E.S.U.

Emergency Service Unit

Rapport de projet

Louis Place
Maxence Oden
Nathan Rabet
Vincent Libeskind

Contents

1 Introduction

Dès la dernière soutenance, nous avions déjà intégré beaucoup plus de choses que ce que nous étions censé produire pour cette soutenance intermédiaire. Pour cette dernière soutenance, nous avons terminé d'implémenter toutes les fonctionnalités du cahier des charges. De plus, de nouveaux systèmes ont été conçu.

En passant par la mort par ragdoll, l'intelligence artificielle de nos PNJ, les particules ou même la création de tous nouveaux sons, ce projet est maintenant plein de richesses inédites qui n'attendent que d'être découverte.

2 Notre projet

2.0.1 Concept de base

ESU est un jeu axé sur l'action et le PVP (joueur contre joueur). Le but de ce jeu est simple : deux équipes s'affrontent sur une carte au cours d'une partie : ***Les Agresseurs***, doivent détruire et tuer les citoyens et les bâtiments de la carte. ***Les Sauveteurs***, doivent défendre les bâtiments et secourir les citoyens.

2.0.2 Les classes

Équipe des sauveteurs:

- Policier: Possède une arme balistique afin de pouvoir attaquer les agresseurs pour protéger ses coéquipiers.
- Pompier: Possède un extincteur pour faire face aux flammes et une hache pour sauver les victimes.
- Médecin: Possède des compétences en médecine. Il peut donc soigner aussi bien les victimes que ses coéquipiers.

Équipe des agresseurs:

- Mercenaire: Possède une arme balistique afin de pouvoir attaquer les sauveteurs.
- Pyroman: Possède un lance-flamme pour brûler les bâtiments.

- Drogueur: Possède des compétences en médecine. Il peut soigner ses coéquipiers et vendre de la drogue au PNJ pour les attirer et gagner des points.

2.0.3 Déroulement d'une partie

À chaque lancement de partie, un minuteur s'enclenchera. À la fin de celui-ci, la partie s'arrêtera. L'équipe qui aura récolté le plus de points aura gagné la partie. Les points se récoltent en fonction du nombre de personnes et de bâtiments sauvés pour les sauveteurs, et du nombre de personnes tuées et de bâtiments détruits pour les agresseurs. Quand l'équipe sauveteurs ramène un PNJ à un point de contrôle ou empêche la destruction d'un bâtiment, elle gagne des points. Quand l'équipe agresseurs tue un PNJ ou casse un bâtiment, elle gagne aussi des points. De plus, si un membre d'une équipe tue un joueur de l'équipe adverse : l'équipe adverse perd des points. Les joueurs doivent combiner stratégie et coopération afin de récolter suffisamment de points pour gagner une partie.

2.0.4 Tableau de répartition des tâches

Tâches	Louis	Maxence	Nathan	Vincent
Map	X			X
Assets	X			X
Animation		X		X
Son			X	X
Menu		X	X	
IA	X	X		
Gameplay	X		X	
Multijoueur		X	X	
Menu/HUD		X	X	

Table 1: Tableau de répartition des tâches

2.0.5 Tableau d'avancement

Soutenance	1	2	3
Tâches	- Multijoueur - Déplacement des personnages - Menu	- Déroulement d'une partie - Mécanisme des personnages - Partie sonore	- IA - Système de points - Équilibrage du jeu
Map	33%	66%	100%

Table 2: Tableau d'avancement

3 Avancement final du projet

3.1 Le Multijoueur

Pour faciliter la création et le paramétrage du mode multijoueur de notre projet, nous avons décidé d'utiliser la bibliothèque *Photon*. L'utilisation de cette bibliothèque nous donne accès à de multiples fonctions et de scripts pour la création d'un multijoueur. Un des grands avantages de *Photon* est qu'il nous donne accès à ses serveurs ce qui permet de simplifier la gestion des salles de joueurs. Nous avons donc une capacité de 20 joueurs en simultané.

Figure 1: Logo Photon

3.2 Assets utilisé

Pour notre projet, nous avons décidé d'utiliser le style graphique nommé ***LowPoly***. Le LowPoly est un style graphique cartoon basé sur le maillage polygonal : il est donc composé de polygons. Son choix est à la fois stratégique et esthétique : en effet, la faible quantité de polygone dans des éléments graphiques permet d'augmenter les performances de rendu du jeu. De plus, malgré son aspect cubique et peu réaliste, un monde LowPoly bien conçu permet de créer une véritable ambiance et d'apporter une touche artistique au jeu. De plus, la conception de graphique LowPoly est relativement légère comparée aux autres styles graphiques.

Pour trouver nos modèles graphiques Low Poly, nous avons utilisé la bibliothèque présente dans Unity : l'Unity Asset store. C'est une bibliothèque proposant des milliers de modèles 3D.

3.2.1 PolyWorld

PolyWorld est un asset ajouté lors de la première soutenance. Notre projet était basé sur un style graphique Low Poly, PolyWorld permet de convertir le composant terrain d'Unity en centaine de polygones. Cela permet donc d'avoir un terrain LowPoly tout en conservant les avantages d'un terrain Unity. Les terrains des deux cartes ont donc été converti en terrain LowPoly. Cet asset permet aussi de convertir des objets non LowPoly en objets

Lowpoly. C'est donc une fonctionnalité intéressante pour notre projet.

3.2.2 Polygon City

Largement utilisé après la seconde soutenance, PolygonCity nous a permis de réaliser entièrement des villes réalistes et en conformité avec la charte graphique de notre projet. En effet on retrouve dans cet asset de nombreux bâtiments comme des restaurants, magasins, boutiques en tout genre, mais également des détails comme des déchets, des signalisations et du mobilier urbain. PolygonCity est venu remplacer l'ancien asset premièrement utilisé pour les bâtiments.

3.2.3 Polygon Nature et FancyClouds

Cet asset est un asset ajouté tout récemment. Il offre de nombreux objets et détails en tout genre concernant le milieu naturel. Il a d'abord été largement utilisé dans la carte deux, notamment pour la richesse en végétaux qu'il propose. En effet cet asset offre plusieurs types de plantes qui ajoute une nouvelle palette de couleurs aux cartes.

Figure 2: Exemple d'asset de Polygon Nature

Finalement, cet asset offre aussi de nouveaux arbres qui sont venus remplacer les anciens, c'est le cas des sapins qui sont maintenant beaucoup plus détaillés et esthétiques. On note également que l'ensemble des végétaux simule un effet de vent en frémissant dans une direction. Un fond de carte a fini de compléter le réalisme et l'esthétisme des cartes.

FancyClouds est un asset simple et pourtant ayant un grand impact dans le rendu des cartes, il s'agit d'un asset qui nous a permis l'ajout de nuages de style *Lowpoly*. Celui-ci a été couplé avec un script permettant aux nuages de se déplacer à l'infini, offrant au joueur la possibilité d'apprécier la beauté du ciel.

Figure 3: Exemple des nuages Lowpoly

3.3 Les Cartes

3.3.1 Introduction

Comme prévu les deux cartes qui composent ESU sont finalisées avec le niveau de détails que nous avons espéré atteindre. Les deux cartes sont bien loin de l'état initial présenté durant la toute première soutenance. La première carte offre maintenant un panel d'environnements riches et variés et la seconde carte propose un environnement fourni et entendu.

3.3.2 Carte 1

Lors de la première soutenance, nous avons commencé le développement de cette carte. La première étape a été de schématiser la carte : sa taille, les emplacements différentes zones ainsi que leurs contenus. Nous avons également réfléchi à des lieux stratégiques utiles à l'expérience de jeu.

Trois zones principales ont été définies :

- La zone urbaine : la ville [marron]
- La zone de verdure : le parc, rivière et lac [vert]
- La zone montagneuse : forêt et montagne [gris]

Depuis la deuxième soutenance, une nouvelle zone à été crée

- La zone désertique : cactus et rochers

La carte à aussi des limites définies et elle mesure 250 x 250 mètres.

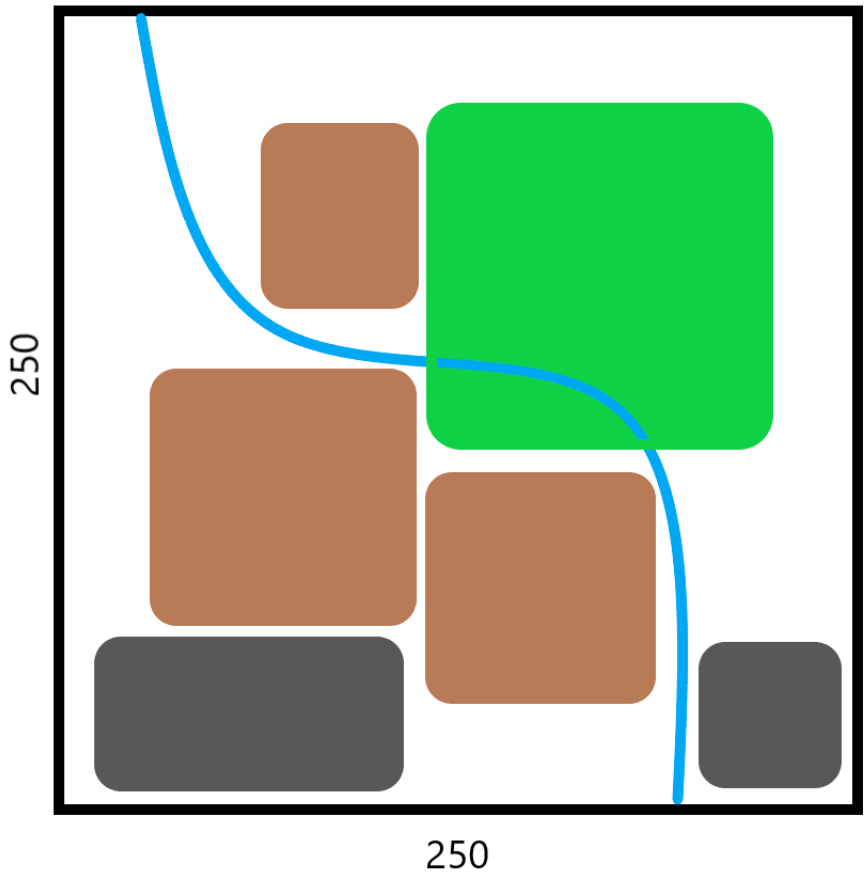


Figure 4: Schéma de la carte lors de la première soutenance

La plupart des lieux stratégiques ont été créés dès la première soutenance. On retrouve une rivière traversant toute la carte en diagonale : elle permet de créer une barrière physique obligeant les joueurs à utiliser les ponts à disposition. De plus, la présence d'un lac permet d'obliger les joueurs à contourner cette zone pour pouvoir accéder au reste de la carte. Des barrières entourent une partie du parc ce qui empêche les joueurs de retourner en ville à certains endroits.

La zone urbaine

La zone urbaine est un regroupement de bâtiments, de voitures, de civils, d'activités commerciales. Elle est une des zones les plus importantes du jeu puisque les principaux objectifs à atteindre s'y trouvent. Lors de la première soutenance, les premières routes et bâtiments ont été implémentés. Un premier plan de la ville a été établi.

La deuxième soutenance à amené de nouveaux bâtiments et la grande majorité de la zone urbaine a été finalisé. On peut y retrouver : des poubelles, des voitures, des plots, des poteaux incendies, des feux, des arrêts de bus, un distributeur de billets, des bouches d'égouts, des bancs, des marchands de nourriture et autres.

Figure 5: Zone urbaine de la deuxième soutenance sur la carte 1

Pour la troisième soutenance, l'objectif était de travailler sur les finitions de la zone urbaine. Nous avons donc travaillé sur les petits détails qui donnent vie à la ville. On peut y retrouver : des déchets par terre, des sacs-poubelles, des lampadaires avec de la lumière, des cartons de livraison, des boîtes aux lettres, des bancs, des panneaux publicitaires, des armoires électriques, etc...

Figure 6: Zone urbaine de la troisième soutenance sur la carte 1

La zone de verdure

Comme pour la zone urbaine, cette zone a été implémenté dès la première soutenance. On retrouve deux parties coupées par la rivière : une petite et une grande contenant le lac. Nous avons ajouté deux types d'arbres ainsi que de l'herbe.

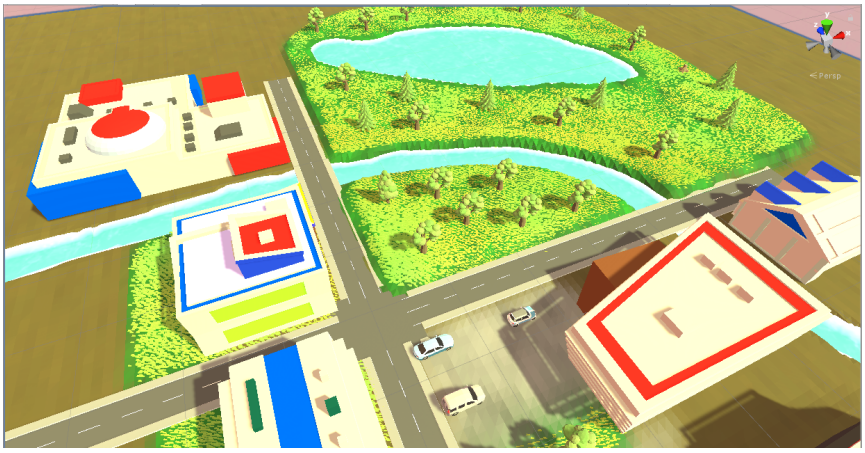


Figure 7: Zone de verdure de la première soutenance sur la carte 1

Pour la deuxième soutenance, nous avons ajouté de nouveaux éléments : des particules de feuilles, des rochers, des champignons et des branches. L'herbe a également été changé. Tous ces petits détails permettent de créer un environnement agréable pour les joueurs.

