Final Report

Falling asleep studios

Luminosité Éternelle

Johan Emmanuelli
Ilan Mayeux
Luca Sarubbi
Matthieu Porte (*Project leader*)

Contents

1 Introduction

1.1 Team presentation

Our team is composed of four first-year students from the A2 class. We decided to join force together in this project to experience an adventure we will remember, forever. Therefore, this team is composed of:

Matthieu Porte (*Project leader*)
I discovered programming as a kid through front-end web taugth on khanacademy at the time, then I started learning C# as well to program games on Unity. I've done some game jams since, started some projects alone but never ended up with something I was completely satisfied. I hope for this project to be the one, but taking a look at my sidekicks I see no doubt about that!

Luca Sarubbi
Hi! I'm Luca Sarubbi. An aspirant low-level software engineer; although it all started from enjoying minecraft's game engine through custom Java plugins, I then got introduced to lower level languages such as C or Rust. I discovered epita thanks to a friend's recommandations and I am ready to give this projet my all. My contribution to the project will mostly be on gameplay and multiplayer in order to gain some experience to someday create my own game-engine.

Ilan Mayeux
I am wandering as Ilan Mayeux. A first-year Epita student from the A2 class. Previously from Le Bon Sauveur highschool, I learned precious knowledge during my scholarship and built a passion in computer science as a result. My goal is to renew this tradition and make this second-semester project a warm memory, something that we can be pround. Sincerely, your typical Linux enjoyer.

Johan Emmanuelli
Hello, my name is Johan. To this day, I really enjoy my education at Epita. I started programming in highschool and I immediately really liked it, to the point that I want to make it my profession. For this project, I am planning to learn and develop many skills. I am also really excited to work with this team, and I am sure that we will all be proud of this project !

1.2 Project presentation

Luminosité Éternelle is a co-op game in a 3D mountainous environment where you travel in hope to bring back your deceased companion. An epic adventure awaits the players. Monsters, puzzles, challenges, explorations. Discover the world, resolves enigmas and find a path towards the light, the key of all mysteries.

This is a game played by two people. Each player fulfills a specific roles:

- The Human: Travel through the world in your weak body. Enemies will try to bring you to the other side, your friend side, Death. You are able to interact with the living world and talk with living beings.

- The Ghost: You are freed of your weak body, but not safe either. While you are able to fly for a short duration, void enemies are a nightmare. Be careful, you can only interact with the dead world. You're also able to go through transparent objects.

This is the story of two friends that loves adventures, where one day, in a tragic event, one died. However, the deceased friend was saved by the temple of the light but not enough to go back to the living world. In order to fully save him, they have to reach the temple, which is in the mountain.

Our strongest ability can easily become our greatest weakness.

1.3 Graphic chart

1.3.1 Logo

Our game project is called *Luminosité Éternelle*.

Its logo represent mountains, the players final destination and its shining top, the light that will guide them. A journey awaits them, and many perils with it.

In a world mixed with darkness, one of the reason it has not fallen down is this temple at the top of everything. The temple of light.

1.3.2 Art direction

The game wants to convey an important message. Thus the art and the global ambiance of the game plays an important part of it. Most of the assets used are from the internet, yet some of them were made by us. For example, waterfalls, bridges, etc.

The art direction for the game is based on a mountain landscape, with a shining temple of light at the top. The mountain itself is rendered in low-poly style, with simple geometric shapes and bold, vibrant colors that give the environment a fantastical feel.

The temple of light is a towering, radiant structure that stands out against the dark and rugged mountainside. The game's creatures are also be fantastical and low-poly: dragons and slimes, with exaggerated features and striking visual designs that make them stand out from the game's environment.

Overall, the art direction for the game aimed to create a surreal and imaginative world that is both visually striking and true to the game's low-poly aesthetic.

subsectionMain source of inspiration Our goal is to bring an original aspect to the game by mixing mechanics of different games to create an unique gameplay. The first major puzzle game that inspired us in the concept would be Myst, released in 1993.

1.4 Origin

We first thought about doing an horror game, with a low luminosity set up because it would be easier for us to make assets and create a great horror immersion. And the genre is already pretty established so we would not struggle figuring that out.

Then we thought more about what we really wanted and realised that we wanted to share more than a few scars with this project. So we opted for a more story-oriented kind of game. Thus, we talked during many hours, made votes, merged ideas until reaching a common goal. And so, this project idea is the result of all our expectations, our passions, our stories, our souls.

We also want to let our game be accessible cross-platforms.

Indeed, while the market is mostly centered around Windows, other platforms such as Linux are often abandoned. To survive, most of Linux players are surviving using Proton from Valve and Wine. Fully compatible Linux games are rare. And since some of us are Linux enjoyer, their pride is not to deceive other Linux users.

And we did it! Our game is fully compatible with Linux, Windows and Mac OS. We are proud of it and we hope that it will be a good example for other game developers.

1.5 Sources of inspiration

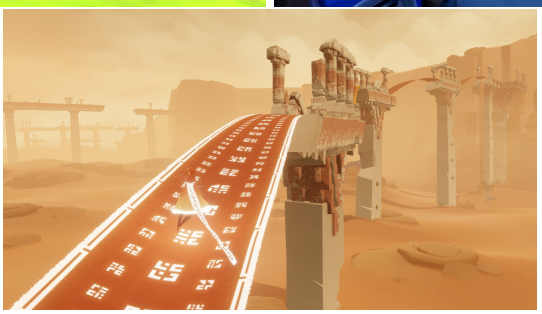
Concerning the adventure, the famous *The Legend of Zelda*, released in 1986, is obviously a large inspiration.

Indeed, the saga is well-known for its mix of puzzles in various environments, its battles against different ennemies and bosses, each having unique weaknesses, is also offering the player limitless combats possibilities. Its dungeons which are a mix of battles and puzzles, with linked rooms leading to a boss.

And of course, its map. A vast one, rich in mysteries and adventures. Villages, mountains, deserts, lakes, forests, castles, volcanoes and other fantastic places, ideal to unleash the full immersion in an epic adventure. Many of these are a great inspiration for many new games, and us!

We can also quote *Journey*, a beautiful and meaningful story where the player tries to understand his goal and travel to reach his destination. This game is mainly playing on the strengths of its map and musics to convey its own message.

More recently we can mention “We were here” for the co-op mechanics.
It is our main source of inspiration for the puzzles and gameplay. It really relies on cooperation since you can do stuff that your partner can’t and have access to information that your partner doesn’t.



2 Evolution of the specifications

2.1 Task Sharing

	Matthieu	Ilan	Luca	Johan
Multiplayer		+	+ +	
Gameplay	+		+ +	
A.I.		+ +		+
Story / A.D.	+ +	+		
3D (characters)	+ +			
3D (environment)			+	+ +
Level design	+	+ +		
Puzzle design	+			+ +
HUD and UI		+ +	+	
Website	+ +		+	
Animations		+		+ +
Music composition	+			+ +
Cinematics	+	+ +		

+ + : **Task leader**
+ : **Task second**

Figure 1: Initial task sharing

	Matthieu	Ilan	Luca	Johan
Multiplayer		+	+ +	
Gameplay		+ +	+	
A.I.		+ +	+	
Story / A.D.	+			+ +
3D (characters)	+ +			
3D (environment)	+ +			+
Level design	+	+ +		
Puzzle design	+			+ +
HUD and UI		+ +		+
Website	+ +		+	
Animations		++		+
Music composition	X	X	X	X
Cinematics	+	+ +		

+ + : **Task leader**
+ : **Task second**

Figure 2: Final task sharing

2.2 Progress Predictions

	1st	2nd	3rd
Multiplayer	60%	90%	100%
Gameplay	60%	80%	100%
A.I.	40%	75%	100%
3D (characters)	50%	80%	100%
3D (environment)	50%	80%	100%
Level design	50%	80%	100%
Puzzle design	45%	65%	100%
HUD and UI	50%	100%	100%
Website	30%	70%	100%
Animations	30%	60%	100%
Music composition	50%	75%	100%
Cinematics	0%	50%	100%

Figure 3: Initial progress prediction

	1st	2nd	3rd
Multiplayer	60%	90%	100%
Gameplay	60%	80%	100%
A.I.	40%	75%	100%
3D (characters)	50%	80%	100%
3D (environment)	50%	80%	100%
Level design	50%	80%	100%
Puzzle design	45%	65%	100%
HUD and UI	50%	100	100%
Website	30%	70%	100%
Animations	30%	60%	100%
Music composition	50%	75%	100%
Cinematics	0%	50%	100%

Figure 4: Final progress

3 Project Advancement

3.1 Gameplay

3.1.1 Players

The Basis - Ilan Creating the player movement and allowing to look around is the first step. Some basic PlayerMovement and MouseLooking class were created to handle these tasks. Thus, our player could now move with W A S D.

Then, the main Player class where Human and Ghost will inherit from it. This class handles animations, inputs and most of the references. The ghost player has a special jump: he can fly for a short time. This can be triggered with [ ].

The Items - Ilan

Then, when the basis of the player were done, some equipments were needed to adventure in this dangerous valley. And therefore, an Item class was created. But to use these items, the player needed an item bar that we called an item wheel. Weapons, Potions and others will inherit from the Item class. Since the items needs to be picked up on the ground, the user needs an indicator, a message, "Press E to pick up the Demon King Sword" for example, that the item will display when dropped. Then, after the basis of an Item come the item themselves. In order to defend himself, the player needed some weapons. A sword to handle enemies at close range, a staff to deal huge AOE damage to enemies, and the most powerful weapon: a spear that can be thrown at enemies or used to quickly attack at a high range. Here are the available items:

- **The Sword** - Attack enemies at close range, swing your sword and kill your foes. This weapon is the first obtained in the game and the most easy to use.
- **The Fire Staff** - Eliminate your enemies by summoning and throwing fireballs against enemies to deal huge AOE damage. This weapon is obtained at the village. Very powerful

against slimes but can become tricky to aim dragons at long range as the travel time of the fireball is not the fastest.

- **The Spear** - The spear is the third weapon of the game a most probably the most powerful one. This weapon has a great range of melee attack, a fast attack speed and, the ability to be thrown at enemies and recalled. This weapon will be the best with the fireball to kill dragons.
- **Potions** - Healing wounds is essential to survive, and therefore, potions can now be picked up and used. Different kinds of potions, with more or less a good amount of healing.
- **Chests** - Chests can now be opened, allowing one to obtain currency and potions or even better rewards. Chests can be opened by both players.

Since the player can manipulate different weapons and items, the item wheel allows the user to change items and unequip them. Yet, the player had no health nor mana. So a HealthManager and a ManaManager were added to fulfill these needs.

Water System - Ilan

Our game has a lot of water in it: Rivers, Lakes, Waterfalls, etc. Yet, until now, nothing was done on it. The player could basically jump and walk in the lakes, which is definitely not a good thing and greatly reduce immersion in the game. Therefore, we improved all that by adding buoyancy when jumping into water, and adding animations depending on the status: if you are in the water, you swim to stay at the surface, then if you start moving, you swim forward. The same way: animations of falling were added, so that the player cannot "run" while falling.

3.1.2 The Dialogue - Ilan

All the player's interactions are handled with the interact key, which is E by default. Interacting with the NPCs is an essential things to create a story and help the player. To add some further interactions, the player can talk back to trigger some special dialogues depending on the choice made.

When the player is nearby an NPC, it can interact with it. When the player interacts with the NPC, it starts the dialogue

A dialogue is a succession of text said by a character. Talking again with the same NPC can trigger a new dialogue. Many characters on the map have their own dialogue.

After coming back on the project, I realized that we did many implementations mistakes. Indeed, Unity provides a class called Scriptable Objects. Subclasses that inherit it does not need a game object to be active. It is called an asset file. And these assets files work better, are easier to update and more intuitive. They're also powerful enough in the memory as they're readonly during the game and inside only once.

Our way of doing it, without use Scriptable Objects, was to make our own Parser. The first thing was to decide how to write the dialogue files efficiently, without too much symbols.

