Projet Solveur de Ricochet Robot

Travail Personnel Approfondi

ACUNA Mithian, BOURNONVILLE Aurélien, HUET Bryan, SAHIN Tolga

Année universitaire 2020

Introduction

Table des matières

1 Implémentation

1.1 Solveur de Ricochet Robot

1.2 Notre version du jeu

2 Organisation

2.1 Répartition des tâches

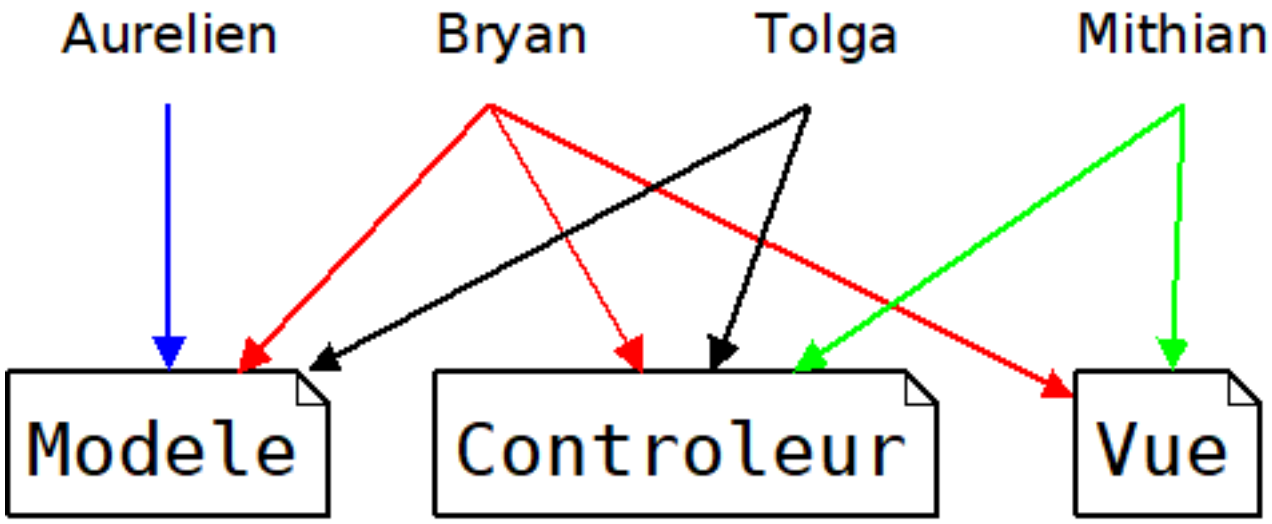

Schéma d'organisation

2.2 Architecture du programme (MVC)

L'architecture MVC consiste a decomposer le code en 3 parties : Le Modèle , c'est là où se trouvent les données. Il s'agit en général d'un ou plusieurs objets Java (move(),state()...). Ensuite nous avons la Vue qui représente ce que l'utilisateur a sous les yeux. La vue peut donc être :

— une application graphique Swing, AWT, SWT pour Java (pour C #...)
— une page web
— un terminal Linux ou une console Windows

Et enfin nous avons le Controleur permet de faire le lien entre la vue et le modèle lorsqu'une action utilisateur est intervenue sur la vue. Le modèle MVC permet de faciliter comment le code a était fait, elle facilite la vision du code.

