GAFAM EATERS

Rapport de projet

Jeu vidéo - Senso No Michi

Antoine Mosimann — E1
Louis Rodet — E1
Mathis Galliano — C1
Abel Chartier — E1

Promo 2027

15 Décembre 2022 — 31 mai 2023

Table des matières

1 Introduction

Vous trouverez dans ce rapport de projet les nouvelles créations, les créations, dessins, changements ainsi que les améliorations techniques du jeu vidéo de l'équipe GAFAM Eaters.

Ce jeu vidéo appelé Senso No Michi est un jeu de rythme sous la forme d'un RPG en trois dimensions.

Nous vous exposerons son contenu, les tenants et aboutissants du projet ainsi que son organisation et sa répartition au sein du groupe.

2 Le projet ZenTianJi

2.1 Le cahier des charges

Un cahier des charges a été écrit en décembre dernier afin définir le scénario du jeu, la répartition des tâches et l'avancement par soutenance.

Il avait été remodifié entre son premier rendu et la première soutenance, mais il n'a pas rechangé depuis.

Nous avons pris la liberté de modifier quelques éléments de ce cahier des charges qui ne convenait pas à chacun mais ceux-ci n'impacteront pas les grandes lignes du projet initial.

2.2 Scénario et gameplay

Le scénario reste le même que défini dans le cahier des charges. Dans le scénario, des guerriers venant de trois grandes familles (La famille des mages, des guerrier et des assassins) ayant pour but de "s'unir et de suivre les pas de leurs ancêtres pour rétablir un monde de paix parmi la terreur semée par les envahisseurs".

Il est possible de jouer au jeu en solo, mais surtout en multijoueur ou la coopération est très importante car elle permet d'unir les forces des trois grandes familles.

Les joueurs jouent, attaquent, se déplacent en rythme avec la musique ce qui leur procurent des bonus pour être plus forts.

2.3 La répartition des tâches

Nous rappelons la répartition des tâches à laquelle nous nous sommes appliqués à suivre pour pouvoir travailler plus vite et mieux.

	Louis Rodet	Mathis Galliano	Antoine Mosimann	Abel Chartier
Chef de projet		Délégué	Suppléant	
Modelage 3D et animation		Délégué		Suppléant
Musique et sons		Suppléant		Délégué
Gameplay			Suppléant	Délégué
Designs 2D	Délégué	Suppléant		
Levels Designs	Suppléant	Délégué		
Menus	Délégué	Suppléant		
Site Internet	Délégué			Suppléant
Intelligence Artificielle		Délégué	Suppléant	
Multijoueur	Suppléant		Délégué	

2.4 Avancement par soutenances

Voilà le détail de l'avancement par soutenance. Nous rappelons l'avancement auquel nous étions à la fin de la première soutenance et à la fin de la seconde soutenance. Aujourd'hui nous avons estimé avoir fait tous ce que nous voulions faire en début d'année, même si l'on pourrait continuer d'ajouter des nouvelles fonctionnalités à l'infini.

	Soutenance 1	Soutenance 2	Soutenance 3
Modelage 3D et animation	60%	90%	100%
Musique et sons	60%	80%	100%
Gameplay	65%	75%	100%
Levels Designs	60%	75%	100%
Menus	65%	90%	100%
Site Internet	45%	90%	100%
Intelligence Artificielle	30%	50%	100%
Multijoueur	70%	90%	100%

3 Réalisation techique

Dans cette partie, nous aborderons la manière donc l'équipe des GAFAM Eaters a su réaliser techniquement son projet de jeu vidéo.

3.1 Les Menus

Commençons par le commencement : Dans cette sous partie nous expliquerons et détaillerons comment nous avons agencé nos menus, et ce qu'ils permettent de faire.

3.1.1 Le menu d'accueil

Figure 1 — Le menu d'accueil

En lançant le jeu, vous arriverez sur le menu ci-dessus. Il a fallu du temps avant de s'accorder sur un design qui plaisait à tout le monde. Mais après quelques soutenances intermédiaires, nous avons su nous mettre d'accord sur celui-ci.

Il est composé d'un fond animé dans lequel une caméra filme un décor que nous avons mis en place.

Il y a ensuite le nom du jeu qui est affiché et quatre boutons qui permettent d'orienter le joueur dans la partie qu'il souhaite.

Le bouton "Solo" fait charger la scène de choix de personnage, le bouton "Multijoueur" oriente le joueur dans la scène de création de lobby, tandis que le bouton "Charger une partie" va ouvrir un menu de choix de parties enregistrées,

mais nous verrons cela plus tard.

Pour finir, un bouton option permet de faire apparaître différentes options, comme vous pouvez le voir ci-dessous.

Figure 2 — Le menu d'accueil avec les options affichées

3.1.2 Les options du menu d'accueil

Dans ce menu, le joueur peut changer la langue du jeu en cliquant sur le bouton "Langue" parmi ces 5 langues : Français, Anglais, Espagnol, Allemand, et Mandarin.

A chaque clic, une variable entière sera incrémentée, qui définira la langue du joueur. A chaque clic, le drapeau du bouton est modifié parmi les 5 langues.

Nous définissons ensuite un getter public qui permet de récupérer la variable entière mais modulo 5, car il n'y a que 5 langues. Nous modifions ensuite les langues de la scène à partir de listes de chaines de caractères de 5 éléments, et nous faisons de même lors du chargement de chaque scène.

Nous avons rencontré plusieurs problèmes pour mettre le jeu en chinois. Il était très important de proposer cette option car le jeu se déroule dans une légende chinoise.

En effet, toutes les polices ne peuvent pas afficher tous les caracètres existants, notamment les caractères de chinois traditionnel. De plus, les polices qui affichent le chinois traditionnel n'affichent pas les caractères latins. Nous étions dans une impasse, mais heureusement, Unity permet de faire fusionner 2 polices

ensembles, en rajoutant à l'une certains caractères de l'autre.

Figure 3 — Paramètres de polices

Comme vous pouvez le voir ci-dessus, nous avons ajouté tous les caractères que nous utilisons dans notre traduction, de créer une nouvelle police que nous mettons sur tous les textes.

Pour finir, nous détaillerons l'option volume plus tard.

3.1.3 Le menu de chargement de partie

Le jeu Senzo-No-Michi étant très long dans la durée, nous voulions permettre au joueur d'enregistrer sa partie pour la reprendre plus tard.

Nous pouvions nous contenter de créer juste une sauvegarde au maximum, ou seulement 3. Mais, nous avons vu les choses en grand. Nous avons décidé d'intégrer notre propre explorateur de fichiers dans notre jeu. Voyez plutôt ci-dessous.

Figure 4 — Le menu de chargement de partie

Lorsque l'on clique sur charger une partie dans le menu d'accueil, cela fait apparaître sur la même scène notre explorateur de fichiers.

En effet, il scanne tous les fichiers dans le dossier de sauvegarde à l'emplacement du jeu, et les affiche sous forme de liste à l'écran.

Si c'est un dossier, cliquer dessus permet d'aller dans le dossier, et afficher le contenu du dossier.

Si c'est un fichier, cliquer dessus permet de déchiffrer le fichier avec la méthode Cesar de clé 128, et de le lire. S'il n'est pas corrompu et qu'il est au format XML, alors le jeu se lance à partir de la sauvegarde.

Figure 5 — Le menu de chargement de partie dans un dossier vide

Nous tenons à préciser que le bouton “Nouveau dossier” permet de créer un dossier dans le répertoire actuel et que le bouton “Retour” permet de remonter d’un dossier, tant qu’on n’est pas au dossier de stockage.

S’il n’y a rien dans le dossier actuel, un message s’affichera pour dire “Dossier vide” comme vous pouvez voir si dessus.

3.1.4 Menu de choix de volume

Étant donné que nous avons un jeu basé sur le son, nous avons ajouté dans les paramètres un curseur pour changer le volume du jeu. Le volume du son est géré sur toutes les scènes à partir de ce menu.

Le volume est une valeur entre 0 et 1. Or Unity permet de stocker des valeurs avec des identifiants. C’est grâce à cela que le volume du son est synchronisé entre toute les scènes.

Figure 6 — Slider de volume

3.1.5 Menu choix de personnage solo

Étant donné que nous avons plusieurs personnages avec des compétences différentes, nous devions intégrer un menu de choix de personnage

Une des possibilités que nous avions était d'offrir au joueur la possibilité de choisir tous les éléments de son personnage : son sexe, sa couleur de peau, sa coupe de cheveux, son armure, son arme, etc...

Les assets que nous avions nous le permettaient, mais après avoir compris comment fonctionnaient les assets, nous nous sommes rendu compte qu'il aurait été bien trop long de faire cela. En effet, ce n'était pas très compliqué de faire cela mais très répétitif.

Nous aurions dû ajouter tous les éléments des personnage (bras, têtes, torses, ...) ainsi que leur "materials" pour pouvoir recréer les personnages, le tout en plaçant correctement les différents éléments du corps.

Nous avons compté, il y aurait eu plus de 600 combinaisons de personnages, donc nous ne pouvions pas créer un “prefab” par combinaison ce qui aurait rendu la tâche bien plus simple.

Nous avons donc décidé d’ajouter un menu dans lequel on voit les 4 uniques personnages possibles, 2 féminins et 2 masculins. Ils sont chacun dans une salle entourée de torches, et sont dans leur animation par défaut. La caméra qui les filme tourne autour d’eux entre un angle de -45° et un angle de +45° pour rendre l’image moins statique.

Lorsque l’on clique sur un personnage, les torches qui entourent le personnage émettent plein d’étincelles, et le personnage se retrouve encadré par un cadre d’herbes. Ainsi, en multijoueur, chaque personnage ne pourra être choisi qu’une seule fois.

Figure 7 — Menu de choix de personnage

Les positions et rotations des caméras sont calculées en temps réel par trigonométrie.

Nous pouvons noter que pour simplifier les calculs, nous avons implémenté des listes de GameObject plutôt que de noter les variables une par unes, ce qui permet d’y voir plus clair, voyez plutôt ci-dessous.

Figure 8 — Les listes dans Unity

Dans ce menu notamment, les designs que nous avons fait et que nous vous présenterons plus tard prennent tout leur sens. Vous pouvez voir ci-dessous notre ancien menu de choix de personnage qui n'avait pas les designs actuels.

Figure 9 — L'ancien menu de choix de personnage

Cette intégration a redonné du courage à l'équipe qui perdait peu à peu la motivation. Après les avoir intégrés nous prenions du plaisir à lancer le jeu et à continuer d'apporter les fonctionnalités nécessaires.

3.1.6 Le menu carte

Le menu de la carte a pour objectif de donner au joueur la possibilité de voir l'ensemble de l'open world autour de lui.

