Royaume du Maroc
Ministère de l'Enseignement Supérieur, de la Recherche Scientifique Et de l'Innovation

Université Hassan II
Ecole Normale Supérieur d'Enseignement Technique

Département Mathématiques et Informatique

Techniques de programmation

Rapport du Projet sur la programmation en Langage C:

La Revanche De Snoopy

Réalisé par :
- KOUSTA Soufiane (II-CCN)

Encadré par :
- Pr. QBADOU Mohamed

ANNÉE UNIVERSITAIRE 2023-2024

Table des matières

Introduction :

La Revanche de Snoopy est un jeu console stimulant où Snoopy doit relever le défi de récupérer quatre oiseaux situés aux coins du niveau dans un temps imparti de deux minutes par niveau. Cependant, cette quête n'est pas aussi simple qu'elle en a l'air. Snoopy doit naviguer à travers un labyrinthe truffé de pièges, avec une balle rebondissante constituant son principal obstacle.

Objectif du Jeu

Le principal objectif de Snoopy est de collecter les quatre oiseaux dispersés aux coins du niveau dans le délai imparti. Cependant, la tâche est rendue complexe par la présence d'une balle rebondissante qui parcourt constamment le niveau, obligeant le joueur à anticiper les mouvements et à planifier stratégiquement.

Défis et Obstacles

Outre la balle rebondissante, le jeu propose plusieurs autres défis pour pimenter l'expérience de jeu. Des téléporteurs peuvent être empruntés par la balle, des cases piégées peuvent causer des problèmes, et des blocs manipulables ajoutent une dimension de réflexion au jeu. Les pièges et obstacles se multiplient au fil des 120 niveaux du jeu, offrant une progression progressive de la difficulté.

La version dans ce rapport est une version plus simple du jeu :

le code sur GitHub : https://github.com/Soufiane-KS/La_Revenche_de_Snoopy-in-C.git

I. Structure du Code :

Le code de La Revanche de Snoopy est organisé de manière à assurer une lisibilité et une maintenabilité optimales. Voici une explication détaillée de sa structure :

1. Déclarations et Préprocesseur :

- **Bibliothèques** : Le code commence par inclure les bibliothèques nécessaires telles que *stdio.h*, *stdlib.h*, *windows.h*, *conio.h*, *time.h* et *stdbool.h.*

- **Constantes** : Des constantes comme HAUTEUR, LARGEUR, GAME_DURATION et NUM_COLLECTIBLES sont définies pour déterminer la taille du terrain et le nombre d'objets à collecter.

- **Variables Globales** : Les variables globales comme GameOver et Paused sont déclarées pour suivre l'état du jeu.

2. Structures de Données :

Plusieurs structures sont définies pour représenter les éléments du jeu :

- **Position :** Définit une position sur le terrain.
- **Snoopy 'S'** : Représente les caractéristiques de Snoopy.
- **Balle 'O' :** Représente les caractéristiques de la balle rebondissante.
- **Collectible 'B'** : Représente les objets que Snoopy doit collecter.

3. Terrain et Collectibles (Oiseaux) :

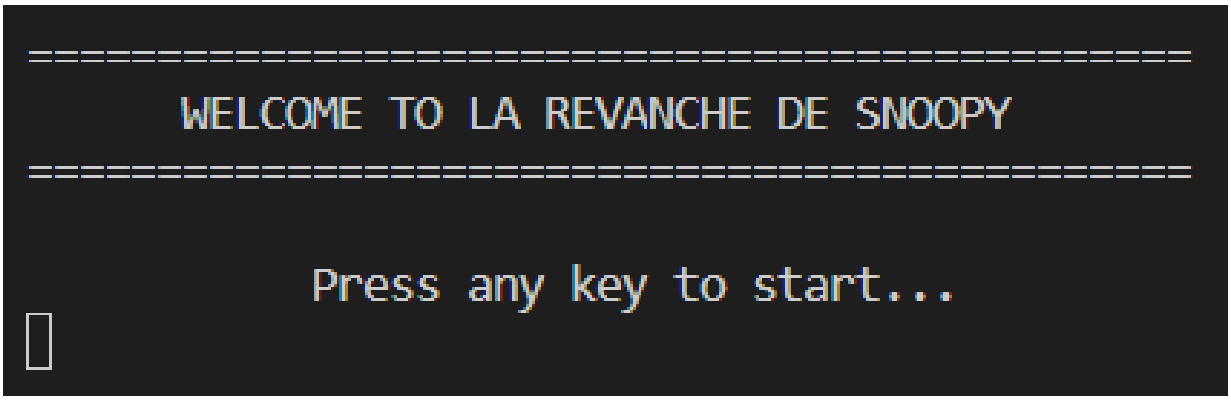
- Un tableau bidimensionnel terrain représente le niveau de jeu.
- Un tableau de structures collectibles stocke les informations sur les objets à collecter.

