Rapport de soutenance n°3

Uzin

Bart Entreprises
Mai 2025

Tristan DRUART (*Chef d'équipe*)
Martin LEMEE
Cyril DEJOUHANET
Maxan FOURNIER

Table des matières

1 Introduction

1.1 Présentation de l'équipe

L'équipe *Bart Entreprises* est composée de quatre étudiants de première année à l'EPITA, liés non seulement par leur passion pour la technologie, mais aussi par une amitié solide. Leur objectif commun : concevoir des projets innovants qui marient créativité et technique. Chacun apporte ses compétences et sa personnalité unique, formant un groupe complémentaire prêt à relever tous les défis.

Tristan DRUART (Chef du projet)
Je suis étudiant en première année à l'EPITA, passionné par l'informatique depuis le lycée, où j'avais choisi les spécialités Mathématiques et Sciences de l'ingénieur. J'ai commencé à coder par curiosité et me suis rapidement intéressé à des langages comme Python. Aujourd'hui, je poursuis mes études en explorant des domaines tels que l'intelligence artificielle et la cybersécurité. J'ai la fierté d'être aujourd'hui chef du projet *Uzin*.

Martin LEMEE
Passionné par l'informatique depuis plus de dix ans, j'ai toujours été intrigué par les possibilités créatives qu'offre ce domaine. Cette fascination m'a naturellement orienté vers des études en ingénierie informatique, ce qui m'a conduit à intégrer EPITA. En explorant divers aspects du numérique, notamment le développement, j'ai acquis des compétences précieuses que je mets aujourd'hui au service de l'équipe dans la création de ce jeu vidéo.

Cyril DEJOUHANET
Ayant toujours été intéressé par l'informatique et les ordinateurs, j'ai décidé au lycée de suivre la spécialité Numérique et Sciences Informatiques. Pris de passion, l'étape suivante a été de trouver une école supérieure, qui partageait cette passion et avec laquelle j'étais prêt à m'engager pour mes études. C'est ainsi que j'ai intégré l'EPITA en première année, et que je suis fier de participer dans un premier vrai projet de programmation.

Maxan FOURNIER
Étudiant en première année du cycle préparatoire à l'EPITA, le choix de mes études fut une évidence au vu de ma passion pour l'informatique développée dès le plus jeune âge. Principalement expérimenté dans les technologies du web et ayant suivi la spécialité Numériques et Sciences de l'Informatique au lycée, je compte employer ces compétences au service de notre projet.

2 Présentation du projet

Uzin est un jeu de gestion et d'exploration multijoueur dans lequel le joueur est plongé dans un environnement inconnu, riche en ressources exploitables. Il commence l'aventure seul, avec pour objectif de récolter manuellement les premières ressources disponibles sur la carte. Progressivement, il débloque des mécanismes lui permettant d'automatiser cette extraction, notamment grâce à la construction de foreuses, ouvrant la voie à une industrialisation croissante de son environnement.

Le cœur du gameplay repose sur une boucle de progression basée sur l'exploration, l'optimisation des extractions et la montée en complexité des ressources. Le joueur débute avec des matières premières simples, comme le fer, puis découvre de nouveaux types de minerais nécessitant une organisation plus poussée de sa chaîne de production. Cette complexification pousse le joueur à réfléchir à la disposition de ses machines et à améliorer constamment l'efficacité de ses installations.

Cependant, cette exploitation intensive de l'environnement n'est pas sans conséquences : plus le joueur étend son activité, plus il déclenche des réactions hostiles de la part de l'écosystème. Des créatures volantes sont alors générées dynamiquement et viennent perturber la progression du joueur. Elles attaquent ses installations ou tentent de le blesser directement, forçant ce dernier à réagir et à se défendre activement. Pour cela, il peut utiliser des armes, renforcer sa mobilité, ou adapter la disposition de ses structures afin de limiter les pertes.

Le jeu intègre également une dimension coopérative : plusieurs joueurs peuvent rejoindre une même session, explorer ensemble, partager des ressources et coordonner leur production. Cette dimension multijoueur ajoute une couche stratégique, notamment lorsqu'il s'agit de répartir les rôles, d'optimiser les zones d'extraction ou de repousser les assauts ennemis en équipe.

En somme, *Uzin* se veut être une expérience hybride entre exploration, extraction, survie et coopération. L'accent est mis sur la fluidité des systèmes, la clarté des interactions, et la montée en puissance progressive du joueur. À terme, le projet ambitionne d'offrir une expérience complète mêlant gestion, automatisation, défense et découverte dans un univers vivant et évolutif.

3 Origine et nature du projet

3.1 Origine du projet

Ces dernières années, notre équipe a eu l'opportunité d'explorer et d'expérimenter une grande diversité de jeux vidéo, aux styles, aux genres et aux objectifs toujours plus variés les uns que les autres. Parmi toutes ces œuvres que nous avons pu tester, un style bien particulier s'est distingué, à la fois par son originalité et par sa capacité à provoquer chez le joueur des sensations et émotions intenses. Ce type de jeu, qui mélange gestion, optimisation, casse-tête et exploration, nous a conduit à opter pour un jeu de création d'usine comme la meilleure manière d'aborder notre projet. Grâce à notre solide connaissance dans ce domaine et à notre expérience accumulée au fil du temps, nous sommes en mesure de proposer aux joueurs une expérience unique et inédite.

C'est dans cette dynamique que le projet *Uzin* est né, marquant par la même occasion la création de notre entreprise.

3.2 Nature du projet

Dans *Uzin*, le joueur aura la possibilité d'explorer, seul ou à plusieurs, un monde inconnu et hostile. Il devra y installer divers "Uzin" (assemblages de machines et de mécaniques ingénieuses mises à disposition par notre projet) pour automatiser différentes fonctionnalités cruciales du jeu (cf. *6.1 - Tâches à effectuer* et *6.2 Répartition des tâches*), tout en s'efforçant de se défendre contre les menaces constantes que représente cet environnement hostile pour son personnage.

Le projet *Uzin* tire sa richesse et ses caractéristiques de plusieurs genres de jeux, chacun apportant des fonctionnalités aussi spécifiques qu'intéressantes. On retrouve notamment des éléments de jeu de survie, où le protagoniste sera confronté à diverses menaces inhérentes à ce monde étrange. Le joueur devra dès le début mettre en place des stratégies simples pour y faire face, mais celles-ci devront évoluer rapidement à mesure que les dangers, provenant de la faune et de la flore locale, deviendront de plus en plus pressants. Enfin, l'aspect gestion et casse-tête du jeu se traduira par la nécessité pour le joueur de faire preuve de réflexion dans l'utilisation des ressources limitées que cette planète offre.

4 Vision initiale du projet (Novembre 2024)

4.1 Tâches à effectuer

4.1.1 Intelligence artificielle

L'intelligence artificielle (IA) dans *Uzin* jouera un rôle crucial en rendant le monde du jeu évolutif et stimulant. Elle sera présente à différents niveaux, notamment dans la gestion des créatures ennemies, qui ajusteront leurs attaques en fonction des actions du joueur. Plus le joueur développera ses infrastructures, plus ces créatures deviendront agressives, créant ainsi un défi constant. L'IA sera aussi responsable de la gestion des ressources et des chaînes de production automatisées, aidant le joueur à les optimiser en fonction des besoins et des ressources disponibles, afin d'assurer une progression fluide dans le jeu.

De plus, l'IA permettra à l'environnement, notamment la flore, de réagir intelligemment aux activités du joueur. Par exemple, certaines zones pourront être bloquées ou devenir dangereuses en raison de la libération de toxines, obligeant le joueur à ajuster ses plans. En mode multijoueur, l'IA jouera également un rôle d'ajustement, modulant la difficulté en fonction du nombre de participants pour offrir une expérience équilibrée à chacun. En somme, l'IA rendra le monde d'*Uzin* imprévisible, forçant les joueurs à constamment revoir leurs stratégies et à rester attentifs aux réactions de leur environnement.

4.1.2 Histoire et storytelling

Dans *Uzin*, développer l'aspect narratif est essentiel pour donner une profondeur au jeu et captiver le joueur. Une histoire bien construite donne un sens aux actions du joueur, allant au-delà de la simple collecte de ressources ou de l'automatisation des processus. Ainsi, nous voulons donner une explication concrète aux défis auxquels le joueur est confronté, qu'il s'agisse des attaques de créatures ou des obstacles posés par l'environnement. Cela permet de renforcer l'engagement du joueur et de le pousser à explorer et à progresser avec une véritable motivation, plutôt que de simplement remplir des objectifs techniques.

Le storytelling permet aussi d'éviter la monotonie en introduisant des éléments de surprise, des rebondissements et des enjeux qui évoluent avec l'avancement du joueur. Plutôt que de se limiter à des tâches répétitives, il pourra découvrir de nouveaux aspects de l'univers du jeu à travers des indices et des événements scénarisés. En ajoutant cette dimension narrative, *Uzin* devient non seulement un jeu de gestion, mais également une aventure où les choix du joueur influencent le déroulement des événements, apportant ainsi une nouvelle dimension à l'expérience de jeu.

4.1.3 Design (tout court)

Le design visuel de *Uzin* jouera un rôle essentiel pour immerger les joueurs dans l'univers du jeu. Il s'agit de concevoir un style graphique cohérent qui reflète à la fois l'atmosphère mystérieuse et hostile de la planète, tout en restant attrayant et fonctionnel. Les tâches incluront la création des textures, des environnements, et des interfaces utilisateur. Les graphismes doivent être optimisés pour garantir une bonne fluidité, tout en restant suffisamment détaillés pour renforcer l'implication du joueur. Le design des personnages, des créatures, et des machines nécessitera une réflexion approfondie pour qu'ils s'intègrent harmonieusement dans cet univers. Utiliser des outils comme GIMP et Blender nous permettra de réaliser ces éléments visuels tout en restant dans un cadre de coûts réduits, tout en conservant un rendu de qualité professionnelle. Le design devra également être conçu de manière à rendre l'expérience utilisateur intuitive, facilitant la navigation dans les menus et l'interaction avec les différents systèmes du jeu.

4.1.4 Game Design

Le game design est un élément central dans le développement de *Uzin*, car il définit l'ensemble des règles et des mécaniques qui façonnent l'expérience du joueur. Un bon game design permet de rendre les interactions entre le joueur et le monde du jeu à la fois fluides et stimulantes. Dans *Uzin*, cela implique de concevoir des systèmes de collecte de ressources, d'automatisation et de défense qui soient intuitifs tout en offrant une profondeur stratégique. L'équilibre entre la simplicité des actions de base, comme l'extraction de ressources, et la complexité des choix à faire pour optimiser ses infrastructures est essentiel pour garantir que le joueur reste engagé et trouve du plaisir à progresser dans le jeu. Le game design va également au-delà de la mécanique : il s'agit aussi de créer une courbe de difficulté bien pensée, où les défis augmentent progressivement et où chaque étape du jeu apporte de nouvelles surprises et possibilités.

Un autre aspect clé du game design dans *Uzin* est la capacité à immerger le joueur dans un monde cohérent et vivant. Cela ne repose pas seulement sur les graphismes ou l'histoire, mais aussi sur la manière dont les différents systèmes de jeu interagissent entre eux. Par exemple, le joueur doit non seulement gérer ses ressources et optimiser ses chaînes de production, mais aussi réagir aux menaces provenant de l'environnement. Ces menaces doivent être conçues de façon à créer des moments de tension et de réflexion, où le joueur doit adapter ses stratégies en temps réel. Le game design est ainsi le fil conducteur qui relie tous les aspects du jeu, assurant que l'expérience soit à la fois captivante et

équilibrée, tout en donnant au joueur une réelle satisfaction à chaque étape de sa progression.

4.1.5 Multijoueur (réseau)

Le mode multijoueur en coopération dans *Uzin* sera entièrement centré sur une expérience collaborative, où les joueurs uniront leurs efforts pour atteindre des objectifs communs. L'implémentation de ce mode se fera via un système en réseau, permettant à plusieurs joueurs de se connecter simultanément à une même carte. Cet aspect sera crucial pour garantir une expérience fluide, notamment en ce qui concerne la synchronisation des actions, la gestion des ressources partagées et la coordination en temps réel. Chaque joueur pourra gérer sa propre partie des infrastructures, tout en collaborant étroitement avec les autres pour optimiser les chaînes de production et défendre les installations communes contre les menaces locales. Une architecture de serveur robuste sera nécessaire pour permettre une connexion stable et un échange d'informations rapide entre les joueurs, évitant ainsi les décalages et les interruptions qui pourraient nuire à l'expérience coopérative.

