Projet S2 EPITA

Rapport de projet

Noé Cornu ; Théo Gille ; Clément Grégoire ; Leandro Tolaini

Juin 2022

Table des matières

Chapitre 1

Reprise du cahier des charges

Pour cette dernière soutenance, nous avons travaillé dur afin d'atteindre nos objectifs fixés. Tout d'abord, nous nous sommes concentrés sur le design et les mécaniques de niveaux, dans le but de proposer des niveaux cohérents avec l'histoire que nous tentons de raconter à travers ce jeu. Nous avons également cherché à offrir des niveaux de plus en plus difficiles, introduisant de nouvelles mécaniques au fil de la progression, afin de défier les joueurs.

Ensuite, nous avons consacré beaucoup d'efforts au développement de l'inventaire et à l'interaction des joueurs avec les objets. Cela nous a permis de diversifier le gameplay en ajoutant des clés, des soins, et bien d'autres éléments. L'objectif était de créer une expérience de jeu riche et immersive, où les joueurs auraient de nombreuses options à leur disposition pour progresser dans l'aventure.

Enfin, lors de la préparation de cette soutenance, une grande partie de notre travail a été consacrée aux finitions, telles que les animations, les cinématiques et quelques mécaniques de combat. Ces détails sont essentiels pour offrir aux joueurs une expérience visuelle et sonore de qualité. Cependant, ils requièrent également beaucoup de temps et d'attention afin d'être réalisés de manière satisfaisante.

De manière plus globale, sur ce projet, nous avons réussi à maintenir une cohésion solide malgré les différentes difficultés rencontrées. Tout au long de sa réalisation, nous avons veillé à maintenir une régularité dans notre travail et à répartir au mieux les différentes tâches, afin d'éviter une surcharge de travail juste avant les soutenances et de garantir une charge de travail équitable pour tous les membres de l'équipe.

Nous sommes fiers d'annoncer que nous avons atteint tous les objectifs que nous nous étions fixés lors de l'élaboration du cahier des charges. De plus, nous avons pu ajouter un grand nombre de fonctionnalités bonus, telles que l'inventaire, qui n'était pas initialement prévu, mais que nous avons jugé pertinent d'ajouter afin de simplifier la gestion des objets et d'apporter une nouvelle dimension stratégique à notre jeu. Parmi les autres fonctionnalités bonus que nous avons intégrées, on compte également les plateformes mo-

biles, qui permettent de diversifier les niveaux et de les rendre plus complexes.

Ces ajouts supplémentaires ont contribué à enrichir l'expérience de jeu, offrant aux joueurs des mécanismes variés et stimulants. Nous avons mis un point d'honneur à les intégrer de manière fluide et à les équilibrer afin qu'ils s'intègrent harmonieusement à l'ensemble du gameplay existant.

2eme soutenance	Pas commencé	En cours	Fini
Design et interface			X
UI & Menus			X
IA			X
Multijoueurs			X
Map design			X
Site Web			X
Son effet			X
Mecanique de niveau			X
Dialogues			X

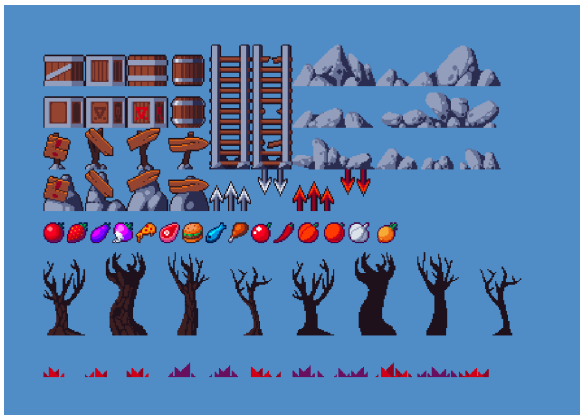
Chapitre 2

Noé Cornu

2.1 Design et Interface

Depuis le début du projet, nous avons travaillé ardemment sur le design du jeu en utilisant divers moyens pour atteindre nos objectifs. Plus spécifiquement, j'ai personnellement utilisé Photoshop afin de créer des designs simples tels que des boutons et une chatbox pour intégrer les dialogues et les indications de gameplay. Cette approche a considérablement amélioré l'interface utilisateur et a offert une expérience plus fluide aux joueurs.

Concernant le reste du design, nous avons choisi de nous appuyer sur des assets trouvés sur Internet, car aucun des membres de notre équipe ne possède de compétences en dessin. Cette solution nous a permis de gagner du temps et de nous concentrer sur d'autres aspects du développement. Nous avons consciencieusement sélectionné des assets de qualité, afin d'assurer la cohérence visuelle et d'offrir un environnement attractif pour les joueurs.



Nous sommes extrêmement satisfaits de l'intégration réussie du design au gameplay, ce qui renforce l'expérience de jeu globale. Les designs simples que nous avons créés grâce à Photoshop ont permis d'améliorer l'interface utilisateur, facilitant ainsi la compréhension et l'interaction des joueurs avec le jeu. Quant aux autres éléments de design, les assets soigneusement choisis ont contribué à créer un environnement visuel cohérent et esthétiquement plaisant.

2.2 Intelligence Artificiel

L'implémentation d'une intelligence artificielle pour les ennemis est l'un des points principaux de notre projet. Après avoir étudié différentes méthodes pour mettre en place l'intelligence artificielle des ennemis, j'ai finalement opté pour la méthode MoveTowards qui permet de déplacer l'ennemi vers un point cible en suivant une ligne droite. Cependant, avant d'utiliser cette méthode, il a fallu prendre en compte certaines considérations pour éviter les erreurs d'exécution.

Tout d'abord, il est important de vérifier si au moins un des deux joueurs est présent sur le niveau et, le cas échéant, déterminer le joueur le plus proche de l'ennemi. Une fois cette information recueillie, le script attribue la direction de déplacement de l'ennemi vers le joueur le plus proche et met à jour cette position à chaque image pour suivre le joueur en temps réel. Cette approche garantit que l'ennemi se déplace de manière fluide et réactive en direction du joueur approprié.

Ces étapes sont cruciales pour assurer un comportement réaliste et adaptatif de l'intelligence artificielle des ennemis dans le jeu.

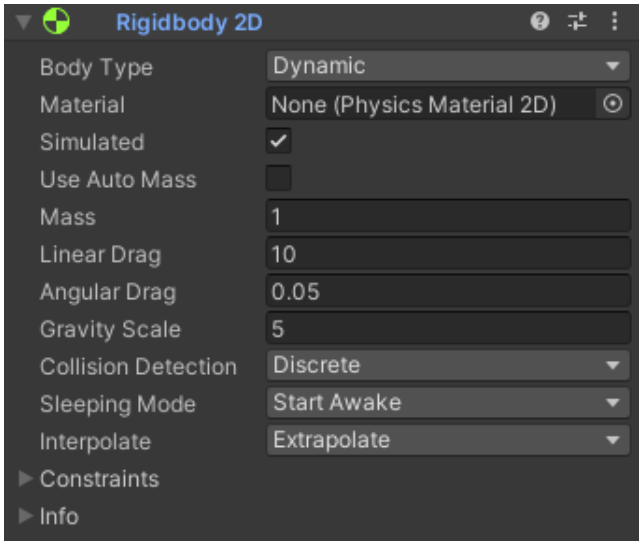
```
transform.position = Vector2.MoveTowards(transform.position, player.position);
```

Afin d'éviter que les ennemis ne tombent dans le vide lorsqu'ils suivent le joueur, un GroundCheck a été ajouté au Prefab des ennemis pour vérifier s'ils sont toujours au sol. Nous avons également ajouté une portée de détection pour que lorsque les ennemis ne sont plus en vue de la caméra, ils cessent de bouger afin de ne pas surcharger les performances du jeu. De plus, cette portée permet également d'éviter qu'un ennemi à l'autre bout de la carte commence à suivre le joueur dès le lancement du niveau.

Puis, nous avons ajouté la capacité des ennemis à attaquer les joueurs. Lorsque les ennemis sont suffisamment proches des joueurs, ils déclenchent leur attaque et infligent des dégâts aux joueurs. Cette fonctionnalité a été ajoutée pour rendre le gameplay plus intéressant et offrir une expérience plus dynamique aux joueurs. Tout cela a demandé beaucoup de temps et d'efforts, mais cela en valait la peine pour améliorer l'expérience de jeu globale.

```
if (distPlayerA < distPlayerB && distPlayerA > stoppingDistance && distPlayerA <
{
    FollowPlayer(targetA);
}
else if (distPlayerB < distPlayerA && distPlayerB > stoppingDistance && distPlaye
{
    FollowPlayer(targetB);
}
else
{
    HandleIdleAndAttack(distPlayerA, distPlayerB);
}
```

Finalement, la mise en place de l'intelligence artificielle pour que les ennemis suivent le joueur linéairement uniquement sur l'axe des x a été une tâche plus difficile que prévue. Nous avons dû passer par plusieurs itérations avant de trouver la solution idéale. Initialement, lors des premiers tests, les ennemis avaient tendance à voler ou à flotter, ce qui les rendait difficiles à utiliser. Après analyse, nous avons identifié que le problème résidait dans la gestion de la gravité qui n'était pas adaptée à cette situation. Ainsi, nous avons expérimenté différentes configurations de la gravité afin de trouver le juste équilibre entre une gravité réaliste et suffisamment puissante pour empêcher les ennemis de décoller du sol tout en suivant le joueur sur l'axe des x. Après plusieurs essais, nous avons finalement trouvé la configuration idéale qui a permis aux ennemis de suivre le joueur sans aucun problème.

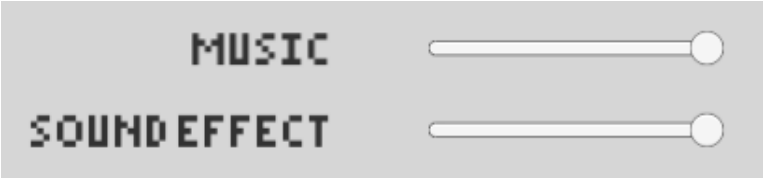


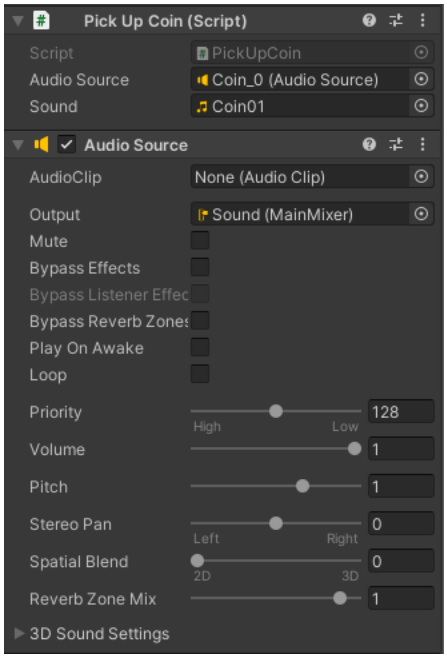
Une fois que toutes ces fonctionnalités ont été implémentées pour les ennemis, nous avons également entrepris la création d'un boss final. Pour cela, nous avons simplement réutilisé les mécaniques mises en place pour les ennemis, en ajustant certains paramètres pour rendre le boss plus difficile à vaincre. Cette approche nous a permis de capitaliser sur le travail déjà réalisé et d'offrir aux joueurs une confrontation finale unique avec un adversaire plus redoutable et complexe à affronter.

2.3 Effets sonores

En ce qui concerne la musique du jeu, nous avons voulu donner une ambiance sombre et oppressante qui rappelle l'enfer. Nous avons donc contacté une amie pour qu'elle nous compose une musique originale qui collerait parfaitement à l'univers du jeu. Grâce à elle, nous avons pu ajouter une touche personnelle à notre jeu et rendre l'expérience de jeu encore plus immersive pour les joueurs. Les effets sonores quant à eux sont essentiels pour renforcer l'immersion et donner une véritable dimension sonore à l'univers du jeu. Nous avons cherché des sons gratuits et de qualité sur internet afin de gagner du temps,

mais nous avons également apporté des modifications pour les adapter à nos besoins spécifiques. Pour que les joueurs puissent régler le volume sonore à leur guise, nous avons intégré une catégorie "Réglages" dans le menu du jeu, où le volume sonore de la musique et des effets sonores peut être ajusté individuellement. Cela permet aux joueurs de personnaliser leur expérience de jeu et de l'adapter à leurs préférences auditives.





