Hyper Pursuit
Final Report

A BAAQ2 Project
Baptiste Arnold, Abigaëlle Panhelleux,
Quentin Rataud, Angela Saade

June 2022

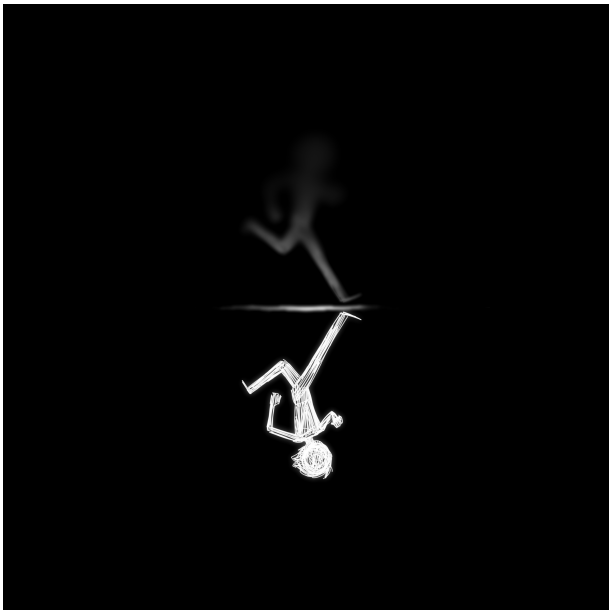

Fig. 1: The game logo

Contents

1 Book of Specifications follow-up **5**
1.1 Introduction . . . 6
1.1.1 The team . . . 6
Composition . . . 6
Name . . . 6
1.1.2 The game . . . 7
Origins . . . 7
Game Backstory . . . 8
Logo . . . 9
Concept . . . 9
1.2 Our Goal . . . 10
Our Ambition . . . 10
Benefit for the players . . . 10
Benefit for the group . . . 10
1.3 State of the Art . . . 11
1.3.1 History . . . 11
1.3.2 Examples . . . 11
Tetris . . . 11
Portal . . . 12
The Witness . . . 12
1.4 Technical Means . . . 13
Git . . . 13
Multiplayer . . . 13
AI . . . 13
1.5 Project Breakdown . . . 14
Planning . . . 14
Distribution of Roles . . . 18

2 Achievements **20**
2.1 Abigaëlle . . . 21
2.1.1 Lore . . . 21
2.1.2 Puzzles . . . 22
2.1.3 Character design . . . 23
2.1.4 Cut scenes . . . 24
2.1.5 Collectables . . . 24
2.2 Baptiste . . . 26
2.2.1 Website . . . 26
I should have done it differently . . . 28
2.2.2 Multiplayer . . . 28
I think i would do it differently . . . 31
2.2.3 Gameplay . . . 32
I think i would do it differently . . . 32
2.2.4 AI . . . 33
I think I would do it differently . . . 34

2.3 Quentin . . . 35
2.3.1 Portals . . . 35
Render . . . 35
Teleportation . . . 36
Proof of Concepts . . . 36
Multiplayer . . . 36
2.3.2 World duplication . . . 37
2.3.3 Collectable . . . 42
2.4 What I would do differently . . . 42
2.5 Angela . . . 43
2.5.1 Test Room . . . 43
2.5.2 First Level . . . 44
The Butterfly . . . 44
The Maze . . . 45
Itinerary of the Players in the Maze . . . 45

3 Conclusion **48**
3.1 What has been achieved . . . 48
3.2 What could have been better . . . 48

1 Book of Specifications follow-up

This section will present how was the book of specifications originally, and will go through every modification that has happened to it. You will find in this section a complete description of the project and its evolution.

Subsection 1.1 will introduce the team and present how the game was planned to be at the beginning of the project and how that has evolved.

Subsection 1.2 will present what our objectives were in creating this game.

Subsection 1.3 will present the game style, and different games of the same style that were part of our inspirations.

Subsection 1.4 will go through every technical mean we planned to use throughout the creation of the game.

Finally, subsection 1.5 will present the expected progression of the game-making process, the initial repartition and how all of that has evolved since the beginning.

Fig. 2: The BAAQ2 team, Lola the mascot included

1.1 Introduction

1.1.1 The team

Composition The team consists of four S2 students and a cat : the four students being two girls and two boys. All of the students are in the A3 class. The cat has been arbitrarily designated as the mascot of the group and the game, as her colors perfectly match the theme of the game. Figure 2 shows a photo of the four (and a half) members of the group. Table 1 shows the complete names and logins of the four (and a half) members of the group.

The group originally planned to include Auguste, another member of the A3 class, instead of Angela. A strong disagreement between the ideas that Auguste had in mind and the ideas that the rest of the group had, has lead to Auguste joining another group planning to do a Virtual Reality project. Angela completed the group afterwards.

The team has not changed at all since Angela joined the group. None of the four team members (neither Lola, as far as we know) has ever completed a team project nor has used Unity nor Blender before this year. Thus, this project has been for all of the four human members a complete discovery of the game-making domain.

Name The team name needed to be short and striking, and should mean something. It was formed with the initials of the first names of the four students : B for Baptiste, the two As for Abigaëlle and Angela, and the Q for Quentin. BAQA was another possibility for the team name, being a word play with “Baka”, meaning “Idiot” in Japanese, but the name opted was a word play

Name	Login
Baptiste Arnold	baptiste.arnold
Abigaëlle Panhelleux	abigaelle.panhelleux
Quentin Rataud	quentin.rataud
Angela Saade	angela.saade
Lola	lola

Table 1: The members of the BAAQ2 team

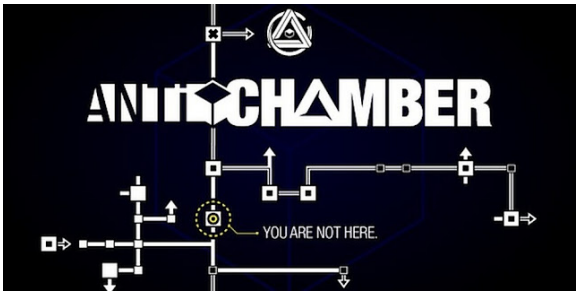

Fig. 3: The confusing map of *Antichamber*

Fig. 4: A screenshot taken from the game *Portal*

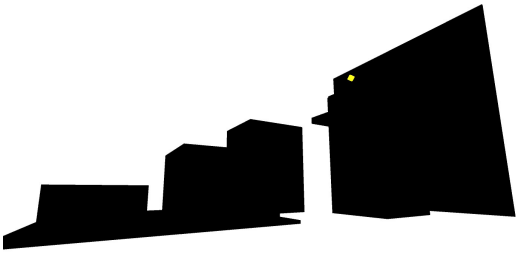

Fig. 5: A typical level of *Contrast*

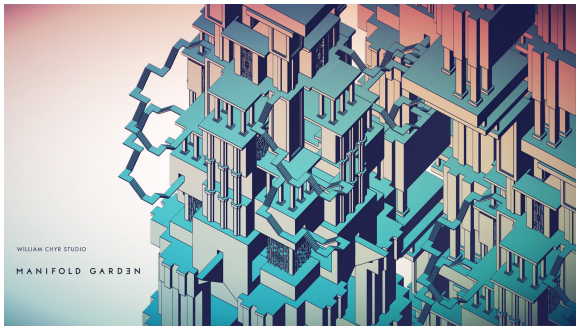

Fig. 6: The impossible architecture of *Manifold Garden*

with “Back to” instead, being a bit more serious. This way, the game can be said to be a “BAAQ2 game”, a “Back to game”.

1.1.2 The game

Origins The first ideas that were brought to the table at the beginning of the project were diversified. There was an idea of a simple maze in augmented reality, or a game where a player was inside the maze and the other one has the map in possession. There was an idea of a lucrative programming game, teaching the player competitive programming and algorithms. There was an idea of a game where you played the shadow of your character.

The final idea that has been followed is an advanced version of the maze, with some twists inspired from other games: The game features some physically impossible events and room disposition. This idea is inspired from *Antichamber*, a puzzle game where the entire world is nothing but a gigantic paradox. Figure 3 shows a humorous map of the game *Antichamber*, where the map is not helping the player. Figure 6 shows a picture taken from another similar game, *Manifold Garden*, where the player interacts in an infinite world inspired by Escher’s work.

The game also features puzzles to solve in an increasing order of difficulty. This is the case in many other puzzle games, *Portal* for instance. Figure 4 shows how the game looks like in game, with obstacles to overcome only using logical skills.

The game is nearly monochromatic, allowing us to easily make optical illusions. From a particular point of view, an entire room could look like a 2D scene in black and white, thus telling a bit of the story behind the game. This is what is called as "collectable", further detailed in subsection 2.3.3. This is an idea that come from a game called *Contrast*, where each room used only two colors, making the disposition of the levels difficult to understand. Figure 5 shows an example of a level in the game *Contrast*.

The game has to be played by two players simultaneously in a cooperative way, like some other cooperative puzzle games, *We were here* for example.

In the game, the two players are chasing a kind of AI that is guiding them through the "maze". This is further detailed in subsection 2.2.4. Because of this and of the characteristics of the game, the game was originally planned to be called "Brain Pursuit", a game that would challenge your logical abilities during the pursuit of a key. The name was changed afterwards to "Hyper Pursuit", to better reflect the "impossible" traits of the game, "Hyper" being a reference to a fourth dimension.

The game backstory features two twins in a coma, having lost their entire memory. This is inspired from the game *Five Nights at Freddy's 4*, that takes place inside a hospital. This is also inspired by a friend having lost his entire memory not so long before the starting of this project.

Game Backstory In order to make the game more interesting, the backstory of the game is a lot more complicated than simply two people in a huge maze. The complete game starts with a cut-scene of the two characters, from the point of view of the white player, looking at the other player through a mirror. The two players try to touch the mirror with one hand, and the cut-scene stops there. It was originally planned that crashing noises would reverberate, the mirror would start to duplicate more and more, creating a mind-blowing effect that separates the two entities,

Fig. 7: A drawing of the two protagonists of *Hyper Pursuit*

pushing them further and further from each other. This is the introduction to the game, and the main event of the backstory: a car crash involving the two protagonists.

In the game backstory, because of this, the two twins fell in a coma, and have entirely lost their memory. In the maze that is their own mind, their goal is to recover memories of the past, in order to wake up finely. Their backstory was supposed to be partly told through cut-scenes and different events. The main story-telling thing is the disposition of the levels, describing some events that the two characters lived in the past. The first collectable for instance (see subsection 2.3.3) relate to a scene involving the two protagonists and a butterfly.

The two players start to play in two very similar rooms, with one major difference: the player controlling the white character will be in a mainly white world, whereas the player controlling the black character will be playing in a similar-looking world, but mainly black-colored. In general, every object being white for a player will look black for the other player and vice-versa.

They need to help each other in order to find an object that is flying in both of their realm, which is the main gaming part : the AI (see subsection 2.2.4). If both players successfully catch it, the game ends with both of them exiting this world forever. But if they do not pick up every collectable representing their memories (see subsection 2.3.3), they do not wake up when they quit this world. This was planned to be represented with the flat sound of vital sign monitors announcing that the characters are gone.

In each level of the game, the two players will be able to pick up a single collectable (see subsection 2.3.3). The collectables are found by placing the character in a very specific place of a level, making the room itself describe a scene, and this was planned to be the starting image of a cut-scene telling what the protagonist is remembering. If both players pick up every single one of them and both catch the final key to end the game, it was planned for them to not end up with the beeping machine: they would have rather opened their eyes, finally.

A deeper backstory was imagined, as a kind of Easter egg, hidden in the game files.

Logo A logo needs to be simple and should give a maximum amount of ideas about the game in a minimum amount of details. The logo is a striated drawing of a man, who is running upside down, with a blurred reflection of himself. The logo only uses the two main colors of the game, black and white. Figure 1 shows the current logo of the game.

The running character represents the protagonist's pursuit of the world exit key, the AI. The striated drawing represents the coma, where nothing is really well-defined. The reflection of the character is its twin, blurred because even the memories involving the other character are initially gone and slowly coming back to their mind. The main character is running upside down to represent the lack of rationality in the bizarre world of the coma.

The logo, as well as the art in figure 7, have been drawn by Nino, using *Autodesk Sketchbook*.

Concept The concept of the game is mainly as follows: the two players are located in two similar huge non-euclidean looking mazes. They have to help each other in order to keep track of the little flying object (see: subsection 2.2.4) that will unlock both the exit and the memories of the two protagonists. For the end to be triggered, both players need to catch the flying object at the same time.

1.2 Our Goal

Our Ambition Our prime ambition was to discover how to have a good workflow for a team project, and to basically learn how to create a game. Secondly, this game has for objective to diversify multiplayer puzzle game. Nowadays, there are much more solo puzzle games than multiplayer puzzle games even though gamers usually prefer playing with friends rather than playing alone. There has always been more enjoyment in playing in co-op than in solo whether it's for chilling, solving problems or just having fun. This is the main reason why the game is a cooperative puzzle game.

Benefit for the players Like any puzzle game, playing *Hyper Pursuit* will make the player improve their cognitive function. In particular, for this game, logic skills will be highly useful in order to solve the different enigmas to progress further and further into the game, and pattern recognition skills will be highly used in order to pick up each collectable. They will also have fun and a huge satisfaction when solving the problems. The game will not disorient the players but will frustrate them: this is where the fun comes ! *Hyper Pursuit* will also allow the player to escape reality for a short time, using its very unique backstory. Moreover, thanks to the multiplayer mode that is in the core of the project, players can have fun with their friends or can meet new people who have the same taste for games.

