Rapport de Soutenance 2

Treep

Kromm Studio

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

D1

Mars 2025

Tables des matières

1 Introduction

Ce second rapport a pour mission de définir et de présenter en détail l'avancement de Treep, les défis rencontrés, ainsi que les étapes restantes nécessaires à sa concrétisation.

Le nom du jeu vient de la fusion des mots *tree* (pour arbre) et *trip* (pour voyage), qui sont les deux axes majeurs de notre jeu.

En effet, dans Treep le joueur incarne l'âme d'une gigantesque colonie d'insectes vivant au pied d'un arbre millénaire. 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, le joueur va être confronté 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.

Treep est donc un Rogue-like[1] multijoueur qui utilise un algorithme de génération procédurale pour rendre chaque partie unique.

2 Bilan des réalisations

Dans l'ensemble, nous avons bien suivi les impératifs et le calendrier que nous nous étions fixées en début d'années. Cependant, un peu de retard a été accumulé, mais cela majoritairement dans des domaines qui relèvent plus d'un travail de finition.

2.1 Mécaniques de mouvement

Dans la conception de jeux vidéo, l'objectif est d'offrir aux joueurs une expérience fluide et captivante. Le mouvement en est responsable en partie. Dans le cadre de l'amélioration continue du jeu, plusieurs nouvelles mécaniques ont été ajoutées afin d'offrir une expérience encore plus fluide et immersive aux joueurs.

Tout d'abord, un système de position accroupie a été implémenté. Ce système prend en compte l'environnement du joueur, lui permettant de se baisser uniquement lorsque l'espace le permet. Cette fonctionnalité améliore l'aspect tactique du jeu en ajoutant une dimension supplémentaire aux déplacements du personnage et la possibilité d'explorer des endroits cachés et difficiles d'accès.

Ensuite, un système de dash[2] a été intégré dans notre jeu. Le dash[2] ne peut être utilisé qu'une seule fois après que le joueur ait touché le sol, garantissant un équilibre dans le gameplay. De plus, la distance du dash[2] s'adapte en fonction de l'environnement afin d'éviter toute collision imprévue et d'assurer une transition fluide entre les déplacements.

La capacité du joueur à grimper les échelles et les lianes a également bénéficié d'une amélioration avec l'ajout d'une animation dédiée. Cette animation rend la mécanique plus visuelle et immersive.

Enfin, une synchronisation des animations avec les inputs[3] du joueur a été mise en place. Cela garantit que chaque action du joueur est représentée visuellement en temps réel, rendant le gameplay plus réactif et satisfaisant pour le joueur.

Les collisions des éléments de la carte ont été revues pour permettre aux joueurs de mieux interagir avec son environnement et éviter des collisions que Unity a du mal à gérer.

2.2 Mécaniques de combat

La mise en place des mécaniques de combat a représenté un défi important, nécessitant la création d'ennemis avant même de pouvoir intégrer le système de combat lui-même.

Sans adversaires à affronter, il était impossible de tester et d'ajuster correctement les différentes interactions et dynamiques de jeu.

Originalement le joueur pouvait seulement taper devant lui en fonction de son vecteur vitesse, mais le combat étant central au jeu, nous ne pouvons pas nous contenter d'un simple coup vers l'avant, c'est pourquoi nous avons décidé d'implémenter plusieurs types d'attaques, permettant au joueur de frapper devant lui, au-dessus et en dessous.

Une part importante du travail a consisté à se documenter en étudiant d'autres rogue-lites, s'inspirant de leur manière d'aborder les combats pour les rendre dynamiques, réactifs et stratégiques, tout en conservant une complexité dans le gameplay. L'objectif était de créer un système engageant, où chaque coup compte et où le joueur doit adapter ses attaques à la situation.

Le combat n'est pas simplement une mécanique dans notre jeu, il en est l'âme, un élément fondamental qui façonne l'expérience et le plaisir du joueur.

2.3 Génération procédurale

Nous avons traduit le code du PoC de génération procédurale que nous avions fait en Rust en C-Sharp pour pouvoir l'intégrer au jeu. Nous avons aussi programmé une interface Unity basique pour pouvoir régénérer facilement des niveaux en cliquant sur un bouton et en choisissant le nombre de salle et la *seed*[14].

