EXAMEN FINAL OPTATIVA I

PYTHON

DOCUMENTACION DEL CÓDIGO

NOMBRE: ERIC ALBERTO OJEDA ZARZA

DOCENTE: ING. RICARDO MAIDANA

SEMESTRE: 9NO

CAACUPÉ – PARAGUAY

2024

Introducción

Este documento describe el código y las funcionalidades de un juego de Ping Pong desarrollado con Pygame. El juego incluye características como la entrada de nombres de jugadores, la selección de puntos a ganar, y un historial de partidas guardadas en un archivo JSON.

1. Introducción

2. **Concepto del juego:** Pong es un juego de tenis de mesa en 2D para dos jugadores. Cada jugador controla una paleta que se mueve verticalmente en el lado izquierdo o derecho de la pantalla. El objetivo es evitar que la pelota toque la pared detrás de tu paleta. Si la pelota toca la pared del oponente, ganas un punto. Si toca tu pared, el oponente anota un punto.
3. **¿Por qué Pygame y Python?**

   Python es un lenguaje de programación versátil que es fácil de usar incluso para principiantes. Se destaca por su comunidad activa y una amplia variedad de bibliotecas, lo que lo convierte en una excelente opción para proyectos de desarrollo. Pygame es una librería de videojuegos específica que nos permite crear mundos interactivos, personajes animados y efectos visuales cautivadores. Pygame nos permite simplemente cargar imágenes, reproducir sonidos y controlar la entrada del usuario.

   Explorar Pygame: Comprender las funciones básicas de Pygame, como crear sprites, cargar imágenes y detectar eventos del teclado. Construir habilidades de programación: Para crear nuestros propios juegos, utilizaremos nuestro conocimiento previo de programación en Python. Fomentar la creatividad: crearemos mundos virtuales distintivos y daremos vida a nuestros personajes.

3.1 Propósito del Programa

El propósito de este juego de Ping Pong es proporcionar una experiencia de entretenimiento interactiva utilizando el módulo Pygame de Python. El juego simula una partida de Ping Pong entre dos jugadores, donde el objetivo principal es lograr anotar puntos al hacer que la pelota pase por el lado del adversario.

4. Código Fuente

4.1 Importación de Módulos

```
python
import pygame
import sys
import json
```

4.2 Definición de Constantes

```
python
ANCHO = 800
ALTO = 600
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)
```

5. Clases

5.1 Clase Paleta

```
python
class Paleta(pygame.sprite.Sprite):
def __init__(self, color, x, y):
super().__init__()
self.image = pygame.Surface([10, 100])
self.image.fill(color)
self.rect = self.image.get_rect()
self.rect.x = x
self.rect.y = y
def mover(self, dy):
self.rect.y += dy
if self.rect.y < 0:
self.rect.y = 0
```

```
if self.rect.y > ALTO - self.rect.height:
self.rect.y = ALTO - self.rect.height
```

5.2 Clase Pelota

```
python
class Pelota(pygame.sprite.Sprite):
    def __init__(self, color, x, y):
        super().__init__()
        self.image = pygame.Surface([10, 10])
        self.image.fill(color)
        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y
        self.vel_x = 6
        self.vel_y = 6

    def update(self):
        self.rect.x += self.vel_x
        self.rect.y += self.vel_y

        if self.rect.y > ALTO - self.rect.height or self.rect.y < 0:
            self.vel_y = -self.vel_y
```

