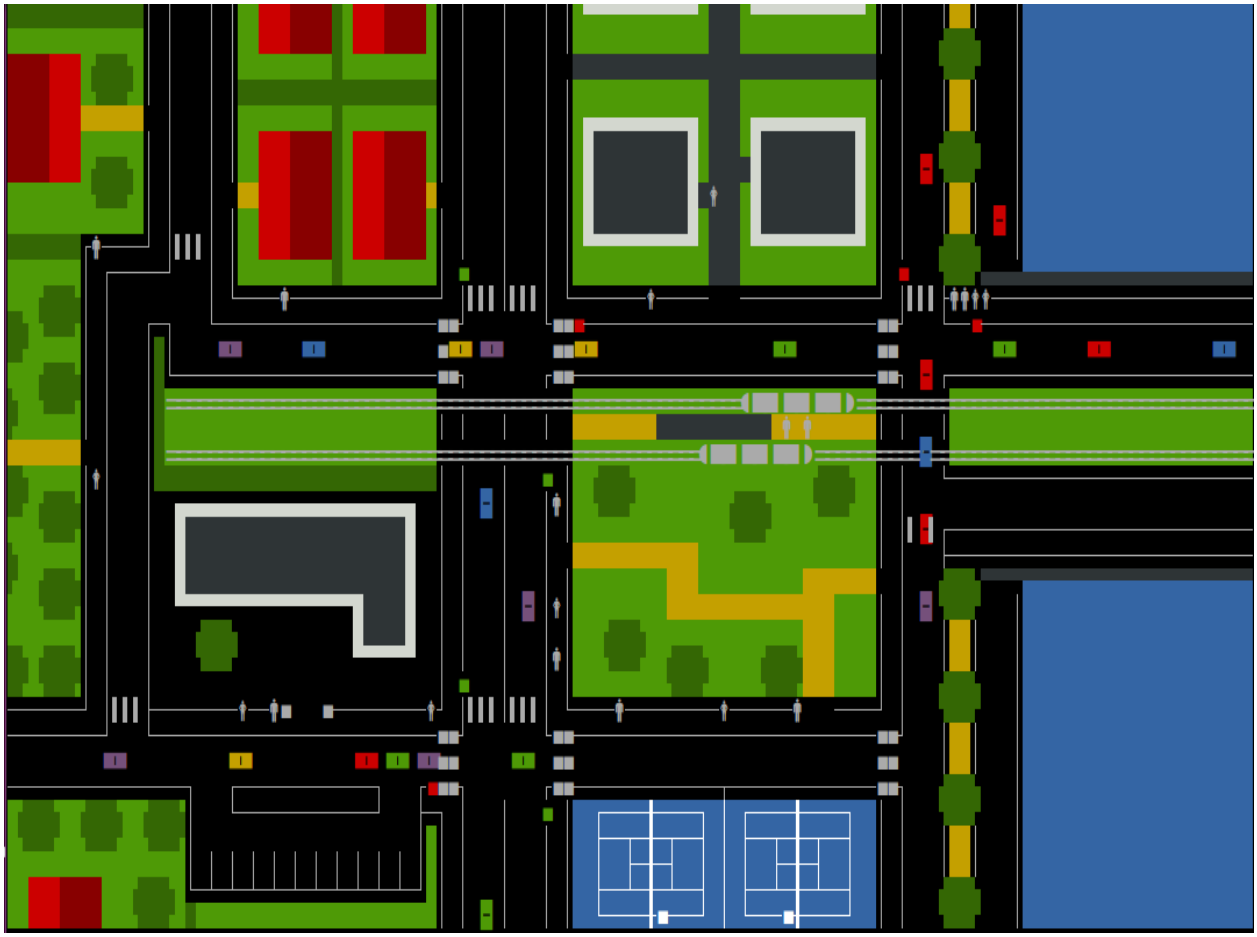
Dilan AKDOGAN, Pierre-Louis CHAYÉ, Classe 33

# Projet C

## Bienvenue à Segmentation City



## Introduction :

Le programme comporte deux modes :

- Un mode circulation normale : dans ce mode, des piétons, des voitures et une ligne de tramway sont modélisés. Leurs mouvements sont régulés au niveau des différents carrefours par des feux ou des carrefours à priorité.
- Un mode « danger » : une voiture percute le tramway au niveau du centre de la carte et prend feu. Les secours arrivent puis la circulation reprend. Les voitures tentent de suivre coûte que coûte leur itinéraire.