2.4 Dialogues

Lors de la création des dialogues pour notre jeu, notre principale préoccupation était de respecter scrupuleusement le cahier des charges établi. Nous nous sommes inspirés de l'histoire que nous avions déjà élaborée, en veillant à ce que les dialogues s'intègrent parfaitement à l'univers du jeu. Notre objectif était de garantir la cohérence des dialogues avec le synopsis de l'histoire, mettant en scène deux personnages principaux, un ange et un démon, évoluant dans un univers mythologique captivant.

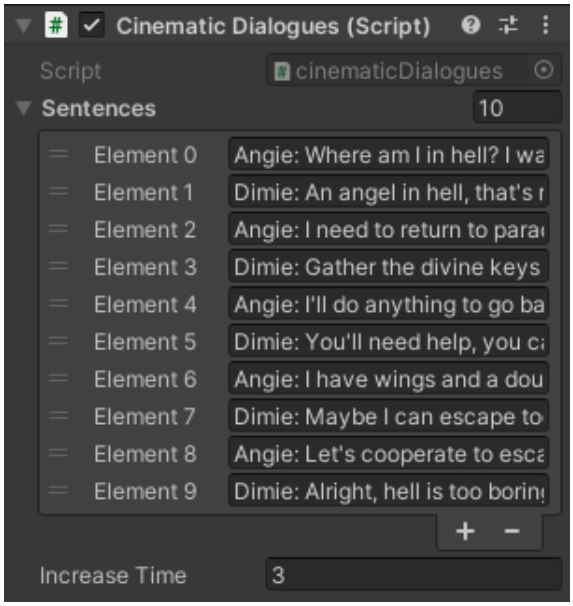
Nous avons accordé une grande attention à la rédaction des dialogues afin de garantir une expérience immersive aux joueurs. En respectant l'essence même du jeu, nous sommes convaincus que ces dialogues ajouteront une dimension supplémentaire à l'his-

toire et permettront aux joueurs de s'immerger encore plus profondément dans l'univers du jeu.

Afin de renforcer davantage l'impact des dialogues et de captiver les joueurs dès le début, nous avons décidé d'ajouter une cinématique d'entrée soigneusement réalisée. Cette cinématique a pour but de saisir les joueurs dès le début du jeu et de les plonger immédiatement dans l'univers captivant que nous souhaitons leur offrir. Grâce à des visuels époustouflants et à une narration engageante, cette cinématique d'entrée établit l'atmosphère et présente les enjeux de l'histoire de manière visuellement captivante.

En intégrant cette cinématique d'entrée, nous avons voulu renforcer la narration et offrir aux joueurs une expérience immersive dès les premiers instants du jeu. Nous voulions qu'ils se sentent véritablement transportés dans l'univers que nous avons créé, et cette cinématique est un élément clé pour y parvenir.

Ainsi, en combinant des dialogues soigneusement écrits et une cinématique d'entrée captivante, nous sommes convaincus d'avoir créé une expérience de jeu encore plus enrichissante et immersive. Ces éléments s'harmonisent parfaitement avec le reste de l'interface du jeu, créant ainsi une expérience utilisateur cohérente et engageante. En somme, notre objectif est de permettre aux joueurs de vivre une aventure inoubliable dans notre jeu en leur offrant une histoire captivante, des dialogues riches en émotions et une immersion totale dans l'univers du jeu dès les premiers instants.



2.5 Gestion du GIT

Au début de notre projet, nous avons commencé à travailler sans utiliser de branches distinctes dans notre dépôt Git. Cette approche a rapidement démontré ses limites, car

la gestion du code devenait complexe et peu organisée. Nous avons donc décidé d'adopter une méthode de travail plus structurée en utilisant des branches dédiées à chaque membre de notre équipe.

L'introduction des branches a considérablement amélioré notre efficacité et notre organisation. Chaque membre pouvait travailler sur sa propre branche, ce qui nous permettait de développer simultanément différentes fonctionnalités du projet. Cela nous a également offert la possibilité de suivre facilement les modifications effectuées par chaque membre et de fusionner les branches lorsque les fonctionnalités étaient prêtes à être intégrées dans la version principale.

Cependant, malgré les avantages des branches, nous avons rencontré des difficultés lors de la fusion de celles-ci. Les conflits de fusion sont devenus une réalité fréquente, ce qui entraînait parfois des dommages au code. Résoudre ces conflits était un processus complexe et chronophage, nécessitant une analyse approfondie du code et des ajustements manuels pour garantir la cohérence de l'ensemble du projet.

Pour mieux comprendre la gestion des branches et améliorer notre expertise dans l'utilisation de Git en équipe, nous avons recherché l'aide de notre enseignant, M. Hervot. Grâce à ses conseils avisés, nous avons identifié certaines erreurs dans notre approche initiale, notamment des branches mal positionnées. Malheureusement, en raison de restrictions de permissions, nous n'avons pas pu supprimer ou déplacer ces branches indésirables. Nous avons donc pris la décision de les abandonner et de créer de nouvelles branches afin de travailler de manière plus saine et organisée.

Cette expérience nous a permis de comprendre l'importance cruciale de la gestion des branches dans un projet collaboratif. Bien que la fusion des branches puisse parfois engendrer des conflits, nous avons appris à les gérer de manière plus efficace au fil du temps. En travaillant sur la résolution des erreurs et des conflits, nous avons acquis une connaissance plus approfondie du fonctionnement de Git en équipe.

Dans l'ensemble, cette expérience a été extrêmement bénéfique pour notre équipe. Nous avons appris à utiliser Git de manière plus professionnelle, à optimiser notre flux de travail et à résoudre les conflits de fusion avec plus de précision. Ces compétences nous seront inestimables pour nos futurs projets, nous permettant de travailler de manière plus collaborative et efficace tout en maintenant l'intégrité de notre code.

2.6 Ressentis

Au cours de ce projet, j'ai vécu une expérience enrichissante et formatrice, au travers de laquelle j'ai pu développer de nombreuses compétences utiles pour ma future carrière professionnelle. En tant que membre de l'équipe, j'ai acquis une expertise significative dans le domaine du développement et de la gestion de projets.

Tout d'abord, j'ai eu l'opportunité d'approfondir mes connaissances en utilisant Unity, un moteur de jeu puissant et polyvalent. Grâce à ce projet, j'ai pu explorer les différentes fonctionnalités offertes par Unity et j'ai développé une solide compréhension de son utilisation pour la création de jeux. J'ai également eu la chance de me familiariser avec des concepts spécifiques tels que l'implémentation d'une intelligence artificielle pour les ennemis du jeu. Les difficultés rencontrées dans ce domaine m'ont permis de mieux appréhender le fonctionnement des ennemis dans un jeu vidéo, ainsi que les aspects cruciaux de la conception de jeux, qu'il s'agisse du développement en réseau ou en local.

Cependant, parmi toutes les compétences acquises, la gestion de projet à l'aide de Git a été le point central de mon apprentissage. Avant ce projet, mes connaissances sur Git étaient assez limitées. J'avais une compréhension superficielle des commits, des branches et des autres fonctionnalités clés. Cependant, grâce à notre utilisation intensive de Git tout au long du projet, j'ai pu développer une maîtrise solide de cet outil de gestion de versions.

J'ai appris à utiliser les branches de manière stratégique, à effectuer des commits réguliers et bien documentés, et à gérer efficacement les conflits de fusion. Les difficultés que nous avons rencontrées avec la fusion des branches nous ont permis de mieux comprendre les subtilités de la gestion des conflits et de développer des compétences essentielles pour les résoudre de manière efficace.

Cette expérience m'a ouvert les yeux sur l'importance de la gestion de versions dans un projet de développement collaboratif. J'ai réalisé à quel point une bonne organisation et une utilisation appropriée des fonctionnalités de Git peuvent contribuer à la productivité et à la coordination de l'équipe. Je suis maintenant confiant dans ma capacité à utiliser Git de manière efficace pour gérer des projets futurs, en garantissant une meilleure collaboration et une traçabilité claire des modifications apportées.

En résumé, ce projet a été une expérience passionnante qui m'a permis de développer mes compétences en développement de jeux, en gestion de projet et en utilisation de Git. Je suis reconnaissant d'avoir eu l'occasion de travailler sur un projet aussi stimulant, et je suis convaincu que les connaissances et les compétences que j'ai acquises me seront précieuses dans ma future carrière professionnelle.

Chapitre 3

Leandro Tolaini

3.1 Introduction

Dès le début du projet et même avant d'avoir une idée concrète de notre jeu, le rôle de game designer était celui que je convoitais. En effet, mon implication dans l'expression de l'idée que j'avais de notre projet à travers différents visuels, et l'imagination des premiers scénarios clés qui formeront le lore du jeu ont mis notre équipe d'accord. J'ai donc pris ce rôle important à cœur afin de rendre l'expérience du jeu la plus immersive, cohérente et agréable possible pour les joueurs, en mettant en œuvre différents moyens détaillés dans cette partie. Ma partie du travail était très majoritairement basée sur cet aspect de game design, level design et lore, ainsi que sur toutes les mécaniques découlant de mon imagination ; mais j'ai également eu des tâches annexes assignées dans le cahier des charges. J'ai joué un rôle dans le développement des personnages, au niveau de leurs premiers déplacements et des animations. Enfin, je devais aider Noé à la conception des différents ennemis de notre jeu, mais cela s'est révélé être une tâche réalisable par un seul membre de l'équipe. Le game design, en revanche, a été une tâche bien plus éprouvante dans mon travail, en répondant aux mécaniques plus ou moins complexes que je me suis imposées d'implémenter.

3.2 Histoire

Angie, un ange, se retrouve coincée en enfer par erreur et dans l'impossibilité de s'en échapper, du moins, pas sans aide. C'est là que Dimie arrive, un démon habitué de l'enfer, qui voit dans la détresse de l'ange une opportunité de s'échapper également de l'enfer en coopérant avec elle. Cette coopération symbolise tout le principe du jeu : pour gagner et réussir à s'échapper de l'enfer, les joueurs doivent se servir des atouts des deux personnages à travers un environnement désolé, et des niveaux de plus en plus infernaux. C'est de là que je tire le nom de notre jeu : "HellScape", traduisible par "Paysage d'enfer" en français, mais qui est également un jeu de mots avec "Escape the Hell" ou "Hell Escape", "s'échapper de l'enfer" en français.

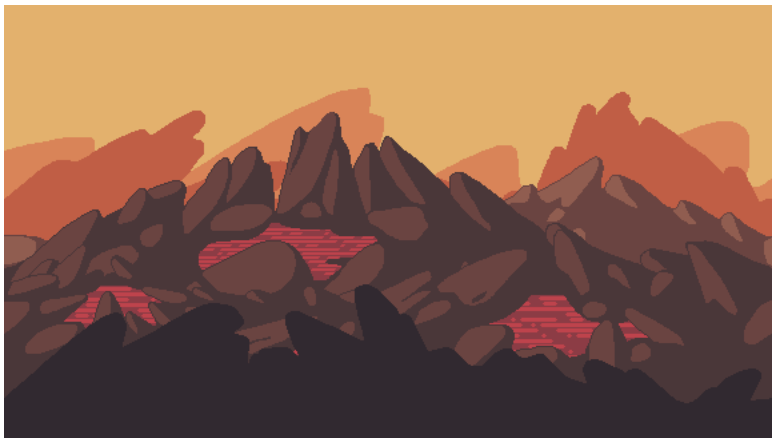
3.3 Assets

J'ai cherché des assets qui pouvaient correspondre à nos attentes : être cohérents avec l'histoire, être en pixel art pour rappeler le côté retro des platformers, et disposer d'animations pour les personnages.

