Development of an Escape Room Video Game about Recycling

Nàdia Montaña Miranda

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

May 22, 2023

Supervised by: Inmaculada Remolar Quintana.

To all the friends who introduced me to video games, and to my family for supporting me in the decision to dedicate my professional career to them.

Acknowledgments

First of all, I would like to thank my Final Degree Work supervisor, Inmaculada Remolar, and all my coworkers from the INIT (The Institute of New Imaging Technologies) for their support through this process, which helped me to reach this final state of the project.

I also would like to thank Reciclatge Forés S.L. and all its team, specially Ismael Forés and Neus Forés, for introducing me to the world of recycling and for teaching me so much about it.

Abstract

This document represents the Final Degree Work report of Nàdia Montaña Miranda, in Bachelor's Degree in Video Game Design and Development.

The main concept is a video game with an escape room format, with the aim of educating young people about recycling. This serious game is expected to be used in educational institutions for students to enjoy while learning through video games.

In order to complete the three levels of the escape room, the user will learn important information about recycling, such as the correct separation of waste, the decontamination process of a vehicle or the plastic recycling process.

Keywords: escape room, first-person, 3D, recycling, serious game

Contents

CHAPTER 1

INTRODUCTION

Contents

This chapter is the explanation of what has been done through the development of the work, as well as the idea that motivated the decision of doing this game.

1.1 Work Motivation

The current environmental problem is one of the biggest challenges the world has ever confronted. According to the European Environment Agency, every European citizen produces 4.8 tones of waste each year, and only the 49% of municipal waste is recycled [1]. Everyone should take measures, such as recycling, to minimize the negative impact of human activity on the environment and preserve the planet for future generations.

I started caring more about recycling thanks to the close relationship I have with the recycling company in my city, Reciclatge Forés S.L. [46], who inspired me to focus my project to teach new generations as they did with me.

Due to all of this, it seemed like an interesting idea to create a video game as a medium to educate the youth about recycling. I also consider raising awareness about the environment a small contribution to help fulfill the Sustainable Development Goals (SDG) [26], since knowledge is a very powerful tool.

Moreover, ever since I found out what an escape room was, I loved them. I started playing mobile video games about escape rooms many years ago, most of them from Rusty Lake [63], like "Cube Escape: Seasons" [22] or "Cube Escape: Paradox" [21]. Besides the mobile escape rooms, I have also gone to do some escape rooms in person. I really enjoyed them all, but I have never seen any of them trying to be something more than just entertainment. So, with the desire to create a video game dedicated to the environment, the love I have for escape rooms and my goal to dedicate myself professionally to 3D modeling, the idea for this video game came up.

1.2 Objectives

There are some goals to be accomplished by this project:

- Develop a tool that could be used by educational institutions to educate about recycling
- Help young people to become aware of the environment
- Offer different levels related to different aspects about recycling
- Engage the player due interesting challenges

In order to achieve these objectives, there are other objectives to be accomplished during the development of the game:

- Develop simple but attractive mechanics so that all users can play
- Create 3D models with Blender that can be visually appealing to young people
- Use Unity 3D to implement the levels of the escape room using the 3D models made
- Animate some of the objects
- Do enough research to provide real information through the game
- Take into account the efficiency of the game, so that any computer can run the game
- Give visual and sound information in order to make all the user actions clear

The expected result is to end up with a good looking, innovative and educational serious game conscious of the environment. I also expect it to be able to be used in schools, high schools or universities for students to enjoy while learning through video games, maybe on the World Environment Day, June 5th.

1.3 Environment and Initial State

Once the theme of the video game was clear and it was decided to give it an Escape Room form, the research process began. This process started by looking for other similar projects, which is detailed in the 5.1.1 section, and by learning about how the recycling processes are in a recycling plant, which is detailed in the 5.1.2 section.

After the research stage, the approach process began. At this stage, the way in which the game was going to be shown to the user became relevant. The game design was proposed with a simple mechanics of interaction with different types of objects, and with a cartoon art style to appeal to the young target audience. The ideas of what was going to be done in each level were conceived during the research at the recycling plant, thanks to actually see what was done there and what was most important for the environment.

Lastly, with all the new information acquired by the previous stages, the development part began, which is explained in the 5.2 section. This stage is the one to which more hours have been dedicated, due to all the 3D modeling, texturing, designing, animating and coding.

Chapter 2

Planning and resources evaluation

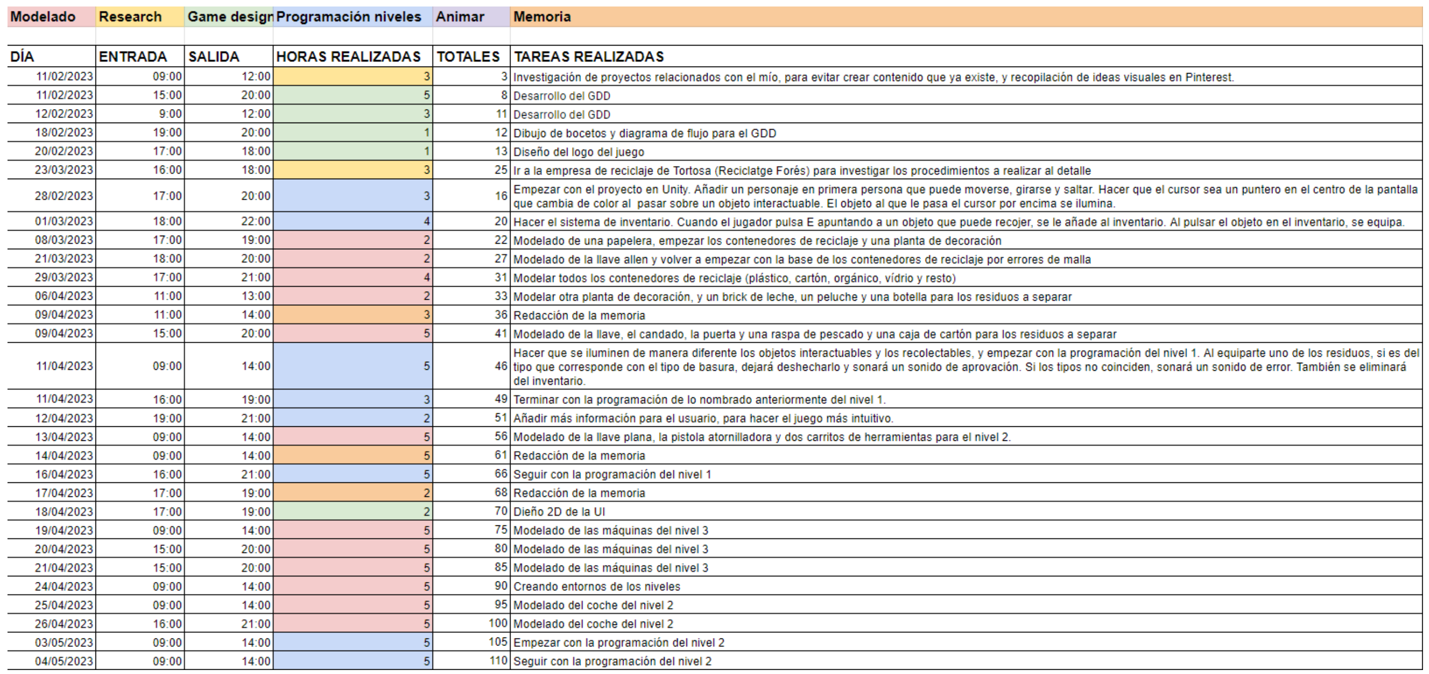
Contents

This chapter will explain the time planning followed and the resources used to develop this project.

2.1 Planning

At the beginning of the project, an initial time planning was made (see Table 2.1). Most of this planning has been followed, but during the development of the video game some parts of the project required more dedication while others required less (see more details about the differences between the planning tables in the 5.2 section). This is how the time distribution ended up being (see Table 2.2).

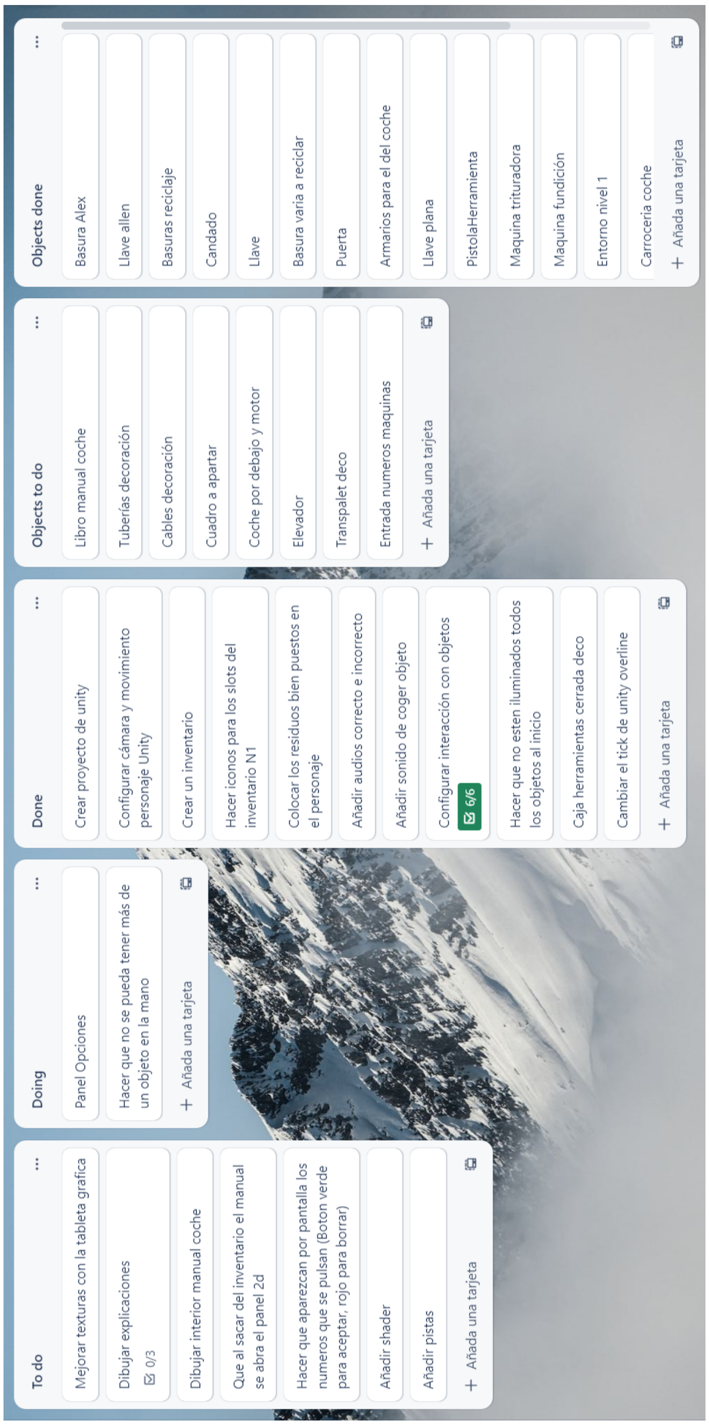
The advances that have been made throughout the process have been noted in a spreadsheet, classifying them among 3D modeling, research, game design, coding, animating and writing this report (see Figure 2.1). The research has been counted as game design hours. Besides, in order to divide all the work to do in smaller tasks, the online tool Trello has been used. I learned about this tool while studying Software Engineering subject (VJ1224), and it has been very useful (see Figure 2.2). During this subject I also learned how Gantt Charts worked, and it has also helped a lot during the planning (see Figure 2.3).



Modelado	Research	Game design	Programación niveles	Animar	Memoria

DÍA	ENTRADA	SALIDA	HORAS REALIZADAS	TOTALES	TAREAS REALIZADAS
11/02/2023	09:00	12:00	3	3	Investigación de proyectos relacionados con el mío, para evitar crear contenido que ya existe, y recopilación de ideas visuales en Pinterest.
11/02/2023	15:00	20:00	5	8	Desarrollo del GDD
12/02/2023	9:00	12:00	3	11	Desarrollo del GDD
18/02/2023	19:00	20:00	1	12	Dibujo de bocetos y diagrama de flujo para el GDD
20/02/2023	17:00	18:00	1	13	Diseño del logo del juego
23/03/2023	16:00	18:00	3	25	Ir a la empresa de reciclaje de Tortosa (Reciclatge Forés) para investigar los procedimientos a realizar al detalle
28/02/2023	17:00	20:00	3	16	Empezar con el proyecto en Unity. Añadir un personaje en primera persona que puede moverse, girarse y saltar. Hacer que el cursor sea un puntero en el centro de la pantalla que cambia de color al pasar sobre un objeto interactuable. El objeto al que le pasa el cursor por encima se ilumina.
01/03/2023	18:00	22:00	4	20	Hacer el sistema de inventario. Cuando el jugador pulsa E apuntando a un objeto que puede recojer, se le añade al inventario. Al pulsar el objeto en el inventario, se equipa.
08/03/2023	17:00	19:00	2	22	Modelado de una papelera, empezar los contenedores de reciclaje y una planta de decoración
21/03/2023	18:00	20:00	2	27	Modelado de la llave allen y volver a empezar con la base de los contenedores de reciclaje por errores de malla
29/03/2023	17:00	21:00	4	31	Modelar todos los contenedores de reciclaje (plástico, cartón, orgánico, vídrio y resto)
06/04/2023	11:00	13:00	2	33	Modelar otra planta de decoración, y un brick de leche, un peluche y una botella para los residuos a separar
09/04/2023	11:00	14:00	3	36	Redacción de la memoria
09/04/2023	15:00	20:00	5	41	Modelado de la llave, el candado, la puerta y una raspa de pescado y una caja de cartón para los residuos a separar
11/04/2023	09:00	14:00	5	46	Hacer que se iluminen de manera diferente los objetos interactuables y los recolectables, y empezar con la programación del nivel 1. Al equiparte uno de los residuos, si es del tipo que corresponde con el tipo de basura, dejará deshecharlo y sonará un sonido de aprovación. Si los tipos no coinciden, sonará un sonido de error. También se eliminará del inventario.
11/04/2023	16:00	19:00	3	49	Terminar con la programación de lo nombrado anteriormente del nivel 1.
12/04/2023	19:00	21:00	2	51	Añadir más información para el usuario, para hacer el juego más intuitivo.
13/04/2023	09:00	14:00	5	56	Modelado de la llave plana, la pistola atornilladora y dos carritos de herramientas para el nivel 2.
14/04/2023	09:00	14:00	5	61	Redacción de la memoria
16/04/2023	16:00	21:00	5	66	Seguir con la programación del nivel 1
17/04/2023	17:00	19:00	2	68	Redacción de la memoria
18/04/2023	17:00	19:00	2	70	Dieño 2D de la UI
19/04/2023	09:00	14:00	5	75	Modelado de las máquinas del nivel 3
20/04/2023	15:00	20:00	5	80	Modelado de las máquinas del nivel 3
21/04/2023	15:00	20:00	5	85	Modelado de las máquinas del nivel 3
24/04/2023	09:00	14:00	5	90	Creando entornos de los niveles
25/04/2023	09:00	14:00	5	95	Modelado del coche del nivel 2
26/04/2023	16:00	21:00	5	100	Modelado del coche del nivel 2
03/05/2023	09:00	14:00	5	105	Empezar con la programación del nivel 2
04/05/2023	09:00	14:00	5	110	Seguir con la programación del nivel 2

Figure 2.1: Fragment of the spreadsheet with advances throughout the process



To do

- Mejorar texturas con la tableta grafica
- Dibujar explicaciones 0/3
- Dibujar interior manual coche
- Que al sacar del inventario el manual se abra el panel 2d
- Hacer que aparezcan por pantalla los numeros que se pulsan (Boton verde para aceptar, rojo para borrar)
- Añadir shader
- Añadir pistas
- + Añada una tarjeta

Doing

- Panel Opciones
- Hacer que no se pueda tener más de un objeto en la mano
- + Añada una tarjeta

Done

- Crear proyecto de unity
- Configurar cámara y movimiento personaje Unity
- Crear un inventario
- Hacer iconos para los slots del inventario N1
- Colocar los residuos bien puestos en el personaje
- Añadir audios correcto e incorrecto
- Añadir sonido de coger objeto
- Configurar interacción con objetos 6/6
- Hacer que no esten iluminados todos los objetos al inicio
- Caja herramientas cerrada deco
- Cambiar el tick de unity overline
- + Añada una tarjeta

