BECHET Thomas
TERBAH Dorian

Rapport : Projet Tutoré
Démineur

07/06/18

BECHET Thomas
TERBAH Dorian

Sommaire

I - Introduction (p.3)

II - Fonctionnalités du programme (p.3)

III - Structure du programme (p.8)

IV - Mécanisme de sauvegarde (p.9)

V - Algorithme de révélation (p.12)

VI - Conclusion (p.15)

BECHET Thomas
TERBAH Dorian

I)Introduction

Description

Le démineur est un jeu solitaire qui consiste en un plateau de jeu dans lequel figurent des cases. Parmi l'ensemble des cases, certaines sont minées, et celles-ci doivent être trouvées par le joueur sans les révéler, auquel cas la partie sera perdue. De plus, le nombre de mines est choisi au préalable par le joueur ainsi que les dimension du jeu, permettant ainsi de choisir la difficulté de la partie. Le joueur gagne s'il arrive à révéler toutes les cases non minées.

Mise en œuvre

Nous avons commencé le projet en réalisant le diagramme de classe (UML) du projet en java. Ce diagramme nous a permi d'avoir une vue globale du projet ainsi qu'une aperçu des problèmes possibles.

Le second point concerne l'utiliser l'anglais comme langage de documentation du projet, sauf pour le rapport. Cette décision à pour but de nous forcer à prendre l'habitude d'utiliser l'anglais pour nos projets structurés. De plus, nous souhaitons tous les deux réaliser notre seconde année en alternance. L'anglais est donc indispensable.

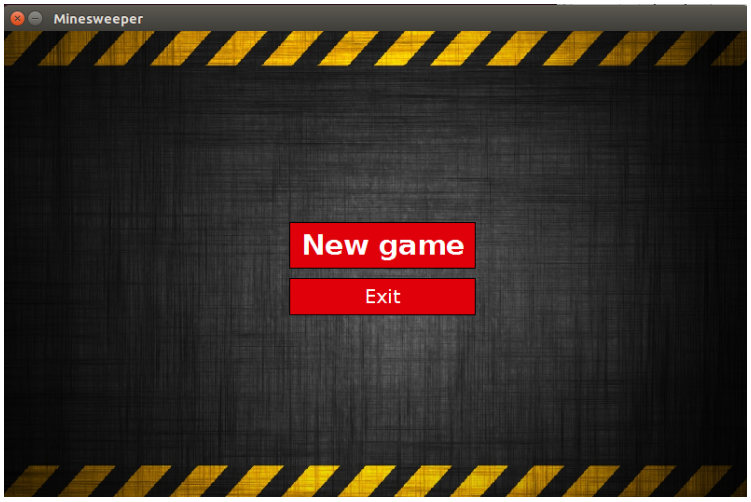
II)Fonctionnalités du programme

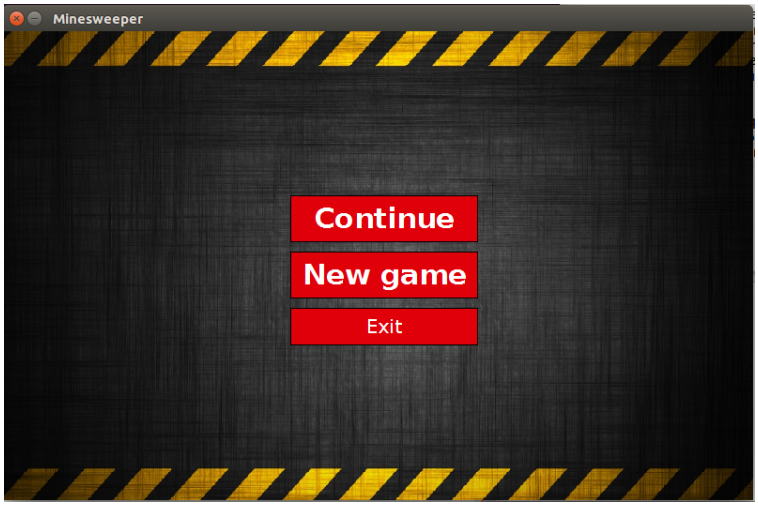
Sauvegarde des données

Le programme mets à disposition plusieurs fonctionnalités. Tout d'abords, lorsque le joueur est dans le menu du début de jeu, il peut démarrer une nouvelle partie ou continuer une partie antérieure sauvegardée, seulement si celle-ci existe. Si le joueur décide de reprendre son ancienne partie, le plateau de jeu de la partie sauvegardée est reconstitué à l'identique au niveau des cases et au niveau du temps écoulé depuis le début de l'ancienne partie.

