Cahier des charges

Treep

Kromm Studio

Kristen Couty - Romain Dischamp - Oscar Cornut
Mahé De Berranger - Milo Moisson

D1

Octobre 2024

Tables des matières

1 Introduction

Ce cahier des charges a pour mission de définir et de présenter en détail les modalités de réalisation du projet en accord avec les critères et les attentes du client. Il constitue le premier pas vers la concrétisation du jeu vidéo *Treep*, un Rogue-like en deux dimensions. Il est possible à l'avenir que les objectifs qui y sont définis évoluent et soient légèrement modifiés selon l'état de son avancement. Bien entendu, ces éventuelles modifications n'impacteront en rien la finalité attendue du projet.

Dans *Treep*, le joueur incarne l'âme d'une gigantesque colonie d'insectes vivant au pied d'un arbre majestueux. Cependant, cet écosystème autrefois prospère est menacé par un champignon parasite, guidant inévitablement la colonie à sa perte. L'objectif est donc de parvenir à le sauver pour assurer l'avenir de la colonie. Pour cela, vous allez être confrontés à une armée d'insectes déchus, parasités par le champignon, protégeant le cœur du parasite. Malheureusement, aucun combat ne se gagne sans sacrifice et tous les insectes de la colonie n'y survivront pas. *Treep* vous plonge dans un cycle de mort et de réincarnation vous laissant ainsi plusieurs opportunités pour sauver la colonie. *Treep* est à la fois une aventure dynamique et un hommage à des mécaniques de jeu intemporelles.

Kromm Studio est une entreprise française et indépendante de développement de jeux vidéos récemment créé par cinq amis : Milo Moisson, développeur senior et chef de projet de l'équipe, ainsi que Romain Dischamp, Mahé De-Berranger, Oscar Cornut et Kristen Couty. Ainsi, l'équipe de développement de *Kromm Studio* est constituée de passionnés et d'experts dans leurs domaines respectifs, chacun apportant à l'équipe une expertise complémentaire, nous permettant de créer un jeu innovant. Nous avons l'ambition de nous démarquer sur la scène indépendante en créant un jeu qui allie parfaitement plaisir et innovation, sans oublier nos racines, l'amour du jeu vidéo avant tout.

Ce cahier des charges a été rédigé à l'intention d'EPITA, qui pourrait à terme investir dans le développement de *Kromm Studio* et ses futurs projets. Il constitue également une feuille de route interne pour l'équipe afin de structurer nos attentes, contraintes et ressources tout au long du développement de *Treep*. Notre objectif est clair, construire un partenariat avec EPITA pour faire de *Kromm Studio* un acteur incontournable du jeu vidéo indépendant.

2 Présentation du projet

Treep a pour objectif principal de procurer une expérience de jeu captivante, adaptée à un large public. Treep est un mélange entre *"tree"*, arbre en anglais, qui est le décor principal de l'histoire et *"trip"*, voyage en anglais, qui décrit ce que le joueur ressent en jouant à notre jeu.

L'histoire de *Treep* se déroule dans un grand chêne malade, gangréné par un champignon. Le joueur incarne l'âme de la colonie d'insectes vivant au pied de l'arbre. Mais un jour, un gardien de la colonie, apparait devant l'entrée principale, tacheté de blanc, les yeux vides et les antennes baissées. Dans sa folie, il tue d'innombrables insectes avant de se faire abattre par d'autres gardiens. C'est le premier contact de la colonie avec les anomalies que provoque le parasite. Ainsi, le champignon aliène peu à peu tous les habitants de l'arbre. Le joueur doit donc atteindre le champignon, puis le détruire pour sauver la colonie et maintenir l'équilibre de la forêt.

Il faudra ainsi explorer l'intérieur du chêne et combattre les sbires pour se débarrasser de la menace champignon. De plus, à chaque mort, le joueur se réincarne en un nouveau membre de la colonie, abandonnant son ancien corps au parasite, ce qui donne une dimension punitive à la mort, car le corps, une fois parasité, se retournera contre son ancien hôte.

2.1 Origine et nature du projet

L'histoire de *Treep* a été inspirée par le livre *Les Fourmis* de Bernard Werber, mais aussi par différents documentaires sur les insectes et sur la faune et la flore. Le jeu vidéo *Ori and the Blind Forest* de Moon Studios, a aussi été une source d'inspiration pour l'histoire. Pour la partie technique, nous nous sommes inspirés à la fois de *Dead Cells*, une référence du genre Rogue-like (cf 2.2), pour la génération procédurale de contenu (cf 3.7) et de *Hollow Knight*, une référence du genre Metroidvania (cf 2.2), pour les mécaniques de mouvement et de combat.

