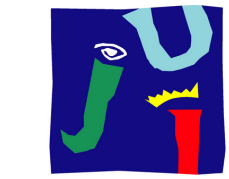

UNIVERSITAT JAUME I

Development of an interactive experience to highlight the gambling addiction caused by betting houses and his consequences in the affected ones' lives

Jorge Nieto Morales

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

July 5, 2020

Supervised by: Emilio Sáez Soro

To my mom, my dad and my sister for supporting me in my adventure

Acknowledgments

First of all, I would like to thank my Final Degree Work supervisor, Emilio Sáez Soro, for helping me find the idea of this project and for his always welcomed suggestions.

Thanks to Marta, Laura and Dani for testing my game and give me some essential tips to improve it.

To Pau for me helping to decide how the game should look like.

To Jon, for making easier for me to start this report.

And to Maria, for being with me in the most stressful moments and supporting me.

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

Abstract

This document represents the Final Degree Work report of Jorge Nieto Morales in Video Game Design and Development.

This work consists of a videogame which shows the daily life of an affected by the gambling addiction caused by sports bettings in betting houses or another kind of gambling. The videogame put the player in the skin of a man with a gambling addiction who tries to get over this situation and continue with his life while he is trying to avoid different elements which could make him return to betting habits. Every intern battle between the main character and these temptations is represented by different puzzles or minigames that the player has to pass. This experience is accompanied by conversations with NPC's who offer their opinion on the gambling houses and related topics.

Contents

CHAPTER 1

INTRODUCTION

Contents

This chapter is an explanation about which were the motivations that took to project's idea, which were the objectives initially fixed and how the idea started to be developed [1].

1.1 Work Motivation

Nowadays, gambling addiction is a worrying problem in society, especially on people who live in working-class neighbourhoods. In recent years there has been an increase in the number of open betting houses, and in consequence, this problem is starting to be more noticeable among people. It can affect people's lives, mentally and physically, and affect their families too.

So it seemed necessary to talk about this problem, or at least, treat this topic through the media. And a good way of doing so is with videogames. It is important to ensure that people know that this kinds of problems are affecting to a lot of people, and for that reason, is necessary to let everyone know this problem and make them understand that gambling addiction is dangerous. In that way, people will be warned and will know the consequences of starting to play this kind of games before they try them.

1.2 Objectives

- Develop a videogame with different minigames on it that gives variety to the experience of the players, allowing them to experience different mechanics and gameplay with a challenging aspect that keeps them playing without getting bored.
- Create four levels, with two sections inside each one: one based on exploration, where the player moves through familiar spaces, as a street, a house, an office or a bar; and another one where the player plays a minigame inspired by a classic arcade game, as *Frogger*, *Simon says* and others.
- Design minigames that include elements that are a reference to gambling houses and casinos (as cards, coins, slot machines, etc).
- Representing the problem with gambling addiction in a way that can be interesting and entertaining for the players, using conversations with NPCs that give their opinion on this topic or make reference to gambling, and through the minigames' elements mentioned before, showing them as something that can harm the player.

1.3 Environment and Initial State

The idea of the work came up in the meetings with the project supervisor. The importance of the work addressing an important topic for the author of the work was stressed, as this would facilitate its realization. The supervisor suggested that the job could be about an issue that affected people nowadays, and it was thus concluded that the topic of work would be gambling addiction.

Once the topic was chosen, it was needed to think about how it would develop. A game that would mix exploration mechanics with more entertaining and interactive mechanics was proposed. That way it would be managed to give it the touch of seriousness that requires a problem as serious as addiction, but at the same time, the entertainment function of a video game would be fulfilled. So the idea arose to divide each level into two parts: exploration and minigame.

The work would be done with the Unity game engine, and its programming would be entirely done by the author of the work. For the visual aspect, both assets created by the author and free assets included in the Unity asset store would be used.

Finally, meetings would be held with the supervisor each time the author of the work fully completes the creation of a level. That way, the supervisor could look at the project and suggest changes to improve it.

CHAPTER 2

Planning and resources evaluation

Contents

This chapter shows the planning that has been followed to complete the project and the resources used to accomplish that purpose.

2.1 Planning

The next lines show the tasks that have been done to build all the project. Not all the tasks were done one by one (finishing one before starting the next), but some were made alternating between them. This section also includes a Gantt chart showing the same tasks in a more visual way (see Figure 2.1)

- **Information gathering and understanding (10 hours):** look for articles or blogs to understand how people with gambling addiction feel, which are the main difficulties in their daily lives and learn about the situation in Spain with this problem. Besides, search for references for the game, as which minigames that could be an inspiration and how those could be adapted.
- **Game concept design (10 hours):** the initial design of all the levels of the videogame, including how will be the map, where the NPCs will be located, which dialogue will be assigned to each NPC, which minigame will be assigned at each level, the interface and the HUD, the mechanics and other details.

- **Game design (190 hours):**
  - **Level 1 (65 hours):** the construction and programming of the level 1, including the first scripts of movement of the player, the option of talking with NPCs, the movement of the camera and the change of perspective trough the level, create the canvas, etc. Moreover, this task included the creation of the first minigame, based on *Frogger*.
  - **Level 2 (45 hours):** the construction and programming of the level 2, including the second minigame, based on the classic game of matching cards.
  - **Level 3 (45 hours):** the construction and programming of the level 3, including the third minigame, based on *Simon!*.
  - **Level 3 (35 hours):** the construction and programming of the level 4, including the fourth minigame, based on *Aerogun field*.
- **Create and search assets (20 hours):** use programs to create 3D models or search for them in the asset store of Unity or the Internet.
- **Documentation (70 hours):** write the Final Degree Work report, the presentation or other necessary documents.

To give a little explanation about the Gantt chart of the tasks, there are two periods of time where the project was paused. The first one took place in December, and it lasted until the last days of January. The reason was that in December there was a deadline to make the technical proposal of the project. After finish it, I had to study for my final exams, so I had to stop the project for a while. Then, in February there was another deadline, but this time was about the GDD. A few days later, I started with my university practices, so I needed two weeks to adapt to my new routine. After that, I continued with my final degree work.

In the Gantt chart, it can be seen that the levels were made in order. For each level, first, it was needed to find the resources that would be used in that level (3D models, inspirations, canvas...), and the next step was to create and develop the level.

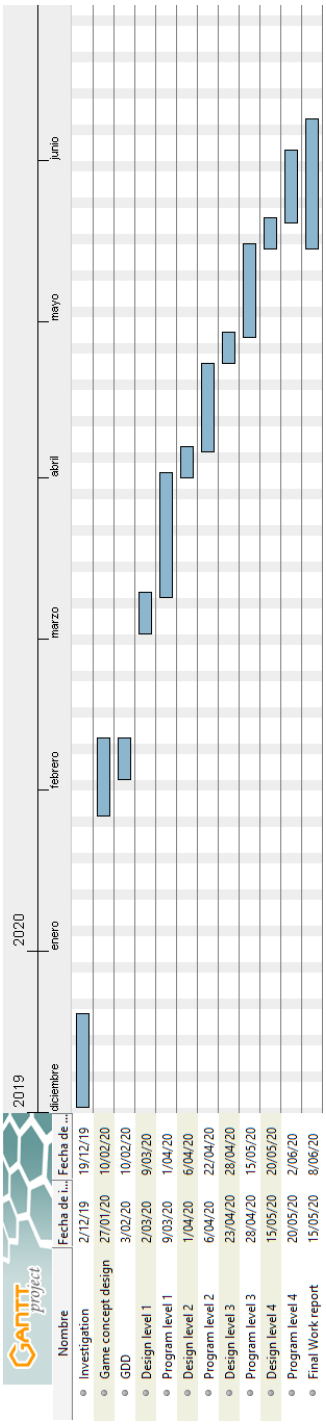

Figure 2.1: Gantt chart of the tasks (made with Gantt Project)

2.2 Resource Evaluation

The resources used for this project are:

- A Lenovo Ideapad 100 PC, with I5, 4GB of RAM and 500 GB of hard disk. Cost: 400 €.
- Unity 2019.2.6 version, used to create the project and work on it [2]. Cost: free.
- Visual Studio 2019, used to program the project [3]. In this case, Visual Studio is attached to Unity, in order to make easier its function. Cost: free.
- 3DSMax 2019, used to create some 3D models to include them on the project [4]. Cost: free, with student license.
- GitHub Desktop, a tool used to create a repository and use it to upload the project and keep a copy of it [5]. Cost: free.
- Gantt Project, a desktop application used to create the Gantt chart [6]. Cost: free.
- Cool Text Graphics Generator, a web page used to create texts for the game [7]. Cost: free.
- Mixamo, a web page used to obtain 3D animations [8]. Cost: free.
- TeXnicCenter, a tool used to edit the LaTex document [9]. Cost: free.
- Unity Asset Store, the store of assets of Unity used to obtain 3D models and other resources [10]. Cost: depends on the asset.
- Grammarly, a google chrome extension used to correct texts in English [11]. Coste: free (standard).
- Visual Paradigm Online, a web page used that allows to create different kinds of diagrams, like a case use diagram [12]. Coste: free.

CHAPTER 3

System Analysis and Design

Contents

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

3.1 Requirement Analysis

Firstly, let's clarify how the game works, and then the requirements will be clearer. The first thing that is seen when the game starts is the main menu. It is composed by four buttons. The first one is *Start game*, and it simply takes the player to the first level. The second one is *Select level*, which allows the player to choose which level he wants to play. However, to select a specific level, the player has to have played it before. The third one is *Options*, which gives the option to turn on or turn off the sound. And the final one is *Exit game*, which allows the player to quit the game.

As it has been said before, each level has an exploration section and a minigame section. The exploration section has the same mechanics in all levels, so it's better to start explaining that part. The player can move through the level with *W, A, S* and *D* keys. The player can move in all directions, but to find the path in the first level he can use the *Q* key, which will activate the indications to reach the objective during a few seconds (arrows in the ground that aim to the target). This action is only available at

the first level, due to that the other levels are smaller and easier to complete. Finally, to interact with NPCs and talk with them, the player has to press the *Space* key. Then, a dialogue will appear on the screen, and to skip it, the player only has to press again the space key.

Now, let's talk about each minigame. The first minigame is based on *Frogger*, where the player is a frog that has to cross a road and a river to reach the target. In this case, the player controls the main character, but instead of walk, the player jumps in all directions with *W, A, S* and *D* keys. The player has to avoid that the coins that move from one side to another touch him, and jump on the cards to avoid falling and reach the four targets.

The second minigame is based on the classic game of matching cards, where there are face down cards, and the player has to choose two to see if they are the same card. The mechanic is the same in this game, and to choose a card the player has to click over that card with the left click of the mouse.

The third minigame is based on *Simon!*, which is a classic game where four buttons with different colours (green, red, yellow and blue) appear in the screen, and the player has to repeat a pattern of colours. In this project, those buttons represent buttons of a slot machine, and the player has to repeat the pattern of colours using the left click of the mouse to press the buttons.

The last minigame is based on *Aerogun field*, an arcade minigame where the player has to shoot to different items that are falling following a rail. In this project, the player doesn't shoot, and he only has to move between for rails to touch the items and avoid them touching the ground. To move from one rail to another, the player can use the *A* key to move to the left, or the *D* key to move to the right.

