PROJECT REPORT

07/06/2022

TOLEDANO Gabriel

MERLE Timothe

PINSON Alexis

TABLE OF CONTENT

I. Introduction

Over the last few months, our team, Jupitr, has been working on the development of a virtual escape game called "Locked Up". This game aims to provide a virtual experience that is close to the emotions one would feel during a real-life experience. Through three different stories and more in the future, players will be able to immerse themselves in the darkest corners of the EPITA school. Indeed, each story takes place on a different floor of the building on the Villejuif campus.

In order to achieve this, each member of our team has worked on the different tasks that were assigned to him at the beginning of the project. These include artificial intelligence, multiplayer and 3D development.

Of course, developing such a game was not a smooth process. Our team encountered bugs of varying degrees of severity during the entire development of the project. This forced us to review our vision of the game, our working methods and our way of developing the elements that are now present in the game. Indeed, as you will discover through this report, we used some tools external to what we did or that the game engine we used, Unity, made available to us. Some of them have created problems that we would never have thought of. It changed our vision of the game development. And it made us realize that this task has many precise and complex responsibilities.

This project report includes all the necessary information to understand why we wanted this project to happen. Moreover, it explains in more depth how the project was made and which technologies the team has used.

II. Project Specifications

A. General Presentation

1. Presentation of the team

The “Jupitr.” Team is composed of three young men studying computer science. Gabriel Toledano, the project manager who also gives the graphic ideas, Timothé Merle who deals with the technical issues and Alexis Pinson who deals with the game design. We are all passionate about video games in general, but also about life-size games. Indeed, playing a game is one thing, but living it is another. When we participate in games such as escape games we are in the center of the action, we, the players, decide how the game will progress. If we find the necessary clues, then we win, if not, we lose. This creates a real stress for us, but a positive one, the one that makes the game progress. The frustration of not finding the clues forces us to look harder to succeed in our mission. We aim to achieve this level of experience for our project.

2. Presentation of the project

i. Nature Of the project

In recent years, we have seen a great growth in the development of escape games. These usually take place in a single room and require players to find clues to escape. Each game can be played in teams or alone. Thus, the Jupitr. team is going to develop a video game taking the form of an escape game. It will be called LockedUp.

ii. Goals

What is LockedUp’s goal? Entertain players as much as a real escape game would. At first glance, one might think that this is going to be complicated,

because nothing can really replace the cognitive sensations of a game. Indeed, the smell and texture of the elements surrounding the players is complex to transmit through a screen. Therefore, Jupitr. aims to overcome the virtual barrier with sound and visual effects. With these objectives, we want to create a unique experience for each game. This means getting as close as possible to real-life experiences. Indeed, as when you choose the theme of the escape game in which you will participate in real life, several atmospheres will be available in the game. This will allow a diversity of gameplay and will prevent the player from getting bored by the game too quickly.

iii. Interests for the team

As a team, we want to improve our game development experience. From this comes several categories. Firstly, we will improve our programming skills. We will also learn how to model 3D environments. The team will understand the constraints and issues surrounding the development of a video game. Finally, this project allows us to improve our teamwork skills. Indeed, we will have to coordinate our actions to respect a precise schedule.

B. State-of-the-art

The first escape games created in 2005 were computer games. We can name Crimson Room and QP-Shot which were designed by the Japanese Toshimitsu Takagi. These games were based on puzzles that had to be solved in quick succession. There was not always a specific theme in each game. We will now look at three games of the same type as the one we are going to develop.

1. Escape First

Escape First was released in 2018. The game allows players to play cooperatively by gathering a team of up to six players. A versus mode is also available, which allows players to compete with their friends. The game contains three different rooms, each with its own theme and story. This limits players' search to a single room so that they don't waste time visiting unnecessary areas. Finally, Escape First is best known for its ability to be played with a virtual reality headset.

2. First Class Escape: The Train of Thoughts

“First Class Escape: The Train of Thought” is an escape room game, where you need to escape from a vintage luxury train. This game is played in the 5 first person and allows you to move through the train. Indeed, thanks to the elements that we collect during our research, we can have access to the different cars of the train. This game contains only one "room" but thanks to its size and the number of puzzles and items to collect each game is a different experience. First Class Escape can also be played solo or in teams. A team can consist of up to five people, which makes the game very interesting. Indeed, five people are not enough to cover the entire train, so it encourages players to progress together.

3. Escape Simulator

Escape Simulator is based on real-life escape rooms, where objects can be taken, inspected, and broken and the player's objective is to find out how to escape from locked locations. Again, the game has a single player mode as well as a co-op mode for playing as a duo. Although it comes with 15 different themed rooms, the game offers players to create their own rooms. This way, the game is constantly fed with user suggestions, which are then made available to everyone.

C. Game development

We now get to the most interesting part, the way the game works! Here we will present the different aspects of the game. First, we will explain how the game is

played in general. Then we will see how the artificial intelligence will bring life to the game. Finally, we will discuss the multiplayer mode.

1. General characteristics

LockedUp is a first-person puzzle video game which will be played on Windows. Once the game is launched, the player will have the choice between the single player mode or the cooperative mode, to which we will return later. After choosing the desired mode, the user will have to decide which adventure they want to embark on. We offer three different themes, each representing a different level of difficulty:

- Easy: Night at school After an intense day at school, you decide to stay a little longer to finish a project. Unfortunately, you dawdle too much, and the guard closes the doors of the school, leaving you locked in. The player will have to solve different puzzles and find items in the environment to go home.
- Medium: School Psycho You decide to go to your school on a weekend because you can't concentrate at home. When you arrive, the site is deserted, but when you enter the building, you hear a shrill scream. You must hurry to find out where it came from to discover the story behind it. In this theme, the player will take on the role of a detective to solve the criminal case that is set before them. The setting will not change much from the previous level, except for the traces of the crime that will be hidden all over the 3D environment.
- Hard: The whisper of darkness During a night of hard work, a power cut stopped your objective. The emergency lights turn on, and you hear whispers without knowing where or from whom they come from. You'll have to scour your school to get the power back on and, if you dare, uncover the secret behind the strange whispers. Finally, the bravest players will be able to choose this horror theme. In this mode, a real change of scenery will appear. Indeed, thanks to the lights, a darker

environment will complicate the task of the players. In addition, the puzzles to be solved will be more complex and fewer clues will be given than in the previous modes.

Three environments with modifications will therefore have to be developed.

We will now present what the scenes have in common. Firstly, each game will last a maximum of twenty minutes. A countdown will then be displayed on the player's screen to indicate the remaining time. Then, a bag system will be available allowing players to store items such as keys, for example.
Similarly, if the player retrieves visual information, they will be able to do a "screemember". This will take a screenshot of what the player is looking at and store it so that they can use the clue later.
The previous two points involve interaction with the environment offered to the players. In this way, we will allow users to jump on furniture or open cabinets to find clues, for instance.
Finally, for a more immersive experience, we will add sound effects. For example, the player will be able to hear their own footsteps or a cat meowing.

2. Artificial Intelligence

Although we have presented what the three themes have in common, we have not mentioned artificial intelligence. However, it is present in all three themes and in both single and multiplayer modes.
Indeed, the master of the game, who is the most important character, is an AI. It is this protagonist who will present the scene in which the player finds himself. However, its purpose is quite different. Indeed, the game master will constantly analyze the state of the game and its progress. Thanks to these two parameters, it will know if the players have been stuck on the same clue for a while or if they are progressing correctly. In this way, the AI will be able to help, or not, the players by giving them more or less precise clues. The accuracy will depend on the current progress of the players in the environment.

Other non-player characters (npgs) will also be controlled by AIs. These npgs will be present mainly to guide players. Their presence and roles will depend on the theme chosen. For example, in the Night at School environment, npgs may help you find a clue or put you on the wrong path. They will serve no purpose other than to annoy you and make the task more complicated in this case.
Finally, a pet will be present in the building. Players will be able to find kibble and bring it to him. In this way, the pet will lead them to a clue. Just like the game master, this pet will be directed by an artificial intelligence. The AI of this animal will also have parameters for the state of the game and will give more advanced clues to the players.

