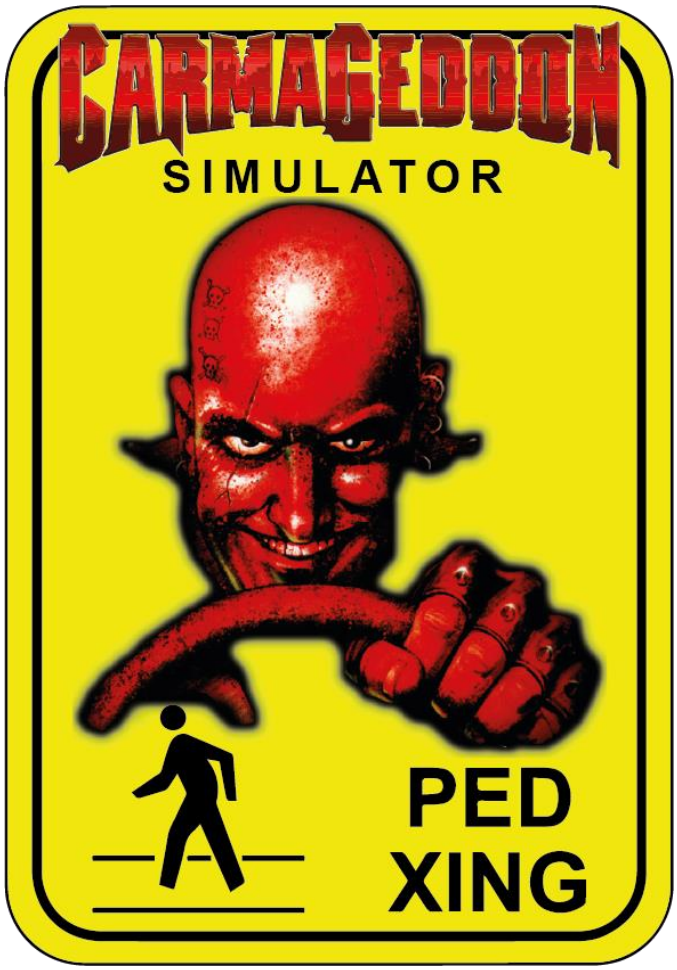
PROJET - INFORMATIQUE



RAPPORT D'ANALYSE

11 avril 2019

Samuel Mermet
Elie-Alban Lescout
Arthur Dujardin

Table des matières

1 Reformulation du sujet

Le projet de simulation de trafic routier consiste à **modéliser un carrefour** de feux de circulation et le **comportement** des voitures et des piétons interagissant sur ce nœud routier. Le comportement des voitures de cette micro-simulation doit respecter certaines règles, en particulier une distance de sécurité, entre elles et vis-à-vis des piétons, le modèle d'accélération et de ralentissement, ainsi que la vitesse limite de circulation.

L'interface de l'application développée permettra à l'utilisateur de modifier le nombre de voitures et de piétons intégrant la simulation, ainsi que le rythme des feux. Elle permettra d'afficher le déroulement de la simulation et de produire des indicateurs sur le temps nécessaire aux voitures et piétons pour traverser le nœud de transport modélisé. Ainsi, l'utilisateur sera en mesure d'**optimiser** la cadence des feux tricolores en vue de **minimiser les temps d'attente** des véhicules et piétons, en fonction des flux qu'il a défini.

2 Objectifs du projet

Nous avons développé les objectifs de Carmageddon Simulator en trois sous-objectifs, selon qu'ils correspondent à la simulation elle-même, au choix des indicateurs statistiques ou à l'interface graphique.

2.1 Objectifs généraux

Carmageddon Simulator s'adresse aux **urbanistes et gestionnaires de voiries** souhaitant optimiser le rythme des feux de circulation à l'échelle d'un carrefour. En fonction de données de flux routiers et piétons définis par l'utilisateur, l'outil que nous développons propose de générer des voitures et des piétons, de les faire interagir visuellement en tour par tour avec les feux et entre eux, afin de produire par micro-simulation des statistiques de congestion.

Dans un premier temps, le carrefour sera **réduit à sa structure la plus simple**, c'est-à-dire le croisement de deux axes de deux voies, soit un total de 8 tronçons de route autour du nœud routier. Selon l'avancée de notre projet, nous souhaitons faire déterminer à l'utilisateur le nombre de voies de chaque axe.

La simulation débutera et prendra fin au choix de l'utilisateur. Ce dernier pourra éventuellement automatiser la clôture de la simulation à l'avènement d'un accident.

2.2 Objectifs statistiques

La **congestion** se mesure par la durée de traversée de la simulation par une voiture ou un piéton. Carmageddon Simulator stockera ainsi l'ensemble de ces **valeurs de temps** et les intégrera à un rapport de congestion comprenant divers indicateurs statistiques : moyenne et écart-type, médiane et quantiles, ainsi que les extrema voire un graphe de la distribution selon la complexité souhaitée par l'utilisateur pour effectuer ses comparaisons entre simulations. Ces valeurs pourront être présentées

en unité de temps ou de vitesse. En effet, cette dernière est calculable en km/h du fait des spécifications spatio-temporelles évoquées dans ce document.

