TRON TRON

Modèles de Conception Réutilisables

Amel Dussier, Ioannis Noukakis & Fabien Salathe

Table des matières

Introduction

Contexte

Ce projet se fait dans le cadre de notre cours « Modèles de conception réutilisables ». Le but de ce cours est de découvrir et de nous familiariser avec des *Design Patterns*. Ce sont des solutions connues répondant à des problèmes récurrents de conception logicielle, et qui permettent de concevoir des applications robustes et modulaires.

Après avoir étudié les différents patrons de conception, de manière théorique et dans des exercices, nous allons maintenant les appliquer dans un projet plus conséquent. Il s'agira d'un travail de groupe, et chaque groupe au sein de la classe aura comme objectif d'utiliser un *design pattern* principal, choisi à l'avance. Evidemment, la taille du projet permettra d'en utiliser d'autres, afin de mettre en œuvre les bonnes pratiques étudiées lors du cours.

En plus du projet de développement et du rapport, chaque groupe fera une présentation orale afin de partager son projet et son expérience avec les autres groupes de la classe.

Objectifs

Notre groupe a choisi de travailler sur le patron de conception « Mediator ». Il s'agit d'un *design pattern* qui permet de simplifier la communication entre objets. Le médiateur est un intermédiaire entre les objets, et ceux-ci n'ont pas besoin de connaître la nature ou le nombre d'objets avec lesquels ils communiquent. Ce patron sera décrit de façon détaillée dans le chapitre « Conception ».

L'exemple classique pour ce patron est une application de messagerie ou un forum de discussion. En effet, un utilisateur n'a pas besoin de connaître chaque personne connectée pour envoyer un message à tout le monde. C'est le serveur (le médiateur) qui s'en charge pour lui, en traitant le message envoyé.

Dans le cadre de notre projet, nous avons décidé de réaliser un jeu de course multijoueur. Il s'agira d'une version simplifiée et revisitée des courses de motos *LightCycle* du film *Tron*. Nous allons mettre en application le modèle de conception « Mediator » pour communiquer par exemple les actions d'un joueur aux autres joueurs de la partie. Cela nous permettra de rendre la communication entre les joueurs plus modulaire et plus simple à gérer.

Analyse

Fonctionnalités

Nous allons commencer par un petit rappel concernant les courses de *LightCycles*. Dans le film *Tron* original, les personnages doivent participer à une course de motos virtuelles, les *LightCycles*. Ces motos évoluent sur un plan en deux dimensions, et ne peuvent faire que des virages à 90 degrés (angle droit). Durant la partie ils ne disposent pas de freins pour ralentir ou s'arrêter, ils ne peuvent qu'avancer et changer de direction. Chaque moto laisse dans son sillage un mur de lumière (« Jetwall » en anglais). Si un concurrent entre en collision avec un mur de lumière, il a perdu la partie. Le dernier joueur dans la partie gagne. Notre jeu va reprendre en grande partie ces principes, et en ajouter quelques nouveaux.

Tout d'abord, les nouveaux arrivants dans la partie vont démarrer dans une zone d'entrée neutre (« lobby »). Dans cette zone, les joueurs sont invincibles et peuvent se familiariser avec les commandes. Lorsqu'ils rencontrent le bord de la carte, ils seront automatiquement repositionnés au centre.

Dans la zone d'entrée se trouve également un portail de téléportation, qui donne accès à la carte principale où a lieu la partie. Sur cette carte, plus grande et d'une couleur différente, les joueurs meurent s'ils rencontrent un bord ou s'ils touchent le mur de lumière laissé par la moto d'un autre joueur. Ils peuvent ensuite recommencer depuis la zone d'entrée.

Nous avons également ajouté des « bonus » sur la carte, permettant par exemple de modifier la vitesse ou la taille des motos, et ainsi donner un avantage temporaire à un joueur. Pour accéder à ces bonus il suffit de rouler dessus et leur effet est immédiat.

L'aspect graphique est important pour que les joueurs prennent plaisir à utiliser notre jeu. Nous allons faire au mieux pour rendre notre application visuellement attrayante, mais nous allons mettre l'effort principal sur l'implémentation du patron de conception « Mediator » et la gestion de la communication entre les joueurs. Il s'agit après tout de la raison d'être de ce projet.

Architecture globale

Un schéma qui montre simplement le server et un ou deux clients

Choix des technologies

Java

Librairie Slick

Justifier pourquoi, ou dire si c'était imposé

Conception

Le patron Mediator

Présentation

C’est quoi un patron de conception? C’est qui le Mediator ? A quoi il sert ?

