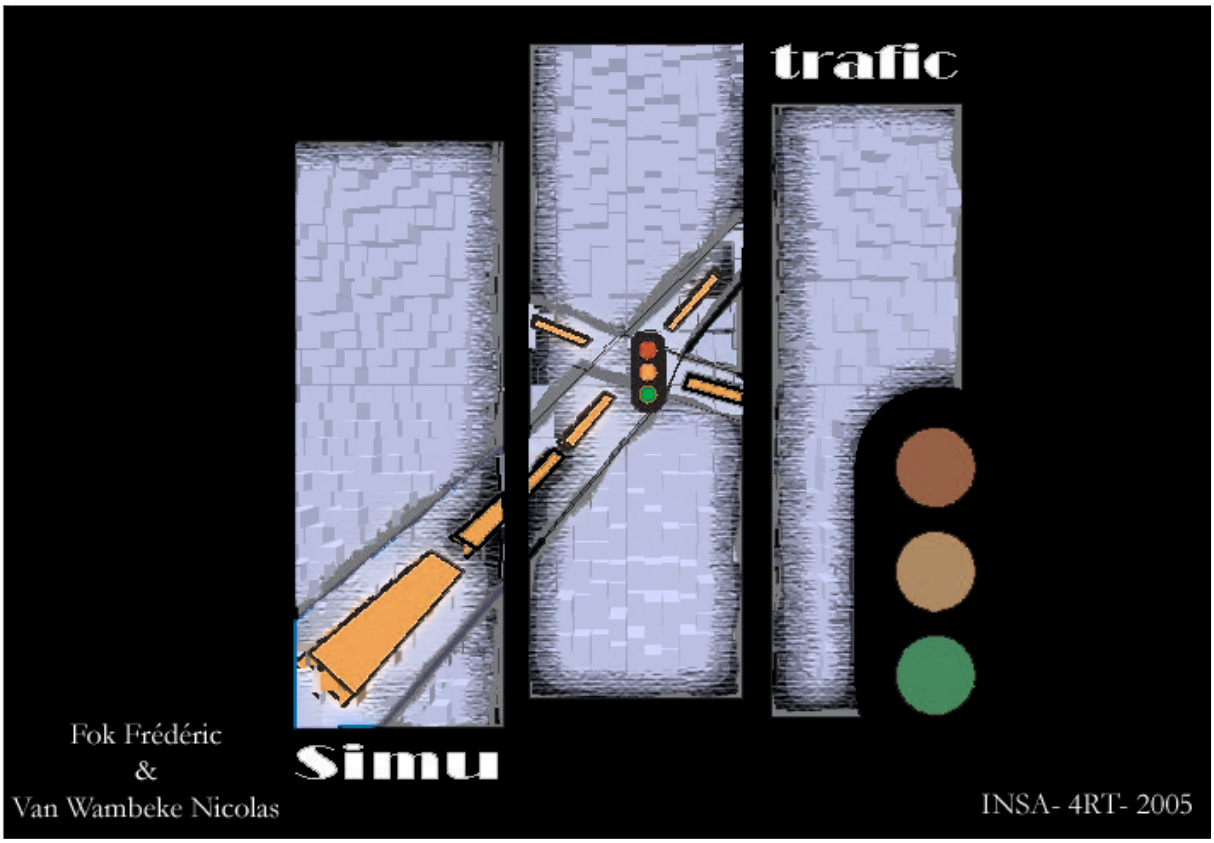
PROJET JAVA



SIMULATEUR DE RESEAU ROUTIER

SimuTrafic 2005

- INTRODUCTION-

Pour la réalisation du programme, nous avons décomposer le problème en deux parties, qui ne sont pas entièrement indépendantes l'une de l'autre. La première partie concerne la création d'un réseau routier et la seconde partie concerne le mécanisme de simulation.

Au départ nous avons été lancé au vif du sujet, à établir des spécifications, diagrammes de vues des cas, de classes, d'objets. Nous nous sommes vite rendu compte que le travail effectué concernait essentiellement un aspect « créatif », à savoir comment gérer la création d'un feu ou bien d'une route, quelle objet dépendait d'un autre, ou bien héritait t-il d'un autre, ou encore aurait il un rôle qui se rapproche du comportement d'un « thread » ? Puis petit à petit, on a commencé à analyser l'aspect simulation du réseau, et nous avons détaché les contraintes du système et analyser comment nous allions gérer la simulation du réseau routier.

Nous détaillons donc dans une première partie la façon dont le programme va réagir face aux actions de l'utilisateur, quelle type d'interface va être mis en place et comment se comportera le programme.
Puis, nous donnerons les explications sur le passage entre le mode création et le mode simulation.
Enfin, dans une dernière partie nous analyserons le mode simulation : la gestion des objets avec leur environnement, les choix effectués, la gestion des comportements et les scénarios possibles.

NOTES SUR L'INSTALLATION

Les sources du projet sont disponibles dans le fichier SimuTrafic2005.zip, une fois décompressées, le répertoire SimuTrafic2005 représente la racine du chemin des classes du projet. (Sous éclipse, il suffit de placer ce répertoire dans le workspace actuel et d'ajouter un projet du même nom)

Le lancement se fait soit via Eclipse en spécifiant la classe principale : GuiSimuTrafic.SimuTrafic

Il est également possible depuis la ligne de commande :

```
C:\SimuTrafic2005>java GuiSimuTrafic.SimuTrafic
```

SPECIFICATIONS DU BESOIN

GESTION DU PROGRAMME DE SIMULATION D'UN RESEAU ROUTIER

A- SPECIFICATIONS APPORTEES PAR LE SUJET

Le sujet présente les caractéristiques des objets. Leurs spécifications sont les suivantes :

Les routes sont caractérisées par :

- *Leur type (route, autoroute)*
- *Le sens de circulation (simple, double)*
- *Le nombre de voies*
- *La vitesse maximale*
- *Le nom*
- *Les côtes physiques*
- *Les tenants et aboutissants*
- *Les coordonnées cartésiennes*

Les carrefours sont caractérisés par :

- *Le nom*
- *Les routes attenantes*
- *Le type (à priorité à droite, à balise prioritaire, à feux)*
- *Les coordonnées cartésiennes*

Des feux sont caractérisés par :

- *Leur cycle (période, durée dans chaque état)*
- *Leur hauteur*
- *Leur principe de mode dégradé*
- *Les coordonnées cartésiennes*
- *Leur position relative sur le carrefour*

Des véhicules sont caractérisés par :

- *Leur type (voiture, camion, moto, vélo, camping car, semi-remorque, train-double, bus)*
- *Leur modèle dynamique (vitesse, accélération, glissement, freinage, etc.)*
- *Le type de conduite*

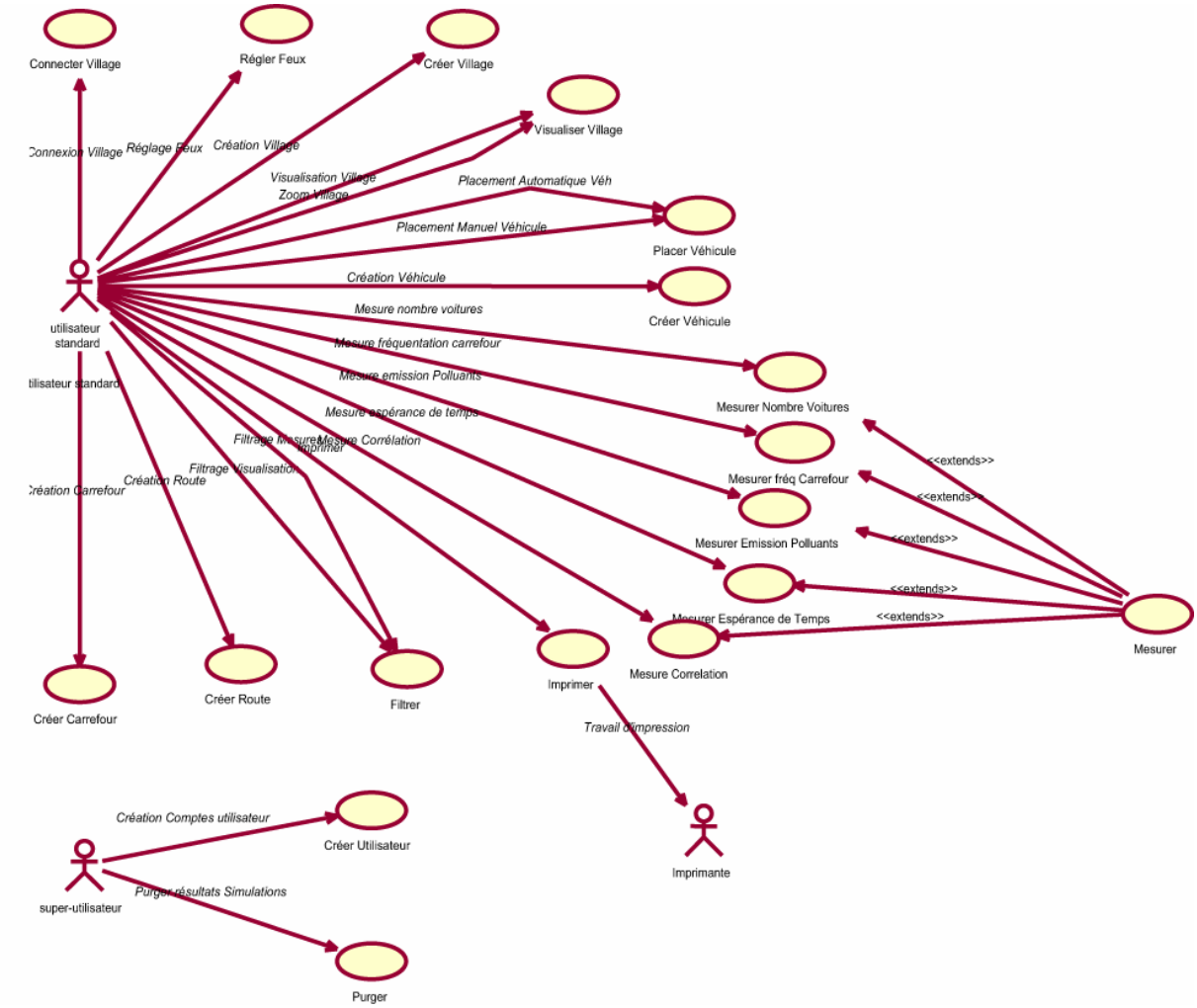
- *Leurs caractéristiques physiques*
  - *Masse, côtes*
  - *Emission de polluants*
  - *Couleur*

B- PRESENTATION DU SYSTEME ET CONTEXTE DE DEVELOPPEMENT

Pour le logiciel que nous réalisons, le langage de programmation retenu est JAVA. La conception est bien entendu orienté objet, et à pour vocation à nous faire développer un premier logiciel en passant par les étapes nécessaires de la conception logicielle. Il s'agit d'un simulateur de réseau routier. Un village et un seul est crée par le programme. L'utilisateur pourra y placer des routes, intersections (carrefours), et voitures. De plus, des paramètres seront réglables tels que le temps, les comportements des voitures sur la routes, aux carrefours, et ainsi de suite. Des mesures seront effectuées pour faire un compte rendu quantitatif du trafic et de l'état du réseau routier.

