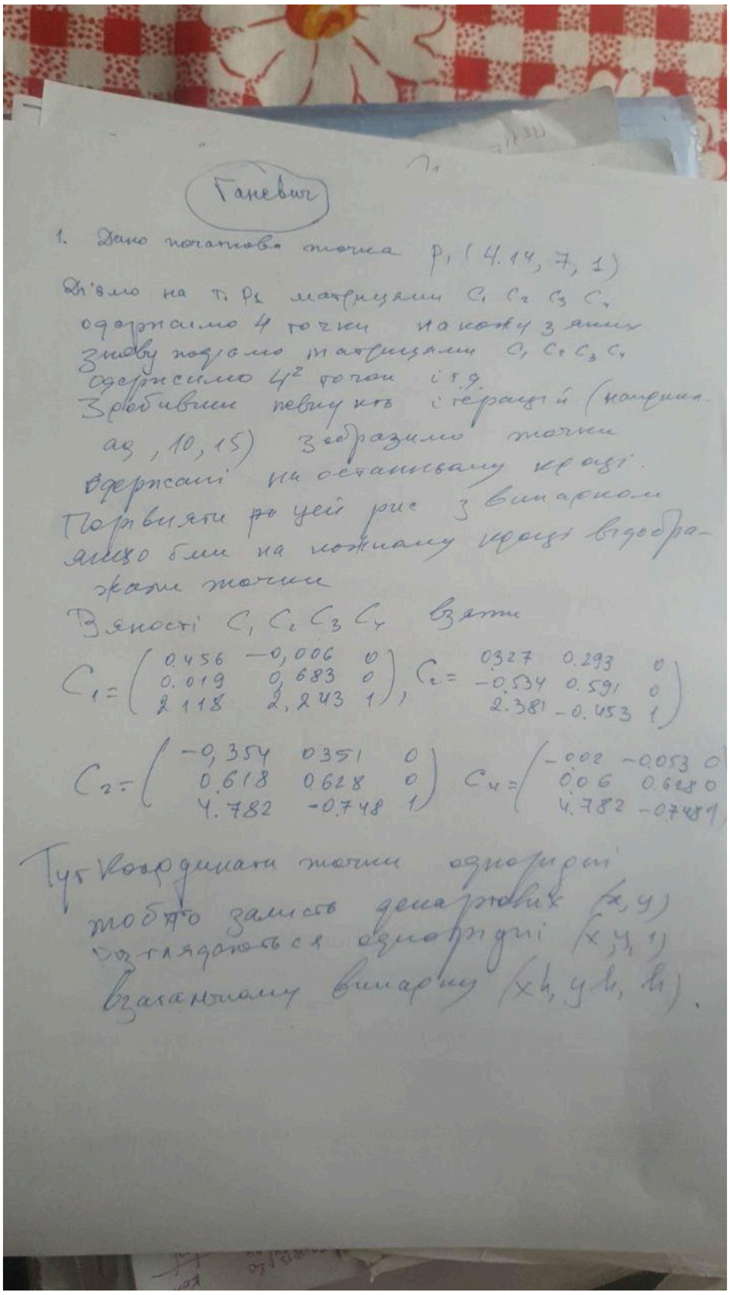
Звіт

з обчислюваної геометрії
та комп’ютерної крафіки

Ганевича Вадіма

29.03.2022

Умова:



Ганевич

1. Дано початкова точка $P_1(4.14, 7, 1)$

Ді'ємо на т. P_1 матрицями C_1 C_2 C_3 C_4 одержимо 4 точки на кожну з яких знову подіємо матрицями C_1 C_2 C_3 C_4 одержимо 4^2 точок і т.д.

Зробивши певну к-ть ітерацій (наприклад, 10, 15) зобразимо точки одержані на останньому кроці.

