SSAO:
Rapport de Projet

Groupe : **Les Rejetés**

Ouwéis MOOLNA, Edouard EISENFISZ, Lucas PINOT, Simon QUESNEY

25 Mai 2018

Table des matières

1 Introduction Du Rapport Final

Cela fait maintenant 6 mois que nous travaillons sur ce projet. Durant cette période, nous vous l'avons présenté à plusieurs reprises, lors des précédentes soutenances.

Nous allons aujourd'hui, dans ce rapport final, revenir sur tout ce que nous avons réalisé depuis le début de ce projet.
Nous reprendrons d'abord une partie du cahier des charges avant de vous présenter et détailler tout ce dont nous avons réalisé jusqu'à présent dans les différentes parties de notre jeu.

2 Présentation Du Jeu

SSAO sera un jeu de tir en coopération. Ce sera un jeu à la première personne, et les joueurs pourront y jouer de 1 à 4 personnes.
Les joueurs devront évoluer dans un environnement et affronter différents types d'ennemis afin d'atteindre certains objectifs. Ces objectifs seront de différentes natures et nous les détaillerons dans ce cahier des charges. L'environnement sera un monde mêlant le style fantastique et le style steampunk.
En effet, le monde en lui-même sera dans le style steampunk, mais plusieurs caractéristiques des mondes fantastiques seront présents, comme la magie. Les joueurs auraient des armes propres à leur personnage, mais pourraient en obtenir d'autres, ou avoir leurs statistiques améliorées s'ils trouvent des objets cachés dans les différentes cartes où ils évolueraient.
Les joueurs devront veiller à ne pas toucher leurs partenaires, le tir allié étant activé, car ceci pourrait les tuer et bloquerait leur progression dans le jeu ou le niveau. Effectivement, certains objectifs nécessiteront la présence de tous les membres de l'instance pour pouvoir être réalisés.
Les mondes du jeux évolueront au cours du temps. Par exemple, au début, les joueurs n'auront que peu de choix pour réaliser les objectifs, comme devoir combattre les ennemis pour atteindre un objectif ou terminer le niveau. Mais au fur et à mesure qu'il avanceront, les choix se diversifieront ; par exemple ils pourront combattre les ennemis, passer discrètement sans se faire repérer, ou même prendre d'autres chemins détournés leur proposant d'autres choix pour passer.

3 Équipe

— **Edouard EISENFISZ**, A.K.A "FireFox"
J'ai toujours la pêche et une coupe entre le feu vivant et un nuage.
J'ai déjà fait un projet l'année dernière, et j'ai grandement apprécié. Je suis un grand fan des jeux Bethesda, et des jeux en coopération, comme Warhammer : Vermineville ou Dishonored. J'ai déjà utilisé Blender et Unity.
L'idée de ce projet étant le fruit d'un partage de nos envies et de nos idées respectives, nous avons tous été à l'origine de l'idée de ce projet.

— **Simon QUESNEY**, **Chef de Projet**, A.K.A "Qui-Gon-Jinn"
"Le dernier Star Wars c'est pas très très bien pour un Star Wars"
FAN des jeux Bethesda (j'aime les bugs), notamment les jeux solo (Dishonored 1 et 2, Skyrim, Fallout), et autres jeux solo comme Bioshock ou The Witcher.
Je compte m'améliorer en C# dans ce projet en abordant des sujets complexes comme l'IA ou la mise en place de la furtivité tel un Dishonored.

— **Lucas PINOT**, A.K.A "PublicStaticVoid"
Je suis moi aussi un fan des jeux Bethesda, plus particulièrement de la saga The Elder Scrolls. Ayant déja fait un jeu l'année dernière en projet d'ISN en Python, je compte faire beaucoup mieux cette année en passant de la 2D à la 3D.

— **Ouwéis MOOLNA**, A.K.A "Bûcheron"
Je suis le seul qui n'ai jamais joué à un jeu Bethesda, sauf le dernier DOOM, bien que j'ai beaucoup entendu parler de leurs jeux. J'ai cependant joué à Bioshock et plus particulièrement Infinite que j'ai beaucoup apprécié et qui nous a aussi influencé dans certains choix pour notre jeu.
Avec ce projet j'espère progresser dans le C# mais aussi apprendre à utiliser de nouveaux outils comme Unity.

4 Distribution

Tâches	Ouwéis	Lucas	Edouard	Simon
Fonctionnalités obligatoires				
Contrôle basiques :	Chargé	Suppléant		
Level Design :			Suppléant	Chargé
IA, et mouvement des ennemis :		Suppléant		Chargé
Graphisme :		Suppléant	Chargé	
Sons :	Chargé			Suppléant
Contrôles avancés :		Chargé		Suppléant
Rapports de Projets	Chargé		Suppléant	
Site :			Suppléant	Chargé
Réseau :	Suppléant		Chargé	
Procédure d'installation :	Suppléant	Chargé		
Fonctionnalités Bonus				
Animation :	Chargé	Suppléant		
Menu :		Chargé	Suppléant	
Mode Furtif :		Suppléant		Chargé
Boss :	Suppléant	Chargé		
Code de triche :	Suppléant		Chargé	
Versus :			Chargé	Suppléant
Canon à ennemis :		Suppléant	Chargé	

5 Progression

module	**oral** 1	**Soutenance 1**	**oral 2**	**Soutenance 2**	**Prévision**
Contrôle de base	75%	En avance	100%	A l'heure	100%
Level design	33.33%	A l'heure	66.66%	A l'heure	?%
IA, et mouvement des ennemis	22.22%	A l'heure	56.65%	A l'heure	100%
Graphisme	25%	A l'heure	60%	A l'heure	100%
Animation	20%	A l'heure	60%	A l'heure	80%
Menu	0%	_	50%	A l'heure	100%
Sons	20%	A l'heure	60%	En retard	100%
Site	10%	A l'heure	65%	A l'heure	80%
Réseau	40%	A l'heure	70%	A l'heure	90%
Procédure d'installation	0%	_	0%	_	100%
Mode furtif :	0.0001%	_	42.123%	En avance	100%
Versus :	20%	A l'heure	60%	A l'heure	?%
Contrôles avancés	20%	A l'heure	50%	A l'heure	100%
Canons à ennemis :	0%	_	30%	-	-%
Boss	0%	_	20%	-	100%
Code de triche	20%	-	50%	-	-%

6 Le Projet

6.1 Les Contrôles

6.1.1 Contrôle Basique

Les contrôles de base sont les déplacements du personnage, que ce soit pour avancer, reculer, aller à droite, à gauche ou bien sauter. Mais aussi le déplacement de la caméra par rapport au curseur. Les contrôles s'occuperont les claviers et souris mais aussi les manettes.

6.1.2 Contrôle Avancé

Les contrôles avancés sont tous les contrôles concernant les attaques mais aussi les lancers de sort. Ils seront comme pour les contrôles de bases implémentés aussi bien sur clavier et souris que sur manette.

6.2 Ennemis

6.2.1 Les Types d'Ennemis

Les Ennemis De Vague

Les ennemis de vague seront les ennemis de base. Ils apparaîtront en groupe et sous forme de vagues. Ce seront les ennemis les plus faibles du jeu sans compétences spéciales en dehors de leurs attaques de base.

6.2.2 L'IA Ennemie

L'IA Des Ennemis En Vague

L'IA la plus simple ici va seulement avoir pour but de trouver le joueur le plus proche et de l'attaquer ; pour ce faire elle va connaître sa position au préalable et juste avancer dans sa direction, dépendant des ennemis elle va soit le cibler à distance, soit l'attaquer au corps-à-corps.

6.3 Graphisme

6.3.1 Level Design

Il y aura différents niveaux adaptés à chaque type de missions et modes de jeux. En effet, le mode Versus ne demandera pas une grande zone de jeu ; celle-ci sera basique et ne comportera sûrement que des obstacles grâce auxquels les joueurs pourront se protéger des tirs de leur adversaire. Les missions à objectifs, seront créés autour de ces objectifs. Par exemple, si l'objectif est de protéger un point, ce point sera au centre de la carte. A l'inverse, une mission où les joueurs devront atteindre un point ; la carte correspondante positionnera les joueurs aux points les plus éloignés du point à défendre.

Ces différents types de carte seront dans des environnements urbains jouant entre un style londonien et un style médiéval, et sur des effets de machine à vapeur créant un ensemble appartenant à une architecture de type steampunk.
Les cartes à objectifs, auront également des paliers séparés par des portes cassables. Il sera nécessaire de casser celles-ci pour continuer. Ces cartes auront également des salles cachées avec des pièges contenant des bonus facilitant la suite du niveau.

6.3.2 Apparence Des Joueurs

Les joueurs auront chacun une apparence unique dans un design steampunk, mais nettement différente des autres et surtout des ennemis. Leurs caractéristiques seront en fonction de leur rôle. Par exemple, le guerrier sera petit et gros car il sera un nain, tandis que le mage sera fin et grand. Leur accoutrement sera également, toujours dans le style steampunk, adapté en fonction de leur classe. Pour l'instant nous n'avons qu'un humanoïde quelconque.

6.4 Sorts

6.4.1 Types de Sort

Les Attaques De Base et Objets Normaux

Les attaques de base seront les attaques normales d'un personnage, l'utilisation d'une attaque de base n'entraîne pas de temps de rechargement ; elle peut donc être utilisée à volonté et aussi souvent que le joueur le voudra.
Les attaques de base pourront être améliorées ou totalement modifiées par l'acquisition d'un objet normal, comme un fusil à pompe pour le Combattant ou un bâton lanceur de flammes pour le Mage Guerrier.

Les Sorts d'Objets Magiques Ou Technologiques

Les objets magiques ou technologiques seront acquis au fil de la partie pendant chaque niveau, et ce définitivement. L'acquisition de ces objets octroiera un nouveau sort au joueur l'ayant récupéré permettant une modification du gameplay du personnage.
N'importe quel personnage pourra récupérer n'importe quel objet, en contrepartie cet objet devra obligatoirement être récupéré pour terminer un niveau ou réussir un objectif du niveau ; obligeant les joueurs à choisir plus ou moins soigneusement le personnage qui recevra l'objet en question et à réussir un objectif intermédiaire du niveau, qui sera donc l'acquisition de cet objet.
Le joueur ayant récupéré un objet durant un niveau recevra en conséquence moins d'expérience en échange de pièces d'or servant à améliorer le ou les objet(s) récupérés au cours de la partie.

Un objet utilisé aura un temps de rechargement avant de pouvoir être réutilisé à nouveau.
La bombe jetable peut par exemple être un objet technologique dont l'utilisation entraînera un temps de rechargement avant réutilisation.

6.4.2 Animations Des Sorts

Les animations des sorts seront toutes les animations en rapport avec les sorts, et elles seront différentes en fonction du sort choisi, par exemple les sorts physiques n'auront pas la même animation que les sorts magiques. Il y aura également une animation sur le personnage quand les sorts auront fini d'être rechargés.

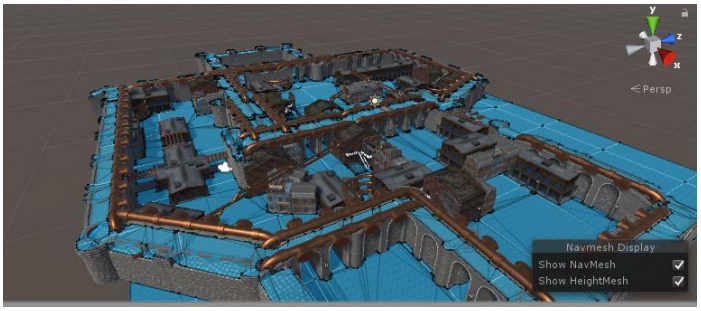
7 Présentation du Travail Réalisé

7.1 Level Design et Graphismes

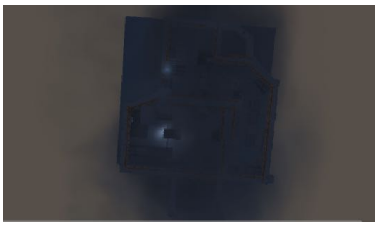
Nous avions prévu pour ce projet plusieurs zones jouables (appelées maps) qui seront toutes différentes dans leur design, ainsi que de nombreux modèles pour rendre le jeu plus riche. Nous vous présenterons ici les niveaux développés, ainsi que certains des modèles développés.

7.1.1 La Première Map

Nous avions pour projet au départ de faire plusieurs modèles via Blender. Nous nous sommes vite rendus compte que les textures que nous avions faites dans Blender n'avaient pas du tout le même rendu sur Unity. En effet, les objets avaient un effet transparent. Nous avons donc dû utiliser de nombreux assets pour faire la map. Il n'y a pas en tant que tel d'asset dans le style steampunk, nous avons donc dû associer différents modèles qui, pris seuls, n'ont aucun rapport avec le thème steampunk. C'est un style londonien et médiéval à la fois. La map ressemble à cela (avec navmesh).



