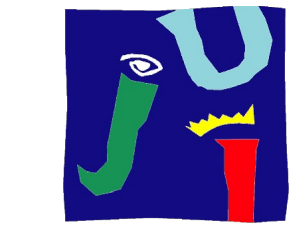

Creating a physical activity motivator as a video game main mechanic

Sergio Badia Ganau

Final Degree Project

Video Game Design and Development

Universitat Jaume I

01.07.2021

ACKNOWLEDGEMENTS

I would like to thank my project supervisor, Joaquín Huerta Guijarro, for his indications that helped me carrying forward some of the most difficult parts for me in this project, in addition to accepting my idea as a project proposal.

I want to thank my family for giving me support and always encouraging me with this project and the degree generally speaking.

ABSTRACT

My Final Degree Project consists in creating an original game in which the physical activity is an important mechanic, so it motivates the player to do physical exercise in order to fully enjoy the game.

The project is composed by two parts then: one of them is the game itself, which can be played on the computer; the other one is a mobile application responsible for counting the user steps in real life. The main idea is to use those counted steps as points in the game to exchange them for different upgrades for the character and additional help for the player.

This document details everything related to this work: game assets and design, game programming, pedometer functionality for the app, as well as updating data from a mobile app to a computer game.

CONTENTS

CHAPTER 1: INTRODUCTION

1.1 – PROPOSAL AND WORK MOTIVATION

The general idea was to develop a short game where I could apply all the knowledge I obtained in the degree in a practical way. To do this, a relatively flexible proposal was needed, one focused on the development of a video game. Having to develop a physical activity motivator mobile app was what outlined the idea of this project: a game that requires doing some physical activity before playing it and gives access to more content the more exercise the user does. All of this required a great variety of work, from designing a computer game to develop a mobile app that should sync with the game somehow. However, it was a good way to put into practice some different subjects of this degree while I would do what I liked.

In order to implement a physical activity motivator mechanic in a natural way (without forcing it in a game in which does not fit), the main character was given a job that actually requires some training to keep up with. In order to help them with the training, this character has a smart device or AI that controls their training, and this training is actually the physical activity the user has to do in real life in order to unlock more in-game content. That is how the steps the user takes in real life are converted to in-game energy points of the protagonist. All this work will be better detailed in later chapters of the document.

As for the game idea, it is a 2D game, divided by levels. Each level has two parts: the first one is about “beat’em up” and platforming mechanics gameplay-wise, a stage full of enemies in which the player will have to move forward between platforms, beating enemies trying to stop the main character. The second part of the level is a little more paused, in which the player investigates the environment (“point and click” mechanics) solving some puzzles occasionally, and tries to find out a criminal leader whereabouts.

As a curiosity, for the game title I chose “Right Path” since it is not a too long title and especially because it is an expression that can have double meaning in my case, referring to “the correct way” or the “direct way” since the protagonist ends up accepting jobs about catching malefactors and delinquents; as well as “path” related to walking, since it is something the player must do in real life to keep playing.

For the game design, the project idea comes from a variety of games. This is not a game highly inspired by other specific game, but one with some different combined mechanics that fit in my project instead, as long as it does it naturally with a physical activity motivator. Also, it had to be something that would give me enough hours of work.

Logically, the game mechanics are not completely original, and there were some inspirations that were helpful for visualizing the game concept.

The “beat'em up” gameplay has been influenced by games like The Friends of Ringo Ishikawa, Senran Kagura: Burst or Katana ZERO. They are all games with “beat'em up” mechanics, with mostly horizontal stages and side-scrolling view, also with some platforming elements (without this being the main mechanic).

Figure 1: Senran Kagura: Burst ©Marvelous

Figure 2: The Friends of Ringo Ishikawa ©Circle Entertainment

As for the investigation gameplay, it is basically a “point and click” experience, inspired by titles like Ace Attorney Investigations: Miles Edgeworth, Root Letter or Professor Layton series. They include scenario investigations, and the players must click on objects or other elements that catch their attention in order to solve a mystery. However, in Right Path, the objective is to catch someone, so the players do not have all the time they want. They need to search for clues quickly while a timer is running down unlike the games I mentioned as examples.

Figure 3: Ace Attorney Investigations: Miles Edgeworth ©Capcom

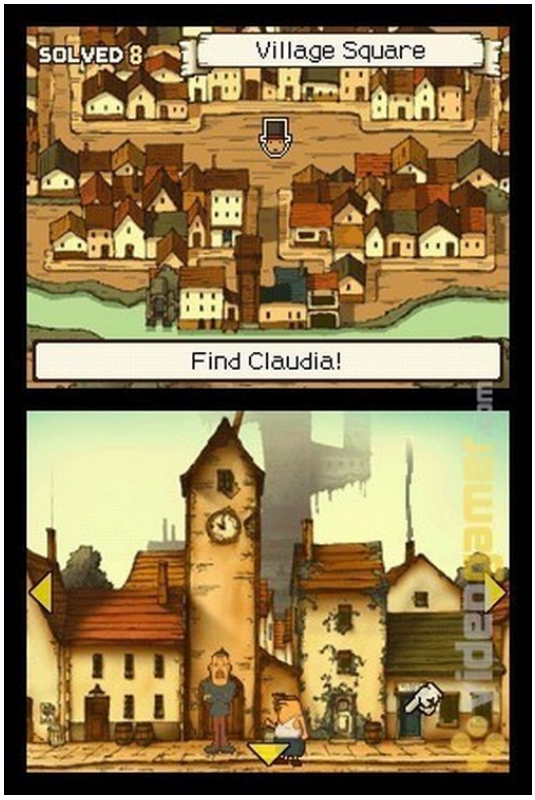

Figure 4: Professor Layton and the Curious Village ©Level-5

1.2 – OBJECTIVES

The objective of this project is to create a game that also motivates the player to do exercise. A game that can be played as a normal game, with its main difference being the actual player having to do physical activity to unlock content. It has been designed in a way that it does not force the player to do lots of exercise if they want to continue playing, but to motivate them to do exercise, with a relatively attainable minimum goal before each level, expanding the game experience in different ways and offering the player optional content as a reward for going beyond the minimum limit each time. The reason for this design is to keep the players interested even if they did not walk regularly at all until then, so they will not feel overwhelmed and quit the game because of that.

This game could be expandable with more levels to extend its duration and the game experience in general. However, this project has a limited number of available hours and a strict deadline, thus a whole or full game can not be done in the available period of time. Take into account that this is a short game that presents a concept and includes a few levels to see how it works, in addition to showing the progression between these levels.

1.3 – ENVIRONMENT AND INITIAL STATE

The idea of the work was to add physical exercise as a game mechanic, but I was not looking to create a game whose purpose is specifically to physically exercise the player. Instead, the idea was to create a game, a normal game, where I could fit the physical activity as a main mechanic. In other words, not doing physical activity while playing, but doing it and playing, as two complementary activities. I informed my supervisor, Joaquín Huerta, about this idea before starting to work on the project.

I knew it was possible to communicate data between a mobile app and a computer program, but I did not know how to do it. Searching the internet, I came up with the idea of uploading data (for example, a text file) to Google Drive from a mobile device, and then downloading this file from Unity and reading it to obtain this data. That way, I could keep updated the game with the steps data. Although this implementation would work and it does not seem too complicated, it's not the most practical way for the users, who would have to create a text file every time they need it and keep this file in their Google Drive accounts. I wanted to skip these two parts, so the user would only upload and download the steps data simply with a button, all other processes would be automatically handled by the mobile app or the computer game themselves.

Finally, I found the way to do this with a database, specifically a remote one, which would be accessed by the mobile app and the computer game to manage the steps data. The users would only have to worry about typing their names to upload or download this data.

The game was created with Unity 2D and the mobile app was developed with Android Studio, two programs we have already worked with throughout the degree.

CHAPTER 2: PLANNING, WORK DONE AND RESOURCES EVALUATION

2.1 – PLANNING

It is important to define a clear planning before starting the project in order to know what to do in each moment, identifying the important and more difficult tasks. Initially, the estimated tasks division was the following:

Art design (40h): This included the main character, enemies and other characters design and animations.
Game programming (180h): Everything related to developing the code of the game.
App programming (30h): Everything related to designing and developing the mobile app.
Music and sound design (20h): Including every sound aspect of the game, from music to the shortest sound effect.
Project memory (20h): Writing this memory.
Project presentation (10h): Basically, preparing myself to the project presentation, as well as preparing the project for it to show everything needed in an efficient way.

In the next section of this chapter there is a more detailed tasks division, with the actual tasks done in order. You can see that there are some significant differences between the estimated tasks and the actual tasks:

- Those related to the art design, project presentation preparation and game development are what I was comparatively expecting before starting the project, and took me a similar amount of time. The game developing took a bit more than I was expecting, due to some tasks such as thinking about the implementation and testing, which were not taken into account in that moment. Nonetheless, it is still an understandable margin given the large amount of hours required for this part of the project.
- The project memory has been modified several times, adding new sections and modifying and improving the existing ones, so that is the main reason why this task has taken some more hours than planned.
- However, the most noticeable difference is the series of tasks related to the mobile app, its design and its development. 30 hours was definitely a wrongly estimated time, an insufficient amount of time to get everything related to the app working as expected. In general, this underestimation was the result of ignoring important tasks due to never having worked on these kind of programs, neither on a pedometer implementation (sensors, background operations, etc.) nor on a remote database. Besides, the first implementation of the pedometer did not work in the end, so a second, new implementation was needed, which required almost twice as long. Creating a database

from zero was not something I had done before, although having operated with existing local databases helped speeding up the process.

- Due to the required hours being greater than planned, the tasks related to music and sound design were finally removed, so the game has no sound. In addition, I have never worked with software like what I planned to use, so the amount of time these tasks would have taken was dedicated to other, more important ones. Despite this, I already expected that this could happen and that is why I saved this part for the end, since it was the least important one of them all.

In the end, despite the difficulties, thanks to the knowledge acquired during each course of the degree, it was possible to come up with solutions for the development problems.

Figure 5: Gantt (estimated work)

2.2 – WORK DONE

Once a clear general idea was established for the project, which can be conveniently divided in the game and the app, I decided to start with the game itself first.

The initial step was to think about every aspect of the game individually, focusing on the main character first, specifically in its design. Since it was going to be the protagonist of the game and therefore, the only character the player was going to control during the whole game, it was necessary a good amount of time to make sure it looked as planned, both in terms of design and animations. After finishing the animations related with the character movement, they were implemented in the game, and everything necessary was programmed to check all the work done so far, with a really simple stage and the camera following the player. The previous procedure was the same for the combat animations and controls. Although checking the enemy response to the player combat was not possible without first implementing some kind of enemy with its appropriate AI.

But before taking care of the enemies, and on the other hand, a dialogue system was created and prepared to be used in certain scenes of the game, as well as the second part of each level.

Back to testing the combat system, I began programming the enemies basic AI. However, since the enemies design was not done yet, a quick dummy design was used provisionally, and so I programmed the enemy movement and attacking. The last thing of this part was showing the player some combat feedback, so I programmed the enemy health and a health bar for the main character. The basic combat system was now finished, though some adjustments were needed and I spent some time doing that everyday until the expected results were finally achieved.

Now that the first part of the gameplay was in a more advanced state and with its main mechanics defined, I started with the second part of the gameplay: the investigation part.

I already had the dialogue system created at that time (I did it while solving some combat bugs), but its implementation was actually not too practical or simple if I wanted to create or expand dialogues in any point of the game. That is why I found important to modify it before taking care of anything else, since not doing it would needlessly waste some of the available time. Once that was done, I designed a few assets in order to place them around the scene and developed the item interaction system. For this type of gameplay, mouse controls are much more intuitive than using the keyboard. Also, the combat controls were not needed for this part of the levels, so I adjusted the camera, besides anything necessary, to create this new kind of control, this time with the computer mouse, including mouse movement around the scene and “point and click” interaction.

