МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут **ІКНІ**

Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи №3
На тему: «Побудова фрактальних зображень»
З дисципліни: «Ком’ютерна графіка»

Лектор:
доц. каф. ПЗ
Левус Є. В.

Виконав:
ст. гр. ПЗ-24
Войтинський Д. О.

Прийняв:
доц. каф. ПЗ
Горечко О. М.

«_____» _____ 2025р.

Σ = ____ _____

Львів – 2025

Тема роботи: Побудова фрактальних зображень

Мета роботи: Вивчити алгоритми побудови фракталів та навчитися їх програмно реалізовувати

Теоретичні відомості

Теоретичні відомості

Фрактал Мандельброта

Фрактал Мандельброта — це математична множина точок на комплексній площині, яка визначається рекурсивною формулою:

$$z_{n+1} = z_n^2 + c$$

де z і c— комплексні числа, $z_0 = 0$, і $n \geq 0$. Точка C належить множині Мандельброта, якщо послідовність z_n залишається обмеженою (не прямує до нескінченності) при нескінченній кількості ітерацій.

Комплексні числа

Комплексне число z записується у вигляді:

$z = x + yi$

де:

- x — дійсна частина
- y — уявна частина
- i — уявна одиниця

Операції з комплексними числами:

- Додавання: $(a + bi) + (c + di) = (a + c) + (b + d)i$
- Множення: $(a + bi) \cdot (c + di) = (ac - bd) + (ad + bc)i$

Для фракталу Мандельброта необхідно обчислити квадрат комплексного числа:

$$z^2 = (x + yi)^2 = (x^2 - y^2) + (2xy)i$$

Модуль (величина) комплексного числа:

$$|z| = |x + yi| = \sqrt{x^2 + y^2}$$

У програмі ітерації обчислюються доти, доки модуль $|z|$ не перевищить певне значення (зазвичай 2) або не досягнуто максимальної кількості ітерацій.

Броунівський рух

Броунівський рух — це випадковий рух частинок, спричинений їх зіткненням з молекулами середовища.

Математична модель

Броунівський рух можна описати за допомогою стохастичного диференціального рівняння:

$$dX_t = \mu dt + \sigma dW_t$$

де:

- X_t — положення частинки в момент часу t
- μ — параметр дрейфу
- σ — параметр дифузії
- W_t — вінерівський процес (математична модель броунівського руху)

У дискретному випадку, який використовується в симуляціях, зміщення частинки на кожному кроці можна обчислити як:

$$\Delta x = \text{rand}(-\alpha, \alpha) \Delta y = \text{rand}(-\alpha, \alpha)$$

де $rand(-a, a)$— випадкове число в діапазоні $[-a, a]$, a — параметр, що визначає розмір кроку.

При моделюванні броунівського руху частинки зіштовхуються зі стінками (межами середовища) і відбиваються від них, зберігаючи енергію руху.

Завдання

Написати програму для візуалізації фракталів та створити з її допомогою галерею фракталів. Програма повинна відповідати таким вимогам:

1) Універсальність алгоритмів побудови фракталів.

2) Масштабування зображень алгебраїчних фракталів.

3) Збереження фрактальних зображень у файлах.

4) Зручний інтерфейс користувача

Варіант 2:

