Multiplayer video game for public spaces

Víctor Rufo Tena

Final Degree Project
Bachelor's Degree in
Video Game Design and Development
Universitat Jaume I

July 18, 2023

Supervised by: José Vicente Martí Avilés

Acknowledgments

To start with, I would like to thank David and Andriu for their interest in this project and thank José Vicente for his guidance.

I want to thank my parents for their constant support during these months as well.

I also would like to thank Sergio Barrachina Mir and José Vte. Martí Avilés for their inspiring LaTeX template for writing the Final Degree Work report, which I have used as a starting point in writing this report.

Abstract

Raid and Go is a two dimensions (2D) raid-style turn-based online video game that aims to be locate to public spaces, such as parks or iconic city locations but it could be used in long-distance public transport as well. The work covers two large development blocks, one consisting on the creation of a system of multiplayer rooms where players can enter and leave whenever they want, and the other that represents the complete development process of a game. Academically, this document consists of the final degree project report of the Video game Design and Development bachelor´s degree at the Jaume I University.

Contents

Chapter 1

Introduction

Contents

When the job proposals were presented by the tutors, the truth is that I placed this one quite high, possibly because days before I caught a flight and the entertainment was playing with other passengers the typical competitive games present in these means of transport and it seemed interesting to me the idea of creating a different game away from them. [9]

There are a lot of fans of RPG video games (Role Playing Games) where the player controls a character, or several, in an adventure of medium or long duration. In particular, what likes the most about these games is their fighting style, which almost always uses a turn-based and rarely finds itself with a multiplayer system. This is the reason why it was found interesting to make a turn-based game adapted to the multiplayer room system that the project requested.

What attracts the most for lovers of these RPG games is the art of the game, that the character sprites need to be pleasant and differentiated, as well as that the functions that are added to the combat system have to be fun and comforting. Therefore, it has always been considered very important to dedicate a lot of time to it.

1.1 Work Motivation

The main motivation to develop the project was to create a multiplayer game different from the common ones in these public spaces. In addition to the fact that I had never faced developing a multiplayer game before and I wanted to learn from this experience.

Because as commented, there was an experience some time before the selection of the project´s topic.

The truth is that the idea of trying to merge two elements that are not often seen together was interesting, that are the cooperation in these public spaces games, which tend to be more competitive, player against player.

And as a personal proposal it was decided to adapt one of the most favorite video game genres, RPGs, to the theme of the project. So the motivation to work daily on it and make everything look as good as possible came without even realizing it.

1.2 Objectives

The main objective was to have a presentable game that was as polished as possible with the time and resources available.

Regarding the individual goals of each section of the project:

- **Multiplayer room system:** One of the large blocks of the project, possibly the most important. The goal was to build a room network where players could enter and to leave space for another. Adapted so that it is completely invisible and random for the players in order to not be able to tackle the same room repeatedly. As the game is cooperative, it was sought that once accessed to a room, both players should help each other until the end of the match.

- **A correct multiplayer game:** Making a multiplayer game was something new, so it would have been impossible to have guessed the amount of time that would have to be put into this to get everything working properly. The objective was to ensure that all that appeared on the screen was going to be shared between the two players, for example the character chosen by the partner or to be able to visualize the actions that he carried out. This was what required the most time but what gave the most happiness when achieved.

- **Art:** As mentioned before, one of the aspects that attracts the most attention in these game is a pleasant art that is friendly to the users. Something that was had in mind when carrying out the main one, the characters´ design, that they had a clear differentiation and wearing elements that made them unique.

- **Consistent menus:** The objective was to have the minimum essential menus for a video game of these characteristics, a button to access the game and galleries that show the art of the characters.

- **Gamefeel:** RPG combat is nothing if it is not fun, so it was decided to implement action buttons that perform the options usually found in games of this genre. Also, something had to be placed to create interest in the users, in this case the appearance of monsters that must be defeated to later be unlocked in their respective gallery as trophies for the player, which will be saved even if we exit the game for future sessions.

1.3 Environment and Initial State

This project could be divided into two blocks, such as video game and multiplayer system, which in turn could be subdivided into game art, the game itself, the room system and the connection between the players.

Since everything related to multiplayer was a bit disconcerting, it was decided to start developing the game, and then build on it the multiplayer system.

So the starting point was the essential art, leaving for the help of the free assets if required and the development of a 2D turn-based game in Unity.

To finally understand how to develop a multiplayer system in Unity.

Also remark the meetings every two weeks with the tutor José Vicente to discuss how the project was progressing.

1.4 Keywords

- **Multiplayer game**
- **Public spaces**
- **Communication between players**
- **Raid-Style**
- **Turn-based combat system**

CHAPTER 2

Planning and resources evaluation

Contents

This chapter is the most technical part of the work. Here the planning is explained as well as the software and resources that have been used.

2.1 Planning

In this section, it is explained how the time in this work it was thought that will be invested divided by tasks. The tasks are shown in a table where they reflected what type of work was and the number of hours estimated for its development and with a Gantt chart shown below. Note that the table is described as an image because Google Tools [13] have been used to create it, since its appearance is better.

TASK	TYPE	HOURS
Initial installation and configuration	Research	5
Study the work´s topic and references	Research	20
Study the different options referring the connection between the players	Research	10
Game Design	Design	50
To draw the UI assets	Art	20
To draw the Characters	Art	35
To make the animations	Art	5
To make a 2D turn-based video game	Programming	10
To create the multiplayer system	Programming	90
Game feel, optimization and testing	Programming/Design	20
To write all the documentation	Documents	35
Total Hours		300

Figure 2.1: Tasks

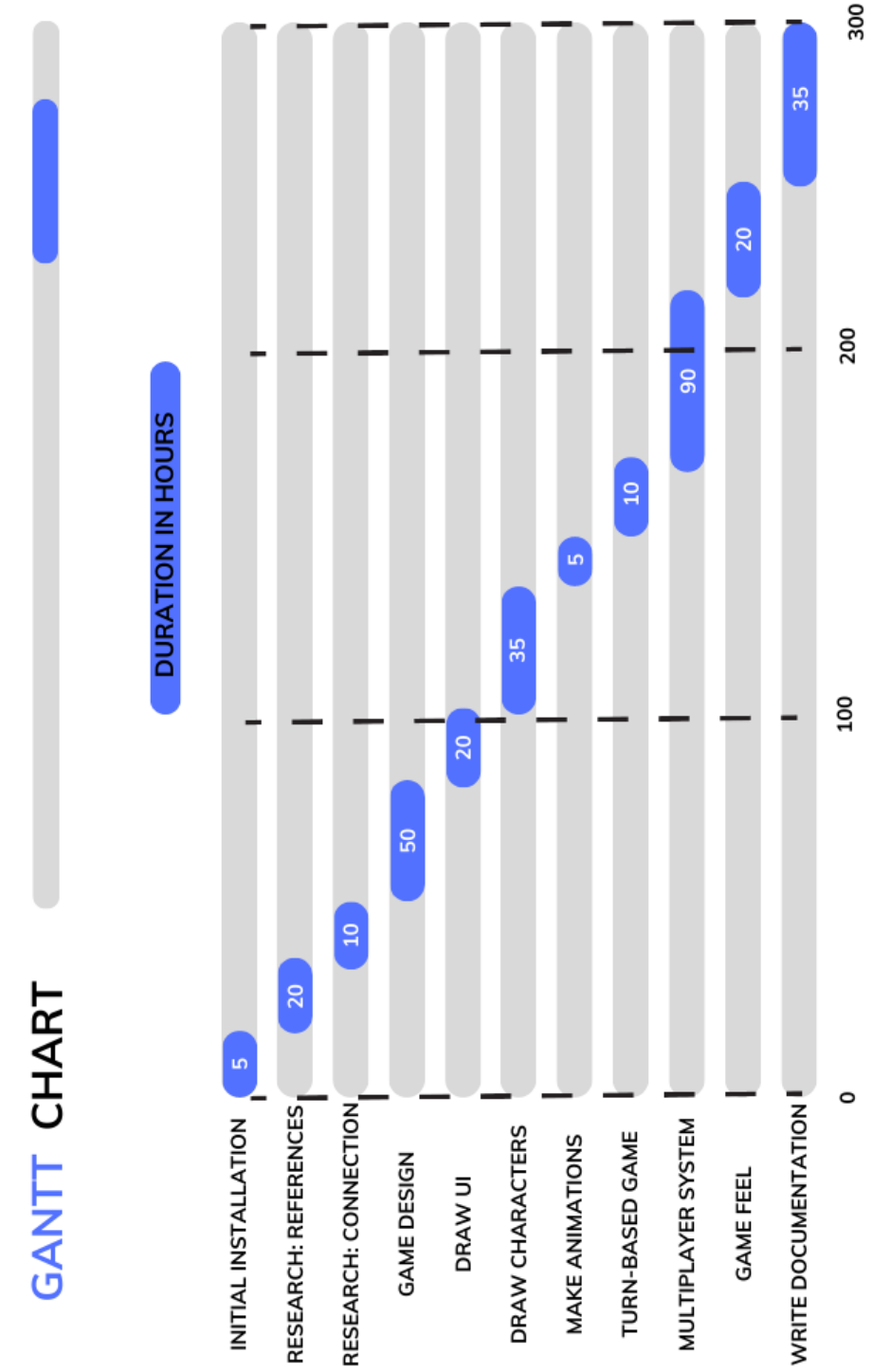


Figure 2.2: Gantt Chart

2.2 Resource Evaluation

Here, an analysis about the theorical amount of money and time that the project would cost is made. Due to the fact that both programming and artistic tasks have been worked on, the cost of one and the other will be calculated separately to later make an approximate sum.

