Rapport de projet 2048

Daubasse Biffert Gallardo

22 avril 2015

Table des matières

1 Présention du jeu 2048

Le but du jeu est de fusionner des nombres ensemble (puissances de deux) afin d'atteindre le nombre ultime '2048' et gagner la partie. L'aire du jeu 2048 est une grille de quatre lignes par quatre colonnes avec donc 16 cellules carrées. Chaque cellule peut être vide ou contenir un nombre. Au début du jeu, il y a

deux carrés (également appelés « tuiles ») avec un chiffre '2' ou un '4' suivant votre chance à l'intérieur.

Lorsque vous parvenez à faire entrer en collision 2 briques avec le même numéro dedans, elles fusionnent en une seule nouvelle brique dont le numéro sera l'addition des deux nombres précédents : 2+2=4, 4+4=8, ... 1024+1024=2048 !

Pour déplacer les briques sur la grille, vous devez juste choisir une direction (haut, droite, bas ou gauche). Toutes les briques vont se déplacer dans la direction choisie, jusqu'à ce qu'elles fusionnent avec une brique de même valeur ou bien qu'elles soient bloquées par une brique avec un numéro différent. A chaque mouvement une tuile va aléatoirement apparaitre dans une case vide, cette tuile a une chance sur dix d'être égale a quatre sinon elle serra égale à deux.

Ici on utilisera simplement les quatres flèches directionnelles du clavier pour déplacer les briques.

2 Implémentation de la grille

2.1 Fonctions simples

2.1.1 Valeur d'une tuile

On donne les coordonnées et la grille afin de directement retourner la valeur de la tuile situé dans cette grille et à ces coordonnée.

2.1.2 Initialiser une tuile

On donne les coordonnées et la grille afin de directement modifié la valeur de la tuile situé dans cette grille et à ces coordonnées.

2.1.3 Valeur du score

On donne la grille et la valeur du pointeur du score de la grille est retourner.

2.2 Structure

Nous avons choisis d'implémenter la grille par un pointeur de pointeur d'entier pour la grille et d'un entier représentant le score. Le pointeur nous sert à crée un tableau deux dimension qui servira de grille, nous avons choisis la solution du pointeur car elle permet de facilement étendre notre grille a une taille supérieure a quatre, le tableau deux dimensions quand a lui nous semblait la solution la plus naturelle pour crée une grille c'est pourquoi nous n'avons pas implémenter la grille avec un tableau une dimension.

2.3 Instanciation d'une grille

Pour l'instanciation, dans la fonction new_grid, d'une grille on alloue la mémoire pour une stucture de grille puis on initialise le score a zéro et ensuite

on alloue le tableau deux dimensions en initialisant chacune des tuiles a zéro. On retourne ensuite la structure nouvellement crée.

2.4 Suppression d'une grille

Pour supprimer une instance de grille nous prenons en argument une stucture de grille et parcourons le tableau deux dimensions et nous libèront la mémoire allouer pour les pointeurs du pointeur tiles de notre structure puis on libère la mémoire prise par le pointeur tiles lui même et enfin on libère la mémoire de notre instance de grille.

2.5 Copie d'une grille

Pour copier une grille nous prenons en paramètre deux grille, src la source et dst la destination, nous parcourons ensuite le tableau deux dimension afin de copier chaque tuiles de src et de la placer dans la grille de dst une fois cela fait on copie le score de src et on le place dans le score de dst.

2.6 Mouvement de la grille

2.6.1 Possibilité de mouvement

Dans la fonction can_move nous prenons en argument une grille et une direction. L'idée ici est de se dire que si une tuile peut bouger alors l'ensemble de la grille le peut c'est pourquoi nous initialisons une variable booléenne a faux qui serra ensuite modifié, au fur et a mesure du parcours de notre tableau, si, en prenant une dirction donnée, deux tuiles consécutives sont égales mais différentes de zéro ou si la tuiles vérifié est égale a zéro, c'est à dire que l'on a un trou entre notre tuile et ce qu'il y a aprés que ce soit une autre tuiles, le bord de la grille ou encore un zéro. On renvoie ensuite notre variable booléenne.

2.6.2 Mouvement

Pour le mouvement nous avons utilisée tois fonctions qui nous sont propre que sont array_to_grid, grid_to_array et compute_array.

