Projet de mathématiques appliquées

2048

Cloé Quirin *&* *Vincent* Scavinner

supervisé par
Miguel Martinez

May 30, 2021

Abstract

Rapport de projet documentant la réflexion et les choix d'implémentation réalisés dans le cadre de la conception d'une version revisitée du jeu 2048, utilisant des lois de probabilités mathématiques.

Introduction

Cela fait maintenant quelques années que le jeu du 2048 a pris d'assaut Internet. Dans le monde entier, des milliers de personnes ont passé des millions d'heures à essayer de créer la tuile 2048.

Outre le côté addictif du jeu, il présente également une possibilité d'explorer les mathématiques. Dans le cadre d'un projet réalisé au cours d'une seconde année d'ingénieur à l'IMAC (Univeristé Gustave Eiffel, Champs-sur-Marne, France), ce document tente de démontrer la manière dont les théories et lois probabilistiques ont été appliquées au jeu original pour l'étendre et le renouveler dans une version revisitée.

Règles

Le 2048 est un jeu *"puzzle block"* développé par Gabriele Cirulli. C'est un jeu joué sur une grille 4x4 avec des tuiles numérotées 2^n où n représente un nombre naturel. L'objectif du jeu est de combiner des tuiles du même nombre pour finalement former le nombre 2048.

L'utilisateur peut se déplacer dans les quatre directions cardinales et après chaque mouvement, une nouvelle tuile est générée aléatoirement dans la grille qui est numérotée 2 ou 4. Un mouvement est considéré comme "légal" si au moins une tuile peut être glissée dans un emplacement vide ou si les tuiles peuvent être combinés dans la direction choisie. Le jeu se termine lorsque l'utilisateur n'a pas de mouvement "légal" à gauche.

Aspect mathématique de la version classique

Après avoir fait un mouvement, une nouvelle tuile est placée sur le plateau. Celle-ci est placée aléatoirement sur un emplacement vide de la grille. Cette nouvelle tuile peut avoir pour valeur un 2 ou un 4, là encore choisi aléatoirement. La valeur 2 a 90% de chance de tomber, tandis que la valeur 4 en a seulement 10. Le jeu continue ensuite jusqu'à ce qu'il n'y ait plus de mouvements possibles.

Outre cet aspect probabiliste, le coeur du jeu et de son intéractivité repose sur une combinaison de transformations matricielles relative à la grille. Néanmoins, il est possible de procéder à l'ensemble des mouvements nécessaires par un enchaînement de rotations matricielles. C'est ce que nous avons choisi de faire dans le cadre de notre implémentation.

Les composantes aléatoires utilisées

Le type de tuile

Espace d'état : Choix du type de tuile.

Technologies

Vue/TypeScript

Nous avons décidé d'utiliser le *Framework Vue* mais dans une version typée grâce au langage de programmation *TypeScript*. Nous avons complété l'implémentation en utilisant du développement objet.

Vue (prononcé comme le terme anglais view) est un *framework* évolutif pour construire des interfaces utilisateur. À la différence des autres *frameworks* existants, *Vue* a été conçu et pensé pour pouvoir être adopté de manière incrémentale. Le cœur de la bibliothèque se concentre uniquement sur la partie vue/*front*, et il est vraiment simple de l'intégrer avec d'autres bibliothèques ou projets existants.

Contrairement à *JavaScript*, le code écrit en *TypeScript* est beaucoup plus fiable et facilement refactorable. Cela permet d'éviter les erreurs et de procéder à de la réécriture plus facilement. La mise en place de types permet d'éviter des erreurs classiques et redondantes dans un code JavaScript classique, et permet de se concentrer directement dessus afin d'éviter leur accumulation et le risque de bug majeur.

TypeScript permet également de s'intéresser plus en profondeur à la manière dont le système fonctionne vraiment. Il permet de rendre abstrait des parties récurrentes afin de les réutiliser.

Néanmoins, même si on peut facilement défendre l'idée d'utiliser *TypeScript* dans un projet qui s'étend dans le temps, il n'en est pas de même pour un projet à court terme comme le nôtre. En effet, typer correctement un projet demande beaucoup de temps, beaucoup de minutie et beaucoup

de réflexion.

Librairies personnalisées

Nous avons fait le choix de créer notre propre librairie mathématique, chargée de nous fournir facilement les distributions et lois dont nous avions besoin. Cela nous a permis de réaliser un couplage faible avec le reste de notre jeu, mais aussi de préparer l'éventualité d'une évolution de l'application avec de nouvelles probabilités. Notre jeu possède donc son propre *Manager*, chargé de faire les appels à la librarie.

En plus des probabilités, le jeu repose également sur des transformations matricielles. Nous avons alors créé notre propre fonction utilitaire afin de réaliser ces transformations le plus simplement possible. Cette fonction est générique.

Au cours d'une partie, les structures de données chargées de stocker les états relatifs au différentes cases (couramment en jeu ou celles déjà jouées) peuvent potentiellement atteindre des tailles importantes. Pour optimiser les recherches, nous avons décidé de réécrire la fonction de filtre (*Array.filter()*) sur tableaux de *JavaScript*. Après quelques tests, notre fonction s'avère en moyenne 60% plus rapide. Nous en avons aussi profiter pour écrire une fonction de filtre mais applicable à des objets. Pour cela, nous avons mis en place le principe de *currying*.

Conclusion

Nous espérons avoir réussi à montrer que derrière un simple jeu, il y a un monde mathématique à explorer. Les concepts étudiés ce semestre et les ressources consultées au cours du projet nous ont aidés à analyser le jeu et son fonctionnement, mais aussi à l'aborder sous des angles différents afin de comprendre comment y implanter des lois de probabilités pour le rendre d'autant plus ludique. L'intéractivité offerte à l'utilisateur via un contrôle sur certains paramètres lui permet de visualiser l'impact qu'ils ont au cours du jeu et l'oblige potentiellement à adopter de nouvelles stratégies pour résoudre le puzzle.