We decided to use Flags.

- The Dialogue FLAGS
  - [: Start of a dialogue
  -] : End of a dialogue
  - **!** : If the next dialogue is automatically triggered at the end of the current dialogue
  - < : The next characters contains the name of the game object of the Next NPC. It works in pair with "!".
  - # : The next characters contains the name of this NPC
- The Choice FLAGS
  - **{** : Start of a dialogue
  - **}** : End of a dialogue
  - **-** : The text of an input choice, followed by a dialogue
  - **1-2-3** : Either 1, 2 or 3, these numbers allow the Parser to quickly tell how many inputs are in this choice.

With that done, we could start writing our owns files. To write a dialogue file, we could hence do:

```
[(!)
#Name
(NextNpc)
Line1
Line2
...
...
LineN
]
```

Figure 5: the dialogue writing format

The parenthesis are here to inform that these parameters are facultative. We then obtained a single dialogue where the NPC goes by the name of "Name"

For the choice one, we could write it this way:

```
{(!)3
#Name
(NextNPC)
What kind of weapon do you like?
-The Sword!
DIALOGUE
-The Fire Staff!!
DIALOGUE
-THE SPEAR!!!!!
DIALOGUE
}
```

Figure 6: The choice writing format

Once again, the parenthesis inform that these parameters are facultative. DIALOGUE is a shortcut of the first example. A DIALOGUE is triggered when the user select its input. There are as many inputs as the number tells (here 3)

If we replace the DIALOGUE with a true dialogue format, we could obtain something like this:

```
{ 1 #Name
What Choice
-Input1
[
#Name
Line1
Line2
...
...
Line5
]
}
```

Figure 7: The choice writing format with the dialogue

Hence, to obtain a file with multiple dialogues and choices, you just have to stack these formats one after another. We made it so that we can't obtain complicated dialogue and that it is linear. Hence, after a choice, you'll go back to the main route. A choice can affect the answer of the NPC, but it does not act as a Dialogue Tree.

3.1.3 Quest System - Ilan

The map is huge (1.5 per 1.5km), and new players may be easily lost on what to do, where to go, etc. We do not wish to implement a minimap as the map is still relatively small and straight forward. Hence, the quest system was implemented. There are for now three types of quests:

- Quest Area: This quest requires the player to reach the designated area. To be completed, it requires 1 or 2 players on the location where a light beam can be seen anywhere on the map.
- Quest Item: This quest requires the player to pick up the designated item. When the quest is active, the item glows up.
- Quest Monster: This quest requires the player to slay all the monsters. When this quest becomes active, it spawns the monsters in the area.

Each quest then point towards the next quest, allowing to easily build the main story.

3.1.4 Tutorial System - Ilan

Games often offer tutorials, and this game is not excluded. The quest system structures the story of the game, but new players still ignore the principle of it. That is where the tutorial system plays. There are for now two different ways for it to be activated:

- By a quest: When a quest becomes active, it may or not has a tutorial attached to it. If it does, then the tutorial overlay is also activated, helping the player.

- By the Weapon Help key: When holding an item with the weapon wheel, press F1 will display information to the player about how to use it and everything else.

Now the player can slowly learn the game by the mix of the quests and tutorials. For example, when the game start, the first game is about the movements: Moving with WASD, etc, then how to pick up weapons, then how to use them and slay weak enemies, etc. Until the player reach its final destination.

3.1.5 Overall architecture

For this project, we had to use a lot of the OOP mindset. Yet, there were still a lot we didn't do correctly, and we paid the price.

We did our best to use inheritance at its fullest and to avoid repetition of code. For example, a sword inherited from Weapons which inherited from Items, an abstract class.

We did also use events to remove dependencies inside the code. This was the first time we used Events, and this is indeed really powerful. However, we didn't do it perfectly how it was supposed to. We had many scripts that had references to others just to subscribe to the event.

The **MOST IMPORTANT LESSON** we have to remember from this project is **be careful of dependencies**. While at the beginning of the project, we had no issue writing the code, we also didn't pay attention enough to how we did thing. And it is also because we were beginners and had no prior knowledge of big projects. Yet, at some point, dependencies started to destroy us from the inside.

A script was relying on three, four, TEN other scripts. And again, these scripts were relying on many others. While using Events avoided issues and simplified code, it was not enough to spare us from dependencies. And where the dependencies struck the best, was the testing part.

It was nearly IMPOSSIBLE to test a feature without having to add half of the project in it. For the slightest thing, we had to play the game from the Start Menu. And that was a lot of time lost and debugging process.

The second major issue was the debugging and fixing. Since a script was relying directly on many others, when we had a bug, fixing it was challenging. Not only because testing if the fixed work was already hard enough, but also because modifying it while others were directly depending on it, could cause bugs to the others and so on. When one script broke, it could cause a hazardous chain reaction.

These are things we started regretting at around the end of the project. We had to give up developing thing and try to improve some code and remove dependencies. But fixing everything would have taken so much time as we had already hundreds of files with hundreds of lines inside of it.

One solution would have been to create a singleton EventManager. A class that would be always here and be referenced by anyone without issues. Hence, any event would just have to add listen to an event or trigger one. These would cut already many connection. Some people

also used Scriptable Objects, assets that does not even require to be on a scene, allowing for scene sharing information.

A second solution would have been to create components that work by themself.
A feature-component.
Completely independent.
That does exactly what it needs, almost by itself.
Hence, removing this component would be equal to removing the feature. And that component could be tested alone without prior requirements.

To conclude, there are many things we should have done to save the project, but we were not aware of it before it became to late. We won't make the same mistakes the next time!!

3.2 AI

This part was entirely managed by Ilan

Searching for the best Algorithm!

Our project is an adventure full of puzzles. Yet, there is no adventure without citizens, soldiers or monsters! Therefore, we had to develop an AI to play against or with. An AI capable of piloting the most dangerous foes while being easily maintainable and upgradable.

There exists many different algorithms to develop an AI pattern, without necessarily going towards machine learning. The most basic one would be an "If else" AI. The issue is that it would turn towards spaghetti code and upgrading it would get harder and harder. The second option we came accross was the Finite State Machine (FSM). It is well-known and very efficient. However, as the previous one, it can also quickly become a mess when trying to modify it. The third one was the Goal Oriented Action Planning (GOAP), an upgrade of the FSM into a dynamic one. It is the one we used.

After going back on the project, maybe a behaviour tree would have been more useful for us. GOAP System was indeed powerful, but it may be more suited for a succession of Actions that are unlocked bit by bit, and not a static monster that has a pattern.

The Goal Oriented Action Planning Algorithm (GOAP Algorithm)

As said previously, we ended up choosing the GOAP system above the others. But what is it? Why did we choose it?

The GOAP is working with 5 main classes, which are the GAgent, GWorld, WorldStates, GPlanner and the GAction. These classes are working together to make an A.I.. This system was modified to fulfill our needs.

- WorldStates: It is a simple class that saves every states it is given, with helper methods.
- GWorld: A singleton with its own WorldStates and other data. This is the Global World States, where AI can communicate with each others and share information (e.g. the alive

slimes)

- GAction: An action realizable by an A.I.. It can be a really simple action, such as walking or more complex, like abilities. Each action has four necessaries methods that need to be implemented: PrePerform, PostPerform, IsComplete and Action and attributes such as the cost, the duration, etc.
  - PrePerform: This method is executed before everything else. It is useful to check and setup additional things. If it returns false, the whole Action is canceled.
  - Action: The main action. It is executed after the PrePerform and contains the goal of this Action. This method is not in the original system.
  - IsComplete: This method returns true if the action is complete. It is useful to make an action last until a certain condition. This method is also no in the original system.
  - PostPerform: This method is executed at the end of an action. It can be delayed with the duration attribute.

  The main importance of the GOAP system is residing in the Preconditions and Effects of an Action. Each action has Preconditions that need to be met in order to be executed. And each action passively brings some Postconditions, also called Effects that can be used to attain other actions.
- GPlanner: The goal of the GPlanner is to build a plan for the GAgent from a certain goal. A goal will be the name of an Effect. Then, the GPlanner will try to find a succession of Action to achieve this Effect. It will try to find the best succession of Actions to achieve the Effect. And to obtain this succession of Actions, it will use a Pathfinding algorithm which, in our case, is an A* Algorithm. Then, if it finds any plan, it will select the best: the cheapest.
- GAgent: The GAgent is the parent of the A.I.. Each A.I. will inherit from this class. The GAgent has its own WorldStates, which will be called LocalStates or ActionStates. These are states that are active and can be updated by the Actions. At the beginning of its lifetime or during processing, the GAgent will receive some Goals that repeats themselves or not, with a certain importance. The A.I. will take the most important goal first and complete it. With the current goal, the GAgent will create its Plan, helped by the GPlanner, and start to achieve one by one the Actions necessary to achieve the goal. If an action fails (The PrePerform returned false), it will start making a new Plan to achieve the goal.

This is the GOAP system that we used. Completely unique since our adaption.
To compare a GOAP and a FSM (Finite State Machine), using graph is the best way. When making the A.I. in a FSM, actions are represented by a Graph and their relations with branches. And as any graph, when relations are getting complex, the graph will look messy. If we take back the same graph in a GOAP system, the nodes are exactly the same. The only difference is that there is no branches, no link, no connection. Each node is alone, yet connected. More precisely, only the Preconditions and Effects matter.

This is where the choice of the GOAP took place. It can be easily maintained and adding an action is just adding its Preconditions and Effects and then make sure that it can be accessed. An action can also safely be removed if there is other way to achieve the Effects.

The First A.I.: Slime

After selecting and implementing the Algorithm, creating the first foes would become the next step. And like every fantasy Rpg games, the slime was chosen. The cute little slime will fast enough become a threat for the player.

The slime is not particularly intelligent, but it is relentless in its pursuit of prey. It will follow the player character around, trying to attack with its slimy body. If the player character is hit by the slime, they will become slowed down. The slime can be defeated by using weapons or spells to attack it, but it is resilient and require multiple hits to defeat.

In addition to its normal form, the slime can also split into smaller slimes when it takes damage. These smaller slimes are weaker and easier to defeat, but they can still be a nuisance if not dealt with quickly. The slime can also regenerate its health over time, so the player will need to keep attacking it to keep it at bay.

When a slime is deeply wounded, they'll either try to fight to death or to flee. A slime prefers staying alive than eliminating their target so it'll flee most of the time.

When a slime dies, it will split in three smaller slime. Their stats will be equivalent to a third of its parent start. It can only occur once. If after 30 seconds, the slime survives, it will grow into a bigger one and increasing its max health, allowing it to regenerate additional health.

The slime special ability is a powerful jump towards the threat to close down the distance. The jump does a certain amount of damage to the player and also slow them. Luckily, a zone inform the player to its destination, which is, often, them. That's why a group of slime quickly become a threat.

This slime was thus implemented using the previous system. In order to differentiate different foes, a Monster class was created. Slime inherits from it. Allowing to handle its animations, stats and helper methods.

