# Rapport SAÉ DEV1.1 : Snake

Par Maxence Raymond et Matthieu Koulibaly Parkoo

## Introduction :

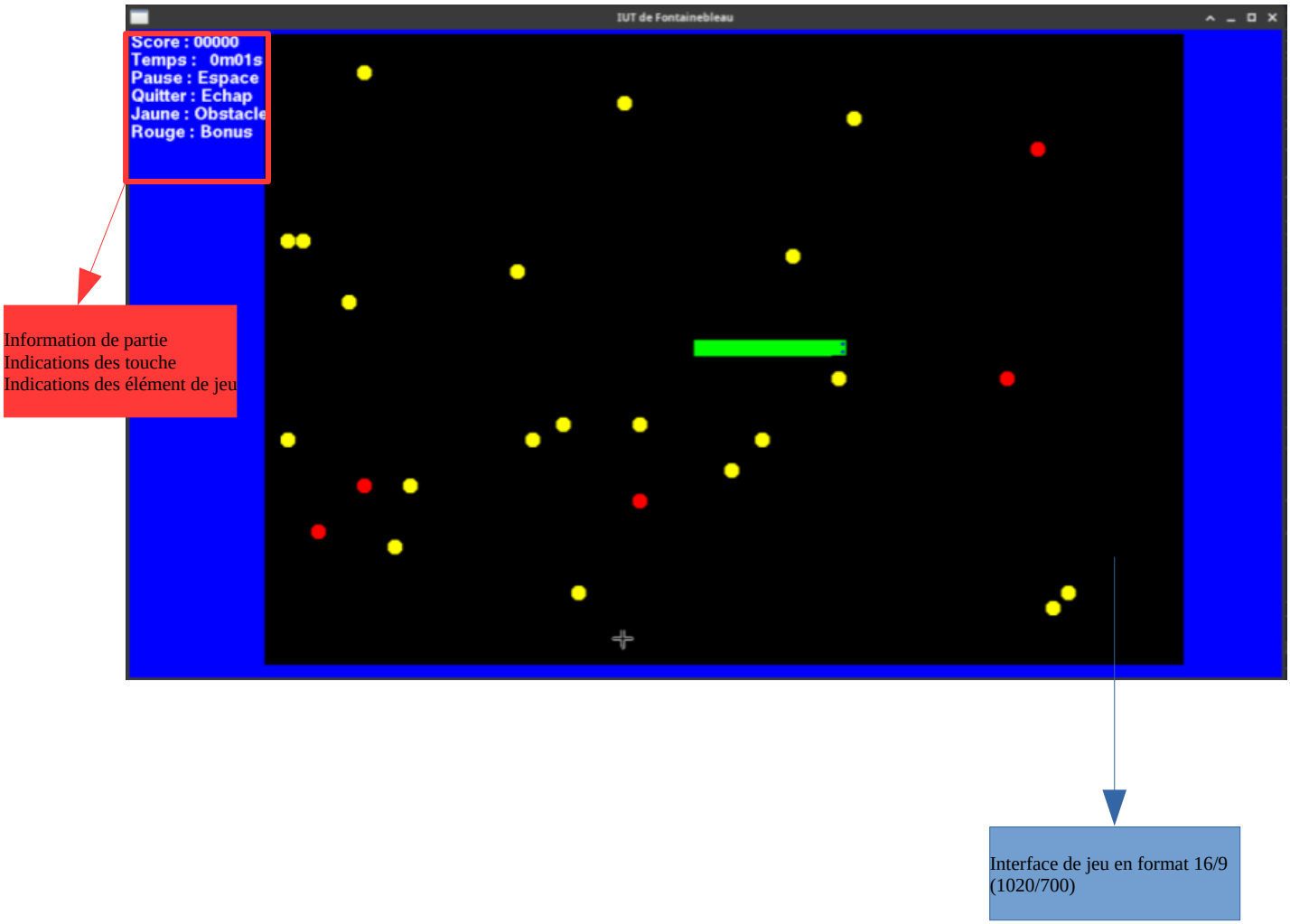
Le présent rapport a pour objectif de détailler la conception et la réalisation d'un jeu de Snake en utilisant le langage de programmation C. Le jeu de Snake est un classique de l'industrie du jeu vidéo qui met en avant la croissance d'un serpent en le faisant manger des aliments tout en évitant les obstacles. Nous allons explorer les différentes étapes pour créer ce jeu, notamment le développement du code, la manipulation des graphiques, la gestion des scores et la création d'une interface utilisateur intuitive et également l’ajout d’une nouvelle fonctionnalité : des obstacles.

