Rapport de soutenance
Première soutenance

Enclave

Louis Plaire
louis.plaire@epita.fr

Sacha Skopan
sacha.skopan@epita.fr

Martin De Dorlodot
martin-de-dorlodot@epita.fr

Angel Pasquin
angel.pasquin@epita.fr

Ethan Bordas
ethan.bordas@epita.fr

Aura Corp.

EPITA

Table des matières

1 Introduction

Depuis sa création, l'industrie du jeu vidéo a connu une progression constante, s'établissant comme un véritable pilier de la culture et de l'art. Néanmoins, dans un monde où les grandes sociétés cherchent souvent à favoriser la rentabilité au péril de l'innovation, les jeux indépendants sont apparus comme un renouveau, proposant des expériences variées et profondément personnelles. Aura Corp., une entreprise à la vision innovante, se démarque dans ce cadre par son aspiration à allier l'art et le divertissement. Enclave, notre projet majeur, illustre parfaitement cette ambition.

Enclave dépasse de loin le statut d'un simple jeu. C'est une expérience immersive et méthodique qui trouve son inspiration dans des jeux vidéos tels que Dead Space pour son ambiance captivante, Rogue pour ses systèmes de mort permanente, et The Escapists pour son style rétro en pixel art. Ce jeu de style rogue-lite place le joueur dans un univers où chaque choix est crucial et où la survie dépend autant de la détermination et de la stratégie que sur la capacité à improviser face à des défis imprévus.

Dans Enclave, le joueur incarne un prisonnier retenu dans un vaisseau spatial transformé en arène mortelle. Son objectif : atteindre le trentième niveau pour s'échapper et retourner sur sa planète d'origine, Marvin-IV. Tout au long de cette aventure, il devra faire face à des hordes de créatures extraterrestres, récolter des ressources limitées et surmonter des niveaux de plus en plus dur. Chaque salle regorge des pièges, des ennemis redoutables et des événements aléatoires qui garantissent une capacité de rejouer presque sans fin.

L'équipe d'Aura Corp. a conçu Enclave comme un hommage aux jeux qui ont su marquer les esprits par leur profondeur et leur originalité, tout en intégrant une touche de nouveauté propre à leur vision. Le résultat est une expérience divertissante et artistique où l'exploration, le danger constant et une gestion stratégique des ressources jouent une place primordial.

Fondée en 2037, Aura Corp. est une entreprise qui repose sur un principe simple : créer des jeux auxquels ses membres aimeraient jouer par eux-mêmes. Ce groupe de passionnés rassemble des talents divers, allant de la programmation au design visuel en passant par le storytelling et la gestion de projet. Chaque membre apporte son savoir-faire unique, enrichissant ainsi chaque aspect des jeux qu'ils développent.

Chaque détail dans notre jeux est soigneusement élaboré pour offrir une expérience mémorable, immersive et qui défie les normes établies. Les membres d'Aura Corp. partagent une philosophie commune : l'art et le plaisir de jouer ne doivent jamais être être renoncé pour des raisons seulement commerciales. Parmi eux, Louis Plaire, chef de projet, apporte son expérience en gestion . Sacha Skopan, responsable du level design, combine ses compétences en création narrative et en systèmes embarqués pour imaginer des niveaux aussi captivants que stimulants. Angel Pasquin, designer visuel et sonore, apporte une énergie unique à chaque personnage et ambiance du jeu grâce à son passé d'artiste et de musicien.Cette synergie entre les différents talents de l'équipe permet à Aura Corp de créer des œuvres originales. Avec Enclave, Aura Corp ambitionne de transformer le genre du rogue-lite en proposant une approche innovante où chaque partie raconte une histoire différente. En redonnant une place importante à l'exploration et à la narration, tout en maintenant un gameplay exigeant, l'équipe espère inspirer une nouvelle génération de joueurs et de développeurs.

Dans ce rapport, nous aborderons les progrès réalisés par chacun sur Enclave. aborderons le site internet et l'intelligence artificielle développés par Louis. Puis, le multijoueur et le level design élaborés par Sacha. Ensuite, nous étudierons le système d'arme réalisé par Martin. Après cela, nous détailleront le système d'interface utilisateur par Ethan. Enfin, le joueur et le pixel art supervisés par Angel.

2 Etat de l'avancement

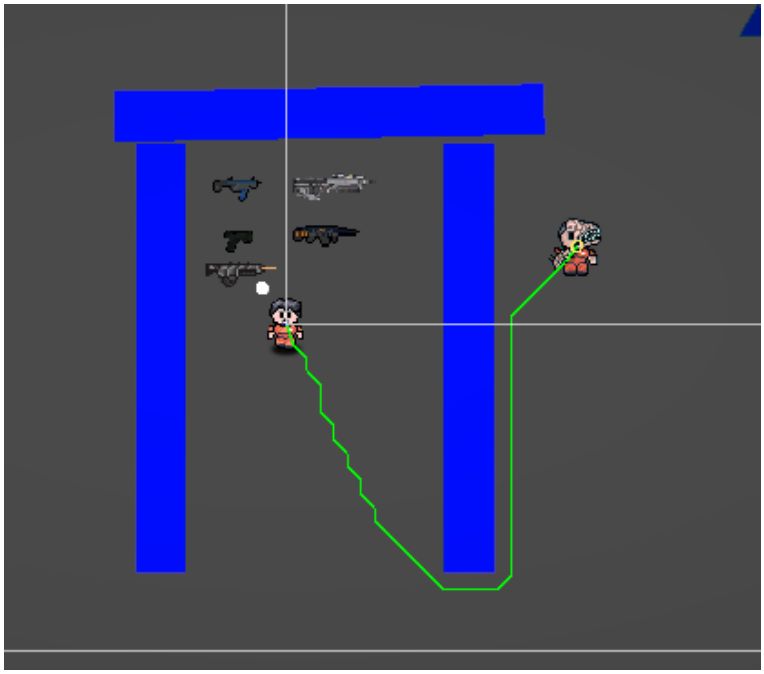
2.1 Louis Plaire

2.1.1 Intelligence Artificielle

L'intelligence artificielle (IA) est au cœur du gameplay d'Enclave. En effet, elle représente la difficulté qui se dresse entre le joueur et la victoire... Elle a pour but de l'éliminer et cherche en général à entrer en contact avec lui que ce soir directement, physiquement, uniquement lorsque le joueur est proche ou par le biais de projectiles. Nous avons pris une avance considérable sur le développement de nos intelligences artificielles et avons implémenté deux des trois types d'intelligences artificielles qu'il est prévu d'implémenter. L'attaquante et la pathfinding.

