First Defense

Incroyqble

April 17, 2023

Abstract

This document covers what has been achieved during the first part of the project. We'll discuss how we handled work, separately and in a group, cover what challenges we've faced as well as what we hope to achieve during the following periods.

Contents

1 Introduction

In a distant future, in a galaxy far, far away, our hero awakens to a scene of devastation. A blood-red moon hangs low in the sky, casting a harsh, unforgiving light across the desolate landscape. The air is dry and still, suffused with a gritty heat that scrapes at his throat and lungs. As he takes stock of his surroundings, he realizes with a growing sense of dread that everyone in his village is bedridden, and a deep instinct tells him that the same fate has befallen his entire universe.
Desperate for answers, the hero seeks out the wisest elder of his clan, hoping to learn the truth about the cataclysmic event that has befallen his world. There, in the sacred chambers of his people, he hears the terrible news: an unknown force is draining the life force from their world, a force so insidious and powerful that it threatens to snuff out all life in the galaxy. And time, he learns, is running out.
With a heavy heart, the hero realizes that he must take action. He sets out to the legendary Ancestor's fountain, a place of great power and wisdom that has always held a special attraction for him. But as he makes his way across the barren landscape, he realizes that he is not alone in his search for the solution. Other cultures, hungry for the power to save their worlds, will stop at nothing to claim it for themselves.
As he nears his destination, the hero senses a growing urgency. Time is of the essence, and the fate of the galaxy hangs in the balance. But even as he approaches the fountain, a dizzying sense of vertigo overtakes him, and he stumbles and falls into the spring.
When he awakens, the hero finds himself in a strange and alien place, surrounded by architectural elements that bear no resemblance to anything he has ever seen before. With mounting trepidation, he realizes that he has been transported to a realm beyond his understanding, a realm where the fate of his world may well be decided.

During this period we've mainly focused on learning how to reach the different goals we've setted while using unfamiliar tools. This implied learning how to efficiently use Github and we also had to learn how to use Unity, as neither of us had a recent experience with this software. The second part of this period was dedicated to creating the workflow that would allow us to build the finished game much more efficiently. Each member will present what they did in this time.
As the project became more and more tangible, we realized that the expectations we had set in the Book of Specifications didn't match with what we needed to achieve, ad how it could be achieved. This is why we moved deadlines and task distribution. We now hope that we will be able to set better and more realistic expectations. Nonetheless we still managed to set up individual components that ensure a good start for the project.

2 Pierre

2.1 Tasks

- Game design
- AI

2.2 Achievements

- Basic game design
- Storyline

2.2.1 Basic Game design

I was tasked with making the design of the game through multiple means, the first one being to create a ruleset for the players.

2.2.1.1 Ruleset

The fundamental aspect of the ruleset is the interaction between the player and their surroundings, as well as other players. As a result, these interactions are categorized into three distinct parts:

- Objects

Throughout the game, players will engage with a multitude of objects, beginning with abilities. These abilities are generally acquired towards the conclusion of a level and can be employed to facilitate the player's advancement, such as executing a double jump or dashing. To ensure a seamless experience for the player, a brief tutorial section will be included upon the acquisition of each ability, thereby mitigating any potential confusion.
Furthermore, players will also encounter weapons during gameplay, which serve to enhance their offensive capabilities and may feature unique characteristics. Some weapons may incorporate specific gimmicks, differentiating them from their counterparts.

- Characters

The players will engage with various characters that can be categorized into three groups. The first group consists of enemies, which will not be unique and are expected to appear in multiple quantities on each map. These entities will be found roaming in specific locations, and some will pursue the player upon entering their line of sight, while others will not. Upon defeating enemies, the player may receive a random chance to obtain a healing item as a reward. These items are not considered objects because they do not have a lasting impact on the gameplay.
The second group of characters that players will encounter are bosses. These are unique entities that will temporarily alter the game mechanics, such as locking the player in a particular area and displaying a health bar. Bosses will either use their own body to attack the player or project projectiles that the player must avoid.
The final group of characters consists of NPCs that players will encounter. These characters may either aid the player in their journey or offer objects that may assist the player in some way. NPCs will be found moving about in specific areas and will engage in conversation or trigger an event when the player interacts with them.

- Players

In certain instances, players may encounter each other and engage in combat at the culmination of a level. In such circumstances, the victor of the altercation will be granted an advantage over the opponent, which may range from an improved weapon to a mere head start in the ensuing race. During these confrontations, the rules governing player-enemy combat will apply, encompassing kill/hurt boxes and health depletion.
It should be noted that the victorious player will not acquire any critical abilities, ensuring that the defeated player has a chance to stage a comeback. In the ultimate level, the players will contend against

both each other and a boss concurrently. In this scenario, a player who perishes will be transported back to the start of the level, thereby affording the surviving player sufficient time to tackle the boss by himself.

2.2.1.2 Entity abilities

The different entities along the way will display different abilities all changing how the player interacts with them. First there will be one "basic" enemy per level which will not have any kind of special ability, they will simply attack the player when they enter its line of sight (see 4.3.1 Bokoblin).
Some enemies however will have special abilities, for now we have not completely outlined the extent of every ability but the guardian enemy is a good showcase of what we expect for the future (4.3.2 Guardian). A few of the other ideas we have for special abilities include:

- Side view
  - flying enemy
  - ranged enemy
  - charging enemy
- Top view
  - ranged enemy
  - area of effect enemy
  - teleporting enemy

2.2.2 Storyline

The plot of the story involves the emergence of an instability within a world abundant in references to nerd culture. The objective of the different players is to locate a unique artifact that has the potential to restore stability to their universe. However, since this artifact is capable of repairing only one universe, the players will have to engage in inter-dimensional travel and compete with one another to acquire it. The final obstacle is presented in the form of a boss character who possesses the artifact, and the player who successfully deals the finishing blow will claim the artifact and use it to save their world. A timeline has been provided below, outlining the fundamental universes that players will encounter throughout their journey.

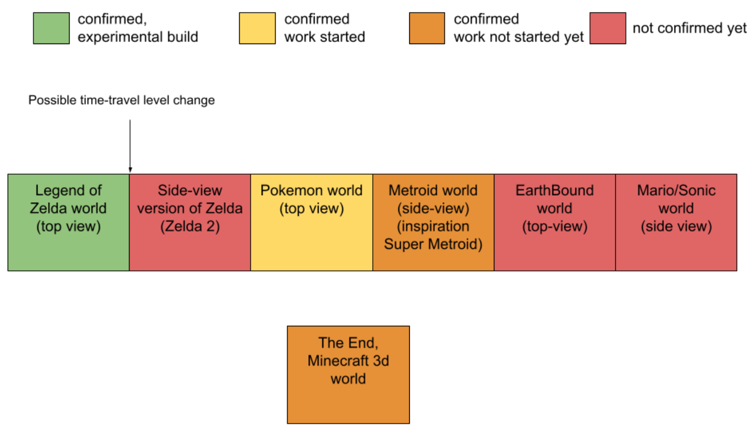

confirmed, experimental build
confirmed work started
confirmed work not started yet
not confirmed yet
Possible time-travel level change
Legend of Zelda world (top view)
Side-view version of Zelda (Zelda 2)
Pokemon world (top view)
Metroid world (side-view) (inspiration Super Metroid)
EarthBound world (top-view)
Mario/Sonic world (side view)
The End, Minecraft 3d world

Figure 1: Storyline basics

2.3 Difficulties

One of the primary challenges that I have encountered during the initial phase of the project has been related to the coding aspect. Despite my prior experience with coding projects in collaboration with peers, I have had to defer a considerable amount of coding work as I have not been adequately familiar with the intricacies of Unity. This is a recurring issue that I have faced in the past, where I tend to feel uncertain about how to proceed with coding tasks at the beginning of a project.
In contrast, I have found the game design aspect to be relatively straightforward and manageable, owing to my pre-existing experience and proficiency in this domain.

