Enforcement of a realistic NPC and environment in order to develop a horror video game in Unreal Engine 4

Author:
Miguel Fernández-Montañés Domínguez

Advisor:
José Vicente Martí Avilés

Final Degree Project
Bachelor's Degree in Video Game Design and Development

Castelló, 2021

Acknowledgements

Firstly to my team partner Sara, for being a constant support.
To my family for being there always when I need them.
To Pablo Lorente, Miguel Ferrer, Àlex Beltran and my sister Silvia Fernández-Montañés for testing the video game.
And finally to our tutor, José Vicente Martí Avilés for his interest and worry of the project results.

Abstract

The hereby document represents the Final Report for a bachelor's thesis on Video Game Design and Development. The following work consists of the design and development of *La Vuelta*, a video game based on the creation of emergence narrative that is created at run time by a NPC (Non Player Character) whose behaviour changes depending on player's decisions. More specifically, the aim is to produce unique narrative experiences to enable the player to construct its own version of the story and to keep the player immersed in a realistic game world. The video game could be classified in the First Person Exploration genre with horror theme, it will be developed in Unreal Engine 4 for PC using systemic design and blueprints.

Key words

Video game, Unreal Engine 4 (UE4), NPC (Non Player Character), Immersion, Game Design.

Contents

Chapter 1

Introduction

In this chapter it is explained where the main idea of the project comes from and which were the first objectives and the motivation behind all the hard work and knowledge obtained during all the process. It is important to know that the nature of this project is collaborative, therefore this entire document will focus essentially on my fields of work, which have been the design and implementation of a Non Player Character (NPC), game mechanics and player interaction in addition to the game narrative and experience. Essential game concepts and features artistic related are relegated to the team component Sara Montagud Rodríguez in her project's memory.

1.1 Work Foundation and Objectives

At first, the goal of this project was to create a unique narrative horror experience with an AI that learns from users decisions and changes its patterns based on what it has learnt in every game. It was intended to be as realistic as possible, to immerse the player in the game environment to a level that produce a feeling close to psychological horror.

On the other hand, one of my personal goals is to become Game Designer and to get a job related to that field, therefore great part of the project consists of designing player's interaction, game rules and mechanics, which combined with the NPC and the environment generates the desired experience.

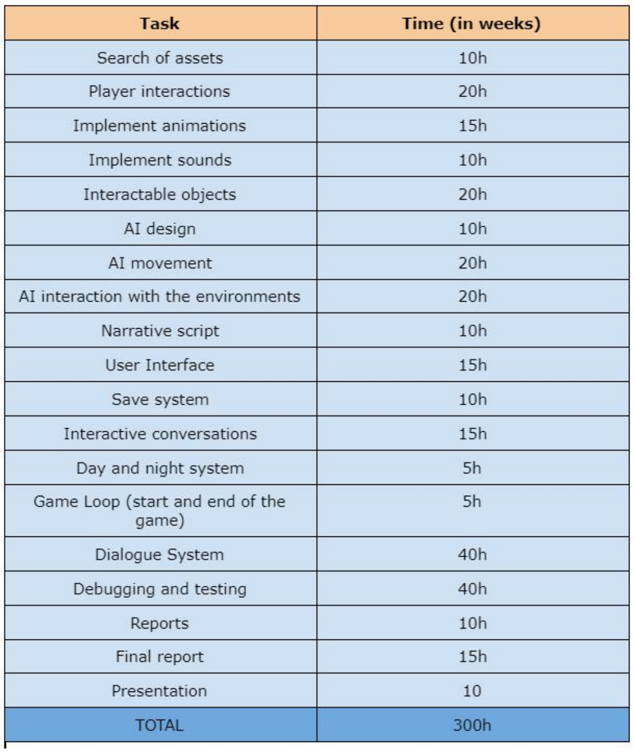
To accomplish this goals and also in order to improve and spread out our video games industry knowledge, the best engine was Unreal Engine 4 - in

Section 1.5 it is discussed in detail the reasons behind this decision - however this means tons of additional hours of learning, searching for information, testing and finding suitable assets for the project.

However the video games complexity does not end here, being a hybrid art and a interactive software they are composed by a wide range of fields, so in this project it has been attempted to reach to the most possible of them not only to learn but also to recreate the design and development of it.

1.2 Initial Planning

First of all, it is important to take into account that the planning has been changing along the development of the project, to adapt a the difference problems and circumstances that were rising up. **Figure 1.1** shows the different tasks with its corresponding hours of work and **figure 1.2** and **1.3** shows the diagram planning of them.



Task	Time (in weeks)
Search of assets	10h
Player interactions	20h
Implement animations	15h
Implement sounds	10h
Interactable objects	20h
AI design	10h
AI movement	20h
AI interaction with the environments	20h
Narrative script	10h
User Interface	15h
Save system	10h
Interactive conversations	15h
Day and night system	5h
Game Loop (start and end of the game)	5h
Dialogue System	40h
Debugging and testing	40h
Reports	10h
Final report	15h
Presentation	10
TOTAL	300h

Figure 1.1: Initial Planning

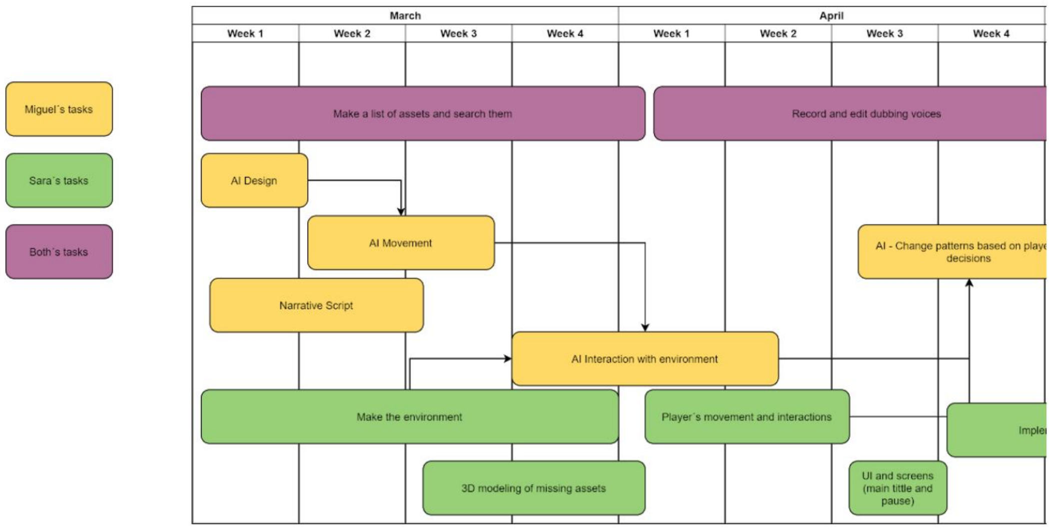

March
April
Week 1
Week 2
Week 3
Week 4
Miguel´s tasks
Sara´s tasks
Both´s tasks
Make a list of assets and search them
Record and edit dubbing voices
AI Design
AI Movement
AI - Change patterns based on playe
decisions
Narrative Script
AI Interaction with environment
Make the environment
Player´s movement and interactions
Impler
3D modeling of missing assets
UI and screens (main tittle and pause)

Figure 1.2: Diagram Planning 1

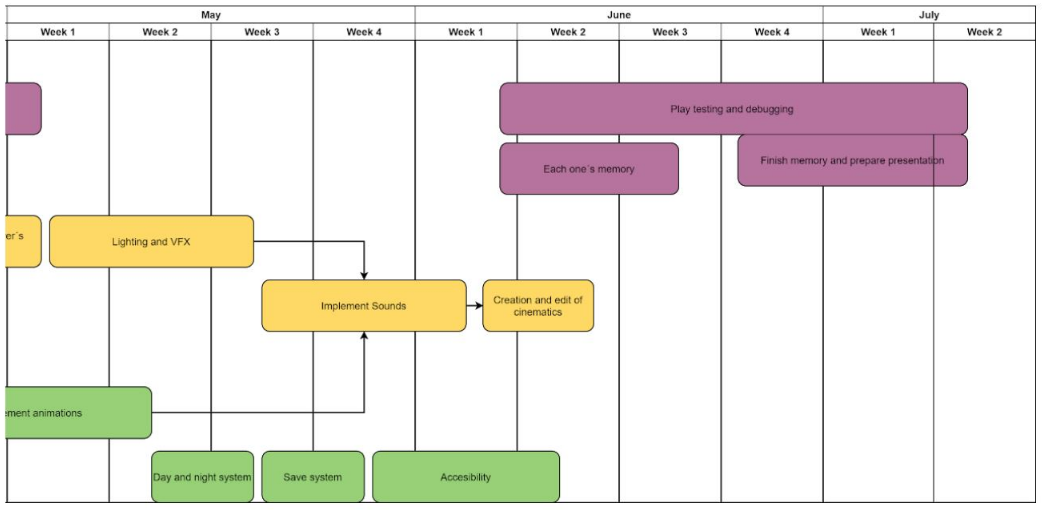


Figure 1.3: Diagram Planning 2

1.3 Related Subjects

- VJ1231 - Artificial Intelligence
- VJ1218 - Hypermedia Narrative and Video Game Analysis
- VJ1222 - Conceptual Design of Video Games
- VJ1226 - Character Design and Animation
- VJ1224 - Software Engineering

1.4 Estimated cost

In this section it will be into account the estimated cost of the project developed for four months of work (see **figure 1.2** and **figure 1.3**).

Taking into account that the entire project development would be approximately 265 hours (as shown in **figure 1.1** without considering non development tasks) and that, both team members would be considered as Junior level workers which could get paid around 7.5€ per hour, thus makes the total cost goes to 3.975€.

Furthermore, software licenses and additional resources should bear in mind. Although most part of the project is developed by the team, there is no time to accomplish all the project goals. Luckily, there are tons of free assets in internet such as 3D models, animations and sounds that have helped fulfill the aims. The reasons behind the choose of the software used are explained in **Section 5.2**. Finally here is a list of the software and resources used in the project and its cost:

- 3D Max (software): goes to 267€ each month, making a total of 1.068€
- Unreal Engine 4 (software): 5 per cent of the revenue (in this case 0€ because the video game will be not published)
- Krita (software): 0€
- Photoshop (software): 24,19€ each month
- Audacity (software): 0€
- Mixamo Animation Converter (software) [13]: 0€

- Mixamo Animations (resource): 0€
- Free 3D models from Epic Games Marketplace (resource) (see **Subsection 1.5.1**): 0€
- Udemy UE4 course (resource) [8]: 15€ in order to learn Unreal Engine 4 basic knowledge.

The total estimated cost would be 5.154€, which is quite low knowing that within the indie video game industry there are projects developed with much more budget.

1.5 Unreal Engine 4

This section is entirely dedicated to Unreal Engine 4 (UE4), the reasons why it was chosen over other video game engines, such as Unity, a brief overview of the engine's potential and varied features and finally its relevance during the design and implementation of the video game.

1.5.1 Engine Overview

Unreal Engine was created back in 1998 by *Epic Games* as an engine only for *First Person Shooters*. Although currently it is *"The world's most open and advanced real-time 3D creation tool"* as it is shown in the official web page [3] and not only for games but also for films, television, architecture or even for broadcasting live events. It has been used to develop video games such as *Fornite* (Epic Games, 2017), *What Remains of Edith Finch* (Giant Sparrow, 2017) or the recent *It Takes Two* (Hazelight Studios, 2021) moreover, it helped to build 3D environments in *The Mandalorian* (Jon Favrau, 2019).

Unreal Engine Interface

In **figure 1.4** it is shown how UE4 interface is divided. It has the typical 3D software environment such as Blender, 3DS Max or Unity. This familiar interface has helped with the first contact with the engine, allowing the focus in other features.

- 1: Used to place essential actors into the 3D world

- 2: The viewport
- 3: Actors placed in the world
- 4: Selected actor's details
- 5: Content inside the project

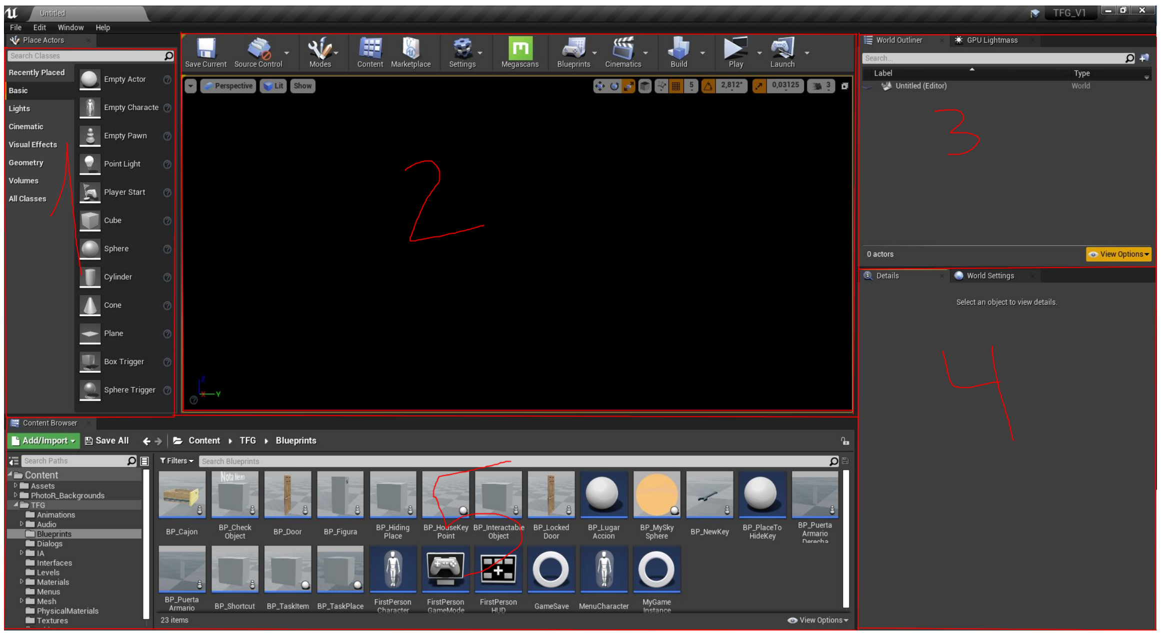


Figure 1.4: UE4 Interface

Marketplace and Assets

Epic Games provides to UE users tons of free content developed by third parties and other users, both paid (**see figure 1.5**) and free (**see figure 1.6**), with maximum quality and even AAA assets used in games such as *Paragon* (Epic Games, 2016). Also some materials and textures were taken from *Bridge*, a platform with many of 3D assets free to use in Unreal projects.

