Contagia Y Pica

Ángel García Sorribes

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

July 3, 2023

Supervised by: Raúl Montoliu Colás, PhD.

Acknowledgments

I would like to thank my mother, my father and my sister for helping and supporting me at the hardest and more stressful moments. I would also like to thank my classmates for giving me moments of caml. I would also like to thank Raúl Montoliu Colás for his patience and help during the writing of this report and Sergio Barrachina Mir and José Vte. Martí Avilés for their inspiring LaTeX template for writing the Final Degree Work report, which I have used as a starting point in writing this report.

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

ABSTRACT

This TFG is made to show, Contagia y Pica videogame. This videogame is a single player, first-person puzzle solver made in Unreal Engine 5. It will be focused in a closed space where the player will interact with the different NPCs. The purpose of this project is to test, improve and show the skills acquired during the Video Game Design and Development degree at UJI applied on a videogame and learn Unreal Engine system of making games. Specifically, it has been made to apply knowledge about lighting and colors managment on a enviroment, programming behaviours in both blueprints and C++ code using artificial intelligence techniques to achieve respectively simple routines and behaviors that are capable of changing according to the actions of the player or the environment.

Contents

Contents **vii**

List of Figures **ix**

1 Introduction **1**
1.1 Game overview and Results preview . . . 1
1.2 Key words . . . 2
1.3 Work Motivation . . . 2
1.4 Goals . . . 4
1.5 Environment and Initial State . . . 5

2 Planning and resources evaluation **7**
2.1 Planning . . . 7
2.2 Resource Evaluation . . . 12

3 System Analysis and Design **13**
3.1 Introduction . . . 14
3.2 Rules . . . 15
3.3 Clothing . . . 20

4 Work Development and Results **21**
4.1 Work Development . . . 21
4.2 Fulfilled goals . . . 36

5 Conclusions and Future Work **39**
5.1 Conclusions . . . 39
5.2 Future work . . . 39

Bibliography **41**

A Source code **43**

List of Figures

1.1 Preview of the mine 2

2.1 A view of all the task approximately scaled to the time frame of the task. . . 11

3.1 A view of the tools of the game 13
3.2 A view of the mine limits 14

4.1 The NPC model created in blender with no clothes or materials 22
4.2 A screenshot of the rig 23
4.3 A screenshot of Unreal blueprints 24
4.4 Variables from the Blackboard. 26
4.5 Distribution and rules of execution of virus tasks. 27
4.6 View of the foreman routine. 28
4.7 View of the miner routine. 29
4.8 View of the builder and builder junior tasks. 29
4.9 A image that shows the different variables and functions stored in the game state. 31
4.10 A screenshot of the NPC clothing. The colors change depending on the NPC wearing them 32
4.11 A screenshot taken in the Unreal Editor used to create the different animation clips 33
4.12 Preview of the mine 34
4.13 A view of all the tools 35
4.14 Tool set in a Unreal default manequinn 36

Chapter 1

Introduction

Contents

1.1 Game overview and Results preview **1**
1.2 Key words **2**
1.3 Work Motivation **2**
1.4 Goals **4**
1.5 Environment and Initial State **5**

As mentioned above, this Final Degree Project (TFG) aims to showcase the skills acquired throughout my Video Game Design and Development career at UJI within a first-person game created using Unreal Engine 5. The project serves as an application of my knowledge in various areas, including project management, game engine optimization for performance, lighting, and color schemes. Additionally, it incorporates programming skills in both C++ and blueprints, demonstrating proficiency in creating interactive gameplay elements.

A piece of gameplay can be found here: `https://youtu.be/R_vUR0KS7-M`

1.1 Game overview and Results preview

The TFG encompasses artificial intelligence techniques implemented to create both straightforward routines and behaviors that challenge the player. By utilizing Unreal Engine 5's powerful features, the TFG offers an immersive and visually appealing gameplay experience. The combination of technical expertise and creativity allows for the effective implementation of various game development principles.

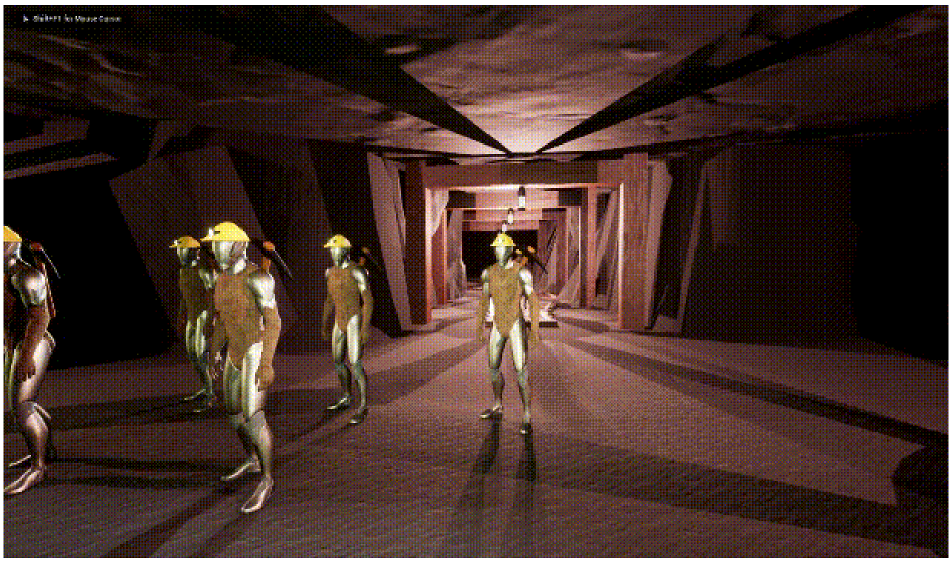


Figure 1.1: Preview of the mine

In the game the player will move through a mine alongside different NPCs who have different jobs, routines, and personalities. Each task has different animations and depending on the personality of a NPC, it will change the animation. On the other hand, there will be a virus that will infect NPCs by contact and will try not to be discovered at the same time.

1.2 Key words

This TFG is made to demonstrate knowledge in:

- AI techniques.
- Blueprints and C++.
- Scene creation and lighting.
- Animations.
- Unreal Engine 5.

1.3 Work Motivation

The main motivation of this work is to be able to develop a small and entertaining game. During the years of my degree I have been making games but always under a series of conditions that, due to time, knowledge or subject restrictions, I have not been able to develop. I do not doubt that they have been necessary to continue learning but now I see in this work an opportunity not only to create a small but complete game in which I am satisfied, but also to put all the knowledge I have obtained to the test.

Games like The biding of Issac[11], Return of the Obra Dinn[12] or books such as I robot[1] inspire me to create games that explore unusual mechanics and interactive stories.

Table 1.1: Three images. From up to down, left to right are the biding of Issac, I robot, and Return of the obra Dinn.

This work also intends to make use of NPCs and enemies controlled by algorithms which make use of AI techniques that respond or act dynamically not only to the player but also to their environment. Also the utilization and learning of AI techniques will be usefull in future developments.

1.4 Goals

The project has the following goals:

Design and develop in Unreal Engine: Unreal Engine is a game engine developed by Epic Games, which has gained widespread popularity in recent years due to its powerful and versatile features. It provides game developers with a comprehensive set of tools and technologies to create high-quality games that can run on multiple platforms, including PC, consoles, and mobile devices.

With the rise of the gaming industry and the increasing demand for high-quality games, it is no surprise that learning Unreal Engine has become a highly valuable skill for aspiring game developers and enthusiasts. By mastering Unreal Engine, developers can create engaging and immersive games that can captivate players and stand out in the competitive gaming market.

Furthermore, Unreal Engine has a large and active community, which provides extensive support, tutorials, and resources for beginners and experienced developers alike. As such, learning Unreal Engine can not only lead to a rewarding career in the game development industry, but it can also provide a platform for creativity and innovation in various fields.

AI bots: As It was said before the motivation of this project is to approach to the field of AI and it was thought that the best way was through developing a series of computer-controlled bots that implemented AI techniques. Even if it is on the project scale, it is intend that these bots could work with more than if-elses statements and to make one action or other taking into account the environment and the player state.

Reinforce programming capabilities: During my studies, I often encountered programming subjects that were focused on specific aspects of game development. Topics such as data management, object-oriented programming, and efficient algorithm design were covered individually, without much integration between them. This project aims to consolidate the knowledge gained throughout my academic career by creating a game that showcases the application of these concepts within the constraints of time and the requirements of the final degree project. Both blueprints and C++ will be utilized in the development process.

Attractive demo: Similar to programming, scene creation, decoration, and setting in game development have often been addressed as separate subjects. However, in this project, the goal is to combine these elements to create a cohesive and immersive mining environment. Instead of relying on temporary placeholders, the intention is to incorporate models with realistic materials and colors that align with the overall aesthetic of the other assets, even if they are relatively simple in design. By integrating these components effectively, the resulting scene will convey the desired atmosphere of an authentic mine, albeit on a smaller scale.

1.5 Environment and Initial State

It has received valuable guidance and assistance from my tutor in preparing the documents that I have submitted to the University. However, when it comes to the design and development aspects of the project, I have worked independently. It is worth mentioning that I have acquired knowledge about the workflow of Unreal Engine and its various features through self-learning. This was accomplished by referring to numerous YouTube videos and the official documentation available on the Unreal Engine website. All the YouTube channels and resources I have referenced for my learning journey are appropriately cited in the bibliography.