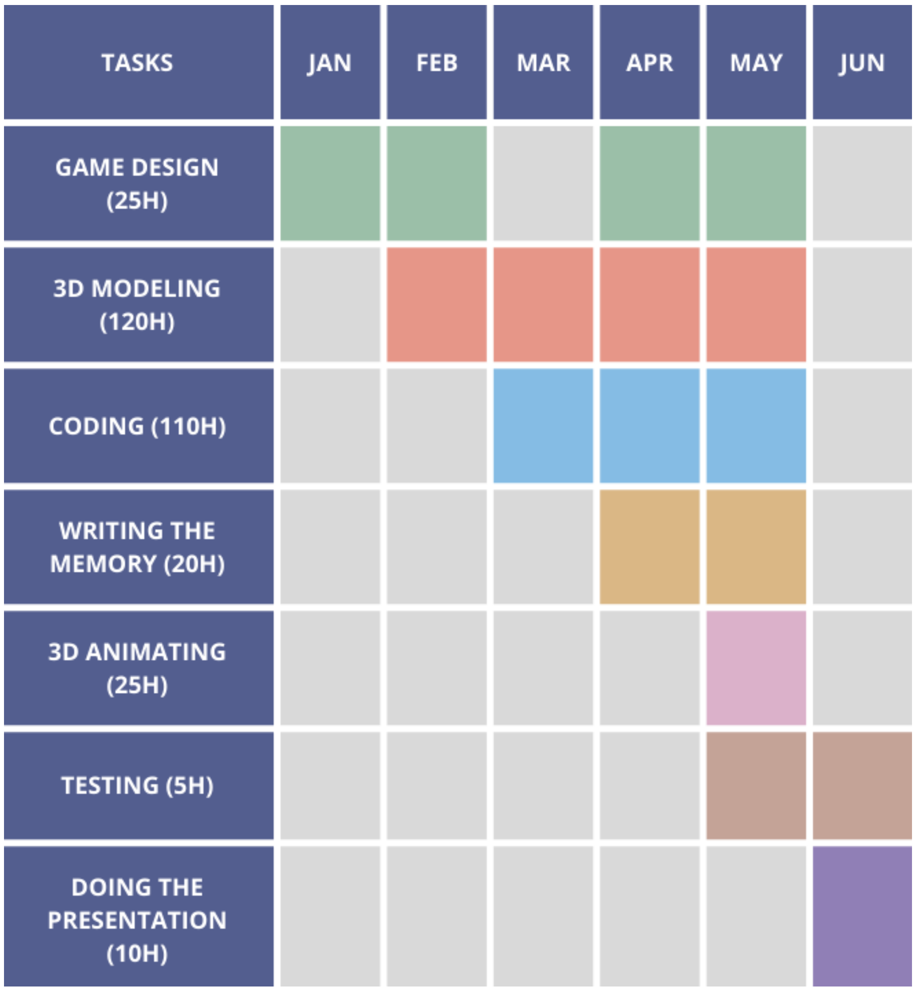
Objects to do

- Libro manual coche
- Tuberías decoración
- Cables decoración
- Cuadro a apartar
- Coche por debajo y motor
- Elevador
- Transpalet deco
- Entrada numeros maquinas
- + Añada una tarjeta

Objects done

- Basura Alex
- Llave allen
- Basuras reciclaje
- Candado
- Llave
- Basura varia a reciclar
- Puerta
- Armarios para el del coche
- Llave plana
- PistolaHerramienta
- Maquina trituradora
- Maquina fundición
- Entorno nivel 1
- Carroceria coche
- + Añada una tarjeta

Figure 2.2: Fragment of the Trello work space



TASKS	JAN	FEB	MAR	APR	MAY	JUN
GAME DESIGN (25H)	🟩	🟩		🟩	🟩	
3D MODELING (120H)		🟥	🟥	🟥	🟥	
CODING (110H)			🟦	🟦	🟦	
WRITING THE MEMORY (20H)				🟧	🟧	
3D ANIMATING (25H)					🟪	
TESTING (5H)					🟫	🟫
DOING THE PRESENTATION (10H)						🟪

Figure 2.3: Gantt chart of the work done

Activity	Expected time
Game design	10h
Modeling	100h
Creating textures	30h
Coding	100h
3D Animating	30h
Producing the sounds	10h
Writing the memory	20h
Total	300h

Table 2.1: Initial planning table

Activity	Time required
Game design	25h
Modeling	120h
Coding	110h
3D Animating	25h
Writing the memory	20h
Total	300h

Table 2.2: Final planning table

2.2 Resource Evaluation

There were needed some hardware and software resources in order to develop this video game. All the software resources were free to use, and the hardware ones were already acquired long before the start of the project. Here is the list of all of them:

- Software:

  - Blender: used to model in 3D all the elements of the game.
  - Unity 2021.2.0f1: used to create the game itself.
  - Unity Hub 2.4.5: used to access and manage my Unity file.
  - Visual Studio 2019: used to program in C# the scripts from Unity.
  - Krita: used to draw elements in 2d like the game logo.
  - OverLeaf: used to write through Latex the Final Degree Work Report.
  - Trello: used to organize the project tasks and to keep track of them.
  - ClickMinded.com: used to generate buttons for the UI.
  - Lucid.app: used to do a flow chart of the game.

- Hardware:

  - Laptop OMEN by HP, 16 GB RAM, NVIDIA GeForce GTX 1070, i7: the computer used.
  - TAL-SUKHOI 8D Gaming Mouse: the mouse used.

Chapter 3

System Analysis and Design

Contents

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

3.1 Requirement Analysis

The interactable elements light up when the user hovers over them. The ones that can be added to the inventory are iluminated in blue, and the ones that can not are iluminated in yellow.

3.1.1 Functional Requirements

R.1. The user can interact with yellow illuminated objects.
R.2. The user can add to its inventory the blue illuminated objects.
R.3. The user can open its inventory.
R.4. The user can equip the objects from its inventory.
R.5. The user can move around the rooms.
R.6. The user can open the settings menu.
R.7. The user can open the controls explanation screen.
R.8. The user can change the music volume.

R.9. The user can control the camera
R.10. The user can return to the start screen.
R.11. The user can exit the game.

3.1.2 Non-functional Requirements

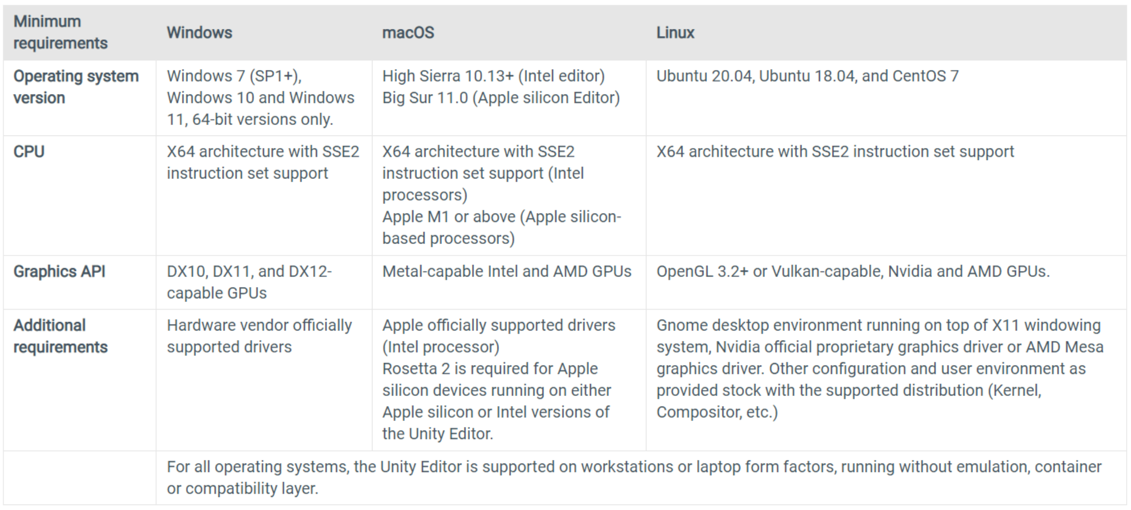
R.12. The game will be in 3D.
R.13. The mechanics will be easy to understand.
R.14. The controls will be simple.
R.15. The UI will be minimalist to avoid overwhelming the player.
R.16. The game will be only available for PC.
R.17. The game will follow a cartoon and creepy aesthetic.
R.18. The sound effects will be used for giving information.

3.2 System Design

This flow chart represents the sequence of actions, decisions, and outcomes that the player can experience while playing the game (see Figure 3.2).

3.3 System Architecture

This video game has been made with the 2021.2.0f1 version of Unity 3D engine, for PC. According to the Unity documentation, these are the minimum system requirements to run the Unity Editor (see Figure 3.1) [6].



Minimum requirements	Windows	macOS	Linux
Operating system version	Windows 7 (SP1+), Windows 10 and Windows 11, 64-bit versions only.	High Sierra 10.13+ (Intel editor) Big Sur 11.0 (Apple silicon Editor)	Ubuntu 20.04, Ubuntu 18.04, and CentOS 7
CPU	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support (Intel processors) Apple M1 or above (Apple silicon-based processors)	X64 architecture with SSE2 instruction set support
Graphics API	DX10, DX11, and DX12-capable GPUs	Metal-capable Intel and AMD GPUs	OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs.
Additional requirements	Hardware vendor officially supported drivers	Apple officially supported drivers (Intel processor) Rosetta 2 is required for Apple silicon devices running on either Apple silicon or Intel versions of the Unity Editor.	Gnome desktop environment running on top of X11 windowing system, Nvidia official proprietary graphics driver or AMD Mesa graphics driver. Other configuration and user environment as provided stock with the supported distribution (Kernel, Compositor, etc.)
	For all operating systems, the Unity Editor is supported on workstations or laptop form factors, running without emulation, container or compatibility layer.		

Figure 3.1: Unity requirements

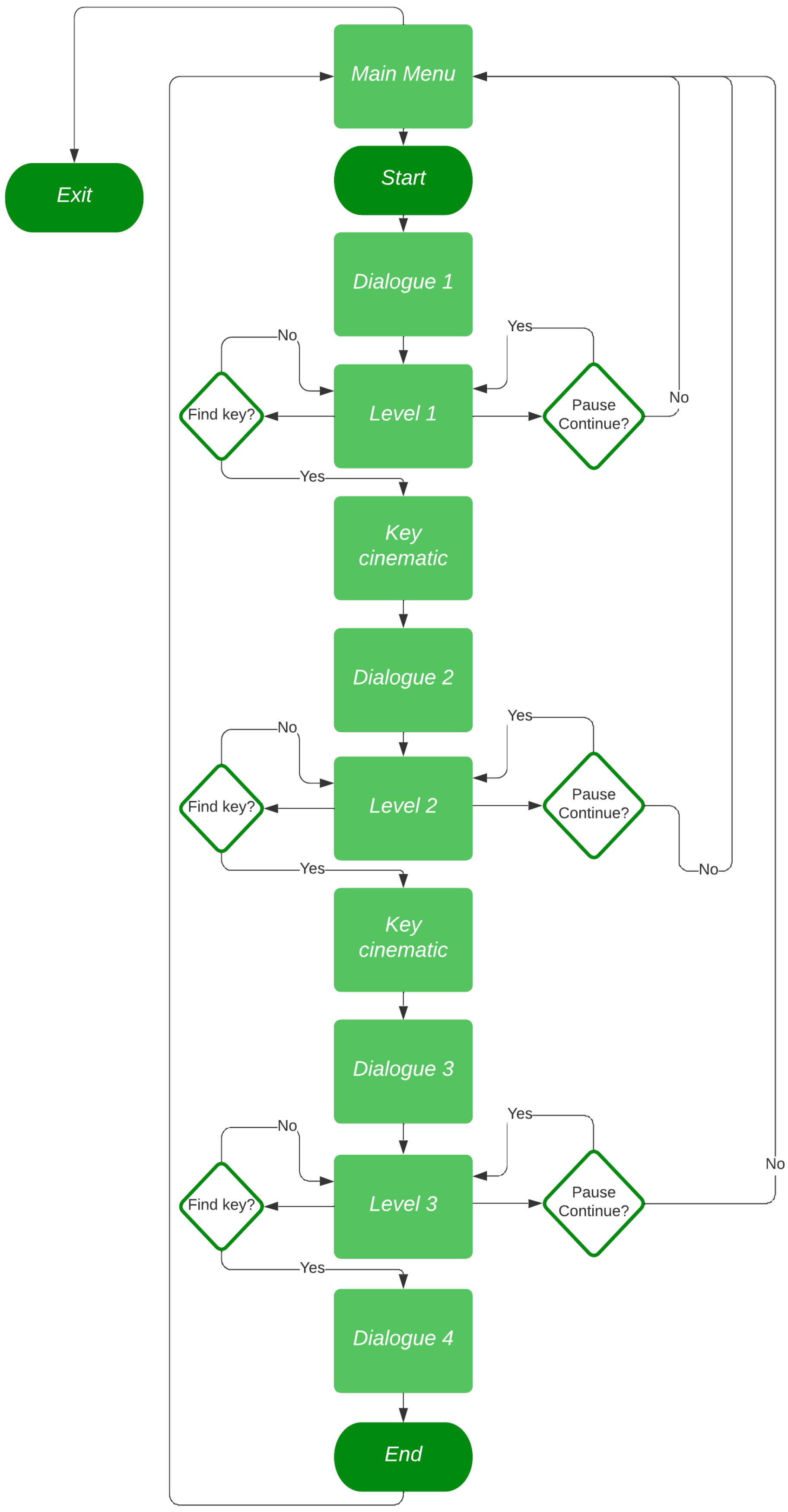


Figure 3.2: Game Flow Chart

3.4 Interface Design

Some of the sprites used in the interface screens were downloaded from the internet (see the A.1 section), except for the clues for the second level (see Figure 3.7), the clues for the third level (see Figure 3.8), and the buttons "empezar", "controles", "salir" and "audio" that were created through the website Click Minded [24]. The icons of the items used in the inventory are captures of their 3D models from Blender. Several decisions have been made about the UI design as well:

- Providing audio feedback to indicate whether the user performs an action correctly or incorrectly, in order to provide additional information.
- Making the cursor invisible to avoid distracting the user, except when necessary (in the inventory, options panel, dialogues, and main menu).
- Changing the pointer and illuminating elements when hovering over an interactive object to provide more additional information (see Figure 5.24).