Figure 1.5: Ultimate River Tool (payment)

Figure 1.6: Project Nature - Foliage (free)

Unreal Learn

Unreal Learn [7] is a online learning web page where professionals from the industry upload courses completely free. There are courses for beginners and professionals who want to improve their knowledge about the engine. Also the courses are divided in distinct topics to fulfill the user interests. This smooths down the learning curve making UE4 so much accessible than it may appear.

1.5.2 Reasons to choose Unreal Engine 4

Firstly, deciding which video game engine to use for a project should be an easy task, simply it is decided by the knowledge about that engine, following this affirmation the chosen engine should have been Unity due to the fact that it is the engine that it is taught along the degree. However, being this a Final Degree Project in which some of its main goals are to investigate and to demonstrate teachers what the student is capable of, and obviously to learn, makes UE4 as the best option.

Furthermore, UE4 together with Unity are the most popular engines in the video games industry, though Unity it is known for its demanding request within the video games indie world, and in the other hand, UE4 it is known both in AAA and indie companies. Still it is arguable that UE4 is more powerful and more accessible than Unity as *Carlos Coronado*, designer and developer of *Mind: Path to Thalamus, 2014* and *Infernium, 2018*, both made in UE4, has said multiple times [8].

Most of the people who work as Game Designer have at least, a minimum knowledge about programming, nevertheless it is common to use visual scripting at this job. Luckily UE4 provides the option to develop games with *Blueprints* (see **Figure 1.7**) which is an excellent feature for designers and artists. All of this in addition to the facts explained in **Subsection 1.5.1** leave UE4 as one of the best options and the chosen engine for the project.

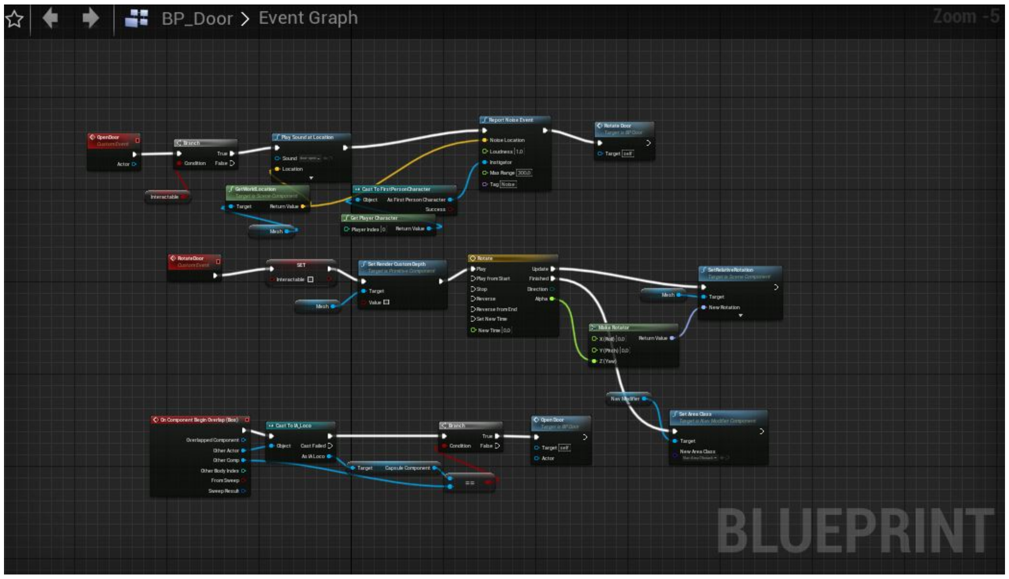


Figure 1.7: Door Blueprint

1.5.3 Relevance in the Project

What Unreal has bring to the project has been quality, experience in a different 3D software, performance and facility to implement the designs. All of this it is due to the fact that Epic Games has created a robust video game engine, not only for developers but also for designers and artists, making their job the easiest way possible. Even though the experience

For further details in technical development and results check **Chapter 5** and **Chapter 7** respectively.

Chapter 2

Research and References

This chapter will cover the understanding of the nature and sources of the project from various video game design related points of view.

2.1 Game concept

First of all, three fundamentals part of the game are the exploration, the movement and the camera, features typical from first person horror games like *Amnesia: The Dark Descent* (Frictional Games, 2010) (See **figure 2.1**). This makes the player feel closer from what is happening within the world, like he is inside of it.

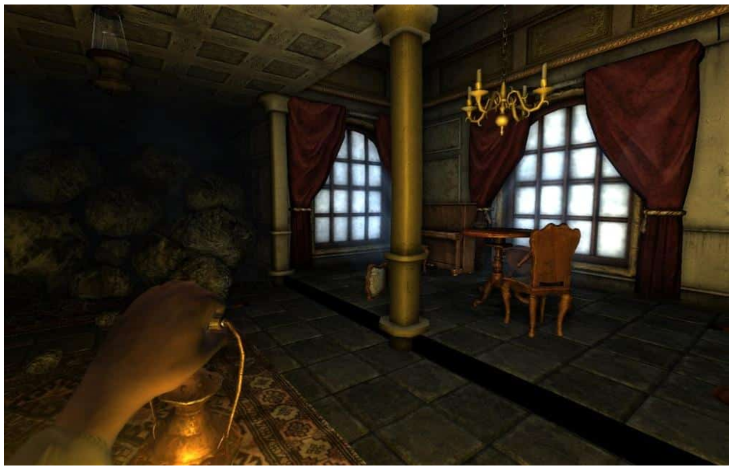


Figure 2.1: Amnesia: The Dark Descent. First person view.

However, though horror is the main theme of the video game, it is not intended that the player feels afraid but rather to feel psychological terror, a hard feeling to produce if some guidelines are not followed.

To finish this section, a clearly inspiration in the game was *Hello Neighbor* (Dynamic Pixels, 2017), which is a survival horror stealth game which generates the perfect *Hide and seek* experience creating an atmosphere with tension and exploration.

2.2 Player's Interaction

Popular games like *Resident Evil VII* (Capcom, 2017) (See **figure 2.2**), are the perfect example for the player's main interactions that are intended to achieve in the game. For instance, player's interactions with objects around the scenery, slow walk in order to make less noise, to open doors and pick up elements. They are easy to execute, to develop and most essential, intuitive.

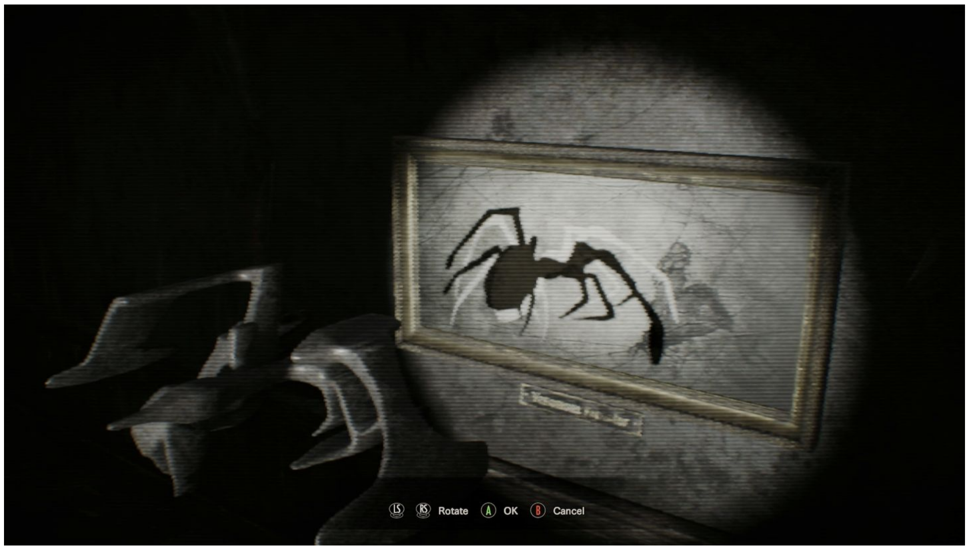


Figure 2.2: Resident Evil VII. Interaction with objects.

Also the interaction with the NPC who changes his behaviour based on player's decisions is inspired in the previously game mentioned game *Hello Neighbor* (Dynamic Pixels, 2017), and obviously the *Hide and seek* mechanic - **Section 5.5** enters in this mechanic with more exactitude -.

2.3 Player's Experience

One of the key creative aspects in the project is the replay and the looping system, which is explained in further detail in **Section 3.4** and **Section 4.2**. Nowadays the popular video game genre *Rogue Lite* is on everyone's lips, it consists in the constant repetition of the game but this changes some features each time it is played, for example the level or the items that the player can find.

The video game it is designed to be replayed between three to four times for the player to complete the entire game. In each game the player is inside the same house, but some things changes, even the dialogues with the NPC are not the same. Again, *Hello Neighbor* (Dynamic Pixels, 2017) does this in a way that the player learns on each replay, this works thanks to the systems behind all the designs. For instance, games such as *The Binding of Isaac: Rebirth* (Edmund McMillen, 2014) changes player's experience in each replay with different systemic levels and randomizing items drops and enemies, this is kind of similar to the random collectables and player's tasks in the project - check **Section 3.4** for more accuracy on this topic -.

Nevertheless, the video game is designed so that the best way to insert a *Procedural Narrative* [11] in the game loop game was to break the story in pieces, let the player find them and let him build his own version of the story, like the video game *Outer Wilds* (Mobius Digital, 2019) accomplish perfectly.

2.4 Game World

In the game it is intended to make feel the player familiar with the environment, a place easy to learn and to remember, but with many places to explore. One last time, in Hello Neighbor (Dynamic Pixels, 2017), all of his gameplay is around a small house (See **figure: 2.3**) which is perfect to develop and to recreate in a similar way within the project.

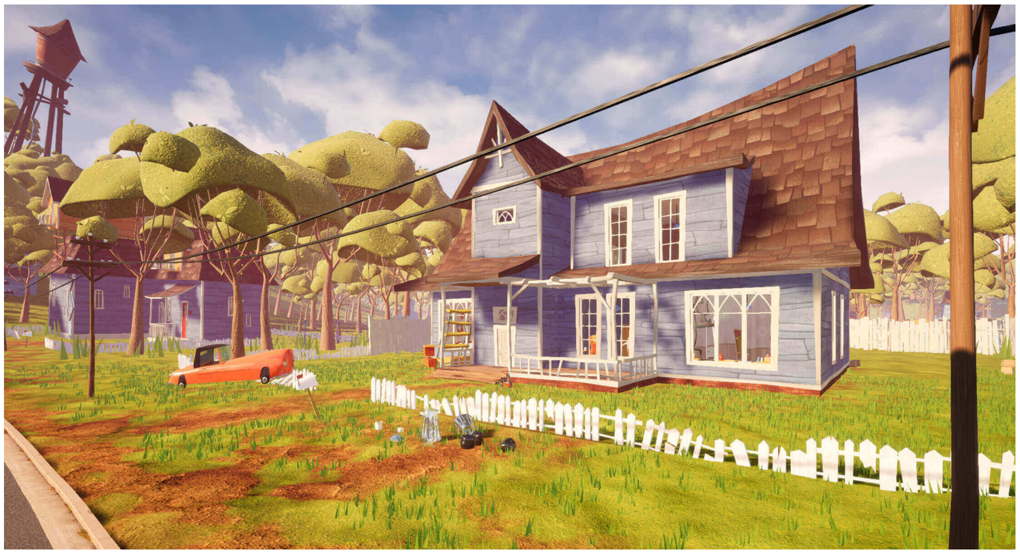


Figure 2.3: House from Hello Neighbor

Chapter 3

Game Design

In this chapter it is covered one of the essential parts of the player experience, how the rules and mechanics work together to accomplish the project's goals, because at the end, *video games are tools for entertainment* [9] , the functional and not functional requirements, the different systems in the game and the UI and UX.

3.1 Gameplay Overview

The game is about the exploration of the main character (controlled by the player) family house, to discover what happened, why the player is there and meanwhile the player must keep the house clean and take care of the NPC. Both the exploration and the interactions with the NPC are intended to be the most realistic as possible in order to immerse the player in the complete experience.

On one hand, player's mechanics are easy to understand and to execute. For instance, all the interactions are made with the same button, thus leave the player focus more on the game world and what is happening within. Also, the environment, what player sees, have to be attractive and faithful to reality.

In addition, gameplay must be engaging, with new stuff in every new replay and random events, again, to keep the immersion. The looping system explained later in **Section 3.4** accomplishes this goals creating different events with the NPC and interactions with the world, in each new gameplay.

To sum up, here it is a few goals that are intended to achieve with the design. Most of them have been thought taking into account concepts from Mark Brown's YouTube channel [1] :

- Keep the player engaged all along the game
- Create an realistic environment both with mechanics and rules to immerse the player
- Generate a feeling similar to psychological terror to the player
- Make an original system to create stories

3.2 Game Progression

In order to recreate the psychological horror atmosphere that it is desired and engage the player, the *Magic Circle* [4] of the game is shown progressively. This makes that step by step understanding a benefit to player's experience, discovering new dialogues with the NPC or new items inside of the house. The game narrative is completely attached to the game progression, as game goes on, player's comprehension about what is happening shows him how to unfold inside of the house, especially, how to treat with the NPC.

Each game has a duration of 15 minutes approximately. On the first play it is important that the player feels interested, to produce the necessity to play another time. Then the second time will be essential because the player must feel that he is learning and do not feel frustration, however the tension must be hold. For last, the final two games should depend explicitly on the player's execution, if the player has not understand the *Magic Circle* it is quite possible that the player gives up.

3.3 Requirement Analysis

Taking into account both previous sections in this chapter, within this section it is shown the functional and non-functional requirements of the video game.