# Sommaire

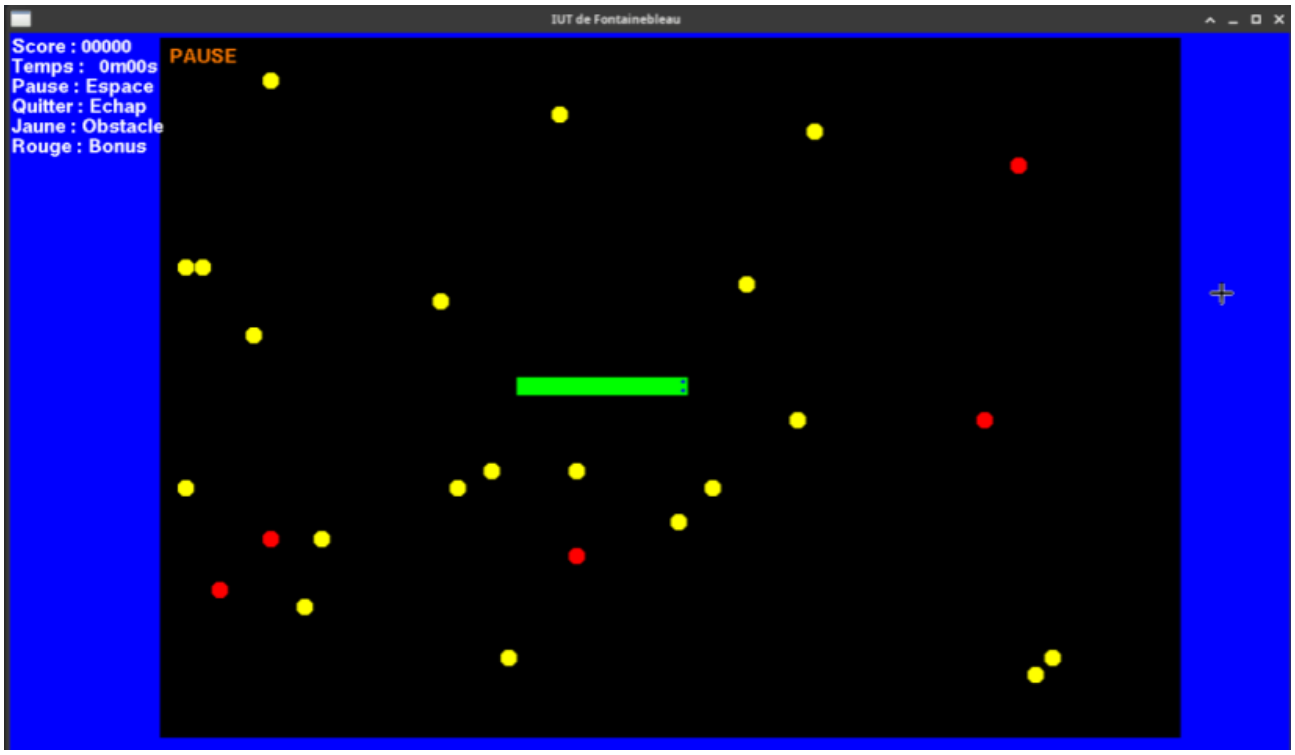
# 4. Conclusion

## 1. Présentation général de l'interface