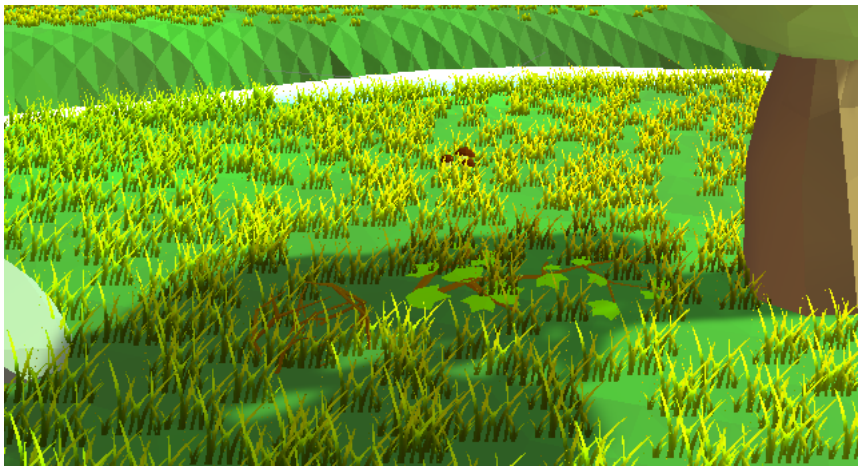


Figure 8: Zone de verdure de la deuxième soutenance sur la carte 1

Enfin nous avons implémenté de nouveaux arbres, de l'herbe et des plantes pour la troisième soutenance. En effet, ces nouveaux éléments sont plus adaptés avec le reste de la carte. Nous avons utilisé, comme pour la carte 2, le système *"paint tree"* d'Unity qui permet de placer les arbres facilement. Des finitions ont également été apporté : il y a maintenant des papillons, des plantes grasses, différents types de fleurs ainsi que des clôtures délimitant le parc.

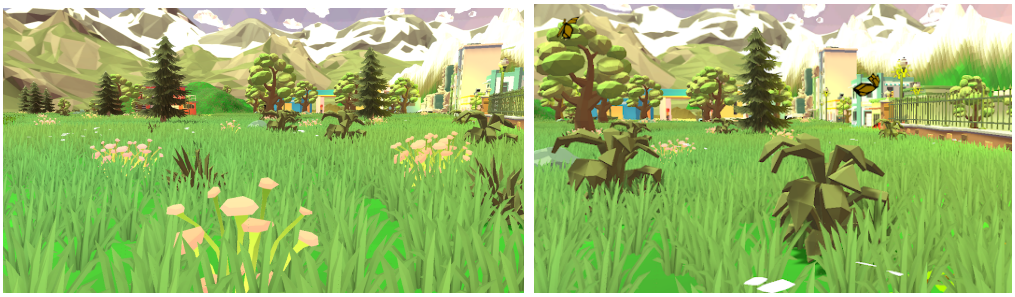


Figure 9: Zone de verdure de la troisième soutenance sur la carte 1

Le lac, situé dans le parc et la zone de verdure, a été embelli. On retrouve désormais des rochers au centre, des nénuphars et autres plantes aquatiques.

Figure 10: Lac de la carte 1

La zone montagneuse

L'emplacement de cette zone a été décidé lors de la première soutenance : elle est située au sud de la zone urbaine. Nous avons commencé par implémenter du relief à l'aide de l'outil de modélisation du terrain en temps réel d'Unity.

Nous avons modélisé la montagne pour la deuxième soutenance : elle est visible sur pratiquement toute la carte et est un repère visuel pour les joueurs. De la neige s'y trouve au sommet. C'est donc un élément important dans l'environnement de la carte.

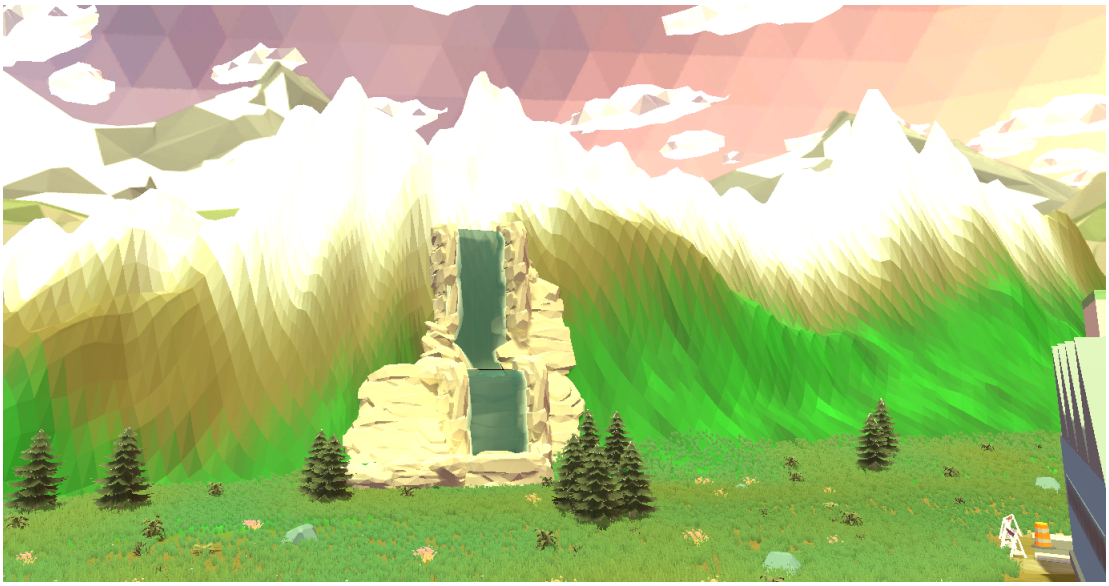


Figure 11: Zone montagneuse

Nous avons décidé d'ajouter une cascade et de modifier les rochers présents sur la montagne. La présence d'une cascade permet d'enrichir la zone montagneuse et d'ajouter un élément de décor.

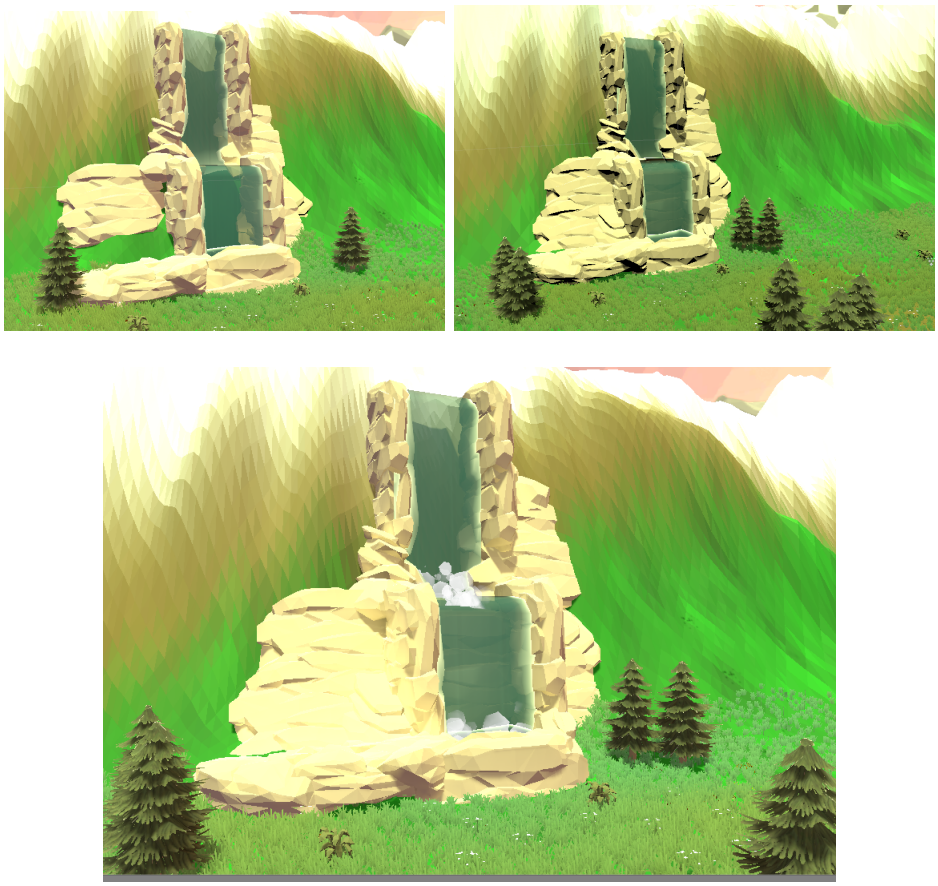
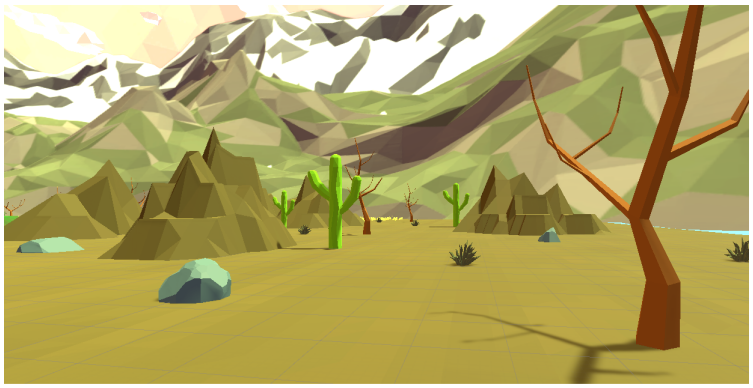


Figure 12: Évolution de la construction de la cascade

La cascade est constituée d'éléments de roches, d'eau ainsi que d'un système de particules permettant de représenter les éclaboussures d'eau.

La zone désertique

Cette zone a été créé lors de la deuxième soutenance afin de remplir les espaces vides de la carte. Cette zone est composée de dunes de roches, de petits rochers, de cactus, d'arbres morts et de plantes grasses. La présence de reliefs permet de cacher les potentiels ennemis.



Zone désertique de la carte 1

Il y a également des particules de sables afin de renforcer l'aspect désertique. Les zones de vides ont été aménagées : on retrouve des herbes ainsi que des rochers. Cela permet de combler le vide.

3.3.3 Carte 2

Présentation générale

La carte 2 a subi de nombreux changements au fil des soutenances, d'abord une carte avec différentes îles qui a finalement totalement changé de visage avec le choix d'une carte en zone montagneuse qui offre de grandes forêts de sapins.

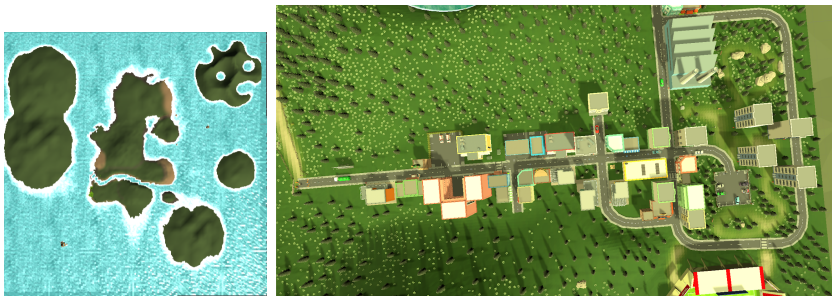


Figure 13: Évolution de la carte 2

Finalement cette carte 2 est plus grande que la première et propose de nombreux environnements :

- Le volcan
- La ville
- Les ruines
- Le lac
- Les collines
- les forêts de sapins

Le volcan

Il laisse s'échapper une épaisse fumée noire et dont l'accès n'est pas facile pour le joueur. Mais son ascension en vaut la peine car cela permettra au joueur de trouver un pouvoir spécial. Celui-ci est particulièrement avantageux par rapport aux autres pouvoirs que l'on peut trouver sur la carte et ce lieu constitue ainsi un endroit stratégique. Il a été implémenté dès la deuxième soutenance mais a subi des ajouts de particules depuis.

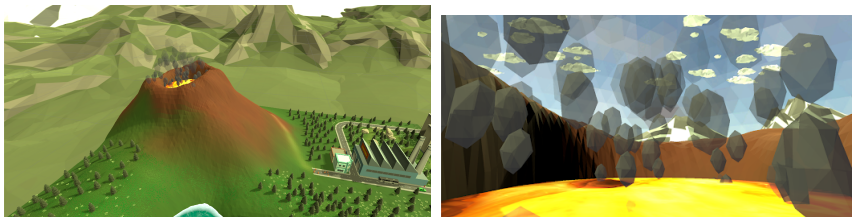


Figure 14: Le volcan de la carte 2

La ville

Elle est étalée dans la vallée, entre le lac et les collines diverses. Elle offre tout l'environnement de jeux avec les bâtiments et des parcs ajoutant un peu de verdure. Dans un souci de cohésion graphique avec la première carte, les routes, certains effets de particules et bâtiments sont semblables.

Figure 15: La ville de la carte 2

La ville s'est bien étoffé depuis la dernière soutenance. On y retrouve le même niveau de détails que sur la première carte avec des magasins, affiches publicitaires, restaurants,... mais également de petits détails comme les déchets qui sont éparpillés sur le sol à certains endroits ou encore un parking, des parcs qui accueillent de nombreuses espèces de papillons et quelques lucioles. Enfin des branchages et quelques gros rochers permettent de finir d'ajouter une importante quantité de détails.

Figure 16: Exemple de la carte 2

La ville était constituée de beaucoup moins de détails à la deuxième

soutenance qu'à l'heure actuelle. Par exemple, les déchets et le niveau de détail général de la carte, comme sur ce terrain vague, n'étaient pas présents précédemment.

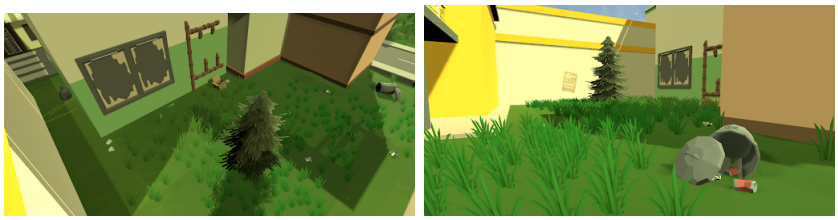


Figure 17: Exemple de déchets sur la carte 2

Les ruines

Elles se situent à proximité du parc et font penser à un site touristique avec les tables de pique-nique. Ces ruines sont celles d'une ancienne civilisation. Cela ajoute une touche de mystère à cette carte. Cette zone abrite tout comme le volcan un pouvoir spécial qui diffère de ceux que l'on peut trouver sur le reste de la carte. Il s'agit là encore d'un lieu tout à fait stratégique.