Chapter 2

Planning and resources evaluation

Contents

2.1 Planning **7**
2.2 Resource Evaluation **12**

2.1 Planning

This has been the original planning it was supposed to be made:

Task 1: Model NPC. The aim of this task is to create a fundamental human model complete with a fully articulated skeleton, which will accurately represent workers in a mining environment. To achieve this, both the mesh and its corresponding skeleton will be designed and constructed using Blender, a popular 3D modeling and animation software. It is crucial for the human model to have similar geometric proportions as the default mannequin model in Unreal Engine, as this will ensure compatibility and facilitate seamless integration into Unreal Engine projects. By aligning the geometry of the two models, the skeleton of the human model can be easily re-targeted to match the existing Unreal Engine mannequin skeleton. This compatibility will enable the utilization of pre-existing animations and tools designed for the Unreal Engine mannequin. To accomplish this task, a dedicated time frame of approximately 12 hours has been allocated. Within this period, the creation and refinement of the human model, including its mesh and skeleton, will take place.

Task 2: Rigging IK. Because the access to MOCAP from UJI facilities is not possible, create a Rig over the NPC model will save time on doing animations and will make easy to change them in case of needing it. This task should last 4 hours and must be done after the first task.

Task 3: Classes. This phase of the project involves defining the various classes of NPCs, including miners and constructors, along with their respective attributes and methods. These NPCs will be implemented using Unreal Engine's Blueprints, a visual scripting system that allows for the creation of interactive gameplay elements.

To ensure a comprehensive implementation of the various NPC classes, a dedicated time frame of approximately 11 hours has been allocated.

By employing Unreal Engine's Blueprints and beginning with the default mannequin, this phase of the project aims to expedite development while delivering tangible and visible outcomes. The combination of well-defined NPC classes and the utilization of existing assets will allow for a more efficient and productive development process, setting the stage for subsequent stages of the project.

Task 4: Events. This stage of the project focuses on defining the routines and behaviors of the NPCs. The aim is to establish the specific actions and interactions that the NPCs, such as miners and constructors, will undertake within the virtual environment. This task will be undertaken concurrently with the development of the NPC classes in Unreal Engine's Blueprints.

By allocating a specific time frame for this task, it is possible to focus on creating detailed and comprehensive routines that encompass various aspects, such as movement patterns, task execution, interaction with objects and other NPCs, and responses to different environmental conditions. This time frame also allows for iterative testing and fine-tuning of the routines to ensure optimal performance and desired outcomes.

The parallel execution of the NPC classes and routines tasks maximizes efficiency and enables quick iteration cycles. Any necessary adjustments to the NPC classes can be seamlessly incorporated into the routines, ensuring a coherent and cohesive system.

Task 5: State machine. This task will handle the game flow. The game will start on a tittle screen, will pass to the game scene and depending if the player wins or losses, a screen from each one will be show and the player will be able to go back to tittle or restart the game. As the tittle screen is not much important, could be done any time before the forward model or the observer.

Task 6: NPC clothes. This phase of the project focuses on creating clothes models and objects for the NPCs within the virtual environment. It is essential to ensure that the geometry of the cloth models is specifically designed to work seamlessly with Unreal Engine's physics simulation. Additionally, after creating the cloth model, it is crucial to assign the appropriate properties to the object and ensure proper attachment to the NPCs' mesh and skeleton, allowing for smooth integration with animations. While the initial stages may rely on the Unreal default mannequin, it is necessary to develop a custom model of the NPC before proceeding with this task, which should be completed within a timeframe of no more than 10 hours.

To guarantee the accurate functioning of the clothing models and objects, careful attention must be given to their geometry. This includes considering factors such as appropriate edge flow, topology, and physical constraints to ensure realistic movement and interactions with the environment. By adhering to Unreal Engine's physics simulation requirements, the cloth models will respond naturally to external forces, enhancing the overall immersion of the virtual world.

To maintain project efficiency, a dedicated time frame of no more than 10 hours has been allocated for this task.

Task 7: NPC clips. All NPC animations of each personality on each task. There are several task such as mining, walking and pulling the wagon. Each of them will have a variation for each personality. It is needed to have the NPC rig and the custom NPC model because is needed to confirm that the re-target works properly.

Task 8: Forward Model. This class will be dedicated to simulating the behavior of the virus within the game. Its primary responsibility will involve interacting with an observer class and taking one step in the game's progression. For instance, when the observer identifies a machinist in its list of nearby NPCs, the virus will cause the machinist to disappear in the next step, as machinists typically don't remain in one location for an extended period.

Given the complexity and potential impact on the overall project, this task is best suited to be undertaken towards the later stages of development. Allowing ample time for adjustments and fine-tuning ensures that the virus's behavior aligns seamlessly with the project's objectives. Implementing this class towards the end of the project offers flexibility, as any necessary changes or modifications can be made with relative ease when most other components of the game are already in place.

Task 9: Observer. This class will serve as the forward model for the virus, responsible for collecting information about the environment based on a given NPC. Its main role will be to analyze and gather relevant data regarding the NPC's surroundings.

As the forward model, this class plays a critical role in the virus's decision-making process and understanding of the game world. It enables the virus to assess the current state of the environment and make informed choices based on the information collected.

To ensure the accuracy and effectiveness of the forward model, it is recommended to prioritize the development and implementation of this class. By dedicating sufficient time and resources to this task, it allows for a comprehensive understanding of the NPC's environment and facilitates more intelligent decision-making by the virus.

Considering the significance of this class in shaping the virus's behavior, it is advisable to integrate it into the project once the foundational components and systems are in place. This approach ensures that the forward model is built upon a solid framework, utilizing the available data and interactions within the game to provide meaningful insights for the virus's decision-making process.

Task 10: Heuristics. This class will serve the purpose of simulating the virus, similar to the previous two classes mentioned. It will be responsible for implementing several methods that take two observers as inputs and return a value. These methods will be designed to facilitate the virus's decision-making process within the game.

Given the critical role of this class as a forward model, it is advisable to prioritize its development and completion towards the later stages of the project. By doing so, the groundwork for the rest of the project can be laid first, ensuring that the necessary components and systems are in place. This approach allows for a more comprehensive understanding of the game's mechanics, enabling the development of more accurate and effective methods within the virus class.

By postponing this task until the majority of the project is completed, it provides the opportunity to incorporate the knowledge and insights gained throughout the development process. This allows for better alignment of the virus's behavior with the overall gameplay, enhancing the quality and coherence of the final product.

Task 11: Walls of the mine. This task focuses on developing a more suitable mine environment. The objective is to create a small landscape object that accurately represents a mining site. This involves shaping the terrain, adding appropriate colors and textures, and implementing suitable lighting to enhance the overall visual quality and realism of the environment.

Task 12: Mine items: This task will consist on making different models to set in the mine to make it look better, such as rails or other objects that will help to the correct execution of the routines

Task 13: Tools. This task involves the creation of pickaxes for miners, a wagon for the machinist, and helmets for all workers.

Task 14: Itch.io page. In the context of the video game industry, effective communication of your project holds great importance. With this in mind, a dedicated task was undertaken to ensure clear and concise presentation of your TFG within this industry.

Task 15: Memory. This task involves thoroughly documenting both the process and the outcomes of the project. Documentation plays a crucial role in capturing the development journey and providing valuable insights into the project's objectives, methodologies, and achievements.

In addition to documenting the process, it is equally important to document the results and outcomes achieved. This involves capturing and presenting various aspects of the project, including the final human model with its skeleton, the defined NPC classes and their attributes, the implemented routines and behaviors, and the cloth models and objects. It is essential to provide visual representations, such as screenshots or renderings, to showcase the project's visual quality and demonstrate the successful integration of the various components.

To ensure clarity and accessibility, it is important to structure the documentation in a logical manner, with clear headings, subheadings, and a well-organized flow of information. Additionally, including references and citations to external sources or relevant academic literature strengthens the credibility of the documentation and demonstrates a thorough understanding of the field.

By dedicating sufficient time and effort to properly documenting the process and results of the project, future developers, researchers, or stakeholders will have a valuable resource that enables them to understand, replicate, and build upon the project's accomplishments. Effective documentation contributes to the project's overall success by facilitating knowledge transfer, encouraging collaboration, and fostering continued innovation in the field.

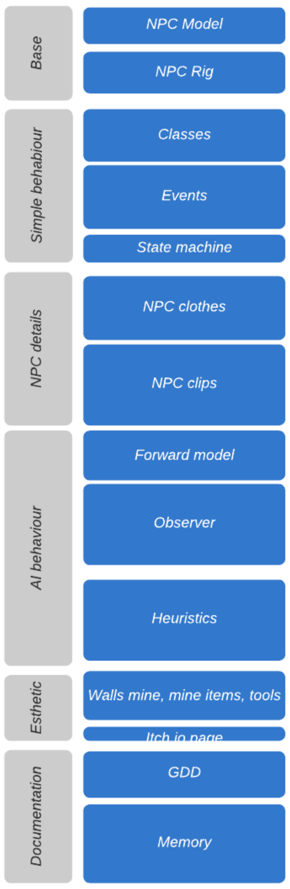


Figure 2.1: A view of all the task approximately scaled to the time frame of the task.

2.2 Resource Evaluation

The resources for the realization of this final project degree, can be divided into three types, hardware, software and human level. Only one person taking care of all the work.

Hardware cost: It has been necessary to have a powerful computer to handle Unreal, that give lots of problems such as crashed.

