**Assignment 1. On-Line Signature Data Handling**

**Student ID:** m5281022 **Name:** LAI HUI SHAN

**Programming Language Used:** Python

**Required Libraries:**

* tkinter
* matplotlib

**How to Run the Program:**

1. **Environment Setup:**

* Ensure Python 3.x is installed on your system.
* Install the required libraries using the following commands:

pip install matplotlib

1. **Executing the Script:**

* Save the code in a file named **signature\_visualization.py**.
* Open a command line or terminal window.
* Navigate to the directory where the script is saved.
* Before running the script, ensure you update the **file\_path** variable in the **parse\_sdt\_file** function to your specific local path where your SDT file is located.
* Run the script using the command:

python signature\_visualization.py

**Code Explanation:**

* **draw\_signature:** This function clears the existing canvas and redraws the signature based on the scaled coordinates from the signature data. It uses thick black lines to represent strokes of the signature.
* **draw\_graphs:** This function creates five subplots for the X coordinate, Y coordinate, Pressure, Direction, and Altitude of the signature strokes over time. It adjusts the layout to prevent overlapping of labels and saves the figure to a specified file path.
* **parse\_sdt\_file:** This function reads data from an SDT file, distinguishing strokes by detecting separator lines. It stores strokes in a list, which is used to draw the signature and plot the graphs.
* **main:** The main function initializes the TKinter window, loads signature data from the specified file, and triggers the rendering of the signature and data graphs. It enters the main event loop of TKinter to keep the application running until manually closed.

|  |
| --- |
| **Source Code:** |
| import tkinter as tk  import matplotlib.pyplot as plt  from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  # Main window creation using tkinter  root = tk.Tk()  root.title("Signature Visualization") # Window title  # Add a canvas widget to the main window  canvas = tk.Canvas(root, width=400, height=150)  canvas.pack()  # Function to draw the signature on the canvas  def draw\_signature(canvas, signature\_data):  canvas.delete("all") # Clear the canvas before drawing a new signature  scale = 1.0 # Scale factor for the signature coordinates  line\_width = 2 # Thickness of the signature lines  # Loop through each stroke in the signature data  for stroke in signature\_data:  # Extract the x, y coordinates from the stroke data  points = [(x \* scale, y \* scale) for x, y, \_, \_, \_, \_ in stroke]  # Draw each line segment in the stroke  for i in range(len(points) - 1):  canvas.create\_line(points[i], points[i + 1], fill='black', width=line\_width)  # Function to draw graphs of the signature data  def draw\_graphs(signature\_data):  if not signature\_data:  return # Do nothing if there is no data  # Create 5 subplots for different data attributes: pressure, direction, altitude, x and y coordinates  fig, axs = plt.subplots(5, figsize=(10, 12))  # Labels for the y-axis of each subplot  labels = ['X coordinate', 'Y coordinate', 'Pressure', 'Direction', 'Altitude']  # Indices corresponding to each attribute in the stroke data  indices = [0, 1, 2, 3, 4]  # Create each subplot  for i, (label, index) in enumerate(zip(labels, indices)):  axs[i].set\_ylabel(label) # Set y-axis label  # Set x-axis label only on the last subplot  if i == len(labels) - 1:  axs[i].set\_xlabel('Time')  # Plot the data for each stroke  for stroke in signature\_data:  times = [t for \_, \_, \_, \_, \_, t in stroke] # Extract time points  values = [s[index] for s in stroke] # Extract attribute values based on index  axs[i].plot(times, values) # Plot the attribute values against time  plt.tight\_layout() # Adjust the layout to prevent overlapping of subplots  # Embed the matplotlib figure into the tkinter window  canvas = FigureCanvasTkAgg(fig, master=root)  canvas\_widget = canvas.get\_tk\_widget()  canvas\_widget.pack()  # Save the figure to a file  fig.savefig('C:/Users/HuiShan/Documents/HAPP/Assignment1/Capture/Data1.png')  # Function to parse the SDT file and return the data  def parse\_sdt\_file(file\_path):  data = [] # Initialize an empty list to hold all strokes  current\_stroke = [] # Initialize an empty list for the current stroke  # Open the file and read the lines  with open(file\_path, 'r') as file:  next(file) # Skip the first line as it's a data count  # Iterate over each line in the file  for line in file:  parts = line.strip().split() # Split the line into parts  if parts[0] == '-1': # Stroke separator detected  if current\_stroke: # If the current stroke has data, add it to the list  data.append(current\_stroke)  current\_stroke = [] # Reset current stroke to start a new one  else:  # Parse each part of the line into respective variables  x, y, pressure, direction, altitude, time = map(int, parts)  current\_stroke.append((x, y, pressure, direction, altitude, time))  # Add the last stroke to the list if it's not empty  if current\_stroke:  data.append(current\_stroke)  return data  # Load the SDT file data  signature\_data = parse\_sdt\_file('C:/Users/HuiShan/Documents/HAPP/Assignment1/SignatureSampleData/001.001.000.sdt')  # Check if there is data to display and call the drawing functions  if signature\_data:  draw\_signature(canvas, signature\_data) # Draw the signature on the canvas  draw\_graphs(signature\_data) # Draw the graphs for signature data  else:  print("No data available to display.") # Display message if no data is found  # Start the tkinter main event loop  root.mainloop() |