Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №7

По дисциплине:“ Современные платформы программирования”

Выполнил:
Студент 3 курса
Группы ПО-11
Сологуб А.В.
Проверил:
Козик И.Д.

Брест 2025

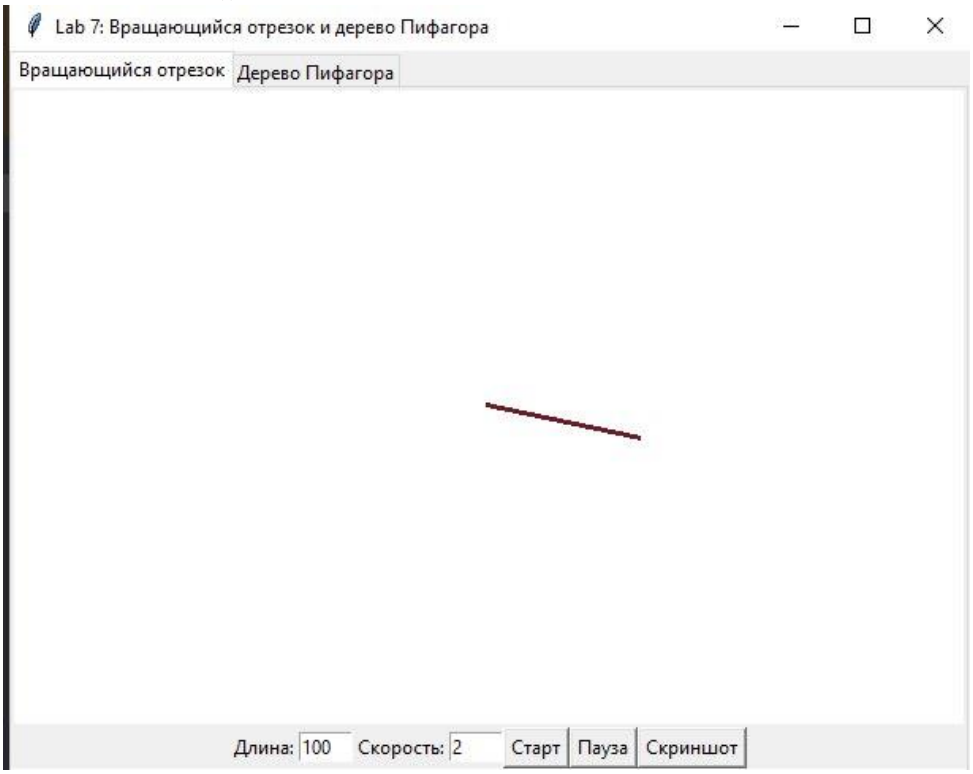
Цель: освоить возможности языка программирования Python в разработке оконных приложений
Задание: Задание 1. Построение графических примитивов и надписей
Требования к выполнению
• Реализовать соответствующие классы, указанные в задании;
• Организовать ввод параметров для создания объектов (использовать
экранные компоненты);
• Осуществить визуализацию графических примитивов
5**) Изобразить в окне приложения отрезок, вращающийся в плоскости экрана вокруг одной из своих концевых точек.** Цвет прямой должен изменяться при
переходе от одного положения к другому.
Задание 2. Реализовать построение заданного типа фрактала по варианту
5) Дерево Пифагора

