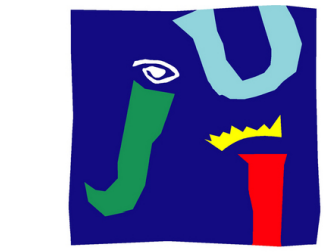

UNIVERSITAT JAUME·I

Development of 2.5D visual puzzle game through a dimension change mechanic

Sergio Gómez García

Final Degree Work

Bachelor's Degree in
Video Game Design and Development

Universitat Jaume I

June 28, 2024

Supervised by: Sandra Catalán Pallarés, PhD.

To my mother, my father, my sister and my friends for being by my side on this arduous path of personal evolution.

Acknowledgments

First of all, I would like to thank my family for watching with enthusiasm how the project was progressing and encouraging me to continue.

Special thanks to my great friend Pau for testing each version that I sent him in order to improve the experience.

Thanks to my sister for the help in the artistic part of the video game.

Of course, I would also like to thank my supervisor Sandra Catalán Pallarés for her patience and effort in carrying out this project.

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

Abstract

This document presents the project report of the Video Games Design and Development Degree Final project by Sergio Gómez García. It is a video game titled *Salva a tu humano* that consists in a set of visual puzzles that you must overcome if you want to save a kidnapped human represented by the main character. To pass the levels it is necessary to use a mechanic that allows you to exchange between a 2D world and a 3D one. In addition, this mechanic is combined with a shooting system that allows you to advance through different puzzles.

Keywords: Roguelike, Switch dimension, Shoot, Puzzle, 2D to 3D

Contents

CHAPTER 1

INTRODUCTION

Contents

This chapter is an explanation of the initial motivations and objectives that have driven the development of this project as well as how it began to be implemented.

1.1 Work Motivation

I have always found it interesting and enriching to implement game mechanics that marked my path as a gamer.

When I was little my parents gave me the Wii [14] and, some time later, Super Paper Mario [13], a game that would remain in my mind forever. In this game you could alternate between 2D and 3D views in order to find hidden areas, overcome visual puzzles, reveal secrets, etc. On the other hand, many years later, I discovered a roguelike called Enter the Gungeon [12] that I loved from the first moment. Its fluid and comfortable gameplay made the game a very dynamic experience. Due to the admiration I felt for this game I decided to investigate how it had been programmed and, to my surprise, it was not a game designed in a 2D world but, thanks to some visual tricks, it was implemented in a 3D world which, apparently, provided certain facilities when developing some mechanics. Upon discovering this I thought it would be a good idea to combine the best of Super Paper Mario with the best of Enter the Gungeon.

My main motivation has been to be able to design and program a video game that fuses the most important mechanics of both games, that is, the change of dimension and a shooting system in a roguelike game.

1.2 Objectives

The main objective of this project is to design and implement a complete game based on visual puzzles, utilizing original mechanics, and ensuring that the movement and shooting systems are especially fluid and comfortable for the player.

More specifically, and based on the motivation behind this project, these are the objectives to achieve:

- Develop a game with two main mechanics: alternate between 2D and 3D views and a shooting system.
- Program a polished player movement based on the context and genre of the video game.
- Design and implement different and original visual puzzles in which the player needs to use both main mechanics.
- Create an immersive experience in which the player feels that their actions really matter.
- Get a build of a game that provides a complete experience from start to finish.

1.3 Environment and Initial State

Initially the idea of the project was to create a complete roguelike game in all its aspects but discovering that my reference game in this genre, Enter the Gungeon, was actually a visually 2D game developed in a 3D environment made me think about the possibility of creating a game that mixed the best of the roguelike genre with the aforementioned dimension change mechanics of Super Paper Mario on the Wii.

Once the base idea was clear, I thought about what key components the game should contain, such as the way to implement the mechanics, the design of the possible visual puzzles to implement, scenes and the story that surrounds the context of the game.

This project has been developed solely by me, Sergio Gómez García in the well-known Unity video game engine [8]. For the assets, the most of them have been chosen from websites such as the Unity Asset Store or Itch.io [2] since this project is focused on programming mechanics. The only assets that are not taken from these web pages have been the antagonist character and the images that act as a tutorial, for all of which I have had the help of my sister.

When I started developing it I had a lot of free time despite working in the afternoons, so I was able to move forward and create a movement and shooting system as polished

as I wanted but, with the arrival of exams, assignments and external practices I had to take a break until that I was a little freer.

CHAPTER 2

Planning and resources evaluation

Contents

This chapter is the most technical of the entire document and is where the project planning and resources will be specified.

2.1 Planning

Planning a project is essential to be able to organize the different goals and objectives that you want to achieve. It is for this reason that below is explained the planning that has been used for developing *Salva a tu humano*.

The order of the tasks has not been strictly linear since some have been carried out at the same time as others.

This section also includes a Gantt chart showing in a visual way how each of this tasks has been done (Figure 2.1).

- **Task 1 (15 hours):** Study the different ways in which a 2.5D video game can be implemented and decide which one best suits my project.
- **Task 2 (10 hours):** Choice of the types of sprites and art that would be used in the video game, taking into account the type of movement intended for the player.
- **Task 3 (30 hours):** Player movement programming. Different possibilities were considered when programming the movement and, once the decision was made, it

was implemented together with a dash and a crosshair that influences the rotation of the character and the position of the camera.

- **Task 4 (40 hours):** Understand how to develop the dimension change mechanics and implement it in the video game. The initial version was modified until the polished result that was sought was achieved.
- **Task 5 (40 hours):** Build a system that allows the weapon to rotate towards the location where the player is aiming, considering that it must change hands if a certain degree of inclination is exceeded. This task also includes the animation of the weapon and the ability to shoot.
- **Task 6 (25 hours):** The implementation of animations for the character, the weapon and its bullets.
- **Task 7 (60 hours):** Design and programming of visual puzzles within which we find interactions with objects such as levers, buttons and torches, rotations of structures and enemies.
- **Task 8 (20 hours):** Creation of other components of the video game such as the menu, UI, the sound section and videos.
- **Task 9 (60 hours):** Project report and other documents.

2.2 Resource Evaluation

The resources used for this project have been:

- **HP Pavilion i5 CPU, 16GB RAM and NVIDIA GeForce GTX 1050 GPU (600€):** The laptop used to develop the game.
- **Unity 3D 2022.3.12f1 (Free):** The video game engine used to create the project [8].
- **Visual Studio 2019 (Free):** It is an IDE that, attached to Unity, makes easy the task of programming, in this case, with C Sharp [6].
- **Github Desktop (Free):** A tool used for version control in which I have a repository where I upload every progress in the project [5].
- **Pyxel Edit (9€):** It is a program to create and edit pyxel art [4].
- **Unity Asset Store (Free):** A website where everyone can obtain price or free assets. In my case every asset i had downloaded has been free [9].
- **Itch.io (Free):** As Unity Asset Store, it is website for obtain free and paid assets. Afresh, every asset i had downloaded has been free [2].