2.4 Future Goals

I aim to expand my coding skills and overcome my apprehension of the unfamiliar, enabling me to achieve greater productivity following the initial defense. My aspirations include pursuing map designing, leveraging my prior experience in this domain, as well as my proficiency in level design. Additionally, I seek to make progress in the side-view environment, where work has only recently commenced.

3 Alexandre

3.1 Tasks

- Multiplayer
- Class tree
- Level design
- Website

3.2 Achievements

As was previously stated, each member deviated a bit from their goals stated by the Book of Specifications. I mainly focused on setting the framework for the universe creation, as well as taking time to ensure my role as the spokesperson.

- Worlds
- Communications

3.2.1 Worlds

My main work on this part was to try and figure out how to organize this workflow as I started to learn Unity for the first time. I had to redo tile-maps over and over so much, that I became really efficient at creating the necessary tools to build them well.

3.2.1.1 Camera

To achieve the desired look and feel of our game, we needed a way to move the camera in response to the player's movements. We were able to accomplish this by attaching a script called "CameraFollow" to our camera, which takes three parameters: the player object, the height/offset relative to the player, and the type of view to be displayed. Currently, the view type is hardcoded into the script, but we plan to make this more dynamic in the future. One possibility is to assign the view type using a tag from either the player or the scene, allowing us to set the camera based on the game mode. This would make the camera more versatile and adaptable to different gameplay situations.

3.2.1.2 Tile-maps

To simulate the 2D perspective, we decided on using the 2D package of Unity, that allowed us to create tile-maps in a 3D project. A tile-map can be seen as a grid containing several squares of size 1*1, each of those square gets a sprite assigned, which allow us to efficiently create maps. This solution has been proven adequate for both view systems, although only a top-view maps has been produced, due to camera difficulty, and the absence of a jumping behavior on most entities. Later on, when referring to a tile-map, it will generally be a XZ-tile-map, as in, a tile-map seen from above.

Layering

To optimize the workflow, as well better graphics, I chose to use several tile-maps for each scenes. Currently a scene has two tile-maps, but the number might increase in the following iterations. They are the following:

- GroundTiles

This layer represents represent the standard environment, with which the player cannot interact, it's the ground, or background of the map. It is set with an order of layer -1 so that it's behind the "TopTiles". It contains vegetation, path, ...

- TopTiles

This layer represents the common non-walkable areas, such as houses, rocks, trees, ... This layer is also used to create colliders, but more on that later. It has on order in layer 0, so that it's displayed above the "GroundTiles". If a square is populated with a sprite in this tile-map, the player cannot move onto this square.

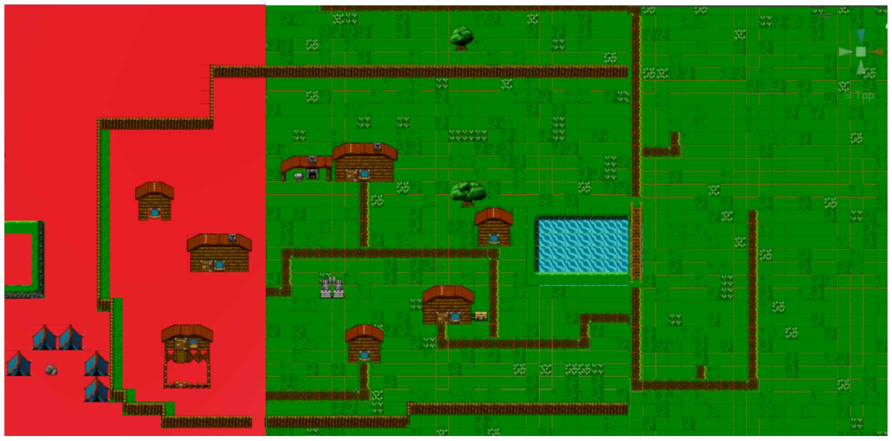


Figure 2: Tile-maps

Hitbox - Colliders

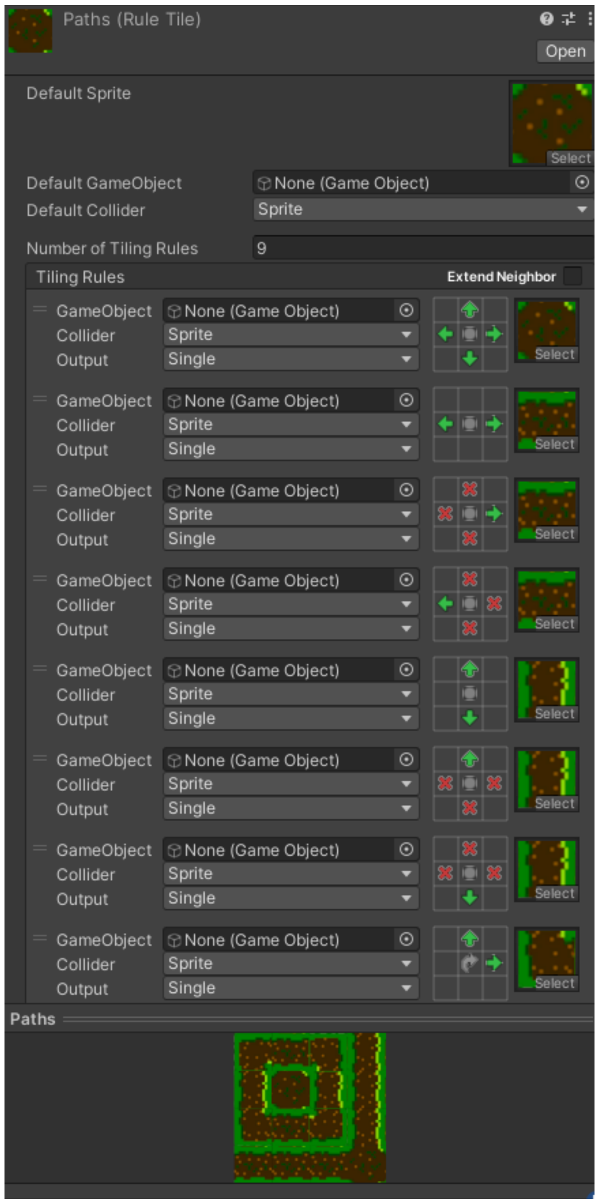
Unity 2D has a neat feature allowing one to automatically put 2D colliders on tile-maps, the "TopTiles" in our case. The main advantages is the generation of streamlined colliders and the ability to simplify them, meaning that if a tile is partially empty - the sprite has a transparent portion - the generated collider might not the entirety of the square. It also allows for simplification, replacing at least two connected colliders by one with the size of both of them. Unfortunately, this collider feature only works with 2D colliders and our players and entities currently have a 3D colliders. Since Unity doesn't allow for 2D-3D collider interaction, I had to improvise. I created a script that maps the position of non-empty tiles - an empty tile being different than a tile with an empty sprite - to the map and creates colliders at this position in the scene. This necessitated making sure that every tile-map was created and positioned with the bottom left corner on the coordinates (0,0,0). The depth or height of the collider was set to 1 so that an entity cannot "glide" over it. The position of each collider is adjusted with an offset on each axis to compensate for it's creation by the center. In addition to this collider generator, I attempted to create a simplify script, that unfortunately doesn't work. To debug the issue I had with the colliders, I also created a script that highlights hitboxes with a yellow , as can be seen in Figure 7, using Gizmo. Activating it during a run decreases the FPS by an average of 60% (100-105 to 39-42).

