Rapport d'avancement

Snake – **Louis BRUNET**

Groupe 1

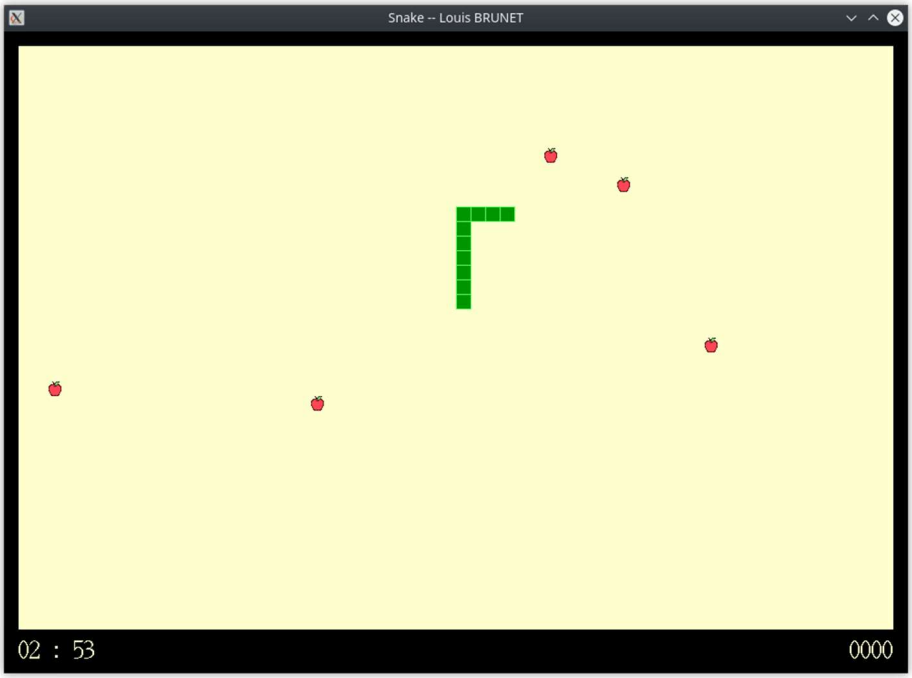


Table des matières

Introduction

Description du sujet

Ce projet tutoré fut réalisé pour les cours d'APL à l'IUT de Fontainebleau. Ce *Snake* a été développé suivant certaines contraintes :

- **Les règles**

Les règles du jeu ressemblent à celles d'un *Snake* classique : un serpent avance sans cesse sur une grille, pouvant tourner à droite ou à gauche selon la saisie de l'utilisateur. Quand le serpent se mord la queue ou sort du terrain, la partie est terminée. Quand le serpent se déplace sur une pastille, sa taille augmente et la pastille est « mangée », faisant apparaître une nouvelle pastille sur une case non occupée du terrain, et augmentant le score du joueur. Le jeu peut être mis en pause avec la touche *Espace* ou arrêté à tout moment avec la touche *ESC*.

En dehors du terrain de jeu, dans la portion inférieure du cadre autour de celui-ci, sont visibles le temps restant de la partie actuelle, ainsi que le score actuel du joueur.

