ROOFTOP RIVALS

RAPPORT DE PREMIERE SOUTENANCE – SAE J3D

SALVAN ACHILLE - BENNJAKHOUKH RAYAN – CLEMENT GREGORY-LUCAS – HADJAB MEHDI – PUIJALON MILAN

JANUARY 2026 / EPITA PARIS

SUMMARY

Table of Contents

INTRODUCTION

In our first year at EPITA, we have to develop a videogame with Python. After a few group meetings, we decided to create a 3D videogame with a future theme. In our videogame, there is a chase game with two persons. One has to catch the other one, but at the same time, this other has to escape from him, when a player is caught, the turn changes and he has to be chased. The player who stays hunted the most time at the end of a round wins. In a game, three rounds are played. Each round lasts five minutes. After we have decided what videogame we would develop, naturally we asked ourselves: "But how to develop all this with Python?"

In a few researches on the web, we decided to develop our videogame with Panda3D, since we believed that with Panda3D, we would have an easier task to code a 3D videogame compared to using other tools like Pygame. With Panda3D, we know that we have a library more suited to our needs.. For the 3D models, we decided to utilize the program Blender because it is free software that is very efficient in creating game resources. Of course, we utilized Github to collaborate and monitor our progress, which helped us ensure there are no conflicts when working on different parts by multiple individuals.

During the first search phase we decided of the videogame concept, the project that we wanted to make and the general atmosphere. After this before beginning the project we gave to each of us a role in the realization of this project based on everyone's skills and interests. Milan was the one that was in charge of the music (soundboard) and the 3D modeling, Rayan was in charge of the creation of the website and the menus, Achille was in charge of the multiplayer system and the movements controls and finally Mehdi was in charge of the GameMode logic, the HUD display and the AI for single player mode. This was the initial repartition for everyone in this project. After this initial planning, every member started to do some researches to learn how to accomplish their work properly. We watched tutorials, read documentation and did some small tests to understand how Panda3D works.

In the early of December everyone has started their part but a little reorganization have been made because Gregory-Lucas have been added to our group, thus Rayan gave the Website to do for Gregory-Lucas even if he already started it because it seemed to be the best decision for everyone and Gregory-Lucas had some experience with web development. Thus Rayan has been also in charge of the 3D with Milan and of the presentation of Methodology for the final report. At this point Achille, Mehdi and Milan already started development and Rayan and Gregory-Lucas were starting their respective parts to catch up with the others.

ORGANIZATION & METHODOLOGY

To work well together we use different tools. First to code our project we use Panda3D, it is the most efficient programming environment for our type of project in Python in 3D, it permits us to code our videogame so this is the main tool that we are using for the development. Additionally we also use a lot the app Discord because this app permits us to communicate about this project efficiently, we can share our screen if we want to show something to the others and it is a social media adapted to computers thus when we work on our computers on the project it is very useful. We have a dedicated server for our project where we can discuss about bugs, share ideas and plan our work sessions. For the artistic side of our project, for the realization of the map and the 3D models we use the software Blender, when we searched for a software for 3D it is the software that seemed the more adapted to our project because it has all the features we need and it's free. Even if we can communicate while coding and share our screen it is important to have a common repository for the group so our codes are in a same place and we don't lose any work. This is why we use also Github because by pushing our work on it it allows us to see the work of the others, maybe give some advices or ask for clarifications to understand the code of others. This permits us to keep tracking of the work of the others and to understand how the project is going, we can also see who did what and when. Finally we use also Youtube because it is a great place to learn some knowledge on something that we have to create, for example it is on Youtube that we understood how to use Blender and also how to implement some features in Panda3D like collision detection or camera controls. There are many tutorials that helped us a lot during the development.

In our group we all have different roles so we can make sure that we are all keeping going in the same direction and that the project stays organized. For example Rayan is the project leader, to make sure that everyone is following the plan every week we do a little meeting to make sure that everyone is good and so if someone is not good or is blocked on something, someone can help him on a task. During these meetings we also discuss about what we accomplished during the week and what we plan to do for the next week. This helps us to stay motivated and to see the progress we are making on the project.

The Methodology classes are also helping us to work on our project. Sometimes it is very hard to find time to keep going on the project and to be motivated but the methods like the "time boxing" or the "Pomodoro" are very useful because it helps us getting organized and managing our time better. For example with the Pomodoro technique we work for 25 minutes without any distraction and then we take a 5 minutes break, this way we stay focused and we don't get too tired. The time boxing is also helpful because we can set a specific time for a specific task and we know that we have to finish it in this time, so we don't spend too much time on one thing. These methods really helped us to stay productive and to not procrastinate on the project. Sometimes when we are stuck on a problem it's also good to take a break and come back later with a fresh mind, the Methodology classes taught us that and it's really true. Thanks to these techniques we managed to keep a good rhythm of work even when we had other projects or exams at the same time.

INDIVIDUALS CONTRIBUTIONS

Rayan BENNJAKHOUKH – *Group Leader*

Objectives

Concerning my position as Group Leader, I have two main tasks. On the one hand, I am responsible for the coordination of the group as well as the management of the global progress of the project. On the other hand, I have specific tasks related to the technologies I have to learn, such as the development of the menus of the game and learning about the use of the program “Blender” to assist in the map development. Among my primary missions are the organization of the weekly meetings, the management of the global planning, the attribution of the tasks based on the abilities of each member of the group, and the development of the interface with which the players will first interact with the game.

