**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **КНІТ**

Кафедра **ПЗ**



**Звіт про виконання лабораторних робіт**

**№1-4**

**з курсу «Комп’ютерна графіка»**

**Варіант** К7

**Лектор:**

доцент. каф. пз

Левус Є. В.

**Виконали студенти групи:**

ПЗ-33   
Лесневич Є. Є.   
та Курганевич А.-М.А.

**Перевірено:**

асист. каф. пз

Коцун В. І.

« » 2023 р.

∑= \_ .

Львів – 2023

Зміст

[Завдання 2](#_Toc153217540)

[Теоретичні відомості 2](#_Toc153217541)

[1. Опис функцій програми 2](#_Toc153217542)

[2. Алгоритми фракталів. 3](#_Toc153217543)

[3. Анотація кольорових моделей 5](#_Toc153217544)

[4. Оптимальний матричний вираз афінних перетворень 7](#_Toc153217545)

[5. Реалізація графічного режиму 8](#_Toc153217546)

[Результати 9](#_Toc153217547)

[1. Wireflow 9](#_Toc153217548)

[2. UI kit 9](#_Toc153217549)

[3. Отримані фрактали 9](#_Toc153217550)

[4. Вигляд програми для лабораторної роботи №3 9](#_Toc153217551)

[5. Вигляд програми для лабораторної роботи №4 9](#_Toc153217552)

[6. Текст програми з коментарями. 9](#_Toc153217553)

[Висновки 9](#_Toc153217554)

[Список використаних інформаційних джерел 9](#_Toc153217555)

# Завдання

Варіант К7.

1) Побудувати фрактальні зображення:

1.1) Різновиди фрактала Коха: «рандимізована» сніжинка, сніжинка «навпаки», острів.

1.2) Множини Мандельброта для 4 та 5 степеня z

2) Колірні моделі: RGB і XYZ. Змінити яскравість зображення. Робота з фрагментом зображення щодо перетворення моделі та зміни атрибуту кольору.

3) Реалізувати поворот трапеції відносно однієї з вершин проти та за годинниковою стрілки та одночасне пропорційне зменшення в N раз. Трапеція вводиться через вершини.

# Теоретичні відомості

## Опис функцій програми

1. Робота з фракталами
   1. Побува фракталу Коха трьох видів: острів, рандомізована сніжинка, Сніжинка “навпаки” із заданою кількістю ітерацій (від 0 до 7) та обраним кольором.
   2. Побудова множини Мандельброта для заданого степеня (від 2 до 15) та заданої кількості ітерацій (від 1 до 100).
   3. Збереження (завантаження) зображення побудованого фрактала у форматі png, jpg, jpeg або jiff.
2. Робота із кольорами зображення
   1. Завантаження власного зображення для подальшого виконання над ним дій.
   2. Зміна яскравості (від -100 до 100) фрагменту зображення, що задається координатами верхнього лівого кута (x, y) та розміром (ширина, висота).
   3. Зміщення значень кольорів по тьром каналам RGB (r, g, b) від -255 до 255 фрагменту зображення, що задається координатами верхнього лівого кута (x, y) та розміром (ширина, висота).
   4. Відображення на екрані значення кольору пікселя у форматі RGB(r, g, b) та XYZ(x, y, z) при наведенні курсора на піксель.
   5. Збереження (завантаження) отриманого зображення у форматі png, jpg, jpeg або jiff.
   6. Скидування раніше внесених у зображення змін.
3. Афінні перетворення
   1. Побудова на графіку трапеції, заданої через координати (x, y) чотирьох вершин.
   2. Виконання повороту трапеції відносно заданої вершини проти та за годинниковою стрілкою на N градусів та одночасне пропорційне масштабування в M разів.
   3. Збереження (завантаження) зображення отриманого графіка у форматі png.
4. Сторінка із навчальними матеріалами.
   1. Сторінка із посиланнями на навчальні відеоматеріали для самостійного опрацювання та закріплення раніше вивченого матеріалу.
   2. Сторінки із тестами для перевірки знань: сторінка із тестом на тему “Фрактали”, сторінка із тестом на тему “Кольорні моделі” та сторінка із тестом на тему “Афінні перетворення”.

