Carriço Matthieu
Minguet Paul

Groupe 3

Projet Tutoré APL 2019-2020

- Snake -

Sommaire

Introduction :

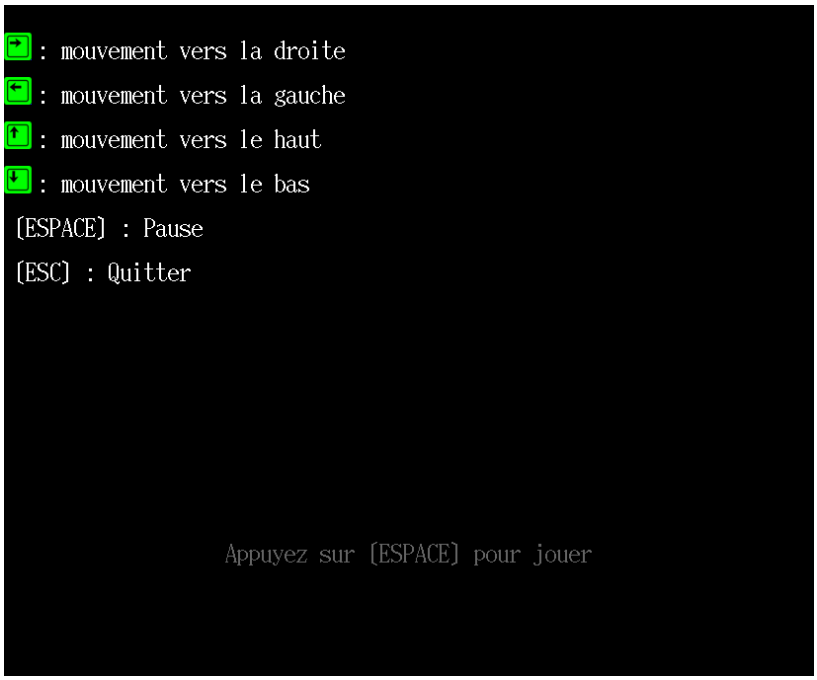
Pour ce projet, nous avons dû créer un jeu de Snake, un jeu apparût en 1976 sur le jeu d'arcade *Blockade*, puis s'est fait connaître notamment grâce au Nokia 3310 et a été repris de nombreuses fois depuis. Le principe de jeu est le suivant : un serpent apparait sur une carte ainsi que cinq pommes. Le but du serpent (du joueur) est de gagner le plus de points (obtenus en mangeant les pommes) sans mourir. Le serpent s'agrandi à chaque pomme mangée et meurt si : il rentre dans une bordure de la carte ou bien s'il se rentre dedans (si la tête touche le corps). Le serpent peut donc se déplacer vers le haut, vers le bas, vers la gauche et vers la droite ; Néanmoins, il ne peut pas se retourner, c'est-à-dire passer de gauche à droite ou de haut en bas en inversement.

Notre programme :

