

Rapport de Projet

The Maze VR

S-TEAM

3 juin 2019



Martin Camincher Lucien Leseigle
Geoffroy du Mesnil du Buisson Hugo Belot-Deloro

Table des matières

1	Introduction	5
2	Cahier des charges	5
2.1	Introduction	5
2.1.1	Présentation du groupe	6
2.1.2	Membres du groupe	6
2.1.2.a	Martin Camincher	6
2.1.2.b	Lucien Leseigle	6
2.1.2.c	Geoffroy du Mesnil du Buisson	7
2.1.2.d	Hugo Belot-Deloro	7
2.2	Présentation du projet	8
2.2.1	Origine du projet	8
2.2.2	Nature du projet	8
2.2.3	Intérêts du projet	8
2.2.4	Etat de l'art	9
2.2.4.a	Amnesia	9
2.2.4.b	Slender	9
2.2.4.c	Alien Isolation	10
2.3	Structure	10
2.3.1	Gameplay	10
2.3.1.a	Pour le joueur en VR	10
2.3.1.b	Pour le joueur 2	11
2.3.1.c	Comportement éléments de jeu	11
2.3.2	Environnement 3D VR	11
2.3.3	Environnement du joueur 2	12
2.3.4	Audio	12
2.3.5	Menus	13
2.3.6	Site Web	13
2.3.7	Budget	13
2.4	Répartition du projet	14
2.5	Avancement du projet	14
3	Réalisations individuelles	14
3.1	Hugo Belot-Deloro	14

3.1.1	Introduction	14
3.1.2	Joueur	14
	3.1.2.a Player 101	14
	3.1.2.b Le cycle de la vie	15
	3.1.2.c Items	15
	3.1.2.d For the Hoard	15
	3.1.2.e Les objets équipables	16
3.1.3	Menus	16
3.1.4	Gestion des scènes	16
	3.1.4.a Scène persistante	16
	3.1.4.b Exemple	17
	3.1.4.c Scènes secondaires	17
3.1.5	Divers	18
3.1.6	Bilan du projet	18
3.2	Martin Camincher	18
3.2.1	Générateur de labyrinthes et structures	18
3.2.2	Détecteur de structures	19
3.2.3	Plus de structures	20
	3.2.3.a La porte	20
	3.2.3.b La lampe	20
3.2.4	Editeur de labyrinthe	20
	3.2.4.a Début de la construction	20
	3.2.4.b La construction	21
	3.2.4.c Après la construction	21
	3.2.4.d Le fonctionnement	21
3.2.5	Encore plus de structures	22
	3.2.5.a Le mur fantôme	22
	3.2.5.b Le mur invisible	22
	3.2.5.c La porte verrouillée	23
3.2.6	Mode multijoueur	23
	3.2.6.a Le joueur 1	23
	3.2.6.b Le joueur 2	23
3.2.7	Toujours plus de structures	24
3.2.8	Bilan du projet	24
	3.2.8.a Bilan des réalisations	24
	3.2.8.b Ressenti personnel	25

3.3	Geoffroy du Mesnil du Buisson	25
3.3.1	Première Partie	25
3.3.1.a	La fabrication de la Map	26
3.3.1.b	La Gestion de la Map	26
3.3.1.c	La fabrication de la végétation et d'une IA .	27
3.3.2	Deuxième Partie	27
3.3.2.a	Les avancements sur la Map	27
3.3.2.b	Le site Web	27
3.3.3	Troisième Partie	27
3.3.3.a	Ajout des items	27
3.3.3.b	Ajout des différentes assets	28
3.3.3.c	Ajout d'une texture	29
3.3.3.d	Ajout des textures	32
3.3.4	Gestion de Projet	33
3.3.5	Bilan du Projet	33
3.4	Lucien Leseigle	33
3.4.1	Musique	33
3.4.2	Bruitages et effets d'ambiance	38
3.4.3	IA	41
3.4.4	Multijoueur	41
3.4.5	Site web	45
4	Conclusion	46
5	Annexes	47
5.1	Hugo	47
5.2	Geoffroy	51
5.2.1	SAND	54

1 Introduction

Ce rapport présente le travail que nous avons accompli au cours de ces cinq derniers mois dans le cadre de notre projet informatique. Il est composé d'un rappel de notre cahier des charges original suivi des réalisations individuelles de chaque membre du groupe afin de récapituler tout le travail effectué par chacun et de pouvoir le comparer aux objectifs initiaux présentés dans le cahier des charges. Une partie annexe présente quelques captures d'écran et schémas pour illustrer le rapport.

2 Cahier des charges

2.1 Introduction

Dans le cadre des projets informatiques proposés au second semestre d'INFO SUP, nous vous présentons ici notre cahier des charges qui constituera le fil conducteur de notre projet. Tous en première année de l'école d'ingénieur EPITA Lyon, nous avons formé notre groupe de quatres élèves : Martin Camincher, Lucien Le-seigle, Geoffroy du Mesnil du Buisson et Hugo Belot-Deloro, réunis sous le nom de S-team, afin de créer un jeu : The Maze VR. Ce projet sera réparti équitablement en diverses tâches afin de pouvoir répondre le mieux possible aux attentes de notre école.

Dans ce cahier des charges, nous présenterons notre projet : The Maze VR créé par S-team. Le principe du projet, la répartition des tâches, « l'état de l'art » du projet, son découpage et sa structure seront expliqués dans les prochaines pages. L'objectif de ce projet est d'acquérir de nouvelles connaissances en programmation ainsi qu'en gestion de projet.

Ce jeu sera développé sur Unity, en C# et le design et les textures du jeu, si nous ne trouvons pas de modèle 3D dans l'asset store d'Unity, seront probablement faits sur Blender étant donné que la 3D est nécessaire. S-team vous souhaite une bonne lecture et espère que vous appréciez notre idée de projet.

2.1.1 Présentation du groupe

Contrairement aux autres groupes nous avons choisi de nous mettre ensemble pour ce projet non pas parce que nous étions proches, mais justement, car nous ne nous connaissions pas ou du moins très peu. Grâce à ce point nous sommes sûrs que nous parlerons toujours franchement entre nous et nous ne serons jamais freinés par l'amitié, néanmoins nous espérons que ce projet nous rapprochera et créera une relation d'amitié solide.

2.1.2 Membres du groupe

2.1.2.a Martin Camincher

Venant d'un petit lycée de campagne, j'ai fait une terminale S SVT spé maths. Bien que n'ayant jamais eu de contact avec le domaine de l'informatique durant ma scolarité, j'ai toujours adoré les jeux vidéo et m'intéressais un peu à leur fonctionnement. Mais mon seul véritable contact avec l'informatique était ma calculatrice TI-82 Advanced programmable du lycée. C'est ensuite au cours d'une journée d'introduction à l'informatique à laquelle j'ai participé lors de ma recherche d'école supérieure en terminale que j'ai découvert que c'était un domaine qui me passionne énormément. Je suis donc arrivé à EPITA en véritable débutant en informatique, mais j'ai pu y gagner de nombreuses compétences et je compte, au moyen de ce projet, améliorer ces compétences acquises jusqu'à maintenant et en obtenir de nouvelles.

2.1.2.b Lucien Leseigle

J'ai fait une terminale S et ayant toujours été attiré par la musique et les jeux vidéo, j'ai voulu en savoir un peu plus c'est pour cela que j'ai voulu en savoir plus non seulement sur ma passion, mais aussi sur l'informatique de manière générale, car je me suis rendu compte que les nouvelles technologies se développent de plus en ce moment. C'est pour cela qu'au moment de m'inscrire sur Parcoursup, je me suis tourné vers EPITA Lyon, car c'est l'école la plus proche de mon projet.

2.1.2.c Geoffroy du Mesnil du Buisson

Je viens d'un lycée français en Suisse et l'idée de travailler dans l'informatique plus tard a toujours été pour moi un objectif. En effet comme aujourd'hui l'informatique et les nouvelles technologies sont au cœur de l'actualité, je me dois de les prendre en compte pour mon avenir. Au cours de ma scolarité j'ai donc fait en sorte de tout mettre en œuvre pour pouvoir arriver à mes fins. J'ai fait une Terminale S avec comme spécialité ISN où j'ai d'ailleurs pu découvrir l'informatique. Grâce à ce projet j'espère pouvoir me développer à la fois dans mes compétences au sein du groupe, mais aussi renforcer mon esprit d'équipe.

2.1.2.d Hugo Belot-Deloro

Ayant découvert la programmation à l'aide de robots programmables en Lua dans le jeu Computercraft en 5^{ème}, j'ai très vite voulu m'orienter dans ce domaine pour mes études supérieures. C'est ainsi que je suis arrivé à Epita. J'ai donc déjà quelques expériences de programmation en Lua, Python, C++ et désormais en CaML et C#. Avec ce projet, j'espère apprendre à travailler en équipe puisque jusqu'ici j'ai toujours codé seul. De plus, je n'ai jamais utilisé Unity ou fait d'animations/graphismes en général, et j'ai hâte d'apprendre !

2.2 Présentation du projet

2.2.1 Origine du projet

Au début, nous voulions créer une application, plutôt destinée aux streamers et aux youtubers permettant d'ajouter des effets à un flux vidéo en temps réel automatiquement en utilisant une API pour détecter les expressions faciales d'une personne. Par exemple, zoomer sur le visage d'une personne qui sourit après une expression concentrée, ou appliquer un effet noir et blanc en réaction à une expression triste. Le problème étant que l'API faisait quasiment tout le travail et que nous aurions seulement eu à coder les effets visuels. Nous avons donc changé complètement d'idée pour un labyrinthe en réalité virtuelle suite à une proposition de Geoffroy, qui possède un casque VR et qui avait entendu parler d'un projet de Pacman en réalité virtuelle l'an dernier.

2.2.2 Nature du projet

Notre projet est donc un jeu d'horreur en réalité virtuelle où le joueur doit s'orienter dans un labyrinthe afin de récupérer des objets lui permettant de terminer le niveau. Ce labyrinthe sera hanté par une mystérieuse créature qui pourchassera sans cesse le joueur. Le labyrinthe abritera également de nombreux monstres mineurs qui, s'il ne peuvent pas forcément tuer le joueur par eux-mêmes, pourront le ralentir dangereusement. Il sera enfin parsemé de pièges ayant eux aussi vocation à ralentir un joueur peu précautieux. L'ambience sera évidemment un élément primordial de ce projet, et devra permettre une immersion totale dans l'univers du jeu.

2.2.3 Intérêts du projet

Ce projet nous permettra d'apprendre à utiliser Unity et d'avoir une première expérience de développement à moyenne échelle. Nous allons pouvoir découvrir les rendus 3D, mettre en place un moteur physique de base et des IA plus ou moins développées avec Unity, utiliser la réalité virtuelle et créer un site web. Nous apprendrons également à travailler en équipe et à gérer notre temps sur des périodes beaucoup plus longues que durant un de nos TP hebdomadaires.

2.2.4 Etat de l'art

Le premier jeu de labyrinthe était Mouse in the maze, créé en 1959 sur l'ordinateur expérimental TX-0 par un étudiant. le joueur devait créer les murs du labyrinthe, et y placer un morceau de fromage, et l'ordinateur faisait ensuite explorer le labyrinthe à la souris jusqu'à ce qu'elle trouve le morceau de fromage.

Cette idée d'explorer un terrain inconnu était particulièrement appropriée aux jeux d'horreur, où le joueur est perdu et doit tenter de se repérer tout en survivant à un ou plusieurs monstres le poursuivant.

2.2.4.a Amnesia

Le jeu Amnesia : The Dark Descent, créé en 2010, met le joueur dans la peau de Daniel, amnésique, devant explorer un manoir et échapper aux différents monstres qui y résident, et ayant pour objectif de tuer le propriétaire du manoir et de s'échapper. les principaux ennemis de ce jeu sont les serviteurs du château, des créatures humanoïdes poursuivant le joueur afin de le tuer, et l'Ombre, une entité cosmique invulnérable, qui traquera le joueur pendant tout le jeu afin de le tuer, laissant là où elle est passée un substance organique blessant le joueur et ralentissant sa progression.

Ce jeu à pour particularité que les créatures peuplant le chateau peuvent non seulement vous tuer directement, mais peuvent aussi affecter votre santé mentale et vous rendre plus vulnérable à leurs attaques.

2.2.4.b Slender

Le jeu Slender : the Eight Pages, créé en 2012 se déroule dans un espace plus ouvert, une forêt où une créature appelée le Slenderman hante le joueur pendant toute la partie, l'observant à distance et se téléportant aléatoirement afin de l'empêcher de collecter 8 pages dispersées dans le niveau.

Dans ce jeu, le Slenderman se démarque grâce à sa façon de traquer le joueur. en effet, celui-ci se téléportera lorsqu'il n'est pas dans votre champ de vision, et peut vous tuer simplement si vous le fixez trop longtemps. De plus, au fur et à mesure de votre progression dans le jeu, celui-ci se montrera de plus en plus im-

prévisible et dangereux. il est donc impossible de prévoir ses mouvements ou de le semer, et le joueur est dans le risque constant de le voir apparaître dans son dos.

2.2.4.c Alien Isolation

Le jeu Alien Isolation créé en 2014, fait explorer une station spatiale au joueur, peuplée de divers ennemis, tels des humains hostiles ou des androïdes, mais le principal ennemi du jeu est l'alien, ou Xénomorphe. Il est la menace principale du jeu et cherchera à tuer le joueur pendant toute l'aventure.

L'alien est particulier grâce à son IA. En effet, le Xénomorphe est une créature très intelligente. Celui ci ne suit donc pas de schémas prédéfinis dans ses déplacements et ses actions, mais s'adapte plutôt à celles du joueur. Il traque le moindre bruit, se déplace dans les conduits d'aération, et connaît toutes les cachettes du jeu. Aucun endroit n'est donc totalement sûr, et la menace du Xénomorphe plane constamment sur le joueur.

Ces trois jeux d'horreur ont donc des mécanique très différentes les uns des autres, et ont tous apporté quelque chose de nouveau au domaine du survival horror, et nous comptons nous inspirer d'eux pour réaliser le meilleur jeu possible.

2.3 Structure

2.3.1 Gameplay

Le concept initial du jeu est le suivant : L'environnement du joueur principal sera en VR et en 3D. Le joueur évoluera dans un environnement effrayant. Un deuxième joueur pourra éventuellement contrôler certains éléments du terrain et divers ennemis sur l'ordinateur auquel le casque VR est connecté. De ce fait, les deux joueurs interagissent de manière complètement asymétrique.

2.3.1.a Pour le joueur en VR

2 Grandes possibilités :

-La carte serait conçue comme un labyrinthe, et le but serait de s'en échapper sans mourir, notamment à cause de la créature qui traque en permanence le joueur.

-Le boss est aussi présent, mais la carte est plus ouverte. Le joueur doit retrouver des objets pour terminer le niveau, par exemple. La durée de la partie sera sans doute supérieure à celle des autres modes. Cependant ces idées seront adaptées pour garantir la meilleure expérience de jeu.

2.3.1.b Pour le joueur 2

-Une interface 2D en début de partie permettra au joueur de paramétriser le comportement et même la présence de certains éléments du jeu. Par exemple l'agressivité des monstres, ou bien la présence de pièges. Une fois la partie lancée, le joueur pourra par exemple par déclencher des pièges, libérer des monstres, ou interagir indirectement avec le joueur en VR pour lui faire obstacle. -Un mode coop (où le second joueur aide le premier) est envisagé. Tous ces éléments seront implémentés en C# (en orienté objet) dans le code du jeu, à l'aide de Unity.

2.3.1.c Comportement éléments de jeu

La diversité des ennemis, des maps, des pièges, des modes, etc... est utile pour améliorer l'expérience de jeu, le minimum étant un mode de jeu principal avec un gameplay intéressant et complet. Programmer le comportement des ennemis (l'“IA”) sera également nécessaire pour chaque action qui n'est pas faites directement par le joueur 2 : pour que les monstres traquent le joueur, ou bien qu'un piège s'active automatiquement...

2.3.2 Environnement 3D VR

-Design des éléments de jeu (Blender ou modèles) : Tous les graphiques du jeu seront implémentés à l'aide de Unity en C#. Blender est envisagé pour créer les modèles 3D.

-Map design Un éditeur de carte est envisagé. Cela nécessitera une interface permettant de configurer rapidement une carte, par exemple en insérant divers éléments comme dans un puzzle. La carte sera perçue en VR par l'autre joueur une fois générée.

-Vue 3D VR La vue du joueur qui porte le casque : en 3D et en VR, le joueur est libre de ses déplacements sur toute la carte.

2.3.3 Environnement du joueur 2

-Vue du dessus : le joueur 2 aura une vision totale de la carte et de tous les éléments du jeu (position, activation, etc...). Cela prendra la forme d'une vue du dessus (type MOBA/STR), possiblement en 2D. Il pourra interagir avec les éléments, cependant le gameplay doit rester intuitif pour le joueur 2.

-Actions sur le gameplay de l'explorateur : en interagissant avec des éléments dans une interface simple, le joueur 2 pourra modifier le cours de la partie en temps réel, de la manière précédemment décrite.

