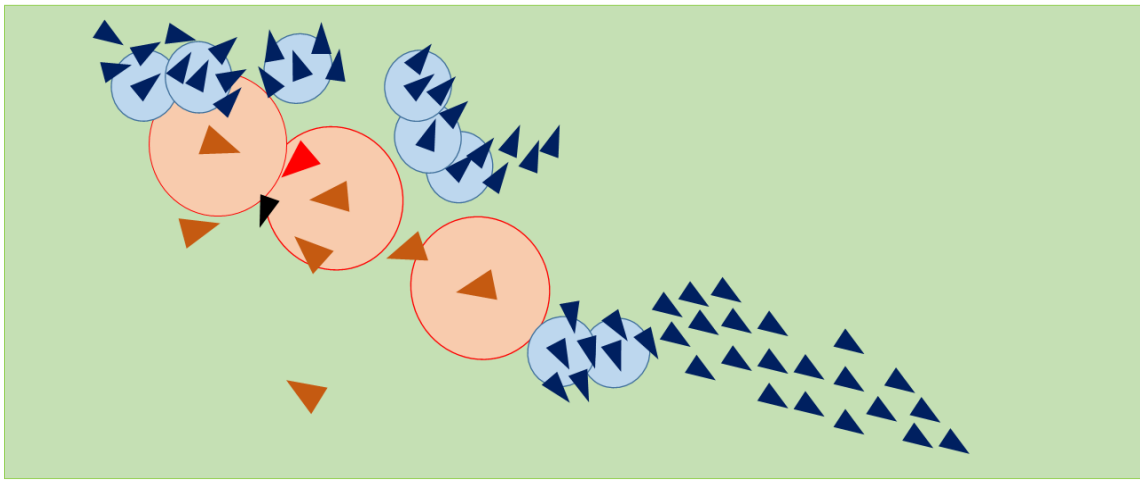


Figure 5.19: Wolves isolating the target

After the isolation of the target, every predator moves individually to terminate the target. When the predators reach the target close enough, they terminate it (Figure 5.20).

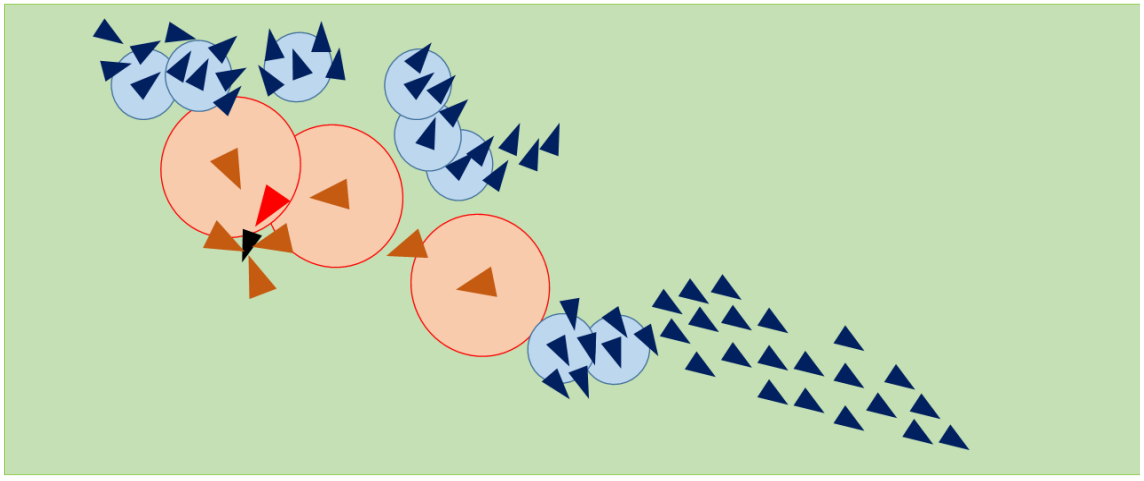


Figure 5.20: Wolves terminating the target

5.2.2 Simulated Results

After the simulation was implemented, we found out the results. At first, the predator showed two states, inactive and active states. At the inactive state, the wolves

or the predators move at their minimum speed toward the deer herd. In the Figure 5.21, black arrowheads are predators, and the red arrowhead is their leader. The red ring around them is the visual range of the predators. Also, the deer appears in a circular pattern at the beginning as we have implemented a equation for choosing the position from where the deer will stat to move.