Pour commencer, en solo, le menu carte lorsqu'il est actif met automatiquement le jeu en pause. En multijoueur, le menu carte empêche simplement le joueur de se déplacer car on ne pourrait pas mettre en pause le jeu pour tout le monde.

Pour cela, une caméra orthographique orientée vers le sol est placée au-dessus du niveau et permet de voir l'ensemble de la carte.

Le personnage peut utiliser la molette de sa souris pour pouvoir modifier le rayon de la caméra et donc pour pouvoir zoomer.

Il y a un zoom maximal et un zoom minimal pour que le joueur ne puisse pas découvrir chaque détail du monde dans lequel il évolue.

Il peut quand même se déplacer dans le monde en appuyant sur ses les touches de déplacement.

Figure 10 — Le menu carte une fois déroulé

Pour lancer la carte, le joueur appuie sur une touche définie avant : la touche M par défaut, et qui permet de mettre le jeu en pause s'il est en solo mais pas en multijoueur.

Elle fait également disparaître la mini-carte, et les informations et boutons annexes.

Lorsque le jeu reprend, la caméra aérienne de la carte est désactivée, pour optimiser la mémoire.

Figure 11 — Le menu carte lorsque l'on se déplace / que l'on zoom

Il est également important de mentionner que le joueur peut se déplacer sur la carte jusqu'à une certaine limite : les montagnes du bord de la carte. Aussi quand il dézoome, la caméra se replace automatiquement pour ne jamais afficher les bords de la carte.

Afin de rendre ce menu plus dynamique, lors de son ouverture, nous avons décidé que la carte s'ouvrirait depuis le centre. De la même manière, lorsqu'on ferme la carte, la carte se réduit et la partie reprend.

3.1.7 Le menu mini carte

Le menu mini-carte a été implémenté très tôt, mais il a connu de grandes évolutions. Il permet au joueur d'afficher un aperçu de ce qui l'entoure à tout moment. Pour cela, comme pour le menu carte principale, il y a une caméra orthographique orientée vers le sol qui suit la position du personnage.

La caméra ne suit pas la rotation du personnage, mais une petite icône directionnelle la suit pour indiquer le sens du personnage.

Figure 12 — Le menu mini carte

À l'aide d'un masque, la carte est rendue ronde, ce qui donne une impression de boussole et est bien plus cohérent.

3.1.8 Le menu arbre de compétence

Le Menu inventaire a été implémenté dans l'objectif de donner la possibilité au joueur de choisir ses compétences. Il est composé d'un titre "Compétences" (qui s'adapte en fonction de la langue actuelle), et de plusieurs images et titres.

Il possède une flèche qui dépasse du haut lorsque le menu est fermé. En cliquant dessus, cela permet de faire dérouler le menu venant du haut. Lorsqu'il est ouvert, la flèche s'inverse et permet de refermer le menu.

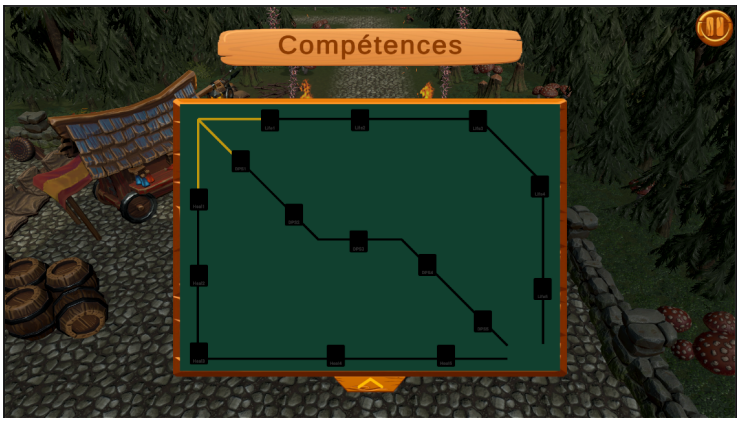


Figure 13 — Le menu inventaire en début de partie

En fonction du niveau du joueur, il pourra débloquer des compétences qui lui accorderont des bonus. Chaque compétence débloquée est sensée s'illuminer en orange. Pour cela, une fonction *Actualize()* a été créée qui permet d'adapter le tableau de booléens des compétences débloquées à la partie graphique.

Ensuite lorsque le joueur appuie sur l'un des boutons, le jeu vérifie que la case précédente a été débloquée, et si oui il modifie le tableau de booléen et actualise l'arbre.

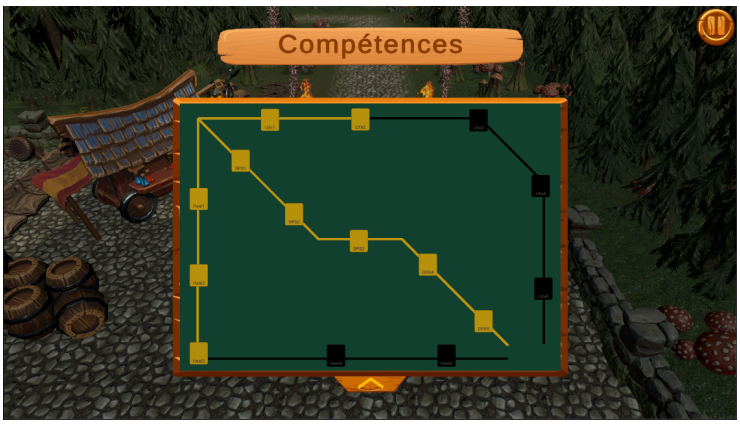


Figure 14 — Le menu inventaire en cours de partie

Il a été jugé inutile de vérifier que toutes les cases précédentes de l'arbre ont été débloquées et mieux pour des raisons d'optimisation.

Le problème majeur rencontré a été l'optimisation. En effet si le bouton du milieu est pressé, le jeu doit changer la couleur de deux traits car la branche de l'arbre tourne. En revanche, pour d'autres boutons, il n'y a qu'un seul trait. Au total, il y a donc vingt traits pour quinze boutons. Il était donc impossible de faire une boucle for, donc il y a actuellement quinze *if/else*.

Pour remédier à ce problème, plusieurs solutions ont été proposées. Premièrement, le design de l'arbre aurait pu être revu pour que la branche du milieu ne tourne pas. Seulement, il est regrettable de sacrifier le design au profit des performances (surtout pour une action qui est très rare dans la partie). L'autre solution aurait été de créer deux traits confondus là où il n'y en a qu'un, mais cela ferait trente traits au lieu de vingt et ne ferait que décaler le problème.

Nous avons donc décidé de garder ce système, en nous disant que de toutes façons, le menu n'est pas actualisé à chaque image mais plutôt très rarement, et le joueur ne voit pas le délai.

3.1.9 Le menu pause

Jusqu'à présent, nous n'avions pas implémenté de menu de pause. Le joueur peut déjà mettre le jeu en pause via la carte ou le menu inventaire, mais le jeu avait besoin d'un menu pause pour pouvoir aller dans les options, ou enregistrer la partie.

Un bouton permet de mettre en pause, mais cela fonctionne aussi lorsque l'on appuie sur échap et qu'on n'est pas dans un menu.

Figure 15 — Le jeu en pause

Comme vous pouvez le voir ci-dessus, il y a un bouton "Reprendre" qui fait la même chose que le bouton play en haut à droite, qui fait la même chose que de réappuyer sur échap.

Le bouton "Enregistrer" permet d'enregistrer le jeu en format XML avec un XElement qui est chiffré en César avec une clé 128. Nous enregistrons :

— La position du joueur
— Le numéro du joueur
— Son niveau
— Sa vie
— Son expérience
— Ses compétences
— Les boss qu'il a tué

Voici un aperçu d'un élément :

```
<SenzoNoMichi Player="1" Life="0" x="105.585" y="0.014916" z="81.18269"
Level="1" xp="0" Competence1="0" Competence2="0" Competence3="0"
BossKilled="000" /\>
```

Figure 16 — Exemple de sauvegarde en XML

La partie pourra ensuite être chargée depuis le menu d'accueil.

Ensuite le bouton "Quitter" permet de quitter la partie pour retourner au menu d'accueil.

Le bouton "Options" permet comme dans le menu d'accueil de faire apparaître le volume et le bouton de choix de volume

Figure 17 — Le jeu en pause avec le menu option déroulé.

Cliquer sur "Langue" permet de mettre à jour le paramètre entier du menu d'accueil et de mettre à jour la langue actuelle.

Nous n'avons pas encore précisé, mais lorsque l'on appuie sur options, le bouton langue qui est caché derrière le bouton options descend de manière fluide. Même chose pour le volume qui apparaît du bas, et le menu de compétences qui apparaît du haut.

Pour cela, lorsque l'on appuie sur options, on change une variable booléenne à vrai, et tant que cette variable est vraie et que les coordonnées Z de volume sont inférieures à une certaine valeur, on augmente volume de quelques pixels.

On fait évidemment la même chose dans l'autre sens pour pouvoir faire rentrer le menu déroulant volume.

Dans cette partie, nous allons expliquer l'architecture des menus dans le jeu.
Nous avons décidé de créer un objet préfabriqué dans lequel nous avons ajouté
— Le menu mini carte
— Le menu carte
— Le menu pause
— Le menu de compétences

— Le menu GUI expérience

Il permet de désactiver certains menus à certains moments bien choisis.

Par exemple, lorsque le joueur ira dans le menu de compétences, la mini carte ne s'affichera plus, on ne pourra plus aller en pause, l'expérience du joueur et son combo ne s'afficheront plus et on ne pourra plus ouvrir la carte jusqu'à ce que le joueur ait quitté le menu.

Nous faisons la même chose avec le menu carte et le menu pause.

Cette ergonomie ne paraît être qu'un détail, mais cela change beaucoup la manière dont le joueur va voir le jeu et il reviendra plus souvent.

Nous nous sommes rendu compte de ça lorsque l'on a commencé à prendre de plus en plus de plaisir à nous connecter à notre propre jeu après avoir fait ces ajouts d'ergonomie.

3.2 Multijoueur

3.2.1 Intégration des personnages dans le jeu

Dans le mode solo nous avons une scène pour choisir les personnages, pour instancier les personnages sur la carte nous avons tout d'abord voulu instancier directement le joueur avec une Prefab. Cependant, cette méthode posait certains problèmes et engendrait des erreurs.

Pour régler cela, tous les joueurs sont tous déjà instanciés sur la carte (cela ne pose pas un problème car nous ne sommes pas en multijoueur) et sont supprimés par rapport au choix que nous avons fait dans l'ancien menu. (Les informations sont transmises de scène en scène avec des instances).

