

Polytech Clermont

Ingénierie Mathématiques et Data Science

Rapport de projet

Tetris



Présenté par :

Corentin Prigent

Felix Baubriaud

Année 2022/2023

Résumé

Pour le projet du cours de Programmation Orienté Objet, Interface Homme-Machine et UML, il nous a été demandé d'implémenter le jeu Tetris.

Dans ce rapport, vous trouverez une brève présentation du Tetris ainsi que les problèmes que nous avons pu rencontrer, et de leurs résolutions, puis le bilan et la conclusion de ce projet .

Ce Tetris a été créé en C++ sous Qt. Nous avons créé des classes pour chaque objet intervenant dans le jeu (cases, puits, tétraminoes...) et des méthodes afin que les règles du jeu soient respectées. Quelques extensions au jeu de base ont été ajoutées : la possibilité de mettre en réserve le tétramino, d'accélérer sa descente ou encore de le faire directement tomber dans le puits.

Mots clés

Tétramino	Héritage	Classe	Méthode	Attribut
Puits	Case	Matrice	Affichage	Jeu vidéo

Abstract

For this project of Object Oriented Programming, Human-Machine Interface, UML, we were asked to make the video game Tetris.

In this report, you may find a brief presentation of Tetris as well as the issues we have encountered and their resolutions, then you will find the review and the conclusion of this project.

This Tetris was created on C++ with Qt. We created classes for each object related to the video game and methods in order to respect the game rules as much as possible. Some extensions were added such as the possibility to put in reserve tetrominos, to accelerate the downfall or also to lay the tetromino in the well.

Keywords

Tétramino	Heritage	Class	Method	Attribute
Well	Case	Matrix	Display	Video game

TABLE DE MATIÈRES

Résumé	2
Mots clés	2
Abstract	2
Keywords	2
Introduction	4
1. Présentation du jeu	5
1.1 Histoire du jeu	5
1.2 Règles du jeu de base	6
2. Planification du travail	7
3. Résolution des problèmes	8
3.1. Difficultés rencontrées	8
3.2. Solutions envisagées	8
3.3. Solutions retenues	9
4. Bilan	10
4.1. Avancement du projet	10
4.2. Les points à améliorer	10
4.3. Fonctionnalités complémentaires	10
4.4 Ressenti par rapport au projet (personnel, de groupe)	10
Conclusion	12

Introduction

Dans le cadre de notre deuxième année de cycle ingénieur en Ingénierie Mathématiques et Data Science, nous avons été amené à faire un tetriz en C++ avec Qt. Ce projet est un long processus de 2 semestres. Il regroupe les cours de UML, C++ et IHM. Nous avons commencé par travailler en groupe de 6 pour faire la partie UML. Puis, nous avons travaillé en binôme pour la partie programmation.

Dans ce rapport, nous commencerons par présenter le jeu. En deuxième partie, nous présenterons la planification du travail. Ensuite, nous parlerons des problèmes rencontrés et de leurs résolutions. Enfin, nous ferons un bilan de ce projet et conclurons ce rapport.

1. Présentation du jeu

1.1 Histoire du jeu

Tetris est un jeu créé en 1984, par Alekseï Pajitnov, ingénieur en informatique soviétique travaillant au centre informatique de l'Académie des sciences de l'URSS. En cherchant à reproduire l'un de ses jeux favoris, le pentomino, qui consiste à empiler des formes, il imagine le jeu Tetris et le programme.[1]

Tetris baptisé par la combinaison de “tetra” qui veut dire 4 en grec comme le nombre de carrés par tétramino et de “Tennis” qui était à l'époque le jeu vidéo favori de Alekseï Pajitnov. Le jeu eut un réel succès à l'international. Il est encore joué aujourd'hui par de nombreux joueurs.[2]

Ce jeu est connu pour être l'un des pionniers du jeu vidéo. Elle est encore aujourd'hui la deuxième série de jeux vidéo la plus vendue avec 494 millions de ventes derrière la licence Mario.[3]

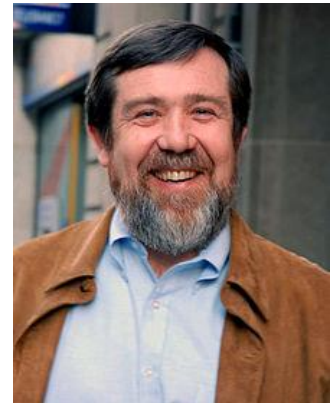

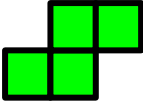
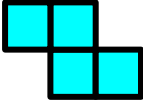