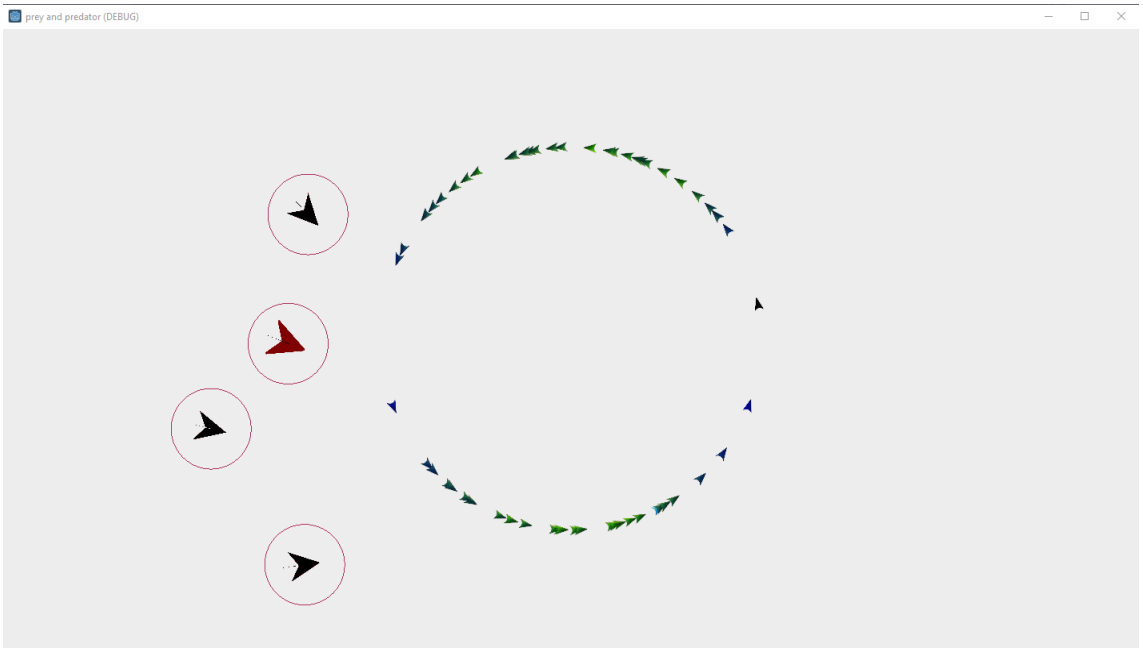


Figure 5.21: Wolves and deer herd's inactive sate

As soon as the wolves or the predator's leader selects a target, all the predators go into active mode and move faster toward the target (Figure 5.22).