Figure 18: Ruines de la carte 2

Le lac

Il demeure au centre de la carte et il est composé d'un îlot central. La grande cascade se jette dans celui-ci et il borde à la fois les ruines et le volcan. Ce lac est un élément esthétique qui ajoute du réalisme à la carte. On y trouve aussi des plantes aquatiques, comme de petits nénuphars en fleurs.

Figure 19: Lac de la carte 2

Les collines et forêts de sapins

Elles représentent une grosse partie de la carte et est composée majoritairement de sapins, de fleurs, d'herbes de plusieurs types et de fougères. Il est possible de trouver des feux de camp ou encore d'éléments qui laisse penser que des promeneurs sont passé par là.

Par ailleurs, on retrouve des effets de particules dans les parcs (qui sont d'ailleurs les mêmes que dans la ville). Le tout se marie parfaitement avec l'environnement sauvage.

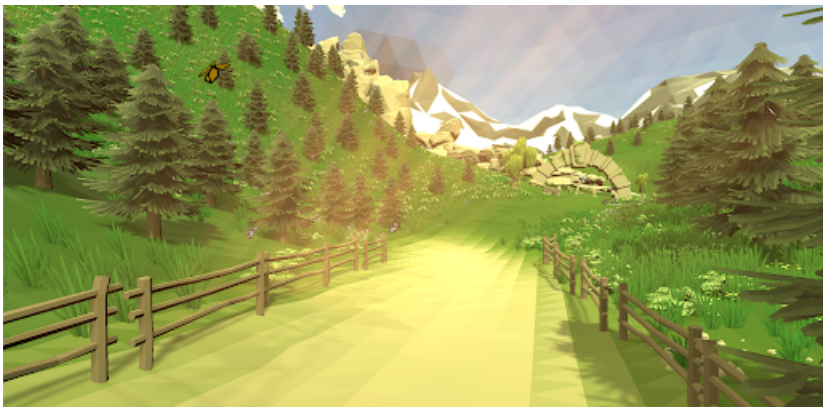


Figure 20: Effets de particules de la carte 2

La cascade

Il s'agit d'un objet massif du décor qui entièrement à fait à la main. Celle-ci est composée de plusieurs pierres et de quelques effets de particules qui ajoutent du réalisme. La cascade se situe au strict opposé du volcan et représente là encore un élément esthétique essentiel qui rend la zone du bord du lac tout à fait appréciable.

Figure 21: Cascade de la carte 2

3.4 Les bâtiments

Dans ce projet, les bâtiments ne sont pas seulement là pour la décoration, ils sont surtout des points stratégiques pour les équipes. En effet, la plupart des bâtiments sont destructibles. Chacun possède une vie et un niveau de feu.

3.4.1 La vie

Si un bâtiment n'a plus de vie, il s'écroule, donne des points à l'équipe des attaquants et fait apparaître des rescapés. Seuls les tirs de l'équipe des attaquants font perdre de la vie aux bâtiments. L'autre moyen de faire baisser la vie d'un bâtiment est d'augmenter son niveau de feu.

3.4.2 Le feu

Le Pyromane peut avec son lance-flamme augmenter le niveau de feu d'un bâtiment. Le Pompier quant à lui peut le diminuer. Les bâtiments ont 3 niveaux de feu. Le premier est null, c'est-à-dire pas de feu. Le deuxième fait apparaître de la fumée sur les bâtiments et leur fait perdre de la vie au fil du temps. Au dernier niveau, le bâtiment prend feu (des flammes se manifestent sur le toit) et perd plus de vie qu'au deuxième niveau. Il est donc important pour les pompiers de garder les bâtiments au plus bas niveau possible.

Figure 22: Exemple d'un bâtiment en feu de niveau 2 et 3

3.5 L'intelligence artificielle

L'intelligence artificielle (IA) est un élément important dans un jeu. Elle permet de le rendre vivant en y ajoutant de la vie et des interactions avec les joueurs. Dans notre cas nous avons 2 types d'IA. Elles ont toutes les deux des réactions différentes avec leurs environnements. La première est appelée *Basic*. Elle est utilisée pour peupler les villes. La deuxième apparaît lors de la destruction d'un bâtiment. On la nomme donc rescapé. Chaque IA possède différents états en fonction de l'interaction avec les joueurs.

3.5.1 Basic

Cette IA est présente dans le code des personnages non jouables (PNJ) qui peuplent la ville. À chaque début de partie, nous en faisons apparaître à plusieurs endroits dans la ville avec pour État "marche". Lorsqu'un PNJ Basic est à l'état de marche, il trouve une destination puis s'y rend en marchant. Une fois arrivé, il en choisit une autre et ainsi de suite... De ce fait les PNJ Basic se baladent dans la ville sans but précis. Pour faciliter la création du chemin entre le PNJ et sa destination, nous avons utilisé les *NavMeshAgent* d'Unity. Ils permettent de générer des chemins avec différents paramètres. Pour les utiliser nous avons besoin de configurer une carte de Navigation.

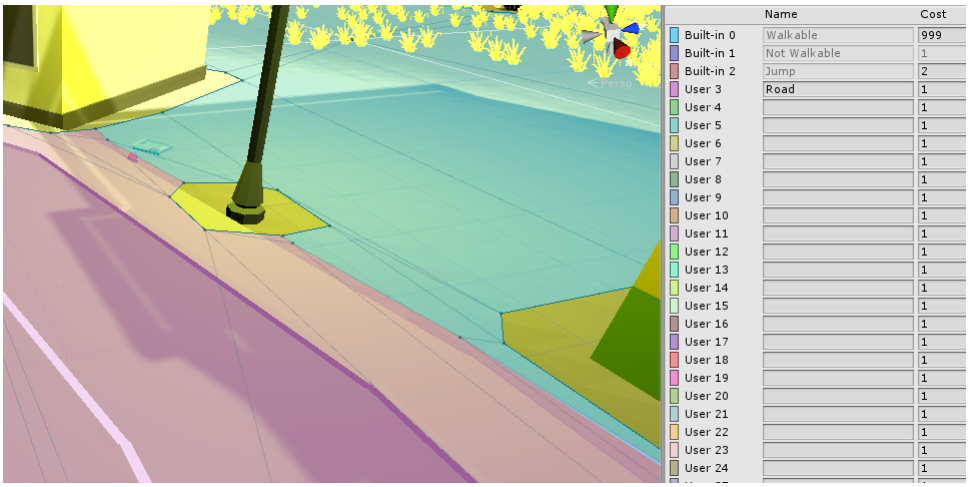


Figure 23: Exemple d'une carte de Navigation

Pour faire marcher les PNJ sur les routes, nous avons créé une nouvelle

aire appelée *Road* ici en rose. Comme on peut le voir, cette aire possède le paramètre "Cost" qui a pour valeur le chiffre 1 contrairement à l'aire *Walkable* qui elle possède le nombre 999. Lors de la création d'un chemin, la fonction va choisir le plus rapide et celui qui "coûte" le moins de points. Le fait que les routes sont à 1 point contrairement aux autres aires empruntables, les PNJ emprunteront les routes.

Lors d'un coup de feu, les PNJ Basic proches se mettent en état *peur*. En état de *peur*, l'IA *Basic* cherche la sortie la plus proche puis court vers celle-ci. Chaque IA a un seuil de panic aléatoire qui correspond au nombre de balle avant de passer en état de *panique*. Lorsqu'une IA panique, elle s'arrête et attend quelques secondes avant de revenir au stade de *peur*. Lorsqu'un PNJ atteint la sortie, il disparaît, donne des points à l'équipe des défenseurs et un nouveau PNJ Basic réapparaît dans la ville.

Un système d'animation (Animator) spécifique à été créé. De base, les PNJ jouent une animation de marche. Un booléen passe à *True* lorsque les PNJ basic se mettent en *peur*. Une animation de course vers l'avant s'active. Un autre booléen gère l'état de *panique* et déclenche une animation adéquate.

3.5.2 Rescapé

Lors de la destruction d'un bâtiment, un PNJ Rescapé apparaît demandant de l'aide. Seuls les joueurs de classe Pompier ou Médecin peuvent les soigner. Quand ils se font soigner, ils donnent des points aux défenseurs. Après avoir été soigné, ils prennent le même état de *peur* que les Basic, c'est-à-dire qu'ils cherchent à rejoindre la sortie la plus proche.

3.5.3 Modèle

Afin de diversifier les apparences des PNJ sans avoir à créer de nombreux modèles, nous avons créé un script qui change aléatoirement les couleurs des vêtements. Pour faire cela nous avons créé une liste de couleurs (ici colors) contenant toutes les couleurs possibles d'un vêtement. Puis on applique à chaque matériaux correspondant à un vêtement d'un PNJ une couleur au hasard de la liste. L'utilisation de cette méthode permet de garder un contrôle sur les couleurs portées contrairement à une génération de couleurs complètement aléatoire.

```
private static List<Color> colors = new List<Color>
{
    Color.black,
    Color.cyan,
    Color.gray,
    Color.white,
    Color.yellow,
    new Color(0.8f,1f,0.7568f),
    new Color(0.6117f,0.6392f,1f),
    new Color(0.949f,0.6705f,1f),
    new Color(1f,0.5529f,0.5529f)
};

2 références
public static void SetColor(GameObject character)
{
    Renderer ren = character.GetComponent<Renderer>();
    float randUp = colors.Count - 0.1f;
    ren.materials[2].color = colors[Mathf.FloorToInt(Random.Range(0, randUp))];
    ren.materials[4].color = colors[Mathf.FloorToInt(Random.Range(0, randUp))];
    ren.materials[6].color = colors[Mathf.FloorToInt(Random.Range(0, randUp))];
    ren.materials[7].color = colors[Mathf.FloorToInt(Random.Range(0, randUp))];
    ren.materials[8].color = colors[Mathf.FloorToInt(Random.Range(0, randUp))];
}
```

Figure 24: Code de la génération des PNJ

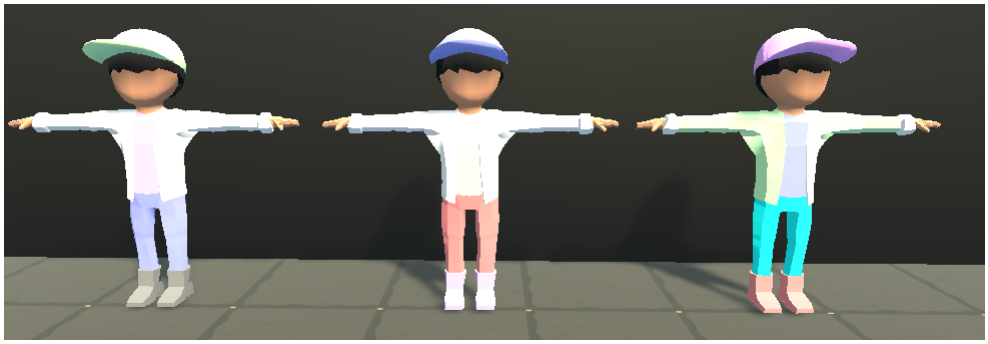


Figure 25: Exemple de génération de modèles PNJ

3.6 Les personnages

Durant la première soutenance, nous avons réalisé l'ensemble des personnages des Sauveteurs, à savoir :

- Policier

Figure 26: Modèle du Policier

- Pompier

Figure 27: Modèle du Pompier

- Médecin

Figure 28: Modèle du Médecin

Pour la deuxième soutenance, nous avons réalisé l'intégralité des personnages des Agresseurs.

- Mercenaire

Figure 29: Modèle du Mercenaire

- Pyromane

Figure 30: Modèle du Pyromane

- Drogueur

Figure 31: Modèle du Drogueur

Enfin, pour la dernière soutenance, nous avons peaufiné les précédents modèles et nous avons conçu le personnage non-jouable.

Figure 32: Modèle des PNJ

3.7 Les Animations

3.7.1 Introduction

L'animation est un élément important d'un jeu vidéo. En effet, c'est ce qui permet de donner du mouvement, de la vie et de l'activité sur les personnages, objets et autres. Sans animation, un jeu vidéo perd tout intérêt. C'est donc une partie très importante de notre projet.

Figure 33: Exemple d'un model en T-Pose

Sans animation, nos personnages seraient dans une position que l'on appelle "T pose". Ils se déplaceraient alors comme cela.

L'implémentation d'un système d'animation a donc été l'une des premières choses que nous avons faites après avoir implémenté notre premier personnage. Lors de la première soutenance, nous avons donc implémenté les animations de déplacements élémentaires des personnages.

3.7.2 Le composant Animator

Le composant Animator est un système permettant de gérer les animations. Il permet la création d'un algorithme sous forme de diagramme. Les animations peuvent être reliées entre elles à l'aide de liens. Aussi, des conditions peuvent y être appliqués.

Lors de la première soutenance, le composant Animator a été créé et implémenté. Il a depuis évolué au cours des soutenances : de nouvelles animations ont été ajoutées et de nouvelles conditions ont été créées.

Voici la partie principale du composant Animator lors de la troisième soutenance :

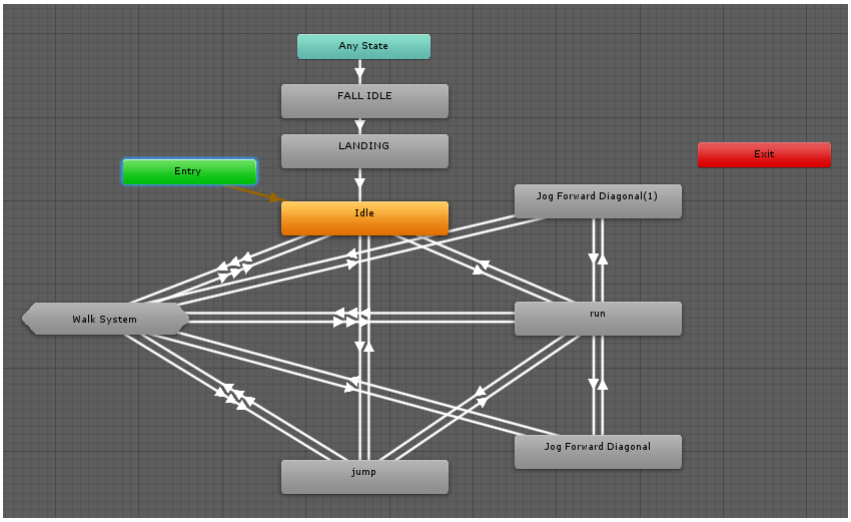


Figure 34: Animator des personnages

Entry représente l'entrée du composant. **Any State** est une case libre d'accès quelque soit l'animation jouée. La case **Walk System** est appelé un "**StateMachine**". Cela permet de créer un sous-algorithme. Ici, toute les animations liées au système de marche (Walk System), sont implémentées via le sous-algorithme "**StateMachine**".