- **Capcut (Free):** It is a video editor that I have used for making every video in the game [1].
- **Voicemaker (Free):** A website where you can convert text to speech [10].

Considering the cost of the materials used, both software and hardware, the total cost of the project would be approximately 609 euros.

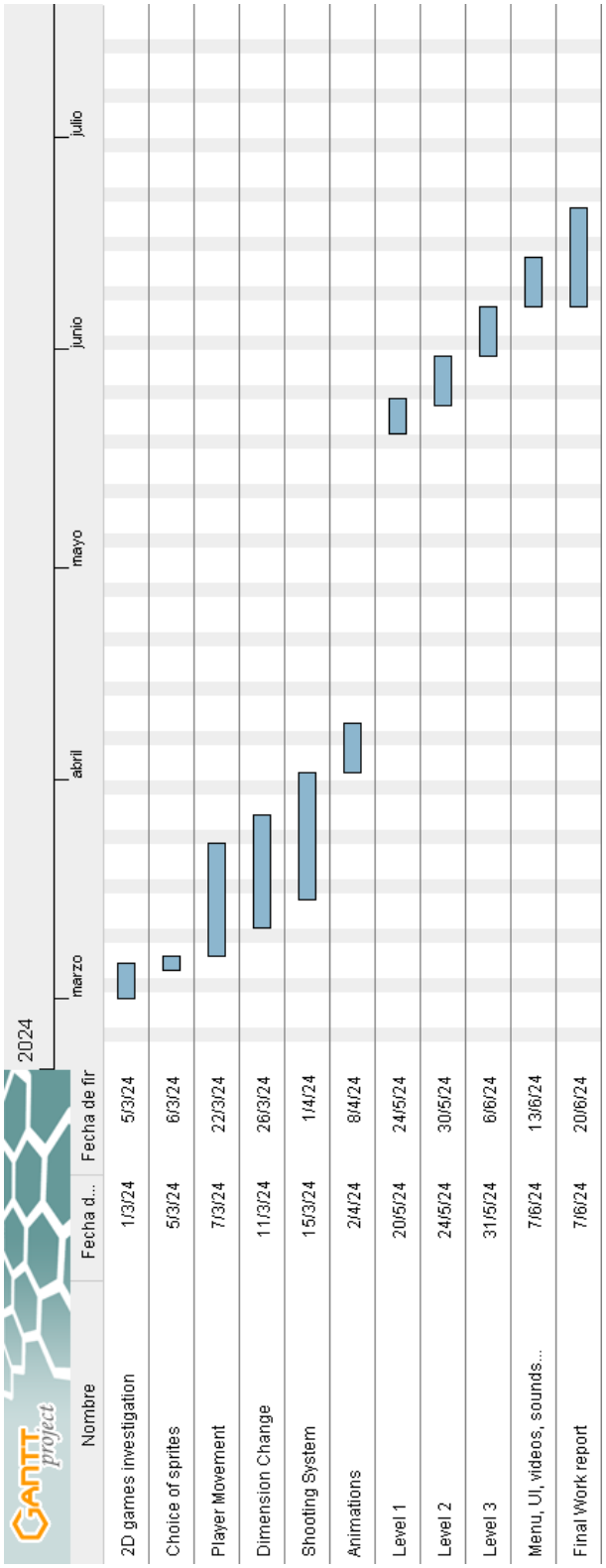

Figure 2.1: Gantt chart of the tasks (made with Gantt Project) [11].

CHAPTER 3

System Analysis and Design

Contents

This chapter presents the requirements analysis, design and architecture of the proposed work, as well as its interface design.

3.1 Requirement Analysis

To carry out a job, it is necessary to make clear both functional and non-functional requirements. It is for this reason that the same are explained in detail below.

3.1.1 Functional Requirements

A functional requirement defines a function of the system that is going to be developed. In this project we can find these functional requirements:

- **R1.** The player can start the game.
- **R2.** The player can quit the game .
- **R3.** The player can move.
- **R4.** The player can switch between dimensions.

- **R5.** The player can interact with levers.
- **R6.** The player can aim.
- **R7.** The player can shoot.
- **R8.** The player can activate colored buttons.
- **R9.** The player can light torches.
- **R10.** The player can dash.
- **R11.** The player can change the color of the bullets.
- **R12.** The player can interact with signs to see instructions.

3.1.2 Non-functional Requirements

Non-functional requirements impose conditions on the design or implementation. In this project we can find these non-functional requirements:

- **R12.** The game will be playable on PC.
- **R13.** The game will use both sprites and 3D models.
- **R14.** The player will have to think to solve puzzles in a certain time.
- **R15.** The puzzles will be solved using dimension change and weapon mechanics.
- **R16.** The UI will be simple for not obstructing the vision of the player.
- **R17.** The movement control will be extremely comfortable for the player.
- **R18.** The player will feel truly immersed in the game.

3.2 System Design

This section must present the (logical or operational) design of the system to be carried out. In the following pages the cases of use (detailed from Table 3.1 to Table 3.12) and a case of use diagram (Figure 3.1) are defined:

Requirements:	R1
Actor:	Player
Description:	The player can start the game.
Preconditions:	1. The player must be in the main menu.
Normal sequence:	1. The player press the *Play* button. 2. The game loads the introduction video.
Alternative sequence:	None.

Table 3.1: Case of use «CU01. Start game».

Requirements:	R2
Actor:	Player
Description:	The player can quit the game.
Preconditions:	1. The player must be in the main menu.
Normal sequence:	1. The player press the *Exit* button. 2. The game system quits the game.
Alternative sequence:	None.

Table 3.2: Case of use «CU02. Quit game».

Requirements:	R3
Actor:	Player
Description:	The player moves trough the level.
Preconditions:	1. The player must be in some of the three visual puzzles. 2. The video of the current level must have finished. 3. The player must have the instructions panel closed.
Normal sequence:	1. The player press *W, A, S* or *D* keys. 2. The player moves in the corresponding direction to that key.
Alternative sequence:	1. The player can not move because some instruction panel is open.

Table 3.3: Case of use «CU03. Move».

Requirements:	R4
Actor:	Player
Description:	The player alternates between two and three dimensions.
Preconditions:	1. The player must be in some of the three visual puzzles. 2. The video of the current level must have finished. 3. The player must have the instructions panel closed. 4. The player is not in the process of changing dimensions.
Normal sequence:	1. The player press *Space* key. 2. The dimension changes from 2D to 3D.
Alternative sequence:	The dimension changes from 3D to 2D.

Table 3.4: Case of use «CU04. Alternate dimensions».