En fonction de l'avancée de notre projet, nous envisageons de **catégoriser les parcours** des voitures et piétons. Ainsi, le rapport statistique pourra différencier les véhicules selon si elles proviennent ou non de l'axe principal, et selon si elles ont tourné à gauche ou à droite ou sont allées tout droit. Pour les piétons, cette catégorisation reposera sur le **nombre de rues traversées**. Le rapport comprendra également le nombre de voitures et de piétons ayant traversé le carrefour, ainsi que la **durée totale de la simulation** et **la cause de la fin** de cette dernière (choix de l'utilisateur ou accident) et le nombre d'accident (que nous espérons réduit à 0 !).

Nous envisageons également d'ajouter un second marqueur de temps, dédié au temps d'attente au feu tricolore. En effet, cet indicateur spécifique nous semble apporter une information supplémentaire au gestionnaire.

Enfin, une piste de développement possible est le chapitrage du rapport. A chaque changement de paramètres (flux de voitures/piétons et durée des feux) de simulation par l'utilisateur, un nouveau chapitre est généré, contenant les statistiques précédemment décrites associées à cette configuration. Ce chapitrage permettra à l'utilisateur de comparer différentes simulations dans une même session.

2.3 Objectifs graphiques

L'application sera munie d'une **interface graphique** qui permettra à l'utilisateur de configurer la structure de la simulation en vue de sa création, puis de modifier certains paramètres en interaction directe : modification du flux de voitures ou de piétons sur chaque axe, rythme des feux…

Une fois la simulation débutée, une fenêtre affichera les différents objets mobiles sur un fond représentant le carrefour (voies routières et piétonnes, dont passages piétons) et les feux tricolores. Les piétons se déplaceront sur des trottoirs adjacents aux routes, dédiées aux véhicules. Cette interface permettra à l'utilisateur d'observer, en tour par tour ou temps réel simulé, le **déroulement** de la simulation.

Enfin, l'utilisateur pourra mettre fin à la simulation.

3 Modélisation

3.1 Premières hypothèses

3.1.1 Axe principal et axe secondaire

Notre nœud est généré par le **croisement de deux axes**. Pour les distinguer, afin de configurer des nombres de voies et des flux variables, nous utiliserons les termes d'axe principal et d'axe secondaire. Le nombre de voies maximum sera toujours attribué à l'axe principal.

Par convention, nous associerons graphiquement l'axe principal à la longueur de l'écran (axe Est-Ouest) et l'axe secondaire à sa largeur (axe Nord-Sud).

3.1.2 Coordination des feux tricolores

Dans un souci de réalisme, nous considérons que les **feux sont coordonnés**. Ainsi, les deux feux de l'axe principal affichent la même valeur, en opposition avec les feux de l'axe secondaire. De la même façon, les feux piétons sont en opposition entre axes et ils sont appareillés (avec quelques précisions de calcul évoquées ci-après) avec les feux des voitures du même axe.

Cette hypothèse triviale a pour principal avantage de réduire drastiquement les risques de collision !

3.1.3 Circulation des véhicules

Les voitures roulent exclusivement sur les voies routières, sans possibilité de se retourner. Elles **respectent le code de la route** : arrêt au feu rouge, arrêt si possible au feu orange, vitesse maximale définie par l'utilisateur, distance de sécurité pouvant amener à s'arrêter. Une voiture en interaction avec une autre voiture ou un piéton génère un accident.

Une fois à l'intersection, la voiture applique la décision qui lui a été attribuée lors de sa création : aller tout droit, tourner à gauche ou tourner à droite.

S'il existe plusieurs voies dans la même direction, la voiture ne change pas de voie. En effet, elle est générée sur une voie correspondant à sa décision de tourner ou non.

3.1.4 Circulation des piétons

Dans un premier développement, les piétons circulent exclusivement sur les trottoirs et les passages piétons. Arrivé à une intersection, un piéton pourra traverser au passage piéton si son feu est vert. Il pourra également continuer son chemin en longeant la route.
Nous effectuons de plus deux hypothèses fortes : nous considérons que les piétons n'oublient pas leurs clefs (pas de demi-tour) et qu'ils ne regardent pas leur smartphone en marchant (ils peuvent se croiser sans se cogner).

Selon notre avancée, nous envisageons de permettre aux piétons de traverser la route, en dehors des passages piétons, si aucune voiture n'est engagée sur le tronçon concerné. Cette traversée se déroulerait orthogonalement à la voirie afin de minimiser le temps de présence du piéton sur cette dernière.

3.2 Fonctionnalités

Une première analyse nous a menés à considérer **trois fonctionnalités principales** :

- Configuration
- Visualisation
- Rapport Statistique

Celles-ci seront accessibles via un menu principal, accessible dès le lancement de l'application.

3.2.1 Configuration

Le panneau de **configuration** de l'application permet à l'utilisateur de configurer différents paramètres, que l'on distingue en deux types :

- Ceux qui concernent l'affichage **graphique** ;
- Ceux qui concernent la **simulation** elle-même. Parmi ces dernirs on divise entre :
  - La configuration **structurelle** ;
  - La configuration de **flux**.

- **Configuration structurelle**
  Elle correspond à des informations structurelles de la simulation : le nombre de voies par axe et sens de circulation. Ces informations doivent être définies avant le début de la simulation. Des valeurs par défaut seront définies (deux axes de deux voies en sens inverse).
  La **configuration structurelle** sera donc effectuée avant la simulation dans un panneau dédié.