Порівняти до цей рис з випадком якщо були на кожному кроці відображали точки

Значення C_1 C_2 C_3 C_4 взяти

$$C_1 = \begin{pmatrix} 0.456 & -0,006 & 0 \\ 0.019 & 0,683 & 0 \\ 2.118 & 2,243 & 1 \end{pmatrix}, \quad C_2 = \begin{pmatrix} 0327 & 0.293 & 0 \\ -0.534 & 0.591 & 0 \\ 2.381 & -0.453 & 1 \end{pmatrix}$$

$$C_2 = \begin{pmatrix} -0,354 & 0351 & 0 \\ 0.618 & 0.628 & 0 \\ 4.782 & -0.748 & 1 \end{pmatrix} \quad C_4 = \begin{pmatrix} -0.02 & -0.053 & 0 \\ 0.06 & 0.628 & 0 \\ 4.782 & -0.748 & 1 \end{pmatrix}$$

Тут координати точки однорідні тобто замість декартових (x, y) розглядаються однорідні $(x, y, 1)$ в загальному випадку (xh, yh, h).

Математичний розв’язок:

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

Використаємо наступну формулу:

Нехай дано дві прямокутні матриці A і B розмірності $m \times n$ і $n \times q$ відповідно:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1q} \\ b_{21} & b_{22} & \cdots & b_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nq} \end{bmatrix}.$$

Тоді матриця C розмірністю $m \times q$ називається їх *добутком*:

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1q} \\ c_{21} & c_{22} & \cdots & c_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mq} \end{bmatrix},$$

де:

$$c_{ij} = \sum_{r-1}^{n} a_{ir} b_{rj} \quad (i = 1, 2, \dots m;\ j = 1, 2, \dots q).$$

Алгоритм розв’язку:

Крок 1:

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

```
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]
    ]
}
```

Крок 2:

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

```
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]];
```

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)

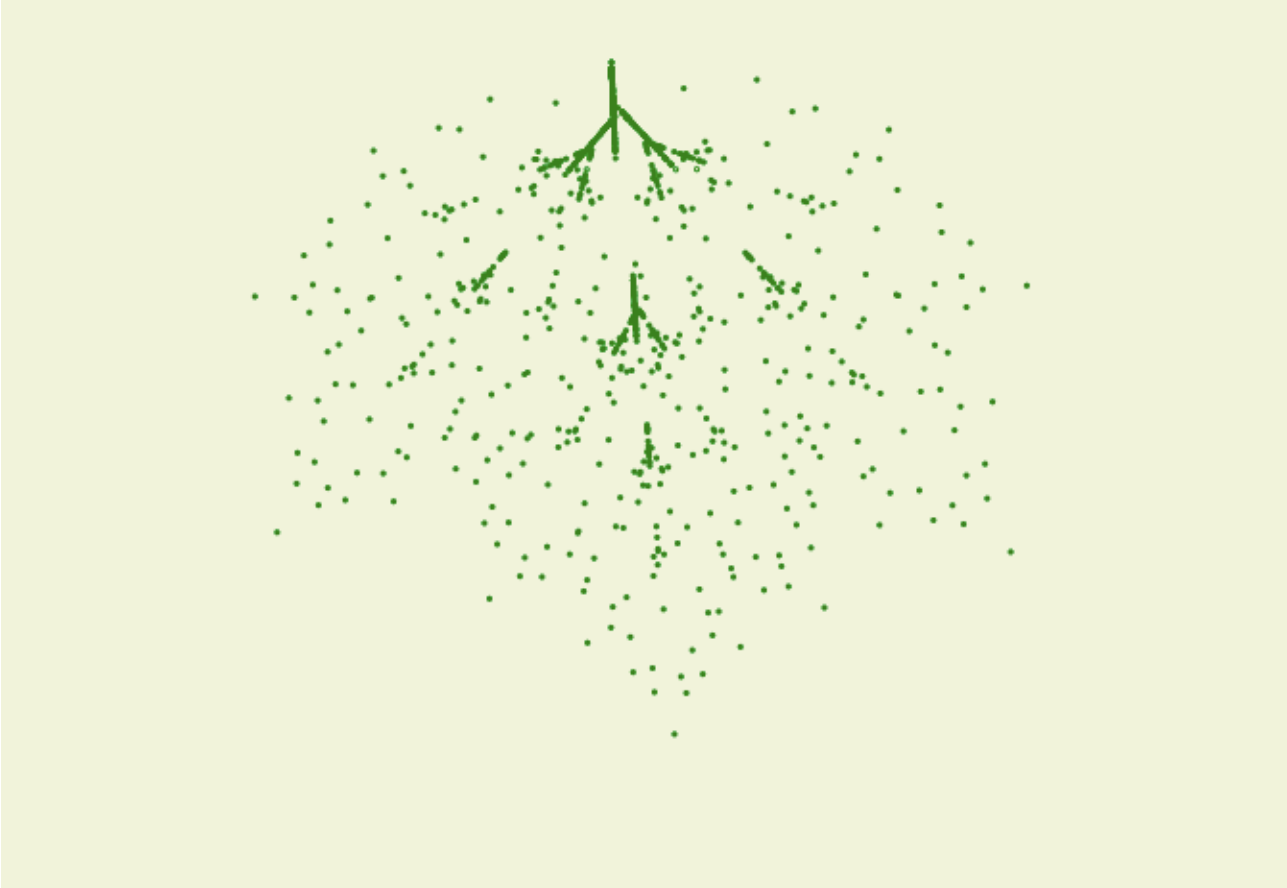
Рекурсивний виклик: Виклик функції для кожної точки, утворених в результаті множення вхідної точки на кожну із статичних матриць. Глибина рекурсії збільшується на 1.