- **Les constantes imposées**

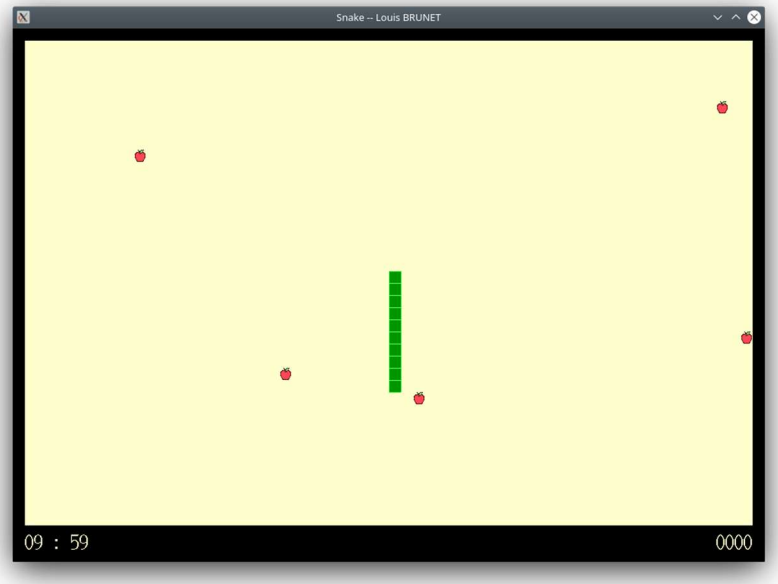
Certaines valeurs sont imposées par le sujet. La grille est composée de 40 lignes et 60 colonnes et contient 5 pommes initialement. De plus, la taille initiale du serpent est de 10 cases. Manger une pastille augmente sa taille de 2 et rajoute 5 points au score du joueur.

- **Variantes**

Le sujet impose également l'addition de variantes, ou règles supplémentaires, du jeu par rapport à une partie classique de *Snake*. J'ai choisi d'intégrer trois règles additionnelles :

- La vitesse de jeu augmente en fonction du temps écoulé. Quand 1/3 du temps imparti est écoulé, le jeu se déroule 2x plus vite, et 3x plus vite après 2/3 du temps maximum.
- Toutes les trois pommes mangées, un obstacle (représenté par le feu de bois) apparaît à l'écran. La partie est perdu si le serpent essaie d'avancer par-dessus.
- Si le score obtenu à la fin d'une partie est dans le « top 10 » des records sur la machine qui exécute le programme, alors le jeu demande à l'utilisateur de saisir un nom sous lequel sauvegarder ce record.

Fonctionnalités du programme



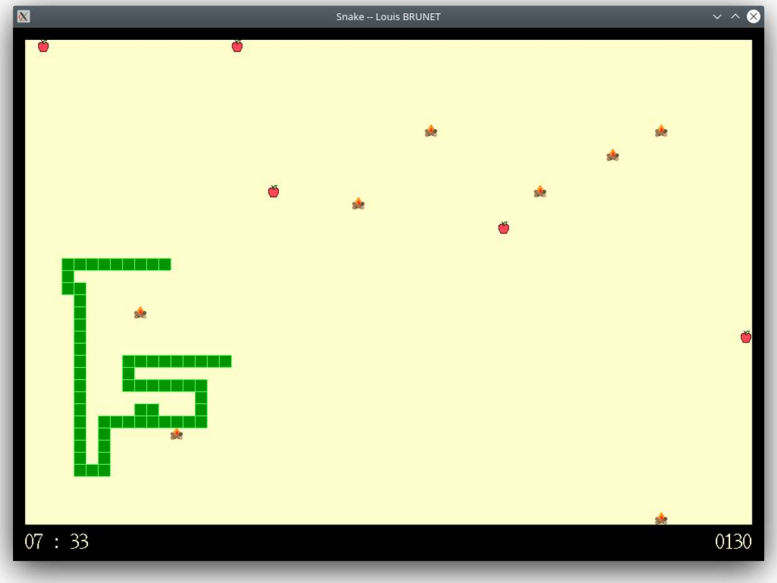
Début de partie

En début de partie, le serpent mesure 10 segments et est en mouvement vers le haut. Cinq pastilles, représentées par des pommes, sont placées aléatoirement dans des cellules non vides du terrain. En dessous du terrain, le temps restant et le score sont visibles.

