

SSAO:

Rapport de Projet

Première Soutenance

Groupe: **Les Rejetés**

Ouwéis MOOLNA, Edouard EISENFISZ, Lucas PINOT, Simon QUESNEY

16 Mars 2018

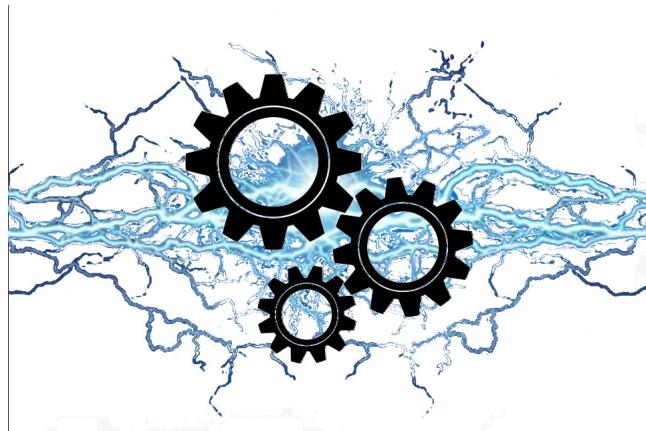


Table des matières

1	Introduction du Rapport de deuxième soutenance	3
2	Distribution	4
3	Progression	4
4	Le Projet	5
4.1	Les Contrôles	5
4.1.1	Contrôle Basique	5
4.1.2	Contrôle Avancé	5
4.2	Ennemis	5
4.2.1	Les Types d'Ennemis	5
4.2.2	L'IA ennemie	5
4.3	Graphisme	5
4.3.1	Level Design	5
4.3.2	Apparence des joueurs	6
4.4	Sorts	6
4.4.1	Types de Sort	6
4.4.2	Animations des sorts	6
5	Présentation du travail réalisé	7
5.1	Edouard AKA FireFox	7
5.1.1	La Map	7
5.2	Simon AKA Qui-Gon-Jinn	10
5.2.1	IA des ennemis (et ennemis)	10
5.2.2	Design du niveau	11
5.2.3	Site	11
5.3	Lucas AKA PublicStaticVoid	12
5.3.1	Les sorts (Animation)	12
5.3.2	La suite des évènements :	14
5.4	Ouwéis AKA Bûcheron	15
5.4.1	Contrôles Basiques	15
5.4.2	Animations Et Sons	15
5.4.3	Réseau	16
5.4.4	Travail à faire pour la prochaine soutenance	17
5.5	Logiciels Utilisés	18
5.6	Problèmes rencontrés	18
6	Annexes	19
6.1	Assets	19
7	Conclusion	19

1 Introduction du Rapport de deuxième soutenance

SSAO sera un jeu de tir en coopération. Ce sera une 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 pour 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 réalisé.

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.

2 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	

3 Progression

module	oral 1	Etat actuel	oral 2	final
Contrôle de base	75%	En avance	100%	100%
Level design	33.33%	A l'heure	66.66%	100%
IA, et mouvement des ennemis	22.22%	A l'heure	56.65%	100%
Graphisme	25%	A l'heure	60%	100%
Animation	20%	A l'heure	60%	100%
Menu	0%	—	50%	100%
Sons	20%	A l'heure	60%	100%
Site	10%	A l'heure	65%	100%
Réseau	40%	A l'heure	70%	100%
Procédure d'installation	0%	—	0%	100%
Mode Furtif :	0.0001%	—	42.123%	100%
Versus :	20%	A l'heure	60%	100%
Contrôles avancés	20%	A l'heure	50%	100%
Canons à ennemis :	0%	—	30%	100%
Boss	0%	—	20%	100%
Code de Triche	20%	//	50%	100%

4 Le Projet

4.1 Les Contrôles

4.1.1 Contrôle Basique

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

4.1.2 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 implantés aussi bien sur clavier et souris que sur manettes.