- Operating system: Windows 11 64-bit
- Processor: Intel I7-7700 @3.60GHz
- Memory: 16GB RAM (15GB usable)
- Graphics Card: NVIDIA 3060

Software cost: All programs which are have used are free as they are to non-commercial, they are:

- Unreal Engine : Unreal Engine 5 is a cutting-edge game development engine created by Epic Games. It offers a wide range of powerful tools and features to empower developers in building high-fidelity and immersive games across various platforms. Unreal Engine 5 represents a significant leap forward in game development technology, empowering developers to create visually stunning, highly immersive, and expansive gaming experiences across multiple platforms. Its cutting-edge features and tools make it an ideal choice for developers seeking to push the boundaries of what is possible in game design. See more information y official site on [5].
- Visual Studio : Visual Studio is an integrated development environment (IDE) developed by Microsoft. It provides a comprehensive set of tools and features to support software development across various platforms and languages. Visual Studio is a versatile IDE that caters to a wide range of developers, from beginners to seasoned professionals. Its feature-rich environment, extensive language support, and integration with other Microsoft technologies make it a powerful tool for building a wide variety of software applications.
- Blender: Blender's open-source nature, combined with its feature-rich tool set, makes it a popular choice among 3D artists, animators, and game developers. Its active community and extensive documentation further contribute to its versatility and accessibility. Blender is continuously evolving, with regular updates and new features being added by the dedicated community of contributors. To see more information and official site [3].

CHAPTER 3

System Analysis and Design

Contents

3.1 Introduction 14
3.2 Rules 15
3.3 Clothing 20

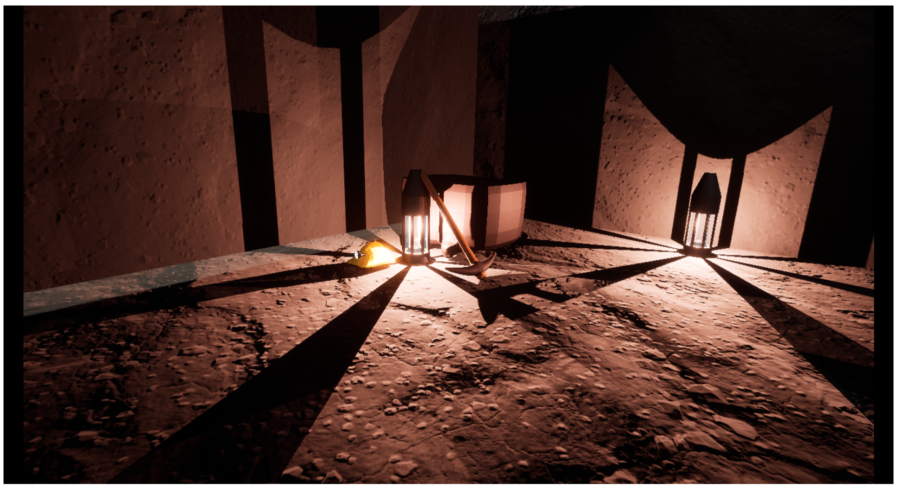


Figure 3.1: A view of the tools of the game

3.1 Introduction

3.1.1 Description

Contagia y Pica is a single-player, first-person game developed using Unreal Engine 5 for the Windows platform. The game revolves around the player's exploration of a mine, accompanied by various non-player characters (NPCs) who possess distinct job roles, routines, and personalities. These roles include mining, digging, and transporting filled carts along rails. Additionally, NPCs will engage in tasks such as building rails and providing structural support while excavating the mine or observing their fellow workers.

Each job is accompanied by unique animations, and the specific animation employed by an NPC depends on their individual personality traits. Meanwhile, a hidden virus will be present, capable of infecting NPCs through physical contact, while actively avoiding detection.

3.1.2 Objectives

The goal of the player is to ensure that no NPC is infected. To do this, the player will do a health test on all those NPCs that they think may be infected. If he is wrong, he will lose credibility and if he loses it three times in a row, he will have lost.

3.1.3 Limits

The player can only test the health of two NPCs in a row by failing, on the third the game ends. Both the player and the workers will be contained in a mine that will expand as the miners dig. The games last from a quarter of an hour to a half.

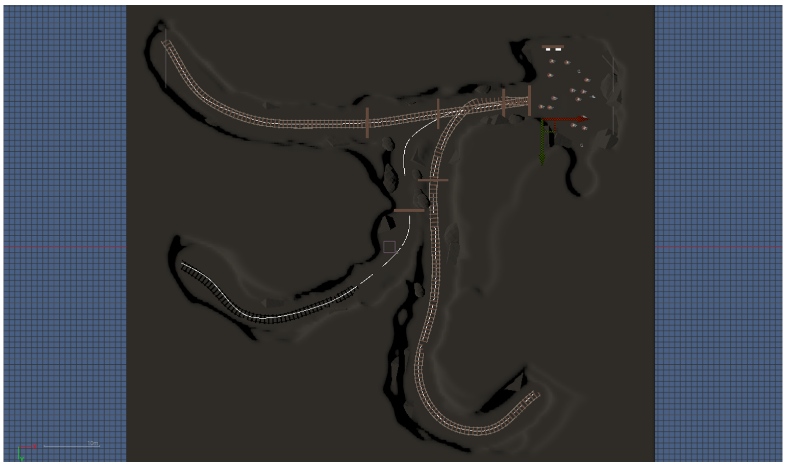


Figure 3.2: A view of the mine limits

3.2 Rules

3.2.1 Operational rules

NPCs

Each NPC in the game will possess a distinct personality and will be assigned a specific job. The assigned job will require them to navigate designated areas within the game map and interact with specific NPCs at predetermined times. Depending on the NPCs they come into contact with, they will adopt the personality traits of one group or another.

NPCs can only transmit the virus through physical contact. Once infected, the virus will gradually spread within the NPC's system, resulting in altered behavior. The virus will fully take control of the NPC once it reaches a certain level of growth, as explained in the foundational rules of the game.

Miner

Miners work in groups of 3 workers. A foreman assigns each group a position in which to mine. They mine until the Machinist arrives in their area. Then they pick up what they have chopped with the shovel and load it into the wagon.

Machinist

Each one goes back and forth moving the cart along the mine on the rails. At the opposite ends to the entrance, are the miners working. When the engineer reaches them the miners will stop mining and load the wagon until it is full with their shovels.

Constructor

The NPC's route in the game follows a path similar to that of a machinist within a mine. At the start of the mine, there are materials available for construction purposes. The NPCs will gather the necessary materials and proceed to the designated location where they need to build supports.

The NPCs will work in pairs, consisting of a senior and a junior member. The senior NPC will take the lead, determining the specific construction site, deciding what needs to be built, and choosing the appropriate path to reach the destination. The junior NPC will simply follow the lead of the senior NPC, assisting in the construction process without making independent decisions.

Foreman

The foreman is in charge of supervising the other workers. If he don't see nothing to build, he'll cyclically will go to the different groups of miners. If there's something to build, he will find a builder and will assign him the place and the type of building.

Virus

At the beginning of the game the virus will start on an NPC with a percentage of 1%. What the virus is capable of according to how much has infected it's NPC, is detailed in the founding rules. Every time the virus infects another NPC, the other NPC will have another instance of the virus that she will think up for itself. They can be coordinated, but it will be necessary for the two NPCs infected, to be able to communicate. Once in the body of an NPC, the virus will have a series of actions available and others that will be unlocked as it infects that NPC. The number of available actions will also vary depending on where you are located relative to other NPCs. The virus will have 4 ways to work, 4 ways to choose your next action, each one of them will be chosen at the beginning of the game and will be applied to all the instances of the virus that will be created:

Random The next action that the virus will take will be random among those available. It does not take into account the player or the consequences it may cause.

Rules In the event that the player is close or in sight, the virus will wait a few moments for it to leave. Whether it is not there or if it has waited, it will try to infect the near NPC that has the personality most similar to that of the group.

OSLA This method of looking for the next action will work on intervals. The intervals have a time frame where the action to be taken is processed and when that moment ends, the virus freezes for a few seconds. Having a series of actions available, they are assigned a heuristic value. Half of the actions with the lowest value are eliminated, if the actions have the same value, the first one in the list of actions remains. Compound actions are then created until the same number of actions as before is reached. A compound action is a list of actions that are executed one after the other, for example, go to a mine site and infect someone. The heuristic value of the compound actions are checked and the lower value of the half are eliminated. This process is repeated until a state in which the player loses is reached or the time of procesing has ended.If the virus has arrived to a lose player state, the list of actions will be saved and the virus will wait twice as long for the next cycle. The next time will calculate start from scratch.

Monte Carlo The different actions take an arrangement of nodes in a tree. According to the heuristic result by simulation of having chosen an action divided among the times that it has been chosen, the node will be taken as ideal. Next, other nodes will be created and this process will be followed recursively. The resulting tree will be saved for the next time by saving the data of each node.

Player

Moves through the mine with WASD, left stick. Move the camera with the mouse or the right stick. You can test an NPC by looking at it and clicking the left mouse button.

With the right he tells him to stop. The NPC will stay there for 30 seconds if you don't right click a location on the map. To do so, the NPC would go to that address. If you use the latter you will not be able to use it again in 3 minutes unless you discover an infected NPC earlier. You can immobilize an NPC with the middle button, but if they are not infected because they have not been tested by the detector, you will lose one of three opportunities until you find another infected.