Nous avons ajouté un brouillard par dessus la carte pour donner et supporter l'ambiance steampunk de notre jeu. Voici quelques images sur l'apparence de la ville.

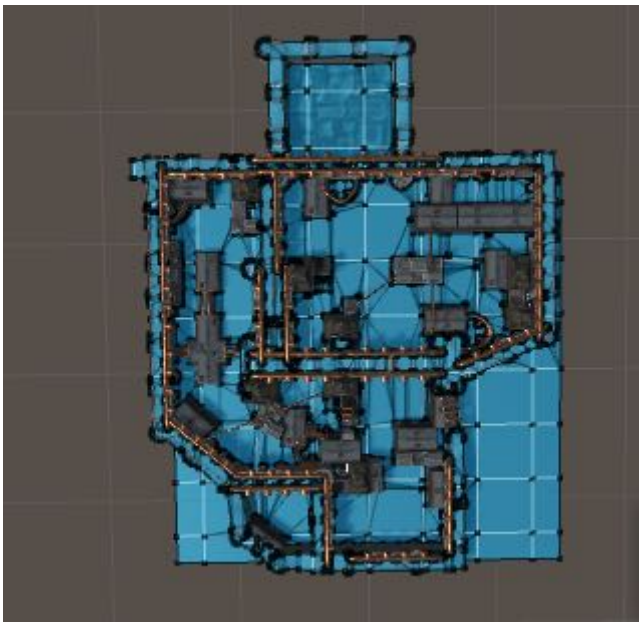


Ceci est l'aspect aérien de la map une fois complète.

La vue à partir de la terrasse d'un bâtiment.

Une petite ruelle.

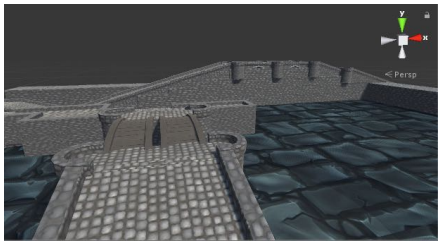
Nous avons eu quelques problèmes pour ce qui est du navmesh. En effet, afin de permettre le passage sous les ponts, nous avons du modifier ceci car pour des raisons de temps de calcul, nous ne pouvions nous permettre d'avoir une map trop grande, et nous étions limités par les capacités de Unity.



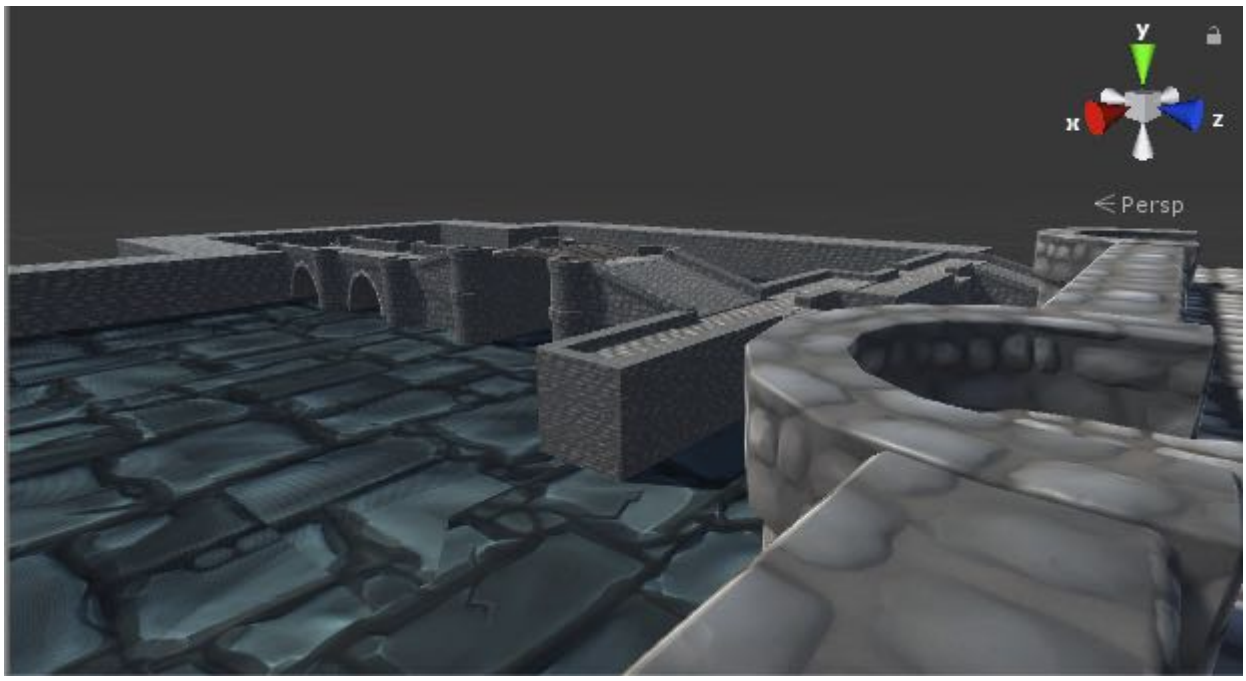
Dans l'idée, les chemins de la map sont comme dans l'image ci-dessous.

7.1.2 La Deuxième Map

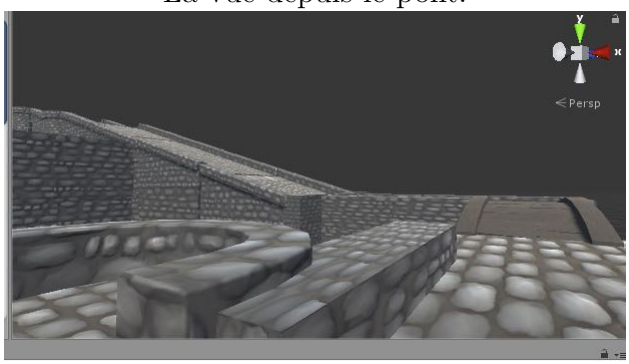
Nous nous sommes lancés, pour la deuxième soutenance, dans la création d'une nouvelle map d'un style différent. Nous avons voulu faire une map ou les joueurs avaient le choix entre se déplacer en hauteur, sur un chemin où il n'y aura pas beaucoup d'ennemis, mais qui sera rempli de pièges ; et un chemin sans pièges, mais qui serait lui rempli d'ennemis. Nous avons eu beaucoup de problèmes avec le navmesh des passages en hauteur.



Au départ, les joueurs ont un chemin obligatoire puis doivent prendre un grand pont qui leur permet de voir le reste de la zone. Il y aura dans ce cas des ennemis et également des pièges mais leur nombre sera réduit afin de laisser au groupe la possibilité de faire son choix après.

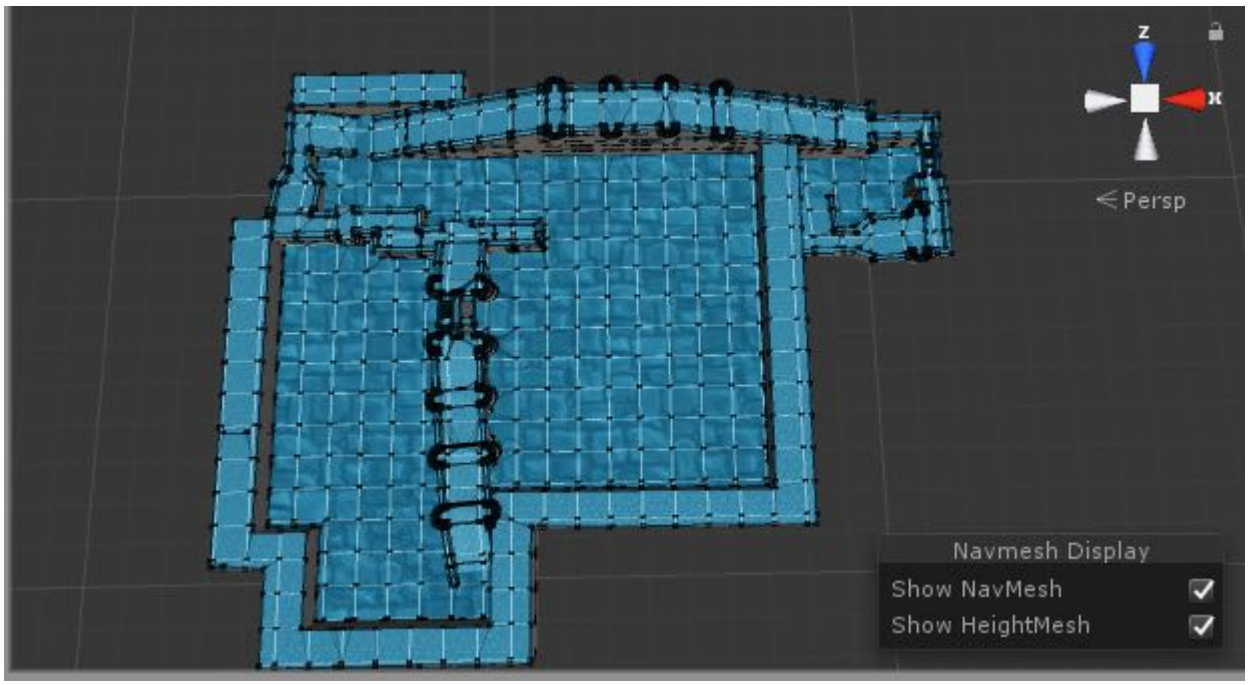


La vue depuis le pont.

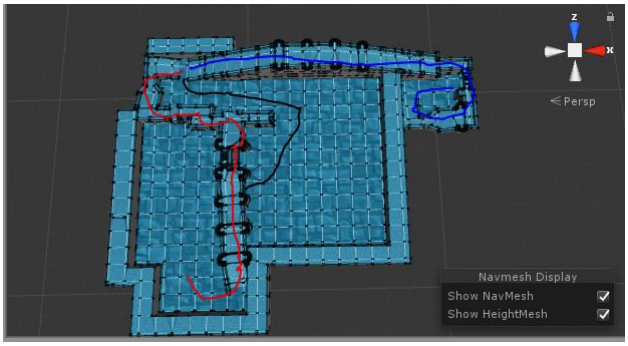


Le chemin de départ.

Nous avons eu quelque problèmes pour ce qui est du navmesh. En effet, pour permettre que l'on puisse passer sous les ponts, nous avons du faire de nombreux tests en changeant les paramètres, ou en modifiant drastiquement l'aspect de la map que nous faisions.



Dans l'idée, les chemins de la map sont comme dans l'image ci-dessous. Le chemin bleu est le chemin de départ. Le chemin rouge est le chemin en hauteur rempli de pièges et le chemin noir est le chemin avec une horde d'ennemis.



7.1.3 Pièges et modèles

Pour la première soutenance, nous avons fait des scripts pour permettre d'avoir des tonneaux explosifs. A ce moment, nous n'avions pas encore certains scripts, mais nous avions besoin de pouvoir tester les tonneaux.

```
using UnityEngine;
using System.Collections;

public class Tonneau : MonoBehaviour
{
    public Transform Trans;
    public Explosion expl;
    public Rigidbody rig = new Rigidbody();
    private int health;
    void Start()
    {
        pos = rig.transform.position;

    }
    void OnTriggerEnter(Collider other)
    {
        if ((other.tag == "Player") || (other.tag == "Enemy") || (other.tag == "Explosion" ))//ajoutez dans les conditions l
        {
            Instantiate(expl, Trans.position, transform.rotation);
            Destroy(Trans.gameObject);
        }
    }
}
```

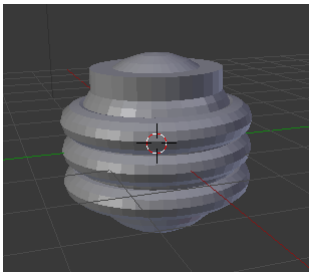
Ce script permet de créer un objet explosion dans la scène.

```
using UnityEngine;
using System.Collections;

public class Tonneau : MonoBehaviour
{
    public Transform Trans;
    public Explosion expl;
    public Rigidbody rig = new Rigidbody();
    private int health;
    void Start()
    {
        pos = rig.transform.position;

    }
    void OnTriggerEnter(Collider other)
    {
        if ((other.tag == "Player") || (other.tag == "Enemy"))//ajoutez dans les conditions le tag des projectiles
        {
            Instantiate(expl, Trans.position, transform.rotation);
            Destroy(Trans.gameObject);
        }
    }
}
```

Ce script contrôle l'objet tonneau. Si le tonneau est touché par un ennemi, un joueur ou même une explosion (pour l'instant), il explosera ; lançant le script explosion. Ceci permettra la création de pièges, voir même de champs de mines. Par ailleurs, afin de pouvoir expérimenter l'IA des ennemis ainsi que leur mouvements, nous avons implémenté des ennemis dans le jeu.

