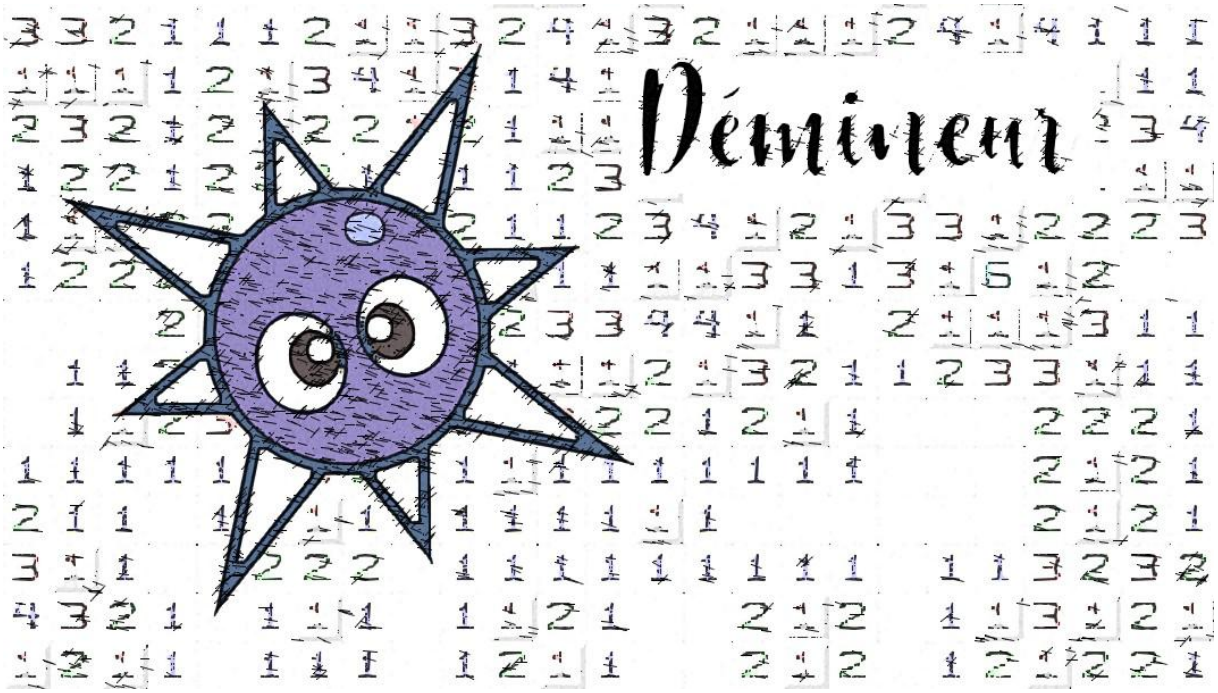
HE2B ESI

Pipers Chris 39864

Directeur : Michel Willemse

Professeur : Vansteenkiste Nicolas

DEV4 - C++II



Remise3

Mai 2018

Executive summary

Le jeu Démineur est un jeu composé d'un tableau de cases. Les cases peuvent être soit vides, soit contenir une bombe.

Le but du jeu est de découvrir toutes les cases qui ne contiennent pas de bombes en un minimum de temps, et cela sans faire exploser une seule bombe. Si le joueur découvre une case contenant une bombe, la partie est terminée.

Si le joueur soupçonne une case de contenir une bombe, il peut la marquer par un drapeau.

Dans le cadre de ce projet, le jeu possédera uniquement un mode solo.

Pour créer l'affichage, le choix fut porté sur 6 classes : ViewCase, ViewBoard, ViewEndGame, ViewParameter, ViewStart et ViewMainWindow.

