Rapport général - 3 : Projet long

GH-02

Sommaire

I - Introduction

L’objectif de ce projet est de **concevoir** et développer un jeu vidéo **2D** du type **top-down RPG** en **monde ouvert**. Ce jeu combine **exploration**, **combat**, et **progression**.

Le joueur incarne un personnage **évoluant** dans le monde ouvert, où certaines **zones** sont initialement **verrouillées**. Pour les débloquer, il devra **explorer** l’environnement, **collecter** des composants et des ressources, et **vaincre** des ennemis dans des combats stratégiques.

Après chaque **victoire**, le personnage **gagne** en **puissance** (force, points de vie, meilleur équipement et meilleures compétences), ce qui lui permet **d’accéder** à des **zones** plus **difficiles**. Cependant, en cas de **défaite**, le personnage **régresse** à un état précédent, obligeant le joueur à **réexplorer** et à se renforcer à nouveau. Cette mécanique ajoute une **dimension tactique** et encourage une **gestion** prudente des ressources et de la prise de **combat**.

Le jeu se "termine" avec un **combat final** contre un boss puissant, qui teste toutes les compétences acquises par le joueur, bien que le **but ultime** est uniquement de **devenir** encore **plus fort** et que ce boss ne signe que la fin de la storyline.

II - Principales fonctionnalités

1. Exploration libre du terrain en open world :

a. Besoin : Permettre au joueur de se déplacer librement dans une vaste carte avec des régions délimitées.

b. Service rendu : Immersion dans le monde du jeu et découverte des différents environnements et de leurs spécificités.

2. Zones verrouillées et expansion du terrain connu :

a. Besoin : Certaines zones de la map sont initialement verrouillées et nécessitent la collecte de composants spécifiques pour être débloquées.

b. Service rendu : Progression dans le jeu et sentiment d’accomplissement / d'amélioration pour le joueur.

3. Collecte de composants et de ressources :

a. Besoin : Le joueur doit explorer le terrain pour trouver des objets nécessaires à l'expansion du monde et à l'amélioration de son personnage et de ses armes.

b. Service rendu : Ajout d'un objectif "secondaire" le long de l'expérience et d'un défi pour le joueur.

4. Système de combat :

a. Besoin : Le joueur doit affronter des ennemis dans chaque nouvelle zone pour progresser / obtenir différentes ressources.

b. Service rendu : Ajout de défis et de diversité dans la collecte de ressource et la progression.

5. Progression du personnage :

a. Besoin : Après chaque victoire, le personnage devient plus fort (augmentation des capacités) et obtient de l'équipement de meilleure qualité.

b. Service rendu : Permettre la répétabilité du jeu et la diversité des profils suivant les compétences et les qualités améliorées.

6. Régression en cas d'échec :

a. Besoin : Si le joueur perd un combat, il retourne à un état précédent (sûrement moins fort) et doit récupérer des ressources / re-combattre certains ennemis pour se renforcer à nouveau.

b. Service rendu : Ajout d'un enjeu et d'une stratégie à adopter pour éviter les échecs.

7. Combat final contre un boss :

a. Besoin : À la fin du jeu, le joueur affronte un boss puissant qui teste toutes ses compétences acquises.

b. Service rendu : Point culminant du jeu, offrant un défi majeur et une récompense significative avant un ajout de prochaines quêtes principales.

8. Quêtes ou objectifs secondaires :

a. Besoin : Proposer des quêtes ou des défis supplémentaires pour enrichir l'expérience de jeu.

b. Service rendu : Variété et durée de vie prolongée du jeu.

9. Système de sauvegarde :

a. Besoin : Permettre au joueur de sauvegarder sa progression après chaque victoire ou étape importante.

b. Service rendu : Sécurité pour le joueur et évolution fluide du jeu.

