PROG
TP#3 : Le canyon de la mort

2022-W38/39

Instructions de rendu

Ce TP fait l'objet d'un rendu *en binôme*. Vous devez envoyer votre travail sous la forme d'une archive à :
philip.scales@univ-grenoble-alpes.fr ;
au plus tard le 2022-09-30 à 18 :00. Celle-ci doit contenir :

- Votre programme implémentant le jeu décrit ci-dessous, où *chaque appel à une fonction de bibliothèque externe devra être commenté pour expliquer son rôle.*
- Un Makefile et les instructions permettant de l'utiliser.
- La documentation des éventuelles extensions implémentées.

Présentation

Le but de ce TP est d'implémenter le jeu du « canyon de la mort » dont le principe est le suivant : un.e cycliste (représenté.e par un `'^'` en bas de la fenêtre d'un terminal) doit descendre un canyon sans en heurter les bords (représentés par des `'*'`). Au début d'une partie les bords du canyon sont parallèles et remplissent la totalité de la fenêtre et le ou la cycliste est centré.e, cf. Figure 1.

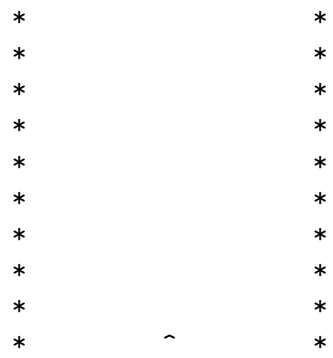

FIGURE 1 – Configuration de début de partie.

Une fois la partie commencée, à chaque coup d'horloge le personnage descend le canyon d'une position. Ceci est illustré par le fait que chaque ligne est décalée d'une position vers le bas (sauf la ligne la plus basse qui ayant passé l'horizon n'est plus affichée) et qu'une nouvelle ligne est affichée à la position la plus haute ; le canyon étant sinueux, celle-ci est aléatoirement décalée d'un caractère vers la gauche ou vers la droite par rapport à la précédente ligne la plus haute avec probabilité 1/3 dans les deux cas (elle reste donc inchangée également avec probabilité 1/3). On donne par exemple dans la Figure 2 une configuration pouvant être obtenue sept coups d'horloge après la Figure 1.

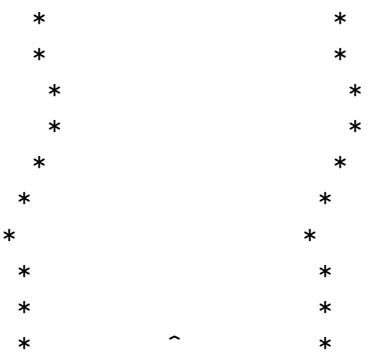

FIGURE 2 – Configuration possible après sept coups d'horloge.

Enfin le ou la cycliste peut être déplacé.e par l'intermédiaire des flèches gauche et droite du pavé numérique du clavier ; le but du jeu est alors d'éviter une collision avec un des bords du canyon. Quand cela se produit la collision est représentée par un `'X'`, la partie s'arrête, et le score correspondant au nombre de lignes franchies est affiché. Un exemple de position de fin de partie est donné dans la Figure 3.

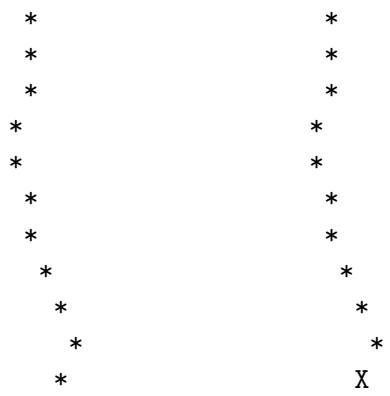

FIGURE 3 – Configuration possible de fin de partie.

Instructions

Le jeu doit être développé sous environnement UNIX à l'aide de la bibliothèque ncurses (dont les entêtes sont définis dans `"ncurses.h"` et qui doit être « liée » à la compilation via `-lncurses`). Celle-ci permet de manipuler un environnement de fenêtre et donne accès à un curseur et à des entrées/sorties non-bloquantes. Les contraintes à respecter sont les suivantes :

— La largeur du canyon doit être configurable à la compilation. On conseille d'utiliser par défaut une largeur de 20 caractères.
— La fréquence de rafraîchissement (c-à-d la fréquence de l'horloge) doit être configurable à la compilation. On conseille d'utiliser par défaut une fréquence de 50 Hz.
— L'affichage doit se faire sur l'intégralité de la fenêtre de terminal courante (le cas où la largeur de celle-ci est inférieure à celle du canyon peut être traité librement) ; en particulier le canyon ne doit pas dépasser les bords de la fenêtre !
— Le curseur ne doit pas être visible.
— Par défaut, le canyon doit être différent à chaque lancement du programme.

— Une fois une partie terminée, la fenêtre doit montrer la position finale obtenue et indiquer le score en nombre de lignes parcourues avant la collision. Le programme doit quitter et rétablir *proprement* l'affichage normal du terminal après pression de la touche `q`.

Conseils & indices. L'ensemble du jeu peut aisément s'implémenter dans une unique fonction `void game(void)` de moins de 100 lignes. Vous pouvez par exemple structurer celle-ci en quatre étapes principales : l'affichage de l'état courant ; la modification éventuelle de la position du ou de la cycliste ; l'avancement du canyon ; le test de collision [1].

Quelques fonctions, variables, et macros pouvant être utiles sont :

— `initscr()`
— `keypad()`
— `noecho()`
— `getch()`
— `refresh()`
— `mvprintw()`
— `nodelay()`
— `endwin()`
— `curs_set()`
— `usleep()`
— `stdscr`
— `LINES`
— `COLS`
— `KEY_LEFT`
— `KEY_RIGHT`

À vous de vous renseigner sur leur utilisation et sur les fonctions supplémentaires éventuellement requises.

Suggestions d'extension. S'il vous reste du temps, vous pouvez en bonus par exemple implémenter :

— Un mode « démonstration » automatique.
— Un mode multijoueur.
— Une gestion des plus hauts scores, avec un *hall of fame*.
— Un menu (utilisant ncurses) permettant de régler la difficulté via plusieurs paramètres.
— Un canyon de largeur variable.
— ...

1. Faites cependant attention à effectuer ce test autant de fois que nécessaire.