6. Funciones

6.1 Función mostrar_menu_principal

```
python
def mostrar_menu_principal(pantalla, clock):
    seleccionado = 0
    opciones = ["Jugar", "Ver Historial", "Salir"]

    while True:
        for evento in pygame.event.get():
            if evento.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            elif evento.type == pygame.KEYDOWN:
                if evento.key == pygame.K_UP:
                    seleccionado = (seleccionado - 1) % len(opciones)
                elif evento.key == pygame.K_DOWN:
                    seleccionado = (seleccionado + 1) % len(opciones)
                elif evento.key == pygame.K_RETURN:
                    if seleccionado == 0:
                        nombres_jugadores = ingresar_nombres(pantalla,
clock)
                        puntos_ganar = seleccionar_puntos(pantalla,
clock)
                        iniciar_juego(pantalla, nombres_jugadores,
puntos_ganar, clock)
                    elif seleccionado == 1:
                        ver_historial(pantalla, clock)
                    elif seleccionado == 2:
                        pygame.quit()
                        sys.exit()

        pantalla.fill(NEGRO)
        titulo = pygame.font.SysFont('consolas', 60).render("Ping Pong",
True, BLANCO)
        pantalla.blit(titulo, (ANCHO // 2 - titulo.get_width() // 2,
100))

        for i, opcion in enumerate(opciones):
            texto = pygame.font.SysFont('consolas', 30).render(opcion,
True, BLANCO)
            if i == seleccionado:
```

```
                pygame.draw.rect(pantalla, BLANCO, [ANCHO // 2 - 100,
300 + i * 50, 200, 40], 3)
            pantalla.blit(texto, (ANCHO // 2 - texto.get_width() // 2,
300 + i * 50))

        pygame.display.flip()
        clock.tick(30)
```

6.2 Función ingresar_nombres

```
python
def ingresar_nombres(pantalla, clock):
    jugador1_nombre = ""
    jugador2_nombre = ""
    entrada_jugador = 0

    while entrada_jugador < 2:
        for evento in pygame.event.get():
            if evento.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            elif evento.type == pygame.KEYDOWN:
                if evento.key == pygame.K_BACKSPACE:
                    if entrada_jugador == 0:
                        jugador1_nombre = jugador1_nombre[:-1]
                    elif entrada_jugador == 1:
                        jugador2_nombre = jugador2_nombre[:-1]
                elif evento.key == pygame.K_RETURN:
                    entrada_jugador += 1
                else:
                    if entrada_jugador == 0:
                        jugador1_nombre += evento.unicode
                    elif entrada_jugador == 1:
                        jugador2_nombre += evento.unicode

        pantalla.fill(NEGRO)
        if entrada_jugador == 0:
            texto = pygame.font.SysFont('consolas', 40).render(f"Jugador
1: {jugador1_nombre}", True, BLANCO)
        elif entrada_jugador == 1:
            texto = pygame.font.SysFont('consolas', 40).render(f"Jugador
2: {jugador2_nombre}", True, BLANCO)
        pantalla.blit(texto, (ANCHO // 2 - texto.get_width() // 2, ALTO
// 2))

        pygame.display.flip()
        clock.tick(30)

    return jugador1_nombre, jugador2_nombre
```

6.3 Función seleccionar_puntos

```
python
def seleccionar_puntos(pantalla, clock):
    puntos_ganar = 5
    confirmado = False

    while not confirmado:
        for evento in pygame.event.get():
            if evento.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            elif evento.type == pygame.KEYDOWN:
                if evento.key == pygame.K_UP:
                    puntos_ganar += 1
                elif evento.key == pygame.K_DOWN:
                    puntos_ganar -= 1
                elif evento.key == pygame.K_RETURN:
```

```
                    confirmado = True

        pantalla.fill(NEGRO)
        titulo = pygame.font.SysFont('consolas', 30).render("Puntos a
Ganar: {}".format(puntos_ganar), True, BLANCO)
        pantalla.blit(titulo, (ANCHO // 2 - titulo.get_width() // 2,
ALTO // 2))

        pygame.display.flip()
        clock.tick(30)

    return puntos_ganar
```

6.4 Función ver_historial

```
python
def ver_historial(pantalla, clock):
    pantalla.fill(NEGRO)
    titulo = pygame.font.SysFont('consolas', 40).render("Historial de
Partidas", True, BLANCO)
    volver = pygame.font.SysFont('consolas', 20).render("Presione
cualquier tecla para volver al menú principal", True, BLANCO)

    historial = []
    try:
        with open('historial.json', 'r') as file:
            historial = json.load(file)
    except FileNotFoundError:
        pass

    y_pos = ALTO // 4
    for partida in historial:
        texto_partida = f"{partida['jugador1']} vs
{partida['jugador2']}: {partida['resultado']}"
        texto_render = pygame.font.SysFont('consolas',
20).render(texto_partida, True, BLANCO)
        pantalla.blit(texto_render, (ANCHO // 2 -
texto_render.get_width() // 2, y_pos))
        y_pos += 50

    pantalla.blit(titulo, (ANCHO // 2 - titulo.get_width() // 2, ALTO //
8))
    pantalla.blit(volver, (ANCHO // 2 - volver.get_width() // 2, ALTO -
ALTO // 8))

    pygame.display.flip()

    while True:
        for evento in pygame.event.get():
            if evento.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            elif evento.type == pygame.KEYDOWN:
                return

        clock.tick(30)
```