Suivant ces critères, les assets retenus sont les suivants :

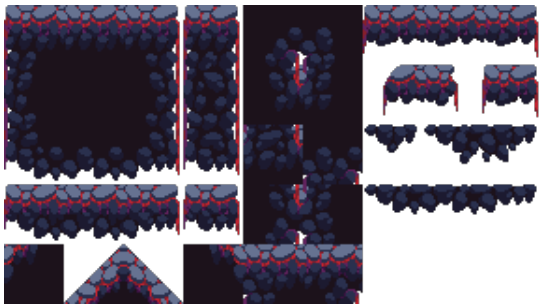
3.3.1 Le background

Cette image correspond parfaitement à l'ambiance recherchée pour notre jeu. Un paysage désolé, presque cauchemardesque, qui correspond parfaitement à une interprétation de l'enfer qui colle à notre jeu. Les tons sombres et les couleurs chaudes sont en adéquation avec l'ambiance que nous voulons transmettre dans Hellscape.



3.3.2 Les fondations

Ce seront les briques de tous nos niveaux, comme nous le verrons dans les premiers exemples. Elles ont été fournies avec le background, donc séléctionnées pour les mêmes raisons.



3.3.3 Le Démon (Personnage 1)

C'est un des héros de l'aventure, et étant un habitant de l'enfer, il se devait d'avoir un design similaire à l'environnement, et également aux autres habitants de l'enfer, qui seront les ennemis des deux héros cherchant à s'enfuir. Il devait posséder une arme pour illustrer sa capacité : attaquer et tuer les ennemis.

3.3.4 L'Ange (Personnage 2)

Ce personnage se devait d'avoir un gros contraste avec l'environnement : l'ange est lui un habitant du paradis qui se retrouve en enfer par erreur. C'est pour cela qu'il est très lumineux et gracieux. Il devait posséder des ailes pour illustrer sa capacité : être très agile (double saut).

3.3.5 Les ennemis

Nous disposons d'une certaine variété d'ennemis grâce à cet asset : il ne seront surement pas tous implémenter dans notre jeu mais nous laisse une bonne possibilité d'évolution.

3.3.6 Les items

Différents objets sont également à notre disposition à but décoratif, ou même afin de développer des mécaniques pour nos niveaux.

3.4 Création

Malgré cela, j'ai eu un travail de création de sprites et d'animations pour répondre aux exigences techniques et visuelles du jeu. J'ai donc passé de longs moments à concevoir nos propres éléments graphiques.

Pour ce faire, j'ai dû maîtriser des logiciels tels que Pixel Art et Aseprite, et développer mes compétences en dessin en suivant de nombreux tutoriels en ligne. Voici, entre autres, mes créations :

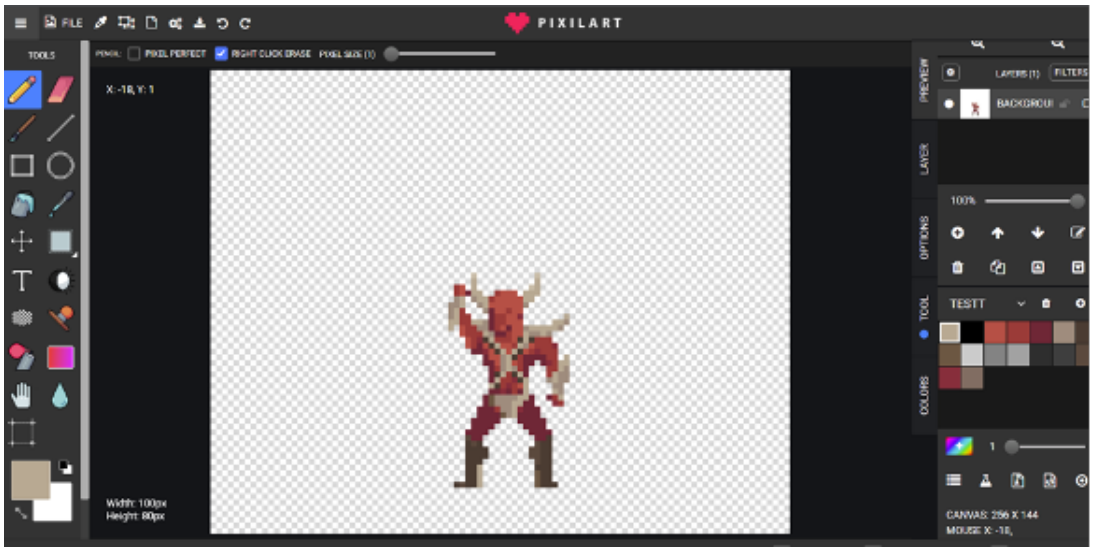
Animation d'escalade des deux personnages :

La première étape a donc dû être de créer moi-même les animations manquantes. Pour cela, je suis allé sur un site spécialisé :

PIXILART

De là j'ai pu m'aider d'un modèle de base du pack d'asset :

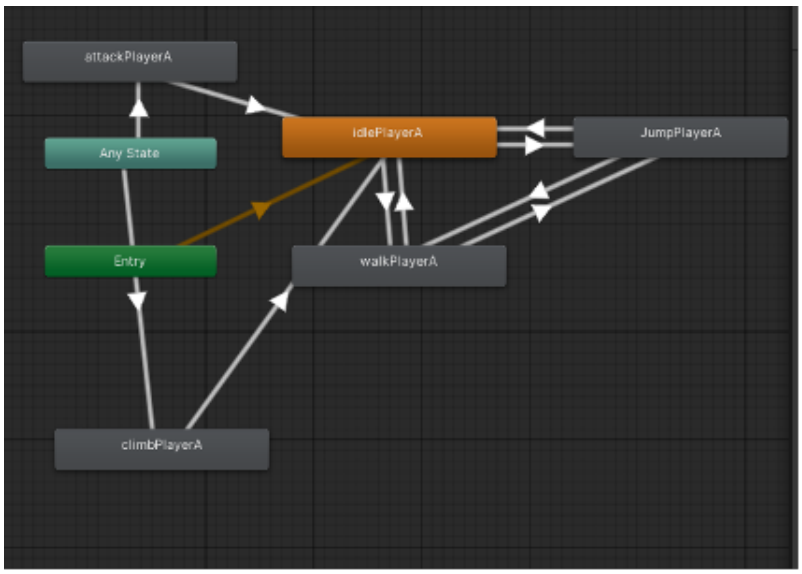
Et avec mon peu de compétences en dessin et beaucoup plus de temps que j'aurais imaginer j'ai enfin pu arriver à un résultat satisfaisant :



Ensuite il m'a suffit de retourner l'image sur l'axe x pour avoir deux frames qui mise à bout donnaient un rendu satisfaisant pour une animation d'escalade.

J'ai fait de même pour l'ange :

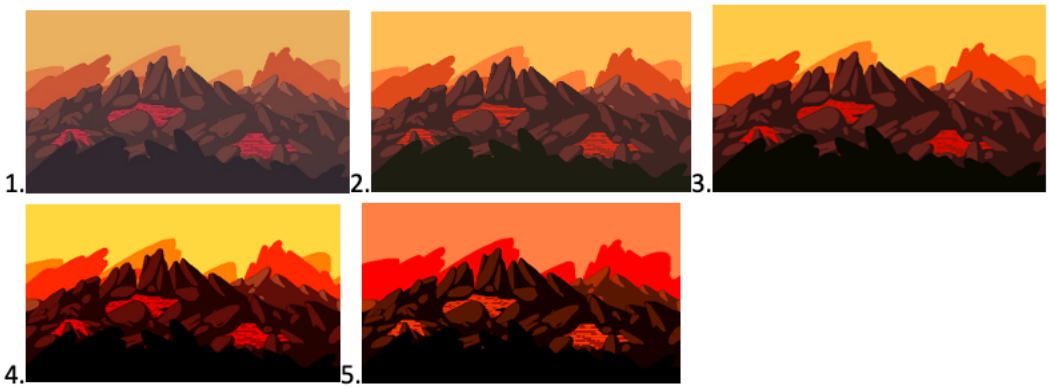
Une fois toutes les animations faites dans Unity, je devais réussir à faire en sorte qu'elles soient jouées au bon moment, en me servant de l'animator. Le schéma pour du démon, assez similaire à celui de l'ange, ressemble à ça :



3.4.1 Les background

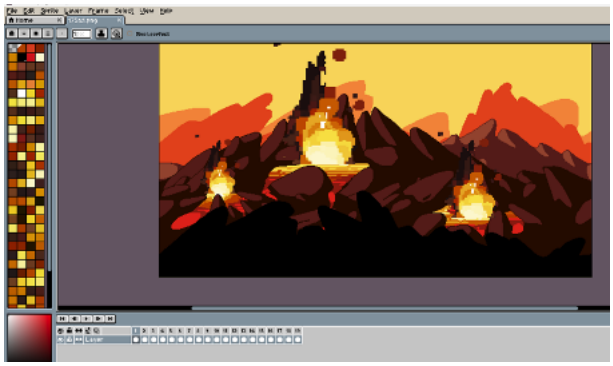
Je voulais m'approprier les éléments graphiques et en faire quelque chose de plus intéressant que de simplement les copier-coller tout au long des niveaux. Le renouvellement du design et du gameplay était une préoccupation importante pour moi, c'est pourquoi j'ai créé des déclinaisons de cet élément graphique.

Je voulais donner une identité plus forte au jeu, tout en essayant de « recycler » nos asset. J'ai donc eu l'idée de modifier en peu le background à chaque niveau :



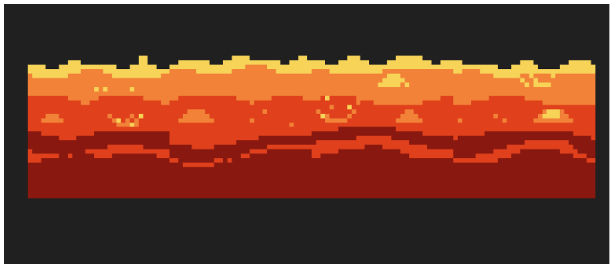
Plus les niveaux avancent, plus l'arrière-plan a de contraste, donnant un effet bien plus vif et cauchemardesque, comme si la température augmentait, ce qui symbolisera aussi l'augmentation de la difficulté des niveaux.

Tout cela nous mène au bouquet final, le dernier niveau. Je voulais qu'il soit spécial et particulièrement marquant, c'est pourquoi j'ai entrepris un travail assez fastidieux d'animation de cet arrière-plan sur Aseprite. J'ai créé une animation personnalisée d'une vingtaine d'images qui représente une éruption volcanique en arrière-plan.



3.4.2 La lave

Nos éléments graphiques ne comportaient pas un élément crucial dans notre univers volcanique, qui aurait pourtant pu apporter des mécaniques intéressantes au projet : des tiles animées de lave. J'ai donc repris le travail pour créer les nôtres. Cela m'a pris énormément de temps, bien plus que ce que j'aurais pu imaginer, mais le résultat est très satisfaisant : des tiles de 32 pixels sur 32, animées en 4 frames.



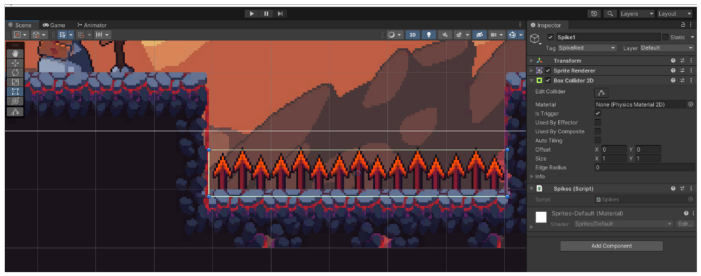
À partir de ce moment, j'avais toutes les clés en main pour créer des niveaux à la fois stimulants et esthétiques.

3.5 Niveaux et mécaniques

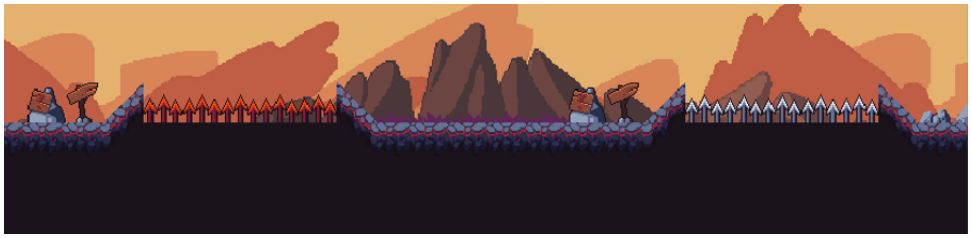
Pour nous, il était essentiel que les niveaux aient une continuité et un lien logique entre eux. J'ai donc décidé de ne développer que cinq niveaux, mais de veiller à ce qu'ils soient de haute qualité, avec chacun leur propre identité visuelle et des mécaniques qui les caractérisent. Ils devaient devenir de plus en plus difficiles et spectaculaires, tout en ayant leur propre histoire.