C- PRESENTATION DU PRODUIT LOGICIEL

Ci-dessous, le diagramme de vue des cas montre l'interaction entre les acteurs (utilisateur standard & super utilisateur) avec le simulateur et les évènements possibles qu'ils peuvent être amené à générer.



C.1- Présentation des acteurs

L'Utilisateur Standard :
Il s'agit de l'utilisateur du simulateur. Il peut créer et régler les différents éléments du réseau routier.

Le Super Utilisateur :
C'est l'administrateur du simulateur. Il a la possibilité en se connectant de crée des Utilisateurs Standards et purge les résultats.

Imprimante :
Ce n'est pas un acteur primaire mais un intervenant. Il permet d'imprimer les résultats obtenus lors d'une simulation.

C.2- Spécifications fonctionnelles

Voici, pour chacun des acteurs, les fonctionnalités qui leur sont associées. Elles récapitulent les fonctions des acteurs vis-à-vis de la simulation du réseau.

Super Utilisateur :

Créer des comptes utilisateur
Purger les résultats des simulations obsolètes

Utilisateur Standard :

Créer un village
Créer des routes
Créer des carrefours
- Créer Feu
-

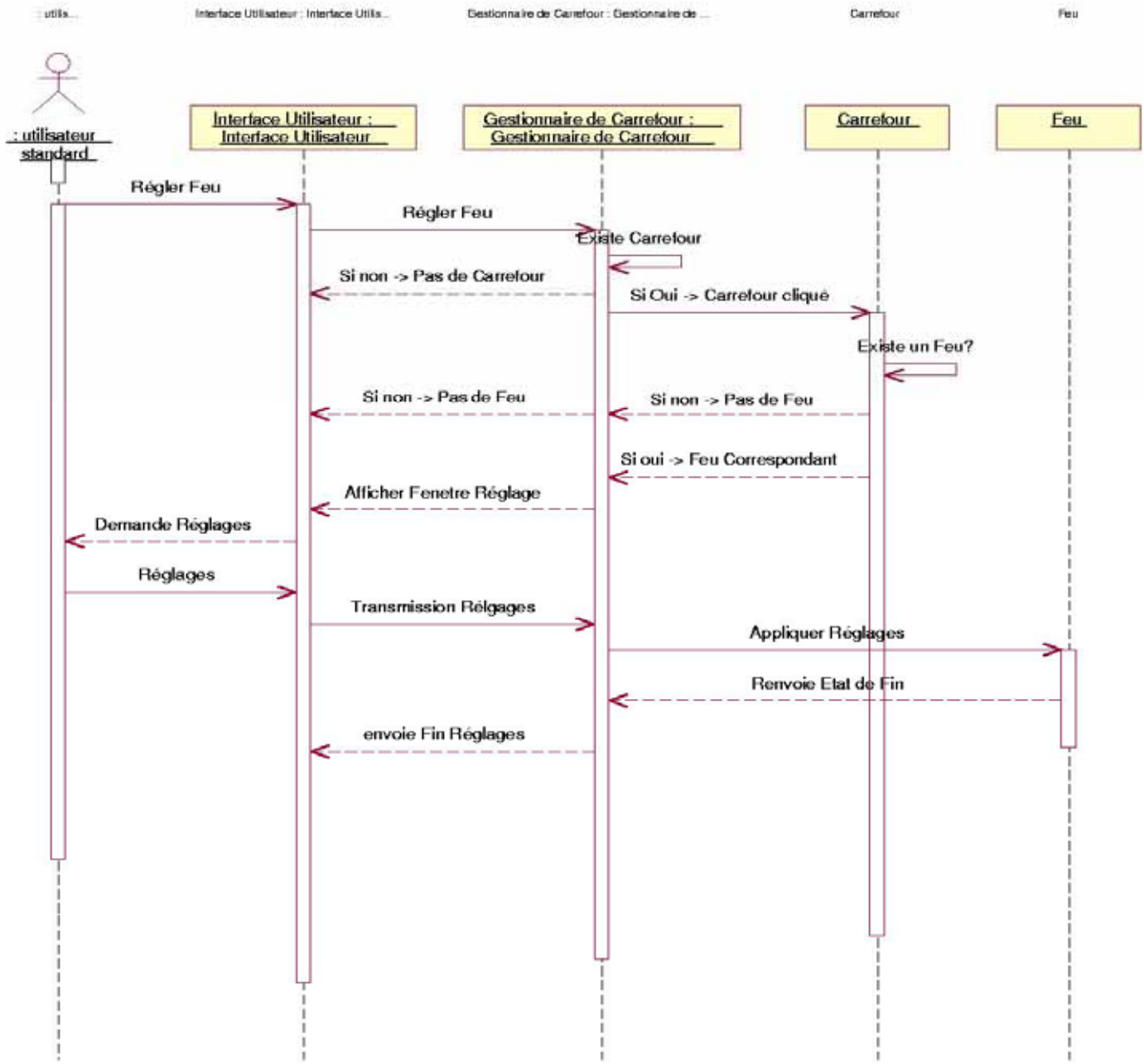
Créer des Véhicules
Placer des véhicules
Visualiser l'état d'un village
Connecter le village à d'autres villages
Mesurer les variables statistiques
- Nombre de voitures par heure a un point donné
- Fréquentation d'un carrefour
- Emissions de polluant à un point et heure donnée
- Espérance de temps pour aller d'un point A a B
- Corrélation entre route et véhicules

Filtrer pour focaliser sur un critère de la simulation
- Filtrage de la visualisation
- Filtrage des mesures

Imprimer les mesures produites

Exemple de diagramme d'évènement :

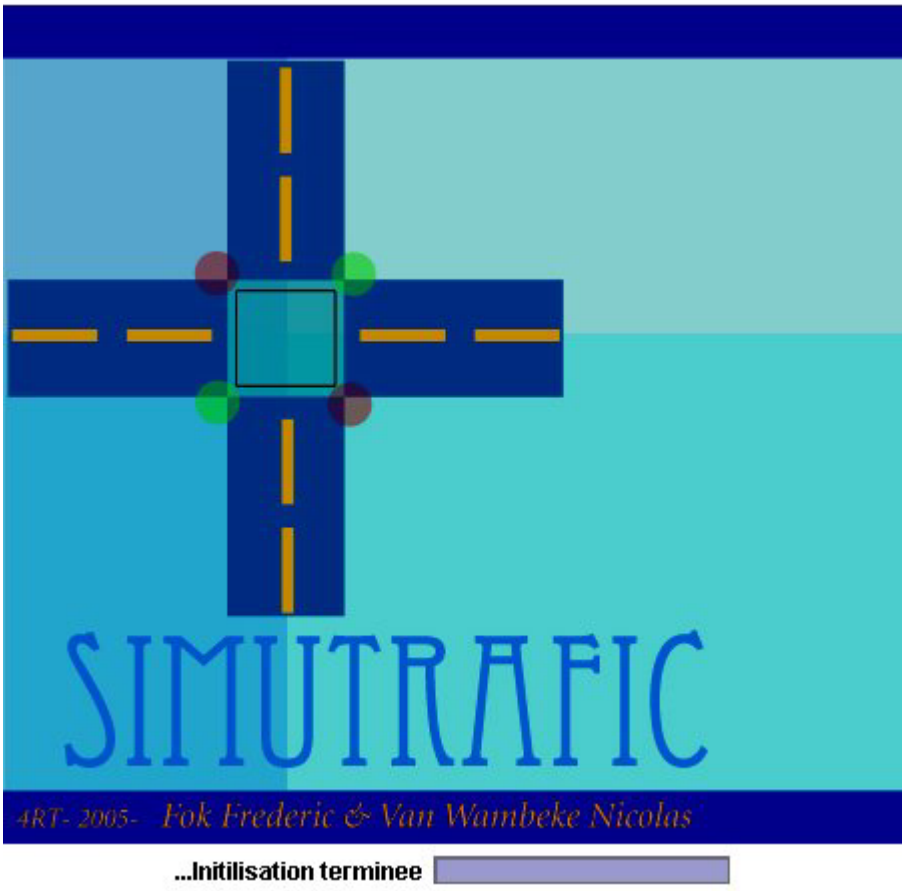
Pour chaque évènement, un diagramme de séquence pourrait être réalisé.
A titre d'exemple, voici celui du réglage d'un feu tricolore dont voici le schéma :



File: /home/nvan_warn/gl/lp/tp1.mdl Tue Jan 4 16:59:12 2005 Sequence Diagram: Use Case View / Régler Feux Page 1

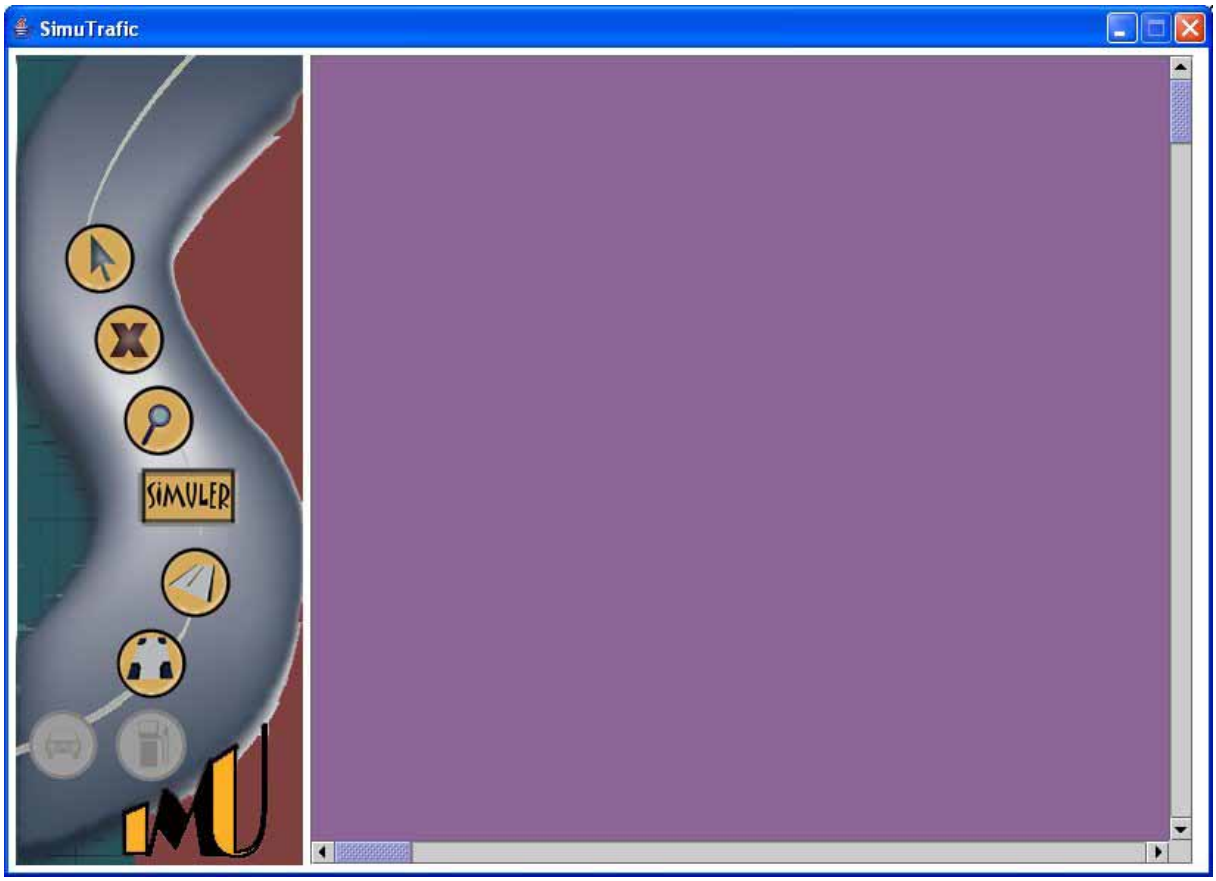
D- PRESENTATION GENERALE DE L'INTERFACE UTILISATEUR.