Tiling

As I'm no level designer and struggle to create life-like structures and environments, I decided against creating my own sprites, but rather use free downloaded tilesets that I spliced. To address environment creation I decided to use rule tiles of two types:

- I also created a special random tile, that places the vegetation for me. I weighted it using 35 sprites. 24 of those are the standard, 3 of those are special vegetation and the rest is split in two, between dense and really dense grass, allowing for more variation. By adjusting the Perlin noise feature of Unity, the result can be made slightly more coherent.

- Structures are used to facilitate path or mountain creation. At this instant the mountain rules is not the most efficient with my layer hierarchy, as it creates tiles with empty sprites where it shouldn't. In order to upgrade the path creation, I need to ensure that I have all the necessary tile sprites - from the current map, a T-path for example - , and need to create them.



Paths (Rule Tile)

Default Sprite

Default GameObject: None (Game Object)

Default Collider: Sprite

Number of Tiling Rules: 9

Tiling Rules

Extend Neighbor

GameObject: None (Game Object)
Collider: Sprite
Output: Single

GameObject: None (Game Object)
Collider: Sprite
Output: Single

GameObject: None (Game Object)
Collider: Sprite
Output: Single

GameObject: None (Game Object)
Collider: Sprite
Output: Single

GameObject: None (Game Object)
Collider: Sprite
Output: Single

GameObject: None (Game Object)
Collider: Sprite
Output: Single

GameObject: None (Game Object)
Collider: Sprite
Output: Single

GameObject: None (Game Object)
Collider: Sprite
Output: Single

Paths

Figure 3: Rule tile example

Portal

I found an animated sprite that can be used to represent portals in top-view, and maybe side-view. It currently serves no purpose as there is yet now way to differentiate it from other tiles, but I might be able to do that using a context menu.

Context Menu

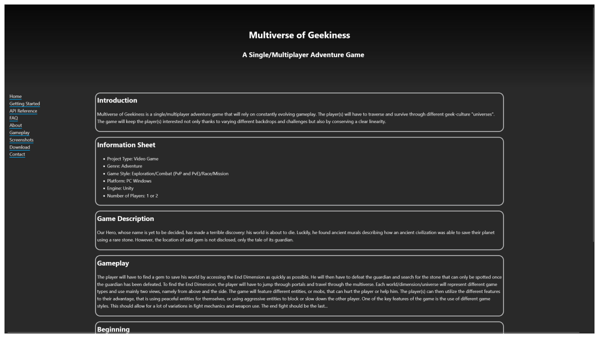
Currently, each time a scene is loaded, it also creates the colliders for each singular tile of "TopTiles", putting additional load on the CPU. Since there is no way to simplify the colliders, using a context-menu to directly and permanently create the `gameObjects` would mean having to store a lot of additional information on the game, which is clearly not ideal for users, and might even reduce their performance. This is why both of these features needs to either be developed simultaneously, or dropped completely.

3.2.2 Communication

As the spokesperson for Incroyqble, I was responsible for managing most of the group's communication efforts. This included creating and maintaining our website, writing up our report, and making sure we were well-prepared for our first presentation. With a keen attention to detail and a dedication to producing high-quality work, I worked hard to ensure that our message was both persuasive and well-received.

3.2.2.1 Website

As of now, the website is still a work in progress and has yet to be completed. While the core structure has been established, there are still many aesthetic elements that need to be refined and perfected. Drawing inspiration from my own personal website, `https://ammon-prints.netlify.app/`, which I designed and developed, I have used it as a foundation upon which to build this new site. As outlined in the Book of Specifications, the website will serve as a central hub for information about the game, including the class tree and ruleset. Additionally, users will be able to download the game directly from the site, making it easily accessible to everyone who wants to play.
It must be noted that this website doesn't exists online yet.



Multiverse of Geekiness

A Single/Multiplayer Adventure Game

Home
Getting Started
API Reference
FAQ
About
Gameplay
Screenshots
Download
Contact

Introduction

Multiverse of Geekiness is a single/multiplayer adventure game that will rely on constantly evolving gameplay. The player(s) will have to traverse and survive through different geek-culture "universes". The game will keep the player(s) interested not only thanks to varying different backdrops and challenges but also by conserving a clear linearity.

Information Sheet

- Project Type: Video Game
- Genre: Adventure
- Game Style: Exploration/Combat (PvP and PvE)/Race/Mission
- Platform: PC Windows
- Engine: Unity
- Number of Players: 1 or 2

Game Description

Our Hero, whose name is yet to be decided, has made a terrible discovery: his world is about to die. Luckily, he found ancient murals describing how an ancient civilization was able to save their planet using a rare stone. However, the location of said gem is not disclosed, only the tale of its guardian.

Gameplay

The player will have to find a gem to save his world by accessing the End Dimension as quickly as possible. He will then have to defeat the guardian and search for the stone that can only be spotted once the guardian has been defeated. To find the End Dimension, the player will have to jump through portals and travel through the multiverse. Each world/dimension/universe will represent different game types and use mainly two views, namely from above and the side. The game will feature different entities, or mobs, that can hurt the player or help him. The player(s) can then utilize the different features to their advantage, that is using peaceful entities for themselves, or using aggressive entities to block or slow down the other player. One of the key features of the game is the use of different game styles. This should allow for a lot of variations in fight mechanics and weapon use. The end fight should be the last...

Beginning

Figure 4: Website overview

3.2.2.2 Report

As part of the report's collaborative process, each member was responsible for redacting their own section, as evidenced by the use of the first-person singular pronoun "I." While each individual contributed their unique perspective and expertise, it was my role to oversee the compilation of the various sections into a cohesive whole. In addition, I had to create and organize the necessary commands for the group, such as `subsubsubsection`.

3.2.2.3 Presentation

I was in charge of the graphics for the presentation, while each member only had to add their parts.

3.3 Difficulties

One of the most significant obstacles I encountered during the initial stages of the project was the challenge of creating something truly innovative and unique. There was a distinct lack of resources and examples available that demonstrated how to effectively leverage Unity 3D to develop a game similar to ours. As such, I spent a considerable amount of time conducting in-depth research and experimenting with various approaches in order to identify the most efficient and effective solution. My initial attempts involved creating 3D objects that were designed to be viewed only from specific angles. However, I soon realized that this approach was both inefficient and would result in a sub-optimal game experience. After considerable experimentation, I discovered that tile-maps could be utilized in a 3D environment. This proved to be a breakthrough, as it significantly simplified the process of creating our game's 2D environment in a 3D context. Overall, the process of identifying and implementing the best approach was challenging, but ultimately rewarding.

3.3.1 Camera

To this point, the camera system has all the functionality needed. The top-view fills all of our criteria to create more game using it. The side-view, however requires more work as the camera seems to point slightly under the player at all times, maybe because of improper sizing.

