

Rapport complément de POO

Lebreton Nohan
Becaye Gaye
Benjamin Guillot
Brian Longuet

8 avril 2022

Table des matières

1	Introduction	2
1.1	Projet complément de POO	2
1.2	Puzzle à glissière (Taquin)	2
2	Manuelle d'utilisation	3
2.1	Mode interface graphique	4
2.1.1	Bouton restart	5
2.1.2	Bouton finish	5
2.1.3	Bouton trick	6
2.2	Mode terminal	6
3	Organisation du projet	7
3.1	Arborescence des fichiers	7
3.2	Architecture du programme	8
4	Conclusion	8
4.1	Objectifs remplis ?	8
4.2	Améliorations possibles	9
4.3	Références	9

1 Introduction

1.1 Projet complément de POO

Pour les étudiants en informatique, la programmation orientée objet apparaît comme un passage obligatoire. Ainsi, afin d'en appliquer les principes et de parfaire notre maîtrise de Java, un projet nous a été proposé dans le cadre de l'unité d'enseignement de TPA (Travail Personnel Approfondi).

Le but de ce projet est de réaliser une application de jeu, dotée d'une interface graphique, (mais pouvant être utilisé sans l'interface graphique) qui consiste en un puzzle à glissière.

1.2 Puzzle à glissière (Taquin)

Une grille de n lignes sur m colonnes comporte $n*m-1$ éléments (c'est-à-dire qu'une case est vide).

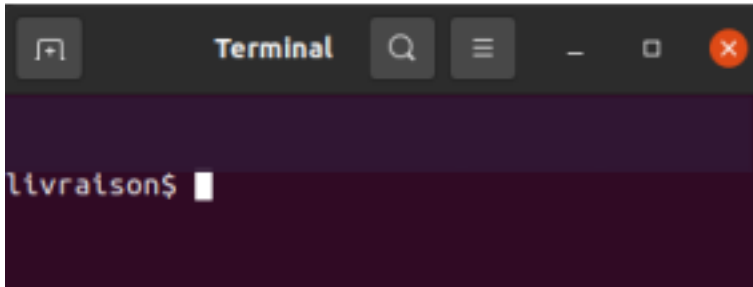
Le joueur peut faire glisser l'un des éléments contigus à la case vide vers cette dernière (l'élément du haut, du bas, de gauche ou de droite).

À tout moment, il y a donc 2, 3 ou 4 éléments déplaçables.

Le but du jeu est de reconstituer l'image du puzzle par déplacements successifs. La partie est finie lorsque le puzzle est reconstitué.

2 Manuelle d'utilisation

Pour pouvoir lancer l'application de la racine du fichier à l'aide d'un terminal :



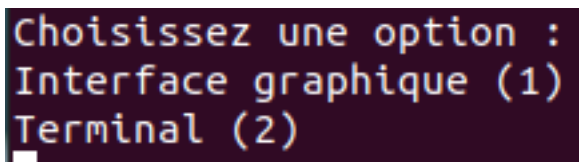
Pour compiler les fichiers :

```
javac -d build
src/modele/taquin/*.java
src/modele/tuile/*.java src/modele/*.java
src/vue/terminal/*.java
src/main/Main.java
src/controle/gui/*.java
src/vue/gui/*.java
src/controle/terminal/*.java
```

