Université de unicaen

Projet

Simulateur urbain

TU KHANH DUY NGUYEN - 21811412
MARTIN MAXENCE - 21807030
MEYER ARTHUR - 21805134

Table des matières

1 Rapport

1.1 Sujet

Le trafic urbain, en particulier la gestion des feux de signalisation et des sens uniques, fournit un cadre d'application très intéressant pour des algorithmes d'intelligence artificielle. L'objectif de ce projet est de réaliser un simulateur d'un tel trafic, en simulant des agents (les automobilistes) munis de politiques d'actions (toujours se déplacer par le chemin le plus court, respecter les feux, passer à l'orange, etc.) et des feux eux-mêmes munis de politiques (par exemple un automate).

Le simulateur doit permettre de mesurer différentes valeurs tel que le nombre de pas d'attente cumulé aux feux rouge, nombre d'accidents, etc ... avec un rendu graphique.

Il doit aussi être facile de configurer de nouvelles politiques, que ce soit pour les feux, les automobilistes ... Mais aussi le nombre d'automobilistes, le point de départ et d'arriver (par exemple départ de la maison et arrivé au travail).

1.2 Backlog

Fonctionnalités	Priorité	Statut
En tant qu'utilisateur, je peux voir la circulation	1	✓
En tant qu'utilisateur, je peux mesurer différentes valeurs (temps d'attente à un feu rouge, nombre d'accidents,...)	2	✓
En tant qu'utilisateur, je peux configurer des politiques	3	✓
En tant qu'utilisateur, je peux contrôler les feux	4	✓
En tant qu'utilisateur, je peux changer le nombre d'automobilistes	5	✓
En tant qu'utilisateur, je peux configurer les départs , les destination	6	✓
En tant qu'utilisateur, je peux modifier le plan	7	

1.3 Paramètres configurables

Nous avons plusieurs paramètres autres que les éléments de bases (voitures, routes, politiques ...) qui sont configurable. Il s'agit :

- Temps entre les mises a jour de la simulation, en millisecondes, au niveau du lancement de la fonction de mise a jour de la simulation.
- Nombre d'images par seconde maximum au même niveau que le temps entre les mises a jour : réalisé en utilisant l'opération 1000/fps pour obtenir le temps nécessaire entre chaque images.
- distance des feux de circulation par rapport a l'intersection, a entrer au niveau de la fonction d'affichage pour le rendu graphique et a la création de la simulation pour la partie fonctionnelle.
- distance de sécurité qui régit la distance entre 2 voitures a partir de laquelle un accident est détecté. A configurer a la création de la simulation.

1.4 Réalisation

1.4.1 Architecture du logiciel - MVC

Nous avons utilisé une architecture du type MVC (modèle, vue, contrôleur) pour réaliser cette application. Le contrôleur mets a jour le modèle et la vue lit l'état du modèle pour en faire un

rendu graphique.

La répartition des classes se fais comme suit :

- Modèle : "Car" qui défini un automobiliste, "Intersection" qui défini les intersections, "Objective" qui défini les objectif (par exemple maison ou travail), "Road" qui défini les routes et "SimulatorMap" qui représente la carte grâce a tous les objets précédents.
- Vue : "Panel" qui s'occupe d'afficher la carte
- Contrôleur : "Simulation" qui s'occupe de toute la partie simulation, principalement des mise a jours de positions d'automobiliste pour le moment, et a terme des mise a jours des feux suivant une politique.

Il existe aussi un package Planner qui contient la classe "DijkstraPlanner" qui s'occupe de planifier le trajet des automobilistes avec l'algorithme de Dijkstra.

1.4.2 Les routes

Une route est représenté par un point de départ et un point de fin ainsi que la liste des objectif se trouvant sur cette dernière. Le vecteur de déplacement d'un automobiliste sur la route correspondant au vecteur entre le point de départ et de fin de la route, chaque route est forcément a sens unique. Pour obtenir une route a double sens il suffit juste de prendre 2 fois la même route mais en inversant le point de départ et d'arriver. Le simulateur devant prendre en compte les routes a sens unique il s'agit du moyen le plus simple de les gérer.

