Projet Autonome en Programmation C

(Encadrement : Michael FRANÇOIS)

------ SIMULATEUR DE CIRCULATION URBAINE -----

Présentation générale

L'objectif de ce projet est de programmer en langage C, un simulateur dynamique de circulation urbaine. Le projet est à réaliser obligatoirement en binôme (ancien & nouveau) sous environnement GNU/Linux. La simulation du trafic doit se faire comme dans la vraie vie. L'affichage doit être effectué uniquement sur console.
La FIG 1 montre un exemple de plan de circulation réalisé sur le terminal.

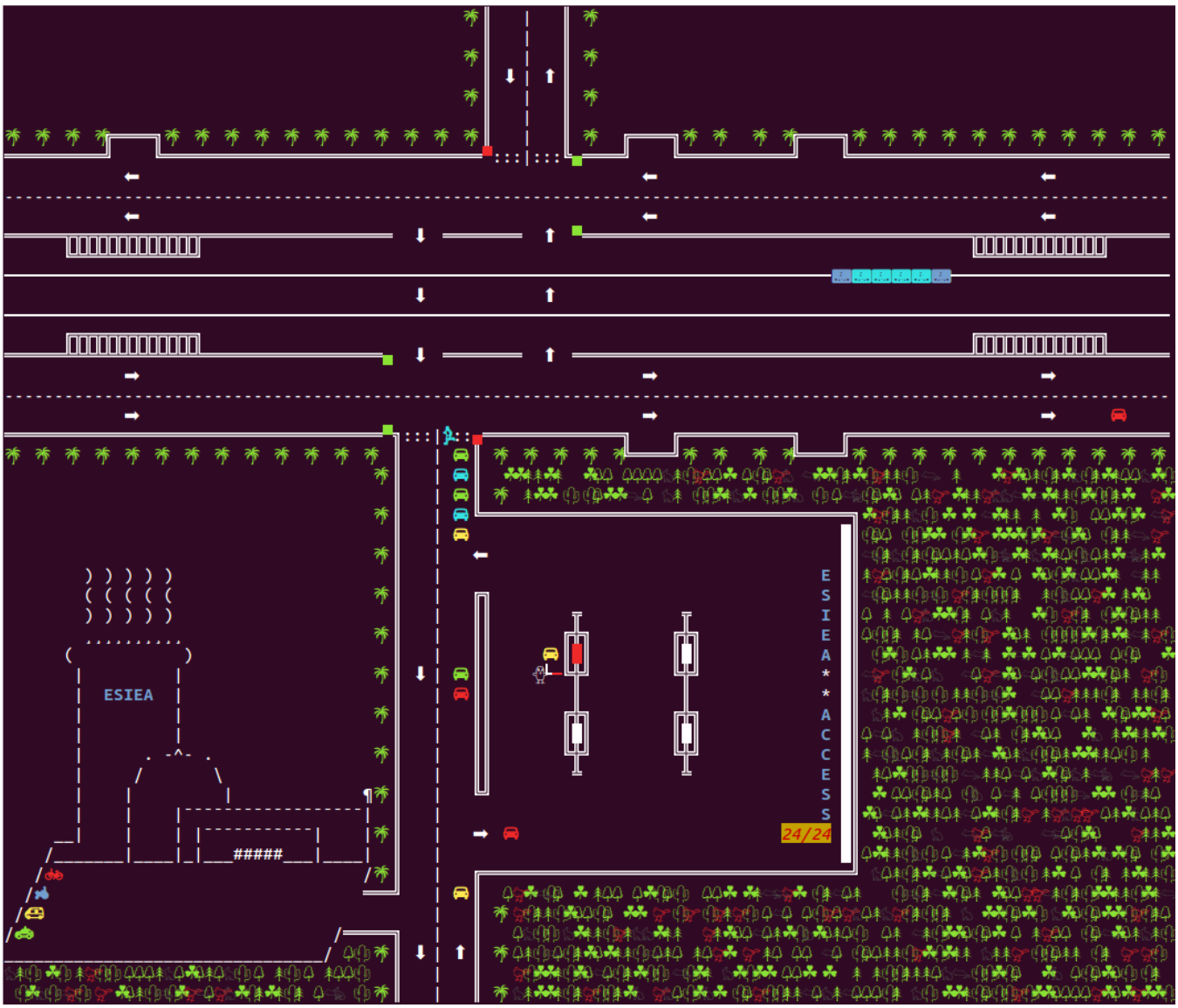

FIGURE 1 – Exemple de plan de circulation réalisé sur le terminal.

