Documentation Technique - Fichier graphic.c

Nous sommes Ilyass et Bryan, étudiant en première année à l'ESIPE, spécialité informatique. Voici la documentation technique de notre fichier graphic.c pour le jeu de type Shoot'em up en temps réel.

Dépendances

Ce fichier inclut plusieurs fichiers d'en-tête dont :

- `"../include/graphic.h"` : Fichier d'en-tête local contenant les définitions de types et les prototypes de fonctions utilisées dans `graphic.c`.
- `"../include/jeu.h"` : Fichier d'en-tête local pour gérer les aspects du jeu.
- `<MLV/MLV_all.h>` : Bibliothèque MLV pour la gestion de la fenêtre graphique.

Fonctions

Voici un aperçu des fonctions principales de ce fichier :

```
MLV_Image* initialize_image (char* image_path, int width, int height)
```

Cette fonction charge une image à partir d'un chemin donné (`image_path`) et redimensionne cette image à la largeur et la hauteur spécifiées (`width` et `height`). Elle retourne un pointeur vers l'image.

```
void draw_player(int index, Player* new_player, int up, int down, int right, int left)
```

Cette fonction dessine un joueur (`new_player`) en fonction de sa position actuelle et des commandes du joueur. Le paramètre `index` est utilisé pour déterminer quelle image du joueur utiliser. Les paramètres `up`, `down`, `right` et `left` indiquent la direction du mouvement du joueur.

```
void move_player(int index, Player* new_player)
```

Cette fonction déplace le joueur en fonction des touches pressées. Elle utilise la fonction `MLV_get_keyboard_state` pour détecter les touches pressées et appelle la fonction `draw_player` pour mettre à jour la position du joueur.

```
Heart* create_new_heart()
```

Cette fonction crée une nouvelle instance de `Heart` (un objet représentant un cœur dans le jeu), lui assigne une position aléatoire sur l'axe x et en haut de la fenêtre sur l'axe y, et charge l'image de cœur correspondante.

```
Player* initialize_player(int width, int height, int lives)
```

Cette fonction initialise un nouveau joueur (`Player`), en chargeant les images du joueur et en initialisant divers autres attributs tels que la position du joueur, le nombre de vies, et d'autres éléments liés à la mécanique du jeu.

```
void draw_frame(int pos, int pos2, int index, Player* new_player, int *duration, Enemy*** enemies_ptr, int *time_passed, int start_time)
```

Cette fonction gère le dessin du cadre du jeu et l'état du jeu. Elle charge les images d'arrière-plan, dessine le feu du joueur, met à jour les cœurs, génère des ennemis, vérifie les collisions et met à jour l'interface utilisateur.

```
void generate_enemy(int *duration, Enemy*** enemies_ptr, int *time_passed, int index, Player* new_player)
```

Cette fonction génère des ennemis en utilisant une minuterie d'ennemi (`enemy_timer`), crée des ennemis qui apparaissent (`enemy_appears`), déplace les ennemis (`move_enemy`), et fait disparaître les ennemis qui ont été touchés (`enemy_disappears`).

```
void draw_enemy(int index, Enemy* new_enemy, int up, int down, int right, int left)
```

Cette fonction dessine un ennemi (`new_enemy`) en fonction de sa position actuelle. Le paramètre `index` est utilisé pour déterminer quelle image de l

'ennemi utiliser. Les paramètres `up`, `down`, `right` et `left` indiquent la direction du mouvement de l'ennemi.

```
Enemy** enemy_appears()
```

Cette fonction génère un tableau de nouveaux ennemis, initialise leur position et leurs attributs en fonction d'un tirage aléatoire (`rand`).

Structures de Données

Ce fichier utilise plusieurs structures de données définies dans les fichiers d'en-tête inclus, comme `Player`, `Heart` et `Enemy`. Ces structures sont utilisées pour représenter les différents éléments du jeu et stocker leurs attributs, tels que leur position actuelle, leur image et d'autres propriétés spécifiques au jeu.

