Rapport du projet Taquin

SAE Dev 1.1

Mathis CHAIGNEAU
Florian CORBIER

Table des matières

Introduction

L'objectif de cette première SAE de Dev 1.1 était la création d'un Taquin. Ce jeu est fait, en général, d'une image découpée en petits carrés qui s'inscrivent dans un cadre et dont le carré supérieur gauche est manquant. Ces carrés sont ensuite désordonnés et le but du jeu est de les remettre dans l'ordre grâce à une succession de déplacements des carrés dans l'emplacement vide.

Nous avions à produire ce petit jeu en langage C89 et avec, pour seul emprunt extérieur, la bibliothèque graphique de l'IUT. Nous avions également des contraintes comme le fait que l'utilisateur puisse modifier le nombre de carrés par ligne et le nombre de carrés par colonne et cela indépendamment l'un de l'autre, lui laisser le choix entre 3 images de dimensions différentes et pouvoir rejouer à la fin d'une partie, pour ne citer que celles-ci.

Fonctionnalités

Menu d'avant-partie

En lançant l'exécutable vous arriverez dans le menu d'avant-partie ci-dessous.

Note : Les interactions avec le menu d'avant-partie ne se feront qu'au clic gauche.

Vous y trouverez :

- les trois images disponibles, à vous de cliquer sur l'une d'entre elles pour la sélectionner;
- le choix du nombre de colonnes, ou nombre de carrés par ligne. Vous pourrez faire descendre le compteur jusqu'à 3 via le bouton moins. Vous pourrez faire monter le compteur jusqu'à 8 via le bouton plus;
- le choix du nombre de lignes, ou nombre de carrés par colonne. Le fonctionnement est les même que précédemment;
- la possibilité de générer une grille pouvant être résolue en un coup, il suffira de cliquer sur le bouton "CHEAT";
- la possibilité de lancer une partie via le bouton "PLAY THE GAME".
  Note : Il n'est possible de lancer une partie que si une image a été sélectionnée.

En jeu

Une fois la partie lancée, vous arrivez dans l'interface de jeu qui se présente comme suit.

Vous y trouverez :

- en haut à gauche, des indications sur comment jouer;
- en haut à droite, le compteur du nombre de coups joués;
- à gauche, l'image d'origine;
- à droite, l'image découpée en tuiles. C'est l'espace de jeu;
- en bas, un bouton pour quitter le jeu à tout moment.

Vous pourrez résoudre la grille via le clic gauche ou les flèches directionnelles et la barre d'espace ou les deux, vous pouvez passer de l'un à l'autre très facilement.

Vous jouerez de la manière suivante, vous aurez à sélectionner une première tuile, via le clic gauche ou en parcourant la grille via les flèches directionnelles puis en enfonçant la barre d'espace.
Vous aurez ensuite à sélectionner une deuxième case, de préférence la case vide si vous voulez avancer, par le même procédé. Si la deuxième case sélectionnée est la case vide et qu'elle est adjacente à la tuile sélectionnée, les deux échangeront leur place, si elles ne sont pas adjacentes alors rien ne se passera. Si la deuxième case est une tuile alors la sélection passera à cette dernière.

Menu de fin de jeu

Si vous arrivez là alors c'est que vous êtes fort, ou que vous avez triché. Quoi qu'il en soit, après avoir résolu le Taquin vous arrivez dans le menu qui suit.