BECHET Thomas
TERBAH Dorian

Ci-dessous une capture d'écran montrant le menu d'accueil si une partie est sauvegardée, ainsi qu'une autre montrant le même menu d'accueil si aucune partie n'a été sauvegardée.

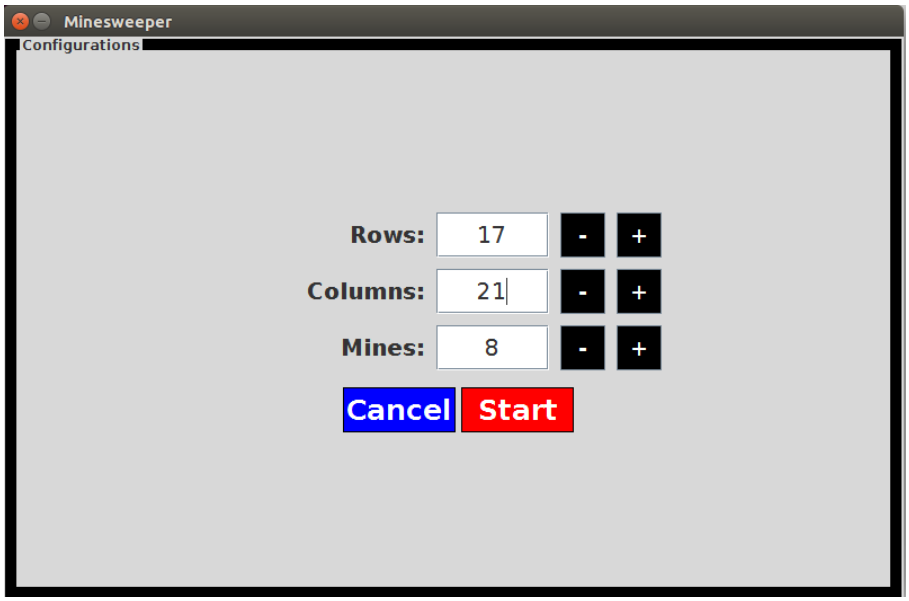




Configuration de la partie

Si au contraire le joueur décide de recommencer une partie, il se retrouve alors dans le menu de sélection où il peut choisir le nombre de lignes, de colonnes et de mines qu'il souhaite. Seulement, le joueur ne peut pas choisir un nombre de colonnes et de lignes inférieur à 4 ou supérieur à 30, et il ne peut pas choisir un nombre de mines nul ou supérieur aux dimensions du plateau de jeu (si le joueur a choisi de faire une partie avec n lignes et p colonnes, le nombre maximum de mines doit être inférieur à n X p).

Ci-dessous une capture d'écran du menu de sélection.