3.5.1 Niveau 1

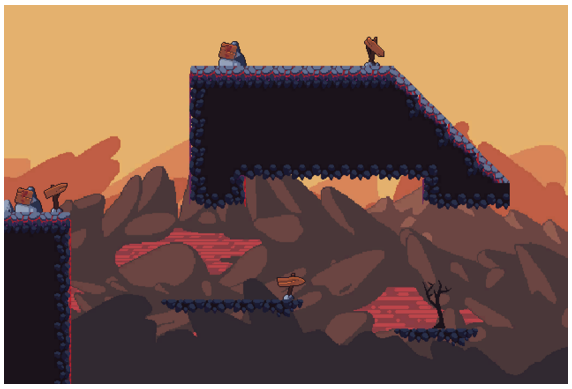
Le premier niveau est conçu pour être très simple, comme un didacticiel. Le joueur est guidé par des indications et il introduit les mécaniques de base du jeu qui seront réutilisées par la suite :



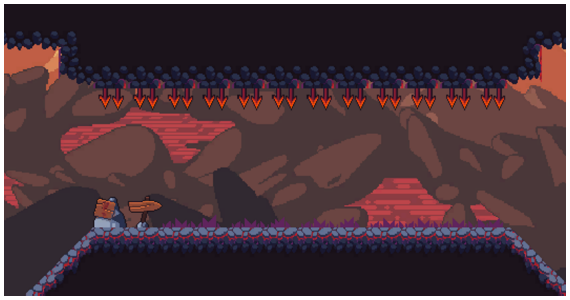
Le premier obstacle sert à prendre en main la physique du jeu et la mécanique de saut.



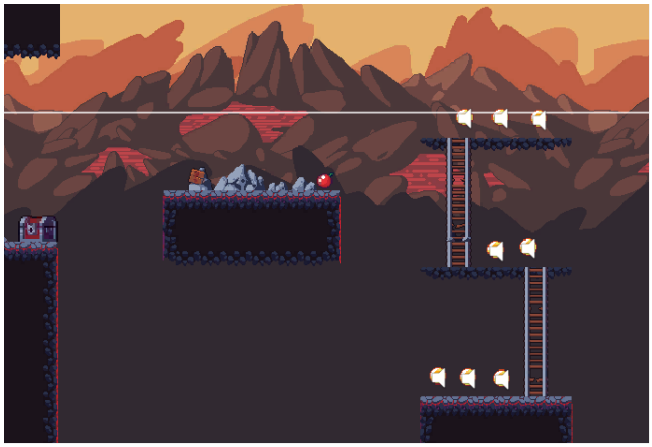
Ensuite, il y a deux types de piques : les rouges sont inoffensifs pour le démon et mortels pour l'ange et inversement pour les piques blancs.



Ensuite, l'ange sera le seul à pouvoir atteindre une clé qui se trouvera en haut de la plateforme (clé nécessaire pour réussir le niveau) grâce à sa capacité de double sauts tandis que le démon devra passer par en dessous.



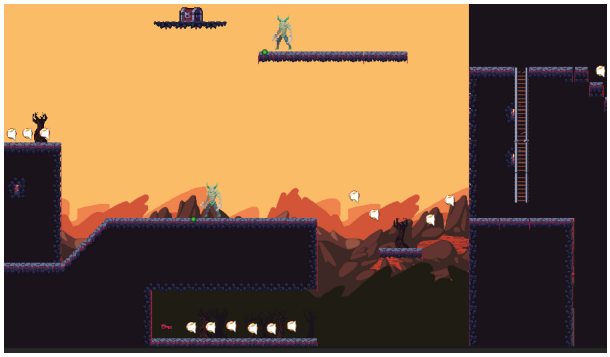
Tandis que au niveau de ce dernier obstacle, le démon devra libérer le passage à l'ange en tuant un ennemi qui sera au bout du couloir grâce à sa capacité à se battre, afin d'atteindre la fin du niveau.



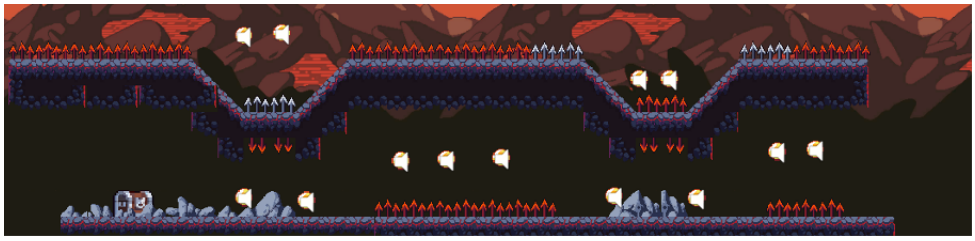
Les mécaniques des coffres, les premiers items et les échelles y sont également introduits.

3.5.2 Niveau 2

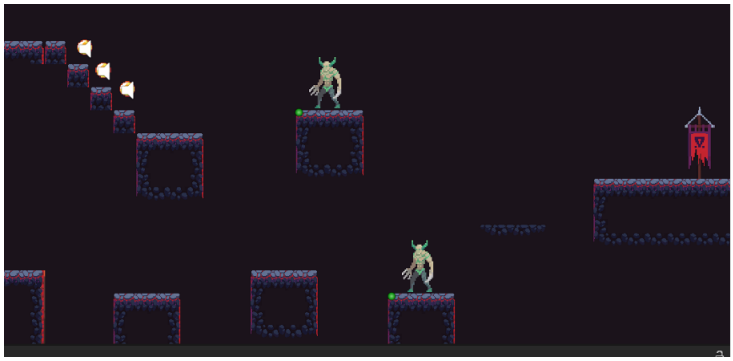
Le niveau 2 se veut comme le premier vrai niveau du jeu, sa conception se voulait plus travaillée. La mécanique des coffres est un bon moyen de cacher des clés dans le niveau, obligeant les joueurs à chercher comme ici :



La clé se trouve dans un passage secret en bas, visible grâce au double saut de l'ange. Le démon devra combattre les ennemis et il faudra aller contre le sens logique du niveau (de gauche à droite) pour atteindre le coffre.



Des parties deviennent plus techniques, comme par exemple en se servant des piques pour atteindre les coffres.



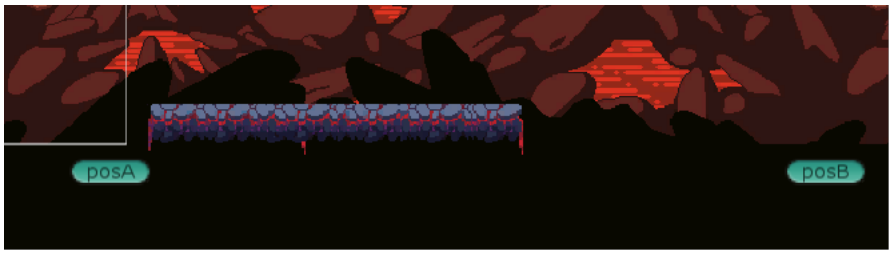
Un genre de grotte est présent dans le niveau, où de l'agilité est nécessaire. Ce genre de passage sera réutilisé et adapté.

Il faut garder en mémoire que le niveau 2 reste assez simple, même si il marque un assez gros contraste avec le premier.

3.5.3 Niveau 3

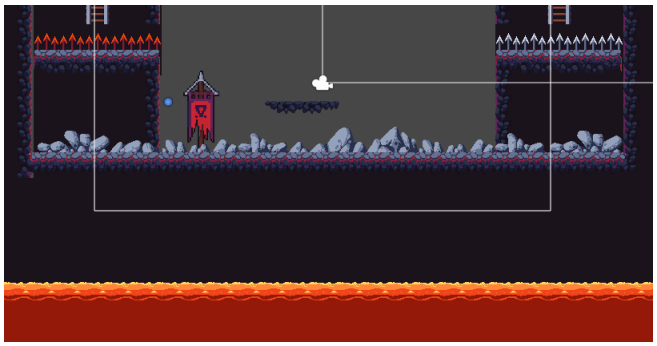
Tandis que l'arrière-plan continue à évoluer, comme si la température augmentait, cela symbolise la progression du parcours des personnages. L'objectif de ce niveau est d'atteindre un volcan et d'y entrer.

Le niveau 3 introduit de nouvelles mécaniques : les plateformes mobiles. Elles représentent un véritable défi pour les joueurs et complexifient considérablement le gameplay. Ce niveau est conçu de manière plus verticale que les précédents.



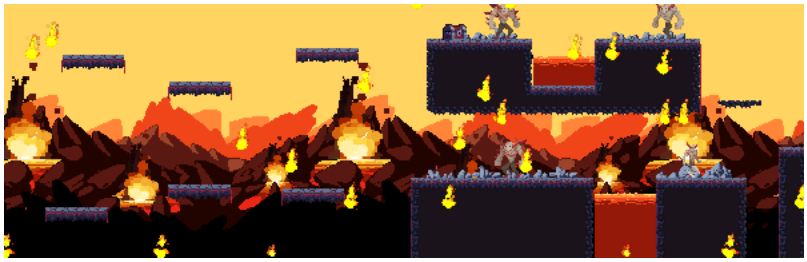
3.5.4 Niveau 4

Le niveau avant-dernier a été un véritable défi à développer. Il met en scène les personnages au cœur du volcan qui se trouvait sur leur chemin, et d'où ils doivent s'échapper. Cependant, le volcan commence à entrer en éruption, entraînant une montée de lave qui force les deux joueurs à respecter un timing précis pour ne pas se faire rattraper. Tout au long du niveau, ils seront sous pression, dans un environnement très vertical et chaotique. Certaines plateformes bougent, tandis que d'autres se détruisent si les personnages y restent trop longtemps. Tout est fait pour les faire chuter.

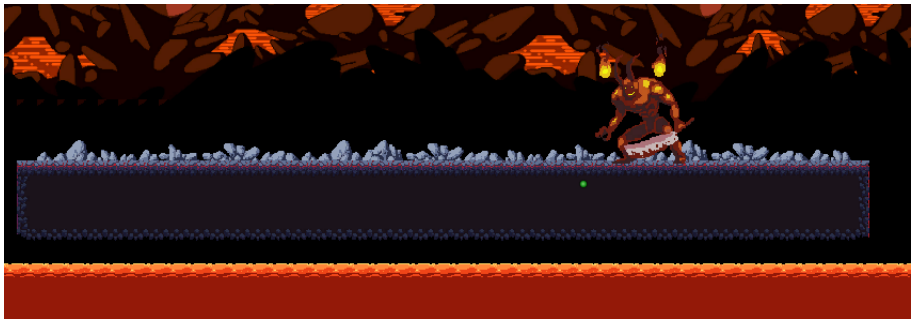


3.5.5 Niveau 5

Le dernier niveau se veut spectaculaire : les héros ont réussi à s'échapper du volcan, mais de terribles éruptions dans le paysage volcanique entraînent la chute aléatoire de boules de feu sur le niveau. Les joueurs doivent les éviter tout en progressant dans un niveau complexe. De plus, de la lave est présente un peu partout, ce qui tue instantanément l'un ou l'autre des deux personnages.



Enfin, ils atteignent l'arène et affrontent le boss final, le gardien des enfers, dans un combat spectaculaire.



Une fois vaincu, le jeu est gagné et les héros peuvent alors quitter définitivement l'enfer.

3.6 Personnages

3.6.1 Déplacements joueurs

J'ai également dû aider mes camarades dans le développement des personnages.

Pour de commencer à attribuer les premières animations aux personnages, j'ai du manipuler les premiers scripts de déplacements de ces derniers, de manière assez basique. J'ai commencé par appliquer les collisions de ma tilemap et de mes personnages, ainsi que la gravité pour avoir la première physique du niveau.

Ensuite j'ai pu écrire mon premier script de déplacement.

