Document du projet :

Ce projet consiste à développer un jeu de rôle en mode texte en utilisant le langage **Rust**. L'objectif principal était d'implémenter un système de **simulation logique du monde physique** où le joueur peut **explorer un univers, interagir avec des personnages non-joueurs (PNJ), accomplir des quêtes, combattre des ennemis, et gérer un inventaire dynamique**.

2. Fonctionnalités principales

Le jeu comprend les fonctionnalités suivantes :

2.1 Création et gestion du personnage

- Chaque joueur commence avec un personnage ayant les attributs suivants :
    - **Santé**
    - **Force**
    - **Agilité**
    - **Intelligence**
- Possibilité de voir les statistiques du personnage.
- Le personnage peut **monter de niveau** en accumulant de l'expérience (XP).

2.2 Exploration du monde

- Différentes zones ont été implémentées :
    - **Village** (PNJ, quêtes, récupération de santé)
    - **Forêt** (trésors, objets, ennemis)
    - **Marais Toxique** (perte de santé progressive)
    - **Temple Mystique** (accessible avec un objet spécifique)
- Les zones peuvent contenir des **objets à ramasser**, des **PNJ à qui parler**, et des **quêtes à accomplir**.

2.3 Interactions avec l’environnement

- Possibilité de **trouver des objets** (potions, armes, armures).
- **Récupération automatique** de la santé dans certaines zones (ex: Village).
- **Effets environnementaux** comme la perte de points de vie dans des zones toxiques.

2.4 Système de quêtes et PNJ

- Les PNJ offrent des quêtes que le joueur peut accepter et compléter.
- Ex: **Le sage du village** donne une quête permettant d'obtenir la **Clé dorée** pour accéder au **Temple Mystique**.
- Les récompenses peuvent être des **objets** ou de l'**expérience**.

2.5 Système de combat

- Combat **au tour par tour** contre des ennemis.
- Le joueur peut **attaquer normalement** ou utiliser une **attaque spéciale**.
- **Les ennemis ont des attaques aléatoires**, y compris des attaques spéciales.
- À la fin d'un combat, le joueur gagne de l'XP et peut monter de niveau.

2.6 Gestion de l’inventaire et des équipements

- Possibilité de **ramasser des objets et de les stocker**.
- Système d’équipement :
    - **Armes** : Augmentent les dégâts en combat.
    - **Armures** : Réduisent les dégâts subis.
- Commande pour **équiper une arme ou une armure**.

3. Organisation du code

Le projet est organisé en plusieurs modules :

- **character.rs :** Gestion du personnage (statistiques, inventaire, XP, niveaux, équipements).
- **world.rs** : Gestion des zones et de l'exploration.
- **npc.rs :** Gestion des interactions avec les PNJ.
- **quest.rs :** Système de quêtes.
- **combat.rs** : Système de combat et résolution des attaques.
- **event.rs :** Gestion des événements aléatoires.
- **item.rs :** Gestion des objets, des armes et des armures.

4. Planning et répartition des tâches

L’équipe de développement était composée de **4 développeurs**. Voici le **planning** et la **répartition des tâches** :

Sprint 1 : Mise en place du projet et création du personnage

- **Développeur 1 :** Initialisation du projet, configuration de GitHub.
- **Développeur 2 :** Définition de la structure du personnage et des attributs.
- **Développeur 3 :** Gestion de l’affichage des statistiques et de la modification des attributs.
- **Développeur 4 :** Sauvegarde et chargement du personnage dans un fichier JSON.

Sprint 2 : Exploration du monde

- **Développeur 1 :** Création des différentes zones.
- **Développeur 2 :** Implémentation du déplacement entre les zones.
- **Développeur 3 :** Ajout des effets des zones (ex: perte de santé dans le marais toxique).

- **Développeur 4 :** Gestion des objets trouvés dans chaque zone.

Sprint 3 : Interactions et mécanique avancée

- **Développeur 1 :** Ajout des PNJ et des dialogues.
- **Développeur 2 :** Implémentation du système de quêtes.
- **Développeur 3 :** Ajout du système de combat avec attaques et ripostes.
- **Développeur 4 :** Gestion de l'inventaire et possibilité d'équiper des objets.

Sprint 4 : Optimisation et finalisation

- **Développeur 1 :** Ajout d'événements dynamiques (coffres, attaques surprises, etc.).
- **Développeur 2 :** Tests unitaires et correction des bugs.
- **Développeur 3 :** Optimisation du code et gestion mémoire.
- **Développeur 4 :** Rédaction de la documentation et rapport final.