3. Multiplayer

We talked about the fact that there was a single player mode. But we didn't elaborate on the cooperative mode. Since we want to respect the rules of the game in real life, the main use of the network will be to form a team. By cooperating, players will have a better chance of finding the final solution. To realize this multiplayer mode, we will use photon engine which will allow us to have access to a server that we can link to our game in Unity.
This will give the players several possibilities:
• Create a lobby by giving it a name: a virtual room where players who want to play together can meet.
• Join a lobby by entering the name in the given input field.
• Select a character and a name so that they are recognizable during the game.
• Have access to the lobby's voice chat: players will be able to vocally share the clues they find.
• Players will be able to share recovered items with each other: For example, a key keeper can be designated by the group members. As soon as a key is found, it is brought to him. Another example would be if one player has three kibbles, and another has two. If it takes five to get a clue from the animal, then one player will give his or her kibble to the other so that they can get a clue.

・Start a game as soon as there are at least two players online.

All these possibilities will only be possible if the multiplayer mode has been previously selected.
To finish with the multiplayer mode, a game cannot be started with more than four players.

4. Game structure

To begin with, "LockedUp." is downloadable from the website. This implies that HTML, CSS, and JavaScript will be used to make it.
Then, the game will be done in C# (C Sharp). It is a multi-paradigm programming language, but we will mostly use its object-oriented features. Moreover, we will develop the game on Unity. Thanks to this, the graphic part is slightly easier to generate in the game. Furthermore, linking 9 the game script to the game elements will also be simplified by this software.

During the installation of the software, a window will appear for the user to choose what to install or not. This may include the installation of a shortcut on the desktop, for instance.

The game structure will be composed of three major parts:

・The home screen, that allows the user to select a mode between solo and cooperation.
・The themes screen, where the leader of the team will choose the story to play.
・The boot screen, which displays the name of the game as well as the team's one.
・The "in-game" scene in which all the magic happens.
・A Win or Lose screen will appear at the end of the game.

Since it should also be possible to uninstall the game, we have developed this procedure. It allows the user an easier uninstallation of LockedUp without having to go in his Control Panel.

III. Work Made

A. Gabriel TOLEDANO

1. Mock-ups

To begin with, I had two mock-ups to do: one for the website and another one for the game in general way. Meaning, I made a quick design to explain to what the menu could look like for example. These two mock-ups were done at the beginning of the project and finished without any problem.
Even if this step adds some time to the global development of the game it decreases the time we spend when implementing the real game. Indeed, if the developer already knows where the elements should go, he will only have to make sure that what he does look like the model.

2. 3D Modelling

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.

The 3D Modelling seemed to be essential for the rest of the game development. In fact, since we need to make three different environments for the three levels that we are going to do, a base is needed for scenery. From that on, the last 20% that need to be done include the decorations and lights of each level.

To get a quick visual easily I, with the help of Timothé, modelled the whole building and most of the assets directly in Unity. In fact, the Prefabs, those 3D models that are dynamically editable, are a very powerful tool. The 3D modelling, in a low poly style, took the entire time from the beginning of the project to this first defense. This includes modeling the building, setting up the environmental lighting, creating all the prefabs used in the game and finally modeling the characters. Indeed, once the building was almost finished, I focused on the appearance of the characters. Those were taken in the Unity Asset Store and then modified, the skeletons and the mesh, directly in Unity.

Nevertheless, modeling everything in Unity made us encounter a major problem: a decrease of fps. In fact, the number of "Vertices" and "Triangles" which could be explained as being the level of detail of the scene, is too high. This is due to the unique number of elements. To counter that, I have modeled some elements together in blender and then import them in Unity.

So during the project, I have worked on making the game more playable, meaning, increase the fps. The issue where the game suffered from too many details made 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.

To finish the 3D modeling part I added the so called “Box Colliders”. These were used by the team to trigger elements when a specific event occurred. For instance, when a player is in the radius of a 3D item than can be picked up, something according to this event will happen. This is only possible this way, by linking the 3D world to the script of the players.

3. Player’s interactions in the environment

This task consisted in making a player move in the scene and making him know how to interact with the different present element. First thing was to choose the type of view we wanted to give to the players. We decided to go for a first-person view since we want to give the most immersive experience. To achieve that, I linked the camera to the player thanks to a script that made the user turn his head using his mouse. At the same time, the script to move the 3D model of the player was made. Its purpose was to make the character go in the 4 directions, jump, and fall thanks to basic physics. Using box colliders, I determined the boundaries through which a 3D model could not go.

The problem in this case was to compute the velocity of the player to make its fall look realistic. To achieve that, we added a tag to all elements on which a character could step. Example: the ground. Then a not visual object has been added to the bottom of the character that recognizes this tag. Thanks to this, we set the velocity to a certain value when the player is “Grounded” (on the ground), otherwise we make the value increase according to gravity.

This allowed me to enable animations on the player’s character. In fact, walk, run and jump/fall animations can be seen in the game. All of them are created thanks to Unity Animator and parameters. The activation of each animation depends on different parameters. For instance, going from Idling to walking to running, depend on the speed of the player. In the same way, depending on the velocity of the player and its position on the vertical axis of the environment, the player may jump or fall. When I implemented this, I did not think a second that this could

not work in multiplayer mode. Nevertheless, it appeared that Photon do not let player see each other animations. Timothe had to fix this issue when working on the network.

Of course, all of this worked well until we all wanted to add Options to the game. They enable players to change the default keys that are bound to specific events. Nevertheless, Unity has already its input manager which handles going forward, backward, left, right and jump. Meaning, instead of simply using these keys, I had to redo the physic of the game to move on a three coordinates vector for instance. In fact, the values can now only change if the default keys or the ones put by the player are pressed.

4. Website

I have worked, as Timothé's assistant on the website. My main goal was to make the landing page. This one consists of a clean design with a classic navigation bar. When you click on "See More" the page goes down and you can see the timeline appear. It is accompanied by a section listing the various problems encountered during the development of the project. These two elements were not filled at the beginning. Indeed, we were waiting to have more content and this filling has been done for the following defense. The timeline lists the most important steps of the projects. It includes the 3D modelling, the development of mini games but also the one of NPC's (Non-Player Character).

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. At this moment, the project page was full, could be updated easily by copy pasting some lines of codes but most important, it was 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 last defense, I have filled the timeline, the different problems which we encountered and added the downloadable files to the correct page.

5. Artificial Intelligences

Now we get to the interesting part of the project: the different A. I's. First, the Game Master handles everything that can happen during a game. I chose to make its program look like a decision tree. In fact, its decision will be taken according to answers to specific questions. I first started by literally write the decision tree. Then, once I had my idea, I started the Game Master in a C# console application. I wanted to try it out instead of directly putting it in a unity scene. The current state of the Game Master allows to do multiple tasks. At the beginning of the games, it computes the time needed for each enigma. This depends on the number of enigmas and the amount of time the players have, to finish a game. The enigmas are loaded at the beginning of the game and each one of them is linked to a clue.

After that, a function constantly calls the Game Master to help or not the players. This decision is, for the moment, made this way:

- Each time the Game Master is called, it computes the amount of time that could be used to finish the current enigma and still win the game. Winning the game means succeeding in all enigmas/steps of it before the global time

goes to 0. The computation is a division between the remaining time and the number of remaining enigmas.

- If this time is less than the value computed in the first place, we chose to give the players a clue.

After that, I planned to add multiple parameters to the decision tree of the Game Master. First, not all enigmas have the same difficulty. In fact, finding a key in a room and winning a chess game don't exactly take the same time. This will result in computing a level of complexity for each enigma. According to this level, the Game Master will allow more or less time to solve one problem.

Secondly, not all games are going to be played in solo. It is easier to win in team than alone. For instance, when there are multiple players, they can cover a larger space of the environment to find an item. This will also be considered when calculating the time needed for one enigma.

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.

This is what I did. I managed the A.I with one game object which appears in each scene. When the game starts, this game object starts all the mechanism with the different attributes for the game: enigmas, number of players and for the game master (the game linked to it). Now, the Game Master constantly analyses the game and if the time is too short for one enigma it helps the players depending on the complexity. If the enigma is done it removes it from the list of enigmas and goes to the next one. One aspect that I had to implement was, how will the Game Master give out clues? I finally made it so that the player could see the clue appear on its screen for a defined time and so easily read it and take a screenshot if needed.

