Projet Programmation 2
Troisième Partie

Amélie Ledein
Mathieu Hilaire
Stefan Schwoon

1 Introduction

La troisième partie du projet consistera principalement à développer **une** amélioration majeure de votre programme faisant appel à des notions plus avancées de programmation. Comme vous ne suivez pas tous 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, quatre 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 Proposition A : Fichiers configuration et parseur

Une première extension possible est de faire en sorte de pouvoir définir les salles à l'aide d'un fichier externe, dont la syntaxe permette d'ajouter un élément aléatoire au jeu, que ce soit en terme d'agencement des salles ou des objets. Pour définir la succession de salles ou niveaux, ainsi que l'apparition des monstres à chaque ouverture de porte ou passage à un nouveau niveau, vous pouvez écrire un fichier texte et parser ce fichier en Scala. Dans ce cas, vous utiliserez les combinateurs de parseurs de Scala qui permettent de gérer simplement des grammaires LL(*) ou bien LR arbitraires. La génération de monstres et de niveaux peut alors être définie avec des règles complexes. Des informations sur les parseurs en Scala sont disponibles ici :

- `https://dzone.com/articles/getting-started-with-scala-parser-combinators`
- `http://www.cs.kuleuven.be/publicaties/rapporten/cw/CW491.pdf`

1.2 Proposition B : Mode deux joueurs et réseau

Une amélioration possible est d'implémenter un mode deux joueurs en utilisant le réseau pour faire communiquer deux ordinateurs différents. Vous pou-

vez utiliser les classes `ServerSocket` et `Socket` pour respectivement recevoir et envoyer des messages via TCP, comme décrit sur cette page : `http://stackoverflow.com/a/6416755/710358`.

1.3 Proposition C : Stratégie collective des monstres

Une autre suggestion est l'implémentation de stratégies collectives des monstres. L'objectif serait alors de mettre en place un système d'intelligence artificielle où les entités ne disposent que d'informations locales mais peuvent communiquer par messages aux autres monstres proches. En établissant un protocole à l'avance, qu'il soit *hardcoded* ou évolutif, il est alors possible d'établir une stratégie collective. **Pour cette amélioration, nous voudrions être sollicités pour valider l'intérêt des diverses stratégies collectives que vous souhaitez implémenter.**

1.4 Proposition D : Diversification supplémentaire

Une dernière suggestion est de diversifier encore une fois le gameplay du jeu, en rajoutant des objets, des actions et des personnages. **Pour cette amélioration également, nous voudrions être solicités pour valider l'intérêt des divers ajouts d'objets que vous souhaitez implémenter.**

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 rencontrés. 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 : 18/05/21
- Date de la soutenance de la troisième partie : 21/05/21