First presentation report

Bounty Hunter

March 2020

1 Introduction

Starting a game (almost) from scratch is not quite an easy task. Thanks to the game engine Unity, we didn't have to create all the 3D engine and all of its modules like physics or movement. Using Unity means that you have a lot of useful tools at your disposition to create your game as you planned it. During this first work period, we were able to reach almost every single of our goals listed in our book of specifications. Unfortunately, the online part (that is an essential part of *Bounty Hunters*) is less programmed than expected. Fortunately, we already know how to apply everything we have learned during these two months to make *Bounty Hunters* a proper online game.
Since we are forming a team, we had to find a way to share each modifications that we are doing to the other members of the group. To do so, we decided to create a GitHub repository that contains our whole project, except the unity libraries and the files that are auto-generated by the code editors and Unity itself.

2 Lancelot Martin

2.1 Engine

My task was to re create a version of the *Brawl Stars*'s graphic engine, which is essential, because without this part, the online part couldn't be done so that's why I have made it as fast as I could to enable my group-mates to work as well. To do so, I had to create an example of 3 vs 3 gamemode map that will be shown only for the first presentation. To enable our game to save different maps, I have decided to create a special file format that the mapReader I've coded can translate a text file into a unity 3D plane matrix.
This algorithm first has to open the desired file, following the right disk path. At this moment, one still has to write a specific path, unique to his computer's disk architecture, but I will implement very soon the feature that will allow the mapReader to get the map's file from the project's repository. After opening the file, the algorithm has to transform each lines of the text file it opened into a 2D array or a 2D matrix. After doing so, it is applying a simple pass-through double loop to get the value of each cell and generate the corresponding plane texture in-game. The shape of the map doesn't matter, the algorithm will build the array with as many elements as there is in the file (for example the map mustn't be square shaped). The map's file format is the following :

- '.' represents a ground cell
- '2' represents a wall
- '3' represents a bush
- '4' represents water
- '5' represents a player healing zone
- back-slash n represents a new line in the 2D array
- to make the map creation easier for the person who does it, the algorithm leaves the choice to the creator to separate each item with a ',' or not.

These values might not be definitive and we will add some new cell-types, with different properties. Right now, the cells only have a different appearance and they don't have special properties. Once all the planes/cubes are initialized in the Unity engine, it handles their absolute position on the screen compared to the camera position.
The camera has been placed in an almost isometric angle, I had to make several tests to make it feel the same than *Brawl Stars*.
In Unity, you have to create different "materials" that you apply to planes or cubes in order to insert images on the different cells of the map, so I had to quickly create some square images that represent each of the different cell type cited above using *Paint*. These images are temporary because we are going to implement some new and better graphics in the game. However, these graphics will probably used for the low graphics option available in the options menu.

After the map loaded properly and the proper materials were applied on each 3D model, I had to create the player sprite. In order to do this and knowing that the player's 3D models are not ready, I decided to represent the player with a cylinder that has the water's material. After that, I had to implement the player's control, to enable it moving in the map. To do so, Unity has a pre-implemented player control interface so I just used it.

To conclude the *Bounty Hunter*'s engine I handled, I had to make the camera follow the player but only in the z axis, for the 3 vs 3 gamemodes, because the 3 vs 3 maps are rectangle shaped and the width of the whole map fits in the screen. To do so I remembered a formula from my past developer experience, when I was programming games using scratch.MIT.edu, which is the following :

$$C_z = \frac{(Player_z + (C_z * 3))}{4} - 1$$

This formula enables the camera to follow the player on the z axis with a slight smooth effect. To prevent the camera from going all the way too far on the z axis and leave the map while following the player, I had to put bounds on two z point of the map, above and below. If the camera position is outside of those bounds, the camera position is set to the proper bound.

2.2 GitHub

As a group, we decided that I will be the Git repository code master. This means that I have to supervise all the changes that the group are bringing to the project. For example, each group member needs to create a new branch when they want to bring changes to the master branch. After pushing their branch to origin branch, they must create a pull request that I have to accept or deny depending on if the changes are valid or not.

Really understanding Git's operation is only given to a few people on this planet, but I think that being in charge of the Git's part of this project really helped me to understand much better Git.

3 Maxence Crouzy

3.1 Musical part

Since we started this project, I wanted to create the music. I have been playing the guitar for 6 years now and this project was the perfect occasion to improve my level, test my capacities and link my passion to my work. This is one of the main reasons that made me the sound designer of *Bounty Hunter*. Even though creating music is not really difficult, I struggled to find something that would represent the "spirit" of the game.
When I started to create samples, I had to find the good tone, the good effects (distortion, Wawa, fade-in, fade-out, etc.) and the good style! Finding the appropriate style of music was the hardest problem I faced in the music creation. Our game starts from scratch, which means that I have no physical concept of the game look or no aspect at all of the character design. This is where the teamwork appears. We thought about the character design and *Bounty Hunter*'s world design in general, so that we could all have a guiding line to follow and have something coherent and harmonious.
Meanwhile, I was trying to create a main menu and a settings menu, which are both operational now.