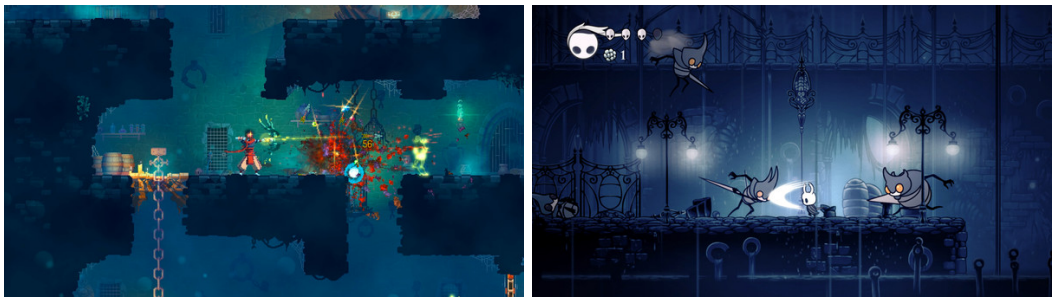

Figure 1: Images des jeux *Dead Cells* et *Hollow Knight*

2.2 État de l'art

Treep est un Rogue-like qui emprunte aussi certaines mécaniques au genre Metroidvania. Nous allons revenir sur ces genres du jeu vidéo avant d'expliquer leurs évolutions respectives.

Le terme Rogue-like désigne l'ensemble des jeux partageant les spécificités suivantes : la génération procédurale, la mort permanente et une progression dans la mort. Les trois caractéristiques précédentes induisent une grande rejouabilité puisque le contenu du jeu est infini. Le terme vient du jeu éponyme, développé par Michael Toy et Glenn Wichman en 1980. Il est inspiré par les jeux de rôle *Donjon et Dragon* qui sera précurseur dans le domaine de la génération procédurale, étant l'un des premiers à l'implémenter pour construire aléatoirement ses niveaux.

Plus récemment, *Hadès* sorti en 2020 et *Outer Wilds* sorti en 2019 sont tous deux des jeux appartenant à ce genre ayant eu un grand succès.

Figure 2: Images des jeux *Hadès* et *Outer Wilds*

Metroidvania est un genre de jeu vidéo, il vient du mot composite formé à partir du nom des jeux Metroid et Castlevania, les premiers du genre. Un jeu Metroidvania, généralement en 2D, possède une grande carte. Le joueur doit alors explorer l'ensemble du monde disponible et acquérir au fur et à mesure de nouvelles capacités (double saut, la possibilité de sauter sur les murs, propulsion avant, etc.) pour lui permettre d'accéder au combat final et de le gagner. La clé d'un Metroidvania est

donc l'exploration, obligeant le joueur à retourner sur ses pas pour accéder à des lieux de la carte auparavant inaccessibles. Les récompenses offertes à un joueur curieux sont des bonus, des nouvelles compétences ainsi que des morceaux de l'histoire. Cette dernière est intimement liée à l'expérience de jeu, puisque que les évolutions du personnage et de l'environnement sont des explications à l'histoire globale du jeu.

Plus récemment, *Hollow Knight* sorti en 2017 et *Ori and the Blind Forest* sorti en 2015 sont tous deux des jeux appartenant à ce genre ayant eu un grand succès.

Figure 3: Image du jeu *Ori and the Blind Forest*

2.3 Objet de l'étude

Ce projet vise à développer un jeu sur un an avec le moteur de jeu *Unity*[1], nous permettant ainsi de nous mettre au défi tout en découvrant de nouvelles compétences techniques. La répartition des tâches au sein du groupe est un élément clé du projet, car elle favorise la collaboration et l'apprentissage mutuel. Chaque membre peut apporter ses forces, que ce soit en programmation, en conception graphique ou en écriture narrative. D'un point de vue technique, ce projet nous permet d'approfondir nos connaissances en programmation à travers le langage *C#*[2] et de nous familiariser avec Unity.

D'un point de vue artistique, la nécessité de créer une histoire captivante pour notre jeu et pour notre entreprise nous a permis de développer nos talents d'écriture et d'imagination. De plus, la conception d'assets en deux dimensions et d'éléments musicaux a permis à chacun de développer sa créativité, d'expérimenter et de découvrir différents styles.

Enfin, innovation et expérimentation sont demandées pour trouver, de façon efficace, des solutions aux différents problèmes et défis pouvant apparaître lors de la création d'un jeu. De plus, ces défis étant techniques et créatifs, ils nous apprennent à sortir de notre zone de confort, ce qui est essentiel pour notre développement personnel et professionnel.

En somme, ce projet est une opportunité précieuse qui nous prépare individuellement aux réalités du monde professionnel, tout en nous apprenant à travailler efficacement en groupe.

2.4 L'entreprise (Kromm Studio)

Kromm Studio est un studio de jeu vidéo indépendant français fondé en 2024. Il est le produit de la collaboration entre cinq amis, tous reliés par leur passion pour le dixième art. C'est ce lien qui donne ainsi naissance à l'acronyme KROMM, pour Kristen, Romain, Oscar, Mahé et Milo. Réunis, les profils uniques qui forment le studio permettent une symbiose liant la technique à l'artistique.

