# Rapport de 1ère soutenance

# Projet Overdrive

Lucas Flacelière, Vasile Ionescu, Clément Bertrand, Mattieu Bourgeois, Anatole Brunelle

Vendredi 16 Janvier 2026

# Table des matières

# 1 Introduction

Aujourd'hui, nous vivons une époque où le monde du jeu vidéo a connu une transformation radicale ; depuis des dizaines d'années, les technologies ne cessent de repousser les limites du possible. Cependant, dans cette industrie devenue ultra-concurrentielle, la puissance technique et la sophistication des moteurs graphiques ne suffisent plus à garantir un succès critique ou commercial.
En effet, nous observons une tendance vers des jeux de plus en plus complexes et chronophages, exigeant un investissement personnel parfois épuisant. Cela a doucement mais massivement exclu une partie des joueurs qui cherchaient eux un moyen de détente et de relaxation (ce pour quoi les jeux avaient été créé en partie à l'origine). Même les jeux qui se revendiquent sans prise de tête ou "chill" ont toujours des mécaniques de classements ou de timer intégrés sous une forme ou une autre.
C'est pour faire face à ce problème que nous avons décidé de créer notre propre jeu, un jeu qui permet une vraie expérience de détente dans laquelle le joueur n'a aucune contrainte.

L'idée principale du jeu est d'errer dans une vaste carte et de créer sa propre ville. Des ressources s'offrent au joueur sous certaines formes, certaines sont plutôt simples à récupérer et utiliser tandis que d'autres moins et devront s'acquérir en explorant ou en résolvant des énigmes présentes dans le monde.

# 2 Avancé du projet

## 2.1 Lucas Flacelière

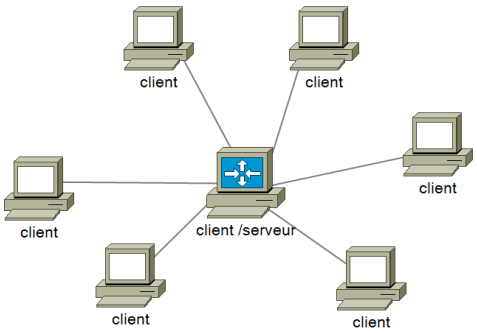
<u>Mes principales tâches dans le projet sont :</u>

1-programmer une architecture réseau en client-serveur de type LAN

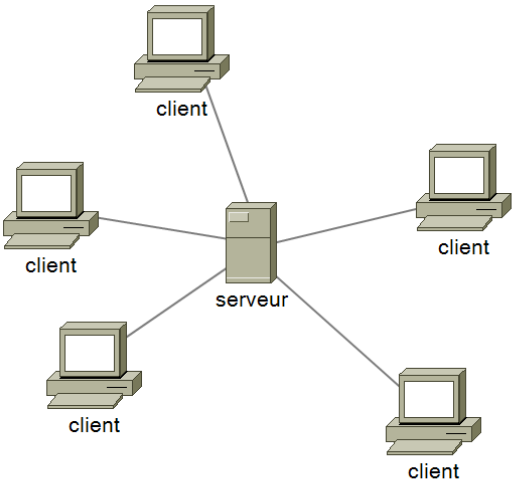
2-l'implémentation des différentes mécaniques de gameplay au réseau avec notamment la gestions des ressources mises en commun entre les joueurs.

1. La programmation d'une LAN

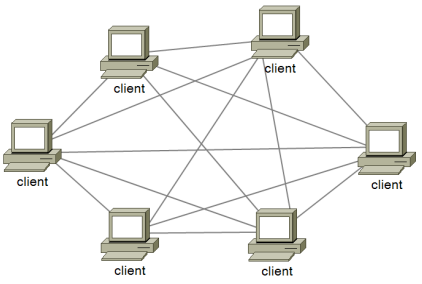
Notre décision quant à une architecture réseau du type client-serveur de type LAN au mépris d'une architecture peer-to-peer ou P2P repose sur le fait qu'une architecture peer-to-peer serait inutilement complexe pour le type de notre jeu. Aussi, grâce à cette architecture nous permettons de garder les avantages de l'architecture client-serveur sans avoir l'inconvénient de devoir trouver un serveur.




LAN Architecture




architecture client-serveur




Architecture Peer-to-Peer

Dans une architecture LAN un joueur appelé hôte se verra obtenir en plus du rôle de client, le rôle de serveur pour les autres joueurs. De ce fait, le joueur avec le double rôle aura donc plus de tâche à gérer simultanément. Pour ce faire nous devons importer la bibliothèque python ‘socket’ qui va nous permettre de créer nos interactions client-serveur.

En effet, la bibliothèque socket permet de faire communiquer des programmes Python entre eux à travers un même réseau (internet, Bluetooth, Ethernet par exemple) mais aussi sur une même machine en envoyant et recevant des données via les protocoles TCP ou UDP.

Ici le protocole TCP sera utilisé pour des raisons pratiques car il permet une connexion constante avec le client. Notre jeu possède pour le moment la fonctionnalité de créer un hôte et de se connecter avec. Le gameplay multijoueur n’a pour le moment pas été implanté.

2. Implémentation des différentes mécaniques de gameplay au réseau avec notamment la gestions des ressources mises en commun entre les joueurs.

Pour le moment j’ai développé dans le langage python une base fonctionnelle d’architecture LAN.

En effet, ici on peut observer que l’on importe la bibliothèque “socket” pour la connexion entre ordinateur puis on définit le type de connexions souhaitées ici internet grace a “socket.AF_INET” et enfin on définit le protocole utiliser. régis par la commande “socket.SOCK_STREAM” qui permet d’utiliser le protocole TCP.

