Projet d'initiation à l'ingénierie logicielle
L1 CMI
Rapport

Curie Logan Gallone Nathan Perrin Jérémy

Année 2019-2020

Table des matières

1 Introduction

Ce rapport vient conclure l'unité d'enseignement "Initiation aux projets d'ingénierie logicielle" de première année du Cursus Master Ingénierie en informatique de la faculté de sciences fondamentales de Besançon. Il s'agit d'un document présentant dans les grandes lignes le déroulement du projet de programmation d'un jeu vidéo (TETRIS). Cette unité d'enseignement était encadré par les professeurs Frédéric Dadeau et Julien Bernard de l'université de Franche-Comté.

2 Le Projet

Tetris est un jeu vidéo de puzzle où différentes pièces composées chacune de quatre carrés, dites "tétrominos" défilent du haut jusqu'au bas de l'écran. Le joueur doit compléter des lignes avec ces tétrominos ce qui lui rapporte des points pour son score. Le score étant plus ou moins impacté en fonction du niveau (plus le niveau est élevé et plus le score le sera en conséquence). Si le joueur remplit plusieurs lignes d'un seul coup avec un de ses tétrominos, le score sera également plus élevé.
Tetris n'a pas de fin, c'est à dire que le jeu s'arrête lorsque le joueur se fait submerger de pièces et que l'une de celle-ci reste bloquée en haut de l'écran.

2.1 Outils et méthodes de travail

Pour analyser le jeu et comprendre ses mécanismes, nous avons effectué de nombreuses recherches sur internet et nous avons également passé de longs moments à y jouer.

Étant donné que nous devions travailler à distances les uns des autres, nous avons d'abord créé un groupe sur Messenger, puis nous avons mis en place un serveur Discord afin de pouvoir communiquer plus facilement en vocal.

Pour ce qui était du développement, nous recherchions un outil de codage collaboratif. Après avoir testé Repl.it puis le partage Visual Studio Live Share, nous avons créé un dépôt sur GitHub, cela nous paraissait être la meilleur solution : en effet il y avait des bugs dérangeants sur Repl.it et Visual Studio Live Share nécessitait que celui ayant créer le partage soit connecté afin que les autres collaborateurs puissent avoir accès au code partagé. Ainsi avec GitHub, chacun codait sur son environnement de développement favoris, puis uploadait ses modifications au programme sur le dépôt et nous pouvions donc travailler en asynchrone sans problèmes.

Afin de coder le jeu, nous nous sommes principalement basés sur les algorithmes en PDL de l'analyse réalisée en amont. Nous avons transcrit ce code PDL en langage JavaScript tout en lui apportant certaines modifications nécessaires.
Pour la répartition du travail, vu que le programme était constitué de multiples algorithme, nous nous les sommes répartis entre nous, tout en s'aidant

mutuellement en cas de difficulté.

2.2 Développement de l'application

Lors du développement de l'application, nous ne pouvions pas simplement copier-coller les codes de notre analyse. D'une part parce que le langage PDL et le JavaScript sont très différents, d'autre part parce que la boucle de jeu dont nous parlions dans l'analyse est devenue une fonction récursive, et finalement parce que nous avons décidé d'implémenter le jeu en tirant profit de l'aspect orienté objet de JS. De plus, le programme contient certaines fonctionnalités qui ne figuraient pas dans l'analyse du jeu.

2.2.1 Orientation Objet du programme

Lors de l'analyse, il est question de structures pour stocker les informations. Puisqu'au cours du projet, nous avons reçu des cours sur la programmation orientée objet, nous avons décidé de les mettre en pratique dans ce projet. Ainsi, une instance de Classe Game nous sert à stocker de nombreuses informations : le tetromino en cours de placement, le prochain tetromino, le score, le niveau, les coordonnées des lignes complétées au dernier placement de tetromino... Cela a rajouté une certaine difficulté dans la programmation car il fallait veiller à ranger toutes les fonctions de l'analyse du jeu dans les classes appropriées. Néanmoins, les fonctions utilisées s'en son retrouvées simplifiées car tous les paramètres étaient désormais des attributs des objets que nous manipulions.

