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
П. А. Захарчук
студент группы ПО11

Проверил
А. А. Крощенко
ст. преп. кафедры ИИТ,
25.04.2025 г.

Брест 2025

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

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
Важное замечание: должна быть предусмотрена возможность приостановки выполнения визуализации, изменения параметров «на лету» и снятия скриншотов с сохранением в текущую активную директорию.
Для всех динамических сцен необходимо задавать параметр скорости!

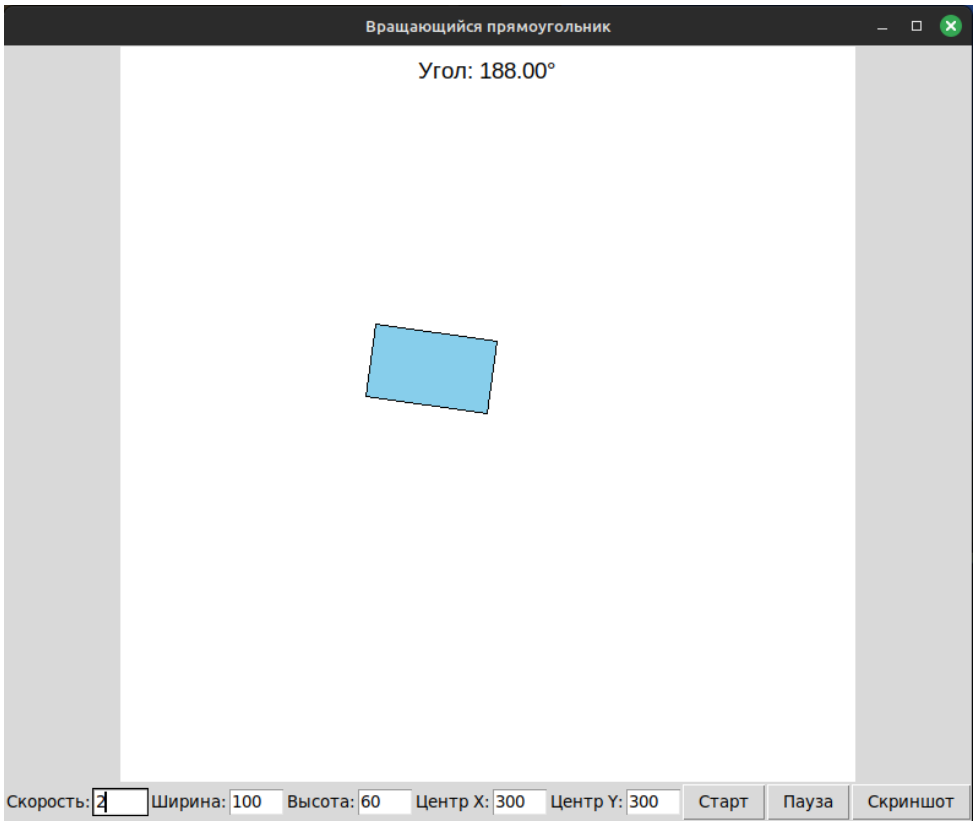
Код программы:

```
    import tkinter as tk
from math import cos, sin, radians
from PIL import ImageGrab
from datetime import datetime



class RotatingRectangle:
    def __init__(self, cx, cy, width, height):
        self.cx = cx
        self.cy = cy
        self.width = width
        self.height = height
        self.angle = 0

    def set_params(self, cx, cy, width, height):
        self.cx = cx
        self.cy = cy
        self.width = width
        self.height = height

    def draw(self, canvas):
        angle_rad = radians(self.angle)

        x0, y0 = self.cx, self.cy

        x1 = x0 + self.width * cos(angle_rad)
        y1 = y0 + self.width * sin(angle_rad)

        x2 = x1 - self.height * sin(angle_rad)
        y2 = y1 + self.height * cos(angle_rad)

        x3 = x0 - self.height * sin(angle_rad)
        y3 = y0 + self.height * cos(angle_rad)

        canvas.create_polygon(x0, y0, x1, y1, x2, y2, x3, y3, fill="skyblue", outline="black")
        canvas.create_text(300, 20, text=f"Угол: {self.angle:.2f}°", font=("Arial", 14))



class RotatingRectangleApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Вращающийся прямоугольник")
```

```
        self.canvas = tk.Canvas(root, width=600, height=600, bg="white")
        self.canvas.pack()

        # Панель управления
        control_frame = tk.Frame(root)
        control_frame.pack()

        # Поля ввода
        self.entries = {}
        for label_text in ["Скорость", "Ширина", "Высота", "Центр X", "Центр Y"]:
            tk.Label(control_frame, text=label_text + ":").pack(side=tk.LEFT)
            entry = tk.Entry(control_frame, width=5)
            entry.pack(side=tk.LEFT)
            self.entries[label_text] = entry

        # Значения по умолчанию
        self.entries["Скорость"].insert(0, "2")