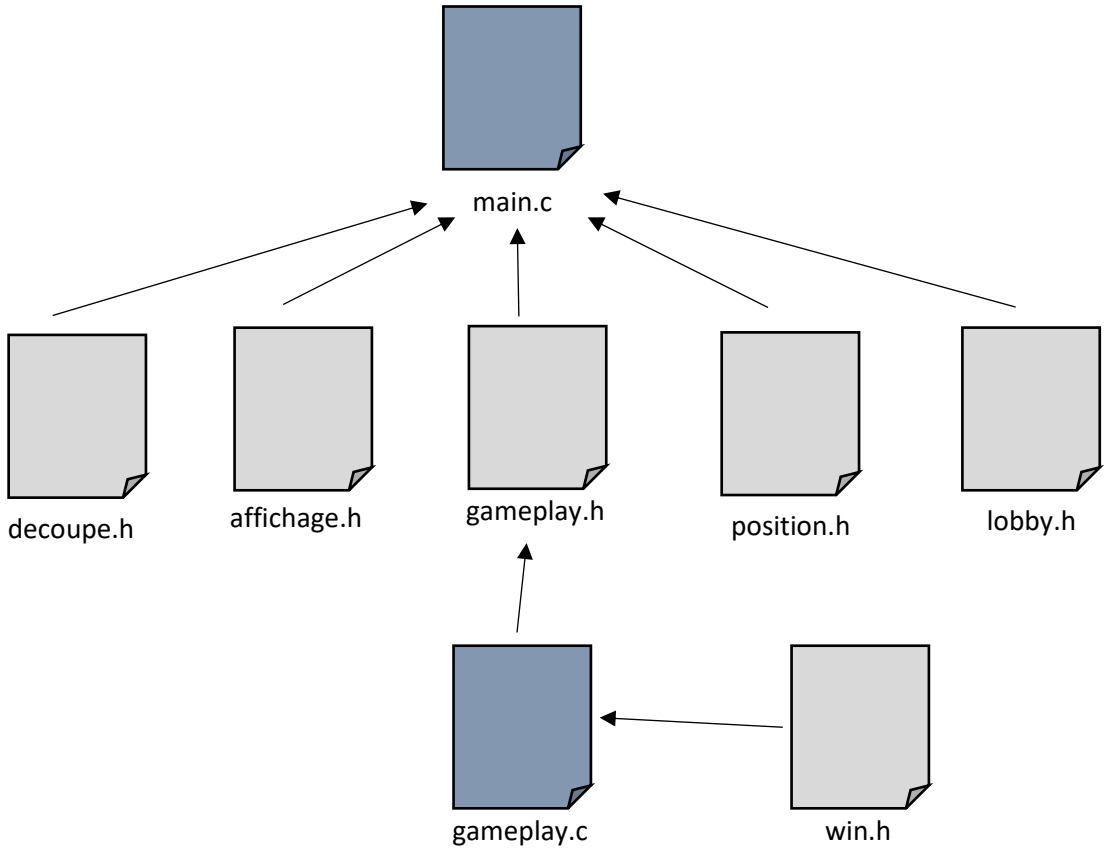
Vous avez alors deux options :

- relancer une partie en cliquant sur le bouton "Rejouer", cela vous ramènera au menu d'avant-partie;
- quitter le jeu en cliquant sur le bouton "Quitter".

Présentation du code

Découpage

Le code est découpé en 8 fichiers, sans compter le Makefile et en comptant pour un fichier les fichiers d'extension .c et .h d'une même fonction. Voici un diagramme pour visualiser le découpage, le fichier variables.h n'y sera pas représenté et nous considérerons que tous les fichiers sauf win.c dépendent de ce dernier. De plus, quand cela n'est pas nécessaire, les fichiers d'extension .c ne seront pas affiché et seront considéré comme seuls fichiers inclus dans les fichiers d'extension .h.



Le fichier main.c contient donc tout le code qui est exécuté. Il n'y a pas beaucoup de code dans ce dernier, il est surtout fait d'appels à des fonctions.

Le fichier lobby.h contient le prototype de la fonction lobby. Le code de la fonction lobby se trouve dans le fichier lobby.c. Cette fonction gère le menu d'avant partie et tout ce que l'utilisateur a à choisir. On peut trouver dans ce même fichier la fonction Encadre2 qui sert à dessiner un rectangle avec une épaisseur et une couleur variable.

Le fichier decoupe.h contient le prototype de la fonction decoupage. Le code de cette fonction se trouve dans le fichier decoupe.c. Cette fonction gère le découpage de l'image choisie par le joueur, en un nombre de ligne et de colonnes également choisies par le joueur. Elle détermine également les emplacements possibles pour les tuiles.

Le fichier position.h contient le prototype de la fonction repartition. Le code de cette fonction se trouve dans le fichier position.c. Cette fonction gère la disposition aléatoire des tuiles.