-Création de niveau avant le début de la partie : un créateur de partie permettra au joueur 2 de personnaliser la carte. Cependant le nombre d'éléments doit rester faible pour éviter que cette phase d'attente pour le joueur en VR ne soit trop longue. Le niveau pourrait également être importé/exporté

-Sur deux machines, ou bien une seule. Deux solutions sont envisagées pour le joueur 2 :

- S'il est possible d'afficher une interface sur l'écran tout en utilisant le casque VR, alors l'interface et la VR pourront être utilisés sur un seul ordinateur.
- Sinon, à l'aide de sockets réseau (implémentés en C#, probablement dans Unity), on utilisera un ordinateur par joueur. Le joueur en VR fera office de serveur (son expérience de jeu ne sera pas affectée par la qualité du réseau) ; les actions du joueur 2 lui seront envoyées puis prises en compte dans la partie.

2.3.4 Audio

-Musique : la musique sera composée sur Ableton Live. Si possible, elle sera conçue et implémentée de manière à évoluer en fonction des ennemis, devenant plus terrifiante à l'approche du danger. Toutefois cette fonctionnalité ne devra pas altérer le gameplay en restant assez subtile, pour ne pas rendre jeu trop facile.

-Bruitages Le sound design sera fait sur Ableton Live, en se basant sur des banques de samples libres de droits. Le déclenchement des sons sera, comme pour la musique, implémenté avec Unity. Si possible, un moteur audio permettra de localiser plus précisément la provenance des sons issus du gameplay.

2.3.5 Menus

Les menus en VR / joueur 2 seront implémentés dans Unity. Ils permettront d'accéder aux différents modes, de démarrer une partie et de changer des options comme le niveau sonore. Un système de récompenses n'est pas envisagé pour ce type de jeu. L'interface sera différente en VR et pour le joueur 2, car les fonctionnalités offertes au joueur ne seront pas les mêmes.

2.3.6 Site Web

Le site web sera créé à la fin du projet, soit avec un outil comme WordPress, soit développé avec divers langages (html, css, et potentiellement ruby on rails, php, django (python)... si les fonctionnalités du site sont plus complexes). Celui-ci permettra de découvrir rapidement le jeu à travers des extraits de gameplay, par exemple.

2.3.7 Budget

Nous avons déjà un casque VR dans le groupe, et allons a priori n'utiliser que des outils gratuits ou bien que nous possédions déjà. Le budget sera donc nul (ou quasi nul), les seules dépenses envisagées étant pour des assets sur Unity.

2.4 Répartition du projet

Tâche	Martin C	Lucien L	Geoffroy M-B	Hugo B-D
Multijoueur	s	R		
Gameplay	R	s	s	s
Interface			s	R
IA	s	R		s
Graphismes		s	R	
Audio		R	s	
Site web	s	s	s	R

2.5 Avancement du projet

Tâche	Soutenance 1	Soutenance 2	Soutenance 3
Gameplay	45%	75%	
Multijoueur	60%	90%	
Interface	40%	80%	
IA	25%	60%	
Graphismes	45%	80%	
Audio	15%	60%	
Site web	15%	40%	

100%

3 Réalisations individuelles

3.1 Hugo Belot-Deloro

3.1.1 Introduction

Ma partie était axée sur le gameplay et les interfaces du jeu.

J'ai travaillé principalement sur le joueur pour implémenter les contrôles, gérer l'affichage et ajouter des éléments comme l'inventaire, l'équipement et les points de vie du joueur.

Je me suis également occupé de la gestion des scènes et des menus.

3.1.2 Joueur

3.1.2.a Player 101

Au cours de ce projet, j'ai beaucoup travaillé sur l'objet représentant le joueur. Après avoir créé un objet cylindre sur Unity, je lui ai attaché un script permet-

tant de déplacer le joueur en utilisant les touches du clavier. J'ai également attaché la caméra principale du jeu au joueur afin de créer une vue à la première personne, et j'ai étayé les contrôles en ajoutant la possibilité d'orienter la caméra pour que le joueur puisse lever et baisser la tête (1).

3.1.2.b Le cycle de la vie

Une fois le joueur capable de voir et de se déplacer, il lui faut pouvoir mourir. Pour cela, je lui ai attaché un script permettant de gérer ses points de vie. Il possède initialement un total de mille points de vie. Lorsque le joueur est blessé, un overlay rouge apparaît sur son écran, d'une intensité proportionnelle aux points de vie manquants. Après un certain temps sans subir de nouveaux dégâts, les points de vie du joueur commenceront à remonter jusqu'à leur maximum et l'overlay disparaîtra ainsi progressivement. Si le joueur perd tous ses points de vie, un écran "You died" (2) apparaît et le joueur a la possibilité d'appuyer sur "espace" pour relancer le jeu ou sur "echap" pour retourner au menu principal.

3.1.2.c Items

Pour progresser dans le jeu, le joueur aura besoin de récupérer des Items (objets récupérables) comme des clés, une boussole ou une lampe torche par exemple. Les Items eux-mêmes sont représentés comme un conteneur comprenant le sprite de l'Item et un membre d'une enum représentant son type (Boussole, générique, vide, etc). Les Items ramassables (dans le monde) sont généralement contenus dans un prefab appelé WorldItem (3) qui attend une collision avec le joueur et ajoute alors l'Item à l'inventaire avant de disparaître. Les Items que le joueur porte sont placés dans un conteneur ItemSlot qui le lie avec un objet image servant à afficher le sprite de l'Item à l'écran.

3.1.2.d For the Hoard

Pour permettre au joueur de stocker les objets qu'il ramasse, j'ai ajouté deux systèmes au joueur : l'inventaire et l'équipement. L'inventaire, situé à la droite de l'écran, stocke des Items divers et généralement consommables, comme des clés ou potions de soin (non implémentées à ce jour). A gauche de l'écran, l'équipement (4) stocke des Items ayant un effet directement lorsqu'ils sont en la possession du joueur. Il ne peut stocker qu'un seul Item par type (mais le joueur ne devrait normalement pas pouvoir ramasser deux copies du même Item).

Les objets sont trouvés dans le monde et ramassés en marchant dessus. Ils sont automatiquement placés dans la bonne structure en fonction de leur type : si l'objet est de type générique alors il sera ajouté à l'inventaire, sinon il sera placé dans l'ItemSlot de l'équipement correspondant à son type. L'inventaire et l'équipement sont des gameObjects attachés au joueur et ayant pour enfants des prefabs ItemSlot contenant les scripts ItemSlot, auxquels l'inventaire accède via une liste et l'équipement via un dictionnaire <ItemType, ItemSlot>.

3.1.2.e Les objets équipables

Les objets équipables sont au nombre de trois : les vêtements chauds (la dou-doune) permettent de se rendre dans la zone enneigée sans mourir, la lampe torche active une lumière directionnelle attachée au joueur, ce qui permet d'éclairer la zone qu'il regarde et la boussole affiche à l'écran l'orientation actuelle du joueur (5). La lampe torche est essentielle pour s'aventurer dans certains labyrinthes qui n'ont pas de lampes ou dont certaines zones ne sont pas éclairées et les vêtements chauds sont requis pour s'aventurer dans la zone froide. La boussole pour sa part est optionnelle, mais reste une trouvaille très utile pour s'orienter dans les labyrinthes les plus durs.

3.1.3 Menus

Pour pouvoir passer d'un mode de jeu à l'autre ou pour modifier les options du jeu, il faut des menus. Le premier d'entre eux est le menu principal (6). Il comporte des boutons pour lancer les modes "un joueur", "multijoueur", l'éditeur de carte et le menu des options, ainsi qu'un bouton pour quitter le jeu.

Ensuite, un menu pause peut être lancé en appuyant sur "echap" en jeu. Il permet d'ouvrir les options ou de retourner au menu principal. Le menu option permet de régler le niveau sonore et la qualité graphique.

3.1.4 Gestion des scènes

3.1.4.a Scène persistante

Pour pouvoir gérer facilement les transitions entre les scènes, j'ai créé une scène persistante, qui ne sera jamais déchargée. Note : quand je parlerais des scènes actives par la suite, cela exclura la scène persistante sauf mention contraire. Cette scène persistante est la scène qui est chargée au lancement du jeu. Elle contient

un objet gameManager qui contrôle le chargement et le déchargement de toutes les autres scènes et tient une liste des scènes actives (chargées). Il peut aussi garder en mémoire la scène principale parmi toutes les scènes actives, et décharge toutes les autres scènes chargées en même temps que celle-ci. La scène principale est définie comme étant la première scène à être chargée quand aucune autre scène ne l'est.

3.1.4.b Exemple

Par exemple, lorsque le joueur lance le jeu, la scène persistante charge instantanément la scène "MainMenu" afin d'afficher le menu principal. Comme aucune autre scène n'est chargée à ce moment-là, la scène "MainMenu" devient la scène principale. Si le joueur lance le mode "un joueur", la scène "MainMenu" sera déchargée, et la scène du mode "un joueur" la remplacera en scène principale. En explorant, le joueur va alors causer le chargement de certaines zones (scènes) de la carte qui ne sont pas chargées par défaut pour réduire la charge du jeu. Mais le déchargement de ces zones une fois que le joueur les a quittées ne doit pas causer le déchargement de toutes les scènes. Plus tard, si le joueur décide de retourner au menu principal, le menu pause à travers lequel il pourra le faire demandera à la scène persistante de fermer la scène principale, ce qui causera la fermeture de toutes les scènes. De plus, en cas de décès, il est facile de décharger toutes des scènes puis de les recharger pour remettre le niveau à zéro grâce à la liste de scènes actives.

3.1.4.c Scènes secondaires

Le principe des scènes secondaires est de permettre de découper la carte en zones. Ainsi, comme la plupart des labyrinthes ne sont pas visibles depuis la surface, il n'y a pas besoin de les charger tant que le joueur ne s'en approche pas. Ces labyrinthes sont donc stockés dans des scènes individuelles qui ne sont chargées que lorsque c'est nécessaire. Pour cela, des prefabs invisibles détectent la présence du joueur et chargent/déchargent les scènes en conséquence. On peut comparer ce système à la méthode de "chunk loading" dans le jeu Minecraft. Le monde de ce jeu est un carré de plus de soixante mille kilomètres de côté, il n'est donc pas possible de garder le monde entier actif. Au lieu de cela, le monde est divisé en carrés de seize par seize mètres appelés chunks (7) (tronçons en français), et seuls les chunks les plus proches du joueur sont actifs.

3.1.5 Divers

J'ai ajouté au mode "un joueur" une lumière directionnelle faisant office de Soleil. J'ai également créé les sprites des icônes des objets et les éléments de l'affichage tête-haute ainsi que des interfaces graphiques en utilisant le logiciel paint.net.

3.1.6 Bilan du projet

Grâce à ce projet, j'ai appris à utiliser de nombreux outils comme Unity et GitHub et j'ai découvert le processus de création d'un jeu en utilisant un moteur de jeu. J'ai constaté que, en utilisant ce genre d'outil très puissant, on peut passer beaucoup moins de temps à réinventer la roue au prix de plus de temps passé à chercher des informations dans la documentation. Une fois l'outil pris en main en revanche, la vitesse de développement est très élevée et permet de créer des jeux bien plus complexes que je ne le pensais possible pour une équipe de quatre débutants. De plus, on peut obtenir des versions "jouables" et donc débuggables du code très rapidement. Au niveau de la difficulté du projet, si la plupart de mes tâches étaient, une fois la documentation lue et comprise, plutôt simples, la création de l'inventaire et de l'équipement a été très intéressante puisque j'ai dû créer une sorte de structure de donnée mêlant programmation orientée objet en C# et gameObjects d'Unity afin de pouvoir aisément déplacer et afficher les Items ramassés par le joueur. Au niveau de la répartition des tâches, j'étais initialement responsable des parties interface et site web. La partie site web a finalement été réalisée par Lucien qui avait majoritairement fini sa partie et qui, contrairement à moi, avait de l'expérience dans ce domaine. Je me suis effectivement occupé de la partie interface, j'ai beaucoup assisté pour la partie gameplay et j'ai réalisé quelques sprites pour mes interfaces.

3.2 Martin Camincher

3.2.1 Générateur de labyrinthes et structures

Dans ce projet, je me suis principalement occupé de la création et manipulation de labyrinthes sur Unity. Le premier outil que j'ai mis au point est un Générateur de labyrinthe. En effet, nous avions besoin de pouvoir rapidement expérimenter sur des labyrinthes en 3D, mais tous les créer avec Blender aurait pris trop de temps, et nous ne maîtrisions pas tous Cet outil. Ce générateur permettait donc de facilement créer et modifier divers labyrinthes. Dans sa première

version, celui-ci utilisait une image dans laquelle chaque pixel représentait une structure, en fonction de sa couleur et de sa position. Différentes structures ont ainsi été conçues, comme du sol et des murs, afin de pouvoir créer de simples labyrinthes. Ce générateur nous a donc permis de tester rapidement certaines fonctionnalités du jeu, tels les mouvements du joueur, dans un labyrinthe.

Le fonctionnement du générateur est assez simple en lui-même : chaque pixel de l'image est observé, puis, en fonction de sa couleur et de sa position, une structure est placée. L'analyse de la couleur permet d'utiliser de multiples structures, et l'analyse de la position permet d'adapter les structures en fonction de leur position. En effet, un mur Nord-Sud a le même code couleur qu'un mur Est-Ouest, mais leur position dans le labyrinthe permet de savoir lequel placer à certaines coordonnées.

Le générateur a ensuite été modifié pour ne plus utiliser d'images, mais des matrices contenant des entiers représentants les structures à placer. Cela permet de faciliter le stockage des labyrinthes définitifs.

3.2.2 DéTECTEUR DE STRUCTURES

Après avoir créé le générateur, je voulais concevoir diverses structures afin de pouvoir apporter plus de diversité dans les labyrinthes. Mais certaines structures, telles les portes, devant être activées directement par le joueur, il me fallait un moyen de savoir quelle structure était observée par le joueur. C'est pour cette raison que j'ai créé le détecteur de structures. Ce détecteur permet de savoir quelle structure se trouve à une certaine distance, directement devant le joueur. Son fonctionnement est le suivant : un Ray, une ligne infinie ayant une origine, est créé, avec comme origine le joueur. Celle-ci se dirige ensuite droit devant le joueur. Elle représente donc la ligne de vue du joueur. Une fonction permet ensuite de savoir si le Ray rencontre une structure à une certaine distance du joueur. Si oui, la structure est alors renvoyée, ce qui permet ainsi de l'identifier et d'interagir avec elle.

Le détecteur a été modifié après qu'Hugo ait implémenté la rotation verticale de la caméra du joueur. La direction du Ray n'est plus simplement droit devant le joueur, mais il peut à présent suivre verticalement la caméra.

3.2.3 Plus de structures

3.2.3.a La porte

Avec la création du détecteur de structures, je pouvais enfin interagir avec des objets en jeu. J'ai donc pu créer de nouvelles structures, telle une porte. Celle-ci est différentes des autres crées jusqu'à présent, car elle est constituée de plusieurs éléments, Les bords de la porte, la porte elle-même et un déclencheur invisible, et elle est mobile. En effet, lorsque le joueur interagit avec la porte, les battants se rétractent et permettent le passage du joueur. Les portes ne peuvent également pas être ouvertes par le monstre, ce qui en fait des moyens de lui échapper temporairement.

Le fonctionnement de la porte est le suivant : lorsque la porte est observée par le joueur, le détecteur de structure repère le déclencheur, ce qui informe la porte que celle-ci peut être activée. Si le joueur appuie alors sur le bouton d'action, les battants de la portes se déplacent jusqu'à atteindre leur nouvelle position, dépendant de si la porte était fermée ou ouverte. Après que la porte ait été ouverte, celle-ci passe dans un état de "sommeil" qui l'empêche d'être réactivé pendant 0.25 secondes, afin d'empêcher que la porte ne s'ouvre et se ferme à chaque frame tant que le bouton est activé.

3.2.3.b La lampe

La deuxième structure créée à ce moment a été une lampe. Celle-ci est suspendue au plafond et diffuse une lumière orangée, rappelant la couleur d'une flamme. Il s'agit en réalité d'un simple modèle 3D, auquel est attaché une area light créant de la lumière.

3.2.4 Editeur de labyrinthe

J'ai ensuite créé l'éditeur de labyrinthe. Ce mode de jeu permet de créer ses propres labyrinthes personnalisés afin de les utiliser en mode multijoueur.

3.2.4.a Début de la construction

Le joueur à ici deux options : créer son propre labyrinthe, ou en modifier un. Si le joueur choisi de créer son propre labyrinthe, il entre alors les dimensions qu'il souhaite. Un bouton permet alors de confirmer la sélection, et génère ensuite une base de labyrinthe ayant les dimensions souhaitées.

Si le joueur choisit au contraire de modifier un labyrinthe déjà existant, il entre un code correspondant à ce labyrinthe et peut ainsi le modifier. Des modèles incomplets sont également à disposition afin de commencer plus facilement la création d'un labyrinthe.

