МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Лабораторная работа №7
По дисциплине “**Современные платформы программирования**”

Выполнил:
студент группы ПО-11
Турабов А. В.
Проверил:
Козик И. Д.

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!

Изобразить в окне приложения отрезок, вращающийся в плоскости формы вокруг точки, движущейся по отрезку.

Код программы

```
import sys
import math
from PyQt5.QtWidgets import (QApplication, QMainWindow, QWidget, QVBoxLayout, QHBoxLayout,
                             QPushButton, QLabel, QSlider, QLineEdit, QDoubleSpinBox, QFileDialog)
from PyQt5.QtCore import Qt, QTimer
from PyQt5.QtGui import QPainter, QPen, QColor, QImage

class DrawingWidget(QWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.parent = parent  # ссылка на главное окно

    def paintEvent(self, event):
        painter = QPainter(self)
        self.parent.draw_line(painter)  # вызываем отрисовку из главного окна

class RotatingLineApp(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Вращающийся отрезок")
        self.setGeometry(100, 100, 800, 600)

        # Основные параметры
        self.line_length = 200
        self.rotation_speed = 1
        self.movement_speed = 0.5
        self.rotation_angle = 0
        self.movement_position = 0
```

```
        self.is_rotating = False
        self.timer = QTimer(self)
        self.timer.timeout.connect(self.update_animation)

        # Цвета
        self.line_color = QColor(0, 0, 255)
        self.pivot_color = QColor(255, 0, 0)

        # Создаем центральный виджет и layout
        central_widget = QWidget()
        self.setCentralWidget(central_widget)
        main_layout = QHBoxLayout(central_widget)

        # Панель управления
        control_panel = QWidget()
        control_layout = QVBoxLayout(control_panel)
        control_layout.setAlignment(Qt.AlignTop)

        # Элементы управления
        self.length_spin = QDoubleSpinBox()
        self.length_spin.setRange(50, 400)
        self.length_spin.setValue(self.line_length)
        self.length_spin.valueChanged.connect(self.set_line_length)

        self.rotation_slider = QSlider(Qt.Horizontal)
        self.rotation_slider.setRange(1, 20)
        self.rotation_slider.setValue(self.rotation_speed)
        self.rotation_slider.valueChanged.connect(self.set_rotation_speed)

        self.movement_slider = QSlider(Qt.Horizontal)
        self.movement_slider.setRange(1, 20)
        self.movement_slider.setValue(int(self.movement_speed * 10))
        self.movement_slider.valueChanged.connect(self.set_movement_speed)

        self.start_button = QPushButton("Старт/Стоп")
        self.start_button.clicked.connect(self.toggle_rotation)

        self.screenshot_button = QPushButton("Сделать скриншот")
        self.screenshot_button.clicked.connect(self.take_screenshot)

        # Добавляем элементы на панель управления
        control_layout.addWidget(QLabel("Длина отрезка:"))
        control_layout.addWidget(self.length_spin)
        control_layout.addWidget(QLabel("Скорость вращения:"))
        control_layout.addWidget(self.rotation_slider)
        control_layout.addWidget(QLabel("Скорость движения точки:"))
        control_layout.addWidget(self.movement_slider)
        control_layout.addWidget(self.start_button)
        control_layout.addWidget(self.screenshot_button)

        # Добавляем панель управления и область рисования в главный layout
        main_layout.addWidget(control_panel)

        # Область рисования
        self.drawing_area = DrawingWidget(self)
        self.drawing_area.setStyleSheet("background-color: white;")
        main_layout.addWidget(self.drawing_area, 1)

    def set_line_length(self, value):
        self.line_length = value
        self.drawing_area.update()

    def set_rotation_speed(self, value):
        self.rotation_speed = value

    def set_movement_speed(self, value):
        self.movement_speed = value / 10

    def toggle_rotation(self):
        self.is_rotating = not self.is_rotating
        if self.is_rotating:
```

```
            self.timer.start(16)  # ~60 FPS
        else:
            self.timer.stop()

    def update_animation(self):
        self.rotation_angle += self.rotation_speed * 0.01
        self.movement_position += self.movement_speed * 0.01

        # Сброс позиции движения, если она выходит за пределы [0, 1]
        if self.movement_position > 1:
            self.movement_position = 0

        self.drawing_area.update()

    def take_screenshot(self):
        # Создаем изображение области рисования
        image = QImage(self.drawing_area.size(), QImage.Format_ARGB32)
        painter = QPainter(image)
        self.drawing_area.render(painter)
        painter.end()

        # Сохраняем изображение
        file_name, _ = QFileDialog.getSaveFileName(self, "Сохранить скриншот", "", "PNG Images (*.png)")
        if file_name:
            if not file_name.endswith('.png'):
                file_name += '.png'
            image.save(file_name)

    def paintEvent(self, event):
        # Основная отрисовка происходит в drawing_area
        pass

    def resizeEvent(self, event):
        self.drawing_area.update()

    def draw_line(self, painter):
        # Получаем размеры области рисования
        width = self.drawing_area.width()
        height = self.drawing_area.height()

        # Вычисляем центр области рисования
        center_x = width / 2
        center_y = height / 2

        # Вычисляем текущую позицию точки вращения на отрезке
        pivot_x = center_x - self.line_length / 2 + self.movement_position * self.line_length
        pivot_y = center_y

        # Рисуем базовый отрезок (серый)
        painter.setPen(QPen(QColor(200, 200, 200), 1))
        painter.drawLine(int(center_x - self.line_length / 2), int(center_y),
                         int(center_x + self.line_length / 2), int(center_y))

        # Рисуем точку вращения (красная)
        painter.setPen(QPen(self.pivot_color, 5))
        painter.drawPoint(int(pivot_x), int(pivot_y))

        # Вычисляем координаты вращающегося отрезка
        end_x = pivot_x + self.line_length * math.cos(self.rotation_angle)
        end_y = pivot_y + self.line_length * math.sin(self.rotation_angle)

        # Рисуем вращающийся отрезок (синий)
        painter.setPen(QPen(self.line_color, 2))
        painter.drawLine(int(pivot_x), int(pivot_y), int(end_x), int(end_y))


if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = RotatingLineApp()
    window.show()
    sys.exit(app.exec_())
```

