Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Ingeniería en computación

Seminario de solución de problemas de sistemas operativos

NRC 164138

Sección D07

Profesor Javier Rosales Martínez

Diego Armando Sánchez Rubio 217570609

Reporte Práctica 9.

Antecedentes.

En esta práctica se revisará un programa el cuál actuará con Hilos, cada imagen será movido con un hilo diferente, para que de está forma se puedan mover simultáneamente y que uno no depende el otro, ya que esto provocaría el programa se trabará o se detuviera cada cierto tiempo, por lo tanto ambos movimientos se gestionan por separado en funciones que se ejecutan en hilos independientes, lo que permite simular animaciones concurrentes.

Metodología.

Lo primero que se realizó en esta práctica fue la importación de las librerías que se van a usar, las cuales son las siguientes, tkinter sirve para crear interfaces gráficas.

PIL.Image y PIL.ImageTk permiten cargar y adaptar imágenes para mostrarlas en tkinter.

threading se usa para ejecutar tareas al mismo tiempo, sin que se bloqueen entre sí.

time se usa para manejar pausas (con sleep), en este caso para controlar la velocidad del movimiento.

```
1   import tkinter as tk
2   from PIL import Image, ImageTk
3   import threading
4   import time
```

Ahora se crea la ventana principal de la aplicación usando tk.Tk(), se le da un título, y se define su tamaño en píxeles con .geometry("600x400"), es decir, 600 de ancho por 400 de alto.

```
# Crear ventana
ventana = tk.Tk()
ventana.title("Movimiento con hilos")
ventana.geometry("600x400")
```

Ahora se crea un Canvas, que es un lienzo blanco dentro de la ventana donde se pueden dibujar imágenes, formas, etc. El tamaño del canvas es el mismo que la ventana, y se coloca (con pack()) para que se muestre. También se cargan dos imágenes (1.png y 2.png) desde el disco y se redimensionan a 50x50 píxeles. Luego, con ImageTk.PhotoImage, se convierten en un formato que tkinter puede mostrar en el Canvas, a cada una se le da una posición en el canvas y se crean dos variables para el control de las posiciones según vaya avanzando.

```
# Canvas
canvas = tk.Canvas(ventana, width=600, height=400, bg="white")
canvas.pack()

# Cargar imágenes
img1 = Image.open("1.png").resize((50, 50))
img2 = Image.open("2.png").resize((50, 50))

tk_img1 = ImageTk.PhotoImage(img1)
tk_img2 = ImageTk.PhotoImage(img2)

# Crear imágenes en el canvas
img1_id = canvas.create_image(0, 150, anchor="nw", image=tk_img1)
img2_id = canvas.create_image(300, 0, anchor="nw", image=tk_img2)

# Posiciones
pos_x = 0
pos_y = 0
```

Ahora la función se encarga de mover la primera imagen (img1_id) de izquierda a derecha:

Se declara pos_x como global para poder modificar su valor dentro de la función.

Se inicia un ciclo while que se ejecuta mientras pos_x sea menor que 550. Esto es porque el canvas mide 600 píxeles de ancho y la imagen mide 50, así que no debe pasarse del borde.

En cada vuelta del ciclo:

Se incrementa pos_x en 2, simulando que la imagen avanza.

Se llama a canvas.move a través de canvas.after(0, ...), que mueve la imagen 2 píxeles hacia la derecha (eje X), y 0 en el eje Y.

Luego se hace una pausa de 0.01 segundos para controlar la velocidad del movimiento y hacerlo más suave.

Esta función se ejecuta en un hilo, así que corre en paralelo con el resto de la interfaz sin congelarla.

```
def mover_horizontal():
    global pos_x
    while pos_x < 550:
        pos_x += 2
        canvas.after(0, canvas.move, img1_id, 2, 0)
        time.sleep(0.01)
```

Ahora esta función hace algo muy similar, pero para la segunda imagen (img2_id) y en dirección vertical:

pos_y se usa para seguir el avance en el eje Y.

El ciclo while asegura que la imagen se mueva mientras pos_y sea menor que 350 (400 es el límite del canvas y la imagen mide 50 píxeles de alto).

En cada iteración:

Se suma 2 a pos_y.

La imagen se mueve 0 píxeles en X y 2 en Y.

Se pausa brevemente para suavizar el movimiento.

```python
def mover_vertical():
    global pos_y
    while pos_y < 350:
        pos_y += 2
        canvas.after(0, canvas.move, img2_id, 0, 2)
        time.sleep(0.01)
```

Conclusiones.

En resumen fue una práctica bastante sencilla para empezar a comprender cómo es que funcionan los hilos de manera independiente, no me costó trabajo hacerla y la verdad fue algo entretenida.

Referencias.

1. Lundh, F. (2001). *An Introduction to Tkinter*. PythonWare. Recuperado de http://effbot.org/tkinterbook/
2. Python Software Foundation. (2023). *threading — Thread-based parallelism*. En *Python 3.11.5 documentation*. https://docs.python.org/3/library/threading.html
3. Python Software Foundation. (2023). *Tkinter — Python interface to Tcl/Tk*. En *Python 3.11.5 documentation*. https://docs.python.org/3/library/tkinter.html
4. Clark, A. (2023). *Pillow (PIL Fork) Documentation (10.0.0)*. https://pillow.readthedocs.io/en/stable/