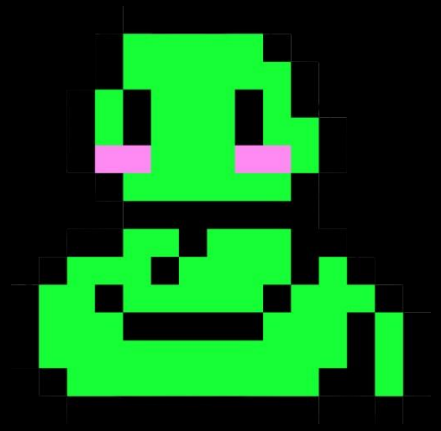
PRESENTACIÓN

SNAKE GAME



PROGAMADO EN PYTHON

LOGICA DE PROGRAMACIÓN

ANDDY LEONARDO
ZAPATA CALDERÓN

Importación de librerías

- **Turtle:** Para dibujar y controlar los elementos del juego.
- **Time:** Para manejar el tiempo y pausas en el juego.
- **Random:** Para generar posiciones aleatorias de la comida.
- **Tkinter:** Para mostrar cuadros de diálogo y mensajes emergentes.

Función para solicitar el nombre del jugador

- Muestra un cuadro de diálogo para que el usuario ingrese su nombre.
- Si no se ingresa un nombre, se usa "Jugador" por defecto.

```python
import turtle
import time
import random
import tkinter as tk
from tkinter import simpledialog, messagebox
#Definir función para pedir el nombre al jugador

def get_player_name():
    root= tk.Tk()
    root.withdraw()
    playername= simpledialog.askstring("Juagador", "¿Cual es tu nombre", parent=root)
    if not playername:
        playername= "Jugador"
    return playername

```

Configuración inicial del juego

- **Delay:** Controla la velocidad del juego.
- **Score y high_score:** Almacenan el puntaje actual y el más alto.
- **Player_name:** Guarda el nombre del jugador.

Creación de la ventana del juego

- Se crea la ventana del juego con **título, color de fondo y tamaño.**
- **win.tracer(0):** Mejora el rendimiento desactivando actualizaciones automática

Definición de los límites del área de juego

- Se definen los límites de colisión con las paredes.

Creación de la serpiente (cabeza)

- Se crea un objeto **Turtle** para la serpiente.
- Se define su **forma, color y posición inicial.**

```
delay = 0.1
score = 0
high_score = 0
player_name = get_player_name()

#Configuración de la pantalla del Juego

win=turtle.Screen()
win.title("JUEGO DE LA SERPIENTE - Turtle")
win.bgcolor("black")
win.setup(width=800, height=800)
win.tracer(0)


# Coordenadas para colisión con las paredes (ajustadas para ampliar el área de colisión)
border_x = 600  # Aumentamos el límite horizontal
border_y = 600  # Aumentamos el límite vertical

# Snake (cabeza)
snake = turtle.Turtle()
snake.speed(0)
snake.shape("circle")
snake.color("lime")
snake.penup()
snake.goto(0, 0)
snake.direction = "stop"
```

Creación de la comida

Se genera la comida con forma cuadrada y color rojo.

Posicionamiento aleatorio de la comida

La comida aparece en una posición aleatoria dentro del área de juego.

Creación del marcador de puntaje

Se usa un **Turtle** para mostrar el puntaje en pantalla

Función para actualizar el puntaje

Borra y actualiza el puntaje en pantalla cada vez que cambia.

```
# Comida
food = turtle.Turtle()
food.speed(0)
food.shape("square")  #
food.color("red")
food.penup()

# Inicializamos la comida en una posición aleatoria pero dentro del área de juego
def reset_food():
    x = random.randint(-border_x + 10, border_x - 10)
    y = random.randint(-border_y + 10, border_y - 10)
    food.goto(x, y)

reset_food()

# Cuerpo de la serpiente
body = []

# Marcador de puntaje
score_display = turtle.Turtle()
score_display.speed(0)
score_display.color("white")
score_display.penup()
score_display.hideturtle()
score_display.goto(0, 260)  # Ajustamos la posición para que no se sobreponga con la comida

# Función para actualizar el puntaje
def update_score():
    score_display.clear()
    score_display.write(f"{player_name} - Score: {score}  High Score: {high_score}", align="center", font=("Courier", 24, "normal"))
```

Funciones de movimiento de la serpiente

Mueve la serpiente en la dirección correspondiente.