III - Découpage de l'application

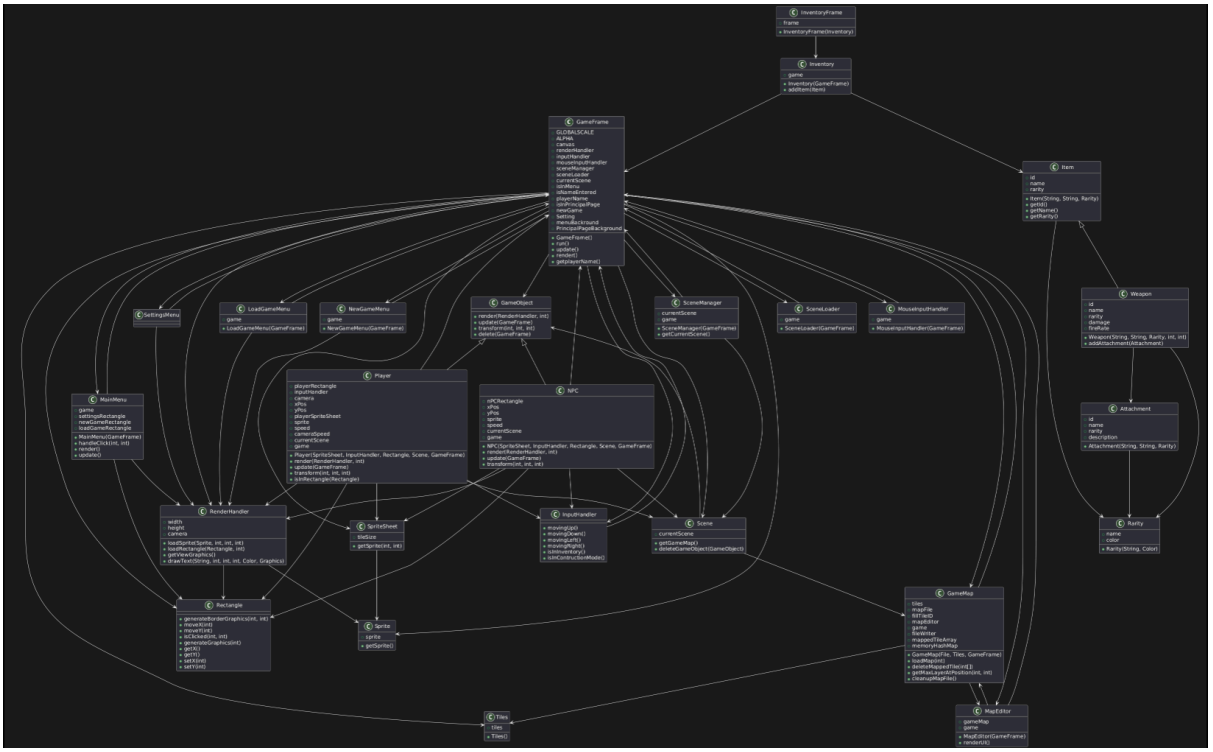
Différents paquetages ont été faits afin de découper les fichiers de manière efficace.

Nous avons :

Paquetage	Classes	Fonctions
Assets		Contient des fichiers .png avec des blocs des images utilisées pour faire le jeu.
Entities	• GameObject • NPC • Player	Gestion des personnages, tels que le joueur, ennemi et personnages non joueurs.
Handlers	• InputHandler • MouseInputHandler • RenderHendler	Gestion des inputs du clavier et de la souris.
Inventory	• Attachment • Grenade • HealItem • Inventory • InventoryFrame • Item • Rarity • Weapon	Tout ce qui est en lien avec les objets et l'inventaire du joueur.
Map	• GameMap • MapEditor • Tiles	Dossier qui permet de modifier, créer, supprimer des éléments sur

		la carte du jeu.
Mapfile		Dossier qui est lié à toutes les scènes, et un fichier qui donne un ID à tous les objets.
Menus	• LoadGameMenu • MainMenu • NewGameMenu • SettingsMenu	Permet de gérer l'affichage du menu principal et des différents onglets.
Scene	• Scene • SceneLoader • SceneManager	Permet de charger les différentes scènes, tels que les niveaux ou encore le menu d'accueil.
Sprite	• Sprite • SpriteSheet	Permet le traitement des images du module assets

IV - Diagrammes de classe



V - Dynamique du système

1. Cycle principal du jeu :

- Le jeu utilise un pattern de cycle de jeu classique avec deux méthodes principales :
  - update() : met à jour le jeu à une vitesse fixe (60 fois par seconde)
  - render() : rend le jeu à une vitesse dépendante du matériel
- Le GameFrame orchestre tout le système.

