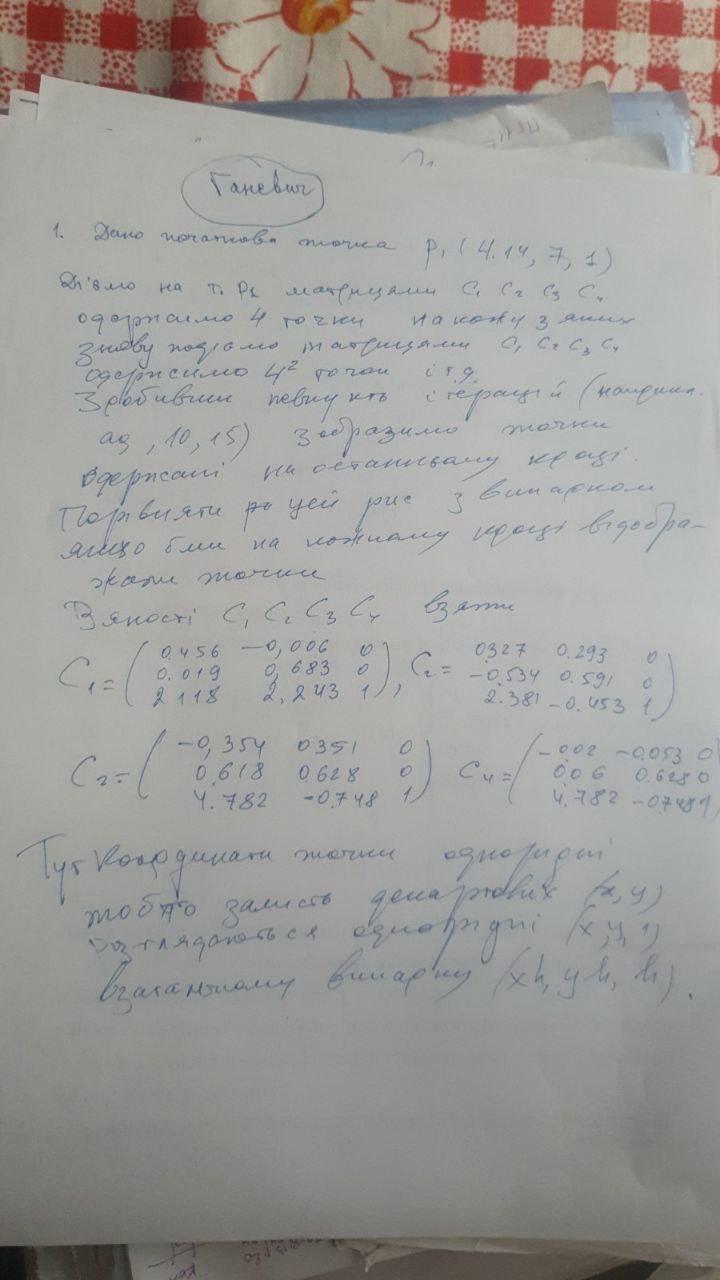
**Звіт**

з обчислюваної геометрії

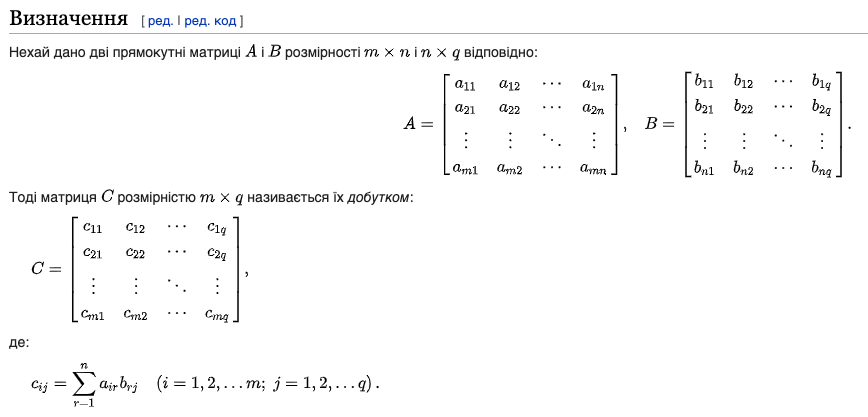
та компʼютерної крафіки

**Ганевича Вадіма**

**29.03.2022**

**Умова:**

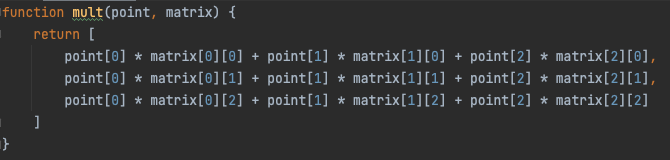
**Математичний розвʼязок:**

**Для вирішення поставленої задачі, нам необхідно множити дві матриці одна на одну.**

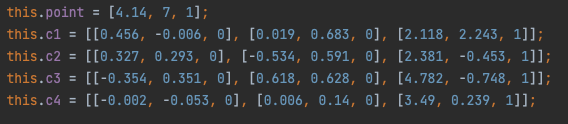
**Використаємо наступну формулу:**

**Алгоритм розвʼязку:**

**Крок 1:**

**Для побудови заданого фрактала, нам необхідна функція для множення двох матриць, реалізуємо її:**

**Крок 2:**

**Задамо статично початкову точку та матриці, якими ми діятимемо на точки:**

**Крок 3:**

**Створимо рекурсивну функцію, яка на кожному кроці рекурсії буде викликатись 4 рази сама собою.**

**Крайній випадок: глибина рекурсії**

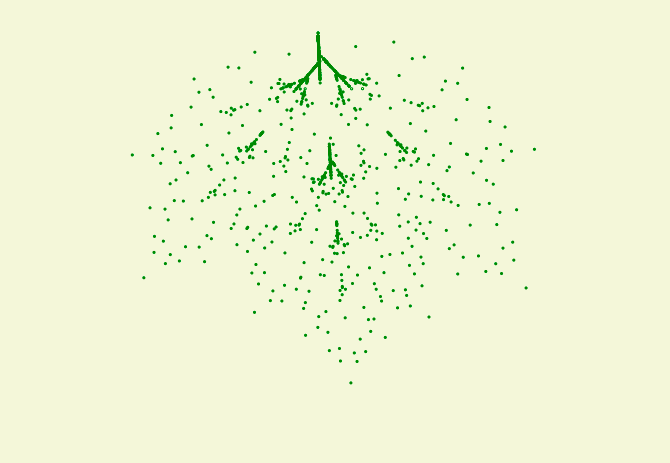
**Тіло: Виведення точки на екран, якщо вона є останньою в своєму дереві (тобто, глибина рекурсії досягла максимуму, тому на цю точку більше не діятимуть матриці)**

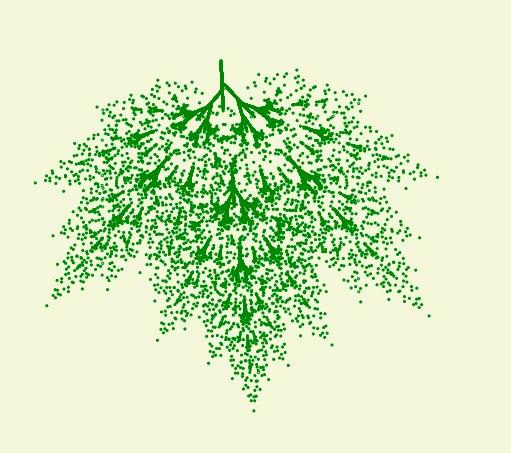
**Рекурсивний виклик: Виклик функції для кожної точки, утворених в результаті множення вхідної точки на кожну із статичних матриць. Глибина рекурсії збільшується на 1.**

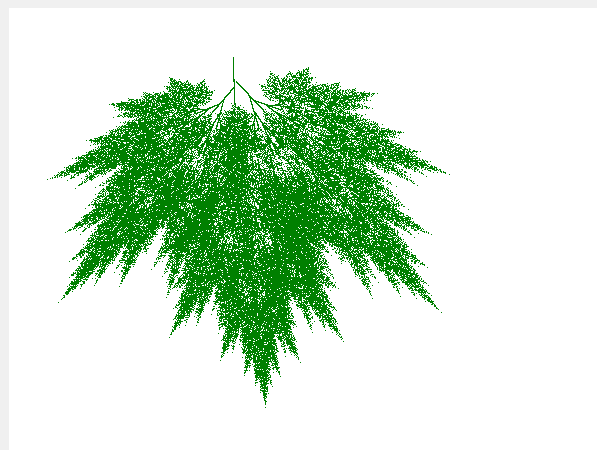