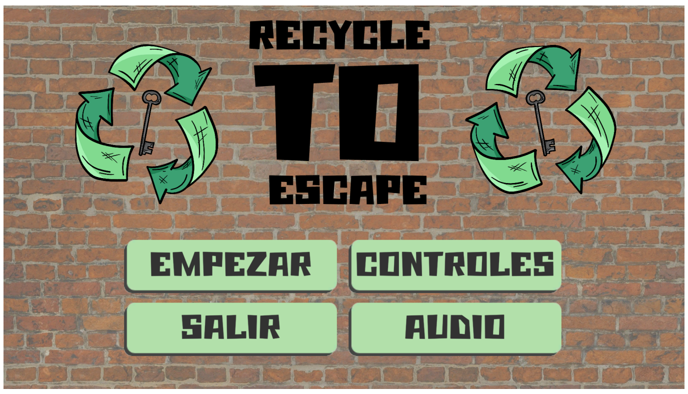


Figure 3.3: Main menu screen

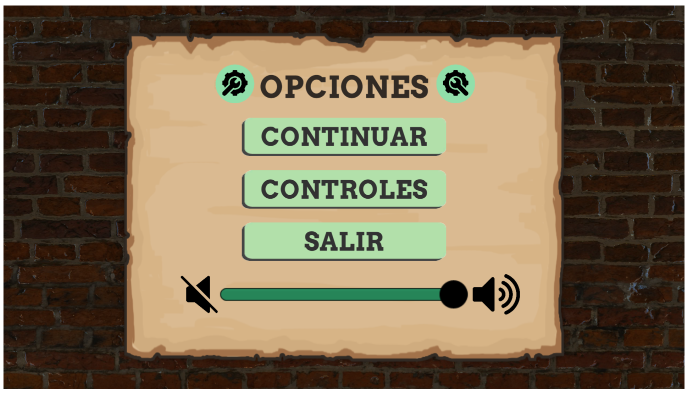


Figure 3.4: Options screen

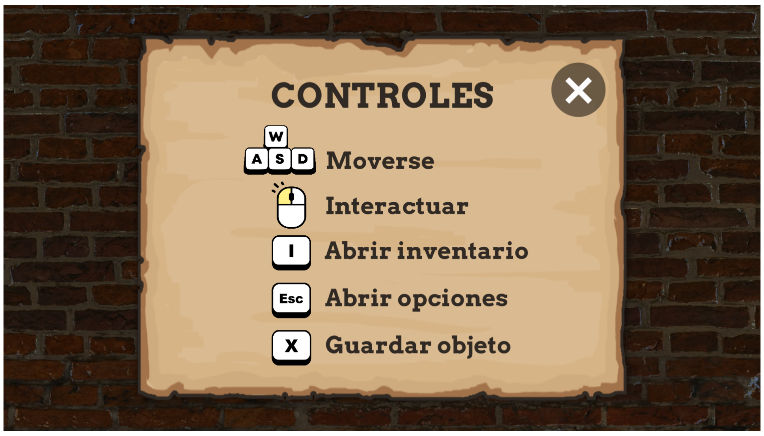


Figure 3.5: Controls screen

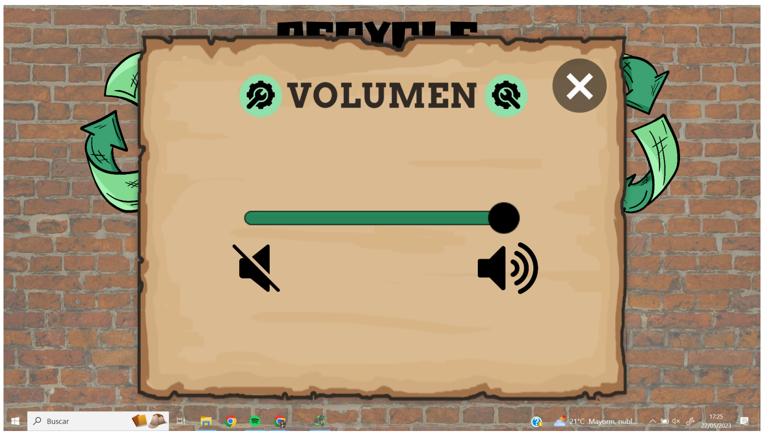


Figure 3.6: Volume settings screen

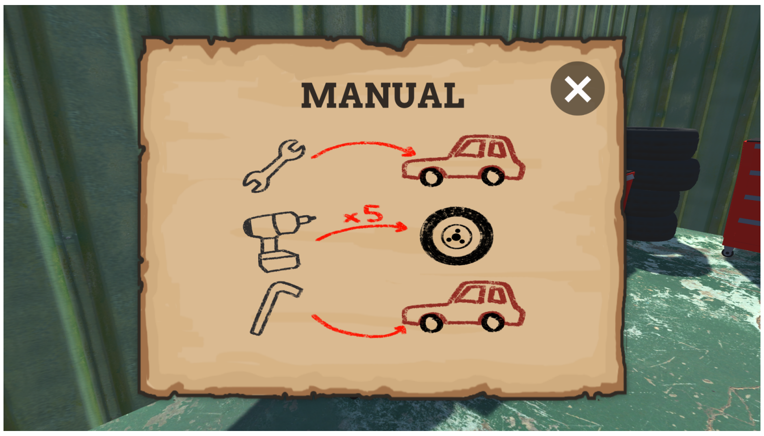


Figure 3.7: Level two clue screen

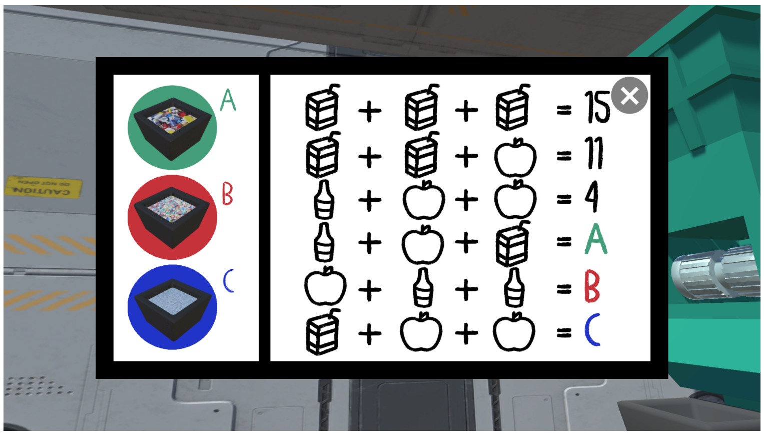


Figure 3.8: Level three clue screen

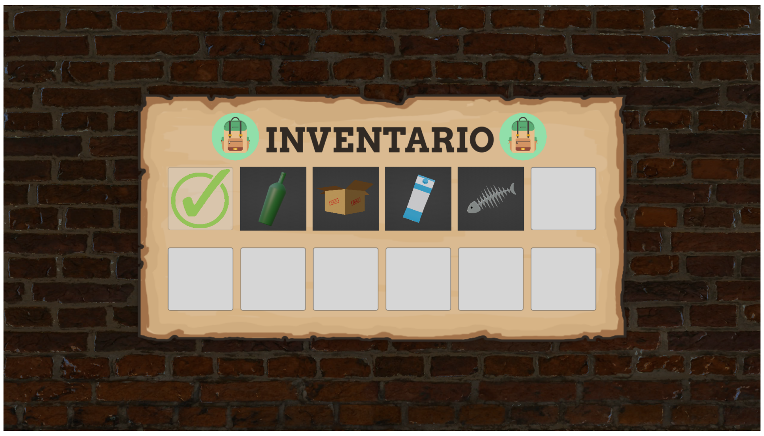
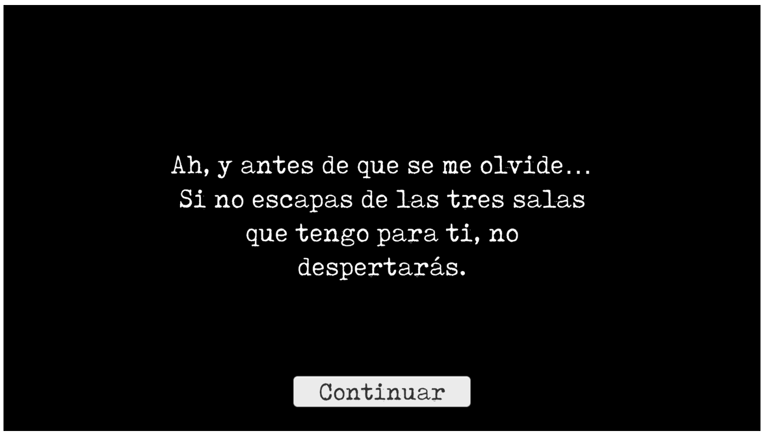


Figure 3.9: Inventory screen



Ah, y antes de que se me olvide...
Si no escapas de las tres salas
que tengo para ti, no
despertarás.

Figure 3.10: Fragment from the first dialogue screen

Chapter 4

Game Design

Contents

In this chapter, the game design will be further described, detailing its summary, its level design, its art and its dialogues.

4.1 Game Summary

4.1.1 Narrative

The player is a teenager who does not give importance to caring for the environment. One night, he/she had a nightmare: he/she was trapped in a room, and a mysterious voice tried to make him/her aware of his/her disrespectful behavior with the environment. To escape from that room and from the other two, the player must find the clues and solve the puzzles, while becoming aware of how to be more careful with the planet. The user will only be able to "wake up" or end the game if he/she escapes from the three rooms.

4.1.2 Target audience

The target audience is in the range of 12 to 30 years. In fact, it does not really have a limit age, because everyone can learn about recycling at any age. However, it will be visually oriented to young people. The puzzles to solve may be too difficult for children of young ages, which is the reason for the chosen age range. About Bartle taxonomy [62], it is expected that some of the user's profiles will be the "achiever" one, since they must be able to solve all the puzzles from all the levels to complete the game. The other user's profile expected is the "explorer" one, since they will have to explore the rooms to discover the clues and the items.

4.1.3 Main characteristics

- Title: Recycle to Escape
- Platform: PC
- Genres: Puzzle, Serious Game, Escape Room (sub-genre)
- Artistic style: All the elements of the environment will be 3D, and will follow a cartoon style, but also a bit creepy due to the nightmare it takes place in.
- Similar games: "The Past Within" (Rusty Lake) [23], "Hello Neighbor" (Dynamic Pixels, tinyBuild) [41] or "Superliminal" (Pillow Castle) [4].

4.2 Gameplay and Level Design

The puzzles of the game are designed to challenge the mental ability of the player. The objective is to escape from the nightmare and, in order to do that, the player must escape from the three rooms of the game. There will be clues for the two last rooms (the most difficult ones) that the user has to find and understand to complete the challenges of the levels. When the user completes each level, he will get a key to open the padlock of each door. During all the levels, the player will be able to open the options screen (see Figure 3.4) and the inventory (see Figure 3.9).

The game is divided into three levels, and the level hierarchy of the game is represented in the flow chart of the system design (see Figure 3.2). To win the game the player must escape from the first room, then from the second room and finally from the third room. It's a linear game in which the difficulty increases with each level, and this difficulty lies in understanding what has to be done to overcome all the tasks, as usual in all escape rooms.

The game starts at the main menu (see Figure 3.3), where the user can set the volume (see Figure 3.6), learn the game controls (see Figure 3.5), exit from the game or start it. Once the start button is pressed, the first dialogue screen appears (see Figure 3.10). The user will be able to read the entire dialogue (which is detailed in the 4.5 section) using the "continue" button that will be on the screen. When the dialogue ends, the first level will begin.

On the first level, the user has to throw every piece of trash from the floor into the correct container: the milk carton to the yellow container, the glass bottle to the green container, the fish scrape to the brown container, the carton box to the blue container and the broken teddy bear to the gray container. Once there is no more garbage, the user has to interact with the key that just popped up. After interacting with the key, a cinematic of the padlock being opened will appear, and after that, the second dialogue. When the dialogue ends, the second level will start.

On the second level, the user must find an Allen wrench, a electric screwdriver and a combination wrench interacting with the drawers. All of them will be used to loosen screws. The Allen wrench will be for removing engine oil, the electric screwdriver for removing the wheels, and the combination wrench for removing the battery. That will be explained in a manual that the user has to interact with. After the vehicle is decontaminated, the user has to interact with the key that just popped up. After interacting with the key, a cinematic of the padlock being opened will appear, and after that, the third dialogue. When the dialogue ends, the third level will start.

On the third level, the user has to interact with the plastic waste, putting it into the crushing machine. Once this step is completed, the machine will have to be turned on introducing a code. This code, as well as the code for the smelting machine and the pelletizing machine, can be found if the player interacts with the painting on the wall. The user has to resolve the sum puzzle from the painting to get the codes. After the crushing machine finishes, the player will put the resulting product (shredded plastic) in the pelletizing machine, and that machine will return another product (granulated plastic). After putting that other product in the smelting machine and introducing the code, the user has to interact with the key that just popped up. After interacting with the key, a cinematic of the padlock being opened will appear, and after that, the fourth and last dialogue. When the dialogue ends, the game will end and the user will be redirected to the main menu again.

The game space is divided into three rooms, which will correspond to the three levels of the game. The main objects of the first room will be the trash can, the recycling bins and the garbage. In the second room, the main objects will be the vehicle to decontaminate, the tools to use, the drawers where they are hidden and the manual with the clues of what to do. In the last room, the main objects will be the crushing machine, the pelletizing machine, the smelting machine, the products obtained from the machines, the keypads, and the painting on the wall that will give the codes to turn on the machines.

4.2.1 Beat Chart

FIRST ROOM

TOD: Night.
STORY: You need to escape from the first room of the game.
PROGRESSION: throw the garbage to its corresponding container to get the key.
EST. PLAY TIME: 4 min.
COLOR MAP: Yellow, brown, blue, green, black, white and gray.
MECHANICS: Move, look around, interact with the trash, the containers and the key.
MUSIC TRACK: Escape room track.

SECOND ROOM

TOD: Night.
STORY: You need to escape from the second room of the game.
PROGRESSION: Decontaminate some of the parts of a car until to get the key.
EST. PLAY TIME: 6 min.
COLOR MAP: Red, gray, blue, black, white, green and yellow.
MECHANICS: Move, look around, interact with the tools, the vehicle, the manual, the drawers and the key.
MUSIC TRACK: Escape room track.

THIRD ROOM

TOD: Night.
STORY: You need to escape from the third room of the game.
PROGRESSION: Get plastic waste, crush it with a machine, pelletize it with another machine and melt it with one last machine to make a plastic key.
EST. PLAY TIME: 10 min.
COLOR MAP: Red, green, blue, black, white and gray.
MECHANICS: Move, look around, interact with the initial plastics, the machines, the keypad buttons, the painting, the products of the machines and the key.
MUSIC TRACK: Escape room track.

4.3 Mechanics

The mechanics are the same for all the levels. The user can move around, rotate the camera and interact with some objects. The camera will always be in first person. The controls will be explained on the controls screen, which can be consulted during any level and in the main menu. The user can also set the volume at any level or in the main menu.

Controls:

- Move: WASD keys or arrow keys
- Rotate the camera: Moving the mouse
- Interact with objects: Left mouse button
- Open the pause menu: ESC button
- Open the inventory: I button
- Put item back into the inventory: X button

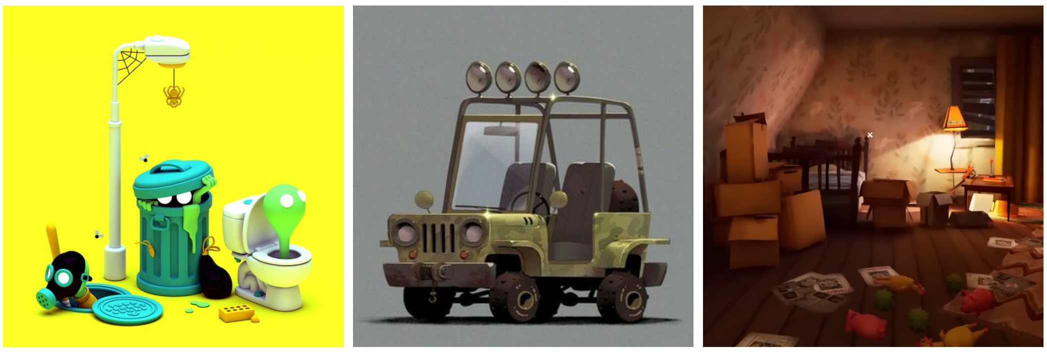
4.4 Game Art

4.4.1 Sound Design

The game has a soundtrack for the dialogues before the beginning of the levels that aims to sound like a creepy lullaby, getting the inspiration from the Lavender Town Theme, Pokemon [43]. The main menu has another soundtrack with a suspense tone to encourage the player to start the game. Finally, the three levels have the same soundtrack, which has a detective tone to engage the player in his/her goal of solving the levels. In addition, there are sound effects for almost every event (see the A.1 section):

- Pressing a button
- Opening a drawer
- Closing a drawer
- Opening a car hood
- Closing a car hood
- Using a key
- Getting an item
- Turning on a machine
- Pouring motor oil
- Ending a level
- Doing a wrong action
- Doing a right action
- Ending the game (clock alarm)

4.4.2 Visual References



(a) Antonio Hitos (3D Artist) -From Fubiz.net-

(b) Jacob Van (3D Artist) - From ArtStation.com-

(c) Hello Neighbor (Dynamic Pixels) -From YouTube-

Figure 4.1: Some of the visual references for doing the 3D models of the game

4.4.3 Game Logo

The game logo (see Figure 4.2) has been created with the software Krita. It is based on the Möbius circle [5], the international symbol of recycling, and the key in its center represents the fact that it is a Escape Room game, where the user need to get the key of each room to escape.

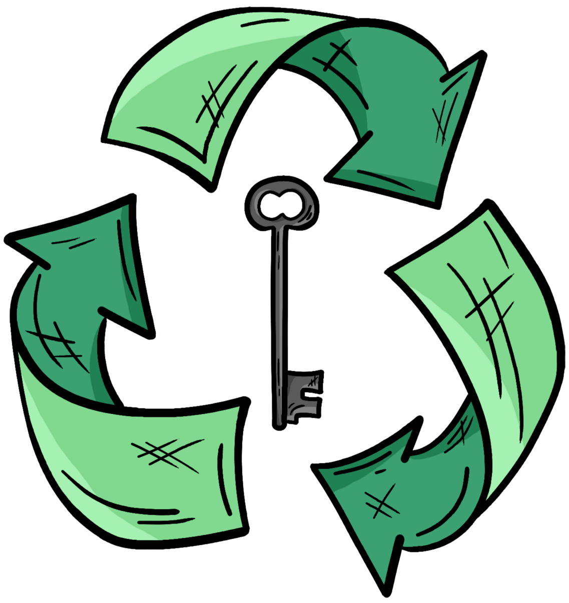


Figure 4.2: Game Logo

4.5 Dialogues

The dialogue text emerge on the screen as the user press the continue button (see Figure 3.10). This is the order in which the dialogues appear:

1. The user press the start button in the main menu
2. The first dialogue pops up
3. The first level takes place
4. The second dialogue starts
5. The second level takes place
6. The third dialogue appears
7. The third and last level takes place
8. The third and last dialogue emerges

4.5.1 First dialogue

- "¡Buenas noches! Te preguntarás qué está pasando… Te has dormido viendo las noticias. No me extraña, todo lo que hacen es hablar sobre el reciclaje, la contaminación, y de lo malo que es el plástico. Te parece aburrido, ¿verdad? No tiene nada que ver contigo, ¿verdad? Pues la verdad es que te equivocas. Pero, por suerte para ti, te he preparado un escape room para que aprendas un poco sobre el tema. Ah, y antes de que se me olvide… Si no escapas de las tres salas que tengo para ti, no despertarás. ¡Disfruta de tu pesadilla!"

4.5.2 Second dialogue

- "No está mal. Veo que, por lo menos, sabes reciclar. Ya haces algo por el planeta. Vamos a ponerlo más interesante… ¿Te gustan los coches? ¿Sabes qué hay que hacer para poder deshacerse de un coche cuando ya no lo quieres?"

