PROYECTO FINAL

Edgar Trujillo

OBJETIVO GENERAL

- Desarrollar un juego clásico (Snake) en Python

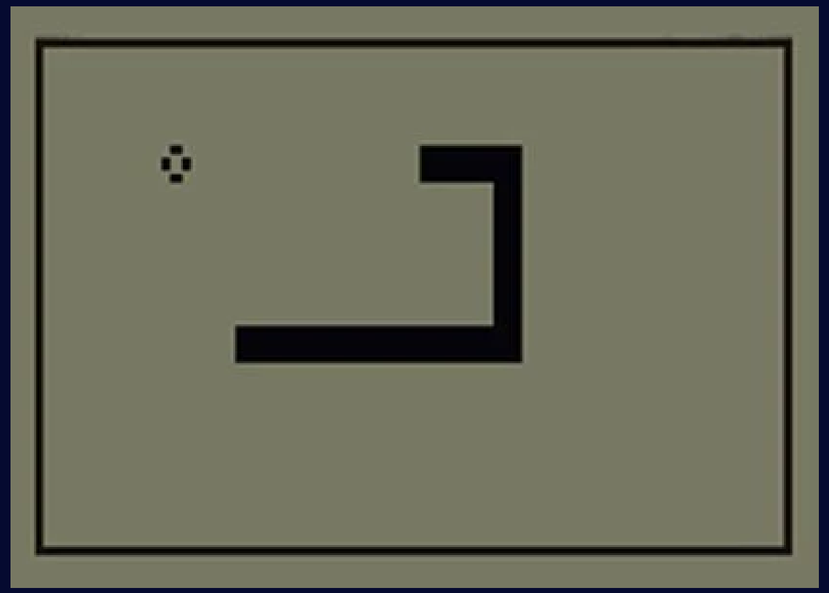
OBJETIVO ESPECIFICOS

- Practicar bucles, funciones, listas y eventos
- Usar Pygame para gráficos y manejo del teclado

CÓMO SE ORIGINO

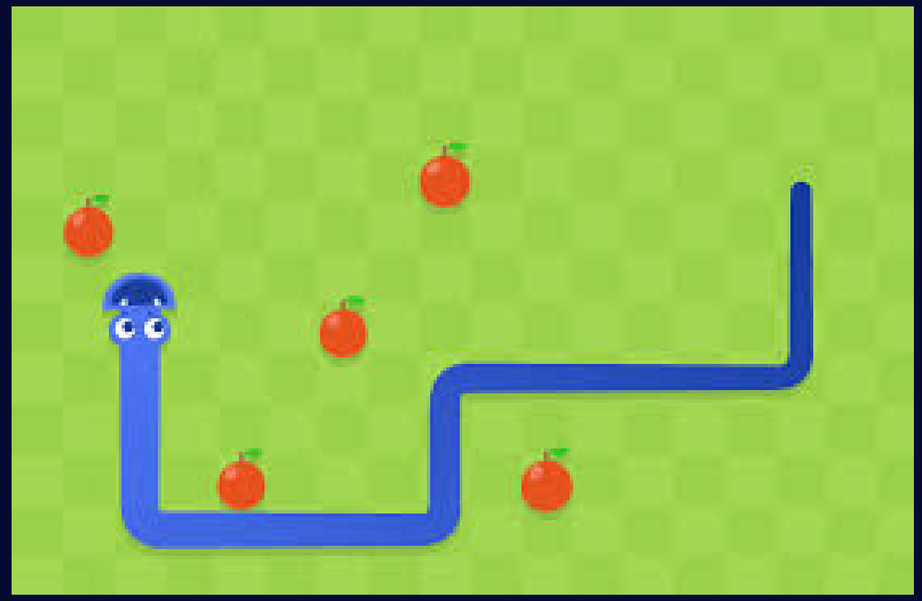
•Origen (1976):

- Diseñadores buscaban mecánicas simples para arcades limitados.
- Gremlin Industries crea Blockade: dos líneas que crecen y no deben chocar.
- Inspirado en la estrategia y anticipación de juegos de mesa.



