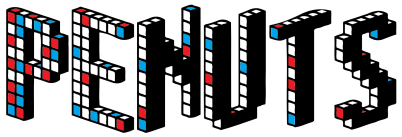
Computer Project File
SUP–2022

PEnuts
Final Report
By CAPZ

Alexandre POIRIER-COUTANSAIS
Cloé LACOMBE
Ziane LAYADI
Pierrick MADE

Friday, May 25, 2018

English Version



Contents

Introduction

This year, as freshman students in Epita we had to do a six months computer project in groups of four. This project allowed us to put into practice all knowledge acquired in lectures, tutorials and practicals but also to improve personal skills, which we have acquired for the chosen project, but which could not be put into practice during course.

Our group is named CAPZ and we have been working on a multi-player game. This game is PEnuts and his main concept is to evolve in a puzzle game with complementarity between players. The complementarity is mainly based on different visions and possibles actions of each player. Indeed, even in a same room, players are not able to see the same things or to interact with the same objects. As a puzzle game, the goal of our players is to resolve enigmas to have access to next levels or rooms. This way, as they cannot evolve in solo, they will improve their ability to communicate and solve puzzle together.

The concept and ideas of the project do not come from a particular game. In fact, many aspects of the game already exist, but we did not want to be based on a specific game. We wanted to find a new game-play involving multi player cooperation and communication. Around this root, we developed the concept of complementarity and different visions of the environment. Which we finally implemented in a puzzle-game.

This Final Report will present you in details the progress carried out since the beginning of this project. A first section will give detailed information about the concept of the game and the project in its globality. Then, a second section will give the progression and organization of this project followed by all the details for each task. In other words, there will be the distribution of the tasks, their progression over time and details about them. This section will try to answer those questions : What was expected in this part of the project and what has been achieved? How this part evolved during this project and how was it done? Finally, a last section will be a conclusion for this project, including personal feedbacks and a conclusion.

1 Type and origin of project

1.1 The Concept of the project

As explained in the introduction, the main concept of our game is a complementarity and communication between two peoples to resolve a puzzle.

To give a better view of what the game looks like, here is an example. In this example, we can see the vision of the level by each player.

Here are the details to be able to understand the sketches of the next page.


This block is movable, by being pushed by a player, a player cannot step on nor can he cross the tile the block is on.

This is a trigger bounded to the signal 2, the signal 2 is active (true) as long as a player or a movable block is on top, otherwise the signal 2 is inactive (false).


This is a void tile, a player cannot cross nor step on it.

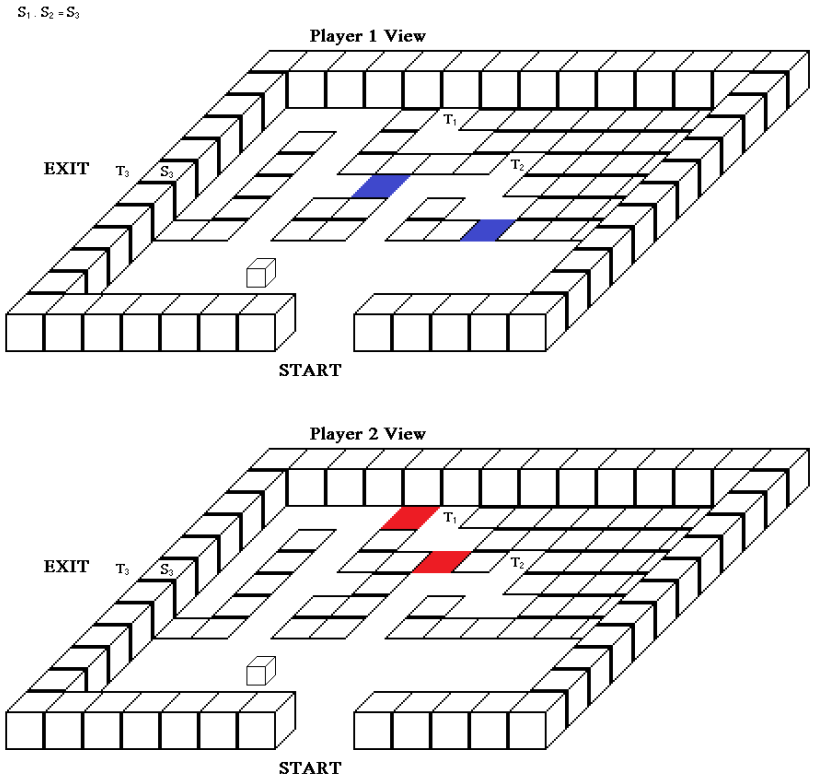

This is a block tile, a player cannot cross nor step on it.

This block tile is bounded to a signal. If the signal 3 is active (true) the block disapear and act as a standard floor tile. If the signal is inactive (false) the block act as a standard block tile.

This is a simple logic operation, here the signal 3 is active (true) if and only if both signal 1 and 2 are active (true).

The colors here represent tiles which are only seen as a standard floor tile by the player seeing the color, for the other player, it acts as an obstacle (either a block tile or a void tile).

In this case the player 2 sees the blue tiles as void tiles, however the player 1 sees them as blue floor tiles.



This level was the first concept for a typical puzzle of our game, the players have to reach the end point together. Here the Blue block is only visible by the Blue player, he is also the only one to be able to cross it, the Red block works in the same way for the Red player. Note that the movable block, represented the small block, can cross both the red and blue block. A player cannot cross where the movable block stands, however in the implementation of the movable block in our game, this is a bit different, the player can in theory push the block to the side and walk around it, but it is a lot harder to do and all levels will be able to be realized without crossing a spot where the movable block is on. Here the two players have to rely on each other to push the block around to be able to push it on the correct trigger. When a trigger is stepped on by a player or a block, it activates a signal, here since an AND gate is present, the door will only open when both trigger are stepped on, the last trigger, trigger 3, allow the player that has stepped through the door to open the door for the next player that was keeping the door open from the inside.

In order to give a better idea of what is our game, we decided to answer the main questions that can be asked for a Video Game.

What is the main concept or specification of our game ?

We wanted to create a puzzle game mainly played in a cooperative way by two players. Then, the specification is the complementarity between the different visions and actions of players.

What is the goal of the players? How can they win ?

The Ultimate goal of the players is obviously to end the game by finishing all rooms. At lower scale, the goal of players is to finish a room to evolve in the game.
But for us, a simple goal and the important goal of this game is the satisfaction to resolve a puzzle with a partner by communicating.

How can they lose ?

There is some objects able to kill players (holes and enemies). But as it is a puzzle game, players are "losing" if they do not manage to solve a room.

In which environment players evolve ? What is the type of the map?

Players mainly evolve in rooms containing simple mechanism objects and obstacles. This will be a cubic environment with very simple details.

What is their possibles actions or movements ?

Players are allowed to move in four directions, to be able to walk through the room. They are also able to interact with some objects like buttons (activating doors) and enemies. Players take damages if enemies touch them. An other restriction is that one of the player can not see those enemies. Hence, they will have special guns to kill enemies.

1.2 Origin of the project

When we started talking about the project we tried to defined what each one of us wanted. In fact, what type of game do we want, what type of actions for the user, what type of environment, and so on.

Then, we pointed out that many games are good with only one physical characteristic changing (time, space, gravity, ...). They are easy to learn but hard to master. That was one of the specifications we wanted. Afterward, searching for a new original idea and concept to implement, we started on the idea of playing with the respective vision of the players. Finally, by developing our ideas and examples we ended up on this type of project.

1.3 Object of the study

This project allowed us to improve many of our skills. As a group or individually, it gave us many useful tools. This six months project was for us one of our first real projects. Hence it was completely rewarding.

Here are the main skills that this project gave us :

- Working as a group :
    - Management
    - Team-working
    - Planning
    - Organization of a project
    - GitHub for a cooperative coding work
- Programming knowledge :
    - Unity as our game engine
    - C# for scripts in Unity
    - LaTex for documents and Reports
    - HTML/CSS for the website
    - Enhancing autonomous learning

1.4 State of the art

To think our game, we obviously had influence from all games we played in our lives, but some are more relevant, for example : *Portal 2*, *Zelda* and *Keep talking and nobody explodes*. Those example are the most relevant examples compared to our project. Therefore we used the specified strengths of those game to improve our project.

The main aspect of *Portal 2* is the same of our project, it is a puzzle-game. *Portal 2* is really popular and is recognized as the best PC game of 2011 according to Metacritic. The player have to use his mind to cross different rooms using the 'portal gun' allowing him to create 2 portals in which he can travel. Here one aspect of the real world is changed and the player has to adapt and familiarize with this change. The cooperation mode allow for two players to cooperate to solve harder puzzles, requiring both player to use their portals to get through the rooms. The strength of the cooperation mode is that it combines both the puzzles and the social interaction between the players to create a unique game-play experience.

The *Zelda* series is also a maze solving game where the player often has to complete different puzzles in Temples. This series is known by almost every gamer as one of the pillars of video-games. In this game the player has to combine different mechanics that he acquires through the game to solve puzzles and enigmas. The player often need to think out of the box (in *Zelda: phantom hourglass* on the Nintendo DS platform, the player, at one point, has to physically close the console in order to press a mold on his chart). It's strength relies on simple to difficult puzzles which forces the player to use his mind.

