Rapport de soutenance
Deuxième 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 un développement important, mais l'innovation a souvent été utilisée à des fins financières plutôt que créatives. Aura Corp. veut changer cela en créant des jeux qui allient plaisir et véritable originalité. Le projet principal d'Aura Corp, Enclave, est un rogue-lite inspiré de jeux tels que Dead Space, Rogue et The Escapists. Il propose un mode multijoueur qui ajoute de la stratégie au gameplay et offre une expérience immersive où chaque choix compte. De plus, l'équipe a optimisé le jeu pour le rendre fluide et amusant, et a retravaillé le design des niveaux pour les rendre plus agréables et plus attrayants pour l'utilisateur. Chez Aura Corp, nous pensons que les jeux vidéo sont une forme d'art en soi, et sans sacrifier la créativité pour des nombres, des expériences uniques Avec Enclave, nous voulons redonner aux jeux vidéo la place qui leur revient dans le monde de l'art.

Dans le cadre de notre projet, nous avons développé un jeu : Enclave. L'objectif était de se concentrer avant tout sur le gameplay, l'optimisation et le mode multijoueur.

Les mécaniques de combat étaient une partie importante à prendre en compte afin que les utilisateurs prennent un maximum de plaisir en jouant. Pour cela, nous avons équilibré les armes de différentes distances pour rendre les combats plus justes et amusants. Nous avons également amélioré la qualité des animations et des interactions entre les joueurs et l'environnement du jeu. Par exemple, nous voulions que les joueurs ressentent chaque coup et observent les réactions de leurs adversaires.

Pour le mode multijoueur, nous avons travaillé sur la synchronisation des actions pour que l'expérience en ligne soit fluide et agréable pour l'utilisateur. Nous avons fait des tests en conditions réelles, ce qui nous a permis de repérer et corriger de nombreux problèmes comme les décalages entre les joueurs, les actions qui ne s'exécutent pas correctement, ou encore les bugs d'affichage pendant les parties. C'est pourquoi ,nous avons choisi Fish-Networking, une solution robuste permettant d'assurer une meilleure communication entre le serveur et les clients. Grâce à cela, le multijoueur est aujourd'hui fonctionnel et stable, bien qu'il reste encore quelques améliorations à apporter, notamment pour la gestion du jeu en réseau local.

Nous avons également retravaillé le level design de certains éléments, par exemple, en rendant les affrontements plus stratégiques et les zones de jeu plus réalistes pour les joueurs. De plus, nous avons bossé sur l'ambiance sonore, en ajoutant des musiques et des effets qui plongent encore plus le joueur dans l'action et qui renforce l'intensité des combats. Un autre élément important du level design était la gestion de l'intelligence artificielle (IA). D'une part, nous avons modifié leur comportement pour éviter qu'elles ne se bloquent dans certaines zones ou ne réagissent de manière incohérente. D'autre part nous avons délimité les distances d'activation et les chemins du jeu afin d'éviter qu'elles ne s'accumulent inutilement devant des obstacles.

Ainsi ,ce rapport présente en détail les avancées réalisées, les problèmes rencontrés et les prochaines étapes du développement.

