EXAMEN FINAL OPTATIVA I

PYTHON

DOCUMENTACION DEL CÓDIGO

NOMBRE: JESÚS DANILO OVELAR DUR

DOCENTE: ING. RICARDO MAIDANA

SEMESTRE: 9NO

CAACUPÉ – PARAGUAY

2024

1. Introducción

1.1 Propósito del Programa

El propósito de este proyecto es meramente para entretenimiento de las personas, el juego en cuestión es un “PingPong” desarrollado en el lenguaje Python utilizando una biblioteca especifica que sería Pygame, es un juego diferente al tradicional pong ya que no necesitaríamos de dos personas para poder jugar.

1.2 Requisitos

- Python 3.x
- Juego Py

2. Estructura del Código

2.1 Importación de Bibliotecas

```
import pygame
import pygame_gui
import sys
import random
from math import
```

- **pygame:** Biblioteca para desarrollar videojuegos.
- **pygame_gui:** Biblioteca para crear interfaces gráficas con Pygame.
- **sys:** Proporciona acceso a variables y funciones del sistema.
- **random:** Para generar números aleatorios.
- **math:** Para funciones matemáticas, como seno y coseno.

2.2 Inicialización y configuración

```
pygame.init()
# Cargar el sonido
pong_sound = pygame.mixer.Sound("pong.ogg")
score_sound = pygame.mixer.Sound("score.ogg")
width = 800
height = 600
display = pygame.display.set_mode((width, height))
pygame.display.set_caption("Pong!")
clock = pygame.time.Clock()
```

Estas líneas inicializan Pygame, configura los sonidos también dimensiona la ventana del juego y el titulo y por ultimo inicializa el reloj para aumentar la velocidad de la pelota

2.3 Definición de Colores y Otras Variables

```
background = (51, 60, 135)
white = (236, 240, 241)
red = (203, 67, 53)
blue = (52, 152, 219)
yellow = (244, 208, 63)
```

```
dark_red = (178, 34, 34)
dark_blue = (0, 0, 139)
top = White
bottom = white
left = white
right = white
margin = 4
scoreLeft = 0
scoreRight = 0
maxScore = 20
playerName = "Jugador"
font = pygame.font.SysFont("Arial", 30)
largeFont = pygame.font.SysFont("Arial", 60)
smallFont = pygame.font.SysFont("Arial", 20)
# Cargar la imagen de la pelota
ball_image = pygame.image.load("pelota.png")
ball_image = pygame.transform.scale(ball_image, (20, 20))
```

Se definen los colores usados en el juego, también se define los márgenes y variables del puntaje y se carga la fuente para el texto y la imagen de la pelota.

2.4 Dibujo del Límite del Tablero

```
def boundary():
    global top, bottom, left, right
    pygame.draw.rect(display, left, (0, 0, margin, height))
    pygame.draw.rect(display, top, (0, 0, width, margin))
    pygame.draw.rect(display, right, (width-margin, 0, margin,
height))
    pygame.draw.rect(display, bottom, (0, height - margin, width,
margin))
```

```
    l = 25
    # Dibuja una línea discontinua
    dash_length = 14
    gap_length = 7
    for y in range(12, height - 20, dash_length + gap_length):
pygame.draw.rect(display, white, (width / 2 - margin / 2, y, margin,
dash_length))
```

Se dibujan los limites del tablero de juego y dibuja una línea discontinua simulando la red en el centro del tablero.

3. Pantalla de Inicio y Fin del juego

```
def guardar_puntuaciones(puntuaciones):     with
open("puntuaciones.pkl", "wb") as archivo:
        pickle.dump(puntuaciones, archivo)
 def
cargar_puntuaciones():
    try:
        with open("puntuaciones.pkl", "rb") as archivo:
            return pickle.load(archivo)
except FileNotFoundError:
return []
```

Define las funciones para la pantalla de inicio y la pantalla de fin del juego.

4. Clases

4.1 Clase Paddle (Pala)

```
class Paddle:
    def __init__(self, position):
        self.w = 10
        self.h = self.w*8
        self.paddleSpeed = 6

        if position == -1:
            self.x = 1.5*margin
            self.color = dark_red
        else:
            self.x = width - 1.5*margin - self.w
            self.color = blue

        self.y = height/2 - self.h/2

    def show(self):
        pygame.draw.rect(display, self.color, (self.x, self.y, self.w,
self.h))

    def move(self, ydir):
        self.y += self.paddleSpeed*ydir
        if self.y < 0:
            self.y -= self.paddleSpeed*ydir
        elif self.y + self.h> height:
            self.y -= self.paddleSpeed*ydir

    def ai_move(self, ball):
        if ball.y < self.y + self.h/2:
            self.move(-1)
        elif ball.y > self.y + self.h/2:
            self.move(1)

leftPaddle = Paddle(-1)
rightPaddle = Paddle(1)
```

Define la clases de las palas e incluye métodos para mostrar, mover y el movimiento automático de la ia.