de jeu

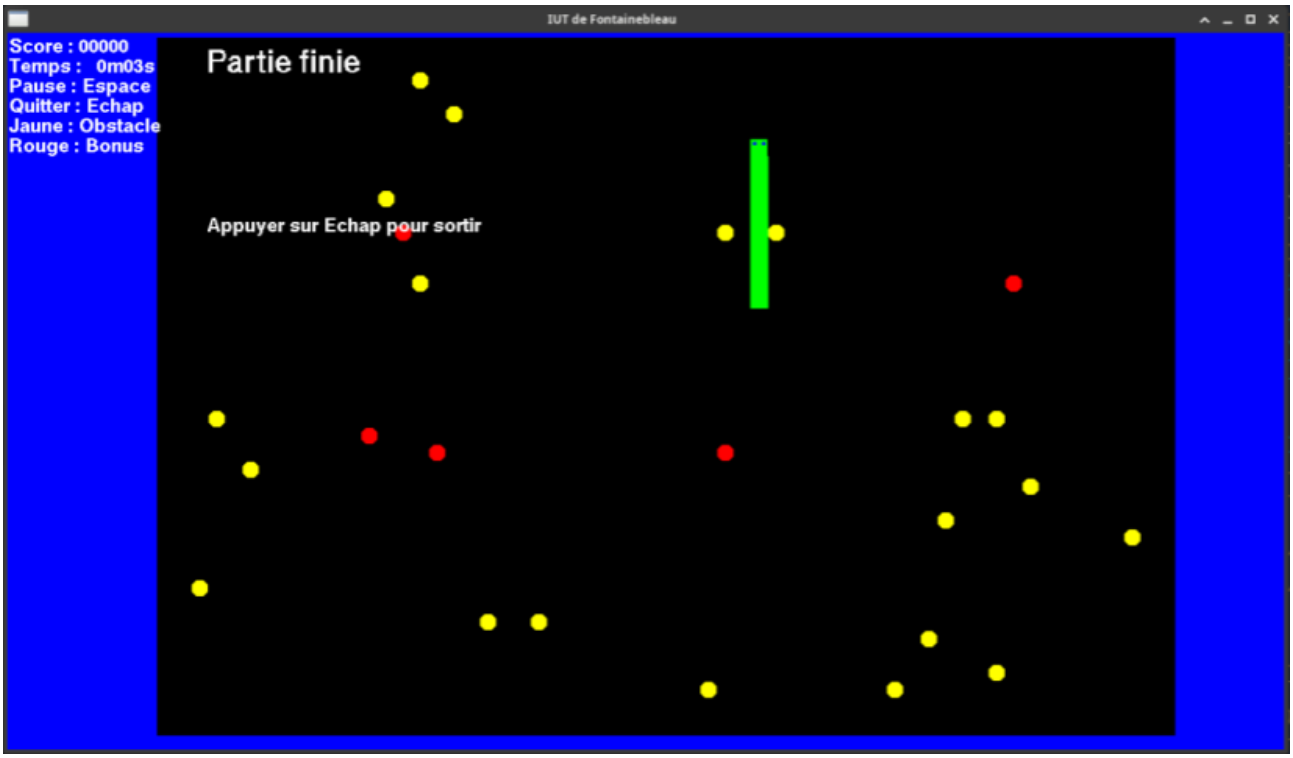
Écran de jeu durant une partie



Écran de pause :