Project Organization and Management

Setting Up Work Rituals

I soon realized that the basic requirement for this newly formed group in October was to organize an efficient working system. Without it, I realized that we might be wasting our time by working in our separate corners. I decided it was a good initiative to organize a systematic meeting between us every Friday after our classes. Why Friday, you might ask? This is the only time when everyone is available in school, and it will help us assess what has happened in the last week and organize our activities for the coming weekend and the coming week.

These meetings turned out to be the backbone of our organization. They are structured in a fixed way: we begin with a round table in which every participant shares their progress since the last meeting, specifying what was done, what’s in progress, and, above all, what is creating issues. This is a decisive point because we can quickly point out technical or organization issues that could cause trouble. After that, we all share solutions for the issues brought up. To finish, we distribute the work for the next week, in a way that no one is overloaded and while respecting the inter-dependencies between tasks.

Strategic Task Distribution

Among the key roles I had was delegating tasks to members of the team. I remember dedicating a considerable amount of time speaking one on one with everyone to know their areas of strength, where their knowledge is, and what they wish to accomplish with this project.

In the case of Achilles, the skillset of advanced networking expertise made him the most qualified to work on the multiplayer part of the architecture. For Mehdi, the keen eye for game design made him the most suitable for the creation of the mechanics and the power-ups for the game. In the case of Milan, his artistic skills made him the most qualified to become our next Creative Director, learning to use the software for the cyberpunk style that we want for the visuals of our game.

Gregory-Lucas's situation is special in the sense that he joined the group towards the end of November. As soon as he joined, he showed interest in contributing to the group’s project website, which I had personally started developing during the first weeks. I decided to wholly entrust him with the website, contrary to my initial plans, which involved continued work on it. This was a perfect way of involving Gregory-Luacs in our group activities, considering the little time he had been part of the group. Moreover, it gave me ample time to handle other aspects related to our group's project.

Managing Unexpected Situations

The project has faced several big obstacles. First, there was the addition of Gregory-Lucas as the fifth member at the end of November. This required a quick reorganization of our work structure. After giving it some thought, I decided to give him the website I was working on. This decision had personal repercussions for me: I had been working for several weeks on that site, and abandoning everything overnight wasn't so easy psychologically. I needed a little time to get back into the project with the same efficiency, especially since this period coincided with a heavy academic workload.

The second impediment, and maybe the most important one, was the unforeseen delay in the generation of the 3D map. Milan faced more problems in Blender than we initially foreseen. The learning slope was much higher than we expected in our initial planning process. This was a constraint on our project schedule, since without the 3D content of the map, other aspects of the game also couldn’t be developed in full detail.

With this critical block, I had an important choice to make: learn Blender myself, as rapidly as possible, in order to support Milan and unblock the situation. During several weeks in December and January, evenings and weekends were dedicated to highly concentrated learning of Blender. I followed tutorials, looking only for what I needed to do to achieve functional assets-basic modeling, UV mapping, simple texturing, and optimized export to Panda3D.

But I must be honest here: I did not have a chance to develop finished assets so far. The truth is, I managed to learn the basics of Blender only. I know how to operate in the interface, I have an idea about the basics of the software, I know how to make simple shapes, and I know how the pipeline is supposed to look, but I have not managed to develop finished assets for the game and include them in the process. The learning process took all the time I had to devote to this project, given my other obligations and duties. My involvement in the creation of the map would be more potential so far, which means I am ready to help and contribute, but I have not had a chance to really contribute so far.

Menu Development

Learning and Technical Choices

After handing over the website to Gregory-Lucas, I decided to fully concentrate on developing the game's menus. This transition represented a complete change of technology and approach. Moving from web development to GUI development in a 3D game engine like Panda3D represented a real conceptual leap.

Although I already had some experience in Python programming, having coded a few simple games when I was fifteen, using Panda3D to create graphical interfaces was completely new territory for me. The choice of Panda3D as our game engine was a collective decision. We absolutely wanted to make a 3D game, and Panda3D offered the advantage of being entirely usable in Python while offering robust 3D capabilities.

However, going along with Panda3D meant I had to learn a whole new environment. This not only meant I had to learn about the basics of ShowBase, which is essentially the base class for anything using Panda3D, but I had to learn about DirectGui, which is in charge of handling everything related to the interface. I had to learn how I could make buttons and interfaces that not only worked, but looked good in our cyberpunk theme.

Learning Method

In December, I chose to fully engage with the code material. In learning Panda3D, my strategy was multi-sourced learning involving Panda3D tutorials on YouTube, learning from the official website documentation, and turning to Stack Overflow when facing particular problems. For me, learning involved moments of frustration. Compared to web development where one could see results almost instantly, learning Panda3D involved more hard work and perseverance for even tiny milestones that were victories in themselves to sustain my drive to learn.

Concrete Achievements and Technical Details

I developed two main menus for the game: the base menu that appears when launching the game, and the in-game pause menu.

