SOMMAIRE

INTRODUCTION

Nous vous présentons notre jeu Snake, fruit de notre collaboration en équipe de deux dans le cadre de nos études. L'objectif de ce rapport est de détailler le processus de création et les fonctionnalités du jeu. Nous vous guiderons à travers le processus de conception, les fonctionnalités de notre jeu et des différentes parties du code. Notre démarche s'articule autour d'un Snake classique avec deux ajouts : la vitesse croissante en fonction du score ou plus facilement dit en fonction du nombre de pommes mangés ainsi que la présence d'obstacles. Nous conclurons par nos impressions personnelles sur ce sujet.

Nous vous invitons à explorer notre création à travers ce compte rendu.

JUSTIFICATION & DÉCOUPAGES

Pour améliorer la lisibilité, la maintenance et la modularité de notre code, nous avons organisé notre jeu en quatre fichiers distincts, tous portant l'extension "*.c", avec le fichier principal "main.c":

evenements.c : Ce fichier se charge de gérer les événements, notamment la gestion des touches et des interactions utilisateur.

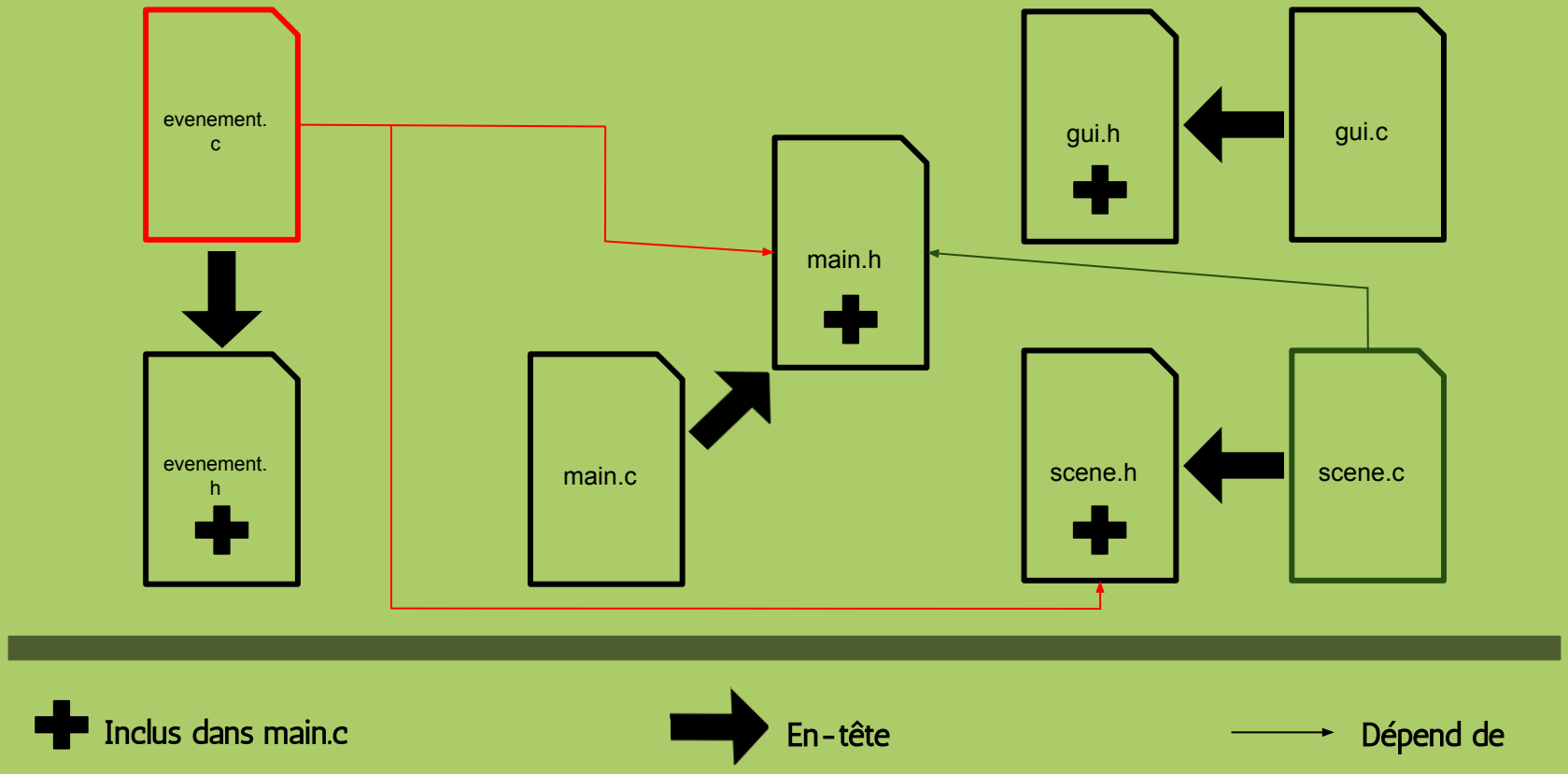
gui.c : Responsable de l'interface graphique, ce fichier gère à la fois le front end et le back end du jeu. Il inclut les fonctions nécessaires pour afficher les menus, les écrans de fin de partie et autres éléments graphiques.

