Projet Autonome en Programmation C

(Encadrement : Michael FRANÇOIS)

------ TRAFFIC RACER -----

Présentation générale

L'objectif de ce projet est de programmer en langage C, un jeu de trafic automobile sur une autoroute. Le projet est à réaliser 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 réalisation technique. On constate une autoroute possédant trois voies de circulation avec des palmiers sur les bords. Le véhicule (en jaune) du joueur se trouve à cet instant à gauche sur l'alignement du bas. On remarque aussi un petit tableau récapitulant certaines informations comme la vitesse du véhicule, le score du joueur et également le meilleur score déjà atteint pour ce jeu. Ceci n'est qu'un exemple de réalisation, vous pouvez bien-sur prendre une autoroute à 4 voies ou même 5 voies. Vous pouvez également rajouter la distance parcourue, le temps, etc. Ici, la vitesse du véhicule ainsi que le score du joueur, sont donnés en temps réel.

Pour alléger l'affichage, l'autoroute doit être figée sur l'écran afin d'éviter de recharger ce dernier à chaque fois pour ne pas ralentir le jeu. Avancer son véhicule équivaut en fait à faire descendre les autres véhicules au fur et à mesure vers le bas ainsi que le décor sur le bord de la route, pour simuler au mieux la réalité.

Dans le cas où le véhicule du joueur accélère (resp. freine), sa vitesse indiquée sur le tableau augmente (resp. diminue). Cependant, dans le cas où le véhicule freine longtemps et que sa vitesse devienne petite (exemple 70 km/h), on verra les véhicules se trouvant déjà sur la plan de circulation, avancer cette fois-ci vers le haut jusqu'à même quitter définitivement le plan de circulation. Par contre si le véhicule accélère de nouveau, il doit à un moment pouvoir rattraper les précédents véhicules avant de commencer à les doubler ($i.e.$ éviter).
En cas de collision avec un autre véhicule, le jeu s'arrête, et le meilleur score du jeu est mis à jour si nécessaire. Le score consiste à incrémenter un compteur au fur et à mesure qu'on dépasse un véhicule. Plus la vitesse du véhicule du joueur est élevée, plus le score obtenu après chaque dépassement est grand.

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 ;
- etc.

