Dungeon Defense

Rapport de Programmation Java

Quentin Ducasse
Quentin Lacoux
Tristan Cuyala
Hugo Lagahe

Encadrant :
Nathalie HOURDEBAIGT
Elisabeth RANISAVLJEVIC
Seytkamal MEDETOV

Rapport du
30 Mai 2020

Sommaire

1 Introduction

Notre projet correspond à la matière Travail en équipe du second semestre de première année d'ingénieur de CY Tech. Il nous a été demandé de réaliser un jeu en Java et nous avions la possibilité d'utiliser JavaFX, essentiel pour la partie graphique. Le contexte particulier nous a obligé à travailler à distance.
Notre objectif est de réaliser un jeu dont nous soyons fier lors du rendu.

2 Dungeon Defense

2.1 Mode d'emploi

Votre château est attaqué, vous devez empêcher les hordes d'ennemis de traverser la salle qui mène au donjon, lieu où la famille royale et les habitants se sont réunis !

Afin de vous défendre, il faudra placer stratégiquement vos unités sur le plateau, en tirant parti des murs déjà présents dans la salle. Pour cela, vous devez appuyer à l'aide du clic gauche de la souris sur l'une des unités se trouvant en bas de l'écran, la faire glisser jusqu'à la case souhaitée et enfin relâcher le clic gauche. Chaque unité coûte un certain montant d'or, qui augmente à chaque achat. La quantité d'or que vous détenez augmente au cours du temps mais vous pouvez également récupérer des pièces d'or apparaissant dans la salle ou bien laissées par les ennemis que vous avez tué.

Pour ne pas perdre, il faudra empêcher les ennemis d'atteindre le côté gauche de la salle : si un seul d'entre eux atteint l'autre côté de la salle, les personnes se cachant dans le donjon ne pourront pas se défendre et votre aventure se terminera.

2.2 Choix de design

Tout d'abord, nous nous sommes dit que l'histoire devait rester simple pour un tower defense, c'est pourquoi celle-ci ne fait qu'une phrase. Cela permet de donner un léger contexte au joueur qui lit le mode d'emploi mais n'empêche pour autant pas un joueur ne le lisant pas de ne pas pouvoir profiter du jeu.

Pour le gameplay, nous avons suivi le système du tower defense "Plants vs. Zombies", où les ennemis arrivent d'un côté de l'écran et doivent atteindre le côté opposé, le but du joueur étant de les en empêcher. Cependant, cela aurait été un peu simple que les ennemis n'avancent que dans une seule direction, nous avons donc décidé de complexifier ce système en ajoutant des obstacles, représentés par des murs, sur leur chemin. Ainsi, les ennemis empruntent des chemins qui leur permettent de se rapprocher de leur objectif tout en évitant ces murs. De même, afin d'éviter que les ennemis qui suivent un chemin commun finissent tous au même endroit, chaque ennemi choisit une ligne sur laquelle il va

finir son trajet, permettant ainsi d'éviter qu'ils soient tous bloqués par le même allié par exemple.

De plus, le jeu se joue uniquement avec la souris car c'est la manière la plus pratique de jouer à un tower defense. Pour placer les unités, nous avons choisi un système "click and drag" : le joueur clique sur l'unité qu'il veut placer, la fait glisser sur la case sur laquelle il veut la placer puis lâche le bouton de la souris.

Pour la partie graphique du jeu, nous avons tout d'abord utilisé "16x16 DungeonTilseset II" de 0x72, disponible gratuitement ou en don sur le site itch.io, car les sprites nous ont beaucoup plu et étaient pour la plupart bien adapté pour un tower defense.

Un autre pack graphique que nous avons utilisé est le pack TOPDOWN SHOOTER de Jason Perry.

2.3 Choix de conception

Les différentes classes que nous avons créé pour notre projet sont : Allié, Attaquant, BarreVie, Carte, Case, Ennemi, Item, Jeu, MainJoueur, Projectile et Sommet.

Attaquant est la classe mère de Allié et Ennemi. Allié est la classe gérant tout ce qui a un rapport avec les unités que le joueur peut jouer, tel que la vérification de la présence d'un ennemi sur la même ligne, l'affichage d'une unité sur le plateau ou bien les différentes façons d'attaquer selon l'unité. La classe Ennemi gère de la même façon les ennemis du jeu, avec des méthodes leur permettant de trouver un chemin entre les obstacles ou permettant d'attaquer

BarreVie est la classe permettant d'afficher la barre de vie des ennemis et des alliés, avec des couleurs différentes respectivement. C'est également grâce à cette classe que le niveau des ennemis et des alliés est affiché.