To come back to the main menu, the player will be able to pause the game in any of the levels (when the game allows it). If the player press the *Esc* key, a pause menu will appear with two options: *Resume*, that allows the player returning to the game, or *Main menu*, that allows the player returning to the main menu.

3.1.1 Functional Requirements

Once the previous explanation is clear, it is easy to identify which are the functional requirements:

- **R1:** the player can start the game.
- **R2:** the player can select level.
- **R3:** the player can mute or unmute the game.
- **R4:** the player can quit the game.
- **R5:** the player can move through the level.
- **R6:** the player can check the indications.

- **R7:** the player can talk with other characters.
- **R8:** the player can jump.
- **R9:** the player can select a card.
- **R10:** the player can press a button.
- **R11:** the player can move between rails.
- **R12:** the player can pause the game.
- **R13:** the player can return to the main menu.
- **R14:** the system will be able to generate items in the scene.
- **R15:** the system will be able to generate random patterns of colors.

3.1.2 Non-functional Requirements

Non-functional requirements impose conditions on the design or implementation. In this project, the non-functional requirements are:

- **R16:** the game will be playable on PC.
- **R17:** the system will be documented by a manual in pdf format that will describe how to manage and use it.
- **R18:** the game will use low poly models.
- **R19:** the exploration sections will be realistic.
- **R20:** the minigames will be unrealistic and will have more fictional elements.
- **R21:** the elements of the game will be related with gambling houses and casinos.
- **R22:** the mechanics will be easy to learn.
- **R23:** the UI will have a bigger role in the minigames.

3.2 System Design

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

Requirement:	R1
Actor:	Player
Description:	The player starts the game pressing by the button *Start game*
Preconditions:	1. The player must be in the main menu
Normal sequence:	1. The player press the button *Start game* 2. The system loads the first level
Alternative sequence:	None

Table 3.1: Case of use «Start game»

Requirement:	R2
Actor:	Player
Description:	The player selects a level by pressing the button *Select level*
Preconditions:	1. The player must be in the main menu 2. The player has to have played the level he wants to play
Normal sequence:	1. The player press the button *Select level* 2. The system takes the player to a new screen where he can choose the level 3. The player press the button that corresponds with the level he wants to play 4. The system loads that level
Alternative sequence:	4.1. The system doesn't load the level, because the player hasn't played it before

Table 3.2: Case of use «Select level»

Requirement:	R3
Actor:	Player
Description:	The player mute or unmute the sound going to the options screen
Preconditions:	1. The player must be in the main menu
Normal sequence:	1. The player press the button *Options* 2. The system takes the player to the options screen 3. The player press the button to mute or the button to unmute 4. The system mute or unmute the game, depending on which button the player has pressed
Alternative sequence:	3.1. If the player is trying to mute the game when it is already muted, the system will do nothing. The same with unmute

Table 3.3: Case of use «Control sound»

Requirement:	R4
Actor:	Player
Description:	The player exits from the game
Preconditions:	1. The player must be in the main menu
Normal sequence:	1. The player press the button *Quit game* 2. The system quit the game
Alternative sequence:	None

Table 3.4: Case of use «Quit game»

Requirement:	R5
Actor:	Player
Description:	The player moves trough the level
Preconditions:	1. The player must be in the exploration section of a level 2. The title introduction has to have ended 3. The instructions panel has to have been passed by the player 4. The player has to be out of the pause menu 5. The player has to be out of a dialogue with an NPC
Normal sequence:	1. The player press *W, A, S* or *D* keys 2. The character moves in the direction assigned to that button
Alternative sequence:	2.1 The character can't move because he is colliding with something

Table 3.5: Case of use «Move»

Requirement:	R6
Actor:	Player
Description:	The player consults for the path to the target
Preconditions:	1. The player must be at level 1 2. The title introduction has to have ended 3. The instructions panel has to have been passed by the player 4. The player has to be out of the pause menu 5. The player has to be out of a dialogue with an NPC
Normal sequence:	1. The player press the Q keys 2. The system shows the arrows that aim to the target during a few seconds.
Alternative sequence:	None

Table 3.6: Case of use «Check indications»

Requirement:	R7
Actor:	Player
Description:	The player consults for the path to the target
Preconditions:	1. The player must be in the exploration section of a level 2. The title introduction has to have ended 3. The instructions panel has to have been passed by the player 4. The player has to be out of the pause menu 5. The player has to be near an NPC with an indicator of dialogue over his head (an exclamation)
Normal sequence:	1. The player press the *Space* key 2. A dialogue panel appears, and the NPC starts an animation 3. The player continues pressing the *Space* key to continue the dialogue until it ends
Alternative sequence:	None

Table 3.7: Case of use «Talk with a character»

Requirement:	R8
Actor:	Player
Description:	The player moves jumping in one direction
Preconditions:	1. The player must be at the minigame of the level 1 2. The countdown has to have ended 3. The instructions panel has to have been passed by the player 4. The player has to be out of the pause menu
Normal sequence:	1. The player press *W, A, S* or *D* keys 2. The character jumps in the direction assigned to that button
Alternative sequence:	2.1 The character collides with a coin and turns to the initial position 2.2 The character falls to the void and turns to the initial position 2.3 If with that movement the player reaches a target, the player wins the round and turns to the initial position

Table 3.8: Case of use «Jump»

Requirement:	R9
Actor:	Player
Description:	The player selects a card to turn it out
Preconditions:	1. The player must be at the minigame of the level 2 2. The countdown has to have ended 3. The instructions panel has to have been passed by the player 4. The player has to be out of the pause menu 5. The countdown of three seconds that appears when the player fails has to have ended
Normal sequence:	1. The player clicks over a card 2. The card turns around
Alternative sequence:	2.1 If the card is already face up, it doesn't do it 2.2 If the card is a card that the player chooses to compare it with another one that has been turned before this one, then the system checks if they are the same cards. If they are, they remain face up and the player can click over other cards. 2.3 If not, the cards turn around after three seconds, and the player can't click over other cards during that time 2.4 If these are the two last cards that remain face down, the player wins the round

Table 3.9: Case of use «Select card»

Requirement:	R10
Actor:	Player
Description:	The player presses a button to repeat the colours pattern
Preconditions:	1. The player must be at the minigame of the level 3 2. The countdown has to have ended 3. The instructions panel has to have been passed by the player 4. The player has to be out of the pause menu 5. It has to be the player's turn
Normal sequence:	1. The player clicks over a button 2. The button brights and makes his animation
Alternative sequence:	2.1 If that button is the correct button in the colour pattern, the player can continue in his turn 2.2 If that button is not the correct one, the player has to restart the round and lost his turn 2.3 If that is the last button of the pattern, the player wins the round

Table 3.10: Case of use «Press a button»

Requirement:	R11
Actor:	Player
Description:	The player moves from one rail to another
Preconditions:	1. The player must be at the minigame of the level 4 2. The countdown has to have ended 3. The instructions panel has to have been passed by the player 4. The player has to be out of the pause menu
Normal sequence:	1. The player press *A* or *D* keys 2. The character moves to the next rail that is in that direction
Alternative sequence:	2.1 If the player tries to go in one direction where there are no more rails, the character doesn't move

Table 3.11: Case of use «Move between rails»

Requirement:	R12
Actor:	Player
Description:	The player moves from one rail to another
Preconditions:	1. The player must be playing a level 2. The title introduction has to have ended 3. The instructions panel has to have been passed by the player
Normal sequence:	1. The player press the *Esc* keys 2. The game stops and the pause menu appears
Alternative sequence:	None

Table 3.12: Case of use «Pause the game»

Requirement:	R13
Actor:	Player
Description:	The player returns to the main menu from a level
Preconditions:	1. The player must be at the pause menu
Normal sequence:	1. The player press the button *Back menu* 2. The system takes the player to the main menu
Alternative sequence:	None

Table 3.13: Case of use «Return to the main menu»

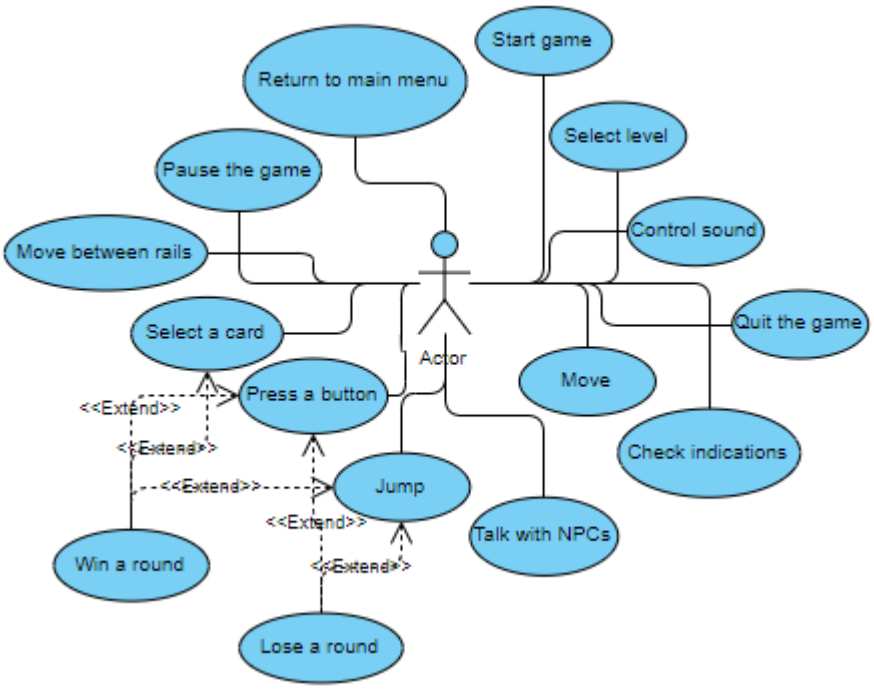

Figure 3.1: Case use diagram (made with Virtual Paradigm)

The functional requirements are included in the case of use diagram. However, it can be seen that the main actions of the three first minigames have an extend than can take the player to win or losing the round (jump, press a button and select a card). That is because, for example, if the player jumps and falls into the void in the first minigame he loses the round, but if he jumps and reaches one of the targets, he wins the round. However, the main mechanic of the fourth minigame (move between rails) doesn't have these extends. That is because in this minigame winning or losing is not a direct consequence of this action. If the timer ends, the player wins, and if the lifebar drops to zero, the player loses.

3.3 System Architecture

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

- The operating system Windows 7 SP1+, at least.
- A CPU with x86 or x64 architecture with SSE2 instruction set support, at least.
- At least a graphic card (GPU) with DX10.
- A keyboard and a mouse or a touch panel.

The requirements have been taken from Unity documentation.

3.4 Interface Design

The role of the GUI depends on the section of the level that the player is playing. Firstly, let's show how the UI looks and which is his function on the exploration sections.

At the start of each level, it can be seen a panel with instructions that gives player a bit of context on the plot and indications of what the target is (see Figure 3.2). This instruction panel appears at the beginning of every minigame, to explain how it works. The other element of interface design that appears at the exploration sections is the panel of dialogue, which is activated when the player talks with an NPC or when the main character talks with himself at the end of some levels (see Figure 3.3).