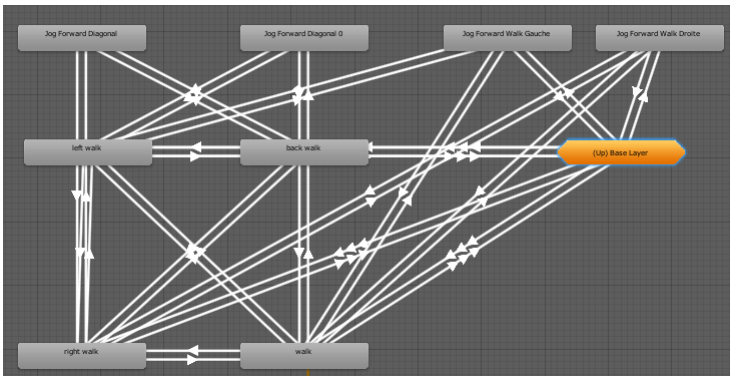


Figure 35: Walk System

Voici à quoi ressemble le Walk System. On remarque qu'il y a beaucoup de lien d'où l'intérêt de créer un StateMachine. La case **Base Layer** représente la partie principale de l'algorithme (celui étudié précédemment).

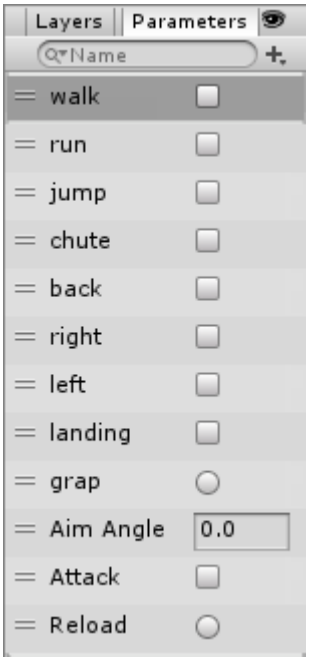


Figure 36: Sous-algorithme Walk System

Lors de la première soutenance, un système basé sur la logique booléenne a été mis en place. Les booleéens utilisé permettent d'établir des conditions dans les liens de l'animator afin de pouvoir passer d'une animation à une autre. Ces booléens sont modifiés par un script en fonction des touches enclenchées. Il y au total 9 booléens. Lors de la deuxième soutenance, les animations spécifiques aux personnages ont été implémentées. De nouveaux booléens ainsi que des trigger et float ont été créés. Un trigger permet de déclencher une animation et de la jouer (en s'assurant qu'elle ne sera jouée qu'une seule fois).

Voici un exemple de conditions sur un lien entre deux animations. Lorsque ces conditions sont vérifiées, l'Animator peut alors jouer l'animation se trouvant au bout du lien.

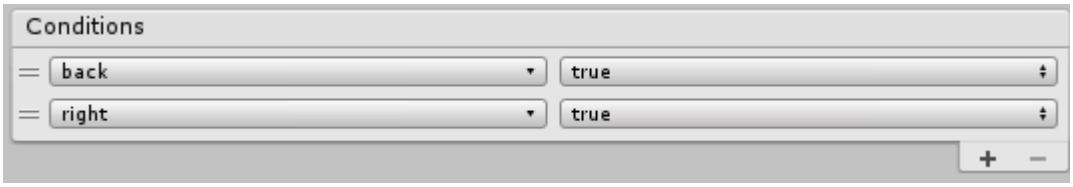


Figure 37: Condition de transition

3.7.3 Le système d'animation principal

Ce système d'animation comprend toutes les animations nécessaires aux bons déplacements des personnages. On peut décomposer le **système d'animation principal** en deux parties : **les animations élémentaires** et **les animations avancées**.

Lors de la première soutenance, nous avons implémenté les animations élémentaires du système d'animation principal.

Les animations élémentaires sont :

1. La marche avant
2. la marche arrière
3. La marche à droite
4. La marche à gauche
5. Le saut
6. La course avant
7. La réception au sol
8. La position de repos

Figure 38: 1

Figure 39: 2

Figure 40: 3

Figure 41: 4

Figure 42: 5

Figure 43: 6

Figure 44: 7

Toutes ces animations permettent aux personnages de se déplacer sur la carte correctement. Le déplacement des personnages était donc notre objectif pour la première soutenance.

Nous avons ensuite voulu améliorer et finaliser le système d'animation principal en implémentant des animations avancées et en résolvant différents bugs.

Les bugs rencontrés correspondaient à des problèmes liés aux avatars des personnages ainsi qu'aux types des animations. Sur Unity, un avatar permet de "représenter" un personnage. Ici, les personnages étant identiques, nous nous sommes donc basé sur un unique avatar pour toutes les animations. Seuls quelques détails les différencient, comme les vêtements et les accessoires, mais cela n'a pas d'impact quand on s'intéresse à l'animation. Toutes les animations implémentées ont été converties en humanoïdes : cela signifie qu'elles peuvent être appliquées à tous les personnages de type humain. Les bugs d'animations ont été résolu grâce à cela.

De plus, des animations avancées ont été implémentées lors de la deuxième soutenance. Elles permettent d'augmenter les capacités de déplacements des personnages.

Il y a :

- La diagonale avant gauche en marchant

- La diagonale avant droite en marchant
- La diagonale avant gauche en courant
- La diagonale avant droite en courant
- La diagonale arrière gauche en marchant
- La diagonale arrière droite en marchant

Figure 45: Diagonale droite / Diagonale gauche

Il y a en réalité seulement deux animations : la diagonale droite et gauche. Un avantage du composant animator est que les vitesses des animations peuvent être modifiées.

En modifiant la vitesse, on peut jouer l'animation en courant, en marchant, et en marche arrière (avec une vitesse négative). Donc seulement deux animations suffisent.

Le système d'animation principal ayant été fini pour la deuxième soutenance, il n'a subi que très peu de modifications pour la troisième. Seules les transitions entre les animations ont été modifiées et ajustées afin de les rendre plus naturelles.

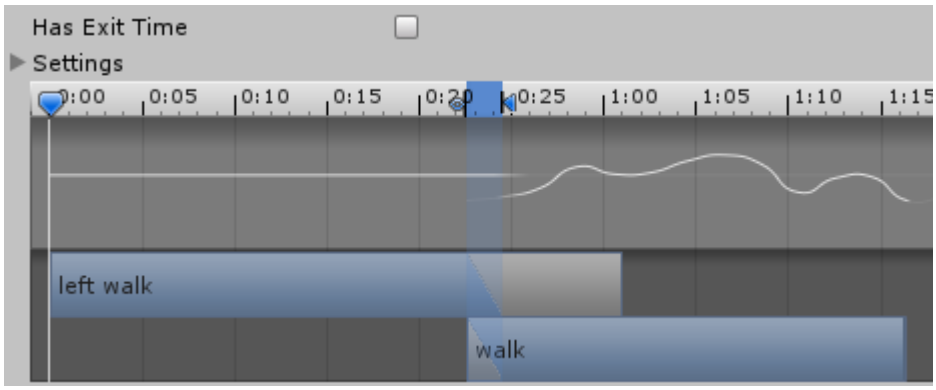


Figure 46: Exemple de transition

3.7.4 Le système d'animation spécifique

La deuxième soutenance a amené la création d'un système d'animation spécifique. En effet, l'implémentation des différentes classes de personnages nécessite certaines animations spécifiques.

Une problématique a rapidement émergé : comment faire pour jouer les animations de déplacements et les animations spécifiques en même temps sur un personnage ?

Unity propose un outil qui permet de répondre à cette question.

Le système de Layers

Le système de Layers est un système permettant de superposer plusieurs animations sur un même personnages. Il permet par exemple de jouer une animation sur le bas du corps et une autre sur le haut du corps.

Pour cela, il faut d'abord créer des layers Mask. Ce sont des masques qui permettent de définir au composant Animator (le composant gérant les animations) quelle zone du corps du personnage doit être activée.

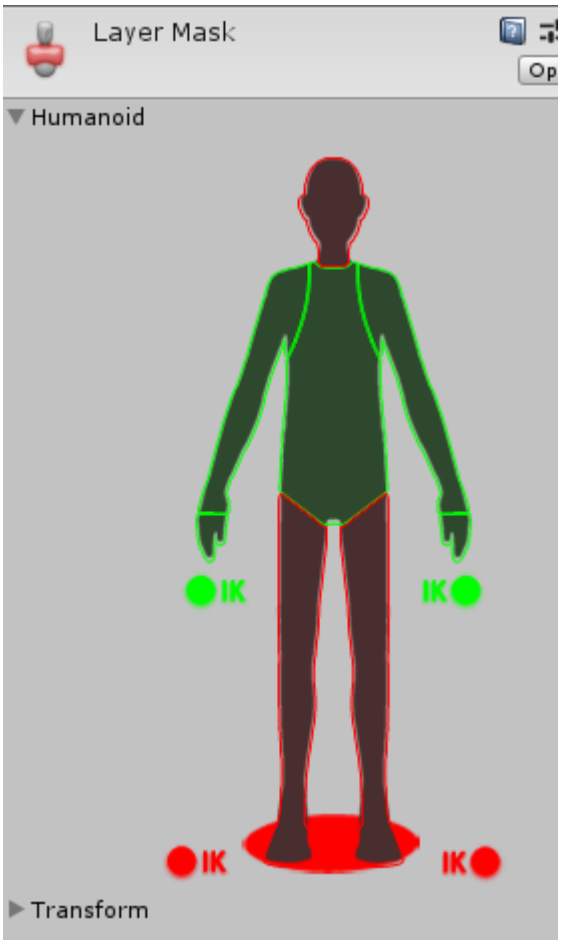


Figure 47: Exemple d'un layer mask

Par exemple, pour les animations spécifiques, on souhaite les appliquer uniquement sur le haut du corps. Ici sur le composant layer Mask les zones vertes correspondent aux zones d'applications de l'animation et les zones rouges aux zones de non-application de l'animation.

Le système de Layers peut donc être implémenté. Pour rappel, il permet de superposer des animations et donc de jouer les animations de déplacements sur la base du corps et les animations spécifiques sur le haut du corps. On aurait pu implémenter par exemple une animation de marche avant et de visée, mais il y aurait alors eu beaucoup trop d'animations différentes.

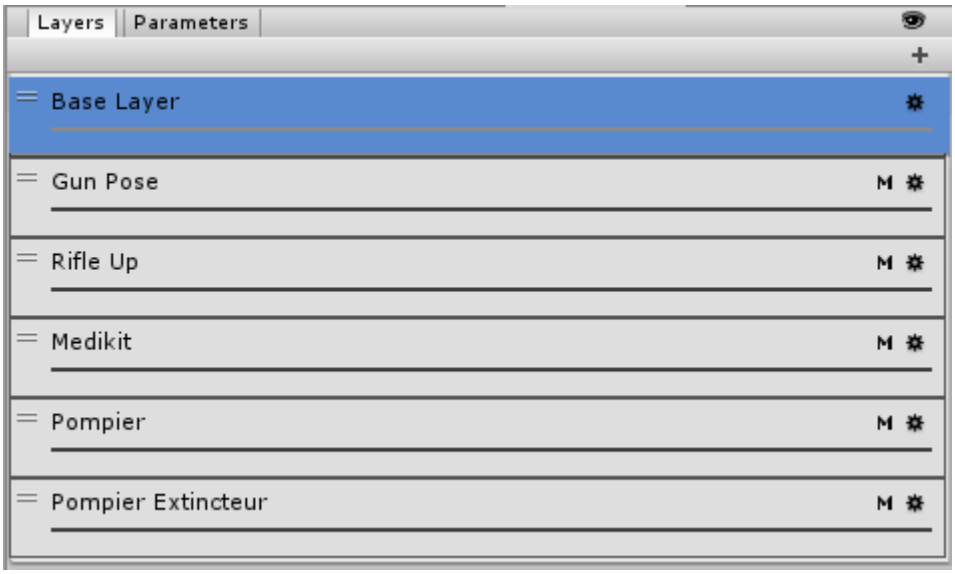


Figure 48: Exemple de layers d'un Animator

Le système de Layers se présente comme cela : on y retrouve une liste de layers qui permettent d'appliquer différentes animations aux personnages. Par exemple, le layer "Pompier" représente les animations spécifiques du pompier.

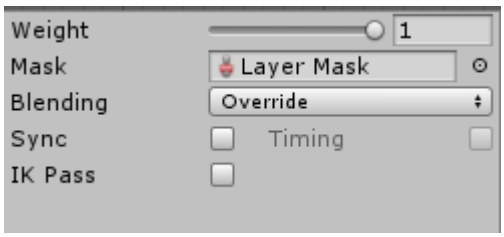


Figure 49: Exemple de layers parameter

Chaque layer possède un onglet paramètre dans lequel on peut configurer le Layer Mask ainsi que le "poids" du layer.

Le poids ("**Weight**") permet d'ajuster la valeur d'activation du **layer**. Cette valeur permet d'activer ou de désactiver certain **layer** en fonction du

personnage sélectionné et des animations devant être jouées. Elle est gérée grâce à des scripts.