Le mode coopératif permettra aux joueurs de mettre en place des stratégies communes, en répartissant les tâches selon les forces de chacun. Par exemple, certains joueurs pourront se concentrer sur l'extraction des ressources, tandis que d'autres géreront l'automatisation ou la défense. La communication et la coordination en temps réel seront des éléments clés du gameplay, renforçant l'aspect collectif et encourageant la collaboration. Le réseau devra également permettre un système de partage de ressources en temps réel, pour que les joueurs puissent échanger des matériaux ou des informations directement, sans retards qui pourraient déséquilibrer le jeu.

L'intégration d'un mode en réseau apportera une toute nouvelle dimension à l'expérience de *Uzin*. Non seulement cela permettra aux joueurs de partager leurs compétences et stratégies, mais cela créera également une dynamique où la réussite individuelle dépend directement de la réussite collective. Le jeu en coopération renforcera la notion d'équipe, et grâce à une architecture réseau efficace, les sessions de jeu resteront engageantes, sans interruption, et ouvertes à une expérience fluide et enrichissante.

4.1.6 Site web

La création d'un site internet sera un élément indispensable pour assurer la visibilité et la communication autour de *Uzin*. Un site web servira de plateforme centrale pour présenter le projet au public, aux potentiels joueurs, ainsi

qu'aux partenaires et peut-être même aux investisseurs ! C'est un outil essentiel pour diffuser des informations clés sur le jeu : sa vision, ses fonctionnalités, les actualités de son développement et, à terme, sa date de sortie. Le site pourra également héberger des captures d'écran, des vidéos de gameplay, et des démos pour donner un aperçu du jeu et attirer l'attention des futurs joueurs. Un espace dédié aux mises à jour régulières du développement permettra également de créer une communauté engagée, intéressée par l'évolution du projet, tout en renforçant la crédibilité de notre travail.

En plus de la présentation du jeu, le site pourra héberger un blog ou des articles destinés à partager les coulisses du développement. Cela permettrait aux visiteurs de découvrir les différentes étapes de la création de *Uzin*, mais aussi d'approfondir certains aspects comme le game design, l'intelligence artificielle ou l'histoire. Un espace de contact serait aussi nécessaire pour permettre aux joueurs de nous joindre directement.

4.2 Répartition des tâches

Nous avons réparti la création du projet de manière équitable, en tenant compte des compétences spécifiques de chaque membre. Toutefois, chacun contribue à la programmation générale du jeu, garantissant une approche collaborative.

Les tâches répertoriées dans le tableau ci-dessous sont relativement spécifiques et visent à organiser efficacement les différentes étapes du développement, tout en laissant la possibilité à chacun d'apporter son aide là où c'est nécessaire.

	Tristan	Martin	Cyril	Maxan
Intelligence artificielle				
Histoire et storytelling				
Design				
Game Design				
Réseau				
Site web				

Responsable
Assistant

4.3 Moyens utilisés

Le cahier des charges est rédigé sur Overleaf, un éditeur en ligne collaboratif basé sur LaTeX. Le développement du programme du jeu se fera sur Visual Studio Code et Rider, deux éditeurs de code. Le jeu sera programmé en CSharp et fonctionnera sur le moteur Unity, l'un des plus utilisés dans l'industrie du jeu vidéo. Les graphismes seront travaillés avec Pixel Studio. Le HTML et le CSS seront employés pour la création du site web. La communication au sein de l'équipe se fera via Discord, et un groupe GitHub sera mis en place pour gérer les répertoires du jeu et du site web.

4.4 Budget prévisionnel

Étant étudiants, notre objectif est de réduire au maximum les coûts liés au développement du projet *Uzin*, en tirant parti des outils gratuits et des ressources disponibles. Cependant, certains éléments pourraient entraîner des dépenses si nous devons recourir à des services ou logiciels payants. Par exemple, si nous décidons d'utiliser une licence professionnelle Unity Pro pour accéder à des fonctionnalités avancées, cela pourrait représenter un coût significatif. Le marketing, comme la publicité sur les réseaux sociaux ou des partenariats avec des créateurs de contenu, pourrait également être coûteux pour promouvoir le jeu efficacement. Le mode multijoueur implique également l'achat de serveurs de bonne qualité. Enfin, la mise en place d'un site web professionnel, incluant l'hébergement et la gestion du nom de domaine, serait une dépense à prendre en compte. Nous cherchons cependant à éviter ces dépenses en privilégiant les solutions gratuites et open-source.

4.5 Avancement du projet

Nous espérons nous tenir aux délais ci-dessous en suivant rigoureusement le planning établi, tout en assurant une progression régulière dans chaque phase du développement pour garantir la qualité finale du projet.

Tâches	**Soutenance 1**	**Soutenance 2**	**Soutenance 3**
Intelligence artificielle	0%	50%	100%
Histoire et storytelling	70%	80%	100%
Design	10%	70%	100%
Game design	50%	80%	100%
Réseau	0%	40%	100%
Site web	0%	90%	100%

5 L'histoire du jeu

5.1 L'importance de l'histoire dans Uzin

Le lore d'*Uzin* n'est pas un simple ajout esthétique ou narratif au jeu : il constitue le pillier de l'expérience immersive que nous souhaitons offrir. Dès le début du projet, nous avons compris que l'histoire du jeu devait servir de fil conducteur, donnant un sens à chaque aspect de gameplay et à chaque choix de design. Dans un jeu de gestion comme *Uzin*, où le joueur doit collecter, automatiser, et défendre ses installations, une trame narrative forte permet de renforcer son engagement et de lui offrir une perspective émotionnelle et intellectuelle sur ses actions.

L'histoire du jeu agit également comme un outil pour connecter les joueurs à l'univers que nous avons créé. Elle contextualise leurs efforts, justifie les défis qu'ils rencontrent, et propose une réflexion plus profonde sur des thèmes comme la survie, l'impact humain sur les environnements naturels, et le progrès technologique. Dans un jeu où la solitude et l'isolation jouent un rôle central, le lore vient enrichir cette ambiance en ajoutant une certaine profondeur philosophique. Voici comment nous avons développé l'univers et les thématiques centrales du jeu.

5.2 Le contexte narratif : un univers dystopique

5.2.1 Un monde en crise

L'histoire se déroule en l'an 2287, dans un contexte où l'humanité a atteint un sommet démographique sans précédent : 42,7 milliards d'habitants. Cette explosion de population a poussé la Terre au bord du gouffre, ses ressources s'épuisant à un rythme alarmant. Incapable de soutenir cette densité humaine, l'humanité a été contrainte de trouver des solutions radicales pour survivre.

C'est dans ce contexte que le programme d'exploration interstellaire a vu le jour. Inspiré des anciens services militaires obligatoires, ce programme impose à chaque individu, dès leur majorité, de quitter la Terre pour participer à des missions d'extraction de ressources sur des planètes lointaines. Cette mesure drastique illustre un monde où le progrès technologique cohabite avec des décisions impitoyables dictées par la nécessité.

5.2.2 Une découverte révolutionnaire

Au cœur de ce programme se trouve une matière organique exceptionnelle qui a révolutionné le voyage spatial. Cette substance, utilisée comme carburant

pour des réacteurs ultra-performants, permet d'envoyer des milliers d'explorateurs à travers l'espace. Cependant, cette avancée cache une réalité sombre : chaque explorateur reçoit uniquement de quoi financer un aller simple vers la planète d'extraction. Le retour sur Terre est conditionné à la réussite de leur mission, faisant de chaque voyage un pari risqué.

5.3 Thèmes centraux et ambiance narrative

5.3.1 Un environnement solitaire et brut

L'histoire d'*Uzin* met l'accent sur la solitude du joueur dans un monde hostile. Bien qu'un mode multijoueur soit prévu, les cartes et environnements sont conçus pour refléter une absence totale de vie humaine. Aucune ville abandonnée, aucun vestige de civilisation : tout est brut, naturel, et exempt d'intervention humaine, du moins au début du jeu.

Cette approche renforce l'idée que le joueur est un pionnier, seul face à une planète indomptée. La narration vient appuyer cette ambiance, en racontant l'histoire de ces jeunes envoyés seuls dans l'espace, sans garantie de retour. Cette solitude devient une composante clé du gameplay, influençant les choix et les émotions des joueurs.

5.3.2 L'impact humain sur l'environnement

Un thème central de l'histoire est la transformation des environnements naturels par l'action humaine. En commençant sur une planète préservée, les joueurs doivent exploiter les ressources et modifier le paysage pour survivre. Ce choix thématique soulève des questions morales : jusqu'où peut-on aller pour garantir sa propre survie ? Quelles sont les conséquences à long terme de ces actions sur un écosystème ?

5.4 Pour conclure sur la narration

L'histoire d'*Uzin* est bien plus qu'un simple cadre narratif : elle est le moteur émotionnel et philosophique du jeu. En mettant en lumière des thèmes comme la solitude, la survie, et l'impact de l'humanité sur la nature, elle enrichit l'expérience des joueurs tout en offrant une réflexion sur des enjeux universels. Ce lore, construit avec soin et en collaboration au sein de l'équipe, constitue une base solide pour soutenir le développement du jeu et captiver son public.

6 Design et graphismes

Le design graphique a constitué l'un des premiers axes de travail dans la réalisation de notre jeu, afin de définir une direction artistique cohérente dès le départ. Nous avons souhaité poser des bases visuelles solides en créant nos propres éléments graphiques, dans le but de garantir une identité unique et maîtrisée. De nombreux éléments graphiques ont été ajoutés depuis la dernière soutenance. C'est notamment le cas des minerais, machines, et des personnages non-joueurs.

6.1 Choix de l'équipement

Dans le cadre de la création graphique de *Uzin*, le choix du logiciel destiné à la réalisation du pixel art a constitué une étape déterminante. Il était essentiel d'opter pour un outil à la fois accessible, performant et compatible avec notre workflow. Nous avons ainsi évalué plusieurs critères tels que l'intuitivité de l'interface, la réactivité des outils de gestion des calques et la capacité d'intégration avec les périphériques tels que la tablette graphique. Après avoir testé plusieurs options, c'est **Pixel Studio** sur *Steam* qui a été retenu.

Nous avons tout d'abord envisagé *Aseprite*, reconnu pour sa spécialisation en pixel art. Bien que ce logiciel propose une large gamme d'outils spécifiques, son interface, bien que fonctionnelle, paraissait vieillissante et ne répondait pas pleinement aux exigences de modernité et de fluidité de notre processus créatif. De plus, le prix relativement élevé, justifié par sa spécialisation, représentait un investissement conséquent pour un projet nécessitant de nombreuses itérations graphiques.

Ensuite, *Photoshop* a été considéré pour sa puissance et sa renommée dans le domaine du graphisme. Toutefois, son caractère généraliste, sa complexité ainsi que le coût de son abonnement mensuel en ont fait une option moins adaptée aux besoins spécifiques du pixel art. La courbe d'apprentissage importante imposée par cet outil est également un frein dans un contexte où l'efficacité et la rapidité sont primordiales.

Quant à *Piskel*, sa gratuité et sa simplicité d'accès en font une option attrayante pour les débutants. Cependant, ses fonctionnalités limitées, notamment en ce qui concerne la gestion avancée des calques et la création d'animations fluides, l'ont rendu inadapté pour répondre aux exigences de qualité et de complexité attendues dans *Uzin*.

En définitive, **Pixel Studio** s'est distingué par sa modernité, sa facilité

d'utilisation et ses nombreuses fonctionnalités spécifiques au pixel art. Son interface intuitive et ses outils de gestion des calques ainsi que des animations se sont révélés être des atouts majeurs, en particulier pour un usage optimisé sur tablette graphique. Le rapport qualité-prix attractif et la disponibilité sur *Steam* ont renforcé notre décision, s'inscrivant parfaitement dans notre workflow et nos exigences techniques.