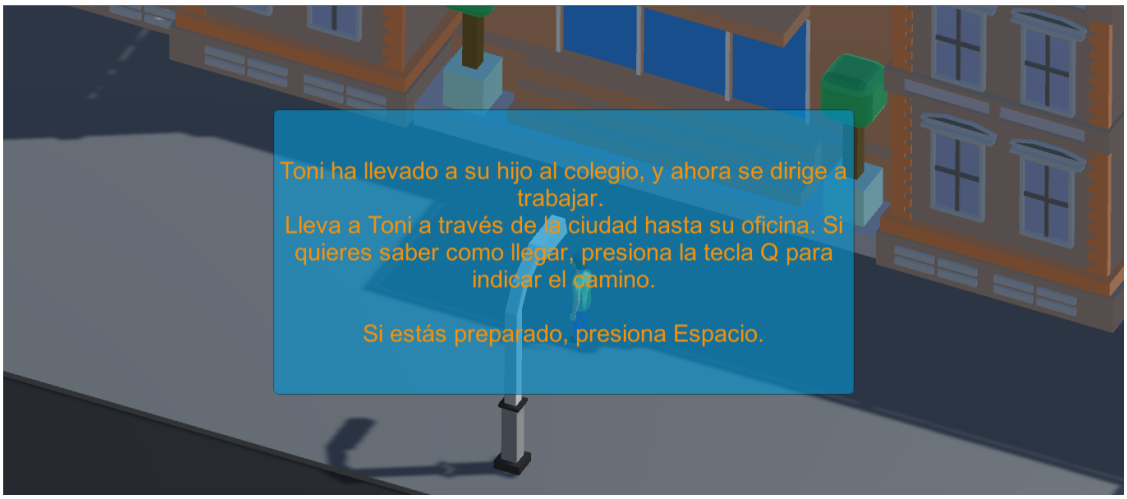

Figure 3.2: The instructions panel at the start of the level

Figure 3.3: The dialogue panel

However, the first level includes an element that allows the player to see which is the path that he has to follow. When the player presses the Q key, some arrows aiming to the target appears on the ground (see Figure 3.4).

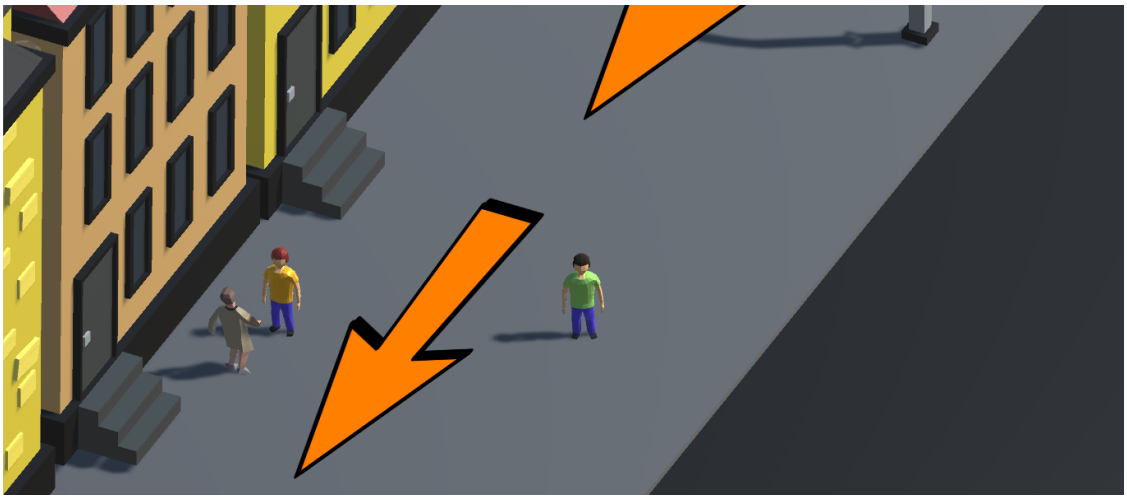

Figure 3.4: The arrows aim to the target

In the minigames, the GUI has an important role. It gives information to the player about the things that happen on it. There are some elements that appear in all the minigames. At the start of a minigame, there is a countdown of three seconds until the game starts (see Figure 3.5). This way, the player can prepare before the minigame starts.

Figure 3.5: The initial countdown

When the player fails, a text appears on the screen, which indicates to him that he has to try again (see Figure 3.6). In case that the player wins a round, this text will congratulate him and tell the player which round he has won (see Figure 3.7).

Some minigames have specific elements of GUI for them. The second minigame, for example, has a timer that indicates to the player how much time he has left to finish the round (see Figure 3.8).

The third minigame has an accountant that informs the player about in which iteration of the pattern of colour he is. In that way, the player can know how many buttons he has to press in that iteration, avoiding being lost (see Figure 3.9).

Finally, in the fourth minigame there is lifebar that shows the life of the player. This bar drops if a bad item reaches the ground, and increases if the player catch a good item. Besides, this minigame includes a timer to indicates how many time is left until the minigame ends (see Figure 3.10).

Figure 3.6: Text that appears when the player fails

Figure 3.7: Text that appears when the player wins a round

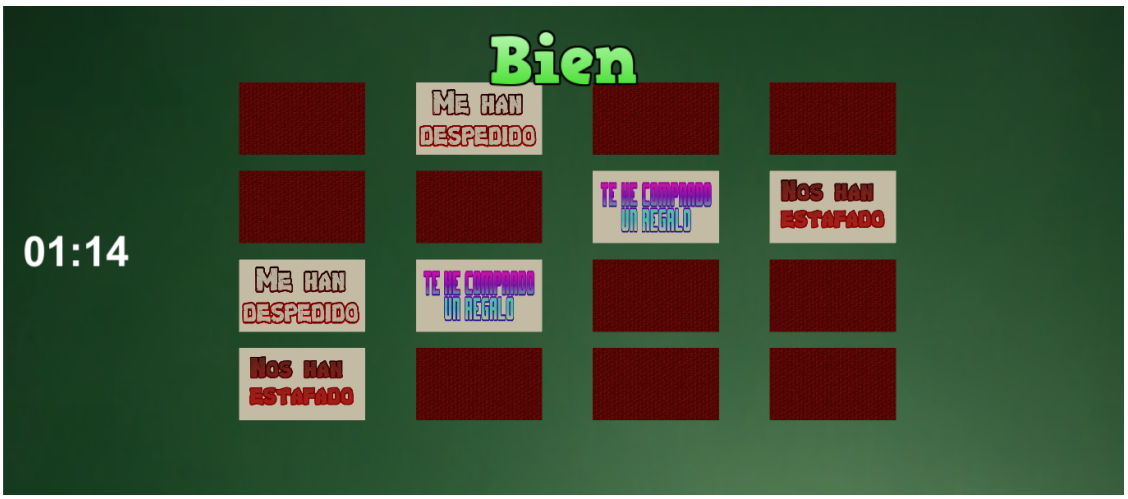

Figure 3.8: Timer in the second minigame

Figure 3.9: Account to indicate the iteration on the round

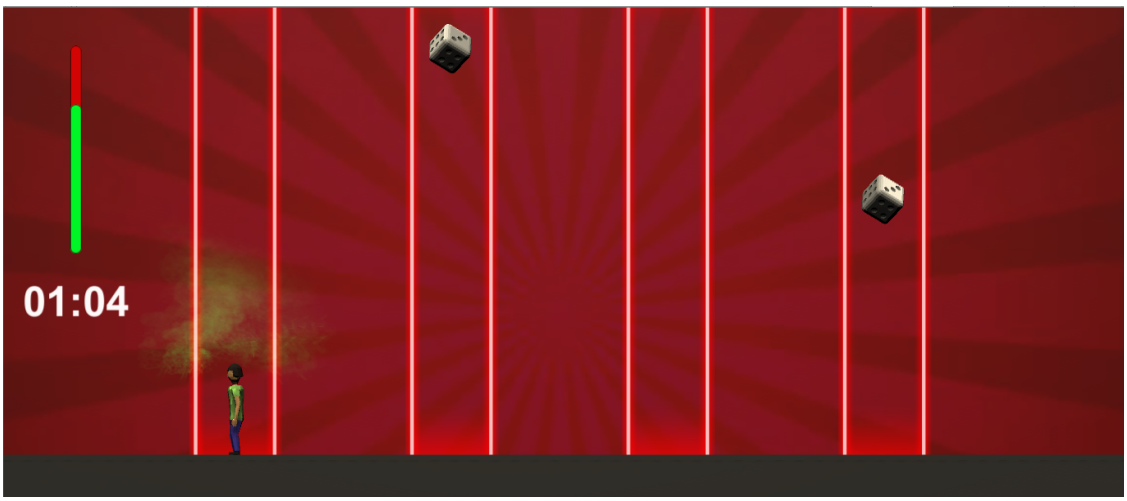

Figure 3.10: Lifebar that show the life of the player

Chapter 4

Work Development and Results

Contents

This chapter is an explanation of how the project has been developed since his start until the end of it. It also includes an assessment of the results and how some original ideas were changed during this process due to changes in the opinion of the author, recommendations of the supervisor or other reasons.

4.1 Work Development

The work development is going to be explained in chronological order. This is the best way for this project because it can be divided into four levels, and those levels were created in order. So the chronological order follows the order of the tasks too. But first, is better to talk about what the game is about.

The main character is Toni, an adult man that lives in Spain. He is married to Mónica, and they have a son, Daniel. However, Toni has a secret that no one knows. He suffers a gambling addiction. Five years ago, Toni started to go to a gambling house with a job's partner. He went there searching for a new way to disconnect and earn some money, but with time he lost the control. Five years later, Toni has decided to get out of that situation. He has been keeping his secret to all the people he cares about, and now he has committed to return to his normal life. However, things are not going to be easy for Toni in a world where gambling games are everywhere.

Let's start for level 1. This was a level that took more time than the rest of the levels. The reason is that it was the start of the project, so the player's movement and the settings of the camera had to be created from scratch. This two elements would be reused on the rest of the levels, reducing the amount of work on them. Besides, the first level has the biggest scene, so his creation took more time than the rest.

The first task was to program the movement of the player. The player moves the character with the *W, A, S* and *D* keys, and the movement works regarding the camera. For the movement, it was used a character controller. The character controller is a component in Unity that makes easier for the programmer to control the main character from script. The 3D model for Toni, the main character, was taken from *Itch.io* [13].

Once the movement was made, it was the turn of the camera. In the conceptual development of the game, it was decided that the view would be isometric. To achieve that, the camera was changed from perspective to orthographic. An orthographic projection is a different way to represent three-dimensional objects. While the perspective projection tries to simulate the human's view, the orthographic projection basically avoids that the elements of the scene change their scale depending on how far they are from the camera. This kind of projection can be seen in games as *'The stillness of the wind'* (see Figure 4.1) and *'Xcom 2'* (see Figure 4.2).

Figure 4.1: The stillness of the wind, by Lambic Studios (2019)

Then, the next thing to do was create a dialogue system. In every level, there are at least a few NPCs with whom the player can talk. Each NPC has his own dialogue, so it was necessary to find a way to store all the different dialogues and access them from a single script attached to all the NPCs. After a deeply search to find out a way to achieve this, the best way looked to be the use of a list of string arrays. Let's explain quickly how this works. Every NPC has an ID, which is unique. Then, in an scriptable object (which is a script whose variables can be used from one scene to another) a list of string arrays is stored. Each array of that list stores the dialogue of one NPC. So when the player interacts with an NPC, the NPC's script access to that list, and searches the array that is in the position ID (for example, for the NPC with ID 3, the array will be in the position 3 of the list).

