

Rapport Final: Isidor's Quest

SAE SS.01 DÉVELOPPEMENT AVANCÉ 2023-2024

COLLOMBET Nathan 301 | ZHANG Anxian 302 |
LIN Xingtong 302 | LIN Oscar 303 | SELVARATNAM Akash 303

Remerciement

Nous tenons à exprimer nos plus sincères remerciements à toutes les personnes qui ont contribué à la réalisation de notre projet de jeu vidéo. Leur aide et leur soutien ont été essentiels à la réussite de cette aventure.

Nous adressons tout d'abord nos remerciements les plus chaleureux à notre tuteur, Monsieur Ouziri, pour son encadrement, ses précieux conseils et son accompagnement tout au long de ce projet. Son expertise pour le développement informatique et sa gentillesse ont été des piliers dans l'avancement de notre travail.

Un grand merci également à Monsieur Brette pour l'organisation du projet. Sa capacité à structurer les différentes étapes a été un facteur clé dans la bonne conduite de notre projet.

Nous souhaitons aussi remercier chaleureusement Madame Tadier pour son aide précieuse dans le domaine de la communication. Ses conseils concernant la présentation et la diffusion de notre travail ont été cruciaux pour assurer la visibilité et le succès de notre jeu.

Nous tenons également à souligner l'importante contribution du corps enseignant et de l'ensemble du personnel académique. Leur soutien, leurs enseignements et leur enthousiasme ont grandement enrichi notre parcours éducatif. Leur disponibilité et leur engagement envers notre réussite ont joué un rôle dans l'accomplissement de nos objectifs.

Ce projet a été une aventure exceptionnelle, non seulement en termes d'acquisition de connaissances et de compétences techniques, mais aussi en termes de travail d'équipe et de collaboration. Les leçons apprises au cours de cette expérience resteront gravées dans notre mémoire.

Merci à tous pour avoir rendu ce projet possible.

Rédigé par COLLOMBET Nathan

Correction, harmonisation et mise en page par COLLOMBET Nathan

Table des matières

| | | |
|--------|--|----|
| 1 | Description du projet : Isidor's Quest : Chasing the glow | 4 |
| 2 | Pourquoi ce projet et pas un autre ?..... | 4 |
| 3 | Phases d'analyses | 5 |
| 3.1 | Le marché actuel | 5 |
| 3.2 | Jeu vidéo indépendant | 6 |
| 3.3 | Ce qui fait notre singularité..... | 6 |
| 3.4 | Choix techniques | 7 |
| 3.4.1 | Le rôle de chaque composant | 7 |
| 3.4.2 | Pourquoi ces outils plutôt que d'autres ? | 8 |
| 3.5 | Charte graphique..... | 11 |
| 3.6 | Notre organisation | 12 |
| 3.6.1 | Planification..... | 12 |
| 3.6.2 | Rôles de chacun(e) | 12 |
| 3.6.3 | Répartition des tâches..... | 13 |
| 4 | Nos deux applications | 14 |
| 4.1 | Les services proposés (architecture fonctionnel)..... | 14 |
| 4.1.1 | Site..... | 14 |
| 4.1.2 | Jeu..... | 15 |
| 4.2 | Nos phases de développement | 15 |
| 4.3 | Architecture fonctionnelle et technique (diagramme UML)..... | 16 |
| 4.3.1 | Diagramme de cas d'utilisation de l'application | 16 |
| 4.3.2 | Architecture globale du projet | 17 |
| 4.3.3 | Diagramme de séquence de la page oublie de mot de passe..... | 18 |
| 4.3.4 | Diagramme de séquence de la page inscription | 19 |
| 4.3.5 | Diagramme de séquence de la page vérification de code | 20 |
| 4.3.6 | Diagramme de séquence de la page de modification de mot de passe | 21 |
| 4.3.7 | Diagramme de séquence de la page de modification des données utilisateur | 22 |
| 4.3.8 | Diagramme de séquence de la page de connexion..... | 23 |
| 4.3.9 | Diagramme de séquence de la page de contact | 24 |
| 4.3.10 | Les états et transitions du joueur..... | 25 |
| 4.3.11 | Les états et transitions d'un ennemi | 25 |
| 4.4 | Conception des applications (explications)..... | 26 |
| 4.4.1 | Site..... | 26 |
| 4.4.2 | Jeu..... | 41 |

| | | |
|------|---|----|
| 4.5 | Intégration du jeu sur le site | 58 |
| 5 | Bilan de montée en compétences..... | 61 |
| 5.1 | Selvaratnam Akash | 61 |
| 5.2 | LIN Xingtong | 61 |
| 5.3 | ZHANG Anxian | 62 |
| 5.4 | LIN Oscar..... | 62 |
| 5.5 | COLLOMBET Nathan..... | 63 |
| 6 | Les difficultés rencontrées | 64 |
| 6.1 | Selvaratnam Akash | 64 |
| 6.2 | LIN Xingtong | 64 |
| 6.3 | ZHANG Anxian | 64 |
| 6.4 | LIN Oscar..... | 64 |
| 6.5 | COLLOMBET Nathan..... | 65 |
| 7 | Ce qu'il reste à accomplir | 65 |
| 8 | Perspectives d'amélioration..... | 66 |
| 9 | Annexes | 67 |
| 9.1 | Annexe 1 : Résumé | 67 |
| 9.2 | Annexe 2 : Abstract | 68 |
| 9.3 | Annexe 3 : Les sources | 69 |
| 9.4 | Annexe 4 : Images du diagramme de Gantt..... | 71 |
| 9.5 | Annexe 5 : Table des illustrations..... | 73 |
| 9.6 | Annexe 6 : Cahier des charges..... | 74 |
| 9.7 | Annexe 7 : Proposition de projet SAE S5..... | 74 |
| 9.8 | Annexe 8 : Poster..... | 75 |
| 9.9 | Annexe 9 : Glossaire | 75 |
| 9.10 | Annexe 10 : Lien GitHub pour le code source | 76 |

Introduction

1 DESCRIPTION DU PROJET : ISIDOR'S QUEST : CHASING THE GLOW

Dans le cadre de notre troisième année de BUT Informatique, il nous a été demandé de concevoir une application en adéquation avec nos projets professionnels et nos passions personnelles.

Dans un univers numérique où l'innovation et la créativité sont les moteurs principaux, notre jeu vidéo de plateforme, accessible via un navigateur, se présentera sous la forme d'une expérience audacieuse entre nostalgie et modernité en adoptant un style en 2D pixel art. Au début d'une nouvelle partie, l'utilisateur sera invité à choisir parmi deux classes de héros : le guerrier ou l'archer. Le joueur sera ensuite confronté à des monstres et à des énigmes de difficulté croissante, nécessitant compétences et stratégie pour progresser aux niveaux suivants.

Rédigé par ZHANG Anxian et COLLOMBET Nathan

2 POURQUOI CE PROJET ET PAS UN AUTRE ?

Après une longue phase de réflexion, nous avons décidé de nous orienter vers le domaine du jeu vidéo, secteur qui représente à lui seul une part non négligeable du marché ! Ce domaine suscite également un intérêt majeur au sein de notre équipe. Malgré l'immensité des possibilités de création de jeux, nous avons finalement opté pour un style emblématique, à l'image de Mario, Kirby ou Sonic.

Ayant tous joué à ces jeux dans notre enfance, réaliser un jeu de ce type serait pour nous un rêve devenu réalité. De plus, la plupart des langages utilisés dans ce projet ne sont pas entièrement maîtrisés par les membres de l'équipe. C'est donc également une occasion idéale pour enrichir notre CV.

Rédigé par ZHANG Anxian et SELVARATNAM Akash

Développement

3 PHASES D'ANALYSES

3.1 LE MARCHE ACTUEL

Aujourd'hui, le marché du jeu vidéo est plus dynamique et diversifié que jamais, avec des géants tels que Nintendo, Sony, et Microsoft dominant le secteur des consoles, tandis que des acteurs comme Valve avec Steam et Epic Games avec leur Epic Games Store révolutionnent l'accès aux jeux sur PC.

En 2023, les recettes du secteur du jeu vidéo sont estimées à 245,10 milliards de dollars américains et devraient même atteindre 378,08 milliards de dollars américains en 2028. Cette croissance s'explique par l'augmentation du nombre de plateformes disponibles. De nos jours, il est possible de jouer à des jeux relativement puissants sur nos smartphones, ce qui était impensable il y a 10 ans. De plus, la diversité des types de jeux attire un nombre croissant de consommateurs. En 2022, on comptait près de 1,8 milliard de joueurs à travers le monde.

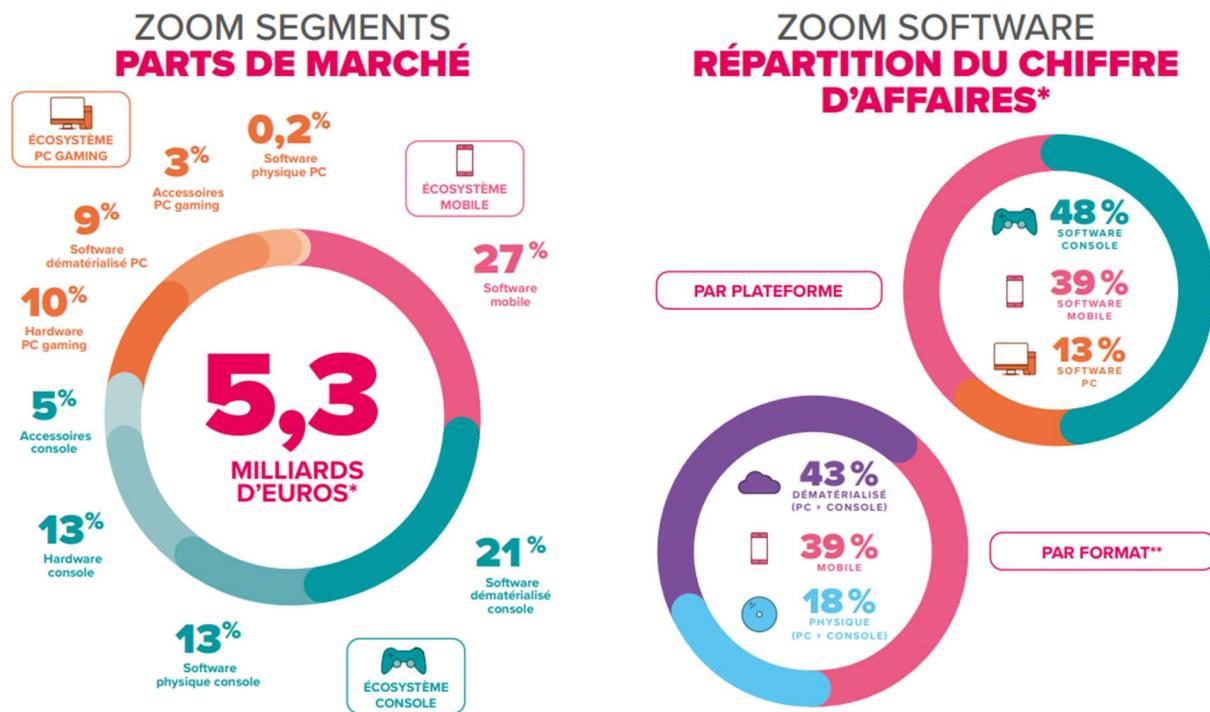


Figure 1: Bilan du marché Français 2020 dans le secteur du jeu vidéo. [L'essentiel-jeu-vidéo \[Fichier PDF\]. Page 8. 2021.](#)

Rédigé par ZHANG Anxian, SELVARATNAM Akash et COLLOMBET Nathan

3.2 JEU VIDEO INDEPENDANT

En parallèle, les jeux en 2D connaissent un renouveau spectaculaire, portés par la vague de « l'indie gaming ». Ces jeux indépendants, souvent caractérisés par leur esthétique en pixel art et leur gameplay innovant, ont su conquérir un public large et varié, prouvant que la simplicité graphique n'entrave en rien la profondeur et l'immersion ludique. De plus, l'essor des technologies web a ouvert la voie à une nouvelle ère de jeux accessibles directement via les navigateurs, élargissant ainsi l'audience potentielle au-delà des gamers traditionnels.

Il fallait donc se démarquer des géants du secteur en surfant sur la vague de popularité des jeux indépendants et l'utilisation d'une plateforme web.

Rédigé par *COLLOMBET Nathan*

3.3 CE QUI FAIT NOTRE SINGULARITE

L'intégration de jeux sur des sites web est une pratique courante, comme on peut le voir sur des plateformes telles que jeux.fr. Cependant, notre jeu, bien qu'hébergé également sur un site web, ne sera pas regroupé avec d'autres jeux. Il s'agit d'un site dédié exclusivement à notre jeu, « Isidor's Quest ». Le site vise à immerger l'utilisateur dans l'univers et l'ambiance du jeu avant même qu'il ne commence à jouer.

En effet, notre jeu se démarque dans ce paysage du jeu vidéo compétitif par son approche unique. En combinant l'aspect rétro du pixel art avec des mécanismes de jeu modernes, nous avons créé une expérience à la fois fraîche et familière. Notre choix de le rendre accessible via le web garantit une facilité d'accès inégalée, permettant aux joueurs de plonger dans l'aventure sans les contraintes d'installation ou de configuration matérielle spécifique. De plus, notre engagement à offrir un contenu riche et une narration captivante place notre jeu comme un incontournable dans le domaine des jeux 2D sur le web.

Enfin, notre jeu intègre un mécanisme de défense, élément rare dans ce type de jeu. Il convient aussi de mentionner le « village », une zone interactive où les joueurs peuvent effectuer diverses actions auprès des PNJ (Personnages Non-Joueurs), comme l'amélioration de leur arbre de compétences ou l'achat d'objets avec l'or acquis dans les différents niveaux.

Rédigé par *ZHANG Anxian, SELVARATNAM Akash et COLLOMBET Nathan*

3.4 CHOIX TECHNIQUES

3.4.1 Le rôle de chaque composant

3.4.1.1 Jeu

Unity Engine : pour le développement du jeu, tant sur le plan UI/UX que développement pur.

Visual Studio : Environnement de développement conseillé par Unity.

Piskelapp : Permet de créer des designs en 2D pixelisé.

Pinetools : Utilisé pour pixeliser une image.

OpenGameArt & CraftPix : Recherche de Sprite (design utilisé pour le jeu).

Rédigé par ZHANG Anxian

3.4.1.2 Web

Figma : Maquettage du site.

React Native : Framework utilisé pour développer la partie Front-end.

Node.JS : Plateforme de développement utilisé pour la partie Back-end.

Visual Studio code : Environnement de développement utilisé pour le site.

MongoDB : Base de données non relationnelle, utilisée pour le stockage des données des utilisateurs.

Rédigé par ZHANG Anxian

3.4.1.3 Gestion et organisation

Trello : Organisation des tâches

Google Drive : Echange de documents

GitHub : Gestionnaire de versions

Discord : Communication

Notion : Création du diagramme de Gantt

Visual Paradigm : Mise en place des diagrammes UML

Rédigé par ZHANG Anxian

3.4.2 Pourquoi ces outils plutôt que d'autres ?

3.4.2.1 Jeu

Lorsqu'il s'agit de développement de jeux, Unity Engine et Unreal Engine viennent immédiatement à l'esprit. Mais quelle est la différence entre les deux, et pourquoi avons-nous choisi Unity ? **Unity** bénéficie d'une communauté plus vaste que celle d'Unreal, ce qui facilite la recherche de documentation et de support. De plus, Unity est davantage adapté aux débutants dans le domaine du jeu vidéo, contrairement à Unreal qui est plutôt orienté vers des jeux de type AAA (jeux développés et produits par des grandes entreprises de l'industrie du jeu).

En ce qui concerne le design, nous avons opté pour des modèles préexistants et open source disponibles sur Internet, dans le but de gagner du temps dans le développement. Nous avons utilisé des sites tels que **OpenGameArt** et **CraftPix**, qui offrent un large éventail d'éléments graphiques en libre accès. Parfois, nous tombons sur des ressources qui ne sont pas en pixel art, auquel cas nous utilisons **Pinetools**, un logiciel de pixélisation. Pinetools est l'une des rares applications que nous avons testées qui ne déforme pas l'image originale. Lorsqu'aucune ressource existante ne correspond à nos besoins, nous créons alors nos propres designs à partir de zéro avec **Piskelapp**.

Rédigé par ZHANG Anxian et SELVARATNAM Akash

3.4.2.2 Web

Vous cherchez à créer des maquettes d'applications web en collaborant avec d'autres membres, tout en bénéficiant d'un outil gratuit et accessible ? **Figma** est l'outil idéal ! Contrairement à ses concurrents, souvent payants ou devenant payants après une période d'essai, Figma reste gratuit. Il est facile à prendre en main et permet également la réalisation de maquettes pour applications mobiles.

Côté Front-end, nous avons opté pour le framework **React Native**, basé sur le langage Javascript. Ce choix s'est imposé pour deux raisons : d'une part, cet outil est l'un des plus utilisés et demandés sur le marché ; d'autre part, contrairement à d'autres frameworks, React Native permet de développer des pages web interactives très optimisées grâce à ses mises à jour rapides. De plus, à la différence de ReactJS, React Native offre la possibilité de développer une application utilisable sur un navigateur, mais aussi de la compiler en tant qu'application mobile.

Pour le back-end, nous avons choisi **Node.js**, en combinaison avec le framework Express.js (utilisant également le Javascript), pour faciliter la création et la gestion rapide de serveurs traitant les requêtes HTTP, la gestion des sessions utilisateurs, ainsi que l'utilisation des cookies. Il s'agit également de l'un des back-ends les plus utilisés en synergie avec React.

Quant à la base de données, nous utilisons **MongoDB**, qui se distingue par son caractère non relationnel, c'est-à-dire qu'elle n'adopte pas le schéma tabulaire classique en lignes et colonnes. Cette particularité offre une plus grande flexibilité et évolutivité. De surcroît, Node.js propose la bibliothèque Mongoose, qui simplifie considérablement les opérations sur MongoDB.

Nous travaillons également avec l'éditeur de code **Visual Studio Code**, choisi pour sa légèreté et la variété de ses extensions. Parmi celles-ci, Simple React Snippets est particulièrement utile, facilitant le développement rapide grâce à l'utilisation de raccourcis. Par exemple, pour créer un composant, il suffit de taper "rsc" puis la touche tabulation, plutôt que de tout saisir manuellement.

Rédigé par ZHANG Anxian et SELVARATNAM Akash

3.4.2.3 Gestion et organisation

Trello se révèle être le meilleur choix pour la gestion de projets simples. Il suffit de créer des tableaux par thèmes généraux et d'y ajouter des cartes. Cette organisation permet de suivre facilement l'avancement de notre projet en naviguant d'un tableau à l'autre.

Qui ne connaît pas Google, l'un des géants du GAFAM ? L'entreprise a développé **Google Drive**, une application permettant non seulement de collaborer avec les membres de l'équipe, mais aussi de créer, éditer et supprimer différents types de fichiers (documents, tableurs, présentations). Nous utilisons cet outil car il offre le plus d'espace de stockage gratuit, avec 15 Go.

Les outils de versionning sont nombreux, mais certains se démarquent par leur pertinence, comme la gestion de version décentralisée (où chaque développeur possède ses propres dépôts et copies locales). Cela permet à chacun de travailler à son rythme. Nous avons choisi Git, et plus spécifiquement **GitHub**, qui offre la possibilité de créer plusieurs branches locales indépendantes et un tableau de bord personnel pour suivre les issues et les pull requests. De plus, en cas de problème, la vaste communauté présente sur GitHub est un atout considérable.

Discord, outil de communication, de visioconférence et d'envoi de fichiers, se distingue par sa flexibilité et la possibilité d'ajouter des bots sur nos serveurs. Ayant tous utilisé cette plateforme pendant plus de 2 ans, son choix s'est imposé naturellement.

Concernant **Notion**, bien connu comme outil no-code de productivité, il permet également de créer des diagrammes de Gantt. Après avoir exploré d'autres applications, souvent payantes ou peu intuitives pour la création de diagrammes, Notion est devenu notre choix de prédilection.

Pour les diagrammes UML, bien que Lucidchart soit une option intéressante, son utilisation est limitée à la création de seulement trois documents. C'est pourquoi nous avons opté pour **Visual Paradigm**, qui permet d'ajouter et de confectionner un nombre illimité de fichiers, tant que la limite de stockage de 1 Go n'est pas atteinte.

Rédigé par ZHANG Anxian

3.5 CHARTE GRAPHIQUE

Dans la conception de notre jeu vidéo en 2D pixel art et de notre site web, le soin apporté aux détails graphiques est essentiel pour créer une expérience visuelle harmonieuse et engageante. Le logo, qui représente la tête d'un des personnages principaux, illustre parfaitement cette démarche. Cette représentation simplifiée mais emblématique favorise une reconnaissance immédiate et une association forte avec le jeu. Elle est non seulement esthétique mais aussi pratique, assurant une visibilité claire sur divers supports.

La palette de couleurs, s'étendant du bleu à l'orange, est directement inspirée de cette icône visuelle. Ces couleurs vives et dynamiques ne se contentent pas de capturer l'essence de notre personnage ; elles créent également un univers visuel distinctif et attrayant qui se démarque dans le paysage des jeux vidéo. Cette palette est utilisée de manière cohérente à travers toutes les interfaces du jeu et du site web, établissant une identité de marque forte et immédiatement reconnaissable.

Quant à la typographie, notre choix d'une police de caractères pixelisée est un clin d'œil direct à l'esthétique classique des jeux rétro, tout en restant fidèle à notre thème de pixel art. Cette typographie ne se limite pas à être un simple élément de design ; elle renforce l'ambiance nostalgie tout en assurant une parfaite harmonie avec le style graphique global du jeu.



Figure 2: charte graphique utilisée pour l'application (capture d'écran de la charte graphique)

Rédigé par ZHANG Anxian et COLLOMBET Nathan

3.6 NOTRE ORGANISATION

3.6.1 Planification

Avant de débuter le projet, nous avons élaboré un diagramme de Gantt que nous avons intégré sous forme de tableau dans le cahier des charges (voir annexe 6). Bien que ce diagramme fût initialement accompagné d'un lien, celui-ci est devenu obsolète, rendant la consultation du diagramme sous cette forme moins idéale. Ainsi, nous avons ajouté en annexe 4 des images illustrant le diagramme de Gantt. Celui-ci est présenté à titre indicatif ; toutefois, au cours du projet, nous avons ajouté des tâches et réalisé certaines fonctionnalités plus tôt que les dates initialement prévues dans le diagramme.

Dans le cadre de la rédaction du cahier des charges, nous avons aussi consacré du temps à l'élaboration du cahier de recette, détaillé dans l'annexe 6.