On the one hand, in this project the number of hours used in programming and designing was 135 hours. With this in mind, in Spain a junior programmer earns 8.39 euros per hour on average, so its final cost would be 1132,65 euros. [20]

On the other hand referencing the art, 60 hours were worked on. About the costs, in Spain a settled 2D designer earns 14,79 euros per hour on average, so its final cost would be 887,4 euros. [19]

Speaking of some expenses that are usually overlooked, such as the cost of the residence (without taking into account food costs or extra expenses) or the Internet connection. A studio apartment for rent in Castellón de la Plana (Spain) costs 450 euros per month on average, so in three months the figure would rise to 1350 euros. [17] For an acceptable speed Internet of fiber type, it costs about 50 euros per month in the main companies, so during this time 150 euros would go to have access. [23] In addition, the electricity consumption of an entire property is usually 30 euros per month, therefore in three months it would mean 90 euros that are added to the price. [25]

With these quick calculations and ignoring the previous research work, focusing only on the more technical development of the project, the total cost of the work would be 3610,05 euros. Now, it is important to differentiate between hardware and software:

- **Hardware:** Even if the used computer does not have the most actual components it is still pretty decent and has a price of about 700 euros, with a useful life of two to four years:
  - **CPU:** Intel Core i7-8550
  - **GPU:** Nvidia GeForce MX150
  - **RAM:** Intel 8GB
- **Software:**
  - **Unity(Free):** It is a cross-platform game engine developed by Unity Technologies to create 2D and 3D games, even VR games. [27]
  - **Photon Engine(Free Version):** Allows projects to be online and can be easily related with Unity. [7]

- **Visual Studio(Free):** It is an IDE developed by Microsoft that is compatible with almost every highly used programming language. It is really helpful in association with Unity because it shows directly errors and other features. [22]
- **Krita(Free):** Digital drawing and illustration program that is perfect for creating 2D art and being able to export it in good quality. [26]
- **Overleaf(Free):** This online LaTeX is the one that has been used to write this report. [16]
- **Google Tools(Free):** Some tools offered by Google have been really useful for the development of the project, such as Docs for the formalization of the documentation and Drive for storing useful files. [13]
- **FakeYou(Free):** This free program has been used to create some audio tracks that have been used in the work. [24]

With all the costs already displayed, the total cost of the project can be calculated.

Being the cost of residence and everything that involves it of 3610,05 euros, and the Hardware section of 700 euros, since in terms of Software programs with free functionalities have been used entirely, the development of this project costs 4310,05 euros.

Chapter 3

System Analysis and Design

Contents

This chapter presents the requirements analysis, design and architecture of the proposed work. It also shows the design of the GUI (see figure 3.1), known as the game user interface.

The functionality of the project will be described, with all the possibilities that are offered to the user. To do this, in the different sections screenshots will be shown to help understanding and varied graphics of the possible actions of the player once inside the game will be presented.

Figure 3.1: Image that shows the Game User Interface

3.1 Requirement Analysis

Having a job to do can also be seen as a problem to solve. The problem to solve here is the development of a video game, one with a free theme that could be used in public spaces and with a multiplayer system that allows the players to connect between them by rooms.

Raid and Go is the particular solution. It proposes a game that is different from those located in current public spaces and that complies with a solid system of multiplayer rooms.

As soon as the game is opened, a title screen (see figure 3.2) will be displayed, with a message inviting the player to click anywhere on the screen to access the game. Once clicked, the main menu will show up (see figure 3.3) where three options appear, the Play button and two galleries, one for the playable characters and another for the enemy monsters, called Bosses. Since the game is supported by a funny character design, the galleries are really useful in this video game style.

Figure 3.2: The title screen

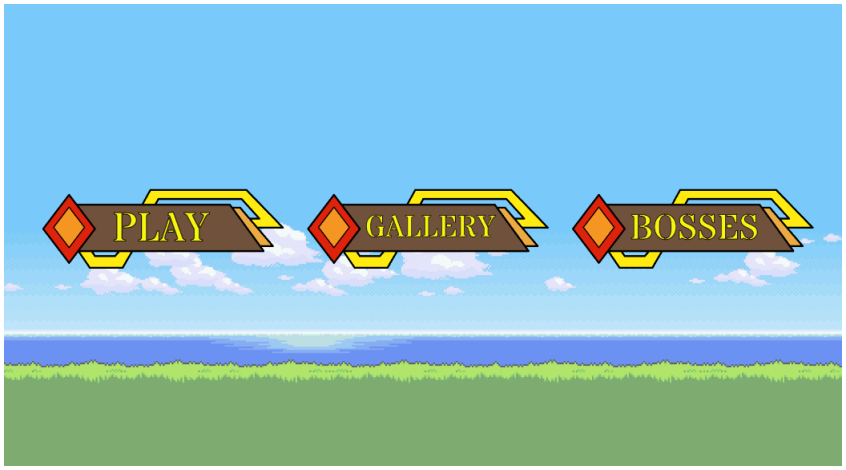


Figure 3.3: The main menu screen

In the first gallery called Gallery (see figure 3.4), the four playable monsters are shown so that when clicking on the one the player is interested in seeing, a description of it and a larger image will appear (see figure 3.5). It includes an exit button to return to the beginning of this section of the menu as well.

Figure 3.4: The playable monsters gallery screen

Figure 3.5: The description screen

If it is the first time accessing the Bosses gallery, they will appear with a dark art, this says that they have not been defeated yet (see figure 3.6). Once achieved this, the image will be replaced with a full color one that will be kept for future play sessions (see figure 3.7).

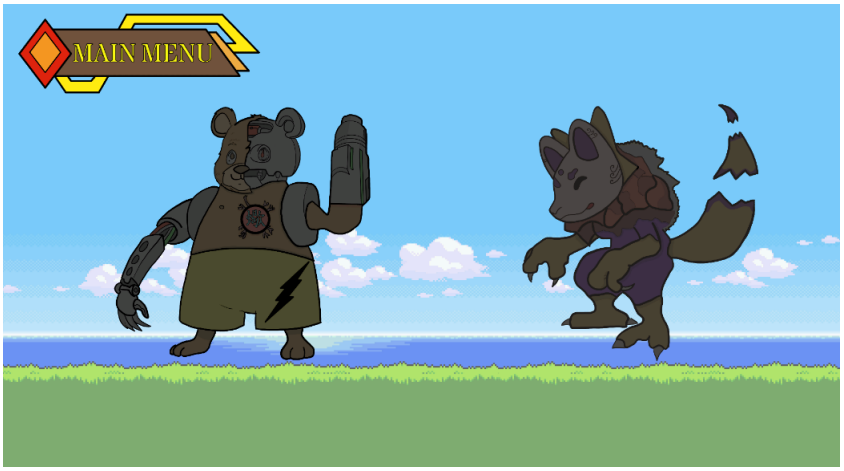


Figure 3.6: First time accessing the bosses gallery

Figure 3.7: Accessing the bosses gallery after defeating one of them

And finally the Play button where the player will access the screen to choose character and select whether he want, to enter a room created by another player or open a new one to play alone while waiting for help (see figure 3.8).

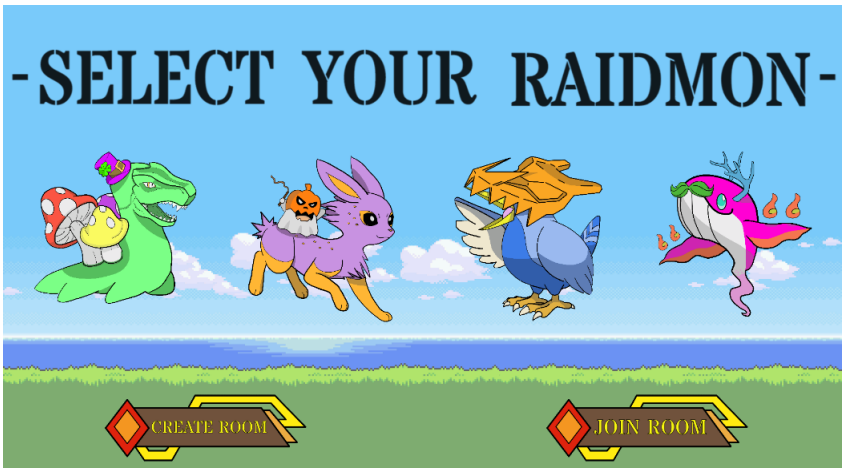


Figure 3.8: Selection screen

Once inside the game, the player will be able to choose the option to be performed from among three possible ones by clicking on them in his turn (see figure 3.1):

- **Attack:** The player´s monster will attack the enemy with his current attack stat, lowering the opponent´s health.
- **Heal:** The player´s monster will recover as much life as the value of its healing stat.
- **Boost:** This will randomly raise a stat from the player´s monster, these can be:
  - **Raise our attack:** The damage that the player´s monster will deal to the enemy will be increased.
  - **Lower the boss´s attack:** The damage that the enemy will deal to the player´s monster will decrease.
  - **Increase our healing value:** The amount of health restored will increase for the rest of the match.