Фрактал Мандельброта $f(z) = z^2 + c$, Броунівський рух

Код програми

scripts.js

```
// Отримання елементів з DOM
const canvas = document.getElementById('canvas');
const ctx = canvas.getContext('2d');
const width = canvas.width;
const height = canvas.height;

// Перемикання між режимами
const tabMandelbrot = document.getElementById('tab-mandelbrot');
const tabBrownian = document.getElementById('tab-brownian');
const mandelbrotSettingsElem = document.getElementById('mandelbrot-settings');
const brownianSettingsElem = document.getElementById('brownian-settings');

let currentMode = 'mandelbrot';
```

```
let animationId = null;

// Галерея фракталів - цікаві області фракталу Мандельброта
const fractalGallery = [
    {
        name: "Повний вигляд",
        xMin: -2.5,
        xMax: 1,
        yMin: -1.25,
        yMax: 1.25,
        maxIterations: 100,
        colorScheme: "rainbow"
    },
    {
        name: "Долина сіри",
        xMin: -0.7436438883807,
        xMax: -0.7436438883774,
        yMin: 0.1318259042924,
        yMax: 0.1318259042957,
        maxIterations: 500,
        colorScheme: "psychedelic"
    },
    {
        name: "Спіраль",
        xMin: -0.761574,
        xMax: -0.761572,
        yMin: 0.084223,
        yMax: 0.084225,
        maxIterations: 300,
        colorScheme: "blueToRed"
    },
    {
```

```
        name: "Міні-Мандельброт",
        xMin: -1.77,
        xMax: -1.74,
        yMin: -0.015,
        yMax: 0.015,
        maxIterations: 200,
        colorScheme: "rainbow"
    },
    {
        name: "Завитки",
        xMin: -0.16,
        xMax: -0.13,
        yMin: 1.035,
        yMax: 1.065,
        maxIterations: 200,
        colorScheme: "grayscale"
    }
];

// Функція для збереження зображення
function saveFractal() {
    saveCanvasAsImage('fractal');
}

// Функція для збереження броунівського руху як зображення
function saveBrownianMotion() {
    saveCanvasAsImage('brownian-motion');
}

// Загальна функція для збереження канвасу як зображення
function saveCanvasAsImage(prefix) {
    // Створюємо тимчасове посилання для завантаження
```

```
    const link = document.createElement('a');
    link.download = `${prefix}-${Date.now()}.png`;

    // Конвертуємо canvas у URL зображення
    link.href = canvas.toDataURL('image/png');

    // Симулюємо клік для завантаження
    document.body.appendChild(link);
    link.click();
    document.body.removeChild(link);

       document.getElementById('status').textContent = 'Зображення збережено!';
}

// Функція для завантаження фракталу з галереї
function loadFractalFromGallery(index) {
    const preset = fractalGallery[index];

    // Встановлюємо налаштування фракталу
    mandelbrotSettings.xMin = preset.xMin;
    mandelbrotSettings.xMax = preset.xMax;
    mandelbrotSettings.yMin = preset.yMin;
    mandelbrotSettings.yMax = preset.yMax;
    mandelbrotSettings.maxIterations = preset.maxIterations;
    mandelbrotSettings.colorScheme = preset.colorScheme;

    // Оновлюємо поля вводу та перемальовуємо фрактал
    updateInputsFromMandelbrotSettings();
    drawMandelbrot();

     // Перемикаємося на вкладку Мандельброта, якщо поточний режим не Мандельброт
```

```
    if (currentMode !== 'mandelbrot') {
        tabMandelbrot.click();
    }
}

// Ініціалізація галереї фракталів
function initFractalGallery() {
                                const       galleryContainer       =
document.getElementById('fractal-gallery');

    // Очищаємо контейнер галереї
    galleryContainer.innerHTML = '';

    // Додаємо елементи галереї
    fractalGallery.forEach((preset, index) => {
        const thumbnail = document.createElement('div');
        thumbnail.className = 'gallery-item';
        thumbnail.innerHTML = `
            <div class="thumbnail-placeholder">${index + 1}</div>
            <span>${preset.name}</span>
        `;

        // Додаємо обробник кліку
        thumbnail.addEventListener('click', () => {
            loadFractalFromGallery(index);
        });

        galleryContainer.appendChild(thumbnail);
    });
}

// Додаємо обробник для кнопки збереження фракталу
```

```
document.getElementById('save-fractal').addEventListener('click',
saveFractal);


// Додаємо обробник для кнопки збереження броунівського руху
document.getElementById('save-brownian').addEventListener('click',