4.2 Clase Ball (Pelota)

```
class Ball:
    def __init__(self, color):
        self.r = 20
        self.x = width/2 - self.r/2
        self.y = height/2 -self.r/2
        self.color = color
        self.angle = random.randint(-75, 75)
        if random.randint(0, 1):
            self.angle += 180

        self.speed = 8
        self.initial_speed = 8
        self.last_point_time = pygame.time.get_ticks()
        self.ball_speed_increase_rate = 0.009
```

```
    def show(self):
        display.blit(ball_image, (self.x, self.y))

    def move(self):
        global scoreLeft, scoreRight
        self.x += self.speed*cos(radians(self.angle))
        self.y += self.speed*sin(radians(self.angle))
        if self.x + self.r > width - margin:
            scoreLeft += 1
            score_sound.play()
            self.resetBall()
            pygame.time.wait(1500)
        if self.x < margin:
            scoreRight += 1
            score_sound.play()
            self.resetBall()
            pygame.time.wait(1500)
        if self.y < margin:
            self.angle = - self.angle
        if self.y + self.r  >=height - margin:
            self.angle = - self.angle

        current_time = pygame.time.get_ticks()
        if current_time - self.last_point_time > 10000:
            self.speed += self.ball_speed_increase_rate

    def resetBall(self):
        self.x = width/2 - self.r/2
        self.y = height/2 -self.r/2
        self.angle = random.randint(-75, 75)
        if random.randint(0, 1):
            self.angle += 180
        self.speed = self.initial_speed
        self.last_point_time = pygame.time.get_ticks()

    def checkForPaddle(self):
        if self.x < width/2:
            if leftPaddle.x < self.x < leftPaddle.x + leftPaddle.w:
                if leftPaddle.y < self.y < leftPaddle.y + 10 or
leftPaddle.y < self.y + self.r< leftPaddle.y + 10:
                    self.angle = -45
                    pong_sound.play()
                if leftPaddle.y + 10 < self.y < leftPaddle.y + 20 or
leftPaddle.y + 10 < self.y + self.r< leftPaddle.y + 20:
                    self.angle = -30
                    pong_sound.play()
                if leftPaddle.y + 20 < self.y < leftPaddle.y + 30 or
leftPaddle.y + 20 < self.y + self.r< leftPaddle.y + 30:
                    self.angle = -15
                    pong_sound.play()
                if leftPaddle.y + 30 < self.y < leftPaddle.y + 40 or
leftPaddle.y + 30 < self.y + self.r< leftPaddle.y + 40:
                    self.angle = -10
                    pong_sound.play()
                if leftPaddle.y + 40 < self.y < leftPaddle.y + 50 or
leftPaddle.y + 40 < self.y + self.r< leftPaddle.y + 50:
                    self.angle = 10
                    pong_sound.play()
```

```
                if leftPaddle.y + 50 < self.y < leftPaddle.y + 60 or
leftPaddle.y + 50 < self.y + self.r< leftPaddle.y + 60:
                    self.angle = 15
                    pong_sound.play()
                if leftPaddle.y + 60 < self.y < leftPaddle.y + 70 or
leftPaddle.y + 60 < self.y + self.r< leftPaddle.y + 70:
                    self.angle = 30
                    pong_sound.play()
                if leftPaddle.y + 70 < self.y < leftPaddle.y + 80 or
leftPaddle.y + 70 < self.y + self.r< leftPaddle.y + 80:
                    self.angle = 45
                    pong_sound.play()
        else:
            if rightPaddle.x + rightPaddle.w > self.x  + self.r >
rightPaddle.x:
                if rightPaddle.y < self.y < leftPaddle.y + 10 or
leftPaddle.y < self.y + self.r< leftPaddle.y + 10:
                    self.angle = -135
                    pong_sound.play()
                if rightPaddle.y + 10 < self.y < rightPaddle.y + 20 or
rightPaddle.y + 10 < self.y + self.r< rightPaddle.y + 20:
                    self.angle = -150
                    pong_sound.play()
                if rightPaddle.y + 20 < self.y < rightPaddle.y + 30 or
rightPaddle.y + 20 < self.y + self.r< rightPaddle.y + 30:
                    self.angle = -165
                    pong_sound.play()
                if rightPaddle.y + 30 < self.y < rightPaddle.y + 40 or
rightPaddle.y + 30 < self.y + self.r< rightPaddle.y + 40:
                    self.angle = -170
                    pong_sound.play()
                if rightPaddle.y + 40 < self.y < rightPaddle.y + 50 or
rightPaddle.y + 40 < self.y + self.r< rightPaddle.y + 50:
                    self.angle = 170
                    pong_sound.play()
                if rightPaddle.y + 50 < self.y < rightPaddle.y + 60 or
rightPaddle.y + 50 < self.y + self.r< rightPaddle.y + 60:
                    self.angle = 165
                    pong_sound.play()
                if rightPaddle.y + 60 < self.y < rightPaddle.y + 70 or
rightPaddle.y + 60 < self.y + self.r< rightPaddle.y + 70:
                    self.angle = 150
                    pong_sound.play()
                if rightPaddle.y + 70 < self.y < rightPaddle.y + 80 or
rightPaddle.y + 70 < self.y + self.r< rightPaddle.y + 80:
                    self.angle = 135
                    pong_sound.play()

ball = Ball(yellow)
```