Carte est la classe permettant de créer et afficher les alliés qu'il est possible de jouer dans la main du joueur. Ses méthodes permettent notamment de placer un allié sur le plateau ou bien de gérer le temps qu'il faut attendre avant de pouvoir reposer un allié(le cooldown).

Case est la classe contenant toutes les informations qu'une case du plateau contient, comme la présence d'un allié ou bien d'un ennemi. Elle a également une méthode qui détecte quelles sont les cases voisines à la case concernée et qui contiennent un mur, cette méthode étant utilisée lorsqu'un ennemi détermine le chemin qu'il doit suivre.

Item est une classe permettant d'afficher les objets ainsi que de déterminer la zone sur laquelle il faut cliquer afin de le récupérer. Bien que cette classe avait été créée dans le but d'avoir plusieurs objets, nous n'en avons finalement qu'un, qui est la pièce.

Jeu est la classe dans laquelle se trouve la méthode main, c'est dans cette classe notamment que sont les méthodes pour initialiser le plateau avec les obstacles, pour mettre a jour tous les elements du jeu et les afficher aussi. Le jeu a differents status qui sont les status "Menu" ,"Jeu" , "Pause" et "Fin"

MainJoueur est la classe qui génère et affiche les alliés que le joueur peut placer et qui sont situés en bas de l'écran.

Projectile est la classe qui permet de créer et d'afficher un projectile tiré par un allié, les méthodes permettant d'afficher le projectile ou bien de produire un résultat différent selon s'il touche un ennemi ou bien un mur. Les projectiles créés sont ajoutés dans une liste de projectiles qui est stockée dans l'allié qui tire ces projectiles.

Sommet est une classe qui stocke le père d'une case, son poids et un booléen check et permet d'appliquer l'algorithme de Dijkstra afin de déterminer le chemin que les ennemis doivent suivre.

2.4 Description des principaux algorithmes

2.4.1 Algorithme de Dijkstra

Autrement connu sous le nom de "algorithme du plus court chemin", celui-ci nous a bien aidé pour trouver l'itinéraire que les ennemis doivent prendre. Le plateau original étant un plateau de Case en deux dimensions, il a fallu prendre en compte premièrement que les cases soient considérées comme des sommets d'un graphe non orienté, non valué. Une fois cela fait, la classe Sommet a été créée de telle sorte à ce qu'elle contienne:

- la Case qui lui sert de père,
- le poids de la case,
- si elle a déjà été vérifiée ou non.

Tout cela conformément aux conditions requises pour l'algorithme de Dijkstra. Il ne reste plus qu'à relier une Case avec un Sommet via une HashMap et le tour est joué pour implémenter l'algorithme de manière normale. Une autre variation a cependant été faite. En effet, les ennemis changeaient de direction au dernier moment, ce qui compliquait considérablement la tâche pour le joueur. Pour pallier cela, la case de fin pour Dijkstra a été fixée sur la même ligne d'arrivée, mais 5 cases avant, pour que le joueur ait le temps de voir venir l'ennemi.

Figure 1: Image du jeu.

2.4.2 Algorithme de génération du terrain

Le terrain est généré aléatoirement grâce a une sorte de parcours en profondeur. On y place 5 murs aléatoirement pour les deux cordonnées. Chaque mur placé a 50% de chance de générer un mur en dessous ou au dessus de lui-même, et ainsi de suite. Nous avons choisi de le faire ainsi car ce projet devait aussi servir à montrer ce que nous avons appris durant le semestre, pas seulement en Java, mais aussi dans d'autres matières, et cela représentait une bonne opportunité pour le montrer.

2.4.3 Evènement MouseRealeased

Ce n'est pas un algorithme du cours, mais c'est un bon exemple de comment les choses fonctionnent en général dans le jeu. Tous les alliés, tous les ennemis, tous les items et tous les projectiles sont regroupés dans des listes. Ainsi, pour pouvoir actualiser tout d'un coup,

il suffit de parcourir ces listes et appeler leur fonctions update(). De même pour vérifier, par exemple, si un allié a été sélectionné, on parcourt tous les alliés dans la boutique et on trouve lequel est en train d'être déplacé depuis la boutique jusqu'au plateau.
Beaucoup de choses dans le code ont été implémentées de cette manière. On regarde dans tous les ennemis, alliés, projectiles... si l'un deux vérifie la condition de départ.