scene.c : Ce fichier est dédié à l'initialisation du jeu. Il contient les fonctions nécessaires pour préparer l'environnement de jeu, y compris le placement initial des éléments tels que le serpent, les pastilles et les obstacles.

main.c : Le fichier principal qui agit comme point de liaison entre les autres fichiers. Il orchestre l'interaction entre eux, permettant ainsi de lancer et de coordonner l'exécution du jeu.

JUSTIFICATION & DÉCOUPAGES

* Ne comporte pas les en-têtes de bibliothèques standards et graph.h



Explications et transformations

Dans notre jeu, le serpent est représenté par une structure de données nommée PIXELS. Cette structure contient deux membres, x et y qui représentent les coordonnées d'un segment du serpent dans l'espace du jeu. La position (x, y) d'un segment déterminent son emplacement sur l'écran.

Le serpent

Déplacement : La transformation la plus fondamentale est le déplacement du serpent. Chaque segment du serpent est déplacé vers une nouvelle position en fonction de sa direction. Les coordonnées (x, y) sont mises à jour en conséquence, créant l'illusion du mouvement continu du serpent à travers l'écran (en réalité il n'y a que la tête et la queue qui se déplace ou disparaît).

Changement de Direction : Lorsque le joueur interagit en changeant la direction du serpent à l'aide du clavier cela déclenche une transformation instantanée des données de direction du serpent.

Explications et transformations

Consommation de Pastilles : Lorsque le serpent atteint une pastille (ou pomme), une transformation survient. Les coordonnées de la pastille sont comparées à celles de la tête du serpent pour détecter une collision. En cas de collision, la pastille est consommée, la longueur du serpent augmente, et le score du joueur est ajusté (+5pts).

La pomme

Collision avec Soi-même ou les Obstacles : Si le serpent entre en collision avec lui-même ou avec des obstacles dans l'environnement du jeu ou les bordures du terrain lui même cela conduit à la fin de la partie.

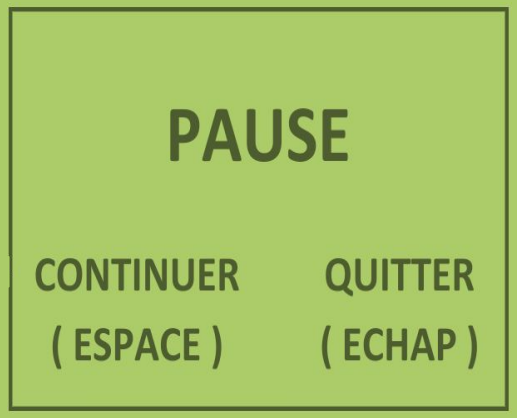
L'obstacle

Réinitialisation du Jeu : À la fin d'une partie, une transformation de réinitialisation peut avoir lieu. Les coordonnées du serpent, sa longueur et d'autres paramètres du jeu sont être réinitialisés pour permettre au joueur de recommencer.

