

Encadré par : Mahjoub DRIDI

mEMBRES DU GROUPE

* pATRICK tCHEPGA
* eDWIGE DJOUKOUO
* PAPA ABDOU

CARREFOUR INTELLIGENT

Utbm | TX52

2019

SOMMAIRE

[I. Contexte du projet 2](#_Toc535475953)

[II. Etude de l’existant 2](#_Toc535475954)

[III. Réalisation 3](#_Toc535475955)

[a. Revue du fichier de données 3](#_Toc535475956)

[b. Différentes configurations 4](#_Toc535475957)

[c. Affichage des données des véhicules 5](#_Toc535475958)

[d. Gestion des collisions 5](#_Toc535475959)

[IV. Résultat obtenu 6](#_Toc535475960)

[V. Améliorations 7](#_Toc535475961)

# Contexte du projet

Dans le cadre de l’UV TX52, nous sommes amenés à travailler sur un projet de recherche pour apporter des solutions par rapport à une problématique donnée. L’objectif étant de produire une application permettant au THESARD de visualiser les résultats de leurs algorithmes sur des flots de véhicules en circulant aux alentours des carrefours.  
Les résultats sont stockés dans un fichier utilisé ensuite par l’application pour les afficher sur une carte. Pour ensuite tirer des conclusions vis-à-vis de l’ordre de passage des voitures ou la gestion de collision. L’ordre de passage ses voitures est donné par exécution d’un algorithme d’optimisation fait par le précèdent groupe de TX. Ce projet consiste à modifier une interface graphique connecté au C++ qui va gérer l’affichage des résultats obtenus grâce aux algorithmes.

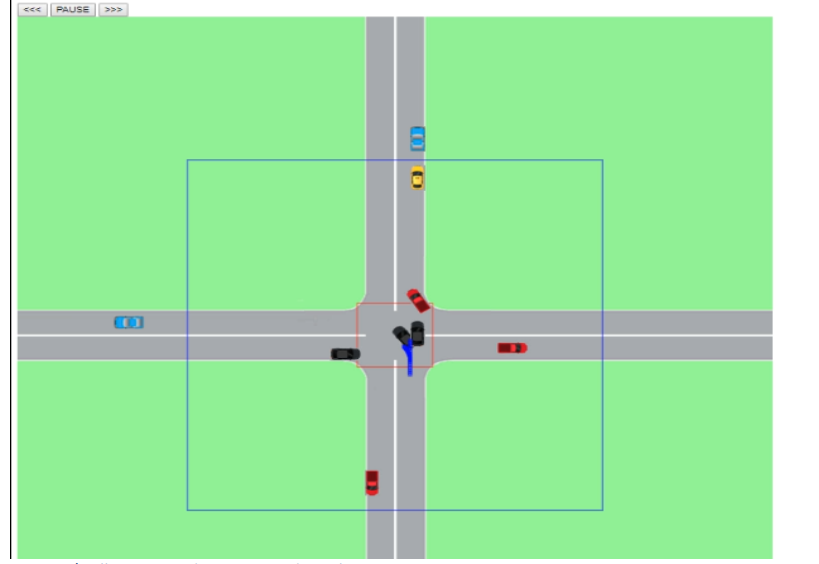
# Etude de l’existant

Il existe plusieurs plateformes pour la simulation du Traffic routier :

* <http://www.traffic-simulation.de/> qui permet de gérer les priorités, les vitesses et accélération sur un carrefour à sens giratoire
* **Anylogic** est un logiciel qui permet une visualisation claire est un moyen simple et rapide des modèles d’infrastructures routières, avec des cartes de densité présentant la congestion et des animations montrant le flux du trafic et les goulets d’étranglement.
* **SUMO :** Les simulations de circulation facilitent l'évaluation des modifications de l'infrastructure sur la route. Par exemple, l’efficacité des zones environnementales ou le contrôle des feux de signalisation peuvent être testés et optimisés dans le monde réel.

Tous ces simulateurs ne traitent pas les interactions entre les véhicules, ils permettent juste de mettre en place les infrastructures routières et de pourvoir voir les effets en temps réels. Pour l’instant il n’existe pas réellement de simulateur d’interaction autonome.

On s’est basé sur ceux que les étudiants du semestre passés ont travaillé, pour avancer.



Il avait déjà mise en place l’architecture de base pour simuler les interactions pour un cas basique. Il avait déjà fait :

* Le zonage mais les véhicules ne ralentissaient pas toujours au bon moment
* La gestion des collisions

# Réalisation

* 1. Revue du fichier de données

Le ficher de donnée a été reformaté pour permettre d’avoir plusieurs séquences de simulation. Du fait de changement de configuration le parcours du véhicule est devenu un peu plus complexe.

