GAFAM EATERS

1ER RAPPORT DE SOUTENANCE

# Jeu vidéo - Senso No Michi


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


PROMO 2027

15 Décembre 2022 — 10 Mars 2023

# Table des matières

# 1 Introduction

Vous trouverez dans ce premier rapport de soutenance les premières réalisations de l'équipe GAFAM Eaters dans son projet de jeu vidéo.

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.

Nous avons pris la liberté de modifier quelques éléments de ce cahier des charges au moment 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, à une différence près : nous avons abandonné le côté "légende chinoise". 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

Les tâches ont été réadaptés pour qu'une chose ne soit pas faite en même temps par deux personnes, et par rapport aux disponibilités de chacun.

| | 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 |
| Levels Designs | Suppléant | Délégué | | |
| Menus | Délégué | Suppléant | | |
| Site Internet | Délégué | | | Suppléant |
| Intelligence Artificielle | Suppléant | | Délégué | |
| Multijoueur | Suppléant | | Délégué | |

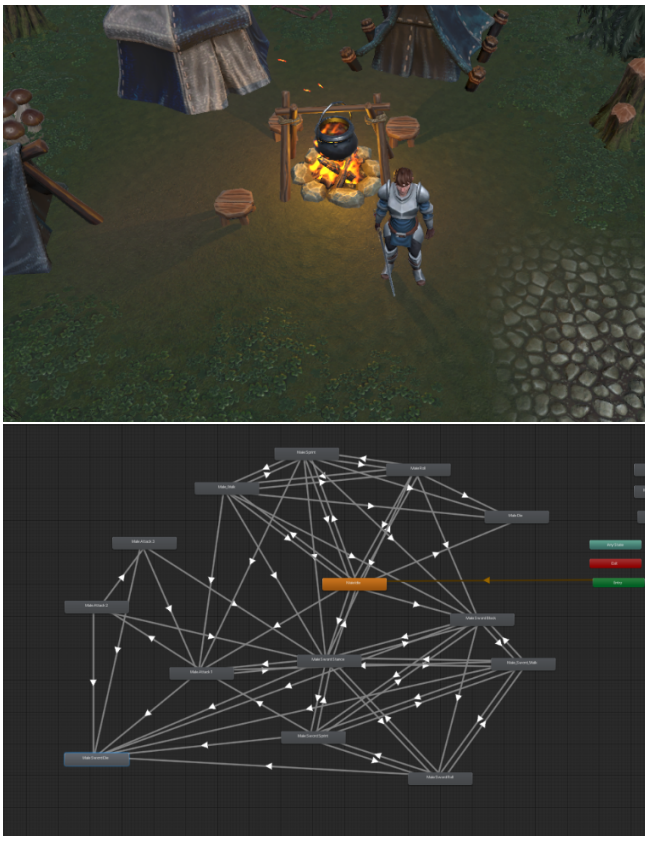
## 2.4 Avancement par soutenances

| | Soutenance 1 | Soutenance 2 | Soutenance 3 |
|---|---|---|---|
| Modelage 3D et animation | 60% | 80% | 100% |
| Musique et sons | 60% | 100% | 100% |
| Gameplay | 65% | 85% | 100% |
| Levels Designs | 60% | 80% | 100% |
| Menus | 65% | 90% | 100% |
| Site Internet | 45% | 70% | 100% |
| Intelligence Artificielle | 30% | 75% | 100% |
| Multijoueur | 70% | 90% | 100% |

# 3 Réalisation techique

## 3.1 Modelage 3D, animation et level design

Le Modelage 3D représente toute l'organisation de la partie graphique. De la Carte du jeu, ses différents décors ; mais aussi les différentes scènes comme la salle du boss ou encore la scène utilisée comme fond pour le menu.



Quant aux animations, elles reposent sur un “Animator”, chacune des animations sont reliées à d’autres animations. Les transitions, elles, reposent sur des booléens qui indiquent les demandes du joueur. Toute la partie transition et création des booléens s’effectue sur Unity. Le personnage est ensuite lié à un script ; en fonction des touches implémentées, le joueur peut alors librement déplacer le personnage.

Le pilier principal de cette tâche étant une partie du code correspondant au gameplay, les difficultés rencontrées durant cette tâche ont été dépendantes des animations déjà présentes sur les personnages choisis. En effet, le personnage choisit pour les essais avait des animations en place : celles-ci n’engendraient aucun déplacement du personnage ; toutefois, les animations du personnage choisit pour être notre joueur principal était différentes.