Chacun des membres a eu la chance de voyager, de s'amuser et de rêver grâce à différentes œuvres du jeu vidéo. C'est pour ça que nous avons pour but de créer de véritables aventures permettant, à notre tour, de faire voyager les joueurs à travers des jeux vidéos au concept amusant, éprouvant ou encore émouvant. Car ces jeux sont faits par des passionnés pour des passionnés.

Mais pour l'heure, d'innombrables défis restent encore à surmonter. Le studio néophyte doit se faire une place parmi les grands studios de ce monde, qui ont tendance à prioriser l'argent et leurs actionnaires au détriment de l'expérience des joueurs. *Kromm Studio* a donc pour unique but de transmettre et de faire perdurer une passion affaiblie par l'avidité de certains studios et de faire renaître la véritable essence du jeu vidéo.

2.5 Membres du projet

2.5.1 Milo Moisson

Je m'appelle Milo Moisson, développeur et chef de projet. Je n'ai jamais fait de jeu vidéo, mais j'ai été emballé par l'idée de rejoindre *Kromm Studio* pour aider à superviser le développement de *Treep*, un projet intéressant tant techniquement qu'artistiquement. Mon expérience en programmation me permet de guider l'équipe dans leurs choix techniques.

Sur ce projet, j'ai un double rôle. En tant que développeur, je m'occupe principalement de l'algorithme de génération procédurale des niveaux qui est fortement liée au *level design*[3]. L'algorithme doit pouvoir être flexible pour répondre à nos besoins, robuste pour être sûr de pouvoir générer un niveau à chaque fois et suffisamment efficient pour qu'aucun joueur ne remarque son exécution. En tant que chef de projet, je dois coordonner les membres sur les diverses tâches que nous devons effectuer pour mener à terme cette mission et m'assurer de la cohérence artistique du projet.

2.5.2 Romain Dischamp

Je suis Romain Dischamp, développeur, graphiste et animateur 2D, membre de *Kromm Studio* depuis sa création. Ayant commencé à dessiner très jeune, j'ai été plongé tôt dans la création artistique. Les *art books*[4] et expositions sur le monde du jeu vidéo m'ont toujours fasciné et donné l'envie de développer à mon tour un jeu vidéo. La programmation, quant à elle, me passionne tout autant, et nourrit aujourd'hui mon besoin de création et d'apprentissage. J'ai donc choisi de rejoindre *Kromm Studio* pour réaliser un objectif personnel, mais surtout pour rejoindre et travailler avec un groupe partageant des valeurs similaires aux miennes.

Mon profil mélange ainsi artistique et technique, ce qui me permet d'être à la fois créateur et responsable des assets 2D. Pour garder une vision commune du projet, je suis aussi en charge de coordonner les différents membres entre eux sur le plan artistique. D'un point de vue technique, je m'occupe de l'implémentation des différentes mécaniques de base et j'assiste aussi d'autres membres dans la conception d'une forme d'*IA*[5].

2.5.3 Mahé De Berranger

Je m'appelle Mahé de Berranger, développeur, concepteur sonore et compositeur chez *Kromm Studio*. Je suis passionné par la musique et le développement de jeux vidéo depuis l'enfance. Pour ce projet, je m'occupe de la conception des *HUD*[6], des menus, des mécaniques de combat, ainsi que la création d'environnements sonores immersifs. Travailler sur l'aspect audio d'un jeu vidéo me permet de fusionner mes deux passions pour la musique et la technologie.

Avant d'intégrer *Kromm Studio*, j'avais déjà participé au développement de plusieurs jeux vidéos, ce qui m'a permis d'acquérir des compétences solides en sound design et en programmation. Aujourd'hui, en plus de composer la musique et de concevoir les effets sonores qui donnent vie à notre univers, j'aide à la création des assets. Je collabore par ailleurs au développement du site web du studio. Ce travail d'équipe nous permet de rester aligné sur une vision artistique et technique commune.

2.5.4 Oscar Cornut

Je suis Oscar Cornut, développeur et *game designer*[13]. Je suis passionné par le code depuis mon plus jeune âge. J'ai participé à plusieurs concours comme les trophées NSI ou la nuit du code qui m'ont permis d'acquérir une rigueur et une capacité de travail avec une échéance. Déjà expérimenté grâce la réalisation de deux jeux vidéos durant le lycée, j'ai décidé de rejoindre l'aventure *Kromm Studio*.

Mon rôle dans l'équipe est plus technique que graphique avant tout car c'est dans ce domaine que j'ai le plus de compétences en plus d'être mon secteur de prédilection. Je m'occupe de l'implémentation de l'IA des ennemis et des *PNJ*[8], mais également de construire une histoire créative et cohérente. En revanche, j'aime beaucoup raconter des histoires ou en lire, c'est pourquoi j'occupe une place importante dans le développement du *lore*[7] et des personnages.