Benefit for the group The creation of this project has proven to be beneficial for every team member. Because the development of the game is made from scratch, the implementation of an entire project has been an important experience for the group. In addition, this type of project illustrated well the situation in the working environment. The usage of git has been very instructive, since nobody in the group has ever had a concrete occasion to use and manipulate git branches (for instance) before this project. Moreover, implementing this game has improved our problem solving skills: in the making of our game, every problem encountered has made every member try their best in order to tackle it on the spot and find the best solutions. What's more, the fact that this was the first real project made using Unity for every member led to many mistakes, many bad manipulations and many bad decisions being taken. But the reason why these are mistakes is because they ended up being hard to work with later on, and this is the main reason why we learned a lot by doing and noticing these mistakes, whether they ended up fixed or not.

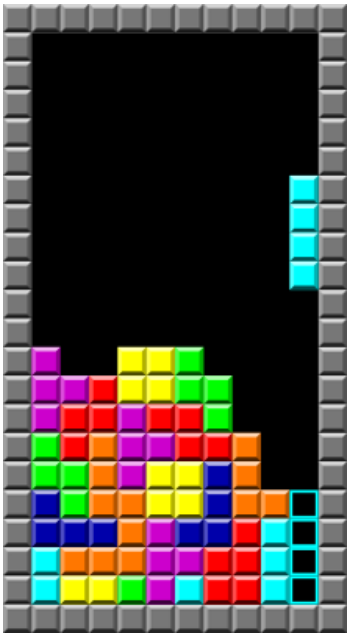

Fig. 8: A typical *Tetris* game screen

Fig. 9: Typical *The Witness* puzzle panels

1.3 State of the Art

Puzzle video games make up a broad genre of video games that emphasize puzzle-solving. The types of puzzles can test many problem-solving skills, including logic, pattern recognition, sequence solving, spatial recognition, and word completion.

1.3.1 History

Puzzle video games owe their origins to brain teasers and puzzles throughout human history. The mathematical strategy game *Nim*, and other traditional, thinking games, such as *Hangman* or *Mastermind*, were popular targets for computer implementation. One of the first puzzle video games was *Blockbuster* released in 1981, it was a computerized version of Rubik's Cube puzzle. Then there were lots of puzzle-platform games like *Lode Runner*, *Door Door*... What revolutionized and popularized puzzle genre video games was *Tetris*, created in 1984. More recently, since the release of *Portal* in 2007, there has been a rise in popularity of physics-based logic puzzle games.

1.3.2 Examples

Tetris *Tetris* is a well known example of a puzzle video game, where players complete lines by moving differently shaped pieces, which descend onto the playing field. The game ends when the playing field is filled. The challenge is to delay this outcome as much as possible, and this requires good logic and pattern recognition skills, as the game speeds up more and more. Figure 8 shows the *Tetris* playing field.

Built on simple rules and requiring intelligence and skills, *Tetris* established itself as one of the great early video games. The Game Boy version is one of the best-selling games of all time, with more than 35 million copies sold.

Portal *Portal* is a more recent puzzle-platform game, consisting primarily of a series of puzzles that must be solved by teleporting the player's character and simple objects using a "portal gun".

Portal was acclaimed as one of the most original games of 2007. It received praise for its originality, unique gameplay and dark story with humorous series of dialogue. Portal is often cited as one of the greatest video games ever made[1].

The Witness *The Witness* is a 2016 puzzle video game involving the exploration of an open world island filled with natural and man-made structures. The player progresses by solving puzzles, which are based on interactions with grids presented on panels around the island or paths hidden within the environment. Figure 9 shows an example of a puzzle panel in *The Witness*. The game provides no direct instructions for how these puzzles are to be solved, requiring the player to identify the meaning of symbols in the puzzles. A central design element to the game was how these puzzles are presented so that the player can achieve a moment of inspiration through trial and error and gain that comprehension themselves.

Within a week of release, the game had sold over 100,000 copies, nearly recouping all of the development costs.

[1] `https://en.wikipedia.org/wiki/List_of_video_games_considered_the_best`

1.4 Technical Means

A lot of different technical means has been used throughout the project. The main tool used to create the game was *Unity*, the game engine, and C# as the main programming language, used for more than 90% of the files. Firstly, for the making of the website, languages like HTML and CSS have been learned and employed. For the trailer, *Cinemachine* allowed us to make beautiful cut-scenes of our game, and *iMovie* has provided the necessary means to edit them. Sounds were planned to be taken for the voices and music with a *ZOOM H4n* mic. The 3D models have been done using *Blender* and *Unity*. *Minecraft* has been used in order to help for the level design. *Photon* has been chosen as the solution for the multiplayer because it was well documented and beginner-friendly. Moreover, it is easily understandable, thus making it easy for the rest of the team to read. In order to easily create an installer for the project, *Advanced Installer* has been chosen for its simplicity.

Git Git is the main software used to optimize our workflow. It allows us to share our codes, our scenes, our prefabs, and to work in parallel. There is a repository[2] for the project, conveniently configured with a specific gitignore[3] and gitattributes[4] made for Unity projects, to avoid uploading binaries and non-necessary files.

Multiplayer The game does not simply use multiplayer mode as a feature: multiplayer mode is the core of the project. It has been decided that the game would only be playable by two players simultaneously. No bot can replace a missing person: without a working multiplayer mode our game is unplayable. When the game executable is launched, the first player can choose to host or to join the party, which will determine the character he will play. The two players need a working internet connection in order to play together, because Photon's servers are used. It is not possible to play using local network.

AI The main Artificial Intelligence present in the game is the little object the players need to catch in order to finish the game. Its behaviour depends on the position of the players, in order to not get caught before the final level of the game. It is visible by both players at the same time, and gives information about the path to take. Subsection 2.2.4 goes into more details about how the Artificial Intelligence works.

[2] `https://github.com/Akilson/Hyper-Pursuit`
[3] `https://github.com/github/gitignore/blob/main/Unity.gitignore`
[4] `https://github.com/github-for-unity/Unity/blob/master/src/GitHub.Api/Resources/.gitattributes`

1.5 Project Breakdown

Planning Table 2 shows the initial expected advancement of every main task throughout different milestones, namely the three defenses. Every concept was expected to be proven to be possible before the first defense. The game was expected to start to be playable, and an entire plan of the game was expected to exist for the second defense. The game was expected to be fully completed and to be working correctly before the final defense. Figure 10 shows the first Gantt diagram that has been made of the project, with the planning for each main task. This diagram has been regularly updated since.

The puzzle conception took more time than expected, because finding interesting puzzles for two players in a world without including nearly any interaction was particularly hard. Apart from this, the planning held until the first defense without much troubles. Two tasks were added to the table in order to take into account the AI creation and the multiplayer system. Table 3 shows the expected advancement of every main task at the time of the first defense. Figure 11 shows the second Gantt diagram that has been made for the project at the time of the first defense.

This new planning was pretty well followed until the second defense. The lack of puzzles due to the unexpected complexity of designing some resulted in only one level being conceived (since there needs multiple puzzles to create one single level). The expectation of a game containing five complete levels was reduced to only one level. Also, the LAN mode project has been given up. Since cut-scenes took longer than expected to make, only some specific cut-scenes have been made, namely the trailer, the introduction and the ending. Finally, due to the presence of only one level, the story could not be entirely told from the levels, but a collectable has been created to show how the concept would work (see subsection 2.3.3). Figure 12 shows the final Gantt diagram corresponding to what really happened in the end.

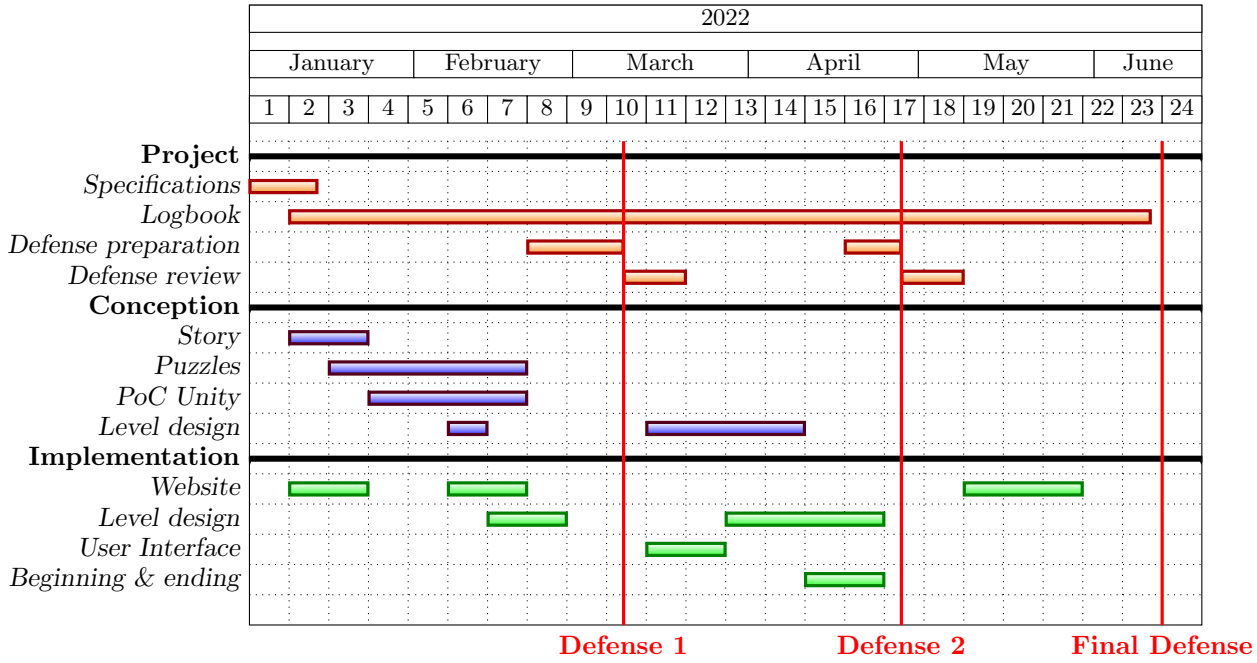

Fig. 10: Initial Gantt diagram for the project

Task	Defense 1	Defense 2	Final defense
Website	Exists	Contains all information	Contains animations
Story	Nearly finished	Entirely written	Told through cut-scenes
Puzzles	Conceived	Implemented	Integrated in the levels
Levels	One is designed	Conceived	Implemented

Table 2: Initial expectation of the state of the project through the defenses

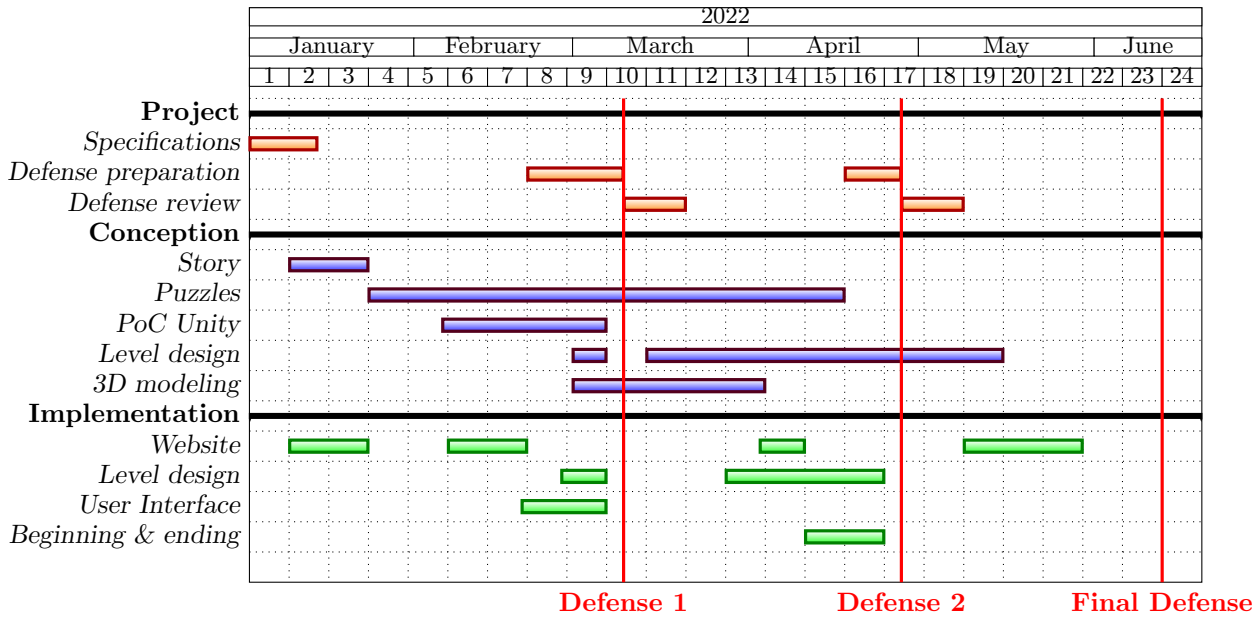

Fig. 11: Updated Gantt diagram of the project at the time of the first defense

Task	Defense 1	Defense 2	Final defense
Website	Exists	Contains all information	Contains animations
Story	Nearly finished	Entirely written	Told through cut-scenes
Puzzles	Nothing	Conceived	Integrated in the levels
Levels	One is designed	Conceived	Implemented
Multiplayer	Nearly finished	Bug fix/finish	LAN mode
AI	Nothing	Nearly finished	Finished

