Acting Shooting Star : Notre version de Crossy Road basé sur le modèle des acteurs

Enzo Picarel, Raphaël Bely, Arno Donias, Thibault Abeille
Encadrants : Vincent Alba, David Renault
ENSEIRB-MATMECA – 2025

6 mai 2025

Table des matières

1 Introduction

Dans le cadre de notre projet de programmation, nous avons choisi de recréer une version personnalisée du jeu *Crossy Road*, un jeu mobile populaire de notre enfance. Ce choix s'est imposé naturellement : d'une part pour son aspect ludique et familier, et d'autre part parce que ses mécaniques simples et répétitives se prêtaient bien à une première implémentation de projet en JavaScript (ou ici, plus précisément en TypeScript).

L'objectif principal du projet était de concevoir un moteur de jeu basé sur le **modèle des acteurs**, dans lequel chaque élément du jeu (joueur, voitures, rivières, etc.) est représenté par un acteur autonome. Chaque acteur devait pouvoir recevoir des messages, réagir en fonction de son état, et potentiellement émettre de nouveaux messages à d'autres acteurs. Une attention particulière a été portée à la **pureté fonctionnelle** de l'implémentation, en évitant les effets de bord et en respectant les principes de la programmation fonctionnelle autant que possible.

Le projet s'inscrivait dans un cadre pédagogique défini, avec des contraintes précises : utilisation de `TypeScript`, affichage dans le *terminal* via le *terminal-kit*, validation du code avec `ESLint`, tests unitaires avec `Jest`, et développement collaboratif via un dépôt `git`. Le code devait également respecter une structure imposée du dépôt.

Parmi les axes de développement retenus, on peut citer la gestion des acteurs et de leurs interactions, la génération dynamique du monde de jeu avec une difficulté croissante, ainsi que des fonctionnalités plus avancées comme le **retour dans le temps**. Enfin, ce projet s'insérait dans une démarche académique plus large, en lien avec un enseignement sur **la programmation fonctionnelle, TypeScript et l'analyse statique de code**.

2 Modèle des acteurs et principes de conception

— Rappel du modèle des acteurs
— Justification de l'approche transactionnelle
— Fonctionnement du runtime : messages, tick, mise à jour
— Intérêts pour la programmation concurrente et déterministe

3 Architecture du jeu

— Types d'acteurs : joueur, voitures, bûches, rivières, arbres
— Gestion de la position et boîte aux lettres
— Génération procédurale des lignes et des obstacles
— Évolution de la difficulté
— Gestion des collisions
— Création dynamique d'acteurs

4 Implémentation technique

— Organisation des fichiers et dépendances
— Utilisation de `TypeScript`, `terminal-kit`, `Jest`, `ESLint`
— Extraits de code clefs (`make_actor`, runtime, génération de niveaux)

5 Affichage et interactions

— Affichage ASCII dans le terminal
— Contraintes liées à `terminal-kit`
— Possibilités d'interactions (touches clavier, etc.)

6 Difficultés rencontrées et solutions apportées

— Collisions sans effets de bord
— Identification des acteurs et passage de tick
— Débogage et visualisation du comportement

7 Évaluation

— Fonctionnalité et stabilité
— Performances en terminal
— Extensibilité du moteur de jeu

8 Conclusion et perspectives

Résumé des apports du projet, limites actuelles, et pistes d'amélioration (graphismes, IA, multijoueur, temps réel).

Références

— Documentation de `terminal-kit` : `https://github.com/cronvel/terminal-kit`
— Tutoriel sur le modèle des acteurs
— Documentation ESLint / Jest / TypeScript