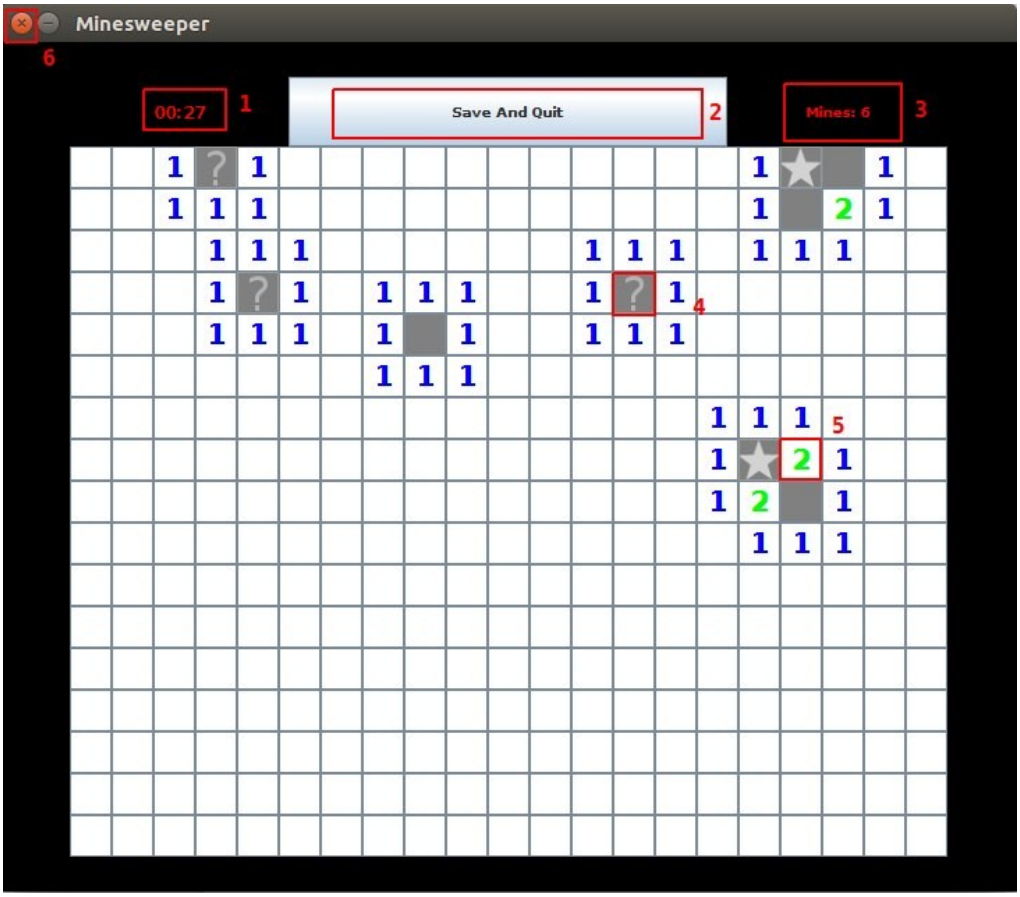
Interface de jeu

Une fois la sélection terminée, la partie peut alors commencer. Le temps écoulé depuis le début de la partie, un bouton de sauvegarde pour pouvoir sauvegarder sa partie actuelle et la reprendre quand il le souhaite, ainsi qu'un compteur qui correspond au nombre de mines à trouver sont disponibles tout au long de la partie.

De plus, le joueur a la possibilité de mettre une indication sur chaque case qu'il pense être minée. Parmi ces indications, on retrouve le point d'interrogation qui signifie que le joueur soupçonne qu'une case est minée et l'étoile signifie que le joueur est sûr que sur telle case est minée. Le nombre d'étoiles posées sur le plateau de jeu correspond au compteur décrit plus haut, qui correspond en fait au nombre de mines total auquel est soustrait le nombre d'étoiles posées. En cas de clic droit, le joueur change l'indication présente sur la case : de vide à l'étoiles, de l'étoile au point d'interrogation et du point d'interrogation à vide. Ces indications sont présentes uniquement pour aider le joueur lors de sa réflexion, elles n'ont donc aucun d'effet sur une case peut importe si elle est minée ou non.

BECHET Thomas
TERBAH Dorian

Ci-dessous une capture d'écran montrant les différents interfaces



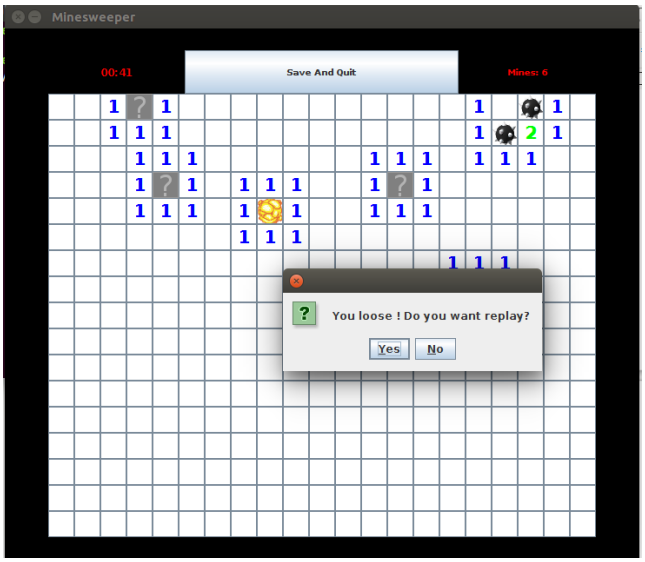
[1] Le compteur
[2] Bouton pour quitter
[3] Nombre de mines restant
[4] Mode de sélection
[5] Nombre de mines voisines
[6] Bouton pour quitter