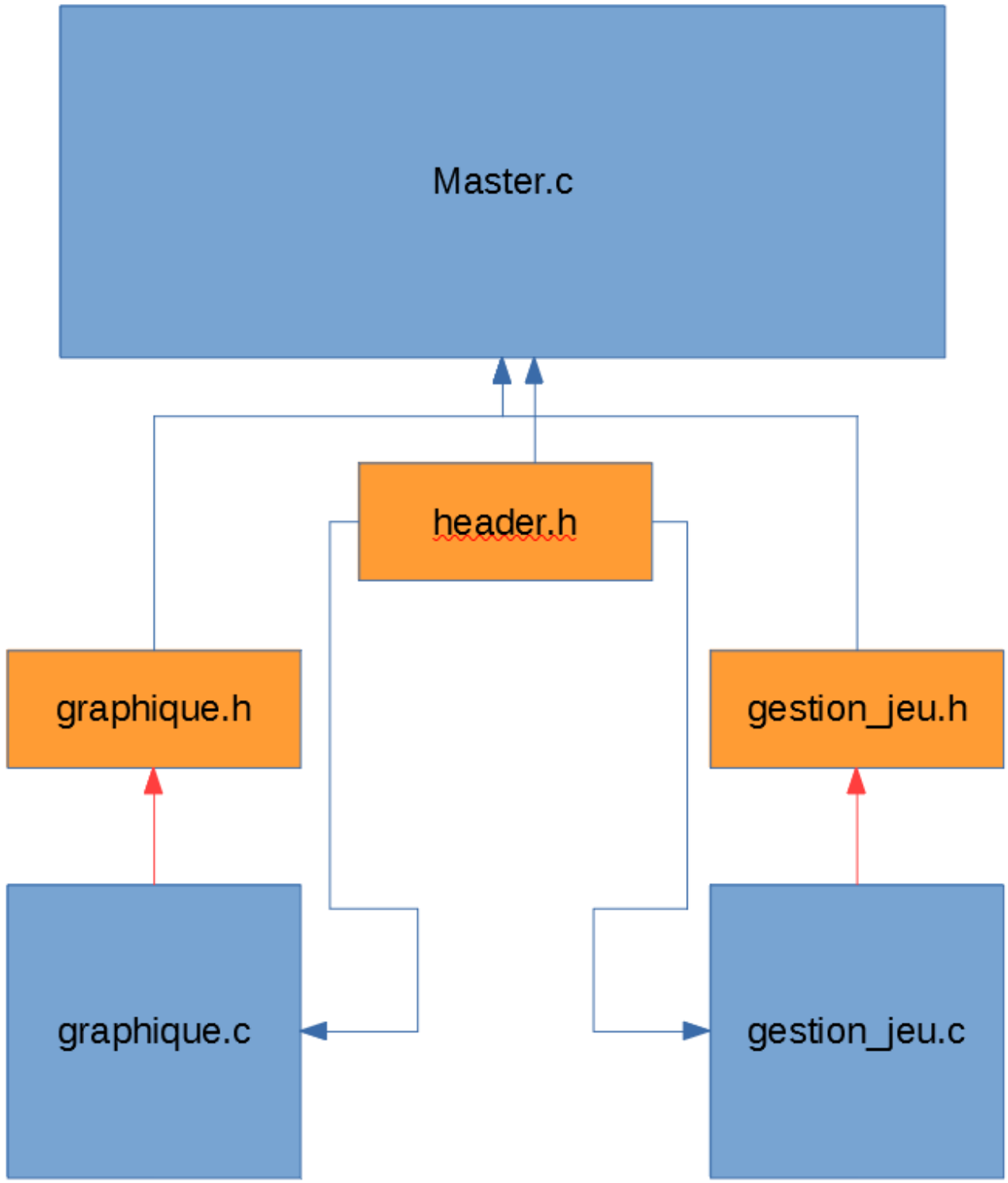



Écran de fin de partie :


IUT de Fontainebleau
Score : 00000
Temps : 0m03s
Pause : Espace
Quitter : Echap
Jaune : Obstacle
Rouge : Bonus
Partie finie
Appuyer sur Echap pour sortir

# 2. Organisation du code

Schéma d'interaction entre les différents fichiers :




*Flèche rouge : le fichier x.c à pour en-tête x.h
*Flèche bleue : le fichier x.c se réfère à x.h

## Description des fichiers

master.c : Ce fichier maître est le fichier principal pour la création du jeu de Snake. Il est essentiel pour coordonner et organiser l'ensemble du jeu. Il sert de point d'entrée du programme, contrôle le flux de jeu et gère les différentes fonctionnalités du jeu.

gestion_jeu.c : Le fichier gestion du jeu (nommé gestion_jeu.c) est celui qui répertorie les fonction qui seront utilisées pour le gameplay[1] et appelées depuis le fichier master.