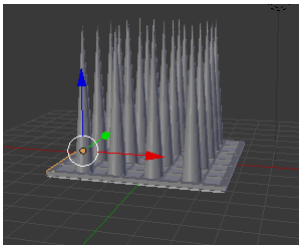
Le design des ennemis est sujet à amélioration mais il est grandement inspiré du film Steamboy et de ses chevaliers à vapeur et originalement tiré de l'asset store (référence en annexe).

Pour la deuxième soutenance, nous avons créés différents modèles de pièges. Nous avons tout d'abord fait le modèle de la bombe, anciennement tonneau explosif. Le script ayant déjà été conçu pour la précédente soutenance, et avec le modèle en main ; nous n'avons eu qu'à lier les deux pour que l'objet soit prêt à être implémenté dans n'importe quelle map, cet objet étant donc un objet destructible par le joueur ou les ennemis.



Nous avons aussi décidé de construire des portes (ou barricades), cet objet éviterait les rush de niveaux en forçant le joueur à être discret ou à tuer tous les ennemis avant de passer. Nous avons donc conçu un script pour faire des portes destructibles qui bloqueraient un passage, et associé le script à un modèle de porte conçu par nos soins à l'aide de Blender. Nous avons ainsi pu placer des portes bloquant l'accès sous les ponts sur la première map.

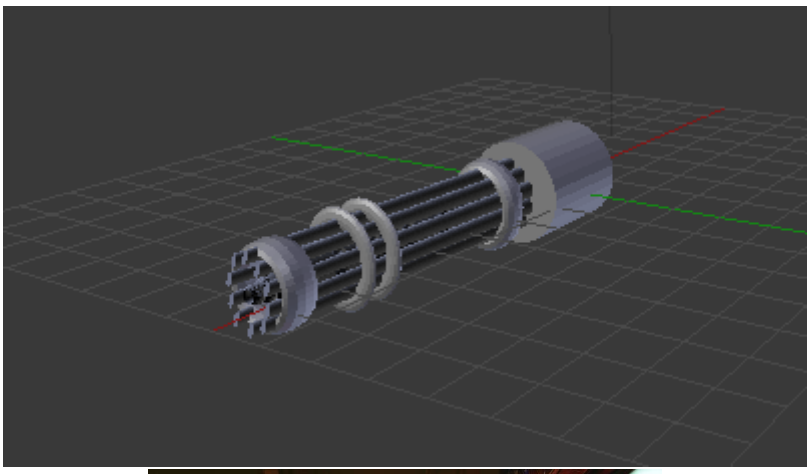
Nous avons également fait un script pour des pièges à piques ainsi que leur modèle (ci dessous).

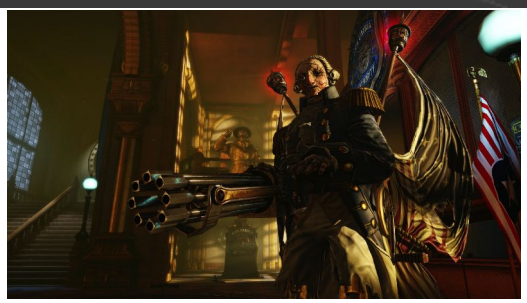


Ce piège est caché dans la structure, le principe étant qu'il met 3 secondes à s'activer ; si le joueur se fait toucher par les piques, il perdra des points de vie. Les piques obstrueront le passage et il sera nécessaire de les détruire pour avancer. Pour faciliter le déplacement et pour les repérer, les pointes des piques sortiront du sol et seront rouges. Les pièges feront des dégâts peu importe les ennemis.

Nous avons également avancé sur les modèles des ennemis. En effet, nous avons créé un petit ennemi lance-flammes. Nous nous sommes inspirés du lance-flammes que l'on trouve dans Bioshock Infinite. Nous avons déjà un script pour un ennemi lance-flammes. un comparatif ci-dessous :

Nous avons également fait un modèle de gatling pour le prochain ennemi qui sera inspiré lui aussi d'un autre ennemi de Bioshock Infinite.





Pour cette nouvelle soutenance nous avons créé une nouvelle map, plus petite. C'est une sorte de place où les ennemis apparaîtront et commenceront à attaquer les joueurs. Ceux-ci arriveront en vague puis au bout d'un certain nombre d'ennemis tués ; le boss final apparaîtra. Pour une telle map, il n'y aura pas de pièges outre des tonneaux explosifs que les joueurs pourront utiliser afin de blesser voire tuer les ennemis. Nous avons désactivé le brouillard, car il semblerait que celui-ci influe grandement sur les performances, étant en partie une raison des longs chargements de la map. Le mini-boss gatling est, on peut le dire, un "tank" ; il a beaucoup de points de vie mais ne fait pas beaucoup de dégâts. En effet, il fait peu de dommages, mais tire en continu, l'éviter devient donc nécessaire pour survivre.

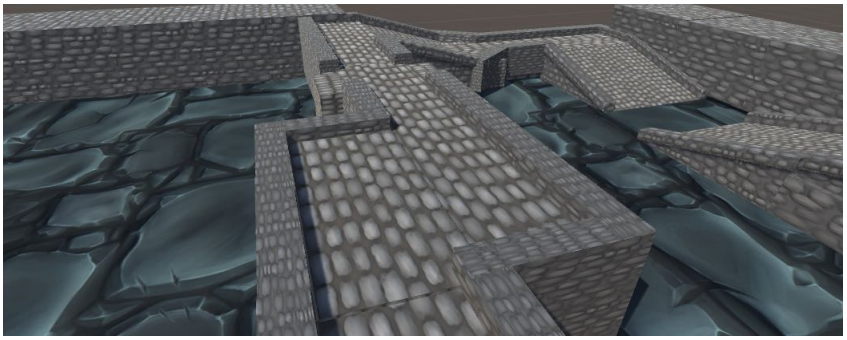
Par ailleurs, maintenant les classes ont chacunes un modèle : le guerrier a un modèle de chevalier, le mage et le healer sont des grands personnages avec un grand chapeau équipés ou non d'un bâton. L'assassin possède maintenant deux dagues.

7.1.4 Design du niveau

Le design de ce premier niveau se rapproche plus d'un niveau de type couloir avec plusieurs chemins possibles, des ennemis humanoïdes, une furtivité possible et une fin de niveau (avec un boss). En effet ; avec Edouard nous avons réfléchi à une conception permettant une diversité des chemins, qui aboutira à : des objets collectibles cachés sur des chemins annexes, des bonus, ou qui vont simplement permettre de combattre moins d'ennemis. C'est un niveau dit de test donc il sera sujet à améliorations, ce niveau étant un des plus durs à faire notamment par rapport à la map de versus ou la map de vague d'ennemis.

Deuxième :

Le premier niveau était un niveau uniquement horizontal avec de nombreux passages et obstacles, qui était donc assez adapté à la furtivité. Cependant, le deuxième niveau est orienté très différemment ; en effet ce niveau est plus vertical, avec des ponts, mais aussi des pièges, et peu de chemins possibles sinon deux chemins. L'un des chemins est rempli d'ennemis sans couverture, et l'autre chemin est rempli de pièges sans couverture. C'est un niveau qui est donc moins favorable a la furtivité, mais reste assez libre de choix. De par l'absence de protection, ce niveau favorise les attaques a distance, et les ponts étant assez étroits, les attaques de zones (mage) peuvent être dévastatrices.



Cependant, la partie des pièges de part le nombre d'ennemis limités ainsi que de l'impossibilité de se précipiter ; une approche tactique est conseillée. Ce niveau s'annonce donc plus guerrier que le précédent tout en offrant une possibilité plus risquée et demandant de la tactique. Là où le premier niveau encourageait l'évitement des ennemis et l'exploration par sa pluralité de chemins, ce deuxième niveau s'oriente plus sur le combat et le risque que le premier. Il reste maintenant à faire un niveau de vague, plus carré que les précédents.

Nous avons également une map spécialement faite pour le Joueur contre Joueur, où les pièges seront présents pour obliger les joueurs à ne pas simplement se focaliser sur les autres joueurs mais également sur l'environnement.

La troisième map prendra place en haut d'une tour et sera donc beaucoup plus centrée autour des joueurs qu'avant. Maintenant, ce seront les ennemis qui viendront au joueur et non l'inverse. Cette map représentera un défi pour les joueurs.

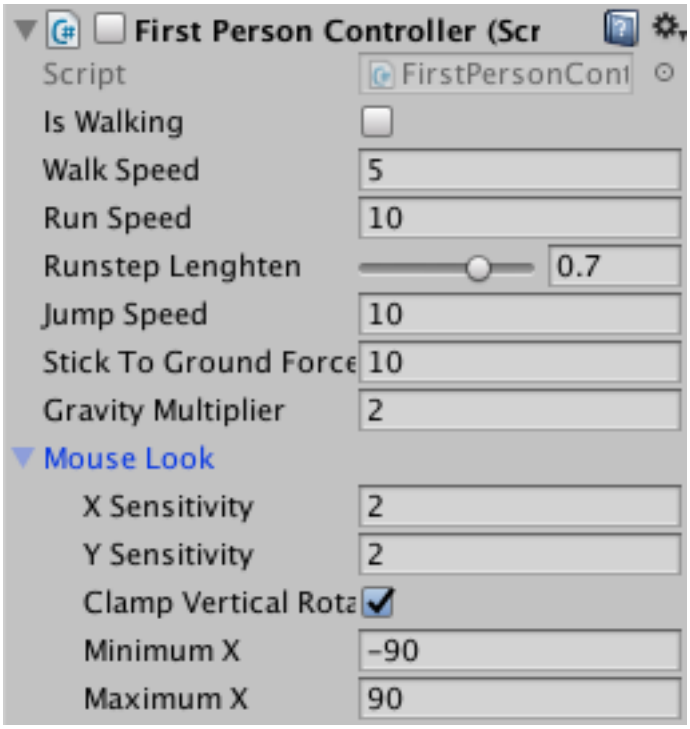
7.2 Contrôles, Animations et Sons

7.2.1 Contrôles Basiques

Pour la première soutenance, nous avons d'abord commencé par implémenter des contrôles basiques de déplacement ; c'est à dire le déplacement grâce au touches ZQSD. Puis après, en regardant un peu la documentation d'Unity, nous avons vu qu'il était possible de récupérer des axes ; c'est à dire l'axe horizontal et vertical par rapport aux touches définies dans les inputs d'Unity.

Grâce à cela nous avons pu implémenter des déplacement au clavier ; mais aussi avec une manette. Nous avons par la suite implémenté les déplacements de la caméra par rapport à la souris, puis grâce à la documentation nous avons pu implémenter des déplacement de caméra avec souris et manette.
Nous avons par la suite implémenté le saut ainsi que la possibilité de courir que ce soit en utilisant un clavier ou une manette.

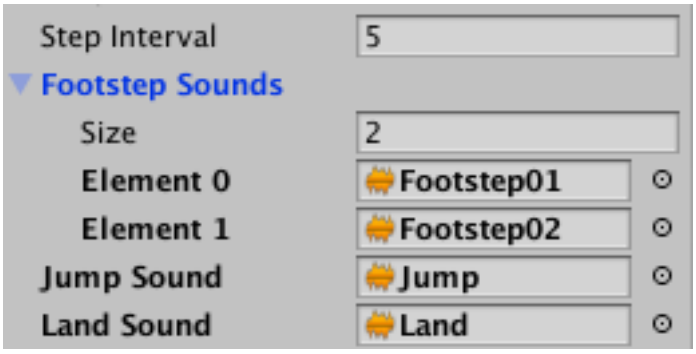
Nous avons donc par la suite implémenté les contrôles en réseau ce qui fait que les mouvements des différents joueurs sont bien effectués sans problèmes, mais aussi que chaque joueur verra les autres joueurs bouger. Ce qui fait que les contrôles basiques étaient donc complétés lors de la première soutenance.



7.2.2 Animations Et Sons des personnages

Lors de la première soutenance, les ennemis, mais aussi le joueur possède des animations lors de leurs déplacement. Le joueur possède de plus une animation lors de son saut. Ces différentes animations sont de plus implémentés en réseaux.

De plus les joueurs effectuent des sons lors de leurs différents mouvements. C'est à dire lors de leurs déplacement mais aussi lors des sauts.
Ces sons sont joués pendant leurs différentes animations et sont aussi partagés en réseaux. C'est à dire que les différents joueurs entendent les sons faits par les autres.