To record my tracks and modify them, I used thee well known software *Audacity*, which is not the best one, but it is free. It took me some time to understand how it works, thankfully, reading Microsoft's Documentation for each new notion in the TPs taught me how to behave in front of a new software or language .
After months of trials, I finally got something that would correspond to *Bounty Hunter*'s world. The final music is little bit jazzy and entertaining as well. The main menu music is composed of :

- A main bass line that is present throughout the whole song.
- A guitar chords progression which is composed of 6 different chords: C, F, D, G, A and B; all played in power chords to get a nice amount of strength and distortion in the sound, without having saturation. This part last for 32 bars.
- A guitar harmonic progression which last for 24 bars.
- A snare every 0,5 beat, and when the guitar riff hits, we added some drums with a 1 and 2 and 3 and 4 pattern (which means adding drums to the snare, the drums completing the spaces between the snares except for the first half-beat of every bar, which is silent).

After showing this part to my group, we were pretty satisfied of the main menu music. It corresponds to the "cartoon" aspect of the game, sounds fun and entertaining, while having a tiny bit of depth. We can easily imagine little cars moving around with this music.

After finishing the main menu music, I started to create the fighting music. This one had to be more aggressive, but it also had to stay in the artistic direction we established with my group mates.
It took me a massive amount of time to figure it out, because I always wanted to have epic fighting songs, which was impossible in this case. I needed something as intense as cool. I suddenly had the idea to implement a guitar line that reminds a honk song. The fighting music is not finished yet, but I can describe its current main components:

- A main bass line that is present during the whole song.
- A succession of guitar chords that sounds like honks, they "sharpen" the sound and add a slightly amount of intensity and depth to the song. It is battle time, not teatime!
- No drums, it would make the song too epic for *Bounty Hunter*'s environment.

As the music is not fully implemented yet, I am still thinking about the best ways to make the music effective only in the menu section, and to play the fighting music whenever the game is launched.

3.2 Main Menu

I oversaw the creation of the player interface. To begin with, I had to make a Main Menu. When it looks easy to build a panel with text on it, it is quite difficult to understand how Unity works. Moreover, it was my first time using it, so my experience on a framework like Unity was nonexistent. We first agreed on a version of Unity, so that we all have the same version for the development of *Bounty Hunter*. By doing a lot of tests (for almost 2 weeks), I understood the basics of Unity, and its capacities. The main point is that I learned how to apply scripts to Unity objects, which allowed me to create my own scripts.
To find a way to start my menu, I asked the students from last year and searched on the internet. I also read some of the Unity documentation in order to use built-In functions. As *Jet Brain's Rider* did not work with Unity, I had to install *Microsoft Visual Studio* and to link it with unity to use Unity functions.
To find a nice font, I searched on the Unity asset store, but the only correct one I found forced me to restart my project from the beginning due to a bug in the asset (it was TextMeshPro), which is annoying because it has apparently on several awards.
The main problem I faced in the Main Menu was how to go from a scene to another and how to launch the game. As Lancelot was having trouble with the conversion or a .txt file into a 2D array, I just worked with an empty scene to see if my scripts worked.
To go from the Main Menu to the Settings Menu, I found the idea of using the function "GameObject.SetActive" to make the Main Menu inactive and set the Settings Menu active. It was a good idea, because I did not have to move

the camera position as I wanted to do at first (Unity Documentation is really helpful).
To launch the game, I had the idea to apply a function to the play button that would launch the scene at the specified index in the Build settings (it is currently 1 but will possibly be 2 later in the project because I have not done a character-selection menu yet).

3.3 Setting menu

This one was harder. In this menu, I decided to implement 4 different functionalities:

- A full screen toggle.
- A volume selector.
- A quality drop down.
- A resolution drop down.

Each of these guided me to problems, except for the full screen toggle, which was the easiest one. I only had to call an implemented Boolean function called "isFullScreen" when the player presses the full screen toggle.
The volume selector was weird because I didn't understand why I had problems at first, but it was because I created an audio mixer based on a integer value, so it was not clear and the selector was almost useless. I swapped the function to a float value which goes from -80 to 0 (otherwise I had struggle, so I made it as Unity uses it).
The quality drop down brought me confusion, because I had to change the normal settings of Unity (we only needed 3 quality settings, Low, Medium and High). Going into Unity parameters was not very easy, but I found the way to make it work through my scripts and by reducing the number of qualities in the parameters to the same number as I wanted. This also allowed me to set the basics graphics at High.
Finally, the resolution drop down was the hardest one. I had to create an array containing every resolution possible depending on the computer the player is playing on. The point is that I had duplicates of resolutions, and some of them were not appearing when the first number was the same (for example I could have 720x480 but not 720x576).
It took me a whole week to resolve this problem. The duplicates were appearing because of the refreshing rate of the screen, so I decided to display the refreshing rate after the resolution. I had to use a specific library ("System.Linq") to make it work. I obviously needed the "UnityEngine.UI" library to access my drop down.
I am still thinking about the other parameters I might need to include further in the project in this Settings Menu.