3.3.1 Functional requirements

- The player can pause the game

- The player can return to the Main Menu
- The player can quit the game
- The player can pick up some items
- The player can check some items
- The player can open doors
- The player can talk to the NPC
- The player can choose a dialogue
- The player can walk slowly
- The player can hide from the NPC
- The player can take shortcuts
- The player can pick up tasks items

3.3.2 Non-functional requirements

- The player can change the game graphics
- The player can change the game volume
- The player can change the outline from interactable items
- The player can choose to show or not help messages
- The player can open the *Notes Menu*
- The game has realistic graphics
- The game will have a reasonable performance

3.4 System Design

As Aleissia Laidacker said [5], *"Systemic means there is a link between all the systems in your game. They have been developed and designed with the intention that one can influence the other"*. This has been the main goal of all the designed systems shown next. In addition to that, the systems has been designed following Michael Sellers guides [10].

3.4.1 Game loop

The next **figure 3.1** shows how the main game loop works, it defines the player's experience. Basically the player can explore the house, complete a task assigned by the AI or interact with it. All of this is inside the big circle, the main loop, where all the gameplay takes place. Taking into account that the video game has multiple finals, the moment that something triggers one of them, then *Reach Final* node will be triggered and the flow will get out of the circle, and finally, the game ends.

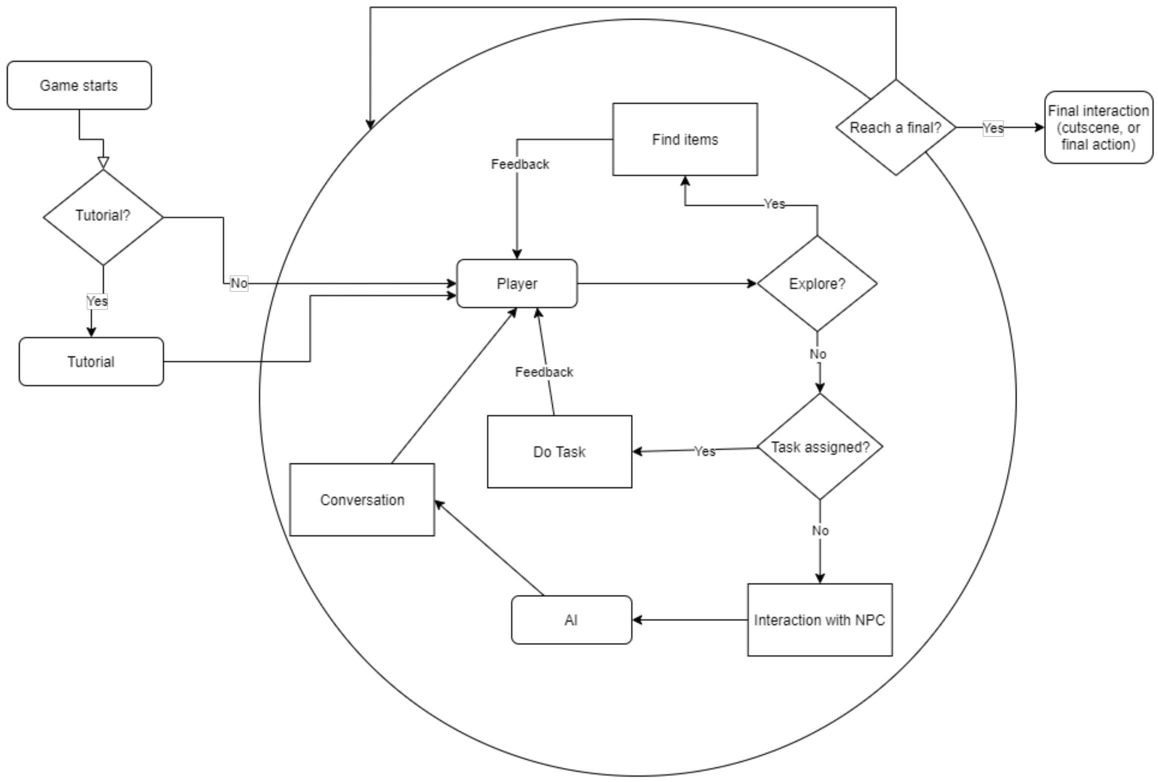


Figure 3.1: Main Game Loop System. Created with diagrams.net

3.4.2 AI and dialogue system

Figure 3.2 shows the AI system with the Dialogue System within, which is directly related to the AI system because it is who decides what lines of dialogues are shown.

The AI has priorities in its conditions when it comes to deciding which task is going to execute, from left to right, starting from the condition "Madness attack?" which it is the most prioritized. If it is executing a task and a more prioritized condition is triggered, the AI will stop doing that task and start doing the new one. In this way the behaviour is unpredictable because conditions are triggered by RNG (Random Number generator), calculated meticulously in order to accomplish a balanced game end time, this means, there is time to complete the game. However once the AI triggers its order to kill the player, when it reaches its maximum madness value, it will do it (as is the most prioritized condition), thus game will reach one of the finals.

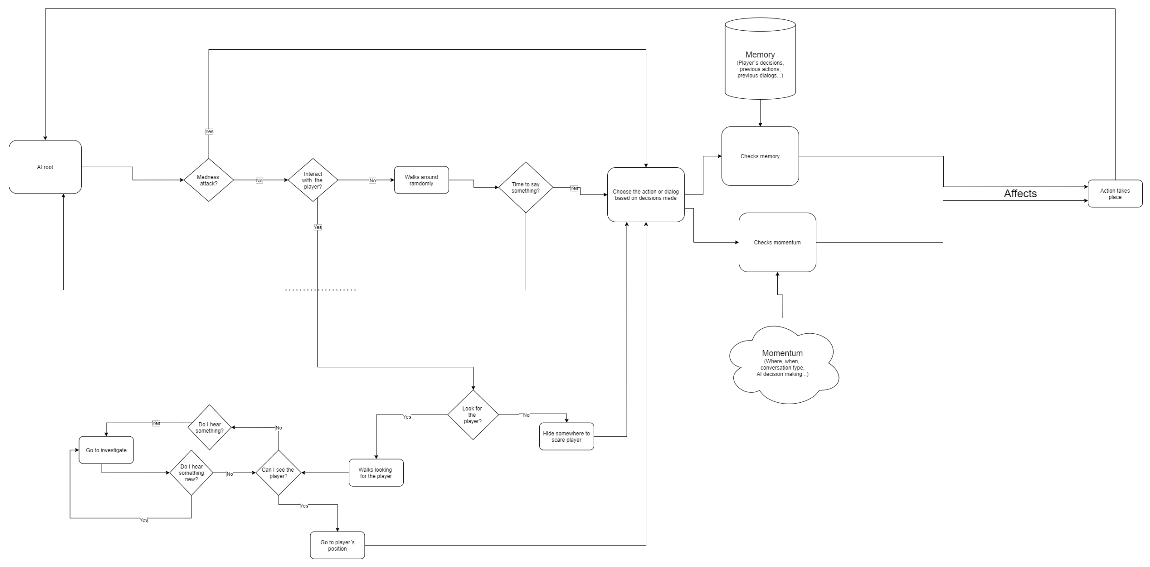
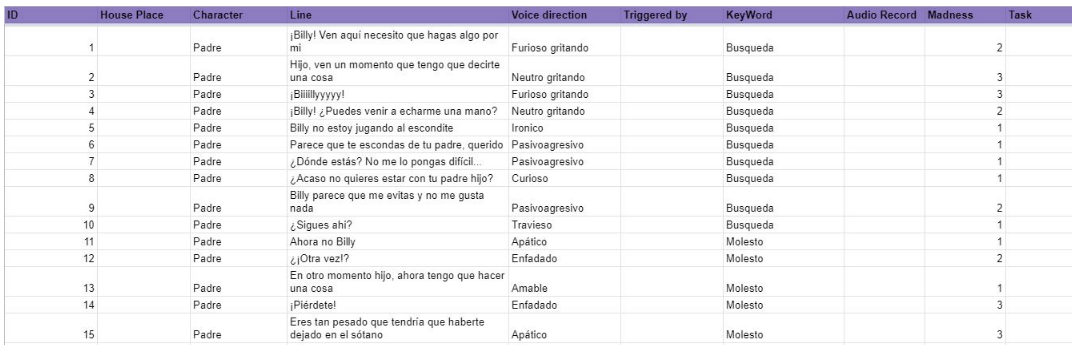


Figure 3.2: AI and Dialogue System. Created with diagrams.net

Once a condition is triggered and the time to choose an action or a dialogue comes, it is when the dialogue system comes into play. Based on AI memory (player's decisions, previous actions/dialogues) and momentum (where, when, conversation type - check **4.2** -) an action or dialogue is chosen from the dialogue data base (**see figure 3.3**). The AI actions are designed to be like a test for the player in order to see if he is understanding the mysteries inside the house. If the player answers wrongly the AI madness will increase, if not, nothing happens and the player can keep exploring.



ID	House Place	Character	Line	Voice direction	Triggered by	KeyWord	Audio Record	Madness	Task
1		Padre	¡Billy! Ven aquí necesito que hagas algo por mi	Furioso gritando		Busqueda		2	
2		Padre	Hijo, ven un momento que tengo que decirte una cosa	Neutro gritando		Busqueda		3	
3		Padre	¡Biiiillyyyyy!	Furioso gritando		Busqueda		3	
4		Padre	¡Billy! ¿Puedes venir a echarme una mano?	Neutro gritando		Busqueda		2	
5		Padre	Billy no estoy jugando al escondite	Ironico		Busqueda		1	
6		Padre	Parece que te escondas de tu padre, querido	Pasivoagresivo		Busqueda		1	
7		Padre	¿Dónde estás? No me lo pongas difícil...	Pasivoagresivo		Busqueda		1	
8		Padre	¿Acaso no quieres estar con tu padre hijo?	Curioso		Busqueda		1	
9		Padre	Billy parece que me evitas y no me gusta nada	Pasivoagresivo		Busqueda		2	
10		Padre	¿Sigues ahi?	Travieso		Busqueda		1	
11		Padre	Ahora no Billy	Apático		Molesto		1	
12		Padre	¿¡Otra vez!?	Enfadado		Molesto		2	
13		Padre	En otro momento hijo, ahora tengo que hacer una cosa	Amable		Molesto		1	
14		Padre	¡Piérdete!	Enfadado		Molesto		3	
15		Padre	Eres tan pesado que tendría que haberte dejado en el sótano	Apático		Molesto		3	

Figure 3.3: Dialogues Data Base in Google Spreadsheets

The AI has also three different mood states:

- Normal: nothing changes, it executes its conditions normally.
- Upset: the AI could be upset if it finds the player inspecting items from the house. Then it will not accept interactions from the player during a short period of time and also the dialogues change.
- Insane: when a madness action has been triggered the AI starts doing a specific action and will not stop doing it until the player settle the conflict.

Furthermore AI has multiple triggers, other systems in the game, that influence its behaviour and also can interact with them, obviously, the player is one of them, but also the doors. The AI can open doors if they are not locked, if so it can not enter to that room, however once the player finds the key and opens that door, the AI will be able to enter in the room - also known as dynamic pathing -. In addition, if the AI sees one of the narrative objects that are around the house it will vary its *pathing* and dialogues.

3.4.3 Game flow

The next figure shows the game flow diagram, starting from ”Execute game” node. It is intended that there is less screens as possible between start game, playing and quitting game in order to do not frustrate the player.

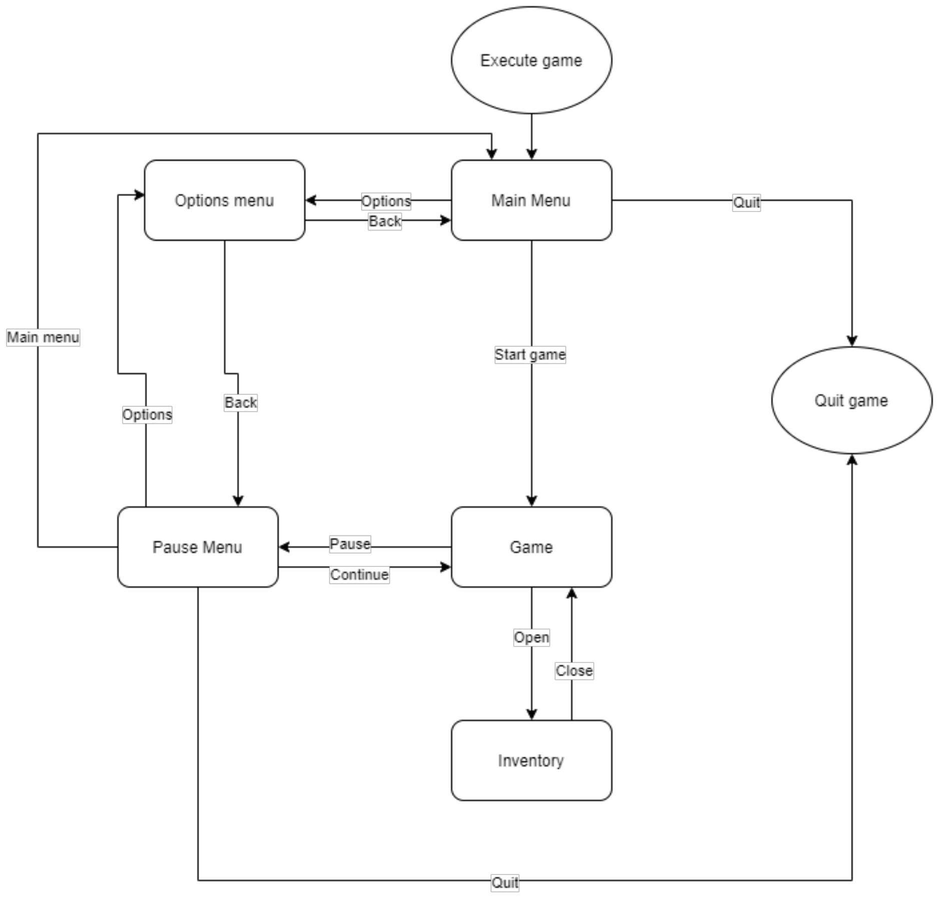


Figure 3.4: Game flow. Created with diagrams.net

3.5 User Interface and User Experience

In this section it is explained and shown how it is the UI and reasons behind the design. Also UX will be explained along the mockups functionality. It should take into account that the lines in the bottom of the menus (See

figure 3.5) are for controlling space between different elements, they will not be shown in the final interface.

