ReConnect

First Technical Defense Report

Louis Harel

Yoann Balasse

Victor Tendron

Adam Franco

Maxime Houwenaghel

A1 EPITA Paris

Tuesday 14th January, 2025

Table of contents

1 Introduction

1.1 Report presentation

This document is the defense report of the ReConnect project, led by LYVAM Studio, it explains the progress of the project from its start in November 2024 to the first defense in January 2024. It describes the difficulties and issues encountered on each task and mentions planned improvements.

1.2 Quick reminder of the nature of the project

ReConnect is a 3D single or multi player educational video game project. It takes place on an extraterrestrial planet named Edenia. The player is sent to this planet for communication maintenance. His goal is to fix these communications and repair his ship so he can return to Earth.

To do so, the player will have to solve electrical and electronic puzzles to evolve in the game and progress. The further the player advances in the game, the more difficult the puzzles will be. At the end of the game, the player will have acquired electrical and electronic skills equivalent to those taught in high school.

2 Teamwork Organization

2.1 Version control

Version Control is an essential concept for any coding project. It is a practice that consists of tracking different versions in the history of source code files, often text files.

At the beginning of our project, before starting coding, we had to decide how we would work together and how we would synchronize our work. We considered two ways to do so.

The first was to use Git with an application such as GitHub or GitLab that could allow us to synchronize, share, and manage our work. The second one was using Unity Version Control System.

We eventually chose to use Git because we were used to it thanks to our previous personal coding experience and our classes at Epita. Then we had to decide if we would use GitHub or GitLab. We chose GitHub because most of us had already used it before, unlike GitLab.

Additionally, GitHub provides various features to enhance productivity and communication among the team, such as Issues reports that help us keep track of bugs, feature suggestions, and assign people to their resolution.

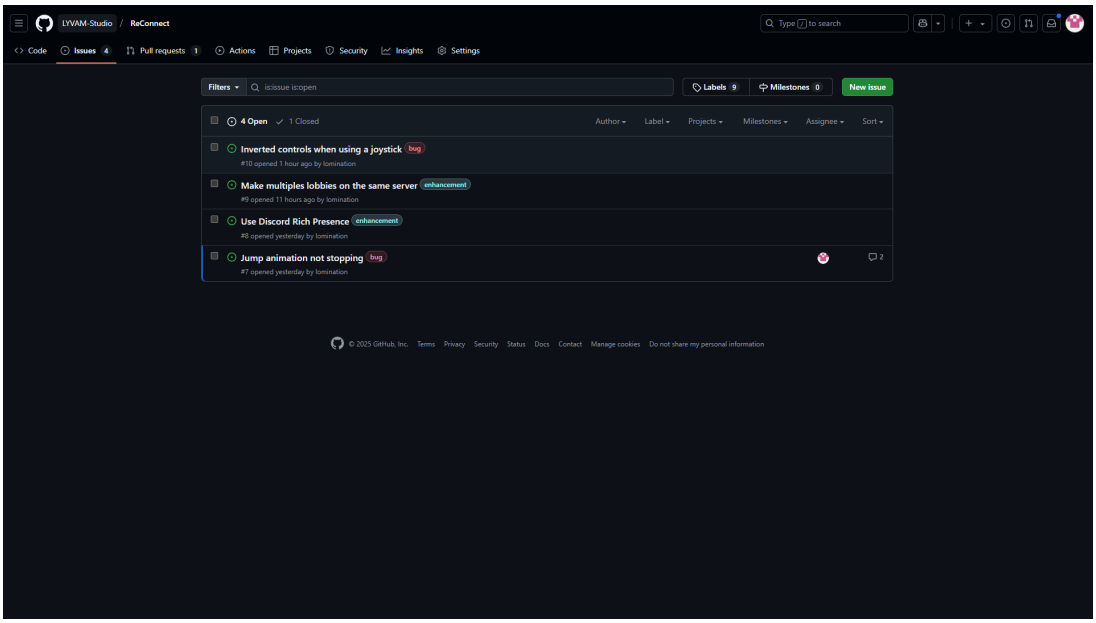

Figure 1: The Issue tab on the Reconnect Github repository

Our studio also set up Git LFS (Git large files storage) to be able to use Git with the large binary Unity files. To manage them, Git needs a plugin because otherwise, it can only handle limited sized files.

To isolate the different tasks during the development process, we use branches. This permits us to avoid modifying the same code at the same time and having to deal with merging conflicts. When a feature is finished, we create a pull request, and the code is reviewed by other team members before being integrated into the main branch.