6.5 Función iniciar_juego

```
python
def iniciar_juego(pantalla, nombres_jugadores, puntos_ganar, clock):
    jugador1_nombre, jugador2_nombre = nombres_jugadores

    jugador1 = Paleta(BLANCO, 20, ALTO // 2 - 50)
    jugador2 = Paleta(BLANCO, ANCHO - 30, ALTO // 2 - 50)
    pelota = Pelota(BLANCO, ANCHO // 2, ALTO // 2)

    todos_los_sprites = pygame.sprite.Group()
```

```
    todos_los_sprites.add(jugador1, jugador2, pelota)

    puntuacion_jugador1 = 0
    puntuacion_jugador2 = 0

    while True:
        for evento in pygame.event.get():
            if evento.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            elif evento.type == pygame.KEYDOWN:
                if evento.key == pygame.K_w:
                    jugador1.mover(-10)
                elif evento.key == pygame.K_s:
                    jugador1.mover(10)
                elif evento.key == pygame.K_UP:
                    jugador2.mover(-10)
                elif evento.key == pygame.K_DOWN:
                    jugador2.mover(10)

        pelota.update()

        if pygame.sprite.collide_rect(pelota, jugador1) or
pygame.sprite.collide_rect(pelota, jugador2):
            pelota.vel_x = -pelota.vel_x

        if pelota.rect.x > ANCHO:
            puntuacion_jugador1 += 1
            pelota.rect.x = ANCHO // 2
            pelota.rect.y = ALTO // 2
        elif pelota.rect.x < 0:
            puntuacion_jugador2 += 1
            pelota.rect.x = ANCHO // 2
            pelota.rect.y = ALTO // 2

        if puntuacion_jugador1 >= puntos_ganar:
            resultado = f"{jugador1_nombre} gana!"
            guardar_partida(jugador1_nombre, jugador2_nombre, resultado)
            mostrar_resultado(pantalla, resultado, clock)
            return
        elif puntuacion_jugador2 >= puntos_ganar:
            resultado = f"{jugador2_nombre} gana!"
            guardar_partida(jugador1_nombre, jugador2_nombre, resultado)
            mostrar_resultado(pantalla, resultado, clock)
            return

        pantalla.fill(NEGRO)
        pygame.draw.line(pantalla, BLANCO, [ANCHO // 2, 0], [ANCHO // 2,
ALTO], 5)
        todos_los_sprites.draw(pantalla)

        texto_puntuacion = pygame.font.SysFont('consolas',
40).render(f"{puntuacion_jugador1} - {puntuacion_jugador2}", True,
BLANCO)
        pantalla.blit(texto_puntuacion, (ANCHO // 2 -
texto_puntuacion.get_width() // 2, 10))

        pygame.display.flip()
        clock.tick(30)
```

6.6 Función guardar_partida

```
python
def guardar_partida(jugador1_nombre, jugador2_nombre, resultado):
    partida = {
        'jugador1': jugador1_nombre,
        'jugador2': jugador2_nombre,
```

```
        'resultado': resultado
    }

    try:
        with open('historial.json', 'r') as file:
            historial = json.load(file)
    except FileNotFoundError:
        historial = []

    historial.append(partida)
    with open('historial.json', 'w') as file:
        json.dump(historial, file, indent=4)
```