Le code implémenté faisait donc continuellement changer la position du personnage même quand il devait être immobile (si on appuyait sur les bonnes touches). Le fait est qu’en fonction du personnage joueur, il faille uniquement changer l’axe de rotation pour que le personnage puisse tourner, et pas l’axe de translation, pour que celui-ci ne change pas de position lorsqu’il devrait être

immobile.

De même avec les personnages non joueurs.

Un second problème que nous n'avons pas encore pu régler est un souci concernant les combos d'attaque du personnage principal. En effet celui-ci dispose de trois attaques différentes et donc de trois animations, la partie script assignée cherche à régler ce combo en fonction du nombre d'attaque effectuer par le joueur.

Une solution à ce problème pourrait être d'implémenter un temps d'attente après chaque attaque permettant alors à chaque animation de se terminer correctement avant d'autoriser une seconde attaque. Une seconde façon de régler cela pourrait être d'implémenter deux booléens supplémentaires pour différencier chaque attaque, puis d'implémenter un compteur qui augmente à chaque attaque effectuée et pour finir, de lancer l'animation en fonction de cette attaque.

## 3.2 Gameplay

### 3.2.1 Orientation de la caméra

Pour la caméra, on a choisi une vue en troisième personne assez large pour permettre au joueur de bien voir son environnement. La camera suit le joueur en gardant une orientation fixe pour un meilleur repérage dans l'espace un peu à la manière de vieux MMORPG ou d'un Moba.

Figure 1 – Orientation de la caméra

### 3.2.2 Déplacement du personnage

La méthode qui nous semblait la plus efficace pour implémenter correctement les déplacements de nos héros consiste en un déplacement sur 8 axes rappelant divers vieux jeux sur console comme les premiers jeux des plus grande franchise *Nintendo* par exemple. On utilise pour cela la classe *UnityEngine.InputLegacyModule* qui permet de récupérer sous forme de valeurs numériques une intensité de déplacement par axes ce qui correspond parfaitement à ce que l'on souhaite faire et nous permettra potentiellement par la suite d'utiliser une manette comme alternative au clavier.

De plus nous avons laissé la possibilité au joueur de courir permettant ainsi de réduire les temps de déplacement sur la carte bien que l'utilité de cette fonctionnalité soit débattable.

### 3.2.3 Le Rythme

Le rythme est l'un des élément si ce n'est l'élément le plus important du gameplay, il est notre façon de proposer un gameplay peu commun et amusant tout en apportant une touche personnel au projet (Abel, chargé de l'implémentation de ce dernier, étant un grand fan de différents jeux de rythmes). L'utilisation de ce dernier est semblable à celle présente dans le jeu Bullet Per Minute bien que ce dernier soit un FPS *Rogue-Like* : des musiques sont jouées en fond et le joueur doit faire les actions autres que les déplacements comme jeter des sorts en rythme avec celles-ci sous peine de voir l'action annulée.

Néanmoins, l'originalité de ce système est la cause principale de la difficulté de son implémentation : la version implémenté actuellement utilisant la méthode *Update()* présente dans la classe *Monobehaviour* de Unity présente de majeurs problèmes et peut sauter des battements. Une autre version a été tenté utilisant les méthodes de la classe *Time* mais sans succès. Une autre méthode à tester passe par un objet dans l'*HUD* afin d'implémenter le rythme de manière similaire aux jeux de rythmes plus classique en observant le rapprochement entre une note et la ligne ou elle doit être pressé à l'aide de collision.

### 3.2.4 Le Combo

Mécanique découlant directement du rythme récompensant les diverses actions effectuées par le joueur en rythme grâce à un compteur qui, passé un certain nombre atteint des paliers confèrent des bonus au joueur tel qu'une augmentation des dégâts ou de la résistance ou une réduction des temps de recharges des actions.

