Minesweeper

Yueran ZHAO, Sicheng SHEN

Zijing WENG, Yikang SU

groupe ASINSA - 97

Villeurbanne, Avril 9, 2022

Sommaire

[1. Contexte et Objectives du Projet 2](file:///F:\\Minesweeper\\modele_rapport.odt" \l "__RefHeading___Toc110_3221088266)

[2. Description de GUI 3](file:///F:\\Minesweeper\\modele_rapport.odt" \l "__RefHeading___Toc273_3221088266)

[3. Description de l'Algorithme 4](file:///F:\\Minesweeper\\modele_rapport.odt" \l "__RefHeading___Toc275_3221088266)

[4. Diagramme des Classes 5](file:///F:\\Minesweeper\\modele_rapport.odt" \l "__RefHeading___Toc277_3221088266)

[5. Resultats et Analyses 6](file:///F:\\Minesweeper\\modele_rapport.odt" \l "__RefHeading___Toc279_3221088266)

[6. Calendrier du Project et Contributions des Membres 7](file:///F:\\Minesweeper\\modele_rapport.odt" \l "__RefHeading___Toc281_3221088266)

# 1. Contexte et Objectives du Projet

Dans notre projet, nous créons un jeu de démineur. Le Minesweeper (Démineur) est un jeu de réflexion dont le but est de localiser des mines cachées dans une grille représentant un champ de mines virtuel, avec pour seule indication le nombre de mines dans les zones adjacentes.

Chaque case de la grille peut soit cacher une mine, soit être vide. Lorsque le joueur clique sur une case libre comportant au moins une mine dans l'une de ses cases avoisinantes, un chiffre indiquant ce nombre de mines apparaît. Si en revanche toutes les cases adjacentes sont vides, une case vide est affichée et une zone vide est entièrement délimitée par des chiffres. En comparant les différentes informations récoltées, le joueur peut ainsi progresser dans le déminage du terrain. S'il se trompe et clique sur une mine, il a perdu.

On peut signaler les cases contenant des mines présumées par un drapeau en cliquant sur le bouton droit de la souris et de cliquer encore une fois sur le bouton droit pour changer le drapeau en marque de question.

Dans notre projet, on a réaliser le jeu Minesweeper avec des fonctionnalités supplémentaires et des propriétés paramétrables.

En utilisant nos connaissances en objet orientée et interface homme machine (IHM), nous avons des paramètres personnalisables, notamment la taille du champ, le nombre et l’emplacement des mines, la difficulté, la texture, le language du jeu et un IA “avancée” pour guider les joueurs.

Le IA "avancée" donne des indications dans le bloc pour le joueur et peut automatiser le déminage. Le taux de réussite doit être le plus haut que possible.

# 2. Description de GUI

Sur notre interface graphique GUI nous pouvons accéder à la modification de plusieur propriétés paramétrables par les menus GAME et OPTION situés dans le coin en haut à gauche de la fenêtre du jeu.

Dans le menu GAME, il y a NEW, RESTART, différents niveaux de difficulté (*BEGINNER, INTERMEDIATE, EXPERT, CUSTOM). CUSTOM* est une fonctionnalité de dessin (mettre soi-même les mines pour le debug et les démonstration).

Dans le menu OPTION, il y a LANGUAGE et TEXTURE. En language, nous avons chinois, français et anglais comme choix. Dans la texture, nous exposons 4 choix qui sont du style *xp, win7, minecraft et mario*. Les styles *xp* et *win7* sont des rétablissements de l’interface xp et win7 avec les mêmes positions de chaque élément. Le style *minecraft* est réalisé par nous même selon le jeu Minecraft et le style *mario* est est réalisé par Jonathan Aldrich from Minesweeper X.

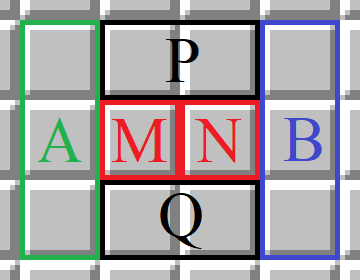
# 3. Description de l'Algorithme

Pour avoir un IA "avancée" qui donne des indications dans le bloc pour le joueur et qui peut automatiser le déminage, nous devons implémenter certains algorithmes. Dans cette partie, nous présentons la partie “scientifique” d’algorithmes de notre IA du projet.

3.1 Analyse de base

Vérifier si toutes les cases avoisinantes d’une case donnée est des mines ou sont tous vides.