4. Fonctions Principales :

a. MainMenu()

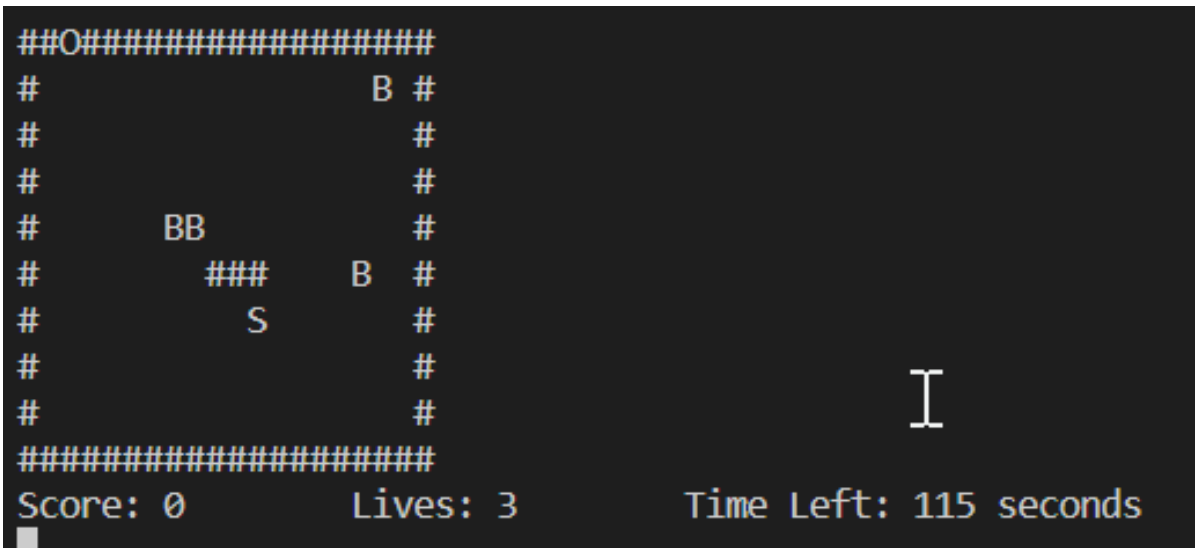
La fonction MainMenu est responsable d'afficher l'écran d'accueil du jeu. Elle imprime un message de bienvenue et attend que l'utilisateur appuie sur une touche pour commencer lejeu. Cette fonction utilise _getch() pour attendre une entrée clavier.

```
void MainMenu()
{
    system("cls"); // Clear the console in Windows

    printf("============================================\n");
    printf("      WELCOME TO LA REVANCHE DE SNOOPY        \n");
    printf("============================================\n\n");
    printf("           Press any key to start...        \n");

    _getch(); // Wait for any key press
}
```



b. Setup(Balle *balle, Snoopy *snoopy) :

La fonction Setup initialise tous les éléments nécessaires pour commencer le jeu. Cela inclut la création du terrain avec des obstacles, l'initialisation de la balle, la position de Snoopy, et la génération aléatoire des collectibles (oiseaux). Cette fonction est appelée au début du jeu.