Результат работы программы

Задание 2. Реализовать построение заданного типа фрактала по варианту

Везде, где это необходимо, предусмотреть ввод параметров, влияющих на

внешний вид фрактала - Снежинка Коха

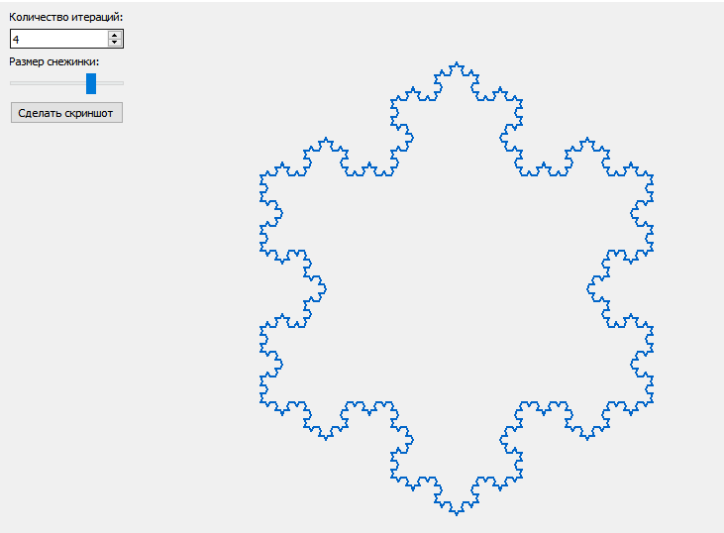
Код программы

```
import sys
import math
from PyQt5.QtWidgets import (QApplication, QMainWindow, QWidget, QVBoxLayout, QHBoxLayout,
                             QPushButton, QLabel, QSlider, QSpinBox, QFileDialog)
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QPainter, QPen, QColor, QImage

class DrawingWidget(QWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.parent = parent

    def paintEvent(self, event):
        painter = QPainter(self)
   self.parent.draw_koch_snowflake(painter) # отрисовка из главного окна


class KochSnowflakeApp(QMainWindow):
  def __init__(self):
    super().__init__()
    self.setWindowTitle("Снежинка Коха")
    self.setGeometry(100, 100, 800, 600)

        # Параметры снежинки
        self.iterations = 3
        self.size = 300
        self.line_color = QColor(0, 100, 200)

        # Создаем центральный виджет и layout
        central_widget = QWidget()
        self.setCentralWidget(central_widget)
        main_layout = QHBoxLayout(central_widget)

        # Панель управления
```

```
        control_panel = QWidget()
        control_layout = QVBoxLayout(control_panel)
        control_layout.setAlignment(Qt.AlignTop)

        # Элементы управления
        self.iterations_spin = QSpinBox()
        self.iterations_spin.setRange(0, 6)
        self.iterations_spin.setValue(self.iterations)
        self.iterations_spin.valueChanged.connect(self.set_iterations)

        self.size_slider = QSlider(Qt.Horizontal)
        self.size_slider.setRange(100, 500)
        self.size_slider.setValue(self.size)
        self.size_slider.valueChanged.connect(self.set_size)

        self.screenshot_button = QPushButton("Сделать скриншот")
        self.screenshot_button.clicked.connect(self.take_screenshot)

        # Добавляем элементы на панель управления
        control_layout.addWidget(QLabel("Количество итераций:"))
        control_layout.addWidget(self.iterations_spin)
        control_layout.addWidget(QLabel("Размер снежинки:"))
        control_layout.addWidget(self.size_slider)
        control_layout.addWidget(self.screenshot_button)

        # Добавляем панель управления и область рисования в главный layout
        main_layout.addWidget(control_panel)

        # Область рисования
        self.drawing_area = DrawingWidget(self)
        self.drawing_area.setStyleSheet("background-color: white;")
        main_layout.addWidget(self.drawing_area, 1)

    def set_iterations(self, value):
        self.iterations = value
        self.drawing_area.update()

    def set_size(self, value):
        self.size = value
        self.drawing_area.update()

    def take_screenshot(self):
        # Создаем изображение области рисования
