МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчёт по лабораторной работе №7

Специальность ПО11

Выполнил
Е. А. Германович
студент группы ПО11

Проверил
А. А. Крощенко
ст. преп. кафедры ИИТ,
22.02.2025 г.

Брест 2025

Цель работы: освоить возможности языка программирования Python в разработке оконных приложений.

Задание 1. Построение графических примитивов и надписей

Код программы:

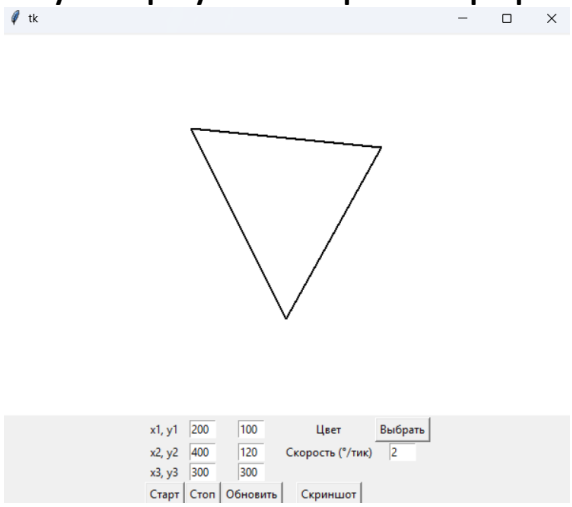
```
import tkinter as tk
from tkinter import colorchooser
from PIL import ImageGrab
import math


class Primitive:
    def draw(self, canvas):
        pass


class Line(Primitive):
    def __init__(self, x1, y1, x2, y2, color):
        self.x1, self.y1, self.x2, self.y2, self.color = x1, y1, x2, y2, color

    def draw(self, canvas):
        canvas.create_line(self.x1, self.y1, self.x2, self.y2, fill=self.color)


class Triangle(Primitive):
    def __init__(self, points, color, angle=0, speed=1):
        self.points = points  # [(x1, y1), (x2, y2), (x3, y3)]
        self.color = color
        self.angle = angle  # текущий угол вращения в градусах
        self.speed = speed  # скорость вращения (градусов за тик)
        self._calc_center()

    def _calc_center(self):
        x = sum(p[0] for p in self.points) / 3
        y = sum(p[1] for p in self.points) / 3
        self.center = (x, y)

    def rotate(self):
        self.angle = (self.angle + self.speed) % 360
        rad = math.radians(self.speed)
        cx, cy = self.center
        new_points = []
        for x, y in self.points:
            # Переводим в систему координат центра
            dx, dy = x - cx, y - cy
            # Вращаем
            new_x = dx * math.cos(rad) - dy * math.sin(rad)
            new_y = dx * math.sin(rad) + dy * math.cos(rad)
            # Возвращаем обратно
            new_points.append((new_x + cx, new_y + cy))
        self.points = new_points

    def draw(self, canvas):
        flat_points = [coord for point in self.points for coord in point]
        canvas.create_polygon(flat_points, fill="", outline=self.color, width=2)


# Аналогично Rectangle, Ellipse, Text
```

```
class App:
    def __init__(self, root):
        self.root = root
        self.canvas = tk.Canvas(root, width=600, height=400, bg="white")
        self.canvas.pack()

        # Элементы управления
        frame = tk.Frame(root)
        frame.pack()
        tk.Label(frame, text="x1, y1").grid(row=0, column=0)
        tk.Label(frame, text="x2, y2").grid(row=1, column=0)
        tk.Label(frame, text="x3, y3").grid(row=2, column=0)
        self.x1 = tk.Entry(frame, width=4)
        self.y1 = tk.Entry(frame, width=4)
        self.x2 = tk.Entry(frame, width=4)
        self.y2 = tk.Entry(frame, width=4)
        self.x3 = tk.Entry(frame, width=4)
        self.y3 = tk.Entry(frame, width=4)
        self.x1.grid(row=0, column=1)
        self.y1.grid(row=0, column=2)
        self.x2.grid(row=1, column=1)
        self.y2.grid(row=1, column=2)
        self.x3.grid(row=2, column=1)
        self.y3.grid(row=2, column=2)

        tk.Label(frame, text="Цвет").grid(row=0, column=3)
        self.color_btn = tk.Button(frame, text="Выбрать", command=self.choose_color)
        self.color_btn.grid(row=0, column=4)
        self.color = "#000000"

        tk.Label(frame, text="Скорость (°/тик)").grid(row=1, column=3)
        self.speed_entry = tk.Entry(frame, width=4)
        self.speed_entry.insert(0, "2")
        self.speed_entry.grid(row=1, column=4)

        self.start_btn = tk.Button(frame, text="Старт", command=self.start_anim)
        self.start_btn.grid(row=3, column=0)
        self.stop_btn = tk.Button(frame, text="Стоп", command=self.stop_anim)
        self.stop_btn.grid(row=3, column=1)
        self.update_btn = tk.Button(frame, text="Обновить", command=self.update_triangle)
        self.update_btn.grid(row=3, column=2)
        self.screenshot_btn = tk.Button(frame, text="Скриншот", command=self.screenshot)
        self.screenshot_btn.grid(row=3, column=3)

        # Переменные для анимации
        self.triangle = None
        self.animating = False
        self.anim_id = None
        self.set_default_triangle()
        self.draw()

    def set_default_triangle(self):
        # Значения по умолчанию
        self.x1.delete(0, tk.END)
        self.x1.insert(0, "200")
        self.y1.delete(0, tk.END)
        self.y1.insert(0, "100")
```

```
        self.x2.delete(0, tk.END)
        self.x2.insert(0, "400")
        self.y2.delete(0, tk.END)
        self.y2.insert(0, "120")
        self.x3.delete(0, tk.END)
        self.x3.insert(0, "300")
        self.y3.delete(0, tk.END)
        self.y3.insert(0, "300")
        self.update_triangle()

    def choose_color(self):
        color = colorchooser.askcolor(title="Выберите цвет")
        if color[1]:
            self.color = color[1]
            self.color_btn.config(bg=self.color)
            self.update_triangle()

    def update_triangle(self):
        try:
            points = [
                (float(self.x1.get()), float(self.y1.get())),
                (float(self.x2.get()), float(self.y2.get())),
                (float(self.x3.get()), float(self.y3.get())),
            ]
            speed = float(self.speed_entry.get())
            self.triangle = Triangle(points, self.color, speed=speed)
            self.draw()
        except Exception as e:
            print("Ошибка параметров:", e)

    def draw(self):
        self.canvas.delete("all")
        if self.triangle:
            self.triangle.draw(self.canvas)

    def animate(self):
        if self.animating and self.triangle:
            self.triangle.rotate()
            self.draw()
            self.anim_id = self.root.after(30, self.animate)

    def start_anim(self):
        self.animating = True
        self.animate()

    def stop_anim(self):
        self.animating = False
        if self.anim_id:
            self.root.after_cancel(self.anim_id)
            self.anim_id = None

    def screenshot(self):
        x = self.root.winfo_rootx() + self.canvas.winfo_x()
        y = self.root.winfo_rooty() + self.canvas.winfo_y()
        x1 = x + self.canvas.winfo_width()
        y1 = y + self.canvas.winfo_height()
        ImageGrab.grab().crop((x, y, x1, y1)).save("screenshot.png")
```

```
root = tk.Tk()
app = App(root)
root.mainloop()
```