Sommaire

Executive summary *3*

Introduction *6*

Les écarts *7*

Méthode « vestige » du model **7**

Taille du plateau limitée **7**

Le chrono **7**

La fin de partie **7**

Le palmares **7**

Les bugs *7*

Paramètre de la taille du tableau **7**

Les tests **7**

Les warnings **7**

Problèmes rencontrer *8*

Problèmes de développement **8**

Estimation du temps *8*

Modification de la partie model *8*

Board **8**

Game **8**

Observer – Observable **8**

Partie graphique *9*

ViewCase **9**

ViewBoard **9**

ViewStart **9**

ViewParameter **9**

ViewMainWindow **10**

ViewEndGame **10**

Conclusion *11*

Bibliographie *12*

Introduction

Dans ce document, nous allons aborder le développement du jeu Démineur dans le langage de programmation C++. Dans cette partie, il sera expliqué que la partie graphique du programme et certaines corrections apportées au modèle. Cela signifie que nous allons développer l'affichage du jeu de manière graphique.

Néanmoins, certaines modifications du modèle ont été faites car certains bugs ont été découverts lors du développement de la partie graphique.

Certains bugs peuvent encore éclore.

Les écarts

Méthode « vestige » du model

Certaines méthodes pour l'affichage du plateau et du jeu sont encore présentes. Pour la classe Game, le constructeur sans paramètre est mis en commentaire car, pour la partie model c'était dans ce constructeur qu'étaient gérées les lectures au clavier pour le paramètre du jeu. Cela n'a plus d'utilité, un autre constructeur recevant tout les paramètres a été ajouté.

Taille du plateau limitée

La taille du plateau à été limitée et ce, de manière à avoir un affichage agréable.

Le chrono

Le chrono n'est pas en temps réel, il est mis à jour à chaque « update » du jeu, donc dès que le joueur fait une action. Mais il est correct.

La fin de partie

Pour la fin de la partie, si l'on perd il y a directement une « QMessageBox » qui apparait mais si on gagne il faut refaire un clic pour que la QMessageBox apparaisse.

Le palmares

L'affichage du palmares n'est pas optimal.

Les bugs

Paramètre de la taille du tableau

Pour créer le plateau de taille 5 sur 5 cela fonctionne, un plateau de 13 sur 5 aussi mais un plateau de 5 sur 13 non, il y a une erreur. Cela provient de la classe viewcase, de la méthode mousePressEvent à la ligne 133, le plateau reste des lors à la taille 9 sur 9, alors que partout ailleurs le plateau est de la taille correcte. Pour le 13 sur 5, il n'y a pas de problème mais 5 sur 13 oui. Pour cette dernière dimension et les variantes possibles (ex : 5 sur 12) aucune solution n'a été trouvée.

Les tests

Nous n'avons pas eux le temps de tester la classe game.

La classe time n'as pas réussi à être testée car nous ne savions pas comment. Nous avons essayé avec la méthode sleep ou l utilisation de thread mais cela n'a pas fonctionné.

Les warnings

Aucun warning n'est signalé.

Problèmes rencontrer

Problèmes de développement

Pour le graphique, la création du plateau avec un QGridLayout et du board fût compliquée car à plusieurs endroit il y avait eu des inversions de longueur et largeur et inversement.

La correction de plusieurs bugs a pris du temps.

L'essai de créer un chrono en temps réel a été fait mais cela est resté sans résultats. J'ai pensé à utiliser la méthode sleep mais si le jeu n'est pas testé sur windows cela peut poser des problèmes. L'utilisation de thread n'a pas été essayeé par crainte de non compilation sur les machines de l'école.

Estimation du temps

En termes de temps, l'estimation est à 40h le temps du développement de la partie graphique plus la correction de plusieurs bugs.

Modification de la partie modèle

Plusieurs modifications ont été apportées à la classe Board, la classe Game et du patron Observer_Observable.

Board

Selon le choix de la partie, le nombre de bombes, choisir entre l'initialisation par défaut, par nombre de bombes sélectionnées ou par pourcentage de bombes sélectionnées fonctionne.