2.5.5 Kristen Couty

Je m'appelle Kristen Couty, et je suis développeur chez *Kromm Studio*. Depuis toujours, je suis passionné par la programmation et la conception des mécanismes backend pour tout type de projet. Le développement de jeux vidéos n'est pas mon secteur de prédilection, mais j'ai rejoint *Kromm Studio* après avoir été séduit par leur vision du jeu vidéo et en quête de défis technique à relever. J'adore travailler en équipe afin de concrétiser des concepts inédits grâce à des solutions techniques novatrices.

Au sein du studio, je suis principalement assigné au développement plutôt qu'à l'aspect artistique, car mon niveau technique dépasse largement mes compétences artistiques. Cela me permet de me concentrer sur la création de mécaniques de jeu solides et performantes. Je supervise aussi le développement du système multijoueur. Cette lourde tâche inclut la gestion des interactions entre les joueurs, la synchronisation des données en temps réel et l'optimisation des performances réseau, autant d'éléments essentiels pour offrir une expérience multijoueur fluide.

3 Spécifications techniques

3.1 Mécanique de jeu et gameplay

3.1.1 Interface du jeu

Treep est un jeu 2D en vue de côté. Le HUD du jeu comporte les éléments de bases de HUD d'un jeu c'est-à-dire :

- Affichage de la vie et des capacités avec leur temps de rechargement
- Un inventaire permettant au joueur de gérer tous les objets qu'il a récupérés au cours de son aventure
- Une map interactive pour faciliter l'évolution du joueur dans le niveau et connaître la position des autres joueurs dans le mode multijoueur.

3.1.2 Déplacement

Nous avons pensé les commandes de déplacement de *Treep* pour rendre l'expérience de jeu intuitive et ergonomique. Ainsi tous les contrôles sont situés sur le clavier ce qui permet de jouer au jeu n'importe où et n'importe quand. Les contrôles de déplacement sont ceux communs à tous les jeux : Z, Q, S, D et espace. Ainsi qu'une touche supplémentaire pour intéragir avec l'environnement (E), et enfin une touche par capacité. Les contrôles sont modifiables dans le menu principal du jeu si ceux définis par défaut ne conviennent pas au joueur.

3.1.3 Structure du jeu

Le mode solo et multijoueur de *Treep* sont tous les deux constitués de trois niveaux de difficultés croissantes correspondant à trois parties de l'arbre: la souche, le tronc, et la cime. A chaque niveau le joueur doit traverser l'ensemble des salles de ce dernier afin d'arriver au combat final, un mini-boss pour les deux premiers niveaux et un boss pour le troisième.

Cependant, le joueur ne pourra pas explorer et compléter l'entièreté du niveau du premier coup, la difficulté étant devenu trop grande pour le niveau du personnage que le joueur incarne. Il devra donc retourner dans les niveaux inférieurs pour les réexplorer et développer son personnage avant de revenir à des niveaux plus complexes.

3.2 Moteur de jeu

D'un point de vue purement technique, nous développons notre jeu avec la version 6 du moteur de jeu Unity sortie le 17 octobre 2024. Cette mise à jour est particulièrement intéressante, car elle apporte de nombreuses nouveautés pour le développement du multijoueur et des jeux en deux dimensions facilitant le travail de développement et de *testing/QA*[9].

3.3 Graphisme

Pour les graphismes du jeu, nous avons choisi la deux dimensions, avec une résolution de 32x32 pixels. Ce choix permet de garder un jeu beau et original visuellement tout en limitant considérablement les ressources et le temps nécessaire à la réalisation des *assets*[10] graphiques. De

plus, de nos jours, peu de jeux adoptent ce style, ainsi, nous pensons que ce choix visuel permettra à notre jeu de se distinguer des autres.

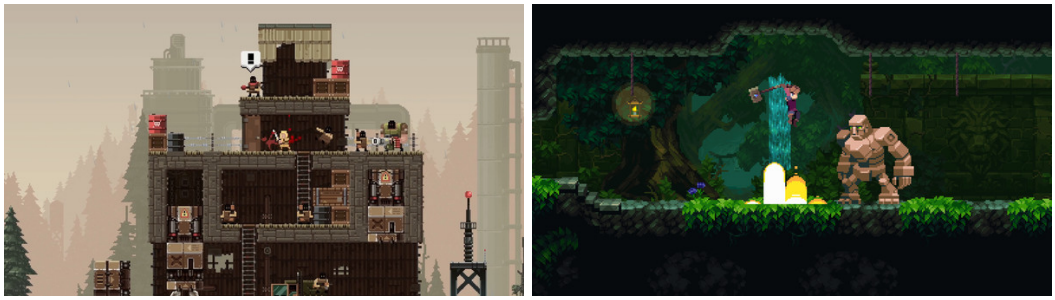

Figure 4: Images des jeux *Broforce* et *Chasm*, nos modèles pour les graphismes

Nous avons pour le moment exclu la possibilité d'avoir recours à des packs d'assets graphiques déjà conçus, cependant nous n'abandonnons pas totalement l'idée dans l'éventualité où nous rencontrerions des difficultés quant à leur conception ou des retards de développement.