Must be careful since the boss can also do all of this.

The objective is clear, to make use of the previous actions to lower the boss´s life to zero while trying to keep the personal character alive. In case of being defeated, the raid will be terminated with a defeat message and a return to the title screen will happen to try again if the player wants to. The same will happen if winning but with a more lively message.

It is appreciated to comment on how the game works in case of entering multiplayer as well (see figure 3.9), where two players will have to cooperate to defeat the boss.

Figure 3.9: Ingame screen with two players

A player must create a room first, if the join button is clicked and none exists, one room will be created where the player will be alone fighting while waiting (acts as the Create button). The boss to fight in that specific room may be different from other players, due to it chooses between the possible two randomly.

By clicking on join, if there are several rooms, the system will match with one randomly, something that is better than simply showing the list of all the available ones, for system speed and for gameplay quality.

Once player two accesses, player one (who created the room) could have decided to go alone, so the boss will already have a decrease on his stats or life, just like the partner player (see figure 3.10).

Figure 3.10: Player two joins an already started game

There is also the possibility that a player does not want to play. In many cases the best move is not to make a move, so the partner can play alone without any problem.

Finally, it is important to comment that once inside the game there will be no way to leave the room until winning or losing. It is an act of cowardice and disrespect for the other player to let him down if the match does not go as expected.

3.1.1 Functional Requirements

A functional requirement defines a function of the system that is going to be developed. Let´s see these requirements:

R1. The player can enter the Main Menu by clicking on the Title Screen (see table 3.1).
R2. The player can enter the characters gallery (see table 3.2).
R3. The player can enter the bosses gallery (see table 3.3).
R4. The player can exit the galleries by pressing the return button (see table 3.4).
R5. The player can enter the character and match selection screen by pressing Play button (see table 3.5).
R6. The player can enter the game by selecting a character and a multiplayer room option (see table 3.6).
R7. The player can attack the boss on his turn(see table 3.7).
R8. The player can heal his character on his turn (see table 3.8).
R9. The player can raise a stat of his character by pressing the Boost button on his turn (see table 3.9).

R10. The boss can perform the same actions as the player on his turn (see table 3.10).

R11. The match can become multiplayer at any time (see table 3.11).

R12. The player two must meet the game as it is (the information of the started match) (see table 3.12).

R13. A player can take no action in combat (see table 3.13).

R14. The system will be able to save the boss defeated information for future play sessions (see table 3.14).

Now the descriptions of these cases are defined:

Input:	Clicking anywhere on the Title Screen
Output:	Go to the Main Menu
The player can click whenever and anywhere he wants to be redirected to the main menu. This is a change of elements that appear in the same Unity scene to save resources.	

Table 3.1: Functional requirement «CRED1. Enter the Main Menu»

Input:	Clicking on the GALLERY button on the Main Menu
Output:	Go to the Characters Gallery
The player can click the GALLERY button while is in the Main Menu and whenever he wants to be redirected to the playable characters gallery. This is a change of elements that appear in the same Unity scene to save resources	

Table 3.2: Functional requirement «CRED2. Enter the characters gallery»

Input:	Clicking on the BOSSES button on the Main Menu
Output:	Go to the Bosses Gallery
The player can click the BOSSES button while is in the Main Menu and whenever he wants to be redirected to the non playable characters gallery. This is a change of elements that appear in the same Unity scene to save resources	

Table 3.3: Functional requirement «CRED3. Enter the bosses gallery»

Input:	Clicking on the RETURN button while on a gallery
Output:	Go to the Main Menu
The player can click the RETURN button while is in a gallery and whenever he wants to be redirected to the Main Menu. This is a change of elements that appear in the same Unity scene to save resources	

Table 3.4: Functional requirement «CRED4. Exit the galleries»

Input:	Clicking on the Play button on the Main Menu
Output:	Go to the character and match selection screen
The player can click the PLAY button while is in the Main Menu and whenever he wants to be redirected to the selection screen. This is a change of elements that appear in the same Unity scene to save resources	

Table 3.5: Functional requirement «CRED5. Enter the character and match selection screen»

Input:	Choose a character and a specific match on the selection menu
Output:	Enter the game
The player needs to select a character and a room option to be sent into the match. In the case of not selecting one of them, nothing will happen. This is a change of scenes in Unity.	

Table 3.6: Functional requirement «CRED6. Enter the game»

Input:	Click the attack button during your turn in a match
Output:	Decrease enemy´s life
The player can click on the attack button during his turn to decrease the enemy´s life with his current character´s attack stat. In case of doing it out of turn, nothing will happen.	

Table 3.7: Functional requirement «CRED7. Attack the boss»

Input:	Click the heal button during your turn in a match
Output:	Recover an amount of your character´s life bar
The player can click on the heal button during his turn to recover the amount of life that stat currently has. In case of doing it out of turn, nothing will happen.	

Table 3.8: Functional requirement «CRED8. Heal your character»

Input:	Click the boost button during your turn in a match
Output:	A random stat of your character will raise
The player can click on the boost button during his turn to increase one of his stats randomly. In case of doing it out of turn, nothing will happen.	

Table 3.9: Functional requirement «CRED9. Raise a stat of your character»

Input:	Be on the boss´s turn and waiting his action
Output:	The boss will do a move depending what decided and conditions
If the enemy is on his turn, depending on the amount of life remaining he will perform one action or another around a subset of actions. In the case of more than a half health, he will attack or boost, in the case of less than half, he will heal or attack. The boss does not attack or perform actions out his turn.	

Table 3.10: Functional requirement «CRED10. Boss´s turn»

Input:	Another player enters a current room
Output:	The partner appears and participates in combat
If a player chooses the option to join an already created room and it exists, he will be transported to it. Clicking this option without a created one, will create a new room.	

Table 3.11: Functional requirement «CRED11. Switching to a multiplayer match»

Input:	Player two click on the join button and there is an existing room
Output:	Player two enter the match with the current information on it
When player two enters an already created room, he find himself the current course of the game. For example the current boss and partner´s life and the stats that they have at the moment. Player two´s character starts with everything basic.	

Table 3.12: Functional requirement «CRED12. Share information with the new player»

Input:	The player does not click anything during combat
Output:	Nothing happens
The player can do nothing, for example because he wants to wait for his partner and nothing will happen in the game.	

Table 3.13: Functional requirement «CRED13. Take no action in combat»

Input:	Exiting the game
Output:	Save the current information
The player can exit after defeating a boss and the information will be saved for future sessions.	

Table 3.14: Functional requirement «CRED14. Save the information for future play sessions»

3.1.2 Non-functional Requirements

Non-functional requirements are requirements that impose restrictions on design or implementation such as restrictions on design or quality standards. These are properties or qualities that the product must have.

R15. The game design will be unrealistic 2D and attractive to the player.
R16. The GUI will be simple, that does not bother the player and easy to understand and remember.
R17. The gameplay will be easy and fun.
R18. It should encourage to being played in multiplayer.

3.2 System Design

This section presents the logical and operational design of the system to be carried out. For this, some diagrams have been added:

- **Case of use diagram:** Helps ensure the correct system is developed by capturing the requirements from the user´s point of view. In this case, all the options that the player has once inside the video game (see figure 3.11).
- **Class diagram:** That serves to visualize the relations between the classes that involve the system. In this case, it shows the relations between the classes presented in the combat phase, once it is in front of the enemy (see figure 3.12).
- **Activities diagram:** Shows control flow and object flow, with special emphasis on sequence and conditions of that flow. With a sample with a start point (open the game) and an end point (ending a match) (see figure 3.13).

- **Interaction diagram:** As a graphical representation of the communication between objects and actors of the system. Showing how the dialogue panel behaves, which informs the player of the state of the game (see figure 3.14).

