Page de garde

Remerciement

Contents

[Description du projet : Isidor’s Quest : Chasing the glow 5](#_Toc155514142)

[Pourquoi ce projet et pas un autre ? 5](#_Toc155514143)

[Phases d’analyses 6](#_Toc155514144)

[Le marché actuel 6](#_Toc155514145)

[Ce qui fait notre singularité 7](#_Toc155514146)

[Choix techniques 7](#_Toc155514147)

[Le rôle de chaque composant 7](#_Toc155514148)

[Pourquoi ces outils plutôt que d’autres ? 8](#_Toc155514149)

[Charte graphique 10](#_Toc155514150)

[Notre organisation 11](#_Toc155514151)

[Planification 11](#_Toc155514152)

[Rôles de chacun(e) 11](#_Toc155514153)

[Répartition des tâches 12](#_Toc155514154)

[Nos deux applications 13](#_Toc155514155)

[Les services proposés (architecture fonctionnel) 13](#_Toc155514156)

[Site 13](#_Toc155514157)

[Jeu 13](#_Toc155514158)

[Nos phases de développement 14](#_Toc155514159)

[Conception des applications (architecture technique) 14](#_Toc155514160)

[Site 14](#_Toc155514161)

[Jeu 14](#_Toc155514162)

[Incorporation du jeu sur le site 14](#_Toc155514163)

[Les tests utilisateurs 14](#_Toc155514164)

[Bilan de montée en compétences 15](#_Toc155514165)

[Les difficultés rencontrées 15](#_Toc155514166)

[Ce qui reste à accomplir 15](#_Toc155514167)

[Perspectives d’amélioration 15](#_Toc155514168)

[List des annexes 16](#_Toc155514169)

[Annexe 1 : Résumé 17](#_Toc155514170)

[Annexe 2 : Abstract 17](#_Toc155514171)

[Annexe 3 : Les sources 18](#_Toc155514172)

[Annexe 4 : Images du digramme de Gantt 20](#_Toc155514173)

[Annexe 5 : Table des illustrations 22](#_Toc155514174)

[Annexe 6 : Cahier des charges 22](#_Toc155514175)

[Annexe 7 : Proposition de projet SAE S5 22](#_Toc155514176)

[Annexe 8 : Poster 22](#_Toc155514177)

Introduction

# Description du projet : Isidor’s Quest : Chasing the glow

Dans le cadre de notre troisième année de BUT informatique, il nous est demandé de concevoir une application qui est alignée avec nos projets professionnels et nos passions personnelles.

Notre jeu (qui sera accessible au travers d’un navigateur) prendra la forme d’une expérience en 2D avec un style pixel art. Au début d’une nouvelle partie, l’utilisateur sera invité à choisir parmi deux classes de héros : le guerrier et l’archer. Le joueur sera ensuite confronté à des monstres et à des énigmes de difficulté croissante, nécessitant des compétences et de la stratégie pour accéder aux niveaux suivants.

*Rédiger par ZHANG Anxian*

# Pourquoi ce projet et pas un autre ?

Après une longue phase de réflexion, notre choix s’est porté sur le domaine du jeu vidéo qui possède à lui tout seul une part du marché non négligeable ! De plus ce domaine suscite un intérêt majeur au sein de notre équipe. Malgré l’immensité des possibilités de création de jeux, nous avons finalement choisi de nous orienter vers un style emblématique, à l’instar de *Mario*, *Kirby* ou encore *Sonic*. Nous avons tous déjà joué à ces jeux plus petits et ce serait un rêve de pouvoir réaliser un tel type de jeu. De plus, la plupart des langages abordé par le projet ne sont pas tous maîtrisés par les membres de l’équipe. C’est donc aussi une occasion idéale d’enrichir notre CV.

*Rédiger par ZHANG Anxian et* *SELVARATNAM Akash*

Développement

# Phases d’analyses

## Le marché actuel

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. Ce chiffre s’explique par l’augmentation des plateformes disponibles de nos jours, on peut désormais jouer à des jeux assez puissants sur nos smartphones, ce qui n’était pas possible il y a 10 ans. De plus, la diversité de type de jeux attire de nombreux consommateurs, en 2022, il existe un peu moins d'1,8 milliard de joueurs dans le monde.

A screenshot of a screen

Description automatically generated

Figure : Bilan du marché Français 2020 dans le secteur du jeu vidéo. [L’essentiel-jeu-vidéo [Fichier PDF]. Page 8. 2021.](https://www.sell.fr/sites/default/files/essentiel-jeu-video/ejv_mars_21_web.pdf)

*Rédiger par ZHANG Anxian et* *SELVARATNAM Akash*

## Ce qui fait notre singularité

L'intégration d'un jeu sur un site web est courante, comme c'est le cas sur des plateformes comme le site web *jeux.fr*. Cependant, notre jeu sera certes également sur un site web, mais il ne sera pas regroupé avec d'autres jeux. Il s'agit d’un site n’intégrant qu’un seul jeu. Tout le site est là pour accueillir notre jeu. Le site, permet avant même de jouer au jeu, de mettre l’utilisateur dans l’univers et dans l’ambiance de *Isidor’s Quest*.

De plus, sur le plan du design, notre jeu se démarque des autres jeux sur le marché. Certes, il s'agit d'un jeu de plateforme 2D que l'on retrouve fréquemment, mais le nôtre est également composé de pixel art. Viens s’ajouter à cela le mécanisme de défense qui est une mécanique rare dans ce type de jeu. Sans même parler du « village », une zone où les joueurs peuvent effectuer diverses actions auprès des PNJ (Personnages Non-Joueurs), telles que l'amélioration de leur arbre de compétences ou encore l'achat d'objets avec l'or obtenu dans les différents niveaux.

*Rédiger par ZHANG Anxian et* *SELVARATNAM Akash*

## Choix techniques

### Le rôle de chaque composant

#### 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** : Utiliser pour pixeliser une image.

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

*Rédiger par ZHANG Anxian*

#### 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.

*Rédiger par ZHANG Anxian*

#### Gestion et organisation

**Trello** : Organisation des tâches

**Google Drive** : Echange de document

**GitHub** : Gestionnaire de versions

**Discord** : Communication

**Notion** : Création du diagramme de Gantt

**Visual Paradigm** : Mise en place des diagrammes UML

*Rédiger par ZHANG Anxian*

### Pourquoi ces outils plutôt que d’autres ?

#### Jeu

Lorsqu'on parle de développement de jeu, on pense tout de suite à Unity Engine et Unreal Engine, mais quelle est la différence et pourquoi avons-nous choisi Unity ? **Unity** possède une plus grande communauté qu'Unreal, il nous est donc plus facile de nous documenter. De plus, Unity est plus adapté pour les débutants dans le monde du jeu vidéo, contrairement à Unreal qui est beaucoup plus orienté vers des jeux de type AAA (jeux développés et produits par des grandes entreprises de l’industrie du jeu).

