HEAD4CHE PRODUCTION COMPANY

S2 PROJECT

2nd defense report

SUBWAY OUT

A1

Promo EPITA 2029

Jeanne Thalie DURAND

Tidiane BATHILY
Adam GRAZIANI

Arthur FIOLET
Valentin KIEP

Contents

I. Introduction

This document is an overall summary of the work done during the last 2 months, taking into account the advancements but also the delays in the creation of the game SubWay Out by the Head4che Production Company. It will explain the various issues encountered and the solutions found to solve them.

SubWay Out is an exciting escape game where the player is confronted with their ability to solve problems and puzzles in a familiar environment: the one of the subway. Understanding what happened and finding a way to leave the subway is their main mission.

To progress as fast as possible in the development of the game, the work was split into different subtasks, allowing the team to be more efficient.

II. Task Distribution

TASK DISTRIBUTION						ADVANCEMENT		
	Jeanne	**Tidiane**	**Arthur**	**Adam**	**Valentin**	B2	B3	B4
DESIGN								
Scenario	M			A		80%	95%	100%
Level design	A			M		65%	95%	100%
PROGRAMMING								
Gameplay		A		M		5%	60%	100%
Game engine		M	A			50%	85%	100%
Website			M		A	15%	50%	100%
GRAPHICS & SOUNDS								
3D Modeling	M		A			40%	80%	100%
Visuals		A	M			10%	40%	100%
Sound		M			A	0%	30%	100%
COMMUNICATION								
Branding	A				M	80%	90%	100%
Trailer				A	M	0%	10%	100%

Table II..1: Task distribution table

M : Manager / **A** : Assistant

B2: 13/01/2025
B3: 10/03/2025
B4: 26/05/2025

III. Project advancement

1. Puzzles

a. Hanoi Towers

Hanoi Towers was the first puzzle for which implementation started. It has undergone three iterations overhauling the physics logic used in the movement of pieces, with a fourth one upcoming. Version 1 was finished for the first defense, and the puzzle's premises have been covered in the last report. This section will highlight the challenges encountered, the tools developed and the workarounds found during the developments of phases two and three, as well as giving a preview of what to expect in version 4.

Hanoi v2

The second version of the Hanoi Towers introduced two main features: models and a plane locking system. In this version, the balls used as placeholders were replaced by more complex and visually distinct pieces, shaped in such a way that large pieces can not fit onto smaller ones, while all small ones can fit onto the larger ones. These new models brought a new challenge with them. The shapes of the pieces could no longer be used to ensure they were kept inside the tracks: the player could remove them from the box. To counter this effect, a first system was created to ensure the pieces were locked onto the plane the puzzle takes place on. It used a four-step mathematical approach to determining the velocity vector to apply to the piece.

1. Computing the vector $\vec{n}\begin{pmatrix}a\\b\\c\end{pmatrix}$ normal to the plane the game is played on, obtained from the quaternion rotation q of the Hanoi Towers prefab: $\vec{n} = q \circ \begin{pmatrix}0\\0\\1\end{pmatrix}$.
2. Computing the parametric equation of a line R spanning from the player's camera (x_A, y_A, z_A) to its grab point (x_B, y_B, z_B) located in front of it.

$$R : \begin{cases} x_A + t(x_B - x_A) \\ y_A + t(y_B - y_A) \\ z_A + t(z_B - z_A) \end{cases} \quad t \in \mathbb{R} \tag{III..1}$$

3. Computing the intersection I between the plane and the line. This is done by finding position t_I of the intersection with the piece (x_T, y_T, z_T) on $\mathbb{R}$. $t_I = -\frac{ax_A+by_A+cz_A-(ax_T+by_T+cz_T)}{a(x_B-x_A)+b(y_B-y_A)+c(z_B-z_A)}$
4. Setting the piece's velocity to a vector going from the piece's current position to the intersection by injecting t_I into the equation of $\mathbb{R}$.

These calculations were run every physics tick (= 0.2 s) when a HanoiGrabbable object was being held. Since the system was mathematically complex, debug tools were created to aid development, namely a toggleable render of the actual plane and line obtained from the equations.

The second version of Hanoi had multiple issues. First of all, the collisions did not match the models. This is covered more in depth in the Models section of this report. Furthermore, the extensive computations used to compute the pieces' movements caused lag spikes which would result in the pieces being pulled out from the bounds of the puzzle area.

Hanoi v3

A third version of Hanoi was developed to address these issues. The fixed models were implemented in this version. To improve the performance issues, a new much simpler mathematical approach was devised:

1. Computing the regular velocity vector $\vec{v_0}$ used by the generic grabbing system, which corresponds to the vector going from the piece's position to the player's grab point.
2. Doing the dot product of this vector and the vector $\vec{n}$ normal to the plane the game is played on. This gives the component $v_n = \vec{v_0}.\vec{n}$ of the velocity vector in direction perpendicular to the plane.
3. Subtracting a vector with a norm v_n in the direction of the plane to the initial velocity vector. This gives the projection $\vec{v}$ of $\vec{v_0}$ on the plane the game is played on.

The final computation is $\vec{v} = \vec{v_0} - (\vec{v_0}.\vec{n})\vec{n}$. A new debug tool was created to replace the two previous. It creates an arrow that directly represents the path the piece is trying to take. It can be toggled on and off via the debug console.

The two above improvements fixed the relevant issues, but final problems still remain to be fixed in the upcoming v4:

- Unity's center for objects does not match up with the models', which can cause a discrepancy between where the physics system tells a piece to go and where its collision can actually pass through, sometimes blocking pieces.
- Fast-moving objects can clip through the rails. A detection system will be implemented to immediately bring the piece back if this happens, without breaking the player's control.
- Not all data relative to the pieces' positions in the game's internal grid is synchronized between clients.
- The rails have sharp turns that make moving the piece in the desired positions sometimes tricky. A new custom track will be modeled to address this.

b. Hint System

Being an escape game, SubWay Out has extremely linear gameplay. All puzzles must imperatively be solved to complete the game. This makes it possible for the player to get stuck at some point during the game. To counter this, a hint system has been implemented. It is not available since the start but can be activated as soon as the game begins. The hint activation puzzle will serve as the game's tutorial.

