جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

PONG GAME

ALGORITHMiQUE PROJECT

Présentation

Pong Game est un jeu vidéo d'arcade classique développé par Atari en 1972. C'est l'un des premiers jeux vidéo jamais créés et a été un énorme succès commercial, ce qui a aidé à populariser les jeux vidéo dans le monde entier. Dans ce Pong Game,chacun des joueurs contrôle une raquette qui se déplace verticalement sur l'un des côtés de l'écran. Le but du jeu est de renvoyer une balle vers l'adversaire en utilisant sa raquette. Si l'adversaire rate le retour de la balle, le joueur marque un point. Le premier joueur à atteindre un certain nombre de points gagne la partie. Le jeu Pong est connu pour sa simplicité, mais cela n'a pas empêché les joueurs de s'y adonner pendant des heures. Le jeu a été un véritable phénomène de société et a été joué par des millions de personnes dans les salles d'arcade du monde entier. Aujourd'hui, Pong Game est encore populaire et peut être joué sur de nombreuses plateformes, y compris sur les ordinateurs, les consoles de jeux et les téléphones portables. Bien que de nombreux autres jeux vidéo aient été développés depuis sa création, Pong Game reste un classique indémodable et continue de divertir les joueurs du monde entier.

Introduction au projet :

Le Jeu a été réalisé avec du code C et avec une bibliothèque indépendante de Code Blocks, qui se nomme Raylib celle-ci nous a fourni multitude de fonctions prédéfinies afin de faciliter la conception du jeu.

Tous les sous-programmes utilisés dans le code :

```c
void InitialiserPong(void);
void afficherPong(void);
void bougerBalle(void);
void bougerRaquette(Rectangle *pRacket, Direction pDir);
void ServirBalle(void);
void winer (int leftScore, int rightScore);
void terrainDeJeu (void);
```

Remarque : **Nous expliquons en détail tous les sous-programmes dans les pages suivantes.**

INITILISER PONG

```
void InitialiserPong (void)
{
    InitWindow (0, 0, "Pong");
    // Calculate size, position and inner limits of window.
    ecran = (Rectangle){ 0, 0, GetScreenWidth ()/1.005, GetScreenHeight ()/1.1};
    bordureDeJeu = (Rectangle){ BASE, BASE, ecran.width - (2*BASE) , ecran.height - (2*BASE)};
    haut = (Rectangle) { ecran.x, ecran.y, bordureDeJeu.width, bordureDeJeu.y};
    bas = (Rectangle) { ecran.x, bordureDeJeu.height+BASE, ecran.width, ecran.y};
    balle = (Rectangle) { 5*BASE, bordureDeJeu.height, BASE, BASE};
    Lracket = (Rectangle) { bordureDeJeu.x + BASE, bordureDeJeu.height/2, BASER, 5*BASE};
    Rracket = (Rectangle) { bordureDeJeu.width - BASE, bordureDeJeu.height/2, BASE, 5*BASE};
}
```

Descriptions

Cette première procédure est utilisée dans le but de calculer et donner des tailles à la fenêtre qui va afficher le jeu afin de rendre le rendu joli et optimal.

Il n'y a pas de données d'entrées ni de sorties.

AFFICHER PONG

```
void afficherPong(void)
{
    SetWindowPosition(ecran.width/400, ecran.height/25);
    SetWindowSize(ecran.width, ecran.height);
    SetTargetFPS(144);
}
```

Descriptions

Nous avons utilisé cette procédure comme demandé dans l'énoncé dans le but d'afficher la fenêtre qui affichera le jeu d'une manière à ce que la taille soit adaptée à chaque écran et résolution d'écran qui lancera l'application.

Il n'y a pas de données de sorties ni d'entrées.

BOUGER BALLE

```
void bougerBalle(void){
    static int xx = BASE/5; static int yy = BASE/5;
    float r = 2;
    if(CheckCollisionRecs(balle , Rracket) ||CheckCollisionRecs(balle , Lracket)){
    xx = -xx; }
    if(CheckCollisionRecs(balle , bas)||CheckCollisionRecs(balle , haut)){
    yy = -yy; }
    if (balle.x < ecran.x) {
                RScore++;
               ServirBalle();
                i =0; r = 0;   }else if (balle.x > ecran.width) {
                LScore++;
               ServirBalle();
                i =0; r = 0; }
           switch (mouvementballe){
           case 21: {
                    balle.x += r * xx;
                    balle.y += yy;
                     i = i +0.006; r = r + 2;
                   Rracket.y = balle.y + i;
                }break;
           case 22: {
                    balle.x += r * xx;
                    balle.y += yy;
                     i = i +0.003; r = r + 2;
                   Rracket.y = balle.y + i; }break;
           case 23: {
                   balle.x += r * 1.5 * xx;
                   balle.y += yy;
                    i = i +0.0001; r = r + 3;
                   Rracket.y = balle.y + i; }break;
           case 24: {
                   balle.x += r * 5 * xx;
                   balle.y += yy;
                    i = i;r = r + 5;
                   Rracket.y = balle.y + i; }break;
           case 1: {
                   balle.x += r * xx;
                   balle.y += yy; r = r + 2; }  default:break; }  }
```

