Rapport de soutenance n°1

Uzin

Bart Entreprises
Janvier 2025

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

Table des matières

1 Introduction

1.1 À propos de notre projet

Notre projet consiste à développer *Uzin*, un jeu vidéo où le joueur incarne un personnage se trouvant sur une planète inconnue. Le gameplay débute par la récolte manuelle de ressources dispersées sur la carte. Progressivement, le joueur pourra construire des infrastructures pour automatiser ces processus, tout en gérant une production énergétique évolutive. Les ressources, initialement simples, se complexifient avec le temps, exigeant la conception de machines avancées et une gestion stratégique accrue. Ce système progressif offre un mélange captivant de collecte, de construction et d'optimisation.

Cependant, la planète ne reste pas passive : l'environnement réagit à l'activité du joueur, mobilisant une faune hostile et une flore défensive. Inspiré du genre *tower defense*, le joueur devra construire des structures pour protéger ses infrastructures tout en explorant et en automatisant davantage. Grâce à cette combinaison de gestion, d'automatisation et de survie, *Uzin* promet une expérience stratégique immersive où chaque décision façonne le déroulement du jeu.

1.2 Le rôle de ce rapport de soutenance

Ce rapport de soutenance présente de manière détaillée l'ensemble des travaux réalisés depuis la validation du cahier des charges. Il retrace les avancées concrètes, les obstacles rencontrés et les solutions mises en œuvre pour respecter nos objectifs. Chaque section met en lumière le rôle de chaque membre de l'équipe, en décrivant précisément leurs contributions respectives, tout en analysant les éventuels écarts par rapport au planning initial. En somme, il s'agit d'une base essentielle pour évaluer le progrès de notre projet et ajuster nos priorités si nécessaire.

En plus du bilan des réalisations, notre rapport anticipe les étapes à venir en définissant clairement les tâches restantes et les responsabilités associées pour chaque membre de l'équipe. Il offre une vision structurée des objectifs à atteindre d'ici la prochaine soutenance, en tenant compte des leçons apprises et des ajustements requis pour maintenir une progression optimale. Ce document permet un suivi transparent et rigoureux de l'avancement du projet.

2 Découpage et avancement du projet

2.1 Répartition des tâches

Pour rappel, 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, de manière à ce que tout le monde puisse collaborer.

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

2.2 Avancement du projet

Tâches	**Soutenance 1**
Intelligence artificielle	0%
Histoire et storytelling	70%
Design	10%
Game design	50%
Réseau	0%
Site web	0%

TABLE 1 – Le planning initial, présent dans le cahier des charges fonctionnel

Nous avions initialement planifié d'atteindre le niveau d'avancement ci-dessus. L'équipe d'*Uzin* a non seulement respecté le planning, mais a également dépassé les attentes en progressant davantage que prévu sur certaines tâches. Pour chacune d'entre elles, ce qui a été réalisé jusqu'à présent par chacun des membres de l'équipe sera détaillé dans les pages suivantes.

Ainsi :
— Le design est complété à **30%**
— Le site web est complété à **75%**

3 Histoire du jeu - Martin Lemée

3.1 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.

3.2 Le contexte narratif : un univers dystopique

3.2.1 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.

3.2.2 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.

3.3 Thèmes centraux et ambiance narrative

3.3.1 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.

3.3.2 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 ?

3.4 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.

4 Système d'inventaire - Martin Lemée

Le système d'inventaire est un élément crucial dans le développement de *Uzin*. Dès le départ, nous avons compris que cette fonctionnalité ne serait pas seulement un outil de gestion pour les joueurs, mais également un moyen d'ajouter du réalisme à l'expérience de jeu. Ce système devait être suffisamment robuste pour répondre aux exigences du gameplay, tout en restant intuitif et accessible pour les joueurs. Ce développement a été pour moi un véritable défi technique.

4.1 Les premières étapes : un dictionnaire simple

Dans un premier temps, j'ai choisi une approche basée sur un dictionnaire pour structurer l'inventaire. Cette méthode m'a semblé adaptée en raison de sa simplicité et de sa rapidité d'implémentation. Chaque clé représentait un objet spécifique (par exemple, "fer raffiné"), et chaque valeur indiquait la quantité de cet objet détenue par le joueur.

Cette solution avait l'avantage d'être facile à comprendre et à mettre en œuvre, surtout pour un projet en phase initiale. Cependant, j'ai rapidement réalisé ses limites. Par exemple :

— **Manque de flexibilité** : Il était difficile d'ajouter des fonctionnalités avancées, comme des limites de stockage par emplacement ou des objets ayant des propriétés uniques.
— **Synchronisation complexe** : Dans un contexte multijoueur, maintenir une cohérence entre l'état de l'inventaire et les événements du jeu s'est avéré compliqué.
— **Incompatibilité avec les autres systèmes** : L'interaction entre l'inventaire et d'autres mécanismes, comme l'automatisation ou la collecte de ressources, n'était pas fluide.

Ces défis m'ont amené à reconsidérer cette première approche et à explorer des solutions plus adaptées.