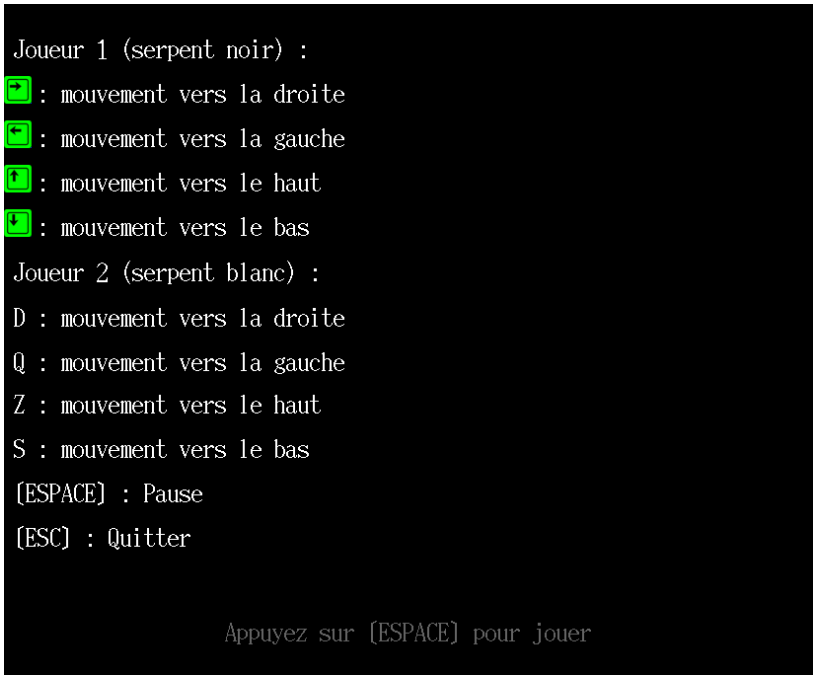
Tout d'abord, nous avons créé un menu sur lequel nous pouvons choisir, à l'aide de la souris, un mode de jeu solo ou multijoueur ; Nous pouvons aussi choisir de quitter le jeu à l'aide du bouton Quitter (cf. capture d'écran n° 1).

Capture d'écran numéro 1 : menu

Après avoir cliqué sur le bouton Solo ou Multijoueur, le joueur est renvoyé sur un écran de chargement qui lui indique les différentes touches pour jouer (les écrans diffèrent en fonction du mode de jeu). Le joueur un aura la possibilité de se déplacer avec les touches directionnelles du clavier et le joueur deux sera doté des touches traditionnelles « ZQSD ». Ces derniers auront la possibilité de mettre le jeu en pause à tout moment en appuyant sur la touche « espace » ou bien de quitter celui-ci en se servant de la touche « échap » (cf. captures d'écran n° 2 et 3).