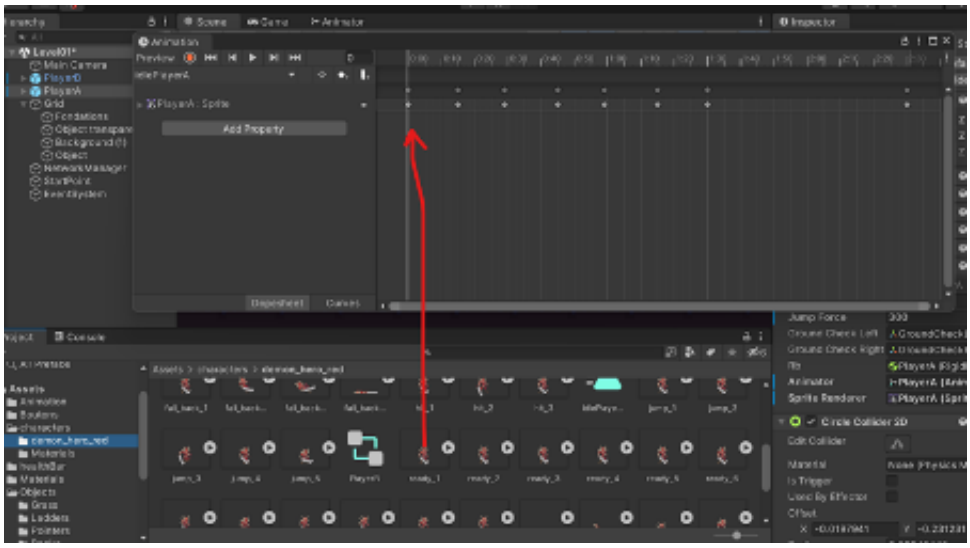
```
public float moveSpeed;
    public float jumpForce;

    private bool isJumping;
    private bool isGrounded;

    public Transform groundCheckLeft;
    public Transform groundCheckRight;
```

Les deux premières variables sont des constantes arbitraires pour déterminer respectivement la vitesse et la puissance de saut du personnages ; ensuite les deux booléens suivant sont déterminés grâce aux objets groundCheckLeft et groundCheckRight.

Ils sont représentés par ces deux cercles juste en dessous du personnage, et servent à détecter si celui-ci est en contact avec une structure ou non.

```
if (Input.GetButtonDown("Jump") && isGrounded)
        {
            isJumping = true;
        }
```

3.6.2 Animations

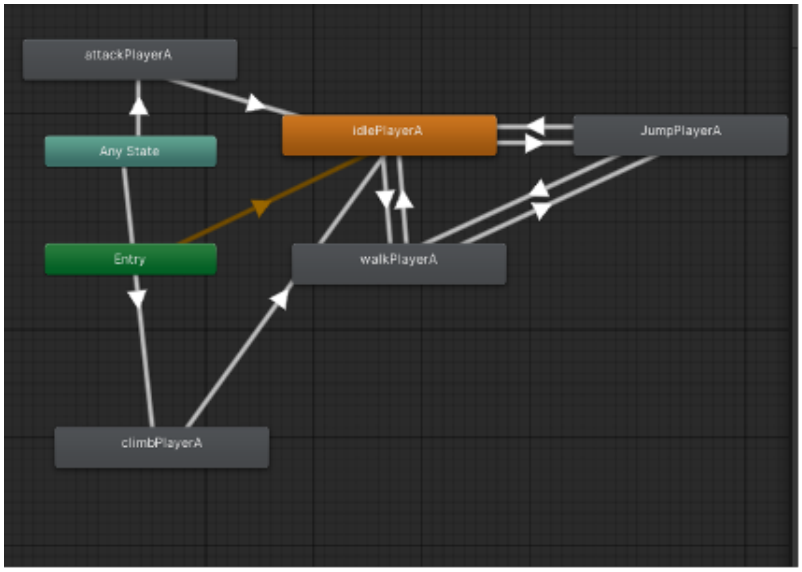


Cette partie est assez redondante : pour chaque personnage, il faut découper chacune de ses frames de la tilesheet, pour ensuite les glisser dans l'outil Animation de Unity à des intervalles de temps que j'ai choisi afin qu'elles soient jouées dans le bon contexte.

Pour se faire, l'ajustement des scripts de déplacement était nécessaire. Prenons l'exemple de l'animation de marche d'un personnage : pour avoir un bon rendu, il fallait assurer un renversement sur l'axe x de l'image du personnage si sa vitesse est négative (si l'utilisateur veut le faire avancer de droite à gauche).

```
void Flip(float _velocity)
    {
        if (_velocity > 0.1f)
        {
            spriteRenderer.flipX = false;
        }
        else if (_velocity < -0.1f)
            spriteRenderer.flipX = true;
    }
```

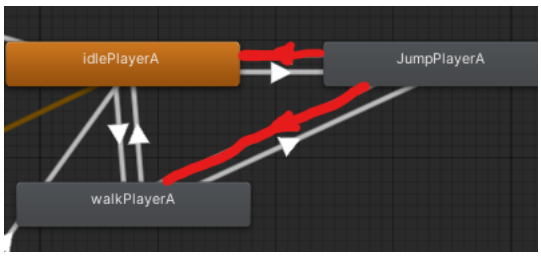
La méthode Flip est donc venue s'ajouter à mon script. l'argument _velocity représente la vitesse actuelle du personnage.

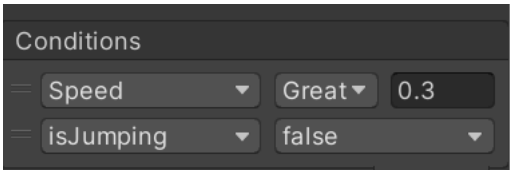


Chaque lien représente une transition possible d'une animation à une autre : par exemple, l'animation d'attaque et celle d'escalade sont liées au bloc « Any State » qui signifie qu'elles peuvent être jouées qu'importe l'animation actuelles. De même l'animation de marche ne peut être jouée qu'après celle de saut ou d'idle.

Ces transitions fonctionnent grâce à des conditions utilisant des variables qui peuvent être de plusieurs types différents. Je me suis servi des conditions suivantes :

Par exemple, j'ai besoin du booléen isJumping et du float Speed pour savoir quelle animation jouer après un saut. Si isJumping est à false, et que la vitesse est supérieure à une certaine valeur, alors l'animation de marche doit être jouer, sinon cela sera l'animation d'idle.





Il suffisait ensuite de modifier les scripts aux bons endroits grâce à ce genre d'instructions :

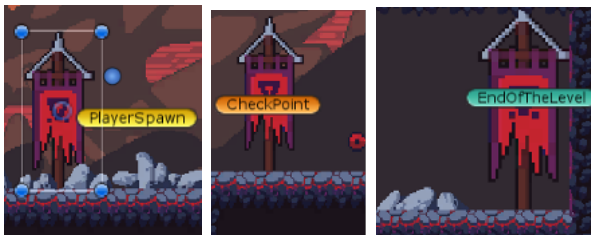
```
animator.SetBool("isJumping", false);
animator.SetFloat("Speed", Mathf.Abs(moveInput));
```

3.7 Aspects techniques

3.7.1 Les pièces

J'ai aussi dû implémenter de nouvelles choses qui seront utiles dans tous les niveaux.

Par exemple le spawn des joueurs, le checkpoint leur permettant de ne pas recommencer le niveau s'ils meurent à un certain stade, ou encore les pièces qu'ils peuvent ramasser dans les niveaux.



Toutes ces zones sont symbolisées par des drapeaux animés.

Le fonctionnement de ce genre d'objet peut presque se résumer à une seule fonction, comme pour le checkpoint par exemple :

```
private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.CompareTag("PlayerA") || collision.CompareTag("PlayerB"))
    {
        playerSpawn.position = transform.position;
    }
}
```

Si le Joueur A ou le Joueur B entrent en contact avec le Checkpoint, alors la position du spawn devient celle de l'objet.

3.7.2 Les pièces

Le ramassage et stockage du nombre de pièces est aussi géré par un script résumable en une fonction principale :

```
private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.CompareTag("PlayerA")||collision.CompareTag("PlayerB"))
    {
        AudioManager.instance.PlayClipAt(sound, transform.position);
        Inventory.instance.AddCoins(1);
        Destroy(gameObject);
    }
}
```

Si le Joueur A ou le Joueur B entrent en contact avec la pièce, on joue un son, on ajoute une pièce à l'inventaire et on détruit cette dernière.

3.7.3 Les différentes plateformes

Nous ne voulions pas que notre jeu soit uniquement composé de plateformes statiques, au contraire on voulait que jouer aux derniers niveaux soit un réel défis pour les joueurs, et pour cela rien de mieux que des plateformes qui bougent ou qui tombent.

```
if (Vector2.Distance(transform.position, posA.position) < .1f)
    targetPos = posB.position;
        if (Vector2.Distance(transform.position, posB.position) < .1f)
            targetPos = posA.position;
        transform.position = Vector2.MoveTowards(transform.position,
        targetPos, Speed * Time.deltaTime);
```

Grace à deux gameObject posA et posB, qui servent à baliser la plateforme et au script ci-dessus, elle peut effectuer des aller-retours à une vitesse définie par la variable Speed.

3.8 Ressenti

Ces 5 mois de projet ont été très riches et très éprouvants : nous avons appris beaucoup de choses, bien sûr, mais nous avons surtout été immergés dans une expérience concrète de travail de groupe, sous la pression des délais et "esclaves" du cahier des charges et de nos propres ambitions. Pour ma part, je suis globalement très satisfait. J'ai réussi à relever les défis techniques que représentaient mes nombreuses idées. En tant que Game Designer, l'expérience de l'utilisateur et l'évolution du gameplay me tenaient vraiment à cœur. Je voulais être fier du jeu, et c'est le cas. Je voulais éviter à tout prix de me reposer sur mes lauriers en proposant des niveaux redondants et linéaires dès que j'avais réussi à prendre en main la création de niveaux dans le moteur Unity. Au contraire, j'ai cherché à me challenger, à me détacher des tutoriels et à créer quelque chose d'unique avec notre équipe et l'expérience accumulée. J'ai essayé d'implémenter de nouvelles mécaniques sans avoir la moindre idée de comment faire, juste parce qu'elles me semblaient enrichissantes pour le jeu et son histoire.

Nous avons fait face à de nombreux problèmes, parfois très décourageants, qui nous ont moralement beaucoup affectés, nous faisant dépenser notre temps et notre énergie sur le projet sans les compter. Mais ce sont tous ces obstacles qui font que, malgré l'imperfection certaine du jeu, je suis fier du travail accompli durant ce semestre.

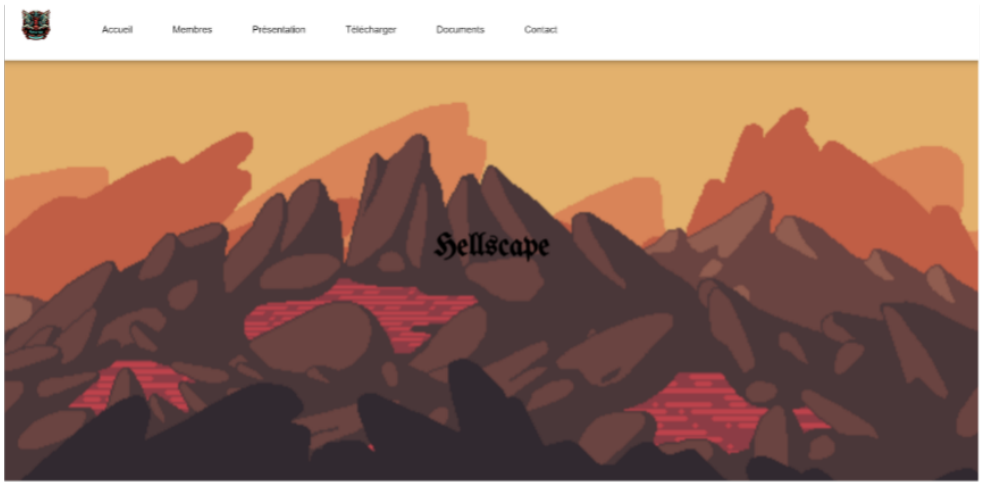
En conclusion, une fois de plus, je suis heureux du travail que nous avons produit. Le jeu a plu à notre entourage, et ce projet est une fierté, même si le parcours a été semé d'embûches.