3.5.1 Main menu

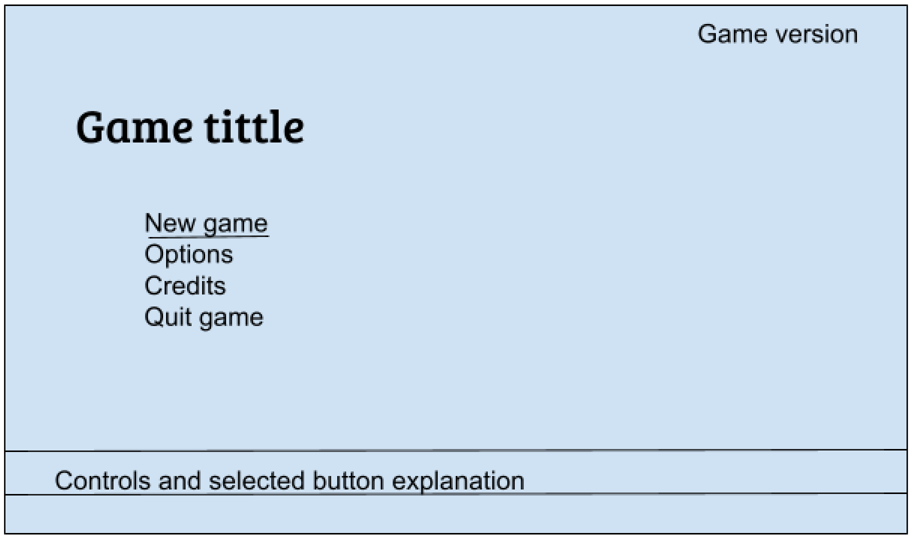


Figure 3.5: Main game menu mockup

The main game menu is what player will see first so it has to be the most attractive as possible. In the blank space there will be a realistic 3D scenery related to the game.

- Functionality: If the player hovers a button with the mouse, a line will be shown below the button and will disappear from the other one if this is the case, also in this exact moment a sound will be played. Depending on the hovered button, an explanation will be shown to indicate what the button does.
- Controls: They will be shown next to the button explanation. Mouse movement to select buttons and left click to triggers them.

3.5.2 Options menu

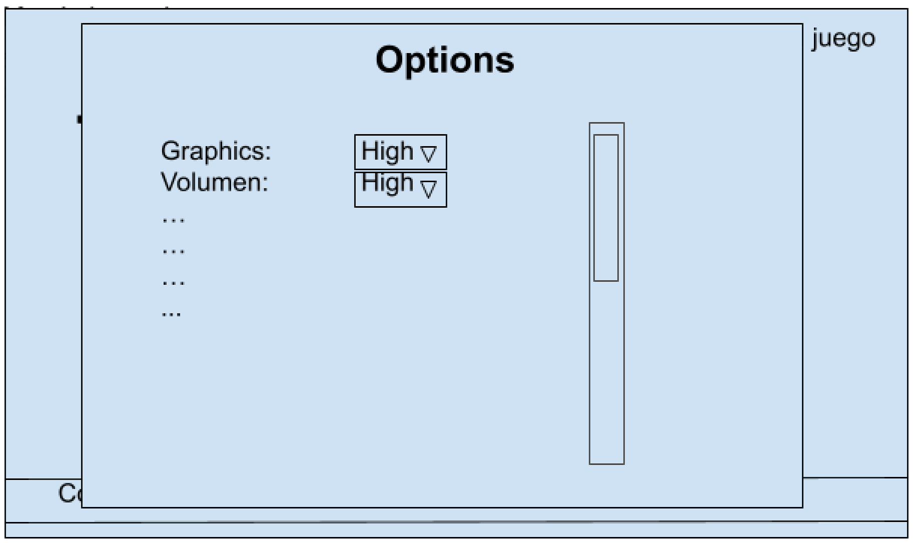


Figure 3.6: Options menu mockup

- Functionality: It is only accessible from main menu and from pause menu. Options menu will appear in the same menu but overlapping the current menu with the options menu window. Player can change an option pressing drop down and selecting the desired value and a sound will be displayed. Options will be displayed in a vertical list in order to add the number of options required.
- Controls: Same controls as explained in **Subsection 3.5.1** in addition for using the mouse wheel to move the scroll bar, and therefore to descend the list.

3.5.3 Pause menu

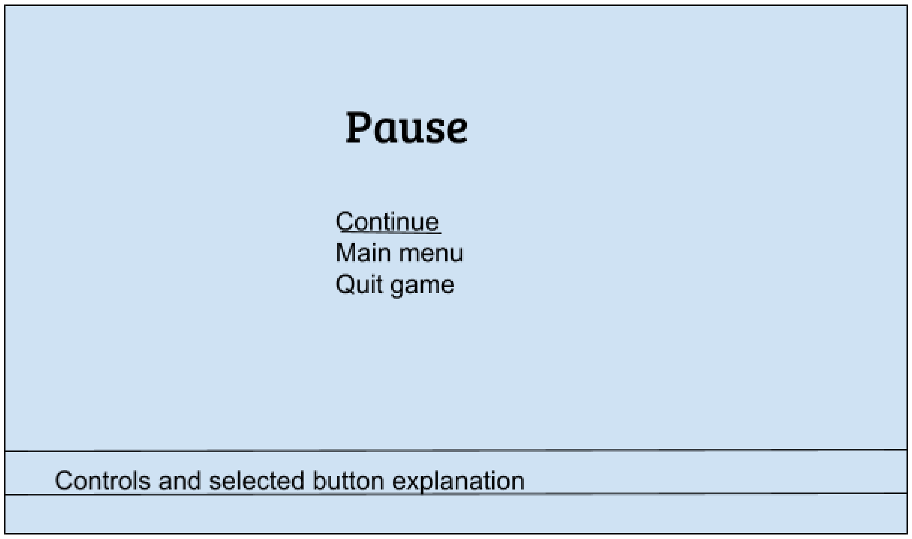


Figure 3.7: Pause menu mockup

- Functionality: It is only accessible during gameplay. Once the player access to the pause menu, the game will be paused. Rest of functionallity works the same as explained in **Subsection 3.5.1**.
- Controls: Same controls as explained in **Subsection 3.5.1**.

3.5.4 Inventory

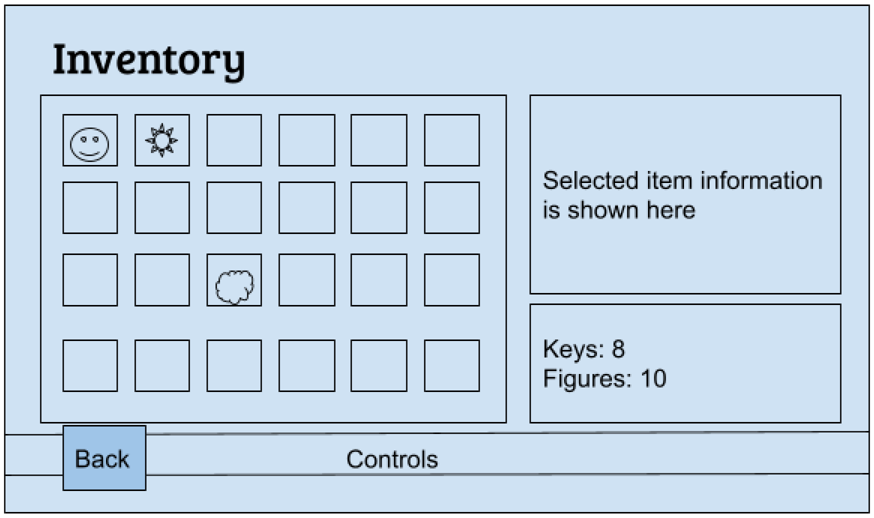


Figure 3.8: Inventory mockup

- Functionality: It is only accessible during gameplay. Once the player access to the inventory, the game will be paused. In this menu it is shown the player's inventory and how many keys and figures player has. If the player left clicks an item, information related to that item will be shown on the right. Back button will close the inventory and return to the game.
- Controls: Same controls as explained in **Subsection 3.5.1**.

3.5.5 In game UI

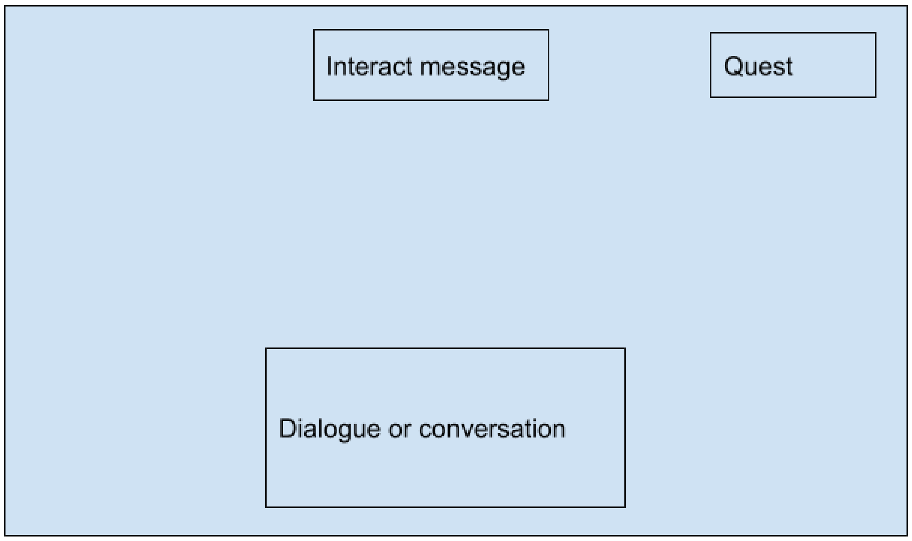


Figure 3.9: In game UI mockup

This interface it is shown only during gameplay, although it is not interactable it is important to see its composition. It is intended to be less intrusive as possible so the player will focus only on the game. The player can deactivate both interfaces from the top in the options menu in order to increase the immersion.

In case the player starts a conversation with the NPC, a dialogue box will be shown (See **figure 3.10**) in the bottom of the screen as shown in **figure 3.9**. See more about this topic in **subsection 5.5.3**.

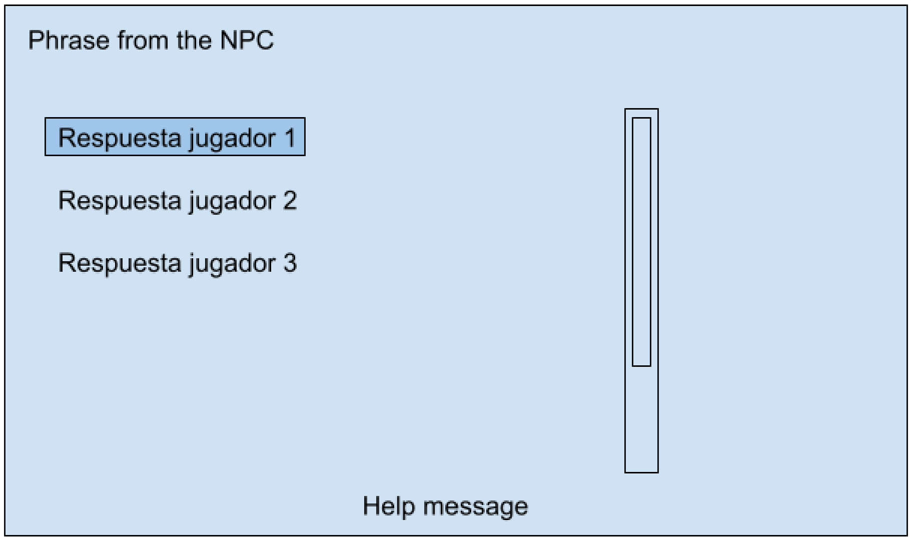


Figure 3.10: Dialogue Box mockup

- Functionality: It will only appear when the player talks to the NPC. It will stop the AI behaviour. A help message will be shown to indicate the player what button has to press to continue reading the conversation. When the player has to select a response to the NPC, responses will be shown in a vertical list.
- Controls: "E" to continue reading, left click to select a response and mouse wheel to scroll down the scroll bar.

3.6 Level Design

This is a brief section about the level design of the game and how it benefits the player´s experience.

As it is shown in **figure 3.11** and **figure 3.12** every room in the house has narrative elements - written in red in the figures - therefore the player will never get lost and always will find something to do.

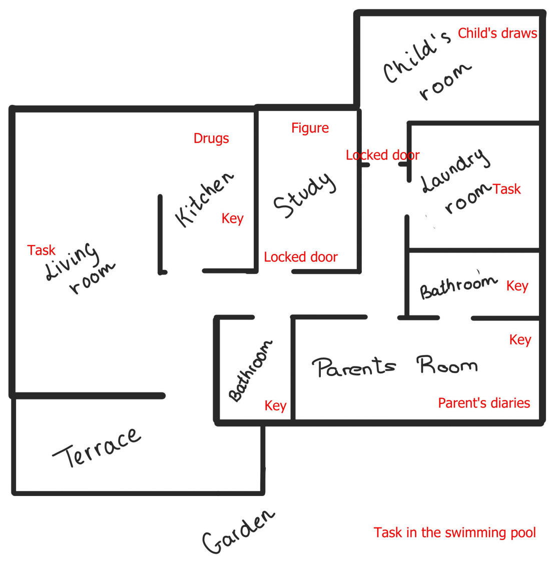


Figure 3.11: First Floor Level Design

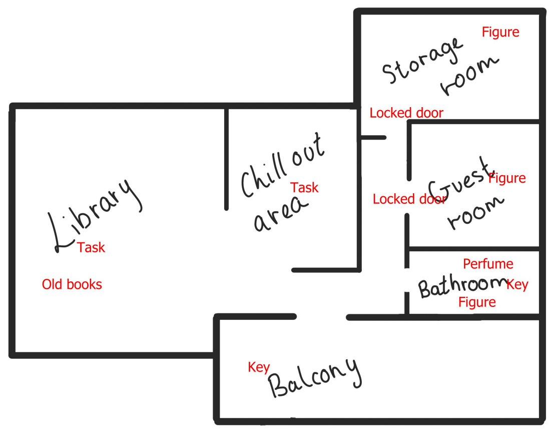


Figure 3.12: Second Floor Level Design

Lighting is an essential part of the level design, it helps to highlight key places that need player's attention because them by alone are not enough. For instance, in the second floor (See **figure 3.12**), in the library, there is no much light but in a specific spot where the old books are hidden there is a spot light to get player's focus.