3.2.4.b La construction

Le joueur voit ensuite le plan du labyrinthe vu de haut, et peut placer diverses structures aux coordonnées qu'il souhaite. Les structures s'adaptent automatiquement aux coordonnées, ce qui rend impossible le placement, par exemple, d'un mur Nord-Sud à l'emplacement d'un mur Est-Ouest, ou d'une lampe.

Il est également possible de supprimer des structures indésirables, ce qui permet de corriger d'éventuelles erreurs de construction ou de modifier des labyrinthes pré-construits.

3.2.4.c Après la construction

Après avoir construit son labyrinthe, le joueur valide sa création, qui est alors transformée en un code permettant de partager le labyrinthe et de l'utiliser en mode multijoueur.

3.2.4.d Le fonctionnement

Le fonctionnement de l'éditeur est assez complexe : Dans un premier temps, la base du labyrinthe ou le labyrinthe d'origine est créé par un algorithme inspiré du générateur de labyrinthe. Une matrice "plan" est également créée aux mêmes dimensions que le labyrinthe, et représente les différentes structures à leurs coordonnées respectives. On passe alors dans une phase de "placement". Une structure est placée aux coordonnées d'origines si celles-ci ne sont pas occupés. Le joueur a alors plusieurs options. Il peut déplacer la structure afin de la placer là où il le souhaite, placer la structure et passer à la suivante, ou passer en mode destruction.

Pour les déplacements et la pose de structures, le programme observe à chaque frame si certaines touches du clavier sont activées. Si oui, le programme agit en fonction. Pour un déplacement, la structure actuelle est détruite si elle était placée à un emplacement originellement vide, puis les coordonnées sont mises à jour et une nouvelle structure est placée si l'emplacement est vide. Il est donc aisément de visualiser où se trouve la structure qui est en train d'être placée. Pour

une pose de structure, celle-ci n'est tous simplement pas détruite, puis est ajoutée à la matrice plan avant de passer à la suivante.

Pour le changement de structure, celles-ci sont organisées dans une liste. Deux touches permettent d'avancer ou de reculer dans la liste. A chaque changement de structure, si celle-ci ne peut pas être placée aux coordonnées actuelles, elles sont alors corrigées.

Enfin, la destruction de structure. Cette option est accessible en fin de la liste de structures. Si l'une d'entre elles se trouve aux coordonnées actuelles, elle devient alors rouge afin de facilement la repérer. Les boutons de déplacements permettent alors de sélectionner une autre structure, tandis que le bouton qui sert à placer celles-ci est ici utilisé afin de confirmer la suppression de la structure actuellement visée, suivi par sa suppression de la matrice plan.

Enfin, une fois que le labyrinthe est terminé, la matrice plan est convertie en un code contenant toutes les informations du labyrinthe, que ce soit ses dimensions ou les coordonnées de chaque structure.

3.2.5 Encore plus de structures

Après avoir créé l'éditeur de labyrinthe, il fallait plus de structures afin de rajouter de la diversité dans les labyrinthes.

3.2.5.a Le mur fantôme

La structure que j'ai ensuite créée est très simple, il s'agit d'un mur d'apparence ordinaire, mais qui ne possède pas de collision, ce qui permet de passer au travers. Il permet ainsi de piéger les joueurs trop hâtifs. Ce genre de mur peut par exemple être placé à la fin d'un couloir semblant être un cul-de-sac, et créé ainsi un passage secret pouvant récompenser les joueurs curieux.

3.2.5.b Le mur invisible

D'autres structures sont les murs et portes invisibles. Ces murs et portes sont identiques aux versions standards dans leur fonctionnement, mais sont presque transparents, et donc difficiles à voir au premier coup d'œil.

3.2.5.c La porte verrouillée

Une autre structure est la porte verrouillée. celle-ci est différente d'une porte normale car elle ne s'ouvre pas simplement. En effet, le joueur aura besoin de trouver une clef dans le labyrinthe afin de pouvoir ouvrir ce type de porte. ces clefs sont stockés dans l'inventaire et déverrouillent la porte, la transformant en porte classique dans son fonctionnement. Une version invisible de ce type de porte existe également.

3.2.6 Mode multijoueur

Le dernier mode de jeu est le mode multijoueur.

Dans celui-ci, deux joueurs s'affrontent avec un gameplay asymétrique.

3.2.6.a Le joueur 1

Le joueur 1 est dans une situation similaire au mode solo. Son but est de récupérer plusieurs objets dans le labyrinthe avant de ressortir en tentant de ne pas mourir. Son gameplay sera donc quasiment identique au mode solo, lors de l'exploration de labyrinthe.

3.2.6.b Le joueur 2

La plus grande nouveauté de ce mode est le joueur 2. Celui-ci joue sur un ordinateur, et voit le labyrinthe de dessus, comme dans l'éditeur de labyrinthe. Son but est de faire de son mieux afin d'empêcher le joueur 1 de compléter le labyrinthe. Il aura pour cela plusieurs outils, tels des pièges et des interactions avec le labyrinthe.

C'est le joueur 2 qui choisira le labyrinthe utilisé, en fournissant au début de la partie un code contenant toutes les informations nécessaires. Il choisira également le design des structures du labyrinthe en sélectionnant un thème, qui définira l'apparence des structures, mais ne changera rien à leur fonctionnement.

La principale caractéristique du joueur 2 est sa jauge d'énergie. En effet, chacune de ses actions consommera de l'énergie, qui se remplira ensuite avec le temps. cela permet au joueur 2 une certaine liberté dans ses actions tout en empêchant le joueur 1 de subir un déluge de pièges.

La plupart des pièges sont des pièges consommables, qui sont activés une seule fois et qui disparaissent ensuite. Leur présence est également limitée dans le temps,

et le nombre de pièges d'un même type présents simultanément est également limité, ce qui empêche au joueur 2 de créer des zones du labyrinthes remplies de pièges et donc infranchissables. Ils auront également un temps d'installation avant d'être actifs.

Les interactions du joueur 2 avec le labyrinthe ne passeront pas uniquement par les pièges. En effet, il aura d'autres pouvoirs, tels ouvrir et fermer des portes, ou bien temporairement éteindre toutes les lampes, ou encore permettre pendant quelques instants à l'IA d'être beaucoup plus précise dans ses déplacements. Le joueur 2 pourra également placer des murs sans collision afin de destabiliser le joueur 1.

3.2.7 Toujours plus de structures

Toujours plus de structures ont ensuite été créés, principalement des pièges pour le joueur 2 en mode multijoueur.

Ces structures sont nombreuses et variées, et ont toutes différents effets.

En premier, nous avons une structure qui attire l'IA. Elle permet ainsi au joueur 2 de repositionner l'IA afin de par exemple anticiper les déplacements du joueur 1.

D'autres pièges ont pour but de gêner directement la progression du joueur. C'est le cas par exemple pour le piège ralentissant, qui baissent sa vitesse de déplacement pendant quelques instants. Le piège de ténèbres créé un voile sombre devant les yeux du joueur 1 afin de le gêner, et le piège à EMP éteint sa lampe torche pendant quelques instants.

Enfin, certaines structures n'ont pas d'effet direct sur le joueur 1, mais peuvent servir à le destabiliser. L'une de ces structures émet ainsi des sons effrayants, tels des grognements, des bruits de pas, où la musique du monstre. Le joueur 2 peut placer cette structure où il veut afin de choisir d'où vient le son. Il peut également créer une illusion du monstre, plus petite, ayant le même comportement que l'original mais disparaissant au contact ou après un certain temps, et incapable d'infliger le moindre dégât au joueur 1.

3.2.8 Bilan du projet

3.2.8.a Bilan des réalisations

Dans ce projet, mon travail a donc principalement été le développement d'outils et de programmes. La plupart des mes réalisations sont en rapport avec la créa-

tion et la manipulation de structures, que ce soit le générateur ou l'éditeur de labyrinthes, ou bien la création même de structures, certaines très basiques comme les murs, d'autres plus complexes, comme les portes ou les pièges.

3.2.8.b Ressenti personnel

Pour moi, ce projet a été une étape longue et difficile, mais également très intéressante et enrichissante. La plus grande difficulté que j'ai eu au cours de la réalisation de ce projet a été d'apprendre à utiliser Unity. En effet, cet outil est très riche en contenu et permet de réaliser des jeux très complexes, mais cela signifie qu'il est lui-même complexe. Or, nous nous sommes lancés dans la réalisation de ce projet sans jamais avoir touché à cet outil auparavant. Cela a donc amené de nombreuses difficultés, souvent dues à un manque de connaissances sur cet outil. La recherche d'information en ligne était donc parfois longue et laborieuse, mais très récompensante quand le programme en cours de création se mettait enfin à fonctionner correctement.

Ainsi, ce projet, bien que souvent source d'angoisse, a été une très bonne expérience pour moi.

3.3 Geoffroy du Mesnil du Buisson

Je suis responsable des graphismes et directeur du projet et dans cette section je vais expliquer en détail le fonctionnement des différents éléments que j'ai réalisés ainsi que les processus par lesquelles je suis passé pour les créer. Je parlerais aussi des problèmes rencontrés ainsi que des solutions retenues.

3.3.1 Première Partie

Pendant la première partie du projet soit le moment entre le début du projet et la première soutenance je me suis surtout concentré sur les bases du gameplay du projet sur la jouabilité du jeu et la gestion des commandes tout en faisant la gestion du groupe qui à d'abord travaillé surtout en autonomie me laissant la possibilité d'être assez libre de mon côté car éviter de devoir m'investir trop dans la gestion du groupe m'a notamment permis de pouvoir me préoccuper à plein temps de mon apprentissage sur Blender afin de pouvoir procurer ma contribution au projet. Blender étant un logiciel très dense et très complexe il

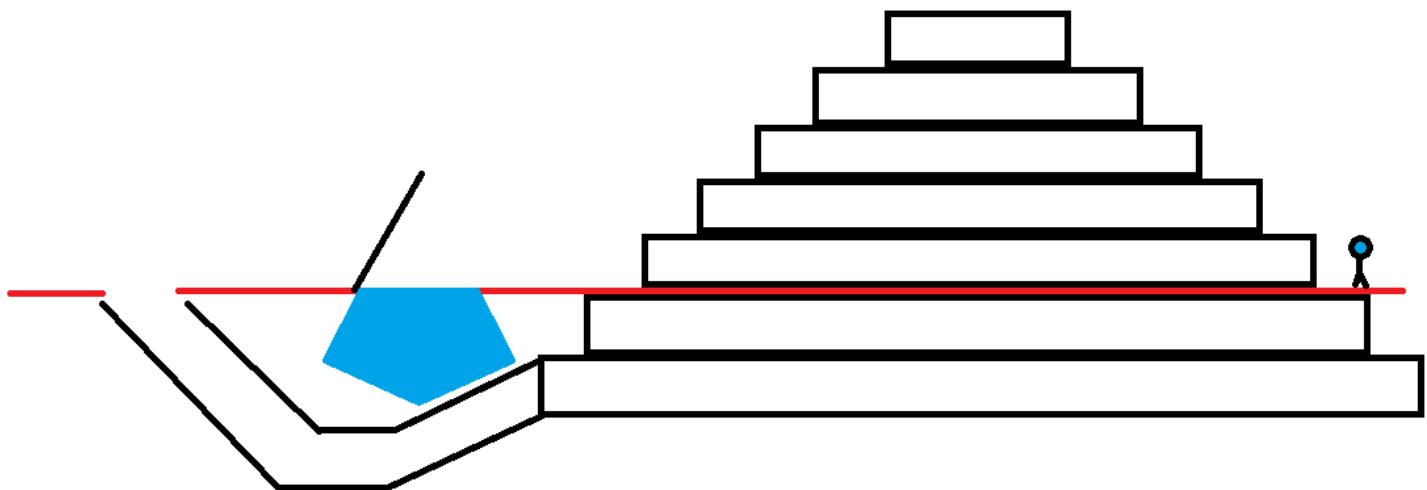
m'a fallu beaucoup de temps pour me faire la main. C'est pourquoi au moment de la première soutenance j'avais avancé sur trois principaux objectifs.

3.3.1.a La fabrication de la Map

Pour la première fabrication de la map je suis passé par Blender et j'ai "sculpté" la map afin de pouvoir créer du relief, des crevasses pour laisser passer la future rivière et j'avais laissé un énorme trou au milieu de la map afin que nous puissions plus facilement par la suite mettre la grande pyramide.

3.3.1.b La Gestion de la Map

Au moment de la première soutenance j'avais déjà fini de créer la surface de la map et les principales fondation pour la futur pyramide que nous avions déjà imaginée :



Le reste de la gestion était encore trop flou étant donné mes notions sur Blender à ce moment là.

3.3.1.c La fabrication de la végétation et d'une IA

Pour la végétation j'avais créé quatre types d'arbres complètement différents les uns des autres. Malheureusement ces arbres n'ont pas été gardés pour la suite du projet car ils étaient en surdose d'arêtes et par conséquent trop coûteux en stockage.

3.3.2 Deuxième Partie

Durant la deuxième partie j'ai surtout avancé sur les graphismes de la map et j'ai réalisé notre site web qui était temporaire car Lucien en a fait un autre par la suite.

3.3.2.a Les avancements sur la Map

Pour la deuxième soutenance j'avais fini la pyramide en ayant tout rattaché bout à bout, j'ai débuggé la map, puis j'ai créé des grandes entrées dans les montagnes pour les différents labyrinthes répartis équitablement sur la carte.

3.3.2.b Le site Web

Pour le site web je suis passé par Wix qui m'a fournis les éléments nécessaires pour le créer facilement et faire un site honorable même si il nous ne servira pas.

3.3.3 Troisième Partie

Cette troisième partie sera surtout consacrée aux différents avancements que j'ai pu réaliser entre la deuxième et la dernière soutenance. Cette troisième partie représente pour moi l'aboutissement du jeu dans son ensemble et la fin d'un gros projet.

3.3.3.a Ajout des items

Pour l'ajout des items j'ai principalement travaillé sur Blender n'ayant pas de texture à ajouter sur les petits objets, j'ai préféré mettre des couleurs unies afin que ce soit plus agréable visuellement.

Le premier Item que j'ai pu créer est la boussole. Pour cela je l'ai décomposée en trois partie : les aiguilles, le cadrant et les lettres. Pour le cadrant je suis parti d'un cylindre que j'ai aplati, pour les aiguilles j'ai utilisé un rectangle que j'ai fusionné avec un triangle et pour les lettres j'ai utilisé les fonctionnalités de Blender qui nous permettent de pouvoir directement écrire en objet 3D.(8)

Le deuxième Item que j'ai réalisé était la lampe. La lampe était plus difficile à faire que la boussole. Pour la lampe je suis parti d'un cylindre que j'ai ensuite extrudé puis déformé pour ensuite lui appliquer des textures unies directement sur les faces. La lampe était plus dure à faire même si il n'y avait qu'un seul élément alors que la boussole en contenait trois, mais la lampe devait contenir de la lumière même si le joueur ne l'a pas encore récupérée. Cela permet déjà au joueur de pouvoir trouver la lampe plus facilement et cela ajoute aussi au coté réalisme.(9)

Le troisième item que j'ai fabriqué est l'échelle. L'échelle était vraiment simple à faire c'est pourquoi pour compenser je l'ai implémentée dans le jeu et j'ai codé l'action de pouvoir monter à l'échelle quand nous sommes dans la range du collier de l'échelle.(10)

Le dernier item que j'ai fait était pour le coup beaucoup plus complexe puisqu'il représentait les vêtements chauds. En effet les vêtements sont des objets difficiles à représenter géométriquement parlant. J'ai donc utilisé pour m'aider le corps de Bob (notre personnage) auquel j'ai retiré la tête, les mains et les jambes pour pouvoir me retrouver avec un simple buste que j'ai dû ensuite déformer et réadapter à ma guise.(11)

3.3.3.b Ajout des différentes assets

-J'ai fait le personnage principal (Bob) sur Blender en m'inspirant d'une asset de Unity pour avoir des références physiques plus simples à exploiter que le corps de mes camarades.

-J'ai ajouter l'ensemble de la végétation, principalement en utilisant le AssetStore de Unity. En effet, celui-ci étant très dense j'ai pu y trouver gratuitement

l'ensemble des choses dont nous pouvions avoir besoin que ce soit des arbre, de la pierre des plantes...

-J'ai pu aussi ajouter la totalité des habitations en passant par le UnityAssetStore.

-J'ai géré l'éclairage de la map ainsi que le ciel et l'eau en appliquant des images 2D

-J'ai rajouté les graphisme de l'IA, graphismes que nous avons achetés. En effet il était préférable de payer pour avoir une IA réaliste qui fasse peur étant donné que l'IA représente le monstre de notre jeu.

L'ajout de tout ces assets ne fût pas difficile mais par contre très long puisque cela consistait à placer avec patience les objets les uns après les autres.

