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

Лабораторна робота № 2

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

Тема: «Візуалізація лінійних зображень. Рекурсивні алгоритми при побудові лінійних зображень»

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

1. Накреслити візерунок, утворений 50 вкладеними квадратами. Сторони першого квадрата паралельні осям координат екрану. Вершини кожного наступного квадрата - це точки на сторонах попереднього квадрата, що ділять ці сторони у відношенні до P = 0,08.
2. Побудувати трикутник Серпінського за допомогою рекурсивного алгоритму з глибиною вкладеності 5 рівнів.

   Опис реалізації

Структура програми

Програма складається з трьох основних класів:

1. GraphicsForm (QMainWindow) — головне вікно програми

- Розмір: 750×750 пікселів
- Чорний фон
- Містить заголовок, інформацію про студента, область малювання та три кнопки управління
- Кастомна іконка вікна (синє коло з білою рамкою)

2. DrawingWidget (QWidget) — віджет для малювання

- Розмір: 700×500 пікселів
- Чорний фон з рамкою кольору #2c3e50
- Підтримує два режими відображення (50 вкладених квадратів та трикутник Серпінського)
- Реалізує метод paintEvent() для відтворення графіки

3. Режими роботи

- Режим 1: Візуалізація 50 вкладених квадратів
- Режим 2: Побудова трикутника Серпінського

Технічні рішення

Реалізація 50 вкладених квадратів

Алгоритм:

1. Задаються початкові координати першого квадрата розміром 400×400 пікселів
2. У циклі 50 разів виконуються наступні дії:

    - Малюються чотири сторони поточного квадрата за допомогою drawLine()
    - За параметричними рівняннями обчислюються координати наступного квадрата
3. Параметр P = 0,08 визначає положення вершин нового квадрата на сторонах попереднього

Математична основа:

Використовуються параметричні рівняння прямої:

- $x = x1 + (x2 - x1) \times P$
- $y = y1 + (y2 - y1) \times P$

При P = 0,08 кожна вершина нового квадрата ділить сторону попереднього у відношенні 0,08 / 0,92.

Реалізація трикутника Серпінського

Рекурсивний алгоритм:

1. Базовий випадок: якщо глибина рекурсії дорівнює 0, малюється трикутник
2. Рекурсивний крок:
    - Знаходяться середини кожної зі сторін трикутника за формулою P = 0,5
    - Викликається метод для трьох кутових трикутників (без центрального)
    - Глибина рекурсії зменшується на 1
3. Додана оптимізація: якщо відстань між серединами сторін менша за 1 піксель, рекурсія зупиняється

Особливості реалізації:

- Початковий трикутник має вершини: (350, 30), (50, 450), (650, 450)
- Максимальна глибина рекурсії: 5 рівнів
- Використовується метод sierpinski_recursive() з параметрами координат вершин та поточної глибини

Графічні елементи інтерфейсу

Кнопки управління:

- "50 Nested Squares" (синя, #3498db) — перемикає на режим вкладених квадратів
- "Sierpinski Triangle" (зелена, #2ecc71) — перемикає на режим трикутника Серпінського
- "Close" (червона) — закриває програму

Стилізація:

- Усі кнопки мають rounded corners (border-radius: 20px)
- При наведенні курсора колір кнопок змінюється на білий з чорним текстом
- Застосовано антиаліасинг для згладжування країв графічних елементів

Код програми

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

class DrawingWidget(QWidget):
    def __init__(self):
        super().__init__()
        self.mode = 1
        self.setFixedSize(700, 500)
        self.setStyleSheet("background-color: black; border: 2px solid
#2c3e50;")

    def paintEvent(self, event):
        painter = QPainter(self)
        painter.setRenderHint(QPainter.Antialiasing)
        painter.setPen(QPen(QColor(255, 255, 255), 2))

        if self.mode == 1:
            self.draw_squares(painter)
        elif self.mode == 2:
            self.draw_sierpinski(painter)

    def draw_squares(self, painter):
        size = 400
        x1, y1 = 150, 50
        x2, y2 = x1 + size, y1
        x3, y3 = x2, y1 + size
        x4, y4 = x1, y3

        P = 0.08

        for _ in range(50):
            painter.drawLine(int(x1), int(y1), int(x2), int(y2))
            painter.drawLine(int(x2), int(y2), int(x3), int(y3))
            painter.drawLine(int(x3), int(y3), int(x4), int(y4))
            painter.drawLine(int(x4), int(y4), int(x1), int(y1))

            nx1 = x1 + (x2 - x1) * P
```

```
            ny1 = y1 + (y2 - y1) * P

            nx2 = x2 + (x3 - x2) * P
            ny2 = y2 + (y3 - y2) * P

            nx3 = x3 + (x4 - x3) * P
            ny3 = y3 + (y4 - y3) * P

            nx4 = x4 + (x1 - x4) * P
            ny4 = y4 + (y1 - y4) * P

            x1, y1 = nx1, ny1
            x2, y2 = nx2, ny2
            x3, y3 = nx3, ny3
            x4, y4 = nx4, ny4