2 Etat de l'avancement

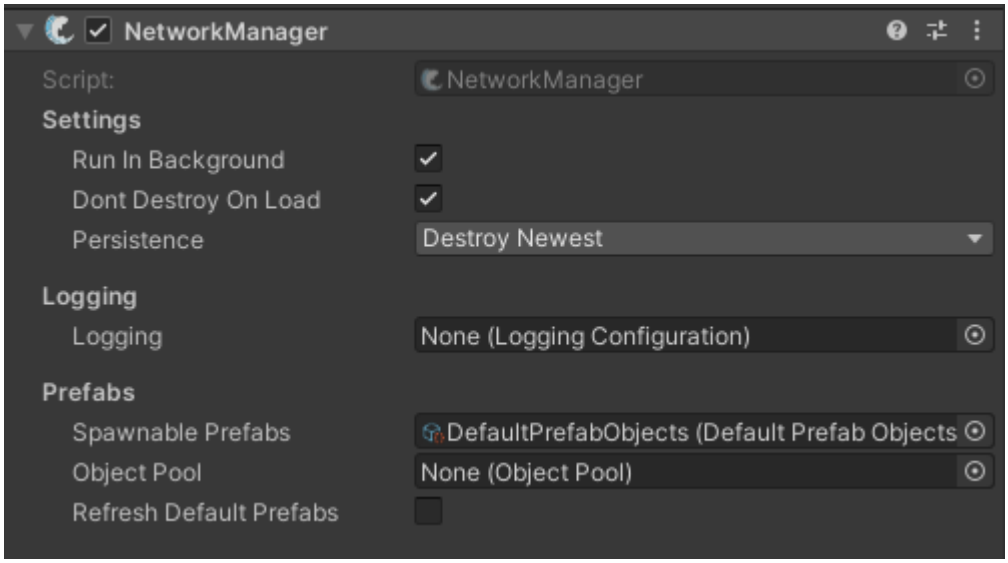
2.1 Louis Plaire

2.1.1 Multijoueur

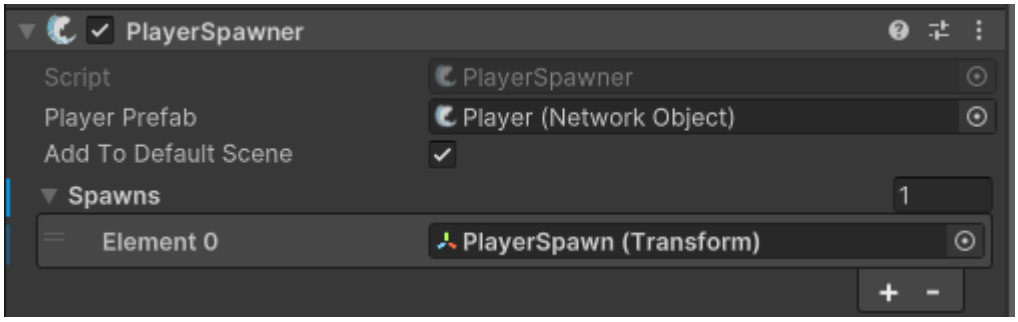
Au cours des derniers mois, il nous a été possible d'implémenter une des fonctionnalités clefs d'Enclave : le multijoueur. Une fonctionnalité qui a pour but de faire d'Enclave un jeu collaboratif, un jeu qui pousse les joueurs à créer du lien, à échanger, à établir des stratégies. En d'autres termes, développer l'esprit d'équipe de nos joueurs. Le développement du multijoueur ne s'est pas déroulé sans difficulté, son développement fut long et fastidieux. En effet, le multijoueur interagit avec chacun des aspects d'enclaves (déplacement des personnages, synchronisation des différents joueurs, apparition d'objets...), c'est en quelque sorte les fondations de notre jeu. Développer une telle fonctionnalité plus tôt nous aurait donc éviter de nombreuses complications car il aurait été possible de lier le multijoueur avec chaque nouvelle fonctionnalité à mesure que celles-ci étaient développées. Malgré ces quelques contre temps, le développement du multijoueur est arrivé à son terme.

Nous utilisons Fish-Networking, un package Unity polyvalent et populaire qui permet le développement de multijoueurs en réseau global ou local en permettant à chaque joueur de se comporter comme client ou serveur.

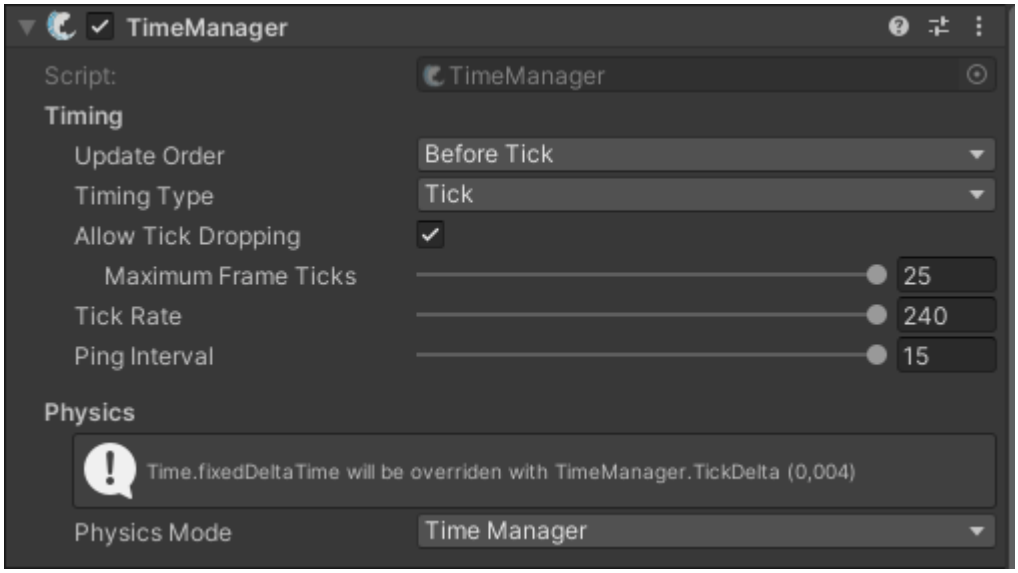
Mettre en place Fish-Net sur notre projet Unity fut chose simple. Il faut importer un prefab nommé NetworkManager dans une scène de notre projet. Le NetworkManager est la pièce centrale du multijoueur, il s'agit d'un GameObject contenant divers Components, chacun gérant un aspect du multijoueur. Tout d'abord nous avons le NetworkManager. Celui-ci est essentiel pour avoir un serveur et des clients en état de marche. Il permet au serveur d'interagir avec les Components d'Unity.



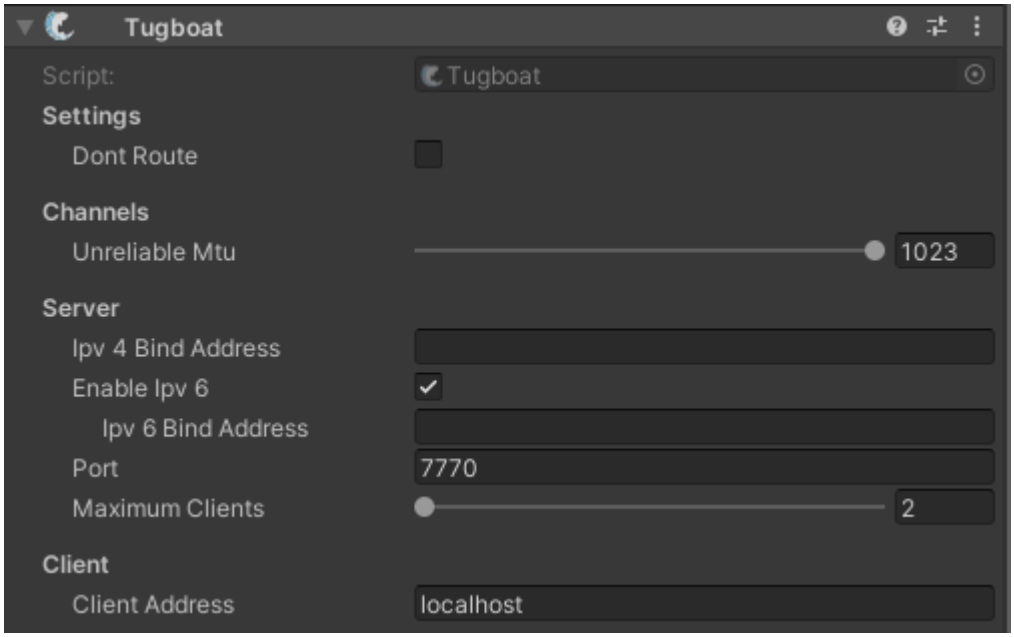
Ensuite nous avons le PlayerSpawner. Comme son nom l'indique, il fait apparaître dans une scène un prefab correspondant au joueur à des coordonnées spécifiées.