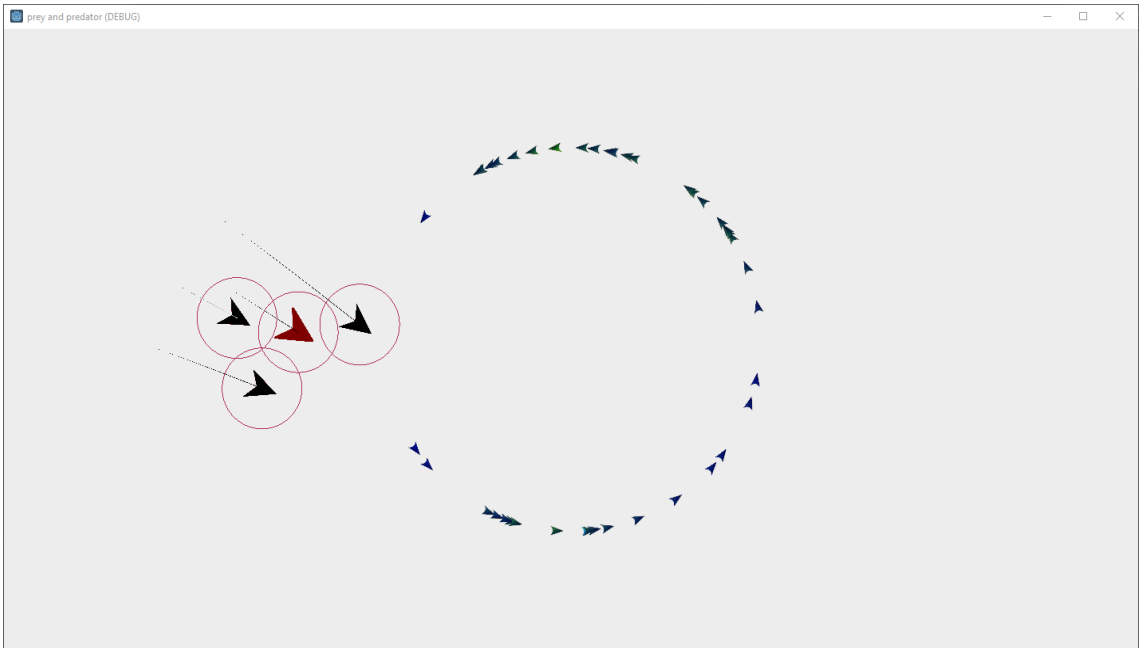


Figure 5.22: Wolves active sate

In the active state a tail appears behind the predators. This tail indicates active sate for both predators and prey.

In order to select the target, the predator chases the herd of deer(Figure 5.23). This chasing makes the deer scattered around. At that time, the target is selected by the leader.

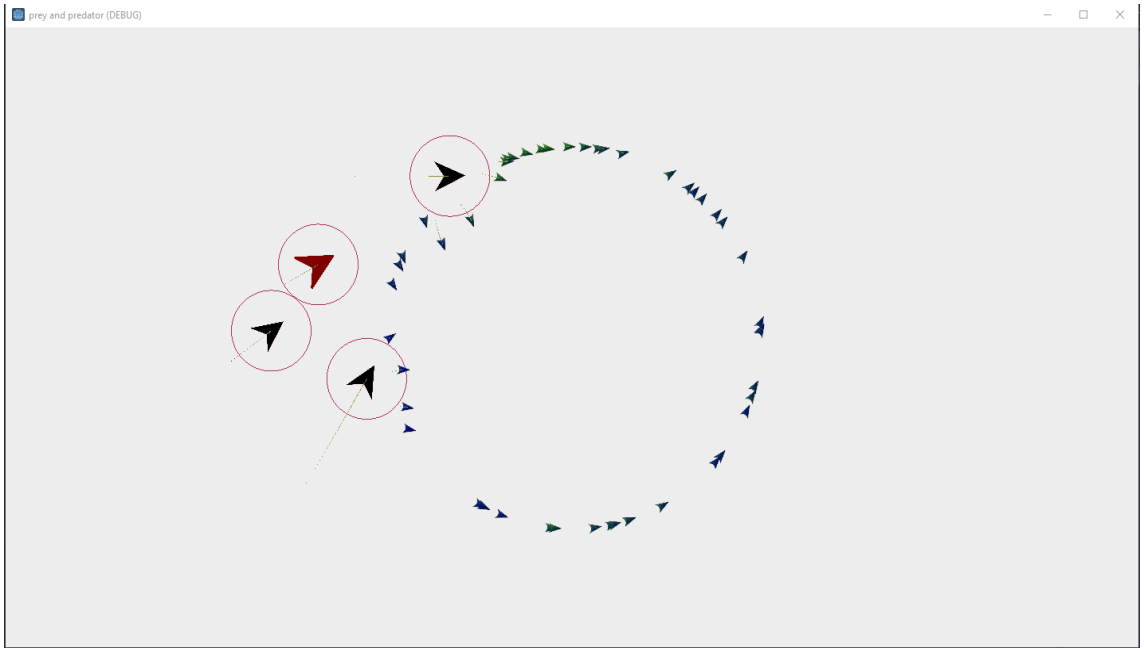


Figure 5.23: Wolves chasing deer hard to find target

After the target has been selected, the predator moves toward it. When the wolves are close enough to the target, they create a formation to isolate the target. In the Figure 5.24, the sky blue colored deer is the target, and the wolves made a half-circular formation around it.