4.2 Ennemis

4.2.1 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 compétence spéciale en dehors de leurs attaques de base.

4.2.2 L'IA ennemie

L'IA des ennemis en vague :

L'IA la plus simple ici qui 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 en sa direction, dépendant des ennemis elle va soit le cibler à distance soit l'attaquer au corps-à-corps.

4.3 Graphisme

4.3.1 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 objectif, seront créées autour de ces objectifs. Par exemple, si l'objectif est de protéger un point, ce point sera au centre de la carte. À 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 le style médiéval, et jouant 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 faudra casser celles-ci pour continuer. C'est que les cartes auront également des salles cachées avec des pièges, contenant des bonus facilitant la suite du niveau.

4.3.2 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 lambda.

4.4 Sorts

4.4.1 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 recharge, 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 bâton lanceur de flammes pour le Mage Guerrier.

Les sorts d'objets magiques ou technologiques (M/T) :

Les objets M/T seront acquis au fil de la partie, pendant chaque niveau, 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 mais 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 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 recharge avant de pouvoir être réutilisé à nouveau.

La bombe jetable peut par exemple être un objet T dont l'utilisation entraînera un temps de recharge avant réutilisation.

4.4.2 Animations des sorts

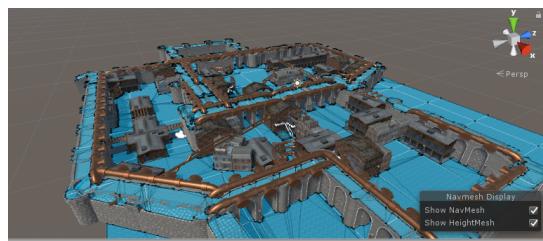
Les animations des sorts seront toutes les animations en rapport avec les sorts, et elles seront différentes en fonction du sort choisi, par exemple les sorts physiques n'auront pas la même animation que les sorts magiques. Mais il y aura aussi une animation sur le personnage quand les sorts auront fini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

5 Présentation du travail réalisé

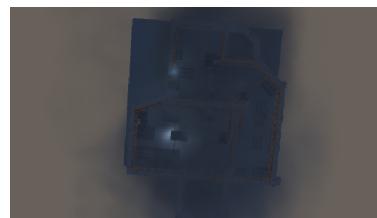
5.1 Edouard AKA FireFox

5.1.1 La Map

Pour permettre aux autres de pouvoir avancer, j'ai également fait une petite map de test, car je savais que la map en elle-même prendrait plus de temps. Nous avons, Simon et moi, donc travaillé de coeur pour la faire. J'avais pour projet au départ de faire plusieurs modèle via Blender. Je me suis vite rendu compte que les textures que j'avais fait dans blender n'avaient pas du tout le même rendu sur Unity. En effet, les objets avaient un effet transparent. J'ai donc utilisé de nombreux asset pour faire la map. Il n'y a pas en tant que tel d'asset dans le style Steampunk, j'ai donc du associer différent modèle qui pris seul, 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 utilisons un fog pour créer une ambiance, mais ceci n'étant pas pratique pour faire les test, nous ne la mettrons pas pour le moment. Voici quelque image sur l'apparence de la ville.



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



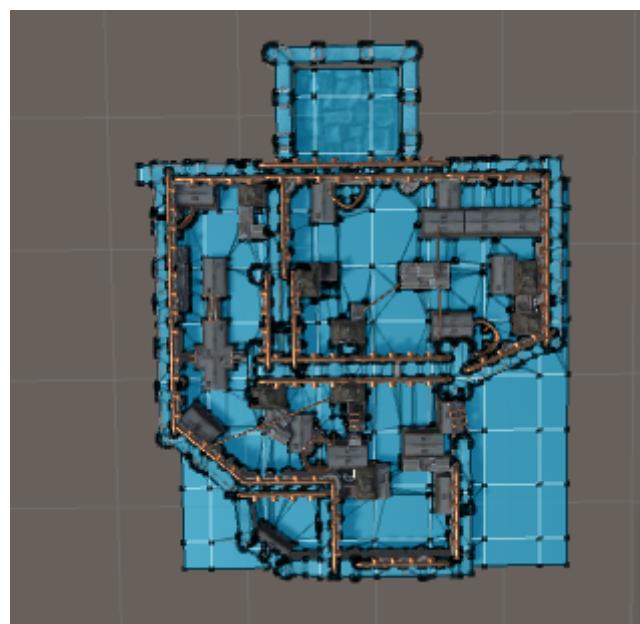
La vue à partir de la terrasse d'un bâtiment.