Puis il y a le TimeManager qui sert à mesurer un temps commun à chaque client de façon à synchroniser diverses actions que chacun pourrait effectuer.



Enfin, nous avons TugBoat qui est un Transport, c'est-à-dire qu'il est le Component qui s'occupe de la transmission des paquets, des informations entre les différents membres du réseau ainsi que de veiller à leur intégrité.



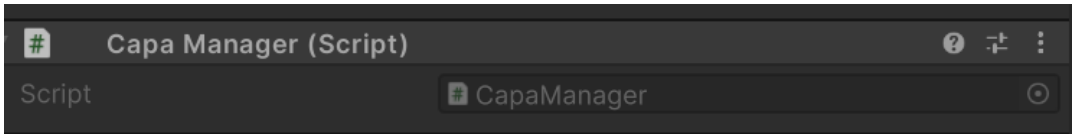
Le principal défi que représentait l'implémentation du multijoueur était de comprendre comment il était possible de synchroniser les objets physiques présents dans notre jeu (Le joueur, les ennemis, les projectiles). Pour ce faire, il est possible d'utiliser le Client Side Prediction ou CSP. Le CSP consiste à calculer les changements de force de nos objets physiques sur le serveur avant de les appliquer aux clients. De cette manière, chaque client est bien synchronisé avec le serveur. Dans notre projet, les différents objets physiques non contrôlés par l'utilisateur utilisent le CSP grâce à un script que nous avons appelé RigidbodySync dans lequel la fonction principale est ReconcileState. Cette fonction synchronise les calculs liés à la physique réalisés par le serveur avec chaque client.

Les éléments principaux du multijoueur ont été implémentés avec succès et celui-ci est parfaitement fonctionnel. Cependant, il reste quelques tâches à effectuer le concernant. Il subsiste notamment quelques problèmes de latence dans la communication entre les clients et le serveur, probablement liés aux limites de Fish-Networking. C'est un problème que nous pensons pouvoir régler. De plus, le multijoueur d'Enclave ne fonctionne pour l'instant que sur une seule et même machine. Or, notre objectif était de pouvoir jouer sur plusieurs appareils en réseau local. C'est une tâche qui nous pose peu de difficultés et nous avons même déjà un plan pour la réaliser.

2.2 Sacha Skopan

2.2.1 Système de capacités

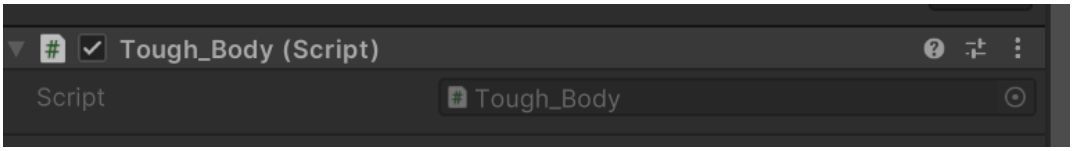
Nous avons créé un script nommé “CapaManager” qui s’occupe d’activer les capacités du joueur lorsque celui-ci les récupère.



Lorsque le joueur entre en collision avec le sprite d’une capacité, il la récupère si celle-ci dispose du tag “Tough_Body”.

Dans le script de “CapaManager” nous avons créé une classe publique CapaManager, dans laquelle nous avons une fonction OnCollisionEnter2D qui utilise un switch qui différencie le tag du sprite touché par le joueur. Ici, si le joueur touche un objet ayant un tag “Tough_Body” le script ajoute le script “Tough_Body” au joueur et détruit le sprite.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