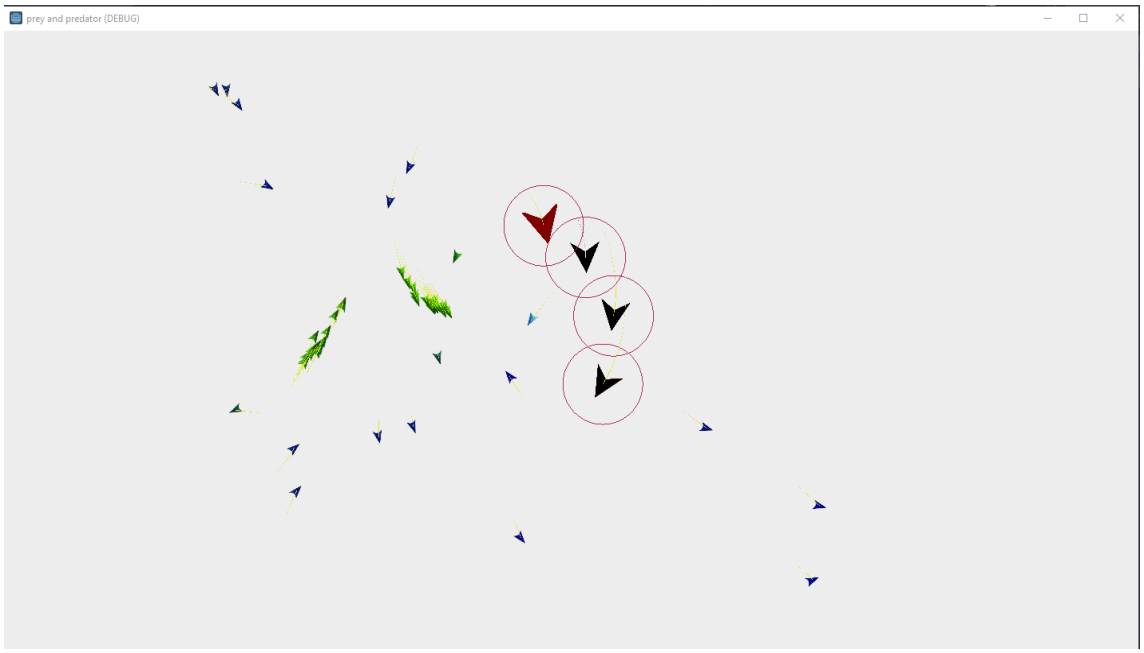


Figure 5.24: Wolves creating formation

Then the wolves continue the isolating process(Figure 5.25) until they successfully isolated the targeted deer from the deer herd (Figure 5.26).