Understanding the Class System and Imports

Everything in Panda3D revolves around classes. I needed to import necessary modules:

```
from direct.showbase.ShowBase import ShowBase
from direct.gui.DirectGui import *
from panda3d.core import TextNode
```

ShowBase handles the rendering window and game loop. DirectGui gives access to GUI elements. TextNode controls text alignment. Then I created my main class:

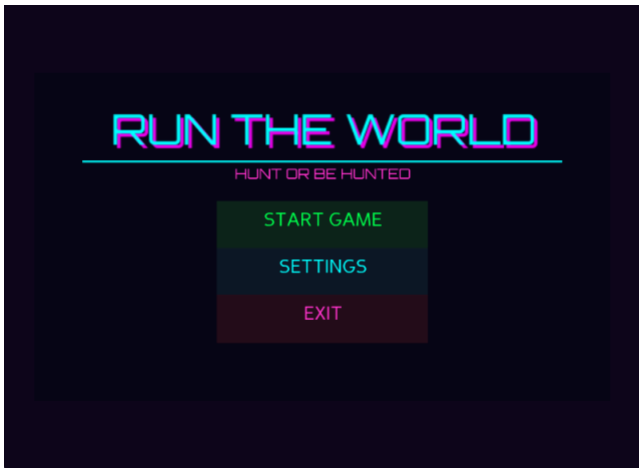
```
class MonJeu(ShowBase):
    def __init__(self):
        ShowBase.__init__(self)
        self.disableMouse()
        self.setBackgroundColor(0.05, 0.02, 0.1, 1)
        self.fontTitre = loader.loadFont('fonts/Orbitron/Orbitron-
Bold.ttf')
        self.creerMenu()
```

The Base Menu

I created the base menu through the creerMenu method:

```
def creerMenu(self):
    self.fondMenu = DirectFrame(
        frameColor=(0.02, 0.02, 0.08, 0.85),  # Dark semi-transparent
background
        frameSize=(-1.2, 1.2, -0.7, 0.7)
    )

    self.titre = OnscreenText(
        text = "RUN THE WORLD",
        pos = (0, 0.4),
        scale = 0.18,
        fg = (0, 1, 1, 1),  # Cyan neon
        shadow = (1,0,1,0.9),  # Pink-magenta shadow
        shadowOffset = (0.08, 0.08),
        align = TextNode.ACenter,
        font = self.fontTitre
    )

    self.btnJouer = DirectButton(
        text="START GAME",
        scale=0.08,
        pos=(0, 0, 0.05),
        relief=DGG.FLAT,
        frameColor=(0.05, 0.15, 0.1, 0.9),  # Dark green background
        frameSize=(-5.5, 5.5, -1.3, 1.3),
        text_fg=(0, 1, 0.3, 1),  # Neon green text
        command = self.afficherMenuJeu
    )
```

The command parameter calls afficherMenuJeu when clicked. I created similar buttons for "SETTINGS" (cyan) and "EXIT" (pink-magenta).



Menu Navigation - Hide vs Destroy

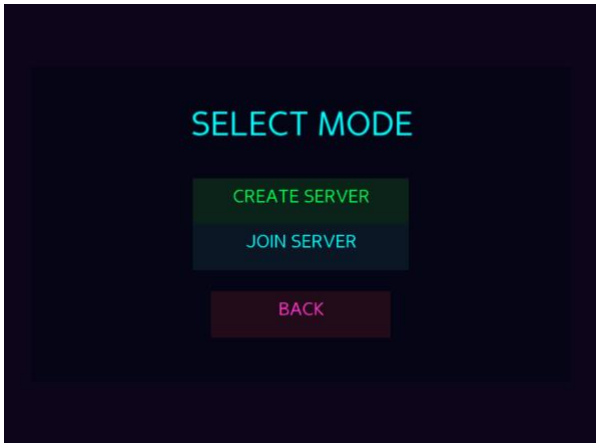
When clicking "START GAME", I transition to a submenu:

```
def afficherMenuJeu(self):
    self.fondMenu.hide()  # Hide but keep in memory
    self.titre.hide()
    self.sousTitre.hide()
    self.btnJouer.hide()
    self.btnOptions.hide()
    self.btnQuitter.hide()
    self.creerMenuJeu()  # Create mode selection submenu
```

.hide() makes elements invisible but keeps them in memory. Then creerMenuJeu() creates the mode selection submenu with "CREATE SERVER" and "JOIN SERVER" buttons. The "BACK" button uses .destroy():

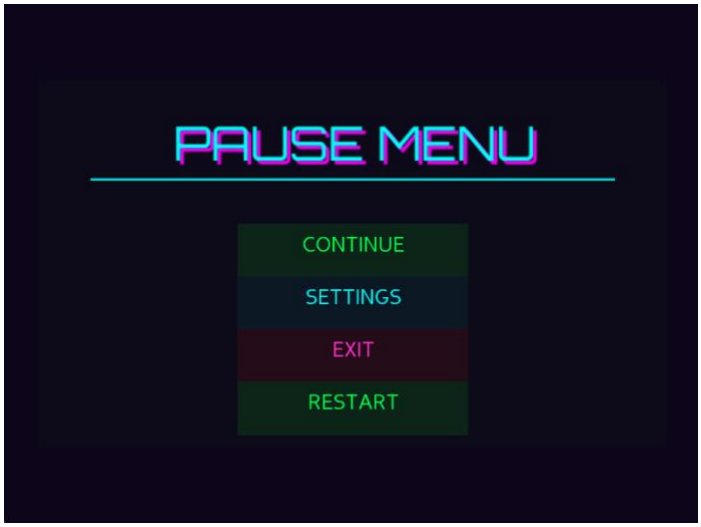
```
def retourMenuPrincipal(self):
    self.fondMenuJeu.destroy()  # Permanently delete to avoid memory leaks
    self.titreMenuJeu.destroy()
    self.btnCreerServeur.destroy()
    self.btnRejoindreServeur.destroy()
    self.btnRetourMenu.destroy()

    self.fondMenu.show()  # Show hidden elements
    self.titre.show()
    self.sousTitre.show()
    self.btnJouer.show()
    self.btnOptions.show()
    self.btnQuitter.show()
```

.destroy() permanently deletes elements to avoid memory leaks. .show() makes hidden elements visible again.



