Rapport SAÉ 1.1

Fabio ETOILE

William GENTIL

Le jeu du serpent

Sommaire :

|) Introduction :

Dans le cadre de notre 1 ère année de BUT Informatique, notre professeur de développement nous donne un projet à réaliser qui consiste à créer le fameux jeu du snake, voici une brève description du jeu :

- Le joueur contrôle un serpent dans une grille.
- L'objectif est de manger des pommes pour faire croître le serpent
- Les commandes de direction (haut, bas, gauche, droite) déplacent le serpent
- La partie se termine si le serpent entre en collision avec lui-même ou les bords de la grille
- Chaque pomme mangée augmente la longueur du serpent.

Sauf que nous devons le réaliser mais avec certaines contraintes :

- Un programme écrit en langage C, plus précisément du C89
- Utiliser aucun emprunt extérieur, mise à part la bibliothèque graphique de notre IUT
- Faire un terrain sous forme de grille de 40 lignes par 60 colonnes

- Placer 5 pastilles de la taille d’une case aléatoirement
- Un serpent formé de 10 segments (aussi de la taille d’une case) à placé au milieu
- Le joueur peut changer la direction du serpent à l'aide des touches directionnelles
- Il peut quitter la partie avec la touche Echap
- Faire en sorte que le joueur voit ses performances (score et temps) pendant une pause ou à la fin de la partie
- Et une variante, qui va consister à ajouter des obstacles fixes que le serpent devra éviter

II) Description des fonctionnalités :

Notre jeu comporte trois menus essentiel :

