



PROJET GALAXY TRAVELLER

by Cygnus

Rapport de projet

Juin 2024

BELPERIN Noé, BLANC Johan,
DEFONTAINE JOLIVET Emilien, THOME Aubin

Table des matières

I	Introduction	6
II	Genèse du projet	7
1	Origine	7
2	Présentation de l'entreprise	7
3	Présentation de l'équipe	8
3.1	BELPERIN Noé	8
3.2	BLANC Johan	8
3.3	DEFONTAINE JOLIVET Emilien	8
3.4	THOMÉ Aubin	8
III	Présentation du projet	10
4	Rappel du concept du jeu	10
5	Répartition des tâches	11
6	Réorganisation après le départ de Thomas	11
7	Etat de l'art	12
7.1	Les autres jeux de type plateforme	12
7.2	Leurs fonctionnalités propres	13
8	Planning avancement des tâches	14
9	Charte graphique	14
9.1	Site Web	14
9.1.1	Accueil	14
9.1.2	Présentation du projet	15
9.1.3	Présentation de l'équipe	15
9.1.4	Ressources	15
9.2	Logo	15
9.2.1	Typographie	15
9.2.2	Couleurs	15
9.2.3	L'Épée	16

9.2.4	Composition et hiérarchie visuelle	16
IV	Produit final	17
10	Déroulement d'une partie	17
11	Aspect technique	17
11.1	Joueur	17
11.1.1	Transform, Rigidbody, CapsuleCollider	17
11.1.2	Animator	18
11.1.3	Player (script)	20
11.1.4	PlayerManager (script)	22
11.1.5	CWork (script)	22
11.1.6	BasicBehaviour, MoveBehaviour et FlyBehaviour (script)	22
11.1.7	PickupBehavior (script)	22
11.1.8	Inventory (script)	23
11.1.9	Graphismes (mesh)	23
11.2	Items	24
11.3	Multijoueur	26
11.4	Intelligence artificielle	28
11.4.1	L'intelligence artificielle distance	29
11.4.2	L'intelligence artificielle mêlée	30
11.4.3	L'intelligence artificielle explosion	30
11.4.4	L'intelligence artificielle mêlée/distance (le boss final)	31
11.4.5	Les attaques	31
11.4.6	L'Animator des intelligences artificielles	33
11.4.7	L'Animator de l'intelligence artificielle explosion	35
11.5	Parcours	35
11.5.1	Plate-formes	36
11.5.2	Terrain de spawn	37
11.5.3	Surface de respawn	38
11.6	Interface utilisateur	39
11.6.1	Menus	39
11.6.2	En jeu	40
11.7	Sons	40
11.8	Installation	42
11.9	Site web	43

V Bilan	46
12 Ressenti global	46
13 Ressenti individuel	46
13.1 BELPERIN Noé	46
13.2 BLANC Johan	47
13.3 DEFONTAINE JOLIVET Emilien	47
13.4 THOMÉ Aubin	47
VI Conclusion	48

Première partie

Introduction

Ces pages constituent le rapport de la dernière soutenance technique du groupe Cygnus pour leur projet Galaxy Traveller.

Dans ce dernier rapport, nous allons résumer l'ensemble du travail accompli sur notre jeu. La plupart des spécificités propres à celui-ci seront expliquées. Des origines du jeu jusqu'aux explications techniques des animations, nous aborderons un maximum de sujets de manière à couvrir au mieux l'ensemble du travail accompli. Cependant certaines parties seront moins détaillées que d'autres car moins importantes ou répétitives.

Nous tenons à préciser qu'il se peut que certains points divergent légèrement avec ce que nous avions pu expliciter dans le cahier des charges ou bien dans le premier rapport. Ces écarts mineurs sont la conséquence de notre volonté de faire au mieux et donc de ré-adAPTER légèrement certaines parties pour proposer un produit réellement divertissant.

Nous commencerons ce rapport en parlant des origines de ce projet, puis nous ferons une présentation générale avant d'entrer dans la partie technique de notre jeu.

Deuxième partie

Genèse du projet

1 Origine

Ce travail a comme origine le projet de première année de l'*EPITA* qui consiste à développer un jeu vidéo en 3D, jouable sur la plateforme *Windows* et en mode coopération (donc en multijoueurs), le tout développé en C#.

Dès l'annonce du projet, nous avons immédiatement constitué notre équipe et s'est rapidement mis en place une excellente synergie de groupe. Très vite, nous nous sommes tournés vers le jeu de plateforme que chacun affectionne et qui se présente comme un intemporel dans le monde du jeu vidéo. Nous avions tous joué à ce type de jeu, en solo ou en multijoueurs et, créer un nouveau monde et de nouveaux mécanismes pour un jeu de plateforme nous a tout de suite motivés et fédérés autour de ce projet.

Après de nombreuses délibérations, nous avons décidé de concevoir un jeu vidéo avec comme imaginaire l'espace et au vu des caractéristiques de celui-ci, nous lui avons rapidement trouvé un nom : “*Galaxy Traveller*”!

2 Présentation de l'entreprise

Cygnus est une entreprise dynamique et en croissance continue dans le domaine du développement de jeux vidéo. Notre siège social, situé dans le quartier de Confluence au centre de Lyon, est le lieu où innovation et créativité se rencontrent et se complètent pour donner vie à des mondes virtuels fascinants. Animés par notre passion pour les jeux vidéo, la science et par notre quête de l'excellence, nous nous engageons à créer les expériences de jeu les plus imaginatives et stimulantes possibles. Notre équipe est composée d'un groupe diversifié et compétent, de développeurs expérimentés, de concepteurs de jeux créatifs, de graphistes et d'ingénieurs en devenir. Ensemble, nous formons une équipe de rêveurs, d'explorateurs et d'innovateurs qui partagent un objectif commun : offrir des moments de divertissement inoubliables aux joueurs du monde entier. Notre succès réside dans notre capacité à combiner des univers à part, les dernières avancées technologiques et une approche centrée sur l'utilisateur pour créer des jeux qui non seulement divertissent mais aussi, surprennent. Pour résumer, nous cherchons constamment à nous surpasser pour garantir LA meilleure expérience de jeu possible.

Chez Cygnus, nous pensons que les jeux vidéo sont un moyen puissant de libérer et de développer votre créativité. Nous nous engageons à faire de cette vision ambitieuse, une réalité.

3 Présentation de l'équipe

Notre équipe est nouvellement composée de quatre membres à la suite d'un départ.

3.1 Belperin Noé

Co-fondateur de l'entreprise Cygnus, j'ai comme missions principales la gestion de l'intelligence artificielle qui se caractérise par les combats de boss, j'ai de plus comme responsabilités la partie graphisme ainsi que le son et la musique de notre jeu. J'assiste Johan Blanc dans la réalisation de l'interface utilisateur et également dans la mise en place des armes et collectables.

3.2 Blanc Johan

Dans le cadre de notre projet, j'ai été nommé chef de groupe. J'essaie de motiver tout le groupe pour mener à bien la réalisation du projet dans les temps impartis. Je suis, de plus, responsable de la partie "Armes et Collectables" ainsi que de la partie "Interface Utilisateur". Pour terminer, je suis suppléant de Noé pour l'implémentation de l'intelligence artificielle ainsi que des sons et musiques.

3.3 Defontaine Jolivet Emilien

Je fais partie du comité d'administration de Cygnus. Je suis le responsable du site internet, et tout ce qu'il induit, ainsi que toute la partie réseau du jeu. J'ai également contribué au développement du parcours, notamment des plateformes et des scripts qui leurs sont propres, c'est-à-dire, les déplacements de celles-ci ou encore leurs propriétés. De plus, tout comme mes collègues, je contribue à la création des niveaux du jeu.

3.4 Thomé Aubin

Je suis actuellement responsable de la programmation fonctionnelle au sein de notre projet. Mon rôle est d'harmoniser l'ensemble des scripts de projets et m'assurer que ceux-ci soient capables de communiquer de manière efficace. Je me suis, dans un premier temps, penché sur l'utilisation de l'héritage et de la composition au sein

de notre projet. En parallèle, je suis aussi responsable de la création des différents parcours de chaque niveau.

Troisième partie

Présentation du projet

4 Rappel du concept du jeu

Aujourd’hui, les jeux sur mobile, PC ou encore console sont multiples et continuent de se réinventer. Cependant, chacun possède son style, sa mécanique et ses caractéristiques. Bien qu’étant parmi les premiers jeux ayant vus le jour, le *jeu de plateforme* est toujours populaire, continue de perdurer et à se décliner en différentes versions, qui chacune, apporte son lot de nouvelles fonctionnalités.

Dans cette dynamique de toujours remettre au goût du jour le jeu de plateforme, nous avons imaginé notre jeu, un *plateforme 3D de coopération*. A chaque niveau, les joueurs doivent survivre à un parcours d’obstacles et récupérer différentes capacités (armes, armures, potions diverses) avant d’atteindre une planète et d’en affronter le boss pour pouvoir ensuite accéder au parcours et au boss suivant.

Dans ce jeu, votre objectif est de passer d’une planète à l’autre en surmontant une série d’épreuves de plus en plus complexes sur votre chemin vers le boss final. Les planètes et leurs atmosphères forment des mondes distincts. Chaque monde est différent des autres et il faut affronter et éliminer un boss contrôlé par une intelligence artificielle pour quitter chaque planète en sachant que la difficulté est graduelle au fil de l’avancée dans le jeu. Pour passer à la planète suivante, vous devez emprunter un chemin de plateformes complexe, parsemé d’obstacles et de dangers. Vous devez naviguer sur des plateformes amovibles, faire très attention aux plateformes qui disparaissent sous vos pieds, et même faire face à de mystérieux maléfices, tels que des visions altérées et des hallucinations. Ces éléments perturbateurs ont pour but de déstabiliser le joueur, le forçant à s’adapter et à développer de nouvelles stratégies pour progresser.

"Galaxy Traveller" offre une expérience de jeu unique, où chaque planète est une aventure en soi, et où la persévérance et la maîtrise des mécanismes de jeu sont essentielles pour explorer l’univers. Vous vous lancez dans un voyage spatial palpitant, où seuls les plus habiles et stratégiques pourront affronter les spécificités de chaque planète et triompher des boss qui se dressent sur leur chemin.

Notre jeu intègre un mode multijoueurs qui demande à nos joueurs de faire preuve de stratégie tout au long des niveaux pour se compléter et arriver à vaincre le boss

de fin. Ainsi des stratégies pourraient être mises en place entre les joueurs, l'un se concentrant par exemple sur la régénération et l'autre sur l'attaque.

5 Répartition des tâches

L'un de nos membres a quitté le projet, c'est pourquoi nous avons établi une nouvelle répartition des tâches de manière à ce que chacun ait la même quantité de travail. Nous en avons profité pour modifier les anciennes responsabilités après s'être rendu compte du travail à fournir dans certains domaines, nous avons donc décidé de fonctionner par paire de deux étudiants en fonction des aptitudes de travail de chacun, les tâches sont donc divisées en deux groupes ce qui nous permet de progresser plus rapidement. De plus, lors de l'avancement du projet nous avons fait émerger de nouvelles responsabilités que nous n'avions pas imaginées lors du cahier des charges. Nous avons opté pour une répartition des tâches en fonction des préférences et aptitudes de chaque étudiant du groupe.

Responsabilité	Responsable	Suppléant
Programmation fonctionnelle	Aubin	Emilien
Graphisme	Noé	Aubin
IA	Noé	Johan
Git	Emilien	Aubin
Site Web	Emilien	Aubin
Réseau et multijoueur	Emilien	Johan
Administratif	Johan	Noé
Rendus	Aubin	Emilien
Parcours	Aubin	Emilien
Armes et collectables	Johan	Noé
Sons et musiques	Noé	Johan
UI	Johan	Noé

TABLE 1 – Nouvelle répartition des tâches

6 Réorganisation après le départ de Thomas

L'un de nos membres ayant quitté le projet, nous avons dû revoir et réorganiser la répartition des tâches afin de garantir une charge de travail équilibrée pour chacun des membres restants. Cette réorganisation nous a également permis de reconsidérer les responsabilités initialement assignées et de les ajuster en fonction du volume de

travail réel requis dans certains domaines spécifiques.

Pour maximiser notre efficacité, nous avons adopté une approche de travail en binôme, regroupant les étudiants par paires selon leurs compétences et préférences individuelles. Cette méthode de travail collaboratif a permis de diviser les tâches en deux grands groupes, favorisant ainsi une progression plus rapide et plus coordonnée du projet. Les binômes ont pu se concentrer sur des aspects précis du développement, ce qui a amélioré la qualité du travail fourni et a permis une meilleure gestion du temps.

Au fur et à mesure que le projet avançait, nous avons identifié de nouvelles responsabilités et de nouveaux défis qui n'avaient pas été prévus lors de l'élaboration initiale du cahier des charges. Par exemple, des aspects tels que l'optimisation des performances, la gestion des retours utilisateurs, et l'implémentation de nouvelles fonctionnalités se sont révélés essentiels pour le succès du projet. Ces découvertes nous ont poussés à adapter notre répartition des tâches de manière flexible et dynamique, en attribuant ces nouvelles responsabilités aux membres les mieux équipés pour les gérer.

Nous avons pris soin de prendre en compte les préférences et les aptitudes de chaque étudiant dans cette nouvelle répartition, ce qui a permis de maintenir un haut niveau de motivation et d'engagement au sein de l'équipe. Chaque membre a pu se concentrer sur des tâches qui correspondaient à ses compétences et à ses intérêts, ce qui a non seulement amélioré l'efficacité de notre travail, mais a également favorisé une atmosphère de travail plus harmonieuse et collaborative.

En conclusion, cette réorganisation, bien que motivée par une contrainte initiale, s'est avérée bénéfique pour l'avancement du projet. Elle nous a permis de mieux exploiter les talents de chaque membre, de répondre plus efficacement aux défis imprévus, et de maintenir une dynamique de groupe positive et productive. Grâce à cette approche, nous avons pu assurer une progression continue et harmonieuse du projet, en nous adaptant aux évolutions et aux besoins émergents tout en maintenant un niveau de qualité élevé dans notre travail.

7 Etat de l'art

7.1 Les autres jeux de type plateforme

Le jeu de plateforme est apparu dans les années 1980 et nous a donc accordé une large palette d'inspirations. Les plus grands classiques du jeu vidéo se retrouvent dans

ce genre, nous pouvons ainsi citer le tout premier *Super Mario Bros*, et par conséquent toute la série qui a suivi. Ici nous nous sommes particulièrement intéressés à *Super Mario Odyssey* (Figure 1) qui par sa 3D et son gameplay se rapprochait le plus de ce que nous voulions faire. Dans cette catégorie du jeu de plateforme, nous retrouvons aussi la série *Prince of Persia*, les jeux *Donkey Kong*, *Sonic* ou plus récemment *Hollow Knight* (Figure 2). Un des autres jeux de plateforme qui a dernièrement séduit de par sa simplicité et son mode de jeu “die & retry” est *Only Up* (Figure 3) qui nous a inspirés surtout pour son aspect et le passage de plateforme en plateforme.



FIGURE 1 – Super Mario Odyssey



FIGURE 2 – Hollow Knigh



FIGURE 3 – Only Up !

7.2 Leurs fonctionnalités propres

Le principe de la quasi-totalité des jeux de plateforme est le contrôle d'un personnage à la troisième personne qui doit sauter de surface en surface sans tomber sous peine de perdre des points de vie, voire même, d'obtenir un game over. En plus de cela, s'ajoutent souvent des ennemis à combattre que ce soit sous la forme de petits ennemis au cours du jeu ou même de boss à certains points clés.

La variété des jeux du type plateforme apporte un grand nombre d'autres fonctionnalités telles que la collecte d'objets pour marquer des points, débloquer des aptitudes, armes et niveaux mais aussi des énigmes ou une trame principale qui racontent une véritable histoire. Certains jeux se dotent même d'un mode multijoueurs. Dans cette

catégorie, l'un des plus populaire est certainement *Super Mario 3D World* qui peut réunir jusqu'à 4 joueurs.

8 Planning avancement des tâches

Nous n'avons accusé aucun retard et avons totalement implémenté l'intégralité des fonctionnalités prévues.

Tâche	Soutenance 1	Soutenance 2	Soutenance 3
Graphisme	100%	100%	100%
IA	100%	100%	100%
Site Web	100%	100%	100%
Réseau et multijoueur	100%	100%	100%
Parcours	100%	100%	100%
Armes et collectables	100%	100%	100%
Sons et musiques	100%	100%	100%
UI	100%	100%	100%

TABLE 2 – Avancement des tâches

9 Charte graphique

9.1 Site Web

Pour le site nous avons respecté le cahier des charges fourni au début du projet. Nous devions créer une page d'accueil qui permettait l'accès à trois parties :

- Une présentation du projet
- Une présentation de l'équipe
- Une partie ressources qui contient les différents téléchargements et liens du projet

Pour faciliter l'accès aux différents éléments nous avons décidé de disposer ces trois parties sur une seule page.

9.1.1 Accueil

Cette première partie d'accueil permet de définir les principaux éléments du jeu et d'accéder facilement à l'ensemble des autres parties. La couleur de cette première partie d'accueil est basée sur des tons foncés qui représente bien notre premier niveau sombre. Le bouton "Télécharger" en blanc est animé de manière à attirer l'attention et pousser le futur joueur à télécharger le jeu.

9.1.2 Présentation du projet

Cette partie permet d'introduire le jeu et d'en donner les points clés, nous avons donc choisi une couleur dynamique en dégradé, du bleu vers le noir.

9.1.3 Présentation de l'équipe

Cette partie permet à l'utilisateur d'en découvrir plus sur l'entreprise ainsi que les principales tâches assignées à chaque membre.

9.1.4 Ressources

Cette dernière partie correspond à toutes les bibliothèques et librairies qui ont permis la réalisation de ce jeu.

9.2 Logo



FIGURE 4 – Logo Galaxy Traveller

9.2.1 Typographie

Le logo de notre jeu fait ressortir le type d'univers dans lequel le joueur va se plonger. Le mot "Galaxy" est écrit en gris clair avec une typographie moderne et angulaire. Les lettres sont majuscules et présentent des coupures nettes, ce qui donne un aspect futuriste et technologique. Le choix de cette couleur et du style de police évoque l'espace et la science-fiction. Le mot "Traveller" est écrit dans une teinte de gris plus foncée ce qui le rend bien distinct. Les lettres sont également majuscules et angulaires, renforçant la cohérence visuelle avec "Galaxy".

9.2.2 Couleurs

Les nuances de gris utilisées sont sobres et élégantes, renforçant le thème spatial et futuriste du jeu. Le contraste entre le gris clair et le gris foncé aide à différencier les différents éléments du logo tout en maintenant une palette de couleurs unifiée.

9.2.3 L'Épée

Le "T" de "Traveller" est stylisé de manière à ressembler à une épée. Cette épée ajoute une dimension d'aventure et de combat. L'épée est ombrée avec un dégradé de gris, donnant une sensation de profondeur et apportant donc un peu plus de relief à notre logo.

9.2.4 Composition et hiérarchie visuelle

Le positionnement de l'épée "T" au centre crée un point focal qui attire immédiatement l'attention. L'épée traverse le "G" de "Galaxy" et la première lettre de "Traveller", créant une connexion visuelle entre les deux mots. La différence de couleur et de taille entre "Galaxy" et "Traveller" crée une hiérarchie visuelle claire, avec "Traveller" se détachant plus fortement en raison de son ombrage et de sa couleur plus foncée.

Quatrième partie

Produit final

10 Déroulement d'une partie

Lors d'une partie en multijoueur les deux players commencent au même point, avec la même vie, le même oxygène et un inventaire vide. Quand la partie commence, les deux joueurs arrivent au premier niveau. Ils peuvent alors explorer la première plateforme et chercher à remplir leur inventaire avec les premières potions. Ils doivent ensuite trouver le chemin vers le début réel du parcours, à l'intérieur du volcan. Très vite, les joueurs vont rencontrer leurs premiers ennemis, des petits robots qu'ils n'ont d'autre choix que de fuir. C'est seulement à la suite de cette rencontre qu'ils pourront découvrir leur première épée. Après de nombreux obstacles, ils arriveront au boss du niveau qu'ils devront tuer pour passer au suivant.

11 Aspect technique

11.1 Joueur

Notre joueur prend l'apparence d'un robot. Il est référencé sous forme d'un gameObject stocké dans une prefab appelée FinalPlayer qui est instantiée au moment de lancer une partie. Cette prefab est composée de l'apparence du robot, des items que le robot peut prendre dans ses mains, d'une EquipmentLibrary et d'un collider qui se place à l'avant du joueur. Nous allons détailler ci-dessous l'ensemble des 16 composants placés sur notre player :

11.1.1 Transform, Rigidbody, CapsuleCollider

Ces trois composants permettent respectivement de stocker la position, d'appliquer une physique à notre player et de gérer les collisions physiques. Ces trois points étant la base du système des gameObjects sur Unity sur lesquels nous n'avons fait aucune modification, nous ne détaillerons pas plus.

11.1.2 Animator

L'Animator possède des paramètres (float, int, bool, trigger) qui en fonction de leur état vont activer les animations. Il possède aussi des layers qui vont permettre une meilleure lecture des graphiques d'animations et une superposition des animations. La majorité des animations ont été trouvées et importées avec le Unity Asset Store ou à partir du site Mixamo. Certaines plus spécifiques ont été réalisées par nos soins, c'est par exemple le cas pour l'équipement des potions. Nous avons dû nous assurer que chaque transition était fluide et que les animations répondaient correctement aux actions du joueur. Nous allons ici détailler chaque layer, sa fonction et les paramètres auxquels il fait appel :

LAYER 1 : Moving

Comme son nom l'indique, ce layer gère les animations de déplacement du joueur. Le joueur peut se déplacer tout droit, à droite, à gauche et repartir dans l'autre sens. Il peut aussi sauter pour passer d'obstacle en obstacle et rouler pour esquiver. Toutes ces actions sont performées par des animations, déclenchées par 6 paramètres :

- **H et V (float)** : Permettent de déplacer le joueur sur l'axe horizontal et vertical en fonction de leur valeur comprise entre -1 et 1.
- **Speed (float)** : Permet de déterminer quelle animation de mouvement jouer en fonction de la vitesse du joueur, elle est comprise en 0 et 2. Le joueur possède ainsi 4 animations pour ses déplacements basiques :
 - walk : entre 0 et 0.6
 - jog : entre 0.6 et 0.8
 - run : entre 0.8 et 2
 - sprint : à 2
- **Grounded (bool)** : Permet de déterminer si le joueur est en contact avec le sol et l'empêche ainsi de déclencher certaines animations en l'air. Il sert aussi à déclencher l'animation de chute si le player tombe d'une plateforme ou saute.
- **Jump (bool)** : Permet de déclencher un saut en avant.
- **Roll (trigger)** : Permet de déclencher une roulade avant.

LAYER 2 : Flying

Ce layer nous a permis de tester plus facilement le jeu en passant certains obstacles de nos niveaux. Il possède un seul paramètre :

- **Fly (bool)** : déclenche l'animation de vol.

LAYER 3 : Gathering

Ce layer gère l'animation de la collecte des items à l'aide d'un seul paramètre :

- **Pickup (trigger)** : déclenche l'animation de collecte de l'item.

LAYER 4 : HoldingWeapon

Ce layer est équipé d'un masque permettant d'agir seulement sur les bras du personnage. Cette animation est activée à l'aide d'un seul paramètre :

- **HoldWeapon (bool)** : verrouille les bras dans une position donnée pour porter le fusil.

LAYER 5 : HoldingItem

Ce layer est équipé d'un masque permettant d'agir seulement sur les mains du joueur. Il possède deux paramètres :

- **HoldPotion (bool)** : verrouille la main gauche dans une position donnée pour porter une potion.
- **HoldSword (bool)** : verrouille la main droite dans une position donnée pour porter une épée.

LAYER 6 : Action

Ce layer permet de performer différentes action à l'aide de 4 paramètres :

- **Drink (trigger)** : déclenche l'animation de consommation d'une potion.
- **Attack (trigger)** : déclenche une animation d'attaque à l'épée. Cette animation implémente un animation event qui appelle une fonction à un moment précis de l'animation : au moment de remettre l'état de l'action du player à false.
- **AttackDistance (trigger)** : déclenche une animation de tir. Cette animation implémente deux animation event qui appellent deux fonctions à des moments précis de l'animation : au moment de la disparition de l'item et pour remettre l'état de l'action du player à false. Ces deux méthodes sont définies dans le script Player.
- **KnockBack (bool)** : déclenche une animation de recul. Durant le temps de cette animation, le Player ne peut rien faire et recule. L'animation implémente un animation event qui appelle une fonction à la fin de l'animation : à la fin du recul, le joueur peut à nouveau se déplacer ou attaquer. Cette fonction est définie dans le script Player.

LAYER 7 : Dying

Ce dernier layer gère les deux animations de mort possible pour notre joueur. Il utilise donc deux paramètres :

- **DeadHp (trigger)** : passe à true au moment de la mort provisoire du joueur.
- **DeadO2 (trigger)** : passe à true au moment de la mort définitive du joueur.

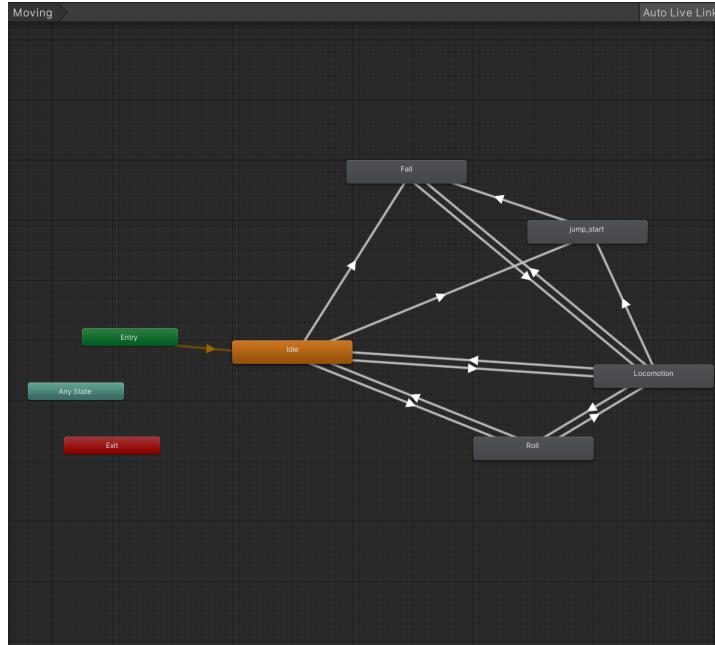


FIGURE 5 – Animateur principal du joueur

11.1.3 Player (script)

Ce script pose les bases du joueur (vie et oxygène), des actions, de la gestion des animations et des visuels des items équipés. La méthode `Update()`, appelée à chaque frame, appelle donc 4 managers :

- **OxygenManager()**

Il gère l’oxygène du joueur en le faisant décroître au fil du temps. S’il est inférieur à 10% le joueur perd en vitesse et s’ il passe à 0, c’est le game over avec l’animation adaptée.

- **HealthManager()**

Le joueur part avec 100 de vie et en perd s'il prend des dégâts. Il reprend cependant de la vie au fil du temps mais plus ou moins en fonction de sa valeur :

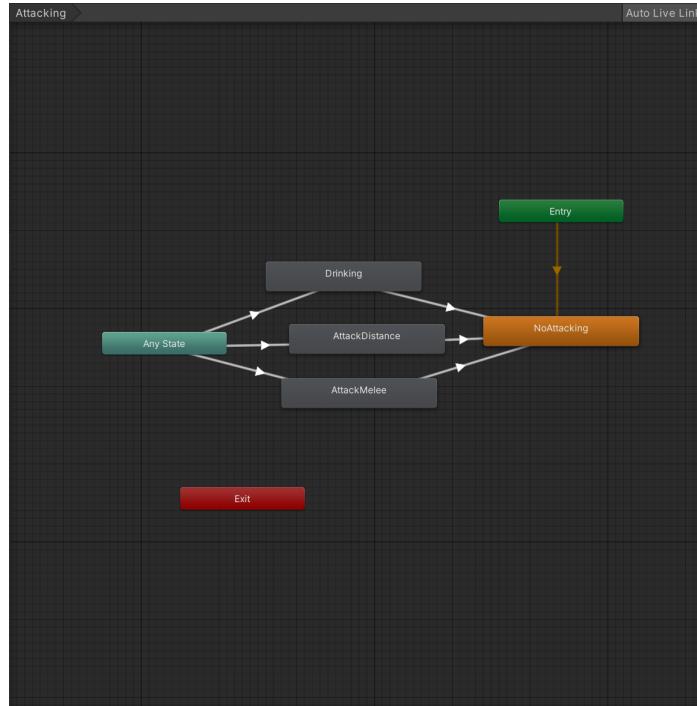


FIGURE 6 – Animateur du joueur pour les attaques

entre 0 et 25 il reprend rapidement de la vie, entre 25 et 50, il en reprend deux fois moins vite et entre 50 et 100, il en reprend 10 fois moins vite. Ce manager s’occupe aussi de faire évoluer l’écran de vie qui devient de plus en plus rouge en fonction des points de vie.

- **ActionManager()**

Cette fonction lance la plupart des actions du joueur, les animations associées ainsi que certains effets si l'action est une potion.

- **HoldingVisualManager()**

Cette fonction active les animations des items et armes que porte le joueur et ce en lien avec l'inventaire.

Une autre méthode importante :

- **TakeDamage()**

Retire les dégâts au joueur et change de couleur la barre de vie à chaque dégât. Cette fonction s’occupe aussi de faire respawn le joueur en cas de mort si les HP tombent à 0.

Ce script gère aussi les attaques du joueur :

- **L'attaque mélée :**

Le joueur ne possède pas de script pour attaquer en mélée : c'est son épée qui en possède un. Elle utilise la fonction de Unity OnTriggerEnter pour détecter les colliders touchés par l'épée. Si le collider touché appartient à un ennemi et que le joueur n'est pas en train de faire autre chose (prendre du recul ou ramasser un objet), alors l'IA prends des dégâts et subit un léger recul.

- **L'attaque distance :** L'attaque distance est gérée dans le script Player.cs grâce à la fonction SendAttackDistance. Cette fonction, lorsque la touche de feu est appuyée et que le player est en train de viser, instancie un projectile en forme de douille pour le tirer en face du player. Ces projectiles sont les mêmes que ceux tirés par les IA et seront expliqués dans la partie IA.

11.1.4 PlayerManager (script)

Ce script associe l'interface utilisateur (UI) du player et définit les paramètres utiles à photon.

11.1.5 CWork (script)

Ce script gère les mouvements de la caméra en fonction des mouvements de la souris. Pour éviter à la caméra de rentrer dans les objets, nous avons utilisé trois Raycast qui permettent de déterminer la distance maximale à laquelle reculer la caméra pour éviter toute collision. La distance s'adapte aussi si le joueur vise avec une arme. Les angles de la caméra ne sont pas limités en mode build pour permettre au joueur de fly pour débugger plus facilement les niveaux.

11.1.6 BasicBehaviour, MoveBehaviour et FlyBehaviour (script)

Ces trois scripts sont tirés d'un asset du Unity Asset Store et ont été modifiés pour s'adapter au mieux à ce que nous voulions pour les déplacements de notre joueur. Nous avons par exemple ajouté le jump et le roll. Concernant le fly, il n'est accessible qu'en mode développement et désactivé dans les builds classiques.

11.1.7 PickupBehavior (script)

Ce script permet de collecter les items et les armes, détectés à l'aide d'un collider fantôme. Il gère aussi le déclenchement de la collecte d'un item et l'ajoute à l'inventaire.

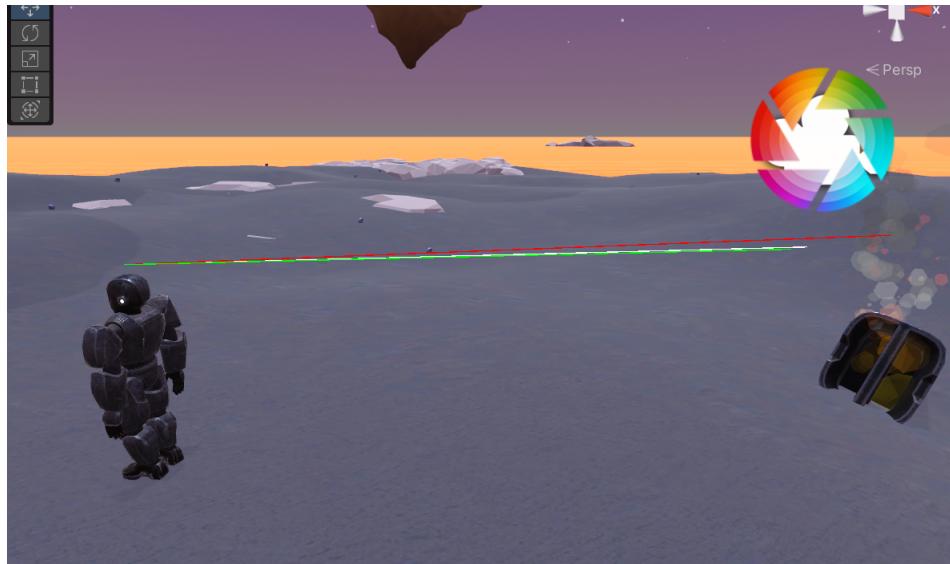


FIGURE 7 – Raycasts de la caméra

11.1.8 Inventory (script)

Ce script est composé principalement d'une liste qui répertorie les items et armes collectés, il s'occupe aussi d'activer le visuel que le joueur a sélectionné dans son inventaire.



FIGURE 8 – Player équipé de son fusil

11.1.9 Graphismes (mesh)

Nous avons développé les graphismes du personnage principal, un robot, que le joueur incarne. Ces graphismes sont conçus pour être à la fois fonctionnels et esthétiques, permettant une immersion totale dans le monde du jeu.



FIGURE 9 – Prefab du Player

11.2 Items

La collecte d'armes et d'autres objets est cruciale pour la progression dans le jeu. En effet, ce sont ces items qui permettent aux joueurs de surmonter les défis et de terminer les niveaux avec succès. Actuellement, nous avons implémenté plusieurs épées, un fusil et trois potions distinctes. Tous ces items sont référencés en tant que Scriptable Objects dans notre projet. Cette approche nous a permis de centraliser les données des items et d'éviter la création de nombreux scripts redondants ayant beaucoup de points communs. En utilisant les Scriptable Objects, nous avons pu gérer les propriétés des objets de manière plus efficace et modulaire, facilitant ainsi leur intégration et leur modification sans avoir à instancier chaque fois de nouveaux objets.

- **La potion de vie (Health Potion)**

Comme son nom l'indique, cette potion permet au joueur qui la consomme, de récupérer des points de vie, ce qui lui permet de continuer à progresser dans le niveau. L'effet de cette potion est immédiat : dès qu'elle est consommée, elle ajoute instantanément 20 points de vie au joueur. Cette augmentation de points de vie peut être cruciale dans des moments de combat intense ou lorsque le joueur est proche de perdre toute sa vie, offrant ainsi une seconde chance et permettant de surmonter des obstacles ou des ennemis difficiles. La potion de gain de vie est donc un élément stratégique important, donnant au joueur la capacité de prolonger son aventure et d'affronter les défis avec une meilleure préparation.

- **La potion d'oxygène (Oxygen Potion)**

Comme son nom l'indique, cette potion permet au joueur qui la consomme de recharger sa jauge d'oxygène de quelques unités. L'effet de cette potion est immédiat : dès qu'elle est bue, elle ajoute instantanément 10 unités d'oxygène au joueur. En augmentant temporairement les réserves d'oxygène, cette potion

offre au joueur un sursis précieux pour terminer des niveaux particulièrement exigeants. La potion d'oxygène est donc un élément crucial dans la gestion des ressources et la victoire du joueur dans notre jeu.

- **La potion d'invincibilité (Invincibility Potion)**

Comme son nom l'indique, cette potion confère au joueur une invincibilité temporaire dès qu'elle est consommée. Une fois sous l'effet de cette potion, le joueur devient invulnérable aux attaques et aux dégâts pendant une courte durée. Cet effet dure précisément 5 secondes, offrant ainsi une fenêtre idéale pour s'attaquer à des ennemis puissants ou pour surmonter des obstacles particulièrement dangereux sans craindre de subir des dégâts. La potion d'invincibilité est donc un atout stratégique majeur, permettant au joueur de prendre l'avantage dans des situations critiques, notamment lors des combats contre les boss ou dans des passages difficiles.



(a) Potion de vie



(b) Potion d'oxygène



(c) Potion d'invincibilité

FIGURE 10 – Ensemble des potions

- **L'épée**

Disponible sous plusieurs coloris, en fonction du niveau, l'épée est la première arme que le joueur peut collecter.

Cette arme est essentielle pour le combat rapproché, permettant au joueur de se défendre contre les ennemis qu'il rencontre tout au long du jeu. Les animations de combat ont été soigneusement conçues pour offrir une expérience fluide et immersive. En plus de son rôle crucial dans le gameplay, l'épée sert également à introduire les joueurs aux mécaniques de collecte et d'utilisation des armes dans notre jeu. Elle est facile à trouver et à utiliser, mais ses multiples coloris encouragent les joueurs à explorer les moindres recoins de notre jeu et peut être trouver d'autres items cachés.



FIGURE 11 – Prefab de l'épée

- **Le fusil**

Cette arme apporte une autre dimension d'autant plus stratégique à notre jeu. Plutôt rare dans les niveaux, le fusil est un atout précieux pour celui qui sait l'utiliser. En duo il sera un atout précieux à l'équipe pour battre les boss les plus importants.



FIGURE 12 – Prefab du fusil

11.3 Multijoueur

Pour le réseau, notre objectif principal était de rendre le jeu jouable en mode multijoueur, avec une coopération à deux joueurs comme mentionné précédemment. Pour atteindre cet objectif ambitieux, nous avons utilisé la librairie Photon, et plus précisément PUN 2, qui s'est avérée indispensable pour gérer la connectivité et la synchronisation des joueurs, de l'ensemble des items et des parcours.

La mise en place de ce système de réseau a nécessité de s'assurer que deux joueurs puissent se connecter et jouer ensemble sur le même niveau. Cela impliquait non seulement de synchroniser leurs positions et actions, mais aussi de gérer des éléments complexes tels que les boss, les interfaces utilisateurs (UI) de chaque joueur, et les différents items disponibles sur la carte. Par exemple, la vie des boss doit être partagée et syn-

chronisée entre les deux joueurs.

Notre jeu comporte également des mécaniques de combat rapproché et à distance, ce qui a nécessité une gestion minutieuse des projectiles en multijoueur. Chaque projectile tiré devait être correctement synchronisé entre les deux joueurs pour éviter toute incohérence dans le gameplay.

De manière générale, cet aspect du projet a été l'un des plus chronophages et exigeants. Initialement, nous avons rencontré des difficultés à synchroniser les joueurs avec leurs caméras respectives, ce qui a nécessité de nombreuses heures de développement et de tests pour résoudre les problèmes. Ensuite, synchroniser les mouvements des joueurs et les différentes interfaces utilisateur liées à chaque joueur a représenté un autre défi majeur. Il était crucial que chaque joueur puisse voir et interagir avec sa propre interface utilisateur (UI) sans interférence, tout en maintenant une cohérence parfaite avec l'autre joueur.

Ce processus a été laborieux, mais essentiel pour garantir une expérience de jeu fluide et agréable en mode multijoueur. Grâce à ces efforts, nous avons pu créer un système de réseau robuste et réactif qui permet aux joueurs de profiter pleinement de l'aventure Galaxy Traveller en coopération.

Le multi fonctionne de la manière suivante : le joueur se connecte à un lobby où il rejoint une partie (room) ou en crée une nouvelle. Cette room charge alors le premier niveau.

Chaque objet à synchroniser doit au minimum implémenter un script photonView. Ce script donne un identifiant unique à l'objet sur le réseau. Une fois cet identifiant donné il faut synchroniser les paramètres qui nous intéressent :

- Les joueurs (position, rotation, inventaire, animations, items dans la main)
- Les plateformes (position)
- Les items à collecter (disparaît si collecté)
- Les IA (position, rotation, animations, scripts attaques)
- Les événements en jeu (start, game over...)

Certains de ces paramètres sont simples à synchroniser. Pour la position par exemple, il suffit d'ajouter un script fourni par photon SmoothSyncMovement pour synchroniser

de manière fluide les mouvements des objets.

Cependant pour d'autres paramètres c'est un peu plus compliqué. C'est par exemple le cas des animations qui malgré un script dédié et inclus dans Photon, Photon Animator View, certains paramètres de l'animator, ceux en trigger ne sont pas compatibles avec ce script.

C'est pourquoi il a fallu utiliser des RPC permettant d'exécuter sur l'ensemble des objets ayant le même identifiant une fonction qui va lancer une action, par exemple une animation ou encore la mise à jour des objets dans les mains des joueurs.

11.4 Intelligence artificielle

Les ennemis adopteront une apparence différente en fonction de l'environnement dans lequel ils se trouvent. De plus, il y aura diverses attaques qui seront actives à partir d'un moment spécifique du niveau ou d'un état d'une des entités. Les ennemis pourront uniquement nous attaquer en mêlée au début de notre jeu et au fur et à mesure du temps, pourront développer d'autres attaques, comme par exemple, celle à distance. Chaque ennemi combat sur une plateforme qui est délimitée par son Nav-Mesh Surface pour éviter qu'il ne puisse sortir de l'arène et qu'il puisse trouver le meilleur chemin entre lui et le joueur en contournant de potentiels obstacles.

Nous définissons dans une classe Enemy les points de vie et le rayon à partir duquel il nous attaque, cela va varier en fonction des types d'ennemis. En effet les boss intermédiaires ne seront pas très difficile à combattre comparés aux boss de chaque fin de niveau. Chaque ennemi va prendre en argument une liste contenant un ou plusieurs scripts d'attaque, lui permettant d'avoir accès à plusieurs attaques différentes si besoin.

Un ennemi muni de l'intelligence artificielle va implémenter à la fois la classe Enemy et la classe AiMovement. La principale difficulté lors de la création de ces ennemis était d'organiser correctement la hiérarchisation de manière à adapter les ennemis à nos besoins de manière efficace.

Dans notre jeu, nous avons créé quatre types d'intelligence artificielle : celle qui attaque à distance, celle qui attaque en mêlée, celle qui combine distance et mêlée et celle qui vient exploser sur le joueur. Les intelligences artificielles distance et mêlée sont très similaires l'une et l'autre. Elles possèdent plusieurs scripts en commun, comme "AIMovement" ou "Enemy". Elles possèdent aussi un animator en commun,

qui sera décrit plus tard. Nos quatre intelligences artificielles possèdent une propriété commune, qui est le NavMeshAgent proposé par Unity.

Le principe de celui-ci est assez simple. Il faut choisir une surface sur laquelle l'intelligence artificielle pourra se déplacer. Une fois cela fait, il faut donner à cette surface un composant nommé "NavMeshSurface". Ce composant donnera la possibilité aux intelligences artificielles de se déplacer sur cette surface. Il faut alors donner le composant "NavMeshAgent" à nos IA. Dans ce composant, il y a une propriété nommée "StoppingDistance", qui correspond à la distance à laquelle l'intelligence artificielle doit s'arrêter de notre joueur. Une fois cette propriété remplie, nous possédons une intelligence artificielle fonctionnelle, et il ne reste plus qu'à implémenter les spécifications propres à nos intelligences artificielles.

Le script Enemy que toutes les intelligences artificielles possèdent est une classe abstraite, qui implémente deux méthodes virtuelles : AttackManager et StopAttack. Ce script possède une méthode notifiable qui s'appelle FindAndLaunchAttack. Cette méthode prend en argument un nom d'attaque, et cherche ce nom parmi la liste des attaques que l'intelligence artificielle possède. Si cette liste contient l'attaque cherchée, alors on utilise la méthode LaunchAttack de cette attaque. Sinon, une exception est lancée. Les scripts des attaques seront détaillés plus tard. Les attributs de ce script sont le script de la plate-forme sur laquelle l'intelligence artificielle se trouve, le rayon d'attaque de l'intelligence artificielle, sa liste d'attaques, l'animator de l'intelligence artificielle, les dégâts qu'elle inflige et son nombre de points de vie.

11.4.1 L'intelligence artificielle distance

L'intelligence artificielle distance est un robot adapté aux combats éloignés. Elle est équipée d'un fusil et tire des projectiles qui prennent la forme d'une balle. Ces projectiles seront détaillés plus tard. L'intelligence artificielle distance possède un script propre à elle, le script "EnemyDistance". Ce script hérite de Enemy, et permet d'ajouter la fonction de l'attaque à distance. De plus, notre intelligence artificielle distance pourra s'échapper lorsque le joueur s'approche trop près d'elle. Deux fonctions permettent cette fonctionnalité. Une se trouve dans EnemyDistance, l'autre se trouve dans AIMovement. La première permet de calculer la distance entre le joueur et l'intelligence artificielle pour lui indiquer qu'il faut fuir via l'activation d'un paramètre dans l'animator. La deuxième permet de déplacer l'intelligence artificielle de quelques pas en arrière en prenant en compte les potentiels obstacles situés dans son dos.



FIGURE 13 – IA distance

11.4.2 L'intelligence artificielle mêlée

L'intelligence artificielle mêlée est elle aussi un robot mais plus épais. Elle possède comme arme une épée. L'intelligence artificielle mêlée ne fait que s'approcher et attaquer, elle ne recule pas. Pour cela, elle possède un script nommé EnemyMelee. Celui-ci hérite aussi de Enemy, de manière à override les méthodes déclarées comme abstraites. La seule méthode notifiable est AttackManager, qui permet simplement à l'intelligence artificielle de savoir si elle est en mesure d'attaquer ou non, auquel cas le script la fait attaquer en appelant FindAndLaunchAttack de la classe mère Enemy.



FIGURE 14 – IA mêlée

11.4.3 L'intelligence artificielle explosion

L'intelligence artificielle explosion se caractérise par un petit robot en forme de sphère avec des pattes. Le principe est simple : lorsqu'elle détecte la présence d'un player sur sa plateforme, elle se met en forme de boule et roule précipitamment en direction du player sélectionné. Lorsque la distance entre les deux est inférieure à la distance d'explosion, la boule explose et met des dégâts au player. Et cela se fait

en continu jusqu'à ce qu'il n'y ait plus d'intelligence artificielle explosion ou que les players ne soient plus sur la plateforme.



FIGURE 15 – IA explosion

11.4.4 L'intelligence artificielle mêlée/distance (le boss final)

L'intelligence artificielle mêlée/distance est une sorte de mélange entre l'intelligence artificielle mêlée et l'intelligence artificielle distance. Elle possède un script nommé “EnemyMD” qui hérite de “Enemy”. Ce script possède des méthodes communes à EnemyMelee et EnemyDistance. De plus, elle override la fonction AttackManager. Dans ce script, cette fonction permet à l'intelligence artificielle de savoir quelle attaque utiliser en fonction de la distance au joueur le plus proche. Son principe est assez simple : si le joueur est dans le rayon d'attaque à mêlée, l'intelligence artificielle court vers le joueur pour l'attaquer à mêlée. Sinon, si le joueur le plus proche est dans le rayon d'attaque à distance, alors l'intelligence artificielle range son épée et dégaine son fusil pour tirer sur le joueur les mêmes projectiles que ceux de l'intelligence artificielle distance. Sinon, elle s'approche du joueur le plus proche. Ce script possède une méthode appelée CheckForChanging, qui permet à l'intelligence artificielle de changer d'arme et d'animation selon la distance au joueur le plus proche et l'animation qui était en cours. Le script implémente aussi les méthodes StopAttackMelee et StopAttackDistance, qui permettent de stopper les visuels et les animations en cours pour les adapter à la situation.

11.4.5 Les attaques

Pour pouvoir attaquer, nos intelligences artificielles possèdent une liste d'attaques. Les attaques sont représentées par des scripts. Elles possèdent toutes un script qui hérite de la classe abstraite "Attack". Cette classe mère possède pour attributs un nom, un lanceur (celui qui lance l'attaque), et des dégâts (les points de vie que le joueur

va perdre). Elle implémente aussi une fonction virtuelle qui s'appelle LaunchAttack. Cette fonction permettra à l'intelligence artificielle de lancer son attaque via les classes filles.

- **L'attaque mêlée**

Cette attaque, comme son nom l'indique, est à utiliser en combat rapproché. Ce sont donc nos intelligences artificielles mêlée et mêlée/distance qui la possèdent. Elle est référencée par le script AttackMelee, qui hérite de la classe Attack. Elle override la méthode LaunchAttack et active le collider de l'épée. Cela va servir pour le script de l'épée.

- **L'attaque distance**

Cette attaque est, elle aussi assez explicite : elle permet aux intelligences artificielles distance et mêlée/distance de pouvoir attaquer à distance grâce à leur fusil. Elle est référencée par le script AttackDistance, qui hérite de la classe Attack. Elle override la méthode LaunchAttack et, dans cette méthode, elle instancie un projectile au bout du fusil afin d'ensuite lui appliquer une force qui permet de tirer le projectile.

- **L'attaque explosion**

Cette attaque est celle possédée seulement par l'intelligence artificielle. Elle est référencée par le script AttackExplosion, qui hérite de la classe Attack. Elle override la méthode LaunchAttack. Dans cette méthode, le script fait disparaître l'ia et fait apparaître une explosion à sa place, infligeant de lourds dégâts au joueur.



FIGURE 16 – Explosion d'une petite IA

- **Les projectiles**

Les projectiles sont les mêmes pour l'attaque du joueur et de nos intelligences artificielles distance et mêlée/distance. Ces projectiles sont référencés sous forme de simples capsules. Ils possèdent un collider, un rigidbody et un script. Ce script

permet de détecter les collisions grâce à la méthode de Unity "OnCollisionEnter". Si le collider avec lequel le projectile est entré en collision appartient à un joueur ou à un ennemi, alors le script fait perdre des points de vie à ce joueur ou à cet ennemi.

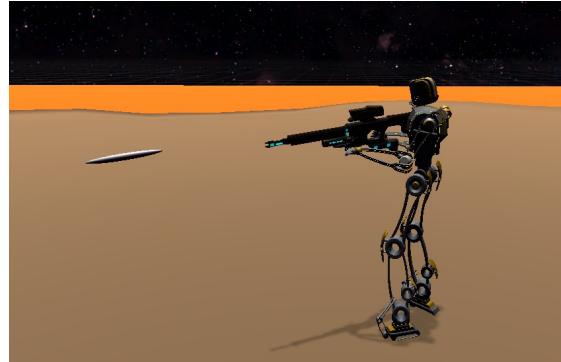


FIGURE 17 – Projectile d'une IA à distance

- **L'épée**

L'épée de l'IA possède un visuel, un collider et un script. Son collider est de base désactivé. Lorsque l'intelligence artificielle attaque en mêlée, alors son collider devient actif. Le script utilise la méthode OnTriggerEnter de Unity. Si le collider avec lequel l'épée entre en collision appartient à un joueur, alors le script inflige fait perdre des points de vie à ce joueur.

11.4.6 L'Animator des intelligences artificielles

Nos intelligences artificielles Mêlée, Distance et Mêlée/Distance possèdent le même animator. Celui-ci est appelé IAAimator. Il possède 5 layers :

- **Base layer**

Ce layer implémente les états Run_backward et Run_forward. Ces états sont assez explicites : le premier contient une animation qui est appelée si la vitesse de l'ennemi est supérieure à une unité et que le paramètre indiquant si l'intelligence artificielle doit reculer est non activé. Le deuxième contient une animation qui est appelée si la vitesse de l'ennemi est supérieure à une unité et que le paramètre indiquant si l'intelligence artificielle doit reculer est activé.

- **Attack Layer**

Ce layer, lui, contient 3 états : l'attaque distance, l'attaque mêlée et le recul. En effet, si l'intelligence artificielle prend des dégâts, elle subit un léger recul, lui

empêchant d'attaquer pendant ce temps. L'état de recul contient une animation qui est appelée lorsque l'intelligence artificielle subit des dégâts. L'état d'attaque à distance contient une animation qui est appelée lorsque l'intelligence artificielle attaque en distance. Cette animation appelle une fonction nommée "IncreaseNbShots" qui permet de savoir combien de coups l'intelligence artificielle a tiré (cela nous sert pour ne pas tirer quand l'intelligence artificielle n'est pas prête à tirer) et une autre fonction nommée "FinishAnim" qui permet de réinitialiser les paramètres de l'Animator pour vérifier s'il faut continuer d'attaquer ou non. L'état d'attaque mêlée contient une animation qui est appelée lorsque l'intelligence artificielle attaque en mêlée. Cette animation appelle la même fonction nommée "FinishAnim" qui permet de réinitialiser les paramètres de l'Animator pour vérifier s'il faut continuer d'attaquer ou non.

- **Shaking Layer**

Ce layer, pour sa part, contient un seul état : Il se nomme shaking. Cet état contient juste une animation qui fait serrer la main à l'intelligence artificielle, de manière que son épée puisse tenir dans sa main sans que ce soit choquant visuellement. Une difficulté rencontrée lors de la mise en place de cette animation a été le fait que cette animation n'annule pas toutes les autres en cours. Pour régler ce problème, il a fallu créer un Avatar mask pour ne rendre l'animation active que sur les mains sans influencer sur le reste du corps. Ainsi, l'intelligence artificielle peut se déplacer ou attaquer avec la main serrée.

- **Holding Weapon Layer**

Ce layer, pour sa part, contient un seul état : Il se nomme holding. Cet état contient juste une animation qui fait serrer la main à l'intelligence artificielle, de manière que son épée puisse tenir dans sa main sans que ce soit choquant visuellement. Une difficulté rencontrée lors de la mise en place de cette animation a été le fait que cette animation n'annule pas toutes les autres en cours. Pour régler ce problème, il a fallu créer un Avatar mask pour ne rendre l'animation active que sur les mains sans influencer sur le reste du corps. Ainsi, l'intelligence artificielle peut se déplacer ou attaquer avec la main serrée.

- **DeathLayer**

C'est le dernier Layer de l'Animator. Celui-ci contient un seul état nommé Death, impliquant une animation de mort de l'intelligence artificielle. Cette animation ne s'active que lorsque les points de vie de l'intelligence artificielle passent à zéro ou en dessous, et ne donne lieu à aucune animation par la suite. L'intelligence

artificielle reste allongée au sol et ne disparaît pas.

11.4.7 L'Animator de l'intelligence artificielle explosion

Le BaseLayer est le seul contenu dans cet animator. Il contient 6 états : Idle, Turn, Walk, OpenToRoll, Rolling, RollToOpen :

- Le premier état correspond au fait de ne pas se déplacer et d'attendre.
- Le deuxième état permet au robot de se tourner sur lui-même pour localiser les joueurs. Celui-ci contient une animation de rotation, appelée lorsque au moins un player arrive sur la plate-forme où se trouve l'intelligence artificielle.
- Le troisième état permet tout simplement au robot de se déplacer de manière fluide et non linéaire. Il contient une animation du robot qui marche de manière déterminée et qui est appelée lorsque le robot est en mode ouvert et que sa vitesse est supérieure à 1 unité.
- Le quatrième état permet au robot de se mettre en boule pour attaquer le joueur. Il contient une animation qui passe de robot ouvert à robot fermé, et qui est appelée lorsque l'intelligence artificielle a un joueur en visuel en face d'elle.
- Le cinquième état est un état d'attaque : c'est un état dans lequel le robot est en boule et peut se déplacer, jusqu'à toucher le joueur pour lui infliger des dégâts. La vitesse du robot est augmentée lorsqu'il est en boule, et le robot ne peut pas subir de dégâts en étant dans cet état.
- Le sixième et dernier état contient une animation pour permettre au robot de passer d'état fermé à état ouvert. Celle-ci est appelée lorsque l'intelligence artificielle ne détecte plus aucun joueur sur la plate-forme sur laquelle elle se trouve.

11.5 Parcours

Tout au long de notre jeu, les joueurs sont amenés à rencontrer différents types de plateformes dont les caractéristiques sont en accord avec l'environnement choisi pour le niveau. De cette manière, dans le niveau de glace, on retrouve des plateformes glissantes et dans le niveau “fantasy”, on retrouve des plateformes rebondissantes.

Il existe également des plateformes plus générales que l'on retrouve dans chaque niveau, comme les plateformes qui disparaissent une fois que l'on passe dessus, les plateformes qui se déplacent d'un point A à un point B ou encore des plateformes pièges.

Concernant l'aspect de nos plateformes, nous avons décidé pour des raisons de délais et de simplicité de garder pour chaque niveau les mêmes corps de plateformes. Cependant, d'un niveau à l'autre, leur apparence diffère toujours en accord avec le biome du niveau.

Ces plateformes nous permettent alors de créer différents niveaux plus ou moins durs, c'est-à-dire que le niveau 1 est moins dur que le niveau 2 qui est lui même plus facile que le niveau 3 et ainsi de suite. Ces parcours sont concoctés de A à Z par nos soins et nous sommes sûrs qu'ils sont réalisables (avec plus ou moins d'entraînement bien sûr). De plus, nous faisons en sorte que chaque parcours comprend au moins un type de plateformes spécifique à l'environnement du niveau, mais également un ennemi intermédiaire/un obstacle qui pourrait lui faire perdre de la vie ainsi qu'un boss de fin qu'il pourra combattre à l'aide des items qu'il a collecté tout au long du parcours.

Sur le niveau 1, cette plateforme spécifique se caractérise par une attaque de petite sphère qui explose et nous inflige des dégâts lorsqu'elle se rapproche du joueur.

Sur le niveau 2, elle se caractérise par une plateforme en pente où le player rencontre des boules de neiges qui tombent du ciel et qu'il devra éviter. Si la boule de neige rentre en contact avec un des players, il perdra des dégâts et reculera.

Sur le niveau 3, la plateforme sera un mini parcours où il faut sauter de plateforme en plateforme avant qu'elle ne disparaisse derrière un mur pour nous forcer à sauter sur une autre. Cela renforce bien l'aspect speedrun de notre jeu.

On pourra retrouver sur plusieurs niveaux un enchaînement de plateforme qui disparaît lorsque le joueur rentre en contact avec. Ce qui ne lui laisse pas d'autre choix que d'avancer au plus vite.

Les plateformes sur lesquelles se trouvent des ennemis implémentent un script, qui permet d'enregistrer les joueurs présents sur ces plateformes grâce à une liste. Elle implémente une deuxième fonction qui retire les joueurs de la liste de joueurs s'ils ne sont plus présents sur la plate-forme après plus de deux secondes.

11.5.1 Plate-formes

Nous avons créé des prefabs de plateformes qui nous permettent de les réutiliser à différents endroits. Nous avons aussi développé des scripts pour assigner différents

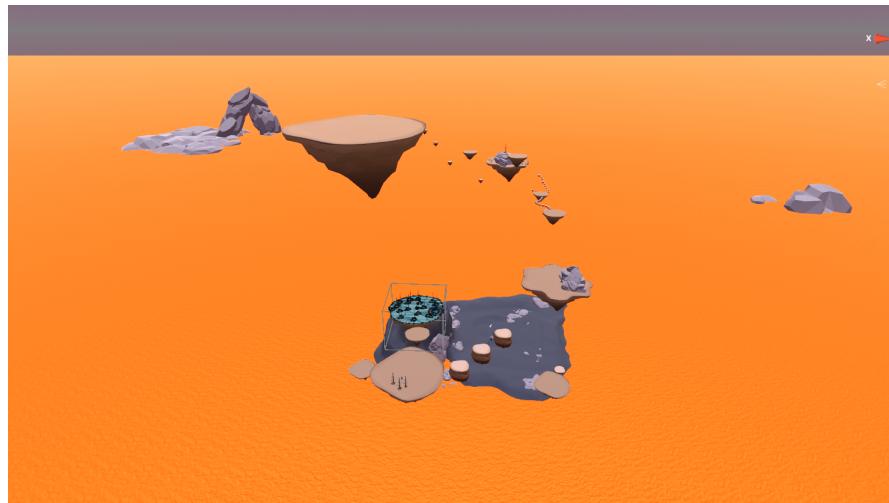


FIGURE 18 – Niveau 1

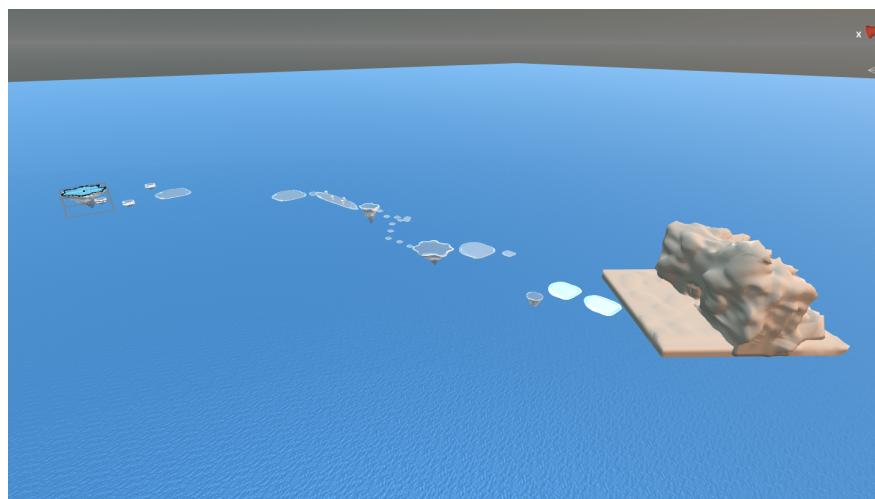


FIGURE 19 – Niveau 2

comportements à celles-ci. Par exemple, nous avons des plateformes qui se déplacent de haut en bas et/ou de gauche à droite en suivant des “waypoints” (le script peut suivre autant de waypoints que nécessaire).

Nous avons aussi fait en sorte que les joueurs restent “collés” aux plateformes lorsqu’elles se déplacent. Nous avons également élaboré des plateformes qui disparaissent lorsqu’un joueur reste dessus pendant plus de n secondes et qui réapparaissent ensuite.

11.5.2 Terrain de spawn

Le terrain de spawn est l’endroit où les joueurs arrivent pour la première fois. Il est en quelque sorte la plateforme de départ. Contrairement aux autres plateformes

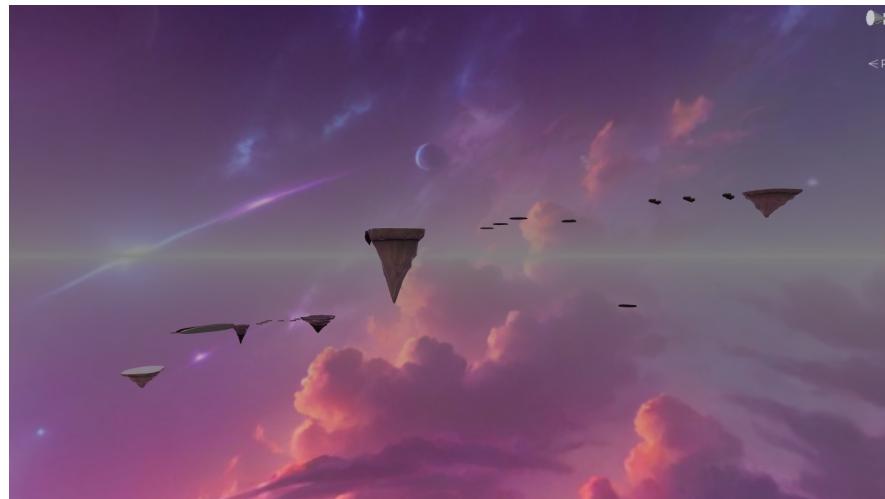


FIGURE 20 – Niveau 3

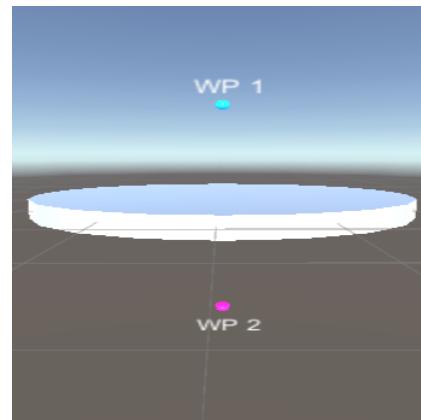


FIGURE 21 – Plateforme avec deux waypoints haut/bas

qui sont des objets 3D classiques, ce terrain est un objet spécifique lui permettant d'être sculpté pour donner un terrain non régulier réalisable depuis Unity et en temps réel. Il peut être peint de plusieurs textures mais pour le premier niveau la texture est uniforme. Nous avons ainsi utilisé les outils disponibles de ce type pour réaliser le volcan de cette première plateforme.