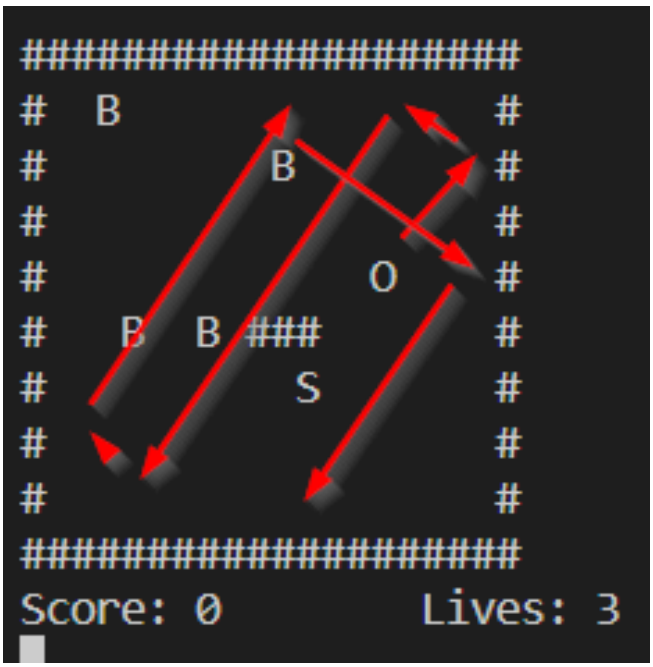
c. ClearTerrain() :

ClearTerrain remplit le tableau terrain avec des caractères représentant les murs et les obstacles. Les trois blocs horizontaux dans le milieu du terrain sont ajoutés, et les bordures sont définies avec le caractère '#'.

```
void ClearTerrain()
{
    for (int i = 0; i < HAUTEUR; i++)
    {
        for (int j = 0; j < LARGEUR; j++)
        {
            if (i == 0 || i == HAUTEUR - 1 || j == 0 || j == LARGEUR - 1)
            {
                terrain[i][j] = '#'; // Fill border with '#' character
            }
            else
            {
                terrain[i][j] = ' ';
            }
        }
    }

    // Add three blocks in the middle horizontally
    int middleRow = HAUTEUR / 2;
    int middleColumn = LARGEUR / 2;
    terrain[middleRow][middleColumn - 1] = '#';
    terrain[middleRow][middleColumn] = '#';
    terrain[middleRow][middleColumn + 1] = '#';
}
```

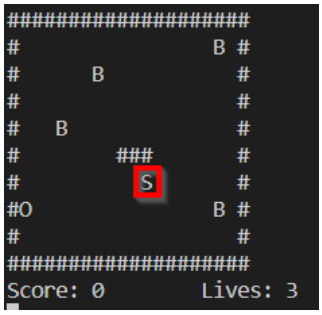
d. MoveBall(Balle *balle) :

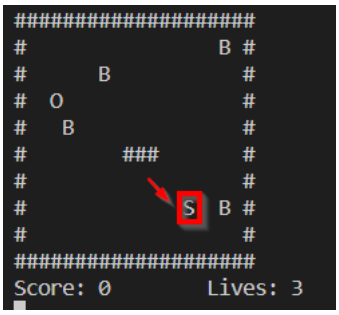
MoveBall met à jour la position de la balle en fonction de sa direction actuelle. Elle gère également les rebonds lorsque la balle atteint les bords du terrain. Cette fonction est appelée à chaque itération de la boucle principale pour animer le mouvement de la balle.



e. MoveSnoopy(Snoopy *snoopy) :

La fonction MoveSnoopy gère les mouvements de Snoopy en fonction des entrées clavier de l'utilisateur. Les touches z, s, d, et q permettent à Snoopy de se déplacer vers le haut, le bas, la droite et la gauche respectivement. La touche p met le jeu en pause.