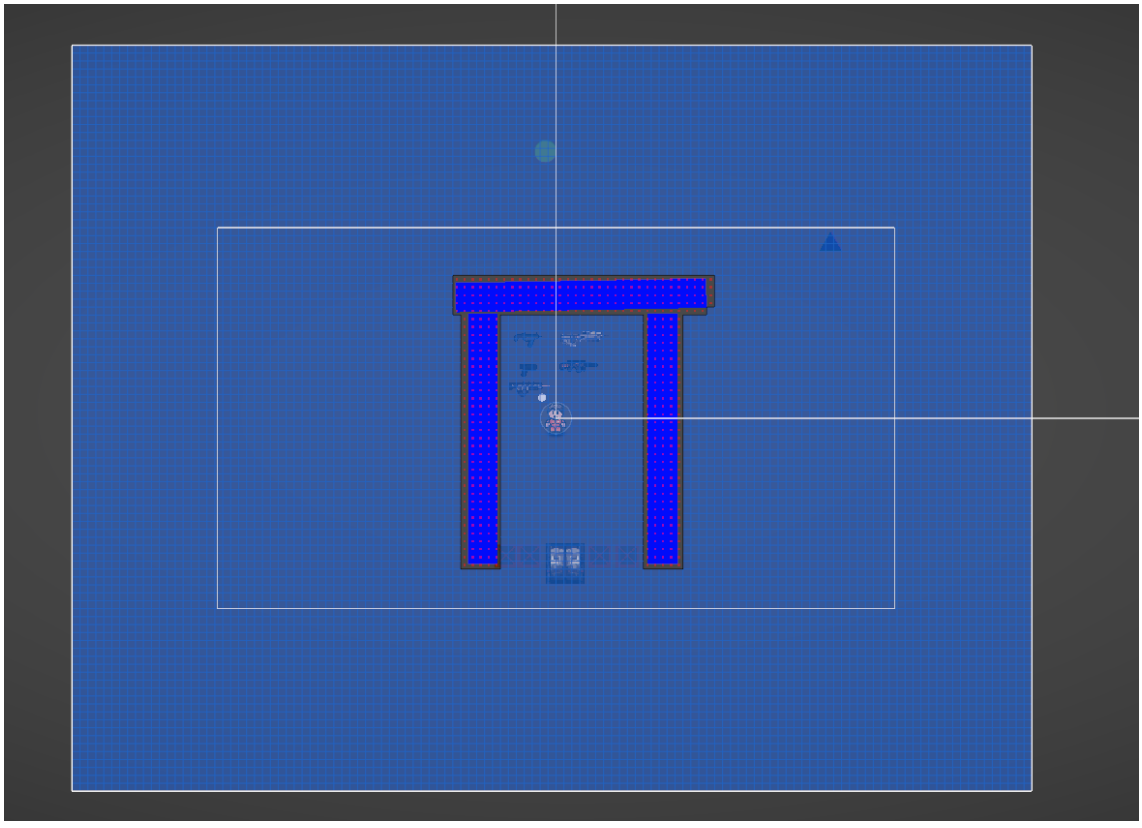
L'IA pathfinding suit le joueur tout en évitant les obstacles sur son chemin. Elle fonctionne grâce à un algorithme de pathfinding nommé A* (A star). Elle est chargée de poursuivre le joueur à travers le niveau et est implémentée sur trois ennemis, le Runner, le StrongOne, et le Spitter, leurs déplacements sont géré par A*. Le Spitter est particulier puisque nous lui avons ajouté un autre script qui le fait s'arrêter à proximité du joueur pour lui envoyer des projectiles. Le Runner se contente de tenter de rentrer en collision avec le joueur pour lui faire subir des dégâts, de même pour le StrongOne qui se démarque simplement par sa vie plus élevée. L'algorithme A* est similaire à l'algorithme de Dijkstra bien plus connu. A* trouve le chemin le plus court entre deux nœuds dans un graphe. En l'occurrence le nœud est un espace défini au sein d'une scène Unity, cet espace est une grille de points. Chaque point peut être dans l'état libre ou occupé. Si un point est occupé, cela signifie qu'un objet occupe sa position, si le point est libre, cela signifie que l'objet géré par A* peut y transiter sans entrer en collisions avec quoi que ce soit.

Le Runner calculant le plus court chemin entre lui et le joueur.



Le Spitter envoyant des projectiles sur le joueur.

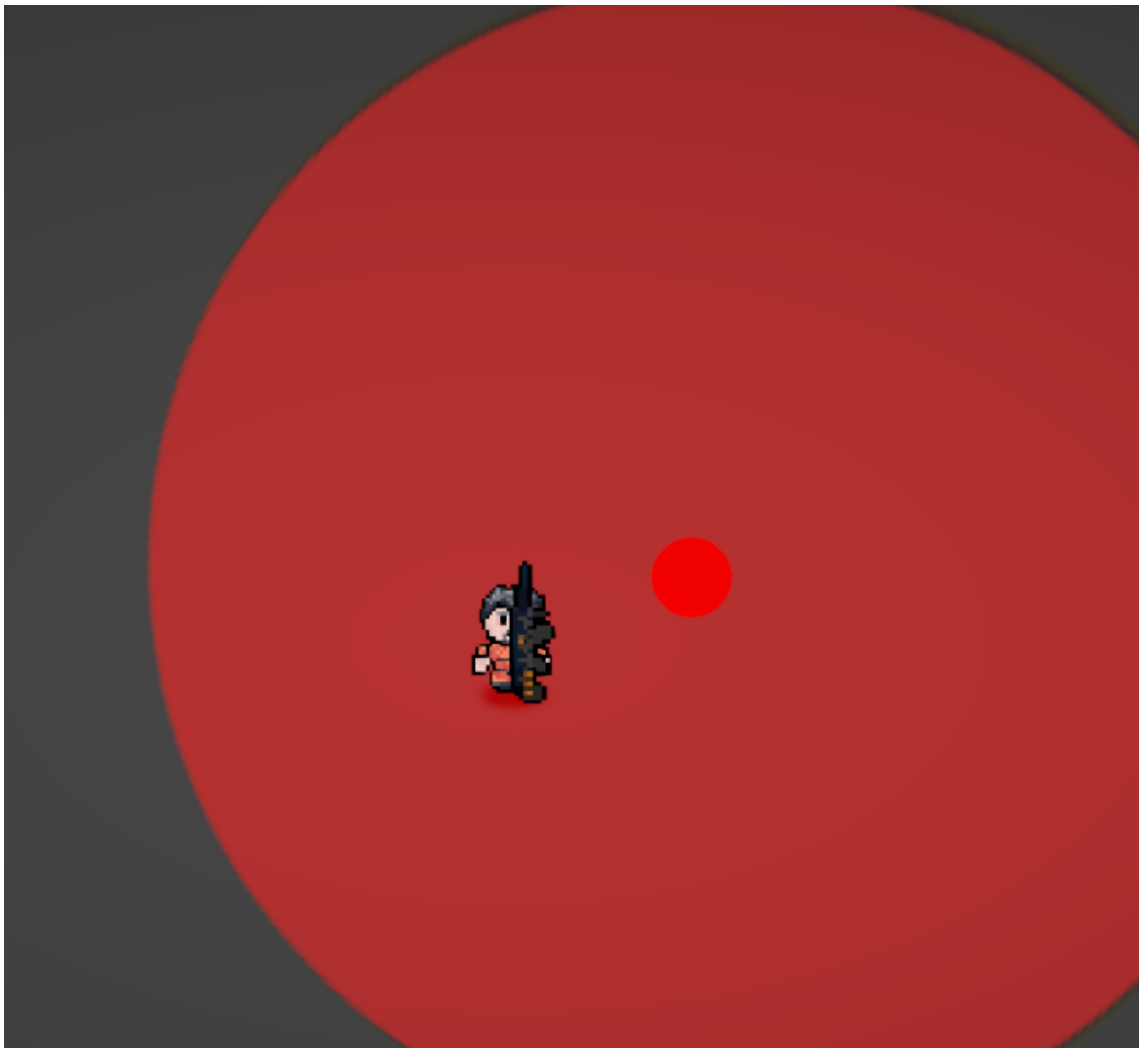
Voici la grille mentionnée, les points rouges sur les rectangles bleus sont des points occupés, l'IA pathfinding ne peut y transiter et les évitera.



