DEV SAE 1.1 JEU DU SERPENT

James Boutaric *Jude-Christ AISSI*

2023-2024

SOMMAIRE

Dans ce rapport, nous allons brièvement présenter le sujet, en exposant le but du jeu **Snake**. Ensuite, nous détaillerons les **fonctionnalités** du **programme**, en utilisant des captures d'écran pour fournir une compréhension approfondie de son fonctionnement.

Dans un second temps, nous justifierons le **découpage** du programme en différents fichiers source, étayant notre explication avec un diagramme explicatif. Cette **structuration modulaire** a pour objectif d'améliorer la compréhension du code du jeu.

Dans un troisième temps, nous explorerons les données représentant le **serpent** dans notre programme. Nous détaillerons les choix de représentation du serpent et les **transformations** qu'il peut subir tout au long du jeu.

Enfin, nous apporterons une **conclusion personnelle** sur ce projet.

I. INTRODUCTION

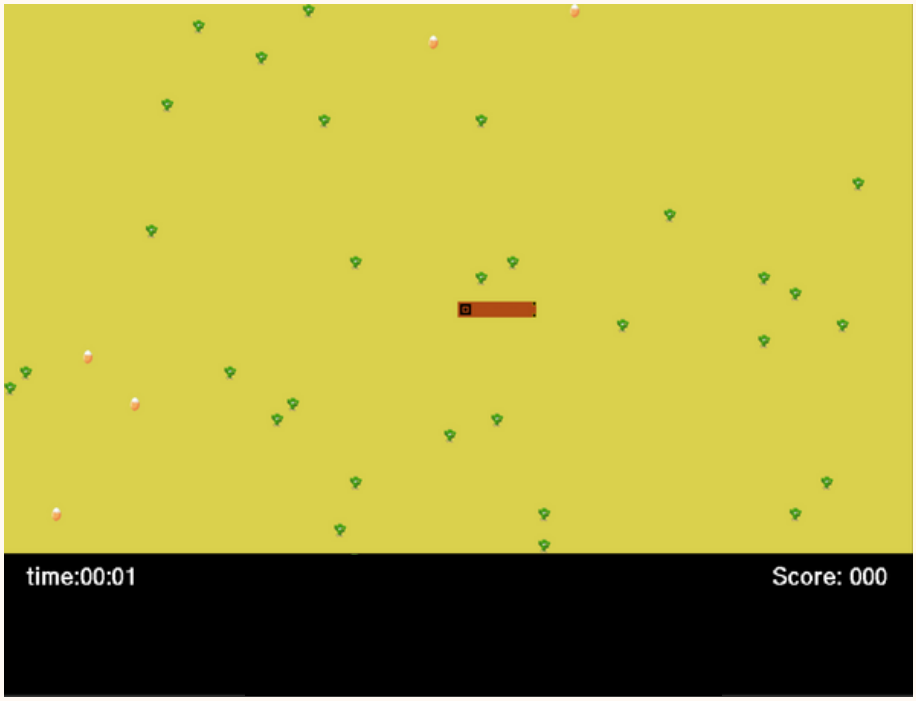
Dans cette **SAE**, nous avons programmer en **C89** un jeu snake. Le jeu consiste a **contrôler** un serpent qui se déplace dans un espace de jeu. L'objectif est de faire grandir le serpent en le faisant manger des **oeufs** tout en évitant les **obstacles**, les bords de l'écran, et le corps du serpent lui même. Le jeu se **termine** quand le serpent se heurte à un **obstacle**.

II. FONCTIONNALITÉS

Tout d'abord, dans notre menu principal, a deux options '**e**' et '**q**' qui sont mises à disposition. La touche '**e**', déclenche une **séquence** d'événements. Elle appelle la fonction **Afficherterrain()** qui vas créer une scène de jeu de dimensions de 60 **lignes** par 40 **colonnes** qui vas ensuite appeler **d'autres fonctions** qui vont créer les autres **éléments** du jeu.

