Project report - Rukmaksii project

Romain LE MIERE Lucas TILLY Alban NAULIN
Emmanuel VOUILLON

June 7th, 2022

Contents

1 Genesis of the project

1.1 Origin of the project

The origin of the project is a message from Romain proposing a video game project a little bit out of proportion. But he said "Avec unity c ez"[1].

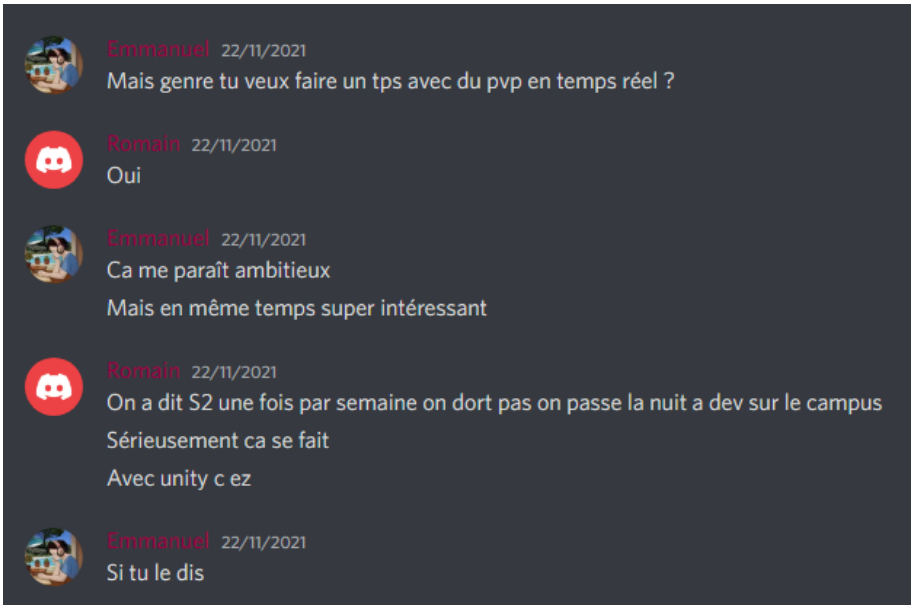

Figure 1: Picture of Romain's message

1.1.1 First brainstorming

Convinced by those words, we decided to do this project together and we did our first brainstorming. It allowed us to agree on what we wanted to do. During this brainstorming we drew a first draft of the map, which strongly resembles the final map, we made a list of all the features that would be in the game: classes, weapons, abilities... And we made a first schedule with time estimates, that gave us a first idea of the organization we should maintain throughout this project.

[1] With unity it is easy

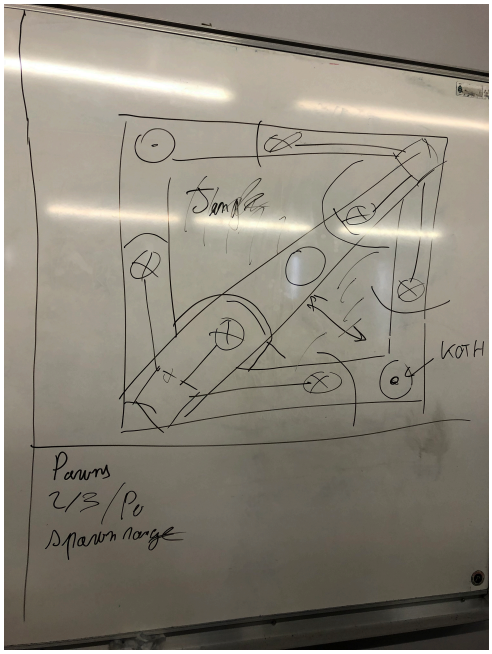

Figure 2: Picture of our first map

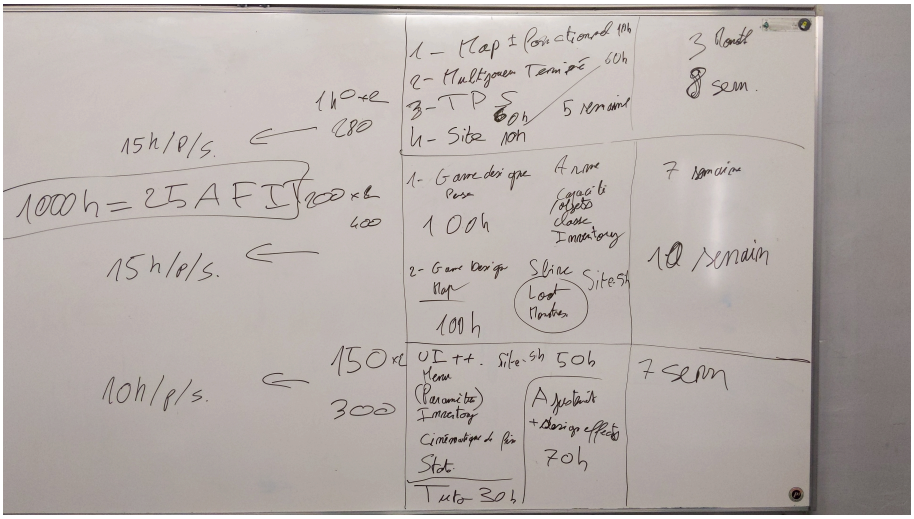

Figure 3: Picture of our first planning

1.1.2 Inspiration of the project

This project was greatly inspired by both Anthem and League of Legends. The idea of the project was to create a MOBA TPS(Third Person Shooter). One of the rare games of this type was Paragon : the MOBA TPS developed by Epic Games. This game and all the features in it are the result of compiled features we liked in our favorite games like Overwatch, League Of Legends and Anthem.

Figure 4: Screenshot of Anthem's gameplay

1.1.3 Origin of the name of the project

After an hour of brainstorming we had a general idea of what we wanted to do as a video game. Then came the dilemma of the name of the project. Since none of us had a good idea to propose we went to a random name generation site. After a few generations that didn't work out, the site suggested "Rukmini" which we liked but the "mini" didn't really match the excessive ambition of the project, so we decided to change the ending by replacing "mini" with "maxi" and then we adapted the spelling to make it really unique. This gave the name of our project: "Rukmaksii".

1.1.4 Work Schedule

In order to keep up with the amount of work we had to do to achieve this project, we worked on the game every Friday at school for at least 7 hours. From the beginning of this year, we worked every Friday night together.

Since we have great consideration for machine rooms we ate in the corridor in front of the class.

Figure 5: Picture of our diners in front of SM

In the final rush, the week before the final defense, we went every day and worked at least 10 hours per day in order to finish this project. This was not easy but it was very interesting and enriching.

1.1.5 Organisation

To keep a clean workflow, we decided to keep 2 main branches:

- develop: the base development branch shared among the members and supposed to be kept clean (without any bugs) at all time
- master: the branch kept for releases merged before every presentation or when a humongous feature has been implemented

Furthermore, we opened some issues every end of the week to specify what we had to do for the following week: an issue for each feature. With every issue opened, a branch was created, rebased on the branch develop, and added to the Github project board.

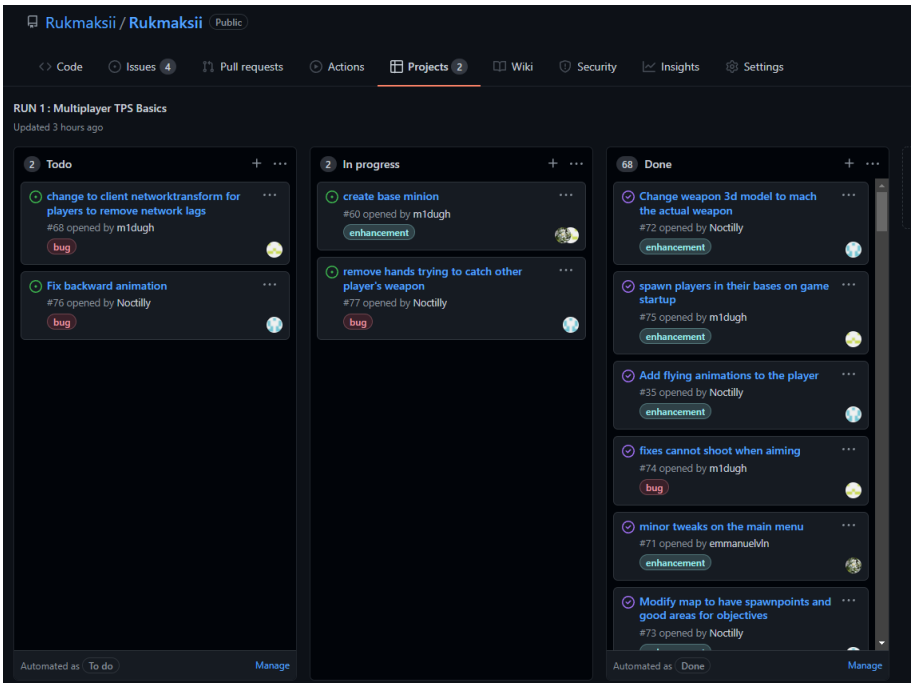

Figure 6: Picture of the project board

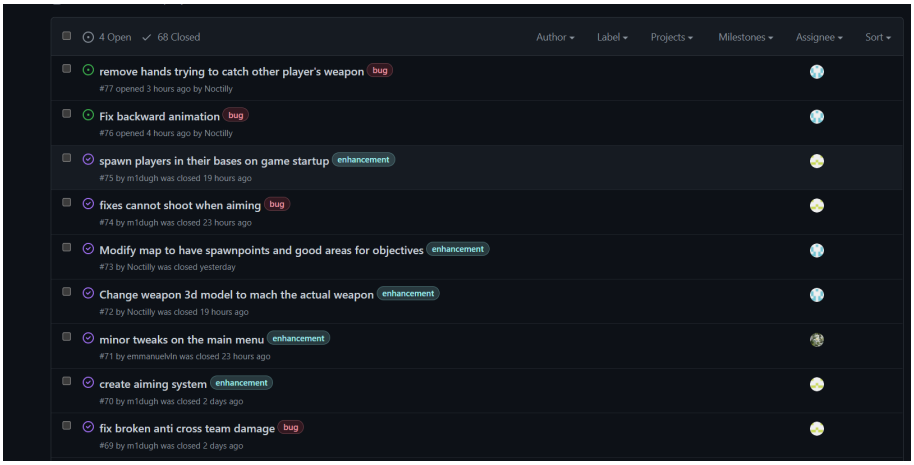

Figure 7: Picture of the issues

To avoid merge conflicts on the branch develop, we often rebase the branch we work on, named after the issues (e.g. bug#42). When it comes time to merge, we first merge with develop on our branch before merging in the develop branch. This allows to resolve merge conflicts outside of the develop branch, leading to a clean workflow.