Table 3: Expectation of the project advancement through the defenses at the time of the first defense

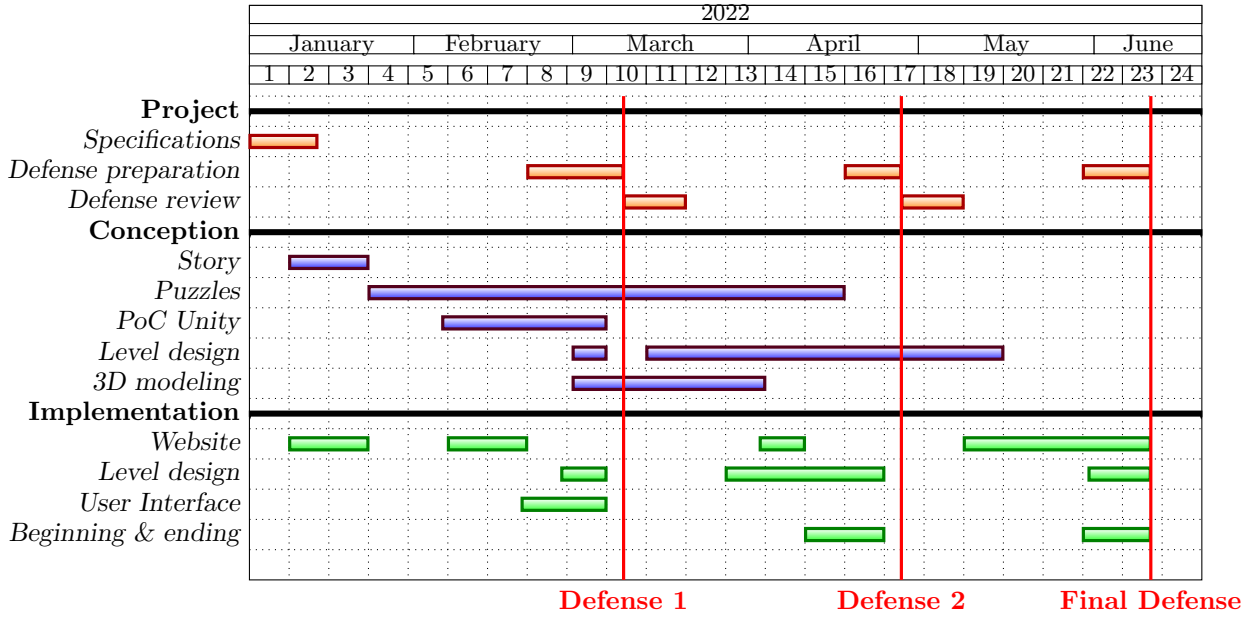

Fig. 12: Final Gantt diagram

Task	Defense 1	Defense 2	Final defense
Website	Exists	Contains all information	Entirely reworked
Story	Nearly finished	Entirely written	A collectable exists
Puzzles	Nothing	Conceived	Integrated in the level
Levels	One is designed	One is implemented	One is playable
Multiplayer	Nearly finished	Bug fix/finish	Fully working
AI	Nothing	Nearly finished	Finished

Table 4: Real advancement of the project through the three defenses

Distribution of Roles Table 5 shows how roles were initially distributed. Even though it was written that a task is taken care of by only two people, each task was planned to be completed with the help of every other member of the group. For instance, the coding process of the website is done by the person in charge of it, but the layout, the design and the texts could come from every member.

Overall, Abigaëlle was mainly expected to be in charge of the game backstory and the cut-scenes ; Angela was mainly expected to be in charge of the level implementation and the multiplayer system ; Baptiste was mainly expected to be in charge of the website and the proof of concepts in Unity ; Quentin was mainly expected to be in charge of the puzzle and level design.

It turns out that this did not hold very much. In fact, Table 6 compares the initial role of every member against how much the task has been carried out by every member. A green cell indicates that the person was the person who did much of the work. A yellow cell indicates that the person was implicated in the application of the task. A orange cell indicates that the person was involved at a small degree in the task. A red cell indicates that the person did not participate at all in the making of the task.

Task	Abigaelle	Angela	Baptiste	Quentin
Website			in charge	substitute
Story	in charge			substitute
Puzzles	substitute			in charge
POC unity		substitute	in charge	
Level design	substitute			in charge
Level implementation		in charge	substitute	
Multiplayer		in charge	substitute	
AI	substitute			in charge
Cut scenes	in charge	substitute		

Table 5: Initial role distribution

Task	Abigaelle	Angela	Baptiste	Quentin	Lola
Website			in charge	substitute	
Story	in charge			substitute	
Puzzles	substitute			in charge	
POC unity		substitute	in charge		
Level design	substitute			in charge	
Level implementation		in charge	substitute		
Multiplayer		in charge	substitute		
AI	substitute			in charge	
Cut scenes	in charge	substitute			
Pet & Cuddle	substitute				manager

Table 6: Final role distribution heatmap

2 Achievements

This section will present for each individual what were his tasks, what he has successfully done or not, what kind of issues were encountered, how these issues were or were not resolved, and most importantly what he has learned from this project, for instance, what he would have done differently if he had to redo the project today. The layout has been chosen to be the individual layout, because every task has been given to different members, and it occurred only rarely that two members were working on the same specific task.

Subsection 2.1 will go through every task assigned to Abigaëlle, that is mainly the backstory, the cut-scenes, the trailer, and the 3D modelling.

Subsection 2.2 will go through every task assigned to Baptiste, that is mainly the website, the gameplay, the multiplayer system, and the AI.

Subsection 2.3 will go through every task assigned to Quentin, that is mainly the portal system, the collectable and the world duplication system.

Subsection 2.5 will go through every task assigned to Angela, that is mainly the level design.

2.1 Abigaëlle

This section will present the tasks that were assigned to Abigaëlle. It will present for each subsection what was expected to be done, what was done and how, the issues encountered and their solutions. Overall, Abigaëlle did what she needed to do, but mostly alone, with no help from the other team members, and that explains why she didn't participate a lot with the other members, and was working a lot more alone.

Subsection 2.1.1 will cover all the work done for the lore, from the research to the modification of the lore. The lore is the most important job of Abigaëlle.

Subsection 2.1.2 will cover where Abigaëlle helped for the puzzle, mostly on Minecraft.

Subsection 2.1.3 will cover all the work done for the character design, stating all the learning that goes with it, and with the implementation of the animations and it's problem.

Subsection 2.1.4 will cover all the work done for the cut scenes, with the creation to the montage of the videos.

Subsection 2.1.5 will cover all the work done for the collectables, that are two, from the research to the creation of the collectables.

2.1.1 Lore

The lore is the most important and hard job Abigaëlle had to do in this project. But it is also, sadly, the part that is the most difficult to see in game.

The first step she had to do was a lot a research. And really a lot. She already had an little idea of what she wanted the lore to be : a puzzle game made to make your brain go crazy needs a lore that will make any brain go crazy too. The most perfect example is the horror game FNAF. Abigaëlle was already a big fan of the Five Nights at Freddy's lore, that is still at this day not finished even after almost 10 years of deep research, and thirteen games released. Abigaëlle was just so impressed by the simplicity of the game but the complexity of the lore, that is still just theories and was not even approved by the creator, Scott Cawthon. For most of the research, Abigaëlle then watched the 50+ videos of at least 20 minutes of a theorist that she like a lot : GameTheory. She had two jobs while enjoying the videos : understand the lore of FNAF, but also see how the creator implemented the lore into it's games. This took her more than two months to fully understand it.

Tasks	Defense 1	Defense 2	Defense 3
Lore	100%	100%	100%
Character design and animations	0%	90%	100%
Cut scenes	0%	50%	100%

Table 7: Expected timetable of Abigaëlle's tasks

Tasks	Defense 1	Defense 2	Defense 3
Lore	50%	100%	100%
Character design and animations	10%	75%	100%
Cut scenes	0%	50%	100%

Table 8: Realised timetable of Abigaëlle's tasks

Fig. 13: Five Nights at Freddy's

Fig. 14: GameTheory Youtube video

This game is the story of 6 characters, all included in a tragic incident. This incident occurred not so long ago : the newspaper says that two police mans were severely hurt during a rescue mission. The two rescuers are twins : Kate and Aiden. They were investigating into a kidnapping case. 4 teenagers, who were studying in a programming university, were kidnapped by a man never caught by the police, James Dow. After an investigation, the two characters discover the location of the missing teenagers. Sadly, as the police mans arrived at the place, the kidnapper made the car crash, wanting to kill them. The twins got severely injured, and were put into a hospital quickly, while James Dow escaped. The only good news in this, is that the teenagers were abandoned by the kidnapper, so were set free and found by the rest of the police. The twins were put into a coma, and died a few weeks later. The game is actually made by the teenagers that were kidnapped. The game's purpose is to thank the twins that lost their lives for them, but also to tell everyone the identity of their kidnapper, find him and bring him to justice. The different rooms represent the five steps of grief. Collectables can be found in the game to give clues about the story. It is hidden, and the players have to be investigators, just like the twins, to gather all the clues and discover the truth. In the game, the game is structured into 5 chapters, each of them representing a room and an emotion of grief : denial, anger, bargaining, depression, and acceptance.

2.1.2 Puzzles

For the puzzles, Abigaëlle mostly worked on them at the beginning of the project, with Quentin. Together, they created a minecraft server to test their ideas. While Quentin worked on the telepor-

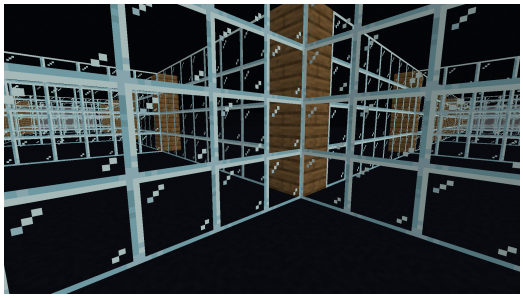

Fig. 15: Some Minecraft puzzle and cut scene tries

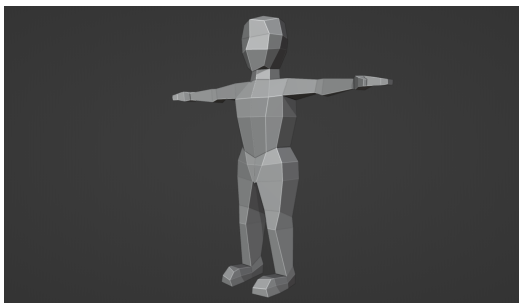

Fig. 16: The character design of the players

tation, Abigaëlle worked mostly on the puzzles. She tested some puzzle that were already created on other games, like pattern switches, and much more. Then, she helped a bit more Quentin with the teleportation, testing them and figuring out how to put some of the lore and the collectables in the ideas.

2.1.3 Character design

Abigaëlle started having this idea after doing a tutorial made by Unity. In their tutorial, it was asked to create a mansion game, and there was a character with a .filename that she did not know. With a bit of research, she found out that we could, with some 3D modeling app, import 3D character. With that in mind, Abigaëlle downloaded Blender. She knew that the team wanted a character design that could be both for a woman or a man. Because she didn't know anything of Blender, she decided to watch the tutorial videos Blender gives to the users on YouTube. Then, it gave a tutorial to make a owl, and decided to put what she learned into motion.

After finishing the owl, Abigaëlle started to work on the character. With the team, they decided to have a low poly character, that is a character with a low number of polygons, with no face. Having no face and minimized hands and feet is a characteristic that dreams have. To do that, I watched a tutorial on YouTube, and changed some details to fit what the team wanted to have. For the shading, Abigaëlle did not touch it, and left it as the basic skin. It was in Unity that the skin was changed to black for one of them and white for the other.

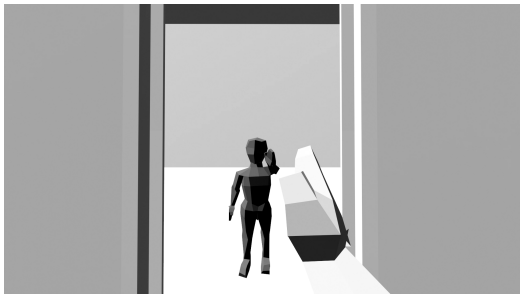

Fig. 17: First cut scene

2.1.4 Cut scenes

During the project, Abigaëlle made three big cut scenes : the beginning cut scene, the ending cut scene, and the trailer. As stated in the character design part (2.1.3), she had to first learn Blender in order to do all the cut scenes. Almost none of the rush are done in Unity (except for the trailer, where parts of the game were used).

The longest cut scene she made is actually the first one : first with the learning that made it so long to make, but also because it needed a lot more animation. It is with this specific cut scene that Abigaëlle first learned how to animate. She watched at a lot of videos and tutorials, and learned to save her work every minute (or else she would lose a day of work... Yes, that did happen a couple times).

