Projet de Language de Programmation Sokoban

Andrius Ezerskis & Moïra Vanderslagmolen

December 20, 2022

1 Introduction

Nous avons séparé ce rapport en plusieurs parties. Tout d'abord, nous présentons les tâches accomplies. Ensuite, nous verrons les différentes classes implémentées et nous les expliquerons. Par après, nous parlerons de la logique du jeu, nous décrirons le déroulement d'un début de jeu, d'une fin de jeu, de l'enregistrement des pas et enfin des widgets. Pour finir, nous discuterons du modèle MVC dans notre projet.

2 Tâches Accomplies

Nous avons accompli 10 tâches. Nous avons implémenté :

- Les boîtes de couleur
- Les cases de téléportation
- Les boîtes légères
- Le compteur de pas
- Le meilleur score de pas
- L'écran d'accueil
- Les niveaux et sélection de niveaux
- La limite de pas
- Le déplacement automatique à la souris
- La détection d'échec

3 Classes

3.1 Classes du Modèle

3.1.1 BoardModel

Nous avons le BoardModel, qui s'occupe de gérer la logique du plateau. Elle fait bouger le personnage, compte le nombre de pas, détermine si la partie est terminée ou si elle ne peut plus être résolue. Elle peut aussi téléporter le personnage d'une case de téléportation à une autre Ensuite

3.1.2 BoxModel

Cette classe représente le modèle de la boîte. Elle contient plusieurs attributs : la couleur de la boîte, si elle est bloquée et son type (légère ou lourde) ainsi que la position de la boîte.

3.1.3 LogicCell

Cette classe représente une cellule ou une case. Elle a une position, peut avoir un joueur ou alors une boîte. Elle a aussi une couleur et un type. Elle peut avoir 4 types différents (mur, vide, téléportation ou position finale d'une boîte). Si le type est la position finale d'une boîte, alors elle est "complète" que si la couleur de LogicCell correspond à celle de la boîte.

3.1.4 Player

La classe Player contient uniquement la position du joueur.

3.1.5 Téléportation

La classe téléportation est un peu plus complexe. En effet, elle prend deux LogicCell en paramètre. Elle est terminée uniquement lorsqu'elle a ses deux bords(ses deux LogicCell)

3.2 Classes de la Vue

3.2.1 CellDisplay

3.2.2 DisplayBoard

Le DisplayBoard va itérer à travers les LogicCell et créer des instances de CellDisplay. Il va aussi dessiner les limite de pas, le compteur de pas et le nombre de pas minimum sur la fenêtre. Enfin, lorsque l'utilisateur clique sur le DisplayBoard, il va demander à chaque Cell si elle a été cliquée et renverra le résultat au MainWindow.

3.2.3 MainWindow

MainWindow s'occupe de tous les widgets, ainsi qu'envoyer les commandes de l'utilisateur au ControllerBoard. Il s'occupe donc de sauvegarder le nombre minimum de pas, de gérer les événements et de gérer la fenêtre principale du jeu.

3.3 StartWindow

Cette fenêtre permet d'afficher un écran d'accueil. Elle se lance en premier et disparaît après 10 secondes.

3.4 Classes du Controlleur

3.4.1 ControllerBoard

Cette classe gère les commandes entrées par l'utilisateur. Il s'occupe aussi des clicks.

4 Logique du jeu

4.1 Démarrer le jeu

Lorsque nous démarrons le jeu, nous donnons un fichier au BoardModel, qui va créer un vecteur de LogicCell. Ensuite, nous allons créer MainWindow, qui va lui-même créer ControllerBoard et un DisplayBoard, ainsi que tous les widgets.

4.2 Jouer

4.3 Widgets

4.4 Quitter le jeu

Lorsque nous quittons le jeu, le callback s'occupant de fermer la fenêtre va enregistrer les pas

4.5 Enregistrer les pas

5 Modèle-Vue-Controlleur

Nous avons implémenté le modèle MVC uniquement pour le board. Premièrement, nous avons d'abord implémenté tout le jeu en modèle MVC, mais FLTK n'est pas adapté au modèle MVC. En effet, les callback des widgets sont premièrement statiques, nous étions donc obligés d'appeller notre Controller dans la Vue, ce qui n'avait pas beaucoup de sens.

6 Conclusion