Année 2023-2024

Rapport de Projet - POOIG

Réalisation d'un Tower Defense - Groupe 37

Projet réalisé par
PERRIER-BABIN Ludivine
YAO Alex

L’organisation pour réaliser les charges :

Nous sommes partis sur l’idée d’un plants vs zombies donc un tower defense sur plusieurs lignes droites. on a commencé par faire plusieurs dossiers qui permettent de séparer les différents fichiers en différents package, entre les fichiers main qui permettent de lancer le code, les fichiers entité qui sont sous divisé entre les ennemis et les tours et qui partent tout les deux d’un fichier abstract Entite, les fichiers pour les tuiles et leur gestion pour implémenter les différentes maps, les ressources qui sont sous divisé par dossier pour séparer les futurs images pour les tours, ennemis, les maps qui seront des fichiers txt pour implémenter celles ci plus facilement, le curseur, et les tuiles pour les maps.

A - Le plateau

On a commencé par implémenter une map de base pour avoir une référence visuel avec map1.txt et un fichier TuileGestion.java pour bien mettre chaque numéro de case avec une image qu’on a dessiner avec ToolKit dans un tableau pour pouvoir les récupérer après avec d’autres méthodes dont Draw qui dessine chaque tuile.

L’affichage graphique avec JPanel de java.swing pour faire un gamePanel donc la fenêtre et tout l’affichage et ce qui va lancé aussi le jeu et le faire tourné. Le tout est appelée dans le fichier Main qui contient donc le main qui ouvre une fenêtre avec JFrame on lui donne le nom du jeu, Time To Tower, et c’est là que l’on crée le GamePanel et qu’on y met les données de la fenêtre, on a aussi personnaliser le curseur. au départ on avait fait un fichier pour contrôler le curseur avec le clavier et selon la touche que l'on appuyait le curseur se déplaçait on a rapidement changé et passer au curseur avec la souris avec MouseListener pour les clics et MouseMotionListener pour suivre chaque mouvement de la souris.

B - Les ennemis

On a ajouté une première entité : un zombie basique au tout début afin d’avoir une idée sur quoi s’attendre. Pour cela on a créé déjà le fichier qui sert de base pour toutes les entité avec des méthodes de base, ensuite pour les ennemis un fichier portant ce nom pour les éléments qui seront en commun pour tous les ennemis puis un fichier EnnemiGestion pour gérer les zombie et les faire apparaître aux endroits défini donc 5 lignes avec 5 zombies. un autre fichier d’un premier zombie le plus basique qui hérite directement de ennemi qui aura sa propre vitesse, sa propre force d’attaque, son propre cooldown et ses propres points de vie.

Ensuite il a fallu faire les fichiers pour les autres types de zombies car ils seront très différents avec certains qui auront plus ou moins de récompenses d'argent, une vitesse plus élevée, plus de points de vie ou encore qui font plus de dégâts. on a fait comme pour les tourelles sauf que ce sont les zombies qui apparaîtront tout seul avec les wave ou les différents niveaux implémentés plus tard.

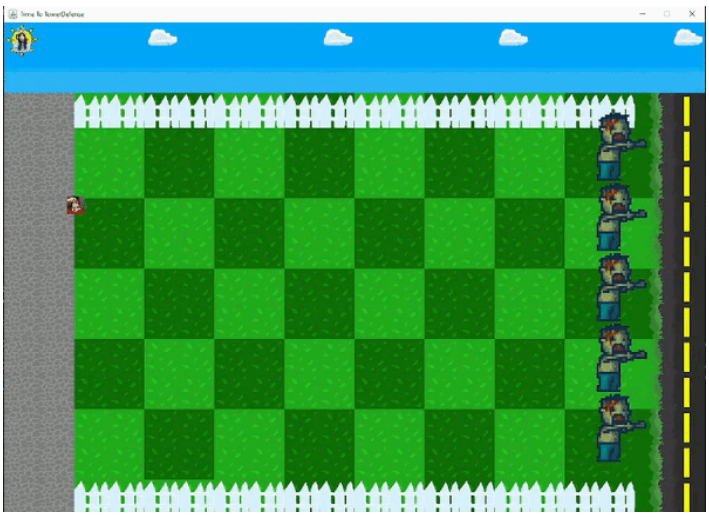
Le système de wave en cours d’implémentation était assez compliqué car on voulait faire d’abord le jeu en mode infini donc avec des waves différentes qui sont