Pour les designs, nous nous sommes orientés vers des designs déjà fait et open source sur internet afin de gagner du temps pour le développement. Nous avons donc utilisé des sites tels que **OpenGameArt** et **CraftPix** qui offrent de nombreux éléments graphiques en libre accès. Il nous arrive de trouver des ressources qui ne sont pas en pixelart, c'est pourquoi nous utilisons un logiciel de pixélisation : **Pinetools**, qui est l'une des seules applications que nous avons testées et qui ne déforme pas l'image originale. Si aucune recherche ne correspond au design souhaité, nous le créons alors de zéro avec **Piskelapp**.

*Rédiger par ZHANG Anxian et* *SELVARATNAM Akash*

#### Web

Créer des maquettes d’application web en collaborant avec les autres membres, tout en étant gratuit et accessible à tous ? **Figma** est alors l’outil parfait ! Contrairement à ses concurrents, qui sont payants ou qui deviennent payantes après une phase de test. Figma est un outil facile à prendre en main et permet aussi de réaliser des maquettes pour applications mobiles.

Côté Front-end, nous avons choisi d'utiliser le framework **React Native** qui se base sur le langage Javascript. Nous nous sommes orientés dans cette direction, d'une part parce que cet outil est l'un des plus utilisés et demandés sur le marché, d'autre part, contrairement à d'autres framework, React permet de développer des pages interactives très optimisées grâce aux mises à jour rapides qu'il apporte. Pour finir, contrairement à ReactJS, React Native permet non seulement de développer une application utilisable sur un navigateur, mais aussi de la compiler en tant qu'application mobile.

Nous avons choisi **Node.js**, accompagné du framework Express.js (le langage de programmation est aussi le Javascript) pour faciliter la création et la gestion rapide des serveurs recevant les requêtes HTTP, la gestion des sessions utilisateur, ainsi que l'utilisation des cookies. De plus, il s'agit de l'un des backends les plus utilisés en combinaison avec React.

**MongoDB** est la base de données utilisée pour stocker les données utilisateur. Elle présente une particularité : elle est non relationnelle, ce qui signifie qu'elle ne suit pas le schéma tabulaire sous forme de lignes et de colonnes communément rencontré dans la plupart des systèmes de bases de données. Cette approche offre une plus grande flexibilité et évolutivité. De plus, Node.js propose une bibliothèque appelée Mongoose qui simplifie grandement la réalisation d'opérations sur MongoDB.

Nous travaillons avec l'éditeur de code **Visual Studio Code**, principalement pour sa légèreté et des extensions qu'il propose. Parmi ces extensions, Simple React Snippets se démarque en permettant un développement plus rapide grâce à l'utilisation de mots-clés. Par exemple, pour créer un composant, au lieu de tout saisir manuellement, il suffit de taper "rsc" puis tabulation.

*Rédiger par ZHANG Anxian et* *SELVARATNAM Akash*

#### Gestion et organisation

**Trello** est le meilleur choix pour la gestion de projets simples. Il suffit de créer des tableaux par thèmes généraux et d'y ajouter des étiquettes. Grâce à cette organisation, il devient aisé de suivre l'avancement de notre projet en naviguant d'un tableau à l'autre.

Qui ne connaît pas Google ? Un des GAFAM ! L’entreprise a développé une application, **Google Drive**, qui permet également de collaborer avec les membres de l’équipe en plus de pouvoir créer, éditer et supprimer différents types de fichiers (documents, tableurs, présentations). Nous utilisons cet outil car c’est celui qui offre le plus d’espaces de stockage sans frais, avec 15 GO.

Les outils de versionning sont nombreux, pourtant il y en a certains qui sont plus pertinents que d’autres, comme la gestion de version décentralisée (chaque développeur possède ses propres dépôts et copie locale), ce qui permet à chacun de travailler à son rythme. Nous nous sommes orientés vers Git, et plus précisément **GitHub**. Offrant la possibilité de créer plusieurs branches locales indépendantes, un tableau de bord personnel pour suivre les issues et les pull requests. De plus, si un problème survient, aucun souci, du fait de la grande communauté qui s’y trouve.

**Discord**, un outil de communication, de visioconférence et d’envoi de fichiers, se distingue par sa capacité à évoluer constamment grâce à la possibilité d’ajouter des bots sur nos serveurs. Étant tous sur la plateforme depuis plus de 2 ans, nous l’avons sélectionnée sans hésitation.

Lorsqu’on évoque **Notion**, l’outil no-code de productivité vient immédiatement à l’esprit. Néanmoins, elle permet aussi de faire des diagrammes de Gantt. Cette fonctionnalité proposée par l’application a été notre dernier recours. En effet, nous avons dans un premier temps porté nos recherches sur d’autres applications, mais elles étaient soit payantes, soit mal conçues (constitution du diagramme avec un assemblage de plusieurs rectangles).

Concernant l’outil pour les diagrammes UML, il y a mieux que **Visual Paradigm**, comme Lucidchart. Cependant, son utilisation est restreinte. Nous pouvons uniquement créer 3 documents et pas plus. Alors que Visual Paradigm permet d’ajouter et de confectionner une infinité de fichiers tant que la limite de stockage n’est pas dépassée (1 GO).

*Rédiger par ZHANG Anxian*

## Charte graphique

Comme pour tous les jeux, notre logo est composé de notre personnage, ou du moins d'une partie de celui-ci. Nous avons uniquement gardé la "partie supérieure" de notre héros, simplement pour des raisons esthétiques et ergonomiques. Notre palette de couleurs a justement été reprise de celui-ci. Étant donné que notre jeu est un jeu 2D de style pixel art, rien de mieux qu’une typographie pixelisé.

