Projet DEV4 – C++

Rapport de la version console de Tetris en C++

Auteurs : Derboven Téo *60073* & Angielczyk Marcel *60453*

Introduction

Le présent rapport est destiné à fournir des explications sur notre jeu Tetris en vue console développé en langage C++.

Changements par rapport à l'analyse

Dans la version théorique de notre classe Line, un tableau dynamique de pointeurs de Line devait former notre Grid, désormais, il s'agit d'une **deque**, une liste doublement chaînée, qui contient toujours des pointeurs de Line. Il est notamment plus simple de remplacer une ligne vidée en tête de liste.

Ensuite, le modèle précédent proposait un constructeur de Game avec des arguments optionnels à rallonge. Ces arguments servent à paramétrer le jeu (Mode de jeu, Blocs placés dans la grille et niveau de base supérieur). Ils sont désormais tous contenus dans une structure GameParameters qui doit être passée à Game pour que celui-ci se construise.

Les problèmes/solutions rencontrés

Le premier problème que nous avons rencontré, était dans la structure de nos tableaux, notamment dans la classe Tetromino ainsi que Line. Dans ces deux classes nous devions représenter des tableaux de Block, cependant et contrairement au langage Java, il était impossible de déclarer un objet avec une valeur nulle. Nous avons donc pris l'initiative d'utiliser l'outil **std::optional** de la librairie standard que nous avions rencontré à l'examen de DEV3.

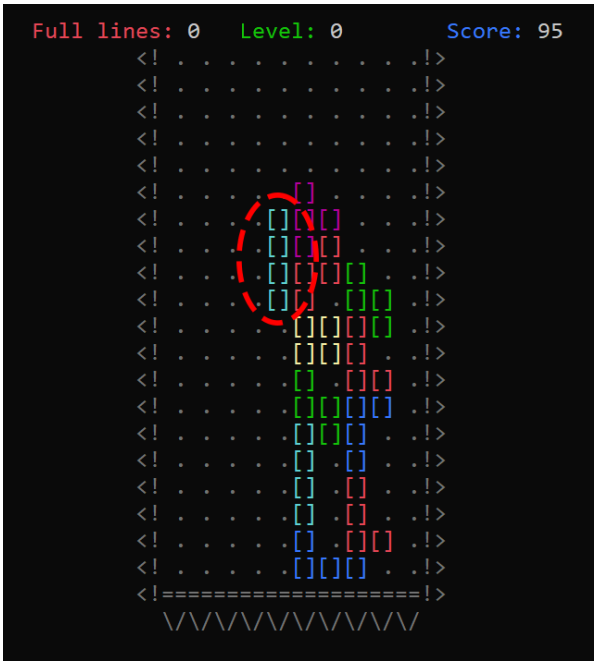
Nous savons qu'il était tout de même possible d'utiliser des pointeurs nuls pour palier à ce problème, cependant nous avons déduit qu'il était bien plus simple d'utiliser la méthode choisie.

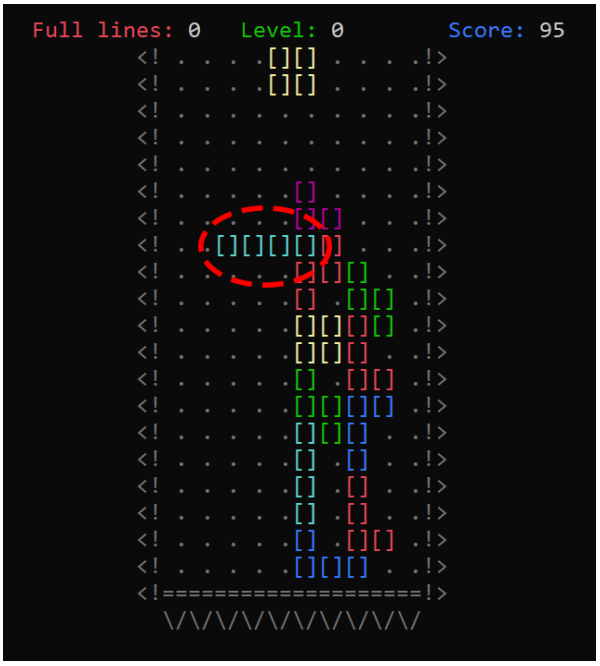
Mais nous avons encore rencontré un autre problème. Nous tenions absolument à afficher les couleurs dans la vue console. Celles-ci sont censées fonctionner sans problème sous Linux, cependant sous Windows cela n'a pas été si simple. Nous avons cherché sur internet des solutions pour cela, mais rien ne fonctionnait comme on le voulait. Nous avons donc décidé de faire appel à l'intelligence artificielle pour réaliser ce bout de code.

```c
//MADE COLORS WORKS ON WINDOWS
#ifdef _WIN32
#include <windows.h>

#endif

void enableVirtualTerminalProcessing() {
#ifdef _WIN32
    // Get the handle to the standard output device.
    HANDLE hOut = GetStdHandle( nStdHandle: STD_OUTPUT_HANDLE);

    if (hOut == INVALID_HANDLE_VALUE) {
        exit( Code: GetLastError());
    }

    DWORD dwMode = 0;

    // Get the current output mode.
    if (!GetConsoleMode( hConsoleHandle: hOut, lpMode: &dwMode)) {
        exit( Code: GetLastError());
    }

    // Enable the virtual terminal processing mode.
    dwMode |= ENABLE_VIRTUAL_TERMINAL_PROCESSING;

    if (!SetConsoleMode( hConsoleHandle: hOut, dwMode)) {
        exit( Code: GetLastError());
    }
#endif
}

// ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___
```

Bugs restants

Un bug persiste cependant, il survient lors de la rotation des Tetrominos.
Lorsqu'un Tetromino est adjacent à un autre, il arrive dans des cas assez rares qu'il puisse "s'encastrer" dans une structure après une rotation comme indiqué sur les images ci-dessous.





Changements par rapport à l'énoncé

Dans notre procédé nous avons malheureusement oublié une consigne, il s'agit de l'implémentation de l'observateur observé entre notre contrôleur et le modèle. Nous étions persuadés que cela était demandé uniquement dans la version avec interface graphique. Cependant nous l'avons remarqué bien trop tard que pour pouvoir modifier notre programme à temps.

Nous assurons cependant que nous implémenterons ce design pattern dans une prochaine version du jeu.

Temps consacré

Nous avons commencé à travailler sur le code du jeu durant notre semaine de vacances de mars. Cependant nous avons peu avancé et avons laissé de côté le projet durant plusieurs semaines par faute de temps.
Nous avons réellement commencé à travailler de manière significative environ 10 jours avant la date de la remise avec environ 1 à 2,5 heures de travail les jours d'école et 3 à 4 heures les week-ends et jours sans cours.

Nous sommes conscients que cela était loin d'être la meilleure solution et nous éviterons de refaire la même erreur

Cependant nous sommes tout de même contents du résultat actuel du jeu par rapport au temps consacré.

Conclusion

La réalisation de cette partie de projet s'est réalisée sans encombres, nous avons suivi ce que nous avions réalisé pour l'analyse. Seuls quelques petits réglages et corrections ont été apportés pour coller aux contraintes du langage C++.

Nous vous invitons à générer et à découvrir la documentation du projet pour de plus amples détails.