4.2 Transition vers une classe dédiée

Pour pallier ces limitations, j'ai décidé de concevoir le système d'inventaire sous la forme d'une classe dédiée. Cette nouvelle structure permettait de mieux encapsuler les données et les fonctionnalités associées à l'inventaire, tout en rendant le code plus facile à maintenir. Chaque joueur disposait désormais

de son propre objet inventaire, ce qui a grandement simplifié la gestion dans un environnement multijoueur.

— **Gestion indépendante** : Chaque joueur a son propre inventaire, éliminant ainsi les interférences potentielles entre les inventaires.
— **Identifiants uniques** : Les objets sont désormais représentés par des identifiants uniques, ce qui simplifie leur gestion et réduit les erreurs.
— **Méthodes spécifiques** : J'ai pu créer des méthodes dédiées, comme AjouterObjet et RetirerObjet, pour gérer efficacement les ajouts et suppressions d'objets tout en gérant les erreurs (par exemple, lorsqu'un emplacement est plein).

Cette approche m'a également permis d'ajouter des fonctionnalités avancées, comme la gestion des limites de stockage par emplacement. Par exemple, un joueur ne peut pas accumuler des quantités infinies d'un même objet, ce qui préserve l'équilibre du jeu.

4.3 L'état actuel des choses

Malgré ces avancées, nous avons constaté que la gestion actuelle des cellules d'inventaire restait trop limitée. La structure actuelle ne nous permet pas de gérer efficacement l'inventaire des joueurs, surtout dans des scénarios où une flexibilité accrue est nécessaire. C'est pourquoi nous explorons actuellement des solutions alternatives, notamment l'idée de remplacer le système basé sur un dictionnaire par un tableau d'objets représentant des cellules. Avec ce nouveau modèle, chaque cellule pourrait être gérée individuellement tout en restant liée au reste de l'inventaire. Cette approche offrirait une meilleure modularité et une gestion plus précise des ressources.

Enfin, il est important de souligner que cette progression sur l'aspect fonctionnel de l'inventaire n'est qu'une partie du travail. L'intégration graphique de cette fonctionnalité dans le jeu reste une problématique importante à résoudre. Une interface intuitive et visuellement cohérente est essentielle pour garantir une bonne expérience utilisateur.

5 Design et graphismes - Cyril Dejouhanet

5.1 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 été une étape cruciale. L'objectif était de sélectionner un outil adapté à nos besoins spécifiques en termes de simplicité d'utilisation, d'efficacité et de compatibilité avec notre workflow. Après avoir exploré plusieurs options, c'est finalement Pixel Studio sur Steam qui a été retenu.

Nous avons tout d'abord envisagé Aseprite, un logiciel bien connu pour sa spécialisation dans le pixel art. Son interface, bien que fonctionnelle, reste quelque peu vieillissante et son prix, relativement élevé pour un outil dédié uniquement au pixel art, a pesé dans la balance. De son côté, Photoshop, outil de référence pour le graphisme en général, ne s'avère pas spécialement conçu pour le pixel art. Bien que puissant, sa complexité et son modèle d'abonnement mensuel ne répondaient pas à nos besoins spécifiques, d'autant plus qu'il nécessitait une certaine courbe d'apprentissage pour être exploité efficacement dans le cadre de notre projet. Piskel, bien que gratuit et facile d'accès, n'offre pas suffisamment de fonctionnalités avancées pour convenir à des exigences plus pointues, notamment en termes de gestion d'animations ou de calques.

Après cette évaluation, Pixel Studio est apparu comme l'outil le plus adapté à nos attentes. Il combine une interface moderne et intuitive, tout en offrant des fonctionnalités robustes dédiées spécifiquement au pixel art, comme la gestion des calques et des animations. Son rapport qualité-prix, ainsi que sa disponibilité sur Steam, en font un choix particulièrement avantageux pour un projet de cette envergure. De plus, il est parfaitement optimisé pour les tablettes graphiques, ce qui correspond à nos préférences en termes de workflow.

En conclusion, le choix de Pixel Studio a été motivé par son efficacité, sa simplicité d'utilisation et sa compatibilité avec les exigences techniques et artistiques du projet. Il constitue ainsi un choix judicieux, permettant de créer les visuels pixel art de *Uzin* tout en offrant une expérience fluide et agréable tant pour les concepteurs que pour les joueurs.

5.2 Le design en soi

Le design et les graphismes de *Uzin* jouent un rôle essentiel dans la création d'une expérience de jeu immersive et engageante. L'utilisation d'une carte générée aléatoirement, comme mentionné dans la partie sur les fonctionnalités,

garantit un renouvellement constant de l'expérience de jeu. Cette génération aléatoire offre à chaque joueur l'opportunité de découvrir un environnement unique, augmentant la rejouabilité tout en enrichissant l'exploration.