Nous avons également développé l'IA attaquante, qui se contente de suivre le joueur lorsqu'elle le détecte sans prendre en compte les obstacles sur son chemin, idéale dans les grands espaces vides. Le

Sleeper et l'Exploder sont géré par ce type d'intelligence artificielle. Le sleeper avance vers le joueur si celui-ci passe près de lui et le "réveille", si le joueur s'éloigne il abandonne sa poursuite. L'Exploder lorsqu'il détecte le joueur avance à toute vitesse vers lui et explose lorsque près de lui lui infligeant des dégâts considérables.

L'Exploder explose faisant apparaîte une zone rouge autour de lui, si le joueur est présent dans cette zone il subit des dégâts.



Chaque ennemi possède son propre script de déplacement. Cependant, ces ennemis ont aussi de nombreuses propriétés communes, notamment la vie, la vitesse ou une fonction TakeDamage() qui leur font subir des dégâts. C'est la raison pour laquelle il existe un script Enemy attaché à tous les ennemis et qui gère ces différentes propriétés.

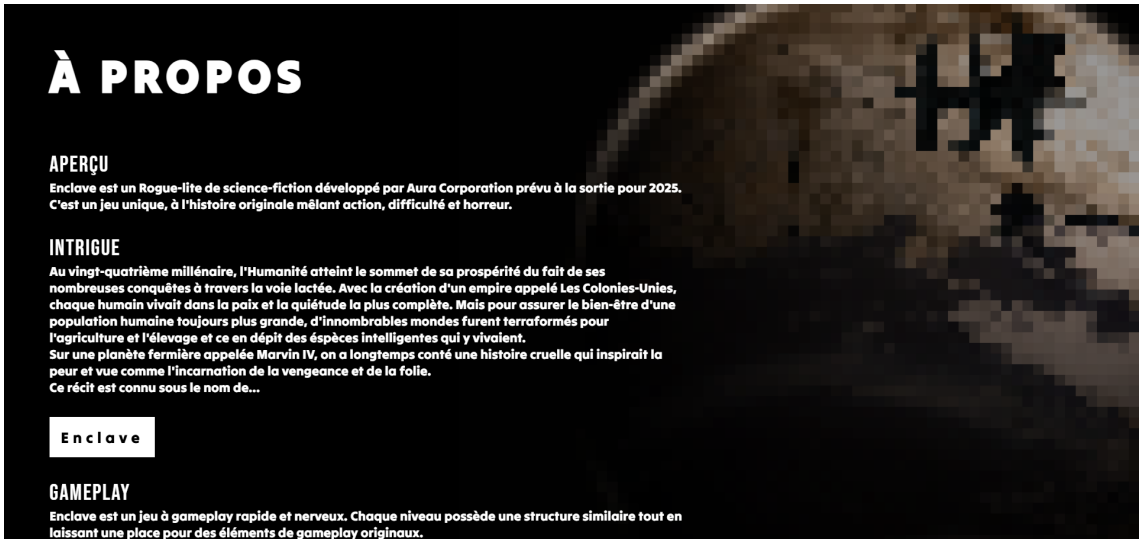
2.1.2 Site Web

Que ce soit pour faire la publicité d'un produit ou pour offrir au client des précisions sur celui-ci, un site internet est souvent la meilleure option. C'est pourquoi nous avons développé nous même un site web parfaitement fonctionnel dans lequel nos joueurs, ou futurs joueurs pourront trouver des informations pratiques sur Enclave (https://louisplaire.github.io/EnclaveWeb/).

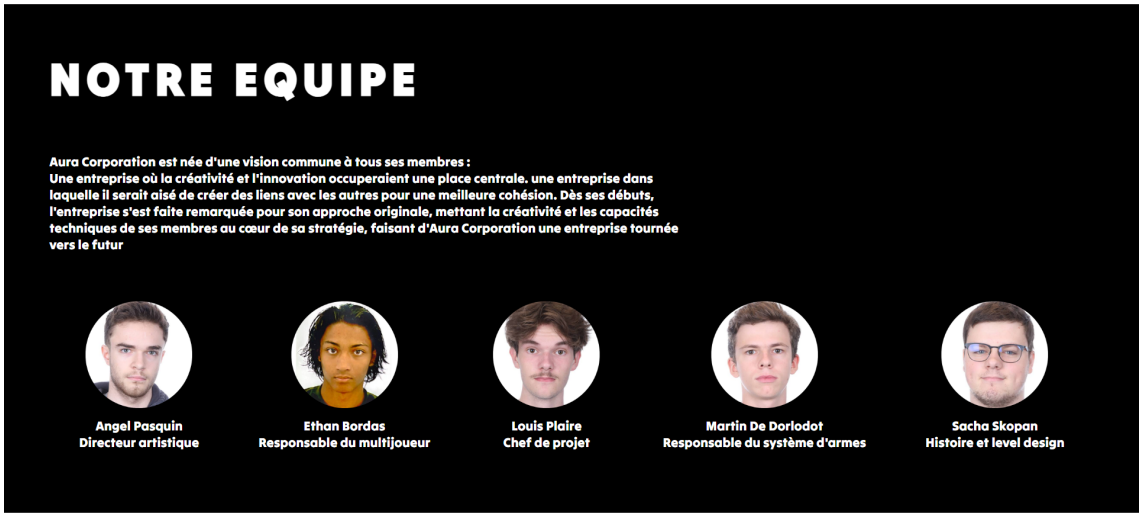
Notre site web dispose de six pages (dont la page d'accueil), chacune contient les informations nécessaires aux clients pour en apprendre plus sur nous, sur notre jeu, sur les outils que nous avons utilisé ou bien sur l'avancement du projet ce qui leur fait profiter d'une transparence totale de notre part et instaure un climat de confiance entre les joueurs et Aura Corporation.

La page d'accueil redirige vers les cinq autres pages.

La première page, la page de présentation de notre projet est accessible en cliquant sur le lien "Notre jeu : Enclave". Il s'agit de la page sur laquelle se trouvent les informations concernant l'histoire du jeu et le gameplay que ce lui-ci offre. On y trouve également la chronologie de réalisation, les problèmes rencontrés et les solutions envisagées.

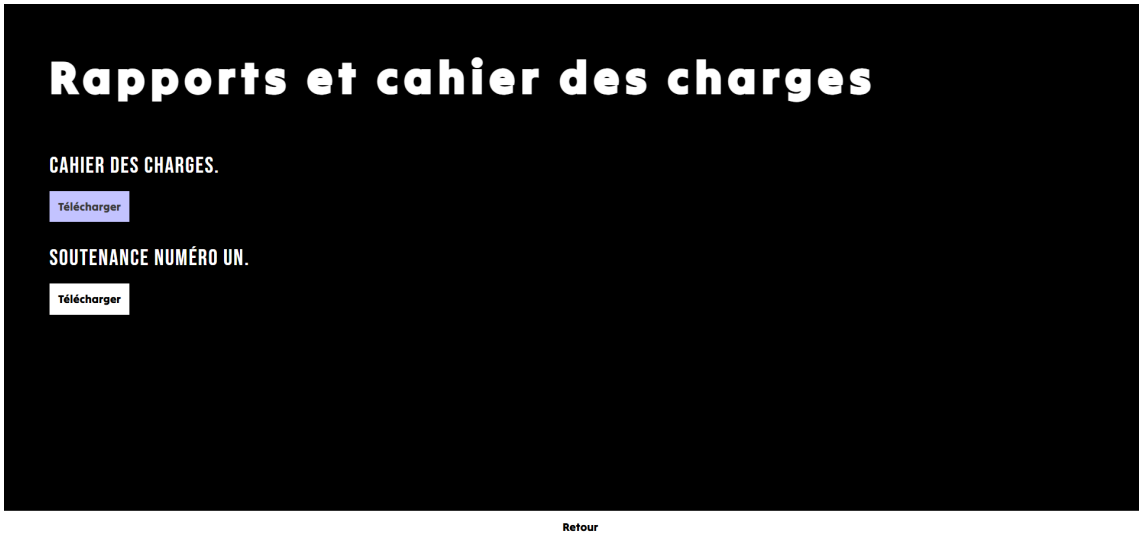


La seconde page est la page de présentation de l'entreprise et de l'équipe. S'y trouvent la photo de chacun des membres ainsi que leur rôle principal dans l'entreprise et une courte description des objectifs d'Aura Corporation.



