Design and implementation of mechanics for the development of introspective and alterbiographical narratives in First Person Exploration video games.

Jorge Fernández Sánchez

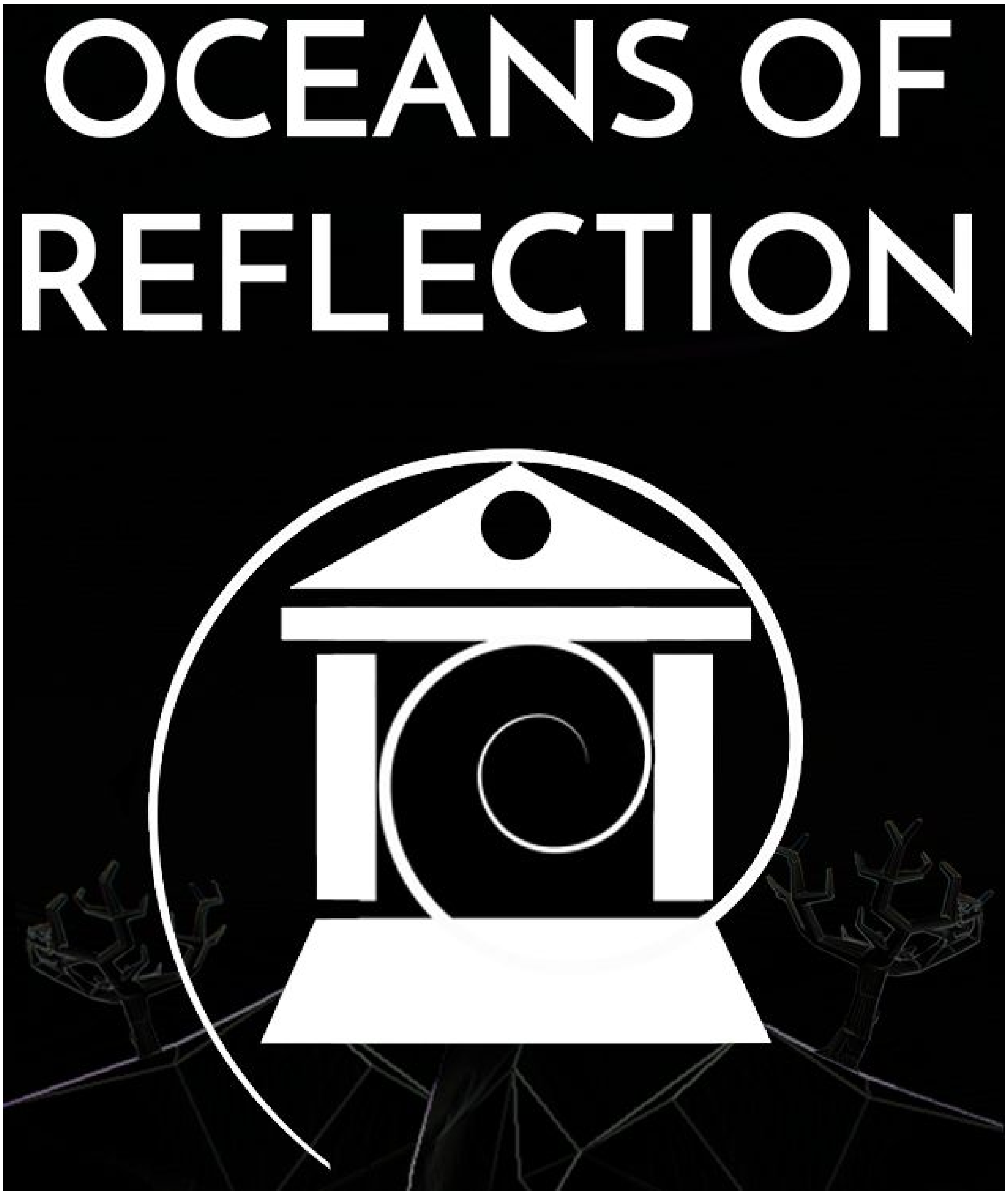
Advisor: **Marta Martín Núñez**

Final Degree Work

Bachelor's Degree in Video Game Design and Development

Universitat Jaume I

Castelló, June 9th, 2020


OCEANS OF
REFLECTION

ACKNOWLEDGEMENTS

To Marta Martin for being our mentor and helping us to cross the line.
To Xavi Perona, my brother, for being the Boyle of my Peralta, in other words, the best journey partner ever.
To Rosana Hernández, Oli Ketill and Juan Antonio Plazas for being the voice of my thoughts.
To my friends for making me blossom in Castellón and taking care of my roots in Murcia.
To my family for trusting me blindly and supporting me in every step I take.
And to the moon for standing by my side each and every night.

To all of you, thank you very much for being part of Oceans of Reflection and of my life.

ABSTRACT

The following document represents the report of my final work in the degree in Design and Development of Video Games. This project consists essentially of the development of **Oceans of Reflection,** a video game based on introspective narratives that allow the player to approach her own being through the mechanics and the environment. More specifically, the main objective is to allow the player to build a narrative together with us, the developers, moving away from traditional storytelling and adapting the experience to each of the users; in other words, exploring the concept known as alterbiography. To do this, we will design and implement in the Unity game engine, innovative mechanics in the sector, which allow to connect with the user in a diegetic way and that she creates a safe and personalized environment according to her experiences, a mirror that we provide but in which the other vital piece is needed to complete this puzzle, an entity that is reflected.

KEY WORDS

Video game, puzzles, introspective, alterbiography, immersion implementation.

INDEX

FIGURE INDEX

CHAPTER 1: INTRODUCTION

Please note that throughout the document I will use the feminine gender as a neutral when mentioning the player as it is often done in documents concerning games studies.

In this chapter I will develop a small prologue which will explain where this project comes from, which is its motivation, path and objectives it pursues as well as the fields with which it maintains a closer relationship. Given that the nature of the project is collaborative, throughout the document I will focus essentially on my main fields of work, which have been the design and implementation of adaptive and empowering mechanics of the narrative environment described below. In this way, I will leave the artistic section relegated to the background and allow my partner, Francesc Xavier Perona, to explain it in his report.

1.1 Work motivation and objective

The main motivation of the work is twofold, born from the desire to promote the video game as a narrative audiovisual medium and to connect comprehensively with the player through a unique interactive experience. It is important to bear in mind that the video game is a medium capable of connecting with the user in a much more direct way than others such as literature or cinema, since interaction is an intrinsic condition of it. The game cannot be understood without the action of the player. That's why we wanted to expand this condition and prevent the narrative section from being displaced from it, as has been happening for years in the sector where mechanics are increasingly rewarded to the detriment of stories. We were aware that in order to make an effective connection from a storytelling point of view, we needed to make the player participate in this story and that it was adequately interwoven with the mechanics, thus obtaining a pseudo-narrative consonance (solidarity between the mechanics and the narrative in a video game). It was then that we began to turn to the field of alterbiography and emerging narratives, in order to achieve our goal. Thus, starting from the base, we need to understand what the concept of alterbiography means; Gordon Calleja, in his paper *Experimental Narrative in Game Environments*[1] defines it as *"interactions with the game environment that generate story through the players' interpretation of events occurring within the game environment, their interaction with the game rules, human and AI entities and objects. It is the combination of these elements that the generation of story during game-play becomes possible."* In short, the player's way of building a story together with the developers through the tools they provide.

Once this concept was explored, the way forward was obvious, through the emerging narratives focused on video games. These emerging narratives[2] have as their star figure the "prosumer" who is the user who is not satisfied with just consuming a content but is capable of reinterpreting and reinventing it, obtaining a new one (producer + consumer). In the world of information technology, this path is no longer the future as we are literally talking about the present. In this way, through a powerful narrative field in which we involve the player, we are creating a strong relationship of dependence between the user and the product, since the latter is now intrinsically part of the former. And this is where we manage to reach the introspective section; the player is necessarily reflected in the video game because she has put a part of herself in it. We get a real and effective connection since the player is an active being, not only in a playful way, as was the custom, but now also in a narrative way and her challenge is no longer purely mechanical, it is also emotional.

1.2 Initial planning*(Fig. 1)*

Task	Time
Investigation - Level, mechanics and narrative design	60 h
Mechanics and puzzles implementation; game flow	60 h
Voice recognition and sound intensity	40 h
Portals' implementation	25 h
Photography implementation	20 h
Subtitles system, recording and implementation of voice notes - Game Sound	20 h
Maze's random generation	10 h
Real Time System	10 h
Settings and UI	10 h
Resources' optimization	10 h
Changes in runtime of the procedural skybox	5 h
Final report	20 h
Final presentation	10 h
Total	**300 h**

Fig 1. *Table of initial planning.*

1.3 Related subjects

- **VJ1207 - VISUAL CULTURE AND MASS MEDIA:** Interest in the audiovisual media and its analysis.
- **VJ1208 - PROGRAMMING II (COMPUTING):** First contact with C# and object oriented programming.
- **VJ1218 - HYPER MEDIA NARRATIVE AND VIDEO GAMES ANALYSIS:** Development of knowledge of audiovisual analysis and introduction to emerging narratives.
- **VJ1227 - GAME ENGINES:** Expanding knowledge of the Unity game engine and theoretical introduction to optimization techniques.

- **VJ1234 - ADVANCED INTERACTION TECHNIQUES:** Development of novel mechanics and their possible implementation as well as introduction to the use of algorithmic bots.

1.4 Estimated cost

For the development of this project by a company or an individual, a large number of factors would have to be taken into account. This analysis will study the most economical way to undertake this video game which is through the hiring of freelancers.

In this way, the costs associated with hiring employees, work tools and even the office can be avoided by using teleworking. Thus, a large part of the production costs are eliminated, leaving only those associated with the services provided by freelancers.

The estimated development time would be three months, making an approximation of 240 hours per worker. The number of workers would be quite modest, only three:

- Programmer.
- Artist.
- Level, narrative and sound designer.

Thus obtaining a total of 720 hours invested in the project, a little more than this project has covered.

Assuming that junior workers are employed, we could estimate that their monthly salary would be around 1400€ gross, thus making a total of 12600€ invested in workers throughout the project.

The viability of the project would be greatly enhanced by eliminating the above-mentioned expenses related to office and its associated costs or hardware; in addition, it would be possible to use software that allows free licenses until income levels are reached that would be, at first, far above expectations.

Finally, a small-scale advertising and marketing campaign through social networks should be considered, hiring companies specializing in this; approximately the money set aside for this purpose could be around 2000€. And publishing on a digital platform, such as Steam, should also be considered. The cost would be 100€ (reimbursable if sales exceed 1000€).

Therefore, this would be an indie project with a very low budget, less than 15000€, which could be carried out with the support of a small publisher or even through personal investments or crowdfunding, since we are not talking about an exorbitant amount of capital.

CHAPTER 2: RESEARCH AND REFERENCES

This chapter will be entirely dedicated to generating a theoretical framework from previous references that will allow the understanding of the nature of the project from different points of view, as well as relating it to something previously known that will evoke a feeling close to the reader.

2.1 Game concept

A fundamental part of the game are the puzzles which are clearly inspired by Jonathan Blow's game, *The Witness* (Thekla Inc., 2016)*(Fig. 2)*, however with a different background as they are not based solely on interactive panels as basic mechanics. Also, from this game and others well known like *The Beginner's Guide* or *The Stanley Parable* (Wreden, 2015 and 2013) we have extracted the personal experience of the user that transcends the diegesis**[3]**.

Fig 2. *Aerial photo of the world of The Witness.*

2.2 Game design

We also took inspiration from popular games such as *Portal* (Valve, 2007)*(Fig. 3)* to structure the design of the world and the player's movement through the use of portals. In addition, we were inspired by *The Sojourn* (Shifting Tides, 2019) due to the use of islands through which the player is able to move, a sort of puzzle containers where the action is compartmentalized.

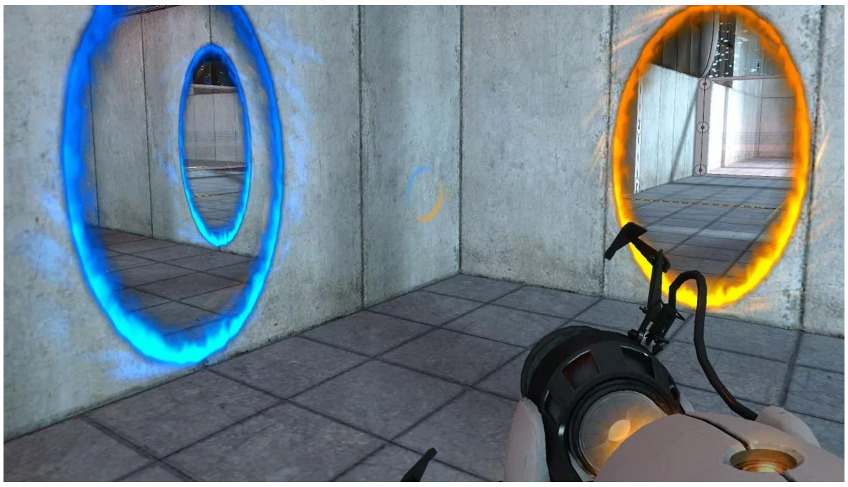


Fig 3. *Portals in Portal.*

2.3 Narrative design

An important video game in this section is *What Remains of Edith Finch* (Giant Sparrow, 2017) due to its non-explicit nature of what is in the text and leaving the player free to discover what is behind it. Furthermore, we could highlight the use of significant mechanics for each story and, therefore, for each part of the gameplay.

2.4 Artistic design

In the artistic section, there are several possible references aimed at forming a joint aesthetic. In general, the examples consist on bright colours and clear shapes that make up a familiar and comfortable visual section that is also aesthetically attractive and which can be found in games as: *Firewatch* (Campo Santo, 2016)*(Fig. 4)*, *Journey* (Thatgamecompany, 2012), *QUBE 2* (Toxic Games, 2018) or *The Gardens Between* (The Voxel Agents, 2018). Special mention should be made of a couple of games of national origin such as *RIME* (Tequila Works, 2017) or *Arise: A Simple Story* (Piccolo Studio S.L., 2019) which, again, use bright colours especially saturated with the use of so-called shaders that give the game a dreamlike aura and feeling of life. Moreover, we would like to adapt the sublime spaces that we find in *1917* (Sam Mendes, 2020)*(Fig. 5)*, that is, the field of cut cherry trees, the forest of soldiers, the singing of one of them... meaning, aesthetically attractive spaces between the moments of action. The general aesthetics could also be expressed by the video clips and music of the Icelandic group Solstafir, which uses their language with music that could define this artistic project, specifically, could be exemplified through the song called *Fjara*(2011).

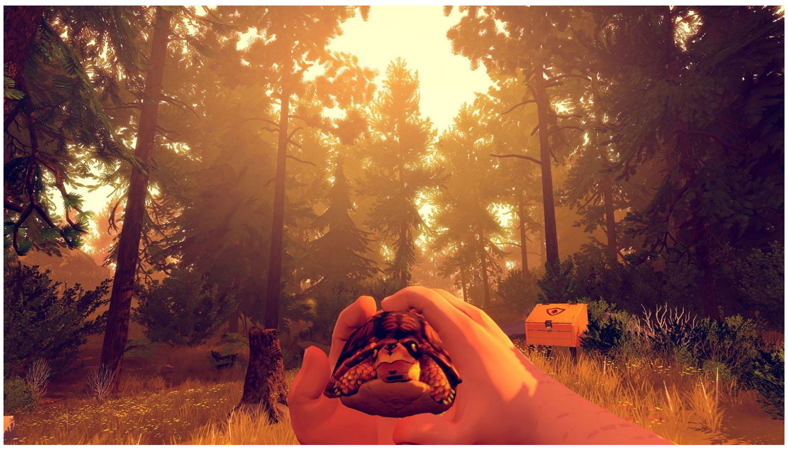


Fig 4. *Petting a turtle in the woods of Firewatch.*

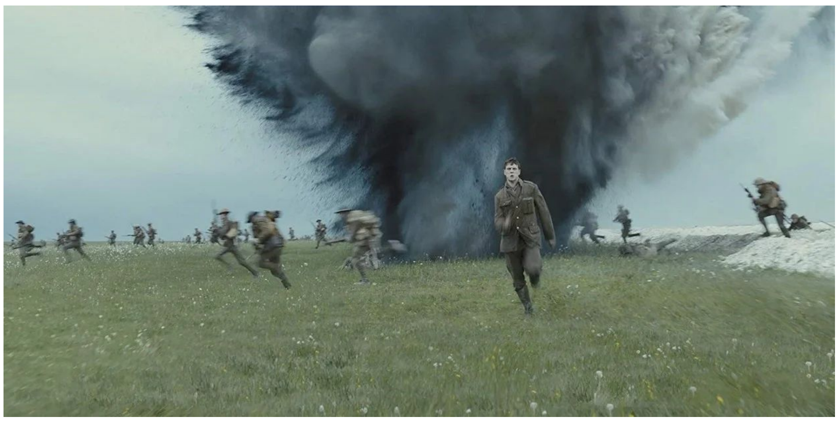


Fig 5. *Frame from 1917.*

2.5 Technical

From a technical point of view there are many references that we have used to shape the playable ecosystem. On the one hand there is, again, the clear reference to *Portal* due to the use of portals so that the player can move from one place to another (although with the exception that in this case the portals are always in the same place and the aim of the player is not to create them but to connect them through the resolution of puzzles). On the other hand there is a clear reference to one of the pioneer games in terms of manipulation of the objects that make up the world as is the case of *Half Life* (Valve, 1998). In this project we use this mechanic to solve a great amount of puzzles throughout the game, giving the player a great capacity of interaction with the world. Besides, we must also mention again *The Witness* -although there are many other examples such as, for example, the case of the *Bioshock* (Irrational Games) saga- to talk about the player's exploration and the partial discovery of the story through a sort of fascicules that are found, or sometimes have to be looked for conscientiously and that are represented by audio tapes, voxophones*(Fig. 6)*, playback devices... Another of the fundamental pillars in the technical section of the project are light and shadow games and in general, visual tricks that force the player to modify the way she understands video games in order to advance, in the style of games like *Superliminal* (Pillow Castle, 2019) or *Superhot* (SUPERHOT Team, 2016).

Finally, closing the technical section, there is a large number of elements that do not have a direct reference in the video game industry since they are completely new as is the case of the mechanics based on voice recognition and sound intensity. It is precisely this novel essence, and therefore unknown to the players, that makes it extremely challenging to transmit these mechanics since it is not possible to evoke their past experiences. However, this is one of the basic pillars of the project since we want to set a precedent connected to these mechanics and thus begin to be used recurrently in the video games of the future.

Fig 6. *Voxophone in Bioshock.*

CHAPTER 3: GAME DESIGN

This chapter will cover the basic and fundamental aspects that are part of this video game, i.e. its nature, how it will work, who it is aimed at, and what the rules will be that govern its behaviour as well as the feedback it will give to the player.

3.1 Overview

The general concept of the game is based on the resolution of puzzles along several islands that will allow the player to advance through them. These puzzles have an explicit mechanical diegetic sense and encourage the creation of the emerging narrative through the various interpretations that players can attribute to them. Furthermore, the game is classified as a First Person Explorer and Puzzle game, being this a genre very concerned with prioritizing the player's personal experience and close relationship with the game. Therefore, we try to achieve this through the aforementioned alterbiography and the direct appeal to the player's feelings, thus reaching the introspective capacity. Therefore, it is necessary to grant the player a great capacity of freedom, not so much in the mechanical sense, since this is certainly restricted to the resolution of the puzzles, although it is true that for this the player will have to make use of her ingenuity and adapt and reinvent her previous and acquired knowledge; rather mainly in the interpretative sense of the narrative, establishing a base story and allowing the players to build the rest with the tools provided, as if it were an empty continent that must be filled by the content generated by the player.

This is a complicated concept because, as long as you have the participation and freedom of the player, there is a risk that the essential message will not be grasped. This is something that we are aware of and that we assumed when we started this project, sacrificing our own vision in order to make it possible for the player to have her own. This is why, throughout the game, psychological and moral[4] dilemmas are established and both sides of the same coin are shown through opposing positions and thoughts that allow the player to adapt to the dogma she most identifies without establishing either as the absolute truth.

In other words, the game is aimed at a specific type of audience, one might even say a niche audience, which does not want to consume a content passively but needs to co-produce it. This, unmistakably, takes a great effort on the part of the player but it is also one of our main objectives that she never feels so free as to become frustrated by the inability to know what to do, a problem into which many games fall and which is also dealt with in the game in a diegetic way in its practical application through the condemnation to be free and the anguish of Jean Paul Sartre[5]. In short, it is difficult to establish an objective experience for the players, because one of the main strengths of this project is precisely the achievement of different experiences depending on each player. In general, the direct appeal to emotions makes it very difficult to make any parameter unbiased, and that is where the power of the project lies. We use the strength, in this case the emotional strength, of the users to adapt the final product in the same way that clay itself does not have a specific shape but is capable of being moulded according to whoever is handling it.

Nevertheless, within this adaptation and personalization, the general climate is that of the calm before the storm, the player not having an explicit incitement to advance yet feeling the need not to get stuck with a self-imposed urgency. All this is also surrounded by an atmosphere of research, exploration and recognition since practically everything that is shown is new to the player; therefore she must leave her comfort zone. Finally, we create a constant learning environment in which very few things can be taken for granted and which requires a great deal of concentration on the part of the player leaving her own destiny in her hands, in the literal and figurative sense as well as the diegetic and extradiegetic.

3.2 Gameplay

One of the key elements within the gameplay of this project is its spiral essence. At first, the story was conceived as a circular structure, since the point from which it began was the same as the point from which it ended. However, throughout the development of the project, we were aware that this was not the loop that we wanted to transmit, since the end point of the game had to be similar to the initial one, but not identical. And so we arrived at the Archimedes spiral*(Fig. 7)*. In this way, the game becomes an infinite loop that can be replayed but that gives the player the possibility to break it and get out of it (we will explain this in the next chapter). The structure, however, is deceptive and can only be correctly appreciated when it is done with a certain perspective, since at first, it is shown as an ascent towards light and knowledge, inspired by the allegory of Plato's cave**[6]***(Fig. 8)*. This metaphor is one of the main ways of approaching the player through the evoked narrative, being able to connect with her through sensations she has already experienced and stories she already knows.

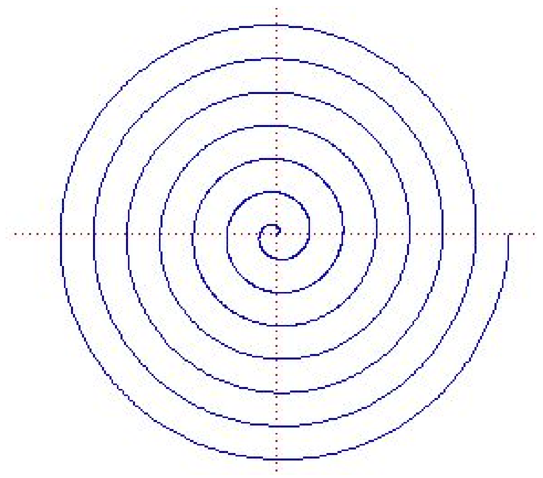


Fig 7. *Archimedean spiral.*

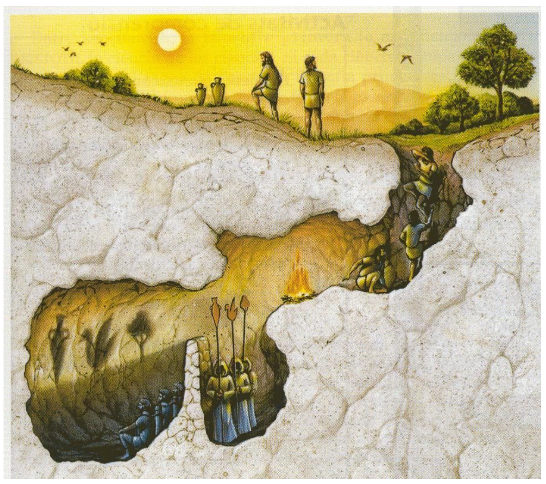


Fig 8. *Plato's cave allegory.*

The gameplay route guides the player through 6 different islands, the final 2 being an idyllic and demonized version of the first one. The navigation through these islands is unidirectional and forces the players to make the decision to leave each one of them, leaving everything related to it behind, except the knowledge acquired, if they want to advance to the next one. This is the mechanical representation of one of the recurring themes of the project, "letting go" through stoicism and the path of TAO. Besides, as we mentioned before, a cyclical structure is followed in the global gameplay, however, there are within this global structure, microstructures with their own mechanics that allow to build a general framework. These microstructures are each one of the islands, in which diverse themes are treated but which follow the same leitmotif, the representation and conception of the self.

