Rapport SAE 2.1

2023

**Table des matières :**

1. Introduction………………………………………………………….p.3
2. Description des fonctionnalités…………………………….p.3
3. Présentation de la structure du programme………...p.10
4. Exposition de l’algorithme déterministe choisi…..…p.11
5. Conclusion……………………………………………………………p.12
6. Introduction

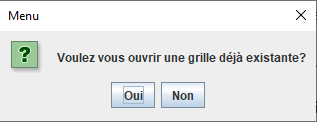
Lors de cette SAE nous devions créer une application qui permet à l’utilisateur de créer ou de charger des grilles qui représentent des labyrinthes et de pouvoir les résoudre de 2 manières différentes :

* La première, d’une façon totalement aléatoire, Thésée doit se mouvoir de façon totalement aléatoire dans le labyrinthe pour arriver à la sortie
* La seconde, d’une façon déterministe, Thésée doit arriver à la sortie de la façon la plus rapide et optimisée possible.

Ces deux méthodes peuvent être observée de deux façons, une automatique et l’autre où l’utilisateur appuie sur une touche du clavier pour avancer d’une étape dans la résolution du labyrinthe.

1. Description des fonctionnalités

Tout d’abord on demande à l’utilisateur s’il veut ouvrir un fichier contenant une grille :



Si l’utilisateur clique sur « Oui » cela ouvre une nouvelle fenêtre qui permet à l’utilisateur de choisir

un fichier à l’aide d’un JFileChooser directement placé dans le répertoire courant :

Une image contenant texte

Description générée automatiquement

L’utilisateur choisi alors son fichier et ensuite clique sur Open, on affiche alors la grille à l’utilisateur :

Une image contenant mots croisés, texte

Description générée automatiquement

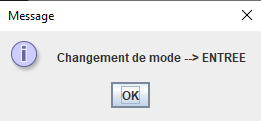
Il peut alors modifier la grille en supprimant et en ajoutant des cases noires :

Une image contenant mots croisés

Description générée automatiquement

Lorsque l’utilisateur appuie sur une touche du clavier on informe l’utilisateur du changement de

mode via un JOptionPane, il peut alors modifier la position de l’entrée :

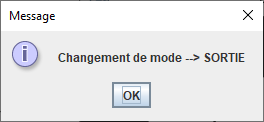


Une image contenant mots croisés, texte, électroménager

Description générée automatiquement

Ensuite, si l’utilisateur appuie encore une fois sur une touche du clavier,on informe encore

l’utilisateur du changement de mode, il peut alors changer la position de la sortie :

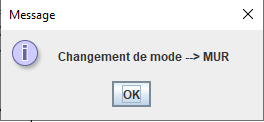


Une image contenant mots croisés, texte, électroménager

Description générée automatiquement

Ensuite, si l’utilisateur appuie encore une fois sur une touche du clavier, on l’informe du changement

de mode, il peut ici remodifier les murs une dernière fois :

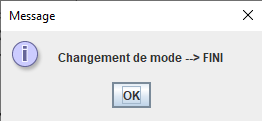


Une image contenant mots croisés

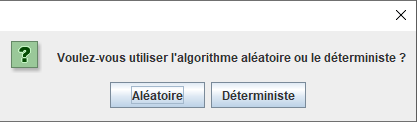
Description générée automatiquement

Si l’utilisateur appuie encore une fois sur une touche du clavier, on lui informe que l’on a fini l’étape

de modification de la grille :

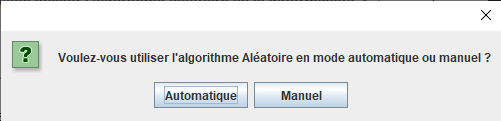


On lui demande alors s’il veut utiliser le mode aléatoire ou déterministe :



Lorsque l’utilisateur clique sur l’un des boutons on lui demande s’il veut utiliser l’algorithme en mode

automatique ou manuel :



Si l’utilisateur clique sur le mode automatique alors le programme essaye de résoudre le labyrinthe et

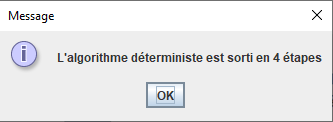
à s’il arrive à résoudre le labyrinthe, l’utilisateur en est informé avec le nombre de coup requis :

Exemple pour l’algorithme aléatoire :

Une image contenant texte

Description générée automatiquement

Exemple pour l’algorithme déterministe



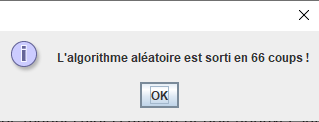
Si l’utilisateur clique sur le mode manuel alors l’utilisateur doit appuyer sur une touche de son clavier

pour avancer dans la résolution, l’endroit où est Thésée est représenté par une case verte claire.