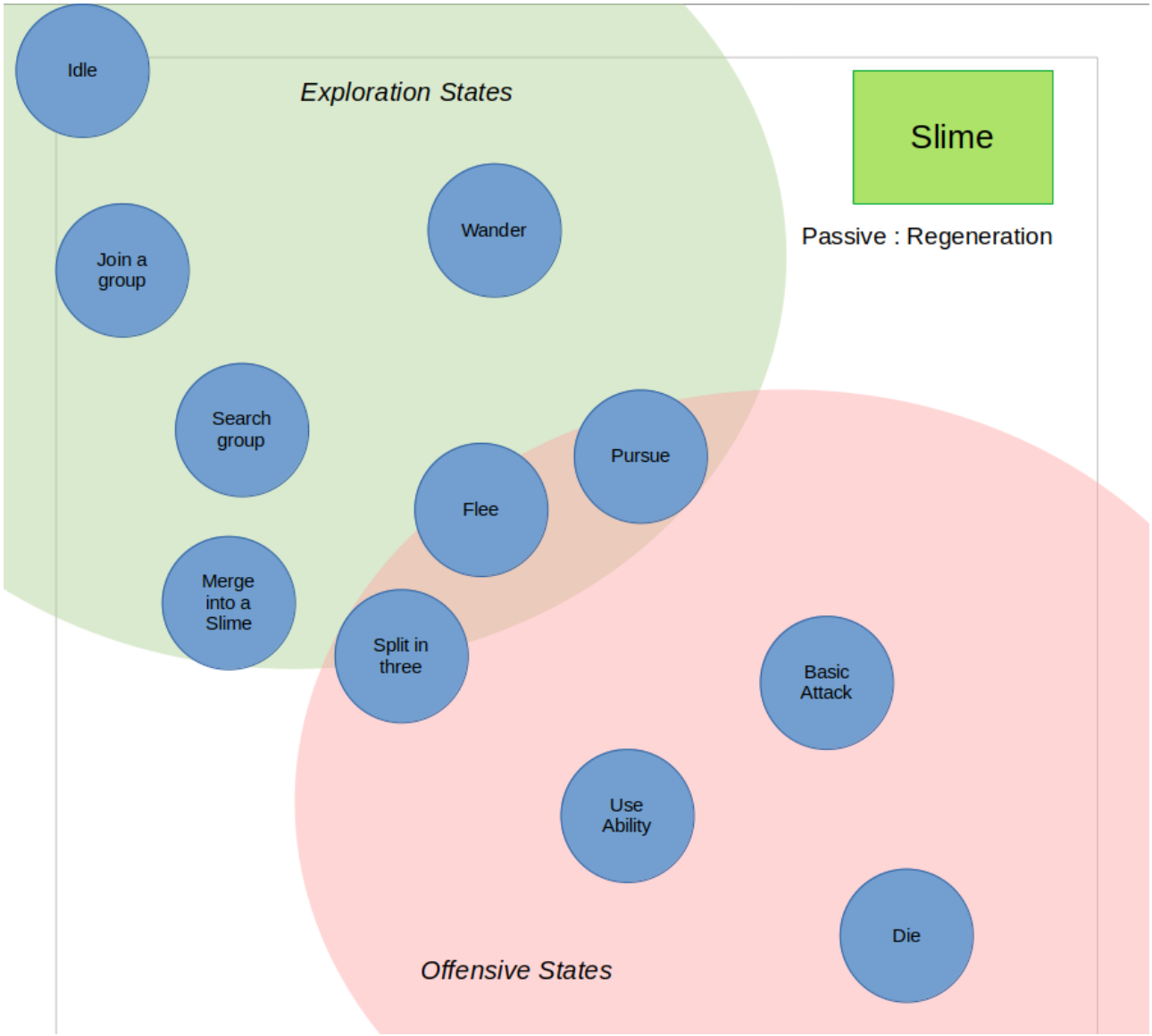


Figure 8: The slime actions as a graph

The second Monster: Dragon

The second monster is the dangerous Dragon. Indeed, there is no mountain nor adventure without dragons. Breathing fire is their specialty after all.

This monster is specialized in attack, making it a dangerous foe. Even more when its a flying creature, meaning that melee attack are not very efficient against them.

The dragon is able to fly, rotate around the player, and breath fire at the player.

The difficulty of making the Dragon was the 3D movement. A dragon can move also on the y-axis. Meaning that it has to choose when gaining altitude and when losing it. If it is not done correctly, the A.I. will look stupid or easy to beat. Ruining the dragon reputation is not part of the game so they need to get the best A.I. possible.

The current solution to this problem is to implement a Volumetric Pathfinding with Astar. The NavMesh system is optimised for AI to follow the terrain, but unity provides not alternative to implement a Pathfinding in volume.

So, how did I implement the volumetric pathfinding? The first step was to split the space. Indeed, unlike the traditional pathfinding, we want to save the spaces where there are no place for the A.I. to move on. Because for a volumetric pathfinding, we want to allow the A.I. to freely move anywhere, except where there are obstacles. To do, so, we used the tree data structure, and more precisely, an Octree.

What is an octree? An octree is a tree that has at most 8 children.

So, the algorithm is working as follow: We add all the game-objects with colliders we want to take in account. Then, the root of the octree will make a cube that fits all these game-objects, as big as we need it to be. And now we have the space to divide, we divide the space into 8 subspaces as long as there is an object that is smaller than the cube. When this is done, any cube that is smaller than the object is deleted.

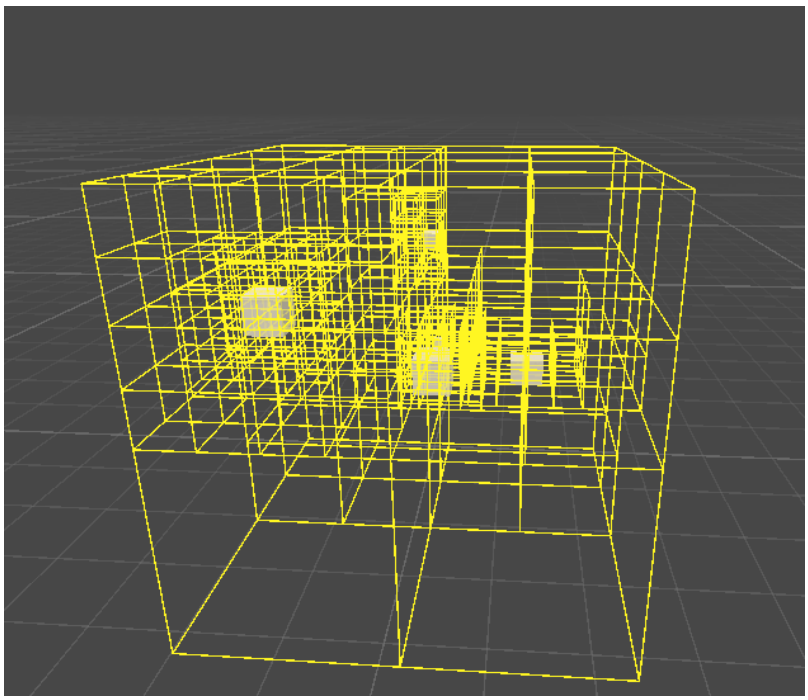


Figure 9: The space divided as an Octree

Now that we obtained our octree and divided the space, we need to build a graph from it. When the graph will be done, making a pathfinding from it is a piece of cake. Yet, building the graph was the hardest path and the slowest as well. The graph needs to be precise enough to have a short path towards the destination but not too precise or with many links otherwise the memory cost will be enormous. I tried many different way, but the one with the less links and still a good amount was linking all the adjacent cube without a game-object in it. By doing so, the links are still good enough to allow one to travel through the octree, and without too

much links.

Now that we obtained our graph, we just need to implement the pathfinding from it. As usual, the A* Algorithm was used. This is not the first time, nor the last (See GOAP) So the Monster class was inherited by the FlyingMonster, adding additional methods to control the monsters When the monster wants to move to a location, it first checks whether or not there is an obstacle between it and the goal. If not, it goes straight forward to it, otherwise, it uses the flying pathfinding to reach it.

While implementing this solution, lot of issues were encountered. For example, when I first tried the Graph, it cost me 26 Go of RAM. Luckily I still had some left to close Unity, but it was not very user-friendly. This was because too much edges were made between the nodes of the graphs. Then, another test took 10 min to generate the graph. This solution was light in term of memory, but consuming in term of time. Still a no, loading the map for 10 minutes is a no. The current solution (adjacent cubes) is the fastest and the lightest solution found until now.

Hence the Volumetric pathfinding was done.

We also added events on certain condition of the monsters that can easily be used to trigger some quests. For example, when a monster dies, it invokes its death so anything connected to it is triggered. This project taught us the power of events, and we didn't hesitate to abuse it.

This is it for the A.I. part.
Some bugs were fixed (slime, dragon), ported to multiplayer, and some more code optimised.

3.3 Story

The story was made by all the team and used class defined in the other part. The following text can spoil the story.

When the player first enter the game, the story starts. The first cutscene informs the players of the goal: reaching the temple. It also teach them to follow the light beams to reach the goal and that monsters and puzzles await them.

Then, the two players can start controlling their character. The first quest is reaching their first destination and they're taught how to move. A sword starts shining and they're told how to pick up items.

Their next quest is to eliminate their first enemy: a weakened slime. They're taught the basis of fighting.

Then they've to reach the village after a few more steps, where they'll be able to get a new weapon: the fireball staff. More enemies will appear to try out this new weapon.

After continuing their journey, they reach the gate of the mountain. However, this door is locked by a padlock. In order to break through the door, the player has 2 choices:

- Resolve the mathematical expression next to it and take the first four digits as the code.

- Return to the small village and talk to the inhabitant. A small book, readable only by the Ghost, contains the answer, however, be ready to feel the anger of a mathematics lover.

A tutorial teach the players how to open the padlock and where to find the solution.
Then, the next quest is picking up a new item: far more powerful they could ever imagine: the Spear. They're, once again, taught about the uses of this weapon. They then have to continue their journey, and after taking a portal, they reach a dungeon.

This dungeon is where the core mechanic of the game is played: the difference in view between the human and the ghost. Together they've to resolve enigmas to reach the main room: jumping, connect 4, and later on, a labyrinth. After reaching the last room, 2 waves of monsters need to be killed before continuing. And after a very difficult fight against dragons and slimes, the player can leave the dungeon. For the defense, we placed the end portal at the top of the mountain to avoid the ascension, but it should leave the player at the bottom of it.

After reaching the top, they have to eliminate a final wave of dragons. And then, they can finally reach the temple. When reaching the temple, the end of game cutscene is played. Afterwards, fireworks are thrown nearby the temple, using 2 pieces of crystals for every firework. A beautiful reward for a beautiful adventure.

3.4 3d modeling

As we are not 3d designers, we had to look on the internet for some assets we could use. Yet, we had a quite precise idea of what we wanted in terms of environment. Matthieu did all the 3d modeling that is original to the project.

3.4.1 What was done

Firstly I learnt how to use the blender, then I focused on doing element of the environment. I created wood and stone bridges, fences, but most importantly I modeled modular houses so that we could get exactly what we need.