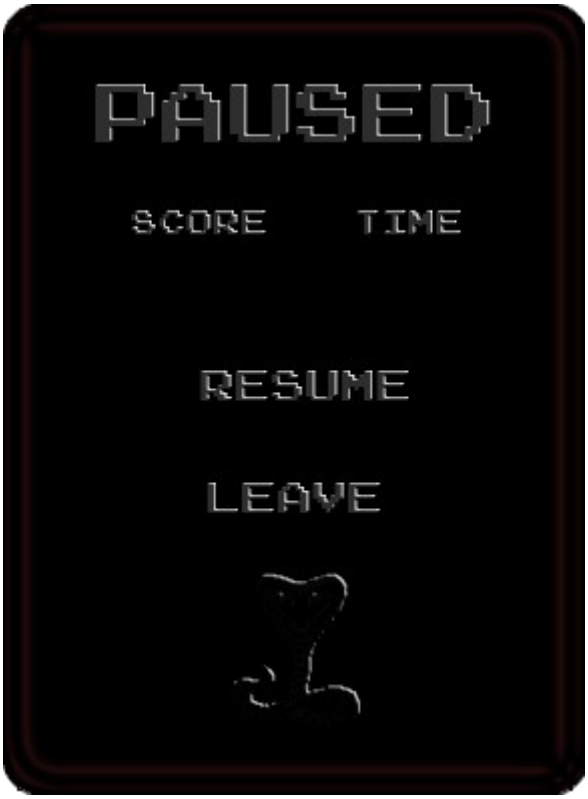
• Menu d’entrée de jeu

• Menu de pause

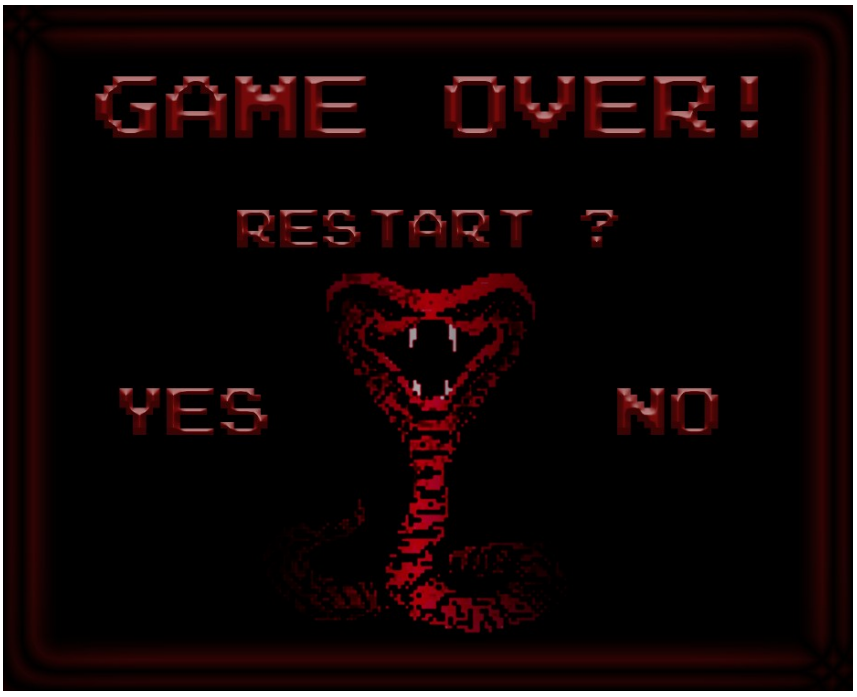
• Menu de fin de jeu

- Tout d’abord le menu d’entrée de jeu, il permet aux joueurs de soit quitter le jeu ou bien lancer une partie en faisant clique gauche sur « Play » pour jouer ou « Leave » pour quitter le jeu**.**

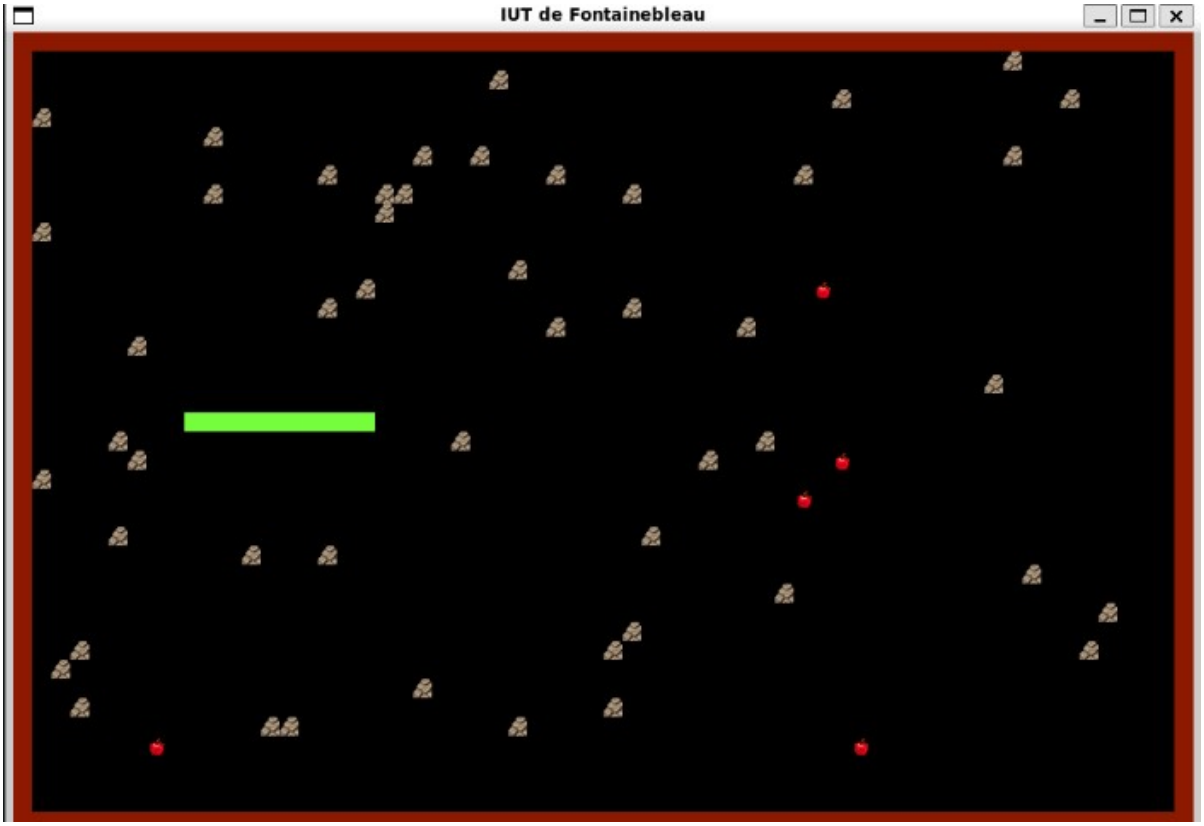
- Puis le menu pause, qui lui s’active lorsque le joueur appuie sur la touche « Espace », le menu pause affiche tout d’abord le score actuel du joueur avec le temps écoulé depuis le début de la partie. Il y a aussi d’autres fonctionnalités comme par exemple : en appuyant sur « Espace » pour reprendre la partie et en faisant « Echap » pour quitter le jeu.



- Et enfin le menu de fin de jeu, il affiche le score et le temps une fois la partie finie et laisse le choix au joueur de relancer une partie ou non en faisant clique gauche sur « Yes » ou « No » .