A l'avenir, nous avons prévu d'intégrer les mécaniques multijoueur au gameplay de tels sortes à ce que lorsqu'un des clients fait une action dans la partie, par exemple récolter des ressources, bouger un bâtiment ou encore construire des bâtiments.

```python
import socket

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('0.0.0.0', 9999))
server.listen()

while True:
    client, addr = server.accept()
    print(client.recv(1024).decode())
```

Aussi, l'un des défis du multijoueur sera de l'optimiser au mieux. En effet, avoir un jeu avec un multijoueur mal optimisé est un vrai problème. Pour cause, il peut être réellement frustrant pour un joueur de devoir attendre pour que ses actions soient réalisées. Par exemple lors du déplacement d'un bâtiment il est particulièrement important qu'il n'y ai pas un délai excessif pour éviter qu'il ne place ce dernier au mauvais endroit.

Enfin, il faudra faire très attention à ce que l'hôte ne se déconnecte pas n'importe comment puisque s'il se déconnecte tous les autres joueurs seront déconnectés du fait de l'architecture LAN choisit. Il faudra donc implémenté des avertissements pour l'hôte si d'autres joueurs sont connectés alors que l'hôte souhaite quitter le jeu. Et peut-être dans ce cas-là essayer de changer d'hôte. Mais cette dernière idée n'est pas la priorité absolue.

Donc, les priorités sont, implémenter le gameplay multijoueur, une fois cette tâche faite, optimiser les connexions pour éviter les délais et enfin faire attention à ce que l'hôte ne puisse pas se déconnecter simplement.

## 2.2 Vasile Ionescu

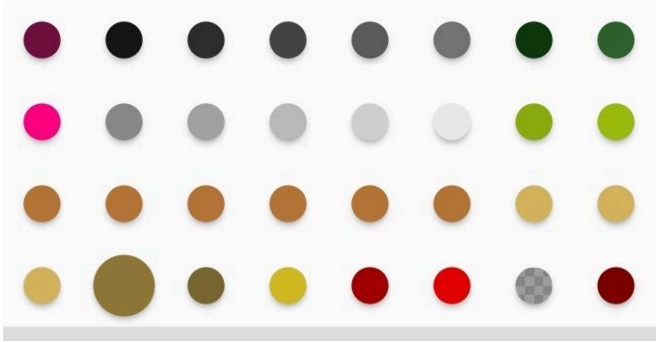
Mes principales tâches dans le projet sont :

1. Le design des interfaces graphiques du jeu.
2. L'implémentation des interfaces graphiques comme l'inventaire, le menu, les différentes boutiques, l'interaction éventuelle avec des NPC, toutes les interfaces nécessaires au joueur comme la vie ou le système de monnaie.

1. Le design des interfaces graphiques du jeu.

Un jeu dans le style que nous recherchons, nécessite des interfaces graphiques qui sont différentes du monde en lui-même car elles vont se mettre par dessus et vont toujours être visibles. Nous avons décidé de nous axer sur le style steampunk en pixel art ce qui implique des interfaces graphiques cohérentes avec cela. J'ai donc décidé d'installer un logiciel de création de pixel art sur ma tablette graphique et j'ai regardé quelques tuto pour apprendre les bases et sur comment bien gérer les proportions afin d'avoir le meilleur rendu possible une fois importé dans le jeu. J'ai donc commencé à réaliser les bases afin de m'habituer avec l'outil et j'ai fait une barre de chargement ainsi qu'une barre de vie.

Pour la réalisation de cette barre de chargement, j'ai dû faire face à certaines contraintes techniques et visuelles. Premièrement, la barre doit être divisée en deux parties, une barre pleine et une vide. Ceci pour pouvoir lors de l'implémentation, animer la barre de chargement. Deuxièmement, il a fallut chercher une palette de couleurs qui correspond parfaitement aux attentes du groupe en terme de couleur et de ressenti steampunk, nous avons fini par choisir celle ci.



J'ai aussi réalisé une barre de vie dans le style steampunk avec les mêmes contraintes de création.

J’ai commencé la création d’autres interfaces plus complexes comme la boutique mais je n’ai pas encore de résultat concluant à cause de la taille et de la quantité de détails auxquels je vais devoir faire face. J’ai donc compris une chose, bien que la conception des interfaces en pixel art soit plus simple pour de petits éléments, elle se complexifie pour les éléments plus conséquents.

La création de ces éléments d’interfaces a été réalisée avec l’outil Pixel Brush qui est un outil gratuit et facile d’utilisation ce qui correspond parfaitement à quelqu’un comme moi qui débute et qui n’a jamais fait d’interfaces avant.

2. L'implémentation des interfaces graphiques comme l’inventaire, le menu, les différentes boutiques, l'interaction éventuelle avec des NPC, toutes les interfaces nécessaires au joueur comme la vie ou le système de monnaie.

L’implémentation des interfaces nécessite en partie l’utilisation de la POO qui est la Programmation Orienté Objet. Cela permet la création de classes qui correspondent à un élément en général. Par exemple, pour créer un bouton pour un menu, je crée une classe bouton et je vais l’utiliser à chaque fois que j’ai besoin d’un bouton et je vais spécifier des méthodes et des fonctions qui vont préciser ce que le bouton doit faire. Pour faire simple, la classe bouton décrit un comportement général du bouton et pour chaque bouton je vais spécifier son utilisation individuelle.

