LYCEE
Claude FAURIEL

PROJET C

Tetros : Compte-rendu

Adrien FRACHET – Lucas BALMÈS – Théo COURT

Table des matières

I. Introduction……2
II. Présentation et commandes……..3
III. Mise en place et développement……4
1) Étapes par étapes……4
2) Difficultés rencontrées……5
3) Améliorations possibles……..5
IV.Conclusion……..6

I. Introduction

Le Tetris est un des jeux les plus populaires du monde encore aujourd'hui. Avec plus de 170 millions d'exemplaires du jeu écoulés, il a marqué l'enfance de beaucoup de gens encore aujourd'hui. Lors des différentes propositions de jeux s'offrant à nous, nous avons immédiatement choisi ce dernier pour notre projet du 1er semestre. Nous avons réussi, non sans quelques difficultés, à coder le jeu complet.

Pour lancer le code, il suffit de faire dans la commande *make* puis *./tetros*. Le jeu est jouable en fenêtré mais il est conseillé d'y jouer en plein écran.

Nous avons donc fait en premier lieu un brouillon de départ présentant les différentes fonctions nécessaires au programme ainsi que leurs prototypes, puis nous avons définis les variables nécessaires au bon déroulé de notre programme. Ensuite, nous avons codés les différents fichiers en C, avec l'ajout de fichiers annexes tels que debug, puis nous avons rajouté différents suppléments à nos fonctions, comme les couleurs, le menu ou encore l'animation lors du lancement du jeu.

Tout ceci nous a permis de rendre l'interface de notre jeu à la fois facile à utiliser et complète.

Notre objectif, tout au long de ce projet, était avant tout de coder le jeu Tetris (sous le nom de tetros), mais également de rendre notre code à la fois documenté et clair (sans pour autant entrer dans un excès de commentaires) via des annexes telles que les documentations schématisées ou encore le glossaire. La documentation est également contenue dans un autre fichier .odt accompagnant le compte-rendu. Le nombre de pré et post-conditions est réduit au maximum car l'héritage entre les fonctions est contrôlé et ne nécessite donc pas de tests supplémentaires pouvant alourdir le code. Même si cette solution implique un certain nombre d'inconvénients, comme des désavantages (une fonction retirée du programme pourra ne pas fonctionner), nous avons fait ce choix ensemble, avec pour objectif de simplifier les programmes.

II. Présentation et commandes :

La grille du Tetros principale est modélisée par un tableau statique de taille 22x10, elle est nommée « grid » dans l'ensemble des fichiers. La pièce actuelle, elle, est modélisée dans un tableau secondaire de taille 22x10 également, nommé « movinggrid ». La convention de prototypage et codage C respectée est la suivante : http://users.ece.cmu.edu/~eno/coding/CCodingStandard.html. Chaque déplacement associe des comparaisons entre les deux grilles.

Les commandes du jeu sont présentes directement dans l'interface du jeu. Voici les détails des commandes particulières :

- HAUT : Fait descendre la pièce actuelle directement en bas
- D : Permet la rotation de la pièce actuelle vers la droite. Dans le jeu original, il n'y a qu'une seule rotation de prévue.
- S : Permet de stocker le bloc actuel et de passer au suivant. (voir « Réserve » dans la documentation du code)

Comme dans le jeu original, un système de niveaux est respecté, chacun associant un nombre de points différents pour chaque ligne finie. Voici le tableau comparatif des points :

Lignes/Niveaux	1	2	3	4
1	40	80	120	160
2	56	112	168	224
3	96	192	288	384
4	136	272	408	544

Le Tetris basique contient un système d'accélération de la vitesse de chute des blocs que nous avons souhaités mettre. Pour cela, nous avons créé deux modes de jeux distincts permettant au joueur de choisir de quelle manière il souhaite que l'accélération se produise : Par points ou par temps. Il est important de noter que la vitesse de descente des blocs ne peut pas descendre au-dessous de 50, en effet, cela rendrait le jeu injouable. La loi reliant l'accélération aux points et au temps est la suivante :

