2019

PROJET APL 2019

SNAKE

SALEH ALADIN

SALEH Aladin
Groupe 3

SOMMAIRE

SALEH Aladin
Groupe 3

INTRODUCTION

Pour mon premier projet en APL, cette année j'ai dû réaliser à l'aide des connaissances acquise durant ces premiers mois un « Snake », ou jeu du serpent en langage C.
Pour cela , j'ai eu la bibliothèque graphique proposée par l'IUT (inspirée de la librairie X11).
Le jeu du Snake est un jeu très connu et très ancien, on le retrouvais notamment dans le célèbre Nokia 3310.
Le but du jeu est simple ; le joueur doit à l'aide de flèches directionnelle dirigé un serpent qui aura pour objectif de grandir en dévorant des pommes , cependant, plus celui-ci dévorera de pommes, et plus celui-ci verra sa vitesse de déplacement augmentée rendant le jeu plus compliqué !

FENETRE DE JEU ET AFFICHAGE:

La fenetre du jeu et logiquement créer à l'aide des fonctions InitialiserGraphique() et CreerFenetre().
La fenêtre du jeu est de LARGEURxHAUTEUR soit dans notre cas 800x600

Toutefois dans cette fenêtre je crée un rectangle de 600x400 de couleur verte qui sera notre vraie zone de jeu (ou le serpent se déplace)
Enfin, pour maintenir ce programme je crée un boucle infinis qui maintient la fenêtre de jeu. Cette fenêtre de jeu ne se fermera que via la touche Echap ou bien si le joueur perd.

FONCTIONNALITE DU JEU :

1)Création du serpent/Son déplacement

Comment est-ce que j'ai créer le corps de mon serpent dans mon programme ?
Initialement mon serpent se trouve être un simple cube de taille 10x10

(Rectangle plein) que je fais apparaître au milieu de mon terrain de jeu. Les coordonnées du serpent sont ensuite enregistrer dans un tableau à deux dimensions. Si j'ai enregistré les coordonnées de mon serpent c'était pour pouvoir les réutiliser lorsque je devais faire grandir mon serpent, cependant, vous verrez un peu plus loin dans mon rapport que tout ne s'est pas passé comme prévu…
Ainsi, à l'aide d'une boucle je créer le serpent de taille 10 en enregistrant les coordonnées de chaque partie du serpent.

Maintenant, attaquons-nous à la partie qui gère le déplacement du serpent ! Il y a 2 types de déplacement, en premier, le déplacement qui se fait automatiquement (sans que l'on est besoin d'appuyer sur une touche),et ensuite il y a les déplacement avec les touches du clavier. Commençons d'abord par le plus simple, le déplacement automatique.

Le déplacement automatique vas se lancer au début du programme, lorsque le joueur vas appuyer sur une des flèches directionnelle, le serpent vas prendre sa direction en prennent +1 ou -1 sur une de ses coordonnées « x » ou « y » à chaque seconde. Cela se fait grâce au rafraichissement de la fenêtre de jeu qui est fait grâce à la fonction de la bibliothèque graphique « Microsecondes() » . Cette fonction est placée dans une condition qui se trouve dans la boucle qui « maintient » le programme.

Maintenant, attaquons nous au déplacement via les touches !
Tout d'abord, il m'a fallu initialiser la fonction ToucheEnAttente() dans une condition qui se trouve elle aussi dans la boucle du programme.
Dans cette condition j'ai déclaré une variable (int) nommé dplmt à laquelle on assigne la fonction Touche().
Ainsi, pour chaque touche directionnelle j'ai créer un condition tel que : if(dplmt == « une touche »){déplacement}

Enfin, après avoir défini chaque touche dans des conditions, on remplit ces conditions par le déplacement de coordonnées.
Ainsi pour aller vers le haut : y = y -1 ;
pour aller vers le bas : y = y +1 ;
pour aller vers la gauche : x = x – 1 ;
pour aller vers la droite : x = x + 1 ;