2.2.1 Modele

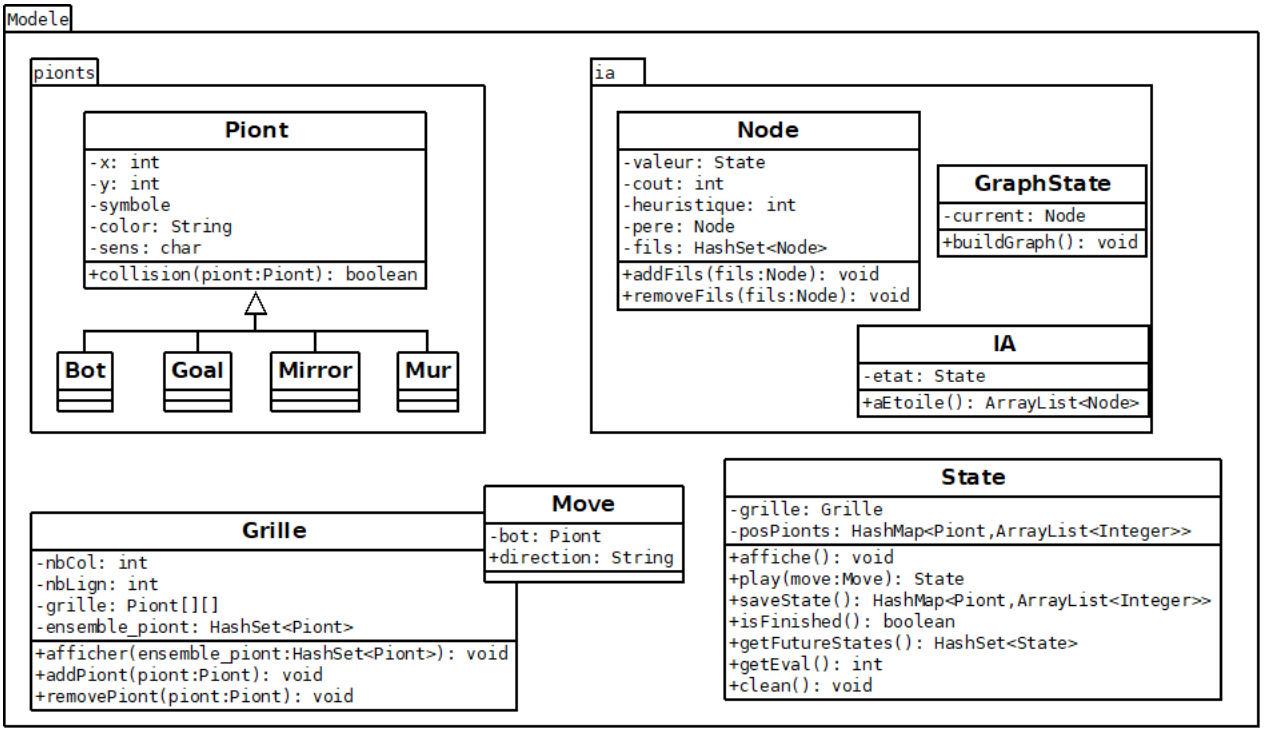

FIGURE 1 – UML Modele

2.2.2 Controller

(Schema UML de Controler)

2.2.3 Vue

(Schema UML de VUE)
Nous avons la classe Vue.java qui contient le code de la fenêtre(JFrame).