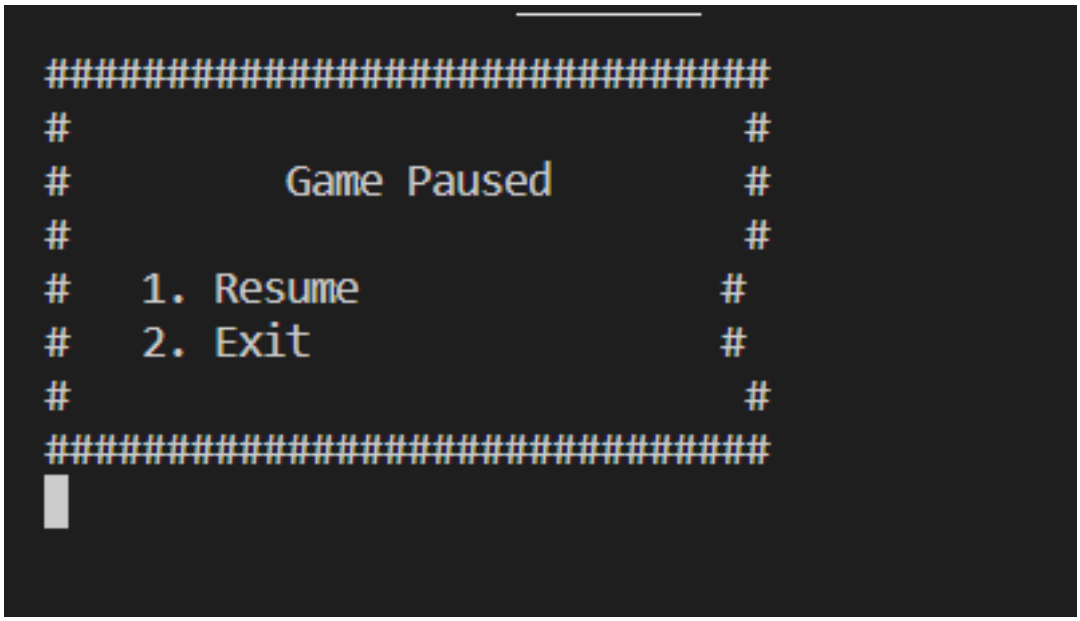
f. Draw(Balle *balle, Snoopy *snoopy , int timeLeft) :

La fonction Draw affiche le terrain, la balle, Snoopy, et les collectibles à l'écran. Elle utilise le tableau terrain pour représenter graphiquement l'état actuel du jeu.

```c
void Draw(Balle *balle, Snoopy *snoopy, int timeLeft)
{
    system("cls"); // Clear the console in Windows
    ClearTerrain();

    // Draw collectibles
    for (int i = 0; i < NUM_COLLECTIBLES; i++)
    {
        if (collectibles[i].vivant)
        {
            terrain[collectibles[i].pos.y][collectibles[i].pos.x] = collectibles[i].v;
        }
    }

    // Draw Snoopy
    terrain[snoopy->pos.y][snoopy->pos.x] = snoopy->v;

    // Draw Ball
    if (balle->vivant)
    {
        terrain[balle->pos.y][balle->pos.x] = balle->v;
    }

    // Display the entire terrain
    for (int i = 0; i < HAUTEUR; i++)
    {
        for (int j = 0; j < LARGEUR; j++)
        {
            printf("%c", terrain[i][j]);
        }
        printf("\n");
    }

    // Display Score, Lives, and Time Left
    printf("Score: %d\tLives: %d\tTime Left: %d seconds\n", snoopy->score, snoopy->vie, timeLeft);
}
```

g. ShowPauseMenu() :

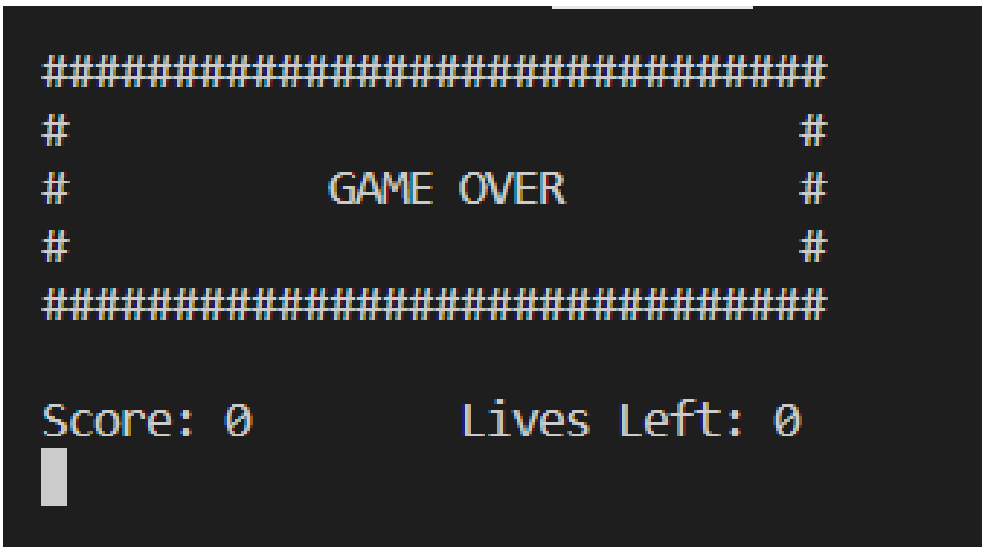
ShowPauseMenu affiche un menu de pause qui propose deux options : "1. Resume" pour reprendre le jeu et "2. Exit" pour quitter le jeu. Ce menu s'affiche lorsque le joueur appuie sur la touche p pour mettre le jeu en pause.