graphique.c : Le fichier Graphique (nommé graphique.c) est celui qui répertorie les fonction qui seront utilisées pour la gestion de l'affichage des différents éléments sur l'interface de jeu.

header.h : Le fichier d'en-tête contient les constantes auxquelles se réfèreront les autres fichier. Ce sont les suivantes:

- Le cycle pour la gestion du temps
- Le nombre de pomme en début de partie
- Le nombre d'obstacle en début de partie
- Les longueur et largeur du tableau
- Les longueur et largeur des marges
- La taille en pixel d'une case de jeu

graphique.h : Fichier d'en-tête pour les fonctions de graphique.c

gestion_jeu.h : Fichier d'en-tête pour les fonctions de gestion_jeu.c

1 Déf: Façon dont le joueur interagit avec le jeu et le système de règle qui régissent ses actions

# 3. Revue des fichiers composant le jeu

## Master

Ce programme est un jeu de serpent écrit en langage C. Voici son fonctionnement en termes d'algorithme :

1. Le programme commence par initialiser les variables nécessaires :
   - sortie : une variable pour indiquer si le joueur souhaite quitter le jeu (0 : non, 1 : oui).
   - score : le score actuel du joueur.
   - temps : le temps écoulé depuis le début du jeu.
   - modifvitesse : le temps écoulé depuis la dernière modification de vitesse.
   - decalage : le décalage temporel du jeu.
   - allongementserpent : le nombre de segments à ajouter au serpent.
   - x et y : les coordonnées de la tête du serpent.
   - fin_x et fin_y : les coordonnées de la fin du serpent.
   - direction : la direction actuelle du serpent.
   - vieux_direction : la direction précédente du serpent.
   - tableaudejeu : un tableau à deux dimensions représentant le plateau de jeu.

2. Le programme initialise le décalage temporel en utilisant la fonction "Microsecondes()", qui renvoie une valeur de temps équivalente à POSIX.

3. Le programme appelle la fonction "Initialisation()" pour initialiser le tableau de jeu, la position du serpent et sa direction, ainsi que les pommes et les obstacles.

4. Le programme appelle la fonction "DessinerScene()" pour afficher la scène de jeu (le plateau de jeu et les informations du joueur).

5. Le programme appelle la fonction "Pause()" pour attendre avant de commencer le jeu (empêche le jeu de démarrer immédiatement).

6. Le programme entre dans une boucle principale, qui s'exécute tant que la variable "sortie" est égale à 0 (le joueur ne souhaite pas quitter le jeu).

7. À l'intérieur de la boucle principale, le programme vérifie si le temps écoulé depuis le dernier cycle est supérieur à un certain intervalle de temps (CYCLE). Si c'est le cas, le programme met à

jour le temps écoulé.

8. Le programme vérifie si une touche est en attente en appelant la fonction "ToucheEnAttente()". Si une touche est en attente, le programme lit la touche pressée à l'aide de la fonction "Touche()".

9. Le programme effectue différentes actions en fonction de la touche pressée :
   - Si la touche est "Escape", le programme met la variable "sortie" à 1 pour indiquer que le joueur souhaite quitter le jeu.
   - Si la touche est "espace", le programme appelle la fonction "Pause()" pour mettre le jeu en pause.
   - Si la touche est "Right", le programme change la direction du serpent en "droite".
   - Si la touche est "Left", le programme change la direction du serpent en "gauche".
   - Si la touche est "Up", le programme change la direction du serpent en "haut".
   - Si la touche est "Down", le programme change la direction du serpent en "bas".
De plus, elle empêche le demi tour du serpent, et donc de perdre la partie sans raison

10. Le programme vérifie si le temps écoulé est supérieur au temps de la dernière modification de vitesse plus un dixième de l'intervalle de temps entre deux cycles (CYCLE/10). Si c'est le cas, le programme met à jour le temps de la dernière modification de vitesse.

11. Si la vérification précédente s’évalue à vrai, le programme met à jour le tableau de jeu en fonction de la direction du serpent et affiche le serpent en appelant les fonctions "AffichageSnake()" et "AffichageTeteSerpent()".