Cette programmation orientée objet permet de simplifier grandement l’implémentation des interfaces graphiques bien qu’elle ne soit pas toujours optimale. Certaines fonctionnalités comme le menu de chargement ne nécessite pas cette approche et doivent être codées individuellement.

Prenons un exemple précis pour expliquer comment je donne vie à mes interfaces. La barre de chargement lors du lancement du jeu. On pourrait croire qu’il faut dessiner l’état de cette barre au fur et à mesure du chargement, cependant Pygame, la bibliothèque python que nous utilisons permet de découper une image. Ainsi, le code va découper de moins en moins la barre et la révéler au fur et à mesure du chargement ce qui va simuler l’effet recherché, comme si la barre se déplace vers la droite alors qu' en réalité on change juste le découpage.
Voici une illustration non représentative du chargement définitif:

## 2.3 Clément Bertrand

Mes principales tâches dans le projet sont :

- L’aspect musical et les effets spéciaux sonores.
- l’aspect combat du jeu ?
- Toute la partie des mécaniques d’automatisation et des ressources qui seront présente dans le jeu

1. La musique

Pour commencer, je me suis attaqué aux musiques du jeu. Le but n'est évidemment pas d'avoir les musiques finales du jeu avant même que ce dernier ne soit créé mais nous avons trouvé important le fait d'avoir une direction artistique claire. Avoir des prototypes audio permet de mieux comprendre l'univers du jeu pour ne jamais s'en éloigner au fur et à mesure du développement. Il en va de même pour l'identité visuelle, travailler dans un bel environnement et avoir un progrès visuel (ou sonore) concret est réellement motivant.

Avant de participer à ce projet, je n'avais aucune expérience en composition musicale. Je n'avais jamais utilisé de logiciel malgré mon goût fort pour la musique. J'ai donc commencé par là: Trouver un logiciel. Les logiciels les plus performants étant tous payants, j'ai choisi une des meilleures options gratuite a mon avis: Soundtrap.

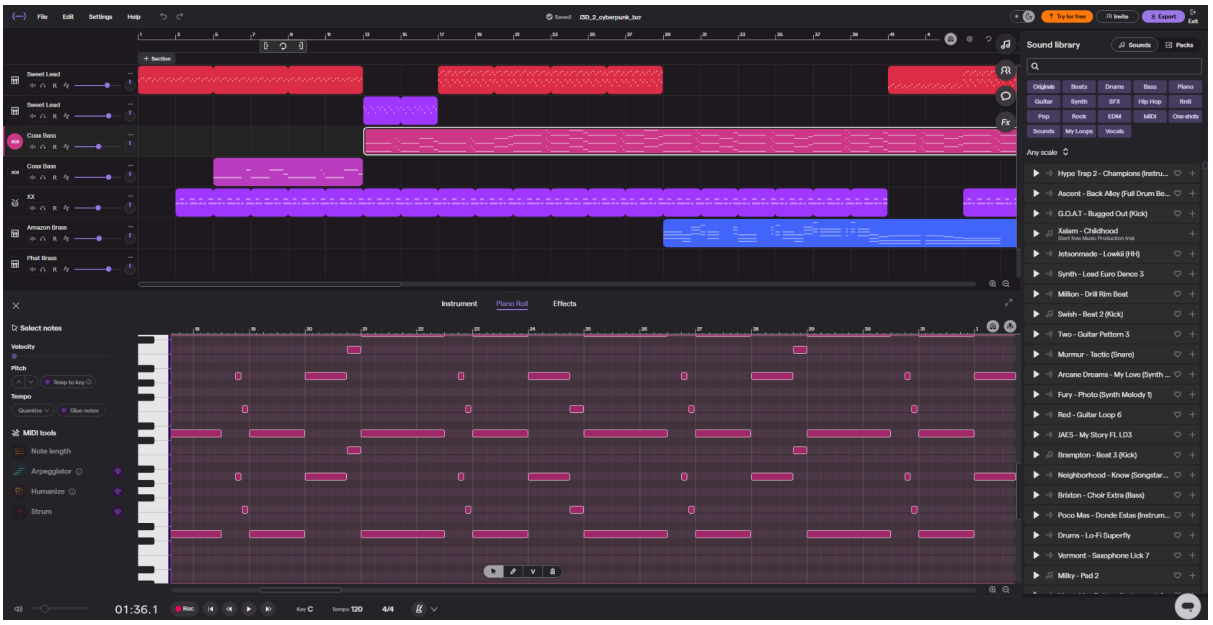
Je me suis alors retrouvé devant cette interface vide et c'était à moi de créer de la musique à partir de rien. Je m'y suis pris de la manière suivante, trouver une mélodie principale appelée "lead" puis construire une musique à partir de ça. Il faut donc ajouter une basse pour la profondeur, des percussions pour l'impact et le rythme, enfin on peut rajouter d'autres instruments pour complémenter la lead et rendre la musique plus intéressante. Croyez-moi cela parait plus simple que ça ne l'est réellement.

N'ayant aucune éducation musicale, même en ayant une idée en tête, il est très difficile de la traduire sur le logiciel. Malgré cela je ressent que je me suis amélioré au fur et à mesure de pratiquer.

Malheureusement, je n'ai pas accès à un logiciel plus puissant donc je suis réellement limité avec ce que je peux faire, je n'ai accès qu'à une petite partie de la bibliothèque de sons et d'effets (le reste étant bloqué derrière un abonnement premium). J'ai tout de même pu me débrouiller pour créer les premiers prototypes de musique pour notre jeu.