The In-Game Pause Menu

For the pause menu, I followed the same principles but added a "RESTART" button alongside "CONTINUE", "SETTINGS", and "EXIT". The background is more transparent to show the frozen game behind. I haven't implemented the settings functionality yet - that's planned for next semester. The other buttons are fully functional.



Key Moment

Creating this first functional menu was a important step for me. It was the first time since the project started in October that I created something really concrete. When I saw my menu appear for the first time, with its neon colors, its buttons responding to clicks, I felt a satisfaction. This moment was like a reward for all the efforts made, for all the hours spent debugging errors. This first concrete success remotivated me and confirmed that our cyberpunk vision was achievable and starting to take shape.

Common Tasks and Documentation

As Group Leader, I naturally handle the majority of the project's common deliverables. Being the one who coordinates everyone's work and participates in everyone's technical discussions, I'm naturally the one who has the most complete vision of the project.

Oral presentations are also my responsibility. For each review, I create slideshows ensuring they tell a coherent story of our project. I coordinate each member's speech to avoid repetitions and guarantee that every important aspect is covered without the presentation being too long or disorganized.

On the technical side, I also contribute to organizing our GitHub repository. I help maintain a clear branch structure and participate in resolving conflicts during merges.

Conclusion

The last semester has been a continuous learning process with regard to technical tasks and organization tasks. It was understood that the tasks and responsibilities of a Group Leader are much larger and complex compared to organizing a meeting. You have to be able to predict problems, improvise in case something unexpected occurs, and keep your team motivated, while also making tough decisions in case of conflicts between personal interests and the project’s interests.

The menus I developed now constitute the first impression players will have of our game. I'm proud of the result: the menus work without bugs, navigation is intuitive, and the neon-on-dark aesthetic immediately creates the atmosphere we're aiming for.

As for the next semester, I will again take up my position as the coordinator/developer. I will maintain the meetings, oversee the intensive testing process, and complete the menus by incorporating the still-pending functionalities, such as the audio settings menu and the end-of-game screen. As has been my objective throughout, my target still stands unchanged. I want to ensure that every member of my team has conditions ideal enough for us to make our game as optimum as possible.

Achille SALVAN – *Technical Lead*

Objectives

My Role: As the Technical Lead, I handle the tasks in the game code involving complex mechanics. My main missions are implementing player movements, which are complex in our game, and managing the multiplayer system to allow players to play together. Additionally, having experience in Web development, I took care of starting the website for this first presentation.

Choice of P2P

At first, we hesitated between using a classic server for our game or choosing P2P. Here are the arguments that made us choose Peer-to-Peer, with a server used only to connect clients before the game starts.

- Latency

By connecting players directly, we remove a step that data has to go through. In a classic network architecture, data goes from Client A to the server, then to Client B. In Peer-to-Peer, as the name implies, data goes directly from one client to the other without a server in the middle. This can potentially cut the latency in half, which is significant in a competitive game.

- Server

Load By removing the server from the equation, we remove a huge burden of calculation and bandwidth. A very simple server costing less than 5€ per month could theoretically handle thousands of simultaneous games because, once the games are launched, no more work is required from it. Usually, the server acts as a communication point between players and must perform all the calculations for the game logic, leaving the clients with only rendering and input management.

- Flexibility

Thanks to Peer-to-Peer, we have great deployment flexibility. Since the game does not need a connection to a remote server to run, this gives us the possibility to implement a LAN mode very easily later on. This means that two players connected to the same Wi-Fi could play together with almost zero latency and without any dependence on the Internet.

Movements

Input Management

Instead of checking the keyboard state at every frame, we use an event system included in Panda3D. It works via the "accept" method, which allows us to bind an action to the pressing or releasing of a key.

```
def _init_input(self):
    """Setup des inputs"""
    self.inputs = {
        "forward": False, "back": False, "left": False, "right": False,
        "up": False, "down": False,
    }
    self.game.app.accept("z", self._set_input, ["forward", True])
    self.game.app.accept("z-up", self._set_input, ["forward", False])
    self.game.app.accept("n", self._toggle_no_clip)

def _set_input(self, key: str, value: bool):
    self.inputs[key] = value
```

For keys that need to be held down, such as movement keys, we save their state in a dictionary and update it when a key is pressed or released.

Camera Management

For mouse movements, we first need to lock the mouse in the center of the window. Once this is done, after every frame, the mouse is reset to the center of the window. Therefore, to get the player's mouse movement each frame, we simply look at the cursor's position relative to the center of the window.

```
def _update_mouse_look(self):
    pointer = self.game.app.win.getPointer(0)
    props = self.game.app.win.getProperties()

    center_x = self.game.app.win.getXSize() // 2
    center_y = self.game.app.win.getYSize() // 2
    dx = pointer.getX() - center_x
    dy = pointer.getY() - center_y
    self.game.app.win.movePointer(0, center_x, center_y)

    self.yaw -= dx * self.mouse_sensitivity
    self.yaw = self.yaw % 360
    self.pitch = max(-89.0, min(89.0, self.pitch - dy * self.mouse_sensitivity))
    self.set_rotation(self.yaw, self.pitch)
```

To prevent the Yaw from accumulating if the player spins around, we use modulo 360. This allows us to keep a Yaw value between 0 and 360 (excluded). We must also apply a constraint to the pitch to prevent the camera from flipping over completely. Therefore, the pitch is kept between -90 and 90.

