

---

# Rapport Projet C 3A

## Simulateur de trafic routier

Franceschini Joffrey & Soltana Mehdi Classe 34

---



---

# Sommaire

I . Introduction	p. 3
II . Mise en place du projet	p. 4
III . Répartition des tâches	p. 5
IV . Réalisation finale	p. 6 - 10
V . Problèmes rencontrés et solutions apporté	p. 11 -12
VI . Amélioration possible à envisager	p. 13
VII. Conclusion et analyse personnelle	p. 14
VIII. Annexes	p. 15

---

# Introduction

Tout au long du cycle ingénieur, nous avons pour but d'apprendre et de persévérer dans de nouvelles matières, théoriques ou pratiques comme l'informatique par exemple.

Cela fait maintenant deux années consécutives que nous avons commencé le langage C avec un projet par ans ou deux et nous entamons la troisième année d'entrée de jeu avec un nouveau projet qui prendra en compte tout ce que nous avons appris jusqu'à présent.

Ce projet est original et mêle l'utile à l'agréable car il va nous permettre de concrétiser et d'améliorer nos connaissances en langage C tout en prenant gout à le faire.

Celui-ci consiste à illustrer sur une map de notre choix, une mise en situation de la vie de tout les jours. C'est a dire mettre en place un trafic routier avec différents moyens de transports tels que la voiture ou le tramway et animer le tout en ajoutant des piétons par exemple.

---

# Mise en place du projet

Dès le départ nous avons bien identifier les grands points clé pour la réussite du projet. C'est à dire la réalisation de la map en fichier « .txt », l'écriture de la map dans le terminal, la transformation en une matrice, et bien évidemment la liste chaînée des voitures, la liste chaînée des trams et toute les fonctions qui en résultent.

Nous avons donc fait un planning pour pouvoir réussir le projet dans les temps. Nous avons consacré nos premières semaines à la réalisation du fichier « .txt » de la map et de la lecture dans le terminal. Ensuite nous nous sommes directement mis à développer les fonctions liées aux le trams et aux les voitures.

Voici-ci dessous notre planning d'organisation entre Mehdi et Joffrey pour le développement du projet

Tâches	
semaine 1 - 2 - 3	Réalisation de la map « .txt » et la lancé sur le terminal
semaine 4 - 5	Réalisation du Tram
semaine 6 - 7 - 8	Réalisation des Voitures
semaine 9 - 10	Réalisation des feux + correction de bugs

Pour pouvoir se tenir informer de l'avancer du projet entre Mehdi et Joffrey, nous avons créer un répertoire GitHub pour faciliter l'échange de code et pourvoir facilement coder chacun nos fonctions sans gêner l'autre.

---

# Répartition des tâches

Pour ce qui concerne la répartition des tâches nous avons tout les deux autant travaillé sur le projet.

Nous avons d'abord commencé par dessiner ensemble, la map sur papier afin de pouvoir visualiser sur quel environnement nous allions travailler.

Après avoir fini l'architecture de la map, nous avons réalisé un schéma du celle-ci avec les différentes directions que les voitures allaient prendre durant leur trajets, comme vous pouvez le voir si dessous:



Suite à l'achèvement de cette étape nous avons tout de suite défini les rôles de chacun au sein du projet.

Joffrey se devait d'animer et de décorer la map. Il devait aussi réaliser les différentes parties du code qui allaient engendrer la mise en place des trams et des feux rouges.

Mehdi a travaillé essentiellement sur la partie du code incluant les voitures.

Nous avons définis des rôles propres à chacun mais il faut savoir que nous avons souvent dû intervenir pour l'un ou pour l'autre afin de débloquer la situation quand cela était nécessaire.

---

# Réalisation finale

## I. Structures de données

Pour ce projet nous avons du réaliser plusieurs structures de données que cela soit pour les entrées, les feux, les trams ou encore les voitures. Voici ci dessous nos trois structures majeurs.