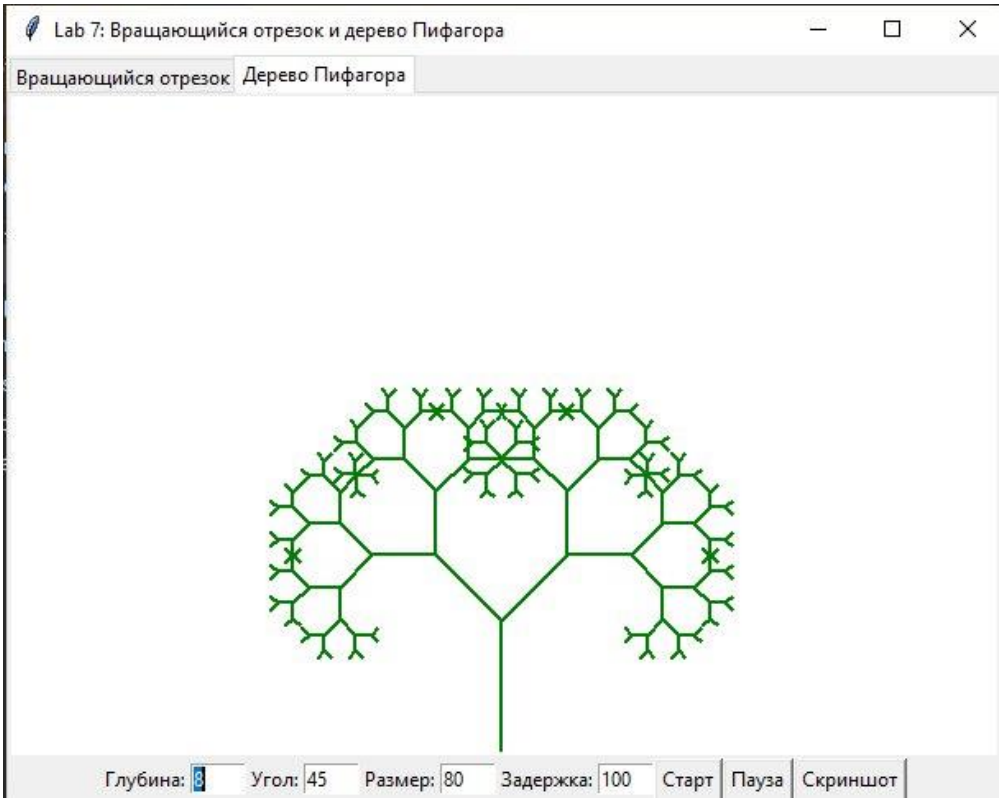
```
import tkinter as tk
from tkinter import ttk
import math
import time
from PIL import Image
import PIL.EpsImagePlugin

PIL.EpsImagePlugin.gs_windows_binary = r"C:\Program Files\gs\gs10.05.0\bin\gswin64c.exe"

class RotatingLineCanvas(tk.Canvas):
    def __init__(self, master, length=100, speed=1, **kwargs):
        super().__init__(master, **kwargs)
        self.length = length
        self.angle = 0
        self.speed = speed
        self.running = True
        self.line = None
        self.update_animation()

    def update_animation(self):
        if self.running:
            self.angle += self.speed
            self.angle %= 360
            self.draw_line()
        self.after(20, self.update_animation)

    def draw_line(self):
        self.delete("all")
        rad = math.radians(self.angle)
        x1, y1 = 200, 200
        x2 = x1 + self.length * math.cos(rad)
        y2 = y1 + self.length * math.sin(rad)
        color = "#%02x%02x%02x" % (
            int(127 + 127 * math.sin(rad)),
            int(127 + 127 * math.sin(rad + 2)),
            int(127 + 127 * math.sin(rad + 4)),
        )
        self.create_line(x1, y1, x2, y2, fill=color, width=4)

    def toggle(self):
        self.running = not self.running

    def set_speed(self, speed):
        self.speed = speed
```

```
    def set_length(self, length):
        self.length = length

    def screenshot(self):
        ps_file = "rotating_line_screenshot.ps"
        png_file = "rotating_line_screenshot.png"
        self.postscript(file=ps_file, colormode='color')
        img = Image.open(ps_file)
        img.save(png_file, "png")
        print(f"Скриншот сохранён: {png_file}")


class PythagorasTreeCanvas(tk.Canvas):
    def __init__(self, master, size=80, depth=7, **kwargs):
        super().__init__(master, **kwargs)
        self.size = size
        self.depth = depth
        self.running = True
        self.draw()

    def draw(self):
        self.delete("all")
        self._draw_tree(300, 500, -90, self.depth, self.size)

    def _draw_tree(self, x, y, angle, depth, length):
        if depth == 0:
            return
        x1 = x + math.cos(math.radians(angle)) * length
        y1 = y + math.sin(math.radians(angle)) * length
        self.create_line(x, y, x1, y1, fill="green", width=2)
        self._draw_tree(x1, y1, angle - 30, depth - 1, length * 0.7)
        self._draw_tree(x1, y1, angle + 30, depth - 1, length * 0.7)

    def set_size(self, size):
        self.size = size
        self.draw()

    def set_depth(self, depth):
        self.depth = depth
        self.draw()

    def screenshot(self):
        ps_file = "pythagoras_tree_screenshot.ps"
        png_file = "pythagoras_tree_screenshot.png"
        self.postscript(file=ps_file, colormode='color')
        img = Image.open(ps_file)
        img.save(png_file, "png")
        print(f"Скриншот сохранён: {png_file}")


class Lab7App(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Лабораторная работа №7")