Chapitre 4

Clément Grégoire

4.1 Site Web

La création du site web a été ma première tâche assignée dans le cadre de ce projet. Pour ce faire, j'ai utilisé les langages HTML et CSS. Le processus de conception du design a nécessité plusieurs itérations avant d'aboutir à une version qui satisfaisait l'ensemble des membres du groupe. Par conséquent, nous avons opté pour un site web composé d'une seule page, où les informations sont réparties dans différentes sections, à savoir : l'accueil, les membres, la présentation du projet, le téléchargement du jeu, les documents et les contacts.



Une barre de navigation, toujours visible à l'écran, permet d'accéder facilement à ces différentes sections. Lors de la première soutenance, la structure générale du site était en place, mais le CSS n'était pas entièrement implémenté sur l'ensemble de la page.

Lors de la deuxième soutenance, le site était entièrement achevé, avec un style et une mise en page entièrement implémentés, et les documents étaient à jour. Le site n'était plus seulement disponible en local, nous l'avons hébergé sur GitHub à l'adresse suivante : https://noecrn.github.io/.

Pour la dernière soutenance, le jeu est disponible sur le site et les documents ont été mis à jour. Un nouveau bouton de sélection de langue a été ajouté, redirigeant vers une version anglaise de la même page pour la présentation en anglais.

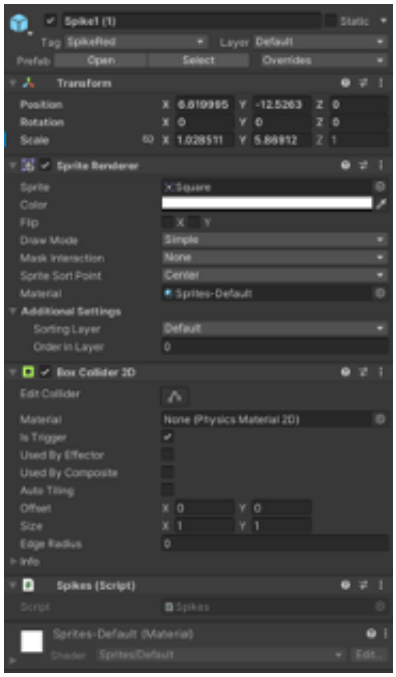
La conception de ce site n'a pas suscité ma passion, j'ai préféré me concentrer sur le jeu et l'utilisation d'Unity. Cependant, n'ayant jamais réalisé de site web auparavant, cette tâche m'a permis de découvrir cette activité et de me familiariser avec les langages HTML et CSS.

4.2 Mécaniques

Une autre de mes missions consistait à développer les mécaniques du jeu. Lors de la première évaluation intermédiaire, je me suis uniquement occupé du site web et je n'avais donc pas utilisé Unity avant la deuxième évaluation.

4.3 Les piques

Pour la deuxième soutenance, j'ai été chargé de mettre en place les piques et de gérer les dégâts qu'ils infligent aux joueurs. Le démon subit des dégâts uniquement s'il entre en contact avec les piques blancs, tandis que l'ange subit des dégâts s'il entre en contact avec les piques rouges. Pour réaliser cela, j'ai ajouté un composant "Box Collider" autour des piques et j'ai attribué des tags indiquant leur couleur (rouge ou blanche).



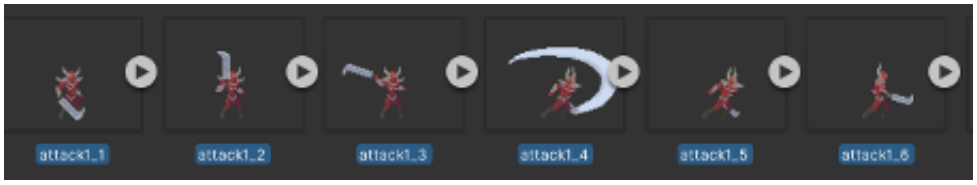
En ce qui concerne la partie du code, il suffit de détecter les collisions du "Box Collider" et de vérifier si le GameObject correspond au joueur dont la couleur diffère de celle des piques. Si c'est le cas, des dégâts sont infligés au joueur ; sinon, aucune action n'est entreprise.

```
void OnTriggerStay2D(Collider2D col)
{
    if (col.gameObject.tag == "PlayerA" && gameObject.tag == "SpikeWhite")
    {
        PlayerHealth.instance.TakeDamage(1, "PlayerA");
    }

    if (col.gameObject.tag == "PlayerB" && gameObject.tag == "SpikeRed")
    {
        PlayerHealth.instance.TakeDamage(1, "PlayerB");
    }
}
```

4.4 L'attaque

Lors de la deuxième soutenance, j'ai implémenté l'attaque du démon. Dans le script "MovePlayerA", dans la méthode "Update", nous détectons si la touche "K" est pressée. Si c'est le cas, nous appelons la méthode "Attack", qui déclenche une animation et inflige des dégâts à tous les ennemis se trouvant à portée du joueur. Cependant, un problème survient : le joueur peut appuyer de manière répétée sur la touche d'attaque, ce qui lui permet d'infliger de nombreux dégâts aux ennemis en très peu de temps. De plus, l'ani-

mation recommence à chaque nouvelle attaque, ce qui donne un aspect visuellement peu réaliste.

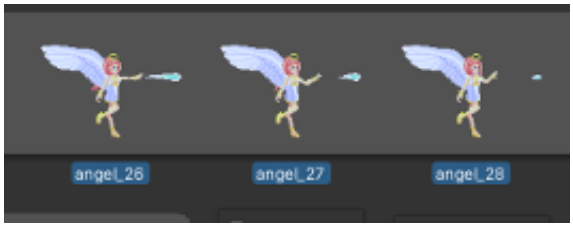


Pour remédier à ce problème, j'ai ajouté un temps de récupération, appelé "cooldown", pendant lequel l'attaque ne peut pas être déclenchée même si le joueur appuie sur la touche "K". J'ai donc introduit un nouveau script appelé "CooldownA". Dans ce script, j'ai ajouté deux variables : "cooldownTime" et "next". "cooldownTime" correspond au temps pendant lequel la touche d'attaque est désactivée et est par défaut définie à 0,5 seconde. "next" est également définie par défaut à 0.

Dans la méthode "Update", nous vérifions si le temps écoulé depuis le début de la partie est supérieur à la valeur de "next". Pour la première attaque, cette condition est toujours vérifiée car "next" est initialisé à 0. Si la condition est satisfaite, nous vérifions si le joueur appuie sur la touche "K". Si c'est le cas, nous appelons la méthode "Attack" de "MovePlayerA" et nous redéfinissons la valeur de "next" comme étant le temps actuel plus la valeur de "cooldownTime". La première condition ne sera donc plus vérifiée pendant 0,5 seconde, empêchant ainsi le démon d'effectuer une nouvelle attaque même si le joueur appuie sur la touche "K".

```
void Update()
{
    if (Time.time > next)
    {
        if (Input.GetKeyDown(KeyCode.K))
        {
            MovePlayerA.instance.Attack();
            next = Time.time + cooldownTime;
        }
    }
}
```

Nous avons également décidé de donner à l'ange la capacité d'attaquer pour se défendre contre les ennemis. J'ai donc répété le même processus pour l'ange, y compris l'animation, la méthode "Attack" et le cooldown. Le démon peut tuer les ennemis en 2 coups, tandis que l'ange est moins puissant et a besoin de 3 coups pour éliminer un ennemi.



4.5 L'esquive

Nous avons décidé d'ajouter la possibilité au joueur d'esquiver les attaques des ennemis. Pour cela, j'ai créé un nouveau script appelé "DodgePlayerA". Le fonctionnement de ce script est similaire à celui du cooldown. Nous avons deux variables, "dodgeTime" et "nextDodge". J'ai également ajouté un booléen "isDodging" dans le script "MovePlayerA".

Dans la méthode "Update" de "DodgePlayerA", si le joueur appuie sur la touche "J" et que "isDodging" est false, nous mettons "isDodging" à true et définissons la valeur de "nextDodge" comme étant le temps actuel plus la valeur de "dodgeTime". Enfin, si le temps écoulé dépasse la valeur de "nextDodge", nous remettons "isDodging" à false.

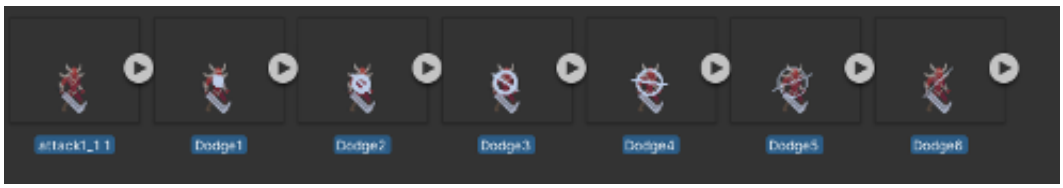
```
public void Update()
{
    if (!MovePlayerA.instance.isDodging && Input.GetKeyDown(KeyCode.J))
    {
        MovePlayerA.instance.isDoging = true;
        nextdodge = Time.time + dodgeTime;
    }

    if (Time.time > nextdodge)
    {
        MovePlayerA.instance.isDoging = false;
    }
}
```

Dans le script "PlayerHealth", dans la méthode "TakeDamage", nous vérifions si "isDodging" est true. Si c'est le cas, nous déclenchons l'animation d'esquive. Sinon, nous appliquons les dégâts au joueur.

```
if (PlayerTag == "PlayerA")
{

    if (!MovePlayerA.instance.isDoging)
```

```
        {
            if (photonView.IsMine && gameObject.CompareTag(PlayerTag))
            {
                currentHealth -= damage;
                healthBarInstance.SetHealth(currentHealth);
                // Appeler la méthode TakeDamage sur tous les clients connectés
                photonView.RPC("UpdateHealth", RpcTarget.AllBuffered, currentHealth);
            }
        }
        else
        {
            MovePlayerA.instance.animator.SetTrigger("DodgeTr");

        }
    }
```

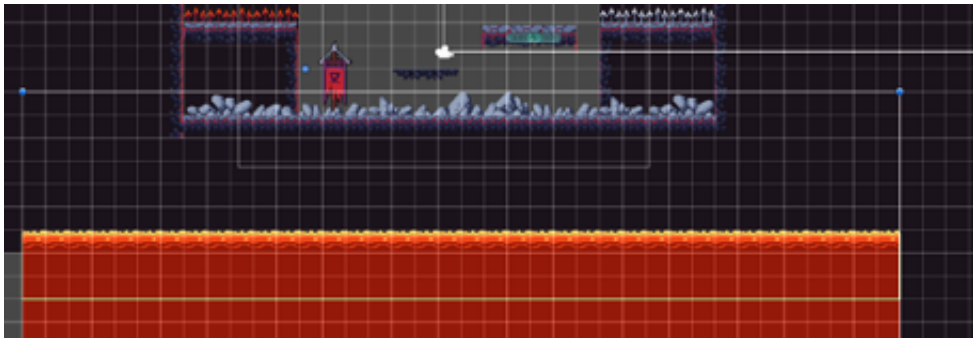
Comme le pack d'assets que nous avons acheté ne contient pas d'animation d'esquive, j'ai utilisé une autre animation existante correspondant au ramassage d'un objet. J'ai ensuite superposé cette animation sur une image du démon à l'aide du logiciel Gimp. De cette manière, j'ai créé une animation d'esquive personnalisée en combinant ces différents éléments visuels.



Nous répétons le même processus pour l'ange. Ainsi, les deux joueurs peuvent effectuer une esquive en appuyant sur la touche "J" au moment d'une attaque ennemie.

4.6 La lave

Dans notre jeu, la lave doit instantanément tuer le joueur s'il la touche. Pour réaliser cela, j'ai créé un script pour la lave, auquel j'ai ajouté un composant "BoxCollider". Lorsque nous détectons une collision avec ce box collider, nous retirons immédiatement 100 points de vie au joueur.