A screenshot of a video game

Description automatically generated

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

*Rédiger par ZHANG Anxian*

## Notre organisation

### Planification

Avant de commencer le projet, nous avons mis en place un diagramme de Gantt que nous avons intégré sous forme d’un tableau dans le cahier des charges (voir en annexe 6). Bien que celui-ci soit accompagné d’un lien, ce dernier est devenu obsolète, et la lecture du diagramme sur un tableau n’est pas la plus idéale. Nous avons donc ajouté en annexe 4 des images illustrant le diagramme de Gantt. Celui-ci est présent à titre indicatif, mais au cours du projet, nous rajouté des tâches et réalisé des fonctionnalités plus tôt que la date indiquée sur le diagramme.

Lors de la rédaction du cahier des charges, nous avons également consacré un certain temps à la composition du cahier de recette (voir en annexe 6).

*Rédiger par ZHANG Anxian*

### Rôles de chacun(e)

Outre le fait que le projet doit correspondre à nos envies personnelles et professionnelles, c’est avant tout une manière d’apprentissage. Nous avons donc chacun choisi plusieurs rôles, qui nous semblent pertinents (voir en 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 |  |  |  | ✓ |  |

*Rédiger par ZHANG Anxian*

### Répartition des tâches

Dans notre organisation il n’y a pas à proprement parler de chef de projet. Nous utilisons une application de gestion des tâches, Trello, qui nous permet d’ajouter des missions et de les labelliser avec un des termes suivants « Terminé », « Pas possible pour le moment », « En cours », « Bloqué » et « À faire ». Cela signifie qu’une fois qu’une mission a été sélectionnée ou commencée par une personne, elle doit se déclarer en tant que membre et indiquer l’état d’avancement sur la carte afin de ne pas refaire des tâches qui ont déjà été réalisées.

A screenshot of a computer

Description automatically generated

Figure : exemple de tâche effectuée et labellisée « Terminer » (capture d’écran)

*Rédiger par ZHANG Anxian et* *SELVARATNAM Akash*

# Nos deux applications

## Les services proposés (architecture fonctionnel)

### Site

Nos différentes pages web :

* Accueil : introduction à l’histoire du jeu et aux personnages jouables.
* À propos : présente le projet, l’utilité du site et de la gestion des données de l’utilisateur.
* Connexion : l’utilisateur doit se connecter pour accéder au jeu.
* Inscription : s’il ne possède pas de compte, il peut s’inscrire.
* Oublie de mot passe : permet la réinitialisation du mot de passe.
* Modification des données utilisateur : offre la possibilité de modifier le pseudonyme ainsi que le nom et prénom associés à l'utilisateur.
* Jeu : contient notre jeu Unity au format WebGL.
* Paiement : propose deux différents types de paiement (PayPal ou par carte bancaire).
* Contact : si l’utilisateur a besoin d’une quelconque aide, il peut nous solliciter sur cette page.
* Vérification : Page permettant de vérifier le code entré par l’utilisateur avec celui envoyé sur son mail lors de l’inscription ou lors d’une modification de mot de passe.

Les différentes librairies utilisées :

* Cors : librairie permettant de personnaliser l’en-tête (origin, content-type, credentials) pour pouvoir se connecter du coté backend.
* Express : Framework permettant de construire des applications web pour Node.js, il s’agit d’un Framework permettant d’exécuter des requêtes http (POST, GET, PUT …).
* Express-session : librairie proposée par Express permettant de sauvegarder des données directement dans un cookie.
* localStorage : librairie permettant de stocker des données directement dans le local.
* Mongoose : mongoose est une bibliothèque JavaScript permettant de créer une connexion entre MongoDB et l’application codée en JS. Il permet d’exécuter différentes opérations sur la base de données (Read, Create, Update, Delete)
* Nodemailer : nodemailer est un module de Node.js permettant d’envoyer des mails.
* paypal-rest-sdk : librairie proposée par PayPal permettant de réaliser des paiements par PayPal pour les développeurs, le mode test est disponible avec cette librairie.
* Stripe : librairie proposée par Stripe permettant de réaliser différents types de paiement, le mode test est disponible avec cette librairie.

*Rédiger par ZHANG Anxian et* *SELVARATNAM Akash*

### 

### Jeu

Nos différentes scènes :

* Accueil : offre la possibilité de commencer une partie ou de configurer les paramètres du jeu, comme le mode plein écran ou le réglage du volume.
* Sélection de personnages : permet de choisir entre les deux classes disponibles, à savoir **l’archer** et le **guerrier**
* Village : zone non hostile où les joueurs peuvent **améliorer leur arbre de compétences** ou **acheter des objets consommables**, notammentles potions.
* Les niveaux de la première région :
  + Niveau 1 : premier niveau du jeu
  + Niveau 2 : deuxième niveau du jeu

*Rédiger par ZHANG Anxian et* *SELVARATNAM Akash*

## Nos phases de développement

Nous avons commencé par une **recherche et la définition** des objectifs à travers un premier document : « Proposition de projet SAE S5 », qui se trouve en annexe 7. Ensuite, nous avons rédigé le cahier des charges en y intégrant la **planification**. Entre-temps, une personne a mis en place une page Figma pour créer une première représentation du site web. En ce qui concerne le jeu, certaines personnes ont conçu l'UI et l'UX des niveaux, ainsi que les menus directement sur Unity. Enfin, après avoir plus ou moins terminé la conception esthétique des applications, nous sommes passés à la phase de **développement**, puis effectué des **tests unitaires** avec la bibliothèque Jest sur les composants du site.

*Rédiger par ZHANG Anxian*

## Architecture technique (diagramme UML)

A diagram of a diagram

Description automatically generated

Figure : diagramme d'état-transition du joueur

A diagram of a diagram

Description automatically generated

Figure : diagramme d’état-transition des ennemies du jeu

## Conception des applications (explication)

### Site

Termes techniques (à ajouter plus tard dans le glossaire):

* useState : détecte localement (limité à des composants) si un état a été modifié
* useEffect : indique à React l’exécution d’une tâche à chaque fois qu’un état change, il gère donc les effets de bord
* TextInput : champ d’entrée de texte
* Text : composant permettant d’afficher du texte
* TouchableOpacity : permet de rendre un élément cliquable comme un bouton ou un lien par exemple, avec un effet d’opacité lors du clique.
* useNavigation : permet de naviguer entre les différentes pages du site web
* Regex : Un regex est une chaine de caractères représentant un ensemble de chaine de caractère possible.
* Nodemailer : Nodemailer est un module de Node.js permettant d’envoyer des mails.
* Express : Framework permettant de construire des application web pour Node.js, il s’agit d’un Framework permettant d’exécuter des requêtes HTTP
* Express-session : Librairie proposé par Express permettant de sauvegarder des données directement dans un cookie
* useRef : permet de référencer une valeur (composants, variables…)
* paypal-rest-sdk : Librairie proposé par PayPal permettant de réaliser des paiement PayPal
* Stripe : Librairie proposé par Stripe permettant de réaliser différents type de paiement
* Mongoose : Mongoose est bibliothèque JavaScript permettant de créer une connexion entre MongoDB et l’application codé en JS. Il permet de créer des documents ou de créer des requête (find, update, delete …) sur notre document MongoDB.
* SelectDropdown : composant proposé par React-native permettant de réaliser des liste avec plusieurs items cliquables.
* Image : composant proposé par React-native permettant d’afficher une image.