Keep talking and nobody explode is a cooperation game where one player has to defuse a bomb while other players have to read a manual to give him instructions. This game is popular among many player groups as it always involve a strong cooperation between players. The manual in this game is really long and exhaustive and forces other players to read it and give instructions to the defuser since he can not in a reasonable time find every specification for each modules the bomb is composed of. Each module always has one and only one combination of actions that result in the deactivation of the module. The bomb is often composed of several modules which allow the defuser to give different task to different players, requiring coordination skills. Strength : Cooperation coming from a simple game-play.

2 Presentation of CAPZ

This year, as freshman students in Epita we had to do a six months computer project in groups of four. Capz was created in these circumstances. The other point that helped the creation of the group was the motivation. Indeed, we are a group of motivated students, especially in computer science and in fact, what we love. Meaning that a chosen project should be a point of motivation for all of us. Finally we thought that we had quietly the same point of view on videos games, so we had the same expectations for this project. All of theses circumstances motivated us to create this group.

In reality it wasn't that easy. Indeed, the project groups weren't created easily. Each student had to think with who he wanted to be for a complete year working on a project... So we were all searching for the good group, the one corresponding to our expectations. The creation of our group was done really fast. The fact that, Pierrick, Alexandre and Cloé are in the same class helped its creation. Indeed, excepted Ziane who wasn't in the international class, others one knew each other. It's was finally Cloé who introduced Ziane to Pierrick and Alexandre. We were all good in programming, we were all motivated and each one of us had something to add to a project like that. That was the creation of the group.

Cloé
An overflowing imagination and energy to share

Alexandre
Motivation in everything linked to logical and computers

Pierrick
Organization and Calm in any situation

Ziane
His dream, becoming a game developer

3 Parts of the project

To be able to keep a project organized and be able to respect deadlines, the tasks division could not be neglect. Therefore, this section will present in details the different tasks of our project, their distribution and their progression over time.

3.1 Tasks Distribution

For a project of this scale - 6 months - the division of the work was really important. Indeed, without doing it, everyone would have worked on the same thing, and the project could not have evolved like that. This distribution was done at the beginning of the project taking into account our wishes and our skills. However, the issue we came across at this time was our knowledge of group projects and skills in Unity. Indeed, it meant that our distribution could not be perfect. Thereby, it's why this distribution evolved during the project depending on our skills and the necessities of the time. The following Table is here to present you the actual division that we used.

Tasks	**Member in charge**	**His substitute**
Player movements	Ziane	Alexandre
Player Actions	Ziane	Cloé
Buttons/Mechanism	Alexandre	Cloé
Camera and Lighting	Ziane	Pierrick
Vision (what the player see)	Ziane	Pierrick
Collisions	Alexandre	Cloé
Color Codes	Alexandre	Pierrick
Graphics	Cloé	Ziane
3D Animations	Alexandre	Pierrick
Level Design	Cloé	Alexandre
Sounds (effects)	Cloé	Pierrick
Website	Pierrick	All
Administration	Pierrick	Alexandre
Multi-player	Ziane	Pierrick
Network	Pierrick	Ziane
Menu and Options	Pierrick	Cloé

3.2 Programming

3.2.1 Players' Movements

In this part, the movements of the player was implemented. Hence, the aim of this part was to link the researches on the game-play we wanted to create to something pleasant for the player. We had to create controls allowing the player to move in rooms but considering the environment. Finally we decided that the better way was to allow him to have only four directions.

The implementation of this feature was done really fast as the player is a key in our game. In fact at the first presentation all the movements of the player were done.

Small changes were done over time to give a better experience to the player. Indeed, testing sessions were done to try and play with those movements. We finally decided that this was the most appropriate movements in our type of game. These are difficult to understand at the beginning but are efficient at the end.

Implementation in game :

- Four directions movements
  - [left/right]-key : rotation y-Axe of $\pm$ 90 °
  - [up/down]-key : moving [forward/backward]
- A jump is available for us if we need it as a cheat code.

Researches :

To do those movements, researches were made on the movement of 3D-Body and on the simulation of gravity in a 3D space. Those researches at the beginning of the project helped us to understand how rigid-bodies are working. Combined with those researches, we also had to understand how to set the position of an object and how to give a movement vector to it. Finally to give those special movements, we used the rotation function. The movement available, the last step was to link our movements to keys from the keyboard. Hence, there were researches to discover how to implement a controller settings. Indeed the goal was to allow the player to choose his own movements/actions' keys. But due to a lack of time we will see if we managed to finish it for this last presentation.

3.2.2 Players' Actions

The aim in this part was to create all the actions of the players in game. Our player is allowed to interact with some objects. In game, the player can grab/push a cube, he can also press buttons, shoot enemies, walk on his colors' blocks or walk through the finish portal.

This part is mainly linked with the game-play and the puzzle opportunities we wanted to add. Indeed, the implementation of new things in this part was done as the levels' creation evolved. The part on Level Design can be seen at page 21. Also, some functionalities were needed, as finish portals for example.

Implemented :

- Movable Blocks : The player is able to push some blocks. Those are special blocks allowing the player to put them on toggle buttons for example.
- Walkable blocks : Players can walk on some blocks only available for them, those are colored blocks. There description can be find in the vision's part below at page 19.
- Shoots : Player is able to shoot tagging bullets to tag the enemies, the other one can only see and kill them with his own bullets when they are tagged. For more details about the enemies, you can go at page 16.
- Finish portals: To finish a room, a player need to walk through a portal positioned at the end of the level.

Movable Blocks :

We needed a block that the player could move to solve certain puzzle. This block had to move in the same direction as the player and not go in some random directions. This first restriction was done for the first presentation, but the block did not work so well. In fact, sometimes even thought the player was pushing it, the block was not moving and this was a serious issue because without those blocks perfectly working the puzzle couldn't be resolved. The aim was to fix this issue for the second presentation. We understood that we should raise the block while pushing it because there was a collider problem between the block and the floor. So what we did was

modifying the script of the block. Now, when the player touches the block, the block raise a little bit and its 'y' axe is blocked so it doesn't fall down. This axes is only released when the player stop pushing the block. In other words, when the player touch the block, the block is "flying" as if the player was grabbing it, then when the player stand back the block goes back to a normal gravity. After some other modifications on the multi-player it was nearly finished. The last problem we had with this block was that when falling, if the hole in the floor was of the size of one cube the block got blocked falling. So the only thing we did was to block the rotations axis so the block could not fall rotating and would only fall straight and not be blocked. Finally, after all this evolution our block is completely functional.

Shoots:

To create those bullets and those gun, we mainly used colliders and Mesh-filter allowing to touch or make disappear the enemy. To initialize a bullet, we created a gameobject which allow the spawn of a bullet then we applied a special vector to it. This game object placed in front of the player allows him to shot. Then, when a bullet touches an object, actions are applied depending on this object. The last feature is an anti-spam function, it restricts the player to a limited number of bullets in a precise lapse of time.

Detection gun : This gun allow the player to tag the enemy. When he tag the enemy, the other player is able to kill it with his killing gun.

Killing gun : This gun belongs to the player which can not see the enemy. It allows him to kill a tagged enemy.

Finish Portal:

To give the player an opportunity to finish a room, we implemented finish portals. A portal is just a special blocks with a collider on it. It also has particles around to put it forward. So the collider is linked to a script which will activate the finish scene when the player touch it. Hence, when a player will touch this portal the team will be teleported to the finish scene.

3.2.3 Buttons & Mechanism

In this section we configured and implemented the different interactions of buttons (or switches) with the map.

This part really evolved linked to the game-play & the type of puzzle mechanics that we wanted to use. This part was really hard in term of thinking, indeed we had to search for new mechanisms but linked to our actual game-play.

Created Mechanisms:

- Signal Handler Class
- Pressure Plate
- Doors
- AND Gate
- NOT Gate
- One Way Signal (laser)
- Order buttons signal

Signal Handler Class :

The signal Handler is a special prefab that doesn't appear on the player screen, however it has a really important role in the mechanism clockwork. This script handles a dictionary of signals label strings and their current state, note that the state of a signal is not a boolean, it's an integer, however, from outside the signal handler script, it is perceived as a boolean through the getSignal method. When the integer corresponding to the signal string is over 0, the signal is perceived as True, False otherwise. When a mechanism request to set a signal to True, the integer value is incremented. It is decremented when a False signal is requested. This allow multiple sources to set a signal to True without overwriting each other, thus, the signal handler acts like an or gate; when at least one source has set the signal to True, the signal is True, even if another source set it to False.

Pressure Plate :

The Pressure Plate is a plate that when a player or a block steps on it, sends a True signal to the signal handler under the signal label it is specified. When the player or block is not longer on top of it, it resets, sending a False signal to the signal handler. Since the two build-in methods OnTriggerEnter and OnTriggerExit where a bit imprecise (at least to the player perspective, they would often not Trigger when it appear they should), we used the OnTriggerStay method, when this method is called, it sets a private attribute "timer" of the door to 10, when the timer changes from 0 or 1 to 10 in the Update method, the scripts sends a True signal to the signal handler, when the timer reaches 1 in the Update method, a False signal is sent to the signal handler, each time the Update method is called, the timer is decreased by one if it is over 0. Since the OnTriggerStay method is called at the same rate (or close to it, when it acts a bit against what we could expect), the timer stay close to 10 as long as a player or a block is on it. When the player or the block goes away, the timer decreases and reach 1, send a False signal to the signal handler, and then reach 0. The animation is just the pressure plate being shifted downwards by a small amount.