Ce qui fait la particularité visuelle d'*Uzin* réside dans son style distinctif, qui combine des couleurs vibrantes et des transitions fluides entre les types de terrain. La carte est composée de cases modulaires où chaque élément visuel a été soigneusement conçu : un sol aux teintes roses éclatantes, des lacs d'un mauve profond, et un système de rebords dynamiques. Ces rebords s'adaptent automatiquement à la disposition des tuiles d'eau et de sol, ce qui permet de créer une apparence naturelle et harmonieuse tout en préservant la structure modulaire des cases, essentielle pour le gameplay.

5.3 Le système de calques

Pour structurer les éléments de la carte, un système de calques a été développé, divisant chaque case en trois niveaux distincts. Cette organisation permet une gestion claire, flexible et extensible des données visuelles et fonctionnelles de chaque case. Voici un détail de ces calques :

— **Le premier calque : le type de sol**
Ce calque constitue la base de chaque case. Il détermine la nature fondamentale du terrain : herbe, eau, roche, sable, etc. Ce niveau influence directement les propriétés de la case, comme la possibilité ou non de s'y déplacer, ou les restrictions concernant la construction. Par exemple, une case d'eau empêche le déplacement ou la construction de certains bâtiments, tandis qu'une case d'herbe peut accueillir des structures simples.

— **Le deuxième calque : les éléments et ressources**
Les objets sont désormais représentés par des identifiants uniques, ce qui simplifie leur gestion et réduit les erreurs. Le deuxième calque ajoute une dimension fonctionnelle à la case en définissant les objets ou caractéristiques qui s'y trouvent :

- **Ressources exploitables** telles que des arbres, des minerais ou des plantations.
- **Caractéristiques environnementales** comme des falaises, des montagnes ou des fosses.

Ce niveau est crucial pour le gameplay car il détermine les interactions possibles entre le joueur et la carte, comme l'extraction de ressources ou les obstacles naturels à contourner.

— **Le troisième calque : les bâtiments et constructions**
Ce calque gère les possibilités de construction sur la case, en prenant en compte les informations des deux premiers niveaux. Par exemple :

  — Une case d'herbe avec une ressource spécifique (comme du bois) pourra permettre la construction d'un bâtiment qui exploite cette ressource.
  — Une case rocheuse pourrait limiter les options de construction à des structures adaptées à un terrain difficile.

Ce calque garantit que chaque construction est logique et cohérente avec les conditions du terrain.

Chaque case est ainsi définie par un "type" combiné, qui intègre les données des trois calques. Cette structure modulaire permet de simplifier la gestion des éléments de la carte :

— **Gestion des déplacements** : empêcher un personnage de marcher sur un lac ou à travers une falaise.

— **Logique de construction** : permettre ou interdire la construction selon les conditions locales.

— **Facilité d'extension** : intégrer de nouveaux types de terrain, de ressources ou de bâtiments sans modifier la logique sous-jacente.

5.4 Une expérience pensée pour l'immersion

Cette architecture technique est directement liée à l'expérience visuelle et ludique. En combinant un design esthétique soigné et un système de calques bien structuré, *Uzin* garantit une lisibilité optimale pour les joueurs. Chaque décision de gameplay est renforcée par une cohérence visuelle qui aide les joueurs à comprendre les interactions possibles sans effort.

Notre objectif est de proposer un monde agréable à explorer, où les visuels captivants et les mécaniques bien intégrées se complètent pour offrir une expérience fluide et immersive. Ce système modulaire pose également des bases solides pour le développement futur, permettant d'ajouter de nouvelles fonctionnalités ou d'enrichir l'univers graphique sans compromettre la cohérence ou la performance du jeu.

6 Génération de la carte - Tristan Druart

Dans *Uzin*, chaque nouvelle partie se déroule sur une carte différente, générée de manière procédurale. Nous avons exploré plusieurs approches pour accomplir cette tâche, notamment l'algorithme **Wave Function Collapse** (WFC) et le **bruit de Perlin**. Ces deux méthodes présentent des avantages et des défis techniques.

6.1 Tentative avec l'algorithme WFC

L'algorithme Wave Function Collapse, ou Fonction d'écrasement de vague, est une méthode largement utilisée pour générer des cartes en remplissant une grille vide à partir de tuiles prédéfinies. Ce processus repose sur des règles d'adjacence entre les tuiles, un peu comme un sudoku. Son fonctionnement est le suivant :

— **Sélection initiale** : On choisit une case sur la grille, soit de manière contrôlée (par exemple, la première case), soit de manière aléatoire. Une tuile est placée dans cette case, comme une tuile "eau" ou "sol".

— **Règles d'adjacence** : Chaque tuile est associée à une liste des tuiles qui peuvent être placées à côté d'elle. Lorsqu'une tuile est placée, les cases adjacentes sont contraintes : seules les tuiles autorisées par les règles d'adjacence peuvent être placées dans ces cases.

— **Effondrement progressif** : L'algorithme sélectionne ensuite la case ayant le moins de possibilités de tuiles (c'est-à-dire la case la plus contrainte) et y place une tuile choisie aléatoirement parmi celles autorisées. Ce processus se répète jusqu'à ce que la grille soit remplie.