Conditions de victoire

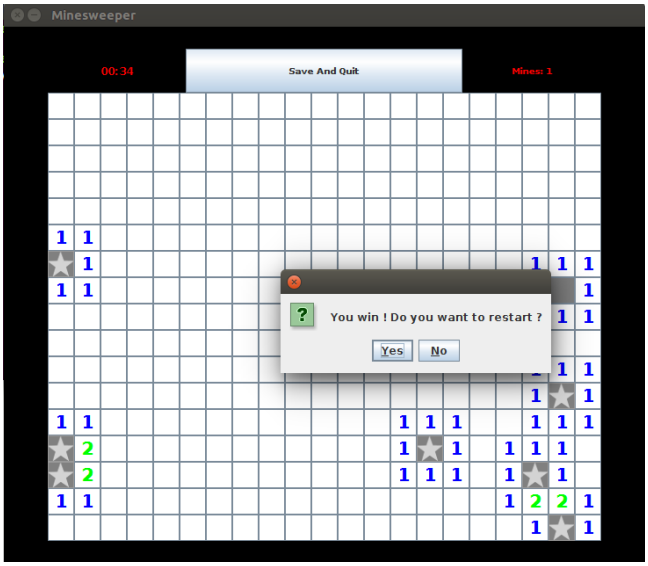
La victoire est atteinte si le joueur révèle toutes las cases qui ne sont pas minées. Ci-dessous deux captures d'écrans de fin de partie suivant que la joueur a perdu ou non.

BECHET Thomas
TERBAH Dorian

En cas de défaite



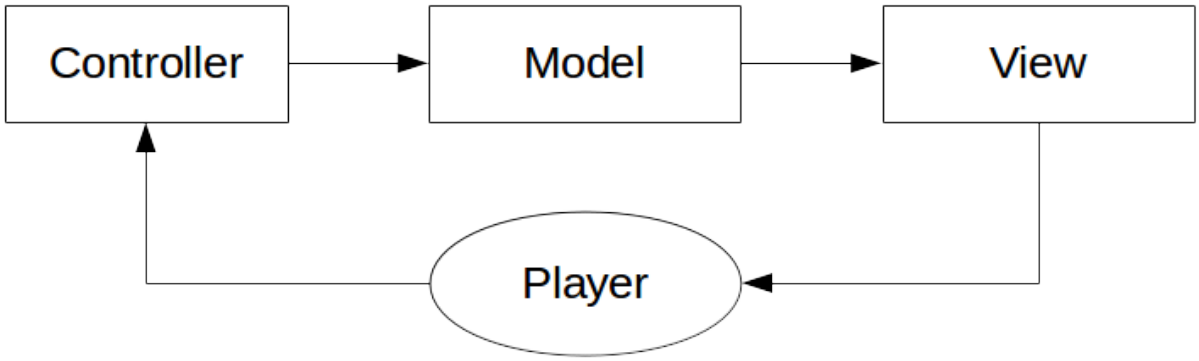
En cas de victoire



III)Structure du programme

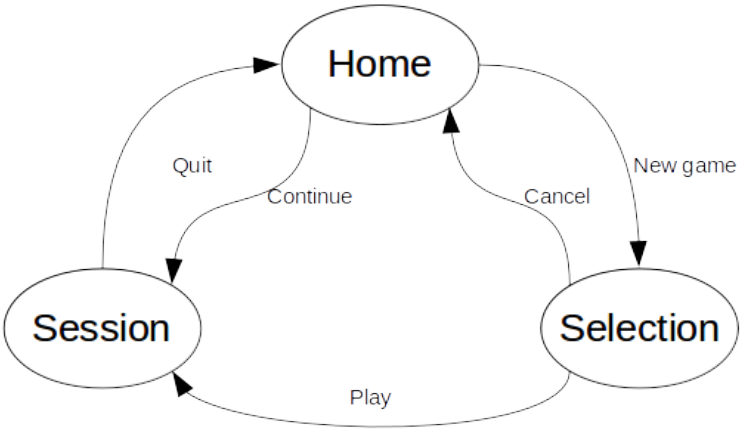
Modèle MVC

Le programme réalisé suit le modèle dit MVC. Il existe en effet une séparation entre les actions du joueur, du rendu et de la logique du jeu. Ainsi les classes comportants le terme *Controller* servent de point de départ de l'action puisque le java est basé sur de l'événementiel. Les classes nommés avec *View* servent uniquement d'interface. Aucun calcul n'est effectué. Cette logique est gérée dans les *Model*. C'est le modèle ensuite qui va interfacer avec la vue.
On peut généraliser la boucle d'événement du programme avec ce diagramme. Le modèle est totalement indépendant de l'action du joueur. Le contrôleur et la vue servent d'interface avec le joueur.