Activation Puzzle

The hint system consists of an emergency call booth and its trigger. The trigger starts on the floor, broken off from the booth. In order to activate the system, the player must learn the four abilities of the game:

- **Moving** around to get to the call booth (which will be highlighted for the purpose of the tutorial)
- ***Zooming*** *to see details better* (optional, to be implemented)
- **Grabbing** the broken trigger (highlighted for the purpose of the tutorial)
- **Interacting** with the box while holding the trigger.

Once the player successfully inserts the trigger into the box, the tutorial is complete. The player is from then on able to request hints about the game.

Hints

An important part of designing the hint system was ensuring it did not break immersion too much. It was decided the hints would be vocal recordings played depending on context. These recordings are, in the game's story, live advice from Lola, an assistant at the company that operates the train network.
Internally, the players' progression is tracked by a list of available hints that is updated dynamically as puzzles are discovered and solved. Requesting a hint plays a random hint from the available list. This is to ensure the player is not constantly stuck on the same puzzle while others are available (see appendix). Because the hints are randomly played, the player will be able to listen to the hint again at some point.

Figure 1.1: Screenshot of the emergency call box in-game

2. 3D Models

The modelization of all the game elements was a challenging part as it was necessary to think about every element that the game needed. SubWay Out has to be a realistic game with accurate and detailed scenery that contrasts with its humoristic characters.

a. Subway

The main parts of the game environment are the subway cars. The head of the train, in which the driver's cabin is located, is the only playable car. The cars attached behind it are non-playable and only used for the scenery.

Non-Playable Cars

The model of the subway shown in the last report became the design for the non-playable subway cars. The 3D model of the door at the back of the car has been reworked in order to prevent the player from glitching through it and escaping the subway without solving any of the puzzles. Some textures were also changed, noticeably the windows. It was too transparent and thus not realistic enough. The car is duplicated in order to obtain a train of 2 non-playable cars. A coupler between the subway

cars was also added. A coupler, also called a coupling, is a mechanism used for railway vehicles or subways in order to connect cars together. It is located at the extremity of each car.

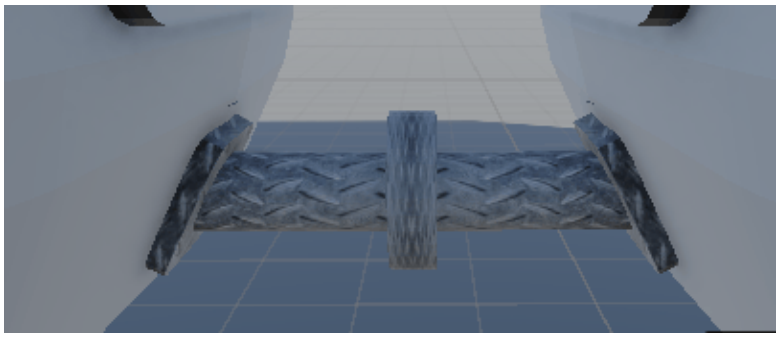

Figure 2.2: Coupler between two subway cars

The non-playable cars were implemented into Unity 3D without any major issues. A simple file conversion to FBX, with embedded textures was enough to put the cars in Unity.

Playable Car

The model of the playable car is separated into two subparts: the passenger part where the player begins their adventure and the driver's cabin where the player solves the final puzzle.

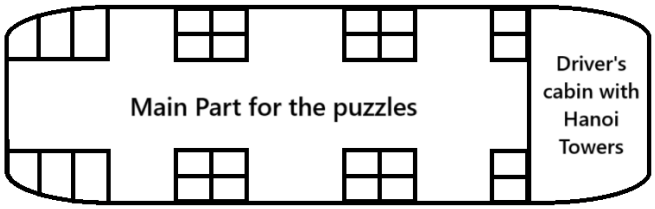

Figure 2.3: Layout of the playable car

The driver's cabin contains a dashboard, control buttons and some screens to add details. In the passenger section, as windows will be used in a puzzle, they are now cut in several interactable sections. However, for the moment, the windows do not have collisions, allowing the player to throw objects through them.

For both types of subway car, lights produce the mysterious and oppressive atmosphere that we desired.

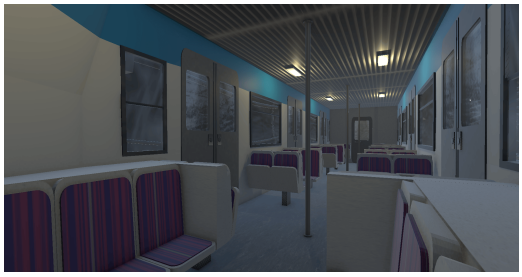

Figure 2.4: Playable car

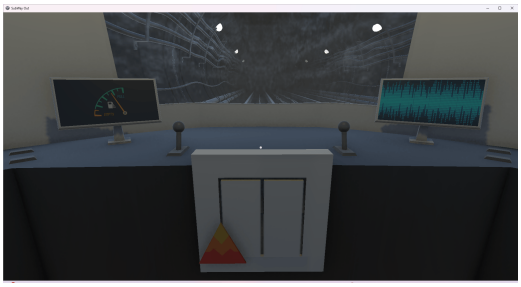

Figure 2.5: Driver's cabin

b. Tunnel

Even if the subway is the most crucial component of SubWay Out, the backdrop, in our case a subway tunnel, is also an important part in the creation of the atmosphere of our game. The tunnel is the only model imported from an external source, in the public domain. To strengthen the ambient effect, dim lights and ominous fog were added to the model.

c. Hint System Models

Another model designed and added into Unity 3D was the box for the hint system. Custom textures were designed to clearly indicate where and how to find help during the game.

Figure 2.6: Texture for the hint system box

To go with this hint system box, a trigger can be inserted in the box. While importing this trigger into Unity, an issue occurred. The trigger handle was not correctly displayed and some meshes were invisible. This was a backface culling issue.

Indeed, in Blender, the open-source 3D graphics software that is used for modelization, the normals of the faces can be calculated from the outside or the inside, and are then either rendered or ignored. For the trigger, some normals were calculated from the inside and, as a consequence, not shown when imported. Therefore, recalculating the normals outside for the faces that were missing was the solution to the issue and then all the faces were visible.