Evolución (1997):

- Se adapta a una serpiente que come y crece.
- Nokia lo populariza: juego sencillo, adictivo y perfecto para móviles.
- Se convierte en un clásico mundial de jugabilidad simple.



DESCRIPCIÓN DEL JUEGO

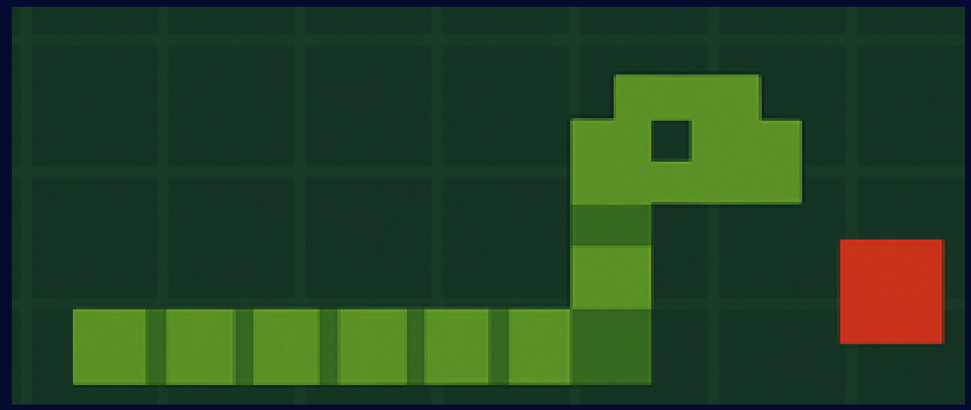
- Controlar la serpiente con las teclas ↑ ↓ ← →
- Objetivo: comer bloques verdes para crecer

El juego termina si:

- La serpiente choca con los bordes
- La serpiente se muerde a sí misma

ESTRUCTURA DEL CÓDIGO

•Menú principal: iniciar, reiniciar o salir
• Funciones principales:
• puntos(): muestra el puntaje
• nuestra_serpiente(): dibuja la serpiente
• mensaje(): muestra textos
• mensaje_perder(): pantalla de derrota
• juego(): lógica principal



Usamos librerias creadas en python que realizan acciones de juegos

```
# Declaramos las librerías
import pygame
import random
```

Declaramos el tamaño de la ventana que se visualizara el juego

```
# Dimensiones de la ventana
ancho = 600
alto = 300

# Tamaño del bloque y velocidad
bloque = 10
velocidad = 15
```

Inicializamos el menú con las opciones que van a tener los mensajes de inicio y salida

