TASK 2 – Game Design Document

Introduction to Artificial Intelligence Ian Mallia – Vincenzo Calafiore – BA Interactive Media 2019 – 2020

# Project Description

The game is a: 2D, single-player, top-down shooting game. The objective of the player is to survive/defeat incoming waves of enemies that behave in different ways. If the player successfully defeats all of the waves he will encounter a Boss. The boss fight will have different stages, and the player must defeat the boss in all his stages to successfully win the game. The player has an amount of life and to regain health he must kill enemies that have a chance of dropping a new weapon or a health refill.

Our AI-related objective is to create simple AIs that are fun to interact with and are the pillars of the overall gameplay system.

Our inspiration and reference in this project have been the game: The Binding of Isaac because in our opinion has a great combination of game design, art style and demonstrates that simple and “arcadeish” mechanics and AI systems can create fun and repayable system.

# Characters

## Player:

The main character is a priest that like exorcisms and he dreams to be a hero, but he knows that God does not exist and exorcisms either, so while he lives his ordinary priest life, filled with: churches, old people, ordinary problems, cleric ranks, and kids; the night he dreams about being a hero in a cemetery, defeating horrible monsters. He has two incredible powers:

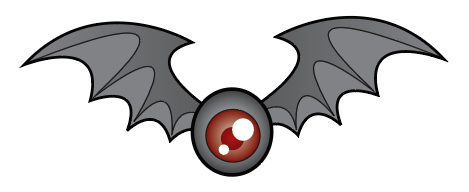
* The power of Walking: that allows him to walk in 4 directions!
* The power of Holding: that allows him to hold long, huge, and powerful staffs that shoot projectiles (like a gun! But better!)

## Currau:

Currau is the representation of a bad kid that wants to touch the priest with melee attacks. Currau has always been a kid that likes going to the church, only with the intent to induce temptation into the priest's mind. But like every bad kid, he will be punished by the good priest.

* This AI does X Attacks
* It has X health points
* It does X damage

## Savvatore:



Savvatore represents the worse nightmare of the poor priest, it is a horrible bat that shoots blood drops, probably sucked from the evil kids! What a horrible monster!

* This AI does X Attacks
* It has X health points
* It does X damage

## Ianuzzu:



Ianuzzu is represented by a tortoise. He was the priest’s pet in his childhood but he thought it was a water turtle so he threw him in water which resulted in an inevitable and dreadful death. Now Ianuzzu is back, and ready to revenge!

* This AI does X Attacks
* It has X health points
* It does X damage

## Purtusu:

Purtusu is the representation of one of the worse fear of the gentle priest, Purtusu is a huge hole (it is well known that priests do not like holes), Purtusu spawns Ianuzzu back to life.

* This AI duty is to spawn waves of enemies

## Caminanti:

These NPCs are the representation of the classic believers that goes to the priest’s church, they go in a group, they believe in god just when they need something, but they do not believe in the church as an institution! that is the reason why they came and goes away from the dream whenever they like; the reason is they do not count as true god’s followers.

* This AI has been implemented as a crowd

# Spiritus Sancti

What is a priest without an abstract representation of his faith! The Spiritus Sancti follows the brave priest where ever he goes, it doesn’t miss a step, it just won’t go away, it is stickier than Spiderman, it worse than a foreigner on a bus that does not respect the social distancing rules!

* This AI works as a follower



## The Boss

The boss represents the dream itself and the priest have to defeat his dream in an endless and epic battle to the edge of sanity!

* This AI does X Attacks
* It has X health points
* It does X damage
* It has X phases were: 1st X 2nd X and 3rd X

# Story

The player has to play within the shoes of the most amazing priest existing in this world; The Priest is an adventurous person; his dream is to defeat the evil in this world! But since this world does not care about a 2000-year-old doctrine, what remains of his goal is a pity dream.

Every night when he goes to sleep, he dreams about defeating his worse enemies and his perverse psychology. Every night he tries to win the dream by defeating hordes of filthy enemies until he gets to the final boss, the representation of the dream itself!

# Story Progression

Since the priest lives a pity life that frustrates him, and since he believes he is doing the right thing in the world, the course of the events will never change, and he is condemned to dream every night an endless loop of nightmares until he defeats the dream’s boss!