Une image contenant mots croisés

Description générée automatiquement

Lorsque Thésée est arrivé à la fin, une pop-up s’affiche :



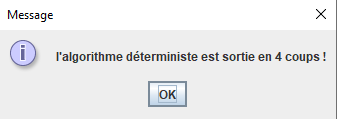
Lorsque l’on est dans l’algorithme déterministe les cases où est passé Thésée sont représentés par

des cases vertes foncées et les cases où est Thésée actuellement sont en vert clair:

Une image contenant mots croisés

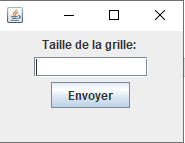
Description générée automatiquement

Lorsque Thésée est arrivé à la fin, une pop-up s’affiche :



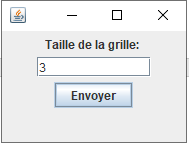
Si l’utilisateur choisit de ne pas charger une grille, alors le programme lui demande la taille de la grille

qu'il veut créer :

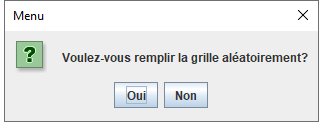


L’utilisateur doit alors écrire un nombre dans le JTextField et appuyer sur envoyer pour passer à la

suite :



On demande alors s’il veut remplir la grille aléatoirement :



Si l’utilisateur clique sur « Oui » on lui affiche alors une grille aléatoire de la taille demandée :

Une image contenant texte, mots croisés

Description générée automatiquement

Il peut alors, comme si il avait choisi un fichier, modifier cette grille puis on lui demande quel

algorithme il veut utiliser comme précédemment.

Si il choisi « Non » on lui affiche une grille vide où l’entrée et la sortie sont chacune dans un coin de la

grille :

Une image contenant shoji, mots croisés

Description générée automatiquement

Il peut alors ajouter ou supprimer des cases noires comme précédemment.

1. Présentation de la structure du programme

Une image contenant diagramme, schématique

Description générée automatiquementVoici le diagramme de classes représentant la structure de notre programme :

On peut y voir les différentes implémentations des interfaces tels que les KeyListener ou les

MouseListener. On voit aussi comment se comporte le programme suivant les choix qu’il fait

(algorithme déterministe ou non, etc.)

1. Exposition de l’algorithme déterministe choisi

On peut considérer la grille que l’on a comme un graphe ou chaque sommet a 4 arêtes. On a donc

choisi de faire un algorithme qui utilise le parcours en largeur. On peut alors utiliser l’algorithme

expliqué grâce au diagramme d’activité suivant :

Une image contenant graphique, diagramme

Description générée automatiquement

Cet algorithme arrive forcément à la fin car on effectue un parcours en largeur, on parcoure alors tout

le graphe, de plus, un parcours en largeur nous donne toujours le chemin le plus rapide jusqu’à

l’arrivée car on prend pas à pas les cases allant le plus rapidement à la fin.

1. Conclusion

Lionel : J’ai trouvé ce projet assez long et moins amusant que le 1e, de part le fait qu’on ne créait pas

un jeu. J’aurais d’ailleurs bien aimé que l’on ai appris les listes chainées pour la résolution du

labyrinthe, cela nous aurait simplifié la partie programmation je pense. Dans l’ensemble j’ai aimé ce

projet car j’ai pu comprendre des aspects de la programmation que je n’avais pas (ou mal) compris

auparavant comme la lecture de fichiers.

Mathieu : Pour ma part, j'ai moins bien vécu cette SAE que la précédente. Déjà car je trouvais le sujet

moins attrayant, mais également car le nombre de cas isolés à contrôler était bien plus important.

Néanmoins, malgré les dizaines et les dizaines d'heures à galérer sur des erreurs stupides

d'algorithmie, j'ai trouvé ce projet intéressant dans la mise en place des différentes façons de faire

une même chose (les différentes manières d'ouvrir une grille), ou encore dans l'utilisation des listener

ainsi que d'un tableau pour l'affichage graphique. Je suis fier d'y être parvenu, et comme pour la

première SAE j'ai été content de pouvoir compter sur mon camarade en cas de soucis, ce qui est

vraiment rassurant.