Projet Programmation : Les Avions

D.Soupilas-A.Pashinin

1 Jeu

1.1 Players

Les joueurs appartiennent à une structure "players". Chaque joueur (Player1 et Player2) est du type Player, a chaque player on définit son score qui change au fur et à mesure du jeu, le x_loc qui varie à chaque mouvement du joueur, le nom du joueur qui est demandé à chaquedébut de partie, le coordonné de son laser qui varie, une taille qui est stricte et ne varie pas pendant le jeu et un entier game_on .

```
typedef struct{
  int score;
  int x_loc;
  char name[50];
  int x_laser;
  int taille;
}player;

typedef struct{
  player Player1;
  player Player2;
  int game\_on;
}players;
```

1.1.1 void laser_P1(players *P, int i) ;

Cette fonction sert à l'apparition graphique du laser du Player1 pendant le jeu, ainsi que l'affectation d'une valeur au x_laser du Player1, elle prend en argument un type Players et un entier.
La fonction charge une image L1="turret_attack_blue.png" qui représente l'attaque du joueur 1 .
Lors de l'appel de cette fonction, on affecte au x_laser du Player1 la valeur : x_loc du Player1 + (taille du Player1 / 2).

(P->Player1.x_laser)=P->Player1.x_loc+(P->Player1.taille/2) ;

Puis la fonction dessine l'image L1 en position x_loc du Player1.

1.1.2 void laser_P2(players *P, int i) ;

Cette fonction sert à l'apparition graphique du laser du Player2 pendant le jeu, ainsi que l'affectation d'une valeur au x_laser du Player2, elle prend en argument un type Players et un entier.
La fonction charge une image L2="turret_attack_blue.png" qui représente l'attaque du joueur 2 .
Lors de l'appel de cette fonction, on affecte au x_laser du Player2 la valeur : x_loc du Player2 + (taille du Player2 / 2).

(P->Player2.x_laser)=P->Player2.x_loc+(P->Player2.taille/2) Puis la fonction dessine l'image L2 en position x_loc du Player2.

1.1.3 void pos_player1(players *P)

Cette fonction permet le mouvement graphique du Player1 en posant des contraintes qui interdisent au joueur de sortir de l'écran, elle prend en argument un type Players.
La fonction charge un son laser1="Shot_01.ogg" qui représente le son lors de l'apparition du laser, et une image Pl1="turret.png" qui est la représentation graphique du Player1. Si la touche flèche droite est appuyée l'image de déplace d'un certain nombre de pixels a droite, si la touche flèche gauche est appuyée l'image de déplace d'un certain nombre de pixels a gauche.
Si la touche espace est appuyée son laser est joué et la fonction laser_P1 est appelée.
De meme le score courant du Player1 est affiché sur l'image.

1.1.4 void pos_player2(players *P)

Cette fonction permet le mouvement graphique du Player2 en posant des contraintes qui interdisent au joueur de sortir de l'écran, elle prend en argument un type Players.
La fonction charge un son laser2="Shot_01.ogg" qui représente le son lors de l'apparition du laser, et une image Pl1="turret.png" qui est la représentation graphique du Player1. Si la touche 'd' est appuyée l'image de dÃ©place d'un certain nombre de pixels a droite, si la touche 'a' est appuyée l'image de déplace d'un certain nombre de pixels a gauche.
Si la touche 'w' est appuyée son laser est joué et la fonction laser_P1 est appelée.
De meme le score courant du Player2 est affiché sur l'image.

1.1.5 void actu_players(players *P)

Dans cette fonction prend en argument un type players, elle appelle les deux fonctions pos_player1 et pos_player2.

1.1.6 int exist(char *i, int j, char *a, int b)

Au moment de sauvegarder les scores a la fin d' une partie cette fonction check si le meme score avec le meme prenom existe deja.

1.1.7 void ajouter(char * i, int j)

Cette fonction ajoute un string et un int qu'il prend en parametre dans le fichier 'score.txt.'.

1.1.8 void print_scores(players *P)

Cette fonction prend en argument un type players, elle sert a afficher les scores des 2 joueurs a la fin d'un partie a deux joueurs. Elle appelle la fontion ajouter afin de sauvegarder les scores des joueurs.

1.1.9 void print_score(players *P)

Cette fonction prend en argument un type players, elle sert a afficher le score du Player1 a la fin d'une partie a un joueur. Elle appele la fonction ajouter afin de sauvegarder le score du joueur.

1.1.10 void get_names(players *P)

Cette fonction prend en argument un type players, elle sert a afficher deux input boxes au debut d'un partie a deux joueurs dans lequels les joueur rentrerons leurs prénoms.
Ces prénoms seront stockés dans le name de chaque joueur.

1.1.11 void get_name(players *P)

Cette fonction prend en argument un type players, elle sert a affiche un input box au debut d'une partie a un joueur dans lequel le joueur rentre son nom.
Ce prénom sera stockÃ© dans le name du Player1.