Getting back to the “beat'em up” gameplay, I developed some advanced movements for the main

character (such as dashing, defending, strong attacks, etc.). These ones were planned to be available only if the user can buy the combat upgrade with energy points (that is, steps taken in real life converted to in-game points) at the beginning of each level. I also designed, implemented and adjusted new types of enemy, each one of them having some unique features which offer more variety of battling situations for the player. The idea is to abet the players to use all the movements they have to be able to adapt better to each combat situation.

On the other hand, though I still did not have all the ideas for the puzzles in the investigation parts, it was important to know how to take care of the puzzle UI. Thus, different options were tested and one of them was kept.

With that, almost everything related to programming the game was taken care of. What was left to do now was putting everything in its place, design new assets and applying the appropriate scripts to them when they were integrated into the game. However, the mobile application was not even started by that moment. Until then, all I did about that aspect was searching for ways to do what I wanted, but I did not actually began implementing it. Therefore, that was the next step in the project development.

The mobile app can be divided in two parts according to the functions it must fulfill. The first one is counting the steps the user takes and the second one is to update these steps to a remote server. This data will be downloaded by the computer game and will convert the steps to in-game points (energy points for the main character).

Regarding the pedometer function, I had some difficulties developing it, and because of that, the development was restarted since the first implementation ended up not giving the expected results. Then, when the second implementation was done, now working correctly, I added some options to improve the user experience (different sensitivity levels, showing the current goal, making a cleaner interface, etc.).

As for the second function, about uploading and retrieving data from a remote database, it took me a pretty long time to finish it, taking into account I had to create a full remote database to make the user store the data from the app UI, update it and retrieve it from the computer game.

2.2.1 – TASKS AND DEDICATED HOURS FOR THE GAME

- **Task 1 (5h):** Thinking the game and the app concepts. Elaborate the idea.
- **Task 2 (25h):** Main character design and animations.
- **Task 3 (15h):** Main character movement and combat. Camera.
- **Task 4 (15h):** Enemies designs and animations.
- **Task 5 (20h):** Enemy AI development.

- **Task 6 (30h):** Combat mechanics development.
- **Task 7 (5h):** Dialogues system.
- **Task 8 (5h):** Investigation mechanics.
- **Task 9 (5h):** Puzzle UI implementation.
- **Task 10 (15h)**: Level design.
- **Task 11 (5h):** Retrieving steps data from the database.
- **Task 12 (5h):** Placing enemies on the stage. Enemies coordination.
- **Task 13 (20h):** Game menus and game loop.
- **Task 14 (10h):** Dialogues writing.
- **Task 15 (10h):** Intermission/extra scenes writing and development.
- **Task 16 (15h):** Testing. Bugs fixing.
  TOTAL: 205 hours.

2.2.2 – TASKS AND DEDICATED HOURS FOR THE APP

- **Task 1 (5h):** Android step counter functionality research.
- **Task 2 (10h):** Setting the UI. App design.
- **Task 3 (30h):** Pedometer functionality development.
- **Task 4 (5h)**: Remote databases research.
- **Task 5 (15h):** Setting the local database and working with it.
- **Task 6 (25h)**: Setting the remote database and working with it.
- **Task 7 (5h):** Testing. Bugs fixing.
  TOTAL: 95 hours.

Additional Tasks (50h): Documents writing and project reports.

2.2.3 – GANTT DIAGRAM

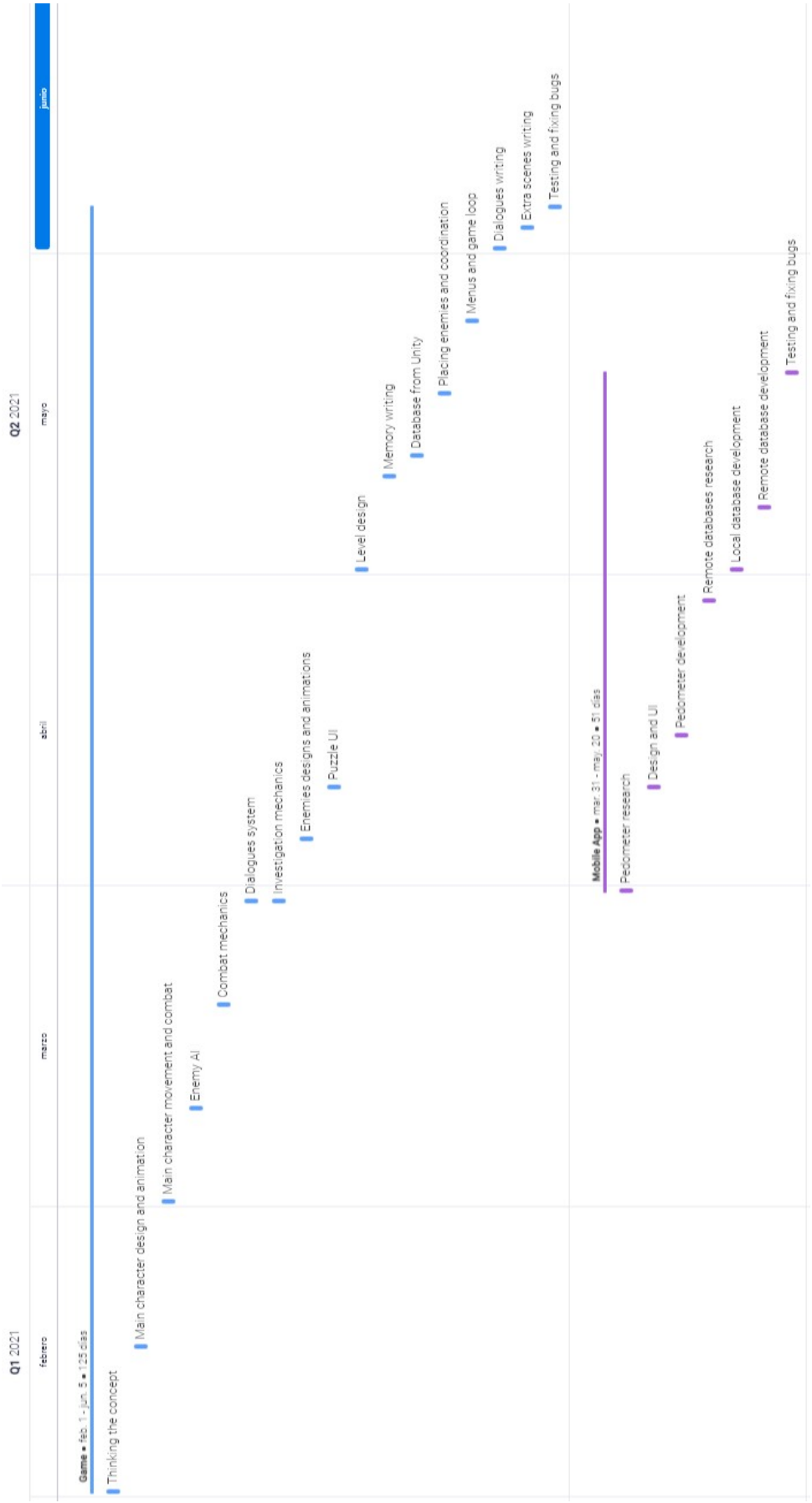

Figure 6: Gantt diagram (dedicated time)

2.3 – RESOURCE EVALUATION

The two physical resources for this project were a computer (PC) and a mobile phone (Android smartphone device). Also, I used a laptop to work outside home and some more smartphones to test the app in different devices.

Regarding the computer programs, I used quite a variety of them, since this project involves both game design and development.

For the visual aspect of the game, I used Aseprite, a program focused on pixel-art graphic style, which offers the user different options to create pixel style assets (design and animations). The concrete version was the v1.2.25. Sometimes, I also have used Adobe Photoshop CC 2017 to make a few sketches or concept idea for some designs (character, stages), though not as much as Aseprite.

Regarding the development of the game, Unity was the game engine I used (Unity 2D), since we have already used it for different subjects during the degree. The concrete version was the 2020.2.3f1. The script coding was taken care of with Visual Studio 2019, and for the SQL (only used to access the database) I used Visual Studio Code and Notepad++.

The database creation was possible thanks to two programs called XAMPP Control Panel and SQLyog Community. XAMPP offers the user access to the Apache HTTP servers, MariaDB database and MySQL, all of them necessary to work with databases with PHP scripting language. SQLyog is an interface to create and manage local databases. I used it to create the database and test it. Both programs are free and open-source, with high compatibility with other programs like Android Studio and services like phpMyAdmin, which I will mention later too.

As for the online server and remote database, I planned to handle it with Amazon Web Services (AWS) first. However, due to this service being free only for the first month (750 hours), I decided to use 000WebHost, which provides a free but limited online server. I only used it to convert the local database to a remote one, which takes care of storing all the steps data for all users. I managed the database (to do some checking and testing in general) with phpMyAdmin, that provides the user an interface for managing, consulting and editing online databases.

Last, I did all the documents writing (technical proposal, GDD, this document…) with LibreOffice 4.4, compatible with Microsoft Office and other related programs.

CHAPTER 3: GAME DESIGN DOCUMENT

Though this chapter title talks about the “game” design, the following pages will cover the game and the mobile app too. It will be, of course, appropriately indicated in the subtitles of this chapter.

This section also serves the purpose of explaining the work done by parts. In the second chapter, there is information about the planning and the work done chronologically (planning section). This chapter will take to pieces all the work done and will detail each part that, all together, compose this project.

3.1 – SYNOPSIS AND CONTEXT

Kaede Hozuki is a brave private investigator from Japan (sometime in the future) who works for the police and takes care of the most difficult cases, searching for dangerous criminals around the city and helping the police arresting them. Since her work requires considerable physical effort, her boss gives her an innovative device capable of analyzing and controlling her physical and mental capacities. This device, called SPAD Model-o (Smart Personal Assistance Device, Model o since it's still a prototype) or just Zero, helps Kaede getting prepared for her work, and tells her if she is ready to get her work done according to her physical condition.

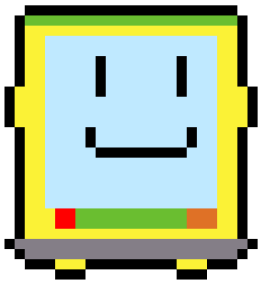

Figure 7: Spad Model-0, Zero

The game levels are basically different cases Kaede has to take care of. The procedure for her work is always the same: she gets a call from her boss telling her the details of the case and the location of the criminal the police is looking for. Then she infiltrates the dangerous neighborhoods or organizations where this wanted criminal hides to catch them, fighting delinquent gangs on her way, and once Kaede has cornered their leader, the police take action and arrests them. Her job will not be easy, and that is why she starts using this new device, which will advice her to get ready for the work and will help her avoiding mistakes she could regret.

3.2 – MOBILE APP FUNCTION IN THE GAME

The app itself is not a game, but an essential part of it. If it's turned on, it will count every step the player walks in real life, and then this data will be stored and passed to the game every time the player starts playing it. In the game, those stored steps become Kaede's physical energy. Depending on the steps the player walked, the player will (or will not) be able to start the next mission. If she is not ready, then the player can't continue playing until they walk more steps (in real life). However, that is not the only function for the energy points. The game also offers the player new movements, useful items, hints, power-ups, extra content... that can expand the game experience in different ways, if the player wants to. It's important to know that these energy points can not be obtained in any other way.

There is a certain number of steps the game requires to continue advancing, and the player must achieve this number of steps to keep playing, but the main objective of this mechanic is making the player go beyond that limit and to get lots of energy points to use them for upgrades, improvements and extra content. In short, the idea is not forcing the players to make lots of exercise (because that can lead them to just quit the game), but to encourage them to do more exercise on their own by offering them some rewards in return.

3.3 – MOBILE APP DESIGN

The idea for the mobile app was to make a simple and intuitive pedometer, and as such, it counts the steps taken by the user always the app is turned on. Apart from the number of the walked steps, the app shows a circular progress bar and the current goal in the main screen.

