Université de technologie de Belfort-Montbéliard

LO43 Flotte de véhicule autonomes

Buri Theo Florian Lacour

Belfort, Décembre 2014

Table des matières

Introduction

Dans le cadre de l'UV LO43 " Bases fondamentales de la programmation orientée objet", il nous a été demandé de réaliser un projet de groupe, afin de mettre en pratique les connaissances acquises lors des cours et TDs du semestre.
Trois sujet nous on été présentés. Nous avons fait le choix de traiter le sujet de la "Flotte de véhicules autonomes", et ceci pour plusieurs raisons :

- (A TROUVER)
- N'étant que deux, sur un maximum de quatre étudiants par groupe autorisés, les autres sujets ne nous ont pas parus réalisables en temps et en heures et sans bugs majeurs...
- (A TROUVER)

Nous présenterons tout d'abord le sujet, ses contraintes, et les libertés prises par rapport à celles-ci. Par la suite, nous parlerons des différents diagrammes UML, et les expliquerons. Enfin, nous terminerons par l'implémentation en Java et l'interface graphique.

Première partie

Présentation du sujet

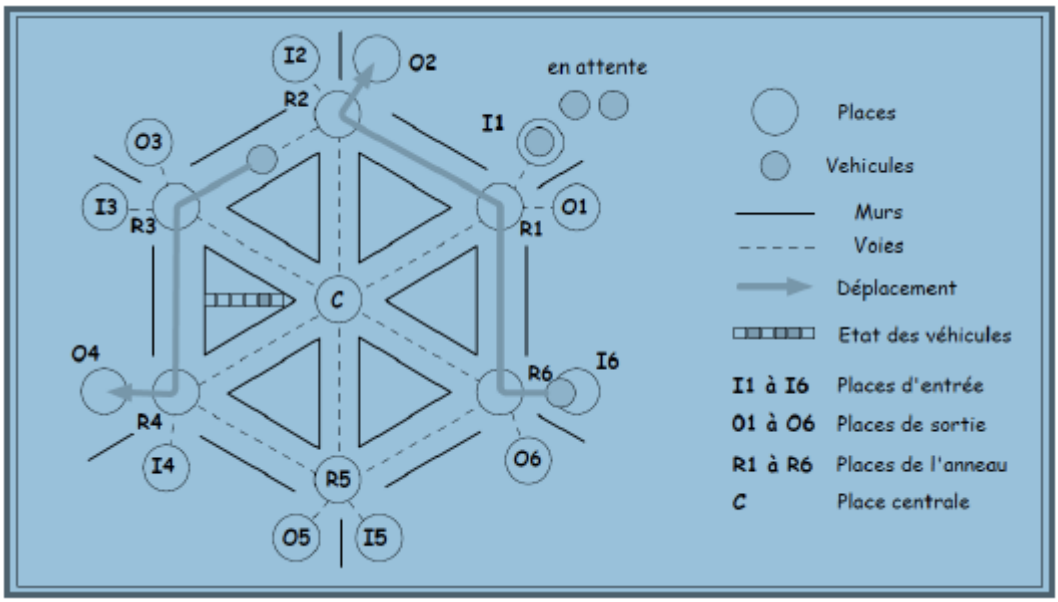
1.2 Objectif

Le programme à réaliser consistait en la modélisation d'une flotte de véhicules évoluant dans une infrastructure de circulation partagée. Pour cela, il a tout d'abord fallu modéliser la partie calculatoire à l'aide du langage UML, puis ensuite l'implémenter en Java et lui donner une interface graphique

1.3 Reformulation du sujet

1.3.1 Plateau

Le plateau donné par le sujet est le suivant :



Chaque place, exceptée celle du centre, est rattachée à une place de départ et une place de sortie. Les voitures disposant d'une mission partent depuis une place d'entrée et, à la fin de celle-ci, rejoingent une place de sortie. La place centrale n'est utilisée que lors d'un trajet reliant deux places opposées...

1.3.2 Missions et contrôleur

Chaque passager doit être transporté d'une place à une autre. Pour ce faire, ils envoient une requête au module qui se charge du contrôle du plateau : le contrôleur. Par la suite, celui-ci assignera cette mission à un des véhicules de sa flotte. Celui-ci calculera ensuite le trajet qu'il suivra pour la mener à bien.
C'est alors le véhicule qui décide de partir : pour cela il doit envoyer une requête au contrôleur pour savoir si son chemin est déjà réservé. Si ce n'est pas le cas, le contrôleur l'autorise, et le véhicule part. La requête se fait sous la forme d'une Request Map, constituée de booléens indiquant l'intention du véhicule de réserver une place ou non.
Pour exemple, lorsqu'une voiture désire aller de la place I1 à O2, elle aura besoin de réserver I1, R1, R2 et O2. La Request Map seras alors la suivante :

I1	I2	I3	I4	I5	I6	R1	R2	R3	R4	R5	R6	O1	O2	O3	O4	O5	O6	C
T	F	F	F	F	F	T	T	F	F	F	F	F	T	F	F	F	F	F

Afin d'éviter toute erreur lors de la réservation, nous avons défini la priorité des places qui suit :

$$I1<I2<I3<I4<I5<I6<R1<R2<R3<R4<R5<R6<O1<O2<O3<O4<O5<O6<C$$

Ceci permettra d'éviter que deux véhicules tentent de réserver la même place à deux moments différents, et que le contrôleur les valide...

