Universidad de Guadalajara.

Centro Universitario de Ciencias Exactas e Ingenierías.

DIVISIÓN DE TECNOLOGÍAS PARA LA INTEGRACIÓN CIBER HUMANA.

DEPARTAMENTO DE CIENCIAS COMPUTACIONALES.

TEMA: Practica1

NOMBRE DEL ESTUDIANTE:

Padilla Perez Jorge Daray

NOMBRE DE LA MATERIA: IA2

SECCION: D02

CICLO ESCOLAR: 2024-B

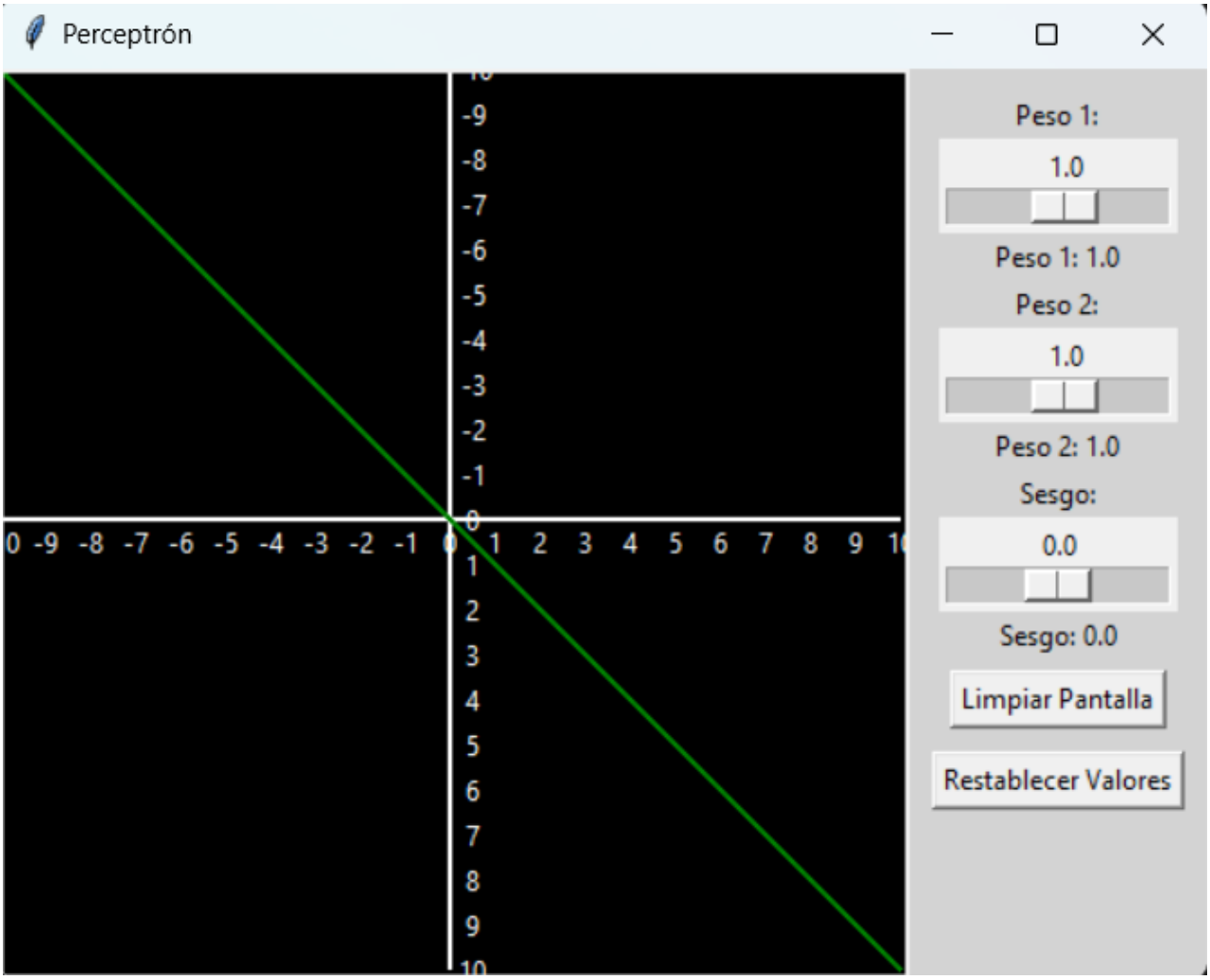
NOMBRE DEL PROFESOR: Julio Esteban

Contenido

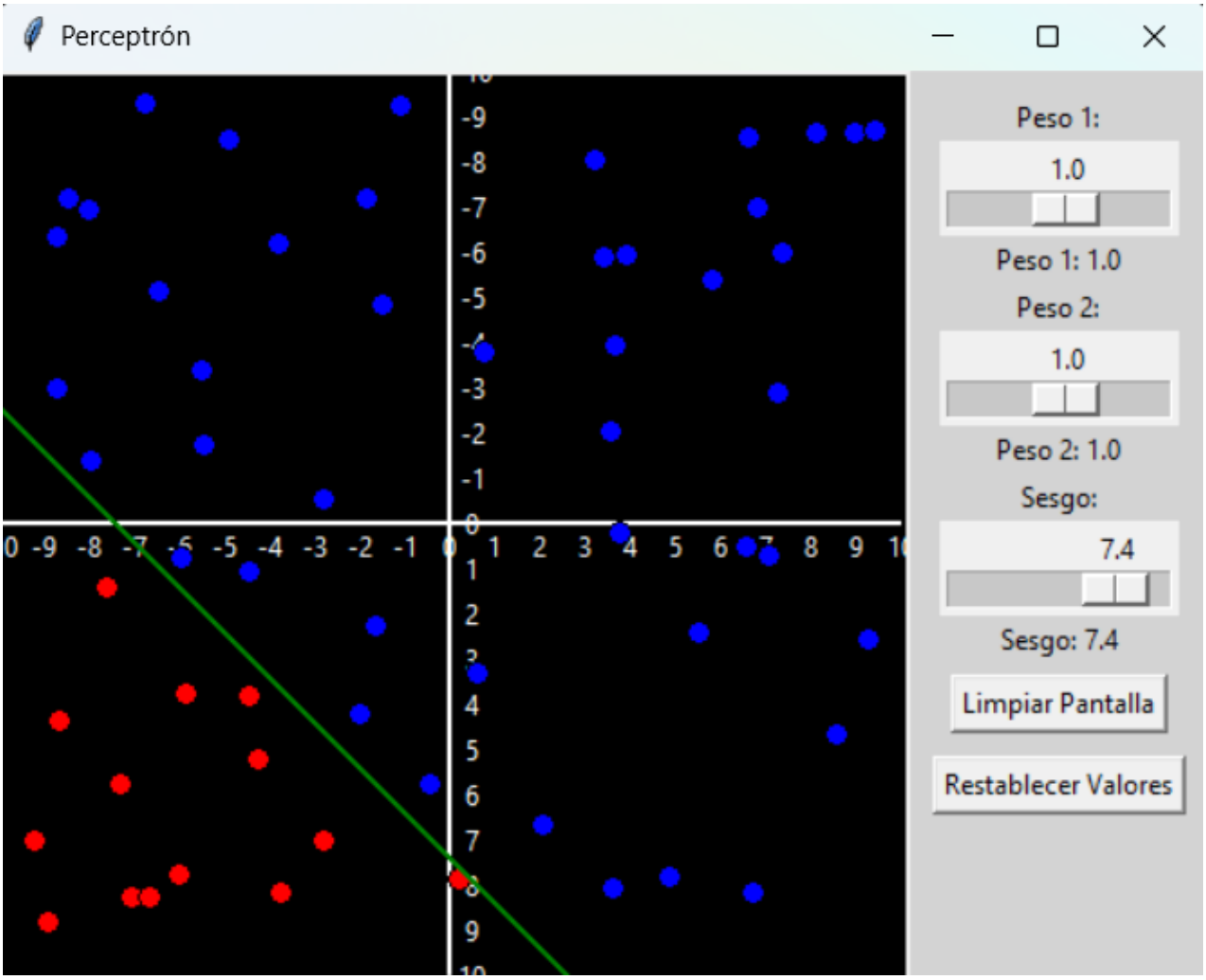
Introducción.

