Elementos fundamentales de los lenguajes de programación.

Alumno: Antonio Ramos Gonzalez

Matricula: 372576

Maestro: Carlos Gallegos

Materia: Paradigmas de la programacion

Fecha de entrega: 30 de mayo de 2024

SNAKE CON LISTAS ENLAZADAS Y RAYLIB

INTRODUCCION:

Existen una gran catidad de elementos que cunplen una funcion muy importante dentro de los lenguajes de programacion, facilitandonos el desarrollo de nuestros proyectos o programas

Para esta actividad se nos encomendo identificar dichos elementos fundamentales e implementarlos en la realizacion del mitico juego Snake utilizando listas enlazadas.

Desarrollo:

Para el desarrollo de esta acitividad primeramente se hara uso del editor de codigo Visual Studio Code y la interfaz grafica raylib para C/C++. El programa cuenta con el uso de listas enlazadas para el crecimiento de la serpiente y estara codificado siguienlo las reglas de estandar en C de Michael Barr. En caso de querer descargar el juego al final del documento se presenta el link de descarga.

1.- Primemente en un archivo .h se definiran dos structs para la creacion de la serpiente. El primero llamado Tbody guardara la posicion del nodo actual y tendra un apuntador al siguinete nodo, mientras que el Snake

tendermos un tipo de dato head que sera la cabeza y un contador de la puntuacion

```c
//Incluimos bibliotecas "stdlib y raylib"
#include <stdlib.h>
#include "raylib.h"

//Definicion dato Tbody
typedef struct _body
{
    Vector2 pos;
    struct _body *next;
} Tbody;

//Definicion de cabeza y puntaje
typedef struct
{
    Tbody *head;
    int points;
} Snake;
```

2.- Ahora se definieron las funciones principales de nuestra lista o serpiente, siendo create_body la funcion para la creacion de un nuebo elemnto del cuerpo guardando su posicion actual, add_body añadiendo el nuevo cuerpo al cuerpo principal, free_body liberando elemntos individuales del cuerpo, init_snake la cual inicializa a la serpiente con una posicion inicial y el contador en 0 y por ultimo free_snake que se encarga de recorrer el cuerpo entero de nuestra serpiente para ir liberando cada elemento del mismo

```c
//Creacion de nuevo cuerpo
Tbody *create_body(Vector2 position)
{
    // Reservar memoria para el nuevo nodo
    Tbody *body = (Tbody *)malloc(sizeof(Tbody));
    // Verificar si se pudo reservar memoria correctamente
    if (body != NULL)
    {
        //Asignamos posicion a nuevo nodo
        body->pos = position;
        body->next = NULL;
    }
    return body;
}
```

```c
//Aniade nuevo cuerpo a serpiente
void add_body(Snake *snake, Vector2 position)
{
    // Crear un nuevo nodo para el cuerpo
    Tbody *newNode = create_body(position);

    // Verificar si se pudo crear el nuevo nodo
    if (newNode == NULL)
    {
        // Manejar el error de memoria insuficiente
        return;
    }

    // Buscar el último nodo en la serpiente
    Tbody *currentNode = snake->head;
    while (currentNode->next != NULL)
    {
        currentNode = currentNode->next;
    }

    // Asignar el nuevo nodo al final de la serpiente
    currentNode->next = newNode;
}

//Libera memoria de cuerpo
void free_body(Tbody *node)
{

    //Libera cuerpo
    free(node);
}
```

```c
//Creacion de cabeza de la serpiente
Snake *init_snake(Vector2 position)
{

    // Reservar memoria para la cabeza
    Snake *snake = (Snake *)malloc(sizeof(Snake));
    //Verifica que se haya reservado la memoria correctamente
    if (snake != NULL)
    {
        //Crea la cabeza de la serpiente
        snake->head = create_body(position);
        //Inicializa los puntos en 0
        snake->points = 0;
    }
    return snake;
}

//Libera memoria de la serpiente
void free_snake(Snake *snake)
{
    //Guarda direccion de la serpiente en nuevo nodo
    Tbody *current = snake->head;
    //Verifica que el nodo actual no sea el ultimo
    while (current != NULL)
    {
        //Guerdamos la poscicion actual en next
        Tbody *next = current->next;
        //Liberamos el nodo
        free_body(current);

        //Mover al siguiente nodo
        current = next;
    }
    //Liberamos memoria de la serpiente
    free(snake);
}
```