Fonctionnalitées & Descriptions

L'affichage !

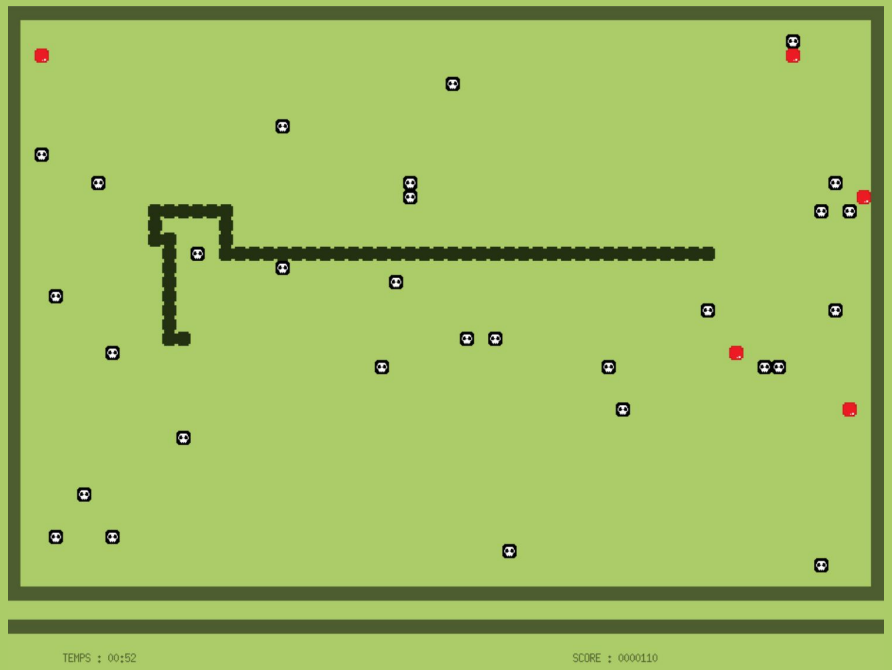
Le jeu est composé de trois menus, chacun équipé de deux boutons qui permettent soit de poursuivre la partie actuelle, soit d'en entamer une nouvelle. Soit de quitter le jeu et ainsi terminer le programme.



Notons que les boutons "Quitter" (non cliquables mais utilisables avec la touche Échap) permettent de terminer le programme peu importe où l'on se trouve dans le jeu, et seule la mort du serpent permet de sauvegarder le score.

Fonctionnalitées & Descriptions

Le jeu !



Une partie commencera toujours avec un serpent de 10 segments, 30 obstacles et 5 pommes sur son terrain. Le temps est initialisé à 00:00 et le score à 000000. Chaque pastille/pomme lui fait gagner 5 points. (Ici, le serpent a mangé 22 pommes en 52 secondes !)

Fonctionnalitées & Descriptions GUI.c

- La fonction Menu appelle la fonction lireScore puis la met dans une variable bestscore et affiche le meilleure score de toute les partie faite en plus d'une image (lancer une partie et quittez le jeu)
- La fonction Pause affiche une image (reprendre et quittez la partie)
- La fonction PerduGUI appelle la fonction lireScore puis la met dans une variable bestscore et affiche le meilleure score de toute les partie faite en plus d'une image (lancer une partie et quittez le jeu).
- La fonction lireScore va ouvrir un fichier (puis le fermer a la fin) et récupérer la valeur écrit dedans et va la renvoyer.