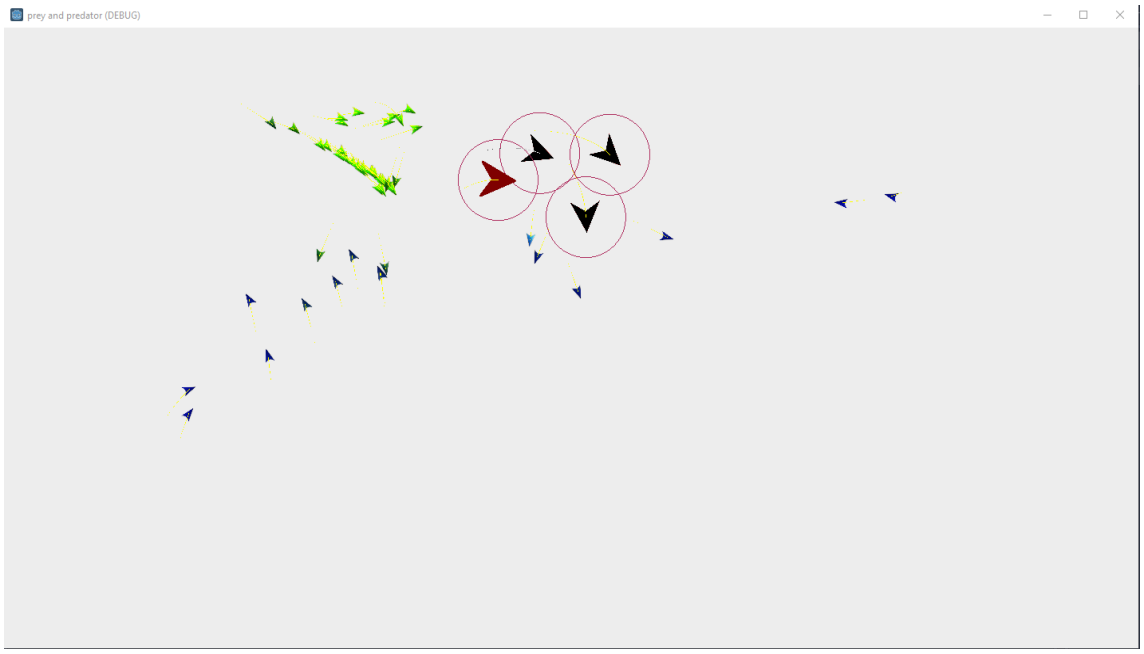


Figure 5.25: Isolating process of wolves

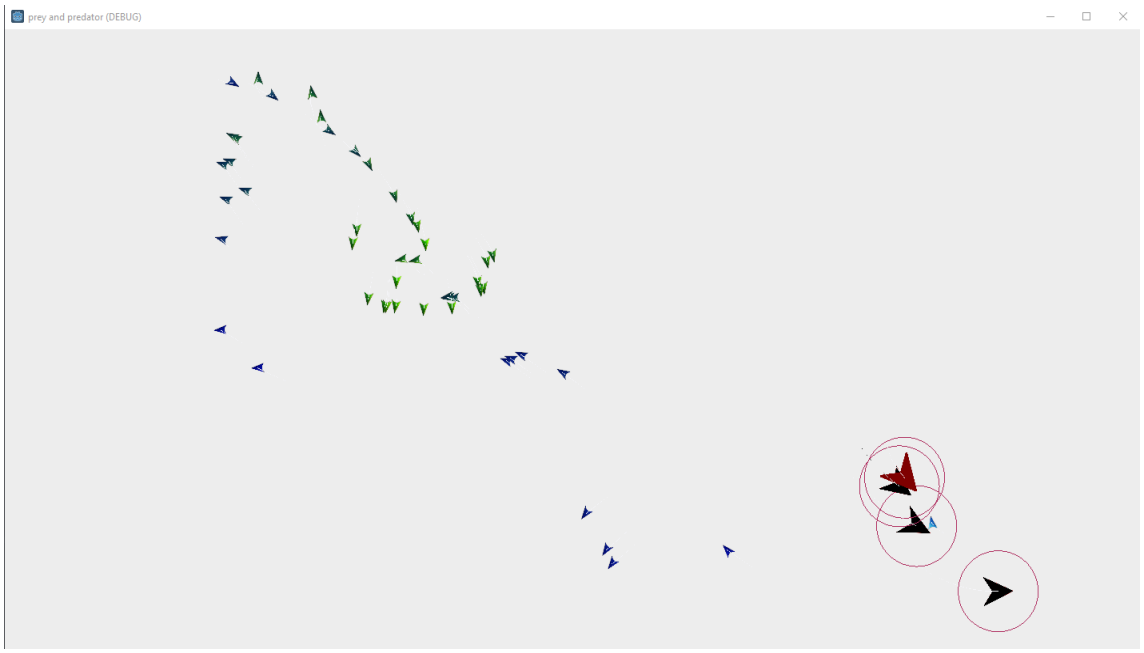


Figure 5.26: Isolated target

The final step of the process is to terminate the isolated target. In the Figure 5.27, the black-colored deer is the terminated deer.