3.3.3.c Ajout d'une texture

Dans cette partie nous parlerons de la fabrication d'une seule texture puisque le procédé est le même pour toutes les textures. Nous nous attarderons un peu sur cette partie car c'est la partie qui m'a demandé le plus de temps pour la troisième partie du projet. Une texture est la couleur que nous appliquons à un objet ou à une face de cette objet. Il y a trois niveaux de texture : les textures simples qui sont l'application d'une couleur unie à une face de l'objet ou à l'objet en entier, les textures dites "patronales" qui sont l'application d'une seule image sur un patron de l'objet souhaité. C'est-à-dire, pour faire simple que si l'on prend une boite en carton et qu'on en découpe les bords pour l'avoir à plat et n'avoir plus que le patron du dit objet, on applique alors une image ou une photo sur celui-ci. Puis pour finir il y a les textures dites complexes qui sont l'application de plusieurs images différentes sur le même patron afin d'obtenir un rendu de texture 3D avec du relief, ces dernières sont plus difficiles à réaliser mais plus réalistes et plus agréable à regarder.

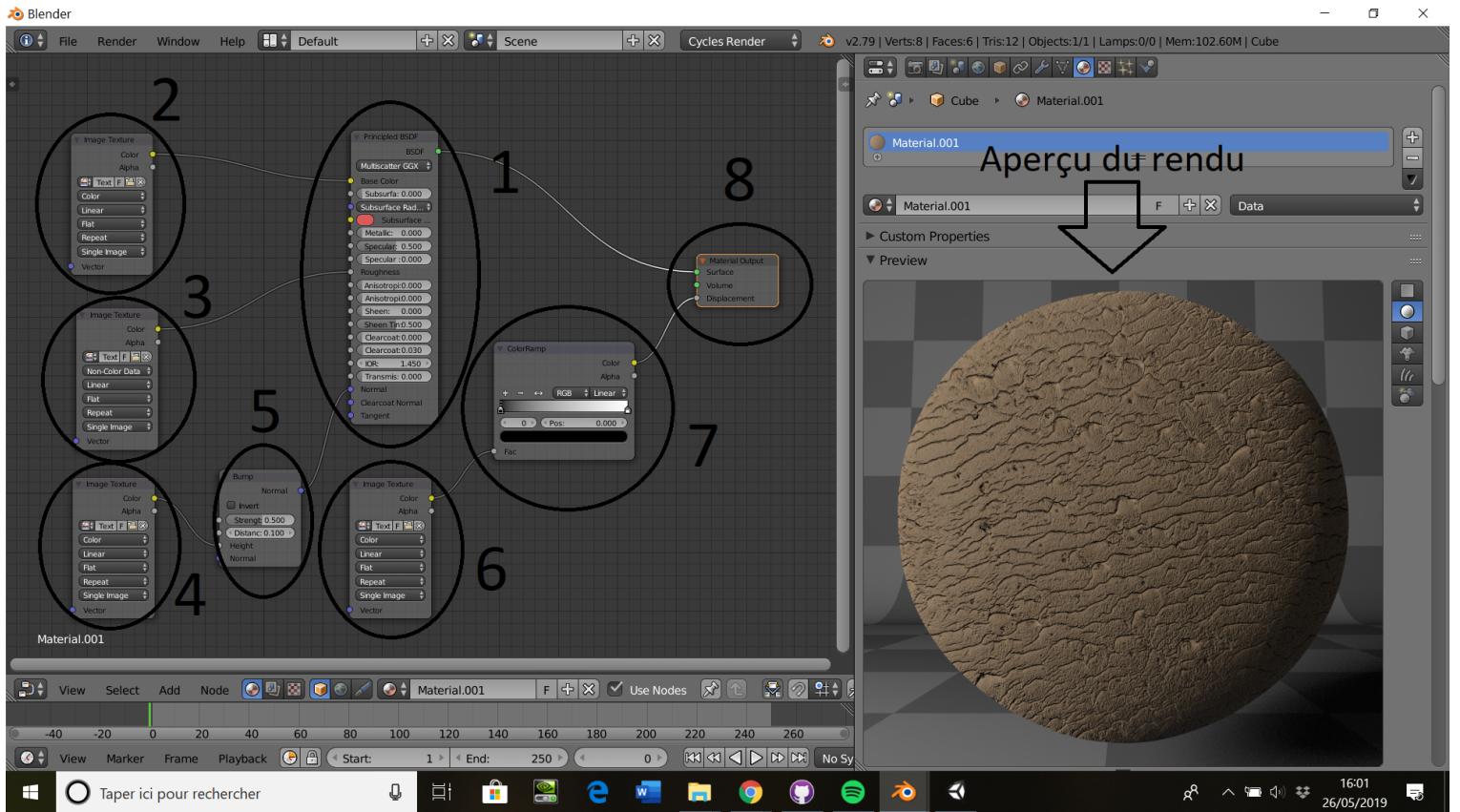
Pour ce projet j'ai principalement utilisé des textures complexes. Nous allons donc voir comment j'ai procédé pour créer ces textures complexes.

Premièrement, la map ayant été réalisée sur Blender, il a fallu passer les fichier de Blender sur Unity en les exportant sous forme de fichiers .FBX, or les fichiers .FBX n'étant pas des fichier .Blender ne conservent pas toutes les modifications apporté audit fichier, notamment les textures. J'ai donc dû trouver une solution pour récupérer mes textures créées sur Blender. En effet nous ne pouvions pas appliquer nos textures directement sur Unity car les objets faits en 3D sont composés de multiples faces plus ou moins grandes qui une fois mises toutes bout à bout donnent l'impression de courbure or comme dit précédemment nous devons appliquer les textures directement sur ces faces (le patron) or Unity n'a pas accès à ces faces. Donc pour chaque texture je devais les créer sur Blender, les modifier afin de vérifier que le rendu soit bien conforme à mes attentes puis enregistrer chacune de mes composantes de rendu en fichier .Jpeg afin de pouvoir recréer la texture ensuite sur Unity en gardant les même propriétés que sur Blender. Une fois sur Unity il fallait insérer la texture et la redimensionner afin de retomber sur le même rendu que sur Blender.

Nous allons donc voir chacune des étapes pour faire cela.

Premièrement la création d'une texture sur Blender. Blender est doté d'un mode : le Cycle Render, qui nous permet de travailler avec des noeuds d'apparence que nous allons développer pour cette partie nous nous concentrerons sur la création d'une seule texture, nous allons donc nous appuyer sur la texture SAND (sable)

qui va aider pour nous projeter.



- 1) Principled BSDF : Ce noeud (Node) est le noeud principal de l'arbre. Parmi tous les noeud que nous allons créer il sert à affecter les principaux rendus visuels c'est ce noeud qui va gérer toute l'apparence de notre texture.
- 2) Image Texture Albedo : Ce noeud est directement rattaché à la couleur de notre texture c'est pourquoi nous attachons ce noeud directement sur le noeud principal au niveau de la couleur. L'image que nous utilisons ici est sous format .tif et est directement issu du site "Testure.com" sur lequel j'ai récupérer toutes mes images de bases que j'ai ensuite modifiées pour les textures.
- 3) Image Texture Roughness : Ce noeud sert lui à représenter les creux, la rugosité , les déformations et les courbures. Etant donné qu'il ne représente pas une couleur contrairement à Albedo nous lui affectons la valeur Non-Color-Data. Il est lui aussi relié au noeud principal.

- 4) Image Texture Height : Ce noeud a lui pour but de créer du relief contrairement au précédent qui créait les creux. Il est relié à un vecteur Bump qui fait office de régulateur.
- 5) Vecteur Bump : Ce noeud est un régulateur de force et de distance. La force permet de creuser plus ou moins les trous et la distance sert à accentuer les courbures.
- 6) Image Texture Ambiant Occlusion : Ce noeud est fait pour créer une occlusion ambiante qui permet d'assombrir les zones naturellement difficiles d'accès à la lumière. Il est relié à un convertisseur de ColorRamp.
- 7) Convert ColorRamp : Ce noeud est fait pour atténuer les zones d'assombrissement de lumière il est directement relié au noeud de sortie Output.
- 8) Material Output : Ce noeud représente la sortie du schéma ce qui permet de faire ressortir le rendu. Nous relions donc le BSDF et le ColorRamp à ce noeud afin de terminer le chemin. Nous pouvons voir un aperçu du résultat. D'autres aperçus de rendu sont disponibles en annexe à la fin.

Après avoir enregistré toutes les images en fichier .JPEG et Unwrap toutes les faces sur lesquelles il faut appliquer le matériau (créer le patron) on exporte la totalité de l'objet en fichier .FBX sur Unity où on doit réappliquer à chacun des noeuds chaque structure de la texture. Unity est plus explicite que Blender, il suffit donc juste de positionner les fichiers aux bons endroits (12).

Et pour finir il fallait redimensionner les fichiers sur le rendu afin d'obtenir le résultat escompté.(13)

3.3.3.d Ajout des textures

Au total j'ai ajouté une quinzaine de textures au total avec la méthode vue juste au-dessus. Cela était long et fastidieux mais ce n'était pas grand-chose par comparaison au temps qu'il m'a fallu pour trouver cette méthode.(16)

3.3.4 Gestion de Projet

En tant que Responsable, je me suis pleinement impliqué dans ce projet afin de contribuer à la réussite de celui-ci. Nous avons créé un groupe de discussion pour faciliter les échanges entre nous. J'ai eu l'idée de monter une page Facebook ainsi qu'un compte Instagram dans l'optique de mettre en avant notre projet, mais c'est surtout afin d'optimiser notre force de travail que j'interviens. En effet j'ai mis à disposition mon appartement tous les jours pendant toute la semaine avant la soutenance afin que nous puissions travailler tous ensemble sur ce projet.

3.3.5 Bilan du Projet

Ce projet m'a permis d'en apprendre énormément sur moi-même que ce soit humainement parlant en ayant pour rôle d'être chef de projet, mais aussi scientifiquement parlant avec le grand nombre de choses que j'ai pu apprendre que cela soit sur Unity ou sur Blender ou encore avec la programmation orienté objet de C#. J'ai apprécié faire ce projet, il m'a beaucoup fait progresser sur beaucoup de facteurs différents.

3.4 Lucien Leseigle

Dans cette section, je vais expliquer en détail le fonctionnement des différents éléments dont je suis responsable, ainsi que le processus de création. Je parlerais aussi des problèmes rencontrés ainsi que des solutions retenues.

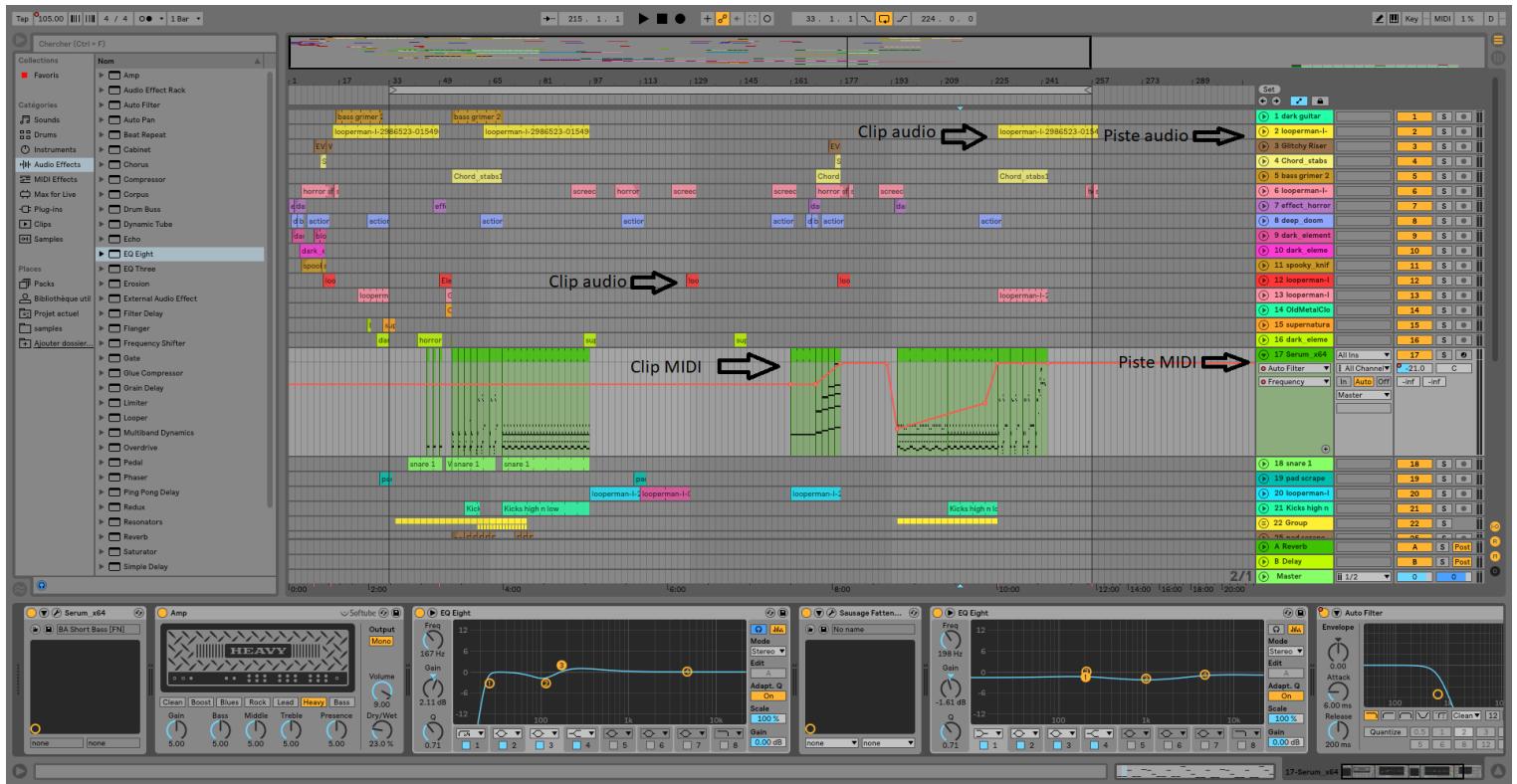
3.4.1 Musique

La musique est le premier élément du jeu auquel je me suis intéressé. Elle comprend la bande son du jeu ainsi que la musique du menu, de durées respectives 12 et 1 minutes. J'ai composée la musique de toutes pièces en utilisant Ableton Live 10, un DAW (Digital Audio Workstation) très complet. Ce logiciel m'a non seulement permis de composer la musique, mais aussi de traiter le signal audio ('mix') et de découper le son de manière optimale, pour que la gestion de l'audio soit la plus simple possible (elle ne prend en effet que quelques lignes de codes, car elle se contente de boucler les bons clips audio, qui sont déjà conçu de manière à enchaîner naturellement). La réalisation de la musique se découpe en 3 étapes : composition, traitement, découpe.

1 : composition

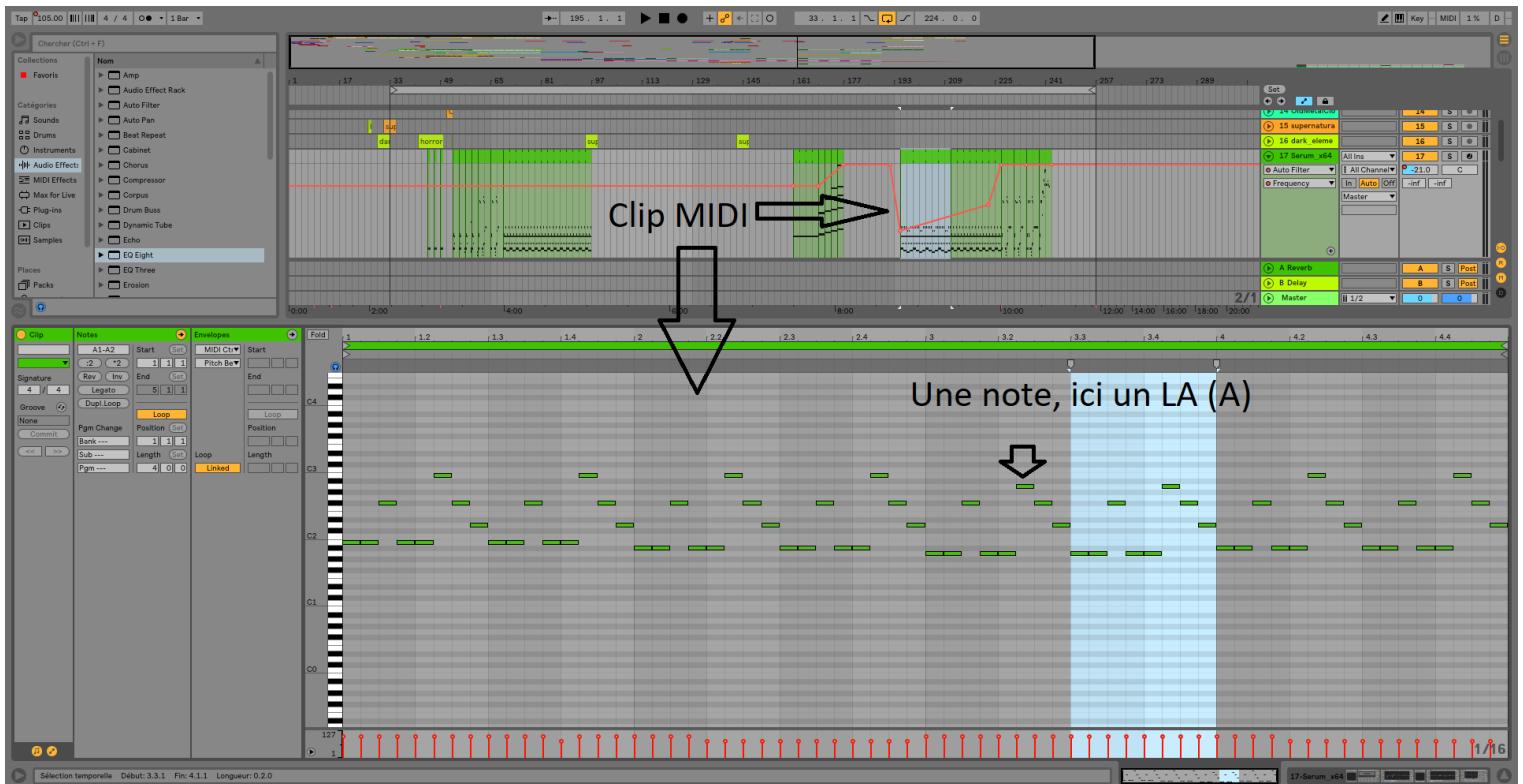
Le morceau final est constitué de clips audio ou MIDI contenus dans des pistes superposées, avec des effets appliqués sur les différentes pistes et qui évoluent dans le temps.