6.2 Le rôle des graphismes

Le design et les graphismes de *Uzin* jouent un rôle central dans l'immersion du joueur. L'adoption d'une carte générée aléatoirement contribue à renouveler constamment l'expérience de jeu, favorisant ainsi l'exploration et offrant à chaque partie un environnement à la fois unique et varié. Cette approche dynamique permet de maintenir l'intérêt des joueurs sur le long terme, en offrant des situations de jeu imprévisibles et stimulantes.

Visuellement, *Uzin* se distingue par un style coloré et dynamique qui capte immédiatement l'attention. La carte est composée de cases modulaires travaillées avec soin, où le sol aux teintes roses vibrantes s'harmonise avec des lacs d'un mauve profond et un système de rebords dynamiques. Ces derniers s'ajustent automatiquement en fonction de la disposition des tuiles d'eau et de sol, garantissant ainsi une cohérence visuelle remarquable tout en préservant la structure modulaire essentielle au gameplay. L'attention portée aux détails permet d'offrir une expérience visuelle immersive, favorisant l'identification rapide des zones de jeu et la fluidité des interactions.

6.3 Le système de calques

Afin d'organiser efficacement l'ensemble des éléments visuels de la carte, un système de calques en trois niveaux a été mis en place. Cette approche hiérarchisée permet une gestion claire et structurée des données et facilite l'interaction entre les différentes composantes du jeu.

— **Le premier calque : le type de sol**
Ce calque définit la nature fondamentale du terrain (herbe, eau, roche, sable, etc.) et influe directement sur les possibilités de déplacement et de construction. Par exemple, une case d'eau ne permet pas le passage ni l'édification de certains bâtiments, alors qu'une case d'herbe peut accueillir des structures simples. Ce niveau sert de base à l'architecture de la carte, établissant les règles premières qui conditionnent le comportement des éléments interactifs.

— **Le deuxième calque : les éléments et ressources**
Ce calque regroupe les objets interactifs et les ressources exploitables, tels que les arbres, les minerais ou les plantations. Il inclut également des caractéristiques environnementales, comme les falaises, les montagnes ou les fosses, qui ajoutent une dimension stratégique au gameplay. En définissant précisément les interactions possibles avec la carte – qu'il s'agisse de l'extraction de ressources ou de la mise en place d'obstacles naturels – ce niveau enrichit l'expérience de jeu tout en offrant une profondeur tactique aux actions du joueur.

— **Le troisième calque : les bâtiments et constructions**
Ce niveau gère la superposition des structures construites sur la carte, en intégrant les informations issues des deux premiers calques. Par exemple, une case d'herbe dotée de ressources spécifiques (comme du bois) pourra accueillir un bâtiment qui exploite ces atouts, tandis qu'une surface rocheuse limitera les options de construction à des structures adaptées aux contraintes du terrain. Ce calque assure ainsi la cohérence et la logique de l'architecture du jeu, garantissant que chaque construction s'insère de manière harmonieuse dans l'environnement prédéfini.

L'intégration de ces trois calques permet de simplifier la gestion globale des éléments de la carte et d'assurer une meilleure lisibilité des interactions possibles :

— **Gestion des déplacements**
La différenciation nette entre les zones navigables et les zones restreintes permet d'interdire le passage sur l'eau ou dans certaines zones escarpées, assurant ainsi une réponse immédiate aux actions du joueur.

— **Logique de construction**
L'organisation en calques permet d'autoriser ou de restreindre la construction en fonction des ressources disponibles et du type de terrain, assurant une cohérence entre l'environnement et les structures édifiées.

— **Modularité**
La structure modulaire du système offre la possibilité d'ajouter facilement de nouveaux types de terrains, ressources ou bâtiments sans bouleverser l'architecture existante, garantissant ainsi une évolutivité continue du jeu.

6.4 Une expérience immersive

L'architecture graphique et technique de *Uzin* a été conçue pour offrir une immersion optimale dès les premiers instants de jeu. Chaque détail visuel, qu'il

s'agisse des textures, des couleurs ou de l'animation des éléments, a été pensé pour améliorer la compréhension du gameplay et faciliter la prise de décisions stratégiques. L'harmonie entre les graphismes et les mécaniques de jeu crée une expérience fluide, où l'utilisateur se trouve rapidement plongé dans un univers cohérent et captivant.

Le choix d'une direction artistique marquée, combiné à un système de calques structurant clairement l'univers du jeu, permet de garantir une excellente lisibilité des éléments et de la carte. Ce système offre également la flexibilité nécessaire pour intégrer de futures améliorations, sans compromettre l'équilibre visuel ou fonctionnel du jeu. Ainsi, l'architecture graphique répond à la fois aux exigences esthétiques et techniques, en assurant une navigation intuitive dans un environnement riche et détaillé.

Enfin, il est important de noter que les graphismes de *Uzin* se situent à un niveau de sophistication très élevé, voire parfois au-delà des autres aspects du jeu. Cette richesse visuelle, bien qu'extrêmement attrayante, implique que lors de la soutenance, il ne sera pas possible de présenter en détail l'ensemble des subtilités graphiques développées jusqu'à présent. Cependant, cette avancée témoigne d'une ambition de qualité, qui laisse présager des améliorations futures destinées à harmoniser l'ensemble des composantes du projet.

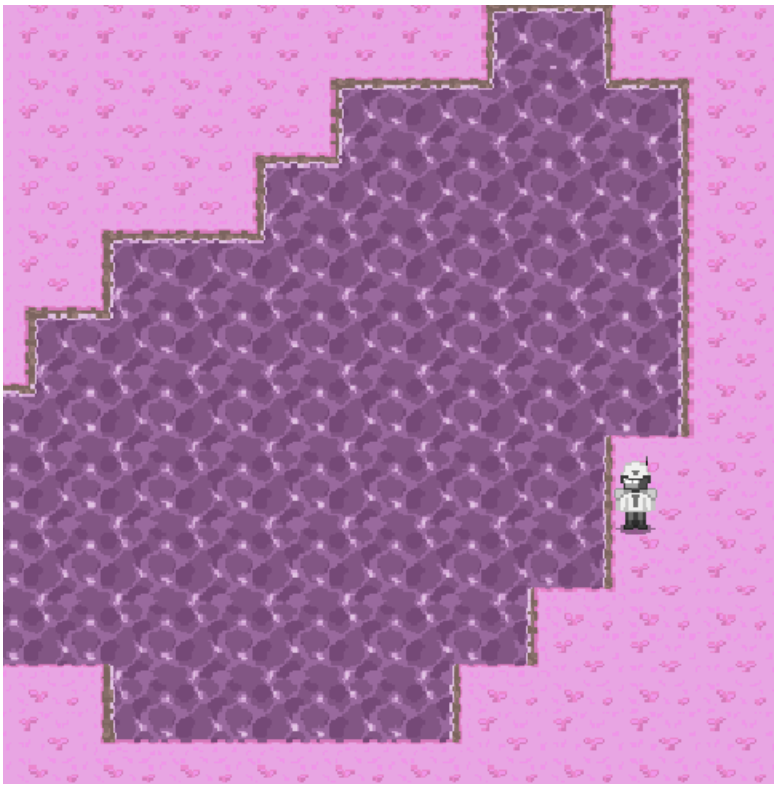

FIGURE 1 – Partie de map avec lac et personnage

7 Génération de la carte

La génération de la carte dans *Uzin* repose sur un système modulaire basé sur des *chunks*, permettant de charger dynamiquement des zones de la carte en fonction de la position du joueur. Ce procédé se divise en deux grandes étapes : la génération des données de chaque chunk à l'aide du bruit de Perlin et le rendu visuel via une Tilemap. Afin d'assurer une expérience fluide, cohérente et synchronisée en mode multijoueur, nous avons dû affiner et optimiser plusieurs aspects techniques du système.

7.1 Création des chunks et génération des données

Chaque chunk est défini par une position dans une grille en deux dimensions et contient un tableau de `TileData` de taille fixe (par exemple, 16 par 16 tuiles). La fonction `GenerateChunk` parcourt chaque cellule du chunk pour calculer, grâce au bruit de Perlin, une valeur d'altitude (ou « height ») qui permet ensuite de sélectionner la tuile correspondante.

Le bruit de Perlin est ajusté par un facteur d'échelle ainsi que par des offsets aléatoires. Ces paramètres garantissent une diversité dans les structures naturelles tout en assurant une certaine continuité visuelle entre les chunks adjacents. Pour renforcer l'immersion, nous avons travaillé à l'harmonisation des variations d'altitude de manière à éviter des ruptures brutales entre zones d'eau, de terre et de bordures. Ce processus itératif a impliqué de nombreux tests, permettant d'affiner les seuils de détection et de mieux simuler des reliefs naturels.

En complément, une attention particulière a été portée sur la gestion de la mémoire et la rapidité d'exécution. La création d'un chunk implique non seulement le calcul de ses tuiles, mais aussi la mise en place d'une structure de données qui permet de les retrouver rapidement lors du rendu ou du déchargement. Cette phase de génération se déroule en arrière-plan, ce qui contribue à éviter des blocages pendant le jeu.

7.2 Gestion des transitions et sélection des tuiles

La fonction `GenerateTile` joue un rôle central dans la détermination de l'apparence finale de chaque tuile. Elle ne se contente pas d'évaluer la valeur de bruit à la position courante, mais prend également en compte celles des tuiles voisines, tant orthogonales que diagonales.

Ce système de vérification complexe permet de gérer avec précision les transitions entre différents types de terrains, notamment entre l'eau et la terre.

Par exemple, lorsque la valeur de bruit est inférieure à un seuil critique (ici, 0.2f), la tuile est considérée comme de l'eau. Toutefois, pour éviter des interfaces trop tranchées, des conditions supplémentaires sont appliquées pour sélectionner des tuiles de bord adaptées aux configurations particulières (coins internes, bords hybrides, coins extérieurs, etc.).

L'optimisation de cette fonction a nécessité plusieurs itérations de tests et d'ajustements. Chaque nouveau cas testé a permis de peaufiner les conditions de transition, rendant les interfaces entre les zones d'eau et de terre plus naturelles et esthétiques. Ces efforts garantissent que le rendu final est en phase avec l'identité visuelle recherchée pour *Uzin*.

7.3 Rendu dynamique sur la Tilemap et synchronisation réseau

Une fois les données d'un chunk générées, la fonction `RenderChunk` se charge de positionner chaque tuile sur la Tilemap en traduisant les coordonnées du chunk vers des coordonnées mondiales. Ce processus assure un rendu dynamique qui s'adapte en temps réel aux déplacements du joueur.

Dans un contexte multijoueur, la cohérence de la carte est primordiale. Pour ce faire, la synchronisation des seeds (via `NetworkVariable`) sur le serveur garantit que tous les joueurs disposent d'une carte identique, même si le chargement des chunks se fait de manière asynchrone. Ce mécanisme de synchronisation repose sur la centralisation de la génération des seeds sur le serveur, évitant ainsi toute divergence dans les valeurs aléatoires qui pourraient autrement provoquer des incohérences visuelles entre les clients.

Le rendu dynamique est également optimisé par un système de mise en cache et de gestion efficace des ressources. En limitant le nombre d'opérations par frame et en actualisant uniquement les zones modifiées, nous avons réussi à minimiser l'impact sur les performances, même lors de déplacements rapides ou dans des environnements très étendus.

7.4 Éléments complémentaires et perspectives d'évolution

En complément de la génération du terrain, plusieurs systèmes ont été intégrés ou amorcés afin d'enrichir la carte sur les plans visuel, interactif et fonctionnel.

Génération des ressources naturelles

Un second bruit de Perlin, indépendant de celui utilisé pour le relief, est exploité pour déterminer la présence de ressources naturelles telles que le

charbon ou le fer. Cette couche repose sur des seeds spécifiques afin de dissocier totalement la logique de génération des minerais de celle des altitudes. Ce découplage permet une distribution plus crédible et maîtrisée des ressources, tout en ouvrant la voie à l'ajout futur d'autres types de matériaux exploitables.

Variation visuelle des tuiles

Pour renforcer la qualité visuelle de l'environnement, un système de variation aléatoire a été mis en place sur certaines tuiles. Par exemple, les sols herbeux peuvent apparaître sous différentes déclinaisons visuelles, sélectionnées aléatoirement à la génération. Cette méthode simple, mais efficace, limite la répétitivité du décor et participe à l'immersion du joueur sans nécessiter d'effort graphique supplémentaire majeur.