En este documento que se muestra como se realizó el programa numero 1 de esta materia, el cual se basa en hacer que un perceptrón se muestre gráficamente separando unos puntos que el usuario puede dibujar en un plano cartesiano,, este ultimo en la interfaz deberá de permitir al usuario ajustar los parámetros del perceptrón (pesos, bias), en mi caso este se hace de manera automática sin necesidad de botón.

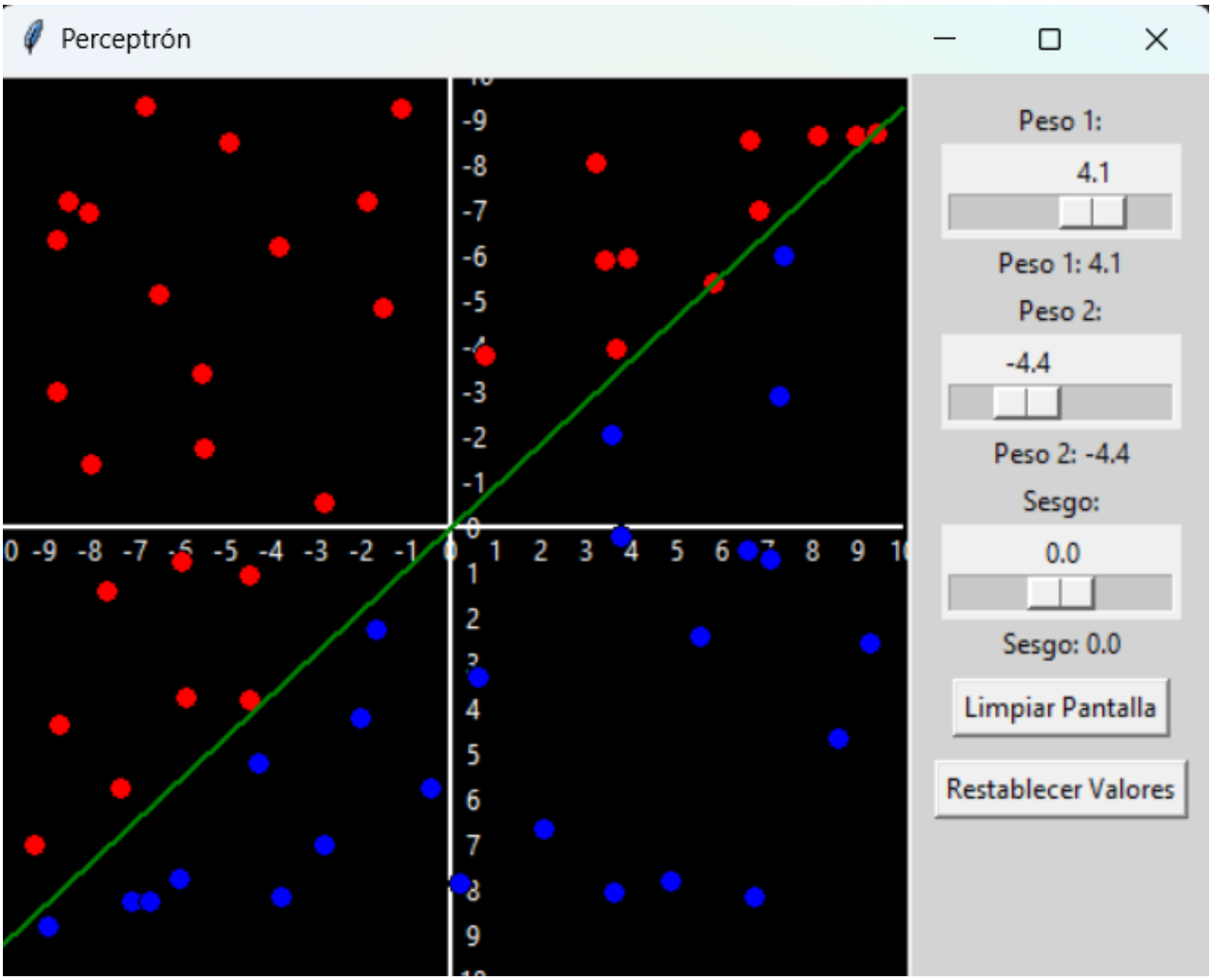
Capturas de pantalla.