Une petite ruelle.



J'ai eu quelque problème pour ce qui est du navmesh. En effet, pour permettre que l'on puisse passer sous les ponts, j'ai 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 limité par les capacités de Unity.



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



J'ai également fait des scripts pour permettre d'avoir des tonneaux explosifs que je vous présenterais durant la présentation. (ils ne sont pas encore dans l'implémentation pour des raisons de problèmes de merge dans Github). A ce moment, je n'avais pas encore certain script, mais j'avais besoin de pouvoir tester les tonneaux.

```

1  using UnityEngine;
2  [using System.Collections;
3
4  public class Tonneau : MonoBehaviour
5  {
6      public Transform Trans;
7      public Explosion expl;
8      public Rigidbody rig = new Rigidbody();
9      private int health;
10     void Start()
11     {
12         pos = rig.transform.position;
13     }
14     void OnTriggerEnter(Collider other)
15     {
16         if ((other.tag == "Player") || (other.tag == "Enemy") || (other.tag == "Explosion"))//ajoutez dans les conditions
17         {
18             Instantiate(expl, Trans.position, transform.rotation);
19             Destroy(Trans.gameObject);
20         }
21     }
22 }

```

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

```

1  using UnityEngine;
2  [using System.Collections;
3
4  public class Tonneau : MonoBehaviour
5  {
6      public Transform Trans;
7      public Explosion expl;
8      public Rigidbody rig = new Rigidbody();
9      private int health;
10     void Start()
11     {
12         pos = rig.transform.position;
13     }
14     void OnTriggerEnter(Collider other)
15     {
16         if ((other.tag == "Player") || (other.tag == "Enemy"))//ajoutez dans les conditions le tag des projectiles
17         {
18             Instantiate(expl, Trans.position, transform.rotation);
19             Destroy(Trans.gameObject);
20         }
21     }
22 }

```

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, lancant le script explosion. Ceci permettra la création de piege, voir même de champ de mine. Pour ce qui est des portes : Je compte créer une entité de type ennemi, sans pour autant que ce soit un bâtiment. Je ne souhaite pas créer un bâtiment car je pense que cela causerait des problèmes avec le navmesh. Elle spawnerait au début de la partie comme les ennemis. Bien évidement, elles ne bougeront pas. Je compte implémenter également le réseau, et donc de permettre à plusieurs joueurs de participer.

Je compte également faire une autre map dans un style, toujours steampunk, mais

5.2 Simon AKA Qui-Gon-Jinn

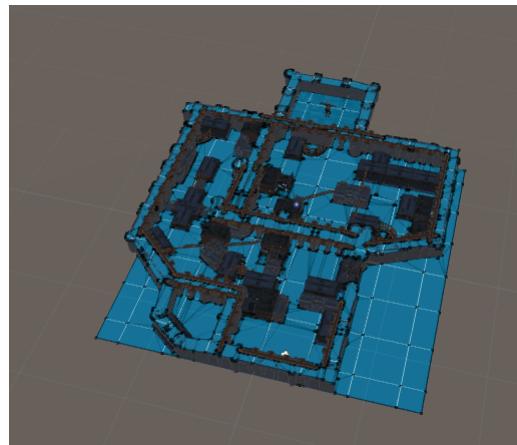
5.2.1 IA des ennemis (et ennemis)

afin de pouvoir experimenter l'IA des ennemis, leur mouvement etc. j'ai implémenté des ennemis dans le jeu



le design des ennemis est sujet à amélioration mais il est 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, j'ai construit pour la map un champ de navigation (nav mesh).