### A- La structure voiture :

Notre structure voiture contient toute les informations que nous avons besoin pour faire déplacer notre véhicule. C'est à dire la direction, la position en X et en Y sur notre matrice, l'état de notre voiture c'est à dire si elle est en panne ou non. Un pointeur sur une entrée de notre carte, un pointeur sur notre voiture suivante et un compteur « tempsPanne » qui permet de gérer la durée de la panne de la voiture.

```
struct voiture {
    char direction;
    int posx, posy;
    char etat;
    int temps;
    POINT_ALEATOIRE* entree;
    struct voiture *next;
    int tempsPanne;
};
```

### B- La structure du tram :

Notre structure pour le tram reprend le meme principe que la structure de la voiture c'est à dire avec une direction, une position en X et en Y sur notre matrice, un état qui contrôle s'il est à l'arrêt ou non, une longueur (son nombre de wagon), un compteur « stopTime » qui permet de gérer la durée de l'arrêt à la station. Un pointeur sur l'entree du tram un pointeur sur sa sortie, et un pointeur sur son maillon suivant.

```
struct tram {
    char direction;
    int posx, posy;
    char etat;
    int temps;
    int longueur;
    int stopTime;
    POINT* entree;
    POINT* sortie;
    TRAM* next;
};
```

### C- La structure des feux

Notre structure feux est plus simple que les deux autres structures expliquées au dessus, elle comporte seulement la position en X et en Y sur la matrice, l'état du feux c'est à dire rouge ou vert, la portée à laquelle agit le feu, un compteur qui permet de gérer le temps d'activation , une direction et un pointeur sur le feu suivant

```
struct feux {
    int x, y;
    char direction;
    char etat; // r -> rouge, b -> vert
    int portee; // nombre de carres affectes par les feux
    int compteur; //compter le temps de feux
    FEUX* next;
};
```

## II. Notre Carte

The image shows a terminal window titled "map\_cp.txt" displaying a text-based map of an airport. The map is a large grid of characters, with the word "AERODROM" prominently displayed in the center in a large, stylized font. The map is surrounded by a border of 'T' characters. The terminal window has a standard macOS title bar with red, yellow, and green buttons.

Si on y regarde de plus près (vous pouvez trouver la carte dans le dossier `fichier_txt/map_cp.txt`), on peut remarquer les lettres ‘r’ qui nous permette de dire a la voiture d’aller à droite (right en anglais) . Voici une liste non-exhaustive des caractères d’information dans notre fichier :

- 7

### III. Affichage dans le terminal

Pour afficher dans le terminal nous récupérons caractère par caractère dans le fichiers « map\_cp » le caractère et nous affichons le caractère d'ascii étendu qui correspondant à la lettre récupérée dans le fichier, ce qui permet cela est la fonction *printCharacter* inclus dans le fichier *vehicule.c*. Nous avons aussi une fonction spécifique qui permet d'afficher les caractère qui change de place comme le tram , la voiture et la voiture en panne. Voici la fonction qui nous permet de faire cela ci-dessous :

```
//Permet d'afficher les caractères spécifiques des voitures et tram
void printCharacterSpe(int c, int terrain) {
    if (c == '$') {
        //Icône pour la voiture
        printf(RED "\033[47m■" NRM);
    } else if (c == '%') {
        // Icône pour le tram
        printf(YEL "■" NRM);
    } else if (c == '!') {
        //icône voiture en panne
        printf(BLK "\033[47m†" NRM);
    } else { //afficher caractère non voiture et non tram
        printCharacter(c);
    }
}
```

Cette fonction *printCharacterSpe* permet d'afficher :

- la voiture avec l'icône '■' quand `c == '$'`
- la voiture en panne avec l'icône '†' quand `c == '!''`
- le tram avec l'icône '■' quand `c == '%'`
- et sinon appelle la fonction *printCharacter* qui affiche tout les autres caractères

### IV. La fonction « *Panne* »

La fonction « *int Panne()* » qui se trouve dans le fichier *vehicule.c*. N'as rien de compliquer mais utilise une syntaxe du C peut courante, elle utilise en *return* assez intéressant.

```
//Faire tomber en panne une voiture avec probabilité de 1/20
int Panne() {
    int i = rand() % 20;
    return i > 0 ? 0 : 1; //return i si i > 0 return 1 si i = 0
}
```