Script Unity | 0 références
public class CapaManager : MonoBehaviour
{
    // Start is called before the first frame update
    Message Unity | 0 références
    private void OnCollisionEnter2D(Collision2D collision)
    {
        switch (collision.collider.tag)
        {
            case "Tough_Body":
                this.gameObject.AddComponent<Tough_Body>();
                Destroy(collision.collider.gameObject);
                break;
        }
    }
}
```

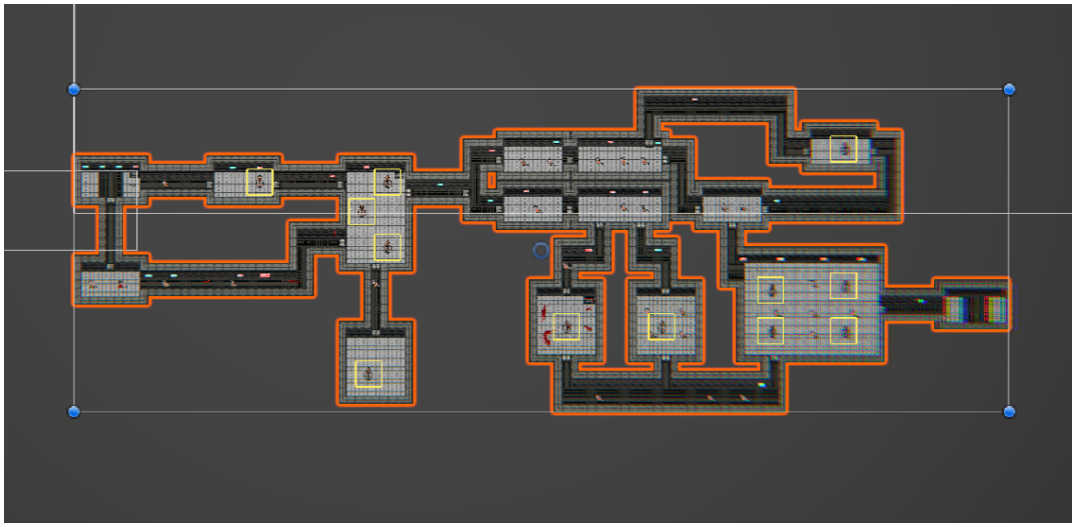


Le script de “Tough_Body” lui permet d’exécuter la capacité “Tough_Body”, la capacité double le nombre de points de vie du joueur pendant 5 secondes et est utilisable toutes les 90 secondes. Si le joueur appuie sur la touche “Alt Gauche” le script exécute Apply() qui applique l’effet au joueur.

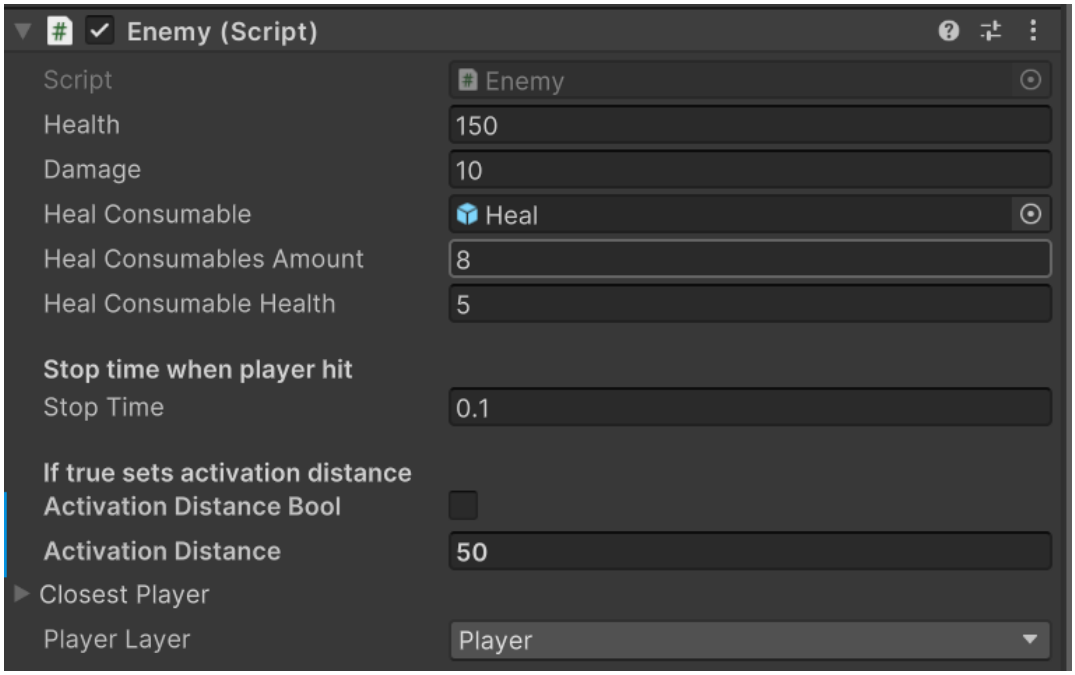
```
using System.Collections;
using System.Collections.Generic;
using MainAssets;
using UnityEngine;


public class Tough_Body : MonoBehaviour
{
    PlayerCharacter character;
    bool CanApply=true;
    // Start is called before the first frame update
    Message Unity | 0 références
    void Start()
    {
        character = GetComponent<PlayerCharacter>();

    }

    // Update is called once per frame
    Message Unity | 0 références
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.LeftAlt))
        {
            StartCoroutine(Apply());
        }
    }