Se define la clase de la pelota, esta clase incluye métodos para mostrar, mover, y resetear la pelota, así como para detectar colisiones con las palas.

5. Función principal

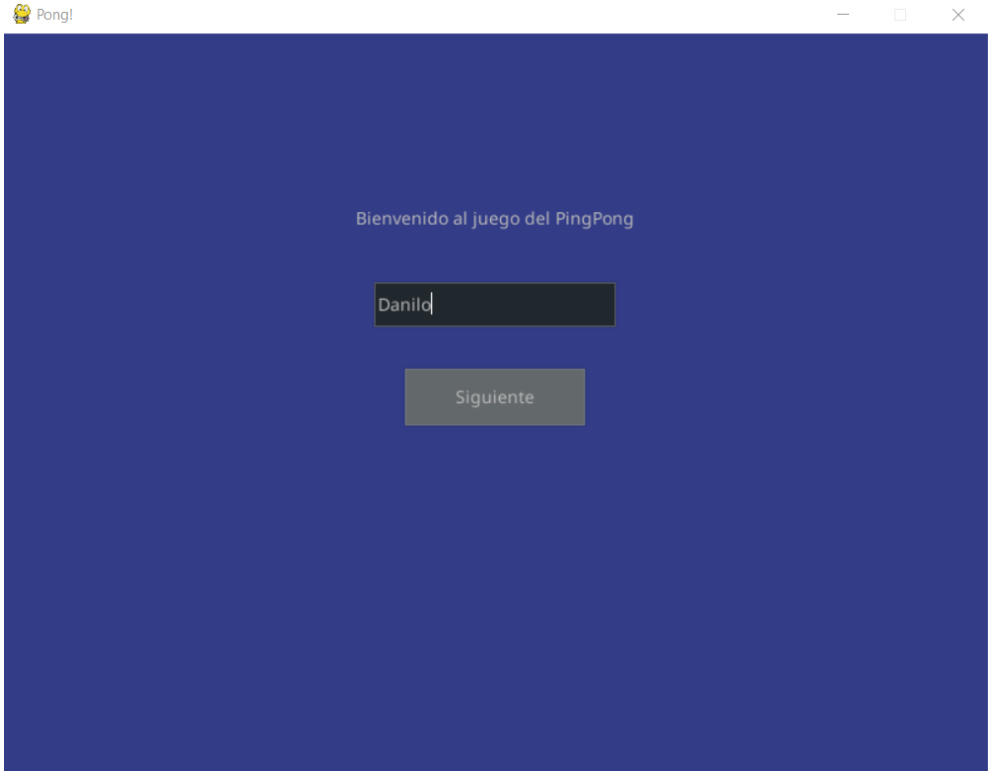
La función “main” inicia el juego, maneja los eventos y la lógica del juego, y actualiza la pantalla.