Rédigé par ZHANG Anxian

3.6.2 Rôles de chacun(e)

Au-delà de la nécessité que le projet corresponde à nos aspirations personnelles et professionnelles, il représente avant tout une opportunité d'apprentissage. Pour cette raison, nous avons chacun endossé plusieurs rôles que nous jugeons pertinents, comme détaillé dans l'annexe 6 dans le cahier des charges.

| | COLLOMBET Nathan | LIN Oscar | SELVARATNAM Akash | ZHANG Anxian | LIN Xingtong |
|------------------------|---------------------|--------------|----------------------|-----------------|-----------------|
| Web designer | ✓ | | ✓ | ✓ | ✓ |
| Game designer | | | | ✓ | ✓ |
| Développeur Web | ✓ | ✓ | ✓ | ✓ | ✓ |
| Développeur | | ✓ | ✓ | ✓ | ✓ |
| Logiciel | | | | | |
| Testeur | | | | ✓ | |

Figure 3 : Tableau des différents rôles de chacun et chacune

Rédigé par ZHANG Anxian

3.6.3 Répartition des tâches

Dans notre organisation, il n'y a pas, à proprement parler, de chef de projet. Nous utilisons l'application de gestion de tâches Trello, qui nous permet d'ajouter des missions et de les étiqueter avec l'un des termes suivants : « Terminé », « Pas possible pour le moment », « En cours », « Bloqué » et « À faire ». Cela implique que, lorsqu'une mission est sélectionnée ou entamée par une personne, elle doit s'inscrire en tant que membre sur la carte correspondante et indiquer l'état d'avancement de la tâche pour éviter de refaire des tâches déjà réalisées.

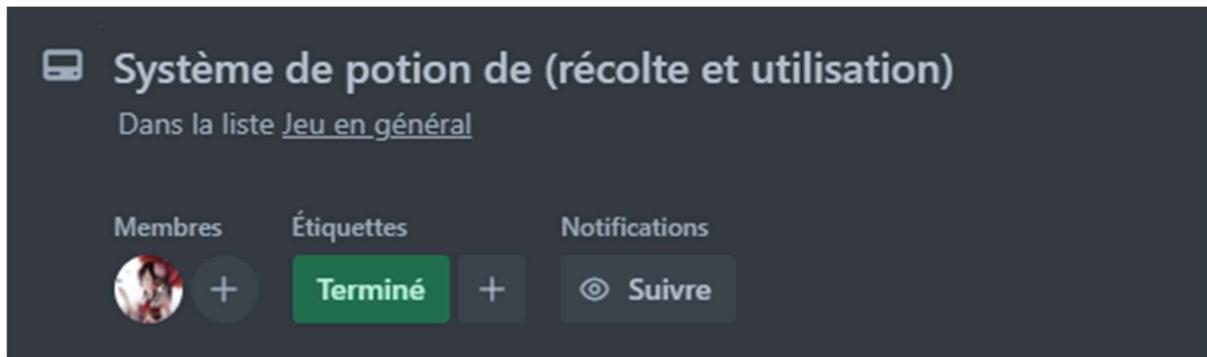


Figure 3: Exemple de tâche effectuée et labellisée « Terminé » (capture d'écran)

Rédigé par ZHANG Anxian et SELVARATNAM Akash

4 NOS DEUX APPLICATIONS

4.1 LES SERVICES PROPOSÉS (ARCHITECTURE FONCTIONNEL)

4.1.1 Site

Nos différentes pages web :

- ❖ **Accueil** : Introduction à l'histoire du jeu et présentation des personnages jouables.
- ❖ **À propos** : Présentation du projet, de l'utilité du site et de la gestion des données utilisateur.
- ❖ **Connexion** : Permet à l'utilisateur de se connecter pour accéder au jeu.
- ❖ **Inscription** : Pour les utilisateurs n'ayant pas encore de compte, cette page offre la possibilité de s'inscrire.
- ❖ **Oubli de mot de passe** : Permet de réinitialiser le mot de passe en cas d'oubli.
- ❖ **Modification des données utilisateur** : Offre la possibilité de modifier le pseudonyme, ainsi que le nom et le prénom associés au compte utilisateur.
- ❖ **Jeu** : Contient notre jeu Unity au format WebGL.
- ❖ **Paiement** : Propose deux modes de paiement différents (PayPal ou carte bancaire).
- ❖ **Contact** : Page dédiée pour toute demande d'aide de la part des utilisateurs.
- ❖ **Vérification** : Page permettant de vérifier le code saisi par l'utilisateur avec celui envoyé sur son e-mail lors de l'inscription ou d'une réinitialisation de mot de passe.

Les différentes librairies utilisées :

- ❖ **Cors** : Librairie permettant de personnaliser l'en-tête (origin, content-type, credentials) pour faciliter la connexion côté backend.
- ❖ **Express** : Framework conçu pour construire des applications web pour Node.js. Il permet l'exécution de requêtes HTTP (POST, GET, PUT, etc.).
- ❖ **Express-session** : Librairie offerte par Express, permettant de sauvegarder des données directement dans un cookie.
- ❖ **localStorage** : Librairie utilisée pour stocker des données localement sur le navigateur de l'utilisateur.
- ❖ **Mongoose** : Bibliothèque JavaScript facilitant la connexion entre MongoDB et une application codée en JavaScript. Elle permet de réaliser différentes opérations sur la base de données (Read, Create, Update, Delete).
- ❖ **Nodemailer** : Module de Node.js dédié à l'envoi d'e-mails.
- ❖ **paypal-rest-sdk** : Librairie fournie par PayPal pour intégrer des fonctionnalités de paiement via PayPal dans les applications, incluant un mode test.
- ❖ **Stripe** : Librairie développée par Stripe pour intégrer divers types de paiements, avec un mode test disponible.

Rédigé par ZHANG Anxian et SELVARATNAM Akash

4.1.2 Jeu

Nos différentes scènes :

- ❖ **Accueil** : Offre la possibilité de commencer une nouvelle partie ou de configurer les paramètres du jeu, tels que le mode plein écran ou les réglages du volume.
- ❖ **Sélection de personnages** : Permet de choisir entre deux classes disponibles, l'archer et le guerrier.
- ❖ **Village** : Zone non hostile où les joueurs peuvent améliorer leur arbre de compétences ou acheter des objets consommables, comme les potions.
- ❖ **Les niveaux de la première région :**
 - **Niveau 1** : Premier niveau du jeu.
 - **Niveau 2** : Deuxième niveau du jeu.

Rédigé par ZHANG Anxian et SELVARATNAM Akash

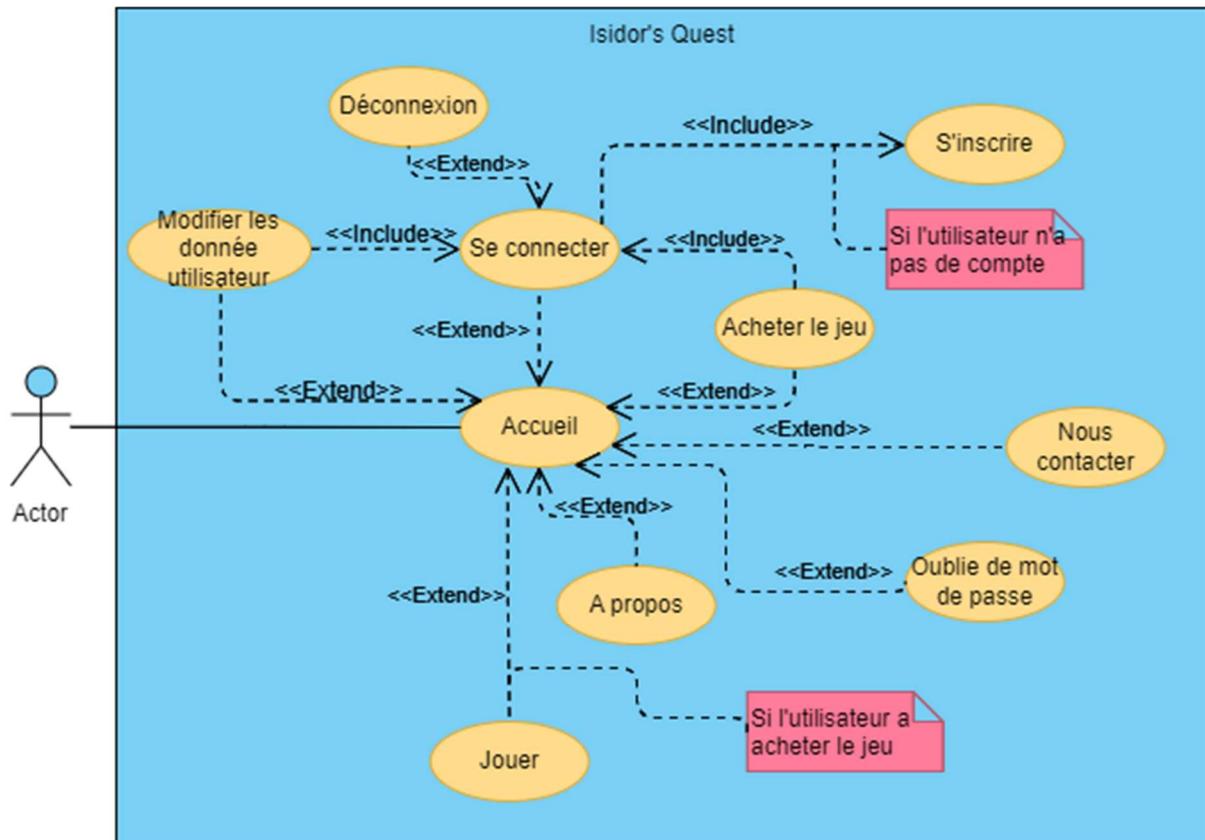
4.2 NOS PHASES DE DEVELOPPEMENT

Nous avons débuté par une phase de recherche et de définition des objectifs, documentée dans le premier rapport intitulé « Proposition de projet SAE S5 », disponible en annexe 7. Par la suite, nous avons rédigé le cahier des charges, y intégrant la planification du projet. Entre-temps, un membre de l'équipe a développé une page sur Figma pour esquisser une première représentation du site web. En ce qui concerne le jeu, certains membres se sont chargés de la conception de l'interface utilisateur (UI) et de l'expérience utilisateur (UX) des niveaux, ainsi que des menus, directement dans Unity. Enfin, après avoir finalisé la conception esthétique des applications, nous sommes entrés dans la phase de développement, suivie de tests unitaires effectués sur les composants du site avec la bibliothèque Jest.

Rédigé par ZHANG Anxian

4.3 ARCHITECTURE FONCTIONNELLE ET TECHNIQUE (DIAGRAMME UML)

4.3.1 Diagramme de cas d'utilisation de l'application



PS : L'UCD ici est basé sur l'accueil en tant que départ, cependant, ces actions peuvent aussi être effectuées à partir de toutes les autres pages.