3.3.2 Tile-maps

One major hurdle I encountered along the way was figuring out how to work with tile-maps. While they proved to be an incredibly useful tool for creating the 2D environment in our 3D game, they also presented some unique challenges. For starters, I ran into some difficulties when initially creating the tile-maps on my laptop, as my Unity package install wasn't functioning properly. Once that issue was resolved, I was confronted with three additional challenges:

- As you can see in Figure 5 some of the tiles borders do not match up. This visual bug is particularly present when the player is moving. It has not yet been seen on the 2D-Side scenes, only on 2D-Top. The issue has not been yet resolved as I'm unsure of what causes this. The main leads are either an improper sprite or the use of 2D elements in 3D. On Figure 6, the tile palette was changed, and most of the incoherence was resolved on the general map, but the "grass" texture seems to still be affected.

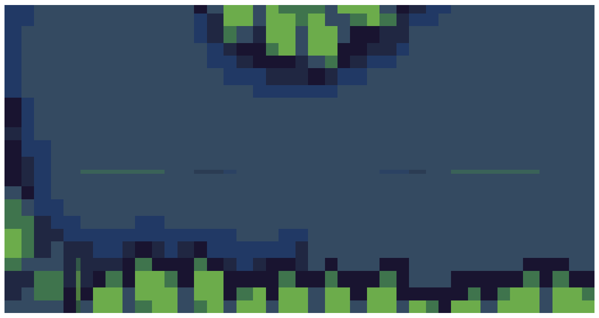


Figure 5: Example of graphical tile bug

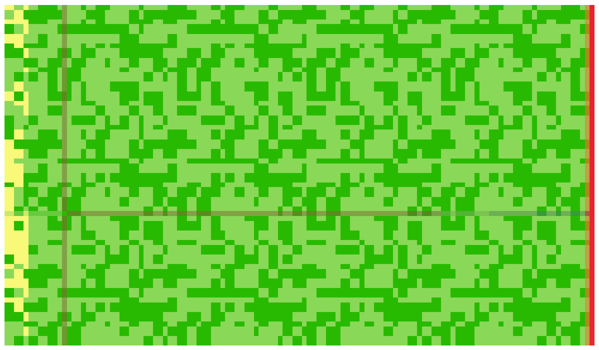


Figure 6: Example of graphical tile bug, with a different sprite

- The second issue was with the colliders generation, as explained before. Since the algorithm generates one collider per tile on a specific tile, I was worried that it would lead to decrease in performance, leading to the creation of a "SimplifyCollider" script, based on the "Composite Collider 2D" of Unity, used to simplify colliders generated on a tile-map. However my current implementation seems do more damage than good to the colliders. Since we've not yet experimented performance issues, this script remains an option but it is not currently maintained. If it were to undergo more development, it would be an idea to further tailor the colliders so that they fit perfectly to the displayed sprite, which usefulness can be seen in Figure 7.

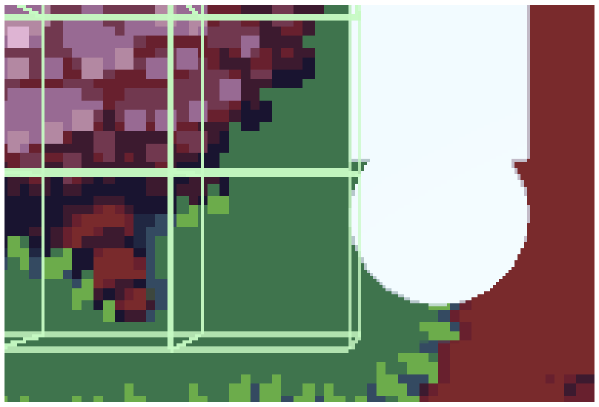


Figure 7: Non simplified colliders, where few pixels create a 1*1 hitbox.

- The last issue was with the size of the maps. We've decided that a map will be approximately 80*80 units squared. To prevent populating 6400 squares by hand, I decided to create auto-tilers. Unfortunately There is still minor problems when filling an auto-tiled area.

3.3.3 LateX

I had a lot of trouble when building the subdivision of the project, since it's much harder than in Word, where it comes pre-supported. The main hurdle was with the Table of Content that didn't display correctly.

3.4 Future Goals

For the next defense, the main goal is to complete the template for side-view, so that it's no longer just a proof of concept.

3.4.1 World building

It would also be beneficial to upgrade the world building tools by fixing any bug they currently suffer. The most pressing would be the graphical bugs that really affect the feel of the game. Next would be the honing of rule tiles, and finally trying to create a context menu that would transform 2D colliders in 3D, in order to be able to use Unity features. Once the tools have been bettered, it should be quite straight forward to create a few more maps, and to teach others how to do so if they want to create theirs.

3.4.2 Quests

If several universes environments have been completed, the next step would be the linking process, through portals. Once this has been figured out, we should be able to create the quest system, inventories, and to populate the maps with mobs.

4 Damien

4.1 Tasks

- Creation of Generic AI that aim to serve as a basis for the mobs we want to create, making zelda-themed mobs.
- Making 3D models as well as animating them for our zelda-themed mobs as well as for the players.
- Begin the creation of zelda-themed/generic weapons.

4.2 AI

4.2.1 Class Implementation

The main AI I worked on was a general zombie-like enemy script. Indeed it is meant to serve as a basis for all the mobs we want to create, and therefore allow us to focus on their special abilities (such as the Guardian's laser and beam) rather than their displacements and player detection algorithms.

This generic AI is firstly implemented through the `AI` virtual class, which is a `Monobehaviour` derived class, from which all mobs classes will inherit. This class's purpose is to create an outline for the format of the mobs as it is only constituted of virtual methods. Using a skeleton and methods common to all mobs is firstly meant to allow for better code flexibility if we decide to change how we want the AI to behave/be implemented. However this structure is also interesting as it allows us to override all methods and thus does not prevent us from making specific behaviors for some mobs if necessary.

For instance all mobs need to be derived from Unity Monobehaviour hence the `Start()` and most importantly the `Update()` event functions are implemented as virtual in the `AI` class and define a very general way for a mob to behave by:

- Updating its movement status
- Updating its attack cooldown
- Updating the player's distance and if it sees them
- Either hunts the player or moves around

for `Update()`, and by instantiating the classic attributes of a mob for `Start()`;

4.2.2 Pathfinding

The pathfinding is done using Unity's Navmesh components and Unity's Physics library. Each mob is created as a gameobject to which is attached its script as well as a Navmesh agent component. Using Navmesh allows pathfinding to be really flexible as map modifications are easily integrated by baking a new navigation mesh. This will be especially useful for us as the purpose of our game is to be an assembly of many different maps of completely different themes thus allowing for our AI to be reused in different maps. In a more general way Navmesh allows for more intelligent mobs thanks to the obstacle avoidance and shortest path computation which will make our game more enjoyable.

4.2.3 Displacements

Now:

The current generic AI uses 3 main functions to manage it's displacements:

- `RoamAround()`

First, when it does not see any player, the mob will wander around its spawn point (here the attribute `_piquet`) within a specific moving range. This is achieved by computing a random point within a circle of radius `_movingrange` and of center `_piquet` using the Physics module of Unity, and by projecting this point on the nearby navmesh. This projection is meant to allow mobs to move on uneven terrain and to avoid computing a moving point within an obstacle. The `RoamAround()` function then sets the destination of the mob to this point through its navmesh agent and changes the mob's status as "moving around" through the boolean `_roaming`.

- `UpdateMovement()`

Secondly, when the mob reaches its destination point it becomes idle for `_waitingtime` seconds which is calculated thanks to the time that passes between each frame using `Time.deltaTime`.