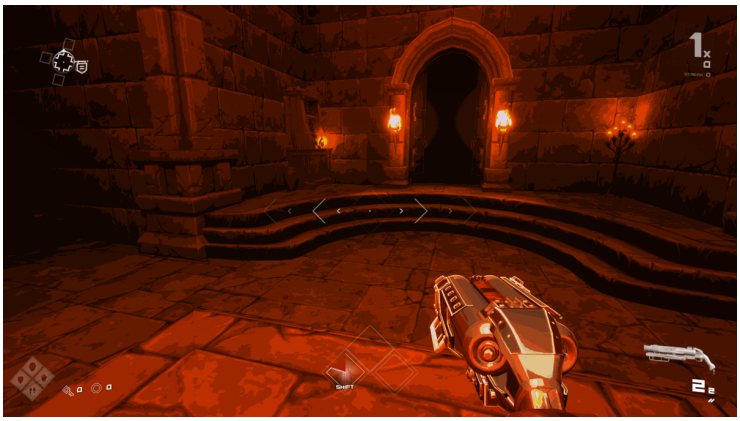


Figure 2 – BPM *Bullet Per Minut*

Néanmoins le combo a aussi la charge de punir les actions ratées par le joueur en diminuant d'un palier si le joueur prend des dégâts ou en se réinitialisant si le joueur tente de faire une action qui n'es pas en rythme. Le combo sera ajouté à l'*HUD* permettant au joueur de savoir en permanence ou il en est.

### 3.2.5 L'expérience

Un système de gain d'expérience est présent dans le jeu quand le joueur tue un ennemi afin de débloquer des compétences passives, des augmentations de statistiques ou des changement de certaines mécaniques principal du jeu tel que le combo, la roulade ou encore les sorts.

Cela se matérialise par la présence d'un menu spécial présentant une forme d'arbre de compétence déblocable avec des points obtenus à chaque monté de niveau.

### 3.2.6 La roulade

Une implémentation supplémentaire touchant les déplacements du personnage mais celle-là se fait en rythme, une roulade semblait être la façon parfaite de rajouter un peu de nervosité au jeu sans trop casser la fantaisie du jeu simple avec peu de contrôle différents.

Elle permettra à terme de se replacer, éviter un sort ou potentiellement annuler des animations afin de rendre le jeu plus fluide et vif.

#### 3.2.7 Les classes de personnages

Ces dernières ne sont pas encore vraiment implémenté car a ce jour une seule classe est jouable et elle n'est clairement pas aboutie, mais l'objectif sera à terme de proposer au joueurs trois classes différentes par leurs statistiques, voix d'amélioration et par leurs compétences.

#### 3.2.8 Les compétences

2 par classes, 6 en tout, elles seront l'identité de leur classes respectives. Tantôt orienté support ou dégâts elle permettront aux joueurs de s'exprimer pendant la bataille. Etant donné que ses compétences ne sont pas interchangeable les implémenter grâce à la classe Scriptable Objects ne semblait pas nécessaire c'est donc toujours la classe MonoBehaviour qui sera utilisé ici.

Les compétences sont soit lancé sur l'ennemi le plus proche à portée ciblé au préalable s'il s'agit d'une compétence offensive ou directement lancé sur le joueur dans le cas échéant, évitant ainsi une utilisation trop compliqué grâce a un système de ciblage ou de missile.‘

## 3.3 Les menus

### 3.3.1 Menu principal

Pour commencer, lorsque le jeu est lancé, on tombe sur le menu principal.


Le menu principal a été créé par Mathis Galliano, son but est simple : permettre au joueur de lancer la partie.

Le fond est une scène 3d et donc animée pour plus d'immersion dans le jeu. Chaque bouton est relié à un mode de jeu, le mode de jeu solo et le multijoueur. Il reste toutefois à implémenter le menu des options qui permettra potentiellement au joueur de changer quelques spécificités du jeu.

### 3.3.2 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 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 déplacer dans le monde en appuient sur ses les touches de déplacement.


Figure 3 – Aperçu du menu carte

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 disparaitre 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.

mapCamera.enabled = false

Plusieurs difficultés ont été rencontrées lors de l'implémentation de la carte, notamment lors de l'implémentation en multijoueur. En effet, lorsqu'un joueur ouvrait sa carte, la carte de chaque joueur s'ouvrait. C'est Antoine qui a résolu le problème en créant une carte avec une caméra par joueur, et en supprimant les caméras des autres joueurs à chaque joueur à l'aide de l'asset "photon". Ensuite, aucun bouton qui permettrait d'afficher la position exacte n'a encore été ajouté pour pouvoir simplifier l'adaptation au multijoueur. Le joueur peut aussi faire sortir la caméra de la carte.

Une seconde version de la carte avec une limite adaptative en fonction du zoom et un curseur pour la position du joueur devront donc être implémentés d'ici la prochaine soutenance.

Figure 4 – Aperçu de la mini-carte

### 3.3.3 La mini-carte

Il permet au joueur d'afficher un aperçu de ce qui l'entoure à tout moment. Pour cela, comme pour le menu carte principal, il y a une caméra orthographique orientée vers le sol qui suit la position du personnage.