Le programme commence et lance un écran d'accueil éphémère et illustré. Cet écran présente les auteurs, la date, ainsi que le cadre dans lequel a été réalisé le projet.



Puis après une brève attente, l'écran principal s'affiche. Il s'agit de l'IHM (Interface Homme Machine). Cette interface utilisateur est simple et permet à l'utilisateur de créer « graphiquement » un réseau routier. Il a sa disposition une barre d'outil sur la gauche qui lui permet de choisir les éléments qu'il souhaite ajouter au réseau routier, initialement vide. A droite se situe la fenêtre de création du réseau routier c'est là que l'on peut dessiner le réseau.

Une fois le réseau crée, l'utilisateur peut lancer la simulation en appuyant sur le bouton Simuler. Les options du mode création ne sont plus disponibles lorsque l'on reste en mode simulation.



Lorsque l'on charge la simulation, on met en place le réseau routier pour en adapter l'esthétique à la simulation. On aboutit à une nouvelle carte routière où est tracé le réseau routier dans sa topologie réelle et où les feux sont en place.

Sur la droite de la fenêtre, on dispose d'un tableau de bord qui permet de gérer la simulation (quitter la simulation, générer des véhicules, lancer un scénario, modifier des paramètres tels que l'écoulement du temps, les conditions atmosphériques, les véhicules à générer, ainsi de suite…).

[Mettre image …futur]

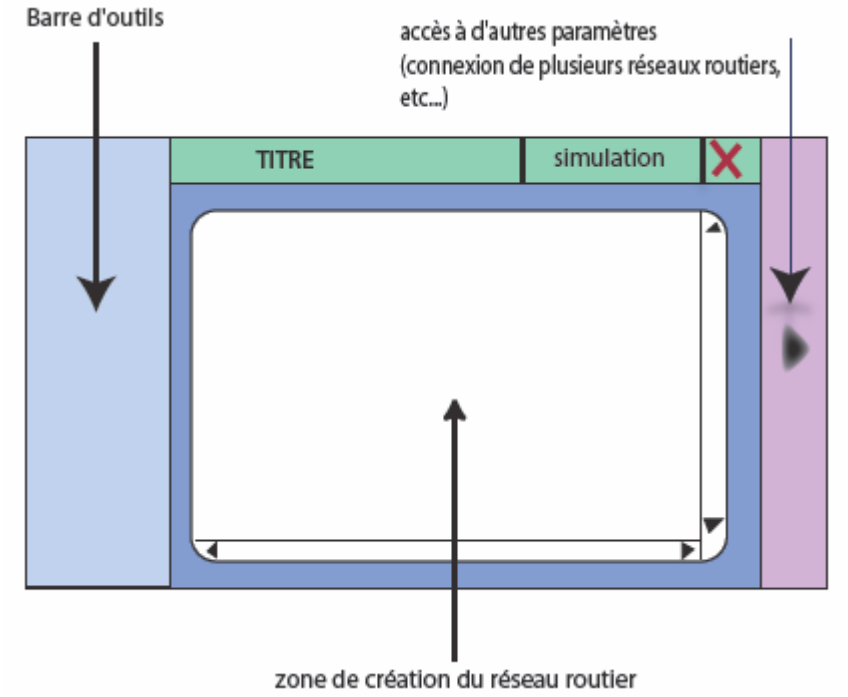
E- GESTION DE L'INTERFACE GRAPHIQUE :

E.1- Interface graphique en mode création

La barre d'outils fourni une quantité d'objets que l'on poser sur la zone de création. Il s'agit des routes et des carrefours *et d'autres objets pourront éventuellement se rajouter par la suite*.
On dispose de plus d'autres outils qui permettent de supprimer un objet du réseau routier, d'une loupe pour zoomer sur une partie de la zone du réseau routier. Tous ces objets sont sélectionnés par un click de souris sur leur icône respective puis par un click sur la zone de dessin à l'endroit ou l'on désire que l'outil prenne effet. La loupe fonctionne différemment selon que le click sur la zone de dessin se fait avec le bouton gauche (grossissement) ou le droit (éloignement) de la souris.

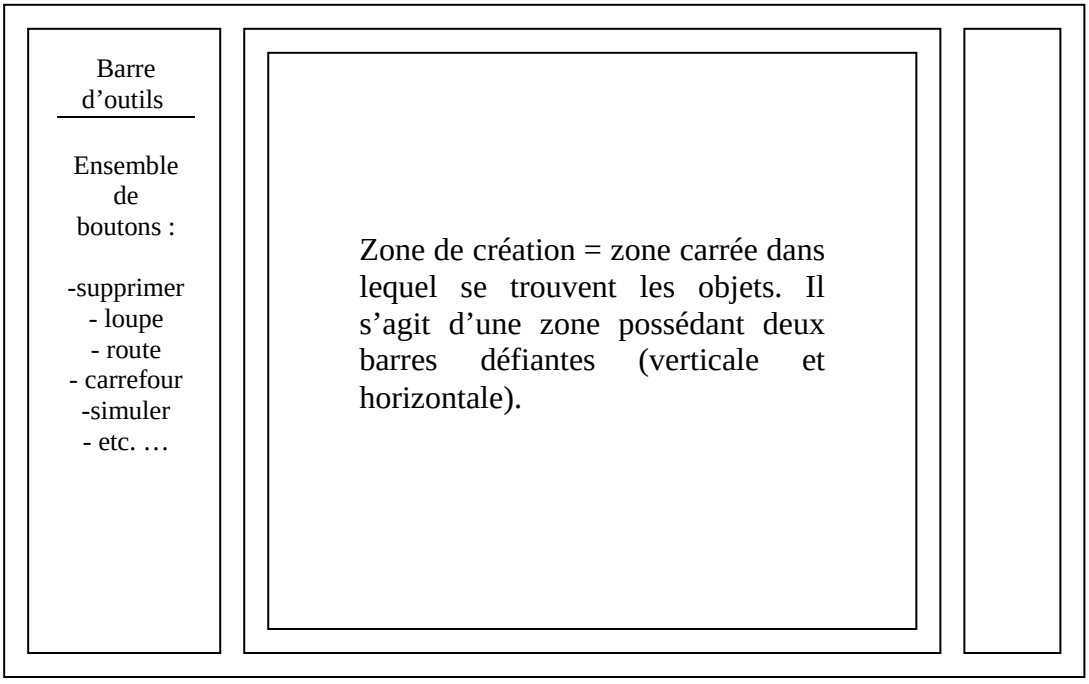
Certaines options ne sont pas accessibles dans les modes création et simulation. Par exemple le bouton d'ajout d'un véhicule n'est disponible qu'en mode simulation.

La fenêtre est découpée de la manière suivante :



Nota : la barre de titre le bouton simulation ont été intégrés à la barre d'outils dans la version finale

Tous les éléments sont à l'intérieur d'un conteneur principal (conteneur_principal).
On dispose ensuite 3 sous conteneurs dans le conteneur principal qui correspondent respectivement à : barre d'outils, zone de création du réseau, zone de paramètres.



Pendant la simulation, il peut arriver qu'une ou plusieurs voitures tombent en panne d'essence. Afin de les relancer, nous avons prévu le bouton de la pompe à essence dans la barre d'outils. Malheureusement son implémentation n'est pas encore fonctionnelle.

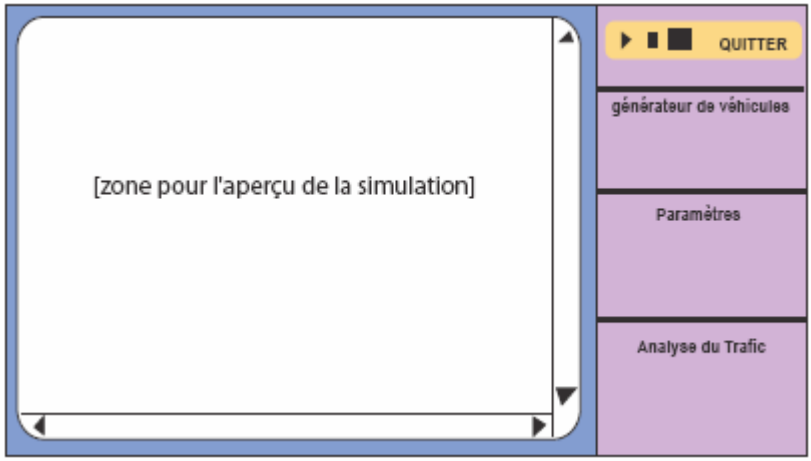
E.2- Interface graphique en mode simulation

Le réseau précédemment construit y est redessiné et les objets y sont placés et prêt à interagir avec l'environnement dans leur forme réelle (ex : carrefour à 3 voies avec passages pour piétons).

Sur la partie droite de la fenêtre, on dispose d'un tableau de bord permettant de contrôler la simulation. On peut enclencher la simulation, mettre en pause, stopper la simulation, générer des véhicules, analyser le trafic (zone du réseau routier à choisir, etc. …), et régler des paramètres (intempéries, durée de la simulation, choisir un scénario, etc. …).

Nota : *Dans la version finale nous avons préféré limiter les fonctionnalités de cette barre et transformer les fonctions que nous implémentons en fenêtres de dialogue. Ainsi, l'outil de sélection permet d'afficher et de régler la durée des feux lorsque l'on clique sur un carrefour en mode simulation et la vitesse maximale autorisée sur une route lorsque l'on clique sur une route.*

Le côté droit de la fenêtre est décomposé de la manière suivante :



Le côté droit se décompose donc en deux grandes zones : la zone de gauche et la zone de droite.
On place un conteneur principal dans la fenêtre.
Ensuite on insère deux sous conteneurs dans le conteneur principal. Le premier (celui de gauche) contient la zone où on peut voir la simulation se dérouler, et le deuxième concernant le tableau de bord de la simulation.
Dans le premier sous conteneur, on place une zone avec des barres défiantes (objet ReseauRoutier).
Dans le deuxième sous conteneur, on place plusieurs conteneurs pour placer la barre de simulation (lecture, pause, arrêt, quitter), la zone de génération de véhicules, la zone de paramètres et la zone d'analyse du trafic.