The user has the option to adjust the pedometer sensitivity to better adapt to their walking, in addition to change what level has to play next, so the goal of steps varies according to this.

Once the user finishes their walking exercise, they can upload the steps data just by tapping a button, and the app will take care of it automatically (updating a remote database) so the next time the user starts playing the game on their computer, the data will be updated in the game and will be automatically converted into energy points for Kaede, the main character (retrieving from the database).

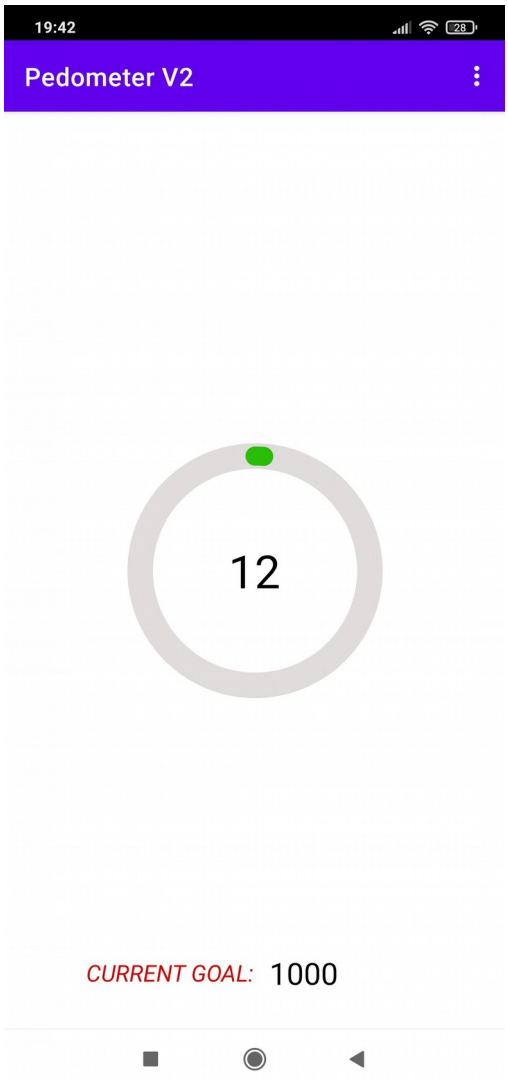

Figure 8: Main screen of the app

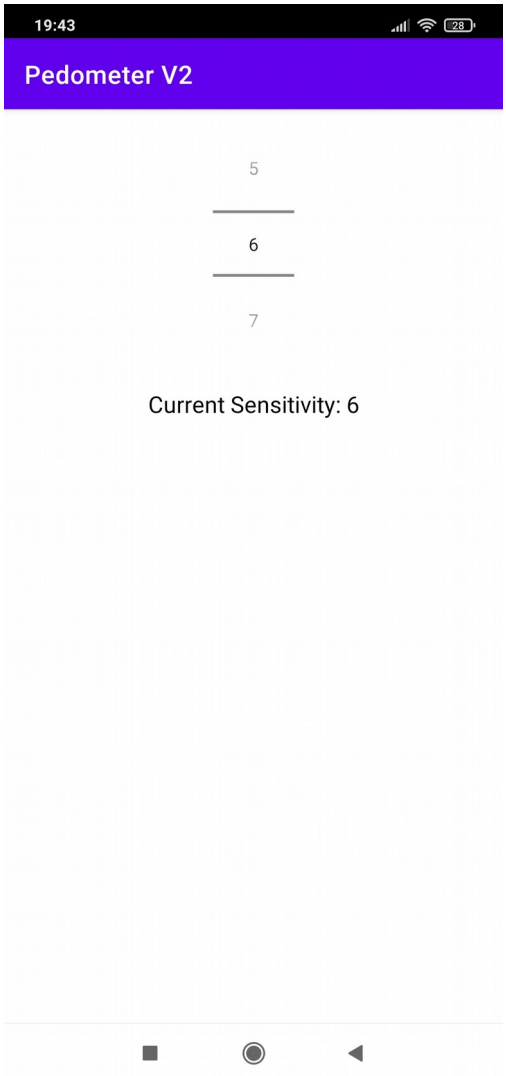

Figure 9: Settings screen of the app

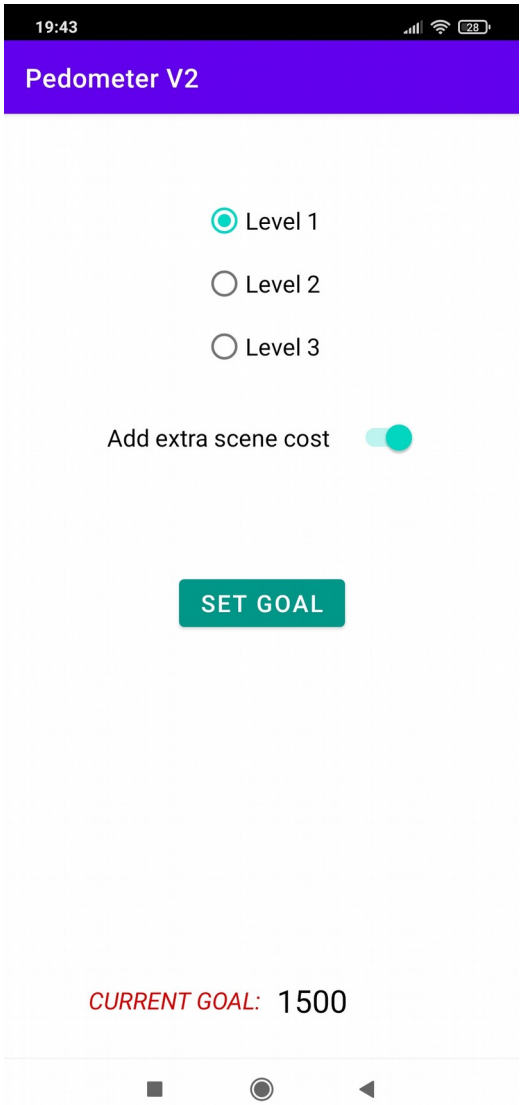

Figure 11: Define Goal screen of the app

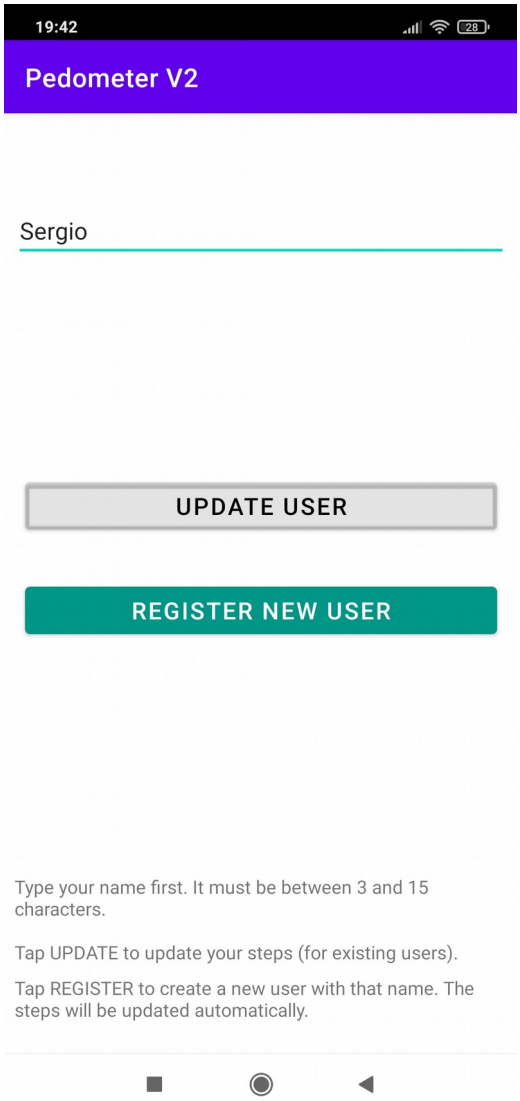

Figure 10: Updata Data screen of the app.

3.4 – GAMEPLAY

As stated before, the main objective of the game is to catch the wanted criminal of each case. The players will control Kaede to fight her way through gang members first, and search for the criminal in question second. Every case then is divided into two parts.

3.4.1 – BEAT'EM UP GAMEPLAY

The gameplay in the first part consists in hand-to-hand combat between the protagonist and a large number of opponents that appear along the way. The player will have a variety of movements which can be combined one after another to fight any kind of enemy.

Since the game is 2D and not a 2.5D one like many other beat'em up games, these sections include some platforming to give some verticality to the stage.

3.4.2 – POINT AND CLICK GAMEPLAY

After the beat’em up section, there is an investigation part in which the players control the protagonist to search for clues to find the wanted criminal. The player will need to act as a detective and search the stage, checking objects and other elements and solving a few puzzles in order to discover where the criminal is hiding or where they may be escaping, but that should be done in a time limit, because if the player doesn't discover where is the criminal before the time hits zero, the criminal will escape and Kaede will fail, leading to a failed mission (game over). Conversely, if the player gets to find the malefactor before the time hits zero, this will be arrested, successfully closing the case.

3.5 – CONTROLS AND MOVESET

In the beat'em up sections of the game, the character controls are the following. But before, just to clarify a recurring concept, “Max Mode” is a fighting mode Kaede can adopt, but only if the player has enough energy points to unlock it (besides from the minimum limit required to unlock the level). This mode adds new movements to Kaede's moveset. Note that none of them are required to beat any level, since their use is only optional. If the players have not enough energy points or they decide not to acquire the new movements, they will not be able to use these moves for that level.

- **LEFT/RIGHT ARROW:** Move the character to the left/right. The camera will follow her automatically.

Figure 12: Kaede running animation

- **SPACE:** Jumps.

Figure 13: Kaede jumping

Figure 14: Kaede falling

C: The character performs a normal attack. Pressing it three times in a row will make a combo of three attacks, each one stronger than the previous, however they have to be

executed between short periods of time, if that is not the case Kaede will attack with their first normal attack each time we end or “cut” the combo.

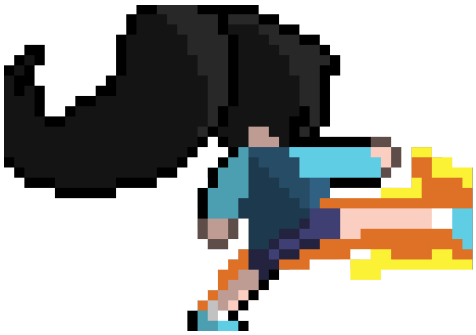

Figure 15: Kaede 3rd attack

- **SPACE + C:** While jumping, the character will perform an air kick against her rival. This is a pretty quick attack, but it is not too powerful (the caused damage is similar to one normal attack).

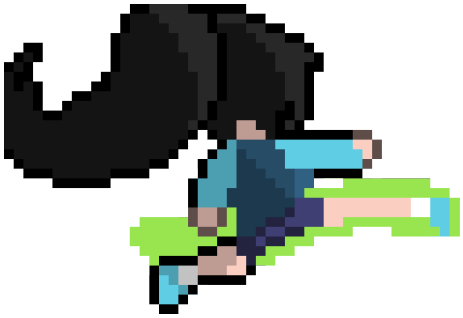

Figure 16: Kaede air attack

- **D:** Shoots the laser gun. Due to her job, the main character has a laser gun in her possession and she is allowed to use it against aggressive delinquents as a tool to neutralize them. Every time the player presses the button, a bullet is fired in the direction that Kaede is looking. The bullet bursts when it hits an enemy (dealing damage to that enemy consequently) or another bullet, or it disappears if it does not hit anyone after traveling a certain distance. This way, the main character can fight long range against enemies, though on the other hand, this bullet is not as powerful as a normal attack. The character can not shoot while jumping.

