

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

Технології та засоби розробки комп'ютерної графіки та мультимедіа

Лабораторна робота №2

Виконав:  
студент групи  
ІО-25  
Льоскін І. В.  
Перевірила:  
Хмелюк М. С.

Київ – 2024

**Тема:** Візуалізація лінійних зображень. Рекурсивні алгоритми при побудові лінійних зображень

**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

**Хід роботи:**

Аналогічно Лабораторній роботі №1 створюємо новий проект. Також створюємо нову іконку (або використовуємо з попередньої лабораторної роботи), встановлюємо її, додаємо задній фон, змінюємо ім'я форми та її текст. Після чого приступаємо до написання коду. Для цього перейдемо в файл SquaresForm.cs:

```
public partial class SquaresForm : Form
{
    public SquaresForm()
    {
        InitializeComponent();
        this.ClientSize = new Size(400, 400);
        this.Paint += new PaintEventHandler(this.OnPaint);
    }

    private const int SquareCount = 50; // number of
    nested squares
    private const float P = 0.08f; // k for the
    next square
    private float size = 200; // size of the
    first square
    private PointF topLeftCorner = new PointF(100, 100); // start
    coordinales of the top left point

    private void OnPaint(object sender, PaintEventArgs e)
    {
        Graphics g = e.Graphics;
        Pen pen = new Pen(Color.Black);

        // Start coordinates or the square's points
        PointF topLeft = topLeftCorner;
        PointF topRight = new PointF(topLeft.X + size, topLeft.Y);
        PointF bottomLeft = new PointF(topLeft.X, topLeft.Y + size);
```

```

        PointF bottomRight = new PointF(topLeft.X + size, topLeft.Y + size);

        for (int i = 0; i < SquareCount; i++)
        {
            // Draw the square with current values
            g.DrawPolygon(pen, new PointF[] { topLeft, topRight,
bottomRight, bottomLeft });

            // Calculate new coordinates for nested square
            PointF newTopLeft = new PointF(
                topLeft.X + (topRight.X - topLeft.X) * P,
                topLeft.Y + (bottomLeft.Y - topLeft.Y) * P
            );
            PointF newTopRight = new PointF(
                topRight.X - (topRight.X - topLeft.X) * P,
                topRight.Y + (bottomRight.Y - topRight.Y) * P
            );
            PointF newBottomLeft = new PointF(
                bottomLeft.X + (bottomRight.X - bottomLeft.X) * P,
                bottomLeft.Y - (bottomLeft.Y - topLeft.Y) * P
            );
            PointF newBottomRight = new PointF(
                bottomRight.X - (bottomRight.X - bottomLeft.X) * P,
                bottomRight.Y - (bottomRight.Y - topRight.Y) * P
            );

            // update points for new square
            topLeft = newTopLeft;
            topRight = newTopRight;
            bottomLeft = newBottomLeft;
            bottomRight = newBottomRight;
        }

        pen.Dispose();
    }

    private void SquaresForm_Load(object sender, EventArgs e)
    {

    }
}

```

Насправді коментарі Пояснюють основні моменти коду, алк додам невеликі пояснення:

*Ініціалізуємо форму, додаємо розмір та підписуємось методом Paint на подію OnPaint*

```
public SquaresForm()
{
    InitializeComponent();
    this.ClientSize = new Size(400, 400);
    this.Paint += new PaintEventHandler(this.OnPaint);
}
```

*Ініціалізуємо об'єкти Graphics та Pen. Задаємо початкові координати кожної вершини квадрату наприклад координати верхньої лівої і правої точок будуть (100; 100) та (300; 100) відповідно*

```
Graphics g = e.Graphics;
Pen pen = new Pen(Color.Black);

// Start coordinates or the square's points
PointF topLeft = topLeftCorner;
PointF topRight = new PointF(topLeft.X + size, topLeft.Y);
PointF bottomLeft = new PointF(topLeft.X, topLeft.Y + size);
PointF bottomRight = new PointF(topLeft.X + size, topLeft.Y + size);
```

*Далі запускаємо цикл та рисуємо квадрат із поточними значеннями (поточні значення змінюватимуться в кінці кожної ітерації)*

```
for (int i = 0; i < SquareCount; i++)
{
    // Draw the square with current values
    g.DrawPolygon(pen, new PointF[] { topLeft, topRight, bottomRight, bottomLeft });
}
```

*Також додаємо код для відкриття другої форми по натисканню на кнопку*

```
private void button1_Click(object sender, EventArgs e)
{
    TrianglesForm secondForm = new TrianglesForm();
    secondForm.Show();
}
```