Esta es la interfaz de usuario.



Aquí se puede ver ya con puntos dibujados y con el perceptrón actualizado.



Aquí mas de lo mismo con otros pesos y bias.

Código fuente.

```
import tkinter as tk
from tkinter import messagebox

class Perceptron:
    def __init__(self, weights, bias):
        self.weights = weights
        self.bias = bias

    def predict(self, x):
        z = sum(w * xi for w, xi in zip(self.weights, x)) + self.bias
        return 1 if z >= 0 else 0

def on_canvas_click(event):
    x, y = event.x, event.y
    input_vector = [(x - 200) / 20, (200 - y) / 20]  # Normalizar las coordenadas
a [-10, 10]
    prediction = perceptron.predict(input_vector)
    color = "blue" if prediction == 1 else "red"

    # Buscar el óvalo existente en las coordenadas (x, y)
    items = canvas.find_enclosed(x - 5, y - 5, x + 5, y + 5)
    if items:
        canvas.itemconfig(items[0], fill=color)  # Actualizar el color del óvalo
existente
    else:
        canvas.create_oval(x - 5, y - 5, x + 5, y + 5, fill=color)  # Crear un
nuevo óvalo

def clear_canvas():
    canvas.delete("all")  # Elimina todos los objetos en el canvas
    draw_cartesian_plane()  # Dibujar el plano cartesiano nuevamente
    update_perceptron()    # Actualizar la visualización después de limpiar

def update_perceptron(*args):
    try:
        weight1 = weight1_scale.get()
        weight2 = weight2_scale.get()
        bias = bias_scale.get()
        perceptron.weights = [weight1, weight2]
        perceptron.bias = bias

        # Eliminar la línea de decisión existente
        canvas.delete("line")
```

```
        # Dibujar la nueva línea de decisión
        x1 = -10
        y1 = (-weight1 / weight2) * x1 - (bias / weight2)
        x2 = 10
        y2 = (-weight1 / weight2) * x2 - (bias / weight2)

        # Convertir los valores de y a coordenadas en el canvas
        canvas_x1 = x1 * 20 + 200
        canvas_y1 = 200 - y1 * 20
        canvas_x2 = x2 * 20 + 200
        canvas_y2 = 200 - y2 * 20

        # Dibujar la nueva línea
        canvas.create_line(canvas_x1, canvas_y1, canvas_x2, canvas_y2,
fill="green", width=2, tags="line")

        # Actualizar los colores de los puntos existentes
        for item in canvas.find_all():
            if canvas.type(item) == "oval":
                x, y, _, _ = canvas.coords(item)
                input_vector = [(x - 200) / 20, (200 - y) / 20]
                prediction = perceptron.predict(input_vector)
                color = "blue" if prediction == 1 else "red"
                canvas.itemconfig(item, fill=color)

        # Actualizar las etiquetas de los valores actuales
        weight1_value_label.config(text=f"Peso 1: {weight1_scale.get()}")
        weight2_value_label.config(text=f"Peso 2: {weight2_scale.get()}")
        bias_value_label.config(text=f"Sesgo: {bias_scale.get()}")

    except ValueError:
        messagebox.showerror("Error", "Se produjo un error al actualizar el
perceptrón.")

def draw_cartesian_plane():
    # Dibujar ejes x e y
    canvas.create_line(0, 200, 400, 200, fill="white", width=2)