3.2.2 Foundational Rules

Creation of NPCs and distribution of personalities

In a normal match, 13 NPCs will appear. The jobs are fixed and there are always 9 miners, 2 builders, 1 machinists and 1 foreman. Each NPC will have 1 traits that they will display in their animations. In each match, 4 personalities will be drawn from all. Each of them is assigned a number. To the first one, a number from 1 to 20 and the following ones, the previous number plus 20. Then foreach NPC, a random number from 1 to 100 will be created in that NPC and the personality with the closest number will be assigned to the number.

Traits and personalities

In each game, 4 of these traits will be chosen:

- Satisfied: Head and shoulders high, back straight.
- Dissatisfied: Declined, the opposite of the above.
- Energetic: Can't sit still, works fast.
- Tired: The opposite of the above. It is difficult for him to start moving and he works reluctantly moving slowly.
- Nervous: Shakes when making any movement.
- Shy: It withdraws when someone approaches it, it is always looking at the ground and tries not to look at others.
- Extrovert: He usually opens his arms, claps his hands and always looks at the face.
- Dancer: He moves to the rhythm of the song he has in his head, taking steps and movements that have nothing to do with what he is doing.
- Animated: When he walks, he walks with his arms and knees. With circular and irregular movements.
- Disciplined: Regular and rigid straight movements.

- Freaked Out: Exaggerate gestures. For example, when chopping, the knee opposite to the hand with which it has the beak goes up and accompanies the blow with the torso. He puffs out his chest and lifts his chin as if he weren't looking where he was going.

Virus Rules

The virus starts the game by infecting an NPC at 1 percent. Each NPC that is infected starts with the 5 percent. The different states are reached:

- From 12 %, give a positive health test. From here the virus begins to think and make calculations for the domain of the mine.
- From 25 %, the NPC already begins to copy the personality of those around him.
- From 30 %, you can communicate with nearby infected NPCs.
- From 45 %, you can move the NPC around the mine. If he is a foreman, he can also give him orders.
- From 60 %, the NPC can infect another. If he infects another, his percentage drops to 15
- After 90 %, you will be unable to heal the NPC and you will lose forever one of the chances you had.

Virus actions: The percentage increases by one point every eight seconds. To get to 100 percent, it takes 13 minutes from 0 percent.

Index	**Action**	**Precondition**
1	Indicate on the arrival of the player	Have reached 30% infection
2	Exchange information among infected NPCs	Both NPCs have passed the 30% infection
3	Move around the mine (Various actions for different points around the map)	Reached 45% infection (This action does not apply to constructors)
4	Break a support (various actions in depending on the supports built)	That an infected NPC has overcome 45% of the infection
5	Infect another NPC	Have reached 60% infection.

Virus actions, heuristics: If the way to calculate the next action needs a heuristic value, in the case of OSLA and Monte Carlo, it will use the following values:

Num	NPC Status	Heuristic Value
1	-	+100
2	Are you a miner?	+200
2	Are you another NPC?	-10
3	Is a junior, miner, machinist or builder	-1000
3	Is a senior foreman or builder	+50
4	Have you isolated the player?	+1000
4	Have you isolated infected NPCs from the player?	+100
4	You have isolated at least one NPC infected with another that is not?	+100
4	Have you isolated an NPC infected with the player?	-100
5	Have you infected a Miner or junior builder?	+50
5	You infected a Machinist, senior builder or foreman?	+1000

(Each row, corresponds to the row of the table of virus actions, which have the same number of the first column)

Conditions of defeat

The primary victory condition in the game is to ensure that none of the NPCs are infected, maintaining a zero infection percentage. Conversely, the player's defeat condition is losing all credibility points.

The player can lose credibility points if any of the following conditions are met:

- Do a health test which outcome be negative
- That a NPC reach the 90% of the infection.

Written Rules

Basic controls, you move with WASD on the keyboard and the left stick on the controller. You move the camera with the mouse and the right stick of the controller. To achieve victory, the player must detect all the infected people. When the virus infects a person, it will make his character like the rest around him. Also, when the virus has lived long enough in an NPC, you can move it around the map.

Player Action Table

Here is shown the player possible actions.

Action	**Description**	**Dynamic Actions**	**Called by**
Move	–	The different NPCs at the sight will know your position	Press the WASD keys or the left stick of the gamepad
Move the camera	–	–	Move the mouse or the right stick of the gamepad
Test health	Will test the NPC infection actual state	Subtract or leave as was the credibility	Left mouse click or button to command.

3.3 Clothing

All NPCs will be wearing a hardhat, miners and machinists yellow and builders and foreman orange. The miners will be dressed in a simple shirt and pants with brown and black suspenders respectively. The drivers are the same but with blue pants. Builders and foreman will wear a reflective vest and blue and black pants respectively.

CHAPTER 4

Work Development and Results

Contents

4.1 Work Development **21**
4.2 Fulfilled goals **36**

4.1 Work Development

For the correct description of the project's development it will be an explanation for each task defined on the planning:

4.1.1 NPC model:

Is a 3d model made in blender. For the realization of the mesh it was used the Unreal default NPC mesh so it could be compatible with the skeleton. This allowed to create NPC classes and events with the default model before making the NPC.

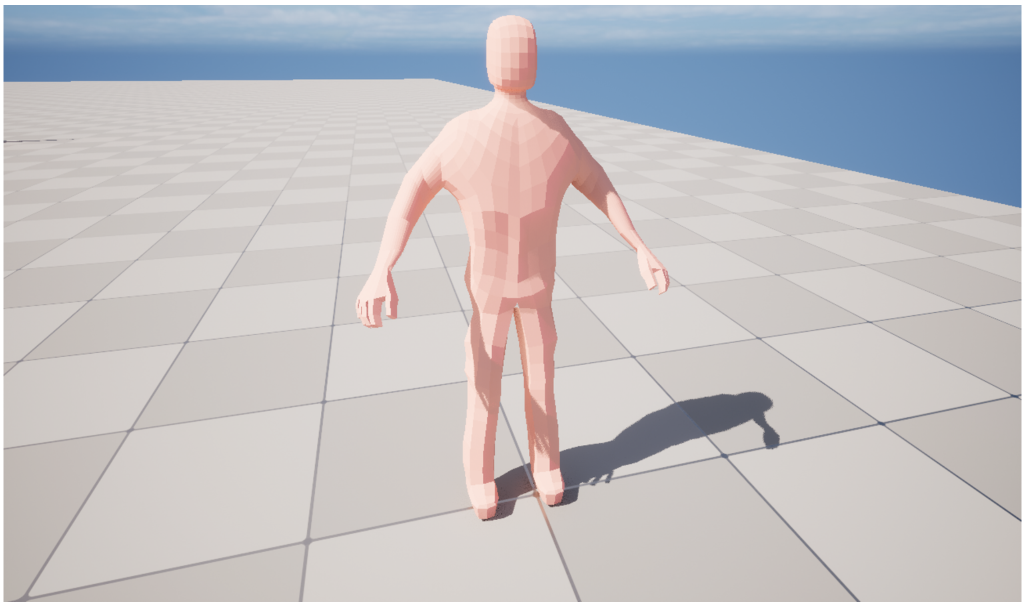


Figure 4.1: The NPC model created in blender with no clothes or materials

4.1.2 NPC rig:

A control rig in 3D computer graphics is a system of control objects or bones that are used to manipulate a character or object in a 3D scene. These control objects or bones are typically placed on a rigging hierarchy and can be used to deform and animate a polygonal mesh.

The control rig defines the movement and deformation of the mesh through the manipulation of these control objects. By adjusting the position and rotation of the control objects, animators can create complex movements and expressions that can be applied to the mesh.

The use of a control rig allows animators to create more realistic and natural movements and expressions, and provides greater flexibility and control over the final animation. Additionally, control rigs can be used to streamline the animation process, making it faster and more efficient to create complex animations.

The rig has been made in the Unreal default mannequin for later retargeting. Animation Retargeting is a feature that allows animations to be reused between characters that use the same Skeleton asset but may have vastly different proportions. Animations are bound to a Skeleton asset. The Skeleton asset is really just a list of bone names and hierarchy data, but it also stores the initial proportions from the original Skeletal Mesh used to define the Skeleton asset. This data is stored as bone translation data. It is important to note that the retargeting system only retargets the bone's translation component. The bone's rotation always comes from the animation data. Also with Animation Retargeting, there is no significant difference in performance between using retargeted and non-retargeted animations. The benefit of using animation retargeting