Figure 4.2: Xcom 2, by Firaxis Games (2016)

After creating the dialogue system, there was a need to have something that could help the player to differentiate between the NPCs with a dialogue and the others NPCs. The chosen option was to identify this NPCs with an exclamation that appears over the NPC's head when the player is in his area (see Figure 4.3). That way, the player can know if he can talk with the NPC or not. When the player is near enough him, if he presses the *Space* key, a panel with the dialogue appears on the screen. The phrases that appear in the panel are taken from the array corresponding to that NPC (as explained before). The exclamation is animated, and once the player starts the conversation, it disappears. When the conversation ends, the exclamation reappears.

Figure 4.3: Exclamation that indicates that the player can talk with this NPC

With these things, the main mechanics of the exploration section of the four levels are completed. Now is the turn of the creation of the levels. It was a clear objective that each level should represents an aspect on the main character's life. For that reason, the first level takes place at the streets of a city. Toni has taken his son to the school, and now he has to go to work. The player will have to control Toni and follow the indications to arrive to his job. In his play, the player can talk with other characters. The exploration section ends when Toni sees the new gambling house that the NPCs have talked about. His starts to speak with himself, but his intern battle takes the player to the first minigame.

The 3D models used for this level were taken from the Unity Asset Store, in specific from the packages *European buildings* [14], *Simple City Lite* [15] and *Viuletti City low poly* [16]. To create the streets, it was taken as a guide the initial design established during the game concept design (see Figure 4.4). We can see it on the result (see Figure 4.5).

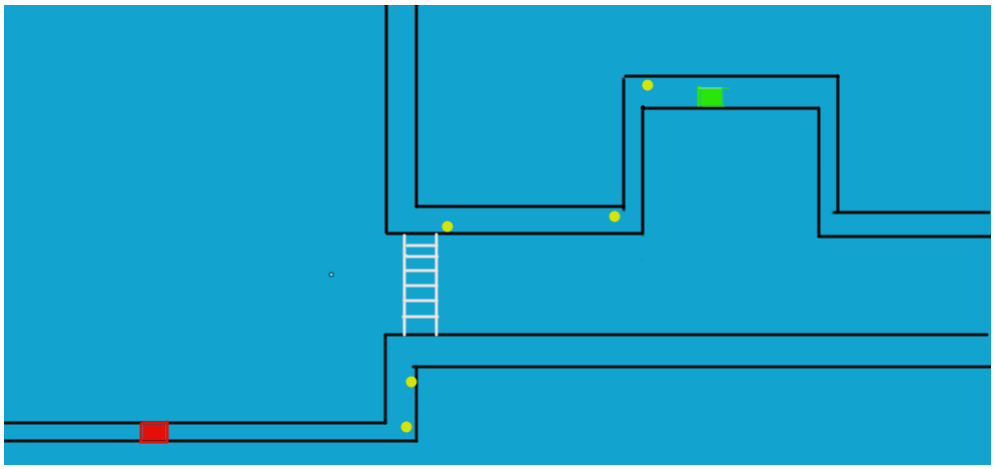

Figure 4.4: The concept design of the first level. The green square represents the start position of the player. The yellow circles represent the positions of the NPCs with dialogue option. The white marks represent the crosswalk. And the red square represents the position where the exploration section ends.

However, once the level was designed, a problem with the camera appeared. There were some sections of the scene where the player couldn't be seen because he was behind a building. To solve this, the only solution was to move the camera to catch a new perspective, so the player can see the main character all the time. An easy solution would be to create three cameras, with different positions and rotations, and turn on or turn off them depending on the position of the player. But finally, the decision was to use a unique camera that will change his position and rotation once the player would arrive at a place where is necessary to change the perspective. That way, the change of perspective would be more attractive and soft. To achieve this, it was necessary not only change the position and the rotation of the camera with a soft movement but change

Figure 4.5: The final appearance of level 1

the value of the distance between the camera and the main character. This task was a bit complicated and took some time, but the result was worth it. There are three points in the level where perspective changes, and we can see in figures 4.6 and 4.7 an example. When the player collides with the colliders placed in these points, perspective changes. The code can be seen in appendix A.

Figure 4.6: The initial perspective of the camera

Figure 4.7: The change of perspective regarding the previous image

To finish the exploration section of level 1 and give it a bit of life, there was added a generator of vehicles and a crosswalk. When the player enters into the crosswalk, the vehicles stop near it, as in real life (see Figure 4.8). Once the player exits from the crosswalk, the vehicles continue. However, the vehicles that are already passing over the crosswalk don't stop (as in real life again). The 3D models of the vehicles were taken from the Unity asset *Simple Vehicle Pack* [17].

Figure 4.8: The player on the crosswalk and the vehicles respecting the traffic laws

As a recommendation of the supervisor, there was added an indications option to know which is the path to reach the target. This was necessary because it was a bit

difficult to find the correct way due to the scale of the scene. These indications are arrows that appear on the ground, which can be seen in figure 3.4.

Now let's talk about the minigame of level 1. In the game conceptual design, it was taken as an inspiration the arcade game *Frogger*. The first *Frogger* is a 1981's game where the player controls a frog. The frog moves jumping in four directions, and the player has to cross a road and a river. In the road, he has to avoid getting hit by a car, while in the river he has to jump over trunks to reach the other shore and avoid falling into the water. However, more than from the 1981's game, the inspiration came from his remake for PS1 that was released in 1997 (see Figure 4.9). This version had 3D graphics, unlike the original game, which only had 2D graphics. Nevertheless, this new version works with the same mechanics as the original.

Figure 4.9: *Frogger*, by SCE Cambridge Studio (1997)

The minigame of level 1 has essentially the same mechanics. The main character stills been Toni, and the player has to control him to reach the four targets. When the player presses the *W, A, S* or *D* keys, Toni jumps in the direction assigned to the key he has pressed. The scene is divided into two areas, as the original game. In the first one, the player has to avoid getting hit by poker coins that go from left to right and from right to left. The second one would be the corresponding to the river in the original game, and here the player has to jump over poker cards that are moving from left to right and vice-versa to avoid falling into the void (see Figure 4.11). Once the player passes this part, he will find four targets, and if he enters in one of them, the game will take the player to his start position, and he will have to repeat the same route to reach another target. When the player reaches the four targets, the level ends. However, if the player is getting hit by a coin or he falls into the void, he will return to the start position and will have to repeat the route (see Figure 4.10).

About the GUI, this minigame only uses it to tell the player how many targets are

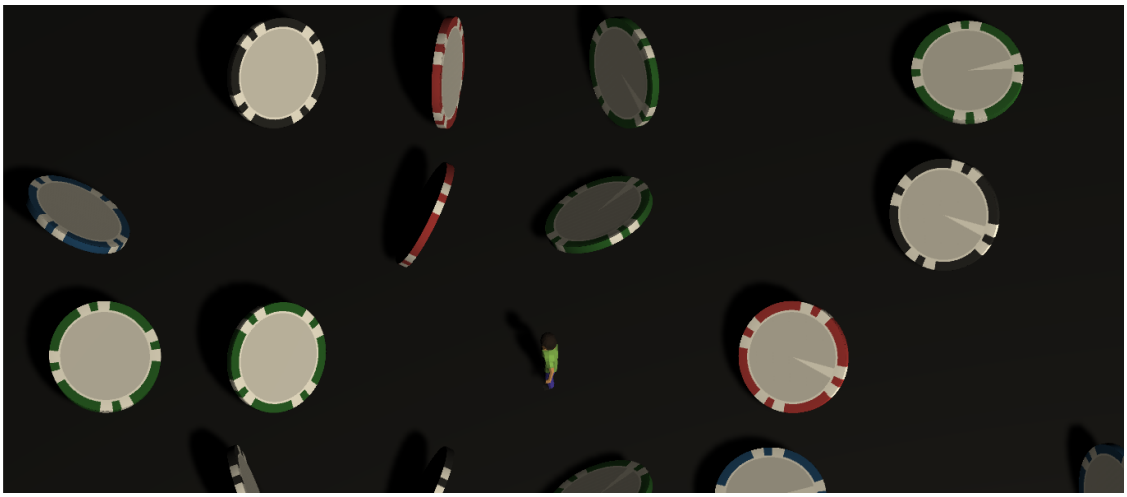

Figure 4.10: The player has to avoid getting hit by the coins

Figure 4.11: The player must jump over the cards to avoid falling into the void

left and when he fails (as it was seen in figures 3.7 and 3.6). And that is all about the first level.

Level 2 takes place at Toni's house. Mónica, who is Toni's wife, wants to talk with him. She doesn't know Toni's gambling addiction. However, she has noticed something strange on their bank account. To talk with her, Toni has to go to the kitchen. On his way, Toni can talk with Daniel, his son. In this dialogue, Daniel asks Toni to give him some money to pay for a lootbox on his game. This kind of products works similarly to the gambling machines, where the prize depends on chance. Talking about this was a clear objective since the first steps of the game.

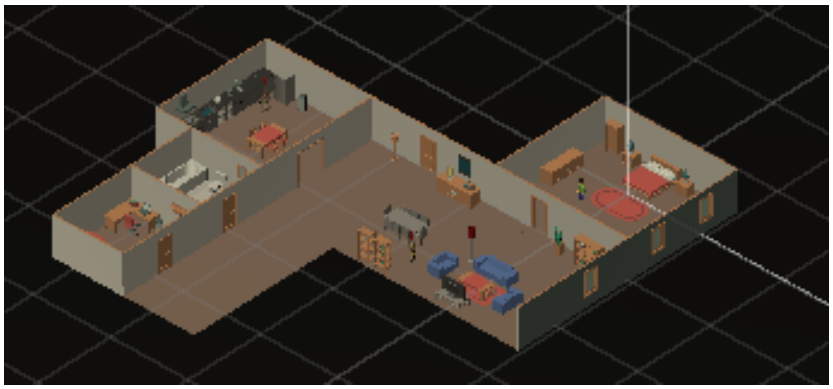

Figure 4.12: The final appearance of level 2

As it can be seen in figure 4.12, level 2 is less big than the first one. So it took less time to finish his creation. There was also no need to add the change of perspective or the indications option of the previous level, due to this level works perfectly without those things. The models used for this scene were taken from *Furniture Kit* [18] and *Home Assets* [19], two assets downloaded from Kenney and Unity Assets Store.

After all the level was complete, it was time to start with the second minigame. For this minigame, it was taken as inspiration the classic card game of matching cards. It is a memory challenge game where an even number of cards appears on the table, with all cards face down, and every card has an identical pair between the others. The player has to select two and check if they are the same card. The game ends when the player finds all the pairs of cards.

As this game has a lot of variants, there was more freedom on his development. For this version, the decision was adding a challenging feeling. To do that, the player would have limited time to find all the pairs. If the timer arrives to zero, the player loses the game. To increase the difficulty from level to level, the minigame was designed to have three rounds. The start state of the timer would be different in each round. In the first one, the player has ninety seconds to find all the pairs, in the second one seventy-five seconds and the last one sixty seconds.

