Projet Programmation 2
Seconde Partie

Louis Lemonnier
Stefan Schwoon

February 18, 2022

1 Introduction

La seconde partie du projet consistera principalement à utiliser la programmation objet afin de diversifier l'expérience du jeu que vous avez créée à la partie 1 et d'y rajouter de nouvelles possibilités. Les capacités des monstres seront plus variées, les adversaires seront plus intelligents et également plus variés.

Pour rendre l'expérience de jeu plus complète et plus intuitive, un petit univers sera mis en place, dans lequel on peut se déplacer pour aller affronter les adversaires de notre choix.

Remarque : la plupart des points suivants donnent une description minimale de ce qui est attendu, vous avez la liberté d'implémenter un sur-ensemble du sujet, tant que vous expliquez tout ce que vous faites dans votre rapport.

1.1 Les types

Afin de diversifier les capacités des monstres, il est attendu qu'un système de *types* soit ajouté. Des types très classiques, sont par exemple les suivants :

- feu
- plante
- eau

Avec les règles suivantes :

$$\text{feu} \to \text{plante} \to \text{eau} \to \text{feu}$$

et la sémantique : $[\![a \to b]\!] =$ "b est faible face à a".
Ainsi, on associe à chaque capacité un type, et chaque monstre un type (ou plusieurs, si votre système est plus varié). Une attaque dont le type représente

une faiblesse du monstre attaqué fera davantage de dégâts (par exemple, un monstre de type *plante* subit davantage de dégâts par une attaque de type *feu*). Le jeu devra informer le joueur si une capacité a fait plus de dégâts par cette raison.
Lors du choix de capacités pour combattre, les types de ces dernières sont affichés.

1.2 Intelligence des adversaires

Il est attendu que les adversaires fassent des décisions meilleures qu'un choix aléatoire. On pourrait par exemple distinguer des adversaires qui ont une stratégie offensive et ceux qui ont une stratégie défensive. Un plus grand nombre de stratégies sera valorisé, mais n'est pas nécessaire.
On pourrait envisager également que les adversaires soient au courant des types des différentes capacités et des différents monstres pour faire des choix plus avisés concernant leur choix de capacités.

1.3 Bibliothèque de monstres

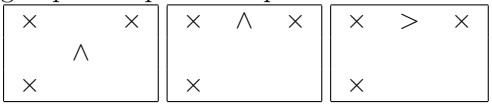
Sachant que votre jeu contient à présent des types, lae joueur·se aura une meilleure expérience de jeu s'iel a la possibilité d'accéder aux informations sur les monstres que l'on peut rencontrer. La bibliothèque serait accessible via un menu, en-dehors des combats.

Un contournement possible de cette section pourrait être d'afficher le type de chaque monstre durant le combat, mais ceci est moins souhaitable.

1.4 Un petit univers

Dans l'objectif de rendre le jeu plus vivant, on peut imaginer que l'équipe de monstres alliée se déplace dans un petit univers (cela peut-être simplement de taille 3×3 pour commencer). L'univers contiendrait également des adversaires, vers lesquels on peut se tourner pour les affronter.

On imagine ici que les $\times$ sont des groupes d'adversaires et $<$ le groupe allié, dirigé vers la gauche. On peut aller lancer un combat avec le groupe ennemi qui se trouve en $(2, 0)$.
Le groupe allié peut se déplacer :



Et on peut alors lancer un affrontement contre le groupe en $(0, 2)$.

Un contournement possible de cette section serait d'avoir un menu avec une liste d'adversaires possibles à affronter et un choix à faire, mais ceci est moins souhaitable.

1.5 Objets

Il est attendu l'ajout suivant : possibilité d'utiliser un objet (par exemple, une potion de soin) à la place d'une capacité lors du combat.

2 Critères d'évaluation

2.1 Rapport et soutenance

Vous devrez rendre un rapport de 2 à 3 pages (dans un pdf) et qui détaille vos choix d'implémentations 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 cette 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 : 29 mars 2022
- Date de la soutenance de la seconde partie : 1er avril 2022