DEV SAE 1.1: Jeu snake

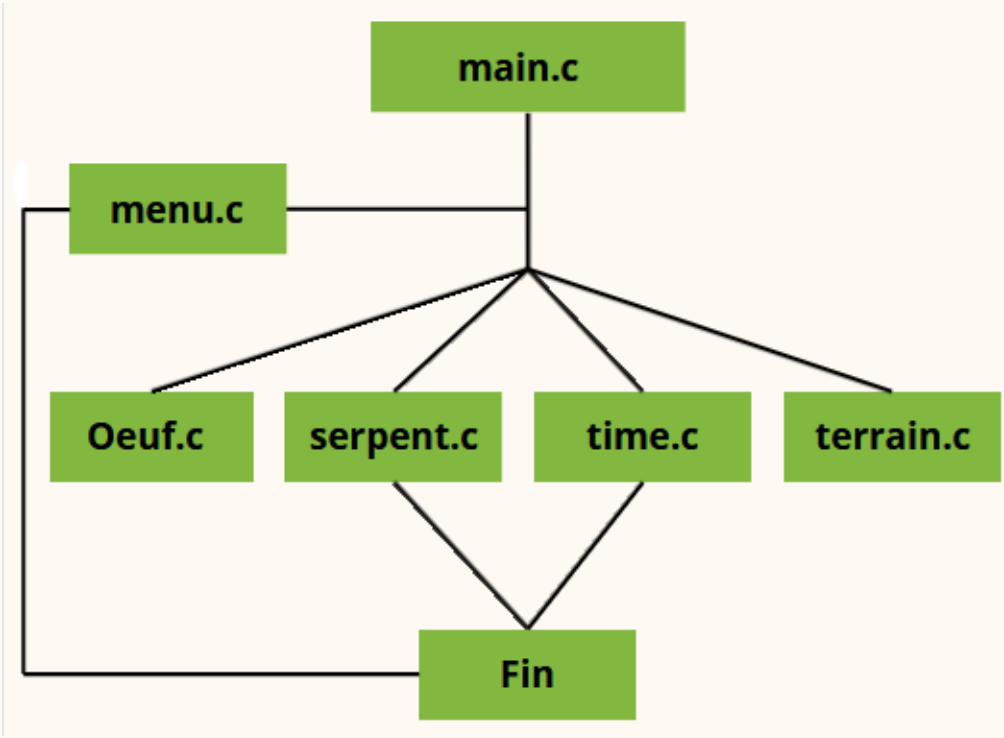
Après avoir appuyé sur la touche ‘**e**’ , nous accédons au jeu principal comme ci-dessous. Nous avons implémenté sur l’écran un **timer** (chronomètre) qui augmente au cours du jeu, et un **score** qui augmente progressivement dans la partie, ce qui vas permettre au joueur de connaitre ses **statistiques**.



Quant à la touche '**q**', permet a l'utilisateur de fermer le programme . Appuyer sur '**q**' ferme simplement le jeu.

Nous avons également implémenté une option '**pause**' qui permet de mettre le jeu en pause. Le joueur pourra donc **s'absenter** pendant ce temps.

III. STRUCTURE



Nous avons élaboré la structure suivante:

- main.c :
  - Fonction principale qui initialise le jeu.
  - Appelle la fonction Menu() du module menu.c pour gérer le menu principal.
  - Si le joueur choisit de jouer (en appuyant sur "e"), appelle la fonction lancer_jeu() pour commencer le jeu.

- menu.c :
  - Contient la fonction menu() qui gère le menu principal.
  - Si le joueur choisit de jouer, retourne à main.c pour lancer le jeu.
  - Si le joueur choisit de quitter, retourne à main.c pour fermer le programme.

DEV SAE 1.1: Jeu snake

- Œuf (oeuf.c) :
  - Contient les fonctions InitialiserOeufs() et Oeuf() pour gérer la génération et l'interaction avec les œufs.

- Serpent(serpent.c) :
  - Contient les fonctions Update_Serpent(), dessinerSerpent(), et Collision_Serpent() pour gérer le comportement du serpent, son affichage et la détection des collisions.

- Temps(time.c):
  - Contient la fonction timer() pour gérer le suivi du temps de jeu.
  - Contient la fonction score() qui gérer le calcul du score en fonction du nombre de segments du serpent.
  - Contient la fonction Update_Timer() qui est responsable de la mise à jour graphique de l'affichage du temps et du score. Elle utilise la bibliothèque graph.h pour effacer l'ancien affichage et écrire le nouveau temps et score à l'écran

- Terrain(terrain.c):
  - Contient la fonction Scene() pour générer le terrain de jeu, notamment les murs.

- Fin:
  - Point de sortie du programme, appelé depuis main.c lorsque le joueur décide de quitter ou lorsque le jeu se termine.

1.
   -

IV. ALGORITHME

L'algorithme du programme repose sur une structure modulaire. Lorsque la touche "**E**" est détectée, le programme appelle la fonction **lancer_jeu()** du fichier **main.c**, initiant ainsi le jeu. Cette fonction coordonne divers éléments du jeu, notamment le **serpent**, le **temps**, les **œufs**, et le **terrain**.

Le **serpent**, représenté graphiquement et géré par le fichier **serpent.c**, dispose d'algorithmes pour mettre à jour sa **position**, détecter les **collisions** avec les **murs** et les **œufs**, et interagir avec les commandes utilisateur.

Le temps de jeu est géré par des algorithmes de **minuterie** dans le fichier **time.c**. Une fois le jeu lancé, un minuteur s'active, affichant le **temps écoulé** à l'écran.

En ce qui concerne le **terrain**, il est associé a un algorithme qui génère **aléatoirement** la position des **obstacles** (représentés par des cactus) pour chaque nouvelle partie.

V. CONCLUSION

JAMES BOUTARIC

Cette **SAÉ** m'a permis de me familiariser avec le langage **C89** et le fonctionnement de **GIT**. De plus ce projet m'a permis de découvrir le **travail d'équipe** et **l'organisation** qu'il peut y avoir autour.

JUDE-CHRIST AISSI

Cette **SAÉ** m'as permis de bien comprendre que le **travail d'équipe** est important, de mieux comprendre les commandes **GIT** que je ne connaissais pas du tout, mais également de plus me familiariser avec le **langage C.**