The mechanics are the basic element on which we rely to achieve the effective immersion of the player and the development of the previously mentioned framework. Therefore, we are talking about very new mechanics in the video game sector, many of them never seen before and that deepen the relationship with the user. Furthermore, one of the essential components is the factor of constant learning; that is why we maintain a continuous balance between the reinforcement of what has already been acquired (the mechanics that the player has managed to handle correctly) and the strengthening of what is new (those mechanics that the player has not yet internalized). So, conceptually, the basic meaning of the game is simple: progress to advance, avoid stagnation and reach knowledge. A sort of pseudo journey of the hero. However, it is the player who sets the pace, since the urgency is self-imposed, except in a single pre-set time event to which very few players will manage to arrive on time, thus working in an intentional way - it will be discussed in the next chapter.

This is why the main objective within the game is precisely to play. This may seem a little confusing or redundant, so I will use a practical example: life. In general, in their life people can set goals for themselves in a shorter or longer period of time, that is irrelevant. The sense is that they always advance according to these goals that they set (or that are set externally). And therefore, their life revolves around these goals. However, moving on to a much more philosophical conception, what is the meaning of these goals? And that leads us to what is the meaning of life? Yes, of course, to move forward and achieve what we set out to do, but why? This has been one of the key questions for philosophy for millennia and there are hundreds of positions and thoughts on this subject. However, what is clear and what has become generally accepted in the bulk of human thought is that life is not about destinies but about the paths that lead to them. Similarly, this video game is not about arriving at a destination in the literal sense of the word, but about what happens in the player's mind along the way. And this is the reason for the spiral structure; within the game players are in an eternal loop where nothing is exactly equal but at the same time nothing changes. The player's main objective will be to achieve knowledge, truth, resolve the conflict of why it is there and move on; however, this effort will seem to be in vain from the point of view of replayability.

This brings us to the save and replay. One of the fundamental pillars of the game is the impossibility of saving progress. This is a design decision that, together with the inability to stop the game world, even in the pause menu, leads us to a realistic and anti-game path. Realistic because in the "real world" nobody can't stop at will; time always moves forward, whether people want it to or not. Maybe it can be controlled how fast or slow people go, or it can even be controlled when to listen or take action. But what cannot be done is stop time. Neither can players in this game. And that brings us to the second point. The checkpoint, the mechanic of video games par excellence. Nowadays it's almost inconceivable to think of a video game without checkpoints and it was one of the aspects we wanted to break with completely. If the player abandons her game, she will lose all the progress of the game. It's a concept based on a sense of realism and which is designed to prevent the player from modifying events as she pleases. In this way, we are transmitting in a mechanical way, what we also express in a narrative way: from the past the player can only learn for the future but she cannot undo what is already done -CTRL+Z syndrome which will be introduced in chapter 4. We allow ourselves this license due to the investigative nature of the project, however, we are aware that this would be practically unfeasible in a larger project or one that was destined for commercialization for simple logistical reasons that, sometimes, condition the nature of the projects.

This unmistakable and unidirectional flow finally leads us to the infinite loop, the spiral we mentioned earlier. At the end of the game, the player is offered a veiled decision based on the choice between action or inaction[7] (and thus understanding the latter as the action of doing nothing). On this decision depends the permanence inside the spiral or the breaking of it and that's why we said that the player is offered the possibility of ending the loop[8] (in a diegetic and extradiegetic sense). If the player decides to act, which is the obvious answer since what is presupposed of the user is her capacity of action by being an active entity, the game will follow its normal flow, taking her to the initial menu and allowing her to play with new knowledge and experience. If, by the opposite, she opts for inaction, the resolution of the conflict will be different, perpetuating a ludonarrative[9] consonance and thus closing the application of the game, avoiding replayability and concluding the loop. The sense of this is that the player feels that the game experience goes beyond the virtual world and that she is able to trace the similes with the real world.

3.3 Game progression and mechanics

In this part we will comment on the general flow of the game*(Figs. 9 and 10)*, observing the course of the player's journey yet only from a mechanical point of view. The narrative development will be seen in chapter 4 and the implementation of the mechanics in chapter 6.

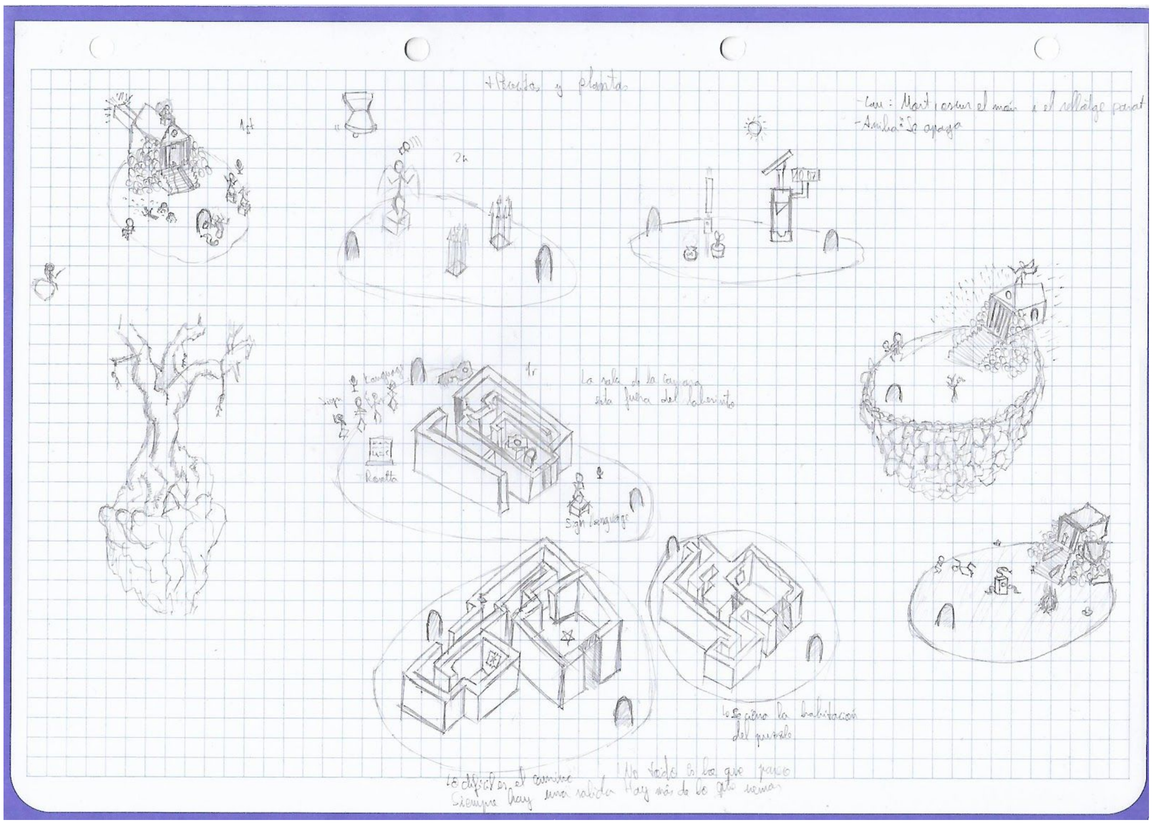


Fig 9. *Game progression.*

3.3.1 Initial Island

First, after an initial sequence with various quotations, the player appears at the beginning of a corridor, at the end of which a light can be seen. At this point, the player will experience two of the basic mechanics of which the video game is composed and which are so established in the standards of the industry, that they do not require any introduction. These mechanics are based on the player's movement (with the WASD keys) and the ability to look around (with the mouse). Within this movement available to the player, jumping is not contemplated because the puzzles are not based on slopes that must be overcome.

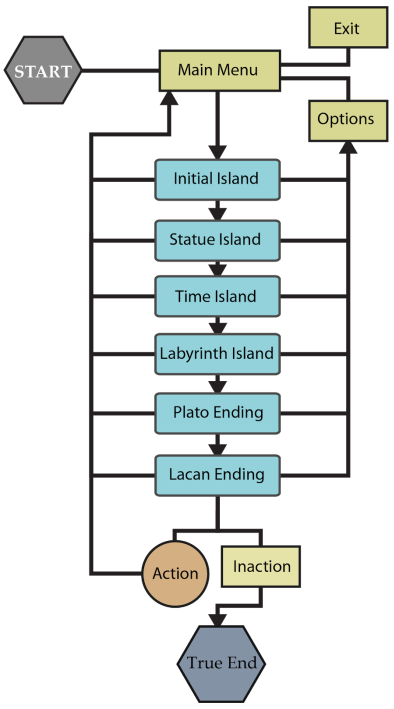


Fig 10. *Gameflow.*

Because of the basic knowledge that people have about video games, the player will try to go forward, towards the light, since it is the most obvious solution (within the Western playful culture, we always tend to go right in the case of a side scroll or forward, that is, in the sense in which we first conceived the game, in the case of a 3D world). At this point, one of the first breaks in the gaming pact with the player will occur, beginning to break these standards of the video game and preventing the player from reaching the place that she has taken as her objective because that is how we have presented it to her, the light. Every time she approaches, she will be made to move backwards, leading her to the starting point and at the same time voices begin to sound urging her to stop being obsessed and change her course. The moment the player understands that the one who seemed to be the target is no more than a trompe l'oeil and that she has to turn around to truly begin the game, the idea that the mechanics presented do not follow a recognizable pattern in most cases will begin to creep into her mind, practically unable to make use of her previous knowledge and urging her to open her mind and relearn.

Once the player leaves the hallway, she will be able to see that she was inside a temple, located on the initial island. Within this, we find the introduction of 3 basic mechanics:

- ❖ **Sound intensity recognition:** on this island, the player will find two statues placed next to each other and with a clear differentiation on their heads: one will have a disfigured skull while the other will have an intact one. As the player approaches them, she will be able to see how one of the 2 (the one without the damaged head) changes some of the properties of his material progressively. This change of properties is given by the sound intensity, in decibels, that the player's microphone is receiving. Therefore, when an element of the game sounds (footsteps or music) or the player makes a sound (hawking, humming, talking...), the material of the statue will change, making it clear that these statues, in particular, are capable of listening to the player. This is the way to make the player see that she can be heard and sometimes understood (as will be seen in the voice recognition mechanics of the next island).

- **Portals*(Fig. 11)*:** the player will be presented with the first of the portals through which she can see another island different from the one she is on. This is a mechanic of which it is possible to have a certain reference due to the acclaimed game *Portal*. However, the main difference with it is that in this case, even though the portal vision is activated, the portal teleportation is not, thus making the player's efforts to cross the threshold useless. Thus we are showing the player that she still has to perform some action to produce the full activation of the portal.
- **Grabbing objects:** also there will be a tree with an apple hanging from one of its branches and through the portal, the player will be able to see the same tree but without the apple. This, added to a series of biblical audios referring to the original sin, will make the player understand that she must take the apple in order to activate the portal. In this way, and as I will explain in the narrative section, we are pushing the player to sin, leave her comfort zone and go into the unknown, that is, we are exercising the role of the evil Cartesian genius.

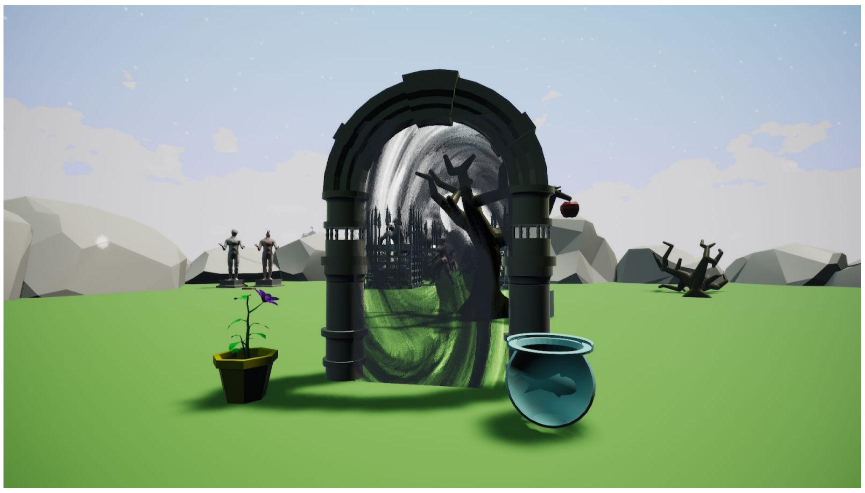


Fig 11. *First Portal in Oceans of Reflection.*

3.3.2 Statue Island

This island will introduce the mechanics of voice recognition and will add to the previously known sound intensity as well as the portals, which are maintained throughout the game. We make a special distinction between sound intensity recognition and voice recognition since in the first case, we are talking in a generic way about sounds that are captured by the microphone, regardless of where they come from, while in the second case we are talking about the recognition of specific words articulated by the player, similar to how chatbots work.

Once on this island, the player will first see that she can't go back to the previous one, because for design and narrative reasons, we decided to make the flow of the progression one-way. The first vision she will have will be of a large number of statues, locked in individual cages and a huge central statue on a pedestal and released from his prison. However, in this case, the visual impact will be nothing compared to the true power of this island that lies in its sound impact. This is because after crossing the threshold of the portal, the player will begin to hear a great deal of screaming, sobbing and wailing coming from each of the enclosed statues. One of the essential factors of these screams is that they will produce a great deal of stress and strain on the player, not only because they won't know how to quiet them down, but also because they will be unable to control the volume of the screams through the game's sound options.

The central statue will order the player to silence the rest of the statues (or more specifically the souls enclosed in them*(Fig. 12)*, although I will explain this in the next chapter). Through this incitement by the statue on the pedestal and also taking into account that the player has already been introduced to the mechanics of talking and being heard by the statues, will end up reaching the conclusion that she has to order the statues to STOP. However, the player is not only aware that the statues are listening to her but also that they can detect whether she is whispering, talking at a normal volume or shouting. Therefore, when she recognizes the right word to silence the souls, she will also have to say it at the right sound intensity level, giving her the appropriate visual and auditory feedback to shout and even beg. In this way, we will be generating a feeling in the player of uneasiness that will finally be resolved with absolute silence when she quiets them down and gives way to calm.

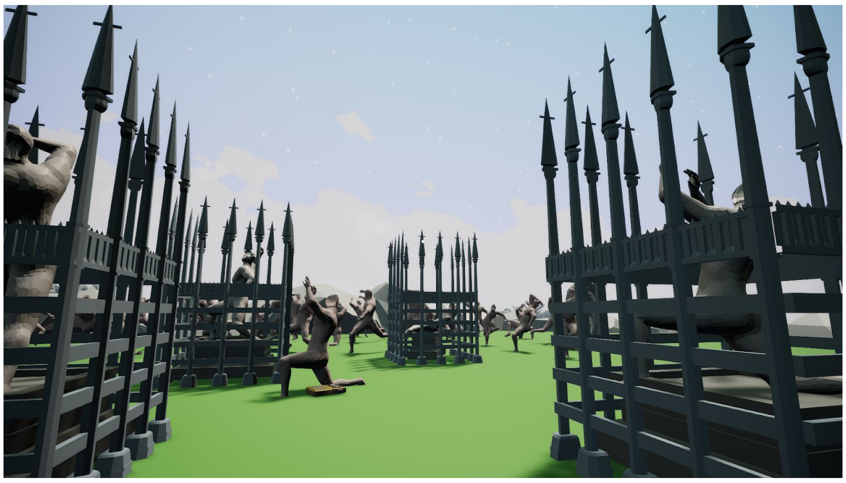


Fig 12. *Prisoners.*

3.3.3 Time Island

On this island, unlike the previous and the next, the mechanics involved are much more cognitive than practical. This is due to the fact that, within the narrative part that I will comment later, this island certainly has a relaxing role for the player since she has been subjected to a great deal of pressure in the previous test and has to face the final straight of the game with energy. In addition, another important detail is that this island is at the same time the reward or the punishment of the actions that have been taken so far and it is precisely this dye of denouement that gives it a much calmer aspect that is, of course, accompanied by the emotions transmitted through the mechanics.

In this case, when the player goes through the portal, she will find two recursive elements that she has been following (since they were part of the environment) and with which she has ended up empathizing while they were arousing her interest. These elements are a fish and a plant; both living beings are about to die and need water to avoid it, but there is only enough water for one of them, who lives and who dies? The player must make this decision, condemning the one who does not receive such precious sustenance. If the player is blocked and is unable to make a decision, both beings will die irretrievably and the player will have lost valuable time to save the third life that was at stake and that is located a few meters ahead, located in a guillotine*(Fig. 13)*. This puzzle starts from the moment the story begins. From that moment on, every 5 minutes, some bells will ring. First 5 bells, then 4 and so on until the decreasing count reaches the last bell. While these bells follow one another, night will fall until the last one is reached, when total darkness will be reached and the time of execution will be over.

Since this puzzle depends on the player's ability to solve the previous ones, it has two possible outcomes:

- The first and least likely, due to the sum that the player will not have a strict sense of urgency beyond the bells and the difficulty to understand and solve the puzzles, in which the player is able to reach the island of time before the execution. In this case, she will notice that the guillotine is connected to a clock that marks the current time of the "real" world. That clock, in turn, is connected to a photovoltaic panel and this is the most important detail so that the player can solve the puzzle in time and avoid execution. This cognitive connection, added to the audios in which special relevance will be given to time, sunlight and night, will allow the player to understand that her mission is to prevent the clock from being powered by solar energy in order to avoid the time of execution and one more death. Thus, having introduced the concepts of changing the perspective with respect to conventional games and interacting with the world, the player will be able to touch the sun to turn it off as if it were a light bulb and thus prevent the clock from reaching the time of execution.
- The second, and almost certainly more frequent, will be the scenario where the player does not arrive on time to the island and therefore night falls with the last bell and with it, death. When the player finally arrives to the island, she will encounter the dantesque scene and will have to deal with the feeling of responsibility. And it is precisely this feeling that she will have to overcome if she wants to continue on her path; this will manifest itself mechanically through the clock that is connected to the guillotine. As it is night, the clock stops at the time of execution (the real time for the player). Nothing can be done to bring back the life of that poor executed soul, it can only go forward, and for this, some buttons have to be manipulated, located in the clock, which allow to change the time; in this way, the clock has to be updated with the present time, implying the overcoming of the past and the acceptance of it; During the time that the clock is updated to the player's current time, the portal will be open. However, if the present time changes and the clock is out of date, the portal will close again, understanding that the player is still stuck in the past.

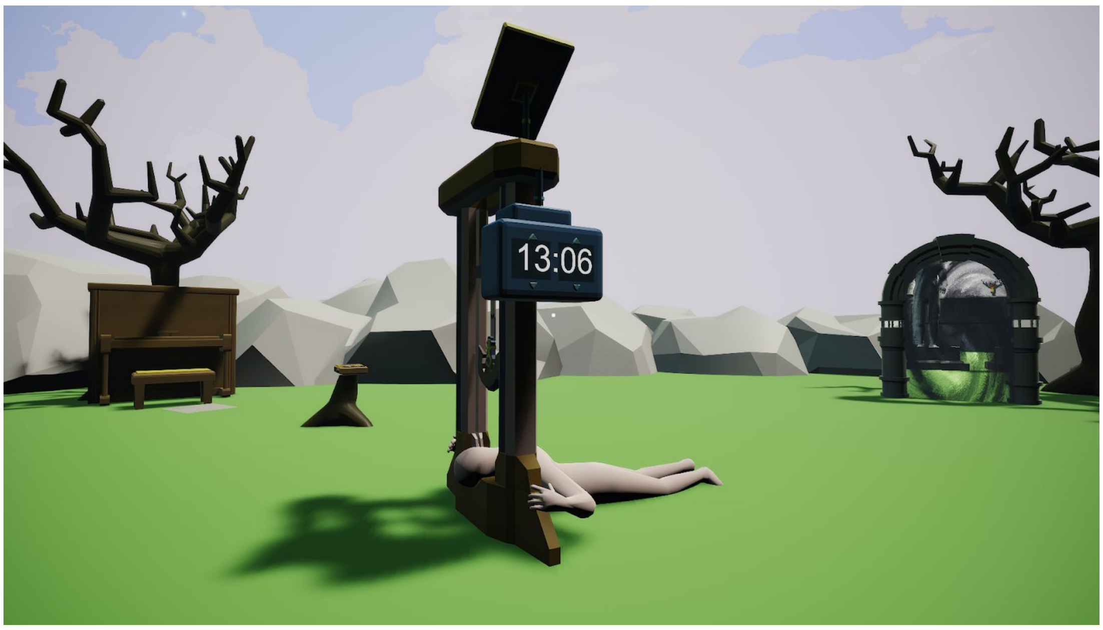


Fig 13. *Guillotine moments before the execution.*

3.3.4 Mazes Island*(Fig. 14)*

Once the time island is overcome, which as we said implied a very light mechanical performance, the player will be faced with one of the most important mechanical challenges along the way; the prelude to the end of the game (apparently). This island will be made up of a total of 6 puzzles in which there will be a great involvement of mazes but each one with some peculiarity that will take it away from the traditional gameplay. That is why this is the most complicated moment with respect to gameplay because it will be the time when the player will have to test the knowledge she has acquired throughout her journey, add new ones and above all, deconstruct her thinking of what she thinks she knows.

When passing through the portal, the player will find a series of transcriptions next to some statues in different poses; even though at that moment the player will not be able to understand the transcendence of that space, it will permeate within her so that later she can resort to that which passes from being unknown to knowledge. Next to this space, she will find a wall on which a key is projected but when she tries to interact with it, something she already knows, she will realize that she is not capable and will have to reinvent the solution. Faced with these two elements, the transcript and the key, against which nothing can be done, the player will continue forward, thus reaching the beginning of the first maze. In this case it is a simple labyrinth, without traps or double bottoms. The player will only have to find the way out, for which there will be only one path. However, and as the reader may already be imagining, this is just a trompe l'oeil, once again. As soon as the player has overcome this labyrinth, she will arrive at a room in which he will find essentially two components: a Polaroid camera and a door with a lock that will have the "non-existent" material by default of design applications such as Photoshop. At this point, the player will try to interact with the camera as she has done with the rest of the elements of the game and this will allow her to take it and use it to take pictures. Since the way forward is blocked, it is obvious that something has to be done with the recently acquired elements in the previously visited places. This is the moment when the player's neural connections come into play, understanding that, since she could not take the key in the physical sense, she must photograph it. This is the moment in which the trap of the previous labyrinth is revealed; the player will be confident to go through it in the opposite order and thus reach the exit but, to her surprise, she will realize that the labyrinth has changed completely. When she manages to solve it again, she will photograph the key and must return to the door room and this is where the second turn takes place: the maze that had acquired two completely different shapes, is now a simple corridor, representing the reduction of the difficulty in the problems once one has adapted to them and sees them with the perspective of time.

By placing the photo of the key in the lock, the door will open but the player will not be able to go through it since she will find a statue, capable of listening, blocking the way; this statue will have a particularity and will be the posture that he will adopt, since this will be the same as one of the postures of the statues that were next to the transcriptions. When the player finds this out, she can investigate until she reaches the conclusion that each of the statues represents a word from the sign language transcription. The player must then return in front of the statue and say that word, causing the statue to fade and move on to the next maze. In this, the change of perspective will be key in a literal sense since, apparently, the labyrinth will not have any exit. However, after going through it, the player will arrive at a room where she will be able to see a screen in which one of the rooms of the labyrinth is observed. This room will be shown from a zenithal view, allowing the player to see that one of the walls of the room is only a visual effect and that she can go through it, thus solving the puzzle. Finally, the exit from this labyrinth will take the player to the final test: the box. Upon entering the box, the player will be completely enclosed within four walls. There will only be one element in this improvised room, which will be a button. By pressing this button, there will be a sound of a door opening; however, once the button is released, the door will close. This is the ultimate test as it tests the blind confidence of the player, who will hear audios related to faith and the myth of Eurydice**[10]**. This way she will understand that the only way to escape is through trust and therefore she will have to keep the button pressed while she walks backwards, thus reaching the end of the island and entering the portal that will lead her to the resolution of the game.