saveBrownianMotion);


tabMandelbrot.addEventListener('click', () => {
    currentMode = 'mandelbrot';
    tabMandelbrot.classList.add('active');
    tabBrownian.classList.remove('active');
    mandelbrotSettingsElem.style.display = 'block';
    brownianSettingsElem.style.display = 'none';
      document.getElementById('gallery-container').style.display =
'block';
    stopBrownianMotion();
    drawMandelbrot();
});


tabBrownian.addEventListener('click', () => {
    currentMode = 'brownian';
    tabMandelbrot.classList.remove('active');
    tabBrownian.classList.add('active');
    mandelbrotSettingsElem.style.display = 'none';
    brownianSettingsElem.style.display = 'block';
      document.getElementById('gallery-container').style.display =
'none';
    resetBrownianMotion();
});


/**************************
 * Фрактал Мандельброта
 **************************/
```

```
// Налаштування відображення фрактала
let mandelbrotSettings = {
    xMin: -2.5,
    xMax: 1,
    yMin: -1.25,
    yMax: 1.25,
    maxIterations: 100,
    escapeRadius: 2,
    colorScheme: 'rainbow'
};

// Початкові налаштування
const initialMandelbrotSettings = {...mandelbrotSettings};

// Створення об'єкту для малювання
const imageData = ctx.createImageData(width, height);

// Функція для обчислення кольору точки в множині Мандельброта
function mandelbrot(cx, cy, maxIterations, escapeRadius) {
    // Початкове значення z = 0 + 0i
    let x = 0;  // Дійсна частина z
    let y = 0;  // Уявна частина z
    let iteration = 0;

    // Обчислюємо ітерації формули z = z² + c
     while (x*x + y*y <= escapeRadius*escapeRadius && iteration < maxIterations) {
        // Обчислення квадрату комплексного числа z = x + yi
        // z² = (x + yi)² = x² - y² + 2xyi
         const xTemp = x*x - y*y + cx;  // Дійсна частина: x² - y² + cx
        y = 2*x*y + cy;               // Уявна частина: 2xy + cy
```

```
        x = xTemp;
        iteration++;
    }

    // Повертаємо кількість ітерацій
    if (iteration === maxIterations) {
        return 0; // Точка належить множині
    }

    // Плавний колір з використанням логарифмічної інтерполяції
    return iteration + 1 - Math.log(Math.log(Math.sqrt(x*x + y*y))) / Math.log(2);
}

// Функція для перетворення значення в колір залежно від вибраної схеми
function getColor(value, maxIterations, scheme) {
    if (value === 0) {
        return [0, 0, 0, 255]; // Чорний для точок у множині
    }

    // Нормалізуємо значення для кольору (0-1)
    const normalized = value / maxIterations;

    switch(scheme) {
        case 'rainbow':
            // Кольорова схема веселки
            const hue = 360 * normalized;
            return hslToRgb(hue / 360, 1.0, 0.5);

        case 'blueToRed':
            // Градієнт від синього до червоного
            return [
```

```
                Math.round(255 * normalized), // R
                0, // G
                Math.round(255 * (1 - normalized)), // B
                255 // A
            ];

        case 'grayscale':
            // Градієнт відтінків сірого
            const intensity = Math.round(255 * normalized);
            return [intensity, intensity, intensity, 255];

        case 'psychedelic':
            // Психоделічна кольорова схема
            return [
                   Math.round(127.5 * (Math.sin(normalized * 15) +
1)),
                  Math.round(127.5 * (Math.sin(normalized * 25 + 2)
+ 1)),
                  Math.round(127.5 * (Math.sin(normalized * 10 - 1)
+ 1)),
                255
            ];

        default:
            return hslToRgb(normalized, 1.0, 0.5);
    }
}

// Функція для перетворення HSL в RGB
function hslToRgb(h, s, l) {
    let r, g, b;

    if (s === 0) {
```

```
        r = g = b = l;
    } else {
        const hue2rgb = (p, q, t) => {
            if (t < 0) t += 1;
            if (t > 1) t -= 1;
            if (t < 1/6) return p + (q - p) * 6 * t;
            if (t < 1/2) return q;
            if (t < 2/3) return p + (q - p) * (2/3 - t) * 6;
            return p;
        };

        const q = l < 0.5 ? l * (1 + s) : l + s - l * s;
        const p = 2 * l - q;
        r = hue2rgb(p, q, h + 1/3);
        g = hue2rgb(p, q, h);
        b = hue2rgb(p, q, h - 1/3);
    }

    return [Math.round(r * 255), Math.round(g * 255), Math.round(b * 255), 255];
}

// Функція для малювання фрактала