Interaction avec les mécaniques de jeu

Le système de génération est étroitement lié au gameplay : certaines tuiles identifiées comme contenant des ressources déclenchent des comportements spécifiques. C'est notamment le cas des gisements de charbon, qui peuvent être détectés automatiquement par le joueur et associés à des objectifs de quête. Cette intégration permet de relier dynamiquement la carte aux systèmes de progression et d'accomplissement, renforçant la cohérence globale du jeu.

Perspectives d'amélioration

La base technique mise en place ouvre plusieurs pistes d'évolution intéressantes. Parmi celles envisagées :

— la génération procédurale de biomes aux caractéristiques distinctes ;
— l'apparition de structures naturelles ou artificielles (villages, cavernes, ruines, etc.) ;
— un système météo influençant l'apparence ou les règles de génération ;
— l'enrichissement des minerais existants et leur profondeur de traitement (raffinage, rareté, etc.).

L'ensemble de ces éléments contribue à poser les fondations d'un monde vivant, modulaire et extensible, conforme à la vision de *Uzin* comme projet évolutif et ambitieux.

7.5 Problèmes rencontrés et solutions apportées

7.5.1 Chargement et déchargement des chunks

Initialement, nous avons constaté que certains chunks restaient en mémoire même lorsque le joueur s'en éloignait, ce qui entraînait une surcharge et

des ralentissements notables.

Pour résoudre ce problème, nous avons instauré une vérification continue de la position du joueur. Grâce à l'utilisation de la fonction `Vector2Int.Distance`, un seuil précis a été défini, permettant de décharger automatiquement les chunks situés au-delà de la distance d'affichage définie (*render distance* + 1).

Cette approche a nécessité la mise en place d'une logique de suivi très fine de la position du joueur, ainsi que des tests de performance pour déterminer le seuil optimal. En conséquence, le système de chargement/déchargement s'est révélé être à la fois réactif et efficace, optimisant l'utilisation de la mémoire et garantissant un jeu fluide.

7.5.2 Incohérences dans les transitions de tuiles

La sélection des tuiles pour les transitions entre zones d'eau et de terre posait problème. Dans certaines configurations, la logique initiale ne suffisait pas à choisir la tuile la plus adaptée, ce qui entraînait des bords irréguliers et visuellement décevants.

Après plusieurs tests et retours d'expérience, nous avons affiné la fonction `GenerateTile`. Nous avons étendu l'analyse pour inclure non seulement les voisins directs mais aussi les voisins diagonaux. Cela a permis d'ajuster avec précision les transitions et de sélectionner des tuiles de bord spécifiques pour chaque configuration.

En parallèle, nous avons intégré un système de logs et de visualisation interne, permettant de repérer rapidement les incohérences et de tester de nouveaux seuils. Ces ajustements successifs ont permis d'obtenir des transitions naturelles et harmonieuses, améliorant significativement la qualité visuelle de la carte.

7.5.3 Synchronisation des seeds en mode multijoueur

En mode réseau, une divergence dans les valeurs aléatoires utilisées pour générer le bruit de Perlin (offsets et seeds) pouvait entraîner des cartes différentes entre le serveur et les clients.

Pour pallier ce problème, la génération des seeds a été centralisée sur le serveur. En utilisant des `NetworkVariable` pour `mapSeedX` et `mapSeedY`, le serveur génère les valeurs aléatoires et les propage à tous les clients de manière transparente.

Cette solution assure que chaque client dispose des mêmes paramètres de génération, éliminant ainsi toute possibilité de divergence. Des tests en environnement multijoueur ont confirmé que cette approche permettait une synchronisation parfaite, garantissant une expérience de jeu uniforme pour tous les participants.

7.5.4 Problèmes de performance lors de la génération des chunks

La génération en temps réel de la carte, notamment le calcul du bruit de Perlin pour chaque tuile, représente une charge de calcul non négligeable. Le chargement simultané de plusieurs chunks pouvait ainsi entraîner des baisses de performance, affectant l'expérience utilisateur.

Pour améliorer les performances, plusieurs optimisations ont été mises en œuvre :

— **Réduction des appels redondants :** Les boucles de génération ont été optimisées pour éviter les calculs inutiles. Les paramètres tels que la taille des chunks et les offsets sont calculés une seule fois par chunk, puis réutilisés.

— **Mise en cache des résultats :** Certaines valeurs issues du calcul du bruit de Perlin sont mises en cache, notamment pour les tuiles ayant des voisins identiques ou pour des configurations répétitives. Cela permet de réduire significativement le nombre d'appels à `Mathf.PerlinNoise`.

— **Gestion efficace des ressources :** L'utilisation d'un dictionnaire pour stocker les chunks chargés permet de limiter les opérations de recherche et de mise à jour aux seuls chunks actifs. Par ailleurs, des techniques de pooling d'objets ont été envisagées pour réutiliser les chunks déchargés, évitant ainsi la création et la destruction répétées d'objets.

En résumé, la mise en place du système de génération de la carte dans *Uzin* a nécessité un travail d'optimisation et de réglage fin afin d'assurer une expérience fluide et immersive. La gestion dynamique des chunks, combinée à des techniques avancées de sélection de tuiles via le bruit de Perlin, permet de créer un environnement de jeu riche et cohérent, tant en mode solo qu'en multijoueur. Chaque étape, de la génération des données à leur rendu sur la Tilemap, a été minutieusement étudiée et optimisée pour répondre aux exigences de qualité et de performance du projet.

8 Multijoueur

Nous avons décidé de prioriser l'implémentation du mode multijoueur dès les premières phases de développement, afin d'éviter de devoir refondre ou adapter ultérieurement l'ensemble du système. Cette approche précoce nous a permis de prendre en compte les exigences réseau dès le début, influençant ainsi la conception globale du jeu.

8.1 Choix de l'architecture réseau

Après une étude comparative des différentes architectures disponibles, deux solutions principales s'offraient à nous :

— **Serveur centralisé :** Dans ce modèle, un serveur dédié gère toutes les données du jeu et tous les joueurs se connectent à ce serveur. Bien que cette approche garantisse une synchronisation quasi instantanée et soit idéale pour les jeux compétitifs nécessitant une faible latence, elle présente l'inconvénient de nécessiter une infrastructure serveur payante et plus complexe à maintenir.

— **Peer-to-peer (P2P) :** Dans ce modèle, l'un des joueurs agit en tant qu'hôte, servant de serveur pour les autres clients. Cette solution est souvent privilégiée pour les jeux coopératifs ou les parties à effectif réduit, car elle permet de réduire les coûts et la complexité de la gestion des serveurs.

Nous avons opté pour l'architecture peer-to-peer, adaptée à notre jeu coopératif. Ainsi, chaque joueur se connecte au PC de l'hôte qui centralise la gestion des données de la partie. Cette décision a permis d'éviter des coûts supplémentaires liés à l'hébergement d'un serveur dédié, tout en offrant une expérience de jeu fluide pour un nombre limité de joueurs.

8.2 Choix de l'outil réseau

Une fois l'architecture choisie, nous avons évalué plusieurs frameworks permettant de gérer le multijoueur sous Unity. Les trois candidats principaux étaient :

— **Mirror,**

— **Photon,**

— **Unity NetCode for GameObjects.**

Étant donné que `Unity NetCode for GameObjects` est déjà intégré dans Unity et offre une prise en main relativement simple, nous avons décidé de l'utiliser. Ce framework nous a permis de bénéficier d'une intégration native, de fonctionnalités telles que les `NetworkVariables` et les `Server RPC`, et d'une communauté active pour le support technique.

8.3 Création et gestion des sessions multijoueur

Nous avons développé un menu dédié qui permet au joueur de créer ou de rejoindre un salon multijoueur. Lorsqu'un joueur crée une partie, son PC se comporte comme l'hôte, initialisant ainsi un serveur local. Dès qu'un autre joueur se connecte, un nouveau modèle préfabriqué (prefab) représentant le joueur est instancié sur la scène, assurant ainsi une représentation synchronisée de tous les participants. Ce mécanisme a nécessité quelques adaptations, notamment pour :

— La gestion des caméras, puisque le jeu initial ne comportait qu'une seule caméra. Lorsqu'un second joueur se connectait, il voyait la même vue que l'hôte. Pour résoudre ce problème, nous avons implémenté une solution où chaque joueur se voit attribuer une caméra dédiée qui suit ses propres déplacements.

— La gestion de la génération de la carte. Initialement, la carte se générait autour du joueur local. Cependant, en mode multijoueur, chaque client devait recevoir les mêmes informations concernant les "chunks" (morceaux de carte) pour garantir la cohérence de l'environnement de jeu. Nous avons donc recours aux `Server RPC` pour envoyer des requêtes depuis les clients vers l'hôte, qui se charge de générer et de renvoyer les données appropriées. Ainsi, la carte se génère de manière dynamique et personnalisée sur l'écran de chaque joueur, tout en conservant une structure globale cohérente.

8.4 Problèmes rencontrés et solutions apportées

8.4.1 Synchronisation des spawn Points et des caméras

Lors des premiers tests, nous avons constaté que lorsqu'un nouveau joueur rejoignait la partie, il utilisait la même caméra que l'hôte, entraînant ainsi un suivi inapproprié et une vision partagée qui nuisaient à l'immersion et à la jouabilité. Ce problème posait également des difficultés en termes de positionnement, le système de spawn n'étant pas suffisamment robuste pour garantir un point d'apparition sûr pour chaque joueur.

Pour résoudre ce problème, nous avons modifié le système d'instanciation afin que chaque joueur se voie attribuer une caméra dédiée, créée lors de la connexion via le prefab du joueur. Par ailleurs, la fonction `FindSafeSpawn` a été améliorée pour rechercher et centrer chaque joueur sur une tuile sûre, garantissant ainsi un spawn cohérent et sécurisé pour tous.

8.4.2 Génération de la carte en mode multijoueur

Initialement, la génération de la carte était centrée sur la position du joueur local, ce qui fonctionnait bien en mode solo mais posait problème en multijoueur. Chaque client recevait une partie de la carte basée sur sa propre position, entraînant des incohérences dans le rendu global et des difficultés pour synchroniser l'environnement de jeu entre les différents joueurs.

Pour pallier cette difficulté, nous avons implémenté des `Server RPC` qui permettent aux clients d'envoyer des requêtes à l'hôte pour générer les "chunks" de carte autour de leur position. Ainsi, l'hôte centralise la génération et la distribution des segments de carte, assurant que chaque joueur visualise une carte cohérente et synchronisée, tout en optimisant les performances en ne générant que les portions nécessaires.

8.4.3 Incohérence de la "seed" de la carte

Nous avons remarqué que, dans la version initiale, la seed de génération de la carte était générée aléatoirement à chaque lancement du jeu, ce qui pouvait conduire à ce que deux joueurs dans la même partie se retrouvent sur des cartes différentes. Cette divergence entraînait des désynchronisations majeures, compromettant l'intégrité de l'expérience multijoueur.

La solution apportée a consisté à centraliser la génération de la seed sur l'hôte en utilisant les `NetworkVariables` de `Unity NetCode for GameObjects`. En diffusant la même seed à tous les clients, nous garantissons que l'environnement de jeu reste identique pour tous les participants, assurant ainsi une synchronisation parfaite de la carte.

8.4.4 Problèmes de latence et fluidité des mouvements

Des tests en conditions réelles ont révélé que les mouvements des joueurs étaient parfois saccadés ou décalés, en particulier dans des situations de latence réseau élevée. Ces saccades affectaient non seulement l'expérience de jeu, mais pouvaient également induire des désynchronisations temporaires entre les actions des joueurs et leur représentation visuelle.

Pour remédier à ce problème, nous avons optimisé le traitement des entrées en normalisant les vecteurs de déplacement et en appliquant les mises à jour dans `FixedUpdate` pour assurer une régularité accrue. Par ailleurs, nous avons implémenté une interpolation des positions des joueurs distants, permettant d'adoucir les transitions et d'offrir une expérience visuelle fluide même en cas de fluctuations du réseau.