{

Sequences : [

[ // **sequence1**

{id : <>,speed\_zone1 : <>,speed\_zonne2 :<>,speed\_inter :<>,turn:<>, lane:<>, postInLine:”auto”}

……

],

[ // **sequence2**

{id : <>,speed\_zone1 : <>,speed\_zonne2 :<>,speed\_inter :<>,turn:<>, lane:<>, postInLine:”auto”}

……

],

]

}

Il est désormais possible d’ajouter plusieurs séquences de simulation.

* 1. Différentes configurations

On a mis en place plusieurs configurations de carrefour :

* Configuration 1 : c’est le cas basique. La carte ne contient qu’un seul carrefour, et les deux zonages de limitation de vitesse
* Configuration 2 : Dans ce cas, la carte contient 2 carrefours.
* Configuration 3 : La carte contient 4 carrefours

Pour chaque configuration, un fichier de données spécifique y est associé.

Pour les configurations 2 et 3, le parcours des véhicules devient un peu plus complexe et entraine une modification du champ de donnée **turn** dans le fichier de données.

Exemple : **turn : « Direction1 -Direction2 » , lane : « Depart ».**  Dans ce cas le parcours sera **Départ – Direction 1 – Direction2**

* 1. Affichage des données des véhicules

Notre étude étant de réaliser un carrefour autonome il est primordial de d’avoir les détails de chaque véhicule afin de réduire les différents accidents possibles. Pour ce qui de notre cas nous avons étudié différentes zones dans lesquelles les voitures se situeront. Les zones sont limitées en fonction des vitesses.

La zone 1 dans laquelle chaque véhicule acquiert sa vitesse maximale. La zone 2 dans laquelle les véhicules réduisent les vitesses afin de limiter les collisions. La zone 3 dans laquelle les véhicules acquièrent leur plus petite vitesse : C’est la traversée de la jointure.

* 1. Gestion des collisions

Dans cette partie-là, il s’agira comme son nom l’indique de gérer les collisions entre les voitures. Pour ce faire, tout d’abord il nous faut une fonction qui détecte la collision. Ici nous avons utilisé une fonction nommée aabbCollide() qui détecte la collision entre deux rectangles. Cette fonction permettra de :

* Résoudre les collisions avec des objets en mouvement rapide
* Régler le niveau de contact pour déclarer s’il y a collision ou pas (étant donné que les voitures sont représentées par des images et que le rectangle bordure de l’image n’épouse pas parfaitement les contours de la voiture elle-même).



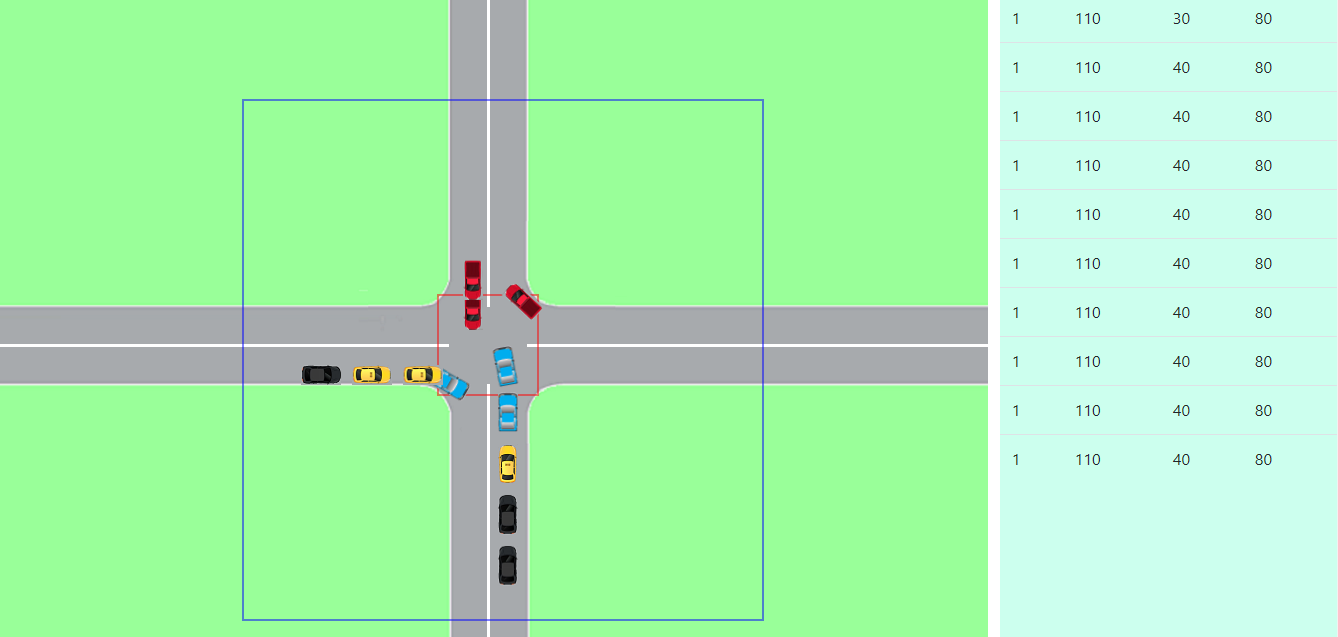
D’après l’implémentation de la fonction montrée sur l’image ci-dessus, nous pouvons constater que mis à part le fait de détecter des collisions, la fonction lance une fenêtre popup à chaque collision pour afficher les numéros des voitures qui sont entrées en collision et le nombre de collisions qu’il y a depuis le lancement de la séquence.