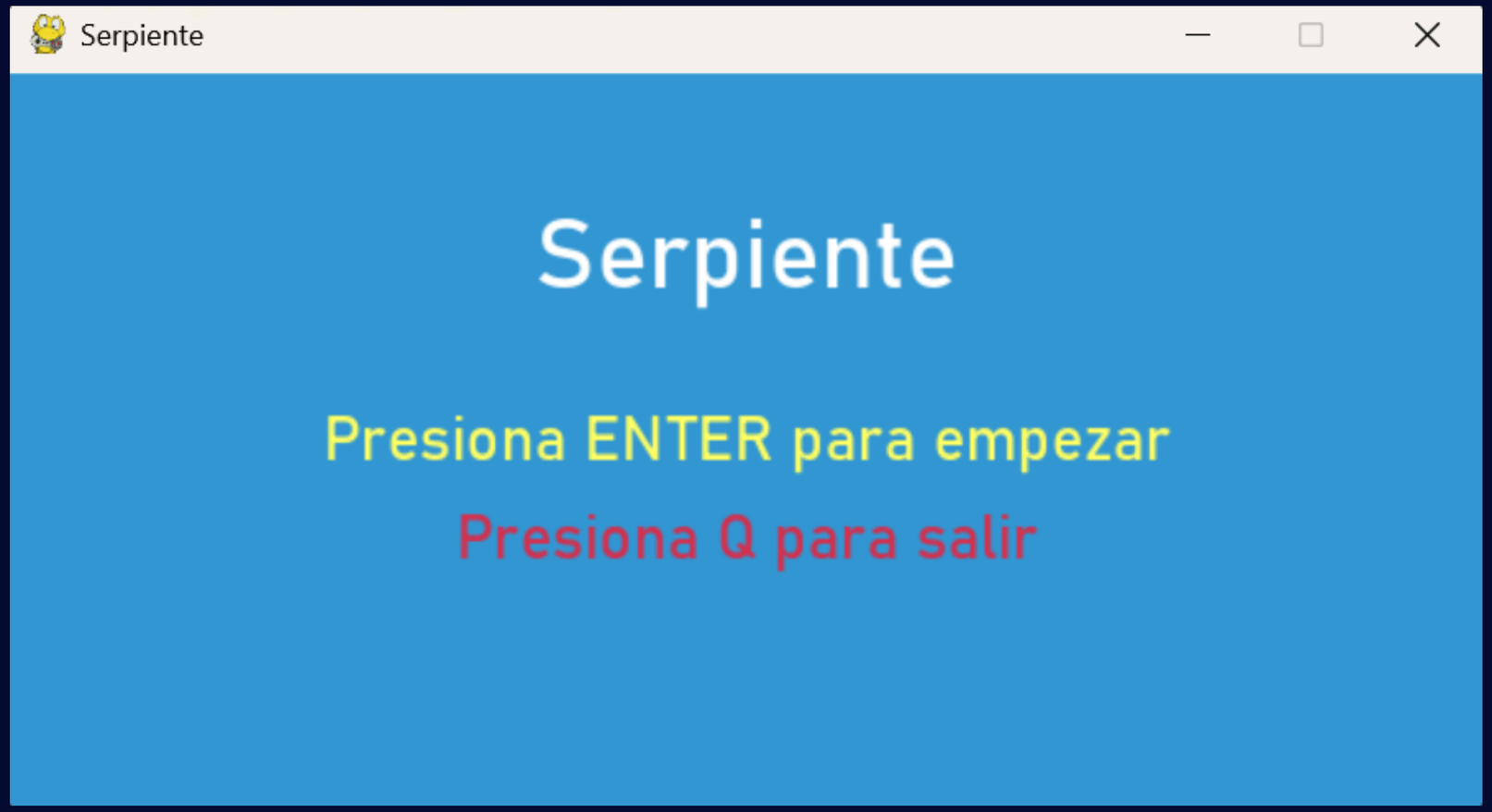
```
def menu():
    en_menu = True
    while en_menu:
        pantalla.fill(azul)

        mensaje("Serpiente", blanco, alto // 4, fuente_grande)
        mensaje("Presiona ENTER para empezar", amarillo, alto // 2)
        mensaje("Presiona Q para salir", rojo, alto // 2 + 40)

        pygame.display.update()
def mensaje_perder(puntaje):
    esperando = True
    while esperando: # Bucle para esperar la acción del usuario
        pantalla.fill(azul)
        mensaje("¡Perdiste!", rojo, alto // 2 - 60, fuente_grande)
        mensaje(f"Puntos: {puntaje}", amarillo, alto // 2 - 20)
        mensaje("R - Reiniciar", amarillo, alto // 2 + 40)
        mensaje("M - Volver al menú", amarillo, alto // 2 + 80)
        mensaje("ESC - Salir del juego", amarillo, alto // 2 + 120)
```