complètement différentes les unes des autres avec les différents zombies implémenté qui ont des valeurs différentes les uns des autres donc avec un random qui inclut des probabilités. le système de wave est implémenté dans update ou on vérifie si déjà il n'y a plus d'ennemis sur le terrain ni d'ennemis restant à mettre sur le terrain si oui mais que la wave est toujours marqué en cours car le boolean est false, c'est que la wave vient de se terminer donc on on change ce boolean en true car la wave n'est plus en cours et on ajoute le temps avant la prochaine wave sinon si la wave est déjà marqué fini et que le temps avant la prochaine wave est terminé on crée alors une nouvelle wave qui appel la fonction newWave sinon si la wave n'est pas fini on continue a vérifié si les ennemis sont en vie et attaque s'ils sont en face d'un tourelle.

Cette fonction newWave est assez longue car on utilise un random et on ajoute des ennemis pour la wave et on fait un switch pour savoir quel type de zombie on va avoir. Pour les probabilité on a ajouté une fonction updateProbaTab qui va changer celles ci jusqu'à la 20e wave avec un switch car après on considère que les variables ne changeront plus. on a aussi une autre fonction pour le random pour l'affichage du zombie sur une ligne au hasard encore avec un switch.

Le problème rencontré en faisant les waves était au niveau du compteur entre deux waves qui continuait pendant la pause, le problème a été réglé en créant une constante qui prend le temps qu'il restait et qui lui attribue constamment ce temps.

On a ajouté les barre de vie pour qu'on puisse bien voir quand les ennemis ou les tourelles perdent des pv avec un drawBarreVie dans TourelleGestion et EnnemiGestion pour qu'elle s'affiche en suivant les zombie lorsqu'ils avancent et au dessus des tourelles.



← A nos débuts, nous n'avions qu'un système qui faisaient avancer en boucle les zombies

C - Les tourelles

Ensuite nous avons fait un fichier de base pour toutes les futures tourelles qui hérite de entité et qui permet de tout préparer ce que les tourelles auront toutes en communs comme pour les ennemis, on a fait le premier fichier pour la première tourelle avec des valeurs par défaut, dont son image, ses points de vie, ses dégâts qu'elle inflige. avec ceci un autre fichier Projectile qui envoie des projectile sur les ennemis par les tourelles qui auront des attaques à distance, les projectiles avancent petit à petit jusqu'à la hitbox de l'ennemi et lui inflige les dégats lié à la tour qui l'a envoyé car cette classe sera appelé par une tour d'attaque à distance avec ses dégâts lié.

On a ajouté le reste des tourelles ce qui nous en fait 9 au total, 7 attaquent de loin avec un cooldown plus ou moins rapide mais aussi la vitesse de ce qu'elles envoient est aussi plus ou moins rapide mais aussi les dégâts des attaques plus ou moins fortes. Chaque tourelle a son propre fichier pour redéfinir chacun de ces détails. Les deux autres tours sont différentes, une est, à l'instar de la plante soleil de plant versus zombie, juste génératrice d'argent. C'est à dire qu'elle n'attaque pas mais produit de l'argent avec un cooldown donné, la dernière est une tourelle en corps à corps qui n'attaque que si l'ennemi est proche. Aussi une des 7 tourelle d'attaque de loin n'attaque que si un ennemi est présent sur sa ligne alors que les 6 autres attaquent en permanence.

