|  |  |  |
| --- | --- | --- |
| Nulp_logo_ukr | **ЗВО:** Національний університет  «Львівська політехніка»  **Навчальний рік:** 2024/2025  **Семестр:** весняний | **Навчальна дисципліна:** Розробка кросплатформених додатків  **Лабораторна робота № 4:**  Розроблення додатку для малювання графічних об’єктів |
| **Кафедра** систем автоматизованого проектування  **Викладач:**  Юрій ПАТЕРЕГА | **Група:** ПП-26  **Студент:** Ігор Якіб’юк |
| Мета роботи | | |
| Розробити додаток на основі бібліотеки tkinter для малювання вказаних графічних об’єктів.  Додаток повинен містити:   * групу віджетів для введення координат; * групу віджетів для введення параметрів; * групу кнопок для керування малюванням, зокрема, кнопку «Draw» та кнопку «Clear»; * полотно для малювання графічних об’єктів із заданими координатами та параметрами. | | |
| Інструкція до виконання роботи | | |
| 1. Вибрати графічний об’єкт та параметри керування згідно із вашим номером варіанту:   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | Лінія  x1, y1  x2, y2  fill  width  dash | Лінія  x1, y1  x2, y2  fill  arrow  arrowshape | Прямокутник  x1, y1  x2, y2  fill= outline  width  stipple | Овал  x1, y1  x2, y2  fill= outline  width  dash | Дуга  x1, y1  x2, y2  start  extent  style | Полігон  x1, y1  x2, y2  x3, y3  fill=outline  width | Текст  x, y  text  font  fill  angle | Бітмап  x, y  anchor  bitmap  foreground  background | | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | | Лінія  x1, y1  x2, y2  fill  width  dash | Лінія  x1, y1  x2, y2  fill  arrow  arrowshape | Прямокутник  x1, y1  x2, y2  fill= outline  width  stipple | Овал  x1, y1  x2, y2  fill= outline  width  dash | Дуга  x1, y1  x2, y2  start  extent  style | Полігон  x1, y1  x2, y2  x3, y3  width  fill=outline | Текст  x, y  text  font  fill  angle | Бітмап  x, y  anchor  bitmap  foreground  background | | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | | Лінія  x1, y1  x2, y2  fill  width  dash | Лінія  x1, y1  x2, y2  fill  arrow  arrowshape | Прямокутник  x1, y1  x2, y2  fill= outline  width  stipple | Овал  x1, y1  x2, y2  fill= outline  width  dash | Дуга  x1, y1  x2, y2  start  extent  style | Полігон  x1, y1  x2, y2  x3, y3  width  fill=outline | Текст  x, y  text  font  fill  angle | Бітмап  x, y  anchor  bitmap  foreground  background | | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | | Лінія  x1, y1  x2, y2  fill  width  dash | Лінія  x1, y1  x2, y2  fill  arrow  arrowshape | Прямокутник  x1, y1  x2, y2  fill= outline  width  stipple | Овал  x1, y1  x2, y2  fill= outline  width  dash | Дуга  x1, y1  x2, y2  start  extent  style | Полігон  x1, y1  x2, y2  x3, y3  width  fill=outline | Текст  x, y  text  font  fill  angle | Бітмап  x, y  anchor  bitmap  foreground  background |   2. Створити головне вікно та налаштувати його параметри.  3. Додати віджети для введення координат та параметрів.  3.1. Додати віджети для введення координат графічного об’єкта.  3.2. Додати віджети для встановлення параметрів графічного об’єкта.  4. Додати кнопки для керування процесом малювання.  4.1. Додати кнопку «Draw» та обробник для малювання графічного об’єкта.  4.2. Додати кнопку «Clear» та обробник для очищення полотна.  5. Додати полотно (Canvas) для малювання графічних об’єктів.  6. Запустити головний цикл обробки подій (mainloop()). | | |
| 1. Теоретичні відомості | | |
|  | | |
| 2. Лістинг коду програми | | |
