**UNIVERSITE ABDERRAHMANE MIRA**

**Département informatique L1**

**Année universitaire 2022/2023**

**Rapport du Projet**

**Réalisation du jeu Pong en langage C avec la bibliothèque graphique Raylib**

**Réalisé par :**

**BOUGUERMOUH Tayeb**

**Sommaire :**

1. **Définition**
2. **Présentation de l’interface graphique**
3. **Fenêtre**
4. **Menu principal**

**2.1-Fond**

**2.2-Bouton**

1. **Table de jeu**

**3.1-Fond**

**3.2-tableau d’affichage**

**3.2.1-Mode Solo**

**3.2.2-Mode Multi**

**3.2.3-Score**

**3.3-Raquettes**

**3.4-Balle**

**3.5-Obstacles**

**4- Fin de la partie**

**4.1-Message**

**III- Explication de la partie technique du jeu**

1. **Mouvement**

**1.1- Raquette**

**1.1.1-Joueur**

**1.1.2-Bot**

**1.2- Balle**

1. **Rebond Balle**

**2.1- Bord**

**2.1.1- Bord du haut**

**2.1.2- Bord du bas**

**2.2- Raquettes**

***I- Définition :***

Pong est un jeu d’arcade classique créé en 1972 par Atari. Il est considéré comme le premier jeu vidéo a succès commercial. Le jeu consiste à utiliser des raquettes pour renvoyer une balle . Le but est de marquer des points en faisant passer la balle derrière la raquette de l’adversaire. Le jeu est joué a un contre un , avec des commandes simples pour déplacer les palettes vers le haut et vers le bas.

La définition que je viens de donner et celle du Pong original. Pour ma copie c’est en peu plus différent , bien sûr j’ai gardé le concept de base du jeu à savoir : 2 raquettes , 1 balle et le but et d’envoyer la balle derrière la raquette de l’adversaire , mais je ne me suis pas contenté que de ça .

J’ai rajouté des Obstacles au milieu de l’écran , et pour corser le tout j’ai utilisé un système de génération aléatoire ce qui veut dire que les obstacles ne seront jamais au même endroit ! Si vous avez donc de la chance vous pourrez avoir une génération qui vous sera plus au moins favorable, mais dans le cas où vous aurez les Obstacles alignés bonne chance pour faire passer la balle de l’autre côté ! Pour compenser donc cette difficulté vous aurez à votre disposition 3 Ralentisseurs qui vous serviront de joker, et comme leur nom l’indique ils permettent de ralentir la balle pour ainsi pouvoir la rattraper en cas de danger.

***II-Présentation de l’interface graphique***

**1-Fenêtre :**

Pour la création de la fenêtre j’ai utilisé les 2 fonctions suivantes :

**InitWindow() :** Cette fonction prend en entré différents paramètres tel que la hauteur et la largeur de l’écran et le titre .

**SetTargetFPS() :** Cette fonction prend en entrée seulement un entier qui permet de définir le nombre de fois par seconde que la fenêtre doit être mise a jour. 

**2-Menu Principal :**

Le menu principal comporte un fond et 3 boutons.

**2.1-fond:**

****

**Dimension :** 1200 x 800 pixel (les mêmes que celle de la fenêtre)

Pour la création du fond j’ai utilisé :

-Le type **Texture2D** pour crée une variable contenant mon image qui est charger avec la fonction **LoadTexture()** qui prend en entré le chemin d’accès au fichier.png . Remarque : dans le cas ou le fichier est dans le même dossier que le .exe vous n’avez pas besoin de préciser le chemin . Ex :

Pour l’affichage du fond j’ai utilisé la fonction :

-**DrawTexture() :** cette fonction prend en paramètre l’image charger ainsi que les coordonnées ou elle sera générer et une couleur pour effacer le fond . Ex :

 **2.2-Bouton:**

Mon menu principal comporte 3 Boutons :

**-SinglePlayer :** permet de jouer en solo contre le bot.

-**MultiPlayer :** permet de jouer en multi contre un ami.