Figure 17: Kaede shooting her laser gun

- **Z:** The main character stands in a guard posture and she is then defending against enemy attacks. While defending, the enemy attacks will not have effect on the character (meaning that they will not damage Kaede). However, while standing in this position, Kaede will neither move nor attack.

Figure 18: Kaede defending herself

- **X:** Strong attack (only available in Max Mode). This attack is more powerful than the normal attack. When the character is in Max Mode, an energy aura surrounds her, and can be used to attack with the whole body and regain the combat posture again relatively quick. This attack is a bit slower than a normal attack.

- **SPACE + X:** Strong air attack (only available in Max Mode). Another strong attack, but this one is executed in the air. Like the previously explained strong attack, Kaede uses her energy aura to attack with her whole body an enemy in the air. Besides, an air kick (that is, a normal attack but in the air) can not propel the enemy, but this one can. If the

character hits an enemy in the air with this attack, the enemy will be pushed some distance in the direction of the attack.

- **S:** Teleport dash (only available in Max Mode). This move was created for situations when the character is in a compromising situation and needs to escape urgently. When the player uses this move, Kaede surrounds herself with smoke to mislead her rivals and will teleport immediately a certain distance in the direction she is facing. Keep in mind that the players can not use this all the time, so in order to avoid them exploiting this move, there is a dedicated charging bar indicating that the player will be able to use this move again once the bar is fully recharged (in a yellowish hue to make things more visual).

- **UP ARROW + C:** Up attack. This attack deals little damage, but it can throw the enemy up for a little while. When the enemy has been thrown into the air, they will start spinning, being unable to move during some seconds, and the player can take advantage of this situation attacking them with air kicks or a strong air attack (if available).

- **DOWN ARROW + X:** Area attack (only available in Max Mode). This attack is useful when the character finds herself surrounded by enemies in front and behind her, because this affects both the back and the front enemies in range.

Figure 19: Kaede area attack

For the investigation sections of the game, the player controls will be different, using the mouse to control the game.

- **Moving the mouse to the left/right borders of the screen:** Move the camera around the place.

- **Clicking on an object:** Checks that object and triggers a dialogue talking about it. This dialogue can vary depending on the situation (this will be explained in another chapter of

this document). Sometimes, clicking on an object can trigger a puzzle.

- **Interacting with puzzles:** The player also uses the mouse to do this, but sometimes the puzzle may require the player to type with the keyboard.

3.6 – GAME LEVELS AND LEVEL FEATURES

The game is composed by the tutorial and three different cases (three full levels).

The first part of these cases consists of two-dimensional scrolling stages. The camera follows the main character across the stage. During the course of the level, some enemies appear trying to stop Kaede by fighting her.

The second section of the level, the investigation part, will develop in a smaller size closed stage in all levels, since the objective of the player is to investigate the room to find hints of the whereabouts of the wanted criminal, interacting with objects and the stage in general.

Due to the nature of the game, there is a saving system, so the player can resume the game where they left it every time the game is open.

The difficulty increases in each level so the first one is easier than the third one, which is the most difficult. However, the difference of difficulty is not too high and every level supposes a challenge to the player. The gameplay and mechanics of all three levels is the same. The differences between them are present in the environment, the stage design (both for the beat'em up and the investigation parts), some enemies and puzzles in the investigation section.

If the character is knocked off, the player will restart the level. However, the player can sometimes find aid kits while moving across the stage. These kits recharge a certain amount of the main character's life bar.

The tutorial serves as a simple case for the player to dynamically try the controls and mechanics and get used to them. For that reason, it supposes no actual challenge for the player in order to practice without any kind of distraction.

3.6.1 – LEVEL 0: TUTORIAL

The tutorial, also called level 0, is located in an industrial estate. There are no human enemies here, it is just a place to practice, and here is where the player will learn the game controls.

Due to its nature as a tutorial, its access is free, and that means no steps are needed for enter this level. At the end of it, the player will be told what to do to keep playing, since the next levels require a minimum of steps.

3.6.2 – LEVEL 1

Located at the riverside, at the outskirts of a town. There are not many platforming elements here since it is the first level.

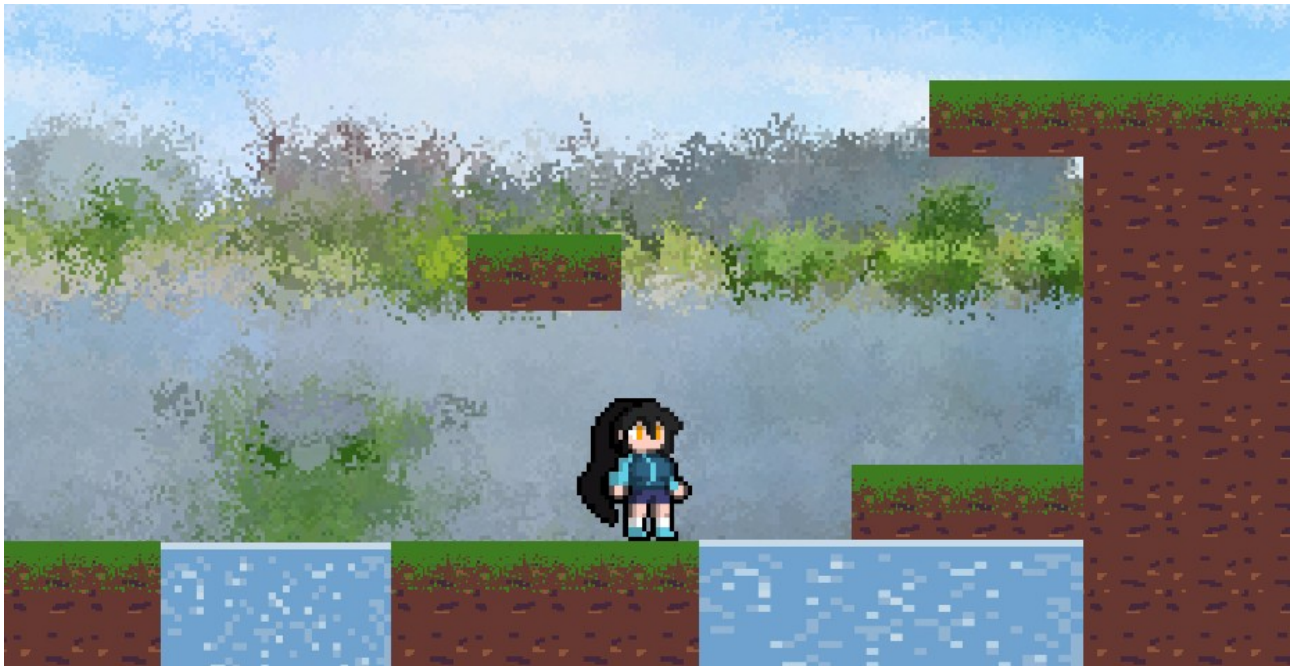

Figure 20: Level 1 stage

To enter this level, 4000 minimum steps are required.

3.6.3 – LEVEL 2

Located in an abandoned back alley. A dangerous criminal gang has established its base in this abandoned alley, so there are some abandoned houses, boxes and other things that act as platforming elements.

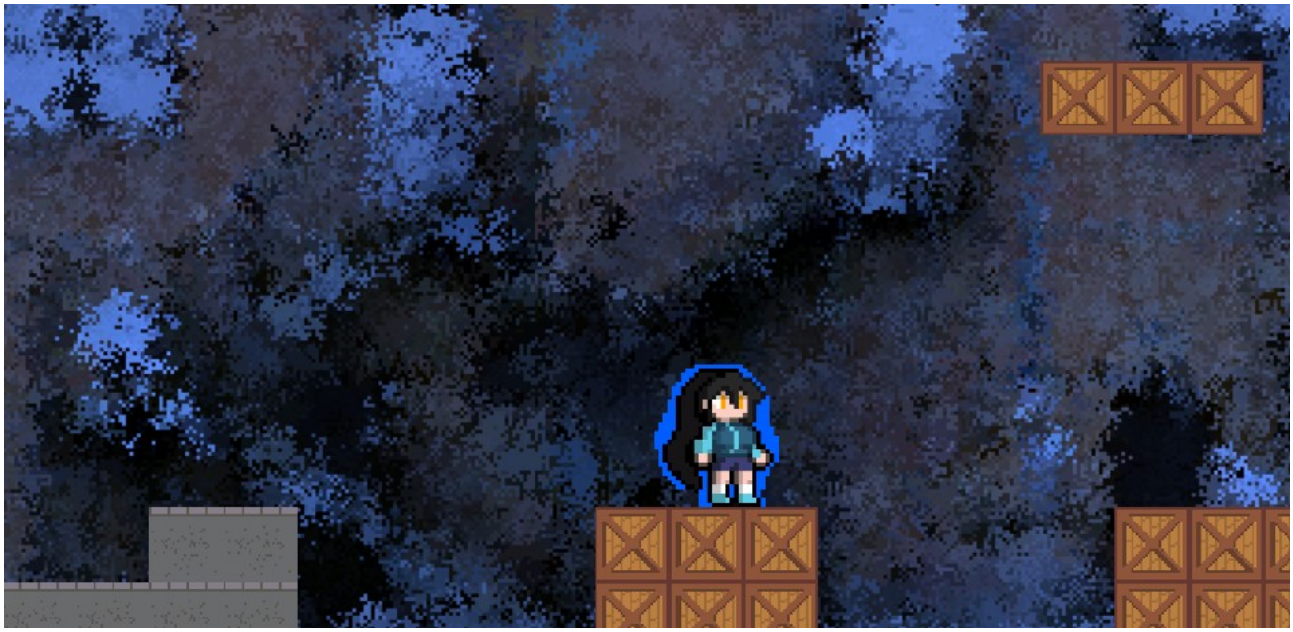

Figure 21: Level 2 stage

To enter this level, 6000 minimum steps are required.

3.6.4 – LEVEL 3

Located in an underground facility used as an actual secret base. Due to the facility being made by the leader of the criminal organization itself, there are a lot of platforming elements, including falling and moving platforms.

Figure 22: Level 3 stage

To enter this last level, 8000 minimum steps are required.

3.6.5 – INTERMISSION EXTRA SCENES

These scenes can be watched between cases. Specifically, there is one scene after every level (except for the tutorial). They are optional and cost some energy points, but in order to always give the player the choice to watch them, these points can be used whether the player has them or not. In any case, the cost will be subtracted from the amount of points, so more steps will be required to enter the next level. In the mobile app we can indicate this in the current goal screen by simply turning a switch button on. This will automatically add the extra scene cost to the minimum goal of steps for the next level.

The extra costs in steps for each intermission scene are the following:

Extra scene after level 1: 1000
Extra scene after level 2: 1500
Extra scene after level 3: 2000

The intermission scenes consist of mini stories with dialogues about Kaede when she is not at work. They contain dialogues and, sometimes, questions with different answers that the player has to choose correctly.

3.7 – ENEMIES

Every level has several enemies. Their function is to fight Kaede and to try to stop her. It has been taken into account that not every enemy attack Kaede at the same time, and instead they have some coordination (depending on the type of enemy in question) within their AI to avoid making the game excessively hard at some points.

Up to five types of enemies can be distinguished.

3.7.1 – MELEE ENEMIES

As the name suggest, they attack in short range. They are the most common type of the game, and that is why they have no other special characteristics, though we can differentiate two subtypes depending on how they attack and move:

3.7.1.1 – MELEE ENEMIES: FIST-ATTACKING TYPE

This is the most basic of the two. When they see Kaede, they approach her at a relatively normal speed and begin attacking her with their fists.

Figure 23: Melee enemy

3.7.1.2 – MELEE ENEMIES: BAT-ATTACKING TYPE

This is a variation of the previous subtype, since it does the same, but these ones have a bat instead, which deal a little more damage to the main character. Besides, these enemies move a little faster than the fist-attacking enemies. These are a more advanced type of enemy and appear later in the game.

