

Léo FARHI (Chef du projet)  
Hugo RUBIO  
Geoffrey ELBAZ  
Léo ALBA SANCHEZ

## **Rapport de soutenance n°3**

Regain The World !

**RELIK**

28 Juin 2021



## ATTENTION !

Ces informations sont très importantes à prendre en compte.

- Notre jeu nécessite beaucoup de puissance pour être lancé, il est donc important de posséder un processeur suffisamment puissant ou d'avoir une bonne carte graphique ainsi que d'avoir suffisamment de mémoire RAM. Malgré les optimisations effectuées pour alléger l'ordinateur, il est tout de même nécessaire d'avoir ce matériel pour profiter pleinement du jeu afin qu'il soit fluide.
- Si vous voulez tester le mode en ligne, il est important d'avoir une bonne connexion. En effet, les zones comme les villes demandent à la machine d'envoyer et de recevoir beaucoup de données à cause de la quantité des PNJ.

Merci pour votre attention, bon divertissement

# Sommaire

<b>Introduction</b>	<b>5</b>
Rappel du projet	5
But du projet	5
Nom du Groupe	5
Logos de RELIK et Regain The World	6
<b>Evolution du scénario et des buts</b>	<b>7</b>
<b>Fonctionnalité du jeu</b>	<b>8</b>
Mode Singleplayer et Multiplayer	8
Une sauvegarde un peu particulière	9
Récapitulatif de ce qui a été convenu et ce qui a été modifié	9
<b>Les Techniques et Méthodes utilisées</b>	<b>10</b>
Les logiciels utilisés pour le projet	10
Communication entre les membres du groupe	10
Croquis	10
Moteur de Jeu	10
Synchronisation des fichiers entre les différents membres	11
Modélisation des personnages et des objets	11
Animations et Cinématiques	11
Musiques et Sons FX	12
<b>Aspects Économique et Opérationnel</b>	<b>12</b>
<b>Découpage des tâches à faire par membre</b>	<b>13</b>
Tableau de répartition des tâches	13
<b>Chronologie des Tâches</b>	<b>14</b>
<b>Présentation générale du projet et des tâches individuelles</b>	<b>15</b>
Avancement général du projet	15
Présentation du planning de réalisation	15
Level design	16
Shader, Texture et Post Processing	23
Gestion des lumières	23
Système jour/nuit	24
Générateur de PNJ avec tenue semi-aléatoire	24
Système d'intelligence artificielle	27
Evolution des IA au cours du temps :	29
Première soutenance :	29
Seconde soutenance :	30
Troisième soutenance :	33

Les boss	33
Les dernières fonctionnalités des IA	33
Les animations	34
Musiques	35
Network	35
Optimisation	38
Modèles 3D	40
Outils extérieurs	43
Jaquette	43
Livret d'utilisation	43
Autres fonctionnalités	44
Système de combat	44
Les Menus	44
Menu principal	45
Menu de sélection des personnages	45
L'inventaire	47
Interfaces	47
Dialogues	48
Le Site Web	50
Problèmes rencontrés	50
Entente du groupe	50
Réalisation	50
Bugs et solutions	51
<b>Récit de la réalisation</b>	<b>52</b>
Ses joies	52
Ses peines	52
<b>Conclusion</b>	<b>53</b>
L'objet de l'étude a-t-il été accompli ?	53
Ce que nous a appris ce projet	53
<b>Annexes</b>	<b>55</b>
Images du jeu (Version Éditeur)	55
Images du jeu (Version Compilée)	63
Les Croquis et Prototype	66
Site Web	82

## I. Introduction

- **Rappel du projet**

Cette partie est un rappel de ce qu'est "Regain The World".

Le groupe RELIK a été créé le 2 décembre 2020, il est composé de 4 personnes (Léo FARHI , Hugo RUBIO , Léo ALBA SANCHEZ et Geoffrey ELBAZ) très intéressées par la création et le fonctionnement d'un jeu vidéo.

Ce jeu est nommé Regain the world et sera un jeu d'Aventures et Open World du genre Heroïc fantasy, jouable en solo et en multi, où le but de nos personnages sera de regagner leur monde (dit réel). Le type, le style graphique, l'environnement, les mécaniques, le scénario du jeu sont inspirés de beaucoup de jeux et d'univers fantastiques ( Zelda, Genshin Impact, les premiers Tomb Raider, ...)

- **But du projet**

Le but premier de ce projet est de pouvoir être capable de se coordonner avec les autres membres du groupe afin de réussir un objectif commun qui est la réalisation de ce jeu. Ce dernier va aussi nous donner une raison de faire quelque chose de qualité dans un temps réduit et de proposer un maximum de Gameplay dans ce jeu.

Travailler notre organisation, faire un projet de façon autonome, faire confiance aux autres membres du groupe, tout cela fait partie des objectifs du projet.

Travailler de façon organisée, c'est certainement l'un des objectifs les plus importants du projet. En effet, le temps a été notre plus grand ennemi. C'est pourquoi un travail organisé a été nécessaire.

Enfin faire confiance aux autres membres du groupe et ne pas tout faire tout seul. Ceci fut un point très important.

- **Nom du Groupe**

Le nom du groupe "RELIK" est une reprise du mot "relique" que nous avons transformé. Nous avons choisi ce mot, car le joueur devra chercher un certain élément dans le jeu comparable à une relique.

- Logos de RELIK et Regain The World

Pour le logo de RELIK, c'est Léo Farhi qui l'a fait. Au début, nous étions partis sur un style manuscrit puis finalement nous avons opté pour un style qui se rapproche plus des jeux vidéo.



Pour le logo de Regain The World, c'est le frère de Léo Farhi qui l'a fait à la main, puis Léo l'a retouché ensuite sur ordinateur. Il représente la tête (un masque) du "méchant" de notre histoire.



## **II. Evolution du scénario et des buts**

Le scénario a été changé par rapport au cahier des charges.

Le jeu commence par un plan sur une prison installée sur une île au beau milieu de l'océan. Puis, nous voyons nos quatre protagonistes enfermés derrière des barreaux. Ensuite, l'un d'entre eux se fait emmener par un garde, mais celui-ci l'assomme et les personnages tentent de s'évader. En cherchant une issue, nos quatre personnages tombent face à un laboratoire et sont contraints d'y entrer pour se cacher des gardes. Une fois à l'intérieur, ils aperçoivent des scientifiques qui sont en train de faire une expérience. Les scientifiques s'apprêtent à créer un portail dimensionnel, nos personnages cachés se font finalement repérer et sont poursuivis dans les couloirs. Mais ils savent qu'ils seront condamnés à rester en cellule s'ils se font attraper. Par conséquent, ils tentent le tout pour le tout et empruntent le portail ouvert par les scientifiques.

Du coup, nos protagonistes se retrouvent dans une autre dimension et sont au beau milieu d'une jungle. Et nos personnages vont essayer de rentrer chez eux sans retourner en prison évidemment. Mais, pas de chance, un être maléfique nommé Giovano a vu le portail du haut de sa tour et a compris ce que c'était et va tout faire pour accéder à ce portail. Nos protagonistes apprennent que pour rentrer chez eux, ils doivent récupérer une relique, mais Giovano va tenter de la subtiliser avant eux. C'est ce personnage qui est représenté sur le logo du jeu. 

Cette relique est cachée dans un temple et nos personnages doivent résoudre des énigmes pour arriver à la fin de celui-ci. Mais en arrivant à la dernière salle, ils se rendent compte que Giovano les a devancés en envoyant un de ses sbires. Un combat se déclenche contre lui. Une fois que nos personnages l'ont battu, ils se mettent en route pour rendre une visite à Giovano afin de récupérer cette relique, avant qu'il l'utilise dans le but d'exécuter ses plans diaboliques. Nos personnages traversent plusieurs paysages avant d'arriver au pied des gigantesques tours de Giovano. Nos personnages rentrent à l'intérieur. Le même sbire vu dans le temple réapparaît pour prendre sa revanche, mais nos personnages sont trop forts pour lui et le vainquent à nouveau. Puis arrive le combat final et nos personnages parviennent encore une fois à battre leur ennemi et peuvent rentrer chez eux.

### **III. Fonctionnalité du jeu**

- Mode Singleplayer et Multiplayer

Le jeu est donc jouable en solo et en multi, et là vous vous posez la question : Comment allons-nous faire pour adapter l'histoire s'il manque des joueurs ?

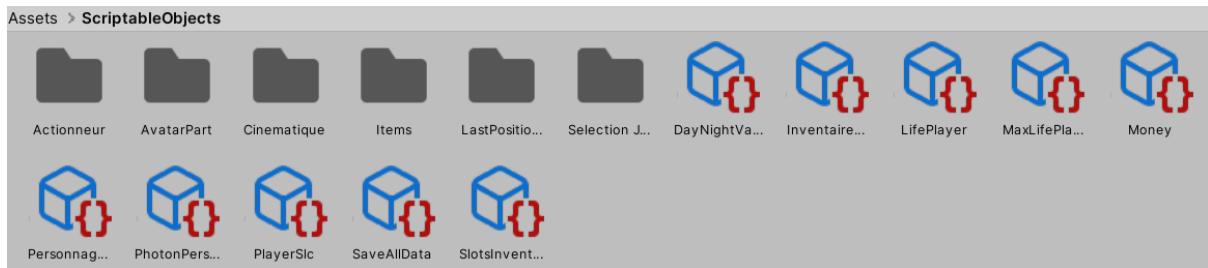
Tout simplement, en remplaçant les joueurs manquants par une intelligence artificielle qui suivra le joueur principal et devra simuler les actions d'une véritable personne.

Pour ce faire, on possède deux versions de chaque joueur. Une version IA et une version Contrôlable par l'utilisateur. Quand on charge une partie, l'ordinateur vérifie combien de joueurs sont présents dans la partie, puis en fonction du nombre, il complète avec des IA. Le joueur peut aussi changer à tout moment de personnage tout au long de la partie à l'exception de certains endroits, comme quand il est proche de certaines portes. En effet, la porte de téléportation invisible (voir dans Optimisation pour comprendre) empêche la possibilité de changer de personnage, car il y a de fortes chances que cela provoque des bugs.

- Une sauvegarde un peu particulière

On aura aussi la possibilité de créer différentes sauvegardes pour différencier les parties en multi de celles en solo. On peut aussi recommencer le jeu tout en conservant ses anciennes parties.

Le système de sauvegarde est basé sur les ScriptableObject, c'est l'essence même du jeu. Les ScriptableObject permettent de sauvegarder les variables dans des fichiers indépendants, ce qui rend leur utilisation beaucoup plus simple :



- Récapitulatif de ce qui a été convenu et ce qui a été modifié

Comme convenu, le jeu pourra se faire en solo ou en mode multi. Le fait de remplacer les joueurs non présents en mode multi par des IA en mode solo fonctionne plutôt bien et on a même la possibilité de pouvoir échanger des joueurs en mode solo.

En effet, afin de rendre le jeu plus dynamique, le joueur aura la possibilité de jouer une IA. De ce fait, l'IA qui s'est fait remplacer devient le joueur et le personnage qui incarnait le joueur avant l'échange devient une IA.

Mais alors comment cela fonctionne dans le mode multi puisqu'il y a plusieurs joueurs ?

Pour changer de personnage, la machine supprime l'IA et fait apparaître la version joueur de même apparence à sa place, puis supprime l'ancien joueur et le remplace par une IA de même apparence.

De même, il est possible de différencier les parties dans le mode solo et le mode multi. On peut donc sauvegarder plusieurs parties en solo ou en multi !

Par contre, il n'y aura pas de mode expert ou facile comme il était convenu au départ. En effet, le manque de temps que nous avons subi en raison de plusieurs facteurs ne nous permet pas de faire ces deux modes.

Par conséquent, il n'y aura pas de bonus comme une tour de combat ou un mode "mort subite" pour les mêmes raisons qu'il n'y a pas de mode expert.

## **IV. Les Techniques et Méthodes utilisées**

- Les logiciels utilisés pour le projet

Pour mener à bien ce projet il faut une bonne organisation. Par conséquent, nous avons trouvé des logiciels qui permettent de réaliser certaines actions clés au développement du projet.

De façon plus précise, nous avons séparé les différentes tâches en plusieurs catégories afin que nous puissions mieux nous y retrouver et surtout de bien pouvoir s'organiser à l'avenir.

Nous avons donc classé les logiciels que nous utilisons selon leur utilité :

- Communication entre les membres du groupe

A cause de la situation sanitaire, nous ne pouvions pas nous voir fréquemment. L'utilisation de logiciels de communication fut donc essentielle. Nous devions donc créer un moyen efficace pour pouvoir communiquer entre nous, que ce soit pour faire des réunions de mise à jour, de partage d'idées et de partage de documents.

Ainsi, nous avons créé un serveur Discord pour pouvoir discuter et faire des réunions vocales sur l'avancée du projet.

Nous utilisons également un Drive pour la communication des différents fichiers extérieurs au jeu comme le cahier des charges, des exemples de chara-design et plein d'autres choses.

De surcroît, ces logiciels de communication sont gratuits et très faciles à manipuler. Nous les avons utilisés constamment pour être mieux organisés. L'avantage de cette méthode est qu'elle nous a fait gagner un temps précieux.

- Croquis

Pour éviter les pertes de temps inutiles nous avons décidé de faire des croquis avant de passer à la modélisation.

- Moteur de Jeu

Pour le moteur de jeu, nous utiliserons la version 2019 d'Unity. Pourquoi cette version et pas la dernière, tout simplement afin que l'ensemble du groupe ait la même version d'Unity et pour des raisons de compatibilité avec les plugins et les Shader.

- Synchronisation des fichiers entre les différents membres

Pour créer un jeu à plusieurs, il nous faut un moyen de synchroniser le travail avec l'ensemble du groupe.

Ainsi, pour synchroniser les fichiers du projet sur chaque ordinateur des membres du groupe, nous avons utilisé Github Desktop et nous avons fixé quelques règles pour éviter de corrompre des fichiers et ainsi empêcher de perdre un temps précieux. De plus, il nous était possible de reprendre une ancienne sauvegarde en cas d'extrême nécessité.

Le fait d'utiliser Github Desktop est un énorme plus pour le projet. C'est un logiciel où il faut faire attention de ne pas provoquer d'erreurs quand on ne sait pas l'utiliser, mais dans le cas contraire, ce logiciel devient LE logiciel dont on ne peut plus se passer. Par ailleurs, Github sera également utilisé pour aider à la création du site internet.

- Modélisation des personnages et des objets

Tout jeu vidéo doit avoir un visuel et il faut donc un travail immense pour modéliser en 3D les personnages et objets qui constituent un jeu.

Nous avons décidé d'utiliser Blender pour modéliser les modèles 3D, les personnages principaux et les objets qui seront présents dans le jeu comme les armes, les habits, les ennemis etc.

Les bâtiments et le monde seront faits avec les outils déjà intégrés dans Unity comme Pro Builder.

Pour modéliser rapidement les PNJ, on a utilisé Vroid, afin de gagner un maximum de temps, sinon cela serait beaucoup trop long.

- Animations et Cinématiques

Pour les animations, nous les avons finalement importées depuis internet.

Quant aux cinématiques, elles ont été faites directement dans Unity, grâce à la TimeLine.

Nous n'avons finalement pas utilisé la Kinect pour faire les animations par faute de temps.

- Musiques et Sons FX

Nous savons à quel point l'audio est important dans les jeux vidéo en général. Un jeu vidéo sans audio n'est seulement qu'à moitié fini. Certains jeux comme Doom vont même axer leur particularité sur l'audio en demandant à des professionnels de se surpasser dans leur domaine.

C'est pourquoi, nous avons fait un effort sur la qualité musicale dans notre jeu. Cette qualité s'est exprimée par une composition intéressante mélangeant différents genres comme par exemple : classique, jazz, électro, dans un orchestre ou non, et par une qualité du son qui sera permise grâce à du matériel professionnel mis à notre disposition par Geoffrey Elbaz.

Dans la pratique nous avons utilisé, pour le logiciel de musique assistée par ordinateur, soit Studion One, soit Logic Pro. Ces deux derniers étant assez équivalents, le choix reste à faire. Ce type de logiciel nous a permis d'enregistrer et modifier de l'audio. Cela nous a été très utile lors de la création des bruitages du jeu pour les musiques qui ne disposent pas d'orchestre. Nous avons utilisé le format MIDI (Musical Instrument Digital Interface) au final. C'est un protocole de communication utilisé pour le dialogue entre certains instruments, séquenceur contrôleur et le logiciel. Dans la pratique, c'est ce qui permet de faire jouer n'importe quel instrument par ordinateur muni d'un clavier.

## V. Aspects Économique et Opérationnel

Comme notre jeu est à but non lucratif, nous ne percevons aucun bénéfice.

Par conséquent, nous avons dû limiter nos dépenses. Malgré tout, nous avons travaillé avec certains logiciels payants comme ceux de la famille Adobe ou encore Visual Studio.

Nous avons aussi payé les CD-ROM ou la clé USB, la jaquette du jeu que nous aurons à rendre lors de la dernière soutenance, le papier, l'impression et la reliure du cahier des charges.

Pour modéliser les objets en 3D, rien de mieux que Blender et Pro Builder. Les logiciels sont gratuits. Il y a de plus une quantité phénoménale de tutos pour apprendre à utiliser ces logiciels. L'utilisation de Vroid est un plus puisqu'il nous permettra de créer des personnages de façon très rapide, chose qui est plus difficile à faire sur Blender.

Nous avons donc plein de logiciels gratuits qui nous ont permis d'avancer à grands pas sans avoir besoin de débourser des sommes faramineuses.

## VI. Découpage des tâches à faire par membre

### Tableau de répartition des tâches

Voici la répartition des tâches décidée au début du projet. Ces dernières ont été choisies de façon à rendre le travail plus facile et surtout de les découper selon leur utilité dans le projet.

		Membres du Groupe	
		Membre Principal	Membre(s) Secondaire(s)
Tâches à Faire	Models 3D	Hugo	Geoffrey, Léo Farhi
	Cinématiques	Léo Farhi	Léo ALBA
	Level Design	Léo Farhi	Léo Alba, Hugo
	Menus	Léo Farhi	Léo Alba
	Dialogues	Léo Farhi	Léo Alba
	Musiques	Geoffrey	Hugo, Léo Alba
	Placer les musiques	Léo Farhi	Geoffrey
	Sons FX	Geoffrey	Hugo, Léo Alba
	Gestions des Lumières	Hugo	Léo Farhi
	PlayerController	Léo Farhi	Hugo
	Animations	Hugo	Geoffrey, Léo Farhi
	Système de Sauvegarde	Léo Farhi (a été fait seul)	
	Système Multijoueur	Léo Farhi	Geoffrey
	Ennemis et PNJ	Hugo	Léo Farhi, Léo Alba
	Système de sous-titre et sélection de langue	Léo Farhi (a été fait seul)	
	Système de Volume	Léo Farhi	Geoffrey, Léo Alba
	Croquis	Tout le monde	Tout le monde
	Système de communication entre les membres	Tout le monde	Tout le monde

Globalement, cette répartition a été bien respectée par l'ensemble des membres du groupe. Il y a eu néanmoins quelques changements comme le level design davantage traité par Léo Alba, et les animations ont été faites à la fois par Léo Farhi et Hugo.

## VII. Chronologie des Tâches

Date	Membre du groupe	Tâches
4 décembre 2020	Léo F	Création du projet
13 décembre 2020	Léo F	Ajout du système de dialogue et de sauvegarde
18 décembre 2020	Léo A	Création de la prison extérieure
	Geoffrey	Création du site web
21 décembre 2020	Léo F	Ajout du système multijoueur Photon
26 décembre 2020	Léo F	Ajout du système de sous-titre
28 décembre 2020	Hugo	Création du masque du méchant
2 janvier 2021	Léo F	Création du logo du jeu
3 janvier 2021	Léo A	Création de la prison intérieure
4 janvier 2021	Léo F	Création du système de téléportation invisible
29 janvier 2021	Léo F	Création du "Système Value" avec les ScriptableObject
	Geoffrey	Mise à jour du network
6 février 2021	Hugo	Création du labo
22 février 2021	Hugo	Création du personnages "Giovano"
27 février 2021	Léo F	Mise à jour des shaders
1 mars 2021	Hugo	Début des IA
2 mars 2021	Léo A	Début de la création de la jungle
3 mars 2021	Tout le monde	Première soutenance
11 mars 2021	Léo F	Création du système de générateur de PNJ
19 avril 2021	Hugo	Création des personnages principaux
	Léo F	Création du prototype de la première Ville
24 avril 2021	Léo F	Mise à jour importante du network
25 avril 2021	Hugo	Mise à jour importante de la première ville
	Geoffrey	Mise à jour du site Web
27 avril 2021	Geoffrey	Ajout des musiques
28 avril 2021	Léo F	Mise à jour importante du système de sauvegarde
29 avril 2021	Hugo	Mise à jour importante du système d'IA
	Léo F	Création du système d'inventaire
30 avril 2021	Tout le monde	Soutenance intermédiaire
4 mai 2021	Hugo	Mise à jour importante des IA et des villes
28 mai	Léo F	Création de toutes les cinématiques
14 juin 2021	Léo F	Création de la jaquette
	Hugo	Création des boss
28 juin 2021	Tout le monde	Soutenance finale

## VIII. Présentation générale du projet et des tâches individuelles

### Avancement général du projet

- Présentation du planning de réalisation

Nous avons mis à jour le tableau de l'avancée des tâches. Celui-ci a varié par rapport à celui du cahier des charges et du dernier rapport.

Tâches à Faire	Déjà fait	Avancement par Soutenance (en %)		
		Soutenance 1		Soutenance 2
				Soutenance 3 <b>(Stade Actuel)</b>
	Models 3D	40	100	100
	Cinématiques	0	0	100
	Level Design	40	90	100
	Menus	95	95	100
	Dialogues	30	40	100
	Musiques	50	100	100
	Placer les musiques	0	80	100
	Placer les sons FX	5	70	100
	Gestion des Lumières	75	95	100
	PlayerController	60	100	100
	Animations	30	95	100
	Système de Sauvegarde	100	100	100
	Système Multijoueur	100	100	100
	Ennemis et PNJ	40	90	100
	Système de sous-titres et sélection de langues	100	100	100
	Système de Volume	100	100	100
	Système de combat	30	90	100
	Système de communication	100	100	100

- Level design
  - Les différentes zones du jeu

Un jeu de type vidéoludique est aujourd’hui apprécié pour sa complexité, son scénario ou sa beauté. Le level design est alors une partie importante du jeu, car c'est en partie elle qui lui permet de prendre vie et de choisir sa forme.

Nous voulons un jeu attractif qui donne envie de jouer, c'est pour cela qu'il fallait que le nôtre soit “beau” et c'est là que cette partie prend toute l'importance qu'on lui consacre. Car notre jeu doit taper dans l'œil du joueur et lui donner envie d'y jouer. Pour cela, il fallait créer plusieurs “régions” ou “lieux” attractifs et tous différents tout en gardant une certaine logique pour la suite de l'histoire.

Nous avons donc commencé par créer la première zone du jeu qui n'est autre que la prison dont l'intérieur a été réalisé par Léo Alba et l'extérieur par Léo Farhi. Pour faire la prison extérieure, nous nous sommes inspirés de la prison d'Alcatraz pour créer la forme de l'île ainsi que le placement de certains éléments. Cette zone sert à la fois de tutoriel pour jouer (le joueur va se familiariser avec les commandes) et de mise en scène pour le scénario. En effet, c'est ici que les personnages principaux vont se téléporter dans un autre monde.

