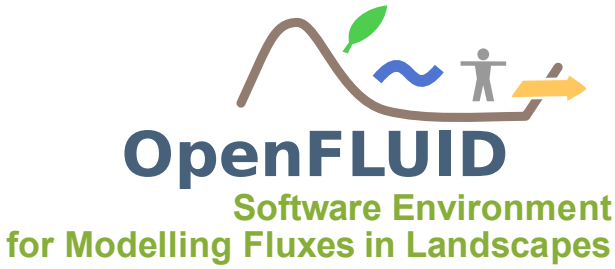


TP9 : Trafic routier dans les rues de Manhattan

Objectifs:	Définir les concepts d'un modèle pour une simulation de circulation routière
Pré-requis:	TP1 à TP7

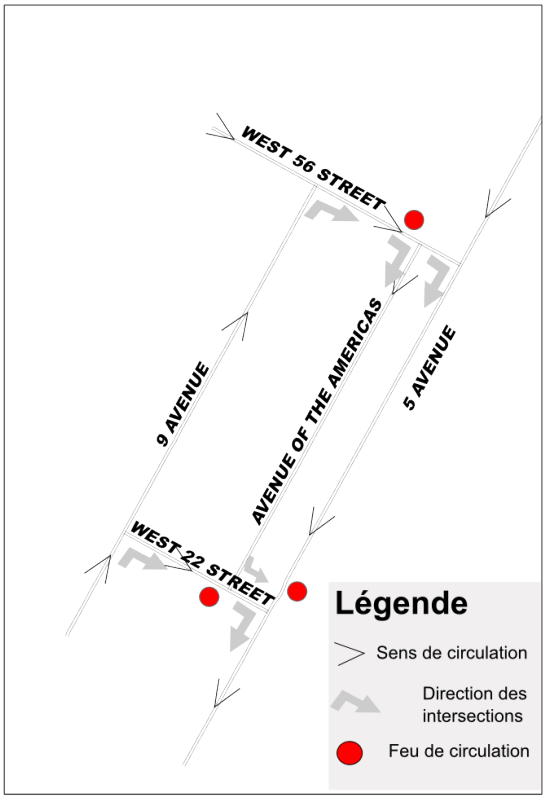
Cet exercice est un exemple de résolution d'une problématique de modélisation à travers le formalisme de la plateforme OpenFLUID. Nous allons vous montrer quelles sont les principales étapes à réaliser pour répondre à cette question. Pour cela, nous allons nous éloigner de la modélisation hydrologique pour répondre à une problématique de modélisation des flux de voitures dans les rues de Manhattan, NY.

1 Présentation de la problématique

Nous voulons modéliser le flux de voitures sur des routes de Manhattan. Notre espace à modéliser est constitué de routes reliées entre elles et dont certaines sont régulées par des feux de signalisation qui arrêtent (feu rouge) ou laissent passer le flux de voitures à la sortie des routes (feu vert). Chaque route contient un nombre de voitures (on partira du principe, certes peu crédible, que les routes peuvent stocker un nombre infini de voitures) et transfère un nombre de voiture par pas de temps (si une route à un taux de transfert de 30, elle peut, par pas de temps, transférer au maximum 30 voitures ou moins si son stock est inférieur à cette valeur). Enfin, toutes les routes sont à sens unique, certaines routes seront dotées d'un système de "générateur" de voitures : ce seront des routes "sources". Nous allons nous intéresser à la dynamique du nombre de voitures sur chaque tronçon de route à chaque pas de temps de la simulation (les voitures ne sont pas individualisées : nous ne suivons pas le cheminement d'une voiture au cours de la simulation). Ainsi à chaque pas de temps, les routes stockent et transfèrent un nombre de voitures en fonction des signaux qu'elles reçoivent des feux de signalisation.

Le schéma suivant montre le domaine que nous allons modéliser : il est constitué de 11 segments de routes et de trois feux de circulation. Pour simplifier le problème, nous imposons les règles suivantes :

- Toutes les routes sont à sens unique,
- la gestion des intersections se fait sans division du flux de voiture : à une intersection donnée, toutes les voitures vont dans la même direction imposée (à gauche, à droite ou tout droit),
- les feux de circulation n'ont que deux états (vert ou rouge),
- pour un feu donné la durée d'un état (rouge et vert) est la même,
- un feu est connecté à un seul segment de route.



2 Conceptualisation du modèle

Maintenant que nous avons pris connaissance de la problématique, nous allons la décomposer en différentes parties.

Après avoir lu le texte, identifiez les différentes unités spatiales qu'il faut modéliser :

Pour chaque unité identifiée, pouvez vous décrire ses propriétés ainsi que ses comportements :

Nom de l'unité	Propriétés	Comportements

Pour chaque unité décrite, nous allons y attacher une fonction de simulation. Cette fonction de simulation va permettre, à chaque pas de temps, de gérer les propriétés et les comportements des unités.