Accueil :

La page d'accueil est la première interface que voit l'utilisateur, elle présente le contexte du jeu, les personnages et comment accéder au jeu. Les utilisateurs peuvent naviguer vers différentes interfaces à partir de la page d'accueil. Le fond de la page d'accueil est une image que nous avons créée, avec des éléments de jeu. En ajustant les distances entre chaque segment de texte et l'image de fond à l'aide d'une vue vide, nous avons créé une mise en page harmonieuse. Les boutons de la page d'accueil et les boutons de l'en-tête sont identiques, réalisés avec TouchableOpacity. Lorsqu'un utilisateur non connecté appuie dessus, il est redirigé vers l'écran de connexion. Pour un utilisateur connecté mais non payé, la redirection se fait vers l'écran de paiement. Seuls les utilisateurs connectés et ayant effectué le paiement sont redirigés vers l'écran de jeu.

手机屏幕的截图

中度可信度描述已自动生成

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

*Rédiger par LIN Xingtong*

Inscription :

Lors de l’inscription, l’utilisateur doit remplir un formulaire comportant différents champs tels que le nom, le prénom, l’adresse mail, le nom d’utilisateur ainsi que le mot de passe. Si l’ensemble de ces champs sont correctement remplie par l’utilisateur, alors l’utilisateur sera redirigé vers la page de vérification de code, si le mail entré par l’utilisateur existe dans notre base de données alors l’utilisateur sera invité à se connecter avec son compte plutôt que de s’inscrire, si le pseudo entré par l’utilisateur existe dans notre base de données, alors l’utilisateur devra changer son pseudo car nous avons opté avec l’équipe pour un nom d’utilisateur unique et si le mot de passe entré par l’utilisateur est différent de celui indiqué dans le champs de confirmation de mot de passe alors un message d’erreur indiquera à l’utilisateur que les deux mot de passe ne sont pas identique. Cette page a été réalisée avec les composants TextInput pour les six différents champs d’entrée, un composant Text pour le titre « S’inscrire » et un TouchableOpacity pour le bouton. Nous utilisons également les hooks useState pour pouvoir gérer les différents états des variables et useEffect pour pouvoir gérer l’état du bouton (activer si l’ensemble des champs sont remplie sinon désactiver) ainsi que de la gestion de l’état du mail qui doit comporter un texte avant et après le caractère @ et d’un point avec deux ou trois caractères à la suite, cette vérification du mail est possible avec un regex.

Une image contenant texte, capture d’écran

Description générée automatiquement

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

*Rédiger par SELVARATNAM Akash*

Vérification du code :

Après avoir rempli les champs pour que l’utilisateur puisse s’inscrire, l’utilisateur sera redirigé vers une page permettant de vérifier que le code indiqué par l’utilisateur et bien conforme à celui envoyé sur sa boîte mail. Il s’agit d’une procédure permettant de vérifier que le mail indiqué par l’utilisateur sur le formulaire d’inscription existe bien. Cette page a été réalisée avec les composants TextInput pour les deux différents champs d’entrée, un Text pour le titre « Vérification du code » et un TouchableOpacity pour le bouton. Nous utilisons également les hooks useState pour pouvoir gérer les différents états des variables et useEffect pour pouvoir gérer l’état du bouton.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

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

Le code est envoyé sur la boîte mail de l’utilisateur avec la bibliothèque nodemailer de node.js, il s’agit d’un code d’une taille de six caractères générés aléatoirement. Si le code entré par l’utilisateur est correct, cela créera son compte dans le document User de notre base de données avec les données fourni par l’utilisateur (le mot de passe fournie par l’utilisateur sera crypté dans la base de données) puis l’utilisateur sera redirigé vers la page d’accueil avec un useNavigation, dans le cas contraire un message d’erreur s’affiche indiquant que le code entré est incorrect.

A screenshot of a chat

Description automatically generated

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

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

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

*Rédiger par ZHANG Anxian et* *SELVARATNAM Akash*

Connexion :

L’utilisateur est invité à entrer son nom d’utilisateur ainsi que son mot de passe, si les données sont correctes alors l’utilisateur sera redirigé vers la page d’accueil et nous enregistrerons dans les cookies au niveau du back-end, le pseudo de l’utilisateur avec la bibliothèque express-session, cela permet de savoir si l’utilisateur est connecté ou non. En cas d’erreur, le nom d’utilisateur ou le mot de passe est incorrect (différent de ce qui est présent dans la base de données) alors un message d’erreur est affiché sur les deux champs, l’utilisateur devra alors remplir à nouveau le formulaire, ou s’inscrire via le bouton « Créer un nouveau compte », ou se diriger vers la page d’oubli de mot de passe via le lien « Mot de passe oublié ? ». Cette page a été réalisée avec les composants TextInput pour les deux différent champs d’entrée, un Text pour le titre « Connexion » et trois différents TouchableOpacity pour les boutons et le lien menant vers la page d’oubli de mot de passe. Nous utilisons également les hooks useState pour pouvoir gérer les différents états des variables et useEffect pour pouvoir gérer l’état du bouton « Se connecter ».

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquement

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

*Rédiger par SELVARATNAM Akash*

Oublie de mot passe :

Dans un premier temps, l’utilisateur est invité à saisir son courriel. L’adresse électronique doit être correcte, comportant un arobase et un nom de domaine. Tant que celle-ci est incorrecte, le bouton « Obtenir le code de vérification » restera désactivé. Cette fonctionnalité a été implémentée avec un useEffect, qui va faire une vérification à chaque mis à jour du TextInput par l’utilisateur.