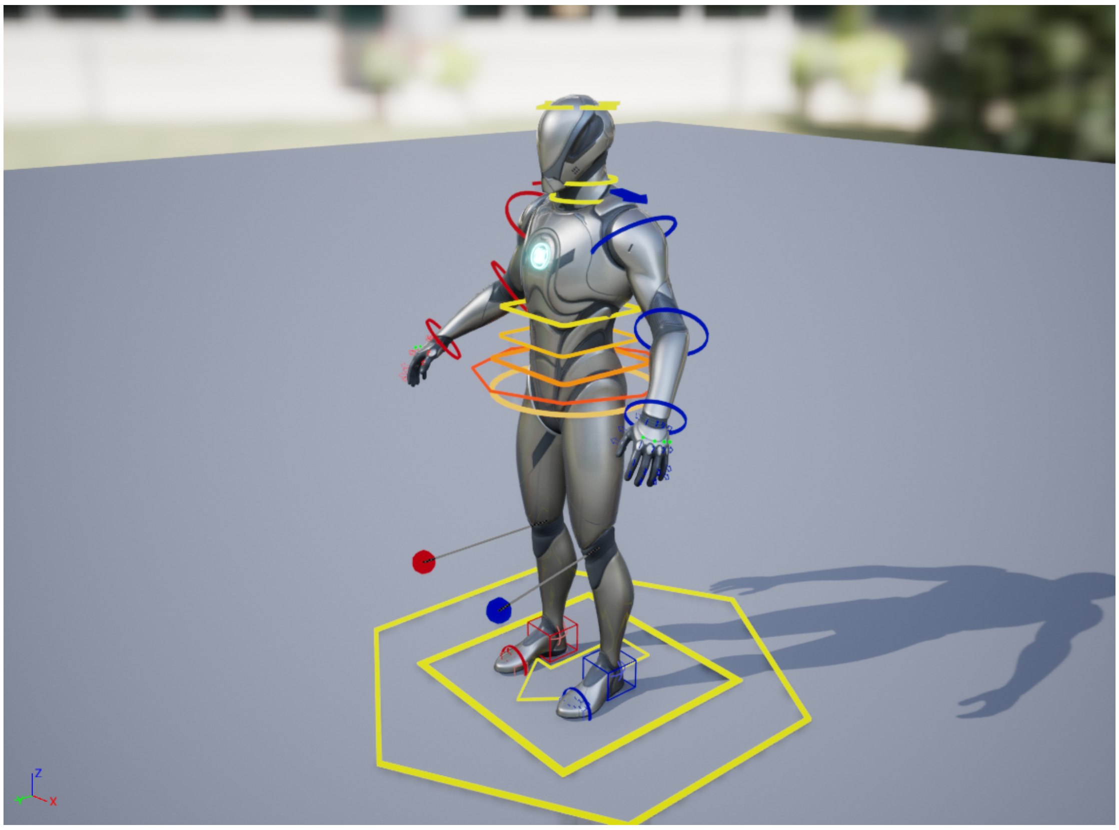


Figure 4.2: A screenshot of the rig

is increasing the number of unique characters without having to create an entirely new set of matching animations which could seriously cut down on your animation memory budget.

The rig it has been created applying the knowledge from Unreal rig[4]. Retargeting has been learn on the Retargeting Unreal documentation page[10].

4.1.3 Classes and Events:

Even if they are two separate task, they go together on the implementation. Classes were made with Unreal Blueprints, a way of programming Unreal custom classes with a visual interface, not in code. This allows developers to program faster even if you don't have a deep understanding of how Unreal works at the time of creating actors.

To see more about blueprints, documentation can be found here [8] in the official page of Epic Games, Unreal 5 documentation.

Classes were made through inheritance. Each of the jobs like Miner, Builder, Foreman, and Machinist inherit from the NPC classes. This is so that when an NPC is infected, it has the same methods as the other NPCs but at the same time it allows

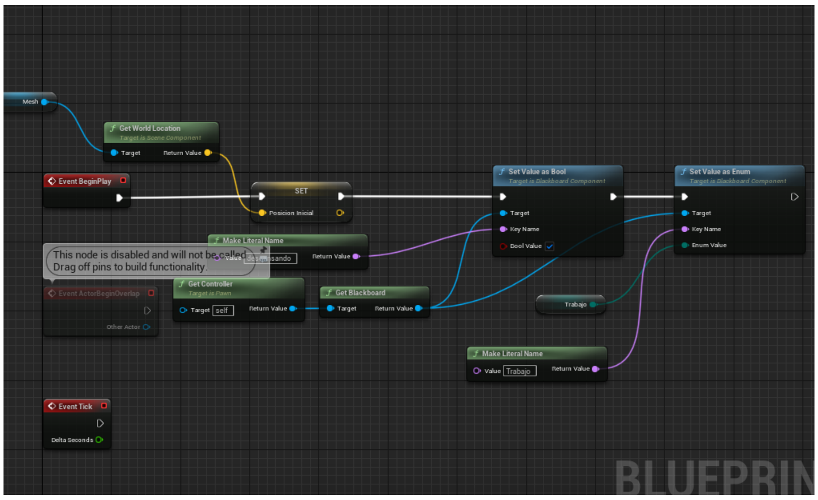


Figure 4.3: A screenshot of Unreal blueprints

NPCs with different routines.

At first the events were done by means of blueprints but due to the difficulty involved when following the thread of execution, methods and variables scattered by classes, it was decided to use unreal behavior trees that group and locate the thread of execution.

Behavior Trees assets in Unreal Engine 5 can be used to create AI for NPC. While the Behavior Tree asset is used to execute branches containing logic, to determine which branches should be executed, the Behavior Tree relies on another asset called a Blackboard which serves as the "brain" for a Behavior Tree. The Blackboard contains several user defined Keys that hold information used by the Behavior Tree to make decisions. For example, you could have a Boolean Key called Is Light On which the Behavior Tree can reference to see if the value has changed. If the value is true, it could execute a branch that causes a roach to flee. If it is false, if could execute a different branch where the roach maybe moves randomly around the environment. Behavior Trees can be as simplistic as the roach example given, or as complex as simulating another human player in a multiplayer game that finds cover, shoots at players, and looks for item pickups.

The Behavior Tree consists of three panels: the Behavior Tree graph, where you visually layout the branches and nodes that define your behaviors, the Details panel, where properties of your nodes can be defined, and the Blackboard, which shows your Blackboard Keys and their current values when the game is running and is useful for debugging.

The Behavior Tree graph is build by composites. This are a form of flow control and determine how the child branches that are connected to them execute.

Composites	**Description**
Selector	Executes branches from left-to-right and are typically used to select between subtrees. Selectors stop moving between subtrees when they find a subtree they successfully execute.
Sequence	Executes branches from left-to-right and are more commonly used to execute a series of children in order. Unlike Selectors, the Sequence continues to execute its children until it reaches a node that fails.

Task nodes are the actions that you want the Behavior Tree to perform. More documentation of behaviour tree can be found here [7].

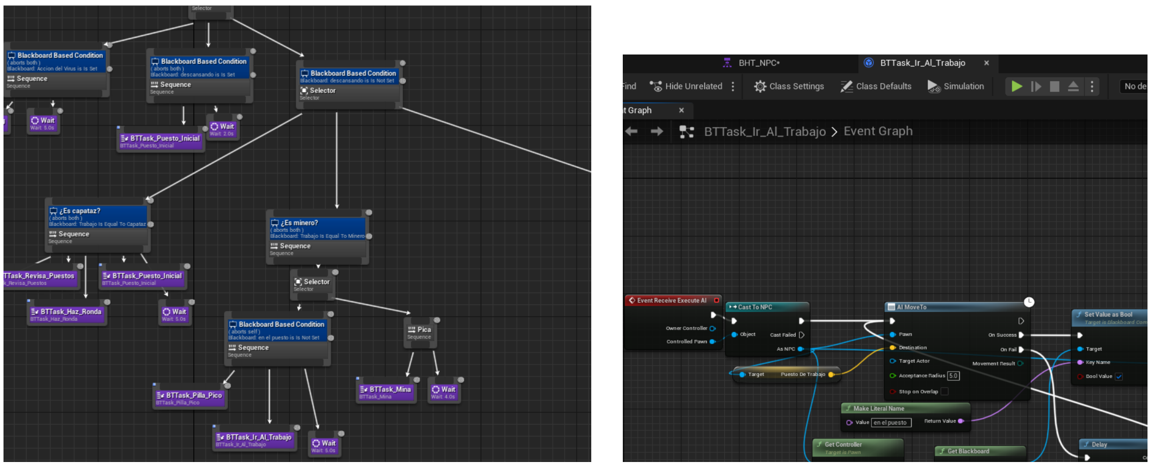


Table 4.1: A screenshot of the Unreal behaviour tree blueprint. The left image defines the flow of the different task meanwhile the other is an example of a task.

Blackboard variables:

It has been considered that at the time of making events, it is important to group for later utilization few variables. All NPC classes have access to this Blackboard variables which can be accessed and changed any time.

Self actor: It is included by default and cannot be removed. It is a reference to the Actor who is using this Behavior Tree blueprint.

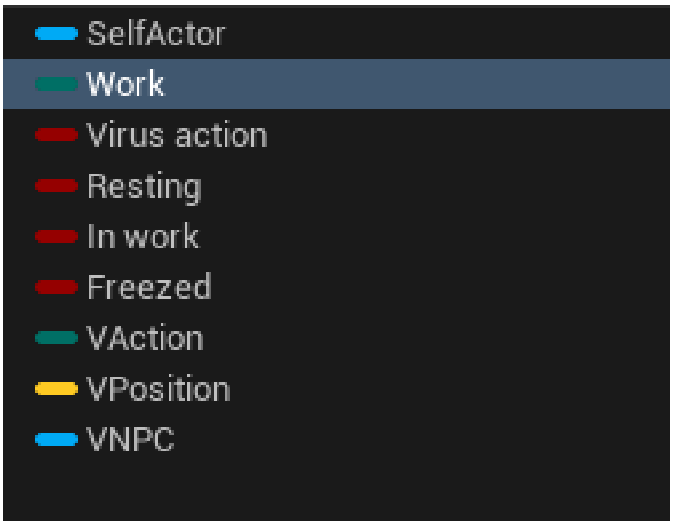


Figure 4.4: Variables from the Blackboard.

Work: It is a enumerable type that mark the NPC work. Depending on whether it is a miner, builder, builder junior, foreman or machinist a task or several of them will be executed.