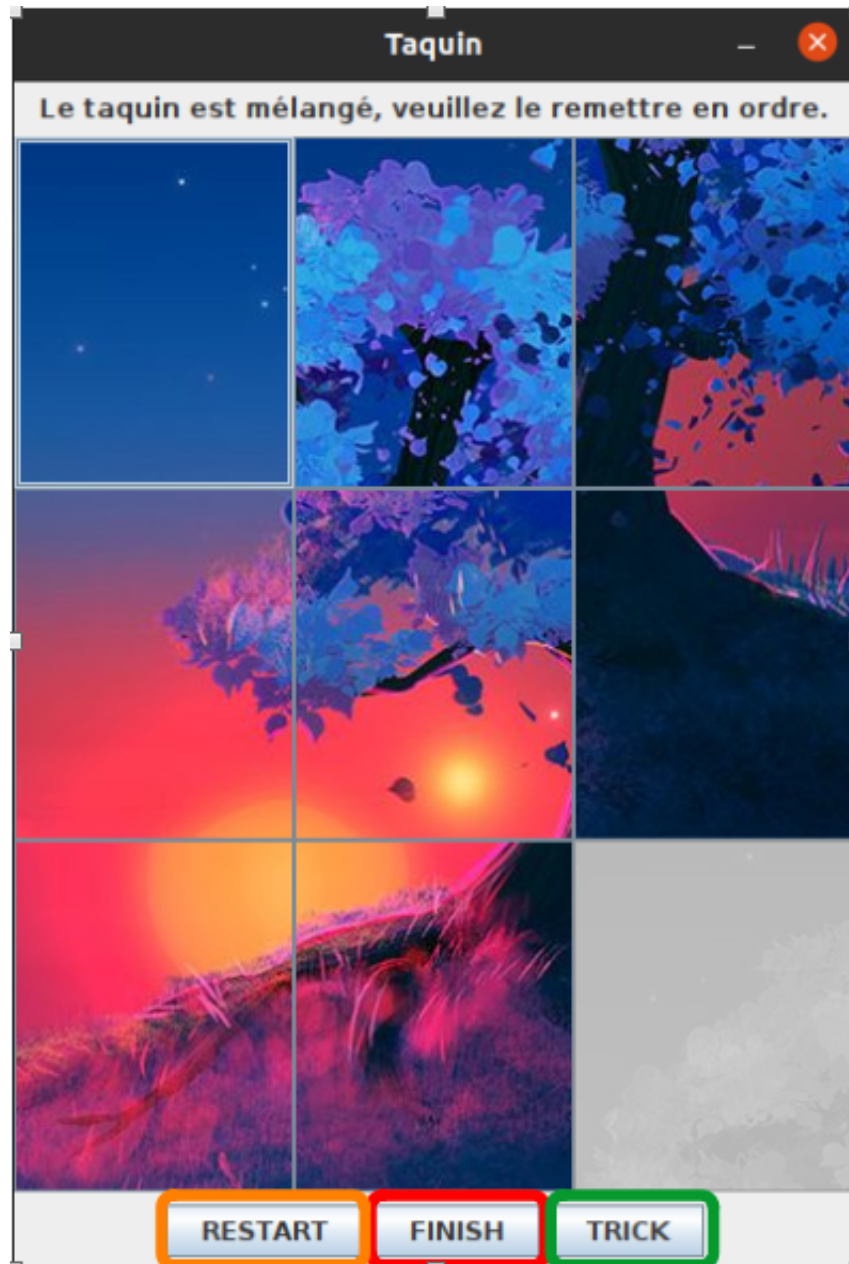
Pour executer le prog :

```
java -cp build main.Main
```

Dans le but de choisir le mode de lecture, la fenêtre suivante s'ouvre. Rentrez "1" pour le mode interface graphique et "2" pour le mode terminal seulement.

A screenshot of a terminal window with a dark background. It displays a menu with two options: 'Interface graphique (1)' and 'Terminal (2)'. A cursor is positioned at the end of the second line, ready for input.