    1 référence
    IEnumerator Apply()
    {
        if (CanApply)
        {
            CanApply = false;
            character.health *= 2;
            yield return new WaitForSeconds(5);
            character.health /= 2;
            yield return new WaitForSeconds(90);
            CanApply = true;
        }
    }
}
```

2.2.2 Level Design

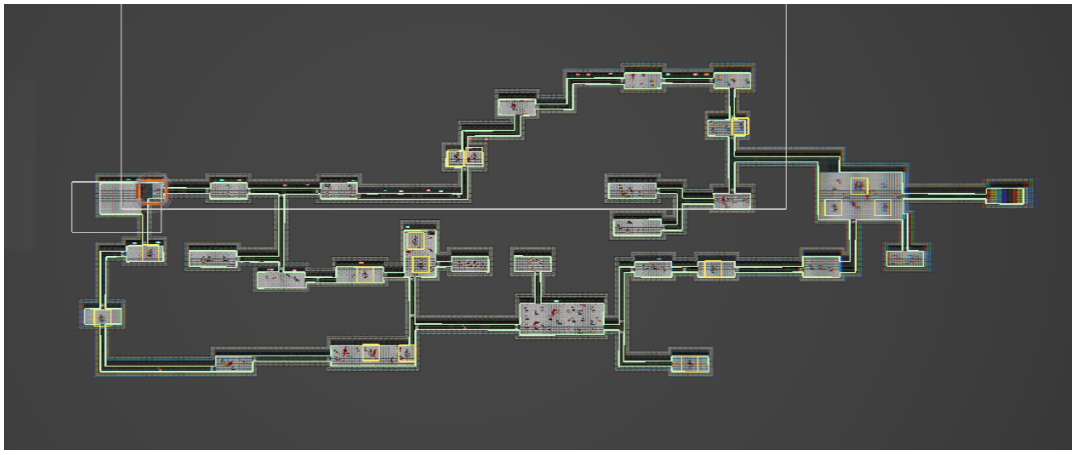
Le level design est terminé, nous avons implémenter les 30 niveaux comme prévu. Ils sont tous créés grâce à une grille sur laquelle nous avons placé différentes tuiles.



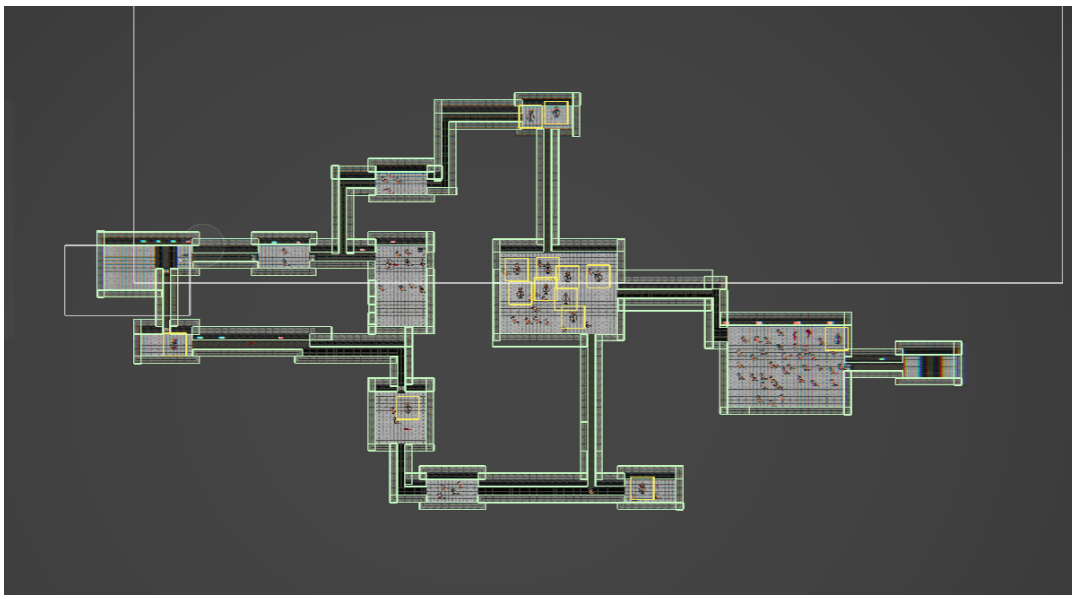
Dans chaque niveau nous avons réussi à bien implémenter les ennemis “Sleeper” et “StrongOne”. Pendant leur implémentation nous avons remarqué que l’ia “StrongOne” essayait dès le début du niveau d’atteindre le joueur, ce qui les amenait à s’accumuler devant les portes qui leur font obstacle au reste du niveaux. Pour palier à ce problème, nous avons ajouté dans le script “Enemy” une distance d’activation de 50 unités ce qui permet de rendre le parcours des niveaux plus fluide.



Pour que le joueur et les IA ne sortent pas des niveaux nous utilisons plusieurs EdgeCollider2D qui font le tour de l’extérieur et de l’intérieur des niveaux.

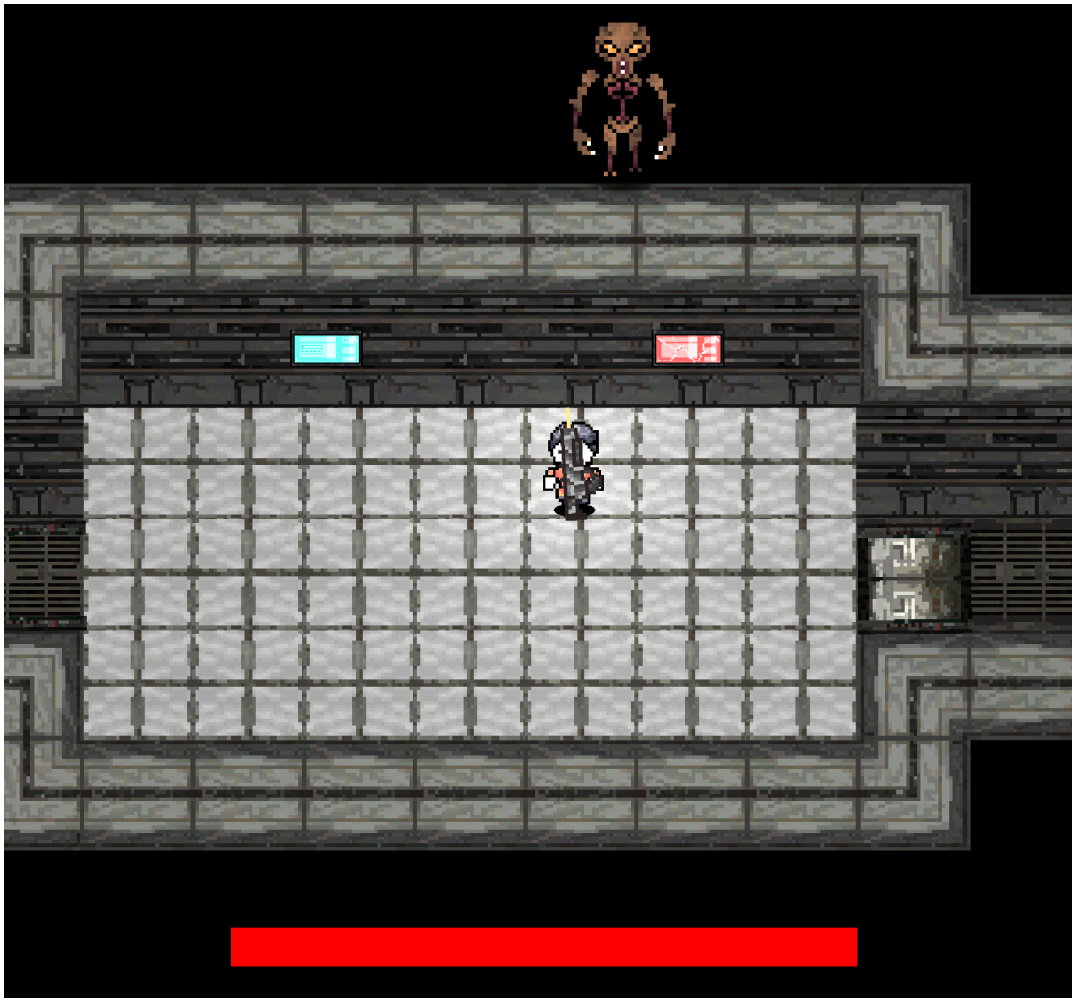


Nous savons que les EdgeCollider2D ont une grande complexité due au nombre de côtés qu'ils peuvent contenir. Même si nous n'avons pas de problèmes de performance nous avons tout de même trouvé une alternative aux EdgeCollider2D. Pour cela nous avons remplacé les EdgeCollider2D des niveaux 17, 18 et 19 par une multitude de de BoxCollider2D qui font aussi le tour du niveaux.



Après plusieurs essais nous n'avons pas vu de changement comparé aux EdgeCollider2D donc nous les avons laissé sur ces niveaux.

Nous avons fait plusieurs tests, et remarqué que certaines IA passent à travers les EdgeCollider2D lorsque elles sont proches de portes.



Elles passent à travers les EdgeCollider2D une fois puis restent bloqués. Nous pensons que cela est dû au fonctionnement des portes. Toutes les portes des niveaux possèdent un BoxCollider2D qui fait la taille de la porte, mais lorsque le joueur s'approche, la porte s'ouvre et le BoxCollider2D est désactivé. Le BoxCollider2D de la porte étant en contact avec le edge collider 2D sur un court espace le désactive ce qui permet à l'IA de traverser les EdgeCollider2D.

Ce phénomène n'a pas l'air de se produire dans les niveaux 17, 18 et 19, ceux qui contiennent des BoxCollider2D au lieu des EdgeCollider2D. Nous prévoyons donc remplacer les EdgeCollider2D de tous les niveaux par des BoxCollider2D.