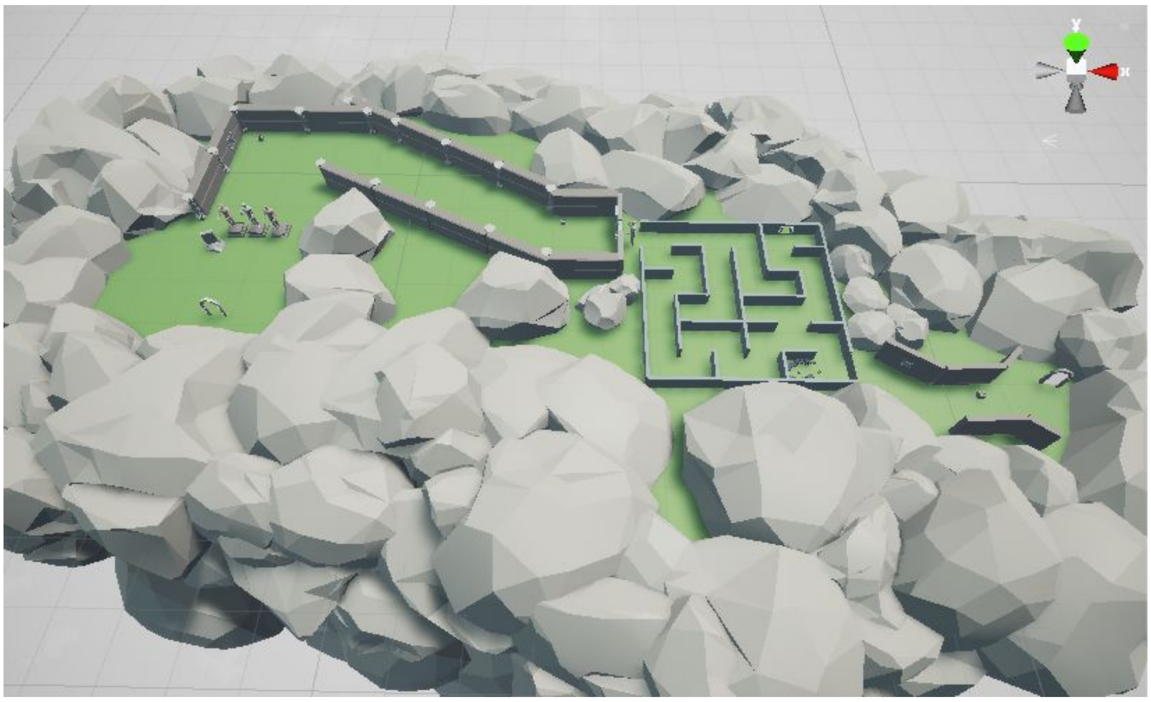


Fig 14. *Aerial image of the mazes island.*

3.3.5 Endings

In the false ending, the player will walk around the idyllic island of the Platonic ending, which is a paradisiacal representation of the initial island, entering the temple and reaching the light. However, this will only lead to the real ending, the Lacanian**[11]**, which is a demonized version of the initial island. At this point the player will see a button and begin to hear the cries of the statue island but with a peculiarity, this time also heard herself begging to stop. This way she will understand that her mission is to help herself, pressing the button that opens the cages and making the souls in pain shut up while allowing her past version to continue forward. However, by doing this, she will be perpetuating the loop and will remain locked forever in the infinite spiral, doomed for all eternity and, in an extradiegetic sense, returning to the home menu to play again. What the skillful player might think is: whenever an action-based option is offered (pressing a button) you always have the alternative option based on inaction (not pressing the button). This is a relatively complex concept in the world of video games because the players are designed and educated to be active and always act. That's why we wanted this decision to be based on the breaking of one of the canons of the video game and therefore, not to present an alternative button but the alternative itself was, not to interact with it. In the event that the player decides this, the cries of supplication of her past version will be added to those of the souls in pain; yes, she will have broken the loop and saved her soul but condemned the souls of all those before her. This decision will logically have its extradiegetic repercussion, making the game close and not allowing the story to begin again, part of a book that has already been closed.

3.4 Interface

Since we want the game experience to be as immersive as possible, we don't want an intrusive HUD restricting the connection between the game and the player, that's why we intend to introduce a minimalist HUD formed only by a scope that facilitates the player's vision and that prevents the player from suffering the well-known motion sickness generated by the lack of it; with this we also pretend to avoid any extradiegetic text that guides the player, for example "Press E to interact with this" or "Click here to activate this button".

Regarding the menus*(Fig. 15)*, we also try to make them as minimalist as possible in order to keep the player in touch with the playable experience. We will introduce the relevant configuration settings such as volume, activation or not of the subtitles, fullscreen, resolution and quality of the graphics but always in a very sober style.

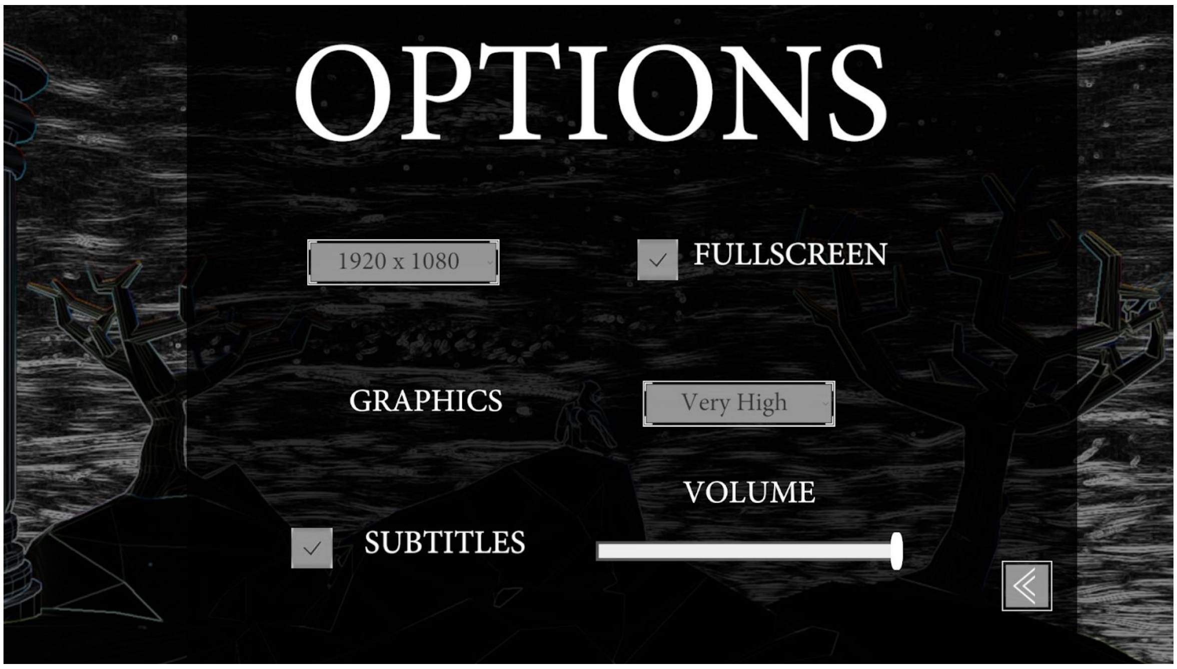


Fig 15. *Options' menu in game.*

CHAPTER 4: NARRATIVE DESIGN

In this chapter, the main narrative aspects of the video game will be discussed in depth, from its general structure to the narratives that compartmentalize it; in addition, some more theoretical aspects related to the theory of narrative in the interactive audiovisual media will also be dealt with.

4.1 Structured narrative and emergent narrative

This is an especially important section within this project, since the development of the alterbiography, has implicitly the development of emerging narratives that allow a joint construction of the narrative between the designers and the players, thus obtaining a great amount of different stories, as many as there are playable experiences, from the same base. To achieve this, it is essential to establish consistent foundations, which in this case are supported by an introspective journey through the player's morality and with strong philosophical reinforcements that allow us to explore the conception of being in an internal sense (entity) and in an external sense (projection). This is done through an arc of argument based on some of the most important moral questions in the history of humanity, such as: destiny, knowledge, the capacity for action and decision, responsibility and guilt, the (non-) existence of God and his benevolent identity as opposed to the malignant and vengeful one, not hanging on to what cannot be sustained, life as a loop and the meaning of living it.

However, these topics are presented in a general way, avoiding the classic error of positioning. If we were to impregnate the video game with our opinions, the overall sense of the narrative construction would be completely lost, since the player would feel a lack of freedom in being conditioned and guided by us. In this way, without imposing anything on the player[12], we are obtaining a greater involvement on her part since she is not being told a story but is herself constructing it, through these emerging narratives. Furthermore, these are greatly helped by the fact that there are no explicit explanations of what is happening during the gameplay, since there is no narrator as such and at no time is explicit reference made to diegesis; what is done is to suggest different paths so that the player can choose which one to take and, based on this decision, make it her own and build it according to her understanding of the world and of herself. In this way, in this videogame, the structured and the emergent reach a perfect harmony, without seams, in which there is no clear defined limit, allowing to obtain an organic but at the same time interesting story and in which the player is both an actor and a director through her thoughts and deductions - this concept will be further explored in point 4.3.

4.2 Storyline and transformation arch

One of the main structural components is the spiral shape of the story, establishing the same starting point as the end but with certain differentiating factors that give the sensation of drawing circles but applying a small offset each time that prevents the starting point from being exactly the same as the end point; No matter how close they seem to be, there will always be something that distances them, in this case, that something is the player's experience and the reinterpretation of what they have lived through based on their personal experience, which can never be the same since, as Heraclitus said *"you could not step twice into the same river"*; nothing stays the same.

Once the dichotomy between the structured, the emergent and the general narrative flow has been presented, I will move on to comment in detail on the ideas behind each of the parts that make up this video game; in this case I will divide it up, as in the rest of the document, on the basis of each of the islands. It is important to point out that this section will essentially talk about the structure designed by us as well as some possible interpretations that this may give rise to; however, as is obvious, it is impossible to think about - let alone write about - all the possible narrative variants that will arise depending on the player's interpretation.

4.2.1 Initial Island

In this first contact of the player with the world, we intend to bring her closer to knowledge and issues that are not extremely alien to her, in order to avoid early saturation caused by a mechanical/narrative difficulty curve that is excessively vertical[13]. That is why on this island, we use the resource of evoked narrative, thereby facilitating the introduction and immersion of the player, generating an environment that is safe and not so unfamiliar.

The key issue at hand is the perception of reality through the allegory of Plato's cave; another main aspect that occurs on this island is the connection between diegesis and the extradiegetic; this is because, as I said before, through the evoked narrative we try to generate a safety zone for the player (extradiegetic) and meanwhile, in diegesis is presented the fear of the unknown and the need to leave that comfort zone if the player is forced to do so, pushing her into the new and unknown. This is one of the first times that can be observed that the line between the diegetic and the extradiegetic is quite thin - a topic that will be discussed in depth in section 4.3.

Finally, on this island the player can also find the analysis of why things happen, that is, whether they are motivated by something or someone and if so whether their intentions are pure or not, thus introducing the figure of the Cartesian evil genius*(Fig. 16)*. This will be a question that will spread throughout the whole game, making the player reflect on the real relevance of her acts, depending on whether they can be considered voluntary, thus being completely responsible for them or on the contrary, they are part of something bigger in which the human being cannot intervene. The intention of this dichotomy is to show the true anguish behind either case: if everything depends on us, we are responsible for everything that happens and if nothing depends on us, we are mere dummies in a theatre play in which we never ask to appear.

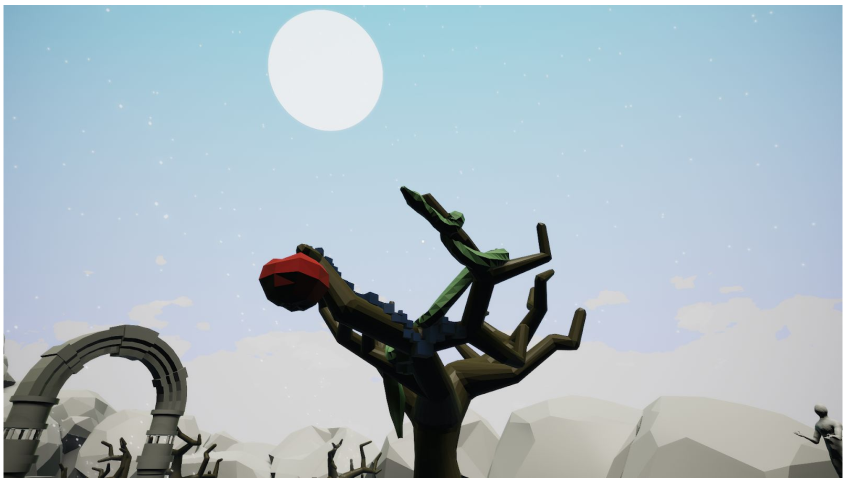


Fig 16. *Temptation of the snake to take the apple.*

4.2.2 Statue Island

This island could certainly be considered transitory because it still conserves some introductory tints like those of the initial island but more complex reflections can also be glimpsed that indicate a point of no return towards knowledge and introspection. Within the introductory branch the player could extract the (im)possibility of eternally escaping from the problems, by means of a contrast of ideas, or it could also be highlighted the responsibility VS the non-responsibility of the acts, developing what was introduced in the previous island. However, as I said before, there are also aspects that are beginning to escalate in complexity and that are already moving away from the comfort zone of the majority of users. For example, the philosophical current of Stoicism that refers to the power of letting go and understanding the circumstances that do not depend on people but that they can control how they affect them - contrary to the popular belief that Stoicism is based on a complete and passive resignation. We also refer to myths from Classical Greece to talk about punishment and the concept of eternity or the meaning of existence; this is the case of Sisyphus**[14][15]***(Fig. 17)*, which was also analysed by the famous French philosopher, Albert Camus. In addition, the concepts of inaction and the illusion of non-responsibility will be introduced in this island, which will be key to the outcome of the game.

Moreover, one of the signs of the union of the emerging, unspoken narrative and the environmental storytelling is the disappearance of the statues when they are silenced. This is a crucial moment in the game as it introduces the end of the game and also allows the player to generate part of the story through their free interpretation. One of the many that could be given to this event, and specifically the one that we have thought about when designing this fragment of the game, is the existence of souls, whose material bodies have died and therefore have to take refuge in the statues of this place between life and death. However, if they spend too much time inside the statues, the souls are extinguished forever and die, so they need to get out of the statues and transcend when they are ready. The problem that is generated in this island is that the souls sheltered in the statues, are locked inside the cages, preventing them from leaving their temporary rocky home and condemning them to perish inside it. That's why the souls scream, to attract the player's attention and be released from their prisons before it's too late. On the other hand the statue on the pedestal implores us to silence them, not because it is merciful and wants the best for those souls, but because - in relation to Sisyphus - that soul is condemned for all eternity to live caged in a statue, not granted the relief of death, and wishes peace in its eternal punishment**[16]**.

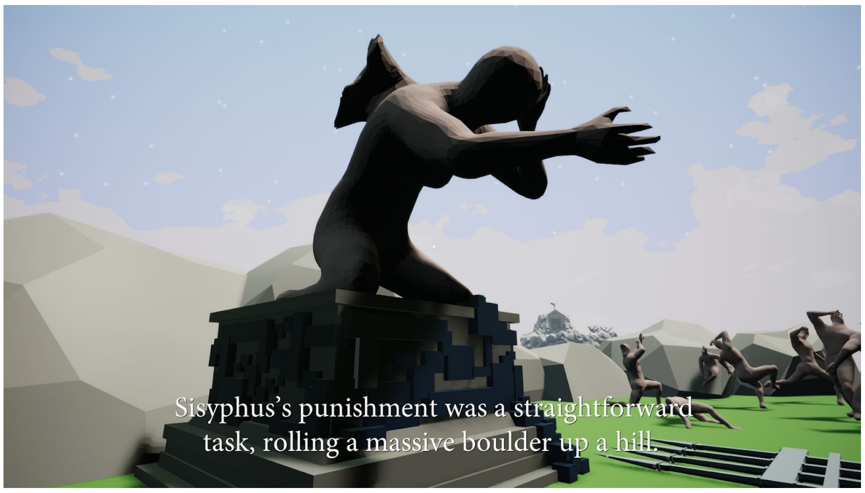

Sisyphus's punishment was a straightforward task, rolling a massive boulder up a hill.

Fig 17. *Sisyphus statue.*

4.2.3 Time Island

On this island, personal reflections are more in-depth, leaving aside those of a more generalist nature. This is why we are once again deepening individual responsibility and guilt, as well as the consequences of our decisions and the assumption of them. With respect to this particular theme, one of the many reflections of our own that are found in the game is introduced; we highlight this one particularly because it is a concept created by ourselves and which we call "CTRL+Z Syndrome". It is based on the tendency of the human being towards a world in which nothing is permanent but everything can be undone and in the frustration that generates the imposition of assuming the consequences of our acts, especially when these are not the expected ones. In addition, another line of thought is introduced parallel to stoicism due to the exaltation of the force of letting go, this is the Tao.

This island, as its name indicates, also speaks of time, of the interpretation given to it, of the control that can be exercised over it and of its unidirectional condition. In this way the video game connects expressly with each one of the players because in an intrinsic way, time is always connected to someone, that is to say, time as such is a human construct and that is why its existence cannot be understood without a living entity to validate it.

4.2.4 Mazes Island

This could be considered, without fear of being wrong, the island where narrative has a greater connection with mechanics and that greatly enhances the freedom available to the player in relation to alterbiography.

Here we are talking essentially, through playability, about representation and the relationship between what one is and what one shows, linked to the image as a capture of a moment; we are also dealing with the change of perspective - in the literal as well as the figurative sense - and we are going deeper into the break with the conventional canons of the video game as well as into the development of the meta-game and the self-awareness of the artwork. In addition, there is also a recurrence of the recourse to explanation through myths, as in this case that of Eurydice, related to faith and trust. Again, as in the previous island, the idea of time as a river is reinforced, impossible to stop and always flowing in the same direction, forward, tireless and ending with anyone who tries to swim against the current.

Other issues are also very important on this island, such as the reinterpretation of the same situation according to the moment in which it is lived and, therefore, the knowledge that is available, as well as truth over perception.

4.2.5 Platonic ending*(Fig. 18)*

This is what could be considered the idyllic ending: the one in which the player acquires enough knowledge to transcend, returning to the starting point but seen in a different way. Where there was fear, ignorance or ruin, there is now confidence, wisdom, happiness**[17]** and beauty. That is why the player returns to the initial island but now it is in its maximum splendor, life reigns, everything is cognition. The light, the knowledge has been achieved and finally the player can leave the cave and discover the real world. At last her eyes are ready.

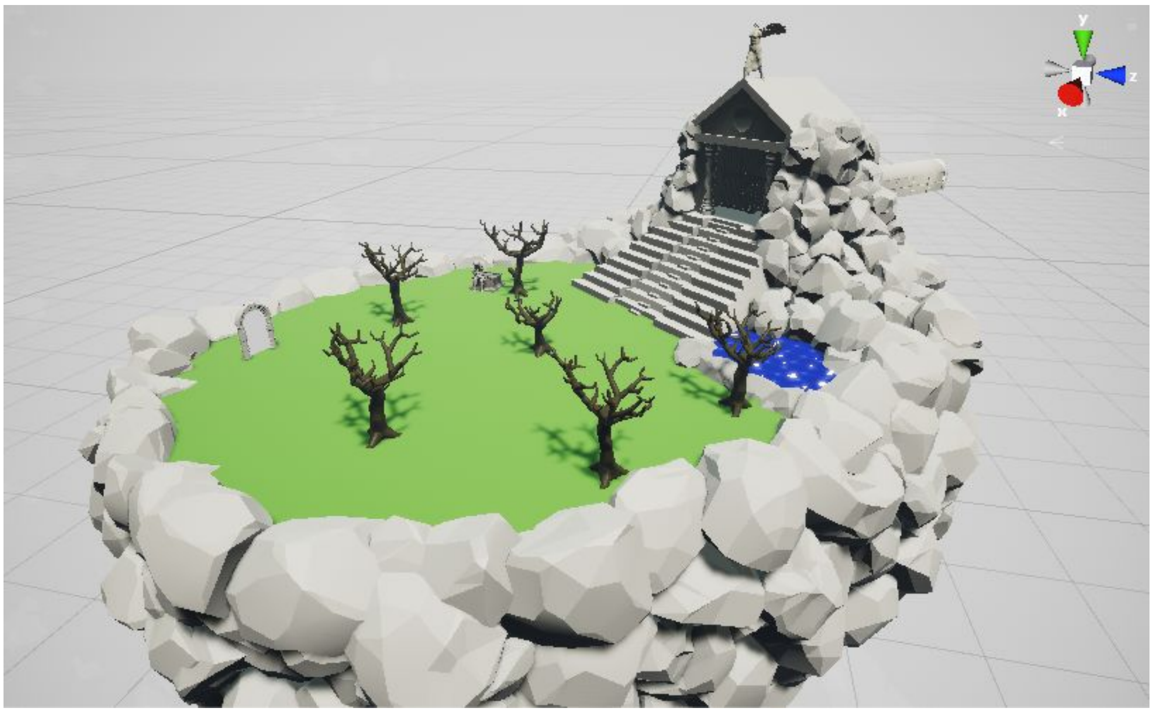


Fig 18. *Platonic Island.*

4.2.6 Lacanian ending*(Fig. 19)*

However, that is not the reality. The reality, crude and sad, is the Lacanian interpretation of the allegory of the cave in which it is exposed that the light, the real world, is nothing more than the shadow of a shadow and therefore the player is plunged into darkness when she realizes that this journey that has torn her from her comfort zone, has only served to make her even more aware of the great amount of things she does not know. The promise, it turns out, is a sham. And so it's back to the original island, only this time even more devastated than at the start of the game. Where there was beauty, there is now decrepitude, and where life reigned, there is only death.

Notwithstanding, this plot twist is not at all unfair to the player as it gives her the possibility to escape from that infinite loop in which she is immersed through the final decision. First of all, it is necessary to contextualize that, when reaching the Lacanian end, the player will hear in the distance, the cries of the statue island, added to her own, imploring them to stop. On this island, the player will be offered a button that will be directly connected to the puzzle of the statues and it will be the moment when the player will understand that she has to help herself in order to advance. By pressing the button, the cages will be opened and life will take its course; however, this will be perpetuating the loop and leaving the player locked in it. It is here that the concept of inaction is finally exploited, allowing the player to make the decision of not pressing the button, thus condemning her past self but allowing her to break the loop in which her present self was trapped. The game ends in the same line in which it has been developed: decisions have their consequences and they have to be faced.

To be faithful and help us as they helped us or to burn our old self to rise from the ashes, like a phoenix?

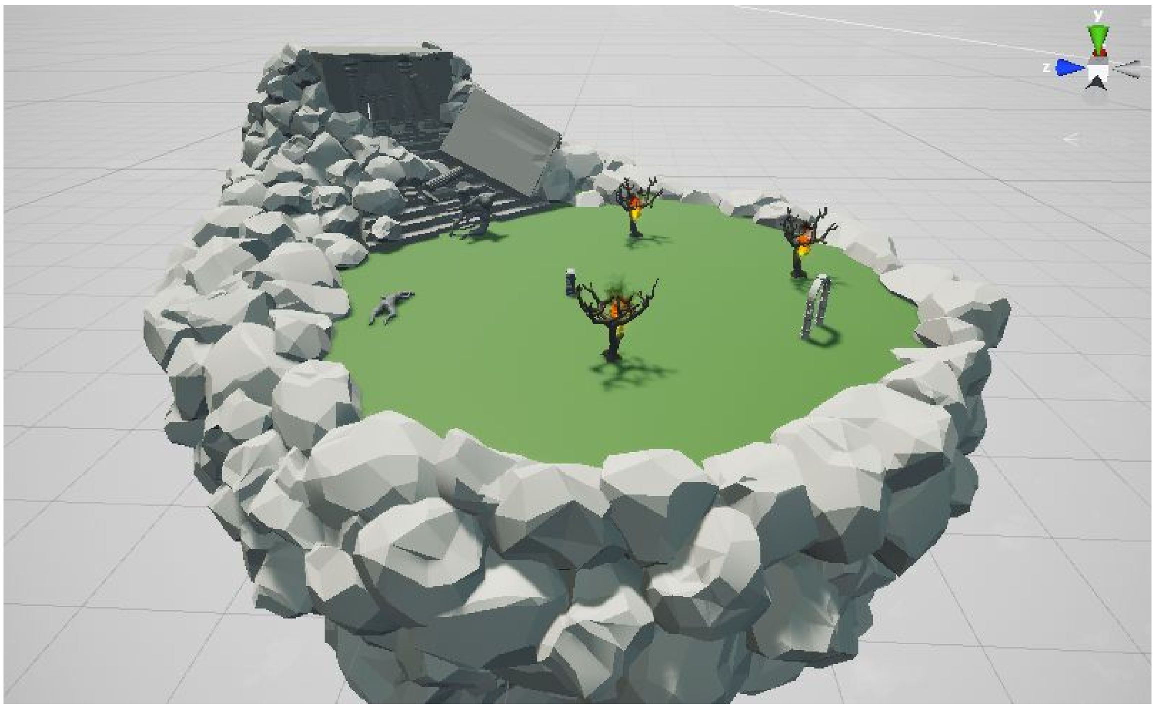


Fig 19. *Lacanian Island.*

4.3 Characters: Interpretation, immersion-identification and incorporation[18]

This has been a recurrent theme throughout our research in the narrative section during the degree. Such is our interest in this particular subject that we have carried out studies on it (practices of Hypermedia Narrative and Analysis of Video Games and Contemporary Audiovisual Models) as well as talks (Sala Caligari). This is why we have dedicated a section to it.

In our video game, there is no tangible figure of any character. Not even the main character, who could be said to be driven by the player. This is due to the different relationships that exist between the game and the player, which we will now describe.