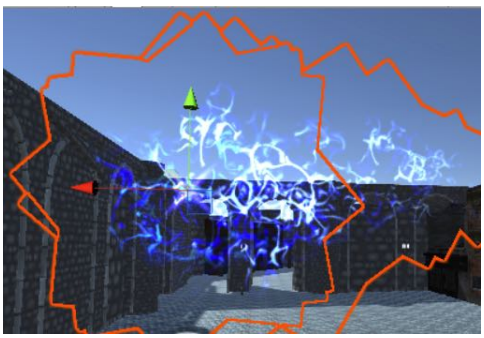
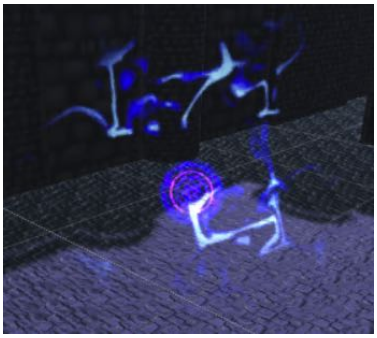
Les ennemis et et les joueur possédait déjà des animations lors de leurs déplacement, lors de la première soutenance. Ils possèdent désormais des animations lors de leurs morts. Ces différentes animations sont de plus implémentés en réseaux. Les joueurs ont aussi des animations spécifiques à leur classe maintenant. Qui se déclenchent en fonction de certains sorts. Nous avons aussi une bande son pour la première map qui est une musique libre de droit. Nous avons choisi la musique "Mr Gear" qui est une musique dans le style Steampunk et correspond donc parfaitement à l'ambiance notre jeu.

7.2.3 Les Sorts

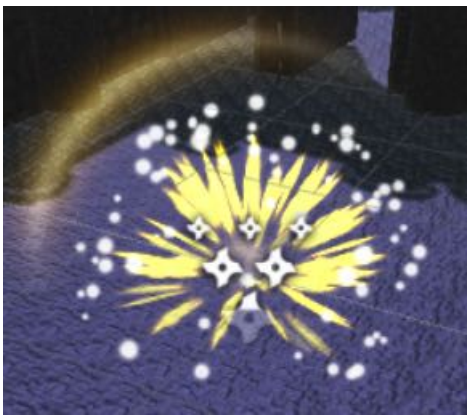
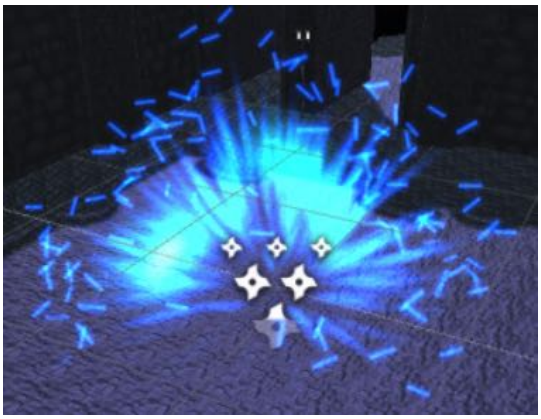
Le tir allié étant activé, nous avons décidé d'implémenter de l'armure, en plus de la Vie et du Mana qui permet de lancer des sorts, qui permet de d'enlever un pourcentage des dégâts subis à cause d'un allié. En effet chaque personnage a des points d'armure (PA) propres qui lui sont attribués : le Mage a 20PA, le Guérisseur a 10PA, le Guerrier en a 50 et enfin l'Assassin a 30PA. Lorsqu'un joueur est pris dans l'effet d'un sort d'un autre joueur, les dégâts subis sont réduis d'un pourcentage équivalent aux points d'armure du joueur. Donc lorsqu'un sort inflige des dégâts à des ennemis, il faut garder à l'esprit que si un joueur est touché il subira des dégâts en fonction de sa classe.
Les joueurs ont aussi des animations spécifiques à leur classe maintenant. Qui se déclenchent en fonction de certains sorts. Les ennemis possèdent aussi des animations dites de combat qui permettent d'identifier leurs attaques.
La création des sorts a pris plus de temps que prévu. En effet, les sorts étant essentiellement des particules, nous avons préféré pour chaque soutenance nous concentrer sur leur animation avant d'implémenter les scripts de leurs effets sur les ennemis ou les alliés (le tir allié sera activé). La création d'un sort commence par la création d'un système de particules.

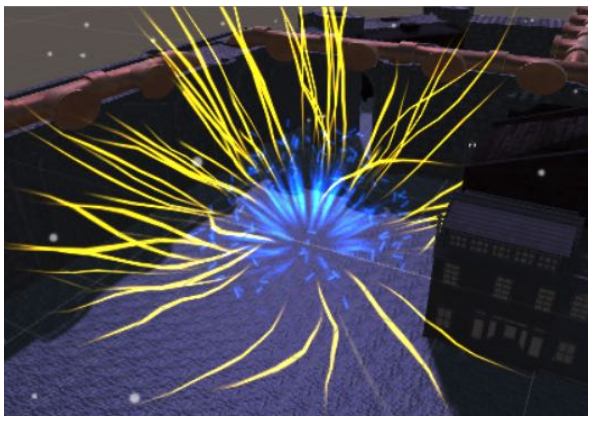
7.2.4 Animation Et Effets Des Sorts

Le Mage : Le premier sort est une boule d'énergie électrique, elle est composée d'un noyau d'énergie statique, d'un réseau de filaments électriques dynamiques et d'une lumière bleutée. Cette boule d'énergie va donc se déplacer et créer un champ électrique autour d'une unité ennemie à son contact qui va ralentir l'ennemi pour une durée de 3 secondes. A l'impact, l'ennemi touché recevra 10 points de dégâts et tous les ennemis dans la zone du champs électrique recevront 2,5 points de dégâts. Le premier sort du Mage est son sort de base, il est donc presque gratuit et peut se lancer aussi souvent que le jouer le veut, il est assez intéressant car il permet un petit contrôle de foule. Voici à quoi le premier sort ressemble :

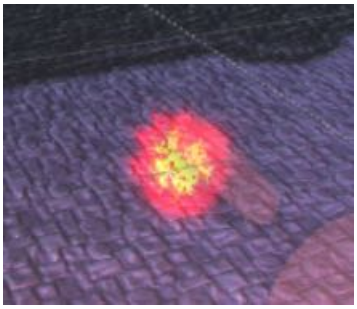



Le deuxième sort du Mage s'articule de la manière suivante : le joueur se téléporte dans la direction pointée par la souris sur une distance donnée et limitée, suite à ce déplacement, une explosion magique se produit qui infligera 20 points de dégâts aux ennemis proches et la moitié aux alliés proches et demande 20 points de Mana. Cette explosion est en réalité composée de deux explosions (graphiquement uniquement) qui constituent la principale animation du sort, voici à quoi ressemble les deux différentes explosions et le résultat final :

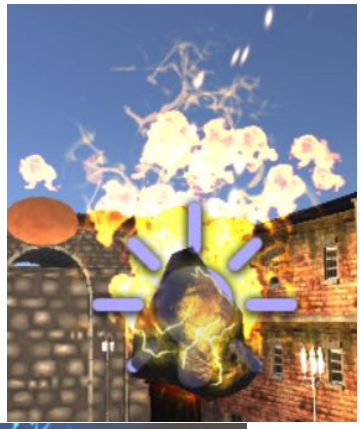


Parlons maintenant des deux autres sorts du Mage, ce sont les deux sorts les plus puissants et les plus impressionnants du jeu. Le troisième sort a été assez simple à créer mais sa création a pris beaucoup de temps car il est composé de trois différents GameObject, là où le premier et le deuxième sort n'en sont composées que de deux de deux (et le dernier de cinq mais nous y reviendrons). Ce troisième sort consiste en l'envoi d'une boule d'énergie chaude et lente en ligne droite (qui se déplace grâce à un script appelé Mover) et si elle touche un ennemi ou un joueur, il est instantanément mort et une colonne de flammes se crée dans une explosion qui dureront pas moins de dix secondes à l'emplacement où la boule a touché le malheureux élu, la colonne de feu infligeant 5 points de dégâts par seconde à tous les ennemis et alliés présents dans celle-ci et l'explosion inflige 2 points de dégât par seconde à tous ceux présents dans la zone de l'explosion. Ce sort coûte 60 points de Mana. Voici quelques visuels du sort, la boule d'énergie, l'explosion et la colonne de feu :

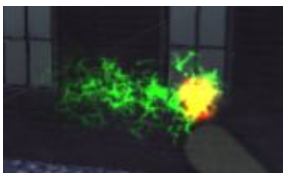


Et enfin, le dernier sort du Mage, le plus puissant, le plus impressionnant et le plus dur à réaliser : l'Astéroïde. En effet, l'ultime sort du Mage et composé de cinq GameObject, nous avons dû également créer un script Mover rien que pour le lancement de ce sort que nous avons appelé MeteorMover. C'est le sort qui nous as demandé le plus de temps à créer à ce jour car nous avons éprouvé quelques difficultés à le faire fonctionner. Le sort s'articule en trois étapes, la Canalisation, la chute de l'astéroïde et enfin l'Impact au sol à la suite de la chute. Lors de la Canalisation, le Mage et entouré d'une aura magique, il utilise tout son pouvoir pour créer l'astéroïde en altitude. En réalité, au début de la Canalisation, un GameObject, appelé MeteorLauncher, invisible dans le jeu est lancé à très grande vitesse pour que le sort puisse se lancer, que le joueur ne rate pas sa cible

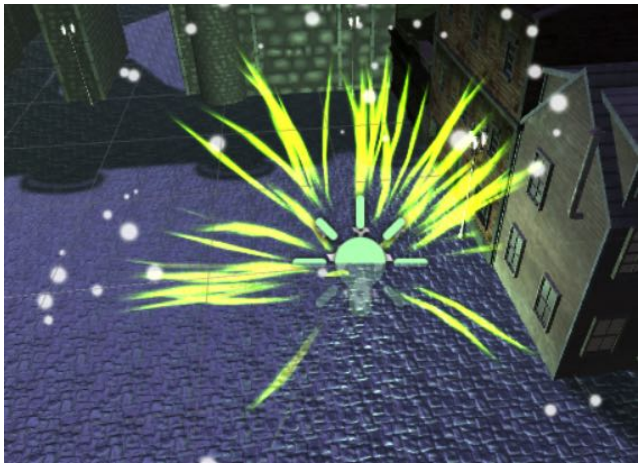
car les conséquences sur le joueur sont dramatiques lorsqu'il utilise sont ultime sort : le joueur doit utiliser 100 points de Mana et 50 points de vie pour utiliser le sort, durant la Canalisation il est immobilisé et ne peux plus lancer de sorts sachant que celle-ci dure 10 secondes. Donc le MeteorLauncher touche la cible et l'astéroïde est créé en altitude. Nous allons faire ici un bref point pour expliquer pourquoi nous avons créé un MeteorMover car nous venons de parler de création d'objet en altitude et non à l'emplacement de la cible. En effet, le Mover initial a été légèrement modifié depuis la dernière fois, au contact d'un ennemi ou d'un joueur, le Mover fait apparaître l'effet voulu à l'endroit où le contact à été établi, or pour l'astéroïde, on veut qu'il apparaisse au dessus de la cible, donc pour ne pas modifier le Mover qui est commun à plusieurs sorts, il faut en créer un autre spécifique à ce sort. L'astéroïde tombe, il traverse le sol (problème rencontré : la vitesse de l'astéroïde est trop grande pour que le collision soit détectée, il fallait donc le ramener au niveau du sol lorsqu'il le dépasse) et une explosion d'énergie se crée, tout joueur ou ennemi dans la zone au centre de l'impact voit ses points de vie descendre à 1 point et dure 12 secondes, l'astéroïde reste sur le sol pendant toute la durée de la partie et tout personnage le touchant meurt(nouveau problème : avec le système habituel de détection de collision, l'explosion était lancée en boucle, nous avons donc créé un nouvel astéroïde qui ne déclenche pas d'explosion, nous avons supprimé le collider du premier qui ne sert plus à rien vu qu'il déclenche l'explosion, crée le nouvel astéroïde sur le sol et est supprimé dès qu'il est plus bas que le sol). Il est temps maintenant de donner quelques visuels de la canalisation, de l'astéroïde et de l'explosion d'énergie :

