Rapport Mini-Projet POO

YMERAJ Eldis

15/04/2023

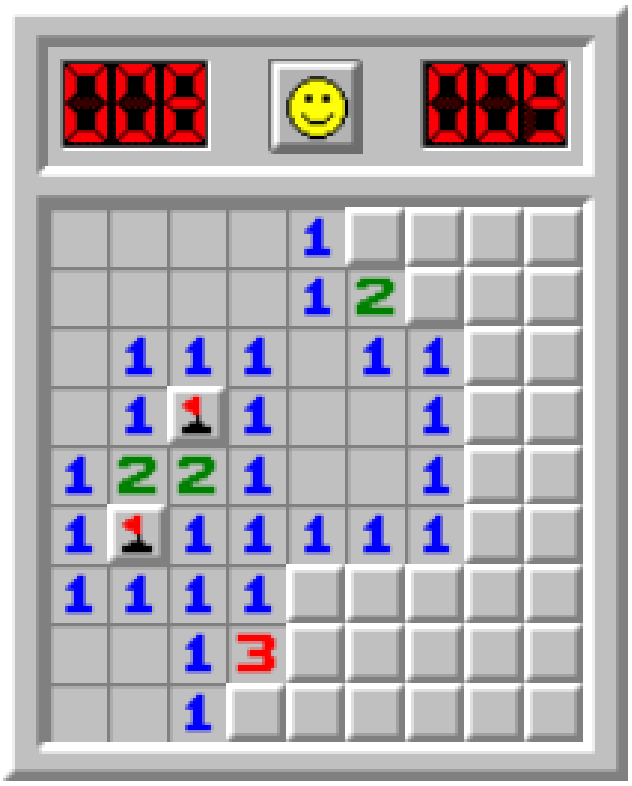

Figure 1 – Jeu de démineur

Table des matières

Introduction

Dans ce rapport, on présente l'implémentation du jeu de démineur en Java. Cette version du jeu utilise une grille de cases carrées, et le joueur doit découvrir les cases sans faire exploser les mines cachées. On a utilisé le concept de programmation orientée objet pour implémenter le jeu, ce qui facilite la maintenance du code et l'ajout de nouvelles fonctionnalités.

1 Jeu de démineur

Compilation du projet :
J'ai cré un fichier MAKEFILE pour que l'utilisateur compile plus facilement le projet. Il suiffit juste de taper sur le terminal la comande **make** pour que tous les fichiers compilent et apres la commande **make test** pour tester le jeu.
Aussi dans le code de cette projet vous allez trouver pas mal des expliquations sur mon travail réalisé.

1.1 Règles du jeu

Le jeu de démineur se joue sur une grille carrée, composée de cases cachées. Certaines de ces cases contiennent des mines, tandis que d'autres sont vides. Le but du jeu est de découvrir toutes les cases vides sans faire exploser les mines.

Au début du jeu, le joueur doit choisir la taille de la grille et le nombre de mines cachées. Le joueur peut alors cliquer sur une case pour la découvrir. Si la case ne contient pas de mine, elle affichera un chiffre qui indique le nombre de mines adjacentes (c'est-à-dire, les mines dans les cases voisines).

Si la case révélée contient une mine, le jeu est terminé et le joueur a perdu. Dans ce cas, toutes les mines sont révélées et le joueur peut voir où il s'est trompé.

Si le joueur découvre toutes les cases vides sans faire exploser les mines, le jeu est terminé et le joueur a gagné.

Pour faciliter la découverte des cases vides, le joueur peut utiliser la fonctionnalité "drapeau" qui lui permet de marquer les cases où il pense qu'il y a une mine cachée.

Il est important de noter que le joueur ne peut pas découvrir une case qui a été marquée avec un drapeau. De plus, si le joueur découvre une case qui ne contient pas de mine, mais qui n'a pas été entourée de drapeaux pour toutes les mines adjacentes, il risque de déclencher une mine cachée.

1.2 Fonctionnalités du programme

1.2.1 Menu

L'utilisateur démarre une nouvelle partie à l'aide de bouton « Nouvelle Partie ». Ce bouton le dirige vers une fenêtre lui permettant de choisir différents paramètres(fenêtre principale avec deux autres boutons).
À l'aide de cette fenêtre, l'utilisateur peut choisir le nombre de lignes, le nombre de colonnes et le nombre de bombes dont la grille sera formée. La taille de la grille est comprise entre du 4x4 et du 30x30 (forme rectangulaire possible) et le nombre de bombe est limité à un ratio ne dépassant pas le tiers du nombre de cases.
Un bouton « Valider » est aussi présent sur cette fenêtre et permet à l'utilisateur de démarrer sa partie lorsqu'il a choisi les paramètres qu'il souhaite.