3.- En el archivo C principal llamdo game.c es donde se funcionara el juego haciendo uso de nuestra libreria Snake.h declarada anteriormente. Tambien definiremos los parametros de nuestro juego, como el numero de columnas y filas, el tamaño en pixeles de las celdas de la matriz, la separacion entre celdas, la velocidad de la

serpiente y la resolucion de la ventana a abrir

```
#include "Snake.h" //Incluimos biblioteca de snake


#define MAX_COLUMNS 41 //Nuemero de columnas
#define MAX_ROWS 20 //Numero de filas
#define CELL_SIZE 40 //Tamanio de las celdas de la matriz
#define PADDING 5 //Separacion entre celdas
#define MOVE_SPEED 0.5f //Velocidad de la serpiente
#define WIDTH 1920 //Ancho pantalla
#define HEIGHT 1040//Largo pantalla
```

4.- Ahora declaramos las funciones para el desarrollo del programa

```
void DrawBody(int startX, int startY, Tbody *head); //Dibujar cuerpo  de serpiente
void DrawMatrix(int posx, int posy, int Matrix[MAX_ROWS][MAX_COLUMNS]);//Dibujar matriz de juego
void MoveSnake(Vector2 *pos, Tbody *head, int Matrix[MAX_ROWS][MAX_COLUMNS],
               int keyPressed, bool *CloseGame); //Mover a serpiente
void BackRec();//Pintar fondo de menu
void GameOver(int points);//Pantalla de fin del juego
void Game();//Funcion principal de inicio de juego
int Detected(int KeyPresed);//Detecta la tecla presionada
```

5.- La funcion main sera la encargada de lo basico de nuestro juego , como inicializacion de la ventana del programa, presentacion del menu de juego y creditos

```c
int main(void)
    InitWindow(WIDTH, HEIGHT, "Snake");

    // Bucle principal del juego
    while (!CloseGame && !WindowShouldClose())
    {
        BeginDrawing();
        ClearBackground(LIME); // Limpiar el fondo
        Mouse = GetMousePosition(); // Obtener la posición del ratón
        back_rec(); // Dibujar el fondo

        // Dibujar el título y los botones "Start" y "Close"
        DrawText("SNAKE", (WIDTH / 2) - 325, 150, 200, font);
        DrawRectangleRec(Start, BLUE);
        DrawRectangleRec(Close, RED);

        // Verificar si el ratón está sobre los botones "Start" y "Close"
        MouseStartGame = CheckCollisionPointRec(Mouse, Start);
        MouseCloseGame = CheckCollisionPointRec(Mouse, Close);

        // Resaltar el botón "Start" si el ratón está sobre él
        if (MouseStartGame)
        {
            DrawRectangleLines(Start.x, Start.y, Start.width, Start.height, BLACK);
            DrawRectangleRec(Start, DARKBLUE);
        }

        // Resaltar el botón "Close" si el ratón está sobre él
        if (MouseCloseGame)
        {
            DrawRectangleLines(Close.x, Close.y, Close.width, Close.height, BLACK);
            DrawRectangleRec(Close, DARKRED);
        }

        // Dibujar los textos "Start" y "Close"
        DrawText("Start", Start.x + 125, Start.y + 30, 100, font);
        DrawText("Close", Close.x + 125, Close.y + 30, 100, font);

        // Verificar si se hace clic en los botones
        if (IsMouseButtonPressed(MOUSE_LEFT_BUTTON))
        {
            if (MouseStartGame)
                game(); // Iniciar el juego si se hace clic en "Start"
            if (MouseCloseGame)
                CloseGame = true; // Cerrar el juego si se hace clic en "Close"
        }

        // Dibujar información adicional
        DrawText("Create by: Antonio Ramos Gonzalez", 1410, 960, 20, GREEN);

        EndDrawing();
    }

    // Cerrar la ventana y salir del programa
    CloseWindow();
    return 0;
}
```