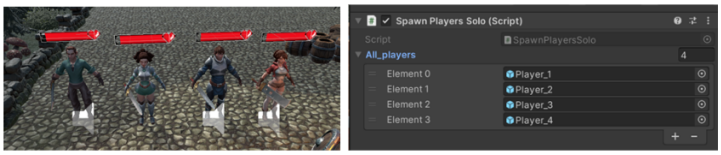


Figure 18 — Joueur instanciés / script qui permet de les supprimer

3.2.2 Choix des assets

Au début du projet nous avions pensé à faire un projet en réseau local, Antoine avait déjà fait des projets de ce type. Ce genre d'application peut facilement se mettre en ligne avec certaines applications comme ngrok ou hamachi qui permettent de transférer les données locales sur internet (bien utile pour héberger des serveurs Minecraft par exemple).

Cependant, nous avons découvert des assets comme Photon ou Mirror qui permettent de créer des jeux en multijoueur. Ce dernier permet de créer de plus grandes infrastructures multijoueur, tandis que Photon est plus simple, et fonctionne très bien pour de petits projets.

Figure 19 — Photon / Mirror

Photon permet d'avoir un nombre de requêtes limitées et cela gratuitement, mais qui suffisent totalement pour notre projet.

3.2.3 Mise en place de Photon sur le projet

Pour utiliser Photon il faut tout d'abord télécharger l'asset Photon pun 2 sur l'asset store et rentrer un identifiant (ID). Cette ID permet de lier une application à son compte, afin que chaque utilisateur de l'application puisse se connecter entre eux.

Photon comporte plusieurs sous possibilités comme : pun, realtime, server, bolt, quantum.

Le choix s'est plutôt tourné vers pun et realtime qui sont les plus communs pour ce genre de projet, mais finalement nous avons pris Photon realtime pour sa plus grande flexibilité et une plus grande maniabilité du serveur et des fonctionnalités utiles pour ce projet.

Il y a une autre raison que nous verrons plus tard dans le chapitre.

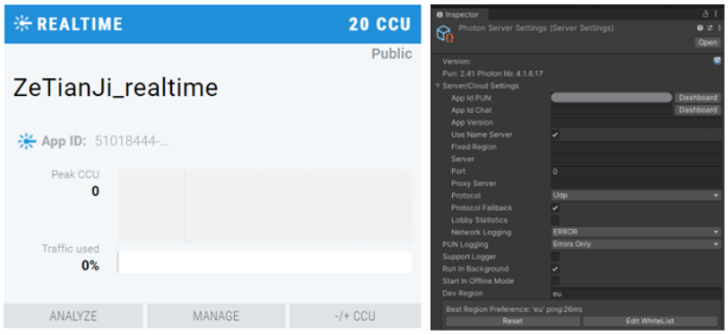


Figure 20 — Dashboard de Photon / Paramètres Photon utilisés pour le projet

3.2.4 Connexion au multijoueur

Pour utiliser le multijoueur, il faut en premier lieu se connecter au serveur. Dans le lobby lorsque l'on clique sur Multijoueur, une nouvelle scène est chargée.

Tout d'abord, des paramètres sont créés comme un pseudo aléatoire, et le numéro de version du jeu. La version du jeu permet, lorsqu'il y a des mises à jour, que les utilisateurs qui ne l'ont pas faite ne puissent rejoindre le serveur.

Ensuite le client est connecté au serveur de Photon en utilisant ces paramètres.

3.2.5 Liste des parties - Menu multijoueur

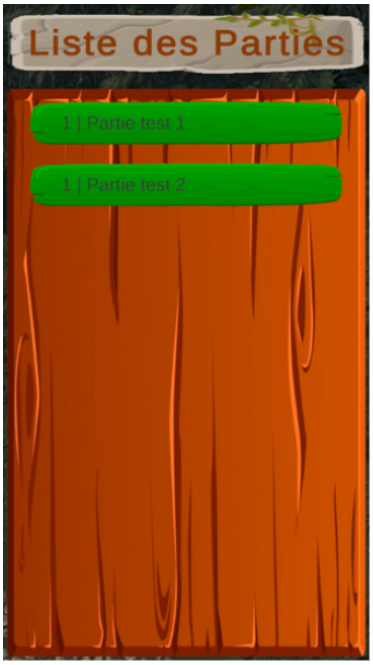


Figure 21 — Liste des parties du menu multijoueur

Photon est en partie basé sur la programmation d'évènements. Un évènement est par exemple lorsqu'une personne rejoint une partie, se déconnecte, etc...

Ainsi lorsqu'une partie est créée sur notre jeu, un évènement est lancé ce qui nous permet de collecter la liste des parties créées, en cours et supprimées. Il a donc fallu rendre notre liste dynamique pour qu'elle affiche uniquement les parties en cours et la suppression les autres. Le nombre de joueurs et le nom de la partie sont alors affichés.

3.2.6 Créer une partie - Menu multijoueur

Figure 22 — Liste des parties du menu multijoueur

Au préalable une « room » sur photon est un espace virtuel où les joueurs peuvent se connecter et interagir les uns avec les autres. Ainsi, une room peut représenter une partie en cours, dans laquelle plusieurs joueurs cohabitent sur une carte donnée.

Lorsque le bouton « Créer / rejoindre une partie » est pressé, des paramètres de room sont créés (le nombre de joueurs maximal) et le texte contenu dans sa boîte associé est récupéré.

Une room est donc créée avec ces paramètres et ce nom. Lorsqu'elle est enfin créée, un évènement est lancé permettant le chargement d'une scène (le menu de choix des joueurs).

3.2.7 Changer de pseudo - Menu multijoueur

Figure 23 — Choix du pseudo dans une boîte de texte

Photon permet de changer dynamiquement les pseudos, ainsi le pseudo par défaut au format de « *Zetianji[nbr_aléatoire]* » peut être changé pour une expérience plus personnalisée.

3.2.8 La vue Photon - Synchronisation Client

La vue Photon, soit « Photon View » est attachée à un objet dans la scène Unity et représente la vue de cet objet sur le réseau. Elle permet de synchroniser les changements d'état de l'objet entre les clients connectés à une room Photon.

En d'autres termes, la Photon View permet de rendre l'objet visible et contrôlable par d'autres joueurs dans le jeu multijoueur. Chacune d'elle possède un identifiant unique. Il y a d'autre composants prédéfinis de photon qui permettent de synchroniser la position, les animations, etc...

3.2.9 Le master client - Syncronisation Client

Sur Photon le premier client à s'être connecté à une room est le client « master ». Celui-ci possède certains privilèges en plus que les autres. Par exemple lorsque nous lançons la partie avec un client qui n'est pas master, toutes les personnes connectées ne sont pas changées de scène.

Il faut donc penser à utiliser cette fonctionnalité de client master, laquelle parfois peut représenter une contrainte plutôt qu'un avantage.

3.2.10 Les RPC - Syncronisation Client

Définition : Les RPC, soit « *Remote Procedure Call* » est un concept utilisé dans la programmation réseau pour permettre l'exécution de fonctions ou de méthodes sur un système distant. Dans le contexte de Photon, les RPC sont des appels de procédure à distance utilisés pour la communication entre les clients et le serveur.

Lorsque vous utilisez Photon, vous pouvez définir des fonctions ou des méthodes spécifiques que vous souhaitez appeler à distance depuis un client vers le serveur, ou vice versa.

Ces fonctions doivent être marquées comme des RPC afin d'indiquer qu'elles peuvent être appelées à distance.

Lorsqu'un client appelle une fonction RPC, celle-ci est exécutée sur le serveur, et si le serveur appelle une fonction RPC, elle est exécutée sur les clients

qui sont connectés à la même *room*.

Les RPC sont utilisés pour envoyer des instructions, des mises à jour de l'état du jeu ou des actions spécifiques entre les clients et le serveur.

Par exemple, lorsqu'un joueur appuie sur un bouton pour attaquer un ennemi, un appel RPC peut être effectué pour informer le serveur de l'action effectuée par le joueur. Le serveur peut ensuite traiter cet appel RPC, et informer les autres clients connectés à la *room* de cette action pour mettre à jour leur propre état de jeu.

Les RPC peuvent également prendre des paramètres pour transférer des données spécifiques, ils peuvent également être envoyés à des clients particuliers, comme juste au master, tout le monde, tout le monde sauf soi, etc. . ..

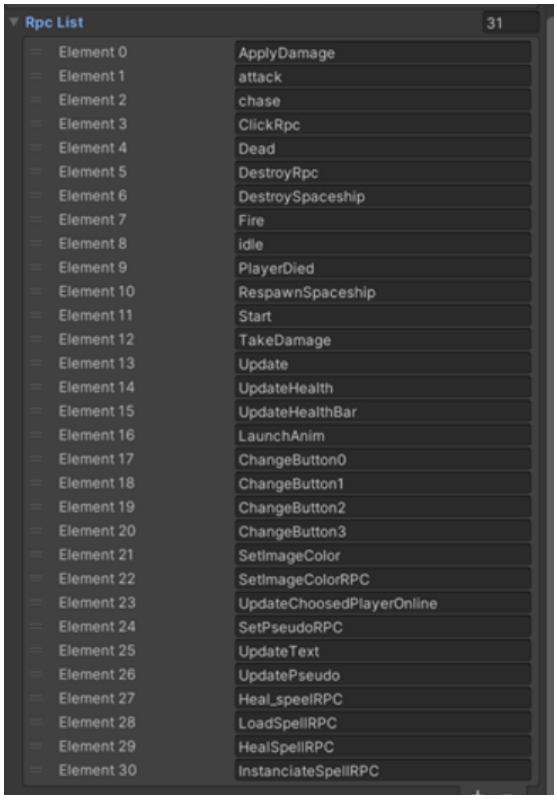


Figure 24 — Liste des RPC de notre projet

3.2.11 Menu de choix de personnage multijoueur

Figure 25 — Un joueur dans le menu de choix de personnage multijoueur

Lorsque l'on clique sur un joueur, il faut en informer tous les clients. Chaque zone de texte possède donc une PhotonView. Une requête RPC est alors envoyée avec le pseudo du joueur choisi ainsi que le personnage qu'il a choisi. Si un nouveau joueur se connecte, les autres clients vont alors lui envoyer les données des personnages sélectionnés pour éviter certains bugs.

Pour la suite ceci est sauvegardé :

— Le pseudo du joueur : il est sauvegardé grâce à une fonction de Unity
— La liste des joueurs choisis : celle-ci est stockée dans le script du menu, une instance en a été réalisée pour être facilement récupérée lorsque la nouvelle scène est chargée.

3.2.12 Instanciations dans la nouvelle scène

Définition : Un préfabriqué, communément appelé « *prefab* » est un terme utilisé pour décrire un objet ou une entité préconstruite et prête à être instanciée dans une scène du jeu.