Virus action: It is a Boolean that indicates whether the next task of the NPC is going to execute would be from its normal routine or a virus controlled action.

Resting: It is a Boolean that indicates if the NPC finished its routine. If so, it will go to the initial place where the NPC was located at the beginning of the match and wait for orders of the foreman.

In work: It is a Boolean that indicates if the NPC is located at its work place.

Freezed: It is used when a task require to wait in a place but it is needed to continue executing code. This variable is used for debugging behaviours and has no real use.

VAction, VPosition, VNPC: These variables are used by the virus execution thread to properly realice their actions. VAction is an enum type which indicates which task is gonna be executed. If is gonna execute an action of going to a location, this location will be indicated in VPosition. If it is going to Infect an NPC or communicate with one its reference and location are sotred in VNPC.

Events from the virus:

All virus tasks are focused on a part of the Behavior Tree graph. To understand how it works. At the top it can be see a gray rectangle that contains a blackboard based condition and a sequencer. When the thread of execution gets to this rectangle, it is checked if Virus action boolean is set to true. If is it, then the sequencer will be executed. A sequencer behavior it is explained above but to sum up, execute all it's tasks from up to down, left to right.

So it will alway execute the wait task, and depending on the action (determined by VAction) it is executen one task or another.

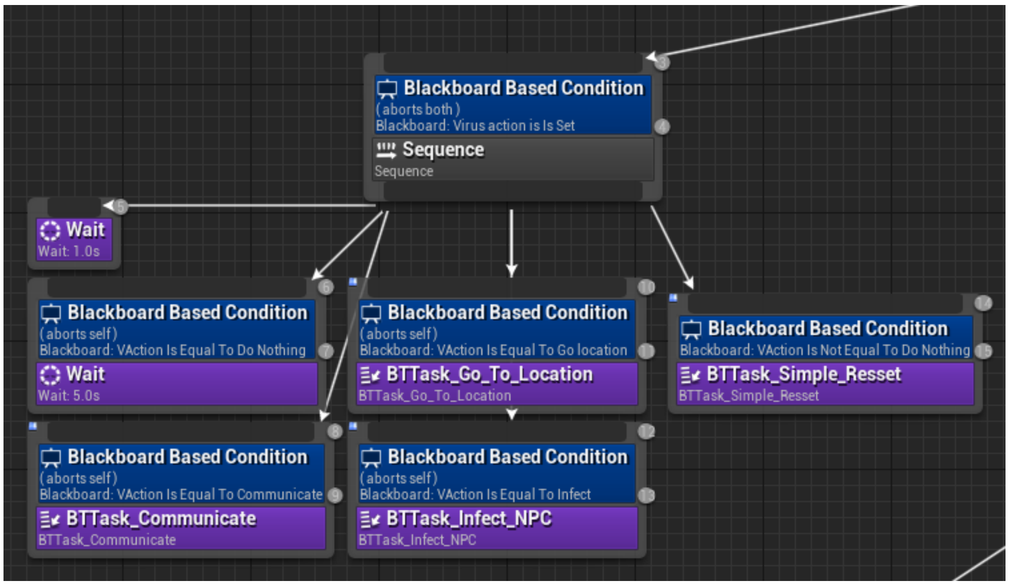

Blackboard Based Condition
(aborts both)
Blackboard: Virus action is Is Set
Sequence
Sequence
Wait
Wait: 1.0s
Blackboard Based Condition
(aborts self)
Blackboard: VAction Is Equal To Do Nothing
Wait
Wait: 5.0s
Blackboard Based Condition
(aborts self)
Blackboard: VAction Is Equal To Go location
BTTask_Go_To_Location
BTTask_Go_To_Location
Blackboard Based Condition
Blackboard: VAction Is Not Equal To Do Nothing
BTTask_Simple_Resset
BTTask_Simple_Resset
Blackboard Based Condition
(aborts self)
Blackboard: VAction Is Equal To Communicate
BTTask_Communicate
BTTask_Communicate
Blackboard Based Condition
(aborts self)
Blackboard: VAction Is Equal To Infect
BTTask_Infect_NPC
BTTask_Infect_NPC

Figure 4.5: Distribution and rules of execution of virus tasks.

Do nothing: It is simple, it just waits 5 seconds. It won't be executed Simple Resset task.

Communicate: The NPC will go to VNPC location and share the list of near NPC it acquired in the past, so the new NPC controlled by the virus, will be able to relate to more NPCs.

Go to location: The NPC will go to VPosition and take a look to near NPCs so it would be related to more NPCs.

Infect: The NPC will aproach to other and will infect it.

Simple Resset: After all three task that have been mentioned before, the NPC will come back to its job.

Foreman:

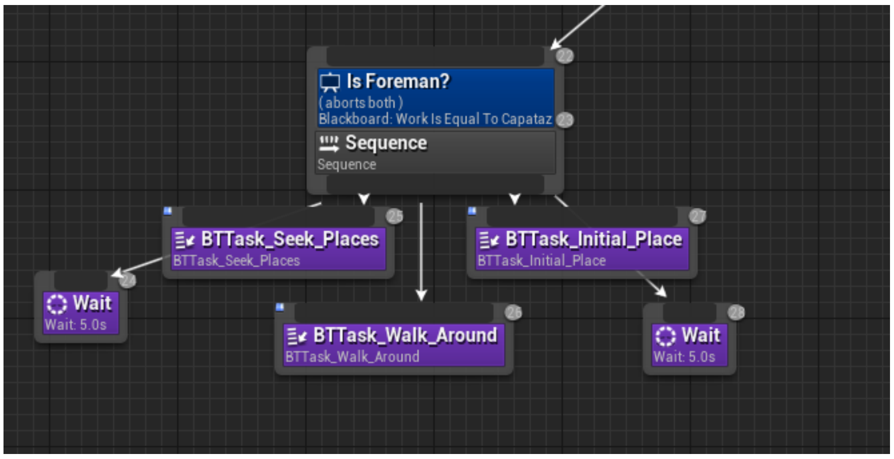


Figure 4.6: View of the foreman routine.

The foreman is the one who leads all the other NPCs. The only blackboard condition here is whether the variable work is set on foreman or other. If it is, all task will be executed with no more conditions. It's routine loop starts waiting five seconds. This is for giving the player time to take a short view to the mine. After this will check all work places and assign to NPC which won't be busy at the moment. It will go to all mine locations and after this will return to its initial place.

Miner:

The miners have a more sophisticated system but still easy to understand. First of all, they take their pickaxe from their back. This is intentionally set before miners get to their job places because otherwise does not work correctly for technical reasons. After this they will go to their work place set to true their boolean In work and start mining.

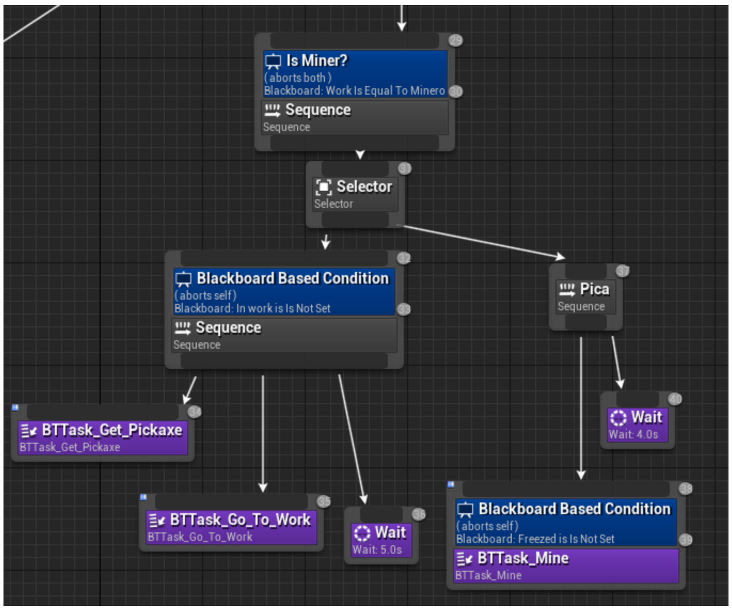


Figure 4.7: View of the miner routine.

Builder and builder junior:

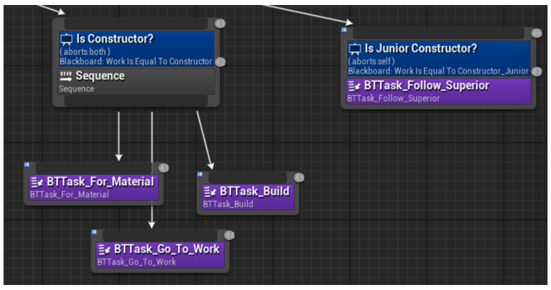


Figure 4.8: View of the builder and builder junior tasks.

The builder routine is simple. The builder goes to the place where the wood is located and grabs one. It goes to the job place, where the construction is meant to be and when it arrives, puts the wood so is construction is more near to the end. When the foreman reviews work places, only assign a place to builders which don't have an assigned construction already. The builder can also be resting at its initial place waitting to the foreman to assign it a construction.

The builder junior just follow its superior.

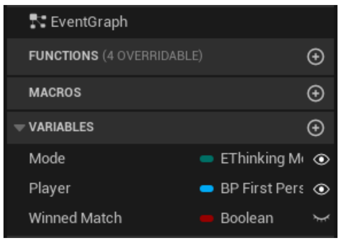
Machinist:

They are not controlled by behaviour tree. To make it, it is needed to acquire more Unreal knowledge. It is not by behaviour tree, because it is needed to access NPC events in a timeline which is not possible through tasks.