Secondly the non-playable characters. 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. To finalize this aspect of the game I modified the curve that was linked to the sound as well as the trigger event that started the sound.

6. Voice Chat

This is the last part that I worked on before the first defense. At this moment I used agora voice channels. Agora is a company providing vocal services to make calls but also to make your own vocal chat project. I started

implemented this on a different scene than the one where Timothé developed the multiplayer. We work like this to limit our number of merge conflict in git. Doing so, when playing the scene, you can:

- Create a lobby/channel
- Join it or a previously created one
- Leave the joined channel

After that, 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. 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 mSanager 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.

This voice chat is spatial. Contrary to non-spatial voice chat in which no matter where the players are in the scene, if one speaks, everybody will hear him. In our case, in a spatial voice chat, I have defined a radius of audibility which is linked to a logarithmic function. It depends on the distance between the player who talks and the ones who are listening, meaning, are close to the speaker. We also put those two types of chats in opposition as 2D and 3D chats.

7. 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.

Of course, this mini game needed to be implemented in the real game. Timothe took care of that. Once it was done, I made it so that if the player wins, a code is given to the player so that he can open a safe. If he loses, he will have to play again until he wins. Else, him or its team may lose…

8. 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. In fact, after writing all scenarios he made it so to only write enigmas one by one so that we could easily know when each enigma was supposed to appear.

First, 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 a message that indicates to the player that he has 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 a message indicating that the player has 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.

So I had to create enigmas that had all this those attributes. I was able to develop that thanks to the Scriptable Objects of Unity. In fact, this type of object is made so that it can be linked to a script but also to a game object present in scene. It means that they behave exactly like a script except it is way easier to handle. The enigmas I created have multiple fields.

The Necessary Fields:

- Their Name: to identify they during the game
- Their Place: in fact, since all the scriptable objects are loaded from a specific file into a list, we have to sort them. This is made so that the current enigma is the first element of the list. In fact, since we delete each enigma when we finish one, the next enigma in the list becomes the first

element of the list. So we implemented a basic sorting algorithm depending on the place of each enigma.

- Their Clue: the Game Master uses this field to change the text that will appear on the player screen when it gives out a clue.
- Their State: this significantly tells us if an enigma is done or not. In fact, this is very important since the condition to delete an enigma is actually if its state is set to True. One problem that I encountered during the development of the enigmas was that each time I launched the game their state was not reset to False. To counter that, I put the state of each enigma to False when I delete one from the list.

The Optional Fields:

- The Item To Pick Up: this field represent the Item to Pick up to solve an enigma. It is an optional field since some enigmas don't require such an action. In some case, this field is simply set to null.
- The Code: this field represent the code to put somewhere to finish the enigma. This field is frequently linked to the previous one because it happens that entering a code unlock an item to pick up. Of course, like its predecessor, this field can be set to null if no code input is needed to solve the current enigma.

Since Alexis developed another mini game called "Asteroids", I implemented an enigma behind it. A player can launch this mini game by getting near a computer. Once the player achieves to make a define score, the game stops and the solution for an enigma appears on the screen of the computer instead of the previous screen. This solution is simply the killer that the player is looking for. To make the game more difficult and more entertaining, I created multiple screens with different killers. Each time the player wins a game of 'Asteroids", our game manager randomly chooses a screen to display. This is directly linked to the end of the general game where the player needs to tell the police the zip code of the killer. Zip code that is written on the random chosen screen.

B. Alexis PINSON

1. Script writing

To realize the different stories, I played about 5 escapes games, and I also watched about ten videos. Indeed, to realize an escape game from scratch takes on average 2-3 years.

i. Night At School

The objective of this first game is to familiarize the user with the interface. Therefore, the difficulty is relatively low.

The escape game is broken down into several rooms. First, the players appear in a locked room. They must get out by simply finding a key in the room. Here, the user will be able to get used to using their inventory and moving around in the great place that is Epita.

After this first very simple part, you will have the possibility to walk around the ground floor to find a way to get out of the school. You will encounter several challenges, including letters written on dice, computers that must be turned on, and a 15 puzzle (which I will explain later).

All these tests will lead to the exit, and you will have finished the first level of this game.

ii. School Psycho

This second level is already more complex, because it uses mechanics that are more difficult to understand.

This level is like a “cluedo”: players find the place, the weapon, and the killer. They put themselves in the shoes of a real detective.

To do this, players must first find and use a key to access the murder room, which is none other than the bathroom. Then, they must play with the light to find the murder weapon which turns out to be a knife. And finally, the players must find the name of the killer thanks to the fingerprints he left on the knife.

iii. The Whisper of Darkness

This level is much harder than the other two. Indeed, players will have to find a way to turn the light back on, especially with the help of cable-based puzzles. However, their mission will not be finished: they must then discover what really happened at the school. To do this, they will have other tasks to complete such as a game of Space Invader, but they will also have to use their remaining architecture course as there will be xor doors with symbols.

2. Code of the mini game “15 puzzle”

i. Code in C#

What is this game? It is a square made up of 15 small squares with a free space. The objective of this game is to reconstitute a pattern by sliding the pieces in the free space. (Figure 11)

At the code level, I used two classes: a class called "Piece" which manages the different pieces and the class "Board" which manages the board.

The "Piece" class has 3 variables: the x and y variables which allow to know its location on the board and the image variable which will store the image that this piece should display. There are also the getters, the setters, the constructor, and another method. This method called "GoEmpty" returns a Boolean indicating if the concerned part can go to the free location placed in parameter. Moreover, if it can, the two parts are reversed.

The "Board" class has only one variable which is a list of "Piece" but has 4 methods. The first one is "Shuffle". It allows to mix as its name indicates the pieces. However, it is in this part that we find the greatest mathematical complexity. It is important to know that it is possible to tell in advance whether the problem posed is soluble or not. Indeed, the initial configuration of a teaser is a permutation of its final configuration. This permutation is called even if it can be obtained by an even number of successive exchanges of two squares, adjacent or not, empty or not, also called transpositions. We show that this notion does not depend on the choice of the sequence of exchanges. It is odd otherwise. We also associate to the empty square a parity: the empty square is even if we can go from the initial position of the empty square to the final position in an even number of moves, odd otherwise. The problem is solved if the parity of the permutation is identical to the parity of the empty square. The difficulty is therefore to mix this puzzle without making it insolvent. The second method is called "Display" and allows the user to see the board in the console. The third one is called "Move". It takes a room as a parameter and returns a Boolean if it has been moved or not. For this method, I use the "GoEmpty" method of the part class. The last method "IsFinished" returns true if the puzzle is finished, false otherwise.

All these classes and methods now make it possible to have a very simple main program. Indeed, it is now enough to initialize the puzzle, to mix it, to display it and to create a loop which stops when the board is finished.

ii. Implementation in a separated scene

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. 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.

3. Code of the mini game “Asteroids”

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 13)

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:

o “Mass”, which is therefore to regulate the mass of the object

o “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.

o “Angular Drag”, which is to adjust the air resistance at the rotation level.

o “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:

o the methods "Awake" and "Start" which allow to initialize them.

o 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. 18

o 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:

o “Duration”, it is the length of time the Particle System is emitting particles. I put it at 1 second.

o “Start Lifetime”, particle will die when its lifetime reaches 0. I put it at 0.5 second.

o In “Emission”, I created a “Bursts” of 10 particles.

o 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.

4. Implementation of a door in a scene apart

I also did the door opening in a separate scene. 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.

Unfortunately, for the final rendering, the doors are not animated because it posed too much problem because of the multi-player.

5. Implementation of sounds

To put the sounds, I first read the scripts of the different stories. Indeed, it is essential to know what sound to put but especially where to put it. Then, I searched online for audio clips representing the sounds I wanted to have in mp3

or .wav format. I separated them into two categories: the general sounds that will be used in all the levels, and the sounds specific to the stories.