Ci-dessous, la vue 'timeline' d'Ableton. Les clips audio et midi sont représentés par les rectangles de toutes les couleurs au milieu de l'écran. chaque rectangle contient soit un clip audio, soit un clip MIDI sur un instrument virtuel. Qu'ils soient MIDI ou Audio, les clips sont répartis sur des pistes (les rectangles, eux aussi colorés, alignés à droite avec des numéros). Le projet utilise en tout 39 pistes audio (souvent les musiques de jeux/films contiennent de nombreux éléments et donc beaucoup de pistes).



Les éléments de base sont les clips, audio ou MIDI. Un clip audio contient un signal audio, par exemple un enregistrement réalisé avec un micro. Un clip MIDI est similaire à une partition (et peut d'ailleurs être écrit très facilement sous

cette forme) : il contient des informations permettant de jouer une mélodie (hauteur des notes, rythme, vitesse...) mais pas de son précis. Ci-dessous, un clip midi.



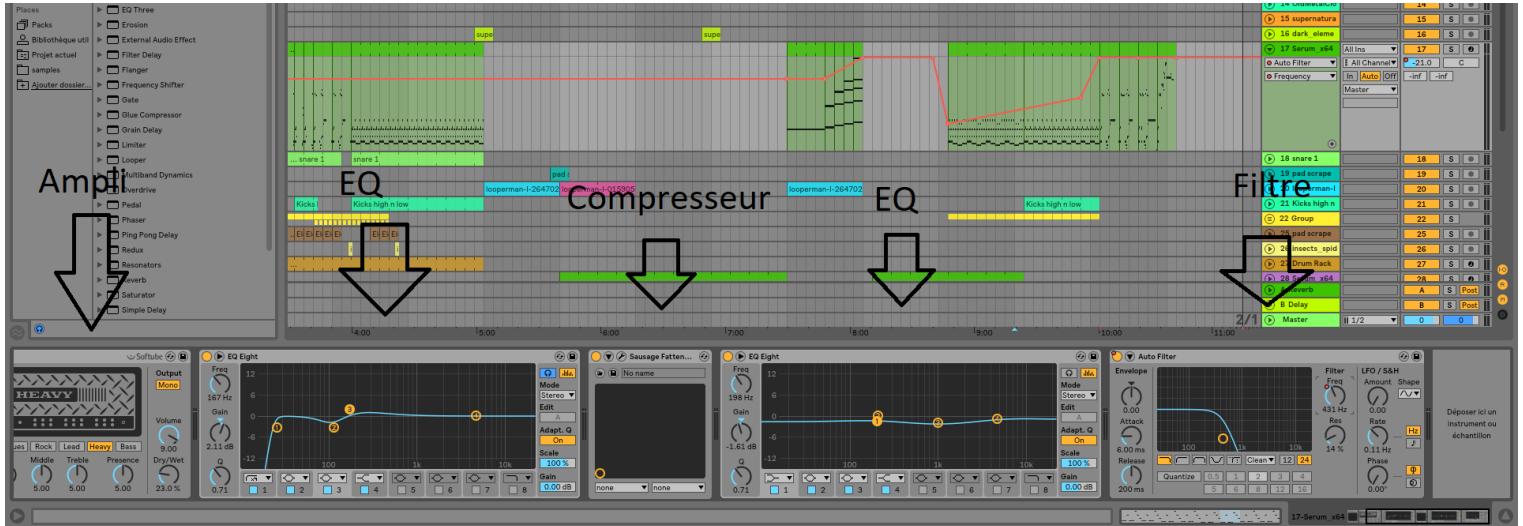
Les rectangles verts représentent les notes jouées en fonction du temps - ici, 4 mesures soit 16 temps - le projet est à 105bpm, soit 105 battements (temps) par minutes. J'ai réalisé de nombreux clips MIDI pour créer les différentes mélodies qu'on entend dans le morceau. Un clip MIDI est ensuite joué par un instrument virtuel, un synthétiseur. Ci-dessous on voit Serum, qui est un programme de type VST ou instrument virtuel ; il s'agit d'un synthétiseur numérique, et il s'utilise exactement de la même manière qu'un synthétiseur analogique. Je ne vais pas détailler le processus de synthèse sonore, mais il s'agit de régler tout les boutons du synthétiseur de manière à obtenir le son souhaité. Pour cela, le

schéma de base est un ou plusieurs oscillateurs, qui suivent une enveloppe, puis passent par un filtre.



Les sons peuvent ensuite être modifiés avec des effets audio. Il en existe une grande variété, les plus connus (et communs) étant les compresseurs (ils 'écrasent' la forme d'onde, ou normalisent les volumes en quelque sorte), equalizers (EQ, permet de régler le volumes par fréquences via une transformée de Fourier), Delay (simule un écho) et Reverb (ajoute des réverberations, simule une pièce). Ci-dessous, la piste de guitare électrique. On peut observer, par exemple, en bas, 2 Equalizers, un Ampli, un compresseur (Saussage Fatner), et un filtre. La ligne rouge sur la piste est une automation : elle est associé à la fréquence de coupure du filtre et évolue en suivant la courbe. Ce sont ces effets qui permettent de

transformer le son de base assez numérique du synthétiseur en une guitare électrique, et de moduler lentement le son au fil du morceau pour le faire évoluer.



2 : traitement

Une fois la musique composée, il faut mixer le son, c'est à dire régler les volumes, la panoramique, et utiliser divers effets pour améliorer le rendu final. Ces effets sont plus ou moins les mêmes que ceux présentés plus haut, simplement utilisés plus subtilement : le but n'est plus de distordre le son, mais d'apporter à chaque élément sa place dans le morceau.

Une fois le mixage terminé, on peut masteriser : il s'agit d'appliquer des effets sur le master, c'est à dire une piste par laquelle toutes les autres pistes sont redirigées. Cela contribue à améliorer le son, mais c'est surtout à cette étape que l'on conforme le son aux standards de l'industrie.

3 : découpe

Un logiciel comme Ableton permet de boucler très facilement des parties du morceau tout en conservant toutes les réverbérations, et d'enchaîner les pistes audio et MIDI sans coupures. Il suffit de faire en sorte que le début de la boucle puisse enchaîner naturellement avec la fin. Une fois cela fait correctement, il n'y a plus aucune transition à gérer dans Unity, et un script minimaliste pour l'audio est largement suffisant. Sur l'image de la vue timeline, on peut voir la boucle entre les deux traits verticaux. Le début et la fin coïncident parfaitement puisqu'ils ont été réalisés en même temps.

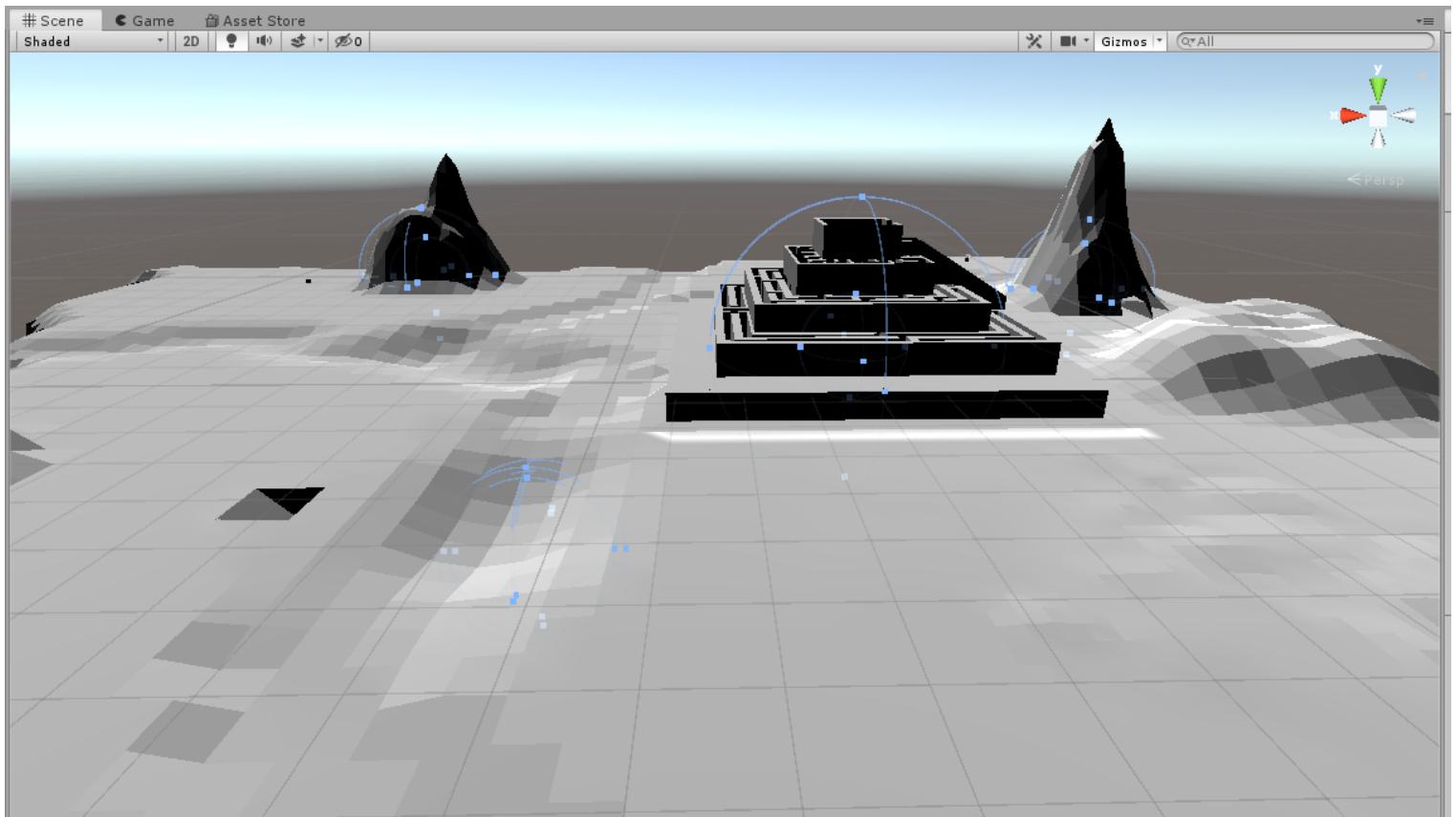
La musique est ensuite envoyée dans une piste audio, ce qui permet d'appliquer des effets dans Unity. La musique est, certes, déjà traitée, mais les effets détaillés dans la prochaine section (bruitages) vont modifier les volumes et il est nécessaire de ré-appliquer des effets pour que le volume ne soit pas trop fort : un compresseur (réduit les pics de volume par un facteur choisi), et un limiteur (écrase la forme d'onde si elle dépasse un certain seuil).

3.4.2 Bruitages et effets d'ambiance

Les bruitages sont réalisés de la même manière que la musique : Chaque bruitage est une combinaison de synthétiseurs qui jouent un clip MIDI, de fichiers audio trouvé sur internet (sample) et d'effets. Les bruitages sont ensuite joué dans le moteur audio 3D d'Unity, ce qui permet de donner l'impression que le son provient de l'endroit où se situent les ennemis. Le déclenchement des différents bruitage est géré par un script qui associe les évènements du jeu aux fichiers audio correspondants. Tout le son est redirigé vers une piste audio (master), comme pour la musique.

Il y a quelques couloirs dans le jeu, j'ai donc ajouté des Reverb Zones dans ceux-ci. Cela donne l'impression que la musique est jouée depuis l'intérieur du tunnel grâce aux réverbération adaptées a ce type d'environnement. plus le joueur s'approche de la "Full Zone", et plus l'effet est intense. Ces zones de réverbération permettent un réalisme et donc une immersion accrue. Ci-dessous les différents

zones dans lesquelles l'effet est activé :

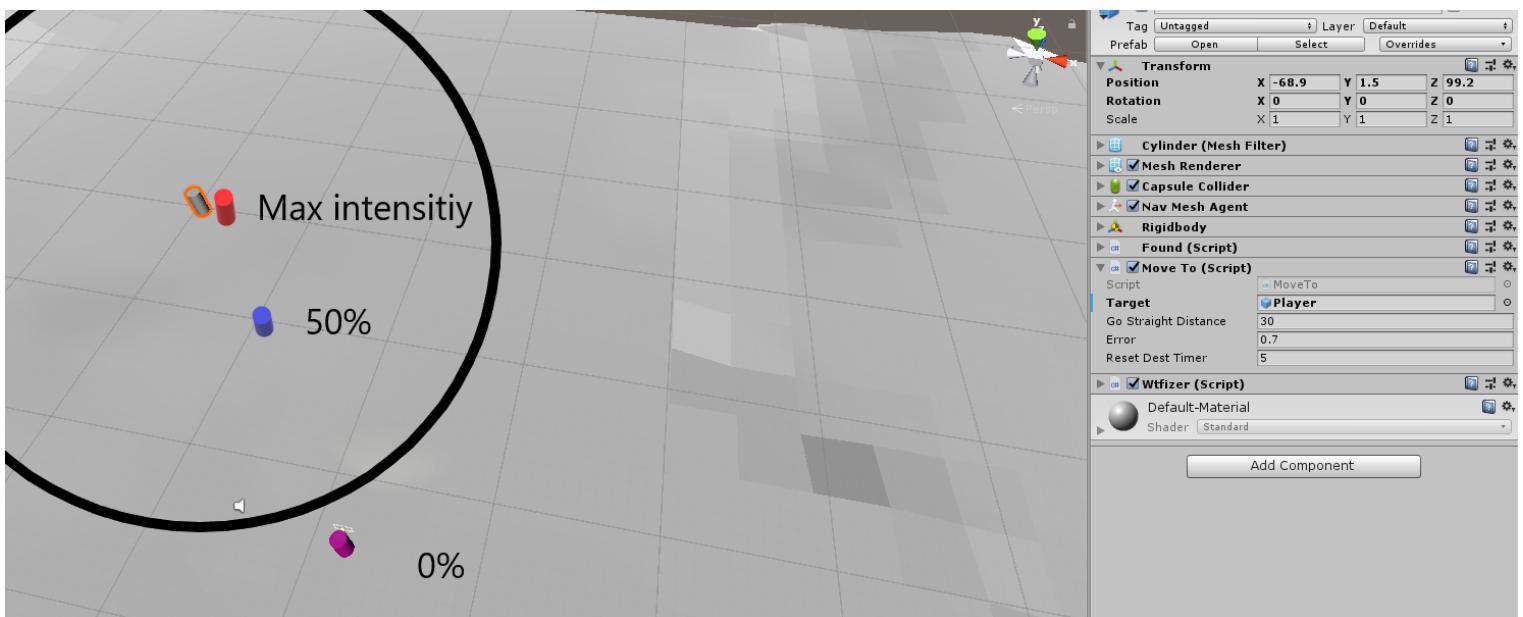


Veuillez noter que les graphismes ne sont pas ceux de la dernière version du jeu. J'ai également ajouté une fonctionnalité qui distord le son quand le monstre s'approche du joueur. Cela crée une sensation de pression qui augmente dans la musique quand le monstre est sur le point de tuer le joueur. Ce script accepte 3 paramètres :

- Threshold : la distance à partir de laquelle les effets s'activent
- Maxchorus : l'intensité maximale de l'effet chorus
- Maxdisto : l'intensité maximale de l'effet distorsion

Le son est "dry" quand la distance entre le joueur et le monstre est supérieure au threshold. Quand le monstre et le joueur sont proche, l'intensité des effets est

proportionnelle à cette même distance, jusqu'à atteindre les valeurs maximales quand les deux objets se touchent.



3.4.3 IA

L'IA qui gère les déplacements d'un ennemi fonctionne de la manière suivante : Elle accepte 3 paramètres,

- Go Straight Distance : distance à partir de laquelle l'IA suit le joueur sans faire d'erreurs.
- Error : intensité des erreurs commises par l'IA
- Reset Dest Timer : durée entre les changements de destination de l'IA.