    def draw_sierpinski(self, painter):
        x1, y1 = 350, 30
        x2, y2 = 50, 450
        x3, y3 = 650, 450
        self.sierpinski_recursive(painter, x1, y1, x2, y2, x3, y3, 5)

    def sierpinski_recursive(self, painter, x1, y1, x2, y2, x3, y3,
depth):
        if depth == 0:
            painter.drawLine(int(x1), int(y1), int(x2), int(y2))
            painter.drawLine(int(x2), int(y2), int(x3), int(y3))
            painter.drawLine(int(x3), int(y3), int(x1), int(y1))
        else:
            mx1 = (x1 + x2) / 2
            my1 = (y1 + y2) / 2

            mx2 = (x2 + x3) / 2
            my2 = (y2 + y3) / 2
            if math.sqrt((mx2-mx1)**2+(my2-my1)**2)<=1:
                print(depth)
                painter.drawLine(int(x1), int(y1), int(x2), int(y2))
                painter.drawLine(int(x2), int(y2), int(x3), int(y3))
                painter.drawLine(int(x3), int(y3), int(x1), int(y1))
                return

            mx3 = (x3 + x1) / 2
            my3 = (y3 + y1) / 2

            self.sierpinski_recursive(painter, x1, y1, mx1, my1, mx3, my3,
depth - 1)
            self.sierpinski_recursive(painter, mx1, my1, x2, y2, mx2, my2,
depth - 1)
            self.sierpinski_recursive(painter, mx3, my3, mx2, my2, x3, y3,
depth - 1)

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

    def initUI(self):
        self.setWindowTitle("Lab №2")
        self.setFixedSize(750, 750)
```

```
        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 №2")
        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()

        btn1 = QPushButton("50 Nested Squares")
        btn1.setFont(QFont("Arial", 12))
        btn1.setStyleSheet("""
            QPushButton {
                background-color: #3498db;
                color: white;
                margin: 5px;
                border: 1px solid #3498db;
                border-radius: 20px;
                padding: 10px 30px;
            }
            QPushButton:hover {
                background-color: white;
                color: black;
                border: none;
            }
        """)
        btn1.clicked.connect(lambda: self.change_mode(1))

        btn2 = QPushButton("Sierpinski Triangle")
        btn2.setFont(QFont("Arial", 12))
        btn2.setStyleSheet("""
            QPushButton {
                background-color: #2ecc71;
                color: white;
                margin: 5px;
                border: 1px solid #2ecc71;
                border-radius: 20px;
                padding: 10px 30px;
            }
            QPushButton:hover {
                background-color: white;
                color: black;
                border: none;
            }
        """)
        btn2.clicked.connect(lambda: self.change_mode(2))
```

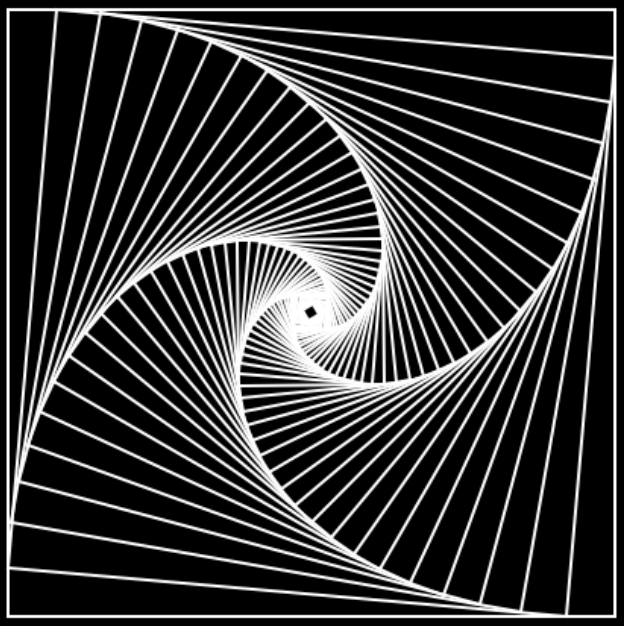
```
        close_button = QPushButton("Close")
        close_button.setFont(QFont("Arial", 12))
        close_button.setStyleSheet("""
            QPushButton {
                background-color: red;
                color: white;
                margin: 5px;
                border: 1px solid red;
                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(btn1)
        layout.addWidget(btn2)
        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_mode(self, mode):
        self.drawing_widget.mode = mode
        self.drawing_widget.update()

def main():
    app = QApplication(sys.argv)
    form = GraphicsForm()
    form.show()
    sys.exit(app.exec_())