Figure 4: Diagramme de cas d'utilisation de notre application

Créé par ZHANG Anxian

4.3.2 Architecture globale du projet

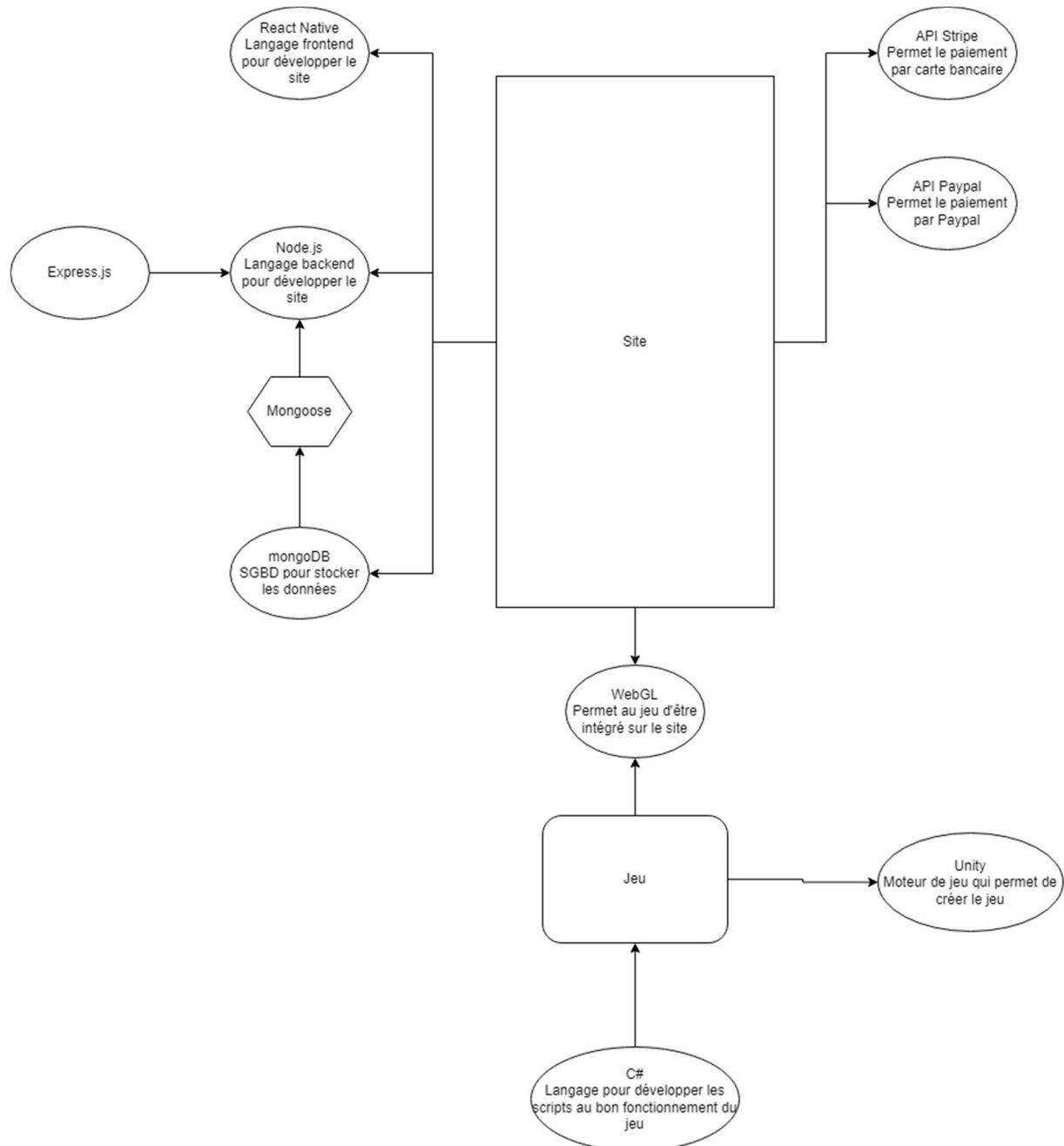


Figure 5 : Architecture technique globale du projet

Créée par COLLOMBET Nathan

4.3.3 Diagramme de séquence de la page oubli de mot de passe

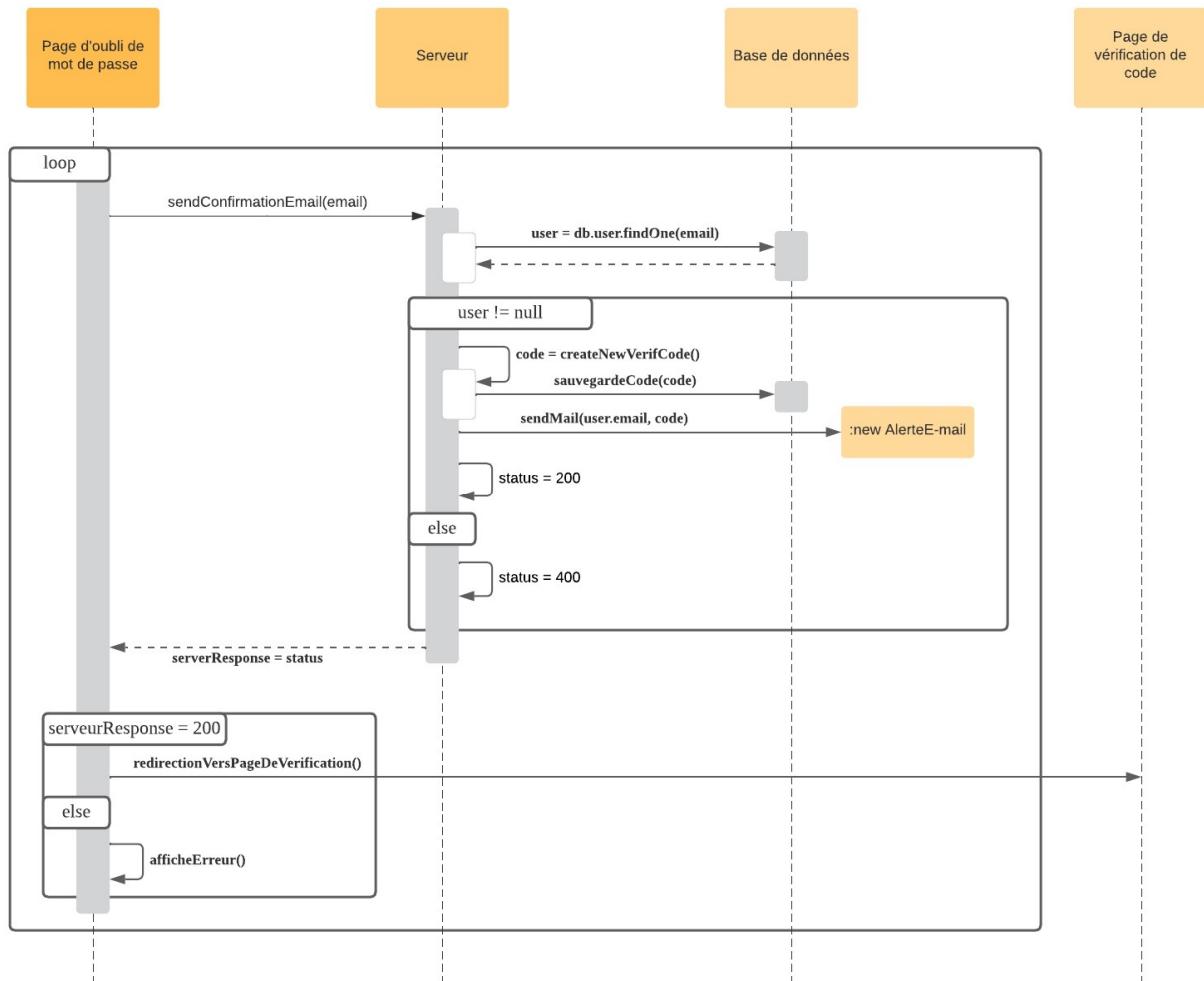


Figure 5: Diagramme de séquence de la page oubli de mot de passe

Créé par ZHANG Anxian

4.3.4 Diagramme de séquence de la page inscription

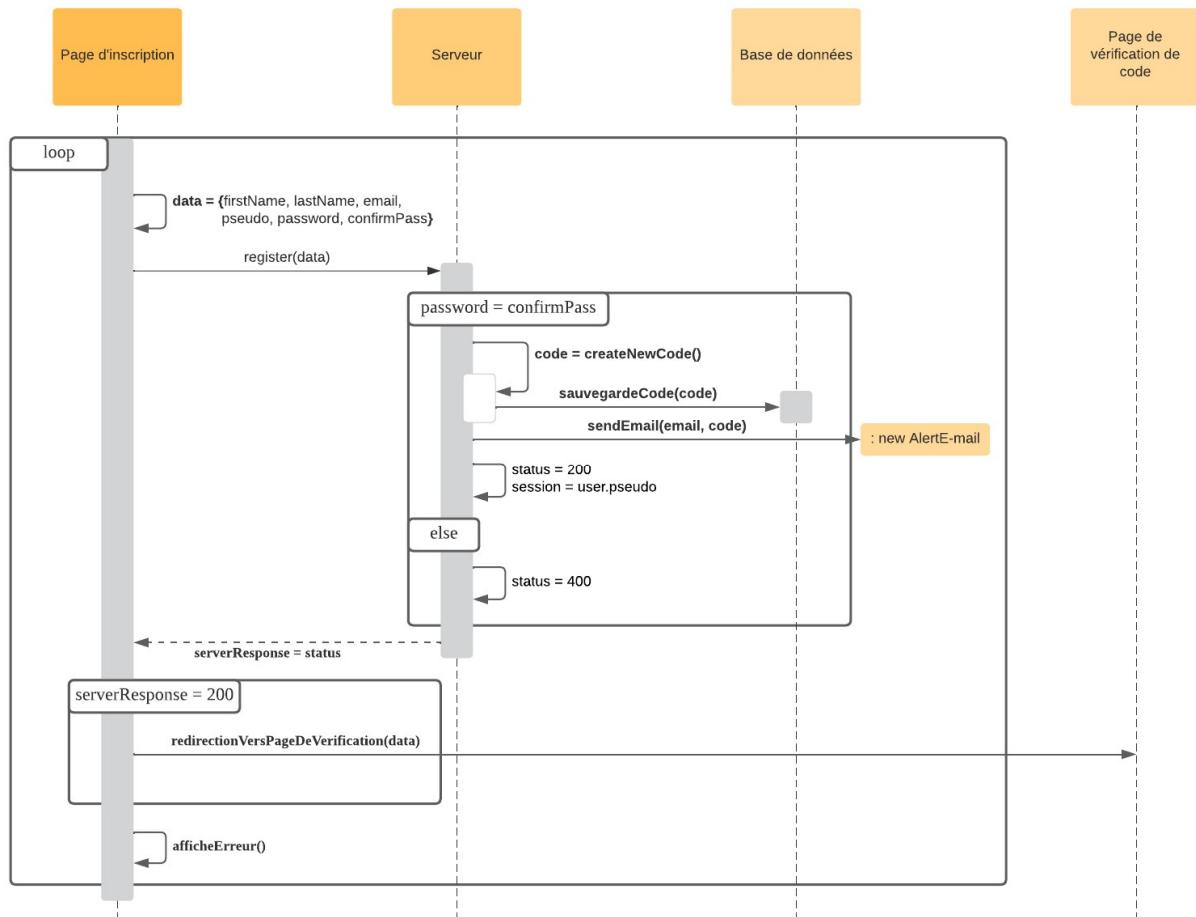


Figure 6: Diagramme de séquence de la page inscription

Créé par ZHANG Anxian

4.3.5 Diagramme de séquence de la page vérification de code

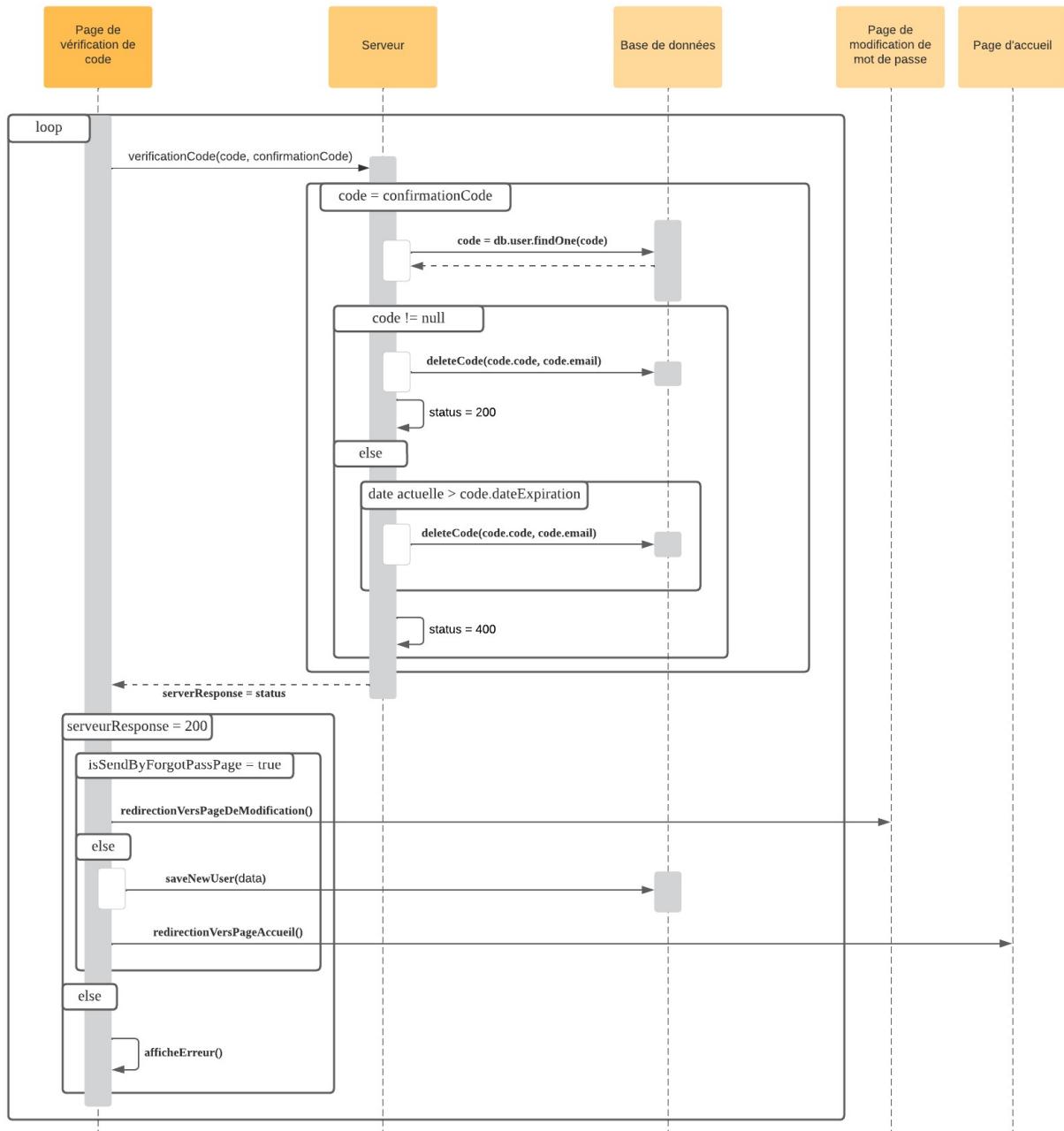


Figure 7: Diagramme de séquence de la page de vérification de code

Créé par ZHANG Anxian

4.3.6 Diagramme de séquence de la page de modification de mot de passe

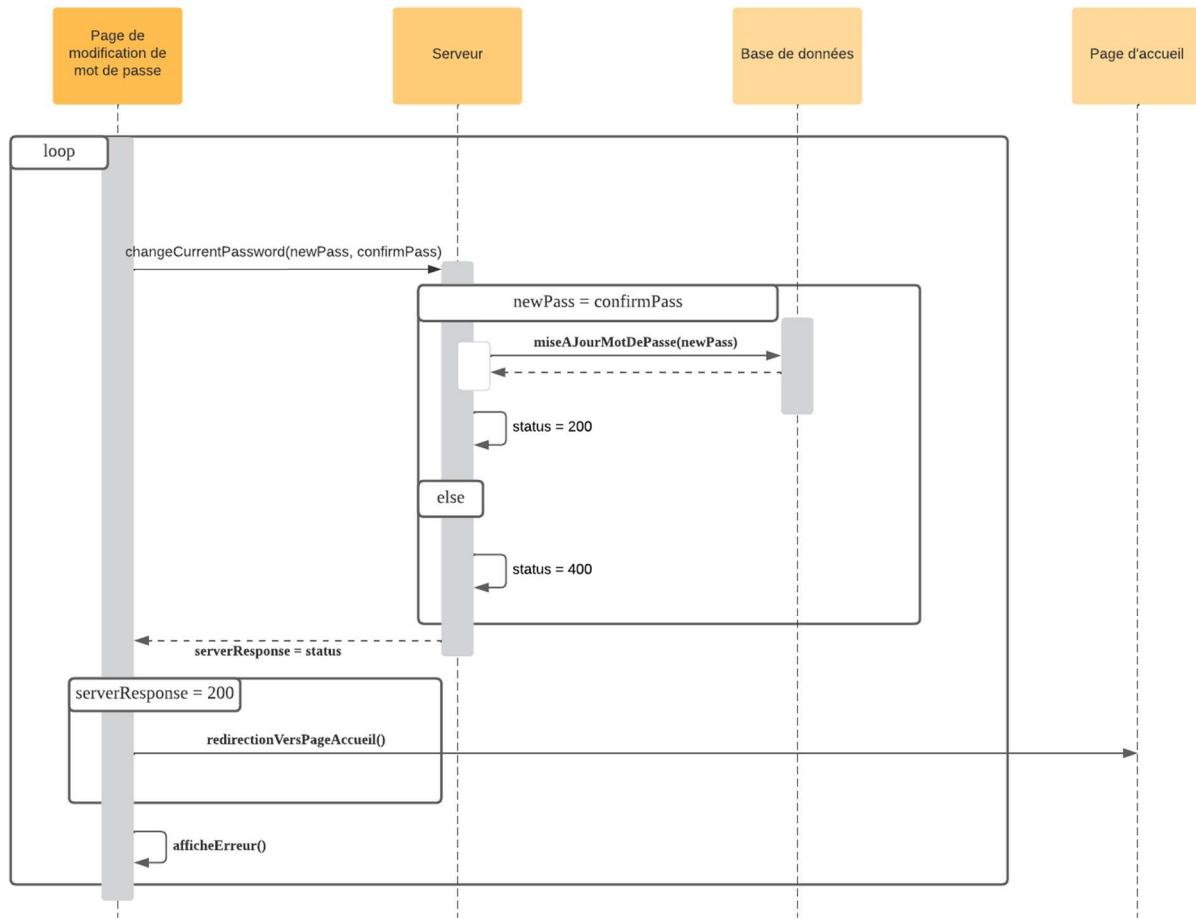


Figure 8: Diagramme de séquence de la page de modification de mot de passe

Créé par ZHANG Anxian

4.3.7 Diagramme de séquence de la page de modification des données utilisateur

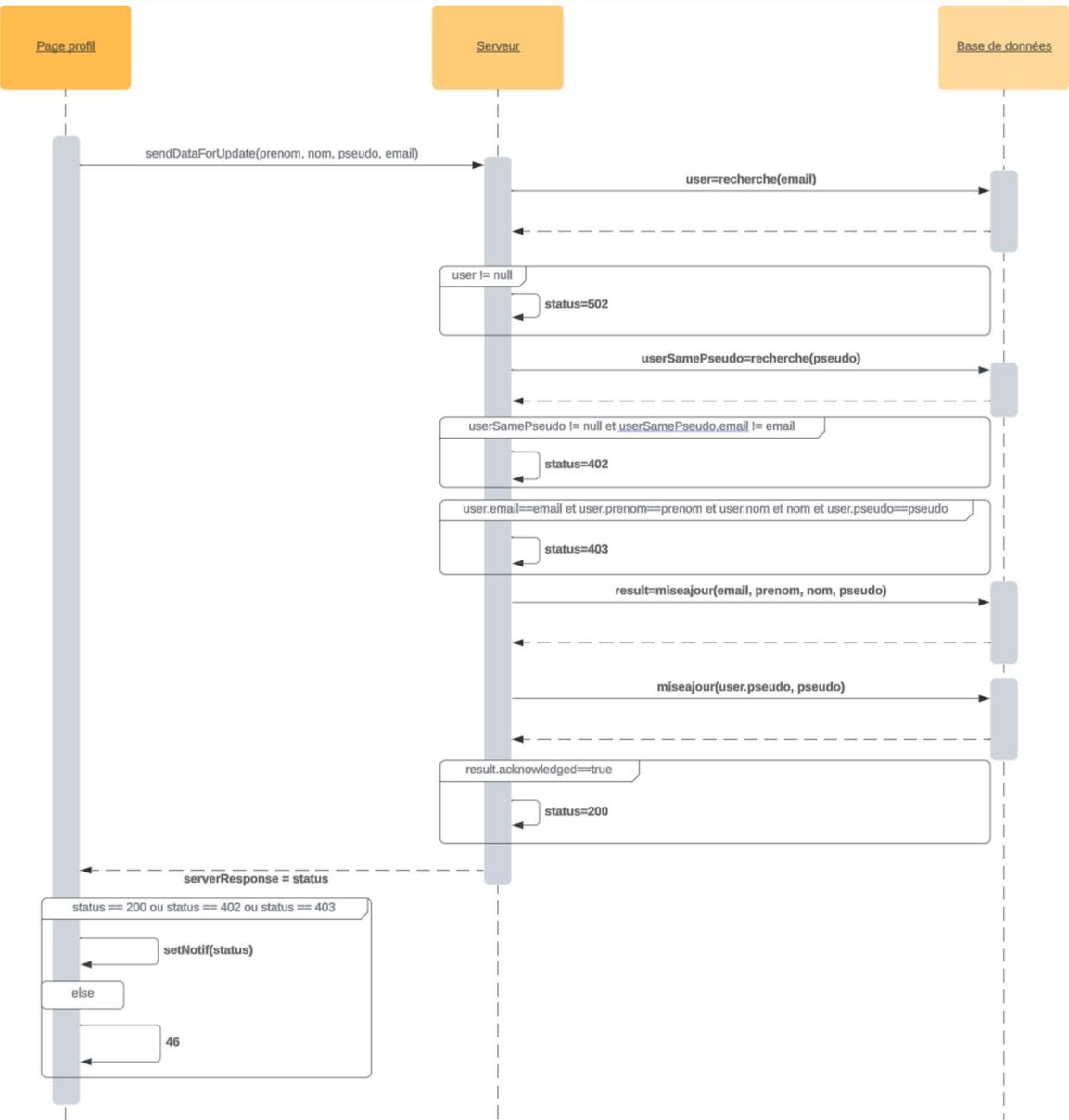


Figure 9: Diagramme de séquence de la page de modification des données utilisateur

Créé par LIN Xingtong

4.3.8 Diagramme de séquence de la page de connexion

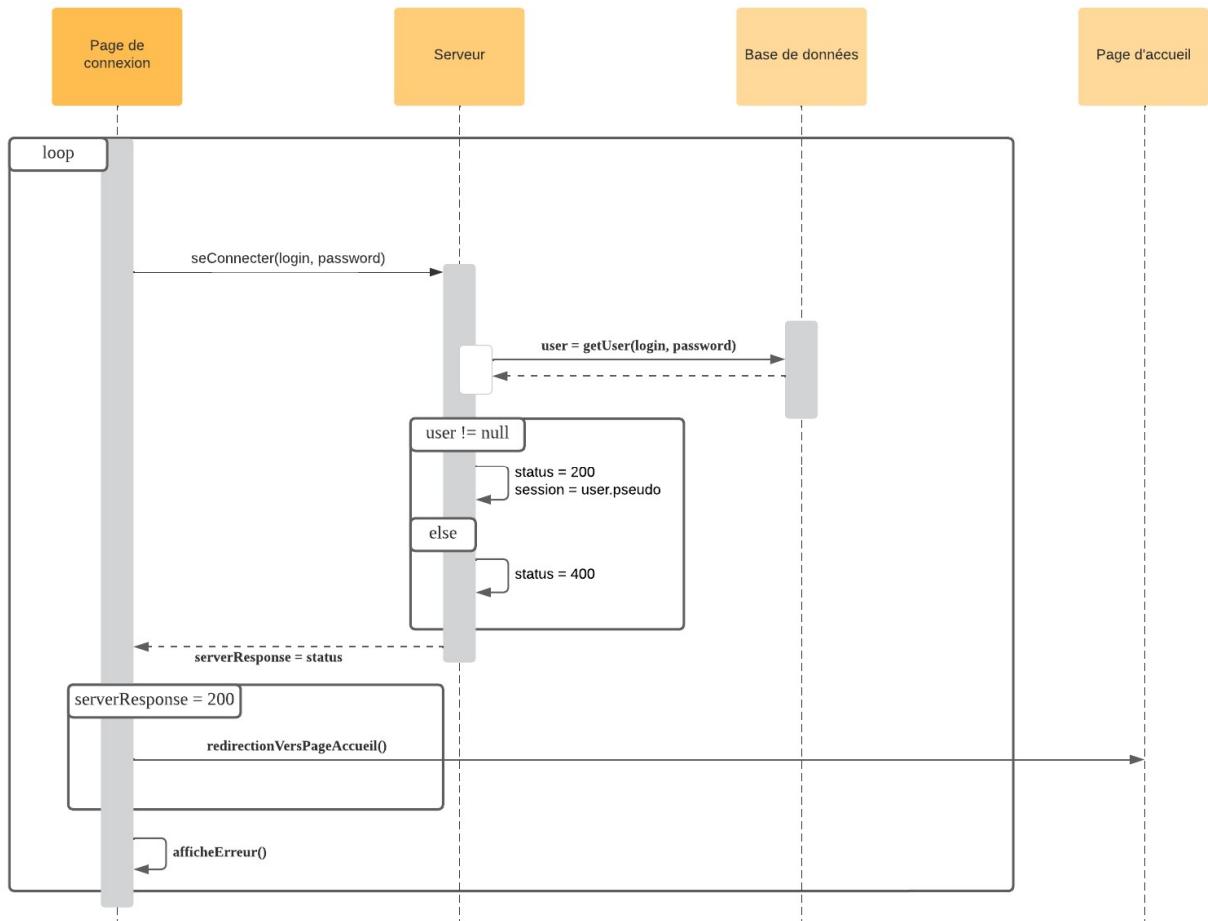


Figure 10: Diagramme de séquence de la page de connexion

Créé par ZHANG Anxian

4.3.9 Diagramme de séquence de la page de contact

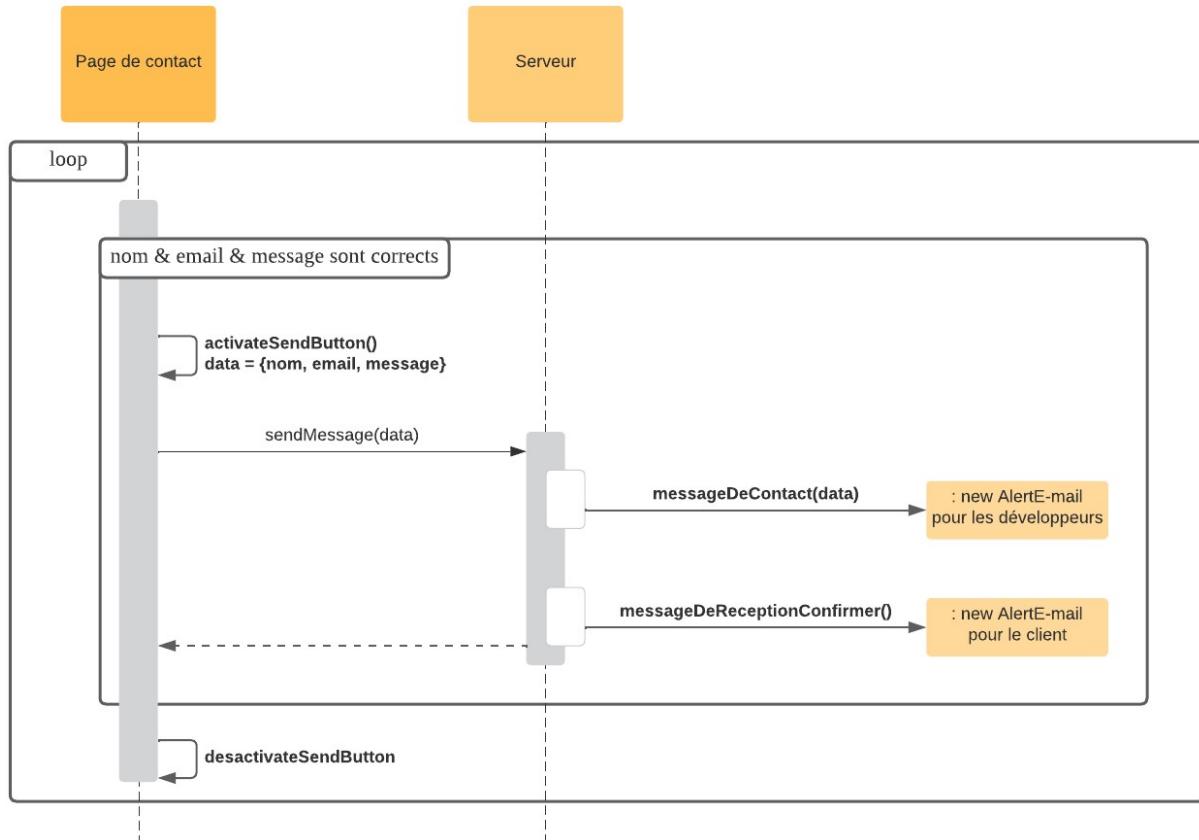


Figure 11: Diagramme de séquence de la page de contact

Créé par ZHANG Anxian

4.3.10 Les états et transitions du joueur

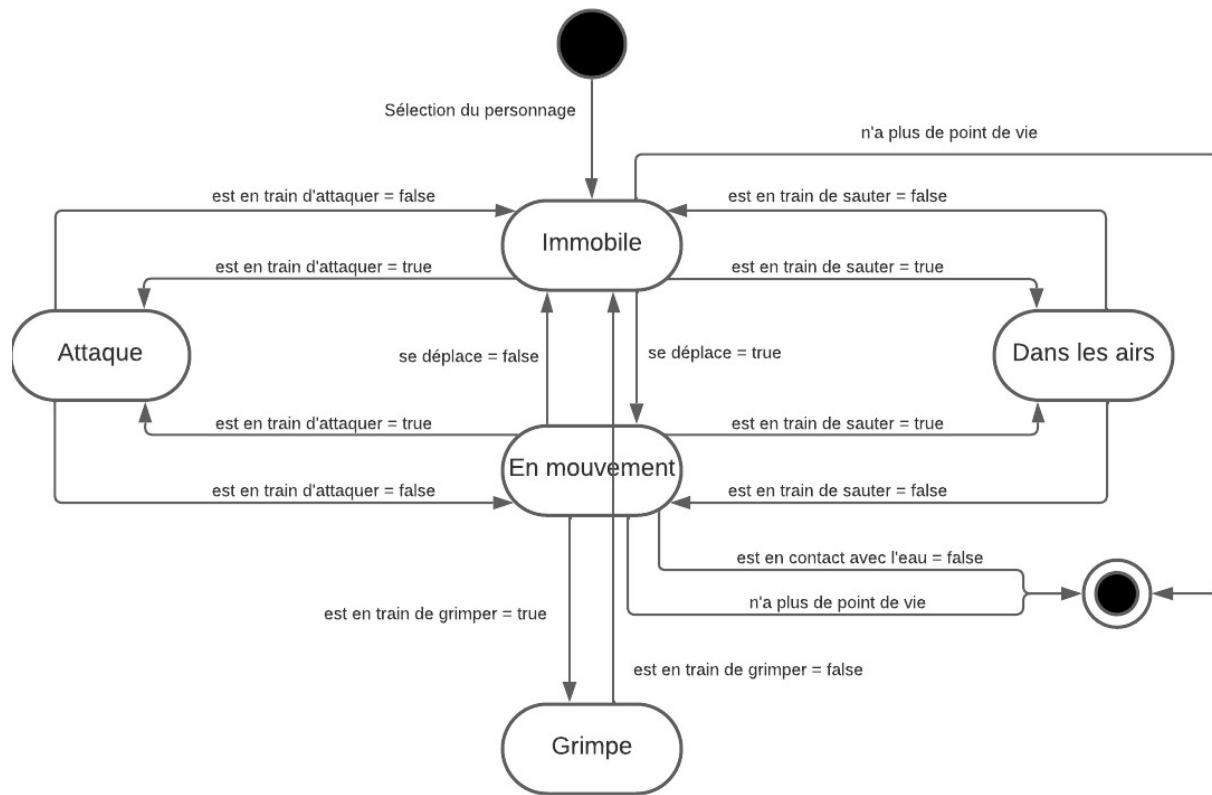


Figure 12: Diagramme d'état-transition du joueur

Créé par ZHANG Anxian

4.3.11 Les états et transitions d'un ennemi

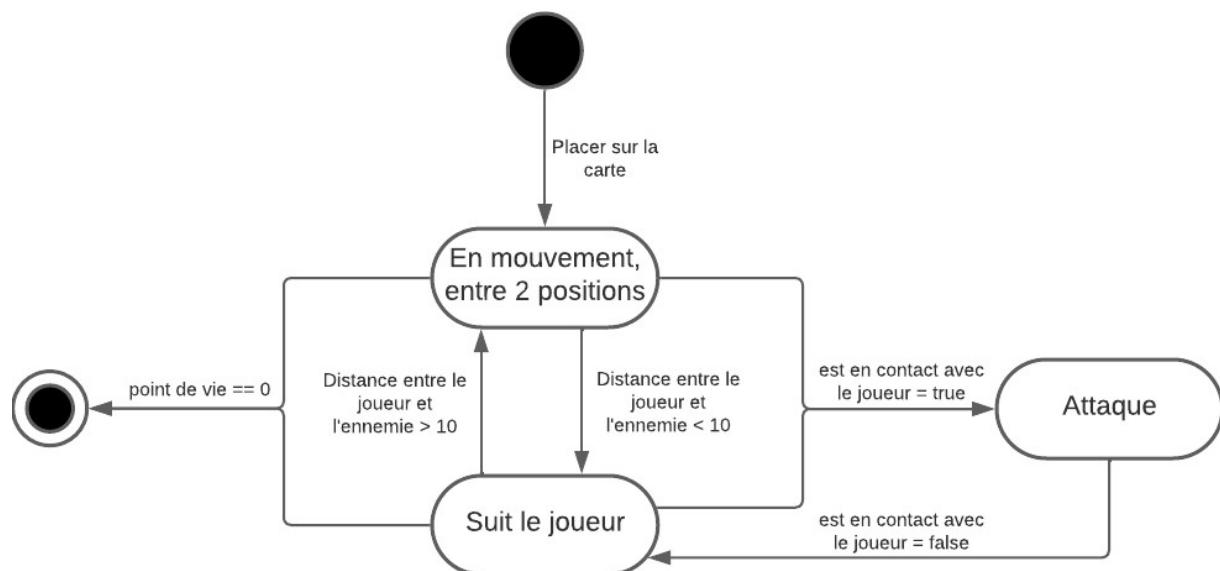


Figure 13: Diagramme d'état-transition des ennemis du jeu

Créé par ZHANG Anxian

4.4 CONCEPTION DES APPLICATIONS (EXPLICATIONS)

4.4.1 Site

4.4.1.1 Accueil

La page d'accueil, première interface visible par l'utilisateur, présente le contexte du jeu, les personnages et explique comment accéder au jeu. Les utilisateurs peuvent naviguer vers différentes sections à partir de cette page. Le fond de la page d'accueil est orné d'une image que nous avons créée, incluant des éléments du jeu. En ajustant les espacements entre chaque segment de texte et l'image de fond à l'aide d'une vue vide, nous avons réussi à créer une mise en page harmonieuse. Les boutons présents sur la page d'accueil, ainsi que ceux de l'en-tête, sont uniformisés, confectionnés avec *TouchableOpacity*.