Aucun curseur n'a encore été créé, nous en avons mis un par défaut, mais il sera fait designé pour la prochaine soutenance.

### 3.3.4 Le menu inventaire

Il a pour objectif d'afficher en haut à gauche un aperçu du joueur, en bas à gauches quelques statistiques du joueur (dégâts infligés, km parcourus, etc...). L'écran de droite doit afficher l'arbre des compétences.

Pour l'instant, la partie aperçu du personnages et statistiques n'ont pas du tout été implémentées, mais la partie arbre l'a été.

Toutes les cases de l'arbre ont un nom qui est adapté en fonction du type de joueur (Mage / DPS / Tank) à l'aide de trois string listes.

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.

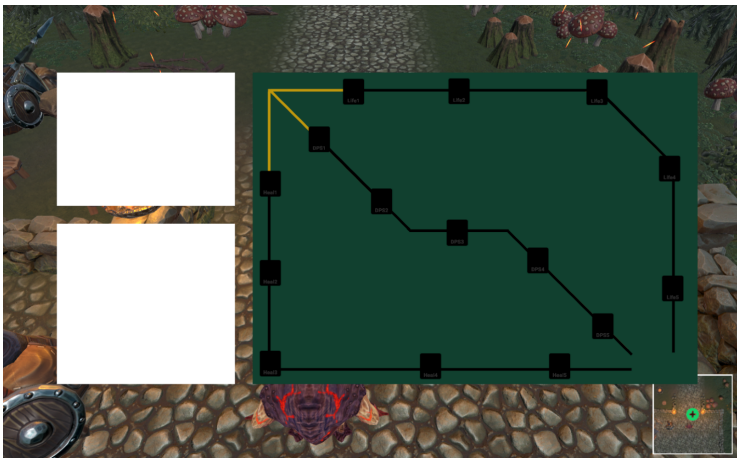


Figure 5 – Aperçu du menu inventaire en début de partie

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 en appellant *Actualize()*.

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 pourrait 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 est 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 déciderons de la solution pour la prochaine soutenance, et nous implémenterons les menus de gauche.

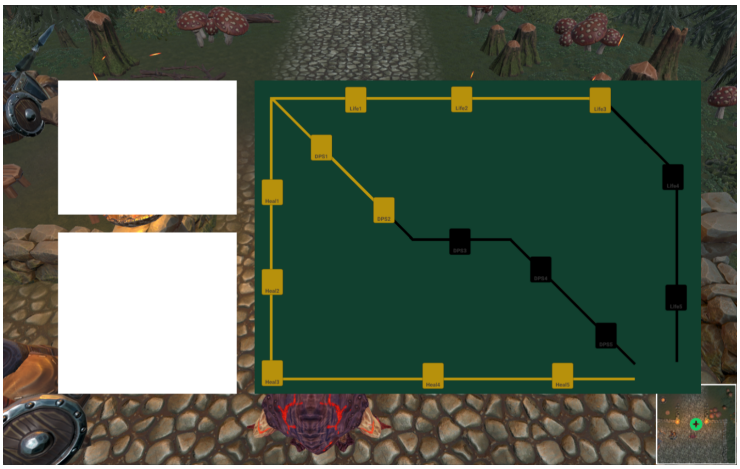


Figure 6 – Aperçu du menu inventaire en début de partie

## 3.4 Site Internet

Le design et la création du site web de présentation a été fait par les soins de Louis Rodet, en en ajoutant un bouton télécharger qui est bien en valeur et une barre de menu qui redirigera vers plusieurs pages.

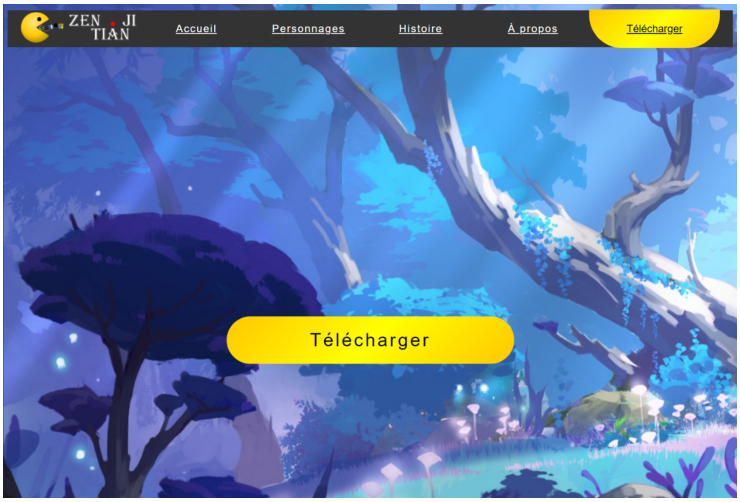



Figure 7 – Page d'accueil du site internet

La page d'accueil pourra accueillir également les "Actualités du jeu" mais pour l'instant le contenu n'a pas été développé.