Requirements:	R5
Actor:	Player
Description:	The player activates a lever.
Preconditions:	1. The player must be in the first or second level. 2. The video of the current level must have finished. 3. The player must have the instructions panel closed. 4. The player is not in the second dimension. 5. The player is inside the lever zone.
Normal sequence:	1. The player press E. 2. The lever makes an animation. 3. A pillar is rotated showing a secret.
Alternative sequence:	None.

Table 3.5: Case of use «CU05. Activate lever».

Requirements:	R6
Actor:	Player
Description:	The player aims with the mouse.
Preconditions:	1. The player must be in some of the three visual puzzles. 2. The video of the current level must have finished. 3. The player must have the instructions panel closed. 4. The player is not in the third dimension.
Normal sequence:	1. The player moves the mouse. 2. The cross hair moves to the corresponding point on the screen.
Alternative sequence:	None.

Table 3.6: Case of use «CU06. Aim».

Requirements:	R7
Actor:	Player
Description:	The player shoots with the gun.
Preconditions:	1. The player must be in some of the three visual puzzles. 2. The video of the current level must have finished. 3. The player must have the instructions panel closed. 4. The player is not in the third dimension. 5. The player has get the gun.
Normal sequence:	1. The player presses the left click of the mouse. 2. A bullet is shot.
Alternative sequence:	None

Table 3.7: Case of use «CU07. Shoot».

Requirements:	R8
Actor:	Player
Description:	The player activates colored buttons.
Preconditions:	1. The player must be in the second level. 2. The video of the current level must have finished. 3. The player must have the instructions panel closed. 4. The player is not in the third dimension. 5. The player has get the gun.
Normal sequence:	1. The player aims to a colored button. 2. The player presses the left click of the mouse. 3. A bullet is shot. 4. The bullet hits the button. 5. The button makes an animation. 6. The number related to the color is added in the code.
Alternative sequence:	None.

Table 3.8: Case of use «CU08. Activate button».

Requirements:	R9
Actor:	Player
Description:	The player lights torches.
Preconditions:	1. The player must be in the first or third level. 2. The video of the current level must have finished. 3. The player must have the instructions panel closed. 4. The player is not in the third dimension. 5. The player has get the gun.
Normal sequence:	1. The player aims to a torch. 2. The player presses the left click of the mouse. 3. A bullet is shot. 4. The bullet hits the torch. 5. The torch is lit with a flame of the same color as the bullet.
Alternative sequence:	None.

Table 3.9: Case of use «CU09. Light torch».

Requirements:	R10
Actor:	Player
Description:	The player dashes.
Preconditions:	1. The player must be in some of the three visual puzzles. 2. The video of the current level must have finished. 3. The player must have the instructions panel closed.
Normal sequence:	1. The player presses the right click of the mouse. 2. The player makes a dash.
Alternative sequence:	None.

Table 3.10: Case of use «CU010. Dash».

Requirements:	R11
Actor:	Player
Description:	The player changes the color of the bullet.
Preconditions:	1. The player must be in the second room of the third level. 2. The video of the current level must have finished. 3. The player must have read the instructions panel. 4. The player has get the gun.
Normal sequence:	1. The player rolls the mouse wheel up or down. 2. The bullet color changes. 3. The UI that shows the bullet color changes.
Alternative sequence:	None.

Table 3.11: Case of use «CU011. Change bullet color».

Requirements:	R12
Actor:	Player
Description:	The player interacts with instructions signs
Preconditions:	1. The player must be in some of the three visual puzzles. 2. The video of the current level must have finished. 3. The player is not in the third dimension. 4. The player is inside the sign zone.
Normal sequence:	1. The player press E. 2. The panel makes an animation.
Alternative sequence:	1. The player press X. 2. The panel closes.

Table 3.12: Case of use «CU012. Interact with sign».

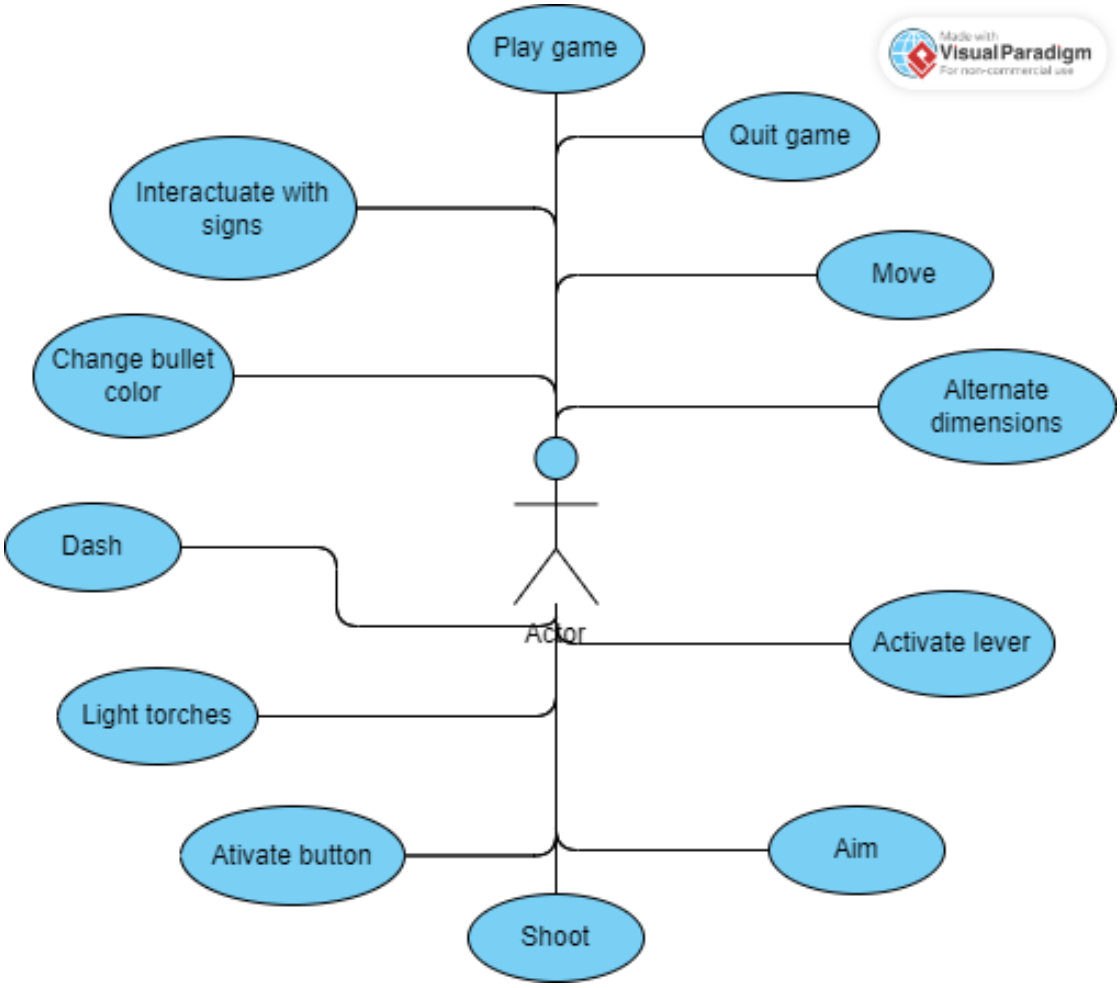

Figure 3.1: Case use diagram (made with Virtual Paradigm) [3].

