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
С. В. Зайченко
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

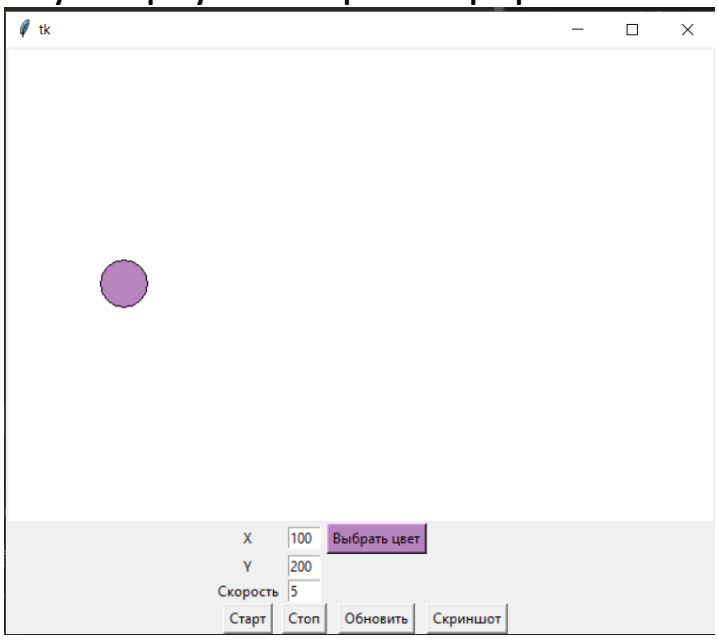
Код программы:

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


class Ball:
    def __init__(self, radius, speed, color, x, y):
        self.radius = radius
        self.speed = speed
        self.color = color
        self.x = x
        self.y = y
        self.direction = 1

    def move(self):
        self.x += self.speed * self.direction
        if self.x - self.radius < 0 or self.x + self.radius > 600:
            self.direction *= -1

    def draw(self, canvas):
        canvas.create_oval(self.x - self.radius, self.y - self.radius,
                           self.x + self.radius, self.y + self.radius,
                           fill=self.color, outline="black")


class App:
    def __init__(self, root):
        self.root = root
        self.color = "#000000"
        self.canvas = tk.Canvas(self.root, width=600, height=400, bg="white")
        self.canvas.pack()
        self._create_controls()
        self.ball = None
        self.animating = False
        self.anim_id = None
        self.set_default_ball()
        self.draw()

    def _create_controls(self):
        frame = tk.Frame(self.root)
        frame.pack()

        tk.Label(frame, text="X").grid(row=0, column=0)
        tk.Label(frame, text="Y").grid(row=1, column=0)
        tk.Label(frame, text="Скорость").grid(row=2, column=0)

        self.x_entry = tk.Entry(frame, width=4)
        self.y_entry = tk.Entry(frame, width=4)
        self.speed_entry = tk.Entry(frame, width=4)

        self.x_entry.grid(row=0, column=1)
        self.y_entry.grid(row=1, column=1)
        self.speed_entry.grid(row=2, column=1)
```

```
        self.color_btn = tk.Button(frame, text="Выбрать цвет", command=self.choose_color)
        self.color_btn.grid(row=0, column=2)

        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_ball)
        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)

    def set_default_ball(self):
        self.x_entry.delete(0, tk.END)
        self.x_entry.insert(0, "100")
        self.y_entry.delete(0, tk.END)
        self.y_entry.insert(0, "200")
        self.speed_entry.delete(0, tk.END)
        self.speed_entry.insert(0, "5")
        self.update_ball()

    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_ball()

    def update_ball(self):
        try:
            x = float(self.x_entry.get())
            y = float(self.y_entry.get())
            speed = float(self.speed_entry.get())
            self.ball = Ball(radius=20, speed=speed, color=self.color, x=x, y=y)
            self.draw()
        except ValueError as e:
            print("Ошибка параметров:", e)

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

    def animate(self):
        if self.animating and self.ball:
            self.ball.move()
            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")
        print("Скриншот сохранен!")


if __name__ == "__main__":
    tk_main = tk.Tk()
    app = App(tk_main)
    tk_main.mainloop()
```

Рисунки с результатами работы программы:



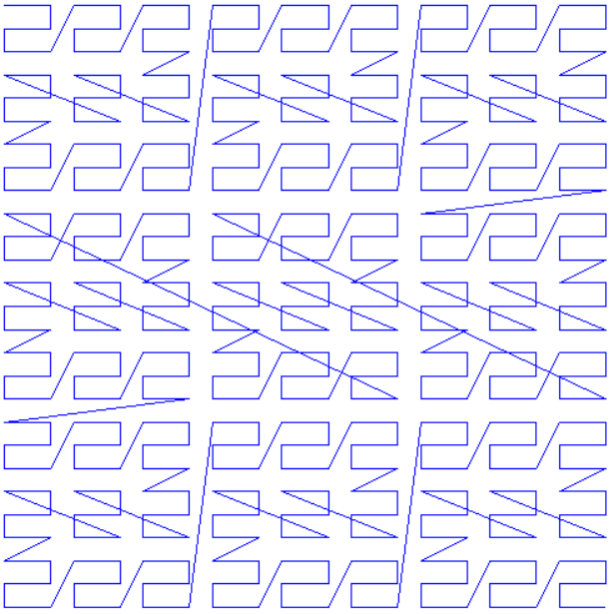
Задание 2. Реализовать построение заданного типа фрактала по варианту

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


class PeanoCurveApp:
    def __init__(self, tk_root):
        self.root = tk_root
        self.root.title("Кривая Пеано")
        self.canvas_size = 600
        self.canvas = tk.Canvas(self.root, width=self.canvas_size, height=self.canvas_size, bg="white")
        self.canvas.pack()

        frame = tk.Frame(self.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, "4")
    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_curve)
    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_curve(self):
    try:
        depth = int(self.depth_entry.get())
        self.canvas.delete("all")
        self._draw_peano_curve(50, 50, self.canvas_size - 100, depth)
    except ValueError as e:
        print("Ошибка:", e)

def _draw_peano_curve(self, x, y, size, depth):
    if depth == 0:
        return

    length = size / 3

    self._draw_peano_curve_segment(x, y, length, 0, depth)
    self._draw_peano_curve_segment(x + length, y, length, 90, depth)
    self._draw_peano_curve_segment(x + length, y + length, length, 180, depth)
    self._draw_peano_curve_segment(x, y + length, length, 270, depth)

def _draw_peano_curve_segment(self, x, y, size, angle, depth):
    if depth == 0:
        dx = size * math.cos(math.radians(angle))
        dy = size * math.sin(math.radians(angle))
        x2 = x + dx
        y2 = y + dy
        self.canvas.create_line(x, y, x2, y2, fill=self.color)
        return x2, y2
    else:
```

```
            length = size / 3
            self._draw_peano_curve_segment(x, y, length, angle, depth - 1)
            self._draw_peano_curve_segment(x + length, y, length, angle + 90, depth - 1)
            self._draw_peano_curve_segment(x + length, y + length, length, angle + 180, depth - 1)
            self._draw_peano_curve_segment(x, y + length, length, angle + 270, 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("peano_curve.png")


if __name__ == "__main__":
    tk_main = tk.Tk()
    app = PeanoCurveApp(tk_main)
    tk_main.mainloop()
```

Рисунки с результатами работы программы:



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