**-Exit :** permet de fermé la fenêtre.

Après que SinglePlayer ou MultiPlayer soit enfoncé les

 3 boutons seront effacés et remplacer par les 2 Boutons :

**-Standard :** permet de jouer au mode Standard avec

seulement les ralentisseurs comme plus .

**-Custom :** permet de jouer au mode Custom qui a 3

obstacles générer automatiquement au centre de l’écran

en plus des ralentisseurs.

Pour la création des bouton j’ai Importé :  
-**Une Police :** avec le type **Font** qui permet de sauvegarder une police charger via la fonction de raylib **LoadFont() . Ex :** 

-**Des Couleurs:** avec le type **Color** qui permet de sauvegarder une couleur charger en utilisant le code RGB de la couleur. Ex : 

**-Un son :** avec le type **Sound** qui permet de sauvegarder un son charger via la fonction de raylib **loadSound() .** Ex :

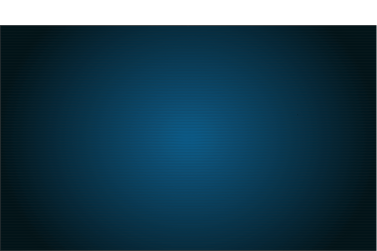
Puis pour les afficher j’ai utiliser les fonctions de dessin de la bibliotheque de raylib comme :

**DrawRectangleRec() :** qui prend en entrer un type **Rectangle** et une couleur pour le dessiner.

Ex : 

**DrawTextEx() :** qui prend en entrer une police de type **Font** , un **message ,** un type **Vector2** pour préciser l’emplacement du message a dessiner , une **Taille** pour la taille du message , un **Espace** pour préciser combien il faut laisser entre un caractère et un autre et enfin la **Couleur** du message. Ex : 

**3-Table de jeu:**

** 3.1-Fond:**

Pour importé et dessiner l’image sur le code , c’est exactement les même étape que **2.1**