Cette fonction calcul le fait qu'une voiture tombe en panne avec une chance de 1/20. On effectue un tirage aléatoire avec « `i = rand()%20` » cela nous permet d'avoir un nombre entre 0 et 19. La syntaxe tu « `return` » qui dit précédemment est original et peut ne pas être comprise par tout le monde c'est pourquoi nous vous l'expliquons maintenant, « `return i > 0 ? 0 : 1` » permet de retourner 1 si `i = 0` ou retourner `i` si `i > 0`. Donc quand nous retournons 1 cela signifie que la voiture devra être en panne. Cette fonction est active que quand le mode DANGER est sélectionné

## V. Notre main()

Notre main contient le moins de chose possible, il fait appel à la fonction `getTaille()` qui nous calcul la taille de la map grâce à cela nous récupérons la hauteur et la largeur. Nous faisons alors deux allocations dynamique de deux matrices une qui contient la carte et une qui contient les informations de la case si elle est occupée ou non. Nous faisons aussi appel à la fonction `affiche_menu()` qui nous affiche un joli menu et qui nous demande de sélectionner entre trois choix :

- 1 NORMAL
- 2 DANGER
- 3 QUITTER

Le résultat de ce choix sélectionné par l'utilisateur entraînera des modifications à l'initialisation de la simulation, c'est à dire si le mode sélectionné est le mode NORMAL alors nous aurons 20 voitures pendant la simulation, si DANGER est sélectionné nous aurons 40 voitures pendant la simulation. Enfin si le choix est de QUITTER alors nous aurons un joli menu en remerciant d'avoir utilisé la simulation.

*Un petit EASTER EGG si vous tapez 4 il y aura 60 voitures pendant la simulation.*

Après avoir choisi le mode de la simulation, nous créons nos entrées, notre liste de feu, notre liste de tram, notre liste des voitures et enfin notre liste des feux. Après cela nous entrons dans une boucle jusqu'à ce que la touche 'q' soit entrée par l'utilisateur. Dans cette boucle nous faisons avancer les trams, l'avion, les voitures, activons les feux et affichons la carte dans le terminal. Voici notre boucle :

```
do {
    usleep(200000);
    avancerTram(listeTram, taille, matrice);
    deplacementAvion(avion, taille);
    deplacementVoiture(setupVoitures, calcul, matrice, occupee, taille, mode);
    activerFeux(listeFeux, occupee, taille);
    system("clear");
    afficherPlanFinal(matrice, taille, setupVoitures, listeFeux, listeTram, avion);
    printf("\033[5;40;1;31m\033[44;1H TAPEZ 'q' pour quitter\033[0m\n");
} while(key_pressed() != 113);
```

