

Rapport de Projet

**IN42**

Printemps 2010

Jonàs Brú – GI01



# Plan

Plan 2

Introduction ET but du projet 3

Implémentation choisie et modèle de données 4

Les classes 4

L’interface 6

Les modes 7

Le mode traits 7

Le mode points 8

Le mode suivi vehicule 9

Mode cumul 10

Conclusion 11

# Introduction ET but du projet

Lors que Mr Lamrous nous a présenté le projet en amphithéâtre, il m’a semblé compliqué et long a réaliser, du moins du premier abord. Cependant, au cours d’un des TDs, l’on a du mettre en place un mini-simulateur de véhicules, ou l’on enregistrait le parcours aléatoire par le quel nos petits véhicules parcourraient l’écran. J’ai trouvé le concept amusant, et pas si compliqué que je l’aurais cru à implémenter. J’ai trouvé aussi qu’il était une très bonne base pour le projet des taxis, j’ai donc décidé de m’y intéresser de plus prés à celui-ci. Après une discussion avec Mr Lamrous, qui m’expliqua le principe le but du projet, j’ai décidé de m’y lancer. Voici ce qui en est ressorti de notre discussion :

L’application devait être réalisée dans le but de répondre à une question précise, à savoir si une application permettant le suivi en temps réel des déplacements de véhicules, notamment des taxis parisiens, ainsi qu’incorporant une partie de ré-visualisation des dits déplacements passés au cours de la journée, est viable en Flash. L’opposition se ferait notamment avec les performances que pourrait avoir une telle application développée en C++/OpenGL.

J’avais donc plusieurs contraintes pour le projet, à savoir :

* L’application devait être la plus fluide possible, étant la le principal défaut de Flash par rapport a OpenGL. Je me devais donc d’optimiser les performances de celle-ci par tous les moyens.
* Les données sur les quelles je travaillais se trouvaient dans un ou plusieurs fichiers texte qui contenaient les historiques GPS des déplacements des taxis parisiens. Un fichier texte, comprenant les déplacements effectués sur une journée de 24h, c’est plus de 800.000 lignes, chacune mettant à jour la position d’un taxi à un instant donné à la seconde prés, pour plus de 5000 taxis différents en tout. Autrement dit, un flot considérable d’informations à traiter !
* Le modèle de données que je devais créer devait être assez souple pour permettre l’ajout de nouvelles positions des taxis en temps réel facilement.
* Ensuite, une seconde ou 20 taxis sont mis à jour et une autre seconde ou ce sont 200 taxis qui sont mis a jour devaient s’écrouler a la même vitesse, ce afin de ne pas avoir de décalage temporel entre les déplacements affichés et l’heure affichée.

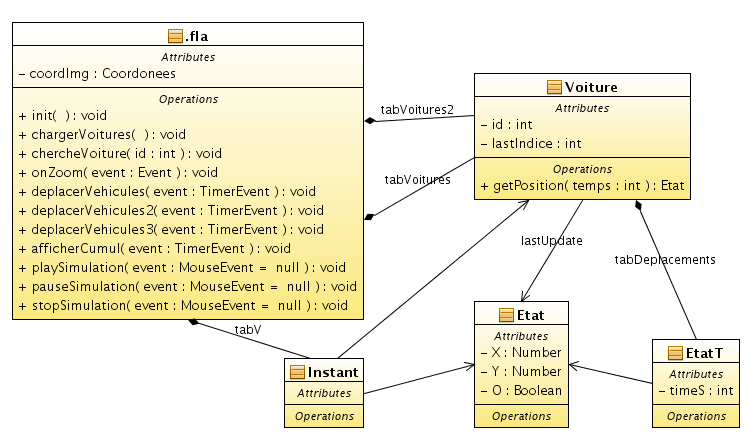
# Implémentation choisie et modèle de données

Comme vu précédemment, j’avais besoin d’un modèle de données réfléchi, car il influencerait grandement les performances de l’application et la facilité d’ajouter des informations en temps réel à celle-ci.

N’ayant jamais fait de Flash auparavant, encore moins de l’Action Script, j’ai choisi d’implémenter le projet en AS3, d’une car il se rapproche beaucoup plus de la conception objet à la quelle je suis habitué avec d’autres langages, et de deux, car je ne voyais absolument pas comment gérer un tel projet en AS2 tout en produisant un code propre et réutilisable.