The last cut scene was actually a lot easier to make, mostly because Abigaëlle had learned a lot and could easily make animations now. She created multiple times the room though : at first she wanted it to be a rooms where the players could play into, and not a cut scene, but wanted it to be in Blender. But when importing the room, there was no collision, so Abigaëlle recreated the room in Unity, before realising that she preferred the last cut scene in a video, just like the first one, so she had to change the Blender room again to fit what she had recreated on Unity. Just after this she was able to render it... Three times, because... Even today she does not know. A lot of complication, that made her computer doing everything but what she asked it to do.

For the trailer, it was a lot easier. Maybe because there just needed footage, and no Blender animations. She used her montage application on Mac, iMovie, and used the footage of the two cut scenes she made, plus some footage of the actual game she made alone by building two times the game to be able to enter the room. But it could not go as planned right? iMovie started to crash every time Abigaëlle wanted to change something. And when, after reloading almost thirty times (yes she counted, she couldn't do anything else either way), she managed to finish the trailer... To have another problem. Her Mac would not let her export the trailer. After doing it 4-5 times, she got fed up and decided to just do a screen capture, and edit it a bit to make it a lot more watchable. Then, she sended it to Baptiste so that he could put it into the website.

2.1.5 Collectables

The collectables that can be find are or the number of two. There is first the collectable of the level : a big butterfly that can be seen at the beginning of the game if the player only look up and does not move. For this collectable, Quentin helped her, and first asked her what pieces of the lore

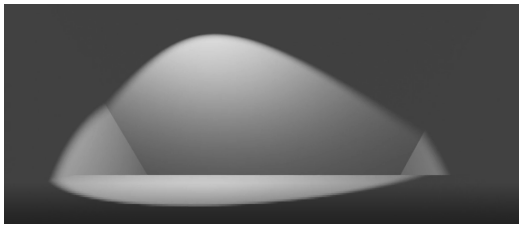

Fig. 18: Second cut scene

Fig. 19: Trailer

were needed for the room, that was called Denial. After Abigaëlle broke up the lore, and Quentin and her had some long complicated discussion, they were able to know a bit more what needed to be done in a lore point of view. While Abigaëlle first wanted a cut scene if the characters find the collectable, Quentin found an idea a little bit less obvious but a lot more poetic : take a butterfly, that can be seen as a sign of hope, or at least what is left of hope, and not add any cut scene, to just leave the players very confused. Abigaëlle already wanted to do a collectable where it needed to be in a particular position in the room for it to be seen, so Quentin took her idea and implemented it into the room.

The second collectable is one that can not be found in the game. The collectable is actually the identity card of both of the characters that can be played : Kate and Aiden. Abigaëlle wanted the identity of the characters to be found, but most importantly that the fact that there are twins be revealed. This collectable can be found in the website : if the word "Hints" is clicked, the two identity cards will automatically be downloaded into the computer. Abigaëlle made it also clear with what is in the identity card : the matching birthday, but also the last name, that is hexadecimal for "Twins". The collectable was made with Paint, but also modeled in Blender because at first Abigaëlle wanted it implemented into the game (but at the end wanted it in the website).

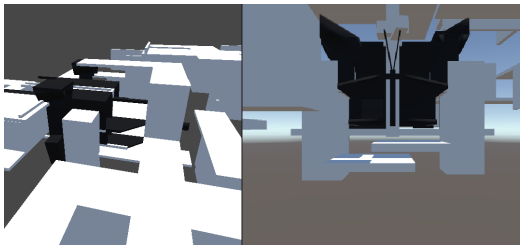

Fig. 20: First collectable : the butterfly

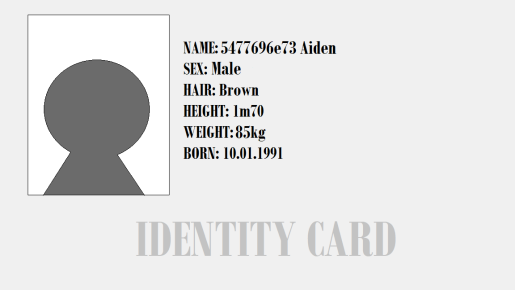

Fig. 21: Collectable : Identity Card of Aiden

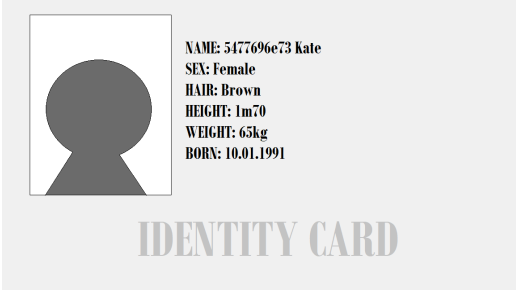

Fig. 22: Collectable : Identity Card of Kate

2.2 Baptiste

This section will present the tasks that were assigned to Baptiste. It will present for each subsection what was expected to be done, what was done and how, the issues encountered and their solutions.

Subsection 2.2.1 will cover the creation and the evolution of the website throughout each of the defenses. It will for instance show why and how the project had, during three months, two different websites.

Subsection 2.2.2 will cover the creation of the multiplayer system and explain how it has been adapted to the project. It will for example explain how a simple exclamation mark may completely destroy the multiplayer mode.

Subsection 2.2.3 will cover how every player moves in the world. It will for instance expose one of the major issues of the game: the camera moving by itself if there is an unfortunate contact with a wall.

Subsection 2.2.4 will cover the creation and the implementation of the AI in levels, as well as his evolution. It will for example explain how to finish the game.

2.2.1 Website

As explained in Section 1, the project is a puzzle-game. So at first the idea was to have a website linked to our game, which means hiding some hints and achievements in the website. Unfortunately, this idea had never been realised and a more classic website have been made. Here is why :

The website was the first thing realised in the project. Before even starting to code something, HTML and CSS needed to be learned by Baptiste, because he had never worked with front-end

before. So, after following a lesson of OpenClassroom[5] the website has been coded. But because it was the first task of Baptiste and that he has never made a project alike before, he went a little bit too fast and made a huge mistake. Because of his lack of experience in the process of making a website, he started to code without any design of the website. Because the backstory was not yet done at that time, he started with only a vague idea of what he wanted to do. So, because the design was not fixed since the beginning, a lot of time have been lost. For the first defense a website was online, but it was not really nice.

Then as it shown on table 9 the website needed to be almost finish for the second defense, but as it can be seen on table 10, between the first and the second the website didn't progress. But it doesn't mean that no work was effectuated in front-end. During the second defense's preparation Baptiste started to have doubt about his abilities to make a nice website, linked to the game. So to make it easier and because no one in the team was able to make a good design of website, a template has been chosen to make a new website (as shown on Figure 24). So, concretely, nothing has been made on the previous website (as shown on Figure 23) but a new website has been created and put online.

For the final defense, a decision needed to be made between the 2 website: which one should be kept ? The template is nice but hard to modify (more than 12 000 lines of CSS...), and the first version is simple to modify (because all of the code has been made by Baptiste) but very ugly. In the end, it has been decided to use the website inspired from the template. The structure of the template was mainly keep unchanged because the responsive-design of the template was really nice. All the images have been changed as it can be seen on Figures 24,26. Two buttons have been added to download a full and a lite version of the game as shown on Figure 24. The trailer of **Abigaëlle** has been added to the website. It has been hard to make it responsive because there was originally no trailer on the template. Then a table with the project breakdown (as shown on Figure 26 has been put to replace an image. A short description of the project has been written too. If more details about the project are required, all of our reports can be downloaded on the website, as shown on Figure 27. Finally, as the footer of the website, there is the contact field, as

[5] `https://openclassrooms.com/fr/courses/1603881-apprenez-a-creer-votre-site-web-avec-html5-et-css3`

Tasks	Defense 1	Defense 2	Defense 3
Website	50%	80%	100%
Multiplayer	60%	90%	100%
Gameplay	50%	100%	100%
AI	0%	50%	100%

Table 9: Expected timetable of Baptiste's tasks

Tasks	Defense 1	Defense 2	Defense 3
Website	50%	50%	100%
Multiplayer	75%	90%	100%
Gameplay	90%	100%	100%
AI	0%	30%	100%

Table 10: Realised timetable of Baptiste's tasks

shown on Figure 28.

I should have done it differently If i need to do a website again i think i would do it differently. I will first ask to my team to make a kind of book of specifications (colors,images,text,...), in order to making it easier for the design. I will also ask for more help, for example I could ask to my brother (that is an artist) some ideas about the design. Then I will draw a draft of my website before starting any code. The draft will represent the website divided in different blocks corresponding to my needs (header,footer,contact block,navigation bar,...). If the project does not need a particular website with special features I will not hesitate to take a template and to modify it. Finally, even though I had a lot of difficulties, it was a good experience to create a website. I discovered new languages (HTML,CSS), and it is a pleasant feeling to see something of concrete and nice coming from my code. Next step is to learn JavaScript to make an even better website !

2.2.2 Multiplayer

As mentioned in Section 1.1.2, the game is a cooperative puzzle game, so a working multiplayer is a must! The only desire of the team about the multiplayer was to have it work as soon as possible in order to make tests. Besides this nothing more were asked, so it has been decided to make a simple, classic multiplayer.

After experiencing the pain of restarting or regressing with the website, the multiplayer has started to be coded only after having in mind the exact objective and some ideas of possible

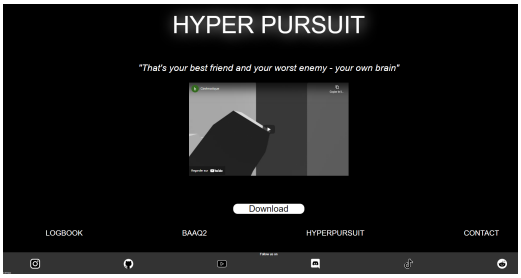

Fig. 23: The first website

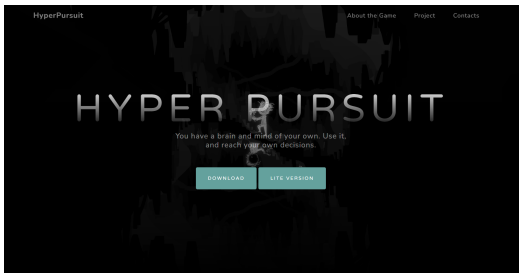

Fig. 24: The website from the template

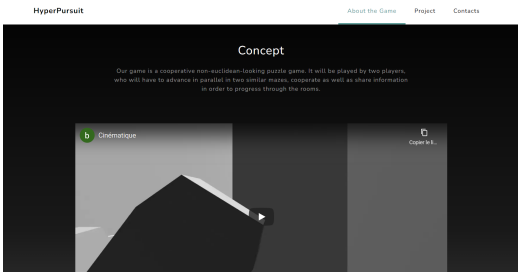

Fig. 25: The first website

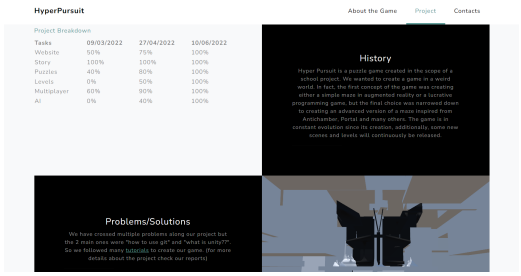

Fig. 26: The website from the template

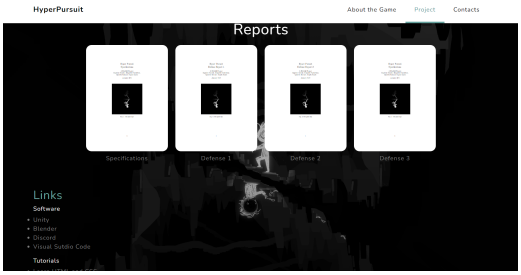

Fig. 27: The website from the template

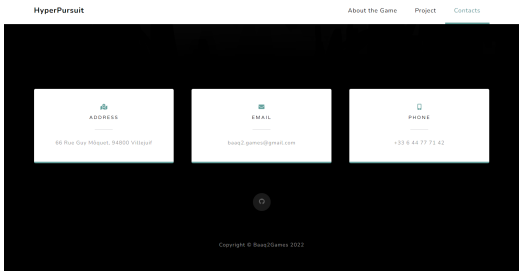

Fig. 28: The website from the template

implementations. For the multiplayer two solutions were possible: Photon[6] or Mirror[7]. The two were interesting. It has been decided to use Photon because it was looking more beginner-friendly and seems well documented with a lot of different tutorial. The only inconvenient of Photon is the limitation of 16 player per room, but in our case it has no impact. The multiplayer is highly inspired of a tutorial designed to create a multiplayer for an fps game. At first there is as shown on Figure 29, a loading scene that will be triggered each time that an action requires some time to be executed. For example when connecting to the server or connecting to a room. Then there is the main menu as shown on Figure 30, it is the first thing displayed after the loading scene when launching the game in lite version. The player can decide from this menu to either create a room, either find a room or just quit the game. If the create room button is clicked then the create room menu is displayed as shown on Figure 31. This menu contains an input field were the player put the name of the room he wants to create. Then by clicking on the button create room the player will create and join the room as the host. There will then be displayed the room menu as shown on Figure 33. It is composed of a list of all the players in the room, a leave room button and a start button active only for the host. The start button launch the first scene when clicked. A C# code, instantiating a room manager that will instantiate a player manager for each player in the room is triggered when the first scene is loaded. Each player manager will then instantiate a playerController (the playerController will be explained more in detail in Subsection 2.2.4). Finally there is also a find room menu as shown on Figure 32. It contains a list of button associated to all the created room. By clicking on one of those buttons the player will join the room associated to it. So this was the multiplayer created for the first defense.

