TOLEDANO Gabriel

MERLE Timothé

PINSON Alexis

REPORT

LockedUp

Second Defense

26/04/2022

Table of content

1. Introduction

Through this report of our second defense, you will be able to follow our evolution since the last presentation.

The team consists of three members, Gabriel, who is the team leader, Timothé, and Alexis. Our game is called LockedUp, and it is a virtual escape game which aims to achieve a high level of immersivity for the players. This time, we all kept our tasks, meaning we finally found what each of us could and wanted to do. In fact, when someone wanted to try a new domain, the member who had more experience in it assisted him. Nevertheless, we did some changes in the game design which are not important in a technical way. Players will only find that these modifications bring more life to the game. For instance, you will discover new -retro- mini games.

Once everyone knew roughly what they had to do, the project could continue. From finishing the modeling of game objects to developing the last rules. Everyone worked hard to meet the deadlines we had set for ourselves. And thanks to all this hard work we managed to meet them and even exceed them.

So, for this defense, we succeeded in keeping all our objectives and we moved forward for the next one.

In order to describe our progress on the project, we have divided our report into several parts, each dealing with a different part of the project.

Have a good reading.

2. Details of the Project

a. Evolution of the team

Since our last project defense, our team has evolved a lot. Indeed, we have learned a lot in between these two moments. Foremost, everyone has been able to work on Unity and thus deepen their knowledge of this software. This one also made us discover that no software is safe from unexpected crashes. Indeed, several times Unity did not answer any more and corrupted all our files. However, we often make complete backups of our game, either locally or on GitHub. This allows us to quickly adapt to similar situations. Secondly, since we work with git technology, we sometimes have file conflicts that used to scare us and have now become a routine problem that we know how to solve. One thing we have implemented that you will discover in more detail when reading this report is that we work in different "scenes". This way, the automatically generated parameter files never conflict. The only drawback is that we always have to pull the elements in one place. However, this is less time-consuming than having to deal with multiple problems that could be avoided. Description of the game experience

b. Game Development

i. General Characteristics

At this moment, the game development is at a very advanced stage. In fact, some links between the code and the Unity game need to be done but everything else is operational. This report explains in more detail what we have done so far, but here is what our game revolves around. First, the multiplayer, an essential technology since an escape game is originally played in teams. This one is now fully operational. In addition, once we have created a path between each puzzle, the Game Master, the artificial intelligence, will finally be able to control the entire game.

c. Project Schedule

Let's now talk about our planning.

This one has been perfectly respected. Indeed, the common and individual tasks, which had been planned, are either on schedule or even ahead of schedule.

i. Common Tasks

These can be quickly summarized. In fact, the only thing that we worked on as a complete group was to finalize the 3D environments because we wanted to be sure that everyone liked the final view of the game. So, some of us made the lightning, other the decorations or even the sounds. This allowed us to finish everything for this defense.

ii. Individual tasks

In this section there is mainly the making of the different minigames as well as the backstage of the game. It includes the multiplayer, the solo mode and all programs that allow the player to interact with the environment that is around him. Of course, this is directly linked to the visual aspect of the game that should also be done individually. Thanks to our organization, every tasks could be done for this defense.

d. Work Breakdown

Since we have finished some of the tasks, we have removed them from the list as they were no longer useful. This allows us to keep a lighter record of our work.

Tasks	**Members**
Game and Web Development	
Website (adding the download file)	Timothé; Gabriel
Minigames	Alexis; Gabriel
NPCs	Gabriel; Alexis
Voice Chat	Gabriel; Alexis
Sound Effects	Gabriel; Timothé
Interact with the environment	Timothé; Alexis
Sharing Items	Timothé; Gabriel

e. Progression

Tasks	Presentation 1	Presentation 2	Presentation 3
Website	80%	100%	100%
Script writing	100%	100%	100%
Minigames	50%	100%	100%
NPCs	50%	75%	100%
Setting up the lobby system	50%	100%	100%
Voice Chat	10%	70%	100%
Sound Effects	0%	50%	100%
Interact with the environment	30%	60%	100%
Sharing Items	0%	50%	100%
Website Mock-Up	100%	100%	100%
Game Mock-up	100%	100%	100%
3D modelling of environments	80%	100%	100%
3D modelling of NPCs	50%	100%	100%

N.B:

- All percentages highlighted in green represent tasks that have progressed as expected.

- All percentages highlighted in red represent tasks that have progressed further than expected.

3. Tasks made by each Member

a. Gabriel Toledano

i. 3D Modeling

Since the project started, I supervised the 3D modeling of the different game environments. Meaning, the graphic side of the three level and their assets was made by me. Of course, the other team member helped me such as Timothé who made one of the three level.

Since the last defense, I have worked on making the game more playable, meaning, increase the fps. In fact, I had had an issue where the game suffered from too many details making it impossible to play. So instead of rebuilding everything a second time in blender -which would have taken too much time and not let me enough to work on the other parts of the game- I manage to make a lighter version of the game. Indeed, I first divided the different level in different scenes. It allowed us to delete some useless parts of the environment in the scenes that did not need them. This was possible because the concept of an escape game is to be locked in a room and not to walk in a free world.

After that, I took some time to improve the decorations, giving each level its own mood. This includes adding lights, some non-playable characters who speak when players are near them, typical elements such as dolls in the horror scenario but also game design elements such as visual enigmas.

Then in a more game development perspective I added 3D item that could be picked up by players. In fact, keys can be found in every level and must be put in an inventory by the player to be used later. This meant that links were created between the 3D environment and the players. They are now actors of the 3D world and not only spectators.

Finally, I will not have to do anything more on the modeling side for the game, except adding some items if needed, if ever we wanted to change a scenario because we feel that it is too hard for instance.

ii. Website

After the first defense, the website was almost done. Nevertheless, we still had to correct some errors and add new pages. For instance, the timeline and the section with the encountered errors were not filled at all. First, I updated the timeline. It gave people an overview of our time and task management in a chronological order. Then I filled the section concerning the encountered problems. It includes some of the issues we went in before the first defense but also the ones that we had during this second period of work. It allows us to give visitors more information about our work directly on the landing page. Now the project page is full, can be updated easily by copy pasting some lines of codes but most important, it is responsive, meaning, all devices, no matter its dimension, can see the website with a correct aspect. The website also contains a trailer for the game. I made it using the CineMachine asset in Unity. It gave me the possibility to create traveling movement with the camera and then capture what it sees.