E..3- Autres éléments graphiques

Il existera des boites de dialogue lorsque l'utilisateur cliquera sur certains objets en mode sélection. Cela permettra de définir certains paramètres. En effet, la création de carrefours et de routes se fait avec des paramètres par défaut de façon à rendre agréable l'utilisation du programme et de pouvoir construire un réseau assez rapidement sans avoir à donner les valeurs des paramètres à chaque fois avant de créer l'objet !

ANALYSE DU MODE CREATION DU RESEAU ROUTIER

A- GESTION DE LA CARTE ROUTIERE

A.1- Les éléments d'une carte routière :

C'est un plateau de taille maximum, découpé en carrés de taille donnée contenant tous des objets de type SimuTrafficBloc. Cet objet de type polymorphe étends un JLabel et permet de représenter plusieurs allures de cases en fonction de la topologie du réseau dessiné par l'utilisateur.
On placera des routes, carrefours, etc. …, dessus. Une route prendra L carrés où L est la longueur de la route.

A.2- Un village valide :

Un village pourra être simulé s'il possède un réseau routier fermé, une entrée de village et une sortie de village.
Une fois le réseau dessiné, le tableau de SimuTraficBlocs est converti en un objet de type ElementVillage et c'est lors de cette conversion que s'effectue le changement d'aspect graphique des cases en fonction des branchements. Ensuite, un élément propre au moteur de simulation est généré de type Village et les connections du moteur au rendu graphique sont effectuées.

B- GESTION DES ROUTES ET DES CARREFOURS

L'utilisateur dispose d'une boite à outil dans laquelle il peut choisir quel élément il veut placer sur le réseau routier. *Le réseau routier du village doit être fermé ! c'est-à-dire qu'il n'existera pas de routes cul-de-sac, toutes les routes aboutiront à un carrefour sauf la route d'entrée ou de sortie du village.* Ainsi, pour vérifier la fermeture du réseau routier, on n'aura pas à examiner les routes puisqu'elles aboutiront forcément à un carrefour ou un raccordement. Il suffira donc de regarder l'état des carrefours. Il est donc primordial de placer en premier les carrefours d'extrémités avant de placer une route entre ceux-ci. C'est le village qui aura connaissance de ces éléments et de leur état. Ainsi, si tous les carrefours interconnectent pleinement les routes, la simulation sera prête à être lancée.

On pourra placer des carrefours en choisissant l'icône carrefour et en cliquant sur un endroit de la carte. L'action crée un carrefour par défaut aux coordonnées choisies, il est par contre impossible de placer deux carrefours l'un à coté de l'autre. *On pourra changer les caractéristiques du carrefour en cliquant dessus (nom, feux ou non, priorité ou non …)*. La création d'une route se fait par un click avec l'outil de création des routes entre deux ou quatre carrefours, la création engendre donc selon le cas une route ou deux routes interconnectées par un carrefour. Dans l'implémentation choisie, on crée donc un carrefour par défaut, avec ses caractéristiques par défaut (4 feux mis en place avec leur comportement par défaut). L'utilisateur a ensuite la possibilité de changer ces caractéristiques en cliquant dessus. Il est possible de zoomer sur des endroits du réseau aussi bien en mode création qu'en mode simulation.
Pour la création des routes, on a plusieurs possibilités : route simple direction, double sens, autoroutes. Si il s'agit d'une entrée ou sortie dans le village, on aura une icône de route spéciale pour le préciser.

Diagramme séquence placer route :

a) L'utilisateur clique sur l'outil route normale, il devra au préalable avoir placé deux carrefours.

b1) L'utilisateur clique sur une case entre deux carrefours. Si la à créer route ne superpose pas un élément, la route est créée automatiquement entre les deux carrefours. Si la route superpose une autre route, un carrefour est crée à l'intersection des 2 routes.

b2) Si il clique sur une case n'entourant aucun carrefour, alors rien ne se passe. C'est-à-dire, pas de création de route.

Séquence placer carrefour :

a) L'utilisateur clique sur un carrefour.
b) Si il clique ensuite sur la carte, un carrefour est positionné à condition qu'il n'y ait aucun carrefour qui lui soit directement connecté
c) Sinon, rien ne se passe.

C- LA GESTION DES OUTILS DE LA BARRE D'OUTILS

On dispose d'outils particuliers qui agissent sur la zone de création. On a une loupe qui permet de zoomer sur une partie de l'écran de création du réseau routier. On a aussi un outil de suppression qui permet de détruire n'importe quel objet de la zone de création.
Enfin on a un outil de sélection qui permet, en simulation, de sélectionner un objet (un clique sur un objet permet d'ouvrir une fenêtre de gestion des paramètres de l'objet : carrefour à feux ou à priorité, etc. …).

D- LE VILLAGE

Ses attributs :

- simule : boolean
  - permet de savoir si la simulation du village est en cours

Ses méthodes :

- villageValide()
  - Retourne un boolean
- créer(simule)
  - Revient en mode création du village
- simuler(simule, scénario)
  - Lance le mode simulation du village

E- SPECIFICATION DES INTERFACES EXTERNES

E.1- Utilisateur Standard :

Il utilise l'interface utilisateur réalisée en Java à l'aide de la librairie graphique SWING. Elle lui permet un accès aux fonctions détaillées ci-dessus à l'aide de composants graphiques (menus, boutons, fenêtres, paramétrage, règles, loupe, curseurs)

E.2- Super Utilisateur :

Idem que l'utilisateur Standard. De plus, elle possède un menu spécifique donnant accès aux fonctions propres du super utilisateur. Ce menu est révélé par l'utilisation d'un mot de passe.

LE PASSAGE DU MODE CREATION AU MODE SIMULATION

Le village est le centre de deux modes, celui de la création et celui de la simulation.
Lorsqu'on lance le programme, on entre directement en mode création. *Le mode simulation ne peut être lancé que si le village est valide, c'est à dire possédant un réseau routier fermée (ouvert ou non vers d'autres réseaux routiers).*
Ainsi, le mode création se détache du mode simulation du point de vue comportemental. On n'a pas à se soucier du comportement des objets (voitures, carrefours, routes…) qui agissent dans leur environnement (réseau routier) en même temps que de la création d'une route.
On pourrait bien sûr faire en sorte de gérer les deux en même temps et laisser la possibilité à l'utilisateur de créer une route en plein milieu de la simulation. Mais cette approche nous conduirait à traiter un très grand nombre d'erreurs. Par exemple, le retrait d'une route doit il se faire lorsqu'une voiture se trouve dessus ? Comment gérer le déplacement de la voiture dans un environnement qui agit non seulement avec les autres objets mais aussi avec les actions éventuels de construction de l'utilisateur ?
Afin de simplifier l'approche de notre simulateur routier, nous avons séparé les deux modes. En effet, quelle est l'utilité, en fin de compte de laisser à l'utilisateur la possibilité de tout changer sur le réseau routier en pleine simulation ? Car après tout, ce qui nous semble important est d'observer l'interaction des objets dans un environnement déterminé.

Une fois que l'utilisateur a créé son réseau routier, on doit vérifier que le réseau routier est fermé. De par l'implémentation choisie, aucune route n'aboutit sur rien. On n'a donc qu'à se préoccuper des carrefours. Lors de leur création, ils sont enregistrés dans une liste. Il suffit de passer en revue cette liste et de vérifier pour chaque carrefour qu'il possède au moins deux voies connectées.

Puis, les objets créées et dessinées sur la zone de création du réseau routier sont « convertis ». C'est-à-dire qu'on adapte le mode création au mode simulation.
Chaque objet crée est stocké en mémoire via l'utilisation de listes. On analyse chacun des objets pour créer un « agent » (sa définition sera expliquée plus loin) qui lui corresponde.
Ainsi, on transforme notre zone de création en zone d'agents prêts à interagir dans leur environnement.

La simulation consistera donc à générer nos véhicules dans le réseau routier et à analyser leur comportement, le trafic généré, etc. …

Pour ajouter un véhicule, l'utilisateur devra sélectionner l'outil véhicule dans la barre d'outils, il cliquera ensuite sur le carrefour depuis lequel il souhaite que la voiture démarre. La direction que prendra la voiture est déterministe en fonction de la topologie du carrefour.

Pendant la simulation, il peut arriver qu'une ou plusieurs voitures tombent en panne d'essence. Afin de les relancer, nous avons prévu le bouton de la pompe à essence. Malheureusement son implémentation n'est pas encore fonctionnelle.

ANALYSE GENERALE DE LA SIMULATION DANS LE RESEAU ROUTIER

LA GESTION DES COMPORTEMENTS

On a besoin d'une classe à part entière pour traiter les comportements des éléments.
Par exemple, les intempéries, les accidents, les actions des véhicules, le temps, la météo, etc…
Dans notre choix d'implémentation, Seul le super-utilisateur pourra modifier la gestion des comportements du réseau routier. Un utilisateur standard pourra changer le type de comportement (nerveux au volant, temps pluvieux, etc. …) mais il n'aura pas accès à la façon dont sont traités les comportements.
Le super-utilisateur pourra donc définir quel scénario associé à un comportement. Par exemple, s'il pleut, le taux l'accident va varier sur la route, mais la valeur du taux ne sera accessible que par le super-utilisateur.
Ainsi, le super-utilisateur aura accès à une fenêtre qui modifie les scénarios en spécifiant les taux, les valeurs, etc…

Pour le comportement de la voiture :

Du point de vue de l'utilisateur :

On va associer, lors de sa création, un comportement à un véhicule. (calme, stressé, fou, dangereux, etc…).
Pour chaque comportement, par exemple calme, il faudra pouvoir être en mesure de gérer aléatoirement son comportement sur la route. C'est-à-dire que ce type de comportement implique que le conducteur circule entre la vitesse autorisée et 20km/h en dessous de la vitesse autorisée. La vitesse variera en fonction de la route sur laquelle la voiture se trouve. Le comportement de ce type de véhicule fait qu'il n'engendrera pas d'accident mais il pourra par contre être victime d'un accident.
Si on prend un conducteur fou, il aura une probabilité de créer un accident (30 à 40 voir 60%). Ce type d'accident lui-même sera aléatoire (non respect à un croisement, changement de file…).
On peut prendre en compte les scénarios possibles : l'utilisateur pourra choisir un jour « vert » (peu de trafic), « orange » (beaucoup de trafic au cours d'une journée), « rouge » et ainsi de suite…

Du point de vue du super-utilisateur :

Dans une version plus améliorée du logiciel, à la valeur fou, on associerait une probabilité d'accident au choix du super utilisateur.
On pourra aussi décider d'un scénario journalier, la façon doit se faire le scénario avec le nombre de voiture à générer à une heure donnée d'une journée, etc…

BARRE DE SIMULATION

- definirScenario
  - Pour spécifier le type de scénario que la simulation doit produire.
  - Si aucun scénario n'est défini : scénario = « default » alors l'utilisateur lancera la simulation et c'est lui qui génèrera des véhicule quand il le souhaite.
  - Une simulation sera bornée à une journée, donc le temps s'écoulera 24heures au rythme de 1h=1 minute ⇒ simulation de 24 minute maximum.
  - Si on n'arrête pas la simulation avant la fin de la journée, la simulation reprend à zéro (remise à jour du trafic, du réseau, des véhicules …).