Then I put these sounds on unity. To do this, I had to add the "Audio Source" function to some of the elements so that they could make sounds. However, the work was not finished. You must activate the different sounds at specific moments. For example, the door sound should be triggered only when the players open the door, not when they are standing next to it.

To do this, you have to go through the scripts. First, you must initialize an "AudioClip" with the "GetComponent" method. This way, we can manipulate the sounds in the script. Then, we add the "AudioClip" : these are the different audio. You just have to find the moment where you want to trigger the sound and add the "PlayOneShot" method at this place.

However, it is also necessary to put some sounds audible for all and others not. So I worked with Tim who did the multiplayer part to manage this.

C. Timothe MERLE

1. 3D Modelling

For this first defense, I was mostly concerned with the development side of our project, I didn't touch too much on the graphic side of the project. I worked as an assistant for Gabriel in his 3D modeling task. Indeed, I helped him to model the main building of Epita, the VA, and other objects necessary for the good functioning of our game. I helped him by modeling the 2nd floor of the building and the last one with the machine rooms following the plan provided by Epita, to try to have the best representation possible. For the next defenses we would like to add small details in all the building as for example small Leds of colors, a more realistic ceiling...

To save as much memory as possible, and therefore the game lags as little as possible for the players, we must try to have as few objects as possible because

each object when it is designed directly from Unity. Will have a certain number of "Vertices" and "Triangles" and the problem is that the higher this number is, the more time Unity will take to recalculate them. That's why using Blender, we can reduce the number of "Vertices" and "Triangle". So, we will have to re- designate most of our objects from Blender.

Also, to make it easier for us if we want to add elements or this kind of things, we worked with prefabs. Which allow to make a modification only on the prefab and this modification will be propagated on all the elements using this prefab.

2. Menus

i. First defense

I also made different menus at the beginning of the game to guide the player, like joining or creating a game or playing in multiplayer or solo. I didn't design these menus; I simply followed the models made by my friend Gabriel. For these menus, I didn't meet any major problem during their realization, only at the time of realizing the menu which is displayed when the players are in the lobby and wait for the beginning of the game. During the design I realized that when the group leader changed the game mode, it was not updated for the other players, so I had to rectify this.

To facilitate, the design of the new menus, if we were to create new ones like for example for settings or things like that, I went through several different ones. Indeed, I created a scene for each menu and when the player clicks on a button like for example "multiplayer" then I change the current scene with the predefined function of Unity on the scene named "Multiplayer". This system also allows to find better in the organization of the game because all the objects are not all in the same scene and superimposed on each other.

ii. Last defense

The last little element I added for this presentation, which is not necessarily very hard to do but very practical is a menu allowing the player to go back to the menu where he chooses the game mode. Indeed, if the player decides to leave the game in the middle of the game, he can click on his "Esc" key and a menu will appear. He can either return to the menu or continue playing.

3. Website

i. First defense

In parallel of what I'm going to explain you and what I have already explained to you, I developed the website on which our game is downloadable, and we quickly present the idea of the game. As for the menus of the game, I didn't design the website, I simply put in form what my friend Gabriel designed. The design of this website was fast and efficient, in fact with the expertise of my friend and myself in the field of the web, we finalized it in less than a day.

For the development of the website, we divided the tasks to go as fast as possible. Gabriel took care of the home page, the page where you arrive when you go on the site. While me, I took care of the page where we inform all the libraries that we use with unity, as well as the links towards the various social networks of each member of the group and the software that we used during the realization of our project. And I also have the page where you can see the different members of the group with a picture of each of us and a short description.

ii. Second defense

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. He

filled in on the main page of the site the different problems we encountered during the realization of our project and 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.

4. Multiplayer

i. First defense

I'm going to talk about the part that probably took me the longest to realize. Indeed, I had never made a lobby system allowing players to play with more than one person. After my friend Gabriel and I finished creating the first scenes, I asked about the different ways to set up this lobby system. Several methods were described, but the most practical and fastest to set up in the time we had was

Photon Engine. Indeed, this library available for free on the internet allows to manage the lobby multiplayer system very easily.
However, this system has a small problem that I had to work around. Indeed, as this system works is very simple. First, we must create an account from the Photon Engine website, which will allow us to create an application. This application will give us what we call a UID, unique identifier, that we will then fill directly in unity. Once this is done, we can connect the players to each other. To do this, we must first connect the player to the Photon server, then the player can either join what we call a lobby created by another player or create his own lobby so that his friends can join him and play together. The problem with Photon, is that the lobby names must be unique, which is not practical if we want several players to use the same lobby name, so to solve this problem, I generate a UID, which I define as the name of the lobby, and with which other players can join the lobby. Once all the players are in the right lobby, the group leader can choose which game mode he wants to launch and then can launch the game by pressing the start button. Indeed, there are 3 different game modes, which simply represent 3 different escape game scripts.

The leader of the group then starts the game, once this is done, all players appear on the map at predefined locations, and can control their characters. To succeed in setting up the system that allows each player to control only his character and not the one of his friends, I encountered several problems. Indeed, Photon sees each player as the same if we don't tell him exactly that they are not the same. That's why at the beginning, each player was controlling all the characters including the one of the others. So, I looked more deeply into how Photon recognized the players and their cameras. To solve this problem, I left only one camera at the beginning of the game and assigned it directly to the leader of the group in code and then as soon as a new player appears in the scene, we create a new camera and assign it to him. Also, when the players move, Photon searches in all the objects of the scene what belongs to such or such player, that's why at the start of the scene or at each refresh of the scene, I check if the current camera is the one of the players or not and if it's not the case then I block all actions.

To make it easier to know if the camera is the one of the local players (the actual player, the one who is behind his computer), Photon has set up what they call a Photon View, and which can be attached to any object of the scene. For the next defense, I must make compatible the animations of movement of the characters made by my friend Gabriel with the multiplayer mode because, indeed, currently at the time of the launching of a game in multiplayer the characters do not have animation of movement. And, to return compatible the system of vocal chat.

ii. Second defense

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 must 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.

iii. Last defense

It is on this part that I devoted most of my time to this defense. Indeed, we encountered many problems with the multiplayer as I explained just before with the Game Master. The name of the category is a bit wrong because the problem is not with the multiplayer system at all but with all the other elements of the game that should use it. When we started to develop the game without really knowing PhotonEngine and its power. And most importantly, we had developed the game so that only one player could play at a time and had not considered the possibility of multiplayer. So, I had to rework all the codes to make all the necessary parts multiplayer compatible.

The first thing I did was to allow the player to pick up items on the ground and have that item disappear for all the other players. What I was doing before was to disable the item in the scene when a player retrieved it but then the item remained visible to all the other players, and they could take it away from them which caused many problems. As for the Game Master, I used functions that are called in the network by a player. For example, when player number 1 retrieves the item on the floor, he will send a message to the Photon server to say that such function must be called, and the Photon server will call this function on the computer of all the other players present on the room. Thus, the item disappears in the scene of player 1, player 2, and player 3. In the same way, when a player decided to put an item on the floor, it was not updated in the scene of the other players. So, I used the same system that tells the Photon server that such and such a player has put an item on the floor and that all the other players should put such and such an item on the floor. The only problem with this technique is that the data are sent on the network and must be transformed into bytes to be transferred on the network, but some types of objects can't be directly transformed, and I had to trick them. To overcome this problem, I used tags and names. Indeed, a tag is an attribute that an object of the scene has and that allows us to recover objects very easily.

Another thing I had to fix was the disappearance of doors. When a player approaches a door and has the key to open that door, a message appears in the bottom left corner of the screen telling them that they can open that door. If he

clicks on this button, then the door must disappear to let the player through. However, we didn't consider that the door should also disappear for all the other players in the scene. So, in the same way, I created a function to solve this problem. That's pretty much for the different problems we had with the multiplayer system.

5. Four-in-a-row

To add a little challenge to the players, they will have to face artificial intelligence in several mini games such as the four-in-a-row and beat them to get the clues. For the design of this game, I first developed it in a terminal to visualize it and then thought about an artificial intelligence. The artificial intelligence is not yet ready for this defense, but I am actively working on it. The method I'm going to use is the following: I'm going to take each square of the game board and transform it into a matrix of numbers that will represent the percentage of each square. Indeed, we will determine for each square a percentage or a power to know in which square it is more interesting to play the next move. This percentage is calculated according to the pieces that surround it and therefore the probability of winning.