- **Interpretation**: this is one of the most common relationships in the current model of video games, especially when there is a structured and very well-defined story, with little freedom and clear limits set by the designers. In this case the user plays a character, a role as if it were a theatre play. Some examples could be given in video games such as *God of War*(Santa Monica, 2005) or *Spec Ops: The Line*(YAGER Development, 2012).
- **Immersion-identification**: this is a relationship that is gaining strength in recent years and that occurs when the player is able to identify with her avatar, usually through customization. In addition, the player is usually allowed more freedom when it comes to living the playable experience, strengthening the connection she feels with her avatar but always having a clear differentiation between the two. Some examples of video games that exploit this feature are *Fallout*(Bethesda, 1997) or *Dark Souls*(From Software, 2011).
- **Incorporation**: this is a very recent model and still not very widespread throughout the sector. In this relationship, it is intended to blur the line between the player and the avatar. To do this, the player must not play a role or identify

with a representation in the game, but must be the one who incorporates herself into the game, directly and without intermediaries. This kind of games are able to connect to the players at a maximum level since they are an essential and direct part of it. For this reason, they also allow introspection and alterbiography to a great extent, since there is no protagonist character as such, but rather the player's figure is developed as a character, thus also accessing a state of meta-game and self-awareness. Due to all these characteristics, this is the line we follow in our videogame as well as other titles have done before, such as *The Witness* or *The Beginner's Guide*.

Obviously, none of these categories are comparable with each other as they represent very different models that have different claims. Moreover, when talking about the player's level of connection, this can be done from very different points of view taking into account many factors. That is why it is necessary not to demonize the interpretation, being able to forge a powerful relationship between the player and a character based on empathy and not on identification, nor to glorify the incorporation, having several samples of games that force it to create a powerful link with the player, without any success.

CHAPTER 5: ARTISTIC DESIGN

First of all it is necessary to point out that both this part regarding artistic design and the section that could be included in the previous chapter referring to "Gameworld and environment storytelling" will be dealt with by my colleague, Francesc Xavier, who will seek to establish a good relationship between the game and the player through the visual. He will try to develop a coherent world within its own rules that welcomes the player from the aesthetic point of view and that fits the proposed mechanics and narratives.

In this way we will be able to generate an aesthetic that goes hand in hand with the narrative and feeds back into it. In this sense, the concept of worldbuilding, which is the main one around which his work revolves, is essential to obtain a world that feels real and that captures the player's attention in a visual way, incorporating her into the action.

Francesc Xavier will develop 3D assets that help to create joint and individual narratives that are still real but add some dreamlike feeling to the game, providing a safe place for the player to feel, explore and experience. These assets will cover a very wide arch, ranging from inorganic and artificial forms such as portals or ruined monuments from the Hellenistic period to more organic forms such as those that make up nature or even statues of human beings that will have a great involvement in the narrative design.

Besides the 3D modeling, he will also dedicate a great part of his artistic effort to the implementation of shaders that will give the world a more dynamic aesthetic and closer to the standards that players have but without getting close to hyperrealism since we want to maintain a relatively minimalist style that low-poly can bring us.

The sound design part will be done together, where we will try to enhance what the game offers, using natural sounds that do not interrupt but accompany, thinking about the insertion of the player in the game and her experience and makes it "disturbed" only in the case of audio tapes, which will be recorded by professional dubbing actors.

This is a value to bear in mind since it is not usual to have professional dubbing unless a large-scale production is involved; on the other hand, it is important to point out that, although our first intention was to produce a OST composed by Francesc Xavier, in the end this has not been possible due to lack of time; thus, the sound of the video game has been designed by us but composed by third parties - who have assigned the rights of their intellectual property products.

CHAPTER 6: FUNCTIONAL AND TECHNICAL SPECIFICATIONS

Within the functional part and the technical specifications, which will be my main field of work, there are a wide range of tools that will transform a conceptual idea or a set of assets into a fully functional game that will be enjoyed 100% by players, giving them the feeling of being in front of a professional product that encourages a total immersion, in a complementary way to narrative and art, and that will never be a barrier for the players to immerse themselves in the story we want them to live.

This is a fundamental aspect since the environment related to computers, programming or in general, with the practical functionality of an audiovisual product such as a video game, usually only calls attention, in the eyes of the general public, when it does not work accurately. This is intended to imply that, in most cases, any average user of an audiovisual product has grown up being able to discern the artistic or even the narrative section of the works she has consumed, essentially because these are fields in which one can go deeper through popular knowledge, even without being an expert; however, this does not happen in the same way when we refer to the computer/programming section of an audiovisual work. The vast majority of the public does not fully understand how to make a video game works, and that is why it is more complicated to appreciate the effort behind it. For this basic reason, we want to offer the player a transparent experience in which the technical section never blocks this experience and which allows her to enhance it.

In addition, in this chapter, I will focus on the implementation - leaving the development of the problems found and additional information for annex 12.2 - of the most extraordinary mechanics, that is, those farthest from the academic field, learned in the university, taking it for granted and not dedicating more time than necessary given its simplicity.

6.1 Requirements and recommendations

In this project, due to the essence of research and the non-commercialization of it, there are a series of essential requirements and others recommended for players. However, this is not so far from the videogame sector, having launched to date many titles uniplatform and unconditional needs of hardware (graphics card, CPU ...) or software (without backward compatibility).

In this game these needs basically boil down to:

- ❖ **Operating System**: it is only compatible with Windows 10 due to the use of Unity's speech recognition library which will be discussed later in section 12.2.3
- ❖ **Microphone**: the device from which the game is played, must have audio input through a built-in microphone or a connected external one. This, as the reader can imagine, is a sine qua non since voice recognition and sound intensity are core mechanics.

There are also some recommendations as they are:

- ❖ **Headphones**: for a more effective immersion as they improve the quality of the sound as well as the identification of the sound source through 3D sounds, usually switching from Mono to Stereo.

6.2 Tools

To carry out this project, we will use the Unity 3D game engine, in its version 2018.4.15f1 - which is LTS (Long Term Support) and therefore ensures its maintenance throughout the development of the project. In addition, we will use the High Definition RP Template that will allow us to use the Shader Graph that will be vital for the performance of the work by my colleague, Francesc Xavier. In addition, we will also use the Visual Studio Code application as a C# code editor. Other software used has been Illustrator and Photoshop for 2D art design, 3DS Max and Zbrush for modeling and sculpting or Audacity for sound design.

6.3 Player movement and picking up objects.

For the camera we will use the tool that Unity provides us for easy handling in first person games, called First Person Controller. In this way, we are allowed to place the camera at the player's eye level and to always have her position under control in order to determine, if necessary, where the player is looking at. This is one of the easiest ways to manage if the player is looking at an object with a specific tag through the throwing of a RayCast. This way of controlling the point/look/act event is one of the most optimal (in the case that it is only checked when an input is received and not constantly) together with the native function of Unity OnMouseDown; furthermore, it is necessary to take into account that in both situations the existence of a Collider in the receiving object will be required. This functionality will be essential when manipulating objects, playing with their physics and their hierarchy to give the feeling that the player is moving them freely. It will also be essential with some other uses such as turning off the Sun on the time island (if it is reached before the execution takes place), using a physical Sun object without mesh renderer located between the player's vision plane and the visual Sun plane (which is part of the skybox) and which is programmed to detect if the player clicks on it, thus changing the daytime scene to a nighttime one based on the colors of the shader of the procedural skybox.

6.4 Portals[19]

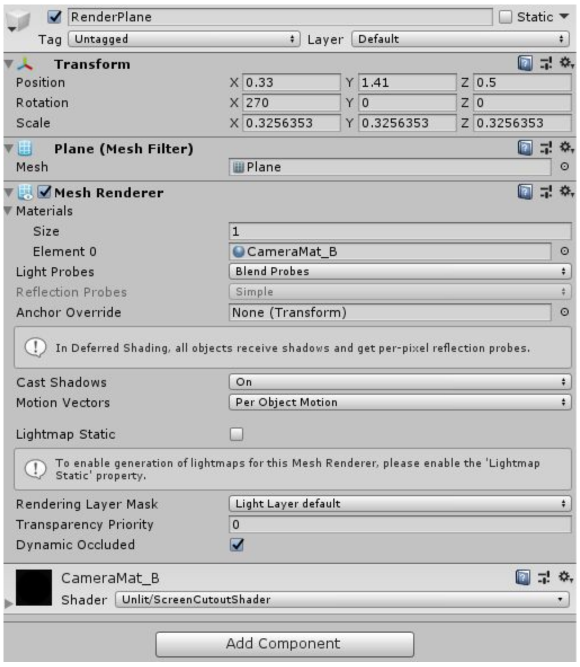
The portals make it possible to connect areas that are spatially separated and to envision what is on the other side through them; they consist essentially of 3 parts:

The **camera** points to the place where players want to move, Portal B, being the Portal A the one we will use to move; that way, applying a script on the camera, which is able to detect the look of the character (depending on the position of the player and the one of the Portal B), the position of the same one can be changed and that way it is given the sensation of seeing another scene through the portal, avoiding this way to be facing a fixed image applied as texture.

Another fundamental part of this illusion is that of the **plane of vision***(Fig. 20)*, since a modified texture is applied to it by means of a shader. This shader is the one that finally allows to apply the image obtained from the camera correctly and that is modified according to the movement of the player with respect to the portal. Thus, through an image obtained from the place the player wants to move by means of a camera, a script to update the movement in it and a texture generated procedurally through a shader, the illusion that a static image would cause is avoided.

The last element of this triad consists of a **teleportation collider***(Fig. 21)* that is the culmination of the illusion, allowing the player to move to the place seen through the portal when entering it. Its operation is very smooth and makes a very good impression, being imperceptible the moment in which the "teleportation" takes place and producing a perfect continuation of the movement.

On the other hand and finishing this specific point, it should be noted that the collision plane has nothing to do with the rendering plane and therefore, I will be able to treat them independently, being able to have the destination image without the possibility of traveling to it, as it is the case of the portals that have not been activated yet or those that have already been crossed, which, as a result of a design and narrative decision, do not allow to return to the place of origin.



RenderPlane — Static
Tag: Untagged — Layer: Default

Transform

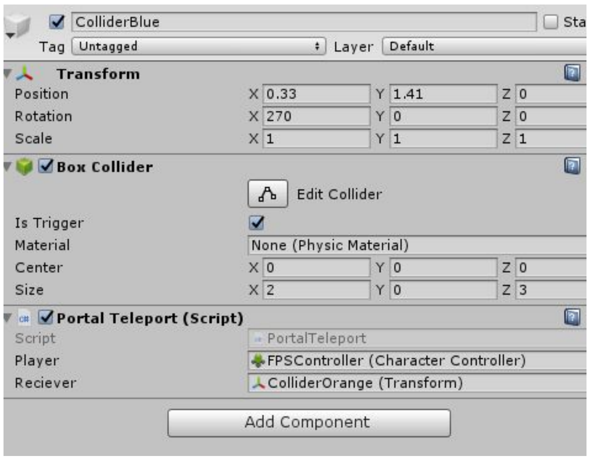
	X	Y	Z
Position	0.33	1.41	0.5
Rotation	270	0	0
Scale	0.3256353	0.3256353	0.3256353

Plane (Mesh Filter)
Mesh: Plane

Mesh Renderer
Materials
Size: 1
Element 0: CameraMat_B
Light Probes: Blend Probes
Reflection Probes: Simple
Anchor Override: None (Transform)
In Deferred Shading, all objects receive shadows and get per-pixel reflection probes.
Cast Shadows: On
Motion Vectors: Per Object Motion
Lightmap Static: ☐
To enable generation of lightmaps for this Mesh Renderer, please enable the 'Lightmap Static' property.
Rendering Layer Mask: Light Layer default
Transparency Priority: 0
Dynamic Occluded: ✓

CameraMat_B
Shader: Unlit/ScreenCutoutShader

Add Component



ColliderBlue — Sta
Tag: Untagged — Layer: Default

Transform

	X	Y	Z
Position	0.33	1.41	0
Rotation	270	0	0
Scale	1	1	1

Box Collider
Edit Collider
Is Trigger: ✓
Material: None (Physic Material)
Center: X 0, Y 0, Z 0
Size: X 2, Y 0, Z 3

Portal Teleport (Script)
Script: PortalTeleport
Player: FPSController (Character Controller)
Reciever: ColliderOrange (Transform)

Add Component

Fig 20. *Render Plane of vision in the Portal.*

Fig 21. *Collider Plane in the Portal.*

6.5 Dialogue and subtitles system*(Fig. 23)*

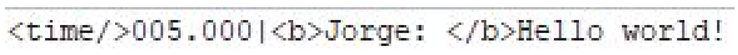
This is one of the closest tools to a large production of a AAA video game. Usually during the degree, due to the lack of time, the semester projects tend to have a very small scale; this means that they are composed of a few scenes (maybe 3 or 4 at most) of great simplicity, few lines of dialogue, few levels, few classes...

Nevertheless, when talking about a bigger project, that is, a professional production, the scalability and modularization of everything that is done must be taken into account, since the aim is to avoid at all costs the dependence within the code. That is to say, that everything can work in a relatively generic way to be reused but at the same time fulfill a very concrete function to be independent from any other fragment of the project, thus facilitating fixes or future scalability (more languages, more dialogue lines…). It is in this line that the dialogue systems are made.

As I said before, in the projects we have done throughout the career, we have already made dialogue systems, but in a very rudimentary way, since these were inserted "manually" in the code, hindering its reusability. This has never been a big problem since we were dealing with accessory and complementary dialogues that had no special relevance within the videogame. However, in a First Person Explorer, one of the immediate ways to develop the story is the impulse of it through the monologue of a narrator; although in this game there is no narrator as such -instead it has audio pieces distributed in tapes-, from the technical point of view, we are facing the same problem.

Fortunately, we will be able to count on the collaboration of professional actors for the realization of these audios and that is why it might seem that the system of dialogues (subtitles in this case) may be less necessary; nothing could be further from the truth; one of the main challenges of a good production is that it be as inclusive as possible in the face of any type of functional diversity. Therefore, despite not having enough time to include closed captions (text display on a TV, video display or other means to provide interpretative or additional information), concerning for example the sound of shouting or door locks, we established the basis for a universal subtitling system for possible future development.

The main problem of such a wide system of subtitles lies, essentially, in the huge amount of dialogue lines*(Fig. 22)* in our game, despite its short duration, and that's why we consider necessary a sustainable and scalable structure like the one shown to us, in a theoretical way, in the Hypermedia Narrative and Video Game Analysis course and that works through a text document (or a spreadsheet) where the dialogue lines are located, the moment they have to be shown and even triggering events. In this way, the people who modify the dialogues (which in this case are ourselves but it is not common that the narrative designer be the same person as the programmer) can access them without having to program anything at all; in fact, they do not even have to install any software; it is as simple as a text editor (.txt).

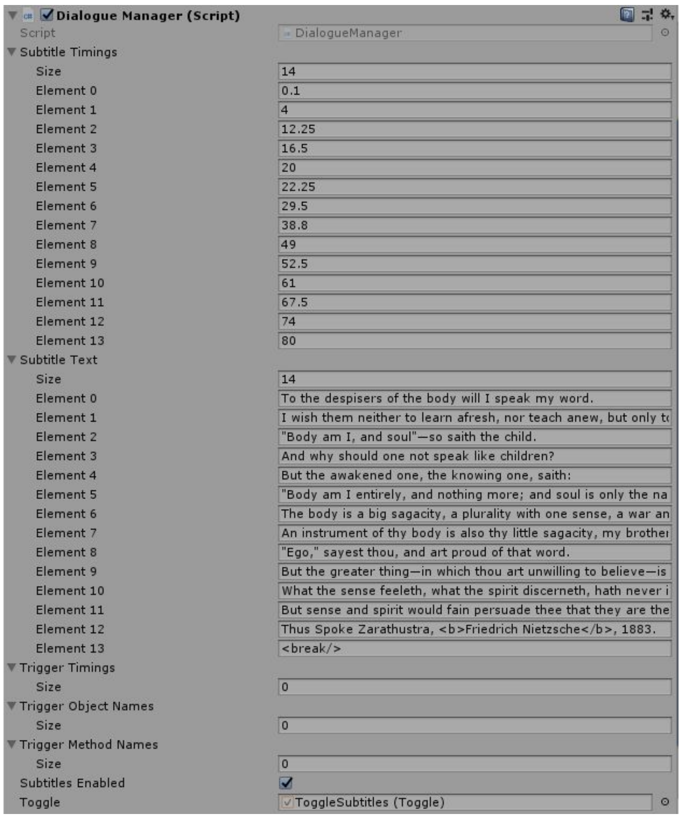


```
<time/>005.000|<b>Jorge: </b>Hello world!
```

Fig 22. *Basic line of dialogue. Extension in section 12.2.2.*

In this system, it is essential to use sub-strings to divide these according to the elements we encounter: "<time/>" indicates that time comes next, "|" indicates that object and function come next if followed by "<trigger/>" or a line of dialogue if not.

It is also very important to note that a conversion from string time to float must be performed to manage the numerical values and also to take into account their formatting to avoid syntax problems between periods and commas. In addition, the subtitles are generated automatically thanks to the tools offered by the GUI in Unity, allowing us to choose, through the GUIStyle the font color, size, scale with respect to the screen, alignment...
And finally, this entire system is controlled by a Boolean variable that persistently indicates whether the user wants subtitles on or off. This is done through the Unity PlayerPrefs tool (which doesn't implement a boolean variable as such and therefore has to use an integer, 0 or 1 and a ternary operator). Thus, this option is saved throughout the game, between menus and even between different executions.



Dialogue Manager (Script)	
Script	DialogueManager
Subtitle Timings	
Size	14
Element 0	0.1
Element 1	4
Element 2	12.25
Element 3	16.5
Element 4	20
Element 5	22.25
Element 6	29.5
Element 7	38.8
Element 8	49
Element 9	52.5
Element 10	61
Element 11	67.5
Element 12	74
Element 13	80
Subtitle Text	
Size	14
Element 0	To the despisers of the body will I speak my word.
Element 1	I wish them neither to learn afresh, nor teach anew, but only t
Element 2	"Body am I, and soul"—so saith the child.
Element 3	And why should one not speak like children?
Element 4	But the awakened one, the knowing one, saith:
Element 5	"Body am I entirely, and nothing more; and soul is only the na
Element 6	The body is a big sagacity, a plurality with one sense, a war an
Element 7	An instrument of thy body is also thy little sagacity, my brother
Element 8	"Ego," sayest thou, and art proud of that word.
Element 9	But the greater thing—in which thou art unwilling to believe—is
Element 10	What the sense feeleth, what the spirit discerneth, hath never i
Element 11	But sense and spirit would fain persuade thee that they are the
Element 12	Thus Spoke Zarathustra, <b>Friedrich Nietzsche</b>, 1883.
Element 13	<break/>
Trigger Timings	
Size	0
Trigger Object Names	
Size	0
Trigger Method Names	
Size	0
Subtitles Enabled	✓
Toggle	ToggleSubtitles (Toggle)

Fig 23. *Dialogue Manager in the Editor. Thus Spoke Zarathustra.*

6.6 Voice Recognition and sound intensity*(Fig. 24)*

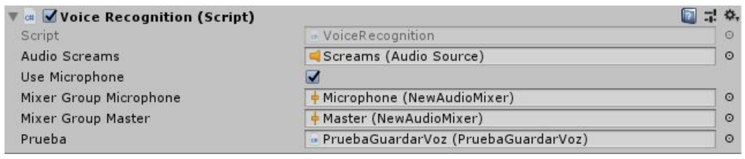
In the voice input part, it could be found a really new mechanic in the video game environment - since it is used in chatbots but with a much less ludicrous sense - as is the case of voice recognition and sound intensity. From a technical point of view it is complex and laborious but, fortunately, there are tools that can help us in this work as is the case of the Windows library in Unity known as Speech. This is one of the best ways to add to the video game the recognition of certain words and is also one of the most scalable due to the great versatility of languages available.

In addition, the video game has the recognition of the microphone and its sound intensity; this is executed essentially through the modification of an AudioSource and an AudioMixer*(Fig. 25)* that prevents the echo effect and that, through a script, detects the peaks of the waves entering through the microphone and is able to calculate the average value in decibels of this sound, depending on the desired sensitivity value.

A small problem that arose in reference to these mechanics is the delay time that exists between saying a word and the recognition by the program; although it is a time of just a few tenths of a second (the time necessary for the program to understand that it has stopped speaking and that therefore, the required word is being said) it was very relevant since it prevented the sound intensity from being captured at the moment in which the word was recognized, essentially because at that moment, the user would have stopped speaking and therefore, the sound intensity would be 0. Therefore I had to make a periodic record of the collected sound intensities (which has a continuous spectrum, not discrete) to determine periodically which was the sound with the highest intensity captured and after a while, reset it to 0 again. In this way, when the desired word is recognised, it is observed if there is a record of a peak of sound intensity greater than that established; if not, nothing is done because the desired word has been said too low; if positive, the pre-established action is executed because the word has been said with the appropriate sound intensity.

I also had to deal with a big problem concerning the load of the microphone that caused a delay equal to the time that the AudioClip lasts in which the sound of the microphone is recorded and played, thus obtaining always incorrect inputs. This made the system useless because there was no correlation between the input and what was happening in the gameplay, as it was 10 seconds out of phase. However, this problem and its solution will be explained in depth in section 12.2.3 of the annex.

Finally, I implemented a visual feedback system that allowed the player to see which statues are listeners and which are not. This system is mainly based on the collection of the sound intensity by the Voice Recognition, obtaining an array of power size 2 (in our case, 256) that adds up each of the audio blocks collected by the AudioSource that records and plays at the same time the microphone input. Once the sum of these blocks is obtained, it is multiplied by a sensitivity value and this allows us to determine an approximation of the sound intensity, called loudness. This loudness value is the one we use, dividing it by another sensitivity value so that we have a wide range of values between 0 and 1 within the percentage variable. This variable is the one we use to make a lerp in the material and change its properties according to the sound intensity that is collected.



Voice Recognition (Script)

Script	VoiceRecognition
Audio Screams	Screams (Audio Source)
Use Microphone	✓
Mixer Group Microphone	Microphone (NewAudioMixer)
Mixer Group Master	Master (NewAudioMixer)
Prueba	PruebaGuardarVoz (PruebaGuardarVoz)

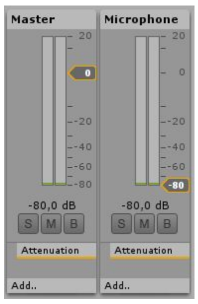


Fig 24. *Voice Recognition Script in the Editor.*

Fig 25. *Microphone Audio Mixer muted.*

6.7 Real time system

There is an area of the game in which a digital clock is represented, in 24h format, which shows the real time of the system (the PC on which the video game is running). This is done through a canvas located in the World Space, to be able to place it on top of our clock object. This canvas, in turn, is composed of a Panel and this one is made up of a Text. This Text will be the one that will be modified with the current time through **System.DateTime** library; however, it must be taken into account some important aspects regarding the format and that are based on:

- In order to deal with the 24h format instead of AM/PM, it is necessary to use the ToString("HH:mm") method.
- The previous method also allows to keep only the hour and minutes, disregarding other values that are not interesting in this context such as the year, month, day or seconds.

It should also be noted that, at a certain point in the game, the clock will stop and no longer update its time with the system's time. At this point, the player must solve the puzzle by setting the clock to the correct time, thus opening the portal to the next island. Therefore, it is necessary to take into account that:

❖ The portal will only be open as long as the time on the game clock matches the system time.
❖ When setting the clock, the value of hours or minutes is increased or decreased by one unit; however, an object of type **System.DateTime** does not accept hour values lower than 0 or higher than 23 or minute values lower than 0 or higher than 59; therefore, some control must be exercised over the change of time so that it never results in the frustrated creation of a **System.DateTime** object due to the mismatch of parameters entered in the constructor.

6.8 Taking photos

This is one of the mechanics with the most playable potential and with the most challenges with respect to implementation - along with voice recognition and the subtitle system.

The way the player grabs the camera is the same as during the rest of the gameplay, by left-clicking on it. Once the player has the camera equipped, she can use the right button to focus and while doing so, the left button to take a picture. The focusing process is really very simple and it is done essentially to give the player visual feedback by means of a different canvas and a small animation that simulates the approach of the ocular to the player's eyes. It is also used to remove any element of the scene view (camera, photo, reticle) so that only the environment to be photographed remains. Once the focus is on and the photo is taken, the complexity of the programming begins.

Pressing the button to take a picture will call a coroutine that first creates a string that will serve as the name of the image to be stored; since it is a temporary name and this image will not be saved permanently, it is used the date of the moment the picture is taken, as in the old cameras, thus avoiding any possible coincidence in the name of the files, although this will not really be a problem since, when the photographing process is finished, this file is deleted from the persistent data, in order to avoid an undesired collapse of the memory in case many pictures were taken.

Once there is a name for the image to be extracted and the path where it will be saved, as well as the termination of the file (in this case .png), there is the complete path and on this, the screen capture function will be called. Before continuing, a little time is left (around 0.1 seconds) to avoid visual interferences and continue converting that screenshot (.png image) located in a system folder, into a sprite that can be used in runtime.