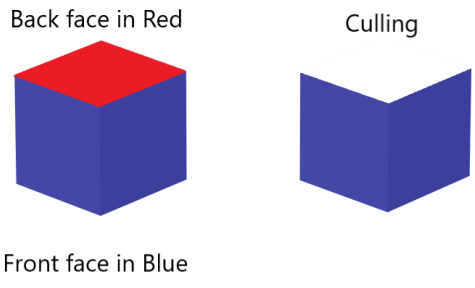

Figure 2.7: Explanation of the backface culling issue

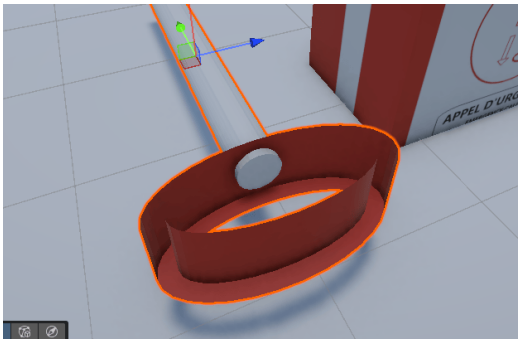

Figure 2.8: Backface culling issue for the trigger

d. Hanoi Puzzle Modelization

The pieces of the Hanoi Towers puzzle consist of a ball, attached to a cylinder, that is itself attached to the main section of the piece. The pieces underwent several iterations to fix various issues in their implementation inside the puzzle.

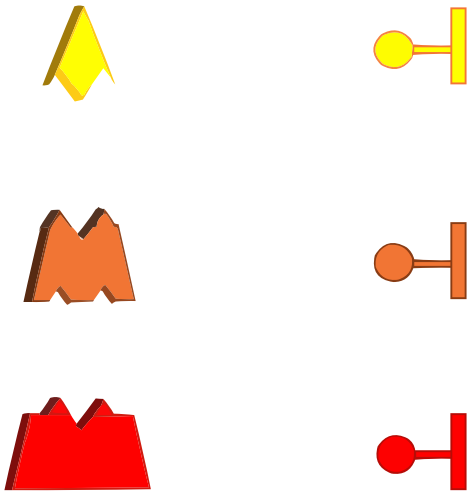

Figure 2.9: Concept "art" of the Hanoi pieces

The main problem was the mesh colliders for the different parts of the pieces. Indeed, they were not applying well due to the different convex meshes of the pieces.

In the following development images, the green lines represent the shape of the mesh collider for the orange piece: the collider does not take the shape of the piece. As a consequence, when the pieces do not fit in the puzzle. This issue was fixed by cutting the pieces into subparts, allowing the collider to take the proper shape of the orange piece.

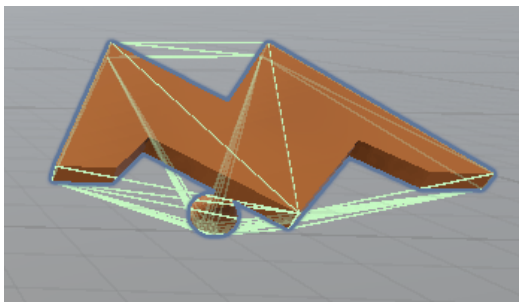

Figure 2.10: Before the cut

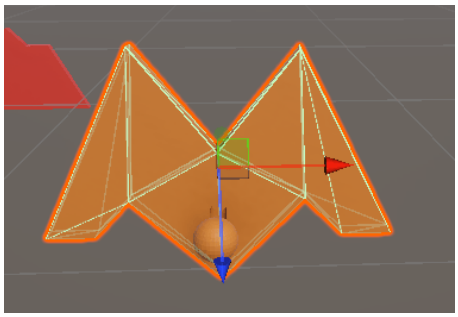

Figure 2.11: After the cut

Despite this, more work needs to be done on these models.
The cylinder is placed at the center of the main section, but, the centers of objects in Blender and Unity being different, this is not the case for the current models. Another problem is the limit of authorized polygons in Unity. A convex mesh must have 256 polygons at most. Although the pieces are small, the ball currently used is too high-poly.

Figure 2.12: Unity warning for high-poly convex meshes

e. Characters

The characters of SubWay Out are designed in a humorous style, similar to the Miis of Nintendo. Creating a model matching the style envisioned by the studio was a challenge. Different variations of the models were created before settling on a final one. For instance, earlier player models were more square and low-poly...

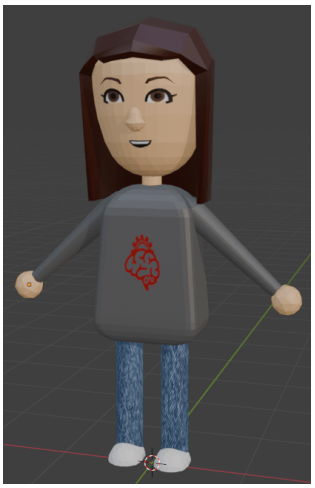

Figure 2.13: In development model

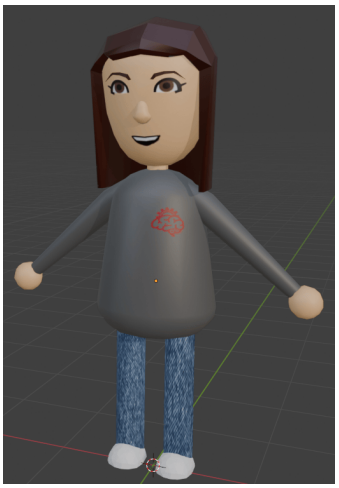

Figure 2.14: Final model

The character was also difficult to make because of its hair. It was important to keep a low-poly style but, at the same time, the member of the Head4che Production Company represented by the model had to be recognizable.

3. Animations

Working on the animations of SubWay Out was one of the studio's objectives. In the last 2 months, the studio worked on various animations, used at different stages of the game.

The game takes place in a moving train. To simulate its movement, the 3D model of the surrounding tunnel is moving endlessly. This allows the player to experience the same feeling as a traveler in a moving subway. While trying to reproduce this impression of movement, succeeding in finding the correct pace at which to play the animation was something that needed to be taken into consideration. If it was played too quickly, the tunnel would not have been visible at all, thus ruining the efforts taken to put in place the detailed environment. On the other hand, if it was played too slowly, the illusion would have been broken.

Moreover, the door to access the driver's cabin in the subway is animated. When the player clicks on the handle, the door opens. Figuring out which was better between a boolean activation or a trigger was complicated at first. After testing both solutions, working with a boolean activation was the easiest and most understandable way to do it.