[6] `https://www.photonengine.com/en-US/Photon`
[7] `https://mirror-networking.com/`

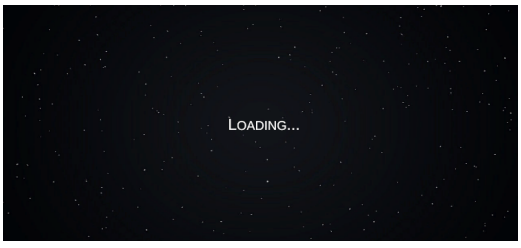

Fig. 29: Loading menu

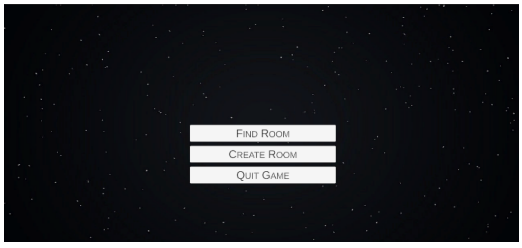

Fig. 30: Title menu

Now that the game had a working multiplayer, the new objective is to adapt it to the game. So for the second defense the multiplayer should be working and in cohesion with our game. Before adapting our multiplayer some bug had to be fixed. For example in the find room menu there were no list of rooms. Each time a room was created the precedent was destroyed. It has been fix easily, the error that was coming from the youtube tutorial had been discovered by a viewer. Thus a solution was proposed in comments. As explained before the game is co-operative and played by two players, so the room must be limited to two players. To do it, in the join room menu, the button of a room is enabled only when there is less than two players in the room. Moreover when a room contains at least two players his name is change to "room is full", so that players understand why they can't join the room. Then the start button in the room menu should be fixed. It is active as long as there is a host. It means that a player could start a game alone, and it is something to avoid because the game is designed to be played by two players. To fix this instead of setting active the start button to the host, it has been set active to the Client of the room. If there is a client there is a host, so there is two players. Thus we can launch the game. Then the last fix to do was concerning the instance of the playerController. Quentin needed two different playerController for the portals. So a clone of the playerController have been added to the prefabs, with a subtle change in material(one is white, an the other is black). To instantiate two different prefabs, it just check whether the player is the host or not. Depending on this a different prefab is instantiate. This fixed and adapted multiplayer have been presented for the second defense and it was considered as working and finished.

Nothing should be change or done on the multiplayer for the last defense. But a problem has been detected just before the last defense. The first scene was not loaded for the two players, it was not synchronized. Each time a room was launched only the player that had the start button was loaded to the game scene, the other player was stuck in the room menu. A lot of time has been lost to understand were this bug was coming from. Finally the problem was coming from the fact that it was the client that was calling the function "LoadLevel" from Photon. Unfortunatly this fonction loads the level given in index for all the players in the room if it is called by the host of the room. But the start button was set active to the client, so it was the client that was calling this function. Thus a simple exclamation mark as shown on Figure 34 had destroyed the multiplayer. The exclamation mark has been removed, and the multiplayer is working again. Finally the start button will not be fixed to be active only if there is two players.

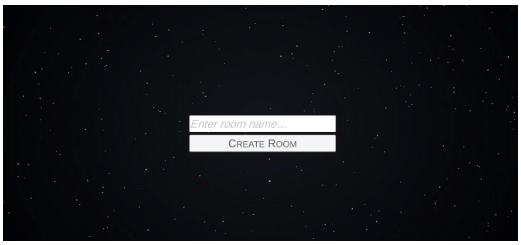

Fig. 31: Create room menu

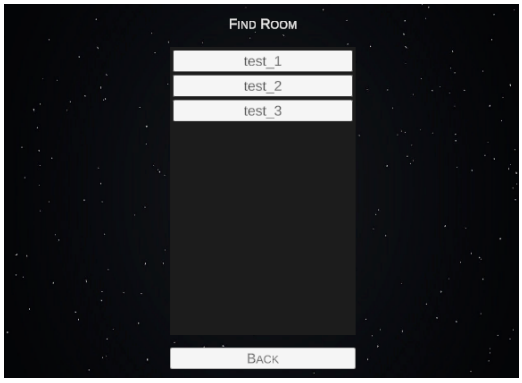

Fig. 32: Find room menu

I think i would do it differently If I was asked to make a multiplayer again, I think I would do it differently. First I will ask to be followed by at least someone else in this task, because it was really stressing and scary to know that I was the only one able to fix or create bugs in the multiplayer. For example when I discovered that the multiplayer was not working anymore 3 days before the last defense, I was terrified. No one was able to help me and if it was not working for the last defense it could have been disastrous for the group. Even though it was a simple exclamation mark to remove to fix the multiplayer, it would have been debugged faster at two. And it would have been less stressful during all the project if I had known that I have a backup. Otherwise It was really a nice experience, Photon is simple to understand. For someone who had never coded before arriving in Epita, it procured a really nice feeling when I saw that I succeed to make a multiplayer and that I was understanding the code. Next step is to do a multiplayer without following a tutorial, just with Photon[8] documentation and my (Hyper)brain.

[8] `https://www.photonengine.com/en-US/Photon`

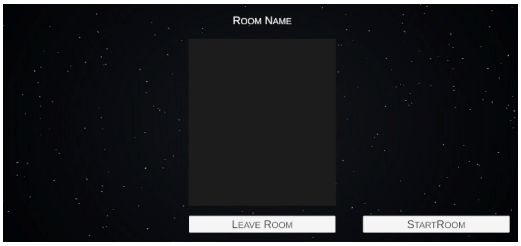

Fig. 33: Room menu

```
    //the start button is active if only we are the host of the server
    startGameButton.SetActive(!PhotonNetwork.IsMasterClient);
}
//call when the host leaves the room
3 references
public override void OnMasterClientSwitched(Player newMasterClient)
{
    //the start button is only accessible to the new host
    startGameButton.SetActive(!PhotonNetwork.IsMasterClient);
}

    //the start button is active if only we are the host of the server
    startGameButton.SetActive(PhotonNetwork.IsMasterClient);
}
//call when the host leaves the room
3 references
public override void OnMasterClientSwitched(Player newMasterClient)
{
    //the start button is only accessible to the new host
    startGameButton.SetActive(PhotonNetwork.IsMasterClient);
}
```

Fig. 34: How to destroy a multiplayer

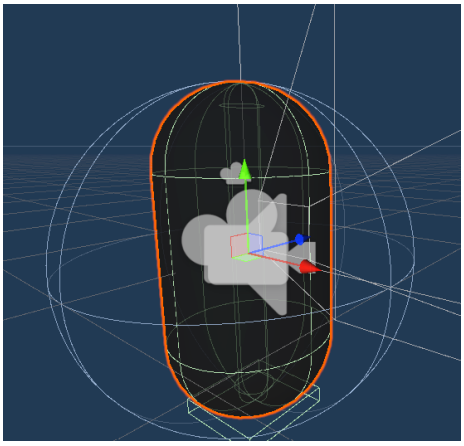

Fig. 35: CharacterController

Fig. 36: Pause Menu

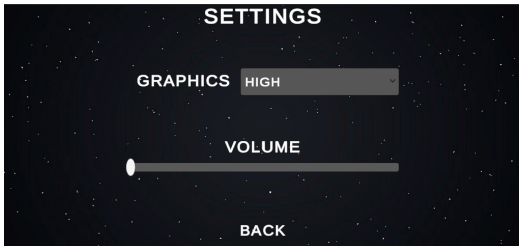

Fig. 37: Settings Menu

2.2.3 Gameplay

The Gameplay is really basic. The game only requests to have a player that can move freely on the X, Y Axis and that can jump.

For the first defense the objective was to have done 50% of the gameplay as shown on Figure 9. What it really means is that for the first defense a working character-Controller was expected. Whether it comes from an Asset or it is coded manually didn't matter. Finally it has been done manually. As shown on Figure 35, the player is represent by a capsule that contains a camera in it. Classically the movement depends on the inputs w,a,s,d and space. The movement also depends of the mouse, the character is always moving to what's pointing the camera. The jump is made thanks to a gameobject attached to the bottom of the capsule. It checks whether the capsule is touching the ground or not by checking if the gamobject is in contact with the ground. If it is not touching the ground jumps are not available.

Because the capsule was working well during the first defense it didn't seems like it was necessary to modify it for the second defense. This was true until we discovered a little problem. When the capsule is entering in collision with a corner, the capsule start to turn on himself. One of the solution could be to freeze the Y axis of the capsule. But the movement with the mouse depends on the rotation of the capsule so the Y axis can't be froze. Finally it has been determined as not to important so it has not been fixed for the second defense. Even though this problem has not been fixed for the second defense the characterController have evolved between the first and the second defense.

A Pause menu has been added to the controller as shown on Figure 36. It is set active when the key "esc" is detected. It stops the game and allow the player to have access to the settings menu as shown on Figure 37 by clicking on the settings button. To quit the Pause mode there is the resume button that will, when triggered, un-pause the game and disable the pause menu. Finally the controller with the pause menu has been presented to the second defense.

For the last defense the only thing to do was to fix the rotation problem. But because the time wasn't managed correctly and because new more important problems have occurred it has not be done. So the final character controller has 2 problems unfixed. T he rotation and another that was here since the beginning but that can't really be fixed. When the controller is passing trough a portal if his speed is to high it will not be teleported. This problem of teleportation can't be fixed because the speed can't be set depending on the performances of the player's computer.

I think i would do it differently If i should create again a character controller i think i would do it differently. I think that i will just take an asset of Unity because it is free, well coded, and

now that i know more about unity it doesn't seem to difficult to understand. Even though i didn't pass a lot of time on the character controller, it was fun to do it, and it helps me understand how unity works in general.

2.2.4 AI

As it has been told in the Section 1.1.2 the game needs to have an AI that will guide the players to the end of the game. The first idea of AI the group had was a fleeing AI. It means that the AI was moving towards a destination but if a player comes to close to it, it change it's trajectory to go to the opposite side of the player. Moreover the idea was to calculate the new trajectory in function of the two players, making then an untouchable AI.

For the first defense there were no AI, not even a slice of code about it. As scheduled the creation of the AI started after the first defense. A patrolling AI has been created for the second defense as shown on Figure 38. It has been made in another room, it was just to show that we add the method. The AI works in quite simple way. First, the field of movement needs to be determined, so that the AI knows where he can move. Once done, waypoint(It is an empty gameobject with just a transform component) can be created and put on the field of movement of the AI. Then the script will make move the AI toward a waypoint and each time it is close to the waypoint, a new destination for the AI, that correspond to the next waypoint in the array of waypoints as shown on Figure 39, is set. So the script loop in the array of waypoints until the application is quit.

Then between the second and the last defense a new idea of AI has been chosen because an AI that calculates the distance between the players and move in accordance with was to complicated

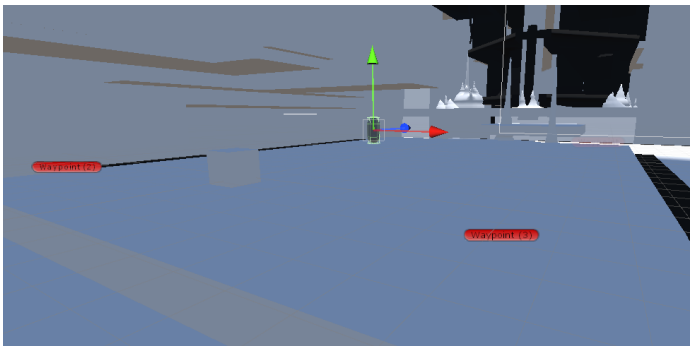

Fig. 38: Patrolling AI

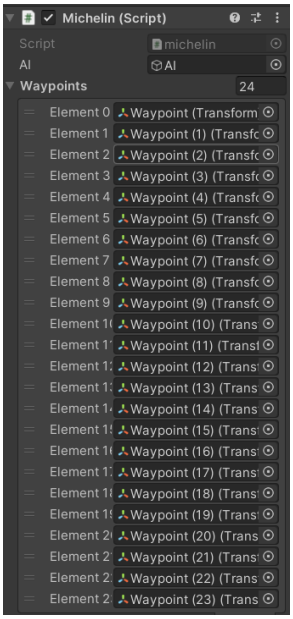

Fig. 39: Waypoints array

to implement in the limited time available. So the game's AI became a simple patrolling AI that change patrolling area each time it is triggered by a player. To trigger the AI one of the two players needs to be close enough of the AI. But there is a little problem, the AI doesn't know how to jump so it can only move on planes. On the patrolling field there is no problem because it is always planes but to pass from a patrolling area to another jump are required. To avoid the jumps, when it comes to changing patrolling area the AI "flies" thanks to a translation. The translation is made thanks to the builtin function Lerp. To finish, the AI that is a capsule, like every elements that have movement in the project, has his mesh render disabled. Instead of the capsule, there are particles. It is nicer. So the final AI is a guide that show the path to follow.

I think I would do it differently If I am asked to do an AI again, I will accept with joy but I think I would do it differently. First I will give me more time to do it. Then I will look into more documentation and try to make something of more interesting than just a simple patrol AI. To finish I will try to find someone else to help me in the creation. Next step machine learning ?

2.3 Quentin

This section will present into more details the tasks that were assigned to Quentin. It will present for each subsection what was expected to be done, what was done and how, the issues encountered and how they were resolved or why they were not resolved.