Effets des pommes

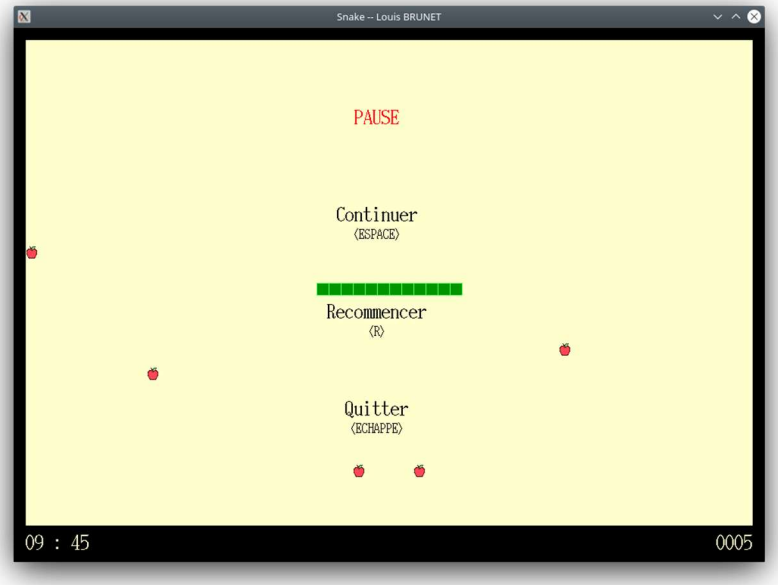
Quand le serpent mange une pomme, sa taille augmente de 1 à chaque itération de la boucle de jeu principale jusqu'à avoir atteint le gain de taille voulu (2 segments par pomme mangée).

Pour trois pommes mangées, un obstacle permanent est créé sur une case non vide du terrain .



Menus

Jeu en pause

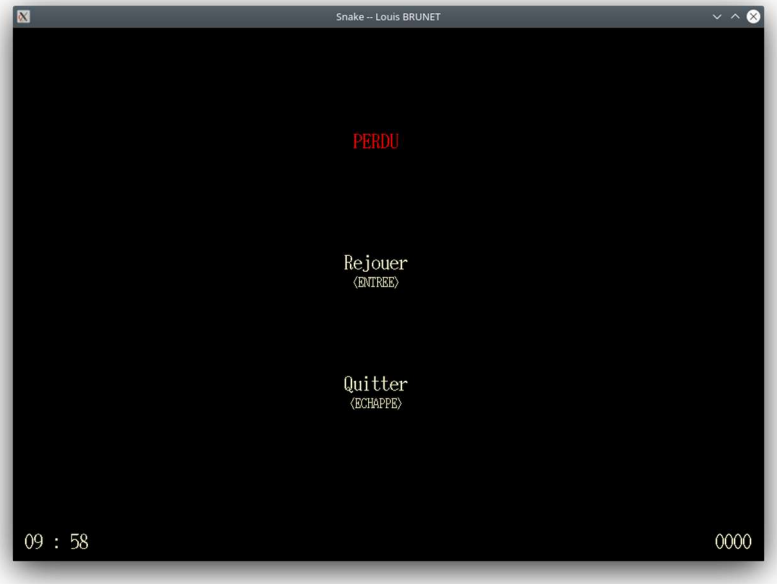


Quand la touche *Espace* est appuyée pendant la partie, le jeu est mis en pause et le menu ci-contre apparait. Le joueur peut ensuite choisir entre continuer (*Espace*), recommencer (*R*) ou quitter (*ESC*) la partie.

Si le joueur choisit de recommencer la partie, le terrain sera réinitialisé avec des pommes placées différemment, et sans vérifier si le score atteint est un nouveau record.

Comme à tout autre moment, si le joueur enfonce la touche *Esc*, la partie est terminée sans vérifier si le score doit être ajouté aux records, et la fenêtre est fermée.

