Projet Programmation 2
Seconde Partie

Amélie Ledein
Mathieu Hilaire
Stefan Schwoon

March 12, 2021

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. Dans ce but, vous allez mettre en place de multiples catégories d'objets ayant chacune leurs sous-catégories et comportements distincts. Nous attendons ainsi une augmentation du nombre d'objets avec lesquels le personnage du joueur peut interagir, mais aussi une diversification des autres personnages et des modes d'actions proposés au joueur.

À l'aide d'une mise en place de divers niveaux ou salles, vous exploiterez ensuite cette diversité afin de générer un gameplay varié où les objectifs du joueur peuvent changer d'un niveau à l'autre.

1.1 Diversification

Le premier objectif de cette partie consiste à diversifier les objets, les personnages et les moyens disponibles pour interagir avec ces objets et personnages. Ainsi, pour cette partie, nous demandons que vous rajoutiez au moins quatre nouveaux types d'objets que le personnage du joueur puisse ramasser et stocker dans un inventaire affiché à l'écran. Leurs natures et effets exacts restent restreints uniquement par votre imagination, mais il est important que vous vous serviez ici de la programmation orientée objet. Ainsi nous souhaiterions que vous mettiez en place certaines catégories d'objets autant que des objets en eux-même, c'est-à-dire faire en sorte que certains de vos objets aient des éléments et des méthodes en commun, mais qui seraient en partie spécialisés dans leur implémentation. Vous pourrez créer de nouveaux objets héritant de classes et traits déjà mis en place, de même que vous pourrez créer de nouvelles classes et traits avec lesquels créer plusieurs objets avec des implémentations de méthodes différentes.

Ainsi nous suggérons tout de même des exemples de catégories que vous pourriez décider de mettre en place :

- Des Consommables (des pommes, des bananes, des chèques de caution, des tartes à la crème, du poisson, des pansements ...) utilisables une seule fois et modifiant certaines statistiques comme la vie ou le score, de manière permanente ou temporaire.
- Des Armes (des épées, des épées de feu, des pistolets, des haches, des machettes, de l'anti-moustique, un rouleau à patisserie, un lasso...) aux effets actifs influant sur l'environnement et les personnages, et pouvant par exemple retirer du jeu certains ennemis ou obstacles.
- Des Vêtements (des vestes de feu, des armures, des bottes ailées, des T-shirts hawaïens, des moufles, un chapeau de cowboy ...) aux effets statiques modifiant certains effets circonstanciels ou certaines statistiques de façon constante.

Il est ici demandé d'utiliser le principe de hiérarchie sur les objets (et avoir au moins deux types d'objets qui héritent d'une même classe). Une action s'appliquant à la classe dont les autres héritent (ou un objet sur lequel on applique une action dont d'autres actions héritent) devra alors avoir des comportements différents suivant les différents types héritiers. Par exemple, manger une pomme peut redonner des points de vie là où manger une banane donnerait non seulement des points de vie mais créerait aussi un nouvel objet peau de banane, le lancé d'un boomerang aurait un effet différent de celui réalisé par le lancé d'un marteau, et un anneau de pouvoir pourra avoir un effet différent si on l'enfile sur son index ou sur son auriculaire.

Nous demandons également que vous rajoutiez trois nouveaux types de personnages. Leurs natures et leurs effets exacts eux aussi ne sont restreints que par votre imagination, mais nous suggérons tout de même des exemples de catégories que vous pourriez décider de mettre en place :

- Des Personnages non-hostiles (comme des marchands, des médecins, des oracles, des prêtres ...) avec lesquels le joueur pourrait interagir pour récupérer des bonus.
- Des Personnages agressifs (comme des gobelins, des robots tueurs, des dragons, des rats garous, des champignons vénéneux...), qui poursuivraient le joueur ou du moins essaieraient de l'agresser afin de l'empêcher d'atteindre ses objectifs.
- Des Personnages fuyards (comme des voleurs, des licornes magiques, des écoliers turbulents, des vaches, ...), qui fuiraient le joueur du mieux qu'ils peuvent tandis que les rattraper lui permettrait d'atteindre ses objectifs.

