Projet Programmation 2
Troisième Partie

Louis Lemonnier
Stefan Schwoon

1 Introduction

La troisième partie du projet consistera principalement à développer **deux** améliorations majeures – ou une seule, si elle demande énormément de travail – de votre programme faisant appel à des notions plus avancées de programmation. Comme vous ne suivez pas toustes les mêmes cours de programmation, et que vos projets ont tous pris des tournures un peu différentes, vous êtes *relativement* libres du choix de l'amélioration que vous allez effectuer. **Vous trouverez, dans la suite, plusieurs propositions d'améliorations possibles. Si vous souhaitez implémenter autre chose, sollicitez nous pour que nous validions son intérêt pédagogique.** De façon générale, ces améliorations sont gourmandes en calculs plus intensifs : il devient vite nécessaire d'utiliser plusieurs threads en parallèle pour rendre le jeu fluide.

1.1 Intelligence artificielle

Vous pouvez implémenter de la manière de votre choix, des adversaires plus intelligents, qui s'adaptent lors des combats. Vous pouvez également proposer différents niveaux d'adversaires. Il serait plaisant de pouvoir tester les différents niveaux d'aversaires en lançant des affrontement IA vs. IA, que vous pouvez ajouter au jeu de la manière que vous souhaitez.

1.2 Sauvegarde et niveaux

Vous pouvez implémenter une possibilité de sauvegarde au fil de la partie, lae joueur·se pourrait reprendre à l'endroit où iel se trouvait au moment de la sauvegarde, ou choisir de commencer une nouvelle partie.

L'ajout de sauvegarde irait à ravir avec un système de niveaux ; autrement dit, les monstres présents dans le jeu gagnent de l'exprérience au fil des combats, et peuvent augmenter de niveau. Le passage d'un niveau à l'autre augmente les caractéristiques du monstre, lui permet d'apprendre une nouvelle capacité, etc.

On peut ajouter à cela des évolutions : un monstre qui atteint un certain niveau devient un autre monstre (qui peut alors posséder des types différents ou "améliorés").

1.3 Éditeur de carte et d'histoire

Afin d'obtenir un jeu assurément complet et sans limite, on pourrait imaginer que lae joueur·se ait le droit de créer son propre univers pour le jouer.

1.4 Multijoueur

On peut imaginer la possibilité de faire des combats multijoueur, sur une même instance du programme par exemple. Il est également possible d'implémenter cela en réseau : par exemple, en lançant le code sur deux machines de la salle, lae joueur·se pourrait indiquer avec quelle machine iel voudrait lancer le combat.

Si vous préférez implémenter autre chose, sollicitez nous pour que nous validions son intérêt pédagogique.

2 Critères d'évaluation

2.1 Rapport et soutenance

Vous devez rendre un rapport de 2 à 3 pages (dans un pdf), qui détaille vos choix d'implémentations, ainsi que les problèmes et difficultés que vous avez rencontrées. Dans le cas où certaines difficultés n'auraient pas pu être surmontées et que votre programme présenterait des défauts, vous pouvez 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 troisième partie du projet où vous nous ferez une démonstration de votre programme.

2.2 Fonctionnalité du code

Votre projet sera évalué sur ses fonctionnalités. S'il remplit tout ce qui est demandé, rajouter d'autres fonctionnalités pourra apporter un bonus. La qualité graphique peut jouer un rôle, mais l'évaluation de l'interface graphique reposera avant tout sur son caractère intuitif et facile à prendre en main.

2.3 Organisation du code

Votre projet devra être organisé de façon hiérarchique, et il vous faudra le séparer en fichiers, classes et méthodes. Il vous est recommandé de séparer le plus possible les différentes fonctionnalités, notamment les aspects "interface graphique" et "fonctionnement du jeu".

2.4 Qualité du code

L'évaluation prendra en compte la qualité de votre usage de la programmation orientée objet ainsi que la qualité de votre utilisation du langage Scala. Nous évaluerons notamment votre utilisation de la hiérarchisation des objets. Utilisez plutôt des directives fonctionnelles que des boucles imbriquées. 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

- Deadline pour le rendu du code du projet : 18 mai 2022
- Date de la soutenance de la troisième partie : 20 mai 2022