- **Configuration de flux**
  Elle concerne le nombre de voitures et de piétons générés à une certaine fréquence, le rythme des feux... Ces informations peuvent évoluer au cours de la simulation.
  La **configuration de flux** pourra donc se trouver dans une colonne adjacente à la visualisation.

Visualisation

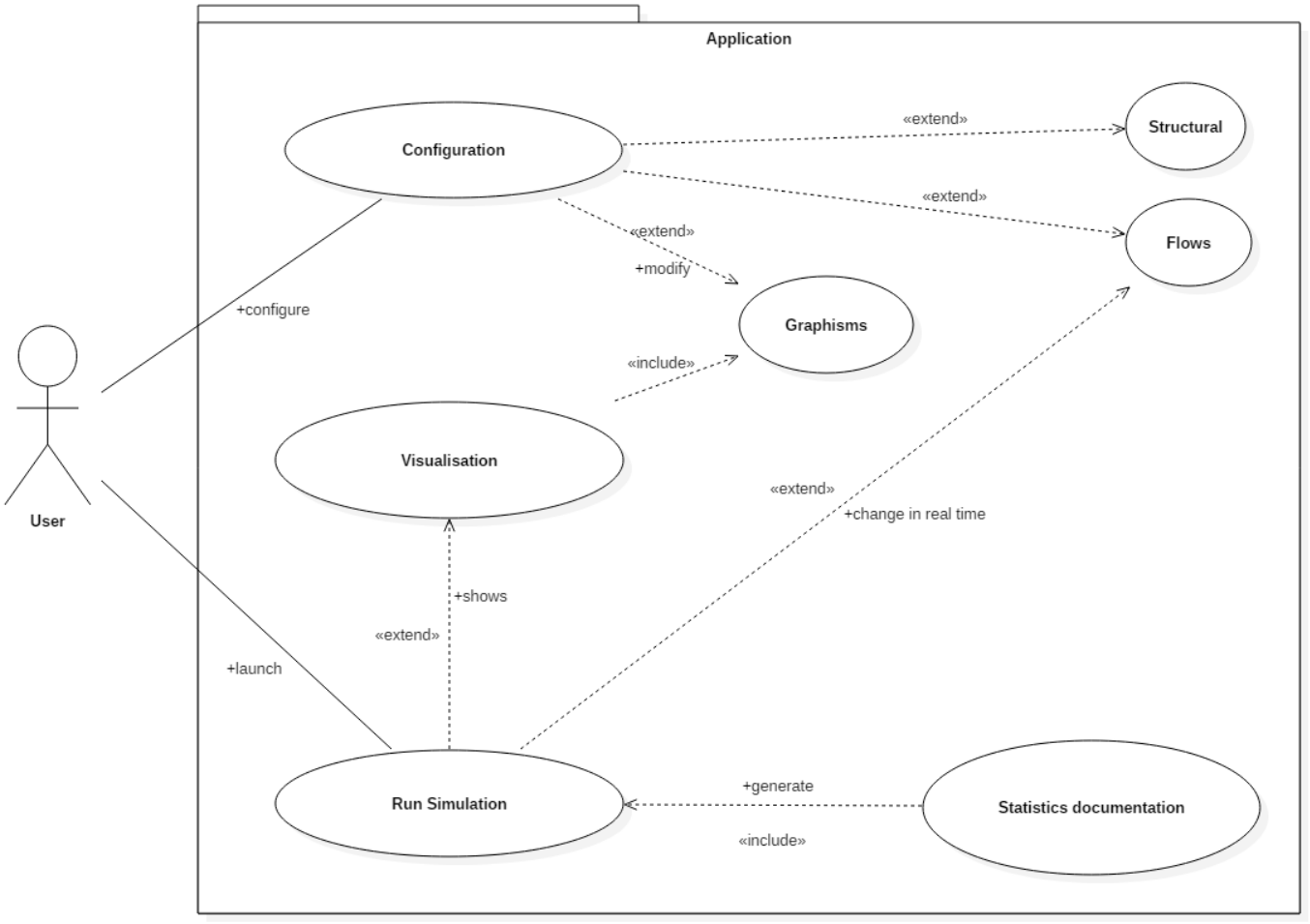
La Visualisation affiche l'état de la simulation à une étape donnée. L'état est mis à jour à chaque tour, à une fréquence définie (1 seconde d'après les fonctionnalités spatio-temporelles définies ci-après).
Ainsi, la visualisation permettra à l'utilisateur de voir en temps réel le déroulement de l'application. Celle-ci pourra prendre en compte les paramètres de visualisation renseignés par l'utilisateur (sinon, la simulation utilisera des paramètres par défaut).

Lancement de la simulation

Grâce aux paramètres renseignés au début de la partie (telle que les différents flux), la simulation pourra se lancer. Cette simulation permettra de générer une documentation statistique.