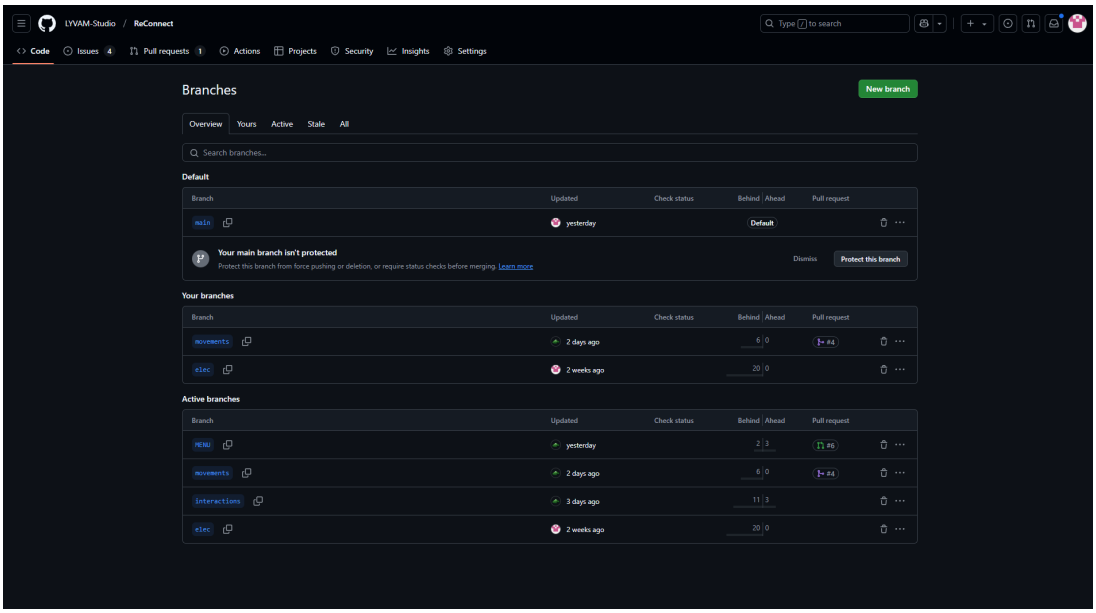

Figure 2: The different branches we currently have on our repository

2.2 Continuous integration

We wanted a faster mechanism to build our project than to do it manually from the Unity editor. Since it is especially important for us that our project is multi-platform and our members use Windows Linux and Mac, we also had to build for all these platforms efficiently. Thus, we made a continuous integration thanks to the CI tool provided by GitHub.

First, we created a workflow to build for Windows, Linux and MacOS. It uses various GitHub actions such as: the checkout action – that allows the CI to have access to the repository, – the cache action – that allows libraries to be stored to avoid their installation each time the workflow is run, – the unity-builder action – that allows to build any Unity project for different platforms – and finally, the upload-artifact action – that allows to make the builds available from the GitHub website.

Then we needed to build a server version for our server. Although a server could be run from the normally built client, a proper and distinct server version allows to directly run a server without having to click on any button and to run on headless servers i.e., servers that have no graphical environment. This is important since most of the servers do not have a graphical environment for efficiency reasons.

To do so, we created a second workflow to build this server version of our game. We did it by giving argument to the unity-builder action, indicating that the game should be built for a Linux server.

We chose to make the server version only available on a Linux machine since most of the servers run on Linux and the server that we own is on Linux.

Finaly, we made a workflow to be able to run both previous workflows at once by clicking only on one button.

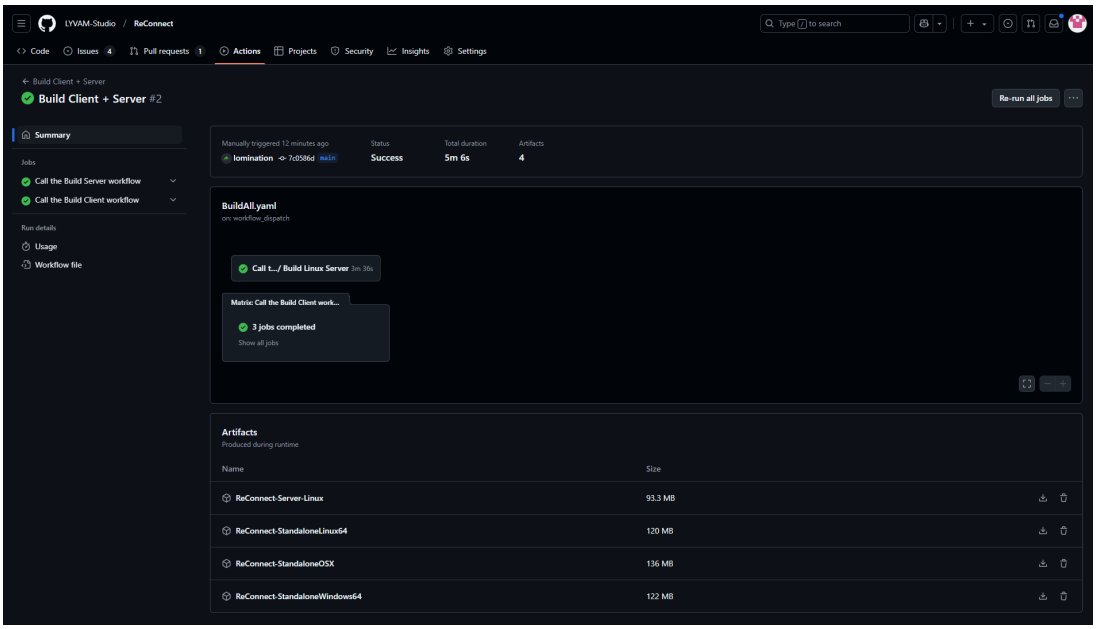

Figure 3: Our one-click build workflow in Github Actions

3 Multiplayer and Networking

We began the project with multiplayer and networking because it had been recommended by various people including Epita's students and teachers, to multiple of the members of the studio. Indeed, creating a single player local game as a first step and then, trying to migrate to a multiplayer game seems to be a dangerous and fastidious thing to do.

3.1 Difference between networking and multiplayer

Although they go together, multiplayer and networking are two distinct concepts. We are going to explain each of them in the context of video games.

- For a game to be a multiplayer game, multiple players must be able to play together at the same time.
- However, networking is about multiple machines to be synchronized over the network i.e., the internet here. Note that a game can be multiplayer without being networked (for instance, Untitled Goose Game, Overcooked 2, It takes two). In the same way but more rarely, games can be networked without being multiplayer.

3.2 Package choice

Adam being the person responsible for the multiplayer and networking, he tried first the PUN 2 package (Photon Unity Networking 2) with the advice of Maxime who is substitute of this task and Louis who has already had some experience in making games with Unity. The benefits of PUN are the worldwide community that comes with many resources, tutorials, and examples. So, Adam started following a tutorial made by SRCoder on YouTube. However, we realized that PUN 2 was outdated and so were the available resources. PUN 2 is still supported by the newer version of Unity, but has no new features since 2019. The main issue with PUN 2 was that there were no resources with the newest version of Unity. In addition, Unity is a software that evolves constantly and quickly, making old resources difficult to use. This led us to choose another networking package.