, qui permet de définir leur zone parcourable. et leur ait 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. la prochaine étape sera l'amélioration de cette IA avec des patterns de déplacement et un repérage des ennemis au son.

5.2.2 Design du niveau

Le design de ce premier niveau se rapproche plus d'un niveau de type couloir, avec plusieurs chemins possibles, des ennemis humanoides, 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 aboutiront à des objets collectables 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 dur à faire notamment par rapport à la map de versus ou la map de vague d'ennemis.

5.2.3 Site

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, j'ai créé un site en utilisant un fichier Markdown dans lequel j'ai mis ce qu'on souhaitait avoir dans notre site, nous avons de plus choisi un thème pour ce site.



Présentation

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 pour 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 jeu 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. Notre but dans ce projet est de créer un jeu intégrant différents concepts déjà présents dans plusieurs jeux qui nous ont grandement inspirés et qui sont :

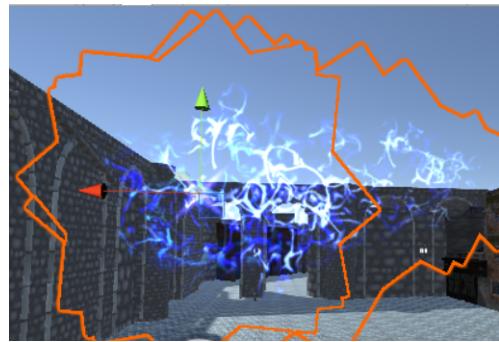
- Warhammer : End Times - Vermintide
- DISHONORED
- Call Of Duty
- Splinter Cell : BlackList
- Deux Ex : Human Revolution

ce site contient les liens vers notre cahier des charges et notre rapport de projet, il est pour l'instant constitué d'une seule page mais va par la suite être multi-pages.

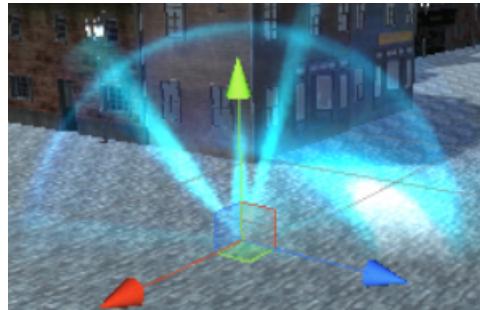
5.3 Lucas AKA PublicStaticVoid

5.3.1 Les sorts (Animation)

La création des sorts a pris plus de temps que prévu. En effet, les sorts étant essentiellement des particules, j'ai préféré me 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. 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. Voici à quoi elle ressemble :



L'animation du deuxième sort n'est pas finie, le concept est que 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. Voici à quoi ressemble l'explosion :



Les systèmes de particule du premier et du deuxième sort proviennent d'un Asset nommé "KY Magic Effects Free", ce qui m'a permis de récupérer les graphismes voulus pour mes sorts. La difficulté résidait essentiellement dans la capacité du joueur à activer le sort, activation qui se résume dans une suite d'action (pour le premier sort uniquement) : création d'un GameObject qui est la boule d'énergie en elle-même, déplacement du GameObject dans une direction donnée et suppression du GameObject au contact d'un mur, d'un collider quelconque. Pour cela, il m'a fallu ajouter un collider et un rigidbody au système de particule de la boule électrique, modifier le script du FirstPersonController et créer le script du mouvement en ligne droite dans la direction donnée par la souris que j'ai appelé Mover. La modification du script FirstPersonController me permet de faire apparaître le sort voulu, à l'endroit voulu, dans la direction voulue et à une fréquence maximale voulue. Je vous donne un visuel du script pour une meilleure compréhension :



Pour l'explication du lancement des sorts, je vais me concentrer sur cette zone en particulier :



Les champs "Impulsion" et "Dash Rate" sont les champs servant à l'activation du second sort, donc je ne vais pas en parler, je préfère parler du premier sort car son animation est finie.