2.1 Mode interface graphique

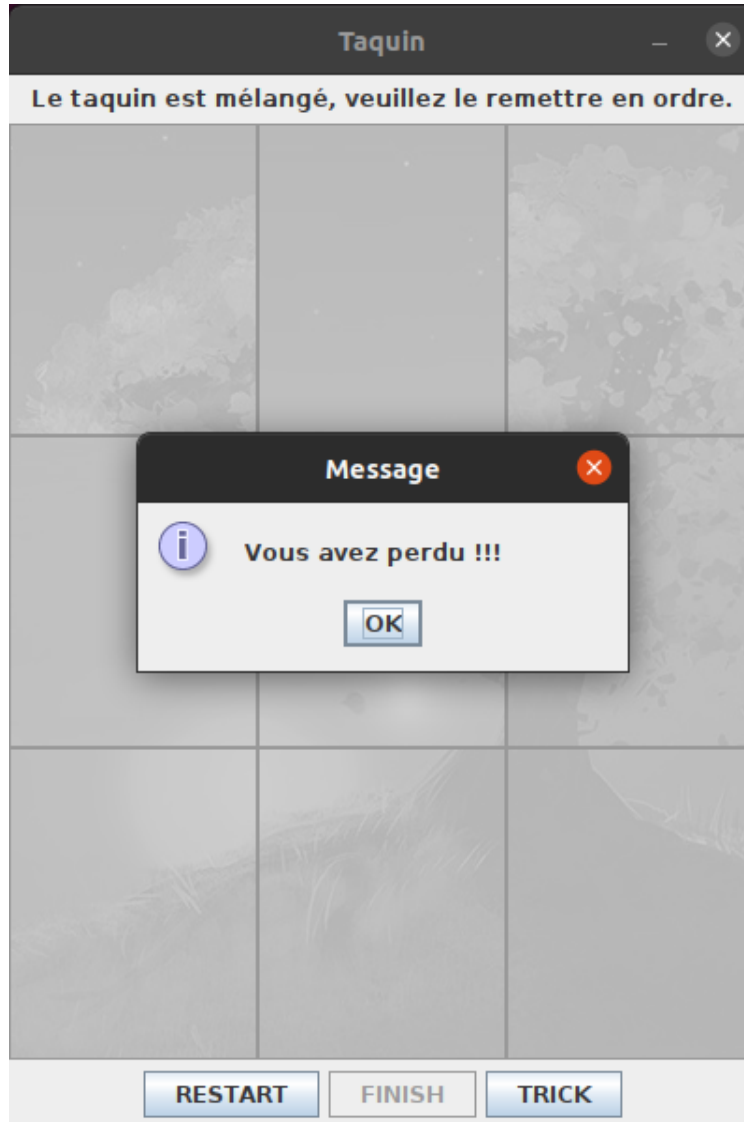


On peut bouger les tuiles en cliquant sur l'une d'elles ou bien en utilisant les flèches directionnelles.

2.1.1 Bouton restart

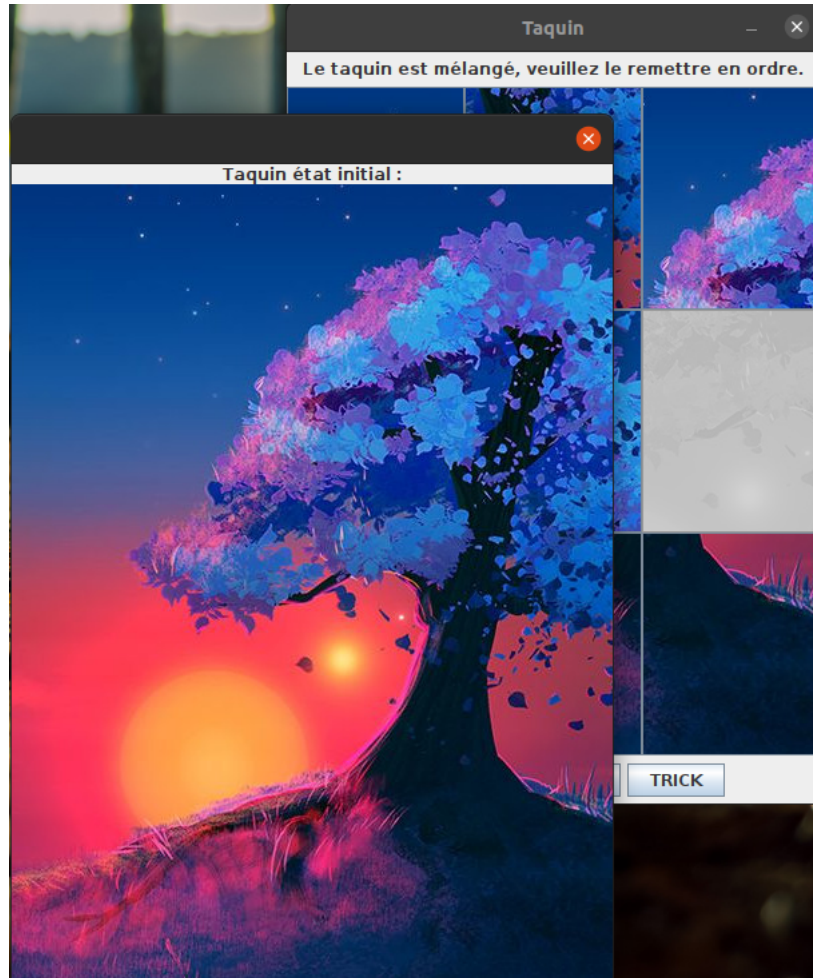
Mélange le taquin de façon aléatoire.