Lorsqu'un utilisateur non connecté clique sur ces boutons, il est redirigé vers l'écran de connexion. Pour un utilisateur connecté mais n'ayant pas encore effectué de paiement, la redirection mène à l'écran de paiement. Finalement, seuls les utilisateurs connectés et ayant effectué le paiement sont redirigés vers l'écran de jeu.



Figure 14 : page d'accueil, un exemple après avoir ajusté la distance à l'aide d'une vue vide (capture d'écran)

Rédigé par LIN Xingtong

4.4.1.2 *Inscription*

Lors de l'inscription, l'utilisateur est invité à remplir un formulaire comprenant divers champs tels que le nom, le prénom, l'adresse mail, le nom d'utilisateur et le mot de passe. Si ces champs sont correctement remplis, l'utilisateur est alors redirigé vers la page de vérification de code. Dans le cas où l'adresse mail saisie existe déjà dans notre base de données, l'utilisateur sera invité à se connecter avec son compte existant plutôt que de s'inscrire. Si le pseudonyme choisi est déjà pris, l'utilisateur devra en sélectionner un autre, car nous avons opté pour des noms d'utilisateur uniques au sein de notre équipe. En outre, si le mot de passe saisi diffère de celui entré dans le champ de confirmation du mot de passe, un message d'erreur informera l'utilisateur que les deux mots de passe ne correspondent pas.

Cette page a été conçue en utilisant les composants TextInput pour les six champs d'entrée différents, un composant Text pour le titre « S'inscrire » et un TouchableOpacity pour le bouton. Nous utilisons également les hooks useState pour gérer les différents états des variables, ainsi que useEffect pour gérer l'état du bouton (activé si tous les champs sont remplis, sinon désactivé). Cette dernière fonction est aussi utilisée pour la vérification de l'adresse mail, qui doit comporter un texte avant et après le caractère @, suivi d'un point et de deux ou trois caractères. Cette vérification est réalisée à l'aide d'un regex.



Figure 15 : page d'inscription, exemple avec l'erreur nom d'utilisateur déjà pris (capture d'écran)

Rédigé par SELVARATNAM Akash

4.4.1.3 Vérification du code

Après avoir rempli les champs nécessaires à l'inscription, l'utilisateur sera redirigé vers une page dédiée à la vérification du code. Cette étape consiste à s'assurer que le code saisi par l'utilisateur correspond à celui envoyé à son adresse email. Cette procédure vise à confirmer l'existence de l'adresse mail renseignée dans le formulaire d'inscription. Pour la création de cette page, nous avons utilisé des composants TextInput pour les deux champs d'entrée, un composant Text pour le titre « Vérification du code » et un TouchableOpacity pour le bouton. Les hooks useState nous permettent de gérer les différents états des variables, tandis que useEffect est utilisé pour contrôler l'état du bouton.



Figure 16 : page de vérification du code (capture d'écran)

Le code est envoyé à l'adresse mail de l'utilisateur grâce à la bibliothèque Nodemailer de Node.js. Il s'agit d'un code composé de six caractères générés aléatoirement. Si l'utilisateur saisit le bon code, son compte sera créé dans le document User de notre base de données, en incluant les données qu'il a fournies (le mot de passe sera crypté dans la base de données). Ensuite, l'utilisateur sera redirigé vers la page d'accueil grâce à l'utilisation de useNavigation. Si le code saisi est incorrect, un message d'erreur s'affichera pour l'informer de cette erreur.

Code de vérification ➔ Boîte de réception x

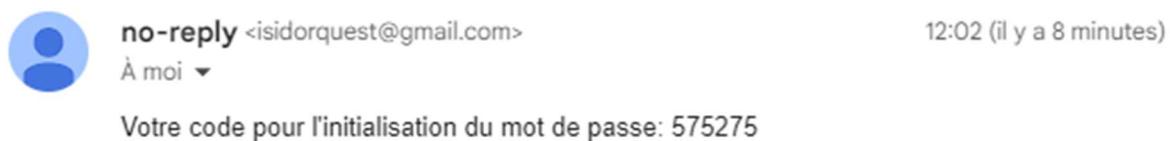


Figure 17 : Exemple de mail envoyé à un utilisateur avec le code à entrer

```
_id: ObjectId('656dd0fa19200d8755551317')
prenom: "a"
nomFamille: "a"
email: "a@apoa.aa"
pseudo: "pach"
isPay: false
password: "$2b$08$/ObCkLVQ0gMNMDxLunK8SeDiAd8hsAeSr5U/hppYnyEPQ6LnHLhta"
__v: 0

}

_id: ObjectId('656dd80fe8f576ae265ce485')
prenom: "bob"
nomFamille: "tom"
email: "am@ajhd.sdh"
pseudo: "ata"
isPay: false
password: "$2b$08$hCh0EYiTlyLhGd0YjhayoyeTT76ijwxExeokgGeMHkwpRXAxoe45CW"
__v: 0
```

Figure 18 : Exemple d'utilisateur dans le document User (capture d'écran)

Rédigé par ZHANG Anxian, SELVARATNAM Akash et COLLOMBET Nathan

4.4.1.4 Connexion

L'utilisateur est invité à saisir son nom d'utilisateur et son mot de passe. Si les données entrées sont correctes, il sera redirigé vers la page d'accueil. Dans le même temps, nous enregistrerons son pseudonyme dans les cookies au niveau du back-end à l'aide de la bibliothèque express-session, ce qui nous permettra de savoir si l'utilisateur est connecté. En cas d'erreur, si le nom d'utilisateur ou le mot de passe ne correspondent pas à ceux présents dans la base de données, un message d'erreur apparaîtra sur les deux champs. L'utilisateur devra alors soit remplir à nouveau le formulaire, soit s'inscrire via le bouton « Crée un nouveau compte », soit accéder à la page de réinitialisation du mot de passe via le lien « Mot de passe oublié ? ».

Cette page a été conçue avec des composants TextInput pour les deux champs d'entrée, un composant Text pour le titre « Connexion », et trois TouchableOpacity différents pour les boutons et le lien vers la page de réinitialisation du mot de passe. Nous utilisons aussi les hooks useState pour gérer les différents états des variables et useEffect pour contrôler l'état du bouton « Se connecter ».

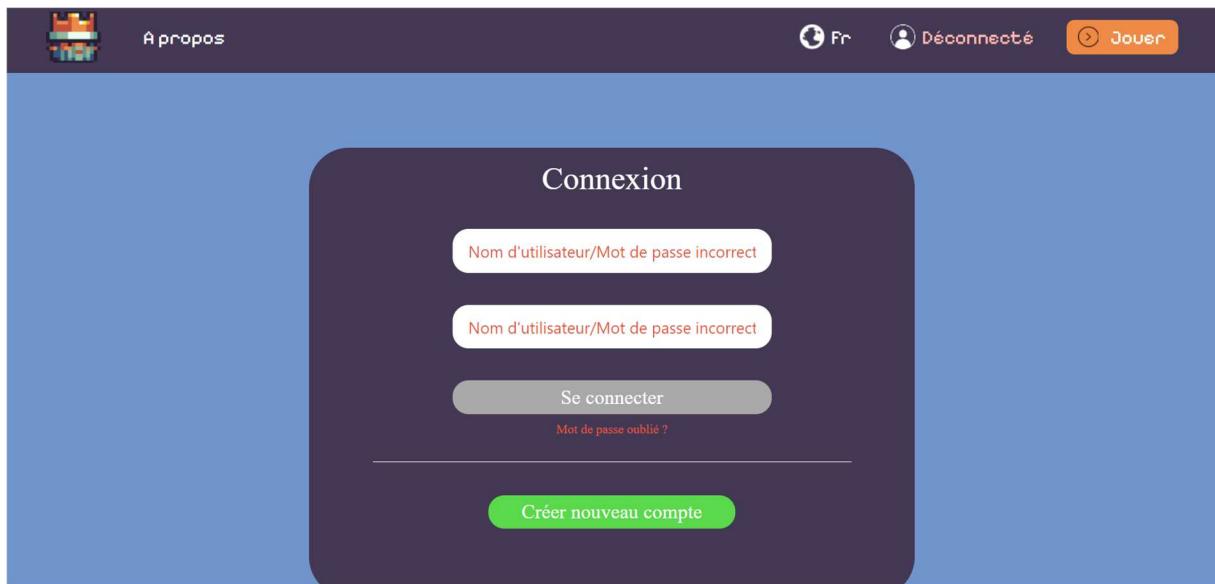


Figure 19 : Page de connexion, exemple de connexion avec un nom d'utilisateur ou un mot de passe incorrect (capture d'écran)

Rédigé par SELVARATNAM Akash

4.4.1.5 Oublie de mot passe

Initialement, l'utilisateur est invité à entrer son adresse électronique. Cette dernière doit être valide, comprenant un arrobase et un nom de domaine. Tant que l'adresse n'est pas correctement saisie, le bouton « Obtenir le code de vérification » demeurera désactivé. Cette fonctionnalité a été mise en œuvre à l'aide d'un `useEffect`, qui réalise une vérification à chaque mise à jour du `TextInput` par l'utilisateur.



Figure 20: page d'oubli de mot de passe, exemple d'un mail non valide (capture d'écran)

Lorsque l'utilisateur appuie sur le bouton, il sera redirigé vers la page de vérification du code qui lui aura été envoyé par email. Après avoir réussi cette vérification, il aura la possibilité de modifier son mot de passe. Ce changement sera géré au niveau du backend : en effet, lorsque le bouton de modification est pressé, une requête HTTP sera envoyée au serveur pour mettre à jour les informations dans la base de données.

Rédigé par ZHANG Anxian

4.4.1.6 A propos

La page « À propos » détaille le contexte du projet, l'utilité du site ainsi que la gestion des données utilisateur. Pour sa réalisation, nous avons employé divers composants React, notamment le composant Text pour présenter notre projet. Nous avons aussi créé notre propre composant afin de construire la table des matières, nécessitant l'usage du composant TouchableOpacity pour les liens. De plus, le hook useRef a été utilisé pour naviguer vers le bloc correspondant dans la page lorsqu'on clique sur un élément de la table des matières.



Figure 21 : Page « A propos » (capture d'écran)

Rédigé par SELVARATNAM Akash

4.4.1.7 Contact

L'utilisateur est invité à remplir les différents champs du formulaire de contact, incluant son adresse mail, son nom, ainsi que le message à envoyer à notre service. Pour développer cette page, nous avons utilisé divers hooks, comme le useState pour gérer les valeurs des différentes variables, et un useEffect pour contrôler l'état du bouton d'envoi. Nous avons également employé le composant Text pour le titre « Formulaire de contact », des composants TextInput pour les champs d'entrée, et un TouchableOpacity pour le bouton d'envoi.



Figure 22 : Page de contact (capture d'écran)

Lorsque l'utilisateur remplit tous les champs du formulaire et appuie sur le bouton "Envoyer", un email est envoyé à notre service contenant le message de l'utilisateur. Ceci est réalisé à l'aide de la bibliothèque Nodemailer. Parallèlement, un email automatique est envoyé à l'adresse indiquée par l'utilisateur sur le formulaire, confirmant ainsi la bonne réception de son message.

Confirmation du message pour Isidor's Team



Figure 23: Mail de confirmation envoyé à l'utilisateur (capture d'écran)

Rédigé par SELVARATNAM Akash et COLLOMBET Nathan

4.4.1.8 Paiement

Afin d'offrir la possibilité d'acheter notre jeu, nous avons mis en place deux moyens de paiement distincts : par carte bancaire ou via PayPal. Pour concevoir cette page, nous avons utilisé deux composants TouchableOpacity, chacun correspondant à l'un des deux modes de paiement proposés.

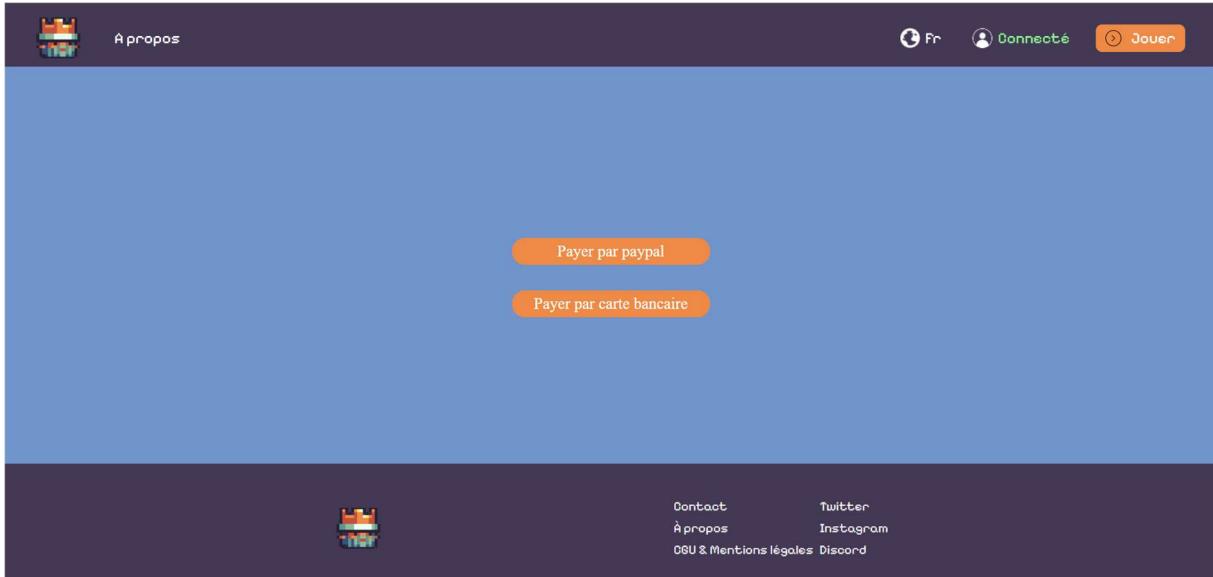
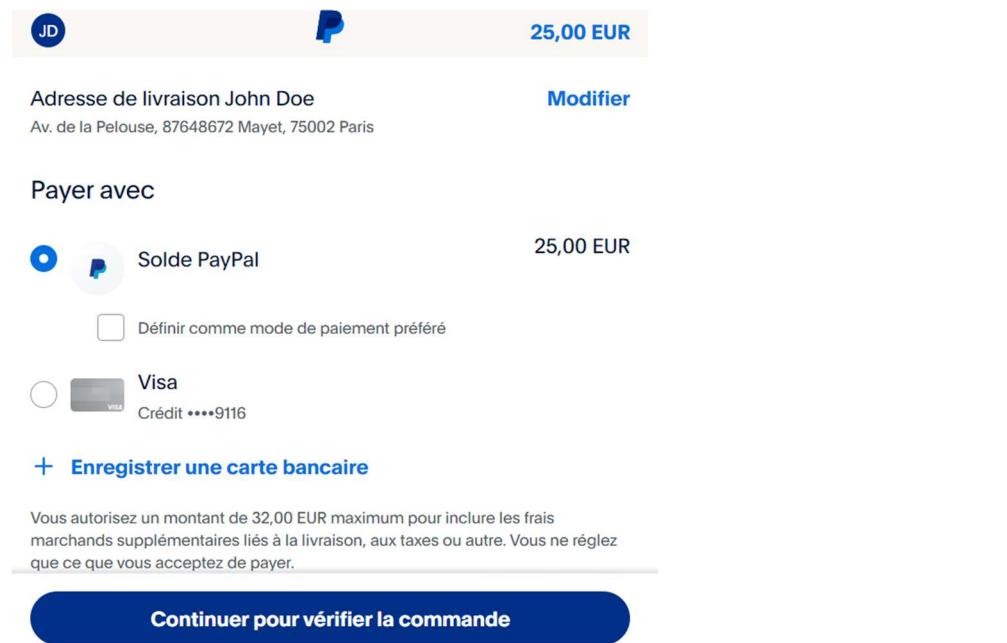


Figure 24 : Page de paiement (capture d'écran)

Lorsque l'utilisateur appuie sur le bouton « Payer par PayPal », une requête est envoyée à l'API de PayPal via la bibliothèque `paypal-rest-sdk`. Cette requête inclut le montant de l'achat et les URL de redirection après l'achat, que ce soit en cas de succès ou d'échec du paiement. L'utilisateur est ensuite redirigé vers un lien où il peut se connecter à son compte PayPal et valider la transaction, qui est entièrement gérée par PayPal. Après un achat réussi, un mail de confirmation d'achat est envoyé à l'adresse associée au compte PayPal de l'utilisateur, ainsi qu'à notre compte marchand pour confirmer la réception de la transaction. La bibliothèque `paypal-rest-sdk` doit être initialisée avec des identifiants spécifiques à notre compte PayPal pour établir la connexion.

Lorsque l'utilisateur choisit « Payer par carte bancaire », une requête est envoyée à l'API de Stripe à l'aide de la bibliothèque Stripe, nécessitant une clé associée au compte marchand Stripe. Cette requête comprend la liste des objets à vendre, préalablement créés sur le compte Stripe dont nous utilisons les identifiants pour les afficher sur la page de paiement. Comme pour PayPal, les URL de redirection en cas de réussite ou d'échec du paiement sont intégrées à la requête. Une fois la requête complétée, l'identifiant de l'achat est extrait pour une éventuelle utilisation ultérieure, et l'utilisateur est immédiatement redirigé vers le lien de paiement après avoir cliqué sur le bouton.



JD  25,00 EUR

Adresse de livraison John Doe [Modifier](#)

Av. de la Pelouse, 87648672 Mayet, 75002 Paris

Payer avec

| | |
|--|-----------------|
|  Solde PayPal | 25,00 EUR |
| <input type="checkbox"/> Définir comme mode de paiement préféré | |
|  Visa | Crédit ****9116 |

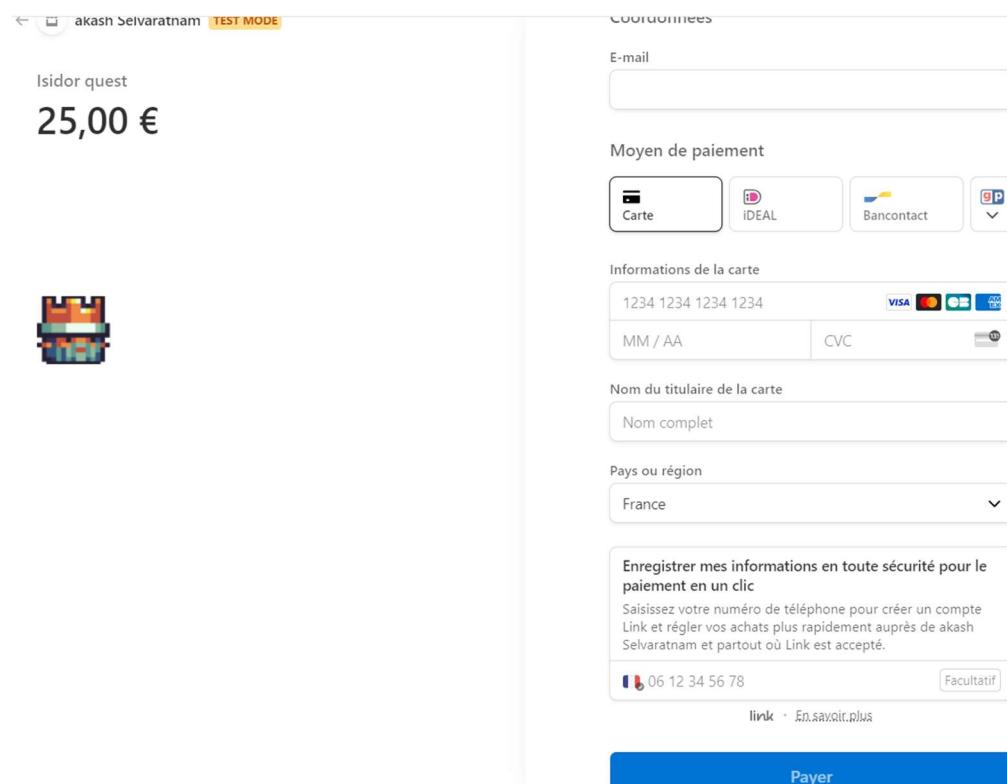
[+ Enregistrer une carte bancaire](#)

Vous autorisez un montant de 32,00 EUR maximum pour inclure les frais marchands supplémentaires liés à la livraison, aux taxes ou autre. Vous ne réglez que ce que vous acceptez de payer.

Continuer pour vérifier la commande

[Annuler et retourner sur Test Store](#)

Figure 25 : Paiement par PayPal (capture d'écran)



akash Selvaratnam TEST MODE

Isidor quest

25,00 €

COORDONNÉES

E-mail

Moyen de paiement

| | | | |
|---|--|--|---|
|  Carte |  IDEAL |  Bancontact |  |
|---|--|--|---|

Informations de la carte

| | |
|---------------------|---|
| 1234 1234 1234 1234 |    |
| MM / AA | CVC |

Nom du titulaire de la carte

Nom complet

Pays ou région

France

Enregistrer mes informations en toute sécurité pour le paiement en un clic

Saisissez votre numéro de téléphone pour créer un compte Link et régler vos achats plus rapidement auprès de akash Selvaratnam et partout où Link est accepté.

| | |
|--|------------|
|  06 12 34 56 78 | Facultatif |
|--|------------|

[link](#) • [En savoir plus](#)

Payer

Figure 26: Paiement par carte bancaire (capture d'écran)

Pour la gestion de la facture chez Stripe, une fois redirigé vers la page de confirmation, nous utiliserons l'identifiant de l'achat précédemment stocké pour récupérer les informations qui lui sont associées. Ceci se fait via une nouvelle requête API, où l'identifiant de l'achat est le seul paramètre requis. En récupérant l'identifiant de la facture à partir de cette requête, nous utiliserons ce dernier pour effectuer une nouvelle requête afin d'obtenir des informations détaillées sur la facture. Nous obtenons ainsi un lien que nous associerons à un bouton sur lequel l'utilisateur pourra cliquer pour être redirigé vers une page Stripe. Sur cette page, il aura la possibilité de télécharger sa facture.