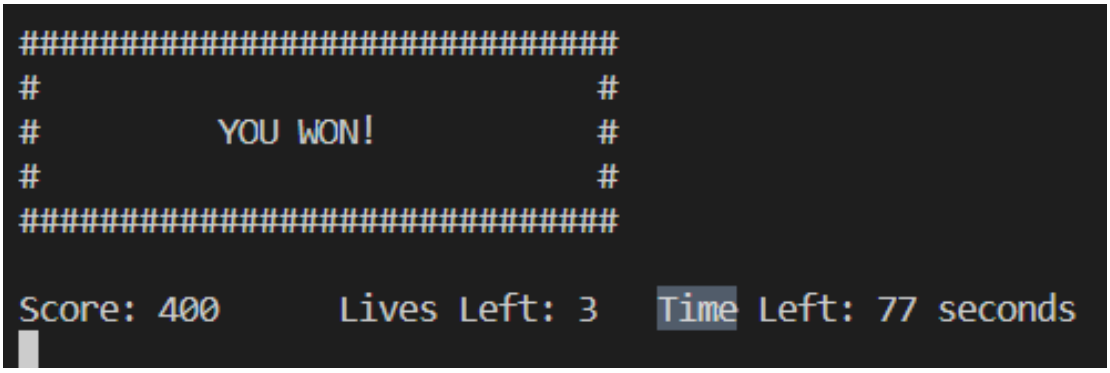
h. ShowGameOver(Snoopy *snoopy) / ShowYouWon(Snoopy *snoopy , int timeLeft) :

Ces fonctions affichent l'écran de fin de jeu en cas de défaite (ShowGameOver) ou de victoire (ShowYouWon). Elles montrent le message correspondant ainsi que le score final et le nombre de vies restantes.

GameOver :



YouWon :



i. Logic(Balle *ball, Snoopy *snoopy, time_t startTime) :

La fonction Logic est cruciale dans la gestion de la logique du jeu. Elle est responsable de plusieurs aspects tels que la collecte des objets par Snoopy, la collision entre Snoopy et la balle, et la vérification de la condition de victoire.

void Logic(Balle *ball, Snoopy *snoopy, t startTime***)***

- **Collecte des objets :**

La boucle for parcourt tous les collectibles (oiseaux) présents sur le terrain.

Si un collectible est encore vivant et que Snoopy occupe la même position que lui, cela signifie que Snoopy a réussi à le collecter.

Dans ce cas, le score de Snoopy est augmenté de 100 points, et le collectible est marqué comme non vivant (mangé).

- **Collision avec la balle :**

La fonction vérifie si la position actuelle de Snoopy (snoopy->pos) est la même que celle de la balle (ball->pos).

Si c'est le cas, cela signifie que Snoopy a été touché par la balle.

Le nombre de vies de Snoopy est décrémenté, et si ses vies atteignent zéro, le jeu est terminé en défaite. Sinon, Snoopy est replacé au centre du terrain.

- **Condition de victoire :**

La boucle for vérifie si tous les collectibles ont été collectés. Si au moins un collectible est encore vivant, la variable allCollected est mise à false.

Si tous les collectibles ont été collectés (aucun n'est vivant), la variable allCollected reste à true.

Si allCollected est true, cela signifie que Snoopy a collecté tous les objets, et le jeu est terminé en victoire.

```
if (snoopy->vie <= 0)
{
    GameOver = true;
    ShowGameOver(snoopy);
    Sleep(5000);
    ball->vivant = false;
    return;
}
else
{
    snoopy->pos.x = LARGEUR / 2 + 1;
    snoopy->pos.y = HAUTEUR / 2 + 1;
}

if (allCollected)
{
    ShowYouWon(snoopy);
    Sleep(5000);
    ball->vivant = false;
}
```

Si le nombre de vies de Snoopy atteint zéro, le jeu est terminé en défaite, et l'écran de fin de jeu est affiché.