4.1.4 State machine:

This task involves creating and displaying a title screen, victory screen, and defeat screen as needed, and transitioning to the game scene when necessary. This has been accomplished using the Unreal Engine's level blueprint functionality and UI blueprints.

For storing global variables it has been used a blueprint class that it does inherits from game instance.



EventGraph
FUNCTIONS (4 OVERRIDABLE)
MACROS
VARIABLES

Mode	EThinking M
Player	BP First Pers
Winned Match	Boolean

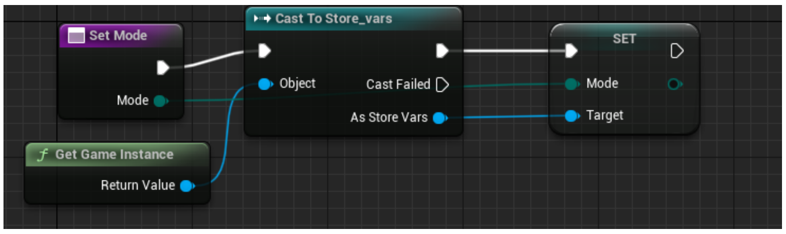
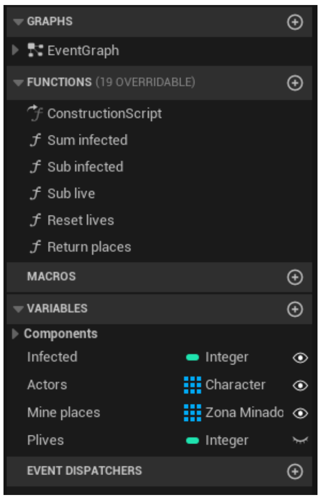


Table 4.2: A image that show the global variables and how they can be accessed.

In the same scene it is use to control the flow of the match, the variables and the functions of the game state.



GRAPHS
EventGraph
FUNCTIONS (19 OVERRIDABLE)
ConstructionScript
Sum infected
Sub infected
Sub live
Reset lives
Return places
MACROS
VARIABLES
Components

Infected	Integer
Actors	Character
Mine places	Zona Minado
Plives	Integer

EVENT DISPATCHERS

Figure 4.9: A image that shows the different variables and functions stored in the game state.

4.1.5 NPC clothes:

In Unreal Engine, a cloth object is a type of simulated mesh that can be used to create realistic clothing, flags, banners, curtains, and other flexible or deformable objects. The cloth object is simulated using physics-based algorithms that calculate how it would behave in response to external forces like gravity, wind, or collision with other objects.

The mesh form has been done in blender for later customization and application in Unreal.

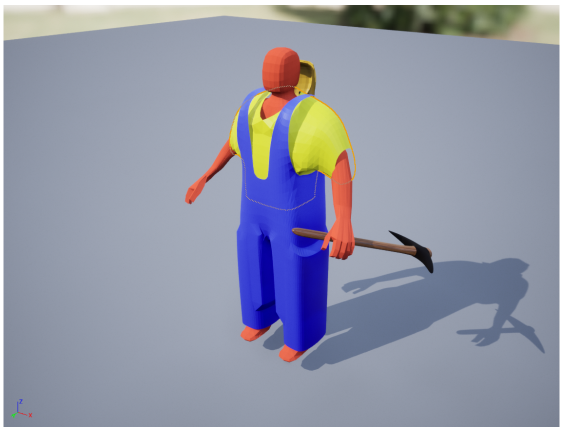


Figure 4.10: A screenshot of the NPC clothing. The colors change depending on the NPC wearing them

To create properly the cloth objects it has been seen Unreal's tutorials such as Unreal cloth[2] and Custom Clothes[6] just to mention a few.

4.1.6 NPC clips:

A big set of animations were made in the Unreal editor moving the NPC rig objects and saving their transforms. The animations are made to recreate different actions such as walking, pulling a wagon and mining. Each of this task have a variation for each of the personalities written in the third section of the memory.

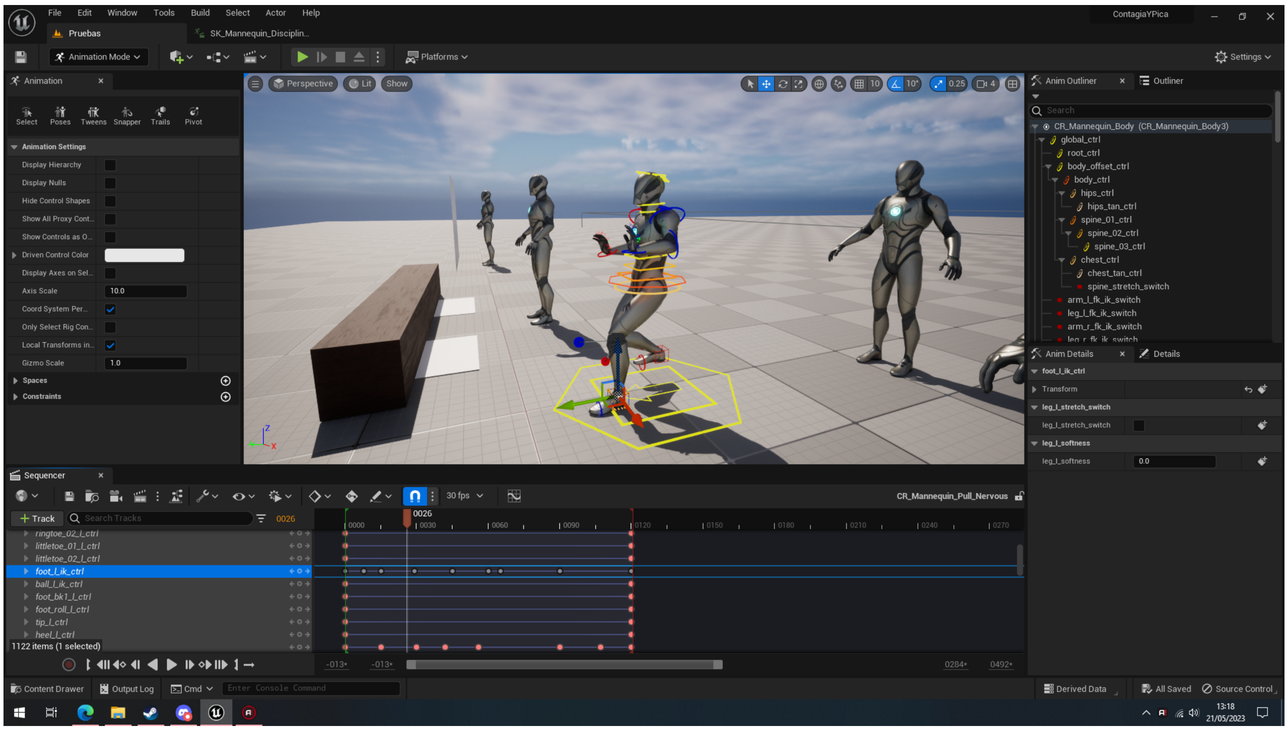


Figure 4.11: A screenshot taken in the Unreal Editor used to create the different animation clips

4.1.7 Forward Model, Observer and Heuristics:

Although these tasks are separate, they have been approached and developed in a unified manner using C++ classes. The virus class, implemented as a blueprint, interacts with these classes as attributes. The algorithm for each class was initially designed and documented on paper before the actual implementation took place in Visual Studio. Extensive planning was undertaken to ensure optimal results, emphasizing the importance of thoughtful design over rushed implementation.

Algorithms have been made inspired on [9] and [13] videos.

4.1.8 Walls mine:

This task involves creating the atmospheric environment within the mine. It includes designing and placing objects such as the floor, walls, and rocks to shape the scenario. Additionally, attention is given to lighting and materials to enhance the overall visual experience. The lighting within the mine is primarily provided by lanterns, which are defined in the tools task discussed below. However, fine-tuning and adjusting the lighting falls under the responsibility of this task.

The Unreal Editor is utilized to accomplish these objectives, employing object instancing techniques and utilizing a landscape object to shape the mine. Furthermore, a ceiling is created, and the intensity of the light sources is adjusted to achieve the desired visual effect. This includes adjusting the lighting within the helmets of NPCs and the player, as well as ensuring proper illumination for each construction within the mine.

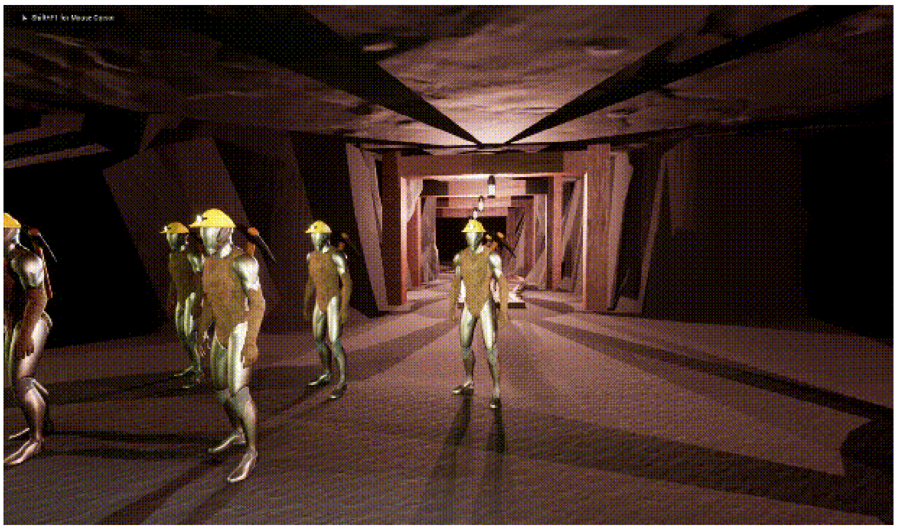


Figure 4.12: Preview of the mine

4.1.9 Mine items:

There are three models made for the mine, constructions and rails. The rails are created in blender and the constructions in the Unreal Editor.