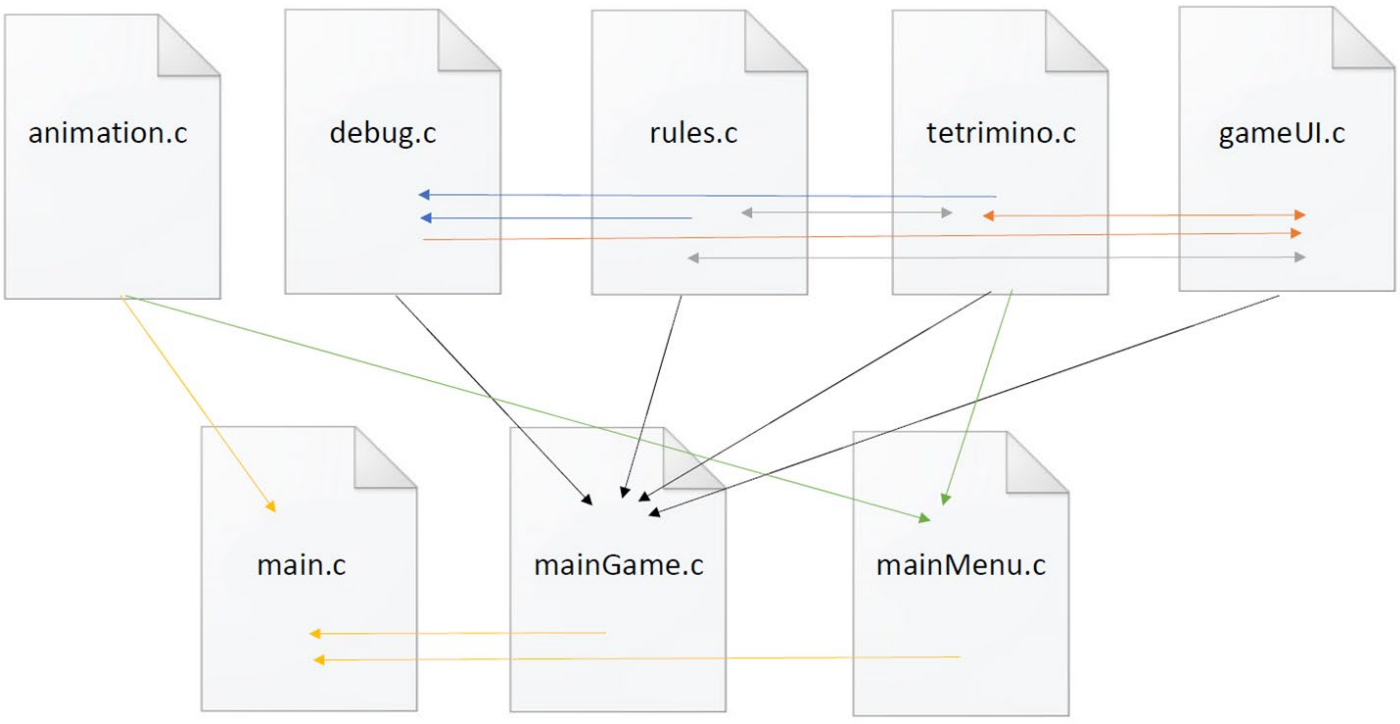
Mode de jeu	Points	Temps
Loi mathématique	Tsuiv=tpréc · $\frac{sSuiv - sprécédent}{10}$	Toutes les 10s, les blocs descendent de 20ms plus vite

tSuiv est le temps après l'accélération, et tPréc avant. De plus, sSuiv est le score actuel et sprécédent le score obtenu par le joueur au moment de l'accélération précédente.

III. Mise en place et développement :

1) Étape par étape

Nous avons, en premier lieu, placé sur un brouillon l'ensemble des fichiers nécessaires au bon déroulé de notre programme, ainsi que l'héritage les reliant (d'après la méthode que nous avions déjà suivie en cours pour le morpion, voir figure 1). Une fois ce dernier terminé, nous avons écrits les prototypages de toutes les fonctions nécessaires dans chacun des fichiers avant de nous occuper de l'héritage des fonctions ; enfin, nous nous sommes concentrés sur l'interface de jeu ainsi que sur des ajouts esthétiques tels que l'animation ASCII art au lancement du jeu ou encore l'indicateur [] de chute permettant de prévisualiser la position finale du tetrimino.



2) Difficultés rencontrées

Notre première version du jeu comportait, au lieu d'une grille secondaire (movinggrid), un tableau statique représentant le tetromino comme 4 carrés collés entre eux, et qui associait à chaque carré ses coordonnées. Cette solution s'étant avérée beaucoup trop complexe et coûteuse en mémoire à mettre en œuvre (La fonction goDown prenait à elle seule une quarantaine de lignes de comparaisons et de boucles « for »), nous avons eu l'idée de créer une seconde grille à part contenant uniquement le bloc, rendant alors la comparaison entre les deux grilles ainsi que le placement des blocs beaucoup plus facile.

La seconde grande difficulté que nous avons rencontrée se situait au niveau de la fonction « rotation ». En effet, il faut vérifier si la rotation est possible, garder les blocs assemblés entre eux, faire une rotation différente en fonction du bloc choisi, etc… Tant de conditions ne simplifiant pas la tâche ! Pour réussir cela, nous avons au final dû utiliser notre cours récent sur les matrices en réduisant mobileGrid au maximum possible avant d'utiliser sa transposée et d'autres propriétés sur ces dernières. Après avoir accompli celle-ci, nous nous sommes confrontés à un troisième problème.

En effet, notre version du jeu, complète à ce moment-là, constituait un ASCII ART représentant « Tetros » en diagonale et un choix de difficulté sans menu. Cette interface étant peu esthétique, nous avons alors rajouté dans le fichier mainMenu.c des fonctions permettant de créer un menu animé avec une sélection via les flèches du clavier.

3) Améliorations possibles

Si il est possible de choisir si l'accélération de la descente des blocs se fait en fonction du temps ou des points,il n'y a actuellement qu'un seul mode de jeu disponible concernant le gameplay du joueur , il s'agit du Tetris de base. Dans certains dérivés, d'autres modes de jeux ont été créés et auraient pu être disponibles également dans notre version.

Nous aurions également pu colorer l'animation de lancement mais nous avons décidé de nous en arrêter là pour cette dernière. Le bloc suivant n'est pas non plus affiché, alors qu'il l'est dans les autres Tetris (sauf le premier Tetris, celui que nous avons souhaité copier), et que cet ajout aurait pu moderniser notre jeu.

IV. Conclusion :

Ce projet nous a donc permis d'apprendre la cohésion d'équipe entre différents camarades au sein d'un groupe. Nous nous sommes tenus informés de l'ensemble des modifications entre nous par le biais de discussions vocales et écrites (notamment via discord) pour au final arriver à dépasser nos propres attentes en réussissant à coder entièrement Tetris, en rajoutant un système de points, deux modes d'accélération du jeu, un indicateur de l'endroit où va tomber le bloc, différentes couleurs pour les blocs, rendant le jeu fidèle à l'original, optimisé, documenté et esthétique autant que possible. Nous sommes fiers de la réussite de ce projet qui, nous avait confronté à un certain nombre de difficultés lors de ses débuts, mais qui, grâce à notre obstination et notre amour pour le codage, s'est concrétisé.