Doors :

The Door is a regular block that gets deactivated when it's attributed signal is set to True, It is reactivated when the signal is False. Note that this block can also be used as a floor piece, allowing a cleaner level.

AND/NOT Gate :

The AND gate is a gate that checks for two signals in the signal handler and outputs the "and" operation of the two signals in the signal handler.
Therefore, the NOT gate is a gate that checks for one signal in the signal handler and outputs the "not" operation the the signal in the signal handler.

One Way Signal (laser) :

The pressure plate wasn't really intuitive for the laser's mechanic, the goal was to lock the pressure plate once it was pressed, but since the player would expect it to come back up, it could often lead to misunderstandings of the puzzle. By introducing a new feature, the player can better understand the puzzle. The laser is basically a pressure plate that stays locked in the same position when it's triggered. The Laser only sends a True signal when the

OnTriggerStay method is called for the first time, then stays is it's position regardless of any interactions. The animation is the deactivation of the laser beams. (This laser can be seen at page 23)

Order buttons signal :

We did a system that handled the order buttons were triggered, so we could use this as an enigma in some level. It is a brand new system. We had the signals before but we didn't used them at all for this, instead, an array was used; each button is at its position in the line of buttons. A counter to check at which position in the array we are. Each time a button is triggered, the signal handler check if its the right button by verifying if the button at place i (the counter) in the button array is the same than the button that was just triggered. If it is, we add one to the counter. Else, we do nothing. When the counter is at the same number as the number of element in the array, the door is destroyed. To use this system, we had a puzzle idea. This idea was having two room, each accessible by only one player (a blue room and a red one for example). In one room, the player enter, and, when entering, a text showing a number is displayed in the room. To do this, what we did was creating a 3D text and we put a collider in it. Basically, the mesh renderer is set as off, so the text is not showing. But, each time the player enter the collider, the mesh renderer is set as "on" and each time the player get out of the collider, the mesh renderer is set as "off". To detect only the player and no other gameobject, what we did is that we checked to fact that the gameobject had the gametag "player" when anything enter the collider. In other words, the text does not display when anything else than a player enter the room. To do this, what we did is that we created a signal handler with an array, in which we put the buttons in the right order. Each time a button is pushed, the signal checks if it is the right one or not. If not, nothing happens. If it is, then a counter is incremented and the signal is awaiting for the next button. For now, both theses system perfectly work. It is going to be implemented in one level we made and maybe some others.

3.2.4 Enemies & AI

In our game we chose to create enemies with mechanics linked to our game's specification : the vision. At the beginning we didn't thought that adding enemies could be a good idea. We were saying that it would completely change our way of playing. But indeed, it was necessary to change a bit our way of playing. Without enemies the player could stay without moving while he was solving the level. In this way, we created enemies to add something "moving" to our game.

Created :

Enemies following the approaching player were created. More specifically, when a player enter into the selected range of the enemy, the enemy start to focus him and follow him while he still is in his range. Those enemies are able to find there way through the room even if there are holes, he will take the shortest way to reach his target. Moreover, when there is no more target, the enemy come back to his starting place. Finally, when tracking a player a small animation appears on the enemy, to alert the player that he is in danger.

Detection of the player :

The monster detect a player once the later enters the detection radius. This enemy will focus the closest player in his range. Hence, if another player comes closer, the enemy will change its target. In fact to do this, a function calculating the distance of every player was done, then we just needed to add a condition for the range of its vision. To find these players we used a function called "FindGameObjectWithTag(string tag)" this function allow us find the list of existing players. Once a player is found, this monster start to move.

Movement of the Enemy :

Those enemies aren't flying, thus they need to find their way through the obstacles to reach the player. Enemies use a control script utilizing a NavMesh to move around the map. To implement this feature, many researches on AI were made to finally find this one. This method of NavMesh allow us to move an object automatically on a given path. Once a player is found, a backtracking algorithm is activated to determine the better path to reach him (all of that while avoiding holes and walls). Finally this enemy come back to its starting position when there is no more players in his range.

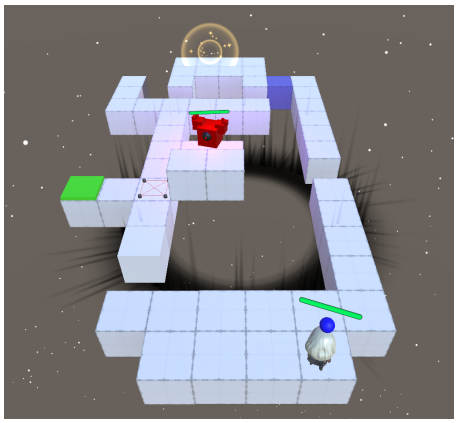
Damages :

Now that the enemy can reach the player, something must happen. The monster will chase the player across the map. If it gets close enough the player takes damage. To kill the player, the enemy can inflict damages by staying close to him or can simply eject him from the map.
We did not want the player to be one-shotted by the enemy. Hence, we created a life-bar for both, the player and the enemy. This life-bar is standing just above their head. These bar can be seen in the image below. There were also some work done on them to make sure they are always in front of the camera.

Evolution of the concept :

As said in the introduction of this section, we chose to implement enemies only at the second presentation. It was not for their difficult implementation, it was mainly for our game-play. Indeed, we did not want them to change completely our way of playing. Enemies can be really hard to work with in level designing to keep the level interesting, without putting them in as if they don't belong here. Hence, there were a work to do to integrate them well. Finally we decided to play a bit with those enemies and especially with their specifications : one player is able to tag the monster with his gun where the other one is able to see and kill it only once it has been tagged. Indeed, we tried to use those enemies as a new puzzle-mechanism even if it was really hard.

In the following picture, we can see the enemy in red with its life-bar above him.



3.2.5 Camera and Lighting

This section is here to give the player a better experience of our game. It's exactly the same aim as for graphical and audio sections. Indeed, if the game is not pleasant in terms of beauty you will play less to it than a pleasant one. Hence, it's why some details like that have a whole section in this report. Moreover, a good implementation of the camera really help the player to understand the environment and adapt himself to it.

Evolution of the Camera :

At the beginning of the project we thought that a first person camera could be a good idea to immerse the player into his character.

Finally, we understood that only a third person view at the top would be better. The aim of this point of view for the camera is to give the player an easy overview of the map and facilitate the identification of the environment. Hence, we did not have any problem to implement this view. As the camera do not need to move, there were not many researches in this part. The main researches were to find this good point of view.

Evolution of the Lighting :

The Lighting part is mainly for a most pleasant aspect. Hence, this part evolved during the creation of the game. In fact this part was something we did each time a level was finished. We adapted the lighting to the camera and to the level. For its creation we used basics lights' tools of Unity. You can see screen-shots of our levels at page 21.

3.2.6 Vision

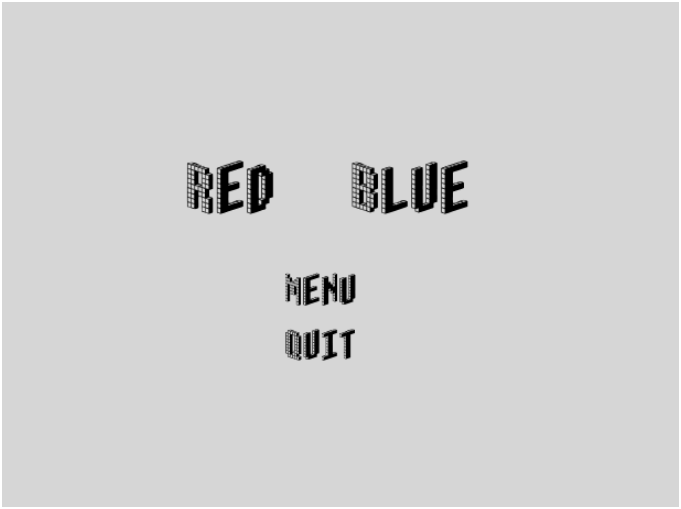
Specific to our project, it is the part where the differentiation between the two players in term of vision appears. Indeed, if we want the two players to have a differentiation in the game we had to do a menu at the beginning allowing them to choose their color. Then, a script will be called to load the map differently in terms of the chosen color.

Menu :

A complete menu linked to the other menus of the game was created. This menu was done with the same design specification used in the main menu (see page 28). In this menu have access to two buttons allowing him to select the color he wants. He can also go back to the main menu or exit the game. When the player chooses a color it loads the map with the good features.

Vision Selection :

When the player select the color, a script is called. This script load the map selected just before with the good features. Those features are : the visible blocks (in terms of the selected color), the enemy (visible or not), the gun of the player (tag or kill) and his spawn point.



3.2.7 Collisions

Collisions are the interactions between the player and his environment (object, holes, shoots and so on). In function of an object characteristics, the player will interact differently with it. Those collisions are really important in game, if they are not well done, issues and bugs can happen easily. It's why this part has to evolve with the game all the time. To avoid issues due to collisions, the work was done with them each time.