8.5 Interactions multijoueur : intégration poussée au gameplay

Au-delà de la simple synchronisation des déplacements ou de la carte, nous avons intégré en profondeur les mécaniques multijoueur dans le cœur du gameplay. L'objectif était d'offrir une expérience coopérative cohérente et immersive, où chaque action importante est reconnue, partagée et validée à l'échelle de la session. Cette section présente les systèmes interactifs majeurs mis en œuvre dans notre architecture réseau.

8.5.1 Synchronisation des entités dynamiques

Dans un environnement multijoueur, la gestion des entités interactives (machines, ennemis, ressources) représente un défi essentiel. Nous avons implémenté un système basé sur le composant `NetworkObject`, qui permet de faire exister une entité de façon cohérente entre tous les clients. Lorsqu'un joueur souhaite interagir avec le monde (placer une foreuse, activer une machine), une requête `ServerRpc` est envoyée au serveur via la classe `NetworkSpawner`, qui instancie l'objet concerné et le synchronise automatiquement avec tous les clients.

Le choix de centraliser les instanciations côté serveur permet d'assurer l'intégrité du jeu, d'éviter les doublons ou désynchronisations, et de valider chaque action par une autorité unique. Cette architecture s'est avérée particulièrement efficace pour maintenir la stabilité lors des interactions fréquentes, comme le placement en série de structures.

8.5.2 Système de combat multijoueur et gestion des dégâts

Le système de tir repose sur des raycasts 2D, déclenchés à chaque clic du joueur. Lorsqu'un projectile touche un ennemi, la détection s'effectue localement, mais la gestion des dégâts est relayée au serveur. C'est le composant `HealthNetwork` qui gère l'état de santé des ennemis grâce à une `NetworkVariable` représentant les points de vie actuels.

Ce modèle garantit une autorité serveur sur toutes les décisions critiques (dégâts, mort, effets visuels), ce qui prévient toute tentative de triche ou d'incohérence entre clients. En cas de mort confirmée, le serveur déclenche les effets visuels comme le `FadeOut` et gèle le comportement de l'ennemi, le retirant ainsi proprement du champ d'action.

Nous avons également intégré un effet de retour immédiat via `CameraShake`, pour améliorer la sensation d'impact, bien que cette partie reste localisée sur chaque client.

8.5.3 Collecte et traitement des ressources à plusieurs

Le système de minage repose sur la classe `Mining`, qui identifie la tuile sur laquelle se trouve le joueur, vérifie sa nature (charbon ou fer), et déclenche un ajout dans l'inventaire correspondant. Chaque ressource est représentée visuellement sur la carte grâce aux tuiles générées par le bruit de Perlin (cf. section précédente), ce qui rend leur exploitation à la fois visible et interactive.

L'action de minage est couplée au système de quêtes via la détection de conditions spécifiques (extraction de charbon, progression de quête en cours, etc.). Cette mécanique est donc entièrement intégrée à l'expérience multijoueur, où chaque joueur peut contribuer à des objectifs communs ou progresser individuellement selon l'état de la partie.

À terme, ce système pourra être enrichi par une gestion d'inventaires partagés, des coffres synchronisés ou des échanges de ressources entre joueurs.

8.5.4 Gestion avancée du spawn et de la caméra

Lorsqu'un joueur rejoint une partie en cours, il est automatiquement instancié via un prefab spécifique. Pour éviter les conflits de caméra (chaque client voyant initialement la même vue que l'hôte), une caméra indépendante est générée et assignée dynamiquement à chaque joueur. Cette caméra suit son mouvement grâce à un système de liaison, garantissant une expérience fluide et isolée pour chaque participant.

Concernant la position de spawn, nous avons développé une fonction `FindSafeSpawn` qui recherche une tuile terrestre libre (non recouverte d'eau) autour du centre de la carte. Cette vérification garantit que chaque joueur commence dans un espace sûr, sans collision immédiate avec un obstacle ou un autre joueur.

8.5.5 Performances et fluidité en situation réelle

Afin d'assurer la fluidité des interactions dans des conditions de réseau variables, plusieurs optimisations ont été apportées :

— les déplacements sont traités dans `FixedUpdate` pour une meilleure stabilité ;
— les entrées sont normalisées pour éviter les accélérations non souhaitées ;
— les positions des autres joueurs sont interpolées pour adoucir leur mouvement, même en cas de latence.

9 Inventaire

La mise en place du système d'inventaire dans *Uzin* constitue l'un des piliers fondamentaux du gameplay. Il permet aux joueurs de collecter, stocker, utiliser, organiser et potentiellement échanger les objets obtenus tout au long de leur progression. Dans un environnement multijoueur, il est essentiel que ce système soit à la fois réactif, fiable et parfaitement synchronisé entre tous les participants pour garantir une expérience fluide et équitable.

9.1 Synchronisation et fonctionnement général

Dès sa conception, l'inventaire a été pensé pour fonctionner dans un environnement connecté. Toute action effectuée sur l'inventaire d'un joueur — que ce soit l'ajout d'un objet après extraction, sa suppression, ou son utilisation — est instantanément répercutée sur tous les clients concernés. Cela garantit que l'état des inventaires reste identique entre les différents joueurs, évitant toute divergence qui pourrait compromettre l'équilibre de la partie.

Chaque objet est défini par un identifiant, une image représentative, une quantité, et des attributs associés (par exemple, sa catégorie, sa rareté ou ses propriétés d'utilisation). Lorsqu'un objet est manipulé, sa représentation graphique est mise à jour en conséquence. Le système prend également en charge l'empilement des objets similaires, ce qui permet d'optimiser l'espace et de faciliter la gestion pour le joueur.

9.2 Interface utilisateur et accessibilité

L'interface d'inventaire a été pensée pour allier clarté, accessibilité et réactivité. Elle se présente sous forme de grille d'emplacements organisés dans un panneau dédié, accessible à tout moment via un raccourci clavier. Chaque emplacement affiche une icône d'objet, son nom, et la quantité actuellement possédée. L'ouverture de l'inventaire interrompt partiellement l'interaction avec le monde, permettant au joueur de se concentrer sur sa gestion d'objets sans perturbation.

L'inventaire est accompagné d'animations d'ouverture et de fermeture, ainsi que de transitions visuelles qui rendent l'expérience plus fluide. Le joueur peut ainsi consulter ses ressources, en ajouter ou en retirer, tout en conservant une bonne lisibilité de l'état général de son équipement.

9.3 Catégorisation, tri et filtrage

Une attention particulière a été portée à la lisibilité et à la classification des objets. Les objets sont regroupés en différentes catégories : ressources, matériaux de construction, objets consommables, équipements, etc. Cette catégorisation facilite la navigation au sein de l'inventaire et permet au joueur d'accéder plus rapidement aux éléments dont il a besoin.

Des fonctionnalités de tri automatique ou manuel peuvent être activées pour organiser les objets par type, quantité ou date d'obtention. Cette possibilité de filtrage offre une plus grande maîtrise de l'interface, en particulier lorsque le joueur commence à accumuler un grand nombre d'objets variés.

9.4 Interactions avec le monde et mécaniques de jeu

L'inventaire n'est pas un élément isolé : il est étroitement lié aux autres systèmes du jeu. Par exemple, lorsqu'un joueur extrait une ressource ou interagit avec un objet du décor, l'inventaire est immédiatement mis à jour pour refléter ce changement. De la même manière, l'utilisation d'un objet (comme un minerai servant à alimenter une machine ou un composant de construction) déclenche sa suppression ou sa transformation dans l'inventaire.

Certains objets ont des effets spécifiques lorsqu'ils sont utilisés : augmentation temporaire de la vitesse, déclenchement d'un événement, déblocage d'un plan de fabrication, etc. Ces interactions enrichissent l'expérience et donnent au joueur un véritable contrôle sur son environnement via la gestion de ses possessions.

9.5 Évolutivité et modularité du système

Le système d'inventaire a été conçu pour être modulaire et extensible. Il est possible d'y ajouter de nouvelles catégories d'objets, des effets spéciaux, des limitations de quantité, ou encore des emplacements verrouillés à débloquer en cours de jeu. Cette modularité permet d'adapter l'inventaire à la progression du joueur et aux phases successives du gameplay.

De plus, le système prend en charge les extensions futures comme l'ajout d'un stockage partagé entre plusieurs joueurs (ex. : coffre d'équipe), d'un système de fabrication basé sur les objets collectés, ou d'un espace personnel personnalisable.

9.6 Défis rencontrés et solutions mises en place

9.6.1 Synchronisation entre les joueurs

L'un des premiers défis a été de garantir que tous les joueurs voient le même contenu d'inventaire au même moment. Pour cela, nous avons mis en place un système centralisé validant toutes les actions critiques (ajout, suppression, échange) afin d'éviter les désynchronisations, en particulier dans les situations de latence.

9.6.2 Réactivité et confort d'utilisation

Un certain délai pouvait être ressenti entre une action (comme miner un minerai) et sa prise en compte dans l'inventaire. Des optimisations ont été appliquées sur les mécanismes de retour visuel et sur la fréquence de mise à jour de l'interface pour que les interactions soient perçues comme immédiates.

9.6.3 Empilement automatique et fusion des objets

Des doublons pouvaient apparaître lorsque plusieurs objets identiques étaient ajoutés simultanément. Le système d'empilement a été ajusté pour détecter les cas similaires et fusionner les objets en un seul emplacement, tout en actualisant leur quantité.

9.6.4 Gestion des conflits lors d'échanges

Lors d'interactions rapides (ex : extraction suivie d'une consommation immédiate), des conflits d'état pouvaient apparaître. Un mécanisme de vérification en plusieurs étapes a été introduit pour garantir que l'objet existe bien avant toute manipulation critique.

9.7 Perspectives d'évolution

Le système d'inventaire, dans sa forme actuelle, offre une base solide, stable et extensible. Il a été conçu pour répondre aux besoins fondamentaux du gameplay, mais il peut être enrichi de nombreuses fonctionnalités supplémentaires afin d'accompagner la montée en complexité du jeu et d'offrir davantage de possibilités aux joueurs. Les axes suivants représentent les pistes principales d'évolution identifiées à ce stade du développement.

9.7.1 Échanges entre joueurs

Actuellement, les joueurs disposent chacun de leur inventaire personnel. Introduire un système d'échange permettrait d'instaurer une nouvelle couche

d'interaction sociale, renforçant la coopération et l'économie interne du jeu. Cela pourrait se traduire par une interface dédiée où les deux joueurs déposent les objets qu'ils souhaitent échanger, avec validation mutuelle avant confirmation de l'opération.

Un échange sécurisé impliquerait aussi des vérifications du côté du serveur afin d'éviter les duplications, les pertes accidentelles ou les manipulations malveillantes. Des options avancées pourraient être envisagées, telles que des propositions de troc, des échanges conditionnels (ex. : 10 minerais contre 1 foreuse), voire des systèmes de crédit ou de prêt d'objets. Cette fonctionnalité ouvrirait la voie à un gameplay de type marchand ou diplomatique en multijoueur.

9.7.2 Durabilité et usure

Dans sa version actuelle, l'inventaire gère les quantités, mais tous les objets sont persistants tant qu'ils ne sont pas consommés. L'ajout d'un système d'usure introduirait un facteur temporel, limitant l'utilisation de certains objets à un nombre défini d'actions ou à une durée précise.

9.7.3 Favoris et marquage

Plus un inventaire s'étoffe, plus la navigation devient complexe. Permettre au joueur de marquer certains objets comme "favoris", ou de les regrouper selon des usages spécifiques (combat, construction, revente…), faciliterait grandement leur gestion quotidienne.

9.7.4 Compatibilité avec les quêtes et événements

Un lien plus direct entre l'inventaire et le système de quêtes permettrait d'intégrer les récompenses ou les objectifs narratifs de manière plus fluide. Lorsqu'un joueur accomplit une mission, les objets reçus pourraient apparaître automatiquement dans son inventaire, accompagnés d'un message ou d'une animation dédiée.

En somme, l'inventaire de *Uzin* n'est pas simplement une réserve de ressources, mais un espace d'interactions entre le joueur, l'univers du jeu, et les autres participants. Sa conception ouverte et évolutive permet non seulement d'accompagner les besoins immédiats du gameplay, mais aussi de soutenir les ambitions futures du projet.