2.4 IA

Le début de la réalisation de l'IA s'est fait dans un premier temps lors de la réalisation de niveaux. Des coordonnés pour l'emplacement des ennemis ont été renseignées dans chaque niveau. Les ennemis devaient se trouver à un endroit logique, ne bloquant pas le joueur dans le parcours de la salle (ni à l'entré, ni à la sorti). De plus, des zones "optionnelles" dans les salles ont été réalisées, avec plus d'ennemis, pour permettre au joueur de combattre plus d'ennemis si il le souhaite.

Les ennemis ont pour l'instant la capacité de marcher de droite à gauche, changeant de sens le mur atteint. Enfin, les ennemis ont une barre de vie au-dessus de leur tête, mais cette partie a été réalisée pour régler les différents problèmes des mécaniques de combat.

2.5 HUD/UI

La réalisation de l'HUD/UI[4] est composée de deux éléments : les interfaces graphiques et un système de barre de vie pour les ennemis.

Les interfaces graphiques ont été réalisées dans l'objectif de rendre le lancement du jeu plus lisible, et plus simple à effectuer, mais ces interfaces ne sont pas définitives, et dans le futur, il y aura surement de nouvelles interfaces graphiques, plus en accord avec notre projet.

Pour la barre de vie, nous avons implémenté une barre de vie temporaire, se trouvant au-dessus de la tête des ennemis. Nous ne sommes pas certains de garder une barre de vie visuelle pour les ennemis et nous réfléchissons à des solutions plus originales pour indiquer au joueur la vie restante des ennemis. Cela pourrait passer par des blessures sur les ennemis ou des vêtements déchirés. Mais, pour l'instant, pour faire nos tests avec les ennemis et avec l'équilibrage de notre jeu, nous allons garder cette barre de vie visible pour nous faciliter le travail.

Nous avons aussi ajouté un HUD[4] permettant au joueur de se mettre prêt quand ils sont dans le lobby pour pouvoir lancer la partie. Cet HUD[4] aussi est pour l'instant très basique et peu esthétique, mais nous permet d'avancer dans le développement.

De manière générale, nous retravaillerons les visuels de toutes les interfaces utilisateurs pour la sortie finale du jeu.

2.6 Génération procédurale

Pour rappel, notre algorithme de génération procédurale est le fondement de la création des niveaux dans notre jeu. Son fonctionnement est simple, il essaye d'associer les salles que nous avons créées selon plusieurs critères jusqu'à ce qu'il parvienne à un enchainement valide.

Lors de la première soutenance, nous avions pu présenter un prototype de génération procédurale des niveaux dissocié du jeu. Ce prototype à été complètement intégré dans le jeu actuel et permet de simplifier la création des morceaux de niveaux.

Chaque niveau est décrit par une suite de type de salles ainsi que les morceaux de salles préfabriqués qui sont utilisés pour composer le niveau. À l'aide d'un simple bouton, il est possible de tester différentes combinaisons de niveaux.

2.7 Multijoueur

Par rapport à la dernière soutenance, nous avons fait évoluer le multijoueur, cependant les changements apportés sont majoritairement invisibles pour l'utilisateur. Nous avons retravaillé le script[5] du multijoueur pour le rendre plus approprié à notre jeu et nous sommes détachés des scripts[5] fournis par la librairie Mirror que nous utilisons.

Nous avons entre autre fait notre propre script[5] pour faire apparaitre et créer des joueurs afin de pouvoir leur assigner un pseudo ou encore une couleur.

Nous avons aussi commencé à lier la génération procédurale et la gestion des parties avec le multijoueur, mais nous avons encore du travail sur ces points et plusieurs problèmes à résoudre.

Nous avons implémenté une *state machine*[15] pour la partie et pour les joueurs. La partie est soit dans l'état *lobby*[6], *jeu* ou *fin*. Un joueur peut être prêt ou mort et inversement. Selon l'état dans lequel la partie se trouve, différents scripts sont exécutés. Par exemple, lors du passage de l'état *lobby*[6] à *jeu*, la génération de la map est déclenché.

