[Optativa I]

Examen Final Optativa I
[Pygame]

Presentado por:

[Aldo Rene Barrios Gómez]

Director del Trabajo:

[Ing. Ricardo Maidana]

Curso:

Quinto Año, noveno semestre

Turno:

Noche

Caacupé-Paraguay

INTRODUCCION

El programa es un juego llamado “Dispárando Corazones con Aldo Barrios” desarrollado en Python utilizando la biblioteca pygame. El objetivo es que el corazón que va cayendo no toque el piso o no toque al jugador.

Descripción del Juego

"Disparando Corazones con Aldo Barrios" es un juego de disparos en 2D donde el jugador controla un personaje que dispara corazones a los enemigos. El objetivo es obtener la mayor cantidad de puntos posible antes de ser golpeado por un enemigo. El juego cuenta con una pantalla de inicio para ingresar el nombre del jugador y presenta una tabla de puntajes altos.

Requisitos del Sistema

- Python 3.x
- Pygame

Instalación

1. **Instalar Python:** Asegúrate de tener Python 3.x instalado en tu sistema. Puedes descargarlo desde python.org.
2. **Instalar Pygame:** Ejecuta el siguiente comando en tu terminal o consola de comandos:

bash

pip install pygame

3. **Descargar el Código del Juego:** Descarga o clona el repositorio con el código del juego.

Ejecución del Juego

Para ejecutar el juego, navega al directorio donde se encuentra el archivo main.py y ejecuta el siguiente comando:

bash

python main.py

Estructura del Código

El código del juego está organizado en varias funciones principales para manejar diferentes aspectos del juego:

1. **show_start_screen():** Muestra la pantalla de inicio donde el jugador puede ingresar su nombre.
2. **pause_game():** Alterna el estado de pausa del juego.
3. **update_score(score):** Actualiza y devuelve el texto del puntaje actual.
4. **show_game_over(score):** Muestra la pantalla de Game Over y el botón de reinicio, además de los puntajes más altos.