10 Système de fabrication (crafting)

La fabrication d'objets, aussi appelée "craft", constitue une mécanique centrale dans *Uzin*. Elle permet aux joueurs de combiner des ressources collectées dans le monde pour produire des objets plus avancés, nécessaires à leur progression : outils, machines, composants ou éléments rares. Ce système prolonge logiquement l'extraction de ressources en introduisant une phase de transformation, rendant l'inventaire et la gestion des stocks encore plus stratégiques.

10.1 Fonctionnement général

Le système de craft repose sur un principe simple : chaque recette est composée d'un certain nombre d'ingrédients (ressources de base) et produit un ou plusieurs objets finis. Le joueur sélectionne une recette via une interface graphique, et, s'il possède les ressources nécessaires, il peut déclencher la fabrication.

Chaque recette est définie par :
— un objet résultant (produit fini),
— une quantité à produire,
— une liste de ressources à consommer (ingrédients).

La fabrication peut être effectuée directement depuis une interface dédiée, accessible via un raccourci clavier. Une fois ouverte, cette interface affiche l'ensemble des recettes disponibles, avec leur nom, leur image, la quantité produite, et les ressources requises.

10.2 Interface et expérience utilisateur

L'ouverture du panneau de fabrication est déclenchée par la touche `C`. L'interface de crafting présente une liste de recettes, chacune affichée sous forme de carte contenant une icône, un nom, et une quantité. Lorsqu'une recette est sélectionnée, ses détails sont mis en avant dans une section dédiée, permettant au joueur de confirmer son choix.

La sélection d'une recette configure dynamiquement le panneau de confirmation. Si le joueur tente de fabriquer sans avoir les ingrédients requis, un message d'erreur s'affiche brièvement à l'écran. Ce retour visuel rapide permet de comprendre immédiatement si l'action est possible ou non.

10.3 Vérification et exécution du craft

Lorsque le joueur déclenche la fabrication :

1. Le jeu vérifie s'il dispose bien de toutes les ressources requises.

2. Si oui, les ressources sont retirées de l'inventaire, puis l'objet fabriqué est ajouté en quantité spécifiée.
3. Si non, aucun objet n'est fabriqué, et les ressources partiellement consommées (s'il y en a) sont rendues immédiatement au joueur.

Cette gestion garantit une sécurité maximale : le joueur ne peut jamais perdre d'objets sans contrepartie. Elle rend le système à la fois transparent, fiable, et permissif pour l'expérimentation.

10.4 Ajout et gestion des recettes

Les recettes sont organisées dans des objets indépendants, ce qui permet de les modifier ou d'en ajouter facilement, sans impacter la structure globale du système. Chaque recette contient une référence au produit final et aux ressources nécessaires. Cette modularité offre une grande souplesse pour l'évolution du jeu : les développeurs peuvent introduire de nouvelles technologies, recettes avancées, ou variantes saisonnières sans avoir à modifier le cœur du système.

10.5 Cas d'usage et intérêt ludique

Le crafting n'est pas uniquement une mécanique fonctionnelle : il s'inscrit pleinement dans l'expérience de jeu. Il sert à :

— transformer les ressources brutes extraites dans le monde en outils utiles ;
— produire des objets nécessaires à l'automatisation (ex : foreuses, convoyeurs) ;
— débloquer des éléments rares ou avancés, inaccessibles autrement ;
— initier une spécialisation progressive du joueur selon ses priorités ;
— rythmer la progression en introduisant des seuils de fabrication.

En ce sens, le crafting agit comme un moteur de progression, mais aussi comme un pont entre les autres systèmes du jeu : inventaire, extraction, défense, etc.

10.6 Limites et axes d'amélioration

Le système de craft actuel, bien que fonctionnel, est encore basique et pourrait être étendu selon plusieurs axes :

10.6.1 Affichage des recettes manquantes

À ce stade, seules les recettes disponibles sont visibles. Il serait utile d'afficher aussi les recettes bloquées, avec une mise en forme différente (grisées), pour que le joueur sache ce qu'il peut débloquer à terme et prépare ses collectes en conséquence.

10.6.2 Retour visuel avancé

Outre le simple message d'erreur, des animations ou effets sonores pourraient accompagner la réussite ou l'échec d'un craft, renforçant la satisfaction ou la compréhension du joueur. Des indicateurs de progression ou de surbrillance pourraient améliorer l'ergonomie.

10.6.3 Crafting en masse

Aujourd'hui, chaque fabrication s'effectue individuellement. Un bouton "Fabriquer en x5" ou "Fabriquer autant que possible" permettrait aux joueurs avancés d'automatiser certaines tâches sans devoir cliquer plusieurs fois.

10.6.4 Intégration à l'automatisation

À l'avenir, certaines machines pourraient être capables de fabriquer automatiquement des objets si elles reçoivent les bonnes ressources. Le joueur pourrait ainsi déléguer la fabrication de composants simples, tout en se concentrant sur les choix stratégiques liés aux ressources rares.

10.6.5 Niveaux de complexité des recettes

Pour enrichir le système, des recettes avancées pourraient nécessiter non seulement plus de ressources, mais aussi des objets intermédiaires (ex : alliage $\rightarrow$ nécessite fer raffiné + charbon transformé). Cela introduirait un système hiérarchique de fabrication, proche de celui rencontré dans des jeux comme *Factorio* ou *Satisfactory*.

10.6.6 Verrouillage et progression technologique

Certaines recettes pourraient être bloquées au départ et se débloquer via des recherches, des quêtes ou la construction de bâtiments spécifiques. Cela permettrait d'établir une courbe de progression plus maîtrisée, tout en valorisant les efforts du joueur.

En résumé, le système de crafting de *Uzin* est à la fois simple d'utilisation et robuste. Il répond aux besoins fondamentaux de transformation des ressources, tout en étant suffisamment modulaire pour accompagner les ambitions futures du projet, que ce soit en termes de complexité, d'immersion ou de stratégie.

11 Intelligence artificielle des ennemis

L'intelligence artificielle joue un rôle fondamental dans l'univers de *Uzin*, en introduisant une couche d'imprévisibilité, de tension et de challenge. Elle est responsable de la vie comportementale des ennemis, de leur agressivité, de leur réaction à l'environnement, ainsi que de la manière dont ils interagissent avec les joueurs. Le système mis en place repose sur une logique simple en apparence, mais suffisamment flexible pour s'adapter à divers scénarios de jeu et poser les bases d'une IA plus évoluée à l'avenir.

11.1 Comportements de base : patrouille et agression

Les ennemis du jeu, à ce stade de développement, adoptent un cycle comportemental divisé en deux grandes phases : la phase d'errance et la phase d'attaque. Durant leur phase d'errance, les ennemis se déplacent dans un rayon défini autour de leur position d'origine. Leurs déplacements ne suivent pas un parcours fixe : ils se dirigent aléatoirement vers différents points à l'intérieur de cette zone, ce qui rend leur trajectoire moins prévisible.

Cette errance est entrecoupée de phases d'observation statique, ou d'un léger mouvement vertical simulant une lévitation ou un vol stationnaire. Cela donne vie aux créatures et les distingue visuellement des entités inactives ou scriptées. Lorsque la phase d'attaque est déclenchée, l'ennemi abandonne son comportement aléatoire et se dirige de manière directe vers le joueur.

La transition vers l'état d'attaque peut être déclenchée soit par le temps (après une durée aléatoire prédéfinie), soit par un stimulus externe, comme des dégâts subis. Cette alternance constante entre calme apparent et agressivité brutale crée une dynamique de jeu rythmée et stimulante.

11.2 Réactivité et perception du joueur

L'une des caractéristiques intéressantes de cette intelligence artificielle est sa capacité à réagir à l'environnement. Les ennemis ne se contentent pas d'attendre passivement : s'ils sont attaqués ou blessés, ils réagissent immédiatement par une contre-attaque. Ce comportement crée un lien de causalité direct entre les actions du joueur et les réactions du monde, renforçant l'immersion.

Les ennemis peuvent également repérer le joueur à distance lorsqu'ils entrent dans leur cycle d'attaque. Ils orientent alors leur trajectoire vers lui, réduisant la distance progressivement. Ce système donne l'impression que les ennemis ont une forme de perception du monde autour d'eux, même si elle est encore très basique à ce stade.

11.3 Gestion de l'engagement et du désengagement

Après avoir infligé des dégâts à un joueur, l'ennemi retourne automatiquement dans un état d'errance. Ce mécanisme évite que le joueur ne soit pris dans une boucle d'attaques consécutives sans possibilité de réagir. Cela offre un moment de répit et crée un rythme de combat plus juste et équilibré.

De plus, chaque cycle d'agression est isolé : l'ennemi engage un seul joueur à la fois, puis se désengage temporairement. Cela ouvre la porte à des mécaniques futures, comme des groupes d'ennemis attaquant tour à tour ou de manière coordonnée.

11.4 Effets visuels et animation adaptative

Les ennemis disposent d'animations contextuelles qui évoluent selon leur état. En errance, ils présentent un comportement visuel calme et continu. Lorsqu'ils entrent en phase d'attaque, leur animation devient plus rapide, leurs trajectoires plus agressives, ce qui permet au joueur d'anticiper le danger simplement en observant leur comportement.

Un effet d'oscillation verticale vient également renforcer leur présence dans l'espace. Ce mouvement sinusoïdal donne une impression de flottement constant, en accord avec leur nature volante, et permet de mieux les différencier des autres entités statiques ou ancrées au sol.

11.5 Système de génération dynamique

Les ennemis ne sont pas placés manuellement dans le monde du jeu : ils apparaissent de manière dynamique selon la progression du joueur. Lorsqu'un joueur est détecté dans une zone active, un processus d'apparition est enclenché. Ce système repose sur des intervalles de temps aléatoires et une distance minimale au joueur pour garantir un équilibre entre surprise et jouabilité.

Les types d'ennemis générés sont choisis aléatoirement parmi une liste prédéfinie, chaque type ayant un poids spécifique influençant sa probabilité d'apparition. Cela permet d'introduire progressivement des ennemis plus puissants ou plus rares, sans bouleverser l'équilibre global du jeu.

11.6 Impact sur le rythme de jeu

Ce système d'IA, bien que simple dans sa logique, a un impact direct sur l'expérience du joueur. Il empêche l'installation d'une routine trop confortable, pousse à l'attention permanente, et oblige à sécuriser les zones d'extraction ou

de construction. Les attaques, bien que brèves, interviennent à des moments imprévisibles, ce qui alimente un sentiment de danger latent.

Les joueurs doivent ainsi organiser leurs déplacements et leurs interactions en fonction de la probabilité d'apparition de créatures hostiles, ce qui renforce le rôle stratégique de certaines décisions : faut-il continuer à miner ou se replier ? Est-ce que la zone est encore sûre ?

11.7 Gestion de la vie et de la mort des ennemis

Chaque ennemi possède une jauge de vie intégrée et synchronisée en temps réel entre tous les joueurs. Lorsqu'il subit des dégâts, cette jauge est réduite, et une fois vide, le cycle de disparition de l'ennemi s'enclenche. Pour renforcer l'impact de ces moments, une animation de disparition ainsi qu'un effet de gel sont appliqués avant le retrait effectif de l'ennemi de la scène.

Ce processus garantit une lisibilité parfaite de la situation : un ennemi est visuellement identifiable comme étant "vaincu", ce qui réduit les ambiguïtés et améliore l'ergonomie visuelle du jeu.

11.8 Robustesse en multijoueur

Toutes les décisions critiques liées aux ennemis (apparition, déplacement, attaque, dégâts, disparition) sont prises par l'hôte de la partie. Ce choix garantit une cohérence totale entre les clients : tous les joueurs voient les ennemis aux mêmes endroits, dans les mêmes états, et au même moment.

Des mécanismes de mise à jour fréquente et de détection de collision centralisée permettent de s'assurer que même dans des conditions de latence, les interactions restent fiables. Ce modèle pourra évoluer vers des comportements encore plus synchronisés à mesure que les ennemis deviendront plus complexes.

