UNIVERSITÉ DE TECHNOLOGIE DE BELFORT MONTBÉLIARD

Croisement de trains de véhicules

IA54 : Systèmes multi-agents et résolution distribuée de problèmes

BECK Florian - HESTIN Alexis

05/01/2015

Automne 2014

GAUD Nicolas

GECHTER Franck

Sommaire

Introduction

Dans le cadre de l'unité de valeur IA54, nous avons choisi le projet de croisement de trains de véhicules. Pour résoudre ce problème, nous devons utiliser une approche orientée agent. La résolution de ce problème permettrait à des automobilistes ou des trains de véhicules automatisés de franchir un croisement idéalement sans s'arrêter et sans avoir besoin de feux tricolores ou autres priorités.

I. Description et objectifs du sujet

L'objectif de ce projet est de développer un modèle à base d'agents réactifs pour offrir la possibilité de croiser des formations de trains de véhicules sans collisions lors d'une intersection. Deux trains de véhicules sont formés, se déplacent le long d'un parcours et doivent se croiser à une intersection de la façon la plus fluide possible en adaptant leur vitesse.

a. Analyse

La perception de chaque agent véhicule est limitée. La perception de ceux-ci est limitée aux autres véhicules (pas d'autres obstacles/entités). Les véhicules peuvent envoyer des messages à l'environnement pour indiquer leur position à chaque cycle. A chaque cycle, l'environnement envoie aux véhicules la liste de leurs « voisins ».

Ce mixage/croisement doit se faire automatiquement suivant des messages envoyés par les trains aux véhicules, des messages entre véhicules, et la réflexion et l'analyse indépendante de chaque agent.

b. Principales étapes envisagées

Nous avons choisi de développer notre projet en suivant une méthodologie de cycle : analyse des améliorations, conception, implémentation puis tests.

- Le premier cycle consiste à afficher une voiture/agent sur un circuit et de la faire se déplacer dessus
- Le second cycle consiste à créer un train de voitures qui se suivent sans heurts
- Le troisième cycle consiste à gérer plusieurs trains de véhicules, ainsi que leurs croisements

Des cycles intermédiaires ont été mis en place. Ils avaient pour but d'améliorer successivement le comportement des agents vis à vis des objectifs impliqués.

II. Description des agents

Dans notre simulation, nous avons implémenté trois types d'agents ayant des rôles particuliers. La description de chacun de ces agents est fournie ci-dessous.

a. Voiture

Toutes les voitures possèdent des valeurs d'accélération, de décélération et une distance de sécurité qui ont été choisies arbitrairement par nous-même afin d'obtenir le meilleur rendu possible lors de la simulation.

A l'initialisation, chaque voiture s'enregistre dans le groupe du train qui l'a créé et s'ajoute à la position initiale du circuit.

La voiture va adopter un comportement réactif à chaque cycle. En effet, elle vérifiera constamment si des voisins sont présents devant elle ou non. Si des voisins sont devant et proche d'elle, la voiture va ralentir. Sinon elle va accélérer à moins qu'elle ait déjà atteint la vitesse maximale, définie initialement ou modifiée par son train.

De plus, à chaque cycle, elle vérifie si elle a reçu des messages provenant de son train lui indiquant de changer sa vitesse maximale et sa distance de sécurité.

Enfin, à la fin de chaque cycle, la voiture bouge de quelques pixels sur le circuit et envoie sa position à l'environnement.

b. Train

Chaque train appartient à un groupe de la forme : « train0 », « train1 », etc...

A l'initialisation, le train crée les voitures qui appartiennent elles aussi au groupe du train qui les a créés. Tout au cours de la simulation, le train peut envoyer des messages aux voitures pour leur indiquer la vitesse maximale et la distance de sécurité qu'elles doivent adopter en temps réel.

Lorsque deux trains de voitures vont se croiser, le train reçoit des messages provenant de l'environnement. Dans ce cas il traite le message et adapte sa vitesse pour que le croisement se déroule au mieux.

c. Environnement

L'environnement a connaissance de la position et des adresses de toutes les voitures à n'importe quel moment de la simulation. En effet, à chaque cycle, les voitures envoient leurs positions et leur adresse à l'environnement sous forme de message.

A chaque cycle, l'environnement vérifie si un croisement entre deux trains va avoir lieu. Si c'est le cas, il va envoyer un message au train le plus loin du croisement en lui indiquant les coordonnées du croisement et les voitures de l'autre train ne sont pas encore passés.

III. Solution orientée agent

La solution que nous proposons est un modèle à base d'agents réactifs. Les voitures réagissent en fonction de leur environnement proche.

