Réalisation de programmes

Analyse & modification du projet

Dans ce compte rendu nous allons vous présenter le projet Flappy Bird qu'on a eu pour l'étape 4. Ceci consiste à étudier le projet des autres, le comprendre et apporter des modifications.
Ce projet est composé de plusieurs fichier.

1)

Nous allons expliquer le contenu de chaque fichier dans la suite :

-cerveau.c

Ce fichier permet de jouer

-collision.c

Il permet de détecter s'il y a une collision entre le joueur et les obstacles.

-header.h

Ce fichier inclus tous les prototypes de toutes les fonctions.

-inis.c

Il initialise les positions des obstacles et le score.

-mainF.c

Il initialise des variables globales et lance le programme menu.c.

-Makefile

Il rassemble tous les fichiers *.c pour constituer ce programme.

-menu.c

Il affiche le menu du jeu sur l'écran.

-menuoption.c

Il affiche les options du menu.

-menurejouer.c

Il permet de rejouer.

-menuscore.c

Il affiche le meilleur score du jeu.

2)

Ceci étant terminé nous allons vous expliquer le rôle de chaque fonction de chaque fichier en brève.

cerveau.c :

a.description

Ce fichier il contient qu'une seule fonction qui est : void jouer (void) ;
Cette fonction permet d'initialiser tous les objets du jeu, elle définit aussi leurs positions de façon aléatoire avec rand().
Dans cette fonction on effectue le déplacement des objets et les obstacles.
Il fait la mise à jour des positions.

b.critique

Dans void jouer (void) ; les initialisations (images,sons) et le jeu sont regroupés dans une seule fonction. Il serait préferable de les séparer en quatre fonctions différentes, et d'ajouter une fonction game over.
Void init_images() ;
Void init_sons() ;
Void jouer() ;
int gameOver() ;

collision.c :

a.description
int Collision (SDL_Rect positionFlap);
int Collision2 (SDL_Rect positionFlap);
Ces deux fonctions testent si le joueur rentre en collision avec un obstacle(tuyaux). La première fonction vérifie ce test avec les tuyaux qui se situent sur la partie haute de l'écran et la deuxième avec les tuyaux du bas.

b.critique
Ici au lieu de faire deux fonctions on aurait pu les rassembler en une seule fonction parce qu'il y a une seule condition qui les différencies.
Le nom de la fonction n'est pas clair, car on ne distingue pas la collision entre le joueur et l'obstacle.

i**nis.c :**
a.description

void score (int sco) ;
Cette fonction permet de comparer le score actuel avec le meilleur score, et si c'est le cas il écrit le score dans le fichier « score.txt ».

int randii (int a, int b);
Elle renvoie une valeur à l'aide de la fonction rand().

void pos (void);
Cette fonction définie les différentes positions des obstacles (en haut et en bas).

b.critique

Dans ce fichier il y a trois fonctions différents qui n'ont pas de rapport entre elles. On aurait pu mettre la fonction void score(int sco) dans le fichier **menuscore.c .** Et la fonction void pos(void) aurait pu etre dans le fichier cerveau.c car dans cette fonction on initialise les obstacles.

mainF.c :
Ce fichier contient la fonction main, ici on lance la fonction « menu()» pour démarrer le jeu.Il y a aussi des variables globales qui sont initialisées.

menu.c :
void menu (void) ;
Dans cette fonction on initialise l'écran et on crée la page menu. On définit 3 boutons sur la page : jouer, score et option.
Jouer permet de commencer le jeu, score permet de voir le meilleur score et puis option permet de choisir le personnage avec le quel le joueur voudrait jouer.

menuoption.c :
void menuoption (void);
Cette fonction permet d'avoir la possibilité de changer le personnage du jeu. On peut choisir soit FlappyBird, soit dragon ou bien avion(sonic).
Les images des personnages sont inistialisées et chargées ici. Chaque personnage correspond à plusieurs images.
L'image du son est aussi initialisé dans cette fonction.

menurejouer.c :
void menujouer (void);
Ici on permet au joueur de redémarrer le jeu.
On initialise l'image « rejouer » et on relance la fonction « jouer() ».

menuscore.c :
void menuscore (void);
Cette fonction permet d'afficher le meilleur score du jeu qui est stocké dans le fichier score.txt.
On peut voir ce score à partir de l'option menu => score.

3) Globale – Structure – Enum

Dans ce programme il n y a pas d'enum. Il y a deux structures qui sont globale et pos.
La structure Globale contient toutes les données du jeu. Pour notre part nous aurions fait autrement, nous aurions fait des structures différentes pour chaque objet, par exemple pour le joueur principal « Flappy » : struct flappy{

```
SDL_Surface *img ;
int x,y ;
int vitesse;} ;
```

La 2ème structure c'est pos, celle ci contient toute les positions des tuyaux(obstacles).

Il manque des variables globales car dans le code il y avait des valeurs constantes qui par exemple definissait les positions de l'image, la taille de l'écran...

4) Conclusion :
Le code en lui meme nous parraisait plutot compliqué à l'apperçu car les noms des fichiers n'étaient pas en cohérence avec les fonctions à l'intérieur. Il y avait trop de ligne inutile dans les fonctions par exemple pour les chargements d'images
on a économisé plus de 100 lignes de codes juste en faisant une boucle for qui permet d'effectuer le même travail donc on peut en déduire que le code n'est pas optimal.

Après avoir étudier le programme, nous avons eu quelques modifications à faire afin d'améliorer le jeu. Nous avons eu trois modifications.

Partie 2 :

Rajouter des objets :

Nous avons réussi à ajouter des objets comme l'étoile qui font augmenter le score de 50 points. Nous avons crée un tableau qui contient chacun des objets. Nous avons aussi fait en sorte, qu'une fois un objet est capturé par le joueur, il apparait à nouveau dans une position aléatoire.

Détection de la collision :

Le joueur traversait les obstacles sans que la partie se termine. Ceci était à cause d'une erreur d'affectation de valeur dans la variable « bonus ». La valeur de « Bonus » était égale à 2 d'ou les conditions de collisions n'étaient pas verifiées. Il fallait affecter la valeur 0 à cette variable pour pouvoir rentrer dans les conditions de collision et terminer ainsi le jeu.

Agrandir la fenêtre :

Cette partie est un peu compliqué à modifier car premièrement la taille de l'image est fixe. Nous avons essayé d'agrandir la fenêtre avec SDL_Resizable mais lorsqu'on a agrandi la fenêtre, l'image resta fixe sur sa taille initiale. Nous avons trouvé une méthode pour agrandir l'image mais ceci difficile à réaliser par manque de temps. Il faut programmer une fonction qui puisse faire ce travail et ceci nous demande beaucoup de temps.