Código del Juego

```
 import pygame
import sys
import random
import json

# Función para la pantalla de inicio
def show_start_screen():
    global player_name
    title_font = pygame.font.Font(None, 60)
    title_text = title_font.render("Disparando Corazónes", True, (255, 255, 255))

    subtitle_font = pygame.font.Font(None, 24)
    subtitle_text = subtitle_font.render("Ingresa tu nombre y presiona Enter", True, (255, 255,
255))

    input_box = pygame.Rect(width // 2 - 100, height // 2, 200, 40)
    active = False
    color_inactive = pygame.Color('lightskyblue3')
    color_active = pygame.Color('dodgerblue2')
    color = color_inactive
    player_name = ''

    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                if input_box.collidepoint(event.pos):
                    active = not active
                else:
                    active = False
                color = color_active if active else color_inactive
            if event.type == pygame.KEYDOWN:
                if active:
                    if event.key == pygame.K_RETURN:
```

```
                return
            elif event.key == pygame.K_BACKSPACE:
                player_name = player_name[:-1]
            else:
                player_name += event.unicode

        screen.blit(background_image, (0, 0))
        screen.blit(title_text, (width // 2 - title_text.get_width() // 2, height // 2 - 100))
        screen.blit(subtitle_text, (width // 2 - subtitle_text.get_width() // 2, height // 2 - 50))

        txt_surface = subtitle_font.render(player_name, True, color)
        width_box = max(200, txt_surface.get_width() + 10)
        input_box.w = width_box

        screen.blit(txt_surface, (input_box.x + 5, input_box.y + 5))
        pygame.draw.rect(screen, color, input_box, 2)

        pygame.display.flip()

# Función para manejar la pausa y reanudación del juego
def pause_game():
    global paused
    paused = not paused  # Alternar entre pausa y reanudación

# Función para manejar los puntajes
def update_score(score):
    score_font = pygame.font.Font(None, 36)
    score_text = score_font.render(f"Puntaje: {score}", True, (255, 255, 255))
    score_rect = score_text.get_rect()
    score_rect.centerx = width // 2
    score_rect.top = pause_button_rect.bottom + 10  # Alineado debajo del botón de
pausa/reanudar
    return score_text, score_rect

# Función para mostrar pantalla de Game Over y botón de reinicio
def show_game_over(score):
    game_over_font = pygame.font.Font(None, 80)
    game_over_text = game_over_font.render("Game Over", True, (255, 0, 0))
    game_over_rect = game_over_text.get_rect()
    game_over_rect.center = (width // 2, height // 2 - 50)

    score_font = pygame.font.Font(None, 36)
    score_text = score_font.render(f"Puntaje: {score}", True, (255, 255, 255))
    score_rect = score_text.get_rect()
    score_rect.center = (width // 2, height // 2 + 50)

    # Botón de reinicio
    restart_button_rect = pygame.Rect(width // 2 - 100, height // 2 + 100, 200, 40)
    pygame.draw.rect(screen, (0, 255, 0), restart_button_rect)
    restart_text = score_font.render("Reiniciar", True, (255, 255, 255))
    screen.blit(restart_text, (restart_button_rect.centerx - restart_text.get_width() // 2,
restart_button_rect.centery - restart_text.get_height() // 2))
```

```
    screen.blit(game_over_text, game_over_rect)
    screen.blit(score_text, score_rect)

    # Mostrar puntajes más altos
    high_scores = load_high_scores()
    high_scores_text = score_font.render("Puntajes Más Altos", True, (255, 255, 255))
    screen.blit(high_scores_text, (width // 2 - high_scores_text.get_width() // 2, height // 2 +
150))

    y_offset = height // 2 + 200
    for entry in high_scores:
        entry_text = score_font.render(f"{entry['name']}: {entry['score']}", True, (255, 255, 255))
        screen.blit(entry_text, (width // 2 - entry_text.get_width() // 2, y_offset))
        y_offset += 40

    pygame.display.flip()

    waiting = True
    while waiting:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                if restart_button_rect.collidepoint(event.pos):
                    save_score(player_name, score)
                    return True  # Reiniciar el juego

    return False  # No reiniciar el juego

# Función para cargar los puntajes altos desde un archivo JSON
def load_high_scores():
    try:
        with open('high_scores.json', 'r') as f:
            return json.load(f)
    except FileNotFoundError:
        return []

# Función para guardar un nuevo puntaje alto en el archivo JSON
def save_score(name, score):
    high_scores = load_high_scores()
    high_scores.append({'name': name, 'score': score})
    high_scores = sorted(high_scores, key=lambda x: x['score'], reverse=True)[:10]  # Mantener
solo los 10 puntajes más altos
    with open('high_scores.json', 'w') as f:
        json.dump(high_scores, f)

# Inicializar Pygame
pygame.init()

# Configuración de la pantalla
width, height = 700, 775
screen = pygame.display.set_mode((width, height))
```

```
pygame.display.set_caption("Disparando ♡ con Aldo Barrios ")

# Cargar imágenes y escalarlas
player_image = pygame.image.load("imgs/messi.png")
player_image = pygame.transform.scale(player_image, (60, 60))

bullet_image = pygame.image.load("imgs/corazón.png")
bullet_image = pygame.transform.scale(bullet_image, (50, 50))

enemy_image = pygame.image.load("imgs/antonela.png")
enemy_image = pygame.transform.scale(enemy_image, (60, 60))

background_image = pygame.image.load("imgs/fondo_amor.jpg")
background_image = pygame.transform.scale(background_image, (width, height))

# Jugador
player_rect = player_image.get_rect()
player_rect.centerx = width // 2
player_rect.bottom = height - 10
player_speed = 10

# Bala
bullet_speed = 10
bullets = []

# Enemigo
enemy_speed = 5
enemies = []

# Botones
pause_button_rect = pygame.Rect(20, 20, 200, 40)  # Aumentamos el ancho del botón
exit_button_rect = pygame.Rect(width - 120, 20, 100, 40)

# Puntaje
score = 0
score_text, score_rect = update_score(score)

# Reloj
clock = pygame.time.Clock()

# Mantener registro de teclas presionadas
keys_pressed = {'left': False, 'right': False, 'up': False, 'down': False}

# Estado inicial del juego
paused = False

# Mostrar pantalla de inicio
show_start_screen()