2 Book of Specifications

2.1 Recap

The main aspect of this game was to be a MOBA TPS game built to be played either 2v2 or 3v3. The main goal of the game is to destroy the enemy's base by capturing points on the map which allowed a team to lower the enemy shield on the base to destroy. The game was also designed to give the player a unique experience each time the game was played by creating huge customization during the game through stat changes, changeable weapons or objects, or even classes.

2.2 Game Description

2.2.1 Player Spawning

When the game starts, the player is asked to choose an available class among a list of classes allowing him to play at his best during each game based on his liking. Once the player is spawned, he is given a few weapons and items to start the game depending on the class he chose. Moreover, only some classes can pick given weapons, giving a huge role to classes.

2.2.2 The Map

The map features three objectives to be captured by teams, a base for each teams and different obstacles such as houses, height variety and bushes to give the player a better immersion when playing.

The objectives are activated one at a time in a random order every few minutes and after a global announcement. If captured they give access to a shop and disable the enemy's shield.

2.2.3 Shops

When capturing a point, a shop is unlocked for all the players in this team allowing them to buy weapons and items. Each capture point has a bound shop which is available only when the the corresponding objective is captured.

2.2.4 Death of a Player

The player can be killed during the game. When killed, the player keeps his weapons but looses his items. A killed player is spawned again 5 seconds after his death in his base, with half his money. The other half is transferred to the player who killed him.

2.2.5 Classes

Classes are a way for players to match their mindset to a given playstyle in the game. A Soldier class which is an unspecialized class giving more choice to the player in terms of weapons but neither good nor bad in anything. A tank class with more health point and less speed but having access to more specialized weapons, and a Scout class, stealthier, faster but weaker than the other classes.

2.2.6 Weapons

The weapons can be dropped and picked up by any player in the game. However, only attributed classes can pickup weapons. Otherwise, it is transformed into money.

2.2.7 Items

Items are consumables that can be bought in shops or dropped from monsters. They give the player considerable advantage over an enemy player with no items. The applications of items are infinite, whether it is healing, increasing stats for some time, or even throwable items like grenades. They are a key point in the game. A player can drop any item on the ground that can be later picked up by any other player disregarding the initial player team or class. Finally, if the player already has the maximum amount of the item he wants to pick up, this item is transformed into money.

2.2.8 Abilities

Abilities consists in a mono-branch tree modifying permanently stats for the player during the game. It could either be the amount of damage taken, the duration of the jet-pack and so much more. Abilities are kind of buy-once-use-everywhere objects. Each class have a defined tree of classes it can

buy. at each layer of the tree, only one ability can be chosen, with a maximum of five abilities. Abilities allow the player to have a unique experience every time they play.

2.2.9 Minions

Minions are another key point of the game. They are AIs that can be spawned by the player. They can either protect the player that called them, defend a given area, or attack enemy players. They are of huge help in any situation of the game. Therefore, any player cannot spawn more than a given count of minions altogether.

2.2.10 Monsters

The monsters are autonomous, they spawn randomly on the map and as soon as a player passes too close to it, it chases him. If the monster catches up with the player, it inflicts damage, but if the player kills it, an item appears and the player can get it back. There is always the same number of monsters on the map, as soon as a player kills one, another one spawn randomly on the map.

3 Game Development

3.1 Introduction

This section covers the whole development of the game from the start of the project up to the end. Since all issues where covered in groups, we decided to do the recap of the whole project covering the game advancement as a group rather than as issues covered by a sole member of the group each time. This section will cover chronically the advancement of the project.

3.2 Basics of the game

3.2.1 Networking

We decided to implement the network features right at the beginning of the game to avoid taking the risk of having to redo all the code we had done before. The networking solution we decided to use was the solution backed by unity called Unity netcode for GameOject, an evolution of the well-known MLAPI opensource library. Since this library is still in development, we experienced quite some trouble when working with it (including documentation examples not working due to 2-weeks-basis release) which forced us to try and play with the library to understand its working.

The first thing that was implemented in the game was a moveable player in Network which gave us good foundations for the rest of our game. Since we cared about performances, we preferred not to overload the server with useless data transfer and therefore decided not to attribute a camera for each player in the script. A jump was also implemented for the players.

3.2.2 Camera Controller

Since the game was a TPS, the camera had to follow certain specifications like rotating around the player. An easy solution would have been to use the cinemachine library with its free licences. However, we decided that cinemachine would not allow us to do everything we wanted. Therefore, a new script for controlling the camera was added to act as a reliable TPS Camera manager.

The given script allows the camera to move of a given angle and to stay always at a given position from the player. This script even allows the main camera to be bound to any player or minion resulting in great possible features.

3.2.3 GameController Basics

To handle the whole game, a game controller script was added. The role of this script was to handle proper spawning during the game, objective locking and unlocking, shop opening control and any feature linked to the game in itself.

3.2.4 Prototype Map

Since the game only featured working multiplayer movement, a map was added to the game to have a better overview of what the game would be. A 3D map exactly following the specifications book was added as a test map.

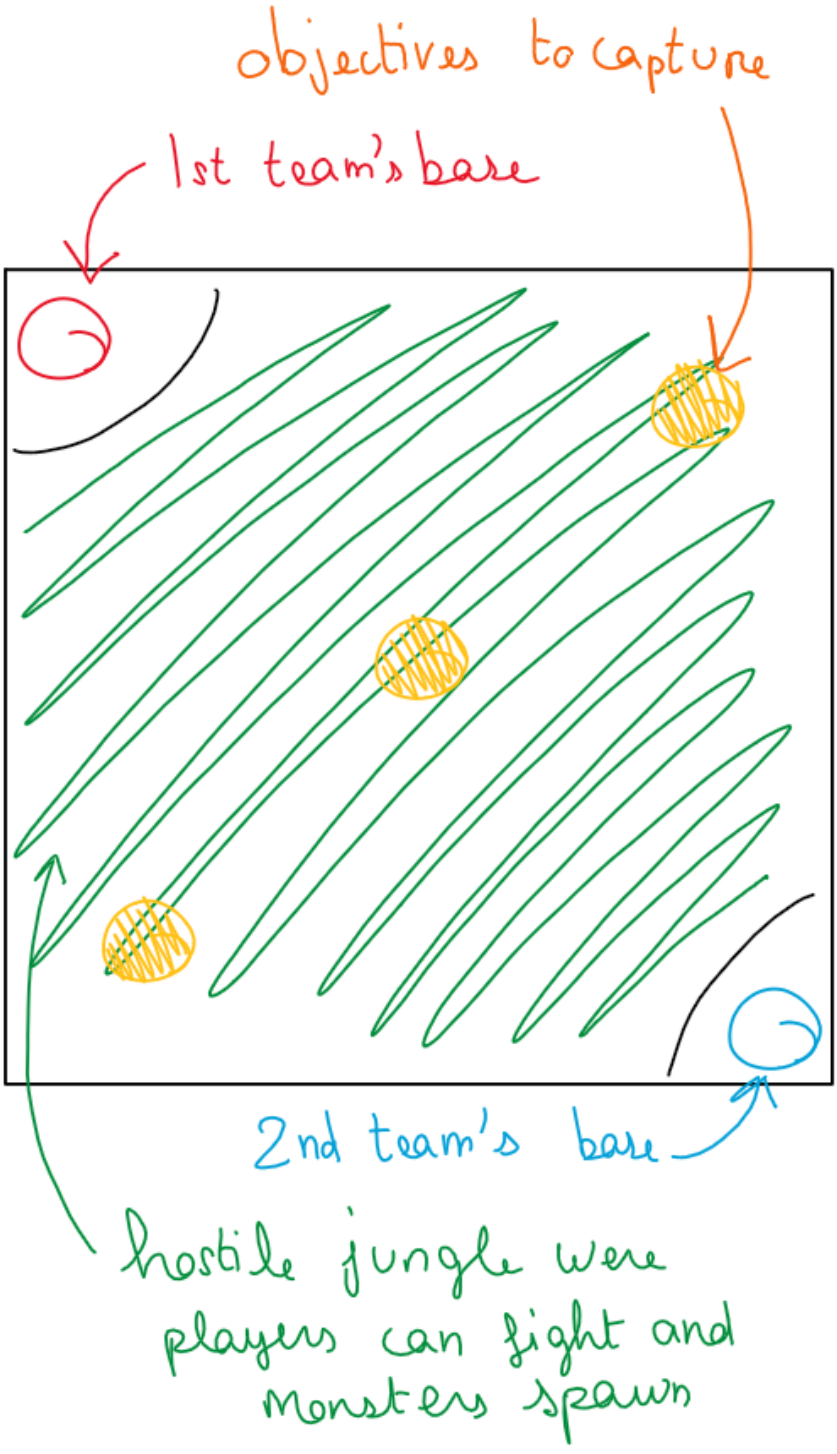

Figure 8: Simplified drawing of the arena

Figure 9: A top picture of the prototype map

3.2.5 Flying system

In order to give our game of more dynamic gameplay, a flying system was implemented in the game. This flying system was intended to be used as a jet-pack allowing the player to fly for a given amount of time consuming fuel. This fuel is reloaded when not flying.