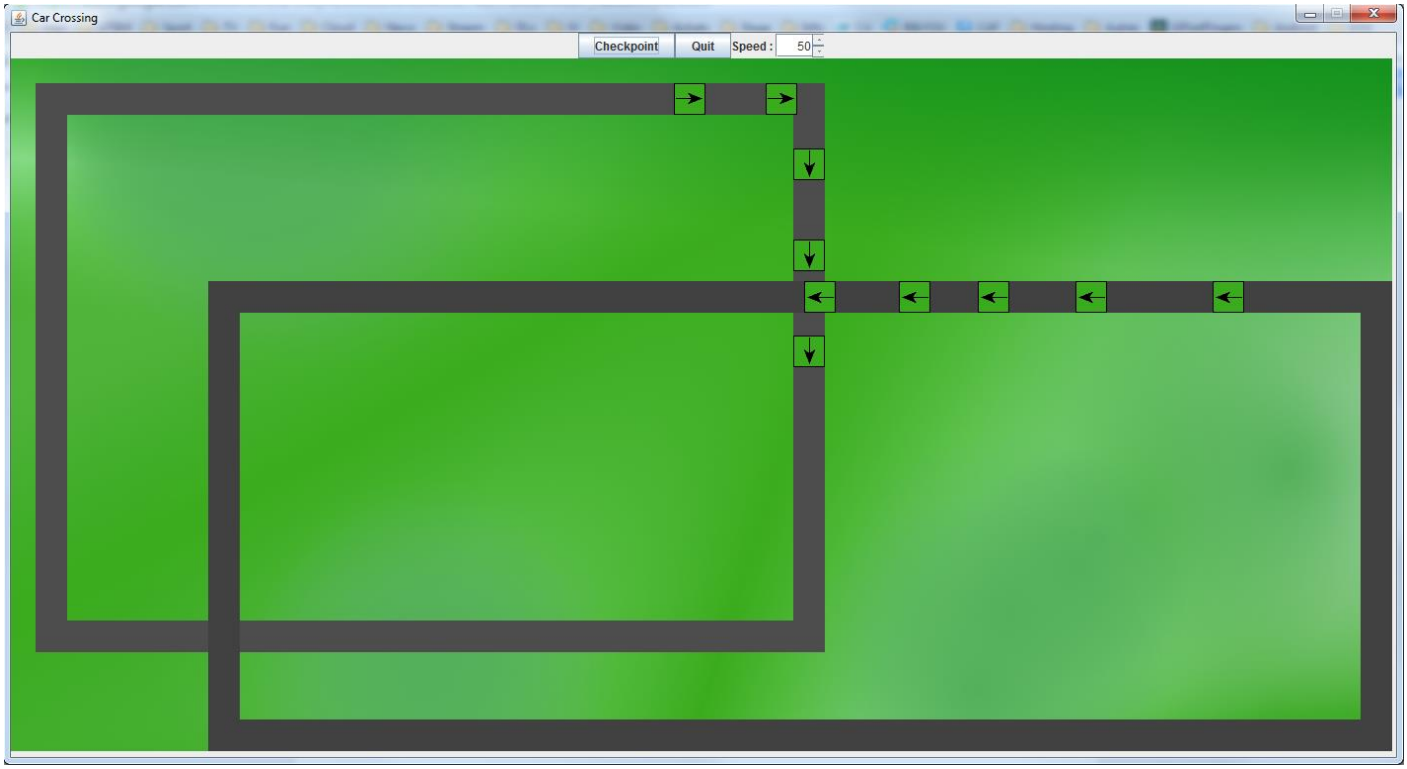
a. MadKit

Nous avons choisi d'utiliser la plate-forme **MadKit** afin de simuler le système multi agents. En tant que débutant dans le domaine des SMA, nous l'avons trouvée légère, bien documentée et facile d'utilisation.

Les points que nous avons trouvés attractifs étaient la gestion automatique d'un ordonnancement des agents (assurance qu'ils aient tous une et une seule exécution par cycle) ainsi que la classification des agents selon un modèle de type communauté/groupe/rôle, rendant ainsi plus simple et plus propre la communication.

b. Interface graphique

L'interface graphique a été développée à l'aide de la bibliothèque graphique **Swing**. Elle est constituée d'un layout superposé : l'arrière-plan contient le fond d'écran/circuit et le premier plan contient les voitures.



c. Création d'un circuit

Nous avons besoin d'une définition du circuit que parcourent les voitures, autant du point de vue visuel que de l'application. Nous avons choisi de représenter un parcours par une série de segments. Un circuit est matérialisé par un fichier XML de la forme :

```
<path background="Circuit_4.png">
          <train>
                    <point>
                        <x>25</x>
                        <y>25</y>
                        <angle>90</angle>
                    </point>

                    <point>
                        <x>443</x>
                        <y>25</y>
                        <angle>180</angle>
                    </point>

                    <point>
                        <x>443</x>
                        <y>643</y>
                        <angle>270</angle>
                    </point>

                    <point>
                           …
                    </point>

             </train>

             <train>
                    …
             </train>
</path>
```

- le background est un lien relatif vers l'image qui représente le circuit;
- la balise "train", qui définit le parcours des voitures d'un train. Le circuit comportera autant de trains de véhicules que de couples de balise "<train></train>;
- les segments du trajet sont définis par une liste de points ordonnés, comportant leur coordonnés ainsi que l'orientation de la portion de circuit les succédant.