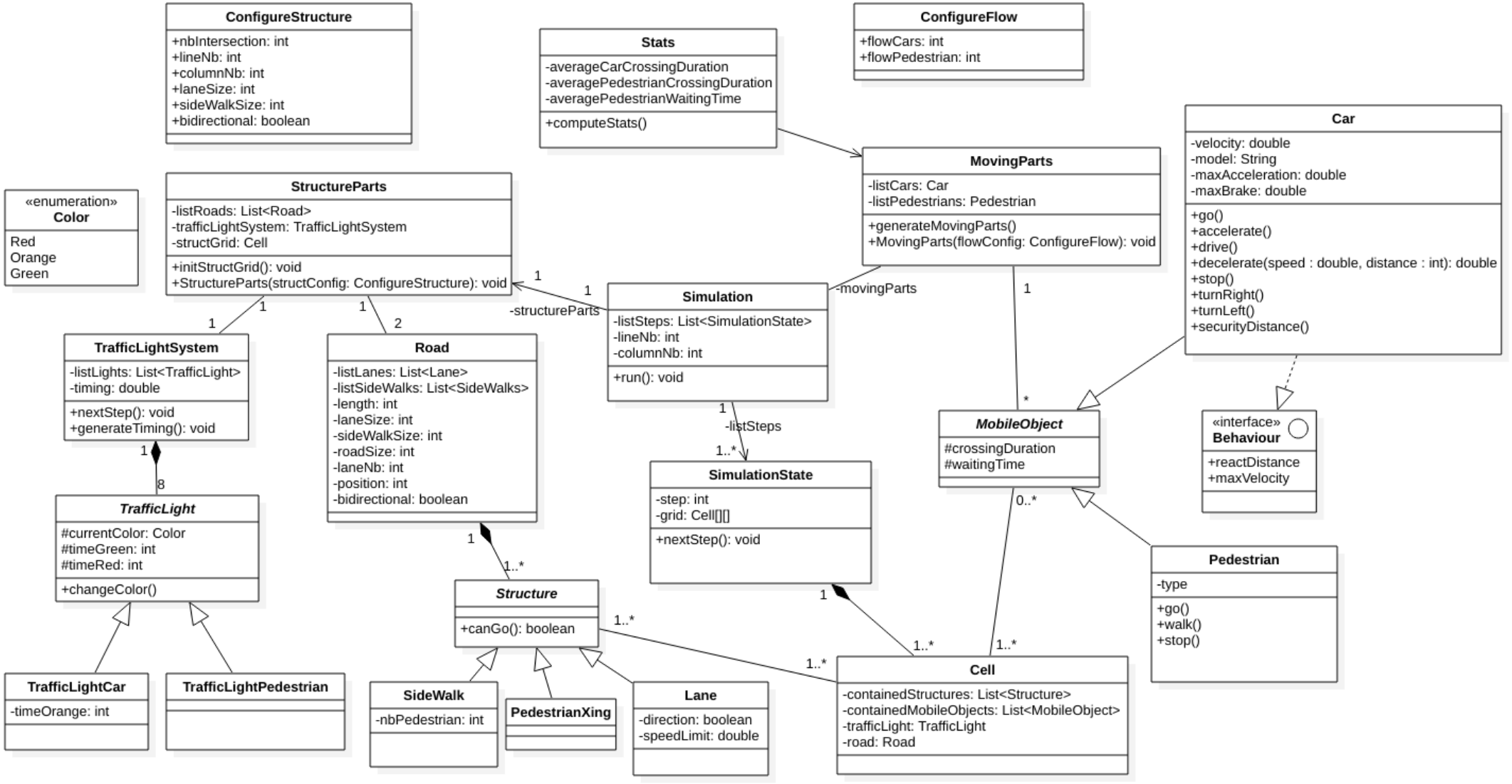
- **Documentation statistique**
  Durant la simulation, le temps qu'un objet traverse le carrefour, sa vitesse, la durée où il est arrêté ou les accidents rencontrés permettra de générer un rapport statistique sur l'ensemble des objets présent lors de la simulation.

3.3 Diagramme de cas d'utilisations



La structure de l'application respectera le diagramme de cas d'utilisation ci-dessus. Chaque fonctionnalité sera implémentée en Java.

3.4 Diagramme de classes



La simulation sera structurée autour de **trois classes majeures** :

- StructureParts
- MovingParts
- SimulationState

3.4.1 StructureParts

Cette classe regroupe **l'ensemble des structures** présentes sur le carrefour. Plusieurs attributs composent cette classe :

- listRoads : est un attribut de type List recensant les objets route du carrefour. Cette liste contient des objets de type Road.
- trafficLightSystem : est un attribut de type TrafficLightSystem. Ce dernier objet permet de conceptualiser et gérer les feux tricolores (leur couleur, leur rythme, leur état). Certains

Cette classe est accompagnée d'une méthode initStructGrid() générant la base de la grille qui servira pour tous les états de la simulation.

3.4.2 MovingParts

Cette classe permet de définir les **objets pouvant se déplacer** dans la simulation. Plusieurs attributs composent cette classe :

- listCars : est un attribut permettant d'enregistrer l'ensemble des voitures de la simulation. Chaque voiture présente sur la simulation est enregistrée dans cette liste contenant des objets de type Car.
- listPedestrians : est un attribut permettant d'enregistrer l'ensemble des piétons de la simulation. Chaque piéton présent sur la simulation est enregistré dans cette liste contenant des objets de type Pedestrian.

Ces deux listes permettront de générer des statistiques du déroulement de la simulation et de prévoir la position de chaque élément au prochain état de la simulation (en fonction de la vitesse actuelle et d'autres paramètres).