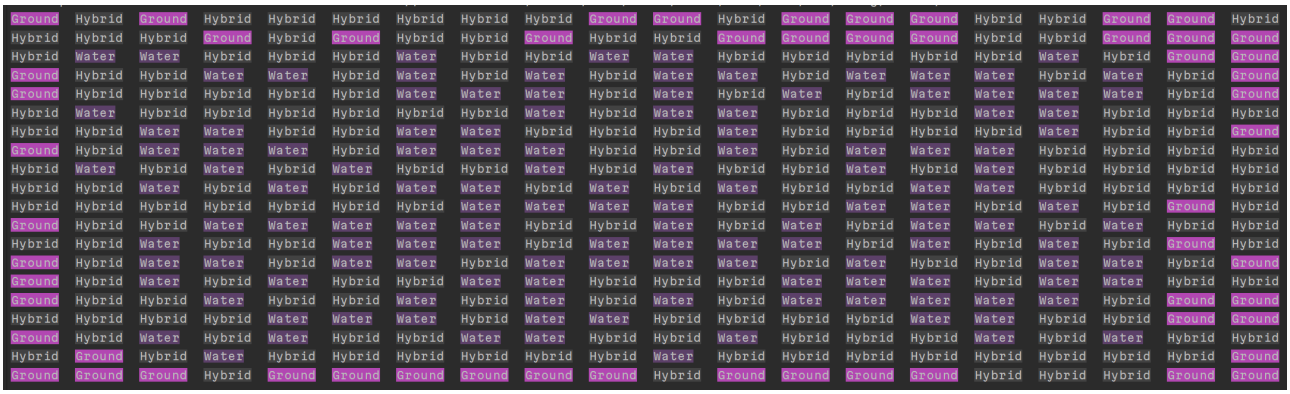


Figure 1 – Essai d'implémentation WFC infructueux dans la console Rider

6.1.1 Les défis rencontrés

Malgré ses avantages théoriques, l'implémentation de l'algorithme WFC a posé plusieurs problèmes :

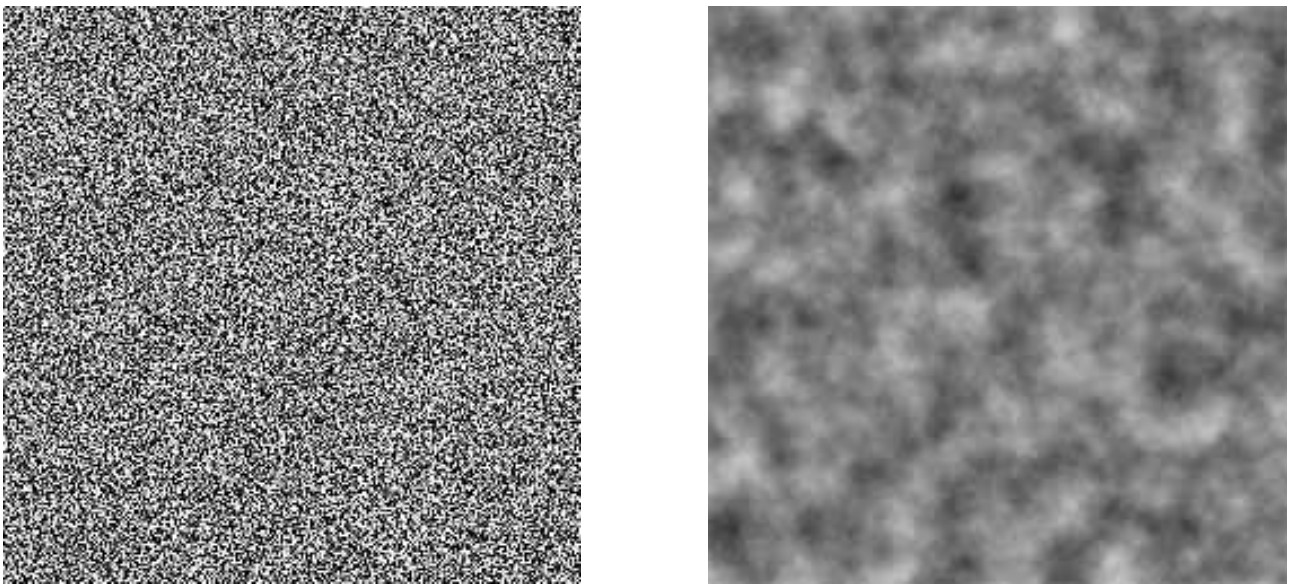
— **Création manuelle des règles d'adjacence** : Il fallait définir une liste exhaustive des tuiles voisines possibles pour chaque tuile, ce qui s'est avéré laborieux et peu évolutif. Ajouter une nouvelle tuile aurait nécessité de modifier manuellement les listes pour toutes les tuiles existantes.

— **Solution envisagée** : Nous avons tenté de résoudre ce problème en créant une scène d'exemple où l'algorithme pouvait observer les relations entre les tuiles pour générer automatiquement les listes d'adjacence.

