Projet — Démineur — C++

Bryan Irimia Moles et Dylan Levy Morini

25 février 2018

Table des matières

1 Introduction **3**

2 Règles du jeu **3**

3 Déroulement d'une partie **3**
3.1 Arrêt du jeu (Perte/Victoire) 3
3.2 Initialisation 3
3.3 Un tour de jeu 4
3.4 Fin de jeu 4

4 Modélisation métier **5**
4.1 Classe Board 5
4.2 Classe Square 6
4.3 Classe Player 6
4.4 Classe Chronometer 6
4.5 Classe highScore 6
4.6 Classe Game 7

1 Introduction

Dans le cadre du cours de CPP, nous vous présentons un projet imposé. Il s'agit du célèbre jeu Démineur. Ce rapport consiste à expliquer nos choix ainsi que notre vision du projet.

2 Règles du jeu

Démineur est un jeu de réflexion qui se joue à un seul joueur. Il est composé d'un champ de mines représenté par un plateau rectangulaire de cases le contenue de toutes les cases sont cachées au lancement. Certaines de ces cases peuvent donc contenir des bombes !

Le but du jeu est de dévoiler en un minimum de temps toutes les cases non minées sans faire exploser aucun mine. Tout au long de la partie, le joueur aura la possibilité de dévoilé une case masquée, mais aussi de marqué une case (ou démarqué une case) qu'il pense minée avec comme seul aide, un indice numérique sur chaque case directement voisine d'une bombe.

3 Déroulement d'une partie

3.1 Arrêt du jeu (Perte/Victoire)

Lorsque le joueur dévoile une case minée, la partie s'arrête instantanément. Le joueur a donc perdu !

La partie est gagnante uniquement lorsque le joueur est parvenu à démasquer la totalité des cases non minées. Attention, le fait de réussir à marquer toutes les cases minées ne constitue en rien la réussite du jeu !

3.2 Initialisation

- Le niveau de difficulté est demandé au joueur (le joueur peut choisir longueur et largeur du plateau ainsi que le nombre de bombes qu'il contient varieront en conséquence).
- Le chronomètre est initialisé à 0 (le chronomètre démarre automatiquement à la première action du joueur.
- Calcul du positionnement des bombes sur le plateau.
- Toutes les cases du plateau sont initialisées en état caché.

La première case que le joueur dévoile n'est jamais une bombe, il est donc impératif de ne pas comptabiliser la première action.

3.3 Un tour de jeu

À chaque tour de jeu, le joueur aura la possibilité de :

- Soit marquer une case comme minée.
- Soit dévoiler une case cachée et non marquée
  - Si la case contient une mine, fin du jeu immédiat ! Le joueur a donc perdu.
  - Si la case ne contient pas de mines, on va calculer le nombre total de mines des cases directement voisines (voir fonction (int countNeighbourMinedNumber(Square square)).
- Soit démarqué une case préalablement marquée.

3.4 Fin de jeu

Le jeu se termine perdant, si le joueur dévoile une case minée ou abandon. Le jeu se termine gagnant, uniquement lorsque le joueur réussit à dévoiler la totalité des cases non minées. (voir fonction bool isUnminedSquareZero()).

- Le chronomètre s'arrête.
- Un nom est demandé au joueur
- Le chronomètre ainsi que le nom du joueur sont mémorisés dans la liste des HIGHSCORES.

4 Modélisation métier

Diagramme de classe voir annexe A
Diagramme de classe avec observateur voir annexe B

4.1 Classe Board

Démineur est un jeu qui se joue sur un plateau rectangulaire, nous avons donc décidé de créer une classe Board qui permettra de représenter le plateau du jeu.

Notre Board possède un attribut longueur ainsi qu'un attribut largeur. Cela nous permettra de gérer facilement la taille de notre plateau et dans un futur, pouvoir augmenter la difficulté du jeu. En effet, un joueur pourra choisir son niveau de difficulté et donc la taille du plateau. Par défaut, la taille du plateau de jeu est de 10X10. La Board est composée exclusivement de case. (Représenté par la classe "Square").