Then, we looked for another networking solution. We discovered that Photon developed new networking softwares to replace the old PUN 2: Fusion and Quantum. However, choosing them seemed risky in our opinion due to their lack of documentation, tutorials, and resources in general. We also read in some forums that it was often not recommended to begin Unity networking with these packages due to their technicality.

Eventually, we chose Mirror. Miror is a famous library for Unity networking games. It has a large community of users and contributors, as it is an open-source networking solution.

Figure 4: Mirror is a high level Networking library for Unity

3.3 Networking setup

More concretely, to set up a multiplayer game using Mirror we initially had to create a player prefab. A prefab is a Unity object that is not directly present in a scene but that can be instantiated and spawned in any scene thanks to a script or certain components. In this case, the player object must be a prefab because we do not want to have a player in the scene by default when a server is launched, and we want to create a new player object each time a client connects to the server.

Secondly, we had to configure the NetworkManager object. It is the object responsible for the global networking management. For instance, it has functions to run a server, connect a client to a server, or run a host. Configuring this object consisted in choosing the network protocol (KCP), referencing the player prefab, setting up the spawner method and choosing the default server ip address and port.

Although we will probably add some features to the networking part, to synchronize objects that we will add later, we consider the main part of the networking task as done at 100%.

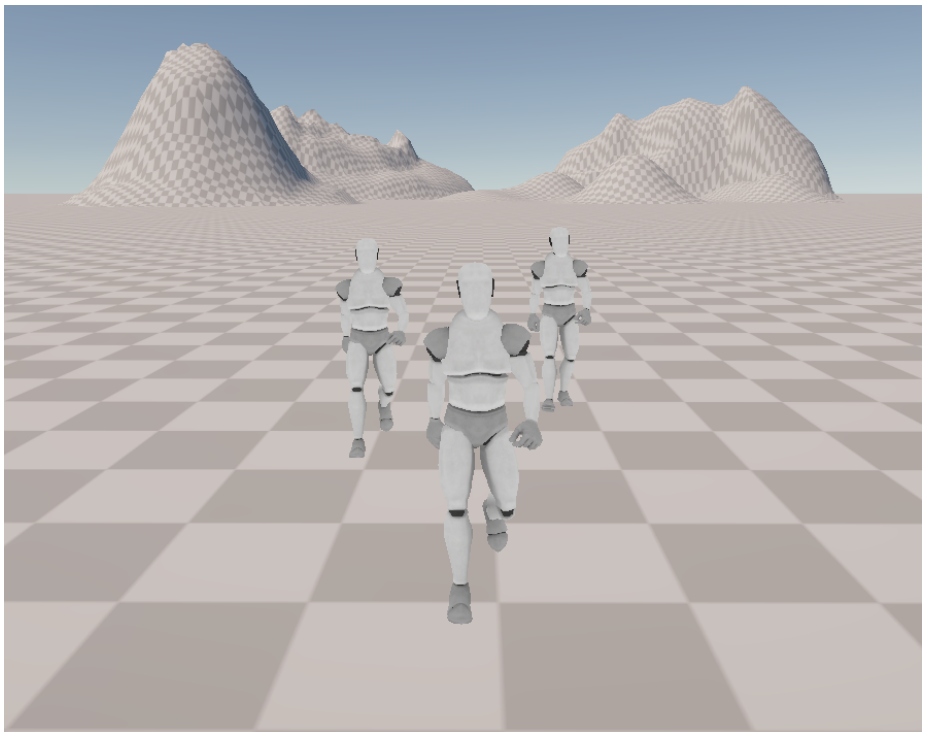

Figure 5: 3 players connected to a distant server walking together in the same game

4 Menu and User Interface

The development of the menu and interface focused on creating an intuitive and visually appealing experience for players, while ensuring functionality for single-player and multi-player modes.

4.1 Current state of advancement

Our menu is composed of three pages: the main page, the multiplayer page, and the settings page. The main page is the one displayed when the game is launched. It has four buttons:"Singleplayer", "Multiplayer", "Settings" and "Quit". The "Singleplayer" button allows the player to run the game in host mode i.e., creating a local server and connecting a client to it. The "Multiplayer" and "Settings" buttons go to the multiplayer and settings menus described below. The "Quit" button basically quit the game.

Figure 6: The main menu of the game

The Settings interface contains various input formats that are not yet linked to real settings. It also contains a cross on the top right corner to go back to the main menu and a full-screen mode toggle.