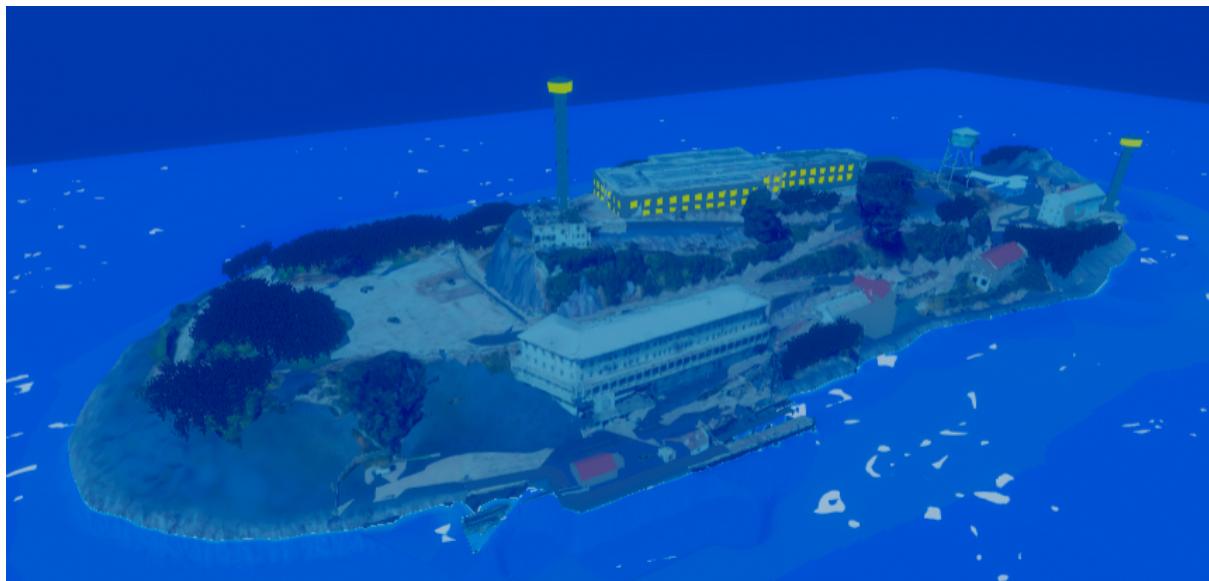


Image de la prison extérieure

La prison extérieure a été conçue à l'aide de Google Earth, Léo Farhi en a extrait le modèle 3D de l'île pour la refaire en partie. Le modèle 3D est affiché lors de la cinématique pour rendre l'animation plus jolie, mais lorsque nous contrôlons le personnage et que nous passons sur la partie extérieure, celui-ci disparaît et laisse place au Terrain (d'Unity) car le modèle de base possède quelques défauts.

Quant à l'intérieur de la prison, c'est Léo Alba qui l'a modélisé, Léo Farhi quant à lui l'a texturé et a rajouté la lumière. La modélisation s'est faite grâce à Pro Builder et le design est une invention de Léo Alba. Le design de cette prison devait répondre à certains critères et si possible être différent d'un style de prison classique. C'est pourquoi cette prison a un style tout à fait atypique qui mêle utilité et originalité.

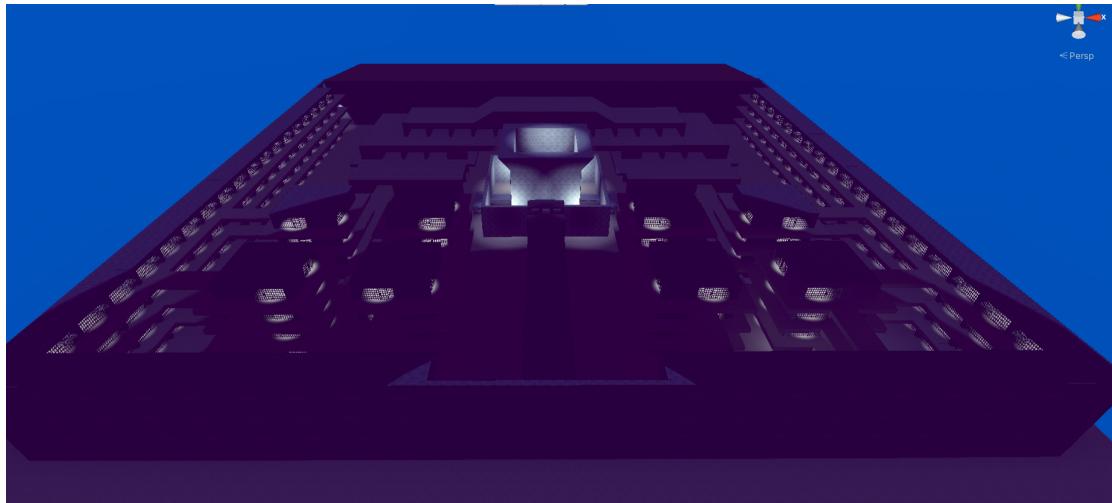


Image de la prison intérieure

Puis Hugo Rubio et Léo Farhi ont fait le laboratoire qui permet aux joueurs d'accéder à une "Jungle" (aller à la partie "scénario" pour comprendre)



Image de l'intérieur du laboratoire

Par la suite, les personnages principaux ont été téléportés dans la jungle. Cette zone représente le premier obstacle majeur du jeu. Le joueur devra, en plus de tuer quelques ennemis, résoudre des minis énigmes d'un temple pour avancer. La jungle a été créée par Léo Alba et Léo Farhi. Léo Alba s'est chargé de créer les minis défis que le

joueur devra passer pour avancer. Ainsi en cherchant bien, le joueur accédera alors à différentes salles .

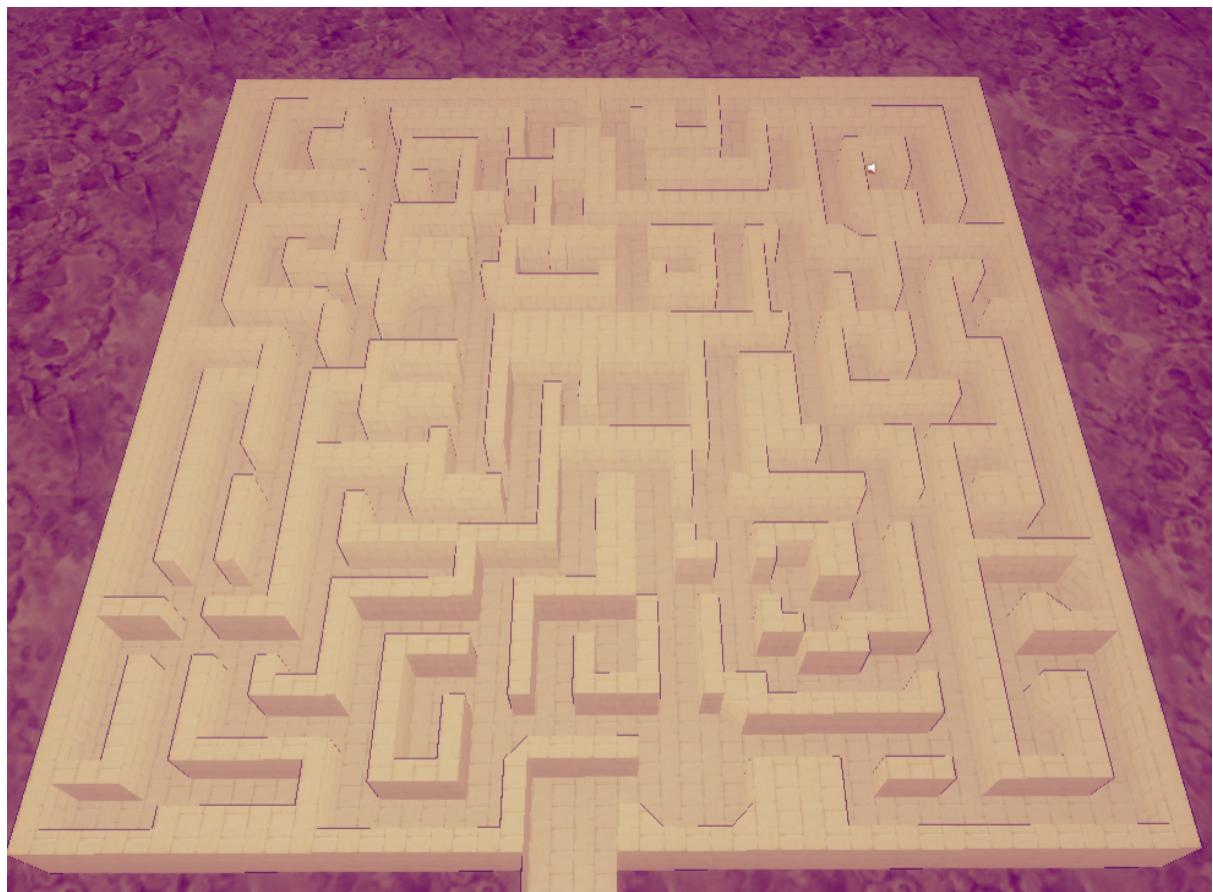


Image vue du dessus du labyrinthe du temple de la jungle

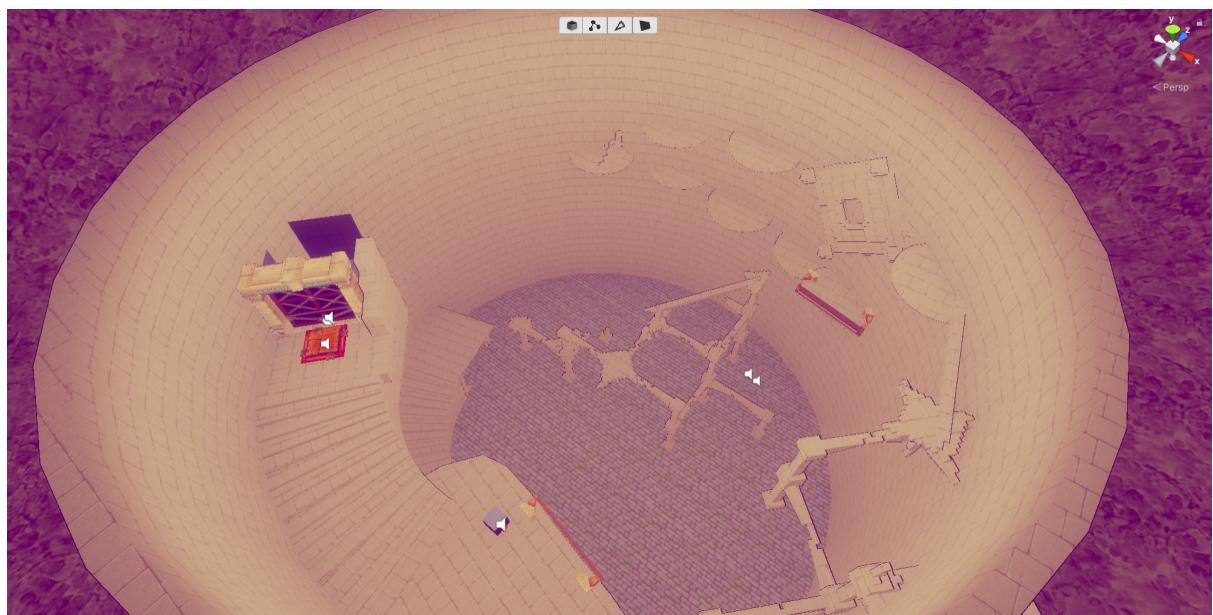


Image du parcours d'obstacles du temple de la jungle

Il a aussi fait toutes les ruines que le joueur devra fouiller/exploré, ces ruines qui permettent au joueur d'être installé dans une jungle qui a été la cible d'une grande catastrophe.

Mais il a aussi placé des collisions afin que le joueur ne puisse pas sortir des limites imposées. Il a aussi manipulé partiellement le Terrain en créant différent relief (fosse, petit chemin et montagne lointaine...), et s'est chargé de la végétation. Quant à Léo Farhi, il s'est chargé de donner la forme principale du Terrain et l'a texturé, il a aussi fait les espaces d'eau présents un peu partout dans cette scène. (Cette Zone est très grande, voir l'annexe pour plus d'images)



Image vue du dessus de la jungle

Le Temple aquatique a été fait par Léo Farhi qui s'est fortement inspiré du jeu "Zelda Skyward Sword".



Image du hall du temple de la jungle

Pour faire la transition entre la jungle et la suite du jeu et, comme dans tous les jeux de type RPG (Role player game), nous avons mis une ville. Pour cela, nous avons d'abord utilisé un algorithme pour placer des cubes indiquant où il fallait poser une maison et comment l'orienter afin de créer une ville. Suite à cela, Hugo a remplacé tous ces cubes par des "Assets" de bâtiments pour le rendu final. Nous avons procédé de cette manière pour rendre la tâche de la création de la ville plus simple et plus rapide. En effet, cela aurait pris plus de temps pour réfléchir au design de la ville puis de la construire.

Après la création de la ville, Hugo a placé un par un tous les nœuds constituant le "Population Engine" (cette notion est expliquée dans la partie IA du rapport), puis il a relié tous les nœuds entre eux afin de pouvoir implémenter les PNJ.



Image vue du dessus de la ville 1

Dans le scénario, les personnages principaux doivent directement se rendre dans la ville finale pour affronter Giovano, le boss final. Mais avant de faire cela, ils devront traverser une grande plaine réalisée par Hugo. Cette zone n'est qu'une zone de transition qui n'a pas vraiment d'importance hormis le côté visuel du jeu. En effet, le joueur peut soit explorer la zone en tuant des ennemis (il sera bien entendu récompensé pour cela) ou tout simplement aller directement à la ville finale pour continuer le jeu. Les ennemis de cette zone peuvent donner des objets très puissants (c'est la récompense) pour rendre les combats plus faciles.

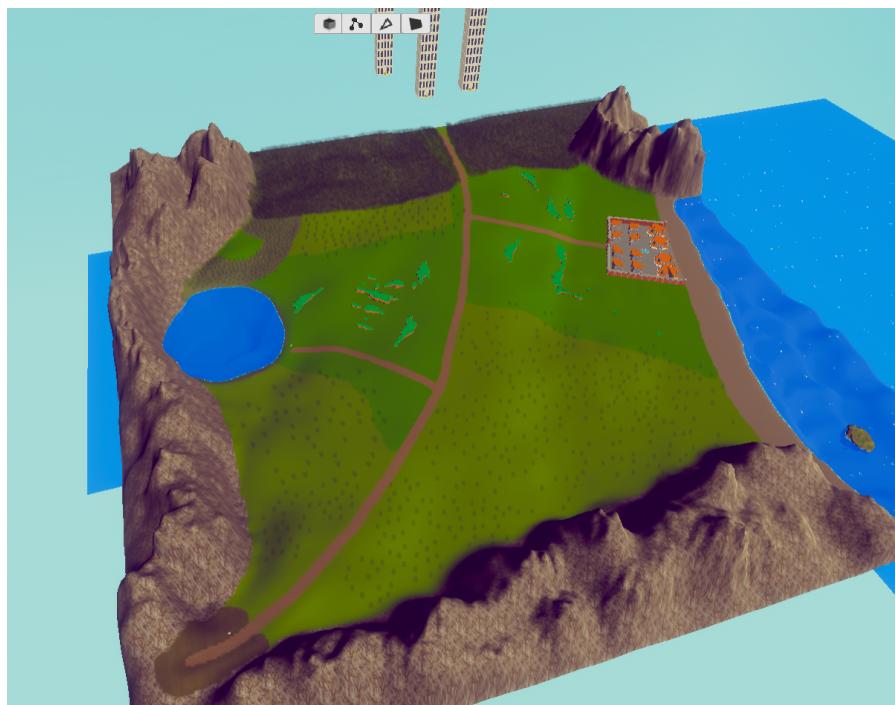


Image vue du dessus de la plaine

Enfin le joueur arrive dans la dernière ville du jeu. Le joueur doit trouver un moyen pour entrer dans la grande tour et affronter un premier boss avant d'y combattre le boss final et terminer le jeu.



Image vue du dessus de la ville 2



Image de l'étape N°1 de la Tour

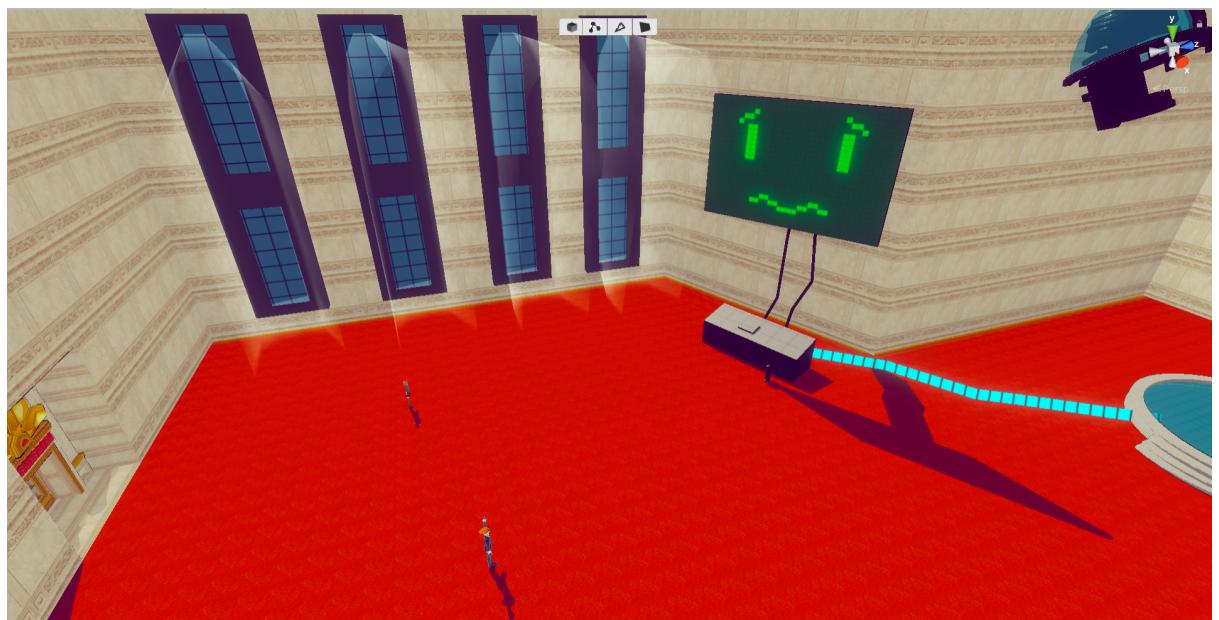


Image de la salle Finale du jeu

- Shader, Texture et Post Processing

Les shaders et les textures vont faire que le jeu soit beau graphiquement. Le post processing est un calcul de lumière qui s'ajoute à tout le reste pour rendre le tout encore plus beau, c'est une sorte de filtre. Cette partie est principalement gérée par Léo Farhi.

Les Shaders permettent des choses vraiment incroyables. Nous pouvons créer des objets ou des bâtiments avec peu de relief ou de basse qualité, mais les shaders sont là pour corriger nos erreurs grâce à leur effet d'embellissement, qui font que ces défauts passent inaperçus.

Les Shaders que nous avons sélectionnés permettent de rendre les modèles 3D plus mangas, l'image ci-dessous (prise sur internet) explique comment le calcul de lumière est fait sur un modèle 3D réaliste (sphère de gauche) et sur un modèle 3D style manga (sphère du milieu et de droite).



Différents types de shaders

- Gestion des lumières

Léo Farhi a fait la majorité de la gestion des lumières de toutes les scènes. Sélectionner l'intensité ainsi que l'angle du soleil par exemple, sont des choix très importants, un mauvais éclairage pourrait faire mal au yeux des joueurs à long terme.

Là où l'éclairage (l'intensité) a été important est notamment la prison car la scène se passe de nuit. Il fallait donc imaginer comment serait la lumière de nuit avec par exemple la réflexion de la lune sur le sol. Bref, plein de paramètres à prendre en compte.

Là où l'angle a été le plus important est la jungle, car si nous inclinons trop le soleil, l'ombre des falaises cacherait une grande partie de la zone, c'est pourquoi le soleil est plus incliné à la verticale pour éviter un maximum d'ombrages.

- Système jour/nuit

Afin que le jeu ne se déroule pas uniquement de jour, Léo Farhi a implémenté un système jour/nuit pour un peu plus de cohérence. Mais celui-ci n'est pas disponible partout. En effet, on ne peut pas se permettre de changer l'angle du soleil pour la jungle pour les raisons expliquées ci-dessus. C'est pourquoi nous avons décidé de ne le mettre que dans les villes et fermer les portes de la ville la nuit tombée.

Le ciel change de couleur en fonction de l'heure, les nuages bougent eux aussi.

- Générateur de PNJ avec tenue semi-aléatoire

Le système de générateur de PNJ permet de créer des PNJ qui ont des tenues et des parties du corps différentes afin de rendre une ville réaliste. Ce système a été fait par Léo Farhi et les modèles 3D quant à eux, ont été faits par Geoffrey Elbaz à l'aide de Vroid et de Blender.

Le principe de ce système est relativement simple contrairement à sa réalisation. Nous fournissons au système une multitude de modèles 3D (vêtements, cheveux, pantalons, yeux, etc...) et le système sélectionne de façon aléatoire ces éléments et crée une combinaison qui forme à la fin un PNJ. Mais le système n'est pas bête, il vérifie si les éléments (exemple : vêtement) sont en adéquation avec le genre du PNJ, autrement dit, il ne va pas mettre des tenues de femme sur un homme et vice-versa.



Exemple de modèle 3D utilisé pour le générateur de PNJ

Les informations liées aux éléments comme par exemple le genre, l'emplacement du model 3D, etc... sont sauvegardées dans un “ScriptableObject”.



Standar Girl

Open

Script	CharacterCreatorMesh
Name Part	
Sex	Girl
Type	Body
Model 3D	Standar Girl
Mesh Name	Body

GirlJacket1

Open

Script	CharacterCreatorMesh
Name Part	
Sex	Girl
Type	Cloth
Model 3D	vetement
Mesh Name	Body

“ScriptableObject” de sauvegarde des PNJ

En résumé, un ScriptableObject est un type de Class spécifique à Unity. Ce type permet de concrétiser une variable en permettant de la stocker dans un fichier. Ainsi, sauvegarder les différentes parties d'un modèle 3D n'a jamais été aussi simple.

Bien que les ScriptableObject nous aient facilité la tâche, il y a eu tout de même une partie très complexe : synchroniser les vêtements avec le squelette du PNJ.

Pour ce faire, Léo Farhi a créé un système qui permet de fusionner les deux squelettes (celui du vêtement et celui du joueur).

Tout d'abord il fallait vérifier si les deux squelettes avaient les mêmes os “Bones”, donc l'algorithme parcourrait de manière récursive à travers les deux squelettes pour s'assurer qu'ils avaient bien des parties identiques. Si ce n'est pas le cas alors le système recrée les parties manquantes.

```
⌚ Frequently called 2 usages Leo Farhi
private void ChildInstantiate(GameObject obj, GameObject TargetRoot)
{
    if (null == obj)
        return;
    foreach (Transform child in obj.transform)
    {
        if (null == child)
            continue;
        Transform childTarget = TargetRoot.transform.Find(child.gameObject.name);
        if (childTarget == null)
        {
            GameObject Instance = new GameObject(); //Instantiate(new GameObject(), new Vector3(0, 0, 0), Quaternion.identity);
            Instance.name = child.gameObject.name;
            /*
            Debug.Log(child.gameObject.name);
            Debug.Log(child.parent.gameObject.name);/*
            Instance.transform.parent = TargetRoot.transform;
            Instance.transform.localPosition = child.localPosition;
        }
        ChildInstantiate( obj: child.gameObject, TargetRoot: TargetRoot.transform.Find(child.gameObject.name).gameObject);
    }
}
```

Puis dans un second temps, le système fusionne les deux squelettes pour n'en former qu'un.

```

public void ReBone()
{
    ChildInstantiate( obj: root, rootTarget);

    Dictionary<string, Transform> boneMap = new Dictionary<string, Transform>();
    foreach (Transform bone in TargetMeshRenderer.bones)
        boneMap[bone.gameObject.name] = bone;

    SkinnedMeshRenderer myRenderer = OriginMeshRenderer;//gameObject.GetComponent<SkinnedMeshRenderer>()
    myRenderer.rootBone = rootTarget.transform;

    Transform[] newBones = new Transform[myRenderer.bones.Length];
    for (int i = 0; i < myRenderer.bones.Length; ++i)
    {
        GameObject bone = myRenderer.bones[i].gameObject;
        if (!boneMap.TryGetValue(bone.name, out newBones[i]))
        {
            newBones[i] = GetChildRecursive(rootTarget, bone.name).transform;
            /*
            Debug.Log("Unable to map bone '" + bone.name + "' to target skeleton.");
            break;*/
        }
    }
    myRenderer.bones = newBones;
}

```

Mais ces opérations sont très coûteuses à calculer pour la machine.