Cette classe est accompagnée d'un constructeur generateMovingParts().

3.4.3 SimulationState

Cette classe permet de gérer le **fonctionnement de tour par tour** de la simulation. Plusieurs attributs composent cette classe :

- step : symbolise le numéro de l'étape de la simulation. Cet attribut est un élément important afin d'avoir un ordre de grandeur de la vitesse de déplacement des objets qui se déplacent. Cet attribut est de type int.
- grid : est la grille, matrice, permettant de localiser l'ensemble des StructureParts ou MovingParts de la simulation. Cette grille est une matrice de type Cell[][].

Cette classe comporte également une méthode :

- nextStep() : permet d'appeler l'étape SimulationState suivante.

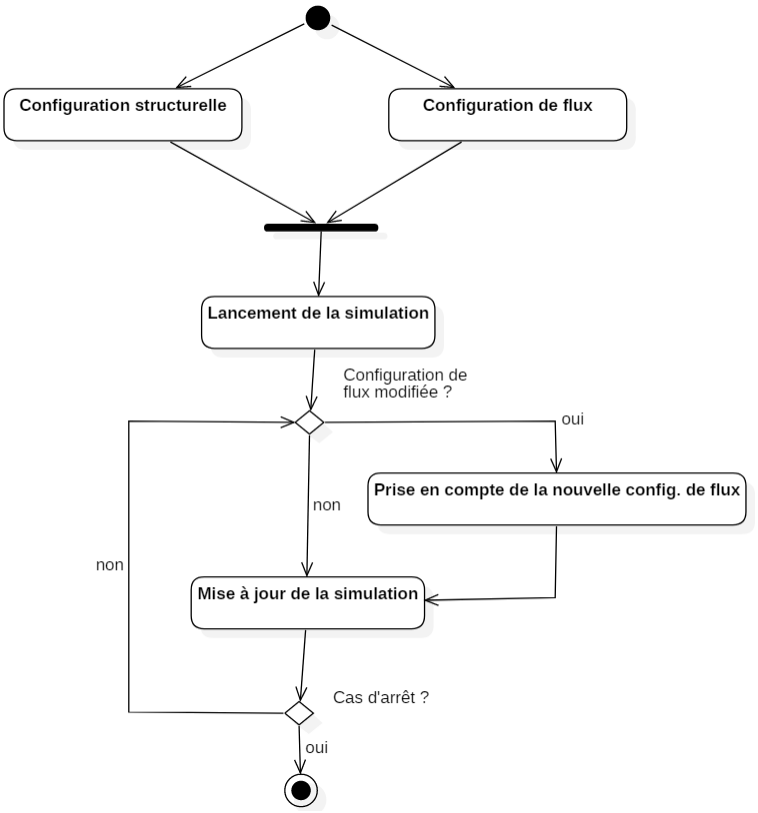
3.4.4 Remarques

À noter que l'ensemble du programme sera organisé selon plusieurs paquets (modem, engins, mobile, immobile, display) pour ordonner toutes les classes.
Les classes ConfigureStructrure et ConfigureFlow ne sont associées à aucune classe et servent juste en paramètre de certains constructeurs, elles regroupent des informations de configuration qui doivent être facilement accessibles (d'où le caractère « public » des attributs).

3.5 Diagrammes d'activités

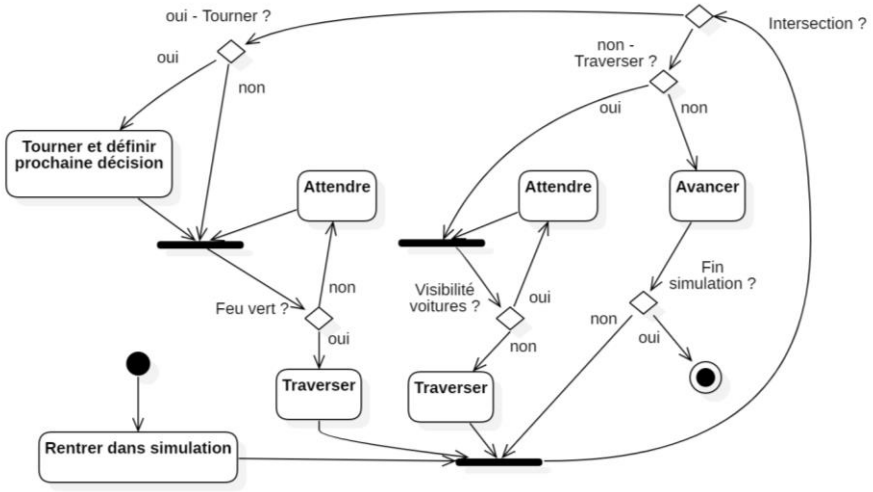
3.5.1 General

Au lancement de Carmageddon Simulator, l'utilisateur configure la partie. Seulement après, la simulation commence. Si l'utilisateur modifie des paramètres de flux véhicules ou piétons, de rythme des feux, alors la partie se met à jour. Dans tous les cas, la simulation continue jusqu'à rencontrer un cas d'arrêt.