Ce dernier est donc, comme vous l'avez lu précédemment, axé sur la détente et le plaisir, il a donc fallu traduire cela dans l'univers sonore. On a donc décidé de partir dans un rythme plus lent et dans le style "lofi" pour faire ressentir au joueur un sentiment de confort. Le bpm (battements par minute) va donc être entre 60 et 90, c'est un rythme qui est plutôt lent comparé à la majorité de la musique d'aujourd'hui.

On peut voir sur la capture d'écran ci-dessous l'interface de Soundtrap:




On peut y voir:

- la “timeline”, c’est la ou les pistes sont affichées, c’est ici qu’on va travailler sur la structure du projet, on décidera quels instruments apparaîtront à quel moment etc.
- La partie du dessous est l’endroit où on va mettre les notes une par une. C’est donc ici qu’on va réellement créer la musique.
- Le navigateur, le menu à droite, c’est ici qu’on va chercher et sélectionner les instruments qu’on va ensuite utiliser.

Il est très difficile de ne pas se décourager car malgré les nombreux tutoriels sur internet, quand on veut créer quelque chose de nouveau et qui nous correspond, c’est à nous que revient 100% du travail. Mais en continuant et en franchissant cette première barrière, on remarque vraiment le côté addictif de la discipline. Quand on travaille pendant des heures sur une musique et que finalement on arrive à un résultat agréable à écouter et qui nous plait c’est vraiment très agréable et écoutant moi-même beaucoup de musique, pouvoir moi-même créer la musique que j’aime est une énorme source de motivation et je n’ai aucun doute que je continuerai la composition musicale même à la fin de ce projet.

2. Le combat

Malheureusement, cette mécanique du jeu est inaccessible pendant le début du développement. En effet, il faut déjà avoir avancé sur le jeu un minimum pour pouvoir commencer à travailler sur cette mécanique. Notre jeu est principalement un “city-builder” et l’aspect combat n’est que secondaire à l’expérience globale du jeu. Pour l’instant notre temps est investi dans:

- la création artistique du jeu, les premiers prototypes visuels et sonores.
- Les mécaniques des bâtiments et leurs interactions avec le personnage.

La création artistique est en cours comme vu précédemment et les bâtiments sont le gros du jeu, il donc est normal que nous n’ayons que des versions non-abouties de ces mécaniques.

Cependant, il est tout de même possible de réfléchir à ces mécaniques même si elles ne sont toujours pas implémentables immédiatement:
Le combat aura un rôle secondaire dans notre jeu, il servira en partie de “passe-temps” mais aussi le joueur pourra récupérer des ressources qui seront défendu (ou lâché) par des personnages non-joueurs(PNJ) qui marcheront aléatoirement autour de la zone de construction.
Ces PNJ seront des “ia” et auront des mouvements et un comportement pseudo-aléatoire. L’ia et le comportement des NPC ne sont pas ma responsabilité principale, cela sera expliqué plus en détail dans la partie d’Anatole.

3. Automatisation

Comme vous le savez maintenant, le joueur va devoir utiliser des ressources pour développer sa ville. Il faut donc se poser la question de la progression: Comment gérer la progression du joueur pour ne pas qu’elle soit trop rapide ou au contraire trop lent.

Pour cela, nous allons utiliser un système de ressources qu’on débloque au fur et à mesure. Certains puits de ressources de base se trouveront en abondance dans le monde et seront exploitables dès le début et vont permettre au joueur de se lancer sur de bonnes bases. On pourra donc par exemple mettre un bâtiment miniers sur ces minerais pour en extraire en continue. Un système de transport de ressource sera aussi implanté pour permettre leur utilisation et traitement en masse. Ces ressources seront soit prêtes à l’emploi soit utilisées dans d'autres machines avec d’autres ressources pour un traitement supplémentaire etc.

C’est donc comme ça que la progression va être gérée. Certaines ressources sont soit plus rares à obtenir sur la carte, soit elles requièrent plus de transformation pour les obtenir et

en échange, elles permettent de construire des bâtiments plus avancés qui permettent eux même d'en produire plus etc.

Il y a aussi une ressource qui sera plus importante que les autres, l'énergie. Nous n'avons pas encore décidé d'une version finale ou de quelle forme elle sera implantée mais c'est elle qui va être la composante majeure de l'avancement de la ville. En effet, presque toutes les structures auront besoin d'énergie pour fonctionner. Sans énergie, il est impossible de construire plus de bâtiments et donc d'avancer dans le jeu. Il sera possible d'obtenir cette ressource de différentes manières, certaines simples et possible tôt dans le jeu mais elles ne permettent qu'une quantité limitée de cette dernière tandis que d'autres méthodes et structures permettent une production plus importante mais au coût de nombreuses ressources.

