Projet Programmation 2
Première Partie

Amélie Ledein
Mathieu Hilaire
Stefan Schwoon

January 28, 2021

1 Introduction

Le Projet Programmation 2 consiste cette année en l'implémentation en Scala d'un jeu *roguelike* avec une interface graphique simple (voir le programme *castle.scala* sur la page du cours pour se donner une idée de ce à quoi cela peut ressembler). Dans un roguelike, le joueur controle un personnage qui évolue dans un environnement en vue de dessus, avec des éléments aléatoires, et intéragit avec des éléments présents dans ce dernier. Le jeu demande généralement du joueur d'explorer l'environement, de récupérer des objets dans ce dernier, de combattre des monstres ou bien de remplir des quêtes en intéragissant avec des personnages non joueurs. Les objets ramassés se retrouvent dans l'inventaire du joueur, généralement limité, et l'utilisation du bon objet au bon moment revêt souvent un certain aspect tactique.

Le but du projet est d'utiliser les principes de la programmation orientée objet afin de gérer au mieux les différentes composantes du jeu. Vous travaillerez en binômes avec éventuellement un trinôme, envers lequel nos exigeances seront bien sûr plus élevées.

2 Le jeu à programmer

La première partie du projet consiste à programmer les composantes principales du jeu, l'interface graphique sera gérée par la librairie graphique de votre choix (par défaut, nous vous proposons d'utiliser Swing, toutefois il ne s'agit pas d'une contrainte). Le jeu se déroule dans un environement en vue de dessus rempli d'objets ainsi que de personnages, et où le joueur controlera un personnage capable d'interagir avec lesdits objets ainsi que les autres personnages. Les éléments du jeu comme des objets et des monstres peuvent avoir des positions prédéfinies au début du jeu, ou bien générées aléatoirement.

2.1 Actions du joueur

Le jeu doit comporter un personnage controlé par le joueur. Il est nécessaire que ce dernier puisse effectuer des actions de déplacements (quand le joueur utilise les touches directionnelles, le personnage se déplace dans les directions correspondantes), ramasser des objets, et également que le joueur puisse faire réaliser au personnage différentes actions (au moins deux actions différentes en plus des déplacements et du ramassage) lui permettant d'interagir avec son environnement et les objets et monstres présents (par exemple, manger des objets, les équipper ou encore les jeter).

2.2 Les objets et les monstres

Le personnage du joueur ne se déplace pas dans un paysage vide ! Ce dernier est rempli d'objets et/ou de monstres. Les objets peuvent être de plusieurs types (armes, armures, cadavres, colliers, bagues, gemmes, potions, livres, nourriture, parchemins, pièges, statues, décorations, arbres, et caetera), tout comme les autres personnages présents sur la grille, qui peuvent être des monstres ou des marchands (par exemple). Le projet reste libre sur les aspects créatifs du jeu, et vous pouvez coder un jeu où un aventurier parcourt un donjon en battant des monstres et récupérant des trésors tout comme un jeu où un patissier doit récupérer des morçeaux de recettes en intéragissant avec des marchands puis ramasser des ingrédients et les combiner pour créer un gateau. Il est néanmoins demandé, pour la première partie du projet, d'implémenter au moins trois types d'entités (autres personnages, objets) différents. Ici, l'utilisation de la programmation objet est importante, et il est demandé que les différents types d'objets et de personnages aient des comportements et des actions possibles distinctes.

On remarquera ici que la programmation objet est particulièrement adaptée à gérer ce genre d'actions et d'objets : plusieurs classes d'objets (potions, parchemins, armes...) peuvent être diversifiées (potion magique, potion d'eau, sirop pour la toux...). Les actions peuvent alors utiliser cette hiérarchie, par exemple, faire en sorte que tout objet puisse être lancé (donc code commun qui décide où l'objet va aller), que toute potion se casse sur un impact, éventuellement avec des effets selon le type de potion.

Pour simplifier la première partie, les autres personnages peuvent être gérés par de l'aléatoire, et il n'est pas demandé de leurs donner la capacité de se déplaçer.

2.3 Suggestions de discrétisation

Il est recommandé de modéliser le jeu par une grille rectangulaire (ex : 20×9 pour commencer) où les cases peuvent :

- être vides,
- contenir un personnage ou un monstre,
- contenir un ou plusieurs objets.

L'unité de distance est alors un nombre de cases.

La solution recommandée pour gérer le temps est de faire un *tour par tour* où chaque action du joueur prendrait un certain nombre de tours, mais où aucun temps ne s'écoule entre les actions du joueur. Une solution plus avancée consiste à discrétiser le temps en *ticks* qui ne durent qu'une fraction de seconde (ex : 100ms pour commencer). L'unité de temps est alors un nombre de ticks. Toutefois, le but n'est pas de coder un jeu d'action, mais de mettre l'accent sur les interactions et la diversification des objets et personnages.

2.4 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 l'interface graphique et non pas dans la console. Les objets sur la grille pourront être représentés par des images, ou même des lettres (voir la Section 5.3 de l'introduction à Scala). Lorsque plusieurs monstres/objets sont présents sur une même case, vous pouvez choisir de n'en visualiser qu'un seul.

3 Évaluation

3.1 Rapport et soutenance

Vous devrez rendre un rapport de 2 à 3 pages et qui détaille vos choix techniques 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 première partie du projet où vous nous ferez une démonstration de votre programme.

3.2 Fonctionalité 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. La qualité graphique peut jouer un role, mais ce cours est avant tout un cours de programmation objet, ainsi, la perspective principale se portera sur les fonctionalités, et l'évaluation de l'interface graphique reposera avant tout sur son caractère intuitif et facile à prendre en main.

3.3 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 fonctionalités, notament les aspects *interface* et *fonctionement du jeu*.

3.4 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. Faites attention à bien utiliser les propriétés d'héritages et à ne pas dupliquer du code inutilement. 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.

4 Dates Importantes

- Deadline pour le rendu du code du projet : 2 mars 2021
- Date de la soutenance de la première partie : 5 mars 2021