Chapter 4

Narrative Design

This chapter will be dedicated to the player's immersion and experience, the main narrative aspects of the video game will be discussed in depth, as well as the narrative structure and how it influences the player's perspective and understanding progression.

Although the base storyline it is an important part of the project, and sometimes depending on player experience could be the heart of the game, it is not concerned in this project report. Nevertheless, in this section it is explained how with few resources an interactive story is created, making the gameplay a memorable and unique experience [6].

4.1 Procedural narrative

Trying to create a full emergence narrative is a though task [12], this is the reason why in this project the narrative is constructed by procedural narrative [11] which progresses with player and NPC's decisions at runtime. This means, the base storyline will be broken into pieces and spread out across the game world, leaving the player discover those pieces step by step. However sometimes the AI will create emergence events which will contribute to the experience with a varied points of view and making it unique for each player.

4.1.1 How narrative is spread?

One of the main mechanics of this project is the exploration, the player will be rewarded by inspecting around the house, looking for multiple narrative elements. The narrative comes from two different parts, one of them is the narrative elements, the more the player discover, the more he will understand about its duty in the game. The second one is from the AI, who transfers information through the dialogues (See **figure 3.3**) and actions.

However, the player can feel frustration if he is not comprehending what it is going on within the game, therefore he has to receive new feedback constantly. To avoid that, narrative items not only provides player with some information about the house and his father (the NPC), but also they unlock new conversations and questions with the NPC once the item is discovered, for this reason player will obtain enough feedback about the story but also without feeling overwhelmed.

4.2 Run time Narrative

Run time Narrative is a concept used to refer to the narrative that is created by various events generated at run time. For instance, in *Middle-earth: Shadow of Mordor* (Monolith Productions, 2014) "*the story evolves unpredictably - because its entirely based on your interactions with the orcs. It is constructed at run time, encounter by encounter, using your interactions as narrative brickwork*" as Mark Brown says [2], and this is the goal that it is intended to achieve with the encounters with the NPC but in a smaller way.

As explained in **Subsection 3.4.2**, all the decions the AI makes are based on the current situation and previous decisions and encounters with the player - with RNG inside on conditions - causing a tiny version of the *Nemesis System* from *Middle-earth: Shadow of Mordor*(Monolith Productions, 2014).

Chapter 5

Functional and Technical Specifications

During this chapter it is explained how it is managed to find, understand and use a wide range of tools - mainly UE4 related - to help with the implementation of the designed ideas for the video game. In addition to the implementation of the Artificial Intelligence and the Dialog System, both truly important parts of the project.

5.1 Requirements and recommendations

These are the minimum requirements for playing the video game.

- OS: Windows Vista SP2 64-bit or later.
- CPU: Intel i3 2125 3.30 GHz or later
- RAM: 2 GB
- GPU: GeForce GTX 750/AMD Radeon 7790 or later
- Storage: 5 GB free
- CPU and an OS 64 bits.
- Mouse and keyboard for controls.

Nevertheless if the player is experimenting low performance or the player does not accomplish the system requirements, it should be considered to try reduce the graphics in the video game options menu. Also, as the project uses Raytracing, a RTX GPU will increase the performance hugely.

It is recommended to use headphones for a more effective immersion as they improve the quality of the sounds as well as the identification of the sound source through 3D sounds.

5.2 Tools

These are the tools that were used during the development of the project:

- Unreal Engine 4 v.4.26.2: Game engine to develop the project
- 3DS Max 2020: for 3D modeling and materials
- Audacity: for sound design
- Krita and Photoshop: for textures and interfaces
- Quixel Bridge and Epic Games Marketplace: to search assets
- Autocad: to create the level of the house
- PureRef: to help with 3D models references
- Mixamo Animation Converter: for retargeting mixamo animations to UE4 skeleton [13]
- Google docs: for documenting
- Google spreadsheets: for dialogues and balancing
- Diagrams.net: for designing the systems

5.3 Player Movement and Interaction

5.3.1 Player controls

At the beginning of the development it was intended to implement gamepad controls to the video game, however due to the way of implementation of some mechanics more time was needed to implement the gamepad controls. Nevertheless, here are shown both of them:

- WASD keys // Left joystick: Player´s movement.
- Mouse movement // Right joystick: Camera control.
- E key // X button : To select and interact with different functionalities such as dialogs, items interactions
- Mouse left click // A button : To select in menus and dialogs, house tasks
- B button: To cancel or go back to the previous menu
- Shift key // B button: To slow walk inside of the game
- I key // Y button: Access to inventory
- P key // Start button: Access to pause menu

5.3.2 Player movement

Player movement was an easy task to do due to the fact that Unreal Engine 4 has a template for first person characters (See **figure 5.1**) in which the player movement and camera control was already implemented. It is composed by hands and a gun that are not needed in this project and some more stuff that were simply deleted.

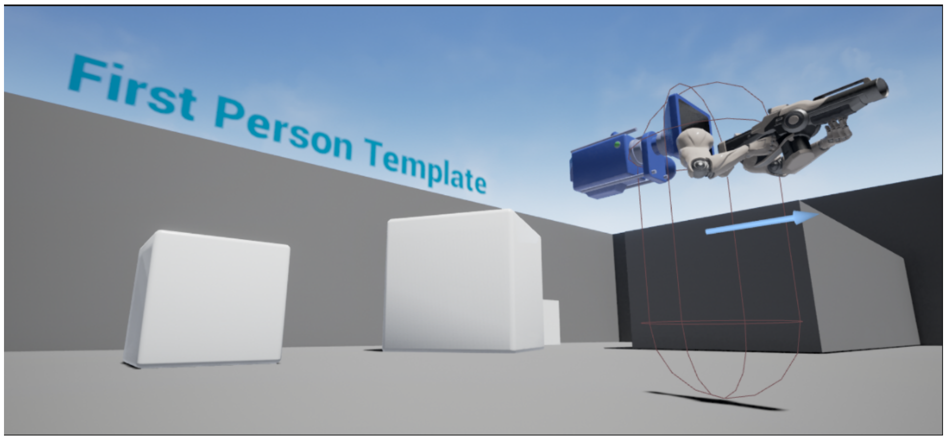


Figure 5.1: First Person Template UE4

Nevertheless the player needs a mesh to be able to collide with the floor and produce footsteps sound (See **Subsection 5.7**) when he is walking but it will be invisible during gameplay.

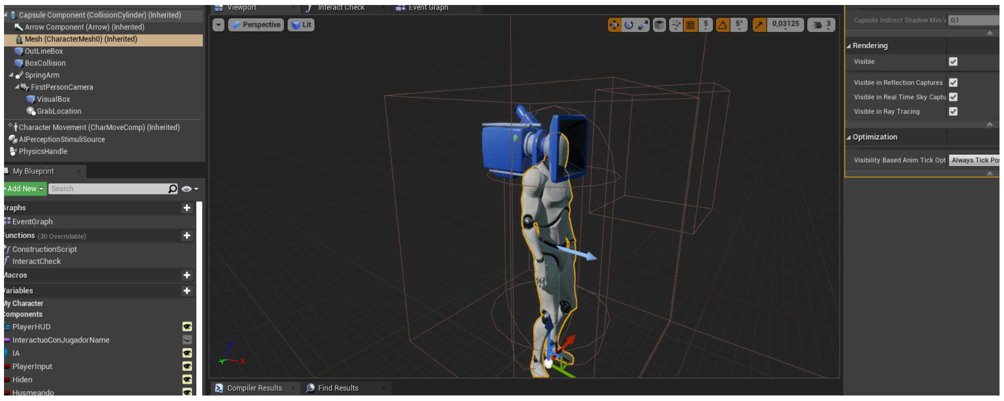


Figure 5.2: FirstPersonCharacter Blueprint Viewport - Mesh Visible

5.3.3 Player interactions

Basically, all the interactions are checked firstly by the node *LineTraceBy-Channel* which throws a raycast in the camera direction and gets the actor's type. If it has implemented an *Interact Interface*, that means the actor is interactable and the proper interact message will be shown (if the player has not deactivated it in options menu). Then depending on the actor the input and the element, the interaction will be different (See **Subsection 5.3.1**).

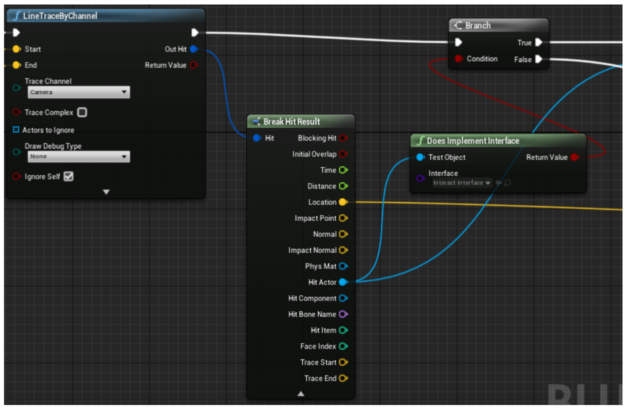


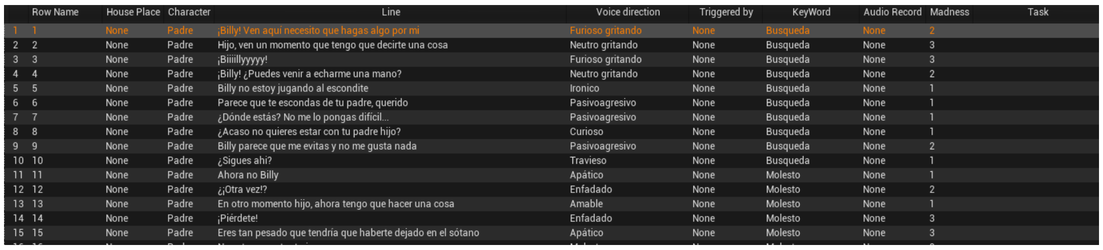
Figure 5.3: Interact Check Function

5.4 Dialogue System

One of the main problems of the *Procedural Narrative* is to be able to manage all the narrative content due to the fact that the player should not feel frustration for not receiving feedback or overwhelmed with too much information. In this sections it is explained how the dialogue system is organize in order to solve the narrative management problem and how it is implemented in UE4.

As shown in **figure 3.3** the dialogues are managed in a *Google Spreadsheet* divided in columns. Each column has a variable that the AI will have into account when it is selecting the appropriate dialogue. One of the most essential columns is the *Keyword* column, it is used to recognize what type of dialogue is, the choice of the *Keyword* is determined by the topic of the conversation.

The reason to choose *Google Spreadsheet*, is not only to manage comfortably the dialogues, but also because the multiple variables of each dialogue can be edited easily. In addition to that, when the spreadsheet is exported to UE4, it is imported as a *Data Table* (See **figure 5.4**).



	Row Name	House Place	Character	Line	Voice direction	Triggered by	KeyWord	Audio Record	Madness	Task
1	1	None	Padre	¡Billy! Ven aquí necesito que hagas algo por mi	Furioso gritando	None	Busqueda	None	2	
2	2	None	Padre	Hijo, ven un momento que tengo que decirte una cosa	Neutro gritando	None	Busqueda	None	3	
3	3	None	Padre	¡Biiiillyyyyy!	Furioso gritando	None	Busqueda	None	3	
4	4	None	Padre	¡Billy! ¿Puedes venir a echarme una mano?	Neutro gritando	None	Busqueda	None	2	
5	5	None	Padre	Billy no estoy jugando al escondite	Ironico	None	Busqueda	None	1	
6	6	None	Padre	Parece que te escondas de tu padre, querido	Pasivoagresivo	None	Busqueda	None	1	
7	7	None	Padre	¿Dónde estás? No me lo pongas difícil...	Pasivoagresivo	None	Busqueda	None	1	
8	8	None	Padre	¿Acaso no quieres estar con tu padre hijo?	Curioso	None	Busqueda	None	1	
9	9	None	Padre	Billy parece que me evitas y no me gusta nada	Pasivoagresivo	None	Busqueda	None	2	
10	10	None	Padre	¿Sigues ahí?	Travieso	None	Busqueda	None	1	
11	11	None	Padre	Ahora no Billy	Apático	None	Molesto	None	1	
12	12	None	Padre	¿¡Otra vez!?	Enfadado	None	Molesto	None	2	
13	13	None	Padre	En otro momento hijo, ahora tengo que hacer una cosa	Amable	None	Molesto	None	1	
14	14	None	Padre	¡Piérdete!	Enfadado	None	Molesto	None	3	
15	15	None	Padre	Eres tan pesado que tendría que haberte dejado en el sótano	Apático	None	Molesto	None	3	

Figure 5.4: Data Table from UE4

Firstly, the node *Get Data Table Row Names* has to be called to access the Data Table, then *Get Data Table Row* is called for each row name from the Data Table and finally they are stored in different arrays by its *Key word*, in order to access them in the *Event Graph* - where blueprints are called - or in other blueprints (See **figure 5.5**).