2.1.2 Bouton finish



Vérifie si le taquin est valide si oui c'est gagné sinon c'est perdu ! Une fois la verification faite on ne peut plus toucher au taquin. Il faut donc restart ...

2.1.3 Bouton trick



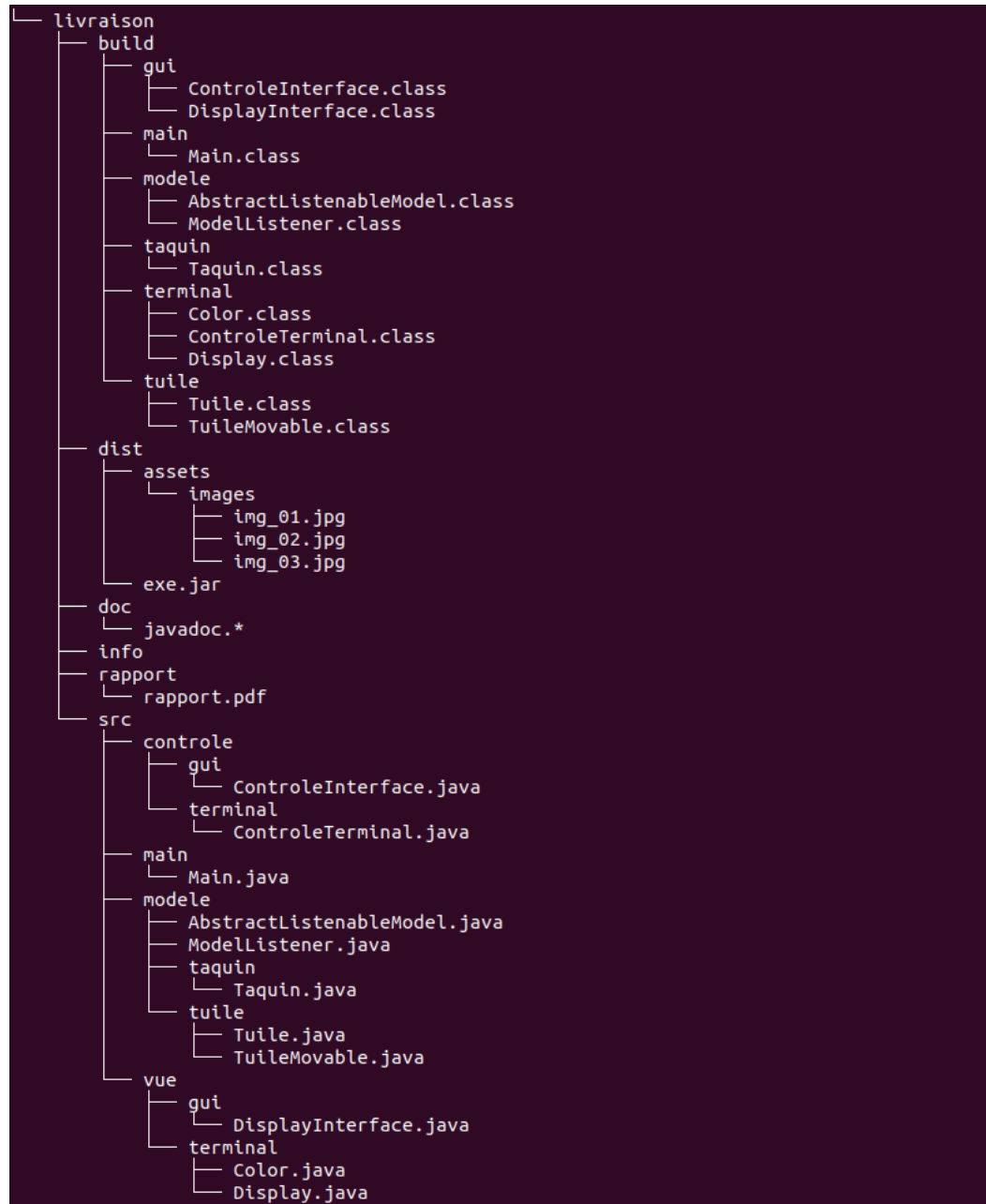
Ouvre une nouvelle fenêtre contenant l'image final.