- executer
- pauser
- arret

REFLEXION SUR L'ASPECT COMPORTEMENTAL VIA L'EXEMPLE D'UNE VOITURE

Dans cette partie, on détail le raisonnement nous ayant amené à choisir une implémentation pour gérer les comportements de nos objets.

Généralités

Une voiture aura un nom (identificateur) et sera paramétrable, on pourra choisir sa couleur, etc. … L'utilisateur aura la possibilité de créer des voitures via un générateur de véhicules. La génération d'un véhicule se fera à un endroit aléatoire en début d'une route. Par ailleurs, si un scénario est choisi, il y a aura génération automatique de véhicules à certaines heures précises définies par le scénario.

Diagramme séquence :

Lorsque l'utilisateur génèrera le véhicule de son choix dans la zone de commande de la simulation, celui-ci sera généré dans le réseau routier, à un carrefour aléatoire !
Ce mode de choix consistera à envoyer un agent voiture sur le réseau routier.

Implémentation d'une voiture :

La classe voiture n'a pas à gérer l'ajout de voiture ni son emplacement…
Il s'agit d'un objet simple paramétrable via ses propres méthodes :

- changer_couleur
- type
- vitesse_maxi, vitesse_mini

Et il est défini par ses caractéristiques initiales: vitesse_maxi, vitesse_mini, couleur, masse, côtes, emission_poluant, type_conduite (comportement)
La voiture sera un thread (on incrémentera sa position et donc l'affichage au fur et à mesure). Une voiture aura un comportement qui lui est propre, un rôle. Il sera réactif puisqu'il agira en fonction des évènements extérieurs. Ainsi, il possèdera des compétences à savoir : s'arrêter, ralentir,

avancer, rouler à une vitesse qui dépend de la route sur laquelle la voiture se trouve. La voiture aura aussi une compétence « temps de réaction » qui dépendra du type de conducteur au volant !! La voiture connaîtra la route sur laquelle elle se trouve et l'existence d'une voiture à proximité. Pour cela, il faudra définir sur quelle distance on cherche l'existence d'une voiture.
Ainsi, une voiture détectant trop tard une voiture du fait de son temps de réaction trop long, heurtera la voiture qu'elle suit.
La voiture possède ses méthodes : modifNom(), modifCouleur(), modifComportement, modifVitesse. Pour gérer ces paramètres,étant donné le grand nombre possible de voitures en fonctionnement sur le réseau routier, le générateur de véhicule aura aussi connaissance des véhicules qu'il a crée.
Afin d'optimiser cette gestion, on pourrait inclure ces méthodes dans les compétences des véhicules. Ainsi, on pourra envoyer en masse des véhicules avec moins de compétences mais répondant à un certains nombre de contraintes (connaissance de la route, des véhicules qui suivent, des carrefours, etc. ...). Et on pourra envoyer des véhicules spécifiques afin d'étudier le comportement et les effets de ce véhicule sur le trafic du réseau routier.

Comment gérer la simulation des voitures sur notre réseau routier.

Il va sans dire que la conception à cette étape doit être réfléchi. Il est donc important de déterminer les cas de figures possibles pour une voiture, quelque soit le modèle qui sera retenu (simples threads et librairie standard java, ou alors l'utilisation des environnements multi-agents).
PB : en effet, comment la voiture sait elle sur quelle route elle se trouve, comment avance t-elle, de quelle façon, quel est son comportement et comment réagit-elle aux évènements extérieurs ?

Quelle implémentation choisir dans le cadre du choix d'un environnement multi-agents ?

En effectuant diverses recherches, nous avons remarqué l'efficacité des environnements multi-agents ainsi que les perspectives plus qu'intéressantes qu'ils offrent. La notion est encore récente, et aucune norme n'oblige l'implantation d'un plateforme respectant des critères, car la notion d'agent peut prendre plusieurs sens dans sa conception même si les bases de sa sémantique sont communes. Il existe néanmoins la norme donné par la FIPA. On a donc remarqué et même étudié divers plateformes tels que MadKit, Swarm, Volcano, ou encore Magique, et il en existe d'autres. Finalement, nous avons porté l'attention sur la plateforme JADE qui est une plateforme en open source, respectant les normes de la FIPA et utilisable avec JAVA. La version retenue est la dernière en date : Jade3.2 (26 Juillet 2004). Après plusieurs essaies, il s'est avéré que les possibilités de cette plateforme étaient bien trop étendues pour la simulation que nous voulions créer.
Ainsi, nous avons décidé de mettre en place la notion d'agent en implémentant une mini plateforme qui se caractérise par des gestionnaires de messages : envoie et réception de messages pour la communication.

La notion d'agent vis-à-vis de notre réseau routier.

Nous nous sommes posé la question de savoir comment gérer la création des voitures sur un réseau routier et comment ces voitures allaient prendre conscience du réseau routier qui l'entoure. En effet, comment une voiture se comporte t-elle : quelle vitesse, respect d'un feux rouge, que fait

elle à une intersection, comment se déplace t-elle sur les routes, comment évite elle ou non les voitures, comment gérer les accidents.
Nous avons alors abordé la notion d'agent. Ainsi chaque agent serait représenté ici par une voiture ayant un **rôle**, des tâches à accomplir (séquentielles ou non, etc…) et une **compétence**, sachant que l'on ne peut pas représenter le rôle sans la notion de compétence. La voiture aurait plusieurs compétences possibles, dont celles expliquée avant) et le rôle serait d'avancer sur la route. Elle modifierait son comportement en fonction en réagissant aux évènements provenant de l'environnement.
Un carrefour serait aussi un agent dont le rôle serait de contrôler les voitures au niveau du carrefour.
Enfin, la route serait elle aussi un agent dont le rôle serait de contrôler la voiture qui s'y trouve.

Et le transfert de voitures d'un réseau routier à un autre ?

Les agents peuvent être distribués en utilisant de simples **sockets** ou en utilisant des techniques telles que **RMI** ou CORBA.

Si on considère qu'un véhicule doit son existence au noyau d'origine qui l'a conçu alors, dans notre cas, l'adoption d'un tel fonctionnement permettrait de gérer les voiture sur un autre noyau : « **Un agent peut quitter une machine pour s'exécuter sur une autre en étant toujours géré par le même noyau. »**
Lorsqu'un agent désire être transféré, il en fait la demande à son noyau qui à son tour adresse au noyau cible un message de demande de transfert qui contient le nom de l'agent à transférer. Si le noyau cible est d'accord il informe le noyau demandeur et l'agent peut migrer sur le noyau cible. Sinon, le noyau cible renvoie une réponse négative au noyau demandeur qui avertit l'agent.

On peut aussi, tout simplement sans noyau, envoyer des états de véhicules et créer les véhicules à l'entrée d'un village sur un autre ordinateur. Les véhicules seront créées si un véhicule est accepté sur la route d'entrée d'un village voisin.

Néanmoins , vis-à-vis des autres villages :
Les autres binômes n'auront pas forcément décider d'adopter la notion de multiagents, et donc de la notion de noyau (aussi un agent). Donc, transférer des voitures ou accueillir des voitures dans notre village ne serait donc pas forcément évident. Ou alors, il faut trouver un moyen d'importer et d'exporter les voitures : ***traduction du passage d'un réseau à un autre ?***
Mais si les autres villages sont issus de notre programme … :
Si on réalise l'interconnexion de même type de village, c'est-à-dire à partir de notre propre programme, on peut alors aisément réalisé l'interconnexion des villages (tournant sur des machines séparées) via la RMI et l'échange d'agent, sachant qu'un agent (voiture, je le répète…) dépend de son noyau d'appartenance.

Un agent : Comment s'en sert t-on dans une plateforme multi-agent?

Agent Générique : dans les plates-formes multi-agents classiques, il est usuel de dériver une classe MonAgent de la classe Agent et d'y ajouter des méthodes de comportement. Ces méthodes sont codées de façon statique et affinées en surchargeant des méthodes existantes dans les superclasses

Agent réactif : un agent réactif possède quelques compétences lui permettant d'accomplir une ou plusieurs tâches. Les compétences associées sont statiques et n'évoluent pas en fonction de l'agent. Un agent réactif se borne donc à exécuter certaines tâches prédéfinies. Cette classe dérive directement de la classe AbstractAgent. Il est possible d'ajouter ou de retirer dynamiquement une compétence réactive à un agent réactif mais l'agent ne peut le faire de lui-même.
Agent cognitif : Un agent cognitif peut donc se comporter exactement de la même façon qu'un agent réactif s'il n'utilise que des compétences réactives avec toutefois la possibilité ***d'acquérir dynamiquement de nouvelles compétences***. L'intérêt d'instancier un agent cognitif est de bénéficier des classes cognitives. Le savoir et le savoir-faire d'un agent cognitif peuvent changer dans le temps.

Les agents sont implémentés comme des threads d'exécution Java et les événements Java sont utilisés pour la communication efficace et légère entre agents sur un même hôte.
Pour la plateforme JADE : Un agent peut exécuter des tâches parallèles et JADE planifie ces tâches d'une manière plus efficace (et même plus simple pour le programmeur) que la planification faite par la Machine Virtuelle Java pour les threads d'exécution.
Rappel : JADE est plate-forme multi-agents compatible FIPA.

L'Agent Noyau est seul habilité à acquérir une compétence spéciale traitant de représentation de groupes d'agents. Outre ses compétences natives de gestion d'agents qui l'autorisent à jouer son rôle de Manager, il est aussi possible de lui adjoindre une compétence d'observation et de représentation afin qu'il endosse un rôle supplémentaire d'observateur.

Il s'agira dans notre modèle de notre village.

Définition de la notion d'agent pour le choix d'implémentation que nous avons conçu.

Les agents de notre réseau routiers sont des agents réactifs. Ils réagissent en fonction du message reçu. Ils peuvent adopter un comportement différent d'un autre. En effet, chaque agent a sa propre identité, son propre comportement. Il est cognitif dans la mesure où ses compétences peuvent évoluer dans le temps. En effet, si un utilisateur clique à un moment donné de la simulation sur une voiture, il aura la possibilité de modifier son comportement dans la limite des comportements possibles codés.

Les agents & leur environnement :

Un environnement est vu comme une vaste zone d'échange. Dans notre modèle, nous avons un village et l'environnement correspond au réseau routier du village.
Plus particulièrement, un environnement réactif est une classe regroupant un certain nombre de cellules que les agents utilisent comme parcours élémentaire pour se déplacer d'un endroit à un autre. Pour nous, ces cellules correspondent à des routes (simples, doubles ou autoroutes).
Ainsi, l'environnement se réduit à une carte liant des cellules adjacentes (routes adjacentes). Il peut donc être un plan (2 dimensions) permettant de simuler notre trafic routier.

