Cycle Fondamental
3A Semestre 5
LAB3040

ESIEA Paris/Ivry-sur-Seine
13 Septembre 2021

Projet Autonome en Programmation C

(Encadrement : Michael FRANÇOIS & Geoffroy VAN ELSUVE)

------ SPACE INVADERS -----

1 Présentation générale

L'objectif de ce projet est de programmer en langage ℂ, le célèbre jeu "Space Invaders". L'objectif est de détruire une vague de vaisseaux ennemis lançant des missiles. Le joueur gagne des points à chaque fois qu'il détruit un envahisseur. Le projet est à réaliser obligatoirement en binôme sous environnement GNU/Linux. L'affichage doit être effectué uniquement sur console. La FIG 1 montre l'aspect général du jeu, pour une réalisation sur le terminal.

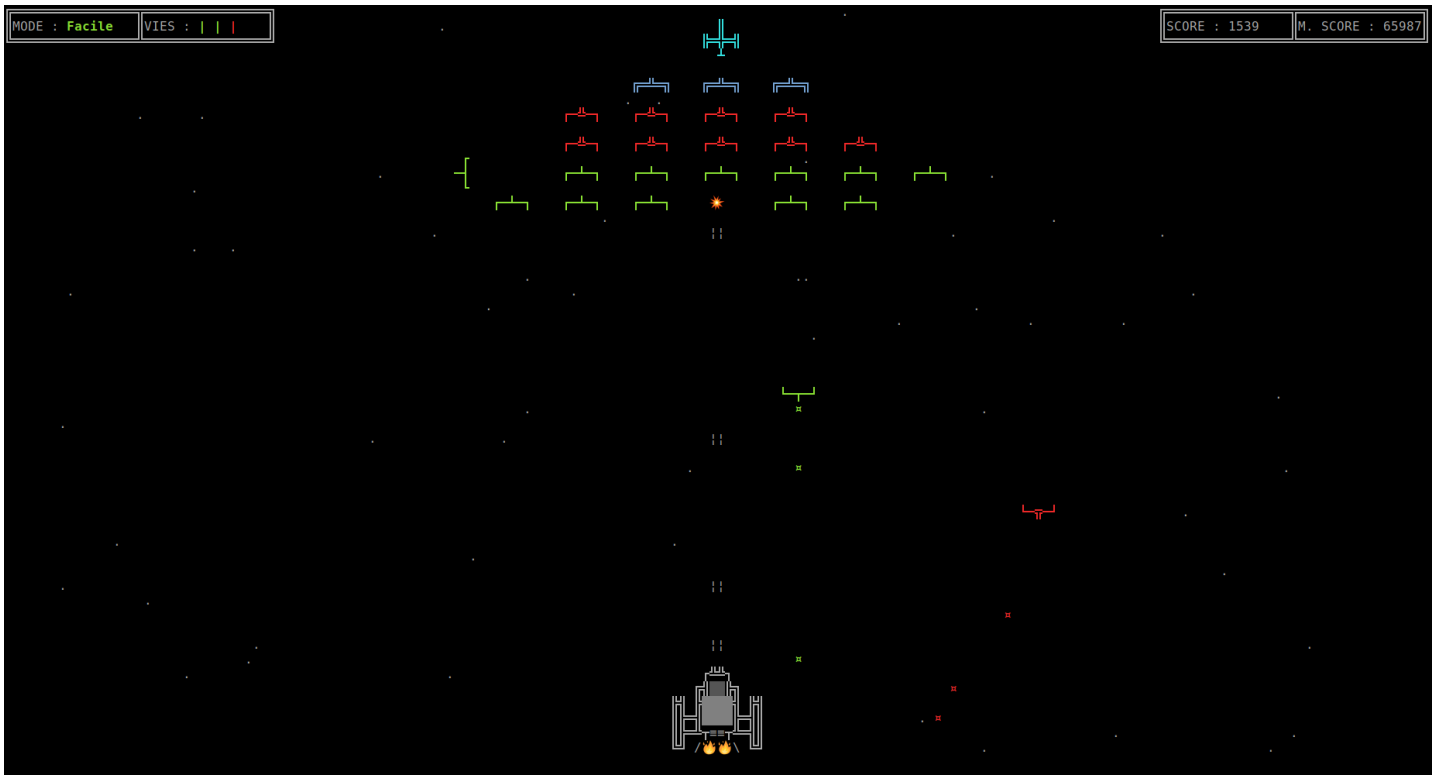

FIGURE 1 – Exemple de plan de station de métro.

Sur le plan on peut distinguer :

- un grand vaisseau blanc appartenant au joureur ;
- des vaisseaux rangés au dessus appartenant à l'ennemi ;
- parmi les vaisseaux de l'ennemi, on peut constater que ces derniers diffèrent en fonction de leur blindage ;
- quelques indicateurs concernant le mode en cours, le nombre de vies restantes, le score, etc.
- des étoiles pour symboliser l'espace.

Pour rendre le jeu plus réaliste, vous pouvez faire bouger les étoiles de manière ordonnée pour donner l'impression de déplacement dans l'espace.

Concernant les vaisseaux ennemis, en fonction du blindage, le vaisseau doit être atteint plusieurs fois par un missile avant d'exploser définitivement.

Comme pour le vrai jeu, l'alignement des vaisseaux ennemis par rapport au vaisseau du joueur ne doit pas être statique. En effet, la plateforme des vaisseaux ennemis bouge d'un cran latéralement au fur et à mesure que je jeu avance.

Pour rendre votre jeu plus fun, 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 simulateur.