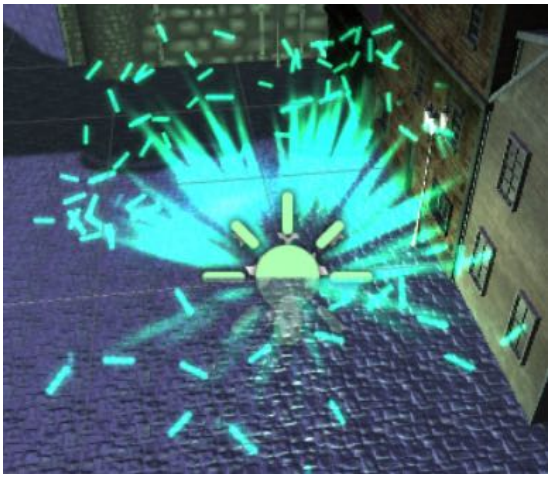


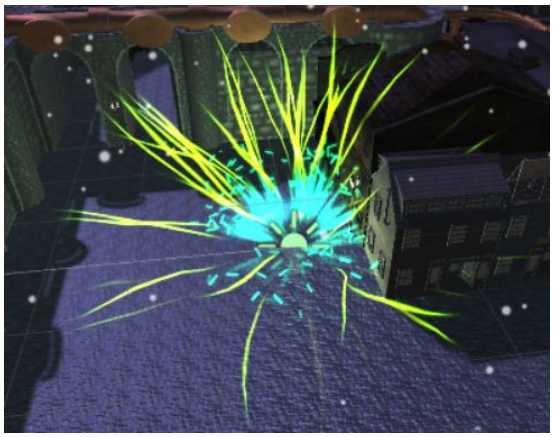
Le Guérisseur Le premier sort du Guérisseur est le même que celui du Mage sauf que les dégâts infligés aux ennemis sont de 5 au lieu de 10 et rendent la moitié de ces dégâts en points de vie au joueur s'il est touché au lieu de lui faire perde la moitié de ces dégâts. Ce sort coûte 5 points de Mana autant pour le Mage que pour le Guérisseur. Voici à quoi il ressemble :



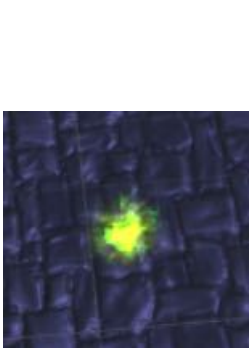
Le deuxième sort du Guérisseur est le même que celui du Mage pour ce qui est de l'animation, pour ce qui est des effets, ils sont totalement différents sur les alliés : l'explosion infligera 10 points de dégât aux ennemis proches et redonnera en points de vie la moitié aux alliés proches et nécessitera 20 points de Mana pour être lancé. Cette explosion est en réalité composée de deux explosions (graphiquement uniquement), voici à quoi ressemble les deux différentes explosions et le résultat final :

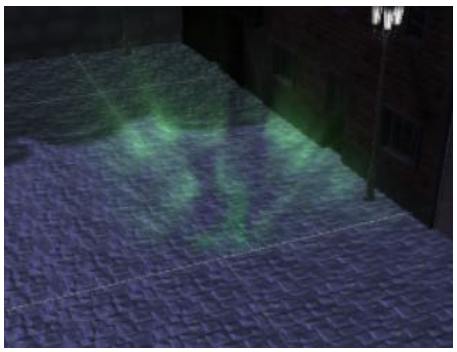






Le troisième sort du Guérisseur consiste en l'envoi d'une boule d'énergie vitale et lente en ligne droite et si elle touche un ennemi ou un joueur, l'ennemi touché perd 50 points de vie et le joueur touché regagne 25 point de vie. Ce sort nécessitera 60 points de Mana pour être lancé. Voici quelques visuels du sort, la boule d'énergie et l'explosion :



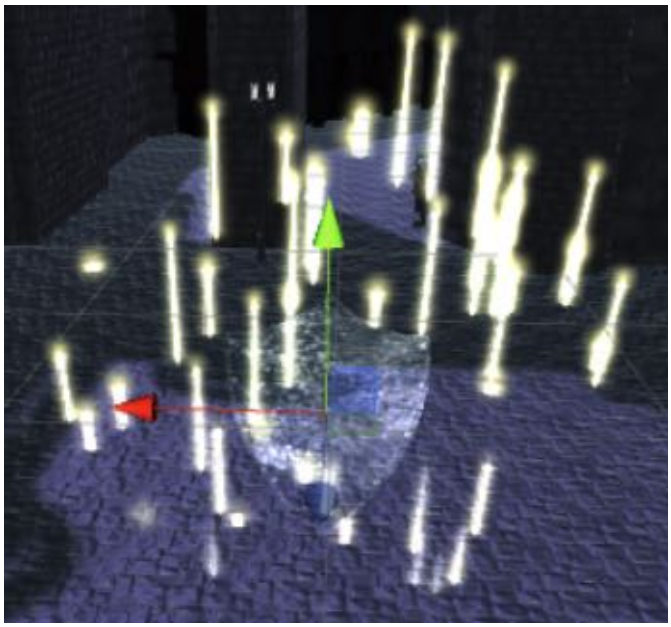


Et enfin, le dernier sort du Guérisseur, le plus utile : la zone de soin. En effet, l'ultime sort du Guérisseur va rendre 5 points de vie par seconde à tous les joueurs présents dans la zone de soin et inflige 2.5 points de dégât aux ennemis présents dans la zone, le sort nécessite 100 points de Mana et le joueur ne peut plus lancer de sorts pendant la durée de celui-ci :

Le Guerrier Le Guerrier n'utilise pas les éléments de la magie comme le feu, l'électricité et la vie mais il utilise la matérialisation d'objets offensifs comme une épée ou défensif comme un bouclier. Le Guerrier a autant de sorts offensifs que défensifs : le premier et le dernier sort sont offensifs, le deuxième et le troisième sort sont défensifs. Ce personnage est également basé sur le contrôle de foule ce qui fait de lui un des piliers de l'équipe.

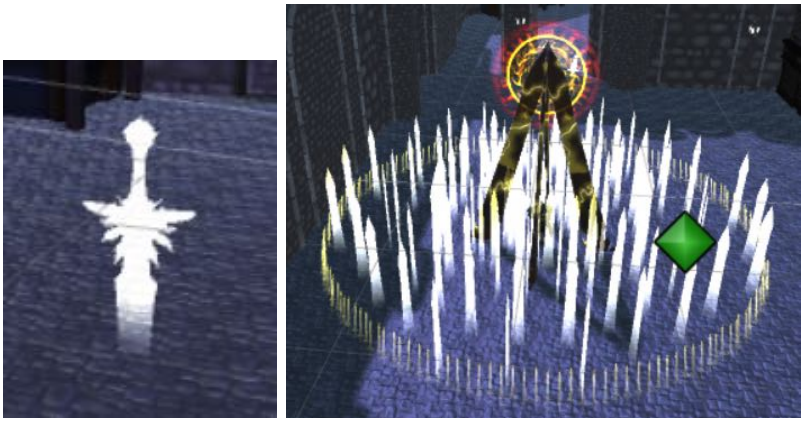
Le premier sort du Guerrier est totalement différent de celui du Mage et du Guérisseur de par la nature du personnage. En effet, le joueur fait apparaître une épée lumineuse qui se déplace en ligne droite, traverse les ennemis et leur inflige 30 points de dégâts et se plante sur un mur ou dans le sol. Lorsqu'un ennemi touche une épée sur un décor, il reperd 30 points de vie mais lorsqu'un allié touche une épée, il perd des points de vie en fonction de sa classe et l'épée disparaît. Comme le Mage et le Guérisseur, le premier sort est presque gratuit. Voici à quoi ressemble l'épée :

Le deuxième sort du Guerrier consiste en une augmentation progressive de la vitesse de déplacement du personnage. Le Guerrier effectue donc une grande charge en faisant apparaître un bouclier translucide devant lui qui immobilisera tous les ennemis touchés et les immobilisera pour la durée du sort tandis que les alliés recevront un boost d'armure et des dégâts doublement réduits dû au boost d'armure (armure multipliée par 2) lui aussi pour la durée du sort. Ce sort coûtera 20 points de Mana.



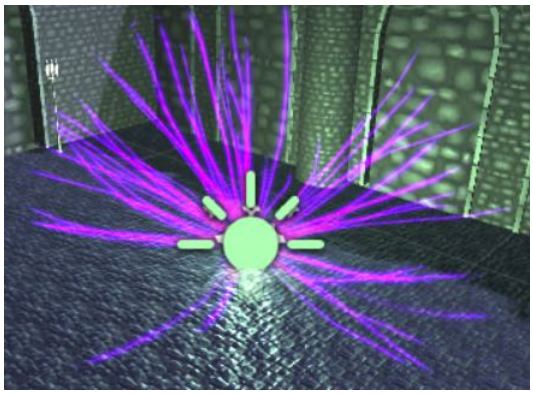
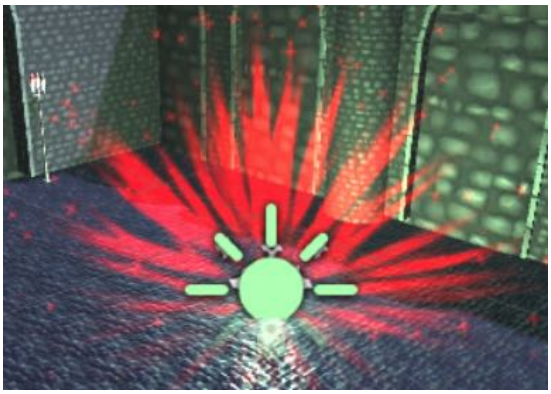
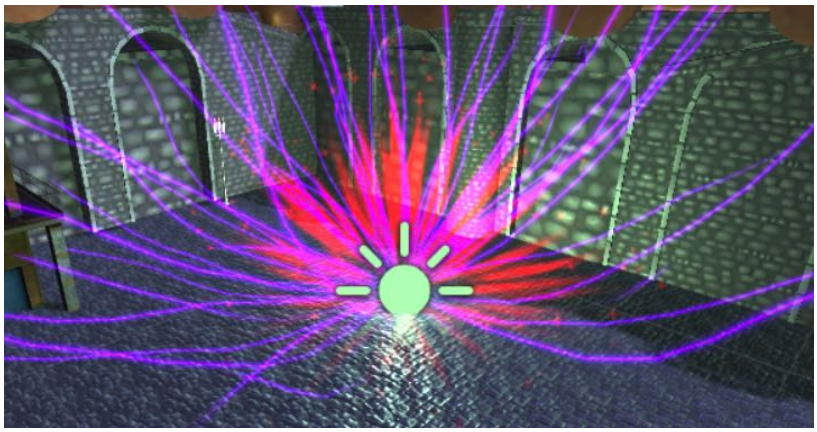
Le troisième sort est un mur de boucliers, sont seul effet est d'immobiliser pour le reste de la partie quiconque est touché par ce mur, que ce soit un joueur ou un ennemi, jusqu'à ce qu'un autre sort rétablisse le déplacement de base des ennemis/joueurs immobilisés. Ce qui implique que si un joueur est touché par le sort de base du Mage, par exemple, il pourra bouger correctement en échange de quelques points de vie. En effet, tous les sorts qui ont pour effet de modifier la vitesse de déplacement d'un ennemi/joueur doivent la rétablir à sa valeur de base à la fin de la durée de l'effet de ce sort. Le coût en Mana de ce sort est de 60 points de Mana.

Le dernier sort du Guerrier est le plus puissant de ses sorts et se déroule de la manière suivante : le Guerrier plante une épée dans le sol sur sa position, seul le Guerrier peut la toucher, si un ennemi ou un joueur la touche, une multitude d'épées plus grandes que les joueurs sortent du sol dans une créant une explosion. Les ennemis et les joueurs pris dans le champ d'épées recevront 2,5 points de dégâts lorsqu'ils toucheront une épée et seront coincés entre les épées. Le dernier sort du Guerrier permet donc un énorme contrôle de foule et inflige de nombreux dégâts à tous ceux pris dans la zone. La particularité de ce sort est qu'il s'active comme un piège ce qui permet également un bon moyen de fuite au cas où les choses dégénèrent.



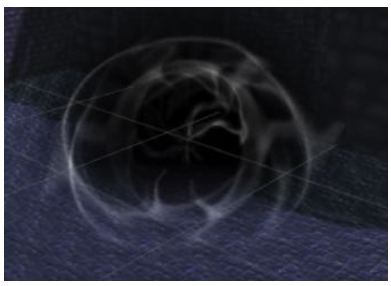
L'Assassin L'assassin est le personnage le plus fort du jeu à partir du moment où il reste non détecté par les ennemis. En effet, lorsqu'il est détecté, tous les dégâts qu'il inflige sont divisés par deux mais il a évidemment un moyen de redevenir furtif lorsqu'il est détecté. Les dégâts infligés aux autres joueurs sont sur la base des dégâts furtifs.
Le premier sort de l'Assassin lance un couteau comme le Guerrier lance une épée mais le couteau infligera 40 points de dégâts lorsque le joueur sera furtif et 20 lorsqu'il sera détecté, le couteau ne se plantera pas dans le décors et disparaîtra au contact d'un ennemi/joueur. Étant le sort de base de l'Assassin il est aussi presque gratuit.