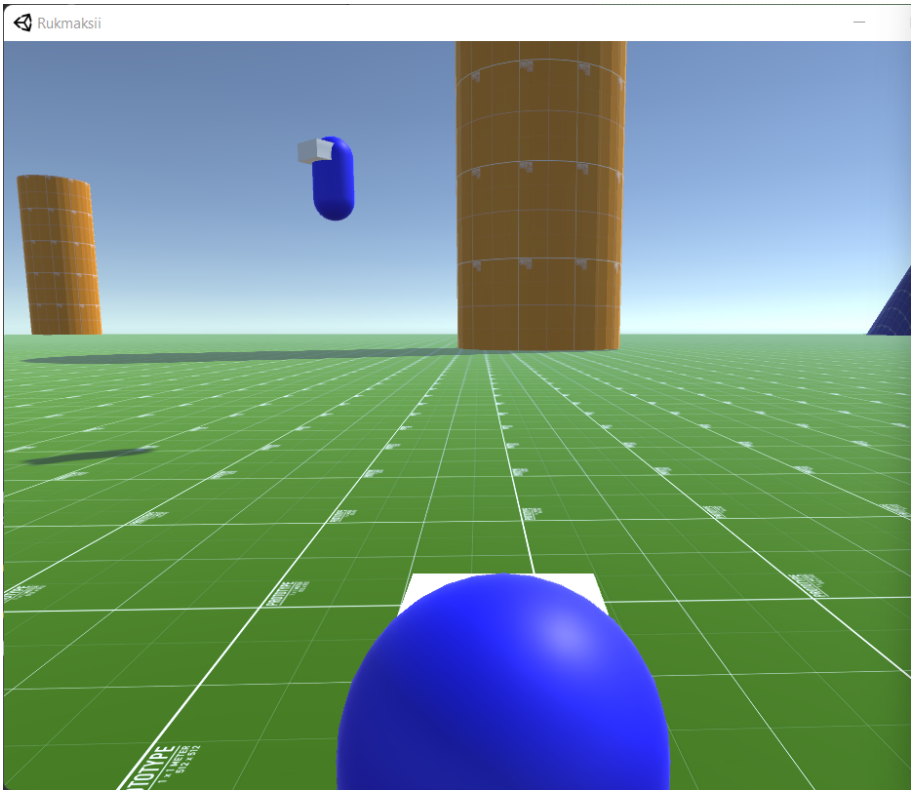

Figure 10: A prototype flying system for the base player at the very beginning of the game

3.2.6 Dash Control

A new movement was added to the game to make it more dynamic : the dash !! The dash allows player to have a better feeling when playing the game due to the considerable acceleration given by the dash.

3.2.7 Damaging system

With a working movable player, a proper health and damage system had to be implemented. Since it is a multiplayer game, the health had to be synchronized between all clients starting with a bunch of synchronized variables. Since the library we used for networking was at a lower level than Photon for example, a lot of work had to be handled by us concerning value synchronization resulting in better performances since only the required properties were synchronized over the network.

3.2.8 Prototype HUD

A prototype HUD was quickly implemented in the game so that the player can have an overview of how he performs in the game. This basic HUD allowed the player to know the remaining fuel and bullets he has and his current health.

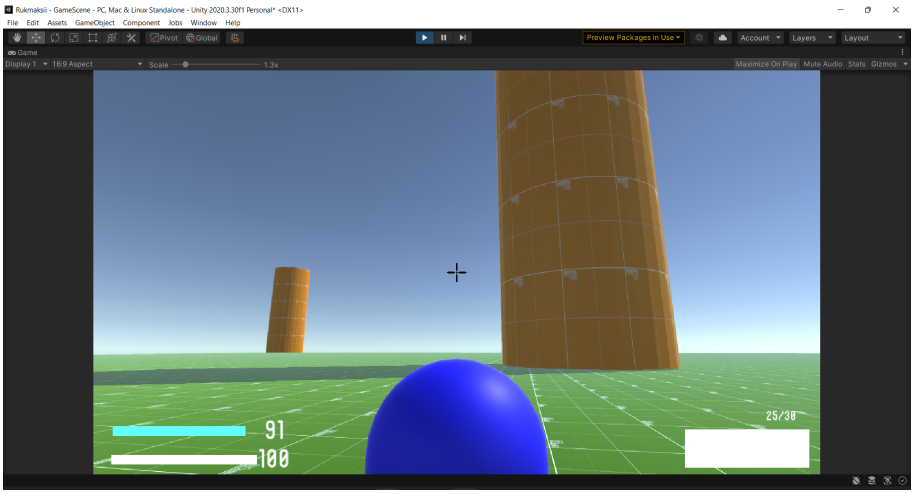

Figure 11: The first HUD in the game

3.2.9 The monsters

As a proper MOBA, the game has to feature a large amount of monsters dropping loots as they were killed during the game and giving the player a chance to come back in the game.

Monsters were approached as Jungle guardians defending their own territory. They have a basic AI system making them go straight toward the player, but it is already enough to make the player feel chased and force him to be aware of his surroundings.

Figure 12: Picture of the first in-game monsters

3.2.10 The shooting system

To make the shooting system feel good to the player, we decided, right from the beginning that there would be no time between the moment the player shoots and the moment the bullet his the target. That is why we decided to use the ray casting method to implement this feature.

3.2.11 The inventory

To have weapons, items etc, the player first needed an inventory to store them. This is why, even though it was empty for a while, the inventory system was created soon with what was needed to store any kind of objects in it. The main use of the inventory was to handle all the renderers for both the items and the weapons to avoid having multiple renderers overlapping each other when a weapon was selected, and it was responsible for giving

an abstraction layer for the BasePlayer considering the huge amount of both items and weapons that can be stored by the player. It handle both the player mode (item when player was using an item and weapon when player was using a weapon). The inventory was therefore responsible for linking over the network the items and weapons bound to the assignated player.

3.2.12 The class system

As for the inventory, the class system was not used right after its creation, but the implementation was there with a test class, allowing us later on to add new classes to the game easily and quickly.

3.2.13 HUD enhancement

With a solid weapon implementation now in the game, the Heads-Up Display needed an overhaul. First the crosshair was changed to improve visibility. A hit marker, an indication appearing when the player registers a hit on another player or a monster, was added around the crosshair to improve feedback. On the bottom-left corner the weapon currently held in hands is now displayed properly with an icon. Finally an ammunition counter displaying live the size of the weapon's magazine and the number of bullets was added on top.

In addition, a dash indicator was created to show whether the dash is on cooldown or not, and if it is, show the remaining time before being able to use it.

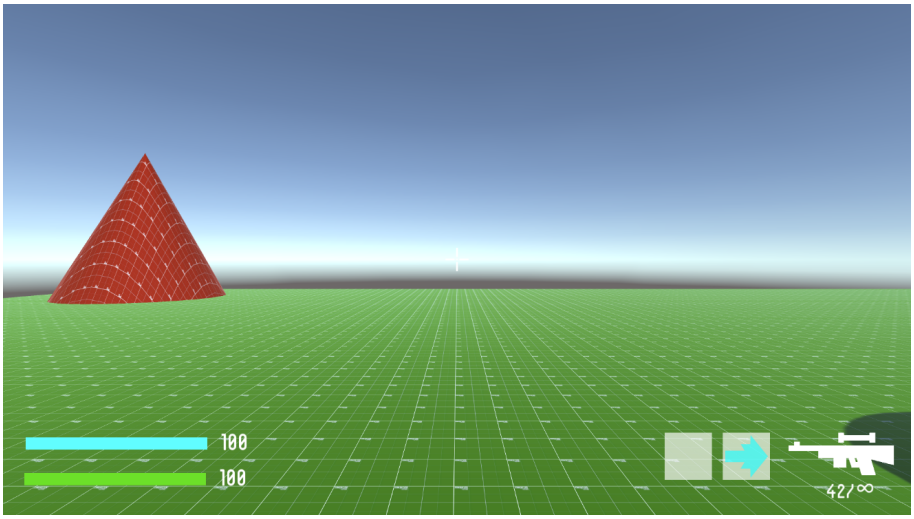

Figure 13: Picture of the updated HUD

3.2.14 3D models

To make the game more pleasant to the eyes and to bring some dynamic, models for the player and monsters were added along with some animations. For the player, we wanted some kind of robot, and during our research, we found a free 3d model of Iron Man which we could use in a non-profit way. Our research for the monster's 3d model went through approximately the same process. We didn't have an exact idea of what we wanted and we found a free 3d asset which had no textures.

3.2.15 Death screen & re-spawn

With being able to shoot others and take damage, death was added alongside the possibility to re-spawn and a death screen. When a player's health reaches zero points, he dies whether he was killed by a monster, another player or himself. This is when he is shown a death screen consisting of a simple black screen indicating that the player died. In addition, a timer appears on the screen, indicating to the player the time he has left before spawning again. While the death screen is displayed the player cannot move, shoot or do anything else than waiting for the re-spawn timer to run out. At the end of the timer, he is summoned back at his base with full health.

Figure 14: Picture of the death screen

3.2.16 Animations

For the animations, there was a lot of problems, mostly due to the non-user-friendly animator tool from unity. In fact, finding the animations was not very difficult thanks to a website providing the animations we wanted. The hard part was to link those animations between them smoothly and to make them work across the network. But even with those difficulties, we managed to add animations for all of the actions of the monster and the player, including walking, running, flying, etc. and we made them work well between them and even on the network.