Рисунки с результатами работы программы:



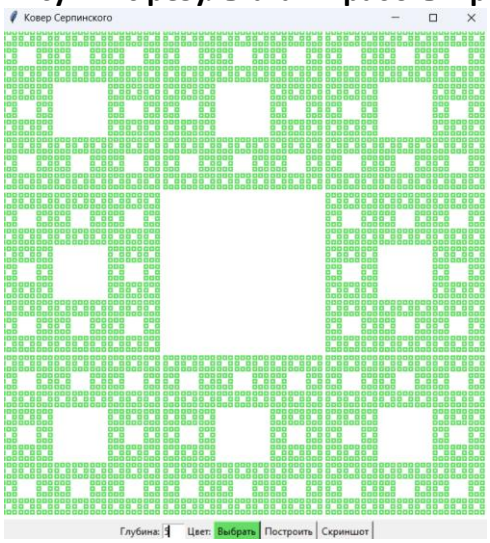
Задание 2. Реализовать построение заданного типа фрактала по варианту
Ковер Серпинского

Код программы:

```
import tkinter as tk
from tkinter import colorchooser
from PIL import ImageGrab


class SierpinskiCarpetApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Ковер Серпинского")
        self.canvas_size = 600
        self.canvas = tk.Canvas(root, width=self.canvas_size, height=self.canvas_size, bg="white")
        self.canvas.pack()

        frame = tk.Frame(root)
        frame.pack()
        tk.Label(frame, text="Глубина:").grid(row=0, column=0)
        self.depth_entry = tk.Entry(frame, width=4)
        self.depth_entry.insert(0, "3")
        self.depth_entry.grid(row=0, column=1)

        tk.Label(frame, text="Цвет:").grid(row=0, column=2)
        self.color_btn = tk.Button(frame, text="Выбрать", command=self.choose_color)
        self.color_btn.grid(row=0, column=3)
        self.color = "#0000FF"

        self.draw_btn = tk.Button(frame, text="Построить", command=self.draw_carpet)
        self.draw_btn.grid(row=0, column=4)
        self.screenshot_btn = tk.Button(frame, text="Скриншот", command=self.screenshot)
        self.screenshot_btn.grid(row=0, column=5)

    def choose_color(self):
        color = colorchooser.askcolor(title="Выберите цвет")
        if color[1]:
            self.color = color[1]
            self.color_btn.config(bg=self.color)
```

```
    def draw_carpet(self):
        try:
            depth = int(self.depth_entry.get())
            self.canvas.delete("all")
            self._draw_carpet(0, 0, self.canvas_size, depth)
        except Exception as e:
            print("Ошибка:", e)

    def _draw_carpet(self, x, y, size, depth):
        if depth == 0:
            self.canvas.create_rectangle(x, y, x + size, y + size, fill=self.color, outline="")
        else:
            new_size = size // 3
            for dx in range(3):
                for dy in range(3):
                    if dx == 1 and dy == 1:
                        continue  # центр не закрашиваем
                    self._draw_carpet(x + dx * new_size, y + dy * new_size, new_size, depth - 1)

    def screenshot(self):
        x = self.root.winfo_rootx() + self.canvas.winfo_x()
        y = self.root.winfo_rooty() + self.canvas.winfo_y()
        x1 = x + self.canvas.winfo_width()
        y1 = y + self.canvas.winfo_height()
        ImageGrab.grab().crop((x, y, x1, y1)).save("sierpinski_carpet.png")


if __name__ == "__main__":
    root = tk.Tk()
    app = SierpinskiCarpetApp(root)
    root.mainloop()
```

Рисунки с результатами работы программы:



Вывод: освоил возможности языка программирования Python в разработке оконных приложений.