Les agents « voitures » sur le réseau routier :

L'agent voiture, agent social et réactif peut interagir avec :

- ***route,***
- ***autres voitures,***

- *carrefour (et donc feux ou priorités),*
- *et les conditions atmosphériques (pluie, beau temps…).*

Elle possède une multitude de compétences :

- accélérer
- s'arrêter
- avancer en fonction de la route sur laquelle elle se trouve
- connaître la route sur laquelle elle roule
- connaître la voiture qui la suit et qui la précède.
- Se faire heurter

Et selon le type de voiture :

- Klaxoner
- Heurter un véhicule (cela dépend du temps de réaction du conducteur)
- Déclencher ses sirènes (pompiers, policiers …)
- Arrêter autre véhicule
- Créer un accident (conducteur fou)
- Ne pas respecter les feux aux carrefours
- Ne pas respecter les priorités aux carrefours
- S'arrêter à un carrefour et ne plus bouger volontairement un temps donné bouchons de circulation
- …

Pour simplifier le modèle, on interdira à la voiture de reculer.

Chaque agent voiture devra alors être capable de « dialoguer ». Cela se fait via des messages. La question qui se pose est de savoir comment traiter ces messages, comment associer le traitement de ces messages au programme et comment décider de la prochaine action à effectuer sur réception de ce message.
On a vu que chaque agent voiture réagit à l'environnement, il doit donc être doté de *capteur* pour la *perception* et d'effecteur pour *l'action.*
Il existera ainsi des canaux de communication entre voitures/voitures adjacentes et entre voiture/route.

Deux voitures partagent-ils certaines ressources ou objets ?

La question que l'on se pose est de savoir si deux voitures peuvent entrer en conflit et partager en même temps le même objet…
Nous n'avons pas encore de réponse concrète à ce sujet.

Les agents route :

Ils agissent avec les voitures. Ces dernières donnent régulièrement leur actions et la route met à jour sa « table de correspondance » (*qui reste à définir).* Les voitures reçoivent aussi des évènements venant de la route qui les informe de l'environnement. En effet, la route est l'environnement sur lequel les voitures circulent. Donc, la route saura si deux voitures se rapprochent l'une de l'autre, ou si une voiture se rapproche d'un carrefour. La route enverra alors des messages pour informer de la situation : <voiture, voiture>, <voiture, carrefour>, etc. … et les voitures commenceront un dialogue de message pour agir en conséquence.

Les agents carrefour :

Rappelons-le, ils contrôle les agents voitures.
En fait, pour être précis et exacts, les carrefours en eux même n'ont pas de rôle d'agent. Ce ne sont que des infrastructures avec leurs propriétés (nombre de routes connectées, nombre de feux, type de carrefour). Dans le cas d'un carrefour à feux, ce seront donc les feux qui régiront le comportement du carrefour en fonction de l'état du feu.

Le paradoxe de la connaissance

Il y a lors de l'interaction entre agent, une connaissance commune, connu par les deux agents routiers qui interagissent. Cette connaissance est représentée par un système de communication avec la mise en place de « boites aux lettres » appelées dans notre code « Gestionnaire de messages ». Chaque agent routier consulte régulièrement sa « boite aux lettres », ce qui lui permet de connaître son environnement.
Mais qu'en est il au niveau du temps ? Le système est en effet asynchrone, ce qui signifie que la connaissance commune n'est pas instantanée!!
Les méthodes d'accès aux messages sont définies par chaque gestionnaire.
Enfin, pour être interprétés, les messages doivent être « compréhensibles » par les entités du réseau routier. On a donc mis en place un langage qui défini les messages possibles et interprétables.
Bien entendu, certains messages ne seront compréhensibles que par certains agents.

LE REPERAGE DANS L'ESPACE

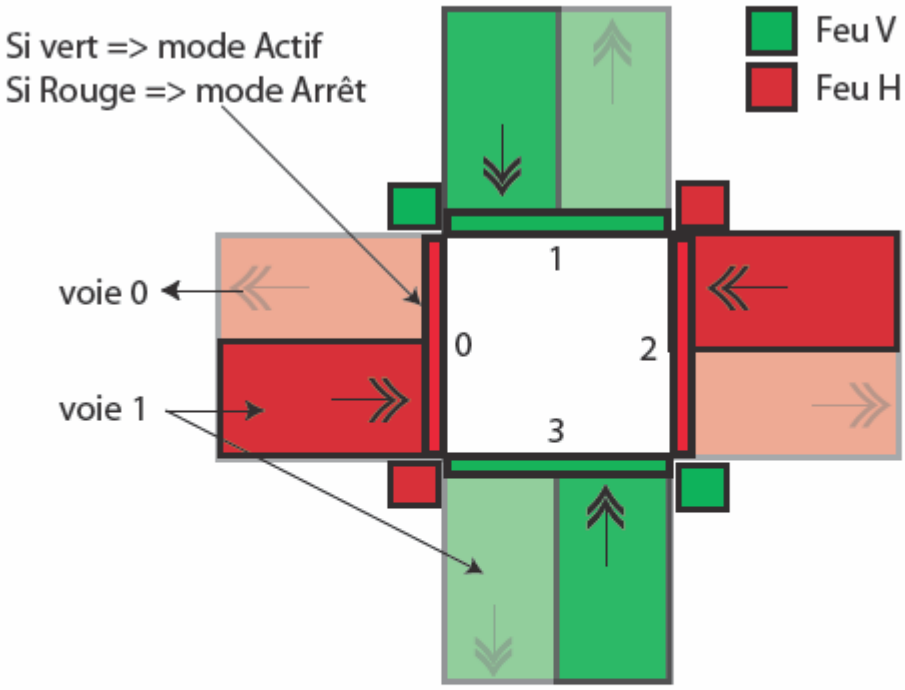
Conventions

>> Convention générale en mode création et simulation :
Pour un élément route, on repère ses deux extrémités. On associera à l'extrémité la plus haute et la plus à gauche la valeur 0 et à l'extrémité la plus basse et à droite la valeur 1.
Autrement dit, on affecte à l'extrémité (Gauche ou Haut) la valeur 0 et à l'extrémité (Droite ou Bas) la valeur 1.

>> Détermination des deux carrefours d'extrémités d'une route :
On note « carrefourD » le carrefour de début de route qui se trouve toujours soit à gauche soit en haut d'une route.
On note « carrefourF » le carrefour de fin de route qui se trouve toujours soit à droite soit en bas d'une route.

>> Repérage des voies sur une route :
Voie 0 = voie dont la circulation se fait de **droite à gauche OU de bas en haut**.
Voie 1 = voie dont la circulation se fait de **gauche à droite OU de haut en bas**.

Représentation conventionnelle pour la notation et la représentation du carrefour et des voies connectées:

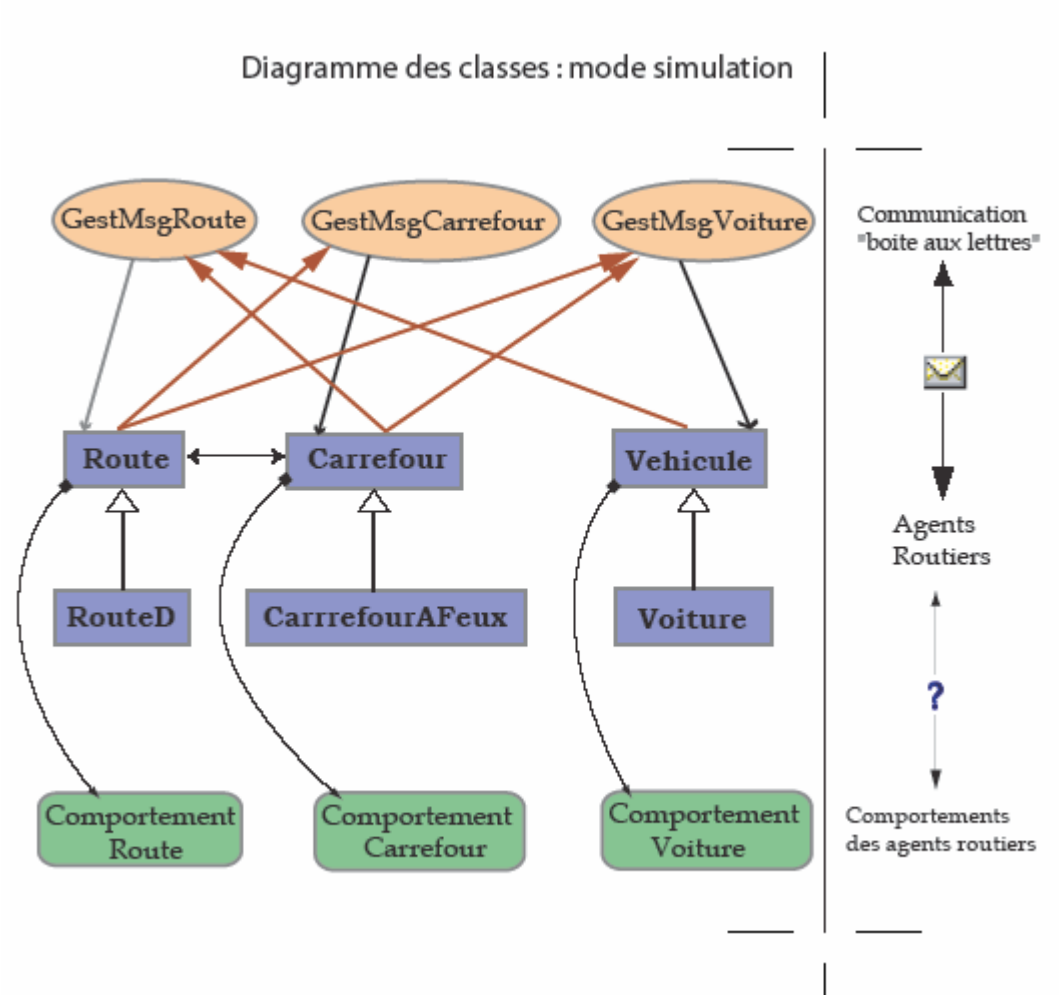


Mode Arrêt : Attente message
Si message non null => envoyer message "refus voiture"
Si message null => rien faire, se remettre en attente

Mode Actif : Attente message (soit Feu V, soit Feu H)
on aiguille les voitures si la voie à laquelle elle souhaite aller est libre.

« Sous Diagramme des classes » du mode simulation.

Dans ce diagramme, il n'est pas représenté les liens avec les autres sous systèmes ni le lien fort du village avec les éléments routiers. C'est-à-dire que le village crée et donc possède les routes, carrefours et voitures.