2.2.2 Rotation des tetrominos

L'une des différences majeures entre notre code PDL et le programme en JavaScript est l'algorithme de rotation d'un tetromino. Dans l'analyse, nous comptions réaliser un tableau de tetrominos de 4x7 qui nous servirait à stocker toutes les coordonnées des blocs d'une pièce selon si elle a déjà effectué 1, 2, 3 ou 4 quarts de tours. C'était une solution très fastidieuse que nous avons abandonné rapidement au profit de "switchs imbriqués dans un switch". Le résultat final n'est pas beaucoup plus lisible que le tableau d'origine, cependant, il évite de recalculer toutes les coordonnées des blocs de tous les tetrominos puisqu'ici, on ne s'intéresse plus qu'à celles du tetromino en cours de placement.

2.2.3 chute rapide avec la barre espace

Une fonctionnalité à laquelle nous n'avions pas pensé lors de l'analyse est la chute rapide d'une pièce. Elle a lieu à chaque fois que le joueur appuie sur la barre espace. Nous l'avons donc implémenté.

2.2.4 Gestion de la vitesse de chute des tetrominos

Dans l'analyse de Tetris réalisée en amont du développement, nous avions simplement évoqué que la vitesse de chute des tetromino était fonction du niveau du joueur sans même entrer plus dans les détails ni expliciter une formule permettant de calculer ce temps de chute. Ainsi il nous a fallu déterminer cette équation pour la mettre en place dans le programme, pour cela nous avons effectué des tests sur le jeu pour Game Boy. Les test consistait à chronométrer le temps que mettait un tetromino à parcourir la grille de haut en bas pour chaque niveau. Ainsi, pour chaque niveau de 1 à 9, nous faisions une dizaines d'essais puis calculions le temps moyen de parcours. Ensuite nous avons réalisé la courbe suivante, puis extrait son équation.

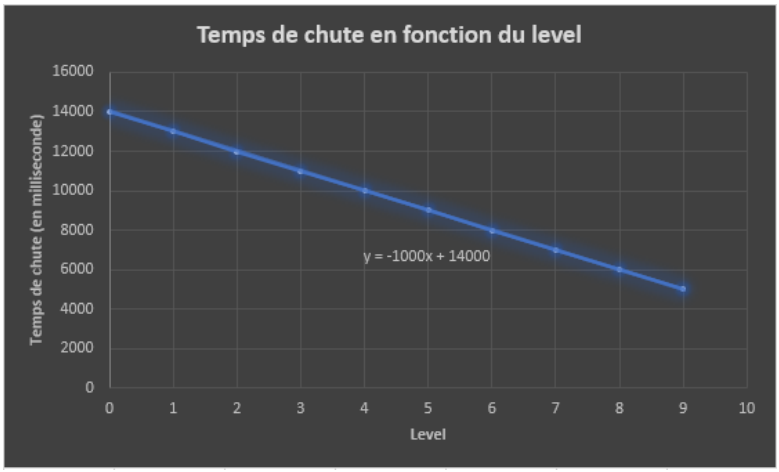

FIGURE 1 – Temps de chute moyen d'un tetromino en fonction du niveau du jeu

Comme nous pouvons le constater, le temps de chute en fonction du niveau est linéaire, cela nous posait problème car, arrivé à un niveau important, le temps de chute deviendrait négatif (et donc causerait des bugs). Ainsi, plusieurs solutions nous sont parvenues :

— Garder cette formule et brider le jeu à un niveau limite ou le temps de chute n'est pas négatif,
— Garder cette formule, mais ne plus changer la vitesse de chute à partir d'un certain niveau,
— Concevoir une nouvelle expression du temps de chute en fonction du niveau qui converge vers 0^+ lorsque le level tend vers $+\infty$.

Les deux premières idées citées était contraire à l'optique du jeu, l'une empêche l'évolution du niveau alors que le joueur pourrait potentiellement compléter de nouvelle lignes. L'autre empêcherait d'accélérer la chute du tetromino (et donc la difficulté) lorsque le niveau augmente.