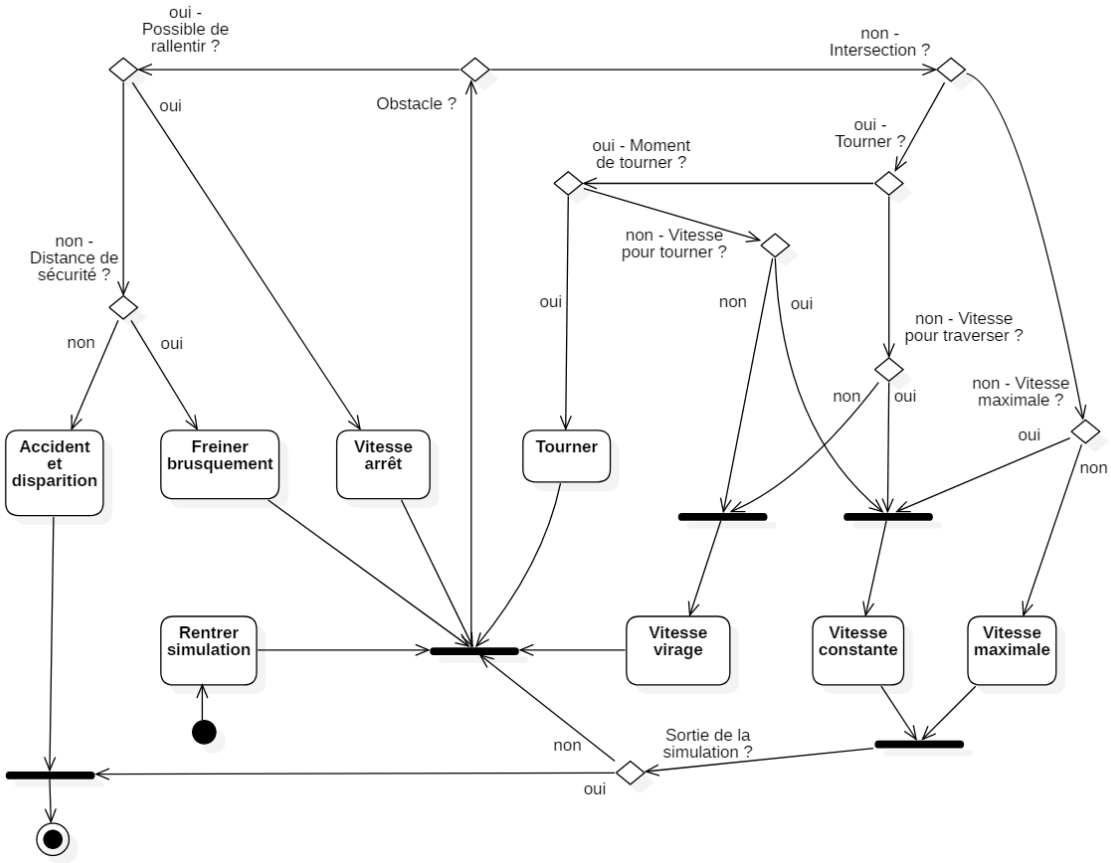
3.5.2 Pedestrian

Une fois entré dans la simulation, les piétons avancent dans une direction qui leur est propre jusqu'à rencontrer une intersection. Si le piéton souhaite traverser, il traversera uniquement si son feu est vert et qu'il peut s'engager. Dans le cas contraire, il attend. Si le piéton ne souhaite pas traverser, il peut continuer son chemin en marchant sur le trottoir.

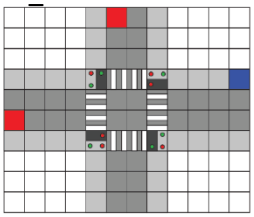


3.5.3 Cars

On retrouve le même schéma, plus détaillé et complexe, pour un véhicule.

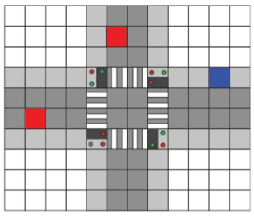


3.6 Application : Carrefour simple



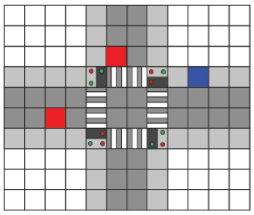
Etape 1 :

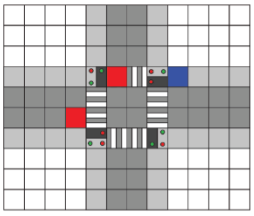
Les voitures (rouge) et piéton (bleu) sont instanciés.



Etape 2 :

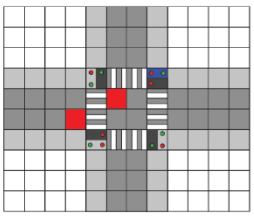
Les différents objets se déplacent dans la direction qui leur est attribuée. Les objets s'arrêtent au feu s'il est orange ou rouge.

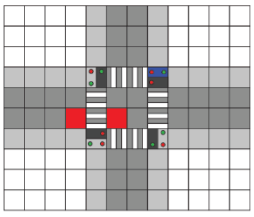




Etape 3 :

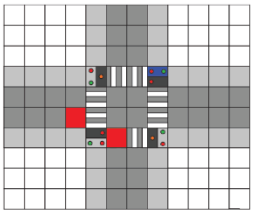
La voiture de gauche et le piéton s'arrêtent car leur feu est rouge. La voiture du haut continue son chemin et traverse le carrefour car il n'y a aucun obstacle, tout en ralentissant.