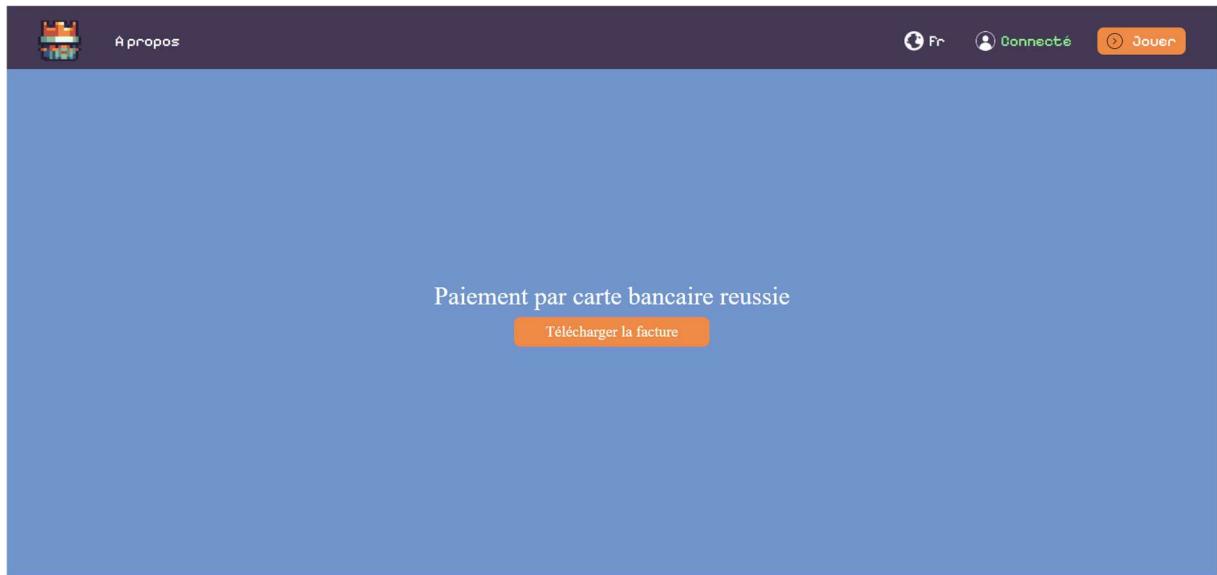


Figure 27 : Paiement par carte bancaire réussie (capture d'écran)

Pour les deux différents moyens de paiement, il s'agit de paiement de test donc les paiements ne sont pas réellement effectués.

Données test pour le paiement par Paypal :

- Email : sb-lfkvx26095575@business.example.com
- Mot de passe : ztY=j8Q&

Données test pour le paiement par Stripe :

- Le numéro de la carte bancaire : 4242 4242 4242 4242
- Date d'expiration : une date supérieure à la date actuelle
- CVC : écrivez un code à trois chiffres. (Exemple : 123)

Rédigé par LIN Oscar

4.4.1.9 Modification des données utilisateur

Cette page est conçue pour permettre aux utilisateurs de consulter et de modifier leurs informations personnelles, telles que l'email, le prénom, le nom et le pseudonyme. Elle comprend trois composants TextInput, chacun destiné à un champ d'entrée différent, afin que l'utilisateur puisse à la fois consulter et modifier ses données. On y trouve également des composants Text pour le titre « Informations personnelles » et pour afficher l'adresse mail, ainsi qu'un TouchableOpacity pour le bouton de modification. De plus, cette page utilise des hooks, notamment des useState pour gérer les valeurs des variables, et des useEffect pour appeler la méthode côté backend. Cette méthode, réalisée à l'aide de la bibliothèque Mongoose, permet de modifier les données de l'utilisateur dans le document User de notre base de données.

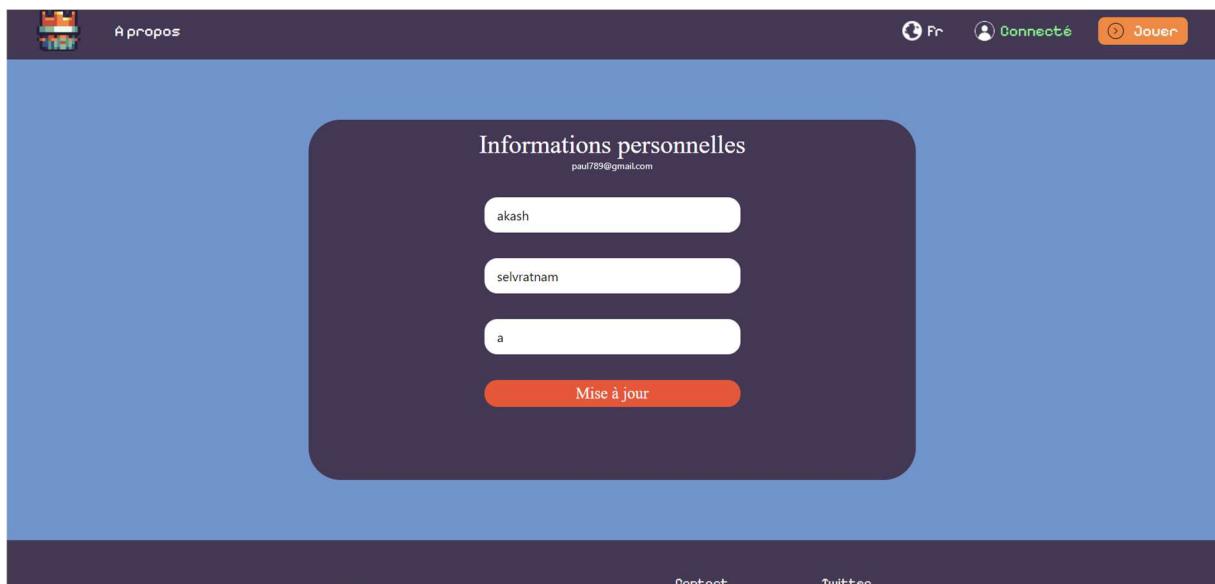


Figure 28: Page de consultation/modification des données (capture d'écran)

Pour initier la modification des données, l'utilisateur doit appuyer sur le bouton « Mise à jour ». Il peut alors éditer les différents champs d'entrée. Une fois les modifications terminées, en appuyant sur le bouton « Valider », une requête est envoyée côté backend pour actualiser les données de l'utilisateur dans le document User de notre base de données.



Figure 29 : Modification des données (capture d'écran)

Rédigé par SELVARATNAM Akash

4.4.1.10 Header (composant commun à toutes les pages)

Le header de notre site web, présent sur toutes les pages, inclut notre logo, conçu à l'aide du composant TouchableOpacity. Lorsqu'on appuie sur ce logo, l'utilisateur est redirigé vers la page d'accueil. De plus, le lien menant à la page « À propos » est également réalisé avec le composant TouchableOpacity.



Figure 30: Header (capture d'écran)

Quand on appuie sur le bouton "Fr", une liste déroulante apparaît, offrant le choix entre les langues Français et Anglais. Ce sont les deux options disponibles pour la traduction de notre site web. Pour implémenter cette fonctionnalité, nous avons créé deux fichiers JSON, l'un pour le Français et l'autre pour l'Anglais. Chaque texte sur notre site est référencé dans ces fichiers JSON. Lorsque l'utilisateur sélectionne la langue française, une mise à jour de la langue est effectuée sur le site web via un useState qui prend alors comme valeur le fichier JSON correspondant. Cette liste déroulante est réalisée avec le composant SelectDropdown de React-Native.

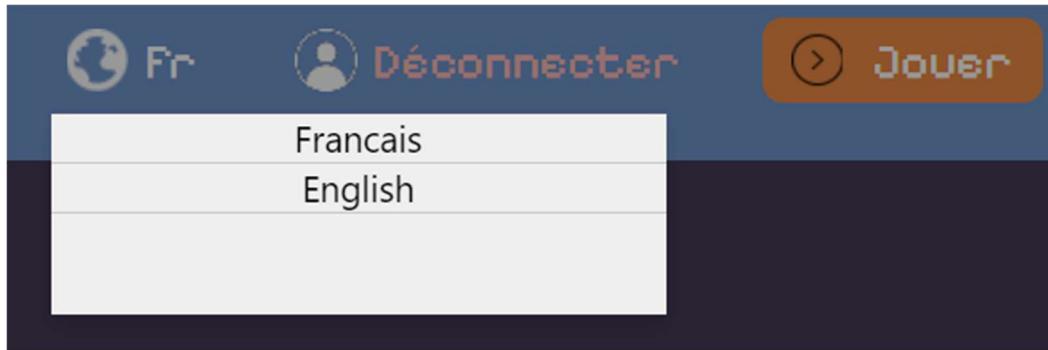


Figure 31: Liste des langues disponibles sur notre site web (capture d'écran)

```
"Home" : {
    "buttonPlay" : "Jouer",
    "gameTitle" : "Isidor's Quest : le jeu fantastique de plateforme",
    "gameDescPartOne" : "Cher joueur, dans un lointain royaume, règne",
    "gameDescPartTwo" : "De nombreux aventuriers ont été malencontreusement",
    "playerTitle" : "Incarne l'une des 2 classes",
}
```

Figure 32: Fichier JSON Français (capture d'écran)

```
"Home": {
    "buttonPlay": "Play",
    "gameTitle": "Isidor's Quest: The Fantasy Platformer",
    "gameDescPartOne": "Dear Gambler, in a distant realm, there reigns a powerful god respected by",
    "gameDescPartTwo": "Many adventurers have been inadvertently sucked into his kingdom, but none",
    "playerTitle": "Embody one of the 2 classes",
```

Figure 33 : Fichier JSON Anglais (capture d'écran)

Le bouton de connexion/déconnexion affiche une liste qui propose l'option de connexion lorsque l'utilisateur est déconnecté, et de déconnexion lorsqu'il est connecté. Ce bouton permet également d'accéder à la page de profil, où l'utilisateur peut consulter ou modifier ses données. Cette liste déroulante a été réalisée avec le composant SelectDropdown de React-Native.

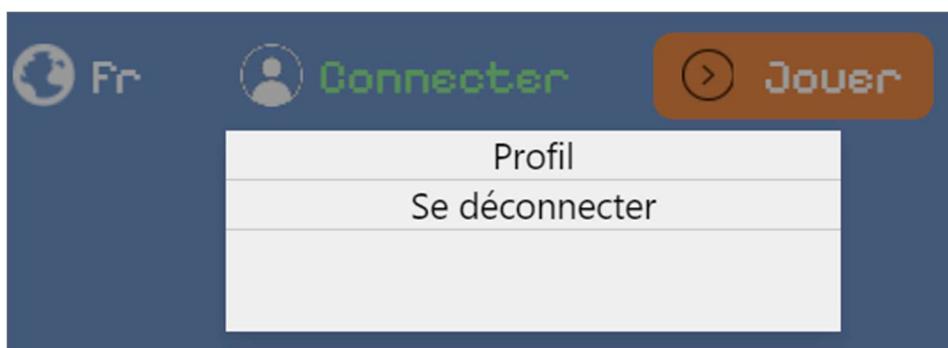


Figure 34: Connexion (Header, capture d'écran)

Pour conclure, le bouton "Jouer" dirige l'utilisateur vers la page de connexion s'il n'est pas connecté, ou vers la page de paiement s'il est connecté mais n'a pas encore acheté le jeu. Si l'utilisateur est connecté et a déjà acheté le jeu, il sera alors redirigé vers la page de jeu. Ce bouton a été conçu à l'aide du composant TouchableOpacity.

Rédigé par SELVARATNAM Akash et COLLOMBET Nathan

4.4.1.11 Footer (composant commun à toutes les pages)

Le Footer, présent sur toutes les pages du site web, comprend notre logo, qui redirige vers la page d'accueil lorsqu'on clique dessus. Il a été conçu en utilisant les composants Image et TouchableOpacity. De plus, nous avons intégré plusieurs liens, également réalisés avec le composant TouchableOpacity, qui permettent de naviguer vers différentes pages telles que le contact, à propos, les mentions légales, ainsi que les différents réseaux sociaux associés au jeu.



Figure 35 : Footer (capture d'écran)

Rédigé par SELVARATNAM Akash

4.4.2 Jeu

4.4.2.1 Écran d'accueil

Notre écran d'accueil est constitué de plusieurs panneaux, créant ainsi différentes couches d'interface utilisateur (UI) qui se superposent. La couche située le plus en arrière est l'image de fond, visible sur l'illustration ci-dessous. Au-dessus de cette image, nous avons placé le titre du jeu, ainsi que les boutons « Start Game » (Démarrer le jeu) et « Settings » (Paramètres).



Figure 36 : menu d'accueil (capture d'écran)

La fonctionnalité du bouton « Start Game » a été implémentée en lui associant un événement. Tout comme en JavaScript, le C# permet d'ajouter des événements avec la méthode `onClick.AddListener(maFonction)`. Lorsque le bouton est cliqué, le jeu charge la scène de sélection de personnage. Cette action est rendue possible grâce à la ligne de code `SceneManager.LoadScene(maScèneACharger)` présente dans `maFonction`.

L'implémentation de l'événement onClick sur le bouton « Settings » a été réalisée de la même manière que pour le bouton « Start Game ». Cependant, dans ce cas, au lieu de charger une scène, un clic sur le bouton « Settings » ouvre un menu, comme illustré dans la figure ci-dessous.

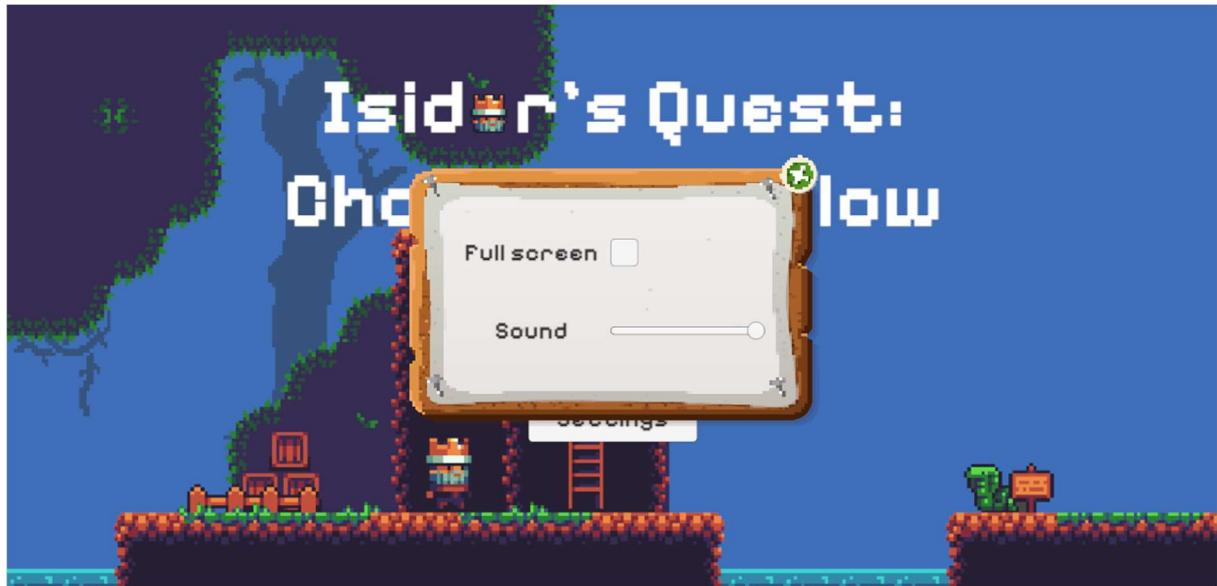


Figure 37: menu de configuration (capture d'écran)

La fonctionnalité du mode plein écran a été implémentée en ajoutant un événement de type OnValueChanged(maFonction). La fonction passée en paramètre, ici maFonction, exécute l'opération suivante : Screen.fullScreen = étatDeLaCase, ce qui permet de basculer en mode plein écran ou d'en sortir à chaque clic sur la case correspondante. De manière similaire, le volume est ajusté en fonction de la valeur entière fournie (comprise entre -80 et 0) en fonction de la position de la barre de volume.

Rédigé par ZHANG Anxian

4.4.2.2 Scène de sélection des personnages

La scène de sélection de personnages offre la possibilité de choisir entre deux classes disponibles : le guerrier et l'archer. Pour chacun de ces personnages, nous affichons l'ensemble de leurs statistiques (défense, attaque, vitesse, points de vie) en utilisant une fonction qui récupère les statistiques initiales de la classe concernée à partir d'un fichier JSON. Afin de présenter les deux classes différentes, nous avons créé un tableau contenant les sprites du guerrier et de l'archer.

Lorsque le joueur appuie sur une flèche, une fonction est exécutée pour sélectionner le personnage approprié en fonction de l'indice du tableau et des statistiques de la classe concernée. Une fois que le joueur appuie sur le bouton « Select », il est dirigé vers la scène du niveau 1 avec le personnage qu'il a choisi.

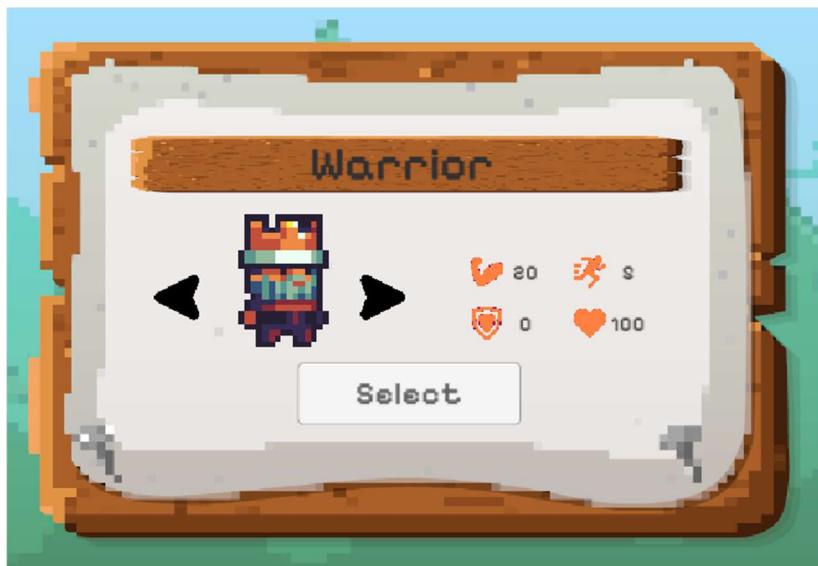


Figure 38 : Scène de sélection de personnage (Capture d'écran)

Rédigé par SELVARATNAM Akash

4.4.2.3 Fonctionnalités dans l'ensemble des niveaux

- ❖ Barre de vie (Joueur et ennemis)

La barre de vie du joueur et des ennemis est représentée par une barre de couleur verte dotée d'un attribut nommé « fill amount ». Cet attribut gère le taux de remplissage de la barre. Par exemple, si l'attribut « fill amount » est réglé à 0.5, alors la barre sera remplie à 50%.



Figure 39 : Barre de vie du joueur à 50% et barre de vie de l'ennemi à 100%

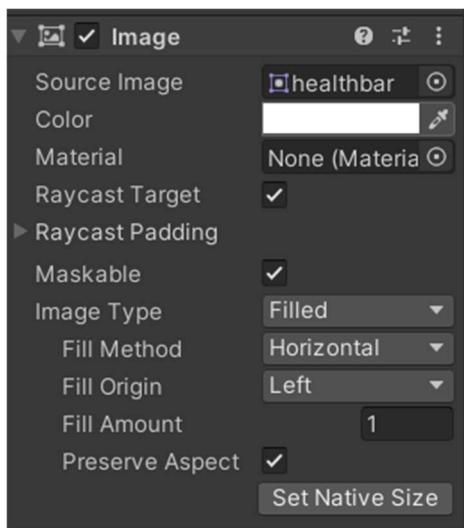


Figure 40 : Les attributs de l'image de la barre de vie

Pour calculer le taux de remplissage de la barre de vie du joueur ou de l'ennemi, nous utilisons la formule suivante :

- Points de vie maximaux du joueur ou de l'ennemi : 60
- Points de vie actuels du joueur ou de l'ennemi : 35

Nous convertissons d'abord les points de vie maximaux en un format décimal (pourcentage) : $60 * 0.01 = 0.6$, ce qui représente les points de vie maximaux sous forme de pourcentage.

Ensuite, nous calculons les points de vie actuels en pourcentage : $35 / 0.6 = 58.33$, ce qui indique le pourcentage actuel des points de vie.

Nous convertissons ensuite ce pourcentage en une valeur décimale : $58.33 * 0.01 = 0.5833$, ce qui représente les points de vie actuels sous forme décimale (pourcentage).

Finalement, nous attribuons cette valeur décimale, 0.5833, à l'attribut « Fill amount » pour ajuster le remplissage de la barre de vie en fonction des points de vie actuels du joueur ou de l'ennemi.

Rédigé par SELVARATNAM Akash

❖ Barre de chargement d'attaque

La barre de chargement d'attaque, représentée en bleu, indique au joueur s'il est prêt à attaquer ou non. Pour qu'un joueur puisse attaquer, cette barre doit être complètement remplie. Comme pour la barre de vie, nous utilisons l'attribut "Fill amount" pour gérer le remplissage de cette barre.

Pour déterminer la valeur de l'attribut "Fill amount", nous utilisons une fonction qui compare le temps d'attaque maximal de la classe de joueur concernée au temps d'attaque actuel de cette classe. Ensuite, nous convertissons cette valeur en format décimal (pourcentage) et l'attribuons à l'attribut "Fill amount". Par exemple, si le temps d'attaque maximal est de 2 secondes et que le joueur a attendu seulement 1 seconde, sa barre de chargement d'attaque sera remplie à 50 %.

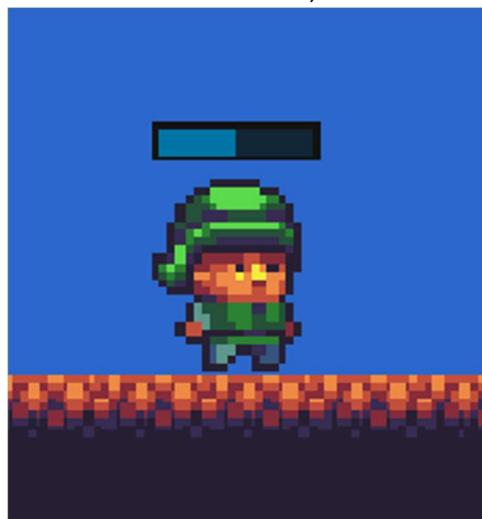


Figure 41 : barre de d'attaque à 50%

Rédigé par SELVARATNAM Akash

- ❖ Les ennemis
 - Système de patrouille

Les ennemis se déplacent de manière aléatoire au cours du jeu, cette fonctionnalité étant implémentée par un algorithme. Cet algorithme prend en paramètre deux positions : l'extrême droite et l'extrême gauche, puis il impose des limitations pour que l'ennemi se déplace entre ces deux positions. Ensuite, l'algorithme utilise la fonction "random" pour obtenir deux nombres aléatoires, l'un représentant un déplacement vers la gauche et l'autre vers la droite. En ajustant les positions de l'ennemi, de l'extrême gauche et de l'extrême droite, l'algorithme évite que l'ennemi reste trop longtemps coincé à l'extrême gauche ou à l'extrême droite.

