Xiong Adele Yue
Trouet Kilian
Groupe 8

Rapport TP5 Programmation en C

I Introduction

Ce tp nous à permis de traiter plusieurs notion différentes du cours: liste chaînées, partie graphique, modularisation, ouverture/lecture de fichier.

II Questions

Exercice 1:

2) Nous avons décidé de découper ce tp en six modules différents. Tout d'abord les modules case, serpent monde et graphique sont les modules de base, puis nous avons ajouté deux modules (Config et Sound) pour les améliorations.
Le module case regroupe les structures et fonctions en rapport avec les cases. Serpent regroupe les fonctions et la structure de type serpent. Graphique contient toutes les fonctions graphiques et monde contient la structure monde et les fonctions principal pour le déroulement du jeu.

3) Au fil du projet nous avons adapté nos modules pour y ajouter un module sound et config. Ces modules ayant un rôle avec les améliorations.

4) Cf makefile

Exercice 2:

1)Le type case possède deux coordonnées. Abscisse et ordonnées.
Ce type sera utilisé pour matérialiser des cases pour le serpent et les pommes.

2)Direction possède les quatres points cardinaux. (Nord, Sud, Est, West).

3)Serpent possède une direction et une liste chaînée de cases.

4)Le type pomme est le même type que case. Par la suite pour générer une liste chaînée de pommes nous utiliserons la structure Cases contenant une case et un pointeur vers le prochain élément de Cases. Le type Cases est également utilisé pour la liste chaînée de dans la structure Serpent.

5)Le type Monde possède trois int (lignes, colonnes et nb_pommes), le type serpent et le type Cases appelées par Liste à fin de matérialiser une liste chainées pour les pommes.

Exercice 6: Amélioration

Nous avons opté pour les amélioration 1(*), 2(*), 3(*), 7(**) et 9(***).

La première amélioration consistait à pouvoir mettre en pause une partie en appuyant sur ESPACE. Nous avons ajouté un case ' ' dans le switch de la fonction interface_piloter.

Pour les améliorations 2 et 3, nous avons dû modifier la structure de Case à fin d'y ajouter un Type de case que nous définissons par énumération. Etant donné que nous stockons simplement 4 types différentes de cases dans les listes chaînées (Serpent, Pomme normal, Pomme empoisonnée et Pomme "Super"). Par conséquent nous avons dû modifier les fonctions qui avaient un impact sur le jeu lors de la rencontre d'une pomme.
Une super pomme (pomme blanche) permettant d'augmenter le score de 2.
Une pomme empoisonnée (pomme magenta) cause la défaite instantané du joueur.

L'amélioration 7 nous a donné quelques difficultés. Etant donné que ncurses ne possède pas de gestion du son. Nous devions définir une librairie que nous pouvions utiliser pour réaliser cette amélioration. Nous avons supposé que nous étions obligés d'utiliser la libmlv. En premier temps nous voulions passer par MVL_MUSIC puis on s'est rendu compte que MLV_SOUND était mieux adapté. Suite à cette amélioration nous avons créé une nouvelle structure Sound permettant de stocker les sons du jeu, ainsi qu'une fonction init_sound permettant de charger les sons nécessaires

L'amélioration 9 a été pour nous la plus simple et rapide à réaliser (après l'amélioration 1). Nous avons créé un module Config contenant une structure permettant de garder en mémoire les informations de la partie, tel que la hauteur, largeur, nombre de pomme, nombre de pomme empoisonnée/super pomme, taille du serpent et le taux de rafraîchissement (duree_tour). Puis nous avons écrit une fonction d'initialisation de Config qui ouvre un fichier et récupère les informations de paramétrage. Une fois les informations initialisées dans une struct Config nous les utilisons pour initialiser le monde (la partie de jeu).

III Difficultés rencontrées

Nous avons rencontré des difficultés lors de l'assemblage du programme, car certaines fonctions ne pouvaient pas être testées avant. A plusieurs reprises nous avons dû modifier les structures afin de ne pas manipuler des structures trop complexes et ne pas perdre de temps, nous avons essayé de regrouper les informations dans les structures en gardant une lisibilité et une compréhension peu complexes.

IV Répartition du travail

En premier temps, afin d'avoir une vision globale du travail à réaliser. Nous avons décidé de travailler ensemble pour l'écriture des structures. Une fois que nous avions avancé ensemble aux premières questions de l'exercice 3, nous nous sommes réparti le travail. Adèle s'est occupée de réaliser les fonctions de l'exercice 3., et moi (Kilian) de

réaliser les fonctions de la partie graphique. Puis lors du travail à la maison, Adèle s'est occupée d'assembler le jeu (exercice 5), de corriger les erreurs graphiques (stylisé le plateau) et s'est occupée des améliorations une et deux. Pour ma part je me suis chargé de corriger les bugs restants (ex: les pommes qui ne pouvaient pas être mangées), du makefile ,puis de l'amélioration 9 et 3 (au départ l'amélioration 2 ne fonctionnait que pour une simple pomme empoisonnée. J'ai donc modifier l'amélioration d'Adèle en y ajoutant un type cases. Permettant de répondre aux améliorations 2 et 3). Adèle s'est également occupée de l'amélioration 7 (ajout de son) mais n'a pas pu la tester en raison de problèmes avec la libmlv. J'ai donc ajusté l'amélioration pour la faire fonctionner.

V Instruction de compilation

Programme	make Snake

VI Instruction d'utilisation

Programme	./Snake