View System

The game will be played in first-person, but for the development of movements, I added the option to switch to third-person (camera behind the player). This makes it easier to visualize the player's movements, and later, it will allow us to code animations while watching them.

```
self.game.app.camera.setPos(0, -self.height * THIRD_PERSON_DISTANCE_RATIO,
                            self.height * THIRD_PERSON_HEIGHT_RATIO)
self.game.app.camera.lookAt(self.camera_eye)
```

Since the camera is attached to the player's head, 0 0 0 represents the player's eyes. We simply add an offset to the camera to move it back and up. We also modify its rotation, so it looks at the player.

Movement Physics

For physics, we need to handle several situations, but they all require an initial calculation of the player's movement vector. The desired direction vector is obtained from the player's inputs (WASD/ZQSD). Its Z-axis is equal to 0, and it is normalized, meaning it has a length of 1. Once we have this vector, we multiply it by a constant to modify the player's speed.

1. Ground Movements

We simply apply the vector to the player; no modifications need to be applied to the vector.

2. Free Fall Movement

When the player is jumping, we cannot apply the movement vector directly to the player, or it would look like they are gliding. Instead, we add a value to the X velocity to make it approach the X value of the desired direction vector. We do the same for the Y-axis. This makes air movements slower without simply reducing speed, keeping the effect natural.

3. "No-Clip" Mode

This is a development tool and will be removed from the final version. It allows flying and passing through walls. To fly, we use the same logic as the movement inputs but add the SPACE and SHIFT keys to move up and down on the Z-axis. To disable collisions, we use removeCharacter, which detaches the player node from the world, causing it to ignore the world's collision boxes.

Multiplayer

A Hybrid Architecture

As mentioned above, we are not using pure Peer-to-Peer. This would have a major downside: players would need to open ports on their connection and share their public IP to connect. We chose a Signaling server that is only useful for the connection phase. It registers the started games and their associated codes. A guest can join a host using only this code, which is much simpler. From that moment on, the real Peer-to-Peer begins.

Connecting Players

Our goal here is to open a DataChannel. A DataChannel is a route between two clients that allows them to exchange information without going through a server. Establishing a connection between players is the most technical part of Peer-to-Peer. It happens in several steps:

1. Creating the game

A player creates a game by sending a request to the server via a WebSocket. A WebSocket allows the server to send information to clients at any time without the client making a request. The server creates a game, adds the host to the party, and sends back a 6-letter party-code.

2. A player joins a game

Once the game is created, a player can use this code to try to connect. They send the code to the server. If a game is associated with this code, the server adds the guest to the party and notifies the host that a player has joined.

3. Sending the WebRTC Offer

The host creates an "Offer" and sends it to the server, which acts as a transit point to send it to the second player via the WebSocket. The offer contains information about the host and what they want to do (in this case, create a DataChannel) and how they can communicate

4. Sending the WebRTC Answer

Once the guest receives the WebRTC offer, they send an answer back to the host via the server. Their answer is stricter and defines how they will communicate. For example, if the host said, "I can speak French, English, and Spanish," and the client knows they can speak English and Spanish, the client replies, "We will speak in English."

5. Exchanging ICE Candidates

After the host receives the client's answer, they send ICE candidates. These are different addresses/paths to use for communication. Sending multiple paths helps maintain the connection if one path becomes inaccessible. The client then sends their own ICE candidates.

6. Creating the DataChannel

At this moment, both clients know how to communicate and, thanks to the ICE candidates, how to find the other client. They can then create a DataChannel. From now on, the two clients no longer depend on the server to exchange data.

Packet Exchange

We have several ways to exchange data. The easiest option would be for clients to exchange JSON. This format is similar to a dictionary, where we can associate keywords with definitions (e.g., "x" to a value corresponding to the player's X position).

However, this format is very heavy, so we chose binary packets. Each message has a different format, but the first byte of every packet corresponds to the message type. This lets us know what is inside the packet and how to decode it. For example, a packet containing a player's position has the format B5f. This means:

- 1 byte for the message type (always present).
- 5 floats: in this case, X, Y, Z, then Yaw and Pitch.

This economy might seem minimal, but for a player's position, a JSON file would be 80-100 bytes, whereas the packet is only 21 bytes. And that is for the heaviest packet! For unique events, the packet would only be 1 byte (just the message type to say the event occurred).

Movement Synchronization

Before continuing, note that I divided the player code into 3 classes: the Player class, and two classes inheriting from it: Controller and NetworkPlayer. They all play a different role:

- Controller: This is where all movements and player inputs are managed.
- NetworkPlayer: This represents the networked player. It contains the logic to move the player avatar not controlled by the local user.
- Player: This is where the common logic between the two classes resides, such as rendering the player in the world.

20 times per second, we update the other player on the game state (e.g., the player's position). So, each player receives the other player's position 20 times per second and simply moves the avatar to synchronize movements.

However, if we do nothing else, the movements look choppy. If the game runs at 60 frames per second, the player will see the other player stay still for 3 frames, then teleport, then stay still again, etc. To fix this, we add interpolation. Every frame, we calculate an intermediate position between the last received position and the avatar's current position. It is like drawing lines to connect dots.

Website

Choice of Technologies

The first step was choosing how to build our site. In 2026, we had many choices. At first, I hesitated between React and Astro. However, after thinking about it, since our site has no backend and is simply a showcase site, adding such a Framework would have weighed down the site unnecessarily. They are good tools, but in this context, they were not relevant. Therefore, I decided to go with the classic choice: HTML, CSS, JS.