Un *prefab* est un modèle ou un patron d'objet qui peut être réutilisé plusieurs fois dans le jeu. Il sert de modèle de base à partir duquel de multiples instances peuvent être créées. Un *prefab* peut inclure des composants, des scripts, des modèles 3D, des textures, des effets sonores ou toute autre ressource nécessaire pour représenter un élément du jeu.

Leur principale utilité réside dans leur capacité à faciliter la création et la gestion des objets dans une scène de jeu. Au lieu de créer manuellement chaque instance d'un objet avec toutes ses propriétés et ses composants à chaque fois qu'il est nécessaire, on peut simplement créer un *prefab* une seule fois, puis instancier des copies de ce *prefab* dans la scène du jeu autant de fois que nécessaire. Les modifications apportées au *prefab* se répercutent automatiquement sur toutes les instances créées à partir de celui-ci, ce qui facilite la mise à jour et la maintenance du jeu.

Lorsque la nouvelle scène (la carte principale) est chargée, un script est lancé qui permet d'instancier le *prefab* du bon joueur en fonction de la sauvegarde, ainsi que le pseudo du joueur. Chaque personnage du solo possède son propre *prefab* multijoueur, car contrairement au solo, les joueurs ne peuvent pas être de base sur la carte, il faut les instancier en ligne avec Photon.

Si un joueur rejoint après que chaque participant ait choisi son personnage, il pourra rejoindre la partie mais en temps que spectateur. Il pourra se déplacer sur toute la carte mais ne pourra pas interagir.

3.2.13 Conditions simples pour rendre compatible les scripts du solo

Pour fonctionner en multijoueur la majeure partie des scripts doivent avoir pour condition que si la PhotonView est celle du client le script fonctionne, sinon ne fait rien.

Un bon exemple de cette condition est que sans celle-ci un seul joueur peut contrôler tous les joueurs (ce qui est un tout petit peu problématique).

3.2.14 Faire fonctionner les IA du solo en multijoueur

Au début du développement les IA n'étaient pas synchronisées entre tous les clients, nous avons donc été contraints d'utiliser les RPC et la fonction de synchronisation de photon pour les animations et la position.

Lorsque nous ne sommes pas sur serveur il est simple de savoir sur quels objets le joueur va interagir, et par conséquent insérer des liens directs ou détecter des "tags". A part qu'en multijoueur il peut y avoir un nombre indéterminé de joueurs (nous en avons quatre, mais il est préférable d'en prévoir plus). Ainsi, il est plus compliqué d'avoir des liens.

Une méthode que nous avons utilisée, est de chercher les objets par composants (*FindObjectsOfType* de Photon). Grâce à ces composants nous pouvons récupérer certaines données, comme la position, l'orientation, la vie...

3.2.15 Problèmes rencontrés lors du développement

Unity...

Unity à des comportements déconcertants en multijoueur, comme par exemple de travailler avec la version éditeur du jeu et une version build laquelle ne se comportera pas de la même manière que deux versions build. Il faut alors procéder à la corrections de ces anomalies, réaliser de nombreux tests pour vérifier si tout fonctionne correctement selon le rendu souhaité.

Changement de la clé Photon

Après des semaines de développement et une ID Photon PUN, le projet ne fonctionnait plus du tout en multijoueur. Le simple fait de changer l'ID en Photon Realtime à réglé notre problème. Cela est probablement lié au fait d'avoir utilisé des fonctionnalités Realtime.

Téléportation entre les scènes

Au début nous voulions juste changer de scène le joueur, pour le faire passer de la scène principale aux scènes de boss. Mais cela créait des problèmes de duplications mais également des problèmes de synchronisation de données.

Nous avons opté simplement pour la solution de téléporter le joueur à une autre position sur la carte, ceci comporte beaucoup d'avantages. Unity étant bien optimisé et optimise la vision des éléments lointains, cela n'entache en rien l'expérience de jeux (et améliore les temps de chargement).

Figure 26 — Toutes les cartes sur la même scène

Problème multijoueur soutenance 2

Après une recherche dichotomique, le fichier qui faisait dysfonctionner le mode multijoueur était *EditorUserBuildSettings.asset*. Ce fichier permet apparemment de stocker les paramètres spécifiques à la construction du projet, notamment les paramètres de compilation, les plateformes cibles, les options de génération de l'exécutable et d'autres configurations de construction. Il a donc été rajouté au git.

3.3 Le site internet

3.3.1 L'architecture du site web

Le site internet est composé de 5 pages :
— L'accueil
— La présentation des personnages
— La présentation de l'univers autour des personnages
— La présentation du projet et de l'équipe
— La version lite du site
Nous avons organisé les quatre premières pages dans un menu en-tête qui est un bandeau.
Nous avons ajouté à ce bandeau le logo de l'équipe et un bouton "Télécharger".

Le site internet a été mis en ligne sur github.io. Au début nous projetions de le faire héberger sur free.fr. Nous avions créé un compte free pour l'occasion au nom de l'équipe, mais la mise en ligne était trop longue, donc nous avons abandonné le projet et finalement mis le site en ligne sur senzo-no-michi.github.io.

De plus, le fait que le site soit sur github nous permet de le modifier très facilement en cas de nécessité.

3.3.2 Le menu accueil

Le menu d'accueil est composé du titre du jeu, d'images du jeu et de liens placé à plusieurs endroits bien choisis qui redirigent vers les autres pages du site.

Nous avons choisi de prendre un desing de page mais aussi de site plutôt sombre pour rappeler un esprit jeu vidéo.

Un script permet de faire défiler les images qui sont sur la page d'accueil pour rendre la page plus dynamique.

Figure 27 — Le menu d'accueil

3.3.3 La page personnages

Cette section nous a servi à présenter nos quatre personnages, leurs rôles et pourquoi il sont présents. Nous tenons à relever qu'il y a deux personnages féminins et deux personnages masculins pour rajouter une certaine inclusivité.

Il est important d'écrire l'imaginaire, cela permet de mieux développer une communauté sur le jeu.

3.3.4 La page univers

Cette section nous a servi à écrire l'univers du jeu et l'histoire du jeu. En effet, la carte du jeu est immense, et il nous fallait la présenter.

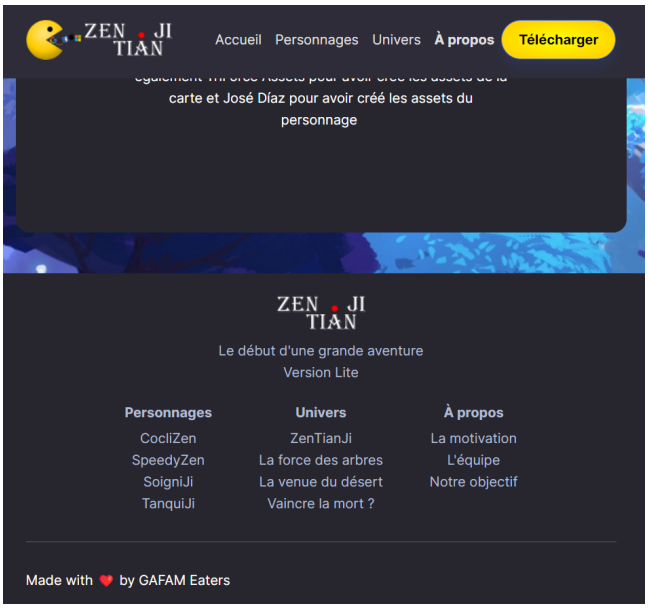


Figure 28 — La page univers

3.3.5 La page présentation de l'équipe

Il était important de présenter l'équipe dans notre site web. C'est ce que nous avons fait longuement et que vous pouvez lire si vous avez du temps à perdre.

Nous avons fait une mention à l'Epita, en disant que l'école pouvait parler de notre jeu car le projet a été fait dans le cadre de nos études.

Pour finir, nous avons fait un remerciment à Thomas N'Zaba qui a composé la bande son de notre jeu spécialement pour nous. Nous avons aussi remercié TriForce Assets et José Díaz pour avoir créé les *assets* de nos personnages et de la carte.

3.3.6 La version lite

Figure 29 — Version lite

Lien de cette page : https://senzo-no-michi.github.io/lite.html

Le lien vers cette version est dans un seul fichier a part et n'est référencé dans le site qu'en petit dans le bandeau de pied de page des autres pages.

Elle permet de donner des liens pour télécharger :

— Le rapport de soutenance 1
— Le rapport de soutenance 2
— Le rapport de projet
— Le manuel du jeu
— Le manuel d'installation
— Le jeu
— Le site web

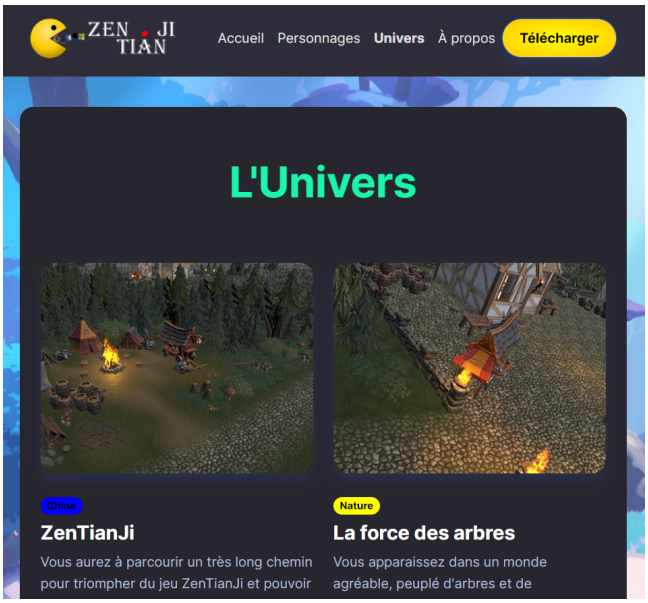


Figure 30 — Bandeau pied de page

3.4 IA

Lors de la première soutenance, nous avions implémenter une première intelligence artificielle qui suivait simplement le joueur en tout point du jeu, qui l'attaquait en boucle, et quoi qu'il se passe, réapparaissait une fois mort pour vous chasser de son territoire.

Lors de la seconde soutenance nous avions implémenté une intelligence artificielle plus complète qui avait pour but de détecter le joueur une fois qu'il entrait dans une certaine zone de vue, puis de le suivre et enfin, a une certaine distance, de l'attaquer.

Si le joueur sort de la zone de repérage, alors l'intelligence artificielle fait retourner l'ennemis a son point de départ et attend que le joueur revienne à une distance suffisamment proche pour redémarrer le processus.