Tout d'abord, le champ "Skillshot" sert à choisir le sort que l'on veut lancer, le système de particule adapté.

Le champ "Shotspawn" sert à définir l'emplacement où le sort va apparaître par rapport à la caméra, cet emplacement est l'emplacement un GameObject fils de la caméra ce qui fait que qu'il sera toujours au même emplacement par rapport à la caméra.

Le champ "Use Rate" sert à limiter la fréquence de lancement du sort, ici, le sort pourra être lancé toutes les deux secondes en appuyant sur la touche configurée. A chaque pression de cette touche, si l'on peut lancer le sort, un nouveau GameObject va être créé et ajouté à la liste des GameObject existants. Il reste maintenant à le faire bouger et disparaître.

Pour cela j'ai créé le script Mover que voici :

```
1  using ...
5
6  public class Mover : MonoBehaviour
7  {
8
9      public float speed;
10
11     private Rigidbody rb;
12
13     private void Start()
14     {
15         rb = GetComponent<Rigidbody>();
16         rb.velocity = transform.forward * speed;
17     }
18
19     private void OnTriggerEnter(Collider other)
20     {
21         Destroy(gameObject);
22     }
23
24 }
```

Ce scrpit est finalement assez simple. On choisi la vitesse à laquelle le sort va se déplacer et on initialise un Rigidbody "rb" quelconque. Dans le Start(), on associe le "rb" avec le rigidbody associé au système de particule du sort et on lui donne une vitesse constante dans la direction donné, donc tout droit, qui est la direction indiquée par la souris multipliée par la vitesse donnée plus tôt. Pour que les GameObject ne s'accumulent pas, et en accord avec ce que doit faire le sort (il doit disparaître lorsqu'il a fini son effet), j'ai ajouté la dernière fonction qui détruit le GameObject (le sort) contrôlé par le script Mover dès qu'il y a collision entre le collider du sort avec un autre collider.

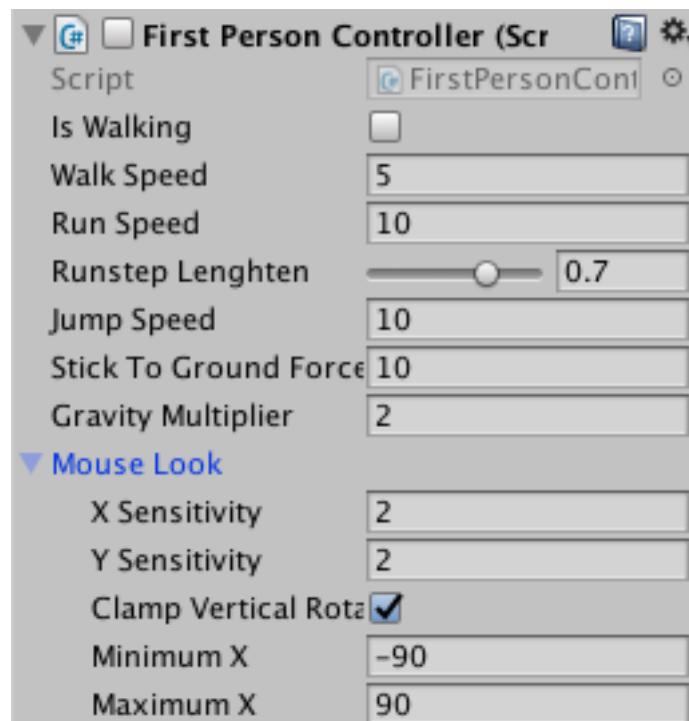
5.3.2 La suite des évènements :

Pour la prochaine soutenance, il faut que je termine l'animation du deuxième sort et que j'en fasse au moins 6 autres avec un objet spécial à récupérer sur la map. Je dois également faire une ébauche du menu principal du jeu et faire un boss de niveau. Et enfin je dois commencer à scripter mes sorts pour qu'ils aient un réel impact sur les ennemis et les alliés.