Parmi les changements visibles par l'utilisateur, nous avons entre autre ajouter un HUD[4] pour lancer la partie, et synchroniser la direction du regard, des animations et plus en général des visuels entre tous les joueurs. Lorsque le joueur clique sur le bouton, un message est envoyé au serveur qui enregistre le joueur comme prêt. Une fois tous les joueurs prêts, la partie et se lance. Ce HUD[4] est disponible uniquement lorsque la partie est dans l'état lobby[6].

2.8 Site web

Le site internet du jeu est entièrement fini. Par rapport à la dernière fois, nous avons uniquement changé des éléments de design comme les couleurs, la présentation ou encore quelques petits effets visuels. Le principal changement parmi cette liste a été de refaire le diagramme de Gantt complètement en HTML et CSS plutôt que d'afficher une image d'un diagramme fait avec un logiciel tiers sur le site.

2.9 Conception des assets et des niveaux

Suite à la création d'un tileset[7] fixée pour la zone 1 du jeu, les membres responsables de la création de niveaux, ont réalisé environ 10 salles pour alimenter l'algorithme de génération procédural. (Ces salles ne sont pas présentes dans le .exe, car elles dépendent d'une branche[8] pas encore fix[9]) Pour créer ces salles, il a fallu respecter certains points :

Le joueur doit pouvoir parcourir la salle sans difficultés, en effet la course étant rapide et les sauts assez hauts, il ne faut pas qu'ils soient entravés par des plafonds trop bas ou des marches mal disposées par exemple. Mais malgré cette contrainte, les niveaux devaient rester organiques, pour rappeler au joueur qu'il parcourt un arbre.

De plus, les salles doivent être toutes différentes dans leur forme et dans leur parcourt pour que le jeu ne soit pas monotone, donnant une impression de découverte au joueur à chaque partie lancée.

Enfin, les salles ne doivent pas bloquer le joueur dans sa progression. En effet, les salles ayant des nombres d'ennemis prédéfinis, le joueur ne doit pas se retrouver bloqué, sous équipé, par rapport aux ennemis qu'il a à vaincre. De plus, à la fin de chaque zone, il y a un boss, un joueur ayant récupéré tout le stuff disposé tout au long de son aventure, doit pouvoir vaincre le boss sans trop de difficulté.

Pour les animations, les membres responsables ont implémentés de nouvelles animations pour le joueur (grimpe, petite attaque) et pour ses variantes colorés représentant les différents joueurs du multijoueur, mais aussi des animations rudimentaires pour les ennemis (coup prit, idle[10]). Le design des ennemis a aussi été longuement réfléchi car un ennemi doit paraître menaçant, et doit rester dans le style insecte humanoïde. Ainsi, des fourmis contrôlées par le champignon ainsi qu'une grande termite ont été réalisées. (À noter que la termite de bois est pour l'instant présente dans la zone 1 correspondant à l'écorce de l'arbre mais sera par la suite présente en grande partie dans la zone 2 correspondant à la ville des termites)

Enfin, suite à la réalisation d'une partie de l'HUD/UI[4], une barre de vie et un menu ont été réalisés.

3 Retards et défis rencontrés

3.1 Mécaniques de mouvement

Lors du développement, un bug permettait au personnage de convertir la vitesse de son saut en propulsion vers le haut. Cela était dû à la génération automatique des hitboxes[11] par Unity, qui ne correspondait pas toujours précisément à la forme des cubes composant la carte.

Au lieu d'utiliser les hitboxes[11] générées automatiquement, une hitbox[11] a été définie manuellement. Cela a permis de mieux contrôler les interactions du personnage avec l'environnement et d'éviter tout comportement anormal, comme le fait d'être projeté dans les airs.

Lors de l'utilisation du dash[1], le joueur pouvait parfois traverser les murs, ce qui posait un problème majeur. Ce problème survenait principalement parce que la vitesse élevée du dash[1] permettait au joueur de passer à travers les collisions automatiques d'Unity.

Pour éviter cela, un lancer de rayon a été ajouté avant l'exécution du dash[1] qui détecte les obstacles potentiels dans la direction du dash[1] et ajuste la vitesse ou stoppe le mouvement si un mur est détecté. Cela empêche le joueur de traverser les murs tout en conservant une fluidité dans le gameplay.

Initialement, le système d'accroupissement ne prenait pas en compte l'environnement au-dessus du joueur. Cela signifiait que le personnage pouvait occasionnellement se relever alors qu'un obstacle (comme un plafond bas) se trouvait juste au-dessus, entraînant des problèmes de collision et un possible blocage du joueur dans les décors.