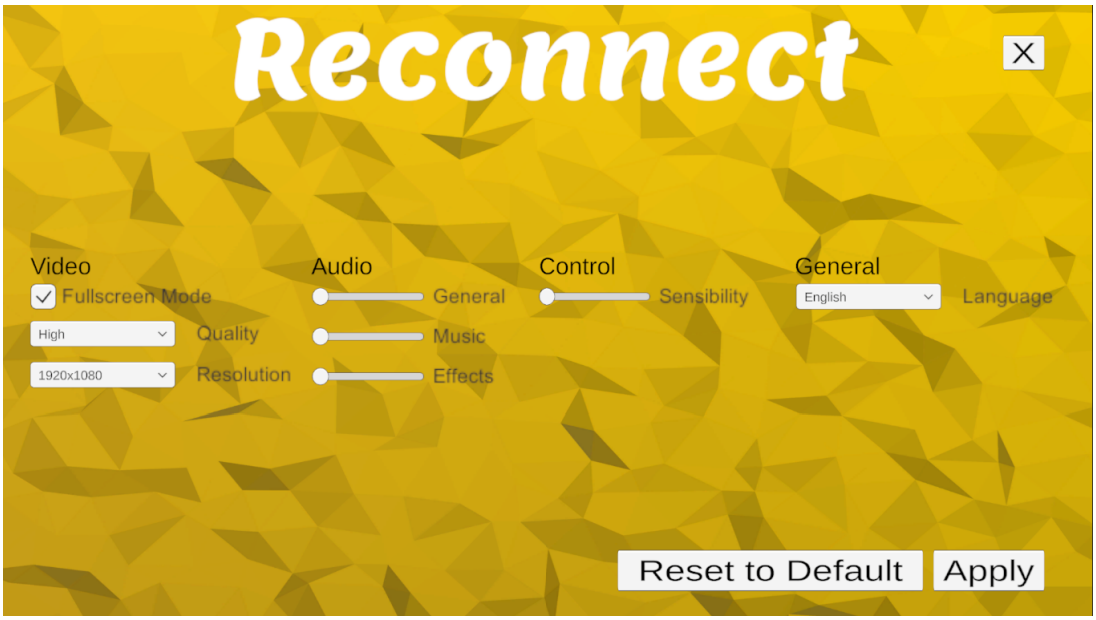

Figure 7: The settings menu of the game

The Multiplayer menu, accessible from the "Multiplayer" button of the main menu, allows the player to choose the ip of the server they want to connect. They can then connect it by clicking on the "Join" button or go back to the menu by clicking on the "Back" button.

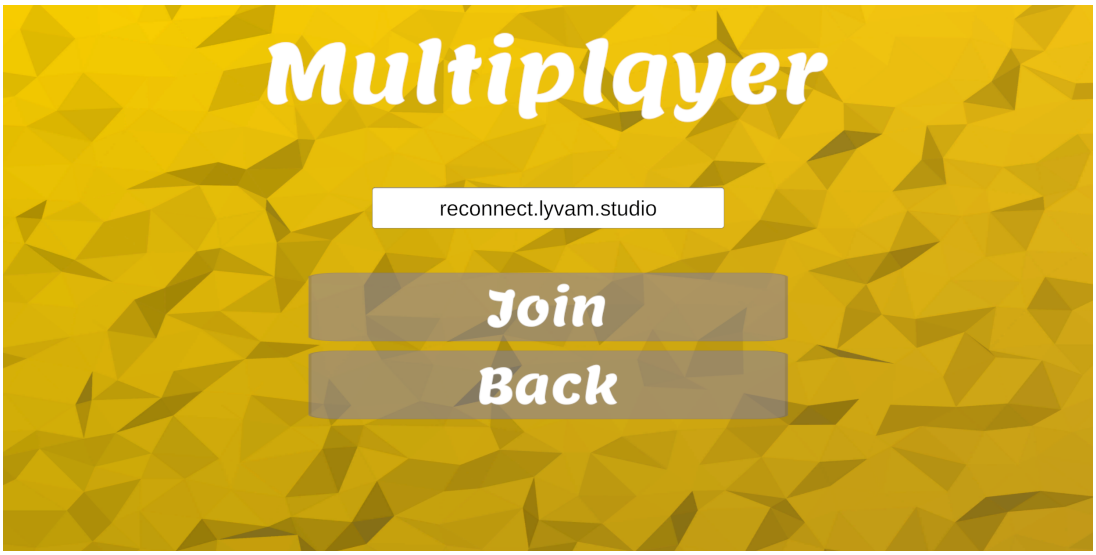

Figure 8: The multiplayer menu of the game

4.2 Future improvements

While the new menu structure is in place, some buttons, such as the "Singleplayer" and "Multiplayer" buttons, are not yet fully functional. Compatibility between the menu interface and the networking system is a priority.

In addition, we can enhance visual for menu interactions by adding features such as hover and click animations. We can also allow the player to change the key bindings.

5 Player Functionalities

5.1 Context

In our video game, the player embodies a human character who is the only survivor of a rescue mission. His goal is to understand what happened to a crew of scientists who were located on an exoplanet to study its viability to accommodate the Earth's human population. The character is thus a key feature of the video game, and it seemed quite difficult to begin the creation of the several levels before having a completely playable character with various movements, speeds, and animations.

Because this task was not really attributed to anyone, Yoann, attributed to the level design, decided to develop these features that are required by his tasks.

5.2 Requirements

At the beginning, we discussed the characteristics of the player's movements and playability.

We decided that they need to have a third person camera, including all the challenges this feature requires (anti-collision, anti-occlusion, movement consistency, ...). We also wanted the player to be able to walk, run, and crouch, each of these moves requiring different speeds, and to jump to be able to climb on some future environment objects.

Additionally, for the realism of our game, we needed to play some animations on the character's body to illustrate the walking, running, crouching, jumping and idling actions and some combinations of these actions such as crouch walking, or crouch idling.

5.3 Third-person camera

5.3.1 Choice of using Cinemachine

First, the camera scripting would have required extensive work and feature development, troubleshooting to provide a suitable and stable camera system. The time that we would have spent on this part of our game could have been used for more specific functionalities of our game such as the electronic simulation, the puzzle design, etc. For these reasons, we decided to use Unity's Cinemachine package which is dedicated to offer complex and complete camera systems easily so that we can focus on developing game-specific behaviors and functionalities.

5.3.2 Free look camera

We used the Free Look Camera with Orbit Camera, which perfectly suits our game needs. The 3 rings that we can set with different radius and height permit to define a sphere the camera will stay on when moving.