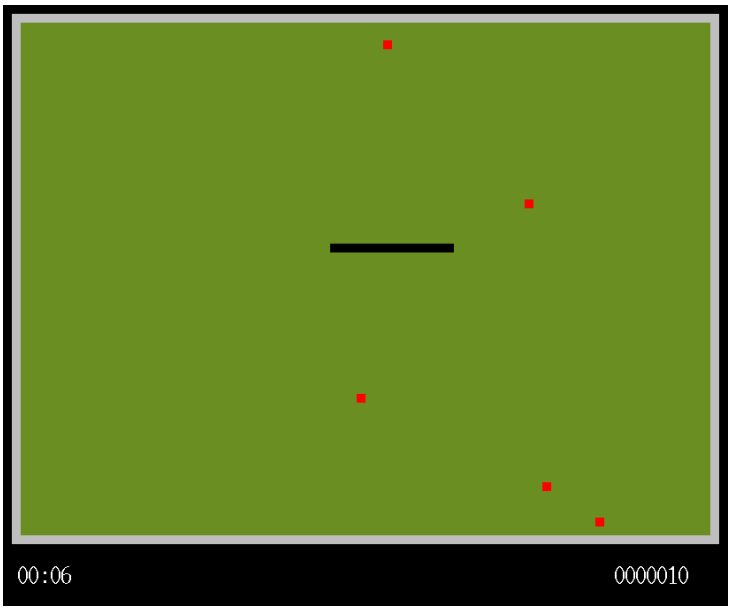


Capture d'écran numéro 2 : écran de chargement solo

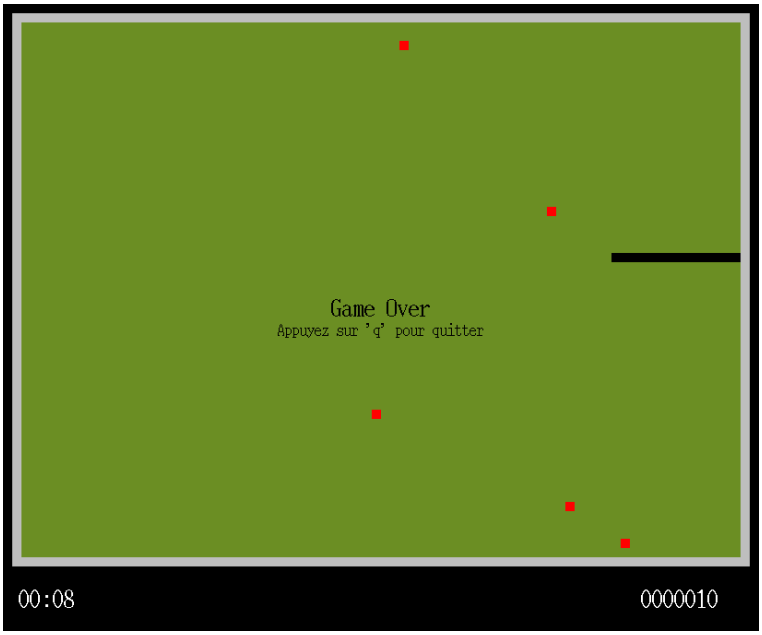


Capture d'écran numéro 3 : écran de chargement multijoueur

Une fois arrivé dans le jeu, après avoir cliqué sur « Solo » ou « Multijoueur », le joueur voit le serpent (noir pour le joueur un et blanc pour le joueur deux si le mode est multijoueur). Nous allons parler du mode un joueur pour l'instant. Le joueur peut alors, en plus de son serpent, distinguer les pommes (carrés rouges) disposées sur la carte de façon aléatoire. Il commence par avancer vers la droite et il peut se diriger avec les flèches directionnelles. Comme dit précédemment, il peut manger des pommes pour grandir et voir son score augmenter de 5 pour chaque pomme mangée s'il touche un mur ou bien son propre corps, il perd la partie (cf. captures d'écran 4 et 5) Comme dit précédemment, le joueur peut décider de faire pause à tout moment en appuyant sur la touche « espace » ou quitter en appuyant sur « échap ».

