C++
Projet 2016

Master Informatique IFI-RIF-MBDS 1ère année
Université de Nice-Sophia Antipolis

1 Tower defense

Le but d'un jeu de tower defense est de placer des **tours** qui tirent sur des **monstres** en mouvement sur un **chemin**. Si un monstre arrive à finir son chemin, le joueur perd une vie. Si le joueur perd toutes ses vies il perd.

1.1 1 tour, 1 monstre, 1 chemin fixe

Dans un damier 12-12, un chemin qui part de (0,4) -> (5,4) -> (5,7) -> (12,7) sera votre chemin fixe initial. Dans toutes les **autres** case du tableau le joueur doit pouvoir placer ses tours. Au lancement (touche 'g' ou boutton sur l'interface) une vague de monstres est lancé.
L'IA d'une tour de base tire simplement sur la cible la plus proche, avec une certaine force (perte de points de vie du monstre) et une certaine fréquence (nombre de tirs pendant un déplacement du monstre) et jusqu'à une certaine distance.
L'IA d'un monstre lui donne sa vitesse de déplacement et son nombre de points de vie.

1.2 Plus de tours

Vous devez créer plusieurs types de tours avec au moins les caractéristiques suivantes:

- Des tirs plus fréquent mais moins puissant.
- Des tirs qui ralentissent l'ennemi.
- Des tirs qui ricochent sur les deux ennemis les plus près avec à chaque fois moins de puissance.

Vous êtes bien sûr invité à essayer d'en rajouter à votre guise (Bonus à la clef?).

1.2.1 Niveau des tours

Les tours doivent avoir un niveau qui changera leur puissance. Ce niveau sera affiché sur leur case et changera leur couleur en légèrement plus foncé. Pour faire monter le niveau d'une tour, l'utilisateur doit effectuer un clic sur la tour puis avoir un affichage de ses statistiques. Dans ce menu, un bouton Niveau+ doit être disponible et afficher son prix.

1.3 Plus de monstres

Vous devez rajouter plusieurs types de monstre dans votre projet. Les monstres seront au minimum:

- Plus résistant mais plus lent.
- Plus faible mais plus rapide.
- Explose lorsqu'il meurt et blesse ses camarades.

Encore une fois, vous êtes invité à en faire d'autres.

1.3.1 Niveau des monstres

Les Monstres aussi doivent avoir un niveau, qui definira leur resistance aux attaques. Leur niveau influera aussi sur leur couleur.

1.4 Plus de chemins

Vous devez être capable de faire avancer les monstres sur n'importe quel chemin. Vous devez proposer au moins 4 chemins au joueur.

1.4.1 Chemin joueur

Le joueur doit pouvoir definir son chemin.

1.5 IHM et jouabilité

Les graphismes ne doivent pas être compliqués.
Nous ne demandons pas plus que des triangles de couleur (une par type) pour les monstres, des petits carrés pour les tours et un chemin d'une autre couleur que les autres cases. Les tirs doivent eux aussi être visible (boule qui bouge, simple trait...).
De plus, l'utilisateur doit pouvoir:

- Choisir la tour qu'il veut placer.
- Choisir la case ou il veut placer la tour.
- Lancer la vague de monstre.
- **Sauvegarder** sa partie
- Utiliser un fichier de configuration qui definit:
  - la taille du plateau (nombre de case)
  - le chemin
  - le nombre de vie
  - l'argent de depart
  - Les vague de monstre (niveau,type, nombre des monstre de chaque vague)

Plusieurs autres critères seront à prendre en compte comme l'argent, la difficulté, on doit pouvoir faire au moins 7-10 vagues sans trop de difficulté mais pas trop facilement non plus!
Bien entendu, on ne peut placer qu'une tour par case.

2 Rendu

Le rendu sera composé de:

- Un archive (zip) contenant l'arborescence suivante:

```
nom_prenom_code/
-----src/
----------/*fichiers .cpp (pas de .o ni de .exe)*/
-----include/
----------/*fichiers .h*/
-----doc/
----------/* fichiers readme + rapport */
-----makefile ou solution Visual Studio
```

Le projet doit fonctionner, soit par la commande make. Soit votre solution visual studio est complement fonctionnelle (un double clic puis clic sur run et le projet fonctionne !). Dans le cas contraire le projet ne sera pas corrigé.

- Le diagramme de classe de votre projet (PDF ou format d'image).
- Un rapport (maximum 4 pages) qui résumera les choix, et expliquera comment utiliser le programme au format PDF.
- Le projet est **individuel**.

Le tout doit être envoyer à l'adresse guillaume.perez06@gmail.com avant le 21 décembre minuit.

3 Library

Si vous avez besoin d'avoir acces au code source de la librairie. Pour compiler une version Debug, ou une version pour votre compilateur, le code se trouve à cette adresse: https://github.com/memo-p/libGraph