However, in this state, the game kept being too easy. A solution was to penalize the

Figure 4.13: Example of a matching cards game (anonymous)

player for matching the wrong cards. If the player had total freedom to select cards, he could easily find all the pairs in less than thirty seconds. But with the penalization of three seconds, the player has to think which card he is going to select, or he will lose time. During those three seconds, the player can't select any other card, and once that time ends, the cards return to be face down and the player can try again.

To give a little help to the player, it was added a new mechanic into the game. At the start of each round, the cards will be face up, and the player can try to remember where the cards are. After five seconds, the cards are turned, and the round starts (the player can select cards and the timer starts).

About the technical part (programming), the cards are saved on a list as they are created. Then, their positions are simply changed for the position of another card in the list. Once the round or the timer ends, the list is emptied, and the new cards generated are saved here instead. That way, every round will be different from the rest. The code can be seen in appendix A

As the idea of the project is that all elements in the game have to be related with gambling addiction and his consequences, the cards represent excuses that Toni uses to hide his secret and the problem with the bank account. Every excuse has a different typography, to make it easier for the player to identify the pairs of cards at the beginning of the round. Besides, the cards have an animation to rotate when they are selected by the player and after the three seconds of penalization.

Finally, the GUI in this minigame is used in a similar way as in the previous one. It informs to the player when he wins or loses a round, indicating in the first case which round he has won and how many rounds are left, and when the player matches correctly (see Figure 3.8) or badly a pair of cards. Moreover, it shows the player's remaining time to find all the pairs. It is important to remark that a suggestion of the supervisor was to add a countdown on the screen when the penalization is applied, to be sure that the

player understands that he can't select another card until that countdown ends.

It has to be mentioned that initially, the second level didn't have a minigame. This minigame was intended to appear at the beginning of level 4 when Toni talks with an old friend of the family. This way, level 2 would not have a minigame (it would only be a transition level with dialogue), and level 4 would have two minigames, one at the beginning and another one at the end to finish the level. However, after a meeting with the supervisor, it seemed much better to include this minigame in the second level, appearing when Toni talks with Mónica and preventing the player from getting bored. For this objective, it was necessary to make a little change on Mónica's background story, due to that at the first steps of the game she knew Toni's problem, and with this change in the distribution of the game, it was necessary that she didn't know anything.

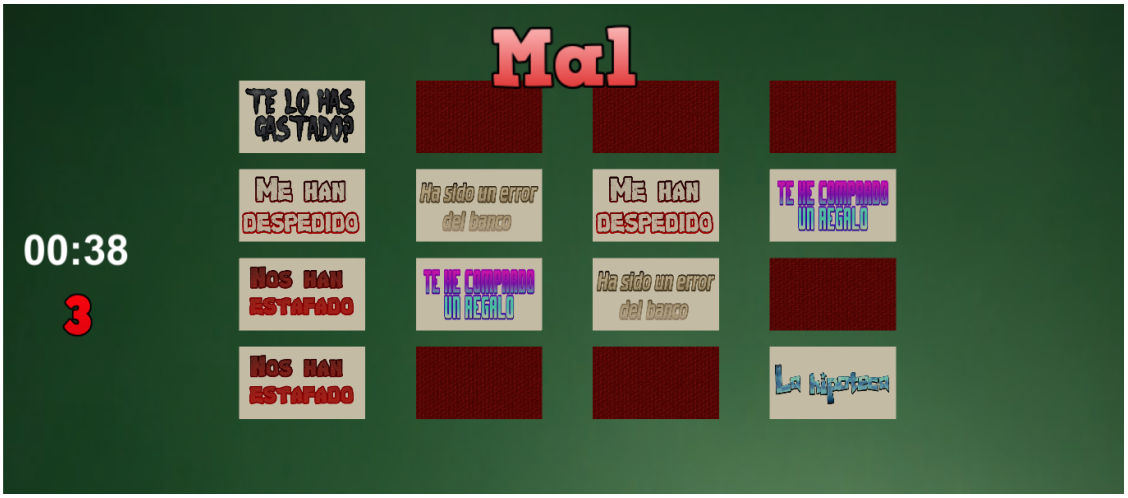

Figure 4.14: Screenshot from the second minigame, with the countdown of penalization

Level 3 takes place at Toni's office. Toni has been working in that office for fifteen years, and he has made very close friends there. However, Iker, one of his partners, is the one who took Toni for the first time to a gambling house, and they used to go together to those places. The level starts when Toni takes a break to go for a coffee to the coffee pot, which is in front of the boss' door. On his way, the player can talk with a partner, which has a little dialogue, and with Rosa. Rosa is Toni's best friend in his job, and they have been friends since he started to work there. However, Rosa is having problems with his son, now that he has started to gambling on the Internet, replacing his old drugs addiction for this one. Toni and Rosa will talk about that if the player interacts with Rosa. Once the player reaches the coffee pot, he will see Iker, and dialogue will start. But this time, the dialogue will be activated automatically (the player collides with a collider and it activates). In this dialogue, Iker tries to convince Toni of going with him to a gambling house and betting for a football match. When the

dialogue ends, the third minigame starts, representing that Toni is having an internal fight to resist the temptation.

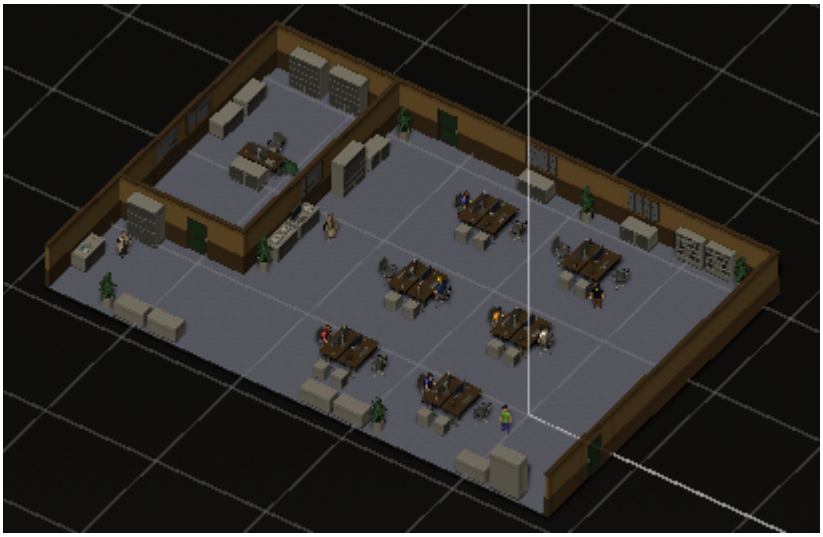

Figure 4.15: The final appearance of level 3

Before talking about the third minigame, there are a few things to comment about the exploration section. The 3D models were taken from the Unity assets *Modern Office Interior* [20], and initially, the level was bigger. However, after some testings made by the supervisor and other people, the decision was to put fewer desks in the office and make it smaller. That way, due to the number of things that the player can do in this scene, the scale of the level would be perfect. Besides, there were added some workers doing his work on their desks. That way, the level would look more alive.

The third minigame suffered a big change regarding of his initial concept. The idea was to repeat a pattern of keys of the keyboard to win the game. Every key would have a different meaning, composing all the keys of the pattern a phrase that Toni would say to Iker to avoid the temptation. However, this idea had a lot of problems. There would be a lot of options for the pattern, due to that all the keys of the keyboard could be used, and that could generate a pattern too difficult to remember. The other main problem is that all the minigames must include elements related to gambling houses and casinos, and in this case, there were poor chances to achieve it.

So after thinking about it, the conclusion was that a good solution could be making the game more similar to the fount of inspiration: *Simon!*. This game wasn't originally a videogame, but it was a board game from the year 1978. However, it has had some adaptations in videogames. The game has four buttons, each one with a different colour (in the classic version, the colours are yellow, blue, green and red), and each round the game establishes a different pattern of colours. First, it illuminates one colour, and the player has to repeat it. In the next iteration, the game adds a new colour to the pattern, and the player has to repeat the same colours int the same order. This happens

successively, adding colours in each iteration until the player makes a mistake when repeating the pattern. The round starts again with a new pattern.

Figure 4.16: *Simon!* board game, created by Ralph Baer and Howard J. Morrison (1978)

In the project's version, there are a few differences from the original. To relate the game with gambling houses and casino's elements, the four buttons are part of a slot machine. The colours are the same ones of the original game, but each button has an image on him, which are the elements that appear on the screen of a slot machine. In the mechanics, there are a few changes too. The minigame is divided into three rounds. The difficulty increases when the player wins a round, showing the pattern of colours more quickly than in the previous round, which makes it more difficult to remember it for the player. Each round has six iterations, which means that each pattern is composed by six colours, and in each iteration the player has to repeat the colours that the game has shown, adding a new colour in each one.

If the player wins the three rounds, the minigame ends. However, if he makes a mistake repeating the pattern, he has to repeat the round from the beginning. To give feedback to the player, when he presses a button, the button moves as a button pressed, and it brights with his colour assigned (see Figure 4.17). To make the button shine, it has been used the spotlight component. Each button has his own spotlight, and this spotlight is modified to shine with the colour of the button assigned. The problem is that in his default settings the spotlight illuminates all the buttons, and that is not the effect searched. To avoid this, the solution was to use the culling mask option. This allows specifying which components of the scene will be affected by the spotlight, depending on their tag. In this case, the components would be the slot machine and the button assigned to the spotlight, and the excluded ones would be the rest of the buttons (see Figure 4.18).

This minigame represents the battle of Toni against Iker's words. If the player can follow the rhythm of the pattern, then Toni successes ignoring the temptation that Iker represents.

Finally, the GUI in this minigame has two roles. For one hand, it indicates to the player when he has won the round (indicating which round he has won and how many rounds are left) and when he has lost the round (when he makes a mistake). For the

Figure 4.17: A screenshot from the third minigame

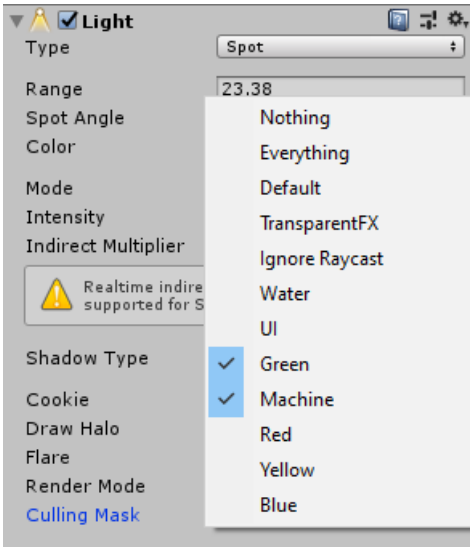

Figure 4.18: The light component of the spot light, where it can be seen which tags are assigned in the culling mask option (in this the green button and the slot machine)

other hand, there is a counter which shows the player in what round is he in. These components where showed in Chapter 3 (see Figure 3.9).