| import tkinter as tk  from tkinter import ttk  import math  import numpy as np  import os  import csv  from classify import predict  import joblib  scaler = joblib.load("scaler.pkl")  root = tk.Tk()  root.title("Shape Data Collector")  root.geometry("670x750+350+0")  # Variables  dash\_var = tk.StringVar()  x1\_var = tk.IntVar()  y1\_var = tk.IntVar()  x2\_var = tk.IntVar()  y2\_var = tk.IntVar()  width\_var = tk.IntVar()  fill\_var = tk.StringVar(value="black")  outline\_var = tk.StringVar(value="black")  shape\_type\_var = tk.StringVar(value="line")  # label\_var = tk.StringVar(value="triangle")  frame = tk.Frame(root)  frame.pack()  # Coordinate Inputs  for label\_text, var in [("x1", x1\_var), ("y1", y1\_var), ("x2", x2\_var), ("y2", y2\_var)]:      frame\_coord = tk.LabelFrame(frame, text=label\_text, bd=1, relief="solid")      frame\_coord.pack()      spin = tk.Spinbox(frame\_coord, textvariable=var, from\_=0, to=500, width=5)      spin.pack(fill="x", padx=2, pady=2)  # Width Input  frame\_width = tk.LabelFrame(frame, text="Width:", bd=1, relief="solid")  frame\_width.pack()  scale\_width = tk.Scale(frame\_width, variable=width\_var, from\_=1, to=9, resolution=1,                          tickinterval=2, orient="horizontal", length=100, sliderlength=16)  scale\_width.pack()  # Dash Style  frame\_dash = tk.LabelFrame(frame, text="Dash:", bd=1, relief="solid")  frame\_dash.pack()  combo\_dash = ttk.Combobox(frame\_dash, textvariable=dash\_var, height=5,                          values=("solid", "dashed", "dashdotted", "dotted"), width=10)  combo\_dash.pack()  # Outline Color for Oval  frame\_outline = tk.LabelFrame(frame, text="Outline Color:", bd=1, relief="solid")  frame\_outline.pack()  combo\_outline = ttk.Combobox(frame\_outline, textvariable=outline\_var,                                values=("black", "red", "green", "blue", "yellow"), width=10)  combo\_outline.pack()  # Shape Selection (Line or Oval)  frame\_shape = tk.LabelFrame(frame, text="Shape Type:", bd=1, relief="solid")  frame\_shape.pack()  combo\_shape = ttk.Combobox(frame\_shape, textvariable=shape\_type\_var,                              values=("line", "oval"), width=10)  combo\_shape.pack()  dashes = {"solid": None, "dashed": (20, 20), "dashdotted": (20, 5, 5, 5), "dotted": (5, 5)}  current\_shape\_lines = []  shapes\_dataset = []  # Canvas  frame\_canvas = tk.LabelFrame(root, text="Canvas area", bd=1, relief="solid")  frame\_canvas.pack(fill="both", expand=True)  canvas = tk.Canvas(frame\_canvas, bg="white", width=500, height=500)  canvas.pack(fill="both", expand=True)  # Drawing functions for Line and Oval  def handle\_draw():      x1, y1, x2, y2 = x1\_var.get(), y1\_var.get(), x2\_var.get(), y2\_var.get()      shape\_type = shape\_type\_var.get()      if shape\_type == "line":          canvas.create\_line(x1, y1, x2, y2,                             width=width\_var.get(), fill=fill\_var.get(),                             dash=dashes.get(dash\_var.get()))          current\_shape\_lines.append((x1, y1, x2, y2))      elif shape\_type == "oval":          canvas.create\_oval(x1, y1, x2, y2,                             outline=outline\_var.get(), width=width\_var.get(),                             fill=fill\_var.get(), dash=dashes.get(dash\_var.get()))          current\_shape\_lines.append((x1, y1, x2, y2))  def clear\_canvas():      canvas.delete("all")      current\_shape\_lines.clear()      shapes\_dataset.clear()  def end\_shape():      if not current\_shape\_lines:          return      def shape\_to\_features(lines):          lengths = [math.hypot(x2 - x1, y2 - y1) for x1, y1, x2, y2 in lines]          avg\_length = sum(lengths) / len(lengths)          perimeter = sum(lengths)          return [len(lines), avg\_length, perimeter, max(lengths)]      features = shape\_to\_features(current\_shape\_lines)      x = np.array([features])      x\_scaled = scaler.transform(x)      probs = predict(x\_scaled)      pred\_class = np.argmax(probs)      class\_names = ['triangle', 'square', 'rectangle']      label = class\_names[pred\_class]      shapes\_dataset.append({          "label": label,          "lines": current\_shape\_lines.copy()      })      current\_shape\_lines.clear()      prediction\_result.set(f"You painted: {label}")    # def save\_dataset():  #     def shape\_to\_features(lines):  #         lengths = [math.hypot(x2 - x1, y2 - y1) for x1, y1, x2, y2 in lines]  #         avg\_length = sum(lengths) / len(lengths)  #         perimeter = sum(lengths)  #         return [len(lines), avg\_length, perimeter, max(lengths)]  #     file\_exists = os.path.isfile("shapes\_dataset.csv")  #     with open("shapes\_dataset.csv", "a", newline="") as f:  #         writer = csv.writer(f)  #         if not file\_exists:  #             writer.writerow(["num\_lines", "avg\_length", "perimeter", "max\_length", "label"])  #         for shape in shapes\_dataset:  #             features = shape\_to\_features(shape["lines"])  #             writer.writerow(features + [shape["label"]])  # Prediction Result  prediction\_result = tk.StringVar()  prediction\_result.set("Painted figure: ...")  result\_label = tk.Label(frame, textvariable=prediction\_result, fg="blue", font=("Arial", 12, "bold"))  result\_label.pack(pady=4)  # Mouse drawing  start\_x, start\_y = None, None  def start\_draw(event):      global start\_x, start\_y      start\_x, start\_y = event.x, event.y  def stop\_draw(event):      global start\_x, start\_y      shape\_type = shape\_type\_var.get()      if shape\_type == "line":          canvas.create\_line(start\_x, start\_y, event.x, event.y,                             width=width\_var.get(), fill=fill\_var.get(),                             dash=dashes.get(dash\_var.get()))          current\_shape\_lines.append((start\_x, start\_y, event.x, event.y))      elif shape\_type == "oval":          canvas.create\_oval(start\_x, start\_y, event.x, event.y,                             outline=outline\_var.get(), width=width\_var.get(),                             fill=fill\_var.get(), dash=dashes.get(dash\_var.get()))          current\_shape\_lines.append((start\_x, start\_y, event.x, event.y))  canvas.bind("<ButtonPress-1>", start\_draw)  canvas.bind("<ButtonRelease-1>", stop\_draw)  # Buttons  button\_draw = tk.Button(frame, text="Draw Shape", bg="orange", command=handle\_draw)  button\_draw.pack()  button\_end = tk.Button(frame, text="End Shape", bg="lightblue", command=end\_shape)  button\_end.pack(fill="x", padx=2, pady=2)  button\_clear = tk.Button(frame, text="Clear Canvas", command=clear\_canvas)  button\_clear.pack(fill="x", padx=2, pady=2)  # button\_save = tk.Button(frame, text="Save Dataset", command=save\_dataset)  # button\_save.pack(fill="x", padx=2, pady=2)  # Defaults  x1\_var.set(0)  y1\_var.set(0)  x2\_var.set(100)  y2\_var.set(100)  width\_var.set(1)  dash\_var.set("solid")  root.mainloop() | | |
| 3. Перевірка та тестування програми | | |
|  | | |
| Висновок Висновок має відповісти на запитання «Що зроблено?», «Як зроблено?», «Що це дало?». | | |
| На даній лабораторній роботі, я навчився працювати з полотном Canvas. Ознайомився з функціями добавлення ліній та інших різноманітних фігур. Реалізував індивідуальне завдання: можливість малювати овали по координатам. Як особливість, добавив можливість малювати мишкою. А також як додаткове завдання написав нейронну мережу, як здатна розпізнавати/класифікувати намальовані фігури. Реалізовані 2 окремі файли: файл з нейронною мережею та файл з її навчанням. Виконана робота продемонстрована вище. | | |

**Презентація додаткових завдань:**

Посилання на репозиторій з кодом нейронної мережі - <https://github.com/Igoryakib/neural_network_classify_python.git>

Посилання на презентацію - <https://www.canva.com/design/DAGkFHVDOEw/_lk0ls1BCGwPCorvxeU8XA/edit?utm_content=DAGkFHVDOEw&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton>  
  