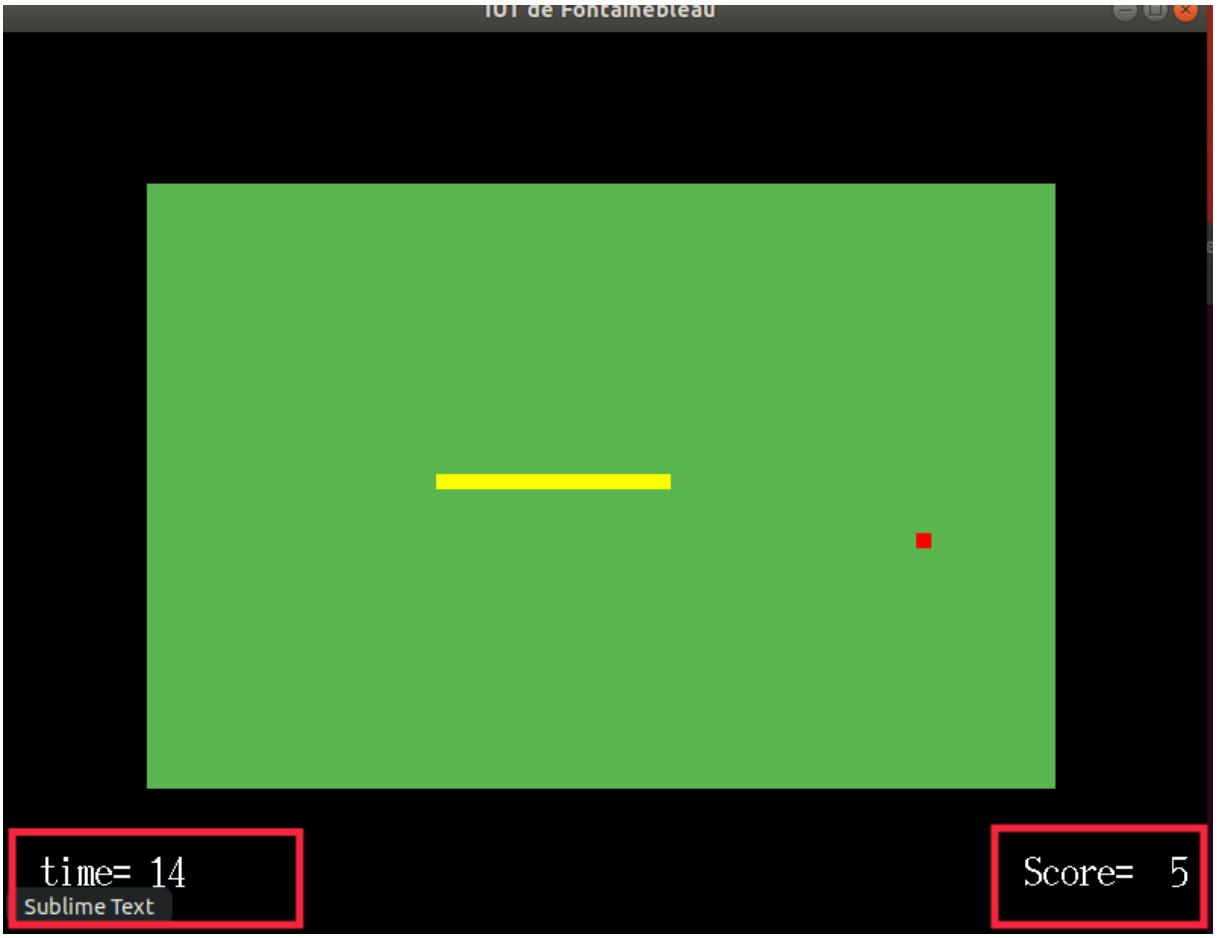
2)Timer du jeu / Score

Lorsque le joueur va exécuter le programme, il pourra voir en bas à gauche un timer qui affiche le temps en seconde.
Comment marche cette fonction ?

Le timer fonctionne grâce à une fonction et a une condition dans la boucle principale du jeu, en effet à chaque rafraichissement de la fenêtre le timer voit sa valeur précédente « écrasé » par la nouvelle qui prend +1. Elle effectue cette action tant que le programme ne c'est pas arrêter ou si le joueur est encore en jeu. Pour afficher le timer on utilise sprintf() ; qui est un printf qui à la capacité de stockée ce qu'elle imprime dans un char[]. Ainsi, cela nous permet ensuite d'utiliser la fonction EcrireTexte() de la bibliothèque graphique.

Maintenant que nous avons vus comment marche le timer, lançons-nous dans l'explication du score !
Le score fonctionne à peu près comme le timer à la seule différence que celui-ci ne se met à jour que lorsque le joueur va dévorer une pomme avec son serpent

NB : Il se peut que selon la taille de votre écran le score ne s'affiche qu'a moitié ou bien qu'il soit coupé, dans ce cas-là je vous invite à agrandir la fenêtre de jeu via la souris.