La page d'accueil a été faite, et il y aura également une page de présentation des personnages, de l'histoire, et une page "à propos" avec une présentation de l'équipe GAFAM Eaters, et un lien vers Epita.

Nous avons rencontré des difficultés pour rendre le site web responsive notamment pour la version mobile où le menu s'affichait très mal.

Le site sera fini et mis en ligne sur http://senso-no-michi.free.fr. Nous réserverons le nom de domaine avec *free*. Il sera fini pour la troisième soutenance.

## 3.5 Intelligence artificielle

Tout d'abord lorsque nous voulons faire une intelligence artificielle pour un personnage, ou un mob, il faut lui donner des zones qu'il a le droit d'explorer. Pour éviter par exemple qu'il puisse marcher dans une maison.

Voici une partie des zones explorables par notre IA du golem :

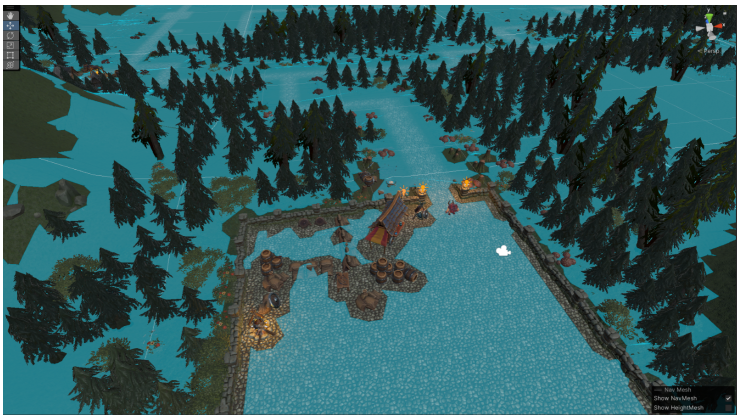


Figure 8 – Aperçu du menu inventaire en début de partie

Le golem a un fonctionnement simple :
— Si le joueur le plus proche du golem est trop loin, le golem se déplace aléatoirement.
— Si le joueur le plus proche du golem est suffisamment proche du golem et qu'ils ne sont pas collés, le golem peut prendre en chasse le joueur le plus proche.

— Si le joueur le plus proche du golem est suffisamment proche du golem et qu'ils sont collés, le golem peut infliger des dégâts au joueur le plus proche.

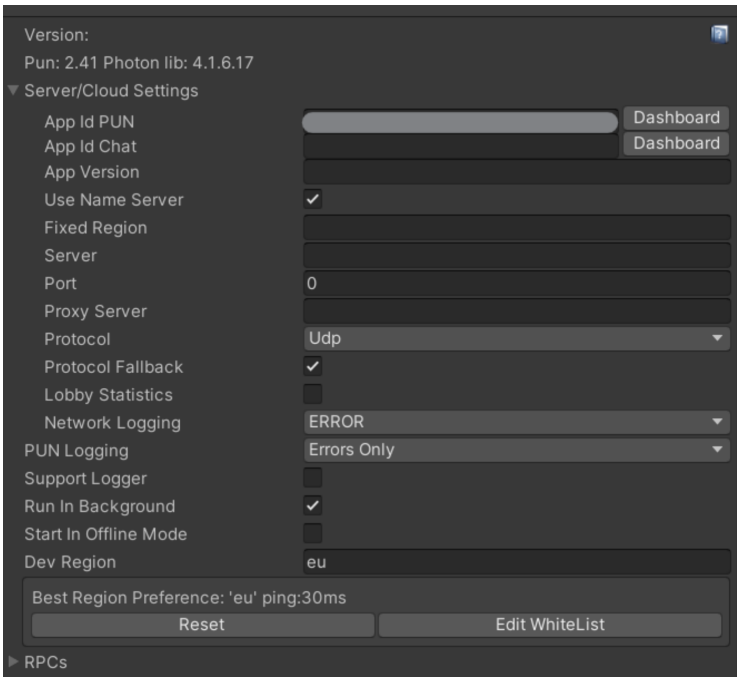
Pour la prochaine soutenance, nous aimerions par la suite implémenter des mécaniques plus complexes pour les boss comme des attaques spéciales, des combos etc... Ce qui donnerait plus d'attrait au jeu.