En quelques lignes, décrivez le comportement de chaque fonction de simulation qui est attachée aux unités que vous avez identifiées :

Nous allons maintenant vous proposer une solution possible pour modéliser cette problématique de flux de voitures à travers le formalisme de la plateforme OpenFLUID...

3 Approche conceptuelle choisie

L'approche conceptuelle que nous avons choisie repose sur le schéma suivant : nous avons identifié des unités *feux de circulation* que nous appelerons **Traffic Light Unit - TLU** et des unités *tronçons de routes* que nous appelerons **Road Unit - RU**. Pour chacune de ces unités, nous allons développer une fonction de simulation. Pour les unités **TLU**, la fonction **training.trafficlight.state** permettra de gérer l'état de ces unités (passage entre les différents états rouge et vert) au cours de la simulation. Pour les unités **RU**, la fonction **training.road.traffic** va permettre de gérer les signaux reçus des feux pour autoriser ou non le transfert des voitures, gérer la variation de stock ainsi que le taux de transfert. De plus la fonction pourra recevoir en entrée des quantités de voitures générées par un **générateur** OpenFLUID.

3.1 Les unités Traffic Light Unit - TLU

Les unités TLU vont posséder 3 attributs :
- Un identifiant numérique unique : SELF_ID,
- un attribut d'état booléen (true = vert ; false = rouge) : state,
- un attribut indiquant la durée de l'état : duration.

3.2 Les unités Road Unit - RU

Les unités RU vont posséder 3 attributs :
- Un identifiant numérique unique : SELF_ID,
- un attribut indiquant le stock initial de voitures : stockini,
- un attribut indiquant le taux de transfert maximum : transfervalue.

3.3 La fonction de gestion de l'état des TLU : training.trafficlight.state

Cette fonction va gérer les changements d'état des TLU en fonction des attributs de ces unités. Elle va nécessiter deux données d'entrée (inputdata) des unités TLU :
- La donnée *state* qui indique l'état de la TLU (false ou true),
- la donnée *duration* qui indique la durée de l'état de la TLU.

Elle va produire deux variables à chaque pas de temps :
- Une variable indiquant l'état de l'unité à chaque pas de temps : training.TLU.S.state,
- une variable indiquant à quel pas de temps aura lieu le prochain changement d'état : training.TLU.T.changetime.

Note: Le code source de la fonction *training.trafficlight.state* est contenu dans le fichier */home/openfluid/formation/datasets/src/TP9/training.trafficlight.state/TLUFunc.cpp*.

3.4 Le générateur de voitures

OpenFLUID propose différents types de générateurs de données (aléatoire, interpolation, injection). Celui que nous allons utiliser injecte à chaque pas de temps un nombre déterminé de voitures aux RU connectées au générateur.
Le générateur va produire, à partir des données d'entrée, la variable *training.RU.car.source*.

Note: Les données injectées sont contenues dans les fichiers suivants */home/openfluid/formation/datasets/TP9/ carsource.txt, carsources.xml, carzero.txt et RUcardistri.dat*.

3.5 La fonction de tranfert sur les RU : training.road.traffic

Cette fonction va gérer les variations de stock et les valeurs de transfert du nombre de voitures sur les RU à chaque pas de temps. Elle va également récupérer la variable *training.TLU.S.state* produite par la fonction *training.trafficlight.state* pour connaître l'état de l'unité TLU connecté à l'unité RU (si une RU n'est connecté à aucune TLU, on autorise tout le temps le transfert). De plus, certaines RU peuvent être connectées à un générateur de voitures ; il faut donc que la fonction reçoive cette entrée.

La fonction va nécessiter deux données d'entrée (inputdata) des unités RU :
- La donnée *stockini* qui indique le nombre initial de voitures sur la RU,
- la donnée *transfervalue* qui indique le taux de transfert maximum autorisé pour une RU.

Elle va pouvoir utiliser deux variables produites si celles ci sont présentes (USED_VAR) :
- La variable *training.RU.car.source* produite par le générateur OpenFLUID qui indique le nombre de voitures créées (les sources),
- la variable *training.TLU.S.state* produite par la fonction *training.trafficlight.state* qui indique l'état de l'unité TLU.

Elle va produire deux variables :
- La variable *training.RU.S.stock* qui est le stock de voitures pour une RU à chaque pas de temps,
- la variable *training.RU.T.transfert* qui est le nombre de voitures transférées pour une RU à chaque pas de temps.

Note: Le code source de la fonction *training.road.traffic* est contenu dans le fichier */home/openfluid/formation/datasets/src/TP9/training.road.traffic/RUFunc.cpp*.