1.4.3 Le graph

Pour aider l'algo Dijkstra, nous devons transformer la carte en un modèle plus simple. Nous décidons de le représenter comme un graph avec sommet est un modèle appelé Intersection.
Une intersection est représentée par une coordonnée et une liste de routes auxquelles elle est reliée.
Pour obtenir le graph, nous calculons tout d'abord l'intersection (coordonnée) à partir de la liste de la route en utilisant cette technique. Ensuite, pour chaque intersection, nous faisons une liste des autres intersections qui y sont reliées en trouvant leur route commune.

1.4.4 Algorithme Dijkstra

Comme dit plus haut nous avons planifié le trajet avec l'algorithme de Dijkstra, a partir de la carte représenter sous forme de graph. Chaque nœud du graph correspond a une intersection représenter par des coordonnées et l'ensemble des routes connecté sur cette intersection. Le cout entre deux nœud (nécessaire pour l'algorithme) est donc la distance entre 2 intersection.

Pour chaque nœuds on regarde pour chaque routes connecté si l'objectif final de l'automobiliste se trouve dessus, si c'est le cas on ajoute le chemin a la liste des chemin correcte sinon on continue l'exploration sur la prochaine intersection. Une fois l'exploration fini on récupère le chemin le plus court grâce a la distance entre chaque intersection ce qui nous donne une liste d'intersection a rejoindre, en prenant la route commune des intersections qui se suivent on obtiens la liste des routes a emprunter.

1.4.5 Distance entre les voitures

La distance entre les voitures est comme dit dans la partie des paramètre régit par un paramètre configurable. Lorsque la distance entre 2 voitures est inférieur a cette distance de sécurité, la

simulation considère qu'il y a accident. Pour faciliter l'implémentation de politiques en rapport avec cette fonctionnalité, une fonction qui simule la prochaine position de la voiture va detecter si la situation est dangereuse, et dans ce cas, demander a sa politique l'action a suivre : avancer ou s'arrêter. Pour éviter les faux positifs, par exemple 2 routes parallèles avec une distance très faible entre elles, la détection d'accident ne se fais que si les voitures sont sur la même route.

"detection situation dangereuse"

```
public boolean incommingAccident(Car currentCar, int safetyDistance){
  double[] roadVector = currentCar.getCurrentRoad().getVector();
  double[] coordinates = new double[2];
  coordinates[0] = currentCar.getCoordinates()[0] + roadVector[0];
  coordinates[1] = currentCar.getCoordinates()[1] + roadVector[1];
        for(Car car: this.cars){
          if(!car.hasFinish()){
            //this line is to check if both car are on the same road
        if(car != currentCar &&
        currentCar.getCurrentRoad() == car.getCurrentRoad()){
                  if(this.distance(coordinates, car.getCoordinates()) <=
                  safetyDistance){
            return true;
                  }
                }
          }
        }
  return false;
}
```

1.5 Problèmes

1.5.1 Planification

Un des premiers problème qui est survenu est la planification du trajet des voitures. Nous avions l'idée d'utiliser un algorithme de Dijkstra pour utiliser le chemin le plus court entre le point de départ et d'arriver, mais il n'est pas utilisable facilement sur une carte avec des routes représenté par des points de départ et de fin. Pour ce faire nous avons transformer la carte en graph, Chaque intersection devient donc un nœud. Ainsi la planification est beaucoup plus simple a implémenter puisqu'il s'agit d'un parcours de graph.

1.5.2 Affichage

Le second problème rencontré était sur l'affichage du trafic, la mise a jour graphique étant asynchrone, la mise a jour de la position des automobilistes était trop rapide et donc il n'y avais un rendu que sur 1 ou 2 images. Pour ce faire nous avons ajouter un Thread.sleep() configurable avec une variable de type int correspondant au temps d'attente, en millisecondes, entre chaque mise a jour de la position des automobilistes pour laisser assez de temps au programme d'afficher chaque mise a jour.

1.5.3 Gestion des feux en fonction du nombre de routes

//TODO