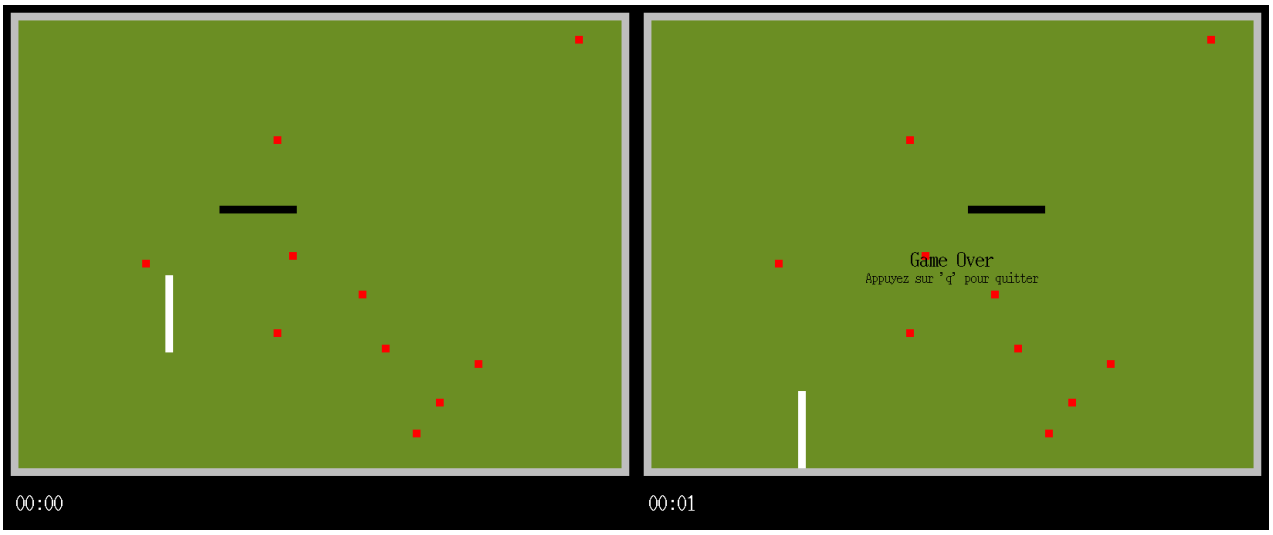


Capture d'écran numéro 4 : Écran un joueur



Capture d'écran numéro 5 : Écran de fin un joueur

Revenons sur le mode multijoueur. Lorsque le joueur lance le mode multijoueur, il tombe sur l'écran de chargement de ce mode (capture d'écran n° 3) puis, lorsqu'il appuie sur la touche « espace », il est redirigé vers la fenêtre de jeu multijoueur. Comme expliqué plus haut, deux joueurs peuvent jouer simultanément : le joueur un avec le serpent noir et les flèches directionnelles et le joueur deux joue le serpent blanc avec les touches « ZQSD ». Le but de ce mode de jeu est un peu le même que le premier mais ayant pour but de plus de coincer le serpent adverse (un peu comme dans le film *Tron* : Deux joueurs s'affrontent à moto et laissent des trainées derrière eux et ont pour but de coincer le joueur adverse). Ici, le terrain est doté de 10 pommes contrairement au mode un joueur qui en a 4. Comme dans le mode un joueur, dans ce mode, les joueurs ont la possibilité de faire pause en appuyant sur la touche « espace » ou bien de quitter en appuyant sur la touche « échap ».



Capture d'écran numéro 6 : Écran multijoueur

Capture d'écran numéro 7 : Écran de fin multijoueur

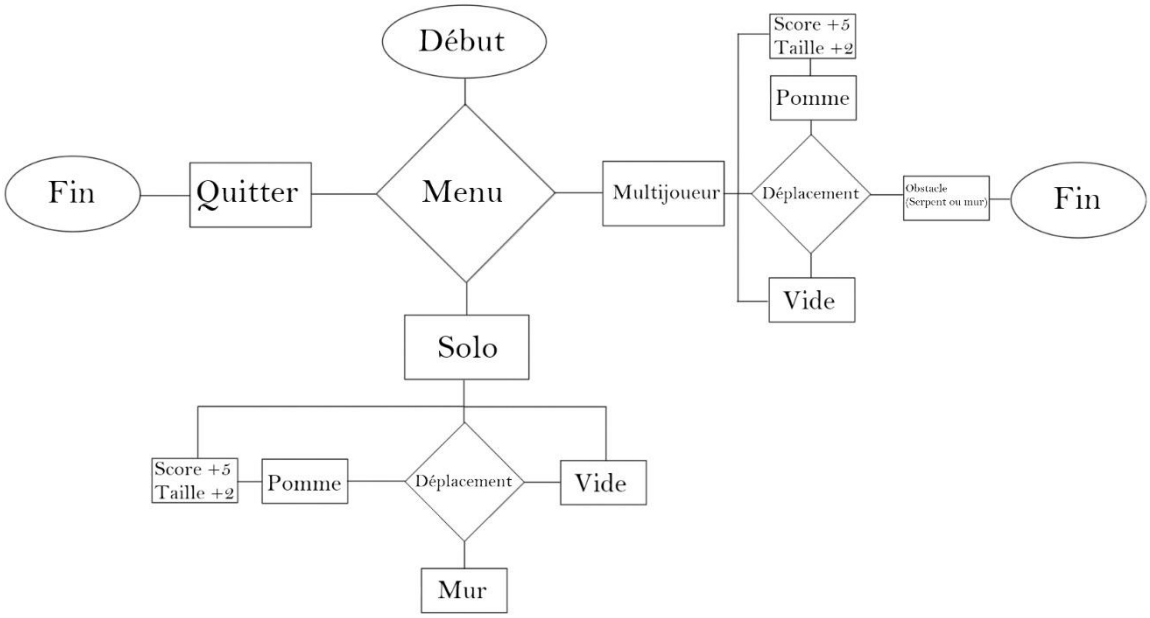
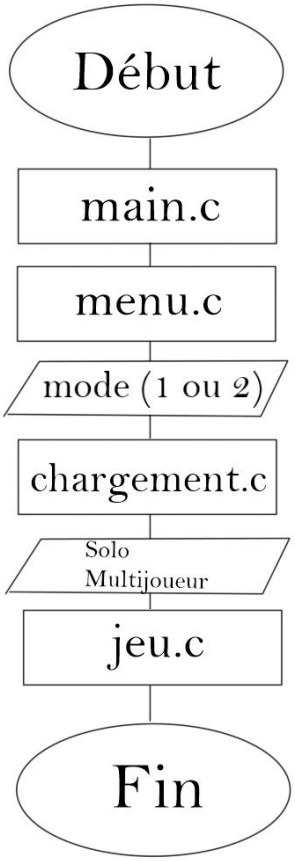