La carte comporte trois carrefours régulés par des feux et trois carrefours sans feux, un arrêt de tramway, un pont et une voie passant en dessous, et trois maisons, deux immeubles et une école d'où peuvent apparaître ou disparaître des piétons.

Chaque véhicule ou piéton est associé à un identifiant unique dans sa liste ; ce paramètre est utilisé pour effectuer des actions sur des véhicules.

**Remarques :**

Le code affiche des erreurs de segmentation sur Ubuntu 14.04 mais je n'ai pas pu savoir d'où elles venaient, n'ayant travaillé que sous Ubuntu 16.04 LTS.

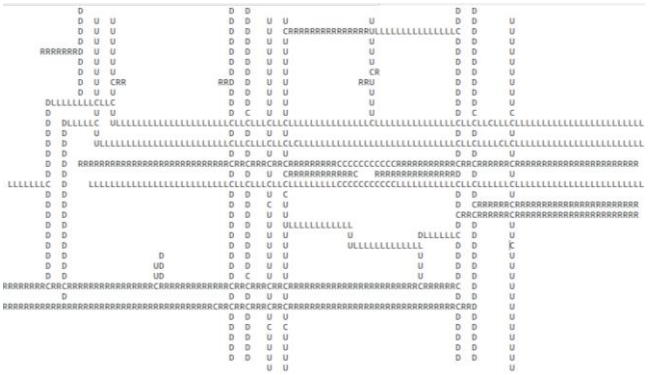
La touche Q permet de revenir au menu pendant le programme

## Fonctionnement global :

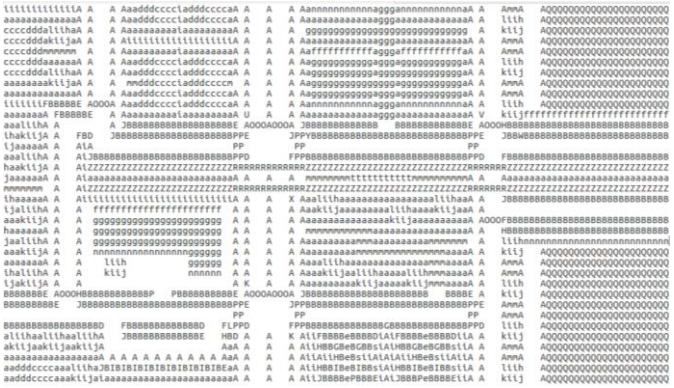
Les voitures sont créées dans des listes chaînées et affichées sur la carte en même temps. Le programme utilise trois listes chaînées : une pour les tramways, une pour les voitures et une pour les piétons.

Le programme fait appel à trois matrices correspondant à deux cartes différentes :

- Une matrice d'**affichage** contenant le décor ainsi que la position des véhicules (ce qu'on appellera la **carte**)
- Une matrice qui ne **varie jamais** et contient uniquement le décor : lorsqu'un véhicule se déplace, il remplace sa position actuelle par l'élément se trouvant à sa place s'il n'y a pas de véhicule (passage piéton, rails de tramway, chemin coloré, …) et le récupère dans cette matrice
- Une matrice de **circulation** spécifiant les sens de circulations sur l'ensemble de la carte – que ce soit pour les piétons, les voitures ou les trams – ainsi que les points critiques. Ces points peuvent être situés à l'intersection de deux voies ou à un endroit où un objet doit marquer un arrêt.



Matrice de circulation



Matrice du décor

## Les fonctions les plus importantes :

- **createCar (~ligne 850) :**

La fonction createCar prend en paramètre la position où il faut créer l'objet, la carte, la liste de véhicules, l'identifiant de la voiture et le type de véhicule à créer. Le véhicule est créé dans la liste et ajouté dans la carte selon le caractère correspondant au type d'objet créé (voiture, tramway ou piéton).

- **createVehicle (~ligne 1550) :**

La fonction createVehicle prend en paramètre : la carte, la liste des voitures, un tableau contenant l'ensemble des points où une voiture peut apparaître, le nombre maximum de voitures pouvant circuler simultanément (défini à 50), et le nombre de points d'apparition possible.

Un point d'apparition est choisi aléatoirement. Si le nombre maximal de voiture pouvant circuler n'est pas atteint, un identifiant est choisi parmi ceux n'étant pas actuellement attribué.