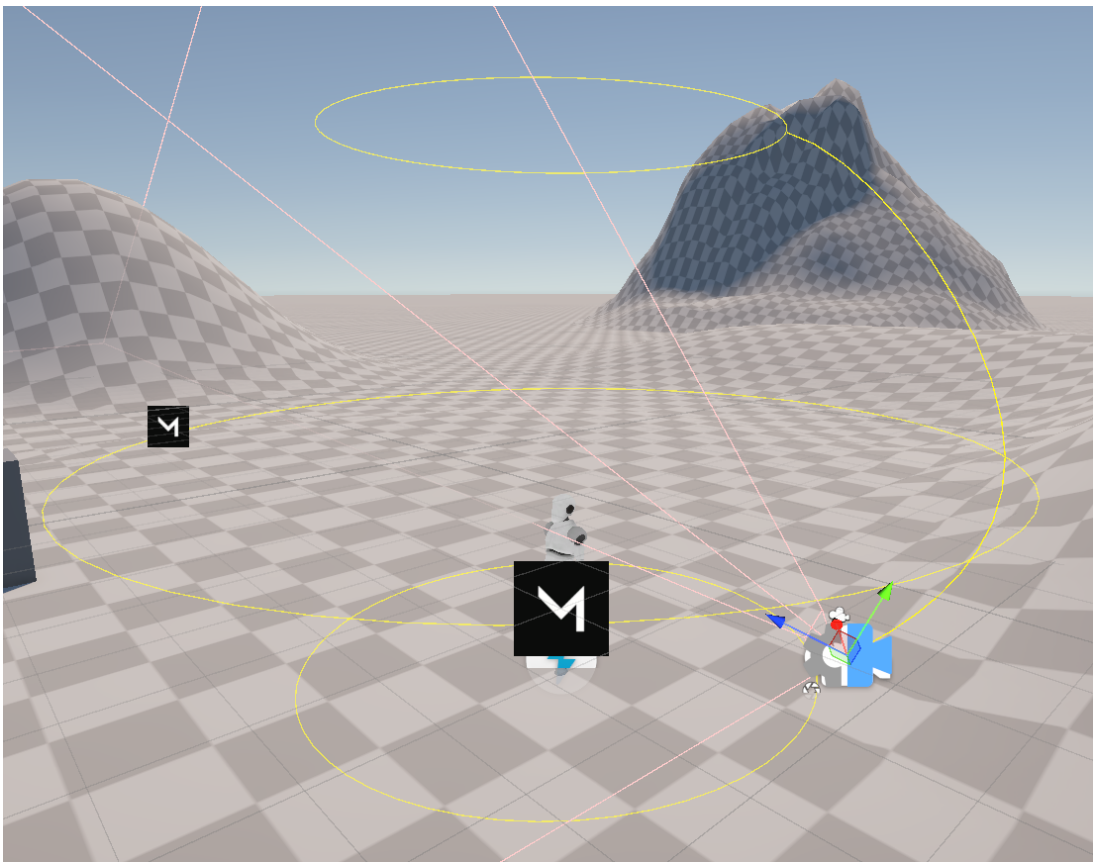

Figure 9: The 3 rings defining the orbit on which the camera moves

5.3.3 Deoccluder and decollider components

Additionally, we configured a Deoccluder Component from Cinemachine to preserve sight with the player and a Decollider Component to prevent the camera from entering objects. The disocclusion strategy used is to Preserve the Camera Height, this means that when an obstacle is in the line of sight of the target (our player), we want the camera to move to an alternative position while staying at the same height to avoid the obstacle.

5.4 Player movements and animations

5.4.1 Temporary 3D model for the character

Then, for the player movement, we needed to have a character model with a skeletal structure that would permit it to be animated. However, the 3D team from LYVAM

Studio has not yet finished the modeling of the player. In the meantime, we decided to import a basic character model from the Unity Asset Store temporarily to continue working on the player movement and animation.

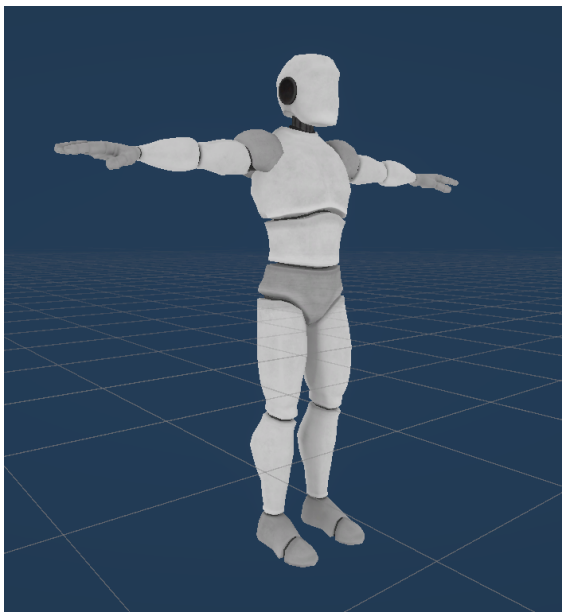

Figure 10: The character 3d model placeholder we imported until ours is ready

5.4.2 Cardinal movements

First, we implemented the basic cardinal movements, then we improved it by rotating the player accordingly to face in the direction they are moving.

The first difficulty we faced was to link the rotation of the camera with the rotation of the player. In fact, we wanted the character to look in the direction of the camera so that the movement "forward"/ "backward" / "left" / "right" feel more natural for the player.

5.4.3 Jump mechanic

Then to implement the jump mechanic, we needed to have the physics knowledge about the velocity of a falling object over time. This required some research in the field of physics, and finally we used for initial vertical velocity the final velocity at impact as described on this Wikipedia page and we subtract the gravity of Earth to simulate the gravity attraction.

Even though the jump works perfectly, it still lacks some improvements when combined with movement. Currently the player can be moved whilst being in the air, which is an unwanted behavior that we intend to fix soon.

5.4.4 Crouching mechanic

Finally, the crouching mechanic is quite simple: it reduces the speed of the player and plays the animation to make the player look crouching. This feature is still primitive and needs to be improved by modifying the character's collision box to adapt to the player's height when crouching, to enable him to move into a space where his original size would not have allowed him to go.