Diagramme de fonctionnement du programme du Snake

Nous avons présenté le jeu, nous allons maintenant vous expliquer comment nous avons programmé ce dernier à l'aide d'un diagramme commenté.

Présentation de la structure du programme

Vous pouvez voir ci-contre l'organigramme de notre programme. Nous allons donc vous expliquer le fonctionnement de cet organigramme et donc par conséquent, de notre programme.



Tout d'abord, « Début » représente le début à l'exécution du programme. Ensuite, *main.c* est lancé par défaut au début. Puis *main.c* renvoie l'utilisateur vers *menu.c* qui est le programme, comme son nom l'indique, qui gère le menu. En fonction du mode de jeu que choisi l'utilisateur, *menu.c* renverra une valeur de 1 ou 2 (1 pour le mode un joueur et 2 pour le multijoueur). *Chargement.c* utilise ensuite les valeurs retournées par *menu.c* pour déterminer si le mode de jeu est un ou deux joueurs. Enfin, après avoir sélectionné le mode de jeu, le programme exécute *jeu.c* qui lance l'interface du jeu afin que le joueur puisse jouer. Le programme se termine lorsque l'on sort de *jeu.c*.

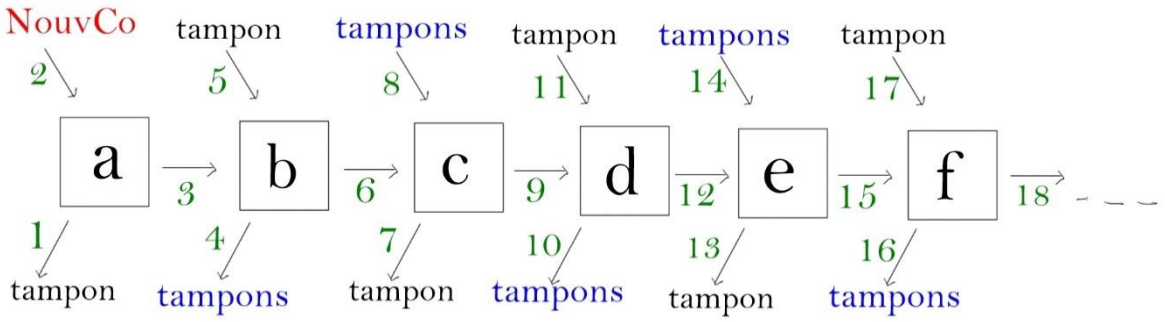
Représentation du serpent dans le code :

Nous allons à présent vous expliquer comment nous avons créé notre serpent et comment nous le faisons se déplacer ainsi que les fonctions additionnelles ou de test que nous avons utilisées.