Nous avons depuis implémenter un système pour faire revivre l'ennemis. En effet, une fois mort, nous ne nous occupions pas de faire réapparaitre l'ennemis. Aujourd'hui, l'ennemis une fois mort, attend que le joueur sorte d'une certaine zone autour de lui, puis, son animator se réinitialise, et il retourne à sa position d'apparition de base.

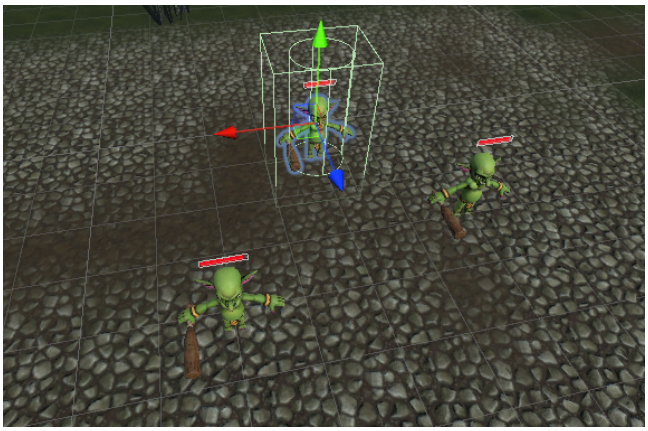


Figure 31 — Ennemis principal de la zone 3

Nous utilisons toujours cette intelligence dans notre jeu, en complément, a été implémenté une intelligence artificielle pour un boss qui est assez générale, elle reprend le même principe qu'a première, décrite plus tôt. Nous reprenons donc le principe de suivis puis de zone d'attaque. A cela nous lui rajoutons de nombreux choix d'actions basés sur un système aléatoire.

L'attaque est choisie parmi trois à choix égales pour avoir une expérience visuelle plus immersive. En plus de cela, nous avons implémenter un temps

d'attente aléatoire pour permettre au joueur d'attaquer le boss à certains moments, le lancement de cette action dépend de la distance du boss avec le joueur.

Cela a pour simple but d'augmenter la difficulté du jeu en ne laissant pas au joueur le temps de s'habituer à un système. Aucun plan d'action du boss n'est donc prévu à l'avance.

L'absence de pattern pour le boss qui tourne en boucle nous permet par ailleurs de renforcer la difficulté du jeu vis-à-vis du rythme puisqu'il faut pouvoir non seulement attendre le bon moment pour que le boss s'arrête mais aussi taper en rythme pour pouvoir faire plus de dégâts.

Nous avons implémenté une variante de cette intelligence artificielle pour le boss final. En effet nous avons repris le code des premiers boss pour garder ce côté imprévisible que nous avons plus tôt décrit. En plus de cela nous avons implémenter un sort de zone pour ce boss ainsi qu'un mode vol ou le joueur ne peux donc pas le taper.

Le but est donc de rendre les boss impossibles à battre si on ne fait pas attention aux jeune effet dans un premier temps l'intelligence artificielle vérifie que le joueur soit bien dans la zone de repérage c'est-à-dire tout simplement dans la zone du boss point après cela si le joueur était une certaine distance alors l'animation de marche se lance et le boss se déplace joueur ; si celui-ci est assez proche alors l'intelligence artificielle a 3 choix à probabilité égale de tomber sur 3 attaques différentes visuellement mais ne compliquant pas plus la tâche aux joueurs qu'une attaque simple (cela a donc pour seul but de renforcer l'immersion du joueur dans ce combat).

De plus, ces attaques s'effectuent certes avec un temps d'attente entre chacune d'entre elles mais tout de même très rapidement ; cela force donc le joueur à user de stratégies afin de vaincre le boss.

Une des actions aléatoires se base donc comme dit précédemment sur une zone comprise entre la zone de détection et la zone d'attaque : si le joueur se situe à une certaine distance du boss alors il y a une certaine probabilité que le boss mobile pendant quelques secondes (une 2nde d'attente pendant lesquels le boss reste immobile sont aussi implémenté via une notion de probabilité : cela peut donc être très court comme suffisamment long pour donner un ou deux coups).

Le boss final comprend prend une troisième zone. Cette zone déclenche la capacité vol. Elle est gérée tout comme la zone aléatoire décrite précédemment mais ce situe entre cette fameuse zone aléatoire et la zone de détection.

Bon il y a tout comme pour la zone précédente une certaine probabilité que le joueur fasse face au mode vol. ce mode rend invulnérable le boss pendant

qu'il est en train de voler et durant ce laps de temps le boss va avoir une chance d'attaquer ou non.

Fameuse attaque sera manifestée par le sort de zone précédemment décrit. Ce laps de temps durant le combat de boss est uniquement implémenté pour augmenter la pression émise sur le joueur pour le combat du boss final ainsi que pour augmenter la difficulté à terminer le jeu.

Évidemment la difficulté présente dans le jeu et surtout une transcription de notre volonté de promouvoir au maximum le mode multijoueur.

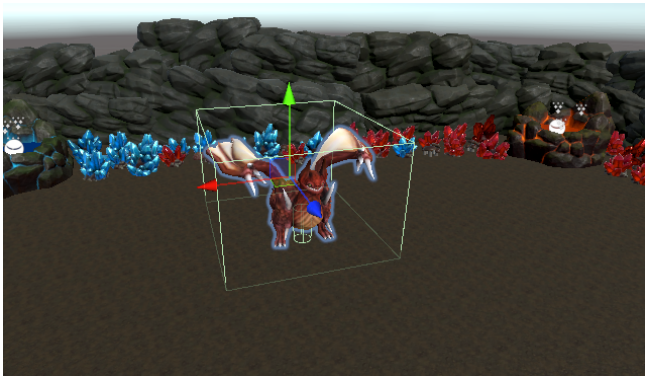


Figure 32 — Boss final dans son milieu naturel

Evidemment il est plus simple de les battre en multijoueur, le nombre aidant à la confusion du boss quant aux actions à effectuer.

De nombreux problèmes sont survenus durant l'implémentation de cette intelligence artificielle. Nous avions notamment prévu de faire des mouvements aléatoires à la place des moments où le boss reste immobile pour rendre le combat plus fluide.

Toutefois, nous n'avons pas réussi à focaliser le boss sur le joueur dans ces moments. Nous avons notamment essayé d'instancier une rotation de l'asset durant ces mouvements pour qu'il ne regarde pas dans la direction de son déplacement, malgré cela il regardait dans le vide ce qui n'était pas l'action souhaitée.

Cette solution a été la seule à réellement donner un résultat presque satisfaisant et après de nombreux essais nous en sommes venus à abandonner cette étape de déplacements aléatoires.

Un second problème a été l'implémentation du système aléatoire puisque le script s'effectue à chaque images par secondes. Or sur notre jeu, une seconde représente 60 images par secondes, nous avons donc dû ajuster les probabilités

pour ne pas que cela arrive toutes les secondes. En effet au début l'animation du boss tournait en rond, et cela même avec un pourcentage tel que 1/100 qui peut sembler peu probable.

Sur le boss les périodes d'immobilité sont donc aléatoires et n'ont pas énormément de chances d'arriver ce qui oblige le joueur à garder une certaine distance avec le boss. Pour renforcer cette nécessité de garder une certaine distance en attendant le bon moment pour frapper, l'attaque du boss est assez élevée, ce qui demande donc un haut niveau de personnage non seulement pour survire mais aussi pour faire des dégâts vis-à-vis de la vie du boss.

Tout cela se passe pendant le combat, mais avant le combat, L'intelligence est programmée pour attendre un certain temps avant de lancer le script. Cela a pour but de ne pas tout de suite pourchasser le joueur et lui laisser le temps de s'habituer à environnement. De plus, la fonctionnalité décrite plus tôt par rapport à la zone de détection sert ici comme déclencheur de l'intelligence artificielle ainsi que de l'animator du boss.

Conquérant l'animator, un des problèmes rencontré a donc été d'adapter l'animator des autres boss à l'intelligence artificielle pour qu'il n'y est pas de conflit vis-à-vis des différente transitions et animations souhaitées. Et cela notamment pour le dernier boss du jeu qui a en plus un mode vol.

En effet, ce mode vol dans l'animator est géré comme le serait une deuxième phase. Ces transitions passant obligatoirement par le décollage ou l'atterrissage, il a été assez difficile d'adapter exactement les animations pour un moment ou une action voulue : d'où la réadaptation du script pour le boss final.

De nombreux temps de pause sont implémentés avec un simple test en se basant sur le temps entre les images par secondes du jeu en appelant "Time.delatatime". Cela a pour but de laisser l'animator finir son animation avant de passer au reste du script, notamment pour éviter le boss avance pendant une transition.

Contrairement à l'intelligence artificielle utilisée pour les ennemis de base, nous n'avons pas préparé cette intelligence artificielle pour une réapparition, en effet, une fois un boss défait, il n'est pas possible de recommencer le combat contre celui-ci contrairement aux ennemis de base qui eux, reviennent à leur position de départ et son réinitialisé vis-à-vis de leurs stats.

3.5 Le NavMesh

Pour les différents déplacements de l'intelligence artificielle, nous utilisons le NavMech. Pour plus de précision, l'outil NavMesh est une fonctionnalité du moteur unity permettant de délimiter une zone ou peuvent marcher ou se déplacer

les ennemis que nous avons implémenté.

Nous avons donc équipé tous nos ennemis d'un "navmech agent" celui-ci permettant à l'intelligence artificielle d'être directement connecté au navmesh que nous avons mis en place sur la carte du jeu. La difficulté avec cet outil a été de correctement prendre en compte les différents éléments que nous avions préalablement placé sur la carte.

En effet, dans un premier temps nous avons fait face à un problème concernant cet outil car nous ne savions pas exactement comment capter correctement les différents décors que nous avions posé sur la carte du jeu l'ennemi pouvait donc à tout instant traverser les arbres, les murs, les buissons, ...

Cela était donc un problème puisque l'intelligence artificielle était capable de traverser certaines parties de la carte du jeu auquel elle ne devait normalement pas avoir accès.

De nombreuses solution était donc envisageable nous avions notamment pensé à créer des collisions au bord des différentes routes afin que les intelligences artificielles ne puissent pas en sortir car le navmesh peut prendre en compte les collisions avec ces espaces-là.

Toutefois, cette solution n'était pas satisfaisante puisqu'elle demandait énormément de temps de travail en plus sur la carte principale du jeu, mais aussi rajoutait énormément éléments sur la carte du jeu : étant donné que notre jeu est en monde ouvert, notre carte est déjà très grande et beaucoup d'éléments la composent, un sur chargement de celle-ci n'était donc pas à souhaiter.