- `HuntPlayer()`

Finally, when the mob sees a player, it will check if the player is in attack range or if it can only be seen. Then it decides either to attack the player or set its destination to the player's position. A player in sight range will be followed smoothly by the mob as its destination will be updated each frame as long as he stays in sight range. However, if the player manages to escape and leave the sight range of the mob, the latter will come back to a random point around its spawn point `_piquet` and will start roaming around it again.

4.2.3.1 Overall - What is coming:

I started to investigate a new navigation system that relies on a mob patrolling around "waypoints". This system could be implemented via invisible gameobjects that contain a "waypoint" script and that could be precisely scattered around the maps. This system should be particularly useful for the incoming side view maps we plan to implement as well as maybe on some bosses.

Thanks to the class structure, this change could be implemented on sub-classes of mobs by overriding the `RoamAround()` method. In the same way, we might need to implement some mobs that are not limited to moving around a specific point by recomputing their destination point from their current position rather than from their `_piquet`.

4.2.4 Player Detection

The player detection is currently occurring in the `UpdatePlayersDistances()` method and is made possible by two attributes which are respectively: `_allplayers`, a static List which contains the Entity component of all the players currently on the map, and `_playersqrdistances`, a non-static float list which contains the squared distances from each of the players on the map to the current mob.

`_allplayers` and `_playersqrdistances` share the same index for the same player. Meaning, if player1's entity component is at index 0 in `_allplayers`, then player1's squared distance to a mob will be stored at index 0 in `_playersqrdistances`. `_allplayers` is currently used for damaging players as well as getting their locations, while `_playersqrdistances` is used to prevent recomputing the distances when it can be used several times per frame (when the player is in sight range, for instance). The distances are kept squared as computing the square root each frame is expensive and not useful to the mob.

The distances and players are kept in lists rather than variables as we wanted the system to work for any number of players since we might raise the number of players in the future.

In the `UpdatePlayersDistances()` method, the distances between the mob and the player are being recalculated and updated in `_playersqrdistances`. This function also updates if the player is seen by the mob or not. In order to check that, the distances are first compared to the sight range. If a player is close enough to be seen, then the mob performs a `Physics.Raycast()` which verifies if any collider is obstructing the view of the mob or not. This defines if the mob will chase the player or not. Note that each player that is in sight range but behind a collider will have its sqrdistance set to something unreachable in _playersqrdistance (by setting the distance to the sight range+1), this avoids the mobs attacking a player hidden behind a wall if the hidden player is closer to the mobs then another player which is also in sight range.

This detection system is used as it is relatively simple to use and it is less costly than other methods such as using Physics.Overlapsphere() at each frame.

4.2.4.1 Overall - What is coming:

Player detection will need to become more accurate in the future for certain types of mobs/ maps, with some mobs only detecting the players if they are directly in front of them or not detecting them at all if the players have some specific object/ability.

4.3 Mobs

I also used the generic AI basis to create two mobs that will be used in our game (in the zelda-themed map):

- A Bokoblin
- A Guardian

4.3.1 Bokoblin

The bokoblin is currently just the generic AI script that is added to a 3D model, its behaviour is as stated in the previous sub-section.

4.3.2 Guardian

The guardian can be implemented in two ways:

- An Idle Guardian
- A Moving Guardian.

For both types: When a player gets into its sight range this mob follows him with a laser for a few seconds before shooting in the direction of the player a projectile that explodes when it collides with anything.

Those mobs work thanks to the `GuardianWeapon`, which is a weapon assigned to both guardian types exclusively. This weapon work by generating a LineRenderer between the closest player and the guardian's eye and once the charging time has passed, instantiating a gameobject with a `GuardianProjectile` script.

As stated in the AI subsection, those two forms barely needed any player detection arrangement as I simply used the already existing one, the only issue being that the `Physics.Raycast()` used to check for walls had to get its starting postion adjusted as the guardian's hitbox blocked it.They however needed the full development of their weapon. It order to get it to work I had to get familiar with Unity's monobehaviour, by learning how to handle collisions, how to make transforms translate, as well getting the Linerenderer to follow the player properly and to disappear if the player got too far.

The Linerenderer script was the most complex to create as it requires constant update of the player's postion as well as of which player it needs to target.The Linerenderer script is also what instantiate the projectile. Hence it works thanks to a `_player` attribute that contains the Entity component of the nearest player which gets updated each frame in the guardian script, which allow for an easy way to access the player's position.

The `Idle Guardian` is unable to move, it acts as a form of turret and will be placed around the map as a warden.

The `Moving Guardian` acts as a moving turret. This mob is the combination of the generic AI's movements and the `GuardianWeapon` used by the idle Guardian. It wanders aimlessly around its spawnpoint and stops to target a player if one is in sightrange. This mob's attack range is the same as it's sight range hence it does not need to chase the player. The main difficulty when creating this mob was to know when it could start to move again, which was fixed thanks to an attribute of the `GuardianWeapon` that keeps track of whether the weapon is being used or not.

4.3.2.1 Overall - What is coming:

I intend to create similarly more mobs for different types of maps. I for now focused on the generic AI as well as the zelda-themed one which was my goal, and I will continue developing various mobs by using custom weapons that they will be assigned to. The next step is most likely going to be the creation of Pokemon-themed mobs.

4.4 3D Models

By far the hardest part I've had to do so far. As we decided to make our game 3D and retro-looking, I had to look up Blender and learn the basics of 3D modeling, animating and converting everything to unity. Only very few free online assets are available (at least those which match our needs for this project), especially as our game "MOG" requires assets from preexisting games such as Zelda which are not available online.

The first map we wanted being Zelda themed, two Zelda-mobs were made and animated in a cubic-voxel retro way: a Bokoblin and a Guardian (+ the player model).