Level 4 is the last one. It was created in the last steps of the project. This level takes place in a bar where Toni has to meet with two friends. At first, Toni can't move through all the level. There is an invisible collider to avoid Toni from passing to the bathroom area. First, he has to speak with his friends. In this conversation, it will be cleared that Toni has economical problems as a consequence of his gambling addiction because Toni borrowed to one of them some money. After the conversation, Toni will need to go to the bathroom, and his friends will tell him how to arrive. At that moment the invisible collider will disappear, and Toni will be able to explore a new area. Before or after speaking with his friends, Toni can talk with a stranger. This stranger will recognize Toni from a gambling house, and Toni will try to escape from that situation. The exploration section ends when Toni is on his way to the bathroom and he sees a gambling machine. Then the final minigame starts.

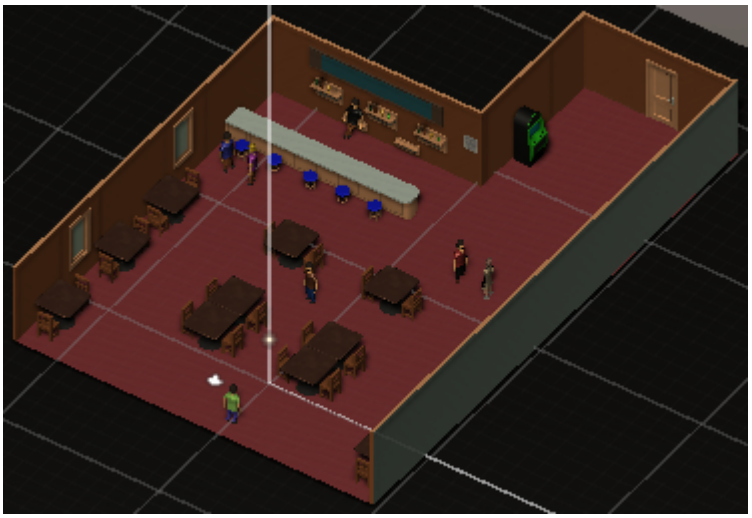

Figure 4.19: The final appearance of level 4

The 3D models of level 4 were taken from the Unity assets *Bar props* [21], and as it has been said before, initially this level was composed by two minigames, the minigame of the level 2 and another. However, a change on the game design left this level with only one minigame, like the rest of the levels.

The final minigame is based on *Aerogun field*, an arcade game from the year 1987 and developed by Tronica (see Figure 4.20). In this game, the player has to move between four rails and fire against warplanes that are trying to reach the base of the player by falling through that rails. If a warplane reaches the player's base he will lose one life. Once the player loses three lives the game ends.

For the project's version of this minigame, there were made a few changes from the

Figure 4.20: Aerogun Field, created by Tronica (1987)

original. In the original, the game didn't end until the player lost, but in this version, there is a victory condition. The player has a lifebar, and this lifebar decreases when an item touches the ground. If the lifebar drops to zero, the minigame restarts. To win the game, the player has to survive during two minutes, trying to catch all the items that are falling before they reach the ground (in the original game the player had to shoot against them, but in this one, the player has to collide with them). However, the player can increase the lifebar catching the health item, which will appear when the player is losing health.

As in the original game, there are four rails. The player can move between them with the *A* key (left) and the *D* key (right), but also with the left arrow key and the right arrow key. To make the game more challenging, the difficulty increases with time. Every thirty seconds three variables change and these are the speed of the items (which fall faster), the time between items (which is reduced) and the probability of a healthy item to appear (in this case, this variable increases, giving a little help to the player in the most difficult sections).

To achieve this, the different scripts of this minigame take variables from the script which manages the minigame. In this script (which controls the timer) the time is checked, changing the speed, the time between items and the probability of the health item depending on the time. Then, the other scripts (the script of the items and the script to spawn them) access to these variables. Besides, health items only appear when the lifebar is not full, so for this, the script that spawns items has to access to the value of the slider that represents the lifebar, and which is managed in the manager script. The code can be seen in appendix A.

Initially, the increase in the difficulty only affected the time between the items.

However, in the testing phase, the supervisor and other people said that the minigame was too easy. So changing the speed of the items would increase the difficulty, but maybe too much. The conclusion was to increase the probability of the healthy items to appear too.

The GUI ins this minigame is used to inform the player when he loses or when he wins the minigame. When the player loses, the instructions panel appears, and when he presses the *Space* key, the countdown of three seconds starts and then the minigame restarts. This happens in the rest of the minigames too. The GUI also includes the lifebar and the timer. Also, to give feedback to the player, there was added a particles system. When an item reaches the ground a red smoke appears in that position, and when the player catches a health item, a green smoke appears (see Figure 4.21). The particles were created using the Unity Asset *Particle Ribbon* [22].

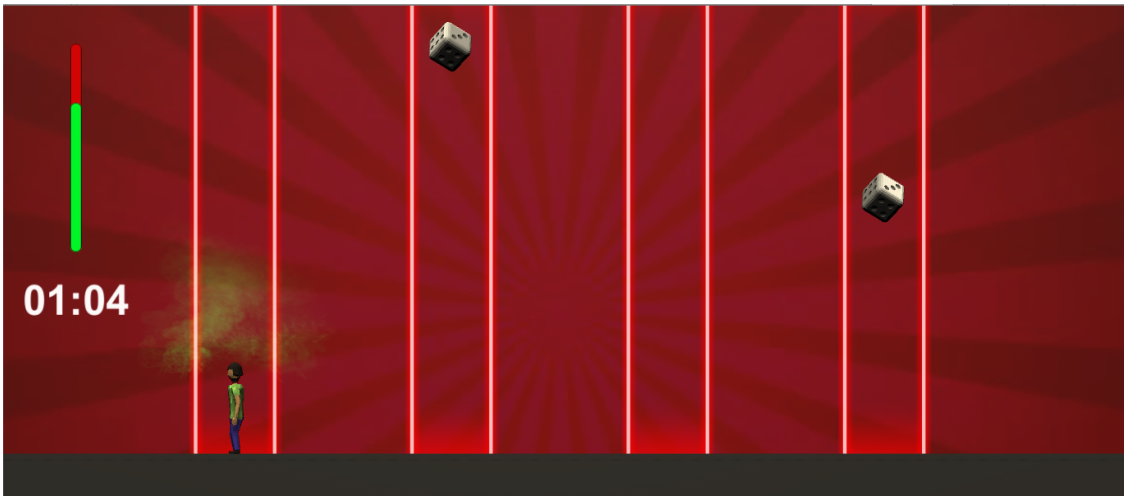

Figure 4.21: A screenshot from the fouth minigame, with the green smoke that appears when the player catches a health item. There are also dices falling, which are objects that the player has to catch to avoid them reach the ground.

All the game music has been taken from the web page Patrick de Arteaga [23].

4.2 Results

The initial objectives can be divided into two parts. For one hand, the main objective was to design and develop four levels, mixing exploration and minigames without losing the entertaining aspect of the game. As it has been explained in the previous lines, the game has suffered some changes regarding the initial concept, like moving one of the minigames of the level 4 to the level 2 and considerable changes in the third minigame. However, these changes have been necessary to achieve this objective, making the game entertaining in all levels and giving to the player a variety of experiences with very different minigames between them.

The mechanics of each minigame are different from the rest of the minigames, so the initial objective of giving variety to the experience of the player has been accomplished. Besides, the four levels were created, as it was set in the conceptual design of the game.

On the other hand, another objective to accomplish was representing the problem with gambling addiction in a way that could be interesting for the players. This representation can be seen in both parts: exploration and minigames. In the exploration section, the world where the game takes place shows that everyone has his opinion about the topic. In the conversations with the NPCs, the player can see how everyone is affected by this, and what forms the gambling can adapt, as the lootboxes of the videogames, the people that change one addiction for another, the people who suffer gambling addiction thinking that they control the situation... In the minigames, there are included elements that evoke to gambling houses and casinos, and which have a function in the mechanics (objects that are used in poker and slot machines have a negative meaning). Then, this game is clearly about gambling addiction, which was one of the objectives.

You can access to the repository of the project with this link:

`https://github.com/JDex18/The-winner-loses-it-all`

And also access to the builds of this project for Windows and Mac with this link:

`https://drive.google.com/drive/folders/1zPJmagGMNp7TwvN0KNkV0MYbvuuRVxN_?usp=sharing`

CHAPTER 5

Conclusions and Future Work

Contents

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

5.1 Conclusions

During the degree, we have made some videogames with different characteristics for different subjects. Most of them were made in group, making his development easier and not having all the responsibility. However, this project is different. This is my final degree work, and it is the closest experience of developing a videogame that I have ever had.

A few months ago, before I started with this project, I would have considered this work too difficult to achieve with my skills. However, my final degree work has coincided with my university practices. I have learned a lot about Unity in those practices, and my experience there gave me enough confidence to develop this work. But at the same time, the time that I dedicated to this project helped me with my practices.

So in conclusion, these months has given me the confidence that I need to consider myself capable of doing what I want the most: making videogames. This project has been very important to me, and I think that the amount of time that I have dedicated to it has made me understand how important is for someone to work in something that he cares about. In my case, it has been gambling addiction. I am so concerned about this

problem, and I consider that it needs more visibility. So mixing that with videogames is what I can do for what I consider one of the most important things right now.

5.2 Future work

The game that I have achieved is the game that I had in mind when I started to think about it, with only a few necessary changes. However, during the development, some ideas came to my mind. They were good ideas, but maybe too difficult to implement before the deadline. If I would continue with this project, I would probably add some kind of difficulty selection. This project already has a changing difficulty, but maybe I could find the way to adding this modification and giving to the player a motivation to replay the game.

Another thing that I would consider to add could be more NPCs to talk with. I like the conversations that you can have with the characters of the game because their different opinions and experiences make look the game more alive. And another thing to add related to the dialogues could be different dialogue options. That way, Toni could have multiple responses, and create a great number of possibilities in the conversations. This could even make the game more immersive, and as a consequence, maybe the player could become more concerned with gambling addiction.

Bibliography

[1] Germán Fabregat Llueca. Guía para la redacción de las memorias. http://mermaja.act.uji.es/itis/IS31/guia_memoria.pdf. Accessed: 2020-06-07.

[2] Unity Technologies. Unity download archive. https://unity3d.com/es/get-unity/download. Accessed: 2020-06-07.

[3] Microsoft. Visual studio download archive. https://visualstudio.microsoft.com/es/downloads/. Accessed: 2020-06-07.

[4] Autodesk. 3ds max download archive. https://www.autodesk.com/education/free-software/3ds-max. Accessed: 2020-06-07.

[5] GitHub. Github desktop download archive. https://desktop.github.com/. Accessed: 2020-06-07.

[6] Gantt Project. Ganttproject download archive. https://www.ganttproject.biz/download. Accessed: 2020-06-07.

[7] Cool Text Graphics Generator. Cool text graphics generator web page. https://cooltext.com/. Accessed: 2020-06-07.

[8] Adobe Inc. Mixamo web page. https://www.mixamo.com/. Accessed: 2020-06-07.

[9] TeXnicCenter. Texniccenter download archive. https://www.texniccenter.org/download/. Accessed: 2020-06-07.

[10] Unity Asset Store. Unity asset store web page. https://assetstore.unity.com/. Accessed: 2020-06-07.

[11] Grammarly. Grammarly web page and download archive. https://app.grammarly.com/. Accessed: 2020-06-07.

[12] Visual Paradigm International. Visual paradigm online web page. https://online.visual-paradigm.com/es/. Accessed: 2020-06-07.