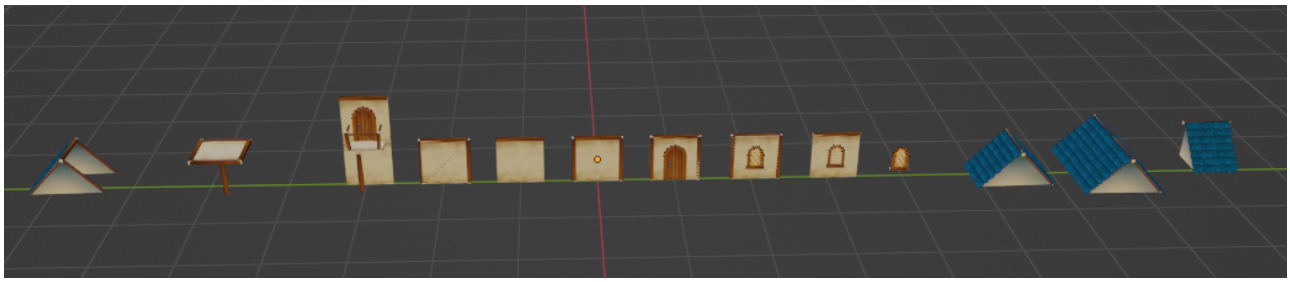


Figure 10: The pieces of the modular environment

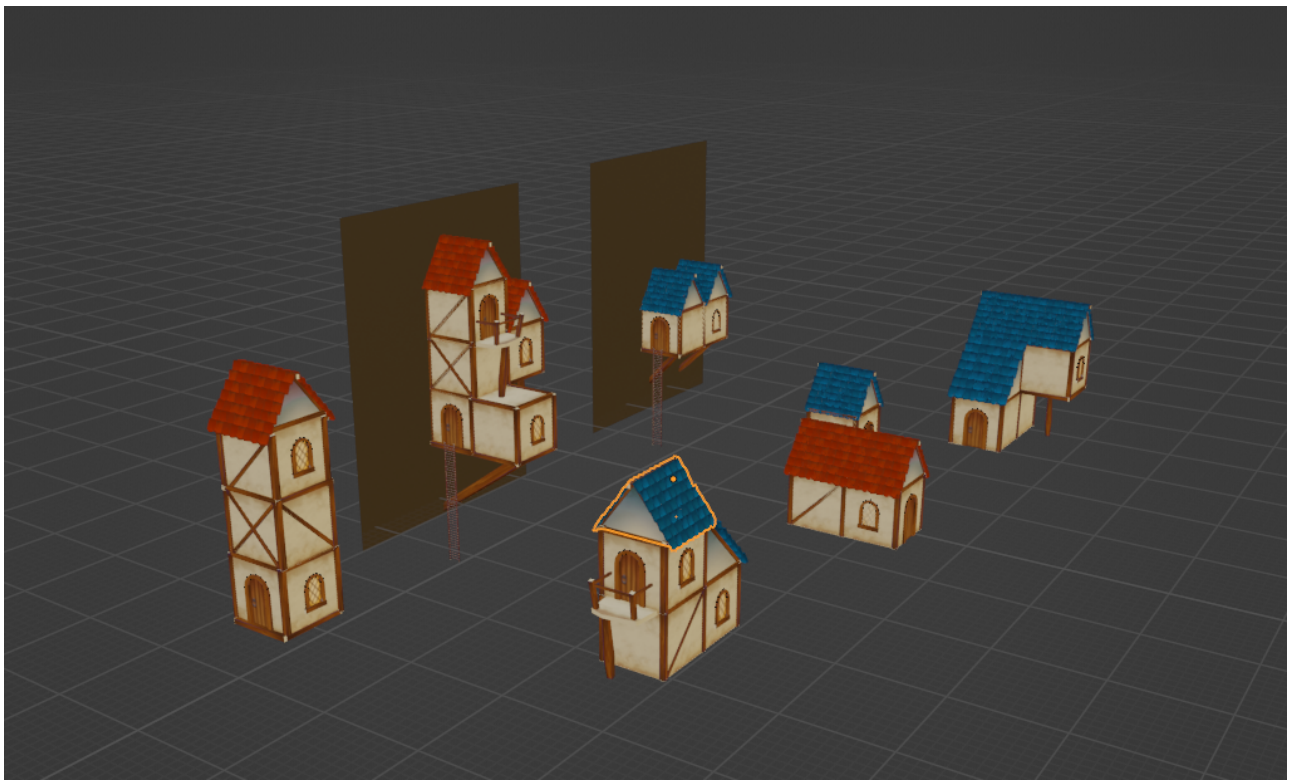


Figure 11: The pieces of the modular environment

Not everything can be shown here, neither everything will be used in the game. But through experimenting I have learnt more in depth how 3d models work.

3.5 Level Design

General Map (Ilan)

The map was already well advanced, yet, it was quite empty, Ilan worked to improved the general level handler while everyone added different scenes to populate the map.

As the leader of this task, Ilan created the basis of the level design. While doing the puzzles, Matthieu created playable areas to integrate in the map.

A map, the place where everything happens. There is no game without a map. A map decides whether a game can be interesting to play or not, since it is the first visual things that the players will see. Our game is in a valley environment. Mountains, hills, snow, forests, lakes, rivers. This is the kind of environment we want to show to the player. A beautiful adventure needs a beautiful map.

The first thing to make is the terrain. In our case, a 1.5 kilometers squared map. The second part was painting the very ground. Creating mountains, hills, etc.

Then, the next step was adding textures to this terrain: snow, rock, grass.

Then, after all that, trees were added on the map. Depending on the location, it was snow trees or normal trees, with a size that differs.

All of these concludes the basis of the map.

We later on revisited the map because the ascension of the mountain was not possible without pressing the Jump action. So for a bit more user-friendliness, we had to soften the slopes, and we used the occasion to change the textures placement.

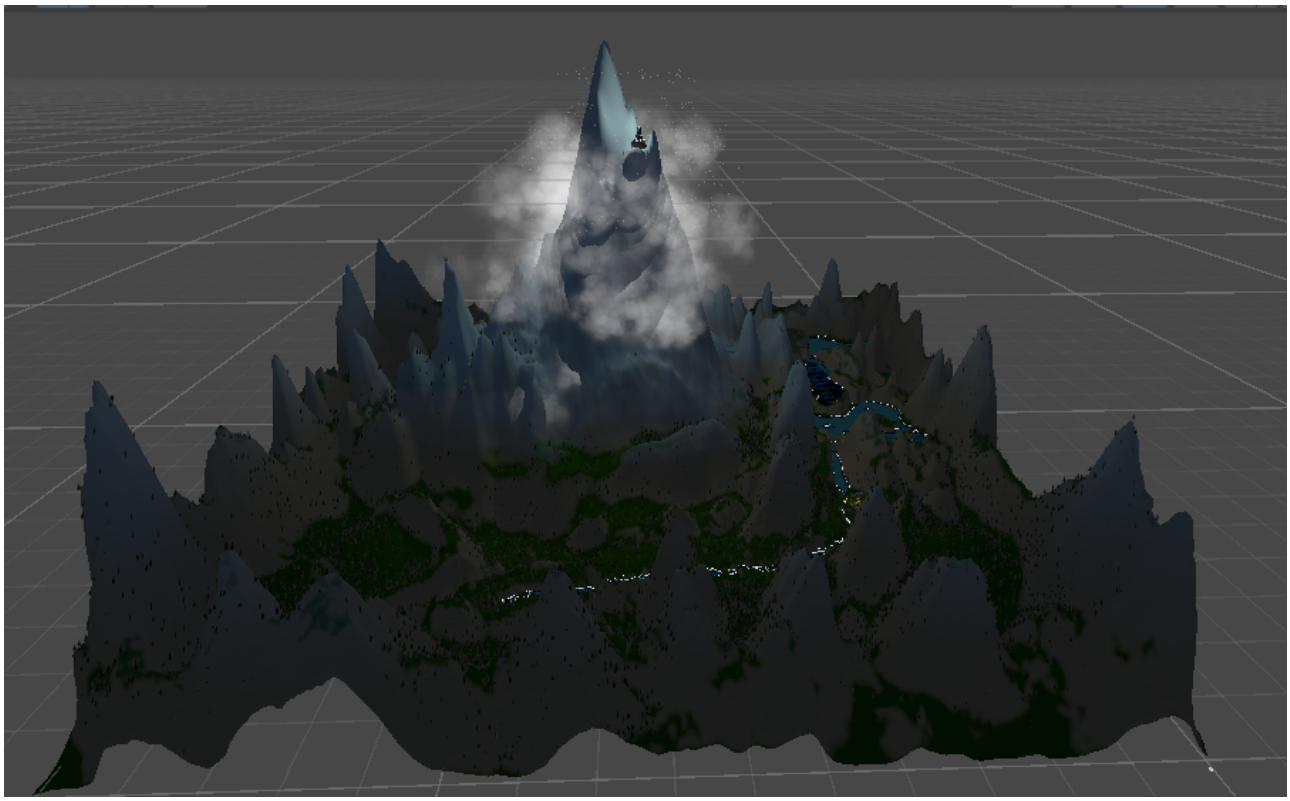


Figure 12: The textured map

In order to bring to life the snowy atmosphere, some particles effects were created: a fog particle and a snow fall particle. Particles can be costly if not handled correctly. Mostly when its about rains and snow fall effects. So it needs to be carefully set up.

Then, some structures were added: A village and the temple of light, also called the church. This church rings the bells at a certain time, triggering enlightenment on the world.

Most of the buildings are from external assets, but some were modelled by the 3D team.

Inside the mountain is the dungeon. Which is accessible towards the end of the game. This allows us to change the mood of the game, to constrain the player in a smaller area to make the fighting harder, to put more pressure on the player to make it less able to think about the puzzles.

Then, the whole structure of the main mountain was revisited, and it now definitely allows the player to climb to the very top without pressing the space bar.

We also added more trees and new varieties of them and grass.

Ephemeral Scenes (Ilan)

A huge map can easily grow heavy. Heavy means performance issues. Performance issues means non-user-friendly. And this is not acceptable as Falling Asleep Studio is the best friends of users.

Therefore, we created the ephemeral scenes, which could save a lot of resources. What is an ephemeral scene? As the name suggests, it is a scene that can easily be loaded or unloaded. When the player enters the area of the ephemeral scene, it loads additively the scene. When he exits, it unloads the scene.

As it can sound very simple, the performance gained are non negligible. And since the game is split in different scenes, it allows us to work on different scenes without impacting each others.

Therefore, the map was split into different scenes, while a Main Scene remains, the one handling everything and the spawning of these ephemeral scene. The MainScene is the one that has all the data that we still want to display: for example the silhouette of the villages, the rivers, etc.
Each ephemeral scenes can check how often it verify if it needs to load itself. To force the players to stay together, only the host of the game triggers the scene loads.

We made several ephemeral scenes, for each area:

- Gate
- LowerLake
- Starter
- UpperLake
- TempleOfLight
- Village2
- Village

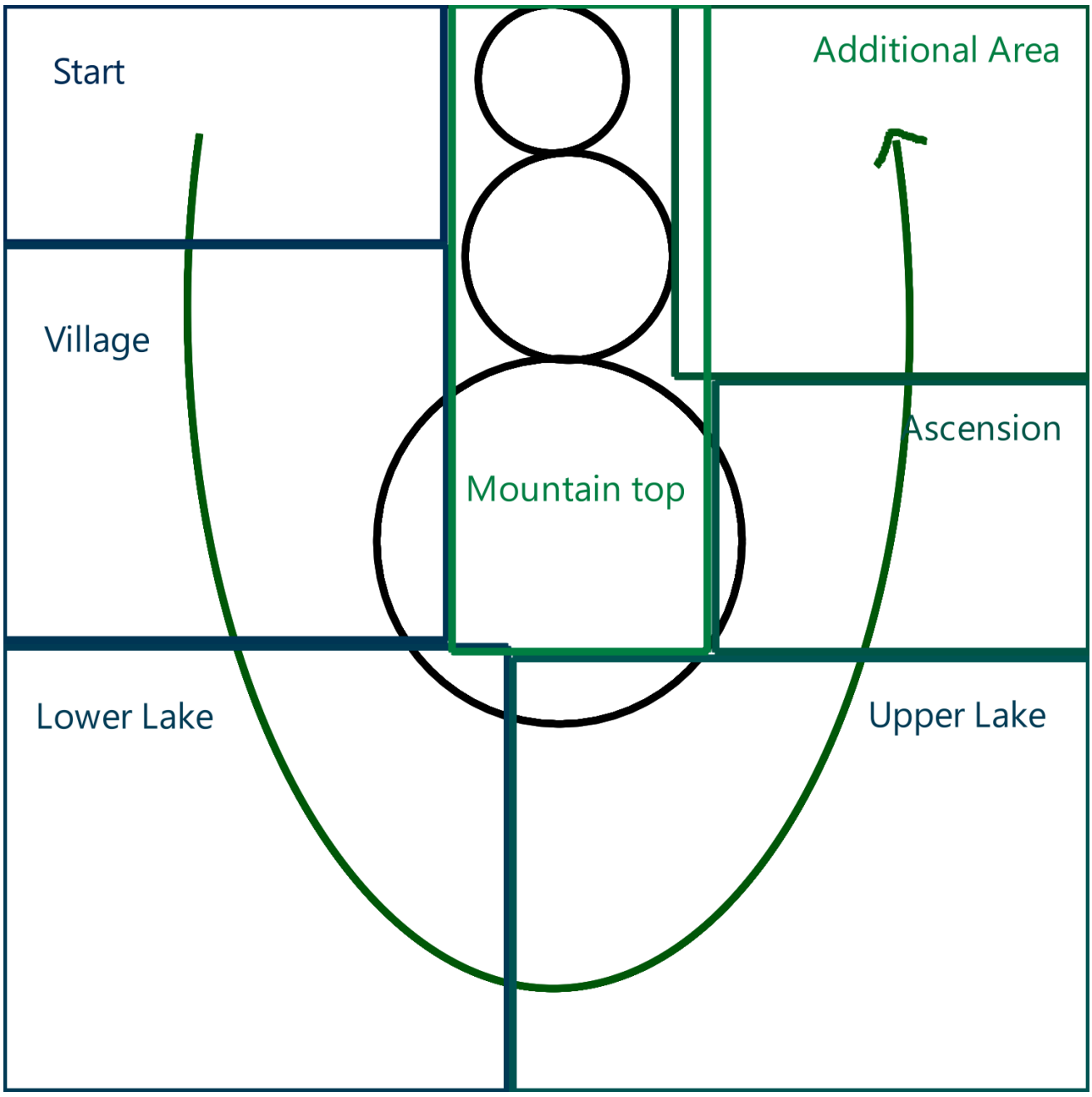


Figure 13: A simplified image with some of the major scenes that splits the map.