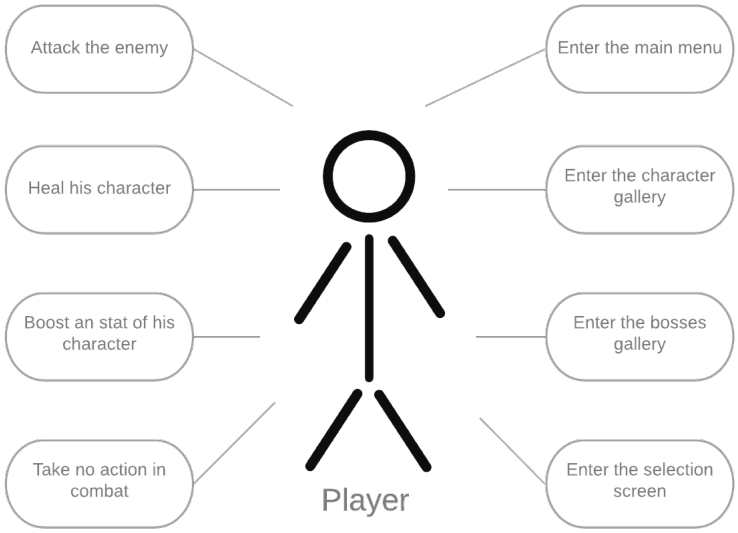


Figure 3.11: Case of use diagram

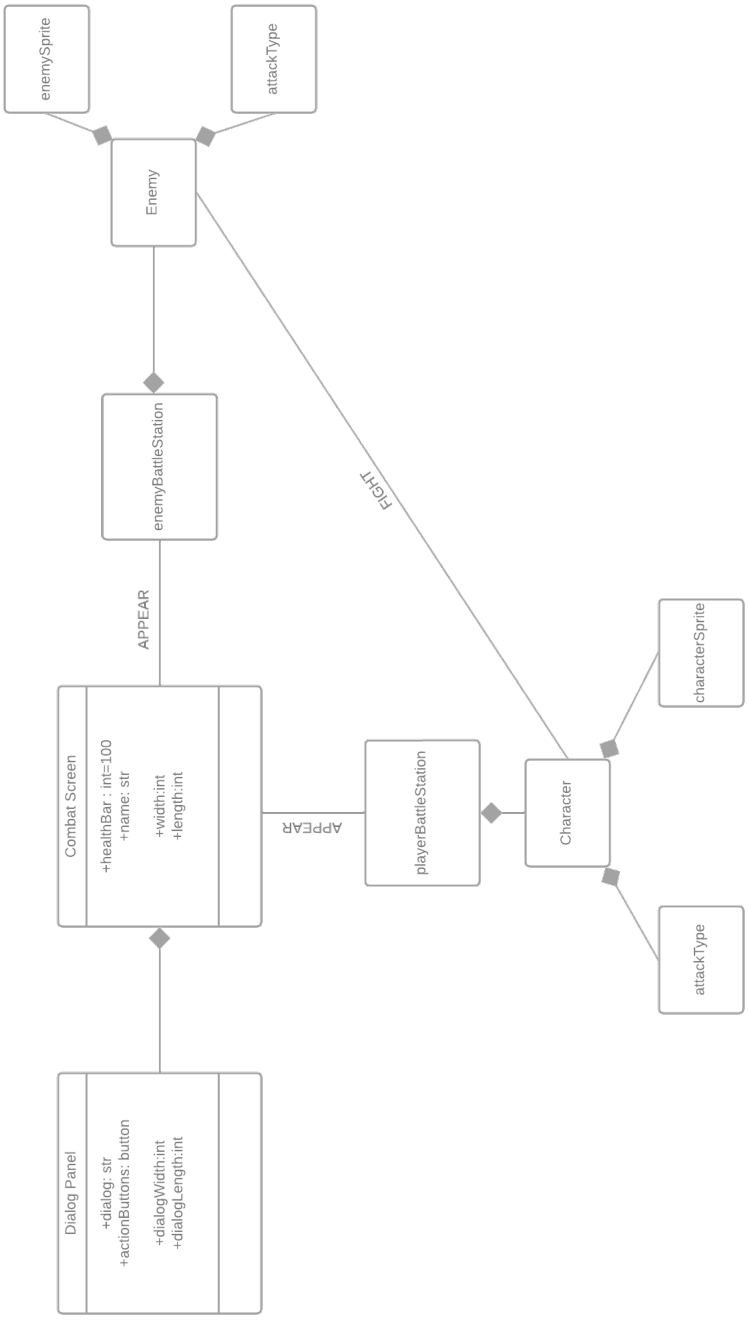


Figure 3.12: Class diagram

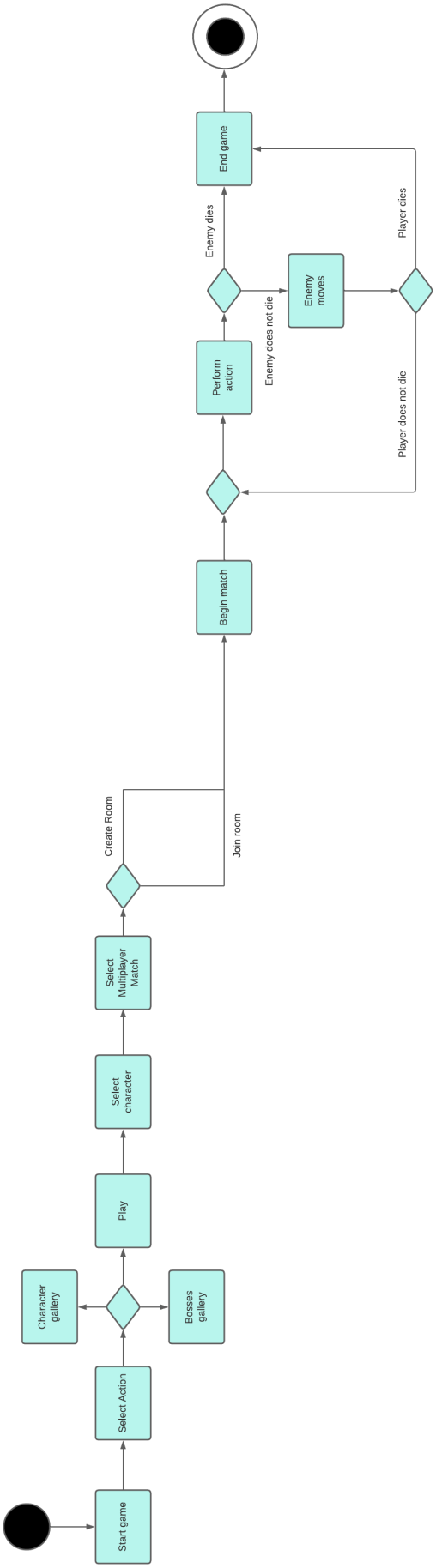


Figure 3.13: Activities diagram

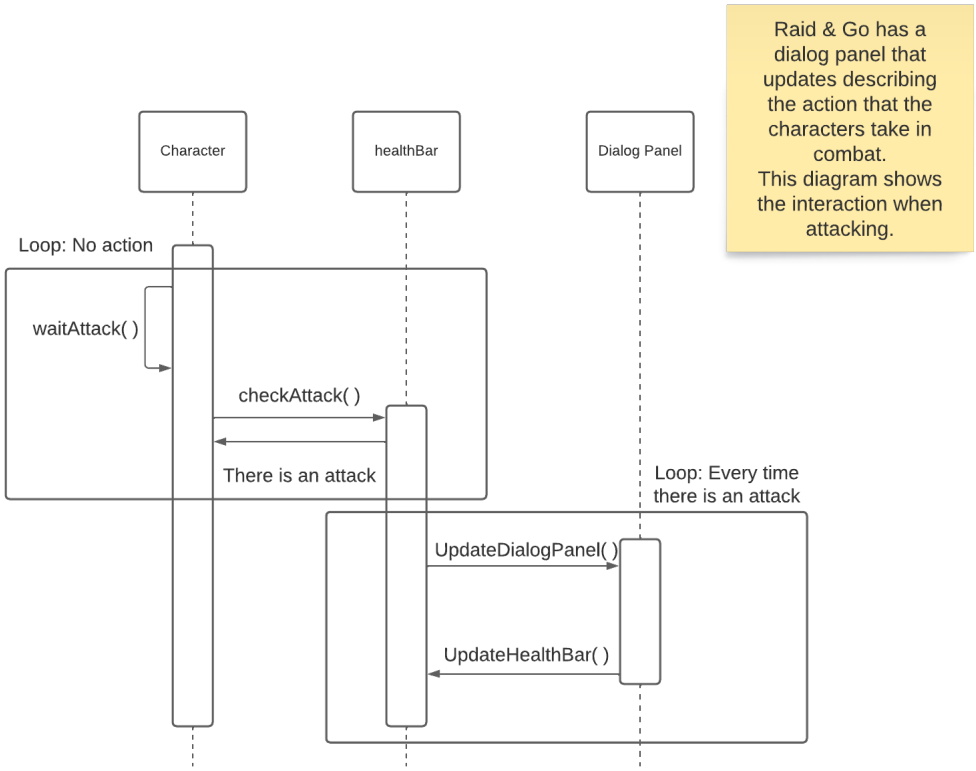


Figure 3.14: Interaction diagram

3.3 System Architecture

This section describes the minimum architecture a PC must have in order to be able to run the game. These data is taken from the Unity documentation. [28] Note that the project has been developed with Unity2D engine, specifically the 2021.3.15f1 (LTS) version.