function drawMandelbrot() {
    // Оновлюємо статус
                document.getElementById('status').textContent = 'Обчислення...';

     // Використовуємо setTimeout для оновлення UI перед початком обчислень
    setTimeout(() => {
        const startTime = performance.now();
```

```
        for (let py = 0; py < height; py++) {
            for (let px = 0; px < width; px++) {
                    // Перетворення координат екрана у координати комплексної площини
                        const cx = mandelbrotSettings.xMin + (mandelbrotSettings.xMax - mandelbrotSettings.xMin) * px / width;
                        const cy = mandelbrotSettings.yMin + (mandelbrotSettings.yMax - mandelbrotSettings.yMin) * py / height;

                    // Обчислення значення для точки
                            const value = mandelbrot(cx, cy, mandelbrotSettings.maxIterations, mandelbrotSettings.escapeRadius);

                    // Отримання кольору
                        const [r, g, b, a] = getColor(value, mandelbrotSettings.maxIterations, mandelbrotSettings.colorScheme);

                    // Встановлення пікселя у imageData
                    const index = (py * width + px) * 4;
                    imageData.data[index] = r;
                    imageData.data[index + 1] = g;
                    imageData.data[index + 2] = b;
                    imageData.data[index + 3] = a;
            }
        }

        // Відображення на канвасі
        ctx.putImageData(imageData, 0, 0);

        const endTime = performance.now();
        const elapsedTime = Math.round(endTime - startTime);
          document.getElementById('status').textContent = `Готово. Час обчислення: ${elapsedTime} мс`;
    }, 10);
```

```
}

// Оновлення налаштувань з полів вводу
function updateMandelbrotSettingsFromInputs() {
                                    mandelbrotSettings.xMin         =
parseFloat(document.getElementById('xMin').value);
                                    mandelbrotSettings.xMax         =
parseFloat(document.getElementById('xMax').value);
                                    mandelbrotSettings.yMin         =
parseFloat(document.getElementById('yMin').value);
                                    mandelbrotSettings.yMax         =
parseFloat(document.getElementById('yMax').value);
                              mandelbrotSettings.maxIterations      =
parseInt(document.getElementById('maxIterations').value);
                               mandelbrotSettings.escapeRadius      =
parseFloat(document.getElementById('escapeRadius').value);
                                mandelbrotSettings.colorScheme      =
document.getElementById('colorScheme').value;
}

// Оновлення полів вводу з налаштувань
function updateInputsFromMandelbrotSettings() {
                        document.getElementById('xMin').value       =
mandelbrotSettings.xMin;
                        document.getElementById('xMax').value       =
mandelbrotSettings.xMax;
                        document.getElementById('yMin').value       =
mandelbrotSettings.yMin;
                        document.getElementById('yMax').value       =
mandelbrotSettings.yMax;
                  document.getElementById('maxIterations').value    =
mandelbrotSettings.maxIterations;
                   document.getElementById('escapeRadius').value    =
mandelbrotSettings.escapeRadius;
                    document.getElementById('colorScheme').value    =
mandelbrotSettings.colorScheme;
}
```

```
// Обробники подій для фракталу Мандельброта
document.getElementById('zoomIn').addEventListener('click', () => {
    const centerX = (mandelbrotSettings.xMin + mandelbrotSettings.xMax) / 2;
    const centerY = (mandelbrotSettings.yMin + mandelbrotSettings.yMax) / 2;
    const newWidth = (mandelbrotSettings.xMax - mandelbrotSettings.xMin) * 0.5;
    const newHeight = (mandelbrotSettings.yMax - mandelbrotSettings.yMin) * 0.5;

    mandelbrotSettings.xMin = centerX - newWidth / 2;
    mandelbrotSettings.xMax = centerX + newWidth / 2;
    mandelbrotSettings.yMin = centerY - newHeight / 2;
    mandelbrotSettings.yMax = centerY + newHeight / 2;

    updateInputsFromMandelbrotSettings();
    drawMandelbrot();
});

document.getElementById('zoomOut').addEventListener('click', () => {
    const centerX = (mandelbrotSettings.xMin + mandelbrotSettings.xMax) / 2;