    canvas.create_line(200, 0, 200, 400, fill="white", width=2)

    # Etiquetas de coordenadas
    for i in range(-10, 11, 1):
        canvas.create_text(200 + i * 20, 210, text=str(i), fill="white")
        canvas.create_text(210, 200 - i * 20, text=str(-i), fill="white")

# Crear la ventana principal
```

```
root = tk.Tk()
root.title("Perceptrón")

# Crear un marco para los controles
control_frame = tk.Frame(root, padx=10, pady=10, bg="lightgray")
control_frame.pack(side=tk.RIGHT, fill=tk.Y)

# Crear un canvas para dibujar
canvas = tk.Canvas(root, width=400, height=400, bg="black")
canvas.pack(side=tk.LEFT)

# Dibujar el plano cartesiano
draw_cartesian_plane()

# Crear un perceptrón con pesos y bias iniciales
initial_weights = [1.0, 1.0]
initial_bias = 0
perceptron = Perceptron(initial_weights, initial_bias)

# Escalas para los valores de los pesos y el sesgo
weight1_label = tk.Label(control_frame, text="Peso 1:", bg="lightgray")
weight1_label.pack()
weight1_scale = tk.Scale(control_frame, from_=-10, to=10, orient=tk.HORIZONTAL,
resolution=0.1, command=update_perceptron)
weight1_scale.set(initial_weights[0])
weight1_scale.pack()

weight1_value_label = tk.Label(control_frame, text=f"Peso 1:
{initial_weights[0]}", bg="lightgray")
weight1_value_label.pack()

weight2_label = tk.Label(control_frame, text="Peso 2:", bg="lightgray")
weight2_label.pack()
weight2_scale = tk.Scale(control_frame, from_=-10, to=10, orient=tk.HORIZONTAL,
resolution=0.1, command=update_perceptron)
weight2_scale.set(initial_weights[1])
weight2_scale.pack()

weight2_value_label = tk.Label(control_frame, text=f"Peso 2:
{initial_weights[1]}", bg="lightgray")
weight2_value_label.pack()

bias_label = tk.Label(control_frame, text="Sesgo:", bg="lightgray")
bias_label.pack()
```

```
bias_scale = tk.Scale(control_frame, from_=-10, to=10, orient=tk.HORIZONTAL,
resolution=0.1, command=update_perceptron)
bias_scale.set(initial_bias)
bias_scale.pack()

bias_value_label = tk.Label(control_frame, text=f"Sesgo: {initial_bias}",
bg="lightgray")
bias_value_label.pack()

# Actualizar el perceptrón con los valores iniciales
update_perceptron()

# Botón para limpiar la pantalla
clear_button = tk.Button(control_frame, text="Limpiar Pantalla",
command=clear_canvas)
clear_button.pack(pady=5)

# Botón para restablecer los valores
def reset_values():
    weight1_scale.set(initial_weights[0])
    weight2_scale.set(initial_weights[1])
    bias_scale.set(initial_bias)
    update_perceptron()

reset_button = tk.Button(control_frame, text="Restablecer Valores",
command=reset_values)
reset_button.pack(pady=5)

# Asociar el evento de clic al canvas
canvas.bind("<Button-1>", on_canvas_click)

root.mainloop()
```

Conclusiones:

Padilla Perez Jorge Daray:
Para concluir con esta actividad interesante ya que es nuestra primera neurona por así decirlo, esta se comporta de manera sencilla por así decirlo, al ser nuestro primer programa es fácil relativamente por que la interfaz fue lo mas tedioso, siendo este lo que mas dure en hacer porque el perceptrón ya lo tenia hecho de otra clase.