12. Le programme vérifie si le serpent atteint une condition de fin de jeu en appelant la fonction "Verif()". Si c'est le cas, le programme met à jour la variable "sortie" pour indiquer que le joueur a échoué et met à jour le score et le nombre de segments à ajouter au serpent. Elle vérifie aussi que la «rencontre» avec une pomme ne fait pas quitter le jeu ainsi que les conséquences attendues.

13. Le programme met à jour la fin du serpent si le nombre de segments à ajouter est égal à zéro en appelant la fonction "DisparitionSnake()". Dans le cas contraire, la valeur diminue et la taille du serpent augmente donc de 1.

14. Après la fin de la boucle principale, le programme vérifie si le joueur a échoué ou terminé le jeu en appelant les fonctions "AffichageEchec()" ou "AffichageFinPartie()".

15. Le programme ferme l'interface graphique du jeu en appelant la fonction "FermerGraphique()".

16. Le programme se termine en renvoyant "EXIT_SUCCESS".

# Gestion du jeu

Le fichier gestion du jeu (nommé gestion_jeu.c) est celui qui répertorie les fonction qui seront utiliser pour le gameplay dans le fichier master.

- Initialisation :

*Arguments

- un tableau de caractère (servant de tableau de jeu)
- un caractère (pour la direction)
- 2 entiers (en tant que coordonnée du serpent)

*description : initialise l'interface de jeu (de 1280 sur 720 pixel) de couleur noir et des bordures bleues, place (et affiche) les différents élément du jeu en fonction des valeurs de header.h (5 pommes, 20 obstacles)

- Ajout_Pomme :

*Arguments

- un tableau de caractère (servant de tableau de jeu)

*description : Ajoute une pomme à un emplacement de tableau dimensionnel donnée, remplaçant la valeur 0 par le caractère 'p' dans le tableau et l'affiche à un emplacement aléatoire.

- Ajout_Obstacle :

*Arguments

- un tableau de caractère (servant de tableau de jeu)

*description : Ajoute une pomme à un emplacement de tableau dimensionnel donnée, remplaçant la valeur 0 par le caractère 'o' dans le tableau et l'affiche à un emplacement aléatoire.

- Verif :

*Arguments

- un tableau de caractère (servant de tableau de jeu)
- 2 entiers (coordonnées du serpent)
- un caractère (direction du mouvement)
- pointeurs vers des variables de sortie (sortie, score, allongementserpent).

*description : Vérifie les conséquences de l'interaction du serpent avec le terrain de jeu. Elle commence par récupérer la position du serpent dans le tableau de jeu. Si le serpent se trouve sur une case contenant une pomme, le score est augmenté de 5, l'allongement du serpent est augmenté de 2, et une nouvelle pomme est ajoutée au tableau de jeu.
Si ce n'est pas le cas, la fonction vérifie ensuite si le serpent a atteint les bords du tableau de jeu. Si c'est le cas, la

variable de sortie est mise à 2 pour indiquer l'échec de la partie.
Enfin, la fonction vérifie si le serpent se mange lui-même ou rencontre un obstacle. Si c'est le cas, la variable de sortie est également mise à 2 pour indiquer l'échec de la partie.

# Graphique

Le fichier Graphique (nommé graphique.c) est celui qui répertorie les fonction qui seront utiliser pour la gestion de l'affichage des différents élément sur interface de jeu.

- AffichageTemps :

*Arguments

- un entier non signé long (pour le temps)

*description : Prend en argument un entier représentant le temps en microsecondes et le transforme en texte pour ensuite l'afficher à l'écran.

- AffichageScore :

*Arguments

- un entier (pour le score)

*description : Prend en argument un entier représentant le score et le transforme en texte pour ensuite l'afficher à l'écran.

- AffichageInfos :

*Arguments

- aucun

*description : Cette fonction affiche des informations à l'écran, telles que les touches pour mettre en pause ou quitter le jeu, la couleur des obstacles et des bonus via des tableaux de caractère initialisés au préalable.