Le monstre choisit une destination basée sur la position du joueur, avec une erreur plus conséquente plus il est loin du joueur. Cela a pour effet que les déplacements du monstre semblent assez aléatoire en début de partie, quand l'IA est loin du joueur. L'IA se rapproche du joueur plus ou moins rapidement en fonction du taux d'erreur et du délai entre chaque changement de trajectoire.

Quand la distance qui la sépare du monstre est inférieure au threshold, l'IA suit le joueur sans se tromper jusqu'à l'attraper ou bien que le joueur la sème. Les fonctionnalités des NavMesh dans Unity prennent en charge le pathfinding entièrement. En réalité, dans le gameplay, l'IA est comme un timer : elle se rapproche peu à peu du joueur, et si jamais ce dernier est bloqué trop longtemps à un endroit, alors il pourra difficilement lui échapper.

3.4.4 Multijoueur

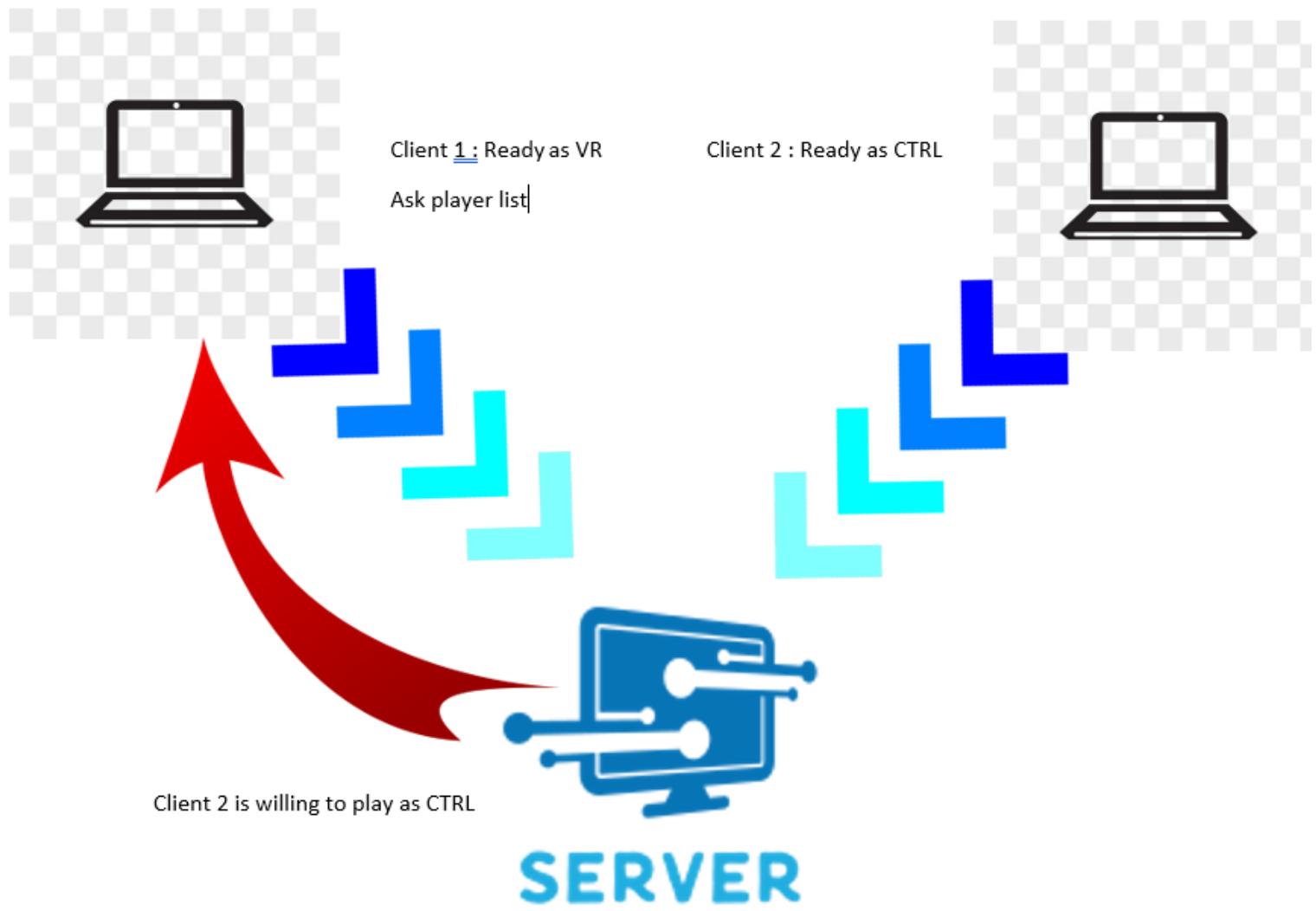
Les premières versions du Multijoueur étaient faites d'une simple connection socket, et ne fonctionnaient qu'en réseau local. Cependant il est préférable de pouvoir jouer en ligne ; j'ai donc décidé de changer complètement le fonctionnement du multijoueur. Le multi fonctionne désormais avec un serveur, qui permet tout d'abord de jouer hors réseau local, mais aussi de récupérer des statistiques et de les communiquer avec le site web. Le serveur reste assez simple, car même si le jeu est en temps réel un petit délai n'a aucun impact négatif sur le gameplay des joueurs : le joueur VR ne peut pas avoir de problèmes à cause du réseau, et les actions du joueur contrôle ne nécessitent pas de temps de réponses infimes comme sa serait le cas avec un jeu de tir multijoueur par exemple. Une partie en ligne fonctionne de la manière suivante :

Le client, lorsqu'il rejoint le menu de recherche de lobby, envoie son identifiant au serveur, lui déclarant par la même occasion s'il joue en mode VR ou bien contrôle (mode CTRL). Le client est ensuite considéré comme quelqu'un voulant lancer une partie par le serveur.

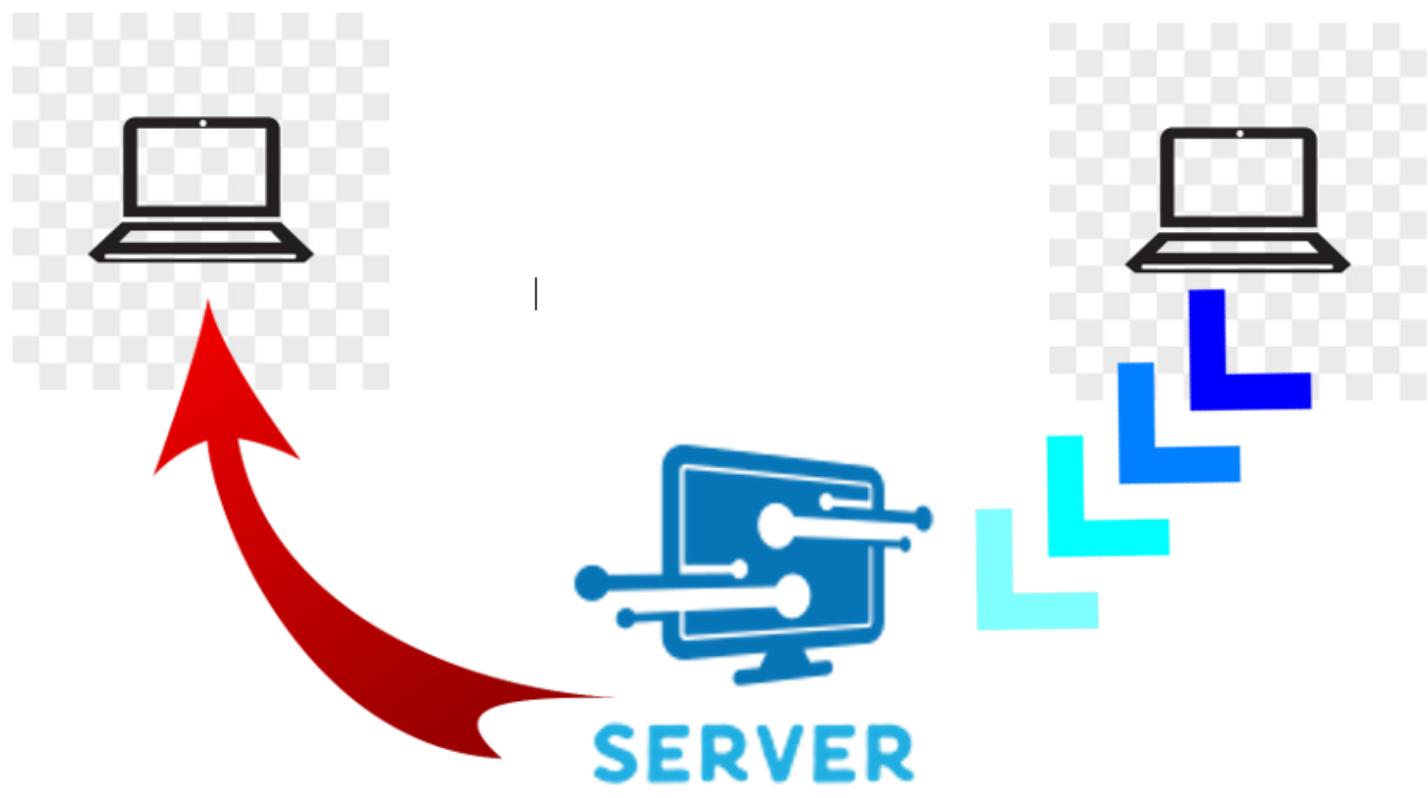
Il peut aussi demander au serveur la liste de tout les joueurs souhaitant jouer, c'est à dire tout les joueurs CTRL pour un joueur VR et inversement. Le serveur envoie les éléments uns à uns.

Ensuite, il peut demander au serveur de se connecter avec un adversaire. Si l'autre joueur accepte, le serveur lance une partie : il retire les deux joueurs de la liste des joueurs en recherche de partie, et agit désormais comme relais entre les deux joueurs, renvoyant les commandes aux bons clients sans que cela ait besoin d'être précisé à chaque fois.

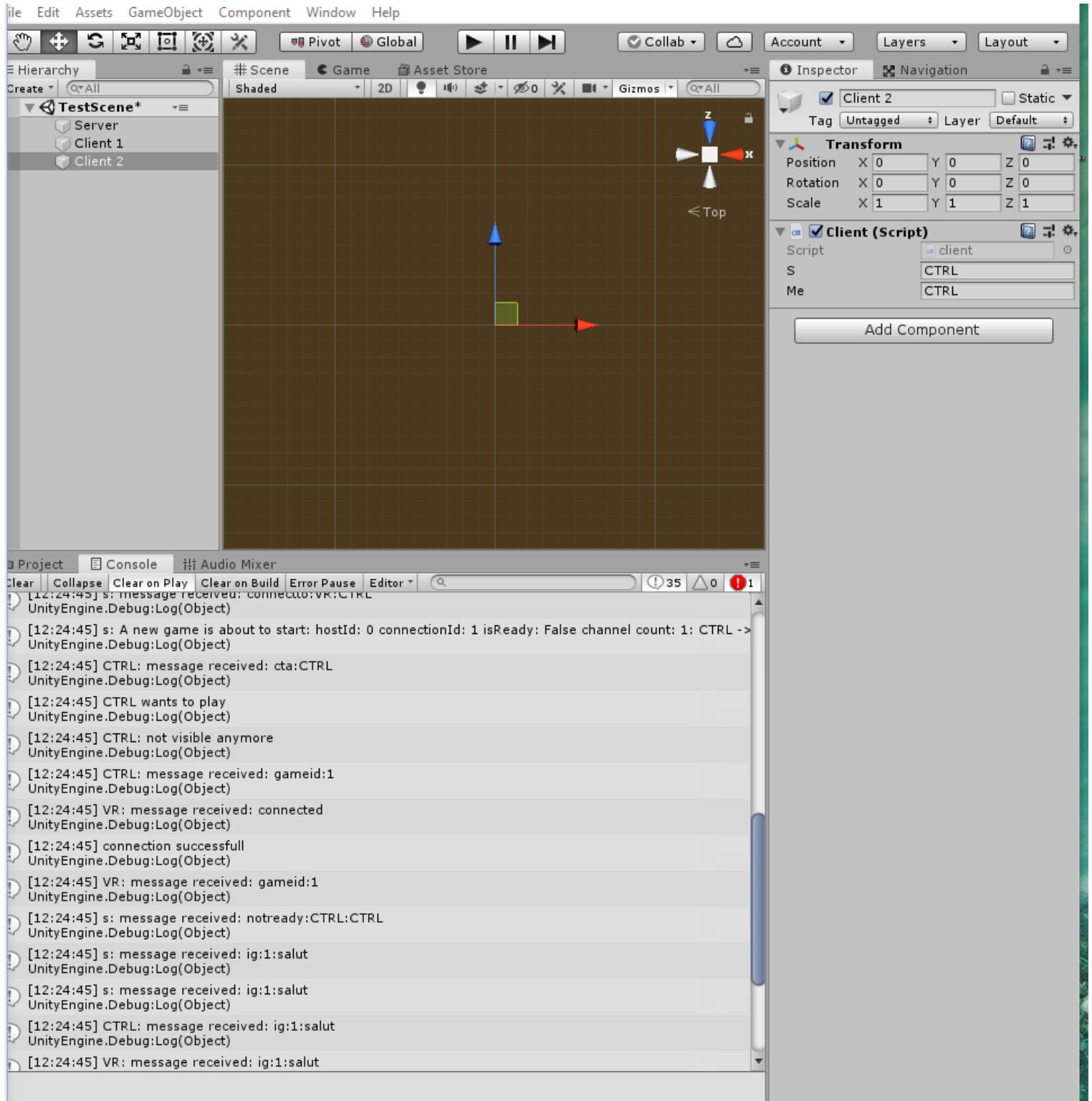
Les clients peuvent aussi envoyer diverses informations sur la partie, comme le vainqueur par exemple, qui seront stockées par le serveur. Le serveur a donc potentiellement accès à toutes les informations du jeu.



Dans une premier temps, le serveur et les clients communiquent de la manière décrite ci-dessus.



Ensuite, le serveur transmet directement les messages reçus au bon client.



Ci-dessus, deux clients communiquent entre eux, sans qu'ils ne s'envoient de message directement ; c'est le serveur qui relaye tout, comme expliqué plus haut. L'intérêt premier est le suivant : Pour protéger les périphériques du réseau local, un routeur ne permet pas par défaut aux appareils connectés d'être visible depuis l'extérieur. C'est pour cette raison qu'il n'est pas possible de communiquer directement entre deux clients. Toutefois, les clients peuvent se connecter sans problèmes à un serveur si celui-ci est visible depuis l'extérieur. J'ai donc ouvert les différents ports utilisés par le serveurs sur le routeur pour les rediriger vers le serveur. N'importe qui peut communiquer avec le serveur, qui joue donc le rôle de relais entre des clients qui ne peuvent normalement pas se connecter entre eux.

De plus, le serveur permet de récupérer autant d'informations sur la partie que souhaité, même en temps réel, et de les communiquer au site web : on peut ainsi afficher les vainqueurs, faire des classements, voir utiliser ces informations pour améliorer par exemple l'équilibrage...

Pour plus de détails, le serveur et les clients communiquent en UDP sur le port 433, en utilisant l'implémentation des sockets conçue par Unity. Le multi-threading est déjà géré par Unity, facilitant grandement l'implémentation du serveur à l'aide d'évènements pré-intégrés au serveur. Les messages échangés sont des chaînes de caractères contenant une commande et des arguments, séparées par le caractère ':' . par exemple, la commande "ready :VR :pseudo" signifie que le joueur 'pseudo' est prêt, et qu'il souhaite jouer en mode VR. Une fois une partie lancée, le serveur crée un objet de type Game, c'est à dire une partie. Si un joueur est dans une partie, quand il envoie une commande "ig :commande", la commande est directement transmise par le serveur au 2ème joueur dans l'objet Game, jusqu'à ce que la partie se termine. Les identifiants des joueurs sont stockés dans une base MySQL.

3.4.5 Site web

Le site Web était réalisé avec le site Wix.com, qui automatise une grande partie du processus de conception. Pour rappel, il était prévu de faire un site plus complet à la fin du projet si le temps le permettait (le site n'étant pas notre première priorité). J'ai donc refait un site web, qui offre un avantage important par

rapport à un simple site Wix : le site web est hébergé directement sur le serveur du jeu, ce qui permet de mettre en commun les bases de données du site et du jeu. De ce fait, il est donc possible de créer son compte sur le site, et de l'utiliser ensuite dans le jeu. De plus, le site peut comptabiliser les victoires et tenir un historique des parties, afficher des statistiques sur le jeu (nombre de joueurs, nombre de joueurs connectés, en partie, en recherche...)