## Les classes

Voici le modèle de données :



On y retrouve quatre classes en plus du code principal placé dans le .fla. Je n’y ai indiqué que les attributs et fonctions utile à la compension globale de la structure de l’application, j’ai donc omis un bon nombre d’attributs et de fonctions notamment présents dans le .fla, qui ne méritent pas d’être détaillés ici (chargements et événements, interaction avec l’UI, etc.).

En détail :

La classe de la quelle toutes les autres dépendent, c’est la Classe Etat. Celle-ci est tout simple, elle contient juste l’état d’un taxi a un moment donné, c'est-à-dire sa position X/Y et si il est occupé ou libre (O).

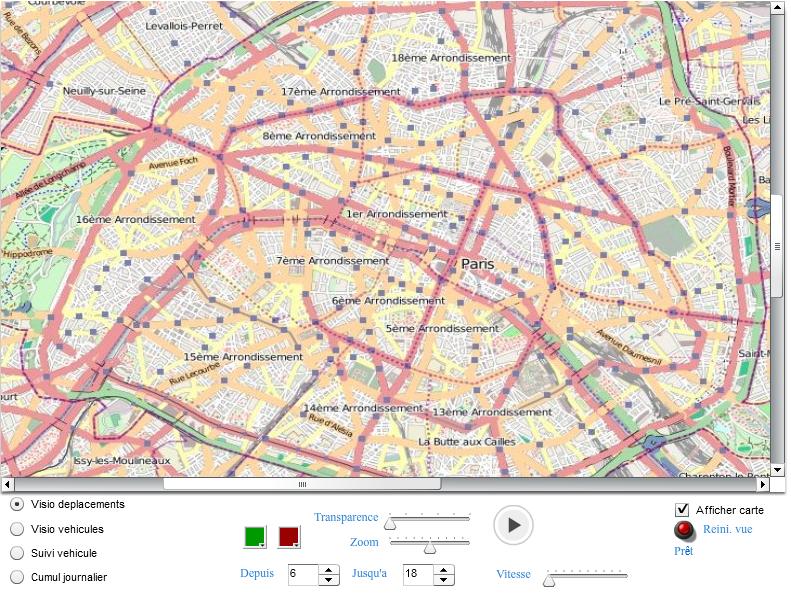
Les deux classes qui en découlent sont Instant et EtatT. La première contient un Etat et une Voiture. La deuxième un Etat et un temps, autrement dit la seconde de la journée a la quelle cet état a lieu.

Dans le .fla, on a deux tableaux importants. Le premier est un tableau d’Instants nommé tabV. Le tableau est à deux dimensions. La première dimension désigne la seconde de la journée, la deuxième tous les Instants qui se déroulent a cette seconde. En gros, tabV[200] retournera le tableau d’Instants de la 200ieme seconde, donc au bout de 3 min 20 sec, tabV[200][10] nous donnera le 10eme Instant de la 200ieme seconde de la journée. Et tabV[200][10].voiture.id nous donnera l’id de la voiture référencée par cet Instant. Cette séparation par secondes a un deux avantages, l’un c’est de pouvoir récupérer facilement et sans faire de boucle tous les taxis qui bougent à l’ieme seconde. L’autre, c’est que si l’on veut ajouter des déplacements, dans le cas du temps réel, il suffit d’ajouter des tableaux d’Instants à la fin du tableau.

Le deuxième tableau est le tableau des Voitures. Cependant, vous remarquerez qu’il est en double dans le modèle. Ceci est du au fait que tabVoitures est un Array de Voiture tout simple, qui permet de réaliser des boucles dessus si besoin facilement, et d’un autre coté tabVoitures2 est une table de hachage, avec l’id des voitures en clef, et l’objet Voiture correspondant en attribut. On peut ainsi récupérer facilement la voiture à l’id 5340 par exemple. Ce tableau sert surtout au chargement, ou l’on cherchera pour les 800.000 lignes si jamais la voiture concernée par la ligne existe déjà. L’utilisation de ce tableau fait passer le chargement de 4-5 minutes avec un Array tout simple, ou l’on doit parcourir tout le tableau pour savoir si une voiture existe, à moins de 10 secondes avec la table de hachage. On voit donc l’intérêt de l’utiliser. Cependant, certaines fonctions ont besoin de boucler sur toutes les Voitures à chaque itération, donc on doit garder le premier tableau. Cependant, même si les informations sont en double, ce ne sont que des tableaux de références, qui pointent tous les deux à chaque fois sur le même objet Voiture. On remarquera aussi en regardant le code, que j’ai utilisé les classes d’Eric Feminella pour l’implémentation de la table de Hachage, qui en interne implantent un Dictionnaire.