Le deuxième sort de l'Assassin est celui qui lui permet de redevenir furtif. Comme les trois autres personnages jouables, le deuxième sort influe sur son déplacement. En effet, au lieu de se téléporter en avant comme le font le Mage ou le Guérisseur, il se téléporte en hauteur et l'explosion a lieu à l'atterrissage du joueur sur le sol ou sur un bâtiment. Le seul personnage pouvant aller sur un bâtiment est l'Assassin ce qui lui permet de sortir du champ de vision des ennemis et de faire un maximum de dégâts. Le sort coûte 20 points de Mana et inflige 40 points de dégâts à tous les ennemis alentours (divisé par deux si découvert) et aux alliés.

Le troisième sort de l'Assassin consiste en la propagation d'une "maladie" à tous les ennemis/joueurs qui toucheront l'ennemi ou le joueur infecté en premier, infligeant 1 point de dégât par seconde à l'infecté. Les infectés seront signalés par une marque rouge sur le corps. Le coût sera la perte de la furtivité du personnage et 60 points de Mana.

Le dernier sort du personnage est une explosion ténébreuse qui inflige 75 points de dégât instantanément à tous les ennemis dans la zone de l'explosion (divisé par deux si découvert) et à tous les joueurs et donc à lui même. La perte de points de vie de l'Assassin fait partie du coût du sort ainsi que les 100points de mana nécessaires à son activation.



7.3 IA, Réseau et Menus

Pour notre jeu, nous avons voulu faire des ennemis intelligents qui peuvent repérer le joueur, nous avons aussi implémenté un multijoueur et une interface utilisateur le permettant, un site web pour le promouvoir.

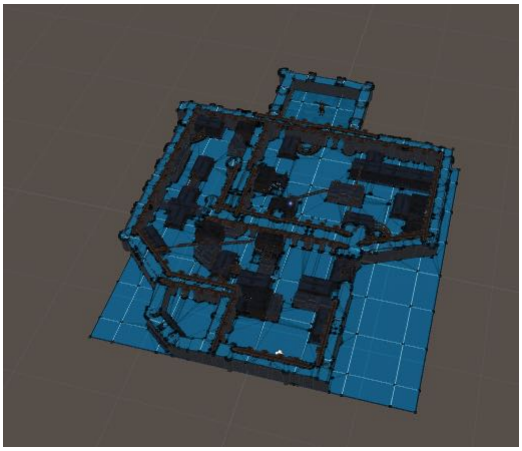
7.3.1 L'IA des Ennemis

Nous allons maintenant vous parler de l'intelligence artificielle des ennemis et de leur mouvements. Originellement, nous voulions que les ennemis puissent repérer un des joueurs selon certains critères et puisse le poursuivre, et perdre sa trace s'il arrive à s'échapper. Pour ce qui est de la détection, nous avions pour but d'en faire une assez élaborée car notre jeu se devait de pouvoir permettre un certain degré de furtivité de la part des joueurs. Pour la poursuite, nous nous sommes beaucoup appuyés sur l'outil Unity de navigation.

Pour la première soutenance, afin de pouvoir expérimenter l'IA des ennemis ainsi que leur mouvement nous avons implémenté des ennemis dans le jeu.

Le design des ennemis est sujet à amélioration mais il est grandement inspiré du film steamboy et de ses chevaliers à vapeur.

Pour ce qui est du comportement des ennemis, les ennemis détectent le joueur si il est à une certaine distance d'eux et dans leur angle de vision (de 120 degrés pour l'instant), et qu'il ne sont pas cachés par un objet. pour ce faire, nous avons construit pour la map un champ de navigation (navmesh).



Le navmesh permet de définir leur zone parcourable, et nous leur avons assigné un navmesh agent qui est basiquement l'IA de Unity. La condition de détection est la suivante :

```
NavMeshHit hit;
Vector3 targetDir = Player.position - transform.position;
float angle = Vector3.Angle(targetDir, transform.forward);
if ((Player.position - transform.position).magnitude < 2f && !nav.Raycast(Player.position, out hit) && angle < 60f)
{
    nav.isStopped = false;
    nav.SetDestination(Player.position);
    anim.SetBool("player in sight", true);
}
```

Ainsi dès que le joueur sera dans le champ de vision de l'ennemi, l'ennemi se dirigera donc vers le joueur, lorsque que le joueur lui aura échappé, l'ennemi reviendras à sa position (et rotation) initiale. Tous les déplacements de l'ennemi sont animés. L'IA ainsi faite se rapproche de l'IA des ennemis de couloir, (humanoïdes) L'IA des ennemis de vague étant une version très simplifiée de celle ci, ainsi sa conception ne seras qu'une formalité.

Pour la première soutenance, j'avais mis en place la détection basique de l'ennemi (champ de vision), et la base de l'outil de navigation de unity pour les ennemis (navmesh), ainsi qu'un chemin de retour dès que l'ennemi n'est plus en vue. Pour cette soutenance, j'ai considérablement amélioré la détection de l'ennemi. En effet, maintenant l'ennemi peut détecter plusieurs joueurs, ce qui n'était pas pris en compte avant, et ainsi, adapte l'IA au multijoueur. Les joueurs étant tous tagués avec le tag "Player" je peux utiliser la commande "GetGameObjectsWithTag("Player") pour détecter tous les joueurs sur la scène et renvoyer un tableau de tous ces joueurs (un maximum de 4 joueurs).

```
NavMeshHit hit;

if (!isFocused)
{
    foreach (GameObject player in Players)
    {
        if (player.activeSelf)
        {
            if (Vector3.Angle(player.transform.position - transform.position, transform.forward) < _fov && (player.transform.position - transform.position).magnitude < 3f
                && !nav.Raycast(player.transform.position, out hit) || player.GetComponent<AudioSource>().minDistance > (player.transform.position - transform.position).magnitude)
            {
                Player = player.transform;
                audio = player.GetComponent<AudioSource>();
                isFocused = true;
                break;
            }
            if ((player.transform.position - transform.position).magnitude < (Player.position - transform.position).magnitude)
            {
                Player = player.transform;
                audio = player.GetComponent<AudioSource>();
            }

        }
    }

}
```

Par exemple ce bout de code définit que si l'ennemi ne détecte pas de joueur ou qu'il ne détecte plus sa précédente cible, il cherchera un nouveau joueur, si un autre joueur est détecté, il changera de cible. Sinon, il va surveiller la cible la plus proche. Par ailleurs, la détection d'un personnage isolé a elle aussi été améliorée. Maintenant, l'ennemi peut détecter un joueur au son. si l'ennemi entend le son du joueur faiblement, il deviendra suspicieux (augmentation de l'angle de détection, changement d'animation), et si il détecte fortement le son du joueur (sprint, trop proche) le joueur sera détecté, même hors du champ de vision.

```
if (hint || ((targetDir).magnitude < 3f && !nav.Raycast(Player.position, out hit) && angle < _fov || audio.minDistance > targetDir.magnitude))
{
    hint = false;
    isFocused = true;
    _time = 0f;
    nav.isStopped = false;
    anim.SetBool("walking",false);
    anim.SetBool("suspicious", false);
    anim.SetBool("detected", true);
    nav.SetDestination(Player.position);
}
```

Ici on a hint (dont nous parlerons plus tard), la détection en fonction du champ de vision, et la détection par le son. J'ai rendu la détection du son possible en changeant directement les valeurs de distance minimale et maximale de la source audio diffusant les bruits de pas dans le script qui contrôler le personnage, ainsi, la détection sera directement liée au son que nous entendons (pas de détection entre deux pas). Par ailleurs, l'ennemi si il ne détecte plus le joueur, ne va plus retourner immédiatement à sa place d'origine, en effet il suffisait de se placer dans son angle mort pour qu'il retourne a sa place. maintenant, si le joueur est a une très courte distance de l'ennemi et qu'il n'est plus détecté, l'ennemi ne perdra plus sa trace, et si le joueur est a distance de l'ennemi et qu'il n'est plus détecté (par exemple après avoir tourné au coin d'une rue) l'ennemi va aller au dernier endroit où le jouer a été détecté et regarder autour de lui quelques instants avant de revenir à son point de départ, comme le montre le bout de code ci-dessous.

```
if ((nav.destination - transform.position).magnitude <= 0.2f)
{
    anim.SetBool("detected", false);
    if (anim.GetBool("walking"))
    {
        if (_time > 3f)
        {
            anim.SetBool("suspicious", false);
            _fov = 60f;
            _time = 0f;
        }
        anim.SetBool("walking",false);
        nav.isStopped = true;
        transform.rotation = initialDir;
    }
    else
    {
        anim.SetBool("suspicious",true);
        if (_time > 3f)
        {
            anim.SetBool("suspicious", false);
            anim.SetBool("walking", true);
            nav.SetDestination(initialPos);
            _time = 0f;
        }
    }
    if (anim.GetBool("suspicious"))
    {
        _time += Time.deltaTime;
    }
}
```

Reparlons maintenant de la variable "Hint" vue précédemment, elle permet a l'ennemi, si il se fait attaquer de loin, de le faire se déplacer vers la source du tir afin de vérifier la présence ou non d'un joueur. l'activation de Hint est gérée directement par le script du sort avec la collision et en détectant le tag "ennemy". Ces améliorations de la détection permettent un

meilleur traitement de la furtivité, ainsi qu'une augmentation de la difficulté du jeu, en effet, si le joueur court, il se fera détecter d'assez loin. ces améliorations vont de pair avec la partie furtivité que je dirige aussi, qui sera nécessaire pour un personnage furtif, ou même pour finir des niveaux plus vite ou sans se battre. Par ailleurs, avec l'implantation des dégâts, les ennemis ont maintenant la possibilité de mourir (mort animée).

Nous avons maintenant implémenté les dégâts des ennemis, en développant leur modèles, en leur donnant des armes, pour qu'ils puissent potentiellement tuer un jouer et s'inscrire dans l'univers du jeu (steampunk).

L'IA des ennemis dépend par ailleurs du type d'arme utilisé, les ennemis à distance vont pouvoir se tenir loin des joueurs tout en attaquant.

7.3.2 Le Site Internet

Pour ce projet nous avons crée un site web qui présente de manière générale notre jeu et qui propose différent liens qui permettent de télécharger tout ce qui tourne autour de notre projet, comme le cahier des charges, les différents rapports ou le jeu en lui même !

Ce site est accessible à l'adresse suivante : **https ://wes974.github.io/SSAO**

Grâce à un module de GitHub, GitHub Pages, qui permet de créer des sites simplement, hébergé par GitHub et directement dans le dossier de notre projet, nous avons créé un site en utilisant un fichier Markdown dans lequel nous mis ce qu'on souhaitait avoir dans notre site.

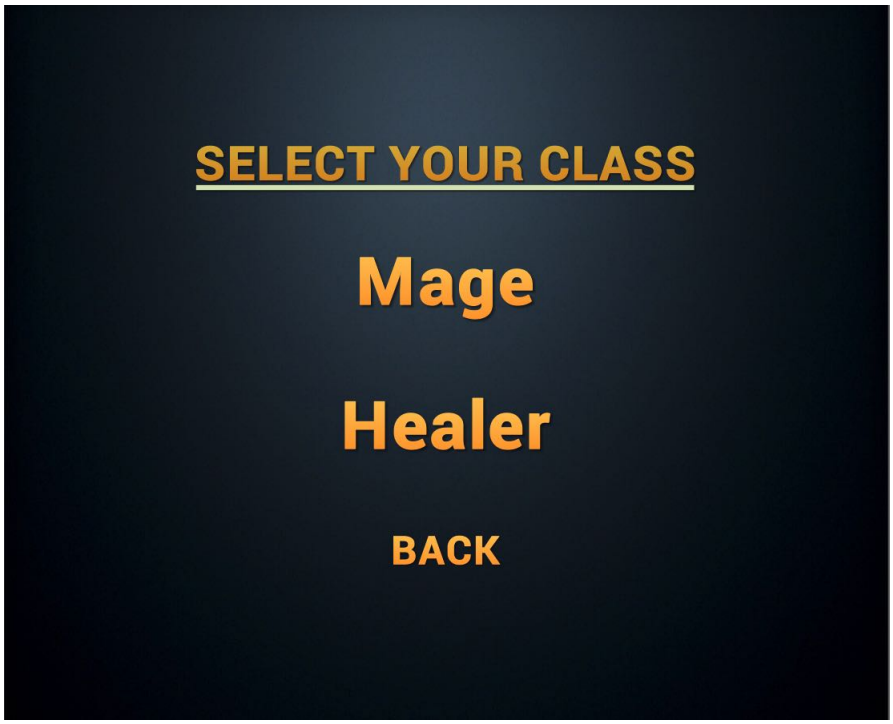
Ce site contient les liens vers notre cahier des charges ainsi que notre rapport de projet, il n'était pour la première soutenance uniquement constitué que d'une seule page.

Le site, toujours créé avec l'outil de GitHub, a été mis à jour pour la deuxième soutenance et propose désormais plusieurs pages.