— **Résultats insatisfaisants** : Malgré ces efforts, la génération des cartes n'était pas naturelle. Les tuiles hybrides (reliant eau et sol) étaient placées de manière incohérente, donnant des résultats visuellement décevants.

6.2 Passage au bruit de Perlin

Face aux limites du Wave Function Collapse, nous avons adopté une approche différente : le bruit de Perlin. Cette méthode, très populaire dans la génération de cartes de jeux vidéo, permet de produire des paysages plus naturels et visuellement cohérents.



(a) Illustration pour le bruit "simple"

(b) Illustration pour le bruit de Perlin

FIGURE 2 – Différents types de bruits

Le bruit de Perlin est une fonction mathématique qui génère des motifs lisses et continus, souvent utilisés pour représenter des terrains, des nuages ou

des textures. Contrairement au bruit classique (qui produit des valeurs aléatoires et abruptes), le bruit de Perlin assure que les valeurs adjacentes varient progressivement, créant des dégradés harmonieux.

6.2.1 Un exemple d'implémentation possible

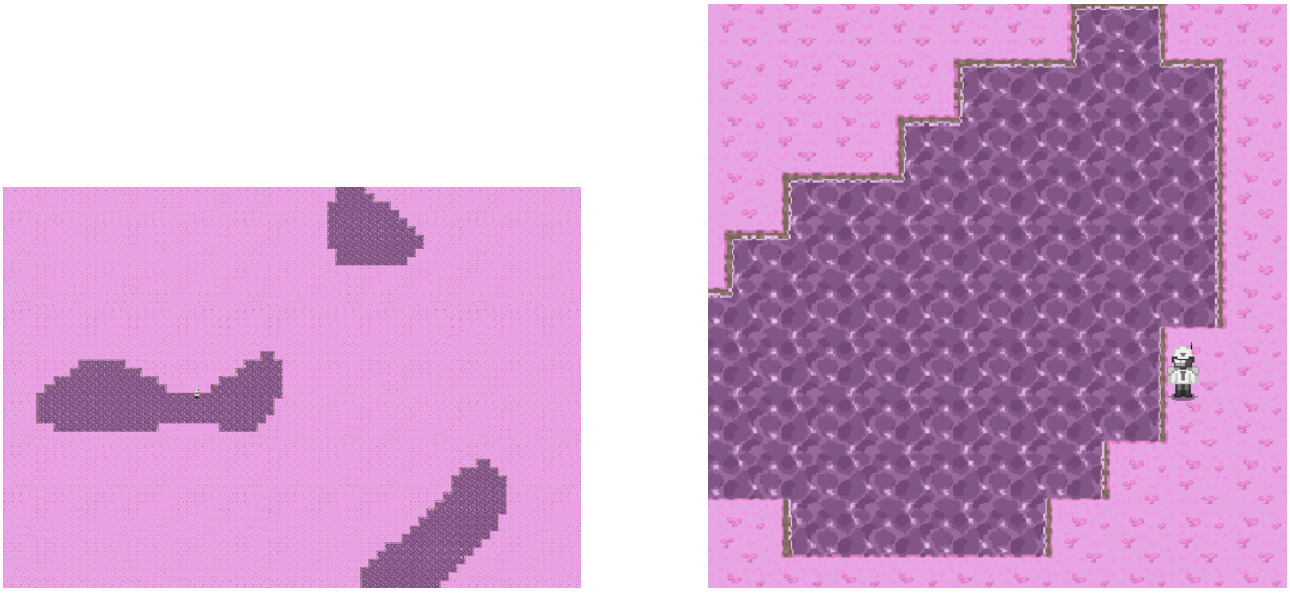
— **Génération d'une carte de bruit** : Nous utilisons la fonction de bruit intégrée en C# pour générer une image où chaque pixel a une valeur entre 0 (noir) et 1 (blanc).

— **Traduction des valeurs en tuiles** : Les pixels ayant une valeur inférieure à un seuil (par exemple, 0.2) sont convertis en tuiles d'eau. Les autres valeurs représentent des tuiles de sol ou d'autres types de terrain.

— **Résultats visuels** : Cette méthode garantit la création de lacs et de reliefs cohérents, évitant les ruptures visuelles observées avec le Wave Function Collapse.

6.3 La tuile de fin

En combinant nos apprentissages des deux méthodes, nous avons opté pour le bruit de Perlin pour la génération des cartes dans *Uzin*. Cette approche, bien que moins personnalisable que le Wave Function Collapse, nous permet de créer des environnements visuellement cohérents et d'économiser du temps de développement. À l'avenir, nous pourrions envisager de combiner ces deux algorithmes pour tirer parti de leurs points forts respectifs, offrant ainsi des cartes encore plus immersives et variées.



