Année universitaire 2018-2019

Rapport de projet Conception Orientée Objet

Licence 3 Ingénierie Informatique

Réalisation d'un programme de Sokoban en Java

Projet réalisé par :
Mathilde MERLANGE
Soufian JADI
Farabé Timothée KAMATE

Encadré par :
Jean-Michel COUVREUR

Table des matières

I. Introduction :

Nous avons effectué notre projet de troisième année de licence informatique en Conception Orientée Objet sous l'encadrement de Monsieur Jean-Michel COUVREUR à l'université d'Orléans.

Cette année, le sujet portait sur la réalisation d'un programme de sokoban que l'on devait réaliser en java.

Le diagramme UML de notre projet est joint en annexe.

II. Description du travail :

a) Résumé :

Pour réaliser ce projet nous sommes partis du fait que le sokoban était comme un petit jeu de puzzle. L'objectif de ce jeu est assez simple. Il met en scène un personnage qui n'est autre que le joueur, qui se déplace sur un terrain, représenté par une grille de cases, et dont le but est de ranger des caisses sur des cases cibles (que nous appellerons destination).

Le terrain de jeu est constitué de deux types de cases, des murs que ni le joueur ni les caisses n'ont le droit de franchir et de cases vides. La structure du terrain de jeu et le nombre de caisses varient d'un niveau à l'autre.

Les règles du jeu sont les suivantes :

- Il y a autant de cases cibles que de caisses,
- Le joueur peut se déplacer dans les quatre directions (gauche, droite, bas, haut),
- Le joueur n'a le droit que de se déplacer vers une case non occupée (et qui n'est pas un mur) ou de pousser une caisse et de se déplacer ainsi sur une case libérée. Par contre, le joueur n'a pas le droit de tirer une caisse ni de la passer par-dessus ;
- Pour pousser une caisse la case adjacente de la caisse dans la direction de poussé doit être libre. Le joueur n'a pas la possibilité de pousser deux caisses en même temps,
- Le joueur ne gagne que lorsque toutes les caisses sont sur les cases cibles, peu importe l'ordre.
- Le score du joueur est le nombre de poussées réalisées (indépendamment du nombre de déplacement du joueur).

Notre objectif dans un premier temps était de réaliser une modélisation du jeu.

Le jeu se présente comme une grille en deux dimensions et on devait représenter le mur, les caisses, la destination et le soko lui-même (le joueur). La modélisation que l'on propose utilise la classe **VueIHMFX** pour stocker ces éléments du jeu.

b) Déplacement du soko :

Pour les éléments immobiles (mur, sol), il n'a pas été nécessaire d'enregistrer leur position. Par contre pour les éléments mobiles, vu que ceux-ci ne sont pas représentés dans la grille, nous avons jugé nécessaire qu'ils stockent eux-mêmes leur position, par le biais d'une classe **ModeleConcret** qui représente l'abscisse et l'ordonnée.

Afin de pouvoir faire évoluer les éléments mobiles, nous avons implémenté dans la même classe **ModeleConcret**, les déplacements à droite, à gauche, en haut et en bas.

Droite = 6

Gauche = 4

Haut = 8

Bas = 2

La classe est également composée d'une méthode `public int[][] getEtat()` qui renvoie l'ensemble des directions.

c) Condition de victoire :

Cette partie du code détermine si le joueur a gagné ou pas et donner par la suite une configuration en suivant les règles de représentation du sokoban.

```
/* Correspondance entier élément:
* 0 : mur (#) soko[0]
* 1 : caisse ($) soko[1]
* 2 : destination (.) soko[2]
* 3 : caisse sur une zone de rangement (*) soko[3]
* 4 : personnage sur une zone de rangement (+) soko[4]
* 5 : sol ( ) soko[5]
* 6 : soko (@) soko[4]
*/
```

d) Undo/ redo :

Undo : Lors de chaque exécution de commande, on mémorise l'état du jeu (déplacements…). Ainsi quand on annule la commande, on revient à l'état précédent. En d'autres termes, annule la dernière action effectuée.

Redo : Après avoir tout repositionné dans l'état initial, pour annuler la dernière commande, on ré-exécute toutes les commandes sauf la dernière. En d'autres termes, rétablit la dernière action annulée.

e) Lecture fichiers niveaux :

Représente le niveau en cours d'exécution.

f) Affichage du résultat :

Contient principalement tout ce qui concerne l'interface du jeu. Affiche une représentation textuelle du niveau en cours de jeu.

III. Conclusion :

Ce projet nous a permis de comprendre et d'expérimenter les étapes de conception d'une application. Il nous a également permis de nous familiariser avec le MVC et les entrée/sorties à partir d'un fichier. De plus, la programmation en javaFX nous a permis d'améliorer nos connaissances du langage java.

Il nous reste cependant quelques fonctionnalités mineures à implémenter :

- Lecture fichiers niveaux
- Affichage du résultat

Annexe

Diagrammes de classes