Projet Programmation 2
Deuxième partie

Stefan Schwoon, Mathieu Hilaire

28 février 2020

Le sujet de la deuxième partie du Projet Programmation 2 est de diversifier les comportements des objets, les tours et les monstres, en utilisant efficacement les concepts de la Programmation Objet, et également d'implémenter un déplacement des monstres sur l'intégralité du plateau.

Table des matières

1 Diverses améliorations

1.1 Déplacement des Monstres et Plus Court Chemins

Dans cette partie du projet, on supprime la contrainte qui forçait les monstres à se déplacer en ligne droite, et les tours à ne pas être placées sur ce chemin. Les tours peuvent maintenant être placées arbitrairement sur le plateau, et les tours bloquent les monstres, mais il est impossible de placer une tour si elle bloque tout les chemins menant au point vital. Il faudra donc développer une stratégie de recherche de chemin dynamique pour les monstres, tels que l'algorithme de Dijkstra ou A*. Vous choisirez si les monstres calculent leur chemin de sorte à éviter les tours ou non.

1.2 Les Tours

Votre programme devra comporter au moins 6 types de tours différents (pas forcémenttoutes accessibles au joueur dès le début d'une partie). Grâce à une bonne implémentation, rajouter un type de tour ne devrait quasiment pas modifier votre code plus que dans la(les) classe(s) qui définit ce nouveau type de tour. Typiquement, vous éviterez d'utiliser des listes de tests pour déterminer le type d'une tour. Quelques uns de ces types de tours devront produire des comportements différents, qui ne se définissent pas simplement par un attribut. Voici quelques suggestions de comportements intéressants pour le joueur :

- des tours qui ralentissent temporairement les monstres au lieu de faire des dégâts,
- des tours qui font des dégâts sur tout les monstres proches de leur cible,
- des tours qui font des dégâts sur tout les monstres situés en ligne droite partant de la tour (le projectile traverse les monstres en ligne droite),
- des tours qui améliorent les tours voisines.

1.3 Les Monstres

De même, votre programme devra comporter au moins 6 types de monstres différents, dont certains devront produire des comportements différents. Voici quelques suggestions de comportements :

- des monstres qui soignent les autres monstres,
- des monstres qui ne calculent pas leur chemin de la même façon que les autres (par exemple en évitant les tours, ou en volant au dessus des tours),
- des monstres qui bloquent certains types de projectiles.

Notez qu'il peut être intéressant de combiner ces comportements, par exemple un monstre qui soigne pourra préférer soigner les autres plutôt que d'avancer vers le point vital.

1.4 Davantage d'améliorations

Il est également suggéré de diversifier les types de terrains du plateau, comme des cases *eau* où on ne peut pas construire de tour et où seuls certains monstres peuvent avancer. Pour aller plus loin, vous pouvez implémenter des stratégies élaborées pour les monstres, en modifiant potentiellement les règles du jeu. Par exemple les monstres pourraient détruire les tours si elles bloquent tout les chemins.

2 Critères d'évaluation

2.1 Rapport et soutenance

Les mêmes critères s'appliquent pour cette partie. Vous devrez rendre un rapport de 2 à 3 pages (en pdf généré par latex) qui détaille vos choix d'im-

plémentations et les problèmes et difficultés que vous avez rencontrés. Dans le cas où certaines difficultés n'auraient pas pu être surmontées et que votre programme présenterait des défauts, vous pourrez expliquer ici ce que vous avez essayé d'entreprendre pour les résoudre et pourquoi cela n'a pas marché. Une soutenance de 15 minutes par groupe sera organisée à la fin de la deuxième partie du projet où vous nous ferez une démonstration de votre programme.

2.2 Fonctionnalités du code

Votre projet sera evalué sur ses fonctionalités. S'il remplit tout ce qui est demandé, rajouter d'autres fonctionnalités pourra apporter un bonus.

2.3 Organisation du code

Les mêmes consignes qu'en partie 1 s'appliquent. 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. Gardez sous le coude la règle de ne jamais avoir de fonction trop longue ou de fichier trop grand.

2.4 Qualité du code

De même, 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. 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 31 mars à 23h59.
- La soutenance pour la deuxième partie sera le vendredi 3 avril.