Les animations

Voici les animations spécifiques des personnages :

- La visée (haute et basse) : mercenaire, policier et pyroman (1)
- La prise et le rangement de l'arme : tous sauf drogueur et médecin (2)
- Le rechargement : mercenaire, policier, pyroman et pompier (3)
- Le combat rapproché : pompier (4)
- La distribution de soin : médecin et drogueur (5)

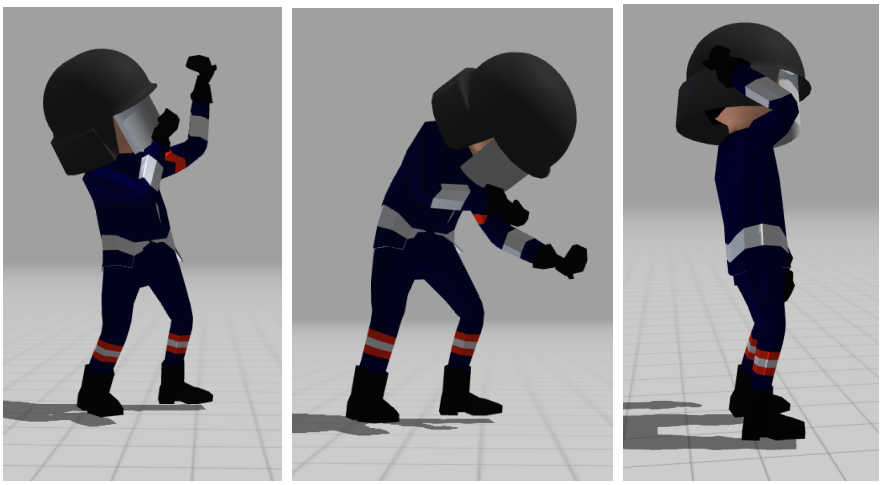


Figure 50: 1/2

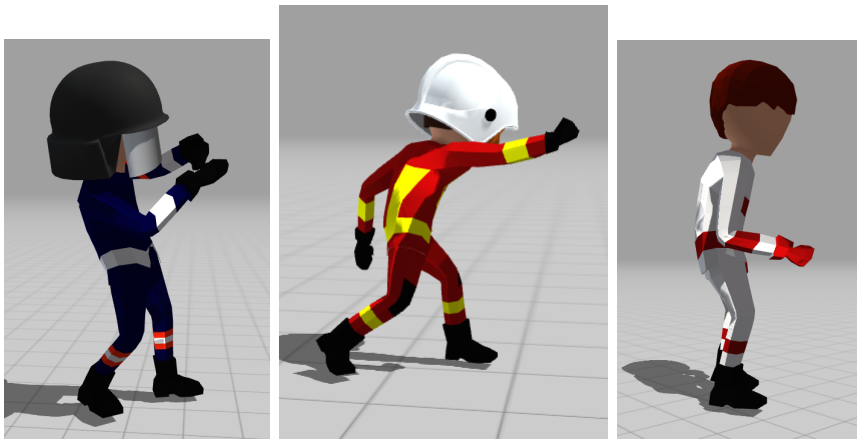


Figure 51: 3/4/5

Lors de la deuxième soutenance, nous avions aussi développé le système de visée. Il permet aux personnages de pouvoir ajuster l'angle de la visée avec une arme en fonction du mouvement de la souris.

Il faut donc pour cela ajuster l'animation de visée, afin que l'on puisse viser au milieu, en haut ou en bas. Unity propose un système appelé Blend Tree qui est disponible dans le composant **Animator**. Ce système a pour particularité de pouvoir "mixer" ou "mélanger" plusieurs animations.

Cela est très intéressant dans notre cas car nous pouvons alors mixer deux animations : une de visée vers le haut et une de visée vers le bas.

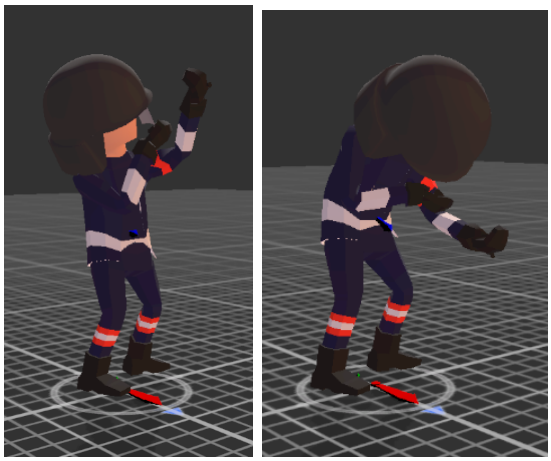
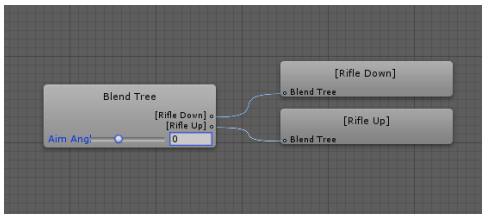


Figure 52: Visée haut / bas



Le composant **Blend Tree** prend en paramètre ces deux animations (ici Rifle Down et Rifle Up) et peut alors les mélanger.

Figure 53: Blend tree

Par exemple, s'il prend tout autant en compte la première animation que la deuxième, une animation de visée vers le milieu sera joué. En revanche, s'il donne plus d'importance à celle de la visée vers le bas que celle de visée vers le haut, l'animation résultante sera une visée plus orientée vers le bas.

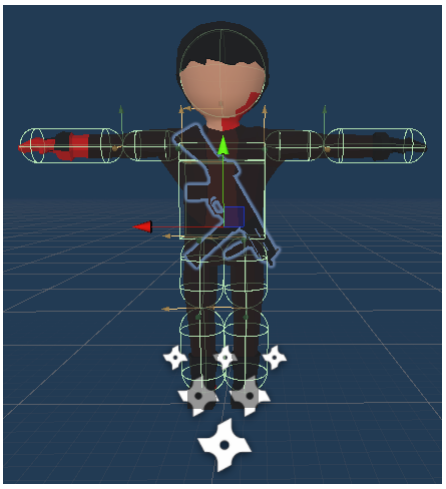
Une variable nommée Aim Angle s'occupe de gérer ce taux d'utilisation d'une animation par rapport à une autre. Un script s'occupe de récupérer en temps réel la valeur de la souris sur l'axe vertical afin de modifier Aim Angle. L'animation de visée peut donc s'ajuster en fonction de l'endroit où vise le joueur.

3.7.5 La mort RagDoll

La mort des joueurs a déjà été implémenté lors de la première soutenance. Le joueur disparaissait du jeu et réapparaissait à un point donné. Nous

avons donc décidé d'améliorer le système de mort des personnages pour cette dernière soutenance. Plusieurs possibilités existent : la plus simple est d'implémenter une animation qui se joue quand le personnage meurt.

Nous avons décidé d'aller plus loin et d'implémenter un système de **Ragdoll**. Le ragdoll est une technique d'animation procédurale qui assimile un corps à un ensemble d'éléments solide pouvant s'articuler. Concrètement, le système **Ragdoll** d'Unity crée pour chaque partie du corps un **Collider** (un élément permettant de créer des collisions). Par exemple, il a deux **Colliders** par bras et par jambes, un pour la tête, etc...



Voilà ce que donne un personnage quand un système Ragdoll est appliqué.

Figure 54: Exemple Ragdoll

Tous les personnages du jeu ont donc un **Ragdoll**. Lorsqu'un personnage est en vie, tous ses composants **Rigidbody** (les composants gérants toute la physique du personnage) donnent le contrôle du personnage aux animations. Les **Colliders** du **Ragdoll** sont désactivés.

Lors de la mort d'un personnage, les composants **Rigidbody** désactivent le contrôle du corps aux animations. Les collisions et articulations du **Ragdoll** affectent désormais le personnage. Cela permet donc d'avoir un résultat d'un corps s'écroulant sur le sol très réaliste. Mais le principal intérêt est de ne jamais avoir la même animation lors de la mort d'un personnage.

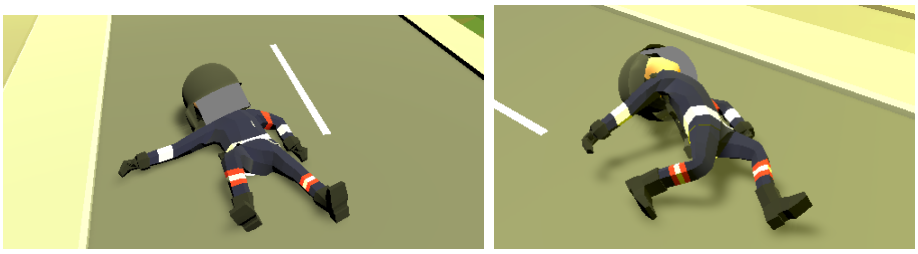


Figure 55: Exemple de mort avec le système **Ragdoll**

3.8 Les Pouvoirs Spéciaux

Les pouvoirs spéciaux ont grandement évolué depuis la dernière soutenance avec l'arrivée d'un système d'inventaire. Jusqu'à maintenant, il n'y avait qu'un seul type de pouvoir spécial et qui se déclenchait immédiatement, dès qu'on marchait dessus. En effet chaque joueur dispose maintenant de 4 espaces dans son inventaire pour amasser des pouvoirs spéciaux durant la partie. Ils peuvent les déclencher à tout moment de la partie, à la pression d'une simple touche.

Les pouvoir spéciaux sont de quatre natures :

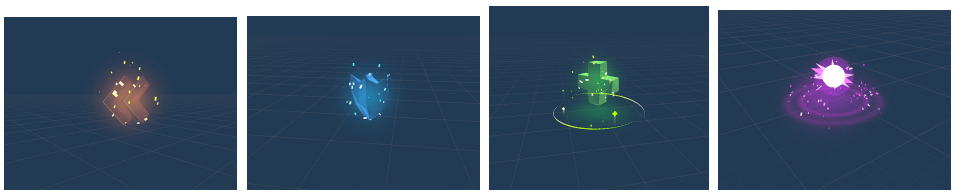


Figure 56: Pouvoir spéciaux

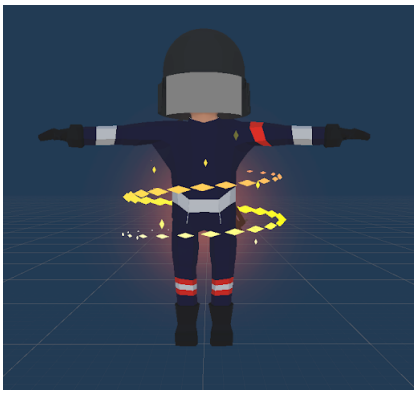


Figure 57: Effet de rapidité

La rapidité :

Le *"Speed"* est le pouvoir de la rapidité procurant au joueur une rapidité de déplacement partielle qui dure 5 secondes et qui lui permettra par exemple de fuir ou encore

d'engager l'affrontement avec une rapidité éclair.

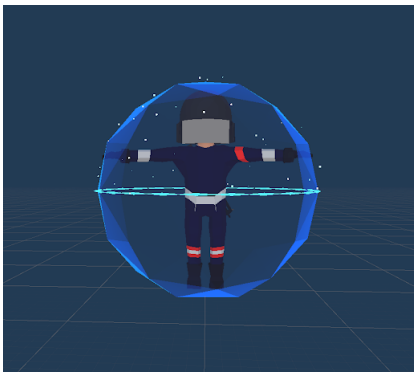


Figure 58: Effet de bouclier

Le bouclier :

Le *"Shield"* est le pouvoir du bouclier, il procure une invincibilité partielle de 5 secondes au joueur lui permettant ainsi d'engager le combat ou de fuir sans perdre le moindre point de vie.

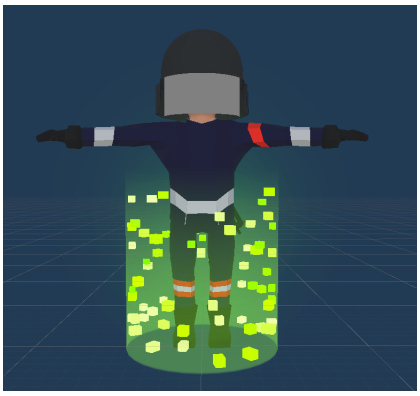


Figure 59: Effet de vie

La vie :

Le *"Heal"* est le pouvoir de la guérison, il met à 100% les points de vie du joueur.

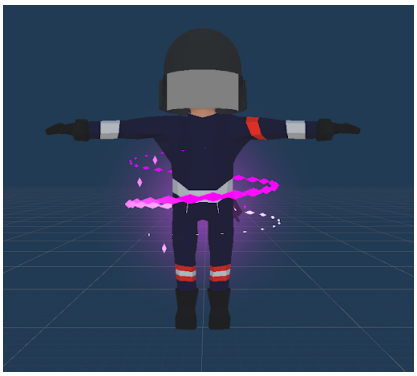


Figure 60: Effet de saut

La hauteur des sauts :

Le *"Jump"* est le pouvoir du saut. Il offre au joueur la possibilité de sauter plus haut et donc d'accéder aux toits des bâtiments où il aura l'avantage sur ses ennemis.

Tous les pouvoirs spéciaux sont répartis sur les deux cartes, mais certains demeurent plus rares que d'autres. Les joueurs auront ainsi un intérêt à se battre pour être les premiers à tous les prendre. Les parties de ESU demandent maintenant un peu de stratégie grâce à ce système. Il faut réussir à utiliser ses pouvoirs spéciaux correctement et dans la bonne situation, ce qui peut demander de la technique. De plus, cela oblige les joueurs à se déplacer et à explorer la carte. Pour finir chaque pouvoir spécial provoque un effet graphique sur l'écran du joueur qui l'utilise.

3.9 Les Menus et HUD's

3.9.1 Introduction

La partie menus et HUD's (Affichage Tête Haute en français) concerne tous les éléments graphiques autres que le jeu : les informations sur les armes, la vie du personnage, le réglage du volume, le choix des cartes, etc...

Nous avons décidé d'utiliser un style graphique commun à tous les menus et HUD's afin qu'ils s'harmonisent correctement. Nous avons choisi une palette de couleurs provenant du site "**flatuicolors.com**". La particularité de cette palette est qu'elle propose des couleurs mates produisant peu d'éclat. Les couleurs ressortent bien, ce qui implique que les menus et HUD's ont une bien meilleure visibilité pour le joueur.