2. Gestion des entrées :

- InputHandler et MouseInputHandler gèrent les entrées utilisateurs
- Les touches ZQSD/Flèches contrôlent le joueur
- Les touches spécifiques gèrent :
  - L'inventaire
  - Le mode construction
  - Les menus

3. Gestion des scènes :

- SceneManager gère les différentes scènes
- Le jeu peut être dans différents états :
  - Menu principal
  - Page principale
  - Jeu
  - Mode construction
  - Inventaire

4. Système de jeu :

- Le joueur (Player) peut :
  - Se déplacer sur la carte
  - Interagir avec les objets
  - Utiliser des armes
  - Gérer son inventaire
  - Construire (mode construction)

5. **Système d'inventaire :**
    - Les items sont gérés par une hiérarchie :
        - Item (classe de base)
        - Weapon (arme)
        - Attachment (accessoires pour les armes)
    - Chaque item a une rareté (Rarity)

6. **Gestion de la carte :**
    - GameMap gère la carte avec plusieurs couches
    - MapEditor permet d'éditer la carte
    - La carte est sauvegardée dans des fichiers

7. **Cycle de vie :**
    - Le jeu démarre avec le menu principal
    - Le joueur peut créer un nouveau jeu ou charger une partie
    - En jeu, il peut :
        - Explorer la carte
        - Combattre
        - Gérer son inventaire
        - Construire
    - À la fermeture, le jeu nettoie les fichiers de carte

8. **Gestion des ressources :**
    - Les images sont chargées et converties
    - Les sprites sont gérés par SpriteSheet
    - Les textures sont optimisées pour le rendu

VI - Principaux choix de conception

Nous avons choisi de travailler **sans bibliothèque** ni **librairie externe** pour avoir un **contrôle** total sur l'**architecture** et les **mécanismes** du jeu. L'application est structurée en plusieurs **paquetages** spécialisés (assets, entities, handlers, inventory, map, mapfile, menus, scène, sprite), chacun étant dédié à un aspect **fonctionnel** précis du jeu (création de la map, gérer les inputs, personnages …). La carte est découpée en **tuiles carrés** formant une **grille**. Chaque tuile peut contenir un élément graphique appartenant à une **couche** spécifique ce qui permet de **superposer** plusieurs éléments (background, obstacles, décorations) selon l'**identifiant** de la couche. De plus, le personnage peut interagir avec l'**environnement** sous certaines conditions (présence d'un obstacle par exemple) en fonction des **entrées** clavier du joueur.

Problèmes rencontrés et solutions:

- **Problème** : difficultés à gérer le principe de superpositions des éléments et des collisions avec les obstacles.

  Solution : Introduction de la notion de couche par indice de priorité : il n'est pas possible de superposer un élément d'indice de couche supérieur ou bien pour le joueur de traverser une tuile ayant un indice de couche supérieur à celui de la couche de fond)
- **Problème** : difficultés dans la mise en place d'un affichage du menu principal adapté pour toute résolution

  Solution : rendre les tailles dépendantes du viewport et adapter la taille à chaque frame.

VII - Organisation de l'équipe

Epic	Membres
Menu	Hocine Mediani
Handler des événements	Hocine Mediani/Benjamin Krief/Ilian Kraifi
Carte/éléments graphiques	Hocine Mediani/Benjamin Krief/Ilian Kraifi/Achraf Al Oumami
Personnages	Hocine Mediani/Benjamin Krief/Ilian Kraifi
Inventaire	Othmane Zenbi/Medkhalil Abdellaoui
NPC	Achraf Al Oumami
Sprites	Hocine Mediani/Ilian Kraifi

VIII - Conclusion

Pour conclure, ce projet nous a permis de découvrir les bases du développement d'un jeu vidéo en Java, et la complexité d'en développer un. Parmi tous les objectifs que nous voulions, certains n'ont pas pu être finis, voire commencés, tels que le système d'inventaire ou encore le système de combat.