3)Collision avec les bordures

Abordant maintenant une des fonctions les plus importante du programme !
Les collisions avec les bordures a été l'une des fonctions les plus simples à réaliser, en effet pour cela il ne fallait que « joué » avec les coordonnées en effet le programme vas toutes les secondes comparer les coordonnées du serpent à celle des bordures de la zone de jeu.
Lorsque elles sont égales alors le programme vas afficher « GAME OVER ! » et initialiser la vitesse de rafraîchissement à une grande valeur pour qu'il soit plus lent (en vérité c'est comme si le programme s'était arrêter car devenu trop lent)

NB : De base, dans sa première version mon programme était censé directement fermer la fenêtre en appelant la fonction FermerGraphique() ; mais je l'est changé pour que l'on puisse afficher « GAME OVER ! », le seul

inconvénient c'est que pour fermer le programme, le joueur doit utiliser la souris.

4)Coordonnées des pommes et affichage

Maintenant voyons une autre fonction importante du jeu qui n'est autre que les pommes !

NB : Initialement le projet demandé de générer 5 pommes cependant cela m'a causé plusieurs problèmes de fluidité et de score, et il me semble qu'il y a toujours eu qu'une pomme dans le jeu de base donc j'ai décider de n'en générer qu'une.

Cette fonctionnalité est découpée en 3 fonctions dans mon programme :

- **La génération des coordonnées x et y des pommes.**
- **L'affichage de la pomme.**
- **Le test de collision entre la pomme et le serpent.**

GENERATION DES COORDONNEES X E Y :

Cette fonction vas tout simplement générer des coordonnées aléatoire (je les aient les nommés xp et yp) comprise entre les coordonnées xmax xmin et ymax ymin de la zone de jeux, de plus dans le cas ou une pomme est générer en dehors de la zone de jeu elle est automatiquement déplacer dans la zone de jeu.

NB : La boucle est présente au cas ou vous voulez changez le nombre de pomme générer

AFFICHAGE DE LA POMME :

L'affichage est simple, dans une fonction nommé PommeAffichage() ; le programme vas affiché une pomme (soit un rectangle de couleur rouge) qui a pour coordonnées x,y celle de xp et yp .

NB : Ici aussi on retrouve une boucle au cas ou vous voudriez generer plus qu'une pomme.

TEST DE COLLISION ENTRE LA POMME ET LE SERPENT :

Le test de la collision entre la pomme et le serpent marche de la même manière que les collisions avec les bordures sauf qu'ici on doit vérifier et comparer les coordonnées x,y du serpent avec les coordonnées xp,yp de la pomme.
Cependant dans ce cas suivant j'ai rencontrés un problèmes de marge d'erreur, d'où le fait que je compare x et y avec xp +/- une valeur inferieur à 5 et yp +/- une valeur inférieur à 5 . Cela peut paraitre assez « sale » et pas très professionnel, mais c'était la seule solution que j'avais trouvé avec le temps que j'avais

NB : J'avais pensé à stocker les pommes dans un tableau de 60x40 avec le serpent et dès que le serpent et la pomme avait le même indice considérer une collision.

Enfin, des qu'il y a une détection de détecter la fonction va régénérer de nouvelle coordonnées et l'affiché.
De plus elle vas augmenter la valeur du score de +5 .
(La variable du score se nomme nscrs et vaut 0 de base)

5)Accélération du serpent

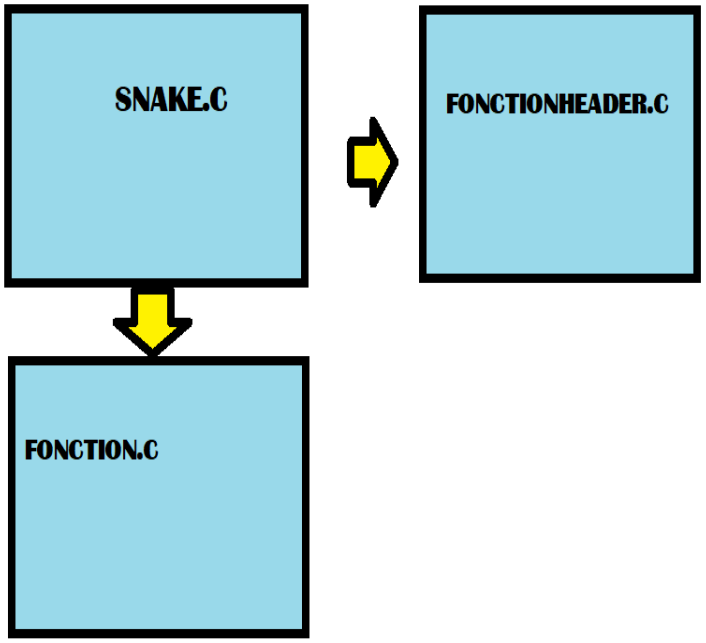
Pour terminer avec les fonctionnalités du jeu, parlons de l'accélération !