- Operating system:
  - Windows 7 SP1+
  - macOS 10.12+
  - Ubuntu 16.04+
- CPU: SSE2 instruction set support.
- GPU: Graphics card with DX10(shader model 4.0) capabilities.
- It is obligatory the use of a mouse or touch panel for playing.

3.4 Interface Design

The GUI is really simple and referenced by the classic RPGs (see figure 3.15). Being the main objective to show the art that, as has been commented, must be pleasant for the player. This art not only includes the design of the characters, which is clearly the main one, also the bases where they stand, the status boxes located on their sides and the dialogue panel.

The background must be simple so that everything is perfectly visible, being in classic RPG´s a completely matte background, which is the one chosen for the project.

Continuing with the order mentioned at the beginning of the section, the art of the characters and the bases are completely inanimate, that is, they do not make any changes during the course of the game.

On the other hand, the status boxes of each character show his name and his amount of life, which varies depending on whether he is attacked or healed. And the dialogue panel where the action buttons that the player can press on his turn are located and that also shows the course of the game at all times, for example the action taken by each monster.

Figure 3.15: *Vision Raiders Entertainment* RPG combat sample

CHAPTER 4

Work Development and Results

Contents

This chapter is an explanation of how the project has been developed since his start until the end of it. It also includes an assessment of the results and how some ideas were changed during this process. And how others can be treated for convenience in the future.

4.1 Work Development

In this section it will be discussed what has been done and the decisions that have been taken. Much of the information for the realization of this project comes from articles and tutorials on YouTube [14] that will be cited as they arise. This includes the implementation of certain functionalities in Unity.

Now proceed to comment the development of the project following a chronological order.

When understanding the type of project that was proposed, all the thoughts were about trying to find a different game capable of be settled in public spaces. Being all the common ones Tetris [6] or Angry Birds [18], among others.

It was at this point that it occurred to choose to develop a competitive player versus player card game. But the idea that the second player could join a game started where

he trailed by an abysmal distance is not fun. So it was decided to discard the competitive idea for a cooperative one.

The truth is that cooperative games have a bad point that is shared by the most important point of them, and this is that each player, wanting or not, depends on the other, so it was interesting the idea of making one that allowed a player to win without depending on anyone.

This idea was mixed with one of the favorites video games players genre, RPGs, from which the combat style was taken. This is how the idea of Raid And Go would be born, a 2D raid-style turn-based game that drank from the great classic Pokémon games [10], due to its very nice looking appearance.

Once discussed with the tutor and giving green light, the research work started, trying to get all the resources ready for when decided to enter into the project programming.

From this research, it was extracted that making a functional multiplayer game is really complicated, and also, since it is something that is not learned on any subject, it was decided to leave it for the last.

As a personal objection the development of a video game can not be started without its minimal art done, so it started with drawing the characters with their normal and combat stances, the HUD, the logo and the different buttons and texts, in that order. Immediately below are samples of all the art made. The character´s normal stances can be seen in figure 4.1, the HUD art in figure 4.2 and their combat stances in figure 4.3.

Figure 4.1: Characters representation art

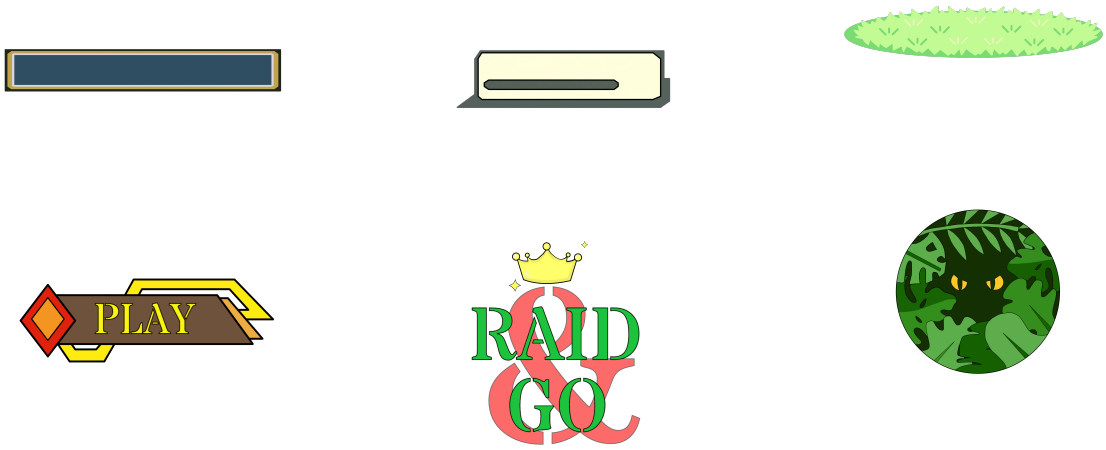


Figure 4.2: HUD assets art

Figure 4.3: Characters combat art

When the minimal art was done was moment to open Unity and began to design a 2D video game with a turn-based system. With the intention to later link it with a multiplayer room system.

The development of the turn-based game ended without any problem, taking as reference a simple tutorial [4]. First up was the placement of the HUD in Unity´s 2D project space. This includes the status boxes of the characters, with a text that is the name and a slider with the amount of life, the dialogue panel with the buttons and the art of the bases. And then the creation and staging of the player prefabs, with their own characteristics such as the life, base damage and current healing.

With the scene built, it was time to develop the turn-based system, that means that the game changes between some states: start game (make appearance of the elements), player´s turn, enemy´s turn and end game.

The next step was to make game state changes happen. Which was consistent with developing a basic turn-based game. At the start of the game it will ask if everything is in place, so it will go to the player´s turn state, once a button is touched, it will wait a few seconds and it will go to the enemy´s turn, which again, will wait for a few seconds and it will go back to the player´s turn, until one of the two has zero life, and then it will go to the end game state.

The state change was done, so it was time to make the button press happen what was requested, lower health on attack button and restore on heal. And most importantly, that the dialogue panel will be updated with each action, friendly or enemy, changing its displayed text every time something happens. Or what is the same, change its content within each function in code.

At this time the turn-based game was done, just needed to polish it while working on other parts of the project, and proceed with the multiplayer system.

It was at this point when saw that aesthetically making the game for two players was the right option, the first thought was to make it for three or even four. But this is not a problem for future revisions, since it could perfectly be adapted for more, only that the perspective of the game should be changed so that the art would be seen correctly, neither very small or with rendering errors.

Once reviewed the information that it was thought that will be useful for knowing how to make a multiplayer network, result of the research work, it was realized that practically none of it was worth, so looking for new one was necessary. One of the problems with creating a multiplayer game is that few people dare, again, because of how expensive and complicated it can be, so finding useful information is often impossible.

The main problem encountered was that most of the options to create the multiplayer system for rooms were not free, with high monthly charges to keep servers active that could not afford. In addition, making the connection with Unity was very complex.

Luckily a tutorial was found [12], one that would help to understand how to create a multiplayer system in Unity. In a free and apparently simpler way.

This tutorial gave as a starting point the use of the Photon technologies for multiplayer, using its free features of having a server with the maximum capacity of 20 simultaneous people, so it is perfect for this type of project (see figure 4.4). The first and most important thing was to link this newly created server with the Unity´s project in question (the 2D turn-based game) in order to use the online features offered.

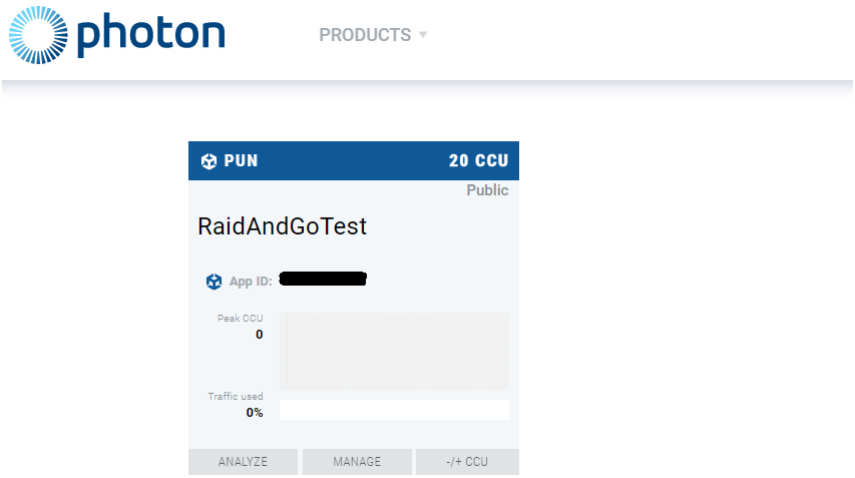


Figure 4.4: Raid and Go´s Photon project link

Once the server offered by Photon and the project to be provided with an online system were linked, the tutorial continued to leave an application that allowed several players who were in the same room to see the same information. This information was about being able to witness the movement of the other player´s avatar on our screen.