# Gameplay

The gameplay consists of a top-down shooter, where the player has to defeat hordes of enemies until he survives and defeats the final boss.

## Goals

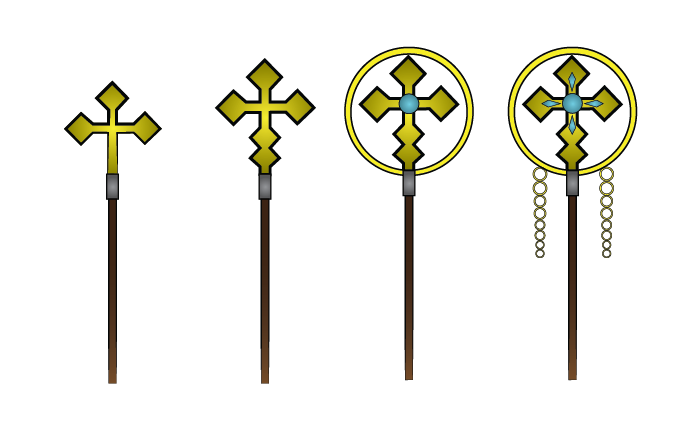
Defeat the final boss

## User Skills

Arrow keys to move in all 4 directions

Left mouse button to shoot fire bullets

The game has 4 different weapons that are differentiated by type:



* Base weapon
* Fast bullets Low damage weapon
* Slow bullets High damage weapon
* Ulitmate weapon

## Game Mechanics

* Move-in 4 direction
* Shoot enemies
* Collect enemy health drops
* Collect enemy weapon drops

# Artificial Intelligence

The Overall Enemy AI is Based on One base script (Enemy.cs) that has a comprehensive set of features that each enemy share. Upon the base script, specific scripts have been implemented.

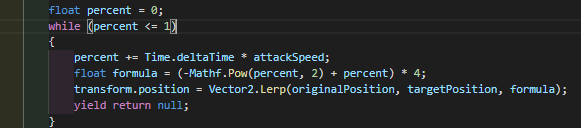
## Enemy

Enemy.cs is the base script for all the enemy AIs (not the boss). The manin features are:

Public: health, speed, time between attacks, array of pickups, drop rate, particle effect. It has to interact with the player, so it will look for the “Player” tag. It has a take damage function that checks how much health has the enemy, if it dead, destroy itself, if not -> evaluate the drop rate between 0 and the number of the drop array and choose a random drop.

## Melee Enemy

This scropt works upon the base Enemy.cs , so instead of referencing a MonoBehaviour, it reference Enemy. This script regulate the specifics of the melee Enemy, that are: StopDistance, AtackTime and Attack Speed. At the start check if the player is alive, than draw a vector between this object and the object with the player tag, than move towards. But, if the Time of the game is greater than attack time, start the attack coroutine. The coroutine works with an IEnumerator that reference the takedamage function on Enemy.cs. Trace a vector from the original position of the object and change with the player.transform position and set the the target position as the player position. And use a while loop with a float between 0 and 1 to calculate the time of the attack, the attack speed and the position.



## Ranged Enemy

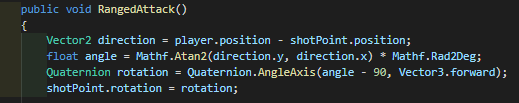
The ranged enemy Ai works upon the Enemy.cs so Monobehaviour has been changed with Enemy.

This Ai has to shoot a bullet when arrives to a certain distance form the player, the bullet will have it;w own speed.

As seen previously in melee enemy we need: stop distance, attack time a reference to the animator (since it has to change animation when attacks), a transform point that works as crossbow, and and a bulle object.

This script will use the same start function of the Enmy.cs but including the animator component.

It will check if the player is alive, if it is, trace a vector and move towards and if the game time is greater or equal of the variable attack time, set the attack animation.