Sinon, Snoopy est replacé au centre du terrain.

La gestion de l'écoulement du temps est dans la fonction main() .

II. Main Loop :

La fonction main est le point d'entrée du programme, déclenchant l'exécution du jeu Snoopy.

```
int main()
```

1. Déclaration des structures et des variables :

- Les structures Balle et Snoopy sont déclarées pour représenter respectivement la balle en mouvement et le personnage principal, Snoopy.
- Les variables balle et snoopy sont créées en utilisant ces structures.

```
Balle balle;
Snoopy snoopy;
```

2. Affichage du menu principal :

La fonction MainMenu est appelée, affichant un écran de bienvenue indiquant au joueur de presser n'importe quelle touche pour commencer.

```
    MainMenu();
```

3. Initialisation du jeu :

La fonction Setup est appelée pour initialiser le terrain, placer Snoopy et la balle à leurs positions de départ, et positionner les objets collectibles (oiseaux) de manière aléatoire.
Un compteur de deux minutes est initialisé pour définir la durée du niveau.

```
Setup(&balle, &snoopy);
```

4. Boucle principale du jeu :

1. La boucle while s'exécute tant que la balle est vivante et que le jeu n'est pas terminé (!GameOver).
2. À chaque itération de la boucle, les fonctions suivantes sont appelées :

- **MoveSnoopy(&snoopy)** : Gère le mouvement de Snoopy en fonction des entrées clavier du joueur.
- **MoveBall(&balle)** : Met à jour la position de la balle en fonction de sa direction actuelle.
- **Draw(&balle, &snoopy) :** Affiche le terrain, Snoopy et la balle à l'écran.
- **Logic(&balle, &snoopy)** : Gère la logique du jeu, y compris la collecte d'objets et les conditions de victoire/défaite.
- **Sleep(100) :** Met en pause l'exécution du programme pendant 100 millisecondes pour contrôler la vitesse du jeu.

```
while (balle.vivant && !GameOver)
{
    // Move the ball and draw
    MoveSnoopy(&snoopy);
    MoveBall(&balle);
    currentTime = time(NULL);
    elapsedTime = difftime(currentTime, startTime);

    if (elapsedTime >= GAME_DURATION)
    {
        GameOver = true;
        break;
    }

    Draw(&balle, &snoopy, GAME_DURATION - elapsedTime);
    Logic(&balle, &snoopy, startTime);
    Sleep(100); // Sleep for 100 milliseconds
}
```

5. Affichage du résultat du jeu :

- Si le jeu n'est pas terminé (!GameOver), la fonction ShowYouWon est appelée pour afficher l'écran de victoire.
- Sinon, la fonction ShowGameOver est appelée pour afficher l'écran de défaite.

```
if (!GameOver)
{
    ShowYouWon(&snoopy, GAME_DURATION - elapsedTime);
    exit(0);
}
else
{
    ShowGameOver(&snoopy);
}
```

Conclusion :

Cette implémentation de Snoopy met en évidence les éléments clés du jeu, la structure du code, et les fonctionnalités offertes. Malgré sa simplicité, le jeu offre une expérience engageante où le joueur doit naviguer à travers le terrain pour collecter des objets tout en évitant les obstacles, représentés ici par la balle rebondissante.

La structure du code est clairement organisée, avec des déclarations de structures en début de fichier, suivies de fonctions dédiées à différentes tâches, facilitant la maintenance et la compréhension du code. Les interactions clavier, la logique du jeu et les éléments visuels sont bien séparés, ce qui rend le code modulaire.

Le code offre également des fonctionnalités de pause (p pour Pause) et gère la fin du jeu avec des écrans distincts pour la victoire et la défaite. Les commentaires judicieux dans le code facilitent la compréhension des différentes parties.

Il est à noter que pour une expérience utilisateur optimale, des améliorations telles que des graphismes plus élaborés, une interface utilisateur améliorée, et des niveaux de jeu plus complexes pourraient être envisagées.