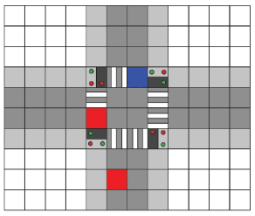
Etape 4 :

Les feux passent à l'orange.



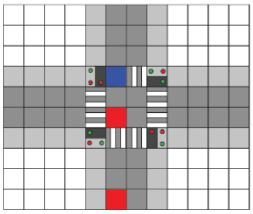
Etape 5 :

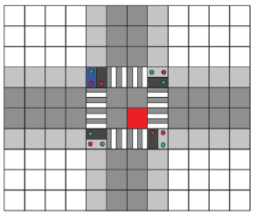
Tous les feux ont changé de couleurs.



Etape 6 :

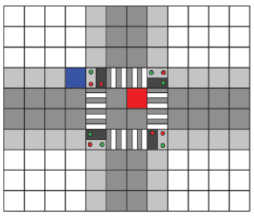
Les objets auparavant à l'arrêt se mettent en mouvement, et traversent.

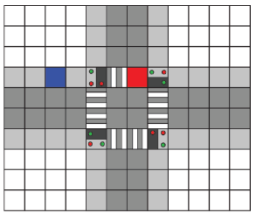




Etape 7 :

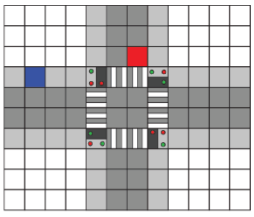
La voiture souhaite tourner à gauche, elle attend que le piéton ait finit de traverser avant de s'engager.

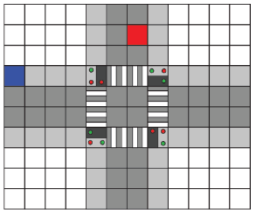




Etape 8 :

Tous les objets continuent leur chemin, jusqu'à sortir de la simulation.





3.7 Fonctionnalités spatio-temporelles

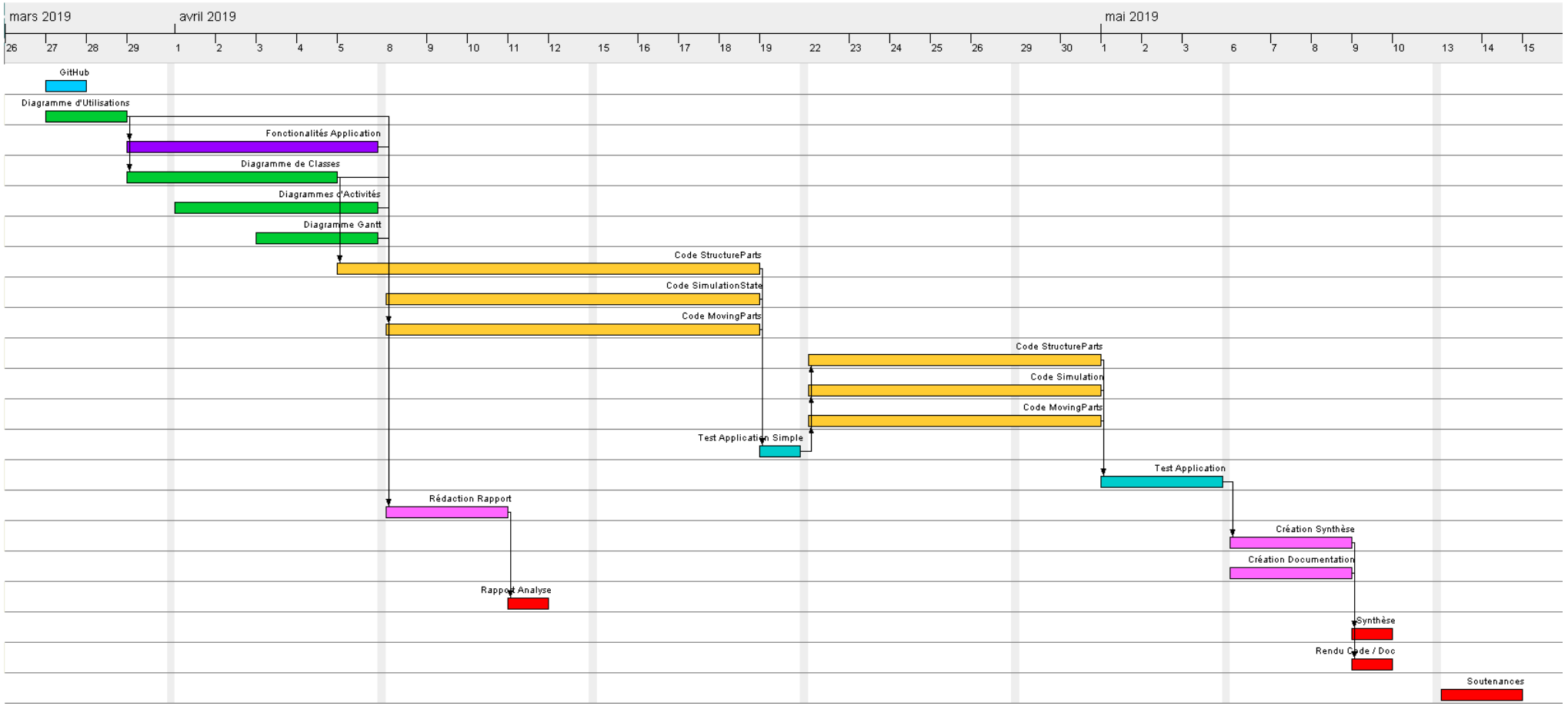
L'unité de base est définie par la taille et le déplacement d'un piéton. En effet, un piéton avance à une vitesse moyenne de 3 km.h-1, ce qui équivaut à un peu plus de 0,8 m.s-1. Cette vitesse minimale correspond à un déplacement d'une case par tour dans notre simulation.