Site Content

For the theme, we decided on a dark atmosphere with a flashy accent color to recall the neon lights of cyberpunk cities. We went with a one-page design (everything on the same page with scrolling); it is modern and fits the game's aesthetic. For the download page, we decided to make a single one where the user can choose what to download. It will be possible to download the files for all presentations and the game, centralizing all downloads in one place.

Conclusion

During this first semester, I laid the foundations for movement, on which all future mechanics will rely. But above all, I built a functional multiplayer system where we can easily add events and message types. Today, if we want to add an action to synchronize, we simply assign it an unused number between 0 and 15, and then handle the packet creation and reading.

In a multiplayer game, it is essential to have functional multiplayer early on because many mechanics rely on it. This avoids having to recode them later, which justifies my choice to focus on developing the multiplayer system rather than advancing the player's movements.

Mehdi HADJAB – *Game Manager*

Introduction

As part of our video game project, we aimed to design a gaming experience that is dynamic, accessible, and strategic. The main objective was to offer gameplay that is easy to understand while remaining rich enough to keep players engaged over time. To achieve this, we worked on game mechanics, code structure, and the addition of original features.

Rooftop Rivals is inspired by a tag-like game concept: one player takes on the role of the mouse, while the other plays the hunter. The core of the game is based on time management, movement, and interactions between entities. To enrich this foundation, we decided to implement a system of temporary powers called **SUPER Powers**, which add an extra layer of strategy.

This report presents the technical and conceptual choices made, as well as the different stages of the project's implementation.

The SUPER Powers

The SUPER Powers were designed to make matches more dynamic and to avoid repetitive gameplay. The idea was to randomly spawn special entities on the game map. When a player comes into contact with one of these entities, they receive a temporary power that alters their abilities.

These powers serve several purposes:

- Introducing variety into matches
- Rewarding risk-taking and map exploration
- Creating unpredictable game-changing situations

Among the powers envisioned, the first one implemented was **SUPER Speed**, which allows a player to increase their movement speed for a short period of time.

Technical Implementation

To implement this system, I first researched how classes work in Python. I consulted various tutorials and resources to understand how to structure the code in a clean and scalable way.

After several attempts and corrections, I was able to design a clear class architecture, making it easier to manipulate game entities. Each SUPER Power is represented by a specific class inheriting from a generic power class.

Since the game physics are defined using several variables (speed, acceleration, position, etc.), the implementation of SUPER Speed involved temporarily modifying the SPEED variable. Concretely, the value of this variable is multiplied by a coefficient (for example, ×1.5) for the duration of the power.

```
SPEED = 100
FLY_SPEED = 150
MOUSE_SENSITIVITY = 0.12
```

A dedicated function was also created to detect collisions between the player and SUPER Power entities. This function uses simple distance and position calculations to determine whether the player has come into contact with the entity.

```
def check_collision(player, item):
    dx = player.x - item.x
    dy = player.y - item.y
    distance = (dx * dx + dy * dy) ** 0.5

    return distance < (player.radius + item.radius)
```

General Game Mechanics

The overall functioning of the game is based on a system of successive rounds. When a player starts a game, several parameters are initialized:

- The role of each player (mouse or hunter)
- The round duration
- The initial scores

Each round takes place over a limited amount of time. During this period, the main objective is to survive as long as possible when playing as the mouse. At the end of the round, the player who held this role for the longest time earns the point.

Future Improvements

In the long term, several improvements could be made to the game:

- Adding new SUPER Powers (invisibility, slowing down the opponent, teleportation)
- Improving artificial intelligence
- Finer balancing of power coefficients and durations

Round and Time Management

The logical core of the game relies on rigorous round management. The demarrer_round function is called at the start of the game and at each transition between rounds. Each time it is called, the current round number is incremented, the remaining time is reset based on the predefined round duration, and the time accumulator is set back to zero.

```
def demarrer_round(self):
    self.round_actuel += 1
    self.temps_restant = self.duree_round_s
    self._accumulateur_temps = 0.0
```

Role Assignment

An important aspect of RTW's mechanics is role alternation. During the first round, player 1 plays as the mouse while player 2 plays as the cat. In the second round, these roles are reversed. This logic is directly integrated into the round-start function, preventing any ambiguity during gameplay.

```
if self.round_actuel == 1:
    self.souris = self.joueur1
    self.chat = self.joueur2
elif self.round_actuel == 2:
    self.souris = self.joueur2
    self.chat = self.joueur1
else:
    self.terminee = True
```

Game Loop and Real-Time Management

The tick function represents the game's logical loop. It is called repeatedly with a constant time step (dt, for example 0.016 seconds, which corresponds to approximately 60 frames per second). This approach is commonly used in video game development to synchronize game logic with real time.

Milan PUIJALON – *Creative Director*

Objectives

There are 3 main objectives:

- Create a high quality soudtrack that reflects the spririt and atmosphere of the game.
- Create characters that fit well into the game's universe and can be implemented in a parkour and chase game.
- Create this universe with a convincing map that connects to the gameplay.

More specifically:

- The soundtrack must convey a cyberpunk universe—a chaotic, futuristic city where anarchy reigns and everything is becoming robotic, but without forgetting a certain sense of sadness about the decay of the human condition and its amorality.
  To highlight these two dominant aspects, the soundtrack should include both calm, detached music and more aggressive, dynamic music.
- The characters must be futuristic and semi-robotic to fit into the cyberpunk universe. They must also connect to the gameplay by representing the game of tag.
- The map must convey a convincing cyberpunk universe while being functional for parkour and chase gameplay. Players must be able to interact with environmental elements and use them to their advantage.

Planning

How to achieve these different objectives in terms of time and technique?