To achieve this, Photon offers features that dictate what should happen depending on what our multiplayer system is looking for. For example, this time we are looking for a room system that allows one to be created on the server, and if there is a space, allow a player to enter. This is summarized in two functions granted by the system: create and join room.

These functions gave the possibility that two users were in the same place within the server, but obviously still did not see anything on the screen. This is solved by providing the objects with impunity when passing through the server, that is, that this object is visually shared at all times by all users who are in the same room. This is just another feature that had to be added to every asset.

For this project, the creation of a multiplayer room system was requested. Where players could join and leave continuously, with sharing the state of the game between the players who joined.

Put like that and being completely honest, this tutorial only served to create the room system. For our project, everything else was practically useless.

Note that the connection to the server is made as soon as you access the game, so that the user will have a guaranteed place to play. In addition, it is necessary to have access to a public network, as the game is intended for public places, it is assumed that it will always be available.

The next step was simple: to create the menus and to link the click action to Play to transport players to the game screen.

Buttons are classic Unity elements, to which a function is associated so that when are pressed something happens, in this case, changing what is seen on the screen.

Creation of the menus proceeded without any problem, then next step was to make the turn-based game multiplayer and share actions between the players.

This was a very difficult task due to the number of problems that a multiplayer game entails, where the information seen by all the players present in the same match must be correct.

These problems that had to be faced in the development of the adaptation of the game to multiplayer were several, and as commented there was not much information online that could be useful so the Photon documentation [8] was to be resorted many times and then test new features.

Starting with the biggest problems, the first of all was to make appearance of the partner´s character. This entails several actions that must occur simultaneously thus many errors occurring at the same time.

Let´s remember to enter into context that when creating the room the game must be one against one, to later become an asymmetric two against one, so the appearance of the first player´s character must be displaced to make space. So the solution was to ask the system if the game was for one or two players, in a loop to know what the position should be and if it is the correct one. It seems like it is a simple problem but what about the information that the joining player loads, what would be?

Answering the previous question, it was really hard. The information that was loaded when a second player entered was simply duplicating all the elements on the screen, some of them in the wrong position.

The solution was to clean all the duplicated as soon as accessing, otherwise errors would be seen on screen that do not want, and then leaving the ones that are on interest or do not bother.

After working on it, finally the information seen for the two players was correct, that is, the same. If one player attacked, the other could see the enemy´s health decrease and it could be seen which was the partner´s chosen character. The turn-based game was already multiplayer, once a player defeated the boss, the game ended for both players, and when the boss defeated a player, he disappeared, it could be said that it was perfectly connected.

Next was to ask what information wanted to share, that is, for both players to see it when it happened. The most interesting and indispensable would be the attacks, knowing when your partner is attacking and when the boss is. The truth is that once the previous problem was solved, which was quite hard, these came out rolling, also with clear complication due to the fact of not knowing how to program for multiplayer

system and having to consult functions and techniques in the documentation many times. The resolution method was similar to eliminating before entering elements that do not interest.

Then was time to embellish the current state of the game, when attacking there was no feedback, just like when player two appeared, it had not any warning to the user who was in the room.

The animations were taken from a free Unity animated attack pack [1], which seemed to be made to the millimeter for the project because of how well they fit together.

And there was also the idea of showing on the network the moment a new player joined, to notify the owner of the room, again, extracted from an asset of the Unity Store. [3]

Again it seemed that everything related to multiplayer was going to fail when trying to implement it. A rather frustrating visual bug appeared where player two´s pre-join animations would all flash on screen at the same time. The fix was to place a new remove of all past room items before joining.

Despite the errors that had to be resolved, the game had information shared and visible to both players now, which was amazing and very comforting.

With the spirits quite high from achieving so much it was decided to leave the remaining menus done, as the galleries and the title screen. Also to make a research for free music and sound for the game, most of which are copyright-free remixes of well-known songs. [2]

Also to find free backgrounds to place them on the title screen and menus. All of them taken from a free image bank. [5]

It was time to add sound feedback to each action, this are when attacking, healing or clicking on the menus. It was decided to do it through a sound management system included in the scene that would activate the corresponding sound depending on what happened in the game. All these sounds came from the Freesound page. [15]

Now the game already looked like something, it had a menu system, each action had sound feedback and the multiplayer system worked correctly. Just missing a few personal touches.

This next work block could be grouped in the post match, what should happen after finishing one?

First of all, when finishing the game, the player should be returned to the first screen of the application, in this case the title screen, and save the boss trophy in case of defeating him.

Added to the End Game state that upon reaching it, it would wait for a few seconds, a return prompt would appear, and then a scene change. Boss saving is done with Unity´s PlayerPrefs storage, which allows this information to be saved for future sessions.

After this a full game test was necessary, and that´s when a new problem arose. At the end of the game, the players were not disconnected from the server, so (although it was not true) it was still stored that they were in a room, so once again the Photon documentation was the salvation, with a function that allowed them to disconnect players of choice, allowing immediate reconnection.

To improve the gameplay, it was decided to add the Boost button that would make each match funnier, entertaining and varied. This button would access the statistics of each player and would increase or decrease the values, up to minimum value in the second case to prevent the game from becoming unplayable (a character having zero attack for example).

It was also necessary to improve the intelligence of the boss, since at this moment he was only dedicated to attacking. Therefore, the same instructions that the user had were used to adapt them to be made by a NPC or uncontrollable character. This was done with a random system in the case of a condition. The later being the amount of life that the enemy had. Something quite common in RPGs´ combat.

If the boss has life to spare, he can attack or dare to boost his stats. Instead, if his life is low, he will only think about healing or attacking.

In the following tests, some minor errors were corrected and the gamefeel improved, in order to leave the project finished. Like the app icon change, that caused the save game to have to be restarted simply by renaming the project.

4.2 Results

Beginning with the subjective sections of the memory, based on the objectives cited in point 1.2 I can ensure that all of them have been completed. To dot this, I am going to put them again below with a comparison between what I would like to have achieved and what I actually achieved.

- **Multiplayer room system:** I think the room system is perfectly built as I imagined. Players can enter and play and when they decide to close the game, the room slot is open again for another player to enter.
- **A correct multiplayer game:** The main thing was to make the game shared between the two players. This was achieved by making the actions, in this case the attacks, visible even if our character is not the one doing it.
- **Art:** I sincerely believe that the art is one of the best things in the game, something essential in turn-based games. The character design seems perfect to me and all the elements that can be presented to the player are very nice.
- **Consistent menus:** The menus were placed based on what I established at first. being the simplest and most essential in these games. What would be the play

button and the button to access the galleries, since as I have commented, art is vital.

- **Gamefeel:** Firstly, the save game functionality is a goal accomplished, and I would also confirm the "fun" section. The game has the basic functionalities of the turn-based games, and has a factor that favors replayability and collecting in the players, which is to see the enemies as trophies to get, something that makes the user to be hooked.

The truth is that the realization of this project has been beautiful and hard in equal parts, the classic phrase of *without effort there is no reward* fits perfect.

When choosing this project so high in the list of preferences it is possible that it was carried away by curiosity and ended up in a rather complicated project for the simple fact of being multiplayer. But I am very happy that thanks to this work I have learned something that many developers fear. And I could almost certainly say that I could face another project with online functionalities with less respect than when I began this.

I also liked working on a game with features from one of my favorite video games genres, which has made the motivation I felt every day when working greater.
In summary I am very happy with the result, the online system works perfectly and the gameplay is great.

Comment that the game has been slightly adapted so that the matches are faster and for the demonstrations that must be carried out. In case of releasing a definitive version for the public, I would make the boss to have more life and the damage that it does would be higher.

Before concluding with this section, I would like to add that the development of the project went more or less as I initially planned. I expected that the difficult part would be to make the multiplayer system work correctly, but not so much, since in the end I had to spend a lot more time than I though at first, having to test instructions in a trial and error plan on certain occasions.

Also comment that the project is too heavy to be uploaded without loss of information to an online repository, so the entire project will simply be shared on Google Drive. In this link the build with the executable of the game can be found too.

You can access the project and the build for Windows and Mac with this link:
https://drive.google.com/drive/folders/17oRcWdjqkKjz-Y8EnPRP-w0PiSnMrW3A?usp=sharing

Chapter 5

Conclusions and Future Work

Contents

In this chapter, the conclusions of the work, as well as its future extensions are shown.

5.1 Conclusions

During the degree I have developed some projects and video games but none comparable to this one.

First of all and the most important for being alone, something that forces you to put the maximum and more of yourself into each section of everything you have learned over the years. And another because of the need to know how to get out of really complicated circumstances, as it has been in my case, the multiplayer, something for which I was not prepared and I have ended up understanding it quite well.