3.4 GitHub

This is the one of the most problematic part of this report. Before coming to EPITA, I had never used Git. It is a strong tool, but it takes so much time to learn and understand the commands! Even though I understood some of them through the TPs we did in class (git clone, git add, git commit or git push), using them for a real project was different. I was always scared of deleting files or merging weird folders.
Thankfully, Lancelot knew how to use Git better than us, so he managed to create a repository and teach us some useful things about Git in general. Furthermore, he is in charge of the merges/pull requests so that we avoid many mistakes.

3.5 Goals for the next presentation

For the next presentation, all the musics and the sound effects should be implemented in the game, as well as a character-selection menu. I think that the HUD will be at an advanced point too (it should be displaying health, ultimate charge etc.).

4 Gabriel Glazman

As our game project Bounty Hunter is a MOBA (Multiplayer Online Battle Arena), it is of course fundamental for it to have an effective online multiplayer, on which the whole game is based on. Our game will mostly have different game modes, in which two teams of players will fight against each other.In the 3 vs 3 gamemodes, each team will be supported by three different players; therefore, there will be more gamemodes, that won't have to be 3 vs 3 but will be including a maximum of 10 players.

My goal was then to create a server, capable for every game, of connecting the players to it, making the server send and receive data from the clients (unity clients), consequently making the game possible to play online with other users.

But my very first question that came to our mind was : how should we make the online multiplayer ? There are several ways of connecting different players in a video game. To begin with, I first thought about the most simple and commonly used way of implementing a connection between two players in a video game : the P2P approach (non-authoritative peer-to-peer multiplayer). Here is a simple scheme describing how the P2P implementation works :

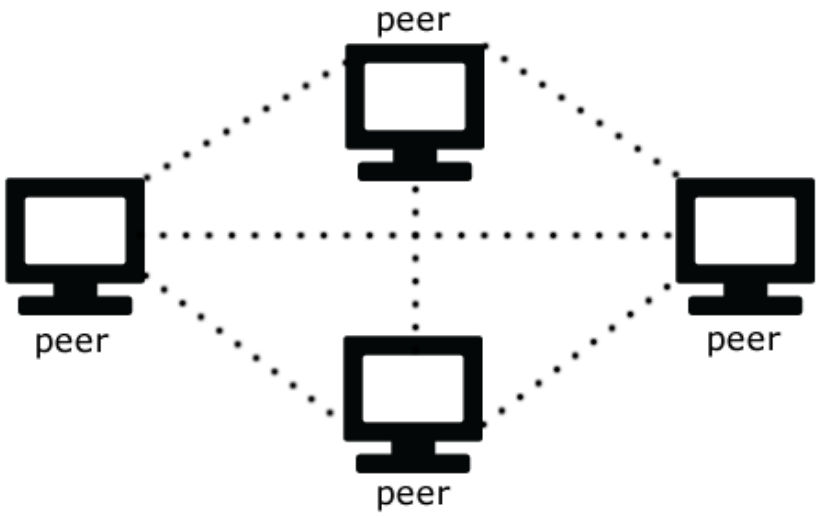

Figure 1: P2P operation

As you can see, all players are interconnected, with no first connection to a remote server, handling the data.

Secondly, I also thought about the second most common way of implementing an online multiplayer ; the server-client based multiplayer. Here is another simple scheme depicting how the server-client implementation works :

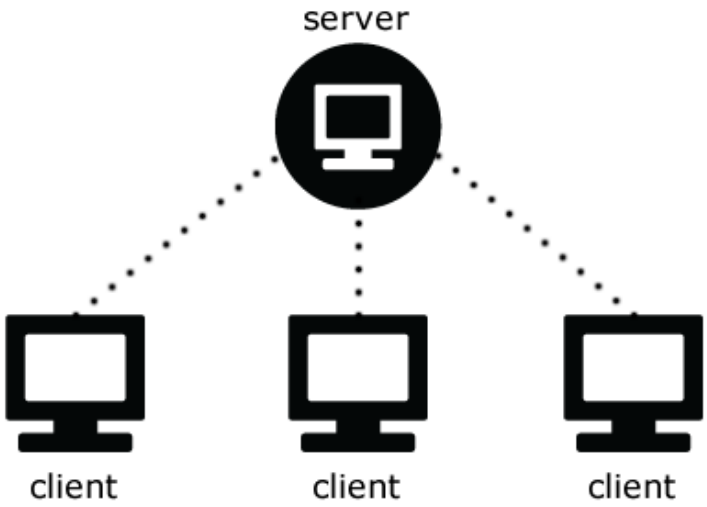

Figure 2: Server-client based multiplayer

But after consideration and looking at all the pros and cons of both server-based and P2P-based online multiplayer, we chose the server-based multiplayer. The three main reasons of this choice were :