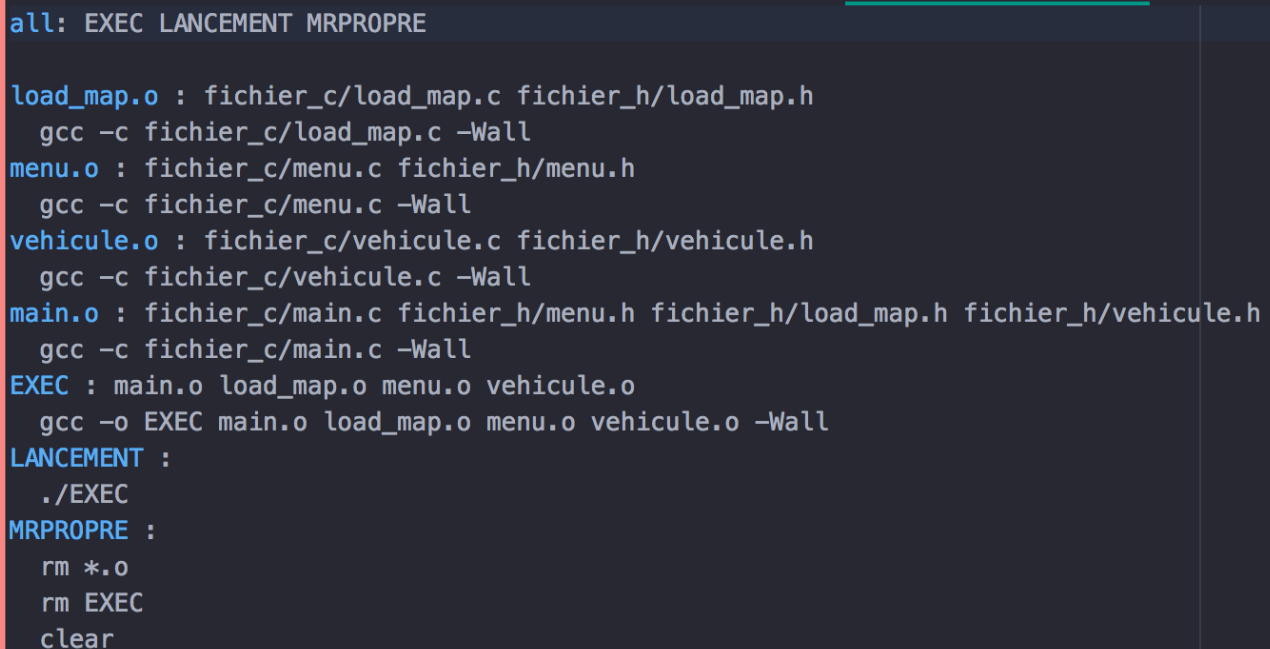
---

La condition de fin de boucle est utiliser avec la fonction `key_pressed()` qui permet de récupérer la valeur d'un caractère en ASCII, ici 113 pour 'q'.

Enfin en quittant cette boucle nous affichons les remerciement pour avoir utiliser notre simulation. Le message s'affiche un certain nombre de fois avant de s'effacer et de remettre le terminal à son état d'origine.

## VI. Notre MakeFile

Notre MakeFile rend la compilation de notre projet extrêmement final en tapant juste *make* dans le terminal la compilation se fait toute seule. Et quand vous quitterez le programme elle supprimera tout les fichier créer lors de la compilation (tout les \*.o) et l'exécutable. Ci-dessous vous trouverez une capture d'écran de notre MakeFile :

A screenshot of a terminal window showing the content of a Makefile. The text is as follows:

```
all: EXEC LANCEMENT MRPROPRE

load_map.o : fichier_c/load_map.c fichier_h/load_map.h
    gcc -c fichier_c/load_map.c -Wall
menu.o : fichier_c/menu.c fichier_h/menu.h
    gcc -c fichier_c/menu.c -Wall
vehicule.o : fichier_c/vehicule.c fichier_h/vehicule.h
    gcc -c fichier_c/vehicule.c -Wall
main.o : fichier_c/main.c fichier_h/menu.h fichier_h/load_map.h fichier_h/vehicule.h
    gcc -c fichier_c/main.c -Wall
EXEC : main.o load_map.o menu.o vehicule.o
    gcc -o EXEC main.o load_map.o menu.o vehicule.o -Wall
LANCEMENT :
    ./EXEC
MRPROPRE :
    rm *.o
    rm EXEC
    clear
```

---

# Problèmes rencontrés et solutions apportés

Tout au long du projet nous avons rencontré une quantité de problèmes indénombrables.

Le premier problème qui nous a fait face est celui concernant l'affichage de la map car nous avons quelques décalages quand celle-ci fut figé sur le terminal. Nous avons résolu ce problème nous-même en très peu de temps et nous avons continué d'avancer jusqu'au problème suivant.

Concernant les trams, nous avons mis une certaine période avant de les finaliser et de les faire fonctionner car nous avons un problème sur ce sujet. Les deux trams ne fonctionnaient pas exactement comme nous le souhaitions, c'est à dire que les deux ne se lançaient pas en même temps. Nous nous sommes lancés, pour commencer, dans des doubles boucles pour pouvoir afficher le trams pas à pas grace à deux incréments. Cependant nous avons vite lâché cette directive et avons effectuer des recherches pour comprendre exactement comment cela devait fonctionner.

Nous avons donc demandé quelques conseils au professeurs qui nous a mit sur une piste. Celle-ci consistait à réaliser un code avec des listes chaînées. De ce fait, nous avons entamé une phase de travail assez chargé car c'est à ce moment la que nous avons commencé a rencontrer notre premier réel problème. Nous avons mis un délai de une à deux semaines pour pouvoir enfin avoir accès un trams entièrement fonctionnel et optimisé.

