Tower defense en C++

de Arthur Finkelstein

I-Introduction :

Le sujet est de faire un tower defense qui utilise principalement les règles de plant vs zombie en C++, le tout en utilisant un maximum la programmation orienté objet et en gérant bien la mémoire (ce qui est la spécificité de C++ comparé à Java).

II-Lancement du projet et règle :

Le projet a été fait en utilisant des gettimeofday() et time() qui sont des fonctions qui ne marche que sous linux, j'utilise comme compilateur clang mais il marche avec g++.
Aucun changement n'as été fait sur le makefile, donc la commande pour lancer le projet est toujours « make clean ; make linux ; make run ».

Le jeu lance des vagues d'astéroïdes qui commence sur la ligne du bas, avant de prendre de plus en plus de ligne (une de plus par vague). Nous avons 5 tourelles offensives, 4 qui font des dégâts purs et une qui ralentit les astéroïdes, et 1 tourelle de soutien qui nous fait gagner 4 argent par seconde.
Les tourelles ne peuvent être poser que pendant les phases de pause entre deux vagues et on peut lancer la vague en appuyant sur le bouton Launch.
Chaque vaisseau a 5 points de vie et quand un astéroïde le touche il perd un point de vie, de plus quand un astéroïde touche le bord de l'écran a gauche, le joueur perd un point de vie et à zéro point de vie c'est le game over.

III-Le code :

1) L'échiquier :

L'échiquier est la classe qui fait apparaître le damier à l'écran, il va permettre de délimiter les espaces où on peut placer les tourelles.
Il contient un tableau pseudo-dimensionnel, c'est à dire un tableau unidimensionnel qui est utilisé comme un tableau bidimensionnel, l'utilisation d'un tableau pour stocker un élément dans une case me paraissait plus simple a mettre en œuvre qu'une classe Case, qui n'aurait eu comme attributs que sa place dans l'échiquier et l'objet qu'elle contient, ce qu'un tableau fait à merveille.

2) Les tourelles :

Les tourelles vont se poser sur une case de l'échiquier et vont tirer à intervalle régulier, chaque

tourelle a une liste chainée qui contient ses tirs. L'utilisation d'une liste doublement chainée me paraissait évident vu que les tirs d'une même tourelle ont tous la même vitesse et donc se suive dans la liste sans jamais se dépasser, qu'on ne connaît pas le nombre de tir que la liste devra contenir à l'avance et que le seul élément nous intéressant est le dernier afin de gérer les collisions avec les astéroïdes.
De plus le fait que chaque tourelle possède sa propre liste de ses tirs a été mon choix car comme sa l'astéroïde regardera toutes les tourelles sur la même ligne que lui et si le dernier tir d'une tourelle le touche alors il prendra des dégâts. L'ordre ici dans lequel on parcourt les tourelles a peu d'importance car si un tir d'une tourelle passe avant l'autre cela se jouera au pixel près et donc sera très difficilement discernable par un humain.

3) Le bandeau :

Il affiche les informations courantes sur le jeu, ainsi que des détails sur les tourelles, celle que nous avons sélectionné, les dégâts, vitesse d'attaque, …

4) Les astéroïdes

Les astéroïdes sont mis dans une liste doublement chainée et on a un tableau de liste chainée (autant que de lignes dans l'chéquier), car encore une fois seul le dernier nous intéresse et on ne connaît pas à l'avance le nombre d'astéroïde que la liste pourra contenir.

5) L'environnement :

L'environnement est une classe qui contiendra l'échiquier, le bandeau et les astéroïdes. J'ai fait ce choix pour appeler dans les engines uniquement une unique fonction act ou draw et transférer tout le code dans la classe Environnement.
L'environnement est le centre névralgique du code car c'est lui qui demande à tous les éléments de se dessiner ou d'agir, le fait de dessiner tous les éléments est trivial mais pour les faire agir entre eux cela devient plus compliqué.
Il faut ainsi qu'un astéroïde plus rapide puisse échanger sa place dans la liste chainée avec un autre astéroïde plus lent car on regarde uniquement l'élément de fin quand on fait les collisions.

6) Lancement des vagues :

Les vagues se lance avec un simple appel a la fonction launch qui va augmenter les stat des astéroïdes de 10-30 % (vitesse, points de vie, nombres), ainsi le niveau du jeu pourra augmenter indéfiniment tant que le joueur arrivera a survivre.

IV-Conclusion :

En utilisant mon programme, on remarque rapidement que le jeu passe de détruire les astéroïdes avec les tourelles à protéger nos points de vie en utilisant les tourelles comme protection car leur points de vie devient trop important pour pouvoir les détruire avant qu'ils ne touchent une tourelle.