1.2 Explosion

Cette fonctions crée l'animation d'une explosion dans les coordonnes donnes avec un son d'explosion.

1.3 Planes

Les avions appartienent a une structure "planes".
Chaque avion est de type plane, il existe 3 types d'avions,hely1,hely2 et plane1.
A chaque avion on dÃ©ffini une position x qui varie avec le mouvenment de l'avion (voir hellycop1droite), une taille et un nombre de points.

```
typedef struct{
  int x;
  int taille;
  int points;
}plane;

typedef struct{
  plane hely1;
  plane hely2;
  plane plane1;
}planes;
```

1.3.1 Planes_Droite

Animations des avions aparaissant de la droite.

1.3.1.1 int killedplane(planes *GR, players *P) Cette fonction sert a voir la mort d'un plane (s'il est touché). Elle retourne 1 si le plane est touché et 0 si il est toujours vivant. Elle prend en argument un type planes et un type players.
Si le x_laser du Player1 ou du Player2 = x du plane alors la fonction retourne 1 sinon il retourne 0.
Appelle la fonction explosion si l'avion est touché.

1.3.1.2 int killedhely1(planes *GR, players *P) Cette fonction sert a voir la mort d'un hely1 (s'il est touché). Elle retourne 1 si le hely1 est touché et 0 si il est toujours vivant. Elle prend en argument un type planes et un type players.

Si le x_laser du Player1 ou du Player2 = x du hely1 alors la fonction retourne 1 sinon il retourne 0 ;
Appelle la fonction explosion si l'avion est touché.

1.3.1.3 int killedhely2(planes *GR, players *P) Cette fonction sert a voir la mort d'un hely2 (s'il est touché). Elle retourne 1 si le hely2 est touché et 0 si il est toujours vivant. Elle prend en argument un type planes et un type players.

Si le x_laser du Player1 ou du Player2 = x du hely2 alors la fonction retourne 1 sinon il retourne 0 ;
Appelle la fonction explosion si l'avion est touché.

1.3.1.4 void helycop1droite(planes* GR, players* P, int pos) Cette fonction prend en argument un planes, un type players et un entier, ce dernier represente le coordoné x du début de l'animation.

Cette fontion construit l'animation graphique du hely1, elle prend des photos et crÃ©e l'animation du petit helicopter rouge. Sa boucle while sert a orchestrer tout le jeu puisque elle dessine l'image de background et elle appelle la fonction actu_players pandant que l'hely1 est PAS MORT.

⚠A chaque debut de boucle les valeur des x_laser des deux joueurs est mise a -1000, si on neglige cette etape la valeur de x_laser reste dans le jeu ce qui provoque une mort aux avions automatique sans action du joueur.

1.3.1.5 void helycop2droite(planes* GR, players* P, int pos) Cette fonction prend en argument un planes, un type players et un entier,ce dernier represente le coordone x du debut de l'animation.

Cette fontion rconstruit l'animation graphique du hely2, elle prend des photos et cree l'animation du moyen helicopter rouge. Sa boucle while sert a orchestrer tout le jeu puisque elle dessine l'image de background et elle appelle la fonction ACTU_PLAYERS pandant que l'hely2 est PAS MORT.

⚠

1.3.1.6 void planedroite(planes* GR, players* P, int pos) Cette fonction prend en argument un planes, un type players et un entier,ce dernier represente le coordone x du debut de l'animation.

Cette fontion construit l'animation graphique du plane, elle prend des photos et cree l'animation de l'avion civile. Sa boucle while sert a orchestrer tout le jeu puisque elle dessine l'image de background et elle appelle la fonction ACTU_PLAYERS pandant que le plane est PAS MORT.

⚠

1.3.2 Planes_Gauche

1.3.2.1 void helycop1gauche(planes* GR, players* P, int pos) Miroir de helycop1droite.

1.3.2.2 void helycop2dgauche(planes* GR, players* P, int pos) Miroir de helycop2droite.

1.3.2.3 void planegauche(planes* GR, players* P, int pos) Miroir de planedroite.

1.4 Game

Ce fichier orchestre tout le jeu, il appelle les fonction precedentes afin de construire le jeu et son petit menu.

1.4.1 int randomise()

Cette fonction choisi aleatoirement un nombre, si le nombre est pair elle retourne 1 sinon elle retourne 0 ;

1.4.2 int game_on(clock_t time_beg, int duration_sec)

Cette fonction sert comme timer pour le jeu, elle prend en argument un temps et une durée (en secondes), elle retourne 0 si le temps restant est supperieur a 0, sinon elle retourne 1.

1.4.3 void init_1P(planes *GR, players *P)

Cette fonction sert à initialiser une partie à un joueur, elle prend en argument un type planes et un type players.

Les scores des deux players sont mis a 0. Les tailles des deux players sont mises à la longuer de l'image des joueurs en pixels. Le x_loc du Player1 initialise l'image du joueur au milieu de l'ecran. Le x_loc du Player2 est initialisé dans les negatif (-1500). Chaque avion prend sa taille en pixels. Les coordones des Avions sont egalement initialisés dans les negatifs (-2000). Les x_lasers des joueurs sont egalement initialisés dans les negatifs (-1000). Game_on est initialisé à 1.