4.5.3 Third dialogue

- "¡Bien hecho! Quiero decir, no ha sido un desastre. Solo te queda una sala, pero no te creas que va a ser fácil. Ya que te importa bien poco el impacto negativo que tiene el plástico, vas a tener que pasar por todo el proceso que convertirá unos cuantos vasos de plástico en tu llave de escape."

4.5.4 Fourth dialogue

- "Vaya, vaya. Has conseguido reciclar correctamente, descontaminar un coche y darle una nueva vida a lo que otros consideraban basura. Me has impresionado. Creo que ya es hora de que despiertes. Eso sí, espero que lo hagas como una persona nueva, más concienciada con el medio ambiente. O por lo menos recicla un poco, anda. ¡Buenos días!"

Chapter 5

Work Development and Results

Contents

In this character, the work developed during the main stages of the planning (see Figure 2.2) will be detailed. There will be also explained the deviations from the initial planning and the results achieved.

5.1 Related Research

Before starting with the work development itself, an approach for the game design was made through a research of similar projects, to see if the idea of this game has been already created. It also helped me see some elements that I wanted my video game to have and some others that I did not want. After this investigation, the research of recycling processes began, in order to carry out an accurate and realistic work.

5.1.1 Research of Similar Projects

There are some projects that could be related to my final degree project. The first one to talk about is “El Vidrio Misterioso” (EcoVidrio) [8], an online virtual escape room to educate on glass recycling in Castilla y León. EcoVidrio also made two other games in collaboration with Castilla- La Mancha and with Madrid, very similar to that one, named “El Quijote Recicla Vidrio” [7] and “La Historia del Vidrio” [10] respectively. However, neither of these two is available anymore. These three projects are the only recycling-themed escape room video games I have found, and they are basically the same game with other names.

Furthermore, there are some interesting initiatives of escape rooms “in real life” about recycling. Some of them are events from EcoVidrio that took place in places like Loeches (Madrid) or Torrijos (Castilla-La Mancha) among other localities [9]. The activities were made for students in high school and in the last years of primary school. Another example of a face-to-face recycling-themed escape room is the one developed by the UPNA (Public University of Navarra), whose objective has been to raise awareness about this subject and bring it closer to the university community [58].

To end up with, “Escape From Recycling” (School for Games) [12] is a 3D video game that takes place in a recycling facility, with escape room mechanics, but it does not educate about recycling. It can be found on the platform Itch.io, and it is still in development.

Contributions

This final project degree is going to be related to “El Vidrio Misterioso”, “La Historia del Vidrio” and “El Quijote Recicla Vidrio” for being an educational escape room about recycling, but it is going to have more visual and interactive goals, not only with the goal of resolving words. These three examples are 2D video games where the user is only able to scroll a vertical window, but this project is going to be in three dimensions. Thanks to that, the player will be able to actually move the character with freedom through the environment of every level and interact with its elements. Moreover, it will address more aspects of recycling, not only the glass part.

Concerning the escape rooms by EcoVidrio and by the UPNA, the main difference is that this project is going to be a virtual escape room, not a face-to-face one. Unlike these ones, there will be no need to mobilize people in a specific space, they will be able to play this game remotely through their computer.

Technically speaking, the most related project is “Escape From Recycling”, because it is a first person video game in three dimensions with an escape room format, and it also takes place in a recycling facility. However, the main difference will remain on the objective of the game, because this one is not related to recycling.

5.1.2 Research of Recycling Processes

In order to learn more about the processes on which the game levels are based, I visited the recycling plant from Tortosa, Reciclatge Forés S.L. (see Figure 5.1). There they explained to me step by step the procedure for waste triage (for the first level), the procedure for the decontamination of a vehicle (for the second level) and the procedure for the crushing and melting of plastic (for the third level).

Figure 5.1: Reciclatge Forés S.L. company

First of all, part of the waste arrives already separated thanks to the city's recycling containers (see Figure 5.2). However, most of the waste must be separated manually due to citizen errors at the time of recycling, or due to the waste in the containers for "other" trash. The containers in the recycling plant are not like the usual ones, but in this game the user must separate the waste in the containers that he will find on the street, to learn how to do it in his/hers everyday life.

Secondly, while decontaminating a vehicle many parts are removed: all contaminating fluids (engine oil, brake fluid, fuel, antifreeze fluid, gearbox valve, etc.), the battery, the gases, the airbags, the windows, the wheels, etc. At this level of the game, only the wheels (see Figure 5.3a), the battery (see Figure 5.3b), and the engine oil (see Figure 5.3c) are going to be removed, as they are very visible, accessible, and understandable parts of the process.

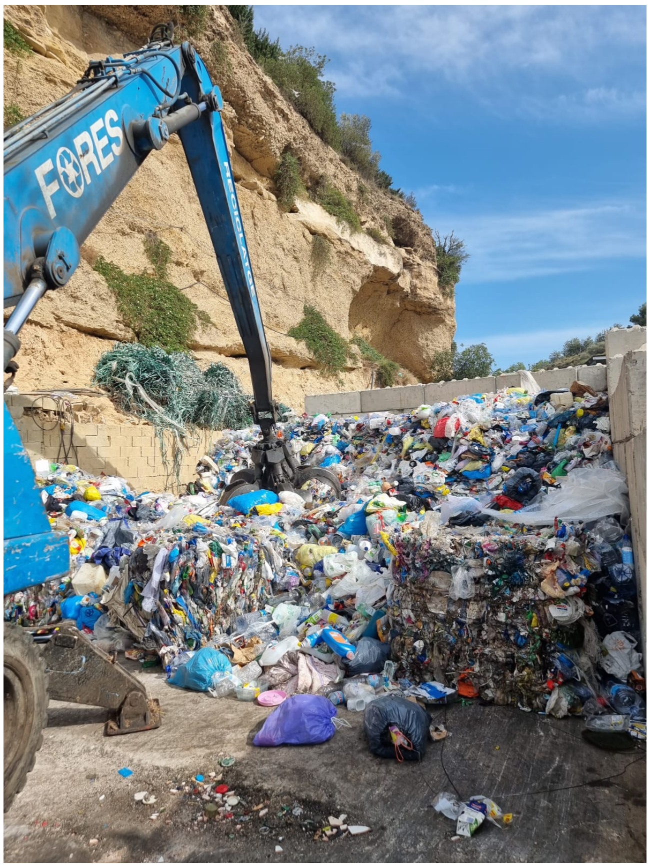
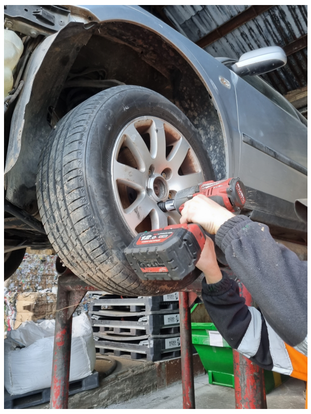
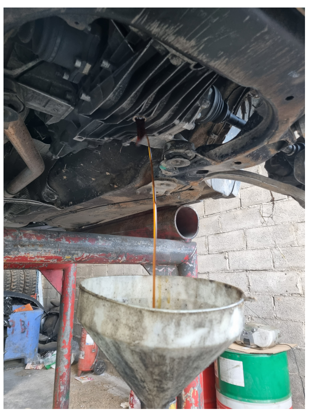


Figure 5.2: Plastic waste separated from the other garbage



(a) Process of removing the wheels

(b) Process of removing the battery



(c) Process of removing the motor oil

Figure 5.3: Process of decontaminating a vehicle showed by Reciclatge Forés S.L.

Finally, the process of creating new objects with recycled plastic is not done entirely in a recycling plant, but they have shown me the first steps that they carry out and they have explained to me how the following ones are carried out. In general, this process is summarized in that a shredding machine crushes the recyclable plastic elements (see Figure 5.4a), a pelleting machine converts the resulting pieces into small granules (see Figure 5.4b), and finally a plastic injection machine melts said granules and injects them into a mold to create the new object. For the game, the processes are more simplified than they actually are in real life, as are the machines that are used, for the purpose of not confusing the player and so that a clear and more direct message reaches them.

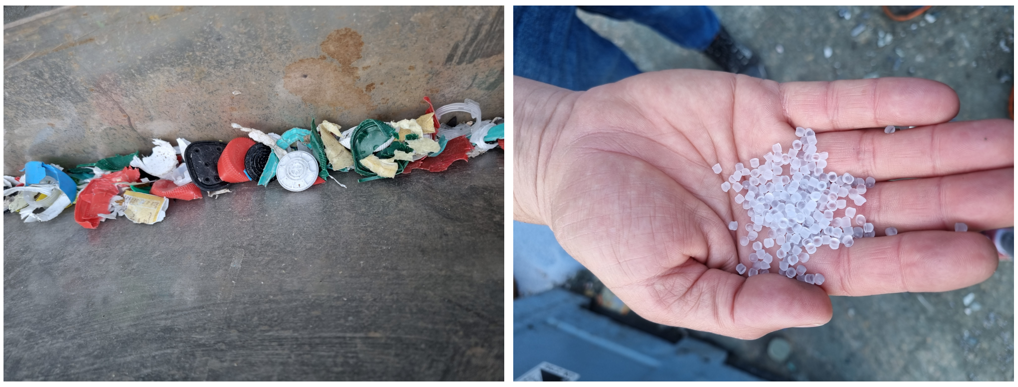

(a) Shredded plastic (b) Granulated plastic

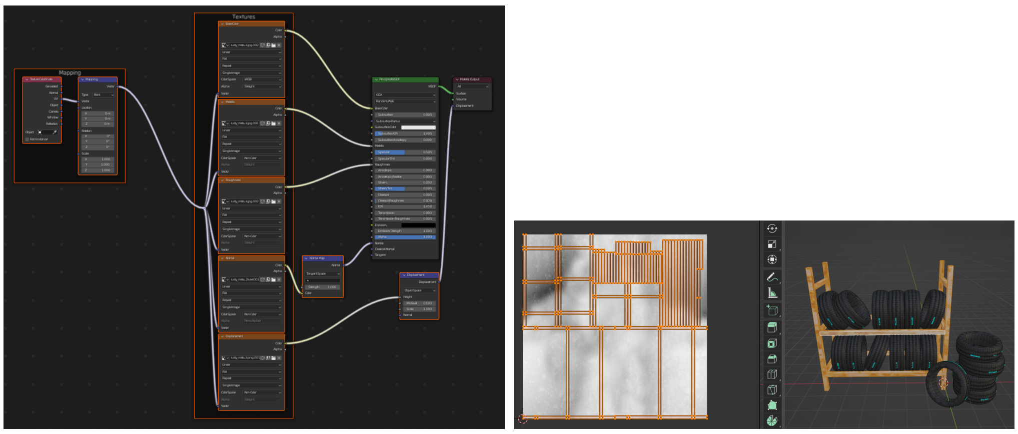
Figure 5.4: Two states of the plastic before being melted

5.2 Work Development

There will be explained most relevant aspects of the developed work: the Modeling, the animating and the coding of the whole video game. The main differences between the initial planning (see Table 2.1) and the final planning (see Table 2.2) are that, in the end, I didn't create my own textures and sounds. I preferred to invest my time in more relevant aspects, such as programming, 3D modeling and designing the game. I downloaded all the textures and sounds from the Internet, because if I had created it myself, apart from the time I would have spent producing it, I would have also spent time learning how to do it right. However, once the textures were downloaded, I did the process of texturing the 3D models (see more details in the 5.2.1 section).

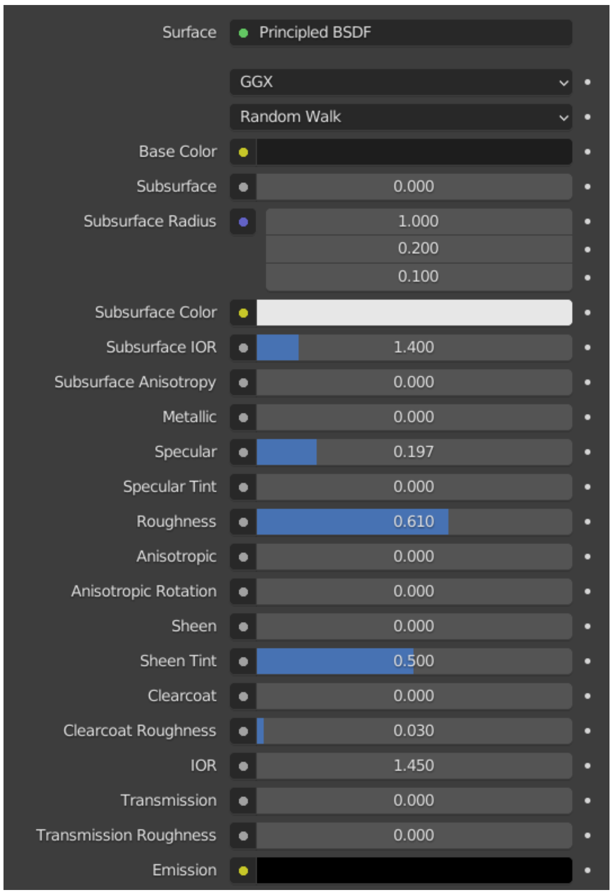
5.2.1 3D Modeling

All the 3D models of the video game are made by me, using the tool Blender. I also textured some objects using that tool, editing its shader nodes on the shader editor (see Figure 5.5a) and unwrapping its faces and editing them on the UV editor (see Figure 5.5b). For the models that have not needed textures, I edited its material values (see Figure 5.6).



(a) Shader Nodes for the textures (b) Unwrap of the faces of the model

Figure 5.5: Texturing process



Surface	Principled BSDF
	GGX
	Random Walk
Base Color	
Subsurface	0.000
Subsurface Radius	1.000
	0.200
	0.100
Subsurface Color	
Subsurface IOR	1.400
Subsurface Anisotropy	0.000
Metallic	0.000
Specular	0.197
Specular Tint	0.000
Roughness	0.610
Anisotropic	0.000
Anisotropic Rotation	0.000
Sheen	0.000
Sheen Tint	0.500
Clearcoat	0.000
Clearcoat Roughness	0.030
IOR	1.450
Transmission	0.000
Transmission Roughness	0.000
Emission	

Figure 5.6: Material properties of a tire

During the modeling processes, I used some Blender modifiers (see Figure 5.7):

- Array modifier: for objects like the drawers from the workshop cabinet of the second level, because it duplicates and arranges objects along an axis.

- Bevel modifier: for almost every model of the game, because it adds rounded edges to the objects, giving them a more cartoon look.

- Boolean modifier on its difference mode: for objects like the electric screwdriver of the second level, to combine the tool itself with its base in a subtractive way (see Figure 5.8).

- Decimate modifier: for a few models to reduce their polygon count.

- Mirror modifier: for objects like the bins of the first level, because it duplicate the geometry of an object through an axis, creating symmetrical models.

- Screw modifier: for the rope that holds the light bulbs of the first level, because it extrudes and rotates a 2D object, creating a 3D object with a helical shape (see Figure 5.9).

- Solidify modifier: for objects like the carton box of the first level, because it gives thickness to a thin surface, creating a solid 3D object. Without the solidify modifier, some faces would look transparent in unity because of where their normals are pointing to.

- Subdivision Surface modifier: for few objects like the wheel tiles of the second level, because it smooths and subdivides the geometry of an object, creating a more detailed and rounded surface, but also a more high poly one.