6. 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 must 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 Figure 9). 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 10). 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 9). 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.

7. 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 must 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.

8. Game Master

i. Second defense

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.

ii. Last defense

I didn't have to do much for this part. Indeed, my classmate Gabriel did most of the work. He thought about how to create this Game Master and how to set it up. This one was very functional for players who decided to play solo. However, since I was the only one who set up the multiplayer system, he didn't know much about it. So, we had to go back behind a few lines of code. The way the Game Master works is simple, no player can do a puzzle if the puzzle that precedes it has not been done before. The main problem is that if a player does a puzzle, then it must be considered as done by all the other players, otherwise, none of these other players will be able to do the next puzzles. To do this, we have once again used Photon and the different services it provides. Indeed, thanks to Photon you can make requests to other players connected in the room. Then you just have to make a function that will be called when such or such message is received. For you to understand better I will give you an example. To know all the puzzles that the players have to do to win we store them in the form of a list and the first element of this list represents the current puzzle. So, when the players perform the puzzle in question then it must be removed from the list of puzzles. So, I

created a function that will remove this puzzle from the list and this function is called by all the players when one of them solves a puzzle. If the puzzle in question is to retrieve a key, then as soon as one player retrieves the key in question, the puzzle will be removed from the list of puzzles by all players. As for this part, we've done the trick.

9. Mini games

i. Second defense

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.

ii. Last defense

For the previous defenses, my classmates had created several mini games. For the first defense, my classmate Alexis had developed in a separate scene the game of teasing. However, this game could not be directly integrated into the scene in which the players can play so I had to make some modifications. And so, for this defense, two other mini games were developed Asteroids and Space Invaders respectively developed by Alexis and Gabriel. But the same way as for the game of the tease they both developed the games in a separate scene and so I took care of integrating these games into the scene of the players. This time it was easier because I didn't have to modify anything. Indeed, as I explained above, for the game of the tease I had to add two or three elements and modify the code but for Asteroids and Space Invaders I used another method. Indeed, unity provides us with what is called multi-scene, which allows loading several scenes at the same time. So, when a player launches one of his mini-games I load the scene that contains this mini-game and I define this scene as the main scene. And to avoid any problem with the cameras I disable all the elements of the old one. The scene is the one where the player can play. And depending on whether the player manages to finish the game or dies during the game I update the Game Master. For these mini games there is no notion of multiplayer except when the player finishes the game but otherwise, the game is personal, all the players can play on their side and if one of the players manages to finish the game then nobody can play the game anymore and all the players must move to the next puzzle. The main problem I encountered was when I tried to disable all the elements in the game scene to have no problems. When an element is inactive

then it can't be recovered thanks to its name or its tag, so I had to use another method to deactivate and reactivate the different players and elements of the main scene. To do this I deactivate all the players and the elements and then I launch the new scene. Then when the game is finished the scene is reloaded so I can reactivate all the elements.

10. 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 must 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.

11. 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.

12. Options

This category was not necessarily planned in the schedule but seeing that we still had some time before the defense we decided to add some extra elements to the game. And the options are part of these elements. Indeed, we found it interesting to let the players decide their key depending on whether they have an AZERTY or QWERTY keyboard or even if they have preferences. To do this my friend Gabriel created the menu design and layout and I took care of the technical part. I did not encounter any problems during the development of the options. For the operation, so that the player only sees the keys update and doesn't necessarily understand that I'm using a text box that turns on and off, I disabled all the texts and only use the placeholders. Once the player has changed an option, if he wants it to be considered, he must click on the "Save" button which is located right next to the option in question. And once he has clicked on this button, the whole game is updated to consider these new options.

IV. Our joys

1. Gabriel TOLEDANO

For the past 5 months we have been working hard on this project to make it the more playable and enjoyable for the different players. I am happy that we have succeeded in doing what we worked hard for. Moreover, working on Unity as much as we did during this project really made me understand how it works. Of course, some very specific parts are still very complex for me, however, I managed to understand most of the software. In fact, at the beginning I did not understand basic aspect such adding public fields to a script to directly put game object from Unity and not having to link them thanks to the script. At the end, I

could just think to something I wanted to do and then try different approaches until I found the best way possible to do it. Most importantly, without looking anywhere to do it. Achieving such level in Unity really made me like the project more and more.

Of course, this was directly linked to my skills in C# that have also improved during this period. It is definitely due to the different TPs we have done in class but also all the experiments I did in the Unity playground, a land where everything is possible. Moreover, developing an entire game with this language but more specifically an object-oriented language made me realize how great this invention was. In fact, managing everything thanks to attributes, objects and classes made it really easy to link the Game Master to the game for instance. A task that sounded more harder than what we finally had to do.

Another technology that I enjoyed using was git. In fact, this tool is useful to work as a team. Not only did I learn to manage the basics of it but also how to use branches, reset commits and mostly manage merge conflicts.

This is directly linked to working as a team which was a nice experience. In fact, during our scholarship we had the opportunity to work as a group for presentations but never for an IT project. In fact, it is very different because the tasks need to be given more precisely so that nobody crushes the code another made before for instance. Moreover, I was designated as being the project manager, this was a first for me but thanks to my perfect team, my work was very easy. Everybody worked so that he had finished his tasks in time.

A joy that could be seen as obvious is the joy of creating something that works, not only the final product but also all the little steps to get there. As developers we know how frustrated encountering a bug can be, but we also know how joyful it is to fix one.

2. Timothe MERLE

As far as my joys are concerned, I think that the project has been instructive and very nice to develop. Indeed, even if we didn't get the project we wanted, we were all convinced by our game idea and so developing it was a lot of fun, well, in the main lines because there were some problems. Moreover, the fact that we could choose our colleagues, helped us a lot because we all know each other, and know what such and such is capable of and therefore we went very fast on the distribution of tasks. Even during the development of the game, it's very frustrating because we have a lot of functional parts but each one on its own and so we don't see anything that is progressing and being done but, in the end, once everything is well connected it's very satisfying that several hours of work gives something concrete that works. The fact that we were free and that we had to give accounts like reports or defenses was also very interesting because it allowed us to write a lot and to get familiar with the oral presentation in front of a jury or at least someone who judges our creation. This is something that can happen very often in the engineering world.

3. Alexis PINSON

The group work was an aspect of the project that I enjoyed. At first, I found it quite complicated to work together without supervision, but we were well organized and there were no problems on that point. So, I found it quite pleasant to work on a rather complex project while being totally free and autonomous.

Another joy I discovered on this project is the joy of having accomplished something that works. Indeed, in class, we code but with no real goal other than to answer the assignment of the exercise. Here, when our code works, we can see how it is useful for our project and this gives us a huge feeling of accomplishment.

V. Our sorrows

1. Gabriel TOLEDANO

Even if the project was very interesting, and very joyful, the darkness always hides itself in the bright light. In fact, making a game in Unity is synonym of multiple bottles of EPITA student's tears. As a team we went through a multitude of bugs that were for the biggest part caused by us but sometimes the external tools were the sinners. Meaning, to fix them we had to go in the settings given by Unity to modify how tools were handled by Unity.

Another side effect of using git as a team is constantly fixing merge conflicts or not having the possibility to work in a Unity scene without crushing others work.

Then, since our game needed a multiplayer mode, we had to develop the methods for it. Nevertheless, when I started to code, I did everything as if the player was the only one in the game. This makes it very hard to fix once we realized that developing a game should be done in the opposite. Meaning, we should have made the game in multiplayer and then a solo version from it.

2. Timothe MERLE

My biggest pain is the fact that our first project, artificial intelligence that can solve puzzles thanks to a photo and that could even guess the missing pieces, was not accepted. I have some other sorrows, notably the fact that we were forced to develop a video game and therefore to use unity. And having already developed video games and therefore using unity, I knew all the issues that unity caused. In particular, the numerous crashes. More than once I worked for several hours and unity crashed from one second to the next and corrupted my whole project and I had to start from scratch.