Puis le véhicule est créé à l'aide de la fonction **createCar**. La fonction **createPedestrian** permettant de créer des piétons fonctionne de manière similaire.

- **createTram (~ligne 1600) :**

Cette fonction prend en paramètre la liste des wagons du tram, la carte et le nombre de frames écoulés depuis le lancement du programme. Toutes les 400 frames, un tramway apparaît à gauche de la carte et toutes les 300 frames, un tramway apparaît à droite. Chaque partie du tramway est créée grâce à la fonction **createCar**.

- **moveCar (~ligne 950) :**

Cette fonction prend en paramètre la liste des véhicules (ou des piétons) à déplacer, la matrice de décor statique, la matrice des sens de circulations, la carte dynamique et le type de véhicules que l'on manipule.

Tout d'abord, cette fonction traite les points critiques. C'est-à-dire que si la position d'une voiture correspond à un point critique dans la **matrice des sens de circulation**, on va chercher à le remplacer par une direction à prendre.

Les points critiques sont essentiellement situés au milieu des carrefours, avant les passages piétons, au milieu des passages piétons, au niveau des feux de circulations, au niveau de l'arrêt du tramway ou encore au niveau du pont. Tous ces cas sont à gérer différemment selon l'orientation du carrefour.

Pour déplacer un objet selon une direction précise, il faut d'abord vérifier si aucun autre objet ne se trouve devant lui (une voiture prend plus de place qu'un piéton et doit vérifier si aucun autre objet ne se trouve dans les deux cases devant lui verticalement et trois cases horizontalement).

Ensuite, il suffit de remplacer la case où se situait la voiture par la case correspondante dans la **matrice de décor statique**, d'incrémenter la position de la voiture dans la liste chaînée et de réafficher la voiture sur la carte à sa nouvelle position.

- **killVehicles (~ligne 1475) :**

Cette fonction permet de supprimer les véhicules atteignant le bord de la carte dans la liste chaînée seulement.

Elle prend en paramètre la liste de véhicules choisie, la carte, la matrice qui ne varie jamais et contient le décor, le type de véhicules manipulé et le nombre de points différents où les véhicules disparaissent (ce nombre varie selon le type de véhicule choisi).

Un tableau est créé et rempli avec la liste des coordonnées où les véhicules doivent disparaître. On parcourt ensuite la liste des véhicules et on supprime tous les éléments positionnés sur un point listé dans le tableau précédemment

défini à l'aide de la fonction **deleteCar** (la fonction deleteCar est une fonction qui supprime un élément dans une liste et rechaîne la liste).

- **clearDeadVehicles (~ligne 1510) :**

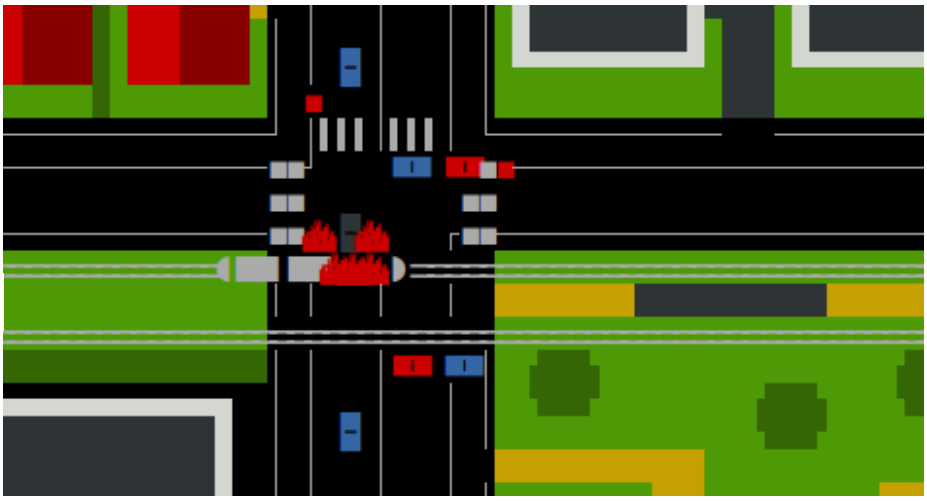
Cette fonction permet de supprimer les véhicules atteignant le bord de la carte sur la carte seulement.

Elle prend en paramètre la carte, la matrice qui ne varie jamais et contient le décor, le type de véhicules manipulé et le nombre de points différents où les véhicules doivent être effacés.