(a) Map simple (b) Map avec tuiles hybrides

FIGURE 3 – Exemples de génération de map

7 Site web - Maxan Fournier

7.1 L'importance du site web

Le site web de *Uzin* est conçu pour être bien plus qu'une simple vitrine : c'est une passerelle interactive entre les développeurs et les joueurs, offrant un aperçu captivant de l'univers du jeu. Chaque page est pensée pour transmettre les valeurs fondamentales du projet, tout en engageant le visiteur grâce à un design moderne et immersif. La page d'accueil, par exemple, met immédiatement le joueur dans l'ambiance avec un slogan dynamique, des illustrations attrayantes et des conseils clés sur le gameplay, tels que la collecte, l'automatisation, la défense et l'exploration. Cette approche guide les visiteurs dans la découverte des mécaniques du jeu, tout en leur permettant de s'imaginer en pleine action.

La section **Équipe** joue un rôle crucial dans l'humanisation du projet. Elle met en avant les membres de l'équipe avec des biographies illustrées et engageantes. Cela permet de souligner les compétences et la diversité des talents qui se cachent derrière *Uzin*. En identifiant chaque rôle, comme le chef de projet, le designer, le storyteller ou le développeur, cette page renforce la transparence et suscite un lien personnel avec les joueurs.

La page **Documents** propose un accès organisé aux ressources liées au projet, comme les cahiers des charges fonctionnel et technique, ou encore les rapports de soutenance. Ce volet est essentiel pour partager avec les parties prenantes les étapes clés du développement et les choix techniques effectués. Le design épuré et les icônes explicites facilitent la navigation et rendent la consultation des documents agréable, même pour des utilisateurs non techniques.

7.2 Le choix d'un framework web

Pour concevoir le site web d'*Uzin*, j'ai fait le choix d'utiliser un framework web moderne, en l'occurrence **Nuxt.js**, plutôt qu'un simple site HTML statique. Cette décision repose sur plusieurs avantages essentiels pour un projet dynamique comme le nôtre. Ayant déjà une bonne maîtrise du langage **Vue.js** et appréciant particulièrement sa syntaxe claire et intuitive, ce choix m'a paru évident et m'a conforté dans l'idée de m'appuyer sur son extension Nuxt.js pour bénéficier d'outils encore plus avancés. Un framework comme Nuxt.js me permet de structurer efficacement le code, d'optimiser les performances avec un rendu côté serveur (SSR) et de gérer les routes de manière fluide, ce qui est indispensable pour offrir une expérience utilisateur de qualité.

Par exemple, la possibilité de charger des données dynamiques ou de créer des composants réutilisables facilite grandement la maintenance et l'évolution du site. De plus, grâce à Nuxt.js, j'ai pu intégrer facilement des fonctionnalités avancées comme la gestion des états interactifs, ce qui aurait été beaucoup plus complexe avec un site HTML statique. En somme, ce choix me permet non seulement de gagner du temps lors du développement, mais aussi d'assurer une meilleure adaptabilité à l'avenir.

7.3 Le CSS revisité, une bonne idée ?

Pour le développement du site web d'*Uzin*, j'ai choisi d'utiliser **TailwindCSS** comme framework CSS, plutôt que d'écrire du CSS classique. Ce choix s'explique principalement par les avantages significatifs que Tailwind offre en termes de productivité et de gestion des styles. Contrairement au CSS traditionnel, où les règles de style sont définies dans des fichiers séparés, Tailwind adopte une approche utilitaire qui permet de concevoir des interfaces directement dans le HTML, grâce à des classes prédéfinies. Cette façon de procéder m'a permis de gagner un temps précieux et d'assurer une plus grande cohérence visuelle sur l'ensemble du site.

7.3.1 Pourquoi TailwindCSS plutôt que du CSS classique ?

L'un des principaux avantages de TailwindCSS est sa modularité. Chaque classe utilitaire représente un style précis (comme `text-center`, `bg-purple-500`, ou `rounded-lg`), ce qui permet de concevoir des composants rapidement sans avoir à écrire manuellement des règles CSS répétitives. Avec CSS classique, il aurait fallu maintenir un fichier séparé pour chaque style et écrire beaucoup de code personnalisé, ce qui aurait rendu le projet plus long à développer et plus difficile à maintenir. En revanche, avec Tailwind, j'ai pu créer un design réactif et esthétique en me concentrant sur l'assemblage des classes existantes.

Un autre point fort est la gestion native des styles réactifs. Tailwind intègre des classes spécifiques pour différentes tailles d'écran (`sm:`, `md:`, `lg:`, etc.), ce qui m'a permis d'adapter facilement le site à tous les appareils. Par exemple, certaines sections comme la page d'accueil et la galerie d'équipe nécessitaient un affichage différent selon qu'on les consulte sur un smartphone ou un écran d'ordinateur. Grâce à Tailwind, ces ajustements étaient rapides et intuitifs.

7.3.2 Une utilisation simple et efficace

L'utilisation de TailwindCSS repose sur l'ajout de classes utilitaires directement dans le HTML ou les fichiers Vue. Par exemple, pour concevoir un bou-

ton, j'ai utilisé une combinaison de classes comme `bg-purple-500 hover:bg-purple-700 text-white font-bold py-2 px-4 rounded`. Ces classes s'appliquent instantanément, permettant de visualiser immédiatement les changements de style. Pour les cas plus complexes, j'ai également utilisé le fichier de configuration `tailwind.config.js`, qui m'a permis de personnaliser les couleurs, les polices et les espacements afin qu'ils s'harmonisent parfaitement avec l'identité visuelle d'*Uzin*.