**3.2-Tableau d’affichage:**

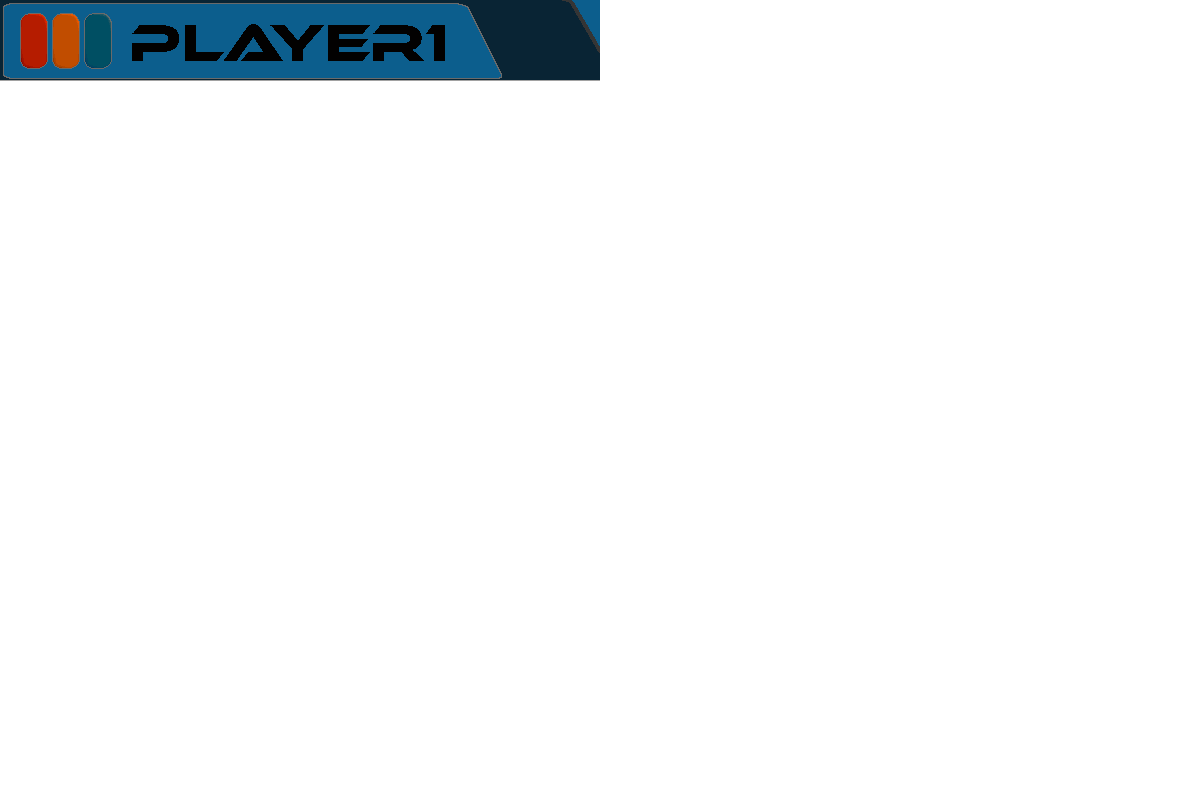
**3.2.1-Mode solo:**

Le tableau d’affichage du mode solo contiens un espace pour le score et a de base 3 ralentisseurs de balle activable avec la touche **Espace .** Pour l’importation et l’affichage c’est déjà expliquer dans **2.1 .**

****

**3.2.2-Mode Multi:**

Le tableau d’affichage du mode Multi contiens un espace pour le score et le joueur 1 et 2 ont tout 2 droit a 3 ralentisseurs au début de partie .

** **

**3.2.3-Score:**

Pour l’affichage du score j’utilise la fonction **DrawText()** de la bibliothèque raylib.

Mais vue que cette fonction ne prend en compte que les type **Char** , alors on utilise **Sprintf()** Pour convertir nos compteurs de score en chaîne de caractère .



**3.3-Raquettes:**

Pour la création des raquettes j’ai utilisé le Type **Rectangle** qui prend en paramètre d’entrée :

**x , y , Height, Width .**

Ex :

 Pour l’affichage c’est avec la fonction DrawRectangleRec() que j’ai récemment expliquer . **3.4-Balle:**

Pour la création de la balle j’ai utiliser le Type **Vector2** qui prend en paramètre d’entrer :

Les coordonnées **x** et **y .**

 D’abord, on définie la vitesse de la balle qu’on stocke dans un **Vector2 :**



Dans notre cas on a déclaré :

Ensuite, on crée un autre **Vector2** ou sera stocker la position de la balle :



Dans notre cas on a déclaré :

Enfin, on dessine en utilisant la fonction :

**-DrawCircleV() :** qui prend en entrée la **position de la balle** ainsi que son **rayon** et la **couleur** de celle ci . Ex :

Remarque : Déclarer une constante en utilisant ne prend pas de place en mémoire, contrairement au  . Ce qui explique le fait que j’ai principalement utiliser sa dans mon code.

**3.5-Obstacles :**

Pour la création des obstacles j’ai fais un système de génération aléatoire au centre de l’ecran et pour un rendu optimal j’ai crée une fonction pour mettre mes instructions dedans :

La fonction **afficherObstacles()** prend en paramètre :

-**numberObstacle** : le numéro de l'obstacle à afficher

-**positionObstacle** : une structure de type Vector2 qui contient les coordonnées de l'obstacle

-**largeurObstacle** : la largeur de l'obstacle

-**hauteurObstacle** : la hauteur de l'obstacle

-**obstacle** : un tableau de structures de type Rectangle qui représentent les obstacles

-**randomX** : une variable qui sera utilisée pour stocker la valeur aléatoire de l'abscisse de l'obstacle

-**randomY** : une variable qui sera utilisée pour stocker la valeur aléatoire de l'ordonnée de l'obstacle

**-hauteurEcran** : la hauteur de l'écran

Le but de la fonctions n’est pas seulement de générer 3 obstacles aléatoirement au centre de l’écran mais aussi de faire en sorte qu’ils n’apparaissent pas au même coordonnées . Pour donc éviter tout risque de Bug j’ai mis une distance de sécurité de 60 pixel entre chaqu’un d’eux.