        image = QImage(self.drawing_area.size(), QImage.Format_ARGB32)
        painter = QPainter(image)
        self.drawing_area.render(painter)
        painter.end()

        # Сохраняем изображение
        file_name, _ = QFileDialog.getSaveFileName(self, "Сохранить скриншот", "", "PNG Images
(*.png)")
        if file_name:
            if not file_name.endswith('.png'):
                file_name += '.png'
            image.save(file_name)

    def paintEvent(self, event):
        # Основная отрисовка происходит в drawing_area
        pass

    def resizeEvent(self, event):
        self.drawing_area.update()

    def draw_koch_snowflake(self, painter):
        width = self.drawing_area.width()
        height = self.drawing_area.height()

        # Центрируем снежинку
        center_x = width / 2
        center_y = height / 2

        # Вычисляем координаты начального треугольника
        size = self.size
        h = size * math.sqrt(3) / 2

        # Три вершины равностороннего треугольника
        points = [
```

```
            (center_x, center_y - h * 2/3),
            (center_x - size/2, center_y + h/3),
            (center_x + size/2, center_y + h/3)
        ]

        # Рисуем три кривых Коха (по одной на каждую сторону треугольника)
        for i in range(3):
            start = points[i]
            end = points[(i + 1) % 3]
            self.draw_koch_curve(painter, start, end, self.iterations)

    def draw_koch_curve(self, painter, p1, p2, iterations):
        if iterations == 0:
            # Базовый случай - просто рисуем линию
            painter.setPen(QPen(self.line_color, 2))
            painter.drawLine(int(p1[0]), int(p1[1]), int(p2[0]), int(p2[1]))
        else:
            # Разделяем отрезок на 3 части
            dx = p2[0] - p1[0]
            dy = p2[1] - p1[1]

            # Точки, разделяющие отрезок на 3 равные части
            a = (p1[0] + dx / 3, p1[1] + dy / 3)
            b = (p1[0] + 2 * dx / 3, p1[1] + 2 * dy / 3)

            # Вычисляем вершину равностороннего треугольника
            angle = math.atan2(dy, dx)
            length = math.sqrt((dx / 3)**2 + (dy / 3)**2)

            # Поворачиваем на 60 градусов
            c_angle = angle + math.pi / 3
            c = (
                a[0] + length * math.cos(c_angle),
                a[1] + length * math.sin(c_angle)
            )

            # Рекурсивно рисуем 4 отрезка
            self.draw_koch_curve(painter, p1, a, iterations - 1)
            self.draw_koch_curve(painter, a, c, iterations - 1)
            self.draw_koch_curve(painter, c, b, iterations - 1)
            self.draw_koch_curve(painter, b, p2, iterations - 1)


if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = KochSnowflakeApp()
    window.show()
    sys.exit(app.exec_())
```

Результат работы



Вывод: освоил возможности языка программирования Python в разработке оконных приложений