Vous l'aurez compris, notre jeu se base sur l'exploitation du monde dans lequel on évolue et l'utilisation efficace des ressources présentes. Ces mécaniques ne sont pas encore implanté dans le jeu mais nous aurons très bientôt une base assez robuste et complète pour pouvoir travailler sur ces mécaniques plus spécifiques

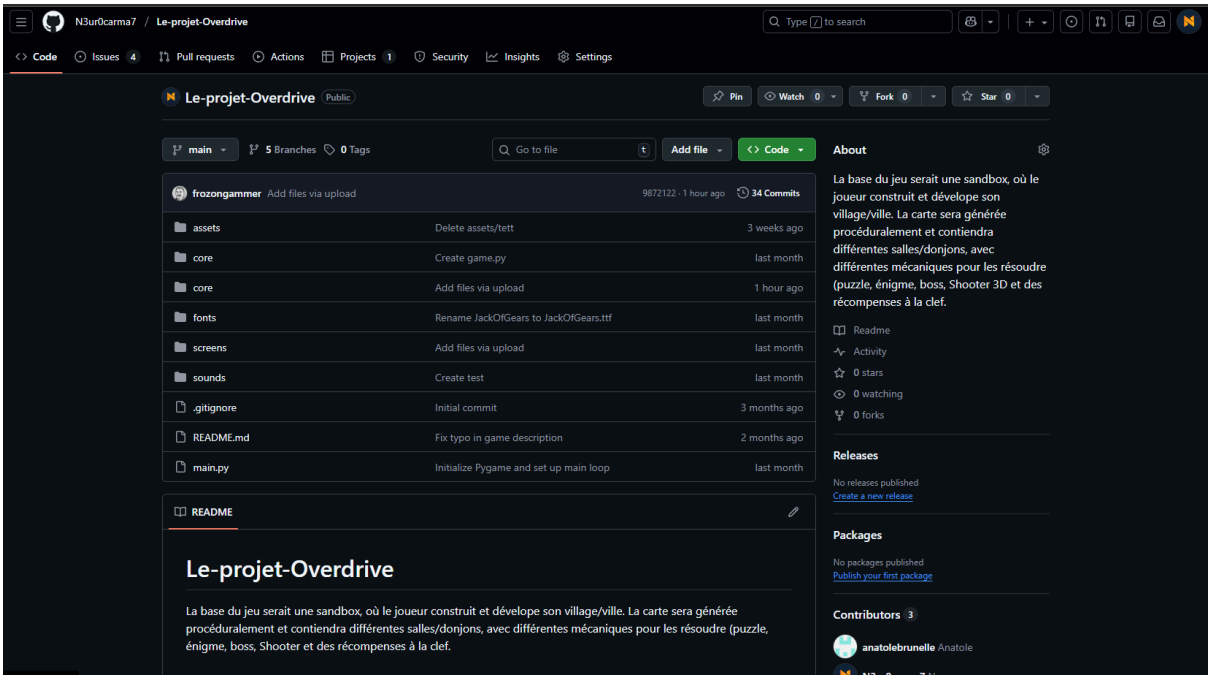
## 2.4 Matthieu Bourgeois

Mes principales tâches dans le projet sont :

- L'organisation du projet avec des outils externes
- L'implémentation des NPC et du joueur dans le jeu (affichage, contrôles)

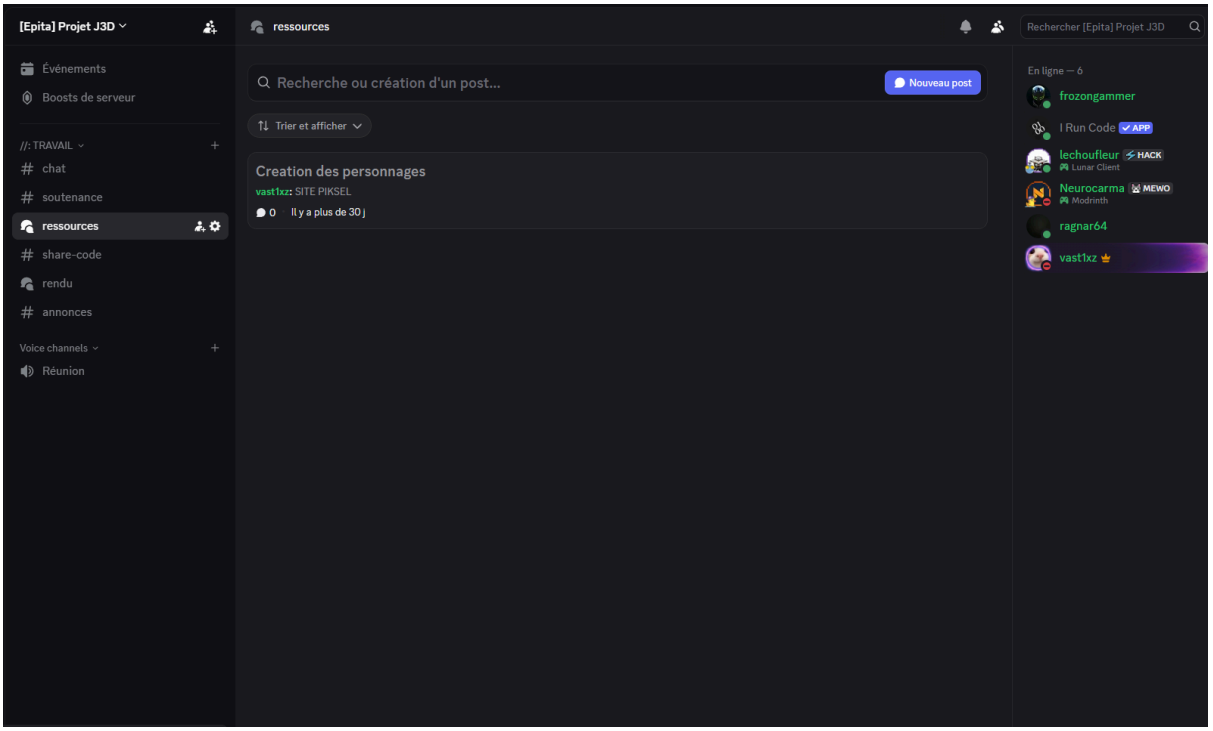
1. L'organisation

Pour répondre au besoin d'un projet organisé où tous les membres du groupe peuvent travailler sans déranger les autres et à plusieurs, j'ai choisi d'utiliser GitHub, une plateforme d'hébergement de code dont voici la page principale :