Pour le système d'événement aléatoire chaque niveau a une ou deux salles dans lesquelles il peut se passer un événement aléatoire. Nous savons quelles salles seront concernées par les événements aléatoires et en avons déjà élaboré plusieurs types mais nous ne les avons pas encore amélioré.
Pour finaliser les niveaux, il nous reste à implémenter le système d'événements aléatoires ainsi qu'à placer différents sprites répartis au sein de niveaux comme des chaises, des comptoirs, des caisses et d'autres objets pour rendre les niveaux plus naturels et permettre au joueur d'en apprendre plus sur l'univers d'Enclave à travers des éléments de décors.

2.3 Martin De Dorlodot

2.3.1 Système d'armes

Le système d'arme a quelque peu évolué depuis la dernière soutenance, bien que de nombreux défis se soient imposés, particulièrement avec l'implémentation du multijoueur.
Tout d'abord, nous avons implémenté un système d'armes à base de lasers, qui traversent les ennemis et s'arrêtent lorsqu'ils entrent en contact avec un obstacle.
Pour ce faire, nous avons utilisé le Raycsat2DAll, il s'agit d'un type Unity qui va créer une liste de Raycast2D. Un Raycast2D est un type de Unity qui va projeter un laser invisible à partir d'un point de départ dans une direction. Ce Raycast peut prendre en argument ce qu'on appelle un layer. Ce layer permet au Raycast de nous renvoyer uniquement les données d'éléments qui sont sur le même layer, c'est un filtre. Un Raycast2DAll envoie un Raycast2D et lorsque ce dernier entre en collision avec un élément, il s'arrête et le Raycast2DAll crée alors un autre Raycast2D qui fera la même chose que le précédent, jusqu'à ce que la distance entre l'origine du premier Raycast2D et le dernier ai atteinte une valeure prédéfinie. Ces Raycast2D sont ajoutés, au fur et à mesure qu'ils sont créés, à une liste de Raycast 2D, ce qui nous permet de traiter les cas de manière individuelle. Par exemple, nous avons fait en sorte que si un Raycast entre en contact avec un élément qui a le même layer que les ennemis, et bien le Raycast suivant est créé, et l'ennemi prend des dégâts. Par contre, si un Raycast entre en contact avec un élément qui a le même layer qu'un obstacle, alors il s'arrête et le Raycast suivant n'est donc pas créé. Donc si notre laser a en face de lui un ennemi, un obstacle puis un autre ennemi, alors il touchera le premier ennemi mais sera arrêté par le mur.

Les lasers peuvent également appliquer différents effets aux ennemis, tels que des dégâts au fil du temps, une réduction de la vitesse de déplacement ou bien une réduction des dégâts que ces ennemis font. Tous les effets ne sont pas encore implémentés.

