IN301 – Langage C
Slider – Projet de Programmation

Nesrine Ben Hassine, Pierre Coucheney, Maël Guiraud, Franck Quessette

11 novembre 2017

Table des matières

Le but est de programmer le jeu Slider pour pouvoir non seulement jouer mais également créer des niveaux.

1 Le principe du jeu

Un niveau de Slider se joue à un joueur sur une grille formée de $L \times H$ cases. Sur les bords extérieurs il y a des murs et il peut y avoir également des morceaux de mur entre deux cases adjacentes. Il y a également un slider et une sortie et le but est d'emmener le slider sur la case de sortie.

Le contrôle du slider se fait avec les flèches. À chaque appui sur une flèche, le slider se déplace dans la direction de la flèche jusqu'à ce qu'il rencontre un mur ou la sortie.

2 Objectifs du projet

Les objectifs du projets sont de pouvoir :
— jouer à un niveau ;
— revenir en arrière dans les coups déjà joués ;
— enchaîner plusieurs niveaux ;
— créer des niveaux.

3 Décomposition du projet

3.1 Structure de données

Vous devez définir une structure de données :

```
struct slider {
      ...
      };
typdef struct slider SLIDER;
```

permettant de stocker le jeu à un instant donné, c'est à dire : la grille, la position du slider et celle de la sortie, le nombre de coups joués et tout ce que vous jugerez nécessaire, en particulier pour pouvoir revenir en arrière dans les coups déjà joués. Vous pouvez bien sûr définir d'autres structures de données si c'est nécessaire.

3.2 Fichier

La position de départ d'un niveau est stockée dans un fichier dont l'extension doit être `.slider`. Le format de stockage est le suivant :

```
Largeur Hauteur
x_slider y_slider
x_sortie y_sortie
N
x1 y1 d1
x2 y2 d2
...
xN yN dN
```

Où `N` est le nombre de murs et pour chaque mur, on donne les coordonnées d'une case (`xi`,`yi`) et sur quel bord de cette case est le mur : `di`, avec la correspondance suivante :

```
di = 0 ⟹ Mur à en haut
di = 3 ⟹ Mur à droite
di = 6 ⟹ Mur en bas
di = 9 ⟹ Mur à gauche
```

Avec cette notation, pour mettre un mur entre la case $(3, 4)$ et la case $(3, 5)$, on peut le faire avec : `3 4 0` ou bien `3 5 6`.

Il est imposé que les numéros des cases commencent à 0.

3.3 Interface

L'interface doit utiliser `uvsqgraphics`, permettre d'afficher le jeu et de jouer. Elle servira également à la création de niveau.

Pour l'interface de création de niveau, l'utilisateur utilisera la souris pour ajouter et supprimer des murs et pour positionner le slider et la sortie. La touche `S` permettra d'écrire le niveau dans un fichier.

Pour l'interface, vous utiliserez la fonction :

```
int wait_key_arrow_clic (char *touche, int *fleche, POINT *P);
```

qui permet d'attendre les flèches ou une touche, par exemple, la touche `U` pour faire le `undo`.

3.4 Ligne de commande

La syntaxe de la ligne de commande est :

— Pour jouer à un niveau stocké dans le fichier `fichier.slider` :

```
slider fichier.slider
```

— Pour jouer à tous les niveaux stockés dans le dossier `dir_slider` :

```
slider dir_slider
```

— Pour créer un niveau de taille $L \times H$ et le stocker dans le fichier `fichier.slider` :

```
slider -c L H fichier.slider
```

4 Notation du projet

Dans la notation du projet il sera très fortement tenu compte :

— du découpage pertinent en fonctions et en fichiers ;
— de la longueur des fonctions (moins de 15-20 lignes sauf exception) ;
— du nommage pertinent des structures de données, des variables et des fonctions ;
— de la présentation du code. En particulier vous pouvez utiliser la commande indent :
`indent -linux fichier.c`
pour aligner votre code proprement. L'idéal étant d'intégrer cette commande au `Makefile` ;
— de la gestion parcimonieuse de la mémoire. En particulier toute la mémoire devra être libérée avant de quitter le programme ;
— de la qualité du Makefile ;
— de la qualité des commentaires (peu nombreux, mais pertinents) ;
— de la lisibilité de l'affichage.

Dans la notation du projet il sera très très faiblement tenu compte :

— de la qualité esthétique de l'affichage.

Vous pouvez ajouter des fonctionnalités supplémentaires, mais celles-ci n'auront pas d'impact sur la notation. Par exemple détecter que le slider est dans une position où il est impossible d'atteindre la sortie.

5 Modalités pratiques

Vous devrez rendre dans un dossier compressé en `.zip` ou `.tgz` qui doit s'appeler :

```
VOTRE_NOM.zip
ou
VOTRE_NOM.tgz
```

tous les fichiers sources et les fichiers nécessaires à la compilation (Makefile ou autre), et également des fichiers et/ou dossiers de niveau qui pourront être lancés en faisant :

```
$ make test
```

- Le fichier `.zip` ou `.tgz` doit être obligatoirement déposé sur e-campus. Aucun envoi par email ou remise sur clé USB ne sera accepté.
- La date limite de dépôt est le **lundi 15 janvier 2018 à 08h00 (Date à confirmer)**. Toute seconde de retard sera sanctionnée de $\frac{1}{2^5 3^3 5}$ point en moins, le projet étant noté sur 20.
- Une présentation et une démonstration de votre projet aura lieu le jeudi 18 janvier 2018 (Date à confirmer).
- Le projet est à faire individuellement.
- Le projet est à faire en langage C à l'exclusion de tout autre langage.
- Un logiciel de comparaison de code sera appliqué à l'ensemble des projets rendu. Ce logiciel permet de détecter très facilement les projets similaires.

Ce projet est long et il est impossible de le faire sérieusement en s'y mettant une semaine avant la date de remise. Il faut donc le commencer le plus tôt possible.

De plus les statistiques des années précédentes montrent que ceux qui avaient beaucoup avancé dans le projet avait 4 à 5 points de plus à l'examen que ceux qui ne l'avaient pas commencé ou peu commencé. L'explication est simple : faire le projet sérieusement est une manières très efficace de comprendre et de connaître le cours.

A Exemple de fichiers

Aller tout droit :

```
niveau_001.slider
_____________________________________________________________
1 20
0 0
0 19
0
```

Droite, gauche, droite :

```
niveau_002.slider
_____________________________________________________________
10 10
0 0
9 9
```

```
2
0 4 3
0 9 3
```