La troisième page dispose du lien vers le site qui héberge notre jeu (itch.io). C'est depuis ce site qu'il sera possible de télécharger Enclave.

Sur la quatrième page, il est possible de télécharger notre cahier des charges ainsi que nos différents rapports de soutenances.

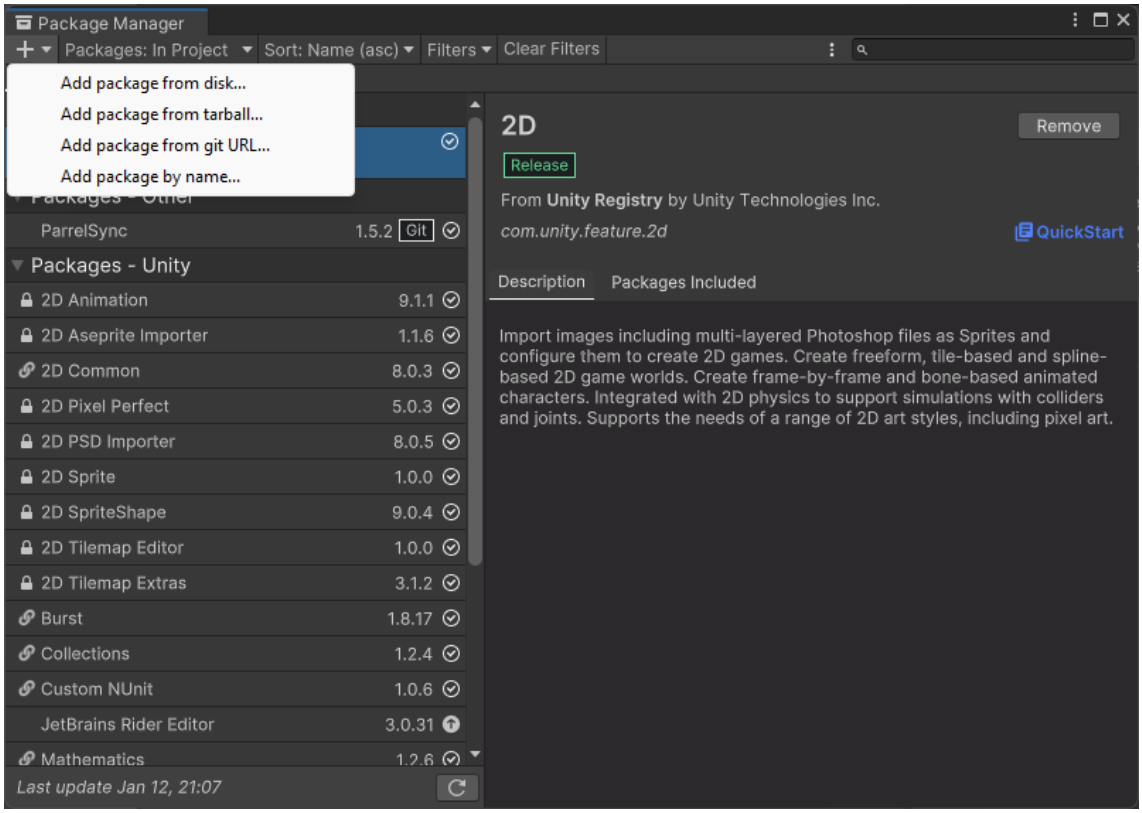


Enfin, sur la cinquième et dernière page se trouvent les liens annexes. Sur les sites des logiciels, bibliothèques, packages que nous avons utilisé dans la réalisation d'Enclave.

2.2 Sacha Skopan

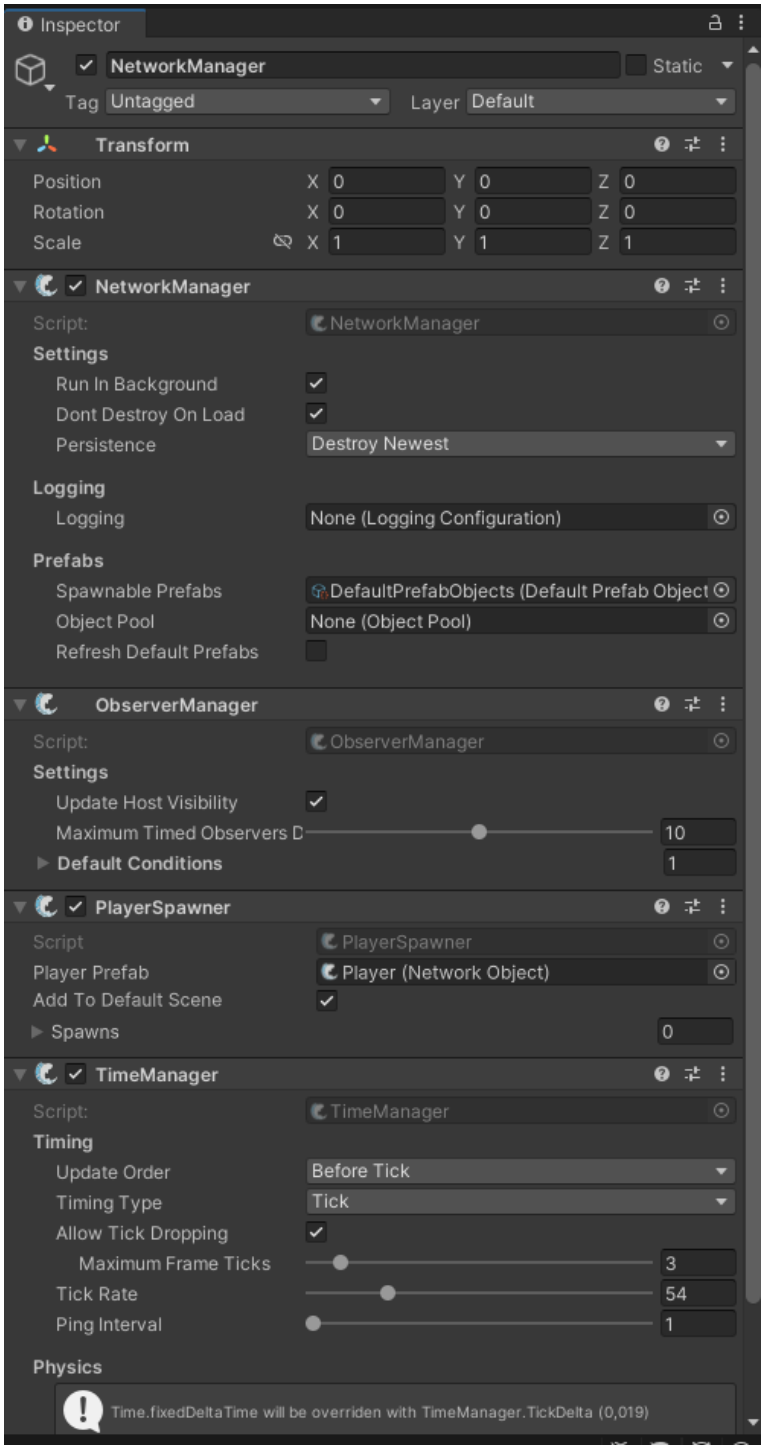
2.2.1 Multijoueur

Pour importer et mettre en place le multijoueur dans unity, il nous a fallu suivre les étapes suivantes : Nous avons importé le package FishNet grâce à son url git dans le package manager:

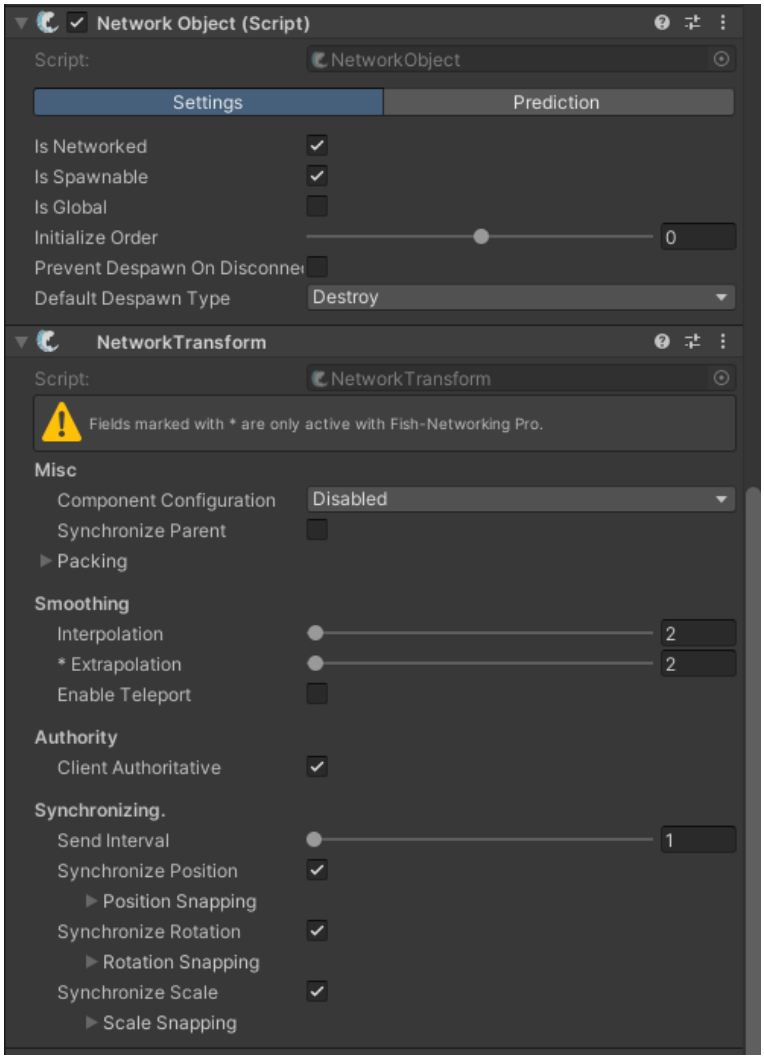


Dans les Assets on recherche le prefab NetworkManager que l'on ajoute à la scène:

Nous allons utiliser principalement les composants suivants: NetworkManager qui permet de gérer les appareils du serveur au client et PlayerSpawner qui permet de faire apparaître dans une scène, comme son nom l'indique, des joueurs et donc de simuler une partie entre plusieurs clients et un serveur.

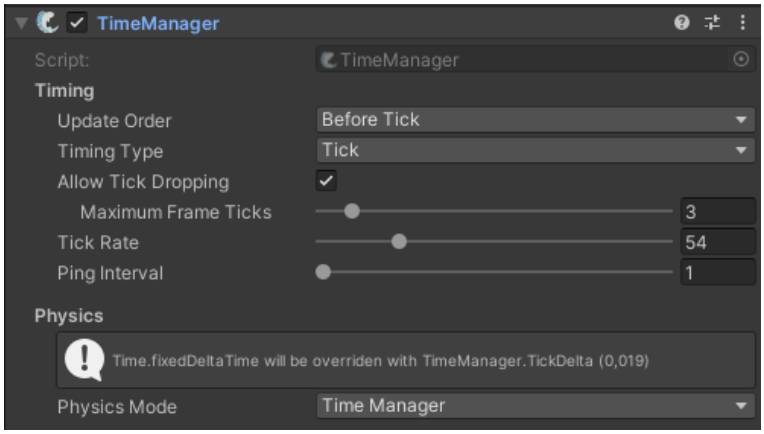


Tous les objets qui sont reliés au réseau sont censés avoir deux composants, le Network Object et NetworkTransform. Le composant Network Object nous permet de synchroniser tout objet entre le serveur et le joueur et de paramétrer les objets qui utilisent des prédictions. Le composant Network Transform permet de synchroniser les positions de tous les objets ayant ce composant.



Mais ces composants fonctionnent seulement si les objets ayant ces scripts n'utilisent pas des rigidbody. Nous avons rencontré quelques problèmes avec les sprites physiquement simulés utilisant des rigidbody2d car ils se désynchronisent entre les clients et le serveur. En effet, le serveur ne se met pas à jour assez souvent pour bien simuler les ridgidbodies. Pour régler ce problème nous utilisons le client-side-prediction de Fish-Net qui permet de recalculer les simulations physiques et de synchroniser les objets grâce à 2 fonctions qui gèrent les prédictions :
-replicate qui permet de simuler les déplacements sur le serveur puis sur le client
-reconcile qui permet de finaliser la synchronisation entre le serveur et le client.

Nous utilisons aussi Time Manager qui permet de faire des action comme des corrections toute les x seconde


TimeManager
Script:
TimeManager
Timing
Update Order
Before Tick
Timing Type
Tick
Allow Tick Dropping
Maximum Frame Ticks
3
Tick Rate
54
Ping Interval
1
Physics
Time.fixedDeltaTime will be overriden with TimeManager.TickDelta (0,019)
Physics Mode
Time Manager

2.2.2 Level Design

Tous les niveaux sont basés sur des tilemaps faites par Angel Pasquin qu'on a ensuite disposé sur chaque niveaux :

Nous avons aussi programmé des animations comme "prison" qui amène le joueur de l'arène a la prison grâce à un script que l'on a appelé "animation Manager". Ce script comme son nom l'indique s'occupe du lancement des animations et cinématique. L'animation "prison" est lancé par la coroutine:

```
IEnumerator PlayPrison()
{
    if (Anim)
    {
        if (Rifle is not null)
        {
            StartCoroutine(((UniversalRifle)Rifle).Unequip());
        }
        MainCamera.SetActive(false);
        Camera2.SetActive(true);

        Camera2.tag = "MainCamera";

        animator.enabled = true;
        Anim = false;
        animator.Play("test");
        yield return new WaitForSeconds(20.4f);
        animator.enabled = false;
        Camera2.SetActive(false);
        MainCamera.SetActive(true);
        MainCamera.transform.position = player.transform.position;
    }
}
```

Nous utilisons la coroutine "Unequip()" qui nous permet de faire poser l'arme au joueur puis nous utilisons une deuxième caméra pendant la cinématique pour la rendre plus fluide.

Nous avons aussi une animation "Elevator" lancée par :

```
IEnumerator PlayElevator()
{
    if (Anim)
    {
        MainCamera.SetActive(false);
        Camera2.SetActive(true);
        animator.enabled = true;
        Anim = false;
        animator.Play("Elevator");
        yield return new WaitForSeconds(10.4f);
        animator.StopPlayback();
        animator.enabled = false;
        Camera2.SetActive(false);
        MainCamera.SetActive(true);
        MainCamera.transform.position = player.transform.position;
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex+1);


    }
}
```

Elle nous permet de faire changer de scène le joueur.