After that the download page had to be done, most of it was made by Timothé but I assisted him to style it.

Finally, I made so that the website could be seen online. Thanks to GitHub Pages it could be done easily since they provide everything that is needed. This gathers a hostname, a ssl certificate by only adding a index.html file in our base repository that points to all other pages. For the next defense, we will have to fill the timeline, the encountered problems if any and add the downloadable files to the correct page.

iii. Voice Chat

For this defense, some new functionalities were supposed to be added to the game. The spatial voice chat was a part of them. It is the most important step for players to feel like in a real escape game. To make it possible I used the free software called agora. This web application is used to make voice and text chats. First, I created a project on their website. It gave me an app ID that I had to link in Unity. Something notable is that it can be set up as Photon, the service used to make the multiplayer mode. In fact, those two are connected since the multiplayer made with Photon is directly used in Agora to connect the players together in one lobby.

To achieve that, a chat room is created after a team manager starts a room. From there, people can click the button "Join the Voice Chat" and they will automatically be assigned to the voice chat linked to their room. We can ensure that there will never be two rooms with the same name at the same time since we use unique identifiers. In fact, to create a lobby for the multiplayer, we needed to assign it a UID that is randomly generated and most important, generated only once. So, to instantiate a chat room, I reused these UID as names for it. By doing so, I make it impossible for players to join a voice chat room in which they were not supposed to go.

Now, if they have the possibility to join one there is also a button to quit one. Joining and quitting trigger precise events that make requests to the agora servers. This service then sends us an answer that our code read and depending on it, throw an error message or assign the correct evaluation to it. So, from now on, if two or more players are in the same lobby, they can speak to each other easily. Even if it seems logical, the voice chat continues during the game and don't stop only in the waiting room scene. In fact, this could happen since, when a new scene is loaded the previous one is fully destroyed. However, we made it so that the needed elements of the waiting room stay during the entire game. It includes the Multiplayer and the Voice Chat Managers. For the moment, this voice chat is not spatial. This implies that no matter where the players are in the scene, if one speaks, everybody will hear him. On the opposite, in a spatial voice chat, I will define a radius of audibility which will be linked to a logarithmic function. It will depend on the distance between the player who talks and the ones who are listening, meaning, are close to the speaker. We also put them in opposition as 2D and 3D chats.

For the next presentation, the spatial voice chat will be activated, since it is not the difficult part to do to create a voice chat, the percentage that we had put in our schedule for this task is respected.

iv. Non playable character

Let's begin to explain the evolution of the Game Master. At the last defense, I had worked to make it a console application, to understand how it could work. After implementing its decision tree with only lines of codes I had to import it in Unity. That was quite a challenge since Unity uses different method than the one, we are used too. For instance, to instantiate an object, we should no longer use the famous **"ClassName object = new ClassName(someParameters)".** Instead, the method **"Instantiate( )"** ,that can also take coordinates as parameters, is needed. So, to create a link between the different game objects and the AI, everything needs to be created when the game starts. For example, once the game is running, the game master is created, and all its attributes need to be filled. For instance, the enigmas of one scenario. One way I did it, was to create enigmas from the elements that were already in the scene, such as **Canvas**. Nevertheless, the global logic of the decision tree did not follow. In fact, trying to apply an oriented object code to Unity is not the best way to do.

I understood that I ran in a new problem at this moment. After some thinking, I realized that one way to do will be to recode the AI directly in Unity. Meaning, I will take my basic code and then translate everything in a more "Unity Language" with less classes and more static method that will no longer depend on one object. This object was the previously the Game Master. So, this "translation" will be done for the final defense because we first need to change some details in the global game design.

Moreover, not having finalize the main AI was not an issue since we wanted to add other NPCs. In fact, during a game, players will make the discover of students who are also locked in the school. This people will, maybe, give the players clues or, at the opposite, give them fake leads. It will be to the concerned player to make its decision, between, believing, or not, the NPC. This is the goal of these. To achieve that I had to use the physic laws of Unity. These includes collisions. In fact, I added box colliders to the player and the NPC. A box collider is a defined area around a game object which makes it impossible for a character to go through. Nevertheless, there exist an option called **"On Trigger"**, which is a Boolean value, it allows us to create functions that are called if an object enters the collision area of another one. In my case, I made a function that starts an audio linked to the NPC if one of the players gets near it. Instead of making one script per non playable character I wrote one general that could be applied to every and each of them. Moreover, as said previously for the voice chat, giving a 3D effect to a sound makes the experience more immersive. To respect that, I made the sound fade depending on the distance between the players and the audio source, here, the NPC. For the next defense, improving any NPCs other than the Game Master will not be necessary.

v. Space Invader

During the game, players will have the possibility to play mini games to solve enigmas. There will be one per level since they will only have 20 minutes to finish the game and this step can sometimes be longer than expected. One of the three mini games that we have done is a copy of Space Invader. To access this minigame, a player needs to find a computer in a room on which to play. Once he clicks on a define key, such as **“P”**, a new scene will be loaded, only for the current player, others will neither be teleported in the new scene nor be allowed to enter the mini game.