Les menus

L'une des difficulté rencontrée est celle de la conception des menus qui devait répondre à la question suivante : *Comment gérer le changement de d'interface de manière générique ?* La réponse : *Une StateMachine.*



Il suffit ensuite de définir une vue affichée et traduire cette Statemachine en suite de condition afin d'instancier la bonne vue. L'état courant est représenté par une énumération (*ViewState*).

La session et le plateau

La classe responsable du plateau est le *Board.* Cette classe contient les cellules formant alors le plateau. Chaque accès aux cellules doit se faire en passer par le plateau. Le plateau est contenu par la *Session.* La session sert d'intermédiaire entre le contrôleur et le plateau. La génération des mines et l'algorithme de révélation est géré par le plateau.

Le GameModel

Cette classe permet de faire le lien entre la session et les différentes vues du jeu. Lorsque le joueur décide de changer de menu, par exemple passer au menu de sélection, c'est le *GameModel* qui va changer d'état. Au sommet, la *GameWindow* qui possède le *GameModel* est instanciée dans la fonction main.

[***Toute la structure du programme est résumée dans un diagramme disponible en annexe***]

IV)Mécanisme de sauvegarde

Objectifs

Une partie de démineur est représentée par une session. Sauvegarder les données d'une partie revient à sauvegarder les données de cette session. Les données (métadonnées) de la sessions sont gérés par la classe *DataSession.* La séparation des données et de l'implémentation du jeu permet de gérer plusieurs instances de jeu. Le rôle principale de cette *DataSession* est de synchroniser les données de la session entre la mémoire de l'ordinateur et le fichier de sauvegarde.

Problèmes

La principale problématique est celle du protocole de la restitution correct de l'information simples et complexes de la session. Une information simple est par exemple celle de la présence de mine ou non sur une cellule. Une complexe est celle des voisins et/ou du nombre restant de mine dépendant du nombre de cellules marquées par un drapeau. On voit que selon l'ordre de rétablissement de ces informations, elles peuvent changer.

Pour éviter d'obtenir un fichier trop volumineux, nous avons décidé de minimiser la quantité d'information sauvegardée. Ainsi la structure du fichier est assez simple.

BECHET Thomas
TERBAH Dorian

Structure du fichier

Les données nécessaires à sauvegarder sont les suivantes:

-Les dimensions du plateau
-La difficulté du jeu
-Le temps écoulé
-Le plateau (cellules)