Figure 61: Palette de couleurs

3.9.2 La barre de vie

Lors de la première soutenance, le système de la barre de vie avait été créé mais n'avait pas été implémenté. On retrouve dans la barre de vie un cadre, un élément de texte et un fond dont le remplissage peut être adapté. L'image du fond (en bleu) est assimilé à un composant **Slider** qui peut modifier le remplissage d'une image par exemple.

Figure 62: Barre de vie

Concrètement, un script s'occupe de récupérer la valeur de la vie du joueur et la compare à la valeur temporaire qu'il stocke. Si les deux valeurs sont différentes, il modifie la valeur du **Slider** (donc aussi le taux de remplissage de la barre de vie). Si la valeur du slider est à 50, l'image sera remplie de moitié.

Nous avons utilisé la méthode **Mathf.Lerp** qui permet d'afficher un remplissage de la barre de vie de façon graphiquement plus naturelle. Cette méthode utilise l'interpolation linéaire. Elle prend en argument trois paramètres : la valeur actuelle, la valeur souhaitée et enfin la valeur d'interpolation (comprise entre 0 et 1). La nouvelle valeur va progressivement passer de la valeur actuelle à la valeur souhaitée, proportionnellement à la valeur de l'interpolation.

Plusieurs design graphiques ont été pensés afin de réaliser la barre de vie :

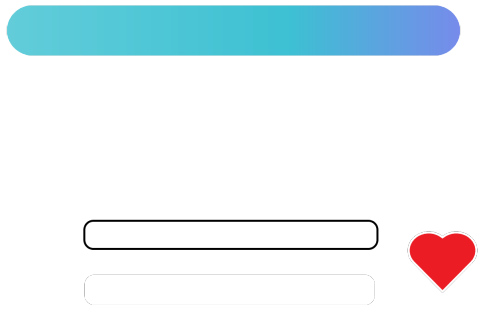


Figure 63: Autre design de la barre de vie

Le design final a été adopté lors de la deuxième soutenance. Un texte présent à côté de la barre de vie affiche en pourcentage la quantité de vie restante du joueur. La valeur de ce texte est modifiée dans le même script que celui de la barre de vie.

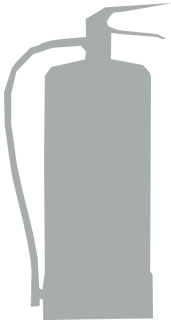
3.9.3 Les informations sur les armes

Lors de la deuxième soutenance, nous avons implémenté le système de munitions pour les armes ainsi que certains HUD's. Ils ont été réalisé à l'aide du logiciel "**Photoshop**".

Figure 64: Arme du policier et du mercenaire

Figure 65: Hache du pompier

Figure 66: Kit de soin du drogueur et du médecin



Extincteur du pompier

Figure 67: Lance-flammes du pyromane

L'arme du policier et du mercenaire possède une barre de munitions. Le même procédé que pour la barre de vie est utilisé. Cette fois, la barre est circulaire et un chiffre indique le nombre de munitions restantes. Un script s'occupe de récupérer le nombre de munitions restantes et de modifier le taux de la barre ainsi que le texte en fonction de cela.

Figure 68: Barre de munition

Le cercle au centre joue une animation lorsque le joueur recharge en appuyant sur la touche **R**.

Figure 69: Barre de capacité

Le lance-flammes et l'extincteur possèdent une barre indiquant la capacité restante. Elle est verticale et utilise le même procédé que la barre de vie et de munitions.

3.9.4 Les menus

Les menus constituent une partie cruciale pour notre projet : ils permettent aux joueurs de paramétrer et d'accéder au jeu, ainsi que d'effectuer des réglages et d'avoir diverses informations.

Le menu principal est composé de quatre boutons : **Jouer**, **Options**, **Crédits** et **Quitter**. Chacun de ces boutons possède une animation lorsque l'on clique dessous ou quand on y passe son curseur. Un script attaché au menu permet de gérer les actions possibles. Par exemple, lorsque l'on actionne le bouton **Quitter**, le script lance une fonction qui ferme le jeu. Les boutons **Options** et **Crédits** désactivent le HUD du menu et activent leur propre HUD.

Figure 70: Menu Principal

Idem pour le bouton **Play** qui active le HUD permettant de sélectionner le mode de jeu.

Figure 71: Menu mode de jeu

Lorsqu'on clique sur le bouton **Rejoindre**, le script gérant le menu charge une nouvelle scène en fonction de la carte sélectionnée.

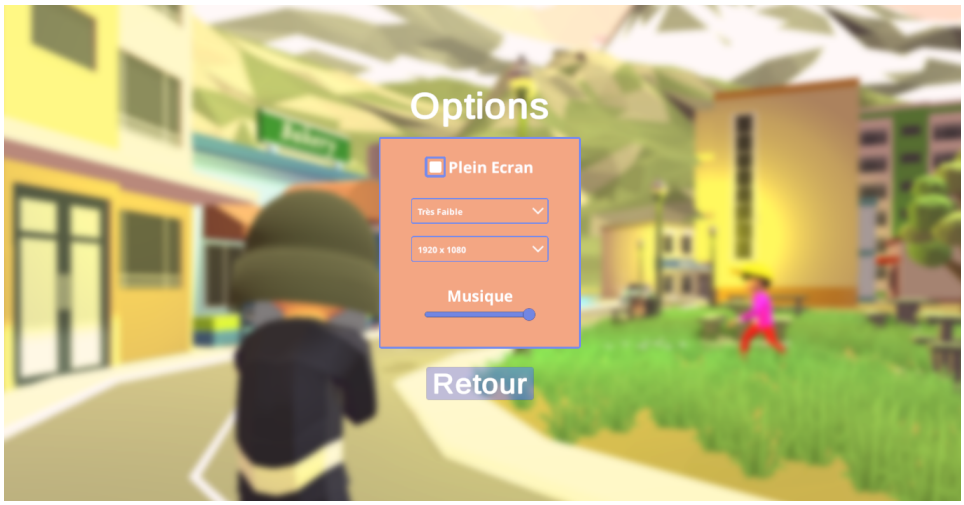


Figure 72: Menu options

Le menu options propose différents réglages comme l'activation du mode plein écran, le choix de la qualité graphique, le choix de la résolution d'affichage ainsi que le volume de la musique de ce même menu. Le tout est bien évidemment géré par un script.

3.9.5 Les pouvoirs spéciaux

Le HUD s'est adapté à l'arrivé du système d'inventaire des pouvoirs spéciaux. En effet on retrouve l'interface ci-dessous qui représente l'inventaire constitué de quatre emplacements.

Figure 73: Inventaire des pouvoirs

Les touches "**W**", "**X**", "**V**" et "**C**" correspondent aux touches qui permettent d'utiliser les différents emplacements de l'inventaire. Ainsi la pression d'une touche consomme immédiatement le pouvoir spécial qui se trouve dans l'emplacement correspondant.

3.10 Les sons

Les sons font partie intégrante d'un jeu vidéo, ils permettent d'immerger les joueurs dans les différents environnements qu'il occupera et dans les différentes actions qu'il va entreprendre.

Les sons d'un jeu vidéo comportent la musique mais aussi les bruits. Les bruits permettent d'indiquer au joueur l'état de ce qui l'entoure, comme un personnage qui tire, qui cours, qui frappe un mur ou même un autre joueur. Sans ces sons, le joueur ressentirait une moins bonne immersion dans le jeu, ce qui implique un divertissement de piètre qualité.

Dans ce projet, nous avons décidé de créer de toutes pièces les musiques et les bruits. Cette contrainte nous a certes pris du temps en ce qui concerne la création des compositions sonores mais elle nous a aussi permis de véritablement choisir quels effets et émotions nous voulions faire ressentir au joueur.

3.10.1 Généralités sur la production

Afin de composer l'intégralité des sons du jeu, nous avons utilisé tout du long le séquenceur **Ableton Live** ainsi que la suite de VST **Komplete de chez Native Instruments**.

Ableton Live

Komplete
Ultimate Edition

Pour cette dernière, elle nous a permis de gagner beaucoup de temps sur la création de sons émotionnels mais elle nous a aussi servi de base harmonique pour notre musique du menu principal.

3.10.2 Les différentes productions

Les sons ont été développés en même temps que le jeu, cela implique qu'ils ont été créés en rapport à des besoins particuliers. Ainsi, à la fin de notre jeu, nous avons composé des sons pour 3 problématiques différentes.

1. Pendant une partie
2. Les bruits
3. Le menu principal

Les deux premières problématiques ont trouvé solution dès la 2ème soutenance. En ce qui concerne la troisième, elle fut résolue pour la dernière.

3.10.3 Écoutez les sons de la partie !

Afin de pouvoir vous rendre compte du travail fourni par nos soins, nous vous invitons à écouter les musiques implémentées dans notre jeu en scannant le code QR ci-contre.

3.10.4 Les musiques pendant une partie

Dès le début de sa création, nous voulions rendre notre jeu dynamique mais surtout adaptable à un changement de rythme. De ce fait, nous avons décidé de créer 1 musique par phase de jeu. Notre jeu en comportant 3, nous avons donc **3 musiques pendant une partie**. Une partie étant d'une durée de 10 min, nous avons décidé de la découper ainsi :

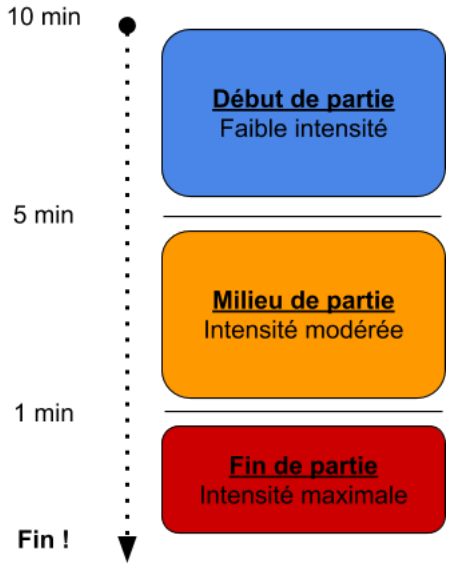


Figure 74: Graphique représentant les variations d'intensités musicales en fonction du temps restant

Pour le *début de la partie* (quand le chronomètre est entre 10 et 5 min), l'intensité doit être faible car la partie vient tout juste de commencer et les actions entreprises par les joueurs n'ont pas encore eu le temps de changer véritablement le sort des équipes alliées et adverses.

Pour le *milieu de partie* (quand le chronomètre est entre 5 et 1 min), les joueurs ont déjà eu plus de temps pour faire gagner ou échouer une équipe. C'est pourquoi l'intensité doit être modérée, c'est-à-dire plus intense que le début de partie sans pour autant inquiéter le joueur sur un éventuel événement trop fervent.

En ce qui concerne la *fin de partie* (pour la dernière minute), c'est à ce moment que le joueur doit se sentir poussé à finir rapidement ses actions.

C'est pourquoi l'intensité doit être maximale.

3.10.5 Musique d'intensité faible

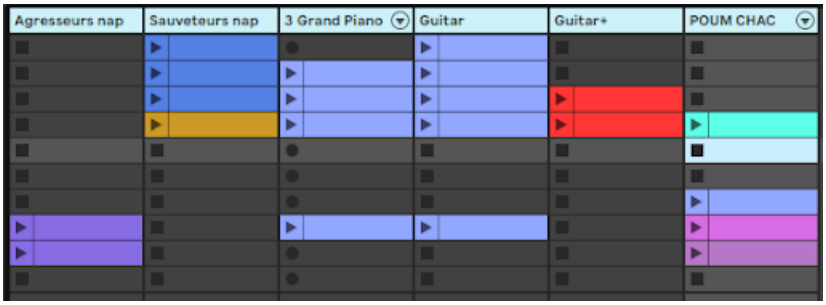


Figure 75: Instruments de la musique d'intensité faible

Base rythmique

Afin de rendre notre première intensité suffisamment faible tout en la gardant intéressante à l'écoute, nous avons choisi de mettre **une base de guitare** avec peu d'accords. Ce choix permet de rythmer la partie sans la rendre inintéressante.

Suite à cela vient s'ajouter dès la deuxième boucle un **accompagnement au piano**. Ce dernier nous permet d'accentuer un peu plus la guitare et de lui donner plus d'importance.

Lors de la montée, nous ajoutons une **seconde guitare**. C'est celle-ci qui fera office d'instrument principal dans notre son : elle est un peu plus intense que la première mais limite tout de même le son à une intensité relativement basse.

Pour respecter le critère d'intensité, nous avons décidé de ne mettre aucune batterie, excepté au refrain, car cela aurait rendu l'intensité beaucoup trop forte.

Effets

Pour cette première musique, nous n'avons utilisé qu'un seul effet et ce de façon très restreinte : **l'égaliseur paramétrique** (ou Equalizer/EQ en anglais)

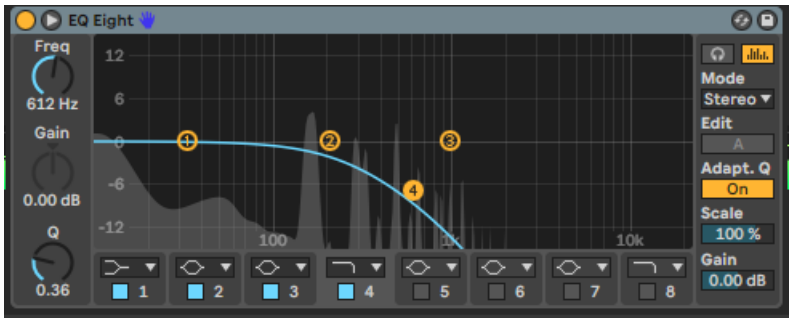


Figure 76: Égaliseur paramétrique filtrant les aigus

La seule utilité que nous lui avons trouvé lors de la composition de la musique était la réduction des fréquences hautes (aigus) au niveau du piano et une transition plus fluide lors du passage du pont vers le refrain pour la deuxième guitare.