Descriptions

Nous avons utilisé cette procédure afin d'éviter de répéter la manière dont devait bouger la balle a chaque mode de jeu ou difficulté choisis par l'utilisateur, donc nous avons choisis de mettre tous les cas dans une même structure et de laisser le code choisir la manière de jeu selon la valeur d'une variables extérieure à la structure qui varie en fonction du choix de l'utilisateur, également pour faciliter le code nous avons combiné les sous-programmes demandé suivant ("Deplacerballe","RebondirBalleBord", "RebondirBalleRaquette") dans celui-la . Egalement nous avons définit des limites que la balle ne peut pas dépasser lors du jeu c'est à dire les bord du haut et du bas et aussi les rackettes gauche et droite, et nous avons fait en sorte que lorsqu'elle rentre en contact avec l'une des bordures que sont angle soit modifié à 180° par rapport à celui de base comme demandé dans l'énoncé.

En ce qui concerne les difficulté et les mode de jeu nous avons eu l'idée de constituer un robot qui joue contre l'utilisateur de sorte que la raquette de ce robot suit en temps réel la position de la balle, et que la raquette se déplace de plus en plus de balle à mesure que le jeu continue, ce décalage et de plus en plus petit selon le niveau de difficulté choisis par l'utilisateur, et selon cette même difficulté la balle va aller de plus en plus vite au fur et à mesure que l'un des deux camp ne marque pas de points grâce à la variable "r" qui augmente en temps réel.

Nous proposons également de jouer en Local contre un ami de sorte que l'ami puisse contrôler la raquette de gauche.

Les donnés d'entrées sont "int mouvementballe" qui selon sa valeur va déterminer le mode et la difficulté du jeu et nous n'avons pas de donnés de sorties.

BOUGER RAQUETTE

```
void bougerRaquette(Rectangle *pRacket, Direction pDir)
{
    int step = (pDir == UP)? -BASE/2: BASE/2;

    if ((CheckCollisionRecs(haut, *pRacket) && pDir == UP) ||
        (CheckCollisionRecs(bas, *pRacket) && pDir == DOWN))
            return;
    pRacket->y += step;
}
```

Descriptions

Cette procédure va nous servir à bouger les raquette contrôlés par l'utilisateur donc la raquette de gauche en cas de jeu contre l'ordinateur et celle de droite dans le cas de jeu entre deux utilisateurs.
Pour faciliter les choses nous avons utilisé un pointeur afin de ne pas répéter la même choses pour les deux raquettes donc selon la touche du clavier utilisés la raquette répondra en bougeant dans le sens voulu.
En donnés d'entrées nous avons le pointeur de la raquette ("Rectangle *pRacket") et la direction des raquette ("Direction pDir").

SERVIR BALLE

```
void ServirBalle(void)
{
    balle.x = bordureDeJeu.width/2;
    balle.y = GetRandomValue(bordureDeJeu.y + 10, bordurkeDeJeu.height);
}
```

Descriptions

Cette procédure est simplement là pour servir la balle au milieu du terrain a chaque fois que l'un des deux côtés marque un point.
La balle va donc être redistribuer au milieu de l'axe x et dans une valeur aléatoire de l'axe y pour ne pas facilité la tâche au premier receveur vu que la balle a un mouvement de 45° elle va toujours aller au même endroit si elle est exactement au milieu. (pas de données d'entrées ni de sorties)

GAGNANT

```
void winer (int LScore, int RScore) {
if (LScore > RScore) {
        win = 1;
}else{
        win = 2; }
    return 0;   }
```

Descriptions

Cette procédure va nous servir à déterminer la gagnant entre les deux côtés à la fin de la partie avec une simple condition de base.
Les données d'entrées son ("leftScore") et ("rightScore") les scores des deux côtés, et nous n'avons pas de données de sorties.