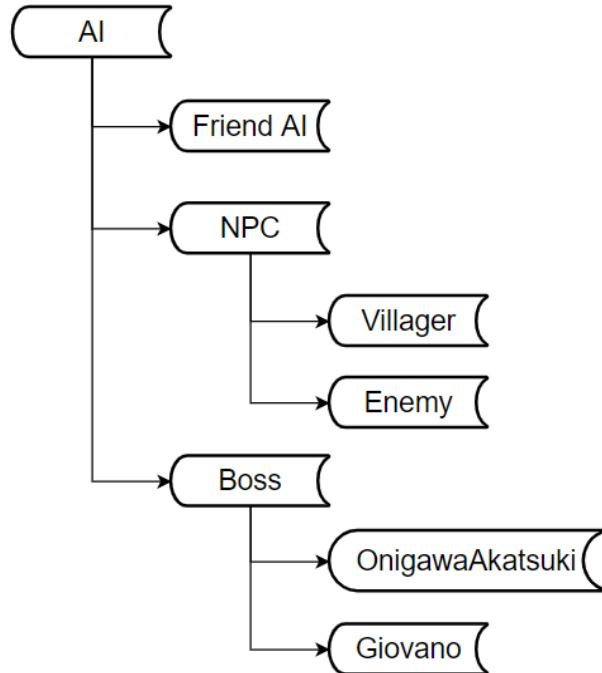
- Système d'intelligence artificielle

C'est Hugo Rubio et Léo Farhi qui ont travaillé sur cette partie-là. Pour rappel, l'IA est de 4 types dans ce jeu :

- Les personnages non joueurs (PNJ), qui sont juste une IA capable de se déplacer dans une ville sans pouvoir attaquer ni dialoguer. Leur but est à titre décoratif pour rendre le jeu plus vivant.
- Les ennemis, qui sont des IA plus élaborées que les PNJ puisque en plus de pouvoir se déplacer sur un terrain comme les PNJ, elles peuvent aussi engager un combat contre le joueur.
- Les boss, qui sont des ennemis plus puissants, possèdent chacun leur script car un boss est une IA unique.
- Enfin, les IA qui suivent le joueur. C'est l'IA la plus complexe car en effet, elles suivent le joueur tout le long du jeu, ce qui implique qu'elles ont le même comportement que le joueur. Elles sont donc capables de se déplacer sur n'importe quel terrain et, bien sûr, elles sont capables de combattre.

## Généralité sur les IA :

En utilisant la programmation orientée objet (POO) et le langage de programmation C#, l'architecture des scripts des IA est faite comme suit :



## Architecture des scripts pour les IA

La classe mère "AI" regroupe tous les attributs et les méthodes qui sont utilisés dans les différentes classes filles : "Friend AI", "NPC" et "Boss". En particulier, c'est dans cette classe que sont écrites les méthodes concernant la mort, l'application des dommages, et les différentes fonctions pour attaquer. En effet, cela est pratique de pouvoir utiliser les attaques d'un boss pour un ennemi juste en changeant l'animation qui est à part.

Pour les classes filles "Friend AI" et "Boss", elles regroupent plus précisément le mouvement des IA (expliqué plus bas) et les différents patterns (ici on parle d'un enchaînement de plusieurs attaques par exemple) propres à ces classes.

La classe "Boss" aura autant de sous-classes qu'il y aura de boss. Cela permet donc d'avoir plusieurs boss avec des patterns différents selon chacun. Il y aura donc 2 boss dans notre jeu qui auront chacun un script à leur nom.

Enfin la classe fille "NPC" contient deux sous-classes "Villager" et "Enemy". En fait, ces scripts ont été placés de cette manière car les PNJ villageois et les ennemis ont le même déplacement. Les avoir séparés nous permet de mieux nous y retrouver. Ainsi,

la classe “NPC” contient le déplacement des PNJ et les sous-classes “Villager” et “Enemy” contiennent respectivement tout ce qui est lié aux PNJ (à savoir interaction avec le joueur) et tout ce qui est lié aux ennemis (à savoir le pattern des combats).

Par ailleurs, tous les déplacements des IA sont gérés en utilisant les propriétés du “Navmesh” qui est un système intégré à Unity qui sert à faire du pathfinding, un autre système permettant à un objet de trouver le chemin le plus court pour atteindre une destination. Les IA peuvent donc à chaque instant trouver la position du joueur et se diriger vers lui en cherchant le chemin le plus court. Le Navmesh permet donc une implémentation simple et efficace des IA qui peuvent se déplacer vers une cible. Dans l'image ci-dessous, le “navmesh” est la zone en bleu recouvrant toutes les parties du terrain où les IA pourront se déplacer.

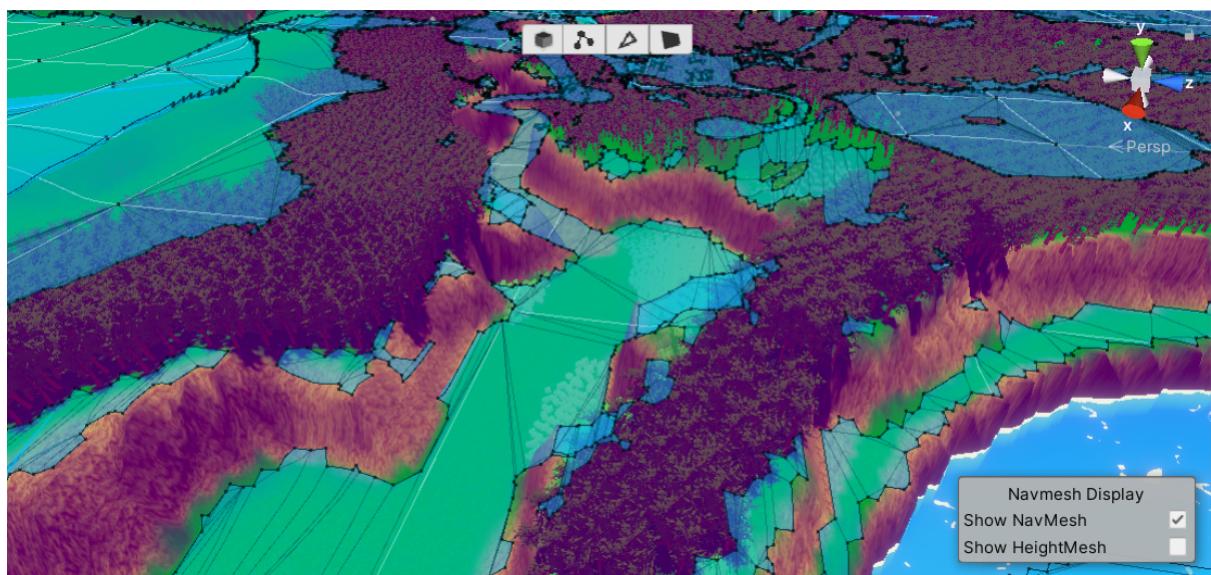


Image du “navmesh” dans la zone de la jungle

- Evolution des IA au cours du temps :
  - **Première soutenance :**

L'IA n'était pas la priorité de la première soutenance, mais la fonctionnalité des IA amies du joueur était terminée. Ils vont donc utiliser le Navmesh et avoir comme cible le joueur. Cela aura pour conséquence un déplacement des IA vers le joueur à chaque fois que le joueur se déplace. Ce système de déplacement a été pensé et implémenté par Hugo.

Par conséquent, ce déplacement fonctionne de la façon suivante : si l'IA est à une distance proche du joueur, alors elle marche. Si elle est à mi-distance, alors elle court et enfin si elle est loin du joueur, alors elle sprinte. Le déplacement est donc facilement implémenté par une suite de conditions comme le montre le code suivant :

```

5 références
public override void AIMove()
{
    distanceTarget = Vector3.Distance(target.position, transform.position);

    //A améliorer
    if (distanceTarget > distanceIdle)
    {
        if (distanceTarget > distanceSprint)
        {
            ChangeAnimation("sprint", true, sprint);
        }
        else if (distanceTarget > distanceRun)
        {
            ChangeAnimation("run", true, speedRun);
            ChangeAnimation("sprint", false, speedRun);
        }
        else if (distanceTarget < (distanceRun - 0.1))
        {
            ChangeAnimation("run", false, speedWalk);
            ChangeAnimation("walk", true, speedWalk);
        }

        agent.destination = target.position;
    }
    else
    {
        ChangeAnimation("walk", false, 0);
    }
}

```

La méthode *Distance* permet de calculer la distance entre le joueur et l'IA, *ChangeAnimation* permet de changer d'animation (détails dans la partie animation) ainsi que la vitesse de déplacement de l'IA (en effet, si elle court, elle sera alors plus rapide que si elle marche !), et enfin *agent.destination* permet d'utiliser le Navmesh et faire en sorte que l'IA se déplace vers le joueur en prenant le chemin le plus court.

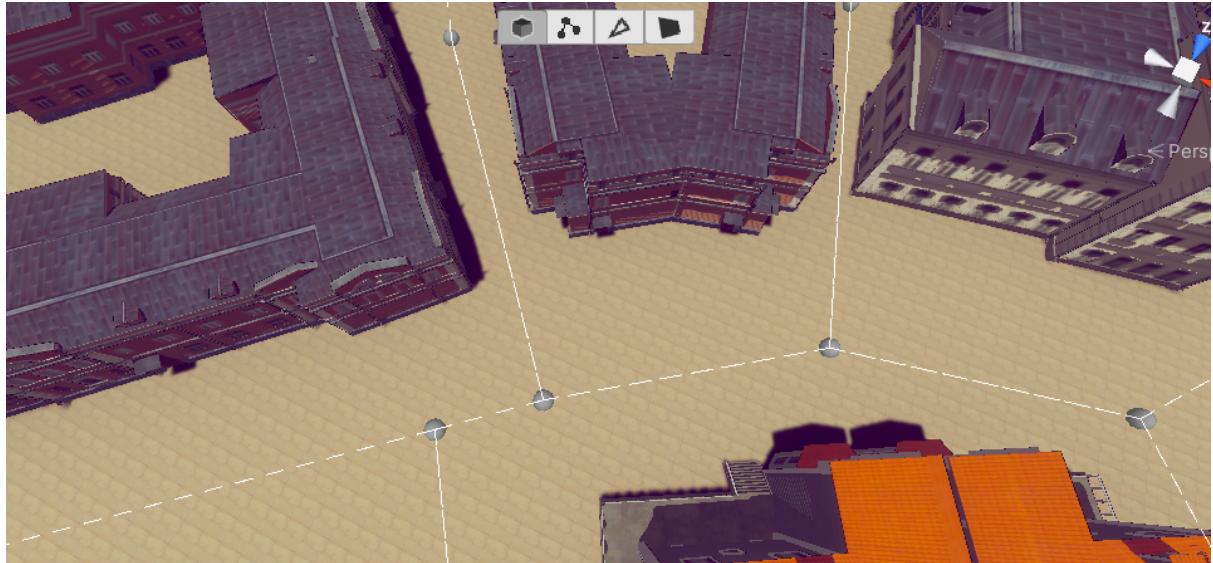
- **Seconde soutenance :**

Pour cette soutenance, nous nous sommes davantage concentrés sur le déplacement des PNJ/ennemis ainsi que sur le système de combat des IA.

Pour faire les PNJ, nous les avons créés sur Vroid, puis nous en avons cloné quelques-uns dans la ville avec le “Population Engine”, mais qui auront différents vêtements pour les distinguer. Cela donne l’illusion d’avoir un semblant de ville peuplée. Et ces PNJ auront une IA qui leur permettra de se promener librement dans la ville. Leurs déplacements seront un peu aléatoires mais un minimum réfléchis, c'est-à-dire qu'une ville sera par exemple composée de "nœuds" (invisibles pour le joueur) placés à chaque intersection de rue. Ainsi, il pourra se diriger grâce au “NavMesh” intégré à Unity. C'est le système de Population engine que nous avons implémenté.

Plus précisément, le “Population Engine” a été fait par Léo Farhi et il est utilisé par Hugo pour implémenter les PNJ. C'est un système de guidage pour les IA qui se baladent dans la ville. Ce système fonctionne à l'aide de nœuds (Node) et de lignes

(Edge), les nœuds représentent chaque intersection d'une ville, quant aux lignes, elles représentent les routes que peuvent emprunter les PNJ.



Vue du dessus de la ville 1 avec les noeuds du population engine

Lorsqu'un PNJ arrive sur un nœud, celui-ci est redirigé vers un autre. Pour faciliter l'installation de ce système sur les villes, Léo Farhi a dû modifier le comportement de l'éditeur afin que nous puissions voir les lignes et les nœuds dans l'éditeur et non en jeu à l'aide des "Gizmos".

Par conséquent, les nœuds sont les cibles des PNJ utilisant le Navmesh. Pour se déplacer, les PNJ vont chercher le chemin le plus court pour arriver au prochain nœud et avancer vers lui.

Concernant les combats, tout le système qui suit a été principalement pensé et fait par Hugo avec l'accord des autres membres du groupe. Les ennemis standards n'auront que la possibilité d'attaquer ou de se déplacer pendant le combat de façon aléatoire. Les boss quant à eux auront leurs propres scripts, donc leurs propres styles de combat. Bien évidemment, tous les boss attaquent et se défendent. Enfin, pour les IA qui remplacent les joueurs, leur système de combat est plus complet que celui des ennemis standards.

Dans le jeu, un combat se déclenche lorsque le joueur ou l'IA est assez proche d'un ou plusieurs ennemis. Il a été décidé que c'est l'ennemi qui déclenche un combat pour éviter tout bug avec les autres IA (sauf si le joueur attaque en premier). Un combat se termine si tous les ennemis sont morts ou si tous les joueurs meurent.

Ce système est géré suivant un "diagramme à conditions" (que vous pouvez voir plus bas dans cette partie). Pendant les combats, les IA ont différents états, mais ne peuvent pas en avoir plusieurs en même temps. Ces états sont : Dodge (esquive), Impact,

Random Move (mouvement aléatoire), Attack (attaque), Dead (mort) et enfin None (nul) qui est l'état par défaut. Ces différents états sont activés en fonction de plusieurs conditions. Par exemple, l'état Attack ne peut s'activer que si l'IA est à une distance proche de l'ennemi. Ainsi, si aucune condition n'est remplie, alors, l'IA va se déplacer de façon aléatoire. En fait, pendant un combat, l'IA se déplacera de façon aléatoire (avec une probabilité plus grande de se rapprocher de son ennemi) jusqu'à ce qu'une condition soit remplie et change son état.

Par la suite, une fois que l'IA a atteint un état (Dodge, Impact ou Attack), c'est là qu'un système de probabilité intervient. En effet, comme il s'agit d'une sorte d'IA, nous ne voulons pas qu'elle exécute une action dès qu'une condition est remplie (un humain ne ferait jamais cela). C'est pour cela que nous introduisons ce système de probabilité. L'IA aura donc une certaine probabilité de faire son action.

Par ailleurs, cette probabilité est variable en fonction de l'ennemi. Par exemple, un ennemi faible aura une probabilité plus faible d'attaquer et une probabilité nulle d'esquiver. Alors qu'un ennemi plus fort pourra esquiver et attaquer plus fréquemment.

Enfin, certains états sont prioritaires sur les autres (ici "Impact" et "Dead"). Cela signifie que si la condition pour activer cet état est vérifiée, alors que l'IA est dans un autre état, c'est l'état prioritaire qui sera actif. Cela permet par exemple d'éviter que l'IA bouge alors qu'elle est morte ou que l'IA continue d'attaquer alors qu'elle vient de se prendre un coup (si la probabilité de l'état "Impact" est vérifiée bien sûr).

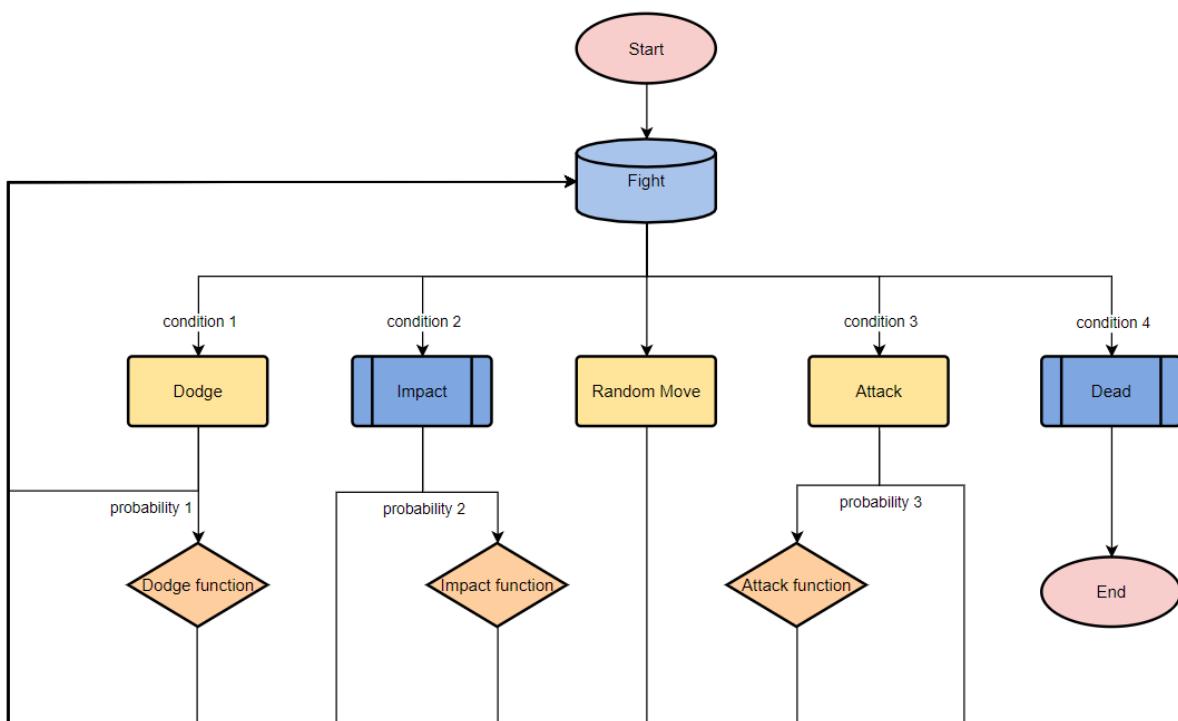


Diagramme de fonctionnement des combats des IA

- **Troisième soutenance :**

Pour la dernière soutenance, nous avons tout d'abord fait les scripts des différents boss, des corrections de bugs et des ajouts de fonctionnalités pour les IA qui suivent les joueurs afin de terminer le jeu sans problème.

❖ **Les boss**

Les boss utilisent presque le même procédé pour les combats que les ennemis. Là où la différence est flagrante, c'est au niveau des dégâts, des points de vie du boss mais aussi dans la façon de se déplacer de celui-ci lors d'un combat. En effet, alors qu'un ennemi est dans l'état "random move", les boss n'auront pas le même état. Les mouvements du boss seront alors plus prévisibles mais celui-ci attaquera continuellement le joueur ou les IA qui accompagnent le joueur. Par conséquent, ce qui rend un combat de boss plus difficile c'est le peu de temps de répit lors du combat.

De plus, lorsque le boss arrive à la moitié de sa vie, il devient plus puissant faisant alors plus de dégâts et il bouge plus vite.

Concernant le premier boss Onigawa Akatsuki, sa particularité est d'avoir un gain de dégâts supplémentaire lorsque sa vie atteint la moitié de son maximum. Il aura aussi la possibilité de se soigner en plein combat, le rendant plus difficile à tuer.

Pour Giovano, le boss final, il ne pourra pas se soigner mais lorsque sa vie sera de moitié, une cinématique se lancera et il aura également un gain de dégâts. De plus, lorsqu'il lui restera un tiers de sa vie maximale, il aura encore un gain de puissance. On parle du boss final, il ne faut pas rigoler !

❖ **Les dernières fonctionnalités des IA**

Pour pouvoir finir le jeu sans rencontrer trop de bugs visuels, certaines fonctionnalités ont été ajoutées pour corriger cela.

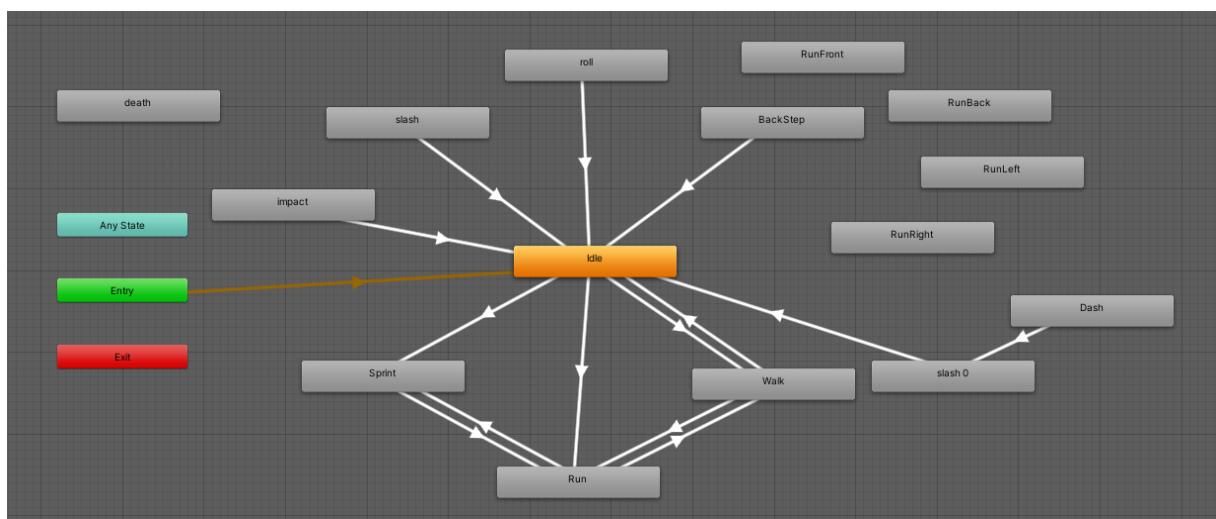
Tout d'abord, la possibilité des IA de se téléporter vers le joueur lorsqu'elles sont trop loin de celui-ci. Par exemple, si une IA est bloquée dans une texture (oui ça arrive !), il suffit que le joueur s'éloigne pour que l'IA se débloque et puisse continuer à jouer. De même lors d'un changement de scène.

Une autre fonctionnalité est de bloquer volontairement l'IA lorsque le joueur effectue certaines actions. En effet, il arrive qu'une IA doive rester sur une plateforme.

- Les animations

Pour les animations, comme nous l'avons précisé dans le cahier des charges, nous avons importé des animations. Ensuite, Hugo et Léo Farhi ont utilisé ces animations dans l'Animator de Unity pour les appliquer aux personnages du jeu.

L'Animator de Unity peut être assez complexe d'utilisation mais nous nous sommes contentés du minimum pour avoir un bon rendu. Pour ce qui est du fonctionnement de l'Animator, nous avons juste importé les animations déjà faites de mixamo (un site où l'on peut importer des animations) et placé les animations dans l'Animator. Par la suite, nous avons placé des transitions (symbolisées par des flèches dans l'Animator) entre les animations pour les rendre plus fluides. De plus, pour savoir quelle transition doit être effectuée, un booléen leur est attribué.



## Animator des IA

Une fois les animations mises dans l'Animator, celles-ci sont contrôlées dans le script des différents personnages (joueur ou IA). De surcroît, il y a autant d'Animator qu'il y a de personnages ayant des animations différentes. La grande difficulté des animations est de créer un lien entre elles pour rendre le visuel fluide. En effet, une mauvaise gestion de l'Animator et des propriétés des animations peut avoir pour conséquence des bugs visuels. Il est donc important d'avoir une bonne gestion des animations.

Une autre difficulté des animations est de voir toutes les animations sans latence ou de voir les bonnes animations dans le mode multijoueur du jeu. Pour cela, nous avons ajouté un composant à chaque objet possédant une animation. Ce composant est un photon animator view et est un script permettant de synchroniser les animations en mode multijoueur. Une fois le composant mis, il faut mettre toutes les animations de

l'Animator en mode “Continue” pour voir toute l'animation s'effectuer sur tous les écrans. En effet, si ce n'est pas le cas, les joueurs ne verraient seulement que le début de l'animation puis un freeze.

- Musiques

Geoffrey Elbaz s'est chargé de composer 5 musiques pour les différents lieux ou moments du jeu:

- le menu principal
- l'intérieur de la prison
- l'intro de la jungle
- la jungle
- la ville n°1

La composition de ces musiques fut effectuée grâce au logiciel de production musicale assisté par ordinateur logique. l'inspiration est totalement personnelle, cependant les instruments proviennent de sampleur virtuel contrôlé par du MIDI (Musical Instrument Digital Interface, qui est un langage de communication entre le clavier et le logiciel) et non par des enregistrements studio. Certaines autres de ces musiques sont composées à l'aide d'intelligence artificielle comme AIVA, tandis que d'autres sont des musiques libres de droit.

- Network

Le Network a principalement été fait par Léo Farhi et il a créé un système qui s'adapte à toute situation grâce au ScriptableObject.

Par exemple, ces deux fonctions permettent de synchroniser l'apparence des PNJ chez tous les joueurs :

```

private void SerializeData()
{
    this.m_StreamQueue.SendNext(DataPNJ);
}

private void DeserializeData()
{
    string data = "";
    try
    {
        data = (string) this.m_StreamQueue.ReceiveNext();
        if (data=="")
            return;
        GameObject.FindObjectOfType<PNJgenerator>().GenerateWithBase(this.gameObject, data);
        Destroy(this);

    }
    catch{ }

}

```

Celui qui héberge la session envoie à l'aide de *SerializeData* les informations relatives aux PNJ, et les clients reçoivent les données à l'aide de *DeserializeData* qui récupère l'information, puis la transmet au *PNJgenerator* afin que le système puisse générer un PNJ identique à celui de l'hébergeur.

