Rapport intermédiaire pour "Projet de programmation" Date d'échéance : 15 mars 2025

Guillaume Xue et David Yu

15 mars 2025

1 Introduction

Le but de ce projet est de développer une version personnalisée de *Pokémon Donjon Mystère* en OCaml en mettant l'accent sur la génération procédurale de cartes, l'affichage des éléments du jeu et les interactions du joueur.

1.1 Métriques de succès

Le succès du projet sera mesuré à travers les critères suivants :
— La génération procédurale des donjons de façon cohérente et diversifiée.
— Fonctionnalités du menu (création et chargement de partie).
— Bonne gestion des collisions et animation du joueur.
— Stabilité et performance du jeu.
— Organisation en modules bien structurés.
— Code réutilisable et bien documenté.

2 Implémentation

2.1 Développement et organisation du code

— **Module Génération de cartes** : Implémente les algorithmes (automate cellulaire, Prim, Flood fill, Voronoi).
— **Module Affichage** : Gère le rendu graphique du menu, du joueur et des biomes.
— **Module Interaction** : Assure les déplacements du joueur et les collisions avec l'environnement.
— **Module Gestion des données** : Stocke les statistiques du joueur et les sauvegardes de partie.

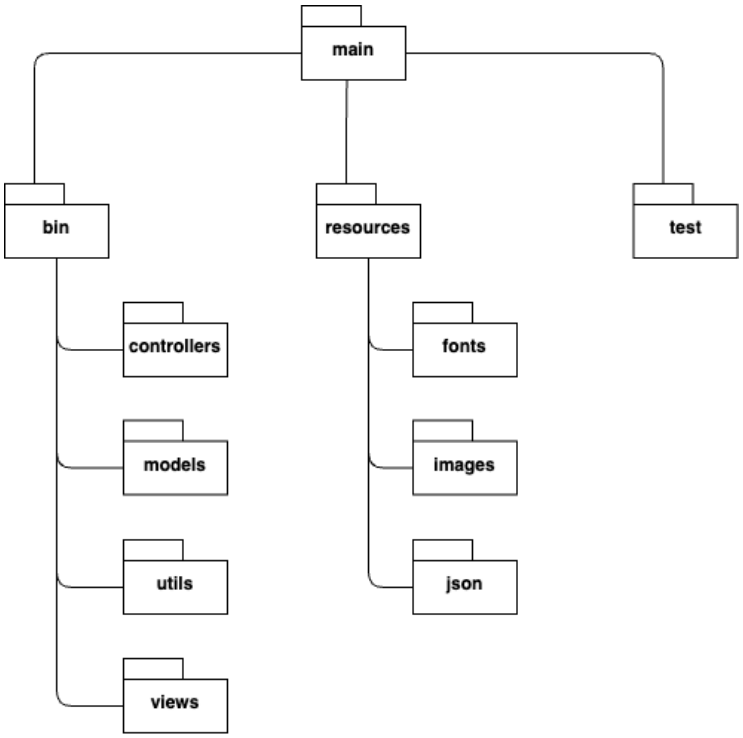

FIGURE 1 – Schéma de la structure du projet

2.2 Structure des données

Dans notre projet, nous avons structuré les données en plusieurs catégories pour organiser les informations essentielles du jeu :

— Une structure pour la carte, stockée sous forme de grille de tuiles.
— Une structure pour le joueur, avec position, statistiques et apparence.
— Un regroupement des données dans une structure globale pour l'état du jeu.
— Un système de conversion JSON <-> OCaml pour gérer les sauvegardes.

Nous utilisons des types algébriques, des enregistrements et des structures adaptées à OCaml pour gérer ces éléments de manière efficace.

2.3 Technologies utilisées

Le projet repose sur les technologies suivantes :

— Langage : OCaml
— Algorithmes : Automate cellulaire, algorithme de Prim, Flood fill, Voronoi
— Bibliothèques graphiques utilisées : Raylib

3 Jalons

3.1 Avancement actuel

Les tâches déjà accomplies sont :
— Génération procédurale de la carte.
— Affichage du menu principal.
— Affichage du joueur et des animations.
— Affichage des biomes.
— Gestion des collisions.
— Affichage des statistiques du joueur.
— Système de sauvegarde.

3.2 Prochaines étapes

Les tâches à venir sont :
— Génération des ennemis, obstacles et des objets.
— Système de combat.
— Système d'inventaire.
— Génération des boss.
— Amélioration de l'interface graphique.

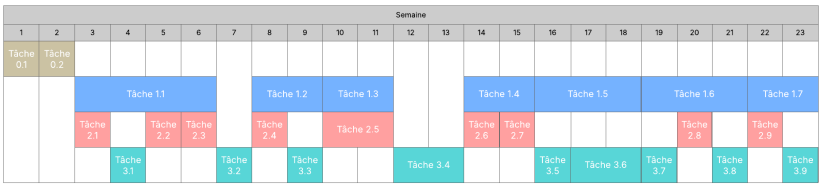

FIGURE 2 – Diagramme de Gantt

— Tâche 0.1 : Définition des algorithmes de génération de carte.
— Tâche 0.2 : Creation de la structure du projet.
— Tâche 1.1 : Automate cellulaire.
— Tâche 1.2 : Algorithme de Prim, Flood fill.
— Tâche 1.3 : Algorithme de Voronoi.
— Tâche 1.4 : Génération des obstacles.
— Tâche 1.5 : Génération des ennemis.
— Tâche 1.6 : Génération des objets.
— Tâche 1.7 : Génération des boss.
— Tâche 2.1 : Affichage de la carte et des biomes.
— Tâche 2.2 : Affichage du menu principal.
— Tâche 2.3 : Affichage du menu de chargement nouvelle carte.
— Tâche 2.4 : Affichage du menu de sauvegarde.
— Tâche 2.5 : Affichage du joueur avec camera centrer.
— Tâche 2.6 : Affichage des obstacles.
— Tâche 2.7 : Affichage des ennemis.
— Tâche 2.8 : Affichage des objets et inventaire.
— Tâche 2.9 : Affichage des boss.
— Tâche 3.1 : structure du joueur.
— Tâche 3.2 : système de chargement.
— Tâche 3.3 : système de sauvegarde.
— Tâche 3.4 : déplacement, collision, statistiques du joueur.
— Tâche 3.5 : structure des ennemis.
— Tâche 3.6 : IA des ennemis.
— Tâche 3.7 : système de combat.

— Tâche 3.8 : système d'inventaire.
— Tâche 3.9 : système d'objets.