Un autre attribut important est le nombre de cases non minées encore présentes dans le jeu. (Soit le nombre de cases non minées que le joueur doit encore découvrir avant de gagner Démineur).

Notre Board possède plusieurs fonctions qui permettront d'effectuer des actions sur le plateau du jeu :

- Deux constructeurs s'offrent à Board. Le constructeur par défaut ainsi qu'un autre constructeur permettant de directement construire une Board aux dimensions adéquates.
- **int countNeighbourMinedNumber(Square square)** : cette fonction va permettre de calculer le nombre de bombes directement voisines d'une case donné en paramètre. Elle est récursive, c.-à-d. qu'elle calculera également le nombre de bombes sur les cases voisines de la case initiale (celle donnée en paramètre) à condition que la case en paramètre ne soit pas une bombe.
- **void reveal(in Coordinate coordinate)** : cette fonction va permettre de révéler une case sur le plateau de jeu. Cette fonction devra prendre en compte le fait qu'une case marquée ne puisse pas être révélée.
- **bool isUnminedSquareZero()** : Cette fonction est une des conditions principales d'arrêt de jeux. Elle retourne "vrai" lorsqu'il ne reste plus aucune case non minée sur le plateau de jeu. Le joueur a donc gagné la partie. Nous avons choisi de placer cette fonction dans la Board, car seul Board est composé de Squares et pour maximiser au maximum la cohésion ainsi que limiter le couplage. Seul Board peut exécuter cette action, car elle est responsable de son plateau.

4.2 Classe Square

Notre plateau de jeu est en effet composé entièrement de cases qui à l'initialisation du jeu sont toutes (sans exception) masquées. Chaque case possède une coordonnée du type (X-Y) afin de connaitre son emplacement sur la Board.

Certaines de ces cases peuvent contenir une mine et toutes possèdent un seul état. Vous retrouverez donc un attribut permettant d'indiquer si la case est minée et un attribut permettant d'indiquer l'état de la case. Ses attributs de position dépendent de la classe Coordinate.

Un autre attribut est le nombre de bombes directement voisines. À l'initialisation du jeu, sa valeur est null. Il sera uniquement mis à jour par le biais de la fonction (**int countNeighbourMinedNumber(in Square square)**). Une fois la case découverte, si celle-ci ne contient pas de bombe, le nombre de bombes voisines sera affiché sur la case. Ce nombre est uniquement affiché lorsqu'il y existe au moins une bombe directement voisine.

Une case peut prendre comme valeur :
- Non minée et caché
- Non minée et découvert
- Non minée et marqué
- Minée et caché
- Minée et marqué
- Minée et découvert

4.3 Classe Player

Cette classe va permettre de représenter un joueur. Tout joueur sera identifié par un pseudonyme˙ le nom du joueur est uniquement demandé lorsque le joueur réussit à vaincre Démineur.

4.4 Classe Chronometer

Cette classe permet de gérer entièrement le chronomètre de la partie. Le temps est compté en nombre de secondes uniquement. Le chronomètre se met en marche lors de la première action du joueur et s'arrête directement lorsque le joueur perd ou gagne la partie.

4.5 Classe highScore

Cette classe permet de gérer tous les highscores des vainqueurs de démineur uniquement. Lorsqu'un un joueur remporte la partie, un nouveau highscore est crée et place dans une liste. Cette liste sera triée dans l'ordre croissant

afin de permettre d'afficher les X meilleurs joueurs.

Le choix d'une liste de structure nous était évident, car il nous fallait un moyen de conserver plusieurs objets, mais aussi de pouvoir les triées.

4.6 Classe Game

Cette classe est notre façade et va nous permettre de «communiquer» avec l'ensemble des autres classes. Elle seule pourra échanger avec le contrôleur.

- **void Start()** : Va nous permettre d'exécuter une partie entière et donc feront appel à l'ensemble des méthodes d'autres classes.
- **bool isOver()** : Va nous permettre de déterminer la fin de partie. Celle-ci continuera temps que isOver() retournera vrai.