PAN Julien
MISTRY Jay
BERHOUMA Dhia
MAGASSA Souleymane

Réalisation de programme L2

- Présentation du projet :

Le projet consistait à programmer un tetris qui diffère du tetris classique. Le tetris contenait douze pièces différentes avec des règles modifiées, tels que le placements des pièces où nous le voulons tant qu'elle ne dépasse pas, et que ce soit des cases vides à remplir du tableau (25x20). Ensuite, quand une ligne ou colonne est totalement remplie, elle est vidée, puis selon la ligne ou colonne, les pièces se décale d'un rang de haut en bas, ou de gauche à droite, pour que le tetris est de la place pour continuer à y jouer.

- Présentation du code :

Fichier d'entête (tetris.h) :

Nous avons commencé à déclarer les différentes variables globales :

```
#define LIGNE 25
#define COLONNE 20
#define MAX 100
```

Puis, à introduire une structure **grille** qui contient la valeur de la case de la **grille**, déclarer en entier **val**. Suivi d'un **typedef** afin d'éviter une écriture plus longue.

```
struct grille {
        int val;
};
typedef struct grille create[LIGNE][COLONNE];
```

Cette structure va permettre d'être appelé en paramètre de fonction pour que le tableau soit le même durant l'exécution du programme. Nous avions essayé tout simplement de déclarer un tableau d'entier en variable globale, aussi en **static int**, mais elle ne nous a pas permis d'afficher le tableau avec les pièces jouées. Elle affichait à la place un tableau vide de vue mais remplie dans le programme car quand nous avions essayé de placer des pièces au même endroit, il y avait une erreur de placement car les cases n'étaient pas vide.

Fichier code (tetris.c) :

Pour que l'utilisateur connaisse les pièces du jeu Tetris, nous avons commencé par une fonction **affpiece()** permettant d'afficher toutes les pièces possibles avec un numéro associé en précisant le point principale de la pièce pour que le l'utilisateur sache où et comment placer la pièce en utilisant des printf, avec les caractères en gras pour que nous puissions mieux voir le tableau.
Exemple : **printf("\033[1m...**

Les cases de la grille du jeu sont initialisées à 0 avec la fonction **initialise** prenant en paramètre la structure du tableau. L'aléatoire du jeu est fait en créant un autre tableau **valp[MAX]** avec une fonction **shuffle()** qui va pouvoir mélanger les entiers contenus dans **valp**.
Une fonction est créé afin que l'interaction entre le terminal et l'utilisateur soit possible. Cette fonction **entrer** permet de rentrer les numéros des lignes et colonnes où le joueur désire jouer, mais aussi de vérifier, que les valeurs entrées n'excède pas la taille du tableau et d'afficher une erreur. Les fonctions **allp** et **verifier** vont permettre au placement des pièces du jeu. La fonction **verifier** va analysé s'il est possible d'ajouter la pièce actuelle dans la grille en appelant **allp** qui remplit les cases, tout en vérifiant si une ligne est remplie avec la fonction **points**. Nous n'avons pas réussi à coder dans la fonction **points** pour la colonne, mais aussi le décalage des pièces car elle ne nous donnait un résultat qui ne respectait pas correctement les règles. Soit, quand les valeurs des cases se décalent de haut en

bas, le ligne du dessus de la ligne vidée était rempli par la valeurs de la première ligne, soit une partie était vidée, alors qu'elle ne devrait pas.

Fichier main (main.c) :

Le déroulement du programme démarre par remplir le tableau **valp** et de le mélanger, d'initialiser la grille du Tetris, puis d'afficher les pièces du jeu pour le joueur. Le jeu se lance donc avec 100 coups possibles, arrivé au 100ème coup, le programme s'arrête.

- Factorisation du programme :

Au départ, la fonction **verifier** était beaucoup trop longue et répétitive, elle est encore, mais moins qu'avant. Nous l'avons raccourci en créant la fonction **allp** qui va permettre de réunir toutes les pièces, et y modifier les pièces directement dans cette fonction. La fonction **verifier** contenait aussi l'interaction avec l'utilisateur et aussi les limites de la grille et les messages d'erreurs. Ce qui compliquerait la fonction, et devenait illisible rapidement. Ces fonctionnalités sont donc réunies dans la fonction **entrer**, où elle affiche un message d'erreur selon le problème, puis demande à l'utilisateur de rentrer d'autres valeurs pour la ligne et la colonne.