Pour créer notre serpent, nous avons décidé d'utiliser une liste chaînée car son utilisation semble plus « logique » étant donné que le mouvement d'éléments en éléments de la liste est semblable à celui d'un serpent (la tête reliée au corps et toutes les parties du corps qui font le même mouvement). De plus, la liste chaînée est un avantage par rapport au tableau/tableau dynamique car celle-ci comble les trous disponibles dans la mémoire pour créer ses éléments.

Nous avons associé le premier élément de la liste chaînée à la tête du serpent et les éléments suivants aux éléments du corps du serpent. Nous avons donc tout d'abord créé cette liste chaînée. Ensuite, nous avons plusieurs fonctions d'insertion : une fonction d'insertion au début de la liste chaînée (devant le premier élément) et une fonction d'insertion à la fin de la liste (derrière le dernier élément). Nous avons fait ces deux fonction afin de voir quelle manière était la plus efficace pour créer notre serpent : nous en avons conclus que l'insertion en fin de liste était plus pratique pour la création

de notre serpent et le repérage des éléments (tête et queue). Nous avons aussi une fonction suppression qui permet de supprimer un élément de la liste (pour effacer le dernier élément par exemple). Nous avons aussi créé une fonction qui nous permet d'afficher le serpent (qui ne servait que pour les test avant l'affichage du serpent, lors de la création de la liste). Une autre fonction vérifie si le serpent ne se rentre pas dans lui-même (car si oui, le jeu est fini d'après les règles). Ensuite, nous avons deux fonctions qui contiennent les coordonnées respectives du premier et du dernier élément de la liste, ce qui nous permet de savoir à tout moment la position du serpent sur la carte ainsi que la position de sa queue. Enfin, nous avons créé une fonction de déplacement qui sert à déplacer tous les éléments de la liste chaînée : nous allons vous l'expliquer sous forme de schéma :



Tout d'abord, l'élément actuel, le premier élément de la liste, donne ses coordonnées au tampon afin de les conserver pour le suivant. Il reçoit ensuite les nouvelles coordonnées transmises par *jeu.c*. Ensuite, l'élément actuel avance dans la liste (il passe de a à b). Maintenant, le nouvel élément actuel donne ses coordonnées au tampons (pour tamponSuivant) et prends les coordonnées du premier élément depuis tampon. On avance dans la liste. Le nouvel élément actuel donne ses coordonnées au tampon, qui est maintenant libre et prends ensuite les coordonnées de tampons (celles du deuxième élément). On avance dans la liste, etc… On répète ce processus en boucle jusqu'à arriver à une valeur de la liste NULL (qui désigne la fin de liste). Nous appellons cette fonction à chaque fois que le serpent se déplace sur le terrain.

Conclusions personnelles :

Matthieu :

Pour ma part j'ai apprécié faire ce projet tout simplement puisque je trouve ça toujours amusant de revisiter les « classiques » du jeux vidéo, c'est d'ailleurs pour cela que ça m'a tenu a cœur de faire un mode de jeux qui sortait un peux de l'ordinaire. Et puis apres comme d'habitude ça reste toujours très interessant de travailler avec des personnes différentes puisque cela m'a permis de découvrire des nouvelles façons de procédés et de réfléchir.

Paul :

De mon côté, j'ai trouvé ce projet assez intéressant car il m'a permis de prendre en main peu à peu la bibliothèque graphique de l'IUT avec un projet qui n'étais ni trop simple ni trop compliqué (même si on a eu quelques après-midis de recherches pour résoudres certains bugs ou créer des fonctions (comme le déplacement de la liste chaînée)). De plus, ce projet m'a permi de voir en avance le principe de liste chaînée. La difficulté qui aura été, pour moi la plus grande, est le côté graphique car nous n'avions pas vraiment eu de cours dessus même si la bibliothèque était plutôt simple d'utilisation. Malgré cela, ce projet nous a permis de créer notre (du moins mon) premier jeu vidéo. Le mieux a été que je connaissais déjà le jeu et cela m'as donc plus de pouvoir « customiser » moi-même un jeu auquel j'ai déjà joué.