2.2 Mode terminal

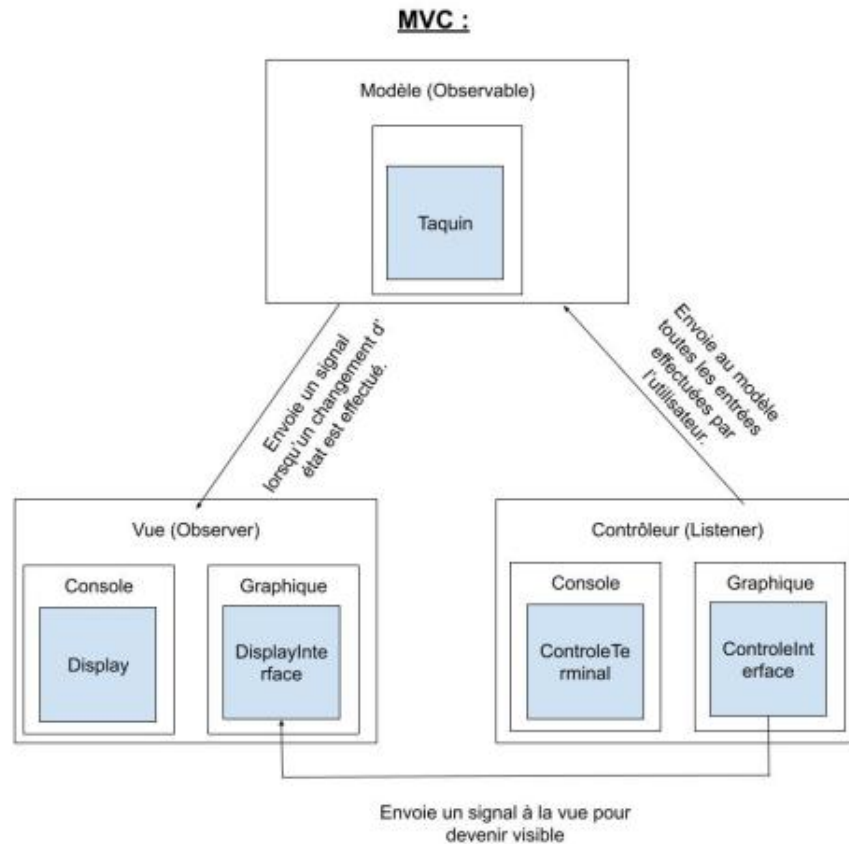
Le jeu est jouable à l'aide du terminal, il suffit simplement de suivre les instructions.

3 Organisation du projet

3.1 Arborescence des fichiers



3.2 Architecture du programme



4 Conclusion

4.1 Objectifs remplis ?

Au cours de la réalisation de ce projet, nous avons été à de nombreuses reprises pris par le temps. Malgré cela, le jeu est fonctionnelle en console et sur une interface graphique. Par ailleurs, nous sommes satisfaits de notre code. Il est clair, commenté et donc facile à comprendre. Une bonne répartition des packages et des classes. Facile à maintenir et à faire évoluer. Les critères sont remplis.

4.2 Améliorations possibles

Il apparaît comme évident que de nombreuses améliorations sont possibles pour un tel projet.

- Par exemple avoir un meilleur reponsive disigne notamment lorsque l'on a découpé l'image.

- Notamment dans la direction artistique avec par exemple du son (en fond mais aussi au déplacement d'une tuile) pour une meilleur immersion dans le jeu.

- De plus un déplacement fluide des tuiles afin de pouvoir suivre la tuile lors de son déplacement. Mais encore un autre aspect du jeu comme pourvoir affronter une IA avec différents niveau de difficulté. Avec différentes méthodes : recherche en profondeur, heuristiques, distance de Hamming, distance de Manhattan, algorithme Greedy, A* ...).

- Rendre possible que l'utilisateur donne sa propre image a jouer.

4.3 Références

- <https://fr.wikipedia.org/wiki/Taquin>

- <https://www.w3schools.com/java/>

- <https://kalanir.blogspot.com/2010/02/how-to-split-image-into-chunks-java.html?fbclid=IwAR1o6Q09EVY8hs2tFPoZmkUYlI28GLdt82YaJqYs1HQCNGgxlgU8bnsGR0w>