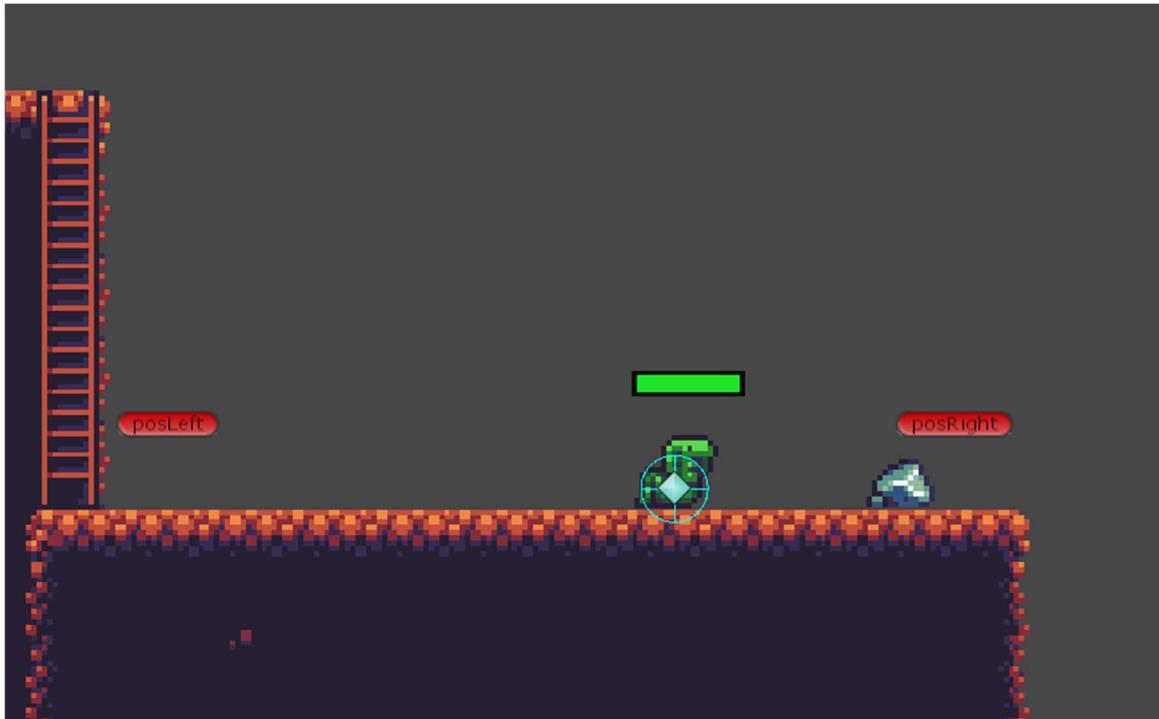


Figure 42 : L'un des ennemis et ses deux extrémités de position (capture d'écran)

Lorsque le joueur pénètre dans le périmètre de l'ennemi, nous désactivons l'algorithme qui permet à l'ennemi de patrouiller entre deux positions données. Ensuite, nous utilisons une méthode proposée par Unity appelée "Vector2.Distance(position1, position2)" qui prend en paramètre, dans notre cas, la position du joueur et la position de l'ennemi, et renvoie un résultat en float qui représente la distance entre le joueur et l'ennemi. Si la distance est inférieure à 10.0 float et que l'ennemi et le joueur sont à la même hauteur, cela déclenche un algorithme permettant à l'ennemi de suivre le joueur tant qu'il est dans son périmètre.

Si le joueur n'est plus dans le périmètre de l'ennemi, alors l'algorithme permettant de suivre le joueur se désactive, et l'algorithme permettant à l'ennemi de patrouiller entre les deux positions fournies en paramètre de la fonction est réactivé.

Rédigé par LIN Xingtong et SELVARATNAM Akash

❖ Inventaire

L'inventaire permet de stocker divers objets jusqu'à un nombre maximal de 4 objets. Les différents objets, principalement des potions, seront présentés dans une future partie, celle du magasin. L'interface de l'inventaire apparaît lorsque le joueur appuie sur la touche "I", faisant ainsi apparaître la fenêtre suivante :



Figure 43 : Inventaire (capture d'écran)

On peut apercevoir sur cette capture d'écran 3 potions qui sont stockées dans l'inventaire et que l'utilisateur peut cliquer dessus pour boire la potion. Ce clic est un évènement ajouté lors de l'ajout de l'objet dans l'inventaire. L'inventaire est une classe C# qui existe en tant qu'attribut de la classe du Joueur. Cela permet, lors du ramassage d'un objet par le joueur, de lancer une fonction sur l'inventaire pour le rajouter dans ce dernier. Pour éviter la suppression des objets de l'inventaire, la fonction qui permet d'ajouter les objets les assigne en tant que ses enfants dans la hiérarchie des objets.

Rédigé par LIN Oscar

❖ Coffres et pièces en or

Au cours de l'expérience de jeu du joueur, il pourra faire la rencontre de coffres et de pièces en or. Les pièces en or sont intégrées directement dans la disposition du niveau en tant qu'objets et apparaissent lorsqu'un ennemi est vaincu ou lors de l'ouverture d'un coffre. Le coffre est également un objet prédisposé dans le niveau, et il s'ouvre au contact du joueur. L'animation d'ouverture se déclenche, et un nombre aléatoire de pièces ainsi qu'une potion de soin sont projetés en l'air, que le joueur peut récupérer au contact.

La collision des pièces, des coffres et des potions est gérée par l'événement `OnTriggerEnter2D` (`Collider2D other`), qui s'active lorsque qu'un nouveau `Collider2D`, c'est-à-dire une zone de collision, entre en contact avec celui de l'objet actuel. Ce nouveau `Collider2D` est alors passé en tant que paramètre de la fonction, et on vérifie si ce `Collider2D` appartient bien au joueur.

Rédigé par LIN Oscar et COLLOMBET Nathan

4.4.2.4 Mini-carte

La mini-carte permet au joueur de connaître en temps réel ce qui se trouve devant lui, d'identifier les chemins et de mieux comprendre le terrain. Pour implémenter cette fonctionnalité, nous avons d'abord créé une caméra de mini-carte que nous avons placée comme enfant de la caméra principale. Ainsi, elle peut suivre le joueur et afficher la scène du jeu. Ensuite, nous avons utilisé une `Render Texture` pour capturer et afficher l'image de la caméra de la mini-carte.

Pour permettre au joueur de zoomer et dézoomer sur la carte, nous avons ajouté deux boutons sur le côté droit de la carte, modifiant ainsi la valeur de `orthographicSize` de la caméra pour implémenter la fonction de zoom. Dans le code, nous avons défini des valeurs maximales et minimales pour `orthographicSize` afin de limiter le zoom de la carte. Nous avons également ajouté une condition

dans le code : si la scène n'est ni de niveau 1 ni de niveau 2, la mini carte ne sera pas affichée.



Figure 44 : Un cas d'utilisation de la mini-carte, nous ne voyons plus ce qui se trouve en dessous, au niveau de la partie noire (capture d'écran)

Rédigé par LIN Xingtong

4.4.2.5 Sauvegarde des données

Pour la sauvegarde des données, nous avons réalisé deux types de sauvegarde. Tout d'abord, nous réalisons une sauvegarde de l'ensemble des données d'une partie jouée par un joueur. Il s'agit d'une fonction permettant de récupérer l'ensemble des données d'une partie, notamment le pseudo du joueur, le niveau, le nom du personnage choisi, le nombre de pièces récupérées, les points de vie du joueur, un booléen permettant de définir si le niveau est réussi ou non, et le pourcentage d'accomplissement du niveau. Ensuite, nous appelons une fonction côté backend afin de pouvoir

enregistrer les données récupérées dans la base de données, dans le document Game.

```
_id: ObjectId('65a3e698252854e46f94ec5d')
pseudo: "pam"
levelName: "WorldOneLvl1"
chooseCharacter: "Warrior"
coins: 208
health: 82
reussie: true
successPercentLevel: 100
__v: 0
```

```
_id: ObjectId('65a5030b2075104ac829e7ef')
pseudo: "Fla"
levelName: "WorldOneLvl1"
chooseCharacter: "Archer"
coins: 207
health: 0
reussie: false
successPercentLevel: 75.76
__v: 0
```

Figure 45 : Document Game

Nous avons également réalisé une sauvegarde des données du joueur dans le jeu. Pour ce faire, nous utilisons une fonction qui permet de récupérer le nombre total de pièces du joueur, ainsi que les niveaux des différentes capacités de l'archer et du guerrier, stockés sous forme d'un tableau. Ensuite, l'ensemble de ces données est transmis à une fonction côté backend, qui permet d'enregistrer les données dans le document UserGame de notre base de données.

```
▶ _id: ObjectId('6589ab8b3feedfae488b7254')
  pseudo: "se"
  coins: 0
  ▶ Archer: Object
  ▶ Warrior: Object
  __v: 0
```

```
_id: ObjectId('6589b951bafec2d8173388c8')
pseudo: "sq"
coins: 202
▶ Archer: Object
▶ Warrior: Object
__v: 0
```

```
_id: ObjectId('659ac2828a39a84f51ad7e3e')
pseudo: "nu"
coins: 0
▶ Archer: Object
▶ Warrior: Object
__v: 0
```

Figure 46 : Document UserGame

Rédigé par SELVARATNAM Akash

4.4.2.6 Son

Pour la gestion des sons dans le jeu, nous avons utilisé un prefab contenant plusieurs composants Audio Source, qui contiennent les différents sons du jeu (le son du joueur qui se déplace, le son du joueur lorsqu'il prend des dégâts, etc.).

Le script nommé "GameSound" contient de nombreuses fonctions pour gérer les sons. Il permet d'activer un son avec la fonction "Play()", de vérifier si un son est en cours de lecture avec la fonction "isPlaying()", et de mettre le son en pause avec la fonction "Pause()". Ces fonctions sont activées en fonction des actions qui se déroulent dans le jeu, telles que l'attaque d'un joueur ou l'ouverture d'un coffre, par exemple.



Figure 47 : Prefab permettant de gérer l'ensemble des audios du jeu

Rédigé par SELVARATNAM Akash

4.4.2.7 Niveau 2

Après avoir terminé le niveau 1, l'utilisateur passera au niveau 2, qui sera légèrement plus complexe. Il introduira des éléments tels que davantage d'échelles, de lianes, de plates-formes mobiles et des plates-formes qui disparaissent au contact du joueur. De plus, le risque de tomber dans l'eau, synonyme de décès, sera accru. Pour la conception du terrain principal, nous avons utilisé Tilemap, en créant plusieurs Tilemaps pour différentes parties de la carte, comme un pour le premier plan, un pour l'arrière-plan, un pour les échelles, etc., afin d'améliorer l'affichage des éléments de la carte et de gérer les collisions de manière plus efficace.

Rédigé par LIN Xingtong

- ❖ Plates-formes mobiles

Une plateforme se déplace de manière continue horizontalement ou verticalement, pouvant emmener le joueur avec elle. Cette fonctionnalité est implémentée grâce à un algorithme. La direction du déplacement dépend de deux positions fournies dans les paramètres. Ces deux positions représentent également les extrémités de déplacement possibles de la plateforme. Le changement de position de la plateforme est effectué avec la méthode `MoveTowards`. Lorsque la plateforme atteint l'une de ses extrémités, elle change de direction vers l'autre extrémité. Pour permettre au joueur de suivre la plateforme, la fonction `OnCollisionEnter2D` est utilisée : lorsqu'il entre en collision, le joueur est placé comme enfant de la plateforme, maintenant une cohérence de position. Lorsque le joueur quitte la plateforme, la fonction `OnCollisionExit2D` est utilisée pour annuler cette modification pour le joueur.



Figure 48 : Joueurs sur plateformes mobiles (capture d'écran)

Rédigé par LIN Xingtong

❖ Plates-formes cassées

Une plateforme qui se brise après que le joueur la touche. Cette plateforme est principalement réalisée grâce à StartCoroutine. Lorsque la plateforme entre en collision avec un objet taggé Player, la coroutine est déclenchée. Dans cette coroutine, une fonction est utilisée pour créer un effet de clignotement en réduisant la transparence de la plateforme pendant une courte période, puis la plateforme est détruite avec la méthode Destroy.



Figure 49 : moment où la plateforme cassée clignote après avoir été touché par le joueur (capture d'écran)

Rédigé par LIN Xingtong

4.4.2.8 Village

Le village est un lieu où il n'y a pas d'ennemis. Cette zone est présente afin que le joueur puisse développer son arbre de compétences et acheter des potions. Lorsque le joueur entre dans la boîte de collision, la fonction `OnCollisionEnter2D` du `GameObject` se mettra en marche, affichant ainsi un message indiquant qu'il y a une interaction possible.



Figure 50: village du jeu (capture d'écran)

Rédigé par ZHANG Anxian

❖ Arbre de compétence

Tout à droite, il y a un mage. Il suffit de lui parler pour qu'il propose l'amélioration de l'arbre de compétence contre des pièces en or, récupérables dans tous les niveaux.



Figure 51: menu d'amélioration de l'arbre de compétences (capture d'écran)

Rédigé par ZHANG Anxian

L'UI et l'UX de cette fenêtre ont été réalisées avec un panel. Nous y voyons quatre boutons d'ajout, distingués par leur fond vert et le symbole « + ». Du point de vue du code, nous avons d'abord référencé tous les GameObjects nécessaires au Script pour la mise en place de l'arbre de compétence (voir image ci-dessous).

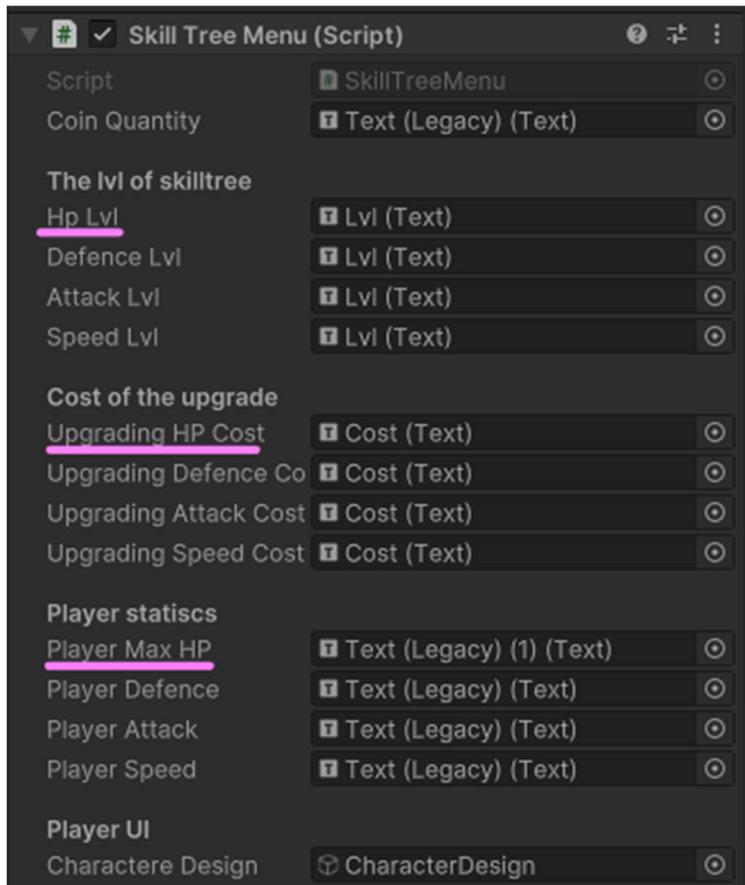


Figure 52: références fournis pour le Script du menu de l'arbre de compétences (capture d'écran)

Prenons les points de vie en tant qu'exemple. Lorsque le bouton d'ajout est pressé, le programme extrait dans un premier temps l'entier qui se trouve dans les textes « Hp Lvl » et « Upgrading HP Cost », correspondant aux deux premiers GameObjects soulignés en rose (voir ci-dessus). Ensuite, il vérifie si le joueur possède suffisamment de pièces.

Si c'est le cas, il réduira le nombre de pièces que le joueur possède en fonction du coût de l'amélioration, puis effectuera une mise à jour au niveau du coût initial, qui augmentera de 5 à chaque amélioration, du niveau de l'aptitude, et enfin des statistiques du joueur. Les statistiques gagnent 10% de celles qu'il possède actuellement, mise à part la défense, qui lui augmentera de 2% à chaque amélioration.

Rédigé par ZHANG Anxian

❖ Magasin

Le magasin se situant dans la structure remplie de boîtes permet au joueur d'acheter des potions en échange de pièces qu'il a pu récolter lors de son aventure.



Figure 53 : Magasin (capture d'écran)

Le menu du magasin affiche le nombre de pièces du joueur ainsi que les potions proposées avec leur coût. Une fonction a été réalisée pour l'achat d'une potion ; cette fonction prend en paramètre le prix de l'objet ainsi que l'objet lui-même. Elle vérifie d'abord que l'utilisateur a assez d'argent, puis que son inventaire n'est pas plein, pour finaliser l'achat de la potion en l'ajoutant dans l'inventaire du joueur et en lui retirant la somme correspondant à l'objet.

La fonction est appelée lors d'un événement OnClick, qui est affecté aux images de chacune des potions. Ces images sont associées à un bouton qui lance alors la fonction spécifique pour la potion concernée.

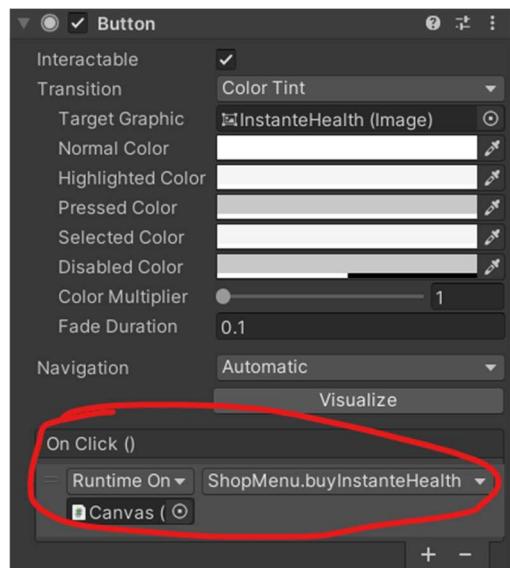


Figure 54 : élément bouton associé à chaque potions (capture d'écran)

Rédigé par LIN Oscar

❖ Portail de sélection de niveau

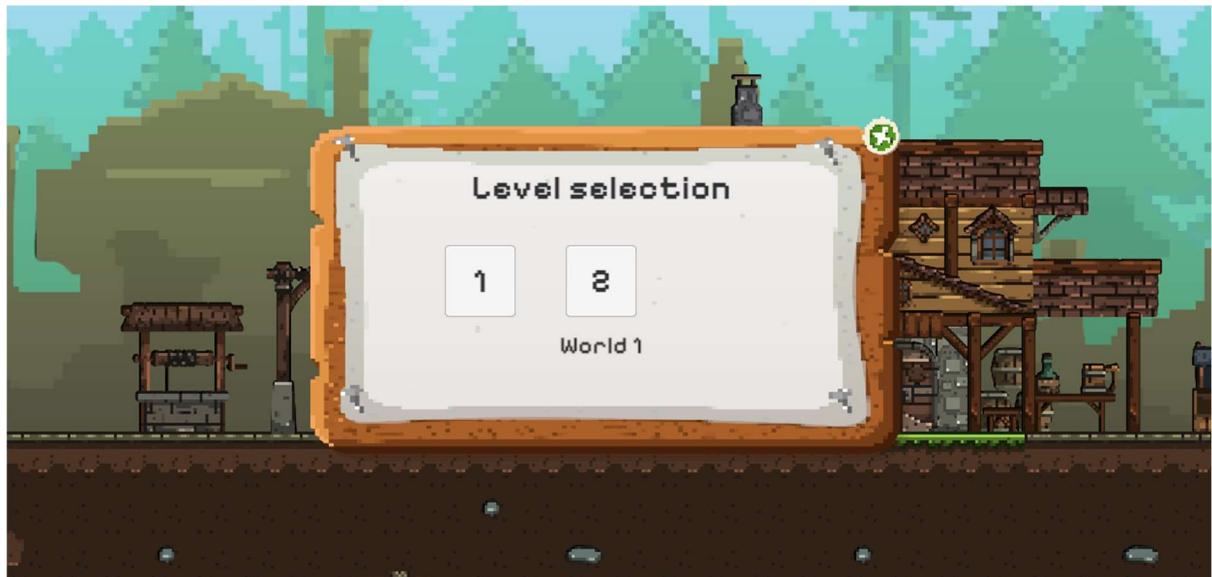


Figure 55: portail de sélection de niveaux (capture d'écran)

Se situant à l'opposé du mage, le portail permet de se téléporter vers les niveaux qui ont été débloqués. La redirection vers l'un des niveaux se fait en ajoutant un événement onClick à chaque bouton, utilisant la fonction Scene.LoadScene(*maScene*).

Rédigé par ZHANG Anxian

4.5 INTEGRATION DU JEU SUR LE SITE

Pour pouvoir réaliser l'incorporation du jeu sur le site, il faut tout d'abord créer une application WebGL directement dans l'application Unity. Il faut se rendre dans la section "Build and Settings" de l'application Unity, ensuite choisir la plateforme sur laquelle on souhaite convertir notre jeu, dans notre cas le format WebGL. Pour finir, il faut appuyer sur le bouton "Build" afin qu'Unity génère notre jeu au format WebGL.

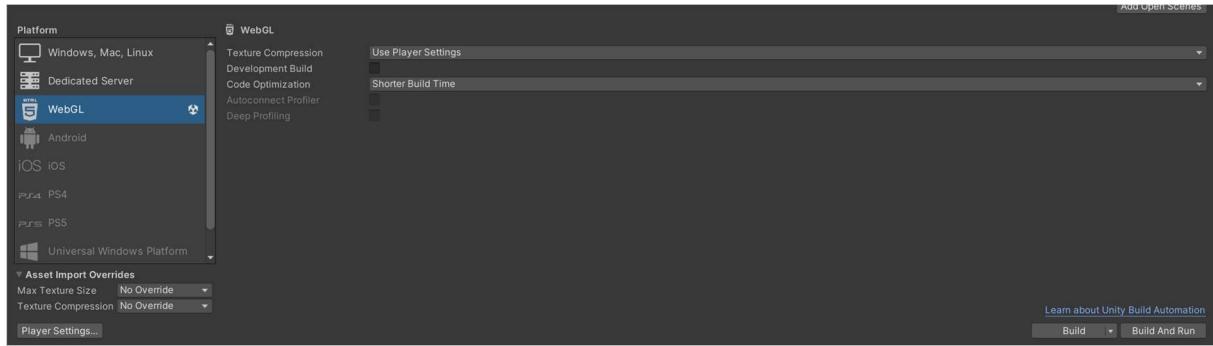


Figure 56 : Configuration format WebGL sur unity (capture d'écran)

Quand Unity a terminé la configuration du WebGL, il crée deux dossiers différents :

- **Build** : un dossier qui contient l'ensemble des fichiers de configuration nécessaires à la compilation du jeu au format WebGL.
- **TemplateData** : un dossier qui contient des images (logo Unity, icône de plein écran) ainsi que du style CSS pour le WebGL.

