ESIEE Paris

IN3R12 - Programmation C

Projet Sokoban dynamique
Documentation développeur

Auteur :
M. Kévin Ta

Professeur :
M. Damien Masson

26 juin 2016

ESIEE PARIS

Table des matières

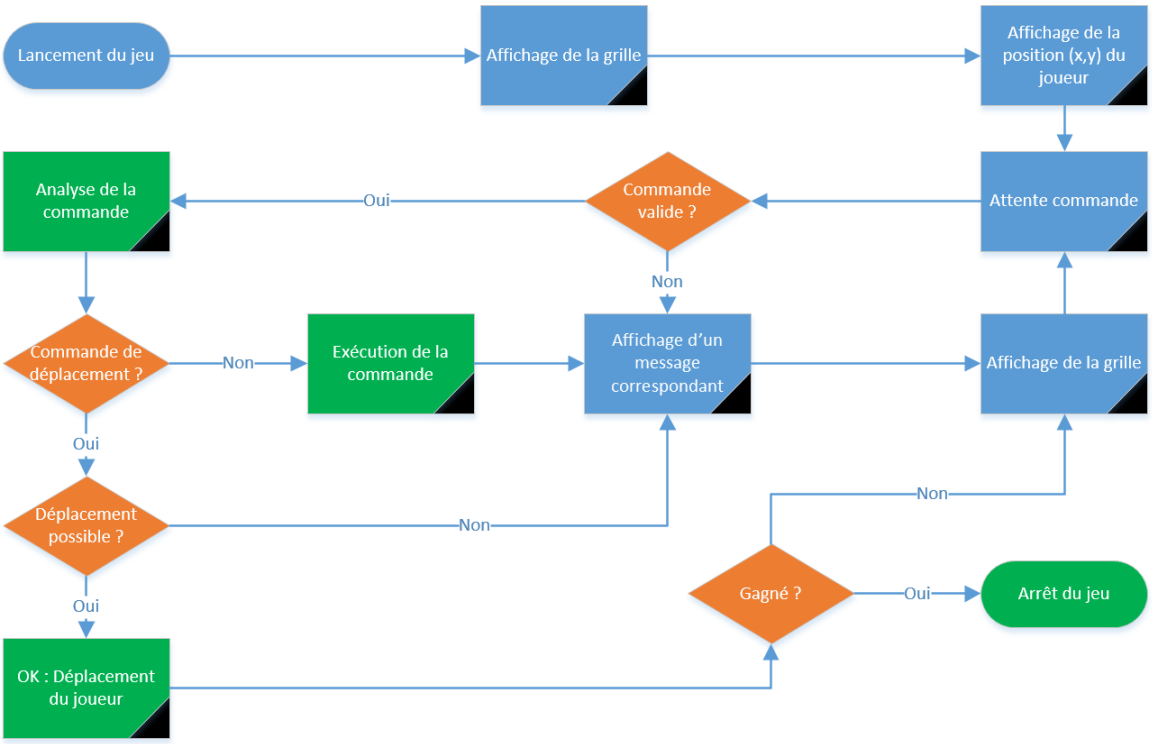
1 Organisation du programme

Le programme a été séparé en plusieurs fichiers afin de faciliter la compréhension du code source. Les fichiers *.c contiennent l'intégralité des fonctions utilisées dans le programme. Les fichiers *.h sont utilisés pour la déclaration des librairies, des constantes, des structures, des enums et des fonctions.

Plus précisément, les fichiers sokoban.* représentent le coeur du programme avec toutes les fonctionnalités minimums et les fichiers pileGrille.* sont utilisés pour la fonction "Undo".

De plus, un fichier grille1.sok contenant la grille peut être aisément modifié, afin que l'utilisateur final puisse personnaliser le niveau.

2 Workflow



Les étapes du programme ont été conçu de manière simple afin de ne pas perdre le joueur. L'étape d'initialisation du programme comprend un affichage de la grille et de la position du joueur dans celle-ci. Le programme entre dans une boucle jusqu'à ce que le joueur gagne. Dans cette boucle, le programme envoie un prompt afin que le joueur puisse taper des commandes. Le programme analyse la commande tapée et l'exécute. Si le joueur se déplace, un test de déplacement est utilisé. Une fois la commande exécutée, le programme ré-affiche la grille avec les éventuelles modifications, et renvoie un prompt.

3 Fonctions du programme

3.1 PileGrille.h

3.1.1 freeCell

Cette fonction prend en paramètre un pointeur de type Cell. Elle permet de libérer la mémoire allouée à la Cell, tout en libérant la mémoire allouée à son attribut Grille.

3.1.2 freeList

Cette fonction prend en paramètre un type List. Si la variable List n'est pas NULL, cette fonction libère la mémoire allouée à la variable List par récurrence. En effet, une variable de type List est composé de Cell. En libérant la mémoire de chaque Cell qui compose la List, il est certain que toute la mémoire allouée à la variable List sera libérée.

3.1.3 newCell

Cette fonction retourne un pointeur de type Cell. Elle permet de créer un nouveau Cell et l'allouant dans le tas et en initialisant son attribut next à NULL.

3.1.4 pop

Cette fonction prend en paramètre un pointeur de type List. Elle permet de supprimer la première cellule de la liste passé en paramètre. Pour cela, la fonction remplace le pointeur de List par le pointeur de List->next qui aura pour résultat de remplacer la première cellule par la deuxième. La fonction libère ensuite la mémoire allouée par la cellule avec la fonction freeCell.