    const centerY = (mandelbrotSettings.yMin + mandelbrotSettings.yMax) / 2;
    const newWidth = (mandelbrotSettings.xMax - mandelbrotSettings.xMin) * 2;
    const newHeight = (mandelbrotSettings.yMax - mandelbrotSettings.yMin) * 2;

    mandelbrotSettings.xMin = centerX - newWidth / 2;
    mandelbrotSettings.xMax = centerX + newWidth / 2;
    mandelbrotSettings.yMin = centerY - newHeight / 2;
```

```
    mandelbrotSettings.yMax = centerY + newHeight / 2;

    updateInputsFromMandelbrotSettings();
    drawMandelbrot();
});

document.getElementById('reset-mandelbrot').addEventListener('clic
k', () => {
    mandelbrotSettings = {...initialMandelbrotSettings};
    updateInputsFromMandelbrotSettings();
    drawMandelbrot();
});

document.getElementById('apply-mandelbrot').addEventListener('clic
k', () => {
    updateMandelbrotSettingsFromInputs();
    drawMandelbrot();
});

// Клік на канвас для збільшення конкретної області (для фрактала)
canvas.addEventListener('click', (event) => {
    if (currentMode !== 'mandelbrot') return;

    const rect = canvas.getBoundingClientRect();
    const x = event.clientX - rect.left;
    const y = event.clientY - rect.top;

      // Перетворення координат екрана у координати комплексної
площини
     const cx = mandelbrotSettings.xMin + (mandelbrotSettings.xMax
- mandelbrotSettings.xMin) * x / width;
     const cy = mandelbrotSettings.yMin + (mandelbrotSettings.yMax
- mandelbrotSettings.yMin) * y / height;
```

```
    // Зум навколо вибраної точки
                const    newWidth    =    (mandelbrotSettings.xMax    -
mandelbrotSettings.xMin) * 0.5;
                const    newHeight    =    (mandelbrotSettings.yMax    -
mandelbrotSettings.yMin) * 0.5;

    mandelbrotSettings.xMin = cx - newWidth / 2;
    mandelbrotSettings.xMax = cx + newWidth / 2;
    mandelbrotSettings.yMin = cy - newHeight / 2;
    mandelbrotSettings.yMax = cy + newHeight / 2;

    updateInputsFromMandelbrotSettings();
    drawMandelbrot();
});

/**************************
 * Броунівський рух
 *************************/

// Налаштування для броунівського руху
let brownianSettings = {
    particleCount: 500,
    particleSize: 2,
    stepSize: 1,
    trailLength: 50,
    colorMode: 'position',
    fadeTrail: true
};

// Початкові налаштування
const initialBrownianSettings = {...brownianSettings};

// Масив частинок
```

```
let particles = [];

// Клас для частинки броунівського руху
class Particle {
    constructor(x, y, size, colorMode) {
        this.x = x;
        this.y = y;
        this.size = size;
        this.colorMode = colorMode;
        this.trail = [];
        this.vx = 0;
        this.vy = 0;

        // Створюємо випадковий колір для частинки
        switch(colorMode) {
            case 'random':
                this.color = `hsl(${Math.random() * 360}, 100%,
50%)`;
                break;
            case 'position':
                // Колір залежить від позиції
                const hue = (x / width + y / height) * 180;
                this.color = `hsl(${hue}, 100%, 50%)`;
                break;
            case 'velocity':
                // Колір буде визначатися в методі update
                this.color = `hsl(0, 100%, 50%)`;
                break;
        }
    }

    update(stepSize, trailLength) {
```

```
// Зберігаємо поточну позицію в слід
this.trail.push({x: this.x, y: this.y});

// Обмеження довжини сліду
if (this.trail.length > trailLength) {
    this.trail.shift();
}

// Випадковий крок у Броунівському русі
this.vx = (Math.random() - 0.5) * stepSize * 2;
this.vy = (Math.random() - 0.5) * stepSize * 2;

this.x += this.vx;
this.y += this.vy;

// Обробка зіткнення зі стінками
if (this.x < 0) {
    this.x = 0;
    this.vx *= -1;
}
if (this.x > width) {
    this.x = width;
    this.vx *= -1;
}
if (this.y < 0) {
    this.y = 0;
    this.vy *= -1;
}
if (this.y > height) {
    this.y = height;
    this.vy *= -1;
}
```

```
        // Оновлення кольору залежно від режиму
        if (this.colorMode === 'velocity') {
             const speed = Math.sqrt(this.vx * this.vx + this.vy * this.vy);
             const normalizedSpeed = Math.min(1, speed / (stepSize * 2));
             const hue = normalizedSpeed * 240; // від 0 (червоний) до 240 (синій)
            this.color = `hsl(${hue}, 100%, 50%)`;
        }
    }

