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()

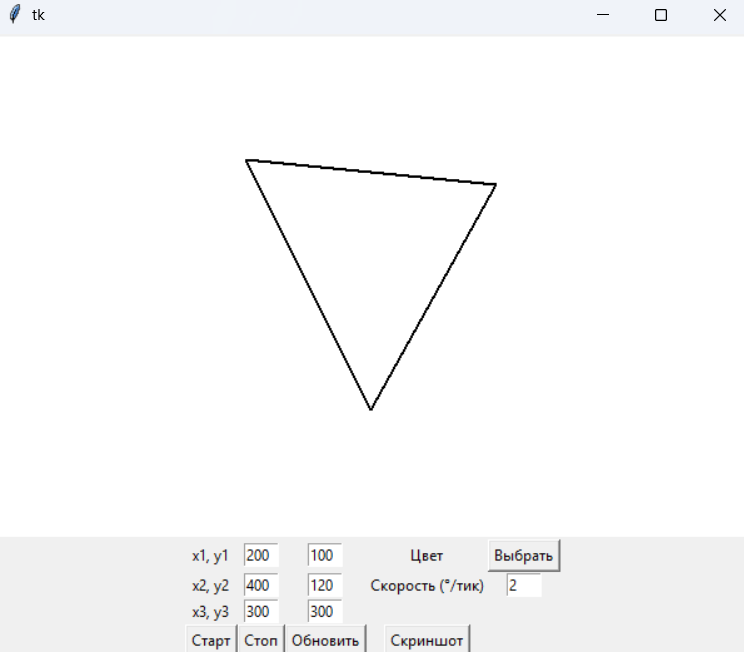
ImageGrab.grab().crop((x, y, x1, y1)).save("screenshot.png")

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")

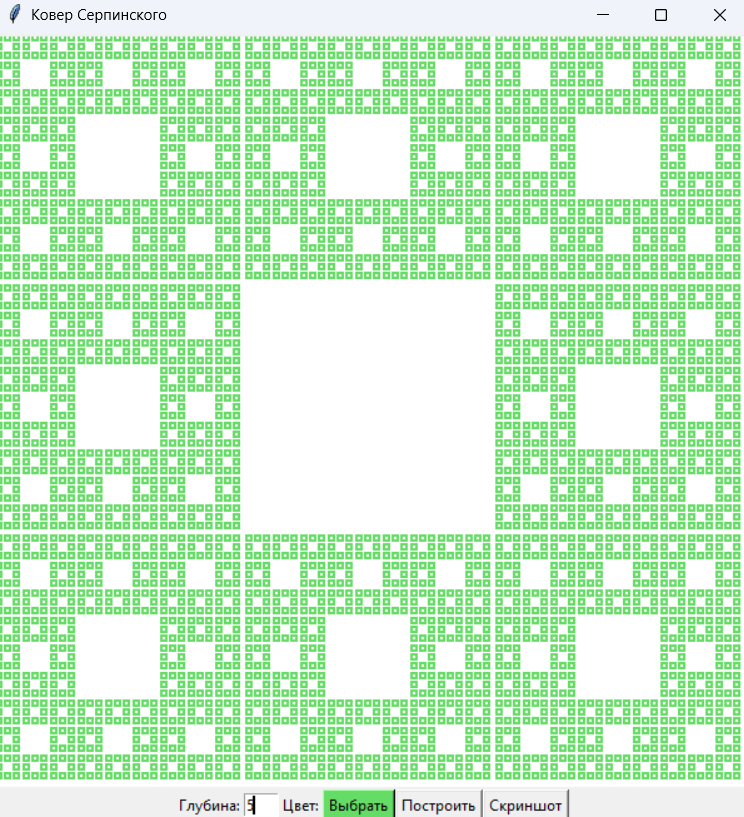
if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = SierpinskiCarpetApp(root)

root.mainloop()

**Рисунки с результатами работы программы:**

****

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