3.3 System Architecture

The minimun requirements to play the build of this project in a PC are:

- The operating system Windows 7 (SP1+).
- A CPU with x86, x64 architecture with SSE2 instruction set support.
- A GPU with DX10.
- 8GB of RAM.

The requirements have been taken from Unity [8] documentation.

3.4 Interface Design

The UI design is determined by the needs of each puzzle. In each level, there is a sign that provides essential information needed to solve the puzzle. For example, in the first level, it explains how to switch dimensions, in the second one, it provides the code you need to obtain and in the third level, it informs the player that they need to memorize the relationships between letters and colors, in addition to a second sign that explains how to change the color of the bullets (Figures 3.2 to 3.6).

Figure 3.2: Interactive sign.

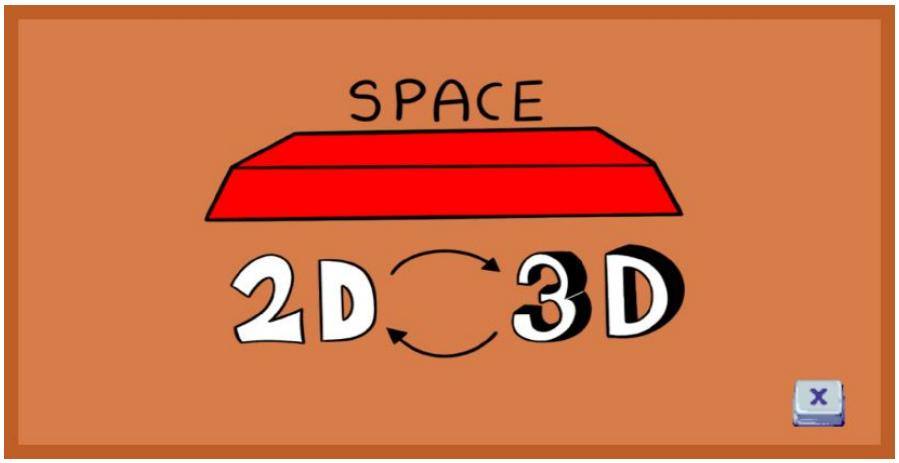

Figure 3.3: Instructions panel level 1.

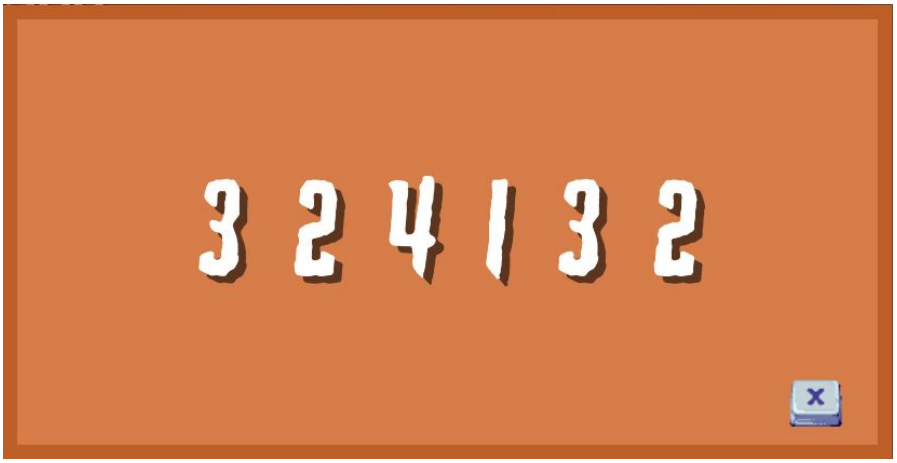

Figure 3.4: Instructions panel level 2.

Figure 3.5: Instructions panel level 3 Room 1.

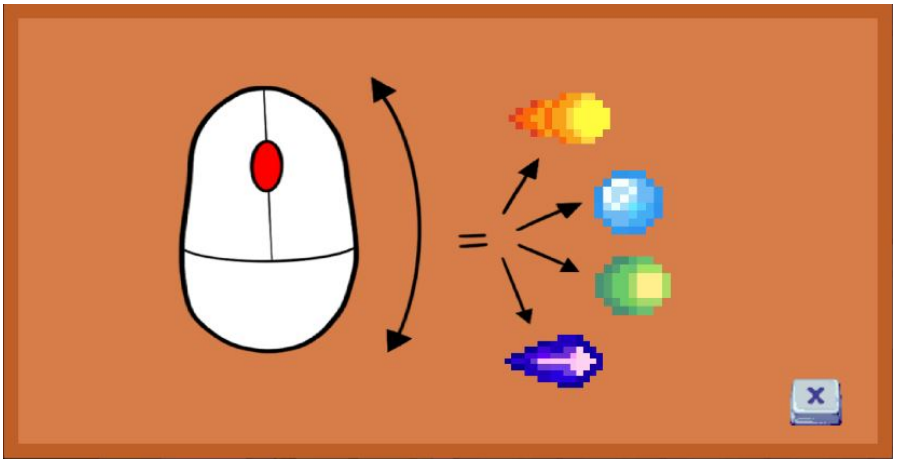

Figure 3.6: Instructions panel level 3 Room 2.

When interacting with certain elements, an animation of the E key appears to make the player understand that they can perform certain actions (Figure 3.7).

Figure 3.7: E key interaction.

Additionally, in each level, you have a specific amount of time to complete the puzzle, creating a defeat condition if the time runs out. This countdown is shown at the top right (Figure 3.8).

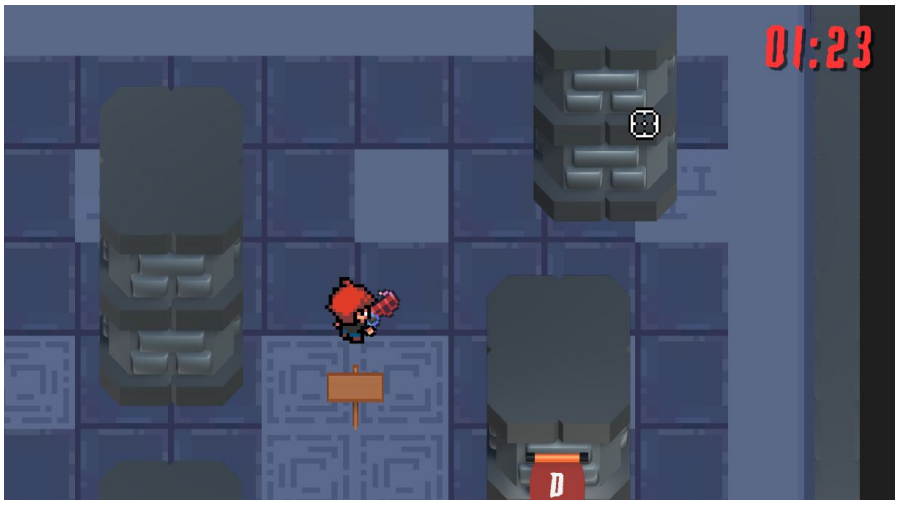

Figure 3.8: Countdown.