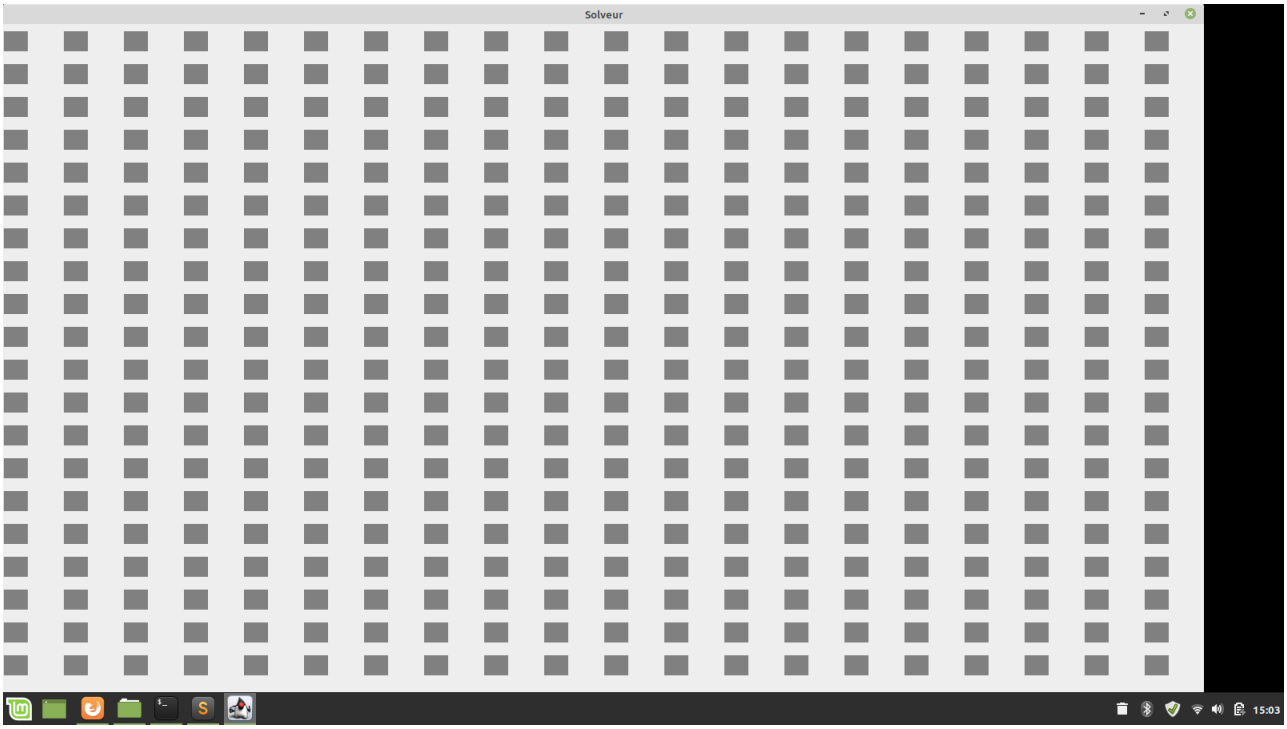

FIGURE 2 – Frame du Solveur

Cette ici ou on ajoute les autres classe comme "PanelNord","PanelWest","VueGrille",etc...
Tout d'abord nous avons la classe "VueGrille.java" qui permet de creer les cases de la grille

FIGURE 3 – Frame de la VueGrille

Ensuite dans le "PanelNord" nous avons 2 boutons(JButton) qui un permet de mettre en marche le Solveur, et un autre boutton qui permet de ouvrir le Menu à l'aide d'un "ActionListener".