3.10.6 Musique d'intensité moyenne

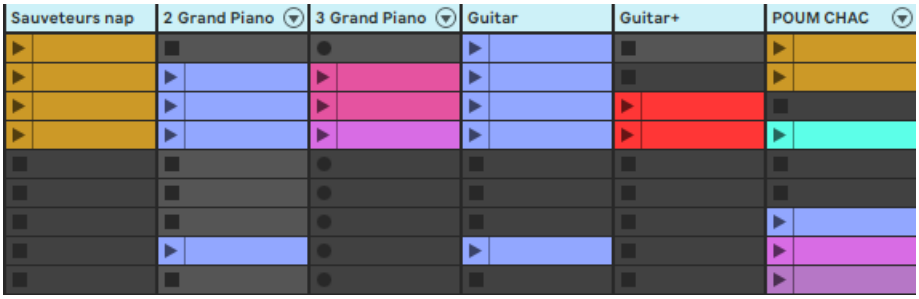


Figure 77: Instruments

Changements par rapport à la version d'intensité faible

Le rythme entre la première musique et celle-ci sont restés les mêmes, cependant nous avons ajouté un peu de détail pour que l'augmentation d'intensité se fasse ressentir.

De ce fait, nous avons commencé par mettre de la **batterie à chaque temps**, ce qui permet de rendre cette composition plus rythmée d'une part, mais surtout de faire ressentir une intensité plus forte.

Aussi, nous avons ajouté un **second piano** qui sert d'accompagnement léger pour l'introduction, la montée et le pont mais d'instrument principal (au même titre que la guitare 2) pour le refrain.

En ce qui concerne l'accompagnement, les accords sont souvent constitués de plusieurs notes très proches les unes des autres afin de rendre cet instrument beaucoup plus dynamique que tous les autres.

Pour le refrain, nous avons utilisé un **générateur de notes aléatoire**. Ce générateur prend une note fondamentale (ici, nous avons choisi la fondamentale du morceau : le B_3 [où le Si_3]) et prend tous les quarts de temps une note à quelques octaves aux dessus ou en dessous.

Effets

Figure 78: Réverbération et égaliseur paramétrique

Ici, pour cette séquence, nous avons utilisé 2 effets : la *réverbération* et l'*égaliseur paramétrique*.

Pour l'égaliseur paramétrique, nous n'avons pas fait de changement par rapport à la précédente version, il s'occupe toujours de filtrer certaines fréquences des instruments trop stridents comme le piano.

En ce qui concerne la réverbération, elle est peu audible mais elle est disposée sur la nappe de départ afin de rendre la spatialisation du son plus évidente.

3.10.7 Musique d'intensité élevée

Changements par rapport à la version d'intensité moyenne

Dans cette version, nous voulions une intensité très élevée. De ce fait, nous sommes passé d'un rythme de **125 BPM** à **140 BPM**.

De plus, l'instrument principal (la guitare) a changé de puissance, elle est à présent beaucoup plus battante que les 2 précédentes versions.

Suite à cela, pour être en rapport avec le thème du jeu, nous avons ajouté une corne 2-tons afin d'y augmenter encore plus la tension.

Finalement, pour toujours plus souligner la virulence de cette composition, nous avons inséré des claquements de doigts (snap) et des claquements de mains (clap).

3.10.8 Bruits du jeu

Afin de rendre les interactions des joueurs plus réalistes, nous avons implémenté une multitude de bruits qu'un joueur peut entendre lors d'une partie :

Figure 79: Fichiers audio correspondants aux bruits

Comme vous pouvez le voir, nous avons réalisé des bruits pour chaque action qu'un joueur peut entreprendre.

De plus, les bruits de pas sont adaptés à l'allure que le joueur peut avoir : Un joueur se déplaçant très lentement sera accompagné du bruit **Marche50bpm**, un joueur courant sans bonus se déplacera avec le bruit **Course110bpm** et un joueur se déplaçant avec un bonus de vitesse aura à ses côtés le bruits de course **Course130bpm**.

Bruits de tir

Afin de garder un effet cartoon pour notre jeu, nous n'avons pas utilisé de plugin permettant de générer des détonations réalistes. Nous avons fait beaucoup plus simples : nous avons utilisé une **caisse claire** (snare) pour simuler un effet de tir de paintball.

Cependant, les bruits étaient un peu trop aigus. De ce fait, nous avons utilisé un **égaliseur paramétrique** pour filtrer les fréquences trop hautes (ainsi que certaines basses parasites).

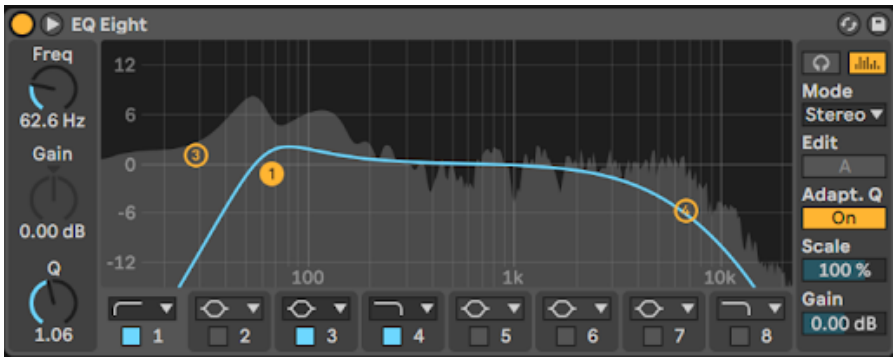


Figure 80: Égaliseur paramétrique filtrant les aigus et basses parasites

En ce qui concerne le tir en rafale, nous avons tout simplement pris le même fichier audio que nous avons dupliqué avec un tempo élevé.

Bruits de lance flamme

Le lance-flamme a fait l'objet d'une plus mûre réflexion. En effet, il est plus dur de reproduire le son d'une combustion et d'une projection en même temps. C'est pourquoi nous avons choisi d'utiliser un instrument permettant de simuler un son qui est à la fois **périodique** et avec une **attaque lente** (qui correspondent respectivement à un effet de combustion et de projection)

Cet instrument, c'est la **Wavetable**.

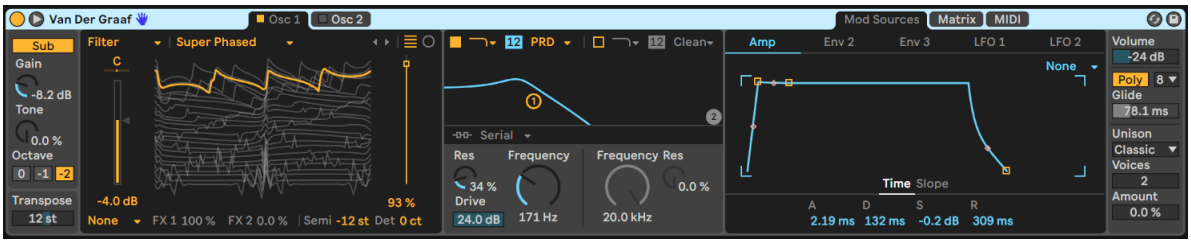


Figure 81: Wavetable

Ainsi, grâce au son produit par cet instrument, nous pouvons simuler un effet tout aussi cartoon que les autres bruits. L'effet d'attaque lente a pu être réalisé grâce à un premier égaliseur paramétrique (réglé en mode :

filtre passe-bas) qui varie des fréquences basses vers les fréquences hautes. L'effet périodique quant à lui a été réalisé grâce au fonctionnement même d'une wavetable, à savoir la variation d'une onde sonore vers une autre à une certaine fréquence.

Pour en finir avec ce bruit, nous avons ajouté un dernier égaliseur paramétrique qui vient filtrer les aigus de la sortie sonore de cet instrument.

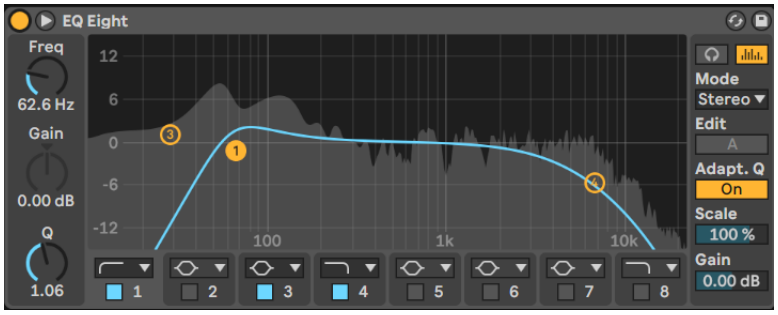


Figure 82: Filtre des parasites sonores

Bruit de marche/course

Les bruits de marche/course ont été les plus simples à implémenter. Pour se faire, nous avons tout simplement utilisé un **tom moyen** de batterie sur lequel nous n'avons gardé qu'une plage de fréquence assez basse, entre 100Hz et 800Hz.

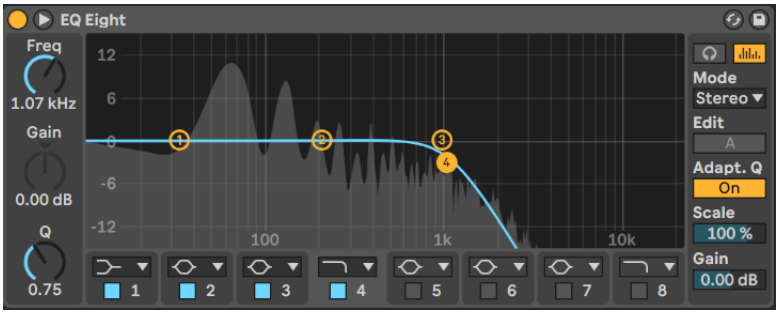


Figure 83: Filtre passe-bas

Ensuite, afin de simuler une marche, nous avons placé à la suite des notes

MIDI liés à cet instrument, et ce, **toutes les croches** (autrement dit, sur les temps et les contre-temps).

Figure 84: Tom moyen de batterie en notes MIDI

En ce qui concerne la différentiation entre la marche et la course, elle réside juste dans le tempo donné au croches. Pour les marches, nous avons définit arbitrairement **50 bpm** et **70 bpm**, pour la course, nous avons définit **100 bpm**, **110 bpm** et **130 bpm**.

Bruit de coup

Pour le bruit de coup, nous avons utilisé un **sample de tir de golf** dans *Ableton Live* que nous avons un peu filtré avec un égaliseur paramétrique.

3.11 Musique du menu principal

La musique du menu principal fut la composition sur laquelle nous avons passé le plus de temps. De l'écriture, en passant par une multitude d'effets pour aller jusqu'au mixage, cette piste a demandé plus de 3 mois de travail.

Nous allons ici vous décrire les différentes étapes de construction de ce son.

3.11.1 Les nappes

Afin de baser notre son sur quelque chose de solide, nous avons choisi d'utiliser une nappe de fond en Si que l'on joue à l'aide d'une **Wavetable** pour donner une impression d'aléatoire et d'un synthé de la suite de VST de Native Instrument : **Kontakt**, pour donner un aspect de précision.

Figure 85: Nappes de fond

De plus, nous voulions un son plus doux que ce que l'instrument nous fournissait. De ce fait, nous avons supprimé les aigus et arrondi un peu les basses.

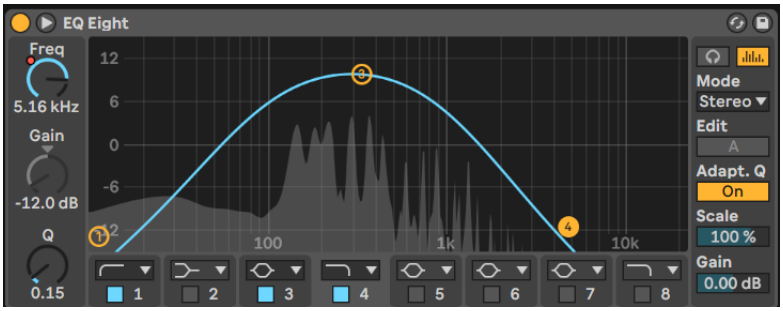


Figure 86: Filtre de fréquences basses et hautes

Dès la fin de l'introduction, pour effectuer une transition vers la montée, nous avons disposé ce que l'on appelle un **bruit blanc**.

3.11.2 La batterie

Afin de rendre le son plus moderne, nous avons choisi d'orienter le rythme la batterie vers de la **trap**. Ainsi, vous pouvez entendre des Charleston (Hihat) à intervalles très réguliers pour pouvoir donner cet effet.

De plus, si vous écoutez les sons avec des enceintes de bonnes qualités, vous pourrez entendre le travail que nous avons réalisé sur les **basses** qui accentuent encore plus le style trap.

3.11.3 Instrument principal

Juste après l'introduction intervient la montée. Dès celle-ci, l'instrument principal se fait déjà ressentir : il s'agit d'un synthé produit par un synthétiseur qui porte le nom du titre du morceau : **Massive**. Cet instrument est très strident ce qui permet de faire un son puissant et les nombreux effets de transitions fourni avec le rende beaucoup plus adaptable. On peut prendre comme exemple le *Glide*, qui consiste à faire un changement de note de façon logarithmique (qui nous a d'ailleurs servi d'effet émotionnel lors du refrain).

3.11.4 Variation d'intensité de l'instrument principal

Afin de rendre tous les passages cohérents avec une montée d'intensité, nous avons fait des jeux d'égaliseurs paramétriques avec l'instrument principal. Ainsi, au début de la montée, le son semble un peu étouffé, puis il récupère ses fréquences perdues au fur et à mesure que l'on avance dans le temps, pour enfin retrouver toute son énergie sur le refrain.