Pour les mêmes raisons nous n'avons pas décidé de dédoubler la carte comme il aurait été possible de faire pour souligner avec une seconde carte les endroits où les ennemis auraient eu le droit de marcher. Nous avons donc finalement réussi à trouver une solution nous convenant en effet si tous les éléments qui composent la carte sont repérés comme étant "static" aux yeux d'unity, ceux-ci sont plus facilement pris en compte par le navmesh.

Loin d'être parfaite cette technique offre moins de liberté à l'intelligence artificielle vis-à-vis de ces déplacements ; de plus, plus la zone est difficile moins celle-ci dispose de paysage complexe.

Cela rejoint donc notre volonté d'augmenter la difficulté du jeu en fonction des zones parcourues les 2 premières réduisant au maximum les mouvements de l'intelligence artificielle en dehors des routes dessiner au préalable ; contrairement 2 dernières zones qui elle ont un environnement moins fourni ce qui laisse plus de liberté à l'intelligence artificielle vis-à-vis de ces mouvements.

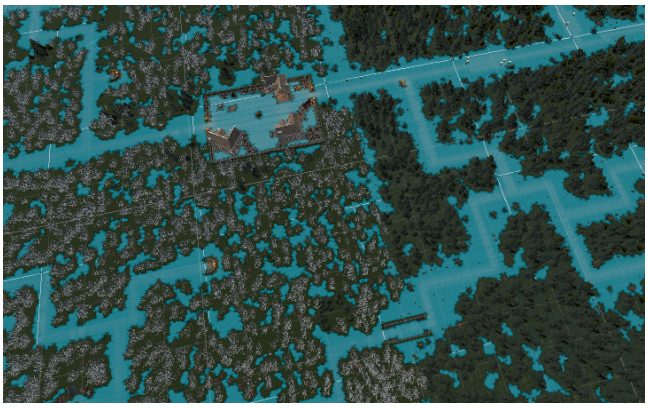


Figure 33 — NavMesh de la carte principale

En ce qui concerne les différentes salles des boss, nous avons directement délimité les limites du terrain avec des collision.

En effet, il est plus simple dans de petits environnements d'instancier les collisions aux 4 coins d'une petite salle carrée plutôt que d'une grande carte aux divers chemins et horizons. Cela donne donc une entière capacité au boss de se déplacer librement dans sa salle afin de mettre plus de pression sur le joueur.

La difficulté était donc de trouver une solution concernant le navmesh permettant non seulement que les intelligences artificielles et donc les ennemis puisse correctement se déplacer dans les zones pourvus à cet effet sur la carte principale tout en laissant l'entier contrôle de sa zone au boss.

En effet toutes les salles de boss étant directement présente sur la scène de la carte principale, il fallait trouver un compromis permettant non seulement d'avoir un Navmech correct l'entièreté de la carte principale tout en étant assuré d'en avoir un dans les différentes salles de boss.

Enfin, le timing de certaines actions de l'intelligence artificielle et donc des ennemis ont été prévus spécialement pour certains types de personnages.

En effet nous pouvons par exemple prendre l'exemple du personnage typé assassin : ce personnage dispose des parfaits timings pour attaquer entre le temps d'attaque des ennemis. Cela permet aux joueurs de chercher un moyen pour tirer profit au maximum des personnages et de leurs capacités.

3.6 Niveaux et design du jeu

Lors de la première soutenance, nous avions posé les bases de nôtres jeux, une seule zone de la carte était disponible comprenant un village d'épart de

notre aventure ainsi qu'un portail téléportant à la zone du premier boss.

Lors de la première soutenance Nous n'avions pas de boss, Nous avons lors de la seconde soutenance, présenté notre ennemi principal que nous pourrons retrouver à des endroits spécifiques de la carte prévus à cet effet.

En effet, nous pouvons tout au long de la carte retrouver de nombreuses zones au sein desquels sont disposées plusieurs ennemis qui ont pour but de non seulement freiner la progression du joueur mais aussi et surtout de le conforter dans ses habilités au combat avant les boss fight mais aussi de lui faire monter de niveau avant ceux-ci.

Tout au long de la carte du jeu nous avons donc 4 zones et l'ennemi principal qui est le gobelin et répartis selon 4 niveaux dans chacune de ces zones.

Ici l'IA reste la même mais la difficulté de celle-ci et donc augmenter via les différentes caractéristiques suivantes : l'attaque, la vitesse de déplacement, et les points de vie.

De nouveaux ajouts ont donc été fait au niveau de la carte, en effet nous pouvons maintenant retrouver 4 zones de tailles plus ou moins égales et décorée dans des styles différents comme une zone désertique, une zone marécageuse, la zone de foret que nous avions déjà présentée et enfin une zone chargée en magie.

Ces zones sont organisées pour voir une certaine nuance en therme de population à travers les villages comme travers les décors, devenant de plus en plus arides et de plus en plus vide.

Trois portails sont disponibles à travers ces zones. Les trois portails sont donc dans les trois zones autres que le foret pour laisser le joueur de s'habituer aux jeux ainsi que lui laisser le temps de monter en niveau pour ne pas qu'il y est trop de difficultés supplémentaires à vaincre le boss.

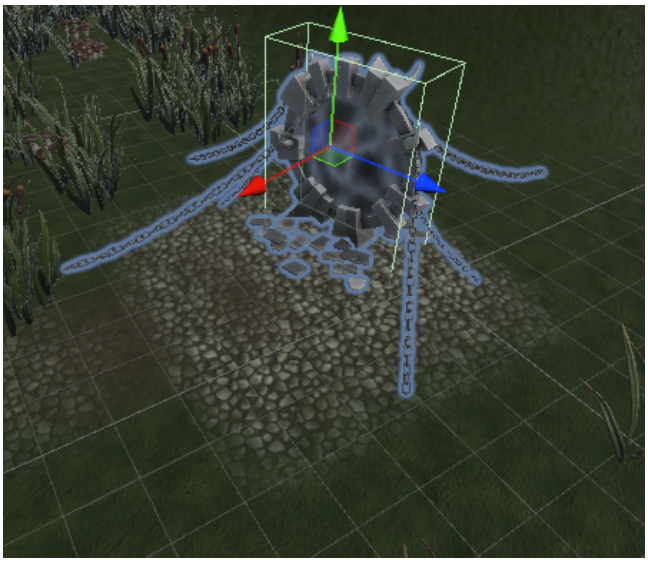


Figure 34 — Portail 1 zone 2

Chaque zone a ses caractéristiques, de plus, on peut retrouver dans chaque d'elles quelques coffres de niveaux 1 à 3 qui donneront donc de plus en plus d'expérience au joueur pour faciliter sa montée en niveau. Cela a aussi pour but de renforcer l'intérêt du joueur vis-à-vis de l'exploration de la carte.

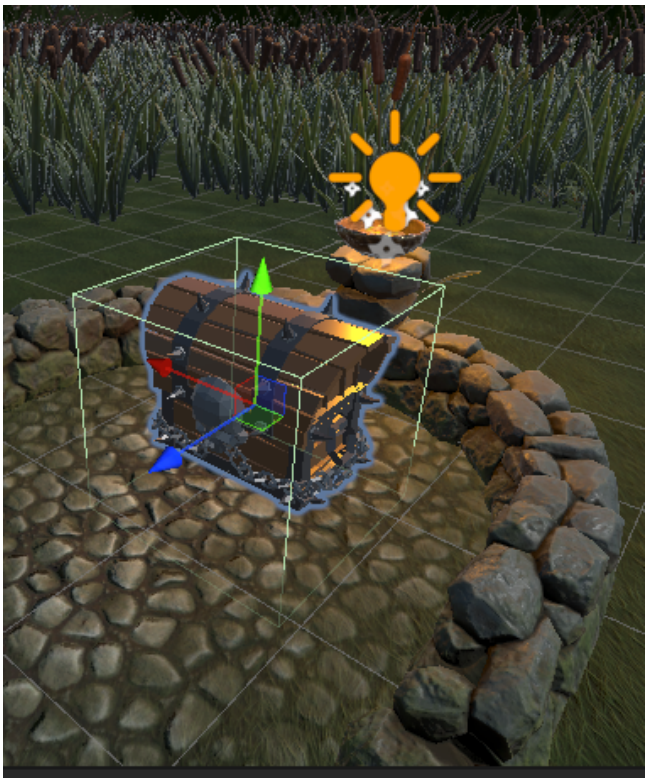


Figure 35 — Coffre niveau 2 Zone 2

À travers la carte du jeu nous pourrons notamment retrouver quelques campements de survivants ou de soldats. De plus, dans la seconde zone, nous retrouvons un second village. Le fait de ne retrouver un village que dans les deux premières zones démontre notre volonté de transmettre une impression de frontière.

Nous retrouvons d'ailleurs dans la troisième zone, des restes d'un village, un campement de soldats et de survivants s'y trouve mais cela montre bien que les monstres ont pris le contrôle de la zone.

Figure 36 — Petit campement dans la Zone Finale

La dernière zone quand a-t-elle ne comprend pas de village mais quelques mines désaffectées ainsi que des fontaines de magies, on retrouve d'ailleurs un léger changement de paysage autour des fontaines.

Les trois portails présents sur la carte du jeu mènent tous à un combat de boss différent. De plus les zones de boss sont dans le thème des zones précédemment citées et dans l'ordre logique de découverte.

La seule zone de boss n'étant pas en accord avec la zone de jeux dans laquelle le portail ce trouve est la première qui se trouve être designer dans une forêt alors que le portail se trouve dans la zone des marquages.

La téléportation dans les différentes salles de boss se fait sur la même scène à cause de différents conflits ; en effet d'autres personnages non joueurs et immobiles apparaissaient dans la salle du boss quand nous chargions une autre scène.

De nombreux problèmes sont par ailleurs au fur et à mesure que nous décorions la carte du jeu. En effet celle-ci est au fur et à mesure devenue beaucoup trop grande et fournie ; le temps requis pour la charger ainsi que l'espace qu'il fallait via git pour nous la transférer entre développeur il n'était pas assez grand pour permettre le transfert de la carte.

Nous sommes donc tout d'abord passés par une solution plutôt simple : la clé USB ; par la suite nous avons découvert qu'il était possible de former plusieurs préfabriqués avec les différents composants de la carte.

Réduire considérablement l'espace qu'elle prenait dans le projet, nous a également permis de nous la transférer beaucoup plus simplement et rapidement, et enfin, nous a permis de la manipuler plus simplement.

3.7 Tutoriel

En ce qui concerne la phase tutoriel que nous l'avons implémentée pour rendre le jeu beaucoup plus simple à prendre en main. En effet loin d'avoir des touches et fonctionnalités qui sortent de l'ordinaire plus que de raisonnable, il se peut que certaines personnes aient encore du mal à prendre en main ce genre de jeu.