Fonctionnalitées & Descriptions GUI.c

- La fonction sauvegarderScore prend en argument le nouveaux meilleure score et va ouvrir un fichier en mode write fichier (puis le fermer a la fin) et va écrire dans le fichier le nouveau score.
- La fonction CheckScore prend en argument le nouveau score , elle va appeler la fonction lireScore pour stocker dans une variable le meilleure score puis va comparer le nouveau score avec le meilleure si cela renvoie 0 alors on appelle la fonction sauvegarderScore avec le nouveau score.
- La fonction AfficherTimerEtScore prend en argument le score , les minutes et les secondes et permet d'afficher le temps et le score de la partie.

Fonctionnalitées & Descriptions

EVENEMENT.c

- La fonction MourrirSerpent prend en argument les pixel du serpent , ceux des obstacles , la taille du serpent et le nombre d'obstacle et que si la tête du serpent touche un obstacle ou son propre corps ou sors de la limite du jeu cela renvoie 1 et au contraire 0.
- La fonction MangerPastille prend en argument les pixels de serpent, ceux des pastilles/pommes, ceux des obstacles ,le score de la parti, la taille du serpent ,le nombre obstacles et la vitesse de déplacement du serpent et qui va générer une pastilles quand la tête du serpent va toucher les coordonnée d'une pomme augmenter le score de 5 , diviser la vitesse de 1.008 et la fonction va renvoyer 1 sinon 0.

Fonctionnalitées & Descriptions

EVENEMENT.c

- La fonction DeplacementSerpent prend en argument la direction actuel du serpent, les pixels du serpent et la taille du serpent et va permettre au serpent de se déplacer tout en déplaçant tous ces pixels.
- La fonction Serpent prend en argument le pixels de serpent, ceux des pommes, ceux des obstacles, le score de la partie, la taille du serpent, le nombre d'obstacle, la vitesse de déplacement du serpent et la direction actuel du serpent et va renvoyer 2 si MourirSerpent renvoie 1, si MangerPastille renvoie 1 elle renverra augmentera la taille du serpent et renverra 1 sinon 0.

Fonctionnalitées & Descriptions SCENE.c

- Le struct PIXELS prend deux argument qui sont les coordonnée en x et y.
- La fonction ArrondirPixel prend en argument un nombre et qui va arrondir les coordonnée du pixels et renvoie la valeur de l'arrondie trouver.
- La fonction PIXELS gen_pastille prend en argument les pixels de serpent, ceux des pommes/pastilles, ceux des obstacles, la taille du serpent et le nombre d'obstacle. Elle va empêcher la génération des pastilles sur le serpent et sur les pastilles ainsi que sur les obstacles. Elle va décaler les pixels pour qu'il apparaisse au bon endroit sur le jeu et qui renvoie un pastille (qui est un struct).

Fonctionnalitées & Descriptions SCENE.c

- La fonction PIXELS gen_obstacle prend en argument les pixels de serpent, ceux des pommes/pastilles, ceux des obstacles, la taille du serpent et le nombre de pastilles et qui va prendre en compte que les obstacles de peuvent pas être générer sur le serpent et sur les obstacle et sur les pastilles et va décaler les pixels pour qu'il apparaisse au bon endroit sur le jeu et qui renvoie un obstacle(qui est un struct).
- La fonction InitialisationDuSerpent prend en argument les pixels du serpent et permet d'afficher/générer les 10 premiers pixels du serpent.
- La fonction InitialisationPastilles prend en argument les pixels du serpent, ceux des pommes, ceux des obstacles, la taille du serpent et le nombre d'obstacle et permet d'afficher/générer les 5 premiers pixels des pommes.

Fonctionnalitées & Descriptions SCENE.c

- La fonction InitialisationObstacle prend en argument les pixels du serpent, ceux des pommes, ceux des obstacles, la taille du serpent et le nombre pastilles et permet d'afficher/générer les 30 pixels des obstacles.
- La fonction DessinerScene prend en argument les pixels du serpent, ceux de pommes, ceux des obstacles, la taille du serpent, le nombre de pastilles et le nombre d'obstacles et affiche le fond puis appelle les fonction InitialisationDuSerpent ,InitialiserPastilles et InitialiserObstacle.