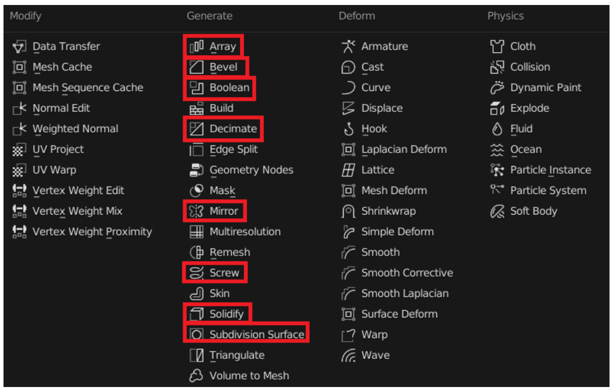


Figure 5.7: Blender modifiers used

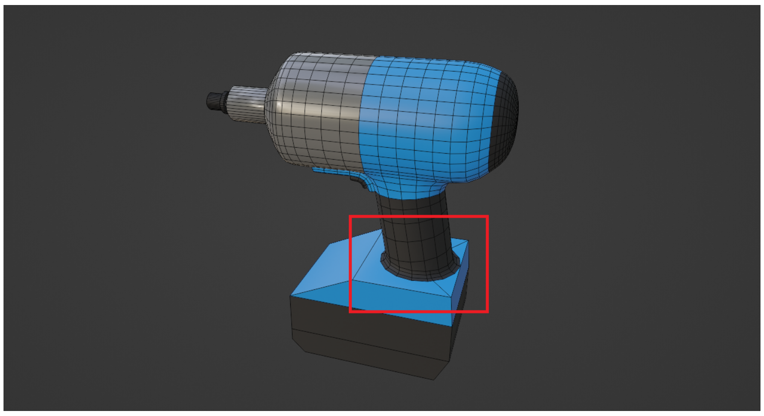
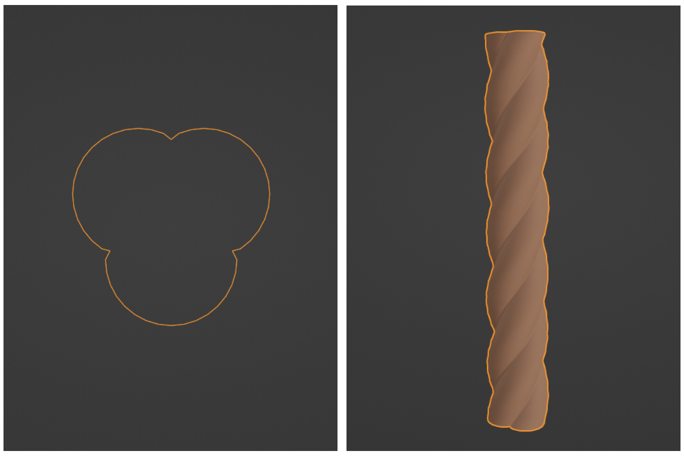


Figure 5.8: Combination of the base and the tool of the electric screwdriver

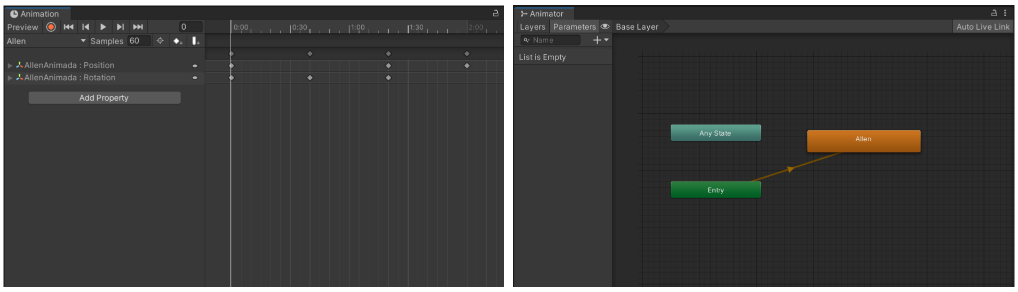


(a) 2D shape of the rope before using the screw modifier (b) 2D object extruded with the screw modifier

Figure 5.9: Screwing process

5.2.2 3D Animating

All the animations of the elements of the game have been made in Unity. Some of them have been produced through key frames at animations (see Figure 5.10a) and animators (see Figure 5.10b) that control the animations: the black Fade in at the beginning of the levels, the movement of some pieces of the machines from the third level, the falling of the key when finishing all the tasks, the apparition of elements like the keypads or the plastic products from the machines at the third level, the rotation of the logos at the main menu screen (2D animation) and the cinematic of the key entering into the padlock, rotating and opening it at the end of the levels. This cinematic (see Figure 5.11) has been recorded in Unity too, using the Recorder package from Unity Technologies, which allows to record animations from the Unity Editor during Play mode.



(a) Animation of the Allen key (b) Animator of the Allen key

Figure 5.10: The keyframes of the animation of the Allen key and its animator

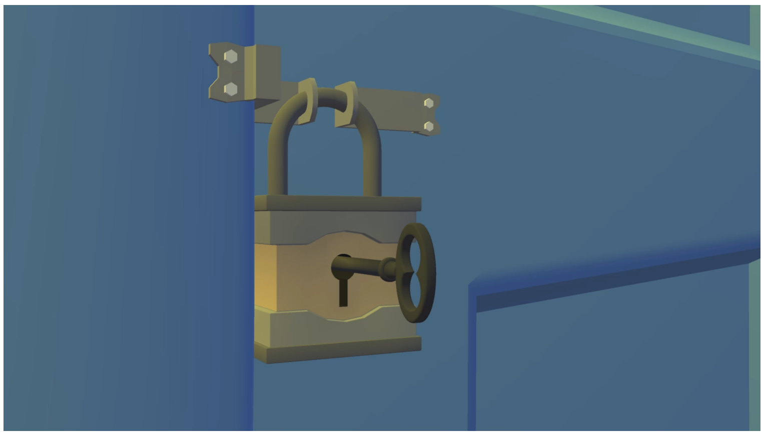
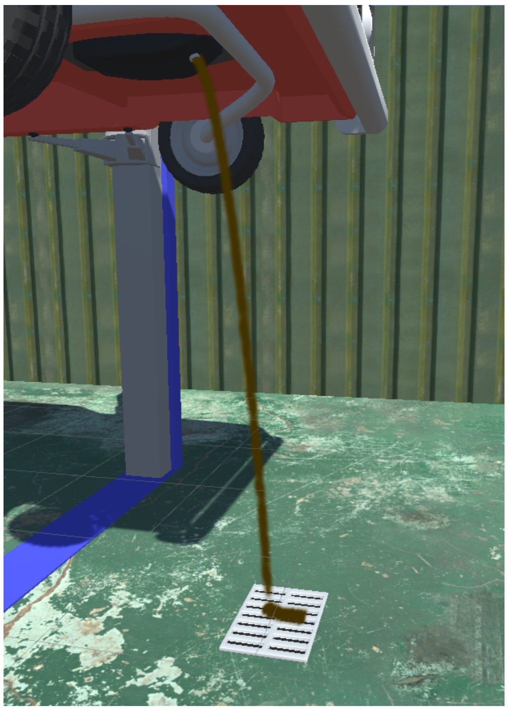


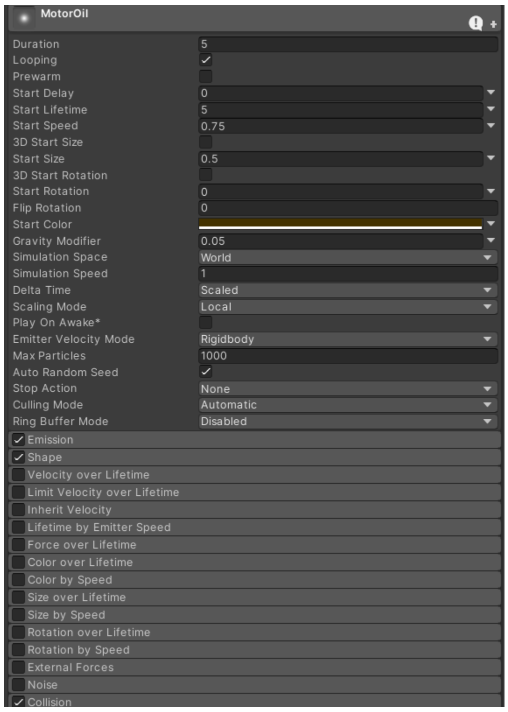
Figure 5.11: Fragment of the final animation of the levels

Other animations have been produced by code, because it was easier to take into account some other aspects at the same time, like if the drawer was already open or if the car was already on the top of the elevator. All the scripts are very similar, the only elements that change between them are the Boolean variables and the type of animation, changing its position or its rotation. These scripts (see Figure B.2) initialize their booleans (if they have any), their position (if necessary) and their sound effects (if they have any) at the **Start()** function. Then, they have a **ObjectInteraction()** function changing the word "object" for the object name, drawer in this case, that is called from the **RaycastScript** script (see Figure B.5) when the interaction starts. The **ObjectInteraction()** function starts the co-routine that does the animation, in this case **OpenTheDrawer()**, that checks the booleans, starts the sound effects and changes the position or rotation of the object (in this case, the position through the X axis).

Another type of animation has been created, the motor oil pouring of the third level (see Figure 5.12a). For this one, I used a Particle System to imitate the effect of a liquid falling from the motor. I changed some of the particle values to achieve that visual effect: the start speed, the start size, the color, the gravity, the render mode (to stretched billboard), the speed scale, the length scale, the rate over time from the emission, the shape and its angle (with a cone) and the collision values (it collides with the sewer) (see Figure 5.12b).



(a) Particle System for motor oil pouring



MotorOil

Duration	5
Looping	✓
Prewarm	
Start Delay	0
Start Lifetime	5
Start Speed	0.75
3D Start Size	
Start Size	0.5
3D Start Rotation	
Start Rotation	0
Flip Rotation	0
Start Color	
Gravity Modifier	0.05
Simulation Space	World
Simulation Speed	1
Delta Time	Scaled
Scaling Mode	Local
Play On Awake*	
Emitter Velocity Mode	Rigidbody
Max Particles	1000
Auto Random Seed	✓
Stop Action	None
Culling Mode	Automatic
Ring Buffer Mode	Disabled

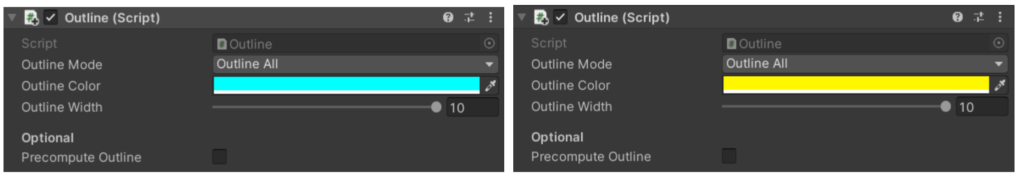
✓ Emission
✓ Shape
Velocity over Lifetime
Limit Velocity over Lifetime
Inherit Velocity
Lifetime by Emitter Speed
Force over Lifetime
Color over Lifetime
Color by Speed
Size over Lifetime
Size by Speed
Rotation over Lifetime
Rotation by Speed
External Forces
Noise
✓ Collision

(b) Particle System values

Figure 5.12: Particle System imitating the effect of motor oil pouring

5.2.3 Coding

For this project, I used the free asset Quick Outline, by Chris Nolet, from the Unity Asset Store [27]. This asset has allowed me to illuminate the shape of the objects with the color I wanted. For using it, I added an Outline component to the items (see Figure 5.13a) and to the other intractable objects (see Figure 5.13b), as well as a tag "Item" and a tag "Interactuable". Then, with an script for every level (see Figure B.3), all the item objects, intractable objects and key objects are find and its outline component gets disabled after a short time with the co-routine **Illumination()**. Having all the outline components enabled at the beginning of the game and disabling them later allows them to keep their assigned color, otherwise all of them would have the same default color. Then, at the **Update()** function from the **RaycastScript** (see Figure B.4), it is determined that the outlines of the objects will be enabled when the cursor hovers over them, and at the moment in which it stops doing so, the outline will be deactivated.



Outline (Script)

Script	Outline
Outline Mode	Outline All
Outline Color	🟦
Outline Width	10
Optional	
Precompute Outline	☐

(a) Outline for items

Outline (Script)

Script	Outline
Outline Mode	Outline All
Outline Color	🟨
Outline Width	10
Optional	
Precompute Outline	☐

(b) Outline for intractable objects

Figure 5.13: The outline component for items and the other intractable objects

One of the most important scripts in the game is the **RaycastScript**, which is located at the main camera. This script interacts with almost every other script of the project (see Figure B.1), and it is in charge of three main functionalities:

- Detecting the type of object the cursor hovers and of changing the cursor sprite to the hand one. It does it in the **Update()** function (see Figure B.4), using a raycast that detect the colliders of the objects, as well as its tags. Then, it uses its tags to determinate their pertinent actions. For the interactable objects, it calls another function, **InteractWith()** (see Figure B.5), sending it all the information of the object that it might need: its interacting component, the current item that the player have equipped (a string variable from the **RaycastScript**), the ID of that current item (an int variable from the RaycastScript), the drawer component to call the **DrawerInteraction()** function (see Figure B.2) from its animation script if the object is a drawer, the wheel component to call the **WheelInteraction()** function from its animation script if the object is a wheel, and the button component to call the **SendKey()** function if the object is a keypad button.

- Determine what happens when the user interacts with an item while having the correct item equipped. Every interactable object has its interacting component which says what type of object it is, and every item has its item component which have a string variable that says what type of item it is. The **InteractWith()** function (see Figure B.5) compares these two types, and if they match, the code determines which actions have to be done: remove the equipped item from the inventory, play a sound effect, add one point to an int variable, start an animation, send a number, etc.

- The last utility of the **RaycastScript** is to start a co-routine when the key of each level is interacted (see Figure B.6) that controls the next cinematic and dialogue.

There are three scripts to control the inventory and its items: the **Inventory** script, the **Slot** script and the **Item** script:

- The **Item** script (see Figure B.7) is used as a component of each item, contains their type, their ID and their icon for the inventory. It is the one that has the **ItemUsage()** function, that equips the item picked from the room or selected from the inventory and updates the current item variable from the **RaycastScript**. Its **Update()** function determines that pressing the X button equals to put the item back to the inventory.

- The **Inventory** script (see Figure B.8) is a component of the Player, the controllable character, and is the one that detects if the user have pressed the I button to open the inventory, the one that has the **AddItem()** function to add the items to the inventory and the function **DeleteItem()** to delete them. That script is able to do all those things thanks to access to the Slot component of its slots.

- The **Slot** script (see Figure B.9) detects the pointer click in the slots of the inventory, and works like a bridge between the item components and the inventory script. It also updates the item icons to ticks with the **UpdateEmptySlot()** function when the user will not use these items anymore.

The Player, the character that the user moves in first person, is actually a capsule with a capsule collider, a character controller, an inventory component (as said before) and a playerScript component. The **PlayerScript** script (see Figure B.10) is the one to determine that the user can rotate the camera (which is placed at the "head" of the character) with the mouse, using the Input.GetAxis() mouse X and mouse Y, and move the character with the WASD or the arrow buttons, using the Input.GetAxis() horizontal and vertical. This script was prepared to add a jumping mechanic if that were necessary because of its use of gravity and ground detection, but in the end that mechanic was not useful for any of the resulting levels.

There are two scripts to control the three keypads of the third level, the **Keypad-Script** script (see Figure B.12) and the **ButtonKeypad** script (see Figure B.11):

- All the buttons of the keypad have a component buttonKeypad of the **ButtonKeypad** script, which only have a string with the number of the button and the function **SendKey()**, that gets the keypadScript component from its parent and send it the password introduced through the function **PasswordEntry()**.

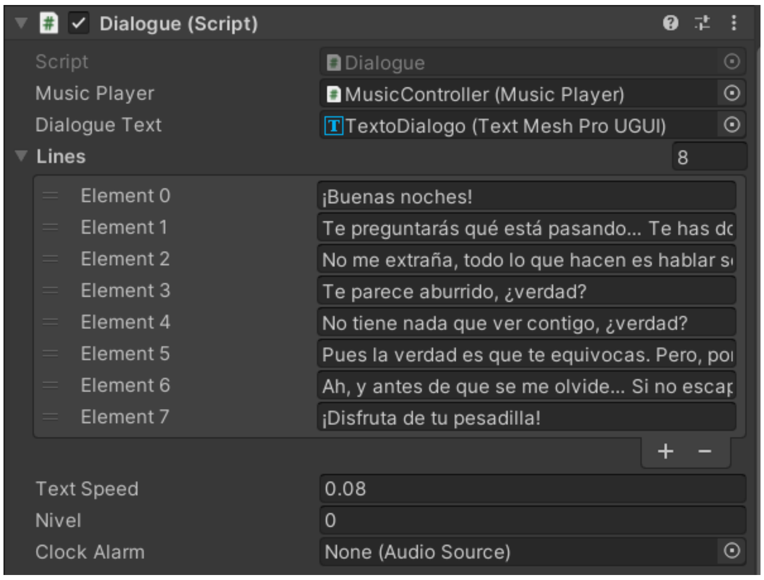
- In the **KeypadScript**, three things could happen in that function:

  A) The key pressed is a number so, if the number of digits pressed does not exceed the number of digits in the password, it will add it and display it on the screen.

  B) The key pressed is the red button, so it clears the keypad screen calling the function **Clear()**.

  C) The key pressed is the green button, so it calls the **Enter()** function that will compare the code pressed with the password initially determined. If the password is correct it will turn green (see Figure 5.28b), the machine will produce its new product by the co-routine **CallMachine()**, and the keypad screen will be cleared after less than a second by the co-routine **WaitAndClear()**. If the password is incorrect it will turn red (see Figure 5.28a), it will only clear the screen.

Every level has its own scene in unity, and there is also a scene for the main menu and for every dialogue. In those scenes, there is a panel where the text appears, and it has a Dialogue component (see Figure 5.14). Te **Dialogue** script (see Figure B.13) starts as some other scripts of the game, making the cursor visible in the **Start()** function. In the same function, after finding the **MusicScript()** script to control the music and emptying the dialogue text, it calls the function **StartDialogue()** that puts the index to cero and starts the co-routine **TypeLine()**. This co-routine writes character by character all the characters of a line with the specified speed. When the button "Continuar" is pressed, it calls the function **ContinueDialogue()**, that continues with another line if there is one calling the **NextLine()** function, and if not, it stops all the co-routines. In the **NextLine()** function, the **TypeLine()** is called again with the text of the new line, an when all the text of all the lines is over, depending on which level the player is in, it starts one scene or another, or end the game by going back to the main menu scene with the **BackToMainMenu()** co-routine.