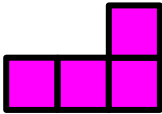
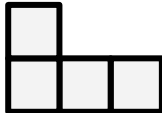
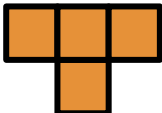


figure 1: Alexey Pajitnov
[4]

1.2 Règles du jeu de base

Le joueur manipule un tétramino à la fois parmi 7 formes différentes:

- Le I: 
- Le S: 
- Le Z: 
- Le O: 
- Le L: 
- Le J: 
- Le T: 

Il faut créer avec ces formes des lignes horizontales afin de faire un maximum de points. Le joueur peut pivoter et translater les pièces afin de les emboîter correctement. Plus le joueur supprime de lignes en un coup, plus il gagne de points. Supprimer 4 lignes en un coup s'appelle un tetrис. Le joueur gagne des niveaux à chaque fois qu'il supprime un nombre donné de lignes. À chaque niveau la vitesse de descente des pièces augmente, rendant le jeu de plus en plus difficile. La version la plus classique et la plus connue est sur NES. Le nombre de niveaux s'élève à 29 sur cette version. Dans le jeu de base, le score augmente à chaque tétramino posé. Dans notre cas, le score augmente seulement en faisant des lignes.

Quand les tuiles pleines arrivent jusqu'au niveau d'apparition du prochain tétramino le jeu s'arrête, c'est perdu. Une partie peut donc être jouée indéfiniment.

2. Planification du travail

DIAGRAMME GANTT						
		du 16/10/22 au 29/10/22	du 30/10/22 au 03/11/22	du 04/11/22 au 18/11/22	du 19/11/22 au 02/03/23	du 03/03/23 au 27/03/23
PHASE UML	Diagramme de cas d'utilisation					
	Diagramme de séquence système					
	Diagramme de classe d'analyse					
	Diagramme séquence					
	Diagramme de classe technique					
	Programmation					
	Rédaction des rapports					

figure 2: Diagramme de Gantt

Nous étions 6 personnes au départ pendant la phase UML. Nous avons continué à 2 pendant la programmation et aussi nous avons effectué quelques modifications des diagrammes afin qu'ils soient ajustés à notre tetriss. En effet, nous sommes partis sur un autre chemin de ce qui avait été proposé au départ lorsque nous étions 6 personnes.

Lors de la partie programmation, nous avons principalement travaillé ensemble à Polytech afin que l'un de nous deux ne se perde pas sur l'avancement du projet.

3. Résolution des problèmes

3.1. Difficultés rencontrées

1. Nous nous sommes rendus compte assez tard que dans les coefficients $(a_{i,j})$ des matrices, i représente les lignes donc lorsque i augmente, c'est la coordonnées y qui augmente. Inversement, lorsque j augmente, c'est la coordonnée x qui augmente.

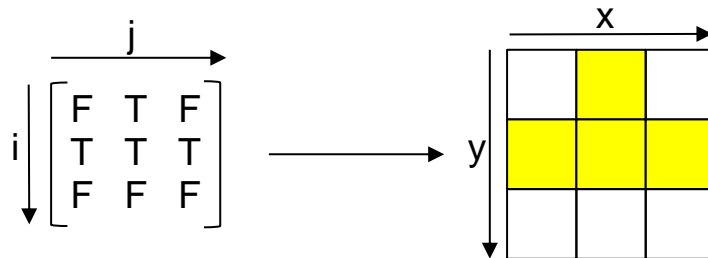


Figure 3 : matrice et affichage du tetromino 6

2. Nous avons passé beaucoup de temps sur la méthode *possibTranslation(direction)* de la classe Partie.
3. La méthode pour vérifier si le puits est plein a posé problème.
4. Lorsqu'on posait très vite les tétraminoes avec la barre d'espace, c'était toujours le même type qui apparaissait. Ceci était dû au fait que nous avons mis l'instruction *srand(time(0))* dans la méthode *pieceAleatoire()* alors qu'il fallait la mettre dans le constructeur de Partie. Mais même une fois cette instruction mise dans le constructeur, il subsistait un problème : le premier tétramino qui apparaissait était le même que le troisième, et le second le même que le quatrième.