Le système de communication est composé de deux principaux types d'envoi de données. Le premier principe a pour but de synchroniser la même variable pour tout le monde (par exemple, l'heure du système jour-nuit). Quant à l'autre type de données, c'est la conservation indépendante des données des joueurs, par exemple, le choix de notre joueur. Chaque client envoie le personnage qu'il a choisi au serveur sans pour autant modifier la valeur des autres clients, cette donnée est purement indicative.

Pour synchroniser certains évènements comme par exemple le déclenchement des cinématiques ou le changement de zone, Léo Farhi a créé un système nommé “Photon Event” qui permet d'envoyer à tout le monde un signal. Ce système est utilisé très fréquemment dans le projet, il est même indispensable pour naviguer entre les différentes zones. Le script n'est pas très grand mais il remplit correctement sa tâche :

```

public class PhotonEvent : MonoBehaviour
{
    public UnityEvent actions; // 16+ methods

    private const byte INVOKE_EVENT = 0;

    # Eventfunction & Leo Farhi
    private void OnEnable()
    {
        PhotonNetwork.NetworkingClient.EventReceived += EventReceived;
    }

    # Eventfunction & Leo Farhi
    private void OnDisable()
    {
        PhotonNetwork.NetworkingClient.EventReceived -= EventReceived;
    }

    # Frequently called [ 2 usages & Leo Farhi
    private void EventReceived(EventData obj)
    {
        if (obj.Code == INVOKE_EVENT)
        {
            object[] datas = (object[])obj.CustomData;
            if (this.GetInstanceID() == (int)datas[0] // (bool)datas[0])
            {
                actions.Invoke();
            }
        }
    }

    # 23+ asset usages [ 6 usages & Leo Farhi
    public void Invoke()
    {
        object[] datas = new object[] {this.GetInstanceID()};//true};
        if (PhotonNetwork.IsConnected // && PhotonNetwork.IsMasterClient)
            PhotonNetwork.RaiseEvent( eventCode: INVOKE_EVENT, datas, RaiseEventOptions.Default, SendOptions.SendUnreliable);
        actions.Invoke();
    }
}

```

Donc pour l'expliquer, on l'exécute à l'aide de la fonction "Invoke" et celle-ci va envoyer le signal aux autres joueurs et les joueurs recevront grâce à "EventReceived". Une vérification du signal est effectuée aussi.

Pour synchroniser les joueurs et les PNJ, on utilise les fonctions déjà existantes de photon avec "Photon View", "Photon Transform View", "Photon Animator View".

Il y a aussi un autre point très important, c'est la synchronisation des Items. Pour être clair, quand on fait apparaître un item sur la scène, c'est le joueur qui la fait apparaître qui possède les "droits" dessus. En d'autres termes, c'est lui qui a le droit de lui apporter des modifications et donc par exemple le supprimer, mais cela engendre un problème qui est que si un autre joueur souhaite le ramasser, il n'a théoriquement pas le droit. Pour pallier ça, lorsqu'un joueur souhaite ramasser un item, il va d'abord demander les droits à celui qui les possède, puis il effectue des actions en fonction de ce qu'il compte faire avec.

Par exemple, le script ci-dessous montre une partie du script qui permet de ramasser un item. On peut voir que le système effectue la requête de transfert de droit à l'aide de la fonction “photonView.RequestOwnership()” et attend qu'on lui donne avec la “boucle while” marquée juste en dessous. Puis effectue le transfert de données dans l'inventaire.

```

↳ Frequently called 1 usage  ↳ Leo Farhi
public void PickUp()
{
    if (PhotonNetwork.isConnected)
    {
        timer = Time.time;
        StartCoroutine(routine: PickUpOnline());
    }
    else
    {
↳ Frequently called 1 usage  ↳ Leo Farhi
        inventairePlayer.inventory.Add(item: dataObject.Clone());
        inventairePlayer.canPickUp.Remove(item: this);
        DestroyImmediate(this.gameObject);
    }
}

IEnumerator PickUpOnline()
{
    photonView.RequestOwnership();
    while (!photonView.IsMine && timer+10>Time.time)
    {
        yield return null;
    }
    if (photonView.IsMine)
    {
        inventairePlayer.inventory.Add(item: dataObject.Clone());
        inventairePlayer.canPickUp.Remove(item: this);
        try
        {
            PhotonNetwork.Destroy(this.gameObject);
        }
        catch
        {
            inventairePlayer.canPickUp.Add(item: this);
            inventairePlayer.inventory.RemoveAt(index: inventairePlayer.inventory.Count - 1);
        }
    }
}

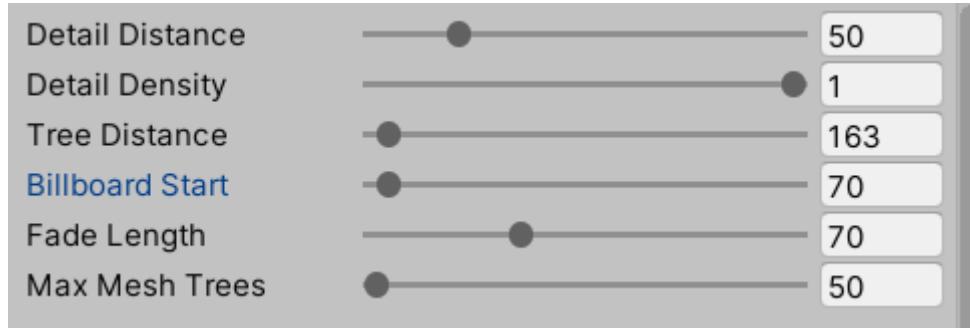
```

- Optimisation

La partie optimisation est essentielle afin qu'un ordinateur peu puissant puisse faire tourner le jeu, mais il est préférable d'avoir une carte graphique et un bon processeur pour être sûr de n'avoir aucun souci.

Pour optimiser le jeu du point de vue visuel, nous utilisons l'**occlusion culling**. Ce système implémenté par Unity permet de cacher les éléments qui ne sont pas dans le champ de vision de la caméra et donc d'alléger grandement le rendu graphique.

Pour optimiser les Terrains, notamment celui de la Jungle, on joue sur certains paramètres. Le plus important est celui de la distance d'affichage des arbres, par exemple l'image ci-dessous montre les paramètres sélectionnés pour la jungle, car dans cette zone il y en a vraiment beaucoup.



Une autre optimisation est par exemple dans le menu principal (expliqué plus haut), l'arrière représente une ville mais en vérité ce sont quelques maisons bien placées qui donnent cette illusion :



Nous utilisons aussi un système de téléportation invisible fait par Léo Farhi. Ce système a trois fonctions :

- Faciliter les échanges entre les différents lieux
- Alléger le rendu graphique
- Gagner de l'espace

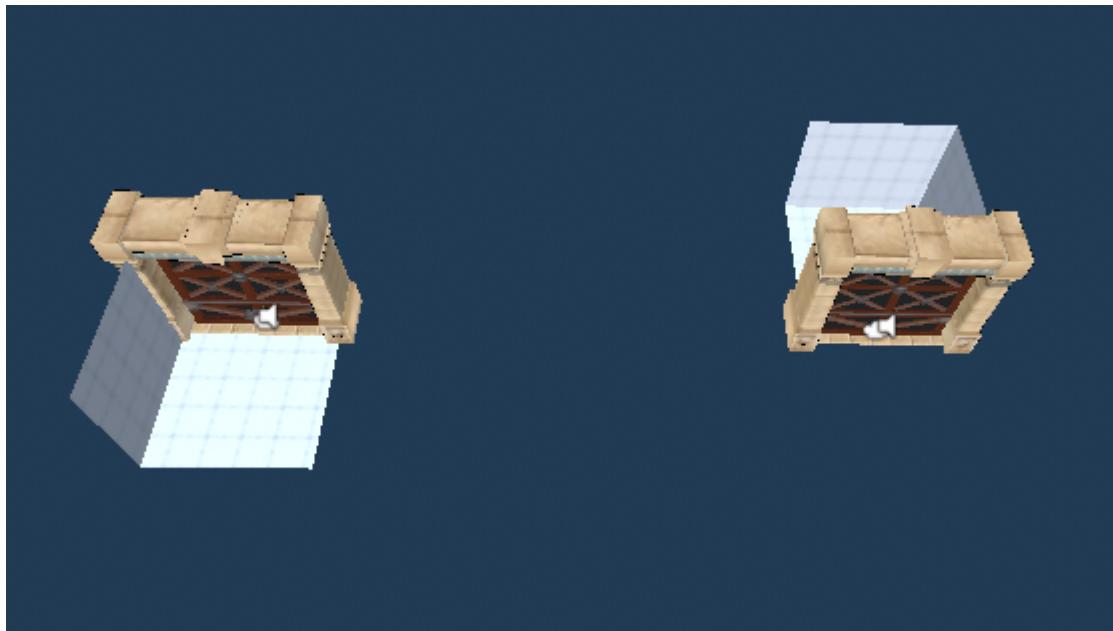


Image du système de téléportation

En pratique, le joueur rentre d'un côté et en sort de l'autre. Quand le joueur s'éloigne des portes, un écran noir se place devant la porte et le système désactive le portail afin d'alléger le rendu graphique.

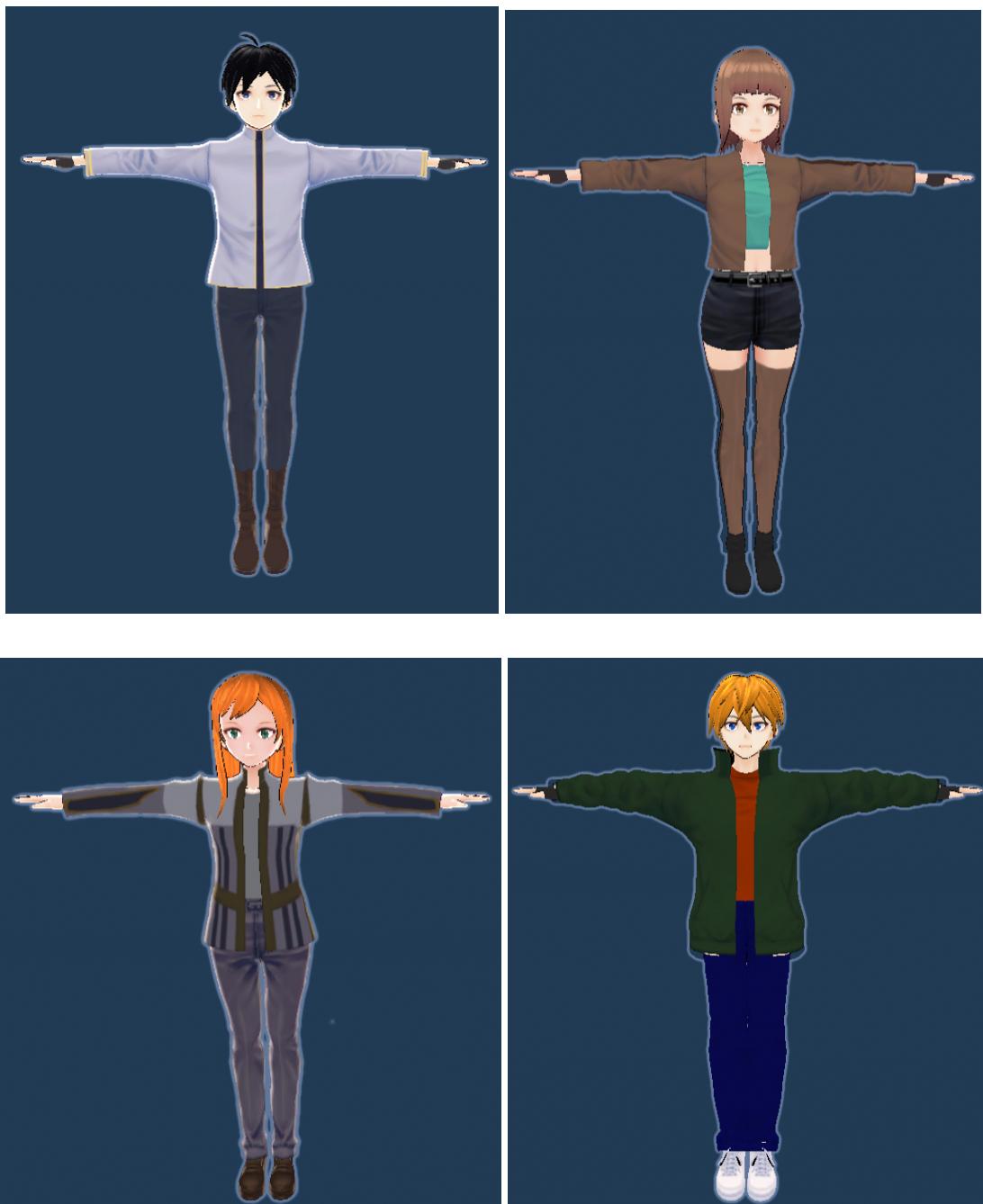
En outre, cela permet comme dans différents jeux (Pokemon, Mario Bros,...) de gagner de la place. En effet, certaines salles ne sont pas à l'échelle de leur bâtiment, ce qui permet de gagner énormément d'espace et de pouvoir manipuler ces salles sans que cela n'affecte l'aspect extérieur du bâtiment.

- Modèles 3D

Dans notre jeu, nous avons décidé de modéliser et d'avoir un visuel sur le style manga car plus facile à manipuler et offre un large panel de possibilités. Pour ce faire, nous avons décidé de faire la majeure partie de la modélisation et de l'importation des modèles 3D le plus tôt possible afin d'avoir une base solide sur laquelle s'appuyer.

Avant la première soutenance, Hugo Rubio a modélisé les personnages principaux ainsi que tous les ennemis et boss que va composer le jeu. Léo Farhi a pris le relais afin que nous puissions varier un peu les styles des différents personnages. Il s'est occupé principalement des personnages secondaires. Geoffrey Elbaz a lui aussi

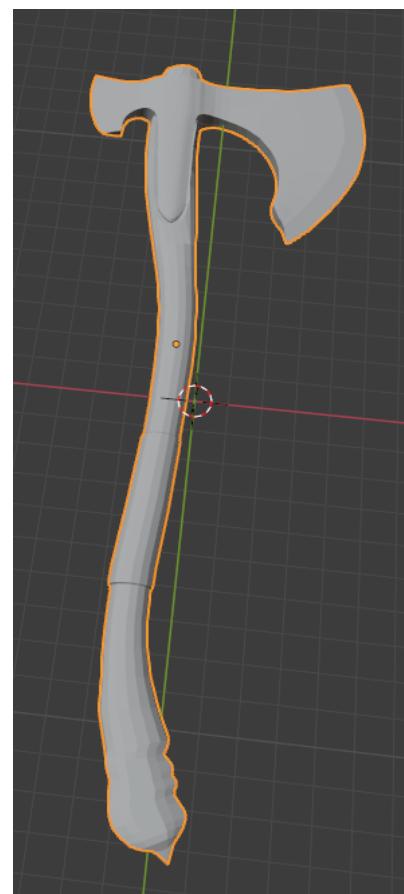
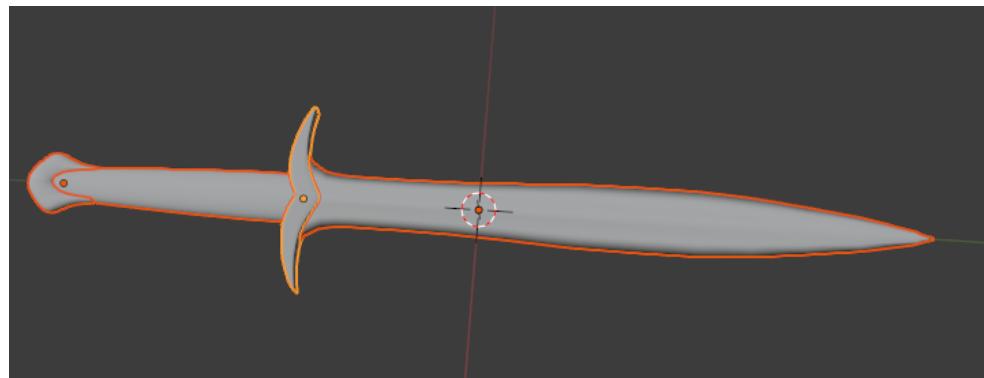
participé à la réalisation des habits réservés aux personnages non joueurs (PNJ), Geoffrey a réalisé 3 modèles 3D dans chaque catégorie de vêtements (cheveux, haut, bas) et ceci pour chaque genre (homme, femme), puis l'implémentation est effectuée presque aléatoirement. Nous utilisons Vroid pour les faire, car ce logiciel nous simplifie grandement la tâche, la forme du personnage est déjà faite. Il nous suffit de renseigner les paramètres liés à sa morphologie, de lui appliquer des textures aux vêtements, de lui faire une coupe de cheveux et c'est terminé.



Personnages principaux du jeu

Concernant la modélisations des objets, nous avons décidé de nous concentrer plus sur la modélisation des objets uniques de notre jeu comme les armes.

Pour cela Geoffrey a modélisé des objets tels que certaines armes pour les personnages principaux ou les ennemis à l'aide du logiciel Blender.



#### Les armes modélisées pour les personnages principaux

Tout cela a été réalisé avant la seconde soutenance. En effet, la modélisation n'est qu'un bonus pour le jeu. Il est bien plus agréable de jouer avec de bons graphismes mais le plus important est le côté fonctionnel.

- Outils extérieurs
  - Jaquette

La jaquette a été faite entièrement par Léo Farhi. Pour réaliser cette jaquette, Léo a créé une scène spéciale afin de faciliter sa fabrication et s'est inspiré des designs des jaquettes Nintendo Switch en respectant certains logos.



Jaquette recto-verso de la boîte du jeu

- Livret d'utilisation

Le livret d'utilisation est quant à lui disponible dans les boîtes remises au jury et comporte toutes les explications utiles à connaître pour jouer au jeu. Il expliquera comment installer le jeu mais aussi les différentes commandes utilisées par le joueur pour pouvoir jouer.

En parlant des commandes de jeu, nous allons résumer de quoi il s'agit. Le jeu permettant au joueur de se déplacer dans une dimension 3D, les touches 'z', 'q', 's' et 'd' permettent au personnages principaux de se déplacer dans le jeu. Il en est de même avec les flèches directionnelles du clavier. Avancer tout en appuyant sur la touche 'maj' permet de sprinter. Enfin, appuyer sur la touche "espace" permet au joueur de sauter.

Pour accéder à un menu, il faut appuyer sur la touche ‘échap’ du clavier et on utilise la souris pour sélectionner les objets que l’on veut par exemple utiliser.

Pour prendre un objet au sol ou encore interagir avec le milieu extérieur, il faut appuyer sur la touche ‘E’. Bien évidemment il est impossible d’interagir avec tout l’environnement. C’est pourquoi un ‘E’ apparaîtra à l’écran lorsque cela est possible.

Pour le système de combat, le joueur doit appuyer sur la touche ‘tab’ pour attaquer. Pour rendre le jeu plus difficile, il est impossible d’esquiver ou de bloquer une attaque. Par contre, le joueur aura la possibilité d’échanger de place avec une IA en appuyant sur la touche ‘w’ du clavier.

- Autres fonctionnalités
  - Système de combat

Pour rendre les combats plus épiques, Léo Farhi a créé une arène de combat qui change en fonction du rythme de la musique. Le script récupère le spectre audio de la musique et affiche ses variations en jeu.

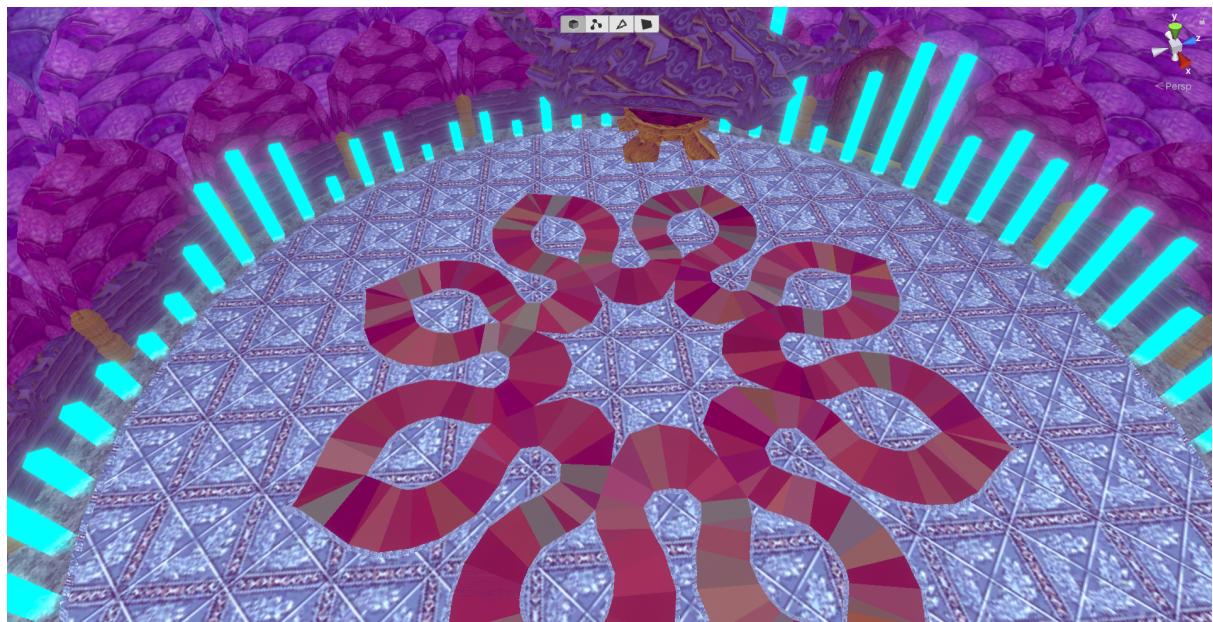


Image du spectre audio lors des combats importants

- Les Menus

Les menus d’un jeu constituent peut-être la partie la plus importante d’un jeu. Ce sont eux qui font le lien entre le jeu sur un écran et le joueur assis en face de l’écran. Pour cela, les menus doivent être faciles à comprendre pour que les joueurs ne se perdent pas dans les menus, qu’il n’y ait pas trop d’informations inutiles. De nombreux jeux sont critiqués pour la complexité de leurs menus. Nous avons donc décidé de faire des menus simples, qui donnent les informations nécessaires et suffisantes tout en

gardant un côté artistique et visuel sympathique. Léo Farhi a fait la majorité des menus, que ce soit le menu d'inventaire ou celui de pause ainsi que le menu principal. Geoffrey a travaillé sur celui du multijoueur.

- Menu principal

Le Menu Principal est sur une scène dédiée où nous jouons sur la perception en manipulant correctement l'angle de la caméra. Nous avons l'impression que nous faisons face à une ville, mais en vérité, ce sont quelques bâtiments correctement placés qui donnent cette illusion. Il suffirait de changer l'angle de la caméra pour s'en rendre compte (voir dans l'annexe pour plus d'images). Mais cette illusion n'est pas faite pour rien, c'est une question d'optimisation. En effet, il ne faudrait pas que l'ordinateur du joueur surchaaffe dès qu'il lance le jeu. Non, il faut que le menu soit fluide, c'est pour cette raison que cette illusion a été créée. Nous avons aussi rajouté quelques PNJ pour rendre le résultat encore plus réaliste.



Image du menu principal du jeu

- Menu de sélection des personnages

Le Menu de sélection des personnages n'est disponible que pour le Mode en Ligne, car dans le mode solo, le joueur peut changer de personnage en appuyant sur W. Dans le mode multi, seul le maître de la partie peut changer d'où le fait d'avoir créé ce menu spécifique. Ce menu permet aussi d'attendre que tous les joueurs soient prêts

avant de lancer la partie. Pour lancer une partie, seul le maître de session a accès au bouton commencer.



Menu de sélection des personnages en mode multi

- L'inventaire

Le système de l'inventaire permet d'avoir autant de slots d'emplacement que nous souhaitons grâce à un système de glissement vertical et horizontal.

Quand nous nous déplaçons de gauche à droite, nous avons la liste des quêtes (les missions du jeu) en cours, l'inventaire et le menu de sauvegarde.



Menu de l'inventaire

- Interfaces

Le menu d'inventaire a été fait de telle sorte que nous puissions rajouter autant d'éléments que nous le souhaitons. Le problème que nous avons cherché à régler est celui du tri par catégorie, par exemple ranger les armes avec les armes etc... Ce n'est pas aussi simple que nous pouvons l'imaginer car il faut aussi penser à l'aspect esthétique et pas seulement pratique.

Il y a une véritable réflexion autour des Interfaces, comme par exemple l'affichage des points de vie. Nous avons hésité entre des coeurs comme dans Zelda ou Minecraft mais finalement nous avons choisi une barre de vie comme dans Sword Art Online ou Dark Souls.