## Алгоритми фракталів.

2.1. Алгоритм мовою JS для рекурсивної побудови фракталу Коха (Острів, Рандомізована сніжинка та сніжинка навпаки) заданої ітерації.

drawKochFractal(ctx, p1, p2, iterations, size, color, selectedKochFractal) {

    if (iterations === 0 || iterations === "0") { // Якщо кількість ітерацій != 0

      // Малювання сегменту (лінії)

      ctx.beginPath();

      ctx.moveTo(p1.x, p1.y);

      ctx.lineTo(p2.x, p2.y);

      ctx.strokeStyle = color;

      ctx.stroke();

      ctx.fillStyle = "rgb(0,0,255)"

    } else {

      let rand1 = 1; // змінна для рандомізації координати x

      let rand2 = 1; // змінна для рандомізації координати y

      let inverted = 1;

      if (selectedKochFractal === "randomized") { // рандомізована сніжинка

        rand1 = Math.random() \* (1 - 0.1) + 0.1;

        rand2 = Math.random() \* (1 - 0.1) + 0.1;

      } else if (selectedKochFractal === "inverted") { // сніжинка навпаки

        inverted = -1; // для інвертування координат

      }

      // Обчислення координат точки

      const p3 = {

        x: p1.x + rand1 \* (p2.x - p1.x) / 3,

        y: p1.y + rand2 \* (p2.y - p1.y) / 3,

      };

      const p4 = {

        x: p1.x + rand1 \* (p2.x - p1.x) / 2 + inverted \* (p2.y - p1.y) \* (Math.sqrt(3) / 6),

        y: p1.y + rand2 \* (p2.y - p1.y) / 2 - inverted \* (p2.x - p1.x) \* (Math.sqrt(3) / 6),

      };

      const p5 = {

        x: p1.x + rand1 \* (p2.x - p1.x) \* 2 / 3,

        y: p1.y + rand2 \* (p2.y - p1.y) \* 2 / 3,

      };

      // Рекурсивне малювання інших менших сегментів

      this.drawKochFractal(ctx, p1, p3, iterations - 1, size, color, selectedKochFractal);

      this.drawKochFractal(ctx, p3, p4, iterations - 1, size, color, selectedKochFractal);

      this.drawKochFractal(ctx, p4, p5, iterations - 1, size, color, selectedKochFractal);

      this.drawKochFractal(ctx, p5, p2, iterations - 1, size, color, selectedKochFractal);

    }

  }

2.2. Алгоритм мовою JS для ітеративної побудови множини Мандельброта заданого степеня для заданої ітерації.

drawMandelbrot() {

    const width = this.state.width; // Ширина зображення

    const height = this.state.height; // Висота зображення

    const exponent = this.props.exponent; // Степінь множини

    const maxIterations = this.props.iterations; // Кількість ітерацій

    const mainColor = this.props.color; // Колір точки

    if(maxIterations > 100 | maxIterations < 0) return;

    if(exponent > 15 || exponent < 2) return;

    const canvas = this.refs.canvas;

    const ctx = canvas.getContext('2d');

    for (let x = 0; x < width; x++) {

      for (let y = 0; y < height; y++) {

        const zx = (x - width / 2) \* 4 / width;

        const zy = (y - height / 2) \* 4 / height;

        let cRe = zx; // Дійсна частина комплексного числа

        let cIm = zy; // Уявна частина комплексного числа

        let i = 0;

        for (; i < maxIterations; i++) {

          const re2 = cRe \* cRe; // Дійсна частина комплексного числа

          const im2 = cIm \* cIm; // Уявна частина комплексного числа

          if (re2 + im2 > 4) break; // Умова виходу

          const newRe = Math.pow(re2 + im2, exponent / 2) \* Math.cos(exponent \* Math.atan2(cIm, cRe)) + zx;

          const newIm = Math.pow(re2 + im2, exponent / 2) \* Math.sin(exponent \* Math.atan2(cIm, cRe)) + zy;

          cRe = newRe;

          cIm = newIm;

        }

        // Отримання кольору пікселя залежно від числа ітерацій

        const color = this.getGradientColor(i, maxIterations, mainColor);

        ctx.fillStyle = color;

        ctx.fillRect(x, y, 1, 1);

      }

    }

  }

## Анотація кольорових моделей

* 1. Модель sRGB(Standard RGB(Red, Green, Blue))

RGB - це адитивна кольорова модель, що використовує комбінацію червоного (Red), зеленого (Green) та синього (Blue) для утворення кольорів. Кожен колір представлений числовим значенням від 0 до 255 (в бітових системах) або від 0 до 1 (у вигляді дробу).