1.4.4 void init_2P(planes *GR, players *P)

Cette fonction sert à initialiser une partie à deux joueurs, elle prend en argument un type planes et un type players.

Les scores des deux players sont mis à 0. Les tailles des deux players sont mises a à la longuer de l'image des joueurs en pixels. Le x_loc du Player1 initialise l'image du joueur au 1/4 de l'ecran. Le x_loc du Player2 initialise l'image du joueur au 3/4 de l'ecran. Chaque avion prend sa taille en pixels. Les coordones des Avions sont egalement initialisés dans les negatifs (-2000). Les x_lasers des joueurs sont egalement initialisés dans les negatifs (-1000). Game_on est initialisé à 1.

1.4.5 void Players2_new(int duration)

Cette fonction orchestre tout le jeu a deux joueurs, elle demande les noms des joureurs, ensuit initialise toutes les valeurs, et finalement appelle les fonctions des avions en utilisant la fonction randomise pendant que le temps de la partie n'est pas ecoulé.
Si la touche echape est clické, elle appelle le menu du jeu.

1.4.6 void Player1_new(int duration)

Cette fonction orchestre tout le jeu a un joueur, elle demande le nom du joureur, ensuit initialise toutes les valeurs, et finalement appelle les fonctions des avions en utilisant la fonction randomise pendant que le temps de la partie n'est pas ecoulé.
Si la touche echape est clické, elle appelle le menu du jeu.
⚠Une partie à un joueur ne peut pas etre sauvegardée.

1.4.7 void game_load(planes* GR, players *P, int duration)

Cette fonction fonctione de la meme manière que les précedantes avec la seule difference qu'elle initialise pas le valeurs, les valeurs sont initialisés et traités au préalable. Elle recoit toutes les information sure les joueurs et les avions et execute le jeu.

1.4.8 menu-jeu

Utilise exactement le meme principe que le menu principal avec la seule difference la fonction savegamej qui sauve les scores et les noms des joueurs d'une partie a deux joueurs dans un fichier 'save.txt'..

2 Menu

2.1 void nav(int x1, int y1, int i)

Cette fonction sert a rendre le menu plus joli en rendant bleu les cases selectionnes par l'utilisateur.

2.2 void menu_princ()

Cette fonction affiche le menu principal.

2.3 void blinc_test(int i)

Cette fonction attend un click du joueur sur le menu. Pandant ce temps l'utilisateur peut 'naviguer' dans le menu avec la fonction nav. Lorsque l'utilisateur click sur son choix, la fonction button est appellé.

2.4 void button(int z, int x1, int y1)

Cette fonction prend trois arguments z, x1 et y1. Le 'z' represente la couche ou il se trouve et dependant de ce 'z' l'utilisateur peux clicker sur les options du menu qui lui sont proposés.
Nous avons choisis ce système de couches car lorsque l'utilisateur rentre dans une sous partie du menu si le systeme des couches existe pas alors le porgramme est en confli ne sachant pas ce qu'il doit faire ce qui resulte a une action qui n'est pas la bonne.
Chaque partie du menu a son propre 'z' qui lui est affilié.

2.5 void ab_authors()

Cette fonction affiche la sous partie du menu ou l'on peut trouver les informations sur les auuteurs. Son 'z' de retour vaut 5.

2.6 void newgame()

Cette fonction affiche la sous partie du menu ou l'on peut trouver les options de type de jeu (1 Joueur, 2 Joueurs). Son 'z' d' options vaut 1. L'utilisateur peut choisir une partie a un Joueur, ou une partie a deux Joueurs.

2.7 void cont_game()

Cette fonction fait partie de options de loadgame (si dessous), c'est elle qui charge l'anciene partie.
Elle prend les informations d'un fichier 'save.txt' qui lui contient les information de l'anciene partie, initialise la partie, charge les vielles informations et appelle la fonction game_load.

2.8 void loadgame()

Cette fonction affiche la sous partie du menu ou l'on a l'option de reprendre une partie sauvegardé au paravant. Son 'z' d' options vaut 1.

2.9 int read_scores(FILE* score, scores S)

Cette fonction ouvre le fichier contenant les scores des parties deja joues, les sauvegardes dans un tableau scores de type scorez et renvoi leur nombre.

2.10 arange_scores(scores S, int i)

Cette fonction trie le tableau scores et range les scores par ordre decroissant.

2.11 void tableau_score()

Cette fonction appelle les deux fonctions precedantes (read_scores et arange_scores) et ensuite dessine le tableau des scores.

2.12 void score()

Cette fonction affiche la sous partie du menu ou l'on voit les hightscores en appelant la fonction precedante (tablea_score). Son 'z' d'options vaut 4.

2.13 void gamemap()

Cette fonction affiche la sous partie du menu ou l'on voit la map du jeu.