Le fichier affichage.h contient le prototype de la fonction affichage. Le code de cette fonction se trouve dans le fichier affichage.c. Cette fonction est chargée d'afficher la grille de jeu.

Le fichier gameplay.h contient le prototype de la fonction gameplay. Le code de cette fonction se trouve dans le fichier gemaplay.c. Cette fonction gère la partie et la fin de partie.
Elle fait appel aux fonctions, définies dans ce même fichier, suivantes :

- DetectMarge, qui sert à savoir si un clic a eu lieu entre deux tuiles;
- DetectCase, qui sert à savoir sur quelle case de la grille un clic a eu lieu;
- Encadre, qui sert à dessiner un rectangle;
- PreSelect, qui gère la présélection des cases au clavier;
- SelectDeuxiemeCase, qui gère la sélection d'une deuxième case.

La fonction utilise également la fonction win, qui gère le menu de fin de partie. Le prototype de cette fonction se trouve dans le fichier win.h et son code, dans le fichier win.c.

Le fichier variables.h contient des variables globales telles que les chemins vers les images utilisées, les dimensions de certaines d'entre elles ou encore l'espacement entre deux cases.

Le fichier Makefile contient des ordres de compilation pour la commande make. Ainsi la compilation est plus rapide.

Structure du code

Le programme est dans le fichier main.c et il appelle les différentes fonctions citées ci-dessus ce qui permet d'avoir un programme concis et plus compréhensible. Voici donc sa composition.

Premièrement, nous déclarons les variables principales du programme, peu sont initialisées car elles doivent s'adapter aux choix de l'utilisateur dans le menu d'avant-partie.

Nous ouvrons ensuite une fenêtre graphique et mettons un fond uni dont la couleur a été définie un peu plus haut. La boucle qui suit sera exécutée tant que le joueur n'aura pas décidé de quitter le jeu et une exécution entière de cette boucle correspond à une partie jouée.

Dans cette boucle nous trouvons premièrement un appel à la fonction lobby pour placer le joueur dans le menu d'avant partie. Nous travaillerons ensuite sur différentes variables en fonction de ses choix. Nous utilisons le passage par adresse de certaines variables importantes pour récupérer ses choix.
Pour continuer, nous appelons la fonction decoupage puis la fonction ordre.

Nous voilà prêt à construire l'interface de jeu. Nous commençons par effacer la fenêtre graphique avec une couleur de fond unie puis nous affichons les premiers éléments de cette interface. Nous appelons la fonction affichage pour terminer l'interface.

Le jeu peut à présent commencer. Pour cela nous appelons la fonction gameplay dans un if pour tester son code de retour, c'est-à-dire l'entier que revoie la fonction à la fin d'une partie et selon le choix du joueurs (rejouer ou quitter). Ainsi si le joueur décide de quitter le jeu nous sortons de la boucle while, nous fermons la fenêtre graphique et libérons les variables dont la mémoire a été allouée dynamiquement. Dans le cas contraire, nous effaçons le contenu de la fenêtre par une couleur unie et recommençons une nouvelle exécution de la boucle.

Pour parler rapidement de la fonction gameplay, qui est le cœur du jeu, la majorité de cette fonction est faite d'une boucle qui s'exécute tant que l'ordre des tuiles n'est pas bon. Cette boucle contient quatre principaux blocs if, un qui gère la sélection d'une première case avec le clavier, un deuxième qui gère la sélection d'une deuxième case au clavier et les deux autres qui ont la même fonction mais pour le clic gauche cette fois ci. Le teste qui détermine si les tuiles sont dans le bon ordre est effectué à chaque fois qu'il y a un mouvement.
Une fois que nous sommes sortis de la boucle, c'est-à-dire que le joueur a résolu le Taquin, nous retournons son choix dans le menu de fin de partie grâce au code de retour de la fonction win.

Présentation des variables

Outre certaines moins importantes, comme les variables de couleur ou encore des variables de placements, les variables les plus importantes sont déclarées dans le main et affectées par une valeur plus tard, pour convenir aux choix du joueur. Voici donc les variables les plus importantes.

Les variables rec_x et rec_y, qui sont de type int, contiennent respectivement le nombre de colonnes et le nombre de lignes, souhaités par l'utilisateur. Elles sont affectées par valeur une dans la fonction main.