Diagramme de classe du GoF

Implémentation dans notre application

Mettre l’accent sur cette partie, c’est la raison du projet !

Diagramme simplifié de nos mediators, avec explication comment est géré la communication, etc.

Protocole de communication

Fonctionnement

Comment notre protocole va fonctionner

Les messages qui seront envoyés

Utilisation des sockets et de la sérialisation

Diagramme de classes

Ci-dessous le diagramme de classes de la partie protocole et des modèles :

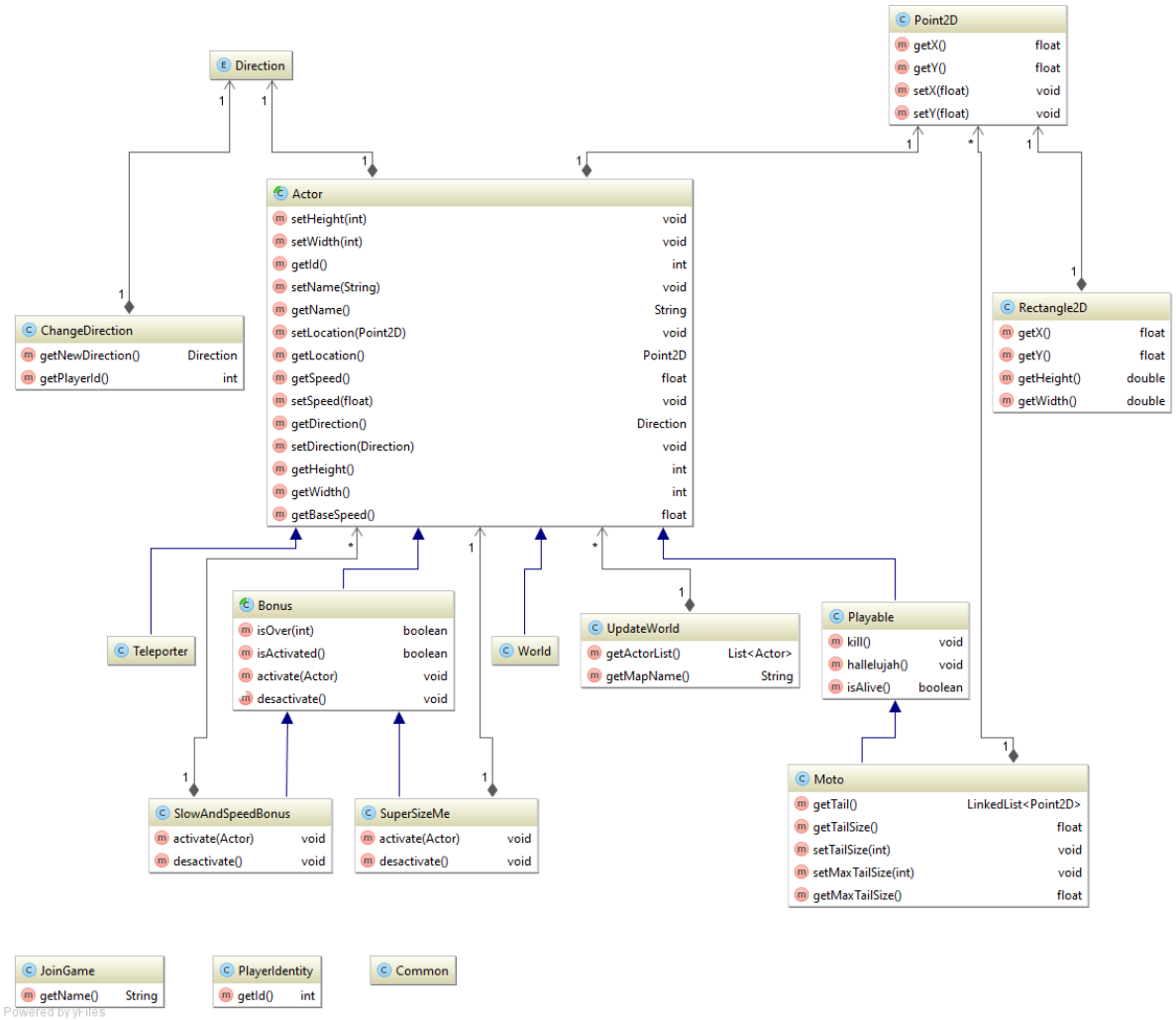

Figure 1 : Diagramme de classes de la partie protocole et modèles

Serveur

Fonctionnement

Explications

Diagramme de classes

Ci-dessous le diagramme de classes de la partie serveur :

