Projet Programmation 2

Troisième Partie

YOAN GERAN STEFAN SCHWOON

25 avril 2024

1 Introduction

La troisième partie du projet a pour but de mettre à l'oeuvre les principes de l'orienté objet et de se rapprocher encore plus d'un jeu. Pour cela, nous vous proposons donc d'implémenter des fonctionnalités comme la sauvegarde, le jeu en multijoueur ou contre l'ordinateur, ou encore la gestion de score.

Nous vous demandons de choisir au moins deux parmi les propositions suivantes (réseau, IA, évenements, sauvegarde). Si vous avez un autre projet qui nécessite un trvailcomparable, il est possible de le faire comptabiliser sur *validation préalable*.

2 Exemples de fonctionnalités

2.1 Jeu multijoueur

Avec un jeu qui consiste à transporter des choses d'une ville à un autre, un jeu multijoueur pourrait consister à avoir des compagnies adverses qui font la même chose. D'un point de vue programmatique, un joueur devient donc une classe qui se diversifie en des acteurs contrôlés par le joueur propre et de différents IA. En fonction de votre jeu, ces compagnies pourraient emprunter les même lignes de transport que vous, ou devraient construire les leurs. De plus, l'interaction entre les différentes compagnies pourrait également faire partie du jeu. Cela donne alors lieu à la possibilité de l'implémentation d'IAs ou d'un mode mulitjoueur en réseaux.

2.1.1 Réseaux

La création d'un mode multijoueur en réseaux vous demandera de faire de la programmation réseaux. Dans ce cas, on peut imaginer la possibilité de créer une partie, et celle de rejoindre une partie. Ce mode de fonctionnement demande de bien séparer le frontend (qui accepte les commandes du joeuer et en affiche les résultats) et le backend (qui décide l'état du jeu). Ici il faut gérer la communication asynchrone entre frontend et backend et les fautes potentiels de réseau, il ne faut pas sous-estimer cette tâche.

2.1.2 IA

Un mode IA est également intéressant. De plus, vous pouvez implémenter des IAs aux comportement différents. Par exemple, certaines IAs pourraient vouloir optimiser les trajets, d'autres pourraient préférer se concentrer sur les ressources, etc.

2.2 Évènements

Pour rendre votre jeu moins monotone, un système d'évènements peut être implémenté. Un véhicule qui a un accident, une grève, une ligne de transport qui n'est plus empruntable, etc. Ceci permet l'implémentation d'un système d'évènements. On peut imaginer certains évènements totalement aléatoires, mais également des évènements avec une probabilité d'apparition dépendant des actions du joueur ou d'autres choses (on peut par exemple simuler les saisons et avoir des problèmes plus fréquents en hiver).

2.3 Sauvegarde

La sauvegarde permet de stocker l'état actuel dans un fichier, terminer le programme pour y revenir plus tard. On ne parle notamment pas d'une sauvegarde en mémoire. Cet objectif nécessite de *sérialiser* et *désérialiser* les objets, c'est à dire les convertir en texte et puis reconvertir le texte en objet. Il existe plusieurs bibliothéques à cet effet qui pourront vous aider.

3 Évaluation

3.1 Rapport et soutenance

Vous devez rendre un rapport de 2 à 3 pages et qui détaille vos choix techniques et 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 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 Fonctionnalités du code

Votre projet sera évalué sur ses fonctionnalités. S'il remplit tout ce qui est demandé, rajouter d'autres fonctionnalités pourront apporter un bonus. La qualité graphique peut jouer un rôle, mais ce cours est avant tout un cours de programmation objet, ainsi, la perspective principale se portera sur les fonctionnalités, 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* (*frontend*) et *fonctionement du jeu* (*backend*).

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 : 21 mai, 21h
— Date de la soutenance de la troisième partie : 23 mai