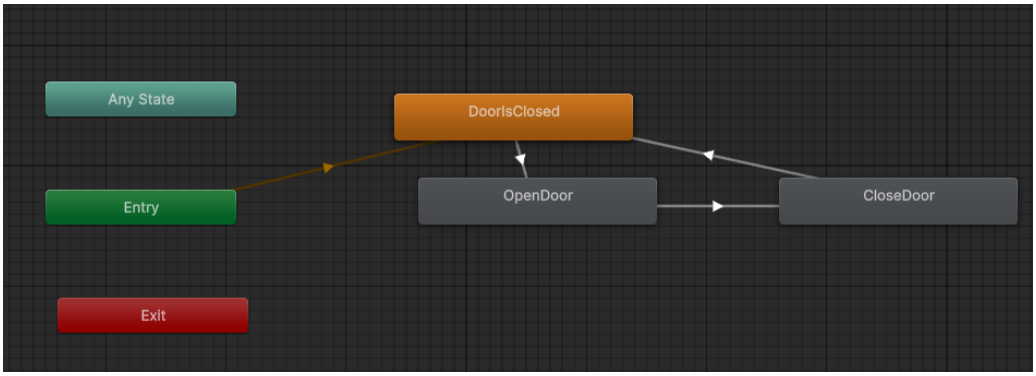

Figure 3.15: Animator for the door animation

The above figure shows how the animation of the door works. When the game begins, the state of the animation is set to `DoorIsClosed`. The boolean representing whether the door is open or not is set to `false`. When the player interacts with the handle, the boolean becomes `true`. This activates the state `OpenDoor` and the animation. If the handle is clicked again, the boolean switches back to `false`, the state of the animation is brought to `CloseDoor`. The animation is played with a speed of -1 so that is reversed. The state then instantly switches to `DoorIsClosed`, allowing the loop to be performed again.

Finally, the trigger used for the hint system has an animation too. When it is inserted into the hint box, an animation is played. In addition, each time the player pulls the trigger to receive a hint, it goes down so it is easy to understand that the emergency trigger was successfully pulled.

4. Networking

The multiplayer part of the game being almost fully built, we started working on the networking around it. Essentially, what is called networking in our game is the system of peer-to-peer connection, and how clients interact and are synchronized through the network.

a. Session System

It was decided to use a client/host configuration. First, the host starts what is called a session, allowing them to choose between three actions next: they can either stop the session and return to the main menu of the game, start to play alone, or wait for a friend to join their session to play together. The other player must then be able to join the session of their friend, and afterwards they will have to either wait for the host to start the game or leave the session whenever they want.

b. Peer-to-Peer Connection

A non-functional interface had already been shown in the last technical report. The first change made was a redesign of the user interface, splitting the multiplayer menu into two simpler ones: a menu for the host (also used as the single-player menu) and another one for the client.

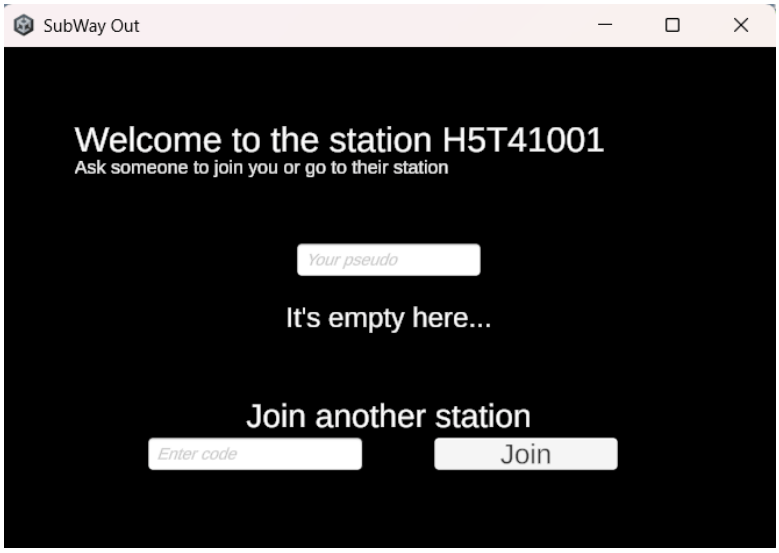

Figure 4.16: First Multiplayer Menu

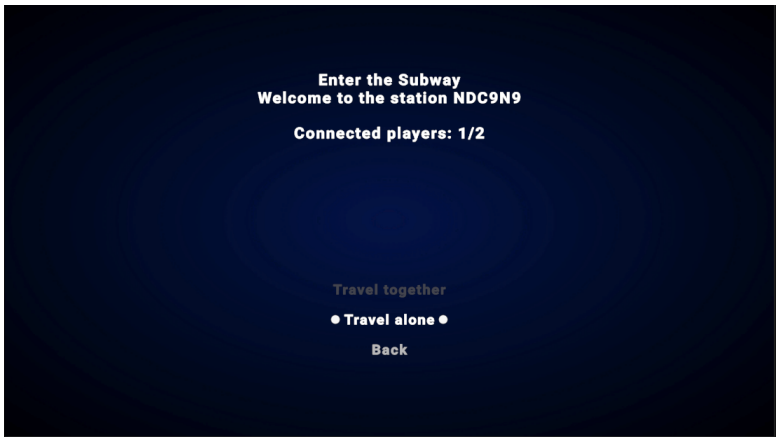

Figure 4.17: Start Menu

Figure 4.18: Join Menu

The first version of the networking system was very simple. The multiplayer interface displayed an 8-digits code, which was simply a base-34 representation of the host's IP address, using digits and letters without the letter O and the digit 0 to avoid confusion. The client would use the IP obtained from the code to link the two instances of the game and play together. However, this system caused a lot of trouble because of the necessity to open ports on the host's computer and network, which was too technical to set up on a machine.

A second version of the networking system corrected this by using Unity Solutions, a package to ease the implementation of multiplayer. Two parts of this package were particularly useful:

- **Unity Lobby** allows for simpler join codes and methods to link two instances with ease.
- **Unity Relay** enables relay servers that act as intermediaries between the two clients. No logic is processed in a relay server, its one and only purpose is to forward the requests from a player to another. This prevents issues caused by closed ports. Therefore, it significantly simplifies the process of connecting players.