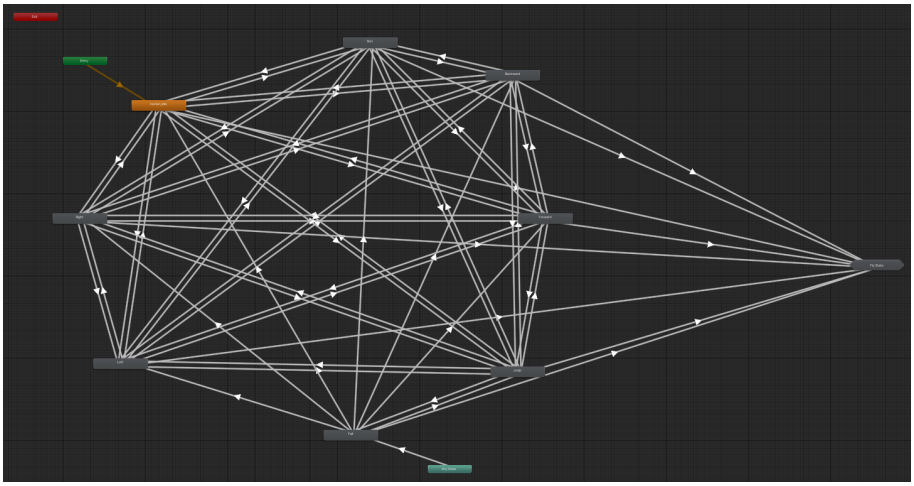

Figure 15: Picture of the Unity Tool Animator we used

3.2.17 Pick up and drop

We decided to implement a pick up and drop system for both weapons and items. Since both of these items had to be independent from the player yet linked to the right player when picked up, the transforms had to be client-authorative when picked up by a player and the transform rights had to be given back to the server when the item was picked up.

3.2.18 Item implementation

To enhance the game experience, items were implemented. Items were developed to be pickable and droppable, therefore exchangeable between players. For balance purposes, a max amount for each items were implemented.

3.2.19 Main menu

To greet the player when he first launches the game we needed a proper main menu so we created a new scene dedicated to this task. At first, this main menu contained four buttons: the first two to play the game, either as a host or as a client. The other two buttons were an option button and one to quit the game. Upon clicking on one of the first two buttons, the user was

directed to a new menu where he could select the class he wanted to play and finally join the game.

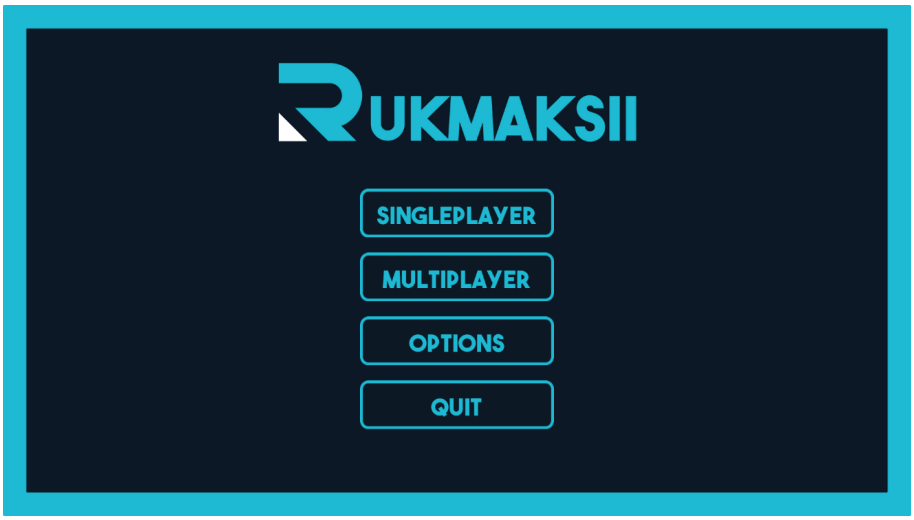

Figure 16: Picture of the first main menu

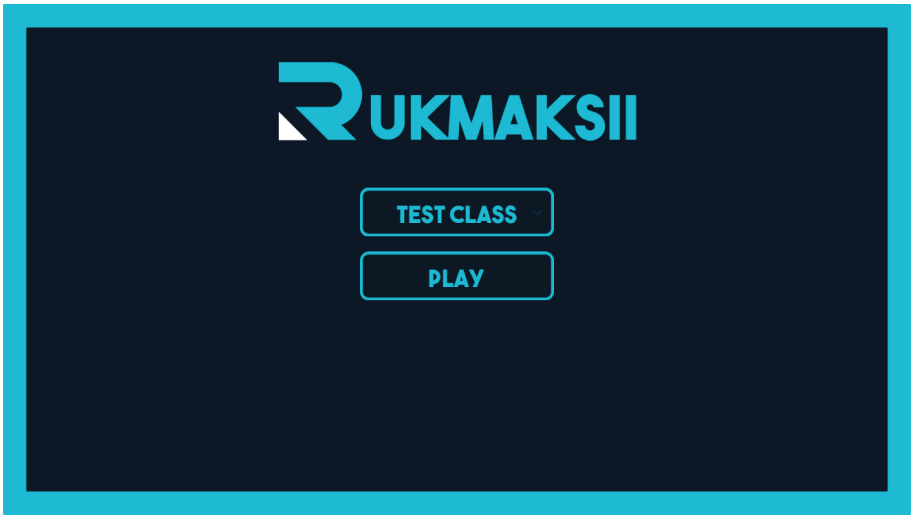

Figure 17: Picture of the first main secondary menu

3.2.20 Capture points

Capture points or objectives are a fundamental component of this game so we implemented a first version of objectives; they were represented by white circles on the ground. If a player stood on it, an indicator would appear on his screen and he would slowly capture the point. The larger the number of players on the point, the faster they could take over it. If the players left the point while capturing it, their progress was saved and they could resume by stepping back on it. Once the point was captured, it would change color and could not be captured again.

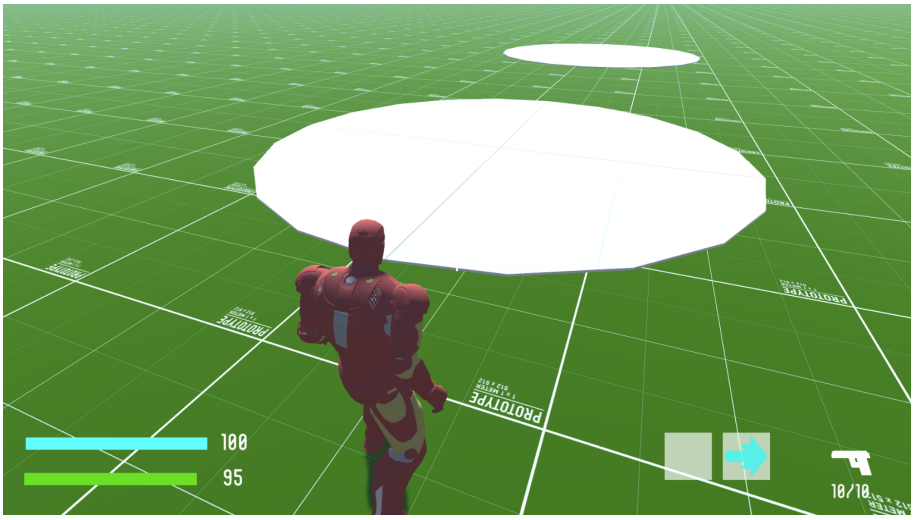

Figure 18: Picture of the first objectives

3.2.21 First Terrain Model

With all those previous enhancements to the game, we needed now more than ever a new map, with obstacles, variations of height, world border, etc. With all of that in mind, we used the terrain tool from unity to create a basic map with features that we needed. It helped us test our game and added more variation of gameplay, giving us some directions for the future of the game. For the obstacles, the terrain tool allows us to place down trees

really easily, with various settings with which we could place different types of trees so that the map wouldn't be too monotone.

Figure 19: Picture of our first terrain model

3.2.22 Teams

One of the main goal of our game is to make two teams of two or three players fight against each other in order to win the game, which is why it was necessary to add those teams. We could then make each team appear in its

corresponding base. We also made sure that we couldn't hit our teammates and that an objective point captured by on of the members of a team captured it for its entire team.

3.2.23 Minions

To add even more diversity and AI to our game, we wanted each player to be able to spawn a minion which will do some actions depending on the strategy the player who spawns them chooses. The player can spawn them through the map and will choose the strategy through a dynamic wheel. The minion will then spawn by the player's sides and will keep its strategy until its inevitable death. This feature makes the strategy decision very important since a player can only have up to 2 minions at a time. In order to be able to spawn a new minions, another one has to leave to the afterlife.

3.2.24 3D enhancement

After we had implemented 3d models, animations and some weapons, we had to make the player hold its weapon and adapt the animations to that new feature. We used an free addon from unity allowing us to place a weapon in the player's hand and moving its bones accordingly to make the whole body seem natural. All that was left was to go through each animation and make sure that nothing was broken, and when it was broken, we had to modify the animation to prevent for example the shoulder from doing weird back and forth movements.

3.2.25 Mini map

In order to help the player navigate the map easily, we implemented a mini-map which is displayed on the right-top corner of the screen. It was made up of an areal view of the map with indications on top. First of all, the player is represented by a white arrow. As the player moves around the map, the arrow showed his exact position on the mini-map and points in the direction he is facing. Nearby monsters and nearby ally minions also were displayed on the map, respectively represented by red dots and blue dots.