**4-Fin de partie :**



A chaque fin de partie un message de victoire ou de défaite sera affiché avec 2 bouton :  
**Replay :** pour rejouer au même mode mais bien sûr avec les score réinitialiser .

**Menu :** pour revenir au menu principal du jeu où il sera possible de changer de mode ou de quitter le jeu .

***III- Explication de la partie technique du jeu***

**1-Mouvement :**

**1.1-Raquettes :**

**1.1.1-Joueur :**

Pour le mouvement de la raquette du joueur j’ai créé une fonction **Joueur1() :**  
 La fonction prend en entrée un pointeur sur un objet de type **Rectangle**, qui représente la raquette, la vitesse de déplacement de la raquette, la hauteur de l'écran et la hauteur de la raquette. L'utilisateur peut déplacer la raquette vers le haut ou vers le bas en appuyant sur les touches «Z» ou «S» du clavier. La fonction vérifie aussi si la raquette n'est pas sortie de l'écran et si tel est le cas, elle met à jour la position de la raquette en conséquence.

Remarque : C’est exactement pareil que la fonction du **Joueur2()** Sauf que la les touche sont les touche directionnels «Up» ou «Down» du clavier.

**1.1.2-Bot:**

Pour le mouvement du bot j’ai créé une fonction **Bot() :**

La fonction prend en entrée un pointeur sur un objet de type **Rectangle ,**qui représente la raquette, la vitesse de déplacement de la raquette, la hauteur de l’écran et la hauteur de la raquette.Le rôle de cette fonction est de déplacer la raquettes en fonction du y de la balle tout en faisant attention a ne pas sortir de l’écran en vérifions que le **y** de la raquette est inférieure a **HauteurEcran-HauteurRaquette** et supérieure a **HauteurOverlay .**

**1.2-Balle :**

Pour le mouvement de la balle on rajoute la **vitesse de la balle** a sa position **x** et **y .**

**2-Rebond Balle :**

**2.1-Bord :**

**2.1.1-Bord du haut :**

Pour faire en sorte que la balle rebondisse sur le bord du haut de l’écran, au moment où la position y de la balle veut dépasser 80 (qui est la hauteur de l’overlay), la vitesse de celle-ci s’inverse et donc change de direction.

**2.1.2-Bord du bas :**

Pour faire en sorte que la balle rebondisse sur le bord bas de l’écran, au moment où la **position y** de la balle + sa taille veut dépasser la hauteur maximale de la fenêtre, la vitesse de celle-ci s’inverse, et change la direction de son mouvement.

**2.2-Raquette :**

La trajectoire que va prendre la balle après le rebond dépendra de son point d’impact sur la raquette , autrement dit plus la balle se rapproche par exemple du bord du haut plus sa **vitesse Y** augmentera négativement ,et inversement si la balle se rapproche du bas sa **vitesse Y** augmentera positivement , Bien sur si la balle se retrouve entre les 2, sa **vitesse y** tendra vers 0 ou sera égale à 0, ce qui la redirigera tout droit, ou presque tout droit.

Une image contenant texte

Description générée automatiquement

Le «collisionPointRatio» est un nombre qui représente l'endroit où la balle a frappé la raquette. C'est un nombre compris entre 0 et 1, où 0 représente le bas de la raquette et 1 représente le haut.

Le code calcule le «collisionPointRatio» en divisant la différence entre la position Y de la balle et la position Y de la raquette par la hauteur de la raquette.

Ce ratio est ensuite utilisé pour déterminer la direction de la nouvelle vitesse de la balle après la collision avec la raquette. Si «collisionPointRatio» est inférieur à 0,2, la vitesse Y de la balle est définie comme étant négative, ce qui signifie que la balle se déplace vers le haut. Si «collisionPointRatio» est supérieur à 0,8, la vitesse Y de la balle est définie comme étant positive, ce qui signifie que la balle se déplace vers le bas. Si «collisionPointRatio» est compris entre 0,2 et 0,4 ou entre 0,6 et 0,8, la vitesse Y de la balle est définie en conséquence pour indiquer une direction intermédiaire. Sinon, la vitesse Y de la balle est définie comme étant 0, ce qui signifie que la balle ne se déplace pas verticalement.