*Тепер обчислюємо нові значення та присвоюємо їх як поточні*

```

// Calculate new coordinates for nested square
PointF newTopLeft = new PointF(
    topLeft.X + (topRight.X - topLeft.X) * P,
    topLeft.Y + (bottomLeft.Y - topLeft.Y) * P
);
PointF newTopRight = new PointF(
    topRight.X - (topRight.X - topLeft.X) * P,
    topRight.Y + (bottomRight.Y - topRight.Y) * P
);
PointF newBottomLeft = new PointF(
    bottomLeft.X + (bottomRight.X - bottomLeft.X) * P,
    bottomLeft.Y - (bottomLeft.Y - topLeft.Y) * P
);
PointF newBottomRight = new PointF(
    bottomRight.X - (bottomRight.X - bottomLeft.X) * P,
    bottomRight.Y - (bottomRight.Y - topRight.Y) * P
);

// update points for new square
topLeft = newTopLeft;
topRight = newTopRight;
bottomLeft = newBottomLeft;
bottomRight = newBottomRight;

```

*На кінець звільняємо зайняті ресурси*

```
pen.Dispose();
```

Тепер створюємо нову форму та пишемо код:

```

public partial class TrianglesForm : Form
{
    public TrianglesForm()
    {
        InitializeComponent();
        this.ClientSize = new Size(600, 600);
        this.Paint += new PaintEventHandler(this.OnPaint);
    }

    private const int Depth = 5; // depth of recursion
    private const float P = 0.5f; // k of shift (1/2 for Sierpinski triangle)

```

```

private void OnPaint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;

    // original triangle with three points
    PointF p1 = new PointF(300, 50); // top point
    PointF p2 = new PointF(50, 500); // bottom left point
    PointF p3 = new PointF(550, 500); // bottom right point

    // draw the Sierpinski triangle
    DrawSierpinski(g, p1, p2, p3, Depth);
}

// recursive function for drawing the Sierpinski triangle
private void DrawSierpinski(Graphics g, PointF p1, PointF p2, PointF p3, int
depth)
{
    // if we have reached the recursion depth, we draw a triangle
    if (depth == 0)
    {
        DrawTriangle(g, p1, p2, p3);
    }
    else
    {
        // calculate the midpoints of each side of the triangle
        PointF mid1 = new PointF((p1.X + p2.X) * P, (p1.Y + p2.Y) * P);
        PointF mid2 = new PointF((p2.X + p3.X) * P, (p2.Y + p3.Y) * P);
        PointF mid3 = new PointF((p3.X + p1.X) * P, (p3.Y + p1.Y) * P);

        // recursively draw three smaller triangles
        DrawSierpinski(g, p1, mid1, mid3, depth - 1);
        DrawSierpinski(g, mid1, p2, mid2, depth - 1);
        DrawSierpinski(g, mid3, mid2, p3, depth - 1);
    }
}
}

```

```
// function to draw a triangle by three vertices
```

```
private void DrawTriangle(Graphics g, PointF p1, PointF p2, PointF p3)
{
    g.DrawPolygon(Pens.Black, new PointF[] { p1, p2, p3 });
}
```

```
private void TrianglesForm_Load(object sender, EventArgs e)
{

}
}
```

Пояснення коду:

*Опис класу форми в якому вказуємо її розмір та викаємо метод OnPaint коли треба відрендерити форму, та щось намалювати за допомогою модуля Paint:*

```
public TrianglesForm()
{
    InitializeComponent();
    this.ClientSize = new Size(600, 600);
    this.Paint += new PaintEventHandler(this.OnPaint);
}
```

*Константа глибини рекурсії та коефіцієнт зсуву для трикутників*

```
private const int Depth = 5;    // depth of recursion
private const float P = 0.5f;  // k of shift (1/2 for Sierpinski triangle)
```

*Метод OnPaint в якому створюються три три верхівки трикутника та викликається метод DrawSierpinski*

```
private void OnPaint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;

    // original triangle with three points
    PointF p1 = new PointF(300, 50); // top point
    PointF p2 = new PointF(50, 500); // bottom left point
    PointF p3 = new PointF(550, 500); // bottom right point

    // draw the Sierpinski triangle
    DrawSierpinski(g, p1, p2, p3, Depth);
}
```