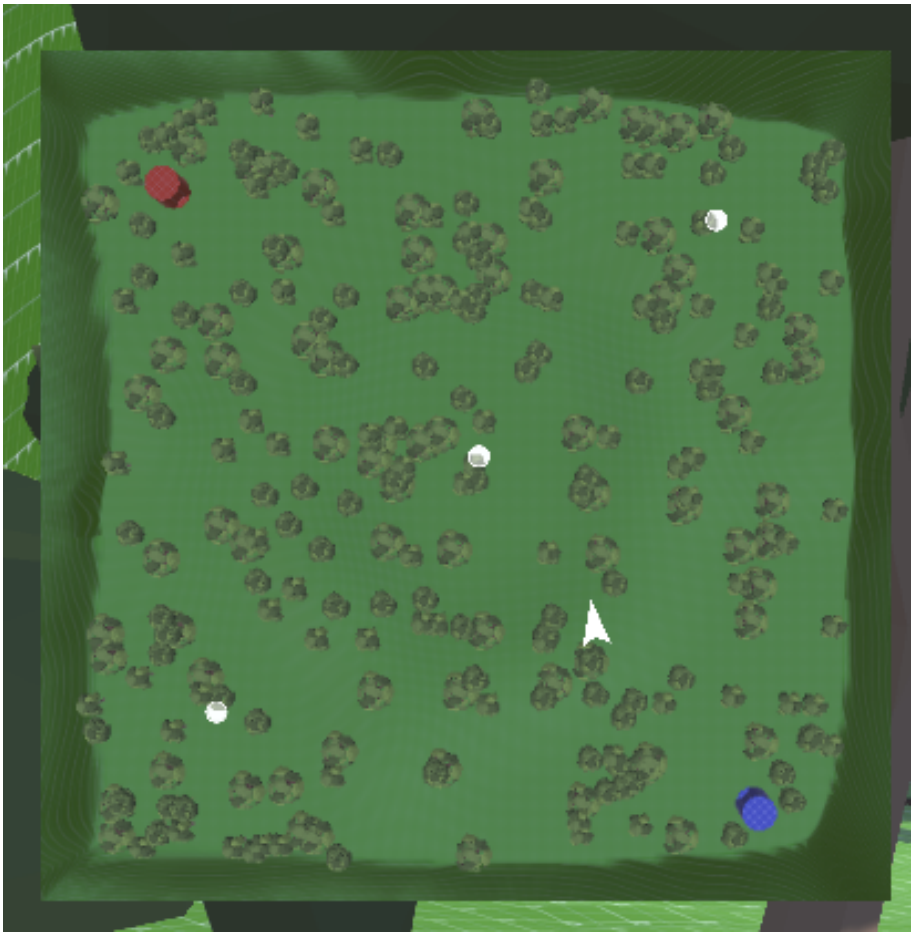

Figure 20: Picture of the mini-map

3.3 Game enhancement

3.3.1 Gameloop

For the gameloop we started by implementing the timer which will be the base of it, in order to manage the game from a beginning up to the end. The timer is always displayed on the screen of each players. Then we implement the random spawn of the monster on the map and the management of the number of monsters on the map at the same time. Finally, we add to the gameloop the management of the capture points and shields. The capture points are unlocked randomly with each cycle. A cycle lasts about 5 minutes.

During those 5 minutes, the point is unlocked and once captured, it is locked again until the end of the cycle, giving the capturing team a significant advantage over their enemies. During the game, the objectives must be captured by the teams in order to lower the enemy's shield, which allows them to damage the enemy's base. The gameloop also ensure all along the game that each player can pass through its shield.

3.3.2 Grenades

Since a base implementation of the items was already in place, we needed some proper items to finally be in the game. The first one we settled on was a simple grenade; not an original idea as it is a standard of most shooter games but an effective one.

When held in hands, a simple left click is sufficient to throw the grenade in the direction the player is looking. When it is thrown it behaves as normal projectile; it can bounce of structures and is subject to gravity. Three seconds after being primed, the grenade finally explodes and deals damage to monsters or players around it, in a defined blast radius.

This means that launching a grenade is not as straight forward as shooting an enemy and requires some small adjustments like aiming above the target. In addition, some particles have been added to the explosion as a visual indicator.

Figure 21: Picture of a grenade in hands

Figure 22: Picture of a grenade exploding

3.3.3 Shields

We then add the shields to the bases, at first they were invisible. It is a simple sphere that protect the base from the enemy, except when it's disable it block the access to the base to all players that are in the enemy's team, and so block the damage that the enemies could inflict to the base.

3.3.4 Item wheel

We now had two items, the fuel tank and the grenade, we had to implement an easy way to use them. We improved the HUD to make an item wheel appear at the center of the screen when a certain key is pressed. The player then moves its mouse in the direction of the item he wants to use and when he releases the item wheel key, the chosen item is selected and put in the hands of the player.

3.3.5 Money

The money in the game is a way to the players to customize its game. At the beginning each player starts with 500€. He could gain more by killing other player. When a player kills an other player he recovers half of the enemy's money. Or he could kill monsters, since when we kill a monster an item spawns and if the player has already the maximum number of this item, he recovers the price of the item. This money allows the player to buy new weapons, items or abilities.

3.3.6 Huge map enhancement

After having implemented a bunch of new features, we realized that if we let our map as it was, a round of our game would only last a couple of minutes, and the players were able to fly across the whole map really quickly and easily. We decided to make the map more than three times bigger, to give it unusual shapes to add more dynamic to the game.

Figure 23: Picture of the unusual shape in our map

However, that was not the only reason we had to change our map. In fact, even though the terrain tool from unity is very easy to use and very useful, it doesn't have any depth which mad some of our items and weapons go through the ground when we were dropping them.

To prevent this problem, our new map was designed using the free 3D modelling software blender to have the power of modifying the depth of our map. However, removing this problem had brought a new one: We had lost the tool to place down hundreds of trees quickly and easily. We looked for some blender addon which could do that the same way but didn't find any free one. We were then left with one only solution: We placed more than 600 trees by hand.

Along with this new map, we could not keep our old objective points and bases. We used Blender yet again to design new ones which were more detailed and pleasant for the eyes.

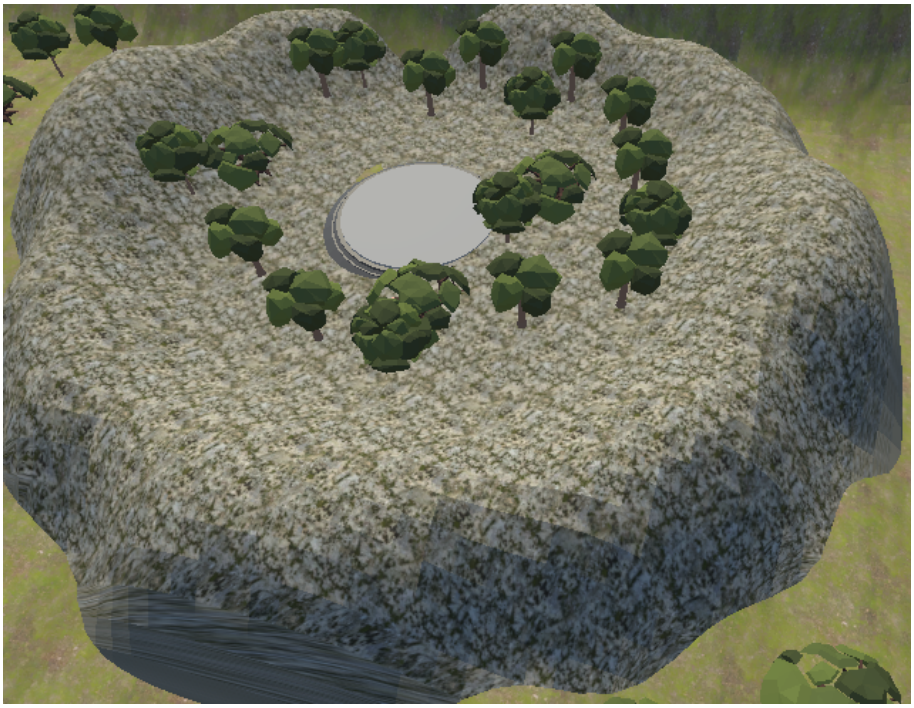

Figure 24: Picture of the new objective points

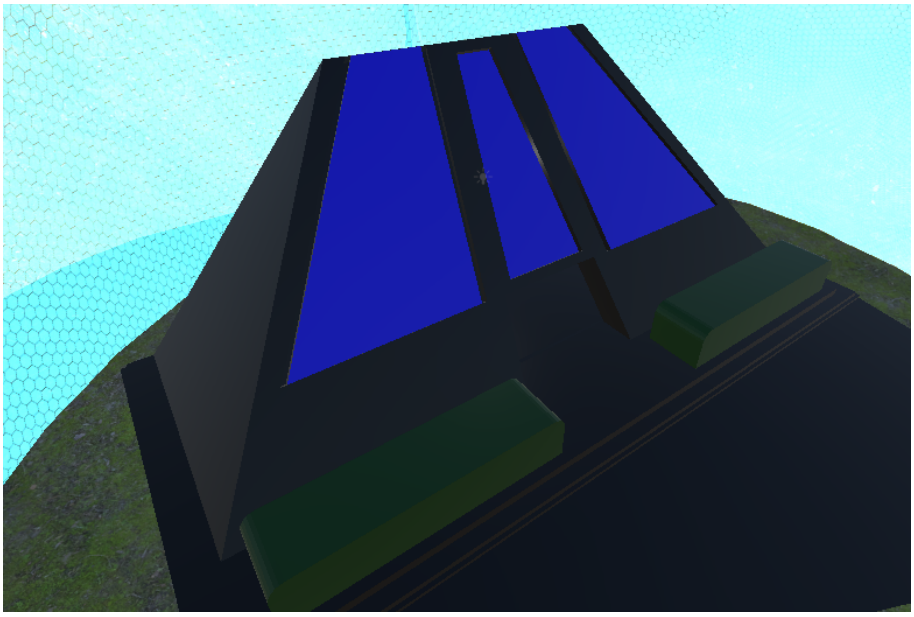

Figure 25: Picture of our new bases

To add more obstacles and diversity to the map, we also decided to add ruined house around the map.