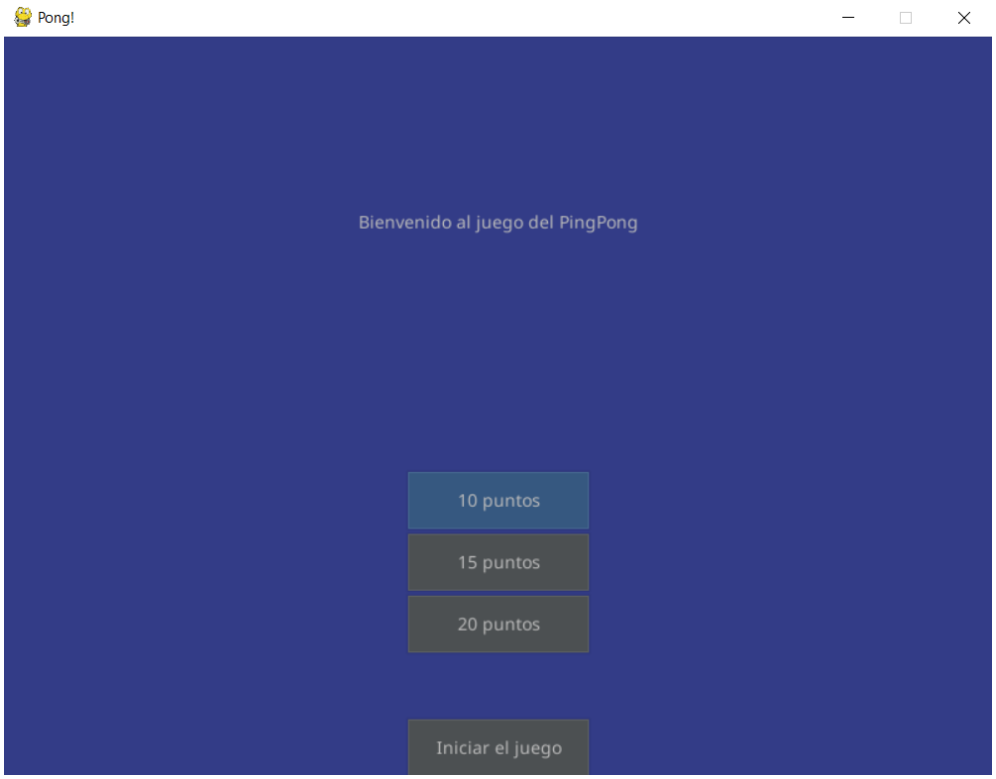
```
def main():
    global scoreLeft, scoreRight
    scoreLeft = 0
    scoreRight = 0
    startScreen()

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

        keys =
pygame.key.get_pressed()
        if keys[pygame.K_w]:
            leftPaddle.move(-1)
        if keys[pygame.K_s]:
            leftPaddle.move(1)
        if keys[pygame.K_UP]:
            rightPaddle.move(-1)
        if keys[pygame.K_DOWN]:
            rightPaddle.move(1)

        ball.move()
        ball.checkForPaddle()
        rightPaddle.ai_move(ball)

        display.fill(background)
        boundary()
        leftPaddle.show()
        rightPaddle.show()
        ball.show()

        text =
font.render(str(scoreLeft), 1, white)
        display.blit(text, (width/4 -
text.get_width()/2, margin*4))
        text =
font.render(str(scoreRight), 1,
white)
        display.blit(text, (width*3/4