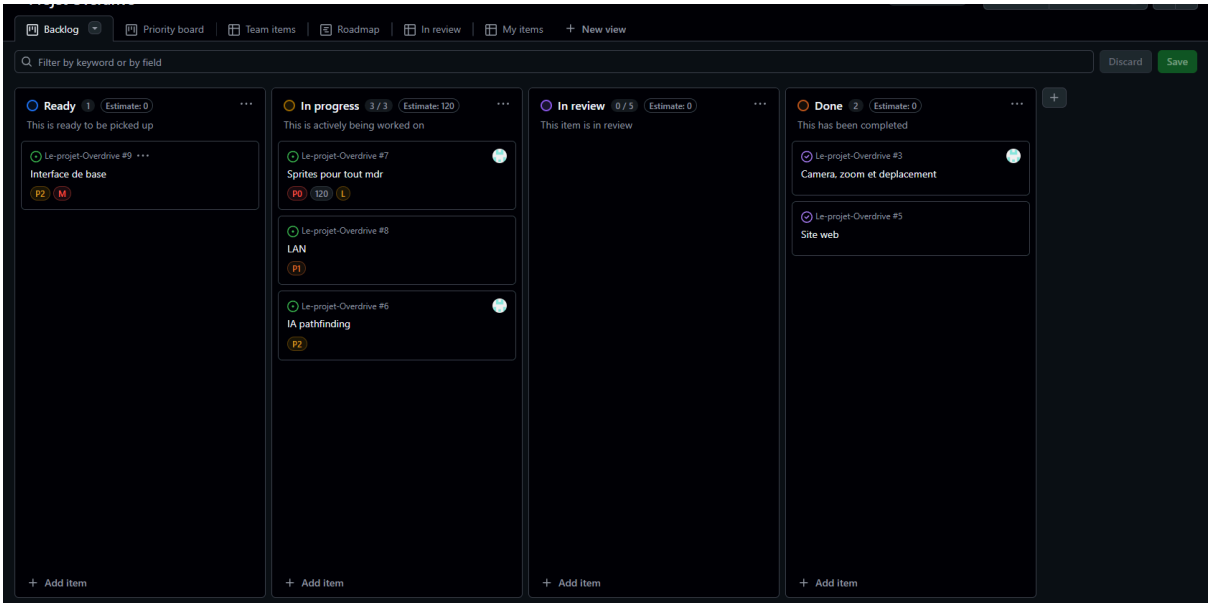



J'ai créé un dépôt GitHub dans lequel j'ai invité l'ensemble des membres du groupe et ajouté une description ainsi qu'une base de code. Cependant, un problème persiste : si plusieurs personnes veulent travailler en même temps, les changements que chacun fera risque de s'entremêler. La solution que j'ai choisie est d'utiliser les "branches" de GitHub qui permettent à chacun de travailler sur une copie du code principale et d'effectuer des modifications uniquement sur leur branche et non sur la principale. Une fois que les modifications ont été vérifiées et acceptées, les changements de la branche personnelle sont appliqués sur la branche principale.

Après avoir résolu un problème d'organisation au niveau de la programmation, il fallait s'occuper de la communication. Cela a été résolu en utilisant le réseau social Discord. J'ai créé un serveur Discord dans lequel j'ai mis plusieurs salons textuels qui sont renommés en fonction de leur utilité (exemple : le salon "annonce" afin de notifier l'ensemble des membres du groupe d'informations importantes ou bien le salon "ressources" afin de noter l'ensemble des ressources externes utilisés pour le projet). dont voici une image exemple ci-dessous :




Pour ce qui est des tâches à faire, des délais, etc… J'ai opté pour GitHub Projects, une solution simple et très utile qui se présente sous cette forme :




Cet outil nous permet de créer des tâches comme "Interface de base" par exemple, et de définir : qui travaille dessus, la priorité, la date objectif de fin, la taille de la tâche (représente aussi un peu la difficulté de celle-ci), ainsi que d'autres informations qui détaille ce qui est à faire dans la tâche.

Une fois la tâche créée, la/les personne/es assignée peuvent la prendre et la déplacer dans l'onglet "In Progress" afin de notifier l'ensemble du groupe que cette tâche est en cours de développement. Le reste est explicite.

Bien que ce n'est pas le rendu final de l'ensemble de nos tâches, cela représente bien notre façon de s'organiser en équipe.

2. Implémentation NPC/Joueur

Afin d'implémenter correctement les NPC et le joueur, j'ai choisi d'utiliser la POO (Programmation Orientée Objet) en utilisant des classes python ce qui permet de définir des stats différents pour chaque NPC/Joueur et d'avoir une structure de données qui est simple d'utilisation.

Le choix d'utiliser les classes pour l'implémentation nous permet de définir des stats tels que la vie, les dégâts, le niveau, l'expérience, etc.. pour le joueur.
Pour les NPC on a défini des dialogues qui seront stockés dans l'objet (le NPC) sous forme de liste de chaîne de caractères.

Cette partie de l'implémentation est encore loin d'être finie, notamment l'implémentation sur l'interface de PyGame, mais il faut attendre que la base de l'interface soit finie afin de pouvoir continuer sur ce point.