6.7 Función mostrar_resultado

```
python
def mostrar_resultado(pantalla, resultado, clock):
    pantalla.fill(NEGRO)
    texto_resultado = pygame.font.SysFont('consolas',
40).render(resultado, True, BLANCO)
    volver = pygame.font.SysFont('consolas', 20).render("Presione
cualquier tecla para volver al menú principal", True, BLANCO)

    pantalla.blit(texto_resultado, (ANCHO // 2 -
texto_resultado.get_width() // 2, ALTO // 2))
    pantalla.blit(volver, (ANCHO // 2 - volver.get_width() // 2, ALTO -
ALTO // 8))

    pygame.display.flip()

    while True:
        for evento in pygame.event.get():
            if evento.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            elif evento.type == pygame.KEYDOWN:
                return

        clock.tick(30)
```

6.8 Función Principal main

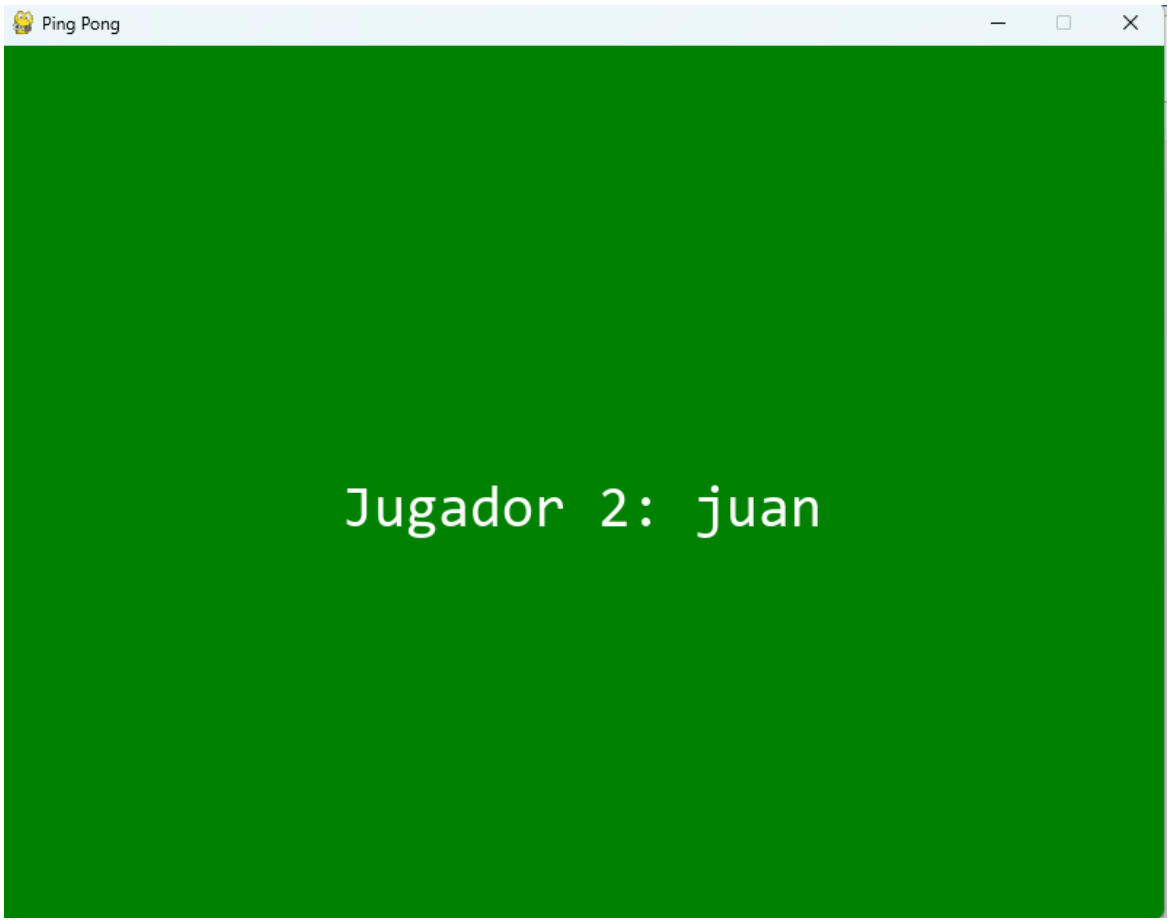
```
python
def main():
    pygame.init()
    pantalla = pygame.display.set_mode((ANCHO, ALTO))
    pygame.display.set_caption("Ping Pong")
    clock = pygame.time.Clock()

    while True:
        mostrar_menu_principal(pantalla, clock)

if __name__ == '__main__':
    main()
```