Реалізація описаної рекурсивної функції в коді:

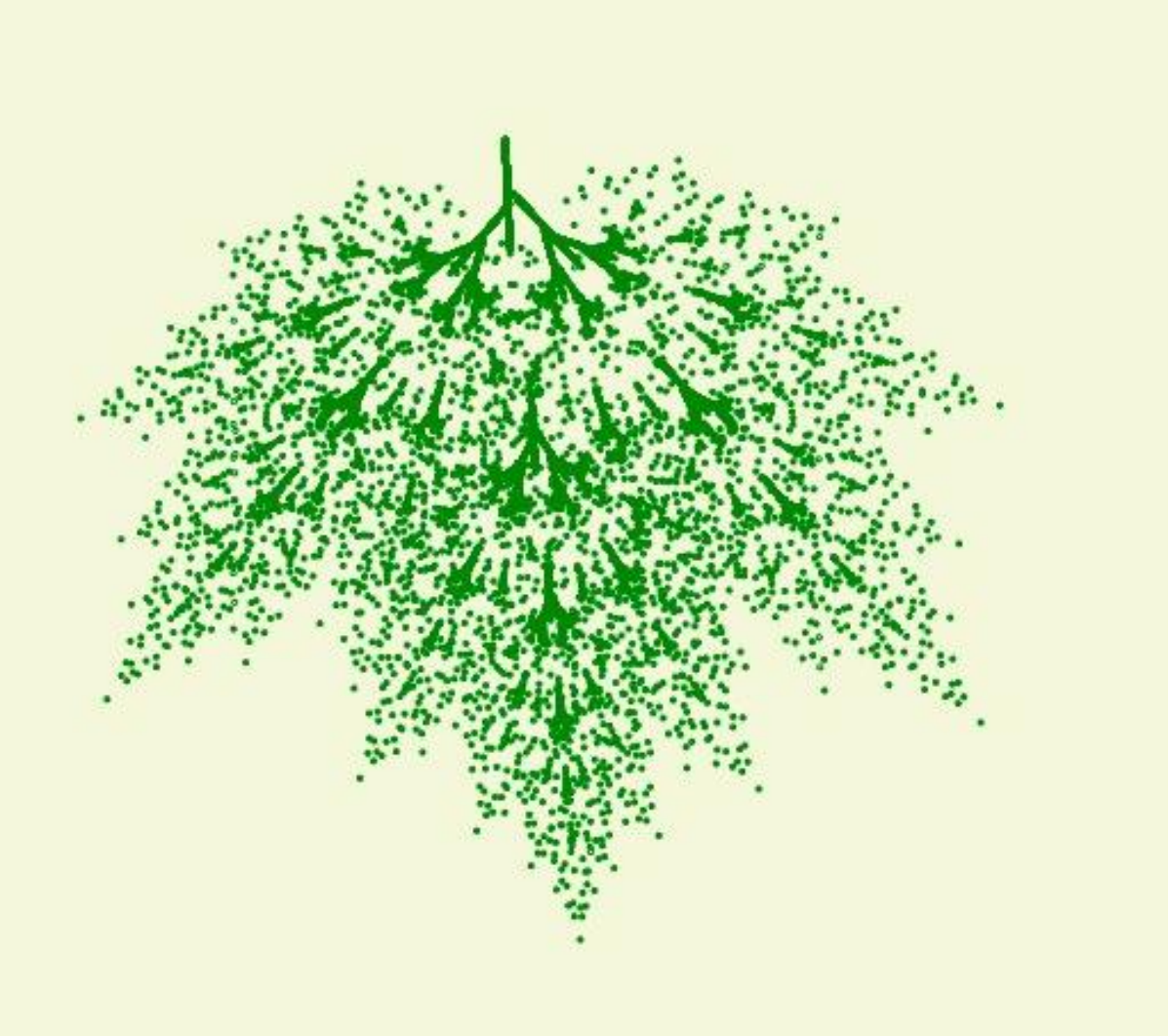
```
#recurs(point, step){
    if (step > 7) return
    if(step === 7 ) {
        this.stage.circle(point[0] * 50 + 200, point[1] * 50 + 200, 1)
            .stroke('green');
        return;
    }

    this.#recurs(mult(point, this.c1),  step: step+1)
    this.#recurs(mult(point, this.c2),  step: step+1)
    this.#recurs(mult(point, this.c3),  step: step+1)
    this.#recurs(mult(point, this.c4),  step: step+1)
}
```