Le script associé à la lave est très similaire à celui des piques. Il fonctionne de la manière suivante : lorsqu'une collision est détectée avec le box collider de la lave, une action est déclenchée pour retirer instantanément 100 points de vie au joueur. Cela garantit que si le joueur entre en contact avec la lave, il perd immédiatement la totalité de ses points de vie.

```
void OnTriggerStay2D(Collider2D col)
{
    if (col.gameObject.tag == "PlayerA")
    {
        PlayerHealth.instance.TakeDamage(100, "PlayerA");
    }

    if (col.gameObject.tag == "PlayerB")
    {
        PlayerHealth.instance.TakeDamage(100, "PlayerB");
    }
}
```

4.7 Le double-saut

Une autre capacité de l'ange est la possibilité de réaliser un double saut. J'avais déjà travaillé sur cette fonctionnalité avant la deuxième soutenance, mais elle n'était pas encore disponible.

Dans le script "MovePlayerB", nous avons deux attributs : "jumpLeft" et "maxJump", dont la valeur par défaut est 2.

Dans la méthode "Update", nous détectons si le joueur appuie sur la touche d'espace. Si c'est le cas, nous vérifions la valeur de "jumpLeft". Si elle est égale à 2, nous appliquons une force sur l'axe Y de l'ange et nous définissons "jumpLeft" à 1. Cela permet d'effectuer le premier saut.

Si "jumpLeft" est égal à 1, cela signifie que le joueur a déjà réalisé un saut. Dans ce cas, nous appliquons à nouveau une force sur l'ange et nous décrémentons la valeur de "jumpLeft". Cela permet d'effectuer le deuxième saut, c'est-à-dire le double saut.

```
if (Input.GetKeyDown(KeyCode.Space))
{
    if (jumpLeft == 2)
    {
        rb.AddForce(new Vector2(0f, 4), ForceMode2D.Impulse);
        jumpLeft = 1;
    }

    if (jumpLeft == 1)
    {
        rb.AddForce(new Vector2(0f, 2.1f), ForceMode2D.Impulse);
        jumpLeft = 0;
    }
    //Envoi d'un appel de fonction RPC pour synchroniser le saut avec les autre
    photonView.RPC("Jump", RpcTarget.Others);
}
```

Enfin, si le joueur est au sol, nous réinitialisons la valeur de "jumpLeft" à 2. Une fois au sol, le joueur peut à nouveau effectuer un double saut.

```
if (isGrounded)
{
    jumpLeft = maxJump;
}
```

Grâce à cette implémentassions, l'ange peut réaliser un double saut en appuyant sur la touche d'espace, ce qui lui donne une plus grande liberté de mouvement dans le jeu.

4.8 Installeur

Pour finaliser le processus, ma dernière tâche consiste à créer l'installeur du jeu. À cet effet, j'ai suivi une procédure que j'ai trouvée sur YouTube à l'adresse suivante : https://www.youtube.com/watch?v=7nxKAtxGSn8

Dans les paramètres de Unity, j'ai pu définir le logo du jeu, qui correspond au logo de notre groupe. Cette étape permet de personnaliser le jeu en lui donnant une identité

visuelle propre à notre équipe.

Ensuite, j'ai utilisé le logiciel Inno Setup Compiler pour compiler le jeu en un fichier exécutable (.exe). Une fois le fichier créé, je l'ai mis en ligne sur notre site internet afin de permettre son téléchargement par les utilisateurs intéressés. La procédure d'installation est détaillée dans le manuel d'installation, fournissant ainsi des instructions précises pour guider les utilisateurs tout au long du processus d'installation.

4.9 Divers

Finalement, la veille de la soutenance, j'ai consacré mon temps à résoudre les derniers problèmes rencontrés dans les différentes parties du jeu.

Dans le niveau 4, nous avons implémenté un mécanisme dans lequel la lave monte progressivement, obligeant ainsi le joueur à avancer rapidement dans le niveau. Si le joueur échoue à progresser suffisamment vite, la lave le rattrape et le joueur meurt. Pour réaliser cela, un script spécifique pour la lave, appelé "risingLava" est utilisé. Cependant, nous avons remarqué qu'en cas de mort du joueur, la lave continuait son avancée sans revenir à sa position de départ initiale. Afin de remédier à ce problème, nous avons décidé de ne pas inclure de checkpoints dans le niveau. De cette façon, si le joueur meurt, il suffit de recharger le niveau. Cela permet à la lave de revenir à sa position de départ initiale, et le joueur peut ainsi recommencer le niveau depuis le début.

```
if (PlayerHealth.instance.currentHealth <= 0)
{
    SceneManager.LoadScene("Level04");
}
```

Dans le dernier niveau, le boss est considérablement plus grand que les joueurs. Cependant, nous avons remarqué un problème où les joueurs pouvaient pousser le boss lorsqu'ils entraient en contact avec lui, ce qui donnait un aspect peu réaliste au jeu. Pour remédier à cela, j'ai ajusté la masse et le linear drag du boss. J'ai augmenté les deux valeurs à 100. Cette modification a eu pour effet d'empêcher les personnages de pousser le boss, renforçant ainsi l'immersion et la cohérence du jeu.

À la fin du jeu, une fois que les joueurs ont vaincu le boss, ils peuvent se diriger vers le drapeau de fin de partie, ce qui déclenche le déroulement des crédits du jeu. Cependant, nous avons remarqué un problème où l'ange peut sauter par-dessus le boss et atteindre le drapeau de fin de niveau sans avoir combattu le boss. Afin de résoudre ce problème, j'ai modifié le script du drapeau de fin de niveau. Désormais, les crédits ne seront déclenchés que si le niveau de vie du boss est inférieur ou égal à zéro. Ainsi, les joueurs sont contraints de vaincre le boss avant de pouvoir accéder à la fin du jeu. Cette modification garantit que les joueurs doivent remplir l'objectif principal du jeu en battant le boss pour pouvoir progresser et apprécier pleinement la fin du jeu.

Nous avons également effectué des tests en mode multijoueur pour tous les niveaux afin de nous assurer de leur faisabilité. L'objectif était de vérifier si les niveaux pouvaient être complétés avec succès par les joueurs ou d'apporter les modifications nécessaires pour les rendre réalisables.

Enfin, j'ai procédé à la création d'un nouvel installeur pour le jeu, intégrant la version complète et finalisée. De plus, j'ai mis à jour le site web en y ajoutant la dernière version du rapport de soutenance, ainsi que l'installeur du jeu disponible via le bouton "Télécharger". Les utilisateurs pourront ainsi accéder facilement à la dernière version du jeu via le site web.

4.10 Ressenti

En raison de mon manque d'expérience avec Unity avant la deuxième évaluation intermédiaire, j'ai pris du retard dans l'utilisation du logiciel. Cela peut donner l'impression que mes tâches étaient moins grandes et complexes que celles des autres membres du groupe. Cependant, malgré ces difficultés, j'ai tout de même rencontré des défis importants, notamment lors de la mise en place du double saut et des esquives des personnages. Ces problèmes ont nécessité beaucoup de temps et d'efforts pour les résoudre, et j'ai dû consacrer du temps supplémentaire à identifier leurs origines. Heureusement, les autres membres du groupe ont été très utiles, notamment pour m'aider avec le double saut. J'ai également dû résoudre divers problèmes plus petits mais chronophages, tels que la gestion des collisions avec les sols et les murs des niveaux.

Dans l'ensemble, j'ai apprécié ce projet. Nous avons connu des moments de stress et de tension, mais j'ai également énormément appris sur le développement d'un jeu vidéo. Je suis extrêmement heureux de pouvoir bientôt jouer à mon propre jeu et de le faire découvrir à mes amis.

Chapitre 5

Théo Gille

5.1 Multijoueurs

Tout au long du projet, j'ai été la personne chargée de l'aspect multijoueur du jeu, qui est un élément principal de notre projet. Il était donc essentiel d'avoir un multijoueur opérationnel pour assurer le bon fonctionnement de notre jeu.

5.1.1 Technologie utilisée

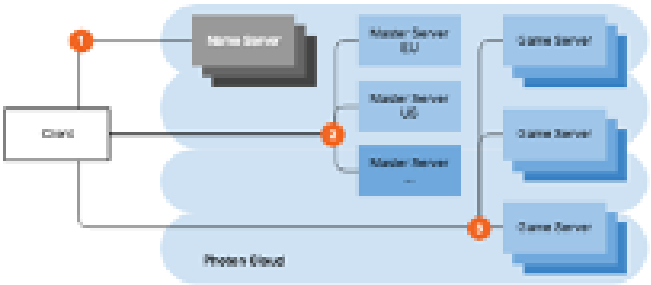
Il existe plusieurs technologies possibles pour implémenter le multijoueur. Personnellement, j'ai choisi d'utiliser Photon PUN 2, qui est gratuit, et la documentation associée à ce module est très complète. Ainsi, il est nécessaire de faire des recherches pour comprendre les fonctionnalités principales.

Asset utilisé :

https://assetstore.unity.com/packages/tools/network/pun-2-free-119922Fonctionnement

5.1.2 Fonctionnement

Le fonctionnement de Photon PUN 2 n'est pas très complexe. Tout d'abord, j'ai dû créer un serveur sur leur site en renseignant différentes options, telles que le nombre maximum de joueurs et les régions de jeu. Cette étape est cruciale, car à la fin de la configuration, un identifiant vous sera attribué, contenant toutes les informations du serveur.



https://doc.photonengine.com/pun/current/connection-and-authentication/regions
Pour notre projet, j'ai décidé d'utiliser uniquement la zone Europe des serveurs afin d'avoir une latence minimale par rapport à l'utilisation de serveurs aux États-Unis.

Après avoir configuré le serveur, il est maintenant nécessaire d'implémenter chaque méthode qui peut être utile pour le multijoueur. Depuis le début du projet, mon rôle a été de gérer la synchronisation de chaque mouvement, capacité et animation de chaque joueur. Le package Photon nous a été d'une grande aide grâce à ses différentes fonctionnalités, telles que :

PhotonView : Le composant PhotonView est utilisé pour synchroniser les objets du jeu entre les clients et le serveur. Il permet aux joueurs de voir les actions des autres joueurs dans le jeu.

PhotonTransformView : Le composant PhotonView est utilisé pour synchroniser les transformations des objets du jeu. Il permet aux joueurs de voir les mouvements des autres joueurs dans le jeu.

PhotonAnimatorView : Le composant PhotonView est utilisé pour synchroniser les animations des personnages dans le jeu. Il permet aux joueurs de voir les animations des autres joueurs dans le jeu.

PhotonNetwork : La classe utilisée pour gérer la connexion des clients au serveur de jeu est le PhotonNetwork. Elle permet aux joueurs de se connecter au serveur de jeu et de communiquer entre eux.

Ce sont autour de ces fonctionnalités que le multijoueur va fonctionner. Bien sûr, cela n'est qu'une partie, car il faut également faire appel à des procédures appelées RPC (Remote Procedure Call). Le fait d'appeler ces RPC avec Photon permet de déclencher et de partager des actions/informations avec l'ensemble des joueurs connectés.Exemple d'appel de RPC :

```
photonView.RPC("UpdateMovement", RpcTarget.Others, rb.velocity, animator.GetFloat
```

On remarque que cela est décomposé en plusieurs partie, tout d'abord le nom de la méthode/fonctions, les cibles de cet appel suivies des paramètres de la fonction.

5.1.3 Correctifs et Ajouts

Pour cette dernière soutenance, je me suis occupée de finaliser la synchronisation des animations et des capacités entre chaque client, ainsi que de revoir les nouveaux scripts pour y implémenter Photon et le multijoueur.

En effectuant ces différentes tâches, j'ai remarqué une sorte de latence constante sur l'un des deux joueurs. Dans notre cas, c'était le Joueur B qui observait une latence avec le Joueur A.

Pour corriger cela, j'ai d'abord terminé la synchronisation de toutes les animations. Cela a réduit cette sorte de latence, mais elle était toujours présente. Après avoir effectué diverses recherches, j'ai compris que c'était un problème récurrent, et même Photon lui-même propose une solution sur leur forum : https://doc.photonengine.com/pun/current/gameplay/lagcompensati

J'ai donc dû utiliser des techniques d'interpolation et de prédiction pour compenser les décalages du réseau.