6.- La funcion back_rec dibuja los bordes del menu

```
//------------------------Funciones-----------------------------
void back_rec()
{
    Rectangle leftBorder = {0, 0, 150, HEIGHT}; // Borde izquierdo
    Rectangle rightBorder = {WIDTH - 150, 0, 150, HEIGHT}; // Borde derecho
    Rectangle topBorder = {0, 0, WIDTH, 70}; // Borde superior
    Rectangle bottomBorder = {0, HEIGHT - 120, WIDTH, 70}; // Borde inferior

    // Dibuja los bordes
    DrawRectangleRec(leftBorder, DARKGREEN);
    DrawRectangleRec(rightBorder, DARKGREEN);
    DrawRectangleRec(topBorder, DARKGREEN);
    DrawRectangleRec(bottomBorder, DARKGREEN);
}
```

7.- La funcion game es la funcion principal donde se ejecuta el juego. Dentro de ella se inicializa la serpiente, se generan las manzanas que se comera la serpiente para crecer, se dibuja la matriz, se detectan colisiones y se llaman a las demas funciones que se usan para la correcta ejecucion del codigo

```c
void game()
{
    //Declaracion de variables
    Vector2 Apple = {25, 9};
    Snake *snake = init_snake((Vector2){11, 9});
    int startX = 40;
    int startY = 80;
    int Matrix[MAX_ROWS][MAX_COLUMNS];
    int detctKeyboar = 4;
    int temp;
    bool CloseGame = false;

    //Ciclo que limpia la matriz
    for (int i = 0; i < MAX_ROWS; i++)
    {
        for (int j = 0; j < MAX_COLUMNS; j++)
        {
            Matrix[i][j] = 0;
        }
    }
    //Se indica la posicion de las manzanas y la serpiente
    Matrix[(int)Apple.y][(int)Apple.x] = 2;
    Matrix[(int)snake->head->pos.y][(int)snake->head->pos.x] = 1;
    //Inicia ciclo de juego
    while (!CloseGame && !WindowShouldClose())
    {
        //Dibujamos la matriz, el titulo y los puntos
        BeginDrawing();
        DrawText("SNAKE", (WIDTH / 2) - 100, 30, 50, BLUE);
        DrawText(TextFormat("Points: %i", snake->points), 100, 40, 30, BLUE);
        draw_matrix(startX, startY, Matrix);

        //Checamos colision de la cabeza de la serpiente con la manzana
        if (CheckCollisionRecs((Rectangle){snake->head->pos.x * (CELL_SIZE+
            PADDING) + startX, snake->head->pos.y * (CELL_SIZE + PADDING) +
            startY, CELL_SIZE, CELL_SIZE}, (Rectangle){Apple.x * (CELL_SIZE
            + PADDING) + startX, Apple.y * (CELL_SIZE + PADDING) + startY,
            CELL_SIZE, CELL_SIZE}))
        {

            //Verificamos la posicion de la serpiente
            Tbody *Nodo=snake->head;
            //Eliminamos a la manzana de la matriz
            Matrix[(int)Apple.y][(int)Apple.x] = 0;
            while(Nodo!=NULL)
            {
                //Le damos valor 1 para que las manzanas no se generen en su posicion
                Matrix[(int)Nodo->pos.y][(int)Nodo->pos.x]=1;
                Nodo=Nodo->next;
            }
            free_body(Nodo);
```

```
            //Generamos una nueva manzana en una posicion random
            do
            {
                Apple.x = GetRandomValue(1, MAX_COLUMNS - 1);
                Apple.y = GetRandomValue(1, MAX_ROWS - 1);
            } while (Matrix[(int)Apple.y][(int)Apple.x] != 0);

            //En la nueva posicion dibujamos manzana
            Matrix[(int)Apple.y][(int)Apple.x] = 2;

            //Aniadimos un nuevo cuerpo a la serpiente y sumamos puntos
            add_body(snake, (Vector2){snake->head->pos.x, snake->head->pos.y});
            snake->points++;
        }

        //Comprobamos que la serpiente se encuentre en los limites de la matriz
        if (snake->head->pos.x == -1 || snake->head->pos.x == MAX_COLUMNS ||
            snake->head->pos.y == -1 || snake->head->pos.y == MAX_ROWS)
            CloseGame = true;

        //Dibujamos cuerpo
        draw_body(startX, startY, snake->head);

        //Obtener tecla y mover a la serpiente
        int keyPresed = GetKeyPressed();
        if (keyPresed != 0)
        {
            detctKeyboar = detected(keyPresed);
            move_snake(&snake->head->pos, snake->head, Matrix, detctKeyboar,
                       &CloseGame);
        }
        move_snake(&snake->head->pos, snake->head, Matrix, detctKeyboar,
                   &CloseGame);
        //Limpiamos el fondo
        ClearBackground(BLACK);
        EndDrawing();
    }
    //Movemos a la pantalla de game over y liberamos memoria
    game_over(snake->points);
    free_snake(snake);
}
```