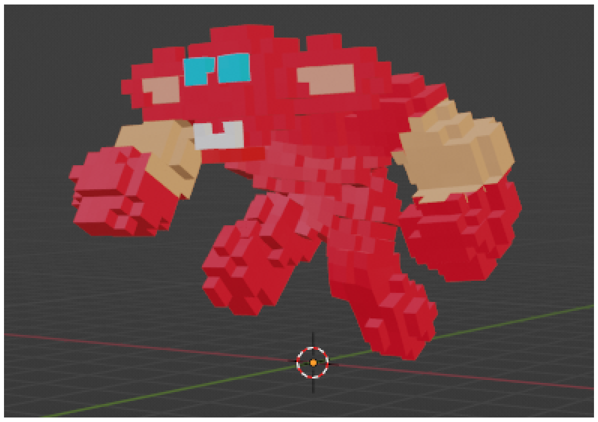


Figure 8: Bokoblin

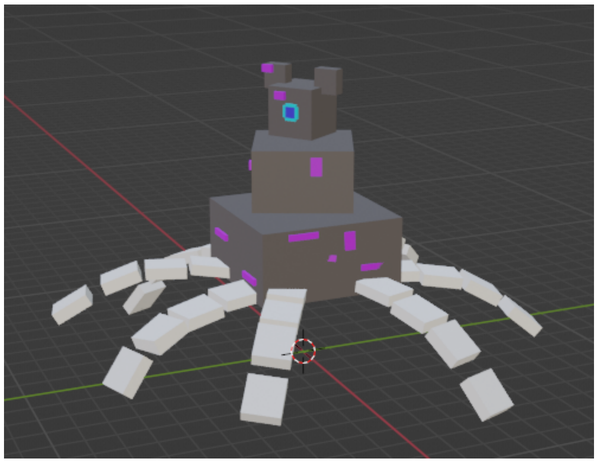


Figure 9: Guardian

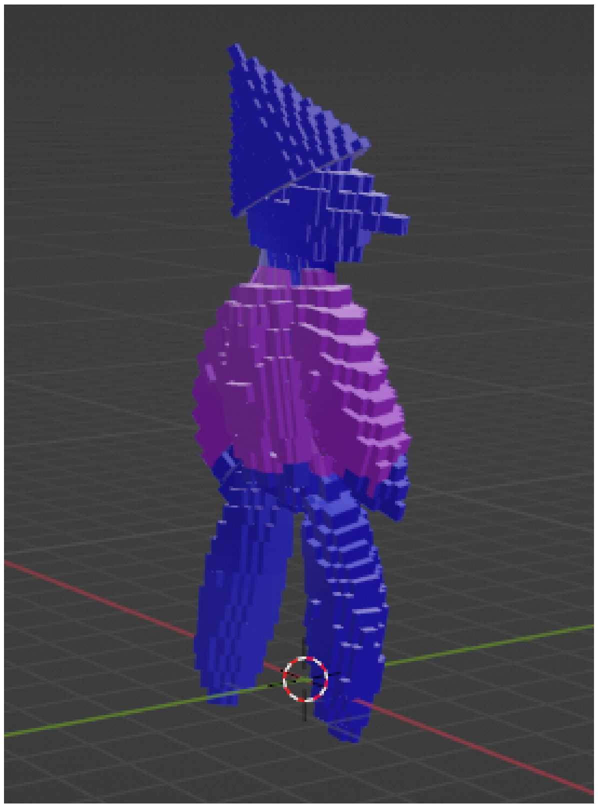


Figure 10: Player

4.4.0.1 Making the models

Making models was more complex than expected; especially for the first model I made since I had to learn how to manipulate the different meshes, modifiers and modes of Blender. To make those models I decided to make them using regular non-cubic components at first and then to convert those into cubes using a modifier. However once the model was properly sculpted a problem that I still face arose: the remesh modifier that allows you to convert a model's mesh into cubes also destroys part of the model you made before converting it which had to be fixed manually.

4.4.0.2 Animating the model

The models currently have between 2 and 3 animations:

- A walking animation
- An Idle animation
- An Attack animation - currently only for the Bokoblin

Learning how to rig (make a skeleton) and make animation was interesting as well as tedious since it first took some time to get how to correctly distribute the weight of the flesh on the bones but most importantly to make the different poses for each animation and try to make them as fluid as possible.

4.4.0.3 Transferring the models to Unity

This part was actually the most challenging as I had to fix an immense amount of bugs ranging from textures and files not being imported properly to the imported model's axis not being aligned with the other game objects' axis. The latter issue was very problematic as it meant that the Navmesh agents of the mobs could not make the models move where they were looking as their Z axis (the front axis in Unity) was pointing in the wrong direction. This Issue had to be fixed in Blender by separating the armature and the mesh of the models, and by setting their default orientation to be rotated by 90 degrees. Once the models were imported I used the Unity animator controllers in order to create triggers allowing the animations to be started through code.

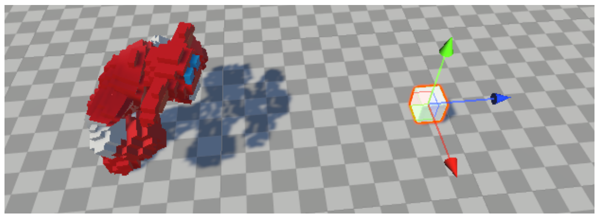


Figure 11: Regular axis

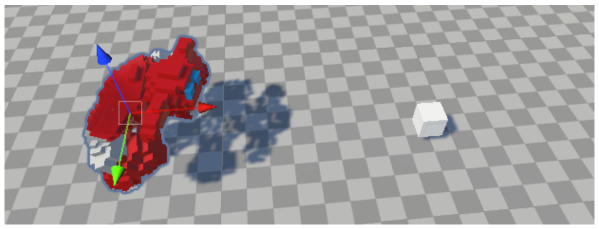


Figure 12: Bokoblin's axis are not correctly aligned

4.4.0.4 Overall - What is coming:

I feel like making those 3 models took way longer than it should have, however I now know how to fix/avoid a very large quantity of problems that I've encountered. Hence I believe I will be able to make models and animations way more quickly and efficiently now that I am familiar with creating and animating in Blender/Unity which should be very useful for the rest of this project.

4.5 Weapons

Weapons are essential for our game, like the mobs they will be themed accordingly with the different maps we want to create. For this project we use the `Weapon` class both for mobs and players. However as stated in the Mob subsection, some mob's weapons will be specific to each mob and mostly not usable by players since they will act as a "part" of the mob itself.

4.5.1 Class Implementation

The `Weapon` class is currently not a monobehaviour-derived class, indeed this choice was made we want weapons to be easily manipulated with code (for example, their easy to create, can be stored in arrays) as they are not necessarily linked to a gameobject. However weapons often do need to have a gameobject or multiple gameobject to represent them in-game, thus the class has access to a `_monoBehaviour` of Monobehaviour type that allows for an easier manipulation of gameobjects. For instance this attribute allows us to us the `StartCoroutine()` of Monobehaviour which is very useful for spreading actions through several frames.

Currently the `Weapon` class is inherited by three weapons that the players can use:

- A Sword
- A Bomb
- A Bow

4.5.2 Sword

This Item is the first that I created, it is meant to be the basic weapon of our player and is hence not themed after any particular map. uses the `Physics.OverlapBox()` method of Unity to create a damaging zone in front of the player when the weapon is selected and the player presses left-click.

4.5.2.1 Overall - What is coming:

This sword is overall a very basic implementation that was actually meant to be an experimentation more than a weapon that will stay as it currently is for the rest of the project. Truly I will replace it by something less imprecise than a damage area, the sword will most likely be linked to a sword-gameobect and will deal damage when the latter will collide with an enemy.

4.5.3 Bomb

This item is meant to be a zelda-themed bomb. Like every weapon that is not the sword, it will be earned by the player by overcoming a challenge in the map where it belongs. This weapon is made thanks to Unity's `Coroutines`. It works by instantiating a gameobject that is removed after five seconds as a `Physics.OverlapSphere()` is called in order to get all the nearby entities to damage..

4.5.3.1 Overall - What is coming:

This bomb can currently can only be used to deal damage to entities however in the future it might be interesting to give it the ability of destroying some gameobjects in order to uncover hidden game content. The bomb might also gain the ability to be thrown for the sideview maps that we want to create.

4.5.4 Bow

This item is once again zelda-themed. It will be earned on the same map as the bomb. This weapon is also made using Unity's `Coroutines`. In order for a projectile to be shot the player has to hold left click for five seconds, then a gameobject with a projectile script will be instantiated directly in front of the player and will translate forward for ten seconds or until it hits something (dealing damage if it's a player or a mob).

4.5.4.1 Overall - What is coming:

This item will probably stay as it is once it will have a 3D model as well as for the arrow.

4.6 Difficulties

In conclusion it at first was also difficult to get used to unity. However the most difficult part was learning about how to model and animate in blender as I had virtually no knowledge of any of it and it took the largest part of my time.