TERRAIN DE JEU

```
void terrainDeJeu (void)
{
    // Le terrain et les bordures.
    DrawRectangle(ecran.x, ecran.y, ecran.width, ecran.height, WHITE);
    DrawRectangle(ecran.x, bordureDeJeu.y, ecran.width, bordureDeJeu.height, PURPLE);
    DrawRectangle((ecran.width/2) - 5, bordureDeJeu.y, BASE, bordureDeJeu.height, WHITE);
    // La balle.
    DrawRectangle(balle.x, balle.y, balle.width, balle.height, BLUE);
    // Les raquettes.
    DrawRectangle(Lracket.x, Lracket.y, Lracket.width, Lracket.height, WHITE);
    DrawRectangle(Rracket.x, Rracket.y, Rracket.width, Rracket.height, WHITE);
    // Le score.
    DrawText(TextFormat("%d", RScore),(ecran.width/2) +40, 50, 60, WHITE);
    DrawText(TextFormat("%d", LScore), (ecran.width/2) -60 , 50, 60, WHITE);
}
```

Descriptions

Cette procédure a été ajoutée dans le but d'optimiser le programme principale, afin de ne pas répéter deux fois cette procédure dans les deux cas ou le joueur joue contre le robot ou un autre joueur.

Le but de cette procédure est d'afficher le terrain tout entier.

Nous n'avons pas de donnés d'entrées ni de sorties.

Touches personnelles :

Ajout de Musique et Fx Sound :

```
InitAudioDevice();

    Sound fxBack = LoadSound("resources/sound.wav");
    Sound fxBounce = LoadSound("resources/bounce.wav");
    Sound fxScore = LoadSound("resources/score.wav");
    SetSoundVolume(fxScore, 0.15f);
    SetSoundVolume(fxBack, 0.08f);
    SetSoundVolume(fxBounce, 0.5f);
    PlaySoundMulti(fxBack);
```

Descriptions

Nous avons décidé de mettre une musique de fond afin de ne pas ennuyer le joueur, nous avons donc choisis une musique dynamique et retro, et nous avons aussi mis en place des Fx Sound pour rendre le jeu plus interactif par exemple avec l'effet de rebond .

Possibilité d'interagir avec la souris :

```
if(IsGestureDetected(GESTURE_TAP))
                {
                    if(GetMouseX() <= ecran.width/2+70 && GetMouseX() >=
ecran.width/2-30 && GetMouseY() <= 530 && GetMouseY() >= 450)
                    {
                        ecranaff = mode;
                    }else if(GetMouseX() <= ecran.width/2+670 &&
GetMouseX() >= ecran.width/2+600 && GetMouseY() <= bordureDeJeu.y+35 &&
GetMouseY() >= bordureDeJeu.y)
                        {
                            StopSoundMulti();
                        }
                }
```

Descriptions

Nous avons longtemps cherché une solution pour pouvoir interagir avec la souris, et nous avons finalement réussie a le faire à l'aide de deux fonctions de Raylib "IsGestureDetected()" et "GetMouseX()" ou "GetMouseY()", nous avons donc décidé de créer une condition pour voir si l'utilisateur clique sur la souris et ensuite récupérer les coordonnés de la souris a ce moment la, et donc interagir avec le jeu seulement la position de la souris sur l'écran

Analyse générale non détaillée du problème

Le problème de base qui se posait même avant de commencer a penser a l'algorithme était comment on allait faire pour passer de la programmation de base a de la programmation imagés, pour ca nous avons opté pour une bibliothèque externe nommé "Raylib" qui nous permettent d'avoir des fonctions prédéfinie pour nous simplifier la tâche.

Ensuite, nous avons réfléchis à comment nous allions faire pour créer un robot contre qui jouer pour ca nous avons trouver l'idée de faire en sorte que les coordonnés dans l'arc y de la raquettes suivent en temps réel celle de la balle et de créer un petit décalage qui augmentera au fur et à mesure que la partie ne se termine pas, et chaque niveau de difficulté a non seulement une vitesse de balle différente mais aussi un décalage différent à chaque fois afin de rendre le robot toujours plus fort.

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

Crédit

Membre du groupe :
BELKASMI Menad (A3)
BELKACEMI Cirine (A3)
BENALI Meriem (A3)
BENALI Ramy (A3)
RAHMANI Rayan (B4)

IDE Utiliser : Code Blocks

Bibliothèque utilisé : Raylib, Stdio, Stdlib

Bibliographie : https://www.raylib.com