Creation : A big part of collisions were created at the same time of objects themselves (thanks to Colliders and rigidBody). So the researches were made directly for the creation itself.

Kill Zone : A kill zone was created to detect a fall of something. Without something like that, a player falling would never die, it would be a bit annoying. Hence, when something falls, it is detected. When a block is detected, it re-spawns at its original position. However, when a player falls, the level will restart completely.

3.2.8 Level Design

This part was a major part of our project. It was in this section that we considered the creation of new type of puzzle, using mechanisms, visions and in fact the complementarity of the players.

Introduction & Evolution :

Level Design can appear simple at first glance, however, creating a level that feels finished and polished is a bit more time consuming. Indeed, the more time you spend on a single level, the more refined it will appear. Thus, we spend a lot of time on details in order to hide the solution without leading the player in false paths with unwanted details. The way we usually think about it is that we start with a basic puzzle mechanic we want to play with, then we evolve around it, checking for loopholes and adding elements around it. As the puzzle grow we start compressing it to make it harder to solve, then we redo a pass of loopholes checking and potential element addition. When finishing a puzzle, we try to get ourself in different player's mind, a player that has a hard time understanding and goes for brute-force (trying many different approaches to the solution) and a player that tries to find the quickest way through, maybe even trying to find unwanted solutions. If the puzzle isn't as good as we expected it to be, we restart from the basic puzzle element we wanted to evolve around. We never start creating a puzzle directly in Unity's scene creator, we always start on paper or on a big veleda board, since we often have to redesign parts of the puzzle to make it look better and harder.

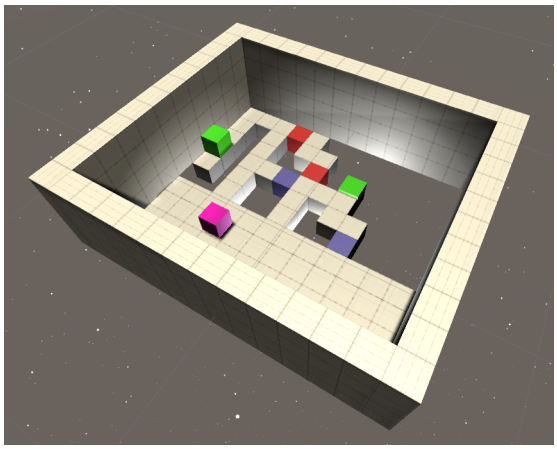
Tutorials :

We implemented a few levels and some tutorials, the tutorials are only simpler levels which are meant to introduce the players to a new mechanic or a way to solve a puzzle. For example the first tutorial consist of a simple line with a door and two pressure plates on either sides of the door, one player has to step on the first pressure plate and the second one goes through to opened door to the second plate allowing the first player to get through. The players will then understand the way the pressure plates and the door works. In the second tutorial there are two pressure plates in front of the door and a movable block, one player cannot by himself allow the second player to reach the other side of the door, he has to push the block on the second pressure plate to be able to press both pressure plates at the same time, opening the door for the second player. Then the second player can maintain the door

open with another pressure plate on the other side to allow the first player to come through. The block mechanics is then understood. The third tutorial introduce blocks of color, as before one door separate the players from the end of the puzzle, two pressure plates stand before the door and one after the door. The two first pressure plates are preceded by a color block, one pressure plate is after a blue block and one after a red block. A movable block is also present. As in the second tutorial, both pressure plates have to be pressed to open the door. Thus one player has to push the block over one of the color boxes (The one he sees) and put it on the pressure plate behind it. The other player can then stand on the other pressure plate, opening the door, and as in the same first two tutorials the player that gets through can maintain the door open for the player behind. This will allow them to understand the difference in the vision between the two players.The fourth tutorial is just two lines linked to the start and end of the map, each one of them have either a blue block, either a red block in the beginning, a laser and an inversed door (a door and a not gate). When a player reach a laser, the door is closed, preventing the player from coming back, the two players can both reach the end point through their respective line (blue and red), there is no difficulty in solving this one, but the player learn how the laser mechanics works.

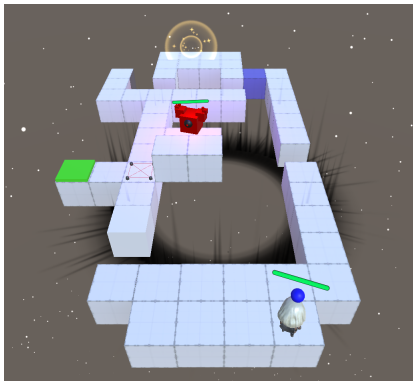
Level 1 :

The level 1 is the same level as the first one in the example page 4, tweaked a bit according to our current level design.



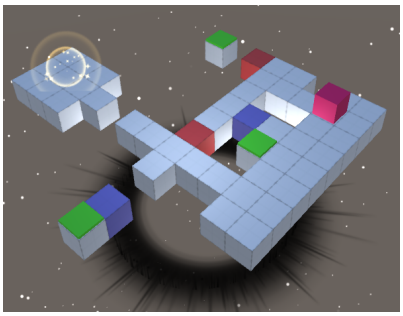
Level 2 :

The level 2 as shown in the image below, is about the enemy mechanic, basically, one player has to tag the enemy by positioning himself in a precise spot, then the other player can advance, kill the enemy and open up the way for the second player. The laser prevent one player to bring back the enemy in a spot the other player can tag it.



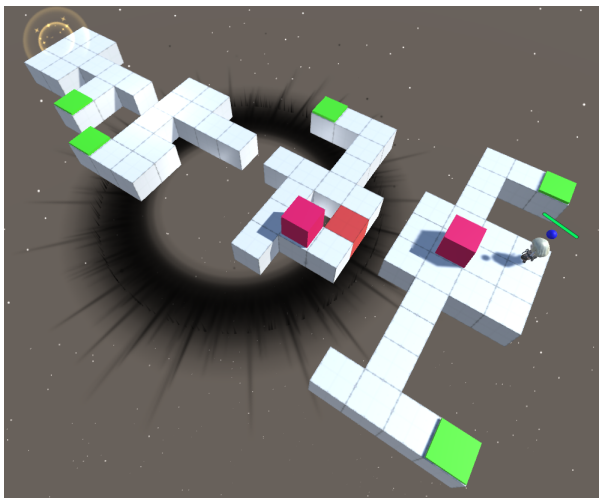
Level 3 :

The level 3 as shown in the image below is about tricking the player in many false solutions, they initially believe that many solutions are possible, but finally get stuck at one point and have to explore a different one. There are 3 pressure plates, two allow access to another pressure plate and one open the final door. a movable block is present and can be placed in all 3 pressure plates, however players have to act together in order to place the block on the correct one.



Level 4 :

The level 4 as shown in the image below is composed of 4 parts, one starting position with a movable block and some pressure plates, a first middle part with a movable block and a pressure plate, a second middle part with a pressure plate, and the end part with a pressure plate. These parts are separated by doors that blocks the player from traveling freely between them. The solution here is a bit complex since the players have to organize to press the pressure plates and to move the block around. placing the block on some pressure plates require both players to handle the block, for example in the starting part, a player has to be placed in one spot, another then push the block in front of him, and then he can push the block on the pressure plate.



Level 5 :

For the last level, we wanted something composed of several mechanics and puzzles, so it would be a little bit more complex than the other levels and stand for a last one. We used three puzzles for this level:

First, we used the usual pressure plate trigger puzzles.

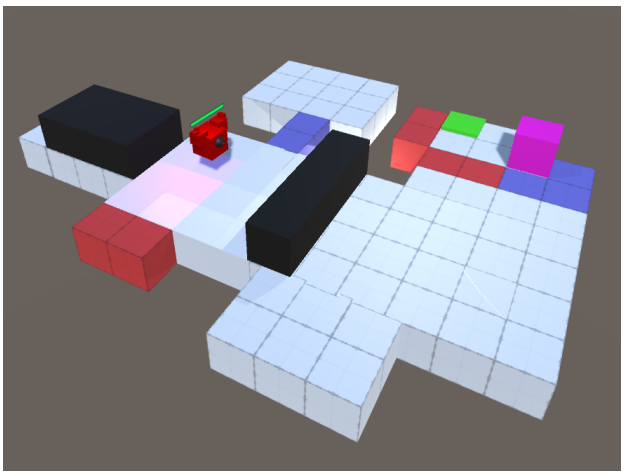
For the second puzzle, we had a new puzzle idea. This idea was having two room, each accessible by only one player (a blue room and a red one for example). In one room, the player enter, and, when entering, a text showing a number is displayed in the room. Finally, we used the enemy.

Using all these puzzles allowed us to create a level where the two players needed to solve a puzzle to access the next one: here, they first need to solve a puzzle with the pressure plates to access the enemy, and then they need to defeat the enemy to access the last pressure plate puzzle, and solving it allow the players to access the ordered buttons trigger. The main difficulty with this level design was the same difficulty we had with every level, that is to say we needed to create a level that was interesting to play, and that combined being difficult but not impossible, with tricky enigmas but not to long to understand because if they were the player would just give up. Apart of this basic level design difficulty, we also faced some difficulties with this particular level. We faced three main problems:

First, everything we needed was not systematically working: for example, the member doing this level in unity did not know how to use every part she did not do, so she had to debug a little bit the level and understand how to use everything.