Dialogue (Script)

Script	Dialogue
Music Player	MusicController (Music Player)
Dialogue Text	TextoDialogo (Text Mesh Pro UGUI)
Lines	8
Element 0	¡Buenas noches!
Element 1	Te preguntarás qué está pasando... Te has dc
Element 2	No me extraña, todo lo que hacen es hablar s
Element 3	Te parece aburrido, ¿verdad?
Element 4	No tiene nada que ver contigo, ¿verdad?
Element 5	Pues la verdad es que te equivocas. Pero, poi
Element 6	Ah, y antes de que se me olvide... Si no escap
Element 7	¡Disfruta de tu pesadilla!
Text Speed	0.08
Nivel	0
Clock Alarm	None (Audio Source)

Figure 5.14: Dialogue component from the dialogue panel

The last script I am going to talk about is the **MusicPlayer** script (see Figure B.14). In the **Start()** function of this script, aside from playing the chosen audio, the music volume value is set. By default it will be 1, the maximum, but the user can chose the value that he/she wants by using the volume slider from the volume settings screen of the main menu (see Figure 3.6) or the one from the options screen of any level (see Figure 3.4). Thanks to the **PlayerPrefs** float "volume", it doesn't matter if one scene changes to another one, the value of the music volume will still the same that the user decided, even if he/she closes and opens the game again.

5.3 Results

All the initial objectives set out in section 1.2 have been achieved. According to the main milestones, it has been developed a video game with different levels related to different aspects about recycling: separating waste, decontaminating a vehicle and the plastic recycling process. This game could be used by educational institutions to help young people to become aware of the environment, and the challenges are enough interesting to engage the young people, to whom it is focused. It can be said that the secondary milestones have also been accomplished. All the 3D elements of the game are made in Blender by me, taking into account the amount of polygons that they have to get an efficient game. All their animations are made by me too in Unity, the software used to implement the game itself. Enough research has been made to provide real information, and the mechanics are simple but attractive. Moreover, there is given a lot of visual and sound information to make all the user actions clear. In brief, the result of this video game is as expected, a somehow innovative and educational serious game conscious of the environment.

In addition, the heads of the company Reciclatge Forés S.L., the recycling company where the research has been done (see section 5.1.2), have been able to play the escape room once it has been finished (see Figure 5.29). They confirmed that the information about the recycling processes is veracious, and that they found it a very entertaining way to teach those facts.

The access to the GitHub repository of the Unity project: https://n9.cl/gwq7j

The build of this video game for Windows: https://n9.cl/f9jwf

The link to access the video resume of this project: https://youtu.be/EgO41DcrxS8

To end up with, there are some in-game pictures of all the three levels to see the visual results.

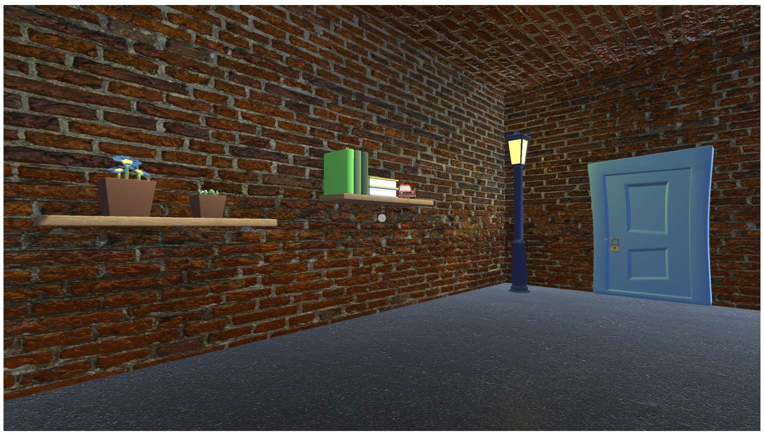


Figure 5.15: In-Game image from the first level: Decoration

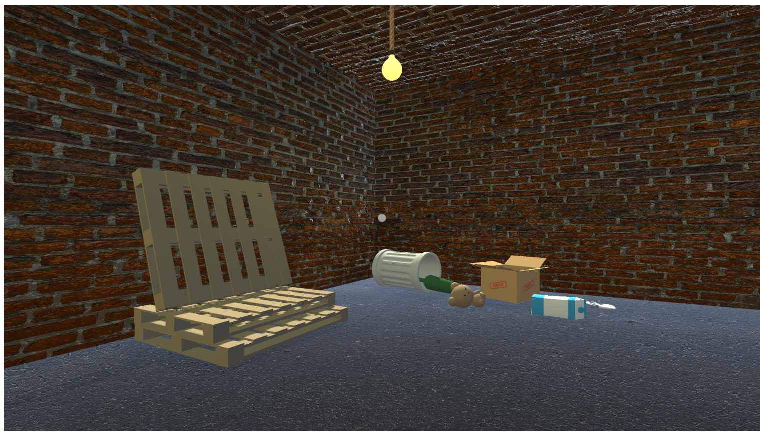


Figure 5.16: In-Game image from the first level: The waste to recycle

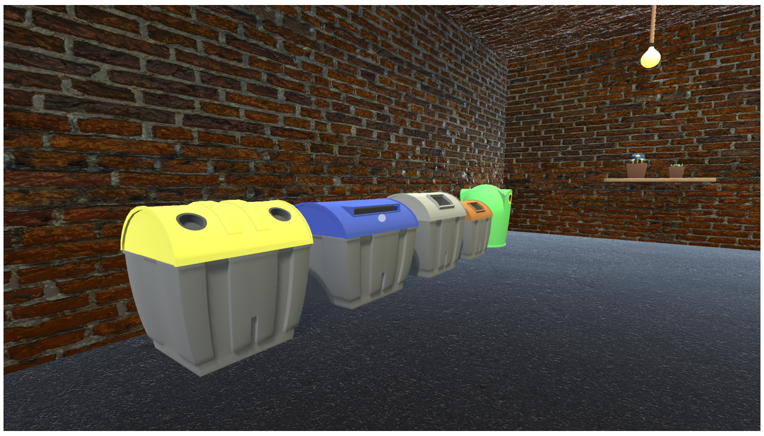


Figure 5.17: In-Game image from the first level: The recycling bins

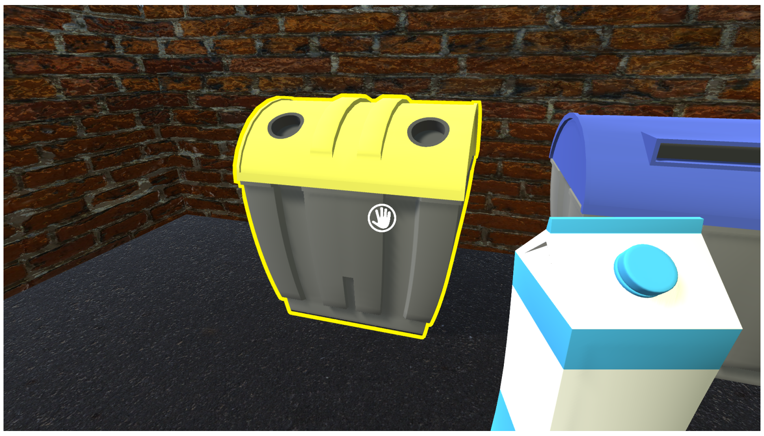


Figure 5.18: In-Game image from the first level: Having an item equipped and interacting with the right recycling bin

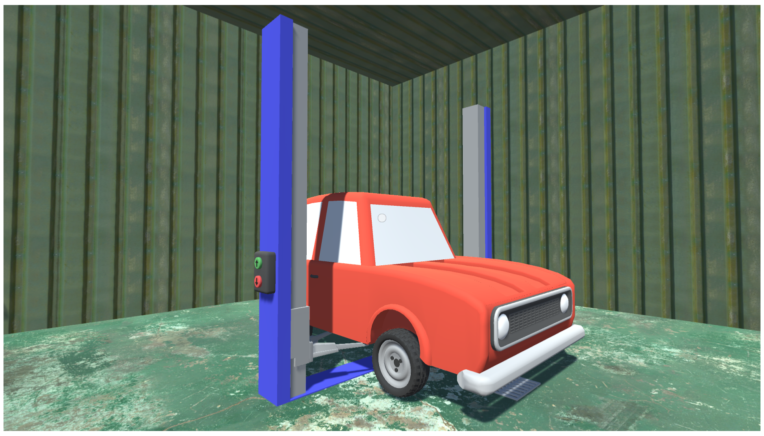


Figure 5.19: In-Game image from the second level: The car to be decontaminated

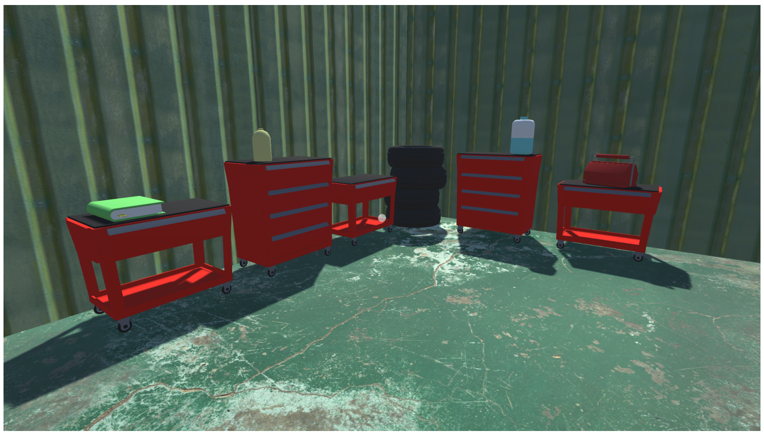


Figure 5.20: In-Game image from the second level: The drawers where the tools are hidden and the manual with the clues

Figure 5.21: In-Game image from the second level: Decoration

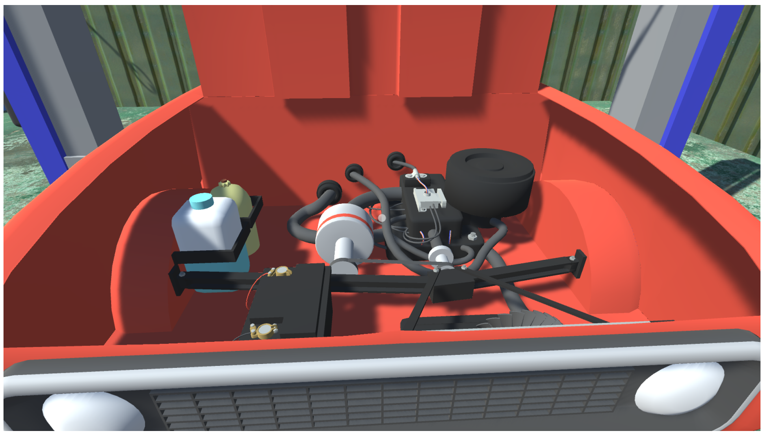


Figure 5.22: In-Game image from the second level: The engine from which the battery is removed

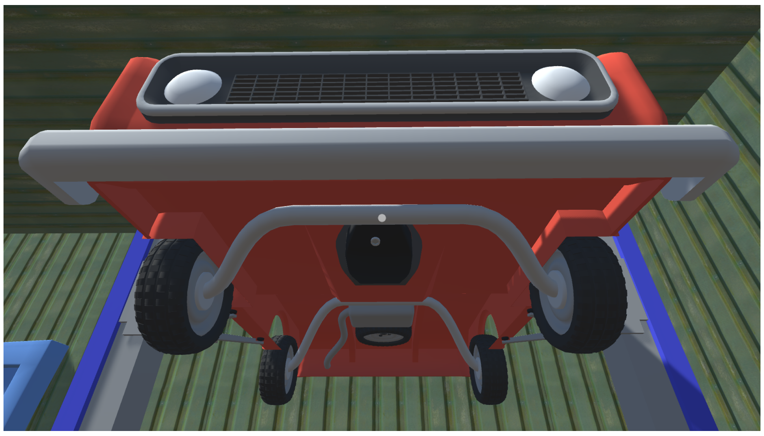


Figure 5.23: In-Game image from the second level: The underside of the car where the Allen key is used

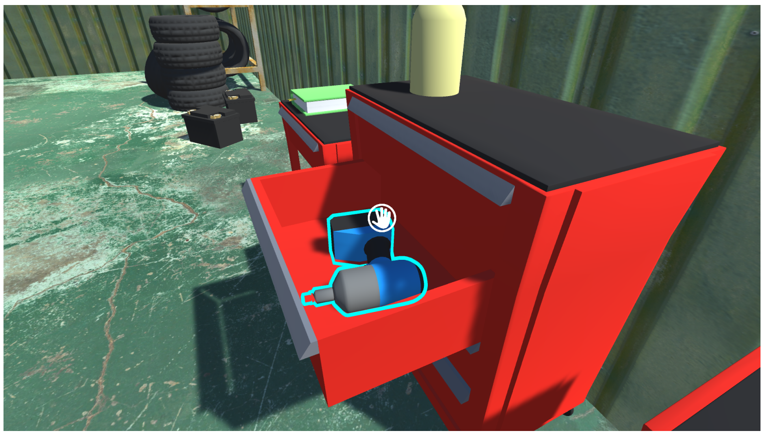


Figure 5.24: In-Game image from the second level: Interacting with a tool from inside a drawer

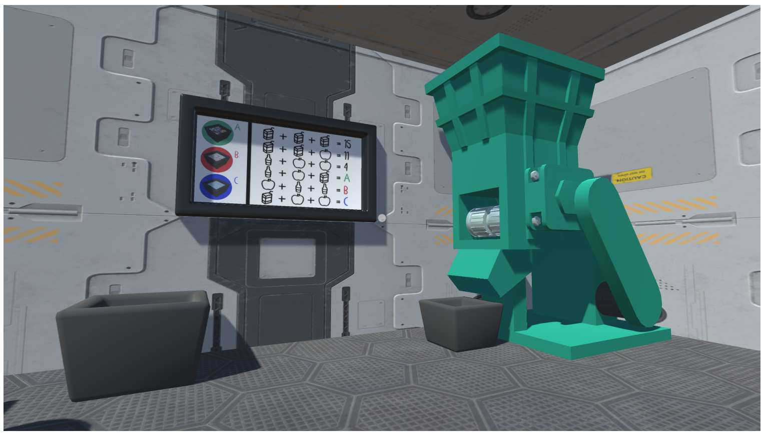


Figure 5.25: In-Game image from the third level: The crushing machine, the painting with the clue and the first item to get

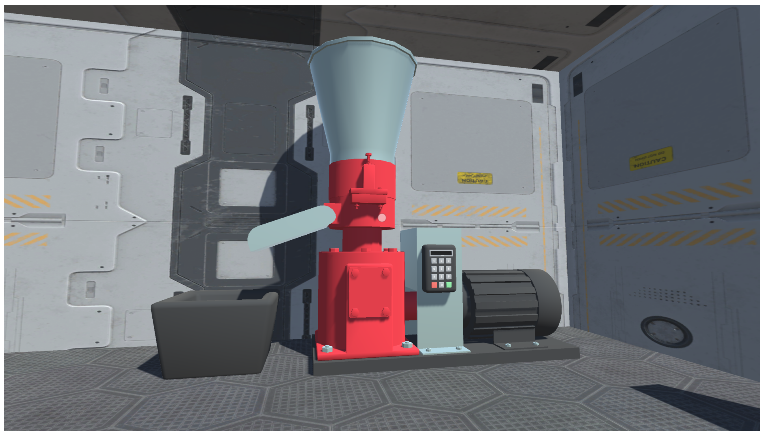


Figure 5.26: In-Game image from the third level: The pelletizing machine

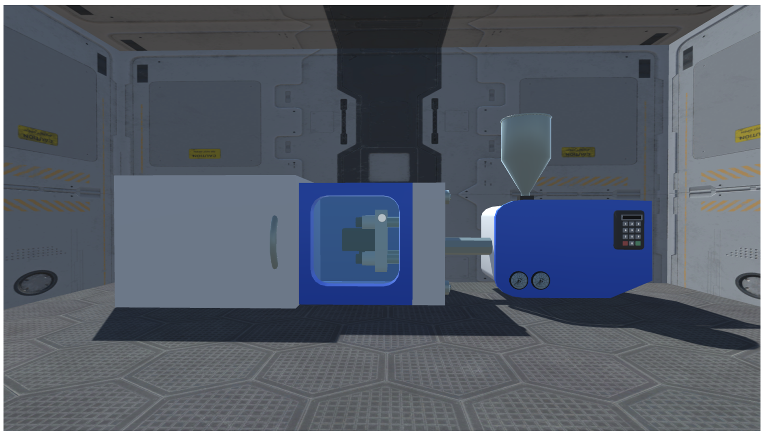
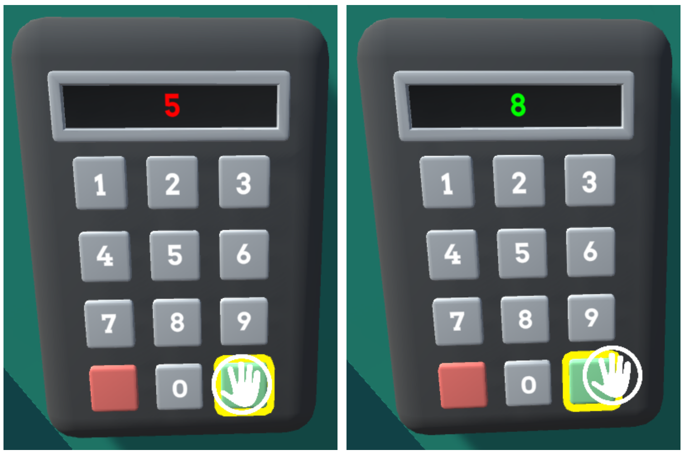