Au lancement, l'utilisateur doit pouvoir choisir au minimum entre trois modes :

- **Facile :** moins de vaisseaux ennemis blindés, le vaisseau du joueur lance des missiles rapidement, et le jeu peut terminer après avoir atteint un certain score ou si le joueur ne possède plus de vies.
- **Difficile :** plus de vaisseaux ennemis blindés, le vaisseau du joueur lance des missiles avec une cadence normale et le jeu peut terminer après avoir atteint un certain score ou si le joueur ne possède pas de vies.
- **Progressif :** ce mode combine les deux précédents modes avec un début de jeu facile et le jeu devient difficile de manière progressive. Dans ce cas de figure le jeu ne se termine pas tant que le joueur est en vie.

Pour plus d'inspiration, vous pouvez trouver plusieurs liens sur internet, pour voir comment se joue le jeu "Space Invaders".

2 Design et création des vaisseaux

Avant de commencer la partie programmation, le premier travail consiste à dessiner d'abord **vos propres modèles de vaisseaux**. Le plus difficile dans cette phase est de mettre en place des modèles identiques quel que soit la direction du vaisseau. Plus le modèle est simple plus il est facile de créer les 4 modèles pour chacune des directions (Nord, Sud, Est, Ouest).

- D'abord le modèle de vaisseau du joueur qui ne doit être que dans un sens et qui ne bouge que latéralement pendant la partie. Rien ne vous empêche de proposer au joueur plusieurs modèles de vaisseaux au choix avant le début d'une partie.
- Concernant les modèles des vaisseaux ennemis, il doit y avoir 4 types. Du vaisseau le moins blindé au vaisseau maître le plus blindé. Comme indiqué au dessus, plus le vaisseau est blindé plus il résiste aux tirs du joueur.

Vous pouvez dessiner vos vaisseaux directement dans un fichier `.txt` en utilisant des caractères ASCII étendus pour un meilleur rendu. Un fichier par modèle de vaisseau. Concernant les couleurs elles seront gérées pendant l'affichage.

3 Aspects techniques

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

Chaque modèle de vaisseau se trouve dans un fichier "`.txt`" à part, et doit être chargé au lancement du jeu. Voici un exemple de structure utilisée pour rassembler toutes les informations liées à un vaisseau :

```
-------------------------------------------------------------------------
typedef struct vaisseau VAISSEAU;
struct vaisseau
{
  char Direction ;            /*N => Nord, S => Sud, E => EST, O => OUEST*/
  int PosX;                   /*Position courante coin gauche X du vaisseau*/
  int PosY;                   /*Position courante coin gauche Y du vaisseau*/
  int Blindage;               /*Niveau de blindage en cours du vaisseau
                               (0=>rien, 1=>blindé, 2=>ultra-blindé, etc.)*/
  int Touches;                /*Nombre de fois que le vaisseau est touché par
                                un missile*/
  char Carros_ennemi [3][16];/*Carrosserie du vaisseau ennemi, servira pour
                                l'affichage du vaisseau ennemi à tout moment*/
  char Carrosserie [6][40];  /*Carrosserie du vaisseau du joueur, servira pour
                                l'affichage du vaisseau du joueur à tout moment*/
  char Type;                  /*'M'=> mon vaisseau, 'E'=> vaisseau ennemi*/
  char Couleur[30];           /*Couleur du vaisseau*/
  int Etat;                   /*État du vaisseau 1=> actif, 2=> en destruction,
                                3 => inactif*/
  int Mise_a_jour;            /*utile pour la suppression du vaisseau en tenant
                                compte d'un certain délai*/
  struct vaisseau * NXT;      /*Pointeur vers un prochain vaisseau, servira
                                pour la liste chaînée*/
  /*Vous pouvez rajouter d'autres variables si nécessaire */
};
-------------------------------------------------------------------------
```

Rien ne vous empêche de créer deux structures au lieu d'une seule, comme indiqué au dessus. Du coup, une pour le vaisseau du joueur (qui contiendra sans doute moins de variables) et l'autre pour l'ennemi.

Vous aurez sans doute besoin d'autres structures, notamment pour les missiles (qui contiennent plusieurs types d'infos comme par exemple : la position, la direction ou même la provenance de l'obus). Pour faire vos choix dans le menu ou même pendant le déroulement, vous pouvez utiliser la fonction `key_pressed()` qui est la suivante :

```
-------------------------------------------------------------------------
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 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 vaisseau/missile 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 correspondant à la carrosserie.

La commande :
`printf("\033[%d;%dH", x,y);`
déplace de curseur à la position `(x,y)` sur le terminal et attend le prochain affichage.

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 (vous avez plusieurs choix) :
`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. L'utilisation de cette librairie est tout de même difficile et demande beaucoup de maîtrise surtout des pointeurs.

4 Infos pratiques

Le projet est à réaliser obligatoirement en binôme sous environnement GNU/Linux, et doit être déposé sur la plate-forme pédagogique *Moodle* au plus tard le 05/12/2021 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.

Votre programme doit obligatoirement présenter les différentes thématiques suivantes :

- tableaux, pointeurs et allocation dynamique ;
- structures et liste chaînée. La liste chaînée peut être utilisée pour les vaisseaux ennemis ;
- fichier `makefile` contenant les commandes de compilation de votre projet ;
- des lignes de codes commentées, lisibles et bien agencées ;
- un choix cohérent de noms de variables.

L'évaluation sera globalement scindée en 5 parties :

1. la présence des différentes thématiques indiquées au dessus (**liste chaînée, makefile**, etc.) ;
2. le **design** de votre jeu dans l'ensemble (vaisseaux, décor, menu d'accueil, etc.) ;
3. test du mode **Facile** ;
4. Test du mode **Difficile** ;
5. Test du mode **Progressif**.