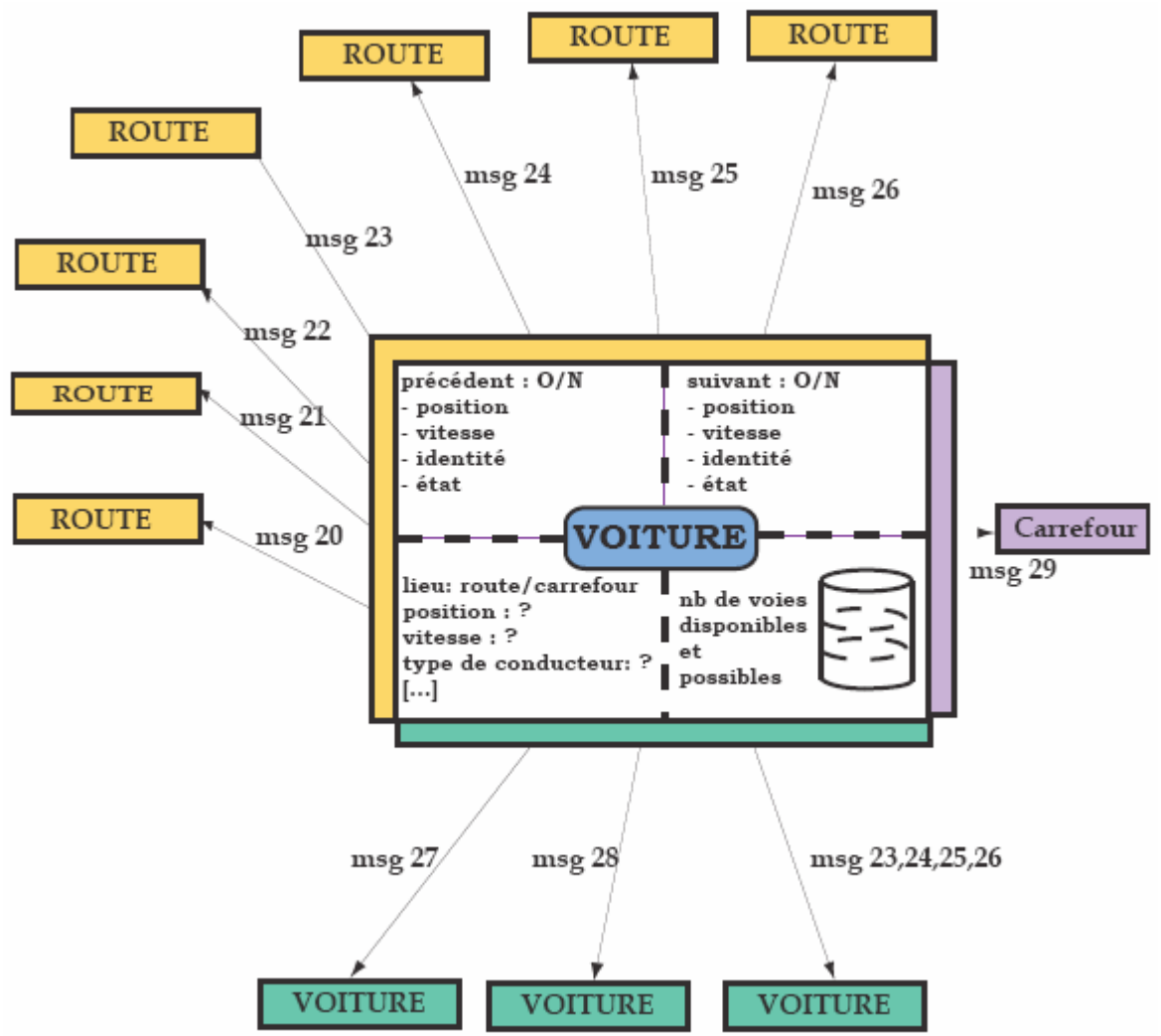


ANALYSE COMPORTEMENTALE DES AGENTS DU RESEAU

Dans ce qui suit, nous donnons les aboutissements de notre réflexion. Nous analysons chaque agent fondamental pour le réseau routier ainsi que leurs actions (messages envoyés) et comportement.

ANALYSE APPROFONDIE DE CHAQUE ENTITE DU RESEAU ROUTIER : COMPORTEMENT-ROLE-ACTION-STRUCTURE

L'AGENT VOITURE



msg 20	«entrer sur carrefour »	**msg 26**	« indiquer ma position »
msg 21	« je klaxone»	**msg 27**	« avance-toi »
msg 22	« je change de files »	**msg 28**	« arrête-toi »
msg 23	« je m'arrête» *(direction, position…)*	**msg 29**	« random choix voies»
msg 24	« J'avance » *(direction, position…)*		
msg 25	« Je suis mort » *(date, position…)*		

Remarque : (23 & 26 sont fusionnés) , (24 & 26 sont fusionnés)

Role de la voiture

Son rôle se résume à : Avancer sur la route (plusieurs méthodes privées et/ou publiques permettent de gérer son déplacement).
La voiture est donc en fait un **capteur** et un **effecteur** sur l'environnement. En effet, il détecte les signaux provenant de l'extérieur, effectue un traitement de ces signaux en fonction de son comportement et renvoie des signaux pour fournir des informations sur ses actes.

La mémoire de la voiture

- La voiture doit savoir sur quoi elle roule (un carrefour, une route, une autoroute ?), elle sait à quelle vitesse elle roule, son type de conduite, son rapprochement vis-à-vis d'un carrefour où d'un autre véhicule (informations données par la route en interaction avec les véhicules).

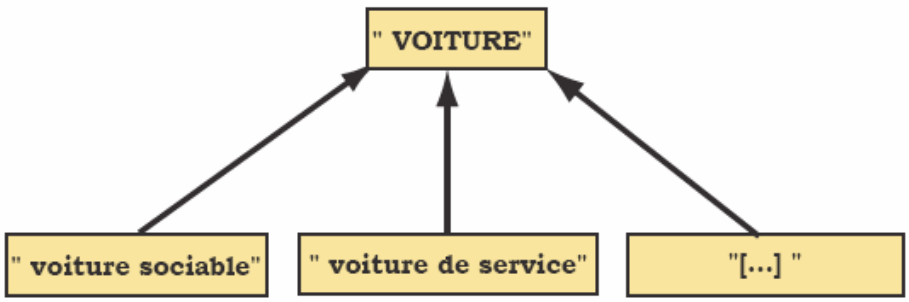
- Lorsqu'une voiture arrive à un carrefour, elle doit savoir le nombre de voies disponibles et quelles sont parmi celles-ci ceux qui sont possibles (priorité, interdiction…). En effet, cela permet au véhicule d'effectuer un choix, au hasard ou déterministe : c'est à nous de gérer.
  Cela permet donc au véhicule d'avoir un comportement vis-à-vis du choix de la prochaine route à prendre à un carrefour. Ce choix est pris en compte par le carrefour qui procèdera au « scheduling » des véhicules en fonction des priorités, du type de carrefour, et enfin du choix du véhicule !

- Enfin, deux autres parties de mémoires sont réservées pour l'interaction entre véhicules. En effet, la voiture pourra poursuivre un véhicule, se faire poursuivre ou bien les deux.
  Avoir une mémoire associée à ce type de données permettra de gérer un algorithme de comportement du véhicule.
  En fonction de la distance qui la sépare d'un autre véhicule, chaque agent « voiture » aura son comportement.
  Bien entendue, dans un premier temps, il s'agira de coder un comportement simple pour tous les véhicules que l'on pourra qualifier de « standard » ou de « sociable ». C'est-à-dire qu'ils respecteront le code de la route. Cela permettra dans un premier temps d'observer le bon fonctionnement du réseau avant d'introduire des évènements particuliers.
  Ainsi, par la suite on pourra créer des agents « voiture » dérivées avec leur propre comportement. On pourrait aussi parler de « réaction à l'environnement ».

Le comportement

On a précédemment esquissé une partie du comportement.
On aura des comportements différents suivant le type de conducteur. Le premier conducteur ou agent « voiture » que l'on codera sera sociable, obéissant à des règles stricte de conduite.
L'évolution de différents caractères se fera au fur et à mesure.

Afin d'évaluer les comportements, on se basera sur l'analyse de quelques scénarios possibles illustrant le comportement des agents.



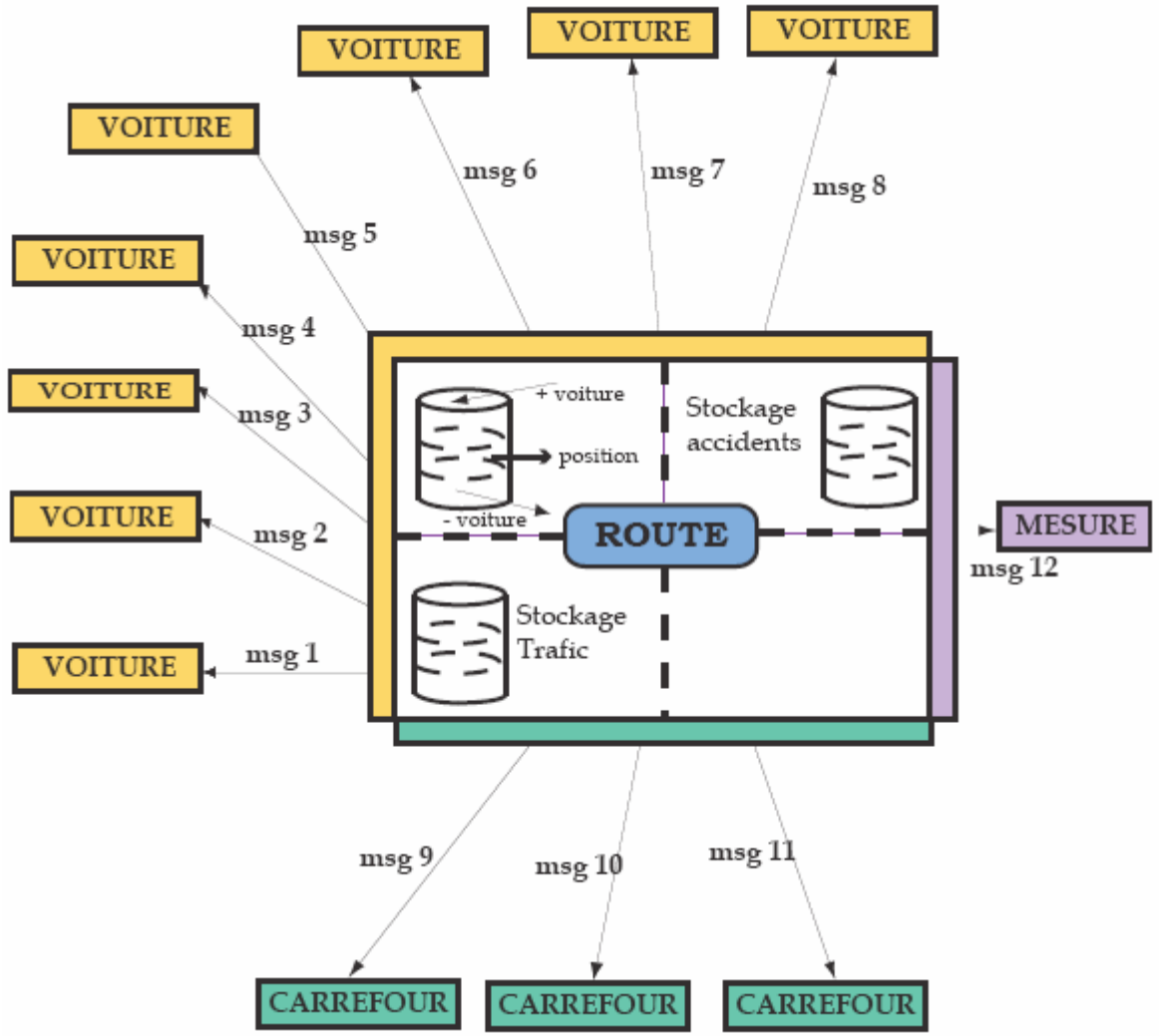
Compétences possibles pour une voiture

Accélérer,
Ralentir,
Tenir_vitesse_constante,
Connaître la route sur laquelle elle roule,
Connaître le véhicule qui la précède ou qui la suit.
Heurter une voiture
Changer son temps de réaction
Modifier la vitesse
[…]

Caractéristiques :

Temps de réaction,
Vitesse,
Taux de chance d'heurter une voiture
Taux de chance de tomber en panne
[…]

ROLE DE LA ROUTE



msg 1	« tu approches d'une voiture »	**msg 7**	« tu approches d'un carrefour »
msg 2	« une voiture s'approche de toi »	**msg 8**	« … ?.... » *à determiner*
msg 3	« tu approches d'une voiture & une voiture s'approche de toi »	**msg 9**	« donner voiture ? »
msg 4	« Tu peux entrer sur la route »	**msg 10**	« faire entrer une voiture »
msg 5	« Boom »	**msg 11**	« refuser voiture »
msg 6	« Arret obligatoire »	**msg 12**	« envoyer mesures »

Mémoire de la route :

- La route a connaissance des voitures qui se trouvent dessus, elle gère une liste dynamique. Le fait qu'une voiture rentre sur la route résulte d'une négociation entre le carrefour et la route. Le carrefour demandera à la route si elle veut bien de la voiture. Ainsi, quand une voiture entre sur la route, la route l'enregistre dans une liste.