7.3.3 Inconvénients et défis rencontrés

Malgré ses nombreux avantages, TailwindCSS présente aussi certains inconvénients. Le principal problème que j'ai rencontré est la longueur des classes dans le HTML, qui peut rapidement rendre le code difficile à lire et à maintenir. Par exemple, une simple `div` peut se retrouver avec une dizaine de classes, ce qui complique la lisibilité. Pour résoudre ce problème, j'ai utilisé des directives comme `@apply` dans des fichiers CSS spécifiques, ce qui m'a permis de regrouper plusieurs classes utilitaires sous un seul nom.

Un autre défi était lié à la taille initiale des fichiers CSS générés. En raison du grand nombre de classes utilitaires, le fichier CSS final peut devenir volumineux. Pour y remédier, j'ai utilisé PurgeCSS, intégré dans Tailwind, qui élimine les classes inutilisées lors de la compilation, réduisant ainsi la taille du fichier CSS en production.

Enfin, bien que Tailwind simplifie le processus de stylisation, il exige un certain temps d'adaptation. La mémorisation des classes utilitaires était déroutante au début, mais j'ai rapidement gagné en efficacité grâce à la documentation claire et complète de Tailwind.

7.4 Hébergement et nom de domaine

Actuellement, le site web d'*Uzin* est hébergé sur la plateforme **Vercel**, qui offre une solution performante et adaptée pour le déploiement d'applications web modernes. Le choix de Vercel s'explique par ses nombreux avantages : une intégration transparente avec Nuxt.js, un processus de déploiement simplifié, et des performances optimales grâce à un réseau de distribution de contenu (CDN) intégré. Vercel gère automatiquement le rendu des pages et leur mise en cache, ce qui garantit une rapidité d'accès pour les visiteurs, où qu'ils se trouvent.

En ce qui concerne le nom de domaine, le site est actuellement accessible via l'URL `uzin.vercel.app`, fournie par défaut par la plateforme. Ce sous-domaine est temporaire et sert principalement à tester les fonctionnalités et

assurer une visibilité pendant la phase de développement. À l'avenir, nous envisageons d'acquérir un nom de domaine personnalisé, tel que `uzin-game.com`, pour renforcer l'identité du projet et offrir une expérience utilisateur plus professionnelle. Cela permettra également de mieux refléter l'envergure du projet et d'améliorer sa crédibilité auprès des visiteurs.

L'hébergement sur Vercel est une solution idéale pour cette première phase, car elle combine facilité d'utilisation et coûts réduits, ce qui est crucial pour un projet étudiant comme le nôtre. De plus, la possibilité de configurer rapidement un domaine personnalisé nous donne la flexibilité nécessaire pour évoluer vers une version finale du site une fois que le développement sera abouti.

7.5 Ce qu'il reste à faire pour les prochaines soutenances

Pour les prochaines phases du projet, le site web d'*Uzin* devra évoluer afin de mieux refléter l'état d'avancement du jeu et offrir une immersion plus complète aux visiteurs. Actuellement, le site met en avant les concepts principaux et les fonctionnalités du jeu de manière textuelle et visuelle, mais il est essentiel d'ajouter des éléments concrets directement issus du projet pour permettre aux utilisateurs de mieux s'y projeter.

7.5.1 Intégration de captures d'écran et de visuels du jeu

L'une des priorités absolues est d'inclure des captures d'écran et des illustrations tirées directement du jeu. Ces visuels permettront de donner vie au contenu du site en montrant aux visiteurs à quoi ressemble réellement *Uzin*. Par exemple, des images des usines en construction, des créatures hostiles ou des environnements pourraient renforcer l'intérêt et l'engagement des joueurs potentiels. Ces éléments serviront également à valider les efforts visuels et techniques réalisés par l'équipe.

7.5.2 Démonstrations interactives

Outre les captures d'écran, il serait pertinent d'intégrer des démonstrations interactives, comme des animations de gameplay ou des vidéos courtes, pour illustrer des aspects spécifiques du jeu : la collecte des ressources, l'automatisation des usines ou encore les mécanismes de défense. Cela offrirait aux visiteurs une idée concrète de l'expérience utilisateur et les inciterait à explorer davantage le projet.

7.5.3 Création d'une roadmap visible

Pour les visiteurs, il est important de comprendre où en est le projet et quelles seront les prochaines étapes. Une section dédiée à la roadmap du projet, avec des étapes clés clairement définies et des illustrations des fonctionnalités en cours de développement, renforcerait la transparence et inciterait les joueurs à revenir régulièrement pour voir les progrès réalisés.