Le nouveau site Web, réalisé avec python-django, est hébergé sur le même serveur que celui du jeu. L'avantage étant qu'il suffit d'écrire dans la base MySQL, avec le serveur Unity, et on peut y accéder directement avec python. De la même manière, tout ce qui est fait sur le site web est accessible pour le serveur, et donc potentiellement aussi pour le client. Le site web permet donc de créer un compte. Ces données sont stockées dans une base MySQL, et ne sont pas encryptées. Un compte TheMaze ne présente aucun intérêt pour quelqu'un de mal intentionné, et ces données sont contenues sur le serveur et ne sont pas accessibles. Ensuite, quand on joue en utilisant ses identifiant, les parties sont comptabilisées. Les statistiques sont constamment mises à jours, au fil des parties, et accessibles sur le site. On crée donc bien un seul compte TheMazeVR, qui permet d'accéder à la fois au site, au jeu, et à ses stats.

Pour le reste, le site est constitué de page explicatives ou de liens, permettant de télécharger les différents rapports de soutenance et évidemment l'installateur du jeu.

Le site est donc une combinaison d'HTML, de CSS et de python pour gérer les actions un peu plus complexes (les comptes, les stats...)

4 Conclusion

En conclusion, nous avons porté ce projet bien plus loin que nous ne l'imaginions. En effet, notre idée de départ était simplement une suite de labyrinthes, sans monde autour, or nous avons aujourd'hui une carte très grande pour un projet de cette taille, des objets, un éditeur de carte et du multijoueur. Nous sommes donc allés beaucoup plus loin que prévu, grâce aux outils que nous avions à notre disposition.