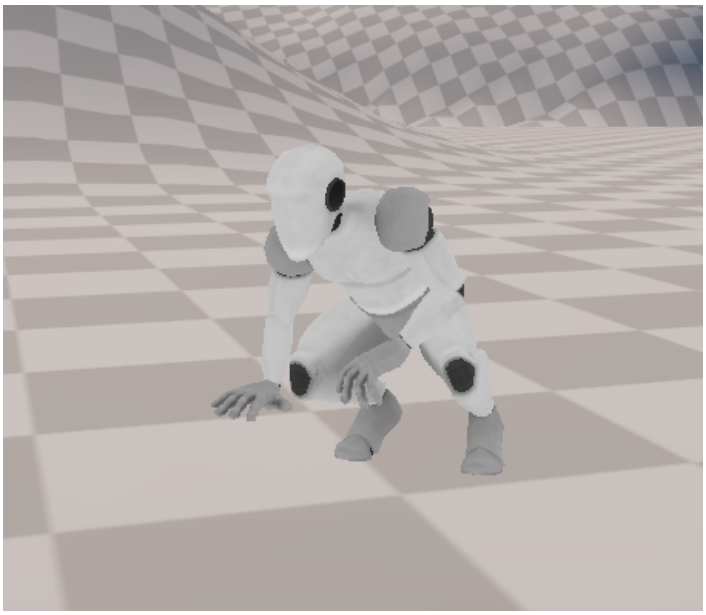

Figure 11: Our player crouching

5.5 Animations

For the animations, we used Maximo.com, a royalty free library of animations provided by Adobe. Once we configured the Animator Component in Unity to switch between the different states and play the animations accordingly. We used the new Input System from Unity that permits us to use a GUI to define our keymaps with multiplatform inputs.

5.6 Future features

Soon, we intend to add a sliding mechanism when the character is on a slope that is too steep. Additionally, we would like to make some animation of the player being happy that would be triggered when he finishes a level.

6 3D Modeling

Creating our own 3D environments is essential for the immersion and authenticity of our universe, so we first made sure we had a complete mastery of Blender, so we could make our assets, characters and maps all at once.

For instance, we tried to make a map of a planet that has been impacted by asteroids. It was important to make the seed of the plane a random seed such that it looks like more natural. To be able to do so, we used Voronoi diagrams (partitions of a plane) and set the colors of the map as height such that it creates summits and craters in a random way.