- Pour ce qui est du jeu, on a une partie qui commence tout d’abord avec un serpent de 10 segments, 50 obstacles (avec un accord commun nous avons décider de représenter ça sous former de rocher) et 5 pommes sur le terrain, lorsque le serpent mange une pomme le score s’incrémente de 5.



III) Justification du découpage :

Afin d'optimiser la clarté, la compréhension et la modularité de notre code, nous avons divisé notre jeu en cinq fichiers, ils sont structuré sous formes d’ un « .c » et d’un « .h » chacun , avec le fichier principal nommé "main.c". Cette structuration améliore la gestion du code source en facilitant la compréhension, la localisation des différentes fonctionnalités du jeu et la compilation et l’exécution.

Les différents fichiers sont :

affichage.c / affichage.h : Gère l'affichage graphique du jeu, y compris les écrans de menu, le terrain de jeu, l'écran de fin de partie, et la mise à jour du score. Fonctions principales :

- Screen_Menu(): Affiche l'écran de menu principal.
- Background(): Gère l'affichage du fond du jeu.
- Playground(): Dessine le terrain de jeu et le serpent
- GameOverScreen(): Affiche l'écran de fin de partie en cas de défaite
- Update_Score(): Met à jour l'affichage du score.
- Affiche les touches de contrôle.

apple.c / apple.h : Gère la génération aléatoire et la collision des pommes, ainsi que les interactions liées à la consommation des pommes. Fonctions principales :

- Apples_Random(): Génère aléatoirement des pommes sur le terrain
- Apple_Collision(): Vérifie s'il y a collision entre la tête du serpent et une pomme.
- Apple_Eating(): Gère la consommation de pommes par le serpent.
- Apples_Redraw(): Redessine les pommes sur le terrain.

game.c / game.h : Fournit des fonctions de gestion du temps et de synchronisation du jeu. Fonctions principales :

- Timer(): Met en pause le jeu pour un certain nombre de microsecondes. -
- MicrosecondesDepuisDebut(): Retourne le temps écoulé depuis le début du jeu en microsecondes.
- SecondesDepuisDebut(): Retourne le temps écoulé depuis le début du jeu en secondes.
- debut(): Initialise le temps de début du jeu.
- getsuivant(), setsuivant(), getdebutJeu(): Gèrent le suivi du temps.

main.c : Contient la fonction principale main() qui coordonne l'exécution du jeu en interagissant avec les autres fichiers. Fonctions principales :

- main(): Gère le déroulement du jeu dans son intégralité

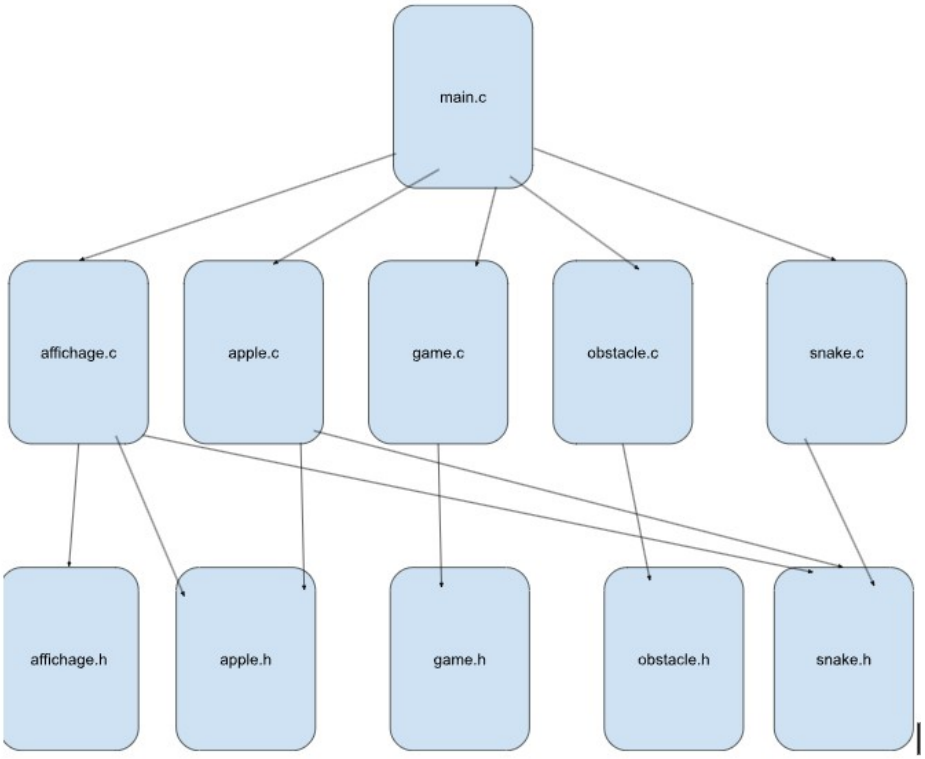
obstacle.c / obstacle.h : Gère la génération aléatoire des obstacles sur le terrain et la détection des collisions avec le serpent. Fonctions principales :