Cela permet de définir l'image du circuit ainsi que le parcours de chaque train lors de l'exécution. Le nombre de voitures par train est lui un paramètre définissable dans le fichier Const.java.

IV. Problèmes rencontrés et principales améliorations/solutions

a. Premier cycle

- **La rotation des véhicules**

La représentation des voitures est gérée par un **RotateLabel**. Pour permettre le déplacement des voitures, ainsi que leur rotation, nous avons étendu le **JLabel** en RotateLabel, afin de lui ajouter la notion d'angle lors de l'affichage d'une voiture. L'angle est géré en radians et actualisé au besoin lors de l'établissement de la nouvelle position. La principale difficulté était de redessiner l'image correctement sans modifier ni dégrader les axes de l'image.

- **La gestion d'un trajet**

Il était impossible de se contenter d'écrire un trajet « en dur », nous avons donc développé une classe qui interprète un fichier XML comme décrit plus haut. Elle est chargée d'initialiser la position des voitures ainsi que de leur donner leur prochaine position en fonction de leur trajet (propre à chaque train, à leur position actuelle et à la distance à parcourir).

b. Second cycle

- **La gestion des voisins**

Pour coordonner un train de véhicules, nous avons en premier lieu initié une communication entre la voiture et sa précédente potentielle lors de son initialisation. Cette communication servirait à suivre la position de celle-ci et ainsi de gérer une « distance de sécurité ».

Si la solution peut sembler viable, elle devient lourde lors de changement d'ordre des voitures. De plus, l'objectif étant de faire se croiser des trains de voitures, il se posait le problème de changer de voiture « suivie » fréquemment. Nous avons donc abandonné cette solution pour porter notre choix sur un environnement qui serait informé des positions de tous les véhicules. La responsabilité de l'environnement est d'avertir l'agent voiture de la présence et de la position de tous ses voisins présents dans une zone de visibilité.

- **La prévision des collisions**

En détectant seulement les infractions aux distances de sécurité, nous nous sommes aperçus que les voitures étaient souvent obligés de piler, ce qui était pénalisant pour la fluidité du trafic. Nous avons alors mis en place la gestion de « distance de vue » plus grande que la distance de sécurité qui permet d'adapter sa vitesse à celle de son plus proche prédécesseur.

c. Troisième cycle

➢ Procédure de croisement

La procédure pour faire croiser deux voitures étant différentes de celle qui vise à faire suivre deux voitures sur un même trajet, il a fallu modifier l'algorithme de suivi des « membres de l'environnement » pour en tenir compte. En premier lieu, nous avons souhaité fusionner les détections et réactions des deux cas mais finalement nous avons opté pour un traitement séparé.

Nous avons ainsi atteint le stade actuel de **l'algorithme de vie d'une voiture**, nous allons maintenant le synthétiser pour le rendre plus clair :

- ❖ Vérifier la présence de nouveaux messages (de mon train, ou de véhicules que je vais croiser)
- ❖ Simuler la vitesse optimale (accélérer jusqu'à atteindre la vitesse maximale)
- ❖ Calculer la position après ce pas.
- ❖ Demander à l'environnement de m'informer des voisins présents dans cette nouvelle configuration.
- ❖ Si des voisins sont présents
  - traiter le cas du voisin le plus proche
  - s'il est avéré que c'est la prochaine voiture que nous allons croiser, on applique la priorité définie
  - si en revanche ce n'est pas l'actuel croisement mais qu'il fait partie de la liste des croisements prévus, on considère que les précédents croisements sont terminés et on met ladite liste à jour en appliquant les priorités établies.
  - finalement, si aucune priorité n'est définie, on vérifie si les voitures sont effectivement sur le point de se croiser,
    - si celles sont sur le point de se croiser, la plus proche du croisement obtient la priorité
    - celle qui détecte le croisement prévient l'autre voiture concernée, avec la distribution des priorités.