Conclusión

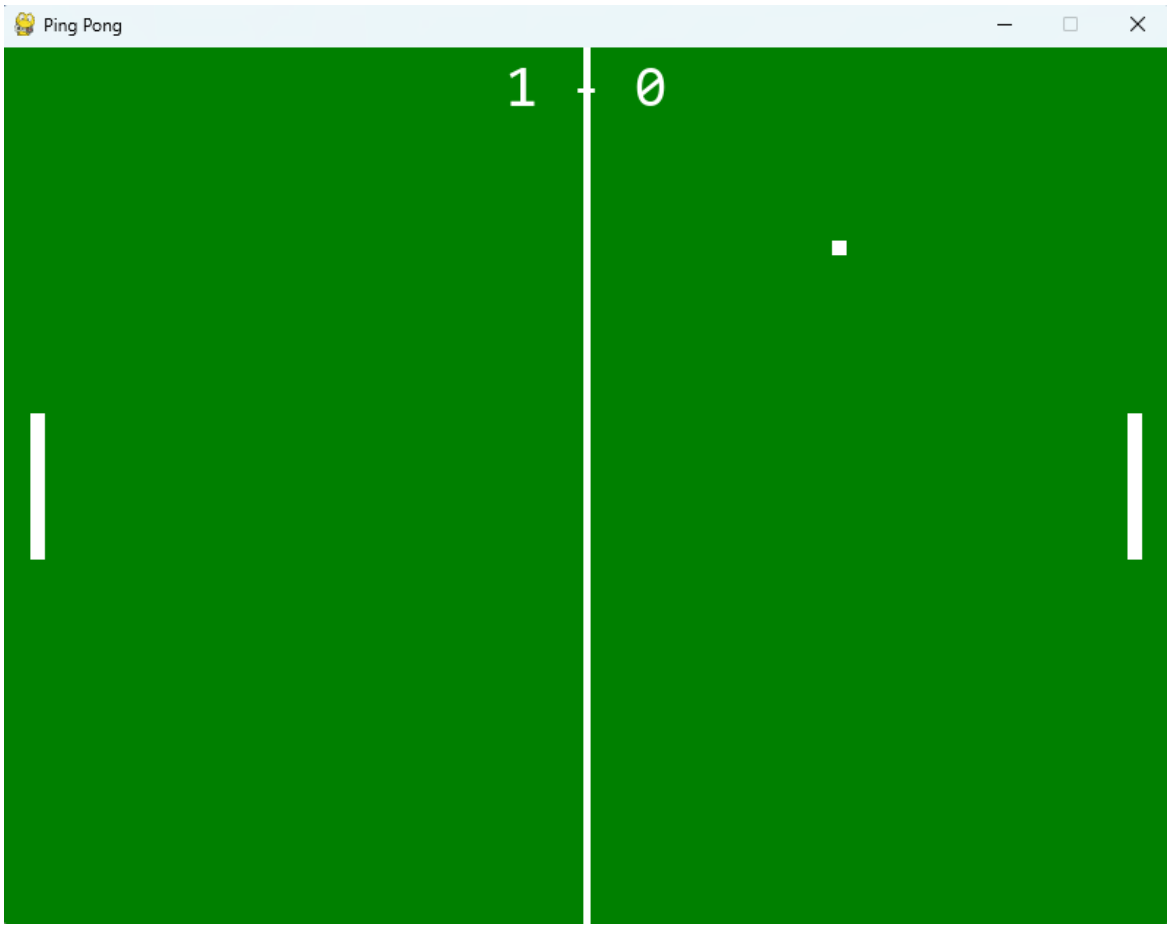
Este código implementa un juego básico de Ping Pong con Pygame, que incluye la gestión de nombres de jugadores, la selección de puntos para ganar, y un historial de partidas. Las clases `Paleta` y `Pelota` manejan los objetos del juego, mientras que varias funciones controlan el flujo del juego y la interacción del usuario.

ANEXO


Ping Pong
Jugador 1: eric


Ping Pong
Jugador 2: juan


Ping Pong
Puntos a Ganar: 5


Ping Pong
1 - 0