A blue box with white text

Description automatically generated

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

Lorsque le bouton est appuyé, l'utilisateur sera redirigé vers la page de vérification du code qui lui sera envoyé dans sa boîte mail. Une fois la vérification réussie, il pourra alors modifier son mot de passe. Ce changement sera traité au niveau du backend, c'est-à-dire, lorsque le bouton de modification sera pressé, une requête HTTP sera envoyée au serveur qui mettra à jour la base de données.

*Rédiger par ZHANG Anxian*

A propos :

La page à propos présente le contexte du projet, l’utilité du site ainsi que la gestion des données de l’utilisateur. Pour pouvoir réaliser cette page, nous avons utilisé différents composants React telles que le composants Text pour pouvoir présenter notre projet ainsi que la création de notre propre composant pour pouvoir réaliser la table de contenus qui nécessite le composant touchableOpacity pour les liens. Nous avons également utilisé le Hook useRef pour pouvoir naviguer vers le bloc correspondant lorsqu’on clique sur la table de contenus.

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Figure : Page à propos (capture d’écran)

*Rédiger par SELVARATNAM Akash*

Contact :

L’utilisateur est invité à remplir les différents champs du formulaire de contact avec son mail, son nom ainsi que le message à envoyer à notre service. Cette page utilise différents hooks tels que le useState pour gérer les valeurs des différentes variables ainsi qu'un useEffect pour pouvoir gérer l’état du bouton, nous utilisons le composant Text pour le titre « Formulaire de contact », les composants TextInput pour les différents champs d’entrée et un TouchableOpacity pour le bouton.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Figure : Page de contact (capture d’écran)

Lorsqu’e l’utilisateur remplie l’ensemble des champs du formulaire et appuie le bouton envoyer cela envoie un mail à notre service avec le message de l’utilisateur à l’aide de la bibliothèque nodemailer et un mail automatique est envoyé au mail indiqué par l’utilisateur sur le formulaire afin de confirmer la bonne réception du mail.



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

*Rédiger par SELVARATNAM Akash*

Paiement :

Pour pouvoir acheter notre jeu, nous avons décidé de deux différentes moyen de paiement par carte bancaire ou par PayPal. Pour pouvoir réaliser cette page, nous avons utilisé deux TouchableOpacity pour les deux différents boutons.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Figure : Page de paiement (capture d’écran)

Lorsqu’on appuie sur le bouton « Payer par PayPal, cela nous redirige vers la page de connexion de paiement de PayPal puis après s’être connecte à son compte, l’utilisateur devra acheter le jeu sur PayPal, l’ensemble des opération, la création d’un paiement, la réussite ou l’échec d’un paiement est géré par la bibliothèque paypal-rest-sdk proposé par PayPal.

Lorsqu’on appuie sur le bouton « Payer par carte bancaire », cela nous redirige vers la page de paiement de Stripe, l’utilisateur sera invité à remplir le formulaire en fournissant son mail ainsi que les données de sa carte bancaire. L’ensemble des opération, la configuration et la création d’un paiement, l’échec ou la réussite d’un paiement est géré par la bibliothèque Stripe.

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

Donnée test pour le paiement par Stripe :

- Le numéro de la carte bancaire : 4242 4242 4242 4242

- Date d’expiration : une data supérieure à la data actuelle

- CVC : écrivez un code à trois chiffres. (Exemple : 123)

Une image contenant texte, capture d’écran, Page web, Police

Description générée automatiquement

Figure : Paiement par PayPal (capture d’écran)

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Figure : Paiement par carte bancaire (capture d’écran)

Si le paiement à bien été effectué alors cela dirigera l’utilisateur vers la page indiquant que le paiement a bien été effectué avec un bouton permettant de télécharger sa facture pour le paiement par carte bancaire, pour le paiement par PayPal, la facture est directement envoyée sur son mail.

En cas d’échec de paiement, l’utilisateur sera dirigé vers la page indiquant que le paiement a été refusé.

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquement

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

*Rédiger par SELVARATNAM Akash*

Modification des données utilisateur :

Il s’agit d’une page permettant aux utilisateurs de consulter leurs informations personnelles (email, prénom, nom et pseudo) et qui permet également de modifier leurs données. Cette page contient trois TextInput pour les trois différents champs d’entrée afin que l’utilisateur puisse consulter ainsi que modifier leurs données, des composants Text pour le titre « Informations personnelles » et pour son adresse mail, un touchableOpacity pour le bouton. Cette page contient également des hooks, des useState pour gérer les valeurs des variables ainsi que des useEffect pour pouvoir appeler la méthode cotée backend permettant de modifier les données de l’utilisateur sur la base de données dans le document User à l’aide de la bibliothèque mongoose.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

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

Pour pouvoir modifier les données, il faut appuyer le bouton Mise à jour, on peut ensuite éditer les différents champs d’entrée et lorsqu’e l’utilisateur à terminer de modifier ses données alors il peut appuyer le bouton valider qui va modifier via une requête cotée backend, les données de l’utilisateur dans le document User de notre base de données.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Figure : Modification des données (capture d’écran)

*Rédiger par SELVARATNAM Akash*

Header (composant commune à toute les pages):

Le header de notre site est présente sur l’ensemble des pages de notre site web, il est composé de notre logo réalisé avec le composant TouchableOpacity, lorsqu’on appuie sur le logo, il nous dirige vers la page d’accueil, ensuite, nous avons le lien qui mène vers la page à propos qui est également réalisé avec le composant TouchableOpacity.



Figure : Header (capture d’écran)

Quand, on appuie sur le bouton Fr cela nous affiche une liste de deux items (Français, English), il s’agit des deux langues disponible pour la traduction de notre site web, pour pouvoir réaliser cette fonctionnalité, nous avons mis en place deux fichier JSON (Français et Anglais), chaque texte sur notre site web est appelé dans ces fichier JSON, lorsqu’e l’utilisateur sélectionne la langue Française alors une mise a jour de la langue est effectuer sur le site web dans un useState qui prend comme valeur le fichier json Français. Cette liste a été réalisé avec le composant SelectDropdown proposé par React-Native.



Figure : Liste des langues disponible sur notre site web (capture d’écran)



Figure : Fichier JSON Français (capture d’écran)



Figure : Fichier JSON Anglais (capture d’écran)

Le bouton de connexion/déconnexion est également une liste qui propose une connexion lorsqu’e l’utilisateur est déconnecté et une déconnexion quand l’utilisateur est connecté. Il permet également de se diriger vers la page de profil permettant de consulter ou de modifier ses données. Il s’agit d’une liste également réalisé avec le composant SelectDropdown.