Jeu perdu

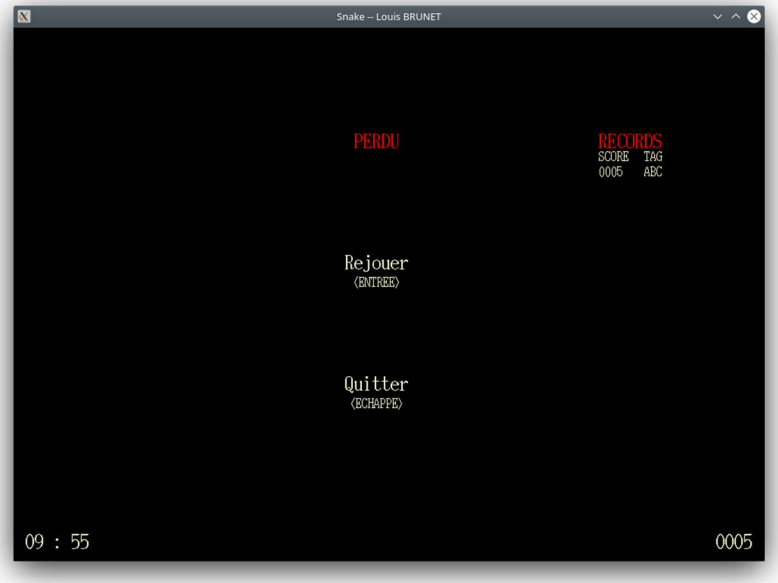


Si le joueur perd avec un score égal à zéro, alors le score n'est pas ajouté au fichier des records, même si celui-ci contient moins de 10 scores. Si ce fichier est vide, la table des records n'est pas affichée sur cet écran. Le temps restant et le score de la dernière partie sont affichés en bas de l'écran.

Depuis cet écran, le joueur a le choix de rejouer en appuyant sur *Entrée*, ou de fermer la fenêtre en appuyant sur la touche *Esc*.