3. Alexis PINSON

What affected me the most in this project are the bugs. Indeed, after having had the impression to do everything well, to have tested everything, when comes the final test, the code compiles but does not work in the way I wanted.

Let's take a concrete example. The part I had the most problems with because of the many bugs was the implementation of the door in a separate scene. At first, I tried to do it by myself: using the different unity tools, I linked the door to a script. In this script, I coded so that when the user presses the 'F' key, the door opens by rotating on the axis opposite the handle. However, I had a lot of trouble with it that I couldn’t handle. So, I decided to watch a tutorial on the internet. Indeed, making a door on unity is something that is well done. It is not difficult to find a tutorial. I looked at some tutorials to find the one that best fit my expectations. I found one I liked, so I started copying it. However, the door was not operational because there were many elements already present in the scene that prevented it from functioning. So, I tried to fix all the bugs and it took me a lot of time. Debugging time is for me the most frustrating time long.

VI. Conclusion

In conclusion, this project, LockedUp, is a virtual escape game and our team, Jupitr., hopes to have brought it to a level where players will be able to live the experience close to the real thing. Thanks to the different artificial intelligences present in the game and the multiple modes, players will be able to dive into the heart of the action. Moreover, our organization has allowed us to perform more tasks than initially planned, which enhances the experience of the players. Thus, thanks to our game, any human will be able to immerse themselves in EPITA to discover how much sleep is overrated. In fact, either you work, or you sleep.

Thank you for reading, we wish you a good holiday.