- **Music**

Since I was young, I have played various instruments, especially drums. So, I already have some musical experience. Additionally, for a few years, I have been composing music using my computer and have already released an album on streaming platforms under the name stylo bleu.

I plan to continue creating music continuously and integrating it into the video game, but this time with time, style, and spatial constraints in the creation process.

- **Characters**

I have no past experience in creating 3D models. So, it is necessary to start by learning how to use quality software for 3D model creation, while experimenting on my own and trying to create models from scratch. Also, I need to think about the artistic direction for the characters.

- **Map**

Knowing how to use 3D modeling software is also necessary for the map. Additionally, I need to start thinking about the layout of the city so that it is functional for parkour while remaining immersive. However, since creating a map takes time, I will start by thinking about the atmosphere I want to give it, but prioritize the characters and learning the software.

Progress

- **Music**

When I started working on the project, I focused on music creation because it was the only area where I had experience.
But this time, I had constraints to ensure my creations aligned with the game's artistic direction, as well as spatial and time constraints.
The artistic direction decided with the group is a cyberpunk aesthetic. To fit this aesthetic, I decided to create music primarily in these two genres:

1. Electronic music, for its futuristic and technological feel, as well as its dynamic side that can emphasize the tension during chases between the hunter and the prey.
2. Chill-hop, a style of instrumental hip-hop that is relatively calm, reflecting the melancholy of the modern world in which the characters evolve.

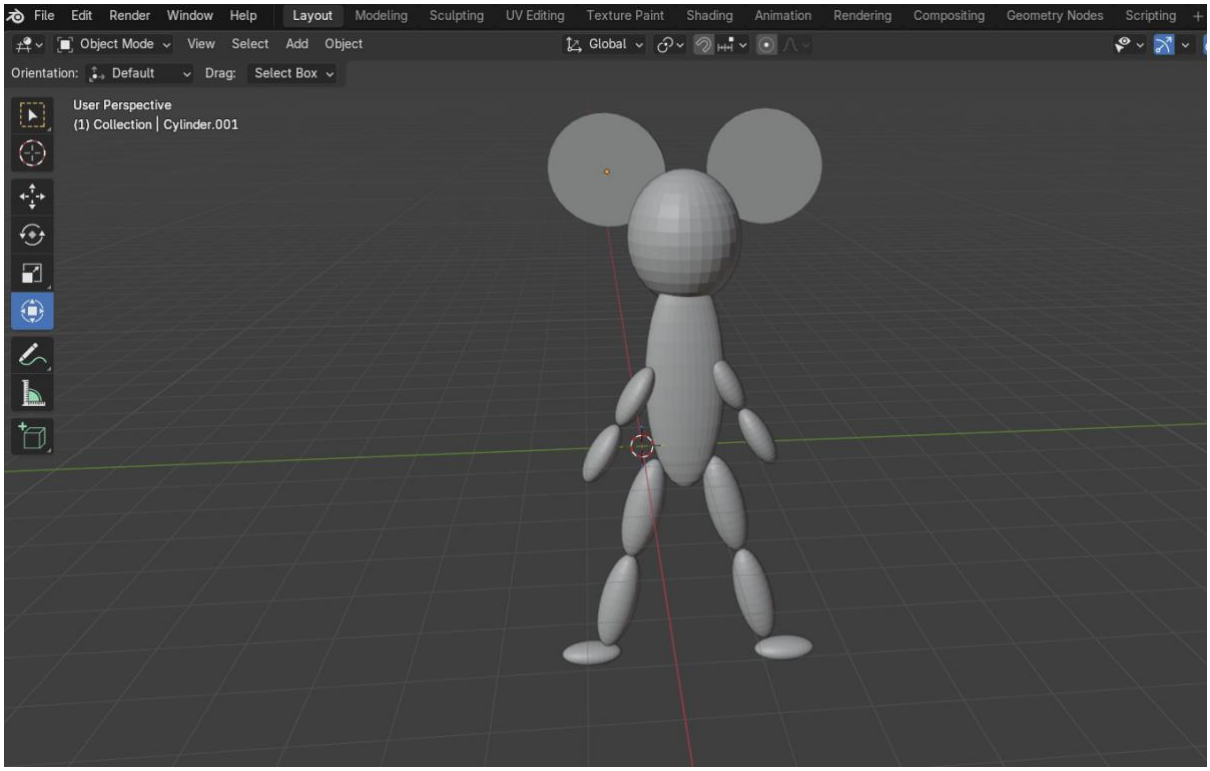
For music creation, I primarily use Studio One, a DAW with very interesting features when you know how to use it properly. However, it requires some investment to master, which is why, alongside creation, I also watch tutorials or research how the DAW works.
I compose mainly using a MIDI master keyboard, which allows me to send information to my computer and use a wide variety of sounds without necessarily needing to record acoustic instruments. This is very useful when you don't have space for physical instruments at home.