Waterfalls (Ilan)

Rivers and lakes were set, yet there was no waterfalls as I was unable to do something clean. After many takes, and creating my own texture, I finally managed to make something that looks like what humans call a waterfall. As these waterfalls are made of particle system, they were put in the ephemeral scenes of their regions.

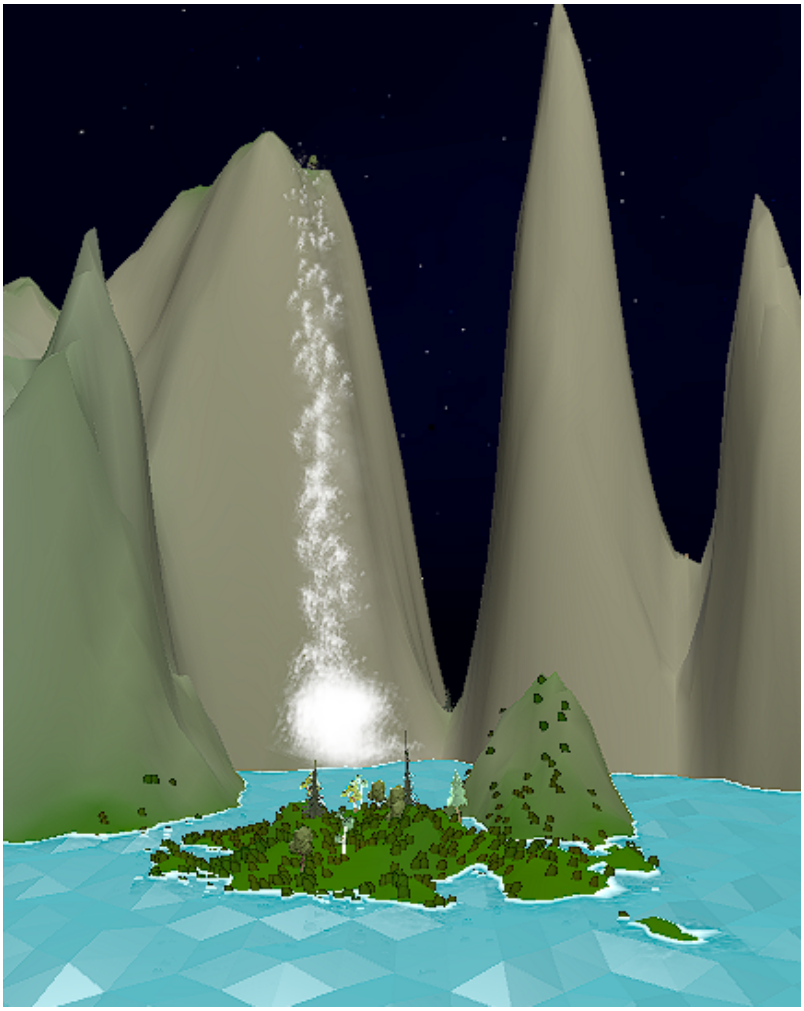


Figure 14: A waterfall

Dungeon (Matthieu)

Welcome, courageous players, to the final and most formidable challenge of Luminosité Éternelle! Prepare yourselves to embark on an exhilarating journey through the daunting depths of the dungeon. This ultimate trial will test your skills, teamwork, and determination in a hostile environment, leading you towards the culmination of your epic adventure.

As you step into the dungeon, be prepared to face a series of puzzles that will require your strategic thinking. Laser puzzles await you, where the manipulation of mirrors and beams of light is crucial. Use your intellect to redirect the lasers, unlocking doors and creating pathways forward.

Once you have conquered the laser challenges, brace yourselves for a thrilling parkour section. Traverse treacherous platforms, swing on ropes, and demonstrate your agility and coordination. Timing and precision will be essential as you navigate through this acrobatic obstacle course. Push your limits and overcome these physical trials to advance further into the depths of the dungeon.

But the dungeon's challenges don't stop there. Prepare to engage in a game of Connect4, but with a twist. A formidable AI opponent will put your strategic thinking to the test. In fact there is a trick here where you'll need to be clever to find the way forward.

As you delve deeper, you will encounter a labyrinth that conceals secrets, hidden switches, and unexpected turns. Work together as a team to navigate the intricate passages and find the correct path leading you closer to your ultimate goal. Communication and cooperation will be paramount as you conquer this labyrinthine challenge.

Amidst the trials and tribulations, be prepared to face fierce monsters lurking in the shadows. Engage in epic battles that will push your combat skills to the limit. Unleash powerful attacks, combine your abilities, and synchronize your actions with your teammate to overcome these fearsome adversaries. Adapt your strategies to exploit their weaknesses and emerge victorious from these grueling encounters.

Remember, the dungeon is the culmination of your journey through Luminosité Éternelle. It is a testament to your growth and development as players. The challenges you face will be daunting, but your bond as a team has grown unbreakable. Your perseverance and determination will guide you through the darkest of moments.

May your senses be sharp, your reflexes honed, and your spirits unwavering. The dungeon beckons, holding the key to your ultimate triumph. Good luck, valiant adventurers, and may the radiant light of Luminosité Éternelle guide you to your well-deserved victory!

Village 2 (Johan)
This village is the second village of the game, it is located at the bottom of the mountain. It is a small village with a two houses, two NPCs and a secret book. The houses, the NPCs and the secret book were all found on the asset store.
The first NPC is Jack, the main NPC of the village. He is the one who gives the quest to the player. The second NPC is Amanda, is the wife of Jack, but she don't bring much in the lore. Finally, the secret book is a book that the player can find in the village. It was belonging to the best friend of Jack, and it contains a secret code that the player can use to open a the gate that lead in the dungeon.

This village, is situated at the base of the majestic mountain range, adding an element of natural grandeur to its surroundings. It is a quaint and picturesque village, characterized by its intimate size, consisting of merely two charming houses that perfectly blend into the serene landscape. Both of these houses, along with the intriguing NPCs and a hidden treasure in the form of a secret book, were assets that were selected from the asset store, contributing to the overall immersive experience of the game.

The first NPC that players will encounter in this village is Jack, an essential character who holds a pivotal role in the quest's narrative. Jack assumes the position of the main NPC in the village, guiding them on their heroic journey. His profound knowledge of the mountain's history and deep connection to its inhabitants make him a compelling character, adding depth and intrigue to the gameplay.

Amanda, Jack's wife, is the second NPC found in the village. Although her significance in the game's lore may be relatively minimal, her presence adds a touch of realism to the village's atmosphere, showcasing the importance of community and relationships within this tiny village.

Within the boundaries of this quaint village lies a remarkable secret book, awaiting discovery by the intrepid player. This book once belonged to Jack's dearest friend, and its pages hold a valuable secret code that, once deciphered, grants access to a hidden gateway leading to the treacherous depths of the dungeon. The anticipation of unearthing this hidden knowledge adds an element of mystery and excitement to the gameplay.

As players delve into this charming yet enigmatic village, they will uncover the tales of its inhabitants, forge alliances, and unlock the secrets hidden within the secret book, ultimately setting the stage for the next phase of their heroic adventure.

Figure 15: The second village

Gate (Johan)

The gate is right after the second village. It is just at the bottom of the mountain and it leads to the dungeon. The gate is composed of three walls, some torches, a door and a padlock. All of the components were also found on the asset store.
On one of the wall, there is a hint for the player to find the code to open the gate. With that hint, the player can find the code in the secret book in the second village or by solving the integral. The padlock itself is composed of four rullers, each one can be rotated by the player. There is a tutorial that explain how to use the padlock. Once the right code is entered, the door opens and the player can enter the dungeon.

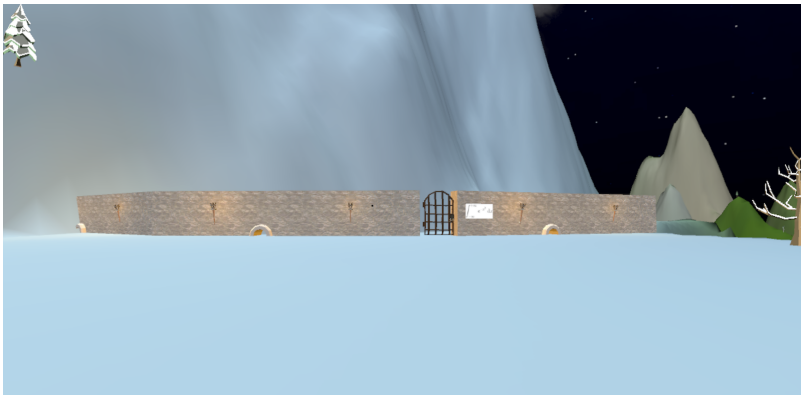


Figure 16: The gate

3.6 Quest organisation

The map we have created is very large, far too large to be shown in its during the final presentation. This is why the organisation of the quests, their placement and their order is very important to show our project in its entirety. The questions follow one another to give a fluidity to the presentation. We plan to speak at the same time as the game unfolds, starting on a plain where we can introduce the introduce the mechanics of the game in a friendly environment.

Throughout the throughout the game we've used portals to allow you to move around more quickly the most interesting parts of the game. Everything is intuitive, with each completed quest triggering the next and so on.

Once the player has discovered the mechanics of the game and the stakes involved, the player is offered a more difficult quest. This takes place further up the mountain mountain and can be solved in several ways. It requires cooperation of both players as well as creativity! Solving this quest takes players to a portal that leads directly to the game's final dungeon. dungeon. This is the moment when both players will be able to put the talents they have acquired their adventure.

The dungeon is designed to use all the mechanics of the game in a very small space, a bit like the finale of a firework display. This allows us to return to aspects of the game that we might have missed in previous parts of the presentation. This dungeon, in the imagination of the game, is inside the main mountain. Finishing it is like climbing to the top of the mountain from the inside.

Once at the top players can finally approach the ultimate goal, the cathedral, and start the the cinematic ending.

3.7 Puzzles

Matthieu took the responsibility to plan and setup the basis we will need to create all the puzzles on the map.

Our game in its duality of worlds has two facets: the first is the combat system which forces players to fight certain monsters and to leave the others to their partner. The second facet is the reflection around puzzles with a particularity in each dimension.

The two players do not usually have access to the same information, nor do they have the same freedom of action. When we design a puzzle, we follow a method: each problem must be new in some way for the player, to some extent. The player must have almost all the knowledge to solve it and each puzzle has a new aspect, which we will be able to use in future puzzles. Each level is therefore in its own way a small tutorial in disguise.

The puzzles themselves are composed as follows: The player should not feel lost, he should even have an intuition of the solution. By following this intuition the player will naturally realise the real difficulty of the puzzle: the "catch".
It can take the form of an inconsistency, an action still unknown to the player or a detail that the player may have missed (a missing object for example). A problem that is too small The moment the player starts to find the game simple, he realizes that that he had underestimated it. This is how we hope to keep the player interested in our project.