2.3 Martin De Dorlodot

2.3.1 Système d'armes

Les armes sont le principal moyen que le joueur possède pour défaire, vaincre et venir à bout de ses adversaires. Il est donc primordial d'avoir un système d'armes non seulement fonctionnel, pratique et satisfaisant à utiliser, mais également équilibré, afin de pouvoir retenir une certaine difficulté, tout en n'étant pas non plus attaqué par une quantité accablante d'ennemis trop puissants pour les armes que le joueur peut obtenir.

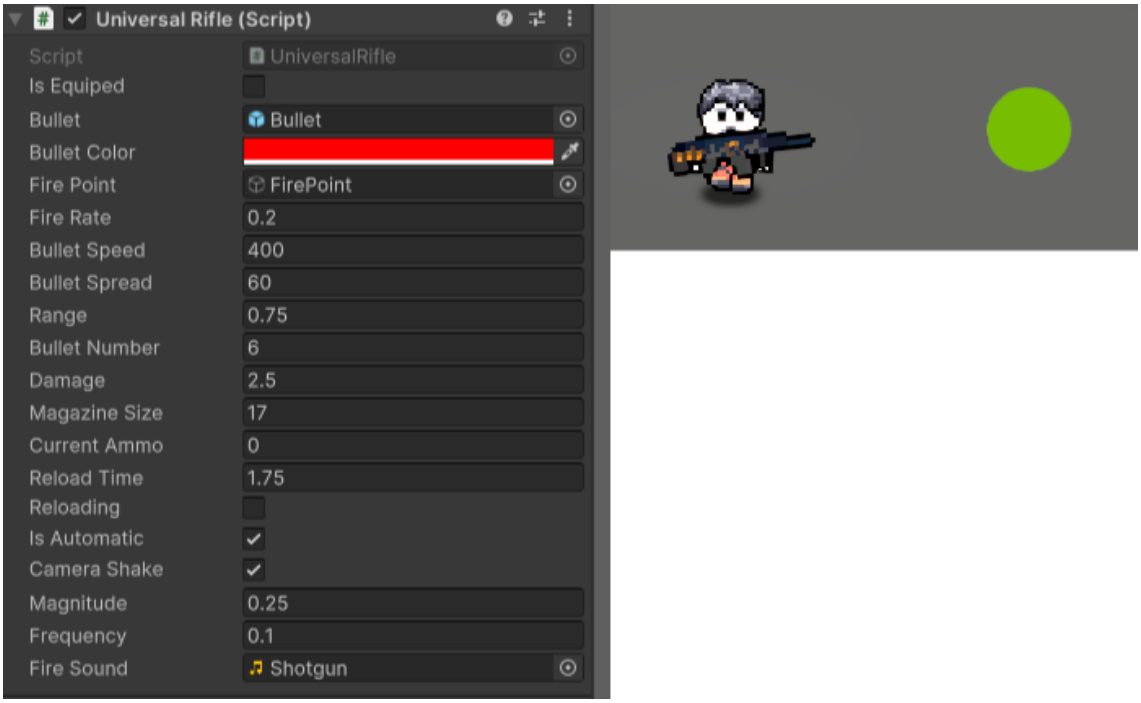
Nous avons séparé le système d'arme en 3 classes, toutes reliées à une interface "Weapon", qui rassemble des éléments communs aux armes : par exemple, nous avons un booléen IsEquipped qui nous permet de savoir si l'arme est équipée ou non, on utilise un flottant atkRate pour mettre en place un temps minimum entre chaque tir dans le cas d'une arme de distance, et le temps entre chaque instance de dégâts pour une arme de mêlée par exemple.

Les armes de distances utilisent un Fire Point, un GameObject qui représente une position sur l'arme, qui va déterminer le point de sortie de la balle.

Exemple d'un Fire Point sur un fusil à pompe :

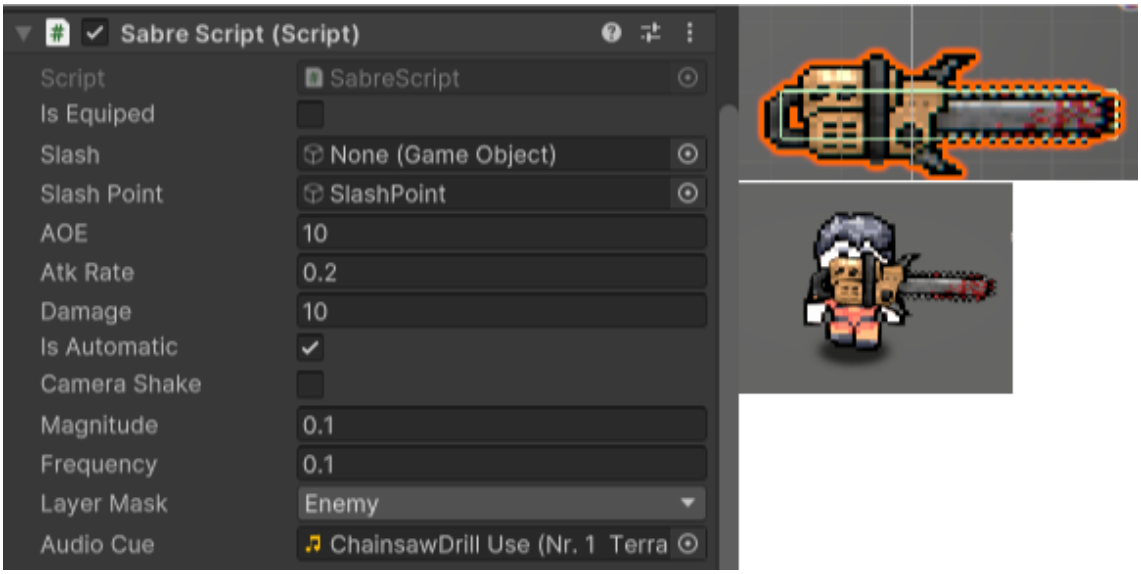
Les armes de distances dépendent toutes du script UniversalRifle.cs, qui permet de déterminer la couleur du projectile, sa vitesse, le nombre qui sera tiré par coup, la capacité de balles du chargeur de l'arme, la distance et la dispersion du projectile. On peut aussi permettre à la caméra de trembler quand le joueur tire, pour simuler l'effet de recul.

Pour reprendre l'exemple a partir de notre fusil à pompe, on peut voir que le fusil à pompe peut tirer 6 balles toutes les 0.2 secondes, à une vitesse de 400, et qui se disperse à un angle de 60°. Chaque balle fait 2.5 dégâts, il y en a 17 dans le chargeur et ce dernier prend 1.75 secondes à être chargé. (Images à la page suivante)



Pour les armes de mêlée, le fonctionnement est assez différent : ces dernières dépendent d'un script Sabre.cs. Ce dernier permet de déterminer la portée de l'attaque, les dégâts et la cadence de dégâts. Les armes mêlées utilisent un Raycast 2D, qui va appliquer les dégâts de l'arme quand l'ennemi entre en contact avec lui.

Exemple d'utilisation d'une arme de mêlée : la tronçonneuse



Ici, on peut voir que la tronçonneuse qu'utilise le joueur fait 10 dégâts toutes les 0.2 secondes, et qu'elle a une portée de 10.

La dernière partie du système d'arme, les lasers, utilisent également un Raycast, mais leurs caractéristiques seront largement différentes, avec notamment la capacité d'infliger des effets négatifs, que nous mettrons en place à partir d'une interface différente.

