Rapport du jeu Sokoban

Baskevitch Claire 21500568

January 2, 2017

Part I

Présentation du programme

Chapter 1

Les structures

- La première structure est celle du plateau:
Elle est définie dans le fichier "constantes.h". "struct plateau" est composée de deux autres structures, "struct une_case" qui permet d'enregistrer si une case est vide, une caisse, un mur etc et "struct coordonnées" qui dans la structure du plateau sert à avoir les coordonnées du personnage. "struct plateau" prend donc en arguments les coordonnées du personnage et un tableau à deux dimmensions de type struct une_case et de la taille largeur sur hauteur, définies dans le fichier "contantes.h".

- La deuxième structure est "struct info":
Elle est définie dans le fichier "jouer.h". Elle prend en arguments le nom du fichier, le nombre de coups joués, le numéro du niveau et le nombre de niveaux totals. Son rôle est de transporter un maximum d'informations necessaires à travers beaucoup de fonctions tout en passant une seul argument par fonction. Le nombre total de niveaux est utile pour le test de changement de niveau, le numéro du niveau et le nombre de coups joués changent souvent et apparaissent sur la ligne des informations sur la fenêtre graphique.

- La troisième structure est l'hitorique:
Elle est définie dans "historique.h". Elle prend en arguments les coordonnées du personnage et trois entiers :

test : qui indique si c'est un déplacement normal ou une initialisation, car la gestion d'un déplacement normal et d'une initialisation d'un niveau ne sont pas traités de la même manière dans l'historique.

caisse : qui indique si une caisse a été déplacée ou non.

direction : qui indique la direction du déplacement du personnage.

Chapter 2

Les fonctions

2.1 Les fonctions concernant la lecture et l'écriture des fichiers textes

La fonction de lecture qui permet de charger un niveau lis le fichier jusqu'à atteindre le numéro du niveau demandé. Ensuite il fait correspondre chaque caractère lu au mode de case correspondante. Cette fonction renvoie ensuite le plateau.

La fonction d'écriture va dans un premier temps calculer les bornes du niveau créé pour ne pas recopier l'entièreté du plateau. Une fois les bornes calculées, la fonction fait correspondre chaque mode de case comprise entre les bornes par le carractère correspondant.

2.2 Les fonctions concernant la jouabilité

Une seule fonction est appelée dans le main: "faire_action". Elle attend une action de la part du joueur, il peut y en avoir différentes :

La première est un déplacement du personnage. La fonction, appelée "deplacer_perso", va dans un permier temps tester si le personnage peut se déplacer dans la direction désirée et si une caisse va être déplacée. Puis dans un second temps, elle va affecter les nouvelles valeurs des cases en fonction du résultat du test. La fonction, un peu longue est découpée en 4 parties correspondant aux quatres directions. Les nouvelles affectations différant pour chaque directions, je ne voyais pas comment racourcir cette fonction. La fonction qui attend l'action du joueur étant non bloquante, le personnage bougeait beaucoup trop vite pour jouer. J'ai donc ajouté une fonction sleep dans le code pour ralentir le personnage. Or le sleep dure un peu trop longtemps et le personnage bouge un peu trop lentement. En effet, en dessous d'une seconde, même pour 0,99 secondes, le personnageest trop rapide mais à une seconde il est trop lent.

La deuxième est un clic de souris et la troisième une touche de clavier pressée

en dehors des flèches directionnelles. Ces deux actions appellent deux fonctions différentes mais qui jouent le même rôle: "gestion_touche" et "gestion_clic". Ces deux fonctions renvoient une demande d'action: undo, redo, niveau précedent ou suivant, initialisation du niveau, quitter ou rien. Ensuite, cette demande d'action est analysée et effectiuée par la fonction "gestion_action_bouton".

Chapter 3

La gestion de l'historique

L'historique fonction avec une pile. A chaque déplacement de personnage, le déplacement est empilé dans la pile undo en enregistrant différentes informations, voir le chapitre 1 sur les structures pour plus de détails sur les informations enregistrées.
Si le joueur demande un retour arrière, la pile undo est dépilée de un élément puis cet élément est empilé dans la pile redo et les informations sont affectées au plateau grâce à la fonction "a_l'envers". De même si le joueur demande l'action redo, la pile redo est dépilée de un élément puis cet élément est empilé dans la pile undo et les informations sont traitées par la fonction " a_l'endroit".
Si le joueur demande une initialisation du plateau, tous les déplacments du peronnages précedant l'initialisation sont réempilés dans la pile undo dans le sens inverse. C'est-à-dire que la dernière action effectuée se retrouve être la première empilée et la première action effectuée se retrouve être la dernière empilée. Lors de cette copie des déplacements l'argument test de la strucutre de l'histoique est mis à 1. En effet, si le joueur venait à demander un retour arrière après l'initialisation, le dépilement de l'initialisation va dépiler tous les éléments dont l'argument test vaut 1. Cela est effectué par les fonctions "undo_init" et "redo_init" du fichier "historique.c". Voir le code source des fichiers "historique.c" et "historique.h" pour plus de détails sur l'implémentation de l'historique.

Chapter 4

l'édition de niveau

4.1 Gestion des actions

Une variable de type entier nommée "mode_action" garde le type de mode d'action dans lequel le joueur se trouve. Les différents modes sont:

Le mode placer : où le joueur peut placer les différents objets sur le plateau.