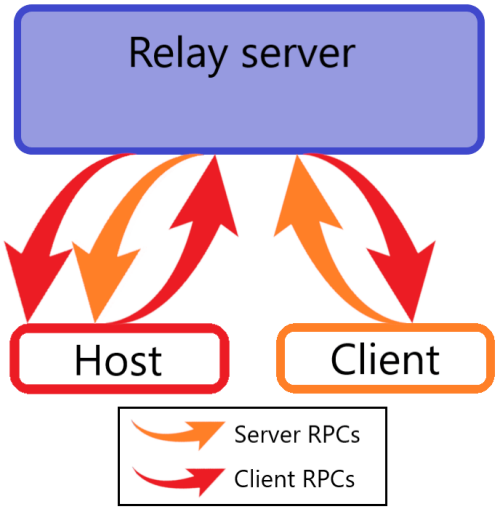

Figure 4.19: Interactions between clients and the relay server

The networking system is now complete. However, it is not entirely the case for the multiplayer experience in general, as sessions, network objects and coroutines are currently not properly reset.

c. Synchronization with Netcode for GameObjects

Unity's Netcode for GameObjects is a Software Development Kit (SDK) used in this project to manage GameObjects' workflows over the network. It is host authoritative based, meaning that all `NetworkObjects` are owned by the host and have authority over spawning and despawning these.

In this project, two methods of this SDK were used to synchronize game states and events on the network: Remote Procedure Calls (RPCs) and `NetworkVariables`.

- **RPC**s are used to call methods on objects that the host owns, like applying a force on a grabbed object.

- **NetworkVariables** are used to synchronize values between clients, like if an object is currently grabbable.

These methods are commonly used in this project to communicate between the host and others clients during the game.

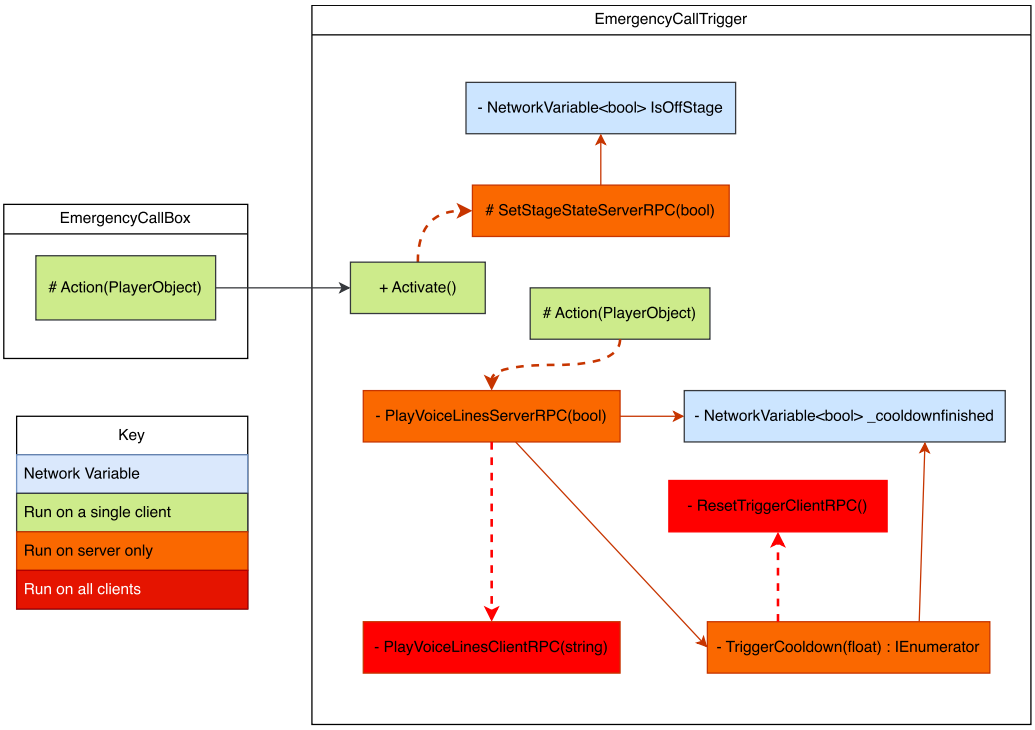

Figure 4.20: Example of RPC and NetworkVariable usages in a puzzle

5. Interactables Objects over Network

SubWay Out is an escape game that relies entirely on players' interactions with their environment. The PlayerInteraction system has been reworked, by adding actionables objects and changing when the system is triggered.

When a player presses a button on their mouse, a ray is cast from the player's camera to detect collisions with objects of the current scene. Depending on the button pressed, current conditions and encountered objects, the system can have one of these behavior:

- Interact/Action an `ObjectActionable`, with left click, or
- Grab or drop an `ObjectGrabbable`, with right click

Before this rework, a ray was cast every frame, which caused performance issues due to raycasting's high costs. Now, rays are only cast when the player presses a button on their mouse.

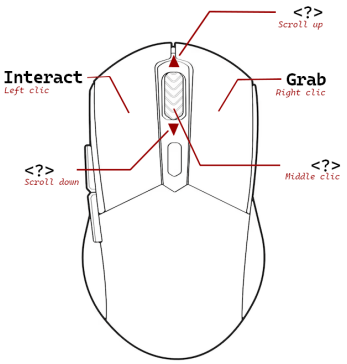

Figure 5.21: Available controls on the mouse

a. Actionables Objects

`ObjectActionable` is a generic term to define interactible objects that have a fixed position, like buttons, doors, or windows. This allows the creation of sub-behaviors for each `ObjectActionable` using only one entry-point.

b. Grabbables Objects

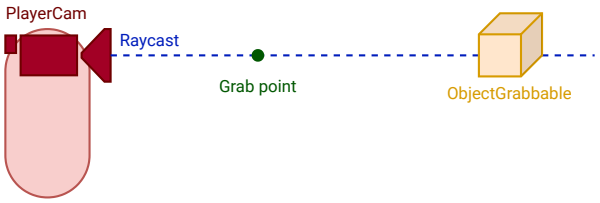

Figure 5.22: Components of the grab system

The grabbing system relies on multiple elements. Each player has an associated "grab point". Grabbed objects will have a force applied on them going towards the point.

This is the only way for the player to move objects in their environment. When the player presses the right button of their mouse:

- If no object is currently grabbed and the interaction system detects a grabbable object, the player grabs it.
- If the player is already holding an object, they drop it.