2.4 Angel Pasquin

2.4.1 Joueur

Dans le cadre du développement du jeu Enclave, nous avons créé 2 scripts concernant le joueur. Ces deux scripts sont des composants Unity en C-sharp, qui s'intègrent directement dans l'écosystème du moteur pour gérer la caméra et le joueur. Tout d'abord : PlayerCamera : Ce script gère une caméra qui suit dynamiquement un joueur. Voici un extrait clé du script :

```
transform.position = Vector3.Lerp(initialPos, targetPosition, Time.deltaTime * 6);
```

Ici, Vector3.Lerp() est une méthode Unity utilisée pour interpoler entre deux positions. Cela permet à la caméra de se déplacer doucement entre sa position actuelle (initialPos) et la position cible (targetPosition), qui est déterminée par la position du joueur moins un offset. Ce comportement rend le suivi du joueur fluide et naturel, évitant tout mouvement brusque.
Ce script tire parti de l'architecture d'Unity en appelant :

- GameObject.Find("Player") pour localiser dynamiquement le joueur dans la scène.
- Transformations spatiales pour adapter la position de la caméra à celle du joueur.

L'avantage d'un tel système est qu'il s'adapte dynamiquement à la position du joueur sans nécessiter de configuration manuelle complexe. Ensuite nous avons : PlayerCharacter : Ce script est le cœur du comportement du joueur. Plutôt que de simplement stocker des variables, il établit des bases solides pour l'intégration avec d'autres systèmes Unity, comme les collisions, les mouvements et l'interface utilisateur. Par exemple :

```
[SerializeField] Slider healthSlider;
```

Grâce à l'attribut [SerializeField], cette variable est exposée dans l'inspecteur Unity, permettant de lier facilement une barre de santé à l'état du joueur sans modifier le code. Unity se charge alors de mettre à jour l'interface en fonction des changements dans le script. La gestion des collisions et des mouvements est également facilitée grâce à :

```
private Rigidbody2D rb;
IL code
private BoxCollider2D collider;
```

Ces composants Unity intégrés gèrent automatiquement la physique 2D, comme les forces et les interactions avec l'environnement. Ces scripts profitent pleinement des fonctionnalités d'Unity, comme l'inspecteur pour relier des objets, et des composants physiques pour simplifier la gestion des interactions. En résumé, PlayerCamera assure le confort visuel du joueur, tandis que PlayerCharacter établit les bases pour son contrôle et ses interactions. Ces scripts, bien que simples dans leur état actuel, forment les blocs de construction d'un jeu structuré, où chaque élément a son rôle défini.

2.4.2 Pixel Art

Dans Enclave, l'identité visuelle repose sur un style pixel art minutieusement élaboré pour garantir une esthétique unique et immersive. Pour atteindre cet objectif, nous utilisons une application spécialisée appelée Pixel Studio, qui est parfaitement adaptée à la création de graphismes en pixel art. Cet outil nous permet de concevoir avec précision chaque élément visuel du jeu, qu'il s'agisse des personnages, des décors ou des objets interactifs.

Chaque visuel est dessiné à la main, un choix artistique qui apporte une authenticité et une singularité à l'univers du jeu. Ce processus manuel nous donne un contrôle total sur les détails et les couleurs, tout en capturant l'essence rétro qui caractérise le pixel art. Grâce à cette approche, chaque élément de l'univers d'Enclave est une véritable œuvre artisanale.

Pour donner vie à ces créations, nous produisons des sprite sheets. Une sprite sheet est une grille regroupant plusieurs images ou "frames" qui représentent les différentes étapes d'une animation. Par exemple :

• Un personnage en train de marcher aura une sprite sheet contenant des frames pour chaque étape du mouvement : lever un pied, déplacer le corps, poser le pied, etc.
• Une animation d'attaque pourrait inclure des frames illustrant le moment où l'arme est levée, l'impact, et le retour à la position initiale.

Le Processus de Création :
1. Conception des Frames : Chaque frame est dessinée séparément dans Pixel Studio. Cela nécessite une attention particulière aux proportions, aux détails et à la cohérence entre les frames pour assurer une fluidité dans l'animation.

2. Assemblage : Une fois toutes les frames créées, elles sont placées dans une grille unique pour former la sprite sheet. Cela permet une organisation claire et facilite leur utilisation dans le moteur de jeu.

3. Exportation : Les sprite sheets sont ensuite exportées en un format compatible avec notre moteur de jeu (comme Unity). Ce format contient toutes les informations nécessaires pour recréer l'animation.

Implémentation dans le Jeu :
Dans le moteur de jeu, les sprite sheets sont utilisées pour créer des animations :
• Les frames sont jouées de manière séquentielle
• Ce processus donne l'illusion du mouvement, transformant des images statiques en une action fluide et dynamique.

Par ailleurs, le moteur permet d'associer ces animations à des événements spécifiques dans le jeu :
• Lorsque le joueur appuie sur une touche pour déplacer son personnage, le jeu déclenche l'animation de marche correspondante.
• De même, lorsqu'un ennemi attaque ou subit des dégâts, une animation adaptée est jouée.

Le Résultat Final :

Ce processus, bien qu'exigeant, permet de transformer de simples dessins en une expérience visuelle riche et immersive. Grâce à cette attention portée à chaque détail, Enclave propose un univers visuel cohérent et captivant, où les interactions semblent naturelles et les personnages prennent véritablement vie.

2.5 Ethan Bordas

2.5.1 Interface Utilisateur

1. Dans le développement de jeux vidéo et d'applications interactives, la capacité à modifier dynamiquement les éléments de l'interface utilisateur est cruciale pour améliorer l'expérience utilisateur. Unity, avec son puissant système de gestion de l'interface utilisateur, permet aux développeurs de créer des interactions riches et réactives. Le script ChangeText.cs est un exemple de la manière dont les développeurs peuvent changer la couleur du texte en fonction des interactions de l'utilisateur. Cette dissertation examine la structure, la fonctionnalité et les implications de ce script dans le contexte du développement d'interfaces utilisateur.
I. Présentation du Script ChangeText.cs

Le script ChangeText.cs est conçu pour changer la couleur d'un texte lorsque l'utilisateur passe la souris dessus. Il utilise les interfaces IPointerEnterHandler et IPointerExitHandler de Unity pour détecter les interactions de la souris. Le script se compose de trois méthodes principales : OnPointerEnter, OnPointerExit, et une déclaration de variable publique TMPText.

A. La Déclaration de la Variable TMPText
Le script commence par déclarer une variable publique TMPText appelée theText. Cette variable est utilisée pour référencer l'élément de texte dont la couleur sera modifiée.

B. La Méthode OnPointerEnter
La méthode OnPointerEnter est appelée lorsque le curseur de la souris entre dans la zone de l'élément de texte. Dans cette méthode, la couleur du texte est changée en noir (Color.black).

C. La Méthode OnPointerExit
La méthode OnPointerExit est appelée lorsque le curseur de la souris quitte la zone de l'élément de texte. Dans cette méthode, la couleur du texte est réinitialisée à blanc (Color.white).

II. Analyse de la Fonctionnalité
Le script ChangeText.cs démontre l'utilisation efficace des interfaces d'événements de Unity pour créer des interactions utilisateur dynamiques. En utilisant seulement quelques lignes de code, le script permet de changer la couleur du texte en fonction des mouvements de la souris.