1.3.3 Contraintes et libertés prises sur le sujet

S'ajoutent alors quelques contraintes, qui régissent le programme :

- Les véhicules ne peuvent utiliser la place centrale qu'à la seule condition qu'ils doivent se rendre à la place en face de la leur.
- Une place ou une route (reliant deux places) ne peut être utilisée que par un seul et unique véhicule.
- Un véhicule qui souhaite emprunter un chemin occupé doit attendre que celui-ci se libère.

Ainsi, par rapport à ces différentes consignes, nous avons choisi de prendre certaines libertés :

- Un véhicule ne peut pas contourner une place occupée pour mener sa mission à bien.
-

Deuxième partie

Conception UML

Chapitre 2

Les Diagramme UML

2.1 Diagramme des cas d'utilisation (annexe A)

Le diagramme des cas d'utilisation permet de représenté les action que l'utilisateur peut faire. Dans notre projet, en lançant le jeux l'utilisateur a plusieurs solution :

- voir les crédit.
- Choisir de jouer en mode manuel, dans ce cas il devras après choisir le chemin de la voiture
- Jouer en simulation, dans ce mode toute les requête sont générer par le programme
- axé aux option, une fois dans ce menus, il peut choisir de changer les graphique du jeu, il peut aussi changer la rapidité des voitures.

2.2 Diagramme de sequence(annexe B)

Le diagramme de séquence permet de représenté les interaction entre les différentes classes selon un ordre chronologique.

2.3 Diagramme de classes(annexe C)

Nous verrons en détail, une description de chaque classe.

Chapitre 3

Description des classes

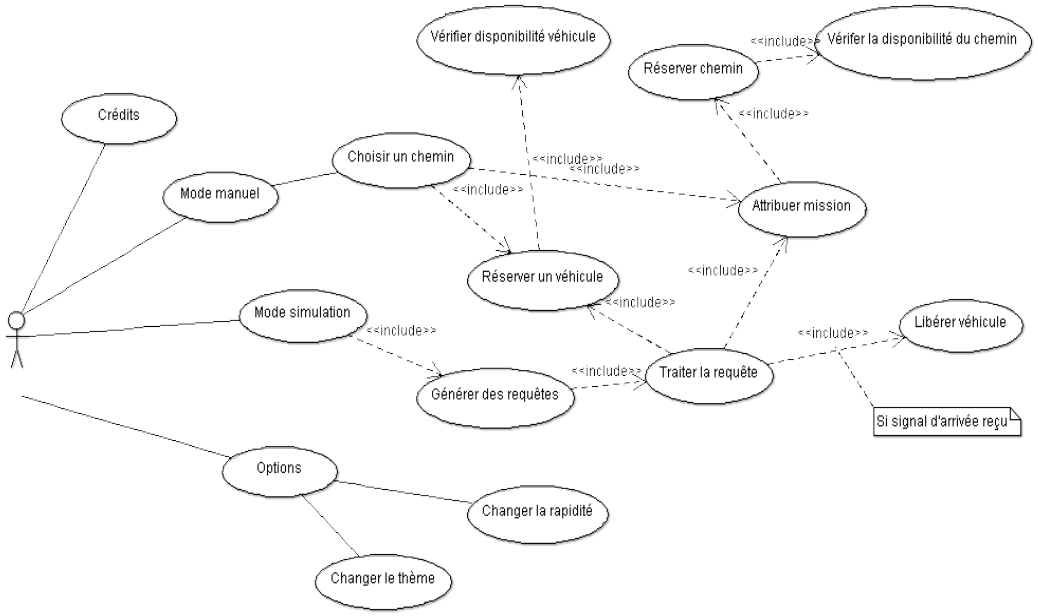
3.1 Classe BoiteAuxLettre

Il s'agit de la classe recevant toute les requêtes, qu'elle soit envoyer de la partie graphique, ou du modèle. Il y a 3 requête :

- Une liste de requête de départ cette liste contient toute les demande de trajet. Ces trajet peuvent être demander soit par l'utilisateur, soit générer par le modèle.
- une liste nommé rMap, ce sont les requête qu'envoies chacun de véhicule pour demander si leurs route sont libre ou non.
- rFin, ce sont les requête qu'envoies touts les véhicules lorsqu'ils ont terminer leurs trajets. Elle permettent de libérer toutes les routes

La méthode getFirst() renvoi une requête celons un ordre de priorité. En effet dans un premier temps elle renverras une requête de fin de trajet afin que les trajet soit libérer le plus vite possible. Puis nous regardons si il y a une requête de Depart, afin que la liste de véhicule désirant partir soit le plus complet possible

Annexe A


Vérifier disponibilité véhicule
Réserver chemin
<<include>>
Vérifer la disponibilité du chemin
Crédits
Choisir un chemin
<<include>>
<<include>>
Mode manuel
<<include>>
Attribuer mission
<<include>>
Réserver un véhicule
<<include>>
<<include>>
Mode simulation
<<include>>
Libérer véhicule
<<include>>
<<include>>
Générer des requêtes
Traiter la requête
Si signal d'arrivée reçu
Options
Changer la rapidité
Changer le thème

Annexe B