Here are the puzzles concepts that we did :

3.7.1 Connect 4

In order to offer puzzles that are not as linear as they sometimes are, we decided to implement a variant of Power 4 in one of our levels.

This is a cooperative version of the board game (both players can play together, which encourages communication and listening). The two players play against a computer that predicts all moves to a certain depth using the minimax algorithm. This is an algorithm that applies to game theory for two-player zero-sum (and full-information) games consisting of minimising the maximum loss. It causes the computer to review all the possibilities for a limited number of moves and to assign a value to them that takes into account the benefits to the player and to his opponent.

The best choice is then the one that minimises the player's losses while assuming that the opponent seeks to maximise them. This algorithm is very memory and computationally intensive when one wants a consequent search depth and a very high level of play. This is why we have reduced this depth to a medium level, so as not to block the player or consume too many resources.

Figure 17: A screenshot of the puzzle involving the connect 4

To come back to the idea of "catch" mentioned above, the players are naturally led to think that they need to win a game to keep going in the adventure, but in fact, it doesn't matter if you win a game or not. To keep going you need to jump on the stone of the game to reach a platform. We know that it can potentially be tough to understand for the player, so there will eventually be an npc here to give advice to the players.

However, the players will soon enough realize there are no other way to reach the platform. While they may think they need to beat the A.I. to win, they will also awaken their human instinct: cheating. In order to continue, they'll try to find another way to reach their goal: which is jumping on the connect4.

3.7.2 Lasers reflexion

Some puzzles are meant to be more about reflexion. One idea we had it to introduce laser that can bounce on mirrors but not on anything else. The ray is coming from a turret and needs to get to an eye. If it does, the puzzle is solved. The turret throw a raycast, and the angle of reflexion is computed by unity every time it hits a gameObject tagged "mirror". A pickup script is also implemented to be able to move the mirror around and solve the puzzle. The ray also does damages so it can be used as a kind of weapon, but the player need to be careful because they will get damage from them eventually. This creates many opportunities of puzzles, with different difficulty throughout the game.

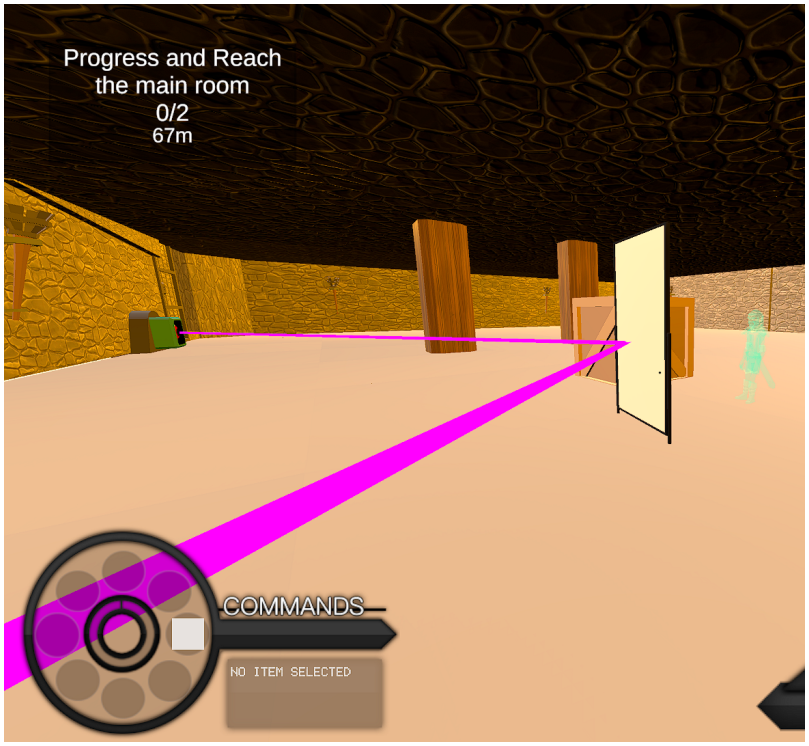


Figure 18: A screenshot of the puzzle involving the lasers

3.7.3 Labyrinth

Who doesn't enjoy a good old Labyrinth ?

The Labyrinth is the last puzzle of the dungeon. It leads to the final room of the dungeon

This Labyrinth is also a puzzle that forces the two players to cooperate to reach the exit. It is made so that the Human falls inside the labyrinth while the Ghost is above it.

The Ghost can see the whole labyrinth structure and freely move above it, unlike the Human that has no clues where to go. Hence, the Ghost has to guide the Human to the exit.

It could lead to many difficult situations of communication between the two players. Luckily, they can see each others so there are many techniques that could be used to solve it.

We love it a Falling asleep studios and tried to give it a little twist. Here as you can see on the screenshot below there is a green platform that only the ghost can see and interact with. For the human it is invisible and he will pass straight through. This will allow the ghost to be able to help the human finding his way through the labyrinth.

Figure 19: A screenshot of labyrinth in the dungeon

3.8 HUD and UI

3.8.1 HUD

This part was done by Ilan.

The main HUD

The Head User Diplay (HUD) is the link between game information and the user. This allow the player to be aware of hidden stats: his health, monsters health, his current weapon and way more.

Therefore, the basis of the HUD were made: the health bar, the username, the mana bar were made and synchronised with the gameplay. Thus allowing to be updated from the player script.

This was done without much issues. The next step was displaying items information. We didn't choose the easiest way to implement since its a wheel. The item that can be selected is on the East side. The player can rotate the wheel with F and G

To implement this system, a WeaponSlot class that can take an Item was created for the 8 slots. Then, the ItemWheel main class followed up with many useful methods. When selecting the next or the previous item, the wheel and slots have to rotate a certain degree in order to keep the images in the right direction and the selected item to the right.

The player can also directly select the item he wants with the keys from 1 to 8.

We also added the Quest display that inform the play of the current quest and its location. It updates the distance between it and the player.

The Quest display listens to the changes of quest by adding an event listener to the On-QuestChange event that is triggered whenever a quest changes. Since the quest changed, it means that the previous one was accomplished (or skipped). Hence, a little overlay to inform

the player that the quest is accomplished appears, and the next quest is then displayed.

The tutorial works exactly the same way. However, the tutorials are locals, so it is not multiplayer connected. When there is a change of the active tutorial, the UI is informed and it updates itself.

With it, the tutorial display, it tells the player how to play the game step by step

Figure 20: The Full Player HUD

The Dialogue

While the gameplay handled the code part, some interactions were needed for the user. When a dialogue is entered, the dialogue box appears from the bottom while the HUD disappear.

Then, the player enters an unlocked stats. Which means that he switches to a new input map. Press [ ] or the left mouse button to trigger the next sentence. Since the player is unable to fight back in the dialogue state, he can exit with [ESC]. Finishing a dialogue also triggers this effect.

When a dialogue needs the user to make a choice, at most 3 choices box will appear above the dialogue box. Selecting a choice triggers a special dialogue or an event.

Exiting a dialogue makes the main HUD re-appears while the dialogue is animated off.

3.8.2 UI

One of Johan's contribution to the project was the creation of the menu, which is the first screen that appears when the game is launched. One of the features Johan has implemented in the menu was the use of a moving camera tool to transition between different options selected by the user.

Menu Description (Johan)

The menu was created using several tools in Unity such as buttons, a moving camera, sliders, sound effects, canvas, and input text. The layout of the menu was designed to be user-friendly and intuitive, with easy navigation between different options.

The buttons were used to allow the user to select different options, such as starting a new game, accessing the options menu, or quitting the game. The moving camera was used to create an animated effect when transitioning between different options.

In fact, the background here was the rendered map in different point of views. For example, the main menu background displays the mountain. By clicking on the "Settings" button, the camera will move in order to focus on the village, while providing game options and audio options adjustments for the user.

The sliders were used to allow the user to adjust settings such as volume and mouse sensitivity.

By clicking on the "Play" button, the camera will focus on the top of the mountain, that is to say the church that has on its top the "Luminosité Éternelle".

Sound effects were used to provide feedback to the user when buttons were clicked, and to add to the overall user experience.

Input text boxes were used to allow the user to enter the room names for creating and joining a room when the room is private.

If the room is not private, it will appear in the scroll view with several attributes such as the room name, the number of player currently
in the room and the number of players that the room can handle. Here, this last number is always set to 2.

Challenges encountered (Johan)

One challenge that I encountered was ensuring that the user experience was consistent throughout the menu. This required careful attention to detail, and making sure that each tool was used in a way that was consistent with the overall design and user experience. Unfortunately, some tools need to be improved but we will see that in the next part.

Syncing the HUD to the Game (Ilan)

The Menu done by Johan was on the multiplayer part of the game, and was not linked to the game. So after merging the two branches, I took the role to sync everything with the gameplay. So that, when the player adjust the sensibility, the inversion of the mouse, the sound, it modifies the settings.

For the sound, an Audio Handler was made. By default, the volume are at 100%, having the Audio Handler giving the Audio Source 100% of its original sound. When reducing it, it reduces the volume settings and all the audio handler adjust the sound to a certain percentage of the initial volume. Therefore, it allows having a variety of volume and not everything to 1

or 0.

Of course, these settings are saved and the player won't have to set them everytime the game is started.

We later linked the change of resolutions.

3.8.3 Room Menu (Johan and Luca)

The same tools were used to create the room menu, which is the screen that appears when the user creates or joins a room. The room menu was designed to be consistent with the main menu, and to provide a similar user experience. The background is different as the main menu, but the buttons are the same.
The name of the room is displayed at the top of the screen. The user can change role by clicking on the buttons on the left side of the screen and on the right side of the name of the player. By clicking on the "Quit" button, the user will leave the room and return to the main menu. Finally, by clicking the "Ready" button, the user will be ready to start the game and it will activate a checkbox. It will also lock the role selection, so that the user cannot change role anymore. When all players are ready, the game will start.

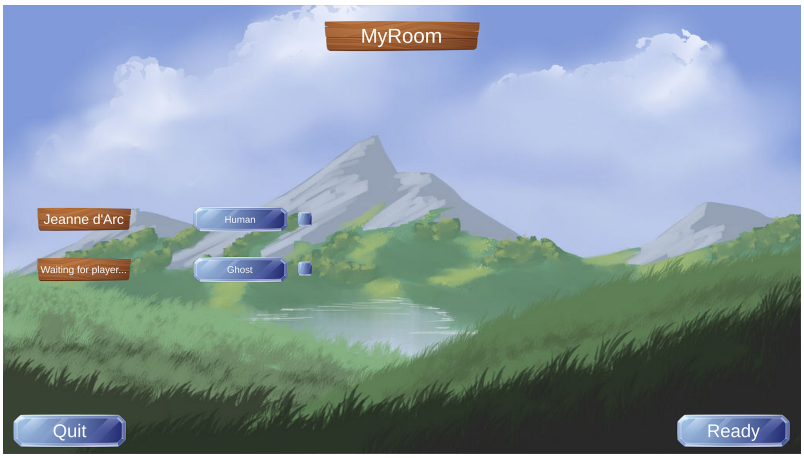


Figure 21: The Lobby to start the game and choose the roles

3.9 Website

Welcome to the official website of Luminosite Eternelle! We have designed this website to encapsulate the spirit of our game and provide you with a seamless and immersive experience. Rejecting conventional templates, we have crafted a unique interface that reflects the essence of our game, bringing you closer to its world from the very moment you land on our homepage.

Upon entering our website, you will be greeted by a captivating video trailer that plays right on the homepage. We believe that visual storytelling is a powerful tool, and we want to provide you with an immediate glimpse into the captivating world of our game.

The website features a comprehensive presentation of the game itself, giving you an in-depth understanding of its unique features, gameplay mechanics, and storyline. We have carefully curated the content to provide you with an immersive experience, allowing you to delve into the world we have created.

As creators, we have dedicated a section to introduce ourselves. We want you to know the minds and hearts behind this project. Learn about our team, our inspirations, and our vision for the game. We believe that this personal touch will help you connect with us on a deeper level and understand the passion and dedication we have poured into this project.