Fonctions

```
void move_enemy(Enemy** enemies, int* time_passed, int index)
```

Cette fonction déplace les ennemis à l'écran. Les ennemis sont passés par un tableau et déplacés en fonction du temps passé (`time_passed`). L'index est utilisé pour déterminer l'image d'ennemi à afficher. La fonction gère également le dessin de l'ennemi et le tir de la boule de feu par l'ennemi.

```
Fire* initialize_fire_enemy(Enemy* new_enemy)
```

Cette fonction initialise une nouvelle instance de `Fire` (qui représente une boule de feu tirée par un ennemi dans le jeu) en chargeant les images appropriées et en définissant la position initiale de la boule de feu en fonction de la position de l'ennemi (`new_enemy`).

```
Fire* initialize_fire_player(Player* new_player)
```

Semblable à la fonction `initialize_fire_enemy`, cette fonction initialise une nouvelle instance de `Fire` qui représente une boule de feu tirée par le joueur (`new_player`). Elle charge les images appropriées et définit la position initiale de la boule de feu en fonction de la position du joueur.

```
void draw_fire_enemy(int index, int up, int down, Enemy* new_enemy, int time_passed)
```

Cette fonction est responsable du dessin de la boule de feu tirée par un ennemi. Elle parcourt la liste des boules de feu associées à l'ennemi (`new_enemy`), dessine chaque boule de feu à l'écran, met à jour sa position et vérifie si la boule de feu est sortie de l'écran. Si c'est le cas, la boule de feu est retirée de la liste. L'ennemi tire une nouvelle boule de feu à intervalles réguliers, déterminés par un compteur de temps de tir (`time_shoot`).

Structures de Données

Ces fonctions utilisent la structure `Fire` pour représenter une boule de feu dans le jeu. Cette structure contient un tableau de positions qui stocke les images de la boule de feu, ainsi que des coordonnées x et y qui représentent la position actuelle de la boule de feu sur l'écran. Elle contient également un pointeur `next` pour pointer vers la boule de feu suivante, permettant de gérer plusieurs boules de feu dans une liste chaînée. Les structures `Player` et `Enemy` sont également utilisées, comme décrit dans la première partie de la documentation.

Fonctions

```
void draw_fire_player(int index, int up, int down, Player* new_player)
```

Cette fonction gère l'action de tirer une boule de feu par le joueur. Elle vérifie si le joueur est prêt à tirer (en fonction d'un compteur de temps de tir) et si la touche d'espace est pressée. Si c'est le cas, une nouvelle boule de feu est créée et ajoutée à la liste des tirs du joueur. Ensuite, elle parcourt la liste des boules de feu du joueur, dessine chacune d'elles, met à jour leur position et vérifie si elles sont sorties de l'écran. Si c'est le cas, elles sont supprimées de la liste.

```
Enemy* initialize_enemy(int x, int width, int height, int lives, int right, int left)
```

Cette fonction crée et initialise une nouvelle instance de `Enemy`. Elle prend en entrée les coordonnées x, la largeur et la hauteur de l'ennemi, le nombre de vies et les directions (droite et gauche). La fonction charge les images correspondantes, définit la position initiale de l'ennemi et initialise la boule de feu de l'ennemi. Elle retourne un pointeur vers l'ennemi créé.

```
void draw_UI(Player* player, int start_time)
```

Cette fonction dessine l'interface utilisateur du jeu. Elle prend en entrée un pointeur vers le joueur et le temps de départ du jeu. Elle convertit le score et le temps écoulé en chaînes de caractères, puis les dessine à l'écran. Elle dessine également la barre de vie du joueur. Enfin, elle actualise la fenêtre pour afficher les nouvelles informations.

Structures de Données

Ces fonctions utilisent les structures `Fire`, `Player` et `Enemy` décrites précédemment. `Player` et `Enemy` ont des listes de `Fire` qui représentent les boules de feu qu'ils ont tirées. `Player` a également un score et un nombre de vies qui sont utilisés pour dessiner l'interface utilisateur. `Enemy` a des positions pour stocker les images de l'ennemi, ainsi qu'un nombre de vies et une direction (droite et gauche).