8.- La funcion draw_matrix se encarga de dibujar la matriz de juego y las manzanas

```
void draw_matrix(int posx, int posy, int Matrix[MAX_ROWS][MAX_COLUMNS])
{
    Color DARKRED = {192, 0, 0, 255};

    // Ciclo para recorrer filas
    for (int i = 0; i < MAX_ROWS; i++)
    {
        // Ciclo para recorrer columnas
        for (int j = 0; j < MAX_COLUMNS; j++)
        {
            // Calcula las coordenadas de la celda
            int cellX = posx + j * (CELL_SIZE + PADDING);
            int cellY = posy + i * (CELL_SIZE + PADDING);

            // Define el rectángulo de la celda
            Rectangle cellRect = {cellX, cellY, CELL_SIZE, CELL_SIZE};

            // Verifica el valor de la matriz en la celda
            switch (Matrix[i][j])
            {
                case 0: // Si es 0, dibuja con DARKGREEN
                    DrawRectangle(cellRect.x, cellRect.y, cellRect.width,
                                  cellRect.height, DARKGREEN);
                    break;
                case 2: // Si es 2, dibuja con DARKRED
                    DrawRectangle(cellRect.x, cellRect.y, cellRect.width,
                                  cellRect.height, DARKRED);
                    break;
                default: // Si no es ninguno de los anteriores, no dibuja nada
                    break;
            }
        }
    }
}
```

9.- La funcion draw_body pinta los elementos de la serpiente

```
void draw_body(int startX, int startY, Tbody *head)
{
    bool isFirst = true;
    Color bodyColor = {100, 156, 0, 255}; // Color del cuerpo
    Tbody *currentNode = head;

    // Ciclo para recorrer los nodos del cuerpo
    while (currentNode != NULL)
    {
        // Calcula las coordenadas de la celda
        int cellX = startX + currentNode->pos.x * (CELL_SIZE + PADDING);
        int cellY = startY + currentNode->pos.y * (CELL_SIZE + PADDING);

        // Define el rectángulo de la celda
        Rectangle cellRect = {cellX, cellY, CELL_SIZE, CELL_SIZE};

        // Dibuja la celda
        if (isFirst)
        { // Dibuja la primera celda en negro
            DrawRectangle(cellRect.x, cellRect.y, cellRect.width, cellRect.height,
                          BLACK);
            isFirst = false;
        }
        else
        { // Dibuja las demás celdas con el color del cuerpo
            DrawRectangle(cellRect.x, cellRect.y, cellRect.width, cellRect.height,
                          bodyColor);
        }

        // Avanza al siguiente nodo del cuerpo
        currentNode = currentNode->next;
    }
```

