Test Driven Documentation (TDD)



***‘Boids’***

This page: Table of Contents and Team Member Listing

|  |  |  |
| --- | --- | --- |
| Table of Contents  [1 Executive](https://n-xovwktmtjsnaxyc2mwes2xu7pohqedmdm6zjw5q-2lu-script.googleusercontent.com/userCodeAppPanel) Summary  [1.1 Game](https://n-xovwktmtjsnaxyc2mwes2xu7pohqedmdm6zjw5q-2lu-script.googleusercontent.com/userCodeAppPanel) Overview  [1.2 Game](https://n-xovwktmtjsnaxyc2mwes2xu7pohqedmdm6zjw5q-2lu-script.googleusercontent.com/userCodeAppPanel) Summary  [2 Equipment](https://n-xovwktmtjsnaxyc2mwes2xu7pohqedmdm6zjw5q-2lu-script.googleusercontent.com/userCodeAppPanel)  [2.1 Hardware](https://n-xovwktmtjsnaxyc2mwes2xu7pohqedmdm6zjw5q-2lu-script.googleusercontent.com/userCodeAppPanel)  [2.2 Software](https://n-xovwktmtjsnaxyc2mwes2xu7pohqedmdm6zjw5q-2lu-script.googleusercontent.com/userCodeAppPanel)  [3](https://n-xovwktmtjsnaxyc2mwes2xu7pohqedmdm6zjw5q-2lu-script.googleusercontent.com/userCodeAppPanel) Evaluation  [3.1 Game Engine](https://n-xovwktmtjsnaxyc2mwes2xu7pohqedmdm6zjw5q-2lu-script.googleusercontent.com/userCodeAppPanel)  [3.2 Target Platform](https://n-xovwktmtjsnaxyc2mwes2xu7pohqedmdm6zjw5q-2lu-script.googleusercontent.com/userCodeAppPanel)  [4 File Formats & Naming Convention](https://n-xovwktmtjsnaxyc2mwes2xu7pohqedmdm6zjw5q-2lu-script.googleusercontent.com/userCodeAppPanel)  [5 AI Behavior](https://n-xovwktmtjsnaxyc2mwes2xu7pohqedmdm6zjw5q-2lu-script.googleusercontent.com/userCodeAppPanel)  [5.1](https://n-xovwktmtjsnaxyc2mwes2xu7pohqedmdm6zjw5q-2lu-script.googleusercontent.com/userCodeAppPanel) AI Flocking  [5.2](https://n-xovwktmtjsnaxyc2mwes2xu7pohqedmdm6zjw5q-2lu-script.googleusercontent.com/userCodeAppPanel) Boid Controller  [5.3 Asset](https://n-xovwktmtjsnaxyc2mwes2xu7pohqedmdm6zjw5q-2lu-script.googleusercontent.com/userCodeAppPanel) List     |  | | --- | |  | | Game Development Team Members    PRODUCER  Christian Barajas    PRODUCTION MANAGER  Christian Barajas    PRODUCTION COORDINATOR  Christian Barajas  GAME DESIGNERS  Christian Barajas  SYSTEMS/IT COORDINATOR  Christian Barajas  PROGRAMMERS  Christian Barajas  TECHNICAL ARTISTS  Christian Barajas  AUDIO ENGINEERS  Christian Barajas  UX TESTERS  Christian Barajas |

|  |
| --- |
|  |

# 1 Executive Summary

## Game Overview

Title: Boids

Platform: PC Standalone + iOS & Android

Genre: Artificial Intelligence

Rating: (10+) ESRB

Target: Casual gamer (aging from 12 - 30)

Release date: April, 2017

Publisher: C && C Corporation

Description: Boids project implements a 3D simulation of flocking behaviors that would generally see within a flock of birds, but in this case we are using a school of fish. The objective of this project is to simulate flocking behaviors of a school of fish. The flocking AI will also be able to switch between three flight modes which are lay flight, circle a tree, and follow the leader. The simulation takes place underwater within an ocean. The flocking AI will have certain boundaries in which they can swim in. In addition, a user will be able to switch between flight modes with the use of a GUI.