        self.geometry("1000x600")

        # Вращающийся отрезок
        frame1 = tk.LabelFrame(self, text="Задание 1: Вращающийся отрезок")
        frame1.pack(side=tk.LEFT, fill=tk.BOTH, expand=True, padx=10, pady=10)

        self.line_canvas = RotatingLineCanvas(frame1, bg="white", width=400, height=400)
        self.line_canvas.pack()
```

```
        control1 = tk.Frame(frame1)
        control1.pack()

        tk.Label(control1, text="Длина:").grid(row=0, column=0)
        length_entry = tk.Entry(control1)
        length_entry.insert(0, "100")
        length_entry.grid(row=0, column=1)

        tk.Label(control1, text="Скорость:").grid(row=1, column=0)
        speed_entry = tk.Entry(control1)
        speed_entry.insert(0, "2")
        speed_entry.grid(row=1, column=1)

        tk.Button(control1, text="Применить", command=lambda: self.update_line(length_entry,
speed_entry)).grid(row=2, column=0)
        tk.Button(control1, text="Пауза", command=self.line_canvas.toggle).grid(row=2, column=1)
        tk.Button(control1, text="Скриншот", command=self.line_canvas.screenshot).grid(row=2, column=2)

        # Дерево Пифагора
        frame2 = tk.LabelFrame(self, text="Задание 2: Дерево Пифагора")
        frame2.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True, padx=10, pady=10)

        self.tree_canvas = PythagorasTreeCanvas(frame2, bg="white", width=400, height=400)
        self.tree_canvas.pack()

        control2 = tk.Frame(frame2)
        control2.pack()

        tk.Label(control2, text="Размер:").grid(row=0, column=0)
        size_entry = tk.Entry(control2)
        size_entry.insert(0, "80")
        size_entry.grid(row=0, column=1)

        tk.Label(control2, text="Глубина:").grid(row=1, column=0)
        depth_entry = tk.Entry(control2)
        depth_entry.insert(0, "7")
        depth_entry.grid(row=1, column=1)

        tk.Button(control2, text="Применить", command=lambda: self.update_tree(size_entry,
depth_entry)).grid(row=2, column=0)
        tk.Button(control2, text="Скриншот", command=self.tree_canvas.screenshot).grid(row=2, column=1)

    def update_line(self, length_entry, speed_entry):
        try:
            length = float(length_entry.get())
            speed = float(speed_entry.get())
            self.line_canvas.set_length(length)
            self.line_canvas.set_speed(speed)
        except ValueError:
            print("Введите корректные числа для длины и скорости")

    def update_tree(self, size_entry, depth_entry):
        try:
            size = float(size_entry.get())
            depth = int(depth_entry.get())
            self.tree_canvas.set_size(size)
            self.tree_canvas.set_depth(depth)
        except ValueError:
            print("Введите корректные значения для размера и глубины")
```

```
if __name__ == "__main__":
    app = Lab7App()
    app.mainloop()
```





Вывод: освоила возможности языка программирования Python в разработке оконных приложений