Figure 2 – Menus de choix

Au cours de la partie, l'utilisateur peut sauvegarder et quitter sa partie pour la continuer plus tard. Sur le menu principal, il est possible de quitter l'application, de démarrer une nouvelle partie, ou de continuer une partie à condition que celle-ci ait belle et bien était sauvegardée auparavant.

1.2.2 Jeu

- Chaque case peut être dévoilée par l'utilisateur.
- La case numérotée affiche le nombre de bombes dissimulés dans les 8 cases disposées autour de cette même case. Si celui-ci vaut 0, alors il n'est pas affiché.
- Une case non dévoilée peut être marquée par le joueur à l'aide de son clique droit :
  - **" ?" :** Indique un doute sur le positionnement d'une bombe sur cette case.
  - **"Un drapeau rouge" :** Indique que le joueur est certain qu'une bombe se situe sur cette case.

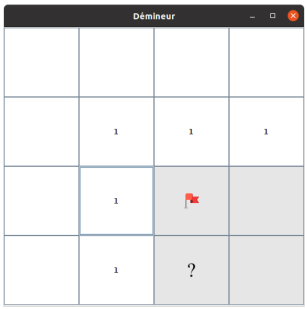

Figure 3 – Symboles de jeu

- Lors de la partie, le nombre total de mines est affiché, il diminuera lorsque le joueur posera un drapeau sur la grille.
- Si le joueur clique sur une bombe, il perd immédiatement et on lui indique sa défaite en affichant « perdu » en rouge. Si le joueur parvient à déminer la grille alors on lui indique sa victoire en affichant « gagné » en vert.
- En cas d'échec, les bombes sont affichées clairement sur la grille.

Figure 4 – Modes gaigné et perdu

2 Structure du programme

2.1 Présentation générale

2.1.1 Découplage du programme

Le programme est découpé en 8 fichiers, 7 fichiers sources et un fichier Makefile afin de réduire le volume des sources et de structurer celle-ci.

- Le premier fichier source permet de démarrer le jeu (source : Test.java).
- Le second fichier source concerne l'interface utilisateur (source : MenuTest.java). Menu principal et menu de saisie des paramètres de jeu (Taille de la grille et nombre de bombes).
- Le troisième fichier source permet de démarrer une partie et de gérer la partie en cours (source : Partie.java).
- Le quatrième fichier source permet d'afficher la grille de jeu, le menu de sauvegarde, l'état de la partie et son avancement, le menu contenant le nombre de mine restantes. En fin de partie, on notifie une défaite ou une victoire, et affiche un menu avec deux choix : « Nouvelle partie » et « Quitter » (source : Affichage.java).
- Le cinquième fichier source permet de faire la vérification de cases s'ils sont vides ou non (source : Changement.java).
- Le sixième fichier source effectue une sauvegarde d'une partie. Il sauvegarde l'état de la partie, la position des bombes et la taille de la grille de jeu dans le fichier foo.bin (source : Save.java).
- Le dernier fichier source permet de charger une partie sauvegarder auparavant. Il récupère l'état de la partie, la position des bombes et la taille de la grille de jeu (source : Load.java).
- Le fichier Makefile qui permet la compilation et donc le fonctionnement du jeu et la commande Make test qui permet l'exécution du jeu.

2.2 Bibliothèques utilisées

Les choix des bibliothèques **javax.swing**, **java.io**, **java.awt** et **java.awt.event** pour la réalisation d'une version du jeu Démineur en Java sont pertinents pour plusieurs raisons.

Tout d'abord, la bibliothèque **javax.swing** fournit une large gamme de composants d'interface graphique tels que des boutons, des champs de texte, des cases à cocher et des barres de défilement. Elle permet également la création d'interfaces utilisateur modernes pour les utilisateurs.

Ensuite, la bibliothèque **java.io** fournit des outils pour lire et écrire des fichiers, ce qui est essentiel pour sauvegarder les scores des joueurs ou les configurations du jeu. Cela permet également une personnalisation du jeu et une sauvegarde des préférences des joueurs.