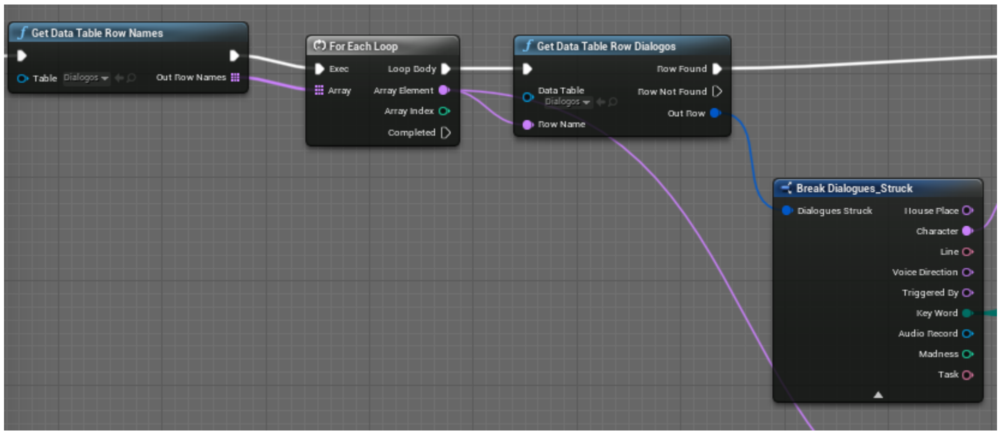


Figure 5.5: Accessing the Data Table in the Event Graph

Now for the purpose of accessing an exact dialogue topic, it has to use the corresponded array directly, and to call again *Get Data Table Row* node for each element (all are row names from the Data Table) (See **figure 5.6**). Once the appropriate row is selected (the choice depends on the momentum and memory of the AI (see **Subsection 3.4.2**), this row is removed from the array and then added to the end of it. In this way it is assured that given a specific situation, the dialogue will not be repeated until there is no more new dialogues which adequate to the given circumstances.

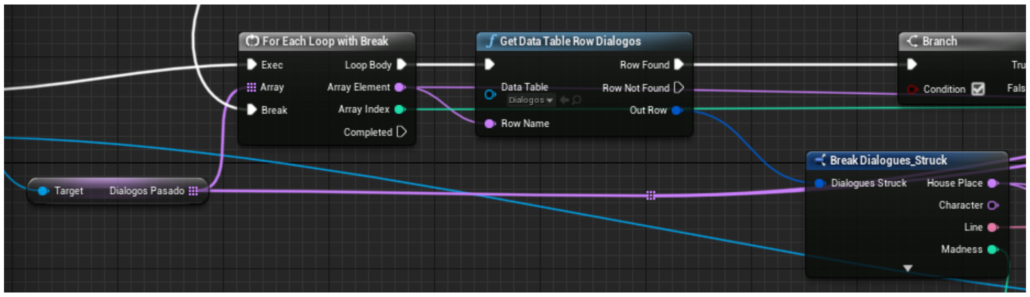


Figure 5.6: AI Task Display Line - Accessing Dialogues

5.5 Artificial Intelligence

In this section it is explained the most important aspects of the AI, how they are implemented in UE4 and crucial features that have to be shown.

5.5.1 Behaviour Trees

Following the design in **subsection 3.4.2** the best way to implement it was with UE4 behaviour trees, although it could be done with blueprints, one of the main goals of this project is to learn as much as possible from this awesome video game engine and take as much advantage as possible of its possibilities.

Behaviour trees in UE4 are composed by two different but connected elements, the behaviour tree himself and the blackboard. In the blackboard is where all the variables are stored and the behaviour tree can access them easily through blueprints.

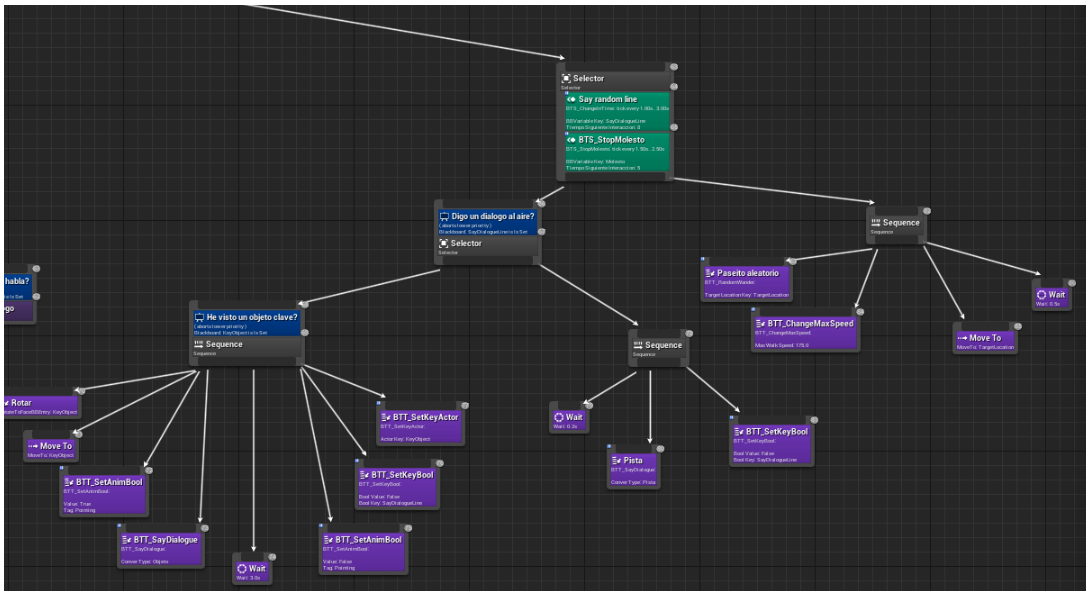


Figure 5.7: Branch from AI Behaviour Tree

The flow starts from the root node and goes through the node trees from left to right checking the *Decorators* (the blue boxes) (See **figure 5.7**), if the condition is true, the flow will enter that branch, if not it will continue checking the right nodes. Green boxes are *Services*, which function is to change variables from the blackboard when the flow reaches the node they are inside. And finally the purple boxes are the tasks or executions of the AI.

Knowing this, implementing the designed AI system was tedious but intuitive. Nevertheless, there are more complex actions from the AI that can not be placed in the same behaviour tree, mainly, in order to keep the tree comprehensible. To do this, more behaviour trees were made for smaller tasks, such as the conversation with the player, AI actions (kill the player or sleepwalking) or wait for the player to finish an assigned quest.

In **Figure 5.8** the *Asignar Tarea* node assigns a quest to the player and then the *Espero tarea* node is executed, but it has a *Decorator* with a conditional loop, that means until a variable is set, in this case *TaskDone*, the tree's flow will execute that node. In this way it is assured that meanwhile a behaviour its being executed, the others are not. The rest of the behaviour and actions are implemented very similar as explained.

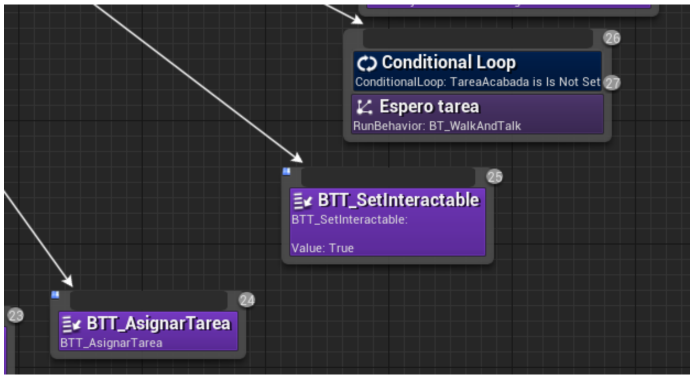


Figure 5.8: Assign task to player then wait

5.5.2 Perception

The AI perception is a fundamental aspect, on one hand it has to see the narrative objects which are the game world in order to interact with them, and in the other hand, it has to see and to listen to the player thus it can reach him when it is looking for him. The AI has a *AIPerception* component which gives it the option to add multiple senses, in this case sight and hearing. Also the actors that will be sensed by the AI, will need a *AIPerceptionStimuliSource* in order to report a stimuli to the AI.

For instance, in **figure 5.9**, it is shown how the player reports a noise event depending on his velocity - as in real life as faster someone is walking noisier will be the steps - and if this normalized velocity exceeds a given threshold, the player will report the noise in his position with the normalized velocity as value of loudness (which goes from 0.0 to 1.0) and the tag *Noise* to differentiate from sights reports.

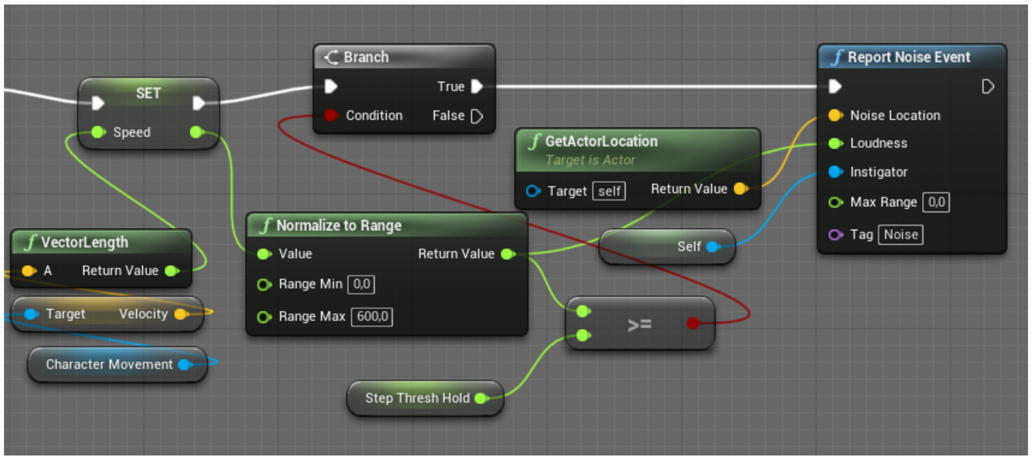


Figure 5.9: Report Noise Event - FirstPersonCharacter

Afterwards, if the AI is in range, in this case, to hear the stimuli the player has reported, the node *On Target Perception Updated* will be called (See **figure 5.10**). If the *Percieved Actor* is the player then it is needed to know if the stimulus was sensed by sight or noise, then variables from the *Blackboard* must be changed therefore the *Behaviour tree* will do the corresponding tasks. If it is not the player, that means the AI has seen a narrative object, it will update the appropriated *Blackboard* variables related to seen an object.

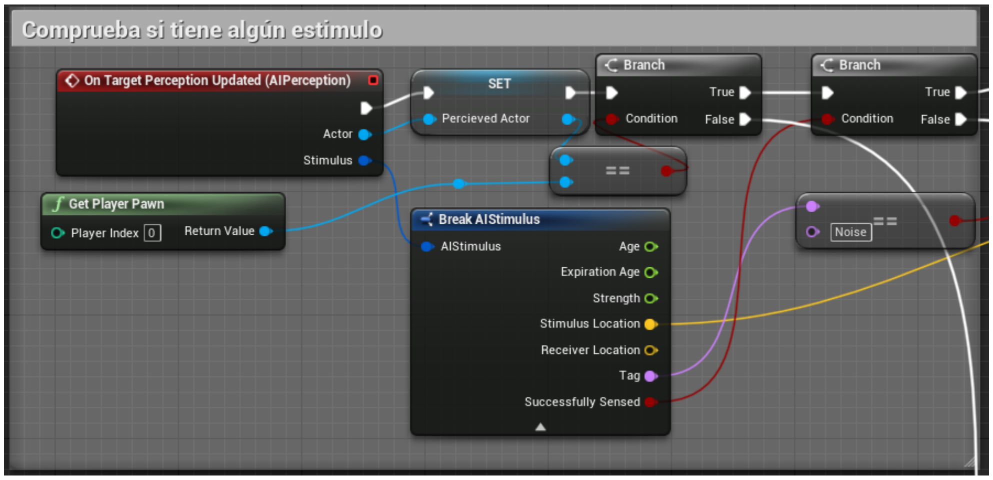


Figure 5.10: AI Perception Updated - AI

5.5.3 Conversations

The conversations with the player has the typical tree structure. Firstly, the AI chooses the *Starting dialogue* which introduces the player to the conversation. Then given a number of different responses for the player, for instance, A, B or C, if the player selects the A, the AI will respond with its correspondent A answer.

There are two types of conversations, the first ones are given when the player interacts with the AI but this is not in a madness state (is not executing a madness action). These types of conversations has multiple responses - actually they are questions or affirmations from the player but they are called responses due to the fact that the AI receives an input, a response - whose number depends on the narrative items the player has discovered. The AI will response in relation to that topic answer.

The second type of conversation are given when the AI is executing a madness action and the player interacts with it, here the player has between three or four options that are always the same, although given the fact that the actions are selected taking into account AI's memory and the momentum, rarely the action will be repeated.

Furthermore, the AI has dialogues to say aloud, without the interaction of the player, for example when it sees a narrative object or when it is looking for the player.

5.6 Saving System

In this project it is vital that the dialogues are saved each time the player quits the game or finish it, thus next time he plays the dialogues will not be repeated. If they are not saved, when the player replays the game, he will encounter repeated dialogues and that breaks one of the project goals (See **section 3.2** and **section 4.2**).

Simply, it is implemented by checking if there is a game saved already, if it is, the dialogues are set from the save slot, if not, the dialogues are created from the *Data Table* (see **figure 5.4**). The game has only one save slot that is overwritten every time the player leaves the game level. Furthermore, the values from the options menu are saved and set the same way as the dialogues, and more stuff could be added to the save slot if it is necessary, for instance, save the player's positions and inventory therefore the player can leave the game and then continue later without having to start from the beginning.

5.7 Sound

Sound is an essential part of the player's experience and interaction. First, it immerses the player in the game environment and makes the game world more faithful to reality, and also, the sound interferes in the gameplay because the player knows where the NPC is based on the loudness of its steps and also it can detect the player steps if he is walking too fast (see **figure 5.9**).

5.7.1 Area Loops

The area loops are placed in the game world to generate looping sounds for a particular element. **Figure 5.11** shows the area loop of the swimming pool. The orange sphere indicates its attenuation radius, the closer the player is to the inner sphere, the louder the sound will be, and the further the player is from the inner sphere the lower the sound will be. The player will only hear this sound if he is inside of the sphere, although area loops can be placed without attenuation radius so they can be heard from all of the scenery.