4 Construction du domaine

Note: La donnée shapefile *NY_street_pw9_wgs84.shp* est contenue dans le dossier */home/openfluid/formation/datasets/TP9/shapefiles/*.

Nous allons maintenant nous intéresser au domaine de notre exemple. Vous pouvez visualiser le shapefile à l'aide du logiciel SIG QGIS ou à l'aide du Mapview de OpenFLUID-builder. Protitez en également pour regarder la table attributaire de la couche *NY_street_pw9_WGS84.shp* pour observer les liens de topologie entre les différentes RU à l'aide des colonnes suivantes :
- SELF_ID : identifiant numérique de l'unité,

- DownRU : identifiant numérique de l'unité aval,
- ProcessOrd : ordre de traitement (process order) de l'unité.

Note: Il n'y pas de données shapefiles concernant la position des feux de circulation TLU.

La topologie des unités TLU et RU peut être visualisée grâce au fichier */home/openfluid/formation/datasets/TP9/domain.fluidx*.

5 Création du modèle

Nous allons maintenant créer notre modèle en enchainant les deux fonctions de simulation et le générateur. L'odre d'appariemment des différentes parties est important. Par exemple, nous savons que la fonction *training.road.traffic* nécessite des variables issues du générateur et de la fonction *training.trafficlight.state*.

Le modèle peut donc être écrit de la sorte :

```
<?xml version="1.0" standalone="yes"?>
<openfluid>
  <model>
    <function fileID="training.trafficlight.state">
    </function>

    <generator varname="training.RU.car.source" unitclass="RU" method="inject">
      <param name="distribution" value="RUcardistri.dat"/>
      <param name="sources" value="carsources.xml"/>
    </generator>

    <function fileID="training.road.traffic">
    </function>
  </model>
</openfluid>
```

Nous devons également définir les fichiers *run.fluidx*, *output.fluidx*. Vous pouvez visualiser le contenu de ces fichiers dans le dossier */home/openfluid/formation/datasets/TP9/*.

6 Lancement de la simulation

Désormais tout est prêt pour lancer notre simulation. Créer le projet OpenFLUID *NY_model* à l'aide de OpenFLUID-builder dans le dossier /home/openfluid/formation/projects/TP9 en important les données présentes dans le dossier */home/openfluid/formation/datasets/TP9/*.

Lancez la simulation. Vous pouvez visualiser les résultats dans le dossier */OUT* de votre projet pour chacune des unités TLU et RU.

7 Visualisation des résultats sour Google Earth

Nous allons agrémenter un peu notre simulation et visualiser les résultats de la simulation sous Google Earth. Pour cela, téléchargez sur le Market Place *Formation2012* les fonctions *utils.export.spatialdomain-vars.kml-anim* et *utils.export.spatialdomain-vars.kml-plot*.

Note: Pour plus d'informations sur ces deux fonctions, lisez le document *Utilisation de Google Earth pour visualiser des résultats de simulation*.

Ce projet va contenir les deux fonctions supplémentaires permettant la visualisation des résultats d'une simulation sous Google Earth :

- La fonction *utils.export.spatialdomain-vars.kml-plot* crée un fichier kmz qui permet de visualiser la dynamique des variables par des courbes,
- la fonction *utils.export.spatialdomain-vars.kml-anim* crée un fichier kmz permettant de visualiser la dynamique d'une variable à l'aide du curseur chronologique de Google Earth.

Complétez votre modèle pour obtenir le fichier *model.fluidx* suivant :

```
<?xml version="1.0" standalone="yes"?>
<openfluid>
  <model>
    <function fileID="training.trafficlight.state">
    </function>

    <generator varname="training.RU.car.source" unitclass="RU" method="inject">
      <param name="distribution" value="RUcardistri.dat"/>
      <param name="sources" value="carsources.xml"/>
    </generator>

    <function fileID="training.road.traffic">
    </function>

     <function fileID="utils.export.spatialdomain-vars.kml-anim">
      <param name="configfile" value="kmlanim.conf" />
    </function>

    <function fileID="utils.export.spatialdomain-vars.kml-plot">
      <param name="configfile" value="kmlplot.conf" />
    </function>

  </model>
</openfluid>
```

Lancez la simulation, puis allez dans le dossier */OUT* et double cliquez sur le fichier *kml-plot.kmz*, zoomez sur la ville de New York ; double cliquez sur une unité RU, une fenêtre s'ouvre et vous permet de voir les graphiques de dynamique des variables *training.RU.S.stock* et *training.RU.T.transfert*. Fermez Google Earth.

Allez dans le dossier */OUT* et double cliquez sur le fichier *kmlanim.kmz*, zoomez sur la ville de New York ; à l'aide du curseur chronologique (dont vous pouvez paramétrer la vitesse d'affichage à l'aide de sa boîte de dialogue), lancez l'animation pour voir la dynamique de la variable *training.RU.S.stock*.