Control de dirección

Evita que la serpiente se mueva en dirección opuesta, lo que evitaría colisiones instantáneas.

Asignación de controles de teclado

Se vinculan las funciones de movimiento con **las teclas de flecha.**

```python
# Funciones de movimiento
def move():
    if snake.direction == "up":
        y = snake.ycor()
        snake.sety(y + 20)
    if snake.direction == "down":
        y = snake.ycor()
        snake.sety(y - 20)
    if snake.direction == "left":
        x = snake.xcor()
        snake.setx(x - 20)
    if snake.direction == "right":
        x = snake.xcor()
        snake.setx(x + 20)

def go_up():
    if snake.direction != "down":  # Evita que se mueva hacia abajo cuando está moviéndose hacia arriba
        snake.direction = "up"

def go_down():
    if snake.direction != "up":  # Evita que se mueva hacia arriba cuando está moviéndose hacia abajo
        snake.direction = "down"

def go_left():
    if snake.direction != "right":  # Evita que se mueva hacia la derecha cuando está moviéndose hacia la izquierda
        snake.direction = "left"

def go_right():
    if snake.direction != "left":  # Evita que se mueva hacia la izquierda cuando está moviéndose hacia la derecha
        snake.direction = "right"

# Controles del teclado

win.listen()
win.onkey(go_up, "Up")
win.onkey(go_down, "Down")
win.onkey(go_left, "Left")
win.onkey(go_right, "Right")
```

Función para reiniciar el juego

- **Guarda el puntaje más alto y pregunta si el jugador quiere jugar otra vez.**

```python
# Función para reiniciar el juego
def reset_game():
    global score, delay, high_score, player_name, body
    # Si el puntaje actual es más alto que el puntaje más alto, lo actualizamos
    if score > high_score:
        high_score = score

    score = 0
    delay = 0.1
    update_score()

    # Limpiar el cuerpo de la serpiente
    for segment in body:
        segment.goto(1000, 1000)  # Mover los segmentos fuera de la pantalla
    body.clear()

    # Mover la serpiente a la posición inicial
    snake.goto(0, 0)
    snake.direction = "stop"

    # Reiniciar la comida
    reset_food()

    # Preguntar si quiere jugar otra vez
    play_again = messagebox.askyesno("Game Over", f"¡Perdiste! Tu puntaje final es: {score}\nTu puntaje más alto es: {high_score}\n¿Quieres jugar otra vez?")
    if play_again:
        # Si quiere jugar otra vez, reiniciar el juego
        start_game()
    else:
        messagebox.showinfo("Gracias por jugar", f"Gracias por jugar, {player_name}!\nTu puntaje más alto es: {high_score}")
        win.bye()  # Cierra la ventana
```

Bucle principal del juego

Maneja la lógica del juego: **movimiento, colisiones y crecimiento de la serpiente.**

```python
# Función para iniciar el juego
def start_game():
    global score, delay, body  # Agregar 'delay' como global
    score = 0
    update_score()

    # Bucle del juego
    while True:
        win.update()

        # Colisión con los bordes
        if snake.xcor() > border_x or snake.xcor() < -border_x or snake.ycor() > border_y or snake.ycor() < -border_y:
            reset_game()

        # Colisión con la comida
        if snake.distance(food) < 20:
            reset_food()  # Colocar la comida en una nueva posición

            # Agregar un nuevo segmento al cuerpo
            new_segment = turtle.Turtle()
            new_segment.speed(0)
            new_segment.shape("circle")  # Cambio a forma circular para el cuerpo
            new_segment.color("lightgreen")  # Color más suave para el cuerpo
            new_segment.penup()
            body.append(new_segment)

            # Aumentar puntaje y hacer el juego más rápido
            score += 10
            delay -= 0.002
            update_score()

        # Mover el cuerpo de la serpiente
        for i in range(len(body) - 1, 0, -1):
            x = body[i - 1].xcor()
            y = body[i - 1].ycor()
            body[i].goto(x, y)

        if len(body) > 0:
            x = snake.xcor()
            y = snake.ycor()
            body[0].goto(x, y)
```

Iniciar el juego

Llama a la función principal para iniciar el juego y permite cerrar la ventana al hacer clic

```
# Comenzamos el juego
start_game()

win.exitonclick()  # Cierra la ventana cuando se hace clic
```

GRACIAS POR SU ATENCION

LINK:AnDDy-Z-C-L/-Desarrollo-de-Software-: Desarrollo de Software