La couleur de la jauge de vie change en fonction du nombre de PV (nombre de Point de Vie)

Cette jauge est aussi personnalisable, c'est-à-dire que nous pouvons définir un nombre illimité de points de vie (en pratique ce n'est pas le cas, c'est limité à la valeur maximum que peut atteindre une variable de type "int") et grâce à un pourcentage, celle-ci sera remplie à un certains niveau et sera de couleur verte si le pourcentage est élevé et sera rouge si celui-ci est bas.

- Dialogues

Pour les dialogues, Léo Farhi a réutilisé le système qu'il avait créé pour d'anciens projets. Il l'a modifié pour que nous puissions lancer de l'audio en même temps que du texte, afin de rajouter des doublages audio. Ce système permet aussi de changer de langue, mais pour ce jeu, il sera disponible uniquement en français.

Son fonctionnement est simple : pour sauvegarder le texte, nous l'enregistrons au format JSON afin de sauvegarder d'une part les noms des personnages qui parlent et d'autre part le texte lui-même. Ensuite, le Dialogue Manager se charge de l'afficher tout en figeant le jeu.

Donc le code ci-dessous explique la procédure qui se passe quand le système lance un dialogue.

En premier lieu, le système affiche la boîte de dialogue en faisant l'animation d'ouverture, puis il fige le joueur et sauvegarde dans une "Queue" les phrases qui vont être dites. Puis, il attend que le joueur appuie sur E pour continuer (ou appuie directement sur le bouton affiché sur l'écran)

```

public void StartDialogue (DialogueValue dialogueValue, bool buttonNextAct = true)
{
    persistenceDataScene.mainCanvas.SetActive(true);
    this.buttonNextAct = buttonNextAct;
    if (ContinueButton.activeSelf!=buttonNextAct)
        ContinueButton.SetActive(buttonNextAct);
    persistenceDataScene.InterfaceIsOpen = -1;
    if (ActChoice)
        return;
    if (setting!=null){
        if (setting.Type.ToString() == "Choix" && animator.GetBool("IsOpen") == true)
            return;
    }
    if (boutonChoix[0].active)
        return;
    if (setting!=null){
        return;
    }
    if (player==null){
        player = GameObject.Find("Player");
        //$$playerMovement = player.GetComponent<PlayerMovement>();
    }
    setPosition = player.transform.position;
    if (dialogueValue.dialogue.figePlayer)
    {
        //$$playerMovement.FigePlayer();
    }
    bloquePlayer = true;
    if (dialogueValue.Type.ToString() == "Dialogue"){
        DialogueIsChoice(dialogueValue);
    }
    if (dialogueValue.Type.ToString() == "Choix"){
        animator.SetBool("IsOpen", false);
        ChoixIsChoice(dialogueValue);
        return;
    }
}

```

Les scripts des dialogues ont été faits par Léo Farhi. Ces scripts sont utiles, ils permettent de donner vie aux personnages. Cela les rend plus humains et plus attachants car ils ont, tout comme nous, des émotions et une personnalité propre.

## **IX. Le Site Web**

Ce site web est le même que celui présenté lors des précédentes soutenances dans la forme, mais bien entendu le fond a été modifié afin de correspondre à nos attentes artistiques et techniques. Les responsables du site web sont les mêmes que lors de la première soutenance, soit Geoffrey Elbaz pour le code et Léo Farhi pour le design. La page d'accueil comporte différents liens d'accès aux pages demandées sous forme d'en-tête, ce qui permet une présentation propre et efficace. Elle contient en plus un lien très visible vers la page de téléchargement pour inciter les utilisateurs à télécharger le jeu. Chaque page a la même présentation globale pour permettre une cohérence artistique et une utilisation plus intuitive.

Vous pouvez accéder au site web à l'aide de cet URL :  
<https://epitallhg.github.io/RegainTheWorldWebsite/>

## **X. Problèmes rencontrés**

- Entente du groupe

En général, comme nous sommes dans la même classe, il y a une bonne ambiance dans le groupe, mais à noter que le chef du projet met parfois un peu la pression afin de respecter au plus proche le tableau d'avancement.

- Réalisation

La réalisation du projet ne s'est pas faite sans rencontrer de problèmes. Par chance, aucun des membres du groupe n'est parti en S1# ce qui est un plus quand on connaît les conséquences que cela peut avoir de perdre un membre du groupe en plein milieu d'un projet.

Mais, en se basant sur des faits, il n'y a pas eu de gros problèmes au début du projet. C'est surtout au début du S2 que les choses se sont beaucoup plus compliquées pour l'ensemble du groupe.

Nous avons remarqué un manque d'efficacité chez certains et des erreurs de manipulation ont mené à la perte d'un assez lourd travail de l'un des créateurs (qui ne postait pas régulièrement son travail sur Github). Après concertation, ce manque d'efficacité est principalement lié à la pression subie par ceux n'ayant pas bien réussi leur S1 et qui doivent rattraper en validant le S2.

Au début du projet, nous avions l'intention de tout faire par nous-même. Cela prend en compte les modèles 3D des personnages et des bâtiments. Mais, comme nous l'avons dit un peu plus tôt dans la partie des modèles 3D, suite à un manque crucial de temps (à cause des cours à côté principalement), nous avons abandonné l'idée de créer tout nous-mêmes, et d'importer des assets ce qui nous a fait gagner beaucoup de

temps, notamment pour les bâtiments et les objets de décoration comme des fontaines ou des bancs pour les villes. Mais cette décision a aussi eu comme conséquence de rendre le travail de certains d'entre nous inutile. En effet, certains membres du groupe ont commencé à modéliser des bâtiments incomplets pour le moment. Ils avaient passé un temps non négligeable pour faire cela et leur travail a été réduit à néant pour le bien de tous et la réussite du projet.

Enfin, la réalité nous a ramené les pieds sur terre. Du fait de nos grandes ambitions pour ce projet, et du manque d'efficacité entre la soutenance 1 et 2, nous avons pris un retard non négligeable. Cela a donc forcé les membres du groupe à travailler 4 fois plus pour finir le projet dans les temps. Cela est d'autant plus difficile que des membres du groupe doivent passer les rattrapages de certaines matières pour passer au S3.

- Bugs et solutions

L'un des premiers bugs majeurs que nous avons rencontrés est qu'un des créateurs a perdu une bonne partie de son travail car il ne le postait pas régulièrement. En effet, ce qu'il avait créé a fini par être corrompu. La solution fut alors pour lui de charger une backup ce qui lui a fait malheureusement perdre plusieurs heures de travail.

Il y a aussi un bug avec le système de portail qui affiche mal l'eau quand on regarde à travers celui-ci. Cela est dû à un problème de rendu graphique.

Les bugs d'animation ont été certainement les bugs les plus amusants à voir (comme lorsqu'une IA attaque, elle glisse en même temps sur le sol ou encore des roulades dans le vide etc...). Mais ces derniers ont été aussi les plus difficiles à régler. En effet, comme nous ne sommes pas très habitués aux systèmes et propriétés des animations, trouver des solutions à ces problèmes nous a pris beaucoup de temps et de ressources pour un résultat qui n'est pas forcément de haute qualité.

Enfin les bugs avec les IA sont assez nombreux. Par exemple, l'IA peut prendre beaucoup de vitesse et commencer à glisser sur le sol extrêmement rapidement. Après recherche, ce bug est causé par les IA qui se déplacent sur le "navmesh" tout en étant affectées par la gravité. Le problème vient du "navmesh" qui n'est pas en contact avec le sol. Par conséquent, si une IA marche sur un terrain en pente, elle va glisser sur le sol et comme elle ne touche pas le sol, elle va prendre aussi beaucoup de vitesse. Pour régler ce problème, nous avons donc augmenté la hauteur du terrain pour que l'IA touche le sol. Une autre alternative est d'augmenter le "capsule collider" de l'IA vers le bas pour que celui-ci touche le sol.

Un autre bug majeur avec les IA est un bug permettant de créer un OOB (Out of bound). Cela permet au joueur de se promener partout sur le terrain en passant à travers toutes les textures et donc de finir le jeu plus rapidement en esquivant les parties difficiles ou longues en prenant des raccourcis. Ce bug est provoqué par la fonctionnalité qui permet au joueur d'échanger sa place avec une IA. Comme expliqué précédemment, le joueur peut échanger sa place avec une IA et une IA peut se téléporter auprès du joueur si elle se situe à une certaine distance du joueur. On peut donc à certains endroits de la carte, au niveau d'une zone inaccessible, faire en sorte que l'IA se téléporte vers le joueur mais à l'extérieur de la zone de jeu et à ce moment précis. Le joueur échange sa place avec celle de l'IA. Il se retrouve donc à l'extérieur de la zone jouable. Il a donc créé un OOB.

L'un des bugs que nous avons rencontrés était que nous pouvions ouvrir l'inventaire lorsqu'une cinématique était jouée.

## XI. Récit de la réalisation

- Ses joies

Au début du projet, l'ensemble du groupe était très enthousiaste à l'idée de faire un jeu d'une telle ampleur. Il y avait une réunion par semaine pour mettre à jour tout ce qui a été fait sur le projet au cours de la semaine et tout le monde était présent. Nous arrivions à la fois à faire le projet et les cours (dont les TP qui prenaient du temps). Cela montre à quel point nous étions motivés par ce projet.

De plus, il était fort intéressant de pouvoir tout faire en autonomie (pour la première fois pour certains membres du groupe).

Mais aussi, ce projet nous a permis d'avoir une idée de ce qu'est le travail en groupe.

“Enfin il est important de souligner que même sous ses airs de dictateur, notre chef de projet a su nous écouter. En effet, Léo Farhi a été un chef de projet plus qu'exemplaire qui a su nous motiver dans les moments difficiles, nous aider grâce à son expérience et nous écouter.” : écrit par Léo ALBA SANCHEZ .

- Ses peines

Pour une partie du groupe, les peines sont apparues après le premier partielle, donc au début du S2. La charge de travail est devenue beaucoup plus importante après le passage au S2. Cette charge s'est vue surtout dans le module d'IP. En effet, c'est la période où des QCM de programmation sont apparus et les TP sont devenus beaucoup plus difficiles et plus longs. Il était beaucoup plus dur de trouver du temps pour faire le projet. Et quand bien même, nous trouvions du temps, l'enthousiasme du début d'année était moins présent. La fatigue accumulée ne nous aidait pas...

De plus, c'est aussi à cette période que s'est faite la première soutenance. Par conséquent, nous devions écrire le fameux rapport de soutenance. Cela a été une chose très difficile.

Par ailleurs, la première soutenance s'est très mal passée dans notre groupe, ce qui a changé la vision du projet de la part de certains membres du groupe. A partir de ce moment, ces personnes voyaient le projet comme une corvée, que cela prenait du temps sur le peu de temps libre que nous avions et que pendant les seules "vacances" accordées, il fallait rédiger un rapport de 20 pages minimum et 50 pages pour la dernière soutenance. Ce qui a rendu difficile ce projet c'est que le groupe est devenu beaucoup moins efficace dans la réalisation des différentes tâches. Il en résulte un retard qui n'est pas négligeable. Comme il faut rattraper ce retard, il fallait beaucoup travailler sur le projet. Cela a été bien dur.

Outre les peines que nous avons rencontrées au cours de l'année sur le projet dans sa globalité, un autre facteur a été très dur à gérer et ce facteur est la distance. En effet, à cause de la crise du COVID-19, nous n'avons jamais pu travailler ensemble en présentiel. Les seuls moments où nous étions réunis pour faire le projet étaient les vocales sur discord. Par conséquent, l'efficacité de l'ensemble du groupe pour travailler en était affaiblie. Par exemple, quand un membre ne savait pas comment résoudre un problème, il pouvait attendre peut-être toute la journée avant d'avoir sa réponse, ce qui n'est pas le cas en présentiel.

## XII. Conclusion

- L'objet de l'étude a-t-il été accompli ?

Somme toute, après concertation des membres du groupe et en se basant sur ce qui a été fait au cours de ce projet, nous pensons sincèrement que l'objet d'étude est accompli. Nous avons réussi à créer un jeu de rôle en seulement 5-6 mois. Nous avons dans le cas général, respecté le tableau des tâches ainsi que la répartition des tâches. Avoir respecté cela montre que les membres du groupe se sont fait confiance du début à la fin. La preuve, il n'y a pas eu de mésentente au sein du groupe. Même si le travail fut extrêmement difficile par moments, surtout vers la fin de l'année, nous avons respecté l'organisation que nous nous sommes imposée pour mener à bien le projet.

- Ce que nous a appris ce projet

Outre l'accomplissement du projet, il y a eu beaucoup de hauts et de bas comme vous l'avez remarqué un peu plus haut dans ce rapport. Mais regardons le bon côté des choses, nous allons apprendre de nos erreurs dans ce projet pour éviter de les refaire dans les projets futurs.

Une de nos grosses erreurs, et c'est certainement celle qui nous a fait perdre beaucoup de temps, est le fait que nous n'avons pas travaillé le projet de façon continue.

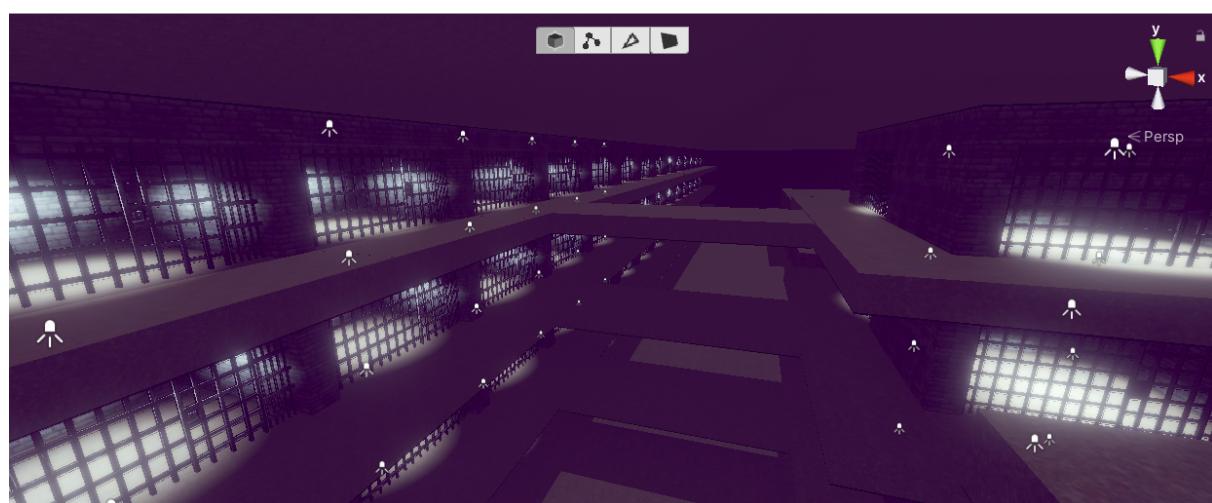
En effet, comme expliqué un peu plus haut, au début, nous travaillions continuellement parallèlement au cours. Mais quand cela s'est arrêté, nous avons accumulé beaucoup de retard que nous avons dû rattraper pendant les vacances. La leçon à tirer de cela est le travail en continu, même s'il est difficile de trouver le temps, sinon les conséquences peuvent être catastrophiques.

Une autre erreur, et elle est liée à la précédente, est qu'il faut garder les pieds sur terre, ne pas viser trop haut dès le début. Nous savons qu'avoir de l'ambition est toujours un plus dans la vie, mais il ne faut pas trop l'être. Pour la plupart des membres du groupe, c'est la première fois qu'ils codent de façon autonome et il est donc difficile de mélanger l'apprentissage de la programmation et la création d'un grand projet. Il vaut mieux viser un peu plus bas au début.

### XIII. Annexes

- Images du jeu (Version Éditeur)

- Prison Intérieure

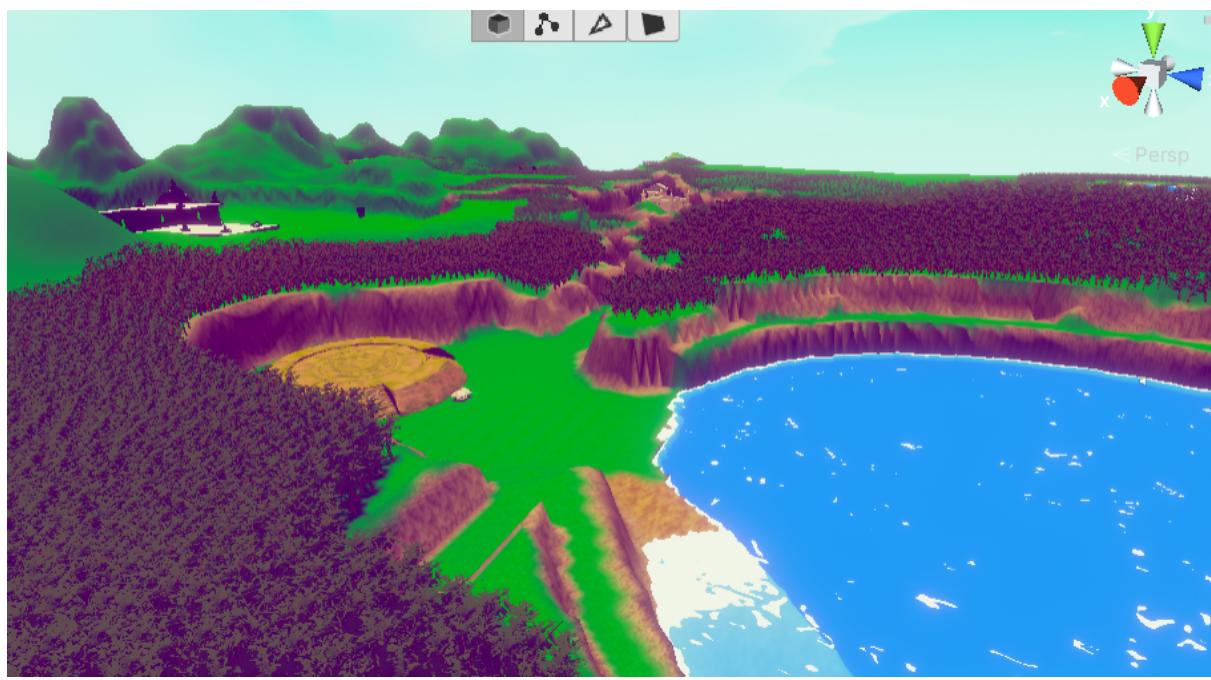


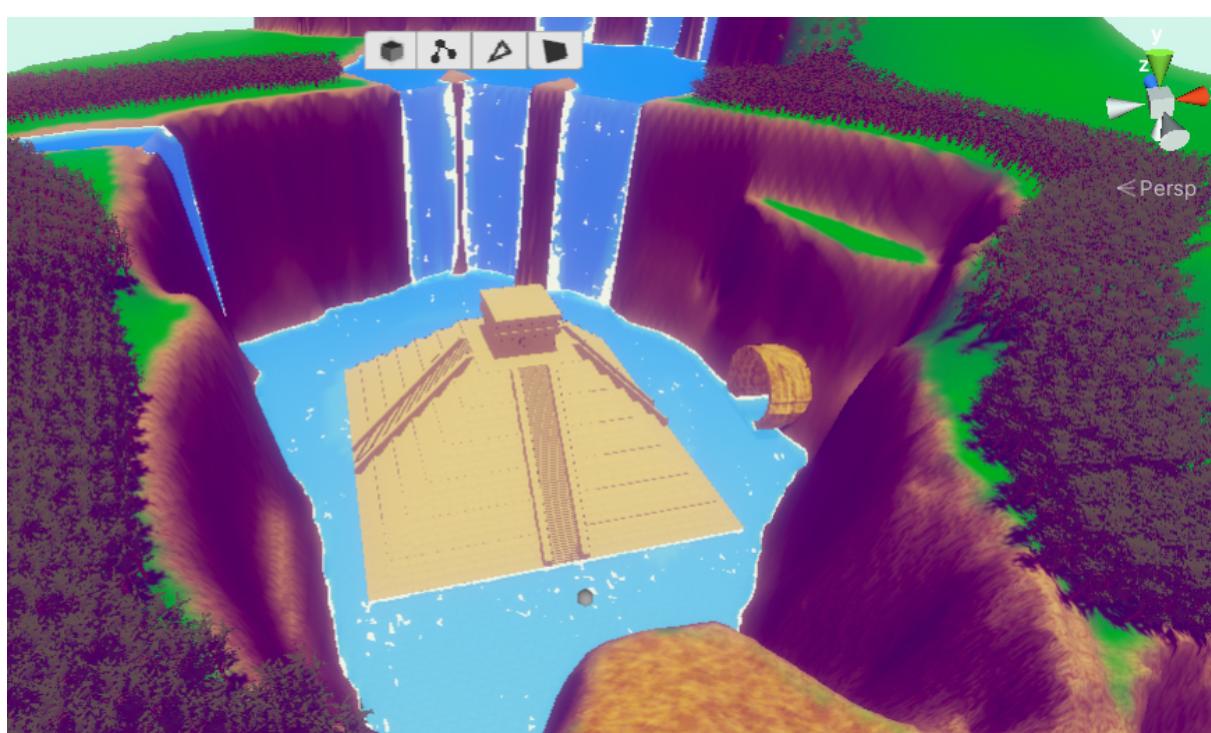
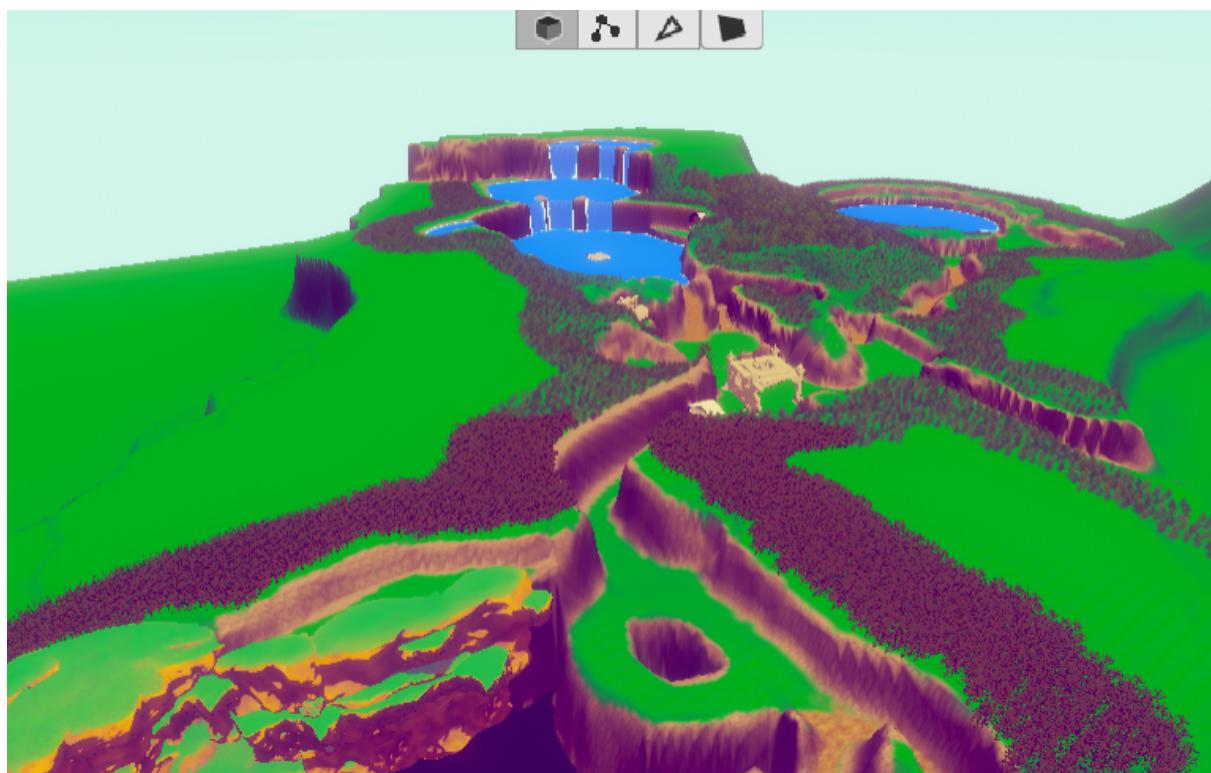
- Prison Extérieure

