Projet Programmation 2
Première partie

Emile Contal & Stefan Schwoon

Le Projet Programmation 2 consiste cette année en l'implémentation en Scala d'un "Tower Defense" avec une interface graphique simple. Vous devrez utiliser avec soin les principes de la programmation orientée objet. Vous travaillerez en trinôme avec éventuellement un ou deux binômes (qui seront évidement évalués en conséquence).

Table des matières

1 Les règles du jeu

La première partie du projet se consacrera au développement du jeu avec des règles simplifiées et son interface graphique utilisant la librairie Swing. Le jeu se déroule sur un plateau où des monstres avanceront et où le joueur placera des tours pour les en empêcher. Initialement le plateau est vide et le joueur possède un nombre de vies fixé et une quantité de monnaie fixée.

1.1 Déroulement des rounds

Le jeu progresse par round. Entre deux rounds, le joueur peut acheter et placer de nouvelles tours sur le plateau. Le joueur appuie sur un bouton pour commencer le round. Lors d'un round, des monstres apparaissent d'un côté du plateau et se déplacent jusqu'à un point vital de l'autre côté du plateau. Lorsqu'ils sont à portée, les tours tirent sur les monstres pour les détruire. Chaque monstre qui atteint le point vital fait perdre une vie au joueur puis se détruit. Le joueur ne peut qu'observer pendant le déroulement du round. Le round se termine lorsque tout les monstres sont détruits. Le joueur a perdu s'il n'a plus de vie, et il a gagné lorsqu'il a survécu à chaque rounds.

1.2 Les monstres

Les monstres peuvent être de plusieurs types qui déterminent les attributs suivants :
- points de vie,
- lenteur (unité de temps nécessaire pour avancer d'une unité de distance),
- butin que gagne le joueur en le détruisant.

Pour la première partie du projet, il vous est demandé d'implémenter au moins deux types de monstres différents. Veuillez utiliser efficacement les concepts de la programmation objet pour factoriser votre code.

Il n'y a pas de collision entre les monstres, c'est à dire qu'ils peuvent se traverser sans se bloquer le passage. Pour simplifier la première partie, les monstres se déplaceront en ligne droite en partant du côté gauche du plateau vers le côté droit.

Vous fixerez la liste des monstres apparaissant au début des rounds en précisant pour chaque monstre à quel moment il apparaît. Prévoyez au moins 4 rounds de difficulté croissante.

1.3 Les tours

Les tours peuvent être de plusieurs types, qui déterminent les attributs suivants :
- dégâts par tir,
- période des tirs (unité de temps nécessaire entre deux tirs),
- portée (unité de distance),
- prix.

De même, il vous est demandé au moins deux types de tours pour la première partie. Pour simplifier, les tours peuvent être placées n'importe où sauf sur la ligne droite empruntée par les monstres.

1.4 Suggestions de discrétisation

Le plateau de jeu peut être modélisé par une grille rectangulaire (ex : 20×9 pour commencer) où les cases peuvent :

- être vides,
- contenir une tour,
- contenir un ou plusieurs monstres.

L'unité de distance est alors un nombre de cases. Vous pouvez considérer que les tours ne tirent que sur un monstre à la fois, même lorsque plusieurs monstres sont sur la même case. Cela donnera lieu à des amélioration dans la suite du projet, où certaines tours pourront tirer sur plusieurs monstres à la fois.

Pendant un round, une solution pour gérer le temps est de le discrétiser en "ticks" qui ne durent qu'une fraction de seconde (ex : 100ms pour commencer). L'unité de temps est alors un nombre de ticks. À chaque tick, les tours qui peuvent tirer le font, puis les monstres qui peuvent se déplacer le font.

1.5 Interface graphique

Pour la première partie il n'est pas demandé d'avoir un design graphique particulièrement esthétique, mais les interactions avec l'utilisateur doivent être intuitives. Les messages au joueur seront affichés par Swing et non pas dans la console. Les objets sur la grille pourront être représentés par des images, voir la Section 5.3 de l'introduction à Scala. Lorsque plusieurs monstres sont présents sur une même case, vous pouvez choisir de n'en visualiser qu'un seul ou éventuellement d'afficher un petit compteur sur la case.

2 Critères d'évaluation

2.1 Rapport et soutenance

Vous devrez rendre un rapport de 2 à 3 pages (en pdf généré par latex) expliquant les choix d'implémentation que vous aurez rencontrés. Dans le cas où votre programme présenterait des défauts, il faudra les mentionner ici en précisant leurs raisons. Une soutenance d'une dizaine de minutes par trinôme sera organisée à la fin de la première partie du projet où vous nous ferez une démonstration de votre programme.

2.2 Fonctionnalités du code

Bien évidemment, votre projet sera évalué par ses fonctionnalités. S'il remplit tout ce qui est demandé, rajouter d'autres fonctionnalités pourra apporter un bonus.

2.3 Organisation du code

Votre projet devra impérativement être organisé hiérarchiquement, en le séparant en fichiers, classes et méthodes. Vous tâcherez de séparer au mieux possible les différentes fonctionnalités en classes, typiquement les aspects "interface graphique" des aspects "règles du jeu". Gardez sous le coude la règle de ne jamais avoir de fonction trop longue ou de fichier trop grand.

2.4 Qualité du code

L'utilisation adéquate de la programmation objet et de Scala sera un critère important dans l'évaluation. Dupliquer du code dans des classes sous-entendrait une mauvaise compréhension de l'héritage. De même préférez des directives fonctionnelles concises à des boucles `for` et `if` imbriquées, voir la Section 4.4 de l'introduction à Scala. La mise en forme, la présence de commentaire et la cohérence des noms de classes, méthodes et variables devront être suffisamment décentes pour une lecture agréable du code.

3 Dates importantes

- Le code et le rapport seront à rendre avant le mardi 1 mars à 23h59.
- La soutenance pour la première partie sera le vendredi 4 mars.