3.1.5 push

Cette fonction prend en paramètre un pointeur de type List, un type Grille et un type Position. Elle permet de placer une cellule en première position de liste. Pour cela, cette fonction crée un pointeur de type Cell avec la fonction newCell, copie la grille passée en paramètre dans son attribut grille, copie la position passé en paramètre dans son attribut position, place la Cell en première position de la liste en initialisant son attribut next avec le pointeur de la liste, puis remplace le pointeur de la liste par le pointeur de la cellule.

3.2 Sokoban.h

3.2.1 afficheGrille

Cette fonction prend en paramètre un type Grille. Elle permet d'afficher une grille par itération.

3.2.2 aide

Cette fonction affiche un texte d'aide.

3.2.3 compte

Cette fonction prend en paramètre un type Grille et un caractère et renvoie un nombre. Elle permet de calculer le nombre d'occurrence d'un caractère passée en paramètre dans une grille passé en paramètre.

3.2.4 copieGrille

Cette fonction prend en paramètre un type Grille et renvoie un type Grille. Elle permet de copier le contenu de la grille passée en paramètre dans une nouvelle grille qui aura été créé par la fonction creerGrille.

3.2.5 creerGrille

Cette fonction prend en paramètre 2 variables de type Integer et renvoie un type Grille. Elle permet de créer et d'allouer une grille dans le tas, en allouant d'abord les lignes, puis ensuite les colonnes par itération.

3.2.6 deplace

Cette fonction prend en paramètre 2 variables de type Grille et 2 variables de type Position. Elle remplace, dans la grille de jeu, le symbole à la position de destination par le symbole de la position de départ puis, remplace le symbole à la position de départ par du vide ou une cible.

3.2.7 error

Cette fonction prend en paramètre un type CodeErreur. Elle permet d'afficher le message d'erreur correspondant au code d'erreur envoyé en paramètre.

3.2.8 freeGrille

Cette fonction prend en paramètre un type Grille. Elle permet de libérer la mémoire de la grille par itération.

3.2.9 gagne

Cette fonction prend en paramètre 2 variables de type Grille et renvoie un type Boolean. Elle permet de tester si le joueur a gagné en détectant par itération sur la grille passé en paramètre si toutes les cibles ont été remplies et si le joueur ne se trouve pas sur une cible.

3.2.10 getSokoban

Cette fonction prend en paramètre un type Grille et retourne un type Position. Elle permet de retourner la position du Sokoban par itération à travers la grille. Pour cela, elle repère le caractère "S" dans la grille et affecte les attributs x et y de la position avec les variables de boucle i et j.

3.2.11 initGrille

Cette fonction prend en paramètre une chaîne de caractère (nom de fichier) et renvoie un type Grille. Elle permet de créer une grille à l'aide d'un fichier .sok. Ce fichier doit contenir la hauteur, la largeur, ainsi que la grille. Cette fonction fait appel à la fonction creerGrille afin d'allouer la grille dans le tas, et ensuite procède à la copie fichier->grille à l'aide d'une itération. NB : Dans l'itération, la largeur a été additionné de 2 afin que la fonction fgets puisse capturer les caractères "\0" et "\n".

3.2.12 joue

Cette fonction prend en paramètre 2 variables de type Grille, un type Commande et un type Position et renvoie un type Position. Elle permet de calculer la future position du joueur en utilisant la fonction pas, elle vérifie que cette future position est une destination possible en utilisant la fonction possible. Si cette fonction possible renvoie true, le joueur est déplacé en utilisant la fonction deplace, sinon, si c'est à cause d'une caisse, la fonction tente de déplacer la caisse pour pouvoir déplacer le joueur.

3.2.13 nouveauJeu

Cette fonction prend en paramètre un type Grille. Elle appelle à la fonction verif et ensuite remplace le Sokoban et les caisses avec du vide par itération.

3.2.14 pas

Cette fonction prend en paramètre un type Position et un type Commande et renvoie un type Position. Elle permet de calculer la position future en fonction de la commande passée en paramètre à partir d'une position de départ passée en paramètre.

3.2.15 possible

Cette fonction prend en paramètre un type Grille et un type Position et renvoie un Boolean. Elle teste si la position passée en paramètre est une position valide dans la grille passée en paramètre. Elle renvoie true seulement si la position correspond à une cible ou du vide. Retourne false dans les autres cas.

3.2.16 verif

Cette fonction prend en paramètre un type Grille. Elle permet de valider la grille afin de pouvoir jouer. Par itération, cette fonction calcule le nombre de caisses et le nombre de cibles présents dans la grille en appelant la fonction compte. Si ces nombres sont différents, le jeu ne démarre pas.

3.3 Les structures

2 structures ont été créées afin de gérer la position et la grille. Ces structures ont été définies en tant que type Position et Grille. La structure Position contient les attributs int x et int y, et la structure Grille contient les attributs int hauteur, int largeur, et un pointeur de type Ligne.

4 Choix de conception

Le programme a été conçu de manière à séparer le maximum de fonctionnalités et de factoriser le code. En effet, un code factorisé sera moins complexe, plus facile à reprendre et à débugger. De manière à alléger les ressources du projet, il a été décidé que le programme sera codé de manière dynamique avec un minimum de copie de variables.

5 Ajout de fonctions

Si vous souhaitez ajouter des fonctions supplémentaires, vous devez dans un premier temps ajouter le prototype de votre fonction dans le fichier sokoban.h et placer le contenu de votre fonction dans le fichier sokoban.c. Il est également possible de faire appel à d'autres librairies. Dans ce cas-ci, il faut placer votre include dans le fichier sokoban.h.