Figure 24: Bat enemy

3.7.2 – SHOOTER ENEMY

This type of enemy has a laser gun in his possession and therefore they can shoot our character. They do not need to stand at short range to start attacking, though the bullets they fire deal less damage than a melee attack.

When defending against this enemy, their bullets will bounce from the character and will be redirected to the enemy, so the player can take advantage of this to attack this enemy this way in particular. However, if the player starts guarding too soon, the enemy will see our intentions and will not begin shooting until the player stops pressing the guarding button.

Note that the bullets fired by the enemy will not damage other enemies and will only deal damage to the player. However, if a bullet is returned after bouncing back from the character guarding, this bullet will be able to cause damage to the first enemy that hits.

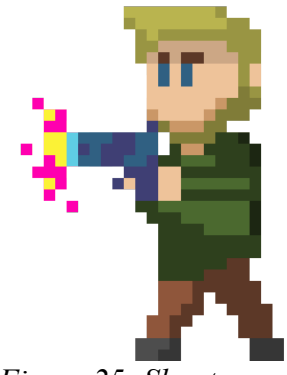

Figure 25: Shooter enemy

3.7.3 – BIG ENEMY

This big sized enemy has the particularity of having more health, causing more damage and pushing Kaede some distance when he attacks. However, they move noticeably slower than

other enemies, and their attack rate is also lower.

Due to the size of this enemy, the character will not be able to throw it into the air after an up attack. Also, if the character is defending against one of this type of enemy attacks, it will not hurt her but it will push her some distance anyway.

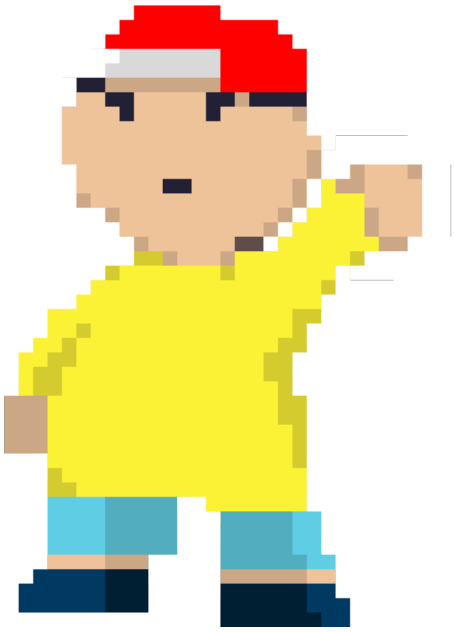

Figure 26: Big enemy

3.7.4 – ELECTRIC MACHINE

This machine is a robot with a simple function: moving left and right alternatively. They travel a set distance and they are extremely heavy. The danger this enemy poses to the character is that it is full of cables that has been cut on purpose in order to cause electric shock to anyone who touches it.

From the player perspective, these machines are invincible and immovable enemies that will not stop moving left and right continuously, so the only solution is to dodge it, either jumping at the right time and walking away quickly or teleporting (if this move is available).

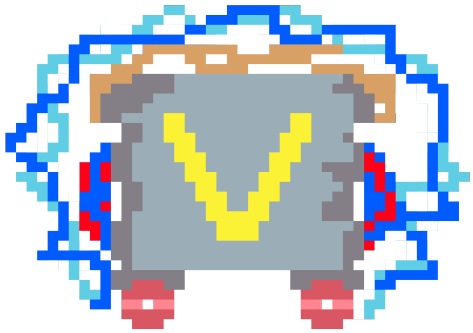

Figure 27: Electric machine

3.7.5 – SHOOTING ROBOT

This machine or robot has the ability to shoot laser bullets. These bullets act the same as the ones fired by the shooter enemy. This machine is too heavy to be moved by Kaede, but unlike the previously explained electric machine, this robot can actually be destroyed.

The robot has a shield protecting it, so melee attacks (neither normal nor strong ones) will not be effective. Instead, the player can use shooting attacks to defeat this enemy since it has an opening in the shield (and that is why they are able to shoot), either shooting themselves with Kaede or defending against the robot bullets and consequently bouncing them back.

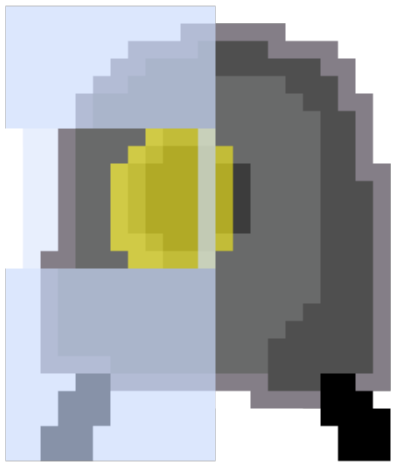

Figure 28: Shooting robot

3.8 – DIALOGUES

During the beat'em up sections of the game, there will not be any dialogues since it is an action-focused gameplay, but dialogues appear in the intro scene, the tutorial and the investigation parts of the game.

Their design is a black box with white letters on it. Also, the name of the speaker is indicated at the top of every dialogue with a characteristic color for each character.

In the investigation parts, the player will not be able to move the mouse when a dialogue is open. To advance to the next dialogue (or close them in case there is no next dialogue) there is a “Continue” button in the lower right corner the player has to click on.

Some objects will trigger different dialogues depending on when the player investigates it, meaning that the player must re-investigate some objects or elements of the stage after checking others.

Just when the player is about to finish the investigation of the stage, and thus finish the level, when Kaede should already know where the criminal is hiding (that is, when the player has investigated everything important), Zero will start questioning her about the conclusion she reached about the whereabouts of the criminal with multi-answer questions. If the player has been paying attention to what they have discovered, they can figure out the correct answers. On the contrary, if they fail, they will lose some energy points.

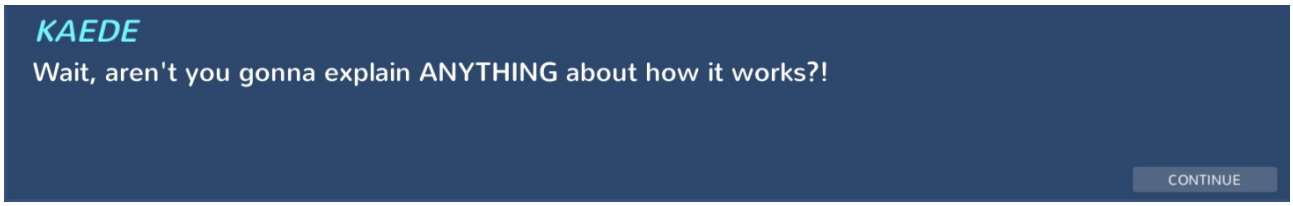

Figure 29: Dialogue in the game

3.9 – MENUS, UI AND HUD

3.9.1 – UI AND HUD DURING GAMEPLAY

The main elements the player can see on screen during the beat'em up sections are:

- **Health bar:** A health-bar is displayed in the top left corner of the screen and indicates the player life or health. This will update accordingly every time the character is hurt or picks up an aid kit, for example. If there is no displayed health left in the health-bar, it means the character has been knocked off. The colors symbolize better her health condition:

  - Blue if her health is greater than 90%.
  - Green if her health is between 60% and 90%.
  - Yellow if her health is between 40% and 60%.
  - Orange if her health is between 20% and 40%.
  - Red if her health is less than 20%.

- **Dash bar:** This bar indicates if the character is able to teleport. It takes some time to recharge after every time the player uses this move. When this bar is fully recharged, it changes its color to a yellowish one. This bar is just below the health bar.

Figure 30: Health and dash bars

- **Time left:** This is a timer indicating how much time does the player have to finish the level (the beat'em up section). It is displayed at the right upper corner of the screen.

- **Number of beaten enemies and enemies left:** The beaten enemies will be displayed in the lower left corner of the screen. Right next to this number, we find the total number of enemies the current level has.

While the elements shown during the investigation part are:

- **Time left:** Indicating how much time does the player have to finish the investigation (the point and click section), specifically until Zero begins his last questioning to Kaede. It is displayed at the right upper corner of the screen.

3.9.2 – GAME MENUS

3.9.2.1 – MAIN MENU

The main menu consists of the next buttons and elements:

- **Play button:** For starting a new game or continuing an already started game, so the players can keep playing from the last point they were in case that it is not the first time they play the game.

- **Game State percentage:** This number indicates the completed percentage of the game. All the goals that count for getting the 100% are the next ones:

  - Finishing all levels successfully: level 0 (tutorial), 1, 2 and 3.
  - Watching all intermission extra scenes: scenes after level 1, 2 and 3.
  - Beating up all enemies in each level (beat'em up sections).
  - Concluding an investigation in a short period of time.

- **Credits button:** This button does not have any effect in the game. Instead, it shows some important information about the project (name and tools used for creating the game).

- **Exit button:** If the player clicks on it, they will quit the game.

Kaede's office is like the main hub of the game and contains this main menu.

Every time the player finishes a level, the progress is saved and every time they turn the game on again, they can continue from the beginning of the next level or from the beginning of the investigation of the last level they have played, depending on where they left.

3.9.2.2 – PAUSE MENU

The pause menu can be accessed anytime during the levels, and contains the following:

- **Resume button:** This closes the pause menu and lets the player continue.

- **Exit button:** This allows the player to go back to the main menu. Note that not finishing the beat'em up part means that the player has to start the level from the beginning.
- **Helps:** They will be covered later.

3.9.2.3 – GAME OVER MENU

When the character health reaches 0, she has been knocked off and the game will display the game over menu. The same happens every time the character falls. Similar to the pause menu, the game over menu contains:

- **Retry button:** To try again from the start if we are in the beat'em up part. In the investigation part, if the time reaches 0, only this part will be started again.

- **Exit button:** it has the same identical function as the Exit button of the pause menu.

3.9.2.4 – LOG IN MENU

Every time a user turns the game on, before going to its main menu, they will have to log in with their name in order to retrieve the walked steps data attached to that name.

This has been meant to be a simple process for the user, since they only need to type the name, and then (after confirming) the steps data will be automatically retrieved and converted to in-game energy points for the character.

The players has the option to skip this process with a Skip button in the right upper corner of this screen. However, no steps will be retrieved this way.

3.9.3 – ENERGY POINTS SHOP

Just before entering a level, Zero will open a panel that the player can use to buy upgrades. These are:

- **Double Health**: This duplicates the default health the character has, so it takes more received damage to being knocked off.
- **Super Strength:** This upgrade increases the damage Kaede deals to the enemies with her attacks.
- **Max Mode:** The player can use some energy points to entering Max Mode. As explained in previous pages, this mode unlocks some new useful moves for Kaede, surrounding her with an energy blue aura indicating she is in Max Mode.

Also, during the investigation in each level, Zero will offer Kaede the following helps (which can be accessed from the pause menu):

- **Puzzle hints:** If the player gets stuck with a puzzle, they can buy a hint from Zero, which will be displayed in the pause menu every time the player opens it.
- **More time:** if the player believes they will run out of time, they can buy some more from Zero. Note that this can only be bought once and the objective is to give more time to the player and not to make these sections infinite.

3.10 – ART STYLE

Pixel Art is the art style for this game, as it can be seen in some of the previous images.

Cartoon art was going to be used in dialogues to show the player the face of the character currently talking, but in the end, that idea has been discarded because it would take some time to draw the character faces using a different art style and since it does not have any other function in the game beyond making the dialogues look better, I considered to use that time to accomplish other more important tasks.

Figure 31: Kaede just after attacking

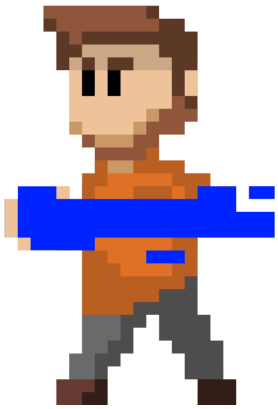

Figure 32: Melee enemy punching

CHAPTER 4: SYSTEM ANALYSIS AND DESIGN

This chapter presents the requirements analysis, design and architecture of the project.