Une vérification a été ajoutée pour s'assurer que le joueur ne puisse se relever que si l'espace au-dessus de lui est suffisant. Cette vérification détecte si un plafond empêche le redressement. Cela garantit un comportement plus réaliste et évite les problèmes de collision.

3.2 Mécaniques de combat

De nombreuses embûches se sont dressées pendant la création des mécaniques de combats, étant donné que ce n'est pas la même personne qui a construit les mécaniques de mouvements et celle de combat, il y a eu un travail de compréhension de code important, cela a induit une communication entre les membres de l'entreprise.

Suite à ça, nous avons pu nous concentrer sur les nouveautés à implémenter. Nous avons donc d'abord implémenté le plus basique des ennemis pour servir de test. Ensuite, nous avons pu créer des attaques sur les côtés. Nous nous sommes alors confrontés à un problème, celui d'empêcher que le joueur puisse frapper très rapidement, nous avons donc rajouté une caractéristique à l'arme : le nombre de coups par secondes autorisé. Cette caractéristique étant un attribut de la classe "weapon", cet attribut pourra être changé en fonction de l'arme.

Enfin, à part quelques bugs mineurs, surtout autour de l'implémentation des animations, le développement de cette partie du jeu s'est déroulé sans réel problème. Nous avons réussi à bien gérer notre calendrier pour ne pas prendre de retard sur cette partie.

3.3 Multijoueur

Nous avons eu des difficultés pour passer du jeu qui marchait localement à des clients synchronisés correctement. Tout d'abord, nous avons changé d'outil pour synchroniser les différents clients Unity. Nous avons dû gérer à la fois le mode solo qui fonctionne avec un serveur local et le mode multijoueur que nous avons choisi d'architecturer autour d'un serveur distant.

C'est une *state machine*[15] qui s'occupe de gérer l'état d'une partie, faire apparaitre les niveaux, déclencher la génération procédurale avec la même graine pour tous les clients et téléporter les joueurs. Apprendre à utiliser les différents composants de Mirror, notre librairie de réseau, pour s'envoyer de telles informations a été un des principaux problèmes. En effet, si la librairie semble simple à prendre en main, dès lors qu'il faut synchroniser des états plus complexes, il faut bien comprendre tous les outils à notre disposition. De plus, la mécanique de client "hébergeur" qui a à la fois un statut de client et de serveur rajoute de la complexité à gérer.

3.4 Conception des assets et des niveaux

La réalisation de salle est plus longue que prévu, car il faut avoir une réelle réflexion dans la réalisation de chacune d'entre elles, réflexion renseignée dans la description de l'avancement de cette partie.

De plus, la taille moyenne des salles étant de 80 par 40 pixels, et chaque tile étant de 16 par 16 pixels, un nombre important de tiles sont à placer et à réfléchir pour créer un niveau original et optimal.

Enfin, une partie des premières salles réalisée sont devenues obsolètes, car ne répondant pas aux critères que nous nous sommes fixés, obligeant leur suppression du projet.

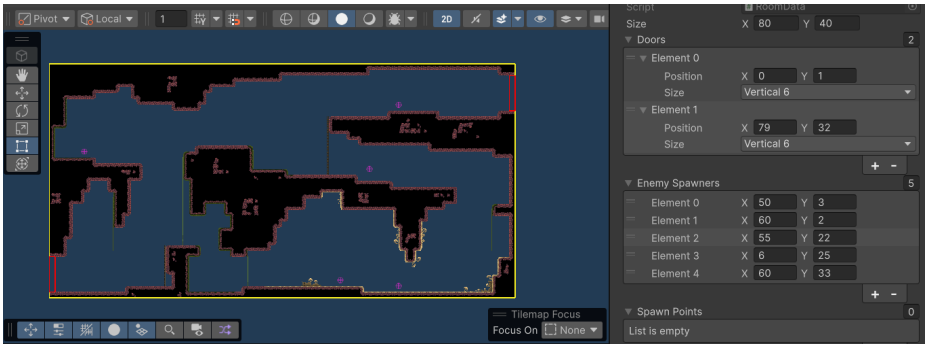

Figure 1: Création d'un niveau

4 Planification à venir

4.1 Interface Utilisateur

Dans n'importe quel jeu, il est possible de changer ses touches, de régler la luminosité ou encore de régler le son. C'est pour cela que nous souhaiterions implémenter un menu "paramètres" à notre jeu.