Now, coding a 2D game in a 3D environment was little challenging since Unity gives the possibility to switch between the two views. So, after creating a new aspect ratio for this scene I could start its development. To begin with I defined the area in which the game will happen. To accomplish this, I created virtual boundaries which could, for instance, stop the projectile either from the players or from the enemies. In fact, in the Space Invader game, the player could only shoot one missile until it was destroyed, meaning, either it touches the boundaries or an enemy. After that, I had to generate the enemy lines. First, I used Sprites which are a type for images that is available in Unity. Then I made three different prefabs, one per type of enemy. Those allowed me to generate several line and columns thanks to basic **“for”** loops. Now the player had to be created. As for the enemies, I used a sprite that I could move left and right without getting out of the game boundaries. The player can shoot, meaning, it can destroy enemies. Each enemy has one life so that only one missile from the player can destroy them. For the game to be a bit more difficult as the level goes, I created 3 variables:

- The number of enemies
- The number of destroyed enemies
- The percentage of enemies that are still alive

Thanks to these, I could increase the movement speed of enemies when their number decreases. I made it as an exponential function so that the more enemies are destroyed the quicker they become. The speed increases slowly at the beginning and once 50 percent of the enemies are killed it goes up faster. Since they also go down when they touched the left or right border, enemies also go down quicker. This is an important feature of the game since an enemy touching the player will result in a **“Game Over”**. It is one of the rules leading to the end of the game. Another way of losing is to be killed by one of the missiles that are randomly shoot by enemies. In fact, I wrote a script that, sometimes, takes a random number and attributes it to the corresponding enemy if it is alive. The less enemies there is, the more missiles they shoot. So now we get to the real endings of the game. First, a player can win and then get a clue if he kills all the present enemies in the minigame. Otherwise, if one of the two losing scenarios happens, the game restarts until he successfully destroys all invaders. In fact, the level in which we will put this minigame, which is the third level, is the most difficult one. It means that we need players to take more time on each enigma in such a way that their success is uncertain.

vi. Game Design

Thanks to the work of Alexis, I could easily know when the game is supposed to stop whether the players has won or lost.

The lose state is not very difficult to trigger since it depends on the time remaining. If it goes to 0 then the player is redirected to a scene in which it is written: **“You Lost!”**. After that, the player can click a button which will redirect him to the waiting room, the place where he can choose which game he starts.

The win state is harder to get since it depends on the enigmas done. In fact, it is done so that when all the enigmas’ states are set on **“True”**, the players are put in a similar scene as the previous one. The only difference being the generated text which will display **“You Won!”.** After that, players can also go back to the waiting room, no matter if they are in solo or multiplayer mode. From there, people can quit the room, quit LockedUp, restart a game or create a new room.

b. Timothé Merle

i. Lobby System

I had to set up a lobby system for the first defense that allowed the player to play together. To achieve this task, I used a tool called Photon Engine. However, for this defense, I had to solve several problems that appeared during the development of our game. First of all, the ID we used to allow a player to join his friends was very long, but we were sure that it was unique because I was using a property of Unity. This property is called UID which means "unique identifier". So, we decided to keep this unique aspect of an ID but make it shorter so that players wouldn't take too long to copy it. So, I decided to look at what other games were doing. And I remembered that a game I played a lot during the confinement, this game being Among Us, was also using a login system to allow players to play with each other with only 6 characters composed only of capital letters and numbers. So, I thought about how to set up such a system. And the only way to do this was to look at the list of groups of players already created, generate a code of 6 characters of capital letters and random numbers, and check if this code is ever-present in the list of groups already created. And if ever a group already has this code, then we must generate a new one and redo the same manipulation until we find a code that is not yet used. In this way, we are sure that each code that is generated is unique and that there is no way to have any conflicts. However, a problem arises. Indeed, using UIDs allows us to be almost sure that we will never reach the maximum number of codes that we can generate because it is almost infinite. Indeed, the UIDs are codes composed of 16 characters with numbers, and lower- and upper-case letters, which represent 62 characters. Each character can be repeated as many times as the total number of characters in the final code. So, we can simply use enumeration to know the total number of possibilities. The number of possibilities is, therefore, 62^16 possibilities. But with the new technique, we had to implement we only have 36^6 possibilities of code. Since this number is also very large, we thought that we would probably have a server problem long before we would have a problem with the code generation.

We also aim to make life as easy as possible for the players. In the previous paragraph, I explained how we generated our new code. But it's still tedious to have to memorize it to pass it on to your friends. So, we thought of a way to make it much faster to send the code to your friends. We thought that if we could simply copy the group code by clicking on it, then the players could simply copy it into a chat room that they were chatting with. So, I went online to see if Unity allowed this kind of manipulation. And fortunately for us, Unity does allow this kind of thing. So, I turned the group code into a button and not into a simple text. And then I added a text right next to it that is not invisible. And with the button properties in Unity, we can know when the player's mouse is over a button. And by knowing this information, we can change the background color of the button to let the players know that they can click on that button. Once the player decides to click on the group code, then the background color of the button disappears and so I bring up the text that is right next to it that says, "ID Copied". The group code is copied to the clipboard of the player's computer and all he has to do is copy this code and give it to his friends so that he can join. To add a little more style and understanding for the player. We decided to remove the text that gives the player the information that the code has been copied after a few seconds. Again, on this task, Unity helps us a lot because it allows us to call a function after a certain time. So, I used this property to call the function that makes the invisible text disappear 1 second after it has been copied to the player's clipboard.

ii. Inventory

I'm now going to tell you about a new feature that we have implemented in the game to give the player a better game experience. Indeed, so that the player can see all the keys he has collected or even just the different clues he has collected during his game. To do this, we have set up an inventory. Nothing very complicated, just a visual palette where all the objects of the player are listed. We have given him 10 places in which he can store what he wants. The player can open his inventory at any time during the game, he just has to click on the "E" key of his computer keyboard. Once this is done, a small rectangle will appear with several small rectangles inside, representing the space he has in his inventory. To make it easier for the player to know when he has an item in his inventory, an image representing the item he has picked up will be displayed (see Figures 1 and 2). Also, to allow players to exchange the different items they have picked up, we had to set up an exchange system. To allow players to exchange items even if they are located miles away from each other, we simply allow the player to throw an item on the ground. All he has to do is open his inventory, click on the item he is interested in and follow the instructions that will appear on the right side of his inventory (see Figure 3). The instructions include the name of the item in question as well as an indication of how to proceed to throw this item on the ground. If he decides to throw the item on the ground, it will disappear from his inventory and reappear in the eyes of all the other players so that they can pick it up in turn. And similarly, when the player picks up an item on the ground, it will disappear from the eyes of all the players to prevent this item from being duplicated endlessly.