Figure : Connexion (Header, capture d’écran)

Pour finir, le bouton jouer permet de se diriger vers la page de connexion si l’utilisateur n’est pas connecté ou de se diriger vers la page de paiement si l’utilisateur est connecté, mais qu’il n’a pas acheté le jeu. Si l’utilisateur est connecté et qu’il a acheté le jeu alors il sera redirigé vers la page de jeu. Il s’agit d’un bouton réalisé avec le composant TouchableOpacity.

*Rédiger par SELVARATNAM Akash*

Footer (composant commune à toute les pages) :

Le Footer est présent sur l’ensemble des pages du site web, il est composé de notre logo qui mène vers la page d’accueil quand on cliquer dessus, il à été réalisé avec les composants Image et TouchableOpacity, nous avons également mis en place plusieurs liens également réalisé avec le composant TouchableOpacity permettant de naviguer vers la page de contact, à propos, les mentions légales et les différents réseaux sociaux du jeu.

Une image contenant texte, capture d’écran, Système d’exploitation

Description générée automatiquement

Figure : Footer (capture d’écran)

*Rédiger par SELVARATNAM Akash*

### Jeu

Les parties sont a rédiger par les développeur des fonctionnalités

Termes techniques (à ajouter plus tard dans le glossaire):

* Sprites : élément graphique qui peut se déplacer sur l’écran (personnages, coffre, pièce …).
* JSON : fichier textuel conçu pour l’échange de données.
* Panel : élément permettant réaliser l’UI et l’UX, en fournissant tous les outils nécessaires (bouton, champ de texte etc).
* GameObject : un objet qui se trouve dans une scène Unity
* Script: fichier contenant le code source d’un objet ou fonctionnalité
* Paramètre : variable de type connu mais ne sont précisés qu'au moment de l'exécution.
* MoveTowards : une fonction qui déplace progressivement un objet d'une position à une autre dans l'espace en fonction d'une vitesse donnée.
* OnCollisionEnter2D : une méthode appelée automatiquement lorsqu'un objet entre en collision avec un autre dans un environnement en deux dimensions (2D) dans Unity.
* OnCollisionExit2D : une méthode appelée automatiquement lorsqu'un objet quitte une collision avec un autre dans un environnement en deux dimensions (2D) dans Unity.
* StartCoroutine : une fonction utilisée pour démarrer l'exécution d'une routine (méthode) en parallèle avec le reste du programme, souvent utilisée pour créer des actions temporisées ou des animations dans Unity.
* Destroy : une fonction qui détruit un objet ou un composant dans Unity, libérant ainsi les ressources associées à cet objet.
* Render Texture : une technique qui permet de capturer et d'afficher la sortie graphique d'une caméra dans Unity.
* orthographicSize : une propriété de la caméra qui définit la taille de la vue orthographique, influençant ainsi le zoom de la caméra.
* Tilemap : une grille que l'on utilise pour “peindre” le décor d'un jeu.
* Prefab : Le prefab est un composant permettant de contenir un GameObject avec ses composants et ses propriété.

Écran d’accueil :

Notre écran d'accueil est composé de plusieurs panneaux. Il y a donc plusieurs couches d'UI qui se superposent. Celle qui se trouve le plus en arrière, c'est l'image en arrière-plan que vous voyez sur la figure ci-dessous. Ensuite, nous retrouvons le titre du jeu et les boutons « Start game » et « Settings ».

A video game screen with text and a ladder

Description automatically generated

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

La fonctionnalité du bouton « Start game » a été réalisée en y ajoutant un événement. Comme en JavaScript, le C# permet aussi l'ajout d'événement avec *onClick.addListener(maFonction)*. Une fois le bouton cliqué, le jeu chargera alors la scène de sélection de personnage, rendue possible avec la ligne de code *SceneManager.LoadScene(maScèneACharger)* qui se trouve dans *maFonction*.

La mise en place de l’événement *onClick* sur le bouton « Setting » s’est faite de la même manière que pour le bouton « Start game ». Sauf que cette fois, lors du click, ce n’est pas une scène qui apparaîtra, mais un menu (voir la figure ci-dessous).

A screenshot of a video game

Description automatically generated

Figure : menu de configuration (capture d'écran)

La fonctionnalité du plein écran a été réalisée en ajoutant un événement de type *OnValueChanged(maFonction)*. La fonction fournie en entrée (qui est ici *maFonction*) de l'événement fait ceci : *Screen.fullScreen = étatDeLaCase*, ayant pour but de changer à chaque clic réalisé sur la case. Sur la même logique, le volume changera en fonction de l'entier fourni (entre -80 et 0) grâce à la position de la barre de son.

*Rédiger par ZHANG Anxian*

Scène de sélection des personnages : (Parler de la sélection)

La scène de sélection de personnages permet de sélectionner le personnage parmi les 2 classes disponible (guerrier et archer), pour chacun des personnages, on affiche l’ensemble des statistiques (défense, attaque, vitesse, point de vie) grâce à une fonction permettant de récupérer les statistiques initiales de la classe concerné dans un fichier JSON. Pour afficher, les deux différentes classes, nous avons réalisé un tableau qui contient les sprites du guerrier et de l’archer, lorsqu’ e le joueur appuie sur la flèche, une fonction est exécutée permettant de sélectionner le personnage adéquat grâce à l’indice du tableau ainsi que les statistiques de la classe concernée. Quand le joueur appuiera sur le bouton « Select », il naviguera vers la scène du niveau 1 avec le personnage qu'il a choisi.

Une image contenant texte, capture d’écran, Rectangle

Description générée automatiquement

Figure : Scène de sélection de personnage (Capture d’écran)

*Rédiger par SELVARATNAM Akash*

Fonctionnalités dans l’ensemble des niveaux:

* Barre de vie (Joueur et ennemis)

Pour la barre de vie du joueur et des ennemis, on peut apercevoir une barre de couleur verte possède un attribut nommé fill amount permettant de gérer le taux de remplissage de cette barre (par exemple : Si l’attribut fill amount est égale à 0.5 alors la barre verte sera remplie à 50%).