In the end, I really liked this experience and it seems to me that I was right to place this option at the top of the priority list, even with the fear that I felt when investigating the world of multiplayer systems.

5.2 Future work

During the development of the project I had some good ideas that I would like to implement in the future, but due to the project deadlines and even because it might not

work well in demos I did not add them.

Being a game dedicated to public spaces, it seemed a good idea to me that the appearance of a specific boss was related to the location, using the GPS position of the players, instead of being random as it is now. This will force players to move to other locations if they want to complete the trophy gallery and will improve the gameplay. This is a very interesting topic for which I would like to dedicate a section to it below.

And finally, the extension that we would all do in our video games, expand the content, in my case, adding new playable characters, more bosses and making them be controlled when they were defeated.

5.3 Platforms and Market analysis

Currently the game is focused on PC, again, because it is a project that has had to be developed under various parameters such as deadlines and presentations.

The game as it is at this moment is perfect to be placed as entertainment on public transport, such as planes or trains. Because it is not very practical to take a laptop to a park to play, for example.

So we could talk about its future becoming a mobile game, where we can transport our small collection of pocket monsters wherever we go and make use of them in raid battles in different real-world places.

This change takes time, but it would not be hard because the game is based on clicking, which would be transformed into touch screen taps.

Once on mobile devices, what would it face? Well, it is evident that its main competitor would be Pokemon Go [11], which has similar mechanics and charismatic characters of a lifetime. But this would not be a concern since there are already similar games on the market that have made their way and are profitable, such as Nexomon. [21]

Bibliography

[1] Ansimuz. Warped shooting fx. https://assetstore.unity.com/packages/2d/textures-materials/abstract/warped-shooting-fx-195246. Accessed: 2023-06-24.

[2] Various Artists. Songs playlist. https://www.youtube.com/watch?v=hcWJ$_9$453$z4list = PLRkD5IVTTWuN - FMRBgmlql5LOyEe90 - xIindex = 5. Accessed : 2023 - 06 - 24.$

[3] AurynSky. Simple gems pack. https://assetstore.unity.com/packages/3d/props/simple-gems-ultimate-animated-customizable-pack-73764. Accessed: 2023-06-24.

[4] Brackeys. Turn-based game. https://www.youtube.com/watch?v=$_1pz_ohupPs.Accessed$: $2023 - 06 - 24.$

[5] Wallpaper Cave. Wallpaper cave. https://wallpapercave.com/. Accessed: 2023-06-24.

[6] The Tetris Company. Tetris. https://tetris.com/play-tetris. Accessed: 2023-06-24.

[7] Photon Engine. Photon. https://www.photonengine.com/. Accessed: 2023-06-24.

[8] Photon Engine. Photon manual. https://doc-api.photonengine.com/en/pun/v2/class$_photon_{11p}un_{11p}photon_network.html. Accessed$: $2023 - 06 - 24.$

[9] Germán Fabregat Llueca. Guía para la redacción de las memorias. http://mermaja.act.uji.es/itis/IS31/guia_memoria.pdf. Accessed: 2019-02-28.

[10] Nintendo Game Freak. Pokémon. https://www.pokemon.com/es. Accessed: 2023-06-24.

[11] Nintendo Game Freak. Pokémon go. https://pokemongolive.com/. Accessed: 2023-06-24.

[12] Aangmar Games. Room system for unity. https://www.youtube.com/watch?v=pQxscGB7S4c. Accessed: 2023-06-24.

[13] Google. Google tools. https://www.google.es/. Accessed: 2023-06-24.

[14] Google. Youtube. https://www.youtube.com/. Accessed: 2023-06-24.

[15] Music Technology Group. Freesound. https://freesound.org/. Accessed: 2023-06-24.

[16] John Hammersley and John Lees-Miller. Overleaf. https://www.overleaf.com/. Accessed: 2023-06-24.

[17] idealista. Studio apartment price. https://www.idealista.com/alquiler-oficinas/castellon-de-la-plana-castello-de-la-plana/centro/zona-hospital-plaza-del-real/. Accessed: 2023-06-24.

[18] Jaakko Iisalo. Angry birds. https://www.angrybirds.com/play/. Accessed: 2023-06-24.

[19] Indeed. 2d designer salary. https://es.indeed.com/career/dise%C3%B1ador-gr%C3%A1fico/salaries. Accessed: 2023-06-24.

[20] Indeed. Junior programmer salary. https://es.indeed.com/career/salaries/programador%20junior%20Salaries%7D%7D. Accessed: 2023-06-24.

[21] Vewo Interactive. Nexomon. http://www.vewointeractive.com/. Accessed: 2023-06-24.

[22] Microsoft. Visual studio. https://visualstudio.microsoft.com/es/. Accessed: 2023-06-24.

[23] Rastreator. Fiber type internet price. https://www.rastreator.com/telefonia/tarifas/tarifas-de-fibra-mas-baratas.aspx. Accessed: 2023-06-24.

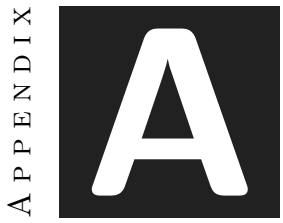
[24] Storyteller. Fake you. https://fakeyou.com/. Accessed: 2023-06-24.

[25] tarifaluzahora. Electricity consumption price. https://tarifaluzhora.es/info/consumo-mensual-luz. Accessed: 2023-06-24.

[26] Krita Team. Krita. https://krita.org/. Accessed: 2023-06-24.

[27] Unity Technologies. Unity. https://unity.com/es. Accessed: 2023-06-24.

[28] Unity Technologies. Unity manual. https://docs.unity3d.com/Manual/UnityManual.html. Accessed: 2023-06-24.



Appendix

List of Figures

List of Tables

Appendix B

Source code

This appendix compiles the most relevant and the commented scripts throughout the entire document, more specifically in the development section. The project has about 17 scripts, so it is understandable that not all of them are included here. If you are interested in seeing all the code of the project, you can find the link to it in section 4.2 .

Create Room Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
using Photon.Realtime;

public class Crear_Room : MonoBehaviourPunCallbacks
{
    public int numero;


    // Start is called before the first frame update
    void Start()
    {

    }

    public void CrearRoom()
    {
        numero = Random.Range(1, 100);

        Debug.Log("Se va a crear una nueva sala");

        PhotonNetwork.JoinOrCreateRoom("Sala no." + numero, new RoomOptions() { MaxPlayers = 2 },
        TypedLobby.Default);

        Debug.Log("Se creo una nueva room" + numero);

        PhotonNetwork.LoadLevel("Combate"); //

        //PhotonNetwork.LoadLevel("Multijugador");
    }

    public override void OnCreateRoomFailed(short returnCode, string message)
    {
        Debug.Log("No se pudo crear la sala, se vuelve a intentar");

        CrearRoom();
    }
}
```

Join Room Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
using Photon.Realtime;

public class Unirse_Room : MonoBehaviourPunCallbacks
{
    public int numero;


    // Start is called before the first frame update
    void Start()
    {

    }

    public void UnirseRoom()
    {
        PhotonNetwork.JoinRandomRoom();
        Debug.Log("Se unio a una room aleatoria");
    }

    public override void OnJoinRandomFailed(short returnCode, string message)
    {
        Debug.Log("No se pudo unir a ninguna sala, se creara una nueva");
        CrearRoom2();
    }

    public void CrearRoom2()
    {
        numero = Random.Range(1, 100);

        Debug.Log("Se va a crear una nueva sala");

        PhotonNetwork.JoinOrCreateRoom("Sala no." + numero, new RoomOptions() { MaxPlayers = 2 },
        TypedLobby.Default);

        Debug.Log("Se creo una nueva room" + numero);

        PhotonNetwork.LoadLevel("Combate"); //

        //PhotonNetwork.LoadLevel("Multijugador");
    }

    public override void OnCreateRoomFailed(short returnCode, string message)
    {
        Debug.Log("No se pudo crear la sala, se vuelve a intentar");

        CrearRoom2();
    }
}
```

Characters Unit Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Unit : MonoBehaviour
{
    public string unitName;

    public int damage;

    public int maxHP;
    public int currentHP;

    public int healing;

    public bool TakeDamage(int dmg)
    {
        currentHP -= dmg;

        if(currentHP <= 0)
        {
            return true;
        }

        else
        {
            return false;
        }
    }

    public void Heal(int amount)
    {
        currentHP += amount;
        if(currentHP > maxHP) currentHP = maxHP;
    }
}
```

On joined room Function

```
public override void OnJoinedRoom()
    {
        //if(battleCreated) return;

        jugadores = PhotonNetwork.PlayerList;

        jugador = jugadores.Length;

        Debug.Log("Se unio el jugador no." + jugador + "a la sala.");