7.5.4 Optimisation technique

Enfin, il sera important d'améliorer la configuration actuelle pour supporter une charge croissante d'utilisateurs, surtout une fois que le jeu sera disponible en téléchargement. Cela pourrait inclure la migration vers un domaine personnalisé, l'optimisation du site pour de meilleures performances, et peut-être même l'ajout d'un système d'authentification pour permettre aux joueurs de se connecter et de suivre leurs statistiques directement depuis le site.

7.6 Balise fermante

Pour les prochaines soutenances, le site web d'*Uzin* doit évoluer pour refléter fidèlement l'avancée du projet. Grâce à l'ajout d'éléments visuels issus du jeu, d'animations explicatives, et d'une roadmap claire, il deviendra un outil essentiel pour communiquer efficacement sur le travail accompli et les étapes à venir. Ces améliorations renforceront la présentation du projet tout en mettant en avant son originalité et son potentiel.

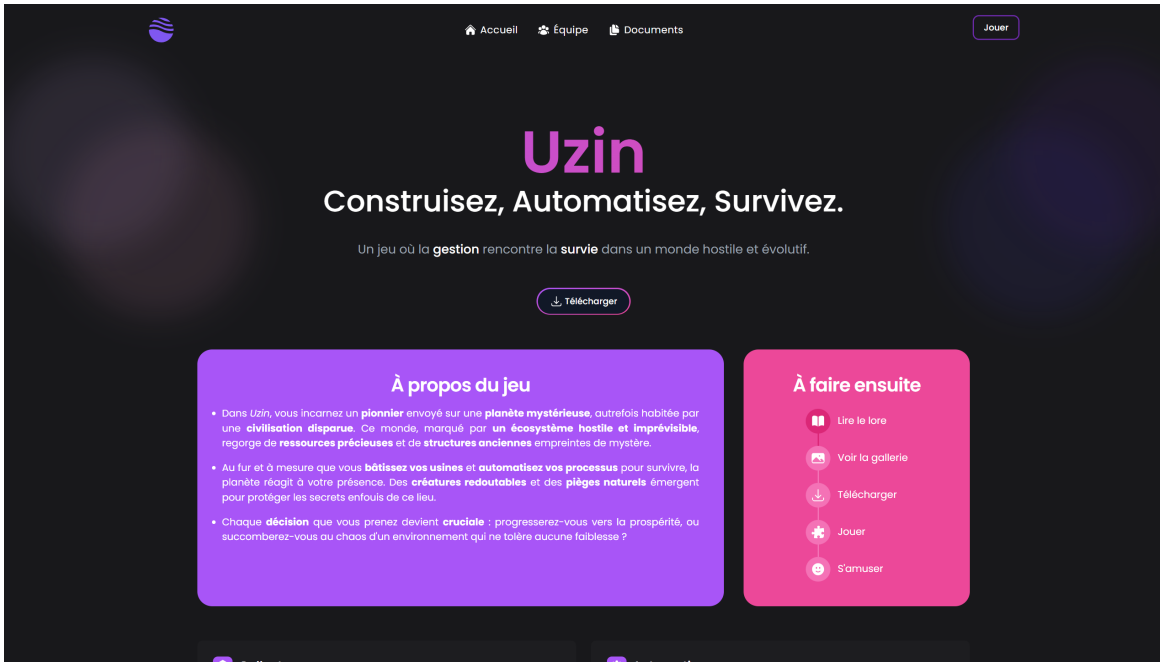


FIGURE 4 – Page d'accueil du site internet du jeu-vidéo

8 Conclusion

Le projet *Uzin* est une aventure passionnante qui va bien au-delà du simple développement d'un jeu vidéo. Depuis le début, chaque membre de l'équipe a mis à profit ses compétences et son énergie pour avancer sur des aspects variés, comme la création d'un univers captivant, la mise en place de systèmes techniques complexes, et le développement de graphismes immersifs. Chacun a apporté sa pierre à l'édifice, ce qui nous a permis d'atteindre des résultats concrets et encourageants.

Cette première phase a été l'occasion d'expérimenter de nouveaux outils et d'affiner notre organisation. Par exemple, le choix de technologies modernes comme Nuxt.js ou Pixel Studio nous a permis de gagner en efficacité et de poser des bases solides pour le futur. L'univers du jeu, avec ses thèmes profonds comme l'impact humain sur l'environnement et la survie dans un monde hostile, commence également à prendre forme et à offrir une expérience riche en réflexion et en stratégie.

Cela dit, le travail est loin d'être terminé. Nous avons identifié plusieurs priorités pour les prochaines étapes, comme l'intégration d'éléments visuels sur le site web, l'ajout de mécaniques de gameplay, et l'optimisation des systèmes techniques déjà en place. Ces améliorations sont essentielles pour rendre *Uzin* encore plus abouti et immersif.

En résumé, cette première étape marque un bon départ pour notre projet. Nous sommes fiers du chemin parcouru et motivés pour continuer à avancer. Avec les idées et l'élan que nous avons maintenant, nous sommes impatients de voir ce que les prochaines phases vont nous permettre d'accomplir.