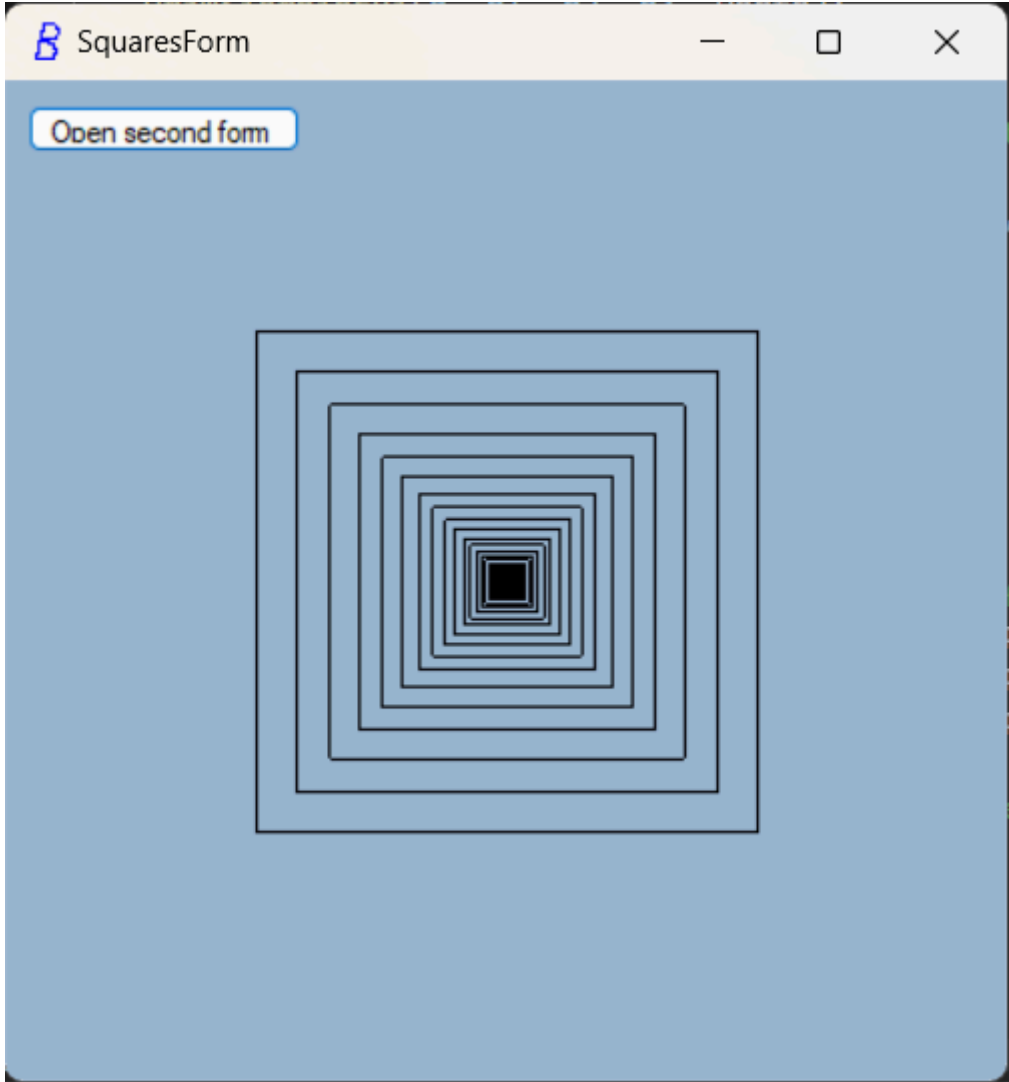
*Якщо глибина рекурсії рівна 0, то ми просто рисуємо трикутник з поточними зтаченнями, інакше - ми обчислюємо середини сторін поточного трикутника та викликаємо метод DrawSierpinski рекурсивно для трьох менших трикутників, кожен з яких визначається трьома новими точками*