[13] Trex Games. Low poly character from itch.io. https://trexgames.itch.io/low-poly-character?download. Accessed: 2020-06-07.

[14] karboosx. Low poly european city pack asset from the unity asset store. https://assetstore.unity.com/packages/3d/environments/urban/low-poly-european-city-pack-71042. Accessed: 2020-06-07.

[15] Codery Games. Simple city lite asset from the unity asset store. https://assetstore.unity.com/packages/3d/environments/urban/simple-city-lite-133329. Accessed: 2020-06-07.

[16] Viuletti. Low poly city asset from the unity asset store. https://assetstore.unity.com/packages/3d/environments/urban/low-poly-city-from-viuletti-132536. Accessed: 2020-06-07.

[17] Myxer Man. Simple cars pack from unity asset store. https://assetstore.unity.com/packages/3d/vehicles/land/simple-cars-pack-97669. Accessed: 2020-06-07.

[18] Kenney. Furniture kit asset from kenney. https://www.kenney.nl/assets/furniture-kit. Accessed: 2020-06-07.

[19] Mohelm97. Simple home stuff asset from the unity asset store. https://assetstore.unity.com/packages/3d/simple-home-stuff-69129. Accessed: 2020-06-07.

[20] Asset Store Originals. Snaps prototype | office asset from the unity asset store. https://assetstore.unity.com/packages/3d/environments/snaps-prototype-office-137490. Accessed: 2020-06-07.

[21] SimpleModelsForMe. Bar props asset from the unity asset store. https://assetstore.unity.com/packages/3d/props/barprops-137130. Accessed: 2020-06-07.

[22] Moonflower Carnivore. Particle ribbon asset from the unity asset store. https://assetstore.unity.com/packages/vfx/particles/spells/particle-ribbon-42866. Accessed: 2020-06-07.

[23] Patrick de Artega. Royalty free music. https://patrickdearteaga.com/es/musica-libre-derechos-gratis/childs-nightmare/. Accessed: 2020-06-29.

Appendix A

Source code

CameraFollow script for level 1

```
1
2  using System.Collections;
3  using System.Collections.Generic;
4  using UnityEngine;
5
6  public class CameraFollow : MonoBehaviour
7  {
8
9      public Transform player;
10     private Vector3 cameraOffset;
11     private Vector3 cameraOffset1;
12     private Vector3 cameraOffset2;
13     private Vector3 cameraOffset3;
14
15     public Transform cam2;
16     public Transform cam3;
17     public Transform object2;
18     public Transform object3;
19
20     public Transform[] views;
21     private Transform currentView;
22     private float transitionSpeed;
23     public bool cameraTransition;
24     public static bool cambio1;
25     public static bool cambio2;
26     public static bool cambio3;
27
28     public GameObject trigger1;
29     public GameObject trigger2;
```

```
30      public GameObject trigger3;
31
32
33
34      [Range(0.01f, 1.0f)]
35      public float smoothFactor;//para suavizar el movimiento de la cámara
36      // Start is called before the first frame update
37      void Start()
38      {
39          cameraOffset1 = transform.position - player.position;
40          transitionSpeed = 2f;
41          cameraTransition = false;
42          currentView = views[0];
43
44          //cam2 = new Vector3(39.44464f, -47.32929f, -83.11659f);
45          cameraOffset2 = cam2.position - object2.position;
46          cameraOffset3 = cam3.position - object3.position;
47          cameraOffset = cameraOffset1;
48          cambio1 = false;
49          cambio2 = false;
50          cambio3 = false;
51      }
52
53      // Update is called once per frame
54      private void Update()
55      {
56          if (cambio1 || cambio2 || cambio3)
57          {
58              cambioCamara();
59          }
60
61
62      }
63
64      void LateUpdate()
65      {
66
67          if (!cameraTransition)
68          {
69              Vector3 newPos = cameraOffset + player.position;
70
71              transform.position = Vector3.Slerp(transform.position, newPos, smoothFactor);
72          }
73
74          else
75          {
76              transform.position = Vector3.Lerp(transform.position, currentView.position, Time.deltaTime * transitionSp
77              //transform.rotation = currentView.rotation;
78              transform.rotation = Quaternion.Lerp(transform.rotation, currentView.rotation, Time.deltaTime * 2);
79
80              if (Vector3.Distance(transform.position, currentView.position) <= 5)
81              {
82                  cameraTransition = false;
83              }
```

```
84          }
85
86      }
87
88
89      void cambioCamara()
90      {
91
92          if (cambio1)
93          {
94              cambio1 = false;
95
96              if (cameraOffset == cameraOffset1)
97              {
98                  currentView = views[0];
99                  cameraTransition = true;
100                 cameraOffset = cameraOffset2;
101                 trigger1.transform.position = new Vector3(-2.3f, 1.1f, -77.59f);
102                 return;
103             }
104
105             if (cameraOffset == cameraOffset2)
106             {
107                 currentView = views[1];
108                 cameraTransition = true;
109                 cameraOffset = cameraOffset1;
110                 trigger1.transform.position = new Vector3(-3f, 1.1f, -76.8f);
111                 return;
112             }
113         }
114
115         if (cambio2)
116         {
117             cambio2 = false;
118
119             if (cameraOffset == cameraOffset1)
120             {
121                 currentView = views[2];
122                 cameraTransition = true;
123                 cameraOffset = cameraOffset2;
124                 trigger2.transform.position = new Vector3(57.2f, 1.1f, -125.28f);
125                 return;
126             }
127
128             if (cameraOffset == cameraOffset2)
129             {
130                 currentView = views[3];
131                 cameraTransition = true;
132                 cameraOffset = cameraOffset1;
133                 trigger2.transform.position = new Vector3(57.2f, 1.1f, -126.3f);
134                 return;
135             }
136         }
137
```

```
138         if (cambio3)
139         {
140             cambio3 = false;
141 
142             if (cameraOffset == cameraOffset3)
143             {
144                 currentView = views[4];
145                 cameraTransition = true;
146                 cameraOffset = cameraOffset2;
147                 trigger3.transform.position = new Vector3(-46.97f, 1.1f, 8.05f);
148                 return;
149             }
150 
151             if (cameraOffset == cameraOffset2)
152             {
153                 currentView = views[5];
154                 cameraTransition = true;
155                 cameraOffset = cameraOffset3;
156                 trigger3.transform.position = new Vector3(-45.86f, 1.1f, 8.05f);
157                 return;
158             }
159         }
160 
161 
162     }
163 }
```

PlayerMovement script for level 1

```
1
2  using System.Collections;
3  using System.Collections.Generic;
4  using UnityEngine;
5  using UnityEngine.SceneManagement;
6
7  public class PlayerMovement : MonoBehaviour
8  {
9      private float playerSpeed;
10     private float horizontalMove;
11     private float verticalMove;
12
13     private CharacterController player;
14     private Animator anim;
15     private Vector3 playerInput;
16
17     public Camera cam;
18     private Vector3 camForward;
19     private Vector3 camRight;
20     private Vector3 movePlayer;
21
22     public Controller controller;
23     private float gravity;
24
25     public static bool play;
26
27     public Level1Canvas levelCanvas;
28
29
30     // Start is called before the first frame update
31     void Start()
32     {
33         playerSpeed = 3f;
34         player = GetComponent<CharacterController>();
35         anim = GetComponent<Animator>();
36
37         controller.enConversacion = false;
38         gravity = 9.8f;
39
40         play = false;
41     }
42
43     // Update is called once per frame
44     void Update()
45     {
46         walk();
47
48     }
49
50     void walk()
51     {
52         if (!controller.enConversacion && play) //SI ESTÁ EN UNA CONVERSACIÓN NO PODRÁ MOVERSE
```

```
53          {
54              horizontalMove = Input.GetAxis("Horizontal");
55              verticalMove = Input.GetAxis("Vertical");
56
57              playerInput = new Vector3(horizontalMove, 0f, verticalMove);
58              playerInput = Vector3.ClampMagnitude(playerInput, 1);
59
60              camDirection();
61              movePlayer = playerInput.x * camRight + playerInput.z * camForward;
62
63              movePlayer *= playerSpeed;
64
65              player.transform.LookAt(player.transform.position + movePlayer);
66
67              SetGravity();
68
69              player.Move(movePlayer * Time.deltaTime);
70
71              if (Input.GetAxis("Horizontal") != 0 || Input.GetAxis("Vertical") != 0)
72              {
73                  anim.SetBool("isWalking", true);
74              }
75
76              else
77              {
78                  anim.SetBool("isWalking", false);
79              }
80          }
81
82          else
83          {
84              anim.SetBool("isWalking", false);
85          }
86      }
87
88      void camDirection()
89      {
90          camForward = cam.transform.forward;
91          camRight = cam.transform.right;
92
93          camForward.y = 0;
94          camRight.y = 0;
95
96          camForward = camForward.normalized;
97          camRight = camRight.normalized;
98      }
99
100     void SetGravity()
101     {
102         movePlayer.y = -gravity;
103     }
104
105     private void OnTriggerEnter(Collider other)
106     {
```

```
107         if(other.tag == "Trigger1")
108         {
109             CameraFollow.cambio1 = true;
110         }
111 
112         if (other.tag == "Trigger2")
113         {
114             CameraFollow.cambio2 = true;
115         }
116 
117         if (other.tag == "Trigger3")
118         {
119             CameraFollow.cambio3 = true;
120         }
121 
122         if (other.tag == "TriggerCoches")
123         {
124             controller.enPaso = true;
125         }
126 
127         if (other.tag == "Finish")
128         {
129             levelCanvas.finish();
130         }
131     }
132 
133     private void OnTriggerExit(Collider other)
134     {
135         if (other.tag == "TriggerCoches")
136         {
137             controller.enPaso = false;
138         }
139     }
140 }
```