Now that I have explained the global functioning of its inventory, we must tell you that we encountered some problems during the realization of this one. Indeed, the inventory must be something unique for each player. We could have decided to make a common inventory, but that would have been too powerful in an escape game. So, we opted for an individual inventory for each player. However, unity does not distinguish between local and online players. However, Photon does. Therefore, to overcome the problem of inventory synchronization, we only had to place the object that represents the inventory in Unity as a child of the object that represents the player. The player is initialized by Photon Engine when the player starts the game. And we added a Photon Engine compartment on the player called Photon View. This compartment makes the difference between the local player and the other players.

Another feature that we have developed and that I have already started to tell you about is the possibility for a player to pick up an item on the ground. Indeed, when the player approaches an item on the ground, a small message in the bottom right corner of his screen will appear to explain that he can pick up this item without any problem. Also, when the player clicks on an item in his inventory, the name of the item will be displayed. To do these two things, I had to create several scripts. First, I created a script that allows giving an item its name, a short description and an image. This image is the image that is displayed when the player opens his inventory (see Figure 3). This script will inherit from one of the unity scripts which is named "Scriptable Object", which allows unity to say that this script is not a script that will add an object to the game but simply code that we will use. Secondly, a script is assigned to the item that will be present in the game. This script takes in its variables the Item object that we created just before, so we can get the name of each item as well as its short description and its image to display it. And thirdly, I had to create a script that is assigned directly to the different players. This script allows us to do all the functionalities of the player, like opening his inventory, this kind of thing ... And in this script, there is a part where we look when the player collides with another object in the game, as an item for example. And if the player is in the collision box of a key for example and he decides to press the "P" key on his keyboard then if he still has enough room in his inventory, the key disappears from the world and goes into the inventory of the player in question. Another feature that we have developed and that I have already started to tell you about is the possibility for a player to pick up an item on the ground. Indeed, when the player approaches an item on the ground, a small message in the bottom right corner of his screen will appear to explain that he can pick up this item without any problem. Also, when the player clicks on an item in his inventory, the name of the item will be displayed. To do these two things, I had to create several scripts. First, I created a script that allows giving an item its name, a short description and an image. This image is the image that is displayed when the player opens his inventory (see Figure 3). This script will inherit from one of the unity scripts which is named "Scriptable Object", which allows unity to say that this script is not a script that will add an object to the game but simply code that

we will use. Secondly, a script is assigned to the item that will be present in the game. This script takes in its variables the Item object that we created just before, so we can get the name of each item as well as its short description and its image to display it. And thirdly, I had to create a script that is assigned directly to the different players. This script allows us to do all the functionalities of the player, like opening his inventory, this kind of thing ... And in this script, there is a part where we look when the player collides with another object in the game, as an item for example. And if the player is in the collision box of a key for example and he decides to press the "P" key on his keyboard then if he still has enough space in his inventory, the key disappears from the world to be stored in the inventory of the player in question.

iii. Screemember

Another feature we thought of to simplify the life of the players is to be able to take screenshots. A screenshot is a capture of the player's screen that allows immortalizing the moment. Because the game is an escape game that takes place in Epita, a very large school with many floors. However, allowing players to use screenshots is very powerful in the context of an escape game where the objective is to use their memory and think to solve the various puzzles. That's why we decided to enable or disable the possibility to take screenshots depending on the level chosen by the players.

To begin with, I had to set up the screenshot system. To do this, I did some research, and while doing so I realized that unity allowed me to take screenshots very easily. A library is pre-installed in unity that allows taking screenshots simply by calling a simple function. This function takes as a parameter the name of the file and therefore its link in the files of the computer. So that players don't get lost once the game is finished and they have closed the game. We decided to store all the screenshots in a folder that we created called LockedUp and then Screenshots. And in this folder, we create a folder with the name of the day they are playing, so they can find their screenshots much faster. To stylize a bit the way the screenshots are taken and to tell the user that the screenshot has been taken and saved on his computer I added a text that is invisible at the launch of the game. When the player decides to take a screenshot, he just has to press the "I" key on his keyboard. When the game detects that the player has pressed his key, then it makes the text visible and writes in it "Screenshot has been Taken:" followed by the name of the new screenshot that has just been taken. In this way, the player knows the name of the screenshot he has just taken and can find it very easily.

However, while playing the game and using the screenshots, we quickly realized that having to leave the game to go into the computer files to look for the different screenshots we took during the game is very tedious and is a very long manipulation to perform which breaks a little bit all the dynamics of the game. That's why we decided to set up a small interface allowing us to see the different screenshots that the player took during the game directly from his game and that he doesn't need to pause his game. Unfortunately, we didn't have the time to implement this feature for this presentation, but we have already started to put the basics of this feature in place. The system allowing to define the level and the number of screenshots that players can take is already in place. However, I will tell you more about how I developed this feature in the report for the next presentation. This way all information will be centralized and therefore much more understandable.

iv. Website

For this defense, we also had to finish the website. We already had a good start for the last defense, but we were missing a page where anyone can download our game. I simply added this page. And my friend Gabriel filled on the website. That is to say, he filled in on the main page of the site the different problems we encountered during the realization of our project and also what we did and when we did it. But he also hosted the website online. So, anyone can find our website on the internet and access it through GitHub. On the download page, we find not only the main game that you can download. But also, a lite version of our game. Indeed, less powerful computers can sometimes have trouble generating all the objects we use and therefore, the computers can either crash and make the game simply inaccessible or have a very low FPS number. FPS is simply the number of frames that the computer will display per second, so the lower the FPS the jerkier the game will feel. Therefore, it is necessary to allow any type of player to be able to play our game with minimal comfort. Moreover, on this download page, you can also find our first report. And after that, you will be able to find the future reports and manuals that we will write as we go along with our game.

So, our website is now finished in the main lines, we will only add information as we go along.

For the design of the site, we tried to keep something very pure in any case we tried to do what we could with the knowledge of graphics that we had. As a result, the site is easily understandable for anyone who visits it and you can get to the page you want to go to very easily and very quickly.