// La priorité, si elle a lieu d'être, est maintenant définie

- Dans le cas d'un croisement où notre voiture n'aurait pas la priorité, son objectif sera d'adapter sa vitesse pour passer juste après la voiture qu'elle doit laisser passer. On estime le temps restant avant que l'autre voiture passe, donc le temps dont nous disposons pour arriver à une certaine distance de sécurité du croisement. On en déduit sans mal notre vitesse modérée par les coefficients d'accélération et de freinage si besoins.
- Si la situation étudiée n'est pas un croisement, nous déduisons que nous suivons la voiture en question (le voisin le plus proche). 2 cas sont possibles :

- Nous sommes très proches de la voiture, à une distance inférieure à la distance de sécurité. dans ce cas, nous freinons autant que nécessaire.
- Si le voisin est dans notre champ de vision sans être dans le champ de sécurité, nous analysons sa vitesse, pour essayer de l'atteindre. L'objectif exact est d'atteindre sa vitesse au moment où nous serons à la distance de sécurité, ayant ainsi un espacement optimal, sans prendre de risques.

// La vitesse optimale a été révisée en vitesse adaptée à l'environnement

❖ valider le mouvement, avertir l'environnement et actualiser la représentation graphique.

L'algorithme de décision pour chaque cycle de vie de la voiture est notre point de réflexion principal, comme vu nous avons souhaité y placer toutes nos prises de décisions.

➢ **Synchronisation de trains en amont d'un croisement**

L'objectif final étant la fluidité maximale, nous avons souhaité améliorer la préparation du croisement à l'échelle des trains. Originellement, nous avons implémenté le fait qu'une voiture qui entre en croisement avertisse son train, visant à ce que celui-ci modère la vitesse du convoi pour faciliter le passage.

Ensuite, sous les conseils de notre superviseur, nous avons modifié cet algorithme en délégant la détection d'un croisement de trains à l'environnement, qui se charge d'avertir le second train d'une présence d'un train en amont. Le second train se charge de modifier la vitesse de ces voitures pour qu'elles s'intercalent avec celles du premier, de sorte à nécessiter le minimum de freinage et d'adaptations une fois les voitures dans le croisement.

Toutefois, concernant ce point, le système n'est pas au point, en effet par soucis de gain de temps, nous avons simplifié les calculs mathématiques (accélérations et décélération du convoi) et considéré que les intervalles entre voitures ainsi que leurs vitesses respectives étaient les constants au sein de chaque train.

Dans les faits, les voitures ne sont pas toujours réparties uniformément, notamment si le convoi sort d'un croisement. Comme le train effectue le calcul pour faire passer le mieux possible la première voiture, on se retrouve souvent avec les voitures suivantes qui sont mal adaptées, ne profitant donc d'aucun gain de fluidité.

Conclusion

Pour conclure, ce projet nous a fait découvrir le domaine des systèmes multi-agents et certaines problématiques qu'il peut engendrer. Notons particulièrement un certain degré de hasard au niveau de l'ordre d'exécution duquel découle également l'ordre de détection et d'action. En outre, comme les autres projets réalisés, c'est un bon exercice de travail en groupe, de partage des tâches et de coordination.

- **État du projet**

Notre premier objectif, le déplacement de véhicules, bien qu'estimé trivial a nécessité un peu de recherches pour permettre la rotation de ceux-ci. Nos limitations aux portions de routes verticales et horizontales, non handicapantes dans le cas de la simulation, nous ont permis d'alléger la charge de travail.

Notre second objectif consistait principalement à faire suivre des véhicules sans qu'ils ne se heurtent. Cet objectif est également rempli, en effet, après la mise en place de mesures de sécurité et de prévention, nous assistons à une bonne fluidité des agents au sein d'un même train.

Notre troisième objectif et véritable but du projet était de faire croiser deux files de véhicules le mieux possible. Nous avons réussi à faire croiser deux files de véhicules, cependant la fluidité n'est pas optimale. Malgré les prévisions, de la part des trains à l'échelle globale (vitesse) ou de la part des voitures avec les attributions de priorités et anticipations, la fluidité n'est pas optimale.

- **Limites et éventuelles améliorations futures**

Nous pouvons expliquer en partie cette situation par l'écart entre la distance théorique entre les voitures et les mesures réelles. Ces écarts faussent les estimations, et, après le premier couple de voitures, les erreurs rendent les calculs globaux obsolètes. C'est pourquoi nous n'avons pas eu le temps de mettre en place des outils statistiques qui permettraient d'améliorer la coordination des voitures dans le train.

Nous avons envisagés plusieurs points pour des améliorations :

- Un modèle mathématique plus poussé, notamment au niveau de la vitesse des véhicules rendrait la simulation plus utile
- Le traitement de plusieurs voisins à chaque tour améliorerait l'anticipation, surtout au niveau des croisements
- Principalement, améliorer la cohésion au sein d'un train nous permettrait de rendre plus efficace les modifications de comportements globales aux trains