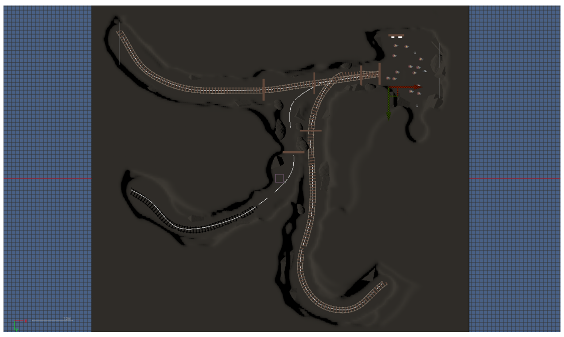
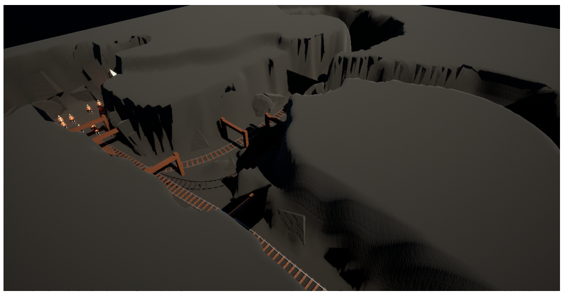



Table 4.3: The entire mine from the top and from a perspective view.

In none of these pictures ceiling is visible but this is only for showing the entire mine.

4.1.10 Tools:

The task consist on making three models a pickaxe, a lantern, a helmet and the helmet lantern. After making the model it has given a light object and adjust the lighting for both lanterns. They go all in specific places. The pickaxe, the helmet and it's lantern are attached in a socket on the skeleton mesh. These are transforms attach to the bones. When an object or a model is attach to a socket, it will move as the bone does. The helmet and it's lantern are attached to all NPC but the pickaxe only to the Miner.

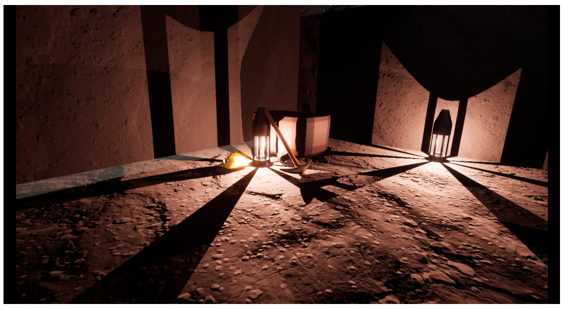


Figure 4.13: A view of all the tools

In the skeleton of the mannequin mesh, are defined several of sockets on a certain position and rotation which will provide a platform to attach the different tools to a specific bone. It can be seen here

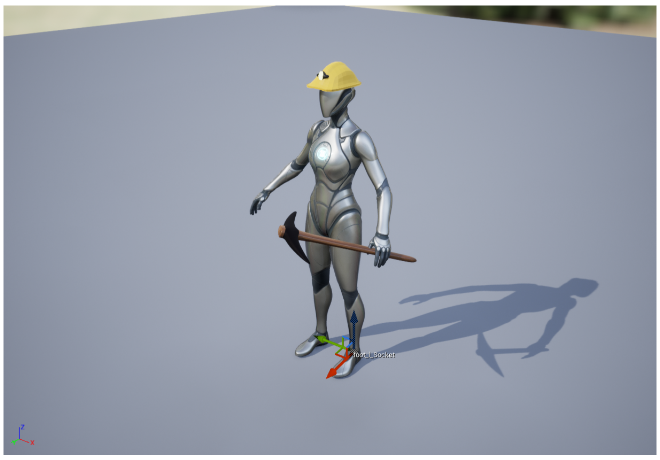


Figure 4.14: Tool set in a Unreal default manequinn

4.1.11 Itch.io page:

The page will be available at: Here

4.2 Fulfilled goals

4.2.1 Design and development in Unreal Engine:

After several months of development, my understanding of Unreal Engine 5 has significantly improved. While I wouldn't claim to have an extensive expertise, my knowledge is sufficient to work with procedural meshes and utilize a wide range of objects available in Unreal Engine. Although there is still more to learn, my current level of understanding is adequate to achieve the desired goal of the project.

4.2.2 AI bots:

The bots developed for game operation have successfully met the objectives outlined in this project. The primary goal was to ensure that each bot is capable of issuing a sequence of actions to an NPC, thereby presenting a challenge to the player. Above all, our aim was to achieve this behavior without relying on conditional statements such as if-else, but instead enabling dynamic changes in response to the player's actions.

To accomplish this, a sophisticated system was implemented to allow the bots to adapt their behavior dynamically based on the player's actions. Instead of using fixed if-else conditions, the bots employ flexible mechanisms that dynamically adjust their actions. This approach ensures that the gameplay remains engaging and unpredictable, as the bots intelligently assess the player's strategies and adapt their own tactics accordingly.

By moving away from static if-else conditions, it has achieved a more fluid and responsive gameplay experience. The bots' ability to dynamically alter their behavior based on the player's actions adds an element of surprise and challenge, enhancing the overall enjoyment and replay value of the game.

Through extensive testing and iterations, it has refined the bot system to deliver a seamless and immersive gameplay experience. The dynamic behavior of the bots not only meets our initial expectations but also surpasses them by providing an engaging and interactive encounter for the player.

4.2.3 Reinforce programming capabilities:

An effort has been made to transition from Unity to Unreal Engine 5, with the aim of changing the programming approach. Initially, extensive research was conducted on Unreal functions, libraries, and pre-existing classes to facilitate their adaptation and utilization in the project. Numerous blueprints were created and their functionality was comprehended in a general sense. Understanding the distinctions between events, functions, and macros proved pivotal in the project's execution. Lastly, and perhaps most importantly, I not only gained a comprehensive understanding of the sublanguage employed by Unreal, but also familiarized myself with its unique notation. This allowed me to develop C++ code that is both compilable and compatible for use in blueprints, thereby combining the advantages of the two programming paradigms. These advantages include the simplicity and readability offered by blueprints, as well as the precision and accuracy provided by C++ code.

4.2.4 Attractive Demo:

The visual quality of the demo is subjective, but considering the scene and taking into account the time and limitations of the project, it can be considered a visually appealing demo.

Chapter 5

Conclusions and Future Work

Contents

5.1 Conclusions **39**
5.2 Future work **39**

5.1 Conclusions

The development process has been challenging and demanding, particularly since it was my first experience working with Unreal Engine. While I had previously conducted a small test using Unreal Blueprints, creating an entire game presented new difficulties. Furthermore, completing the final degree project alongside practical work added an additional layer of complexity. Despite these challenges, I am content with the final result considering the project's scope, time frame, and my level of knowledge. However, given the opportunity, I would have welcomed more time to further refine and expand upon the project.

5.2 Future work

My aspiration is to complete and release my demo, aiming to potentially sell it on platforms like itch.io. Due to time constraints, I was unable to include all the features and ideas I had envisioned for the project.

Looking ahead, my goal is to complete my degree, preferably by next year. I am optimistic about the prospects of working in the video game industry or, alternatively, in a role involving game engines. Ultimately, my dream is to be actively involved in game development, creating games of my own in the near future.

Bibliography

[1] Isaac Asimov. *I robot.* Gnome Press, Boston, 1950.

[2] Matt Aspland. How to create and use cloth simulation in unreal engine 5 (tutorial) | flag physics with wind in ue5. Video file, 2022.

[3] Blender. Blender.

[4] Evans Bohl. How to animate characters in ue5. Video file, 2022.

[5] Epic Games Inc. Unreal official pagpage, 2004-2023.

[6] Nils Gallist. Custom clothes for ue metahuman / ue4-5 / blender 3+ /marvelous-designer 11. Video file, 2022.

[7] Epic Games Inc. Behavior tree overview. Webpage, 2004-2022.

[8] Epic Games Inc. Introduction to blueprints. Webpage, 2004-2023.

[9] John Levine. Monte carlo tree search.

[10] Unreal magic. Rig a character for unreal engine 5 to use it for retargetting and all the animations of mannequin. Video file, 2022.

[11] Edmund McMillen. The binding of isaac. Microsoft Windows, OS X, and Linux, 2014. https://store.steampowered.com/app/250900/The$_Binding_of_Isaac_Rebirth/$.

[12] Lucas Pope. Return of the obra dinn. Windows and macOS, Nintendo Switch, PlayStation 4, and Xbox One, 2018. https://store.steampowered.com/app/653530/Return$_of_the_Obra_Dinn/$.

[13] Videojuegos UJI. Evolutionary algorithms in games.

Appendix A

Source code

The project is located on a git hub repository; Contagia y Pica

To see the code, as is it written in blueprints it is need to have Unreal Engine 5.2 version installed.

To access a build of the project, can be found here.