Introduction

(Guillaume)

Recréer un jeu riche et immersif en Ocaml ? C'est le défi que nous avons relevé. Notre projet est une réinterprétation de Pokémon Donjon Mystère, codée entièrement dans un langage fonctionnel.

Les jeux sont traditionnellement codes en langages orientes objet comme Java ou C#. Ces approches permettent une gestion claire des entités du jeu – ennemis, objets, cartes – sous forme de classe et d'héritages.

Mais en Ocaml, on adopte une vision radicalement diffèrent : chaque fonctionnalité repose sur des fonctions pures, et types algébriques, et des structures immuables. Cela renforce la fiabilité du code et facilite les tests.

Allons voir maintenant notre jeu.

Contenue

(David)

Le joueur peut choisir entre charger une partie existante ou créer une nouvelle aventure.

En démarrant une nouvelle partie il choisit un nom pour sa sauvegarde et sélectionne un starter parmi plusieurs Pokémon disponibles.

Le joueur explore un donjon, il progresse étage après étage, combat des ennemis, ramasse des objets, et tente de survivre.

La carte est générée procéduralement, diffèrent du jeu original qui utilise des salles rectangulaires, nous avons voulu utiliser des automates cellulaires pour créer des salles uniques.

Le système de combat est au tour par tour. Chaque attaque inflige des dégâts spécifiques a des effets de type.

Il n'y a pas que des ennemies qui peuvent vous inflige est dégâts mais aussi des pièges invisibles, jusqu'à ce que vous marchiez dessus ils peuvent vous infliger des dégâts, modifier votre positon, déclencher des effets spéciaux

(Guillaume)

Le joueur peut utiliser des objets pour se soigner, gérer sont inventaire et anticiper les déplacements des ennemis, qui sont gérés par une IA réactive.

Pour l'immersion du joueur, nous avons un système de shadow casting, il ne voit que ce que son personnage peut réellement percevoir. Les ennemis et les objets sont masques tant qu'ils ne sont pas dans son champ de vision.

Et pour renforce l’immersion, non seulement on a la partie visuel et textuel des interactions du joueur et du jeu, mais aussi des effets sonores, et une musique de fond.

Conclusion

(David)

Ce projet prouve qu’un jeu complet : génération procédurale, IA, combat, vision, gestion de sauvegardes, peut être conçu dans un langage purement fonctionnel comme Ocaml.