Notre plus grande difficulté au sein du projet a été surtout focalisée sur la mise en circulation des voitures. Nous sommes parti sur plusieurs versions de code afin de pouvoir enfin aboutir a une version fini et fonctionnel de celui-ci. Premièrement nous avons commencé par afficher la map en dur en mettant un « %c » partout où choques voiture étaient sensées défiler. Nous sommes parti sur une très bonne lancée car nous avons effectué le déplacement des voitures sur les deux premières routes assez rapidement.

Cependant, nous nous sommes très vite rendu compte que des erreurs s'étaient glissées dans notre affichage en dur, ce qui a complètement faussé nos calculs sur les points critiques, c'est à dire les coordonnées du point où les voitures se devaient de tourner.

Nous avons essayé de rectifier cela mais en vain. Après une longue période de recherche, nous n'avons toujours pas trouvé nos erreurs.

---

Du coup, nous avons laissé tomber l'idée d'afficher la map en dur et avons repris notre idée de départ en affichant la map directement à partir d'un fichier «.txt ».

Nous avons donc commencé tout doucement à entamer le code pour le déplacement des voitures. Nous sommes parti sur l'utilisation des listes chaînées tout comme pour les trams et commençons à comprendre petit à petit comment aboutir au résultat final.

Peu à peu, nous avons réussi à faire bouger les véhicules mais ils n'étaient pas encore aux points car nous avons rencontré beaucoup de bugs car les voitures se déplaçaient là où elle ne devait pas.

Suite à cela et à la deadline du projet qui approchait à grands pas, nous avons choisi de travailler ensemble coûte que coûte et de s'entraider pour pouvoir finir le projet en temps et en heure mais surtout d'obtenir la satisfaction de rendre un projet fonctionnel avec un code optimisé.

---

# Amélioration possible à envisager

Sur ce projet, nous nous sommes vraiment donnés à fond afin de pouvoir combler nos lacunes et terminer le projet à temps.

Une amélioration qui sera envisageable est de mettre une vraie icône de voiture comme celle-ci '🚗' à la place de celle que nous avons aujourd'hui. Une nouvelle icône pour le tram aussi sera intéressante à avoir.

L'intégration de piétons sortant de notre tram et qui rejoindra notre aéroport pourra être aussi implémenté.

De ce fait, le projet perçoit, nous le pensons, quelques améliorations possibles telle que la mise en circulation des voitures avec peut être une vitesse mieux adaptée, un graphisme mieux travaillé et une mise en place de la SDL.

---

# Conclusion et analyse personnelle

Pour clôturer tout ce qui a été dit précédemment, nous avons tout les deux effectué un travail proportionné pendant toute la durée du projet. Nous avons également su s'entraider et se conseiller mutuellement lorsque nous rencontrions une difficulté et pensons avoir adopté la bonne méthode de travail en groupe dès le départ.

En ce qui concerne notre avis personnel, nous pensons que de réaliser des projets comme celui-ci est une très bonne expérience que ce soit sur le plan humain ou technique car cela nous met dans des situations que nous serions susceptible de rencontrer à l'avenir. Nous avons appris à faire face aux difficultés tout en gérant notre stress et à communiquer au sein du projet.

Enfin nous avons vraiment apprécié le fait d'avoir été suivi dans un projet qui en valait la peine et sommes très satisfait du résultat que nous vous proposons.

---

# Annexes

Webographie :

- <https://openclassrooms.com/>
- <http://devdocs.io/c/>
- <https://www.developpez.com/>

Lien du GitHub :

- <https://www.github.com/JoffreyFrancesch/Projet-C>

Lien de Téléchargement du projet (GoogleDrive) :

- [https://drive.google.com/drive/folders/1OZAfqTEZ4PDZFdJe86AKTwV0fBs2\\_fIq?usp=sharing](https://drive.google.com/drive/folders/1OZAfqTEZ4PDZFdJe86AKTwV0fBs2_fIq?usp=sharing)