As for the pages we had already done, I simply refocused the elements to have better management of the elements on the different pages.

v. Game Master

When my friend Gabriel finished developing the Game Master in his terminal, it was necessary to implement it in unity to make it compatible with the different mini games we developed and the different puzzles to solve. To do this, I helped him to transform the code he had written and make it compatible with unity. However, during this transformation we encountered several problems, that's why today the Game Master is still not fully operational, but we are actively working on it to make it work for the next defense. We have already started to rearrange the bass to make it compatible, but we still have a lot of work to do to make this artificial intelligence fully operational and autonomous.

vi. Puzzle mini game

I didn't develop the different mini games that players will be able to play. However, I was able to help some of my comrades. I particularly helped Alexis, not in the development of the teaser game but in the way to implement the game in the main scene to allow the players to interact with it. Indeed, Alexis developed the game engine in a separate scene, allowing him to have no obstacles and to be quiet during its development. Only by doing this, do we forget the main problem, which is that players must be able to interact with it without too many problems. So, we first thought about changing the scene. But this change would require loading time, and therefore waiting time for the players. So, I dedicated myself to trying to find a better way that wouldn't involve changing the scene. I ended up deciding to display the puzzle on the player's screen only and not on the other players' screens. To achieve this, I had to modify very slightly what Alexis already had to make the game compatible. To display anything on the player's screen we can only display objects that are UI objects. However, Alexis didn't have this problem and was just using unity objects. So, I added images to the prefab objects that Alexis had created and created some scripts so that when the player clicks on an image in the puzzle then the prefab object behind the image moves as well.

Once this was done, I ran into a problem, when the puzzle was displayed for one player all the other players saw it too. We don't want that at all because we want only the player who is playing the game to see the puzzle. But thanks to Photon we can know which player is in local and which player is online and we can do different actions depending on this result. So, I simply cancelled the display of the puzzle if the player who activates the puzzle is not the local player.

Once the problem was solved, the puzzle was functional and usable by all players. All that remains is to call the Game Master when one of the players has finished the puzzle to give the different clues and continue in the escape game.

vii. Solo Mode

Once the multiplayer was up and running, we also had to deal with the case where the player decides to start a game alone without any buddies. Thanks to Photon we can know if the player is ever in a Photon server or not. Then depending on this result, we can do some things. Indeed, if the player is connected to a server, then it means that the player is in multiplayer and we have to instantiate the different players in multiplayer and make the different actions that I have told you in the previous report. Only if the player is in a solo then it is not necessary to instantiate a Photon player. To solve this problem, I put a player that is disabled by default in all the game scenes. And if ever the player is solo then we just need to activate this default player with the camera and the different objects like the inventory.

Once we reactivate the default player, then the player can play alone in the different game scenes.

We also had to create different menus that allow the player to choose which game mode they prefer to play. I didn't take care of these menus; it was my friend Gabriel who created and managed them.

viii. End Game

My friend Gabriel has created a timer that keeps track of how much time has passed since the game started. If the players have not finished the game before the timer ends then it means that the players have lost the game. So, I helped my friend Gabriel to this end page with a single script that allows knowing if ever the timer is over or if ever the players have done all the puzzles and therefore to display "You win!" or "You lose!" according to this result.

c. Alexis Pinson

For this second defense, my tasks were mainly to create the mini games in separate scenes to be able to put them more easily in the main scene later.

i. Implementation of the 15-puzzle game

For the first defense, I had already coded this game so that it could be played in the console, not in unity. So, I had to find a way to make it playable on this platform.

To do this, I created a new scene. The first step was to make the game operational in a canvas. To make the game, I needed to create several new classes.

The first one is "NumberBox". This is the one that is associated with the room. It is therefore initiated 16 times. This class has several methods. The first one is "UpdatePos" which takes in parameters two integers. These two integers will be the new coordinates x and y of the part. The second one is "IsEmpty" which returns true or false if the room is empty or not.

The second one is called "Puzzle" and it manages all the pieces. It is therefore called only once. It has several methods. The method "Shuffle" returns a list of 16 integers (Figure 4.1). It allows to shuffle the puzzle while leaving it solvent. Then, there is the "ClickToSwap" method which will manage the movement of the pieces. It is also at the end of this method that we check if the puzzle is finished or not, and consequently if we stop it or not. Then we have two getters "getDx" and "getDy".

Of course, these two classes also both have their "Init" function.

The operation of the game is quite like the operation in console. When the game is launched, it is the "Init" of the puzzle script that will be triggered. The first step is to shuffle the list with the "Shuffle" method. Then, the different squares of the game will be generated thanks to a double loop. Finally, the game will detect when the player clicks on a square and adjust the puzzle accordingly with the "ClickToSwap" method.

It is Timothé who then took care to put the mini game in the main scene.

ii. The asteroid game

I will first explain the principle of the game. You control a ship that can shoot. Your goal: shoot at asteroids to increase your score without hitting them with your ship. If your ship hits an asteroid, you lose a life. You have three. (Figure 4.2)

To code this game, I separated the tasks:

- First, I set up the scene. To do this, I had to delimit the playing space by putting invisible objects that prevented the ship from leaving the scene. I also had to create the player in the scene and put the background in black.
- Second, I had to code the movements of the ship. So, you can move it with the arrow keys or even with the letter “Q”, “D” and “Z”. To do this, in the "Update" function, we see if the various keys concerned are pressed. If this is the case, the direction is changed accordingly.
- Third, I had to allow the ship to fire. For this, I created a "bullet" prefab. I generate these "bullets" when the player presses space or makes a click with his mouse. As before, it is in the "Update" function that we check if the space bar is pressed or not. Unlike the player, the shots must go out of the scene. They must not be held by the edges created in the first step. So, I used tags and layers to define who has collisions with whom. To do this, there is a place in unity called "Physics 2D" where you can manage which tags and which layers have collisions between them. It is also necessary to define a certain speed for the bullets. For that, on unity, we can modify 4 variables:
    - “Mass”, which is therefore to regulate the mass of the object
    - “Linear Drag”, which is to regulate the air resistance at the level of the movements. Here, we set it to 0 because the balls must not slow down.
    - “Angular Drag”, which is to adjust the air resistance at the rotation level.
    - “Gravity Scale”, which is to regulate the gravity that the element undergoes.
- Fourth, this is the longest and most complicated step: the implementation of the asteroids. Indeed, since they interact differently with all other objects, their code is complex. First, you must create the "asteroid" prefab. For the asteroids script, I had to create a class. This class has several methods:
    - the methods "Awake" and "Start" which allow to initialize them.
    - the "SetTrajectory" method which allows to give a direction to the asteroid. Indeed, the asteroid cannot be free: it must go towards the player to try to hit him.