5.4 Ouwéis AKA Bûcheron

5.4.1 Contrôles Basiques

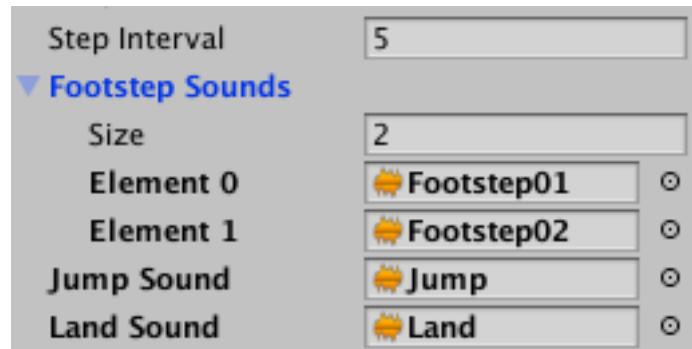
J'ai d'abord commencé par implémenter des contrôles basiques de déplacement, c'est à dire le déplacement grâce au touches ZQSD. Puis en regardant un peu la documentation d'Unity, j'ai vu qu'il était possible de récupérer des axes, c'est à dire l'axe Horizontale et Verticale par rapport aux touches définies dans les Inputs. Grâce à cela j'ai pu implémenter des déplacement au clavier, mais aussi avec une manette. J'ai par la suite implémenté les déplacements de la caméra par rapport à la souris, puis grâce à la documentation j'ai pu implémenter des déplacement de caméra avec souris et manette. J'ai par la suite implémenté le saut, et la possibilité de courir depuis un clavier ou une manette. Ce qui fait que les Contrôles Basiques étaient complets. Puis grâce à l'aide d'Edouard nous avons implémenté les contrôles en réseaux ce qui fait que les mouvements des différents joueurs sont bien effectués sans problèmes mais aussi que chaque joueur verra les autres joueurs bouger.



5.4.2 Animations Et Sons

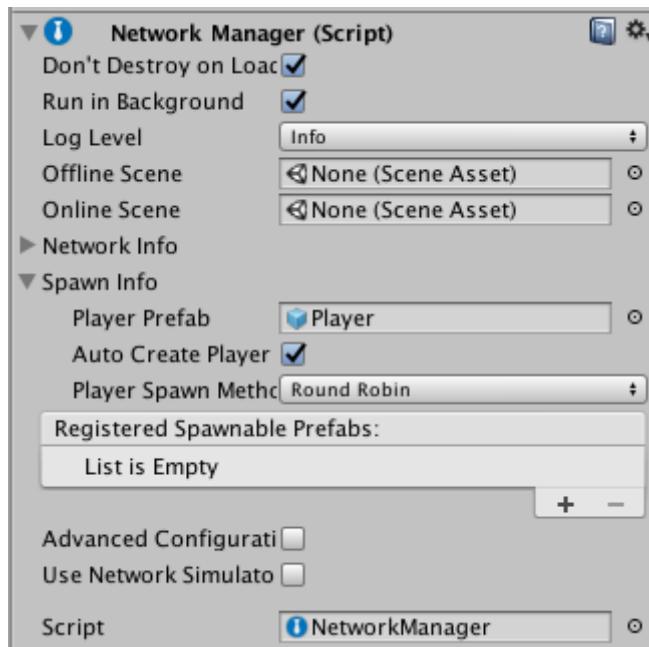
Pour l'instant les ennemis, mais aussi le joueur possède des animations lors de leurs déplacement. Le joueur possède de plus une animation lors de son saut. Ces différentes animations sont de plus implémentés en réseaux.