Приклади роботи програми:

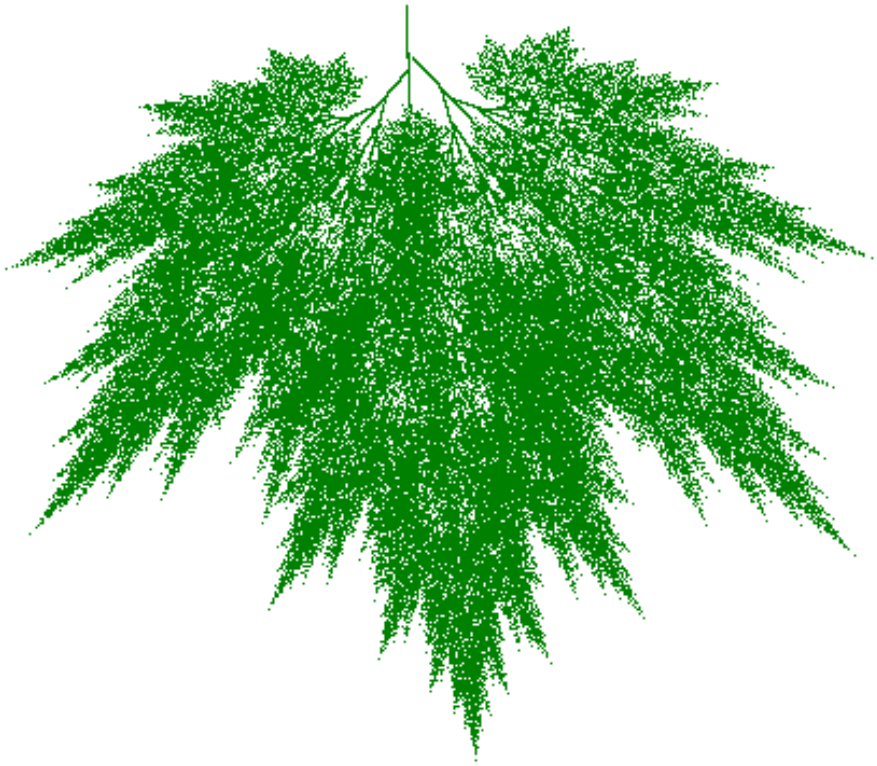
Глибина 6:



Глибина 7:



Глибина 10:



Глибина 11: