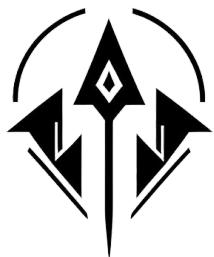


Rapport de la soutenance finale  
*Lands of Azerith*



Ayemane Bouarbi Louise Fussien  
ayemane.bouarbi@epita.fr louise.fussien@epita.fr

# Stonks Industries

## EPITA

18 juin 2024



# Table des matières

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Objectif de la soutenance et du projet . . . . .	5
1.2	Synopsis du jeu . . . . .	6
1.3	Notre entreprise . . . . .	7
1.4	L'équipe . . . . .	7
1.4.1	Ayemane Bouarbi . . . . .	7
1.4.2	Louise Fussien . . . . .	8
1.4.3	Michaël Museux . . . . .	8
1.4.4	Martin Pasquier . . . . .	9
1.5	Changements depuis la dernière soutenance . . . . .	9
1.6	Annonce de plan . . . . .	9
<b>2</b>	<b>Gestion des échéances et organisations</b>	<b>11</b>
2.1	Analyse et progrès . . . . .	11
2.1.1	Status actuel . . . . .	11
2.1.2	Délais rencontrés et raisons . . . . .	11
2.2	Organisation du groupe . . . . .	11
2.2.1	Communication . . . . .	11
2.2.2	GitHub . . . . .	12
2.2.3	Répartition des tâches . . . . .	12
2.2.4	Cahier des charges techniques . . . . .	13
<b>3</b>	<b>Partie technique</b>	<b>16</b>
3.1	Environnement de travail . . . . .	16
3.1.1	Introduction à l'Environnement de Travail . . . . .	16
3.1.2	Fonctionnement de Godot et Intégration avec Rider . . . . .	16
3.1.3	Défis Associés aux Changements de version de Godot . . . . .	16
3.2	Manipulation des Fichiers JSON : Défis et Solutions . . . . .	17
3.2.1	Introduction à JSON et son Utilisation dans le Développement de Jeux . . . . .	17
3.2.2	Compréhension et Manipulation des Fichiers JSON . . . . .	17
3.2.3	Structure et Syntaxe de JSON . . . . .	18
3.3	L'architecture du jeu . . . . .	19
3.3.1	Structure des Dossiers . . . . .	19
3.4	Menu du Jeu . . . . .	21
3.4.1	Introduction à l'Effet Parallax dans le Menu . . . . .	21
3.4.2	Implémentation de l'Effet Parallax . . . . .	21
3.4.3	Structure de la Node ParallaxBackground . . . . .	21
3.4.4	Animation et Effet Visuel . . . . .	22
3.5	Le Multijoueur . . . . .	22

3.5.1	Système de Connexion Multi-joueurs via Envoi de Paquets . . . . .	23
3.5.2	Fonctionnement du Système . . . . .	23
3.5.3	Avantages du Système de Paquets . . . . .	23
3.6	Contrôles et Interaction dans le Jeu . . . . .	24
3.6.1	Actions Spécifiques . . . . .	24
3.7	Intelligence Artificielle : Comportements des Monstres . . . . .	25
3.7.1	Types de Monstres . . . . .	25
3.7.2	Implémentation Technique . . . . .	26
3.8	Déplacement des Monstres avec Navigation A* . . . . .	27
3.8.1	Algorithme A* . . . . .	27
3.8.2	Implémentation de la Node <code>Navigation2D</code> . . . . .	27
3.8.3	Déplacement Dynamique des Monstres . . . . .	28
3.8.4	Aléatoire dans le Rayon de Déplacement . . . . .	28
3.8.5	Spawning des Monstres dans une Zone Définie . . . . .	28
3.9	Gestion de l'inventaire du joueur . . . . .	29
3.9.1	Structure de l'Inventaire . . . . .	29
3.9.2	Interaction avec l'Inventaire . . . . .	29
3.10	Gestion des Points de Vie, Checkpoints et Loot dans le Jeu . . . . .	29
3.10.1	Points de Vie du Joueur . . . . .	29
3.10.2	Calcul des Dégâts . . . . .	29
3.10.3	Gestion des Checkpoints . . . . .	30
3.10.4	Mécanique de Loot des Monstres . . . . .	30
3.11	Génération de Map et Gestion des Transitions avec Nodes dans Godot .	30
3.12	Génération de la Map et Utilisation des Textures . . . . .	31
3.12.1	Textures et Tiles . . . . .	31
3.12.2	Tileset pour les Maps . . . . .	31
3.13	Gestion des Transitions entre les Maps . . . . .	31
3.13.1	Nodes de Transition . . . . .	32
3.13.2	Changement Dynamique de la Map . . . . .	32
3.14	Animation des Personnages et des Objets . . . . .	32
3.14.1	Sprites Animés . . . . .	32
3.14.2	Création de Mouvements . . . . .	32
<b>4</b>	<b>Design et musique</b>	<b>33</b>
4.1	Design . . . . .	33
4.1.1	Notre logo . . . . .	33
4.1.2	Inspirations Design . . . . .	35
4.1.3	Création et implémentation design . . . . .	36
4.1.4	Utilisation du logiciel GIMP . . . . .	39
4.1.5	Processus de Création Collaboratif . . . . .	40
4.1.6	Intégration des Textures dans Godot . . . . .	40
4.2	Musique et bruitages . . . . .	41
4.2.1	Musiques . . . . .	41

4.2.2	Bande sonore . . . . .	41
4.2.3	Exemples de Musiques de Biomes . . . . .	41
4.2.4	Musiques des Boss et Zones Pacifiques . . . . .	42
4.2.5	Voix Off . . . . .	42
<b>5</b>	<b>Site web</b>	<b>43</b>
5.1	Objectif du site web . . . . .	43
5.2	Développement . . . . .	43
5.3	Apparence . . . . .	44
5.4	Responsive design . . . . .	44
<b>6</b>	<b>Conclusion</b>	<b>46</b>
6.1	Récapitulatif . . . . .	46
6.2	Les joies et les peines . . . . .	49
6.3	Remerciements . . . . .	49

# 1 Introduction

## 1.1 Objectif de la soutenance et du projet

Dans le cadre de notre formation, nous avons entrepris un projet ambitieux visant à développer un jeu vidéo. Ce projet, qui s'étend sur l'ensemble de l'année académique, nous a permis d'acquérir et de perfectionner nos compétences en programmation, en design de jeux et en gestion de projet. La soutenance de fin d'année marque l'aboutissement de nos efforts collectifs et individuels. Elle offre une occasion privilégiée de présenter l'évolution du projet, les défis surmontés et les résultats obtenus.

Nous sommes fiers de notre travail et des obstacles que nous avons surmontés pour réaliser *Lands of Azerith*, et nous sommes impatients de partager notre expérience avec vous.

Ce rapport de soutenance, élaboré par l'équipe composée de Louise Fussien, Martin Pasquier, Ayemane Bouarbi et Michaël Museux, présente en détail le projet *Lands of Azerith*, un jeu de plateforme multijoueur innovant développé dans le cadre de notre formation. *Lands of Azerith* se distingue par son concept unique où les joueurs évoluent dans un monde en 2D, surmontant des obstacles et résolvant des énigmes pour progresser dans le jeu.

Le développement de *Lands of Azerith* s'est déroulé en plusieurs phases, marquées par des défis techniques et des moments de réflexion collective. La première étape a consisté en une analyse approfondie des mécaniques de jeu existantes, suivie d'une étude de faisabilité pour définir les grandes lignes du projet. Ensuite, nous avons élaboré un cahier des charges détaillé, précisant les fonctionnalités principales du jeu, les contraintes techniques, et les délais à respecter.

Chaque membre de l'équipe a apporté ses compétences spécifiques, qu'il s'agisse de programmation, de design graphique, de création sonore, ou de développement de l'intelligence artificielle. Cette collaboration étroite nous a permis de surmonter de nombreux obstacles et de progresser de manière constante vers notre objectif. Nous avons utilisé des outils tels que Godot pour le développement du jeu, et GDU pour la modélisation en 2D, guidés par la volonté de créer un produit final de haute qualité.

Ce rapport de soutenance décrit en détail chaque aspect du projet *Lands of Azerith*, depuis sa conception initiale jusqu'à sa réalisation finale. Il inclut des sections sur les choix technologiques, la répartition des tâches, les défis rencontrés, et les solutions apportées. En documentant ce processus, nous espérons fournir une vue d'ensemble claire et complète de notre travail, ainsi qu'un témoignage de notre capacité à travailler en équipe et à innover dans le domaine du jeu vidéo.

## 1.2 Synopsis du jeu

*Lands of Azerith* est un jeu d'aventure immersif où le joueur incarne un héros cherchant à libérer les terres d'*Azerith* de l'emprise tyrannique d'un empire dictatorial. L'histoire principale se déroule en trois actes, mais pour l'instant, nous nous concentrerons uniquement sur l'acte 1, qui est le seul développé et forme la base du jeu. Voici une description détaillée de l'acte 1, avec tous ses éléments, mécaniques et quêtes :

### Acte 1 : Introduction et Tutoriel (Emberwood)

L'acte 1 commence dans le village pittoresque d'Emberwood, une petite communauté nichée au milieu des ruines celtiques. Ce village sert de point de départ pour le joueur et est conçu comme une zone de tutoriel où le joueur peut se familiariser avec les différentes mécaniques du jeu. Voici les éléments clés de cette première partie :

- Choix de la Classe de Personnage : Le joueur peut choisir parmi 9 classes différentes, chacune avec ses propres compétences et styles de jeu. Les classes comprennent des guerriers, des mages, des archers, et d'autres archétypes classiques, offrant une variété de choix stratégiques.

#### Apprentissage des Mécaniques du Jeu :

- Maniement des Armes : Le joueur apprend à manier différentes armes comme les haches, épées, et arcs. Chaque arme a ses propres avantages et inconvénients, et le choix de l'arme dépend de la classe choisie.
- Équipements par Classe : Le joueur découvre les différents types d'armures (lourdes, légères) adaptées à chaque classe, influençant la mobilité et la protection.
- Éléments : Le jeu introduit les éléments (feu, eau, vent), chacun ayant des effets spécifiques en combat et sur l'environnement.
- Consommables : Le joueur apprend à utiliser des potions de régénération, de défense, et d'autres consommables pour survivre et se renforcer.
- Commerce en Ville : Interactions avec les forgerons, marchands, et autres commerçants pour acheter et améliorer l'équipement.
- Mouvements du Personnage : Le joueur apprend à se déplacer, sauter, courir, et interagir avec l'environnement.

#### Combat Tutoriel :

- À la fin du tutoriel, le joueur participe à un combat d'entraînement où il met en

pratique la gestion des mouvements, l'utilisation des compétences, des pouvoirs élémentaires, des armes, et des consommables. Ce combat sert de test final pour les compétences acquises.

#### **Lore et Contexte :**

- En plus des mécanismes de jeu, cette première partie introduit les bases du lore de *Lands of Azerith*, plaçant le joueur dans un monde riche en histoire et en mystères à découvrir.

### **1.3 Notre entreprise**

La prestigieuse Stonks Industries, fondée en 2023 par un groupe de cinq étudiants, s'est érigée en un pilier de l'industrie du jeu vidéo en l'espace d'un mois seulement. Animée par une passion débordante pour l'innovation et la créativité, l'entreprise a su imposer sa vision novatrice. Guidée par l'audace de ses fondateurs, elle s'est donnée pour noble dessein de réinventer l'expérience ludique à travers des récits immersifs, des visuels à couper le souffle et un gameplay captivant, incarné notamment par leur projet phare, *Lands of Azerith*.

Issue d'une humble salle de l'EPITA, les origines modestes de la Stonks Industries témoignent de sa volonté farouche de repousser les frontières de l'imaginaire et de l'innovation. Implantée en plein cœur de la capitale, Paris, l'entreprise est saluée pour son engagement à repousser les limites de la technologie et du gameplay afin de créer des expériences immersives et captivantes qui transcendent les générations. À sa tête, Martin Pasquier, en qualité de directeur général, insuffle à l'entreprise une vision visionnaire et un leadership éclairé.

### **1.4 L'équipe**

Avant de poursuivre, permettez-nous de vous présenter les personnalités fascinantes qui insufflent vie et passion à la Stonks Industries. Chacun d'eux apporte une expertise unique et une vision singulière au sein de notre équipe. Leurs contributions exceptionnelles ont façonné l'identité de notre entreprise et ont contribué à notre succès fulgurant dans l'industrie du jeu vidéo. Découvrez les portraits et les parcours inspirants de ces figures emblématiques qui font de la Stonks Industries un foyer d'innovation et de créativité.

#### **1.4.1 Ayemane Bouarbi**

En tant que lead game artist, Ayemane, étudiant de génie à EPITA, à 18 ans seulement, se voit confier la lourde responsabilité de piloter la direction artistique du projet de jeu vidéo ambitieux *Lands of Azerith*. Et ce choix n'a pas été réalisé sans raison : Ayemane est un artiste du jeu vidéo ayant construit sa propre réputation avec

ses plus de 10 années d'expérience dans l'industrie. En effet, passionné par les jeux vidéo et de dessin depuis l'enfance, il commence à créer des jeux vidéo sur scratch dès l'âge de 7 ans avant de commencer à en faire sur python, et plus tard, en C++. Il a également de l'expérience dans la création de concepts art, la modélisation 3D, le texturing, le shading et le lighting. Dans son parcours professionnel, il a participé à la direction artistique de plusieurs jeux vidéos, que cela soit avec de petits studios indépendants ou bien comme sur certains jeux classés AAA. Dans son rôle actuel de Lead Game Artist chez les Studios Stonks Industries, il est responsable de la gestion et de la vision artistique du jeu, ainsi que de la supervision de la création des assets et des textures du jeu.

### 1.4.2 Louise Fussien

Louise est fascinée par le numérique qui pour elle, est la fusion entre la science et l'art, entre la logique et l'intuition. C'est un domaine qui requiert à la fois des compétences techniques pointues, mais aussi une grande créativité. Louise s'est donc tout naturellement tournée vers des spécialités scientifiques au lycée et en particulier a eu l'occasion de concevoir des jeux vidéo et de s'initier à la programmation avant d'intégrer EPITA. En dehors de son travail, Louise pratique le karaté depuis l'âge de 8 ans en club et lors de nombreuses compétitions de combat. Cet art martial lui a appris l'exigence, la gestion de la pression et le travail en équipe, qualités essentielles pour travailler au sein de la Stonks Industries. Louise parle, en plus de l'anglais, l'espagnol et le russe. Elle n'en délaisse pas moins sa langue maternelle, le français et a un fort attrait pour la littérature française dont la richesse et la beauté des textes ont développé une part de sa créativité.

### 1.4.3 Michaël Museux

Michaël Museux, âgé de 17 ans, est étudiant à l'EPITA en première année de classe préparatoire intégrée. Depuis un très jeune âge, il est passionné par l'informatique, et plus particulièrement par la programmation. Après avoir découvert Scratch, il a par exemple écrit des programmes pour résoudre des calculs complexes en mathématique, parfois même impossibles à faire à la main. Michaël est aussi passionné par l'algorithmie, son excellent niveau dans ce domaine lui a même permis de participer à des concours d'algorithmie, comme le concours Castor ou Algorea où il s'est classé parmi les meilleurs, alors qu'il n'était qu'au lycée. Attriré par l'idée de se faire ses propres outils, il s'est aussi intéressé à l'intelligence artificielle (IA) et notamment les technologies d'automatisation. En effet, à une époque où la popularité des IAs grandissait de plus en plus, il semblait évident pour lui de s'y intéresser, et peut-être même d'en faire son métier. Par évidence, Michaël intègre l'équipe de développement de Stonks Industries, qui commence le développement d'un nouveau jeu : *Lands of Azerith*. Son rôle dans l'équipe est donc tout trouvé, il sera le développeur principal du jeu, avec un rôle important dans l'intelligence artificielle du jeu. De plus, ses quelques compétences en développement web lui permettent

de participer au développement du site web de l'entreprise.

#### 1.4.4 Martin Pasquier

Originaire de Normandie, jeune provincial déraciné, Martin Pasquier est le directeur technique du projet, à l'âge de 17 ans, il a déjà une bonne expérience dans le domaine de la programmation et du développement web. En tant que directeur technique, il est responsable de la gestion de l'équipe de développement et de la planification des tâches. Il est également responsable de la conception et de la mise en oeuvre du site web. Sa passion pour la programmation ayant commencé à un age où le jeu vidéo était très présent dans sa vie. Il semble évident que Martin a toujours voulu concevoir son propre jeu vidéo. De par son attrait pour la culture médiévale fantastique, il assistera sans problème le game designer en prenant part à la conception du scenario et de l'histoire du jeu. Martin est également un grand fan de musique, il aime écouter tous types de musique, ce qui lui sera utile pour la partie sonore du jeu. La diversité de ses goûts musicaux lui permettra de créer une bande sonore riche et variée, qui s'adaptera parfaitement à l'ambiance du jeu.

### 1.5 Changements depuis la dernière soutenance

Depuis la dernière soutenance, plusieurs changements significatifs ont eu lieu au sein de notre équipe et dans le développement du projet. Tout d'abord, nous avons dû faire face au départ d'Alexandre Colsch, qui a quitté EPITA pour des raisons personnelles. Ce départ a été compensé par l'arrivée de Louise Fussien, dont les compétences en programmation et design ont été précieuses pour la progression du projet.

Ce changement fait suite à une autre transition importante : Mohamed Aziz ben Amor, un membre clé de l'équipe, était parti juste avant notre première soutenance. La dynamique de notre équipe a donc été profondément impactée, nécessitant une adaptation rapide et une redistribution des responsabilités.

Malgré ces bouleversements, nous avons réussi à maintenir un rythme de travail soutenu et à progresser de manière significative.

### 1.6 Annonce de plan

Dans ce cahier des charges, nous allons vous présenter un compte rendu détaillé de l'avancement de notre projet, en décrivant les différentes étapes de son développement, les obstacles rencontrés et les solutions mises en place. Nous aborderons également les aspects organisationnels et techniques qui ont structuré notre travail.

Tout d'abord, nous discuterons des deadlines et de l'organisation. Nous commencerons par une analyse des objectifs initiaux et des progrès réalisés jusqu'à présent. Ensuite, nous ferons un point sur l'état actuel du projet. Nous expliquerons également les délais que nous avons rencontrés et les raisons pour lesquelles ils ont été nécessaires.

Ensuite, nous examinerons l'organisation du groupe. Nous décrirons les méthodes de communication utilisées au sein de l'équipe pour garantir une collaboration efficace. Nous présenterons également la répartition des tâches entre les différents membres de l'équipe, et comment nous avons utilisé GitHub pour la gestion du code source et le versionnage.

Nous aborderons ensuite le nouveau cahier technique, où nous fournirons une description détaillée des aspects techniques du projet. Cela inclura des sections spécifiques sur les différents modules et composants que nous avons développés.

Pour le design, nous partagerons nos sources d'inspiration et expliquerons le processus de création et d'implémentation de l'interface et de l'expérience utilisateur. Nous traiterons également de l'intégration du son et d'autres éléments multimédias.

Enfin, nous discuterons du site web du projet. Nous expliquerons l'objectif du site web, son développement et son apparence finale. Nous aborderons également la question du responsive design et de l'optimisation pour les appareils mobiles.

La conclusion fournira un récapitulatif de tout ce que nous avons accompli, ainsi que des remerciements aux personnes et aux ressources qui nous ont soutenus tout au long de ce projet.

Nous inclurons également des annexes pour fournir des informations supplémentaires et les documents pertinents pour une compréhension complète du projet.

Ce document vise à fournir une vue d'ensemble complète et détaillée de notre travail, en mettant en lumière notre méthodologie, nos défis, et les solutions que nous avons trouvées pour les surmonter.

## 2 Gestión des échéances et organisations

### 2.1 Analyse et progrès

#### 2.1.1 Status actuel

Depuis le début de notre projet, nous avons effectué une analyse approfondie de nos objectifs et des tâches à accomplir. L'acte 1 a été entièrement créé avec succès, ce qui représente une étape majeure dans l'avancement de notre projet.

#### 2.1.2 Délais rencontrés et raisons

Toutefois, notre progression a été marquée par des défis imprévus. Deux membres clés de notre équipe ont quitté le projet, ce qui a significativement impacté notre capacité de travail. Bien que l'administration ait ajouté un nouveau membre à notre groupe, cela n'a pas suffi à compenser entièrement la perte de compétences et d'expérience des membres sortants.

Les délais que nous avons rencontrés sont principalement dus à ces départs inattendus. L'intégration du nouveau membre a pris du temps, non seulement pour qu'il se familiarise avec le projet, mais aussi pour qu'il soit pleinement opérationnel. De plus, notre ambition de perfectionnisme a joué un rôle important. Nous avons été très méticuleux dans chaque étape du développement, ce qui, bien que bénéfique pour la qualité du projet, nous a obligé à raccourcir le nombre d'actes prévus initialement.

### 2.2 Organisation du groupe

#### 2.2.1 Communication

Pour assurer une collaboration efficace au sein de notre groupe de projet, nous avons opté pour l'utilisation de plusieurs outils de communication. Principalement, nous avons utilisé Discord, une plateforme de communication instantanée, pour faciliter les échanges en temps réel entre les membres du groupe. Discord nous a permis de maintenir une communication fluide et réactive, essentielle pour coordonner nos efforts et résoudre rapidement les problèmes rencontrés pendant le développement du projet. Les fonctionnalités de chat vocal et textuel de Discord ont été particulièrement utiles pour organiser des réunions spontanées et des discussions en groupe, améliorant ainsi notre efficacité collaborative.

En complément de Discord, nous avons également profité des sessions de travail en groupe en présentiel à l'EPITA. Ces rencontres physiques nous ont offert l'opportunité de discuter en profondeur des aspects techniques et conceptuels du projet, de partager

des idées de manière plus interactive et de travailler sur des tâches spécifiques ensemble.

Cette combinaison d'interaction en ligne via Discord et en personne à l'université nous a permis de tirer parti des avantages uniques de chaque mode de communication, renforçant ainsi notre cohésion d'équipe et notre efficacité dans l'accomplissement des objectifs du projet.

### 2.2.2 GitHub

Github a joué un rôle central dans la gestion et le partage de notre code source tout au long du projet. Nous avons utilisé GitHub comme plateforme principale pour héberger notre repository de projet, ce qui nous a permis de collaborer efficacement sur le code. Chaque membre du groupe a pu créer des branches de développement distinctes pour travailler sur des fonctionnalités spécifiques sans perturber la branche principale, tout en utilisant des pull requests pour fusionner les contributions et effectuer des révisions par les pairs.

L'intégration de GitHub avec notre flux de travail nous a offert plusieurs avantages. Premièrement, cela a facilité la gestion des versions du code et le suivi des modifications apportées à mesure que le projet évoluait. Deuxièmement, GitHub a facilité la détection et la résolution des conflits de fusion, assurant ainsi l'intégrité du code source partagé entre les membres du groupe.

### 2.2.3 Répartition des tâches

La répartition des charges au sein de notre équipe a été stratégique et s'est adaptée aux défis que nous avons rencontrés tout au long du projet. Initialement, chaque membre de l'équipe s'est vu attribuer des responsabilités spécifiques en fonction de ses compétences et de son expertise, telles que définies dans le cahier des charges technique initial.

Cependant, notre progression a été marquée par des changements significatifs. Nous avons dû faire face au départ inattendu de deux membres clés de l'équipe. Cela a eu un impact direct sur notre capacité à respecter les délais et à maintenir la cohésion au sein du groupe. Pour pallier ces départs, sans recruter de nouvelles personnes, l'administration a intégré un nouveau membre existant dans notre groupe.

Cette transition n'a pas été sans défis. L'intégration du nouveau membre a demandé du temps pour s'adapter aux dynamiques de l'équipe et pour assimiler les responsabilités en cours. Malgré ces ajustements, nous avons maintenu une communication ouverte et

des réunions régulières pour aligner les attentes et assurer une collaboration harmonieuse.

Au fur et à mesure que nous nous sommes adaptés à ces changements, nous avons également mis à jour notre cahier des charges technique pour refléter les nouvelles répartitions et les ajustements nécessaires. Chaque nouvelle itération du cahier technique a intégré les modifications stratégiques et les décisions prises en réponse aux défis rencontrés, y compris la redistribution des tâches et des responsabilités entre les membres existants.

#### 2.2.4 Cahier des charges techniques

Pendant le développement du projet, nous avons élaboré plusieurs versions du cahier technique pour répondre aux évolutions et aux défis rencontrés. Chaque itération a été cruciale pour ajuster notre approche et nos spécifications en fonction des besoins changeants du projet.

Initialement, notre cahier technique définissait les fondations et les objectifs techniques de notre jeu. Nous avons spécifié les fonctionnalités essentielles, l'architecture logicielle prévue, ainsi que les technologies clés à utiliser. Cette première version nous a permis de commencer le développement sur des bases solides.

Les deux prochaines pages sont dédiées au cahier des charges technique. Ce document détaille les contraintes et les caractéristiques techniques nécessaires pour répondre aux besoins du projet. Il synthétise toutes les réponses aux questions techniques que l'on peut se poser sur le projet.

Le cahier des charges technique est divisé en deux parties. La première partie détaille les spécificités du jeu et les choix techniques qui ont été réalisés. La deuxième partie donne des informations sur la répartition des tâches et leur avancement.

Des changements ont été apportés au cahier des charges technique depuis sa dernière version. Le nom du jeu a été modifié, le départ de Mohamed Aziz et Alexandre a été pris en compte, et des changements ont été apportés au diagramme de Gantt pour refléter l'avancé du projet.

Le dernier cahier des charges technique, intégré dans ce rapport final, reflète ces ajustements. Il décrit non seulement les spécifications finales du jeu telles qu'elles ont évolué tout au long du processus, mais aussi les décisions stratégiques prises pour surmonter les défis rencontrés.

Nom du groupe :	Stonks Industries		
Nom du projet :	Lands of Azerith		

Noms des membre :				
Nom :	Prénom :	Login :	Classe :	
Ben Amor	Mohamed Aziz	mohamed.aziz.ben.amor	B1	
Pasquier	Martin	martin.pasquier	B1	
Cölsch	Alexandre	alexandre.colsch	B1	
Bouarbi	Ayemane	ayemane.bouarbi	B1	
Museux	Michaël	michael.museux	B1	

Type de jeu :				
Action/Aventure	Battle Royale	Beat them all	Combat	Simulation
FPS	MMORPG	MOBA	Party Games	Survival Horror
Plateforme	Puzzles	Reflexion	Rogue Like	TPS
RPG	RTS	Sandbox	Shoot them up	Course
Autre :				

Caractéristiques générales du jeu :				
IA :	Errer	Attaquer	S'échapper	"Path Finder"
Multijoueurs :	Coopé	Battle (2-4)	Massif	
Réseau :	P2P	Lan	Online	

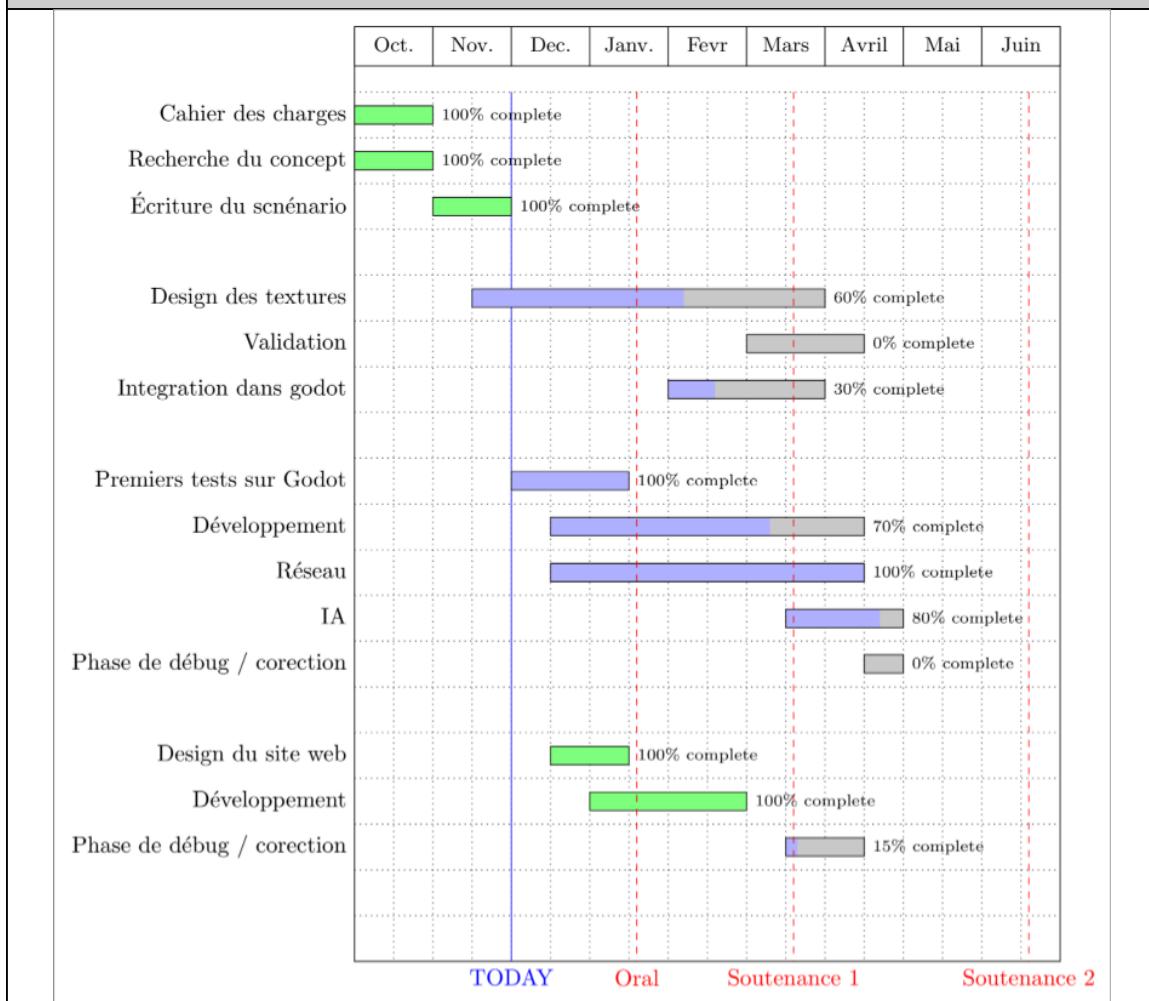
Caractéristiques graphiques :				
Dimension :	2D	3D	Autres :	
Particularités :	Stéréoscopie	AR	VR	
graphiques :	Perso	Custom	Existant	
Précisions :				

Caractéristiques sonores :				
Musique :	Perso	Custom	Existant	
FX :	Perso	Custom	Existant	
Précisions :				

Autres caractéristiques :				
Site Web :	Perso	Custom	Préfabriqué	

Répartition des tâches : deux personnes par tâche: (R)esponsable & (S)uppléant					
Tâches	mohamed-aziz.be n-amor	ayemane.bouarbi	alexandre.colsch	michael.museux	martin.pasquier
Ecriture de l'histoire		S	R		
IA				R	S
Site web				S	R
Système d'équipement			R	S	
Design personnages / terrains / objets		R	S		
Design de la carte et intégration dans Godot		R	S		
Musiques et effets sonores			S		R
Intégration textures dans Godot		S		R	
Multijoueur				R	S

Tableau d'avancement des tâches



# 3 Partie technique

## 3.1 Environnement de travail

### 3.1.1 Introduction à l'Environnement de Travail

Dans le cadre de notre projet, nous avons choisi d'utiliser le moteur de jeu Godot en conjonction avec l'environnement de développement intégré (IDE) Rider. Cette configuration a été sélectionnée pour ses capacités avancées en matière de développement de jeux vidéo, ainsi que pour la flexibilité et les outils de débogage puissants offerts par Rider.

### 3.1.2 Fonctionnement de Godot et Intégration avec Rider

**Godot** est un moteur de jeu open-source, reconnu pour sa facilité d'utilisation et sa capacité à gérer des projets de jeux vidéo en 2D et en 3D. Il offre une gamme complète de fonctionnalités, y compris un éditeur visuel intuitif, des systèmes de scripting flexibles grâce à GDScript (un langage de script similaire à Python), ou C# et des outils robustes pour l'animation et la physique. Le choix de Godot a été motivé par sa communauté active et ses ressources abondantes, facilitant ainsi l'apprentissage et la résolution de problèmes.

**Rider**, développé par JetBrains, est un IDE multiplateforme qui offre un support étendu pour de nombreux langages de programmation, notamment C#. Son intégration avec Godot permet d'exploiter pleinement les capacités de scripting C# de Godot, offrant ainsi une expérience de développement plus enrichissante et productive. Rider propose également des outils de refactoring, une navigation améliorée dans le code, et des capacités de débogage avancées, qui sont particulièrement utiles pour les projets complexes de jeux vidéo.

### 3.1.3 Défis Associés aux Changements de version de Godot

L'un des défis majeurs rencontrés dans notre environnement de travail est lié aux changements de version de Godot. Le moteur est en constante évolution, avec des mises à jour fréquentes qui apportent de nouvelles fonctionnalités et des améliorations de performance. Cependant, ces mises à jour peuvent également introduire des modifications significatives qui affectent la compatibilité du code existant.

Pour une partie de notre groupe, moins expérimentée dans la conception de jeux vidéo, ces changements de version ont posé des problèmes d'adaptation. Les différences entre les versions peuvent inclure des modifications dans l'API, des changements de comportement dans les outils d'animation ou de physique, et des ajustements dans les méthodes de rendu graphique. Ces modifications nécessitent une compréhension

approfondie des nouvelles fonctionnalités et des ajustements potentiellement complexes du code existant pour maintenir la stabilité et les performances du projet.

Afin de surmonter ces défis, nous avons mis en place plusieurs stratégies :

- Documentation et Formation : Nous avons organisé des sessions de formation interne pour familiariser l'équipe avec les nouvelles fonctionnalités et les changements apportés par les mises à jour de Godot. Des tutoriels et des guides spécifiques ont été créés pour faciliter l'apprentissage.
- Environnement de Test : Avant de migrer vers une nouvelle version de Godot, nous avons établi un environnement de test dédié pour évaluer l'impact des changements sur notre projet. Cela nous permet de repérer et de corriger les incompatibilités potentielles avant d'intégrer les modifications dans la branche principale du projet.
- Révision de Code et Pair Programming : Nous avons encouragé la révision de code et le pair programming, où des membres plus expérimentés de l'équipe travaillent en étroite collaboration avec ceux moins expérimentés, pour partager des connaissances et assurer une transition en douceur entre les versions.

## 3.2 Manipulation des Fichiers JSON : Défis et Solutions

### 3.2.1 Introduction à JSON et son Utilisation dans le Développement de Jeux

JavaScript Object Notation (JSON) est un format de données léger et facile à lire et écrire pour les humains, et facile à analyser et générer pour les machines. JSON est largement utilisé dans le développement de jeux vidéo pour stocker et échanger des données structurées. Dans notre projet, nous avons utilisé des fichiers JSON pour stocker divers éléments du jeu, y compris des images de texte, des configurations de jeu, et des données de sauvegarde.

### 3.2.2 Compréhension et Manipulation des Fichiers JSON

Lors de notre première tentative d'intégration de fichiers JSON dans notre projet de jeu vidéo, nous avons rencontré plusieurs difficultés en raison de notre manque initial de familiarité avec ce format de fichier. Pour remédier à cela, nous avons entrepris une étude approfondie de JSON et de ses applications dans le contexte de notre projet.

### 3.2.3 Structure et Syntaxe de JSON

JSON représente des données sous forme de paires clé/valeur et de tableaux. Une paire clé/valeur se compose d'un nom de champ (clé) et de sa valeur associée. Les valeurs peuvent être des chaînes de caractères, des nombres, des objets (des paires clé/valeur imbriquées), des tableaux, des booléens (true/false), ou null.

```
{  
  "menu": {  
    "id": "file",  
    "value": "File",  
    "popup": {  
      "menuitem": [  
        { "value": "New", "onclick": "CreateNewDoc()" },  
        { "value": "Open", "onclick": "OpenDoc()" },  
        { "value": "Close", "onclick": "CloseDoc()" }  
      ]  
    }  
  }  
}
```

FIGURE 1 – Exemple de structure JSON

L'intégration de fichiers JSON dans notre code s'est révélée être un défi considérable mais stimulant. Voici un aperçu détaillé du processus et des solutions mises en œuvre pour surmonter les obstacles techniques :

#### A. Lecture et Écriture de Fichiers JSON :

La première étape consistait à lire les données à partir des fichiers JSON et à les écrire dans ces fichiers. Nous avons utilisé les bibliothèques de gestion des fichiers JSON disponibles dans les langages de programmation utilisés dans notre projet. En C#, par exemple, nous avons exploité les classes JsonSerializer et JsonDeserializer fournies par le framework .NET.

#### B. Sélection des Attributs à Sauvegarder :

Une des tâches cruciales était de déterminer quels attributs des objets de jeu devaient être sauvegardés dans les fichiers JSON. Pour cela, nous avons effectué une analyse approfondie des besoins du jeu et de la pertinence des données à long terme. Les attributs essentiels comme les statistiques des personnages, l'état de l'inventaire, et les positions des objets dans le monde de jeu ont été jugés importants. En revanche, les attributs temporaires ou dérivables, comme les états intermédiaires de certaines animations, ont été exclus pour éviter un gonflement inutile des fichiers de sauvegarde.

## C. Récupération des Objets à Partir de Fichiers JSON :

Une fois les fichiers JSON créés, la prochaine étape consistait à récupérer les objets de jeu à partir de ces fichiers. Ce processus impliquait de déserialiser les données JSON en objets utilisables dans notre moteur de jeu. L'un des défis majeurs ici était de garantir que les données récupérées soient correctement mappées aux structures de données de notre jeu, en respectant les types et les contraintes de chaque attribut.

## 3.3 L'architecture du jeu

L'architecture d'un projet de développement de jeux vidéo est essentielle pour assurer une organisation claire et une gestion efficace des ressources. Dans notre projet, nous avons adopté une structure de dossiers bien définie pour faciliter la gestion des différents composants du jeu. Cette organisation a permis de simplifier le processus de développement, de faciliter la collaboration entre les membres de l'équipe et d'assurer une maintenance efficace du code et des ressources. Voici une description détaillée de la structure de notre projet et du rôle de chaque dossier.

### 3.3.1 Structure des Dossiers

#### 1. Dossier Assets

Le dossier ‘assets’ est le répertoire principal qui contient toutes les ressources visuelles et sonores utilisées dans le jeu. Ce dossier est crucial pour centraliser toutes les images, les sons, les textures et autres médias nécessaires pour le développement du jeu.

- Contenu : Images, textures, sons, et autres médias.
- Exemple : ‘assets/background.png’, ‘assets/music/theme.mp3’.

#### 2. Dossier Caractères

Le dossier ‘caractères’ est dédié aux images des personnages du jeu, incluant à la fois le joueur et les monstres. Cette séparation permet une gestion plus facile et une localisation rapide des ressources graphiques liées aux entités interactives du jeu.

- Contenu : Images des personnages joueurs et des monstres.
- Exemple : ‘caractères/joueur.png’, ‘caractères/monstre.png’.

#### 3. Dossier Items

Le dossier ‘items’ contient les fichiers JSON associés aux différents objets (items) présents dans le jeu. Ces fichiers JSON définissent les attributs et les propriétés des objets, tels que leur nom, description, valeur, et effets en jeu.

- Contenu : Fichiers JSON des objets.
- Exemple : ‘items/potion.json’, ‘items/épée.json’.

#### 4. Dossier Loottables

Le dossier ‘loottables’ est associé aux données d’objets donnés par la mort des monstres. Ces fichiers permettent de générer les objets qui seront donnés par les monstres à leur mort.

- Contenu : Fichiers JSON pour les objets des monstres.

## 5. Dossier Mob

Le dossier ‘mob’ contient les fichiers JSON des monstres, décrivant leurs caractéristiques, comportements et statistiques. Ces fichiers jouent un rôle crucial dans la définition des ennemis que les joueurs rencontreront dans le jeu.

- Contenu : Fichiers JSON des monstres.
- Exemple : ‘mob/zombie.json’, ‘mob/dragon.json’.

## 6. Dossier Quest

Le dossier ‘quest’ contient les fichiers JSON associés aux quêtes du jeu. Chaque fichier JSON représente une quête, incluant des informations telles que les objectifs, les récompenses, et les dialogues associés.

- Contenu : Fichiers JSON des quêtes.
- Exemple : ‘quest/quête1.json’, ‘quest/quête2.json’.

## 7. Dossier Scenes

Le dossier ‘scenes’ contient toutes les scènes utilisées dans le moteur Godot. Une scène dans Godot représente une collection d’objets (nodes) organisés de manière hiérarchique. Ce dossier est essentiel pour structurer les différents niveaux et interfaces du jeu.

- Contenu : Fichiers de scènes Godot.
- Exemple : ‘scenes/menu.tscn’, ‘scenes/niveau1.tscn’.

## 8. Dossier Script

Le dossier ‘script’ contient tous les scripts utilisés dans le projet. Ces scripts sont écrits en C#, et sont responsables de la logique de jeu, des interactions des personnages, et des comportements des objets.

- Contenu : Scripts de jeu.

## 3.4 Menu du Jeu



FIGURE 2 – Menu principal du jeu

### 3.4.1 Introduction à l’Effet Parallax dans le Menu

L’effet parallax est une technique visuelle utilisée pour créer une sensation de profondeur en animant différents plans à des vitesses différentes. Dans notre jeu, nous avons implémenté un effet parallax dans le menu principal pour donner une impression de perspective en 2D. Voici comment nous avons structuré et mis en œuvre cet effet de manière efficace et immersive.

### 3.4.2 Implémentation de l’Effet Parallax

Nous avons créé une node principale `ParallaxBackground` dans Godot, qui agit comme le conteneur global pour notre effet parallax. Cette node a des enfants (fils) correspondant à chaque plan de fond que nous voulons animer. Chaque plan a une image différente et une vitesse de défilement spécifique, ce qui crée une illusion de profondeur lorsque le joueur navigue dans le menu.

### 3.4.3 Structure de la Node ParallaxBackground

La node `ParallaxBackground` est configurée avec plusieurs enfants (fils), chacun représentant un plan de fond distinct. Chaque plan est un sprite ou une texture avec une image unique et est déplacé à une vitesse différente par rapport aux autres plans pour simuler la perspective.

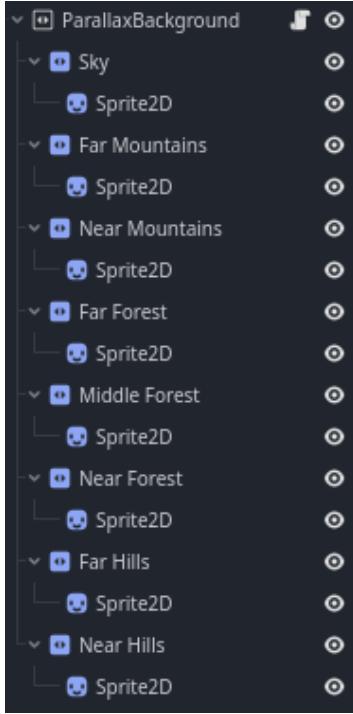


FIGURE 3 – Exemple de structure de la node ParallaxBackground

### 3.4.4 Animation et Effet Visuel

Chaque plan de fond est animé en ajustant sa position horizontale à chaque trame (frame) du jeu, en fonction de sa vitesse définie. Les plans plus proches (vitesse plus faible) se déplacent plus lentement par rapport à ceux qui sont plus éloignés (vitesse plus élevée), créant ainsi un effet de parallaxe réaliste qui renforce l'immersion du joueur dès le lancement du jeu.

## 3.5 Le Multijoueur

Pour pouvoir organiser une partie multijoueur, nous avons décidé d'utiliser l'API intégrée à Godot. Pour cela, il nous faut un serveur, ici l'hôte, ainsi que les joueurs qui rejoignent sa partie appelés les clients. En premier temps, l'hôte doit appuyer sur le bouton 'Host' puis les joueurs doivent accéder à l'adresse de l'hôte, et appuyer sur le bouton 'Join'.

Il est possible de jouer avec un nombre infini de joueurs, mais nous avons établi un maximum de 8, limite que nous utiliserons lors de la conception du reste du jeu.

Maintenant que la connexion est établie entre les différents joueurs, il nous faut synchroniser leurs informations entre eux, tout en faisant attention à ne pas trop

partager d'informations inutiles, pour limiter l'utilisation de bande passante. Par exemple, la synchronisation des positions de chacun entre tous les joueurs est importante, mais la position de la caméra d'un joueur n'est nécessaire qu'en local.

### 3.5.1 Système de Connexion Multi-joueurs via Envoi de Paquets

Pour permettre une connectivité multi-joueurs fluide et intuitive, nous avons implémenté un système de gestion de serveur et de connexion basé sur l'envoi de paquets. Ce système est conçu pour simplifier l'expérience des joueurs en éliminant la nécessité d'entrer manuellement les adresses IP des serveurs.

### 3.5.2 Fonctionnement du Système

#### A. Serveur d'Hébergement :

Lorsqu'un joueur décide d'héberger une partie, le jeu démarre un serveur local et commence à envoyer des paquets d'informations à intervalles réguliers (par exemple, toutes les secondes). Chaque paquet contient des détails essentiels comme le nom du serveur, l'adresse IP du serveur, et le nombre actuel de joueurs connectés.

#### B. Joueurs Rejoignant le Serveur :

Les autres joueurs du réseau écoutent en permanence les paquets émis par les serveurs disponibles. Lorsqu'un paquet est détecté, les détails du serveur sont affichés à l'écran, permettant aux joueurs de visualiser facilement les serveurs disponibles ainsi que les informations pertinentes comme le nom et le nombre de joueurs. Un bouton "Rejoindre" est proposé aux joueurs pour se connecter au serveur sélectionné, simplifiant ainsi le processus de connexion.

#### C. Démarrage de la Partie :

Une fois que les joueurs se sont connectés au serveur et que le nombre requis est atteint, un bouton "Start" devient accessible à tous les participants. Lorsque tous les joueurs ont cliqué sur "Start", la partie commence, permettant ainsi de synchroniser l'expérience de jeu pour tous les participants.

### 3.5.3 Avantages du Système de Paquets

Ce système de connexion basé sur l'envoi de paquets présente plusieurs avantages :

- Simplicité d'utilisation : Les joueurs n'ont pas besoin de connaître l'adresse IP du serveur manuellement.

- Visibilité et Accessibilité : Les informations sur les serveurs disponibles sont clairement affichées, facilitant ainsi la sélection et la connexion.
- Synchronisation Facile : Tous les joueurs sont automatiquement informés des détails du jeu en cours, assurant ainsi une expérience de jeu fluide et harmonieuse.

## 3.6 Contrôles et Interaction dans le Jeu

Les contrôles et interactions jouent un rôle crucial dans l'expérience de jeu, assurant une navigation fluide et intuitive à travers l'environnement virtuel. Dans notre jeu développé sous Godot, nous avons mis en œuvre plusieurs fonctionnalités de contrôle permettant aux joueurs de se déplacer d'interagir avec l'environnement, de gérer leur inventaire et de lancer des actions spécifiques avec facilité.

Les déplacements du joueur sont gérés à l'aide des touches Z, Q, S, et D, qui correspondent respectivement aux directions Haut, Gauche, Bas, et Droite sur le clavier. Cette configuration permet une navigation fluide dans les différents niveaux du jeu. L'utilisation de la librairie d'entrée de Godot facilite la détection des touches pressées et assure une réponse instantanée aux actions du joueur.

### 3.6.1 Actions Spécifiques

#### A. Quitter le Jeu avec la Touche Échap

Pour offrir une sortie rapide et intuitive du jeu, la touche Echap est associée à la fonction de fermeture de l'application. Cela permet aux joueurs de quitter le jeu à tout moment sans effort, tout en libérant proprement toutes les ressources utilisées par le jeu.

#### B. Courir avec la Touche Shift

La touche Majuscule est utilisée pour activer le mode de course du joueur, augmentant ainsi sa vitesse de déplacement. Ce mécanisme permet aux joueurs de se déplacer plus rapidement lorsqu'ils ont besoin d'atteindre des destinations plus éloignées ou de fuir des dangers potentiels dans le jeu.

#### C. Ouvrir l'Inventaire avec la Touche Tab

L'inventaire est une composante essentielle de nombreux jeux, offrant aux joueurs la possibilité de gérer leurs objets, équipements et ressources. Dans notre jeu, l'inventaire est accessible en appuyant sur la touche Tab, facilitant ainsi la gestion des items collectés et des équipements utilisés par le joueur.

#### D. Attaquer avec le Clic Gauche de la Souris

Pour interagir avec les monstres et autres éléments interactifs dans le jeu, les joueurs

peuvent utiliser le clic gauche de la souris. Ce mécanisme permet de lancer des attaques, d'activer des objets ou d'interagir avec des éléments du décor, offrant ainsi une expérience immersive et dynamique.

### E. Implémentation Technique

Chaque fonctionnalité de contrôle est implémentée en utilisant les capacités intégrées de Godot pour gérer les événements d'entrée et les interactions utilisateur. Cette approche assure une cohérence dans le traitement des actions du joueur et garantit une jouabilité fluide et intuitive tout au long du jeu.

## 3.7 Intelligence Artificielle : Comportements des Monstres

Dans notre jeu, nous avons implémenté différents comportements pour les monstres afin d'enrichir l'interaction avec le joueur. Chaque type de monstre possède des caractéristiques et des réactions spécifiques, déterminées par une classe dédiée qui gère son comportement en fonction des interactions avec le joueur.

### 3.7.1 Types de Monstres

Nous avons défini quatre types de monstres, chacun ayant un comportement distinct en fonction de la proximité et des actions du joueur :

- Agressif : Ce type de monstre attaque activement le joueur dès qu'il est à proximité. Lorsque le joueur entre dans sa zone de détection, le monstre commence à se déplacer vers le joueur pour l'attaquer. En cas de contact avec le joueur, le monstre déclenche une attaque ou une réaction offensive.
- Neutre : Les monstres neutres ne sont pas agressifs par nature mais répondent si le joueur les attaque. Lorsque le joueur les agresse, ils changent leur comportement pour devenir agressifs et attaquent en retour. En dehors des attaques contre eux, ils continuent leurs activités normales sans réaction offensive.
- Passif : Les monstres passifs ne cherchent pas à attaquer le joueur et fuient s'il est agressé. Lorsque le joueur les attaque, ils s'éloignent activement du joueur pour éviter tout contact et toute confrontation directe.
- Peureux : Les monstres peureux évitent activement tout contact avec le joueur. Dès que le joueur s'approche, ils tentent de s'éloigner rapidement et de fuir la zone de danger.

### **3.7.2 Implémentation Technique**

Chaque type de monstre est implémenté à l'aide de classes distinctes dans Godot, chacune étant responsable de la gestion des comportements spécifiques décrits ci-dessus. Voici comment nous avons mis en œuvre ces comportements complexes à l'aide des capacités de détection de collisions et de signaux de Godot :

#### **A. Détection de Collision avec CollisionShape2D**

Chaque monstre est équipé d'une CollisionShape2D dans Godot, configurée pour détecter les collisions avec le joueur et d'autres entités du jeu. Cette forme de collision est essentielle pour déclencher des actions basées sur la proximité et les interactions spatiales avec d'autres nodes dans la scène.

#### **B. Utilisation de Area2D pour les Signaux**

Nous utilisons également des nodes Area2D pour émettre des signaux vers la node principale du monstre. Ces signaux sont utilisés pour détecter quand le joueur entre dans la zone de détection du monstre, ce qui déclenche un changement de comportement en fonction du type de monstre :

Pour les monstres agressifs et neutres : lorsque le joueur entre dans la zone, le monstre commence à se déplacer vers le joueur pour l'attaquer (si neutre, il attaque en retour si attaqué). Pour les monstres passifs et peureux : lorsque le joueur entre dans la zone, le monstre s'éloigne activement du joueur pour éviter le contact.

#### **C. Gestion des Comportements Dynamiques**

Chaque classe de monstre gère de manière dynamique son comportement en réponse aux signaux émis par Area2D.

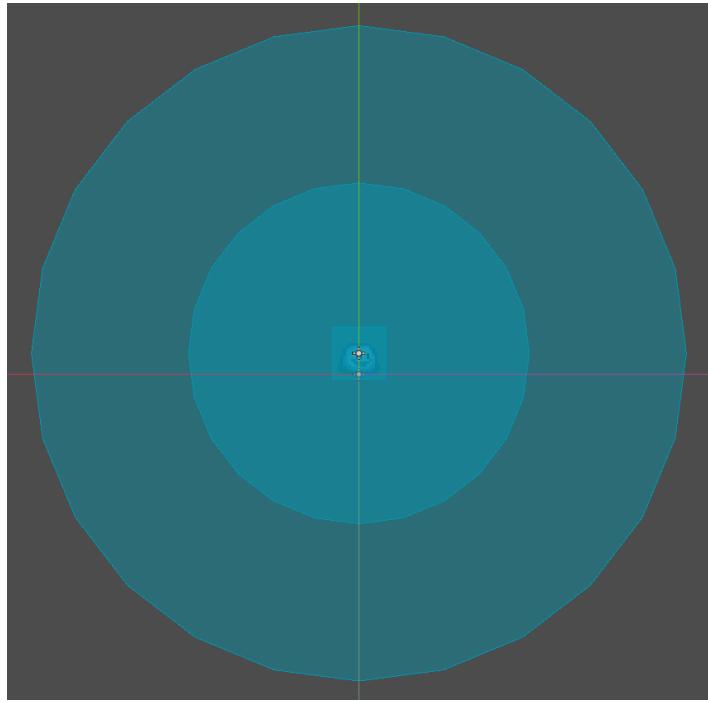


FIGURE 4 – Exemple du signal émis par le mob

## 3.8 Déplacement des Monstres avec Navigation A\*

Dans notre jeu, nous avons intégré des mécanismes avancés pour le déplacement des monstres en utilisant le système de navigation A\* (A-star) et la gestion de l'inventaire du joueur. Ces fonctionnalités améliorent l'expérience de jeu en offrant des déplacements fluides et des interactions riches avec les objets collectés. Voici comment nous avons implémenté ces fonctionnalités techniques dans Godot.

### 3.8.1 Algorithme A\*

Pour permettre aux monstres de se déplacer intelligemment dans l'environnement du jeu, nous avons utilisé la node `Navigation2D` de Godot, qui intègre un système de pathfinding basé sur l'algorithme A\*. Cela permet aux monstres de calculer automatiquement le chemin le plus optimal vers leur cible, comme le joueur ou un point spécifique de la carte.

### 3.8.2 Implémentation de la Node `Navigation2D`

La node `Navigation2D` est configurée pour représenter la carte jouable du jeu, avec des obstacles et des zones de passage définies. Voici comment elle est utilisée dans Godot :

- **Configuration de la Navigation** : Nous avons générée une carte de navigation en plaçant des obstacles et en définissant les zones accessibles par les monstres. Cette carte est utilisée par l'algorithme A\* pour calculer les chemins.
- **Utilisation de Navigation2D dans le Code** : Chaque monstre est doté d'une instance de `Navigation2D`, configurée pour calculer un chemin vers sa cible actuelle à intervalles réguliers.

### 3.8.3 Déplacement Dynamique des Monstres

À intervalles réguliers, les monstres recalculent leur chemin vers leur cible (par exemple, le joueur) en utilisant la fonctionnalité de `Navigation2D`. Cela garantit que les monstres peuvent naviguer efficacement autour des obstacles et trouver le chemin le plus court pour atteindre leur objectif.

### 3.8.4 Aléatoire dans le Rayon de Déplacement

Pour ajouter une variabilité aux déplacements des monstres, toutes les 5 secondes, chaque monstre a une chance sur trois de choisir une nouvelle destination aléatoire à l'intérieur d'un rayon de 300 pixels autour de sa position actuelle. Cela simule un comportement plus organique et imprévisible des monstres, rendant leur présence dans le jeu plus dynamique et réaliste.

### 3.8.5 Spawning des Monstres dans une Zone Définie

Pour enrichir l'exploration du joueur et maintenir la dynamique du jeu, nous avons mis en place un système de spawn de monstres dans des zones prédéfinies :

- **Zone de Spawn Rectangulaire** : Nous avons défini une zone rectangulaire dans la carte du jeu où les monstres peuvent apparaître périodiquement.
- **Instantiation des Monstres** : Toutes les 15 secondes, un nouveau monstre est instancié dans cette zone de spawn si la limite de monstres dans cette zone n'est pas atteinte.
- **Chargement Dynamique depuis un Fichier JSON** : Chaque type de monstre est associé à un fichier JSON qui contient ses caractéristiques, comme ses attributs, ses comportements et ses statistiques. Lorsqu'un monstre est instancié, ces informations sont chargées à partir du fichier JSON correspondant.

## 3.9 Gestion de l'inventaire du joueur

### 3.9.1 Structure de l'Inventaire

L'inventaire est représenté par un tableau de cases, où chaque case peut contenir un objet spécifique avec sa quantité. Les joueurs peuvent visualiser leur inventaire en appuyant sur la touche Tab, ce qui ouvre une interface utilisateur affichant les objets collectés et leur quantité respective.

### 3.9.2 Interaction avec l'Inventaire

- **Ajout d'Objets** : Lorsque le joueur ramasse un nouvel objet, celui-ci est ajouté à l'inventaire s'il n'est pas déjà présent. Si l'objet est déjà présent, la quantité existante est simplement mise à jour.
- **Affichage et Gestion** : L'interface de l'inventaire permet aux joueurs de visualiser, utiliser et organiser leurs objets selon leurs besoins. Par exemple, les objets peuvent être équipés, consommés ou échangés selon leur utilité dans le jeu.

## 3.10 Gestion des Points de Vie, Checkpoints et Loot dans le Jeu

Dans notre jeu, nous avons mis en place un système robuste pour gérer les points de vie du joueur, les checkpoints pour la réapparition, et la mécanique de loot des monstres. Voici comment chaque aspect est implémenté pour enrichir l'expérience de jeu et assurer une progression gratifiante pour les joueurs.

### 3.10.1 Points de Vie du Joueur

Le joueur démarre avec un nombre initial de points de vie (PV) fixé à 1000. Lorsqu'il subit une attaque de la part d'un monstre ou d'un autre ennemi, les dégâts infligés sont calculés en fonction de la force de l'attaque et des attributs de l'arme de l'attaquant.

### 3.10.2 Calcul des Dégâts

Les dégâts subis par le joueur sont calculés en fonction de la formule suivante, basée sur la force de l'attaque (*force\_attaque*) et le nombre de points de l'arme (*points\_arame*) de l'attaquant :  $(points\_arme) + (force\_attaque)$

Cela permet d'ajuster dynamiquement les dégâts en fonction de la puissance de l'attaque et de l'équipement de l'attaquant, ajoutant ainsi une couche de stratégie et de gestion des ressources pour le joueur.

### 3.10.3 Gestion des Checkpoints

Pour assurer une progression fluide et éviter une perte de progrès significative en cas de mort du joueur, nous avons intégré un système de checkpoints :

- **Réapparition au Dernier Checkpoint** : Lorsque le joueur meurt, il réapparaît au dernier checkpoint atteint. Cette fonctionnalité est essentielle pour réduire la frustration du joueur et encourager l'exploration sans crainte de perdre tout son progrès.
- **Restauration des Points de Vie Maximum** : À chaque réapparition, le joueur récupère ses points de vie maximum, lui permettant de reprendre le jeu avec une chance égale de succès.

### 3.10.4 Mécanique de Loot des Monstres

Lorsqu'un monstre est vaincu, plusieurs actions se produisent pour enrichir l'expérience du joueur :

- **Loot des Objets Associés au Monstre** : Chaque type de monstre est associé à un fichier JSON définissant les objets qu'il peut lâcher lorsqu'il meurt. Ces objets peuvent inclure des équipements, des ressources ou d'autres items précieux pour la progression du joueur.
- **Apparition de Débris et Textures** : Pour indiquer visuellement la mort du monstre, des débris et des textures spécifiques à ce monstre apparaissent à l'emplacement où il a été vaincu. Cela crée une immersion et une satisfaction visuelle pour le joueur.
- **Récupération des Objets par le Joueur** : Lorsque le joueur passe sur les débris et textures laissés par le monstre, il récupère automatiquement les objets associés dans son inventaire. Cela simplifie le processus de collecte et de gestion des objets obtenus lors des combats.

## 3.11 Génération de Map et Gestion des Transitions avec Nodes dans Godot

Dans notre jeu, la création et la gestion des textures de la carte ainsi que les transitions fluides entre les différents environnements sont des aspects cruciaux pour offrir une expérience visuelle immersive et dynamique. Voici comment nous avons implémenté ces fonctionnalités techniques avancées dans Godot, en nous concentrant sur la génération de la carte, les transitions entre les maps et l'animation des personnages et des objets.

## 3.12 Génération de la Map et Utilisation des Textures

La carte du jeu est entièrement générée et dessinée par notre designer, ce qui nous permet de contrôler précisément l'aspect visuel et le design de chaque environnement. Voici comment nous avons structuré notre approche :

### 3.12.1 Textures et Tiles

Pour la création de la carte, nous utilisons des textures organisées en tiles. Chaque tile représente une partie spécifique de la carte, telle que le sol et les éléments statiques qui ne bougent pas. Ces textures sont soigneusement intégrées dans la map pour former des environnements cohérents et esthétiquement plaisants.

### 3.12.2 Tileset pour les Maps

Les tilesets sont des images contenant plusieurs textures utilisées pour les maps. Ils permettent de définir les décors, les détails du sol et d'autres éléments fixes qui ne changent pas pendant le jeu. Cela garantit une cohérence visuelle et simplifie la gestion des textures dans l'environnement du jeu.



FIGURE 5 – Nos textures

## 3.13 Gestion des Transitions entre les Maps

Les transitions entre les différentes zones de la carte sont gérées à l'aide de nodes spécifiques qui assurent une expérience de jeu fluide et immersive :

### 3.13.1 Nodes de Transition

Chaque zone de transition est représentée par une node comportant un `Area2D` et une `CollisionShape2D`. Lorsque le joueur entre dans cette zone, des événements sont déclenchés pour faire apparaître la nouvelle map, faire disparaître l'ancienne et ajuster les coordonnées du joueur, créant ainsi une illusion de téléportation fluide et réaliste.

### 3.13.2 Changement Dynamique de la Map

Lorsque le joueur traverse une zone de transition, le jeu procède à un chargement dynamique de la nouvelle map tout en déchargeant celle précédente. Cela permet de maintenir une utilisation efficace des ressources tout en offrant une navigation sans heurts à travers les différentes zones de jeu.

## 3.14 Animation des Personnages et des Objets

Pour donner vie aux personnages et aux objets dans notre jeu, nous utilisons des animations basées sur des sprites animés (`AnimatedSprite2D`). Ces animations sont essentielles pour créer des mouvements fluides et réalistes :

### 3.14.1 Sprites Animés

Chaque personnage, monstre et objet interactif est doté d'un `AnimatedSprite2D` qui contient une séquence d'images. Ces images sont affichées successivement selon un rythme prédéfini, créant ainsi des animations telles que la marche, la course, l'attaque ou d'autres actions spécifiques.

### 3.14.2 Crédit de Mouvements

Les animations des personnages sont conçues pour refléter leur comportement et leurs actions dans le jeu. Par exemple, un personnage pourrait avoir une animation pour lever son arme avant de frapper, ou une animation de marche où les images des pieds gauche et droit sont alternées pour simuler le mouvement.

En intégrant ces techniques avancées de gestion de textures, de transitions de maps et d'animation dans notre jeu sous Godot, nous avons réussi à créer une expérience de jeu immersive et visuellement attrayante. La génération de maps et l'utilisation de tilesets garantissent une conception précise et esthétique des environnements, tandis que les transitions fluides et les animations dynamiques des personnages enrichissent l'expérience globale du joueur. Cette approche technique non seulement améliore la jouabilité mais aussi renforce l'engagement des joueurs envers notre univers de jeu, offrant ainsi une expérience mémorable et captivante à chaque session de jeu.

# 4 Design et musique

## 4.1 Design

### 4.1.1 Notre logo

#### **Symbolisme de la couleur jaune**

Le jaune, au cœur de notre identité visuelle, est choisi avec soin pour ses implications psychologiques profondes et ses associations scientifiquement étayées :

**Énergie et dynamisme** : Le jaune est universellement reconnu pour son pouvoir d'éveiller l'énergie, la vitalité et le dynamisme. Psychologiquement, cette couleur stimule les sens et incite à l'action. Pour une entreprise de jeux vidéo comme la nôtre, ces qualités sont cruciales pour susciter l'enthousiasme et l'engagement des joueurs, en harmonie avec leurs attentes d'expériences immersives et passionnantes.

**Optimisme et positivité** : En plus de son énergie palpable, le jaune est une couleur intrinsèquement positive, évoquant la joie, l'optimisme et une attitude lumineuse envers l'avenir. Des études montrent que cette couleur stimule la libération de sérotonine dans le cerveau, neurotransmetteur associé à la régulation de l'humeur et au bien-être. Ainsi, notre choix du jaune reflète non seulement notre dynamisme mais aussi notre désir de créer des expériences de jeu qui inspirent et réconfortent nos joueurs.

#### **Deux tons de jaune**

**Contraste et dimension** : La dualité de deux nuances de jaune dans notre logo n'est pas uniquement esthétique mais aussi stratégique sur le plan psychologique. Ce contraste subtil mais significatif crée une profondeur visuelle qui attire l'œil et prolonge l'intérêt. La recherche en psychologie de la couleur suggère que l'utilisation de multiples nuances dans une même palette stimule la cognition visuelle, augmentant ainsi la mémorabilité et l'impact du logo.

**Accentuation** : Le jeu de nuances dans notre logo n'est pas seulement pour l'esthétique, mais aussi pour guider l'attention et renforcer la reconnaissance. Le contraste entre les tons de jaune accentue des éléments clés comme le nom de l'entreprise et les motifs iconographiques, améliorant ainsi la lisibilité et la perception du logo dans divers contextes visuels.

#### **Pourquoi un éclair ?**

**Puissance et rapidité** : L'éclair, symbole central de notre logo, transcende les frontières culturelles en évoquant la puissance, la rapidité et l'énergie dynamique. D'un point de vue psychologique, cette forme évoque la perception humaine de la vitesse et de l'action

immédiate, des qualités essentielles dans les jeux vidéo où la réactivité est cruciale. Des études montrent que les formes pointues et dynamiques comme celle de l'éclair activent les zones du cerveau associées à l'attention et à la réaction rapide, renforçant ainsi l'attrait visuel et l'engagement émotionnel envers notre marque.

**Technologie et innovation** : En plus de sa symbolique dynamique, l'éclair représente également l'innovation technologique et la modernité. Dans l'univers des jeux vidéo, où la technologie est au cœur de l'expérience, cette icône suggère notre engagement envers l'avant-garde et notre capacité à repousser les limites créatives et techniques pour offrir des expériences de jeu innovantes et captivantes.

**Impact visuel** : Visuellement frappant, l'éclair captive instantanément l'attention et crée une impression durable. Cette forme distinctive, par sa nature audacieuse et évocatrice, se distingue dans le paysage visuel saturé d'aujourd'hui, renforçant ainsi la visibilité et la reconnaissance de notre marque.

### **Pourquoi souligné ?**

**Accentuation** : Le soulignement stratégiquement placé sous le nom "Stonks" dans notre logo n'est pas seulement décoratif mais fonctionne comme un guide visuel pour l'identification et la mémorisation. Selon des principes de design psychologique, un élément de soulignement attire naturellement l'œil, aidant à focaliser l'attention sur le nom de l'entreprise et à renforcer sa présence visuelle dans l'esprit des consommateurs. Par ailleurs, "Stonks" est une expression internet humoristique qui déforme intentionnellement le mot "stocks" (actions en anglais) pour faire référence de manière ironique aux investissements et aux bénéfices financiers. En français, on pourrait traduire cela par "actions" ou par une expression équivalente comme "les actions qui montent en flèche". Le jaune accentué à l'illusion de pouvoir ne peut que plaire au public.

**Stabilité et structure** : Structuralement, le soulignement ajoute une stabilité visuelle au design global du logo, évoquant une fondation solide et une entreprise fiable. Cette ligne horizontale, en contraste avec les éléments plus dynamiques comme l'éclair, crée un équilibre esthétique qui contribue à une composition harmonieuse et esthétiquement agréable.

**Esthétique et équilibre** : En harmonisant les éléments diagonaux et courbes de l'éclair avec le soulignement horizontal, notre logo atteint un équilibre visuel qui est à la fois esthétiquement attrayant et fonctionnel. Ce contraste de formes et de lignes non seulement renforce la composition visuelle mais renforce également l'impact émotionnel et mémoriel de notre marque.



FIGURE 6 – Le logo de l’entreprise

#### 4.1.2 Inspirations Design

L'esthétique du jeu *Lands of Azerith* est profondément influencée par des classiques du jeu vidéo et des titres emblématiques du pixel art tels que *Stardew Valley* et *Undertale*. Ces jeux ont non seulement évoqué la nostalgie des genres passés, mais ont aussi démontré la capacité du pixel art à offrir une esthétique artistique distinctive tout en restant léger sur les ressources. Nous avons également puisé notre inspiration dans d'autres œuvres de pixel art, ce qui nous a permis d'explorer divers styles et techniques pour notre propre projet.

En plus d'évoquer la nostalgie des jeux classiques, le pixel art nous offre une exigibilité artistique tout en étant relativement léger en termes de ressources. Inspiré par des jeux emblématiques tels que *Stardew Valley*, *Undertale*, ainsi que par d'autres jeux rétro, le choix du style graphique pixel art en 2D pour *Lands of Azerith* provient du choix réfléchi de l'esthétique du jeu et son potentiel à immerger les joueurs dans un monde magique et riche en aventures. En effet, ce choix s'est révélé être une décision artistique judicieuse. Ce style graphique simple et détaillé à la fois confère au jeu un charme unique tout en permettant une représentation qualitative des environnements et des personnages. De plus, l'atmosphère visuelle du jeu est soigneusement choisie en fonction des différentes zones du jeu pour une meilleure immersion des joueurs dans l'univers apporté par *Lands of Azerith*.



FIGURE 7 – Exemple du design du jeu Undertale

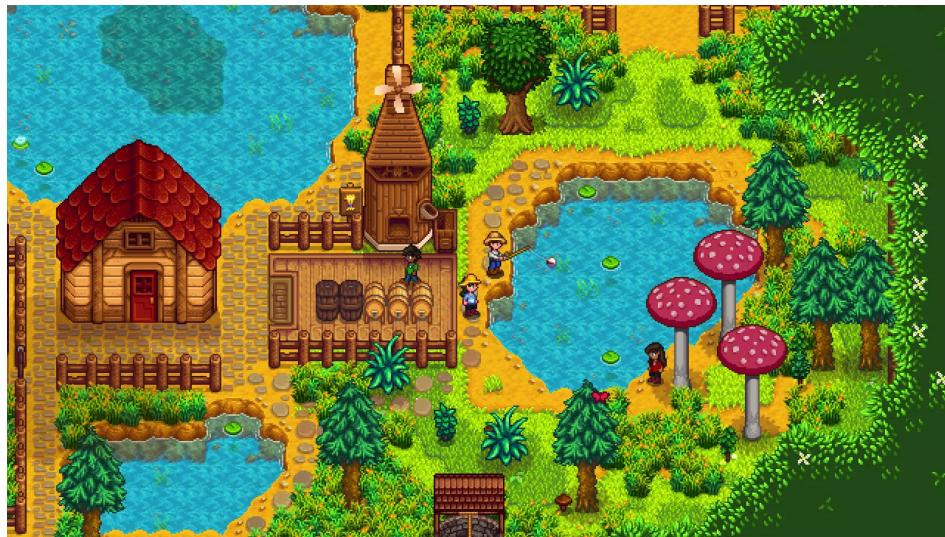


FIGURE 8 – Exemple de design du jeu Stardew Valley

#### 4.1.3 Creation et implementation design

Le choix du pixel art en 2D pour *Lands of Azerith* dcoule d'une dcision dlibre visant  immerger les joueurs dans un monde magique et riche en aventures. Ce style graphique combine simplicit et dtails minutieux, offrant un charme unique tout en permettant une reprsentation visuelle de haute qualit des environnements et des personnages du jeu. Chaque zone du jeu est soigneusement conue pour reflter une atmosphre distinque, renforant ainsi l'immersion des joueurs dans un univers captivant.

## Création des Textures

La création des textures pour *Lands of Azerith* a été un processus méticuleux, réalisé avec des outils comme GIMP pour assurer une flexibilité créative maximale. À partir de concepts initiaux inspirés d'images et de textures d'autres jeux, notre équipe a développé des textures variées pour les environnements, les personnages et les objets du jeu. Chaque texture est peaufinée avec attention pour enrichir l'expérience visuelle du joueur et garantir une cohérence esthétique à travers le monde de jeu :

- **Conceptualisation et Modélisation** : Exploration d'idées et de concepts pour chaque élément du jeu.
- **Réalisation des Textures** : Création détaillée des textures en plusieurs itérations pour atteindre le rendu final souhaité.
- **Intégration et Test** : Validation des textures dans le jeu pour assurer leur intégration harmonieuse et leur impact visuel optimal.



FIGURE 9 – Maisons créées pour le jeu

## Intégration des Éléments Graphiques dans le Jeu

L'intégration des textures et des éléments graphiques joue un rôle crucial dans la création d'une expérience immersive pour les joueurs. Chaque texture est soigneusement ajustée et testée pour s'assurer qu'elle correspond aux spécifications techniques du jeu et contribue à l'atmosphère globale de *Lands of Azerith*. Cette étape inclut l'incorporation des textures dans les décors, les personnages et les objets du jeu, garantissant ainsi une cohérence visuelle tout au long de l'aventure des joueurs.

- **Planification de la Carte** : Définition des zones géographiques, des villes et des donjons pour structurer le monde de jeu.

- **Création des Zones** : Conception détaillée des environnements variés pour offrir une exploration riche et diversifiée.
- **Intégration et Finalisation** : Liaison des zones par des chemins et des routes pour guider les joueurs à travers le monde de *Lands of Azerith*.

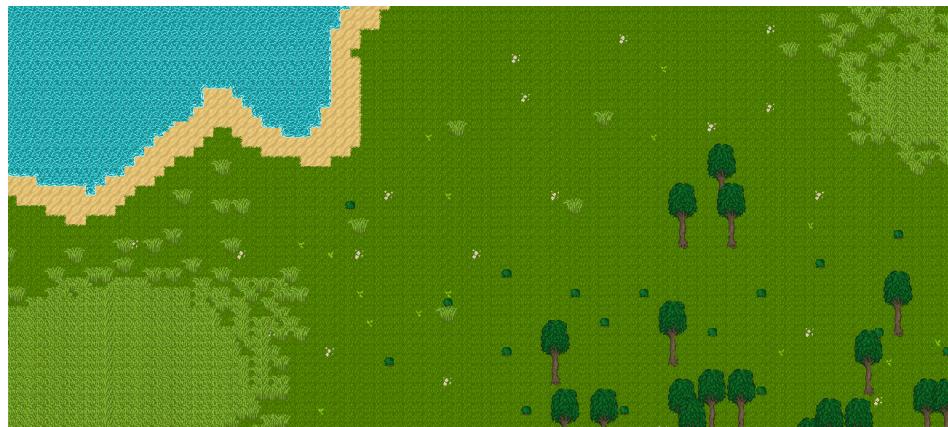


FIGURE 10 – Prairie Paradise



FIGURE 11 – Zone de départ



FIGURE 12 – Emberwood

#### 4.1.4 Utilisation du logiciel GIMP

Dans le développement de *Lands of Azerith*, la création des textures a été une étape essentielle et méticuleuse, guidée par un processus rigoureux utilisant des outils avancés tels que GIMP. Cette approche a permis d'atteindre une flexibilité créative maximale tout en assurant la cohérence visuelle et esthétique nécessaire pour immerger les joueurs dans notre monde fantastique.

GIMP, un logiciel libre et puissant de manipulation d'images, a été choisi pour sa polyvalence et ses capacités avancées de création graphique. Voici comment nous avons utilisé GIMP pour concevoir et peaufiner les textures de *Lands of Azerith* :

- **Édition et Retouche** : GIMP nous a permis d'éditer et de retoucher chaque détail des textures avec précision. Des outils comme les pinceaux, les filtres et les calques ont été utilisés pour ajuster la luminosité, le contraste, et les détails des textures, garantissant ainsi une qualité visuelle optimale.
- **Création de Motifs et de Détails** : Nous avons utilisé les fonctionnalités avancées de GIMP pour créer des motifs complexes et des détails minutieux dans les textures. Cela inclut la création de motifs floraux, de structures architecturales, et d'éléments naturels comme des rochers et des arbres, intégrés de manière organique dans l'environnement du jeu.
- **Conversion et Exportation** : GIMP nous a également fourni des options robustes pour convertir et exporter les textures dans les formats requis par notre moteur de jeu. Cela inclut la gestion des formats de fichier tels que PNG, JPEG, et les formats de texture compressés comme le format DDS pour une intégration optimale dans le pipeline de développement.

#### 4.1.5 Processus de Creation Collaboratif

La creation des textures pour *Lands of Azerith* a et  une initiative collaborative, impliquant des artistes graphiques, des concepteurs de niveaux et des developpeurs. Voici comment chaque  tage du processus a contribu    enrichir l'experience visuelle du jeu :

- **Conception Initiale** : Les artistes graphiques ont commenc  par  laborer des concepts artistiques d taill s, explorant divers th mes visuels et styles pour chaque zone de la carte. Ces concepts ont servi de base pour la creation des textures finales dans GIMP.
- **It rations et R visions** :   mesure que les textures prenaient forme, des it rations et des r visions ont  t  effectu es en collaboration avec les concepteurs de niveaux et les developpeurs pour s'assurer que les textures  taient non seulement esth tiquement plaisantes mais aussi fonctionnelles dans le contexte du jeu.
- **Optimisation pour les Performances** : Une attention particuli re a  t  port e   l'optimisation des textures pour les performances du jeu. Cela inclut la gestion de la r solution des textures, la r duction de la taille des fichiers et l'utilisation efficace des techniques de compression sans compromettre la qualit  visuelle.

#### 4.1.6 Int gration des Textures dans Godot

Une fois les textures cr  es et peaufin es dans GIMP, elles ont  t  int gr es dans Godot, notre moteur de jeu, pour construire les environnements interactifs de *Lands of Azerith*. Voici comment cette int gration a  t  g r e  :

- **Importation dans Godot** : Les textures ont  t  import es dans Godot en utilisant les fonctionnalit s int gr es pour g rer les ressources graphiques. Chaque texture a  t  assign e aux objets correspondants dans les sc nes de jeu, assurant ainsi une correspondance parfaite entre les visuels cr  es et les interactions du joueur.
- **Application des Textures** : Dans Godot, les textures ont  t  appliqu es aux objets de mani re   cr er des environnements riches en d tails visuels et en ambiance. Cela inclut la texture du sol, des murs, des objets interactifs et des  l ments d coratifs qui contribuent   l'immersion globale du joueur.

## 4.2 Musique et bruitages

### 4.2.1 Musiques

Le choix de l'électro, plus précisément de la Dance electro, pour la bande sonore de *Lands of Azerith* provient de son dynamisme et de sa capacité à renforcer l'énergie et l'immersion du jeu. En sélectionnant des compositions d'artistes comme *Cosmograph* et *Zekk*, nous avons assuré une harmonie entre l'audio et les visuels du jeu, offrant ainsi une expérience sensorielle complète et captivante.

### 4.2.2 Bande sonore

La bande sonore de *Lands of Azerith* est composée de 19 musiques existantes, sélectionnées spécifiquement pour enrichir les biomes et les situations du jeu.

Chaque biome dispose de trois versions de musique : jour, nuit et combat, adaptées pour capturer l'ambiance et l'intensité des différentes phases du gameplay.

Les musiques des biomes sont soigneusement conçues pour refléter l'environnement et l'atmosphère spécifiques de chaque zone du jeu. Elles contribuent à l'immersion des joueurs en créant une expérience sonore cohérente et captivante. Les compositions musicales varient en fonction du biome, offrant une diversité d'ambiances et d'émotions pour accompagner les joueurs tout au long de leur aventure.

De plus, les musiques de combat sont spécialement conçues pour intensifier l'action et l'excitation lors des affrontements avec les ennemis. Elles sont rythmées et énergiques, créant une tension et une immersion supplémentaires pendant les combats.

Enfin, les musiques de nuit apportent une atmosphère plus mystérieuse et sombre, renforçant l'exploration nocturne et offrant une expérience immersive unique.

Dans l'ensemble, la bande sonore de *Lands of Azerith* joue un rôle essentiel dans l'expérience globale du jeu, en créant une ambiance immersive et en renforçant l'émotion et l'immersion des joueurs dans cet univers fantastique.

### 4.2.3 Exemples de Musiques de Biomes

- La musique d'ambiance de jour dans les plaines d'Emberwood capture l'atmosphère sereine et pastorale du paysage, offrant aux joueurs une immersion tranquille dans cet environnement ouvert.
- La version nocturne de la même musique intensifie le ton, créant une atmosphère mystérieuse et potentiellement dangereuse alors que les menaces nocturnes émergent.

- Pendant les combats, la musique devient plus dynamique et rythmée pour stimuler l'adrénaline des joueurs, les incitant à réagir avec précision et stratégie.

#### 4.2.4 Musiques des Boss et Zones Pacifiques

Chaque boss est accompagné d'une musique distinctive et immersive, conçue pour refléter la menace et l'importance narrative de ces rencontres épiques. Ces compositions musicales renforcent l'intensité des combats tout en enrichissant l'expérience émotionnelle des joueurs à travers des thèmes mémorables.

#### 4.2.5 Voix Off

La voix off était initialement prévue pour offrir une narration riche et immersive tout au long de *Lands of Azerith*. Cependant, en raison de contraintes techniques liées au départ de la personne censée réaliser les voix, cette fonctionnalité a été modifiée. La narration et les éléments narratifs sont maintenant intégrés dans l'environnement visuel et musical du jeu, renforçant l'immersion des joueurs à travers des visuels évocateurs et des thèmes musicaux dynamiques.

# 5 Site web

## 5.1 Objectif du site web

Le site web est une tâche secondaire du projet, mais il est tout de même important pour la communication et la promotion du jeu. Il est essentiel pour permettre aux joueurs de découvrir le jeu, de suivre son développement et de le télécharger. C'est l'outil central de la partie communication du projet, et il est donc important de le maintenir à jour avec les dernières informations sur le jeu.

## 5.2 Développement

Le site possède son propre dépôt de code source sur GitHub à l'adresse [https://github.com/StonksIndustries/Azerith\\_Website](https://github.com/StonksIndustries/Azerith_Website). Nous avons utilisé l'éditeur de code VSCode pour développer le site web car il offre de nombreux outils très utiles pour cela. On peut citer l'intégration avec GitHub, les extensions pour le développement web, et les outils de débogage intégrés.

Le site web est développé en HTML, SCSS et JS, et est compilé avec Parcel. Leur choix a été motivé par leur simplicité et leur popularité, qui permettent de trouver facilement de l'aide en cas de problème. Parcel permet de compiler le site web sans aucune configuration et une très bonne optimisation des fichiers générés. Le SCSS remplace le CSS pour permettre une meilleure organisation du code et une meilleure maintenabilité. Le JS est utilisé pour les quelques animations et les interactions avec le visiteur.

Il est automatiquement compilé et publié grâce aux outils GitHub Actions et GitHub Pages. L'usage de ces outils permet de simplifier la publication du site web, et de le mettre à jour automatiquement à chaque modification du code source. Chaque nouvelle version envoyée par un développeur est automatiquement compilée et publiée sur le site web si aucune erreur n'est détectée.

## 5.3 Apparence



FIGURE 13 – Page d'accueil du site web, elle représente bien la simplicité voulu pour le site web.

Comme le montre la Figure 13, le site web est simple et épuré. Il est composé d'une seule page, laissant l'utilisateur scrollier pour découvrir les différentes sections. Elle possède un menu de navigation qui permet de se rendre directement aux sections les plus importantes du site web, et un pied de page avec des liens vers les réseaux sociaux et le dépôt de code source du jeu. Les différentes sections sont les suivantes :

- "À propos", cette partie présente le jeu, son style et le concept associé
- "Téléchargement", pour télécharger le jeu pour les différentes plateformes
- "L'équipe", qui présente les membres du groupe de développement de *Lands of Azerith*
- "Technologies", pour présenter les technologies utilisées pour le développement du jeu et du site web
- "Contact", pour contacter l'équipe de développement en redirigeant l'utilisateur vers une adresse mail ou le dépôt GitHub du projet

## 5.4 Responsive design

La plus importante amélioration du site web depuis la dernière soutenance est l'ajout du responsive design. Le site web est maintenant compatible avec les appareils

mobiles, et s'adapte automatiquement à la taille de l'écran de l'utilisateur. Aujourd'hui, la majorité des utilisateurs naviguent sur internet depuis leur smartphone, il est donc essentiel que le site web soit compatible avec ces appareils.

```
1 @media (max-width: 750px) { ... }  
2 @media (orientation: landscape) { ... }
```

FIGURE 14 – Exemple de *media queries* en CSS.

Pour réaliser cette amélioration, nous avons utilisé les *media queries* de CSS pour adapter le site web à différentes tailles d'écran. La Figure 14 montre un exemple utilisant cette fonctionnalités du langage CSS. La première ligne définit le style lorsque la largeur de l'écran est inférieure à 750 pixels, tandis que la deuxième ligne définit le style pour les orientations paysage.

# 6 Conclusion

## 6.1 Récapitulatif

Au terme de notre année académique à l'EPITA, nous sommes fiers de présenter les résultats de notre projet *Lands of Azerith*. Ce projet a été une aventure marquée par des défis techniques, des adaptations organisationnelles, et des innovations créatives. Nous avons entrepris de développer un jeu vidéo immersif qui non seulement nous a permis de mettre en pratique nos compétences en programmation et en design de jeux, mais qui a aussi été un véritable terrain d'apprentissage en gestion de projet et en collaboration d'équipe.

Notre projet a commencé avec une vision claire : créer un jeu d'aventure où les joueurs peuvent explorer un monde riche et interactif, incarner des héros diversifiés, et participer à des quêtes palpitantes. L'acte 1 du jeu, centré sur le village d'Emberwood et ses environs, a été le cœur de notre développement. Nous avons intégré des mécanismes variés tels que le choix de classes de personnages, la gestion des équipements, l'utilisation des éléments, et un système de commerce en ville. Chaque étape du jeu a été soigneusement conçue pour offrir une expérience de jeu captivante et fluide.

Le choix de Godot Engine comme moteur de jeu a été déterminant. Godot nous a offert une flexibilité et une puissance adaptées à nos besoins, sans les contraintes financières des moteurs propriétaires. Nous avons pu exploiter ses capacités pour créer des environnements détaillés et des mécaniques de jeu complexes. La gestion de version via GitHub a été essentielle pour coordonner nos efforts, permettant une collaboration efficace malgré les défis liés aux départs de membres et à l'intégration de nouveaux.

La dynamique de notre équipe a connu plusieurs bouleversements. Le départ d'Alexandre Colsch et de Mohamed Aziz ben Amor a été compensé par l'arrivée de Louise Fussien, qui a apporté une nouvelle énergie et des compétences cruciales. L'utilisation de Discord pour la communication instantanée et des réunions en personne à EPITA a renforcé notre cohésion et notre capacité à résoudre les problèmes rapidement. La répartition des tâches a été ajustée pour tirer le meilleur parti des compétences de chaque membre, garantissant une progression constante malgré les obstacles.

L'environnement de travail, dans le contexte du projet de développement de jeux vidéo, est caractérisé par l'utilisation du moteur de jeu Godot et de l'environnement de développement intégré (IDE) Rider. Godot, un moteur de jeu open-source, est reconnu pour sa facilité d'utilisation et sa capacité à gérer des projets de jeux vidéo en 2D et en 3D. L'adoption de Godot a été motivée par sa communauté active et ses ressources abondantes, qui facilitent l'apprentissage et la résolution de problèmes. En parallèle, Rider, un IDE multiplateforme développé par JetBrains, a été choisi pour

ses fonctionnalités étendues de support de langages de programmation, en particulier C#. L'intégration de Godot avec Rider a permis d'exploiter pleinement les capacités de scripting C# de Godot, offrant ainsi une expérience de développement plus enrichissante et productive.

Cependant, le développement n'a pas été exempt de défis, notamment ceux liés aux changements de version de Godot. Comme le moteur est en constante évolution, les mises à jour fréquentes, bien que bénéfiques en termes de nouvelles fonctionnalités et d'améliorations de performance, peuvent introduire des modifications significatives qui impactent la compatibilité du code existant. Pour surmonter ces difficultés, l'équipe a mis en place plusieurs stratégies, telles que des sessions de formation interne, la création d'un environnement de test pour évaluer l'impact des changements sur le projet, et l'encouragement à la révision de code et au pair programming. La manipulation des fichiers JSON a également été un aspect essentiel du projet. JSON, un format de données léger, a été largement utilisé pour stocker et échanger des données structurées. En dépit des défis initiaux liés à la familiarité avec ce format de fichier, l'équipe a mené une étude approfondie de JSON et de ses applications. Le processus d'intégration de fichiers JSON dans le code a impliqué la lecture et l'écriture de fichiers JSON, la sélection des attributs à sauvegarder, et la récupération des objets à partir de fichiers JSON.

En ce qui concerne l'architecture du jeu, l'organisation claire et la gestion efficace des ressources ont été facilitées par une structure de dossiers bien définie. Le jeu vidéo s'est appuyé sur divers dossiers, notamment Assets pour les ressources visuelles et sonores, Caractères pour les images des personnages, Items pour les fichiers JSON associés aux différents objets, et Loottables pour les données de niveau de détail des monstres.

Dès l'ouverture du menu, l'effet parallax enchantera visuellement en animant différents plans à des vitesses distinctes, créant ainsi une profondeur saisissante. Cette technique, intégrée de manière astucieuse grâce à Godot, plonge les joueurs dans une perspective 2D vivante dès le premier instant.

En parallèle, notre approche du multijoueur repose sur une architecture robuste basée sur l'envoi de paquets, simplifiant la connexion et la synchronisation entre les joueurs sans compromettre la fluidité du gameplay. Cette solution permet à jusqu'à huit joueurs de se connecter sans tracas, chacun pouvant rejoindre une partie en utilisant un système intuitif sans avoir à saisir manuellement des adresses IP.

Les contrôles du jeu sont pensés pour une jouabilité intuitive : de simples touches comme Z, Q, S, D pour les déplacements, Maj pour la course, et Tab pour l'inventaire assurent une navigation fluide dans les environnements variés du jeu. De plus, des actions spécifiques comme l'interaction avec le clic gauche de la souris sont finement intégrées pour une expérience utilisateur enrichie.

L'intelligence artificielle des monstres apporte une autre dimension au jeu, avec des comportements variés (agressif, neutre, passif, peureux) qui réagissent dynamiquement aux actions des joueurs. Chaque type de monstre est méticuleusement conçu pour enrichir l'interaction, utilisant des mécanismes de détection de collision et de gestion de signaux pour ajuster leur comportement en temps réel.

L'intégration des mécanismes avancés comme la navigation A\* pour le déplacement des monstres, la gestion détaillée de l'inventaire du joueur, et la création dynamique de maps avec transitions fluides dans notre jeu sous Godot représente un effort significatif pour offrir une expérience de jeu immersive et engageante. Chaque élément technique, qu'il s'agisse de l'algorithme de pathfinding A\* pour des déplacements intelligents, de la gestion des objets collectés via un inventaire structuré, ou encore de l'animation fluide des personnages et des environnements, contribue à créer un monde virtuel cohérent et captivant pour les joueurs.

L'utilisation de tilesets et de textures bien intégrées garantit non seulement une esthétique visuelle agréable mais aussi une navigation fluide entre les différents environnements du jeu. Les transitions entre les maps sont gérées avec précision, offrant une immersion sans heurts tandis que les personnages et les monstres animés ajoutent une dimension réaliste et interactive à l'expérience de jeu.

En combinant ces aspects techniques avec une mécanique de jeu bien pensée incluant des checkpoints pour la progression du joueur et une mécanique de loot gratifiante à partir des monstres vaincus, notre jeu vise à captiver les joueurs tout en leur offrant un défi stratégique et une exploration enrichissante. En définitive, cette approche technique avancée sous Godot non seulement améliore la jouabilité mais aussi renforce l'immersion et le plaisir des joueurs, faisant de chaque session de jeu une expérience mémorable et divertissante. L'aspect visuel et sonore de *Lands of Azerith* a été élaboré avec soin pour immerger les joueurs dans notre univers.

Nous avons puisé notre inspiration dans diverses sources pour créer un design original et attrayant. L'intégration du son, des effets visuels, et une interface utilisateur intuitive ont été des éléments clés pour enrichir l'expérience de jeu. Le site web du projet a également été développé pour offrir une vitrine de notre travail, facilitant l'accès à des informations sur le jeu et son développement.

Les défis que nous avons rencontrés, notamment les délais imprévus et les changements d'équipe, ont été surmontés grâce à une planification rigoureuse et une flexibilité dans notre approche. Nous avons appris à nous adapter rapidement, à redistribuer les responsabilités, et à maintenir notre engagement envers la qualité du projet. Chaque itération de notre cahier technique a été une occasion de réévaluer et d'améliorer notre travail, assurant que nous restions alignés sur nos objectifs initiaux tout en intégrant de nouvelles idées et solutions.

## 6.2 Les joies et les peines

Le chemin parcouru pour mener à bien notre projet a été parsemé de défis et de moments gratifiants, marquant une expérience enrichissante où la gestion du temps et des ressources humaines a été cruciale. Voici un récit des hauts et des bas que nous avons traversés ensemble.

### Défis Rencontrés

Gérer à la fois les études et le projet simultanément a été une tâche ardue. Trouver des plages horaires communes pour travailler efficacement a souvent représenté un défi majeur, surtout avec les contraintes personnelles et académiques de chacun. Le départ de deux membres du groupe, Momo et Alex, a également constitué un coup dur, nécessitant un réajustement rapide pour maintenir le cap du projet.

### Adaptation et Résilience

L'arrivée de Louise dans notre équipe a été une bouffée d'air frais. Sa contribution a apporté de nouvelles idées et une énergie positive, renforçant notre capacité à relever les défis restants. Ce changement a été une occasion précieuse pour apprendre à gérer efficacement les transitions d'équipe tout en maintenant la cohésion et la productivité.

### Apprentissage de la Gestion du Temps et du Groupe

Naviguer à travers ces dynamiques de groupe nous a offert une expérience précieuse dans la gestion du temps et des ressources humaines. Apprendre à coordonner les efforts, à déléguer les tâches et à maintenir une communication claire a été essentiel pour atteindre nos objectifs collectifs. Cette expérience nous a préparés de manière significative à l'environnement dynamique et collaboratif de l'entreprise.

### Satisfaction dans la Conformité aux Attentes

Finalement, livrer un projet qui répond aux attentes fixées représente une grande satisfaction. Cela démontre notre capacité à transformer des défis en opportunités, à faire preuve de résilience face aux obstacles et à atteindre des résultats de qualité. Cette réalisation renforce notre confiance et notre préparation pour les défis futurs, tant sur le plan académique que professionnel.

## 6.3 Remerciements

À ceux qui, par leur soutien et leur guidance, ont éclairé notre chemin tout au long de ce projet ambitieux, nous adressons nos remerciements sincères et empreints de reconnaissance. À nos mentors et encadrants de l'EPITA, vous avez été bien plus que des guides : des conseillers précieux dont la sagacité a éclairé nos démarches et

nourri nos réflexions. Vos encouragements et vos conseils ont été les pivots sur lesquels s'est articulée notre entreprise, et sans votre expertise, notre parcours aurait été semé d'embûches insurmontables.

À nos proches, familles et amis, nous exprimons notre profonde gratitude pour votre soutien indéfectible. Votre patience et votre compréhension durant les périodes intenses de développement ont été notre bouclier contre les vents contraires. Chaque succès que nous célébrons aujourd'hui porte l'empreinte de votre affection et de votre encouragement constant.

Nous ne saurions également passer sous silence l'importance cruciale des ressources open-source et des communautés de développeurs qui ont enrichi notre expérience. Par leur générosité et leur partage de connaissances, ces précieux collaborateurs ont illuminé notre chemin, offrant des solutions éprouvées et des perspectives nouvelles qui ont enrichi notre approche du développement.

Les forums de discussion et les plateformes de partage de code ont été des carrefours d'échange essentiels, où chaque question technique trouvait réponse et chaque idée trouvait écho. La richesse de ces interactions a été pour nous une source constante d'inspiration et de résolution de problèmes, permettant à notre projet de croître et de s'épanouir au-delà de nos attentes initiales.

Enfin, nous tenons à exprimer notre profonde reconnaissance envers les développeurs dédiés qui ont créé et maintenu les outils open-source que nous avons utilisés. Leur dévouement et leur expertise ont été des piliers essentiels de la fiabilité et de la qualité de notre travail. Leur contribution silencieuse mais indispensable a façonné notre succès et notre satisfaction aujourd'hui.

En conclusion, *Lands of Azerith* incarne bien plus qu'un simple projet académique. Il symbolise notre engagement commun envers l'innovation et la création, ainsi que notre capacité à surmonter les défis avec ingéniosité et persévérance. Que notre gratitude résonne comme une reconnaissance profonde envers tous ceux qui ont contribué, directement ou indirectement, à la réalisation de cette œuvre collective.

Avec respect et reconnaissance,

Ayemane Bouarbi, Louise Fussien, Michaël Museux, Martin Pasquier



Last updated on 18 juin 2024 at 15:09.