CreateCards script for minigame 2

```
1
2  using System.Collections;
3  using System.Collections.Generic;
4  using UnityEngine;
5
6  public class CreateCards : MonoBehaviour
7  {
8      public GameObject cardPrefab;
9      public int width;
10
11     public Material[] materials;
12     public Texture2D[] textures;
13     private List<GameObject> cards = new List<GameObject>();
14
15     public Card showedCard;
16     public bool canShow;
17
18     public CanvasMinigame2 canvasMinigame;
19     private int count;
20     private int x;
21     // Start is called before the first frame update
22     void Start()
23     {
24         //create();//PROVISIONAL. PONLO DESPUÉS DONDE QUIERAS QUE SE EMPIECEN A CREAR LAS CARTAS
25         canShow = false;
26         count = 0;
27         x = 0;
28     }
29
30     // Update is called once per frame
31     void Update()
32     {
33
34     }
35
36     public void create()
37     {
38         for(int i = 0; i < width; i++)
39         {
40             x = 0;
41             for(int j = 0; j < width; j++)
42             {
43                 GameObject card = Instantiate(cardPrefab, new Vector3(x, 0, i), Quaternion.Euler(new Vector3(0, 180,
44                 cards.Add(card);
45                 card.GetComponent<Card>().startPosition = new Vector3(x, 0, i);
46                 x += 2;
47             }
48
49         }
50
51         assignTextures();
52         mixCards();
```

```
53
54      }
55
56      private void assignTextures()
57      {
58          for(int i = 0; i < cards.Count; i++)
59          {
60              cards[i].GetComponent<Card>().assignTexture(textures[i / 2]);
61              cards[i].GetComponent<Card>().numCard = i / 2;
62          }
63      }
64
65      private void mixCards()
66      {
67          int random = 0;
68          for (int i = 0; i < cards.Count; i = i + 2)
69          {
70              random = Random.Range(i, cards.Count);
71
72              cards[i].transform.position = cards[random].transform.position;
73              cards[random].transform.position = cards[i].GetComponent<Card>().startPosition;
74
75              cards[i].GetComponent<Card>().startPosition = cards[i].transform.position;
76              cards[random].GetComponent<Card>().startPosition = cards[random].transform.position;
77          }
78      }
79
80      public void click(Card card)
81      {
82          if(showedCard == null)
83          {
84              showedCard = card;
85          }
86
87          else
88          {
89              if(checkCards(showedCard.gameObject, card.gameObject))
90              {
91                  count++;
92                  if(count == 8)
93                  {
94                      Minigame2Manager.round++;
95                      if(Minigame2Manager.round == 3)
96                      {
97                          canvasMinigame.winGame();
98                      }
99
100                     else
101                     {
102                         canvasMinigame.roundCompleted();
103                     }
104                     Minigame2Manager.start = false;
105                 }
106
```

```
107                 else
108                 {
109                     canvasMinigame.correctCards();
110                     canShow = true;
111                 }
112             }
113 
114             else
115             {
116                 canvasMinigame.wrongCards();
117                 card.hideCard();
118                 showedCard.hideCard();
119             }
120 
121             showedCard = null;
122         }
123     }
124 
125     private bool checkCards(GameObject card1, GameObject card2)
126     {
127         return card1.GetComponent<Card>().numCard == card2.GetComponent<Card>().numCard;
128     }
129 
130     public void resetCards()
131     {
132         cards.Clear();
133         GameObject[] cardsToRemove = GameObject.FindGameObjectsWithTag("Card");
134         for(int i = 0; i < cardsToRemove.Length; i++)
135         {
136             Destroy(cardsToRemove[i]);
137         }
138         canShow = false;
139         count = 0;
140     }
141 }
```

Card script for minigame 2

```
1
2  using System.Collections;
3  using System.Collections.Generic;
4  using UnityEngine;
5
6  public class Card : MonoBehaviour
7  {
8      public int numCard;
9      public Vector3 startPosition;
10     private Texture2D assignedTexture;
11     public Material hideCardMaterial;
12     public CreateCards createCards;
13
14     private float timeDelay;
15     private bool showing;
16     //private Animator anim;
17     private Animation animation;
18
19     // Start is called before the first frame update
20     void Start()
21     {
22         timeDelay = 3f;
23         showing = true;
24     }
25
26     private void Awake()
27     {
28         createCards = GameObject.Find("GameManager").GetComponent<CreateCards>();
29         //anim = GetComponent<Animator>();
30         animation = GetComponent<Animation>();
31         Invoke("show", 0.1f);
32         Invoke("hideAnimation", 5f);
33     }
34
35     // Update is called once per frame
36     void Update()
37     {
38
39     }
40
41
42     public void assignTexture(Texture2D texture)
43     {
44         assignedTexture = texture;
45     }
46
47     public void showCard()
48     {
49         if (createCards.canShow && !showing)
50         {
51             //anim.SetBool("isShowing", true);
52             animation.Play("showAnimation");
```

```
53              //Invoke("idle", 0.3f);
54              createCards.click(this);
55              showing = true;
56          }
57      }
58
59      public void show()//SE LLAMA DESDE UN EVENTO DE LA ANIMACIÓN
60      {
61          GetComponent<MeshRenderer>().material.mainTexture = assignedTexture;
62      }
63
64      public void hideCard()
65      {
66          createCards.canShow = false;
67          Invoke("hideAnimation", timeDelay);
68      }
69
70      private void hideAnimation()
71      {
72          //anim.SetBool("isHiding", true);
73          animation.Play("hideAnimation");
74          //Invoke("idle", 0.3f);
75      }
76
77      public void hide()//SE LLAMA DESDE UN EVENTO DE LA ANIMACIÓN
78      {
79          GetComponent<MeshRenderer>().material = hideCardMaterial;
80          createCards.canShow = true;
81          showing = false;
82      }
83
84      /*private void idle()
85      {
86          anim.SetBool("isHiding", false);
87          anim.SetBool("isShowing", false);
88      }*/
89
90      private void OnMouseDown()
91      {
92          showCard();
93      }
94
95
96  }
```

Minigame4Manager script for minigame 4

```
1
2  using System.Collections;
3  using System.Collections.Generic;
4  using UnityEngine;
5  using UnityEngine.UI;
6
7  public class Minigame4Manager : MonoBehaviour
8  {
9      public Slider slider;
10     public Text time;
11     private float timer;
12     public static bool start;
13     public CanvasMinigame4 canvasMinigame;
14     public SpawnerMinigame4 spawnerMinigame;
15
16     public float Speed;
17     public static int Probability;
18     public static float lifehealth;
19     // Start is called before the first frame update
20     void Start()
21     {
22         slider.value = 100;
23         timer = 0f;
24         start = false;
25         time.gameObject.SetActive(false);
26         slider.gameObject.SetActive(false);
27         Speed = 1.5f;
28         Probability = 8;
29         lifehealth = slider.value;
30     }
31
32     // Update is called once per frame
33     void Update()
34     {
35         if (start)
36         {
37             timer += Time.deltaTime;
38             checkTimer();
39             if (timer >= 120f)
40             {
41                 timer = 120f;
42                 start = false;
43                 GameObject[] coinsToRemove = GameObject.FindGameObjectsWithTag("Coin");
44                 for (int i = 0; i < coinsToRemove.Length; i++)
45                 {
46                     Destroy(coinsToRemove[i]);
47                 }
48                 canvasMinigame.loseGame();
49                 canvasMinigame.winGame();
50             }
51
52             time.text = CalcularTiempo(120 - (int)timer);
```

```
53          }
54      }
55
56      public void loseHealth()
57      {
58          slider.value -= 15;
59          lifehealth = slider.value;
60
61          if(slider.value <= 0 && start)
62          {
63              start = false;
64              GameObject[] coinsToRemove = GameObject.FindGameObjectsWithTag("Coin");
65              for (int i = 0; i < coinsToRemove.Length; i++)
66              {
67                  Destroy(coinsToRemove[i]);
68              }
69              canvasMinigame.loseGame();
70          }
71      }
72
73      public void increaseHealth()
74      {
75          slider.value += 10;
76          lifehealth = slider.value;
77      }
78
79      public void startTimer()
80      {
81          time.gameObject.SetActive(true);
82          slider.gameObject.SetActive(true);
83          slider.value = 100;
84          lifehealth = slider.value;
85          start = true;
86          timer = 0;
87          Speed = 1.5f;
88          Probability = 8;
89          spawnerMinigame.spawnDelay = 3f;
90      }
91
92      private string CalcularTiempo(int tsegundos)
93      {
94          int horas = (tsegundos / 3600);
95          int minutos = ((tsegundos - horas * 3600) / 60);
96          int segundos = tsegundos - (horas * 3600 + minutos * 60);
97
98          string hours = horas.ToString();
99          string minutes = minutos.ToString();
100         string seconds = segundos.ToString();
101
102
103         if (horas < 10)
104         {
105             hours = "0" + horas.ToString();
106         }
```

```
107
108         if (minutos < 10)
109         {
110             minutes = "0" + minutos.ToString();
111         }
112
113         if (segundos < 10)
114         {
115             seconds = "0" + segundos.ToString();
116         }
117
118         return minutes + ":" + seconds;
119     }
120
121     public void resetTime()
122     {
123         time.gameObject.SetActive(false);
124         slider.gameObject.SetActive(false);
125     }
126
127     private void checkTimer()
128     {
129         if (timer >= 30f && timer <= 60f)
130         {
131             spawnerMinigame.spawnDelay = 2.3f;
132             Speed = 1.8f;
133             Probability = 7;
134         }
135
136         else if(timer >= 60f && timer <= 90f)
137         {
138             spawnerMinigame.spawnDelay = 1.8f;
139             Speed = 2.1f;
140             Probability = 6;
141         }
142
143         else if (timer >= 90f)
144         {
145             spawnerMinigame.spawnDelay = 1.3f;
146             Speed = 2.4f;
147             Probability = 5;
148         }
149     }
150 }
```

Object script for minigame 4

```
1
2  using System.Collections;
3  using System.Collections.Generic;
4  using UnityEngine;
5
6  public class Object : MonoBehaviour
7  {
8      //public float speed;
9      private Rigidbody rigidbody;
10     public Minigame4Manager minigameManager;
11     public GameObject wrongEffect;
12     public int objectId;
13
14     // Start is called before the first frame update
15     void Start()
16     {
17         rigidbody = GetComponent<Rigidbody>();
18         switch (objectId)
19         {
20             case 0:
21                 transform.rotation = Quaternion.Euler(new Vector3(90, 0, 0));
22                 break;
23             case 1:
24                 transform.rotation = Quaternion.Euler(new Vector3(-35.663f, 2.367f, -37.463f));
25                 break;
26         }
27
28
29         minigameManager = GameObject.Find("GameManager").GetComponent<Minigame4Manager>();
30     }
31
32     // Update is called once per frame
33     void Update()
34     {
35         rigidbody.velocity = new Vector3(0f, -1f, 0) * minigameManager.Speed;
36     }
37
38     private void OnTriggerEnter(Collider other)
39     {
40         if (other.tag == "TriggerDestroy")
41         {
42             minigameManager.loseHealth();
43             Instantiate(wrongEffect, transform.position, wrongEffect.transform.rotation);
44             Destroy(gameObject);
45         }
46
47         if (other.tag == "Player")
48         {
49             Destroy(gameObject);
50         }
51     }
52 }
```

SpawnerMinigame4 script for minigame 4

```
1
2  using System.Collections;
3  using System.Collections.Generic;
4  using UnityEngine;
5
6  public class SpawnerMinigame4 : MonoBehaviour
7  {
8      public float spawnDelay;
9      private float nextTimeToSpawn;
10
11     public GameObject[] objects;
12     public GameObject[] objectsGood;
13     public Transform[] spawnPoints;
14
15     private int random;
16     // Start is called before the first frame update
17     void Start()
18     {
19         spawnDelay = 3f;
20         nextTimeToSpawn = 0f;
21     }
22
23     // Update is called once per frame
24     void Update()
25     {
26         if (nextTimeToSpawn <= Time.time && PlayerMovementMinigame4.play)
27         {
28             spawnObject();
29             nextTimeToSpawn = Time.time + spawnDelay;
30         }
31     }
32
33     void spawnObject()
34     {
35         random = Random.Range(0, Minigame4Manager.Probability);
36         Transform spawnPoint = spawnPoints[Random.Range(0, spawnPoints.Length)];
37         if (random == 0 && Minigame4Manager.lifehealth < 100f)
38         {
39             Instantiate(objectsGood[Random.Range(0, objectsGood.Length)], spawnPoint.position, spawnPoint.rotation);
40         }
41
42         else
43         {
44             Instantiate(objects[Random.Range(0, objects.Length)], spawnPoint.position, spawnPoint.rotation);
45         }
46     }
47 }
```