Des problèmes liés au positionnement et à l'enlèvement de flag fût corrigés car cela créait des erreurs et arrêts brutaux.

Game

Le constructeur pour la partie modèle a été mis entre commentaire, car pour la partie graphique tous les paramètres sont connus avant la création d'un nouveau « game ». Le nouveau constructeur reçoit tous les paramètres en entrée.

Un attribut boolean win_ a été ajouté pour savoir si une fois la partie finie, si le jeu est gagné ou non.

Observer – Observable

Pour la partie modèle la classe Board dérivait Observable, mais cela créât une erreur lors des tests de la classe board, l'erreur n'a jamais réellement été trouvé.

Pour la partie graphique, la classe Game dérive Observable et la classe ViewMainWindow dérive Observer.

Partie graphique

Pour la partie graphique cela fût porté sur 6 classes. Les 6 classes utilisent des objets de la bibliothèque QT.

ViewCase

ViewCase dérive QLabel. Cette dérivation permet de modifier la couleur de fond ou ajouter une image à l'objet.

Les ViewCase sont utilisées dans le ViewBoard qui hérite de QGridLayout. Chaque case du ViewBoard est un ViewCase et donc un QLabel.

Cette classe représente une case du plateau.

Pour les différentes méthodes, veuillez consulter la documentation du header viewcase.h.

ViewBoard

ViewBoard dérive QGridLayout. Cette dérivation permet de positionner chaque ViewCase dans une grille à une position précise.

Cette classe représente le plateau de jeu.

Pour les différentes méthodes, veuillez consulter la documentation du header viewboard.h.

ViewStart

ViewStart dérive QWidget. Cette dérivation permet d'ouvrir une fenêtre.

Cette classe permet au joueur de lancer un nouveau jeu ou soit de consulter le palmares. Cela introduit le jeu.

Pour les différentes méthodes, veuillez consulter la documentation du header viewstart.h.

ViewParameter

ViewParameter dérive QWidget. Cette dérivation permet d'ouvrir une fenêtre. Cette classe permet au joueur de sélectionner les paramètres qu'il souhaite pour jouer : la taille du plateau, la méthode d'initialisation des bombes sur le plateau et de sont nom.

Certaines limites ont été instaurées pour garantir un affichage agréable ou une sauvegarde simple.

Cette classe permet de paramétrer le jeu.

Selon le type d'initialisation des bombes sélectionné, l'affichage de la fenêtre change.

Pour les différentes méthodes, veuillez consulter la documentation du header viewParameter.h.

ViewMainWindow

ViewMainWindow dérive QWidget et Observer. Ces dérivations permettent d'ouvrir une fenêtre et d'implémenter le patron Observer – Observable. Cette classe contient l'affichage principal du jeu, un ViewBoard, un chrono, le nom du joueur et le nombre de bombes présentes sur le plateau.

Cette classe permet l'affichage principal du jeu.

Pour les différentes méthodes, veuillez consulter la documentation du header viewmainwindow.h.

ViewEndGame

ViewEndGame dérive QWidget. Cette dérivation permet d'ouvrir une fenêtre. Cette classe permet de voir si l'on a gagné ou perdu la partie, mais également savoir le temps mis.

Il est possible dès lors, de relancer une nouvelle partie ou de fermer le jeu.

Pour les différentes méthodes, veuillez consulter la documentation du header viewendgame.h.

Conclusion

La création de la partie graphique via l'utilisation de la bibliothèque Qt.

Les classes utilisées pour la partie graphique sont : ViewCase, ViewBoard, ViewEndGame, ViewParameter, ViewStart et ViewMainWindow.

La difficulté principale de ce travail fût la recherche pour avoir un affichage agréable, aux positionnements des éléments et de faire le lien entre le modèle et la partie graphique.

Tout l'affichage souhaité n'a malheureusement pas pu être fait et ce, par faute de temps.

Bibliographie