- the "OnCollisionEnter2D" and "CreateSplit" methods which allow to manage all collisions according to the tags. If the tag is "bullet", then the asteroid is destroyed and can split into two smaller asteroids (it depends on its size). If not, it is the player and so the player is destroyed.

Then I needed to create the spawner. It is also associated with a class. This class has only one method "Spawn" which creates and sends randomly asteroids on the player.

- Fifth, you must manage the death (when the player meets an asteroid) and the respawn (because I remind you, the player has 3 lives). For the respawn, I made the player insensitive to everything for a few seconds. This way, if he respawns directly on an asteroid, the player doesn't respawn because he is in "invincible" mode. At the code level, we create a "GameManager" for this. It has several methods that will be useful later, but especially the "Update" method that checks if the player still has lives or not. If he has more, the player does not respawn. If not, the player respawns and loses a life.
- And finally, I made animations when the player dies, or the asteroids are destroyed. For this, in unity, I use the "Particle System". It has a lot of factors that can be modified. Here is a list of the ones I used:
    - “Duration”, it is the length of time the Particle System is emitting particles. I put it at 1 second.
    - “Start Lifetime”, particle will die when its lifetime reaches 0. I put it at 0.5 second.
    - In “Emission”, I created a “Bursts” of 10 particles.
    - In “Shape”, I decided to have a circle shape.

Then, I implemented the score and lives so that there is an interface in the game. For this, I used the "TextMeshPro" module, which allows to have text with more choices in fonts for example. So, I added text zones in the canvas that will be modified by the scripts during the game.

iii. Implementing a door in a scene

I also did the door opening in a separate scene (figure 4.3). It's not very armful but it's important to do it in a side scene to not destroy what was already done.

The difficulty here lies in the door itself: indeed, Gabriel had already designed it, and there is a part of the door that must not move. So, I had to separate his prefab.

To create this rotation, I had to create an EmptyObject, and put the door inside. The EmptyObject must be located at the level where we want the door to rotate. This way, when we change its angle, the door will turn too.

Then I created a Sphere Detection that detects if the player is in a certain area. This will be very useful to see if the player can open the door or not.

For this class, there are several methods:

- the "Update" method allows to detect if the "F" key is pressed. If it is the case, it modifies the Boolean variable "open" to open or close the door according to its situation.
- the "OnGUI" method displays the message at the bottom of the screen that says: "press F to open".
- and finally, the two methods "OnTriggerEnter" and "OnTriggerExit" which detect if the player is in the collision sphere or not and modify the "enter" variable accordingly.

In the game, it will also be necessary to check that the user has the key in his inventory so that he cannot open the door all the time but only when he has finished the puzzle, but I will work on that when I put in the main scene.

iv. Implementing sounds in the last level

To do this, I went to the scene corresponding to the third level and put screamers in some places to scare the players. I also added some ambient sounds.

With the help of Gabriel, I looked for the best corners where the player least expects to scare him. For this, it was important to work according to the location of the lights to scare the player as much as possible.

4. Conclusion

In conclusion of this report, we have done most of the client-side aspect of our project and have made an improvement in the developer's side. That is to say that we have for example the inventory, the sharing of items, the interaction with mini games and enigmas but also the multiplayer. Moreover, the details that needed to be settled in the 3D environment are now finalized.
For the next defense, we will have to finish implementing the voice chat, and the ambient sounds of the game. As for the improvements, we must implement the mini games like the Space Invader directly in the game. And we also need to tweak 2-3 things like the collisions bug and the website. We also must link the game master to the game, which is the main part of it. For this one we will principally just make it recognize the state of an enigma.

Thank you for your time.