Pour les touches, nous aimerions les mettre en ZQSD si la langue sélectionnée est francophone, WASD dans les autres cas. De plus, les touches d'interactions, d'accroupissement et autre seront aussi modifiables.

Pour le son, il y aura deux jauges que le joueur pourra modifier : la musique et les effets sonores. De plus, le son pourra passer de mono à stéréo pour garantir au joueur une expérience plus immersive. (Cette future modification est une suggestion, et ne sera peut-être pas implémentée)

Enfin, pour la luminosité, une seule jauge pourra être modifiée, avec une image bicolore grise sur du noir, pour que le joueur puisse effectuer ses changements en fonction de l'image. Il sera indiqué que cette image devra être à peine visible, pour que les parties du joueur gardent un aspect sombre (comme dans le jeu *Lethal Company*)

À noter que toutes les modifications effectuées dans les paramètres ne seront propres qu'au joueur les ayant effectuées, et les autres joueurs d'une partie multijoueur auront le droit à leurs propres modifications.

Enfin, l'implémentation d'une barre de vie visuelle et d'un inventaire pour le joueur sont envisagé, mais rien n'est décidé pour l'instant.

4.2 Mécaniques de mouvement

Dans le cadre de l'évolution des mécaniques de mouvement, nous avons prévu d'octroyer aux joueurs des capacités uniques. Il faudra implémenter en jeu ces mécaniques spécifiques, telles que la possibilité de planer, le double saut et d'autres compétences à définir. Chaque compétence nécessitera des animations dédiées, qui devront être créées et parfaitement synchronisées avec les actions du joueur pour garantir une immersion optimale.

Étant donné que le jeu propose une composante multijoueur, il sera essentiel d'attribuer à chaque joueur les compétences qui lui sont propres tout en assurant une distinction visuelle claire. Pour cela, un système permettant de différencier les joueurs par des variations de couleur dans les vêtements devra être mis en place afin de rendre leur identification plus simple.

4.3 Mécaniques de combat

Nous prévoyons d'ajouter des animations de frappes verticales en complément de celles horizontales, enrichissant ainsi le système de combat.

De plus, plusieurs armes variées feront leur apparition, chacune aura le droit à ses propres dégâts, hitboxes[11] et mécaniques uniques. Par exemple, un arc offrira des attaques à distance, une masse infligera de lourds dégâts, mais ralentira la fourmi. Ces ajouts apporteront ainsi plus de diversité et de stratégie aux combats.

4.4 Multijoueur

Pour cet aspect du jeu, il nous reste principalement des problèmes à corriger et des affinements à faire plutôt que des fonctionnalités à ajouter. Nous devons résoudre les problèmes liés à la génération synchronisée de la map sur tous les clients et ceux liés à notre système d'état pour gérer la partie. Enfin, nous devons aussi continuer de travailler sur la bonne intégration de toutes les mécaniques de notre jeu avec le mode multijoueur.

4.5 IA

Les futures réalisations pour l'IA seront composées d'un algorithme de path finding[12] pour que les ennemis se dirigent vers le joueur à partir d'une certaine distance. De plus, les ennemis pourront attaquer, battre en retraite, etc. avec un comportement différent par type d'ennemi (les fourmis au stade 1 du champignon seront plus craintives que le stade 5, et que les termites soldates).

4.6 Conception des assets et des niveaux

Pour rendre notre jeu moins monotone, nous allons implémenter de nouvelles zones, ces nouvelles zones seront donc accompagnées de nouveaux tilesets[7] ainsi que de nouveaux ennemis. De plus, la réalisation de nouveaux tilesets[7] entraine la création de nouveaux niveaux, raccord avec la logique de la zone (si la salle est faite pour la zone 2, autrement dit la ville des termites, les salles seront moins organiques, plus carrées, pour donner un effet de ville taillée dans le bois).

De plus, la création de nouveaux ennemis ajoutera la nécessité de faire, pour chaque PNJ[13], un set d'animations personnalisé.