Le mode bouger : où le joueur peut déplacer le personnage et les caisses.

le mode bouger au hasard : qui appelle une fonction qui déplace le personnage et les caisses aléatoirement.

Les modes enregistrer et quitter : qui pour le premier enregistre le niveau dans un fichier avant de quitter le jeu.

Comme pour la partie du code qui s'occupe des déplacements dans le mode jeu du sokoban, la fonction appelé dans le main, "faire_action_editeur", attend un clic ou une pression sur une touche avant d'appeler les différentes fonctions qui vont s'occuper de la demande du joueur.
Suivant le mode d'action choisit par le joueur, différentes fonctions s'occupent des actions du joueur. Le mode d'action peut être choisit en cliquant sur la fenêtre graphique ou en utilisant les touches "P", "B", "H", "E" et "Q".

Le mode placer attend un clic sur une case du plateau pour placer un objet. La fonction "placer_objet" s'occupe de cela.

Le mode bouger, dont les fonctions, avec celles du mode bouger au hasard, se trouvent dans le fichier "editeur_action.h" attend soit un clic soit une pression sur une flèche directionnelle. Les flèches directionnelles permettent de bouger le personnage et le clic permet de sélectionner la caisse que le joueur veut tirer. En effet, une variable "caisse_select" enregistre si une caisse a été sélectionnée et où elle se trouve par rapport au personnage.
Le déplacement sans caisse et avec une caisse se fait par deux fonctions différentes.

- le déplacement sans tirer de caisse se fait par une fonction simple appelée "deplacement_normal" qui test si le personnage peut se déplacer dans la direction demandée et qui affecte les nouvelles valeurs des cases en conséquence.

- Le déplacement avec une caisse sélectionnée se fait par la fonction "déplacer_select" qui s'occupe cas par cas des quatres valeurs possibles de "caisse_select". Ensuite la fonction test si le personnage qui tire la caisse peut se déplacer dans la direction souhaitée par le joueur puis affecte les nouvelles valeurs des cases du plateau en conséquence. Une variable nommée "nb_deplacement" compte le nombre de déplacements effectués en tirant la caisse. En effet, si cette variable est plus grande que zéro, le joueur peut pousser la caisse d'autant de fois qu'il l'a tirée. Cette variable joue le role d'un petit historique si le joueur venait à se retrouver bloquer en tirant la caisse. Cette variable est remise à zéro une fois la caisse lachée.

Comme pour le déplacement dans la partie jeu du sokoban, le personnage se déplaçait trop vite sans un sleep.

Les fonctions du mode bouger au hasard se trouvent également dans le fichier "editeur_action.c". Elles s'articulent autour d'une fonction principale qui calcule aléatoirement dans quelles directions le personnage va aller ou s'il va selectionner une caisse ou la lacher. Cette fonction principale appelle des fonctions qui recherchent s'il y a une caisse à coté du personnage et qui calculent aléatoirement dans quelle direction doit aller le personnage.
Cependant ces fonctions ne mélangent pas efficacement les caisses et elle reste à améliorer.

4.2 Les fonctions de tests

Il y a deux fonctions de tests qui se trouvent dans le fichier "editeur.c":

- Le test d'un seul personnage qui parcourt le plateau et compte le nombre de fois où une case est en mode PERSO. Si le test retourne vrai, le programme va calculer les coordonnées du personnage pour l'affecter à la structure du plateau. Sinon, le mode d'action sera mis en mode placer et un message d'erreur apparaitra sur la fenêtre graphique.

- Le test du niveau fermé est basé sur une fonction récursive : une première fonction divise le plateau en quatre partie à partir des coordonnées du personnage puis appelle la fonction récursive nommée "niveau_ferme" sur les quatres parties. La fonction récursive teste case par case si c'est un mur ou en bordure du plateau. La variable "cmp" est l'addtion des résultats des tests. Dès que cette variable est différente de zéro, la fonction s'arrête à la fin du dépilement. En effet, cela permet de ne pas planter le programme

à cause d'un stack overflow. Cependant, cette fonction n'est pas complète et dans certains cas, la fonction dira que le niveau est fermé alors que non.

Part II

Les améliorations

Chapter 5

Valgrind

Utiliser valgrind sur le programme montre qu'il y a eu des fuites de mémoire lors de la fin du programme. Or lorsqu'on utilise l'option "–leak-check=full", on observe que les fuites viennent de la SDL. Une amélioration serait donc d'aller résoudre ces problèmes de fuites même si elles ne viennent pas du programme du jeu.

Chapter 6

Les fonctions

Pour améliorer le programme, il faudrait racourcir les deux grosses fonctions de déplacement qui se trouvent dans les fichiers "jeu.c" et "action_editeur.c".
De puls, certaines fonctions, surtout celles qui permettent de déplacer des personnages (dans l'historique, l'editeur et jeu.c) sont presque les mêmes. Il serait interessant de voir si certaines ne pourraient pas être fusinnées et ainsi le code s'en retrouverait moins long. Cependant, il faudrait remanier le code complètement et elle réécrire en incluant ces paramètres.
Enfin, les fonctions sont remplies de "if" et il est possible de les limiter. Cela serait une bonne amélioration du code.