Projet Programmation Avancée

Paul CHASTEL
Corentin GUILLOT

Janvier 2021

Sommaire

1 Introduction

Pour ce projet de programmation avancée nous devions réaliser un jeu de taquin, pour ce faire nous avons réfléchis a l'ordre des fonctionnalités que nous voulions implémenter, on a donc d'abord commencé par un code de base qui prends des entrées terminal et qui renvoie les réponses dans le terminal le tout sans fonctions annexes pour améliorer la qualité du jeu, nous avons ensuite ajouté les fonctionnalités avancées du programme tel que l'aide au joueur et la génération fichier enfin nous avons fini par un affichage SDL du jeu avec des entrées dans l'affichage SDL et des sorties visuelles et sonores elles aussi dans le SDL.

2 Code de base

Nous avons d'abord pensé a un code qui fonctionne avec le terminal Linux, où l'on vient rentrer la taille de notre tableau de Taquin et le générer aléatoirement dans un premier temps pour ensuite nous déplacer en rentrant les lettre (H, B, D, G) pour toutes les direction possibles, en empêchant bien attendu les déplacements impossibles (vers les bords du tableau notamment).

Cette application demande déjà plusieurs fonctions, en effet il fallait déjà une fonction qui génère un tableau aléatoire résoluble de la taille N demandée par l'utilisateur et dotée d'une case vide. Dans notre programme nous avons

décidé que la valeur $N \times N$ serait la valeur de la case vide, nous créons donc un malloc de taille $N \times N$ remplit de valeurs croissantes jusqu'à $N \times N - 1$ et la dernière case sera toujours la case vide: fonction *initialisation* dans le programme *fonction importante.c*.

Ensuite nous avons créé un programme qui vient mélanger un tableau existant avec une suite de coups aléatoires cela nous permet d'avoir un tableau résoluble, en effet on ne passe que par des coups possible, il a donc fallut pour cette fonction créer les mouvements possibles, c'est l'intérêt du programme *mouvement*, a partir de ce programme nous avons pu faire la fonction *mélange* dans le programme *fonction importante.c*.

Puis il faut que l'utilisateur réussisse le jeu, comme expliqué plus haut il faut pour cela qu'il rentre la direction voulue dans le terminal (H, B, D, G), on s'est servi d'une boucle *While* qui vient comparer après chaque coup le tableau de jeu actuel avec un tableau résolu de même taille la boucle s'arrête en cas d'égalité entre les deux tableaux et donc de réussite du programme.

Il a fallu ,pour ce faire, réaliser une fonction qui vient donner le tableau résolu de même taille N, c'est ce que l'on a fait dans la fonction *résolution* du programme *fonction importante.c*où l'on créé tout simplement un malloc de taille $N \times N$ remplit de valeurs tirées dans l'ordre croissant.

De plus, on a réalisé la fonction *jeu* du programme *jeu.c* qui vient appliquer les changements voulu par l'utilisateur après un coup il vient échanger les cases, comparer avec le tableau résolu en cas de réussite du jeu et affiche un message d'erreur dans le cas d'un coup impossible tel que sortir du tableau.

Une partie de ces programmes sont en commentaire ou on été supprimés avec l'utilisation du SDL qui les rendait inutiles.

Nous avons aussi eu recours a un Makefile pour facilité la compilation de toutes les fonctions et programmes énumérés auparavant, et pour toutes celles a venir.

3 Fonctionnalités ajoutées

Il nous a été demandé de créer une fonction qui permet de récupérer un tableau depuis un fichier texte. Comme il ne nous a pas été fourni de formatage particulier pour le documents texte les conditions sont que: les nombres du tableau ne doivent pas dépasser 99, de plus les nombres seront divisés par des espaces, et enfin la case vide sera matérialisée dans le document par un espace, le tout devra être écrit dans le document texte appelé *tableau.txt* par exemple le tableau si dessus est valide:
1 2 3
4 5
6 7 8
Cette fonctionnalité a été ajouté dans la fonction *lecture* du programme *fonction importante.c* on vient lire le document texte ligne par ligne a l'aide de *fgets* et on "transfert" le nombre dans un tableau initialisé a la taille N, la case vide étant toujours matérialisée par la valeur $N \times N$ dans le tableau final par soucis

d'uniformité.

Ensuite il nous a été demandé de réaliser un programme visant à aider le joueur dans le résolution du jeu pour ce cas ci nous avons décidé de faire une aide au joueur coup par coup. Le programme va trouver le coup qui le rapproche le plus du résultat attendu sans pour autant réfléchir a une solution globale, il y a un cas ou notre aide ne marche pas c'est quand les quatre coups sont égaux pour l'algorithme il n'y aura pas d'aide affichée car les quatre coups se valent pour le programme. Il s'agit de la fonction *aide* du programme *jeu.c*.

4 Le SDL

Nous avons réalisé un affichage SDL pour notre jeu de Taquin nous avons d'abord réalisé un menu pour ce jeu qui nous demande de choisir entre la génération aléatoire d'un tableau résoluble et la génération par document texte comme vu dans les fonctionnalités ajoutées, c'est l'intérêt de la fonction *menu* du programme *affichage.c*.

Par conséquent, nous avons aussi rajouté l'affichage SDL du jeu en lui même après avoir sélectionné une méthode de génération d'un tableau, on vient l'afficher sur la fenêtre de jeu: la case noire représente la case $N \times N$ donc la case vide, la case en vert représente le coup conseillé par la machine, il suffit d'appuyer sur les flèches directionnelles du clavier ou les touches (H, B, D,G) pour ce déplacer dans la direction voulue sur l'écran, c'est le but de la fonction *affichage* dans le programme *affichage.c*.

Nous avons de plus réalisé un écran de réussite avec un ajout sonore en cas de réussite en effet a chaque coup le programme compare le tableau actuel avec le tableau résolu de taille N toujours dans la fonction *affichage* du programme *affichage.c*.

5 Conclusion

Ce projet nous aura permit de mettre en pratique nos connaissance acquisses durant ce semestre en programmation avancée notamment les nouvelles notions telles que le SDL et le Makefile. Ce projet pourrait cependant être amélioré de différentes façons: avec un algorithme d'aide plus performant, avec une saisie fichier qui dépasse les nombres a 2 chiffres et avec un affichage/son plus poussé pour la partie SDL.