- Rock_Random(): Génère aléatoirement des obstacles sur le terrain.
- Snake_Obstacle_Collision(): Vérifie s'il y a collision entre le serpent et un obstacle.

snake.c / snake.h : Gère toutes les fonctionnalités liées au serpent, telles que son initialisation, son déplacement, et la gestion des collisions. Fonctions principales :

- Init_Snake(): Initialise les positions du serpent au début du jeu.
- EraseSnake(): Efface l'affichage du serpent sur le terrain.
- MoveSnake(): Déplace le serpent dans la direction spécifiée.
- Snake_Self_Collision(): Vérifie s'il y a une collision entre la tête et le corps du serpent.
- Snake_OutOfBounds(): Vérifie si la tête du serpent est en dehors des limites du terrain.
- Snake_Collision(): Vérifie s'il y a collision entre la tête du serpent et une partie de son corps.

Diagramme des dépendances de fichiers pour avoir les idées plus clair et concises :



IV) Explication des données du serpent

Dans notre implémentation, le serpent est matérialisé à travers une structure appelée SnakeCase, caractérisée par deux composants, x et y, qui déterminent les coordonnées d'un segment particulier du serpent sur l'écran de jeu.

Mouvement du serpent : L'action essentielle du serpent réside dans son déplacement. Chaque segment ajuste ses coordonnées(x, y) en fonction de la direction en cours, générant ainsi une impression de mouvement continu et fluide à travers l'écran.

Interaction avec les pommes : Lorsque le serpent atteint une pomme, une transformation est initiée. Les coordonnées de la pomme sont comparées à celles de la tête du serpent pour détecter une collision. En cas de succès, la pomme est assimilée, la longueur du serpent s'accroît, et le score du joueur s’incrémente de 5.

Collisions : Les collisions, qu'elles surviennent avec d'autres parties du serpent ou avec des obstacles, marquent la fin de la partie. Ces incidents entraînent une transformation de l'état du jeu, signalant la conclusion de la partie.

Réinitialisation du Jeu : A la fin d'une partie, une transformation de réinitialisation se déclenche. Les coordonnées du serpent, sa longueur, et d'autres paramètres du jeu retrouvent leurs valeurs initiales, offrant au joueur la possibilité de débuter une nouvelle partie.

En adoptant ces termes, la représentation du serpent demeure transparente, avec des transformations claires reflétant les actions du joueur et les événements du jeu, tout en évitant une paraphrase évidente des explications fournies précédemment.

V) Conclusion :

Fabio : Ce projet de création du jeu du snake en langage C a été une expérience particulièrement gratifiante pour moi. Bien que mes compétences en C ne soient pas encore très avancées, j'ai trouvé un réel plaisir dans la résolution de problèmes et la mise en œuvre des concepts appris.

Chaque obstacle rencontré a été l'occasion d'apprendre quelque chose de nouveau, et la satisfaction de voir le jeu fonctionner correctement a été une récompense en soi. Ce n'était pas simplement une tâche technique, mais une aventure où chaque ligne de code ajoutée représentait un petit pas vers la réalisation de l'ensemble du snake.

Bien que je ne considère pas encore la programmation comme une grande passion, cette expérience a définitivement élargi mes horizons et m'a donné une nouvelle perspective sur ce que je peux accomplir avec des compétences en développement en constante amélioration. Prendre plaisir à ce projet a été pour moi la meilleure motivation pour surmonter toutes les erreurs auxquelles nous avons fait face.

William : Ce projet de développement du jeu Snake a été une expérience enrichissante et passionnante pour moi. En tant qu'amateur de programmation, j'ai trouvé dans ce projet une opportunité d'appliquer mes connaissances et de relever des défis créatifs. L'exploration des différentes facettes du développement de jeux a été à la fois stimulante et formatrice.

Au fil de la conception de ce jeu, j'ai pu apprécier la complexité et la diversité des compétences nécessaires pour créer un programme interactif. Du choix des fonctionnalités à l'implémentation des mécanismes de jeu, chaque étape a été l'occasion d'apprendre de nouvelles techniques et de perfectionner mes compétences en programmation.