4.1 – REQUIREMENT ANALYSIS

To better understand the project requirements, let us talk about how the game and the app work.

4.1.1 – GAME REQUIREMENT ANALYSIS

Starting with the game (the game loop), when the user opens it, the first screen seen is the log in menu, in which the user can find a blank space to write text, and the “Get Steps” and the “Skip” buttons. The mouse can be used to navigate this screen. When the user types a name (with the keyboard) in the blank space and presses the “Get Steps” button, in the bottom part of the screen, the name entered and the steps data attached to that name will be shown in case that username exists, and a new button will appear: the “Confirm” button. In the same way, if that username entered does not exist, the game will indicate it the user, also in the bottom part of the screen, and this time the “Confirm” button will not appear.

Once an existing name has been entered, if the user presses the “Confirm” button, a panel will appear asking the user if they want to continue with that steps data (the one attached to the username entered). This panel contains two new buttons: the “Yes” button can be pressed to load the main menu screen, while the “No” button closes this panel. If the user decides to continue to the main menu with that steps data, the number of steps is converted to in-game points and the database is updated with the steps value being now zero.

The “Skip” button can be pressed anytime, and it opens a panel that works just like the panel opened by the “Confirm” button, except that the steps data is not updated, neither in the game nor in the database.

The main menu shows three buttons: the “Credits” button shows information about the project, the “Save & Quit” button saves the current game progress and closes the game, and the “Play” button takes the user to the levels menu. This new menu contains a button for each level of the game (Tutorial, 1, 2, 3). The tutorial simply takes the player to the tutorial level, but the other levels open a panel showing a small summary of each case, and the needed points to enter that level. Clicking on the “Start” button takes the player to that level in question and subtracts the cost for entering from the energy points (always there are enough of them). If a level has already been played before, the player can directly access the investigation part of that level with

only half of the necessary points.

There is a “Shop” button in this levels menu, which opens a panel showing the player the upgrades that they can buy with energy points. Clicking on them subtracts the cost to the current energy points the player has (only if they have enough points), and that upgrade will be used for the next level the player enters.

The last button is the extra scene button. This button can be enabled (green color tone) or disabled (gray color tone), so after finishing a level, the player will be able to see an extra scene (enabled) or not (disabled) before going back to the main menu. If enabled, depending on the level, a different amount of energy points will be subtracted after watching the extra scene in question.

All the panels can be closed by clicking on the “Close” button, and the levels menu is shown again. In the levels menu, there is also a “Back” button that takes the player back to the main menu.

The player controls has been indicated in the second chapter of this document (controls section). Basically, in the first part of each level, the character moves, jumps, attacks (in the ground or in the air), shoots, defends herself, and teleports. In the second part, the player interacts with objects, answers multi-choice questions and other elements on the stage and solves puzzles. All these actions trigger different dialogues. In order to advance dialogues, there is a “Continue” button in the dialogue box that displays the next dialogue when it is pressed, and when the last dialogue is shown, the level ends and the game takes the player back to the main menu. A wrong answer or a wrong puzzle solution subtracts 500 energy points from the player.

The player can access the pause menu anytime (within a level). This menu opens when the player presses ‘P’ in the keyboard. Then, the game is stopped and the pause menu is shown, with two basic buttons in it. The “Continue” button closes the pause menu (which can also be closed by pressing ‘P’ again) and resumes the game, while the “Exit” button takes the player to the main menu. In some investigations, the pause menu shows two more buttons: the “Hint” button displays a hint (a text) in the same pause menu, and the “Add time” button adds more time to the timer. Both of these new buttons subtract 500 energy points (indicated in the pause menu anyway).

Last, the game over menu, which opens when the character falls or loses all her health (“beat’em up” parts), or when the time runs out (investigation parts). It shows two buttons: the “Retry” button allows the player to restart the level and the “Exit” button takes the player back to the main menu.

4.1.1.1 – FUNCTIONAL REQUIREMENTS FOR THE GAME

Now that the game requirements has been explained, the functional requirements are clear:

- R1: The player can write their username.
- R2: The player can access the main menu.
- R3: The player can see the credits.
- R4: The player can quit the game.
- R5: The player can access the levels menu.
- R6: The player can access the shop menu.
- R7: The player can buy upgrades and helps.
- R8: The player can enable or disable the extra scene.
- R9: The player can select a level.
- R10: The player can close menus.
- R11: The player can restart the level.
- R12: The player can return to the main menu.
- R13: The player can pause the game.
- R14: The player can move, attack, shoot and guard.
- R15: The player can interact with objects.
- R16: The player can advance dialogues.
- R17: The player can choose an answer.
- R18: The system will be able to know which key objects have been investigated to trigger the appropriate dialogues.
- R19: The system will be able to save the player progress so the energy points will be constantly updated and the player will be able to access previously unlocked levels.
- R20: The system will know if the player has chosen to watch the extra scene before entering a level.
- R21: The enemies will be able to move towards the player's character.
- R22: The enemies will be able to attack the player's character.

4.1.1.2 – NON-FUNCTIONAL REQUIREMENTS FOR THE GAME

- R23: The game will be playable on PC.
- R24: The game will use 2D pixel-art designs.
- R25: The player will have a variety of movements and attacks available.
- R26: The player will be able to investigate the rooms freely.
- R27: The game will show all the necessary information for the player in the UI.

4.1.2 – APP REQUIREMENT ANALYSIS

When the user opens the mobile app, it shows the main screen with the number of steps walked and the current goal in the bottom. Taping on the options menu in the toolbar displays four options:“Reset” zeroes the pedometer, while the other three display other screens:

- In the “Define Goal” screen, the user can select the level they want to access the next time they play the computer game. Switching the extra scene option on adds the extra scene cost to the selected level cost. To apply changes, the user must tap the “Set Goal” button, and then the current goal will be accordingly updated.
- In the “Settings” screen, the user can select the sensitivity level for the pedometer sensor. A higher sensitivity requires larger steps.
- In the “Update Data” screen, the user can type their name and register it (in case that username is new) or update it (in case that username already exists in the database). In any of these two cases, the steps data is uploaded to the database and then the pedometer is restarted. There are some short instructions in the bottom of this screen.

To return the main screen, the user must tap the back button of their device (3 button navigation).

The app works in the background always the user taps the back or the Home button, or locks the device (a short message is displayed during a few seconds telling the user the app is counting steps). In order to stop the app completely, the user needs to clean it from recent apps (and the app will stop counting steps), and the number of steps is saved. The next time the user opens the app, it continues from where the user left off.

4.1.2.1 – FUNCTIONAL REQUIREMENTS FOR THE APP

The functional requirements for the app are then the following:

- R1: The user can access the “Define Goal” menu.
- R2: The user can access the “Settings” menu.
- R3: The user can access the “Update Data” menu.
- R4: The user can select the level.
- R5: The user can add the extra scene cost to the selected level.
- R6: The user can adjust the pedometer sensitivity.
- R7: The user can type a username.
- R8: The user can register a new username with the steps data.
- R9: The user can update an existing username with the steps data.
- R10: The system will start counting steps in background every time the app is stopped

(but not cleaned from recent apps).

- R11: The system manages any case separately in order to avoid uploading incorrect information into the database.

4.1.2.2 – NON-FUNCTIONAL REQUIREMENTS FOR THE APP

- R12: The app runs in Android devices.
- R13: The app will present a clean and simple UI.

4.2 – SYSTEM DESIGN

This section presents the design of the system. In the following pages are defined the cases of use (taken from the player's functional requirements) for the game and for the app.

4.2.1 – GAME SYSTEM DESIGN

Requirement:	R1
Actor:	Player
Description:	The player writes a username.
Preconditions:	1. The player must be in the log in menu.
Steps normal sequence:	1. The player types a name with the keyboard. 2. The player clicks on Get Steps. 3. The system loads the steps data attached to that name.
Alternative sequence:	None.

Table 1: CU01. Write username

Requirement:	R2
Actor:	Player
Description:	The player accesses the main menu.
Preconditions:	1. The player must be in the log in menu.
Steps normal sequence:	1. The player clicks on Confirm, in the log in menu. 2. The system converts the steps data to in-game energy points. 3. The system loads the main menu.
Alternative sequence:	1. The player clicks on Skip, in the log in menu. 2. The system loads the main menu.

Table 2: CU02. Access main menu

Requirement:	R3
Actor:	Player
Description:	The player sees the credits of the game.
Preconditions:	1. The player must be in the main menu.
Steps normal sequence:	1. The player clicks on Credits. 2. The system shows a panel with the game credits.
Alternative sequence:	None.

Table 3: CU03. See credits

Requirement:	R4
Actor:	Player
Description:	The player closes the game program.
Preconditions:	1. The player must be in the main menu.
Steps normal sequence:	1. The player clicks on Save & Quit. 2. The system quits the game program.
Alternative sequence:	None.

Table 4: CU04. Quit game

Requirement:	R5
Actor:	Player
Description:	The player accesses the levels menu.
Preconditions:	1. The player must be in the main menu.
Steps normal sequence:	1. The player clicks on Play. 2. The system loads the levels menu.
Alternative sequence:	None.

Table 5: CU05. Access levels menu

Requirement:	R6
Actor:	Player
Description:	The player accesses the shop to buy upgrades.
Preconditions:	1. The player must be in the levels menu.
Steps normal sequence:	1. The player clicks on Shop. 2. The system opens the shop menu.
Alternative sequence:	None.

Table 6: CU06. Access shop

Requirement:	R7
Actor:	Player
Description:	The player buys upgrades from the shop menu and helps in the pause menu.
Preconditions:	1. The player must be in the shop menu to buy upgrades or in the pause menu of certain investigations to buy helps. 2. The player must have enough energy points. 3. The player must not have bought the upgrade/help yet.
Steps normal sequence:	1. The player clicks on the upgrade or help they want to buy. 2. The system subtracts a certain amount of energy points from the player. 3. The player can use the upgrade or the help in that level or the next one (depending on the item).
Alternative sequence:	None.

Table 7: CU07. Buy

Requirement:	R8
Actor:	Player
Description:	The player chooses if they want to watch the extra scene after finishing the next level.
Preconditions:	1. The player must be in the levels menu.
Steps normal sequence:	1. The player clicks on the extra scene button. 2. The button is now green (enabled) or grey (disabled).
Alternative sequence:	None.

Table 8: CU09. Extra scene

Requirement:	R9
Actor:	Player
Description:	The player selects a game level.
Preconditions:	1. The player is in the levels menu. 2. The selected level (or part) is unlocked. 3. The player has enough energy points to enter the selected level.
Steps normal sequence:	1. The player clicks on the level button. 2. The system subtracts a specified amount of energy points (it depends on the level). 3. The system loads the selected level.
Alternative sequence:	1.1. The player clicks on the level button starting from the investigation part. 2.1 The system subtracts half of the specified amount of energy points (it depends on the level) if the player starts from the investigation part. 3.1 The system loads the investigation part of the selected level if the player starts from that part.

Table 9: CU09. Select level

Requirement:	R10
Actor:	Player
Description:	The player can close different menus of the game and return to the previous state or menu.
Preconditions:	1. The player must have opened a menu.
Steps normal sequence:	1. The player clicks on Continue (pause menu) or Close (any other menu). 2. The system closes the open menu and returns to the previous state or menu.
Alternative sequence:	1.1. The player can also press 'P' to close the pause menu.

Table 10: CU10. Close menu

Requirement:	R11
Actor:	Player
Description:	The current level is restarted from the beginning of that part.
Preconditions:	1. The player must be in a level. 2. The player must be in the game over menu.
Steps normal sequence:	1. The player clicks on Retry. 2. The system loads the current level again.
Alternative sequence:	1.1. If the player is in the investigation part, the system will only restart that part.