Figure 12: The test map from a near point of view

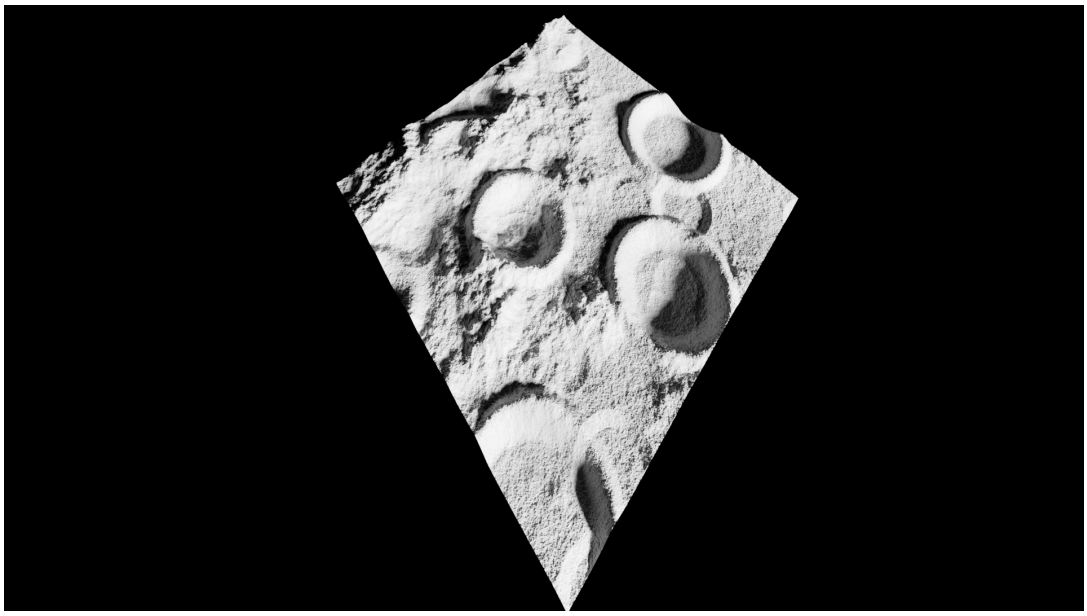

Figure 13: The test map from a high point of view

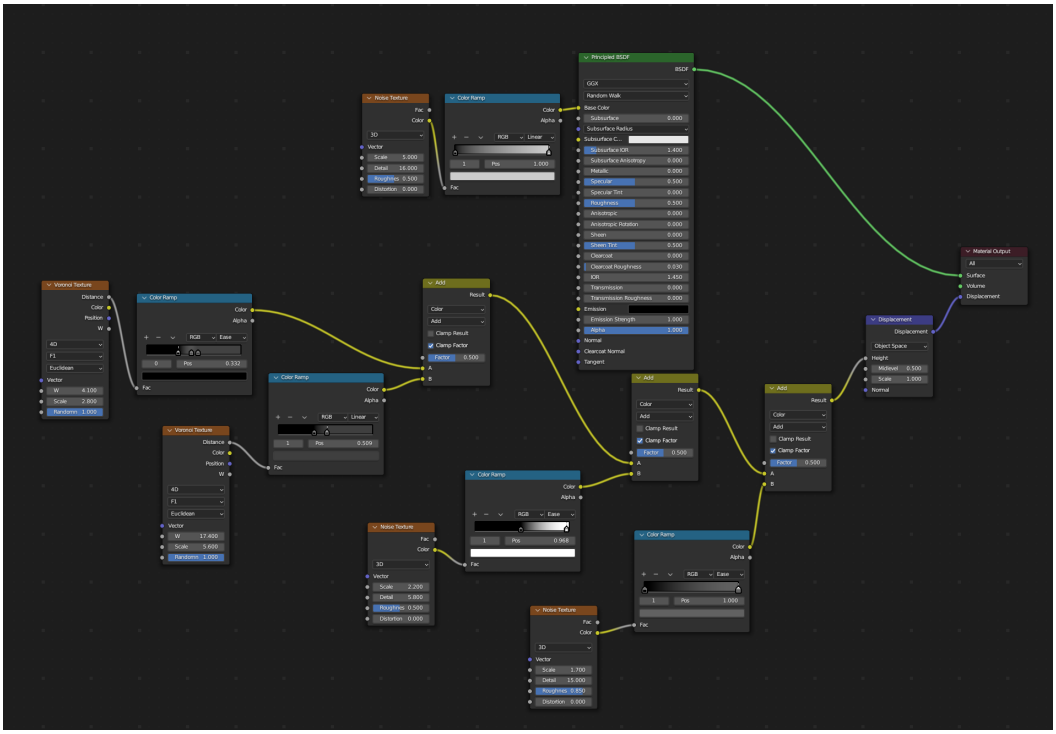

Figure 14: How the shading works

The figure above explain how we use the 2 Voronoi textures and 2 noise textures to set colors (from white to dark) as the factor of the distance (height). We also set the scale and the details of the map.

We did some tests for mobs and other assets but didn't finished them properly yet.

7 Website

The creation of a website aims to introduce our studio “LYVAM Studio” and our new game “Reconnect”. The website was created from zero in HTML and CSS because these languages are easy to understand for everyone. Also, we used Sublime Text as text editor during the conception. It is a simple software, quite clean in contrast to Visual Studio Code. Besides, we used Mozilla Developer Network Web Documentation (MDN Web Doc) to fix our issues.

The final version of the website is accessible at the following URL: lyvam.studio.

Figure 15: The home page of the website at lyvam.studio

7.1 User interface design

A few prototypes were created before the final design of the website. We decided to use Figma to conceptualize the different pages. Figma is a vector graphics editor and a prototyping tool.

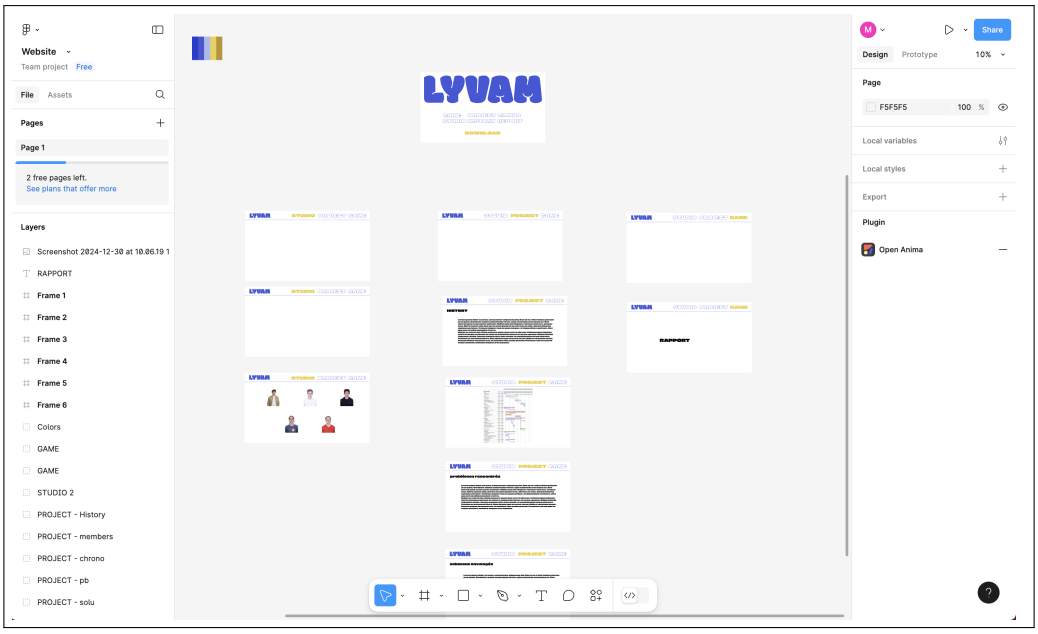

Figure 16: Website prototype made on Figma

We decided to make 3 separate pages, without mentioning the home page. A larger number of pages would tend to lose the visitor. In addition, this allows the visitor to understand the theme of the page and where to search for the desired information quicker.

First, the "Studio" page introduces LYVAM Studio values and our team members. LYVAM Studio develops educational games, mostly intended for high school students, which aims to facilitate the learning of subjects through playful practice.

Second, the "Project" page presents our new game "Reconnect" in development and the assets and software used in the conception of the game and the website.

Finally, the last page "Game" allows the visitor to download the game for Windows, MacOS and Linux. Moreover, the report of the first defense is available on this page.

The website is entirely responsive and adapts to every screen, computer and smartphone. Nonetheless, the website was not designed for smartphone users and the user experience is not completely fluid.

7.2 Domain name and server

The Studio has decided to get a domain name to find easily the website. In fact, the domain name `lyvam.studio` is more memorable because of the use of our studio's name.

This goes in pairs with the communication part of the project. Also, the domain name is used in the game accessibility. We added a subdomain name `reconnect.lyvam.studio` to simplify the connection to our game server instead of typing an IP address.

Additionally, the website is hosted and deployed by Vercel's servers. Vercel is a company that has developed the web framework NextJS and proposes Platform as a service. It is a type of Cloud Computing where Vercel maintains the servers, the OS and all the infrastructure.