Le fichier index.html représente l'aspect visuel de la compilation WebGL. Il utilise les fichiers du dossier Build pour configurer et donner un aspect visuel au jeu, permettant ainsi au joueur de jouer

directement sur le site web. Il utilise également le dossier TemplateData pour afficher les images, telles que le logo Unity.

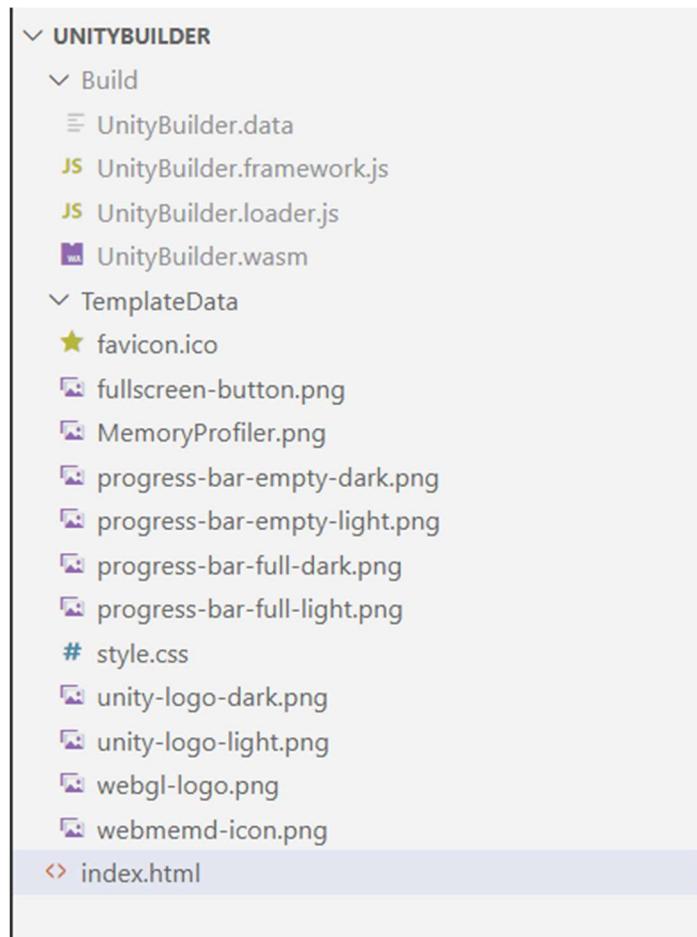


Figure 57 : Architecture de la compilation du WebGL (capture d'écran)

Pour pouvoir lancer le jeu, on peut utiliser l'extension proposée par Visual Studio Code nommée Live Server. Cette extension permet de lancer le jeu au format WebGL en local en appuyant sur le bouton "Go Live".



Figure 58: Extension Live server (capture d'écran)

Au niveau du code sur le site web, il faut utiliser l'élément HTML **iframe**, en spécifiant comme attribut la source (l'adresse HTTP de notre jeu), le style de cette balise et l'autorisation du plein écran.

```
const UnityCompile = () => {
  return (
    <View style={styles.ContainerGame}>
      <iframe
        src='http://127.0.0.1:5500/'
        style={styles.gameSize}
        allowFullScreen={true}
        allow='fullscreen'
      />
    </View>
  );
};
```

Figure 59: Page jeu de notre site web version code (capture d'écran)

Voici le rendu visuel de notre jeu au format WebGL sur la page "jeu" de notre site web. L'ensemble des fonctionnalités du jeu sont présentes dans la version compilée en WebGL sur notre site.

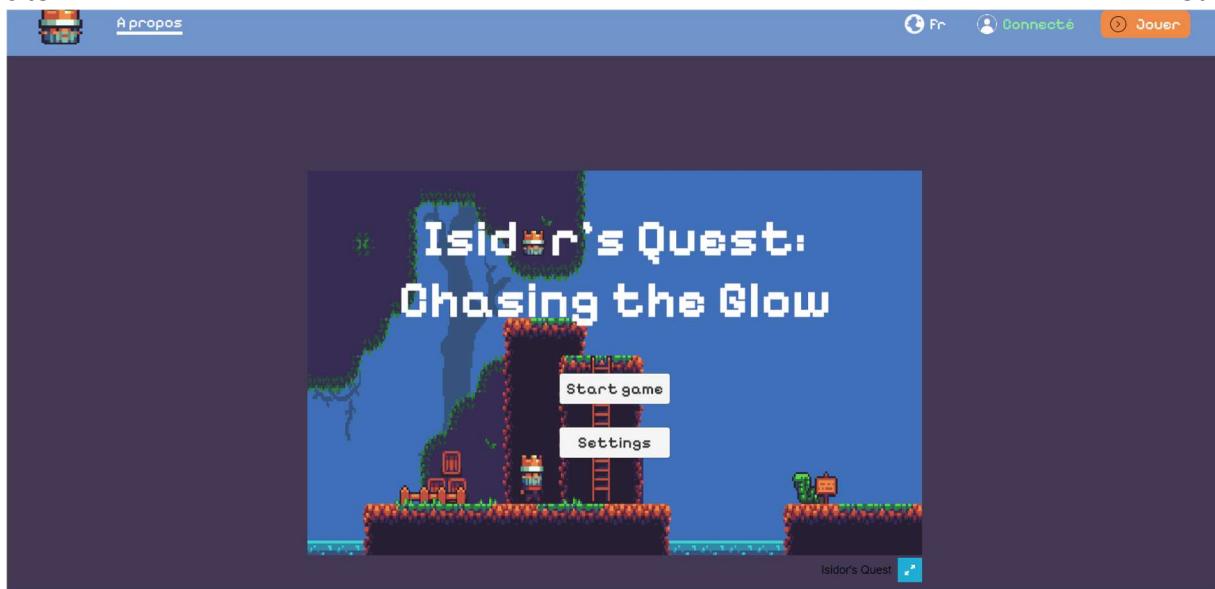


Figure 60 : Page jeu de notre site web (capture d'écran)

Rédigé par SELVARATNAM Akash

Conclusion

5 BILAN DE MONTEE EN COMPETENCES

5.1 SELVARATNAM AKASH

Personnellement, j'ai particulièrement progressé au niveau technique, j'ai pu retravailler avec le Framework React-Native, le langage Node.js et la base de données MongoDB et je pense que j'ai pu progresser particulièrement en node.js car j'ai pu travailler sur certaines thématiques dont je n'ai jamais travaillé auparavant comme le système de paiement avec Oscar ou la mise de la sauvegarde des données dans les cookies avec express-session. J'ai également découvert un nouveau monde au cours de ce projet le monde du jeu vidéo en 2D, j'ai pu apprendre l'utilisation d'Unity et apprendre le langage C# grâce aux différentes tâches que j'ai pu réaliser lors de ce projet. Au niveau des compétences personnelles, j'ai pu progresser au niveau de l'organisation et du travail en équipe et j'ai également progressé au niveau rédactionnel grâce aux rapports hebdomadaires, le cahier de charge ou le rapport final par exemple.

5.2 LIN XINGTONG

Pour moi, j'ai le plus progressé sur Node.js, car je ne connaissais pas du tout ce langage au début. Mais grâce à ce projet, j'ai pratiqué et créé entièrement deux pages qui utilisent Node.js et je le maîtrise maintenant bien. J'ai également progressé en React Native. Avant, je pouvais écrire du React en imitant ce que d'autres avaient écrit, mais je ne pouvais pas commencer de zéro. Après ce projet, je maîtrise bien React Native en codant les pages et en-tête, par exemple, je connais bien la création et l'utilisation de composants et la mise en œuvre du responsive en utilisant l'opérateur ternaire, etc. Pour Unity, j'ai découvert de nouvelles choses telles que Tilemap pour la création du monde du jeu, des boutons liés à des fonctions, comment commit et push des prefab, etc. En ce qui concerne C#, bien sûr, je connais désormais plus de méthodes et de fonctionnalités, et je comprends mieux les logiques pour programmation en C# après ce projet. Mon autonomie a aussi augmenté, car c'est vraiment la première fois que j'apprends trois langages en même temps par moi-même en un court laps de temps. J'ai également développé mon initiative dans ce projet, parce que je choisis activement et réalise rapidement les tâches que je veux accomplir.

5.3 ZHANG ANXIAN

De mon côté, je connaissais déjà l'outil Unity et le langage de programmation C#. C'est pourquoi j'ai pu développer assez rapidement toutes les fonctionnalités du village se trouvant dans notre jeu vidéo. J'ai aussi réalisé beaucoup de designs (avec les sprites récupérés sur les sites open source). Cela a été le cas pour la majorité des menus, mais aussi pour le niveau 1 et le village du jeu, ce qui m'a permis de progresser énormément sur le plan de l'UI et de l'UX. Du point de vue du développement web, j'ai été servie ! J'ai découvert et maîtrisé React Native et React JS. En effet, lors de la période A, dans tous les tutoriels sur React Native, il est indiqué qu'il vaut mieux connaître React JS avant de commencer avec React Native. NodeJS aussi, j'ai appris par moi-même cet environnement de développement. Et grâce au projet, j'ai pu mettre en pratique ce que j'ai appris en période A sur les pages d'à-propos et d'oubli de mot de passe. J'ai également corrigé de nombreux problèmes liés à la qualité du code ou à des bugs, que ce soit sur le frontend ou sur le backend, et cela a amélioré mes compétences en revue de code.

Par ailleurs, j'ai aussi pu mettre en pratique mes compétences d'un point de vue gestion et organisation, notamment en créant un Trello (contenant déjà une bonne partie des tâches à réaliser), un diagramme de Gantt, ou encore en guidant l'équipe sur l'utilisation des outils de gestion. Bien sûr, au niveau de la communication, j'ai essayé de préserver une communication entre tous les membres de l'équipe toutes les semaines.

5.4 LIN OSCAR

J'ai commencé le projet sans aucune connaissance sur chacun des langages proposés et je pense qu'au bout de ce projet j'ai pu y tous les apprendre. Que ce soit C#, avec mes connaissances déjà acquises avec les langages orienté objet avec java qui a pu faciliter mon apprentissage de C# et me concentré principalement sur les classes spécifiques à Unity que le fonctionnement du langage, ou les langages web avec React Native et Node.js, que j'ai pu apprendre en compagnie d'Akash qui en posséder déjà des connaissances dessus, comprenant alors la création d'un composant et de son utilisation sur une page. J'ai pu appliquer ce que j'ai appris pour la méthode paiement avec Stripe et Paypal. Il reste ainsi MongoDB où une simple vidéo m'a servi d'y comprendre son fonctionnement et d'avoir une meilleure vision de ce dernier avec les cours qu'on a pu avoir lors de notre cursus.

Au final je pense y avoir énormément appris lors de ce projet, m'étant lancé dans un petit projet de jeu qui m'est venu par la tête avec l'intention de faire plus attention à la structuration du code car je pense que c'est un point que je n'ai pas beaucoup travaillé lors de ce projet plutôt dans la pensée d'apprendre le fonctionnement des langages et un code qui fonctionne plutôt qu'un code plus facilement maintenable par le futur.

5.5 COLLOMBET NATHAN

Tout au long de ce projet, j'ai significativement amélioré mes compétences techniques, en particulier dans le domaine du développement web. Bien que j'étais déjà avancé en Unity et en C#, c'est dans l'apprentissage et l'application de technologies web telles que React, Node.js et MongoDB que j'ai le plus progressé. De plus, j'ai pu associer ces technologies avec l'utilisation d'API et de bibliothèque comme Mongoose ou Express.js. Cette montée en compétence m'a permis de développer une compréhension plus profonde du développement web en me familiarisant avec ces technologies modernes.

Au-delà des compétences techniques, ce projet a également été une opportunité précieuse pour renforcer mes compétences personnelles. J'ai vu mon esprit d'équipe s'épanouir, favorisé par la collaboration et la communication constantes avec les autres membres de l'équipe. Ma capacité à rédiger des rapports clairs et détaillés s'est améliorée. En outre, j'ai gagné en autonomie et je suis capable de prendre des initiatives.

6 LES DIFFICULTES RENCONTREES

6.1 SELVARATNAM AKASH

Au départ, j'ai eu quelque problème de communication avec mon équipe car c'est la première fois que je fais équipe avec la plupart des membres du groupe mais au fur à mesure de l'avancement du projet, j'ai pris l'initiative de prendre plus la parole sur discord ou lors des réunions de groupe, de plus, la mise en place de Trello m'a beaucoup aidé dans la répartition des tâches de chacun afin de pas reproduire les tâches déjà réalisé.

J'ai également eu quelques problèmes sur GitHub au début du projet concernant les push, les merge de certain éléments d'Unity sur GitHub, mais finalement, nous avons trouvé une solution assez rapidement pour pouvoir résoudre ce problème grâce aux prefab sur Unity.

6.2 LIN XINGTONG

Les difficultés que j'ai rencontrées étaient surtout au début de mon apprentissage de React et de Node. J'ai suivi des tutoriels et en même temps, j'ai écrit les mêmes fichiers du code que ceux des tutoriels. J'ai réussi à les faire fonctionner, mais au fond, je ne comprenais pas vraiment l'utilité de certaines méthodes, quand les utiliser, etc. Cependant, avec la pratique dans la partie du site, j'ai rapidement compris l'utilisation de chaque méthode et comment réaliser des pages web à partir de zéro avec ces deux langages.

Quant à Unity, j'ai rencontré des problèmes au début du développement de certaines fonctionnalités, comme la mini-carte ou le mouvement du joueur sur des échelles. Ce qui est avantageux lors du développement avec Unity, c'est qu'il existe de nombreux tutoriels en ligne pour des diverses fonctionnalités de jeu. Ainsi, après avoir trouvé des tutoriels, la plupart de problèmes ont été résolus.

6.3 ZHANG ANXIAN

Du côté des difficultés rencontrées, celle qui m'a le plus marqué s'est produite lors de la période A. Nous sommes un groupe de 5, cependant, plus le temps passait, plus on voyait une division dans le groupe. Il y avait deux groupes, l'un faisait la rédaction du cahier des charges, les comparaisons et tests logiciels, tandis que l'autre groupe programmait « en cachette » le site web. En voyant cette fracture, j'ai pris l'initiative et commencé à faire la gestion du projet.

6.4 LIN OSCAR

Les difficultés que j'ai pu rencontrer fut l'apprentissage de react native et node.js, trouvant plusieurs ressources sur internet vidéo ou textuelles, je ne semblais pas toujours comprendre comment certaines choses fonctionnaient, mais c'est après s'être lancé dans le projet que j'ai pu être accompagné par Akash lors du projet pour la partie web que ma compréhension de ses langages fut augmentées. Je ne comprenais aussi pas réellement l'organisation du groupe, dû à une sorte de communication je venais à avoir l'impression qu'au début du projet chacun réalisé ses tâches sans

réellement partagés ce qui a été accompli, c'est suite à une mise en concertation d'une réunion par semaine et de l'utilisation de l'outil Trello que ce sentiment de désordre fut parti de moi.

6.5 COLLOMBET NATHAN

Au début de notre projet, nous avons fait face à des difficultés majeures dues à un manque de communication au sein de l'équipe. Cette lacune a causé des malentendus et une perte de temps, entravant la progression du projet. Nous avons vite compris l'importance d'une communication claire et régulière pour coordonner efficacement nos efforts et éviter les redondances de travail. Cette expérience a été une leçon précieuse, nous poussant à améliorer nos méthodes de communication, un aspect clé pour la suite et le succès du projet.

7 CE QU'IL RESTE A ACCOMPLIR

Nous avions initialement prévu d'intégrer la portée, c'est-à-dire la distance d'attaque du personnage, dans notre panneau de compétences. Cependant, au cours du développement, nous avons constaté que la vie du personnage était cruciale. Avec seulement quatre emplacements disponibles dans le panneau, nous avons finalement opté pour les points de vie à la place de la portée.

Une autre fonctionnalité que nous n'avons pas pu implémenter comme prévu concerne le nombre de niveaux. À l'origine, nous envisagions de créer sept mondes différents, chacun comprenant trois niveaux. Cependant, en raison de la charge de travail importante à la fois sur la partie web et la partie jeu, et de notre volonté de développer diverses fonctionnalités telles que le menu pause, la mini-carte, et même une fonctionnalité qui, finalement, n'a pas pu être réalisée : l'enregistrement de la progression du jeu. Tout cela, avec un temps limité, a conduit à la réalisation d'un seul monde avec deux niveaux.

Nous n'avons pas non plus réussi à créer le Totem de résurrection, un objet offrant une deuxième chance de vie après la mort, en raison de contraintes de temps, comme dans le cas précédent.

Concernant la fonctionnalité multijoueur du jeu, bien que des recherches et des essais aient été effectués, sa complexité a rendu son implémentation impossible pour notre équipe.

Nous n'avons pas pu mettre en œuvre complètement l'intelligence artificielle de base comme prévu dans le cahier des charges. Ce que nous avons fait, c'est simplement utiliser des algorithmes pour rendre le comportement des ennemis plus flexible, mais sans développer un véritable modèle d'intelligence artificielle.

Rédigé par LIN Xingtong

8 PERSPECTIVES D'AMELIORATION

Premièrement, l'organisation, et notamment la répartition des tâches, a été un défi. Au début, nous n'étions pas habitués à utiliser Trello ni à mettre à jour les tâches sur cette plateforme. Il est donc souvent arrivé que plusieurs d'entre nous travaillent simultanément sur la même tâche, ce qui a entraîné une perte de temps.

Deuxièmement, le code pourrait être rendu plus clair et plus efficace pour faciliter la compréhension et la maintenance future.

Enfin, il est dommage que nous n'ayons pas découvert Jira plus tôt. Jira peut générer automatiquement une vue chronologique sous la forme d'un diagramme de Gantt. Si nous avions utilisé Jira au lieu de Trello, nous aurions pu économiser le temps consacré à la création du diagramme de Gantt et disposer ainsi de plus de temps pour le développement de notre projet.

Rédigé par LIN Xingtong

9 ANNEXES

9.1 ANNEXE 1 : RESUME

Notre projet consiste en la création d'un site hébergeant un jeu 2D de plateforme au style pixel art. Cette orientation résulte d'une longue réflexion pour plusieurs raisons. D'une part, notre enfance a été marquée par des jeux 2D tels que Mario ou Sonic, et d'autre part, c'est un rêve pour nous de réaliser un tel type de jeu. De plus, les langages et outils nécessaires, tant pour le site web que pour le jeu, ne sont pas tous maîtrisés par notre équipe. Ce projet représente donc une opportunité idéale d'apprendre l'utilisation de nouveaux outils et langages informatiques.

Pour le développement du site web, nous avons choisi d'utiliser divers langages informatiques. Le framework React Native est employé sur le frontend pour permettre une visualisation web ainsi que mobile (iOS et Android). Côté backend, nous avons opté pour Node.js, couramment utilisé avec React. Il facilite le travail avec la bibliothèque Express, qui exécute rapidement de nombreuses fonctions web (requêtes HTTP, sauvegarde de données dans les cookies, etc.). Nous utilisons également MongoDB, une base de données non relationnelle, en synergie avec Mongoose, une bibliothèque Node.js, pour gérer efficacement les documents de notre base de données.

Pour le développement du jeu, notre choix s'est porté sur Unity, grâce à sa grande communauté et sa documentation riche. Unity facilite le développement avec ses nombreuses méthodes et permet de résoudre aisément les problèmes grâce à sa communauté et sa documentation. De plus, Unity peut générer le jeu au format WebGL, ce qui est idéal pour une visualisation web dans le cadre de notre projet.

En termes d'accomplissements, nous avons réussi à développer un jeu 2D comprenant deux classes de personnages jouables, le guerrier et l'archer, chacun avec ses particularités (par exemple, le guerrier attaque à courte distance tandis que l'archer peut attaquer de n'importe quelle distance). Nous avons également implémenté diverses fonctionnalités, telles qu'un village avec un magasin vendant des potions, un panneau de compétences pour améliorer les statistiques des personnages, et un sélecteur de niveaux. À ce jour, notre monde de jeu comprend deux niveaux différents, avec divers types d'ennemis, des pièces à collecter et des coffres contenant divers objets.

Nous avons aussi mis en place un site promotionnel permettant aux joueurs de découvrir le jeu, de s'inscrire, et de sauvegarder leurs parties ainsi que les données globales de leurs personnages (nombre de pièces, niveaux des personnages). Ce site offre également la possibilité d'acheter le jeu et d'y jouer directement, ce qui est pratique car notre jeu ne nécessite aucune installation.

Bien que nous n'ayons pas réussi à implémenter toutes les fonctionnalités initialement prévues, notamment sept mondes avec trois niveaux chacun, nous sommes parvenus à créer un monde avec deux niveaux différents. Cette limitation est due à la charge de travail importante sur le site web et le jeu. Certaines fonctionnalités, comme le totem de résurrection ou un mode multijoueur, n'ont pas été réalisées en raison de leur complexité ou par manque de temps.

Pour nos futurs projets, nous envisageons d'améliorer la répartition des tâches, de rendre notre code plus clair et commenté pour faciliter sa compréhension, et d'utiliser Jira plutôt que Trello. Jira, en proposant des diagrammes de Gantt, aurait pu nous aider à mieux gérer notre temps et à prioriser nos tâches.

Rédigé par SELVARATNAM Akash et COLLOMBET Nathan

9.2 ANNEXE 2 : ABSTRACT

Our project involves creating a website hosting a 2D pixel art style game. After extensive considerations, this direction was chosen for several reasons. Firstly, we were all inspired by 2D games like Mario or Sonic in our childhood, and secondly, it has always been a dream to create such a game. Moreover, the languages and tools required for both the website and the game are not fully mastered by our team. Therefore, this project represents an ideal opportunity to learn new tools and computer languages.

For the website development, we have decided to use various computer languages. The React Native framework is used on the frontend to enable both web and mobile (iOS and Android) views. On the backend, we have chosen Node.js, commonly used with React. It works efficiently with the Express library, which quickly performs numerous web functions (HTTP requests, cookie data storage, etc.). We are also using MongoDB, a non-relational database, together with Mongoose, a Node.js library, for an efficient management of our database documents.

For the game development, Unity was selected due to its large community and extensive documentation. Unity simplifies development thanks to its comprehensive methods and allows for easy problem-solving with its vast community and documentation. Additionally, Unity can generate the game in WebGL format, which is crucial for web viewing in our project.

In terms of achievements, we have successfully developed a 2D game featuring two playable character classes: the warrior and the archer, each with unique abilities (for example, the warrior attacks at short range while the archer can attack from any distance). We have also implemented various features such as a village with a potion-selling shop, a skill panel for improving character stats, and a level selector. Currently, our game world includes two different levels, with various types of enemies, collectible coins, and treasure chests with assorted items.