Nous avons donc retenu la dernière solution consistant à déterminer une nouvelle expression du temps de chute. Pour cela, nous avons considéré la fonction $x \mapsto e^{-x}$, et avons ajouté des paramètres afin d'avoir une vitesse de chute convenable. Nous avons donc effectué de nombreux essais afin de déterminer une accélération de la chute correspondant à nos attentes. Finalement l'expression du temps de chute en fonction du level est définie par $1000 \times e^{-0.16 \times level}$

2.2.5 Intégration de la musique au projet

Une grande différence entre l'analyse purement théorique du jeu et sa réalisation réside dans le fait que nous avons intégré une musique en fond sonore, musique qui d'ailleurs est iconique et fait le charme du jeu Tetris, peu importe sa version. Nous ne pouvions pas, de ce fait , nous permettre d'omettre l'ambiance musicale.
Pour intégrer la musique au projet, nous nous sommes servi des fonctions offertes par le langage JavaScript pour manipuler les contenus sonores. L'idée était d'intégrer une musique qui serait jouée automatiquement au lancement du jeu et qui tournerait en boucle. S'est alors posé un problème auquel nous avons dû nous confronter : depuis les dernières mises à jour des principaux navigateurs utilisés, les contenus tels que de l'audio lancé automatiquement en arrière plan sont bloqués nativement par les navigateurs. Par soucis d'ergonomie, nous aurions préféré que l'utilisateur n'ait pas spécialement à toucher les paramètres de son navigateur pour faire marcher correctement toutes les features du jeu. Nous avons dû chercher une solution pour contourner le problème. Après quelques recherches sur le web, une idée pour corriger ce souci aurait été de faire appel à l'utilisation d'une trigger au lancement du

jeu afin de by-pass la sécurité du navigateur et ensuite lancer le contenu qui nous intéressait. Cependant après avoir testé cette solution, il semblerait que celle-ci ne soit plus d'actualité. Nous avons donc finalement implémenté un autre système pour palier ce problème, de plus, il offrirait d'autres possibilités de contrôle. Il s'agit de commandes de volumes, arrêt/pause de la musique. Le joueur a de ce fait plus d'options et peut s'il le souhaite se passer de la musique.

2.2.6 Gestion des boîtes de dialogue

Pour indiquer la défaite du joueur à la fin d'une de ses parties, nous voulions afficher un message au moyen de boîtes de dialogue JavaScript. Néanmoins JavaScript n'offre pas par défaut la possibilité de choix multiples pour l'utilisateur (oui/non), ce qui était gênant dans notre cas. C'est la raison pour laquelle nous avons implémenté des boîtes de dialogue personnalisées dans le jeu.

Pour modéliser ces boîtes de dialogue, nous avons eu recours à l'utilisation de la librairie de fonctions JQuery afin de nous offrir un panel de personnalisation plus conséquent que celui offert par les fonctionnalités de base du langage JavaScript. Toutefois pour simplifier la tâche d'implémentation et nous gagner un peu de temps, nous avons eu recours à une sous librairie de JQuery intitulée really-simple-jquery-dialog. La librairie a pour but, comme il est mentionné dans son nom, de nous octroyer la possibilité de créer des boîtes de dialogue très simplement, et ce, en seulement quelques lignes de codes.

2.2.7 Esthétique et gestion des liens du projet

Après avoir codé le jeu et obtenu une version fonctionnelle, nous avons voulu avoir une page html un minimum esthétique, et ne pas avoir juste un canvas sur une page blanche. Nous avons d'abord mis la page en noir, ajouté le titre du jeu, et créé une bordure autour du canvas. Puis étant donné que Logan suit des cours sur les langages HTML et CSS, il a voulu mettre ses connaissances acquise au profit du projet, et s'est donc chargé de la mise en page du jeu tout en suivant les idées de ses collègues. Ainsi une image de fond a été posé, le logo de l'UFR ST a également été importé sur la page, puis nous avons choisi une police d'écriture originale pour marquer le titre

du jeu.