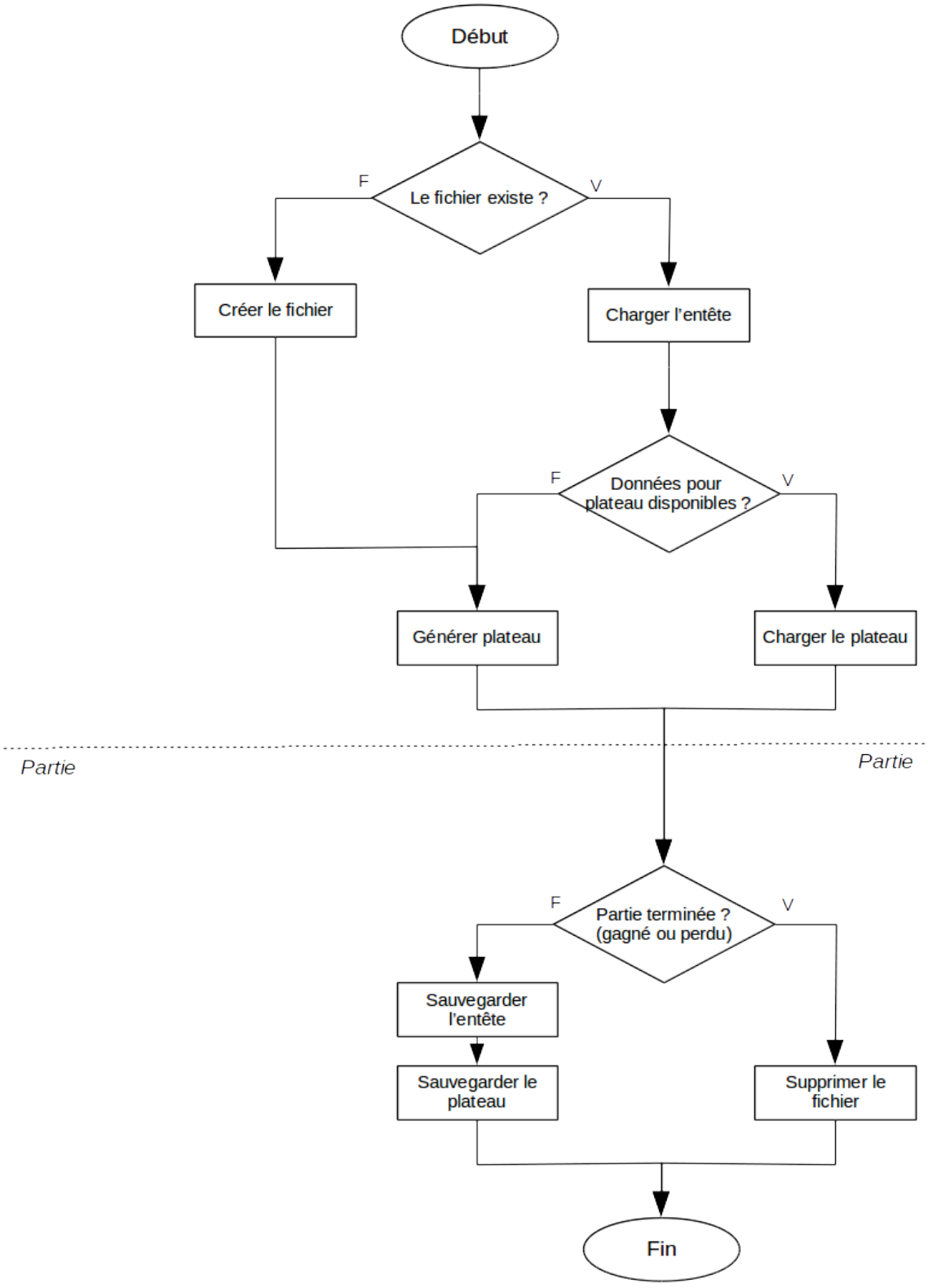
Elles sont séparés en deux parties: l'entête et le corps. L'entête à une taille fixe et le corps non puisqu'il dépend de la taille du plateau.

Entête [*12 octets*]				Corps
Largeur [*byte*]	Hauteur [*byte*]	Difficulté [*short*]	Temps [*long*]	Etats cellules … [*bytes*] …

La largeur et la hauteur du plateau ne dépassant pas 30, on peut stocker les dimensions avec 2 octets. La difficulté, correspondante au nombre de mine peut monter jusqu'à 899 ((30*30)-1). Il faut donc au moins 2 octets pour le nombre de mine sur la plateau. Pour le temps, stocké en seconde, 8 octets sont utilisés. Ensuite, le plateau est sauvegardé grâce à ces cellules de gauche à droite puis de haut en bas. Un octet est nécessaire pour mémoriser l'état d'une cellule. Cet octet comporte l'état visuel de la cellule (flagged, suspicious, revealed, hidden) ainsi qu'un flag déterminant si la cellule est minée.

Les autres informations concernant les cellules tels que le nombre de mine voisine sont calculées de manière différée. C'est-à-dire qu'elles sont déterminées une fois le plateau chargé.

Le diagramme suivant représente le cas classique (sans les éventuelles erreurs) du chemin de vie d'une *DataSession*.



V)Algorithme de révélation

L’algorithme de révélation consiste à révéler d’abords la case cliqué e, puis à révéler les huit cases adjacentes à celle-ci pourvu qu’elles ne soient adjacentes à aucune mine. On recommence cette même opération pour chaque case précédemment révélée de manière récursive, jusqu’à ce qu’on tombe sur soit une case adjacente à au moins une mine, soit une case qui n’existe pas ou soit sur une case qui est déjà révélée. Ci-dessous un exemple de grille de jeu où nous pourrions utiliser cet algorithme.