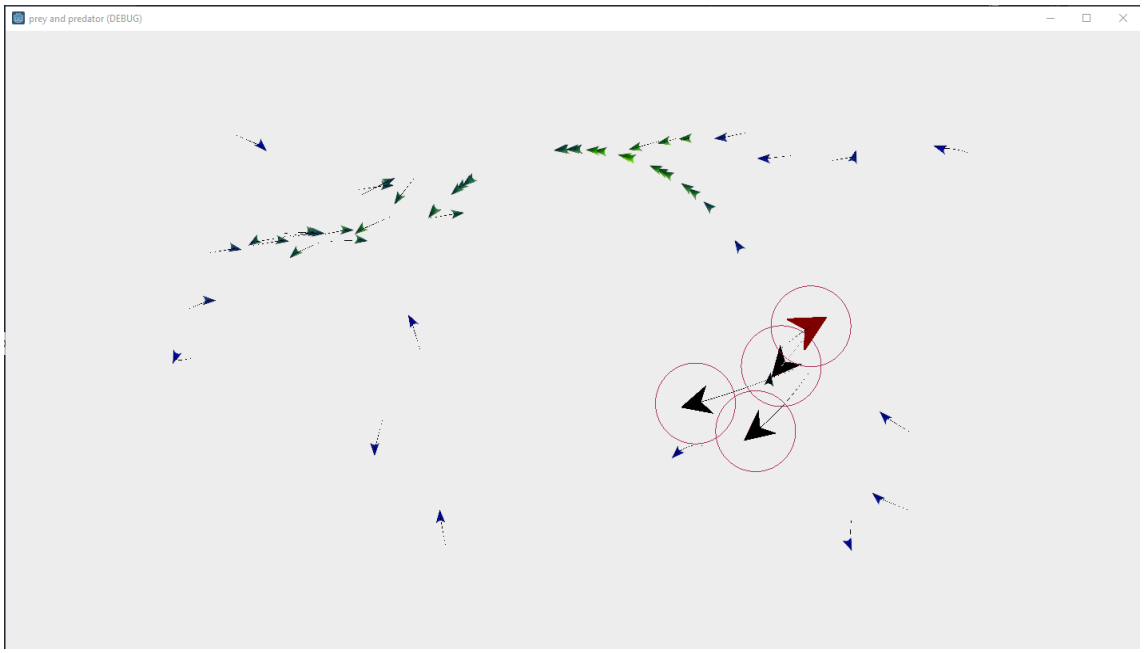


Figure 5.27: Target terminated

In the simulation, the process continues. After the termination is completed, predators select a new target and continue the whole process. Moreover, while the simulation is running, the leader changes the target intelligently. Sometimes the selected target escapes far away. At that moment, the leader selects another closer target.