Ensuite, notre créativité s'est imposé et une page d'informations a été crée, pour y accéder nous avons inséré un lien vers celle-ci depuis la page principale. Comme vous avez pu le constater sur cette page de renseignements, nous avons inséré une fenêtre de crédits et une touche d'humour avec nos interviews. C'est également depuis cette page que l'on peut accéder à la page SoundCloud du compositeur de la musique utilisée, à l'analyse préliminaire du jeu et à ce rapport.

2.3 Quelques pistes d'améliorations

2.3.1 Affinement de l'équation de l'accélération de la chute des tetrominos

Nous avons mentionné plus tôt dans ce rapport la nécessité de modifier l'équation gérant l'accélération des tétrominos en fonction du niveau. Toutefois, la solution que nous avons privilégié (c'est à dire exprimer la chute en utilisant une fonction $\begin{array}{rcccc} f & : & \mathbb{R} & \to & \mathbb{R} \\ & & x & \mapsto & Ae^{-\lambda x} \end{array}$ où $A, \lambda \in \mathbb{R}$) peut amener à nous faire réfléchir quant à la possibilité d'adapter quelques peu la solution afin d'optimiser au mieux la difficulté du jeu. Effectivement nous pensons qu'il serait intéressant d'ajouter au joueur la possibilité de choisir un certain niveau de difficulté auquel il désirerait jouer (on pourrait imaginer 3 niveaux de difficultés différents : facile, moyen, difficile). Nous aurions ensuite à déterminer 3 équations, une pour chaque niveau. L'intérêt est que nous pourrions faire débuter chaque choix de difficultés avec la même vitesse de chute pour le niveau 0 et augmenter plus ou moins fortement à chaque passage de niveau, l'accélération en fonction du choix de difficulté choisi en amont.

2.3.2 Ajout d'une option pour mettre en pause le jeu

Nous n'avons pas implanté d'option pour mettre en pause le jeu dans le projet initial. Cela constitue de ce fait une amélioration notable que nous pourrions ajouter au jeu. Nous aurions à réfléchir à une manière de l'implémenter dans le jeu. En effet, l'une des caractéristiques propre au langage JavaScript est que ses fonctions ont pour particularité de ne pas être bloquante. De ce fait, nous aurions à trouver un moyen de contourner ce pro-

blème puisque l'ensemble du jeu continuerait à tourner en arrière plan si nous nous contentions seulement d'afficher un écran de pause.

2.3.3 sauvegarde des High-scores

Une autre piste d'amélioration serait d'implémenter un tableau des meilleurs scores. À la fin de chaque partie, le programme demanderait à l'utilisateur de saisir un pseudo et enregistrerait le score du joueur dans un fichier. Ce fichier serait lu à chaque fin de partie pour afficher tous les meilleurs scores effectués sur cette machine.

2.3.4 Autres améliorations

Comme nous l'avons exprimé précédemment nous avons une créativité assez fertile, et pour rendre ce jeu plus original que les versions déjà réalisé, quelques piste d'amélioration nous sont venus en tête, tel qu'utiliser une grille d'un autre format, et ne pas se limité à cette 'convention' d'une grille de 20 x 10 carreau.
Il serait également possible d'ajouter une nouvelle pièce bonus qui aurait une action spéciale, par exemple éliminer les lignes sur lesquelles elle se trouve une fois fixée, et ce sans tenir compte de leur complétion ou non. Cette pièce bonus ne pourrait s'appeler tetromino car elle ne pourras être composé de 4 blocks car toutes les combinaisons possibles avec 4 blocks sont déjà réalisé.

3 Conclusion

Au cours de ce projet, nous avons pu constater quelques-unes des difficutés auxquelles on fait face lors de développement de sites web interactifs grâce à JavaScript. La découverte des outils que proposent le langage fut enrichissante pour chacun d'entre nous. La réalisation du jeu nous a fait constater que certains éléments de notre analyse était incorrects ou pas adaptés à un programme concret. Néanmoins nous avons pu nous appuyer dessus pour débuter le projet, ce qui nous laisse quelques pistes si nous souhaitons nous lancer dans la conception d'autres jeux.