Grabbable objects are synchronized over the network. Each object can only be grabbed by a single player at a time. If another player tries to grab it, the game will ignore the action.

6. Debug Console

As we were developing more and more features, it became essential to be able to test them. In order to do so, a development interface allowing us to change the way the game behaves was necessary. For instance, these debugging tools could consist of printing information about the state of the game in a console, changing the value of variables or displaying some additional graphics over the game elements.

The first point to be decided was when and how to toggle this development mode. After some brainstorming, it was decided to allow the player to display this debug console interface whenever the player is active, simply by pressing the right shift key of the keyboard five times in quick succession. This shortcut is unlikely to impede on regular gameplay as it requires a rarely used key to be pressed in an unconventional way.

Afterwards, the actual mechanics of the console had to be worked out. It was decided to use a very simple interface. As the mouse pointer is not active during the game, pressing the T key focuses on the text input of the console, which allows the player to enter a command.

Figure 6.23: The Debug Console

A public method allows for the registration of any command to the console from any script in the game, so that each system manages its commands locally.

When focused on the debug console, a special input map is used in order to prevent interacting with the game while pressing keys to type text. For instance, this prevents the player from going forwards when the Z key is pressed.

The commands created and used so far are:

- `sayHello`: displays “Hello, World!”. Used to test the debug console.
- `inputMode`: displays which input map was being used before switching to the debugConsole
- `loadDemoScene`: loads a test scene and sets it up
- `help`: lists all available commands
- `exit`: quits the game
- `hanoiToggleDebug`: activates Hanoi Towers puzzle debug visuals

Moreover, in order to maintain more precise control over what is typed and to create and navigate in a command history system, we implemented a custom text input system.

7. AI

The AI system is an important part of the rat puzzle, during which the players will need to lure a rat that flees from them. This AI has 3 states: wandering, fleeing and being attracted by bait.

The AI system has not been implemented in the current version, but its functioning is explained in the following section.

The AI has a director managing its behavior, and a separate script file for each of the states. The relevant script is chosen by the director depending on the current situation.

- For the wandering script, the AI draws a line between its current position and a random position in its radius. Then, it walks to the chosen position following the line and waits for a small time (from 0 to 5 seconds). It is the default state chosen by the director.
- The fleeing script takes the positions of the player and the position of the rat. The AI tries to flee to a position further to the player. This script gets called by the director only if the difference between the two positions is under a certain threshold.
- The “being attracted” script simply makes the AI move directly toward the bait. This script is called when the bait exists in the environment.

To allow the AI to navigate the environment, the AI Navigation package is used, allowing the game objects to move in a delimited area. The movement plane is generated automatically by Unity, but can be personalized to determine walkable and non-walkable areas.

8. Translation System

SubWay Out aims to be accessible to as large an audience as possible. Translating the menus in different languages is therefore essential. For this reason, a settings menu was created in order to be able to change the language between English, French and Spanish.

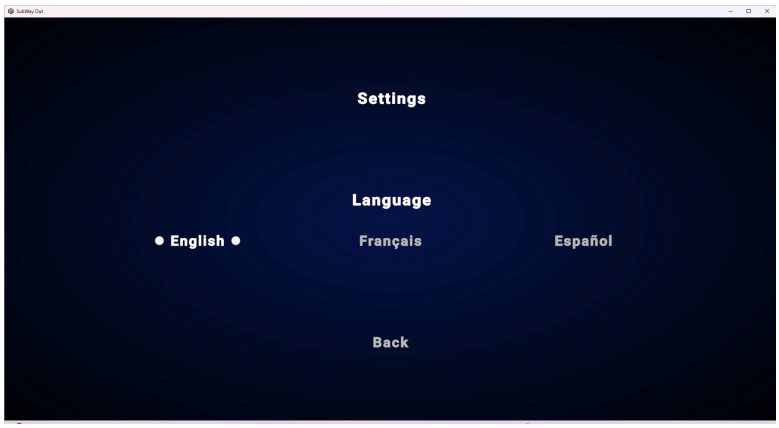

Figure 8.24: Settings Menu to change the language

All the texts and buttons need to change depending on the choice of language. To do that, a table with the different translations was made.

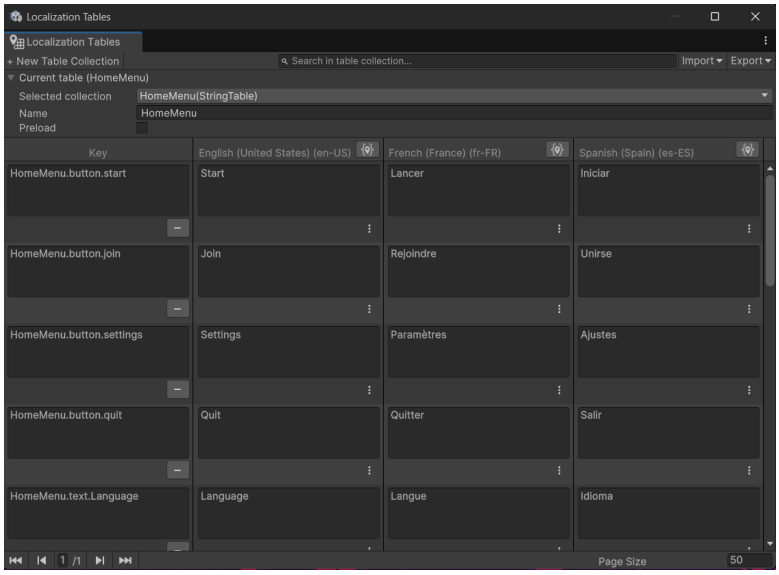

Figure 8.25: Table with the different translations

The Unity Localization package assigns and displays the corresponding text resource depending on the option selected at the beginning of the game. As the text contained in a button changes, it is necessary to adjust the button width. Furthermore, this translation system does not only handle visual texts and buttons but also voicelines used in the hint system.

9. Website

In order to carry on the communication around SubWay Out, working on the website was necessary.