In order to do this, the byte array that forms the file is obtained, a new texture is created and the bytes are loaded on it. Later, this texture is converted into a sprite so that it can be loaded on a photo object that has a Sprite Renderer component. And once the sprite has been created, we apply it over the prefab (which has been instantiated and placed in the right place, in the player's left hand space, if it was the first time a photo was taken) obtaining an instant Polaroid*(Fig. 26)*.

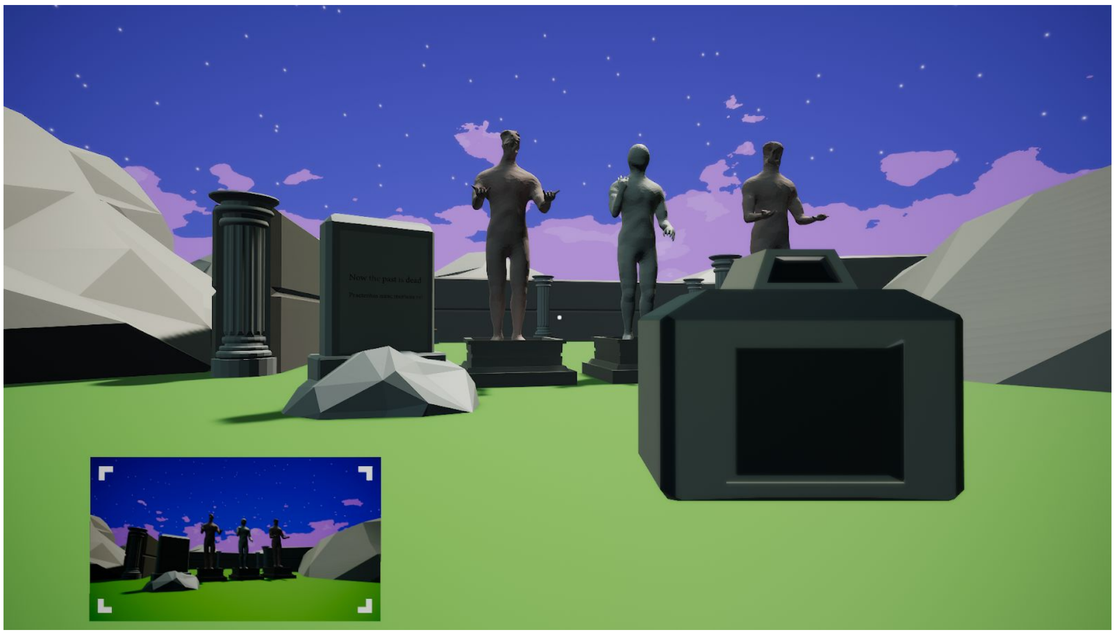


Fig 26. *Photo taken from the statues.*

6.9 Mazes' random generation[20]*(Fig. 27)*

One of the strong points at the programming level is the random generation of mazes through the Hunt and Kill algorithm. This is an algorithm that can also be used for the procedural generation of labyrinths; however, in the context of this video game, this was not of interest to us since, for this mechanics to make sense from a narrative point of view, we need the generated labyrinths to be different in each execution and not to keep any seed-dependent correlation.

An essential characteristic of this algorithm is that it is capable of creating mazes that connect a point A and a point B in a single way, that is, they have a single solution; moreover, its operation is really very basic and intuitive:

1. An initial location is chosen.
2. It starts wandering from that location from square to square, following a pattern based on a seed (if one wants a procedurally generated maze) or a random pattern (if one wants a random maze). This is the step known as KILL because it "removes" boxes from the algorithm when they are visited. There are 4 possible directions (north, south, east and west). This walk continues as long as there are unvisited boxes to reach in the next step. If the current cell does not have any unvisited cells next to it, then it will switch to HUNT.
3. The entire maze is scanned, row by row, until it finds the first unvisited square that has a visited square as its neighbor. If found, this is set as the new starting square.
4. Repeat steps 2 and 3 until no boxes remain unvisited.

The implementation of this algorithm is essentially based on creating a grid in which all the boxes in the maze are composed of 4 walls (forming a sort of cubicles). The size of this grid is completely adaptable as well as the floor or the walls used.

Once this grid is in place, the execution of the Hunt and Kill algorithm starts and as it advances, some of the pre-established walls are removed. In this way, blind paths are built that result in dead ends and simultaneously, a single path is built that connects point A with point B.

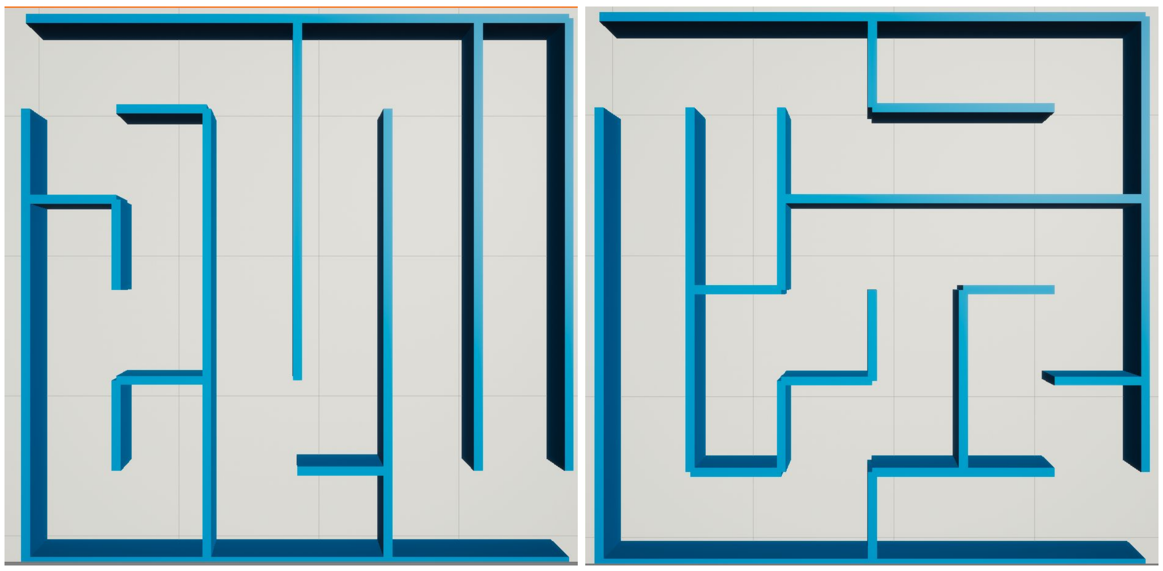


Fig 27. *Examples of randomly generated mazes.*

6.10 Optimization

This videogame requires some very careful optimization features in order to have a good performance because, in the gestation of the project, we established that we wanted to make a seamless videogame in which there were no loading screens to facilitate the immersion of the player and avoid the rupture of the tension. That's why the whole video game is in the same Unity scene and why the passage between islands through the portals is not subject to loading times but is done at runtime. However, this requires a great optimization of the scene, which I will talk about later, and also a powerful initial load as a bridge between the initial menu and the beginning of the game. This bridge is the one we treat as an asynchronous load. We are aware that the scene to be loaded, contains a large amount of elements (modeling, textures, shaders, scripts ...) and as such, requires an initial time needed to load the entire game, but that will be compensated later as it is the only one and does not break the playable experience as it has not yet begun. In order for this load to occur effectively and provide visual feedback to the player, preventing her from restarting the game or thinking it has been frozen, we apply an asynchronous operation. This means that a secondary thread is established in which the entire game starts loading while the main thread continues running. This main thread is essentially a loading screen that visually shows the player the animation of a loading element. The moment the sub-thread signals that it's fully loaded (according to Unity's documentation 1 is the fully loaded value so it will stay at 0.9 when it's completely ready but hasn't been given permission to take control of the main thread yet) is when the effective scene change occurs.

As for the optimization of the scene, we found some serious problems when applying some of the best known techniques such as Occlusion and Frustum Culling*(Fig. 28)*, which is included natively in Unity. However, these problems will be dealt with in section 12.2.6 of the Annex.

Finally, the solution we found for good performance was to apply Levels of Detail (LOD)*(Fig. 29)* to those objects which are seen from far away and to have only those parts loaded that were visible or accessible to the player. In this regard there are two different possible angles:

- ❖ **Meshes**: It's always preferable not to have loaded the models that are not being seen to avoid unnecessary computational load (after all, that's the function of Frustum and Occlusion Culling), but independently, although we do this, it's true that in this videogame it's not a key factor because the models are low poly and therefore don't require a big effort from the CPU and GPU.
- ❖ **Scripts**: It is indeed key to have an exhaustive control of the local scripts - of each one of the islands - and global to avoid making unnecessary operations according to the moment of the game. For this reason, the activation and deactivation of objects with attached scripts is constantly managed according to the moment of the game in which the player is located, thus making it impossible to overload the CPU with elements with which one cannot even interact. This has a special relevance in the rendering of the portals since the constant modification of the procedural textures entails a great consumption of CPU.

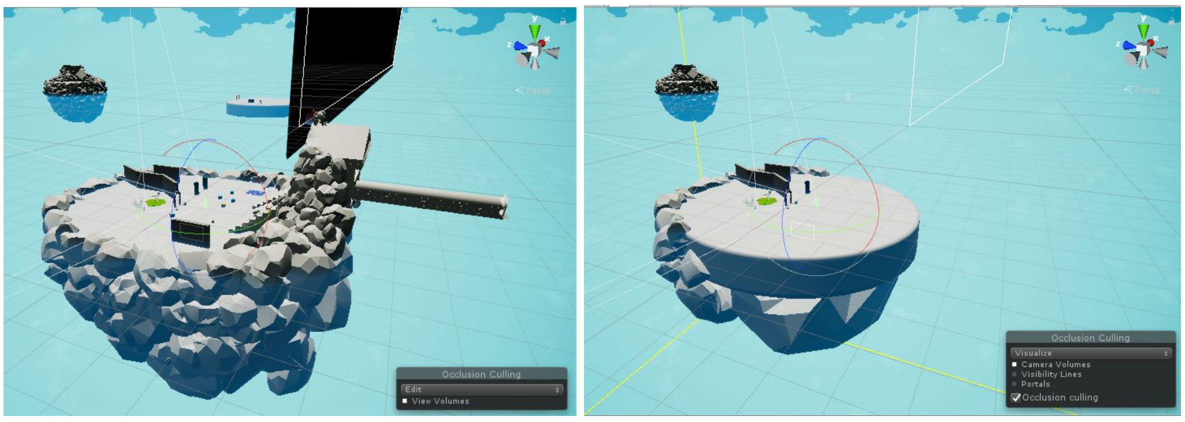
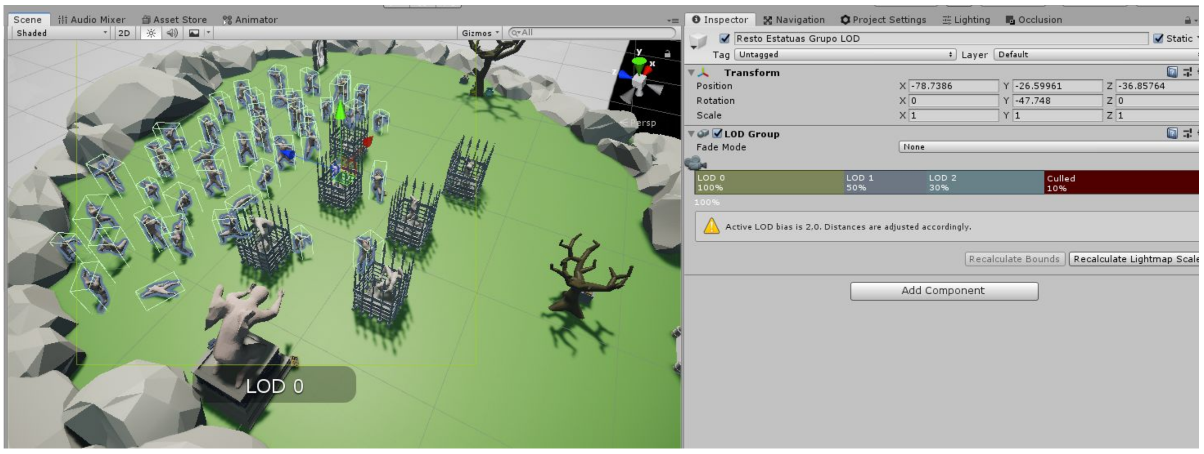


Fig 28. *Initial island with and without culling applied.*



Resto Estatuas Grupo LOD ✓ Static
Tag Untagged Layer Default
Transform
Position X -78.7386 Y -26.59961 Z -36.85764
Rotation X 0 Y -47.748 Z 0
Scale X 1 Y 1 Z 1
✓ LOD Group
Fade Mode None
LOD 0 100% | LOD 1 50% | LOD 2 30% | Culled 10%
100%
Active LOD bias is 2,0. Distances are adjusted accordingly.
Recalculate Bounds | Recalculate Lightmap Scale
Add Component

Fig 29. *Object with LOD applied.*

CHAPTER 7: PROJECT MONITORING AND MANAGEMENT

Fortunately, communication throughout the project has been very direct and fluid in all senses between Francesc Xavier and me. Since we are flatmates, we have been able to coordinate at all times to be in synchrony with respect to the project phase. Sadly, as a result of the quarantine, we returned to our respective cities of origin, but this has not been a problem as we have remained in continuous contact, through consultations and daily meetings - especially in the conceptual design phase. Once this was defined, although the work has taken on a more individual tone (he in the art branch and me in the programming branch) we have kept on coordinating frequently to give feedback to our work, making a thorough mutual follow-up.

The main tool we have used to communicate has been Discord, while to share work and version control, we have used Google Drive (documents, images, papers, Excel with the schedule...) and GitHub.

On the other hand, we have also maintained a constant and fluid communication with Marta Martín, our tutor, who has advised us in the earliest phases of the project - in which it was necessary to establish the design bases - and also throughout the development of the same, giving us feedback on the work shown and providing us with new ideas.

Finally, we have made small modifications with respect to the initial planning, essentially due to the variation, elimination or addition of certain mechanics depending on the evolution of the conceptual design of the video game. However, the time planning has not suffered too many changes. Although it is true that we have compensated hours, the initial calculation was quite accurate and we have only had to devote a little more time than planned to some activities not covered as testing and debugging -crucial to avoid bugs and provide the project with a more professional feel- as well as others to which we did not give as much importance as they deserve, as is the case of the final report or dubbing.

With this, the total scope of the project has reached 350 hours, just from my side, obtaining approximately 17% of overtime. These hours over the initial calculation have been completely necessary to obtain a proper finish, using 15 hours more than the pre-established for the creation and implementation of the dialogues in the game -taking into account the meetings with the actors, recording and subsequent mastering of the audios as well as their final integration in the game-; 10 hours more than the estimated for the final report and between 20 and 30 hours for testing and debugging, in each of the implemented mechanics as well as in the general flow and the final gameplay.

CHAPTER 8: RESULTS

The end result is essentially a vertical slice, which is defined as "a portion of a game which acts as a proof of concept for stakeholders before they agree to fund the rest." Although in this case, the expected investment is not exactly economic. Although a vertical slice is different from a prototype, our project occupies an intermediate space between the two. It fulfills the function of a vertical slice since it integrates the most important components of the project but it only represents a portion of the cake, being expandable in the future to the whole pie; however at the same time, it also has the component of final game sensation that is required from a prototype, just having to develop the bases already implanted. This is a very important feature of a vertical slice and it is that, even though only 10% of the game is shown, it does not imply that 90% of the effort is still missing from the developers; generally 10% of the game can imply 70% of effort since the initial phases are the ones that require more work to lay the foundations.

This project has an approximation of 30-45 minutes of gameplay -which is much more than we initially expected to be around the quarter hour- in which a total of 6 islands can be explored (three times more than we had planned at the beginning). It also features 13 minutes of monologues and dialogues in more than 40 audio pieces written by ourselves and recorded under our supervision, with the invaluable help of three professional dubbing actors.

It could be defined, therefore, as a complete and functional work, although with capacity for future evolution, in which we have designed the narrative ourselves, most of the mechanics, the sound and the art, counting on a great display of original content.

Lastly, a series of interesting links are attached.

- **Trailer: https://tinyurl.com/y9eumb82**
- **Gameplay 1: https://tinyurl.com/yb5lghj7**
- **Gameplay 2: https://tinyurl.com/yckn9jb2**
- **Executable link: https://tinyurl.com/ydh4478b**
- **Repository link: https://github.com/jfez/TFG**

CHAPTER 9: CONCLUSIONS

Firstly, I must express, in spite of the exceptional situation, our satisfaction with the treatment we have received from our tutor, Marta Martín, counting on a full disposition in person -in the pre-confinement stage- and online, through Google Meet -during the alarm state and until the end of the academic year.

In summary, we are extremely happy with the final result of our project, which has escalated immensely from the initial planning due to the great enthusiasm we have placed in it and through our commitment and perseverance, thus avoiding accumulation of workload or failure to meet deadlines. As we have commented before, we are talking about a project of an enormous magnitude, superior to any other carried out during these 4 years of career (both in the curricular and extracurricular fields). In addition, we have been able to put all our heart and soul into it, since we have been fortunate enough to deal with a subject that we are passionate about.

Furthermore, we are aware that this will be our presentation card for the job market, as it is the most extensive and eye-catching project we have to our credit; in addition, we have been able to make the most of the knowledge acquired during the degree, extrapolating a large part of it from theory to practice and expanding it enormously. We place special emphasis on the latter since, if there is one thing that characterizes our project, it is exploring areas that the degree cannot reach for a simple matter of time. We refer to some sections, among others, as for example voice recognition and sound intensity, procedural methods -animations (skybox and portals) and algorithms (mazes)-, connecting, editing and saving local system files, development of alterbiographical and emerging narratives as well as optimization of the video game -seen in its theoretical form- applied to practice, the design of mechanics, creation of shaders... In general we are talking about knowledge that goes far beyond what we have learned in college and that requires a great deal of research and deepening but that also requires basic training and essential conceptual foundations without which we could never have developed this work.

We must also proudly add that in our specific case, we have not only gone beyond the basic training learned in the degree, but we have also gone one step beyond the current standard of the video game sector, exploring a field that currently has very few references and implementing some mechanics that, in fact, have no reference today.

This is for us a primordial point of our project since we have managed to get out of our comfort zone and face the unknown, as it is urged to do in a certain game. However, we would have been delighted to have more time to spend on this project, going deeper into the mechanics developed yet only superficially exposed, in order to bring the development of the latter to its peak and make it the bedrock of a sub-genre widely ignored until now.

CHAPTER 10: FUTURE WORK

As we have mentioned, we are in front of a vertical slice/prototype, however, most of the bases of the development are already established so our long term objective is the post university development of the game, as a personal project, with the intention of finishing it and thus obtaining an extended work that can be compared to a video game released by an indie studio.

Besides the emotional component that links us to this work, there are also professional interests behind this future development, knowing that we have a very interesting project with a good foundation and that with a little more time available to invest in its development - from 2 to 4 months - we could obtain a very interesting final product for our portfolio.

The work will essentially consist of the creation of new islands that allow the exploration of themes that have not been able to be included due to the lack of time, design and implementation of new mechanics as well as the introduction of more assets that favour worldbuilding, shaders, particle systems and audio fragments.

Therefore, we do not consider this project to be a full stop, far from it. We believe that it precisely marks the beginning of the rest of our lives and that even though it closes a stage, we must always remember that, as Davey Wreden said in "The Stanley Parable"*(Fig. 30)*:

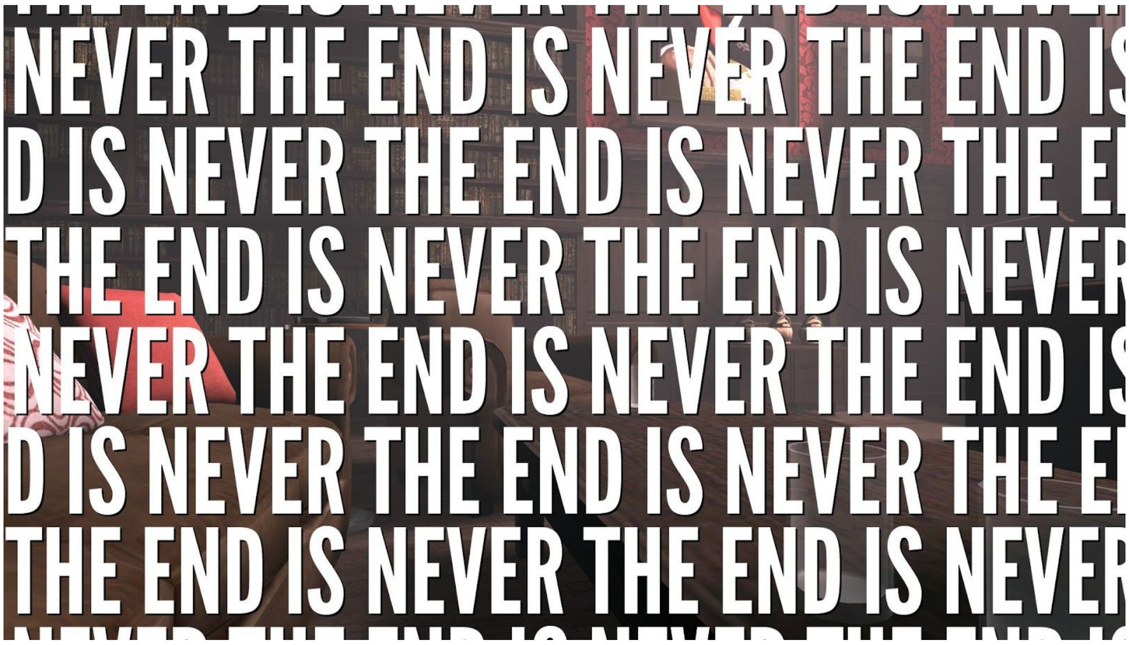

NEVER THE END IS NEVER THE END IS
D IS NEVER THE END IS NEVER THE E
THE END IS NEVER THE END IS NEVER
NEVER THE END IS NEVER THE END IS
D IS NEVER THE END IS NEVER THE E
THE END IS NEVER THE END IS NEVER

Fig 30. *The end is never… From The Stanley Parable.*

CHAPTER 11: REFERENCES

[1] **Calleja, G.** (2011). In-Game: From Immersion to Incorporation, The MIT Press.
Extract available: In-Game

[2] **Jenkins, H.** (2004). Game Design as Narrative Architecture, The MIT Press.
Extract available: Game Design

[3] **Arendt, H.** (2014)**.** "Más allá de la filosofía". Editorial Trotta.

[4] **Navarro-Remesal, V.** (2019). "Pixelated nature: ecocriticism, animals, moral consideration, and degrowth in video games." Logos 26.2: 13-26.

[5] **Sartre, J.P.** (1938). "La náusea". Editorial Época (Edition Year 2008).
Extract available: La Náusea

[6] **Plato** (370 b.C.) "La República". Zeuk Media (Edition Year: 2020).
Extract available: La República

[7] **Scully-Blaker, R.** (2018). "Stasis and Stillness: Moments of Inaction in Video Games". UC Irvine.
Extract available: Stasis and Stillness

[8] **GameReport** (2018). Dueños del tiempo: Rompiendo el bucle.

[9] **Aarseth, E.** (2014). "Ludology" in The Routledge Companion to Video Game Studies edited by Mark J.P. Wolf and Bernard Perron.
Extract available: Ludology

[10] **Mitchell, A.** (2006). "Hadestown"
Trailer available: Hadestown

[11] **Fernández Gonzalo, J.** (2015). Pixelar a Platón, Micromegas.

[12] **McIntyre, L.** (2018). "Post-Truth". Massachusetts Institute of Technology.
Preview available: Post-Truth

[13] **Navarro-Remesal, V / Gómez, S / Aranda, D / Planells, A. J.** (2015) Game & Play. Diseño y análisis del juego, el jugador y el sistema lúdico. UOC Press-Comunicación.

[14] **Camus, A.** (1942). "El mito de Sísifo". Alianza Editorial (Edition Year: 2012).
Extract available: Sisyphus

[15] **Barr, P.** (2011). "Let's Play: Ancient Greek Punishment".
Game available: Ancient greek punishment

[16] **Ocaña, E.** (1993). "Más allá del nihilismo: Meditaciones sobre Ernst Jünger". Editum. Ediciones de la Universidad de Murcia.

[17] **Larrauri Gómez, M.** (2003). "La felicidad según Spinoza: 5 (Filosofía para profanos)". Tandem Edicions.

[18] **Altozano 'Dayo', J.** (2016). El videojuego a través de David Cage, Héroes de papel.

[19] Portals.
Resource available: Portals in Unity

[20] Random maze generator.
Resource available: Mazes

CHAPTER 12: ANNEXES

In this chapter, a series of documents will be attached that are not essential for reading the report; although they are recommended for the correct understanding of the project in all its aspects.