3.2. Solutions envisagées

1. Nous avons pensé à intervertir les variables i et j de sorte que j soit les lignes et i les colonnes, mais ce n'était pas naturel et on pouvait vite se perdre par la suite.
2. La méthode *possibTranslation(direction)* était implémentée autrement au départ. On cherchait dans un premier temps le nombre de colonnes vides à droite ou à gauche de la matrice du tetromino selon la direction. Puis on regardait les cases situées à droite ou à gauche de la matrice du tetromino dans le puits. Si ces cases étaient pleines, on autorisait quand même un déplacement si on avait trouvé auparavant une colonne vide, et deux déplacements si on avait trouvé auparavant deux colonnes vides.
3. Au départ, nous n'avions pas de méthode *estFinie()* dans la classe Partie mais à la place une méthode *estPlein()* dans Puits, qui vérifiait si la case située à l'endroit où le tetromino apparaît était pleine et renvoyait *true* dans ce cas. Mais parfois cette case était vide et le tetromino qui apparaissait traversait une case pleine sans que la partie s'arrête.

4. Nous avons pensé à enlever l’instruction *srand(time(0))*, mais cela générerait alors un autre problème : les deux premiers tétraminoes étaient toujours les mêmes lorsqu’on commençait la partie.

3.3. Solutions retenues

1. Il a juste fallu changer l’affichage en faisant attention à mettre j avec la coordonnée x et i avec la coordonnée y.
2. En réalisant la méthode *possibRotation()*, on a vu qu’on pouvait procéder à peu près de la même manière pour *possibTranslation(direction)*. Nous avons donc opté pour cette méthode car elle est beaucoup plus courte et simple à comprendre.
3. Il fallait accéder au tetromino qui apparaissait pour savoir si elle pouvait apparaître ou non dans le puits. Nous avons donc créé à la place de *estPlein()* une méthode *estFinie()* cette fois dans la classe Partie, vérifiant si le tetromino est situé sur une case pleine du puits, et mettant fin à la partie dans ce cas.
4. Nous avons laissé le *srand(time(0))* dans le constructeur de Partie. Dans la fonction *pieceSuivante()*, nous avons ajouté deux fois l’appel de la fonction *rand()* de sorte que les nombre aléatoire 3 et 4, qui étaient respectivement les mêmes que les nombre 1 et 2, ne soient pas utilisés pour la génération de pièces. En faisant cela, il semble ne plus y avoir de problème.

4. Bilan

4.1. Avancement du projet

Coder le Tetris était une épreuve de patience. Au départ, nous avons implémenté les classes Tetrominos, Puits, Cases, Partie, MainWindows et Tetrominos6. En effet, il n'y avait qu'un seul type de tétramino au départ pour tester toutes les fonctionnalités. Nous avons commencé par les fonctions *afficher()* afin de voir clairement ce qu'on faisait. Ensuite, nous avons géré les déplacements du tétramino dans le puits et sa rotation, avant de s'intéresser aux collisions (*possibRotation()* et *possibTranslation(direction)*). Une fois que cela fonctionnait, nous sommes passés au remplissage du puits avec les tétraminos et à la génération d'un nouveau tétramino. C'est à ce moment-là que nous avons créé tous les autres tétraminos pour que la génération d'un nouveau tétramino une fois l'ancien posé soit aléatoire. Puis nous avons implémenté les méthodes permettant de supprimer les lignes, d'incrémenter le score et de vérifier si le puits est plein. Nous avons alors ajouté le timer pour que le tétramino descende tout seul et on en a profité pour ajouter un chronomètre comptant les minutes et secondes. Enfin, nous avons ajouté les améliorations, c'est-à-dire les méthodes *tomber()* et *reserver()* ainsi que la précondition de cette dernière *possibReservation()*. Bien sûr, nous sommes revenu plusieurs fois sur des méthodes déjà codées afin de les adapter aux nouvelles fonctionnalités. Nous avons également découpé quelques méthodes trop longues en plusieurs pour améliorer la lisibilité du code, et ajouté des méthodes de sorte qu'il n'y ait besoin d'inclure que la classe Partie dans MainWindow.

4.2. Les points à améliorer

Ce qu'on aurait pu faire, c'est un menu lorsqu'on lance le jeu, qui permettrait de consulter les règles du jeu et de choisir une difficulté. On pourrait aussi avoir l'option de faire une pause au cours de la partie, avec la possibilité de quitter, recommencer ou consulter les règles. Enfin nous aurions pu améliorer l'interface graphique en mettant des images, pour les tétraminos et pour le fond.