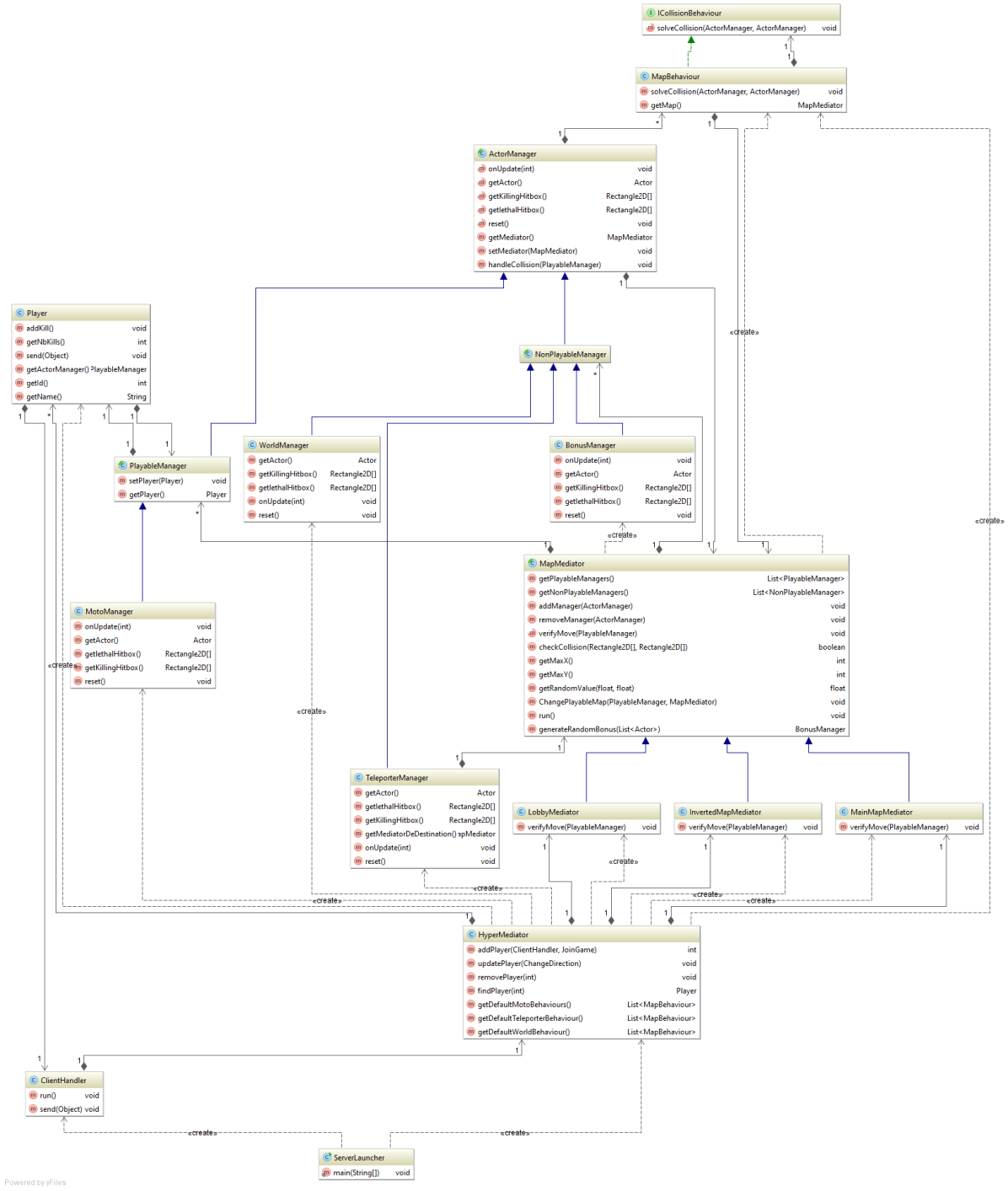

Figure 2 : Diagramme de classes de la partie serveur

Client

Fonctionnement

Explications

Diagramme de classes

Ci-dessous le diagramme de classes de la partie cliente :