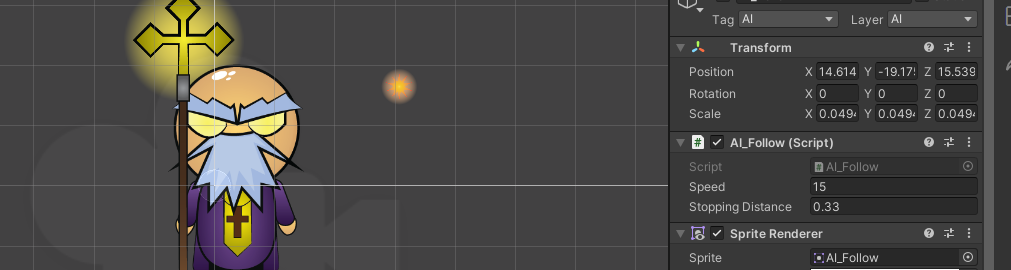
And after instantiate the bullet object.

## Summoner

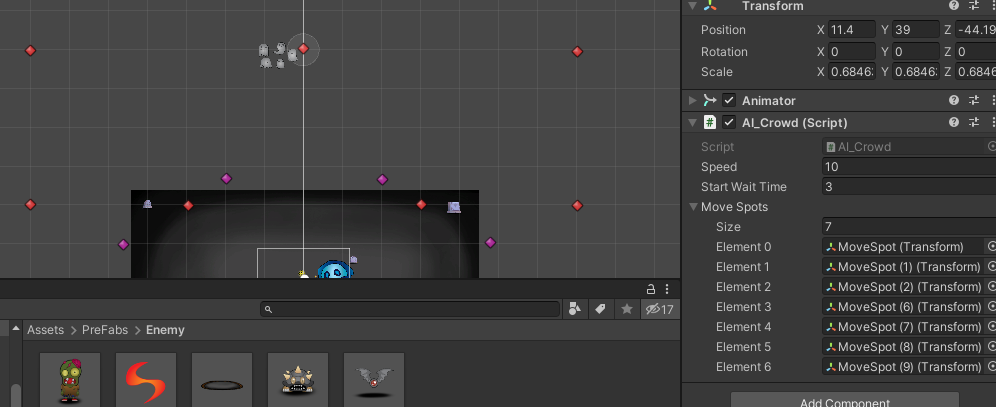
4 public floats were used to tell where the AI can go. It is used also to chose a random position from the map between set min and a max of Y X. Private Animator anim is used to tell the AI to change an animation from idle to moving to summon. Public float summonTime is used to tell how long it will take to summon the selected minion and public Enemy enemyToSummon is used to tell what the summoned/spawn will be. Public override void Start() is used to overrides the start function from the enemy script. Then I still call it with base.start but it also calls the function of creating a random number between the Y and X. After selecting the number it goes to that location and switches animation from idle to move. Private void update first checks if the player is death so if death it stops summoning or moving. Then it checks if the location is given is not reached yet and if is it too far. If yes it keeps moving and still uses the animation walking. If the player reached the location the animation walked called as a boolean “Run” will be stopped and switches the animation to idle. With function summonTime = Time.time + timeBetweenSummons; it tells how long it will take a new enemy to be summoned which also triggers the animation summon. The public void summons also checks if the player is dead so it won't summon the selected enemy to the target position and rotation which is set if the player is alive.



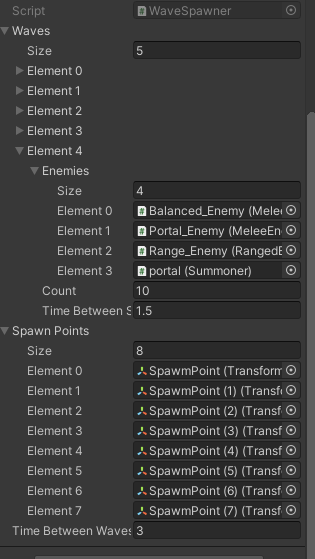
## Follow

Follow has 2 public float variables. Which is speed how fast it goes to the target and stopping distance to tell it how far to stop from the target. The function of the private transform target is used to tell the AI what to follow. The code starts by finding the target with the tag Player. In the update function, it checks if the player is alive by using target != null. If the player is death it destroys itself. If the player is alive the AI can move near the target. If the distance is greater then the number given as the stopping distance the AI can keep moving. With the speed given as variable using Time.deltatime so the speed remains the same on different computers

## Crowd

For the crowd AI, it had 3 float variables which are speed how fast the crowd will move from one spot to another. Wait time and start wait time are the 2 other float variables these variables. Start wait time is used to tell when the wait time to start. The wait time is used how much longer will the AI wait in the location. Created an array for move spots to be randomly selected. The move spots were created with empty game objects with icons and increased the size of the gizmo to see. The function starts by choosing, a random move spot from the arrays. Then the update function tells the crowd AI sprite to move to the location chosen with speed chosen in the float variable. Time.deltatime is used so the movement/speed will be always the same from different pcs. When it reaches the position, it waits the stated time given if not the wait time will be decreased before the start wait time goes. The system repeats itself without stopping.