First, a design was made, and then, as the game was evolving and had more and more satisfying graphics, generic images on the website were replaced by pictures from the game. Therefore, adapting some parts of the design, especially the colors to match with these new darker images was essential.

a. Website Design

The Subway Out's website is composed of 3 sections:

- **Hero** section: It is the section that visitors will first see on the website. It includes a download button for the game and an animated background with in-game captures.
- **News** section: To inform visitors of latest news about the development of the game and events about it, with articles on the website and an embed to the Instagram page of the Head4che Production Company.
- **Documents** section: Where visitors can find all documents related to the development of the game such as specifications.

The layout of this homepage is mostly inspired by others games' websites such as Genshin Impact, Honkai: Star Rail (by Hoyoverse), VALORANT (Riot Games) and also Strinova (iDreamSky).

The website's colors palette is based on the RATP's colors palette. It will be changed to darker colors to match with the atmosphere of the game.

Figure 9.26: Webpage of the game as designed

b. Website Development

As the website design is done, developing it is the last remaining task. Doing so is relatively fast, as everything is already set up. Until today, only a small part of it is available online. Indeed, as it was not a priority, most of the work has been done on the game rather than on the development of its website.

Therefore, the elements left to be implemented:

- A “news” part about SubWay Out
- A page about the studio Head4che Production Company and its members

For now, some buttons on the website are disabled. It is totally normal, and they will be activated when the corresponding files are uploaded. A demo will be uploaded before the release of the full game.

In addition, making the website responsive (especially mobile-friendly) is still an issue to fix.

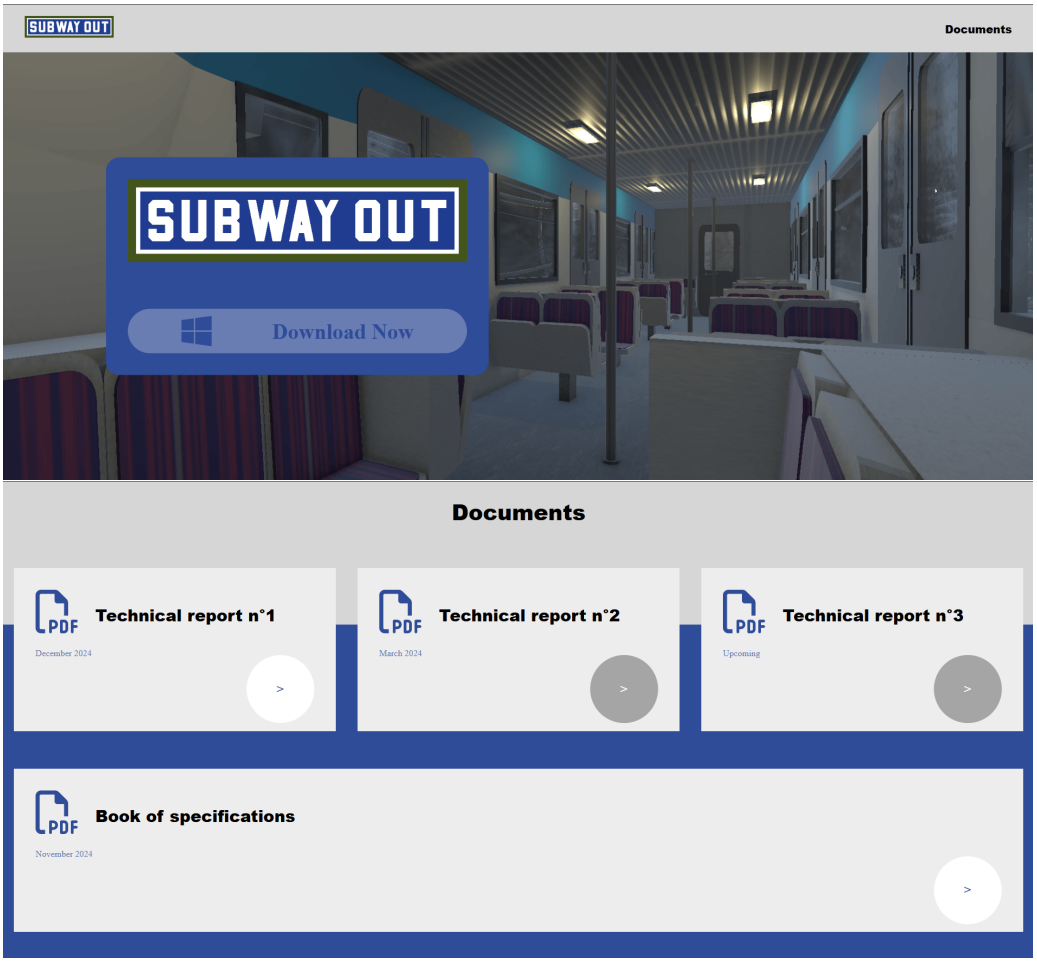

Figure 9.27: Current website

IV. Previsions and Planning

1. Summary

Here is a table summarizing the previsions of the advancement in the different tasks we have to work on, comparing it to the previous report and where we currently are.

	Planned advancement	Current advancement	Previous advancement
Scenario	95%	100%	80%
Level design	95%	90%	50%
Gameplay	60%	10%	5%
Game engine	85%	100%	75%
AI	20%	15%	0%
Multiplayer	80%	85%	80%
Networking	80%	100%	20%
Website	50%	50%	15%
3D Modeling	80%	80%	60%
Visuals	40%	40%	0%
Sound	30%	5%	0%
Branding	90%	90%	90%
Trailer	10%	5%	0%

Table IV..1: Advancement comparaison
Green : On time or better / **Yellow** : Nearly achieved / **Red** : Late

2. Explanations

a. Scenario

The scenario has been fully written. The entirety of the flow of the story is ready. No work is left to do about this point.

b. Level Design

Almost all the puzzles of the game have been designed.
Some details are left to be ironed out, but can be neglected with respect to what has been done.
On the graph in appendix, which shows all the puzzles of the game, the logical associations left to design are denoted by "???".

c. Gameplay

The game engine has been fully implemented. It was a priority as the game needs it to work, and its implementation is virtually entirely complete, ahead of schedule.

d. Game Engine

The advancement of gameplay – in this game, the puzzles – is considerably behind schedule. This is due to the fact all puzzles rely more heavily on the game engine than anticipated, and a lot of work has been shifted towards the core foundation of the game rather than the puzzles.
This has led to a faster progress on the game engine than expected, but a lag in the actual implementation of the puzzles.
Despite not being ideal, this situation is strongly mitigated by the fact the underlying mechanics for all the remaining puzzles are fully complete.