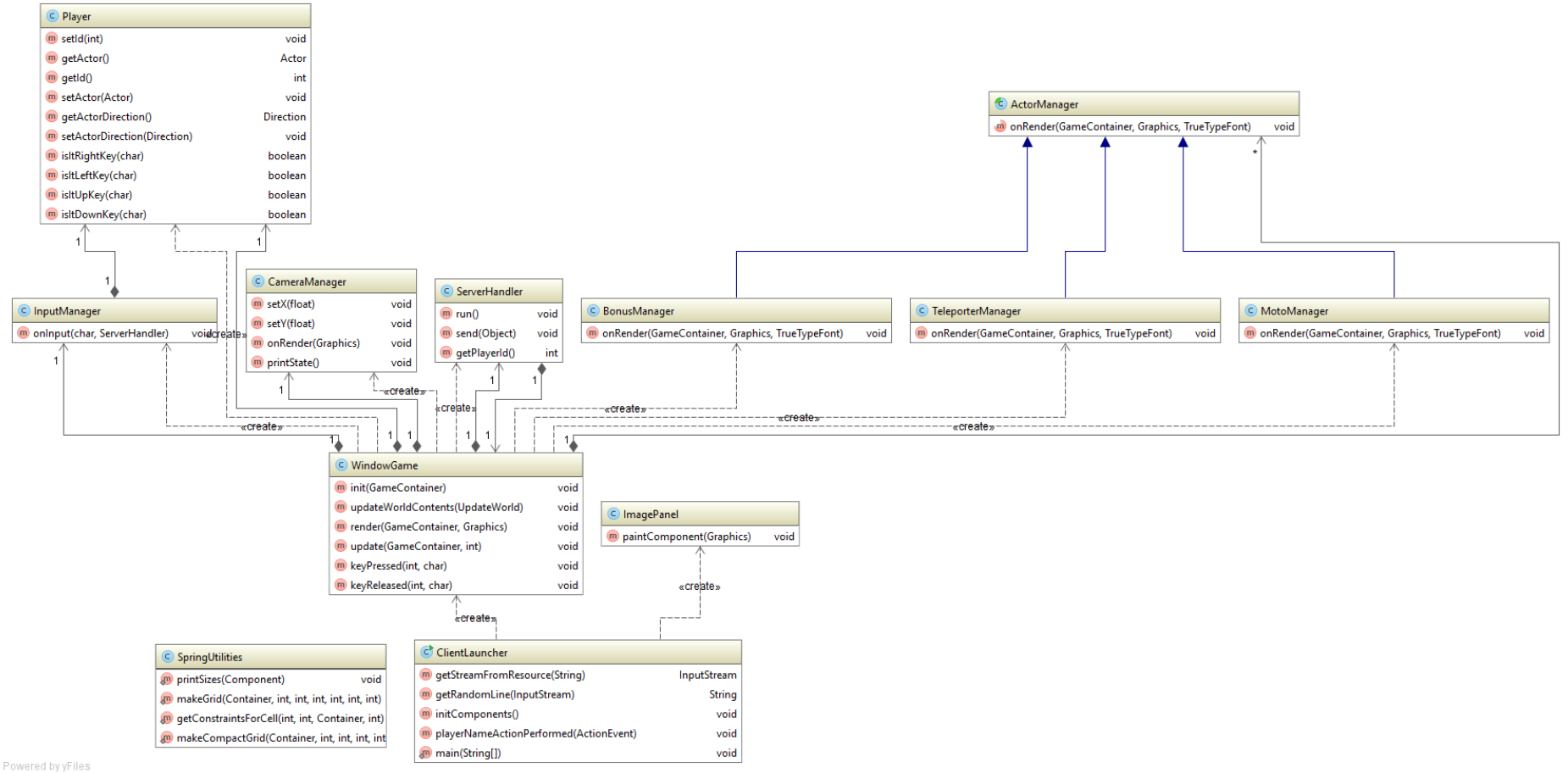

Figure 3 : Diagramme de classes de la partie cliente

Captures d’écran

2-3 captures d’écran sympa avec une description

Gestion du projet

Rôles des participants

Analyste

Développement

Tests

Documentation

Outils communs

IntelliJ

Pour coder, décrire vite fait l'éditeur

Tout le monde utilise le même éditeur pour simplifier les choses

Git

Gestion des sources

Décrire GIT en quelques mots

Décrire notre stratégie, nouvelles branches, commits réguliers, etc.

Stratégie de tests

Comment on a testé nos fonctionnalités les unes après les autres

Décrire quelques tests fonctionnels (changement de carte, collision, etc.)

Planification

Planning prévisionnel

Diagramme de Gantt

On en invente un, le plus simple possible ;)

Planning effectif

Basé ou similaire au journal de travail

Conclusion

Etat du projet à l'échéance

Où on en est

Ce qui n'a pas pu être réalisé

Problèmes rencontrés

Ce qui n'a pas marché dans l'équipe

Ce qu'on aurait pu faire mieux

Améliorations possibles

Fonctionnalités cool à ajouter

Annexes

Table des illustrations

Journal de travail

Liste des semaines

Qui a fait quoi