    draw(ctx, fadeTrail) {
        // Малюємо слід
        if (this.trail.length > 1) {
            ctx.beginPath();
            ctx.moveTo(this.trail[0].x, this.trail[0].y);

            for (let i = 1; i < this.trail.length; i++) {
                ctx.lineTo(this.trail[i].x, this.trail[i].y);

                // Якщо включено згасання сліду
                if (fadeTrail) {
                    const alpha = i / this.trail.length;
                        ctx.strokeStyle = this.color.replace('hsl', 'hsla').replace(')', `, ${alpha})`);
                } else {
                    ctx.strokeStyle = this.color;
                }

                ctx.lineWidth = this.size / 2;
                ctx.stroke();
```

```
                ctx.beginPath();
                ctx.moveTo(this.trail[i].x, this.trail[i].y);
            }
        }

        // Малюємо частинку
        ctx.beginPath();
        ctx.arc(this.x, this.y, this.size, 0, Math.PI * 2);
        ctx.fillStyle = this.color;
        ctx.fill();
    }
}

// Ініціалізація частинок
function initParticles() {
    particles = [];
    for (let i = 0; i < brownianSettings.particleCount; i++) {
        const x = Math.random() * width;
        const y = Math.random() * height;
                            particles.push(new Particle(x, y,
brownianSettings.particleSize, brownianSettings.colorMode));
    }
}

// Анімація броунівського руху
function animateBrownianMotion() {
    ctx.clearRect(0, 0, width, height);

    for (const particle of particles) {
                        particle.update(brownianSettings.stepSize,
brownianSettings.trailLength);
        particle.draw(ctx, brownianSettings.fadeTrail);
    }
```

```
    animationId = requestAnimationFrame(animateBrownianMotion);
}

// Зупинка анімації
function stopBrownianMotion() {
    if (animationId) {
        cancelAnimationFrame(animationId);
        animationId = null;
    }
}

// Скидання симуляції броунівського руху
function resetBrownianMotion() {
    stopBrownianMotion();
    updateBrownianSettingsFromInputs();
    initParticles();
    ctx.clearRect(0, 0, width, height);

    // Малюємо одиночний кадр
    for (const particle of particles) {
        particle.draw(ctx, brownianSettings.fadeTrail);
    }
}

// Оновлення налаштувань броунівського руху з полів вводу
function updateBrownianSettingsFromInputs() {
                            brownianSettings.particleCount       =
parseInt(document.getElementById('particleCount').value);
                             brownianSettings.particleSize       =
parseFloat(document.getElementById('particleSize').value);
                                brownianSettings.stepSize        =
parseFloat(document.getElementById('stepSize').value);
```

```
                                    brownianSettings.trailLength        =
parseInt(document.getElementById('trailLength').value);
                                     brownianSettings.colorMode         =
document.getElementById('colorMode').value;
                                    brownianSettings.fadeTrail          =
document.getElementById('fadeTrail').checked;
}


// Обробники подій для броунівського руху
document.getElementById('apply-brownian').addEventListener('click'
, resetBrownianMotion);
document.getElementById('reset-brownian').addEventListener('click'
, () => {
    brownianSettings = {...initialBrownianSettings};
                document.getElementById('particleCount').value    =
brownianSettings.particleCount;
                 document.getElementById('particleSize').value    =
brownianSettings.particleSize;
                    document.getElementById('stepSize').value     =
brownianSettings.stepSize;
                  document.getElementById('trailLength').value    =
brownianSettings.trailLength;
                   document.getElementById('colorMode').value     =
brownianSettings.colorMode;
                  document.getElementById('fadeTrail').checked    =
brownianSettings.fadeTrail;
    resetBrownianMotion();
});


document.getElementById('start-brownian').addEventListener('click'
, () => {
    if (!animationId) {
        updateBrownianSettingsFromInputs();
        animateBrownianMotion();
    }
});
```

```
document.getElementById('stop-brownian').addEventListener('click',
stopBrownianMotion);


// Початкова ініціалізація
initFractalGallery();
drawMandelbrot();
```

