Projet programmation 2

Yohann D'ANELLO

The Game

Présentation globale

The Game est un jeu de Tower Defense. Sur une grille pré-définie, des mobs apparaissent aléatoirement sur la carte en quantité aléatoire. Leur objectif : rejoindre la partie gauche de la carte.

Une partie se déroule en 4 manches. Dès qu'une manche se termine, une nouvelle commence.

Un mob contient diverses propriétés :

- Un sprite (une image de taille 16x16 représentant le mob)
- Un nombre de points de vie
- Une vitesse
- Un butin.

Le joueur a 5 points de vie. À chaque fois qu'un mob atteint la partie gauche de l'écran, celui-ci perd un point de vie. Il perd lorsqu'il n'a plus de points de vie. Il gagne s'il tue tous les mobs à l'issue des 4 manches.

Pour faire face aux mobs, l'utilisateur peut acheter et placer des tours. Une tour a un prix, et le joueur ne peut l'acheter que s'il a la somme requise.

Une tour contient également diverses propriétés :

- Des dégâts par tir
- Une vitesse de tir
- Un prix
- Une fonction permettant de récupérer les mobs sur lesquels tirer

Le jeu fonctionne par tick. Toutes les 50 millisecondes a lieu un tick. Toutes les tours sont mises à jour, puis les mobs se déplacent éventuellement. La vitesse des tours et des mobs influe sur le fait de faire quelque chose pendant le tick ou attendre un suivant (une vitesse de 10 indique qu'une action est faite tous les 10 ticks).

Si une tour doit tirer pendant un tick, elle tire sur tous les mobs à portée, et blesse les mobs en fonction des dégâts réalisés. Si le mob n'a plus de point de

vie, il disparaît.

Si un mob doit se déplacer, un chemin est calculé jusqu'au bord de la fenêtre via un parcours en largeur. Un système de collisions est en effet géré, empêchant 2 mobs ou tours de se trouver au même endroit. Si un tel chemin existe, alors le mob avance d'une case selon ce chemin. Sinon, il reste sur place.

Implémentation

Le projet est intégralement fait en Java. On ne détaillera pas ici la partie éditeur de niveau, bien qu'elle soit conséquente.

Une fois que l'utilisateur a choisi la carte à utilisateur via un menu de sélection, le jeu se lance dans une fenêtre distincte, gérée par la classe `fr.ynerant.levelleditor.game.GameFrame`.

Le coeur du jeu est géré dans un Thread en boucle infinie, une boucle attendant 50 ms avant de passer à l'itération suivante. Le fonctionnement est détaillé plus haut.

L'affichage du jeu est quant à lui géré dans un `JPanel` dédié, avec la fonction `paintComponent` réécrite. Dès que le panel est paint, on demande à ce qu'il soit repaint pour s'assurer que l'affichage est correct.

Dans l'ordre, on dessine les sprites de la carte (couche 1, couche 2 puis couche 3), puis les différents mobs et les tours via leurs sprites.

La carte est composée d'une longueur, d'une largeur et de la liste des cases.

Une case est une position et 3 sprites (couche 1, couche 2, couche 3).

Un sprite est une image de taille 16x16, qui contient des informations sur l'endroit où le chercher.

Un Mob est une classe abstraite contenant des informations abstraites (détaillées plus haut). Un type de mob sera donc une classe héritant de `Mob`, telles que `Mob1`, `Mob2` et `MobCancer`.

Il en est de même pour les tours, avec `BasicTower`, `NullTower` et `AutoTower`.

L'intérêt de l'héritage par rapport à un type donné à une classe Mob (paramètres donnés dans une enumération `MobType` par exemple) est de pouvoir mieux personnaliser les fonctions, par exemple en imaginant des dégâts aléatoires.