12.1 Quotes

12.1.1 Opening sequence

- ❖ Dear child, stop working, go play. Forget every rule… There's no fear in a Dream. **Song of myself. Nightwish.**
- ❖ You are master of what you silence and slave to what you speak. **Sigmund Freud.**
- ❖ To know how to speak, you must know how to listen. **Plutarch.**
- ❖ Talk very little about yourself, little about others, a lot about things. **Paolo Mantegazza.**
- ❖ If something is never talked about, it's as if it never happened. **Oscar Wilde.**
- ❖ Every tongue is a temple in which the soul of the speaker is enclosed. **Oliver Wendell Holmes.**
- ❖ "Most people do not really want freedom, because freedom involves responsibility, and most people are frightened of responsibility." **Sigmund Freud.**

12.1.2 Lighted corridor

- ❖ That the same fears that blind us are windows overflowing with light.
- ❖ Those who do not move, do not notice their chains. **Rosa Luxemburgo.**
- ❖ And when the time comes, the slave will be freed from his chains and will be free to leave his prison. But... can you call your home a prison? The only place you know, your comfort zone... No...
- ❖ It is knowledge that is the true pain because once you are aware of that which exists but cannot be reached, life is condemned to a permanent state of anguish.
- ❖ Insanity is doing the same thing over and over again expecting different results. **Unknown**.
- ❖ The goal can never be reached when the right path is not taken.
- ❖ The light... the knowledge... the unknown. Not being prepared is just one more part of the preparation.
- ❖ Everything that made up reality is only a mirage, a shadow; but before looking at the light, the eyes must be prepared.

12.1.3 Initial island

- ❖ Speak so I may get to know you. **Socrates**.
- ❖ Born from silence, silence full of it. A perfect concert my best friend. So much to live for, so much to die for... If only my heart had a home. **Dead Boy's Poem, Nightwish.**
- ❖ What is it you dream of, child of mine? Never met a kinder heart than yours. Let it bleed, leave a footprint on every island you see. **Ghost River, Nightwish.**
- ❖ Thus the serpent said "For God knows that when you eat from it your eyes will be opened, and you will be like God, knowing good and evil." **Genesis 3:5.**
- ❖ The path to follow is clear, the path of knowledge, the truth... but what exactly is the truth... that which I know with certainty? Or that which no one doubts? How

can I be sure of anything then, if everything can be the fruit of the cruelest deception? Does this question even make sense, or is it just one more of the threads pulled by the evil genius?
- The turtle, swept away by the fierceness of the rushing river and fearing for its life, locked itself in its shell without being aware that the shelter it needed was not there.

12.1.4 Statues island

- Sisyphus's punishment was a straightforward task – rolling a massive boulder up a hill. But just as he approached the top, the rock would roll all the way back down, forcing him to start over …and over, and over, for all eternity. **Greek myth.**
- It's common to be fearful, one cannot face what one is not prepared for. Running away is the smartest approach when the other side is facing a certain death. / Certain death? The only true death is that of uncertainty. Ignorance and anguish will be that which kills you. / I'd rather choose uncertainty with a glimmer of hope than the certitude of defeat. / From certitude one can draw lessons; in uncertainty all there is left to do is to flee, for there is no pain more intense than that of being unable to escape, since what one flees from is within oneself. / It is all too easy to speak from your position of safety, where nothing bad might happen to you. It's me who's playing this game. / You must never forget that it's not just your game, it's ours.
- No matter how hard one wishes, one cannot save everyone; one is only responsible for what one does, not for what one does not do; and as was demonstrated in Coventry, to make an omelette, one has to crack a few eggs. Especially if they never find out who got their hands dirty. We could say that all is fair in love and war.
- Inaction is just one of the other actions available, the action of doing nothing. Choosing inaction is likewise a decision and, as such, the "non" executor must be held responsible as well.
- We suffer not from the events in our lives but from our judgement about them. **Epictetus.**
- Time, a relentless enemy, strikes at the once inert and now eroded rocky shelter as an unmistakable symbol of the past. There is no mercy, no exceptions, just a constant ticking, tick tock, tick..tock…
- Shut them up. I can't stand this any more. Make them stop! Tell them to STOP!
- Tell them to STOP!
- MAKE THEM STOP!
- SHOUT THEM TO STOP!
- STRONGER! ORDER IT TO THEM!

12.1.5 Time island

- It is not the blade that renders the sword mortal, but its irredeemable action. It must be honed, not the weapon, the mind, for the wound does not always heal.
- Everything in life has its consequences; we may not be aware of them or we may be reaching a decision without knowing that it is one, yet that does not stop one from being responsible for it.
- Can anyone be held accountable for the path that has been laid out for them? Destiny is fixed and we are just puppets to the music of strings that never belonged to us. We are insignificant and our decisions are even more so, since they were certainly never ours. And whosoever believes themselves to be the

owner of their destiny and master of their actions, must know that by believing so and acting accordingly, they are only fulfilling that which had already been written.

- There is no way out; it is like a madman in a madhouse; if he says he is mad, everyone will testify to it; and if he tries to deny it, his madness will be further proven.
- Actions, unlike words, do not understand intentions. It is what it is. Objectivity in its purest form. What is triggered by these actions is something beyond control. The only thing that can be done in the face of this is to embrace it stoically and move forward, as the past cannot be changed, only its echo in the present can be modified.
- The world is in the throes of a desperate desire to change the past. Yet, no, the past is unchanging. There is no key to undo our actions and that hurts us as it takes us away from our **Control Z**one.
- And one of the reasons why the past is unreachable is because time flows unidirectionally forward. There's no way to go back or to stop. Time is a river and trying to swim against it will only leave us exhausted and frustrated. The only way to live is to let go.
- Phaeton, son of Helios, riding the sun chariot, rode away and provoked blizzards, came close and scorched the earth. The sun, wounded, cursed the child for taking him away from his kingdom. And Zeus, unyielding, condemned the boy. Thus returning the sun king to his heights, from where he watches over on his errant journey. **Greek myth.**
- One cannot live trapped in the past. Mistakes cannot be erased but one can learn to live with them. The only way not to die in life is to not lose sight of now. The present. This moment.
- Night descends and sleep awaits. Helios, pursued by his sister Eos, makes way for Selene who makes the night shine and with sparkling reflections populates the sky. She and her white-haired horses guide the being on its way through the night until welcoming the new day whose sun blesses the earthly pastures. **Greek myth.**
- "Everywhere I go I find a **poet** has been there before me." **Sigmund Freud.**

12.1.6 Mazes island

- Because sometimes it seems that the mind is the greatest enemy. Blessed and treacherous, it casts darkness and light into every corner. And it is in that moment, locked in the depths of my beaten mind, where I find chimeras wherever I look, when I remember that there is always a way out and I rise to walk, determined.
- Upon this ridge surrounded by cherry trees, accompanied by butterflies, contemplating the meadows of heaven before me, it is when I look back and recall that if something defines us, it is the journey, each step, more confident than the one before it, and that the destination is really only a sigh. Because in the end, "Life can only be understood backwards; but it must be lived forwards." and that titanic problem, just a water drop in the sea.
- In these times we cling to the past, we carry idols of past moments, deaths of instants that we intend to immortalize, equally mortal framed absences. The only opportunity to grasp eternal life. Hoping, still, that they will be the key to take us through the next crossroad.

- Is the one who only sees with the eyes not blind? Or the one who only hears what they want him to hear deaf? Five senses were gifted to us, and we only seem to use half of them; we cannot even discern between truth and illusion.
- I'll tell you where the real road lies: Between your ears, behind your eyes, that is the path to Paradise. Likewise, the road to ruin. **Wait for me II. Hadestown**
- Where are you? Where are you now? -EURYDICE: Orpheus, hold on, hold on tight. It won't be long. Cause the darkest hour of the darkest night comes right before the dawn. **Doubt comes in. Hadestown**

12.1.7 Platonic ending

- Beyond all mortality we are, swinging in the breath of nature. In early air of the dawn of life… A sight to silence the heavens. **Song of Myself. Nightwish**

12.1.8 General gameplay (peals of bells: Tempus Fugit)

- Thus every creature , and of every kind ,the secret joys of sweet coition find: not only man's imperial race
- but they... that wing the liquid air ; or swim the sea, or haunt the desart...
- . . . rush into the flame; for love is Lord of all; and is in all the same.
- While we too far the pleasing path pursue; surveying nature , with too nice a view
- But time is lost , which will never renew. **Virgil. Translation by Rhodes**

12.2 Difficulties faced and additional information on the mechanics implemented

This section is an extension of chapter 6 in which will be explained some of the main difficulties I have encountered when implementing some of the core mechanics of our video game as well as its final resolution, if any, or discarded if not.

12.2.1 Portals

One of the main problems we faced in the implementation of the portals was at the moment when the portal was transferred from zone A to zone B. For the effect to be really achieved, it is necessary to take into account the position coordinates of the portal model (GFX) since, in case it is not aligned with the texture rendering plane, a blurred or obstructed vision is obtained by the columns. In addition, it must be taken into account that, after all, the image is being obtained from a camera located at the destination of the portal, building then the texture procedurally as we mentioned before; this implies that if, at the destination, the camera interferes with some element of the environment, it will also obtain a distorted vision; therefore, this is something to take into account, avoiding that the illusion of the effect is lost.

However, we were able to solve this problem without major inconveniences, rotating the destination GFX 180º at the moment of crossing the source portal so that there were no problems between the pivot of the model and the rendering plane; in this way we achieved our 2 objectives: on the one hand that the portal can have a front and a back part (which makes it aesthetically more attractive although more complex to handle than Unity's native 3D objects) and on the other hand that the illusion of "interdimensional" and fluid movement based on the symmetry of zone A and zone B is still maintained (what is in front of portal A is what is behind portal B and vice versa).

12.2.2 Dialogue and subtitles system

Considering the basic line of dialogue we presented in section 6.6

<time/>005.000|<b>Jorge: </b>Hello world!

This line indicates to the game that in the second five after the start of the audio playback that has the same name as this text file (let's imagine it is dialogue 1), it should be displayed on the screen:

Jorge: Hello world!

On the other hand, a basic event trigger line would be:

<time/>007.000|<trigger/>Rock-Scale-

This line indicates to the game that in the second seven after the start of the audio playback that has the same name as this text file, function Scale must be called (which can do whatever we want, in this case, scale the gameObject) on objects that have "Rock" as a tag.

This whole complex system works essentially through lists of strings that keep the tags of the objects over which functions will be called, the names of these functions and the lines of dialogues as well as lists of floats that keep the times in which each of these operations has to be carried out.

Furthermore, for the correct conversion of time, from string to float, the library **System.Text.RegularExpressions** is used so that we can already work with numbers; once we have these, it is vital to use **System.Globalization.CultureInfo.InvariantCulture** when doing the conversion to avoid syntax problems between those formats that use the period and those that use the comma to separate time.

Finally, one of the most important considerations is that the text document recognizes all the lines in the text, even when those 'lines' are only a carriage return. This was a small problem at the beginning of the testing of the subtitle system because, there were some lines in the .text document that appeared empty but were being recorded in Unity, thus producing a gap in the subtitles and indexing errors.

It was when I started debugging that I realized that what was happening was that those "non-existent" lines were actually carriage returns that I had inadvertently introduced and were causing the subtitle system to malfunction by recognizing that there were more lines of dialogue than there really should be.

12.2.3 Voice Recognition and sound intensity

This system, because it is part of a Windows library for Unity , only works for Microsoft platforms; more specifically, the KeyWord Recognizer is only supported in Windows 10.

Due to time constraints, we will take Windows platforms as the standard, however, the game could become scalable to other platforms such as Playstation 4 through the use of the speech recognition system offered by IBM. However, this particular tool is discouraged because a license is required and also the response time is much greater than the Windows library in Unity, being able to access it locally and natively and with immediate response, while in the IBM system is necessary to access a server and this search tends to increase the response time, obtaining in a large number of occasions less than optimal results in which the game experience is impaired.

On the other hand, one of the problems that could occur concerning the small delay in recognizing words would appear if an unwanted word was shouted, thus reaching a peak of sound intensity, and then the desired word was said with a low sound intensity. In this case it would be recognized that the desired word has been said with the required sound intensity when it has not really been so.

To avoid this, we calibrate the time between recordings to match the approximate time needed to pronounce the word, thus ensuring that, regardless of when the word is pronounced, the sound intensity of its articulation can be recorded; in the same way, we also prevent the recording from being reset at the exact moment when the desired word is recognized.

However, the main problem was encountered when recording the microphone when the scene was not loaded for the first time, i.e. in the case of starting a new game. This was the biggest programming problem I encountered within this section because the malfunction was caused by a bug in Unity that took some time to report in order to get a solution.

The error occurred due to the difficulty Unity has in recording and playing audio simultaneously. In short, this is how sound intensity capture works, since an AudioClip is used that has a desired duration and records what comes in through the microphone input. At the same time this AudioClip is being recorded, it is also being played and this is the way to be able to detect the sound intensity, since, if it were not being played, Unity would have no way to access it. The key is that, when playing, a custom AudioMixer is used that has the sound intensity in -80dB, eliminating all the sound coming from the microphone; however, Unity is able to capture it even if the player does not.

It must be taken into account that Unity is not an engine that is expressly designed for this type of input, and as such, it has a certain delay - 220 milliseconds according to Unit's documentation - therefore its use would not be optimal for a karaoke, for example, but it does fulfill the function required in our project since we do not need a literally immediate response.

The problem, arose when restarting the scene, once the flow of the initial menu and the reset of a game was established. At this point, when starting to record an AudioClip again with the recorded microphone input, there was a delay in the audio input corresponding to the set length of the AudioClip itself. That is, if each audio recording was set to last 10 seconds, this was the time it took the microphone to receive the first input, thus producing a delay of that magnitude and making it impossible to use voice recognition.

After a hard investigation, I ended up finding Unity's documentation which showed this situation as an error widely known by the developers of the engine and which was due to the internal change, without any command being executed, of the preset latency value, thus causing the delay in the emission of the signal captured by the microphone; In this situation, Unity has established a function of the microphone that, through a while loop, allows to wait until it is completely ready for the recording / playback, putting it in the queue until the time is exceeded in milliseconds (samples) of desired latency, which in our case, will be 0.

12.2.4 Mazes' random generation

In the following link, the detailed step-by-step explanation of the algorithm is shown along with some demonstrations and a possible implementation. On the other hand, I will briefly develop the procedural generation of the labyrinths.

This procedural generation is essentially based on obtaining the numbers that indicate the path to follow from the algorithm while it is "wandering" until it runs out of an unvisited cell, neighboring the cell in which it is in that iteration, to move to.

The way this algorithm is usually implemented is through numbers, from 1 to 4, which indicate the place to move to (up, down, left and right). Therefore, it is in this number generation where the random or procedural component of the maze resides. In our case, as it is random, given our design needs, we only have to obtain a random number between 1 and 4 and take into account that we cannot access previously visited boxes. Unity's Random Library allows us to do this without any major problems.
However, it is in the development of the "procedural" number where there may be complications and, although we did not implement it, since I did study it when investigating the implementation of this mechanism, I will detail it in brief.

In short, this procedural number usually has a large number of digits (tens of them) ranging from 1 to 4 - since we must remember that these figures indicate the direction to take by the algorithm. A series of modifications are made on this number (generally based on treating the number as a chain and use substrings at different points of it) to end up getting unique numbers that mark the direction. So where does the procedure reside?

Essentially, this is found in that the number is a constant and the order in which the modifications to obtain substrings are made is always the same, thus being able to obtain infinite modifications but always following a pattern; therefore, extrapolating it to our example of labyrinths, we can obtain infinite of these but always following the same pattern of order according to that initial procedural number -also known as seed.

This seed can change at our whim and even this can be done in an automated way, through random seed generators or if you want to meet a stricter pattern, through seeds obtained based on modifications of the date, for example, thus getting volatile seeds that have a lifetime of only 24 hours.

12.2.5 Avoiding the replay when the loop is over

I include the implementation of this mechanism only in the appendix and not in chapter 6 of the report because the main aspect to highlight of this mechanism is what I tried to do but finally could not achieve.

At first, the goal I set myself regarding this mechanic was to get the Unity executable -which allows to start the application and play- removed from the user's system.

I understood that this was a complicated thing to do but I was hopeful that it would be possible. After investigating, I discovered that there was no way to do it natively in Unity because of the way the Windows Task Manager works. This is because it is not possible to remove a process that is currently running (unless it is done manually, killing the process); the main problem with this is that it was the application (the Unity executable) that had to perform the action of removing itself and this was not possible.

However, I kept on researching and found a way to execute the "auto-elimination" of a process; this could not be done through Unity but some piece of code had to be used through a Bash file. Despite not having much knowledge about this, I kept on investigating to try to get what I wanted and I got this .bash called from the Unity application, which executed a command line in the Windows console.

It seemed that I was close to achieving my goal, all that remained was for the command to be executed on the console to remove a file from a given directory on the local system. Researching this matter, I found a lot of forums on the Internet, most of them with not very good intentions regarding programs capable of modifying and removing local data of the user who runs them. Due to the malicious intentions of a very high number of users, the rest had great misgivings when it came to helping them with these morally dubious tasks and that is why the information, in a large number of cases, was limited.

I finally found some answers that could help me to fulfill my goal, given by users who trusted the good faith of the people who read their posts. However, to my misfortune (and the joy of the majority of Windows users) SSOO's data protection is robust enough to prevent the modification or deletion of local files by external applications (even by granting administrator permissions). The settings to enable this do exist but they are well hidden enough that no one can innocently change them. That's why I finally discarded this initial idea, which involved a configuration that was too special to ensure that all users would comply with it and which could also get through the red lines of their privacy.

Therefore, in order to achieve the effect of rendering the game unplayable, a variable is used within the PlayerPrefs system that is evaluated at the beginning of the game to determine whether the loop has been broken or not.

If not, the application continues without major mishaps; however, if it does (due to the fact that in the previous execution of the game the loop was broken, the application closed automatically and the player has opened it again), the interaction with the Play button is disabled, thus avoiding rejugability, and an audio is played that talks about the break of the loop.

There is no going back to what has already been overcome.

Developers note: if the loop has been broken, the Play button can be enabled again by pressing simultaneously F1 and F2 in the initial menu*(Fig. 31)*.

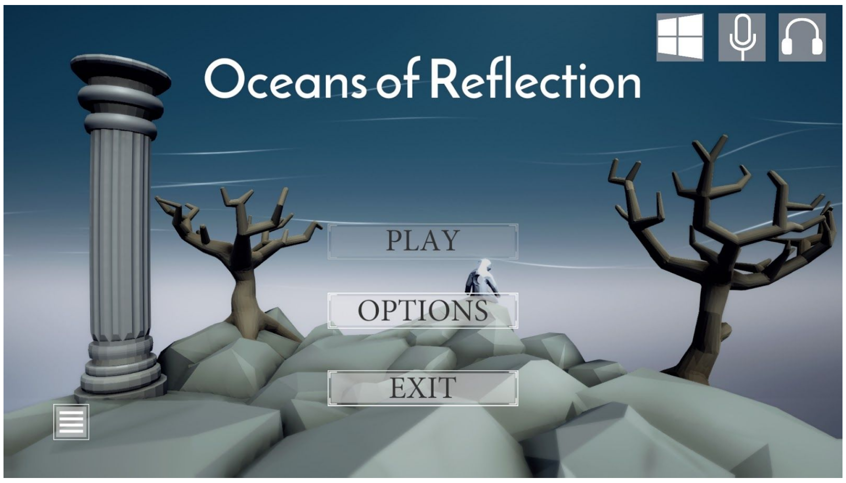


Fig 31. *Initial menu with the loop broken.*

12.2.6 Optimization

The main problem we encountered regarding optimization was with the native implementation of the Occlusion Culling*(Fig. 32)* offered by Unity. This tool, allows the engine not to render what is covered by other objects in the scene. However, this is not the real problem; the difficulty we found is born from the union that Unity has between Occlusion and Frustum Culling, being necessary to incorporate the first one to benefit from the second one.

It is at this point that there is a head-on collision with the operation of our game, due to the use of portals.

The Frustum Culling, takes care that the engine doesn't render those objects of the scene that are out of the Frustum of the main camera (in this case, the vision of the player in first person). The problem that arises with this is that, in the case of the portals, the player is not looking directly at the area that the portal shows, but rather at the procedural texture that is in the portal and that works thanks to a camera; therefore, in the case that Frustum Culling was applied, the vision of the portal would never be correctly rendered since the player would not be looking at the B zone but rather at its visual representation.

Due to this performance and our needs within the project, it was impossible for us to implement the Frustum (and therefore the Occlusion) Culling and we had to resort to other similar but non-native techniques.

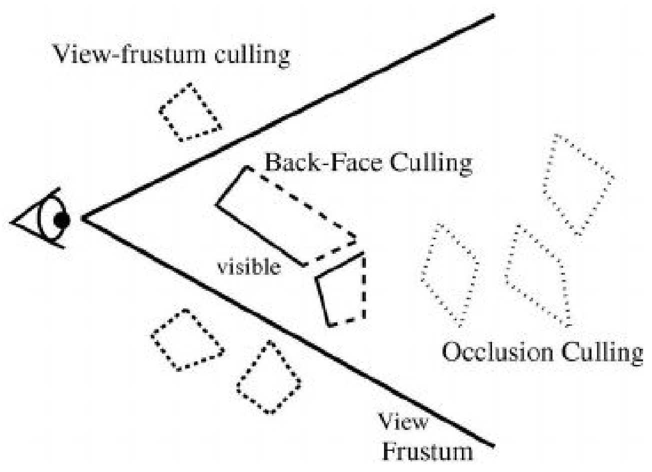


Fig 32. *Representation of different kinds of culling.*

12.3 Dubbing

This phase of the project deserves its own section since it means a great confluence for the branch of video games to be able to mix with other artistic branches such as acting.

This collaboration could be carried out because these three actors - who have recently finished their studies at the School of Dramatic Art of Murcia (ESAD) and who have participated in a great number of projects both in musical theatre and in acting in front of the camera in regional films, as protagonists, and national films, as extras - are friends of mine since we were very young and therefore, they agreed to help me with this project.

In addition to their collaboration, we were also able to count on the necessary material because they had access to it through ESAD itself and contacts within it. This material was essentially a microphone, microphone stand, control station, anti popper, spotlights, panels, supports and specialized headphones. In addition to these tools, I also have the contribution of my PC as well as the free audio editing software, Audacity.

Unfortunately, due to the exceptional situation we found ourselves in at the time of the recording, within the phased de-escalation plan, we could not have access to a recording studio where the sound conditions would have been optimal and therefore, we had to improvise a studio at my grandparents' house, the only one available at that time, and adapt the living room to achieve the best possible environment for the recording (avoid reverberation, obtain a clean sound, deal with external noise factors...). However, through the subsequent mastering process, we were able to obtain a sound quality much higher than expected and that, to a large extent, was possible thanks to the use of professional equipment as well as the participation of professional actors used to dealing with it.

It was a long day of recording in which the actors were kept conditioned at all times and provided with catering services and even personal whims, as we were taught in the subject **VJ1230 - THEORY AND PRACTICE OF AUDIOVISUAL PRODUCTION** in order to obtain the best possible results.

To finish with this section, a figure with a set of images*(Fig 33)* taken during the initial meeting with the actors in which a first reading of the script was made and the general flow of the game was explained; as well as in the day of recording giving certain guidelines to the actors during the development of the session.

As it can be assumed, my colleague Francesc Xavier was not present on this day of recording for reasons related to the restriction of interprovincial travel, however we kept in touch throughout the day and he was one more in the process.