Ensuite, le dernier tableau on le retrouve dans la classe Voiture. Chaque voiture à un tableau d’EtatT, en somme elle a un tableau vers tous ses Etat, avec en plus l’information sur quelle seconde ils ont lieu. Ce tableau est utile dans certaines fonctions que l’on verra plus en avant.

## L’interface



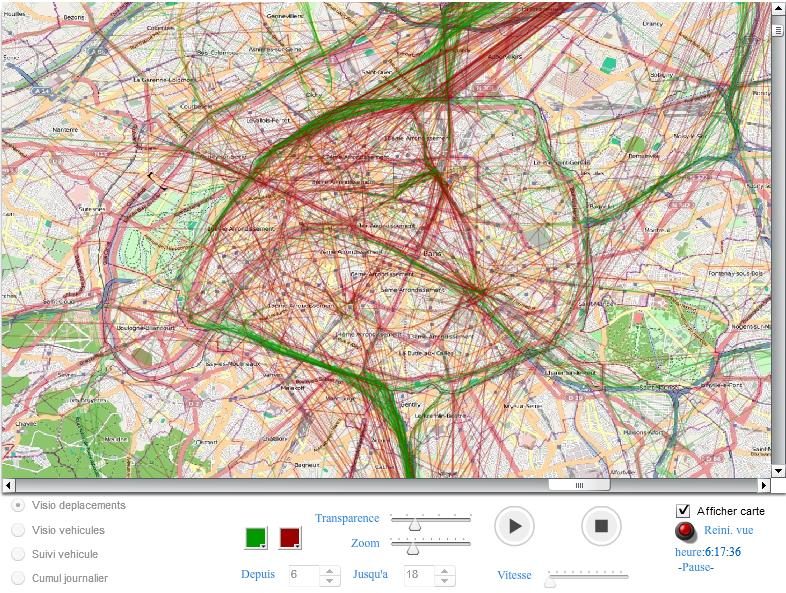
On retrouve donc la carte de Paris, et des options permettant de régler la simulation. A gauche, on a les différents modes d’affichage. On peut changer la transparence et le zoom des tracés des voitures, on peut cacher la carte de Paris pour n’avoir que les déplacements, on peut régler a partir de quelle heure jusqu'à quelle heure l’on veut faire la simulation, on peut régler la vitesse de celle-ci, on peut réinitialiser la vue si jamais on se perd en zoomant/deZoomant et finalement on peut si l’on veut changer les couleurs des tracés selon si le taxi est libre ou occupé (les carrés vert et rouge sur l’écran). Pour lancer les différentes simulations, on appuie sur le bouton Play, et par la suite on pourra mettre en Pause ou Stopper une simulation (boutons cachés tant que la simulation n’a pas débuté)

## Les modes

En tout j’ai mis en place quatre modes d’affichage, chacun utilisant des mécanismes différents pour optimiser l’affichage. Tous les modes ont plus ou moins le même mode de fonctionnement, a savoir l’on sélectionne une heure début/fin, une vitesse et on lance la simulation. En cours de route on peut changer le zoom, la transparence, l’affichage de la carte et la couleur des traits. Pour changer la vitesse on est cependant obligés d’arrêter la simulation, ré-régler la vitesse, et la relancer.

### Le mode traits

Le tout premier que j’ai mis en place, accessible par le bouton radio Visio déplacements. Pour un taxi ou a 2min30 se trouve en F, et a 6min17 se trouve en G, une fois le compteur arrivé a 6min17 un trait sera tracé de F a G (la couleur changeant selon si le taxi est occupé ou pas). Si à 10min57 on a un nouveau mouvement du taxi vers H, un trait sera tracé de G à H. Et ainsi de suite pour tous les taxis simultanément.