Використання: Використовується у комп'ютерних моніторах, телевізорах, фотокамерах та програмах для обробки зображень.

Переваги:

* Простота використання: Ця модель дуже популярна для відображення кольорів на екранах пристроїв через свою простоту в роботі з пікселями.
* Придатність для світлодіодних пристроїв: RGB відмінно підходить для роботи з сучасними світлодіодними технологіями.
* Добре відображає кольори для сенсорних пристроїв: RGB часто використовується для побудови кольорових моделей у сенсорних пристроях, таких як камери та дисплеї.

Недоліки:

* Обмежена гама кольорів: RGB не може точно відобразити всі кольори спектра, а саме насичені зеленосині, оскільки гама кольорів обмежена трьома основними кольорами.
* Залежність від пристрою: Кольори можуть відрізнятися на різних пристроях через різні характеристики дисплеїв та інших факторів.
* Обмеженість у застосуванні, лише на пристроях, які працюють за принципом випромінювання.
  1. XYZ D65/2° (CIE 1931 Color Space)

XYZ – це еталонна колірна модель, колірний простір, розроблений Міжнародною комісією з освітлення (CIE), що базується на сприйнятті кольорів людиною. Цей колірний простір надає абстрактну трикомпонентну модель, яка описує колір як суму трьох значень X, Y та Z. Зазвичай, він використовується як початковий для перетворення між різними колірними просторами. Відзначимо, що визначення колірних просторів XYZ включає у себе параметри освітлення та сприйняття кольору людиною.

Щодо позначення "XYZ D65/2°", це вказує на параметри стандарту. D65 вказує на стандартне джерело світла (приблизно відповідає денному світлу). Число "2°" стосується кутового розділу кольорового простору, оскільки в спектральних дослідженнях використовується стандартний кут огляду 2 градуси.

Граничні значення для XYZ D65/2°:

X = [0, 94.811], Y = [0, 100], Z = [0, 107.304].

Переваги:

* Стандартизація: XYZ розроблена Міжнародною комісією з освітлення (CIE) і є міжнародним стандартом для опису кольорів. Це дозволяє використовувати модель у наукових та технічних дослідженнях, що стосуються кольорів.
* Незалежність від пристроїв: XYZ не залежить від конкретних характеристик пристроїв (наприклад, дисплеїв), тому вона використовується для аналізу кольорів незалежно від способу їх відображення.
* Точність та об'єктивність: Модель XYZ базується на наукових дослідженнях і математичних принципах, тому вона забезпечує більш об'єктивний опис кольорів з точки зору фізичних вимірювань.

Недоліки:

* Основним недоліком цієї системи є те, що використовуючи її, ми можемо констатувати тільки збіг чи розходження двох кольорів, але відстань між двома точками цього колірного простору не відповідає зоровому сприйняттю відмінності кольорів.
* Не використовується безпосередньо для відображення кольорів: XYZ використовується для аналізу кольорів та перетворень між різними колірними просторами, але не використовується безпосередньо для відображення кольорів на екранах чи в інших пристроях.
* Складність інтерпретації: Значення XYZ не завжди інтуїтивно зрозумілі для звичайного користувача через їх абстрактну природу і вимагають конвертації до інших колірних просторів для практичного використання.
  1. Псевдокод для перетворення sRGB у XYZ

var\_R = ( **sR** / 255 )

var\_G = ( **sG** / 255 )

var\_B = ( **sB** / 255 )