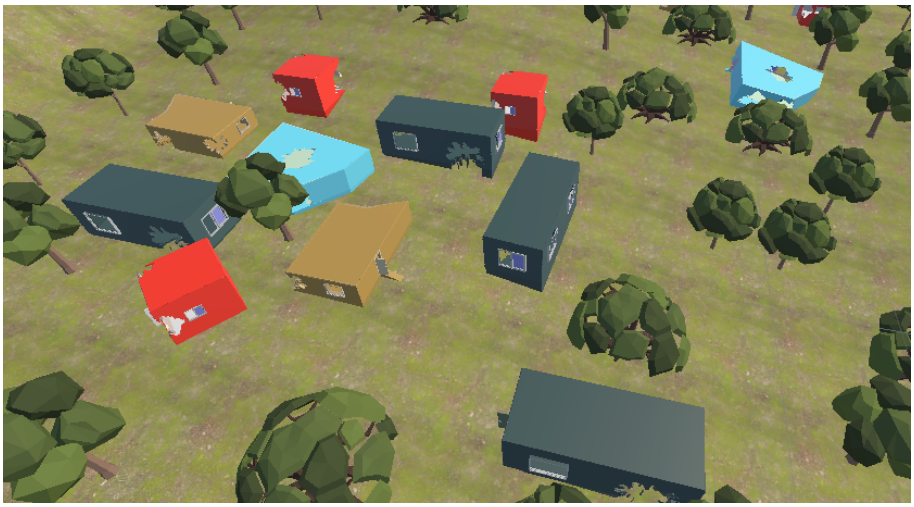

Figure 26: Picture of the houses we added

Finally, some fog was added to the map to prevent anyone to see a player across the whole map.

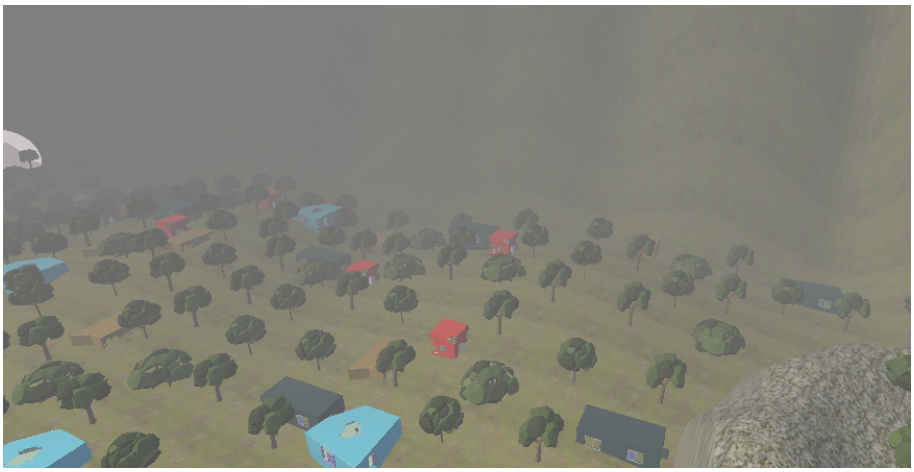

Figure 27: Picture of the fog

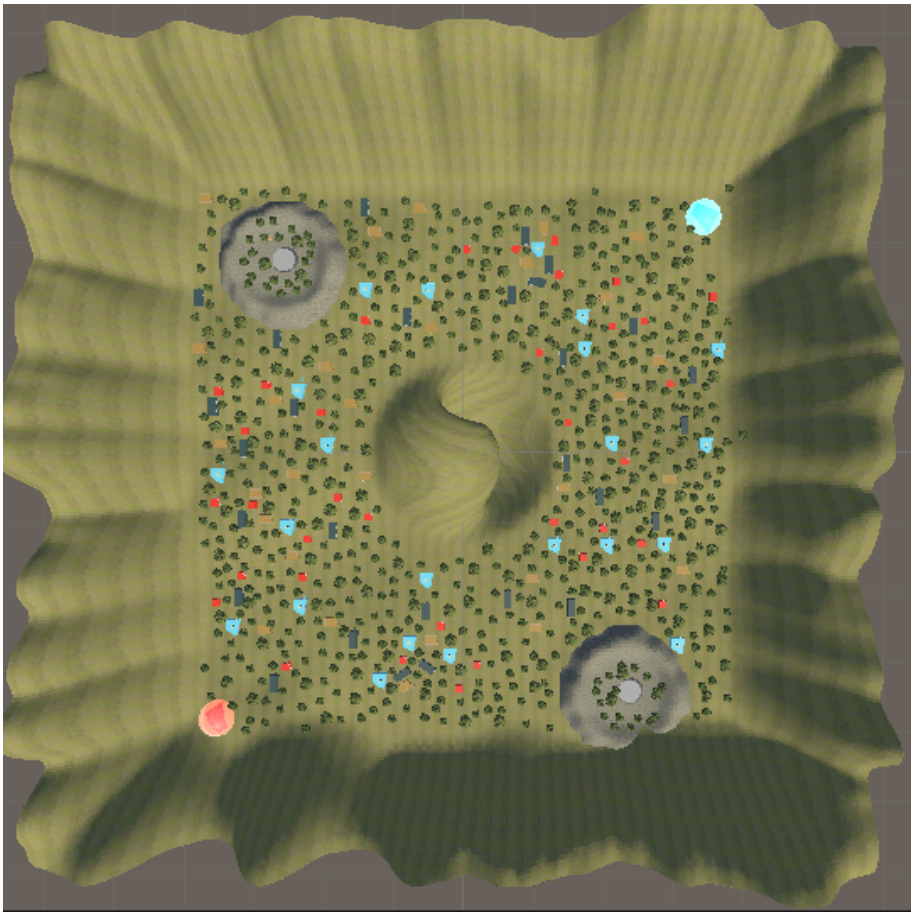

Figure 28: Picture of our new map after the said enhancements

3.3.7 Shops

The shops are the places where the players could buy weapons and items. They are not always available. They are available only when the player's team has captured the objectives. On the map there are three shops, one on each capture points and only the shop bound to the captured objective is available.

Figure 29: Picture of one of our shops

3.3.8 Scoreboard

In order to keep track of the advancement of the game and especially the statistics of the players we added a scoreboard. Though it was not accompanied by a visual interface in game, the scoreboard tracks live and stores in a dictionary data such as the amount of damage dealt, the number of kills, the number of deaths, etc. for each player.

3.3.9 End screen

In order to mark the end of a game and summarize it, a final scene was added. This so called End screen appears for every player when the game ends; when a base is destroyed. Signaling the end of the match, the winner is announced in the middle of the screen.

In addition, a proper scoreboard is displayed with every players, colored based on their team and ranked based on the number of kills they have. Alongside their nickname are displayed a few statistics so the players can see how they performed and compare themselves to the others.

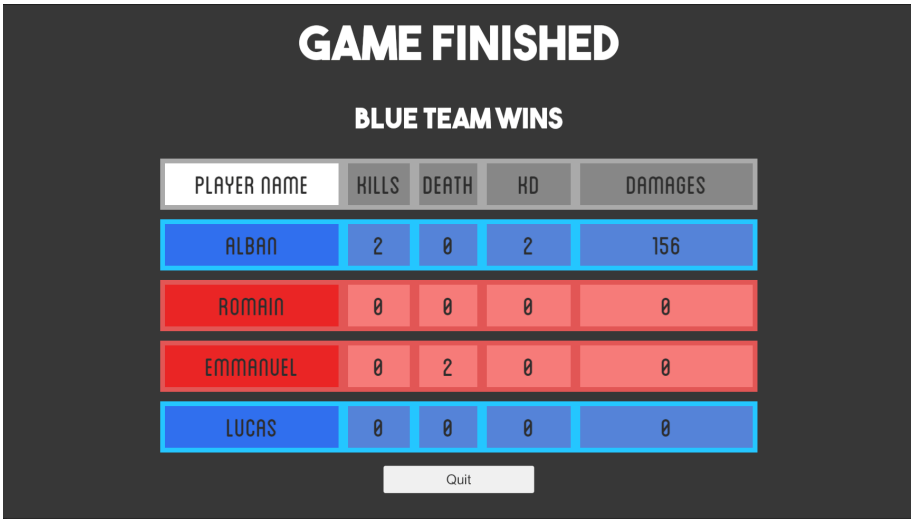

Figure 30: Picture of the end screen

3.3.10 Main menu overhaul & lobby scene

The initial connection scene needed some heavy work as we switched to Photon and finalized the project. Indeed the first menu only allowed for some primitive multiplayer. First of all, we added a mandatory nickname field on the very top, where players must enter their nickname which will follow them the whole match. Then, we removed the option button as we realized we could not find a use for it.

Finally, we changed the way players could connect together as we switched to using rooms: from this point forth, players could either create a new room or join an existing one, using a invite code. As they name indicates, the "Join room" button was created in order to join the room whose code was entered in the specified field and the create room button was added to allow the players to freely create rooms and invite their friends to join them.

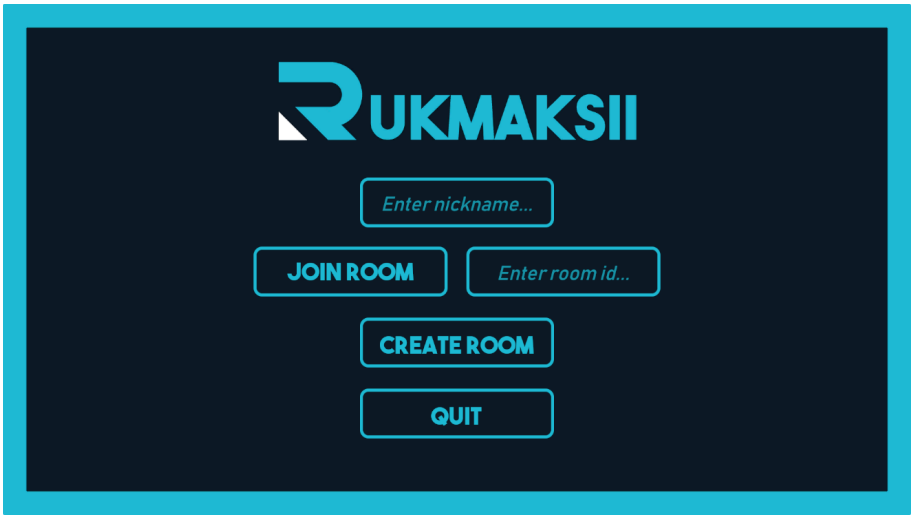

Figure 31: Picture of the new main menu

Alongside those changes, we added a lobby scene where players are sent when they create or join a room. In the middle of the scene is displayed the room code which can be copied by clicking on it and sent to other players so they can join this particular room. On the right of this field, player can click on the quit button in order to leave the room and go back to the main menu. A third button which allows to start the game appears only for the host and can be pressed only if there are four or six players in the lobby.