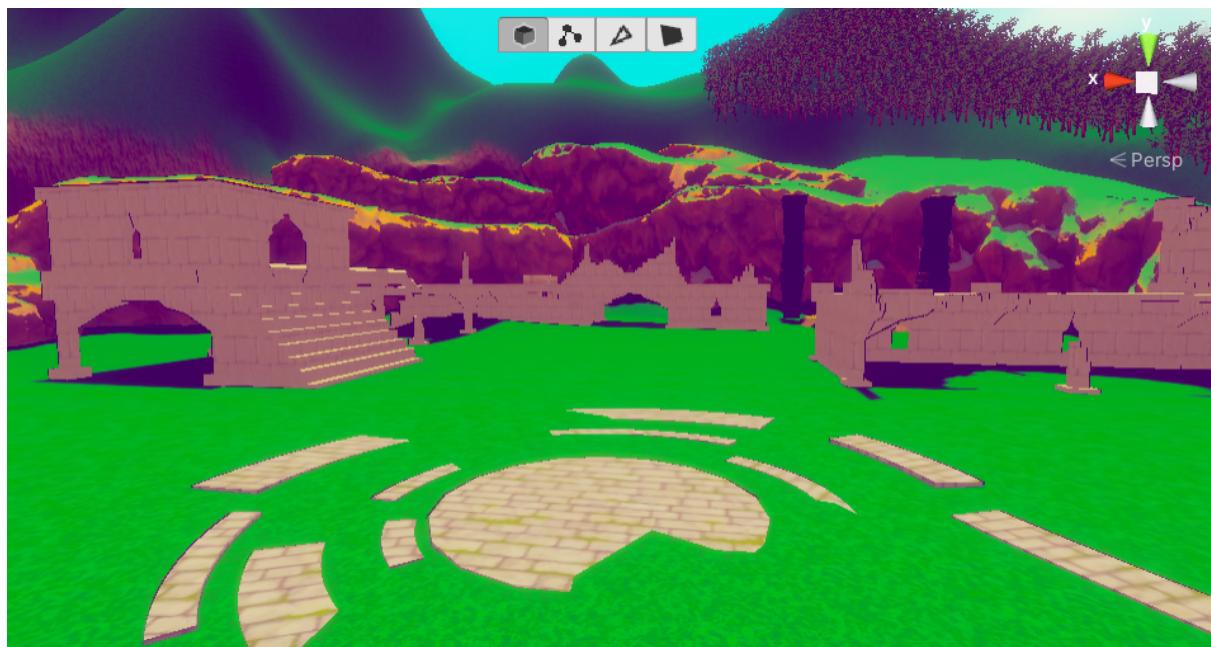


- Jungle





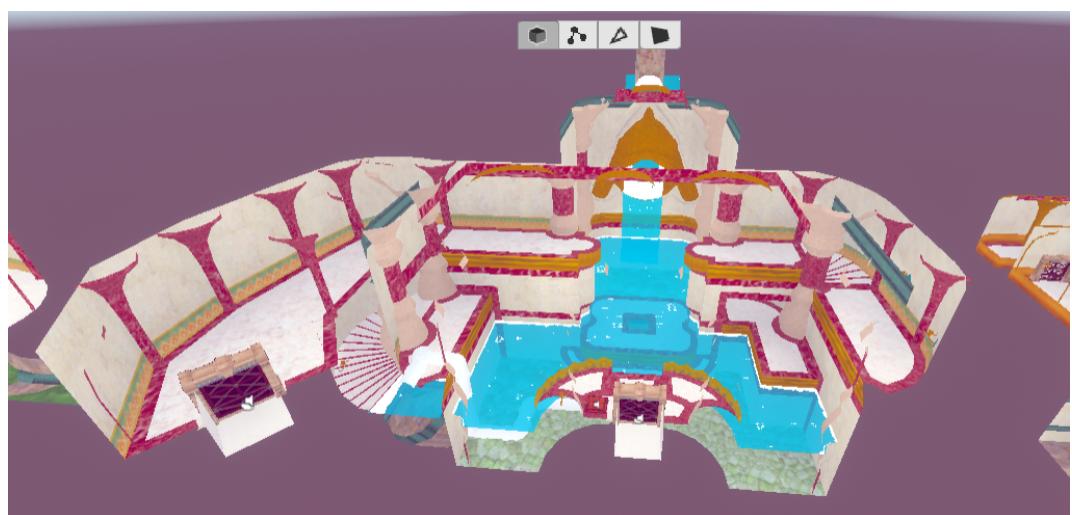




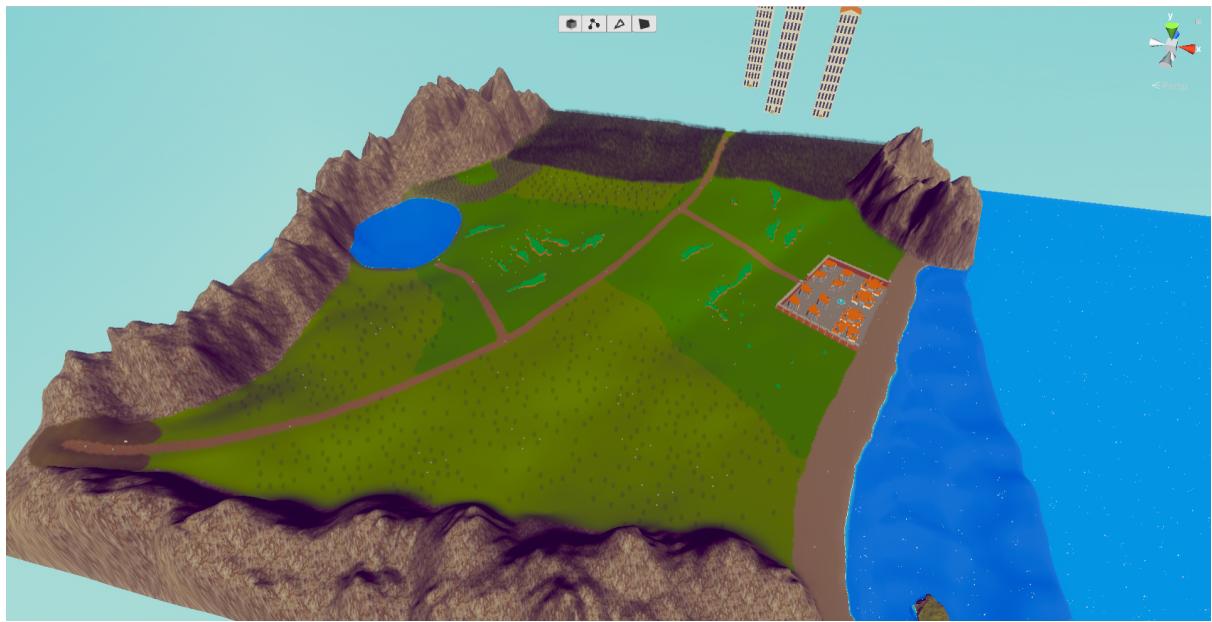
- Ville n°1



- Temple aquatique

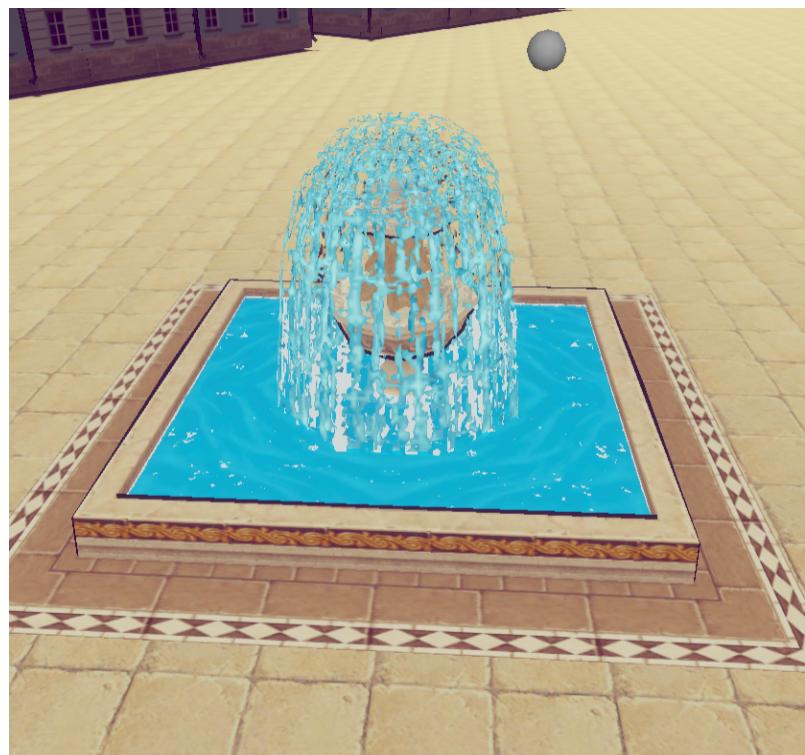


- Plaine

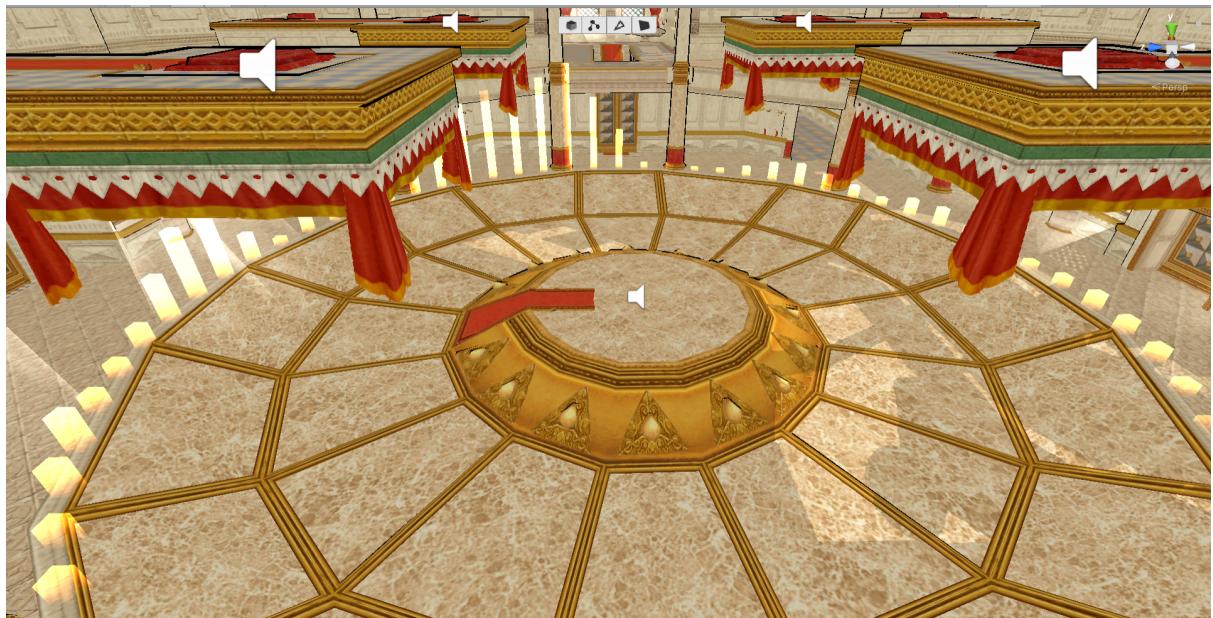


- Ville n°2

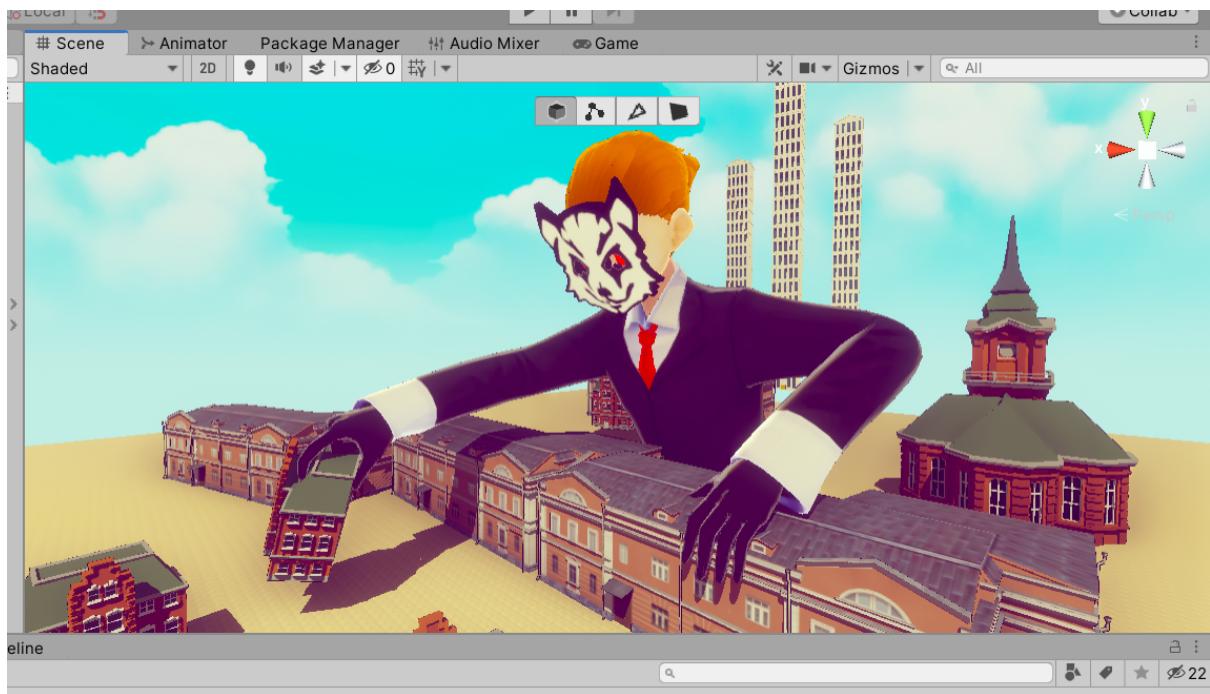




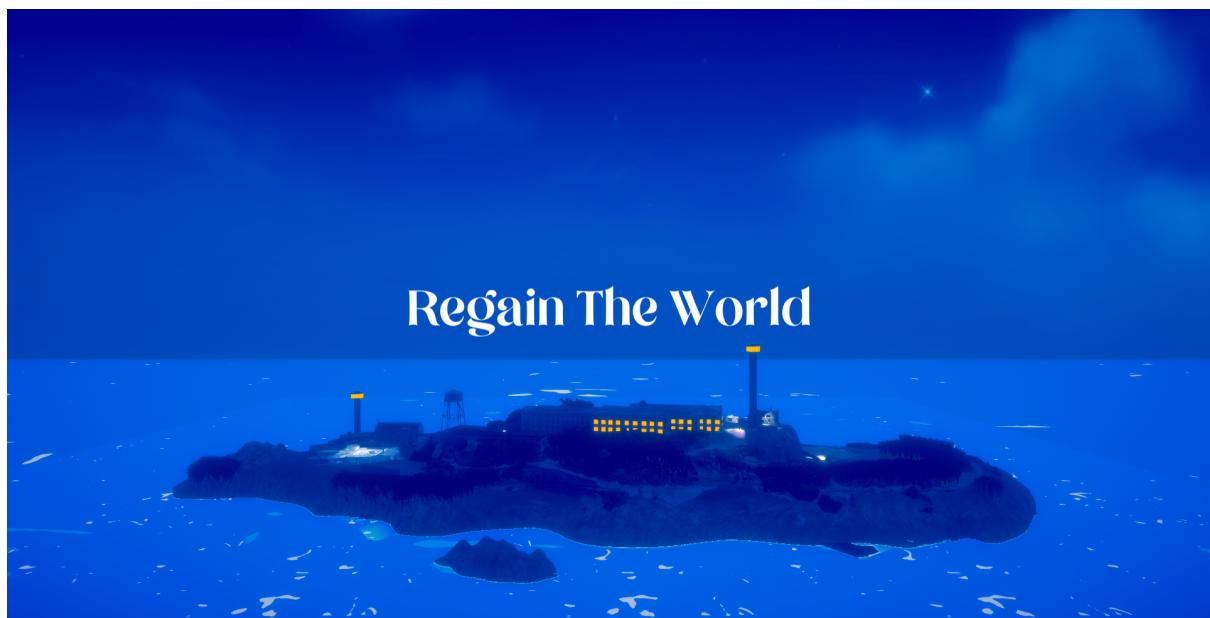
- Intérieur de la Tour de Giovano

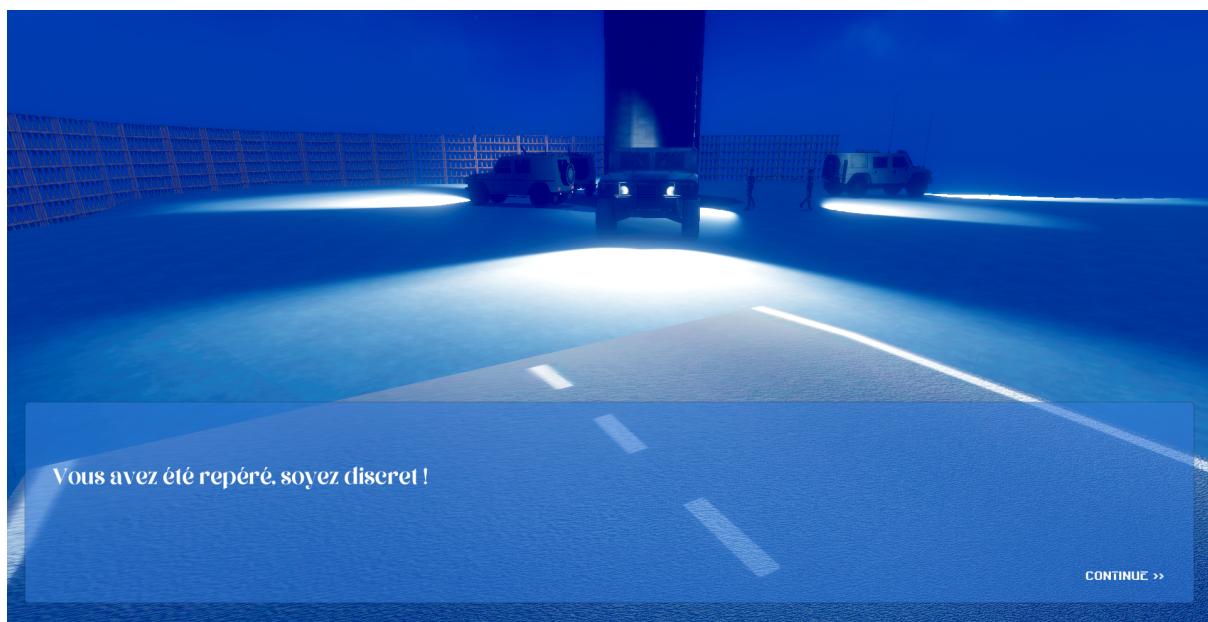


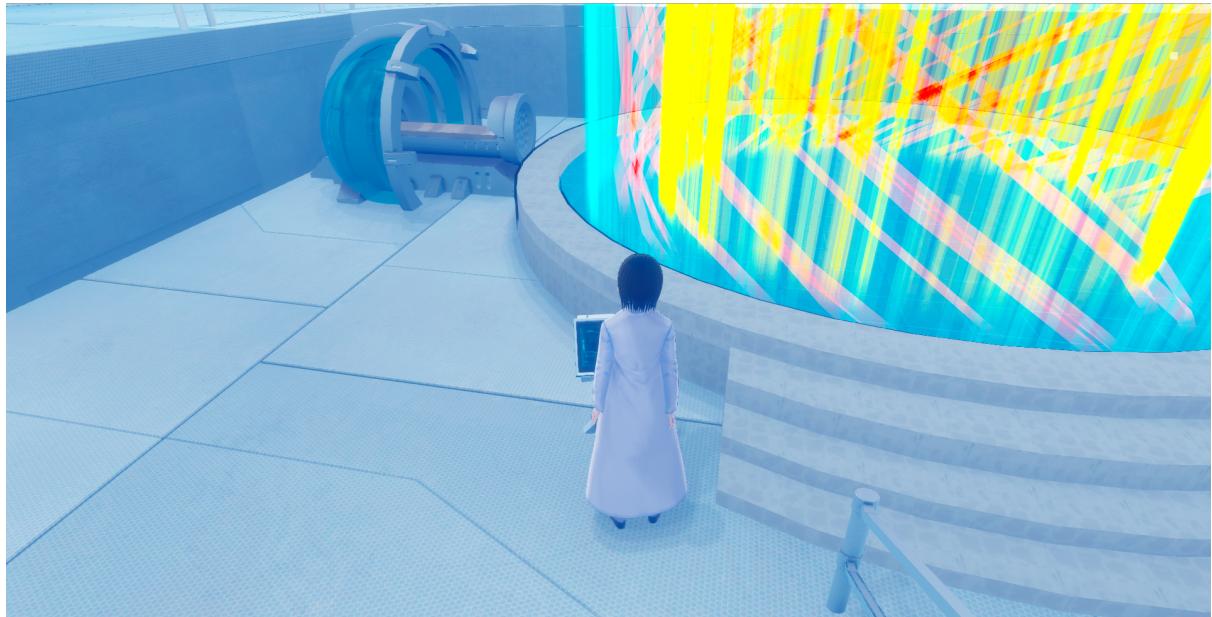
- Scène utilisée pour la jaquette



- Images du jeu (Version Compilée)



