Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 3

з дисципліни «Технології та засоби розробки комп’ютерної графіки та мультимедіа»

Тема: «Візуалізація фракталів»

Виконала:
студентка групи ІС-34
Ященко Олександра.
Дата здачі 08.10.25
Захищено з балом _____

Перевірила:
ст. вик. кафедри ІСТ
Хмелюк Марина Сергіївна

Київ 2025

Завдання

Побудувати сніжинку Коха з можливістю регулювання глибини рекурсії.

Опис реалізації

Структура програми

Програма складається з двох основних класів:

1. GraphicsForm (QMainWindow) — головне вікно програми

- Розмір: 750×850 пікселів
- Чорний фон
- Містить заголовок, інформацію про студента, область малювання, слайдер для регулювання глибини та кнопку закриття
- Кастомна іконка вікна (синє коло з білою рамкою)

2. DrawingWidget (QWidget) — віджет для малювання

- Розмір: 700×600 пікселів
- Білий фон з рамкою кольору #2c3e50
- Реалізує метод paintEvent() для відтворення сніжинки Коха
- Підтримує глибину рекурсії від 0 до 6 рівнів

Технічні рішення

Математична основа

Сніжинка Коха будується на основі рівностороннього трикутника. Початкові координати вершин обчислюються за формулами:

- Центр: (cx, cy) = (350, 350)
- Розмір сторони: size = 400
- Висота рівностороннього трикутника: $h = size \times \sqrt{3} / 2$

Координати трьох вершин:

- $x_1 = cx$, $y_1 = cy - 2h/3$ (верхня вершина)
- $x_2 = cx - size/2$, $y_2 = cy + h/3$ (ліва нижня)
- $x_3 = cx + size/2$, $y_3 = cy + h/3$ (права нижня)

Рекурсивний алгоритм побудови кривої Коха

Базовий випадок (depth = 0):

Малюється пряма лінія від точки (x_1, y_1) до точки (x_2, y_2).

Рекурсивний крок:

1. Кожна лінія ділиться на три рівні частини з кроком $dx = (x_2 - x_1) / 3$, $dy = (y_2 - y_1) / 3$
2. Обчислюються координати ключових точок:
    - Точка A: $(xa, ya) = (x_1 + dx, y_1 + dy)$ — кінець першої третини
    - Точка B: $(xb, yb) = (x_1 + 2dx, y_1 + 2dy)$ — початок останньої третини
    - Точка C: (xc, yc) — вершина рівностороннього трикутника, побудованого на середній третині
3. Координати точки C обчислюються через поворот вектора на 60° ($\pi/3$):
    - $xc = xa + dx \times \cos(\pi/3) - dy \times \sin(\pi/3)$
    - $yc = ya + dx \times \sin(\pi/3) + dy \times \cos(\pi/3)$
4. Рекурсивно будуються чотири нові лінії:
    - від (x_1, y_1) до (xa, ya)
    - від (xa, ya) до (xc, yc)
    - від (xc, yc) до (xb, yb)
    - від (xb, yb) до (x_2, y_2)

Побудова сніжинки

Сніжинка створюється шляхом застосування алгоритму кривої Коха до трьох сторін рівностороннього трикутника:

- Сторона 1: від вершини 1 до вершини 2
- Сторона 2: від вершини 2 до вершини 3
- Сторона 3: від вершини 3 до вершини 1

Елементи інтерфейсу

Слайдер глибини:

- Діапазон значень: 0-6
- Початкове значення: 3
- При зміні значення оновлюється відображення сніжинки та текстова мітка
- Стилізований з використанням синіх відтінків

Кнопка Close:

- Червоний колір (#ff0000)
- Rounded corners (border-radius: 20px)
- При наведенні курсора змінює колір на білий з чорним текстом

Графічні параметри:

- Колір ліній: #3498db (синій)
- Товщина ліній: 2 пікселі
- Застосовано антиаліасинг для згладжування країв