In the second level, the player must shoot the colored buttons in a specific order to obtain a code. The status of their solution is continuously updated in the top left corner (Figure 3.9).

Figure 3.9: Code level 2.

In the third level, there are torches with associated letters. Each letter is linked to a specific color. The player must shoot the torches and light them with the correct color. The color of the selected bullet is displayed in the top left corner (Figure 3.10).

Figure 3.10: Bullet color.

Finally, since the player needs to use the shooting mechanic in all the puzzles, a crosshair is displayed at the point where they are aiming.

Chapter 4

Work Development and Results

Contents

Once all the information related to the project has been exposed, it is time to explain how the development process of the video game has been step by step and its evolution until the final result of which I will evaluate its result.

4.1 Work Development

The first step was to study how a 2.5D view could be implemented and adapt it to this project (Figure 4.1). After a while I decided that the camera had to be rotated 45 degrees towards the player. Since the character is a sprite rather than a 3D model, a script was used to modify the camera matrix to ensure everything was displayed correctly. Subsequently, the Cinemachine tool [7] was utilized to easily implement a system in which the camera follows the character. This initial camera was orthographic (Figure 4.2).

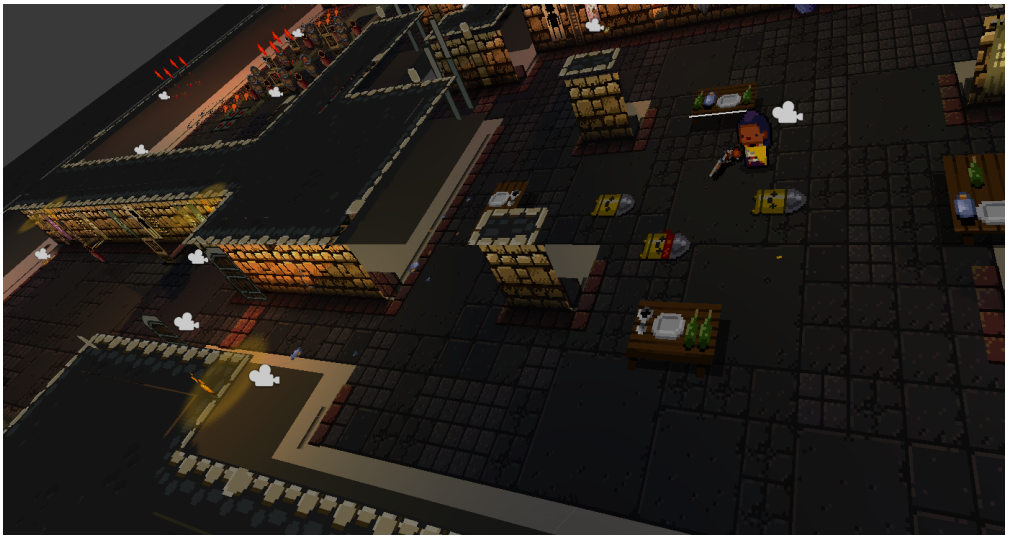

Figure 4.1: Enter the Gungeon “Behind scenes”.

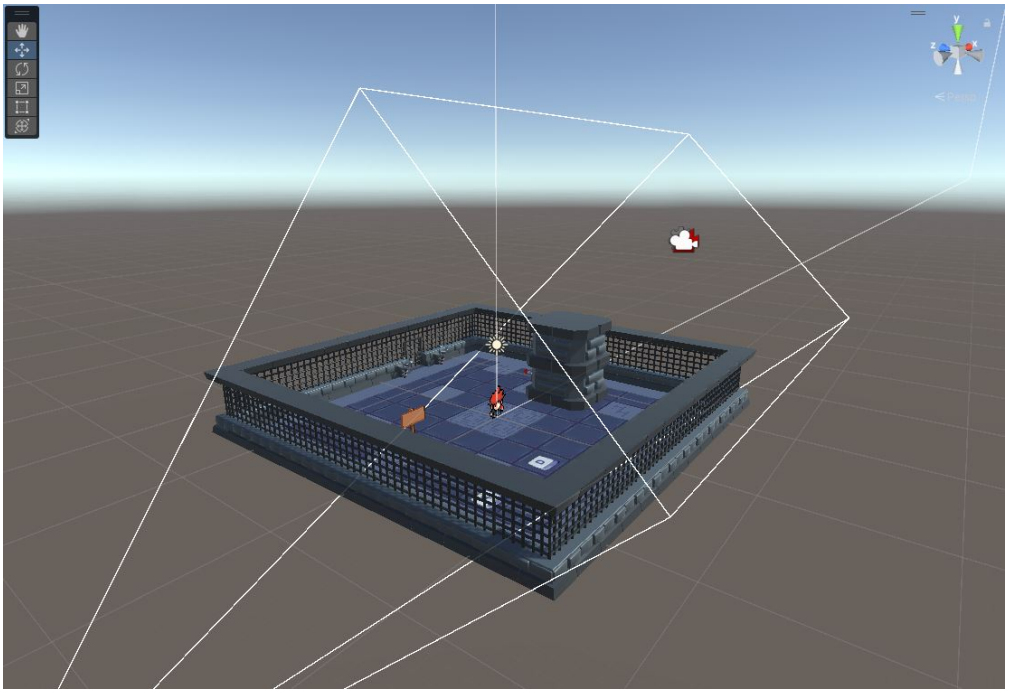

Figure 4.2: Camera and player rotation.

Once I had a basic camera setup, I began programming the player's movement, evaluating different possibilities. Given the nature of the video game, I decided to use the MovePosition function of the Rigidbody in a simple way since I wanted to provide the player with a fluid movement that would immediately respond to his actions. Subsequently, I added a dash ability and integrated all the player animations using blend trees. These animations were later modified to adapt seamlessly to the dimension change mechanic and the shooting system (Figure 4.3).