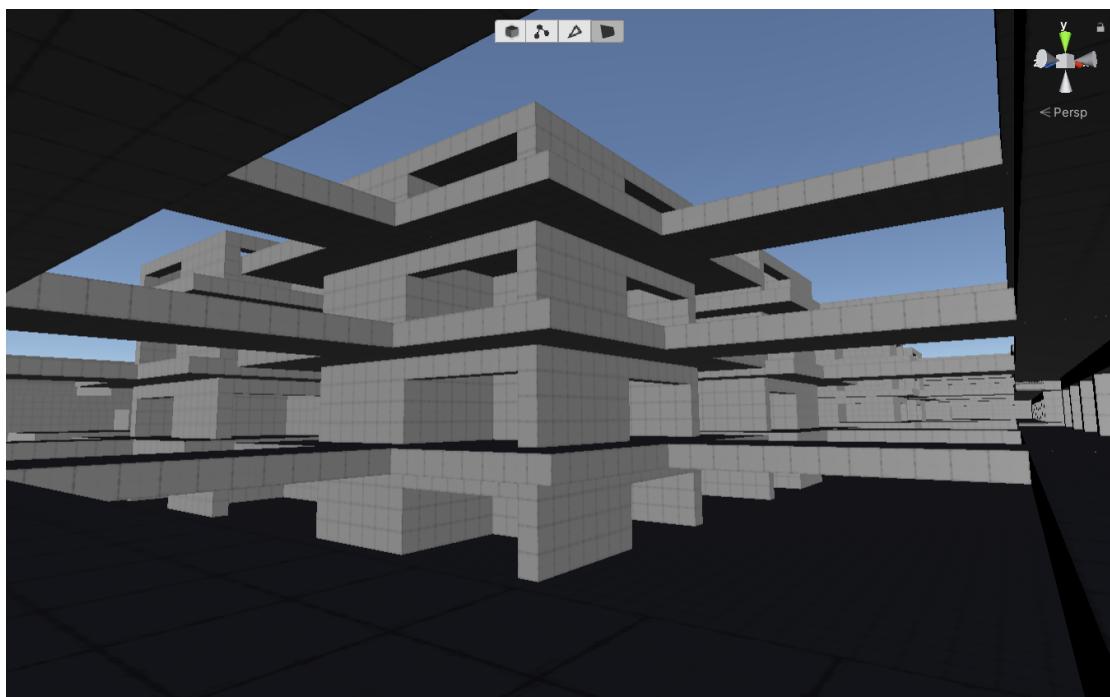


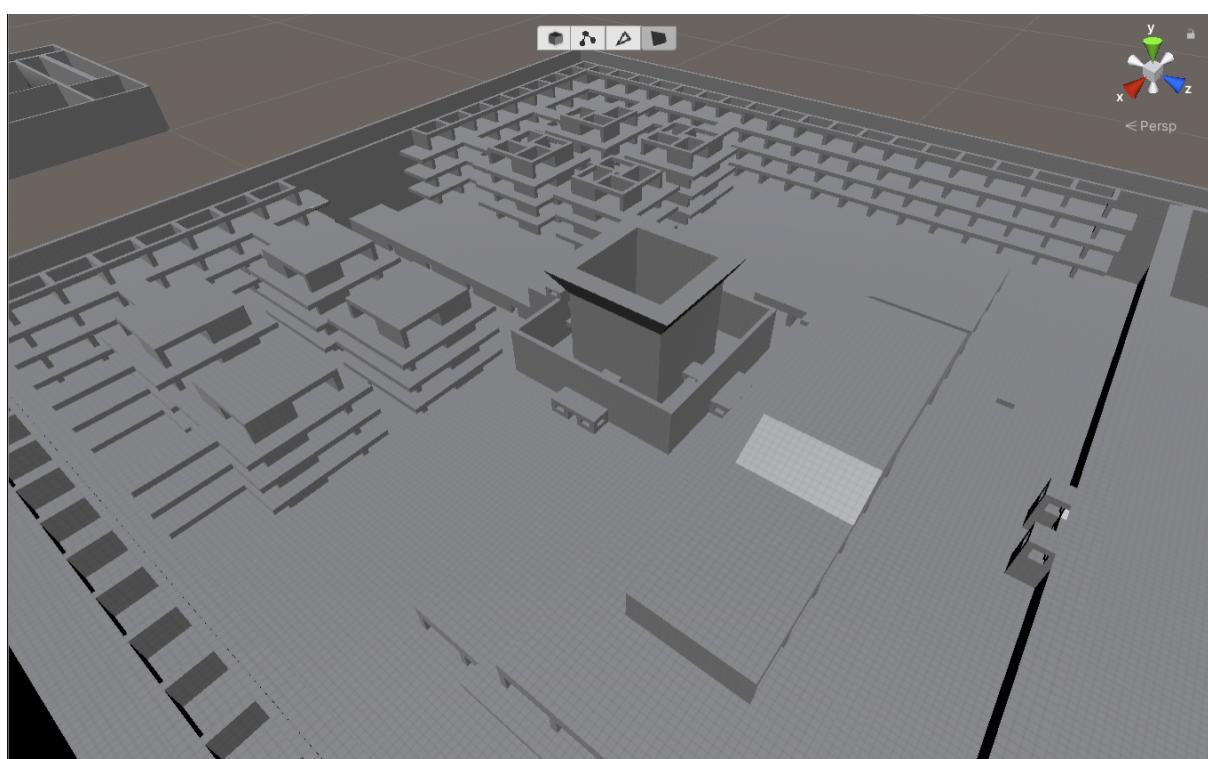
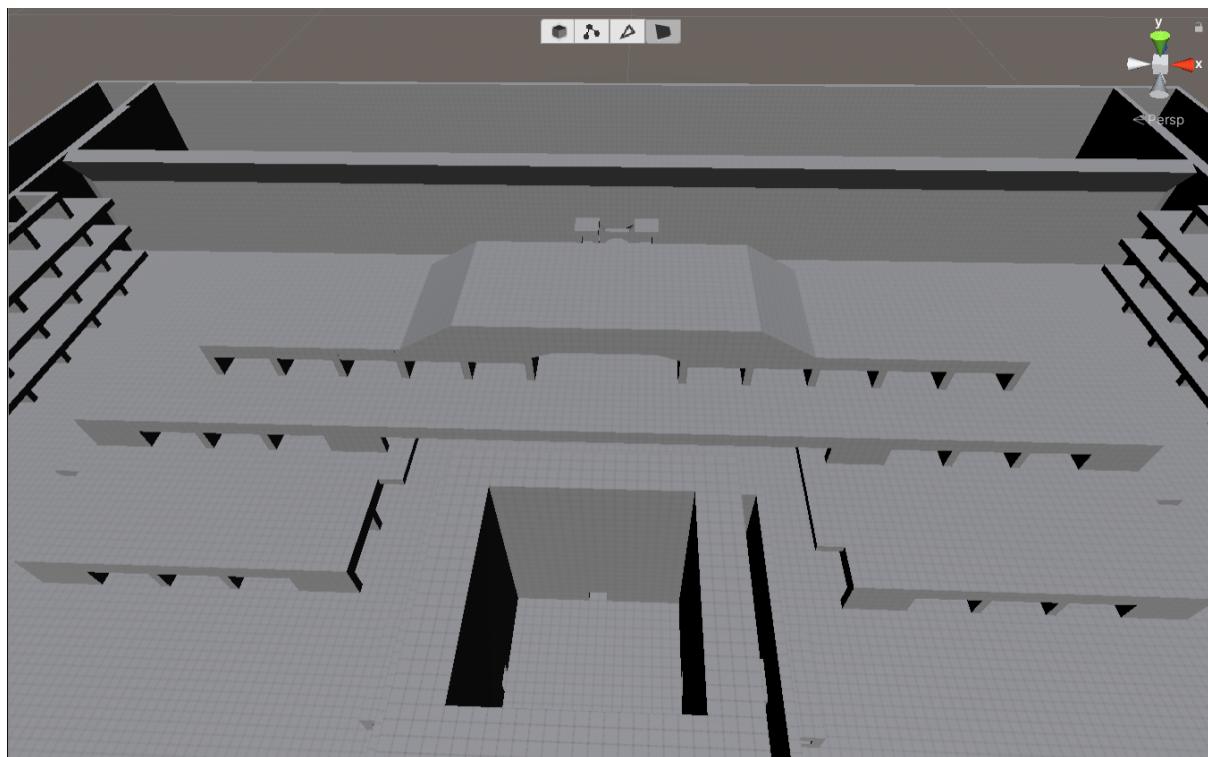


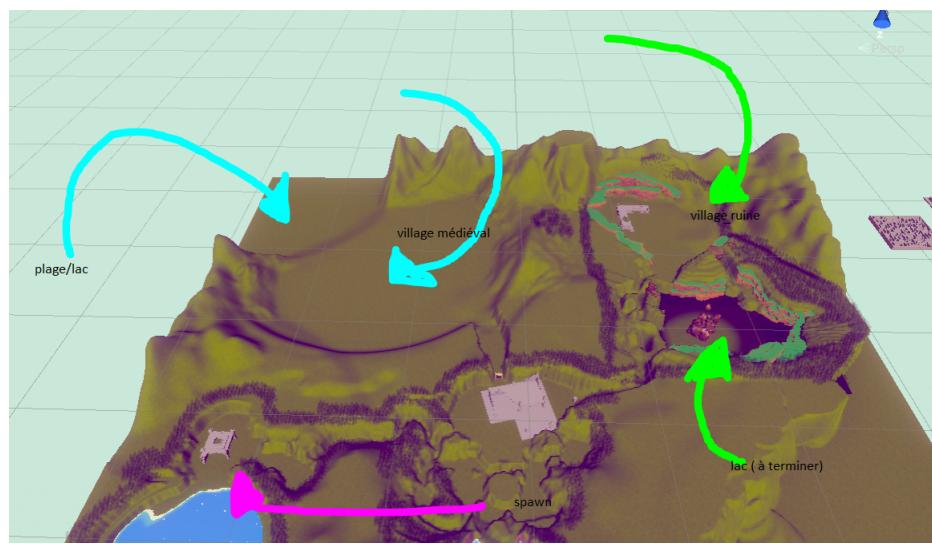
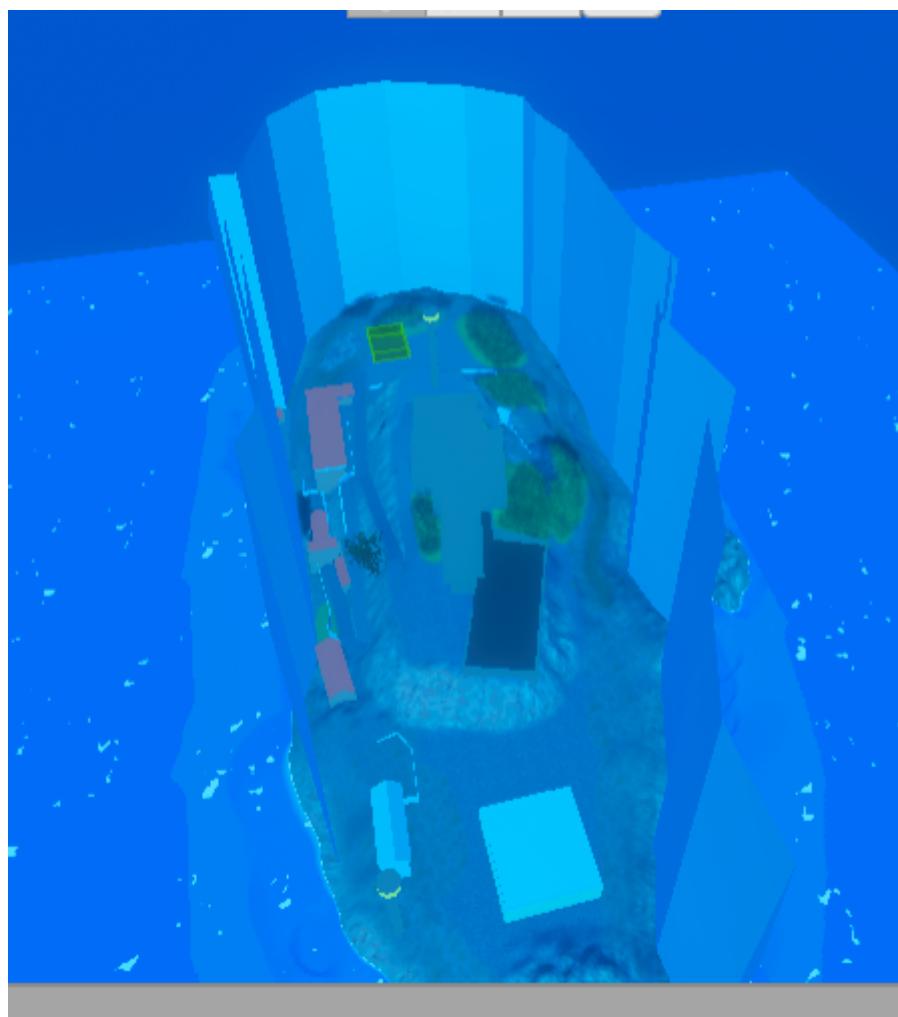


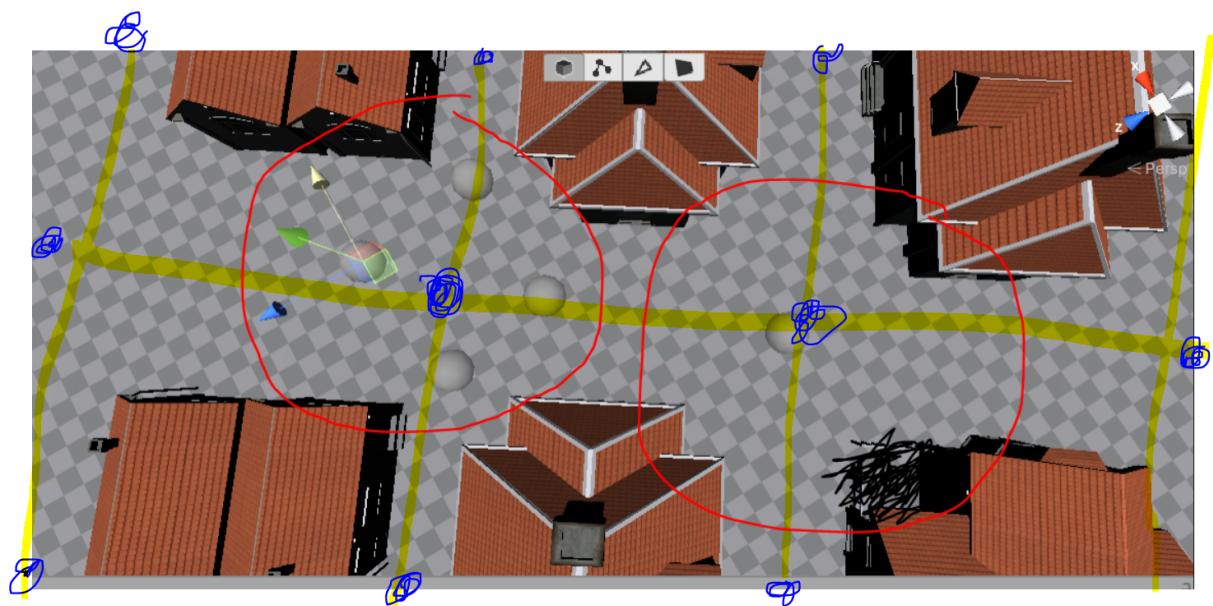
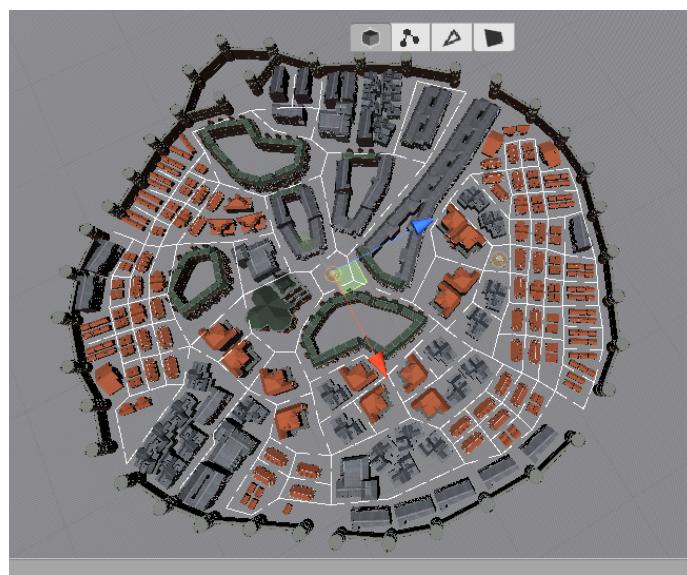
- **Les Croquis et Prototype**

(Des différences peuvent être présentes entre les croquis et le jeu final)

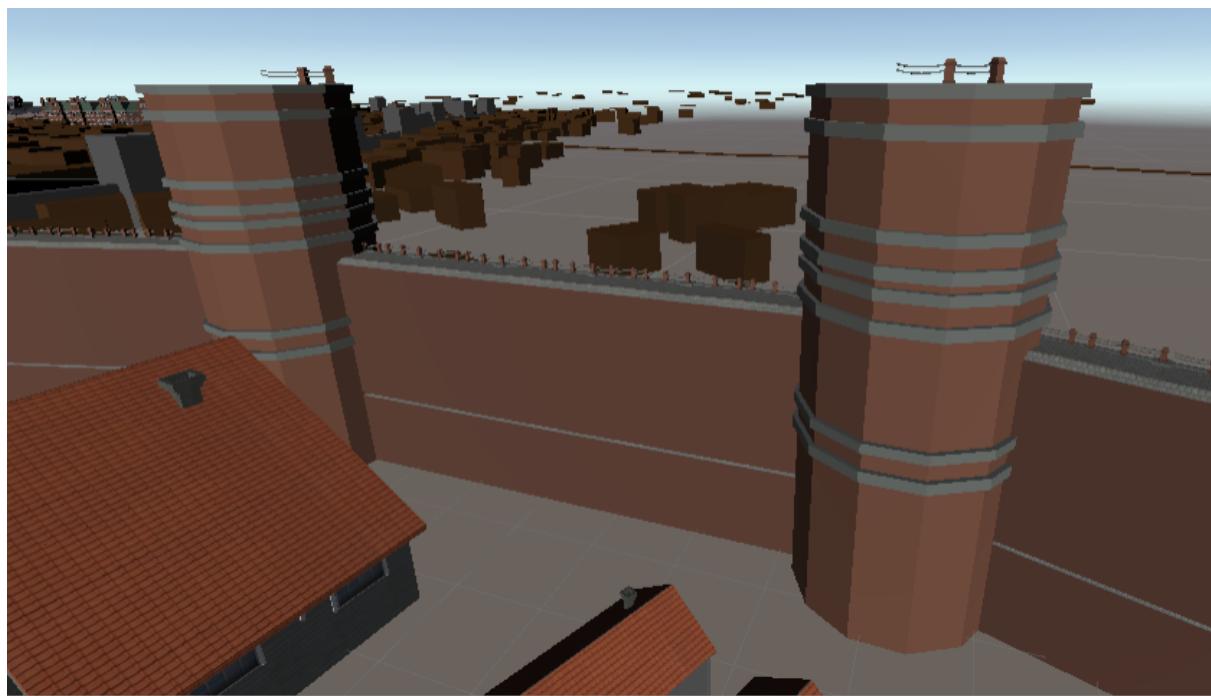


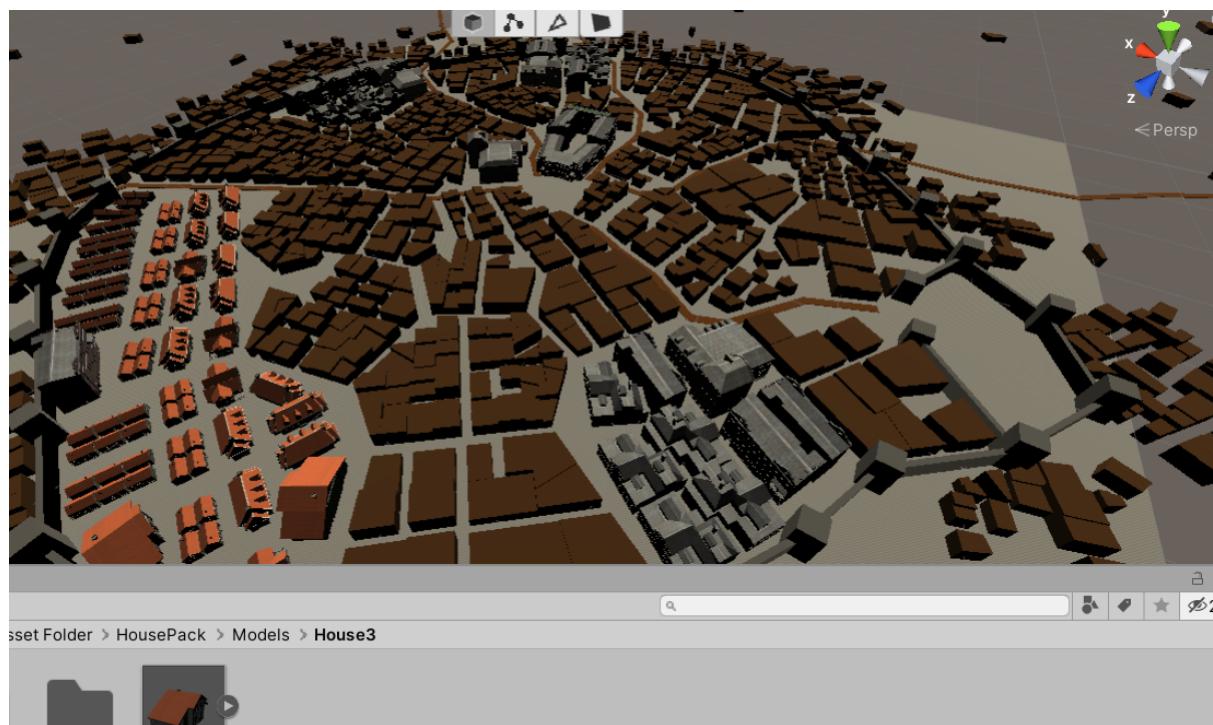


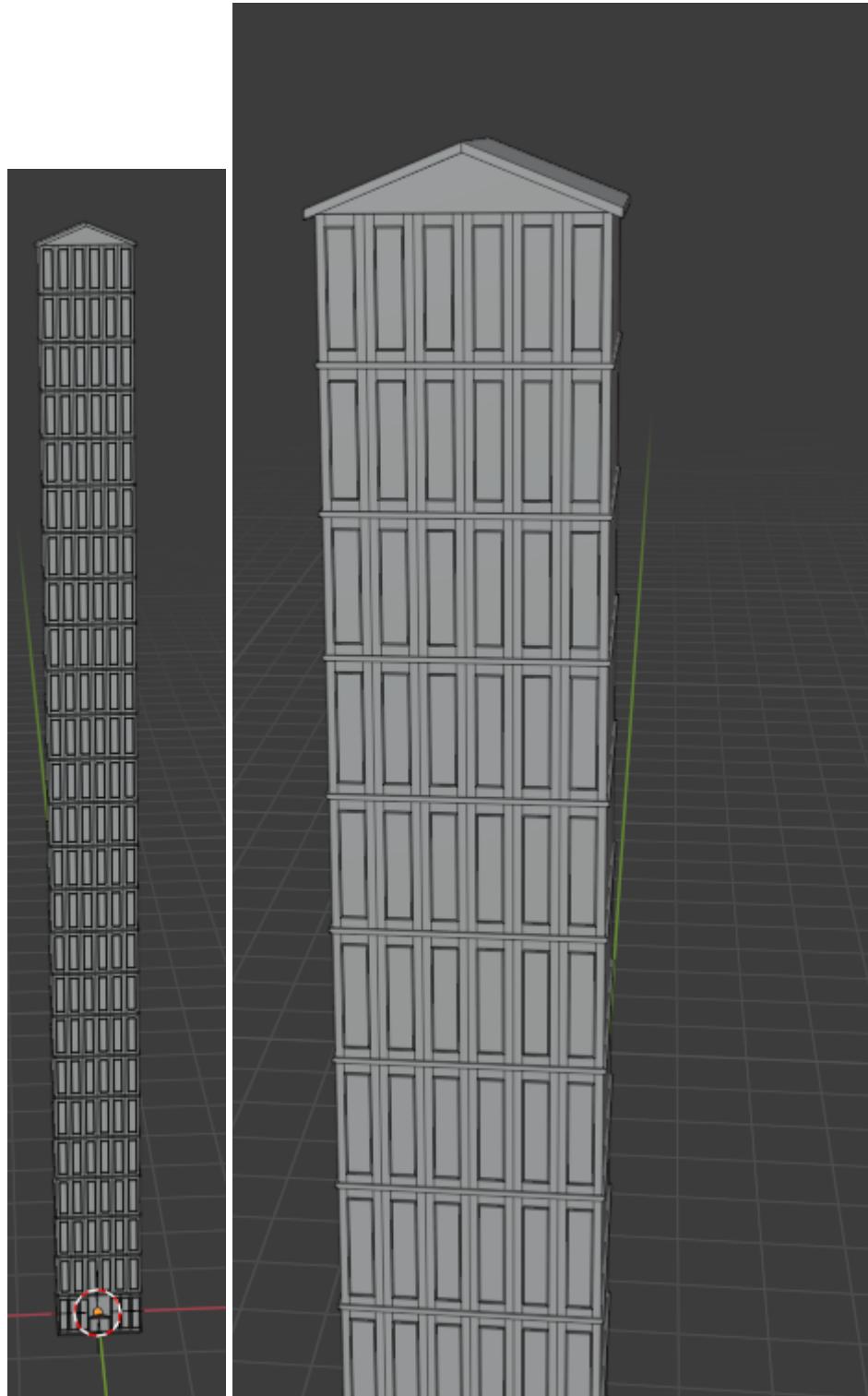


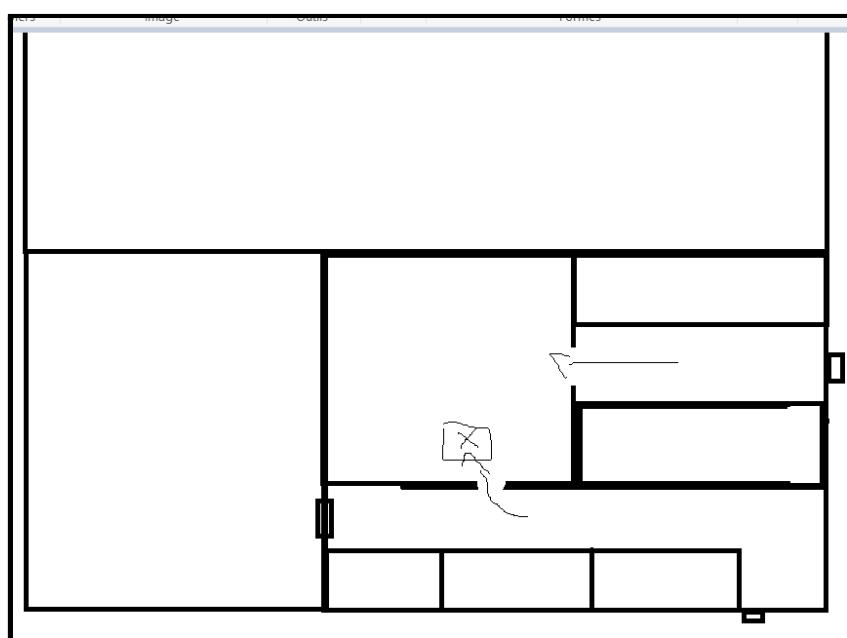
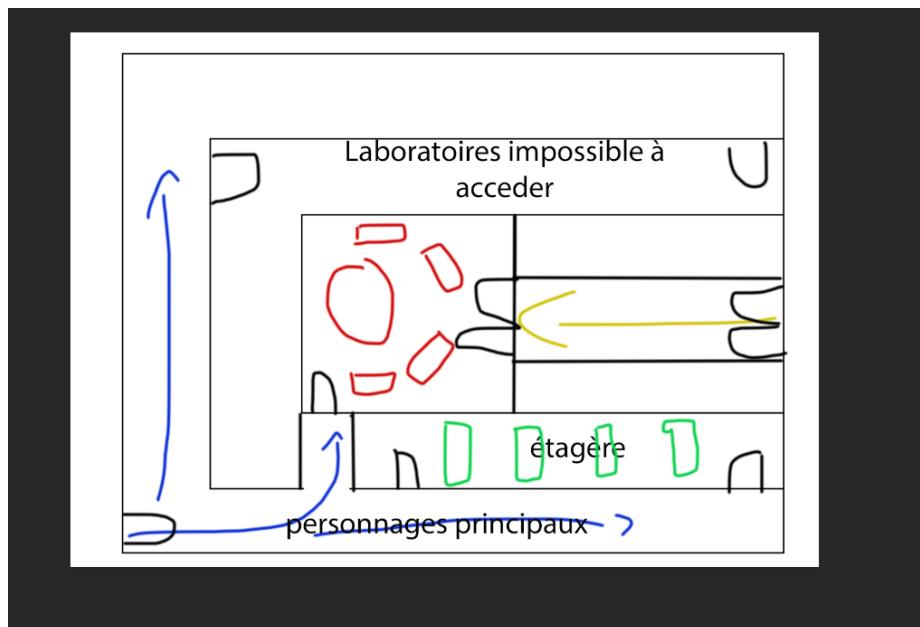