- **1 case = 0,8 mètre**
- **1 tour = 1 seconde**

Le déplacement des voitures sera déduit de ce facteur d'échelle. Par exemple, pour représenter le mouvement d'une voiture se déplaçant à 50 km.h-1, son avatar numérique se déplacera de 50/3 = 17 cases en un tour. Ainsi, ces différentes associations permettront de conserver une cohérence entre la simulation et la réalité.

4 Planning prévisionnel

4.1 Diagramme de GANT



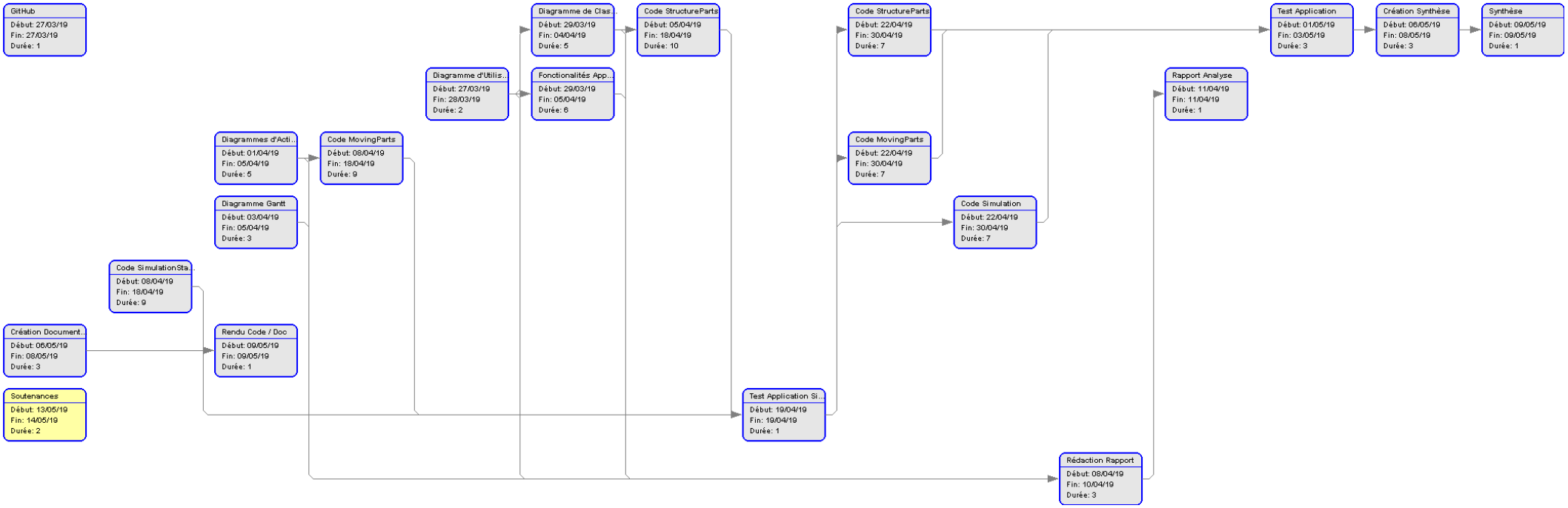
Afin d'**ordonner notre travail** et de mieux appréhender les échéances, nous utilisons un diagramme de GANTT. Dans un premier temps, nous nous sommes concentrés sur la création d'outils permettant le partage de données. Puis, nous avons étudiés le but, le fonctionnement et l'organisation de notre application à l'aide de diagrammes. Ces études préliminaires nous permettent d'implémenter ces fonctionnalités

5 Annexes

5.1 Sitographie

« Traffic Simulation », Wikipedia, WebSite, consulté le 01/04/19.
https://en.wikipedia.org/wiki/Traffic simulation
« NetLogo Traffic Intersection Model », Center fo Connected Learning and Computer-Based Modeling, 1998, Etats-Unis, consulté le 01/04/19.
http://ccl.northwestern.edu/netlogo/models/TrafficInterse- ction
« The Traffic Grid Model », Wilensky, 2015, Etats-Unis, consulté le 01/04/19.
http://www.Netlogoweb .org/launch#http://www.netlogoweb.org/assets/modelslib/Sample%20Models/Social%20Science/Unverified/Traffic%20Intersection.nlogo
« Acceleration and deceleration model », R. Akçelik & M. Belsey, 2001, 9p, consulté le 08/04/19.
http:// citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.603.2520&rep=rep1&type=pdf

5.2 Diagramme de PERT



Le diagramme de PERT nous permet de **planifier.**