Результат виконання роботи

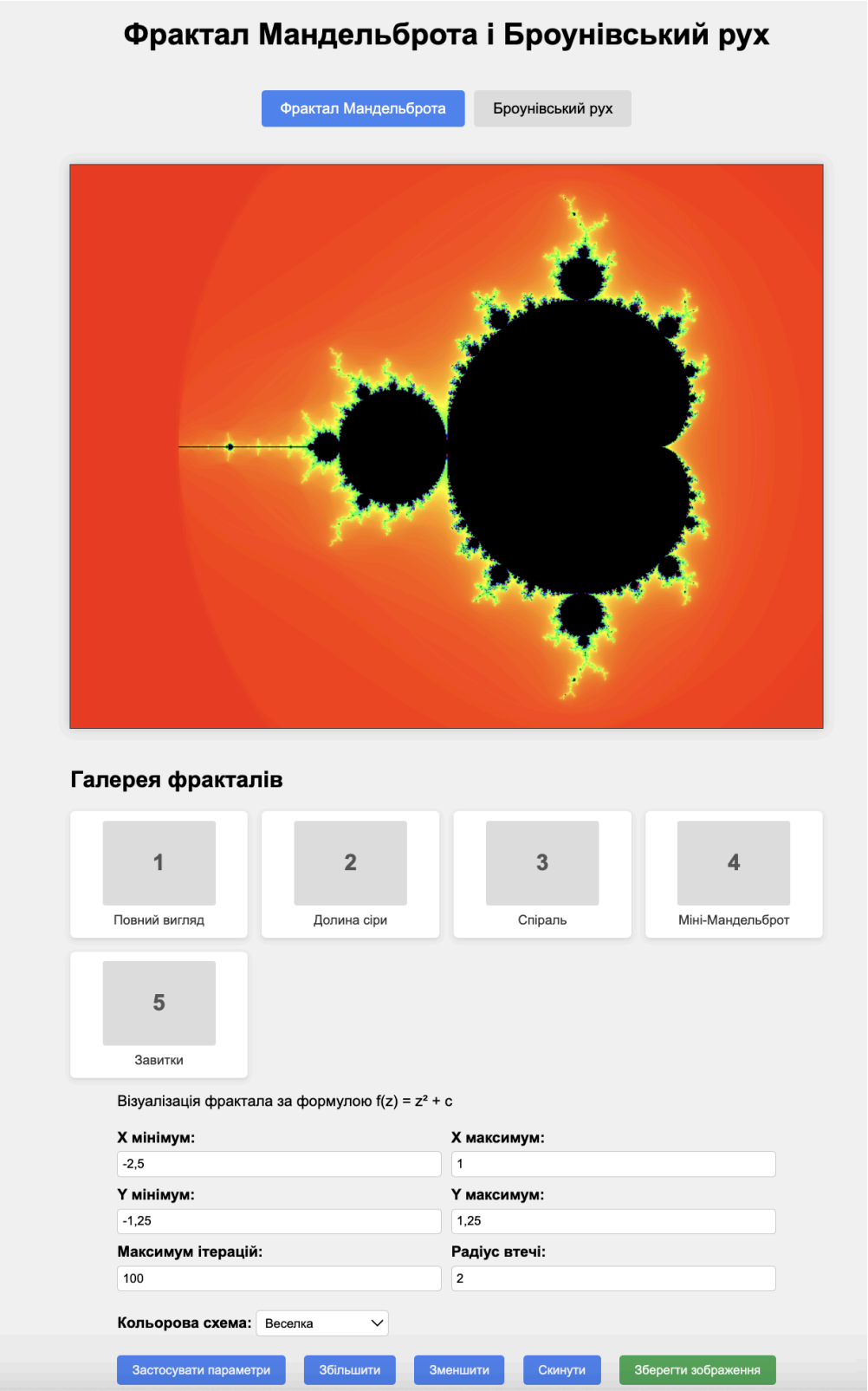

Рис.1. Вигляд фракталу Мандельброта

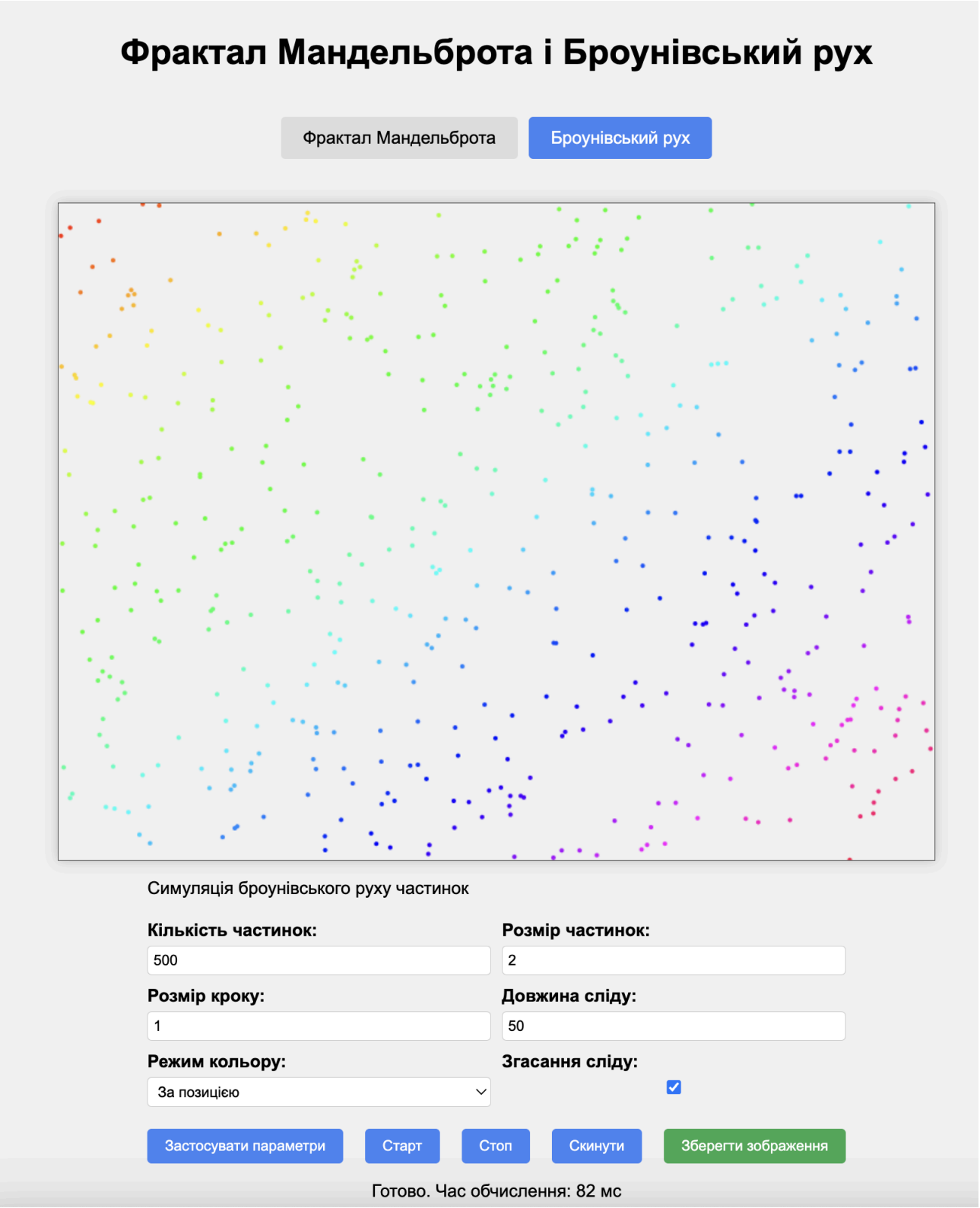

Рис.2. Загальний вигляд Броунівського руху

Рис.3. Фрактал мандельброта з зміненими параметрами

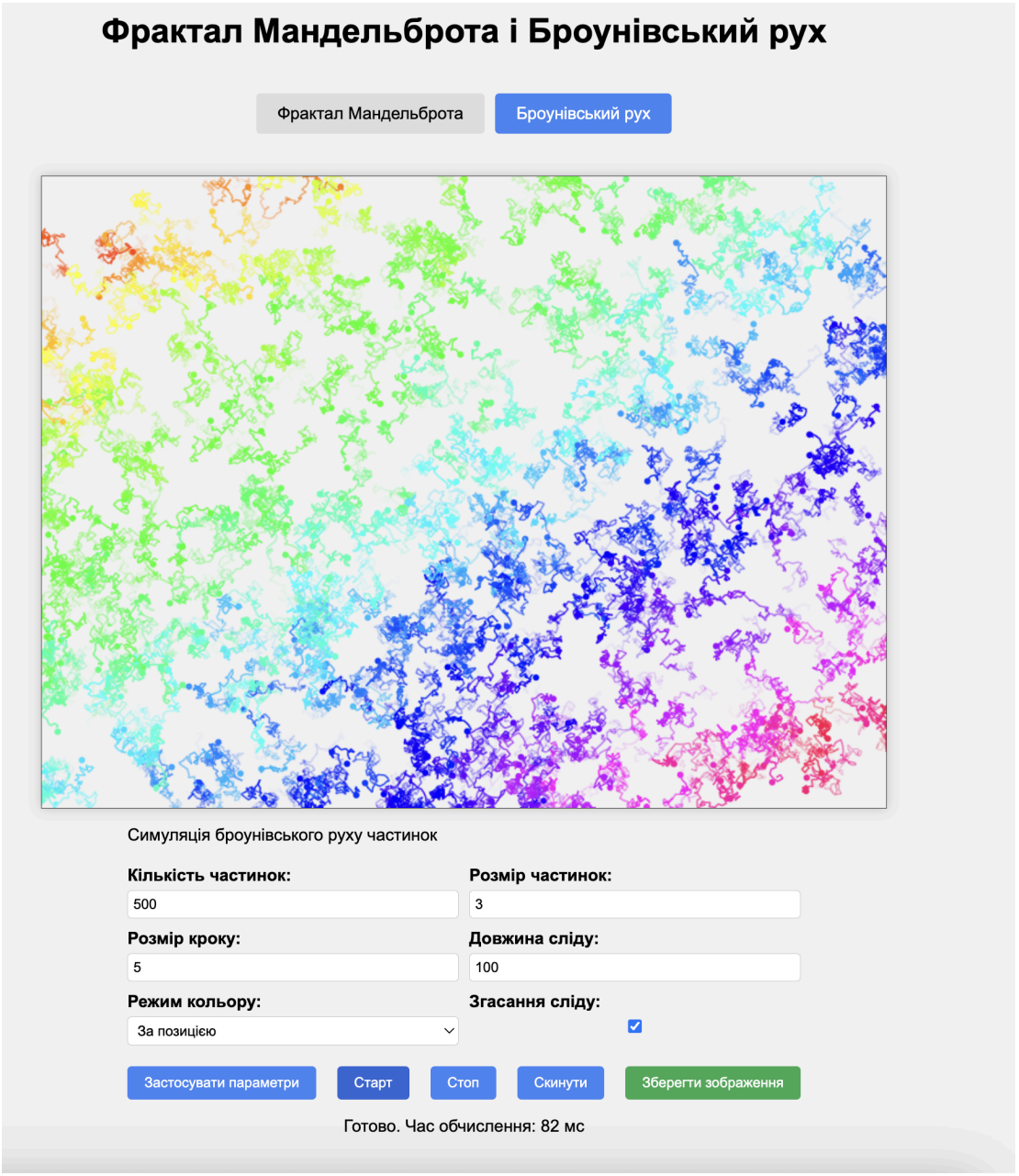

Рис.4. Броунівський рух

Висновок

На цій лабораторній роботі я вивчив алгоритми побудови фракталів та навчився їх програмно реалізовувати, мною було програмно реалізовано та візуалізовано фрактал Мандельброта та дискретно змоделював Броунівський рух.