

	ЗВО: Національний університет «Львівська політехніка» Навчальний рік: 2024/2025 Семестр: весняний	Навчальна дисципліна: Розробка кроссплатформених 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

Shape classify

x1

0

y1

0

x2

100

y2

100

Width:

5

1 3 5 7 9

Dash:

solid

Outline Color:

black

Shape Type:

oval

Painted figure: ...

Draw Shape

End Shape

Clear Canvas

Canvas area

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 додавання ліній та інших різноманітних фігур. Реалізував індивідуальне завдання: можливість малювати овали по координатам. Як особливість, додав 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