# Résultat obtenu

Pour réaliser la simulation de ce projet, nous avons considéré plusieurs types de configuration, à chaque type de configuration correspond un nombre de voies et de jointures différentes. Et aussi le plan est sectionné en trois parties qui sont matérialisées par les différentes vitesses des véhicules.

A cet effet nous le présentons de la manière suivante :

* Configuration 1

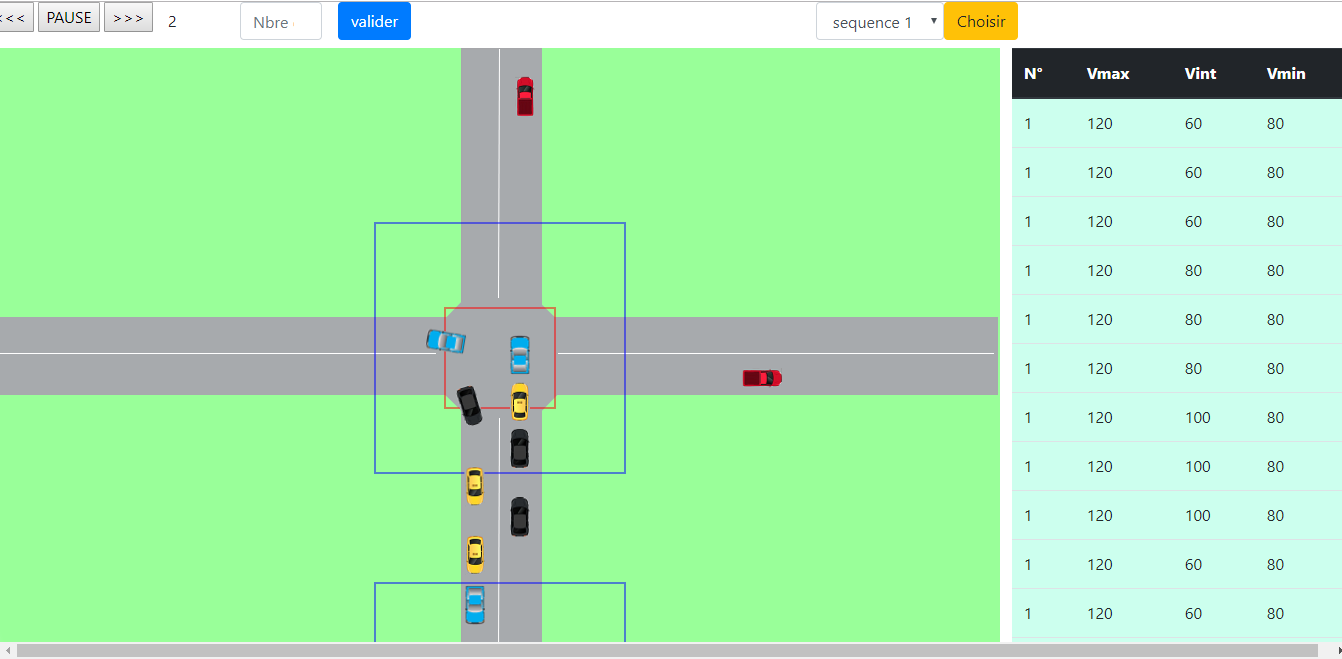


Pour ce cas nous avons deux voies possibles et toutes les voitures ont chacune la même vitesse maximale et minimale mais de vitesse intermédiaire différente. Le tableau à droite nous les détails de chacune d’elle.