11.9 Perspectives d'évolution de l'intelligence artificielle

Le système actuel d'intelligence artificielle dans *Uzin* repose sur des comportements simples mais efficaces : patrouille aléatoire, déclenchement de l'attaque selon des critères définis, et réactions aux dégâts. Il forme une base stable pour étendre les capacités des ennemis et diversifier les situations de jeu. Voici les principales évolutions envisagées.

11.9.1 IA à distance

Jusqu'à présent, les ennemis doivent entrer en collision avec le joueur pour infliger des dégâts. Une évolution naturelle consisterait à introduire des

ennemis capables d'attaquer à distance, à l'aide de projectiles ou d'effets de zone. Cela ajouterait une nouvelle couche de danger, forçant le joueur à se déplacer de manière plus stratégique, à couvrir ses machines, ou à utiliser le décor comme protection.

Les attaques à distance pourraient varier selon le type d'ennemi : tir lent mais puissant, tir rapide en rafale, projection de poison ou d'électricité. Ces ennemis pourraient également viser les structures du joueur, rendant la défense plus complexe et incitant à prévoir des systèmes d'alarme ou de réparation automatique.

11.9.2 Perception avancée

Actuellement, les ennemis repèrent le joueur de manière déterministe (détection automatique ou réaction aux dégâts). Un système de perception plus réaliste, basé sur le bruit, la lumière ou le champ de vision, permettrait d'instaurer de véritables mécaniques d'infiltration et de tension.

Par exemple, le joueur pourrait se cacher dans l'ombre ou limiter le bruit généré par ses machines pour retarder l'arrivée des ennemis. À l'inverse, des actions comme courir, utiliser un projecteur ou activer une foreuse bruyante augmenteraient le risque d'être détecté. Ce système renforcerait l'immersion et donnerait du sens à des choix discrets jusque-là purement visuels.

11.9.3 Ennemis à comportement défensif

Tous les ennemis n'ont pas vocation à attaquer frontalement. Certains pourraient adopter un comportement défensif : fuir lorsqu'ils sont blessés, se cacher ou se regrouper pour revenir plus tard. D'autres pourraient se camoufler dans le décor, imitant des éléments naturels jusqu'à ce qu'ils soient activés par la proximité du joueur.

De tels comportements rendraient les rencontres plus variées et moins prévisibles. Le joueur devrait observer, anticiper, et parfois choisir entre attaquer immédiatement ou suivre un ennemi pour localiser un groupe caché. Cela introduit des dilemmes stratégiques et donne de la profondeur au système d'IA.

11.9.4 Patrouilles prédéfinies

À l'opposé des patrouilles aléatoires actuelles, certains ennemis pourraient suivre des trajectoires fixes ou semi-aléatoires autour de points d'intérêt : sources de ressources, zones d'activité du joueur, ou anciens nids de créatures.

Ce comportement permettrait d'instaurer des zones de danger identifiées, encourageant la planification. Le joueur pourrait observer les routines ennemies,

attendre le bon moment pour passer, ou choisir de saboter un point clé pour rompre la patrouille. L'univers gagnerait ainsi en cohérence et en lisibilité, avec des mécaniques proches de jeux d'infiltration ou de stratégie en temps réel.

11.9.5 Hiérarchisation des ennemis

Introduire une structure hiérarchique au sein des ennemis ouvrirait la voie à des interactions collectives plus poussées. Par exemple, certains ennemis pourraient occuper un rôle de "chef" : ils coordonneraient les déplacements d'un groupe, déclencheraient une attaque groupée ou décideraient de battre en retraite.

Ces chefs seraient plus rares, mais plus intelligents et plus résistants. Leur élimination pourrait désorganiser le groupe ou interrompre une offensive. Cela offrirait des objectifs tactiques intermédiaires au joueur : plutôt que d'éliminer tous les ennemis, il pourrait viser le leader pour désamorcer la menace.

11.9.6 Écosystème intelligent

À long terme, un système d'écosystème dynamique pourrait faire évoluer la population ennemie en fonction du comportement du joueur. Des créatures pourraient migrer, se reproduire, établir des nids, ou au contraire disparaître si leur habitat est détruit.

Ce type d'IA environnementale ferait du monde de *Uzin* un système vivant, capable de réagir organiquement à l'activité humaine. Les joueurs les plus agressifs pourraient détruire les espèces locales… au risque de voir apparaître des créatures mutées. À l'inverse, une approche plus discrète ou sélective pourrait maintenir un équilibre et éviter certaines confrontations. Cela introduirait un lien écologique entre gestion des ressources et biodiversité.

11.9.7 Système d'alerte et de mobilisation

Enfin, une dernière évolution importante consisterait à doter certains ennemis de la capacité à alerter leurs congénères. Lorsqu'un individu repère le joueur, il pourrait fuir pour appeler du renfort, déclenchant une vague d'attaque. Cette dynamique transformerait les combats isolés en séquences de survie ou de fuite, rendant chaque engagement potentiellement risqué.

Ainsi, même si l'IA actuelle repose sur un fonctionnement simple et déterministe, elle a été pensée pour évoluer vers des comportements complexes et interactifs. Elle permet d'ores et déjà de proposer une expérience engageante, en constante tension, et renforce la profondeur de l'univers de *Uzin*.

12 Système de placement des usines

La capacité du joueur à placer des machines est un élément fondamental dans *Uzin*, car elle marque la transition entre l'extraction manuelle de ressources et l'automatisation. Le système de placement permet aux joueurs de sélectionner, positionner et instancier diverses machines sur la carte, tout en assurant une cohérence visuelle, une compatibilité avec le multijoueur et une bonne expérience utilisateur.

12.1 Fonctionnement général

Le système de placement repose sur un processus en plusieurs étapes :

1. Le joueur accède à un menu de sélection en appuyant sur une touche dédiée.
2. Il choisit une machine parmi une liste proposée dans un menu déroulant.
3. Une fois la machine sélectionnée, il valide son choix et la machine est instanciée à la position du curseur.
4. La machine devient alors une entité du monde, prête à fonctionner et interagir avec l'environnement.

Le menu de sélection interrompt momentanément le temps dans le jeu pour laisser le joueur se concentrer sur son choix. Cela permet de ne pas perturber l'action en cours (notamment en cas d'ennemis) et de sécuriser l'interface utilisateur.

12.2 Interface de sélection et ergonomie

L'interface de placement est conçue pour être simple et rapide à utiliser. Elle se présente sous la forme d'un panneau contenant une liste déroulante de machines disponibles. Chaque machine est identifiée par son nom, et peut à terme être accompagnée d'un aperçu visuel ou d'une description de ses effets.

Le menu se déclenche via une touche (par défaut, `P`), ce qui affiche le panneau de sélection. Une fois la machine choisie, un bouton permet de confirmer le placement. Si le joueur change d'avis, il peut annuler la sélection à tout moment.

Ce système rend le placement accessible sans interrompre complètement l'expérience de jeu. L'ergonomie est pensée pour que la sélection se fasse en quelques secondes, tout en laissant au joueur le temps d'observer l'environnement et de positionner ses machines avec précision.

12.3 Calcul de la position de placement

Le joueur place sa machine à l'endroit où il a cliqué au moment de l'ouverture du menu. Le système convertit la position de la souris en coordonnées du monde, puis l'aligne sur une grille invisible afin de garantir une organisation logique et esthétique des installations.

Les machines sont positionnées au centre d'une case virtuelle de la carte, en arrondissant les coordonnées au demi-élément près. Cette méthode garantit que toutes les machines sont alignées de manière cohérente, ce qui facilite à terme la connexion entre les éléments (par exemple : convoyeurs, tuyaux, câbles).

12.4 Multijoueur et instanciation en réseau

Le système de placement est compatible avec le mode multijoueur. Chaque machine placée est instantanément synchronisée avec tous les joueurs grâce à une centralisation sur le serveur hôte. Cela garantit que tous les participants voient les machines au même endroit, avec les mêmes propriétés.

Le placement est validé côté serveur avant que l'objet n'apparaisse. Cela évite les tricheries, les conflits d'état, et garantit une instanciation propre, même en cas de latence. Chaque machine reçoit un identifiant unique et peut ensuite être utilisée, alimentée ou connectée comme toute autre entité du jeu.

12.5 Rôle stratégique du placement

Le placement des machines ne se limite pas à une simple action mécanique : il représente un choix stratégique majeur. En fonction de la proximité des ressources, de la topographie du terrain et des menaces environnantes, le joueur devra adapter la disposition de ses structures.

- Placer trop de machines proches les unes des autres peut bloquer les déplacements ou rendre la maintenance difficile.
- Une disposition trop dispersée risque de ralentir la production ou d'augmenter les délais d'acheminement.
- Il faut également prendre en compte les zones à risque (proches des nids ennemis ou des zones toxiques).

Ce système encourage donc l'anticipation, la planification et la capacité d'adaptation du joueur.

12.6 Problèmes rencontrés

Le développement du système de placement des usines dans *Uzin* a nécessité plusieurs itérations, au cours desquelles divers problèmes techniques et conceptuels ont été identifiés. Cette section présente les principaux obstacles rencontrés et les ajustements qui ont été apportés pour les contourner ou les atténuer.

12.6.1 Coordonnées de placement imprécises

L'un des premiers problèmes rencontrés a concerné la conversion des coordonnées de la souris en position de placement dans le monde. Les machines étaient parfois instanciées à des positions légèrement décalées, ce qui entraînait des superpositions avec d'autres objets ou un alignement incorrect sur la grille de jeu. Il a fallu implémenter un système d'alignement correct sur la `Tilemap`.

12.6.2 Conflits d'entrée dans le menu de sélection

Le menu de sélection des machines s'ouvrait initialement sur simple pression d'une touche, même lorsque d'autres interfaces (comme l'inventaire ou le menu de craft) étaient déjà ouvertes. Cela provoquait des superpositions d'interfaces ou des bugs de navigation où plusieurs menus restaient actifs en même temps. Il a été nécessaire d'introduire une hiérarchie de priorités entre les interfaces pour éviter les conflits et garantir une expérience utilisateur fluide.

12.6.3 Désynchronisation en multijoueur

Dans les premières versions du système, il arrivait que la machine soit placée localement chez un joueur mais non synchronisée correctement avec les autres clients. Ce problème de désynchronisation provenait d'un appel au placement côté client sans validation côté serveur. La solution a consisté à centraliser entièrement l'instanciation sur le serveur, tout en relayant les intentions de placement depuis les clients via des appels contrôlés.

En définitive, le système de placement des usines constitue un pilier fondamental du gameplay de *Uzin*, en facilitant la transition vers l'automatisation tout en laissant au joueur une grande liberté stratégique. Son implémentation a soulevé plusieurs défis techniques, notamment liés à la précision des positions, à la gestion des interfaces et à la synchronisation multijoueur. Ces difficultés ont été progressivement surmontées, renforçant la robustesse du système.

13 Système de quêtes

Le système de quêtes constitue l'un des vecteurs principaux de progression et de découverte dans *Uzin*. Il guide le joueur à travers les différentes mécaniques du jeu, tout en structurant son apprentissage des fonctionnalités essentielles : déplacement, minage, automatisation, transformation des ressources. Plutôt que de proposer un tutoriel passif, le jeu intègre progressivement des objectifs à remplir, de manière interactive, dynamique et contextualisée.

13.1 Fonctionnement général

Le système de quêtes repose sur une séquence ordonnée de missions, que le joueur doit accomplir dans un ordre prédéfini. Chaque quête est constituée des éléments suivants :

— Un titre court et accrocheur, qui donne une indication immédiate de l'objectif à atteindre.
— Une description plus détaillée, précisant l'action à effectuer et les touches ou mécaniques impliquées.
— Une quantité à remplir (exemple : miner 25 unités de charbon), accompagnée d'une barre de progression.
— Un délai entre deux quêtes, permettant de rythmer le jeu et de laisser le joueur s'approprier la mécanique.

Lorsque le joueur accomplit les conditions requises pour une quête, un délai court s'enclenche, au terme duquel la quête suivante est automatiquement lancée.

13.2 Interface et intégration utilisateur

Les quêtes sont affichées via un panneau latéral contenant :
— le titre de la quête en cours ;
— sa description détaillée ;
— une jauge de progression visuelle ;

Ce panneau n'est visible que pour le joueur concerné. En multijoueur, chaque joueur peut ainsi progresser indépendamment dans ses propres objectifs, ce qui permet de conserver une autonomie d'apprentissage même en coopération. Cette approche évite les interruptions forcées et respecte le rythme individuel de chaque participant.