5. APPENDIX

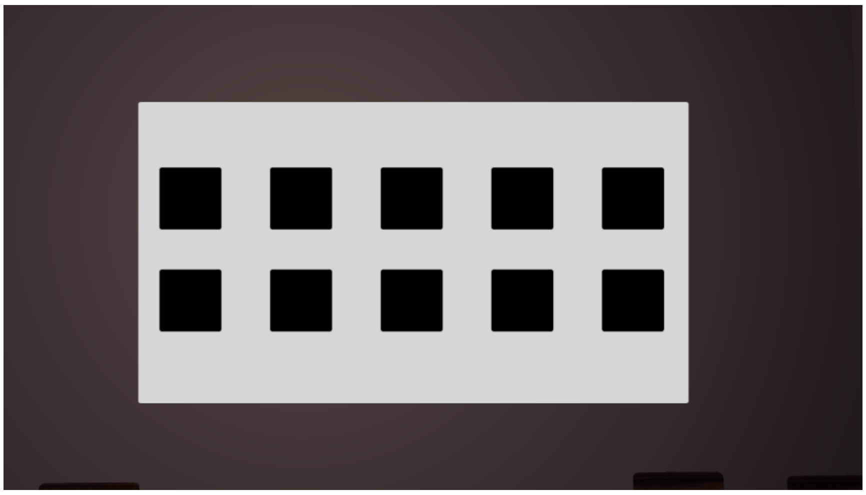


Figure 1

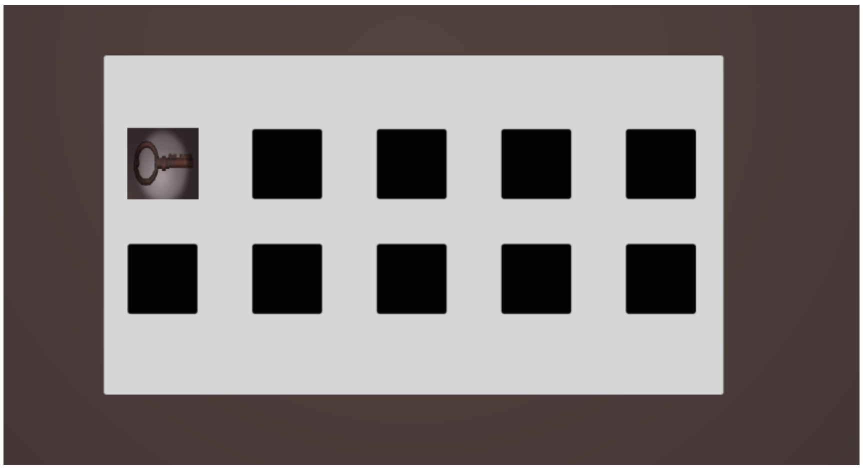


Figure 2

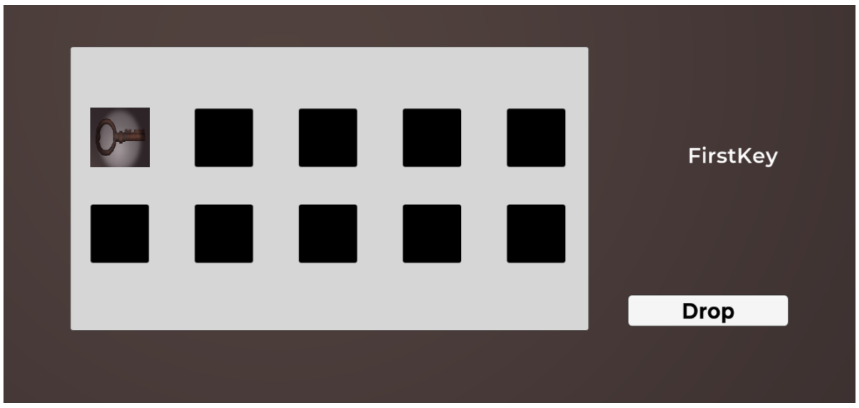
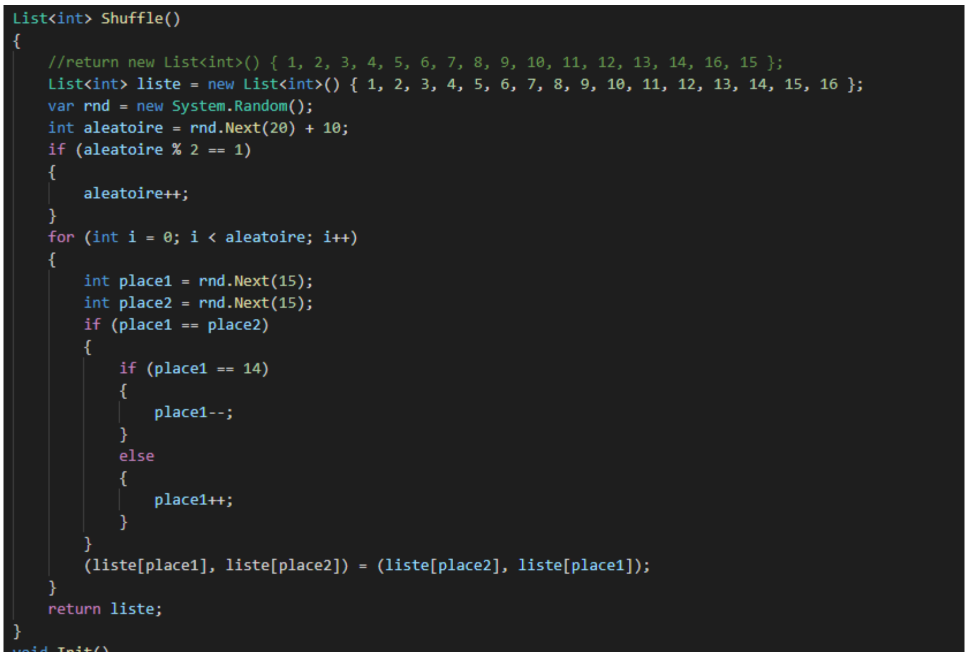


Figure 3



```
List<int> Shuffle()
{
    //return new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 15 };
    List<int> liste = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 };
    var rnd = new System.Random();
    int aleatoire = rnd.Next(20) + 10;
    if (aleatoire % 2 == 1)
    {
        aleatoire++;
    }
    for (int i = 0; i < aleatoire; i++)
    {
        int place1 = rnd.Next(15);
        int place2 = rnd.Next(15);
        if (place1 == place2)
        {
            if (place1 == 14)
            {
                place1--;
            }
            else
            {
                place1++;
            }
        }
        (liste[place1], liste[place2]) = (liste[place2], liste[place1]);
    }
    return liste;
}
```