## Technical Summary

The artificial intelligence in this project, which is a school of fish, simulates three states which include lazy flight mode, circle a tree, and follow the leader mode. This flocking AI controller will allow you to see a simulation of behaviors that are adopted by a school of fish. There are no future plans to fully develop this project for it is only to learn how to implement AI behaviors. The total cost to make this project was $0, with no expected revenue. This game is solely for PC at the moment.

# 2 Equipment

## 2.1 Hardware

\*This project only used a Windows PC laptop in which I already owned.

*Product Task Cost Quantity*

Hp Pavilion Laptop Asset Creation $0 1

Game Development

**Total: $0**

## 2.2 Software

*Product Task Cost Quantity*

Unity Pro Game Editor/Engine $0 1

Adobe Photoshop (free version) Texture Painting

Photo Editing

**Total: $0**

# 3 Evaluation

## 3.1 Game Engine

The game engine utilized for the development of this game is Unity. Unity made the development of this game much smoother and it sped up the development process, optimize the game, and achieve a desired result without having to pull out too much hair.

## 3.2 Target Platform

This game is made for PC and I have a wish to make it compatible with smart phones but the main target platform for this game is for PC. On the other hand, it will be a great learning experience to tackle on the mobile development of this game. Learning mobile development would increase utilization by linking the project in two separate markets.

# 4 File Formats & Naming Conventions

*Asset Type SubType Naming Convention File Format*

Animations AnimationClipName FBX

Graphics Characters CharacterName PNG

Enviornment EnviornmentTileName PNG

PhotoShop

Props PropName PNG

PhotoShop

Materials MaterialName \*.mat

Scripts Character CharacterName C#

Enviornemt EnviornmentName C#

Canvas CanvasType C#

UI UIType C#

UI UIElementState PNG

# 5 AI Behavior

## 5.1 AI Flocking

void Update () {

if(Vector3.Distance(transform.position, Vector3.zero) >= BoidController.tankSize)

{

turning = true;

}

else

{

turning = false;

}

if (turning)

{

Vector3 direction = Vector3.zero - transform.position;

transform.rotation = Quaternion.Slerp(transform.rotation, Quaternion.LookRotation(direction), rotationSpeed \* Time.deltaTime);

speed = Random.Range(0.5f, 1);

}

else

{

if (Random.Range(0, 5) < 1)

ApplyRules();

}

transform.Translate(0, 0, Time.deltaTime \* speed);

}

* Once per frame Update() checks to see if the flock is at the maximum are of the tank size, which is initialized prior and is just the max range the AI can swim within. If they are at the edge of their swim boundaries we set the ‘turning’ instance to true in order to make the flock swim back into swimmable area. If turning is true the code then calculates the direction back toward the center of the swimmable area and rotates the fish. Within the code snippet you can notice that there is a ‘Quanternion.Slerp’, this ‘Slerp’ is so the rotation of the fish can look more natural. If the fish don’t need to be rotated back toward the center then we apply the ApplyRules() method.

void ApplyRules()

{

GameObject[] gos;

gos = BoidController.allFish;

Vector3 flockCenter = Vector3.zero;

Vector3 flockAvoid = Vector3.zero;

float gSpeed = 0.1f;

Vector3 goalPos = BoidController.goalPos;

float dist;

int groupSize = 0;

* This method is the flocking rules for the fish AI. We initialize an array that is going to hold our fish prefab. ‘flockCenter’ points to the center of the group of fish, and ‘flockAvoid’ is so each fish avoids hitting other fish. Then we initialize a goal position for the fish to swim toward which is represented by ‘goalPos’. ‘dist’ is initialized to hold how close the fish can get to one another.

foreach (GameObject go in gos)

{

if(go != this.gameObject)

{

dist = Vector3.Distance(go.transform.position, this.transform.position);

if(dist <= neighbourDistance)

{

flockCenter += go.transform.position;

groupSize++;

if(dist < 1.0f)

{

flockAvoid = flockAvoid + (this.transform.position - go.transform.position);

}

Flocking anotherFlock = go.GetComponent<Flocking>();

gSpeed = gSpeed + anotherFlock.speed;

}

}

}

* We use a foreach() loop so we can apply all the rules to every fish in the scene. We use the ‘if(dist <= neighbor Distance)’ to check the distance of the fish to determine the center of a group of fish near each other. We do this so that if fish are close enough together, they will no longer swim randomly they will swim together with the other fish nearby to swim as one group. Then within this group we use a ‘if(dist < 1.0f)’ so that if fish get to close to each other, they will need to avoid each other. Then we go and get another flock from within our ‘Flocking.cs’ script.