Le même tableau est créé et rempli avec la liste des coordonnées où les véhicules doivent disparaître. On remplace ensuite toutes les positions où les véhicules doivent disparaître par le décor correspondant contenu dans la carte qui ne varie pas.

- **dangerMode (~ligne 1750) :**

Cette fonction crée successivement plusieurs listes. Tout d'abord une liste contenant un tram est créée. Ensuite, une seconde liste contenant la voiture allant avoir un accident est créée. Une fois que l'accident s'est produit, cette seconde liste est vidée et la position de la voiture est affichée statiquement dans la carte. Cette seconde liste est utilisée pour afficher les voitures de secours. Une fois que les voitures de secours se sont réparties autour de l'accident, la liste est à nouveau vidée et leurs positions sont aussi affichées sur la carte en statique. Enfin, la circulation classique est lancée, les voitures évitant tant bien que mal les lieux de l'accident.



Mode danger

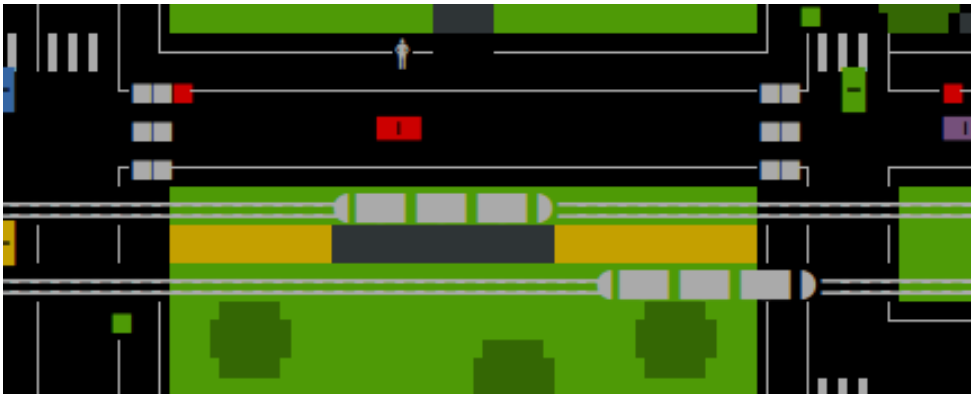
# Problèmes rencontrés :

### Modélisation de la carte :

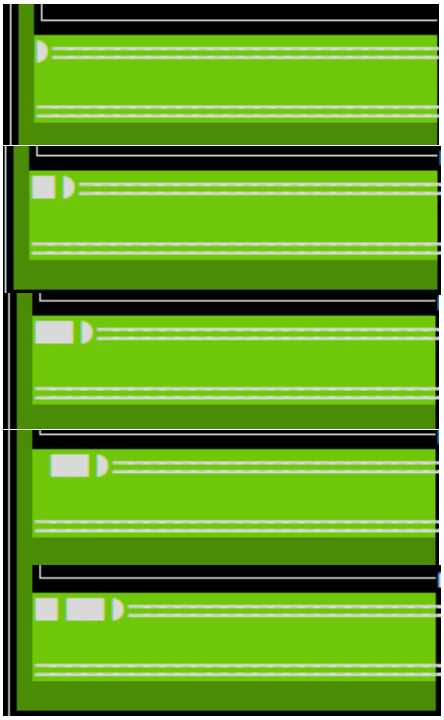
Tout d'abord au niveau de la création de la carte, le plan de circulation était trop complexe et nécessitait des carrefours trop rapprochés, ce qui aurait entraîné des bouchons continuellement. Nous avons donc aligné les carrefours au maximum tout en gardant le plus d'intersections différentes possible.

### Arrêt du tram :

Nous voulions que le tramway ralentisse à l'approche de son arrêt. Nous avons donc placé des points critiques sur toute la longueur du quai dans les deux sens. Ces points critiques ne laissaient avancer le véhicule passant dessus qu'une frame sur deux. Mais lorsque le tram redémarrait, les wagons à l'arrière se détachaient au fur et à mesure (tous les wagons encore au niveau de la gare avançaient lentement mais dès qu'ils sortaient, ils partaient à vitesse normale)
Nous avons donc changé les propriétés des points critiques pour qu'ils soient considérés comme des caractères de direction (droite ou gauche selon le sens) une fois son arrêt terminé.