**Реалізація описаної рекурсивної функції в коді:**

**Приклади роботи програми:**

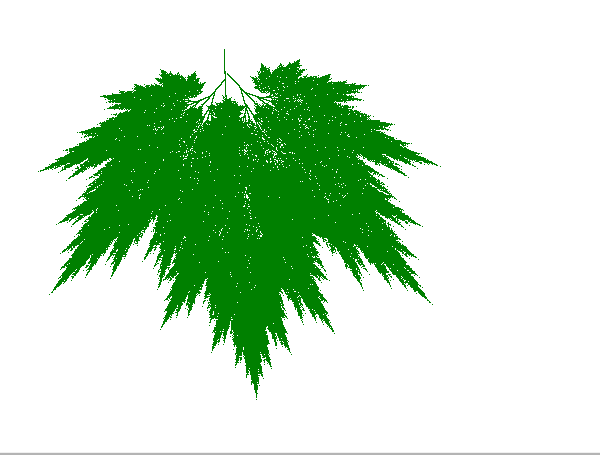
Глибина 6:



Глибина 7:

Глибина 10:

Глибина 11:



Код:

function mult(point, matrix) {

return [

point[0] \* matrix[0][0] + point[1] \* matrix[1][0] + point[2] \* matrix[2][0],

point[0] \* matrix[0][1] + point[1] \* matrix[1][1] + point[2] \* matrix[2][1],

point[0] \* matrix[0][2] + point[1] \* matrix[1][2] + point[2] \* matrix[2][2]

]

}

class Frame {

constructor(container) {

this.stage = acgraph.create(container);

this.point = [4.14, 7, 1];

this.c1 = [[0.456, -0.006, 0], [0.019, 0.683, 0], [2.118, 2.243, 1]];

this.c2 = [[0.327, 0.293, 0], [-0.534, 0.591, 0], [2.381, -0.453, 1]];

this.c3 = [[-0.354, 0.351, 0], [0.618, 0.628, 0], [4.782, -0.748, 1]];

this.c4 = [[-0.002, -0.053, 0], [0.006, 0.14, 0], [3.49, 0.239, 1]];

this.i = 0;

}

start(){

this.#recurs(this.point, 0)

***console***.log(this.i);

}

#recurs(point, step){

if (step > 5) {

return

}

if (step === 5) {

this.stage.circle(point[0] \* 50 + 200, point[1] \* 50 + 200, 1)

.stroke('green');

this.i += 1;

}

this.#recurs(mult(point, this.c1), step+1)

this.#recurs(mult(point, this.c2), step+1)

this.#recurs(mult(point, this.c3), step+1)

this.#recurs(mult(point, this.c4), step+1)

}

}

const frame= new Frame('stage-container');

***frame***.start()