Dans le tutoriel nous pourrons donc retrouver la globale prise en main du personnage et de ses facultés avec évidemment toutes les touches de référencéeun bel sur l'écran mais nous pourrons aussi retrouver un premier combat simple sans aucun mouvement de la part de l'ennemi qui permettra aux joueurs d'effectuer son premier combat sans aucune difficulté.

De plus pour montrer le fonctionnement des coffres malgré sa simplicité, nous avons décidé d'en poser un dans la scène de tutoriel pour familiariser le joueur avec les prochains environnements auxquels il aura accès dans la carte du jeu principal. Une fois le jeu et l'environnement pris en main il suffira donc de cliquer sur fin du tutoriel pour passez au jeux.

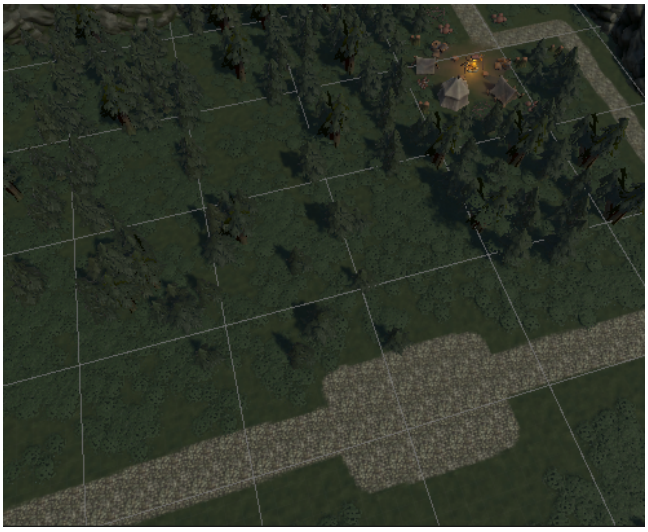


Figure 37 — Tutoriel

Changement d'écran

Après la phase du tutoriel nous nous avons rencontré un problème de chargement vis-à-vis de la prochaine scène.en effet, l'ancienne étant assez imposante il est extrêmement compliqué de passer de la scène du tutoriel à la scène du jeu sans aucun bug visuel.

Finalement nous avons opté pour une image fixe comme nous pouvons retrouver dans beaucoup de jeux vidéos avant le chargement de la phase de la partie.

Cela a tout simplement pour but de fluidifier l'expérience visuelle du joueur sans qu'il le fasse face A quelques problèmes visuels que ce soit dès le début du jeu.

Assets

En ce qui concerne la partie graphique du jeu nous n'avons pas énormément travaillé sur des applications comme blender. n'ayant pas d'expérience dans le 3D ni la volonté de passer plus de temps sur le graphique que sur le code nous avons opté pour l'utilisation assets.

Tous les assets utilisés sont retrouvables sur unity Store. nous avons opté pour utiliser au maximum des assets gratuits que nous pouvions trouver en ligne assez simplement le unity Asset Store présentant un grand nombre de différents assets libre d'accès et donc à disposition pour notre projet.

Malgré cela, il était assez compliqué de trouver des Assets satisfaisantes concernant les décors de La carte du jeu. Et des différentes scènes à disposition du joueur. Dans notre jeu nous pouvons donc retrouver différents types d'assiettes aussi bien dans les décors que dans les ennemis ou encore dans les personnages.

En effet pour simplifier nos actions sur les différents personnages et donc sur les différentes animations il était préférable de prendre des assets Comprenant déjà quelques animations qu'il nous suffisait donc par la suite de simplement agencer entre elles pour fluidifier leur transition.

C'est donc le cas des différents coffres que nous pouvons retrouver sur la carte des personnages principaux des principaux bosses du jeu ainsi que des ennemis que nous pouvons retrouver sur la carte. en ce qui concerne le décor comme dit précédemment il était assez compliqué de trouver un aspect satisfaisant ayant un assez beau rendu sur la carte du jeu.

Nous avons donc opté pour un achat de ces assets nous permettons donc non seulement de gagner du temps vis-à-vis des designs mais aussi d'avoir un rendu plaisant pour l'œil. Évidemment toute la carte du jeu n'est pas entièrement modéliser avec ces assiettes là, nous avons aussi pris d'autres assets comme par

exemple les coffres, les maisons,et quelques autres éléments moins impactants visuellement.

Nous aurions tout aussi bien pu faire un jeu moins beau visuellement et prendre des assets gratuits,Mais notre volonté d'avoir un rendu visuel plus intéressant que de simples polygones nous a poussés à opter pour ce choix.

Dans notre projet les assiettes forment donc le corps du design du jeu ainsi que sa fluidité et sa beauté visuelle.

Par rapport aux animations, comme dit précédemment tous les assets utilisés pour les personnages ainsi que les ennemis ainsi que les éléments du décor ayant besoin d'une animation ont dû être correctement agencé.

En effet bien que les animations Soit fourni avec les assets celles-ci ne sont pas encore agencées pour une autre utilisation du personnage que simplement enchaîner les animations à la chaîne sans but précis. Tout cela se fait donc dans un animator via la plateforme Unity.

Le but de celui-ci est donc de rendre beaucoup plus fluide la transition entre les différentes animations suivant l'utilisation que le joueur fait du personnage.en l'occurrence pour ce qui est du personnage chaque action est définie par rapport à ce que le joueur c'est sur son clavier.

Chacune des transitions relie 2 animations entre elles et peut ou non être activé à l'aide d'un booléen. Cela permet donc soit d'attendre une action pour faire une transition d'animation comme par exemple avancée ou frappée, mais cela permet aussi de revenir à un état neutre comme par exemple à la fin d'une attaque. En ce qui concerne les coffres, c'est la présence du joueur qui déclenche le booléen est donc la transition.

enfin pour les boss tout est directement lié à l'intelligence artificielle qui en fonction des actions qu'elle effectue lance différentes transitions permettant ainsi la fluidité du mouvement de notre ennemi.

Bien évidemment tout ne se résume pas aux transitions mais cela reste la principale chose a instancié si on veut que le personnage réponde aux différentes actions du joueur, évidemment de même pour les ennemis qu'offres et autres les objets qui nécessitent des animations et des interactions avec le joueur.

3.8 Les musiques

En ce qui concerne les musiques nous avons fait appel à une connaissance d'un membre du groupe. Pour renforcer l'originalité de notre jeu.En effet il nous

a semblé moins intéressant de prendre des musiques libres de droit déjà existantes ou une musique simple comme on peut retrouver dans beaucoup de jeux.

Nous avons donc plusieurs musiques. Chacune d'entre elles peut correspondre à une zone, un combat, un combat de boss. Cela permet de renforcer l'immersion du joueur dans le jeu, et bien évidemment, cela permet que le joueur ne se lasse pas d'une musique tournant en boucle pendant tout le jeu quelles que soient les actions qui l'entreprend. De plus notre jeu étant un jeu basé sur le rythme, la musique était un point clé de celui-ci.

3.9 Gameplay

Immersion du joueur :

Afin que jouer soit une expérience agréable pour l'utilisateur, nous nous sommes posés beaucoup de questions sur la façon dont le joueur percevrait le jeu et comment nous pouvions rendre son évolution dans le jeu aussi agréable que possible.

Tout d'abords nous n'avons rien réinventé en termes de caméra, nous avons décidé de partir sur une caméra unidirectionnelle centré en permanence sur le joueur fortement inspiré des mobas/rpg en troisième personne. Ce choix nous permet de rester classique et de ne pas sortir le joueur de sa zone de confort dès le début du jeu. Mais ce choix apporte son lot de problème : l'orientation forcé de la caméra nous demande d'adapter le reste du jeu en conséquence, les objets du jeu doivent être pensé en gardant cette contrainte en tête, le jeu est très linéaire car les centres d'intérêts doivent toujours être présentés le mieux possible à la caméra et si possible en haut du champ de vision de cette dernière étant donné que c'est l'endroit avec le plus d'espace visible.

Les objets flottants tel que les bars de vie doivent toujours faire face à la caméra afin de rester visible.

Figure 38 — Image orientation caméra

Les déplacements sont aussi pensés pour être simple et intuitifs, il ne s'agit pas de réinventer la roue ici ;
Un déplacement sur huit axes contrôlable grâce à 4 touches de déplacement 'wasd' classiques combinables pour marcher en diagonal. Le joueur peut avancer à deux vitesse différentes allures :

- Une vitesse de marche relativement lente afin de permettre au joueur de bien contrôler ses mouvements en combat
- Une vitesse de course 2 à 3 fois supérieure à celle de marche pour les déplacements entre les différents points d'intérêts.

Des animations différentes sont disponibles afin de bien faire la différence entre ses deux allures.

Afin de rendre la mobilité plus satisfaisante, le joueur peux faire une roulade pour mettre les ennemies à distance durant un bref instant.

3.9.1 Personnage et progression

Pour que le jeu propose une expérience diversifiée, 4 personnages différents sont jouables avec des utilités différentes et une approche différente. Malgré ça, les joueurs ont une base commune et des systèmes communs :

- Toutes les statistiques sont les mêmes (point de vie, dégâts de base, vitesse)
- Tous les personnages ont un système d'expérience et d'amélioration des sorts identiques ainsi que le système de combo qui est l'un des points centraux de notre jeu.

3.9.2 Statistiques de base :

Les personnages apparaissent tous avec 100 points de vie et 10 points de dégâts au début de la partie, ces statistiques sont améliorables au fil de la partie via un système d arbre de compétence et certaines classes ont des capacités augmentant ces chiffres.

3.9.3 Point d'expérience :

Le jeu à un système d'expérience qui accompagne le joueur dans toute sa progression, il dictera l'avancement du joueur et récompensera le joueur pour toutes ses actions.

Chaque ennemi tué donnera des points d'expériences qui une fois un palier passer augmentera le joueur de niveau et lui donnera une récompense. Les points d'expérience requis augmentent à chaque palier passé selon la formule suivante :

Soit Xn l'expérience requise pour passer de niveau au niveau n, Xn+1 = 4/3Xn

Le niveau est affiché en permanence sur l'Hud pour que le joueur puisse garder un œil dessus en permanence.

3.9.4 Points de compétence :

Ces points sont gagnés à chaque montée de niveau et servent à améliorer les compétences du joueur.

Comme ces points sont lie à l'expérience et que les améliorations de haut niveau coutent beaucoup plus chère, ces points seront plus durs à avoir et plus important en fin de partie et une augmentation de compétence peux changer drastiquement la puissance d'un personnage.

3.9.5 Compétences du joueur :

Chaque joueur a 2 compétences :

- Une compétence de dégâts pour aider le joueur à tuer les monstres.
- Une compétence utilitaire se présentant généralement comme une augmentation temporaire des statistiques du personnage.

Ces compétences sont toutes améliorables via deux des trois branches de l'arbre de compétence.