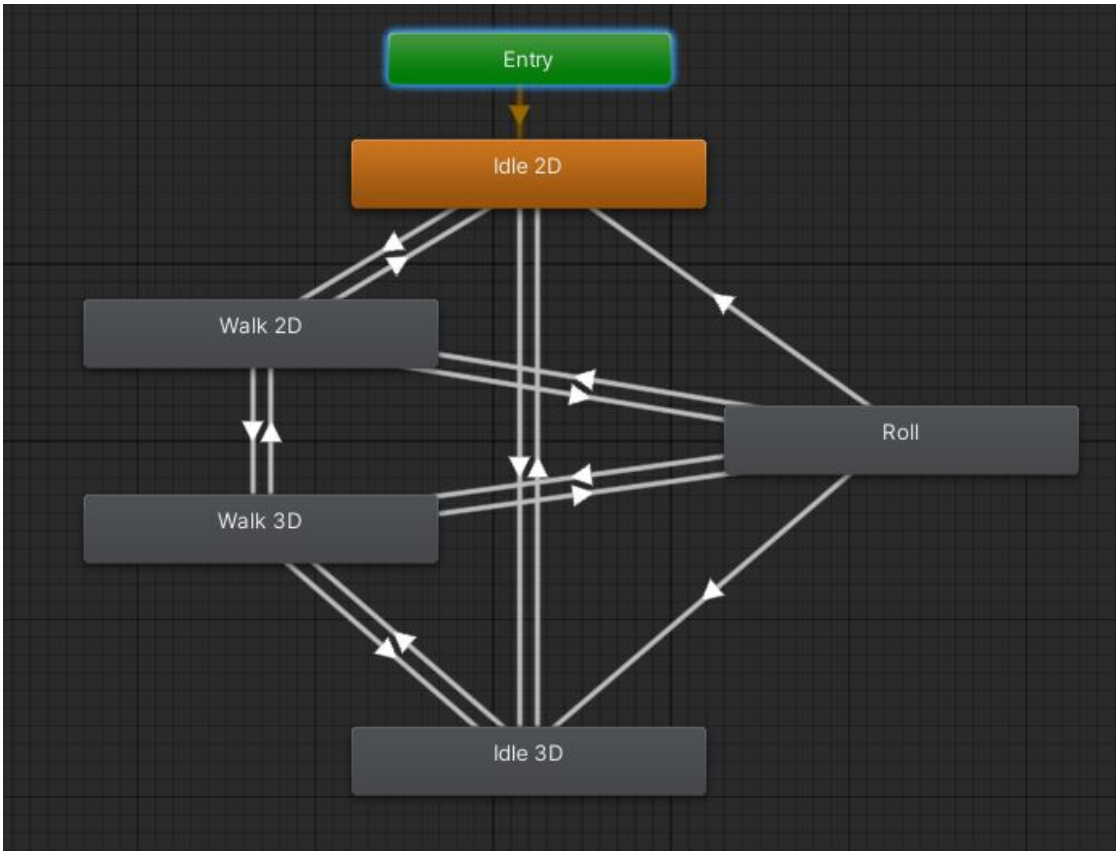

Figure 4.3: Player Animator.

When a decent movement system was achieved, I began implementing one of the project's key mechanics: the dimension change. This was an arduous process since it is not a very common mechanic and, therefore, there was not much information available on Unity forums or, in general, on the Internet. Despite this, with patience and help from various sources, progress was made, and each iteration became more polished.

Initially, I added a second camera to the scene, this one being a perspective camera. Thanks to Cinemachine, transitioning between cameras is straightforward. However,

when trying to transition between the initial orthographic camera and the perspective camera, there was a noticeable cut, creating a notable shift. In Unity, unlike other engines, it is not possible to transition seamlessly between cameras of different types.

To address this, I added a third perspective camera and set its Vertical FOV parameter to a very low value and its Camera Distance parameter to a very high value, simulating an orthographic camera effect despite being a perspective camera. With these three cameras, I programmed a transition that first moved from the orthographic camera to the perspective camera with the orthographic effect, and then from this to the 3D perspective camera. Returning from the 3D view to the 2D view followed the same path in reverse. Finally, I added one last camera to make the final transition smoother (Figures 4.4 and 4.5).

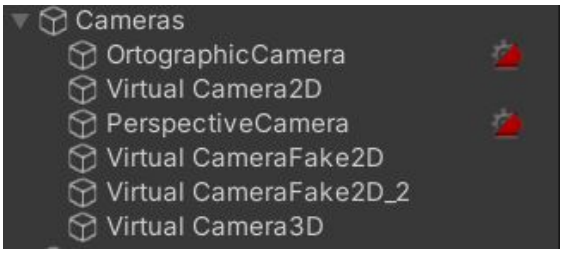

Figure 4.4: List of cameras.

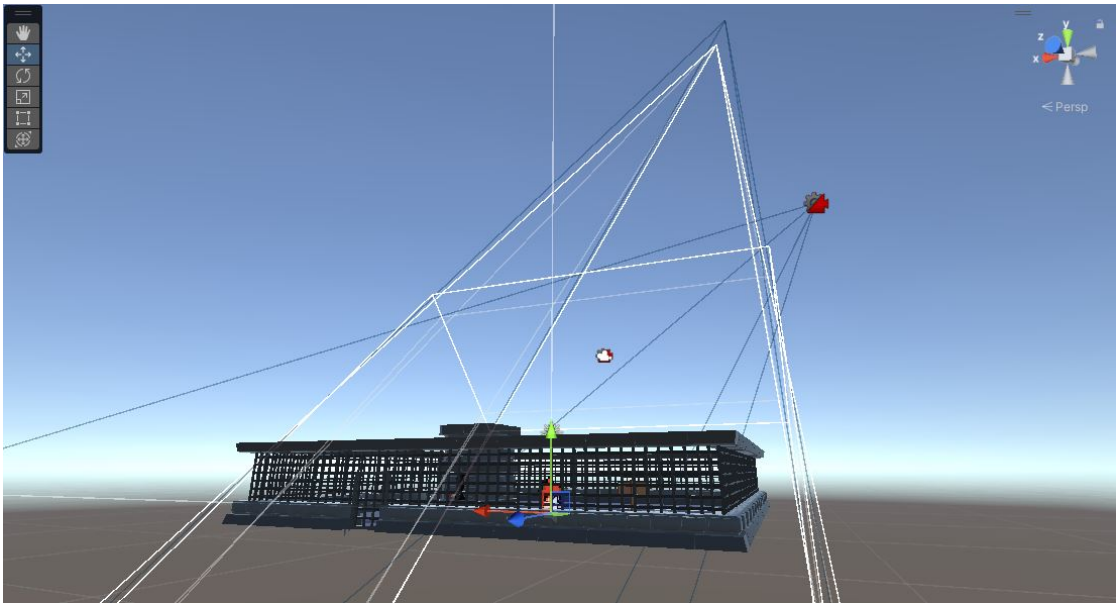

Figure 4.5: Cameras on Scene.