3.4 Musique et son

Les moments de calme sont accompagnés par des musiques douces et apaisantes composées de mélodies acoustiques principalement à base de guitare, mais aussi d'autres instruments. Ainsi, cela renforce l'immersion du joueur en le connectant à l'ambiance dans laquelle il évolue.

Lorsque l'action s'intensifie, des morceaux épiques et dynamiques prennent le relais permettant de faire monter la tension en créant une atmosphère propice au combat, toujours dans le but d'immerger le joueur le plus possible.

D'un point de vue plus pratique, nous allons composer la plupart de nos musiques d'ambiances. Pour les effets sonores liés à l'environnement, aux actions du joueur et aux ennemis, des packs d'assets audio viendront compléter ceux que nous aurons enregistré nous-mêmes.

3.5 Multijoueur

Parmi les contraintes se trouve l'obligation d'intégrer un mode multijoueur à notre jeu. Dans ce mode, les joueurs ont la possibilité de former une équipe de deux à cinq personnes afin d'unir leur force pour affronter les menaces qui pèsent sur la colonie et tenter de la sauver. Leur bonne coopération est cruciale pour triompher des épreuves auxquelles ils vont être confrontés. En effet, chaque joueur incarne un rôle spécifique au sein de l'équipe, chaque rôle disposant d'un ensemble de capacités d'attaques, de défense et de support uniques, rendant chaque joueur indispensable au reste de l'équipe. Selon nous, l'ajout de rôles distincts permet une expérience de jeu plus immersive où la contribution de chaque membre est essentielle à la progression.

De plus, le défi est à la hauteur de la taille du groupe puisque la force et la stratégie du parasite évolue proportionnellement à la taille de l'équipe qui essaye de le détruire. Ainsi, plus l'équipe est nombreuse, plus l'ennemi est coriace, rendant les affrontements plus complexes, ce qui exige une véritable coordination entre les membres du groupe.

Pour renforcer l'aspect stratégique du multijoueur, les niveaux sont conçus de manière à répartir les joueurs à travers tout le niveau les forçant à faire preuve d'entraide et de synchronisation. Chaque membre de l'équipe se retrouve ainsi dans une zone différente de ses partenaires, mais les actions de chacun ont des répercussions directes sur les autres joueurs. Cela accentue la coopération entre les joueurs, qualité essentielle pour progresser dans les niveaux les plus complexes.

De plus, ce mode est doté de mécanismes permettant l'interaction entre les joueurs comme un système de vote ou encore de pointeur sur la carte pour leur permettre de prendre collectivement des décisions importantes comme choisir un itinéraire, indiquer des zones d'intérêt, signaler des dangers imminents, ou encore suggérer des points de rassemblement, tout cela dans le but d'encourager une participation active de chaque membre de l'équipe.

D'un point de vue plus technique, tous les joueurs du groupe se connectent à un serveur distant commun et dédié à leur partie. Ce serveur centralise et synchronise toutes les données envoyées par les clients, comme leurs positions, leurs actions ou encore leurs interactions avec l'environnement. Cette architecture permet de s'assurer que chaque joueur dispose d'une représentation à jour de la partie en cours, minimisant ainsi les désynchronisations. Cette architecture est mise en place avec la librairie *Netcode*[11] fournie par Unity permettant d'intégrer facilement du multijoueur dans son jeu.

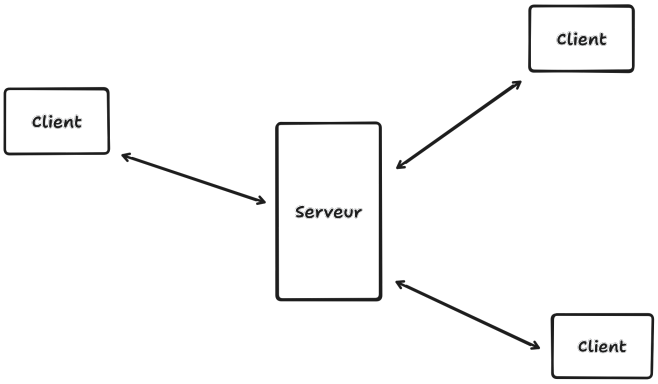

Figure 5: Schéma simplifié de l'architecture du multijoueur

3.6 Intelligence artificielle

L'IA est maintenant présente dans la plupart des nouveaux jeux, et atteint différents niveaux de complexité en fonction du problème qu'elle résout : dans *Red Dead Redemption*, un jeu de Rockstar Games, les personnages non-joueurs ont tous une histoire différente sur laquelle ils s'appuient lors de leurs interactions avec le joueur.