if ( var\_R > 0.04045 ) var\_R = ( ( var\_R + 0.055 ) / 1.055 ) ^ 2.4

else var\_R = var\_R / 12.92

if ( var\_G > 0.04045 ) var\_G = ( ( var\_G + 0.055 ) / 1.055 ) ^ 2.4

else var\_G = var\_G / 12.92

if ( var\_B > 0.04045 ) var\_B = ( ( var\_B + 0.055 ) / 1.055 ) ^ 2.4

else var\_B = var\_B / 12.92

var\_R = var\_R \* 100

var\_G = var\_G \* 100

var\_B = var\_B \* 100

**X** = var\_R \* 0.4124 + var\_G \* 0.3576 + var\_B \* 0.1805

**Y** = var\_R \* 0.2126 + var\_G \* 0.7152 + var\_B \* 0.0722

**Z** = var\_R \* 0.0193 + var\_G \* 0.1192 + var\_B \* 0.9505

* 1. Псевдокод для перетворення XYZ у sRGB

var\_X = **X** / 100

var\_Y = **Y** / 100

var\_Z = **Z** / 100

var\_R = var\_X \* 3.2406 + var\_Y \* -1.5372 + var\_Z \* -0.4986

var\_G = var\_X \* -0.9689 + var\_Y \* 1.8758 + var\_Z \* 0.0415

var\_B = var\_X \* 0.0557 + var\_Y \* -0.2040 + var\_Z \* 1.0570

if ( var\_R > 0.0031308 ) var\_R = 1.055 \* ( var\_R ^ ( 1 / 2.4 ) ) - 0.055

else var\_R = 12.92 \* var\_R

if ( var\_G > 0.0031308 ) var\_G = 1.055 \* ( var\_G ^ ( 1 / 2.4 ) ) - 0.055

else var\_G = 12.92 \* var\_G

if ( var\_B > 0.0031308 ) var\_B = 1.055 \* ( var\_B ^ ( 1 / 2.4 ) ) - 0.055

else var\_B = 12.92 \* var\_B

**sR** = var\_R \* 255

**sG** = var\_G \* 255

**sB** = var\_B \* 255

## Оптимальний матричний вираз афінних перетворень

На вході маємо матрицю однорідних координат точок трапеції, A(x, y) – вершина трапеції, N – рівень масштабування та ϕ – кут повороту. План дій для афінного перетворення:

1) Переміщення початку системи координат у A(x, y) на вектор (x, y) так, щоб точка повороту стала початком координат;

2) Поворот точки на кут ϕ у додатному напрямку відносно початку координат;

3) Пропорційне масштабування точки в N разів;

4) Переміщення одержаного результату назад так (повернення системи координат), щоб точка повороту співпала з точкою A.

– початкова матриця однорідних координат точок трапеції

– матриця переміщення початку системи координат на (x, y)

– матриця повороту на кут ϕ у додатному напрямку

– матриця пропорційного масштабування в N разів

– матриця переміщення початку системи координат назад

– матриця перетворень

*–* матриця перетворених координат.

## Реалізація графічного режиму

# Результати

## Wireflow

## UI kit

## Отримані фрактали

## Вигляд програми для лабораторної роботи №3

## Вигляд програми для лабораторної роботи №4

## Текст програми з коментарями.

# Висновки

# Список використаних інформаційних джерел

1. <https://en.wikipedia.org/wiki/Koch_snowflake> - Koch snowflake: [Електронний ресурс] // Вікіпедія – вільна енциклопедія.
2. <https://plus.maths.org/content/what-mandelbrot-set> - Robert Devaney - “What is the Mandelbrot set?”: [Електронний ресурс] // Plus Magazine
3. <https://www.easyrgb.com/en/math.php> - Color math and programming code examples: [Електронний ресурс] // EasyRGB
4. <https://www.mathworks.com/discovery/affine-transformation.html#:~:text=Affine%20transformation%20is%20a%20linear,with%20non%2Dideal%20camera%20angles> – “What Is an Affine Transformation?”: [Електронний ресурс] // MathWorks
5. https://plotly.com/javascript/react/ - React Plotly.js in JavaScript [Електронний ресурс] // Plotly
6. <https://surveyjs.io/form-library/documentation/get-started-react#configure-styles> - Add a Survey to a React Application [Електронний ресурс] // SurveyJS