        PhotonNetwork.NickName = jugador.ToString();

        state = BattleState.START;
        //string playerName = (jugadores.Length == 1 ? "Player" : "Player2");
        string playerName = "Player" + (PlayerPrefs.GetInt("monster") + 1).ToString();
        if (playerName == "Player1") playerName = "Player";

        string monsterName;

        if(jugador == 1)
        {
            Hashtable hash = new Hashtable();
            int random = Random.Range(0,2);
            hash.Add("monsterType", random);
            PhotonNetwork.LocalPlayer.SetCustomProperties(hash);
            PlayerPrefs.SetInt("monsterType", random);
            monsterName = "Enemy" + (random + 1).ToString();
            if (monsterName == "Enemy1") monsterName = "Enemy";
        }

        else
        {
            monsterName = "Enemy" + ((int)PhotonNetwork.PlayerListOthers[0]
            .CustomProperties["monsterType"] + 1).ToString();
            if (monsterName == "Enemy1") monsterName = "Enemy";
        }

        if(!battleCreated) StartCoroutine(SetupBattle(playerName, monsterName)); //

        this.photonView.RPC("UpdateBossHPFromMaster", RpcTarget.MasterClient);
        Debug.Log("Actualizando vida");

        /*
        if(battleCreated)
        {
            this.photonView.RPC("SetBossHPAll", RpcTarget.All, enemyUnit.currentHP);
            Debug.Log("Actualizando vida");
        }
        */

        battleCreated = true; //
    }
```

Setup the battle Function

```
IEnumerator SetupBattle(string Player, string Enemy)
    {
        GameObject playerGaOb = PhotonNetwork.Instantiate(Player, transform.position,
        Quaternion.identity, 0); //
        playerUnit = playerGaOb.GetComponent<Unit>();
        playerGaOb.transform.parent = playerBattleStation; //
        playerGaOb.transform.localPosition = Vector3.zero; //
        playerGaOb.transform.localScale = new Vector3(1.75f, 1.75f, 1.75f); //

        GameObject Gem = Instantiate(SpawnAnimation, playerGaOb.transform);

        Gem.transform.localPosition = new Vector3(1f, 2.5f, 0f);
        Gem.transform.localScale = new Vector3(0.75f, 0.75f, 0.75f);

        //if(!playerGaOb.GetComponent<PhotonView>().IsMine) playerGaOb.SetActive(false);

        GameObject enemyGaOb = PhotonNetwork.Instantiate(Enemy, transform.position,
        Quaternion.identity, 0); //
        enemyUnit = enemyGaOb.GetComponent<Unit>();
        enemyGaOb.transform.parent = enemyBattleStation; //
        enemyGaOb.transform.localPosition = Vector3.zero; //
        enemyGaOb.transform.localScale = new Vector3(1.75f, 1.75f, 1.75f); //

        //if(!enemyGaOb.GetComponent<PhotonView>().IsMine) enemyGaOb.SetActive(false);

        dialogueText.text = "A␣wild␣" + enemyUnit.unitName + "␣approaches...";

        playerHUD.SetHUD(playerUnit);
        enemyHUD.SetHUD(enemyUnit);

        yield return new WaitForSeconds(2f);

        Destroy(Gem);

        SpawnAlly();

        this.photonView.RPC("SpawnAlly", RpcTarget.MasterClient);

        //this.photonView.RPC("CleanupScene", RpcTarget.All);

        state = BattleState.PLAYERTURN;
        PlayerTurn();
    }
```

Player attack Function

```
IEnumerator PlayerAttack()
    {
        //Damage the enemy

        bool isDead = enemyUnit.TakeDamage(playerUnit.damage);

        //enemyHUD.SetHP(enemyUnit.currentHP);
        this.photonView.RPC("SetBossHPAll", RpcTarget.All, enemyUnit.currentHP);
        dialogueText.text = "The attack is successful!";

        PhotonNetwork.Instantiate(AttackEffects[PlayerPrefs.GetInt("monster")].name,
        AttackEffects[PlayerPrefs.GetInt("monster")].transform.position

        , AttackEffects[PlayerPrefs.GetInt("monster")].transform.rotation, 0);

        yield return new WaitForSeconds(2f);

        //Check if the enemy is dead

        if(isDead)
        {
            //End the battle
            state = BattleState.WON;
            EndBattle();
        }

        else
        {
            //Enemy turn
            state = BattleState.ENEMYTURN;
            StartCoroutine(EnemyTurn());
        }

        //Change state based on what happened
    }
```

Player heal Function

```
IEnumerator PlayerHeal()
    {
        playerUnit.Heal(playerUnit.healing);

        playerHUD.SetHP(playerUnit.currentHP);
        dialogueText.text = "You␣feel␣renewed␣strenght!";

        SoundManager.instance.HealSound.Play();

        yield return new WaitForSeconds(2f);

        state = BattleState.ENEMYTURN;
        StartCoroutine(EnemyTurn());
    }
```

Player boost Function

```
IEnumerator PlayerBoost()
    {

        int random = Random.Range(0,3);

        if(random == 0) //Aumente curacion
        {
            playerUnit.healing += 3;
            dialogueText.text = playerUnit.unitName + "'s␣healing␣raised!";
        }

        if(random == 1) //Subir Ataque
        {
            playerUnit.damage += 5;
            dialogueText.text = playerUnit.unitName + "'s␣attack␣raised!";
        }

        if(random == 2) //Bajar Ataque
        {
            if((enemyUnit.damage -3) <=5) //
            {
                dialogueText.text = enemyUnit.unitName + "'s␣attack␣is␣so␣low!";
            }
            else
            {
                this.photonView.RPC("LowerBossAttack", RpcTarget.All);
                dialogueText.text = enemyUnit.unitName + "'s␣attack␣decreased!";
            }
        }

        SoundManager.instance.BoostSound.Play();

        yield return new WaitForSeconds(2f);
```

```
34
35          state = BattleState.ENEMYTURN;
36          StartCoroutine(EnemyTurn());
37      }
```

Photon room extensions Function

```
1  public static class PhotonRoomExtensions
2  {
3      public static void SetCustomProperty(this Room room, string propName,
4      object value)
5      {
6          ExitGames.Client.Photon.Hashtable prop = new ExitGames.Client.Photon.Hashtable();
7          prop.Add(propName, value);
8          room.SetCustomProperties(prop);
9      }
10
11     public static void SetCustomProperty(this Room room, string propName, object value, object oldValue)
12     {
13         ExitGames.Client.Photon.Hashtable prop = new ExitGames.Client.Photon.Hashtable();
14         prop.Add(propName, value);
15         ExitGames.Client.Photon.Hashtable oldvalueProp = new ExitGames.Client.Photon.Hashtable();
16         oldvalueProp.Add(propName, oldValue);
17         room.SetCustomProperties(prop, oldvalueProp);
18     }
19 }
```

Gallery Manager Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GalleryManager : MonoBehaviour
{
    public GameObject[] Monsters;
    public GameObject[] Texts;

    public GameObject MonsterTab;


    public void CloseMonsterTab()
    {
        foreach(GameObject item in Monsters)
        {
            item.SetActive(false);
        }

        foreach(GameObject text in Texts)
        {
            text.SetActive(false);
        }

        MonsterTab.SetActive(false);
    }

    public void OpenMonsterTab(int index)
    {
        MonsterTab.SetActive(true);
        Monsters[index].SetActive(true);
        Texts[index].SetActive(true);
    }
}
```

On join animation Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
using Photon.Realtime;

public class Gem : MonoBehaviour
{
    Player[] jugadores;
    public int jugador;

    // Start is called before the first frame update
    void Start()
    {
        jugadores = PhotonNetwork.PlayerList;

        jugador = jugadores.Length;

        if(jugador != 1 && gameObject.name.Contains("Enemy"))
        {
            if(GetComponent<PhotonView>().IsMine)
            {
                transform.position = new Vector3(-1f, transform.position.y, transform.position.z);
            }

            else if(!GetComponent<PhotonView>().IsMine)
            {
                transform.position = new Vector3(-5f, transform.position.y, transform.position.z);
            }
        }

        StartCoroutine(DeleteOn2());
    }

    // Update is called once per frame
    void Update()
    {

    }

    public IEnumerator DeleteOn2()
    {
        yield return new WaitForSeconds(2f);

        if(TryGetComponent<PhotonView>(out PhotonView tempPV))
        {
            PhotonNetwork.Destroy(this.gameObject);
        }

        else
        {
            Destroy(this.gameObject);
```

```
        }
    }
}
```

Connect to Master Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
using Photon.Realtime;

public class ConectMaster : MonoBehaviourPunCallbacks
{
    // Start is called before the first frame update
    void Start()
    {
        PhotonNetwork.GameVersion = "0.1";

        PhotonNetwork.ConnectUsingSettings();
        Debug.Log("Se va a conectar al servidor Maestro");
    }

    public override void OnConnectedToMaster()
    {
        PhotonNetwork.AutomaticallySyncScene = true;

        Debug.Log("Se ha conectado al servidor Maestro");
    }

    public override void OnDisconnected(DisconnectCause cause)
    {
        Debug.Log("Problema al conectar al servidor");
    }
}
```