5 Arthur

5.1 Tasks

- Creating a modular system for multiplayer feature including : Player Management (Creation and Deletion), making AI .
- Inheritance of Entity Class, Main Loading system for maps
- Graphical Part : Creation of the HIGH DEFINITION PIPELINE for graphical rendering, making the game.
- GIT Operations : Conflicts resolution, git layout and continuous integration.

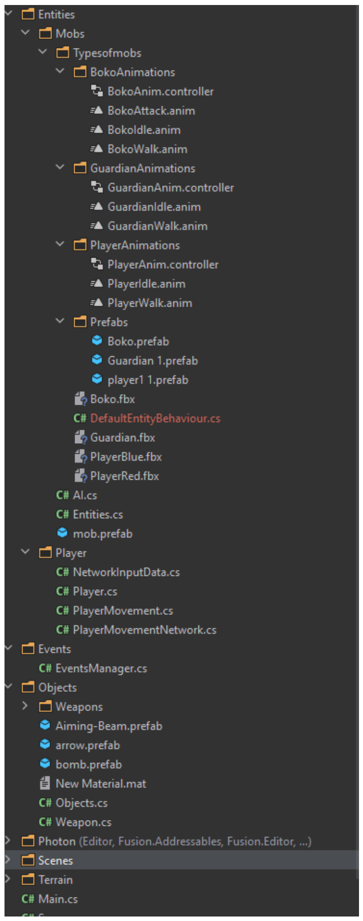
5.2 Achievements

5.2.1 Multiplayer

So far, the multiplayer feature for the game is ready to operate without entities using a peer-to-peer based connection. We're currently thinking of implementing a server-based connection, thus removing the server part. Multiplayer is based on Photon Fusion Engine that provides us the necessary tools to handle multiplayer and networking properly. The first step was to choose which type of connection we will be using. For this project, we picked peer-to-peer connection as the default connection method. This type of connection allows us to not take in account a dedicated server setup which we didn't want since we will loose precious time setting it up. Player movement was an important part to implement since we needed to make the player smoothly moving in the scene. The multiplayer consists of three classes Spawner, PlayerNetworkMovement, NetworkInputData. The Spawner class is responsible of handling the player's connection and sending Network Data directly to the server. The PlayerNetworkMovement is wrapping player input into NetworkInputData which can be sent using the Spawner Class The NetworkInputData is a class that contains a structure which describes Player movement. Photon handles the connection management with their in-cloud service that provides a relay for peer-to-peer based connections, this relay allows players that aren't in the same network to still be able to connect to a game or share one.

5.2.2 Layout

The whole project structure and layout is now completely defined with scripts for every maps and entities. For now a solid object oriented program based is implemented. We used object oriented programming in order to make the code more readable and to facilitate new implementations. Scripts are organized into folders according to their purpose, as shown in the image below.



```
Entities
  Mobs
    Typesofmobs
      BokoAnimations
        BokoAnim.controller
        BokoAttack.anim
        BokoIdle.anim
        BokoWalk.anim
      GuardianAnimations
        GuardianAnim.controller
        GuardianIdle.anim
        GuardianWalk.anim
      PlayerAnimations
        PlayerAnim.controller
        PlayerIdle.anim
        PlayerWalk.anim
      Prefabs
        Boko.prefab
        Guardian 1.prefab
        player1 1.prefab
      Boko.fbx
      DefaultEntityBehaviour.cs
      Guardian.fbx
      PlayerBlue.fbx
      PlayerRed.fbx
    AI.cs
    Entities.cs
    mob.prefab
  Player
    NetworkInputData.cs
    Player.cs
    PlayerMovement.cs
    PlayerMovementNetwork.cs
Events
  EventsManager.cs
Objects
  Weapons
  Aiming-Beam.prefab
  arrow.prefab
  bomb.prefab
  New Material.mat
  Objects.cs
  Weapon.cs
Photon (Editor, Fusion.Addressables, Fusion.Editor, ...)
Scenes
Terrain
Main.cs
```

Figure 13: Project layout

5.2.3 Git

Regarding Git operations, I'm currently in charge of merging and reviewing procedures to ensure that every update done on the whole project doesn't conflict with another. I'm also implementing a semi-automated procedure to build and test the game with auto rejection process. Sourcetree has been a very useful tool for conflict resolution and as a git viewing tool. Using this software we've been able to resolve complicated merging problems, thus reducing time spent on this matter. So far there isn't any major issue regarding the use of Git in our use of it. As of this first defence, the game is currently rendering every texture with a high definition render pipeline (HRP). Currently this feature is not very useful since we're rendering simple 3D objects. However in the future, we hope to be able to render high quality textures while still being able to render low-quality objects. We hope in the future to extend the use of the HRP in different part of the game.

5.3 Difficulties

The main difficulty was to find a project layout that was pleasant to work with and to get everyone's agreement about this architecture. After that, we had to set up everything we needed to make our GIT repository work. Permissions, git ignore, were very important to make a coherent game layout, avoiding delays due to mistakes and typo errors. A considerable portion of our time was dedicated to figuring out how to manipulate github and its large number of features to get the result we wanted.

Another difficulty was to implement the High Definition rendering pipeline. Finding document on it was rather complicated and some objects such as the first grass model we found were not compatible with it. Therefore, we had to start from scratch and lose precious time in the process. However, once we made our first 3D model with our own customized settings, things started to become much more

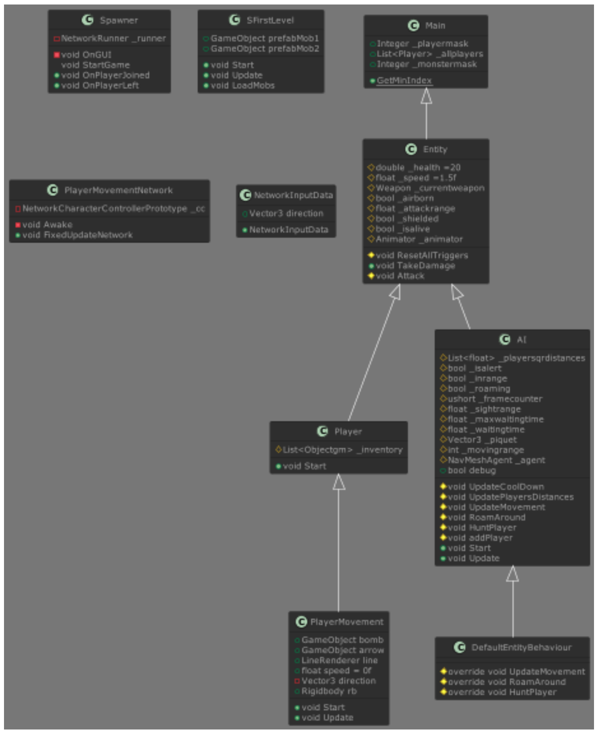
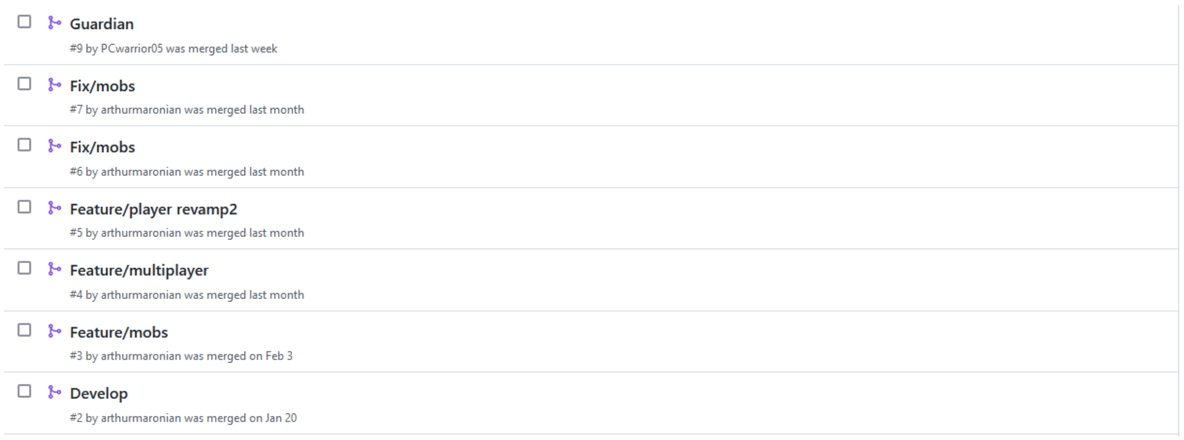


Figure 14: Project's UML

easier for us.