Subsection 2.3.1 will present into details the central element of the game, that is the portal system. The subsection will go through how the portals works, firstly in single player mode, and some examples of proofs of concepts made using them, then secondly in multiplayer mode.

Subsection 2.3.3 will present the collectable.

Subsection 2.3.2 will present the world duplication mechanism that allows both player to play in a similar but different world.

2.3.1 Portals

Render The main method we used to achieve impossible room disposition was the use of portals. Each portal is linked to another portal. This way, when the character traverse through a portal, he gets teleported to the corresponding portal smoothly.

In order to achieve the smoothness of the teleportation, a specific image is computed and displayed on the portal at each frame. This is done with the use of *RenderTextures*. A well-positioned camera outputs its view on a RenderTexture, associated to the Material of the portal. The position of the camera is computed in a C# script. The camera is placed in order to have the same view of the second portal as the player see the first portal. A shader is then used on the material in order to "cut" the second portal from the view of the camera and paste it on the Render plane of the first portal.

Figure 44 shows a simple example of a room which uses two portals. When the character – represented as the default grey capsule – traverse the black door, it gets teleported to the other room as if the other room where right behind the wall, in spite of the room being slightly rotated and located further away. Figure 45 shows the view of the camera and the view of the player. The corresponding portal viewed from the camera is copied to the material of the portal viewed by the player. From the point of view of the player, it looks like the other room is located right behind the door.

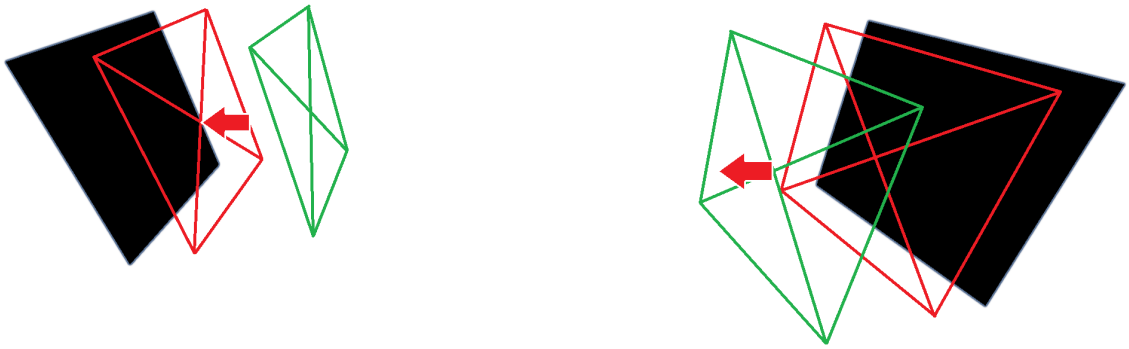

Fig. 40: How the teleportation works through portals

It is to note that, in order for the camera to see the room behind the portal, the black wall between the portal and the camera has been made transparent to the camera. This has been done using *culling masks*, which allows each camera to be set to only display a subset of the available layers.

Teleportation Because we do not want the player to hit the portal and slow down during the teleportation nor to see what is behind the portal, the player needs to be teleported a bit before reaching the render plane. This is why we use a *Collider plane* which will execute the script to teleport the player once he collide with it.

Unfortunately, we do not want the player to get simply teleported to the Collider plane of the corresponding portal, because then the player would either get teleported back (since he would be colliding with the other collider plane), either be able to reach the render plane if we do not teleport it. To fix this issue, we teleport the player to another *Receiver plane*, located further away from the render plane.

Figure 40 shows an example of placement of a pair of portals and their respective planes. The render planes, displaying a cut part of a camera, are represented by a black rectangle. The collider planes are denoted by red rectangles, and the receiver planes are denoted by green rectangles. When the player collide with a collider plane, he is teleported to the corresponding receiver plane.

The offset between the planes needs to not be too small – otherwise the player could collide with the other collider plane when teleported – and not be too large – otherwise the player would get teleported too soon.

The placement of every plane needs to be extremely precise, as any offset would result in a visual glitch that would betray the entire effect.

Proof of Concepts In order to use the portals as efficiently as possible, their use needed to be as simple as possible. That is why a prefab of two linked portals has been created.

To explore the field of possibilities of what can be made using portals, some proof of concepts has been done.

The example Figure 44 is one of them. No matter where the two rooms are located, from the point of view of the player, the two rooms seems to be next to each other.

The second concept done was an impossible tunnel, looking smaller from the inside than from the outside, using only a pair of portals. Figure 46 shows how the tunnel looks like from the inside. Figure 47 shows how the tunnel looks like from the outside. The result is incredibly disturbing. The player can go through it instantly, but has a long way to go to contour it.

This effects is the result of a pair of portals located near the extremities of the tunnel. Figure 48 shows how the portals are placed in order to get this incredible effect.

A third concept was a small room, where it seems like we need to do eight 90-degrees right angle to explore it entirely, even though there only seems to be one pillar in the middle of it. Figure 49 shows the transition as seen by the player when entering the room. It seems like the room is closing right behind the pillar, and by doing two full turns around it the exit appears from behind it.

In reality, this effect is the result of two pairs of portals. Figure 50 shows where the two portals are located. It is noticeable that the effect works the other way around.

Multiplayer The first issue encountered while adapting the portals to the multiplayer mode was the detection of the player. In single player mode, the player was a GameObject present in the scene, as well as the camera of the player, that could be easily dragged as a public attribute of

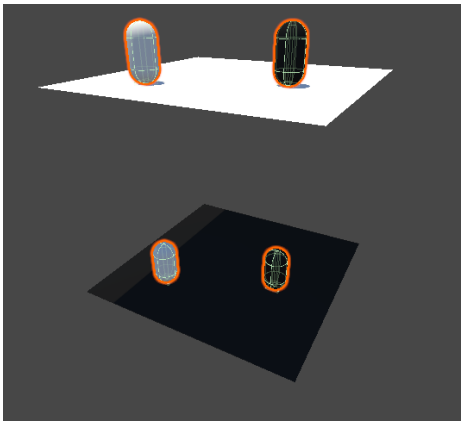

Fig. 41: Each players version of the map

every script without any difficulties. But in multiplayer mode, things are different. Players are not present in the scene, but are instantiated from a prefab as soon as the scene is loaded. Every script were rewritten in order to detect the player using a search by tag as soon as they appear in the scene, and players' cameras are automatically detected from the structure of the players' prefab.

And there comes the main issue : What if there are two players ? As both players have a different position, we would need a specific image for each player to be drawn on the portal. But the portal cannot have two different images rendered at the same time.

2.3.2 World duplication

One solution could have been to choose which image to display depending on whether the game is launched as the host or not. But the trick that was actually used is totally different. Nearly the entire map is automatically duplicated as soon as the scene loads using a C# script, and each player is playing on its own copy of the world. Portals do only work for one specific player, but never ever two players will be able to see the same portal at the same time, so the previous issue is solved !

One issue of duplicating the entire world and separating the two players is that if the two players