Our simulation's results satisfy most of the expected results. Here, the predators and prey states are absent in our expected results, but we later implemented this feature in our simulation to make it smoother. In the simulation, we successfully showed the deer chasing of wolves, target selection by the leader, chasing the target, isolation target, and termination. However, the formation created by the predators is different from the expected formation. In the simulation, the predators always create a half-circle formation for isolation. Nevertheless, the expectation was that the predators might create different types of formation shapes. Here, more works could be done. A dedicated algorithm could be made to create the predators' formations to solve this problem because of the short time of this research that could not be done.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The desire to learn about nature and the environment in which we live drives us to explore wildlife. This exploration becomes the reason for endangering natural life. Also, unexpected natural behavior is uncommon. In order to observe these rare phenomena, one must be in a particular location at a specific moment, which is most of the time impossible. One of these phenomena is wolves hunting deer from a deer herd, which we have simulated in a computer using artificial life intelligence. This simulation required multiple algorithms to duplicate the natural characteristics of wolves and prey, which is deer. The behavior of the deer herd was implemented using a modified flocking algorithm. Also, multiple wolves and their leader was implemented with their predatory characteristics using some algorithms. All this implementation was possible in the Godot game engine. Before the experiment, we study some of the previous works on implementing the behavior of flocks. We also studied for working on a game engine. The concept of the convex hull was studied for implementing the predator's formation creating behavior. After the simulation was appropriately implemented, we ran multiple experiments. Every time we saw some variations in the results, which is similar to the natural phenomenon. The most exciting part of this experiment was changing the parameters' values like the speed of predators and prey and the visual range of entities. This allowed us to create different scenarios for the system. Also, watching the frame freezing sections was very satisfying. We could watch every moment slowly. This whole experiment aimed to learn about artificial life intelligence and use that in a computer-generated environment. We were able to achieve this goal, and some further scopes of working came up.

6.2 Future Work

We have put much effort put into this thesis, resulting in a great deal of study. Though we have only introduced the prey and predator, there was an idea of including surrounding environments like trees, obstacles, water streams, grasslands. These environmental entities would manipulate the movements of prey herds and predators. Also, this whole experiment can be done in a 3D environment with a vast game ground which will show more details of the process. Animated sprites can be used to make the simulation better looking. Much work can be done with the

predator's formation creation. One or multiple algorithms can be developed with the concept of a convex hull which we were unable to do in this thesis for the lack of time.

The central concept of this thesis was to understand the flocking behaviors and reactions of the entities in the presence of moving danger. This scenario can be relatable to other real-life scenarios. An example can be some fire accident in a populated area like a shopping mall or garments. Here, all the frightened people will move like a flock group. Smoke and fire will act like predators that the people want to escape. By adding some modifications to this thesis project, this scenario can be implemented. By analyzing different scenarios results, a perfect place for a fire exit can be determined.

Another possible scope for further work can be controlling procession tactics at the time of riots. From this thesis, the predators' tactics can be followed to simulate a scenario where a group of people is rioting and the law enforcement unit is trying to control them. Further work on this can also be helpful for modern warfare tactics.

From the analysis of flocks movement, the density of a moving population can be predicted and prevented. This feature can be helpful in this pandemic moment. The Covid-19 virus spared extensively in a more populated area. If the density of the population is controlled previously, the spread of the virus can be prevented too.

Analyzing the flocking behavior of locusts, we can predict their moving path and pattern. By combining machine learning with this thesis project, a process can be developed that can predict the moving path of locusts and recognize their attack pattern. Also, a simulation of the prediction can be made.

Flocking, escaping are natural behaviors, we can observe them in our everyday life. We are also a part of these characteristics, making a vast scope for future work on this thesis. Problems related to these natural behaviors can be found from time to time, and solutions can be introduced with the help of simulations, predictions that are similar to the work we have done in this thesis.

Bibliography

[1] R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Info. Pro. Lett.*, vol. 1, pp. 132–133, 1972.

[2] R. A. Jarvis, "On the identification of the convex hull of a finite set of points in the plane," *Information processing letters*, vol. 2, no. 1, pp. 18–21, 1973.

[3] G. T. Toussaint and D. Avis, "On a convex hull algorithm for polygons and its application to triangulation problems," *Pattern Recognition*, vol. 15, no. 1, pp. 23–29, 1982.

[4] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 25–34.

[5] J. Wang and G. Beni, "Cellular robotic system with stationary robots and its application to manufacturing lattices," in *Proceedings. IEEE International Symposium on Intelligent Control 1989*, IEEE, 1989, pp. 132–137.

[6] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proceedings of the IEEE international conference on neural networks*, Citeseer, vol. 4, 1995, pp. 1942–1948.

[7] I. L. Bajec, M. Mraz, and N. Zimic, *Boids with a fuzzy way of thinking*. Citeseer, 2003.

[8] A. V. Moere, "Time-varying data visualization using information flocking boids," in *IEEE Symposium on Information Visualization*, IEEE, 2004, pp. 97–104.