Figure 5.27: In-Game image from the third level: The melting machine



(a) Wrong password (b) Right password

Figure 5.28: Keypad from a machine of the third level

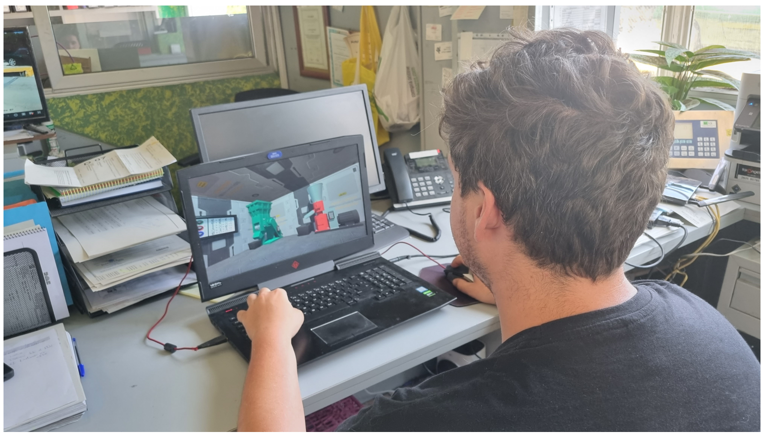


Figure 5.29: One of the heads of the company Reciclatge Forés S.L. testing the project

CHAPTER 6

Conclusions and Future Work

Contents

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

6.1 Conclusions

I have made some other video games through this degree, but all of them dedicated to entertainment only. As soon as I found out what serious games were, I decided I would develop one at some point. The end-of-degree project has seemed like the perfect time for it, since I have been able to apply everything I have learned during these four years and focus it on an objective that is beyond entertaining. Another change respect to the projects carried out during this degree has been to do this work alone. Most of these projects have been done in groups, and I have missed dividing the tasks among all the members. However, it has been a very enriching challenge that has prepared me more for my future professional life.

Besides from what I learned during the degree subjects, I had never worked as much in Unity and Blender as during the external practices, which has given me more knowledge about the two tools that has been very useful during the development of this video game. Even if I used 3dsMax during this degree, I have spent many more hours on Blender, which has made me decide to use this tool. Although the 3dsMax materials have a better finish than those in blender, I have edited the values of my materials in Unity to compensate for that deficiency.

At the beginning of the development of this video game, its narrative was something that had not even crossed my mind. However, as I played various escape rooms for inspiration, I realized that its narrative was a very important aspect. It is not necessary that they have a very elaborate story behind it, but that they follow the thread of a plot makes them much more attractive. That's why I decided to give this escape room its "nightmare plot" summarized in the 4.1.1 section, and I completely believe that this change will attract the target audience more.

Even if I fulfilled my initial objectives, I would have liked it to have more playable time and to be more "professional" looking, but anyways I learned a lot about 3D modeling and coding through the development of this project, and I'm proud of the obtained video game. I also know that if I had more time to code more levels and to detail the models and create my own textures, I would have liked the result more, but I am satisfied with the results taking into account that I have dedicated many hours this semester to the external internships and the scholarship for initiation to research that they gave me at the end of them.

6.2 Future work

According with what was said in the previous section (6.1), if I decide to continue with the development of this project in a future, I would add more levels about more aspects of recycling, like the process of recycling the paper, glass or metal. I would also create my own textures for my 3D models, as I planned initially, and model more objects and add more details to the ones that are already created. A few classmates tested this game, but if I continue developing it, I would look for more people within my target audience to test it.

Besides all the things mentioned, I would like to add different levels of difficulty, adding more clues and reducing the mechanics (like not having an inventory or not having to enter a password in the third level) to make it easier for younger kids. This way, my initial objective of presenting this escape room to adolescents in high schools on the World Environment Day (June 5th), could also be extended to school children.

Another option that I have in mind for the future is, instead of presenting the game myself to educational institutions, focus it on a specific company and sell it to them so they can promote themselves with it. What is more, the company Reciclatge Forés S.L., the recycling company where I did my research (see section 5.1.2), played and liked the escape room and are actually interested in use this project for marketing purposes (see Figure 5.29).

What I will surely do in the very near future is to write an article about this project to present it to the CEV23 ("Congreso Español de Videojuegos"), a spanish video game congress that will take place this year, 2023. In my work environment, my coworkers are writing articles all the time, and they have been the ones who have told me about this congress and who encouraged me to present my final degree project there, as many other students from the video game design and development degree from universities throughout Spain have done in previous years. Moreover, the CEV shares its conference program with the Guerrilla Game Festival, and will be held within the Madrid in Game Summit, the professional meeting to promote the video game industry.

Bibliography

[1] European Environment Agency. Data about the waste produced and recycled in europe. https://www.eea.europa.eu/en/topics/in-depth/waste-and-recycling. Accessed: 2023-05-19.

[2] Ahkam. Sprite for the icon to close the 2d screens. downloaded from freeiconspng. https://www.freeiconspng.com/img/30230. Accessed: 2023-05-19.

[3] Charlotte Baglioni and Dario Barresi. Texture for the floor of the first level. downloaded from poly haven. https://n9.cl/amn1ti. Accessed: 2023-05-19.

[4] Pillow Castle. Superliminal. https://n9.cl/tj0ae. Accessed: 2023-05-04.

[5] Las Palmas de Gran Canaria Town Hall. Möbius circle. https://www.laspalmasgc.es/es/areas-tematicas/limpieza-y-reciclaje/separacion-y-reciclaje/los-simbolos-del-reciclaje/. Accessed: 2023-05-10.

[6] Unity Documentation. Requirements for unity 3d. https://docs.unity3d.com/2021.2/Documentation/Manual/system-requirements.html. Accessed: 2023-05-04.

[7] Ecovidrio. El quijote recicla vidrio. https://www.escaperadar.com/escape-room-online/ecovidrio/el-quijote-recicla-vidrio. Accessed: 2023-05-04.

[8] Ecovidrio. El vidrio misterioso. https://www.escaperadar.com/escape-room-online/ecovidrio/el-vidrio-misterioso-escape-room. Accessed: 2023-05-04.

[9] Ecovidrio. Escape rooms in real life by ecovidrio. https://www.ecovidrio.es/notas-de-prensa/escape-room-del-reciclaje. Accessed: 2023-05-04.

[10] Ecovidrio. La historia del vidrio. https://ampagarcialorca.org/escape-room-la-historia-del-vidrio/. Accessed: 2023-05-04.

[11] FeliUsers. Sound effect of opening a car hood. downloaded from freesound. https://freesound.org/people/FeliUsers/sounds/682343/. Accessed: 2023-05-19.

[12] School for Games. Escape from recycling. https://s4g.itch.io/escape-from-recycling. Accessed: 2023-05-04.

[13] Plastic Soup Foundation. Texture for the shredded plastic of the third level. downloaded from plasticsoupfoundation.org. https://n9.cl/dl3bu. Accessed: 2023-05-19.

[14] FRDMN. Texture for the carton box of the first level. downloaded from vecteezy. https://n9.cl/738hd. Accessed: 2023-05-19.

[15] Freepik. Sprite for the gear icon. downloaded from flaticon. https://n9.cl/3pl5jg. Accessed: 2023-05-19.

[16] Dave Gandy. Sprite for the maximum volume icon. downloaded from flaticon. https://n9.cl/0i566. Accessed: 2023-05-19.

[17] Black Hammer. Sprite for the background of the interface panels. downloaded from unity asset store. https://assetstore.unity.com/packages/2d/gui/fantasy-wooden-gui-free-103811. Accessed: 2023-05-19.

[18] JoelAudio. Sound effect of closing a car hood. downloaded from freesound. https://freesound.org/people/JoelAudio/sounds/135455/. Accessed: 2023-05-19.

[19] Junkohanhero. Font used in the dialogue screens. downloaded from dafont. https://www.dafont.com/es/tox-typewriter.font. Accessed: 2023-05-19.

[20] Anton Koovit. Font used in the options, controls, inventory, manual and volume screens. downloaded from google fonts. https://fonts.google.com/specimen/Arvo. Accessed: 2023-05-19.

[21] Rusty Lake. Cube escape: Paradox. https://n9.cl/sq85s. Accessed: 2023-05-04.

[22] Rusty Lake. Cube escape: Seasons. https://n9.cl/n5s3q. Accessed: 2023-05-04.

[23] Rusty Lake. The past within. https://n9.cl/qn61i. Accessed: 2023-05-04.

[24] Click Minded. Click minded button generator website. https://www.clickminded.com/button-generator/. Accessed: 2023-05-18.

[25] Msx2plus. Sound effect of pouring motor oil. downloaded from freesound. https://freesound.org/people/msx2plus/sounds/678351/. Accessed: 2023-05-19.

[26] United Nations. The sustainable development goals. https://sdgs.un.org/goals. Accessed: 2023-05-04.

[27] Chris Nolet. Quick outline asset from the unity asset store. https://assetstore.unity.com/packages/tools/particles-effects/quick-outline-115488. Accessed: 2023-05-18.

[28] NoraMeld. Soundtrack for the levels. downloaded from freesound. https://freesound.org/people/NoraMeld/sounds/273662/. Accessed: 2023-05-19.

[29] Emoji One. Texture for the down button of the elevator from the second level. downloaded from wikimedia commons. https://n9.cl/ao37m. Accessed: 2023-05-19.

[30] Emoji One. Texture for the up button of the elevator from the second level. downloaded from wikimedia commons. https://n9.cl/24t5z. Accessed: 2023-05-19.

[31] Pixel Perfect. Sprite for the muted volume icon. downloaded from flaticon. https://n9.cl/71re9. Accessed: 2023-05-19.

[32] Pixabay. Sound effect of doing a right action. downloaded from pixabay. https://pixabay.com/es/sound-effects/correct-choice-43861/. Accessed: 2023-05-19.

[33] Pixabay. Sound effect of doing a wrong action. downloaded from pixabay. https://pixabay.com/es/sound-effects/training-program-incorrect1-88736/. Accessed: 2023-05-19.

[34] Pixabay. Sound effect of ending a level. downloaded from pixabay. https://pixabay.com/es/sound-effects/beep-6-96243/. Accessed: 2023-05-19.

[35] Pixabay. Sound effect of ending the game (clock alarm). downloaded from pixabay. https://pixabay.com/es/sound-effects/bedside-clock-alarm-95792/. Accessed: 2023-05-19.

[36] Pixabay. Sound effect of getting an item. downloaded from pixabay. https://pixabay.com/es/sound-effects/item-equip-6904/. Accessed: 2023-05-19.

[37] Pixabay. Sound effect of opening and closing a drawer. downloaded from pixabay. https://pixabay.com/es/sound-effects/drawer-owi-103520/. Accessed: 2023-05-19.

[38] Pixabay. Sound effect of turning on a machine. downloaded from pixabay. https://pixabay.com/es/sound-effects/engine-6000/. Accessed: 2023-05-19.

[39] Pixabay. Soundtrack for the dialogues. downloaded from pixabay. https://pixabay.com/es/sound-effects/vibraphone-loop-1-63039/. Accessed: 2023-05-19.

[40] Pixabay. Soundtrack for the main menu. downloaded from pixabay. https://pixabay.com/es/sound-effects/curious-creep-35588/. Accessed: 2023-05-19.

[41] Dynamic Pixels. Hello neighbor. https://n9.cl/023b3. Accessed: 2023-05-04.

[42] PngAll. Sprite for the green tick icon for the items used in the inventory. downloaded from pngall. https://www.pngall.com/green-tick-png/download/78320. Accessed: 2023-05-19.

[43] Pokémon. Lavender town theme. https://n9.cl/4k97l. Accessed: 2023-05-04.

[44] Andrean Prabowo. Sprite for the cursor. downloaded from flaticon. https://n9.cl/79s80. Accessed: 2023-05-19.

[45] Dimitrios Savva and Dario Barresi. Texture for the floor of the third level. downloaded from poly haven. https://n9.cl/s1ixg. Accessed: 2023-05-19.

[46] Reciclatge Forés S.L. Reciclatge forés s.l. website. https://reciclatge-fores.com/es/empresa-2/. Accessed: 2023-05-04.

[47] StringLabs. Font used in the clue screen of the third level. downloaded from dafont. https://www.dafont.com/es/chalk-board-2.font. Accessed: 2023-05-19.

[48] Modestype Studio. Font used in the main menu screen. downloaded from dafont. https://www.dafont.com/es/orca-chase.font. Accessed: 2023-05-19.

[49] Textures.com. Texture for the floor of the second level. downloaded from textures.com. https://www.textures.com/download/PBR1043/142517. Accessed: 2023-05-19.

[50] Textures.com. Texture for the walls of the second level. downloaded from textures.com. https://www.textures.com/download/PBR0919/140518. Accessed: 2023-05-19.

[51] Boca Tutor. Sprite for the backpack icon. downloaded from iconfinder. https://www.iconfinder.com/iconsets/tutor-icon-set. Accessed: 2023-05-19.

[52] Rob Tuytel. Texture for the shelvings of the first level. downloaded from poly haven. https://polyhaven.com/a/plywood. Accessed: 2023-05-19.

[53] Rob Tuytel. Texture for the tire rack of the second level. downloaded from poly haven. https://n9.cl/qykh0. Accessed: 2023-05-19.

[54] Rob Tuytel. Texture for the walls of the first level. downloaded from poly haven. https://n9.cl/nvo5p. Accessed: 2023-05-19.

[55] Rob Tuytel. Texture for the walls of the third level. downloaded from poly haven. https://n9.cl/lpsyg. Accessed: 2023-05-19.

[56] UNIVERSFIELD. Sound effect of pressing a button. downloaded from pixabay. https://pixabay.com/es/sound-effects/click-button-140881/. Accessed: 2023-05-19.

[57] UNIVERSFIELD. Sound effect of using a key. downloaded from pixabay. https://pixabay.com/es/sound-effects/open-doors-114615/. Accessed: 2023-05-19.

[58] UPNA. Escape room in real life by upna. https://n9.cl/qy4ocb. Accessed: 2023-05-04.

[59] UxWing. Sprite for the "esc" key icon. downloaded from uxwing. https://uxwing.com/esc-button-icon. Accessed: 2023-05-19.

[60] UxWing. Sprite for the i key icon. downloaded from uxwing. https://uxwing.com/i-button-icon. Accessed: 2023-05-19.

[61] UxWing. Sprite for the mouse left click button icon. downloaded from uxwing. https://uxwing.com/mouse-left-click-icon. Accessed: 2023-05-19.

[62] Wikipedia. Bartle taxonomy. https://n9.cl/m2iec. Accessed: 2023-05-04.

[63] Wikipedia. Rusty lake company. https://es.wikipedia.org/wiki/Rusty$_Lake.Accessed:$ $2023-05-04$.

Appendix A

Other considerations

A.1 Bibliography of Downloaded Elements

In this section, all the references to the external sounds, textures, sprites and fonts used for the realization of the work will be referenced.

A.1.1 Sounds

Soundtrack for the levels [28].
Soundtrack for the main menu [40].
Soundtrack for the dialogues [39].
Sound Effect of pressing a button [56].
Sound Effect of opening and closing a drawer [37].
Sound Effect of opening a car hood [11].
Sound Effect of closing a car hood [18].
Sound Effect of using a key [57].
Sound Effect of getting an item [36].
Sound Effect of turning on a machine [38].
Sound Effect of pouring motor oil [25].
Sound Effect of ending a level [34].
Sound Effect of doing a wrong action [33].
Sound Effect of doing a right action [32].
Sound Effect of ending the game (clock alarm) [35].

A.1.2 Textures

Texture for the carton box of the first level [14].
Texture for the tire rack of the second level. [53].
Texture for the walls of the first level. [54].
Texture for the walls of the second level. [50].
Texture for the walls of the third level. [55].
Texture for the floor of the first level. [3].
Texture for the floor of the second level. [49].
Texture for the floor of the third level. [45].
Texture for the up button of the elevator from the second level. [30].
Texture for the down button of the elevator from the second level. [29].
Texture for the shredded plastic of the third level. [13].
Texture for the shelvings of the first level. [52].

