Cours de programmation séquentielle

Le jeu du serpent

1 Buts

— Utilisation d'une file d'attente.
— Implémentation d'un jeu.
— Utilisation de fonctions et de SDL2.
— Allocation dynamique de mémoire.
— Utilisation de `git` et de `make`.

2 Énoncé

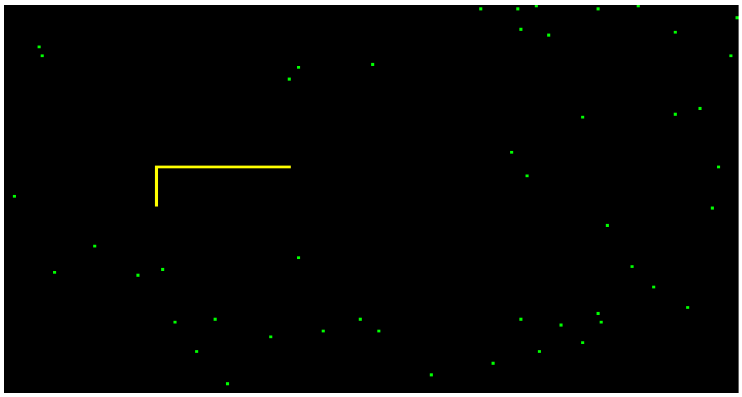

FIGURE 1: Le jeu du serpent, tribute to nokia 3210.

Implémenter un "jeu du serpent" (voir fig. 1) avec les règles suivantes:

— Le serpent avance toujours tout droit (dans sa direction courante) tant que la joueuse ne modifie pas la direction.
— La joueuse peut modifier la direction de déplacement du serpent à l'aide des touches suivantes [1]:
  — `a`: déplacement vers la gauche.
  — `s`: déplacement vers le bas.
  — `d`: déplacement vers la droite.
  — `w`: déplacement vers le haut.
— Le serpent démarre avec une longueur de 3.
— Le serpent démarre en étant orienté vers la droite au milieu de l'écran de jeu.

1. Vous pouvez aussi utiliser les flèches.

— Il y a un nombre maximal, `N`[2], de nourriture qui apparaît à des endroits aléatoires de la carte toutes les `M sec`[3].
— Lorsque le serpent "mange" de la nourriture sa longueur augmente de un.
— L'écran de jeu est entouré d'une paroi.
— Le serpent meurt si:
  — sa tête rencontre une paroi.
  — si sa tête rencontre son corps.
  — si le serpent tente de faire un "demi-tour" immédiat (p.ex. il se déplace vers la droite et on essaie de le faire se déplacer vers la gauche).
— La partie se termine par une victoire si le serpent occupe tout l'écran.
— La touche `escape` arrête le jeu.

L'intéraction avec la joueuse (affichage graphique, touches du clavier) se fait avec la librairie SDL2. Pour l'affichage, il est recommandé d'avoir des épaisseurs de un pixel pour tout ces objets (cela simplifie grandement l'implémentation).

Bien entendu vous devez utiliser `git` et `make` pour la gestion et la compilation de votre projet.

3 Cahier des charges

Chaque segment du serpent est en fait un pixel: il est représenté par sa position (ses coordonnées `x`, `y` qui sont des entiers). Le serpent est en fait une file d'attente contenant la position des pixels qu'il occupe. Lorsque le serpent se déplace, il faut retirer l'élément du devant de la file, et insérer le nouvel élément à l'arrière de la file: la tête du serpent est l'arrière de la file, et la queue du serpent est l'avant de la file.

Sans intervention de la joueuse, le serpent ne changera pas de direction de déplacement. Il faut donc stocker cette information. La nourriture ne se déplace elle jamais, et est uniquement déterminée par sa position sur l'écran.

Afin de déterminer si le serpent meurt ou mange de la nourriture, il est recommandé d'utiliser le tableau de pixels utilisé pour l'affichage. Vous pouvez définir un type énuméré avec quatre variantes (`empty`, `snake`, `food`, et `wall` par exemple) correspondant à des couleurs `COLOR_BLACK`, `COLOR_WHITE`, ... Ainsi en récupérant la valeur des pixels vous pouvez savoir quel type d'élément se trouve à une position donnée.

Pour que le jeu soit "jouable", il faut éviter de mettre à jour son état trop souvent (le serpent bougerait trop vite). Il est judicieux de n'afficher qu'un certain nombre de frames par seconde (ou en d'autres terme fixer le temps qu'il faut entre l'affichage de chaque frame, t_f). Pour ce faire, il faut mesurer le temps nécessaire à l'affichage d'une frame (une méthode pour mesurer le temps passé dans une section du code vous a été proposée il y a quelques semaines), puis faire attendre votre programme le temps nécessaire pour qu'on attende t_f avant l'affichage de la frame suivante.

2. Un argument de votre programme.
3. Autre argument de votre programme.

3.1 Avant de commencer

Cette partie est particulièrement importante. Ce travail pratique est assez peu "guidé", il faut donc bien réfléchir avant de commencer à implémenter quelque chose.

1. Mettez-vous par groupes de 5.
2. Faites un schéma, une liste, etc avec les différentes fonctions et structures de données que vous allez utiliser.
3. Écrivez les *headers* contenant les structures de données et les signatures des fonctions.
4. Écrivez un `Makefile` qui compile votre ébauche de projet (même s'il y a rien dans les fichier à part un main vide).
5. Allez vous pouvez vous y mettre!

4 Remarques

Vous trouverez sur `Cyberlearn` des fichiers `gfx.h` et `gfx.c` qui ne sont pas exactement ceux des travaux pratiques précédents (il faut donc les re-télécharger).

Un certain nombre des fonctions suivantes pourraient vous être utiles (elles ne sont pas toutes dans `gfx.h/c`):

— `gfx_getpixel()`, retourne la "couleur" du pixel à une position `x`, `y` donnée en argument.
— `gfx_putpixel()`, donne une "couleur" donnée en argument au pixel se trouvant à une position `x`, `y` donnée.
— `gfx_keypressed()` retourne le code de la touche pressée par la joueuse[4].
— `usleep()` (se trouvant dans `unistd.h`) met le programme en pause pendant un certain nombre de microsecondes passé en argument.

4. Tous les codes clavier se trouvent à l'adresse https://wiki.libsdl.org/SDL_Keycode.