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.

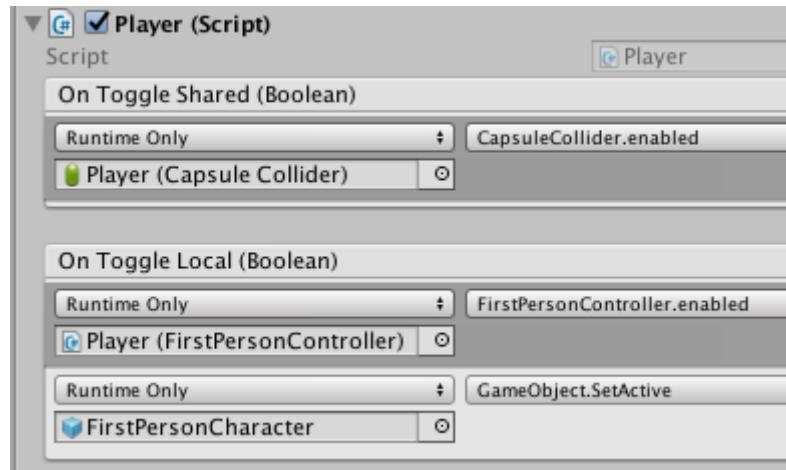


5.4.3 Réseau

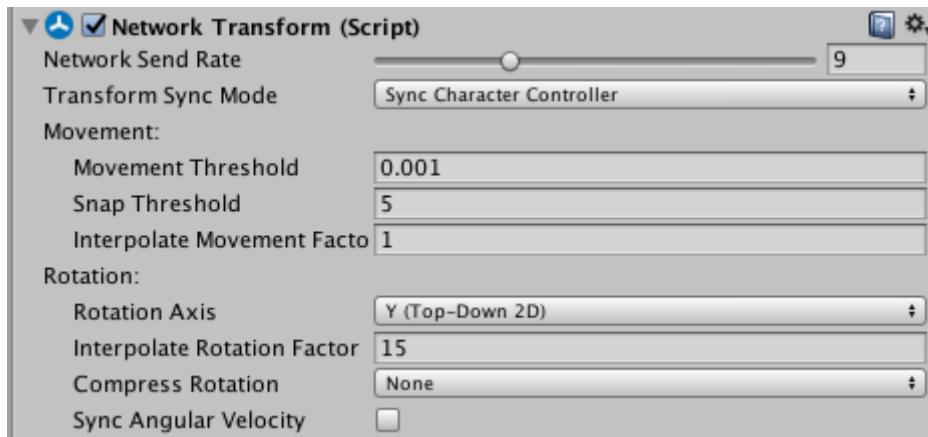
Edouard et moi-même avons crée une map nous permettant de commencer à faire notre première implémentation du réseaux. Pour cela j'ai tout d'abord commencé à implémenté le joueur comme précisé si dessus, puis nous nous sommes penchés sur la documentation d'Unity et plus spécifiquement la partie réseaux. 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.



5.4.4 Travail à faire pour la prochaine soutenance

La partie contrôles basiques étant terminée, je vais à présent pouvoir me concentrer sur la suite c'est à dire les principales les animations et les sons, le réseau et aider à l'implémentation des boss et codes de triches, mais aussi et surtout l'unification de la partie réseau dans la map.

5.5 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#

5.6 Problèmes rencontrés

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 inexplicées 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.

6 Annexes

6.1 Assets

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>

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/environments/fantasy/modular-fantasy-bridges-99940>

Asset employé pour les sorts : <https://assetstore.unity.com/packages/vfx/particles/spells/ky-magic-effects-free-21927>

7 Conclusion

Malgré bien des difficultés tant à la fois technique que de défis dû à nos compétences, nous sommes dans les temps pour ce qui est de notre avancement, nous avons même de l'avance dans certain domaine. Nous avons déjà de nombreuse idées et de solution pour continuer ce projet.