Ce plan de circulation a été inspiré d'un vrai plan existant, hormis le décor (la station et les bâtiments) qui a été rajouté pour complexifier davantage la maquette. On constate que ce plan comporte :

- des voitures sur des routes à sens unique comme à double sens ;
- des feux tricolores (rouge, orange, vert) ;
- deux lignes de tramway avec deux stations chacune ;
- une station service (ESIEA--ACCESS 24/24), contenant quatre pompes. On constate même un conducteur mettant du carburant à son véhicule ;
- un piéton traversant la route via le passage ;
- des bâtiments ;
- des palmiers, des arbres et des animaux divers.

Ce plan de circulation se trouve à la base dans un fichier `.txt` y compris le décor. Pour alléger l'affichage, le plan doit être figé sur l'écran afin d'éviter de recharger ce dernier à chaque fois pour ne pas ralentir l'exécution.

Vous pouvez utiliser ce plan ou choisir un autre (conseillé) qui doit être aussi complexe voire même plus que celui proposé.
Vous pouvez rajouter du son pour votre jeu, afin de le rendre plus plaisant :

- musique de fond ;
- son au moment d'une grosse accélération ;
- son au moment du freinage à une vitesse élevée ;
- klaxon ;
- son au moment d'un crash ;
- cloche de tramway ;
- sirène d'alarme : pompiers, police, etc.
- etc.

Pour cela vous pouvez utiliser la librairie `sox` et lancer les sons via la commande "play" en tâche de fond, afin de ne pas perturber l'affichage pendant le déroulement du jeu.
Au lancement du jeu, le joueur doit pouvoir choisir entre deux modes :

- **Fluide :** la circulation est fluide et sans danger pour les usagers.
- **Danger :** la circulation est dangereuse avec des accidents, des pannes, des travaux sur les routes, etc.

Rien ne vous empêche de rajouter d'autres modes si vous le désirez, d'ailleurs je vous le conseille fortement et ça sera du bonus pour votre note finale.

Aspects techniques

Les aspects techniques qui suivent correspondent à la réalisation précédente. Vous pouvez vous en inspirer pour votre programme.

Le plan de circulation est chargé depuis un fichier "`.txt`" et affiché classiquement sur la sortie standard. Un tableau bidimensionnel a été utilisé pour stocker la présence ou non d'un véhicule à une position (x ,y) donnée. Ceci permet de gérer plus facilement les collisions entre véhicules. Voilà un exemple de structure utilisée pour rassembler toutes les informations liées à un véhicule :

```
-------------------------------------------------------------------------
typedef struct voiture VOITURE;
struct voiture
{
  char direction ;  /*N => Nord, S => Sud, E => EST, O => OUEST*/
  int posx;         /*Position courante x de la voiture*/
  int posy;         /*Position courante y de la voiture*/
  int vitesse;      /*Vitesse du véhicule*/
  char alignement;  /*'g'=>gauche ou 'd'=>droite*/
  char type;        /*'v'=>voiture, 'c'=>camion, etc.*/
  char custom[30];  /*Contient le véhicule customisé*/
  char ravitaillement ; /*1=>ravitaillement 2=>carburant et 0=>sinon*/
  char etat;        /*État du véhicule => actif ou inactif*/
  struct voiture * NXT; /*Pointeur vers une prochaine voiture,
                          necessaire pour la liste chainée*/
  /*Vous pouvez rajouter d'autres variables si nécessaire */
};
-------------------------------------------------------------------------
```