Figure 4.1

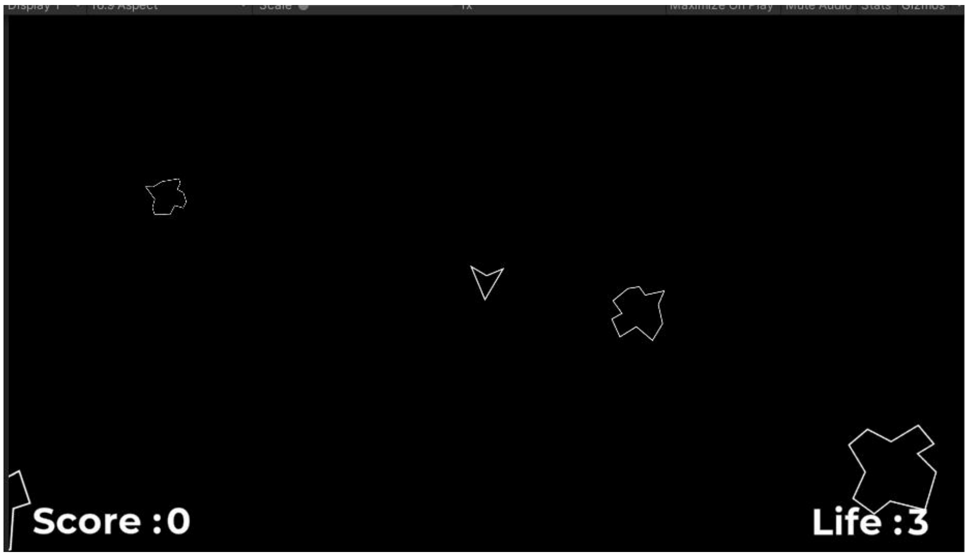


Figure 4.2

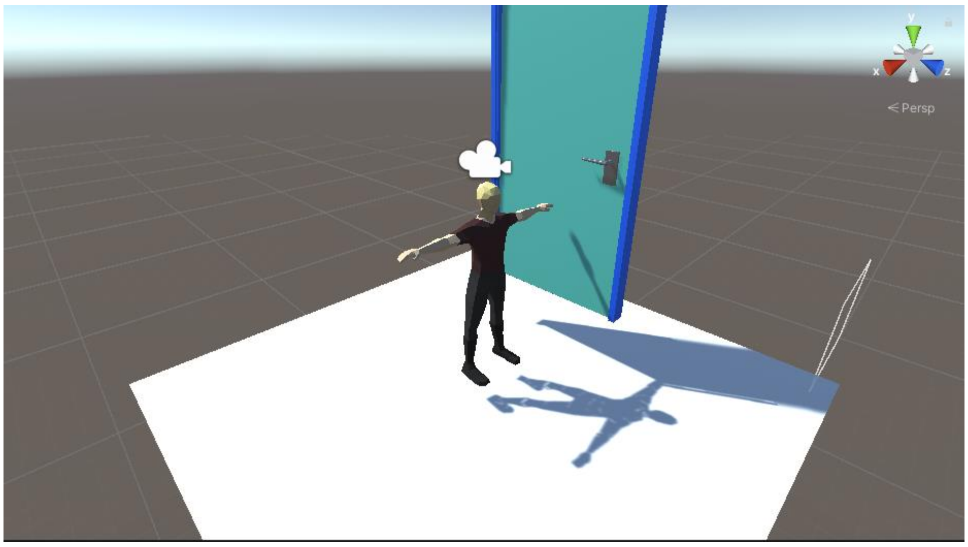
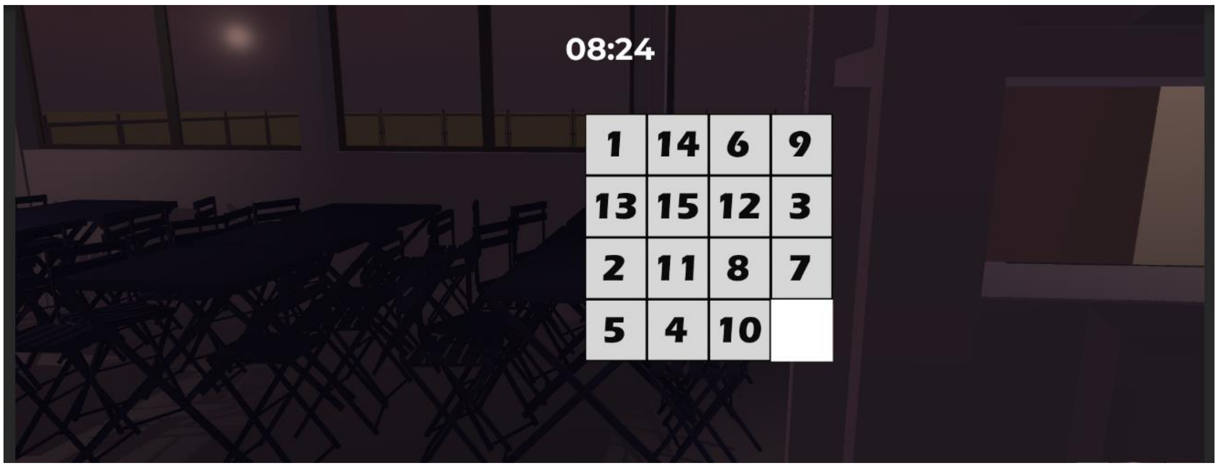
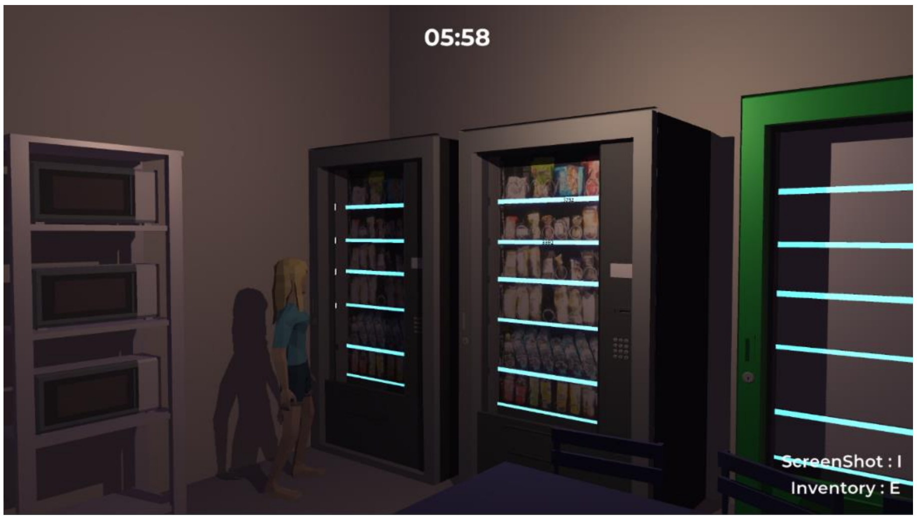


Figure 4.3



15 Puzzle game in-Game



One NPC giving a clue

LockedUp