Enfin, les bibliothèques **java.awt** et **java.awt.event** fournissent des outils pour créer des interfaces graphiques, gérer les événements et manipuler des éléments graphiques. Cela permet une personnalisation plus poussée de l'interface utilisateur et une meilleure gestion des événements, tels que les clics de souris, qui sont

cruciaux dans le contexte d'un jeu de Démineur.

En somme, ces choix de bibliothèques sont pertinents pour fournir une expérience utilisateur agréable et personnalisée, sauvegarder les scores et les configurations du jeu, ainsi que pour manipuler les éléments graphiques et les événements de manière efficace.

2.3 L'affichage

La classe **Affichage** permet d'afficher la grille de jeu et le menu de sauvgarde. Cette classe est une sous-classe de la classe JFrame, d'ou la classe JFrame est une classe de la bibliothéque Swing de Java, qui fournit un conteneur pour créer une fenêtre graphique avec une barre de titre, bordures et des boutons de contrôl.
En utilisant cette classe, ça nous a permit d'afficher le jeu dans un fenêtre et pas direct dans le terminal. Cette classe contient des différents méthodes qui nous ont permit d'avoir l'affichage souahité et en fin de partie, elle permet de notifier une défaite ou une victoire.

2.4 Vérification

Pour vérifier si les cases sont vides j'ai utilisé l'algorithme de révélation, qui est une méthode qui révèle la case donnée en paramètre et qui ensuite indique qu'il n'y a aucune bombe dans les cases adjacentes. Cette fonction se rappelle récursivement sur les 8 cases adjacentes en partant de celle en haut à gauche tout en vérifiant si ces cases existent bien.

2.5 Mécanisme de sauvegarde d'une partie

Pour sauvegarder la partie on sauvegarde des entiers (int) à l'aide d'un flux d'octets. D'abord on sauvegarde le nombre de lignes, puis le nombre de colonnes de la grille pour savoir au moment du chargement d'une partie combien de cases on doit lire. Puis on sauvegarde pour chaque case le fait qu'elle contienne une bombe mais si elle n'en contient pas, on sauvegarde le nombre de bombes dans les cases adjacentes. Puis on sauvegarde pour chaque case l'état de la partie. C'est-à-dire si la case est révélée, caché, avec un **drapeau** ou **?**.

2.6 Programme principal

La classe **"MenuTest"** permet d'interagir avec l'utilisateur à l'aide d'une menu et aussi gère les paramètres d'une nouvelle partie. Elle implémente les interfaces **ActionListener** et **MouseListener** de la bibliothèque Swing de Java pour détecter les événements d'action et de souris.
J'ai implémenté la méthode **actionPerformed()** qui est appelée lorsque l'utilisateur déclenche un événement d'action et dans notre cas, la pression d'un bouton.
La méthode implémenté **actionPerformed()** prends en paramètre la case sur laquelle on a cliqué et Effectue un action spécifique au bouton cliqué dans le menu, soit pour commencer une nouvelle partie, pour continuer la partie ou pour quitter le jeu.
L'interface MouseListener définit cinq méthodes différentes pour détecter les événements de souris : **mouseClicked()**, **mouseEntered()**, **mouseExited()**, **mousePressed()** et **mouseReleased()**. Chacune de ces méthodes est appelée lorsque l'utilisateur effectue une action de souris spécifique.

On a implementé ces méthodes pour effectuer des actions spécifiques en réponse à l'événement de souris correspondant.

CONCLUSION

Le projet de réaliser un démineur en programmation orientée objet est un excellent moyen de consolider ses connaissances et de mettre en pratique les concepts appris en cours. La conception du jeu de démineur implique une réflexion minutieuse sur la structure des classes, la gestion des événements et la gestion de la structure de données pour stocker les informations du jeu.
Ce projet a permis d'approfondir les connaissances en langage Java, notamment lors de l'écriture du code pour le menu principal et la gestion des événements pour détecter les clics de souris sur les cases. De plus, l'écriture du Makefile pour compiler et exécuter le code a également permis de se familiariser avec la gestion des dépendances et la compilation de programmes Java.
Bien que ce projet puisse encore être amélioré pour une meilleure optimisation du code ou une amélioration de l'interface utilisateur, il reste un excellent exercice pour renforcer les connaissances en programmation orientée objet et en Java. En somme, la réalisation du projet de démineur a permis d'acquérir une meilleure compréhension de la programmation orientée objet, ainsi que de l'application de ces concepts dans un projet concret.

Table des figures