Код програми

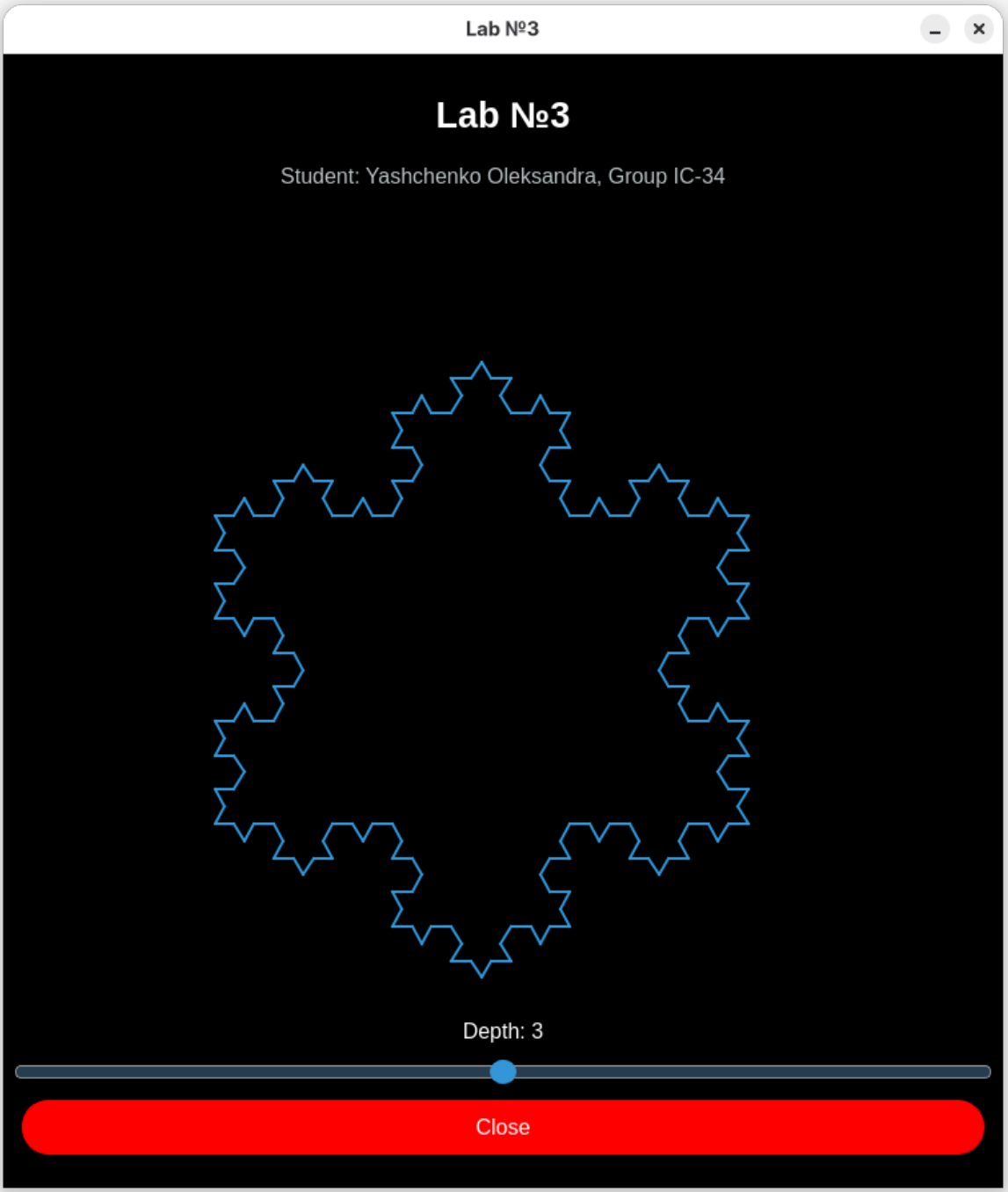
```
import sys
import math
from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget,
QVBoxLayout, QPushButton, QLabel, QSlider
from PyQt5.QtGui import QPainter, QPen, QColor, QFont, QPixmap, QIcon
from PyQt5.QtCore import Qt


class DrawingWidget(QWidget):
    def __init__(self):
        super().__init__()
        self.depth = 3
        self.setFixedSize(700, 600)
        self.setStyleSheet("background-color: white; border: 2px solid
#2c3e50;")

    def paintEvent(self, event):
        painter = QPainter(self)
        painter.setRenderHint(QPainter.Antialiasing)
        painter.setPen(QPen(QColor(52, 152, 219), 2))

        size = 400
        height = size * math.sqrt(3) / 2
        cx = 350
        cy = 350

        x1 = cx
        y1 = cy - 2 * height / 3
        x2 = cx - size / 2
        y2 = cy + height / 3
        x3 = cx + size / 2
        y3 = cy + height / 3

        self.koch_line(painter, x1, y1, x2, y2, self.depth)
        self.koch_line(painter, x2, y2, x3, y3, self.depth)
        self.koch_line(painter, x3, y3, x1, y1, self.depth)

    def koch_line(self, painter, x1, y1, x2, y2, depth):
        if depth == 0:
            painter.drawLine(int(x1), int(y1), int(x2), int(y2))
        else:
            dx = (x2 - x1) / 3
            dy = (y2 - y1) / 3

            xa = x1 + dx
            ya = y1 + dy

            xb = x1 + 2 * dx
            yb = y1 + 2 * dy

            angle = math.pi / 3
```

```
            xc = xa + dx * math.cos(angle) - dy * math.sin(angle)
            yc = ya + dx * math.sin(angle) + dy * math.cos(angle)

            self.koch_line(painter, x1, y1, xa, ya, depth - 1)
            self.koch_line(painter, xa, ya, xc, yc, depth - 1)
            self.koch_line(painter, xc, yc, xb, yb, depth - 1)
            self.koch_line(painter, xb, yb, x2, y2, depth - 1)


class GraphicsForm(QMainWindow):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        self.setWindowTitle("Lab №3")
        self.setFixedSize(750, 850)

        self.setWindowIcon(self.createIcon())

        self.setStyleSheet("background-color: black")

        central_widget = QWidget()
        self.setCentralWidget(central_widget)

        layout = QVBoxLayout()
        layout.setAlignment(Qt.AlignCenter)

        title_label = QLabel("Lab №3")
        title_label.setFont(QFont("Arial", 20, QFont.Bold))
        title_label.setStyleSheet("color: white; margin: 10px;")
        title_label.setAlignment(Qt.AlignCenter)

        student_label = QLabel("Student: Yashchenko Oleksandra, Group
IC-34")
        student_label.setFont(QFont("Arial", 12, QFont.StyleItalic))
        student_label.setStyleSheet("color: #bdc3c7; margin: 5px;")
        student_label.setAlignment(Qt.AlignCenter)

        self.drawing_widget = DrawingWidget()

        depth_label = QLabel("Depth: 3")
        depth_label.setFont(QFont("Arial", 12))
        depth_label.setStyleSheet("color: white; margin: 5px;")
        depth_label.setAlignment(Qt.AlignCenter)
        self.depth_label = depth_label

        slider = QSlider(Qt.Horizontal)
        slider.setMinimum(0)
        slider.setMaximum(6)
        slider.setValue(3)
        slider.setTickPosition(QSlider.TicksBelow)
        slider.setTickInterval(1)
        slider.setStyleSheet("""
            QSlider::groove:horizontal {
                border: 1px solid #999999;
                height: 8px;
                background: #2c3e50;
                margin: 2px 0;
                border-radius: 4px;
```

```
            }
            QSlider::handle:horizontal {
                background: #3498db;
                border: 1px solid #3498db;
                width: 18px;
                margin: -5px 0;
                border-radius: 9px;
            }
            QSlider::handle:horizontal:hover {
                background: #2980b9;
            }
        """)
        slider.valueChanged.connect(self.change_depth)

        close_button = QPushButton("Close")
        close_button.setFont(QFont("Arial", 12))
        close_button.setStyleSheet("""
            QPushButton {
                background-color: red;
                color: white;
                margin: 5px;
                border: 1px solid red;
                border-radius: 20px;
                padding: 10px 30px;
            }
            QPushButton:hover {
                background-color: white;
                color: black;
                border: none;
            }
        """)
        close_button.clicked.connect(self.close)

        layout.addWidget(title_label)
        layout.addWidget(student_label)
        layout.addWidget(self.drawing_widget)
        layout.addWidget(depth_label)
        layout.addWidget(slider)
        layout.addWidget(close_button)

        central_widget.setLayout(layout)

    def createIcon(self):
        pixmap = QPixmap(32, 32)
        pixmap.fill(QColor("#3498db"))

        painter = QPainter(pixmap)
        painter.setPen(QPen(QColor("white"), 2))
        painter.drawEllipse(8, 8, 16, 16)
        painter.end()

        return QIcon(pixmap)

    def change_depth(self, value):
        self.drawing_widget.depth = value
        self.depth_label.setText(f"Depth: {value}")
        self.drawing_widget.update()


def main():
```

```
    app = QApplication(sys.argv)
    form = GraphicsForm()
    form.show()
    sys.exit(app.exec_())


if __name__ == "__main__":
    main()
```