4.3. Fonctionnalités complémentaires

- La réserve (touche R) ;
- La descente plus rapide (touche flèche du bas) ;
- Faire tomber le tétramino dans le puits (touche espace) ;
- Le chronomètre s'incrémentant toutes les centièmes de seconde.

4.4 Ressenti par rapport au projet (personnel, de groupe)

- D'un point de vue organisationnel :

Dans un premier temps, l'organisation par rapport au groupe était assez compliquée. Ça partait souvent dans tous les sens, les diagrammes individuels ne correspondaient souvent pas aux attentes de tous. Donc nous étions souvent obligés de se mettre d'accord lors des réunions. Le suivi était aussi compliqué. Nous ne savions pas si nos diagrammes étaient corrects dans certains cas.

Dans un deuxième temps, l'organisation par rapport au binôme était beaucoup plus simple. Nous étions dans la majorité bien organisé, le travail avançait assez rapidement. Malgré certains différents sur les matrices et sur les valeurs à appliquer dans celles-ci, le projet a bien avancé.

- D'un point de vue langage :

Le C++ et Qt sont très bien adaptés pour coder un jeu comme Tetris, mais nécessite beaucoup de fichiers car pour chaque classe on a le fichier d'en-tête et le fichier sources. On aurait eu moins de fichier à gérer si on avait codé le jeu en Java car il n'y a pas besoin de fichiers d'en-tête.

- D'un point de vue temps :

Nous avons commencé le 11 janvier 2023 le premier cours de projet Programmation Orientée Objet et le rendu est le 28 mars au plus tard. Avec tout ce qu'il y a à faire à côté et la quantité de travail que demande la programmation du Tetris, c'est assez court.

Compétences (personnel)

- Quelles compétences vous ont le plus servi ?

Les compétences en C / C++ et en orienté objet principalement, mais aussi les algorithmes classiques qu'on a vu précédemment pour incrémenter certaines méthodes comme *supprimerLignes()* ou *reserver()*.

- Quelles compétences vous ont manqué ?

Les compétences sur le logiciel Qt. En effet, nous avons peu d'entraînement sur ce logiciel dont certaines fonctionnalités sont assez compliquées à utiliser.

- Sur quelles compétences avez-vous le plus progressé ?

Nous nous sommes bien familiarisés avec le logiciel Qt au cours du projet. Nous avons aussi progressé en C++ et Programmation Orientée Objet en général. Enfin, modéliser les diagrammes sur diagram.net nous a fait progresser en UML.

Conclusion

Ce projet dans l'ensemble nous a apporté beaucoup de compétences en C++ et Qt. De plus, il nous a apporté de fortes connaissances dans la conception de jeux. Avant cela nous devions faire tout sur le terminal, ce qui n'était pas pratique et cela manquait d'accessibilité à un plus large public. Pouvoir coder un jeu qui se joue avec les flèches directionnelles par exemple est un vrai cap dans notre apprentissage de la programmation. Il nous a aussi permis de développer notre créativité et notre travail d'équipe. Nous n'étions pas souvent d'accord sur comment la conception du jeu devait se faire mais nous trouvions toujours une solution qui nous arrangeait.

Coder un jeu comme Tetris est une très bonne base pour coder beaucoup d'autres jeux. On a pu évaluer le temps de travail nécessaire pour implémenter un jeu : vu le temps que nous avons mis pour coder Tetris, qui est un des jeux les plus simples, la quantité de travail doit être énorme pour les jeux qui sortent aujourd'hui.

Bibliographie

<https://www.playstudios.com/tetris/> [image page de garde]

<https://en.wikipedia.org/wiki/Tetris> [1]

<https://dailygeekshow.com/tetris-origines-histoire-creation-aleksei-pajitnov> [2]

https://fr.wikipedia.org/wiki/Liste_des_s%C3%A9ries_de_jeux_vid%C3%A9o_les_plus_vendues [3]

https://fr.wikipedia.org/wiki/Alekse%C3%AF_Pajitnov [4]

<https://openclassrooms.com/forum/sujet/tetris-besoin-d-infos-82930>

<https://tetris.com/play-tetris> [jeu sur lequel on s'est basé]