13.3 Structure et enchaînement des quêtes

Le système repose sur une liste ordonnée de quêtes prédéfinies, initialisées au lancement du jeu. À l'heure actuelle, cinq quêtes composent la séquence

introductive :

1. Découverte de l'environnement et déplacement.
2. Extraction de charbon manuellement.
3. Placement d'une première foreuse.
4. Utilisation de la foreuse pour miner du fer.
5. Cuisson du fer dans un four.

Ces étapes permettent de découvrir progressivement les différentes couches du gameplay : de la collecte manuelle à l'automatisation de la production. Chaque nouvelle étape introduit une mécanique inédite tout en consolidant les précédentes.

13.4 Suivi de la progression et validation des objectifs

Chaque quête possède une variable de suivi, qui enregistre l'avancement du joueur (ex. : nombre de ressources collectées). Dès que la valeur atteint l'objectif requis, la quête est considérée comme accomplie. Un court délai s'enclenche alors avant l'activation de la quête suivante, permettant de célébrer brièvement la réussite et de donner un rythme à la progression.

Ce délai joue également un rôle technique : il garantit que les données sont mises à jour correctement avant de passer à l'étape suivante. Il assure la stabilité du système tout en renforçant la clarté de l'interface.

13.5 Robustesse et comportement en multijoueur

Le système est conçu pour fonctionner indépendamment sur chaque client. Les objectifs sont suivis localement, et les textes affichés dans l'interface sont personnalisés. Cela permet à plusieurs joueurs de progresser en parallèle, à leur propre rythme, sans se gêner.

Cette indépendance garantit également une meilleure tolérance aux erreurs : si un joueur tarde à accomplir une tâche ou se concentre sur une autre activité, les autres peuvent continuer leur propre exploration. À terme, des quêtes coopératives pourraient également être envisagées, en combinant les progrès des différents joueurs.

Chaque quête agit comme une étape sur un chemin de montée en compétence. Le joueur n'est jamais passif : il agit, observe les conséquences, apprend, et débloque de nouveaux outils. C'est cette boucle d'interaction et de progression qui rend le système de quêtes si efficace dans un jeu d'exploration et d'automatisation comme *Uzin*.

14 Retour d'expérience - Tristan Druart

14.1 Implication dans les différentes phases du projet

Ce projet m'a permis d'explorer de nombreux aspects du développement d'un jeu vidéo, de la génération procédurale à l'implémentation du multijoueur. J'ai commencé par travailler sur la génération de la carte, en m'appuyant sur le bruit de Perlin.

Par la suite, j'ai participé à l'intégration du multijoueur, une phase complexe marquée par de nombreux défis, notamment en ce qui concerne la synchronisation entre les clients. Il fallait s'assurer que la carte, les objets, les inventaires, les ennemis ou encore les mouvements soient tous identiques pour chaque joueur, ce qui a nécessité beaucoup de rigueur et de tests.

J'ai également implémenté un système de tir, permettant au joueur de viser avec la souris et de se défendre en temps réel contre les ennemis. D'autres tâches annexes m'ont été confiées, comme la création du menu principal et du menu des options, ainsi que l'aide ponctuelle apportée à mes camarades sur leurs parties respectives du projet.

14.2 Ce que j'ai apprécié

Ce que j'ai particulièrement aimé dans ce projet, c'est la liberté créative dont nous avons bénéficié. Nous avions l'autonomie de concevoir un jeu de A à Z, sans contraintes techniques ou directives strictes extérieures. Chaque décision était prise en groupe lors des réunions que nous organisions régulièrement. Cette liberté m'a permis de m'impliquer pleinement dans les domaines qui m'intéressaient.

Un autre moment fort a été la satisfaction ressentie lorsque des systèmes complexes finissaient par fonctionner après plusieurs heures de travail. Voir une fonctionnalité enfin aboutie, stable et fonctionnelle après une série de tests et de corrections est une source de motivation énorme.

14.3 Difficultés rencontrées

Le principal défi de ce projet a été la synchronisation en multijoueur. De nombreuses fonctionnalités — la génération de la carte, la gestion des inventaires, les animations, les déplacements des ennemis, la rotation des armes — nécessitaient des logiques spécifiques pour fonctionner correctement sur tous les clients. Chaque cas présentait un petit détail technique unique, ce qui obligeait à une adaptation constante. Cela a demandé beaucoup de patience et de rigueur.

15 Retour d'expérience - Martin Lemée

15.1 Responsabilités et contributions principales

Je me suis principalement investi dans la mise en place du système d'inventaire. Chaque joueur possède un inventaire unique, permettant une gestion individuelle des ressources et objets collectés en cours de partie.

J'ai également conçu les interfaces utilisateurs associées aux différentes machines. Chaque interface devait être claire, lisible, mais aussi adaptée à la fonctionnalité spécifique de la machine.

En parallèle, j'ai développé un système de fabrication manuelle d'objets, basé sur des recettes définies. Le joueur peut, via cette interface, combiner plusieurs ressources pour obtenir des objets plus complexes, utilisés par la suite pour d'autres interactions.

15.2 Ce que j'ai apprécié

Ce projet m'a offert l'opportunité de découvrir en profondeur Unity, au-delà de son utilisation basique. J'ai pu expérimenter la conception d'interfaces interactives, la gestion des données persistantes, et la logique réseau. Cela m'a permis de consolider mes acquis en C#, tout en les confrontant à des cas concrets.

Sur le plan humain, j'ai également appris à travailler efficacement en équipe. J'ai dû expliquer mes choix techniques, proposer des solutions, m'adapter à des contraintes communes, et contribuer à des décisions collectives. Le fait d'avoir une vision partagée et de devoir faire converger des systèmes très différents vers une expérience unifiée a été extrêmement formateur.

J'ai particulièrement apprécié la richesse des interactions entre les mécaniques que j'ai développées : voir une machine produire un objet qui est ensuite stocké, transformé, puis réutilisé dans une autre interface, renforce le sentiment d'avoir contribué à un système global.

15.3 Difficultés rencontrées

L'un des plus gros défis techniques a été la synchronisation multijoueur des interfaces de machines. Chaque joueur devait voir en temps réel les modifications apportées à une machine, sans provoquer d'incohérences ou de conflits entre les clients. Cela a nécessité une gestion précise des échanges de données, ainsi que la mise en place de vérifications supplémentaires pour éviter les décalages entre états.

16 Retour sur expérience - Cyril Dejouhanet

16.1 Contributions personnelles au projet

J'ai été désigné comme directeur artistique, ce qui a été ma tâche principale. Une grande partie des graphismes et de l'interface a été réalisée en pixel art, à la main, par mes soins. J'ai également conçu les animations présentes dans le jeu, à l'aide des outils Unity. Seulement, les animations n'étaient pas forcément notre priorité principale, tout comme les graphismes. La partie technique restant considérablement importante, j'ai également participé à toutes les autres parties de ce projet, étant, par exemple, le premier à expérimenter avec le multijoueur.

16.2 Ce que j'ai apprécié

ça a été une expérience intéressante, un projet en groupe sur un an a demandé une bonne cohésion, tout en demandant à chacun de produire le travail qui lui était demandé. Je pense que nous avons réussi à garder une bonne ambiance sur toute la durée de ce travail, ce qui est primordial pour un résultat satisfaisant.

En effet, étant en groupe avec trois amis, la communication entre nous s'est bien passée et a été efficace, facilitant le reste du travail.

Sur ce que j'ai pu faire pendant le projet, le design et les animations ont probablement été les parties que j'ai pu trouver les plus intéressantes, se rapprochant le plus d'une de mes passions : le dessin. J'ai particulièrement apprécié la possibilité de pouvoir imaginer et créer un univers, afin d'en faire un jeu.

16.3 Ce que j'ai moins apprécie

Au cours de ce projet, j'ai pu rencontrer plusieurs difficultés.

Ayant passé trop de temps sur la partie artistique au début du projet, je me suis vu me faire légèrement dépasser par le niveau technique du code Unity, principalement sur la partie multijoueur.

J'ai pu recevoir des informations contradictoires au niveau artistique au cours des soutenances, apprenant que les graphismes "n'avaient pas d'importance" pendant la première soutenance, puis, venant de la même personne, que le design de l'écran d'accueil initial "n'était pas beau" pendant la seconde soutenance.

17 Retour d'expérience - Maxan Fournier

17.1 Implication personnelle

Durant ce projet, je me suis concentré principalement sur trois grands axes : l'implémentation du multijoueur, le système de placement des usines, et le développement de l'intelligence artificielle.

L'aspect multijoueur a constitué l'une des parties les plus techniques du jeu. J'ai travaillé à faire en sorte que tous les éléments essentiels — les joueurs, la carte, les objets, les machines, les ennemis — soient synchronisés correctement entre les clients.

J'ai également conçu le système de placement des usines, permettant aux joueurs de sélectionner des machines via une interface, puis de les placer de manière fluide dans le monde de jeu.

Enfin, j'ai développé l'intelligence artificielle des ennemis, en particulier des créatures volantes.

17.2 Ce que j'ai apprécié

Ce projet m'a permis de travailler au sein d'une équipe où chaque membre avait des responsabilités claires. Cette répartition a favorisé une bonne autonomie tout en créant un environnement propice à l'entraide. J'ai apprécié cette dynamique de travail collaboratif, où chacun apportait sa pierre à l'édifice.

Le développement de l'intelligence artificielle a été une expérience particulièrement plaisante. Concevoir des comportements cohérents, les tester, et voir les ennemis réagir en jeu m'a permis d'allier logique, créativité et gameplay.

J'ai aussi particulièrement apprécié travailler sur l'optimisation du jeu, notamment lorsqu'il a fallu identifier et corriger les ralentissements causés par la génération des chunks de la carte. Travailler sur ce type de problème technique, où l'on mesure directement l'impact de ses modifications sur les performances, a été très motivant.

17.3 Difficultés rencontrées

Malgré les aspects positifs, certains points du projet ont été plus difficiles à gérer. Le premier problème notable fut la gestion des conflits de fusion (merge conflicts) sur Git. Lorsque plusieurs membres de l'équipe travaillaient simultanément sur des fichiers proches, les conflits étaient fréquents, longs à résoudre, et source de frustration. Cela a parfois freiné notre productivité.

18 Conclusion

Le projet *Uzin* nous a permis de concevoir, développer et finaliser un jeu vidéo multijoueur complet, combinant exploration, automatisation, génération procédurale, intelligence artificielle et système de progression. En partant d'une feuille blanche, nous avons construit un univers cohérent, mis en place des mécaniques complexes comme l'inventaire synchronisé, le système de craft, le placement dynamique des usines et la génération de la carte en chunks, tout en garantissant une expérience fluide et stable, y compris en réseau.

Sur le plan technique, chacun des membres de l'équipe a pu se confronter à des problématiques concrètes et variées : conception de systèmes modulaires, optimisation des performances, communication client-serveur, logique d'interface, gestion des ressources partagées… La diversité des tâches accomplies témoigne de l'ambition du projet et du niveau d'engagement de chaque participant. Les obstacles rencontrés, qu'ils soient liés à la synchronisation multijoueur, aux conflits de version ou à des choix techniques complexes, ont été autant d'opportunités d'apprentissage.

Au-delà des compétences techniques, ce projet nous a aussi beaucoup appris sur le plan humain. Travailler en équipe sur un jeu de cette ampleur exige une communication constante, une bonne répartition des responsabilités et une capacité à faire confiance aux autres. Nous avons appris à écouter, à expliquer, à faire des compromis et à ajuster nos méthodes de travail selon les besoins du groupe. Cette collaboration nous a permis non seulement de mener à bien le développement, mais aussi de créer une dynamique de travail motivante pour que chacun puisse contribuer au projet en fonction de ses capacités.

Pour conclure, *Uzin* est bien plus qu'un projet scolaire. C'est une expérience complète mêlant technique, créativité et travail d'équipe, dont nous ressortons avec des compétences renforcées, une meilleure compréhension du développement de jeux, et une véritable fierté du chemin parcouru ensemble.