Records

Affichage

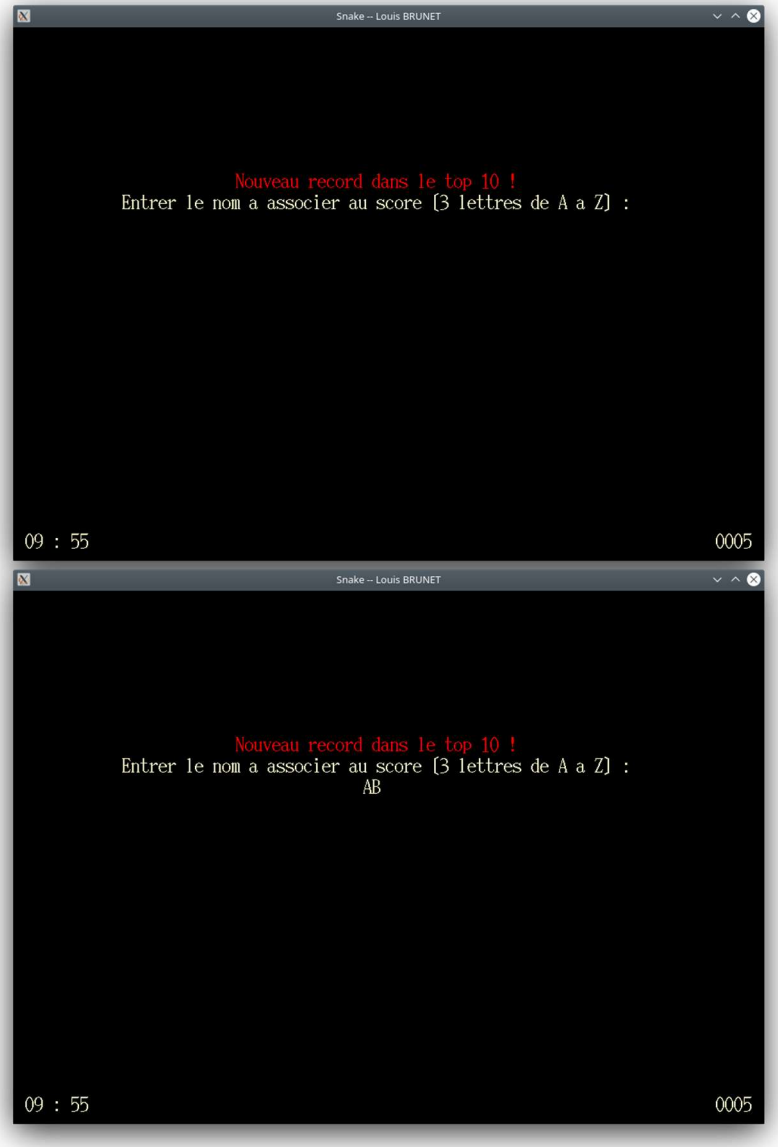


Si le fichier contenant les records n'est pas vide lorsque l'écran « PERDU » est chargé, alors une table des records qui y sont inscrits est affichée à l'écran. Le score est un entier à quatre chiffres en base 10, et le « tag » (nom associé au score) est composé de trois lettres.

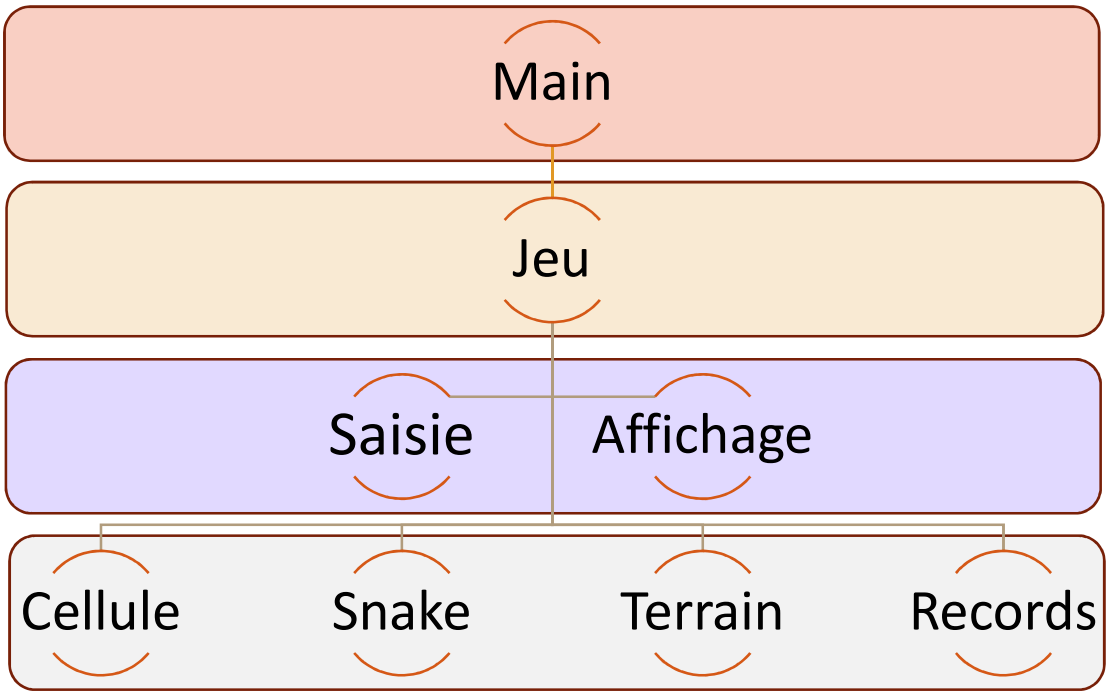
Saisie

A la fin d'une partie, si le score atteint est dans les dix meilleurs scores enregistrés sur la machine exécutant le programme, alors l'utilisateur est demandé un tag à associer au score atteint. Le jeu ajoute alors ce record au fichier records.txt de la machine

Les caractères saisis sont affichés en majuscule à l'écran.



Structure du programme



Dans le fichier cellule.h, on défini les type *Position*, *Cellule* et *ContenuCellule* (qui prend les valeurs SNAKE, PASTILLE, OBSTACLE, VIDE) qui sont utilisés dans la plupart des autres fichiers. Les fichiers snake.c et terrain.c contiennent respectivement les fonctions en rapport avec les types Snake et Terrain, notamment les différentes transformations que ceux-ci peuvent subir lors d'une partie.
Records.c gère la création et la lecture du fichier records.txt.