- DessinerScene :

*Arguments

- un entier non signé long (pour le temps)
- un entier (pour le score)

*description : Est la fonction principale de dessin. Elle gère les écrans virtuels, les appels aux fonctions pour l'affichage du temps, du score et les informations, et l'affichage global de toute la scène.

- Pause

*Arguments :

- un entier non signé long (pour le temps avant la pause)
- un pointeur vers un entier non signé long (mis à jour avec le temps passé lors de la pause)

*Description : Cette fonction affiche le menu de pause et attend un appui sur la touche espace pour continuer le jeu. Elle met à jour le temps passé lors de la pause dans la variable pointée par decalage.

- AffichageEchec

*Arguments :
  - Aucun

*Description : Cette fonction bloque l'écran et affiche un message d'échec, demandant à l'utilisateur d'appuyer sur la touche espace pour quitter le programme.

- AffichageFinPartie

*Arguments :
  - Aucun

*Description : Cette fonction bloque l'écran et affiche un message de fin de partie, demandant à l'utilisateur d'appuyer sur la touche Echap pour quitter le programme.

- AffichagePomme

*Arguments :
  - 2 entier (coordonnée de la pomme)

*Description : Cette fonction affiche la partie graphique d'une pomme (représenté par un cercle rouge) à la position (x, y) sur l'écran.

- AffichageObstacle

*Arguments :
  - 2 entier (coordonnée de l’obstacle)

*Description : Cette fonction affiche la partie graphique d'un obstacle (représenté par un cercle jaune) à la position (x, y) sur l'écran.

- AffichageTeteSerpent

*Arguments :
  - 2 entier (coordonnée de la tête du serpent)
  - un caractère (représente la direction du serpent)

*Description : Cette fonction affiche la partie graphique de la tête du serpent à la position (x, y) sur l'écran, en fonction de la direction du serpent ('h' pour haut, 'b' pour bas, 'd' pour droite, 'g' pour gauche).

- AffichageSnake

*Arguments :
  - 2 entiers (coordonnées du serpent)

*Description : Cette fonction affiche la partie graphique d'une partie du corps du serpent à la position (x, y) sur l'écran.

- DisparitionSnake

*Arguments :
- 2 entiers (coordonnées du serpent)

*Description : Cette fonction efface la partie graphique d'une partie du corps du serpent à la position (x, y) sur l'écran.

# 4. Conclusion

Ce projet nous aura permis d'apprendre de nouvelles choses tant sur l'utilisation de la bibliothèque graphique, que sur les conventions de nommage et de formatage du code qu'autant sur la gestion de projet. Cela nous aura permis de comprendre l'importance d'un planning à tenir et de s'organiser en conséquence afin que le résultat soit présent à la date précisée. Personnellement, je trouve que ce projet est loin d'être parfait, mais que cependant, le résultat délivré est à la hauteur des attentes. Certaines choses auraient pu être faites différemment, tel que par exemple utiliser des structures, mais j'ai estimé cela plus important de rajouter des fonctionnalités telles que l'affichage de la tête du Snake. Pour finir, certains bugs ayant été essayé d'être résolus pendant plusieurs heures auront finalement pu être résolus avec une nouvelle approche, ou avec un point de vue différent. C'est en cela que bien que la plupart des commits soient à mon nom, l'aide de mon coéquipier lors de la rédaction du code aura été vraiment apprécié comme un bonus.
Maxence Raymond

En conclusion, la réalisation de ce jeu de Snake en langage C a été une expérience enrichissante malgré les difficultés rencontrées. Passionné par la programmation, j'ai dû relever plusieurs défis pour aboutir à un résultat fonctionnel. Tout au long de ce projet, j'ai appris énormément sur la structure du langage C, la gestion des tableaux et la logique de programmation. Malgré les nombreux débogages nécessaires, j'ai réussi à créer un jeu de Snake qui répond à mes attentes. Je suis fier d'avoir pu relever ce défi, et cela renforce ma motivation à poursuivre mon apprentissage dans le domaine de la programmation.
Matthieu Koulibaly Parkoo