Mobile Networked Apps

Individual report

Aaron Reboredo Vázquez

As the other assignments Pablo and I have been working side by side, in this case creating the final game to the final delivery. We have tried to be fare with the number of tasks we should be working on.

For this project, we developed together a working plan to achieve our goals in this exam.

When we started working on the project we didn't thought it was going to be as hard as it was.

Having achieved to build a little prototype for the last of the assignments we were excited to start with this one, but when we started developing new functionalities, the difficulties started to appear.

It has required a good number of builds and testing to continue improving the game until achieve a project without (as the Fallout 76 Bethesda launching report said) "amazing" bugs and inconsistencies.

Having the mandatory 4 on mind, our firsts thoughts were to make something similar to a first-person shooter, or something like that. But we knew it was going to be very hard on terms of development, design and working ours. So, we come up with the idea of a simple shooter game based on the funfair games. We brainstormed a little and we come up with a suitable and reachable game, the duck hunter. We stablished some rules and conventions for the programming and we started to work following the schedule.

I started with the **player logic and input** and the correct **camera management**. I found some difficulties with the setting of the camera for the local player when I tried the game by running a second build using the network and Unet. Finally, I realized that the way to do it was to set active on each local player the camera as true to avoid the no local camera players activate their camera and turn the rendering camera to that one.

For the **movement and correct functionality of the input for the Pc and Mobile** device versions, the documentation has been a good source of information.

It was no difficult to come up for a solution for the player initial **spawning positions**, just followed the mandatory 4 schemes, that o implemented on that moment.

For the **lobby and connection between builds and players network behaviours**: I tried to build a **lobby** scene by myself, but the **Unet** options, especially for the UI display, were not the best, so I decided to use a free asset downable (*https://assetstore.unity.com/packages/essentials/network-lobby-41836)* from the asset store, that is based and use Unet and I adapt and use it to our requirements.

I implemented the **bullets logic**, **bullets behaviour**, spawning and interaction with the enemies. I have also worked on the score system with his network functionalities and appearance on the screen.

At the beginning, a Game Manager was the one in charge to count the points for each player communicating this information to canvas to be shown. But when I started to test the interaction against the objects that should give points to each player I found it was not the correct one. On the non-host client, most of the times the points were not being updated correctly.

I discover that the key was to make each player script manage the information of their points, make those variables been synchronised between clients and give that information to the Game Manager each time the punctuations modify their value.

Also, I found some problems to decide which was the best way to make the **interaction bullet, "enemy" (**objects that give us points**),** specially, in terms of giving the correct player the points, I needed a way to make the game know which was the owner of each bullet.

I decided to make an **Id system**, that allows to give each player a unique Id that is going to be the same in the non-host and the host player built for each player. In this way we can have the owner id in each bullet spawned or shot. After that, the point system was almost ready and was working as I wanted.

Once my team mate developed the **pool system** I adapted the bullet system to use it.

The next thing I was working on it was the **Game Over**, **winning conditions,** state and **messages** that must be shown on each client when the game ends.

Having the Id I had to use the game manager and create some functions which, by using that Id, allowed me to show the properly messages on each client screen by the end of the match.

By modifying one of the scripts of the downloaded package I had mention, I added the option to show the “Back to lobby” box that was implemented by them, giving the option to return to the lobby once the game was over.

I have also worked on the **UI** and I find and programmed a way to change the **colour** to de **players** and the **point´s panels** and **bullets** depending on the colour selected by each one on the lobby scene.

My team mate and I worked on the rest of the tasks together (making the final builds, testing and equilibrating the game), and spent, more or less the same number of hours on them.