A la fin de l'exécution de la fonction lobby, nous pourrons allouer dynamiquement de la mémoire aux tableaux suivants en fonctions de la valeur de rec_x et rec_y :

- ordre est un tableau de type int et dont le but est de répertorier l'ordre des tuiles;
- decoupe est un tableau à deux dimensions, de type int et dont le but est de connaître la place de chaque tuile dans l'image de base;
- position est un tableau à deux dimensions, de type int et dont le but est de connaître les emplacements possibles pour les tuiles.

La taille de ces trois tableaux est le nombre de tuiles, c'est-à-dire rec_x*rec_y, dans le cas des tableaux à deux dimensions, il s'agit de la taille de la première dimension. La taille de la deuxième dimension, des deux tableaux à deux dimensions, est 3. Ainsi, pour les tableaux à deux dimensions, chaque valeur de la première dimension correspond à une tuile ou une case, selon le tableau, et

chaque valeur de la deuxième dimension correspond à l'indice, le point de coordonné x et le point de coordonné y des tuiles ou cases selon le tableau.

Voici un petit exemple, la deuxième dimension des deux tableaux à deux dimensions sera résumée par l'indice de la tuile ou de la case. Cet exemple peut tout à fait être imaginer à un stade avancé de la partie.

Soit rec_x = 3 et rec_y = 3, nous imaginons le tableau ordre et obtenons les tableaux position et découpe suivants.

ordre	5	3	0	1	7	4	6	8	2
position	0	1	2	3	4	5	6	7	8
decoupe	0	1	2	3	4	5	6	7	8

Nous obtenons ainsi les grilles suivantes, à gauche la grille contenant l'indice de chaque case et à droite la grille contenant l'indice de chaque tuile.

0	3	6
1	4	7
2	5	8

5	1	6
3	7	8
0	4	2

La variable id_image, qui est de type int, contient l'id de l'image souhaitée par l'utilisateur. Elle est affectée par une valeur dans la fonction lobby. Ainsi, et selon sa valeur, les variables l et h, qui sont de type int et respectivement la largeur et la hauteur de l'image sélectionnée, et la variable path_image, qui est une chaîne de caractères correspondant au chemin vers l'image choisie, sont affectées par les valeurs adéquates à la fin de l'exécution de la fonction lobby.

Le mélange

Pour garantir un système de mélange aléatoire qui produise une grille qui puisse être résolue, un code qui fonctionne comme expliqué ci-dessous va s'exécuter un nombre de fois défini par la variable globale BRASSAGE.

Avant tout, le tableau ordre qui représente l'ordre des tuiles dans la grille est rempli dans le bon ordre. Ensuite nous entrons dans la boucle. Nous réinitialisons le tableau chargé de répertorier les mouvements possibles et modifions les 0 par des 1 lorsque le mouvement est possible à partir de la case vide, les mouvements étant droite, gauche, haut et bas.

Parmi les mouvements possibles il est choisi un mouvement, de manière aléatoire, puis ce mouvement est appliqué, c'est-à-dire qu'il y a une permutation de deux valeurs dans le tableau ordre (le 0 et une autre).

Ainsi, nous avons comme résolu le taquin à l'envers, par une suite de mouvements possibles (la case vide avec une tuile adjacentes). Cela permet de garantir que la grille produite puisse être résolue.

Conclusions

Mathis CHAIGNEAU

J'ai trouvé ce projet tout à fait intéressant car il m'a fait redoubler d'efforts pour trouver des idées de comment construire le programme, que ce soit au niveau code pur ou organisation et découpage du code. De plus cela a permis d'approfondir cette notion importante qu'est le découpage du code en différents fichiers et fonctions.
L'alliance, pour la première fois, de la partie algorithmique et de la partie graphique était également plaisante à explorer.

Florian CORBIER

Le projet était intéressant, cependant, l'impossibilité de faire du « sur-mesure » et dommage. L'ajout d'une fonction qui permettrait de récupérer les dimensions des images aurais été un grand « plus ».
L'obligation de split le code (et donc aussi de créer plusieurs headers) est très intéressante.