En effet dès que le joueur vas atteindre un certains score (100,300,500,1000), le serpent vas voir sa vitesse de déplacement accélérer.
Comment est-ce que j'ai fait ça ?

Cela est assez simple, en effet, il suffit de d'accélérer le taux de rafraîchissement. Pour cela il suffit juste de réduire la valeur « vitesse » qui est une variable à laquelle on avait de base donné la valeur 10000 , plus cette valeur baisse , et plus le serpent se déplacera vite

DIVISION DES SOURCES

Mon projet est divisé en deux fichier, le snake.c qui contient le main, et fonction.c qui comme son nom l'indique contient toute les fonctions



PROBLEME PRESENT

Comme vous avec pu le constater lors de l'exécution du programme, certaines fonctionnalités n'était pas présentes à commencé par le corps du serpent qui n'a pas une taille initiale de 10 et qui n'a à vraiment dire pas de vrais « corps » , cela est du au fait que je n'est pas eu le temps pour mettre mes idées au propre et que je rencontré plusieurs erreurs .
Je me permet donc de vous expliquer les différentes idées que j'envisageais :

Tout d'abord pour le corps du serpent je comptais me servir du tableau dans lequel j'enregistrais les coordonnées pour délimiter le corps en 3 parties : la tête, la queue, et ce qu'il y a entre la queue et la tête, c'est pour cela que dans la création du serpent au début je le met dans une boucle qui s'arrête lorsque la taille du serpent vaut 10.

Pour cela j'envisageais aussi d'utiliser une structure pour la tête et la queue.
Enfin comme je n'ai pas vraiment de corps pour le serpent je n'ai pas pu gérer l'augmentation du serpent lorsqu'il consomme une pomme, mais pour ce problème, j'avais pensé à augmenter la taille du serpent en ajoutant une nouvelle partie à la fin du serpent et en faire la nouvelle queue.

Enfin dernier problème rencontré, il s'agit du Makefile, en effet selon les machines sur lesquelles se lance le programme il arrive que les fonctionnalités du snake ne soit pas activé, ainsi certaine fois on se retrouve avec un timer qui se met à jour beaucoup trop vite, ou encore que les pommes ne changent pas de position après qu'il y est eu une collision en elles et le serpent.
Pour pallier à ce problème, j'ai créer un fichier de secours nommé « SiLeSnakeNeMarchePasBien' » contenant le code du Snake en brut

CONCLUSION PERSONNELLE

Si je devais conclure, je dirais que ce projet a été très instructif et intéressant tout en étant aussi déstabilisant.
En effet, tout d'abord si je dit que ce projet a été instructif et intéressant c'est d'abord due au fait que pour la première fois dans ma « carrière scolaire » j'ai été lâché dans le « vide » et de ce fait j'ai du tout apprendre par moi-même et savoir user des compétences que je venais tout juste d'acquérir via les TD/TP d'APL.
De plus, bien que pour la réalisation de ce projet j'était seul j'ai trouvé cela super intéressant de pouvoir comparer l'avancé des projet avec les autres groupes/élèves et comparé les différentes façons de penser .
Cependant, j'ai trouvé cette expérience assez déstabilisantes car comme je l'ai dit plus haut c'est la première fois que je n'était pas encadrer par un professeur pour la réalisation d'un projet, je pense que cela m'a donné un aperçus du monde professionnelles .
Enfin, j'ai surtout était déstabiliser du au fait que j'ai du réaliser ce projet seul donc je n'est pas pu confronté diffèrent point de vue , et étant encore un novice dans la programmation en langage C cela a était une dose assez conséquentes de travail pour un laps de temps peut être trop court cependant je n'en retire que du positif.