[9] C. Hartman and B. Benes, "Autonomous boids," *Computer Animation and Virtual Worlds*, vol. 17, no. 3-4, pp. 199–206, 2006.

[10] J. Holland and S. K. Semwal, "Flocking boids with geometric vision, perception and recognition," 2009.

[11] Wikipedia. (2009). "Yellowstone (british tv series)," [Online]. Available: https://en.wikipedia.org/wiki/Yellowstone_(British_TV_series). (published: 22.03.2009).

[12] C.-O. Erneholm, "Simulation of the flocking behavior of birds with the boids algorithm," *Royal Institute of Technology*, 2011.

[13] A. V. Husselmann and K. A. Hawick, "Simulating species interactions and complex emergence in multiple flocks of boids with gpus," in *Proc. IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2011)*, 2011, pp. 100–107.

[14] M. Joselli, E. B. Passos, J. R. S. Junior, M. Zamith, E. Clua, E. Soluri, and N. Tecnologias, "A flocking boids simulation and optimization structure for mobile multicore architectures," *Proceedings of SBGames*, pp. 83–92, 2012.

[15] S. Keerthi, K. Ashwini, and M. Vijaykumar, "Survey paper on swarm intelligence," *International Journal of Computer Applications*, vol. 115, no. 5, 2015.

[16] P. M. Mavhemwa and I. Nyangani, "Uniform spatial subdivision to improve boids algorithm in a gaming environment," *International Journal for Advance Research and Development*, vol. 3, no. 10, pp. 49–57, 2018.

[17] J. Connor, J. Nowell, B. Champion, and M. Joordens, "Analysis of robotic fish using swarming rules with limited sensory input," in *2019 14th Annual Conference System of Systems Engineering (SoSE)*, IEEE, 2019, pp. 69–74.

[18] S. Sengupta, S. Basak, and R. A. Peters, "Particle swarm optimization: A survey of historical and recent developments with hybridization perspectives," *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, pp. 157–191, 2019.

[19] P. G. Artaxo, A. Bourgois, H. Sardinha, H. Vieira, E. C. de Paiva, A. R. Fioravanti, and P. A. Vargas, "Autonomous cooperative flight control for airship swarms," *arXiv preprint arXiv:2004.07665*, 2020.

[20] CGTN-Official. (2020). "A flock of birds forming the shape of a bird through the sky in the uk," [Online]. Available: https://twitter.com/cgtnofficial/status/1303528554261610501. (tweeted: 09.09.2020).

[21] Discovery. (2020). "A fish herd forming the shape of a bird in the sea," [Online]. Available: https://www.facebook.com/Discovery/photos/a.106163613585/10159012089428586. (posted: 10.11.2020).

[22] C. Hahn, F. Ritz, P. Wikidal, T. Phan, T. Gabor, and C. Linnhoff-Popien, "Foraging swarms using multi-agent reinforcement learning," in *Artificial Life Conference Proceedings*, MIT Press, 2020, pp. 333–340.

[23] A. Y. Kapi, M. S. Sunar, and M. N. Zamri, "A review on informed search algorithms for video games pathfinding," *International Journal*, vol. 9, no. 3, 2020.

[24] K. Kasmarik, S. Abpeikar, M. M. Khan, N. Khattab, M. Barlow, and M. Garratt, "Autonomous recognition of collective behaviour in robot swarms," in *Australasian Joint Conference on Artificial Intelligence*, Springer, 2020, pp. 281–293.

[25] A. Thorn, "Introducing godot: Why migrate?" In *Moving from Unity to Godot*, Springer, 2020, pp. 1–14.

[26] T. Rowland. (2021). "Transition function," [Online]. Available: https://mathworld.wolfram.com/TransitionFunction.html. (published: 19.09.2021).

[27] Wikipedia. (2021). "Flocking (behavior)," [Online]. Available: https://en.wikipedia.org/wiki/Flocking_(behavior). (edited: 09.09.2021).