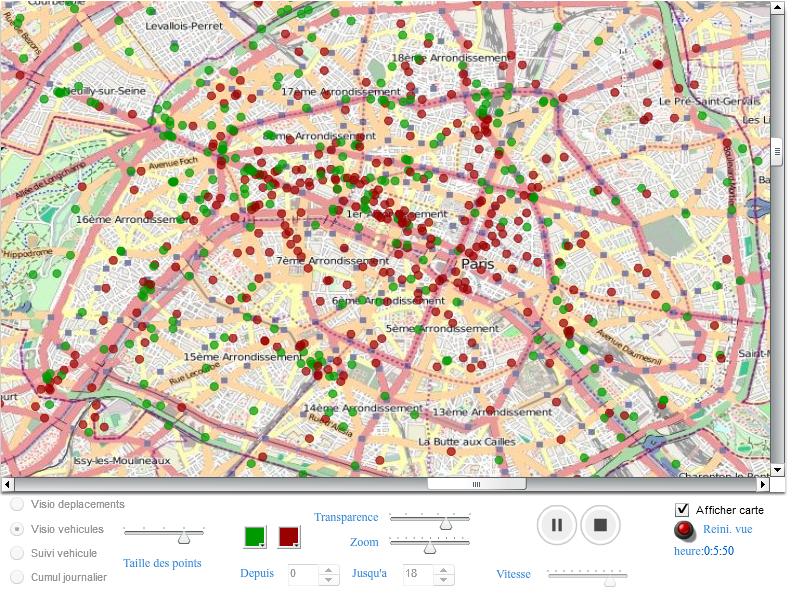


C’est actuellement le mode d’affichage le plus lent, la vitesse d’exécution ralentit au fur et a mesure de la simulation, au fur et a mesure que les traits s’accumulent sur l’écran. Pour ce mode, on utilise le tabV, en mettant en place un compteur qui chaque X temps (X plus ou moins grand selon la vitesse sélectionnée), appelle la fonction deplacerVehicules qui trace tous les traits correspondant a tous les changements d’état des différents véhicules à une seconde donnée S. Au prochain appel de deplacerVehicules, c’est la seconde S+1 qui sera traitée.

Afin de l’optimiser, on stocke dans Voiture le dernier Etat traité pour celle-ci (lastUpdate), ainsi lors d’un nouvel état pour une voiture à une seconde S, on peut récupérer son ancienne position facilement pour tracer le trait du déplacement, puis on remplace lastUpdate par ce nouvel Etat. Une autre optimisation, commune a tous les modes, est le fait d’utiliser un Shape pour dessiner les traits. En effet, Shape est l’objet AS3 le plus adapté a du dessin ou l’on recherche des performances, la ou Sprite et MovieClip sont plus lourds donc plus lents. Par contre, l’inconvénient c’est qu’une fois un objet dessiné dans notre Shape, on n’a plus aucun moyen d’agir sur ce que l’on vient de dessiner !

### Le mode points

Celui-ci au lieu de tracer la trajectoire des taxis, affiche juste la position actuelle des taxis. Cependant, si jamais un taxi a un Etat a la seconde 0 en U et un Etat a la seconde 9 en V, on vera le taxi se déplacer pendant 9 secondes de U à V. A la seconde 3, il sera donc a un tiers du chemin entre U et V. Cela permet d’avoir un mouvement fluide des points, et non pas procéder par sauts comme au mode traits.

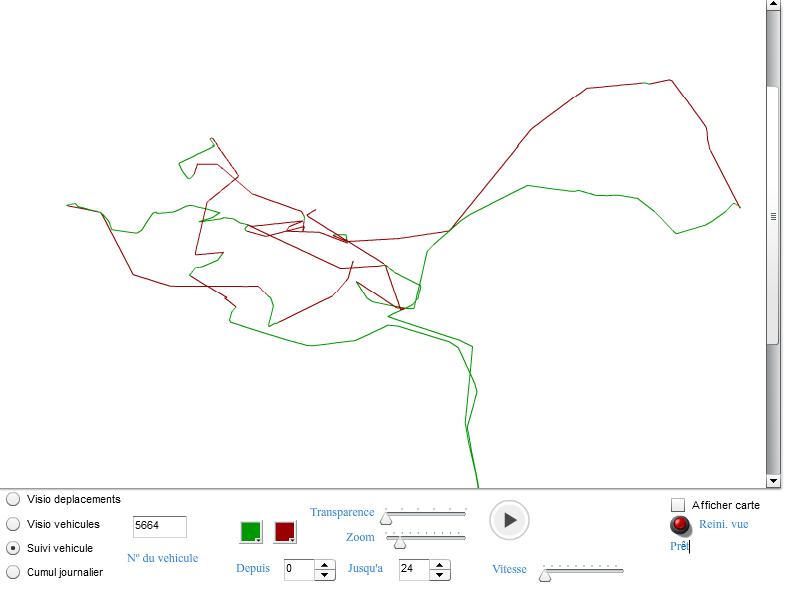


À chaque tour de boucle, on appellera la fonction getPosition de Voiture pour chaque taxi, pour connaitre l’Etat de ce taxi à cette seconde donnée. Si la seconde coïncide dans une seconde ou l’on a un Etat qui change pour le taxi, la fonction renvoie cet Etat. Si la seconde se trouve entre deux EtatsT, la fonction va calculer les cordonnées d’où devrait se trouver le taxi a ce moment la. Pour optimiser la recherche dans le tableau d’états d’une Voiture, on enregistre l’indice du tableau ou l’on a pris le dernier EtatT, afin de ne pas avoir a ré-parcourir du début le tableau d’états a l’itération suivante, comme ça on commence simplement la ou s’était arrêtés a l’itération précédente.