## WaveSpawner

Wave spawner is used to spawn the enemy AI in waves. First, it starts by finding the game object tag name of Player and starts the coroutine for the wave. IEnumerator StartNextWave is used to tell to until the new wave starts. IEnumerator Spawnwave is used to loop the code until there are more waves but stopes if the player is dead. To spawn an enemy it selects a random enemy from the array called Enemy eniems. The enemy selected then will be spawned from a random spawn point from the function Transform spawnPoints. To start the next waves it checks if finishedSpawning is true if false it continues to wait. When true it checks if the player tag with enemy is to an amount of 0. If it is 0 the function changes to false to start a new wave. To check if there is a new wave the if function (currentWaveIndex + 1 < waves.Length) is used by checking if there is a new wave array by adding plus one to the current array. If there is continues the current wave until the tag with enemy is all destroyed and starts the new wave. If there are no more waves the boss will be spawned as the final wave.

## Boss

The Boss is based on a fine state machine that has two phases; in the first phase the boss will patrol the map following patrol points in a random order, in the second phase the boss will chase the player. He will switch phase and animation set when he has half health. In both of the phases every time he will be hit he will spawn a random enemy.

Patrol.Behaviour.cs is attached to the first set of animations. It has an array of Patrol Points, a speed and an integer that will decide randomly witch point choose within the array length. In order to recognize the patrol points he will lokk for the “patrolPoints” tag and then choose randomly in the array. After choose, he will mpove towards the point and when he reach a distance less than 0.1f, repeat the decision making process.

BhaseBehaviour.cs is attached to the second set of animations. It has a private player and speed. At start find the object with player tag, than if he is not dead, move towards the player position.

Boss.cs woks as “connection” and “controller” of the two states. It is attached to the boss prefab and it has: health, an array of enemy that he can spawn with a certain offset from the boss position, a reference to the animator in order to switch animation, a private variable halfHelath, a particle effect and a scene transition variable.

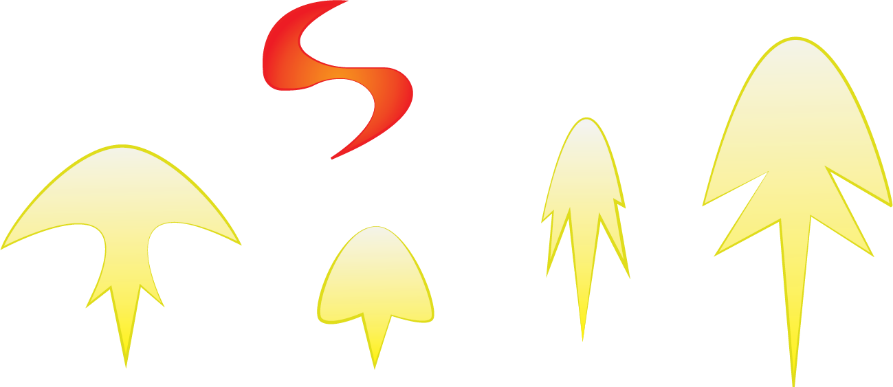
At start define that halthelath means helath divided by 2, define the animation component and define what scene transition has to be made.