- text.get_width()/2, margin*4))

        if scoreLeft >= maxScore:
            gameOver(playerName)
            return
        if scoreRight >= maxScore:
            gameOver("Computer")
            return

        pygame.display.update()
        clock.tick(60)

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

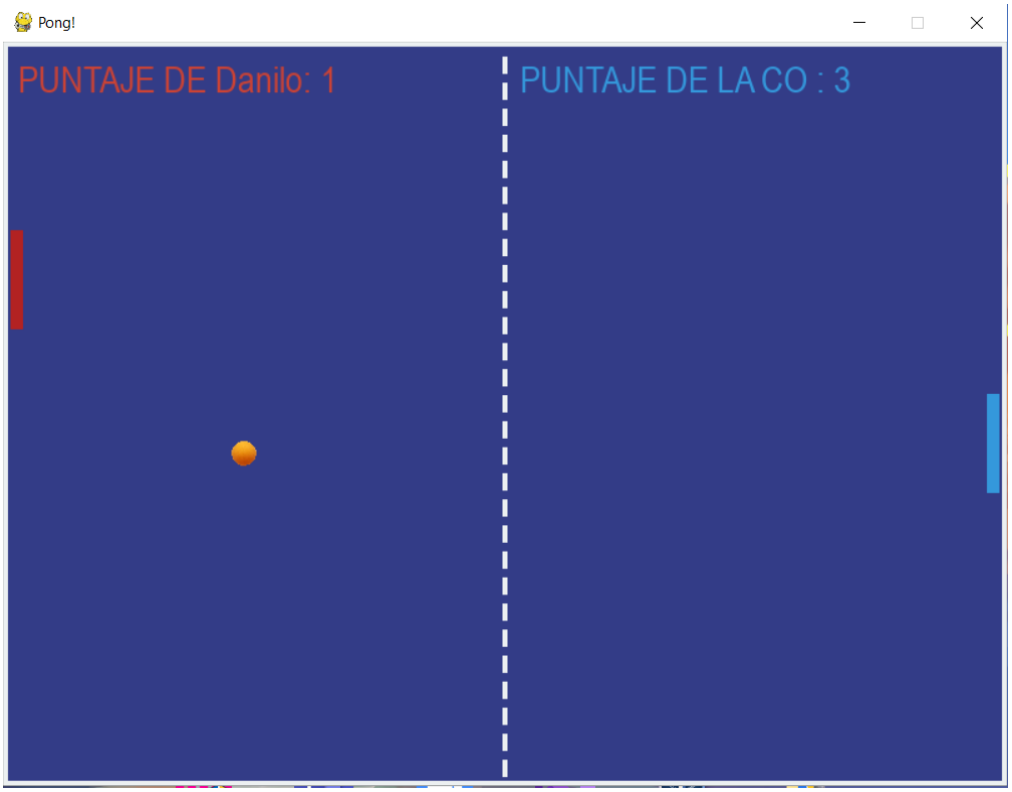
6. Conclusión

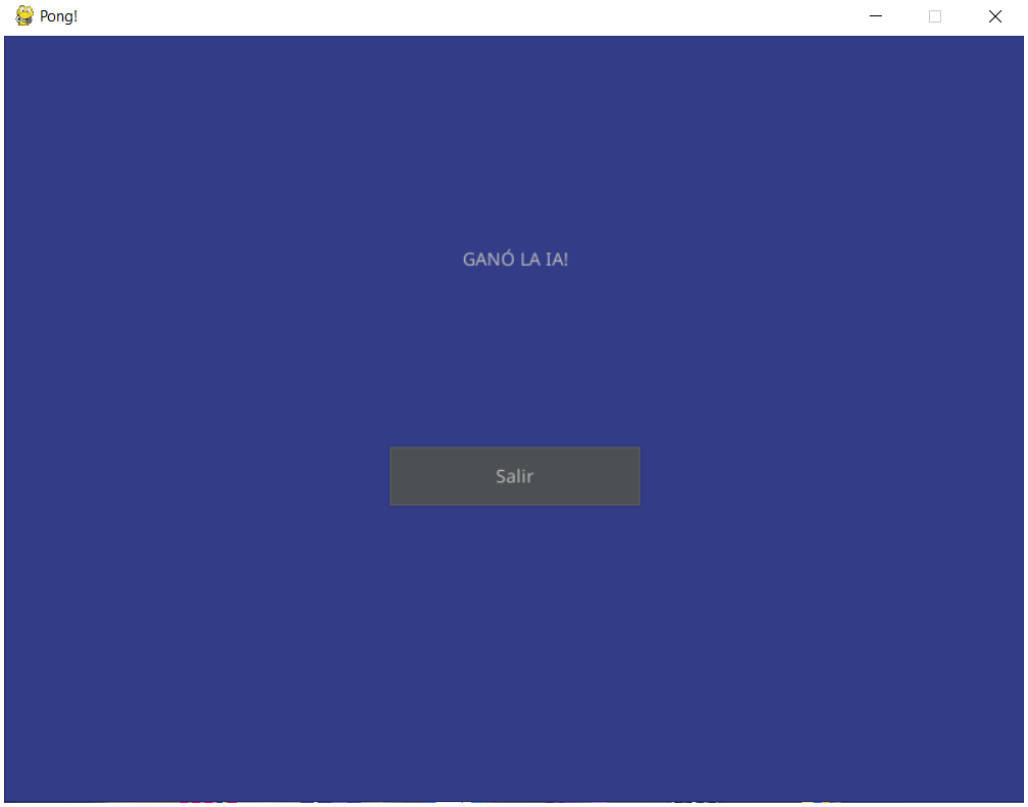
El juego creado en Python llamado “PinPong” esta diseñado para ser divertido y fácil de usar, al no ser necesario otro jugador debido a la implementación de una “IA” como oponente y también al tener un interfaz y menú bastante amigable hace que el usuario tenga una experiencia bastante buena y fluida.

ANEXO


Pong!
Bienvenido al juego del PingPong
Danilo
Siguiente


Pong!
Bienvenido al juego del PingPong
10 puntos
15 puntos
20 puntos
Iniciar el juego


Pong!
PUNTAJE DE Danilo: 1
PUNTAJE DE LA CO : 3


Pong!
GANÓ LA IA!
Salir