Depending on the dimension in which the player was located, I wanted the movement to differ. In the perspective view, I preferred the player not to have a weapon. For this reason, I programmed the distinction in movement mechanics based on the dimension. This ensured that the gameplay experience was tailored to the specific dimension, enhancing the overall immersion and challenge (Listing 4.1).

```
public void movement()
{
    if (!dashing)
    {
        horizontal = Input.GetAxisRaw("Horizontal"); //GetAxisRaw elimina la
            progresión de movimiento (antes utilizaba GetAxis)
        vertical = Input.GetAxisRaw("Vertical");
    }

    if (dimension == 2) //cambian los controles dependiendo de la dimension
    {
        movementDirection = new Vector3(horizontal, 0, vertical).normalized;
    }
    else
    {
         movementDirection = new Vector3(vertical, 0, -horizontal).normalized;
    }

    //Mueve al jugador
    Vector3 newPosition = rb.position + movementDirection * speed *
        Time.deltaTime;
    rb.MovePosition(newPosition);
}
```

Listing 4.1: Script Player Movement

Next, I implemented a shooting system. The first step was to program a crosshair that indicated where the player was pointing with the mouse. To enhance the camera movement, I made it so that depending on where the player aimed, the camera would move smoothly in that direction, maintaining a position at a medium distance between the mouse pointer and the player. This provided a more pleasant and dynamic visual experience.

Once the crosshair was finished, I programmed the rotation of the weapon using vector calculations. This allowed the gun to rotate directly towards the point on the screen where the player was aiming. Although this system worked perfectly, I wanted to replicate a feature I had seen in *Enter the Gungeon*. In that game, the weapon changes

hands depending on the area where the player is aiming at. I set out to program a similar system and achieved a fairly successful result (Figures 4.6 and 4.7).

Figure 4.6: Gun on right hand.

Figure 4.7: Gun on left hand.

Of course, the next step was to implement the shooting mechanics. This included the shot itself, its animation, the bullet animation, the bullet collision animation, a reloading system every eight shots and its corresponding animation. I have to admit that adding recoil animations when shooting was more problematic than I initially expected due to the gun's rotation (Figures 4.8 to 4.10).

Figure 4.8: Gun animation.

Figure 4.9: Reload animation.

Figure 4.10: Bullet collision animation.

At this point, I had implemented the dimension change mechanic, the shooting system and the player movement. The next step was to design and implement visual puzzles that took full advantage of these features.

First level: When designing the first level, I wanted the player to start without the gun. A sign explains to the player that they can change dimensions (Figure 3.3). Upon doing so, they can see a lever hidden behind a pillar. The player can operate the lever by pressing the E key, which causes the pillar to rotate, revealing a part of the pillar where the weapon is located. The weapon can then be picked up with the E key. Then, to introduce a mechanic of one of the next puzzles, the player must shoot some torches to light them, thereby activating a mechanism that opens the doors of the room (Figures 4.11 and 4.12).

Figure 4.11: Level 1 Initial State.

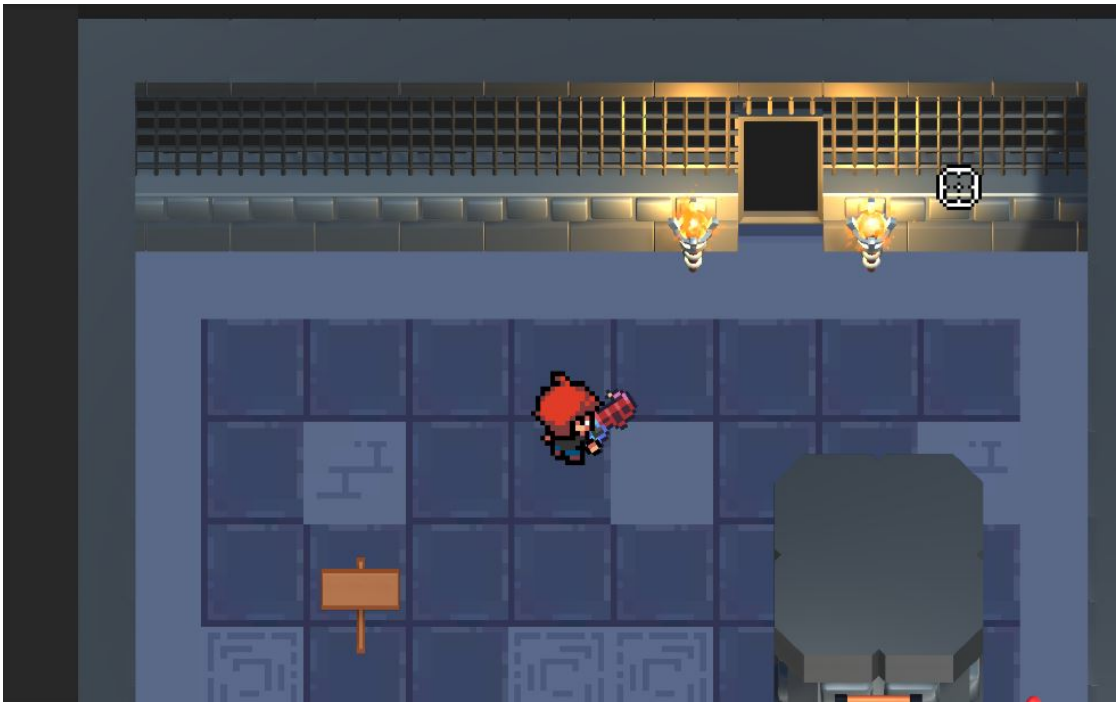

Figure 4.12: Level 1 Final State.

Second level: I had the idea of taking advantage of the change dimension system by putting some clues about a code visible only in the second dimension and others only in the third. In this level there is also a lever that rotates a pillar, thus revealing a new clue and a new button. Once the player had all the clues, which relate a number to a color, he should go to some colored buttons and shoot them in the correct order to form the key that the game asks for at the sign of the level (Figure 3.4, 4.13 and 4.14).

Figure 4.13: Level 2 Second dimension.

Figure 4.14: Level 2 Third dimension.

Third level: In this level, a new game mechanic has been introduced: the ability to switch between four different projectile types, each of a different color. The level consists of two main rooms: the first room features flags displaying letters associated with specific colors, which players need to memorize in thirty seconds (Figure 4.15). In the second room, there are several torches, each one with a letter in a flag above. Players must shoot each torch with a projectile matching the color associated with the corresponding letter (Figure 4.16). The player can change the color of the bullet with the mouse wheel and, so that he know in every moment which type of bullet he has selected, I added an image on the canvas that indicates the color of the current bullet. (Figure 3.10).