Le fichier affichage.c contient les fonctions capables de gérer l'affichage du cadre et du terrain. Le fichier saisie.c gère toute la saisie de l'utilisateur (pendant la partie, dans les menus, et lors de l'entrée du tag à associer à un record).

Le fichier jeu.c défini un type *Jeu* qui contient des attributs comme le terrain de jeu, le temps de restant, le score du joueur et différents états possibles du jeu (e.g. en pause, perdu). Ce fichier gère le déroulement de la partie quand le jeu n'est pas en pause ou perdu.

Le fichier main.c gère principalement la logique de la partie lorsqu'elle est en pause ou perdue, notamment l'initialisation du jeu, son affichage, sa réinitialisation éventuelle et la fermeture de la fenêtre.

Le serpent

Forme des données

La structure *Snake* définie dans snake.h est composée de trois attributs principaux :

- Sa **taille actuelle** (*int*) et sa **taille voulue** (*int*)
- Sa **direction actuelle** (*Direction*) et **sa direction voulue** (*Direction*)
- Un tableau de *Position* représentant les **positions des différents segments** du serpent dans le Terrain dans l'ordre de sa tête à sa queue.

Une *Direction* est un type défini dans cellule.h qui prend uniquement les valeurs HAUT, BAS, GAUCHE et DROITE.

Le type *Positions* est également défini dans cellule.h, et est composé de deux entiers correspondant à une abscisse et une ordonnée dans le *Terrain*.

Le *Terrain*, défini dans terrain.h, est composé d'un tableau à deux dimensions de *Cellules* (contenant chacune un *ContenuCellule* et une *Position*).

Transformations subies

A chaque rafraichissement de la boucle de jeu principale (ou chaque « frame »), le serpent peut subir trois opérations :

- Sa direction est mise à jour en fonction de sa direction voulue. Ceci sert à garder en mémoire la direction actuelle su serpent si plusieurs flèches directionnelles sont appuyées en une frame, afin d'éviter d'ajouter un nouveau segment sur le premier segment après la tête.
- Ajouter un segment en tête du serpent. Ceci consiste en l'ajout de la prochaine position du serpent (déterminée selon sa direction) au tableau de *Positions* du *Snake* à l'indice 0 (l'indice de la tête du serpent). Les positions suivantes sont donc décalées d'un indice et la taille du *Snake* est mise à jour. De plus, le contenu de la *Cellule* correspondante du *Terrain* est mis à jour, et la *Position* correspondante est ajoutée au tableau de positions changées du *Terrain*. Les cellules correspondant aux positions changées du *Terrain* sont réaffichées chaque frame.
- Si la taille du serpent était égale à sa taille voulue, retirer le dernier segment du serpent. Pour ce faire, retirer la dernière *Position* du tableau de positions du serpent et mettre à jour sa taille ainsi que les positions changées du *Terrain*

Conclusion personnelle

La partie la plus compliquée de ce projet a été l'installation de Linux et dual-boot avec Windows 10 pour pouvoir tester le code écrit depuis mon ordinateur. J'ai effacé près de 900GB de données en formattant le mauvais disque. Cela m'a permis d'apprendre les différents types de formatage de disques pour différents systèmes d'exploitation, comment gérer les différentes partitions des disques, et la configuration du menu grub.

La programmation de la logique du jeu n'a pas vraiment été un souci, puisqu'une grande partie de celle-ci était donné dans le sujet. Par conséquent, toutes les fonctionnalités obligatoires du programme étaient finies en un semaine. Un défi que j'ai dû surmonter était la gestion conditionnelle de la saisie dépendamment de la direction actuelle du serpent, dans le cas où plusieurs flèches directionnelles ont été appuyée durant une frame.

Ce projet était mon premier projet de cette taille en C. Il m'a permis de comparer les méthodes de programmation entre le C et le Java, que je connais beaucoup mieux en général.