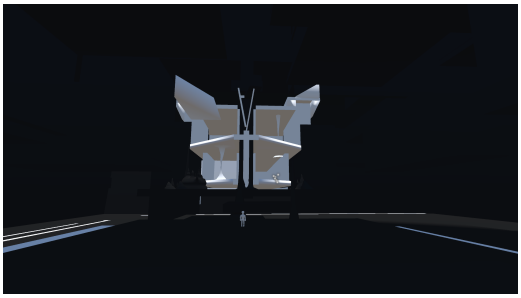

Fig. 42: Black's world

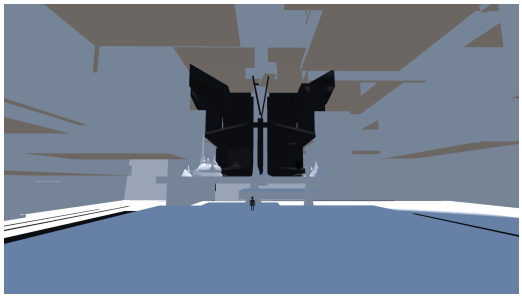

Fig. 43: White's world

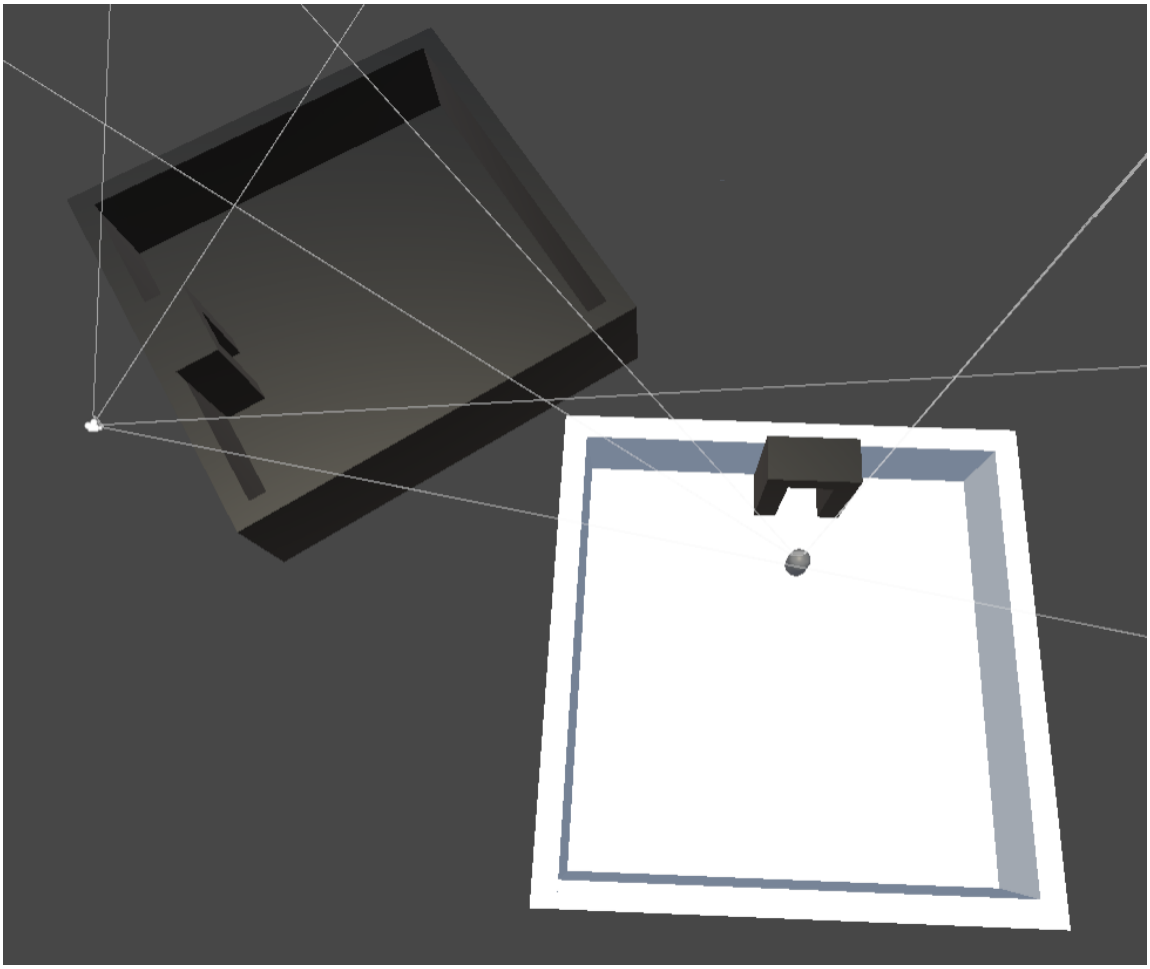

Fig. 44: Example of a simple use of a pair of portals

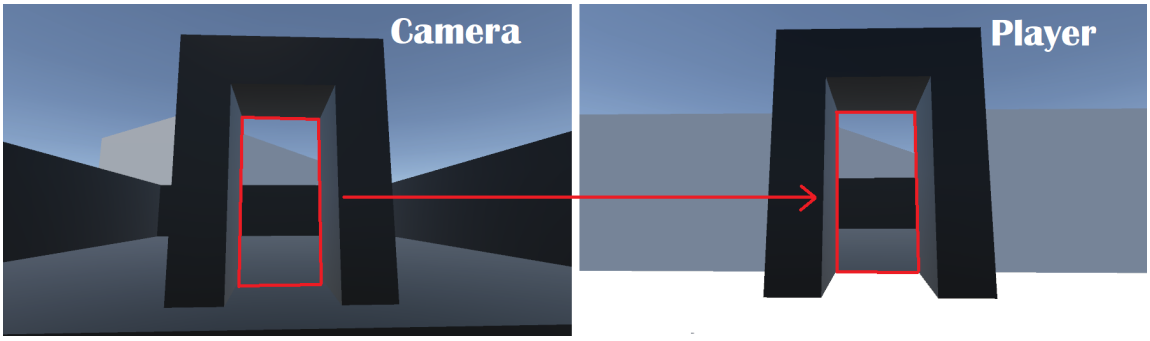

Fig. 45: Functioning of the render plane of the portal

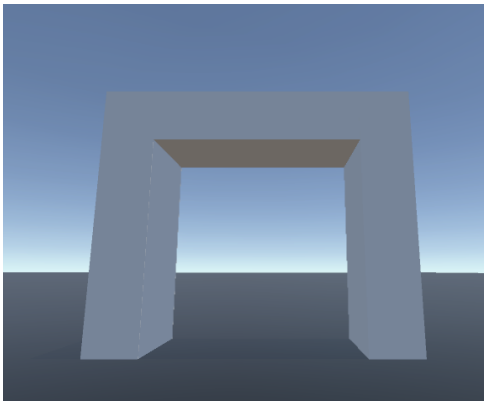

Fig. 46: How the tunnel looks like from the inside

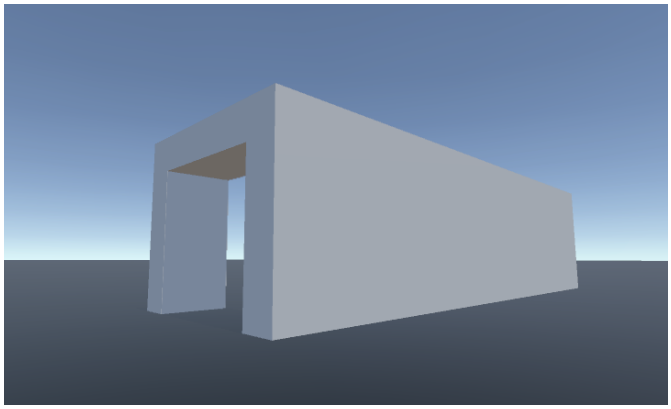

Fig. 47: How the tunnel looks like from the outside

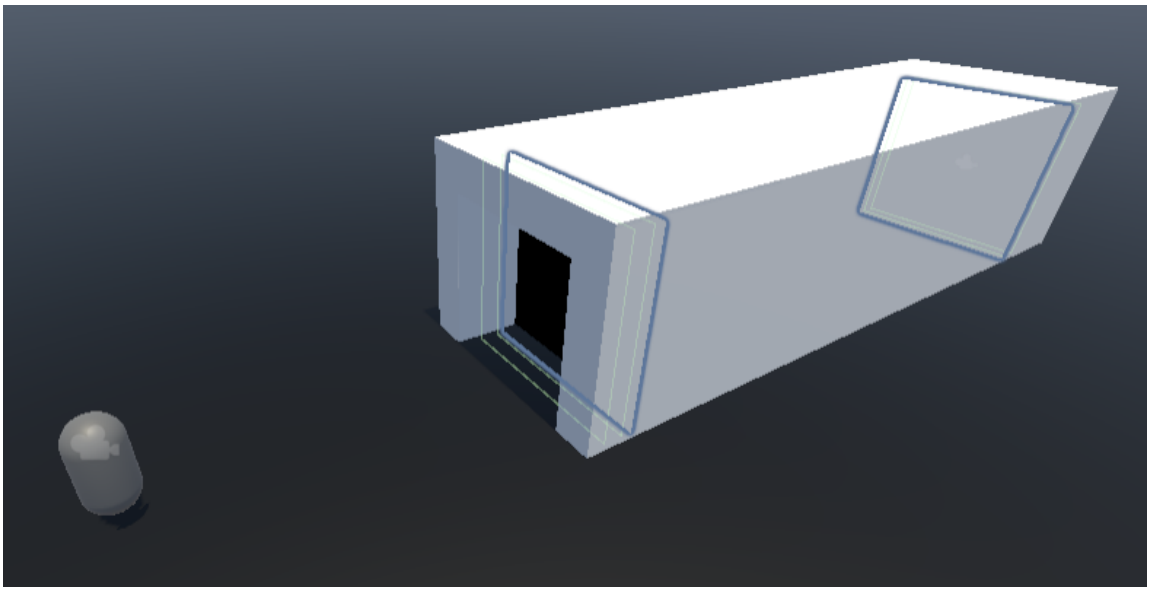

Fig. 48: How the portals are placed

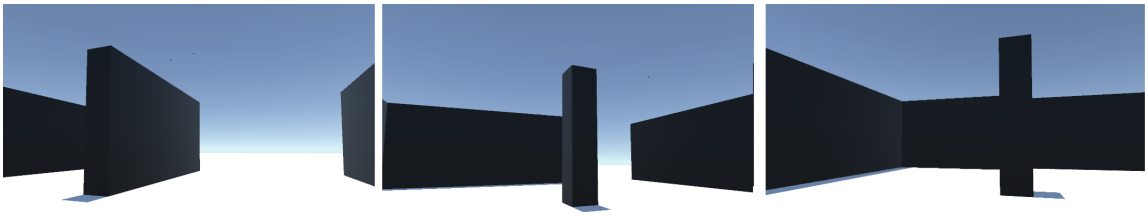

Fig. 49: The transition the player experiences when entering the room

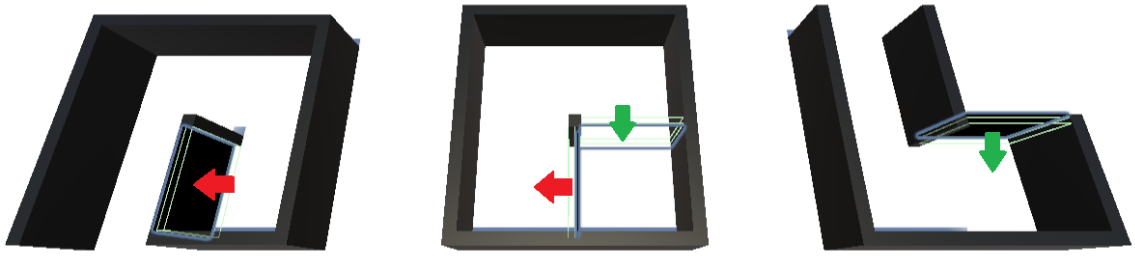

Fig. 50: How the three rooms are combined into one using portals

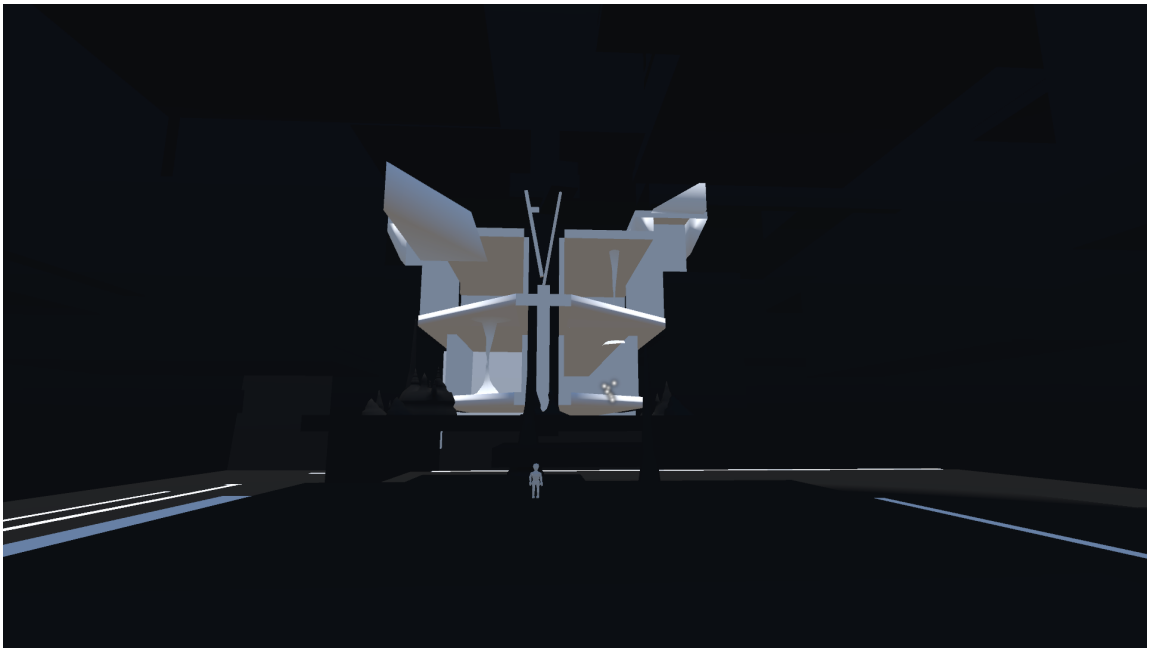

Fig. 51: The butterfly collectable

are in two completely separated worlds, they are not able to see each other directly. That is why each player has, on his own "version" of the world, a *spectrum* of the other player, which is, a visual representation of each player on the other player's part of the map. This allows each player to have a relatively good idea of where the other player is located. Figure 41 illustrates this principle. The white player is located on the top plane, and the black player is located on the bottom plane. Each player can see a spectrum of the other player on its plane, even if the player is not really there. This spectrum has no collisions and can be traversed as well as go through walls as a ghost, as long as the original character does not encounter a wall. This spectrum has been easily made by adding another GameObject (in figure 41 a capsule) as a child of the player controller, while deactivating its box collider.

What's more, this principle allows us to easily make slight modifications between the two players' maps ! For example, we have swapped every colors for one player, so black on white becomes white on black, to match the character's primary color. Figure 43 shows how the white player may see the world, whereas figure 42 shows how the black player would see the duplication of the world. Then, we can give some clues to one specific player by adding the clue on only one of the two maps. We can also base puzzles on this principle. For instance, we can have a path that is only open to one player whereas the other player has a wall, and this might be an indication to show that this path is particular, because only one of the two players will be able to see it. Thus, if both players communicate well, they will spot the difference and know that there is something special to do with the path. This is specifically what has been done for the level in subsection 2.5.2.

2.3.3 Collectable

Throughout the game, many collectables were supposed to be hidden in each room (it was expected to have five of them), if the player places himself in a specific position. It would take the form of a perfect alignment between colors and objects of the room, which will result in a particular image appearing. If both players "collect" and find every single one of these specific positions, this will result in a "Happy Ending" at the very end of the game. Figure 51 shows the first concept of the collectables implemented, namely a "butterfly", appearing when the room is seen from a specific position. Since only one level has been implemented, this is the only collectable we have decided to create. We thus preferred to not make it as a secret for now by setting the correct position as the starting position of the game.

2.4 What I would do differently

if I had to redo the project today: First of all, the portal prefab has a huge inconvenient that makes its utilisation really painful. While the prefab allows one to not have to replace every plane, it does not include the Material and the RenderTexture. Thus, in order to create a portal pair of its own, one has to create a Material, create a RenderTexture, link the camera of the portal to the RenderTexture, link the RenderTexture to the Material, Apply the material on the first portal, redo these steps for the second portal, then link the two portals together by setting the options for the scripts. Figure 52 shows the folder containing every Materials and RenderTextures for the first level.

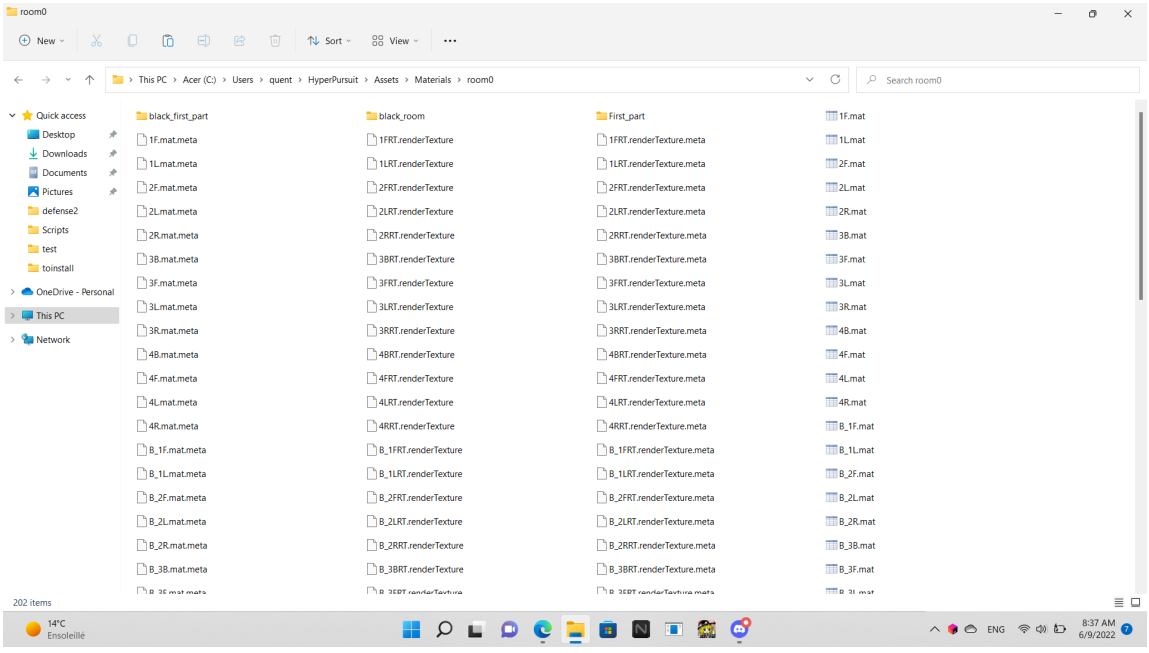

Fig. 52: A folder full of Materials and RenderTextures created by hand

2.5 Angela

This section will discuss the tasks that were assigned to Angela. It will present for each of the following subsections what was expected to be done, what was done and how, the issues encountered as well as their solutions.

Subsection 2.5.1 will explain the tests made using the portals.

Subsection 2.5.2 will explain the first level that was implemented.