- first of all, a P2P multiplayer is much harder to implement than server-client based one
- second of all, security is a very important thing in every online game. In P2P, cheating can be easily done and is hard to prevent, as in a server-based model, cheating can be avoided easily
- last of all, latency is usually much greater with P2P, and a bad internet connection can influence the game for the other players too. Otherwise, with server-based, if the server has a solid connection the latency can be extremely low and a player's internet connection never affects another's game

Our main goal was further on to create a server, on which *Unity* client could theoretically connect, send and receive data. Of course, the server needs to handle the data, making the game possible to be played online, but this wasn't our first objective.
So, we could make a server, but how do we make a server ? What kind of server do we make ? That was the main questions we then asked ourselves. After

documenting ourselves on the matter, we found that we could easily create a server on local-host in C Sharp using two classes inside the System.Net.Sockets namespace. These two classes are Socket and TCPClient.

But first, what is TCP ? TCP (Transmission Control Protocol) is one of the main protocols of the Internet protocol suits. It is a standard that defines how to establish and maintain a network conversation through which application programs can exchange data. It completes the Internet Protocol (IP) and suited together is commonly referred as TCP/IP. TCP/IP is designed to make networks reliable, with the ability to recover automatically from the failure of any device on the network.

Here is a simple scheme of how TCP/IP works :

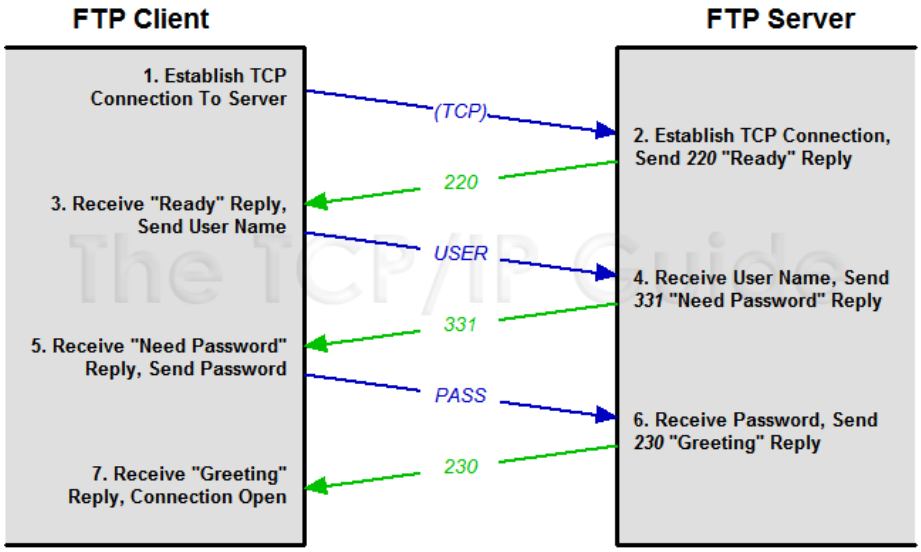

Figure 3: TCP/IP operation

This is this method that are server will use to "communicate" with the different clients and this is the method I've started to implement on my local Git branch.

5 Helena Ataker

While my teammates were doing their parts, I was working on creating maps.I have done 2 maps example for the project *Bounty Hunter*. I did them both considering a $57 * 57$ 2D array which will be used in the Battle Royale mode and some other 3 vs 3 gamemodes.
I had to think differently in order to create these two maps so that they won't look like each other but still have the same concepts and challenges and give the player a nice experience. I had to decide on different concepts for each map and apply them on the 2D array in order to see if they would work.
For the first map i decided to have an easy map for players to play. I did not put a lot of challenges so that it will be considered easy compared to other maps we will create. For this map I tried to give space to the player so that they can discover the format of the game easily without struggle and learn the game better.
On the second map I filled a lot of spaces so that the player will have to think more in order to figure out how to end the game successfully. So the second map is definitely harder compared to the first one because it has more challenges to complete.

6 Conclusion

During this first period, the project *Bounty Hunter*'s development advanced as expected. Each of our group brought what they did best to the project and we are very determined to get our game to the top of all MOBAs. For the next presentation, we will have done everything that is announced in the book of specifications.