A. Simplicité du Code
Le script est simple et facile à comprendre. La déclaration de la variable publique permet de lier facilement l'élément de texte dans l'éditeur Unity, et les méthodes OnPointerEnter et OnPointerExit utilisent des commandes simples pour changer la couleur du texte.

B. Réactivité de l'Interface Utilisateur
La capacité à changer la couleur du texte en fonction des interactions de la souris améliore la réactivité de l'interface utilisateur. Cela peut être utilisé pour attirer l'attention de l'utilisateur sur des éléments interactifs ou pour fournir des retours visuels.

2.
I. Présentation du Script MenuMananger.cs Le script MenuMananger.cs est conçu pour gérer les transitions entre différentes scènes de menu dans un jeu Unity. Il utilise la fonction SceneManager.LoadScene pour charger différentes scènes en fonction des actions de l'utilisateur. Le script se compose de plusieurs méthodes principales : Start(), Update(), Play(), Credit(), HowToPlay(), et parametres().

A. La Méthode Start() La méthode Start() est appelée avant le premier frame update. Dans ce script, elle est actuellement vide, ce qui signifie qu'aucune initialisation spécifique n'est nécessaire au démarrage de la scène.

B. La Méthode Update() La méthode Update() est appelée une fois par frame. Elle est également vide dans ce script, indiquant qu'aucune logique continue n'est requise pour cette fonctionnalité spécifique.

C. La Méthode Back() La méthode Back() est la partie la plus cruciale du script. Elle est déclenchée lorsque le joueur clique sur le bouton pour revenir au menu principal. Cette méthode utilise SceneManager.LoadScene(" pour charger la scène du menu principal.

D. La Méthode Play() La méthode Play() est appelée lorsque le joueur clique sur le bouton pour commencer le jeu. Elle utilise SceneManager.LoadScene("SigmaScene") pour charger la scène principale du jeu.

E. La Méthode Credit() La méthode Credit() est appelée lorsque le joueur clique sur le bouton pour voir les crédits. Elle utilise SceneManager.LoadScene("Credits") pour charger la scène des crédits.

F. La Méthode HowToPlay() La méthode HowToPlay() est appelée lorsque le joueur clique sur le bouton pour voir les instructions du jeu. Elle utilise SceneManager.LoadScene("HowToPlay") pour charger la scène des instructions.

G. La Méthode parametres() La méthode parametres() est appelée lorsque le joueur clique sur le bouton pour accéder aux paramètres du jeu. Elle utilise SceneManager.LoadScene("Parametres") pour charger la scène des paramètres.

II. Analyse de la Fonctionnalité Le script MenuMananger.cs démontre l'utilisation efficace des outils de gestion des scènes de Unity pour créer des menus interactifs. En utilisant des méthodes simples, le script permet de naviguer facilement entre différentes sections du jeu.

A. Simplicité du Code Le script est simple et facile à comprendre. Chaque méthode utilise une seule ligne de code pour charger la scène correspondante, ce qui rend le script facile à maintenir et à déboguer.

B. Efficacité de la Navigation La capacité à naviguer rapidement entre les scènes de menu améliore l'expérience utilisateur. Les joueurs peuvent facilement accéder aux différentes sections du jeu, ce qui rend l'interface utilisateur plus intuitive et réactive. Conclusion Le script MenuMananger.cs est un excellent exemple de la manière dont les développeurs peuvent utiliser les outils de Unity pour gérer les menus de manière efficace. Sa simplicité et son efficacité en font un modèle idéal pour les développeurs débutants et expérimentés. En comprenant et en améliorant ce script, les développeurs peuvent créer des interfaces utilisateur plus engageantes et immersives.

3 Ce qui doit être fait pour la prochaine soutenance

D'ici la semaine du 10 mars 2025, l'objectif sera d'avoir fait tous les effets sonores, ainsi que les musiques ambiantes du jeu. Il faudra également avoir complété le multijoueur et avoir fini intégralement le level design.

L'art sera intégralement fini, avec les sprites de tous les ennemis, les armes, les accessoires, la quasi-totalité des aspects visuels d'Enclave seront achevés. Les animations seront entièrement finies et cinématiques seront également toutes ou presque toutes finies.

Les systèmes d'armes, d'accessoires et de capacités devront être dans sa phase de finition : on verra apparaître une multitude de changements et d'ajouts, pas seulement en termes d'armes, mais aussi de types d'armes, des armes spéciales, ainsi que des effets positifs et négatifs que certaines armes pourraient avoir. Ces effets seraient implantés à partir de classes et d'interfaces qui pourraient permettre de changer différentes stats de l'adversaire, en réduisant sa vitesse, ses dégâts, etc.

Les accessoires quant à eux vont offrir une expérience du jeu entièrement différente, donnant la possibilité d'avoir un gain de stats, d'avoir la possibilité de sacrifier une stats pour augmenter une autre (E.g. Le joueur a moins de points de vie mais les dégâts du joueur sont augmentés). Il est prévu d'utiliser une interface pour mettre en commun les accessoires, de la même manière que les armes ont une interface Weapon, ce qui sera utile pour déterminer quels sont les accessoires que le joueur a équipé.

Le développement de l'IA sera quant à lui en plein développement, avec la majorité des ennemis commun ayant une IA fonctionnelle et au point, et avec certains ennemis majeurs / Boss ayant leur IA en plein développement. Cela vas permettre d'avoir une expérience du jeu immersive et bien plus fidèle au résultat final.

L'interface utilisateur sera proche de la complétion, et le système d'événement aura été entamé, avec les premiers événements que l'on pourra retrouver dans certains niveaux du début, et d'autres plus poussées dans la zone de test.

4 Conclusion

Le développement d'Enclave progresse à un excellent rythme grâce au travail acharné de toute l'équipe. L'intelligence artificielle, un pilier central du gameplay, est déjà bien avancée avec deux des trois types prévus. Elle permet de rendre les ennemis plus réactifs et stratégiques, en utilisant des algorithmes comme A* pour les déplacements et des mécaniques spécifiques pour les attaques. Ces systèmes enrichissent l'expérience de jeu et ajoutent un vrai défi pour les joueurs.

L'interface utilisateur a été conçue pour être simple et intuitive. Grâce à des menus clairs et interactifs, les joueurs peuvent naviguer facilement entre les différentes parties du jeu, comme lancer une partie, accéder aux paramètres, ou consulter les crédits. Cette interface garantit une prise en main rapide et une immersion totale dès les premiers instants.

Côté visuel, les graphismes en pixel art, entièrement dessinés à la main, apportent une identité unique à Enclave. Chaque personnage, décor et animation sont soigneusement créés pour donner vie à cet univers. Les animations, réalisées avec des sprites, permettent des mouvements fluides et renforcent l'immersion du joueur dans l'ambiance du jeu.

Le mode multijoueur, bien qu'encore en développement, s'annonce prometteur. Pensé pour être joué en local, il met l'accent sur le partage et la convivialité. Des outils comme FishNet sont utilisés pour synchroniser les actions entre joueurs, et des solutions techniques sont en cours pour résoudre certains défis liés à la physique des objets en réseau.

Enfin, le système d'armes est conçu pour offrir des combats variés et dynamiques. Fusils, armes de mêlée et lasers disposent chacun de mécaniques spécifiques, comme des temps de rechargement, des effets spéciaux ou encore des projectiles interactifs. Cela permet aux joueurs d'adapter leur style de jeu et d'explorer différentes stratégies au fil des niveaux.

En résumé, malgré quelques aspects encore en cours de finalisation, Enclave repose sur des bases solides. Grâce à la créativité et aux efforts de l'équipe, le jeu s'annonce comme une aventure immersive et captivante qui saura séduire les joueurs.