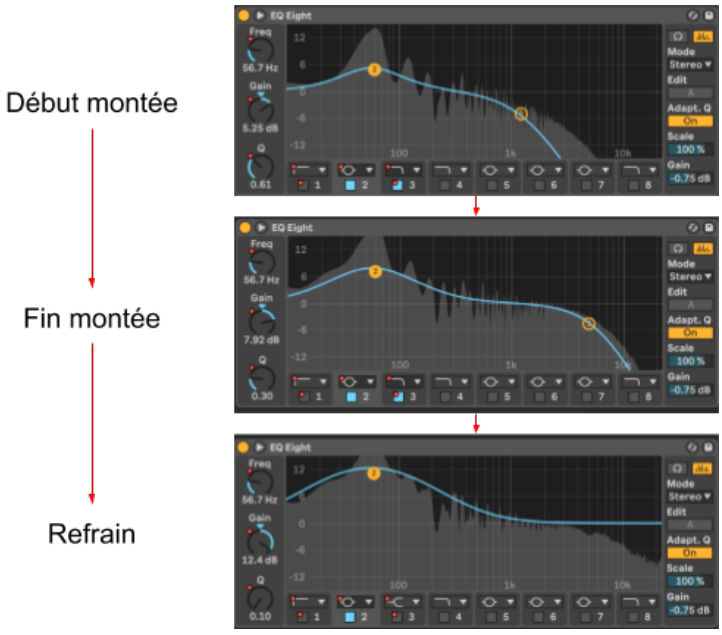


Figure 87: Variations des paramètres des égaliseurs paramétriques en fonction de l'avancement de la piste audio

3.12 Mixage

Le mixage consiste en la modification des sorties audio des instruments, cela permettant de produire une véritable synergie entre tous les instruments qui jouent en même temps.

Dès la fin de la première composition, les instruments semblaient étouffés et certains en masquaient d'autres de par leur force supérieure. C'est pour cela que nous avons dû réaliser un gros travail de mixage pour pouvoir rendre ce son véritablement puissant.

3.12.1 Séparation

Pour pouvoir faire sonner les instruments que nous avons créé avec le plus d'énergie possible, le maître mot est concession !

Par exemple, pour pouvoir faire trembler les enceintes avec des basses, nous avons dû faire une séparation des **pseudo-basses** ($\sim 250Hz$) et des **véritables basses** ($< 50Hz$). Cette séparation permet principalement aux

enceintes de piètre qualité (telles que les enceintes de téléphone ou d'ordinateur portable) de pouvoir entendre les pseudo-basses sans pour autant limiter les autres enceintes de meilleurs qualité qui pourront elles de leurs côtés faire ressentir et faire trembler les pièces dans lesquelles elles se trouvent grâce aux véritables basses.

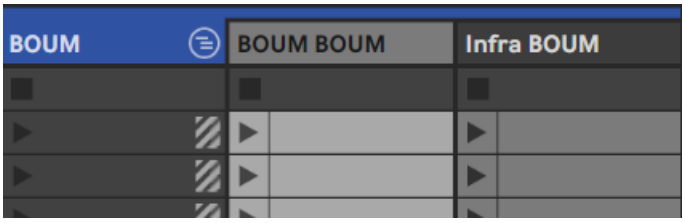


Figure 88: Séparation des pseudo-basses et des véritables basses

3.12.2 Égalisation

L'intérêt de cette partie est la diminution voire la suppression des superpositions audio (et donc des masquages) que les instruments peuvent faire les uns sur les autres. Ainsi, dans cette partie, on cherchera à augmenter les fréquences que nous souhaitons entendre d'un instrument et à en diminuer sur la même fréquence les spectres audio des autres. L'égalisation est donc un jeu de concession où l'on essaie de trouver le parfait équilibre entre diminution sur une fréquence et augmentation sur l'autre (et inversement).

3.12.3 Compression

Lorsque des sons sortent d'un instrument, surtout s'ils sont accompagnés de beaucoup d'effets, il est très probable que de grosses variations d'intensité sonore apparaissent. Le but de la compression, c'est justement de normaliser l'intensité de leur spectre audio.

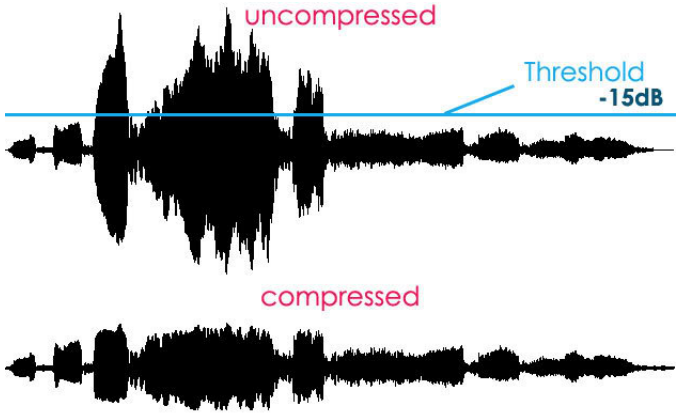


Figure 89: En haut : un son non compressé — En bas : le même son compressé

Lors de la compression des pistes, nous avons pu augmenter l'intensité de beaucoup d'instruments et donc l'intensité globale du morceau.

De plus, nous avons réalisé sur un instrument d'accompagnement (lui aussi réalisé à l'aide de Massive) une **compression** dite **parallèle**. Elle permet de synchroniser l'amplitude d'un son avec un rythme particulier (ici, le rythme de la basse).

Cette compression nous a non seulement permis de limiter la superposition avec les autres instruments, mais elle nous a aussi permis d'obtenir un effet audio très intéressant.

3.13 La scène de fin de partie

3.13.1 Introduction

Cette scène a pour but de célébrer la fin d'une partie. Tous les joueurs sont déplacés dans cette carte et jouent différentes animations en fonction d'une victoire ou d'une défaite de leur équipe. Un tableau des scores est également disponible. Cette scène a été implémentée lors de la 3ème soutenance.

3.13.2 Construction de la scène

L'outil **ProBuilder**, qui est une extension d'Unity, permet de concevoir et modéliser des formes 3D. Il a été utilisé afin de concevoir le bâtiment de cette

scène.

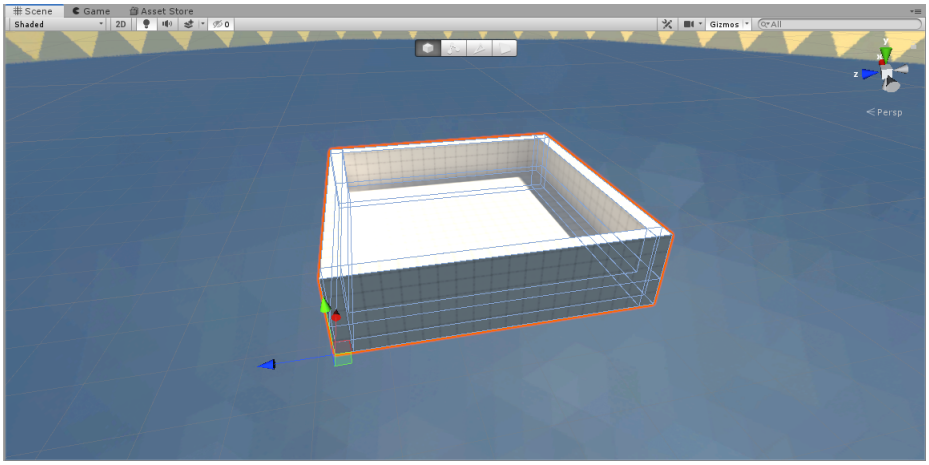


Figure 90: Exemple de construction réalisée avec ProBuilder

Nous avons d'abord conçu un cube qui a été étiré afin de former le sol. Puis, à l'aide de l'outil de construction de murs de **ProBuilder**, nous avons étiré les bords du sol afin d'ériger 4 murs. Puis, une estrade a été modélisée afin d'accueillir les gagnants d'une partie.

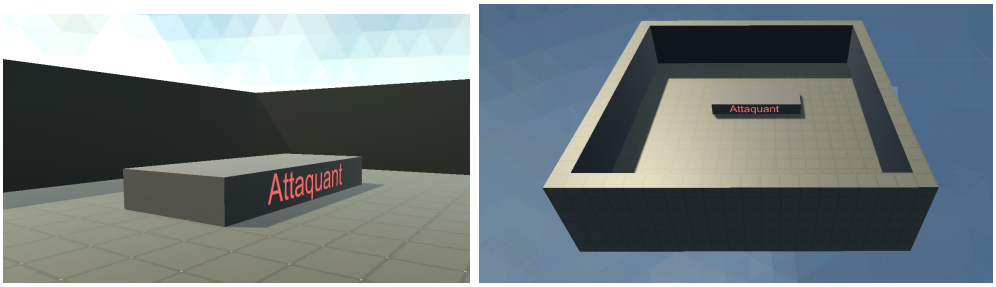


Figure 91: Salle de fin de partie

Des textures ont été appliquées aux murs, sol et estrades. Une **SkyBox** (hexaèdre permettant de créer un ciel) a également été implémentée.

3.13.3 Animations

Un système d'animation a été conçu à l'aide du composant **Animator** présent dans Unity (le composant gérant les animations). On retrouve 3 animations de défaite et 2 animations de victoire. Les animations se jouent en boucle. Un booléen déclenche les animations de défaite ou de victoire dans l'**Animator** en fonction des victoires ou défaites des joueurs.

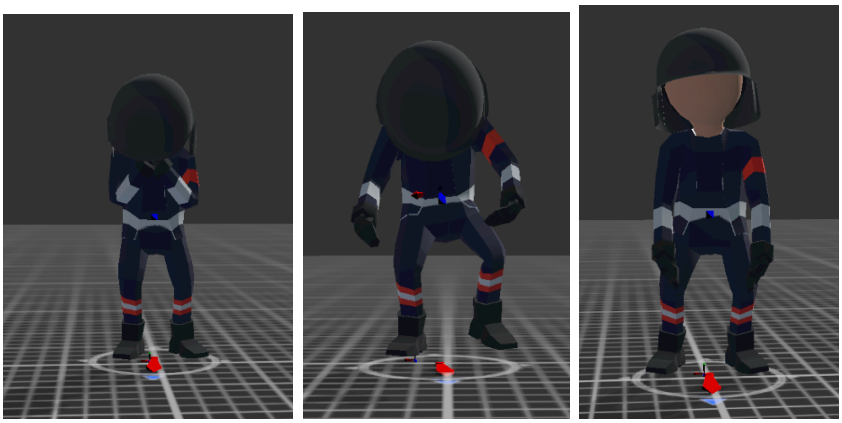


Figure 92: Animations de défaite

Figure 93: Animations de victoire

3.13.4 Fin de partie

Quand le compte à rebours arrive à 0, un fondu noir apparaît sur l'écran. Pendant cette transition un script va récupérer la liste des joueurs, leurs scores et le score global de la partie. Puis une fois ces données récupérées, le script va préparer la scène de fin de partie. Il va définir quelle équipe a gagné. En fonction de ce paramètre, il va placer les joueurs de l'équipe gagnante sur le podium, les afficher sur le podium et afficher le score global en haut. On a aussi accès au tableau des scores dans cette scène.

Figure 94: Exemple de fin de partie : les défenseurs gagnent !

4 Récit de réalisation

4.1 Nathan

Dans ce projet, j'ai majoritairement composé les sons du jeu qui ont permis de rendre ce dernier plus immersif. J'ai aussi modélisé les personnages et leurs armes durant les 5 mois de projet. Durant la conception des musiques, j'ai souvent eu l'inspiration tardive mais les résultats ont fini par être concluant. De plus, des soucis d'ombrages ont fait leur apparition sur Unity, mais ils ont bien heureusement été résolu ! Ce projet m'a permis de perfectionner

mes techniques dans des domaines plus artistiques qu'informatique et j'en suis vraiment satisfait.

4.2 Vincent

Dans ce projet, je me suis principalement occupé du système d'animation, du développement de la carte 1 ainsi que d'autres domaines comme la plupart des HUD's. J'ai pris plaisir à implémenter les animations des joueurs et à modéliser la carte 1. Je me suis parfois heurté à des problèmes qui m'ont bloqué, comme par exemple des bugs d'animation. J'ai toutefois réussi à résoudre ces problèmes et continué le développement du jeu. Je suis très satisfait de mon travail et j'en sors avec beaucoup de connaissances. Cela me donne envie de continuer à développer des jeux et repousser mon imagination.

4.3 Maxence

Pour ma part, je me suis occupé du multijoueur, du gameplay pour ainsi rendre le tout fonctionnel. Ce projet m'a donné des notions sur la création d'un jeu de tir en multijoueur. De la création d'un personnage jouable en troisième personne à la gestion d'une partie en ligne. Pour chaque fonctionnalité, j'ai dû structurer le programme pour le rendre compatible en multijoueur, ce qui a donné lieu à la majorité de mes problèmes. Ce projet m'a aussi appris à utiliser Unity. Il m'a donné envie d'approfondir mes connaissances en programmation de jeu vidéo et de continuer à en créer.

4.4 Louis

De mon coté je me suis occupé de la carte 2 ainsi que des pouvoirs spéciaux et également de l'ajout d'assets. J'ai par ailleurs réalisé le HUD des pouvoirs spéciaux et développé des éléments de gameplay, comme les boucliers de points de réapparitions ou encore les zones de mort de la carte 2. J'ai beaucoup aimé implémenter les pouvoirs spéciaux et le système d'inventaire. C'est d'ailleurs paradoxalement l'élément avec lequel j'ai rencontré le plus de problèmes, par exemple des problèmes de déplacement du joueur. J'ai cependant su passer outre et je suis fier du résultat. Ce projet m'a surtout appris à travailler en groupe durant un projet entier, du début à la fin et à utiliser Unity. Cela a été très formateur pour moi et en conclusion je pense

que je me replongerai sûrement dans le milieu du jeux vidéo dès que j'en aurai l'occasion.

5 Conclusion

Le groupe Deltaplane est heureux d'annoncer son projet E.S.U. (Emergency Service Unit) enfin terminé, après plusieurs mois de travail. Nous avons tous appris beaucoup de choses, que ce soit dans la création de musique, la modélisation de cartes, le code C#, le logiciel Unity et bien d'autres encore.

Malgré des problèmes rencontrés lors du développement, nous avons su trouver des solutions adaptées afin de les résoudre. Nous avons pris plaisir à développer de nouvelles fonctionnalités et à repousser notre imagination afin de créer un univers autour de notre projet.

Certes, notre projet peut toujours être amélioré et comporte certains défauts, mais tous les objectifs principaux définis dans notre cahier des charges sont remplis. Nous sommes même allés plus loin dans certains domaines notamment avec le système de capacités ou encore la scène de fin de partie.

Tous les membres du groupe Deltaplane vous remercient pour l'attention portée à notre projet.

Table des figures

Liste des tableaux