```
foreach (RaycastHit2D hit in hits)
{
    if (hit && canDamage)
    {
        if (hit.collider.TryGetComponent<Enemy>(out Enemy enemy))
        {
            canDamage = false;
            enemy.TakeDamage(damage);
        }

        if (hit.collider.gameObject.layer == 6) break;

        yield return new WaitForSeconds(fireRate);
        canDamage = true;
    }
}
```

Pour réaliser le visuel, nous avons créé un GameObject LineRenderer, un GameObject qui crée une ligne entre 2 points. Donc, le point de départ est le même que le Raycast 2 DAll, et le point d'arrivée est soit le point de collision avec un obstacle, soit le point au bout de la distance du laser.

Pour ce qui est des autres armes, les armes mêlées ont fait l'objet d'un bref débat parmi notre équipe vis-a-vis de leur fonctionnement, il est donc prévu de revoir la manière dont elles font des dégâts pour la majorité.

Les armes à feu étaient déjà mises au point, mais des changements sont envisageables si nous voulons faire en sorte que les armes de distance puissent aussi infliger des effets.

2.4 Angel Pasquin

2.4.1 Musiques

Pour Enclave, nous avons décidé de composer plusieurs musiques, avec pour objectif principal l'immersion. En effet, les musiques sont une étape très importante pour rendre une œuvre immersive. Les musiques dans Enclave sont pensées pour renforcer l'intensité des combats avec des sonorités puissantes et agressives, tout en proposant aussi des musiques plus calmes et mystérieuses.
Nos principales inspirations proviennent des bandes-son de DOOM et Halo.

Deux jeux connus pour la qualité de leur musique. DOOM se distingue par des musiques Heavy Metal avec des sonorités électriques et saturées, mêlant guitares électriques des rythmes rapides et distorsions extrêmes pour accentuer la brutalité du gameplay. De son côté, Halo propose une approche bien plus variée, avec des morceaux parfois épiques et intenses comme DOOM, mais aussi des musiques plus calmes et mystérieuses, jouant sur les chœurs (chant grégorien) et le piano. Nous avons cherché à mélanger ces différentes inspirations pour offrir une bande-son qui accompagne à la fois les moments de violence/tension et des moments plus calmes comme le menu principal par exemple pour ramener de la douceur après avoir affronté des hordes de monstres sans relâche.
Pour le processus de réalisation des musiques, nous avons utilisé deux logiciels : FL Studio et BandLab.

Ces logiciels permettent de composer des musiques et sont simples d'utilisation. Pour obtenir des sons métal percutant, nous avons travaillé sur des guitares saturées mises à disposition par les différents logiciels.

Ensuite, en appliquant des effets de distorsion et de compression nous avons réussi à créer une tonalité et un grain agressif. Des percussions et les basses puissantes ont été rajoutées pour accentuer l'impact et le rythme des morceaux. Des synthétiseurs et du piano ont aussi été utilisés pour les musiques plus calmes et donner un effet d'écho qui correspond bien à notre jeu qui se déroule dans l'espace.

Nous avons ainsi composé pour le moment quatre musiques : Marvin IV, qui sera utilisé en tant que thème principal, dans le menu du jeu. Comme pour la musique de menu de Halo 2, cette musique est différente de celles présente dans le gameplay, elle contient une ambiance calme, mystérieuse, avec du piano et des synthétiseurs.
Chaque morceau a été soigneusement créé pour garantir une cohérence sonore et une immersion totale dans l'univers d'enclave. Grâce à ces compositions, nous avons cherché à renforcer le gameplay et procuré une certaine émotion chez le joueur.

2.5 Ethan Bordas

2.5.1 Site web : étapes à la réalisation

Dans ce projet, nous avons créé une page HTML pour afficher une frise chronologique qui retranscirt les différentes étapes du développement d'Enclave. L'idée était de rendre les informations claires et bien structurées.
L'idée était de montrer l'avancée du projet de façon claire et compréhensible, donc nous avons choisi de le représenter avec un chronographe. Nous avons utilisé du HTML pour la structure et le CSS pour l'apparence.

L'en-tête de la page :
Nous avons codé ¡meta charset="UTF-8"¿pour éviter les problèmes d'affichage des accents et ¡meta viewport¿qui sert à s'adapter aux écrans de différentes tailles.

Le contenu principal :
Un titre ¡h2¿qui annonce "CHRONOLOGIE DE RÉALISATION". Plusieurs blocs ¡div class="line"¿, chacun représentant une étape du projet avec : Un cercle (¡div class="circle"¿) pour marquer l'étape. Un texte (¡p class="task"¿) qui précise la tâche et la date. Une bande (¡div class="stripe"¿) pour montrer les différentes étapes. Un bouton "Retour" (¡a id="retour" href="index.html"¿Retour¡/a¿) pour revenir à la page d'accueil.

Alignement des éléments :
Au début, les cercles et les lignes n'étaient pas bien alignés. Nous avons réglé cela avec display: flex et justify-content pour mieux les positionner. Affichage sur tous ecrans : Sur un petit écran, la frise était mal positionnée. Nous avons utilisé des "@media queries" en CSS pour adapter l'affichage selon la taille de l'écran.

Lien visuelle entre les étapes :
Les lignes (.stripe) ne s'assemblent pas bien entre elles. Nous avons ajusté les marges et les hauteurs en CSS afin d'avoir un contenu plus agréable à regarder.

Pour améliorer le site web, l'ajout d'animations en JavaScript permettrait d'obtenir une frise plus dynamique (par exemple, une apparition progressive des étapes). Pour montrer les différentes phases du projet, je pourrais mettre des icônes ou des images. Enfin, pour rendre la frise interactive, l'utilisateur cliquera sur une étape, affichant ainsi les informations de la tâche concernée.

Ce projet m'a permis de mieux comprendre comment fonctionne une page web, par exemple, l'affichage en CSS ou bien la structure en HTML. J'ai également appris à gérer des problèmes d'affichage et à rendre le design plus agréable pour l'utilisateur.

3 Ce qui doit être fait pour la prochaine soutenance