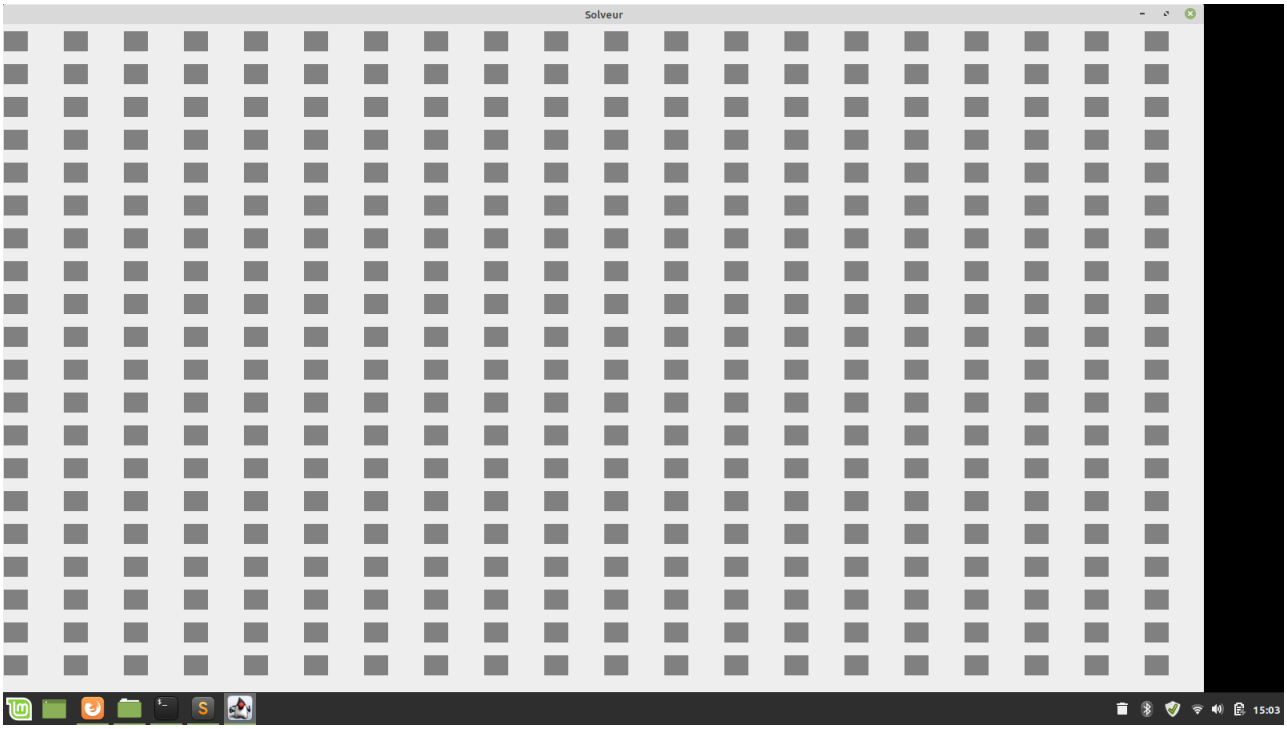

FIGURE 4 – Frame du PanelNord

Nous avons le "PanelWest".

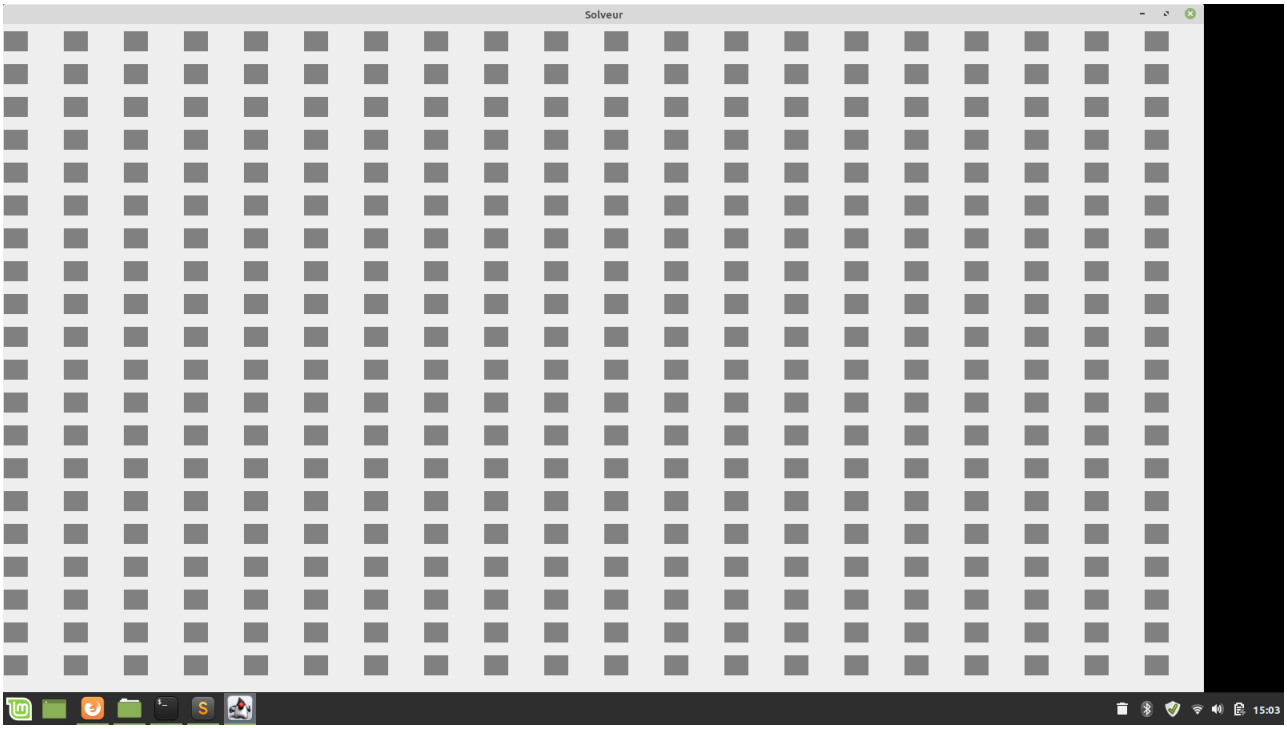

FIGURE 5 – Frame du PanelWest

3 Éléments techniques

3.1 Génération de la carte

Afin de générer la carte en plaçant divers objets de façon homogène, nous avons décidé de diviser la carte proportionnellement à sa taille et d'y placer 4 angles et 4 murs simples.