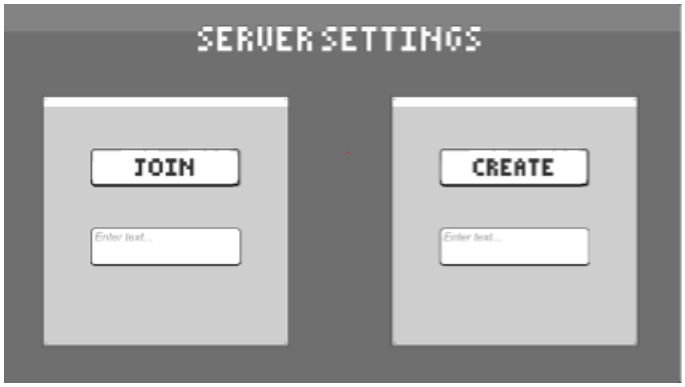
L'interpolation consiste à lisser le mouvement des autres joueurs en calculant des positions intermédiaires entre les mises à jour de position reçues du réseau. Au lieu d'afficher immédiatement la nouvelle position reçue, on utilise une fonction d'interpolation (comme la fonction Lerp de Unity) pour calculer une position intermédiaire en fonction du temps écoulé depuis la dernière mise à jour de position. Cela permet d'obtenir un mouvement plus fluide des autres joueurs sur l'écran du joueur local.

La prédiction consiste à estimer la position future des autres joueurs en se basant sur leurs dernières positions et leur vitesse. Cette technique permet de compenser la latence du réseau en anticipant le mouvement des autres joueurs.

5.2 UI & menu

Pour cette soutenance, mon objectif était d'améliorer l'esthétique des menus et de l'interface utilisateur (UI) du jeu, en mettant l'accent sur la gestion de l'UI et de l'inventaire (voir 3. Interaction et Action du joueur).

Le plus grand changement a été la mise en place d'un menu de paramètres du serveur permettant de rejoindre ou de créer une partie.



Le script pour rejoindre et créer une partie est resté le même, seule l'esthétique a été modifiée.

Dans un jeu, le menu principal est important, car il est l'un des premiers éléments avec lesquels nous interagissons. Ainsi, tout au long du projet, le menu principal et les UI en général ont évolué, que ce soit au niveau de l'esthétique ou des scripts.

En termes d'esthétique, il y a eu des changements pour rendre le menu principal plus attrayant tout en conservant sa forme générale.

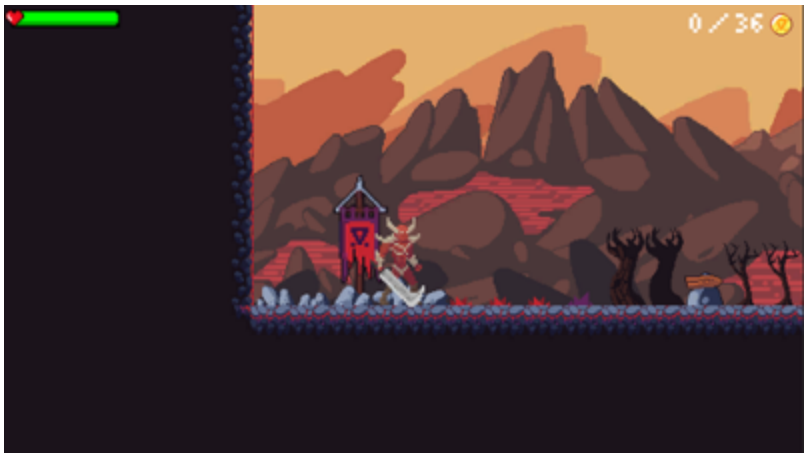
Première Version.

Version Actuelle.

Il reste globalement très similaire, avec deux changements notables : l'utilisation de boutons différents et l'adaptation des Canvas en fonction de la résolution choisie dans l'onglet Paramètres. Comme on peut le voir ici, cela est géré grâce à l'utilisation d'un menu déroulant.

Au niveau du script, des corrections ont été apportées, notamment en ce qui concerne la gestion de la résolution de l'écran, afin de permettre une bonne visualisation des menus en fonction de la résolution choisie.

C'est au niveau de l'UI que nous avons apporté le plus de changements dans notre jeu. Le joueur dispose maintenant d'une barre de vie, d'un compteur de pièces et de menus d'interaction tels que l'inventaire (voir 3. Interaction et Action du joueur). Lorsque le joueur rejoint le niveau 1, par exemple, il a accès à ces différentes informations :

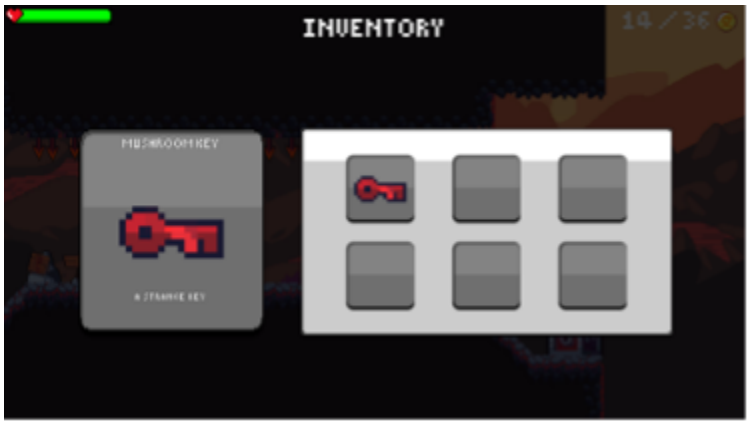


Ainsi, on remarque que le joueur doit ramasser un certain nombre de pièces pour pouvoir passer au niveau suivant, car chaque niveau exige un nombre spécifique de pièces.

Tout au long du niveau, il y aura des panneaux qui fourniront diverses informations ou indices :

De plus, il existe désormais un menu Pause accessible en appuyant sur la touche Échap, qui permet de retourner au menu principal ou de modifier les paramètres. Tout cela a évolué au fil des différentes soutenances.

Maintenant, il y a une nouvelle partie de l'interface utilisateur dédiée aux interactions avec le joueur, notamment une fenêtre d'inventaire où une description apparaît lorsque l'on survole un objet.



Il y a aussi une fenêtre pour Examiné un Objet :

Ainsi, chaque objet peut avoir une interaction spécifique (voir partie suivante). Il y a donc une diversité d'actions que le joueur peut choisir d'effectuer, mais bien sûr, seuls certains objets peuvent être examinés ou placés dans l'inventaire. Cela représente la ver-

sion finale des différentes interfaces utilisateur présentes dans le jeu.

5.3 Interaction et Action de Joueur

Pour cette soutenance finale, mon objectif était de finaliser toutes les interactions possibles entre le joueur et les objets, que ce soit pour ramasser un objet, le mettre dans l'inventaire ou l'examiner, ainsi que de corriger certains bugs liés aux échelles.

Le système d'inventaire n'était pas initialement prévu, mais au fur et à mesure, nous avons réalisé qu'un tel système serait utile. Généralement, dans un jeu de plateforme, il est rare d'avoir un système d'inventaire, c'est pourquoi nous avons décidé d'en créer un. Celui-ci permet aux joueurs de collecter des clés qui leur seront utiles pour ouvrir des coffres. À l'intérieur des coffres, il peut y avoir des pommes qui permettent de se soigner, ou des pièces qui sont nécessaires pour atteindre un certain nombre de pièces demandé. Pour cela, j'ai décidé de séparer les coffres et les clés en deux types. Les coffres et les clés rouges permettent aux joueurs de récupérer de la vie, tandis que les coffres et les clés blanches permettent de récupérer des pièces.

Une pomme permet de récupérer 20 points de vie, et entre chaque niveau, l'inventaire est réinitialisé pour éviter l'accumulation de pommes et rendre le jeu trop facile.

Avant de commencer la partie codage, il a fallu concevoir et réfléchir à un modèle d'inventaire simple d'utilisation tout en permettant de nombreuses actions. Plusieurs actions seront possibles, comme le fait de porter un objet et de pouvoir le jeter :

Le fait d'examiner un objet (voir partie précédente) ou encore de pouvoir le ramasser pour le mettre dans l'inventaire est également possible. Chaque objet aura donc un type bien distinct et chaque action sera gérée en fonction du type de l'objet. Cela est géré dans le script Item.cs par un simple switch case :

```
switch (interactType)
    {
        case InteractionType.PickUp:
            //Add the object to the PickedUpItems list

            FindObjectOfType<InventorySystem>().PickUp(gameObject);
            //Disable
            gameObject.SetActive(false);
            break;
        case InteractionType.Examine:
            //Call the Examine item in the interaction system
            FindObjectOfType<InteractionSystem>().ExamineItem(this);
            break;
        case InteractionType.GrabDrop:
            //Grab interaction
            FindObjectOfType<InteractionSystem>().GrabDrop();
            break;
    }

    //Invoke (call) the custom event(s)
    customEvent.Invoke()
```

L'ensemble de ces interactions vont être géré par plusieurs scripts tels que :

Item.cs : permet la création d'objets interactifs et la personnalisation des événements auquel ils sont associés si on les utilise ainsi que leur type (PickUp, Examine ou GrabDrop)

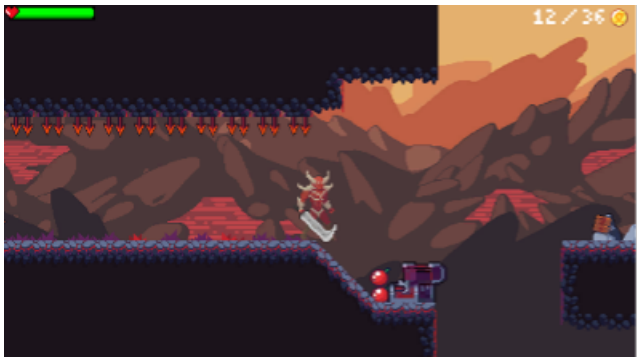
InventorySystem : comme son nom l'indique permet de gérer les différentes fonctionnalités de l'inventaire.

InteractionSystem : permet de gérer les interactions entre les objets.

KeyAndChestInteraction : permet de gérer les interactions entre la clé et le coffre.

Il y a donc eu beaucoup de travail sur cette partie, que ce soit au niveau des interfaces utilisateur pour trouver des idées, de la synchronisation entre les joueurs ou de la gestion des animations.

Lorsqu'un coffre est ouvert, il reste ouvert et aucun butin n'est plus disponible à l'intérieur, ce qui permet aux joueurs de savoir quels coffres ont déjà été ouverts.



5.4 Ressentis & Difficultés :

Durant ces 5 mois de développement, j'ai pris énormément de plaisir à accomplir toutes les tâches qui m'ont été assignées, notamment dans les interactions avec les joueurs. Cela m'a permis de progresser dans ma compréhension de certains aspects de C et du multijoueur, qui étaient des domaines auxquels je n'avais pas vraiment porté une grande attention avant ce projet. J'ai ainsi pu comprendre les requêtes et le fonctionnement général d'un serveur multijoueur.

Ce projet représente pour moi une forme d'accomplissement personnel, car j'ai réussi à relever tous les défis qui se sont présentés, même si au début j'avais des doutes quant à ma capacité à y parvenir. Tout au long du projet, notre groupe a fait preuve d'un excellent travail d'équipe, avec une entraide constante, et chaque membre a contribué à l'avancement du projet.

Malgré les examens et les différentes évaluations, nous avons réussi à maintenir un rythme de travail constant, ce qui nous a permis de terminer le projet à temps en atteignant tous nos objectifs.

Jusqu'à présent, je n'avais jamais travaillé sur un projet de ce type en groupe, et cela m'a fait réaliser que réaliser un tel projet seul peut devenir très complexe, et donc que le travail en groupe est essentiel dans le développement en général. Bien sûr, chaque membre a dû surmonter ses propres difficultés. Pour ma part, j'ai rencontré de nombreux problèmes, notamment en ce qui concerne la synchronisation des animations dans le multijoueur, ainsi que la gestion des interactions avec le joueur, en particulier les échelles. Avant de commencer cette mécanique, je pensais qu'elle serait relativement simple, mais j'ai rencontré de grandes difficultés pour gérer le comportement prévisible de notre personnage, notamment en veillant à ce qu'il ne puisse plus se déplacer horizontalement lorsqu'il monte sur une échelle à partir d'une certaine hauteur.

En conclusion, j'ai adoré collaborer avec Noé, Leandro et Clément. Ce sont des collaborateurs très compétents et nous avons eu une excellente entente dès le début, ce qui a

favorisé le bon fonctionnement du groupe.