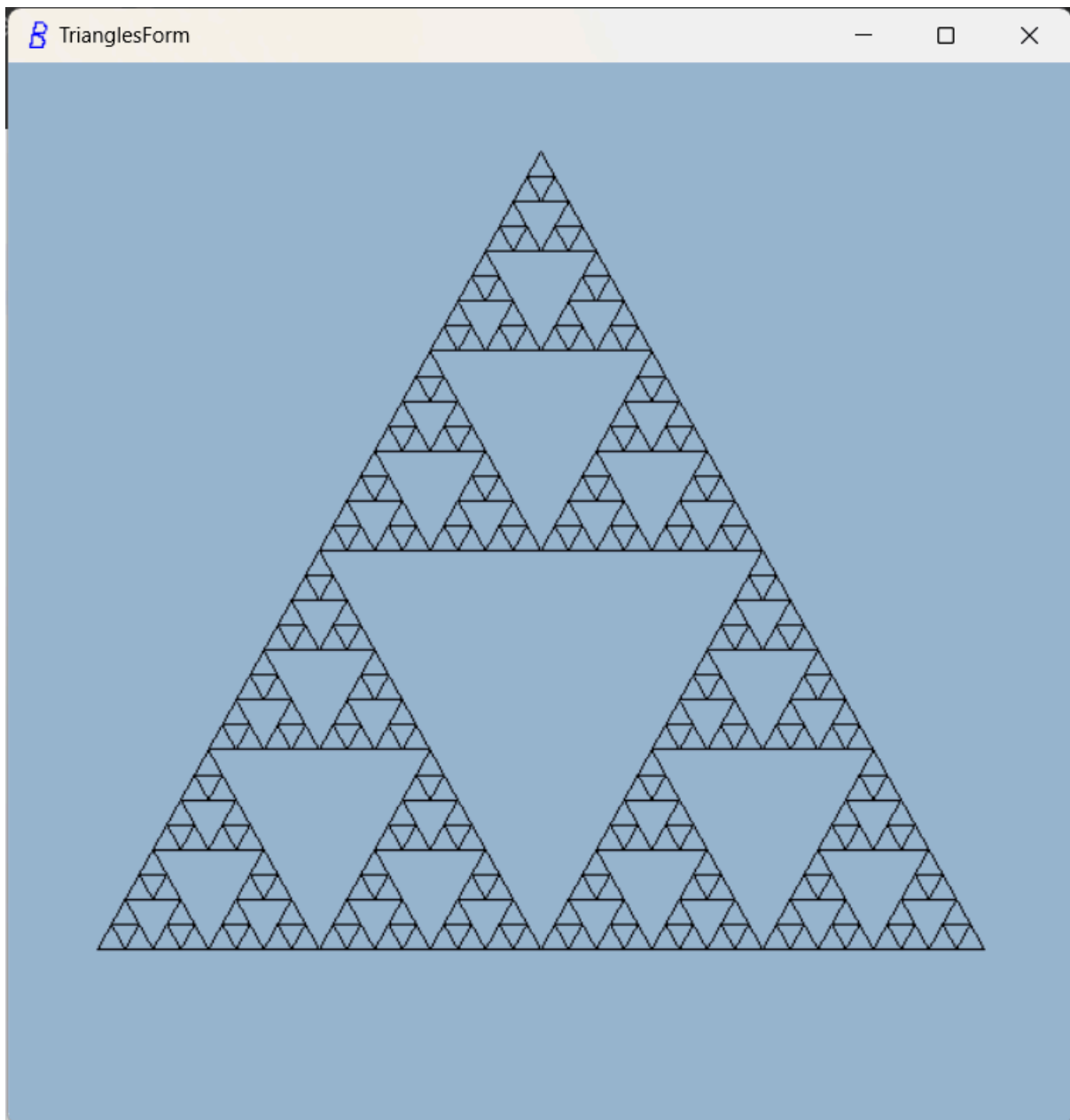
```
private void DrawSierpinski(Graphics g, PointF p1, PointF p2, PointF p3, int depth)
{
    // if we have reached the recursion depth, we draw a triangle
    if (depth == 0)
    {
        DrawTriangle(g, p1, p2, p3);
    }
    else
    {
        // calculate the midpoints of each side of the triangle
        PointF mid1 = new PointF((p1.X + p2.X) * P, (p1.Y + p2.Y) * P);
        PointF mid2 = new PointF((p2.X + p3.X) * P, (p2.Y + p3.Y) * P);
        PointF mid3 = new PointF((p3.X + p1.X) * P, (p3.Y + p1.Y) * P);

        // recursively draw three smaller triangles
        DrawSierpinski(g, p1, mid1, mid3, depth - 1);
        DrawSierpinski(g, mid1, p2, mid2, depth - 1);
        DrawSierpinski(g, mid3, mid2, p3, depth - 1);
    }
}
```

**Результат:**







**Висновок:** Виконуючи цю лабораторну роботу, я попрактикував свої навички у програмуванні мовою C#, зіткнувся з рекурсією - це рідко зустрічається на практиці, але знання є необхідними, тому це обов'язково допоможе мені у майбутньому. Також я попрактикувався у використанні модулів для рисування зображень.