Ainsi qu'une interface un peu changée. Pour l'instant le site ne contient qu'une description des membres et des liens vers GitHub, un lien de téléchargement de notre rapport. Le multi-page permettras à terme de télécharger le jeu, de d'avoir une interface graphique attirante, et décrira notre jeu. Le site est toujours en développement mais va à terme permettre de télécharger le jeu, et l'interface plus accueillante.

7.3.3 Menus

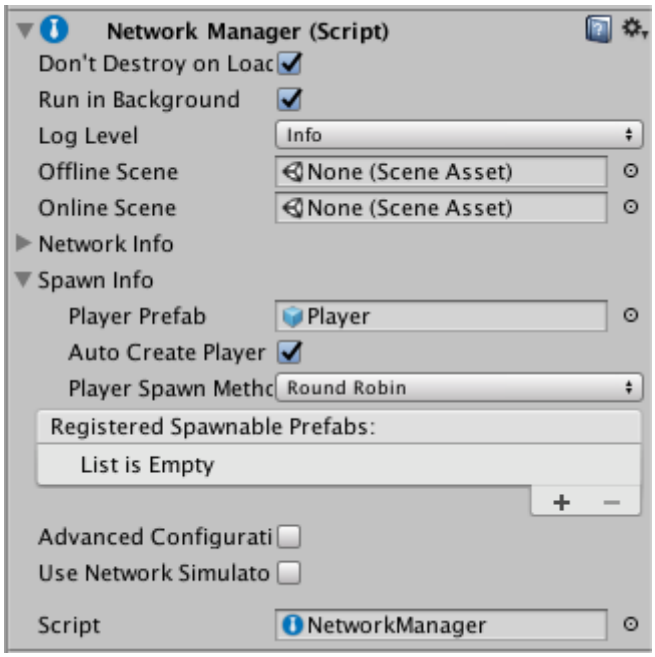
Nous avons aussi implémenté un menu. Il possède 4 options, un pour jouer en solo qui redirige vers la sélection de la classe, une autre option pour pouvoir lancer la partie multijoueur, une troisième option qui redirige vers les options et permet de régler le volume pour l'instant, ainsi qu'une dernière option pour quitter le jeu.



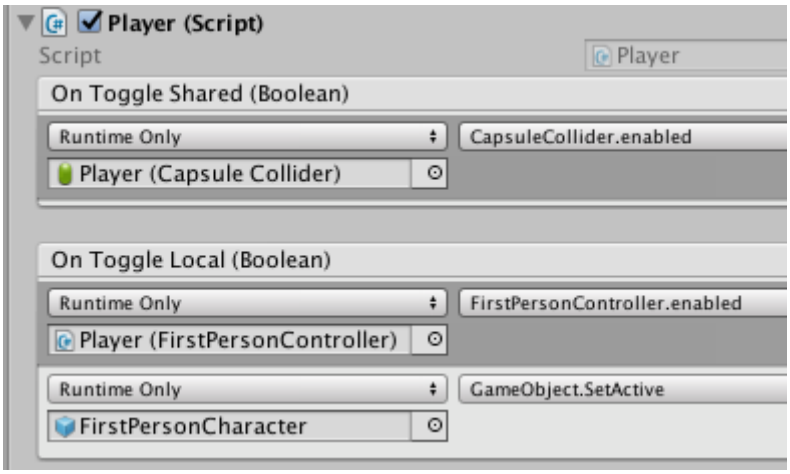
Vous pouvez voir si dessus comment se présente la sélection de la classe.

7.3.4 Réseau

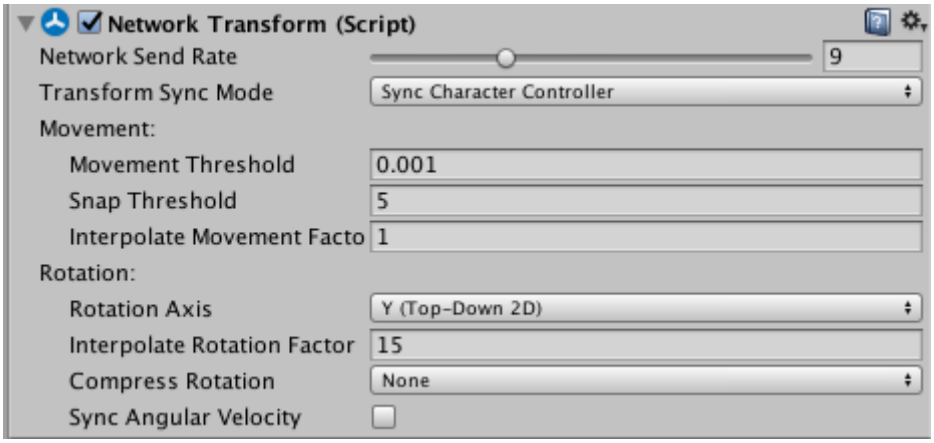
Pour la première soutenance, en attendant la complétion de la map nous avons crée une map nous permettant de commencer à faire notre première implémentation du réseaux. Pour cela nous tout d'abord commencé à implémenté le joueur comme précisé si dessus, puis nous nous sommes penchés sur la documentation d'Unity et plus spécifiquement la partie réseaux. Grâce à ça nous avons vu qu'il est était assez facile d'implémenter le réseaux. Nous avons donc utilisé un composant d'Unity appelé Network Manager et qui nous permet de gérer tout ce qui tourne autour du réseaux.



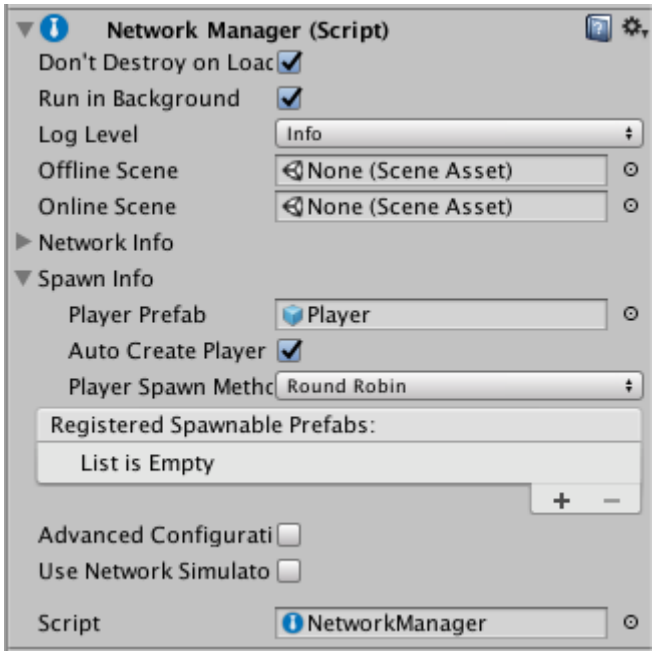
Nous avons remarqué qu'il y a un champ Player Prefab qui permet d'automatiquement fait apparaître un joueur sur une map. Nous avons donc complété notre joueur en lui ajoutant le composant Network Identity qui permet au Network Manager de le détecter bien comme il faut. Nous avons aussi crée un script qui s'occupe de gérer tout les composant de nos différents joueurs, il permet d'activer ou de désactiver notre joueur lors de par exemple l'arrivée d'un joueur ou de son départ.



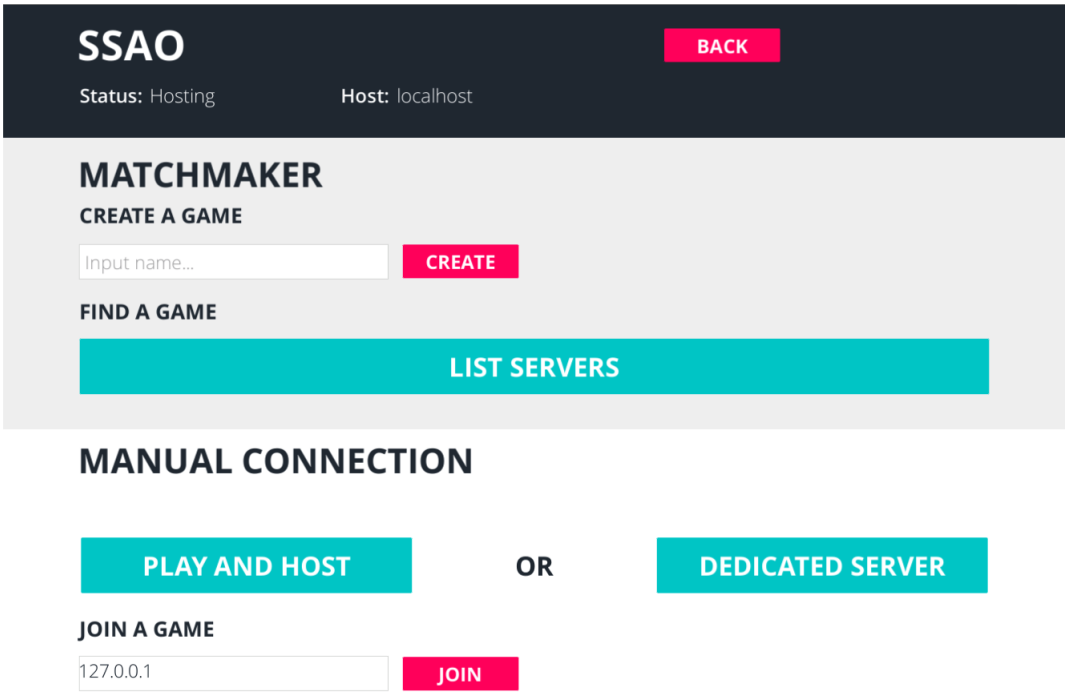
Enfin nous avons ajouté le composant Network Transform ce qui permet au différents joueur d'envoyer leurs déplacements afin que tous les joueurs puisse voir les déplacements des différents joueurs. De plus nous avons aussi crée des spawn point qui permettent de faire apparaître les différents joueurs à des endroits différents.



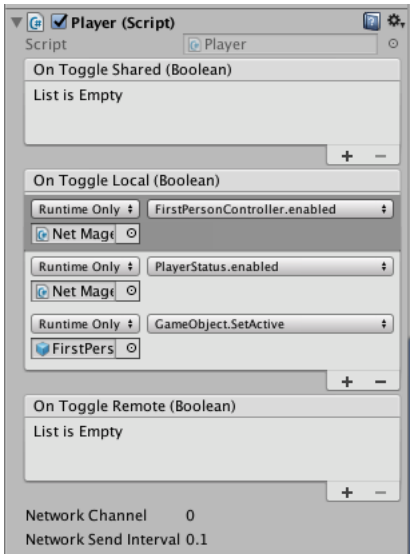
Lors de la dernière soutenance nous utilisions un composant d'Unity, le Network Manager pour gérer la gestion du réseaux et en particulier la connexion ainsi que la gestion du spawn des joueurs grâce au champ Player Prefab.



Mais le Network Manager possèdes aussi des inconvénients, en particulier le fait qu'il est lié à une seule scène ce qui rend le changement entre différentes map plus compliqué. Il a donc fallu nous tourner vers une autre solution. Après quelques recherches nous avons trouvé l'asset Network Lobby de Unity qui implémente toutes les fonctions du Network Manager tout en augmentant les possibilités.



Elle possède pour l'instant une interface basique mais elle nous permet de gérer tout ce qui est en rapport avec le réseaux.
De plus ce Lobby pour le multijoueur est accessible directement depuis le menu principal du jeu.
Elle aura à terme un design plus dans l'esprit de notre jeu ainsi que des fonctionnalités en plus.
Nous avons de plus améliorer le script qui s'occupe de gérer les différents joueurs.



De plus lors de la dernière soutenance la partie réseaux était séparée du projet elles sont désormais réunis.

8 Logiciels Utilisés

Nous avons utilisés les logiciels suivant lors de la réalisation de notre projet :

— **Unity :** Moteur de jeu 2D/3D permettant notamment grâce à des outils intuitifs (moteur physique, audio...) de réaliser des jeux en réseaux et des animations.

— **Git et GitHub :** Logiciel de gestion de version qui nous permettra de partager notre code entre nous et de le stocker sur GitHub.

— **Visual Studio / Rider :** Permettent la programmations des différents scripts C#

— **Blender :** Logiciel de modélisation, d'animation et de rendu en 3D.

9 Problèmes Rencontrés

Lors de la première soutenance nous avons rencontrés comme problèmes que jusqu'à la réalisation de ce projet, personne de notre groupe n'avait jusque la suffisamment utilisé GitHub (voir même pas du tout), ce qui fait que nous nous sommes heurtés à divers problèmes tel que des conflits lors de nos merges, des disparitions inexpliqués de fichiers du à un oubli de commit, voir même à un refus de Git d'ajouter des fichiers ! Ces problèmes nous ont donc fait perdre un temps précieux (voir des fichiers précieux !) mais après moult problèmes nous avons surmonté les difficultés et nous seront donc mieux habilités lors de la prochaine soutenance et pourront délivrer un projet plus complet.
Nous avons eu quelque problèmes également avec les colliders et du navmesh. En effet, vu que nos maps ne sont pas toujours sur le même niveau, nous étions forcé de réduire grandement la taille, pour une question de temps de calcul, ceci nous a forcé a adapter la zone aux contraintes techniques de Unity.