Une image contenant capture d’écran, Logiciel de graphisme, Logiciel multimédia, Logiciel de jeu vidéo

Description générée automatiquement

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



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

Pour pouvoir calculer le taux de remplissage de la barre de vie du joueur ou de l’ennemie, nous avons mis en place une formule :

Point de vie maximale du joueur ou de l’ennemi : 60

Point de vie actuelle du jouer ou de l’ennemi : 35

60 \* 0.01 float = 0.6 float, ce résultat présente les points de vie maximale sous format décimal (pourcentage).

35 / 0.6 float = 58.33 float, ce résultat présente les points de vie actuel sous format d’un pourcentage.

58.33 float / 0.01 float = 0.5833 float, ce résultat présente les points de vie du joueur sous format décimal (pourcentage).

Ensuite, nous attribuons le résultat 0.5833 float à l’attribut Fill amount afin de bien remplir la barre de vie en fonction des points de vie du joueur ou de l’ennemie.

*Rédiger par SELVARATNAM Akash*

* Barre de chargement d’attaque

Pour la barre de chargement d’attaque, nous utilisons cette fois-ci une barre de couleur bleue permettant d’indiquer au joueur s’il peut attaquer ou non, cette barre doit être complète pour que le joueur puisse attaquer, comme pour la barre de vie, cette barre contient l’attribut fill amount.

Pour pouvoir indiquer la valeur de l’attribut fill amount, nous utilisons une fonction permettant de soustraire le temps d’attaque maximale de la classe de joueur concerné avec le temps d’attaque actuel de la classe de joueur concerné puis transformer cette valeur sous format décimale (pourcentage) et attribuer cette valeur à l’attribut Fill amount afin de bien remplir la barre de chargement d’attaque en fonction du temps d’attaque du joueur. Par exemple, si le joueur attaque et que le temps maximum est à 2 secondes et que cela fait qu’une seconde que le joueur attend alors sa barre de chargement d’attaque sera remplie à 50%.



Figure : barre de vie à 50%

* Les ennemis
  + Système de patrouille

Les ennemis se déplacent de manière aléatoire au cours du jeu, cette fonctionnalité étant implémentée par un algorithme. L'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 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.

电子设备的屏幕

中度可信度描述已自动生成

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

Lorsqu’e le joueur entre dans le périmètre de l’ennemie, on désactive l’algorithme permettant de réaliser une patrouille entre deux position donnée puis nous utilisons une méthode proposé par Unity nommé Vector2.Distance(position1, posistion2) qui prend en paramètre, dans notre cas la position du joueur et la position de l’ennemis et retourne un résultat en float qui détermine la distance entre le jouer et l’ennemi, si la distance est inférieur à 10.0 float et que l’ennemis et le joueur soit à la même hauteur alors cela exécutera un algorithme permettant à l’ennemis 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 désactivera et activera l’algorithme permettant à l’ennemi de réaliser une patrouille entre les deux positions fourni en paramètre de la fonction.

*Rédiger par LIN Xingtong et SELVARATNAM Akash*

* Inventaire

L’inventaire permet de stocker divers objets jusqu’à un nombre maximale de 4 objets. Les différents objets, principalement des potions seront présentées dans une future partie, celle du magasin. L’interface de l’inventaire apparaît quand le joueur appuie sur la touche « I », faisant apparaitre alors la fenêtre suivante :

A screenshot of a video game

Description automatically generated

Figure : Inventaire (capture d’écran)

On peut apercevoir sur cette capture d’écran 3 potions qui sont stockées dans l’inventaire 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, ceci permet lors d’un ramassage d’un objet de la part du 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 assignent en tant que ses enfants dans la hiérarchie des objets.

*Rédigé par Oscar*

* Coffres et pièces en or

Au cours de l’expérience de jeu du joueur, il pourra faire la rencontre de coffre et de pièces en or. Les pièces en or sont installées directement dans la disposition du niveau en tant qu’objet, apparaîtra lorsqu’un ennemi soit vaincu ou lors de l’ouverture d’un coffre.

Le coffre un aussi un objet disposé à l’avance dans le niveau qui s’ouvre au contact avec le joueur. L’animation d’ouverture se lançant ainsi qu’un nombre aléatoire de pièces ainsi qu’une potion de soin sont projeté en l’air que le joueur peut récupérer au contact.

La collision des pièces, coffre et potions est gérée par l’événement OnTriggerEnter2D(Collider2D other) qui s’active quand un nouveau Collider2D soit une zone de collision rentre avec celle de l’objet actuel, ce nouveau Collider2D sera alors le paramètre de la fonction et on vérifiera si ce Collider2D appartient bien à celui du joueur.

*Rédigé par Oscar*

Environnements : (Plateforme et échelle portail de fin de niveau et l’eau)

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.

图片包含 地图

描述已自动生成

Figure : Un cas d'utilisation de la mini-carte (capture d'écran)

*Rédiger par LIN Xingtong*

Menu de pause :

Sauvegarde des données :

Pour la sauvegarde des données, nous avons réalisé deux types de sauvegarde, nous réalisons une sauvegarde de l’ensemble des donnée d’une partie joué par un joueur, il s’agit d’une fonction permettant de récupérer l’ensemble des données d’une partie (le pseudo du joueur, le niveau, le nom du personnage choisi, le nombre de pièces récupéré, 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) et ensuite nous appelons une fonction coté backend afin de pouvoir enregistrer les données récupéré dans la base de données sur le document Game.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure : Document Game

Nous avons également réaliser une sauvegarde des données du joueur sur le jeu grâce à une fonction permettant de récupérer le nombre de pièce du joueur au total, les niveaux des différents capacités de l’archer et du guerrier sous format d’un tableau, ensuite l’ensemble de ces données sont transmis a une fonction coté backend permettant de d’enregistrer les données dans le document UserGame de notre base de données.



Figure : Document UserGame

*Rédiger par SELVARATNAM Akash*

Son :

Pour la gestion des sons sur le jeu, nous avons utilisé un prefab contenant plusieurs composant nommé Audio source qui contient différents sons du jeu (le son du joueur qui se déplace, le son du joueur lorsqu’il prend des dégâts …).

Le script nommé Game Sound contient de nombreuses fonctions permettant d’activer un son avec la fonction Play(), de savoir si un son est activé ou non avec la fonction isPlaying ou de mettre le son en pause avec la fonction Pause() , grâce à une fonction retournant une valeur booléen, les sons sont activé lorsqu’une action se déroule telles que l’attaque d’un joueur ou lors de l’ouverture d’un coffre par exemple.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

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

*Rédiger par SELVARATNAM Akash*

Niveau 2 :

Après avoir terminé le niveau 1, l'utilisateur passera au niveau 2, qui sera légèrement plus complexe, introduisant 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, est 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.