3.2 Analyse de soustraction

Vérifier les chiffres des deux cases adjacentes et déterminer si nous pouvons signaler ou cliquer en toute sécurité les cases avoisinantes. La logique spécifique est la suivante :

Supposant que les deux cases aux milieux sont les cases chiffrées. On pose et le nombre de mines restant qui doivent être placées dans les cases avoisinantes (le chiffre affiché dans la case moins le nombre de mines avoisinantes déjà déduis). Puis, représentent respectivement le nombre de mines restant dans les zones encadrées. On a évidemment :

Donc,

Si le nombre de case libre dans la zone est égal à , on peut déduire que tous ces cases sont des mines, et toutes les cases libres dans la zone ne sont pas des mines.

Par exemple, en cours de jouer le démineur, on utilise beaucoup l’analyse de soustraction. Il est la logique la plus fondamentale et la plus importante du démineur. Il existe de divers usages de l’analyse de soustraction, mais on n’a choisi que la situation de deux cases adjacentes parce que c’est le cas plus utilisé est plus facile à appliquer.

3.3 Analyse de probabilité

Calculer la probabilité de mine de chaque case par compter tous les jeux possibles. Si la probabilité égale à 0, la case est sécurisée. Tous les cases qui peuvent être déduits par la logique humaines peuvent être déduit par cette analyse, mais on utilise en priorité les deux analyses précédentes parce qu’ils sont beaucoup plus rapides. En plus, s’il n’y a pas des cases à probabilité 0, on va deviner une case parmi les cases avec la plus faible probabilité de mine.

On peut calculer la probabilité de mine une case par :

À cause ça grande complexité en temps, on utilise « diviser pour régner ». D’abord on trouve tous les composants connectés. On définit un composant connecté (CC) comme un ensemble de points, qui indique une zone connectée sur le bord des nombres, dans laquelle les case peuvent interagir les unes avec les autres. Toutes les cases libres ou les cases de question autour d'une case chiffrée sont dans le même CC. Nous mettons donc la première case chiffrée à l'intérieur d'une queue, et chaque fois que nous trouvons une nouvelle case du CC, nous mettons les cases chiffrées avoisinantes de cette nouvelle case dans la queue et faire un BFS pour trouver le CC en entier.

Soit il y a composants connectés en total, on définit le numéro de CC,

le numéro de case dans un CC, et le nombre de mine dans un CC.

Pour chaque on trouve toutes les configurations pour des nombre de mines total différents en utilisant le retour arrière (backtracking). On compte le nombre de configurations total , et aussi les nombres des configurations où la case est une mine .

On définit la nombre de configuration de l’ensemble de premiers composants connectés avec le nombre total de mine . On a :

Dans le programme on utilise surtout , la nombre de configuration de l’ensemble de tout les CC sauf . On calcul donc les et par programmation dynamique utilisant la première équation, puis utilisant la deuxième équation.

Il existe encore cases inconnues qui ne sont pas dans les CC. Ces cases sont isolées. La nombre de configuration de ces cases avec mines peut être calculé par combinaison mathématique :

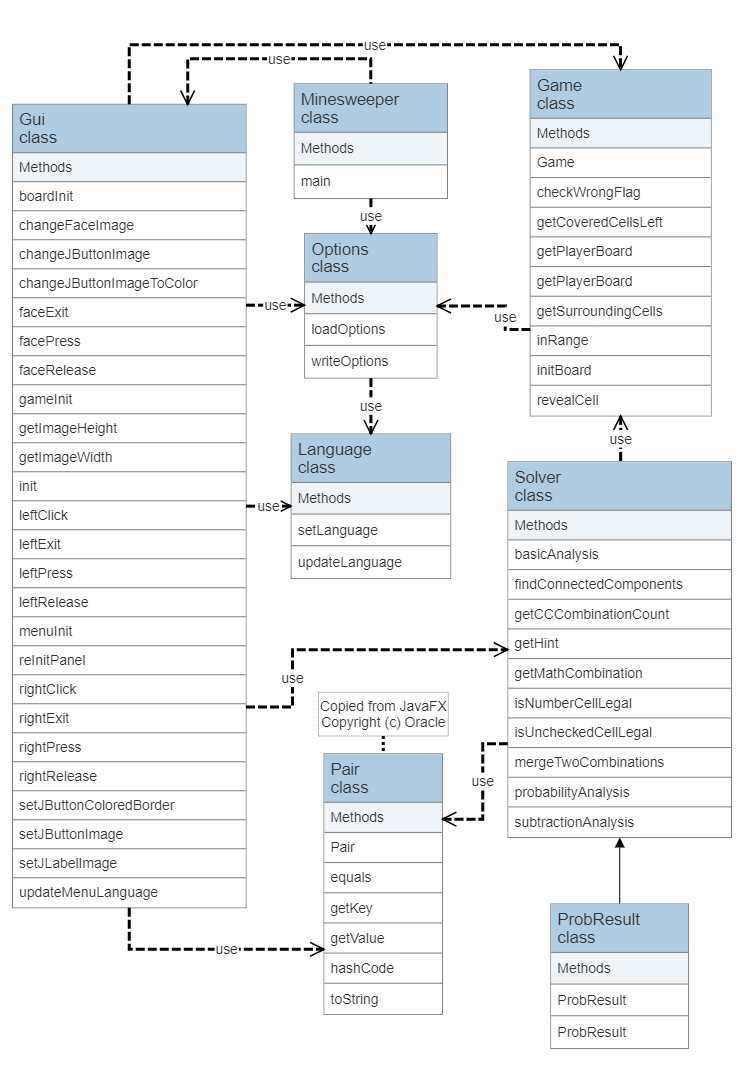
La nombre des configurations de mines de l’ensemble de et les cases isolées :