Regarding AI, a recurring bug was that if the game loaded too quickly or in a multiplayer mode, AIs won't be able to detect players, thus making mobs in a simple roaming situation with no attack being done. A rapid fix was changing how AI detect and "register" players in its list. Rather than AI listing every player. At each new connection, the player registers itself to the AI with an indication of it's current position. The AI just tracks it from there without the risk of losing it. However due to multiplayer complication we weren't able to make the AI follow the players distance in some cases. THis is why AI can't detect players in multiplayer for now.



Guardian
#9 by PCwarrior05 was merged last week

Fix/mobs
#7 by arthurmaronian was merged last month

Fix/mobs
#6 by arthurmaronian was merged last month

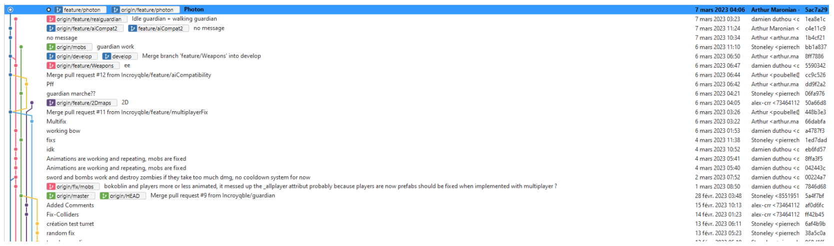
Feature/player revamp2
#5 by arthurmaronian was merged last month

Feature/multiplayer
#4 by arthurmaronian was merged last month

Feature/mobs
#3 by arthurmaronian was merged on Feb 3

Develop
#2 by arthurmaronian was merged on Jan 20

Figure 15: Example of organized pull requests



feature/photon origin/feature/photon **Photon**	**7 mars 2023 04:06**	**Arthur Maronian ·**	**5ac7a29**
origin/feature/realguardian Idle guardian + walking guardian	7 mars 2023 03:23	damien duthou <c	1ea8e1c
origin/feature/aiCompat2 feature/aiCompat2 no message	7 mars 2023 11:24	Arthur Maronian <	c4e11c9
no message	7 mars 2023 10:34	Arthur <arthur.ma	1b4cf21
origin/mobs guardian work	6 mars 2023 11:10	Stoneley <pierrech	bb1a837
origin/develop develop Merge branch 'feature/Weapons' into develop	6 mars 2023 06:50	Arthur <arthur.ma	8ff7886
origin/feature/Weapons ee	6 mars 2023 06:47	damien duthou <c	5590342
Merge pull request #12 from Incroyqble/feature/aiCompatibility	6 mars 2023 06:44	Arthur <poubelle@	cc9c526
Pff	6 mars 2023 06:42	Arthur <arthur.ma	dd9f2a2
guardian marche??	6 mars 2023 04:21	Stoneley <pierrech	06fa976
origin/feature/2Dmaps 2D	6 mars 2023 04:05	alex-crr <73464112	50a66d8
Merge pull request #11 from Incroyqble/feature/multiplayerFix	6 mars 2023 03:26	Arthur <poubelle@	448b3e3
Multifix	6 mars 2023 03:22	Arthur <arthur.ma	66dabfa
working bow	6 mars 2023 01:53	damien duthou <c	a4787f3
fixs	4 mars 2023 11:38	Stoneley <pierrech	1ed7dad
idk	4 mars 2023 10:52	damien duthou <c	eb6fd57
Animations are working and repeating, mobs are fixed	4 mars 2023 05:41	damien duthou <c	8ffa3f5
Animations are working and repeating, mobs are fixed	4 mars 2023 05:40	damien duthou <c	042443c
sword and bombs work and destroy zombies if they take too much dmg, no cooldown system for now	2 mars 2023 07:52	damien duthou <c	00224a7
origin/fix/mobs bokoblin and players more or less animated, it messed up the _allplayer attribut probably because players are now prefabs should be fixed when implemented with multiplayer ?	1 mars 2023 08:50	damien duthou <c	7846d68
origin/master origin/HEAD Merge pull request #9 from Incroyqble/guardian	28 févr. 2023 03:48	Stoneley <8551951	5a4f7bf
Added Comments	15 févr. 2023 10:13	alex-crr <73464112	af0d6fc
Fix-Colliders	14 févr. 2023 01:23	alex-crr <73464112	ff42b45
création test turret	13 févr. 2023 06:11	Stoneley <pierrech	6af4b9b
random fix	13 févr. 2023 05:23	Stoneley <pierrech	38a5c0a

Figure 16: Sourcetree main page

5.4 Future Goals

Future goal is to finally make AI compatible with multiplayer by using NetworkObject component of Photon. We also need to discuss which kind of data do we need to serialize on the server (such as Players and Mobs Health). During the next defense we will present the multiplayer fully implemented. The next future task I'm looking into is the quest system which will guide the player throughout his journey in the game. I'm also considering refactoring some of the classes related to the player movement in order to make it more readable. However, making a loading system for each level, multiplayer-ready, is the main priority at the moment since most of our game's systems need to load flawlessly.

6 Conclusion

6.1 Progress

We, as a group find the advancements of the project to be somewhat in track with was first decided. The Book of Specifications mentioned that the first defense would mainly be about building a good work environment, and to give us all the necessary tools to go further. Damien, for example taught himself how to create 3D models more efficiently, as well as creating AI entities, and their different weapons, while Alexandre explored the possibilities of tile-maps. Arthur decided that the network implementation was much needed in the beginning, which turned out to be true: the lack of compatibility with mobs and multiplayer will be much more difficult to solve once the project is more advanced.

6.2 Team

For all of us, this was the first time working on a long coding project with others. Our main mistake was to work individually on features, with minimal feedback and communication between us, leading to uneven workload being achieved, and merging issues. This struggle will be solved by the implementation of the "agile method" work ethic during further development. This will allow us to create a better medium for communication, help requests, and project architecture as well as deadlines.

6.3 Next steps

For the first defense, we hope to be able to present a more cohesive implementation of our project, as well as something that can already be played. This mainly focuses on team work and time management. Most of the universes with their feature should be achieved by then.

7 Annex - outline

7.1 Introduction

7.1.0.1 Our game

7.1.0.2 Our Progress

7.2 Member presentations

7.2.0.1 Damien

7.2.0.2 Pierre

7.2.0.3 Alexandre

7.2.0.4 Arthur

7.3 Outroduction

7.3.0.1 Expectations

7.3.0.2 Improvements - next steps