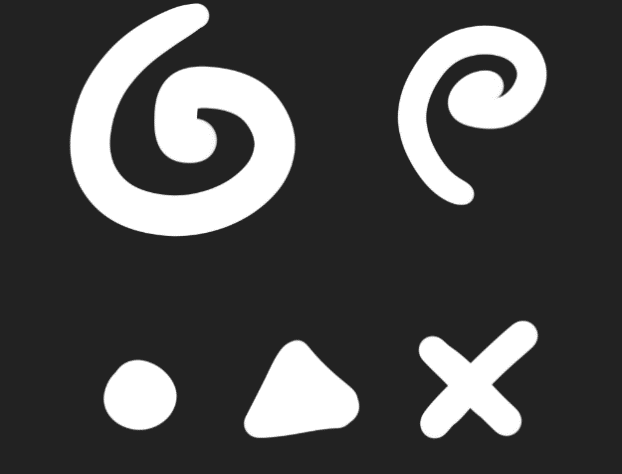
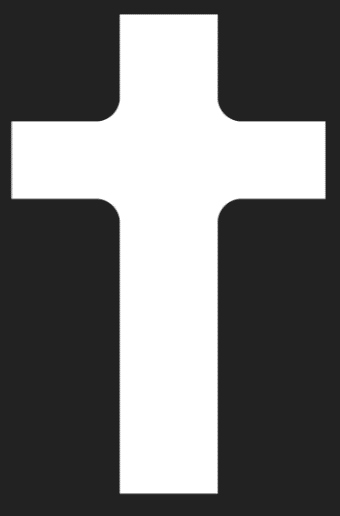
Than if he has half health set pass to the second stage animation, and instantiate enemies when he is hitted.

Than detect collision with the player, recall player component ad use the function takdamage