Le développement d'Enclave arrive à son terme. Nous avons implémenté une grande partie des éléments clefs de notre jeu. Nous développerons donc l'unique fonctionnalité qu'il nous reste à implémenter : le système d'événements. Un système dynamique de générations d'événements aléatoires qui a pour but d'apporter au joueur de nouveaux éléments de gameplay à chaque nouvelle partie de façon à mener le joueur à constamment s'adapter à de nouvelles situations.

Une fois le système d'événements implémenté, nous aurons accompli l'ensemble des tâches. Nous pourrons alors consacrer le temps qu'il nous reste à perfectionner Enclave. Nous améliorerons notamment l'interface utilisateur pour rendre plus intuitif le lancement de parties multijoueurs, nous réduirons les latences entre les serveurs et les clients pour apporter une meilleure expérience de jeu à nos utilisateurs. Nous avons également prévu d'ajouter de nouvelles armes et de nouveaux ennemis en nous basant sur ceux que nous avons déjà développés, cela permettra d'ajouter de la variété dans la difficulté. Enfin il nous restera à rendre compatibles avec le multijoueur les armes de mêlée ainsi que les armes laser.

4 Conclusion

Le développement d'Enclave a connu une progression significative au fil des derniers mois, avec l'implémentation de plusieurs systèmes fondamentaux qui enrichissent l'expérience de jeu. L'un des points majeurs de cette avancée est l'amélioration du système d'armes, notamment les armes laser qui ont été conçues avec une gestion avancée des collisions à l'aide du Raycast2DAll. Cette approche permet d'obtenir un comportement réaliste où les tirs traversent les ennemis mais sont bloqués par les obstacles, ajoutant une dimension stratégique au combat. De plus, plusieurs effets de statut sont en cours d'implémentation pour diversifier l'impact des armes sur les ennemis. Par ailleurs, des réflexions ont été menées sur l'équilibrage des armes de mêlée et à distance, dans l'objectif d'affiner le gameplay et d'apporter plus de variété aux affrontements.

L'implémentation du mode multijoueur représente une avancée majeure dans le projet. Grâce à FishNetworking, nous avons pu mettre en place une architecture client-serveur fonctionnelle, permettant aux joueurs d'évoluer ensemble dans un environnement synchronisé. Ce processus a été complexe, car il a nécessité une refonte de plusieurs systèmes existants pour assurer une synchronisation fluide des objets physiques, des déplacements des personnages et des interactions en temps réel. L'utilisation de la Client Side Prediction (CSP) a permis d'améliorer la réactivité et la cohérence du gameplay entre les différents joueurs. Toutefois, quelques défis persistent, notamment la gestion de la latence et la compatibilité du multijoueur sur plusieurs appareils en réseau local, qui seront des axes d'amélioration prioritaires dans les prochaines étapes du développement.

Le level design est désormais achevé, avec la création et l'optimisation de 30 niveaux, chacun structuré autour d'un système de tilemap et d'un placement rigoureux des obstacles et des IA. Lors des tests, nous avons identifié un problème lié au comportement des ennemis, notamment l'IA "StrongOne" qui avait tendance à s'accumuler devant certaines portes. Pour corriger cela, nous avons introduit une distance d'activation limitant leur déplacement jusqu'à la détection effective du joueur. Un autre défi rencontré concerne la gestion des collisions, où certaines IA peuvent traverser involontairement les limites des niveaux. Après plusieurs tests, il apparaît que le problème pourrait être lié aux interactions entre les edges colliders 2D et les portes, ce qui nous a conduit à envisager le remplacement progressif de ces colliders par des box colliders 2D, qui semblent mieux gérer ces interactions.

Un élément essentiel encore en attente d'implémentation est le système d'événements aléatoires, qui jouera un rôle clé dans la rejouabilité du jeu. Ce système vise à introduire des éléments dynamiques dans les niveaux, en générant des événements imprévisibles qui obligeront les joueurs à adapter leur stratégie en temps réel. Cette fonctionnalité apportera une nouvelle dimension au gameplay, en brisant la monotonie et en renouvelant l'expérience à chaque partie. Parallèlement, nous prévoyons d'intégrer davantage d'éléments de décor, tels que des meubles et des objets interactifs, afin de renforcer l'immersion et de donner plus de profondeur à l'univers d'Enclave.

En ce qui concerne les mécaniques de gameplay supplémentaires, l'ajout des capacités pour les joueurs apporte une nouvelle couche de personnalisation et de stratégie. Le système de récupération et d'activation des capacités a été conçu de manière à être intuitif et accessible, permettant au joueur de déclencher des effets temporaires qui influencent directement son style de jeu. Par exemple, la capacité "Tough Body" double les points de vie pendant une durée limitée, ce qui peut être décisif dans certaines situations de combat. Ce système pourra être enrichi par de nouvelles capacités afin d'offrir encore plus de diversité dans la façon dont les joueurs abordent chaque affrontement.

Enfin, la dimension sonore du jeu a été soigneusement travaillée pour renforcer l'atmosphère et l'immersion. Inspirée des bandes-son de DOOM et Halo, la musique d'Enclave mélange des sonorités puissantes

et dynamiques pour accompagner les moments de combat intense, ainsi que des compositions plus calmes et éthérées pour instaurer une ambiance immersive dans les phases plus tranquilles du jeu. Chaque morceau a été créé avec l'objectif de s'adapter au rythme du gameplay, et l'utilisation de logiciels comme FL Studio et BandLab a permis d'obtenir un rendu sonore de qualité. L'intégration de ces musiques contribue à donner une identité forte au jeu et à marquer les moments clés de l'expérience des joueurs.

À l'approche de la prochaine soutenance, les efforts se concentreront sur la finalisation du système d'événements aléatoires, l'optimisation des performances du multijoueur, ainsi que l'amélioration générale de l'interface utilisateur pour une meilleure accessibilité. Des ajustements seront également apportés aux armes et aux mécaniques de combat pour assurer un équilibre optimal. Avec ces dernières étapes, nous visons à proposer une version complète et aboutie d'Enclave, offrant une expérience fluide, immersive et captivante aux joueurs qu'artistique.