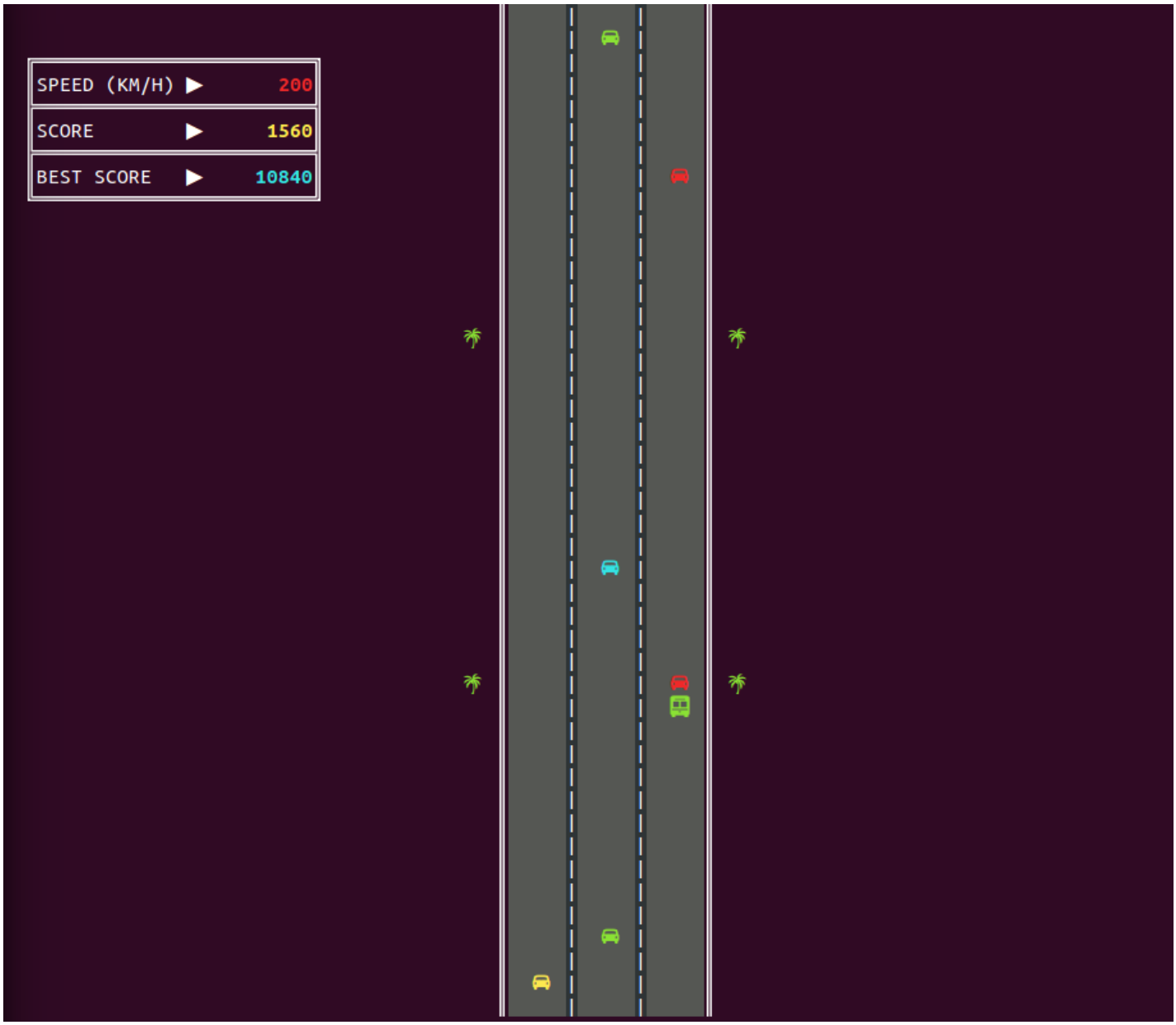

FIGURE 1 – Plan de circulation sur une autoroute à trois voies. Ici le véhicule du joueur est celui de couleur jaune avec une vitesse de 200 km/h et situé en bas à gauche.

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 :

- **Manuel :** le joueur conduit lui même le véhicule et donc gère totalement la conduite.
- **Automatique :** le joueur passe le véhicule en mode pilote automatique et donc le véhicule gère lui même de manière autonome toute la conduite.

Rien ne vous empêche de rajouter un autre mode si vous le désirez, d'ailleurs je vous le conseille fortement.

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
{
    int posx;         /* Position courante x de la voiture */
    int posy;         /* Position courante y de la voiture */
    char alignement;  /* 'g'=>gauche ou 'm'=>milieur ou 'd'=>droite */
    char type;        /* 'v'=>voiture, 'c'=>camion, etc. */
    char couleur[10]; /* Couleur du véhicule */
    char custom[30];  /* Contient le véhicule customisé */
    char etat;        /* État du véhicule => actif ou inactif */
    /* Vous pouvez rajouter d'autres variables si nécessaire */
};
-------------------------------------------------------------------------
```

Pour faire bouger le véhicule, ici on utilise la touche 5 pour avancer, 2 pour reculer, 1 pour aller à gauche et 3 pour aller à droite. Vous aurez besoin d'une fonction pour récupérer la touche saisie par l'utilisateur sans pour autant retarder l'affichage car, le plan de circulation doit être dynamique. Vous pouvez pour cela utiliser la fonction `key_pressed` vue en première année :

```
-------------------------------------------------------------------------
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;
}
-------------------------------------------------------------------------
```

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 programme. 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`

Infos pratiques

Le projet est à réaliser en binôme sous environnement GNU/Linux, et doit être déposé sur la plate-forme pédagogique *Moodle* au plus tard le 30/11/2016 à 23h59, 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 ;
- 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 **Manuel**.
3. Test du mode **Automatique**.
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`