On peut, si on le souhaite, régler la taille des points symbolisant les taxis en cours de route.

## Le mode suivi vehicule

Ce mode permet de suivre le trajet d’un unique véhicule au cours de la journée. Vu le peu de traitements à effectuer, ce mode est actuellement le plus rapide de tous.



On sélectionne l’id du taxi que l’on veut suivre, et on lance. C’est une combinaison des deux précédents modes, a savoir l’on a l’affichage du trajet, mais celui-ci s’affiche non pas par sauts mais graduellement entre deux points au cours du temps. On utilisera les mêmes mécanismes d’optimisation que pour les deux modes précédents.

On trouve cependant un inconvénient en l’état, l’on doit rentrer à la main l’id du taxi à suite. Si on n’en a aucune idée de celui-ci (si par exemple on a repéré un taxi graphiquement), l’on ne peut donc pas le suivre (au moins d’aller fouiller dans le fichier texte des positions pour récupérer son id). Je n’ai hélas pas eu le temps d’implémenter une fonction ou l’on sélectionne son taxi en cliquant sur un des points ou traits des autres modes. Si vous voulez tester, le taxi 5664 a un parcours qui commence très tôt dans la journée et finir a minuit.

### Mode cumul

Ce mode est sensé afficher le cumul à la fin de la journée de tous les déplacements des taxis. Cependant, pour cela, il faut donc faire un premier tour de boucle de tous les déplacements, et les dessiner au fur et a mesure, pour a la fin avoir la carte avec tous les trajets. Faire ceci au lancement de l’application bloque celle-ci jusqu'à que le calcul soit fini, or vu que l’on affiche avec la même méthode que le 1er mode, cela est lent (même si l’on arrive à le faire aller plus vite en n’affichant les traits qu’une fois le traitement fini). J’ai donc préféré ne pas procéder comme ça, et faire simplement un affichage presque identique au 1er mode, ou l’on sélectionne une heure de début et de fin, et la simulation fait au plus vite pour tracer tous les déplacements dans ce laps de temps (notamment en désactivant l’affichage a chaque tour de boucle des nouveaux traits, et tout afficher une fois fini, et en sélectionnant automatiquement la vitesse la plus rapide pour les calculs)

# Conclusion

Pour conclure, sur le plan technique j’ai constaté que Flash n’est pas vraiment optimisé pour ce genre de traitements, et donne des performances inferieures à ce que l’on peut avoir avec du OpenGL. Pour la simulation et l’affichage des données d’une journée, une application en C++ est bien plus adaptée, ou alors il fait disposer d’une bonne puissance de calcul pour arriver à avoir des performances équivalentes avec la même application en Flash. Cependant, pour le temps réel, l’utilisation de Flash peut être adaptée, car le débit d’informations est finalement, assez lent (une seconde dans la réalité correspondra a une seconde dans la simulation)

Sur le plan personnel, faut dire que je me suis bien amusé à réaliser cette application, et le fait qu’il y ait une vraie question à répondre au travers de celle-ci m’y a fait prendre gout assez rapidement. Cela m’a permis de découvrir le Flash et l’AS3 que je ne connaissais pas du tout avant, et m’a permis de passer quelques soirées a me creuser la tête pour trouver des solutions efficaces pour le modèle de données, chose que personnellement je trouve plus intéressant et enrichissant que de réaliser une application plus orientée graphique, ou l’on sait a l’avance que l’on arrivera a un résultat satisfaisant si on y passe le temps nécessaire. Je regrette d’avoir pas eu le temps d’ajouter d’autres fonctionnalités a l’application (6 projets sur un semestre ça ne pardonne pas), ainsi que de laisser un bug un peu gênant sans résoudre (pas faute de n’avoir pas essayé de le résoudre !). Je suis cependant très satisfait de ce que le projet m’a apporté =)