Quand une voiture quitte une route, cette dernière envoie un signal au carrefour qui suit (msg 9) et la voiture est effacée de la liste si le carrefour l'accepte, sinon la voiture est condamnée à rester sur place.

- La route possède aussi une mémoire pour stocker les accidents (si elle envoie un signal « boom » [msg5], elle enregistre l'identité de la voiture, le lieu de l'accident, la date dans une liste).

- Enfin, la route possède une mémoire pour enregistrer le trafic. Cette mémoire est limitée : Tableau contenant 24 entrées (2 colonnes et 24 lignes) qui stocke le nombre de véhicule par heure.
- Chaque heure, un compteur d'enclenche ; et au bout d'une heure, la valeur stockée dans le compteur s'enregistre dans le tableau).
- Au bout de 24 heures, la route envoie ses valeurs à l'agent « mesure ». Ce dernier enregistre, fait des calculs, des moyennes, etc. …, et stocke les valeurs dans une mémoire (fichier).
- Puis, le tableau est réinitialisée.

Remarque : à tout moment, si un signal de l'agent mesure lui demande des données, la route lui envoie ses mesures, puis continue son cycle.

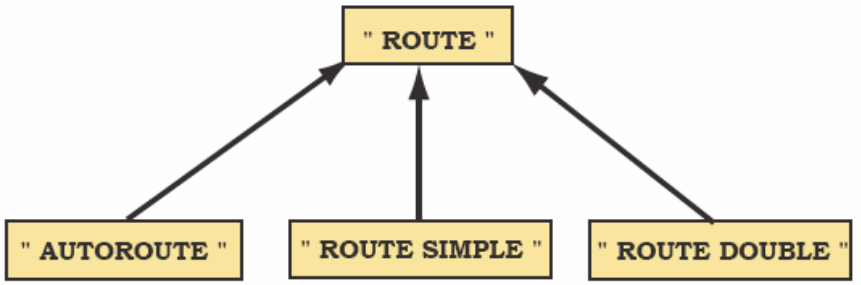
Comportement de la route (agent « Route »)

Le comportement de la route découlera des différents scénarios possibles. Chaque agent « route » aura son propre comportement. On pourra ainsi distinguer le comportement d'une route de celui d'une autoroute.
Le comportement correspondra donc aux actions réalisées en fonction des messages reçus par l'environnement extérieur (voitures, carrefour, conditions atmosphériques…). Elle générera des messages (ses messages) aux éléments avec qui elle interagi pour leur donner une image de l'environnement qui les entoure. En aucun cas ce ne sont des ordres !! (Excepté le msg6, nécessaire pour le fonctionnement du réseau routier). En effet, sur réception d'un message, chaque agent (voiture, ou carrefour…) est libre d'en faire ce qu'il veut, et cela dépend donc du comportement de l'agent.
Par exemple, une voiture sociable obéira au code de la route et une voiture folle n'obéira pas. Donc, si la route envoie un message pour annoncer à la voiture qu'elle se rapproche d'une autre voiture, elle réagira différemment selon qu'il s'agit de la voiture sociable ou de la voiture folle ! La voiture folle risque de provoquer un accident tandis que la voiture sociable va respecter la distance de sécurité.

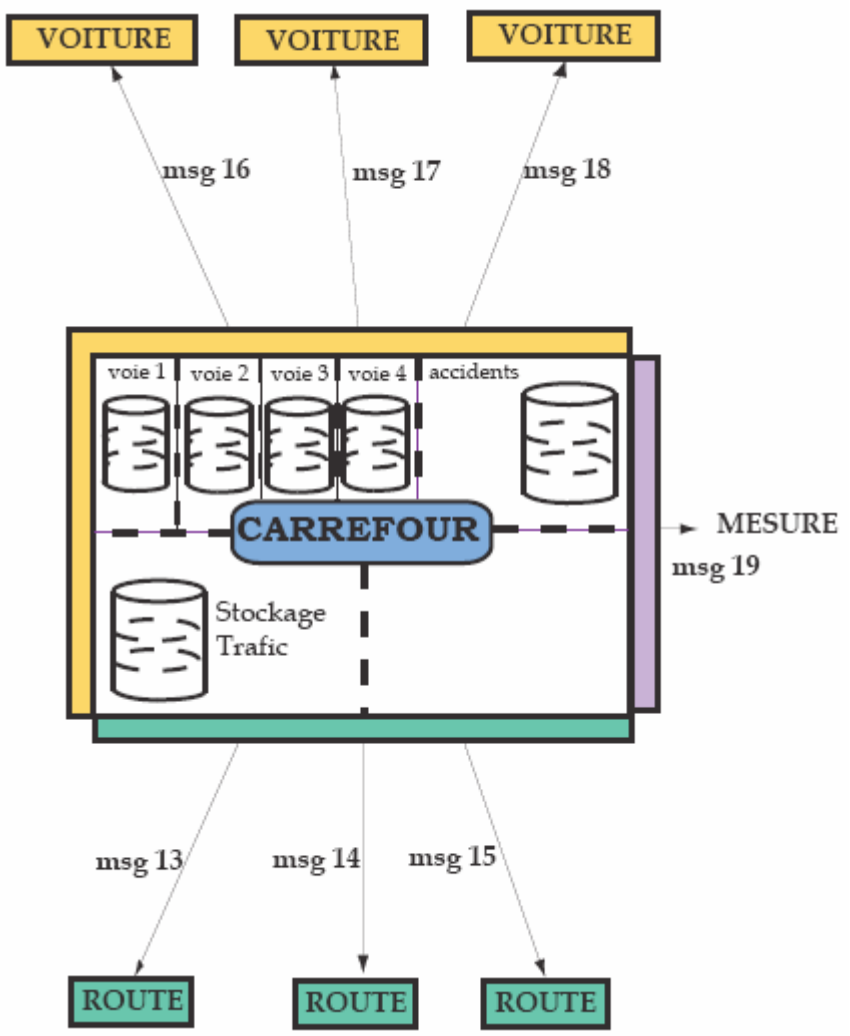
Pour chaque agent, si il ne reconnaît pas le message, il ne le prendra pas en compte.
Un agent « route simple » et un agent « autoroute » n'auront pas le même comportement vis-à-vis des possibilités de circulation.

Héritage

Les agents « route simple », « route double sens », « autoroute », etc. …, hériteront de l'agent « Route ».
Donc, l'agent route aura des caractéristiques de base et un comportement de base. Un agent « route simple » n'aura qu'à spécifier quelques paramètres (taille, vitesse…) pour être opérationnel et utilisable.



ROLE DU CAREFFOUR



msg 13	« refus de voiture »
msg 14	« accepter voiture »
msg 15	« demander de faire rentrer une voiture sur la route»
msg 16	« voie possible »
msg 17	« quelle voie ? »
msg 18	« Arret obligatoire »
msg 19	« envoyer mesure »

Mémoire de l'agent « Carrefour »

- Chaque liste (voie 1 .. 4) correspond à une voie (donc à une route). Si le carrefour a trois voies connectées, il y aura alors trois listes de crées (plus exactement : d'instanciés).

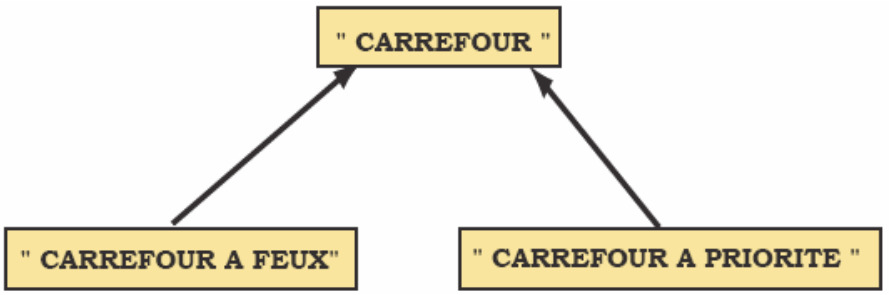
- On a un tableau pour garder le nombre de voitures par heure. Donc on a 24 lignes de tableau pour une journée. Ces données sont transférées à l'agent « mesure » lorsque la ligne 24 est remplie.

- Au bout de chaque heure, on consulte le nombre de véhicules qui sont entrées dans le carrefour. C'est-à-dire qu'on a un compteur qui dès que l'on ajoute une voiture à une liste, s'incrémente. Cette valeur est alors retournée dans le tableau.

- Enfin, on a une liste qui stocke les accidents.

Comportement de l'agent « Carrefour »

Le comportement d'un carrefour dépend du type de carrefour. On a don deux carrefours qui héritent d'un agent carrefour. Il nous suffit de gérer les listes, c'est-à-dire de définir comment choisir quelle voiture retirée d'une liste, pour quelle voie, etc. …
Gérer le comportement d'un carrefour reviendra donc à gérer l'ordonnancement des voitures selon leur destination et l'état du carrefour (feux rouges, verts… ou priorité à droite, etc. …).



CONCLUSION

Le langage Java et l'approche objet nous étaient inconnus il y a encore 6 mois. Le programme présenté dans ce rapport est le fruit de plusieurs mois de travail dont le volume horaire dépasse largement celui fixé par les enseignements. Ceci résulte d'une volonté forte de réaliser un développement logiciel réfléchi et justifiant l'utilisation de techniques de spécification afin d'acquérir une expérience dans ce domaine et de parfaire nos connaissances aussi bien du langage Java que de Génie Logiciel.

Même si le produit fini présente quelques situations dans lesquelles sont exécution ne répond pas totalement au cahier des charges que nous nous étions fixés, nous sommes satisfaits du résultat. En effet, pendant le développement, nous avons été amenés à découvrir des aspects nouveaux de programmation (synchronisations parfois complexes d'objets de stockage de données, multi-threading massif, généricité, composants SWING et AWT) mais aussi des aspects qui relèvent de techniques de conception (intérêt du double pointage, encapsulation, polymorphisme, généricité et héritage, visibilité).

ANNEXES