Asignamos eventos a cada letra propuesta que realizará la acción en el juego

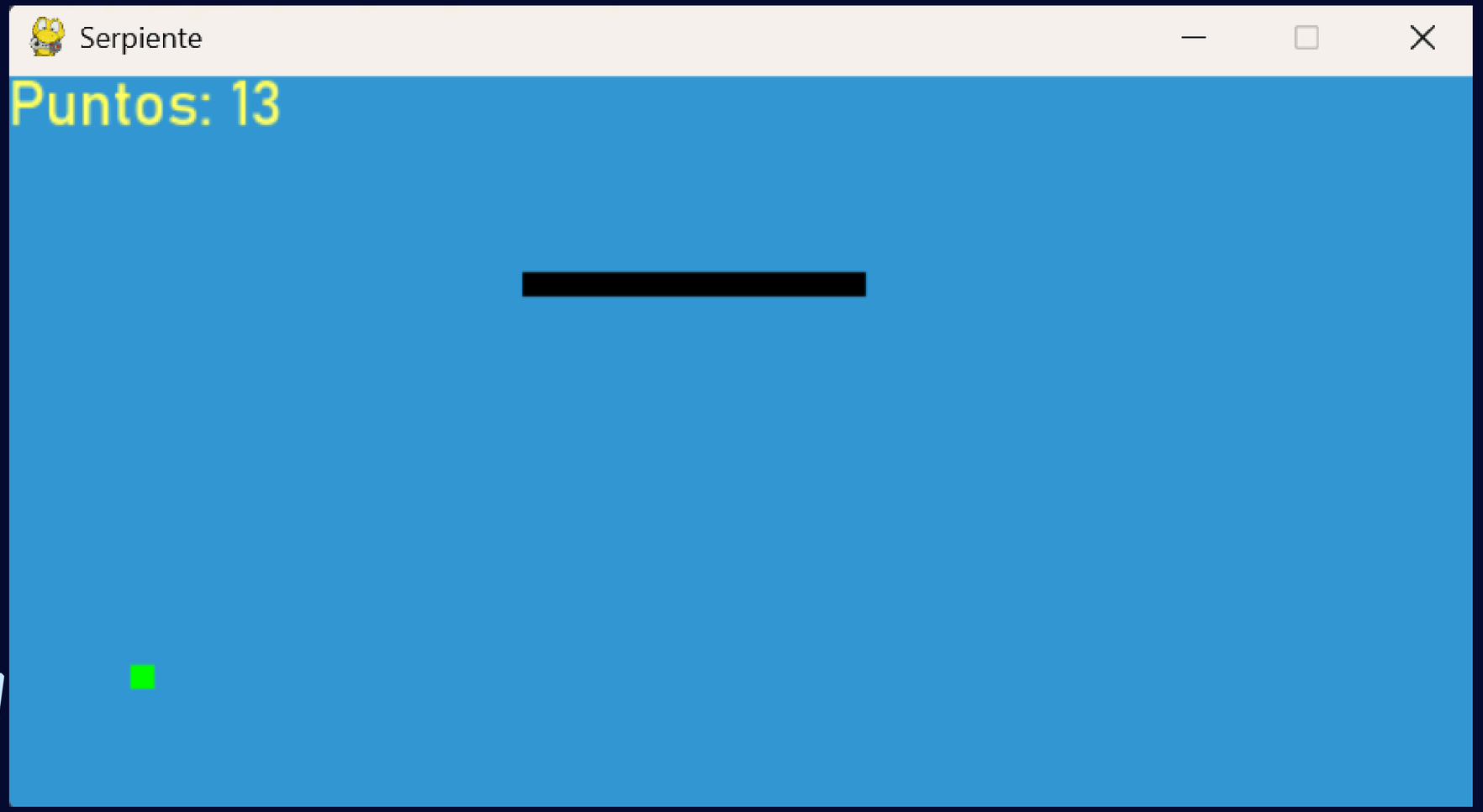
Enter: Iniciar el juego
Q: Salir

```
for evento in pygame.event.get():
    if evento.type == pygame.QUIT:
        pygame.quit()
        quit()
    if evento.type == pygame.KEYDOWN:
        if evento.key == pygame.K_RETURN:  # Enter para iniciar el juego
            en_menu = False
            juego()
        if evento.key == pygame.K_q: # Q para salir
            pygame.quit()
            quit()
```