Vous aurez sans doute besoin d'autres structures, notamment pour le tramway ou même pour le piéton. Pour faire vos choix dans le menu ou même pendant le déroulement, vous pouvez utiliser la fonction `key_pressed()` indiquée ci-dessous :

```
-------------------------------------------------------------------------
char key_pressed()
{
    struct termios oldterm, newterm;
    int oldfd; char c, result = 0;
    tcgetattr (STDIN_FILENO, &oldterm);
    newterm = oldterm; newterm.c_lflag &= ~(ICANON | ECHO);
    tcsetattr (STDIN_FILENO, TCSANOW, &newterm);
    oldfd = fcntl(STDIN_FILENO, F_GETFL, 0);
    fcntl (STDIN_FILENO, F_SETFL, oldfd | O_NONBLOCK);
    c = getchar();
    tcsetattr (STDIN_FILENO, TCSANOW, &oldterm);
    fcntl (STDIN_FILENO, F_SETFL, oldfd);
    if (c != EOF) {
    ungetc(c, stdin); result = getchar();
    }
    return result;
}
-------------------------------------------------------------------------
```

Cette fonction vous permet de récupérer la touche saisie par l'utilisateur sans pour autant retarder l'affichage car, le plan de circulation doit être dynamique. Pour cela vous aurez besoin des librairies supplémentaires suivantes : `signal.h`, `string.h`, `termios.h`, `unistd.h` et `fcntl.h`.

Pour afficher un véhicule sur l'écran, il suffit de déplacer le curseur sur le terminal à la position voulue et ainsi à l'aide d'un `printf` afficher le tableau `custom` correspondant.
Pour avoir un affichage plus joli, les caractères de l'ASCII étendu ont été utilisés. Vous pouvez les retrouver au lien suivant : `http://www.theasciicode.com.ar/`

Vous pouvez également retrouver des émoticônes pour un meilleur rendu de votre jeu à ce lien :
`https://fr.piliapp.com/twitter-symbols/`

Il suffit de cliquer sur le symbole souhaité, le copier puis le coller simplement dans votre fichier source. Dans le cas où le symbole ne s'affiche pas correctement sur le terminal, vous devez installer le package "ttf-ancient-fonts" via la commande :
`sudo apt-get install ttf-ancient-fonts`

NB : pour ceux qui veulent avoir un graphisme plus sophistiqué, ils peuvent utiliser la SDL, mais ce n'est pas nécessaire pour obtenir la note maximale au projet.

Infos pratiques

Le projet est à réaliser obligatoirement en binôme (ancien & nouveau) sous environnement GNU/-Linux, et doit être déposé sur la plate-forme pédagogique *Moodle* au plus tard le 30/11/2017 à 23h55, sous la forme d'une archive `.zip` à vos noms et prénoms (*i.e.* `NOM_prenom.zip`), contenant tous les fichiers sources du projet et également un rapport en `pdf`. Vous expliquerez toutes les démarches effectuées sur chaque partie et conclure en insistant notamment sur les difficultés rencontrées et les problèmes non résolus (s'il en reste).

Votre programme doit obligatoirement présenter les différentes thématiques suivantes :

- tableaux, pointeurs et allocation dynamique ;
- structures et liste chaînée ;
- fichier *Makefile* contenant les commandes de compilation ;
- des lignes de codes commentées, lisibles et bien agencées ;
- un choix cohérent de noms de variables.

L'évaluation sera globalement scindée en 4 parties :

1. le **design** du jeu dans l'ensemble (plan de circulation, décor, etc.) ;
2. test du mode **Fluide** ;
3. Test du mode **Danger** ;
4. le **rapport** en `pdf` qui doit être soigneux, complet et bien détaillé.

Une vidéo d'un exemple de réalisation de ce jeu est disponible sur *Moodle*.

Quelques liens utiles sur les fichiers `Makefile` en C :
`http://gl.developpez.com/tutoriel/outil/makefile/`
`http://icps.u-strasbg.fr/people/loechner/public_html/enseignement/GL/make.pdf`