if(groupSize > 0)

{

flockCenter = flockCenter / groupSize + (goalPos - this.transform.position);

speed = gSpeed / groupSize;

Vector3 direction = (flockCenter + flockAvoid) - transform.position;

if (direction != Vector3.zero)

transform.rotation = Quaternion.Slerp(transform.rotation, Quaternion.LookRotation(direction), rotationSpeed \* Time.deltaTime);

}

}

}

* Here we do checks in order to control a group of fish. First, we check if the group size is greater than zero meaning there exists a group of fish. If this passes we then calculate the average center of the group and the average speed of that group. Then we apply rules so that the fish know what direction to turn when the group turns, or if they even need to turn. And again if they need to turn we use the ‘Slerp’ in prder for the fish to rotate more life like.

## 5.2 Boid Controller

public GameObject fishPrefab; // fish gameObject

public GameObject goalPrefab; // waypoint gameObject for follow the leader

public static int tankSize = 4; // area where fish can swim within scene

static int numFish = 100; // controls total number of fish in scene

public static GameObject[] allFish = new GameObject[numFish];

public static Vector3 goalPos = Vector3.zero; // target position of flock

// flags to control the type of flight mode

private int lazyFlightMode = 0;

private int circleATree = 0;

private int followTheLeader = 0;

- Here we are just initializing our fish prefab, goal prefab, tank size, number of fish, array to hold number of fish gameobjects, and we set flags for each flight mode implemented.

for(int i = 0; i < numFish; i++)

{

Vector3 pos = new Vector3(Random.Range(-tankSize, tankSize), Random.Range(-tankSize, tankSize), Random.Range(-tankSize, tankSize));

allFish[i] = (GameObject) Instantiate (fishPrefab, pos, Quaternion.identity);

}

}

* Here we are using a for() loop to iterate through all fish we need to instantiate. So we first create a position for the fish in 3D space that is within the range of the tank size. So every fish will have a random spawn that is within the swimmable area. Then, we instantiate each fish prefab into our scene.

void Update () {

if (lazyFlightMode == 1)

{

if (Random.Range(0, 10000) < 50)

{

goalPos = new Vector3(Random.Range(-tankSize, tankSize), Random.Range(-tankSize, tankSize), Random.Range(-tankSize, tankSize));

}

}

if(circleATree == 1)

{

}

if(followTheLeader == 1)

{

if (Random.Range(0, 10000) < 50)

{

goalPos = new Vector3(Random.Range(-tankSize, tankSize), Random.Range(-tankSize, tankSize), Random.Range(-tankSize, tankSize));

goalPrefab.transform.position = goalPos;

}

}

}

* Here we are doing checks once per frame in order to determine which flight mode we are in. All flight modes are initialized to 0 and are only set to 1 when a user clicks the UI buttons during run time. If lazy flight mode is clicked, then there will be a randomly set goal position for the fish to swim toward and every frame the goal position will be randomly reset. If the user clicks on circle a tree flight mode, then notice the method is empty, this is so that the fish have no goal position to swim towards so then the ‘Flocking.cs’ script will apply the flocking rules and groups of fish will only swim around the center of their group that they belong to. This simulates a group of fish swimming in circles. Then if the user clicks the follow the leader flight mode, then the goal prefab will be randomly set and instantiated so that the fish fly toward a waypoint that a user can see.

public void LazyFlight()

{

lazyFlightMode = 1;

circleATree = 0;

followTheLeader = 0;

}

public void CircleATree()

{

circleATree = 1;

lazyFlightMode = 0;

followTheLeader = 0;

}

public void FollowTheLeader()

{

followTheLeader = 1;

lazyFlightMode = 0;

circleATree = 0;

}

* These methods are only called when the user clicks on a UI button. Depending which button is pressed it calls the appropriate method that corresponds to the button clicked. So when each button is clicked that flight mode flag is set so that the Update() method can begin the correct flocking behavior.

## 5.3 Asset List

AI Fish Prefab (cruscarp)

Goal Position Waypoint 1

Enviornment groundPlaneMesh

seaGround\_diffuse

seaGround\_diffuse

Daylight Simple Water

Nighttime Simple Water

# 