D - Les boutons

Ensuite pour pouvoir poser des tourelle il faut des boutons, et on pose les tours sur les tuiles de la map donc dans le package tuile on a créé un fichier Bouton.java pour gerer les tuiles cliquable et lorsque la souris est dessus alors la couleur de la tuile change. Pour faire en sorte de savoir quelle tuile est cliquable on rajoute dans GestionTuile un setZoneCliquable qui vient de Tuile.java à true.

Du coup dans TourelleGestion lorsqu'une case qui est cliquable est donc sollicité si on peut y mettre une tour alors la map change pour afficher les tourelles qui sont mettable, avec un switch pour récupérer le bouton qui a été sélectionné, on vérifie s'il la case est vide ou non, si elle est vide on pose bien la tourelle sinon la tourelle posé est enlevée. Pour l'instant on a pas encore implémenter le système d'argent donc on peut poser autant de tours que l'on veut.

E - L'argent

Il a fallu après ajouté le système d'argent déjà des variable étaient prévu pour les zombie avec l'argent qu'ils drop une fois mort qui dépend du type de zombie, on a aussi la tourelle qui génère de l'argent au lieu d'attaquer avec un cooldown. l'affichage de l'argent en haut à gauche de l'écran du jeu avec au lancement 100 d'argent de base. Il a fallu donner un valeur au tourelle et l'afficher à l'endroit où on clique pour ajouter les tourelles pour savoir leur prix, dans le switch de GestionTourelle lorsqu'on en pose une on enlève donc le prix de la tourelle choisie.

F - Le menu

Ensuite il a fallu faire le menu quand on lance le jeu, on fait donc des variables des différents states possible : en jeu, en pause, quand on a gagné ou perdu, le menu de base et les autres sous menu différents, avec tout ça on fait un switch dans paintComponent par rapport à ce gameState que l'on sait grâce à la souris et la fonction cliqueSurUneCase de la classe Bouton qui a des if selon le gameState actuel et des switch pour les menu de pause, gagnant et perdant avec les bouton qui ramène a des endroit différent. mais aussi des switch pour les sous menu pour choisir la difficulté ou le terrain. Dans TuileGestion on a les fonctions qui dessinent les différents menu, et toujours changement map qui se charge de mettre la map qui a été sélectionnée.

G - Les regrouper dans des packages

Pour l'instant on a donc le package Entité avec les sous packages ennemi qui contient les fichiers des zombies différent et le fichier qui les gère : qui les dessine, les update pendant les wave et qui change les probabilité des différentes wave et des waves. Avec des switch on a pu faire selon le mode de jeu qui sera choisi des wave pré faite avec les zombies de plus en plus dur selon le nombre de la wave en cours.

Dans le sous package tour il y a comme pour les zombies un fichier par tourelle, on fera trois niveaux de difficulté donc on a fait des fonctions qui selon le niveau de difficulté choisi le prix des tourelle augmentera, leur attaque diminuera, leur cooldown et leur vitesse aussi. Enfin dans TourelleGestion la fonction cliqueSurUneCase pose la tourelle en fonction de celle qui a été choisie si la case est bien vide. la fonction update qui fait attaquer les tourelles comme décrit déjà avant. maintenant quand on clique sur une tourelle posée on affiche les stats de celle-ci et on peut améliorer ses pv, son attaque, son cooldown ou encore revendre la tourelle pour vider la case, c'est aussi ici que l'on a implémenter le système d'argent donc on débute une partie avec 100 mais on a rajouté dans update le fait que toutes les 15 secondes on a 10 d'argent en plus passivement. c'est tout pour l'instant dans le package entite.

Le package main n'a pas changé avec ses trois fichiers Main ou on lance le jeu, le GamePanel qui gère le jeu en cours de jeu donc avec les update et ce qu'on dessine par rapport au state du jeu en cours. ControleSouris n'a pas changé non plus. On a aussi ajouté une classe Sound dans le main pour mettre du son dans le jeu.