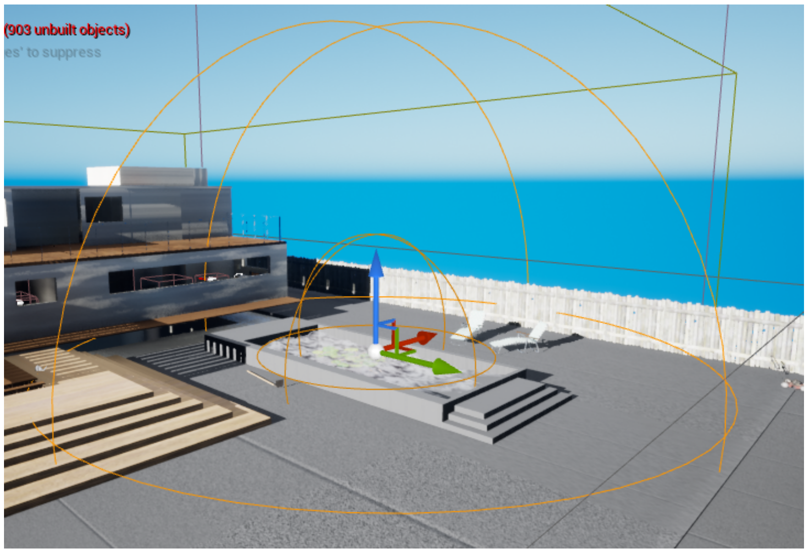


Figure 5.11: Area Loop from swimming pool

5.7.2 Element sounds

These sounds are played when the player interacts with the element. They are easy to implement and do not depend on any other element than the actor that triggers them. For example, doors will play a sound in its position when they open, that sound will depend on if it is locked or unlock. Also, when keys and figures are collected a sound will be played to notify the player.

5.7.3 Dynamic Sounds - Footsteps

The footsteps both from the player and the NPC are played dynamically, due to the fact that they depend on the velocity and the surface they are stepping on.

First, in order the make a real footstep sound, a sound cue is created with multiple footsteps sounds where one of them is selected randomly each time the cue is played (See **figure 5.12**). In this way the player will not hear the same sound for its footsteps.

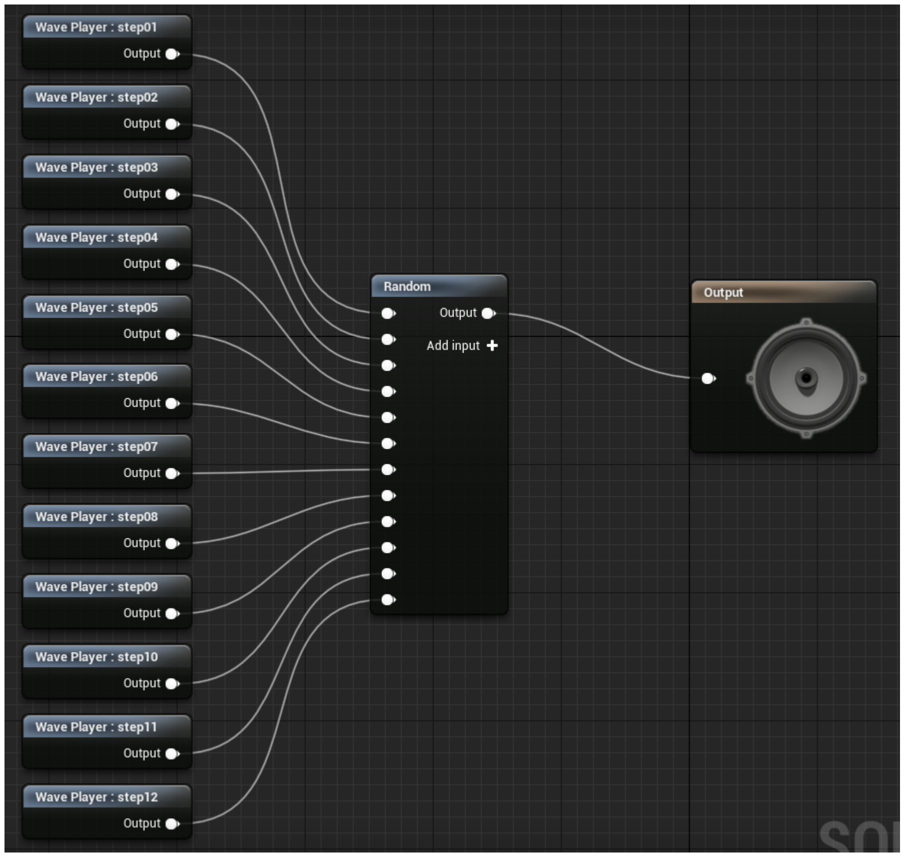


Figure 5.12: Sound Cue - StepsDefault

Secondly, both walk or run animation are related to the footstep sound, it will only be played when the foot collides with the ground. For this reason,it is needed that the animation sends a *Notify Event* in that exact moment to then, make the corresponding operations (see **figure 5.13**).

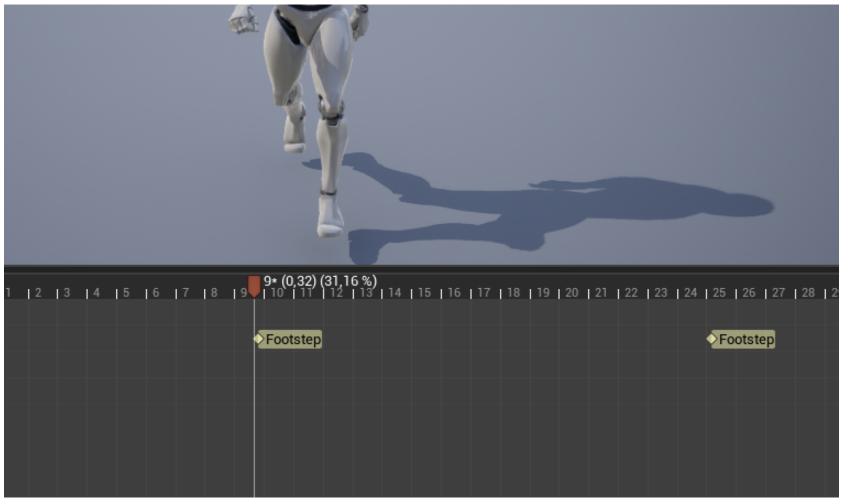


Figure 5.13: Notifying footsteps - Walk Animation

Then, in the animation blueprint, the *AnimNotify-Footstep* node will be called each time an animation reports the notify, and will throw a ray cast from the actor to the floor (see **figure 5.14**). This means as faster the actor goes there will be more footsteps sounds because the running animation will have more notify events.

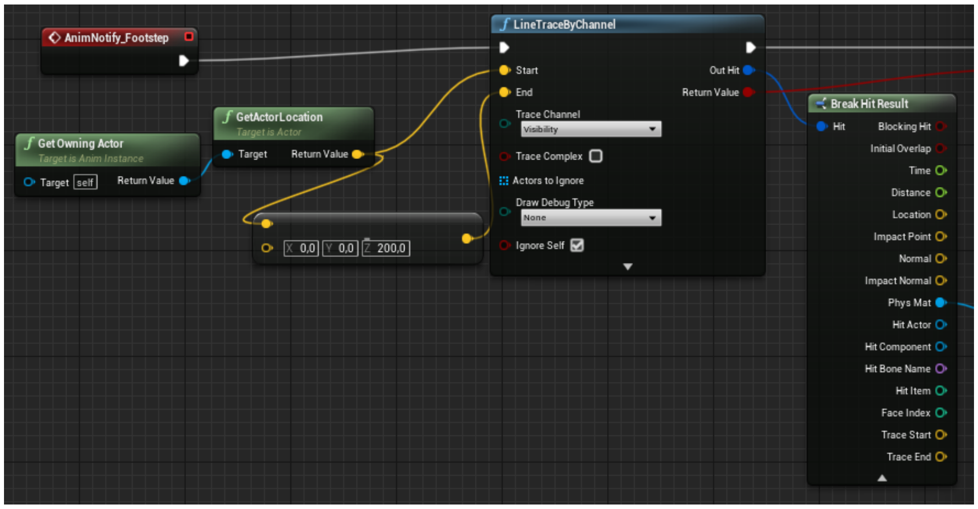


Figure 5.14: AnimNotify-Footstep 1 - Animation Blueprint

Finally, if the ray collides, it checks which surface the actor is stepping on and will play the corresponding sound cue depending on the type of surface - This is known thanks to the *Physical Materials*, a material used to differentiate the materials between them (wood, ground, water...) - with a volume that depends on the actor's velocity (see **figure 5.15**). In the project there are only two different footsteps, one for default and one for the ground.

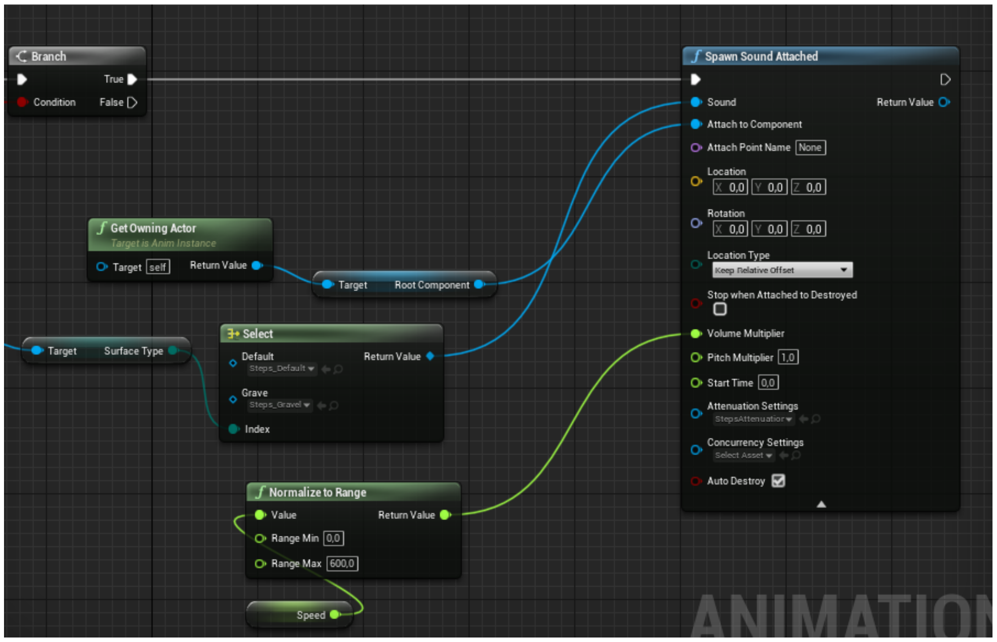


Figure 5.15: AnimNotify-Footstep 2 - Animation Blueprint

5.8 Debugging and Testing

Sometimes someone could have a brilliant idea that works in his head, although the reality is that not everyone feels the same way as he may think, perception is subjective therefore the rest may not think that idea is that great.

This section covers the process of planning for debugging and testing the video game and how with the results taken from testers and hours of debugging, it is managed to add and modify key aspects of the project to take it to a better version of itself.

5.8.1 Strategies

Playing is the most efficient way to test if your game has bugs and works the way it is intended to. However playing, it is not only for finding bugs, but also for checking if the game accomplish the designed goals (see **figure 3.1**). Although in order to make a valuable test, it should be done by people who did not participate in the project development, could be friends or family.

The first project testing was done by both members of the team, in order to find visuals and functionality bugs. Once those problems were solved, a prototype executable of the game was done therefore other people could play it and test it.

5.8.2 Optimization

A simple way to know if the game it is optimized is by checking if the fps drop - in this project it is intended that the game runs with more than 50 fps - , that means something is wrong. The best way to know what is causing this, is by checking the other stats shown in **figure 5.16**, they represent the delay from various game perspectives, thus in the moment the fps drops there will be one of the stats with more milliseconds than before.

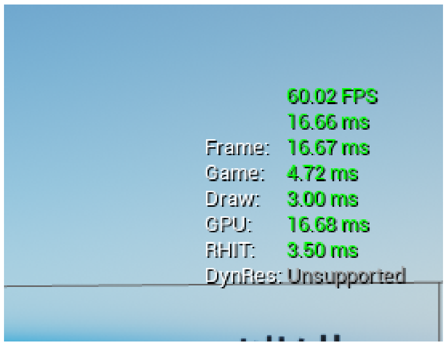

```
         60.02 FPS
         16.66 ms
Frame:   16.67 ms
Game:    4.72 ms
Draw:    3.00 ms
GPU:     16.68 ms
RHIT:    3.50 ms
DynRes:  Unsupported
```

Figure 5.16: Game played in the editor

5.8.3 Results

The testing and debugging helped to be able to make a playable executable of the game, nevertheless, it was a very simple version of the video game so the results given by the testers were helpful with functionality, visual and design problems, but not as they could be with a completed version of the game.

The optimization helped to realize that the game was performing pretty poor due to the transparencies, above all if behind the transparency there was another one, of the house (see **figure 5.17**)[1]. This problem was not solved but some improvements in other aspects (game code, objects or lighting) made the fps rise up.

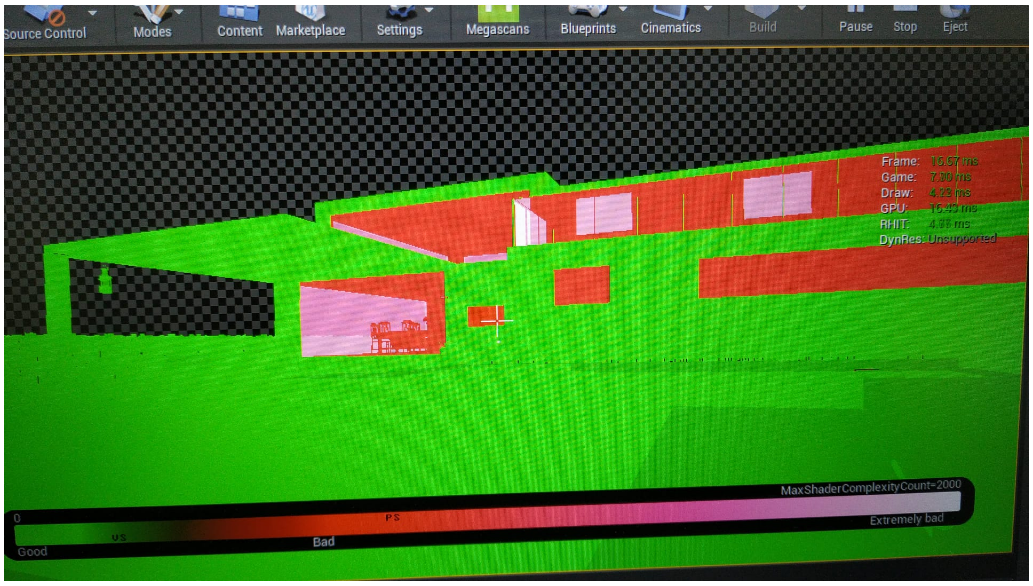


Figure 5.17: GPU render view

[1]White: extremely bad. Red: bad. Green: good.