Par exemple, si le joueur tue un des compères du PNJ, il sera plus agressif et déterminé à tuer le joueur. Si, au contraire, un joueur aide un PNJ, il agira de façon plus clémente à son égard. Ainsi, dans cet exemple, les IA agissent en fonction des actions que le joueur est en train d'effectuer, mais aussi en fonction des actions qu'il a effectuées auparavant.

Des IA plus basiques sont aussi présentes, sans mémoire, comme dans *Pac-Man* où les fantômes chassent tous le joueur d'une manière différente, là où le fantôme rouge suit le joueur à la trace, le fantôme rose utilisera une méthode plus "réfléchie" en essayant de prédire vers où le joueur se dirige.

Conformément à la demande du client, *Treep* intégrera de l'intelligence artificielle pour le fonctionnement de ses ennemis.

- Un ennemi solitaire attaque le joueur, combat à distance si sa vie est faible, ou se cache dans un recoin pour surprendre le joueur.
- Un groupe d'ennemis attaque le joueur, fuit si la vie moyenne du groupe passe en dessous d'un certain seuil ou protège un membre affaibli.
- Les IA agissent donc en fonction des actions du joueur, du terrain sur lequel elles évoluent

et enfin en fonction de leurs propres caractéristiques telles que leurs compétences ou encore l'évolution de leur vie.

L'objectif des IA étant de dynamiser l'expérience de jeu, notre but est de créer un simulacre de réflexion en élargissant le panel des comportements des IA. Un exemple de suite d'actions faites par une IA est le suivant : *Le joueur est loin, je me rapproche. le joueur est maintenant assez proche, je l'attaque. Il m'attaque, ma vie est en dessous du seuil, je fuis.*

3.7 Génération procédurale de contenu

Pour un Rogue-like comme *Treep*, les joueurs traversent, à chaque *run*[12], les mêmes niveaux. Cela induit certains problèmes de *game design*[1]3. Tout d'abord, dans le cas de niveaux statiques, en seulement quelques parties, les joueurs sont aptes à se repérer et n'ont plus aucune surprise des dangers ou de l'emplacement de certains bonus. Cela peut rapidement lasser les joueurs. De plus, les joueurs peuvent commencer à optimiser leur itinéraire pour éviter les obstacles au lieu de s'y confronter pour augmenter ses capacités techniques et ses réflexes de combat. Ses observations sont tirées de *playtests*[14] d'autres Rogue-likes rendus publics.

Pour pallier à ce problème, nous avons choisi de faire un moteur de génération procédurale de contenu qui nous permet de générer des niveaux aléatoirement tout en gardant un certain contrôle. Pour chaque niveau, nous devons ainsi créer plusieurs pièces qui seront assemblées aléatoirement pour créer un niveau. En décrivant un niveau par un graphe, nous avons la possibilité de décider de sa forme générale. Par exemple, à quel endroit de la progression du niveau est placé un marchand de bonus.

Devoir faire plusieurs pièces pour chaque niveau demande plus de travail, mais offre un large avantage puisqu'il permet de garder une cohérence avec l'univers du jeu sur le plan graphique et le plan scénaristique. De plus, cela nous permet d'introduire manuellement des pièces qui mettent en œuvre les capacités précédemment acquises, par exemple le joueur est forcé de faire un double saut.

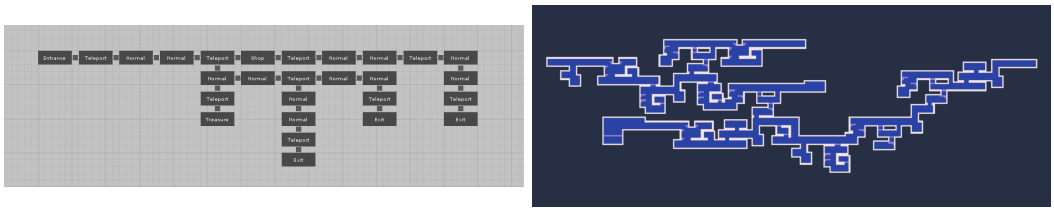

Figure 6: Le graphe de description d'un niveau et un exemple de ce niveau généré procéduralement

4 Contraintes et difficultés

4.1 Contraintes et difficultés fonctionnelles

La contrainte fonctionnelle majeure à laquelle nous sommes confrontés est organisationnelle. En effet, ne perdre aucune informations ou encore ne pas oublier une tâche peut s'avérer compliqué en ce début de projet. C'est une nouvelle expérience pour nous tous et l'organisation est pour le moment l'aspect le plus dur à gérer. Ainsi, quand nous nous regroupons, il nous arrive d'oublier de noter certaines informations ou idées énoncées, ou bien au contraire de nous perdre dans la quantité d'informations notées.