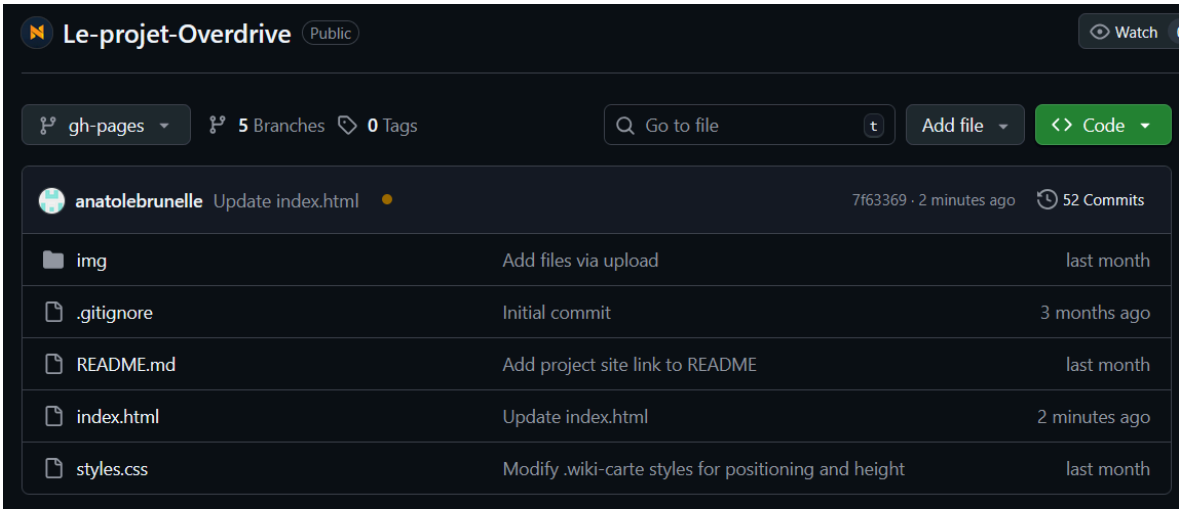
## 2.5 Anatole Brunelle

Mes principales tâches dans le projet sont :

- La création du site web
- La programmation de l'algorithme de path finding des villageois et des potentiels ennemis.
- La création des sprites avec Vasile

1. Le site web

Pour créer mon site web, j'ai choisi d'utiliser GitHub Pages, une solution gratuite proposée par GitHub qui permet d'héberger facilement des sites web statiques. Cette plateforme est particulièrement adaptée aux projets simples ou personnels, car elle ne nécessite pas de serveur complexe ni de base de données. Le site est directement publié à partir d'un dépôt GitHub, ce qui rend la mise en ligne rapide et efficace.

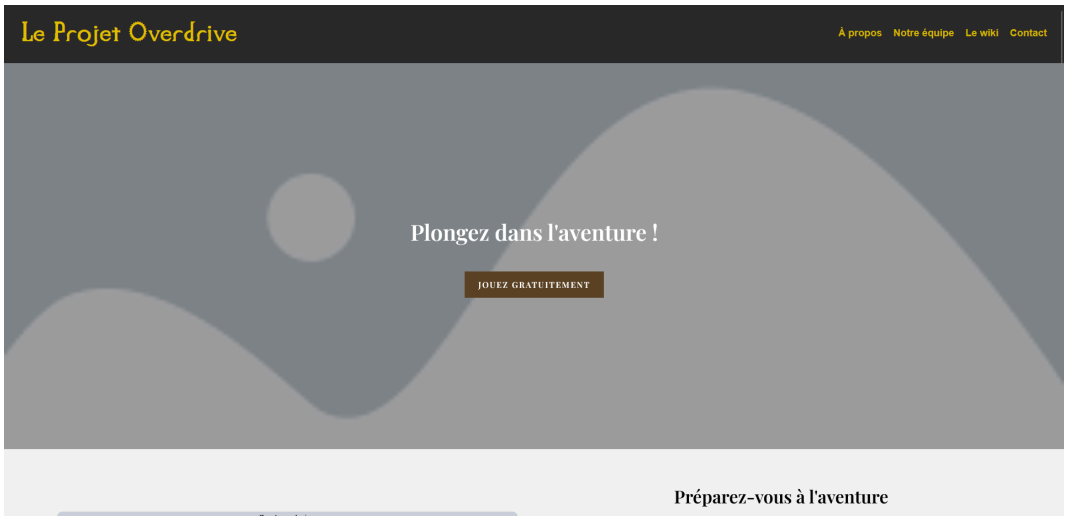



La première étape a été de créer un dépôt GitHub dédié à mon site. Ce dépôt contient tous les fichiers nécessaires, notamment le fichier principal *index.html*, qui correspond à la page d'accueil du site. GitHub Pages fonctionne automatiquement en détectant ce fichier et en l'affichant lorsque l'on accède à l'URL du site. Une fois le dépôt configuré et GitHub Pages activé dans les paramètres, le site est accessible en ligne en quelques minutes.

Le développement du site s'est fait uniquement avec HTML et CSS, ce qui m'a permis de bien comprendre les bases du développement web. Le HTML a servi à structurer le contenu du site, comme les titres, les paragraphes, les images et les liens. J'ai organisé la page de manière claire et logique afin de rendre la navigation simple et intuitive pour l'utilisateur.

Le CSS a été utilisé pour gérer l'apparence visuelle du site. Grâce à ce langage, j'ai pu personnaliser les couleurs, les polices, les marges et la disposition des éléments. J'ai

également utilisé le CSS pour rendre le site plus agréable à regarder et cohérent visuellement. Cela m'a permis de comprendre l'importance du design et de l'ergonomie dans un site web.