5 Annexes

5.1 Hugo

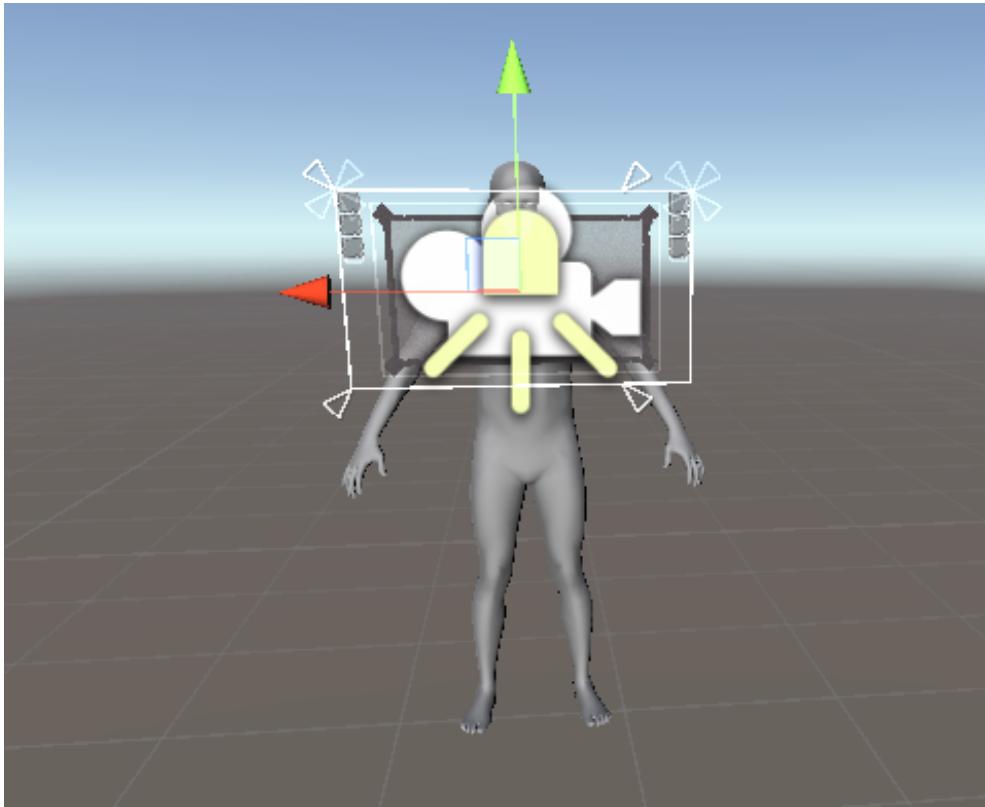


FIGURE 1 – Le joueur



FIGURE 2 – Game Over

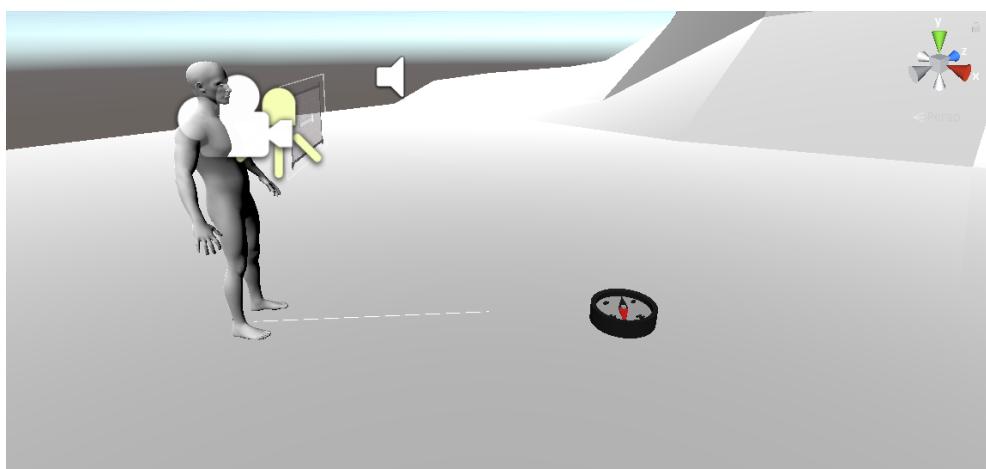


FIGURE 3 – Le joueur et un objet

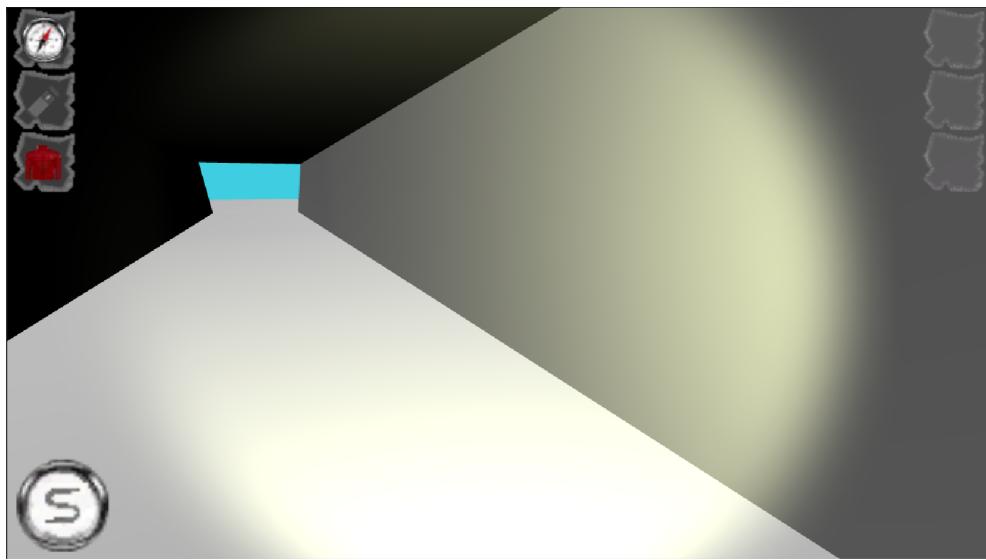


FIGURE 4 – Les trois objets ont été récupérés

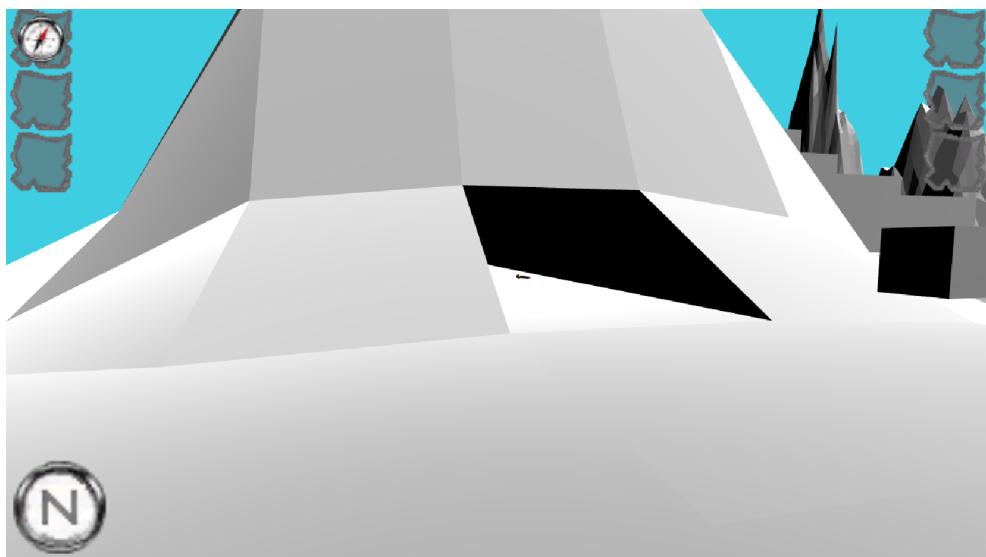


FIGURE 5 – L'affichage de la boussole



FIGURE 6 – Le menu principal

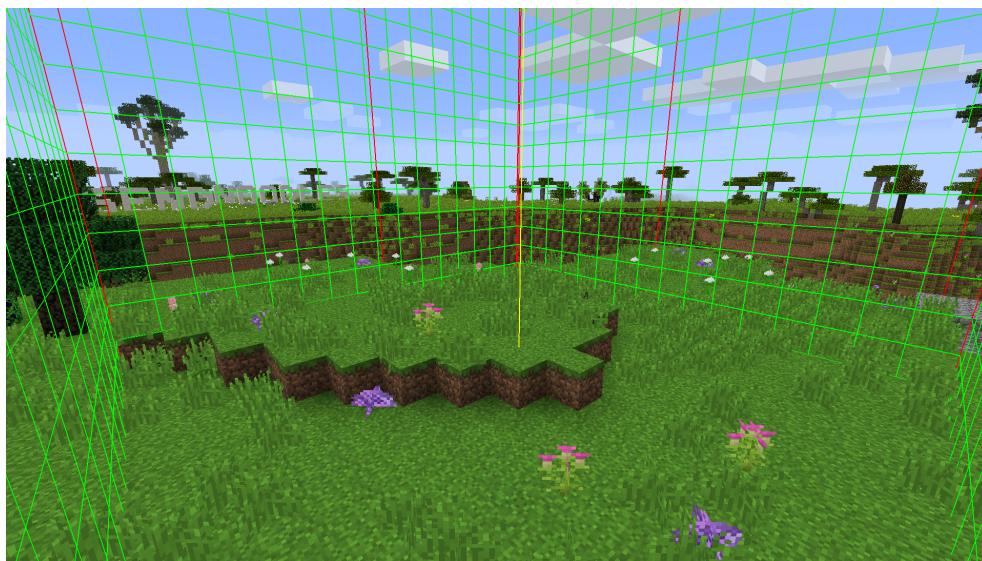


FIGURE 7 – La grille d'un chunk dans Minecraft

5.2 Geoffroy

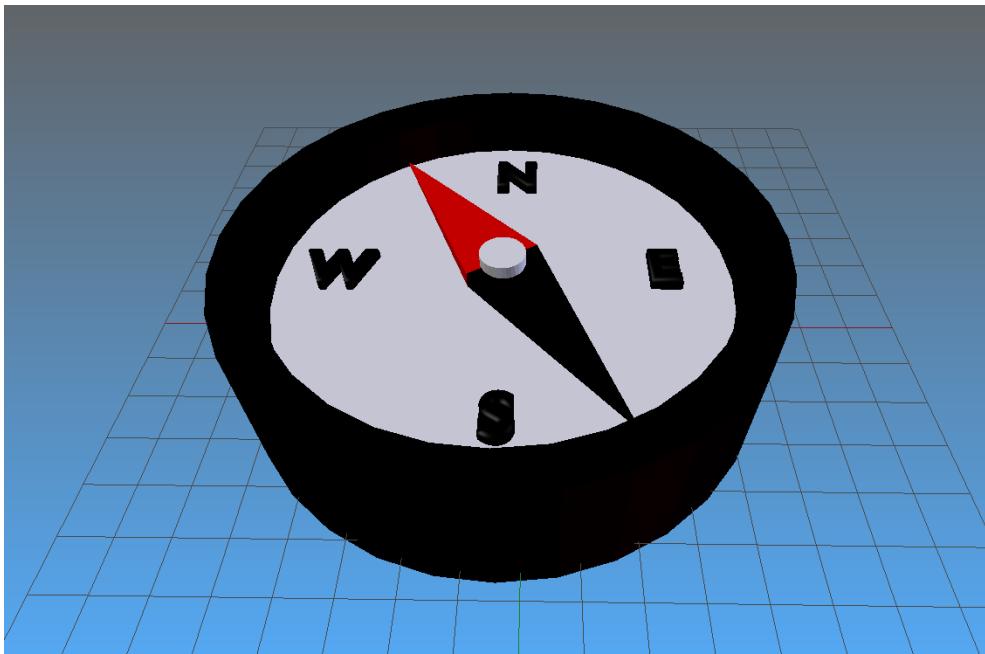


FIGURE 8 – Boussole

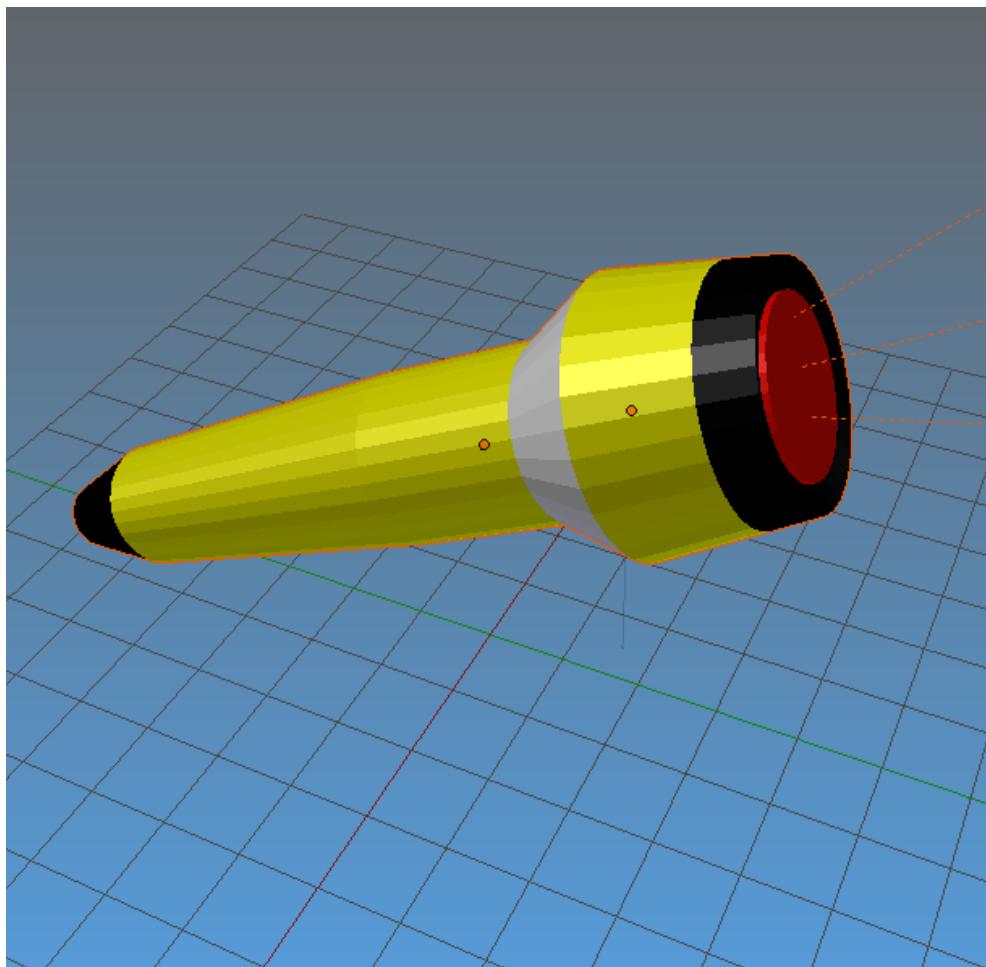


FIGURE 9 – Lampe torche



FIGURE 10 – Échelle

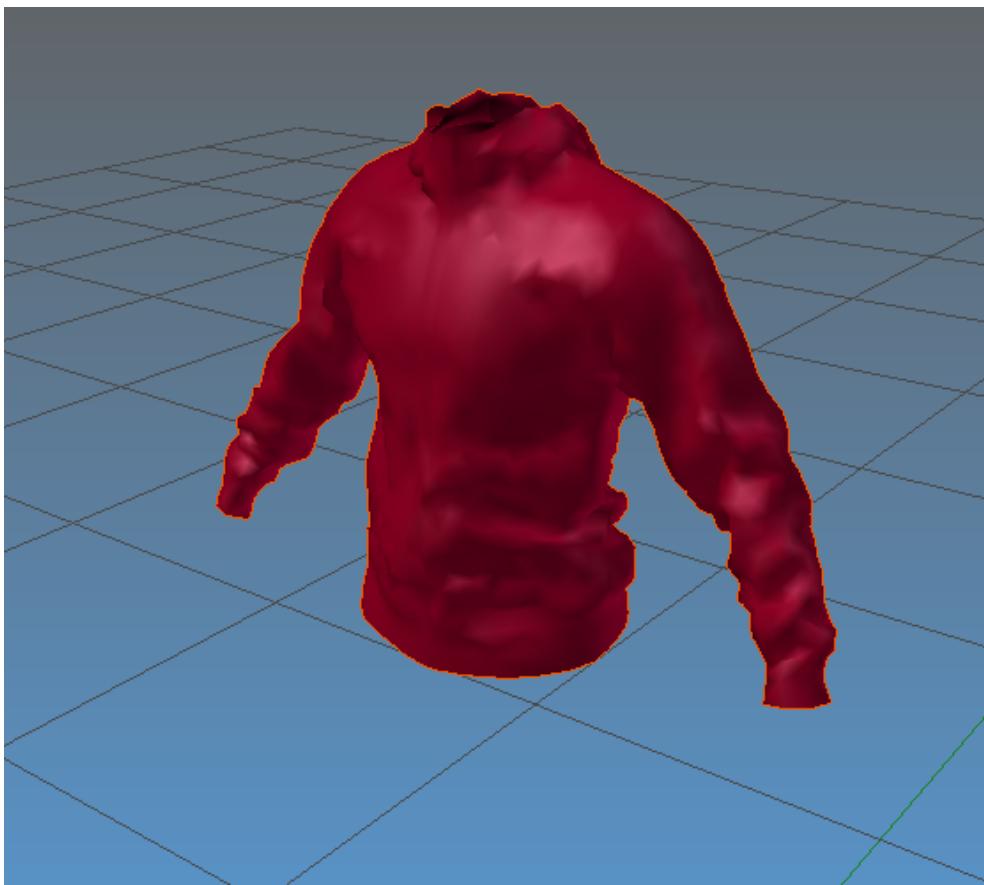


FIGURE 11 – Vêtements Chauds

5.2.1 SAND

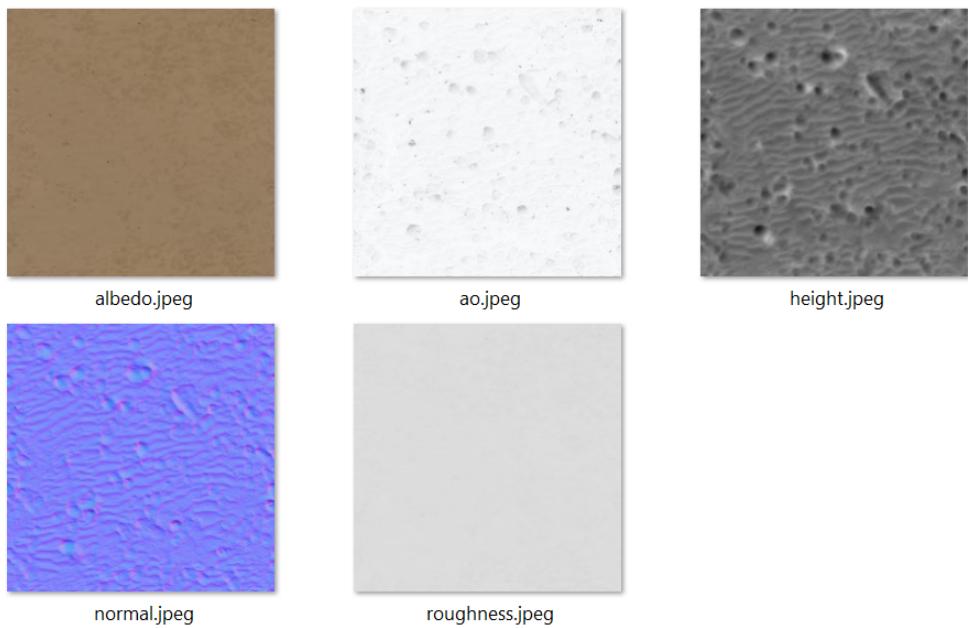


FIGURE 12 – Les fichier enregistré en .JPEG

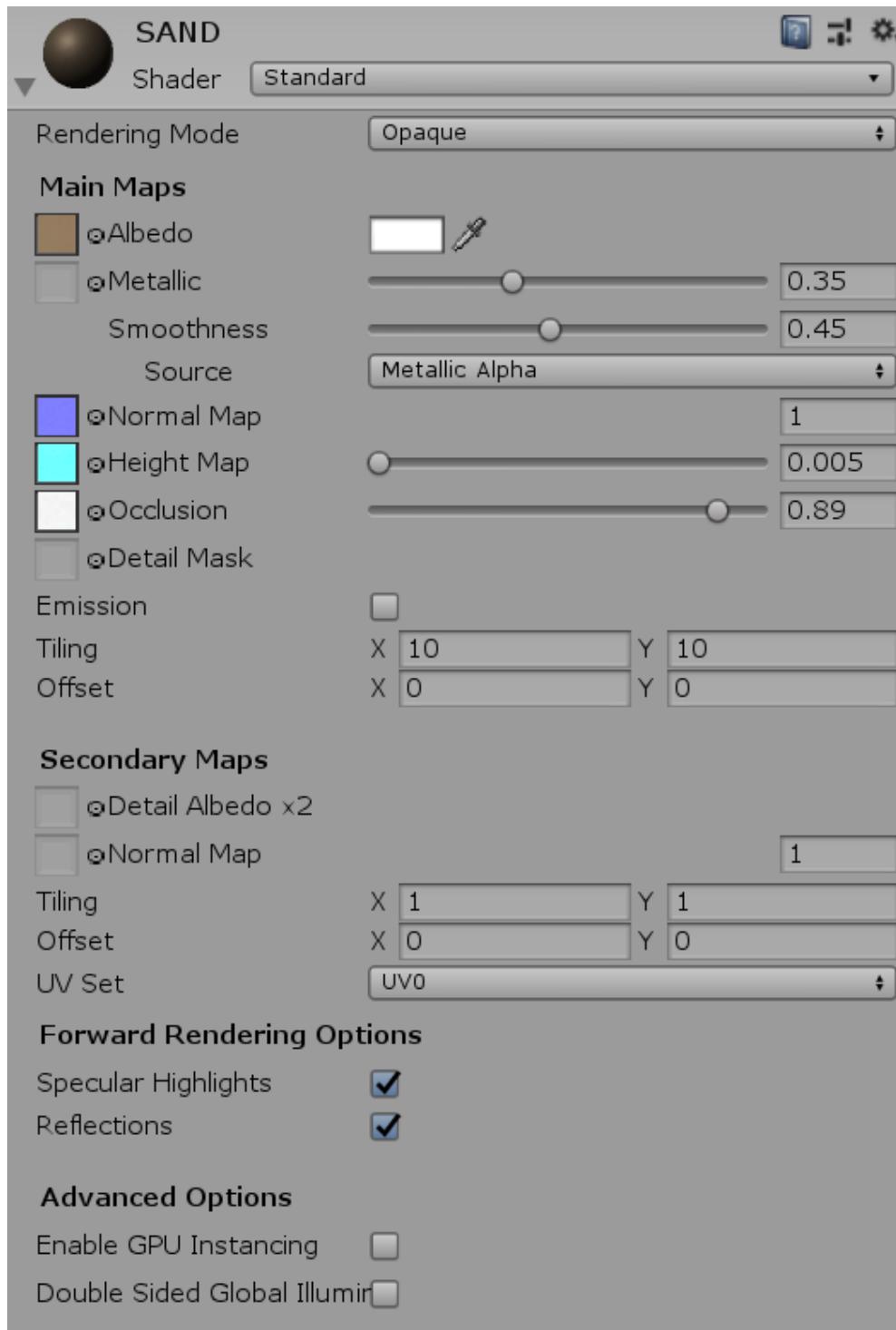


FIGURE 13 – Système de gestion des textures sur Unity

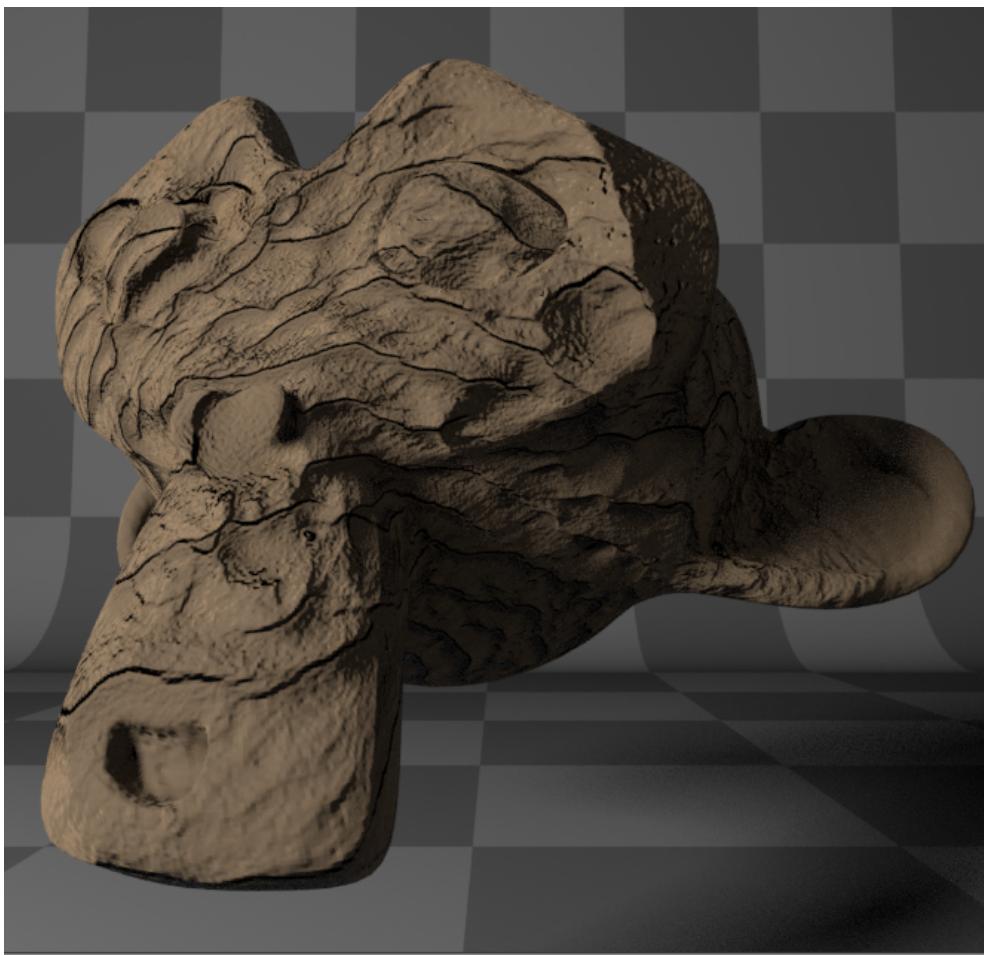


FIGURE 14 – Rendu du sable sur objet

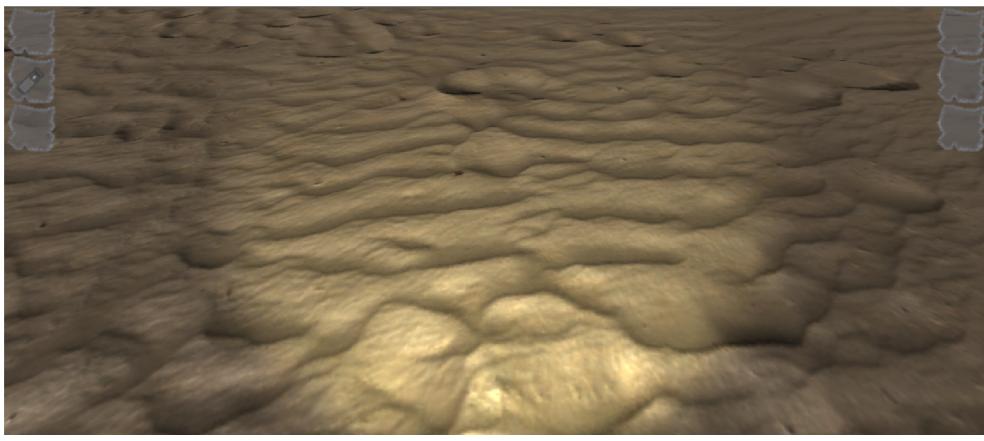


FIGURE 15 – Rendu final en jeu

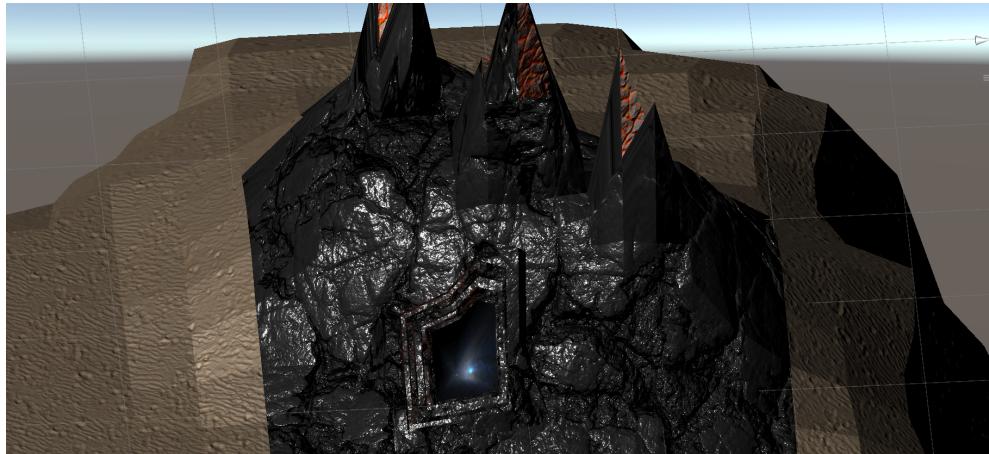


FIGURE 16 – Rendu des Textures finales 1

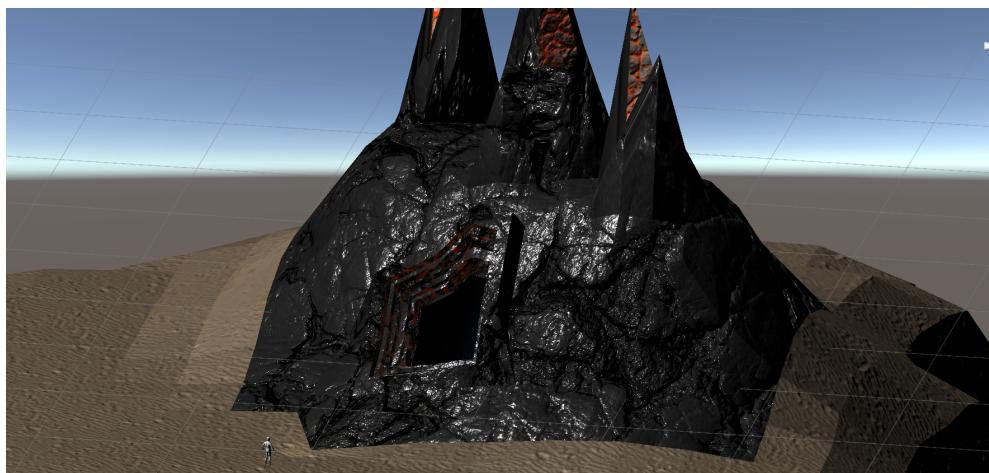


FIGURE 17 – Rendu des Textures finales 2

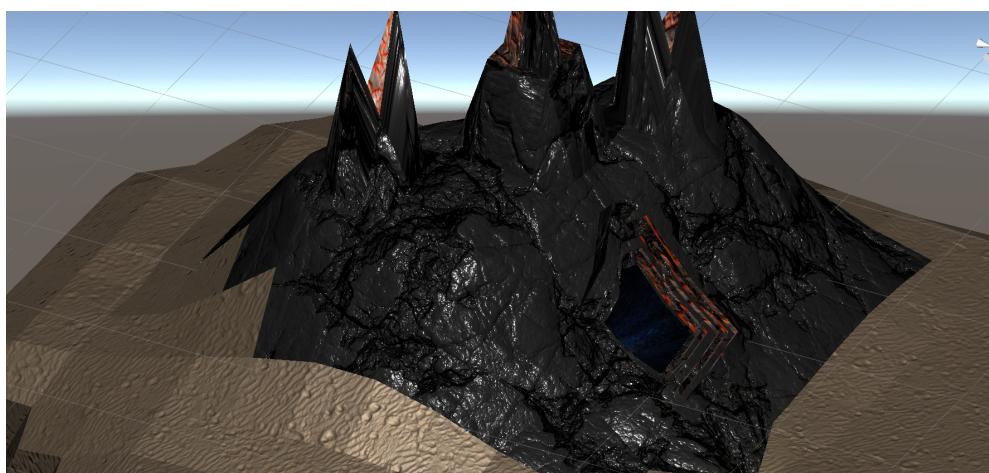


FIGURE 18 – Rendu des Textures finales 3