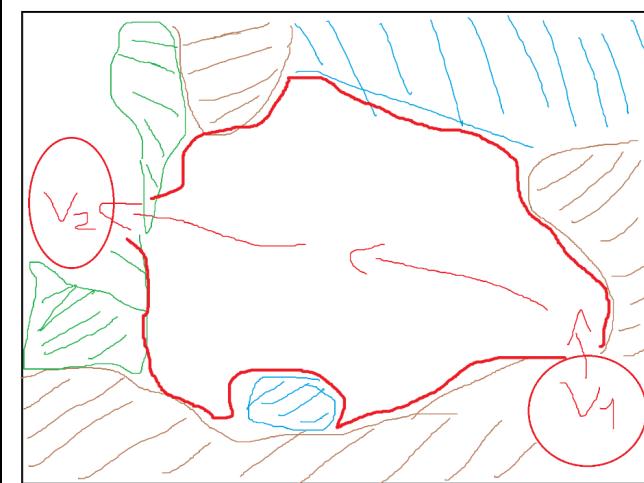












 eau (mer, lac)

 délimitation

 hache (Montagnes d'i)

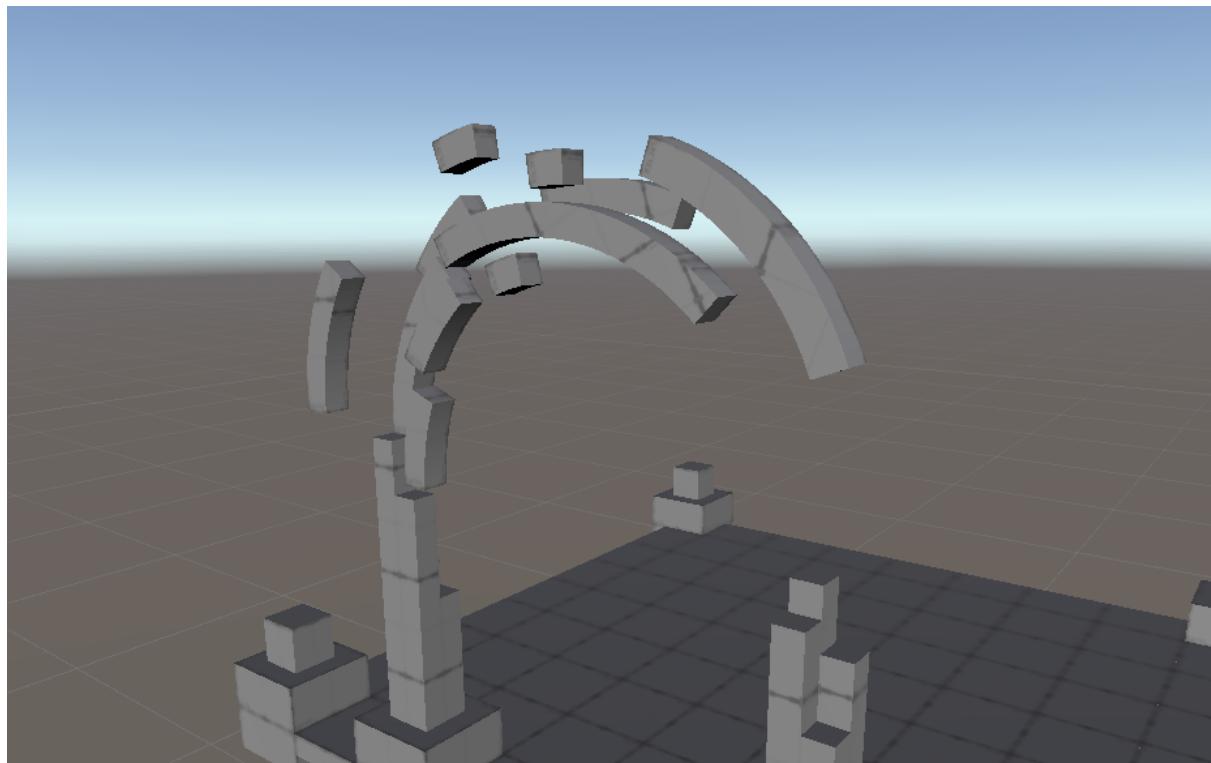
 foret

 villes

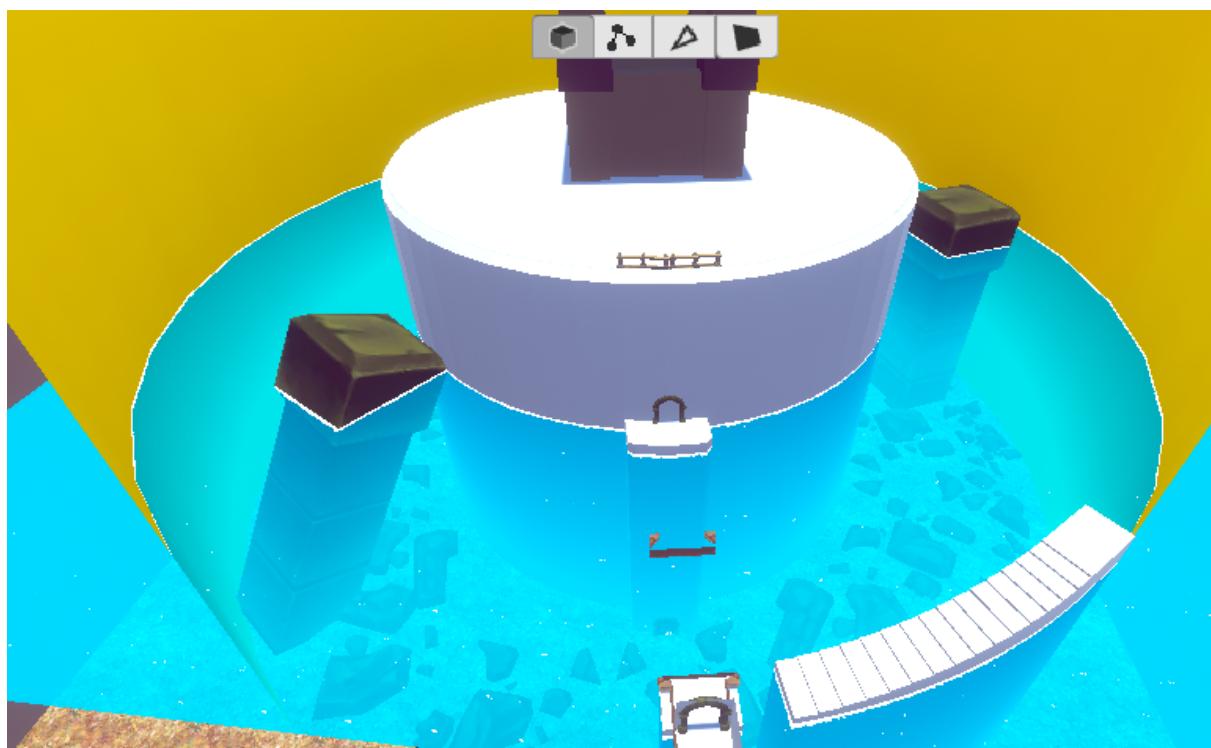
Première version du logo



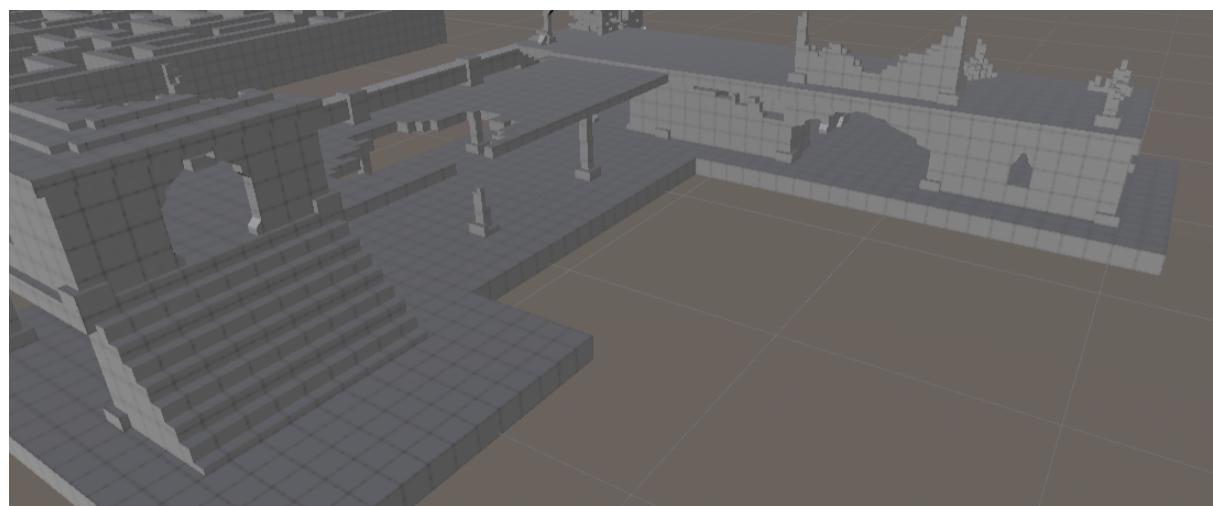
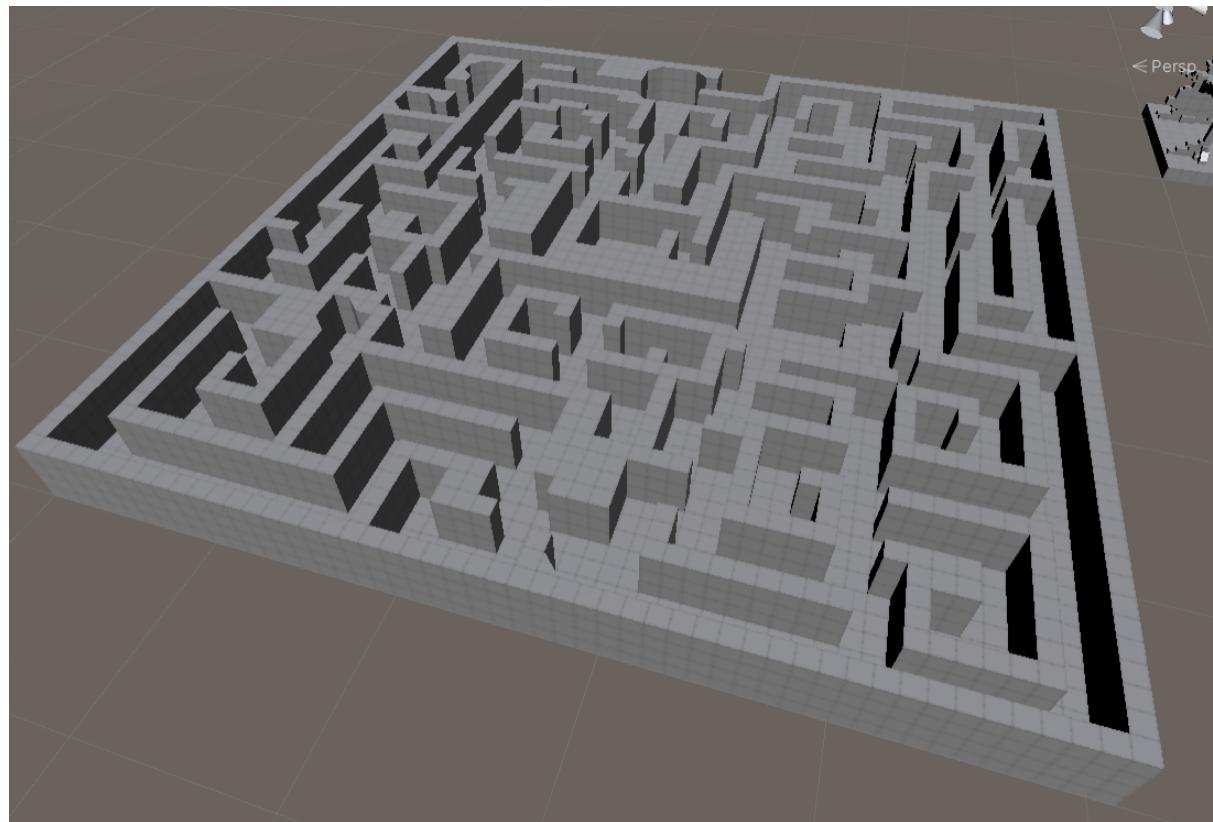
Prototype de Ruine

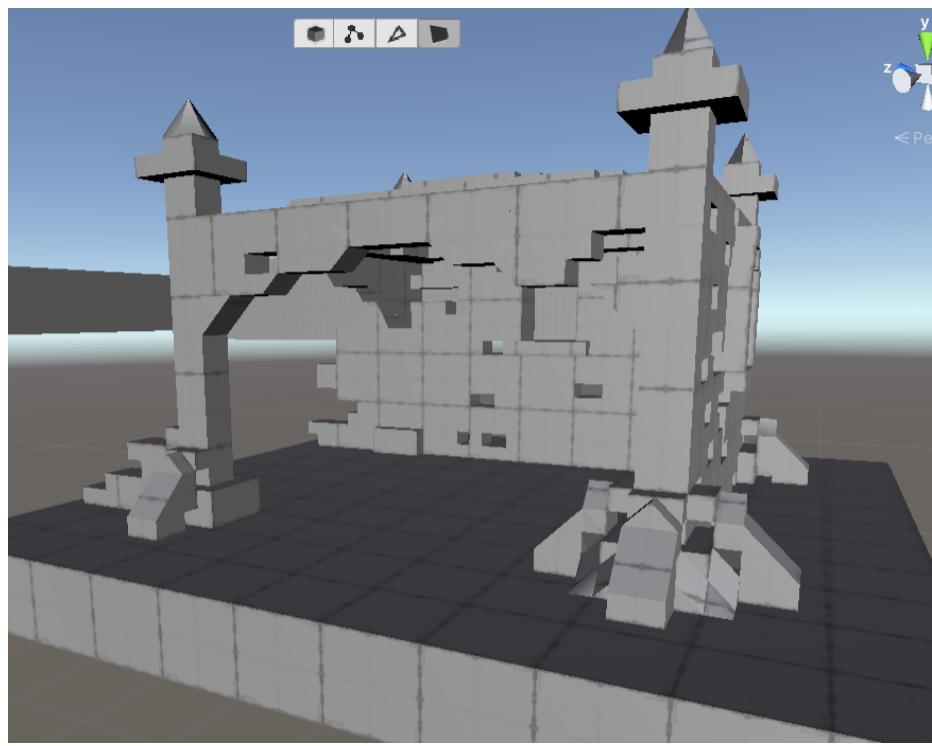


Prototype du temple aquatique



## Prototype Labyrinthe





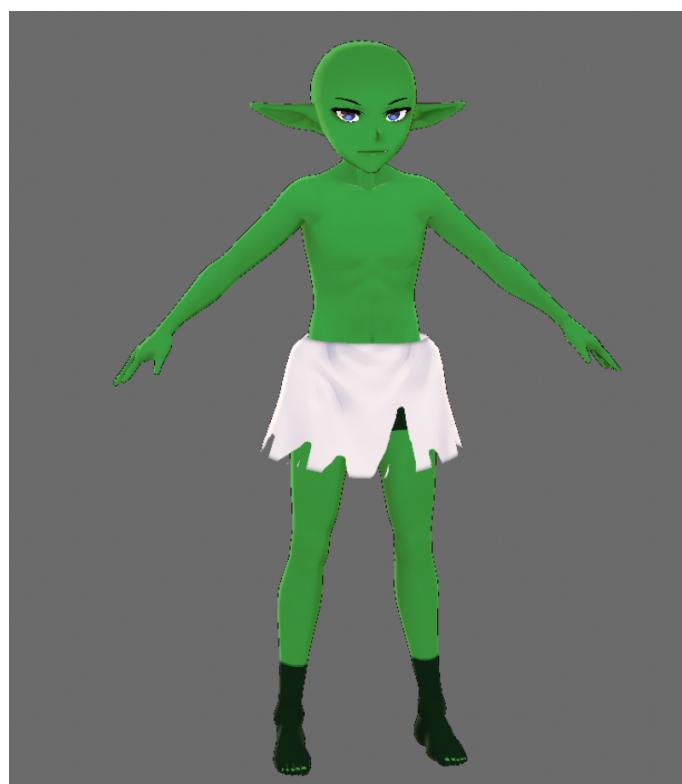
Boss n°1



Prototype de Xaya



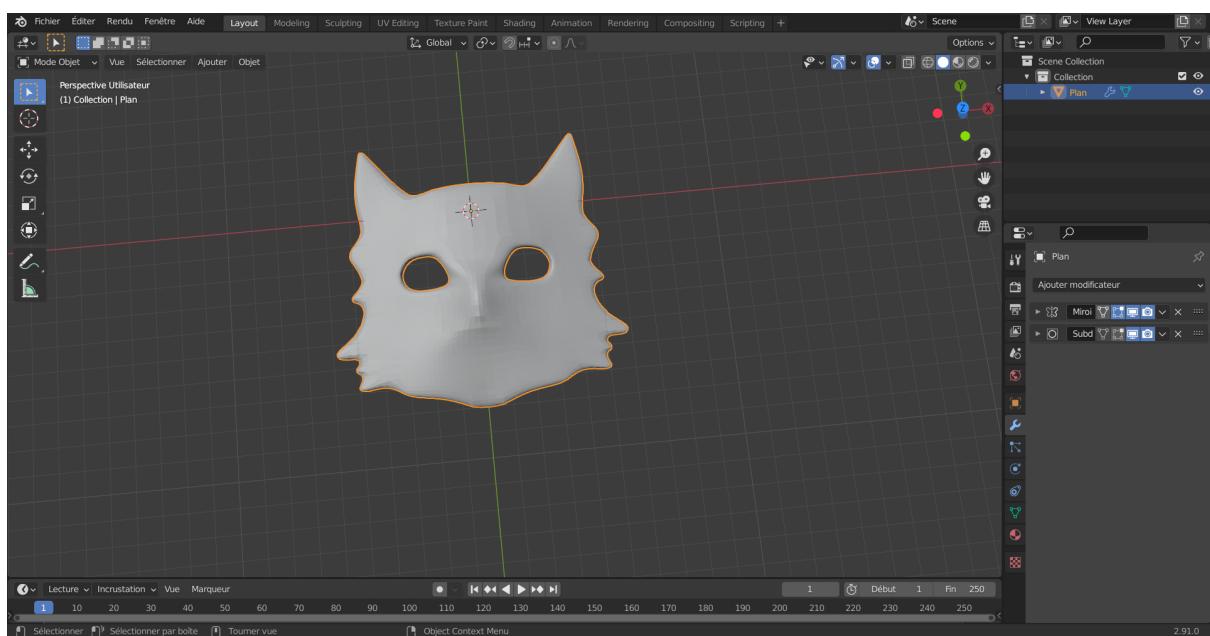
Gobelin :



Militaire :



Prototype Masque



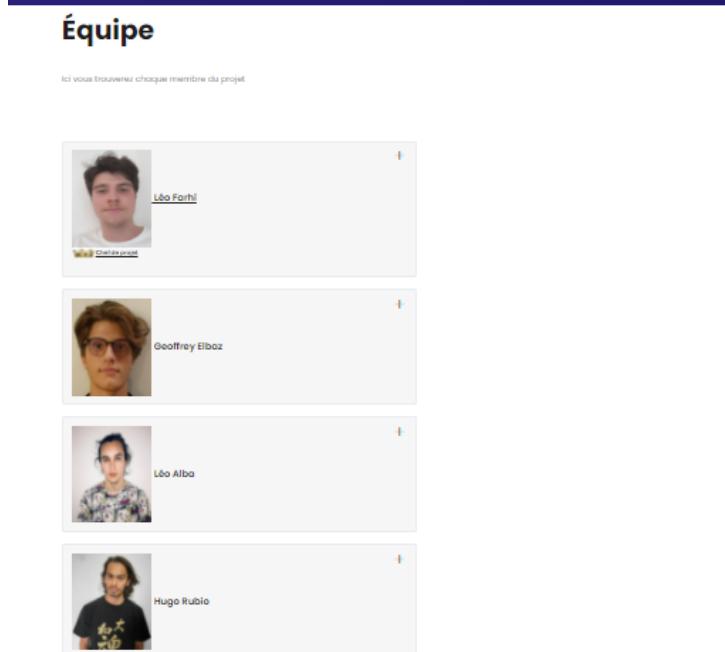
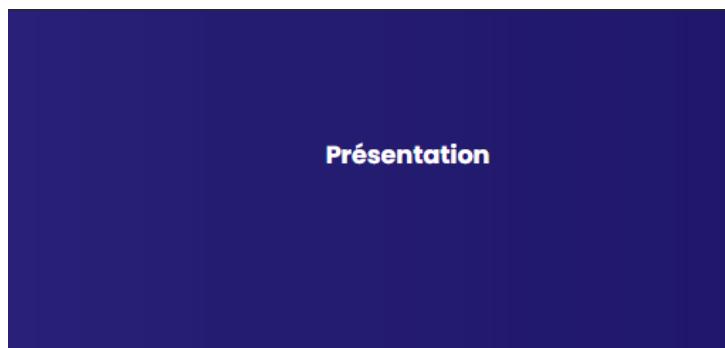
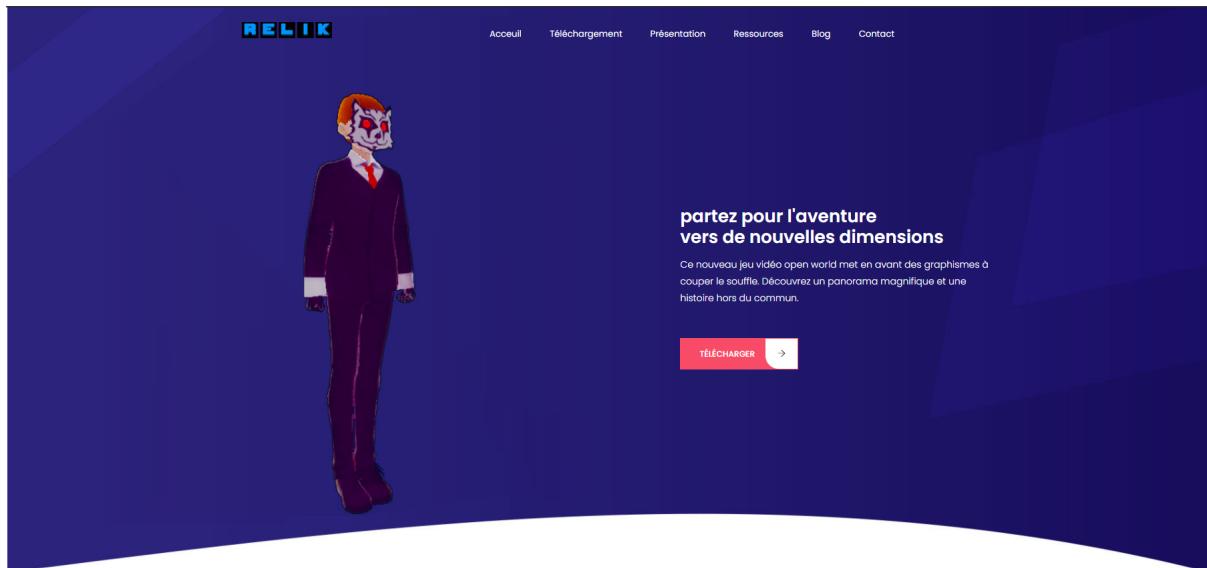
## Prototype de Giovano



## Coupe de cheveux de Giovano



- Site Web



## Historique

Ici, vous trouverez l'historique de notre projet.

**4 Dec 2020 : Léo F**

Création Du projet.  
Ajout d'Asset : Standard Asset, Nature Kit, Skybox, etc.

**13 Dec : Léo F**

Ajout du système de dialogue, et de sauvegarde.

**18 Dec : Léo F**

Création de la prison extérieur.

**18 Dec : Geoffrey**

Création du site Web.

**19 Dec : Léo F**

Création du site Web.