Intersection intelligente

Luc Heydel – Guillaume Muller – Aurélien Saunier – Victor Sonza

# Problématique

Le but du projet est de développé une simulation d’intersection intelligente, dictant le passage de voitures à un carrefour. Le développement de l’application a été effectué en SARL, en utilisant JAVA AWT pour l’interface graphique.

Nous utilisons la bibliothèque mathématique Arakhne pour définir la trajectoire des voitures lors de leur passage dans l’intersection.

# Architecture du projet

## Le fichier Car.sarl

Car contient l’agent qui définit le comportement d’une voiture au sein d’une intersection. Il est représenté par son carBody, qui est la représentation physique de l’agent dans la simulation. Le carBody fait partie d’une liste (waitingList or leavingList) d’une roadSection d’une intersection et es taussin contenu dans une hashMap dans l’environnement

## Le fichier Environnement.sarl

## Le fichier EventAndSharedData.sarl

EventAndShareData permet de définir les events utilisé dans le programme ainsi que les classe de type « Body » permettant d’instancier des objets contenant les données liées aux agents. Ces objets sont liés aux agents via leur UUID.

## Le fichier Intersection.sarl

## Le fichier IntersectionCenter.sarl

## Le fichier IntersectionSimulationLauncher.sarl

Ce fichier est l’agent que nous utilisons pour lancer la simulation. Il contient notamment la fonction launchEnvironment, qui lance l’environnement (lancement du noyau, lancement du contexte par défaut, …)

## Le fichier Population.sarl

Ce ficher permet le stockage de constant relative à un groupe d’agent. Le but est de centralisé les constant. Une instance correspond à un ensemble d’agent car ayant des propriétés communes sous la forme de constantes ayant les mêmes valeurs

## Le fichier RoadSection.sarl

## Le fichier Settings.sarl

Le fichier settings permet de définir des constants globales pour le programme. Le fichier comprend des constantes tel que la taille de l’environnement, le nombre d’intersection le temps de pause, etc…

# Affichage

L’affichage est géré par la classe EnvironmentGui. Celle-ci est inspirée de la classe créée par M. Nicolas Gaud dans son exemple de simulation de Boids, et adaptée à nos besoins. Cette classe peint les bâtiments, les lignes des routes, ainsi que les voitures (représentées sous formes de triangles noirs). Nous disposons d’un timer en haut à gauche de l’interface qui permet de voir le temps moyen passé par une voiture dans l’intersection.

La fonction paint de la GUI est appelée par l’environnement une fois que tous les agents présents dans la simulation ont envoyé leur demande d’action.

La création de la GUI est entièrement codée en brut, les positions sont définies de manières absolues. Il pourrait être intéressant de modifier ce point afin de pouvoir créer un enchaînement d’intersection.

# Evènements

# Points à améliorer

## La création de la GUI

La création de la GUI est codée en dur dans l’application, et est codée en utilisant AWT. Il serait intéressant de rendre la création de l’interface de manière dynamique, pour permettre la création d’enchainement d’intersection.

Qui plus est, il serait aussi intéressant de changer de technologie pour la programmation de l’interface, pourquoi pas en utilisant JavaFX, qui est beaucoup plus récent.

## La gestion de la multi intersection