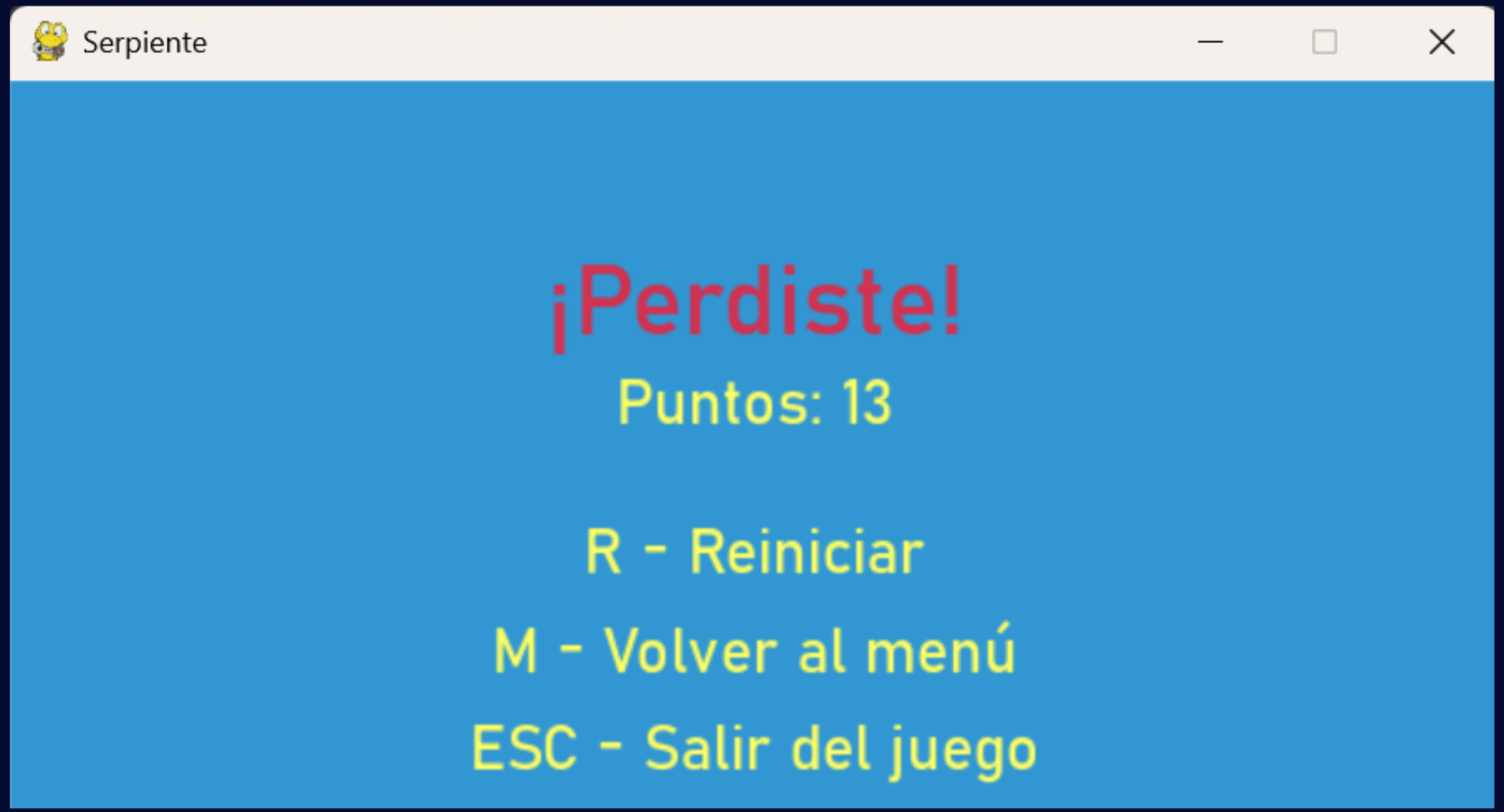
Menú de inicio para el juego



Visualización del juego



Mensaje de perdida con su menú



RESULTADOS

- Juego completamente funcional
- Incluye menú de inicio, reinicio y salida
- Se muestran puntajes y mensajes de Game Over

CONCLUSIONES Y MEJORAS

- Aplicación práctica de lógica de programación

- Posibles mejoras:
  - Aumentar dificultad con la velocidad
  - Guardar récords
  - Personalizar colores y sonidos

GRACIAS