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

Результати

Lab №2

Lab №2

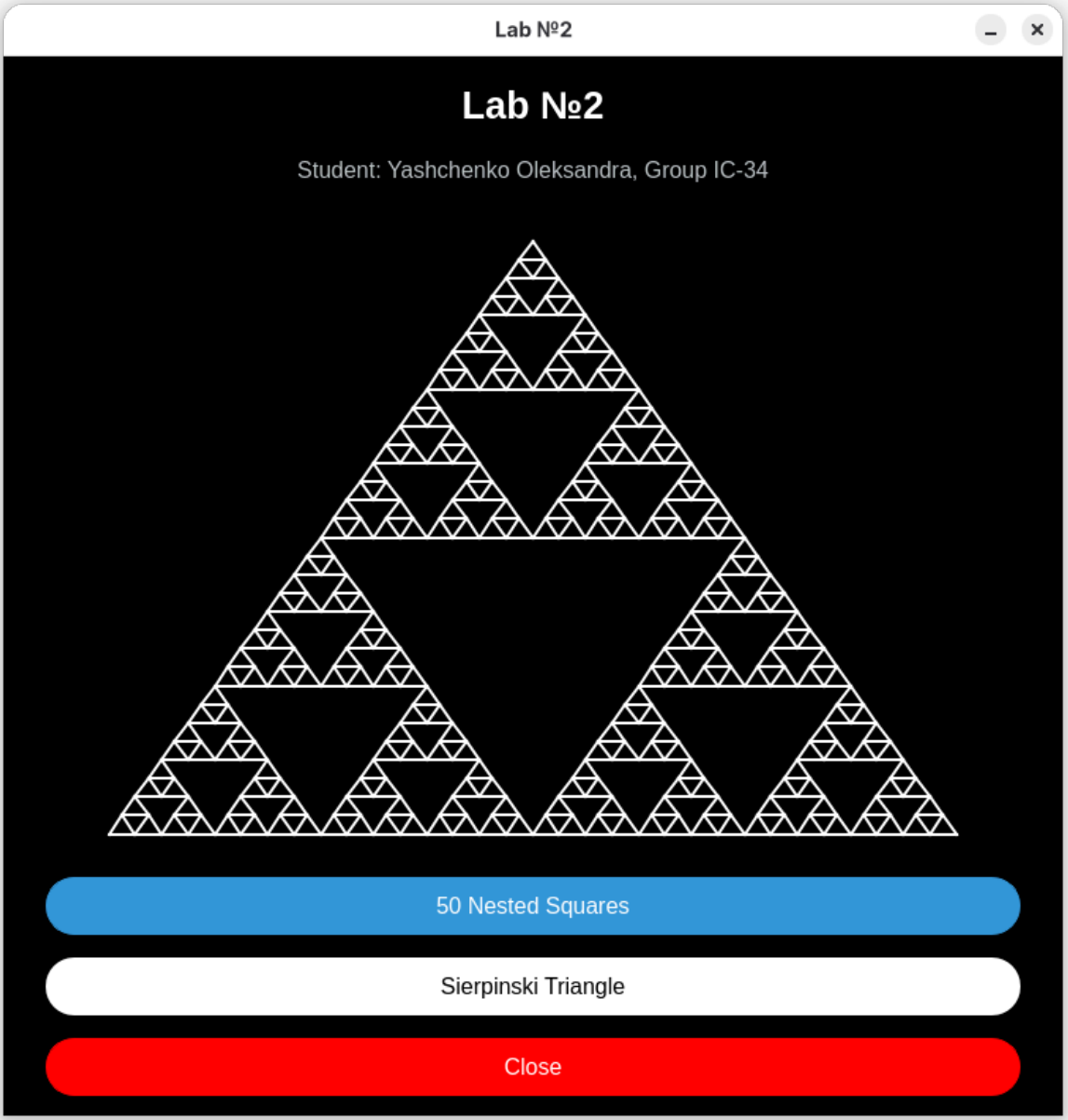
Student: Yashchenko Oleksandra, Group IC-34



50 Nested Squares

Sierpinski Triangle

Close



Програма успішно реалізує обидва завдання:

1. Візуалізація 50 вкладених квадратів

Створено плавний спіральний візерунок, утворений послідовно зменшуваними квадратами. Візерунок демонструє геометричну прогресію та створює ефект перспективи.

2. Трикутник Серпінського

Побудовано фрактальну структуру з 5 рівнями вкладеності. Кількість трикутників на кожному рівні збільшується в геометричній прогресії (3^n). Загальна кількість намальованих трикутників: 3^5 = 243.

Висновки

В ході виконання лабораторної роботи було успішно освоєно:

- Використання параметричних рівнянь прямої для побудови складних геометричних візерунків
- Реалізацію рекурсивних алгоритмів для побудови фрактальних структур
- Роботу з графічними примітивами PyQt5 (drawLine, QPainter, QPen)
- Організацію багаторежимного графічного інтерфейсу з перемиканням між різними типами візуалізацій
- Оптимізацію рекурсивних алгоритмів через додавання умов припинення

Програма повністю відповідає вимогам завдання: коректно візуалізує 50 вкладених квадратів з заданим параметром P = 0,08 та будує трикутник Серпінського рекурсивним методом. Інтерфейс є інтуїтивно зрозумілим та зручним для користувача.

Особливу увагу було приділено математичній точності обчислень координат та ефективності рекурсивного алгоритму. Додана перевірка на мінімальну відстань між точками запобігає надмірному заглибленню рекурсії та покращує продуктивність програми.