The main problem was about the enemy. The nashville was not working and the enemy did not detect the player in this level, which was a huge problem because the level really needed its enemy.

Finally, the difficulty of this level was the camera. Since one player is teleported away but not the other, we needed to create a script for the camera that would detect when the player is teleported and follow only this one.



3.2.9 Multi-player

As the aim of our project is to create a game based on the players complementarity, the multi-player had to be implemented. This part contained all the creation of the multi-player structure. This part has been done really fast for the first presentation. Indeed, as the rest of the game is based on this, we could not do it at the end.

Progression over time :

In fact, after the first presentation we did not touch again the base of the Multi-player implementation. But we had to go along with it each time we were implementing something new (as we had to linked it to the network manager). Hence this part continued to have researches done during all the semester to avoid bugs.

Creation :

For the creation of the multi-player we used the network manager contained into unity. The Network Manager is a component managing the networking aspects of a multi-player game. Here is a list of the Network Manager features we used :

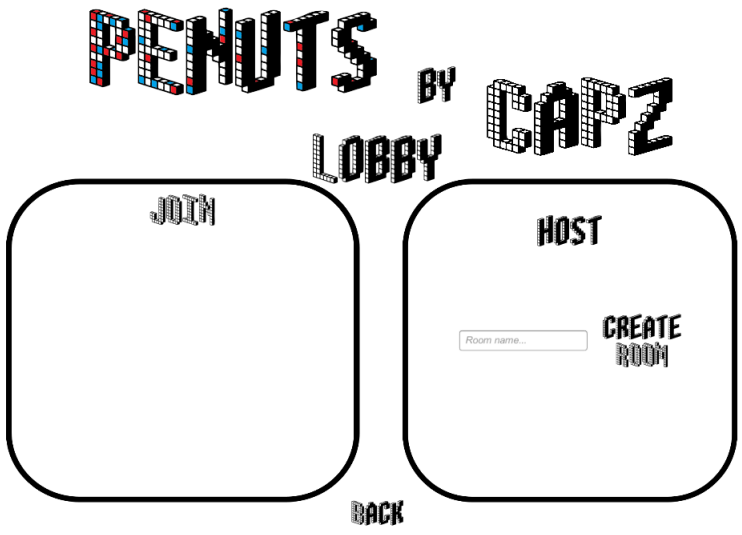
- Game state management : A Networking multi-player game can run in three modes - as a client, as a dedicated server, or as a "Host" which is both a client and a server at the same time.
- Multi-player Service : It was used to publish our game and allow everyone to create a room (Host & Client) or Join a room (Client).
- Spawn management : We used it to manage the spawning (networked instantiation) of networked GameObjects from Prefabs. In other words, we used it to make our players and our objects appear on the map.
- Scene management : Most games have more than one Scene. Indeed, there are a starting menu Scene in addition to the Scene where the game is actually played. The Network Manager is designed to automatically manage Scene state and Scene transitions in a way that works for a multi-player game.
- Matchmaking : Linked to the Multi-player Service, it help to join or create a room.

Lobby Creation :

After the creation of the multi-player settings explained above, the game was functional. But it was not completely linked to our expectations. Indeed, the interface for the user was not pleasant to use or the scene management was not working like we wanted. To fix those problem we had to go further into this network-manager. We started the creation of a lobby Menu which allow the player to create or join a room with anyone. This menu was done with the same design used in the other menu (see page 29). In fact this menu was the hardest one to create. We had to understand completely how the network manager is working to make it functional.

Lobby Presentation :

In this Menu the user has the possibility to Join a room, Host a game or go back to the main menu.
Host : The hosting canvas was the simpler to create (not taking into account the Back Button). We created a canvas for the text input corresponding to the room's name and a button calling a script creating a room.
Join : The Join canvas had to list all available rooms or display a state text to inform the user of its connection validity.



3.2.10 Menu & Options

In this part we implemented different options and interfaces to help and give some configurations to the user. Indeed, if the player can mute the sounds effects, change the quality of the screen, use a pause menu, select his level and so on, he will have a better experience of the game.

Progression :

At the First Presentation, the menu was only a screen where the player could choose whether he wanted to play or quit. For the Second Presentation a great work was done. Indeed, working menu were created adapted to our design and linked to the Multi-player's settings. It took us a long time before the second presentation to create all of them, but it was finally done. Those menu did not really evolved for the last presentation as there was no need for it. In fact except the lobby fixed due to some issue or the level selector changed to have more scenes, nothing was modified.

Created :

- Different working menu scenes adapted to our design. It includes :
  - A Main Menu with Play, Settings, Credits & Quit buttons
  - A Lobby (Play): allowing the player to join or host a game
  - A level Selection Menu : allowing the player to select his level.
  - Color Selection Menu : where the player can select the color he wants to have.
  - Settings Menu : allowing the player to change volume settings or screen settings.
  - Credits & Finish scene : appearing when players finish a level
  - Quit & Pause : Letting the player go back to the menu or leave the game if he wants to
- Design : All those menu were aligned to the same design to give a nice experience.

Researches :

Many videos and resources were used for this part. Indeed, we had to understand clearly how the multi-player settings were working, how the quality & resolution were controlled, the volume, the scene management and so on.

Design :

Those menus were created with a simple design allowing the user to find his way easily throughout all those scenes & buttons. We also wanted the menu to be really pleasant, it's why only useful things are in it. We did it using images & TextMeshPro with a font corresponding to our game (cubic font). TextMeshPro is an asset available in unity allowing to have really beautiful text elements. This design is also specific to our environment, indeed, we wanted something really simple without big details. To see animations on buttons and so on, the better way is to play our game, you can find it on our web-page. But to have a nice overview you can just look at images presented below for each menu.

Main Menu :

This was the first menu created. It was with this one that the design was created. In fact, it was at this moment that buttons, templates for the text or the background were created. All of this being done, we understood the bases of a menu creation.

Each time a button is pressed, it sets to inactive the canvas containing the main menu and set to active an other one. For example, when the settings button is pressed, it's the settings canvas that will be activated, same for the Play button.

Finally the Quit button allowing the player to quit the game use an easy script calling Application.Quit() which quit the game.

Lobby Menu :

This menu was the hardest one to create as we had to understand really well the functionality of the network Manager. The implementation of this menu was explained in the multi-player part at page 27.

The particularity that was not explained in the multi-player part was the way buttons and state text are working. The buttons to join a room are set to be available only if a room is created, else the buttons will do nothing special. For the state's text, we used a script to change his content each time it is updated. If there is no rooms available in the list, it will say "There is no rooms for the time..." else, it will print "There is n rooms !"

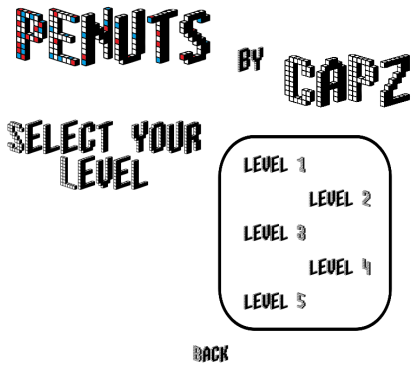
Finally the back button allowing the player to go back to the main menu is working exactly as the button to come into this menu (Set active a canvas or not).

Level Selection Menu :

This menu is here to let the player select the level he wants to play. The menu is also linked to the vision menu which allows the player to change their color at the beginning of a level.

Each level button is loading the specified scene and players are sent here. Hence, those buttons are using the Scene management functions of the network manager

Finally the back button is exactly the same as the one presented in the lobby menu. And it will be the same for each other back buttons.



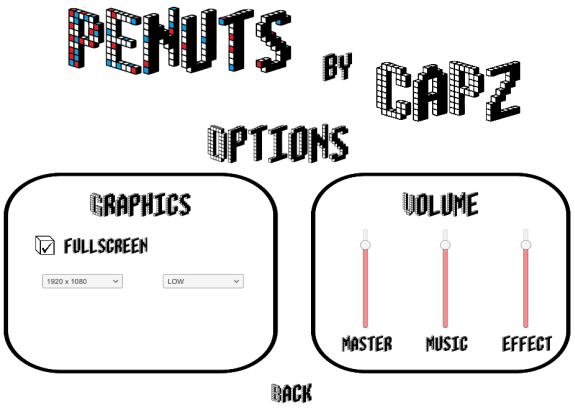
Options Menu :

This menu is really important in any game, and almost every player has to go through it once. It allows the player to change settings like the audio or graphics.

The graphics section let the player enable the fullscreen, change the resolution or change the quality. All these features were implemented with basics UI of unity including Buttons, Dropdown or Toggle. Then those UI had to be linked with scripts to control game settings. To implement it we had to understand how unity supervises graphics settings. Then, by using functions linked to graphics we could change those settings directly within our game.