Table 11: CU11. Restart level

Requirement:	R12
Actor:	Player
Description:	The player goes from any level back to the main menu.
Preconditions:	1. The player must be in a level. 2. The pause or game over menu must be open.
Steps normal sequence:	1. The player clicks on Exit. 2. The system saves the progress and loads the main menu.
Alternative sequence:	1.1. The game automatically returns to the main menu if the player finishes a level.

Table 12: CU12. Return main menu

Requirement:	R13
Actor:	Player
Description:	The player pauses the game, opening the pause menu and stopping the game.
Preconditions:	1. The player must be in a level.
Steps normal sequence:	1. The player presses 'P'. 2. The system stops the game and opens the pause menu.
Alternative sequence:	None.

Table 13: CU13. Pause game

Requirement:	R14
Actor:	Player
Description:	The player can move, perform melee attacks, shoot with a laser gun or guard against enemy attacks.
Preconditions:	1. The player must be in a "beat'em up" part of a level. 2. The movement hast to be available.
Steps normal sequence:	1. The player presses the correct button. 2. The player moves, attacks, shoots or defends depending on the pressed button.
Alternative sequence:	2.1. If that action is not available at that moment, nothing happens then.

Table 14: CU14. Beat'em up actions

Requirement:	R15
Actor:	Player
Description:	The player interacts with the objects on the stage to trigger dialogues.
Preconditions:	1. The player must be in the investigation part of a level. 2. No dialogues are being displayed at that moment. 3. The player must not be solving a puzzle at that moment.
Steps normal sequence:	1. The player clicks on an object. 2. The system triggers a dialogue.
Alternative sequence:	2.1. Depending on the object, if it has been previously investigated, the system will trigger different dialogues for that same object.

Table 15: CU15. Objects interaction

Requirement:	R16
Actor:	Player
Description:	The player can display the next dialogue or close the dialogue box if there are no dialogues left.
Preconditions:	1. The player must be in the investigation part of a level, in the tutorial or in an extra scene. 2. The dialogue box must be open, displaying text.
Steps normal sequence:	1. The player clicks on Continue, in the dialogue box. 2. The system displays the next dialogue or closes the dialogue box (if there are no dialogues left to be displayed).
Alternative sequence:	1.1. The player presses 'E' in the tutorial level to advance dialogues.

Table 16: CU16. Dialogues

Requirement:	R17
Actor:	Player
Description:	The player chooses an answer for a multi-answer question.
Preconditions:	1. The player must be in a concrete point of the level or the extra scene. 2. A question is being displayed.
Steps normal sequence:	1. The player clicks on A or B to choose their answer. 2. If the answer is correct or allowed, the system triggers a dialogue.
Alternative sequence:	2.1. If the answer is wrong, the system subtracts 500 energy points from the player.

Table 17: CU17. Choose answer

4.2.2 – APP SYSTEM DESIGN

Requirement:	R1
Actor:	User
Description:	The user accesses the Define Goal screen.
Preconditions:	1. The user must be in the main screen.
Steps normal sequence:	1. The user taps the options button in the toolbar. 2. The user taps "Define goal". 3. The system loads the Define Goal screen.
Alternative sequence:	None.

Table 18: CU01. Define Goal screen

Requirement:	R2
Actor:	User
Description:	The user accesses the Settings screen.
Preconditions:	1. The user must be in the main screen.
Steps normal sequence:	1. The user taps the options button in the toolbar. 2. The user taps "Settings". 3. The system loads the Settings screen.
Alternative sequence:	None.

Table 19: CU02. Setttings screen

Requirement:	R3
Actor:	User
Description:	The user accesses the Update Data screen.
Preconditions:	1. The user must be in the main screen.
Steps normal sequence:	1. The user taps the options button in the toolbar. 2. The user taps "Update data". 3. The system loads the Update Data screen.
Alternative sequence:	None.

Table 20: CU03. Update Data screen

Requirement:	R4
Actor:	User
Description:	The user selects the level they want to play next.
Preconditions:	1. The user must be in the Define Goal scene.
Steps normal sequence:	1. The user selects a level (1, 2 or 3). 2. The user taps "SET GOAL". 3. The system updates the current goal with the cost of the selected level.
Alternative sequence:	1.1. If the user does not select any level, the default current goal value will be the cost of the first level.

Table 21: CU04. Select next level

Requirement:	R5
Actor:	User
Description:	The user adds the extra scene cost to the selected level cost.
Preconditions:	1. The user must be in the Define Goal Scene.
Steps normal sequence:	1. The user switches "Add extra scene cost" on/off. 2. The user taps "SET GOAL". 3. The system updates the current goal adding (or not) the cost of the extra scene of the selected level to the cost of that level.
Alternative sequence:	1.1. If the user does not change this switch manually, the default value for the switch will be off.

Table 22: CU05. Add extra scene cost

Requirement:	R6
Actor:	User
Description:	The user adjusts the pedometer sensitivity level.
Preconditions:	1. The user must be in the Define Goal screen.
Steps normal sequence:	1. The user selects the sensitivity level. 2. The system updates the sensitivity of the pedometer with the selected sensitivity.
Alternative sequence:	1.1. If the user does not adjust the sensitivity, the default value will be 6.

Table 23: CU06. Adjust sensitivity

Requirement:	R7
Actor:	User
Description:	The user types the username they want to use for retrieving the steps data in the computer game.
Preconditions:	1. The user must be in the Update Data screen.
Steps normal sequence:	1. The user types a name. 2. The system knows what name has been written.
Alternative sequence:	None.

Table 24: CU07. Type username

Requirement:	R8
Actor:	User
Description:	The user registers a new username in the database with their current steps data.
Preconditions:	1. The user must be in the Update Data screen. 2. A username must be written. 3. The username must be between 3 and 15 characters. 4. The username must be new. That is, it must not exist in the database.
Steps normal sequence:	1. The user taps "REGISTER NEW USER". 2. The system checks if the name is valid. 3. The system creates a new user in the database and assigns the current steps data to that new name.
Alternative sequence:	2.1. Any name that does not meet the preconditions will not be valid, thus nothing will happen and the app will show a message to the user indicating an error has occurred.

Table 25: CU08. Register user

Requirement:	R9
Actor:	User
Description:	The user updates an already existing user in the database with their current steps data.
Preconditions:	1. The user must be in the Update Data screen. 2. A username must be written. 3. The username must be between 3 and 15 characters. 4. The username must not be new. That is, it must already exist in the database.
Steps normal sequence:	1. The user taps "UPDATE USER". 2. The system checks if the name is valid. 3. The system updates the user with that same name in the database, overwriting the new current steps data.
Alternative sequence:	2.1. Any name that does not meet the preconditions will not be valid, thus nothing will happen and the app will show a message to the user indicating an error has occurred.

Table 26: CU09. Update user

4.3 – SYSTEM ARCHITECTURE

The minimum requirements to play the build of this project on a PC are:

- Operating System: Windows 7 SP1, at least.
- A CPU with x86 or x64 architecture with SSE2 instruction set support, at least.
- A GPU with DX10, at least.
- A keyboard and a mouse (or a touch panel).

The requirements have been taken from Unity documentation (Unity 2020.2.3f1).

The app requires an Android device with Android version 4.4 or 5.0 at least (not tested in those versions).

CHAPTER 5: DEVELOPMENT AND CODE EXPLANATION

Since no source code should be included in this document (according to the regulations), this chapter talks about how certain functions were developed and implemented. Note that this is not a chapter in which I will show the reader every piece of code or script of my project, since there are many scripts. Instead, I will focus on explaining what I considered some of the most particularly difficult or interesting functions of my project, with the intention of giving the reader an idea in the most understandable way I can, and to elaborate the results of this project at the end of this chapter.

5.1 – SOME GAME FUNCTIONS EXPLANATION

5.1.1 – SOME CHARACTER MOVES

Apart from the basic moves, the character has some other moves and attacks, more complex from a programming point of view.

First, the triple combo, which can be performed pressing C three times in a row. The idea is "rewarding" the player with an attack that deals more damage than the previous one each time. Of course, attacking exposes the player to enemy attacks, so if the player can time up these attacks correctly, they can deal considerable damage to the enemy.

In order to let the character do this, the player needs to press C three times quickly, since not doing it fast will not end up performing the combo. For this purpose, a combo timer has been created and it runs internally, as well as defined a time between attacks. Basically, every time the player press C, a normal attack is performed and the timer begins running down from the established time between attacks and 0. If the player presses the button again before the timer hits 0, then a second attack will be performed, being more powerful than the first one. The same thing occurs with the third attack. On the contrary, if this combo timer hits 0, the combo will be restarted and the next normal attack will be the first one. Since the character is exposed to take damage from enemy attacks while she is attacking, the idea is to encourage the player to attack continuously and to strategically use the character guard in the right moment, so the wasted time in that move will be minimal and a combo can be performed before the time between attacks reaches 0, dealing a lot more damage to the enemies when all these moves are dominated and correctly combined one after another.

There is no fourth attack, so the next attack after the third one will be the first attack again, which will restart the combo.

Another advanced movement of the character is the up attack. This attack throws the enemy to

the air so the character can jump to hit the enemy, who will be immobilized for a few seconds.

This is possible thanks to the Rigidbody component attached to the enemy. There are three functions the enemy scripts have which are executed one after another to accomplish this.

- The first function is entered after hitting the enemy with an up attack, and changes their Rigidbody from Dynamic to Kinematic so the gravity does not affect them. Then, the enemy begins going up until the next function is called.
- The second of these functions stops the enemy in the air, so it is called with an Invoke after a couple of tenths of a second (in order to adjust the enemy reached height to the height reached by the character when jumping). After a few seconds, the third function is called.
- The third function, and thus the last one, is called to make the enemy fall to the ground after some seconds of waiting (also controlled by an Invoke method). Their Rigidbody type is now Dynamic again and gravity affects the enemy.

The objective of all this is to give the player a chance to attack the enemy while they are suspended in the air, but only one enemy at a time is affected by this up attack. The player then can air-kick the enemy or hit it with an air strong attack (if available), which will push the enemy some distance making them fall in that instant.

Some other adjustments were made, such as making the character not move when she attacks, to not attack when she is defending or to not shoot while jumping. A strong attack has a lower attack rate than normal attacks, due to those attacks being more powerful than normal ones, so they left the character a little more time exposed after performing them (so she can not attack or move in those tenths of a second).

To check what enemies has been hit with an attack, I set an attack point attached to the character. Using the Collider2D functions, it is checked if a circle collider overlaps with these enemies' colliders each time an attack is executed (that means, causing damage if they are in range). For the area attack (the one executed after press down arrow and X, the strong attack key), another attack point was set, but behind the character this time, so only when this attack is used, two circle colliders check if they are overlapping with enemies behind and in front Kaede, so the damage is applied to all of them. Of course, these attack points flip with the character every time the player changes the movement direction.

5.1.2 – ENEMY BULLETS BOUNCING BACK

As I explained previously, when the character is defending and an enemy fires a bullet, this bullet will bounce back to the enemy, causing them some damage if it hits.

This has been made by checking if the bullet hits the character and she is defending at the collision moment, causing the enemy bullet to have an inverse velocity and going back to the enemy who shot it. In order to damage an enemy this way, only a reflected bullet will make this possible, so it is necessary to store if the bullet has been reflected with a variable which will be checked before applying damage to the enemy in case this bullet hits them.

An enemy bullet is not reflected if the character shoots another bullet. Instead, both bullets will burst in the air, and they will not deal damage to anyone.

5.1.3 – CLICKABLE SCRIPT FOR GAME OBJECTS

During the investigation parts of the game, the player can click on the objects around the stage. Each one of them will trigger a different dialogue. Since the dialogue system was made to be easy to customize from the Unity inspector, this does not require further explanation for the purpose of this document section, the dialogue scripts are assigned to all those objects and different dialogues have been written for each instance of the script.