En complément du HTML et du CSS, j'ai intégré un peu de JavaScript directement dans le fichier HTML. Ce JavaScript permet d'ajouter une légère interactivité au site, par exemple pour réagir à un clic, afficher ou masquer des éléments, ou modifier du contenu dynamiquement. Même si son utilisation reste limitée, cela m'a permis de découvrir comment rendre une page web plus interactive et plus vivante.

Enfin, l'utilisation de GitHub m'a permis de gérer les versions de mon site grâce aux commits. À chaque modification importante, j'ai enregistré les changements, ce qui me permet de suivre l'évolution du projet et de revenir en arrière si nécessaire. GitHub Pages, combiné à HTML, CSS et un peu de JavaScript, constitue donc une solution simple, efficace et pédagogique pour créer et publier un site web en ligne.

## 2. Algorithme de pathfinding

Pour gérer les déplacements des entités dans notre jeu, nous avons développé un algorithme de pathfinding basé sur A*. Cet algorithme permet de calculer un chemin optimal entre un point de départ et une destination tout en tenant compte des obstacles présents dans l'environnement. Dans un jeu centré sur la construction et l'évolution d'une ville, où le décor change régulièrement en fonction des actions du joueur, il était essentiel de disposer d'un système de déplacement fiable et adaptable.

Le fonctionnement de l'algorithme A* repose sur une représentation du monde sous forme de grille ou de réseau de nœuds. Chaque position possible est analysée comme un nœud, relié à ses voisins directs. Lorsqu'une entité doit se déplacer, l'algorithme explore progressivement les nœuds disponibles afin de trouver le chemin le plus efficace vers la cible. Pour faire ce choix, A* utilise une combinaison de deux informations essentielles : le coût

réel du chemin déjà parcouru et une estimation de la distance restante jusqu'à la destination. Cette estimation, appelée heuristique, permet à l'algorithme de se diriger vers l'objectif sans parcourir inutilement l'ensemble de la carte.

À chaque étape du calcul, l'algorithme compare les différentes positions accessibles et sélectionne celle qui semble la plus prometteuse. Ce processus se répète jusqu'à ce que la destination soit atteinte ou qu'aucun chemin valide ne soit possible. Grâce à cette approche, A* parvient à trouver un chemin à la fois court et cohérent, tout en évitant les zones bloquées par des bâtiments ou d'autres obstacles. Le résultat est un déplacement fluide et naturel, essentiel pour maintenir une expérience de jeu agréable.

Ce choix s'est imposé comme le plus pertinent dans notre cas, car le monde du jeu est en constante évolution. Les joueurs peuvent construire, modifier ou agrandir leur ville à tout moment, ce qui modifie directement les chemins disponibles. L'algorithme A* s'adapte facilement à ces changements, car il recalcule les chemins en fonction de l'état actuel de la carte. Contrairement à des solutions plus simples, il permet d'éviter les détours inutiles et garantit une navigation efficace même dans des environnements complexes.

De plus, A* offre un excellent équilibre entre performance et précision. Dans un jeu au rythme volontairement calme, il est important que les déplacements des entités paraissent intelligents sans pour autant consommer trop de ressources. L'utilisation d'une heuristique permet de limiter le nombre de positions analysées, ce qui rend l'algorithme suffisamment rapide pour gérer plusieurs entités en même temps. Cela contribue à préserver la fluidité du jeu, même lorsque la ville devient plus grande et plus dense.

L'algorithme est utilisé aussi bien pour les déplacements des habitants que pour certaines phases d'exploration ou d'interactions avec l'environnement. Il garantit que chaque entité trouve un chemin logique vers sa destination, tout en respectant l'organisation de la ville créée par le joueur. Les créatures hostiles, bien que secondaires dans le gameplay, utilisent également ce système lorsqu'elles interagissent avec l'environnement, ce qui assure une cohérence globale dans les déplacements.

Enfin, l'intégration de l'algorithme A* nous offre une base solide pour l'évolution future du jeu. Il pourra être amélioré ou étendu, par exemple en ajoutant différents coûts de déplacement selon le type de terrain ou en optimisant les calculs lors des déplacements hors ligne. Ce système de pathfinding joue donc un rôle clé dans la crédibilité et le confort de jeu, en soutenant une expérience à la fois fluide, cohérente et adaptée à un jeu de construction coopératif et relaxant.

3. Graphismes

Concernant les graphismes du jeu, cette partie a été réalisée principalement par Vasile, qui possède plus d'expérience dans ce domaine. De mon côté, je débute en graphisme,

mais j’ai tout de même participé au processus en effectuant des recherches et en commençant à créer certains éléments visuels.

Pour apprendre, je me suis renseigné sur les bases du graphisme pour le jeu vidéo, notamment sur le choix des couleurs, la lisibilité des éléments à l’écran et la cohérence visuelle entre les différents composants du jeu. J’ai également observé des styles graphiques existants afin de m’en inspirer et de mieux comprendre comment créer des visuels adaptés à l’ambiance du jeu.

J’ai ensuite commencé à réaliser mes premiers éléments graphiques à l’aide d’outils en ligne simples et accessibles. Par exemple, le site Piskel permet de créer facilement des sprites et des éléments graphiques en pixel art, ce qui est particulièrement adapté pour les jeux indépendants. Ce travail progressif me permet d’acquérir de nouvelles compétences tout en contribuant, à mon niveau, à l’identité visuelle du projet.