11.5.3 Surface de respawn

Ces surfaces sont présentes à différents endroits du jeu et font mourir le joueur s'il entre en contact avec elles. Pour le premier niveau elles abordent un matériel de lave.

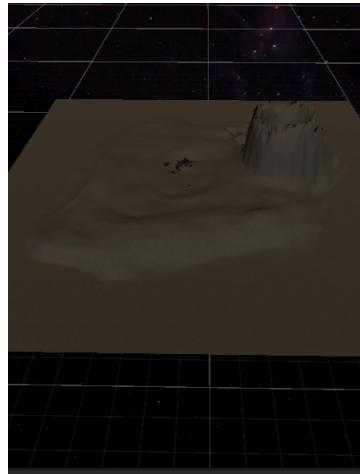


FIGURE 22 – Terrain du premier niveau



FIGURE 23 – Terrain du premier niveau avec le lac de lave

11.6 Interface utilisateur

11.6.1 Menus

La partie menus se divise en deux catégories : les menus principaux et le menu de pause. Pour ces menus nous avons utilisé UGUI, l'ancienne manière de créer les UI dans Unity. Ce choix s'explique par un manque de documentation et d'explications sur le nouvel outil nommé UIToolkit.

Ils nous permettent de lancer une partie solo, une partie multijoueur, d'accéder à une page des crédits et de quitter le jeu. Le menu de réglage permet de personnaliser différents éléments du jeu.

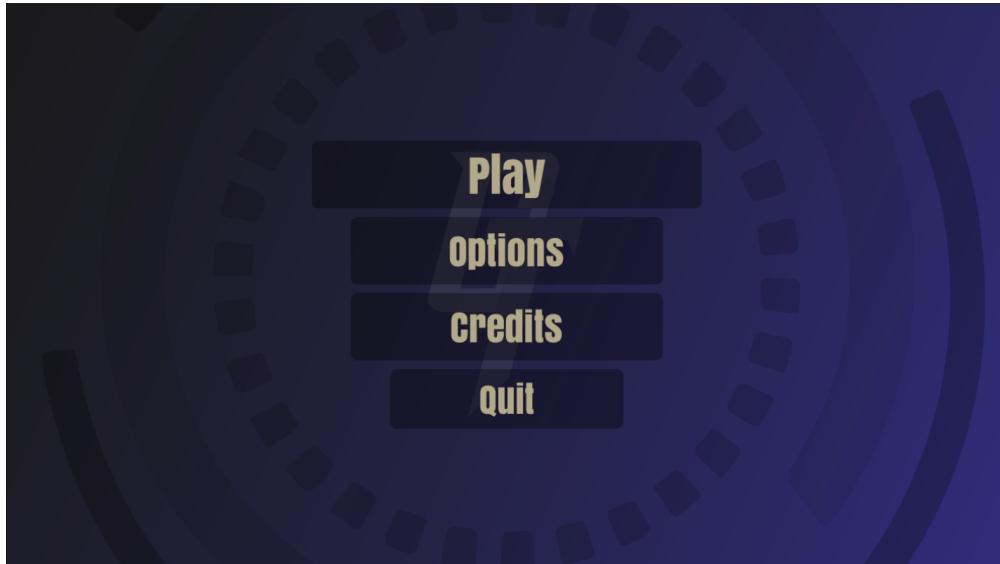


FIGURE 24 – Main Menu

11.6.2 En jeu

- **La barre de vie :** La barre de vie est verte et représente le pourcentage de vie du joueur. Cette barre de vie est propre au joueur, son coéquipier ne peut pas la voir. Elle diminue lorsque le joueur perd des points de vie/prend des dégâts et elle augmente lorsqu'il récupère de la vie, notamment en consommant une potion de vie par exemple.
- **La barre d'oxygène :** La barre d'oxygène est bleue et elle diminue de seconde en seconde, laissant un temps imparti au joueur pour terminer le niveau. Plus le niveau est dur, moins il a d'oxygène, et donc de temps, pour terminer le niveau. Chaque joueur voit sa propre barre d'oxygène sans avoir accès à celle de son coéquipier. Lorsque le joueur arrive à court d'oxygène, il voit sa vie diminuer petit à petit ainsi que sa vitesse de déplacement réduite jusqu'à mourir.
- **L'inventaire :** Il est composé de quatre cases pouvant chacune contenir un sprite d'item. La molette de la souris permet de sélectionner un item.

11.7 Sons

Dans le développement de notre jeu, une attention particulière a été portée à l'intégration des effets sonores pour enrichir l'expérience utilisateur et renforcer l'immersion. Les effets sonores sont une composante essentielle pour rendre le monde virtuel plus vivant et réaliste. Nous avons incorporé une variété de sons correspondant à différentes

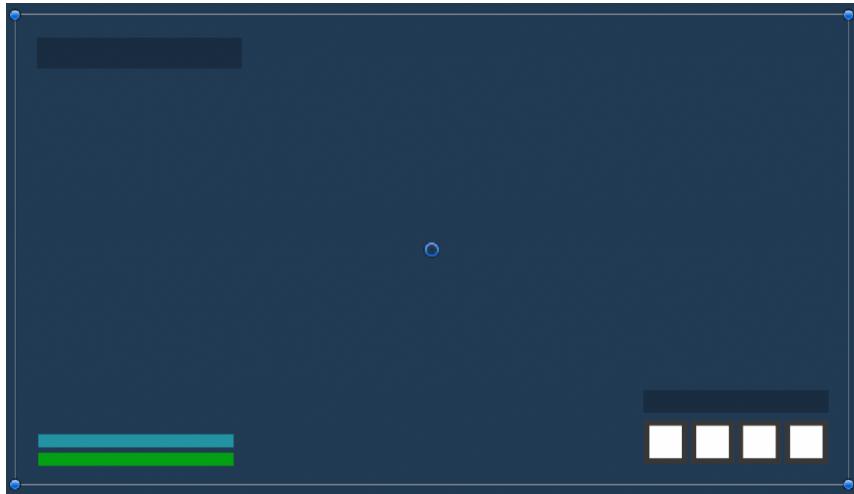


FIGURE 25 – UI du player en jeu

actions et environnements dans le jeu.

• **Bruits de pas**

Les bruits de pas jouent un rôle crucial en fournissant un retour auditif immédiat lorsque le personnage du joueur se déplace. Chaque type de surface sur laquelle le personnage marche – que ce soit de la terre, de la pierre ou de la glace – a des sons distincts associés, ajoutant ainsi une couche de réalisme au jeu. Ces bruits ont été soigneusement sélectionnés et ajustés pour s'adapter au rythme et au type de mouvement du personnage.

• **Les sons des collectables**

Lorsque le joueur ramasse des objets collectables, des sons spécifiques sont déclenchés pour signaler la réussite de l'action. Ces sons sont conçus pour être satisfaisants et distincts, aidant le joueur à identifier instantanément l'action réalisée. Cela contribue à renforcer l'engagement du joueur en fournissant un retour positif immédiat pour ses actions.

• **Les bruits d'ambiance et les sons d'environnement**

Les bruits d'ambiance et les sons environnementaux sont utilisés pour créer une atmosphère immersive et crédible dans le jeu. Que ce soit le murmure de la lave dans un volcan ou le tassemement de la neige sous nos pas, ces sons d'arrière-plan enrichissent le paysage sonore et transportent le joueur dans l'univers du jeu. Chaque zone du jeu a été équipée de sons spécifiques correspondant à son environnement, créant ainsi une expérience sonore cohérente et immersive.

- **Les effets sonores de combats**

Les effets sonores de combats sont essentiels pour ajouter de l'intensité et du dynamisme aux scènes d'action. Nous avons inclus des sons pour les coups et les explosions, chacun soigneusement synchronisé avec les animations correspondantes. Ces sons sont conçus pour être percutants et clairs, permettant aux joueurs de ressentir la puissance et l'impact de chaque action.

- **La musique et les sons dans les menus**

Outre les effets sonores dans le jeu, nous avons également intégré différentes musiques dans les menus pour améliorer l'expérience utilisateur dès le lancement du jeu. Chaque menu dispose de sa propre bande sonore, choisie pour correspondre à l'ambiance et à la fonction du menu. Par exemple, le menu principal peut avoir une musique épique et inspirante, tandis que les sous-menus peuvent avoir des morceaux plus calmes et subtils, facilitant la navigation et créant une atmosphère agréable.

11.8 Installation

Avant de créer l'installateur, nous avons veillé à ce que notre jeu soit entièrement développé et testé. Unity nous permet de compiler le jeu pour différentes plateformes ; dans notre cas, nous avons choisi de cibler la plateforme Windows. La compilation du projet a généré un dossier contenant tous les fichiers nécessaires, y compris les exécutables, les bibliothèques dynamiques (DLL), et les ressources du jeu. Cette étape est cruciale car elle garantit que tous les éléments nécessaires au bon fonctionnement du jeu sont prêts à être inclus dans l'installateur.

Nous avons ensuite personnalisé le script généré pour répondre aux besoins spécifiques de notre projet. Par exemple, nous avons ajouté une icône personnalisée pour l'installateur afin de renforcer l'identité visuelle de notre jeu. De plus, nous avons configuré les paramètres de désinstallation pour garantir que tous les fichiers soient correctement supprimés si l'utilisateur décide de désinstaller le jeu.

Une fois le script personnalisé, nous l'avons sauvegardé avec une extension .iss et avons utilisé l'option de compilation dans Inno Setup. Le processus de compilation a généré un fichier EXE, qui est l'installateur de notre jeu.