(EN ALGO PLUS TARD :)

Fonction decoupage() nb-compartiment = taille/10 //taille, attribut de la classe taille = (taille/nb-compartiment)*nb-compartiment retourner taille/nb-compartiment //Retourne la taille d'un compartiment Fin Fonction

La taille est modifier pour pouvoir diviser la grille en part égale d'environ 10 à 19 cases.

3.2 Gestion des etats

3.3 Intelligence Artificielle

Les consignes nous suggère d'utiliser un algorithme de parcours de graphe, nommé A* (ou A étoile, A star, etc..). Cet algo se base sur le principe de Dijkstra en y ajoutant une évaluation heuristique pour chaque noeud. Le principe est le suivant :

On définit un graphe où les états de jeu sont représentés par des lettres. Le coup représente le nombre de coup entre le noeud de départ et le noeud d'arrivée, et l'heuristique représente la distance (de Manhattan) entre ces derniers. On a donc le graphe suivant :

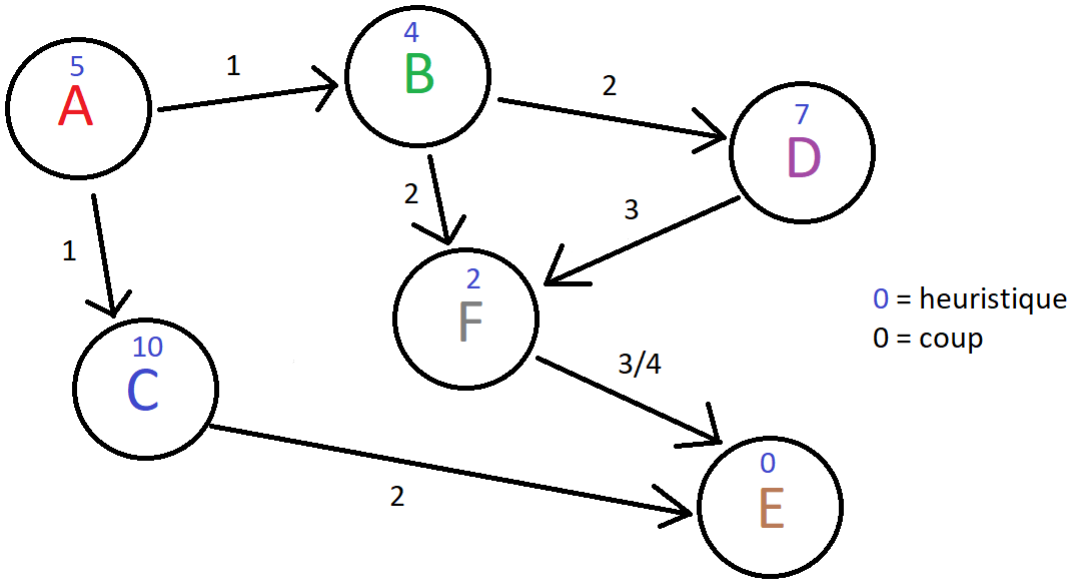

FIGURE 6 – Exemple de Graphe

afewfsafw

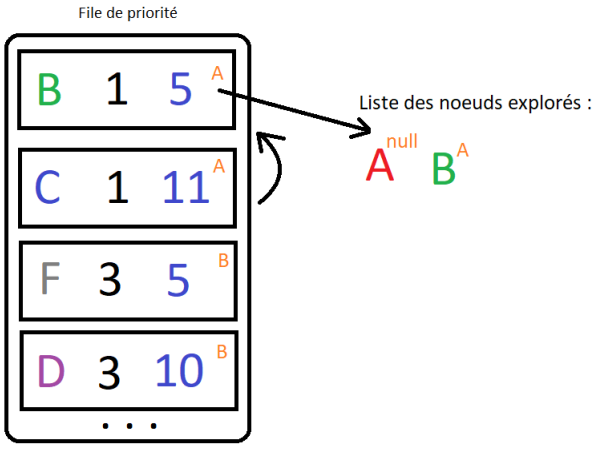

FIGURE 7 – A* phase 1

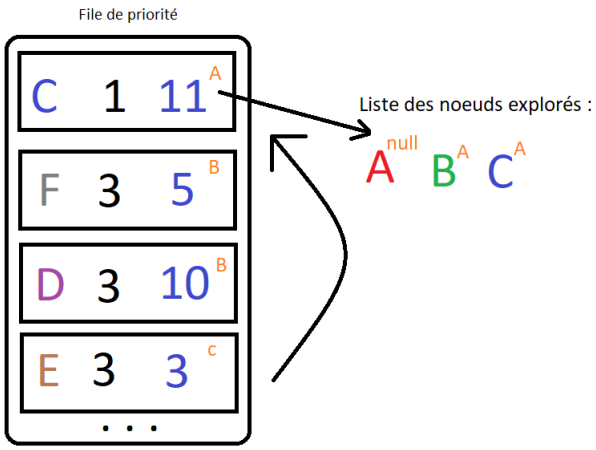

FIGURE 8 – A* phase 2

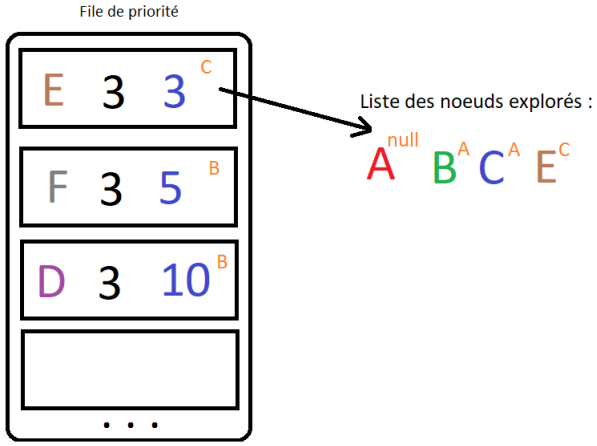

FIGURE 9 – Package ia

FIGURE 10 – Chemin le plus rapide

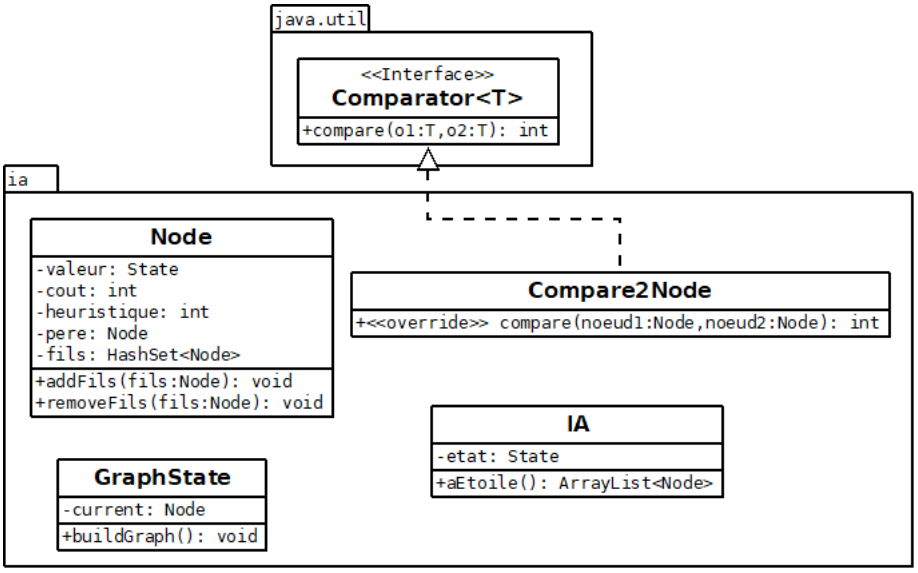

FIGURE 11 – Package ia

4 Conclusion

4.1 Objectifs remplis ?

4.2 Améliorations possibles

5 Bibliographie