import math

import tkinter as tk

def is\_inside\_rectangle(x, y, x1, y1, x2, y2, x3, y3, x4, y4):

# Check if the point (x, y) is inside the rectangle defined by its corners (x1, y1), (x2, y2), (x3, y3), and (x4, y4).

return min(x1, x2, x3, x4) < x < max(x1, x2, x3, x4) and min(y1, y2, y3, y4) < y < max(y1, y2, y3, y4)

def is\_inside\_triangle(x, y, x1, y1, x2, y2, x3, y3):

# Check if the point (x, y) is inside the triangle defined by its corners (x1, y1), (x2, y2), and (x3, y3).

def sign(x1, y1, x2, y2, x3, y3):

return (x1 - x3) \* (y2 - y3) - (x2 - x3) \* (y1 - y3)

b1 = sign(x, y, x1, y1, x2, y2) < 0.0

b2 = sign(x, y, x2, y2, x3, y3) < 0.0

b3 = sign(x, y, x3, y3, x1, y1) < 0.0

return b1 == b2 == b3

def is\_inside\_circle(x, y, cx, cy, radius):

# Check if the point (x, y) is inside the circle with center (cx, cy) and radius.

return math.sqrt((x - cx) \*\* 2 + (y - cy) \*\* 2) < radius

def find\_intersection\_points(shapes):

# Find the intersection points of the given shapes.

result = []

for x in range(-10, 11):

for y in range(-10, 11):

is\_inside\_all\_shapes = True

for shape\_code, coords in shapes:

if shape\_code == 'R':

x1, y1, x2, y2, x3, y3, x4, y4 = coords

if not is\_inside\_rectangle(x, y, x1, y1, x2, y2, x3, y3, x4, y4):

is\_inside\_all\_shapes = False

break

elif shape\_code == 'T':

x1, y1, x2, y2, x3, y3 = coords

if not is\_inside\_triangle(x, y, x1, y1, x2, y2, x3, y3):

is\_inside\_all\_shapes = False

break

elif shape\_code == 'C':

cx, cy, \_, radius = coords

if not is\_inside\_circle(x, y, cx, cy, radius):

is\_inside\_all\_shapes = False

break

if is\_inside\_all\_shapes:

result.append((x, y))

return result

def scale\_coordinate(coord):

return (coord + 10) \* 20 + 100

def draw\_shapes(canvas, shapes):

canvas.delete("shape") # Clear the previous shapes

for i, (shape\_code, coords) in enumerate(shapes):

color = get\_color(i)

if shape\_code == 'R':

x1, y1, x2, y2, x3, y3, x4, y4 = coords

x1\_scaled, y1\_scaled = scale\_coordinate(x1), scale\_coordinate(y1)

x2\_scaled, y2\_scaled = scale\_coordinate(x2), scale\_coordinate(y2)

x3\_scaled, y3\_scaled = scale\_coordinate(x3), scale\_coordinate(y3)

x4\_scaled, y4\_scaled = scale\_coordinate(x4), scale\_coordinate(y4)

canvas.create\_polygon(x1\_scaled, y1\_scaled, x2\_scaled, y2\_scaled, x3\_scaled, y3\_scaled, x4\_scaled, y4\_scaled, outline=color, width=2, tags="shape", fill="")

#canvas.create\_text(x1\_scaled, y1\_scaled, text=f"({x1}, {y1})", anchor='se', fill=color, tags="shape")

elif shape\_code == 'T':

x1, y1, x2, y2, x3, y3 = coords

x1\_scaled, y1\_scaled = scale\_coordinate(x1), scale\_coordinate(y1)

x2\_scaled, y2\_scaled = scale\_coordinate(x2), scale\_coordinate(y2)

x3\_scaled, y3\_scaled = scale\_coordinate(x3), scale\_coordinate(y3)

canvas.create\_polygon(x1\_scaled, y1\_scaled, x2\_scaled, y2\_scaled, x3\_scaled, y3\_scaled, outline=color, width=2, tags="shape", fill="")