e. AI

Some planning about AI has been done and the implementation of these ideas has been started. However, a lot of work is still left to do. For the moment, the AI is a draft that needs to be reworked and upgraded in order to obtain a functional puzzle at the end.

f. Multiplayer

Multiplayer for the current advancement of gameplay is almost finished. Only three things are left to implement: the rotation of players is not yet synchronized, Hanoi's internal state is not fully synchronized across clients, and the game does not properly reset upon completion.

g. Networking

The networking system is fully operational. No extra work will need to be done on this front.

h. Website

The process of designing the website has progressed a lot. One of its two pages has been fully conceived and the design of the other one is largely started.
The design being done and the development environment being really efficient, implementing the remaining pages of the website will be relatively fast.
Therefore, as the website is not a priority and only represents a modest workload, even if it is not as advanced as planned, it is not a major issue.

i. 3D Modeling

Currently, only one of the five character models is implemented but having a template will be helpful to adapt the designs for the others. Thus, it should be relatively easy to finish the models of the characters.
In addition, the modelization of the new design of Hanoi Towers pieces will also be quick as the issue is known (polygon maximum limit).

j. Visuals

Concerning the visuals, the UIs are ready, even though they still might change. The work left to do is entirely about the in-game animations. Some animations already exist, and creating new ones should not be a heavy task.

k. Sound

Only a small part of the voicelines of the hint system have been recorded in three languages and are now present in the game. Apart from this, the sound atmosphere is still left to work on, as it was not a priority until now.

l. Branding

No additional work has been done about the branding since the last report.

m. Trailer

The trailer's creation has not begun yet. However, despite this, some assets necessary to its realization have been recorded. These assets are mainly screenshots and recordings from the game.

V. Conclusion

In summary, during these last 2 months of work, the project has been upgraded a lot thanks to considerable teamwork. From a draft at first, to the concretization of SubWay Out, watching the game progression produces a gratifying reward for every member of the studio.
Being aware of the delays is important in order to be efficient for the last months of work. Some functionalities were prioritized compared to others. These choices allowed the studio to be ahead of schedule for some tasks such as for the game engine, the scenario, the networking, or the multiplayer. However, for other tasks a more significant amount of work will be needed to catch up with the pace that was originally decided on. This mainly concerns the gameplay and the sound design. The team is confident and optimistic in successfully compensating the small delays accumulated.

VI. Appendix

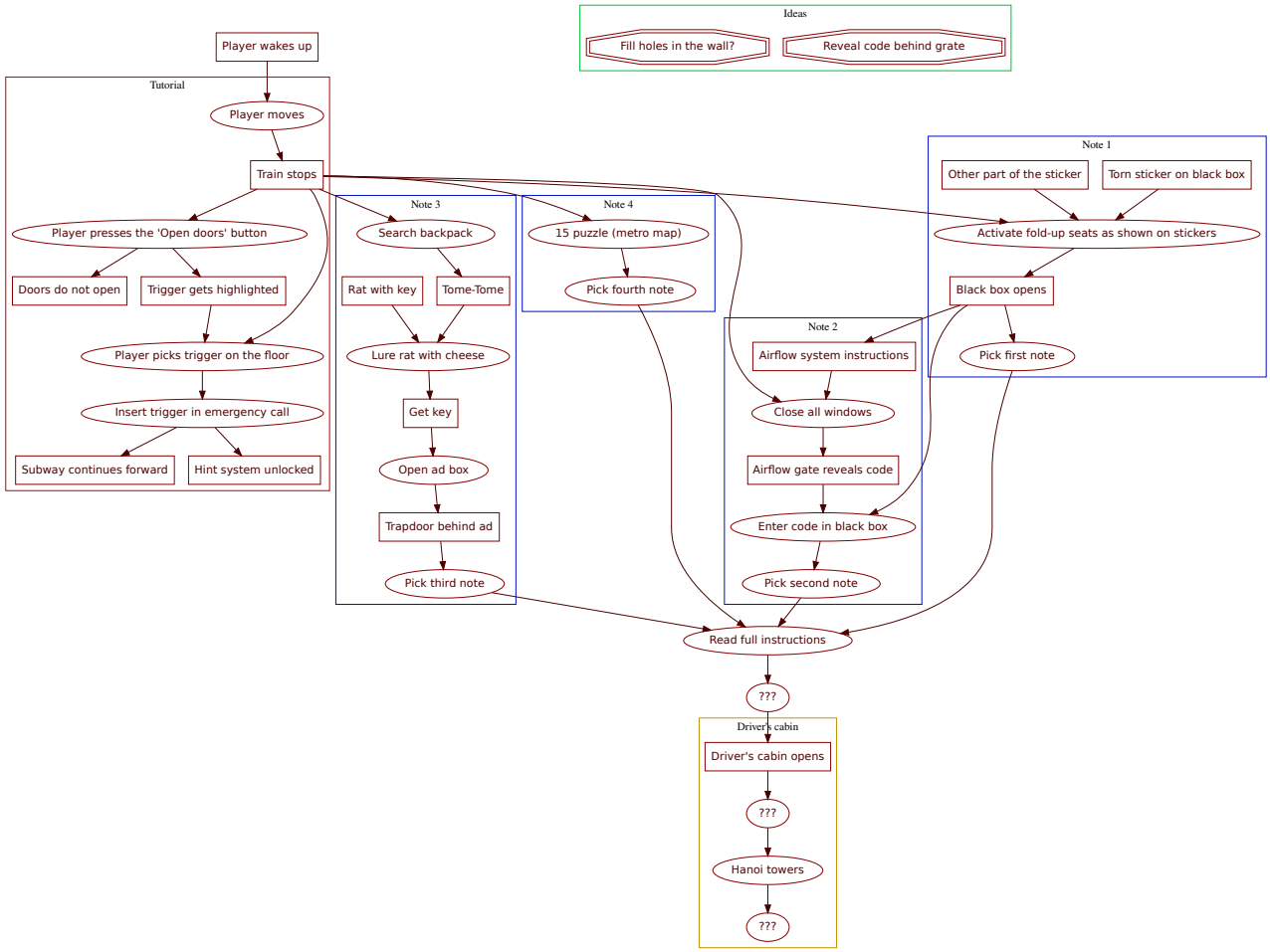

Figure 0.1: Puzzle Graph