11.- La funcion move_snake se encarga del movimiento de la serpiente, asi como el verificar que esta no colisione con ella misma o con alguno de los muros

12.-La funcion dectected devuelve un valor dependiendo de la tecla persionada, esto con el fin de direccionar correctamente a la serpiente

```
int detected(int KeyPresed)
{
    if (KeyPresed == KEY_D)
        return 4;
    if (KeyPresed == KEY_A)
        return 3;
    if (KeyPresed == KEY_W)
        return 2;
    if (KeyPresed == KEY_S)
        return 1;
}
```

13.- La funcion game_over nos muestra un menu para regresar al menu principal, asi como nuestra puntuacion actual vs nuestro record

```c
//Menu de game_over
void game_over(int points)
{
    static int Record = 0; // Record del juego
    Rectangle BackToMenu = {685, 502, 558, 235}; // Área para volver al menú
    Vector2 MousePosition; // Posición del ratón
    Color font = {50, 60, 57, 255}; // Color del texto
    Color purp = {105, 0, 105, 255}; // Color púrpura
    bool MouseBackToMenu = false; // Indica si el ratón está sobre "Back to Menu"
    bool CloseGame = false; // Indica si se debe cerrar el juego

    // Actualiza el record si es necesario
    if (points > Record)
        Record = points;

    // Mientras no se cierre la ventana y no se cierre el juego
    while (!WindowShouldClose() && !CloseGame)
    {
        BeginDrawing();
        ClearBackground(LIME); // Limpia el fondo

        // Dibuja el fondo
        back_rec();

        // Dibuja los puntos y el record
        DrawText(TextFormat("Points: %i", points), 750, 250, 100, font);
        DrawText(TextFormat("Record: %i", Record), 720, 350, 100, font);

        // Dibuja el botón "Back to Menu"
        DrawRectangleRec(BackToMenu, DARKPURPLE);

        // Verifica si el ratón está sobre el botón "Back to Menu"
        MousePosition = GetMousePosition();
        MouseBackToMenu = CheckCollisionPointRec(MousePosition, BackToMenu);

        // Resalta el botón si el ratón está sobre él
        if (MouseBackToMenu)
        {
            DrawRectangleLines(BackToMenu.x, BackToMenu.y, BackToMenu.width,
                               BackToMenu.height, BLACK);
            DrawRectangleRec(BackToMenu, purp);
        }

        // Dibuja el texto "Game Over" y "Back to Menu"
        DrawText("Game Over", 575, 75, 150, font);
        DrawText("Back to", BackToMenu.x + 75, BackToMenu.y + 30, 100, font);
        DrawText("Menu", BackToMenu.x + 150, BackToMenu.y + 130, 100, font);

        // Verifica si se hace clic en el botón "Back to Menu"
        if (IsMouseButtonPressed(MOUSE_LEFT_BUTTON))
        {
            if (MouseBackToMenu)
                CloseGame = true; // Cierra el juego si se hace clic en el botón "Back to Menu"
        }

        EndDrawing();
    }
}
```

CONCLUSION

Esta parctica me ayudo acomprender en mayor medida el funcionamiento de los distintos elementos que son indispensables para la programacion, a si como reforzar mis conocimientos sobre el la aplicacion de listas enlazadas.

Versiones de descarga: WindowsVersion LinuxVersion

Para ejecutar la version de **Windows** basta con descomprimir el archivo, ingresar a la carpeta y ejecutar game.exe En caso de querer ejecutar la version de **Linux** primeramente tendras que instalar raylib en la

computadora, para ello puedes acceder al siguiente link para Instalar Raylib. Una vez instalado solo queda descomprimir el Snake-Linux.zip y ejecutar el game.exe