## 3.6 Multijoueur

La bibliothèque *Photon PUN* qui est l'une des meilleures solutions pour notre projet, car il nous permet de mettre en relation plusieurs clients sur un serveur distant.

Nous avons donc dû créer un compte sur photon et crée une application pour avoir une App id. Cette App ID permet à chaque client la possédant de se connecter à une Scène en ligne.

Voici les paramètres utilisés (par défaut pour le moment) :




Le multijoueur à un fonctionnement très simple. Lorsque nous voulons implémenter un mode multijoueur il est préférable d'avoir un *lobby*, dans lequel nous pouvons

choisir les différents types de mode de jeux, donc le multijoueur.

Le script *MyLauncher* permet de gérer toutes les connexions aux *Room*, une *Room* c'est comme une scène mais mise en ligne sur serveur distant. Ce script créé une *Room* si personne n'est déjà connecté, sinon il la rejoint. Le premier connecté sera maître de la *Room*, Le client maître n'est ni hôte, ni un serveur, il a juste quelques privilèges en plus. Ce script permet aussi de faire des retours de *log* avec des fonctions d'évènements comme *OnJoinedRoom*.

Nous avons rencontré plusieurs problèmes. Premièrement il fallait savoir comment gérer le fait de faire apparaitre plusieurs objets *Player* sur notre Scène. Ce problème a été résolu grâce à l'objet *GameManager* qui est par défaut sur la Scène. Il permet d'instancier dans la *Room* avec la méthode *PhotonNetwork.Instantiate()* un objet qui se trouve dans le dossier Ressources. On a donc dû attacher tous les menus (la carte, la mini-carte, le menu des compétences). Mais cela a entraîné un nouveau problème, c'est des conflits entre plusieurs *UI*. La solution trouvée est juste de supprimer les éléments de l'*UI* s'ils n'appartiennent pas au joueur local.
Un autre problème était le fait que l'on puisse contrôler les deux joueurs avec un seul client, on a pu faire face à ce problème avec la condition *if (photonView.IsMine)*. Cette fonction *photonView* est très pratique, car chaque élément instancié sur la carte en possède une dans leurs attributs et permet de savoir quel client possède quel élément. Donc le *IsMine* permet de pouvoir diriger que le client actuel.
Le dernier problème était de synchroniser les déplacements des IA entre tous les clients. Pour y remédier on a dû changer la fonction de recherche du joueur par *FindObjectsOfType* de Photon, au lieu de simplement utiliser *Find* de Unity.

# 4 Poins négatifs

Durant la réalisation du projet de second semestre, l'équipe GAFAM Eaters à sut se découvrir dans le travail d'équipe, mais a rencontré plusieurs difficultés.

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.

Ensuite, nous sommes obligés d'évoquer les problèmes techniques qui ne qui ne pourront probablement pas être améliorés pour la prochaine fois. En effet, toute l'équipe a des ordinateurs très puissants pour faire tourner le jeu, excepté Louis. Cela était très gênant car il devait initialement gérer le gameplay, mais son ordinateur mettait parfois une minute à charger les scripts et à compiler. Nous avons donc dû lui donner les menus à faire, ce qui était moins gourmand pour son ordinateur, et Abel l'a remplacé sur le gameplay.

Ce sont sur tous ces points que l'équipe aimerait s'améliorer ou s'est déjà améliorée, pour n'avoir qu'à se concentrer sur le projet, et non sur des problèmes d'organisation !

# 5 Conclusion

Pour conclure, ce projet est une excellente manière d'apprendre à coder en équipe, dans la durée. Certains problèmes complexes parfois encore non résolus nous ont permis de renforcer notre niveau en programmation en c. Nous avons appris à nous répartir équitablement les tâches en nous séparant sur les différents sujets pour ne pas créer de conflits de *merge*.

Pour la prochaine soutenance, nous aimerions implémenter quelques menus, et techniques de gameplay complexes comme le rythme pour que notre jeu ait des mécaniques intéressantes.

C'est ainsi que s'achève la première partie du projet de notre second semestre à l'EPITA. Nous vous tiendrons rapidement au courant des évolutions du projet lors de la seconde soutenance.