Projet Programmation 2
Troisième Partie

Nicolas MARGULIES Stefan SCHWOON

9 avril 2025

1 Introduction

Pour la troisième partie, le but est de se concentrer sur quelques nouvelles fonctionnalités plus ambitieuses faisant appel à des notions plus avancées de programmation, la boucle de jeu principale étant en place. Comme vos projets ont commencé à diverger, vous êtes relativement libres de choisir quelles fonctionnalités vous souhaitez implanter. Nous vous proposons la démarche suivante :

— Choisissez les améliorations que vous souhaitez réaliser. Vous trouverez des propositions dans la suite, mais vous pouvez proposer autre chose si vous le souhaitez.
— Faites valider votre choix auprès des responsables du cours lors de la prochaine séance. La quantité d'améliorations attendue dépendra du nombre de personnes dans chaque groupe.

2 Suggestions d'améliorations

2.1 Sauvegarde et reprise de partie

Le but de cette amélioration est de mettre en place un système permettant de sauvegarder sa progression dans un fichier pour pouvoir fermer le jeu et reprendre la partie plus tard. Ceci peut également servir de moyen de proposer différentes cartes (au moyen de sauvegardes en début de partie).

2.2 Système de progression

Le but de cette amélioration est d'avoir un système de progression au fur et à mesure de la partie : un bâtiment permettrait de rechercher des technologies permettant de construire de nouvelles machines, ou d'améliorer celles existantes.

2.3 Présence d'ennemis

La présence de vos bâtiments pourraient alerter des formes de vie hostiles présentes sur la planète qui viendraient attaquer vos usines, et il faudrait donc avoir des moyens de se défendre contre ces attaques.

2.4 Passage en coordonnées continues

Certains objets, voire un personnage jouable pourraient ne pas être limités par les cases de la carte, et pouvoir se déplacer librement. Ils devraient quand même avoir des interactions avec les bâtiments (collisions avec les bâtiments trop grands, se faire entraîner par les convoyeurs, ect.)

2.5 Génération procédurale de cartes

La carte de jeu pourrait être générée de façon procédurale, c'est-à-dire aléatoire, mais respectant une certaine cohérence au niveau macro (avoir certaines zones qui sont des gisements d'un minéral, des zones inaccessibles, ect.). Vous pouvez utiliser des fonctions pseudo-aléatoire comme le bruit de Perlin, et essayer d'imposer certaines règles pour rendre la carte plus agréable (par exemple faire en sorte que les cases inaccessibles n'isolent pas complètement une zone).

2.6 Algorithmes de recherche de chemin

Certains éléments de votre jeu peuvent utiliser des algorithmes de plus court chemin pour choicir le chemin à suivre. Un exemple serait le déplacement de monstres, un autre un système de transport de ressources semi-automatique, qui permet de créer un réseau de transport, et de demander à transporter des ressources à n'importe quel point de ce réseau.

2.7 Réseau et jeu multijoueur

Une amélioration ambitieuse est de séparer le jeu entre un client et un serveur, qui communiquent via TCP. Vous pouvez utiliser les classes Socket et ServerSocket du paquet java.net. Une fois ce découplage fait, il devrait être relativement facile de concevoir un mode multijoueur coopératif où plusieurs joueurs se connectent au serveur en même temps et construisent des bâtiments ensemble sur la même carte.

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 Fonctionnalité 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 : mardi 20 mai 18h
— Date de la soutenance de la première partie : jeudi 22 mai 11h15