We have also established a promotional website allowing players to discover the game, register, and save their game sessions as well as their characters' overall data (coin count, character levels). This website also serves as a platform where to purchase the game, offering the ability to directly play online as it does not require installation.

Although we did not manage to implement all the initially planned features, such as seven worlds with three levels each, we have succeeded in creating one world with two different levels. This was due to the significant workload on both the website and the game. Some features, like the resurrection totem or a multiplayer mode, were not completed due to their complexity or lack of time.

For our future projects, we plan to improve task distribution, make our code clearer and more commented for easier understanding, and to use Jira instead of Trello. Jira, with its Gantt charts, could have helped us manage our time more effectively and prioritize tasks.

Written by COLLOMBET Nathan

9.3 ANNEXE 3 : LES SOURCES

Collectif. « Le marché du jeu vidéo – Faits et chiffres ». 13 décembre 2023. Statista.

<https://fr.statista.com/themes/9063/le-marche-du-jeu-video/#topicOverview>

Collectif. « Industrie du jeu - Analyse de la taille et des parts – Tendances et prévisions de croissance (2023 - 2028) ». 2018. Mordor Intelligence.

<https://www.mordorintelligence.com/fr/industry-reports/global-gaming-market>

Collectif. « AAA (jeu vidéo) ». 2016. Wikipédia.

[https://fr.wikipedia.org/wiki/AAA_\(jeu_vid%C3%A9o\)](https://fr.wikipedia.org/wiki/AAA_(jeu_vid%C3%A9o))

Collectif. « Unity ou Unreal engine, quel moteur de jeux choisir ? ». 20XX. Formation facile.

<https://www.formation-facile.fr/blog/unity-ou-unreal-engine-quel-moteur-de-jeux-choisir>

Pauline Callies. « Les meilleurs outils pour créer votre maquette d'application mobile ». 19 mai 2022. Aventique.

<https://aventique.paris/maquette-application-mobile/#:~:text=Figma%20est%20un%20outil%20de,de%20ses%20fonctionnalit%C3%A9s%20est%20payant.>

Collectif. « 6 alternatives à Google Drive ». 2021. Blogdumoderateur.

<https://www.blogdumoderateur.com/tools/alternatives/google-drive/>

Eloïse Salson. « Suivez vos modifications à la trace avec les 8 meilleurs logiciels de versionning ». 19 janvier 2022. Appvizer.

<https://www.appvizer.fr/magazine/services-informatiques/gestion-versions/outils-versionning>

Alicia Raeburn. « Logiciels et outils de gestion de projet : les meilleurs choix pour 2023 ». 9 octobre 2023. Asana.

<https://asana.com/fr/resources/best-project-management-software>

Collectif. « Top 10 des frameworks frontend les plus populaires à utiliser en 2024 ». 26 septembre 2022. AppMaster.

<https://appmaster.io/fr/blog/frameworks-frontaux-populaires>

Introduction à Express/Node - Apprendre le développement Web | MDN. (s. d.). MDN Web Docs.

[Introduction à Express/Node - Apprendre le développement web | MDN \(mozilla.org\)](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Node.js)

Contributeurs aux projets Wikimedia. (2023, 14 novembre). *WebGL*.

<https://fr.wikipedia.org/wiki/WebGL>

9.4 ANNEXE 4 : IMAGES DU DIGRAMME DE GANTT

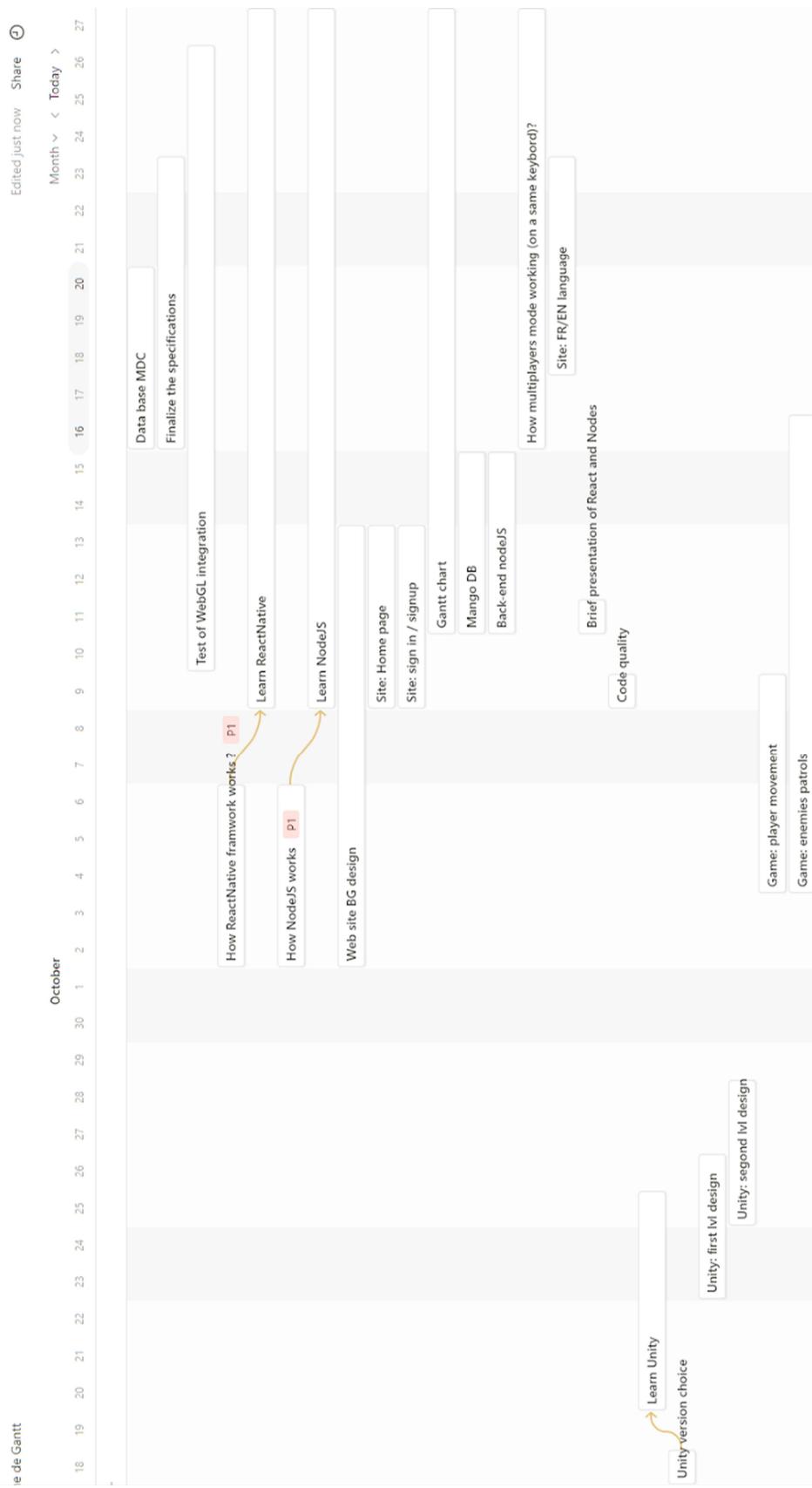


Figure 61: Diagramme de Gantt, première partie (capture d'écran)

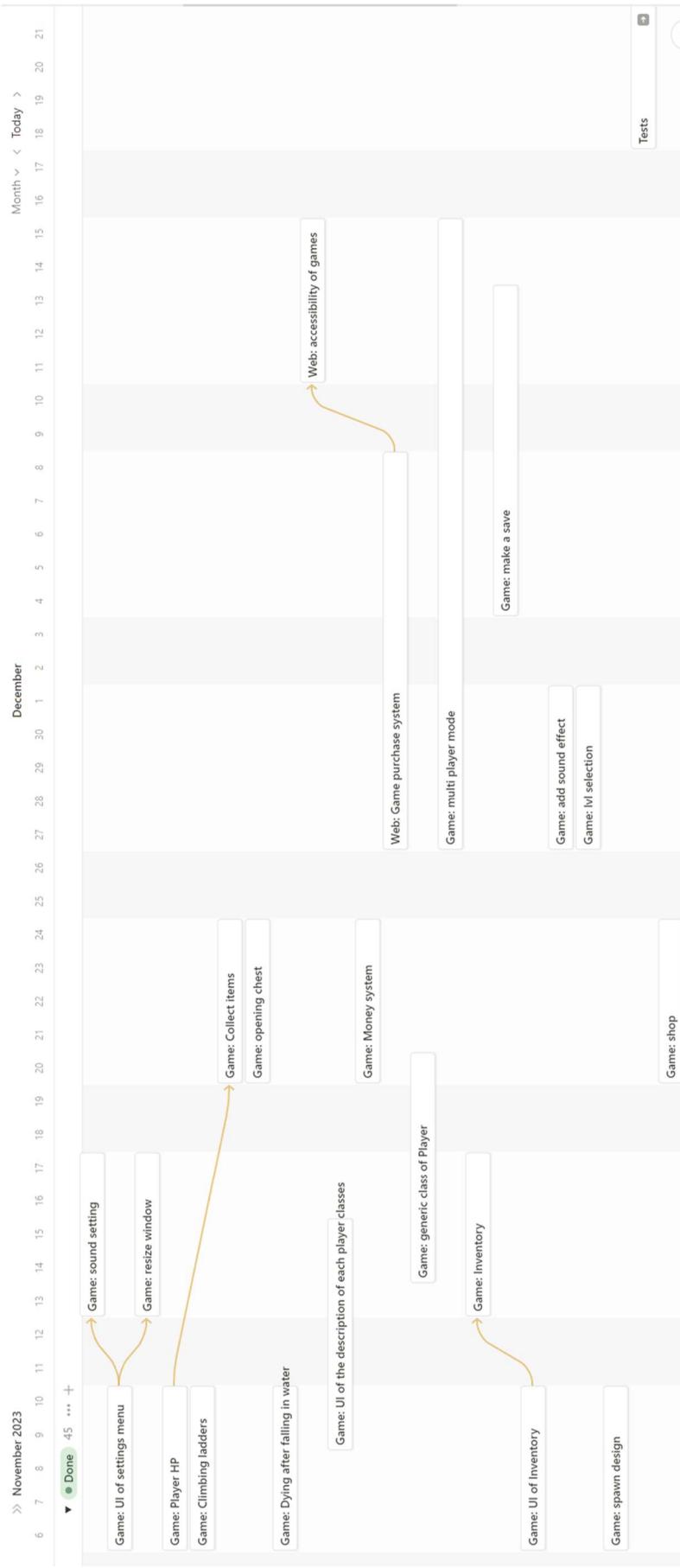


Figure 62: Diagramme de Gantt deuxième partie (capture d'écran)

9.5 ANNEXE 5 : TABLE DES ILLUSTRATIONS

| | |
|--|----|
| Figure 1: Bilan du marché Français 2020 dans le secteur du jeu vidéo. L'essentiel-jeu-vidéo [Fichier PDF]. Page 8. 2021..... | 5 |
| Figure 2: charte graphique utilisée pour l'application (capture d'écran de la charte graphique) | 11 |
| Figure 3: Exemple de tâche effectuée et labellisée « Terminé » (capture d'écran) | 13 |
| Figure 4: Diagramme de cas d'utilisation de notre application | 16 |
| Figure 6: Diagramme de séquence de la page oublie de mot de passe..... | 18 |
| Figure 7: Diagramme de séquence de la page inscription | 19 |
| Figure 8: Diagramme de séquence de la page de vérification de code | 20 |
| Figure 9: Diagramme de séquence de la page de modification de mot de passe | 21 |
| Figure 10: Diagramme de séquence de la page de modification des données utilisateur | 22 |
| Figure 11: Diagramme de séquence de la page de connexion | 23 |
| Figure 12: Diagramme de séquence de la page de contact | 24 |
| Figure 13: Diagramme d'état-transition du joueur | 25 |
| Figure 14: Diagramme d'état-transition des ennemis du jeu | 25 |
| Figure 15 : page d'accueil, un exemple après avoir ajusté la distance à l'aide d'une vue vide (capture d'écran)..... | 26 |
| Figure 16 : page d'inscription, exemple avec l'erreur nom d'utilisateur déjà pris (capture d'écran) ... | 27 |
| Figure 17 : page de vérification du code (capture d'écran) | 28 |
| Figure 18 : Exemple de mail envoyé à un utilisateur avec le code à entrer..... | 28 |
| Figure 19 : Exemple d'utilisateur dans le document User (capture d'écran)..... | 29 |
| Figure 20 : Page de connexion, exemple de connexion avec un nom d'utilisateur ou un mot de passe incorrect (capture d'écran) | 30 |
| Figure 21: page d'oubli de mot de passe, exemple d'un mail non valide (capture d'écran) | 31 |
| Figure 22 : Page « A propos » (capture d'écran) | 32 |
| Figure 23 : Page de contact (capture d'écran) | 33 |
| Figure 24: Mail de confirmation envoyé à l'utilisateur (capture d'écran) | 33 |
| Figure 25 : Page de paiement (capture d'écran)..... | 34 |
| Figure 26 : Paiement par PayPal (capture d'écran)..... | 35 |
| Figure 27: Paiement par carte bancaire (capture d'écran) | 35 |
| Figure 28 : Paiement par carte bancaire réussie (capture d'écran)..... | 36 |
| Figure 29: Page de consultation/modification des données (capture d'écran)..... | 37 |
| Figure 30 : Modification des données (capture d'écran) | 38 |
| Figure 31: Header (capture d'écran) | 38 |
| Figure 32: Liste des langues disponibles sur notre site web (capture d'écran) | 39 |
| Figure 33: Fichier JSON Français (capture d'écran)..... | 39 |
| Figure 34 : Fichier JSON Anglais (capture d'écran). | 39 |
| Figure 35: Connexion (Header, capture d'écran) | 39 |
| Figure 36 : Footer (capture d'écran) | 40 |
| Figure 37 : menu d'accueil (capture d'écran)..... | 41 |
| Figure 38: menu de configuration (capture d'écran) | 42 |
| Figure 39 : Scène de sélection de personnage (Capture d'écran)..... | 43 |
| Figure 40 : Barre de vie du joueur à 50% et barre de vie de l'ennemi à 100% | 44 |
| Figure 41 : Les attributs de l'image de la barre de vie | 44 |
| Figure 42 : barre de d'attaque à 50% | 45 |
| Figure 43 : L'un des ennemis et ses deux extrémitées de position (capture d'écran) | 46 |
| Figure 44 : Inventaire (capture d'écran)..... | 47 |

| | |
|---|----|
| Figure 45 : Un cas d'utilisation de la mini-carte, nous ne voyons plus ce qui se trouve en dessous, au niveau de la partie noire (capture d'écran)..... | 49 |
| Figure 46 : Document Game..... | 50 |
| Figure 47 : Document UserGame | 50 |
| Figure 48 : Prefab permettant de gérer l'ensemble des audios du jeu..... | 51 |
| Figure 49 : Joueurs sur plateformes mobiles (capture d'écran)..... | 52 |
| Figure 50 : moment où la plateforme cassée clignote après avoir été touché par le joueur (capture d'écran)..... | 53 |
| Figure 51: village du jeu (capture d'écran)..... | 54 |
| Figure 52: menu d'amélioration de l'arbre de compétences (capture d'écran)..... | 54 |
| Figure 53: références fournis pour le Script du menu de l'arbre de compétences (capture d'écran).. | 55 |
| Figure 54 : Magasin (capture d'écran)..... | 56 |
| Figure 55 : élément bouton associé à chaque potions (capture d'écran) | 56 |
| Figure 56: portail de sélection de niveaux (capture d'écran) | 57 |
| Figure 57 : Configuration format WebGL sur unity (capture d'écran) | 58 |
| Figure 58 : Architecture de la compilation du WebGL (capture d'écran) | 59 |
| Figure 59: Extension Live server (capture d'écran) | 59 |
| Figure 60: Page jeu de note site web version code (capture d'écran)..... | 60 |
| Figure 61 : Page jeu de notre site web (capture d'écran) | 60 |
| Figure 62: Diagramme de Gantt, première partie (capture d'écran)..... | 71 |
| Figure 63: Diagramme de Gantt deuxième partie (capture d'écran)..... | 72 |

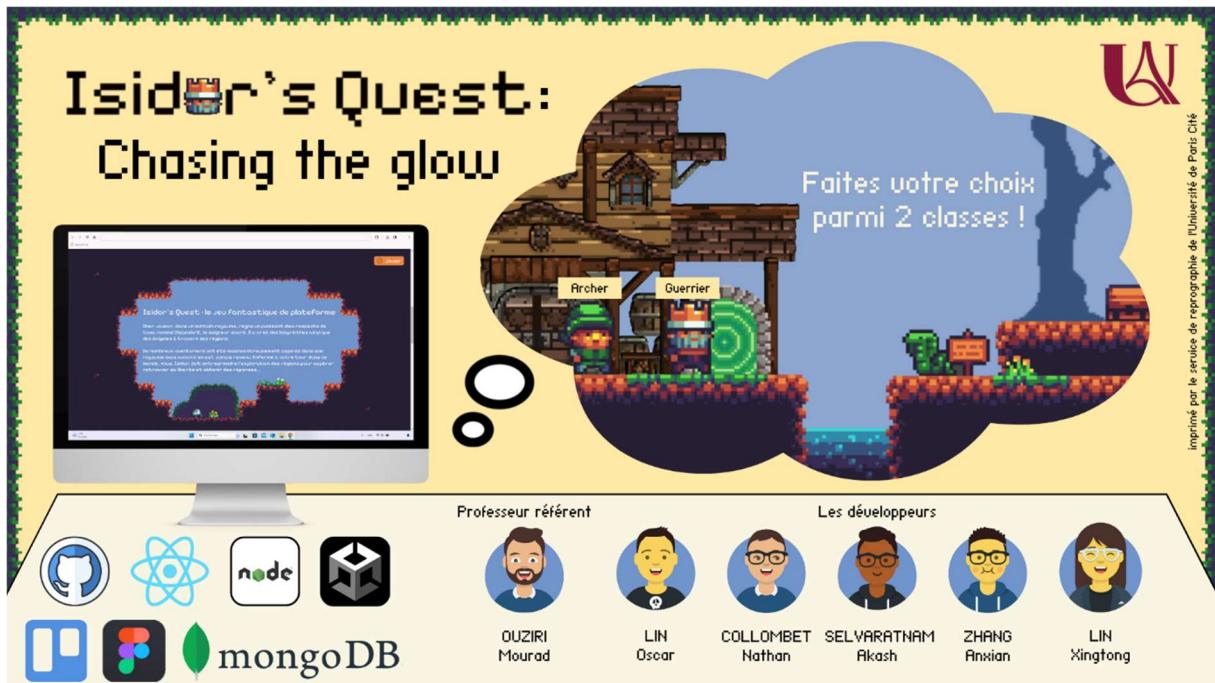
9.6 ANNEXE 6 : CAHIER DES CHARGES

Voir pièces jointes.

9.7 ANNEXE 7 : PROPOSITION DE PROJET SAE S5

Voir pièces jointes.

9.8 ANNEXE 8 : POSTER



9.9 ANNEXE 9 : GLOSSAIRE

- ❖ **useState** : Permet de détecter localement (limité à des composants) si un état a été modifié.
- ❖ **useEffect** : Indique à React d'exécuter une tâche chaque fois qu'un état change, gérant ainsi les effets de bord.
- ❖ **TextInput** : Champ pour saisir du texte.
- ❖ **Text** : Composant servant à afficher du texte.
- ❖ **TouchableOpacity** : Rend un élément cliquable, tel qu'un bouton ou un lien, avec un effet d'opacité lors du clic.
- ❖ **useNavigation** : Facilite la navigation entre les différentes pages du site web.
- ❖ **Regex** : Un Regex est une chaîne de caractères représentant un ensemble de chaînes de caractères possibles.
- ❖ **Nodemailer** : Module de Node.js utilisé pour envoyer des e-mails.
- ❖ **Express** : Framework destiné à construire des applications web pour Node.js, facilitant l'exécution de requêtes HTTP.
- ❖ **Express-session** : Librairie proposée par Express pour sauvegarder des données directement dans un cookie.
- ❖ **useRef** : Permet de référencer une valeur (composants, variables, etc.).
- ❖ **paypal-rest-sdk** : Librairie offerte par PayPal pour réaliser des paiements via PayPal.
- ❖ **Stripe** : Librairie fournie par Stripe pour effectuer différents types de paiements.
- ❖ **Mongoose** : Bibliothèque JavaScript permettant d'établir une connexion entre MongoDB et une application codée en JavaScript. Elle facilite la création de documents et l'exécution de requêtes (find, update, delete, etc.) sur MongoDB.
- ❖ **SelectDropdown** : Composant de React Native permettant de créer des listes avec plusieurs éléments cliquables.

- ❖ **Image** : Composant de React Native utilisé pour afficher des images.
- ❖ **Sprites** : Éléments graphiques pouvant se déplacer sur l'écran, tels que des personnages, des coffres, des pièces, etc.
- ❖ **JSON** : Format de fichier textuel conçu pour l'échange de données.
- ❖ **Panel** : Élément utilisé pour réaliser l'interface utilisateur (UI) et l'expérience utilisateur (UX), en fournissant tous les outils nécessaires (boutons, champs de texte, etc.).
- ❖ **GameObject** : Objet présent dans une scène d'Unity.
- ❖ **Script** : Fichier contenant le code source d'un objet ou d'une fonctionnalité.
- ❖ **Paramètre** : Variable de type connu, dont les valeurs sont spécifiées au moment de l'exécution.
- ❖ **MoveTowards** : Fonction permettant de déplacer progressivement un objet d'une position à une autre à une vitesse donnée.
- ❖ **OnCollisionEnter2D** : Méthode appelée automatiquement lorsqu'un objet entre en collision avec un autre dans un environnement 2D dans Unity.
- ❖ **OnCollisionExit2D** : Méthode appelée automatiquement lorsqu'un objet quitte une collision avec un autre dans un environnement 2D dans Unity.
- ❖ **StartCoroutine** : Fonction utilisée pour démarrer l'exécution d'une coroutine, permettant des actions temporisées ou des animations dans Unity.
- ❖ **Destroy** : Fonction qui supprime un objet ou un composant dans Unity, libérant ainsi les ressources qui y sont associées.
- ❖ **Render Texture** : Technique permettant de capturer et d'afficher la sortie graphique d'une caméra dans Unity.
- ❖ **OrthographicSize** : Propriété de la caméra définissant la taille de la vue orthographique, influençant le zoom de la caméra.
- ❖ **Tilemap** : Grille utilisée pour "peindre" le décor d'un jeu.
- ❖ **Prefab** : Composant d'Unity permettant de contenir un GameObject avec ses composants et ses propriétés.
- ❖ **WebGL** : WebGL est une technologie permettant d'afficher des éléments graphiques en 2D ou en 3D sur un navigateur web.

9.10 ANNEXE 10 : LIEN GITHUB POUR LE CODE SOURCE

<https://github.com/AnxianZhang/SaeSS>