In addition, all the players in the room are displayed on the top or the bottom of the screen, depending on their team. Players cannot yet chose their team; they are automatically distributed between both teams as they join. The lobby scene also is the place where players chose which class they want to play as. On the left-hand side, all of the available classes are displayed and players can click to select one of them.

Figure 32: Picture of the lobby scene from the host's perspective

3.3.11 More weapons

Since the beginning of the project, we were working with only a small handful of weapons; a big change was needed. We added a large number of weapons for all classes and from all category. By playing a bit with the damage, the firing speed, the range, etc. we managed to create a large weapon pool where every weapon is different; has its strengths and its weaknesses.

For instance, the M4 is the weapon of choice for general combat with its medium range and fast attack speed but if a player feels more like an elite sniper, he could buy an AWM for a greater range, higher damage but a very slow firing range. Finally, if one dreams of flashy plays, an MP5 is perfect for quick kills with its very high damage per second but very small magazine.

Figure 33: Picture of different weapons

3.3.12 Mini map enhancement

With the evolution of the map and the implementation of a lot of new mechanics, the mini-map needed a refreshment. First of all, the terrain shown on the mini-map was updated to reflect the new map and its terrain changes (trees, houses, etc.). The main features of the old mini-map have stayed the same but some new ones were added.

To help the player, overlays were added on the positions of the objectives, displaying their status: free to be captured or locked. Whether an objective was captured is also shown on the map as the controlled objective takes the color of the capturing team. The shields also are represented on the map with the color of their team and their status: activated or deactivated.

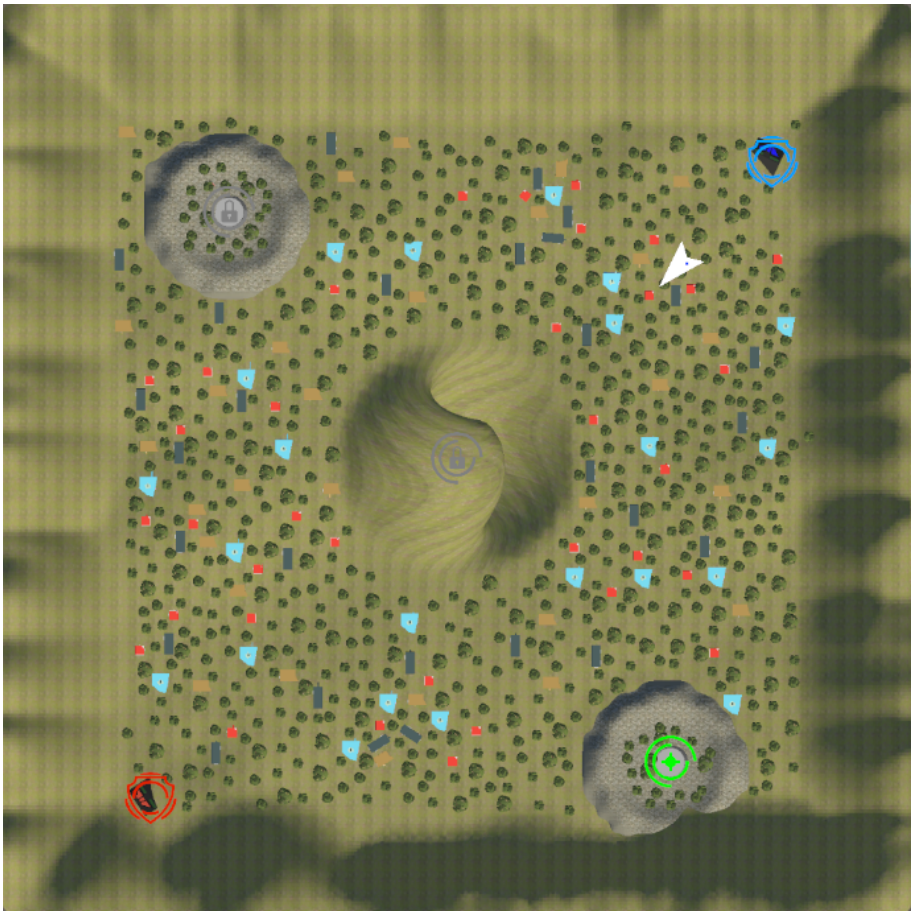

Figure 34: Picture of the new mini-map where an objective is free to be taken while the others are locked; the shields are all up.

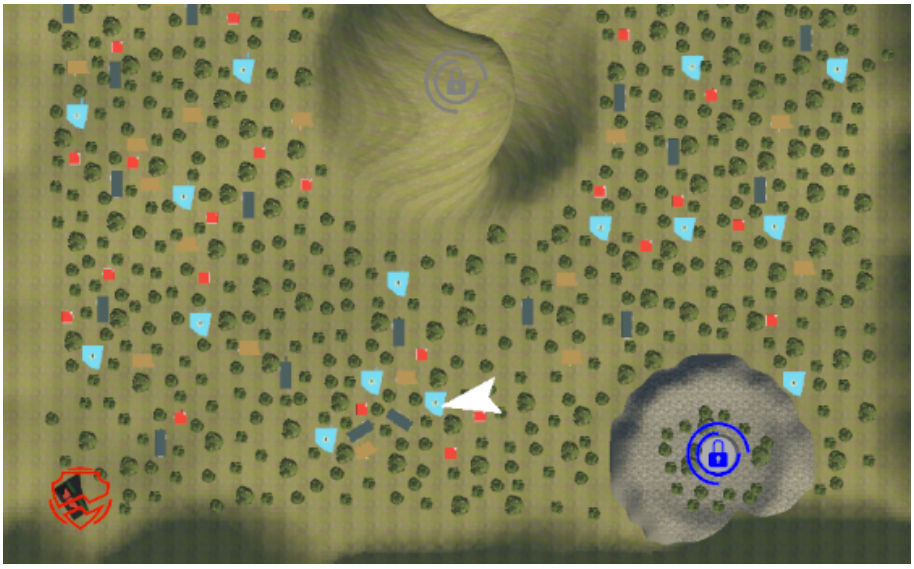

Figure 35: Several icons which can appear on the map

Finally, the map can now be opened by pressing a key, taking over the whole screen. When the map is opened, the player cannot move anymore but it allows him to better see the status of the bases and the objectives.

3.3.13 Minion's strategy selection enhancement

Since selecting a minion's strategy and spawning it was not very clear up to this point, we decided to update the way minions are spawned. Previously, the player had to select a strategy by pressing the X key (which would cycle between the available strategies, displayed on the screen) and then press another key to summon a minion. In addition we did not have a way to tell our minions were to go. To remedy to all of those problems, we have implemented a system close to the item wheel. When the player opens his map, he can now hold right click to open a strategy selector; when he releases, the minion spawns next to him and heads for the place clicked on the mini-map.

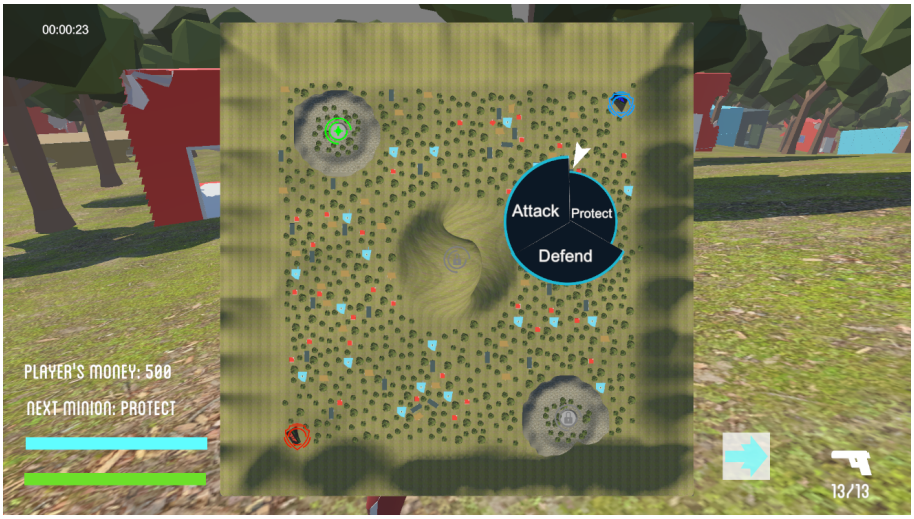

Figure 36: Picture of the strategy selector on an open map

3.3.14 Textures added to shields and monsters

In order to see whether a shield is activated or not we decide to add a texture to the shields. To do that we created two shaders which are the same except that the colors differs depending on the team.