Scrolling down the website, you will find download links to access the game. We have made sure to provide convenient options for different platforms, ensuring that you can easily embark on this thrilling adventure regardless of your device or operating system.

To further enhance your experience, we have also included links to all the assets we have utilized throughout the development of the game. This allows you to explore the behind-the-scenes elements, dive into the artwork, sound design, and any other resources we have employed to bring our vision to life. We believe in transparency and want to share our journey with you.

In conclusion, our website is a testament to the dedication, creativity, and innovation that drives our game. We have forged a unique interface, eschewing templates to create an immersive and captivating experience. From the video trailer that greets you to the presentation of the game and its creators, every element has been meticulously designed to transport you into the world we have created. The download links and access to the assets further enrich your journey and invite you to explore beyond the game itself.

We invite you to explore our website, learn about our game, and join us on this incredible adventure. Welcome to our world!

3.10 Animations

The gameplay adaptation was handled by Ilan.

Animations are what brings the game to life. It brings comfort to the player.

Most of the players and monsters animations are from several assets. The basic animations were handled by us.

The dialogue opening and closing animation for instance. The zone attacks were also handmade as for the snow and the fog.

More animations were added and successfully synced on the multiplayer. This include adding swimming, falling, and other animations.

3.11 Multiplayer

As a cooperation game its multiplayer aspect is essential. This part was done by Luca.

Since Networking is a complicated and tricky field of programming the first step in implement multiplayer gameplay is usually to choose a good framework. As a framework also designed for indie developers, Unity provides its own called Netcode For GameObjects but many other companies sell their own.

Ultimately, we had to choose between Netcode and the PUN (Photon Unity Networking) framework provided by Photon. We ended up choosing PUN due to its emphasis on simplicity of its API while still having all needed features such as Component Syncing, RPCs, Event-Based Callbacks, etc.

PUN

Pun is based on a system of lobby and rooms so a PUN client is either disconnected, connected to a master Server (in a lobby) or in a gameServer (playing online). This system comes in handy as it is similar to the stages of our game: A closed game is disconnected, a client being in a lobby is in the game's menu and finally a client in a gameServer is going on an adventure.

The implementation of Multiplayer is a bit different than other sectors as implementing multiplayer is mainly about integrating it. By that we mean that the main task is to edit assets and Unity Components to support multiplayer.

In The Menu

In order for friends or strangers to play together they first need a way of finding each other. Here PUN comes in handy with its lobby system as it allows to list all opened room. So a player can either join a pending room (a room that only has 1 player waiting) or create a new one possibly private so his friend can find its room by name.

In Game

When a player is in a game, he is according to PUN in a room (a multiplayer scene), but if many people edit the same map at the same time, conflicts may arise. In order to prevent those conflicts between clients we use a master-slaves design meaning that the Client Creating the room (the online game instance) has the authority over the Unity scene (the map), we will refer to him as the host (even though he technically does not truly host the game, he and only he can create and remove all Unity gameObjects outside of Players).

So the host as the authority over all gameObjects outside of other Players, this is defined by the 'PhotonNetwork.IsMasterClient' boolean set to true, so he and only he will actually update any animation, the AI system as well as any NPC.

On the other hand, Players work a bit differently. Any Player has authority over its own Player Unity gameObject, meaning that he will handle, movement, fields and gameplay events such as attacking or doing damage.

But then, How can an AI handled by the host attack a player handled by a client ? Well, for cases like these we use RPCs (Remote Procedure Calls), this technology enable us to call a function over network.
For Instance, a Slime that attacks a player will call the TakeDamage(dmg) function associated to the Unity gameObject on the host, this function will check if the Player is owned by the host, if so, it will simply call the true TakeDamage(dmg) function locally, if not it will call the client's TakeDamage(dmg) function over RPC. The client will then lower its HP (health Points), start animations, etc...

During the first defense, the multiplayer was late on the project.
We didn't meet our objective and were forced to show the project in 2 part: the main game and the multiplayer game. The multiplayer game consisted on slowly adapting the main game in multiplayer.

While the team was working on their tasks, Luca took care of adapting the code to the multiplayer.

On the first defense, the project was split in 2: the main game, and the game with parts being synced to multiplayer and the menu.

This was the main issue. For the second defense, we worked hard to make everything in one project.

First, we had to reconsider our strategy. Instead of having Luca works on adapting the code, we also started writing ourselves the multiplayer version. This meant we had to learn how to use PUN. This way, there was less work for Luca.

At the end, Ilan and Luca were both working on the multiplayer and fixing the errors, doubling the speed.

- Ilan wrote the code to synchronise the quest system. Quests can be changed by anyone, and it informs what is the current quest.
- Luca adapted the AI system to the multiplayer, making it run only on the room master and copying its move to the client.

Luca also worked on the start menu multiplayer system. This system allows the player, after creating a room, to wait for the second player. When the second player joins the lobby, the room master can change the roles of the players. This mean inverting whether or not he is the Human or the Ghost.

When both players agreed with their roles, they can click on Ready. If both players are, then the game is launched.

Ilan fixed the bug where the second player was not spawned. The difficulty in fixing this bug is that it was happening very rarely, and catching it was difficult since it was not making any error.

After a lot of analysis, he discovered (with a lot of luck) that it was because, somehow, the player was instantiated in the LoadingScene and not in the MainScene. By so, when the MainScene finished loading, the player inside the loading scene disappeared with it, having the second player not being able to play... To fix that, instead of relying on the AsyncOperation of the SceneManager, we instead relied on PhotonNetwork current scene. As long as the current scene is not the MainScene, we kept waiting before instantiating because even if Unity told us it was finished loading, it didn't mean it was considered as the main scene.

Moreover, after a lot of debugging, the slime started working again. While it was working perfectly fine, the GOAP system in multiplayer is very difficult to maintain. The fact that it

had to copy its movement made the task difficult.

Issues caused by the multiplayer

Similar to what will be talked in Gameplay, multiplayer made our development progress slower. Mostly because of our system and our way of implementing it and the rest of the project.

During the first part of the development, there was not much issues as the multiplayer was not implemented with the main project. However, when we started to merge the two projects, we had to adapt the code to the multiplayer. And this is where it started to be difficult.

Since the game was multiplayer, we had to make sure that the code was not only working for one player, but for two. Hence, when we coded, we had functions made for the multiplayer. But it started breaking apart with the rest of the thing: the testing. When we had to test the bug, we had to test it with two players, because otherwise, the code was screaming at us. This mean, we had to start the game from the menu, create a room, join the room on a second instance of Unity, start the room, wait for the loading and then go where to test the bug. This was very time consuming, and made the development slower. Each time we wanted to test if one line of code made thing better, it was costing 2 minutes instead of 10 seconds, the timi to load an empty scene and check for the bug.

We don't really know how we were supposed to do it, but we think that we should have made the code work for one player, and then detect if it should behave as a multiplayer instance . This way, we could have tested the code without having to wait for the multiplayer to load.

3.12 Music and sound effects

At first, there was no sound at all. But we ended adding S.O.U.N.D.S!

Sound is important for the immersion in a game. It can change the whole meaning of the game.

That's why we added a maximum of free sounds to the game!

3.12.1 Music System (Ilan)

We wanted to play musics in each area of the game (cf Map Area)

We also wanted to do so that, the music fade out before starting a new one, so that it is smoother when playing. After selecting musics that fit the theme of each area, I made the music system.

The principle is very similar to the Ephemeral Scene system. Different Areas with their musics. When entering the area, it takes the control of the music. Hence, the previous music fade out while the new one fade in. The way it is done is smooth and the user experience is not damaged from a brutal change

3.12.2 Added sounds (Ilan)

Many ambiance sounds were added: water, items, walking and so on. There are improvements to do, but in the general aspects, the result is way better, and the immersion improved.

3.12.3 Setting sound volume (Ilan)

When we wanted to link the Menu to the player settings, we had to find a way to change the volume of all the AudioSource.

To do so, we created a MusicHandler. We put a MusicHandler to each of the AudioSources. The MusicHandler has two types of AudioSource:

- Music: Everything that is linked to a music
- Sound Effect: Everything that is produced by the environment

When a MusicHandler is initialized, it fetch the percentage of volume it has to set himself to. Each time percentage is independent. Hence, when the player modifies the settings volume bar of the corresponding type, the percentage changes.

With hindsight, I realized that what I did was totally useless. I was not aware at all of the usage of the Mixers that Unity gave. And it was exactly what I did, except better and with more functionalities.

Overall, I'm still happy because I think that my method is not the worst one, and at least I know how to do a system like this one now.

3.13 Cutscenes

Part handled by Ilan, as the book of specification mentioned.

Cutscenes were an additional thing mentioned in the book of specifications. They were considered as not mandatory, depending whether or not we succeeded reaching our goals. As we are almost done with our goals, cutscenes are getting added. For example, the end of game cutscene is almost done, roughly 2-3 minutes long and of course, better than any AAA game. The trailer cutscene is getting improved day after day.

A cutscene handler was made. When the player enters the cutscene, it deactivates the HUD, display some "cutscene bar" and change the input map. If the player skip the scene (with N for now) or that the cutscene ends, it goes back to the previous state.

We have hence a total of three cutscenes:

- The Start Cutscene: It shows as said in Story the base of the story of the game. What happens, how the Ghost died, why they have to reach the temple. Duration: 110s
- The Trailer Cutscene: It shows the whole map step by step. Duration: 120s
- The End Cutscene: It shows the two players reaching the temple; the Ghost being saved; the purification wave. Duration: 120s

Figure 22: A screenshot of the start cutscene where the temple talks to the players

4 Personal experiences

4.1 Matthieu

I had heard a lot about this project prior to the S2, before joining epita and during the S1. We were so hyped about it and spent weeks talking about the essence of the game, what we would like to see in it, what we enjoyed playing in other games. We handcrafted ideas about the story and mechanics of the gameplay so that they would go well together. We got really inspired by the zelda games as well as we were here as mentioned in the book of specifications. I immediatly got into blender, following tutorials on youtube. I learn a lot about how 3d objects are made and work, especially how lighting is made by professionals and how sometimes things that we don't think about when playing a video game can have a big impact on it (such as lighting and texturing for example). I got into texturing as well but quickly realised that it was a job to itself and that my time and energy would be better used elsewhere. So we started using other assets for the rest of the creation. I don't think I lost my time though as I learnt and understood a lot about how video games are made. After that we kind of all started making our own unity projects, trying things out, but merging it all was such a pain that some puzzles and some scene have had to be made again.

After some experimentations we all got to to work on the same projects and the merges conflicts, even though we were still scared of them, weren't as bad as before.

Using unity everything was new, and sometimes we learnt stuff the hard way, realising that we could have done stuff that took us a lot of time in a much more efficient manner. I was a good exercice, but I was feeling quite good while taking care of the website after this since I felt much more confident about what I was doing. I am proud of the website as I didn't use any template and yet I learnt how to use a new framework. I think we kept it coherent with the essence of the game and that a good bonus !

Througout this project I learnt a lot about working in group, how it is different from working alone, sometimes more difficult, sometimes more frustrating, but at the end the project we've built is so much greater than anything I could have done alone and that's the true reward.

I'm happy and proud that we got that far in the project and I can't wait to start the next one and keep learning stuff !

4.2 Ilan

This project finally reached an end!

After 7 months on it, I'm glad it's finally over and we completed it! We learned plenty of new things, made our own game, and in my opinion, quite an impressive game with the fact that it was only on the spare time we had. Between the weekly practicals of prog, the midterms, the finals, the MCQs, the TDs, sleep, finding time to work on this project was difficult. Yet, we managed to pull out something!

We had no prior knowledge on Unity, but we slowly learned from it.

After working and learning on this project on 7 months, I realized that we did plenty of thing wrong. For example, we didn't use Scriptable Objects at all. It could have been perfect to store our dialogues, but instead I made my own interpreter of files... We didn't use Mixer at all, and I did my own equivalent, which is of course in a less good shape. If I knew about it and how to use it, we could have done things way better and smoother.