# Bucle principal del juego
running = True
while running:
    for event in pygame.event.get():
```

```
    if event.type == pygame.QUIT:
        pygame.quit()
        sys.exit()

    # Manejar eventos de clic del ratón para los botones
    if event.type == pygame.MOUSEBUTTONDOWN:
        if pause_button_rect.collidepoint(event.pos):
            pause_game()
        elif exit_button_rect.collidepoint(event.pos):
            pygame.quit()
            sys.exit()

    # Manejar movimientos del jugador si no está pausado
    if not paused:
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                keys_pressed['left'] = True
            elif event.key == pygame.K_RIGHT:
                keys_pressed['right'] = True
            elif event.key == pygame.K_UP:
                keys_pressed['up'] = True
            elif event.key == pygame.K_DOWN:
                keys_pressed['down'] = True
            elif event.key == pygame.K_SPACE:
                bullet_rect = bullet_image.get_rect()
                bullet = {
                    'rect': pygame.Rect(
                        player_rect.centerx - bullet_rect.width // 2,
                        player_rect.top - bullet_rect.height,
                        bullet_rect.width,
                        bullet_rect.height
                    ),
                    'image': bullet_image
                }
                bullets.append(bullet)

        if event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT:
                keys_pressed['left'] = False
            elif event.key == pygame.K_RIGHT:
                keys_pressed['right'] = False
            elif event.key == pygame.K_UP:
                keys_pressed['up'] = False
            elif event.key == pygame.K_DOWN':
                keys_pressed['down'] = False

# Si el juego está pausado, no actualizar el juego
if paused:
    continue

# Actualizar posición del jugador
if keys_pressed['left'] and player_rect.left > 0:
    player_rect.x -= player_speed
```

```
    if keys_pressed['right'] and player_rect.right < width:
        player_rect.x += player_speed
    if keys_pressed['up'] and player_rect.top > 0:
        player_rect.y -= player_speed
    if keys_pressed['down'] and player_rect.bottom < height:
        player_rect.y += player_speed

    # Actualizar posición de las balas
    for bullet in bullets:
        bullet['rect'].y -= bullet_speed

    # Generar enemigos aleatorios
    if random.randint(0, 100) < 5:
        enemy_rect = enemy_image.get_rect()
        enemy_rect.x = random.randint(0, width - enemy_rect.width)
        enemies.append(enemy_rect.copy())

    # Actualizar posición de los enemigos
    for enemy in enemies:
        enemy.y += enemy_speed

    # Colisiones entre balas y enemigos
    for bullet in bullets:
        for enemy in enemies:
            if enemy.colliderect(bullet['rect']):
                bullets.remove(bullet)
                enemies.remove(enemy)
                score += 1
                score_text, score_rect = update_score(score)

    # Colisiones entre jugador y enemigos
    for enemy in enemies:
        if player_rect.colliderect(enemy) or enemy.bottom >= height:
            if show_game_over(score):  # Si el jugador decide reiniciar
                # Reiniciar variables
                player_rect.centerx = width // 2
                player_rect.bottom = height - 10
                bullets = []
                enemies = []
                score = 0
                score_text, score_rect = update_score(score)
                paused = False
                keys_pressed = {'left': False, 'right': False, 'up': False, 'down': False}  # Reiniciar el
estado de las teclas
                show_start_screen()  # Mostrar pantalla de inicio nuevamente
            else:
                pygame.quit()
                sys.exit()

    # Limpiar la pantalla con el fondo
    screen.blit(background_image, (0, 0))

    # Dibujar al jugador
```

```
    screen.blit(player_image, player_rect)

    # Dibujar las balas
    for bullet in bullets:
        screen.blit(bullet['image'], bullet['rect'].topleft)

    # Dibujar los enemigos
    for enemy in enemies:
        screen.blit(enemy_image, enemy)

    # Dibujar botones
    pygame.draw.rect(screen, (0, 0, 255), pause_button_rect)
    pygame.draw.rect(screen, (255, 0, 0), exit_button_rect)

    # Texto en los botones
    button_font = pygame.font.Font(None, 24)
    pause_text = button_font.render("Pausar/Reanudar", True, (255, 255, 255))
    exit_text = button_font.render("Salir", True, (255, 255, 255))
    screen.blit(pause_text, (pause_button_rect.centerx - pause_text.get_width() // 2,
pause_button_rect.centery - pause_text.get_height() // 2))
    screen.blit(exit_text, (exit_button_rect.centerx - exit_text.get_width() // 2,
exit_button_rect.centery - exit_text.get_height() // 2))

    # Dibujar área de puntajes
    pygame.draw.rect(screen, (0, 0, 0, 128), score_rect)  # Fondo transparente según el tamaño
del texto
    screen.blit(score_text, score_rect)

    # Actualizar la pantalla
    pygame.display.flip()

    # Establecer límite de FPS
    clock.tick(30)
```

Instrucciones del Juego

1. **Pantalla de Inicio:**
    - Ingresa tu nombre en la caja de texto y presiona Enter para comenzar el juego.
2. **Controles del Juego:**
    - Moverse a la izquierda: Tecla izquierda
    - Moverse a la derecha: Tecla derecha
    - Moverse hacia arriba: Tecla arriba
    - Moverse hacia abajo: Tecla abajo
    - Disparar: Barra espaciadora
    - Pausar/Reanudar: Botón en la esquina superior izquierda
    - Salir: Botón en la esquina superior derecha
3. **Objetivo del Juego:**
    - Disparar corazones a los enemigos para obtener puntos.
    - Evitar colisiones con los enemigos.
    - Al final del juego, los puntajes más altos se mostrarán y se guardarán automáticamente.

Archivos del Juego

- main.py: Contiene el código principal del juego.
- high_scores.json: Almacena los puntajes más altos del juego.
- imgs/messi.png: Imagen del jugador.
- imgs/corazón.png: Imagen de la bala.
- imgs/antonela.png: Imagen del enemigo.
- imgs/fondo_amor.jpg: Imagen de fondo del juego.

Créditos

- **Desarrollador:** Aldo Barrios
- **Asistencia en Documentación:** ChatGPT de OpenAI