Lors de la deuxième nous avons eu beaucoup moins de problèmes avec GitHub parce qu'un membre de notre équipe était le seul qui s'occupait des merges. Cependant, nous avons aussi eu des problèmes avec l'implémentation du réseau qui ne fonctionnait pas comme on le voulais, ainsi qu'un problème avec le temps de chargement des cartes qui est excessivement long.

10 Ressenti Individuel sur le Projet

10.1 Edouard EISENFISZ, A.K.A. "FireFox"

Nous avons eu beaucoup de problème lors de la première soutenance, et ceux surtout à cause de l'utilisation inadapté de GitHub. Nous avons eu quelque problèmes également avec les colliders et du navmesh. En effet, vu que nos maps ne sont pas toujours sur le même niveau, nous étions forcé de réduire grandement la taille, pour une question de temps de calcul, ceci nous a forcé a adapter la zone aux contraintes techniques de Unity. Nous avons néanmoins rencontré des problèmes de temps de chargement, en grande partie car il y avait trop d'élément dans nos maps. Néanmoins, en dehors de problème technique du à Unity, de mon côté, je n'ai pas rencontré beaucoup de problème. Ce projet fut amusant et intéressant, j'ai pu consolider plusieurs base qui était bancale. J'ai apprécier travailler avec mes camarades.

10.2 Simon QUESNEY, A.K.A. "Qui-Gon-Jinn"

Lors de ce projet, j'estime avoir beaucoup appris du travail en groupe. Au tout début, j'avais beaucoup de mal avec github et les très nombreux problèmes de compatibilité des programmes sur lesquels je passait mes journées inutilement et qui me rongeait. Lors de ce projet je me suis beaucoup concentré sur l'envie que j'avais de rendre ce jeu possiblement un jeu de furtivité, et ainsi j'ai beaucoup travaillé en premier lieu sur l'IA des ennemis. Cependant, lors de ce projet j'essayais aussi d'assister mes camarades et je me concentrais sur ce qui me paraissait nécessaire plutôt que sur mon domaine assigné, c'est ainsi que j'ai pu aider sur la mise en place des barres de HP, du premier système de points de vie et de dégâts, ainsi que sur la plupart des animations. Par ailleurs, j'essayais à chaque fois que je voyais certaines erreurs, de les corriger du mieux que je pouvais même si ce n'était pas ma partie. Malgré cela j'estime avoir contribué au projet du mieux que je pouvais, même si j'estime avoir encore beaucoup à apprendre, notamment apprendre à plus compter sur mes coéquipiers. Mais je me suis tout de même amusé sur ce projet qui était pour moi un défi mais aussi, et ce car le domaine des jeux vidéos me passionne, un réel divertissement.

10.3 Lucas PINOT, A.K.A. "PublicStaticVoid"

J'ai eu beaucoup de mal à prendre en main GitHub et GitHub Desktop mais grâce à la grande patience de mes coéquipiers j'ai pu rapidement faire des progrès et diminuer le nombre de problèmes sur ce logiciel. J'ai également eu du mal à faire mes débuts sur Unity mais en regardant plus en profondeur les différents Tutoriels sur le site du logiciel j'ai trouvé comment arriver à mes fins ce qui m'a permis de beaucoup apprécier ce que je faisais. J'ai adoré faire travailler mon imagination et repousser les limites de mes compétence sur Unity en travaillant sur les sorts des différents personnages et en m'appliquant à leur donner un aspect visuel, une animation poussée et plus ou moins réaliste. L'équilibrage des effets et des dégâts des sorts a aussi été un certain défi car il

est compliqué de prendre en compte toutes les situations dans lesquelles un sort va être utilisé et de leur donner une réelle utilité vis-à-vis du personnage, des autres joueurs et des ennemis. Je suis un peu déçu ne pas avoir eu le temps d'implémenter les objets M/T et normaux mais au vu du travail que j'ai produit je suis tout de même assez fier de moi. Je me rend compte maintenant que j'ai vu trop grand lors de la remise du cahier des charges mais cela m'a permis de me donner à fond pour réussir à atteindre un objectif tout en m'appliquant sur chaque détail. Je suis globalement content de ce que nous avons produit.

10.4 Ouwéis MOOLNA, A.K.A. "Bûcheron"

J'avais déjà utilisé un peu Git ainsi que GitHub pour des projets plus personnels par le passé ce qui fait que j'avais déjà une idée de comment l'utiliser et j'ai donc pu aider mes coéquipiers à mieux prendre en main Git. Cependant j'ai eu un peu de mal à appréhender Unity au début mais grâce à l'aide des tutoriels d'Unity eux mêmes, ceux trouvables sur Internet et de l'aide de mes coéquipiers j'ai réussi à m'en sortir sans trop de mal.
Lors de ce projet je me suis beaucoup concentré sur les parties qui m'ont été assignés dans le cahier des charges, mais j'ai aussi essayé d'aider mes coéquipiers quand je le pouvais. Je me suis surtout concentré sur le réseau en grande partie.
Cependant je pense que grâce à ce projet j'ai pu apprendre beaucoup sur le travail de groupe, mais aussi sur la réalisation d'un projet concret avec un cahier des charges et des soutenances. Et je pense que cette expérience va beaucoup m'apporter et m'aider en tant que futur ingénieur.
J'ai fait de mon mieux pour contribuer à ce projet et j'ai grandement apprécié travailler celui ci. Ce fut une expérience très enrichissante puisque j'ai toujours été intéressé par le domaine du jeu vidéo et en créer un, c'est un rêve qui se réalise.

11 Conclusion

Lors de ce projet, nous avons donc développé en grande partie un jeu vidéo. Nous allons ainsi rapidement retracer ce parcours. Tout d'abord pour notre première soutenance, nous avions déjà réalisé un niveau entier et ce avec uniquement des ressources gratuites de l'asset store. Parallèlement, nous expérimentions déjà sur les contrôles du joueur et une ébauche de multijoueur. Ayant un niveau pour tester le jeu, nous avons pu nous lancer dans la conception plus approfondie de notre jeu. Nous avons ajouté des ennemis, qui serviront à développer leur intelligence artificielle. Nous nous sommes servis de l'outil de navigation de Unity, et avons réussi a créer un script de détection des ennemis assez basique. Nous en avons profité pour finaliser les contrôles basiques du joueur ; permettant un contrôle total de celui-ci. Par ailleurs, nous avons pu réaliser un premier sort ainsi que l'animer totalement ; ce sort faisant office de premier pas vers le système de combat de notre jeu, même si pour cette première soutenance nous n'avions pas encore implémenté un quelconque système de points de vie ou de dégâts. Le joueur en lui même n'avait qu'un modèle de base afin de pouvoir l'animer simplement. Pour les sons il n'existait que les pas du joueur et une musique d'ambiance. Et enfin, pour le site Internet, il a été développé avec l'outil GitHub pages, et ne permettait que d'avoir un bref résumé du jeu, ainsi que des liens de téléchargements.
Pour la deuxième soutenance ; nous avons grandement avancé, nous avons développé un autre niveau, plus ouvert, avec des pièges multiples et mortels. L'IA s'est extrêmement développée, permettant de jouer la carte de la furtivité ; car elle prenait maintenant en compte les bruits de pas du joueur, et nous avions corrigé certaines failles de l'ancienne IA. Le panel de sorts s'est considérablement étoffé, avec maintenant deux classes entièrement jouables, avec un sort de déplacement, un sort faible en coût mais peu puissant, un sort plus puissant et plus coûteux, et enfin un sort ultime, extrêmement coûteux et dangereux. Nous avons par ailleurs implémenté des menus pour jouer le jeu en utilisant ou non la fonctionnalité de multijoueur, ainsi que la possibilité de choisir sa classe et si l'on veut jouer avec des amis, choisir un serveur pour jouer en ligne. Comme nous nous sommes plus focalisés sur la création de nouveaux ennemis, pièges et niveaux ; les animations des joueurs et des ennemis n'ont pas tellement évolué, même si nous avons ajouté des animations de mort avec l'apparition d'un système de points de vies et de dégâts. De même pour les sons, même si nous avons fait quelques ajustements sur la musique d'ambiance.
Enfin nous avons ajouté des modèles pour les joueurs et étoffé le panel d'animations. La dernière map est en construction et permettra de se battre au sommet d'une gigantesque tour. Toutes les classes ont leurs sorts au complet. Le jeu en est donc à sa phase finale.
Voici donc venu la fin de notre projet, l'aboutissement de 6 mois de travail. Avant de commencer ce projet, nous ne nous connaissions pas, la réalisation de ce projet nous à permis de nous connaître un peu mais aussi de nous apprendre énormément sur le travail en équipe.
Au cours de la réalisation du projet, nous avons été confrontés à de nombreux problèmes

qui nous ont poussés à nous dépasser pour trouver des solutions. Ainsi, nous avons non seulement bénéficié d'une expérience considérable en C#, mais également dans beaucoup d'autres domaines, langages, et logiciels.
L'expérience de groupe fut bénéfique en tous points, nous enseignant la gestion d'un groupe, du travail à fournir par chacun, ainsi que l'organisation du travail, afin de réaliser des tâches pour une date précise.
Notre projet est quasiment fini et se rapproche donc énormément de nos attentes initiales lorsque l'on avait rédigé notre cahier des charges.
Nous pensons que l'expérience que nous avons acquis lors de ce projet et tout particulièrement l'expérience du travail en équipe avec des gens que l'on ne connaît pas forcément, nous à été très bénéfique et nous servira énormément lors de notre futur métier d'ingénieur.

12 Annexes

12.1 Assets Utilisés

Assets employés pour la map :
https ://assetstore.unity.com/packages/3d/environments/industrial/storage-building-50430
https ://assetstore.unity.com/packages/3d/environments/industrial/building-kit-1-68179
https ://assetstore.unity.com/packages/essentials/asset-packs/unity-particle-pack-73777
https ://assetstore.unity.com/packages/3d/props/exterior/victorian-pbr-streetlights-50828
https ://assetstore.unity.com/packages/3d/environments/urban/victorian-pie-shop-21541
https ://assetstore.unity.com/packages/3d/props/industrial/pipes-kit-64170
https ://assetstore.unity.com/packages/2d/textures-materials/floors/hand-painted-stone-texture-73949
https ://assetstore.unity.com/packages/2d/textures-materials/metals/yughues-free-pbr-metal-plates-35362
https ://assetstore.unity.com/packages/3d/environments/fantasy/modular-fantasy-bridges-99940

Assets employés pour les unités :
https ://assetstore.unity.com/packages/3d/characters/humanoids/strong-knight-83586
https ://assetstore.unity.com/packages/3d/characters/jeremy-base-male-character-33083
https ://assetstore.unity.com/packages/3d/animations/warrior-pack-bundle-3-free-47320
https ://assetstore.unity.com/packages/3d/animations/warrior-pack-bundle-1-free-36405
https ://assetstore.unity.com/packages/3d/animations/warrior-pack-bundle-2-free-42454

Asset employé pour les sorts :
https ://assetstore.unity.com/packages/vfx/particles/spells/ky-magic-effects-free-21927
https ://assetstore.unity.com/packages/3d/props/weapons/low-poly-fantasy-sword-65120
https ://assetstore.unity.com/packages/3d/props/weapons/shield-61351
https ://assetstore.unity.com/packages/3d/environments/sci-fi/space-shooter-asteroids-96444
https ://assetstore.unity.com/packages/essentials/asset-packs/standard-assets-32351

Asset employé pour les textes du menu :
https ://assetstore.unity.com/packages/essentials/beta-projects/textmesh-pro-84126

Asset employé pour le lobby :
https ://assetstore.unity.com/packages/essentials/network-lobby-41836

12.2 Glossaire

Map : Carte
Level Design : Conception du Niveau
GameObject : objet du jeu que voit Unity
Rigidbody : composant d'un objet qui permet une interraction physique entre cet objet et son environnement
IA : Intelligence Artificielle
Collider : Composant d'un objet permettant de gérer les collision entre lui et un autre objet
Script : programme permettant d'implémenter un comportement d'un objet ou un évènement
Gatling : c'est une arme
NavMesh : composant permettant de délimiter une zone dans laquelle un IA peut se déplacer
AssetStore : rubrique du site de Unity dans lequel on peut télécharger des Assets
Assets : fichiers contenant des objets, des scripts ou autres déjà faits par une personne en dehors du projet