The Audio Section is here to let the player change the available volumes. Indeed, there is three sliders which he can move to change the volume. The first one is here to control the whole audio of the game and the other are respectively to change the music volume and the SFX volume. To create them we used the basic functionalities of sliders linked to an audio mixer controlling our volumes. Indeed, a slider can be use to select a value in a certain range, we just had to link it thanks to a script to the volume of each audio mixer.

Finally if we have the time just before this last presentation, we will maybe add some features in this settings menu. The main idea for the time being something to change the background music or something to change the controllers of the player.



Finish & Credits Menu :

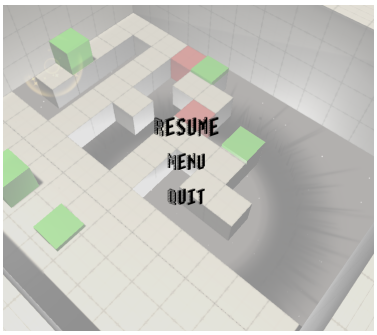
This scene will appear when players finish a level. Indeed, if one of the player enter into a finish portal, the two player are sent to this finish scene.

Buttons used in this menu are really simple. You can quit the game (Application.Quit, you can also go back to the menu (scene manager). Finally you can also click on PEnuts to access our website (using Application.OpenURL).

Pause Menu :

A pause menu is essential in a game, indeed, it allows the player to quit the game or go back to the menu even if he was in game.

For its creation we used a script linked to the player, each time the player press the escape-key the game will be paused. More specifically, the time will be stopped in game and a canvas will appear.



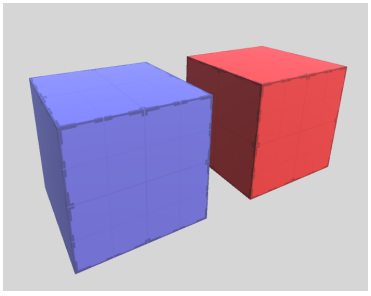
3.3 Design

3.3.1 Color Codes

We had to do some choices to put good color codes in the game. Indeed, those colors help players see something and give information. Those information can be on the type of object, on the possible interactions, and so on. Hence this part could not be neglect to give a pleasant experience to players.

The two principal colors of our game is Blue and Red, which differentiate the Blue box which is only seen by the Blue player from the Red Box which is only seen by the Red player.

We wished the player to be able to directly see the impact of a mechanism on another, for example; how does the player know which pressure plate out of two opens a specific door? To solve this problem we thought about allowing the different mechanism to have a different color, which would match either what it is activated by or what it activates. For this, we had to modify the signal handler system to account for different colors for each signal, a new method has been added which allow a mechanism to retrieve the color from a given signal. Each mechanism can treat the color differently depending on their 3D model, there are no constraints on which parts get colored or not. The colors are taken from a bank of 20 different colors, the last 10 being a bit closer to the first 10, since they are harder to differentiate, they will only be used in case the level requires more than 10 signals. If the 20 colors are all used, the game will randomly generate a new one, not matching any of the precedent colors. The issue with this is that some colors may appear really close to each other, hence why they are only used in the improbable case that a level requires more than 20 signals.



3.3.2 Graphics

The implementation of particles, polish assets and better appearance are in this part. As the color codes, better the graphics appearance are, better the comprehension and the experience of the player can be. Moreover, they are here to give some life and dynamism to the game.

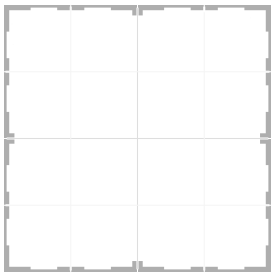
The Evolution :

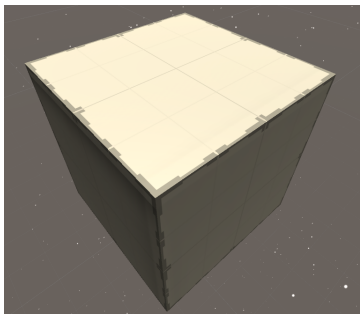
At the beginning of the project, as no one in the group was good in design, we decided that, we would not go far and base our game on our graphics. We understood that starting to do our own design would take really to much time and we preferred the code to be well done. It's why nearly all particles, models, materials and textures of this project came from the Assets Store.

The evolution of graphics was mainly done just before a presentation or a build. Indeed, as they are not dependent on the other sections in terms of coding, they were not complicated to implemented and we could go as far as we wanted. But each time those graphics were essential to have a pleasant experience. Without these, we could not distinguish a door from the finish portal or a button. Hence this part evolved in parallel with the rest mainly for the experience of the player.

The Blocks :

As our character evolves in a block's world, our texture for those blocks had to be well chosen. Finally, the Material used for our blocks was found really fast on the assets store. We took it as it was a simple and nice texture as we wanted.





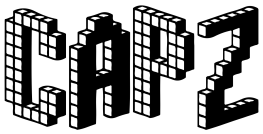
The Character :

For the character we found an asset which corresponds to our expectation in term of style. At the beginning we thought that we would change it fast. Finally it is already nice and give a better aspect to the game.



Writing Font :

Taking our own font was something essential to give more details to our environment. We finally took a font corresponding to our cubic world.



Particles :

Adding particles is a must in a game, it gives life, dynamism, and elements of comprehension. Indeed, those particles make a game more attractive and pleasant. In our game we used particles for many things :

- Background
- Finish Portal
- Enemies
- Bullets

You can see all those particles in images available at page 22. But to see them moving you will need to play our game.

3.3.3 Music

For us the music have a real impact on the player, some good music can help him immerse into the game environment. It's why we tried our best to do something good in this part of the game, even if there is not any musician in the group.

Creation :

We understood really fast, that without any musicians the creation of a music would be compromised. After some reflexion, we decided that we would directly take that music from Internet. At this moment we have been searching for a long time a good music corresponding to our expectations. Firstly we needed something with a free copyright. Then, we wanted something good for the player to let him be concentrated on the game. We found a first music just before the first presentation, but it was not really what we wanted.

After a long time without touching it, we finally decided that we could let the player chose the music he wants. It was a genius compromise, we could put some different styles of music or different tempo, then the player would just have to chose in the settings menu the one he wants.

The problem of finding those music was still here, indeed, we could now find a lot of tracks, but nothing good linked to the game. It is at this time that the idea of taking an algorithm to generate music came to us. Each generated track is a unique combination of tones, rhythm and instruments. Hence, we just had to let the algorithm find the most appropriate song for us.

The algorithm we used is called Computoser. This algorithm is currently experimental. It may generate both good and bad pieces but we could try it again as the sound did not fit to our expectations. This website was perfect, we could chose the instruments used, the tempo, the mood, and so on. Having musics in our game was now really at our scale. We finally found some good background musics for our game thanks to this algorithm.

Implementation :

The implementation of music in unity is really easy. We just used the basics given tools, and created a little script to avoid this music to be cut when the scene was changing.

3.3.4 Sounds Effects

This part, neglect in much projects, can in fact gives a real immersion to players. Hence, we will try to adapt sound effects to the environment to give a more pleasant time. We wanted a game with a good game-play, but also a game where the player could easily immerse himself. We thought we needed a good atmosphere in order to achieve our goal. This is the reason why we made a serious point out of the SFX part. We had to find the sound effect of several game objects.

Creation :

We started the implementation of those sounds effects just at the end of this project. Indeed, it was important only if our game was already well started.

Choosing the sounds effects was the longest part, because we had to find the right sounds, and they needed to be copyright free. To implement the sounds effects, there is a function in unity called "sounds effect". we just had to call it at the right moment. Thus, for example, for the signals it's when they are triggered.

All those sounds are available on our website or in the game.

Buttons & Pressure plate :

We created a sound effect for buttons and pressure plates. We put the same sound effect for those two game objects because, from our point of view, it made more sense that all "clickable" part of the game had the same sound when activated.

The Enemy :

For the enemy we added sounds for all its actions.

Detection Sound : When he will detect a player, in addition to the particles a sound is alarming the player.

Damage Sounds : When damages are done to the player a quick sound is played.

Death : When an enemy die, there is also a sound effect.

The Player :

Finally, we chose to have sounds effects when the player is shooting.

3.3.5 3D Animations

As the Graphics part, animations of players and fluidity of movements can only give a better aspect to the game and the environment.

As a matter of fact, we did not do much animations implementations. It was really a little feature that we would have created with more time, but it was not the case. Indeed, the time spent for animations is to big in terms of the results. Furthermore, animations, do not really impact our game-play. We preferred focus ourselves on graphics, particles, sounds or design more than in animations.

Finally, we touched a bit to animations to change the ones from the character found on the assets store. Indeed, we did not like his walking animations so we changed that, but it was not a hard job.

The second time we touched animations was for the buttons in our Menu and a bit with the pressure plate and the laser mechanic. The rest of the time it was not really animations, it was mostly particles effects.

3.4 Management

3.4.1 Website

As the project evolved, we had to do the same with a website. In this way, a website was created soon after the beginning of the project. Then our goal was to make it evolve through the evolution of our game.

Website URL : `penuts.fr`

Pages of our website :

- Home
- Contact page
- Contribution
- A project presentation :
  - CAPZ
  - History
  - Progression
- Links for downloads :
  - The Game (allowing to choose the wanted OS for the download)
  - Reports (to show our reports & documents)
  - Other things used for the project (musics, pictures, videos,...)

