Projet "Jeu 4096"

R. Craciun P. Ledoyen

septembre 2016

1 Objectif du projet

— Réaliser un jeu de type "2048"

— Fonder l'organisation du programme sur le modèle MVC

— Intégrer des options et modularités : difficulté, interface, mode de fusion, aide

2 Organisation générale

Nous avons progressé à deux, sans séparer en deux parties distinctes (ex : d'un côté les calculs, de l'autre l'affichage). Cela nous a permis de mettre en commun rapidement des morceaux de programme, et de surmonter les problèmes plus efficacement.

Nous avons commencé par concevoir l'affichage du plateau de jeu, puis la fusion classique, et à partir de cette base fonctionnelle, nous avons développé les modules supplémentaires.

3 Découpage des fichiers

L'ensemble du code est réparti en différents fichiers .h et .c :

— `main.c` : contient le main uniquement.

— `4096.h` : contient les macro-constantes de taille (grille de jeu, fenêtre), la déclaration du plateau de jeu en variable globale, et les énumérations de boutons et états de paramètres (interface, fusion, difficulté). C'est le seul fichier commun à tous les autres.

— `calculs.[c|h]` : contiennent les fonctions logiques → correspondent à la partie "Modèle".

— `affichage.[c|h]` : contiennent les fonctions d'affichage → correspondent à la partie "Vue".

— `controle.[c|h]` : contiennent les fonctions de contrôle des boutons → correspondent à la partie "Contrôleur", en parallèle avec le `main`.

4 Explicitation des fonctions principales

Tout d'abord, il est important de noter que le tableau de jeu `plateau([T_GRILLE][T_GRILLE])` contient les *puissances* de 2, et non pas les résultats des puissances de 2. Nous verrons dans la suite pourquoi. (`T_GRILLE` est une macro-constante définissant le nombre de cases dans une ligne ou une colonne)

4.1 Affichage du plateau

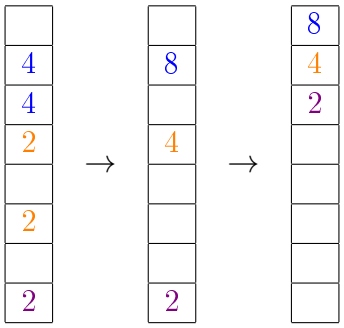
Fonction `void affiche_plateau(modeAffichage choix)`

Cette fonction parcourt toutes les cases du tableau et les dessine une par une. Le bloc d'instructions successives de dessin correspond à l'arrondissage des coins des cases. La couleur des cases est appelée dans un tableau constant de couleurs en trois dimensions : `static COULEUR palette[2][2][13]`. La 1^{ere} dimension permet de choisir le mode d'affichage (variable `choix`), la 2^{eme} oriente vers la couleur du fond de tuile (0), ou la couleur du texte à afficher sur la tuile (1), et la 3^{eme} permet d'accéder à la couleur associée à la *puissance* de 2 à afficher.

Un test sur la valeur permet de ne pas afficher "0" dans les cases vides.

4.2 Fusion classique

Illustration de l'opération pour un déplacement vers le haut, sur une colonne :



A chaque déplacement (haut, bas, gauche ou droite) correspond une fonction adaptée pour les opérations sur les lignes et colonnes. Nous détaillerons ici le déplacement vers le haut, les 3 autres fonctions étant ainsi très similaires.

Procédure

A : On parcourt les colonnes du tableau

B : Pour chaque colonne

1 : On parcourt ses lignes de haut en bas (ici)

2 : On cherche la prochaine tuile non nulle, qui devient alors tuile *de référence* (d'indice `iSelec`)

3 : A partir de cette tuile de référence, on cherche la prochaine tuile non nulle.

- Si elle est de même valeur que la tuile de référence, alors on double la valeur de la tuile de référence et on annule la tuile trouvée.
- Dans le cas contraire, on laisse en l'état, et la tuile suivante devient la nouvelle référence.

4 : On reparcourt ensuite la colonne pour déplacer un à un les résultats obtenus (fusions ou tuiles laissées en l'état non nulles) dans la case libre la plus haute (d'indice `iPlace`)

5 : Une fois toutes les tuiles non nulles déplacées, on complète la colonne avec des zéros (on efface bien ainsi toutes les anciennes valeurs)

4.3 Aide sur la fusion classique

L'aide à la fusion classique est assez rudimentaire : par l'intermédiaire d'une variable comptant le score, il est possible d'utiliser les fonctions de déplacement en fusion classique sans jamais modifier les valeurs du plateau. On introduit pour cela une variable booléenne de garde : `doitJouer`, dont la valeur est donnée en paramètre. Une partie du code est alors ignorée dans ces fonctions de "déplacement".

En pouvant donc calculer le score d'un déplacement, on est capable de dire lequel rapporte le plus de points, et ainsi le proposer au joueur.

5 Conclusion