#canvas.create\_text(x1\_scaled, y1\_scaled, text=f"({x1}, {y1})", anchor='se', fill=color, tags="shape")

elif shape\_code == 'C':

cx, cy, \_, radius = coords

cx\_scaled, cy\_scaled = scale\_coordinate(cx), scale\_coordinate(cy)

radius\_scaled = radius \* 40

canvas.create\_oval(cx\_scaled - radius\_scaled, cy\_scaled - radius\_scaled, cx\_scaled + radius\_scaled, cy\_scaled + radius\_scaled, outline=color, width=2, tags="shape", fill="")

#canvas.create\_text(cx\_scaled, cy\_scaled, text=f"({cx}, {cy})", anchor='se', fill=color, tags="shape")

def get\_color(index):

colors = ['blue', 'red', 'green', 'orange', 'purple', 'brown', 'cyan', 'magenta']

return colors[index % len(colors)]

def add\_shape\_frame():

new\_frame = ShapeFrame(frame)

new\_frame.pack(pady=4)

shape\_frames.append(new\_frame)

class ShapeFrame(tk.Frame):

def \_\_init\_\_(self, master):

super().\_\_init\_\_(master)

self.shape\_var = tk.StringVar(self)

self.shape\_var.set('R')

self.create\_widgets()

def create\_widgets(self):

shape\_label = tk.Label(self, text="Shape (R, T, C):")

shape\_label.grid(row=0, column=0)

#shape\_label.pack(side = "top", anchor = "nw")

shape\_menu = tk.OptionMenu(self, self.shape\_var, 'R', 'T', 'C')

shape\_menu.grid(row=0, column=1)

coords\_label = tk.Label(self, text="Coordinates:")

coords\_label.grid(row=0, column=2)

self.coords\_entry = tk.Entry(self)

self.coords\_entry.grid(row=0, column=3)

def show\_results():

shapes = []

for shape\_frame in shape\_frames:

shape\_code = shape\_frame.shape\_var.get()

coords = shape\_frame.coords\_entry.get().split()

coords = list(map(int, coords))

shapes.append((shape\_code, coords))

intersection\_points = find\_intersection\_points(shapes)

draw\_shapes(canvas, shapes)

result\_label.config(text="The intersection points are: ")

if intersection\_points:

for x, y in intersection\_points:

x\_scaled, y\_scaled = scale\_coordinate(x), scale\_coordinate(y)

canvas.create\_text(x\_scaled, y\_scaled, text="\*", fill="red", tags="shape")

result\_label.config(text=result\_label.cget("text") + f"({x}, {y}), ")

# Create the main window

window = tk.Tk()

window.title("Geometric Figures Intersection")

# Create a frame for shape inputs

frame = tk.Frame(window)

frame.pack(pady=10)

# Create buttons

add\_btn = tk.Button(window, text="Add Shape", command=add\_shape\_frame)

add\_btn.pack(side=tk.LEFT, padx=5)

find\_points\_btn = tk.Button(window, text="Find Intersection Points", command=show\_results)

find\_points\_btn.pack(side=tk.LEFT,padx=10)

# Create a canvas for drawing shapes

canvas = tk.Canvas(window, width=800, height=550, bg="white")

canvas.pack()

# Create an initial shape frame

shape\_frames = []

add\_shape\_frame()

# Create buttons

#add\_btn = tk.Button(window, text="Add Shape", command=add\_shape\_frame)

#add\_btn.pack(side=tk.LEFT, padx=5)

#find\_points\_btn = tk.Button(window, text="Find Intersection Points", command=show\_results)

#find\_points\_btn.pack(side=tk.LEFT, padx=5)

result\_label = tk.Label(window, text="The points in the intersection area are: ", font=("Helvetica", 12, "bold"))

result\_label.place(relx = 0.0, rely = 1.0, anchor = 'sw')

#result\_label.grid(row = 1, column = 0)

# Start the main event loop

window.mainloop()