Ici aussi, il est demandé d'utiliser le principe de hiérarchie sur les objets (et avoir deux types de personnages qui héritent d'une même classe). Ainsi, demander service à un autre personnage aura des effets différents suivant qu'il s'agisse d'un marchand qui vendrait des objets ou d'un médecin qui soignerait des dégâts, les attaques des personnages hostiles seraient différentes suivant le type de ces derniers, certains pouvant attaquer au corps-à-corps, d'autres à distance. De plus, rattrapper un voleur permettrait de récupérer un objet que ce dernier aurait volé, là où rattrapper une licorne pourrait par exemple permettre de gagner des bénédictions sur ses objets.

Jusqu'ici seulement deux actions en plus des actions de déplacement étaient requises. Afin de pleinement pouvoir profiter d'une diversité d'objets et de personnages, nous demandons que vous rajoutiez par rapport à la partie 1 au moins deux nouvelles actions possibles pour le personnage du joueur. Il peut s'agir d'actions de base comme "utiliser l'objet", "parler au personnage" ou "lancer l'objet" mais aussi d'actions un peu plus complexes comme "donner un objet à un personnage" (qu'il soit ami ou ennemi) ou bien "écrire un message sur le sol en se servant d'un objet" ou encore "frotter l'objet A contre l'objet B". Comme dit plus haut, nous souhaitons que vous profitiez de la programmation orientée objet dans le but de diversifier les comportements des différentes actions en prenant en compte les objets qui sont utilisés mais aussi les entités qui vont réagir à ces utilisations.

1.2 Niveaux et objectifs du joueur

Le second objectif de cette partie consiste en la mise en place d'objectifs et de progression narrative dans le jeu. Jusqu'à présent en effet, il n'était pas demandé d'avoir d'objectifs concrets dans le jeu. Il est désormais exigé de mettre en place différents niveaux, environements ou salles entre lesquels le jouer peut progresser (que ce soit de façon linéaire ou non). Ces salles peuvent être préconçues ou générées aléatoirement. Nous vous demandons aussi de mettre en place des **défis ou objectifs** dont l'accomplissement permettra au joueur d'avancer vers d'autres salles et éventuellement de gagner le jeu. Ces objectifs peuvent être de nature variée, et vous devrez implémenter au moins deux types de salles distinguées par des objectifs différents. Il s'agirait, dans un souci de variation de gameplay, d'éviter d'avoir une série de salles ayant toutes pour objectif de terrasser tout les monstres de la salle, typiquement. Ces objectifs peuvent p.ex. consister en :

- des portes nécessitant des clefs.
- des portes nécessitant d'avoir battu tous les ennemis d'une pièce.
- des portes nécessitant un certain score pour passer.
- des portes nécessitant la résolution d'énigmes pour y accéder.

L'objectif étant ici de profiter de la diversité demandée pour générer un gameplay qui évite d'être trop répétitif, et qui fait varier les expériences de jeu.

1.3 Mise en place d'un système de sauvegarde (optionnel)

En dernier lieu, nous voudrions optionnellement que vous mettiez en place un **système de sauvegarde** qui permette de conserver une progression lors d'une session du jeu puis de la reprendre, ou bien la prochaine fois qu'on lance le jeu, on reprend à la dernière sauvegarde, là où l'on c'était arrêté, ou bien la prochaine fois qu'on lance le jeu, un menu de sélection propose de "charger une partie" ou "créer une nouvelle partie".

En guise de simplification, la sauvegarde peut être partielle (et juste retenir le niveau, par exemple, et pas les positions exactes des acteurs dans le niveau). Alternativement, la sauvegarde pourrait être interne à la session de jeu en cours, durant laquelle il serait ainsi possible de revenir en arrière.

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 : 06/04/19
- Date de la soutenance de la seconde partie : 09/04/19