Creation of the Website :

As no one in the group had skills in web coding at the beginning of the project, we decided not to recreate the wheel. Indeed, to have a real and nice website working on every platform (computers, smart-phones, tablets, ...) we needed more time. Hence, for its creation we used a bootstrap and css template to understand how it was working then we updated it linked to our expectations. Understanding how web coding was working was interesting. Indeed, we could discover all of those language by using them and create new features in our website (contact page / menu / buttons / links...). Using parts of templates was really instructive and creating a nice website was now in our skills.

Publication of the website :

The aim of this part was to send our website over the Internet and allow anyone to find it with a simple URL.

Many researches were done to understand how an Internet page is working. The main concepts we had to understand for our purpose were the hosting and the domain name. Indeed, a website need to be host somewhere connected to Internet with its data (content of pages). Then this host need to be findable on the Internet, for this, DNS (Domain Name Servers) are here to reference it with a simpler name (something like 10.27.384.28 is transformed into penuts.fr). Those understandable name are here to help users find a specific page.

- Host : We used GitHub to host our website. Git was the most appropriate hosting available. Indeed, it was free, not hard to use and allowed easy update to the website.
- DNS : We used Infomaniak to buy a domain name. It was the cheapest available to have good URL. We now own penuts.fr for two years which will help users find our project page.

By publishing it, we also had the opportunity to understand how the page referencing of google was working. Indeed, some work was done to reference our website, allowing "google's robot" to find better tags or descriptions.

Content of the website :

The structure of the website and the publication done, we could start to create some content. For that, we tried to add things as the project was evolving.

Menu : The menu is available in all our pages, it's a bar at the top of the page allowing the user to have a link to the other pages. It is responsive, it means that this bar will adapt its width to the screen. In fact all our website is responsive, it's really a pleasant feature for the user.

Home page : This page is the first one that a visitor will discover. Hence, it was the one evolving the most. It was evolving linked to our pictures, to our design and to our other pages. Indeed, this page contains links to every other pages, pictures, and a quick presentation of the project.

Contact page : Obviously this page is here if someone want to contact us. Any comments or feedbacks, a question about the project or about the game, this page is here for those situations. This page contains a PHP form allowing the visitor to send us a message directly from the website. He also have at his disposition our mails just beside.
You can visit it just here : `penuts.fr/contact`

Contribution page : This page is here to let users report an issue, a bug to fix, a dead link or something else about the website, the game or anything else. As a matter of fact, this page contains a document that everyone can fill. This document is structured to let the user understand where he needs to add something if he wants to. He can add, anything linked to our project (ideas, issues, bugs and so on). We used `framapad.org` to create this document. It is collaborative text editor, letting anyone with the link modify the document at the same time.
Give it a look and help us : `penuts.fr/contribute`

CAPZ : This is were you will find information about our group. This project could not have been done well if a member was missing. Hence, discover each one of us, members of this project's group : CAPZ.
Again, the link is just here : `penuts.fr/CAPZ`

History & Progression : As the CAPZ page, this page will describe us. In this page you can discover more about our group & our project. Indeed, behind this game there is people, more particularly students. Hence this project wasn't just done like that, it has some history behind.
Discover this history here : `penuts.fr/history`

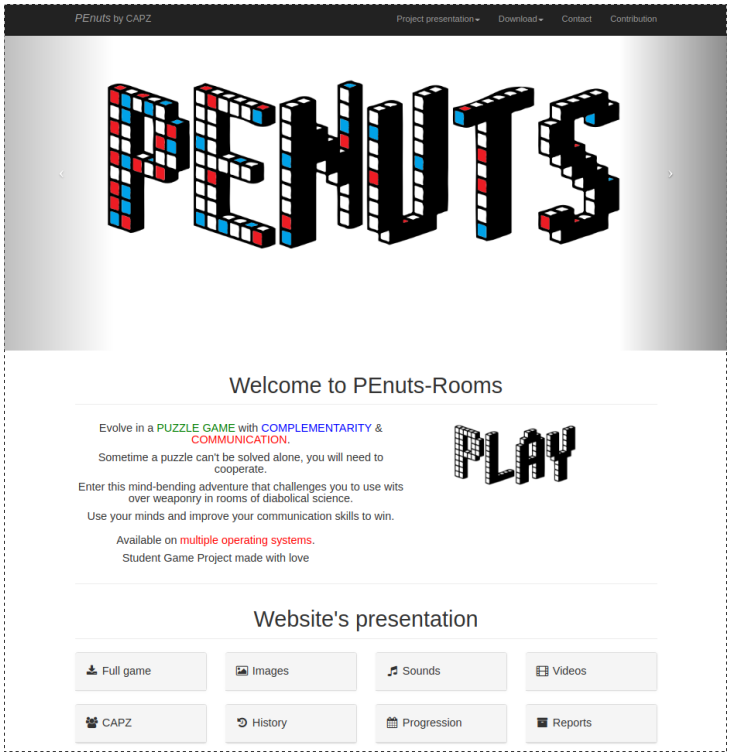
Download - The Game : This first download page is essential. As you understood, we created a game, so this page is here to download it. In its content you will find links for each Operating Systems (Windows, linux and macOS). You will also find all the versions of our game : light-version, full-version, old-versions, and so on...
If you want to play, it's just here : `penuts.fr/download-GAME`

Download - Images : Second download page, this page is clearly a feature. It allow the user to access all images of our project. In this page you can find logos, screen-shoots of the game, wallpaper and other content.
This is the link to see our gallery : `penuts.fr/images`

Download - Music : You want even more content, you can discover in this page the music and sounds-effects used for our game. You can listen to these audio directly on the page, but you can also download them and use them. For more information about sounds you had a section just above.
Listen to our game-sounds just here : `penuts.fr/musics`

Download - Videos : Exactly as the music page, you can find videos used for this project here. It's true that we didn't use videos in our game but there is videos of it to give an overview of the game-play. Also, there is links to many videos used to help us finish the game. Mainly Youtube tutorials.
See more : `penuts.fr/videos`

Download - Reports : You can download this report ! On this page you will be able to see all our reports done for the project : book of specifications, first report, second report and final report. You can read them directly on the page thanks to a pdf viewer but you can also download them.
Read our reports : `penuts.fr/reports`



3.4.2 Economic aspect

This project was a nonprofit game. We will not earn any money on this creation. The aim of this project was mainly a contribution of knowledge. We did not want to go too far in the funding of this project. But we agreed that if we had to give just few euros to have better final project we would do it. Hence, we used some for the website and maybe for the CD.

Website : Finally we decided to buy a domain name for two years (2€/person). Indeed, as explained in the website part, a nice domain name allow the users to find our page. But we also took into account that we might also want to share our website, and to do so it was easier to have a short and nice domain name.

Reports : To print all those reports we had to use a lot of papers and inks (total : 200 pages). As we did not really know how much it was, we decided to buy a breakfast for the one who was printing reports.

CD & User Manual : Arriving at the end of this project, the final presentation is coming. Actually, for this presentation we need to arrive with a Customized CD case, a CD with the game and an installation/operating manual. All the design are done and the manual is created. But we did not print them for the time. Indeed, for the time we tried to find a partnership with a printing house. Not really conclusive, we will see just before the presentation if we print everything ourselves or we if pay to have a really nice box containing all our game.

3.4.3 Resources aspect

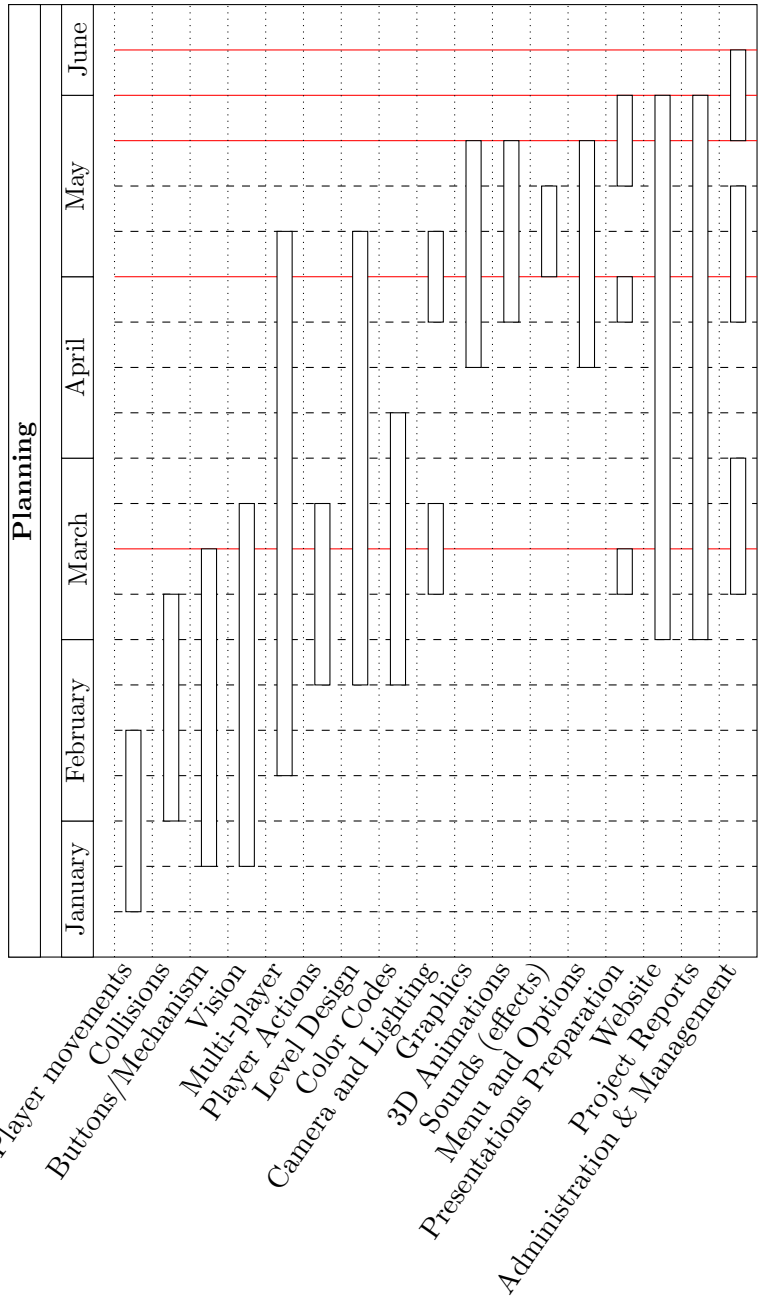
For this project we use mainly free and open-sources softwares or available tools as:

