Extraterrestrial Occupiers

The Engine

Between Unity and C++, I have chosen Unity. Despite the fact that I know how to code in both, I feel more comfortable in Unity. This is accentuated by the fact that in 48 hours, I needed to prototype a fully fledged game, and using C++ would have been too time consuming, and I it would not have allowed me to fully display what I am capable of. In contrast, Unity offered a plethora of already existing systems, like animation and physics.

The Game

My initial idea was to create a similar game to SpaceInvaders, since it is a type of game I am more familiar with.

The Player

The player moves with WASD/Arrow keys and shoots with spacebar. Its implementation was fairly simple, since Unity's physics engine was used to keep the player within the screen. For the cooldown on the attack, a coroutine was used. It has 4 lives, and every time the players get hit, one life is lost. If the player collides with an enemy, it is an instant game over.

The Enemies

There is only one type of enemy in this game, the mechanic beetle. The two challenges here was having the whole sward of enemies move at unison, and knowing which enemies where able to shoot. To solve the first problem, I implemented an invisible collider that encapsulates all the enemy swarm and mimics their movement. This collider's size may change every time an enemy dies, so from the outside, it looks like the swarm is bouncing from side to side, like the original Space Invaders. To solve the line of sight problem, I used a Raycast2D with a mask, that only detects if there are other enemies in front. If there aren't, it lets the beetles shoot. This Raycast is only refreshed each time an enemy dies, instead of every bullet, in order to consume less resources.

Scoring System

For the scoring system, I wanted to reward the player's speed and aim. This is why both missed bullets (aim) and obstacles destroyed (speed) subtract points. For the programming of this system, I decided to implement a singleton class. Despite the scoring system only being used in 2 of the 4 scenes, both game and score scenes are the ones the player will use the most, and having the scores already loaded really eases things. For saving the highest scores, I created a small save and load class, that works using a binary formatter.

Audio System

The audio system is the other singleton class implemented in the game. Both implementations are done using Unity’s functionality in C# and MonoBehaviours. I used a singletion class

so when switching scenes, if the audio track would be the same, the song does not get interrupted.

Particles and Instancing

In order to implement particles, I created a set of prefabs that spawn when an enemy or the player is hit, and despawn after a couple of seconds. I know resource wise is more costly than enabling and disabling components when enemies die, but I found this way more clean and less obfuscating code wise, and I do not think that the performance will suffer greatly, even with large quantities of particles.

How to Play

When in the initial screen, head to options to adjust the volume using the slider. It will be saved for future sessions. If not, press, on Start and the game will start.

- Controls:
    - AD/Left and Right arrow keys: Move left and right
    - Spacebar: Shoot a bullet every half a second

The objective is to kill all the beetles on the screen. But have in mind, that every missed shot subtracts points from the total score. If the aliens manage to destroy one of your defenses, you will also lose points.

If the aliens manage to be in contact for the player, it is game over.

The aliens also have a chance to shoot a bullet. If this bullet hits the player, it will lose one of its four lives. When the player has no more lives left, it is also game over.

In the score screen, we can see our score from the last game, as well as a list of our top 9 highest scores.

Time:

I spent two days making this game. The first day, I spent around 8h, and the second day, I ended up spending around 11h. My original plan was to do two sessions of 8h each, but I misjudged how messy some systems were going to get, like the score system or having the UI work properly. The final amount of hours dedicated to this is 19h.