Dans le package ressource nous avons maintenant plus de map pour trois maps différentes qui changent un peu selon le niveau de difficulté toujours en format txt qui sont traité dans TuileGestion avec les png qui sont dans ressource tuile qui leur sont associé. On a aussi tous les zombies différents ainsi que les différentes tourelles, un curseur et on a ajouté les écrans titres du début du jeu et du menu.

Le dernier package qui s'appelle tuile contient 3 fichiers : Bouton, Tuile et TuileGestion. Tuile étant le fichier de base pour les tuiles qui servent pour l'affichage. TuileGestion a toujours la fonction qui relie l'image png d'une map a un chiffre qui est dans le fichier txt de la map certaines sont plantable d'autres non d'autres sont les tourelles pour mettre dans la barre de sélection dans le jeu et on a aussi ajouté les boutons de retour, rejouer, menu et quitter. changementMap ne change pas. la fonction draw affiche les différents menu de sélection, dès le début du jeu avec jouer, continuer, options, quitter mais aussi le menu des difficultés, des différentes maps, des stages, des différents modes : histoire, marathon, stages, et bien sur le bouton retour a chaque fois sauf le premier où on a quitter a la place. on a aussi une fonction qui dessine dans le cas ou c'est une map, une pour dessiner spécialement celles qui sont cliquable donc drawBouton. on aussi une fonction par chaque finalité : donc une fonction drawDefaiteScreen avec les trois différents boutons : rejouer, menu ou quitter, une fonction drawVictoryScreen avec les trois même boutons, drawPauseScreen avec les boutons : retour au jeu, rejouer et menu, drawMenuScreen avec le fond et les boutons et le texte dans chacun, d'autres variables comme pour pouvoir bien mettre le texte au centre ou verifPrice qui pour update quand l'argent s'actualise on peut cliquer que si on a assez d'argent.
Le fichier Bouton permet d'avoir les fonctions cliqueSurUneCase qui fait changer la map avec une barre pour soit ajouter une tourelle soit voir ses stats et diriger une fois les boutons cliqués avec des switch pour changer le gameState ou les maps selon les choix du joueur.

Nos limites :

A - Nos tentatives

A un moment, nous avons eu un problème d'affichage entre mac et windows, l'un de nous a implémenté l'option de mettre la fenêtre en plein écran mais lorsqu'on a essayer le code sur un mac l'écran était noir avec juste le curseur qui s'affichait, après plusieurs recherche dans le code c'était les appel des fonctions qui servait à mettre en plein écran qui faisait bug l'affichage sur mac donc nous sommes revenu en arrière sur le plein écran pour juste fixer une taille de la fenêtre précise, avec pour idée de faire en sorte de choisir la taille de l'écran plus tard dans les dernière choses à implémenter.

Nous n'avons pas pu finir le mode histoire complètement, nous avons seulement un début d'histoire mais sans gameplay derrière. Tout comme le menu option qui pour l'instant ne permet uniquement de monter ou baisser le son de la musique. Un tutoriel a été ajouté pour que la soutenance se déroule avec et plus simplement, avec des explications sur comment jouer mais nous avions vu les choses plus grandes au départ..

B - Ce qu'on voulait rajouter

Nous n'avons pas pu implémenter les choses suivantes :

-réglage paramètre : on souhaitait faire des options pour changer par exemple la taille de l'écran.
-sauvegarde avec serializable
-un mode histoire avec des niveaux spéciaux
-une sorte d'évolution pour les tourelles (pour l'instant elles ne peuvent qu'améliorer leurs statistiques mais leur façon de jouer ne change pas réellement)

Conclusion

Mais malgré nos erreurs, nous sommes tout de même fiers du résultat et ce même si on prévoyait quelque chose de plus grand. Sur le cahier des charges, nous avons pu réaliser toutes les charges minimales et même avancées.

représentation graphique du modèle de classe :