* Configuration 2

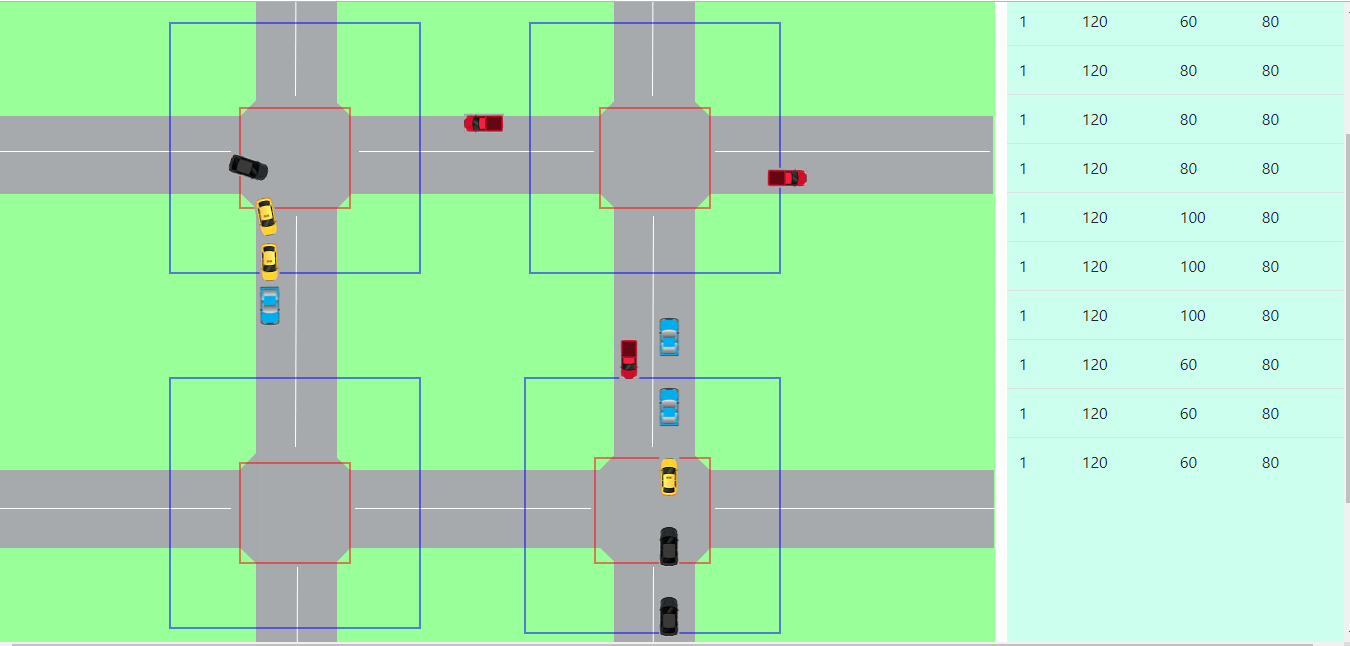
Dans ce cas nous avons deux jointures avec six voies.

Les vitesses intermédiaires étant différentes nous matérialisons les éventuelles collisions possibles entre les voitures sur la capture ci-dessous



* Configuration 3

Dans ce cas nous avons quatre jointures avec douze voies. Schématisé de la manière suivante :



# V- Améliorations

Nous avons pu retenir quelques points sur lesquels il serait bien d’apporter des changements pour mieux optimiser l’application :

* Tout d’abord le fait de pouvoir augmenter le nombre de voies (passer de 2 voies à 4 voies ensuite 6 voies etc.) : sûrement la partie qui sera la plus compliquée à faire.
* Ensuite, avoir la possibilité de changer les paramètres des véhicules directement à partir de l’interface.
* Pour la gestion des collisions, apporter des améliorations sur la fonction aabbCollide() en essayant de pouvoir retrouver le point exact de contact des véhicules lors d’une collision, régler en même temps le problème qui se pose sur la détection même des collisions (Quelquefois on observe des contacts entre les voitures et il n'y a aucun indicateur pour me dire qu'il y a eu un contact réel. D'autres fois, des collecteurs sont générés, mais aucun point de contact)
* Versionner le code : étant donné que le sujet proposé sera à nouveau disponible lors des prochains semestres, nous pensons qu’il serait bien d’utiliser un outil qui permettra de versionner le code et ainsi suivre les différentes modifications qui ont eu lieu sur le projet dans le temps et pour chaque génération.