5 Récapitulatif de l'avancement et de la répartition

	Répartition					Avancement		
Soutenance	Mahé	Oscar	Kristen	Romain	Milo	Janvier	Mars	Mai
Programmation								
Génération procédurale			S		R	90%	100%	100%
Réseau multijoueur			R		S	50%	70% (-30%)	100%
Intelligence artificielle		R		S		0%	30% (-20%)	100%
Mécaniques de base (mouvements, etc.)				R	S	50%	75%	100%
Mécaniques de combat	S	R				10% (-15%)	30% (-20%)	100%
Interface utilisateur (menus, HUD, etc.)	R	S				0% (-50%)	50% (-50%)	100%
Game design								
Histoire/Lore		R		S		90%	100%	100%
Conception des niveaux			S		R	25% (-15%)	50% (-25%)	100%
Conception graphique	S			R		45% (+20%)	75%	100%
Conception sonore		S	R			0% (-10%)	0% (-50%)	100%
Musique	R			S		10%	50%	100%
Site web								
Conception du site web	S		R			90%	100%	100%

Table 1: R correspond au rôle de Responsable et S correspond au rôle de Suppléant. Les pourcentages entre parenthèses au delta entre l'avancement actuel et l'avancement initial

6 Conclusion

Jusqu'à maintenant, nous avons fini le développement du site internet, de la génération procédurale et de la plupart des mécaniques de mouvement.

Nous avons également grandement avancé dans la conception des assets avec le modèle des personnages, mais aussi des tiles pour le design des niveaux. Dans le même domaine, nous avons aussi bien travaillé les animations et la conception des niveaux. Il nous reste cependant encore à faire preuve d'imagination pour créer toutes les salles nécessaires au bon fonctionnement de notre algorithme de génération procédurale.

Le multijoueur et la gestion des joueurs et de la partie est le point sur lequel nous devons encore beaucoup travailler afin de pouvoir proposer un système entièrement fonctionnel pour le rendu finale.

Il reste encore beaucoup à faire, notamment sur l'intelligence artificielle, l'ambiance sonore et les mécaniques de combat que nous n'avons pas profondément développés.

7 Annexes

7.1 Définitions

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

Rogue Like[1] : Sous genre du jeu vidéo, inspiré du jeu Rogue.

Dash[1] : Action très présente dans les jeux vidéos, consistant à faire bondir le joueur vers l'avant.

Input[3] : Ensembles des entrées claviers et/ou souris du joueur, guidant à une action.

HUD/UI[4] : L'HUD/UI pour Head-Up Display/User Interface, désigne toutes les parties visuelles qui ne font pas partie directement du jeu : inventaire, menu, barre de vie ...

Script[5] : Les scripts désignent toutes les parties "techniques" permettant au jeu de fonctionner. Ces scripts sont écrit, dans notre cas, majoritairement en Csharp.

Lobby[6] : Dans les jeux vidéos, les lobby agissent comme une zone pour permettre aux joueurs d'attendre le début de la partie.

Tileset[7] : Image, répertoriant toutes les tiles utilisables dans le jeu.

Branche[8] : Une branche est une partie d'un dépôt Github, permettant aux développers d'un projet de ne pas tous travailler au même endroit, permettant une avancée sur le projet plus fluide.

Fix[9] : "Réparer" en anglais, cela désigne l'action de régler un problème ou un bug.

Idle[10] : Les animations "Idle" désignent celle qui sont appelées quand le personnage reste statique sans rien faire. Elles sont réalisées entre autre pour montrer au joueur que le jeu n'a pas crash.

Hitboxes[11] : Zone invisible permettant de définir le contour de collision d'un personnage ou d'une tile de la tilemap.

Path Finding[12] : Algorithme permettant de trouver le chemin le plus proche dans un environement précis.

PNJ[13] : Personnage Non Joueur, désigne tous les personnages qui ne sont pas contrôlés par un joueur.

Seed[14] : Suite de chiffres choisis de façon aléatoire, pour, dans notre cas, générer un ensemble de rooms de façon aléatoire.

State Machine[15] Système responsable de la gestion de l'état de la partie:

7.2 Références bibliographiques

- *Site internet de Treep* : `https://treep.world/`
- *Rust* : `https://www.rust-lang.org/`
- *Mirror Networking* : `https://mirror-networking.com/`