        self.entries["Ширина"].insert(0, "100")
        self.entries["Высота"].insert(0, "60")
        self.entries["Центр X"].insert(0, "300")
        self.entries["Центр Y"].insert(0, "300")

        # Кнопки
        tk.Button(control_frame, text="Старт", command=self.start).pack(side=tk.LEFT)
        tk.Button(control_frame, text="Пауза", command=self.pause).pack(side=tk.LEFT)
        tk.Button(control_frame, text="Скриншот", command=self.take_screenshot).pack(side=tk.LEFT)

        self.is_running = False
        self.rect = RotatingRectangle(300, 300, 100, 60)

        self.animate()

    def animate(self):
        if self.is_running:
            # Чтение параметров
            try:
                cx = float(self.entries["Центр X"].get())
                cy = float(self.entries["Центр Y"].get())
                width = float(self.entries["Ширина"].get())
                height = float(self.entries["Высота"].get())
                self.rect.set_params(cx, cy, width, height)
            except ValueError:
                pass  # оставляем старые параметры, если ввод неверен

            speed_text = self.entries["Скорость"].get()
            if speed_text.strip():
                try:
                    speed = float(speed_text)
                    self.rect.angle = (self.rect.angle + speed) % 360
                except ValueError:
                    pass  # игнорируем ошибку

            self.draw_scene()

        self.root.after(20, self.animate)

    def draw_scene(self):
        self.canvas.delete("all")
        self.rect.draw(self.canvas)

    def start(self):
        self.is_running = True
```

```
    def pause(self):
        self.is_running = False

    def take_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()

        screenshot = ImageGrab.grab().crop((x, y, x1, y1))
        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        filename = f"screenshot_{timestamp}.png"
        screenshot.save(filename)
        print(f"Скриншот сохранён как {filename}")


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

Рисунки с результатами работы программы:



Задание 2: Реализовать построение заданного типа фрактала по варианту

Везде, где это необходимо, предусмотреть ввод параметров, влияющих на внешний вид фрактала

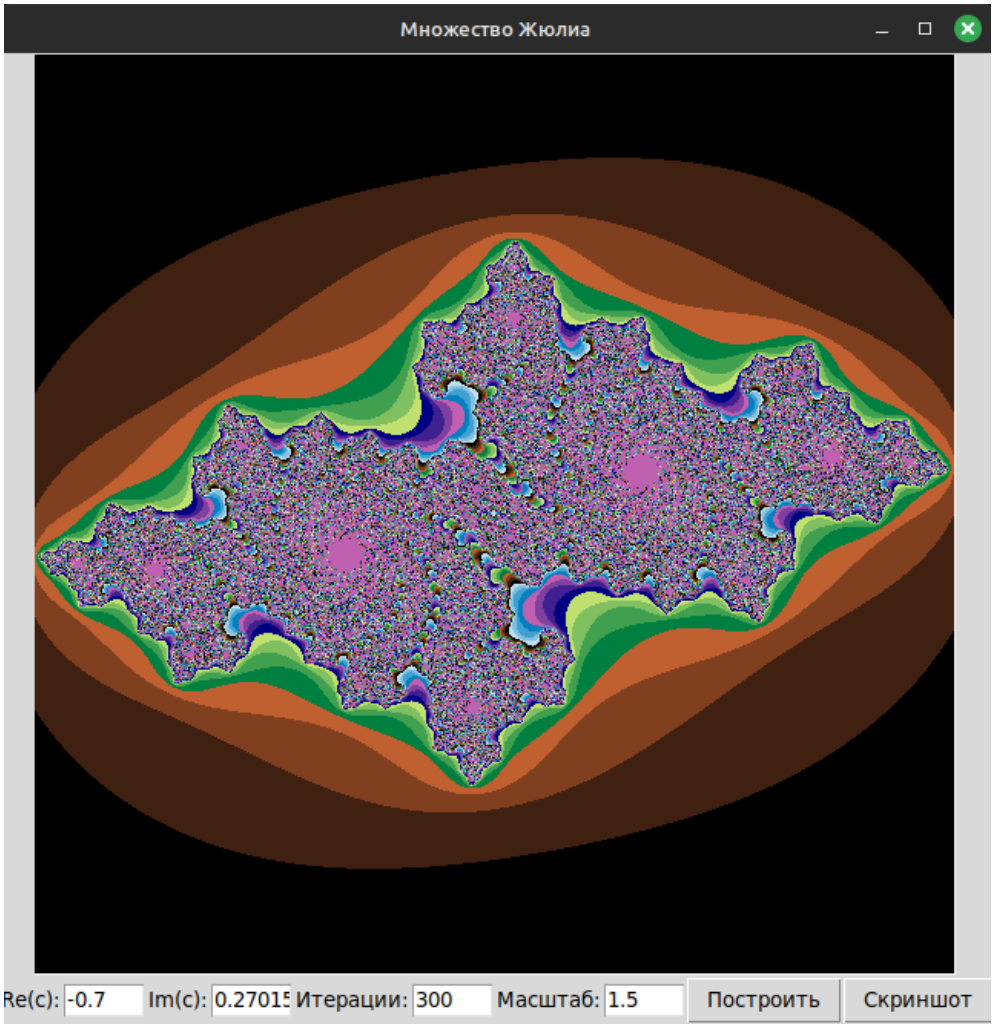
Код программы:

```
    import tkinter as tk
from PIL import Image, ImageTk, ImageGrab
import numpy as np
```

```python
from datetime import datetime


class JuliaFractalApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Множество Жюлиа")

        self.canvas = tk.Canvas(root, width=600, height=600, bg="white")
        self.canvas.pack()

        # Панель управления
        control_frame = tk.Frame(root)
        control_frame.pack()

        self.entries = {}
        for label in ["Re(c)", "Im(c)", "Итерации", "Масштаб"]:
            tk.Label(control_frame, text=label + ":").pack(side=tk.LEFT)
            entry = tk.Entry(control_frame, width=6)
            entry.pack(side=tk.LEFT)
            self.entries[label] = entry

        # Значения по умолчанию
        self.entries["Re(c)"].insert(0, "-0.7")
        self.entries["Im(c)"].insert(0, "0.27015")
        self.entries["Итерации"].insert(0, "300")
        self.entries["Масштаб"].insert(0, "1.5")

        tk.Button(control_frame, text="Построить", command=self.generate_fractal).pack(side=tk.LEFT)
        tk.Button(control_frame, text="Скриншот", command=self.take_screenshot).pack(side=tk.LEFT)

        self.image = None
        self.generate_fractal()

    def generate_fractal(self):
        try:
            creal = float(self.entries["Re(c)"].get())
            cimag = float(self.entries["Im(c)"].get())
            iterations = int(self.entries["Итерации"].get())
            zoom = float(self.entries["Масштаб"].get())
        except ValueError:
            print("Некорректный ввод параметров.")
            return

        width, height = 600, 600
        x = np.linspace(-zoom, zoom, width)
        y = np.linspace(-zoom, zoom, height)
        X, Y = np.meshgrid(x, y)
        Z = X + 1j * Y
        C = complex(creal, cimag)

        img = np.zeros((height, width, 3), dtype=np.uint8)

        for i in range(iterations):
            mask = np.abs(Z) <= 2
            Z[mask] = Z[mask] ** 2 + C
            img[mask] = (i % 4 * 64, i % 8 * 32, i % 16 * 16)

        pil_image = Image.fromarray(img)
        self.image = ImageTk.PhotoImage(pil_image)
        self.canvas.create_image(0, 0, anchor=tk.NW, image=self.image)
```

```
    def take_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()

        screenshot = ImageGrab.grab().crop((x, y, x1, y1))
        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        filename = f"julia_screenshot_{timestamp}.png"
        screenshot.save(filename)
        print(f"Скриншот сохранён как {filename}")


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

Рисунки с результатами работы программы:



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