VII. Appendix

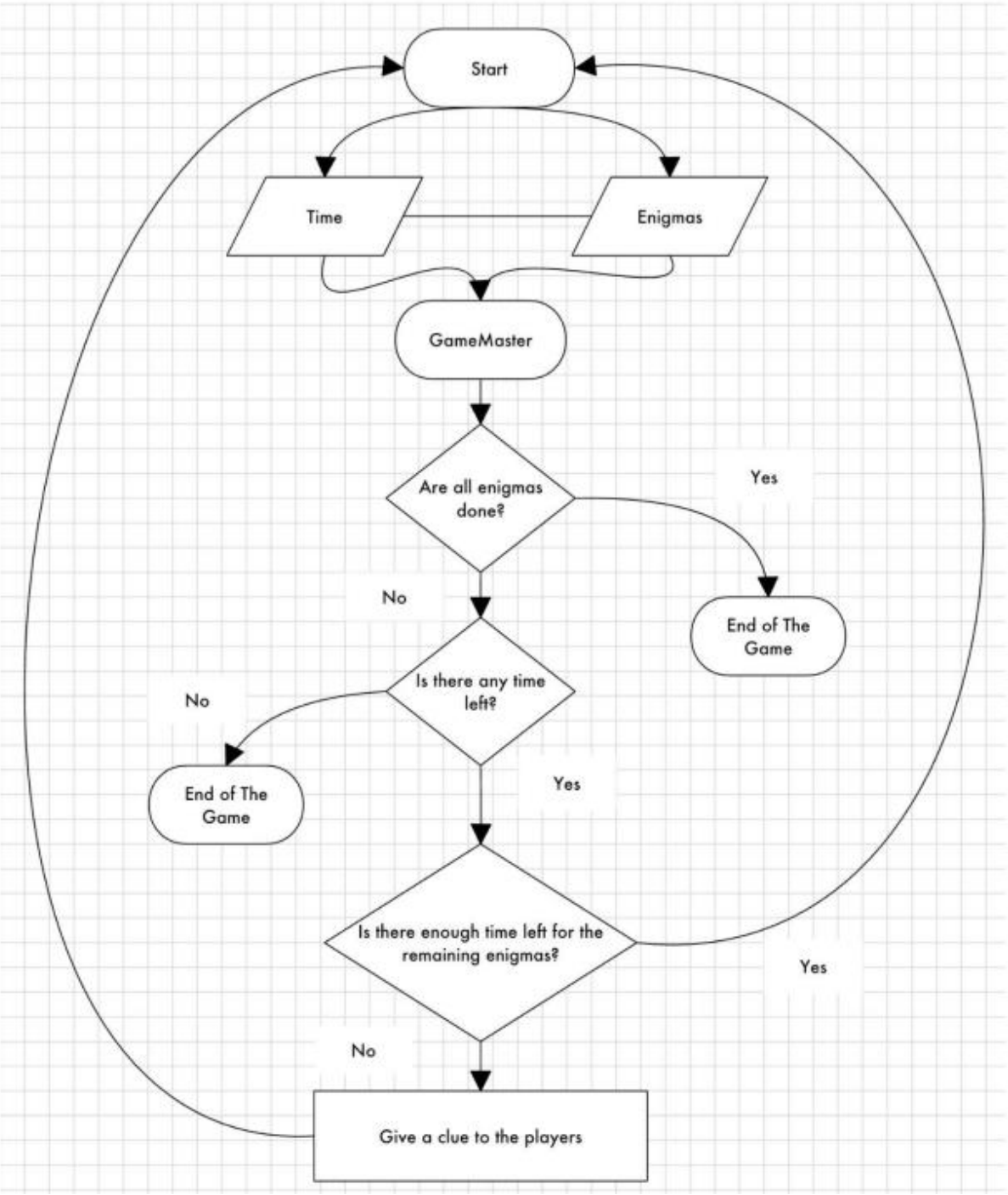


Figure 1: Decision Tree for the Game Master

Figure 2: Night at School Environment

Figure 3: School Psycho Environment

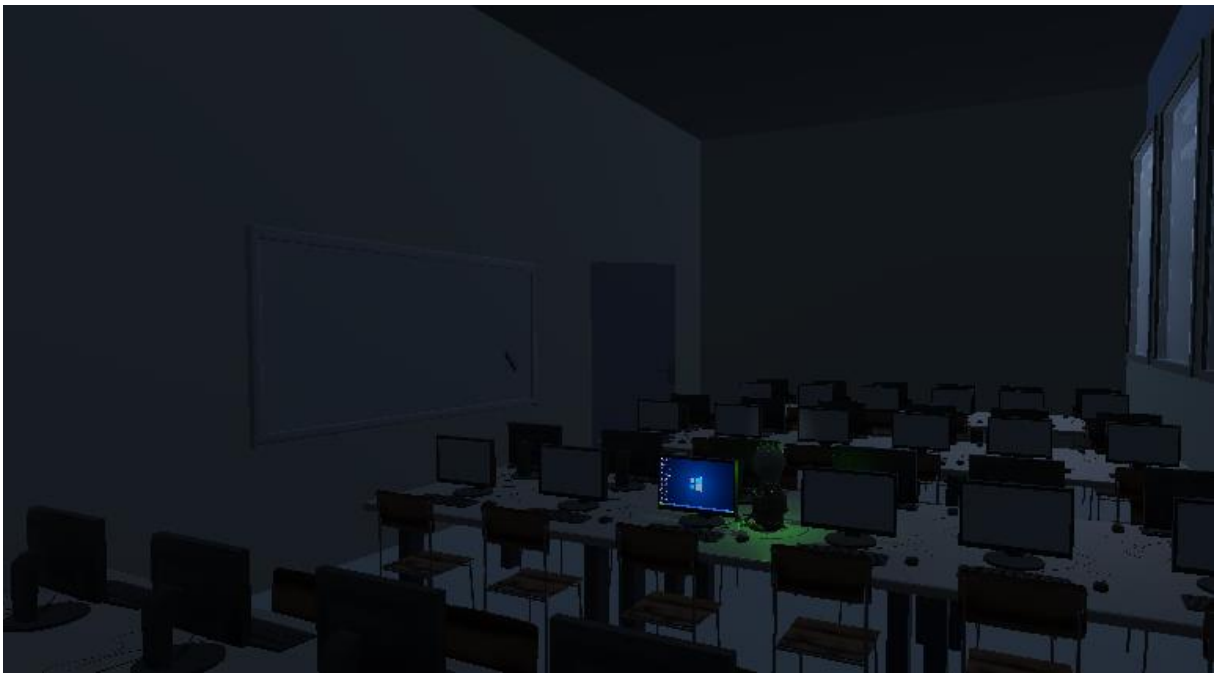


Figure 4: Whisper of Darkness Environment

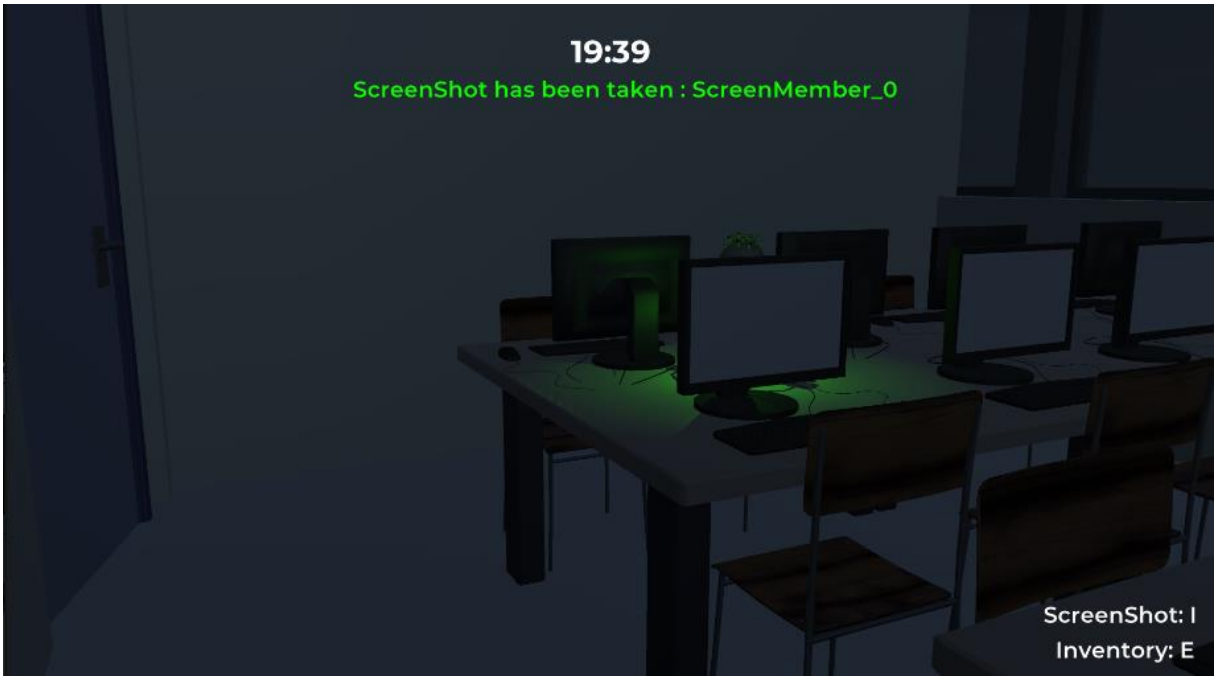


Figure 5: Screenshot Message

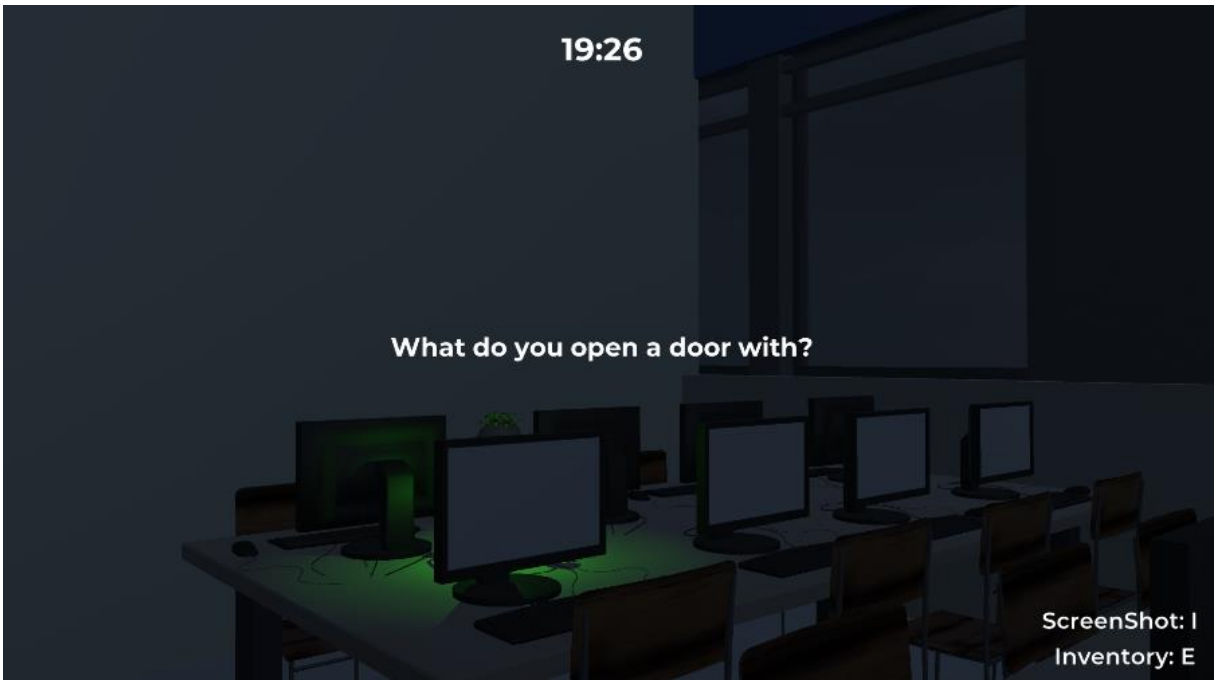


Figure 6: Game Master showing a clue

One NPC giving a clue

Figure 7

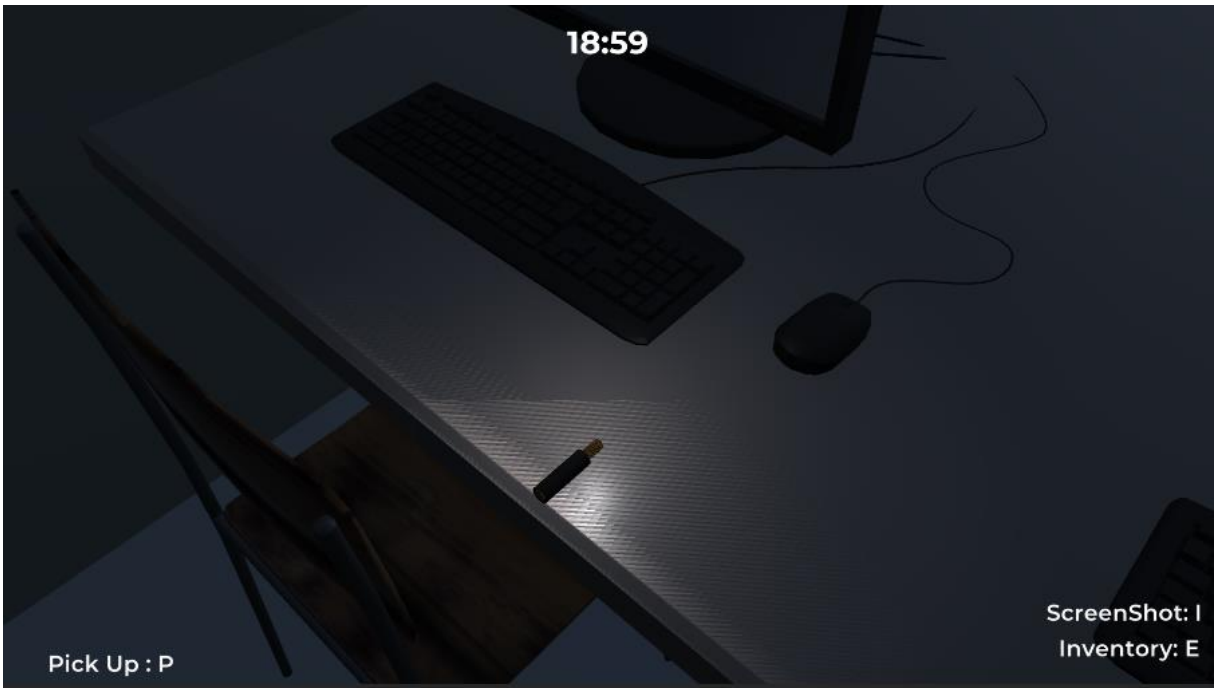


Figure 8: Item to pick up (see the bottom left of the screen)