Screenshot of Studio One:

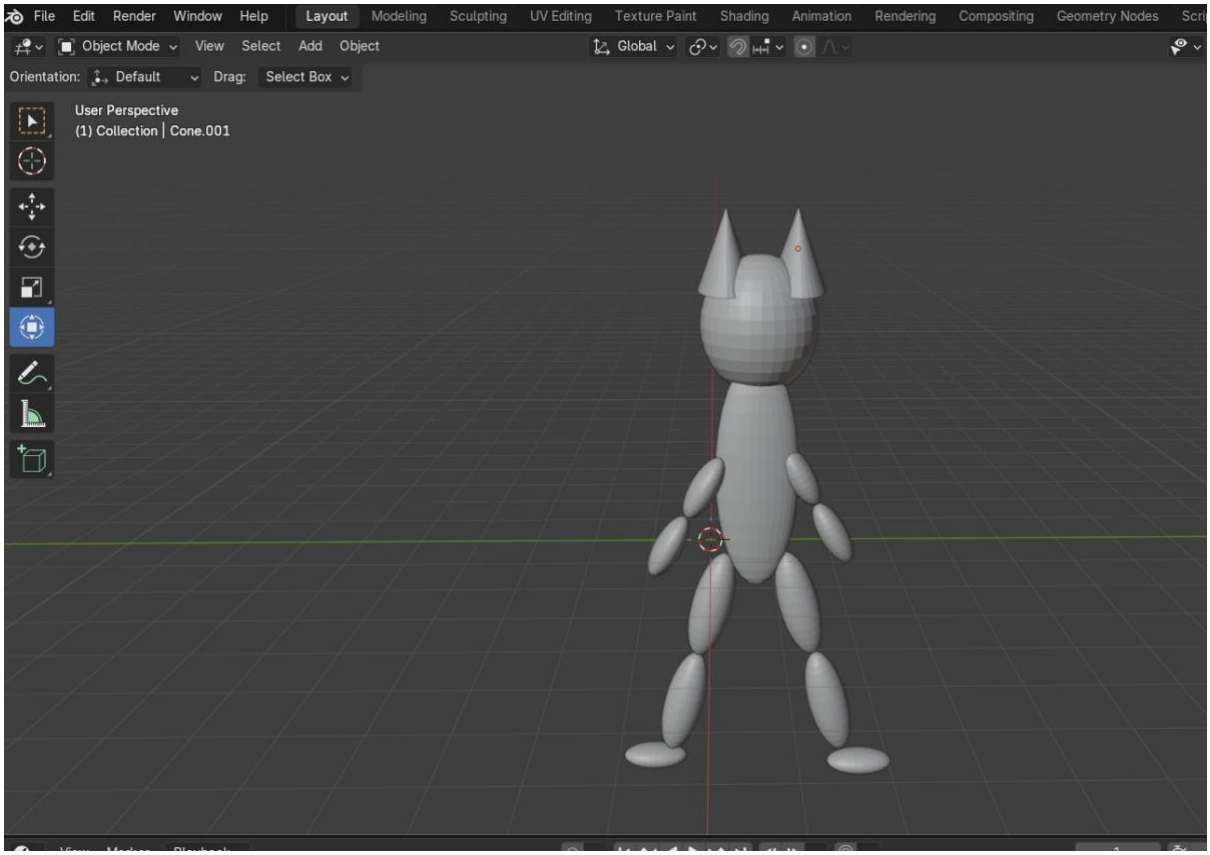
- **Characters**

Before starting to create characters, I needed quality software for 3D modeling. I chose Blender and started watching tutorials online and learning how it works. But I quickly realized that Blender is very difficult to get started with: the controls are not intuitive, there are many possible operations on different shapes, and many shortcuts to know to create efficiently. I soon felt lost about how Blender works and forgot the commands and shortcuts I had learned because I had never practiced them. So, I decided to start creating simple shapes and trying to perform operations on them while learning how Blender works in parallel, aiming to better retain the information I was learning. After some time, I started creating characters by combining basic shapes.
I started by taking spheres and stretching them to give them the shape I wanted. Then, I tilted and moved these spheres to combine them. The result gave me a reference body that provides an idea of the future appearance of the characters once completed.
Given the game's concept—a tag game or "jeu du chat et de la souris" (cat and mouse game)—two distinct and easily recognizable characters were needed so that players could tell at a glance which is the hunter and which is the prey.
One of the characters had to be a mouse (the prey), and the other a cat (the hunter).
Here are images of these two 3D models:

The mouse:



The cat:



- **Map**

Given the difficulty of using and learning Blender, I haven't made progress on the map yet. I've focused more on manipulating shapes to create characters. However, I'm starting to think about the map layout. For example, I plan to create buildings of varying sizes and many horizontal or vertical obstacles that would need to be avoided or used to gain an advantage.

Gregory-Lucas CLEMENT – *Team Coordinator*

CONCLUSION

In this first part of the project, everyone tried to learn as much as we could to build solid foundations for the next steps. Even if there were some incidents, we kept going and tried to do our best. We made a table to represent our progress and our objectives for the next steps.

Tasks	Soutenance 1	Soutenance 2	Soutenance 3
Movements	15 %	50%	100 %
Menus	40 %	60 %	100 %
Multiplayer	80 %	70 %	100 %
Sounds	40 %	65 %	100 %
Map & 3D	15 %	60 %	100 %
Website	33 %	75 %	100 %
AI	0 %	50 %	100 %
GameMode & HUD	20 %	60 %	100 %

Our progress **has** been below our expectations for the Movements, the 3D, the AI, and the GameMode & HUD. On the other **hand,** our progress **has** been higher than our expectations for the Multiplayer and the Sounds. We were a little bit naive at first. Thus, we are going to focus a lot on AI, movements, and 3D, as well as on progressing in all the tasks.

BIBLIOGRAPHY

https://docs.panda3d.org/1.10/python/index
https://www.w3schools.com/python/
https://www.youtube.com/
https://github.com/
https://stackoverflow.com/questions
https://www.youtube.com/watch?v=_5Js5pbvFSw
https://www.youtube.com/watch?v=1FGWgaCyE8E&list=PLuine2he2FmOY1ILTDc1OR9vHgIMBw4_W