Finalement, pour tout cases dans un CC, ou bien , sa probabilité de mine est :

Et pour les cases isolées, ses probabilités de mine sont identiques. Selon la théorie des probabilités, la somme de probabilité de toute cases inconnues est égale à . On peut calculer la probabilité de mine de case isolée :

# 4. Diagramme des Classes

Notre programme contient 8 classes: Minesweeper, Game, Gui, Options, Solver, Language, Pair et ProbResult. Dans le diagramme UML, nous n’affichons que les méthodes car il y a beaucoup trop d’attributs.



# 5. Resultats et Analyses

Dans cette partie, nous allons présenter les résultats de nos projet puis faire des analyses sur le travail.

5.1 Les objectives atteints

Dans notre projet, on avons pu réaliser le jeu Minesweeper avec des propriétés paramétrables.

Nous avons crée un menu où nous avons les options *NEW, OPTION, RESTART, BEGINNER, INTERMEDIATE, EXPERT, CUSTOM et LANGUAGE* pour paramétrer personnellment. Nous proposons une possibilité de modification des paramètres incluent la taille du champ, le nombre et l’emplacement des mines, le niveau de difficulté, la texture et le language du jeu.

Dans la texture, nous exposons 4 choix qui sont du style *xp, win7, minecraft et mario*. En language, nous avons chinois, français et anglais comme choix.

A part cela, nous avons un IA “avancée” qui donne des indications dans le bloc pour le joueur et peut automatiser le déminage.

5.2 La partie la plus difficile du travail

La partie la plus compliquée de la programmation est celle de l’IA.

Les références sont écrits en Javascript et il y a beaucoup de fonctionnalités, notamment des élagages dans l’algorithme de recherche.

Nous avons diminué la complexité du temps de l’analyses de probabilité et utilisé la programmation dynamique pour écrire l’algorithme.

5.3 L’imperfection

Nous voulons avoir un meilleur rétablissement de l’interface win7 mais il est difficile d’avoir la même position du chronomètre. Nous n’avons pas les même effets en cliquant la souris de gauche et droite en même temps.

5.4 L’amélioration

Pour le AI, nous espérons réaliser un calcul de taux de réussite pour chaque case du champ de mine

à fin de donner de meilleurs conseils aux joueurs. Nous espérons également ajouter un teste pour caulculer le taux de réussite avec le IA et de recorder le meilleur score du joueur.

# 6. Calendrier du Project et Contributions des Membres

Le travail du projet a été divisé en plusieurs parties lors de notre première discussion. Nous avons divisé les parties en jeu Minesweeper, fonctionnalités (clics de la souris, dessin, changement de language), GUI, IA, menu, ressource et rapport.

Après cela, chacun choisit et fait sa partie dans quelques jours et en prend une autre, ainsi de suite.

Lors des cours de chaque semaine, nous faisons le debug et ensuite fusionnons notre branche dans notre programme principal.

Github est notre outil collaboratif. Dans ce site, nous pouvons publier notre projet et partager notrecode. En utilisant Github, nous avons pu créer plusieurs branches pour que chacun travaille son code et le fusionne après.

Voici les contributions des membre:

Yueran ZHAO : fonctionnalité de dessin (mettre soi-même les mines pour le debug et les démonstration), ressource, rapport

Sicheng SHEN : lecture du dossier *OPTION*, création du menu, changement de language

Zijing WENG : IA "avancée", mise en page de la texture

Yikang SU : jeu Minesweeper inclus l’installation du début de jeu, fonctionnalités des clics de la souris

Bibliographie