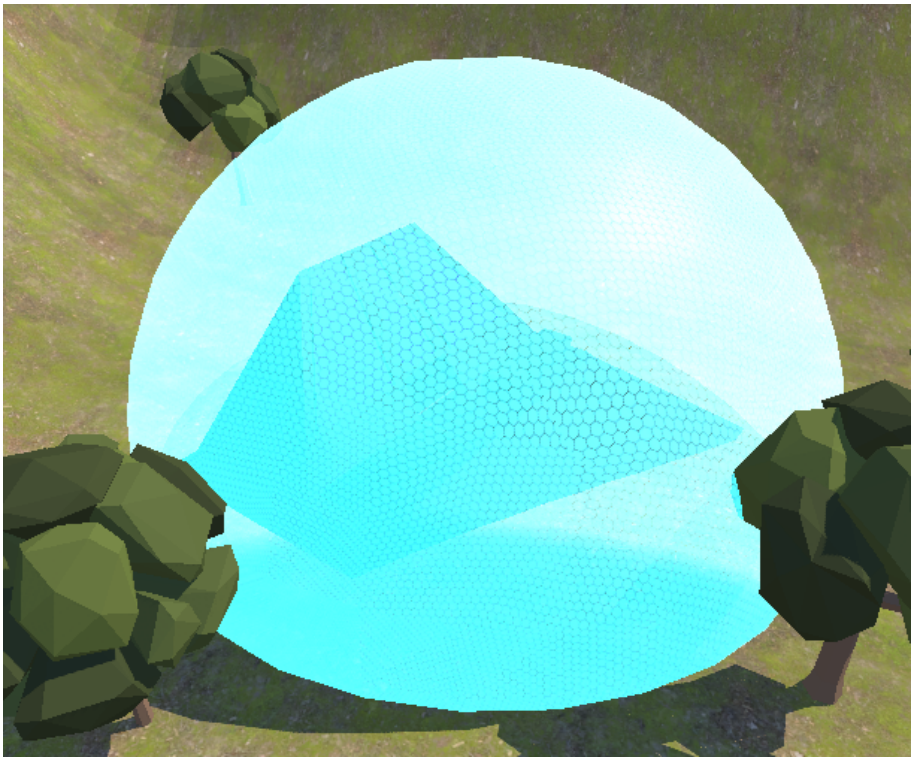

Figure 37: Picture of the shields' texture

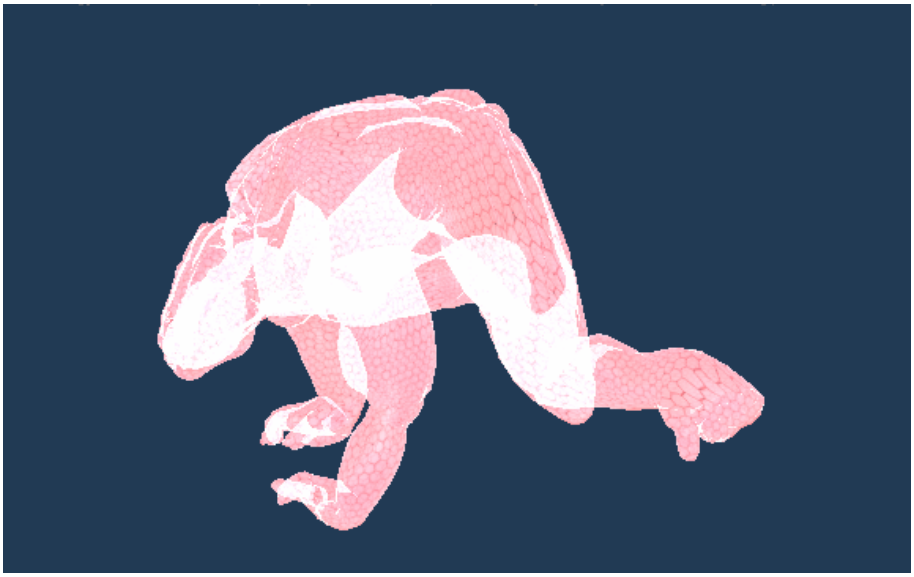

Figure 38: Picture of the monsters' textures

In order to create this shaders we had to move to unity Universal Render Pipeline (URP), this obliged us to modify many materials we used.

3.3.15 Pause menu

Before the implementation of a pause menu, the only way to quit a game was pressing Alt+F4 which was not very convenient so we implemented a pause menu which made exiting a match more user-friendly. By pressing the escape button, a menu now pops up, disabling the player's movement and unlocking the user's mouse cursor.

This menu has two buttons; the first one called "Resume" simply closes the menu and the second one is used to quit the match and redirects to the main menu.

Figure 39: Picture of the pause menu

3.3.16 In game name tags

For the players to know who they are facing, and in which team is their opponent, we decided to add in-game name tags above each player, changing its color depending on the team of that player.

Figure 40: Picture of a name tag

3.3.17 Announcements

In order to help the player to be aware of what is happening in the game we decided to had announcement. At each event, the gameloop displays on the players' screen what's going on, for instance if a new objective is activated or if a team has captured an objective etc.

3.3.18 Timer and Endgame mechanics

If a base is destroyed it means that the game is over, but after 20 minutes if no base has been destroyed the gameloop deactivates all the objectives and shields of the bases, the timer turns red and an announcement, also in red, tells the players that the end of the game is approaching and that all the shields have been deactivated. But this does not change the outcome of the game, it ends once a base has been destroyed.

3.3.19 Ability Tree

At this point of our game's development, we already had most of the features we wanted to implement in our game. However, one main feature hadn't been implemented; the ability tree. In fact, even though we had items, classes and weapons, we still wanted a way to change a player's stat permanently through an ability tree.

With this ability tree, the player can choose up to five different stat changes, each of them coming at a certain price depending on its power.

Once we had this functionality, all that was left was the UI to use it. We chose to make an area with the abilities we had bought and another one with the abilities we could now buy. the player has no idea of what abilities he will discover next, making him want to play again and again to discover all of them.

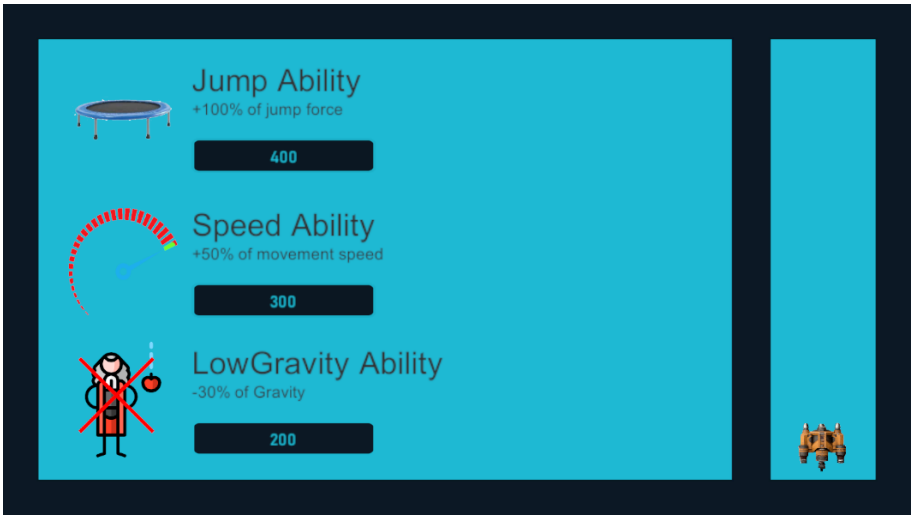

Figure 41: Picture of our ability tree

3.3.20 New abilities

With the ability tree and the ability class created, we now only had to find new ideas of abilities to add to the game. With this way of thinking, we created for example an ability to increase speed, another one to increase jet-pack fuel, etc.

3.3.21 Sound

To each weapon, we added a firing sound. It is played each time the player shoots. In addition, when the player reloads, a reloading sound is played. Unfortunately these are the only sounds we had time to add to the game.

3.3.22 Bug fixes

We spent our last week before the last defense fixing different bugs we found by playing our video game. There were a dozen of bugs but most of them just needed small and quick fixes. Though we cannot guarantee that our project is free from bugs, this last step in the making of this project ensures us a project that will work in many condition without discomfort for the users.

4 Website

As specified in the Computer project file, a website was needed. Though it has not been a pronounced focus, we have assembled on our website[2] the most important information. To begin with, the front page consists of some some images and descriptions of the game alongside a download button, redirecting towards our Github repository. In addition, we created an "About us" section, presenting all the members of our team and giving a small description.

5 Box

With all good games comes a great game box so we decided to design a proper one using a reconditioned DVD box. The focus was on giving it a look as realistic and professional as our game allows. In order to do that, we chose a somehow sleek design inspired by great commercially available games while still sprinkling in as much details as possible. But we did not want our box to only serve as decoration so the back was dedicated to summarizing our game in a few images and and captions. In addition, our names and a QR code to our website were added on the bottom.

[2]https://rukmaksii.github.io/website/

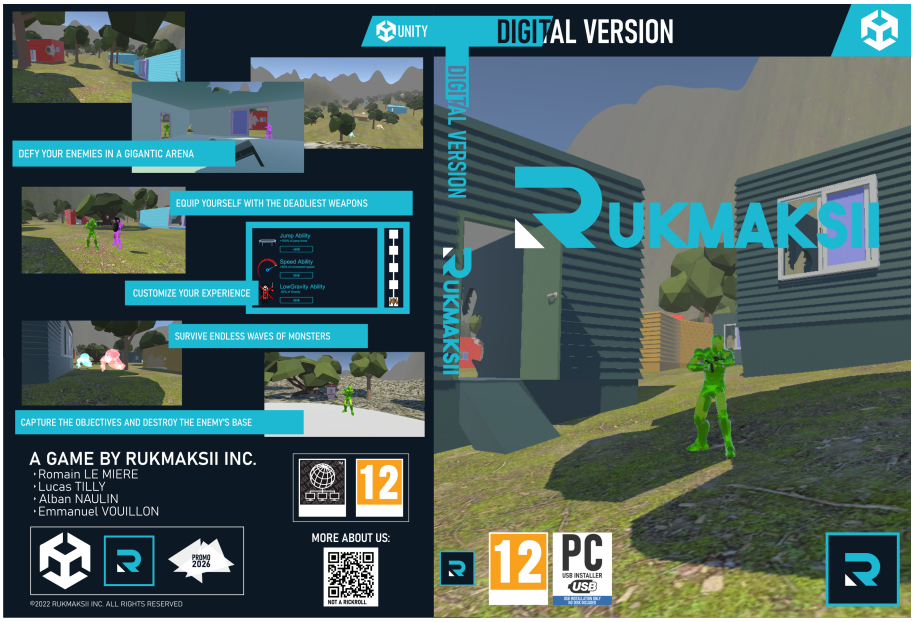

Figure 42: Design of our game's box

6 Conclusion

Making this game allowed each one of us to practice programming a lot and to learn lots of new things. Furthermore, since we all are working in group, it is a first sight at what projects could be in the future. We all learned a lot from this months of project on how to work together as a team and how to manage a team for the project manager. Finally, though it has not been without hard times, we have worked well during this project and mainly succeeded in our goals.