CONCLUSION

RAPPORT

Pendant la première semaine, nous avons commencé notre jeu en intégrant une interface graphique basique, ajouté la génération des pastilles et introduit la boucle d'événements pour les interactions.

La deuxième semaine a été dédiée à l'initialisation du serpent, avec une transition vers l'utilisation d'un tableau dynamique pour stocker ses valeurs. De plus, nous avons intégré un timer pour suivre la durée de jeu.

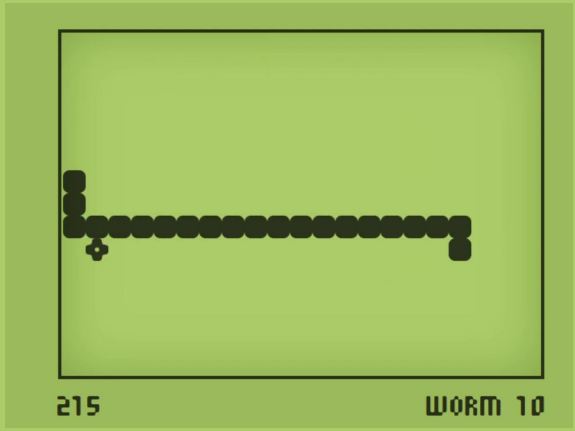
La troisième semaine a été la plus riche en fonctionnalités. Nous avons introduit la possibilité de mettre le jeu en pause, amélioré le timer, et empêché le serpent de se déplacer sur lui-même en martelant les touches. Le code a été réorganisé en plusieurs fichiers pour une meilleure lisibilité. Nous avons également créé une structure "pixel" pour clarifier le code et ajouter la détection des collisions avec les pastilles, ainsi que la fonction d'agrandissement du serpent. Enfin, le rapport.txt a été créé pour documenter le projet.

Au cours de la quatrième et cinquième semaine, nous avons consolidé l'ensemble du code et finalisé le projet, assurant sa complétion dans les délais impartis malgré les contraintes liées aux examens et aux projets parallèles.

CONCLUSION

GRAPHISMES

Nous avons été inspiré par la vidéo de Leonardo's Youtube Channel sur youtube intitulée : " Retro Snake Game ".



Notre palette graphique est la suivante :

- Le vert clair (fond): #abcc68 , vert foncé (bordure): #4e5d2f,
- le serpent : #233012, une nuance de la couleur rouge (pommes/pastilles): #ed1c24
- Ainsi que le noir et blanc classique : #000000 et #ffffff.

CONCLUSION

PERSONNELLES

DAVID : J'ai trouver le projet compliqué selon moi mais je prenais un plaisir quand le code que j'écrivais fonctionnait même si des fois je bloque sur des erreurs bêtes mais c'est en résolvant des petites erreurs que l'on apprend et progresse le plus, néanmoins je suis fier de ce qu'on a réussi à faire avec mon collègue de travail de Vincent mais souvent nous pouvions pas récupérer le code depuis le git car cela nous faisait une erreur lors du git pull.

VINCENT : Je suis particulièrement fier de notre accomplissement, car c'était ma première expérience à coder entièrement un jeu sans recourir à des sources extérieures ni à des parties de code déjà construites. Malheureusement, de nombreuses heures ont été consacrées à résoudre des problèmes qui auraient pu être évités en lisant plus attentivement l'énoncé. Bien que je ressens une légère déception de ne pas avoir résolu un dernier problème, je suis néanmoins fier de mon travail.

Travailler à deux sur ce projet a été une expérience enrichissante. Nous avons réussi à le mener à bien malgré les examens et les projets à rendre, ainsi que notre temps de trajet pour arriver à l'IUT. Trouver une organisation pour les différentes parties du code était à la fois amusant et stressant.