De même, se répartir correctement les tâches et obtenir un travail relativement équitable de tout le monde dans les temps fixé initialement peut s'avérer compliqué à cause de nos différences d'emploi du temps entre autres. Par exemple, la rédaction de ce cahier des charges a été pensé et débuté un peu tard par rapport à ce qu'on avait prévu, ce qui nous a mis dans une situation délicate à l'approche du rendu. Cependant, nous sommes confiants quant au fait que cela va s'améliorer avec l'expérience et la pratique.

4.2 Contraintes et difficultés techniques

A ce stade de développement du jeu, nous n'avons pas encore une idée limpide de toutes les contraintes techniques auxquelles nous allons être confrontés.

Néanmoins, nous ne possédons pas tous des ordinateurs très performants, ce qui peut parfois nous limiter dans notre avancée à cause des lenteurs d'exécutions et autres problèmes de performances. Plus généralement, cela s'apparente surtout à de l'inconfort dans certaine situation plutôt qu'à de vrais problèmes.

Similairement, nous ne sommes pas tous sur le même système d'exploitation, ce qui nous force à faire plusieurs configurations pour notre environnement de développement. De même, certains logiciels utilisés n'ont pas d'équivalent sur Linux ou inversement sur Windows, ce qui nous oblige à devoir à trouver des alternatives.

5 Conclusion

Treep, notre jeu de combat Rogue-like 2D, incarne l'ambition du studio. À travers ce projet, nous cherchons à offrir une expérience de jeu atypique tout en tenant compte de la réalité d'un projet à moyens réduits. Bien que nos fonctionnalités ne soient pas des prouesses technologiques, notre but demeure dans la création d'un univers de jeu divertissant.

Treep transporte le joueur dans un univers mystérieux dans lequel il incarne un personnage atypique, tout en proposant des défis intéressants. Avec le mode en ligne, le jeu offre aussi aux joueurs la possibilité de partir à l'aventure avec leurs amis pour venir à bout du parasite qui ronge l'arbre afin de rétablir la paix et la sérénité au sein de la colonie.

Ce projet illustre notre passion pour l'informatique et l'exploration de nouveaux horizons. Dans *Treep*, nous visons à offrir aux joueurs une expérience de jeu passionnante. Nous sommes impatients de voir notre jeu prendre forme et d'en partager le résultat. Malgré le fait que nos ressources soient limitées, ce jeu est le reflet de notre détermination, de notre désir d'attiser la curiosité des joueurs et de les inviter à s'amuser, explorer, et à s'évader dans un monde unique.

A Cahier des charges techniques — Gestion du projet

A.1 Répartition et avancement du travail

	Répartition					Avancement	
Soutenance	Mahé	Oscar	Kristen	Romain	Milo	Mars	Juin
Programmation							
Génération procédurale			S		R	90%	100%
Multijoueur			R		S	25%	100%
Intelligence artificielle		R		S		25%	100%
Mécaniques de base (mouvements, etc.)				R	S	25%	100%
Mécaniques de combat	S	R				25%	100%
Interface utilisateur (menus, HUD, etc.)	R	S				100%	100%
Game design							
Histoire/Lore		R		S		100%	100%
Conception des niveaux			S		R	25%	100%
Conception graphique	S			R		25%	100%
Conception sonore		S	R			25%	100%
Musique	R			S		25%	100%
Site web							
Conception du site web	S		R			75%	100%

Table 1: R correspond au rôle de Responsable et S correspond au rôle de Suppléant

A.2 Gestion des coûts

Besoins	Moyens utilisés	Coût
Moteur de jeu	Unity 3D	0 €
Éditeur de code	Rider (license étudiante)	0 €
Hébergement du code	GitHub	0 €
Hébergement du site	GitHub Pages	0 €
Design des assets*	Aseprite/Pixel Studio	0 €
Salaires	2500 € par mois par personne	75000€
Total		75000€