A.1.3 Sprites

Sprite for the icon to close the 2D screens [2].
Sprite for the maximum volume icon [16].
Sprite for the muted volume icon [31].
Sprite for the mouse left click button icon [61].
Sprite for the I key icon [60].
Sprite for the "Esc" key icon [59].
Sprite for the backpack icon [51].
Sprite for the gear icon [15].
Sprite for the backpack icon [59].
Sprite for the green tick icon for the items used in the inventory [42].
Sprite for the cursor [44].
Sprite for the background of the interface panels [17].

A.1.4 Fonts

Font used in the main menu screen (see Figure 3.3) [48].
Font used in the options (see Figure 3.4), controls (see Figure 3.5), inventory (see Figure 3.9), manual (see Figure 3.7) and volume (see Figure 3.6) screens [20].
Font used in the clue screen of the third level (see Figure 3.8) [47].
Font used in the dialogue screens (see Figure 3.10) [19].

Appendix

B Source code

```
private void Start()
{
    inventory = FindObjectOfType<Inventory>();
    lvl1Script = FindObjectOfType<Lvl1Controller>();
    lvl2Script = FindObjectOfType<Lvl2Controller>();
    carHood = FindObjectOfType<InteractWithHood>();
    upButton = FindObjectOfType<InteractWithUpButton>();
    downButton = FindObjectOfType<InteractWithDownButton>();
    manualScript = FindObjectOfType<InteractWithManual>();
    battery = FindObjectOfType<InteractWithBattery>();
    motorOil = FindObjectOfType<InteractWithMotorOil>();
    machineDoor = FindObjectOfType<InteractWithMachineDoor>();
    currentItem = "Nada";
}
```

Figure B.1: Start() function - RaycastScript

```
private bool abierto;
private Vector3 posicionInicial;
public AudioSource openDrawer;
public AudioSource closeDrawer;

void Start()
{
    abierto = false;
    posicionInicial = transform.position;
    openDrawer = GameObject.Find("AbrirCajon").GetComponent<AudioSource>();
    closeDrawer = GameObject.Find("CerrarCajon").GetComponent<AudioSource>();
}

public void DrawerInteraction()
{
    StartCoroutine("OpenTheDrawer");
}

private IEnumerator OpenTheDrawer()
{
    if (!abierto)
    {
        openDrawer.Play();
        for (float i = 0f; i <= 1.7f; i+= 0.1f)
        {
            transform.localPosition = new Vector3(transform.localPosition.x - 0.1f,
                transform.localPosition.y,
                transform.localPosition.z);
            yield return new WaitForSeconds(0f);
        }
        abierto = true;
    }
    else
    {
        closeDrawer.Play();
        for (float i = 1.7f; i >= 0f; i -= 0.1f)
        {
            transform.localPosition = new Vector3(transform.localPosition.x + 0.1f,
                transform.localPosition.y,
                transform.localPosition.z);
            yield return new WaitForSeconds(0f);
        }
        abierto = false;
        transform.position = posicionInicial;
    }
}
```

Figure B.2: Drawer’s animation script

```
    GameObject[] interactuableObjects;
    GameObject[] itemObjects;
    public GameObject key;

    private void Start()
    {
        interactuableObjects = GameObject.FindGameObjectsWithTag("Interactuable");
        itemObjects = GameObject.FindGameObjectsWithTag("Item");
        key = GameObject.FindGameObjectWithTag("Key3");
        Invoke("Ilumination", 0.2f); //Desactivar outline para que funcione bien
    }

    void Ilumination()
    {
        foreach (GameObject item in interactuableObjects)
        {
            item.GetComponent<Outline>().enabled = false;
        }

        foreach (GameObject item in itemObjects)
        {
            item.GetComponent<Outline>().enabled = false;
        }

        key.GetComponent<Outline>().enabled = false;
    }
```

Figure B.3: Level3Controller script

```
private void Update()
    {
        //Hacer que lo objetos dejen de estar iluminados al dejar de pasar por encima con el mouse
        if (hightlight != null)
        {
            if (hightlight.gameObject.GetComponent<Outline>() != null)
            {
                hightlight.gameObject.GetComponent<Outline>().enabled = false;
                hightlight = null;
            }
        }

        pointer.enabled = true;
        handPointer.enabled = false;
        RaycastHit hit;
        if(Physics.Raycast(cam.transform.position, cam.transform.forward, out hit, range))
        {
            hightlight = hit.transform;
            if (hit.collider.CompareTag("Interactuable"))
            {
                pointer.enabled = false;
                handPointer.enabled = true;

                hightlight.gameObject.GetComponent<Outline>().enabled = true; //Iluminar objeto

                                if (Input.GetMouseButtonDown(0))
                {
                    Interacting interactingScript = hit.collider.gameObject.GetComponent<Interacting>();
                    InteractWithDrawer drawerScript = hit.collider.gameObject.GetComponent<InteractWithDrawer>();
                    InteractWithWheel wheelScript = hit.collider.gameObject.GetComponent<InteractWithWheel>();
                    ButtonKeypad button = hit.collider.gameObject.GetComponent<ButtonKeypad>();
                    InteractWith(interactingScript, currentItem, currentItemID,
                                  drawerScript, wheelScript, button);
                }
            }
            [...]
        }
        [...]
    }
```

Figure B.4: First fragment of the Update() function - RaycastScript

```
public void InteractWith(Interacting interactingScript, string itemType, int itemID,
                InteractWithDrawer drawerScript, InteractWithWheel wheelScript, ButtonKeypad button)
{
    if (interactingScript.interactingType == "Plastic")
    {
        if (itemType != null && itemType == interactingScript.interactingType)
        {
            milk.SetActive(false);
            currentItem = "Nada";
            inventory.DeleteItem(itemID);
            audioCorrect.Play();
            lvl1Script.wasteRecycled++;
        }
        else audioWrong.Play();
    }

    [...]

    if(interactingScript.interactingType == "Drawer")
    {
        drawerScript.DrawerInteraction();
    }

    [...]

    if (interactingScript.interactingType == "LlaveAllen")
    {
        if (itemType != null && itemType == interactingScript.interactingType)
        {
            allen.SetActive(false);
            currentItem = "Nada";
            inventory.DeleteItem(itemID);
            motorOil.MotorOilInteraction();
            audioCorrect.Play();
            lvl2Script.tasksDone++;
        }
        else audioWrong.Play();
    }

    [...]

    if (interactingScript.interactingType == "Button")
    {
        buttonKeypad.Play();
        button.SendKey();
    }

    [...]

    if (interactingScript.interactingType == "Triturados")
    {
        if (itemType != null && itemType == interactingScript.interactingType)
        {
            cuboTriturados.SetActive(false);
            currentItem = "Nada";
            inventory.DeleteItem(itemID);
            audioCorrect.Play();
            keypadPeletizadora.Play("KeypadPeletizadora");
        }
        else audioWrong.Play();
    }
}
```

Figure B.5: InteractWith() function - RaycastScript

```
private void Update()
{
    [...]
    if(Physics.Raycast(cam.transform.position, cam.transform.forward, out hit, range))
    {
        [...]
        else if (hit.collider.CompareTag("Key1"))
        {
            pointer.enabled = false;
            handPointer.enabled = true;

             hightlight.gameObject.GetComponent<Outline>().enabled = true;

            //Interactuar
            if (Input.GetMouseButtonDown(0))
            {
                StartCoroutine(Level1Complete());
            }
        }
        [...]
    }
    [...]
}

IEnumerator Level1Complete()
{
    cinematic.SetActive(true);
    yield return new WaitForSeconds(5.5f);
    SceneManager.LoadScene("DialogoNivel1");
}
```

Figure B.6: Level1Complete() co-routine - RaycastScript

```
    private void Start()
    {
        raycastScript = FindObjectOfType<RaycastScript>();
        toolManager = GameObject.FindWithTag("ToolManager");

        if (!playersTool)
        {
            int allTools = toolManager.transform.childCount;

            for (int i = 0; i < allTools; i++)
            {
                if(toolManager.transform.GetChild(i).gameObject.GetComponent<Item>().ID == ID)
                {
                    tool = toolManager.transform.GetChild(i).gameObject;
                }
            }
        }
    }

    private void Update()
    {
        if (equipped)
        {
            if (Input.GetKeyDown(KeyCode.X)) //Guardar con la X y avisar al player
            {
                equipped = false;
                raycastScript.currentItem = "Nada";
            }
            if(equipped == false)
            {
                gameObject.SetActive(false);
            }
        }
    }

    public void ItemUsage()
    {
        //Equipar item
        if(raycastScript.currentItem == "Nada") //Equipar si no hay nada equipado
        {
            tool.SetActive(true);
            tool.GetComponent<Item>().equipped = true;
            raycastScript.currentItem = type; //Actualizar el tipo de item que tiene el player equipado
            raycastScript.currentItemID = ID; //Actualizar el ID del item que tiene el player equipado
        }
        else //Cambiar objeto equipado por el nuevo
        {
            int allTools = toolManager.transform.childCount;

            for (int i = 0; i < allTools; i++)
            {
                if (toolManager.transform.GetChild(i).gameObject.GetComponent<Item>().type == raycastScript.currentItem)
                {
                    toolManager.transform.GetChild(i).gameObject.SetActive(false);
                    toolManager.transform.GetChild(i).gameObject.GetComponent<Item>().equipped = false;
                    tool.SetActive(true);
                    tool.GetComponent<Item>().equipped = true;
                    raycastScript.currentItem = type;
                    raycastScript.currentItemID = ID;
                    break;
                }
            }
        }
    }
```

Figure B.7: Item script

```
    void Start()
    {
        [...]
        allSlots = slotHolder.transform.childCount;
        slot = new GameObject[allSlots];
        for (int i = 0; i < allSlots; i++)
        {
            slot[i] = slotHolder.transform.GetChild(i).gameObject;
            if (slot[i].GetComponent<Slot>().item == null) slot[i].GetComponent<Slot>().empty = true;
        }
    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.I)) inventoryEnabled = !inventoryEnabled;
        if (inventoryEnabled)
        {
            inventory.SetActive(true);
            playerScript.speedCam = 0; //Que no se mueva el fondo
            Cursor.visible = true; //Hacer que el cursor esté activo
            Cursor.lockState = CursorLockMode.None;
        }
        else if(!inventoryEnabled && !options.panelActivo && !manualScript.manualActivo)
        {
            inventory.SetActive(false);
            playerScript.speedCam = 300;
            Cursor.visible = false; //Hacer que el cursor no esté activo
            Cursor.lockState = CursorLockMode.Locked;
        }
    }

    public void AddItem(GameObject itemObject, int itemID, string itemType, Sprite itemIcon)
    {
        for (int i = 0; i < allSlots; i++)
        {
            if (slot[i].GetComponent<Slot>().empty)
            {
                itemObject.GetComponent<Item>().pickedUp = true;
                slot[i].GetComponent<Slot>().item = itemObject;
                slot[i].GetComponent<Slot>().ID = itemID;
                slot[i].GetComponent<Slot>().type = itemType;
                slot[i].GetComponent<Slot>().icon = itemIcon;
                itemObject.transform.parent = slot[i].transform;
                itemObject.SetActive(false);
                slot[i].GetComponent<Slot>().UpdateSlot();
                slot[i].GetComponent<Slot>().empty = false;
                return; //Evitar que item se añada en todos los slots vacíos
            }
        }
    }

    public void DeleteItem(int itemID)
    {
        for (int i = 0; i < allSlots; i++)
        {
            if(slot[i].GetComponent<Slot>().ID == itemID)
            {
                slot[i].GetComponent<Slot>().empty = true;
                slot[i].GetComponent<Slot>().UpdateEmptySlot();
            }
        }
    }
```

Figure B.8: Inventory Script

```
private void Start()
{
    slotIconGameObject = transform.GetChild(0);
    inventory = FindObjectOfType<Inventory>();
}

public void UpdateSlot()
{
    slotIconGameObject.GetComponent<Image>().sprite = icon;
}

public void UseItem()
{
    item.GetComponent<Item>().ItemUsage();
}

public void OnPointerClick(PointerEventData pointerEventData)
{
    if (!empty)
    {
        UseItem();
        inventory.inventoryEnabled = false;
    }
}

public void UpdateEmptySlot()
{
    slotIconGameObject.GetComponent<Image>().sprite = tick;
}
```

Figure B.9: Slot Script

```
private void Start()
{
    cam = transform.GetChild(0).GetComponent<Transform>(); //Transform del primer hijo de player
    control = GetComponent<CharacterController>();
}

private void Update()
{
    //Cámara
    float mouseX = Input.GetAxis("Mouse␣X");
    float mouseY = Input.GetAxis("Mouse␣Y");

    transform.Rotate(new Vector3(0, mouseX, 0) ∗ speedCam ∗ Time.deltaTime);
    camRotation -= mouseY ∗ speedCam ∗ Time.deltaTime;
    camRotation = Mathf.Clamp(camRotation, -90, 90); //Bloquear que rote más de lo deseado
    cam.localRotation = Quaternion.Euler(new Vector3(camRotation, 0, 0));

    //Movimiento
    float moveX = Input.GetAxis("Horizontal");
    float moveZ = Input.GetAxis("Vertical");

    Vector3 movement = (transform.right ∗ moveX + transform.forward ∗ moveZ)
                    ∗ playerSpeed ∗ Time.deltaTime; //En eje X y Z del mundo
    control.Move(movement);
    control.Move(new Vector3(0, gravityMove, 0) ∗ Time.deltaTime);

    if (!control.isGrounded) //Si no toca el suelo, que caiga
    {
        gravityMove += gravityForce;
    }
    else //Si toca el suelo, resetea aceleración de la gravedad
    {
        gravityMove = 0f;
    }
}
```

Figure B.10: Player Script

```
public string number;

public void SendKey()
{
    transform.GetComponentInParent<KeypadScript>().PasswordEntry(number);
}
```

Figure B.11: ButtonKeypad Script

```
    private void Start()
    {
        passwordText.text = "";
    }

    public void PasswordEntry(string number)
    {
        if (number == "Clear")
        {
            Clear();
            return;
        }
        else if (number == "Enter")
        {
            Enter();
            return;
        }

        int length = passwordText.text.ToString().Length;
        if (length < passwordLimit)
        {
            passwordText.text = passwordText.text + number;
        }
    }

    public void Clear()
    {
        passwordText.text = "";
        passwordText.color = Color.white;
    }

    private void Enter()
    {
        if (passwordText.text == password)
        {
            correctSound.Play();

            passwordText.color = Color.green;
            StartCoroutine(CallMachine());
            StartCoroutine(WaitAndClear());
        }
        else
        {
            wrongSound.Play();

            passwordText.color = Color.red;
            StartCoroutine(WaitAndClear());
        }
    }

    IEnumerator WaitAndClear()
    {
        yield return new WaitForSeconds(0.75f);
        Clear();
    }

    IEnumerator CallMachine()
    {
        yield return new WaitForSeconds(0.75f);
        machine.StartAnimation();
    }
```

Figure B.12: Keypad Script

```
void Start()
{
    Cursor.visible = true; //Hacer que el cursor esté activo
    Cursor.lockState = CursorLockMode.None;
    musicPlayer = FindObjectOfType<MusicPlayer>();
    dialogueText.text = string.Empty;
    StartDialogue();
}

void StartDialogue()
{
    index = 0;
    StartCoroutine(TypeLine());
}

IEnumerator TypeLine() //Escribir los caracteres 1 a 1
{
    foreach (char c in lines[index].ToCharArray())
    {
        dialogueText.text += c;
        yield return new WaitForSeconds(textSpeed);
    }
}

void NextLine()
{
    if(index < lines.Length - 1) //Al terminar la linea, se vacía y empieza la siguiente
    {
        index++;
        dialogueText.text = string.Empty;
        StartCoroutine(TypeLine());
    }
    else //Cuando termina todo el texto, pasa al siguiente nivel
    {
        if(nivel == 0)
        {
            SceneManager.LoadScene("Nivel1");
        }
        [...]
        if (nivel == 3)
        {
            musicPlayer.pauseAudio();
            clockAlarm.Play();
            StartCoroutine(BackToMainMenu());
        }
    }
}

IEnumerator BackToMainMenu()
{
    yield return new WaitForSeconds(4);
    SceneManager.LoadScene("MainMenu");
}

public void ContinueDialogue()
{
    if (dialogueText.text == lines[index]) NextLine();
    else
    {
        StopAllCoroutines();
        dialogueText.text = lines[index];
    }
}
```

Figure B.13: Dialogue Script

```
    public AudioSource audio;
    public Slider volumeSlider;
    private float musicVolume = 1f;

    private void Start()
    {
        audio.Play();
        musicVolume = PlayerPrefs.GetFloat("volume", 1f);
        audio.volume = musicVolume;
        volumeSlider.value = musicVolume;
    }

    private void Update()
    {
        musicVolume = volumeSlider.value;
        audio.volume = musicVolume;
        PlayerPrefs.SetFloat("volume", musicVolume);
    }

    public void volumeUpdater(float volume)
    {
        musicVolume = volume;
    }

    public void pauseAudio()
    {
        audio.Stop();
    }
```

Figure B.14: MusicPlayer Script