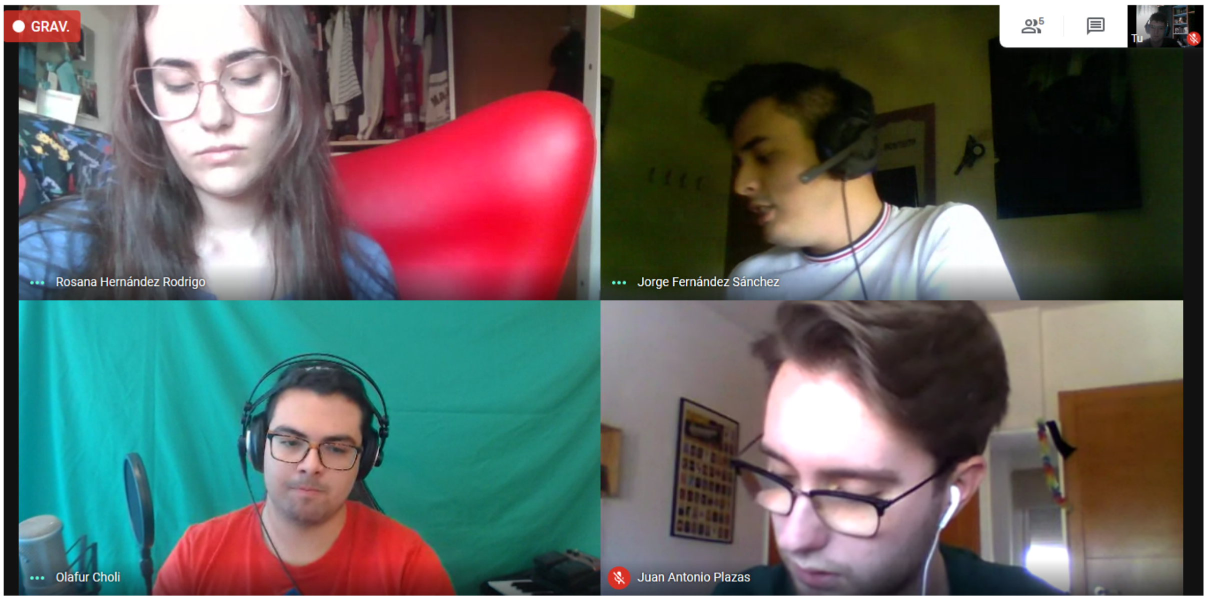


Fig 33. *Virtual meeting with the actors.*

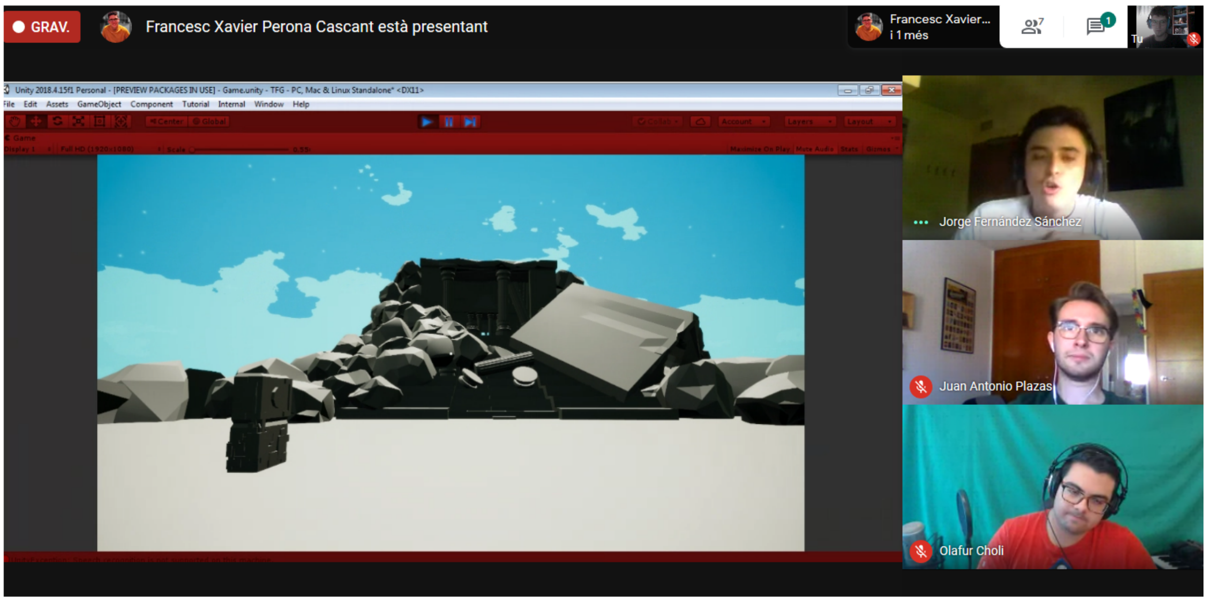
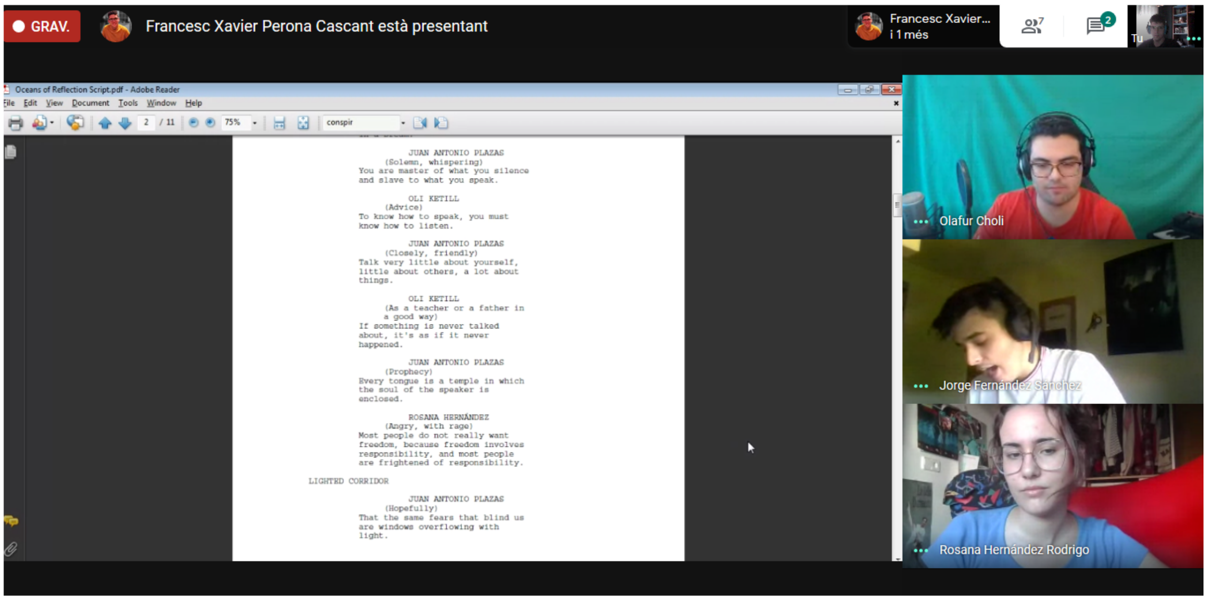


Fig 34. *Project explanation.*


GRAV.
Francesc Xavier Perona Cascant està presentant
Francesc Xavier... i 1 més
Tu
Oceans of Reflection Script.pdf - Adobe Reader
File Edit View Document Tools Window Help
2 / 11
75%
conspir
Olafur Choli
Jorge Fernández Sánchez
Rosana Hernández Rodrigo

```
                    JUAN ANTONIO PLAZAS
               (Solemn, whispering)
          You are master of what you silence
          and slave to what you speak.

                    OLI KETILL
               (Advice)
          To know how to speak, you must
          know how to listen.

                    JUAN ANTONIO PLAZAS
               (Closely, friendly)
          Talk very little about yourself,
          little about others, a lot about
          things.

                    OLI KETILL
               (As a teacher or a father in
               a good way)
          If something is never talked
          about, it's as if it never
          happened.

                    JUAN ANTONIO PLAZAS
               (Prophecy)
          Every tongue is a temple in which
          the soul of the speaker is
          enclosed.

                    ROSANA HERNÁNDEZ
               (Angry, with rage)
          Most people do not really want
          freedom, because freedom involves
          responsibility, and most people
          are frightened of responsibility.

     LIGHTED CORRIDOR

                    JUAN ANTONIO PLAZAS
               (Hopefully)
          That the same fears that blind us
          are windows overflowing with
          light.
```

Fig 35. *Script reading.*

Fig 36. *Performance by Juan Antonio Plazas.*

Fig 37. *Performance by Rosana Hernández.*

Fig 38. *Performance by Oli Ketill.*

Fig 39. *Guidelines to Juan Antonio Plazas.*

Fig 40. *Guidelines to Rosana Hernández.*

Fig 41. *Guidelines to Oli Ketill.*

Fig 42. *Juan Antonio Plazas acting.*

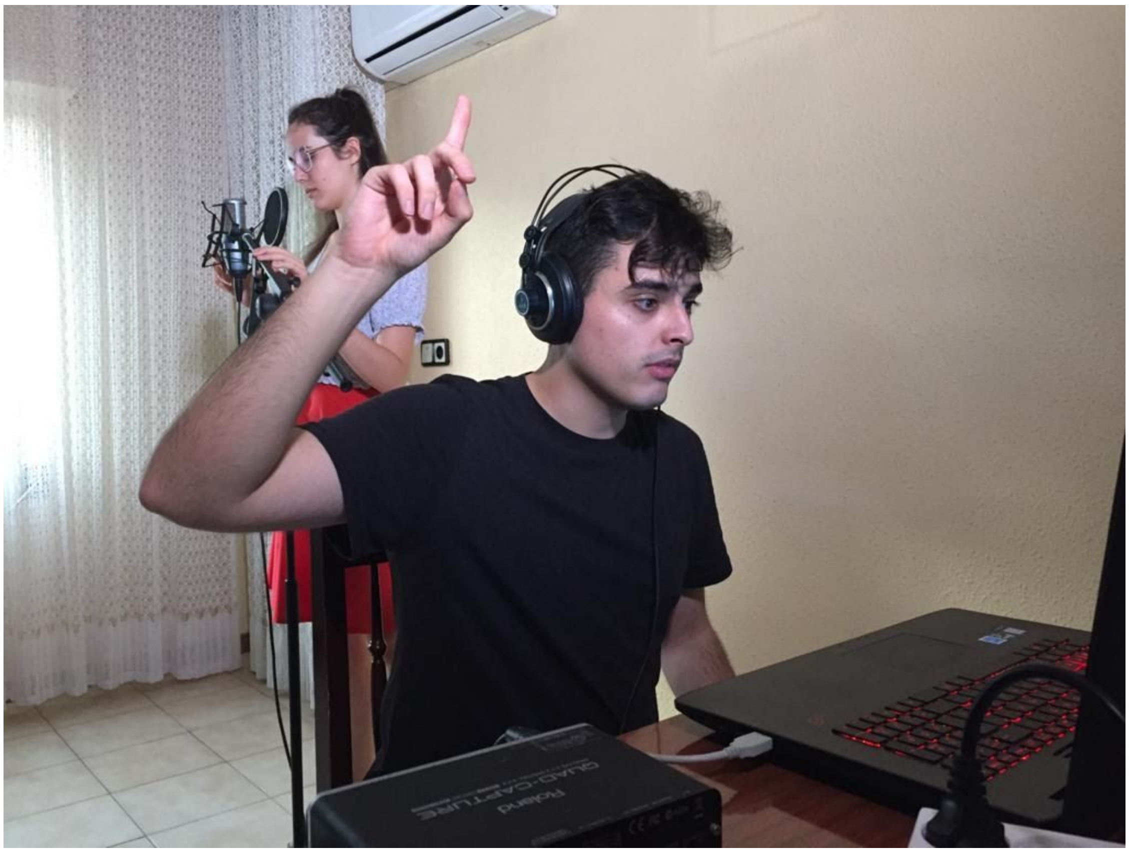
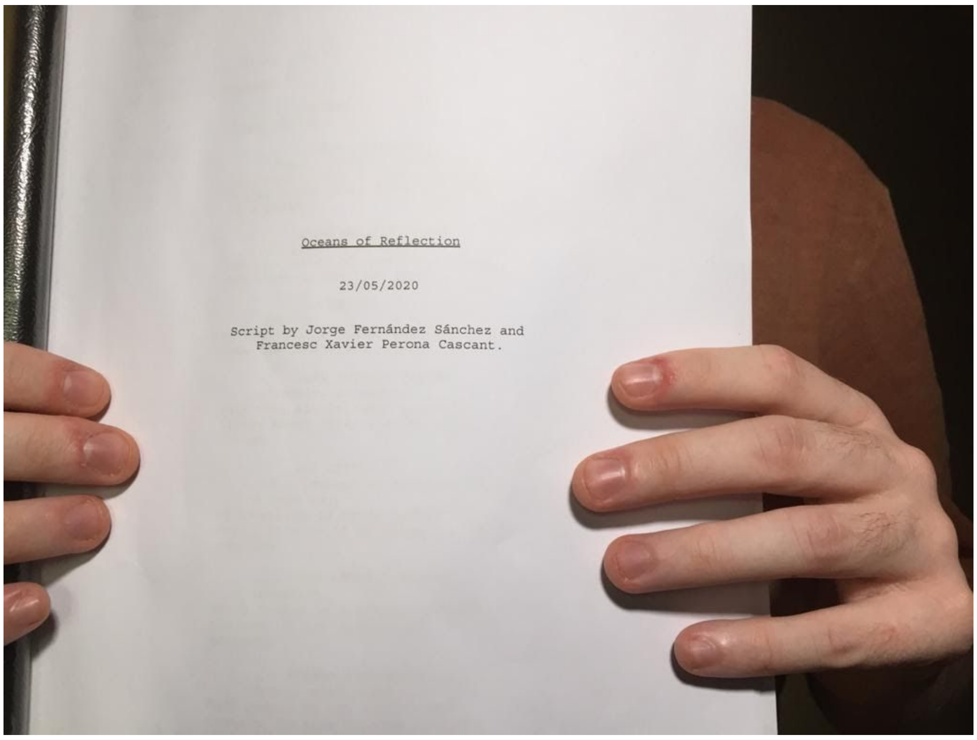


Fig 43. *Rosana Hernández acting.*



Oceans of Reflection

23/05/2020

Script by Jorge Fernández Sánchez and
Francesc Xavier Perona Cascant.

Fig 44. *Oceans of Reflection script.*

12.4 Source code

Only those scripts that have a special relevance due to its difficulty or any other particularity will be attached, trying to avoid this way, the overload of the document with elements of low or null importance, necessary for the correct operation but simple in excess.

12.4.1 Load.cs

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

public class Load : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        //Start async operation
        StartCoroutine(LoadAsyncOperation());
    }

    IEnumerator LoadAsyncOperation()
    {
        //create an async operation
        AsyncOperation gameLevel = SceneManager.LoadSceneAsync("Game");

        while (gameLevel.progress < 1)
        {
            yield return new WaitForEndOfFrame();
        }
    }
}
```

12.4.2 DialogueManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Text.RegularExpressions;
using UnityEngine.UI;

public class DialogueManager : MonoBehaviour
{
    private AudioClip dialogueAudio;
    private string[] fileLines;
    //Subtitle variables
    private List<string> subtitleLines = new List<string>();
    private List<string> subtitleTimingStrings = new List<string>();
    public List<float> subtitleTimings = new List<float>();
    public List<string> subtitleText = new List<string>();
    private int nextSubtitle = 0;
    private string displaySubtitle;

    //Trigger variables
    private List<string> triggerLines = new List<string>();

    private List<string> triggerTimingStrings = new List<string>();
    public List<float> triggerTimings = new List<float>();

    private List<string> triggers = new List<string>();
    public List<string> triggerObjectNames = new List<string>();
    public List<string> triggerMethodNames = new List<string>();

    private int nextTrigger = 0;

    //GUI
    private GUIStyle subtitlesStyle = new GUIStyle();
    private float scaleRatio = 1.5f;
    private float heightRatio = 0.06f;

    //Singleton property
    public static DialogueManager Instance {get; private set;}

    [HideInInspector]
    public AudioSource audioSource;

    private float timer;

    public bool subtitlesEnabled;
    public Toggle toggle;

    private float timeOffset;
    public Font subtitlesFont;
```

```
void Awake()
    {
        if (Instance != null && Instance != this)
        {
            Destroy(gameObject);
        }

        Instance = this;
        audioSource = GetComponent<AudioSource>();

        if (!PlayerPrefs.HasKey("subtitlesEnabled"))
        {
            PlayerPrefs.SetInt("subtitlesEnabled", true?1:0);
            subtitlesEnabled = true;
        }

        else
        {
            subtitlesEnabled =
PlayerPrefs.GetInt("subtitlesEnabled")==1?true:false;
        }

        toggle.isOn = subtitlesEnabled;

    }

    public void BeginDialogue (AudioClip passedClip, float timeOffset)
    {
        timer = 0f;
        this.timeOffset = timeOffset;

        dialogueAudio = passedClip;

        subtitleLines = new List<string>();
        subtitleTimingStrings = new List<string>();
        subtitleTimings = new List<float>();
        subtitleText = new List<string>();

        triggerLines = new List<string>();
        triggerTimingStrings = new List<string>();
        triggerTimings = new List<float>();
        triggers = new List<string>();
        triggerObjectNames = new List<string>();
        triggerMethodNames = new List<string>();

        nextSubtitle = 0;
        nextTrigger = 0;

        //Get everything from the text file

        TextAsset temp = Resources.Load("Dialogues/" + dialogueAudio.name) as TextAsset;
        fileLines = temp.text.Split('\n');

        //Split subtitle and trigger related lines into different lists

        foreach(string line in fileLines)
        {
            if (line.Contains("<trigger/>"))
            {
                triggerLines.Add(line);
            }

            else
            {
                subtitleLines.Add(line);
            }
        }

        //Split out our subtitle elements

        for (int cnt = 0; cnt < subtitleLines.Count; cnt++)
        {
            string [] splitTemp = subtitleLines[cnt].Split('|');
            subtitleTimingStrings.Add(splitTemp[0]);
            subtitleTimings.Add(float.Parse(CleanTimeString(subtitleTimingStrings[cnt]),
System.Globalization.CultureInfo.InvariantCulture));
            subtitleText.Add(splitTemp[1]);
        }

        //Split out our trigger elements
for (int cnt = 0; cnt < triggerLines.Count; cnt++)
        {
            string [] splitTemp1 = triggerLines[cnt].Split('|');
            triggerTimingStrings.Add(splitTemp1[0]);
            triggerTimings.Add(float.Parse(CleanTimeString(triggerTimingStrings[cnt]),
System.Globalization.CultureInfo.InvariantCulture));

            triggers.Add(splitTemp1[1]);

            string[] splitTemp2 = triggers[cnt].Split('-');
            splitTemp2[0] = splitTemp2[0].Replace("<trigger/>", "");
            triggerObjectNames.Add(splitTemp2[0]);
            triggerMethodNames.Add(splitTemp2[1]);
        }
        //Set initial subtitle text
        if (subtitleText[0] != null)
        {
            displaySubtitle = subtitleText[0];
        }
        //Set and play the audioclip
        audioSource.clip = dialogueAudio;
        audioSource.Play();
```

```
    }

    void Update ()
    {
        //The timer is on only if we have a clip ready
        if (audioSource.clip != null)
        {
            timer += Time.deltaTime;
        }

        //If the clip is over, we remove it from the audioSource
        if (audioSource.clip != null && timer > audioSource.clip.length + timeOffset)
        {
            audioSource.clip = null;
        }
    }

    //Remove all characters that are not part of the timing float

    private string CleanTimeString (string timeString)
    {
        Regex digitsOnly = new Regex (@"[^\d+(\.\d+)*s]");
        return digitsOnly.Replace(timeString, "");
    }

    void OnGUI ()
    {
        //Make sure that we are using a proper dialogeAudio file
        if (dialogueAudio != null && audioSource.clip != null && audioSource.clip.name
== dialogueAudio.name)
        {
            //Check for the <break/> or negative nextSubtitle number
            if(nextSubtitle > 0 && !subtitleText[nextSubtitle-1].Contains("<break/>"))
            {
                //Create our GUI
                GUI.depth = -1001;
                subtitlesStyle.fixedWidth = Screen.width / scaleRatio;
                subtitlesStyle.wordWrap = true;
                subtitlesStyle.alignment = TextAnchor.MiddleCenter;
                subtitlesStyle.normal.textColor = Color.white;
                subtitlesStyle.fontSize = Mathf.FloorToInt(Screen.height * heightRatio);
                subtitlesStyle.font = subtitlesFont;

                if (subtitlesEnabled && !ManagerMenu.Instance.paused)
                {
                    Vector2 size = subtitlesStyle.CalcSize(new GUIContent());
                    GUI.contentColor = Color.black;
                    GUI.Label(new Rect(Screen.width/2 - size.x/2 + 1, Screen.height/1.1f
- size.y + 1, size.x, size.y), displaySubtitle, subtitlesStyle);
                    GUI.contentColor = Color.white;
                    GUI.Label(new Rect(Screen.width/2 - size.x/2, Screen.height/1.1f -
size.y, size.x, size.y), displaySubtitle, subtitlesStyle);
                }
                else
                {
                    Vector2 size = subtitlesStyle.CalcSize(new GUIContent());
                        GUI.contentColor = Color.black;
                        GUI.Label(new Rect(Screen.width/2 - size.x/2 + 1,
Screen.height/1.1f - size.y + 1, size.x, size.y), "", subtitlesStyle);
                        GUI.contentColor = Color.white;
                        GUI.Label(new Rect(Screen.width/2 - size.x/2, Screen.height/1.1f
- size.y, size.x, size.y), "", subtitlesStyle);
                }
            }

            //Increment nextSubtitle when we hit the associated time point
            if(nextSubtitle < subtitleText.Count)
            {
                if (timer > subtitleTimings[nextSubtitle])
                {
                    displaySubtitle = subtitleText[nextSubtitle];
                    nextSubtitle++;
                }
            }

            //Fire triggers when we hit the associated time point
            if(nextTrigger < triggers.Count)
            {
                if (timer > triggerTimings[nextTrigger])
                {

GameObject.FindGameObjectWithTag(triggerObjectNames[nextTrigger]).SendMessage(triggerMet
hodNames[nextTrigger]);
                    nextTrigger++;
                }
            }
        }
    }
}
```

12.4.3 VoiceRecognition.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using UnityEngine.Windows.Speech;
using UnityEngine.Audio;

public class VoiceRecognition : MonoBehaviour
{
    private KeywordRecognizer keywordRecognizer;
    private Dictionary<string, Action> actions = new Dictionary<string, Action>();

    public AudioSource audioScreams;

    private float sensitivity;
    private float loudness;
    private float highestLoudness;
    private float timer;
    private AudioSource _audio;
    public bool useMicrophone;
    private string selectedDevice;

    public AudioMixerGroup _mixerGroupMicrophone, _mixerGroupMaster;

    public delegate void Action();
    public static event Action ScreamStop;

    public SaveVoice saveVoice;

    private float minimumLoudness;
    private GameManager gameManager;

    private AudioSource scratchAudioSource;

    public AudioClip eventClip;
    public AudioClip strongerClip;
    private float timeOffset;

    void Awake()
    {
        _audio = gameObject.AddComponent<AudioSource>();
        gameManager = 
GameObject.FindGameObjectWithTag("GameManager").GetComponent<GameManager>();

        if (useMicrophone)
        {
            if(Microphone.devices.Length > 0)
            {
                selectedDevice = Microphone.devices[Microphone.devices.Length - 
1].ToString();
```

```
                Debug.Log(Microphone.devices.Length + ", " + selectedDevice);
                _audio.outputAudioMixerGroup = _mixerGroupMicrophone;
                _audio.clip = Microphone.Start(selectedDevice, true, 10,
AudioSettings.outputSampleRate);
                _audio.loop = true;
                _audio.playOnAwake = false;
                while(!(Microphone.GetPosition(selectedDevice) > 0)) {}
                _audio.Play();
            }

            else
            {
                useMicrophone = false;
            }

        }

        if (!useMicrophone)
        {
            _audio.outputAudioMixerGroup = _mixerGroupMaster;
            _audio.clip = null;
        }
    }

    void Start(){
        actions.Add("stop", Stop);
        actions.Add("past", Past);

        scratchAudioSource =
GameObject.FindGameObjectWithTag("Scratch").GetComponent<AudioSource>();
        timeOffset = 2f;


        keywordRecognizer = new KeywordRecognizer(actions.Keys.ToArray());
        keywordRecognizer.OnPhraseRecognized += RecognizedSpeech;
        keywordRecognizer.Start();

        sensitivity = 100f;
        loudness = 0f;
        highestLoudness = 0f;
        timer = 0f;
        minimumLoudness = 55f;
    }

    void Update ()
    {
        timer += Time.deltaTime;

        loudness = GetAveragedVolume() * sensitivity;
        //Debug.Log(loudness);

        if (loudness > highestLoudness)
        {
            highestLoudness = loudness;
        }

        if (timer > 3f)
        {
            timer = 0f;
            highestLoudness = loudness;
        }
    }

    void RecognizedSpeech(PhraseRecognizedEventArgs speech){
        //Debug.Log(speech.text);
        actions[speech.text].Invoke();
    }

    void Stop()
    {
        Debug.Log("STOP");
        if (audioScreams.isPlaying && !ManagerMenu.Instance.paused &&
GameManager.Instance.indexIsland == 2)
        {
            Debug.Log(highestLoudness);

            if (highestLoudness > minimumLoudness)
            {
                audioScreams.Stop();
                GameManager.Instance.PlayFallDownJail();

                Microphone.End(selectedDevice);

                saveVoice.audioClip = _audio.clip;
                _audio.Stop();
                _audio.clip = null;
                _audio.clip = Microphone.Start(selectedDevice, true, 10,
AudioSettings.outputSampleRate);   //10
                _audio.loop = true;
                while(!(Microphone.GetPosition(selectedDevice) > 0)) {}
                _audio.Play();

                gameManager.OpenSecondPortal();

                if(ScreamStop != null)
                {
                    ScreamStop();
                }

                if (DialogueManager.Instance.audioKindEnum ==
AudioKind.AudioKindEnum.Time)
                {

scratchAudioSource.gameObject.GetComponent<ScratchManager>().ScratchTime();
                }
                else if (DialogueManager.Instance.audioKindEnum ==
AudioKind.AudioKindEnum.Voxophone)
                {

scratchAudioSource.gameObject.GetComponent<ScratchManager>().ScratchVoxophone();
                }