* 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.

游戏机里面的人物

低可信度描述已自动生成

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

* 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.

电脑萤幕画面

中度可信度描述已自动生成

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

*Rédiger par LIN Xingtong*

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 : village du jeu (capture d'écran)

*Rédiger 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.

A screenshot of a video game

Description automatically generated

Figure : menu d'amélioration de l'arbre de compétence (capture d’écran)

*Rédiger 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).

A screenshot of a computer menu

Description automatically generated

Figure : références fournis pour le Scripte du menu de l’arbre de compétence (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.

*Rédiger par ZHANG Anxian*

* Magasin

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

A screen shot of a game

Description automatically generated

Figure : Magasin (capture d'écran)

Le menu du magasin affiche le nombre de pièce du joueur ainsi que les potions proposées avec leur coût. Une fonction a été réalisé pour l’achat d’une potion, cette fonction prends en paramètre le prix de l’objet ainsi que l’objet en lui-même. Elle vérifie d’abord que l’utilisateur a assez d’argent puis que son inventaire n’est pas rempli pour finaliser l’achat de la potion le rajoutant dans l’inventaire du joueur et lui retirant la somme de l’objet.

La fonction est appelé lors d’un événement OnClick qui est affecté aux images de chacune des potions, ses images sont affectées par un bouton qui lance alors la fonction spécifique pour la potion en particulier.

A screenshot of a computer

Description automatically generated

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

*Rédigé par Oscar*

* Portail de sélection de niveau

A screenshot of a video game

Description automatically generated

Figure : portail de sélection de niveau (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 a été réalisée en ajoutant un événement *onClick* à chaque bouton, utilisant la fonction *Scene.LoadScene(maScene)*.

*Rédiger par ZHANG Anxian*

## Incorporation du jeu sur le site

Termes techniques (à ajouter plus tard dans le glossaire):

* WebGL : WebGL est une technologie permettant d’afficher des élément graphiques en 2D ou en 3D sur le navigateur web.

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 diriger dans la section build et setting de l’application Unity, ensuite choisir dans quelle plateforme, on souhaite convertir notre jeu, dans notre cas le format WebGL, et pour finir, il faut appuyer le bouton build afin qu’Unity génère notre jeu sous format WebGL.



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

Quand Unity à terminer la configuration du WebGL, il nous crée deux différents dossiers :

- Build : dossier qui contient l’ensemble des fichiers de configuration permettant de compiler le jeu sous format WebGL.

- TemplateData : dossier qui contient des images (logo Unity, plein écran) ainsi que du style CSS pour le WebGL.

Le fichier index.html est l’aspect visuel de la compilation du WebGL, il utilise les fichiers des dossier Build pour pouvoir configurer, rendre un aspect visuel au jeu et permettre au joueur de jouer au jeu directement sur le site web ainsi que le dossier TemplateData pour afficher les images telles que le logo Unity par exemple.



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

Pour pouvoir lancer le jeu, on peut utiliser l’extension proposé par Visual Studio code nommé Live Server permettant de lancer le jeu sous format WebGL en local en appuyant sur le bouton Go Live.

Une image contenant texte, capture d’écran, Police, Bleu électrique

Description générée automatiquement

Figure : Extension Live server (capture d’écran)

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



Figure : Page jeu de note site web version code (capture d’écran)

Voici le rendu visuel de notre jeu sous format WebGL sur la page jeu de notre site web, l’ensemble des fonctionnalité du jeu sont présente sur la compilation du WebGL sur notre site web.

Une image contenant texte, capture d’écran, Logiciel multimédia, logiciel

Description générée automatiquement

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

*Rédiger par SELVARATNAM Akash*

## Les tests utilisateurs

Conclusion

# Bilan de montée en compétences

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 donnée dans les cookie 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.

*Rédiger par SELVARATNAM Akash*

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.

*Rédiger par LIN Xingtong*

# Les difficultés rencontrées

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 réunion 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 taches 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.

*Rédiger par SELVARATNAM Akash*

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 ecrire 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.

*Rédiger par LIN Xingtong*

# Ce qui reste à 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 positions 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 avions envisagé la création de sept mondes différents, chacun comprenant trois niveaux. Cependant, en raison de la charge de travail à la fois sur la partie web et la partie jeu, ainsi que de notre volonté commune de développer diverses fonctionnalités telles que le menu pause, la mini-carte, et même une fonctionnalité qui n'a pas pu être réalisée à la fin : 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, tout comme le cas précédent.

En ce qui concerne 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 réellement mettre en œuvre 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 il n'y a pas de modèle d'intelligence artificielle.

*Rédiger par LIN Xingtong*

# Perspectives d’amélioration

Premièrement, l'organisation, notamment la répartition des tâches. Au début, nous n'avions pas l'habitude d'utiliser Trello et de mettre à jour les tâches dessus. Ainsi, il est arrivé souvent que plusieurs d'entre nous travaillent sur la même tâche en même temps, ce qui fait perdu du temps.

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

Enfin, c'est dommage que nous n'ayons pas pu découvrir Jira plus tôt, car Jira peut générer automatiquement une vue chronologique sous la forme d'un diagramme de Gantt. Ainsi, 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 de plus de temps pour le développement de notre projet.

*Rédiger par LIN Xingtong*

# List des annexes

[Annexe 1 : Résumé 9](#_Toc154782996)

[Annexe 2 : Abstract 9](#_Toc154782997)

[Annexe 3 : Les sources 9](#_Toc154782998)

[Annexe 4 : Images du digramme de Gantt 9](#_Toc154782999)

[Annexe 5 : Table des illustrations 9](#_Toc154783000)

[Annexe 6 : Cahier des charges 9](#_Toc154783001)

[Annexe 7 : Poster 9](#_Toc154783002)

# Annexe 1 : Résumé

# Annexe 2 : Abstract

# 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)>

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/fr/docs/Learn/Server-side/Express_Nodejs/Introduction#introduction_%C3%A0_express)

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

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

# Annexe 4 : Images du digramme de Gantt

**A screenshot of a computer

Description automatically generated**

Figure : Digramme de Gantt, première partie (capture d’écran)

A screenshot of a computer

Description automatically generated

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

# Annexe 5 : Table des illustrations

# Annexe 6 : Cahier des charges

# Annexe 7 : Proposition de projet SAE S5

# Annexe 8 : Poster