Comme le montre cette capture d’écran, certaines cases révélées sont marquées par des numéros, qui correspondent en fait au nombre de mines voisines à telle ou telle case.

Pour révéler une case, le joueur doit émettre un clic gauche dessus. Voici ci-dessous les différents cas de figure :

- Si la case est déjà révélée, le clic gauche n’aura aucun effet.

- Si la case possède un marqueur (point d’interrogation ou une étoile), un clic gauche n’aura aucun effet par mesure de sécurité.

- Si la case ne possède pas de marqueur et qu’elle n’est pas adjacente à une mine, alors toutes les cases adjacentes sont révélées de manière récursive jusqu’à ce qu’une case soit adjacente à une mine (cela peut donc provoquer une réaction en chaîne).

- Si la case ne possède aucune marqueur et qu’elle est adjacente à au moins une mine, seule cette case sera révélée, et sur cette case apparaîtra le nombre de mines voisines à proximité.

- Si la case ne possède pas de marqueur et que celle-ci est minée, la partie se termine par la défaite du joueur. De plus, les cases minées non marquées et les cases marquées non minées sont automatiquement révélées, en affichant clairement la mine qui a provoqué la fin de la partie.

Légende:
- M = Minée
- ¬M = Non minée

¬M(0,0)	¬M(0,1)	¬M(0,2)
M(1,0)	¬M(1,1)	¬M(1,2)
¬M(2,0)	¬M(2,1)	¬M(2,2)

Supposons que le joueur émet un clic gauche sur la case d'emplacement (0,2). On vérifie que cette case n'a pas de mines à proximité, ce qui est pour celle-ci le cas. On révèle alors les trois cases adjacentes de celle-ci (et non huit car on sera hors limite du plateau).

Ce sont donc au tour des cases (0,1),(1,1) et (1,2) d'être analysée. Prenons la case (1,2); la case (1,1) étant déjà révélée, rien ne se passe; on révèle par contre la case (2,2) et (2,1) car elles ne sont pas encore révélée et ne adjacentes à aucune mine. La case (1,1) est adjacente à une mine, on ne révèle donc aucune case adjacente. La case (0,1) est également adjacente à une mine, on ne fait donc rien non plus.

Prenons maintenant la case (2,2); comme toutes ses cases adjacentes sont révélées, on ne fait rien. La case (2,1) est quant à elle adjacente à une mine, on ne fait donc rien. On se retrouve maintenant avec la grille suivante :

Légende:
- M = Minée
- ¬M = Non minée
- R = révélée

¬M(0,0)	R(0,1)	R(0,2)
M(1,0)	R(1,1)	R(1,2)
¬M(2,0)	R(2,1)	R(2,2)

Le joueur a maintenant le choix entre les cases aux emplacements (0,0), (1,0), (2,0), en sachant que les cases (0,1), (1,1), (2,1) seront marquées par un 1 (ce qui signifie qu'autour de ces cases figurent une mine). Maintenant, si le joueur clique sur la case à l'emplacement (0,0) ou (2,0), seule la case cliqué sera révélée car elles sont toutes deux adjacentes à une mine. Si le joueur clique sur la case à l'emplacement (1,0), le joueur aura perdu.

Pour résumer la situation, on peut représenter l'algorithme de révélation sous la forme d'une fonction r(x) avec x la case cliqué:

M : nombre de mines alentours

$$r(x)=\begin{cases}\text{On révèle x, } r(x) \rightarrow \text{récursivité} & \text{SI } M=0\\ \text{On révèle x} & \text{SI } M>0\end{cases}$$

V)Conclusion/Impressions

Thomas BECHET

Ce projet nous a permis de réaliser un projet concret avec git et la documentation. Le tout en anglais. Malgré les difficultés rencontrées en rapport avec l'affichage, je trouve le java très productif et organisé (notamment par rapport à sa mise en œuvre avec l'UML). Je garde de cette expérience, une bonne impression de coopération.

Dorian TERBAH

Ce projet m'a permis d'acquérir de nouvelles connaissances et de progresser sur certaines notions en Java. De plus, m'a permis de mieux appréhender l'orienté objet dans un projet concret et de voir comment agencer le modèle MVC afin d'obtenir un résultat organisé.
Je considère ce projet comme une bonne expérience car il nous a permis à Thomas et moi de bien se répartir les tâches dés le début pour être le plus efficace et le plus productif possible.