However, since all these objects are connected in one single case, there is an order to investigate some of them and that is how I control if the player has seen all important dialogues before the questioning that closes the case. This means that there are objects whose dialogues will not make sense if the player has not investigated other objects first (and thus, read the dialogues about them), and that is why a series of static variables has been used to act like indicators about what the player has already discovered and what they have yet to discover. Checking those variables in a single script attached to all objects capable of triggering new dialogues makes possible to change already set dialogues depending on the situation, and the players will be able to read these new dialogues if they click on the object in question at the right time. To understand this better, let us use an example that can be seen in the game: if the player investigates what appear to be a paper with a series of numbers, it will not catch the character's interest, and the dialogues about it will reflect these feelings, but if the player already knows that there is a locked safe in that room, the paper with the written numbers has some meaning now: it is actually the safe password. If the player clicks on this object now, it will trigger a different dialogue. This same system is applied for more objects in the game.

5.2 – MOBILE APP: PEDOMETER FUNCTION

The pedometer function of the mobile app has been made with Android Studio, using the Sensor Manager and some of their methods. The idea was making and app that could count steps in a precisely enough way and with the app running in the background, since the user will probably want to keep their phone in their pocket with the screen turned off due to the nature of the app.

Although it seemed a relatively accessible process using Sensor Manager, a lot of time has been required until getting it working correctly, especially because I had to develop two entire different versions since the first one did not provide the expected results.

5.2.1 – STEP COUNTER SENSOR

The first version of the pedometer was coded with Kotlin, and made using the Step Counter Sensor (*Sensor.TYPE_STEP_COUNTER*), registering its values and updating the displayed information when detecting events (*SensorEvent*). The user can see this information with a number of steps walked displayed on screen, in addition to a circular progress bar, already developed (by a Github user called Lopez Mikhael), that I implemented in my Android project. This is basically a circular shaped progress bar, with an amount of customizing options. In my case, I used it to show the progress of the steps the user has walked, with the circle progress bar being full when these steps are equal or great than the established goal.

Also, I defined SharedPreferences to save/load the steps every time the user turns the app off/on. This way, the steps are not reset when closing the app. That means that the user will not be able to reset the steps except if they reinstall the whole app each time they wanted to do so, so that is why I implemented a Reset button in the toolbar that set the steps to 0 each time the user taps on it in case they need it.

However, when the test time came, the app was not responding correctly. I could observe that it worked, though in a very lagged way, counting a great amount of steps each time. Also, the displayed steps were not too precise. A small margin of error can be accepted with this kind of app, especially if these steps are going to be converted to in-game points later, but actually the displayed number of steps and the actual walked steps were too different, and that needed a fix. Besides, I noticed the app worked better in some phones and worse in other ones, and the newness of the phone did not seem to be the case according to the testing results. Because all these things, I dropped this app version and began searching for other alternatives for its implementation.

5.2.2 – STEP COUNTER USING THE ACCELEROMETER

The second implementation consists in using another Android Sensor: the accelerometer, included in most of the smartphones nowadays. This time, I ended up using Java instead due to having worked with it in some subjects of the degree.

The screen display was the same as the older version, as well as the reset button, and I used SensorManager this time as well, though specifically I used *Sensor.TYPE_ACCELEROMETER*. Due to setting it so there is a walking threshold the user has to surpass, calculating the module of the acceleration vector in order to get the magnitude of the user movement in a real space (subtracting the Earth gravity from the formula, since it caused to start counting steps always from 1 instead of 0), it was possible to get an accurate result. However, it depends on someone's walking to get more or less accuracy, and that is why I implemented an option to let each user set the sensitivity according to their walking. This sensitivity is the walking threshold variable, and right now a high number requires larger steps while a low number requires small steps to count.

However, it is important to know that unlike the step counter sensor, the Android accelerometer does not work in the background. That means the developer has to program this as a new function. First, I tried to do a *wake lock*, so if the user locked the screen, it will not actually stop the CPU from working. However, this function does not work well in all phones (each one takes care of that differently, so I would have to consider many different settings), and also the battery gets drained much faster with this kind of lock since the CPU does not stop working while locked.

What I did at the end was to develop the same code as before (the pedometer code) but in an Android Process instead of an Activity. Android Processes work as a normal app while the program actually runs in background, so the accelerometer can actually work in the background this way without consuming more battery than necessary. After setting all necessary things, it ended up working correctly this time. Of course, the app has more activities, so I have taken care of this process so it does not count steps when the user is changing the sensitivity or establishing their current goal for example. Also, if the user cleans the app from recent apps, the app will stop counting steps. This is something that already occurs by default and has been left this way, so the user can simply clean recent apps to stop this one too.

5.3 – MOBILE APP: UPDATING STEPS DATA

Since this app can be used by anyone with a device eventually provided with an internet connection, a server was needed to act as the intermediary between the mobile app and the computer game. But first things first, the app counts steps and these steps have to be stored somewhere, so using XAMPP and SQLyog (two programs I briefly described in previous chapters) it has been possible to create a local database with SQLyog. This database only has one single table ("users"), which stores a username (that acts as the primary key) and the steps the user has walked. All this was done without any coding. The next step was coding the SQL queries and methods of the app that allow the data storage and management.

5.3.1 – WEB SERVICE SQL FILES AND QUERIES CREATION

I created a php file, which was edited using Notepad++. First, I worked with a local database (which was going to be made remote later), so I set the hostname, database name, username and password according to the local database data. This php gets the name of the user and the steps so it inserts (INSERT) a name and the steps data to the database as a new row. While the program has internet connection, it does not throw an error, so an if-else section was added in which the program checks if the name already exists in the database (a query consult with SELECT). If it does not exist, the name is properly registered in the database, and if it already does exist, the name is not inserted and returns the value 'Registered'.

Then I created a second php file, but this time it gets a name and consults if it exists. If this is the case, the row with the specified name is updated (UPDATE) with the new steps data. Like the INSERT php file, it only throws an error if there is no internet connection, so differentiation between getting a name that already exists and getting one that does not is needed. For that purpose, I coded a simple consult looking for that name, so the result returns a number of rows (*mysqli_num_rows($consult)*). If it is higher than 0, then that means the name already exists and it can be updated as the user wish. If it returns 0, then than name does not exist and the user will have to register it as a new one first. A non-existent name can not be updated.

Both consults (the one of the INSERT php file and the one of the UPDATE php file) were implemented in a new php file first, but since it needs to return something different each time, I removed this new php file and kept the two mentioned, coded as I have just stated. This process took days of work.

5.3.2 – HANDLING SQL ORDERS IN ANDROID STUDIO

The display for the data upload screen is basically composed by an Edit Text in which the user will introduce its name (that must be between 3 and 15 characters) and two buttons, one for

handling the register of the user (INSERT query) and another one for updating an existing user (UPDATE query). In order to do this, I had to implement Json Object Request and String Request.

Then, I created two Java methods. One called after clicking the register button and another one called after clicking the update button. They look similar except for the String Request, which is different in each case, and therefore each one has access to a different php file. For the IP, as I said earlier, I used a local database first, so the IP was my PC IP then.

The app notifies the user about every case with a message (*Toast*), indicating them if they have introduced a not valid name, tried to register an already created one or they try to update a non-existent user, handling every case correctly so the database will not insert/update anything unnecessary by mistake.

5.3.3 – CONVERTING THE LOCAL DATABASE TO A REMOTE ONE

For this task, I needed an online server. Initially, Amazon Web Services (AWS) was going to be used for this purpose. However, this service is not free forever, but only the first month during 750 hours. Then, 000WebHost was used. This page offers free services, though they are fairly more limited than commercial hosting services. For the purpose of the project, the offered space was enough though, since I manage a single database with a single table with only a few columns.

Using SQLyog, I exported the local database to this new created web service and converted it to a remote one. The Android Studio project was consequently modified with a different IP, now a remote one provided by 000WebHost. I kept things similar to the previous implementation so the changes were minimal. The remote database testing has been made with phpMyAdmin.

5.3.4 – RETRIEVING THE DATA FROM UNITY

Similarly to Android Studio, Unity can access php files with specific instructions to work with databases and the network (*UnityEngine.Networking*, *WWW*...). Two new php files have been created. The first one consults (SELECT) the specified user in the database, while the second one update (UPDATE) the database, setting the number of steps to zero for the previously specified user, so the user can not retrieve the steps data more than once. If the user name exists, then the steps data is retrieved and automatically converted into energy points, indicating the user the operation was successful and updating the energy points in the game with this new data.

CHAPTER 6: RESULTS, FUTURE WORK AND CONCLUSIONS

6.1 – OBTAINED RESULTS AND FUTURE WORK

Personally, I think it all ended up pretty well. I could create the project as I was expecting to and I could solve the problems that appeared during the development with alternative solutions or different implementations.

I could have implemented larger stages, or maybe more of them. Also, I would have liked to compose my own OST for the game, but that work requires a lot of time I do not have for now (not only creating the music, also learning how to use some program for that purpose), but in the end everything needed is working well, and all game and app functions are there, so the only thing left is expanding the project with more content generally speaking. For now, I believe that what I have done serves the purpose for this project. Expanding the game with more content may be something I would like to do in the future, or maybe to create a new game starting from this same concept.

6.2 – CONCLUSIONS

I can say this project has been possible thanks to all the programming (either with Unity or Android Studio) knowledge and practicing I have acquired these past years.

It is pretty satisfying to see how the project has evolved during these months and I accomplished what I wanted to according to my initial idea for the work, especially taking into account that there are some important things I never did before but I learned to during the course of this project. It has been often tough to balance and combine the work of this project and other subjects I have, but I was able to organize myself despite all that.

Another really important thing to organize myself was keeping a working routine every week, so I reserved a number of hours per week (and thus, per day too) to work on the project since the beginning until the ending of it.

Now I feel confident and ready to start working in this industry in a professional way.

BIBLIOGRAPHY

(1) UJI. Aula Virtual VJ1241. Memorias de referencia. https://aulavirtual.uji.es/mod/folder/view.php?id=4482344 (Accessed 2021-06-21)

(2) Unity Technologies. Unity download archive. https://unity3d.com/get-unity/download/archive (Accessed 2021-06-21)

(3) Microsoft. Visual Studio download. https://visualstudio.microsoft.com/es/downloads/

(4) Microsoft. Visual Studio Code download. https://code.visualstudio.com/download

(5) Adobe. Adobe Photoshop. https://www.adobe.com/es/products/photoshop.html

(6) Aseprite. Aseprite. https://www.aseprite.org/

(7) Android. Android Studio download archive. https://developer.android.com/studio/archive

(8) Notepad++. Notepad++ download archive. https://notepad-plus-plus.org/downloads/

(9) Webyog. SQLyog Community download archive. https://github.com/webyog/sqlyog-community/wiki/Downloads

(10) Apache Friends. XAMPP download. https://www.apachefriends.org/download.html

(11) Amazon. Amazon Web Services. https://aws.amazon.com/es/

(12) 000webhost.com. 000Webhost webpage. https://www.000webhost.com/

(13) LibreOffice. LibreOffice download archive. https://downloadarchive.documentfoundation.org/libreoffice/old/

(14) phpMyAdmin. PhpMyAdmin webpage. https://www.phpmyadmin.net/

(15) Unity Technologies. Unity Asset Store. https://assetstore.unity.com/

(16) Google. YouTube. https://www.youtube.com/

(17) Github. Github Desktop download. https://desktop.github.com/

(18) Brackeys. 2D-Character-Controller. https://github.com/Brackeys/2D-Character-Controller

(19) Stack Overflow. Stack Overflow webpage. https://stackoverflow.com/

(20) Wikipedia. Beat'em Up. https://en.wikipedia.org/wiki/Beat_%27em_up

(21) Wikipedia. Adventure Game. https://en.wikipedia.org/wiki/Adventure_game#Point-and-click_adventure_games

(22) Atlassian. Trello webpage. https://trello.com/es

(23) Monday.com. Monday webpage. https://monday.com/

LIST OF FIGURES AND TABLES

FIGURES

TABLES

Sergio Badia Ganau

Final Degree Project

Video Game Design and Development

Universitat Jaume I

01.07.2021