Результати



Програма успішно візуалізує сніжинку Коха з різними рівнями деталізації:

Depth = 0: Базовий рівносторонній трикутник

Depth = 1: Трикутник з трьома виступами (зірка Давида)

Depth = 2: Формування характерної форми сніжинки з 12 великими виступами

Depth = 3: Детальна сніжинка з 48 виступами середнього розміру

Depth = 4-6: Надзвичайно детальна структура з сотнями дрібних виступів, що демонструє самоподібність фракталу

При кожній ітерації кількість сегментів збільшується в 4 рази: 3, 12, 48, 192, 768, 3072, 12288 сегментів відповідно для глибини 0-6.

Висновки

В ході виконання лабораторної роботи було успішно освоєно:

- Побудову фрактальних структур за допомогою рекурсивних алгоритмів
- Реалізацію класичного фракталу — сніжинки Коха
- Використання тригонометричних функцій для обчислення координат при повороті векторів
- Створення інтерактивного інтерфейсу з елементами управління (слайдер)
- Оптимізацію рекурсивних викликів для побудови складних геометричних структур

Програма повністю відповідає вимогам завдання та демонструє основні властивості фракталів: самоподібність, нескінченну деталізацію та рекурсивну природу побудови. Інтерактивний слайдер дозволяє наочно спостерігати, як з простого трикутника поступово формується складна фрактальна структура.

Особливу увагу було приділено математичній точності обчислень координат точок та коректності реалізації алгоритму повороту для побудови рівносторонніх трикутників на кожній ітерації.