Moreover, we had too many dependencies on our scripts. I don't really wanna imagine the graphs of references we have, but it must be horrible. And because of that, testing was getting more difficult each time we added something.

We also didn't manage to properly work together and tasks were a bit random by the end.

This is of course a great teaching experience. Together, we managed to do all of that from zero. I would not manage to this alone and this is the result of our work. This is already a lot of experiences and I am excited to discover what await us next.

It gave me the motivation on working on video games and develop my own open source game right after the end of this project. I'm just worried that after every projects I want to start a new open-source project on the subject: I would clearly die of exhaust. But this theory is false because I didn't do anything after the AFIT, yet it kept coming back at me.

I managed to learn how to make a map, paint it, plant trees and most importantly: add grass that can be touched (very important for an CS engineer)! I also learned how to code the GOAP system and implement basic monsters' AIs. I managed after a lot of sweat how to make my own particle system for waterfalls and many others (especially waterfall). I also made the whole gameplay basis, the quest system!

At first, I was overwhelmed and had to watch some videos to understand how they made things like mouse controllers. Now I feel like I could do anything I imagine.

To conclude, this project was 7 months of stress, fun and learning!

4.3 Johan

We finally did it !

I'm really happy to have worked on this project. I learned a lot of things, and I'm really proud of what we did. I also think that our team worked well together, and that everyone participated as entended in the projetc. This was my first experience as a game developer, and it as really cool to do it with a team like that one. This project gave us a lot of experience, and I'm sure that it will be useful for the next projects. For example, how to manage (git) conflicts, how to work in a team, how to use Unity, and so on.

I always thought that game development was really cool, and I'm really happy to finally try it. With this project, I've learned some UI development, how Unity works, how scripts are made and how the player can interact with the game. since it was the first time I've done this, it was really nice to create dialogues, and I think it brings an additional good strory into the game.

Finally, it was nice to code the padlock interactions, and I'm really happy with the result.

I'm really happy to have worked on this project with that team, and I'm looking forward to the next one !

4.4 Luca

I like to discover new things. This project is an opportunity for me to broaden my knowledge of computers by working in a group and on an application (unity) that I am using for the first time. We have high ambitions for our project, so I'm looking forward to continue implementing new features

5 Conclusion

The development of our first game, Luminosité Éternelle, has come to a close, marking the end of an incredible adventure for our team of four. Throughout this journey, we have poured our hearts and souls into creating a game that revolves around the theme of friendship and the reliance on our teammates to achieve our goals.

From the outset, we envisioned Luminosité Éternelle as a fantastical experience, drawing players in with its captivating art direction and immersive context. Although we were initially concerned about the game's difficulty for children, we strove to make it accessible and engaging for players of all ages.

One of the core gameplay mechanics that sets Luminosité Éternelle apart is the dynamic of having one player alive while the other is dead. This mechanic allows for unique perspectives and experiences as each player gains access to different conversations, clues, and interactions with the environment. Their shared objective is to ascend a treacherous mountain, hoping to find a means to resurrect the fallen player.

Throughout the development process, we have prioritized delivering a great experience to players. Paying attentions to the details, we developed the gameplay, monsters, puzzles and cinematics to enhance the overall impact. Our intention is to create an experience that resonates with players.

we can't help but feel an overwhelming sense of pride in what we have accomplished as a team. The development of this game has surpassed our initial expectations, and we are grateful for the opportunity to bring our vision to life.

As we conclude this chapter of our journey, we look forward to sharing Luminosité Éternelle with the world. We hope that players will find joy, inspiration, and a renewed appreciation for friendship through our game. The bonds we forged as a team during this development process will forever be cherished, and we eagerly await the next adventure that lies ahead.

This concludes this adventure but far more are awaiting us.

We hope you will enjoy our game,

Sincerely,
Falling Asleep Studio's team.

6 Appendix

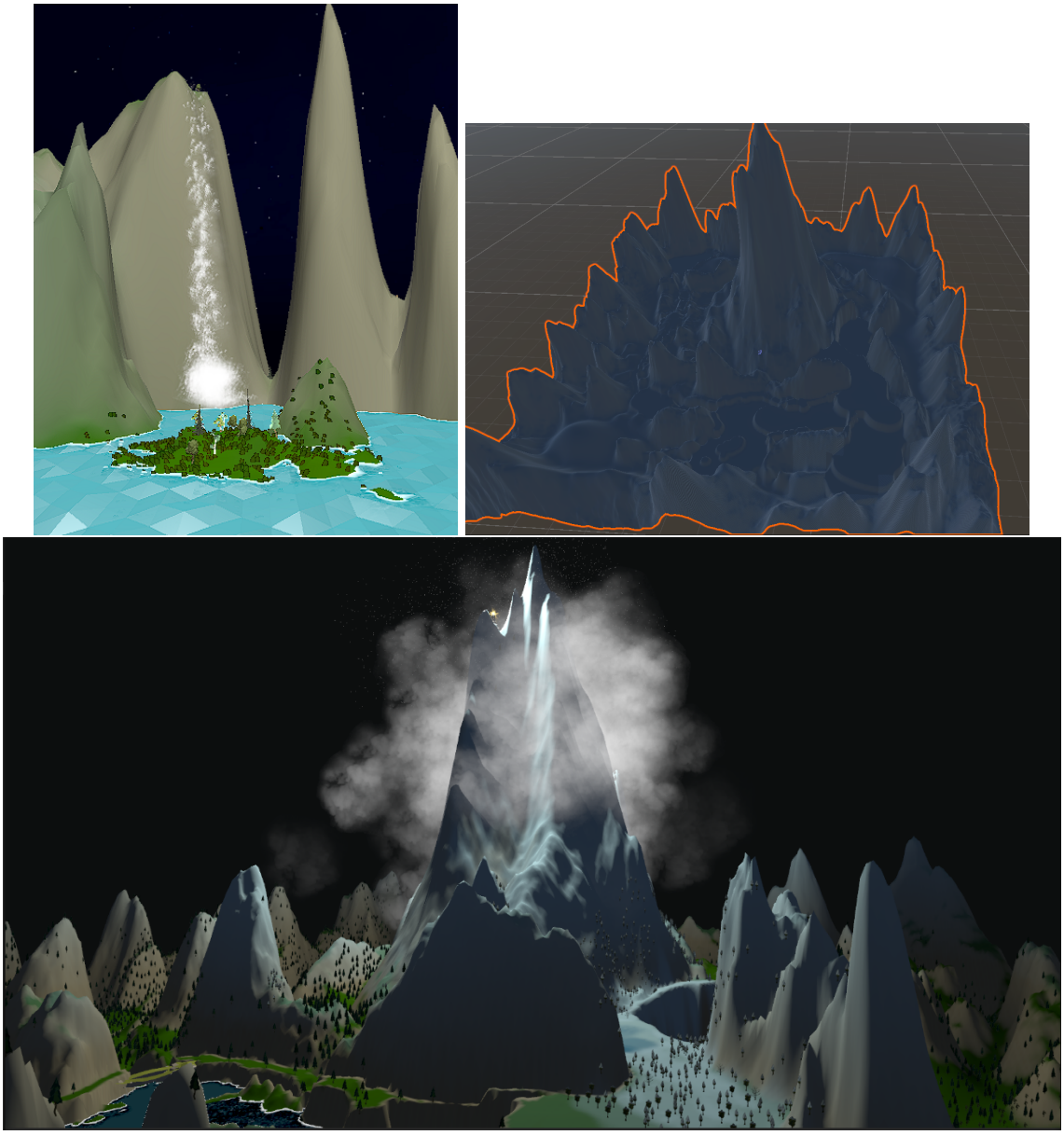


Figure 23: The textured map

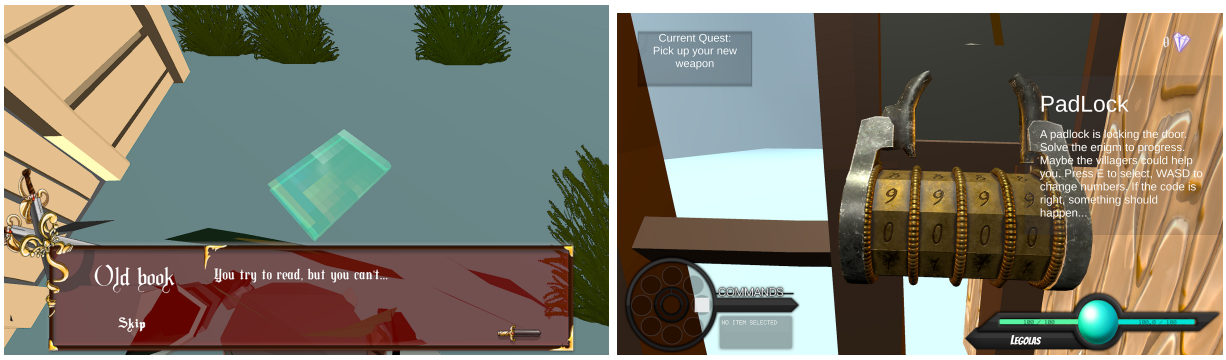


Figure 24: The padlock and the old book read by the human

Figure 25: The fortified village

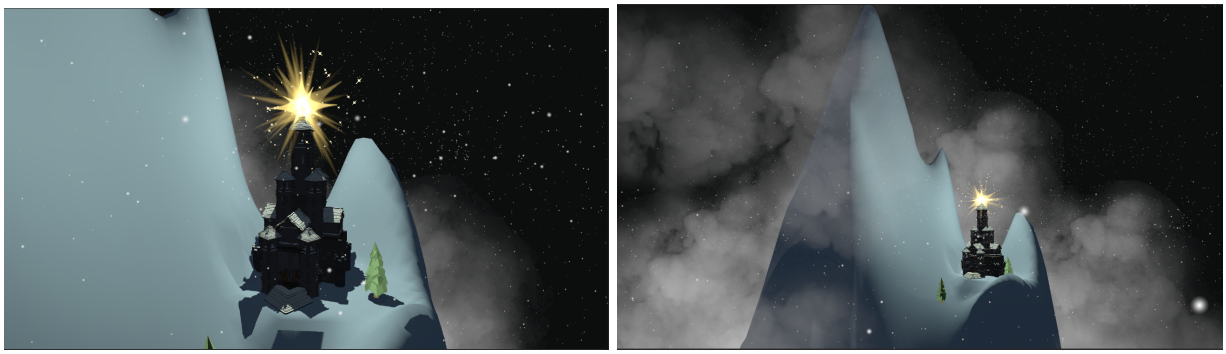


Figure 26: The church / Temple of light

Figure 27: Some gameplay images

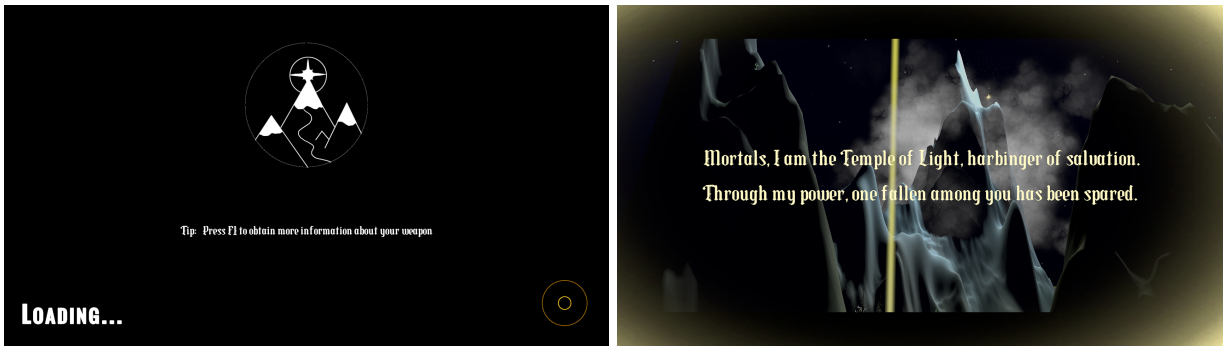


Figure 28: The loading screen and the start cutscene