# Game Assets

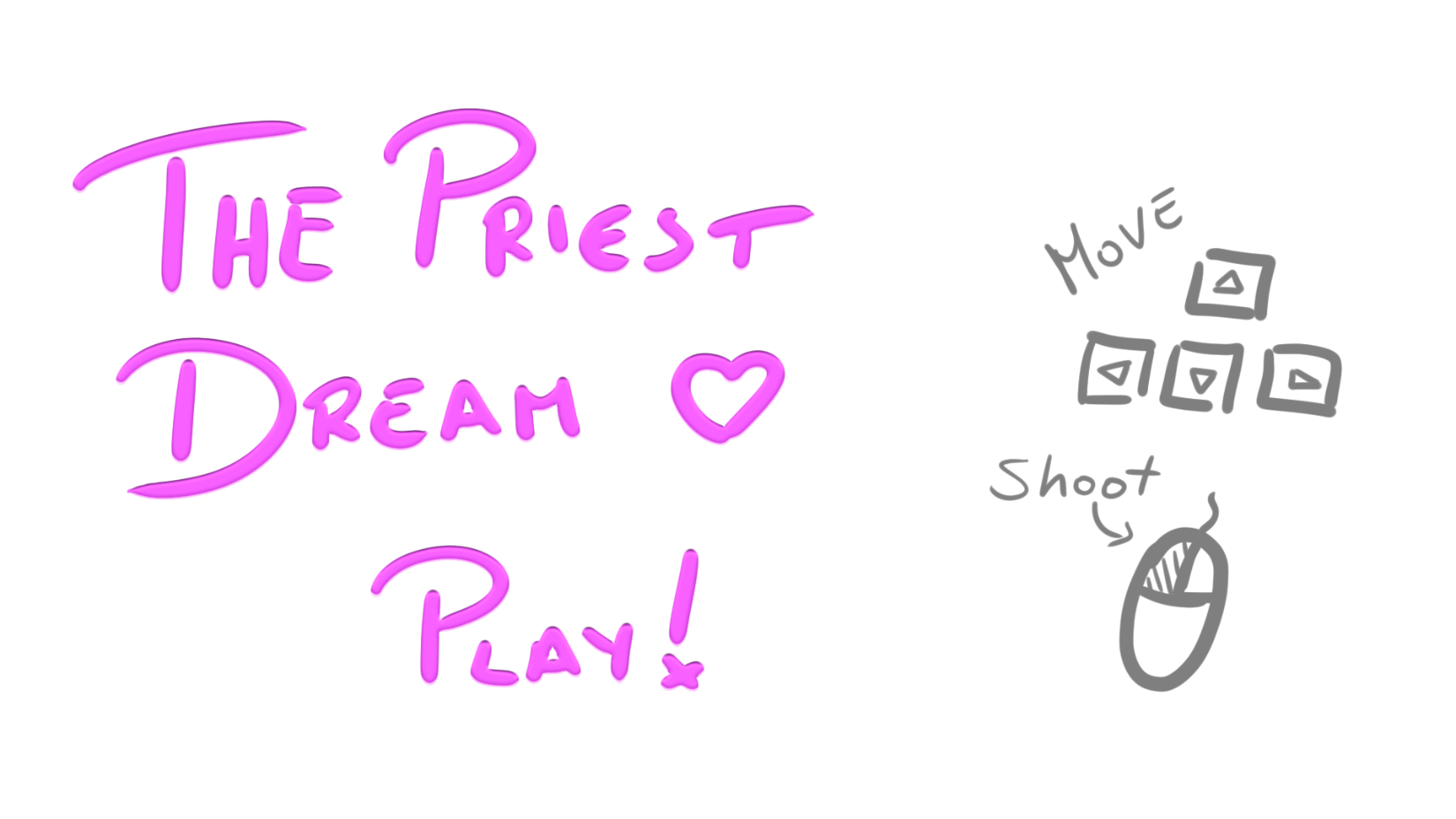
## Projectiles/Particles

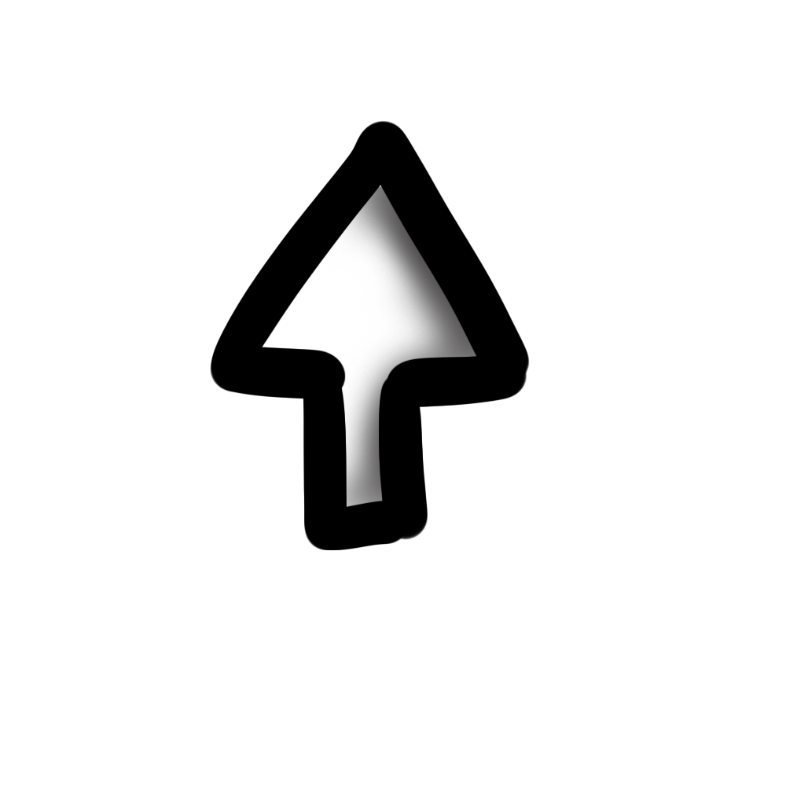




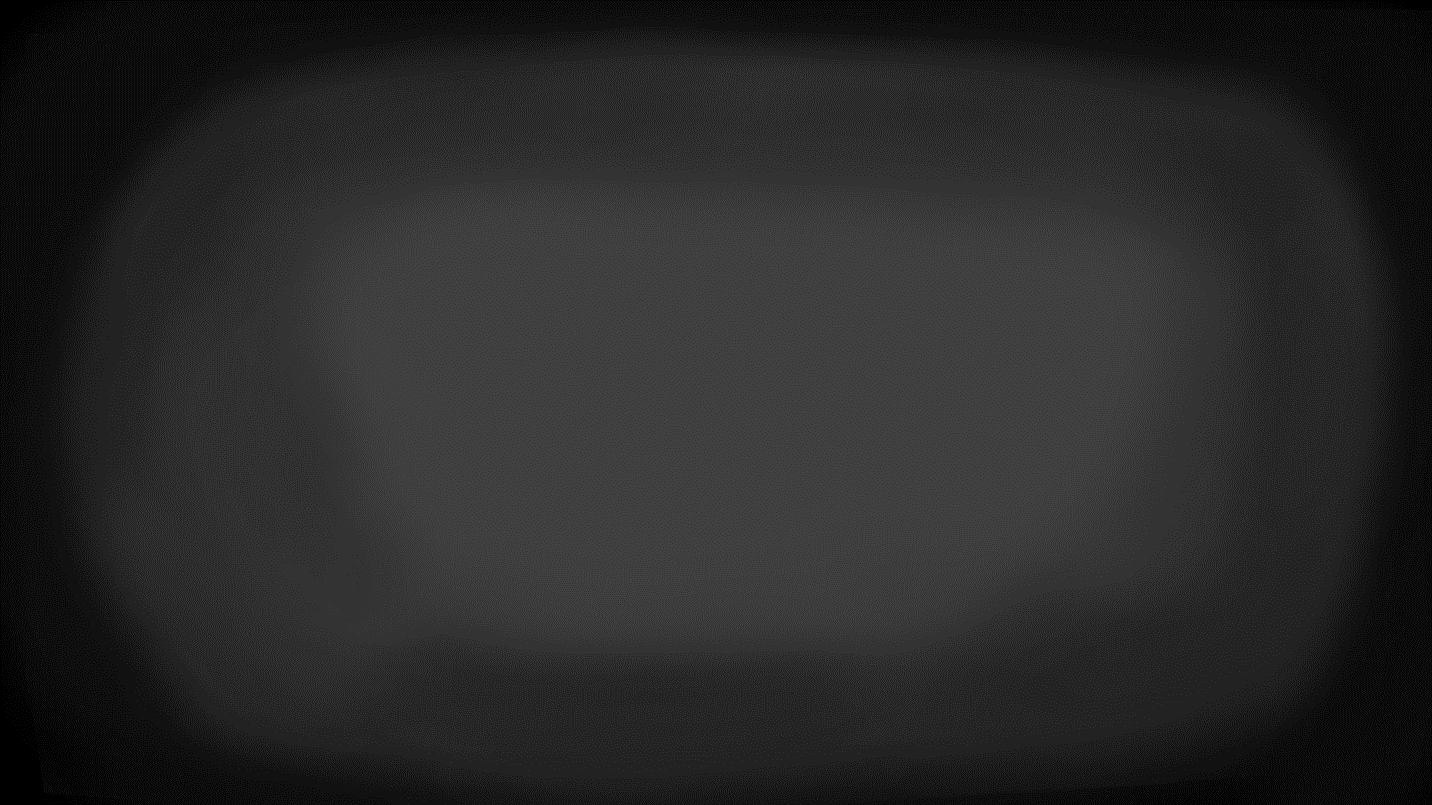
## UI







## Environment





# Test Scenarios

With the help of making a lot of our variables public we could adjust the code there with the required requirements need for that type of AI.

An issue that popped up but with the help of the sir we figured it out why it wasn’t working. The problem was that the wave spawner wasn’t working and we couldn’t find why. We tried to debug the code by using debug.log(“debug”) this function is also used for the other code that we did. It was used to see if the code is being read through the terminal/console. While using the debug.log we found out that the last section of the code wasn’t getting read. We tried switching code places and arranging the code from the start and update function but it didn’t work. Lucky the sir told us that the code was saying that they are still 5 sprites with the tag Enemy. By going through the hierarchy I figured out that the AI crowd children tag was still called enemy because it was used like that before we implemented the wave spawner. Unforntantly we didn’t notice that when we changed the parent of the crowd AI tag and layer it didn’t change the children's tag and layer. By changing the tag of the AI crowed the wave spawner started to work.

Another issue that occurred was with the build. The game worked fine on unity but the build had bugs such as the enemy AIs didn’t damage the player or when health drop they didn’t get picked up and destroyed. After trying different types of builds the bugs stayed there. The sir suggested to change the tag of the player and change the get player tag to MyPlayer. He told us this because he builds the project as a developer build, then he used visual studio debugger which was attached to visual studio to the developer build and debugged it. The debugger was telling that it was not finding the Player component. So after changing the player type and rebuild it again didn’t work. After checking the children of the player body I noticed that they had also the new player tag. I asked the sir if they were needed but he said now cause unity tries to find the object with that tag so it is best having one object with that tag. After removing the tag of the children’s and rebuild it the game worked fine.