- Unity (game engine)
- Overleaf (LaTex editor)
- MonoDevelop, Rider or VisualStudio (C# editor)
- Blender (3D animation and creation)
- Assets Store (powerful design)
- Brackets (CSS and HTML Editors)
- GitHub (Coding organization & host for the website)
- GitHub Desktop / Kraken (GitHub interface)
- Audacity & BOSCA CEOIL & Computoser(Sounds editor)
- Photoshop & Paint (Logo Design)
- Inno Setup (create the windows installer)
- Internet (For documentations and everything else)

For the equipment part, we used mainly already acquired ones :

- Computers (obviously to use all tools & softwares above)
- Printer (to print reports & outlines)

3.5 Progression



4 Conclusion

4.1 Personals Conclusions

Ziane LAYADI : I learned to use unity before entering at Epita, I then improved my skills thanks to a lot a videos. In this project, I faced the group work that made me evolve in my way of setting up a project. Indeed, my skill to respect deadlines evolved and being with international comrades helped me improve my English. I really did not like the use of GitHub and I thought that the task distribution wasn't done well. But even with that, we managed to create something really good. The team work was finally useful. I really loved working on this project, in fact I to love create games in my free time as personal project. This project symbolizes for me a step for my entering in the working world in terms of group working. This experience will be really useful to me.

Cloé LACOMBE : My feelings about our group is that we worked well together. As a matter of fact, we got along well and, when working, we were together. The fact that we got along well also helped us to have a good atmosphere when working. This is one of the reasons why I liked a lot this group project. It was the first time I could work as part of a group, as we all did our part of the work. It helped me improve my group working skills, because before this project I was really bad at working with other peoples. In fact, I could not work at all with other peoples because I often don't have the same point of view on the project or the same ideas than the rest of the group. This time, we just worked together and discussed the ideas, compromising when needed. This is one of the main things this project taught me: I learned how to work as a part of a group. I grew up on many points with this project:

First of all, it learned me how to remain calm: I am very short tempered and I get on my nerves way too easily. I realized it might be a problem when one of our members stopped using github two days before a thesis, and instead of having a calm talk with him, I yelled at him, which is part of the reasons why he completely refused to try GitHub again. This learned me I needed to keep calm instead of just getting so easily on my nerves.

Secondly, I learned how to compromise. Before this project, I wanted everything without compromise and used to do projects just like I saw them instead of asking the others. This time, we just discussed everything and had all our ideas together, compromising with each other and really creating

something out of all our brains instead of just following one's orders, which learned me how to work as part of a group following each other's ideas.

This all semester, we had to work periodically because of the deadlines (the thesis). This learned me how to work for a long time and not only for one project or one week, because we kept the project a long time (the whole semester), so I learned how to plan my work to include it to my schedule. I also evolved on a technical point of view:

First of all, I learned how to use unity (the main tool we're using to create the video game). It was very useful during the project, so we all had to learn how to use it. It was very interesting learning how to use a game engine. Since I did some level design, some scripts, some puzzle I used several parts of unity and several options and I learned how to do a few things.

Second, I had to learn the basis of GitHub because it is the tool we used to work together. I thought this was very interesting because I know we will use it our our next years so I thing it is a good thing that I discovered GitHub this year and learned the real basis.

I also learned a little bit about coding doing the project. I learned how to track bug down and try to fix them, and I improved my skill by working. I did a few features, like a moving block and a system of signals you have to trigger in the right order. I'm happy with what I did, because I had to create the idea of how it would work and I think this was the most interesting part.

Finally, I really liked this project. I liked how we begun from an idea and an empty window on our computers to finally have a working game, and I also really like my project group. I learned a lot of things, on human and technical points, and I also really like our game which is probably one of the main reasons I am that happy with what we did.

Alexandre POIRIER-COUTANSAIS : This project was my first "complex" video-game, though since I don't enjoy video-games as much as I did before, this wasn't a really interesting project for me, though it did bring some really useful knowledge to me. I learned how to use and love GitHub which is to me one of the key point of group programming. I learned to use it on different platforms, through both GUI and shell. I also learned to use Unity though I don't really like it, it feels a bit inconsistent, for example the scene creator often change the values of the position of some GameObjects by some really small amount, which can cause headaches when trying to have a level floor for a block to slide on it. I had a hard time understanding

the way the API in Unity is written, and I still don't like it. I didn't learn much in the coding part since I am already good at C#, though I had to find some workarounds to make some thing function properly because of the way unity works (even though the problem probably came from me). I did learn a bit on how to manage a group, even if we had some hard times when some people don't understand and don't want to understand GitHub which slowed a lot the progress of the work. Though it was still a nice experience! We correctly followed the deadlines even though I wished I had more time to play with funnier things like map generation. I would have loved to make a more refined game, however because of group decisions 'and time I didn't get to it. However I did get to make levels as I wanted them, I tried to make them as clean as possible and to give them as much sense as I liked, and I am pretty proud of them. If there is something I would have done differently, it is the signal mechanics, I feel like they could get a bit more refining and optimizations, and work in a cleaner way, however with the way the unity API is made, it is getting really hard to find a workaround. Now I will probably not make another video-game since I didn't enjoy making it as much as I enjoyed working on my personal projects, but since most students were going for a video-game I didn't really had a choice. I did not hated this project, but I did not liked it as much as I would have liked to.

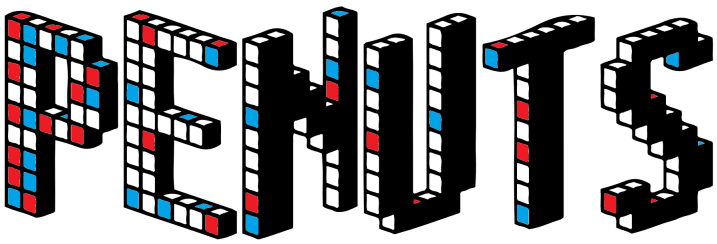
Pierrick MADE : This project really motivated me, even if we did not take enough time to organize work session, it was a nice experience. Indeed, it was the first time that I had to do a project of this scale not supervised. I learned many things thanks to that project, I acquired human and technical skills. The most evident being the technical skills, I will start with them.
I discovered software and tools like Unity, Overleaf, Brackets or GitHub. I'm really happy of that because I know that I will use many of them again, especially GiHub. I also learned how to create document using LaTeX which is a really good tool to do clean reports. It was really hard at the beginning, but it became really powerful. Finally with the creation of the Website, I discovered many features of the HTML and CSS languages then learned how to host and publish a website. Knowing that I have a page that everyone can see on Internet is also really rewarding.
In terms of human skills, I mainly discovered how to supervised and organize a group in a project. That was really hard to deal with every problems and every weakness discovered as the project evolved. But I understood it was a part of the job. But it was really rewarding to see those results with the quantity of problems we came across.

4.2 General Conclusion

This aim of this report was to present you the evolution of this project throughout these six months. Indeed, it was a long project and it's rewarding to be able to conclude it. This project coming to its end, we are happy to give back this game. Those six months were a real experience of work for all of us. It gave us many useful tools for the future, from technical to human ones (described in our personal conclusions). All along the project even if we encountered a lot of problems, we managed to respect the constraints laid down at the beginning in the book of specification and the deadlines of the presentations. However, we are mainly satisfied to give back a game we are proud of.

" Science isn't about WHY,
it's about WHY NOT! "

J.K. Simmons
Portal 2



Credits & Sources

Members of this Projects :

Alexandre POIRIER-COUTANSAIS

Pierrick MADE

Cloé LACOMBE

Ziane LAYADI

Thanks to :

ACDCs

Krisboul

Sources :

Unity Documentation

C# Documentation

Assets store & Blender

Computoser

LaTeX Documentation

HTML/CSS Documentation

Youtube's Tutorials

A lot of video games