2.5.1 Test Room

Firstly, a simple test using a single pair of portals on a simple plane was done just to get familiar with the concept of the portals and their components, which didn't immediately work. Some problems, that will be detailed in the coming paragraphs, were faced but were quickly fixed.

Once the teleportation was successful, our first test room was implemented. It had the form of an inverted "j" as seen in Fig. 53 and it had an exit in a separate hall. This simple "j" room was composed of three portals; A, B and C, where the portals A and B were connected to the portal C. If the player goes through the portals A or B he will be teleported in front of the portal C. Additionally, the exit hall contained a portal D, which the portal C was connected to.

The whole goal of this room was to make a simple prototype of what the upcoming levels might look like on a small scale. Here the goal for the player was to find a way to reach the exit. The player was supposed to make a choice; either going to the right or to the left. But each time, he takes one of those choices he will pop out in front of the portal C smoothly, to the point that he will be clueless that he is taking the same path over and over again. To escape from this loop, he has to do a half turn and go through the portal C to get teleported in front of the portal D and to finally reach the exit.

The first problem encountered when getting the hang of the portals was that the teleportation didn't directly work. In other words, when the player tries to go through the portal to get teleported to the expected destination, he doesn't. He just passes through it like it wasn't there but he seed the correct image he is supposed to see, which was caused by the incorrect place of the collider plane. When moving the portals, mainly a part of the portal was moved instead of all its components, leaving behind the collider plane that does the whole teleportation. Secondly, when placing two portals directly facing each other, a black rectangle appeared in the middle of the image. Those two problems were easily fixed by respectively adjusting the coordinates of the collider plane and moving one of the portals.

The first two test portals were duplicated and used to create the test room. By doing this, the teleportation would not be possible because the teleportation between two portals of the same type is not possible. The problem was fixed by changing the coordinates of portal A to match with the teleportation to portal C. When creating the test room, different halls were used, which caused some problems during the teleportation process because the player could see the difference in the width of the two halls. If I would recreate this test room, I would directly take the advice that Quentin gave me and do the halls with fixed numbers for their coordinates and scales. In addition, the brightness was modified to have the same luminosity in both the place of teleportation and the destination place.

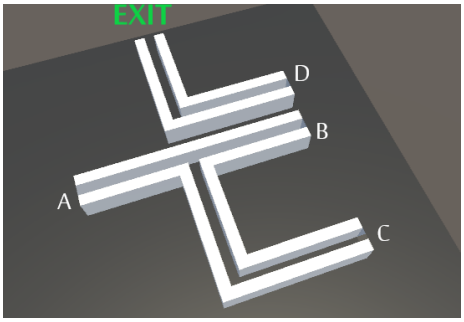

Fig. 53: Test Room

2.5.2 First Level

The first room was created representing the first chapter, the first emotion of grief: denial and the first level of the game. It was initially created in one world only; the world of the white player with all the paths.

This level is mainly composed of the collectable; the butterfly and a complex maze where the player is supposed to make the right decisions in order to go through the maze and finally retrieve the collectable.

The Butterfly The butterfly is made of two floors and each floor is divided into two parts. It is composed of eight portals: two on the first floor and four on the second one. The player initially starts in front of one of the lower parts of the butterfly.

The main goal in this level is to find the entry of the tunnels, go through the maze and finally reach the correct position to claim the collectable.

The player starts off on the first floor of the butterfly where he has no choice other than to discover and try to figure out a way to reach the entrance of the tunnels. Furthermore, each time the player takes the wrong turn or the wrong decision he will restart from the top; so he will find himself again in front of the first floor of the butterfly.

Teleportations will happen in this collectable. To be clearer, their are two main "types" of teleportations: one that will happen on both floors and one that will only happen on the second floor. On both floors, right in the middle of the part that divides the floor in two parts a teleportation will happen. When the player tries to go through from one side to another he will be teleported to the upper/lower diagonal part from where he was. For example, if the player is on the lower left part of the butterfly and he is trying to go from the left to the right, he will not realise that he is now on the right part of the second floor. And similarly for the right and left parts of both of the floors.

In addition, at the end of each right and left part of the second floor there is respectively a right and left turn. And if the player takes one of the turns, he will be teleported to the balcony of the opposite side of the floor. Let's take an example, if the player was on the upper right part of the butterfly and he takes a right turn he will be teleported cluelessly to the left balcony of the upper left part of the butterfly where there is the entrance to the tunnels. And respectively for the left turn. Similarly, if he takes the left turn of the left side he will be teleported to the right turn of the upper right side where he will reach a nice balcony to mess with him.

The Maze The maze is mainly a group of tunnels that are complexly connected together by the famous portals. As mentioned before, the player has to go through the maze to finally retrieve the collectable, but victory reigns only when both the white and the black players collect it. The portals that connect the tunnels are divided into two categories "good" portals that will help him go further in the game and reach their final destination and "bad" ones that will take them to a starting point.

Initially, one room was created in the world of the white player where the butterfly was black and the maze was white. The maze of the first white room contained all of the paths. But it was then adjusted; some paths were added, others were closed, so it can be compatible with the room of the second player.

Itinerary of the Players in the Maze Both players start off the game at the same position but in their own worlds. And they both have to take the same path to reach their goal. The final paths in the rooms are as follows.

For the white room, the player has to of course firstly enter the tunnel, and go straight forward and then take a right unlike the other player who has the options to go either straight forward, to the right or to the left as seen in both Fig 54 and Fig 55. With good communication between the players they will realise that the correct path is the common choices in their maps. The next step to take is to go straight forward as it is the only choice for the black player opposed to the choices that the white player has. By doing so, they will get teleported to a new tunnel, where at the map of the white player there is a dead-end so they don't have a choice but to do a half turn. They will consequently get teleported to a second new tunnel where the white player has the choice to take a right or left but the black player can only go straight. It can thus be concluded that the correct way to proceed is by doing another half turn which will lead to the teleportation of the players to the final tunnel.

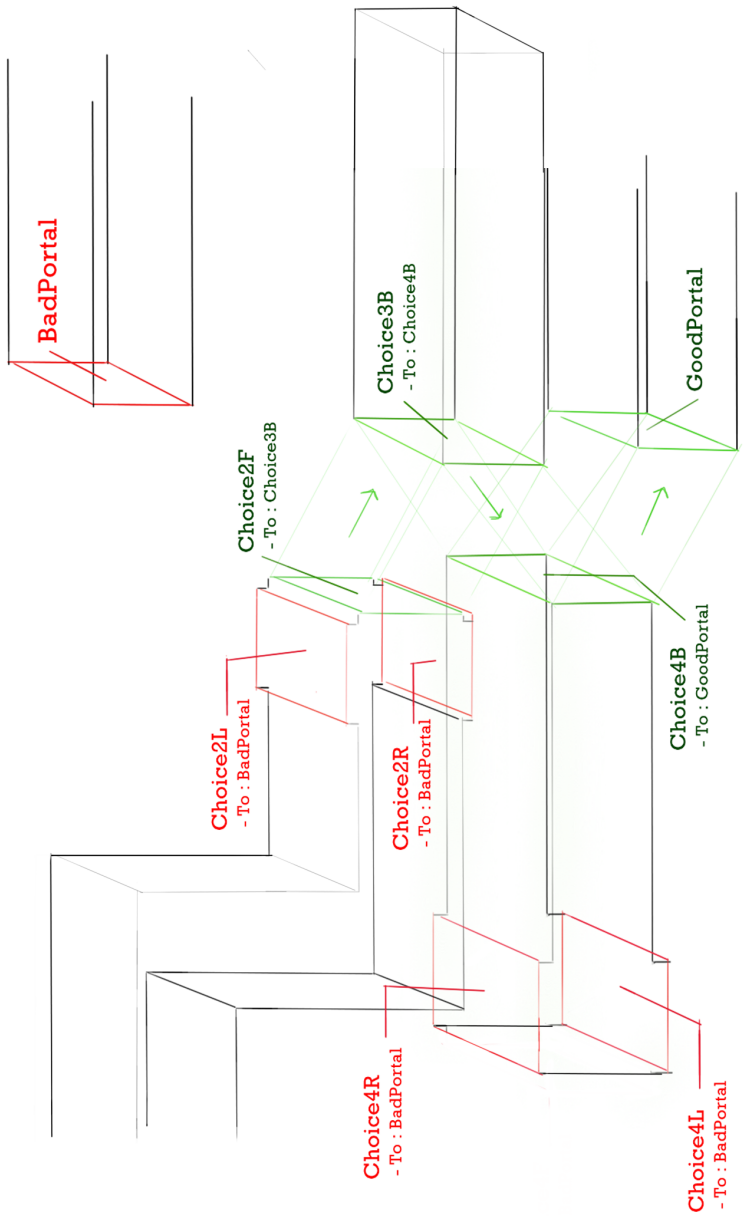

Fig. 54: White player's path in the room

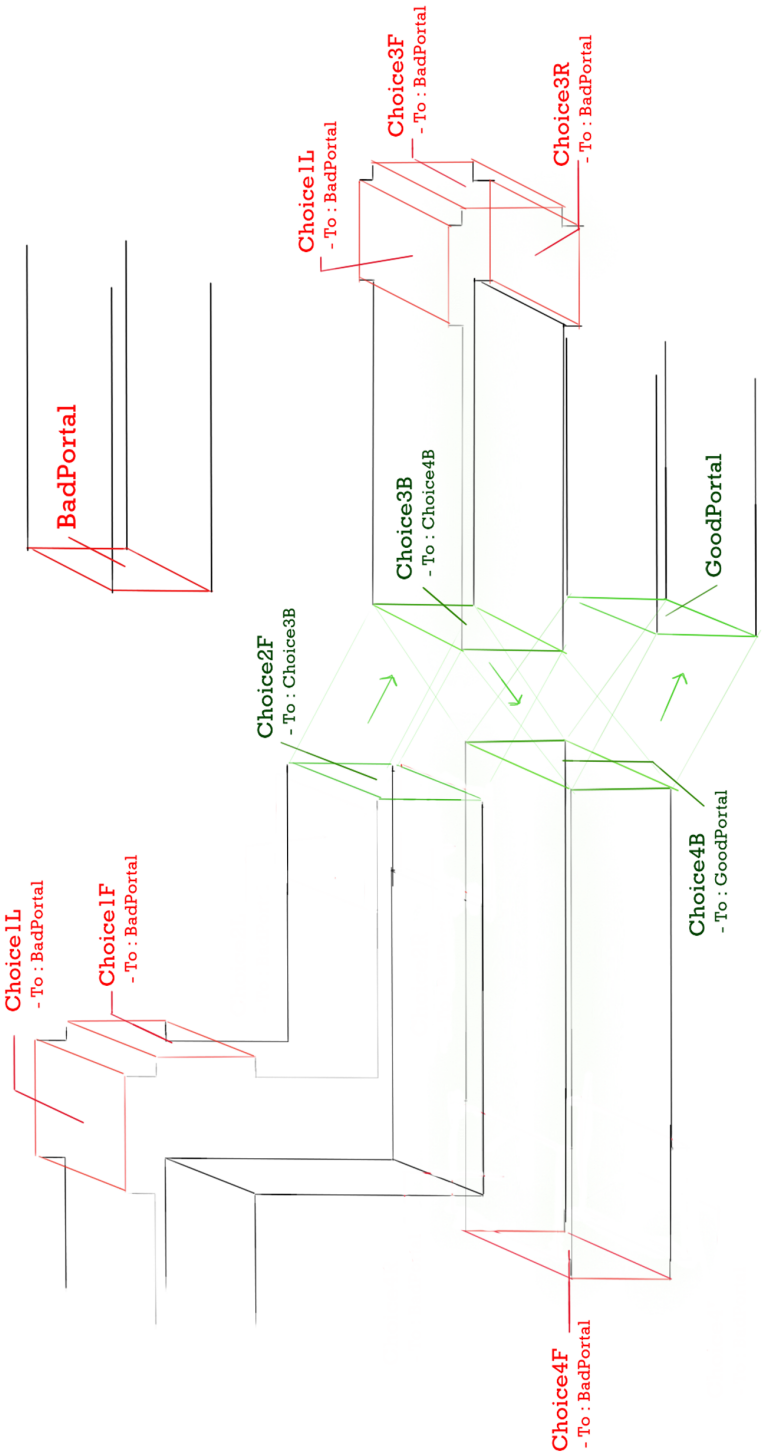

Fig. 55: Black player's path in the room

3 Conclusion

3.1 What has been achieved

Briefly, many achievements were made. Firstly, the multiplayer was finalized and works perfectly as well as the website that contains all the information to give the reader a glimpse of our group and our game. In addition, a working AI was implemented that will guide the player through the maze. Furthermore, the story was conceived, the character animations were created and implemented and some cut scenes were made: the beginning, the end and the trailer of the game. And most importantly, a level was designed, created from scratch and implemented which constitutes the biggest part of our game.

3.2 What could have been better

Many things have been achieved but we could have done more. Firstly, the game was expected to have 5 different levels but only one has been done. Secondly, the backstory could have been implemented better in the levels. In addition the rooms decorations could have been more diversified and more in cohesion with the backstory. Thirdly the some bugs are still not fixed in the final version of the game (the UI, the controller, the portals). And some things could have been done better like the website or the AI. Their original ideas were abandoned due to lack of time and abilities. Finally we should have asked for more external help. For a next project should manage our time better.