- Passage d'un tableau à une grille et d'une grille à un tableau :

Ces fonctions servent à transformer une grille en une ligne (tableau 1D de taille size) selon les parametres de position donnés, c'est à dire le paramètre x donnés en argument, et sans prendre en compte les zeros ensuite pour repasser d'une grille a un tableau on retransforme donc notre ligne 1D en grille en combalnt les tuiles vides par des zéros. En fonction du sens dans lequel nous voulons faire notre mouvement nous inversons ou non notre tableau, à l'aide de la foncton invert_array, afin de traiter toute les directions de la même manière.

- Calcul du score :

La fonction compute_array prend un tableau 1D et une taille, initialise un score à zéro et ensuite parcours ce tableau et pour chaque cases consécutive et

identique du tableau les fussionent ajoutant leur somme à score qui est ensuite renvoyer à la fin de la fonction.

En utilisant ces fonction il est donc plus facile de faire le do_move puisque que nous transformons notre grille en tableau 1D puis calculons notre nouvelle grille et le score dans ce tableau pour enfin remettre ce tableau dans notre grille.

2.7 Ajout d'une tuile

Ici nous créons un tableau deux dimensions et copiont les cases vides de notre grille prise en argument pour ensuite choisir hasard parmis ces cases celle que l'on placera à deux (ou quatre suivant le rand()). Nous choisisons de faire un rand entre 0 et 1000 car en effet le rand() rend des valeurs plus stable s'il est pris entre 0 et 1.

2.8 Jouer un coup

Jouer un coup reviens à vérifier que l'on peut bouger dans la direction demander par l'utilisateur et si c'est le cas faire le mouvement avec le do_move et ajouter une tuile.

2.9 Fin du jeu

Nous détectons la fin du jeu quand aucune des directions n'est jouable, c'est à dire quand on ne peut plus bouger de tuile.

3 Test sur grid.c

3.1 Fonctions générales

Dans test_grid.c nous avons essayer de vérifier le plus de code possible automatiquement. Pour nous aider nous avons crée une fonction result qui affiche, avec un code couleur, si le test est passer avec succés ou non. D'abord on teste que new_grid crée bien une grille vide, on vérifie donc que chaque tuile est nulle. Ensuite on teste que la copy_grid copie bien toute la grille c'est à dire le tableau deux dimension doit etre le même et le score lui aussi doit être copié. On crée donc deux grilles et vérifions à l'aide de la fonction equals du fichier grid_utilities que chaque cases du tableau sont les même puis que les scores sont les même. Le test du score se fait juste par l'anticipation du score de la grille, on fixe donc deux cases et on vérifie que le score de la grille est bien le bon. Pour le test du can_move nous créons deux grilles chacune avec une case occupée, une dans un coin et l'autre entourée par des cases vide. On vérifie ensuite que la première, celle avec la case dans le coin ne peut bougée que vers les directions opposées au coin quand à la deuxième, celle avec la case entourée de cases vides, doit pouvoir bougée dans tout les sens. Avec la deuxième grille nous testons notre do_move en bougeant vers le haut puis vers la gauche et si

on peut bougé dans toute les directions sauf le haut aprés avoir bougé en haut puis que l'on peut bouggé partout sauf vers le haut et la gauche après avoir bougé vers la gauche alors le test est passer avec succés. Enfin pour le add_tile on fait autant de add_tile que de case puis on vérifie que toute les cases sont occupées.

3.2 Fonction perso

Afin d'éviter tout bug innatendu nous avons décider de tester nos fonctions qui se trouvent dans grid_utilities. On remplit donc un tableau de zéro à GRID_SIZE (qui est la taille de toute nos grilles) et on l'inverse à l'aide de notre fonction, on vérifie alors que le tableau est bien inversé. Pour tester la fonction grid_to_array on remplit trois cases d'une grille de façon à avoir les même valeur que le tableau précédent dans les cases du haut, on applique notre fonction a la grille et on vérifie que nos tableaux sont identique. Pour le calcul de la nouvelle grille après mouvement on prend notre tableau de grille et l'on vérifie que ses valeurs sont bien les bonnes, c'està dire qu'elles doivent être rangée dans l'orde décroissant. Pour finir la fonction array_to_grid on prend notre grille que l'on avait au préalable transformé en tableau que l'on retransforme en grille et on vérifie ensuite que la valeur des tuiles du haut corresponde au tableau qui nous sert de référence dans les autres fonctions. Si ce sont les même la fonction ne contient pas de bug.

4 Stratégie pour automatiser la résolution du jeu