Figure 9: Player's inventory

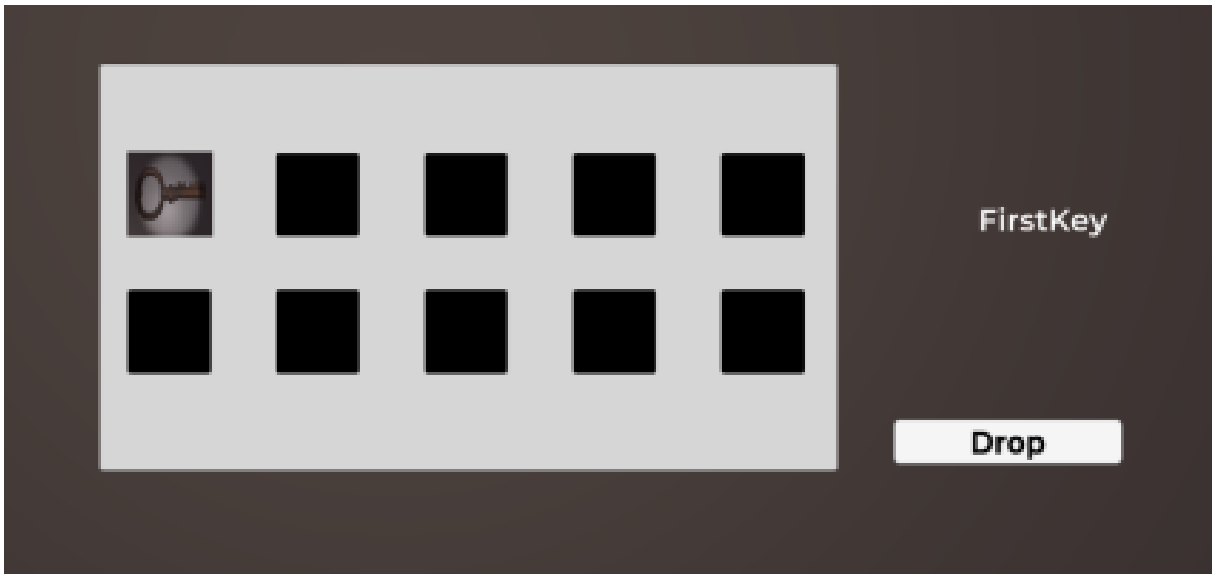


Figure 10: When clicking an item, the possibility to drop it, is displayed

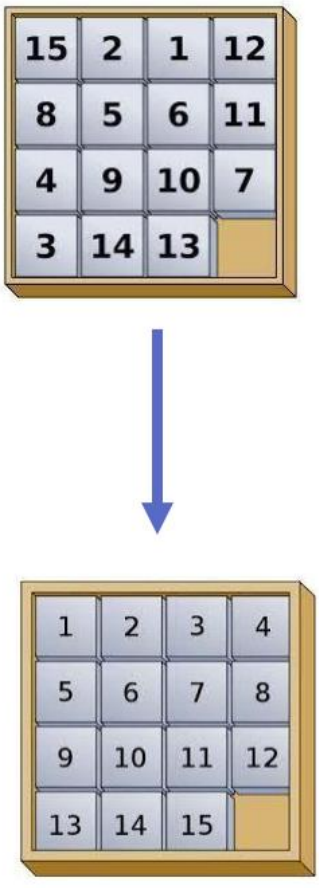
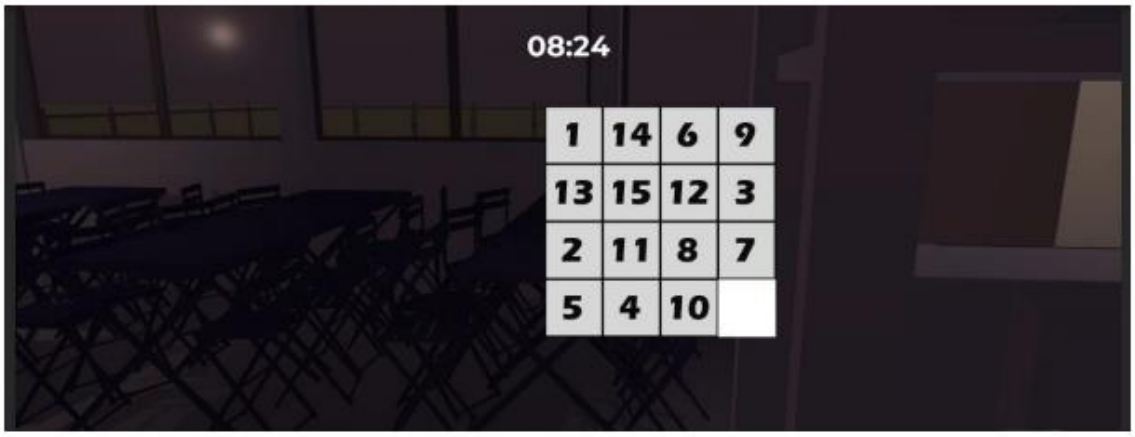


Figure 11: "15 Puzzle"



15 Puzzle game in-Game

Figure 12

Figure 13: Asteroids Mini Game

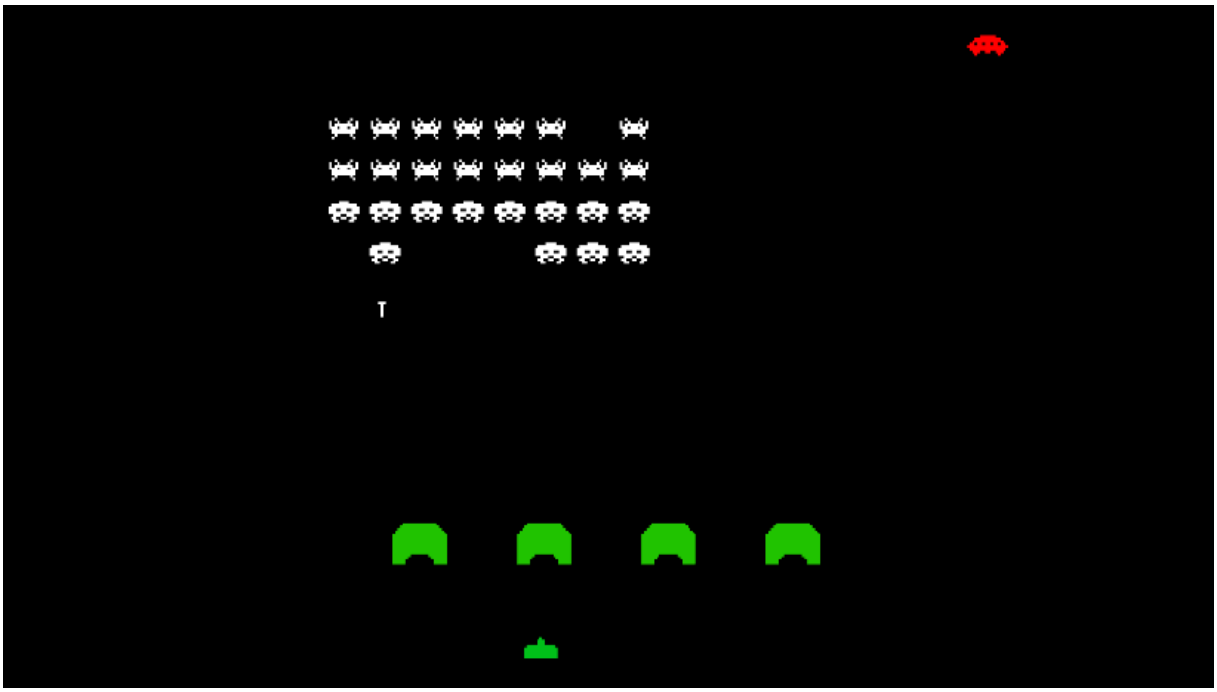


Figure 14: Space Invader Mini Game