Pour ce projet, nous avons utilisé l'IDE Eclipse car c'est l'IDE que nous avons utilisé durant le second semestre et il nous semblait logique d'utiliser un IDE que nous maîtrisions tous.

3 Présentation du sujet

3.1 Présentation du sujet, outils utilisés (QQOQCP)

Le sujet du jeu était simple, utiliser une grille à deux dimensions lors de l'implémentation. Le sujet étant large nous avons d'abord décidé d'effectuer un brainstorming pour avoir le plus d'idées possibles.
Voici les différentes idées de jeu qui en sont ressorties :

- tower defense
- battle royale
- infiltration/stealth
- top down shooter
- metroidvania(jeu de plateforme)
- plant vs zombies

Le mode battle royale nous paraissait trop complexe étant donné le peu de temps à notre disposition, surtout pour les aspects de jouer contre d'autres joueurs ou de créer des intelligences artificielles. Nous avons hésité avec un jeu d'infiltration mais avons finalement décidé de nous inspirer du jeu "Plants vs Zombies" et de la catégorie tower defense.

Après avoir fait notre brainstorming et avoir trouvé la catégorie du jeu, nous avons décidé de faire un QQCOQPC pour pouvoir mieux définir notre jeu, le voici :

- Q (Qui) : Nous, étudiants allons coder le jeu. Il doit être compréhensible pour tout type de personne.
- Q (Quoi ?) : Un jeu de tower defense.

- O (Où ?) : Chacun chez nous, CY Tech étant fermé pendant cette période, l'utilisation de GIT est essentielle.
- Q (Quand ?) : Du 15 mai au 31 mai 2020.
- C (Comment ?) : En utilisant le langage de programmation Java et les librairies JavaFX.
- P (Pourquoi ?) : Pour réaliser un projet de fin de semestre en Java et travailler en équipe.

Nous avons ensuite élaboré un diagramme de Gant pour s'organiser :

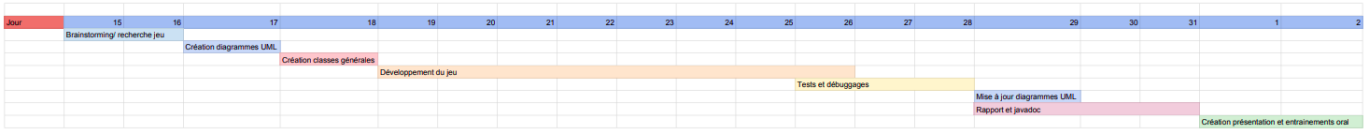

Figure 2: Excel présentant notre diagramme de Gant.

3.2 Difficultés lors du projet

Une des plus grosses difficultés lors du projet a été de faire fonctionner correctement la méthode pour que les ennemis se déplacent. En effet, après vérification, il s'est avéré que la méthode fonctionnait mais que l'affichage de l'ennemi ne suivait pas les bonnes coordonnées, il nous a fallu beaucoup de temps pour résoudre ce problème. De plus, nous avons aussi perdu beaucoup de temps à cause d'une incompréhension dans le groupe sur la manière dont les x et y étaient placés sur le plateau, ce qui a eu pour conséquence que certaines méthodes ne fonctionnaient pas comme voulues.

3.3 Résultat de votre projet

Le résultat de notre projet est un jeu nommé Dungeon Defense, il est disponible pour un joueur. Le but est de survivre aux vagues d'ennemis qui arrivent. Pour cela il faudra utiliser les mines pour générer de l'argent, ramasser les pièces d'or tout en posant de défenses qui détruiront les ennemis.

4 Présentation de l'équipe

4.1 Auto-évaluation

Notre équipe choisit aléatoirement est composée de : Tristan CUYALA, Quentin LACOUX, Quentin DUCASSE et Hugo LAGAHE. Nous avons décider pour se répartir les tâches de

les lister et que chaque personne en prennent au fur et à mesure.
Tristan a ainsi choisi tout d'abord de créer les classes avec leurs constructeurs puis de travailler sur la première version du déplacement des ennemis et finalement d'apporter de l'aide sur les autres fonctionnalités.
Quentin DUCASSE a choisi de créer la structure du jeu ainsi que les Classes Allié, Ennemi, Projectile et MainJoueur.
Quentin LACOUX a lui choisi de se pencher sur les algorithmes de graphes et comment utiliser ces connaissances de cours dans ce projets.
Enfin, Hugo a choisi de travailler le design en priorité avec le menu, la grille de jeu, les sprites (la représentation) des ennemis et des alliés, les projectiles et le son du jeu.