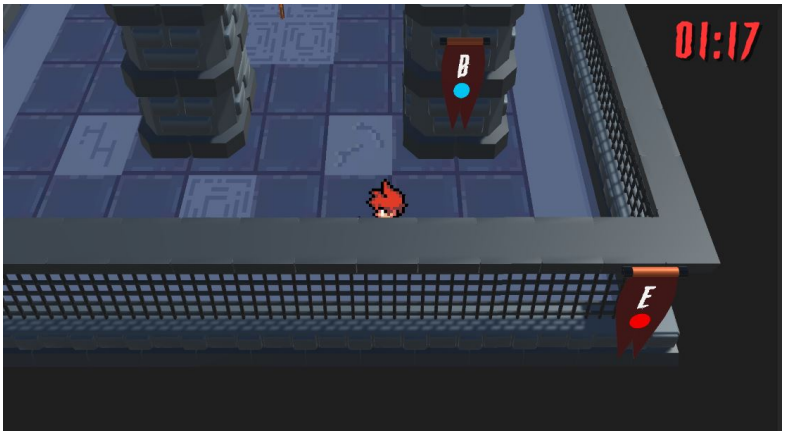

Figure 4.15: Level 3 First room.

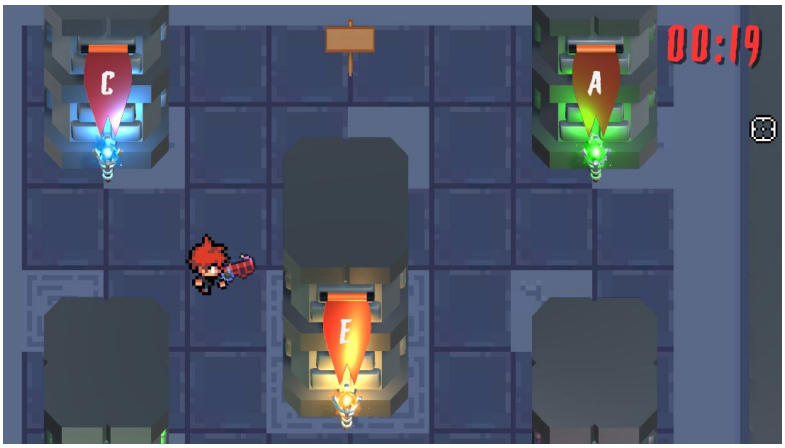

Figure 4.16: Level 3 Second room.

After implementing the main mechanics and puzzles for each level, I focused on developing the narrative aspect of the game. A key objective of this project is to immerse the player deeply, ensuring their actions feel meaningful with significant consequences. To achieve this, I crafted a storyline centered around a kidnapped person whose life is in the player's hands. To save his life the player must complete all the puzzles, otherwise he will be killed.

The way to tell the story is through edited videos in which appears a doll inspired by the Saw movies (Figure 4.17). In these videos, the doll explains to the player what his situation is and he guides him throughout the game. Depending on whether the player successfully solves all the puzzles, one of two possible endings will unfold.

Figure 4.17: Video doll.

To enhance realism and immersion, with the help of my sister, we recorded videos of myself acting like the kidnapped person (Figure 4.18).

Figure 4.18: Video kidnapped person.

Finally, I created a simple menu (Figure 4.19), integrated all puzzles with the videos, added appropriate sound effects and music, and implemented a countdown mechanism to introduce a sense of urgency and a potential defeat condition in each level.

Figure 4.19: Menu.

4.2 Results

Looking back at the objectives set out in Section 1.2, I am pleased to say that I have successfully completed all of them. I am proud to have implemented a unique and rarely seen mechanic like the dimension change, along with the intriguing puzzles it introduces. Additionally, I am very satisfied with the final state of the player's movement and the shooting system, as they feel exceptionally polished.

Furthermore, I believe the chosen narrative makes the game an immersive experience where players feel their actions are significant, thus fully meeting that objective. Finally, I have achieved a completely playable version from start to finish, which I can share with friends and family. Overall, I am very happy and proud of the results obtained.

Chapter 5

Conclusions and Future Work

Contents

5.1 Conclusions

At the driving school you learn the basics of driving but it is not until you get your license and start driving by your own that you really learn to drive. Degrees remind me a lot of driving schools precisely for this reason. In our case, we learn a lot about different sections of video game design and programming, but it is not until we sit down to develop our own ideas that we learn the reality of this world, acquire new skills and evolve as video game creators. Thus, this final degree project has been a great opportunity to face new problems, overcome difficulties and learn along the way.

During our degree we have had all kinds of subjects but, without a doubt, the ones that seemed most interesting to me were those in which we developed a game. The moment you must face the challenge of implementing a game in all its aspects is when you really learn how video game design and development works, being a very nutritious experience. Likewise, I found this project to be a very fruitful process during which I have learned to implement new mechanics and where I have shown myself what I am capable of, using the foundations of the degree to expand my knowledge and overcome new goals.

5.2 Future work

Due to my personal situation and lack of time, I would like to continue developing the project in order to polish certain aspects of it and add new mechanics and puzzle ideas that have crossed my mind but that I have not had time to implement. In any case, I am proud of the project in general, from the idea to the execution and the result obtained.

Bibliography

[1] ByteDance. Capcut. urlhttps://www.capcut.com/es-es/. Accessed: 2024-06-26.

[2] Leaf Corcoran. Itch.io. https://itch.io/. Accessed: 2024-06-26.

[3] Visual Paradigm International. Visual paradigm. https://online.visual-paradigm.com/es/. Accessed: 2024-06-26.

[4] Daniel Kvarfordt. Pyxel edit. https://pyxeledit-com.translate.goog/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=sc. Accessed: 2024-06-26.

[5] Microsoft. Github desktop. https://github.com/. Accessed: 2024-06-26.

[6] Microsoft. Visual studio. https://visualstudio.microsoft.com/es/vs/older-downloads/w. Accessed: 2024-06-26.

[7] Unity Technologies. Cinemachine tool. https://docs.unity3d.com/Packages/com.unity.cinemachine@3.1/manual/index.html. Accessed: 2024-06-26.

[8] Unity Technologies. Unity. https://unity.com/es. Accessed: 2024-06-26.

[9] Unity Technologies. Unity asset store. https://assetstore.unity.com/. Accessed: 2024-06-26.

[10] Voicemaker Technologies. Voice maker. https://voicemaker.in/. Accessed: 2024-06-26.

[11] Alexandre Thomas and Dmitry Barashev. Ganttproject. https://www.ganttproject.biz/. Accessed: 2024-06-26.

[12] Wikipedia. Enter the gungeon. https://es.wikipedia.org/wiki/Enter_the_Gungeon. Accessed: 2024-06-26.

[13] Wikipedia. Super paper mario. https://es.wikipedia.org/wiki/Super_Paper_Mario. Accessed: 2024-06-26.

[14] Wikipedia. Wii. https://es.wikipedia.org/wiki/Wii. Accessed: 2024-06-26.

APPENDIX A

Other considerations

A.1 Source Code

Since this is a project dedicated to programming and the number of scripts is large, I consider it a better option to offer the link to the github repository of the game.

Github repository: https://github.com/sergiogomez30/Toy-Nightmare