                DialogueManager.Instance.audioSource.spatialBlend = 0f;
                DialogueManager.Instance.BeginDialogue(eventClip, timeOffset,
AudioKind.AudioKindEnum.Event);
                GameManager.Instance.finishAudios = true;
            }

            else
            {
                if (DialogueManager.Instance.audioKindEnum ==
AudioKind.AudioKindEnum.Time)
                {

scratchAudioSource.gameObject.GetComponent<ScratchManager>().ScratchTime();
                }

                else if (DialogueManager.Instance.audioKindEnum ==
AudioKind.AudioKindEnum.Voxophone)
                {

scratchAudioSource.gameObject.GetComponent<ScratchManager>().ScratchVoxophone();
                }

                DialogueManager.Instance.audioSource.spatialBlend = 0f;
                DialogueManager.Instance.BeginDialogue(strongerClip, timeOffset,
AudioKind.AudioKindEnum.Event);
            }

        }
    }

    void Past()
    {
        Debug.Log("PAST");
        if (!GameManager.Instance.statueDisolved &&
GameManager.Instance.onPositionToDisolve && !ManagerMenu.Instance.paused)
        {
            GameManager.Instance.DissolveStatueSign();
            //Debug.Log("PAST");
        }

    }
    float GetAveragedVolume ()
    {
        float[] data = new float[256];
        float a = 0;
        _audio.GetOutputData(data, 0);
        foreach (float s in data)
        {
            a += Mathf.Abs(s);

        }
        return a/256;
    }

    public float GetLoudness ()
    {
        return loudness;
    }

}
```

12.4.4 TakePhoto.cs

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

public class TakePhoto : MonoBehaviour
{
    [HideInInspector]
    public bool cameraPicked;

    public GameObject cameraObject;
    public GameObject canvasReticula;
    public GameObject canvasPhoto;
    public GameObject photoPrefab;

    private bool focus;

    private string finalPath;
    private Sprite last_screenshot_save;

    private Transform guide;

    private GameObject photoObject;

    private Animator cameraAnimator;

    // Start is called before the first frame update
    void Start()
    {
        cameraPicked = false;
        focus = false;
        canvasPhoto.SetActive(false);
        guide = GameObject.FindGameObjectWithTag("Player").transform;
        cameraAnimator = cameraObject.GetComponent<Animator>();
    }

    // Update is called once per frame
    void Update()
    {
        if (cameraPicked && !ManagerMenu.Instance.paused)
        {
            if (Input.GetMouseButtonDown(1))
            {
                Debug.Log("enfoque");
                focus = true;
                cameraAnimator.SetTrigger("Focus");
                canvasReticula.SetActive(false);
                canvasPhoto.SetActive(true);
                if (photoObject != null)
                {
                    photoObject.SetActive(false);
                }
            }

            if (Input.GetMouseButtonUp(1))
            {
                Debug.Log("desenfoque");
                focus = false;
                cameraAnimator.SetTrigger("Unfocus");
                canvasReticula.SetActive(true);
                canvasPhoto.SetActive(false);
                if (photoObject != null)
                {
                    photoObject.SetActive(true);
                }
            }

            if (focus && Input.GetMouseButtonDown(0))
            {
                StartCoroutine(TakingPhoto());
            }
        }
    }

    private IEnumerator TakingPhoto()
    {
        Debug.Log("PHOTO");
        string fileName = System.DateTime.Now.ToString("dd-MM-yyyy-HH-mm-ss");

        finalPath = (Application.persistentDataPath + "/Screenshot " + fileName +
".png");

        ScreenCapture.CaptureScreenshot(finalPath);
        yield return new WaitForSeconds(1f);
        last_screenshot_save = LoadSprite(finalPath);



        Debug.Log("changing");

        if (photoObject == null)
        {
            photoObject = Instantiate(photoPrefab, guide.transform.position,
photoPrefab.transform.rotation, guide);
            photoObject.transform.localPosition = new Vector3 (-1,0,1.7f);
            photoObject.transform.localRotation = Quaternion.identity;
            photoObject.SetActive(false);
        }

        photoObject.GetComponent<SpriteRenderer>().sprite = last_screenshot_save;
        yield return new WaitForSeconds(5f);
        File.Delete(finalPath);
    }
    private Sprite LoadSprite(string path)
    {
        if (string.IsNullOrEmpty(path))
        {
            return null;
        }
        if (System.IO.File.Exists(path))
        {
            byte[] bytes = System.IO.File.ReadAllBytes(path);
            Texture2D texture = new Texture2D(1, 1);
            texture.LoadImage(bytes);
            Sprite sprite = Sprite.Create(texture, new Rect(0, 0, texture.width,
texture.height), new Vector2(0.5f, 0.5f));
            return sprite;
        }

        return null;
    }
}
```

12.4.5 HuntAndKill.cs

```
using UnityEngine;
using System.Collections;

public class HuntAndKillMazeAlgorithm : MazeAlgorithm {

    private int currentRow = 0;
    private int currentColumn = 0;

    private bool courseComplete = false;

    public HuntAndKillMazeAlgorithm(MazeCell[,] mazeCells) : base(mazeCells) {}

    public override void CreateMaze () {
        HuntAndKill ();
    }

    private void HuntAndKill() {
        mazeCells [currentRow, currentColumn].visited = true;

        while (! courseComplete) {
            Kill(); // Will run until it hits a dead end.
            Hunt(); // Finds the next unvisited cell with an adjacent visited cell. If
it can't find any, it sets courseComplete to true.
        }
    }

    private void Kill() {
        while (RouteStillAvailable (currentRow, currentColumn)) {
            int direction = Random.Range (1, 5);


            if (direction == 1 && CellIsAvailable (currentRow - 1, currentColumn)) {
                // North
                DestroyWallIfItExists (mazeCells [currentRow, currentColumn].northWall);
                DestroyWallIfItExists (mazeCells [currentRow - 1,
currentColumn].southWall);
                currentRow--;
            } else if (direction == 2 && CellIsAvailable (currentRow + 1,
currentColumn)) {
                // South
                DestroyWallIfItExists (mazeCells [currentRow, currentColumn].southWall);
                DestroyWallIfItExists (mazeCells [currentRow + 1,
currentColumn].northWall);
                currentRow++;
            } else if (direction == 3 && CellIsAvailable (currentRow, currentColumn +
1)) {
                // east
                DestroyWallIfItExists (mazeCells [currentRow, currentColumn].eastWall);
                DestroyWallIfItExists (mazeCells [currentRow, currentColumn +
1].westWall);
                currentColumn++;
            } else if (direction == 4 && CellIsAvailable (currentRow, currentColumn -
1)) {
                // west
                DestroyWallIfItExists (mazeCells [currentRow, currentColumn].westWall);
                DestroyWallIfItExists (mazeCells [currentRow, currentColumn -
1].eastWall);
                currentColumn--;
            }

            mazeCells [currentRow, currentColumn].visited = true;
        }
    }

    private void Hunt() {
        courseComplete = true; // Set it to this, and see if we can prove otherwise
below!

        for (int r = 0; r < mazeRows; r++) {
            for (int c = 0; c < mazeColumns; c++) {
                if (!mazeCells [r, c].visited && CellHasAnAdjacentVisitedCell(r,c)) {
                    courseComplete = false; // Yep, we found something so definitely do
another Kill cycle.
                    currentRow = r;
                    currentColumn = c;
                    DestroyAdjacentWall (currentRow, currentColumn);
                    mazeCells [currentRow, currentColumn].visited = true;
                    return; // Exit the function
                }
            }
        }
    }


    private bool RouteStillAvailable(int row, int column) {
        int availableRoutes = 0;

        if (row > 0 && !mazeCells[row-1,column].visited) {
            availableRoutes++;
        }

        if (row < mazeRows - 1 && !mazeCells [row + 1, column].visited) {
            availableRoutes++;
        }

        if (column > 0 && !mazeCells[row,column-1].visited) {
            availableRoutes++;
        }

        if (column < mazeColumns-1 && !mazeCells[row,column+1].visited) {
            availableRoutes++;
        }

        return availableRoutes > 0;
    }

    private bool CellIsAvailable(int row, int column) {
        if (row >= 0 && row < mazeRows && column >= 0 && column < mazeColumns &&
!mazeCells [row, column].visited) {
            return true;
        } else {
            return false;
        }
    }

    private void DestroyWallIfItExists(GameObject wall) {
        if (wall != null) {
            GameObject.Destroy (wall);
        }
    }

    private bool CellHasAnAdjacentVisitedCell(int row, int column) {
        int visitedCells = 0;

        // Look 1 row up (north) if we're on row 1 or greater
        if (row > 0 && mazeCells [row - 1, column].visited) {
            visitedCells++;
        }

        // Look one row down (south) if we're the second-to-last row (or less)
        if (row < (mazeRows-2) && mazeCells [row + 1, column].visited) {
            visitedCells++;
        }

        // Look one row left (west) if we're column 1 or greater
        if (column > 0 && mazeCells [row, column - 1].visited) {
            visitedCells++;
        }

        // Look one row right (east) if we're the second-to-last column (or less)
        if (column < (mazeColumns-2) && mazeCells [row, column + 1].visited) {
            visitedCells++;
        }

        // return true if there are any adjacent visited cells to this one
        return visitedCells > 0;
    }

    private void DestroyAdjacentWall(int row, int column) {
        bool wallDestroyed = false;

        while (!wallDestroyed) {
            int direction = Random.Range (1, 5);


            if (direction == 1 && row > 0 && mazeCells [row - 1, column].visited) {
                DestroyWallIfItExists (mazeCells [row, column].northWall);
                DestroyWallIfItExists (mazeCells [row - 1, column].southWall);
                wallDestroyed = true;
            } else if (direction == 2 && row < (mazeRows-2) && mazeCells [row + 1,
column].visited) {
                DestroyWallIfItExists (mazeCells [row, column].southWall);
                DestroyWallIfItExists (mazeCells [row + 1, column].northWall);
                wallDestroyed = true;
            } else if (direction == 3 && column > 0 && mazeCells [row,
column-1].visited) {
                DestroyWallIfItExists (mazeCells [row, column].westWall);
                DestroyWallIfItExists (mazeCells [row, column-1].eastWall);
                wallDestroyed = true;
            } else if (direction == 4 && column < (mazeColumns-2) && mazeCells [row,
column+1].visited) {
                DestroyWallIfItExists (mazeCells [row, column].eastWall);
                DestroyWallIfItExists (mazeCells [row, column+1].westWall);
                wallDestroyed = true;
            }
        }

    }

}
```

12.4.6 LoopManager.cs

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

public class LoopManager : MonoBehaviour
{
    public Button playButton;
    private AudioSource audioSource;
    private float timePlaying;

    // Start is called before the first frame update
    void Start()
    {
        audioSource = GetComponent<AudioSource>();
        timePlaying = 0f;

        if (!PlayerPrefs.HasKey("Loop"))
        {
            PlayerPrefs.SetInt("Loop", 0);
        }

        //Loop is broken
        if (PlayerPrefs.GetInt("Loop") != 0)
        {
            playButton.interactable = false;
            audioSource.Play();
        }

    }

    // Update is called once per frame
    void Update()
    {
        //Technical shortcut to play again even with the loop broken
        if (Input.GetKey(KeyCode.F1) && Input.GetKey(KeyCode.F2))
        {
            if (PlayerPrefs.GetInt("Loop") != 0)
            {
                playButton.interactable = true;
                audioSource.Stop();
                timeRunning = false;
                PlayerPrefs.SetInt("Loop", 0);
            }
        }
    }
}
```

12.4.7 GameManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityStandardAssets.Characters.FirstPerson;
using UnityEngine.SceneManagement;

public class GameManager : MonoBehaviour
{
    //Singleton property
    public static GameManager Instance {get; private set;}

    [HideInInspector]
    public int indexIsland;
    [HideInInspector]
    public int indexInsideIsland;

    public AudioSource audioScreams;
    public GameObject colliderSecondPortal;
    public GameObject colliderSecondStop;

    public GameObject colliderThirdPortal;
    public GameObject colliderThirdStop;
    public GlobalTimeManager globalTimeManager;
    public GameObject physicalSun;
    public realTime realTimeManager;

    public AudioClip initClip;
    public AudioSource BSOAudioSource;

    private float timer;
    private bool fadeOut;
    [HideInInspector]
    public bool presentationDone;

    public Image imgWhite;
    public Image imgBlack;

    private FirstPersonController firstPersonController;

    public CharacterController player;
    public Transform beginPoint;

    public AudioClip[] arrayAudiosCorridor;
    public float[] arrayTimesCorridor;

    public AudioClip[] arrayAudiosInitialIsland;
    public float[] arrayTimesInitialIsland;
    public AudioClip[] arrayAudiosStatuesIsland;
    public float[] arrayTimesStatuesIsland;
    public AudioClip[] arrayAudiosTimeIslandDay;
    public float[] arrayTimesTimeIslandDay;

    public AudioClip[] arrayAudiosTimeIslandNight;
    public float[] arrayTimesTimeIslandNight;

    [HideInInspector]
    public float timerAudios;
    [HideInInspector]
    public int indexAudio;
    [HideInInspector]
    public bool finishAudios;
    public AudioSource guillotineAudio;
    public GuillotineManager guillotineManager;
    [HideInInspector]
    public bool manualNight;
    private float timerDecision;
    [HideInInspector]
    public bool decision;
    public PressButton plant, fish;
    private float timeDecision;
    private float timerInaction;
    private float timeInaction;

    public VoiceRecognition voiceRecognition;

    [HideInInspector]
    public bool statueDisolved;
    [HideInInspector]
    public bool onPositionToDisolve;
    public GameObject statueSign;
    public Transform particlesDissolvePosition;
    public GameObject particlesDissolveEffect;

    public SaveVoice saveVoice;
    [HideInInspector]
    public bool action;
    public AudioSource fallDownJail;
    public GameObject[] prisoners;
    public LoopManagerGame loopManager;
    public GameObject shaderSecondPortalDisabled;
    public GameObject thirdSecondPortalDisabled;
    public Transform[] freedSoulPositionsPrisoners;
    void Awake()
    {
        if (Instance != null && Instance != this)
        {
            Destroy(gameObject);
        }

        Instance = this;
    }

    // Start is called before the first frame update
    void Start()
    {
        indexIsland = 1;         //1
        indexInsideIsland = 1;
        indexAudio = 0;
        physicalSun.SetActive(false);

        timer = 0;
        timerAudios = 0;
        timerDecision = 0;
        timeDecision = 90;
        timerInaction = 0;
        timeInaction = 90;
        fadeOut = false;             //false
        presentationDone = false;    //false
        finishAudios = false;
        decision = false;
        action = false;
        statueDisolved = false;
        onPositionToDisolve = false;

        BSOAudioSource.clip = initClip;
        BSOAudioSource.Play();          //outcomment
        imgWhite.gameObject.SetActive(true);     //true
        imgBlack.gameObject.SetActive(false);


        firstPersonController =
GameObject.FindGameObjectWithTag("Player").GetComponent<FirstPersonController>();

        player.enabled = false;
        player.transform.position = beginPoint.position;    //outcomment
        player.enabled = true;
        manualNight = false;

    }
    // Update is called once per frame
    void Update()
    {
        //outcomment
        if (timer < initClip.length - 5)
        {
            timer += Time.deltaTime;
        }

        else
        {
            if (!fadeOut)
            {
                StartCoroutine(FadeOut());
                fadeOut = true;
            }
        }

        if (!ManagerMenu.Instance.paused && !finishAudios && presentationDone)
        {
            timerAudios += Time.deltaTime;
            //Debug.Log(indexAudio);
            //Debug.Log(timerAudios);
        }

        if (indexIsland == 1 && indexInsideIsland == 1)     //corridor
        {
            if(timerAudios > arrayTimesCorridor[indexAudio])
            {
                DialogueManager.Instance.audioSource.spatialBlend = 0f;
                DialogueManager.Instance.BeginDialogue(arrayAudiosCorridor[indexAudio],
2f, AudioKind.AudioKindEnum.Time);
                if (indexAudio < arrayTimesCorridor.Length - 1)
                {
                    indexAudio++;
                }

                else
                {
                    finishAudios = true;
                    timerAudios = 0;
                }

            }

            if (DialogueManager.Instance.audioKindEnum != AudioKind.AudioKindEnum.None)
            {
                timerAudios = 0;
            }
        }
        else if (indexIsland == 1 && indexInsideIsland == 2)        //initial island
        {
            if(timerAudios > arrayTimesInitialIsland[indexAudio])
            {
                DialogueManager.Instance.audioSource.spatialBlend = 0f;

DialogueManager.Instance.BeginDialogue(arrayAudiosInitialIsland[indexAudio], 2f,
AudioKind.AudioKindEnum.Time);
                if (indexAudio < arrayTimesInitialIsland.Length - 1)
                {
                    indexAudio++;
                }

                else
                {
                    finishAudios = true;
                    timerAudios = 0;
                }

            }

            if (DialogueManager.Instance.audioKindEnum != AudioKind.AudioKindEnum.None)
            {
                timerAudios = 0;
            }
        }

        else if (indexIsland == 2 && indexInsideIsland == 1)    //statues island
        {
            if(timerAudios > arrayTimesStatuesIsland[indexAudio])
            {
                DialogueManager.Instance.audioSource.spatialBlend = 0f;

DialogueManager.Instance.BeginDialogue(arrayAudiosStatuesIsland[indexAudio], 2f,
AudioKind.AudioKindEnum.Time);

                indexAudio = Random.Range(1,4);

            }

            if (DialogueManager.Instance.audioKindEnum != AudioKind.AudioKindEnum.None)
            {
                timerAudios = 0;
            }
        }
        else if (indexIsland == 3)      //time island
        {
            if (!decision)
            {
                timerDecision += Time.deltaTime;
                if (timerDecision > timeDecision)
                {
                    plant.NoDecision();
                    fish.NoDecision();
                    fish.PlayAudio();
                    decision = true;
                }
            }


            if (GetIsNight() && indexInsideIsland != 2 && !manualNight)
            {
                indexInsideIsland = 2;
                finishAudios = false;
                timerAudios = 0;
                indexAudio = 0;
            }

            if (indexInsideIsland == 1)     //day
            {
                if (manualNight && !finishAudios)
                {
                    finishAudios = true;
                    timerAudios = 0;
                }

                if(timerAudios > arrayTimesTimeIslandDay[indexAudio])
                {
                    DialogueManager.Instance.audioSource.spatialBlend = 0f;

DialogueManager.Instance.BeginDialogue(arrayAudiosTimeIslandDay[indexAudio], 2f,
AudioKind.AudioKindEnum.Time);

                    if (indexAudio < arrayTimesTimeIslandDay.Length - 1)
                    {
                        indexAudio++;
                    }

                    else
                    {
                        finishAudios = true;
                        timerAudios = 0;
                    }
                }
            }
            else        // automatic night (execution done)
            {
                if(timerAudios > arrayTimesTimeIslandNight[indexAudio])
                {
                    DialogueManager.Instance.audioSource.spatialBlend = 0f;

DialogueManager.Instance.BeginDialogue(arrayAudiosTimeIslandNight[indexAudio], 2f,
AudioKind.AudioKindEnum.Time);

                    if (indexAudio < arrayTimesTimeIslandNight.Length - 1)
                    {
                        indexAudio++;
                    }

                    else
                    {
                        finishAudios = true;
                        timerAudios = 0;
                    }
                }
            }

            if (DialogueManager.Instance.audioKindEnum != AudioKind.AudioKindEnum.None)
            {
                timerAudios = 0;
            }
        }

        else if (!finishAudios)
        {
            finishAudios = true;
        }

        if (indexIsland == 6)   //lacanian island
        {
            if (!action)
            {
                timerInaction += Time.deltaTime;
                //Debug.Log(timerInaction);
                if (timerInaction > timeInaction)
                {
                    DisableMovement();
                    loopManager.BreakLoop();
                    StartCoroutine(FadeIn(false));
                    action = true;
                }
            }
        }

    }
    public void DisableMovement()
    {
        firstPersonController.paused = true;
    }

    public void EnableMovement()
    {
        firstPersonController.paused = false;
    }

    public void ActivateScreams()
    {
        audioScreams.Play();
    }

    private void DeactivateScreams()
    {
        audioScreams.Stop();
    }



    public void OpenSecondPortal()
    {
        if (!colliderSecondPortal.activeSelf)
        {
            colliderSecondPortal.SetActive(true);
            colliderSecondStop.SetActive(false);
            DissolvePrisoners();
            shaderSecondPortalDisabled.SetActive(false);
        }
    }

    public void OpenThirdPortal()
    {

        colliderThirdPortal.SetActive(true);
        colliderThirdStop.SetActive(false);
        thirdSecondPortalDisabled.SetActive(false);

    }

    public void CloseThirdPortal()
    {
        colliderThirdPortal.SetActive(false);
        colliderThirdStop.SetActive(true);
        thirdSecondPortalDisabled.SetActive(true);

    }
    public void ActivatePhysicSun()
    {
        physicalSun.SetActive(true);
    }
    public void DeactivatePhysicSun()
    {
        realTimeManager.running = false;
        physicalSun.SetActive(false);
    }

    public bool GetIsNight()
    {
        return globalTimeManager.isNight;
    }

    private IEnumerator FadeOut()
    {
        for (float i = 3; i >= 0; i -= Time.deltaTime)
        {
            // set color with i as alpha
            imgWhite.color = new Color(imgWhite.color.r, imgWhite.color.g,
imgWhite.color.b, i/3);
            yield return null;
        }

        imgWhite.color = new Color(imgWhite.color.r, imgWhite.color.g, imgWhite.color.b,
0);
        presentationDone = true;
        firstPersonController.paused = false;
        ManagerMenu.Instance.paused = false;
    }

    private IEnumerator FadeIn(bool loop)
    {
        imgBlack.gameObject.SetActive(true);
        for (float i = 0; i <= 3; i += Time.deltaTime)
        {
            // set color with i as alpha
            imgBlack.color = new Color(imgBlack.color.r, imgBlack.color.g,
imgBlack.color.b, i/3);
            yield return null;
        }
        imgBlack.color = new Color(imgBlack.color.r, imgBlack.color.g, imgBlack.color.b,
1);


        yield return new WaitForSeconds (2f);

        if (loop)
        {
            SceneManager.LoadScene("Init");
        }
        else
        {
            Debug.Log("QUIT");
            Application.Quit();
        }

    }

    private IEnumerator FadeInAndOut()
    {
        imgWhite.gameObject.SetActive(true);

        imgWhite.color = new Color(imgWhite.color.r, imgWhite.color.g, imgWhite.color.b,
1);

        yield return new WaitForSeconds(3f);

        for (float i = 3; i >= 0; i -= Time.deltaTime)
        {
            // set color with i as alpha
            imgWhite.color = new Color(imgWhite.color.r, imgWhite.color.g,
imgWhite.color.b, i/3);
            yield return null;
        }

        imgWhite.color = new Color(imgWhite.color.r, imgWhite.color.g, imgWhite.color.b,
0);
    }

    public void PlayGuillotine()
    {

        StartCoroutine(GuillotineSound());
    }

    private IEnumerator GuillotineSound()
    {
        yield return new WaitForSeconds(5f);
        guillotineManager.Execution();
        guillotineAudio.Play();
    }

    public void DissolveStatueSign()
    {
        StartCoroutine(DissolveStatue(statueSign, false, particlesDissolvePosition));
        statueSign.GetComponent<BoxCollider>().enabled = false;
        statueDisolved = true;
    }
    public void DissolvePrisoners()
    {
        for (int i = 0; i < freedSoulPositionsPrisoners.Length; i++)
        {
            StartCoroutine(DissolveStatue(prisoners[i], true,
freedSoulPositionsPrisoners[i]));
        }
    }

    private IEnumerator DissolveStatue(GameObject statue, bool freedSoul, Transform
positionFreedSoul)
    {
        float timeDissolve = 3f;
        float lerpValue = 0;
        if (freedSoul)
        {
            GameObject particlesInstanced = Instantiate (particlesDissolveEffect,
positionFreedSoul.position, particlesDissolveEffect.transform.rotation);
            Destroy(particlesInstanced, 10f);
        }

        // loop over 3 seconds backwards
        for (float i = 0; i <= timeDissolve; i += Time.deltaTime)
        {
            lerpValue = i / timeDissolve;
            statue.GetComponent<MeshRenderer>().material.SetFloat("_Mask", lerpValue);
            yield return null;
        }

        Destroy(statue);
    }

    public void PlayScreamsAndPlayer()
    {
        saveVoice.PlayPlayerVoice();
        ActivateScreams();
    }

    public void StopScreamsAndPlayer()
    {
        saveVoice.StopPlayerVoice();
        DeactivateScreams();

    }

    public void PlayFallDownJail()
    {
        fallDownJail.Play();
    }
    public void FinalFade()
    {
        StartCoroutine(FadeIn(true));
    }
    public void PlatonicFade()
    {
        StartCoroutine(FadeInAndOut());
    }
}
```