Chapter 6

Project Monitoring and Management

First of all, organization, planning and communication between Sara and myself have been key to manage the project goals through out the development. The main tools used to communicate have been Discord, Google Drive and GitHub to share documents and the project, and also Trello to help to create a goal oriented planning.

Finally, project reports has been made and sent by email periodically to José Vicente Martí Avilés therefore he could sent back the corresponding feedback.

Chapter 7

Project Results

There have been modifications from the initial planning, essentially due to the variation of implementation or problems that have been come up. Despite all of this, the time initial planning has not suffered drastic changes.

This project, has an approximation of 15-20 minutes of gameplay and between three to four replays to complete all the game, which is exactly what it was intended. The realistic environment and the designed mechanics help the player to immerse within the game world. Also, the NPC produces interesting narrative and interactions with the player although the animations and dialogues are not that realistic as desired.

To sum up, this project is an original completed and functional video game as the initial planning and design indicates, although it could be improved in some aspects with more time.

- **GitHub repository**
- **Executable Link**

Chapter 8

Future Work and Conclusions

First of all, the gamepad controls should be added because it benefits the some players who are more comfortable with the gamepad instead of the keyboard and mouse. Even gamepad vibration could be implemented to highlight stress situations. Nevertheless, one of the tasks we would have liked to do was dubbing the dialogues however we had no time left for it. Also, the dialogues are very poor so they should be improved and then localized to English. With a few more months better animations and sounds could be added to refine the player's immersion and additionally, better adaptability and menu options.

This project has served us to learn to develop a video game in Unreal Engine 4 and participate in a wide range of fields in the video game creation. Furthermore, this is the most hardworking and eye catching project that we have developed thus make it the most useful tool in order to enter in the job market.

In addition, there have been multiple errors through out the design and development that we will take into account in the future and also, getting over all those inconveniences makes us so proudly of ourselves.

Finally I believe that with this project, we have gone beyond the knowledge and training in the degree and even trying to explore and to research about fields we have never been taught therefore we feel satisfied and glad with the final result.

Chapter 9

Appendices

9.1 Check Object

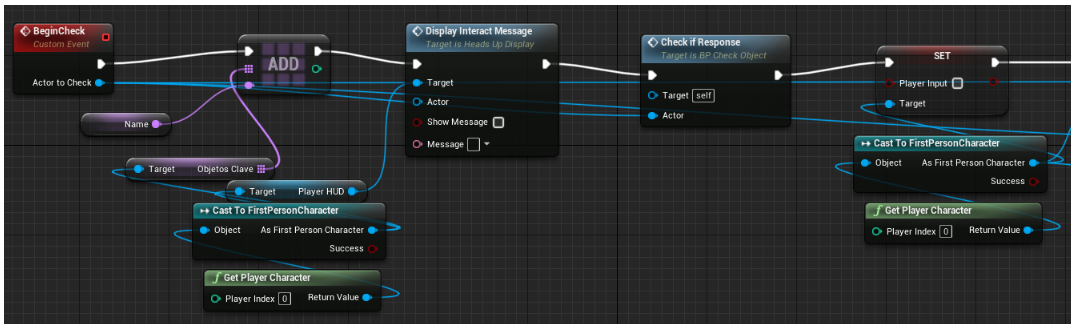


Figure 9.1: CheckObject 1

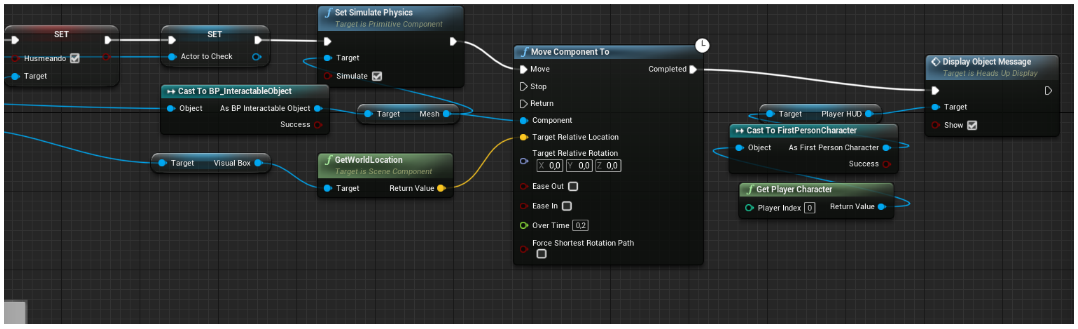


Figure 9.2: CheckObject 2

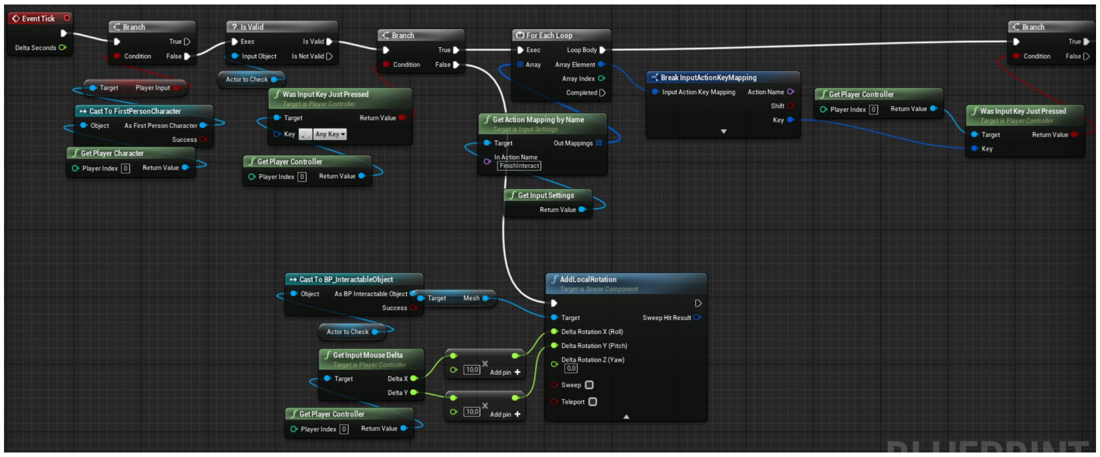


Figure 9.3: CheckObject 3

9.2 Pick up and drop objects

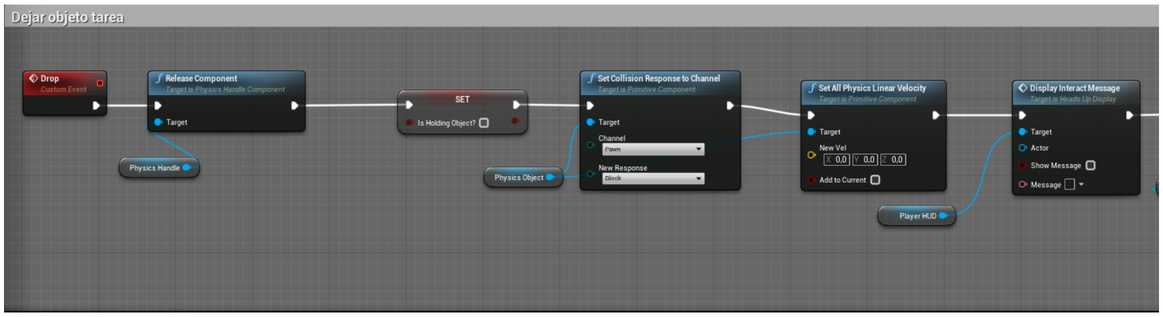


Figure 9.4: Drop Object 1

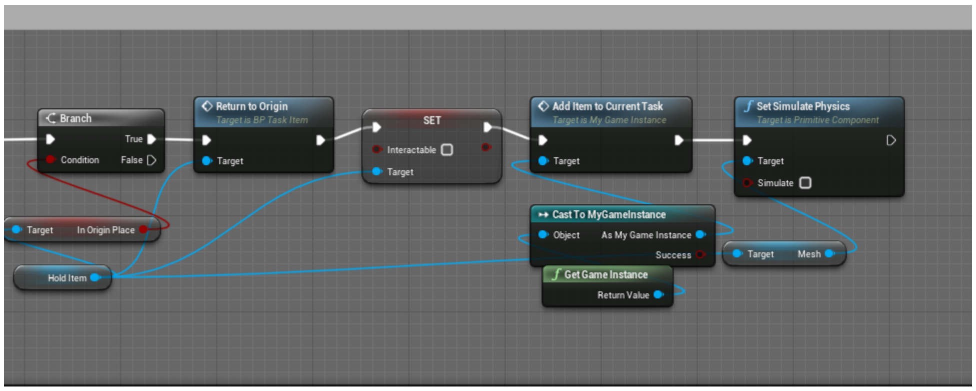


Figure 9.5: Drop Object 2

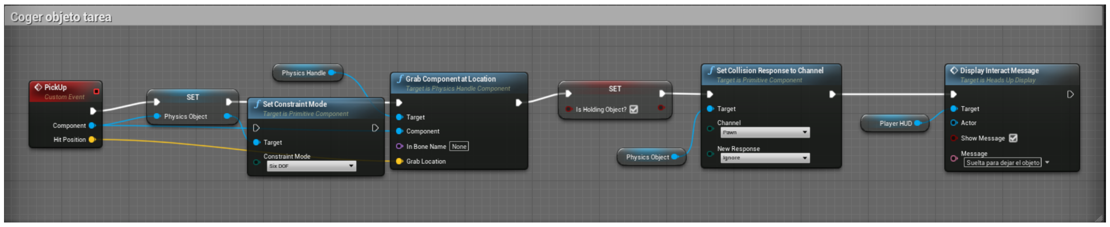


Figure 9.6: Pick up object 1

9.3 Say Dialogue

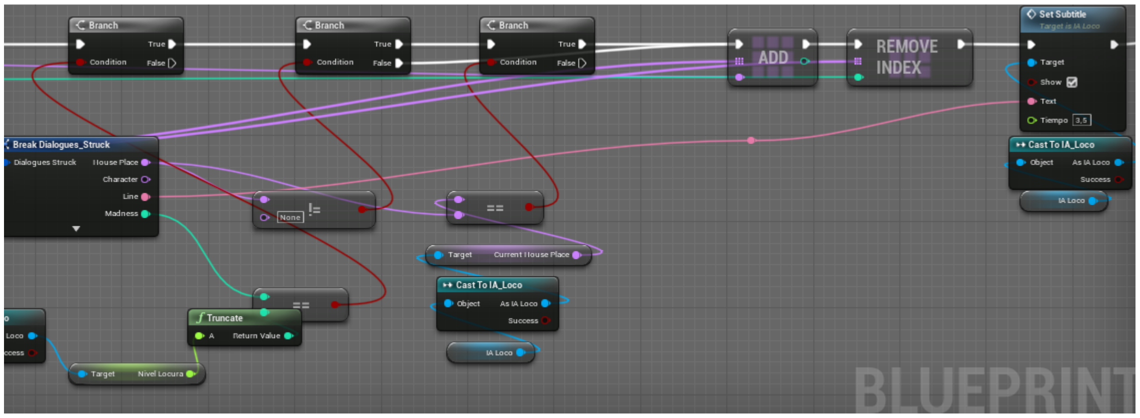


Figure 9.7: Say Dialogue 1

Bibliography

[1] Mark Brown. *Game Maker's Toolkit [Youtube Channel]*. 2006. URL: https://www.youtube.com/c/MarkBrownGMT/featured (visited on 07/06/2021).

[2] Mark Brown. *How the Nemesis System Creates Stories*. 2021. URL: https://www.youtube.com/watch?v=Lm_AzK27mZY&t=710s (visited on 10/06/2021).

[3] Epic Games. *Unreal Engine*. 2020. URL: https://www.unrealengine.com/en-US/ (visited on 10/06/2021).

[4] Johan Huizinga. *Homo Ludens*. Random House, 1938.

[5] Aleissia Laidacker. *GCAP 2016: Systems Are Everywhere*. 2016. URL: https://www.youtube.com/watch?v=Gelpn4mksXQ (visited on 10/06/2021).

[6] Josiah Lebowitz. *Interactive Storytelling for Video Games: A Player-Centered Approach to Creating Memorable Characters and Stories*. Financial Times Prentice Hall, 2017.

[7] Justin Mohlman. *Build a Detective's Office Game Environment [Course]*. 2020. URL: https://learn.unrealengine.com/course/3447824?r=False&ts=637586408002044192 (visited on 07/06/2021).

[8] Carlos Coronado Muñoz. *Desarrollo de juegos con Unreal Engine 4 de 0 a profesional [Course]*. 2020. URL: https://www.udemy.com/course/desarrollo-de-juegos-con-unreal-engine-4-de-0-a-profesional/ (visited on 07/06/2021).

[9] Jesse Schell. *The Art of Game Design: A Book of Lenses, Third Edition*. A K Peters/CRC Press, 2019.

[10] Michael Sellers. *Advanced Game Design: A Systems Approach*. 2017.

[11] Tanya X. Short. *Procedural Storytelling in Game Design*. A K Peters/CRC Press, 2019.

[12] Evan Skolnick. *Video Game Storytelling: What Every Developer Needs to Know about Narrative Techniques*. Potter Craft, 2014.

[13] Terriblis Studios. *Mixamo Converter*. 2021. URL: `https://terribilisstudio.fr/` (visited on 07/06/2021).