A.3 Diagramme de Gantt

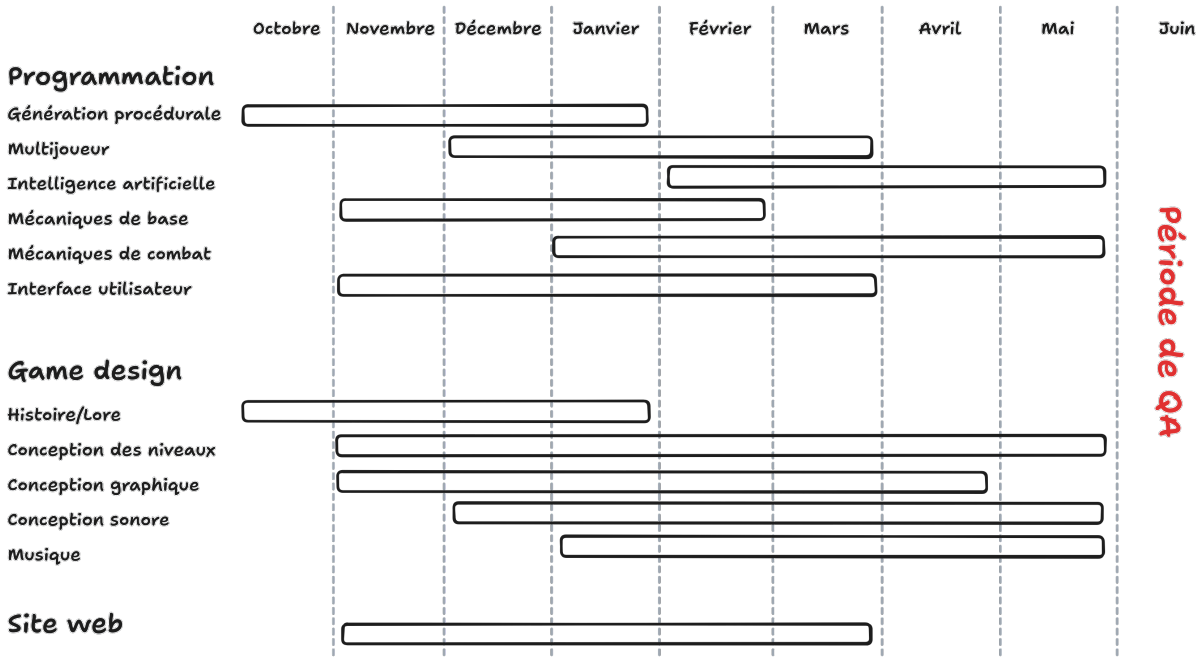

Figure 7: Diagramme de Gantt pour la répartition des tâches dans le temps

B Annexes

B.1 Termes techniques

Définitions de tous les termes techniques et anglicismes utilisés dans ce cahier des charges :

Unity[1] : Moteur de jeu largement utilisé pour créer des jeux vidéo.

C#[2] : Principal langage de programmation utilisé pour le développement de jeux vidéo sur Unity.

Level design[3] : Art de concevoir des niveaux (level en anglais), pour permettre au joueur d'avoir une expérience fluide tout au long du jeu.

Art books[4] : Livre répertoriant le processus de création d'un artiste centré autour d'un projet (jeu, personnage, film, etc.).

IA[5] : L'intelligence artificielle, dans le domaine du jeu vidéo, correspond au comportement (attaque, fuite, etc.) des personnages non-joueurs (ennemis, marchands, etc.) présents dans le jeu.

HUD[6] : Le Heads-Up Display, littéralement affichage tête haute, désigne tous les éléments affichés en permanence sur l'écran d'un joueur (vie, endurance, munitions, etc.).

Lore[7] : Correspond au cadre narratif établi d'un jeu. Il influe sur toute la partie artistique du jeu : son histoire, ses personnages, son environnement, ses musiques, etc.

PNJ[8] : Un personnage non-joueur est un personnage entièrement contrôlé par une IA. Il peut avoir plusieurs rôles comme celui d'un ennemi, de vendre des bonus, de donner des informations, etc.

Testing/QA[9] : Étape dans la création d'un jeu vidéo où le jeu est testé, poussé à ses limites, pour faire ressortir d'éventuels bugs, problèmes lors de la progression ou incohérence narrative.

Assets[10] : Ensemble des éléments graphiques du jeu vidéo, regroupant sprites 2D, modèles 3D, animations, textures, musiques et effets sonores.

Netcode[11] : Ensemble de protocoles permettant la synchronisation de différents composants tels que les joueurs et leurs actions au sein d'un jeu en ligne.

Run[12] : Désigne la partie d'un joueur sur un jeu, du lancement de la partie, à la mort ou à la victoire du joueur. Ce terme est beaucoup utilisé dans les jeux du genre Rogue-like ou le joueur a tendance à relancer beaucoup de parties.

Game design[13] : processus de création et de mise au point des règles et autres éléments constitutifs d'un jeu.

Playtest[14] : Phases de test faites par une petite partie de joueurs désigné par le studio, ayant pour but de tester le jeu, de trouver les failles et les incohérences au niveau du gameplay et de l'histoire.

B.2 Sources

Liens des images utilisées :

- *Hollow Knight* : `https://store.steampowered.com/app/367520/Hollow_Knight/`
- *Dead Cells* : `https://store.steampowered.com/app/588650/Dead_Cells/`
- *Chasm* : `https://store.steampowered.com/app/312200/Chasm/`

- *Broforce* : `https://store.steampowered.com/app/274190/Broforce/`
- *Ori and the Blind Forest* : `https://store.steampowered.com/app/261570/Ori_and_the_Blind_Forest/`
- *Hades* : `https://store.steampowered.com/app/1145360/Hades/`
- *Outer Wilds* : `https://store.steampowered.com/app/753640/Outer_Wilds/`