Pour la communication nous avons choisis 3 applications principalement : Messenger, Teams et Git.
Avec Messenger nous communiquions tout au long de la journée pour se répartir les tâches et donner notre avancée. Teams nous servait pour travailler ensemble, c'est à dire de pouvoir se motiver à travailler plus longtemps, de pouvoir effectuer un partage d'écran pour résoudre un problème (nous avons d'ailleurs très souvent utilisé cette fonctionnalité). Enfin Git pour se partager le code mais également pour voir les commits (les commentaires des fonctionnalités ajoutées), ce qui était aussi essentiel.

4.2 Travail individuel sur le déroulement de ce projet

4.2.1 Hugo LAGAHE

Dès le début je voulais effectuer ce projet car je savais qu'il me serait bénéfique pour le stage de programmation que l'on peut effectuer cette année. J'ai aimé faire ce projet car nous avions tous la volonté de rendre un projet de qualité, pour y arriver nous avons eu une méthode que je n'avais expérimentée avant, qui était plus souple, plus adaptée en cette période. C'est à dire que nous devions avancer dans l'implémentation du jeu et pour cela les horaires étaient variables.
Le fait de travailler à distance ne m'a pas perturbé, les 3 applications étant très utiles. J'ai beaucoup apprécié le fait de recevoir de l'aide lorsque je le demandais, de pouvoir faire un simple partage d'écran pour que l'on réfléchisse à plusieurs. Tout le monde étant toujours très coopératif. J'ai également découvert l'aspect graphique d'un projet et cela m'a montré une nouvelle manière de contribuer.

4.2.2 Quentin LACOUX

Pour ma part, j'ai toujours porté beaucoup d'intérêt au projets informatiques durant mes années à l'EISTI et même antérieurement. Je sais que cette façon de travailler sera celle que je devrais adopter pour mon emploi futur alors autant m'y faire dès à présent. À l'exception d'une seule journée(pour raison médicale) , j'y ai travaillé tous les jours car il y

a de toute manière toujours quelque chose à améliorer. Des réunion sur Microsoft Teams étaient organisées quotidiennement, pour travailler ensemble,pour demander de l'aide, ou tout simplement pour faire le point et se répartir les tâches et cela a aidé à garder le groupe en activité, même à distance.

4.2.3 Quentin DUCASSE

J'ai adoré participer à ce projet car aimant beaucoup les jeux vidéos j'ai pu me placer pour une fois du coté des développeurs et découvrir ce milieu. Je m'étais déjà intéressé au développement de jeu de mon coté avant de commencer le projet donc ce Jeu Dungeon defense m'a permis de consolider mes connaissances acquises et d'en apprendre de nouvelles.J'ai été très satisfait de l'ambiance du groupe qui était très bonne et ce projet m'as permis d'être plus à l'aise en travail de groupe donc cela a été très bénéfique pour moi.

4.2.4 Tristan CUYALA

J'ai trouvé ce projet très intéressant car c'est le premier projet que nous avons fait à l'EISTI qui nous laissait autant de liberté et qui nous permettait véritablement de faire un brainstorming complet car tous les aspects du jeu étaient à définir. Bien que la situation pour travailler était assez complexe, nous avons pu utiliser Teams pour nous organiser efficacement en faisant régulièrement des réunions et pour nous entraider lorsque nous avions des difficultés sur certains aspects du code, il était donc possible de travailler tous les jours et de ne pas prendre de retard sur le planning du projet. C'était également intéressant de travailler sur un jeu de manière plus poussée que durant nos anciens projets, notamment sur l'aspect graphique.

5 Conclusion.

En conclusion nous sommes fiers de notre projet, faire un jeu vidéo nous a tous passionné et chacun a développé la partie qu'il voulait. Nous aurions souhaité pouvoir l'améliorer avec des niveaux, ou une meilleure gestion de la difficulté mais cela ne nous semblait pas réalisable dans le temps imparti. Nous avons géré le travail en équipe en tenant compte des capacités de chacun, la bienveillance et l'envie de réussir nous ont permis de garder un bon esprit tout au long du projet.