3.9.6 Arbre de compétence :

Cet objet est au centre de la progression du personnage et est l'objet qui donne sa diversité de gameplay au jeu.

Il est composé de trois branche :

- Une pour le sort un.
- Une pour le sort deux.
- Une branche de passifs et d'augmentation de statistique de base commune à tous les personnages

Monter une branche au niveau maximum coute 15 points de compétence et le joueur n'en gagnera que 30 au cours de la partie, il sera donc amené à faire des choix quand a la spécificité de son personnage.

Les couts pour les améliorations sont repartis sur une branche de la manière suivante : 1-2-3-4-5. Le joueur peut donc choisir jusqu'à quel niveau il souhaite monter ses compétences sachant que ces derniers changes fortement après le niveau 3.

Branche centrale :

Cette branche est commune à tous les personnages et augmente de manière permanente les statistiques du personnage. Les améliorations sont reparties comme si dessous :

1. Augmentation des pv 100 ¿¿ 120
2. Augmentation des dégâts 10 ¿¿ 15
3. Régénérations passive de 3pv toutes les 5 secondes
4. Augmentation des pv 120 ¿¿ 150
5. Augmentation des dégâts 15 ¿¿ 30

Certaines de ses améliorations sont obligatoire pour certains personnages comme le tank et l'assassin qui n'ayant pas de régénération dans leurs sorts n'ont que les feux de camp pour se régénérer ou le dps qui cherchera à obtenir le plus de dégâts de base possible pour mieux profiter de ses sorts.

3.9.7 Ciblage automatique :

Les sort et attaques de base sont tous ciblé grâce à un système simple : l'ennemi le plus proche est ciblé par le personnage. Les sorts sont donc directement générés sur l'ennemi et les dégâts lui sont directement appliqué.

Nous ne voulions pas que la difficulté du jeu soit basée sur des sorts dur a

toucher ou des combinaisons de touches complexes pour maximiser ses dégâts mais sur le système de rythme et l'amélioration des sorts : pour vaincre un ennemi il faut soit être plus fort, soi être plus nombreux, ce qui évite une durée de vie trop courte pour les bon joueur.

3.9.8 Les personnages

Le jeu présente 4 classes jouables : - Un Mage - Un Dps - Un Assassin - Un tank

Figure 39 — Les 4 personnages

Ces 4 classes sont basées sur 4 archétypes des rpg occidentaux et servent tous un but bien précis.

Le Mage :

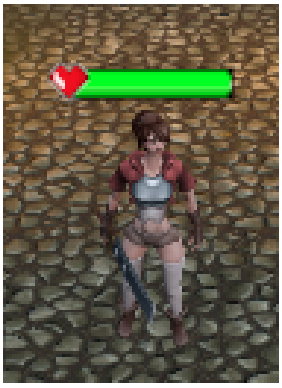


Figure 40 — Mage

Le Mage est la seul classe disposant de sort de zone et a pour but de soutenir l'équipe lors d'un combat de boss. Elle est très forte en début de partie et arrivera très vite à tuer les premiers ennemis.

Compétence 1 : le soin.

Une compétence de soin régénérant 20 pv

Améliorations :
1. Soin 20 ¿¿ 30
2. Temps de recharge 6s ¿¿ 4s
3. Soin de zone
4. Soin 30 ¿¿ 40
5. Soin en 3 temps 40 ¿¿ 3x20'

Figure 41 — Soin de zone

Compétence 2 : Eboulement.

Compétence de dégâts de zone infligeant 20 dégâts.

Figure 42 — Eboulement

Améliorations :
1. Temps de recharge 8s ¿¿ 6s
2. Dégâts 20 ¿¿ 30
3. Eboulement ¿¿ Givre (le sort ralenti la cible)
4. Temps de recharge 6s ¿¿ 4s
5. Dégâts 30 ¿¿ 40

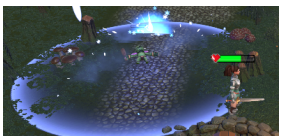


Figure 43 — Chute de givre

Le Dps :

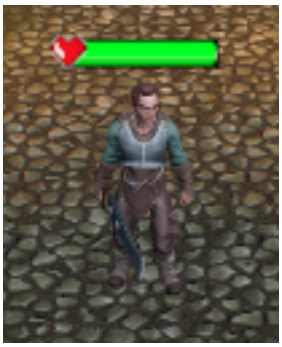


Figure 44 — DPS

Le dps est une classe pensée pour avoir le plus de dégâts possible il peut s'orienter soit vers des dégâts constants soit sur des grosses sommes de dégâts infligées sur un court temps.

Compétence 1 : Augmentation de dégâts.

Une compétence augmentant les dégâts de base durant 4s.

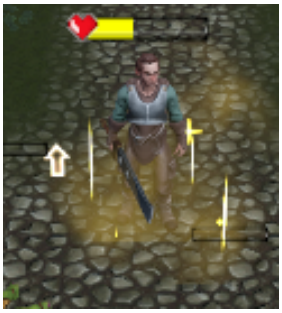


Figure 45 — Dégâts +

Améliorations :
1. Boost 1.5 ¿¿ 2
2. Temps de recharge 15s ¿¿ 13s
3. Durée 4s ¿¿ 7s
4. Boost 2 ¿¿ 2.5
5. Temps de recharge 13s ¿¿ 11s

Compétence 2 : Coup puissant.

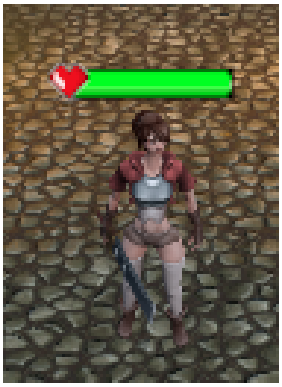
Reset d'animation d'attaque de base qui inflige des dégâts supplémentaires.

Figure 46 — Coup puissant

Améliorations :
1. Dégâts 10 ¿¿ 30
2. Dégâts 20 ¿¿ 30
3. Temps de recharge 6s ¿¿ 4s
4. Dégâts 30 ¿¿ 45
5. Régénère 15pv quand touche

L'Assassin :

L'Assassin est une classe très mobile capable d'infliger beaucoup de dégâts.



Compétence 1 : Augmentation de Vitesse.

Une compétence augmentant la vitesse de déplacement pendant 4s.

Figure 47 — Vitesse +

Améliorations :
1. Durée 4s ¿¿ 7s
2. Temps de recharge 15s ¿¿ 13s
3. Boost + 0.75 ¿¿ + 1.5
4. Durée 7s ¿¿ 10s
5. Temps de recharge 13s ¿¿ 11s

Compétence 2 : Téléportation.

Compétence téléportant le joueur derrière un ennemi.

Figure 48 — Téléportation

Améliorations :
1. Dégâts 0 ¿¿ 10
2. Temps de recharge 6s ¿¿ 5s
3. Portée 5 ¿¿ 9
4. Temps de recharge 5s ¿¿ 3s
5. Dégâts 10 ¿¿ 30

Le Tank :

Le Tank est pensé pour être la classe la plus simple du jeu. Si améliorer d'une certaine manière elle peut être temporairement immortel.

Figure 48 — Résistance +

Améliorations :
1. Réduction 10 ¿¿ 30

2. Réduction 30 ¿¿ 50
3. Réduction 50 ¿¿ 70
4. Réduction 70 ¿¿ 90
5. Réduction 90 ¿¿ immortel

Compétence 2 : Epines.

Renvois une partie des dégâts subis (passif).

Figure 48 — Epines

Améliorations :
1. Renvois 0% ¿¿ 10%
2. Renvois 10% ¿¿ 20%
3. Renvois 20% ¿¿ 30%
4. Renvois 30% ¿¿ 40%
5. Renvois 40% ¿¿ 50%

4 Poins négatifs

Durant la réalisation du projet de second semestre, l'équipe GAFAM Eaters peut rencontrer de nombreux problèmes internes alors que nous découvrions le travail d'équipe.

La création de l'équipe n'était pas évidente, car Louis et Antoine se connaissaient très bien, Abel qui était dans leur classe a ensuite été intégré à l'équipe, et Mathis a été ajouté car il était une connaissance d'Antoine sur *Discord*. Il peut donc être difficile d'exiger du travail de la part des autres quand on ne les connaît pas très bien. De plus le fait que toute l'équipe ne soit pas dans la même classe rendait la communication au sein de l'équipe encore plus difficile car il était très rare que toute l'équipe se voit en même temps en physique.

Pour remédier à ces problèmes, nous avons commencé par programmer des réunions en physique de manière hebdomadaire, mais nous avons mis du temps trop de temps à réellement créer le projet. En effet, Mathis qui était chargé créer toute la partie graphique du jeu (poser des textures, des arbres, des chemins, etc.) a été pour le reste de l'équipe une occasion de ne rien faire, donc nous avons commencé toute la partie programmation qu'à partir de février.
La partie programmation n'ayant pas besoin de la partie graphique pour être développée, nous aurions dut commencer à partir de décembre afin d'éviter le rush d'avant soutenance.

Heureusement, aujourd'hui nous nous connaissons mieux, ce qui nous permet de nous répartir plus facilement le travail sans avoir à forcément nous réunir tout le temps

Nous voulons tout de même aborder le problème le plus important que nous avons rencontré tout au long de ce projet. Il s'agit des tensions entres les membres qui sont apparues tout au long du projet. Nous n'avons pas su trouver vraiment de solutions durable, ce qui est bien dommage. Ce que nous voulons retenir est qu'il est important de compter les uns sur les autres, et d'adopter un managment au sein d'une équipe de travail basée sur la confiance. Si un membre de l'équipe ne travaille pas, inutile de venir l'opresser, il s'en rendra compte de lui même car nous voulions tous la réussite du projet.

Pour finir, nous avons perdu plusieurs fois la motivations, en nous rendant compte que notre jeu ne serait pas assez bien pour concurencer des jeux mondialement connus (ce qui est évident, mais ne l'était pas pour nous en décembre dernier).

Voilà une liste non exhaustive des points où l'équipe GAFAM Eaters a amélioré ou aimerait améliorer.

Conclusion

Pour conclure, ce projet nous fait vivre une très grande aventure : du code en langage C#, aux relations humaines en passant par des *merge conflicts*, nous sommes parés pour l'avenir !

Nous sommes une équipe bien unie et toujours positive qui saura toujours résoudre tout type de problèmes !

Ce projet de jeu vidéo RPG en jeu de rythme nous tous absorbé pendant tout le semestre, et nous en sommes très fiers.

C'est ainsi que s'achève notre projet de semestre deux à l'Epita. Pensez à bien suivre l'actualité du jeu sur senzo-no-michi.github.io !