## Points d'apparition et de disparition du tramway :



Dans notre programme, le tramway est créé comme une succession de 11 voitures. Chaque voiture composant le tram possède un identifiant différent associé à un caractère pour former le tram ci-contre. Nous n'avons pas eu de difficulté pour faire disparaître le tram proprement mais il a été plus difficile de le faire apparaître (il n'est pas symétrique en termes de caractères). De plus, pour le faire apparaître à droite ou à gauche de la carte, l'ordre d'affichage est différent.

La voiture d'identifiant i=0 correspond à l'extrémité droite du tramway, les voitures de i=1 à i=9 correspondent aux wagons et la voiture d'identifiant i=10 correspond à l'extrémité gauche. Si le tram apparaît à gauche de la carte, on fait donc apparaître la voiture 1, la tête du tramway se déplace puis la voiture 2 apparaît et ainsi de suite. Si le tram apparaît à droite de la carte, on fera apparaître la voiture 10 (extrémité gauche du tram) puis elle se déplacera d'un cran et la voiture 9 apparaîtra et ainsi de suite. Voici le code de la fonction :

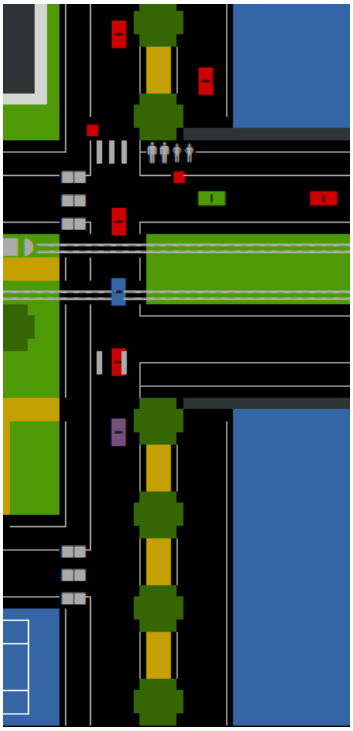
```
List *createTram(List *listOfTram, char **map, int frame){

        int i;
        if((frame%400) >=0 && (frame%400) < 11){
           i = frame%400;
           listOfTram = createCar(listOfTram, 14, 15, i, map, 2);
        }
        else if((frame%300) >= 50 && (frame%300) < 61){
           i = frame%300 - 50;
           listOfTram = createCar(listOfTram, 118, 17, 10-i, map, 2);
        }
        return listOfTram ;

}
```

**Pont :**

La voie verticale la plus à droite passe sous le pont du tramway. Il nous a fallu trouver un moyen de déplacer les voitures dans les listes chaînées sans les afficher sur la carte entre deux points précis. Nous avons donc créé un point critique devant le pont. Lorsqu'une voiture passe sur ce point, son niveau passe de 1 à 0 (le niveau fait partie de la structure de chaque élément).
Dans la fonction déplacement, lorsqu'une voiture dont le niveau est 0 doit avancer, sa position est modifiée dans la liste mais aucun affichage n'est effectué. A la sortie du pont, nous avons placé un point critique qui rétablit le niveau de la voiture à 1.



**Affichage des passages piétons :**

Les voitures horizontales s'étalent sur 2,2 cases en longueur. Lorsqu'une voiture passait sur un passage piéton horizontal, le passage piéton était réaffiché au-dessus de l'arrière de la voiture.
Nous avons donc changé la fonction **moveCar** pour qu'elle efface les deux cases sur lesquelles les voitures se trouvent lorsqu'elles sont en déplacement horizontal. Nous avons aussi changé le réaffichage des voitures en conséquences.

# Améliorations possibles :

### Mode danger :

Nous voulions rajouter des voitures qui tombaient en panne régulièrement dans le mode danger forçant les autres voitures à les contourner mais faute de temps nous n'avons pas pu l'implémenter en C. Nous voulions aussi ajouter des dépanneuses pour tirer les voitures en pannes jusqu'à un bâtiment où elles disparaitraient toutes les deux.

### Court de tennis :

Depuis que nous avons dessiné les courts de tennis sur la carte nous voulions les utiliser et afficher dessus un programme « pong ». Nous avions aussi envie d'ajouter des bruitages et de la musique.

## Conclusion :

Au début du projet, nous avons eu du mal à avoir un aperçu global de la direction à prendre pour implémenter le code. Nous n'avons réussi notre fonction de déplacement que tardivement. Concernant les points critiques, il nous a fallu un peu de temps pour obtenir les résultats que nous attendions partout sur la carte. Nous avons finalement manqué de temps pour l'implémentation du mode danger et la finalisation du code.