11.9 Site web

Le site Internet a été entièrement développé par nos soins, démontrant notre capacité à créer des solutions web de qualité professionnelle. Utilisant le puissant framework ReactJS, nous avons conçu un site performant, réactif et moderne qui répond parfaitement aux besoins de notre projet.

Accessible depuis janvier 2023 à l'adresse suivante : <https://galaxytraveller.fr>, ce site représente une vitrine complète de notre jeu Galaxy Traveller. Nous y avons intégré diverses fonctionnalités pour enrichir l'expérience des visiteurs, telles qu'une navigation intuitive ainsi que des sections interactives.

Le choix de ReactJS s'est avéré judicieux pour garantir une excellente performance et une grande modularité, nous permettant d'ajouter facilement de nouvelles fonctionnalités au fur et à mesure de l'évolution du projet. L'architecture du site a été pensée pour être évolutive, assurant ainsi une maintenance simplifiée et la possibilité d'intégrer rapidement des mises à jour.

La mise en ligne du site marque une étape importante dans notre projet, nous permettant de partager notre travail avec le monde entier et de recevoir des retours précieux de la part de la communauté des joueurs et des développeurs. Nous sommes fiers du résultat et convaincus que le site continuera de jouer un rôle crucial dans la promotion et le support de Galaxy Traveller.

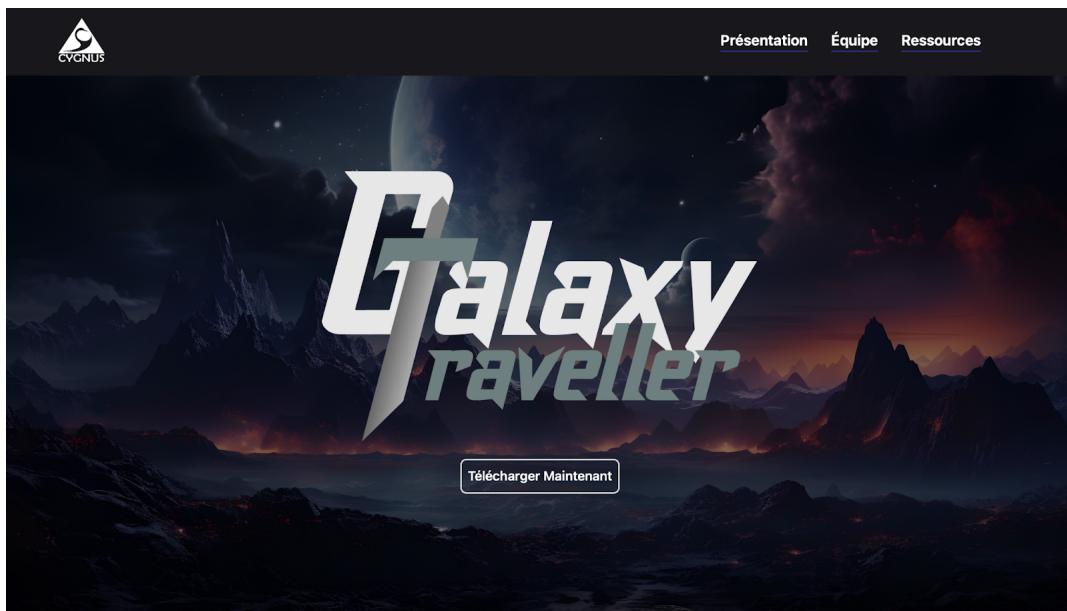


FIGURE 26 – Site du jeu

Sur notre site, vous avez la possibilité de découvrir une présentation détaillée de l'équipe Cygnus ainsi qu'une description approfondie du projet Galaxy Traveller. Cette section inclut des informations sur les rôles de chaque membre de l'équipe. Vous y trouverez également une multitude de documents téléchargeables, tels que des rapports de projet ou des guides d'utilisateur. Ces documents fournissent des informations précieuses et complètes sur les différentes étapes de notre développement et les technologies que nous avons utilisées. De plus, nous avons rassemblé une collection de sources et de références qui nous ont été essentielles tout au long du processus de développement. Ces ressources incluent des tutoriels, des documentations et des outils logiciels qui nous ont aidés à surmonter divers défis techniques.

Le site a été conçu pour être entièrement responsive, offrant une expérience de navigation optimale sur les téléphones mobiles, les tablettes et les ordinateurs de bureau. Nous avons porté une attention particulière à l'ergonomie et à la facilité d'utilisation pour garantir que les utilisateurs puissent accéder à toutes les informations et fonctionnalités du site sans difficulté, quel que soit l'appareil utilisé.

Enfin, le développement de notre site internet n'a rencontré aucun obstacle majeur grâce à l'expertise de nos membres en la matière. Ayant déjà créé plusieurs sites web auparavant, notre équipe disposait des compétences nécessaires pour mener à bien ce projet avec efficacité et professionnalisme. Chaque étape, de la conception initiale à la mise en ligne, a été soigneusement planifiée et exécutée pour assurer un résultat de haute qualité.

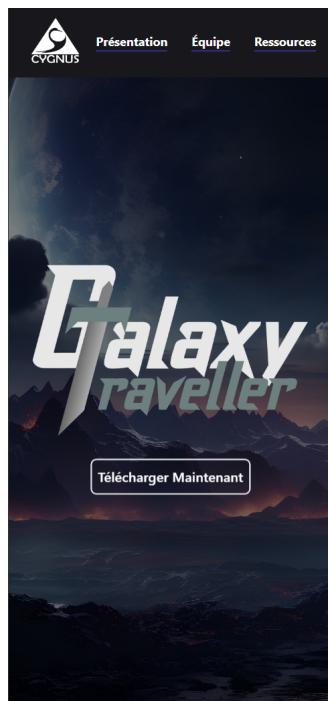


FIGURE 27 – Responsive du site

Cinquième partie

Bilan

12 Ressenti global

Après une année intense de travail collectif sur ce projet, le ressenti global du groupe est un mélange de fierté, de soulagement et de satisfaction. Chacun a contribué avec passion et détermination, surmontant les défis et célébrant les succès ensemble. La progression du projet, depuis ses premières étapes jusqu'à sa finalisation, a renforcé les liens au sein de l'équipe.

La fierté est largement partagée, car le groupe reconnaît les efforts fournis et la qualité du résultat obtenu. Les moments de doute et de difficulté se sont transformés en anecdotes partagées avec un sourire, témoignant de la résilience collective. Le soulagement est aussi présent, car après tant de mois de travail acharné, la pression retombe enfin, permettant à chacun de se détendre et de réfléchir au chemin parcouru.

La satisfaction est évidente, non seulement parce que les objectifs ont été atteints, mais aussi parce que chacun a grandi au cours de cette année. Les compétences développées, les leçons apprises et les relations établies sont des acquis précieux qui perdureront bien au-delà du projet lui-même. La collaboration a révélé des talents cachés, renforcé la confiance mutuelle et posé les bases pour de futures aventures communes.

En rétrospective, cette année a été plus qu'un simple projet. Elle a été une expérience humaine enrichissante, un voyage au cœur de la dynamique de groupe et de la réalisation collective. Le groupe en sort renforcé, prêt à relever de nouveaux défis avec la certitude qu'ensemble, tout devient possible.

13 Ressenti individuel

13.1 BELPERIN Noé

Durant ce projet, je me suis majoritairement occupé de la création des niveaux, de l'intelligence artificielle ainsi que la gestion des sons et musiques. J'ai pris plaisir à conceptualiser les niveaux dans ma tête pour ensuite les créer un rendu qui se lient

aux jeux de plateforme. J'ai appris à travailler en équipe et à affronter les problèmes que moi ou mon groupe avons pu rencontrer. Je suis très satisfait de mon travail et j'en sors avec beaucoup de connaissances.

13.2 BLANC Johan

Pour ma part, je suis chargé principalement des armes et collectables, de l'implémentation de l'intelligence artificielle et de la partie interface utilisateur, j'ai durant ce projet, appris énormément tant sur le point technique que sur la manière de travailler. En tant que chef de groupe, j'ai donné tout mon possible pour une meilleure cohésion et productivité.

13.3 DEFONTAINE JOLIVET Emilien

De mon côté, je suis principalement en charge du site web, du multijoueur et la conception des niveaux. J'ai pu exploiter et enrichir mes connaissances sur le site web. Ce projet m'a appris à séparer le travail d'équipe et le travail individuel pour parvenir à un meilleur résultat.

13.4 THOMÉ Aubin

Dans ce projet, ma principale responsabilité a été la programmation fonctionnelle. Mon objectif était d'harmoniser tous les scripts afin que le travail de chaque membre s'intègre parfaitement. J'ai su relever les défis du groupe et aider mes coéquipiers pour atteindre un résultat final positif. J'ai pris beaucoup de plaisir à travailler sur ce projet avec mon équipe, à m'investir dans mes tâches.

Sixième partie

Conclusion

Ce projet de développement de jeu vidéo a été une expérience profondément enrichissante et formative pour toute l'équipe. Au fil des mois, nous avons su surmonter les défis techniques et organisationnels qui se sont présentés, tout en maintenant une dynamique de groupe positive et productive.

Le choix d'un jeu de plateforme 3D en coopération, avec un thème spatial, s'est avéré judicieux, stimulant notre créativité et nous permettant de mettre en pratique une large palette de compétences en programmation, design et gestion de projet. Chaque membre de l'équipe a apporté sa contribution unique, qu'il s'agisse de la conception des niveaux, de l'intelligence artificielle, des graphismes, du son, ou de l'intégration multijoueur.

Le départ de l'un des membres a certes été un défi, mais il nous a permis de réorganiser nos tâches et de renforcer notre résilience en tant qu'équipe. Cette épreuve a illustré l'importance de la flexibilité et de la communication au sein d'un projet collaboratif.

Nous avons intégré des éléments classiques et innovants pour créer une expérience de jeu unique. Les diverses mécaniques de jeu, l'intelligence artificielle sophistiquée des ennemis, les différents types de plateformes, ainsi que le mode multijoueur, apportent une profondeur et une richesse qui, nous l'espérons, captiveront les joueurs.

Chaque aspect du projet, depuis la première conceptualisation jusqu'aux tests finaux, a été réalisé avec rigueur et passion.

En conclusion, Galaxy Traveller n'est pas seulement le fruit de notre travail acharné et de nos compétences techniques, mais aussi de notre passion commune pour le jeu vidéo et de notre capacité à collaborer efficacement. Nous sommes fiers du produit final et confiants qu'il offrira de bons moments de divertissement aux joueurs. Cette aventure nous a permis de grandir tant技iquement que personnellement, et nous sommes prêts à relever de nouveaux défis avec l'enthousiasme et la détermination que ce projet a su cultiver en nous.

L'équipe *Cygnus*.