Furthermore, the whole code of the website is available on Github, a platform that aims to manage projects and git version. It allows Vercel to upload the website's code and to deploy it on their servers. Thus, when an update is done on GitHub, the website is automatically updated.

7.3 First prototype

At first, we wanted to create a website based on the NextJS web framework. Using the documentation, the goal was to implement animations to improve the fluidity of the user experience. However, it took too much time to learn how a framework works and the specificity of NextJS. Thus, we went back to the basic development of a website by developing in HTML and CSS. HTML is the structure of the website, that is to say it allows us to create a paragraph, a title and implement an image. On the other hand, CSS embellishes the display of HTML by adding colors, spaces and by customizing the different elements of the structure.

7.4 Our font

Alongside the first prototype, we decided to create our own font called "Reconnected" to improve the immersion into the Reconnect's universe. This font manages only capital letters because we wanted it to be implemented in titles and names. We used Affinity Designer, a vector graphic editor, to create the font.

Nonetheless, once the design stage finished, we had few troubles with the implementation of the font in a correct format. That is to say, to transform a drawing in a Web Open Font Format (.woff) file. Thus, the creation of our own font is suspended until we find a solution.

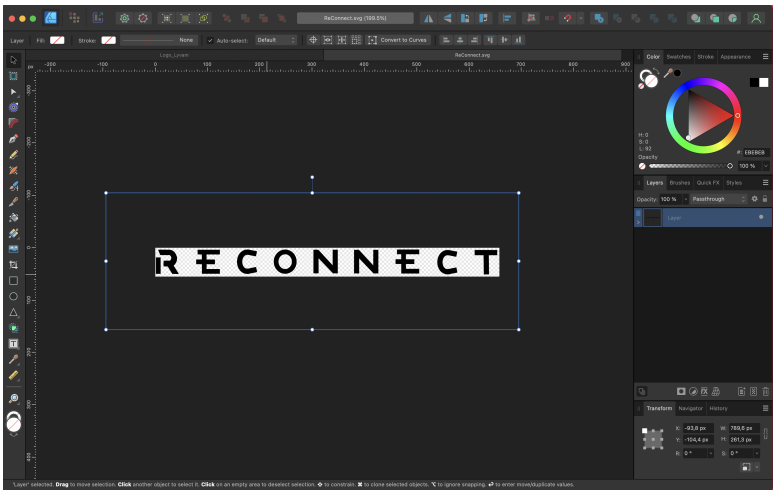

Figure 17: Prototype of our font "Reconnected"

7.5 Avatars

Finally, the implementation of an organization chart with photographs in the "Studio" page is more important that we thought. It shows the ambiance of the studio and how we see ourselves. Hence, we had to make an organization chart with photograph that fits with our values. Formal photos of our team would be very serious and will not reflect how the studio wants to reflect.

Then, the intervention of DiceBear is perfect. It is an open source avatar library that allows you to create an avatar. By customizing all the aspects of a head.

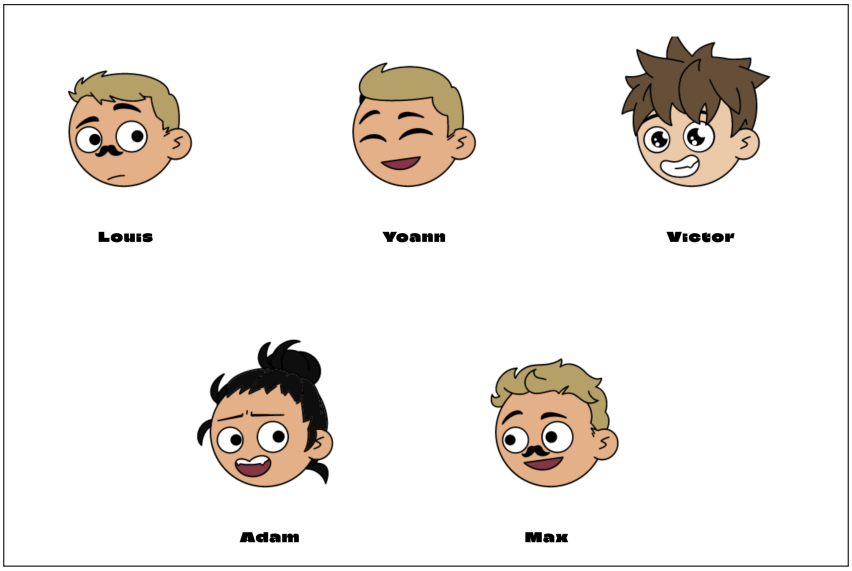

Figure 18: organization chart with photographs of LYVAM Studio

8 Advancement

The following table shows the current advancement on the project in comparison with the advancement that we predicted in our technical book of specifications.

Scenario	70%	55%
Menu	75%	75%
HUD	0%	0%
Physics	30%	30%
First tutorial	0%	100%
Music & FX	0%	100%
First level design	0%	100%
3D modeling	40%	50%
Interactions	0%	0%
AI	0%	0%
Multiplayer & networking	100%	100%
Website	100%	100%
Communication	0%	100%

Table 1: Current advancement compared to the planed advancement

9 Conclusion

For this first technical defense, we got the functional base of our game in terms of networking and player and camera movements. We can start a server, a client or a host (server and client at the same time) in the main menu of the game. Using multiple instances of the game, either on the same machine or on different machines, two clients can connect to a server and see each other synchronized over the network.

We got an advancement on the menus, but the newest version of these are not yet fully integrated to the main version of our project.

We are late on the implementation of the electricity and therefore the level design and the tutorials. This will be the priority for two of our members, so it be well advanced by the second technical defense in March.

We completed and built our website, which is available at the URL lyvam.studio. It contains information about our studio, our team, and our project, including an overview, this report, and downloadable versions of the game for Windows, Linux, and MacOS.