Софийски университет
„Св. Климент Охридски“
Факултет по математика и информатика

Специалност:

„Информационни системи“ Група 3, курс 2
Дисциплина: „Фрактали“

КУРСОВ ПРОЕКТ

Тема: Снежинка на Кох(Koch’s snowflake)

Изготвил:
Николай Николаев Голосманов (ФН: 71721)

Преподавател:
Доц. Д-р Милко Такев

София
Летен семестър 2017/2018

Съдържание

Снежинка на Кох

Рекурсивно изчертаване на фрактал, който се съдържа в себе си.

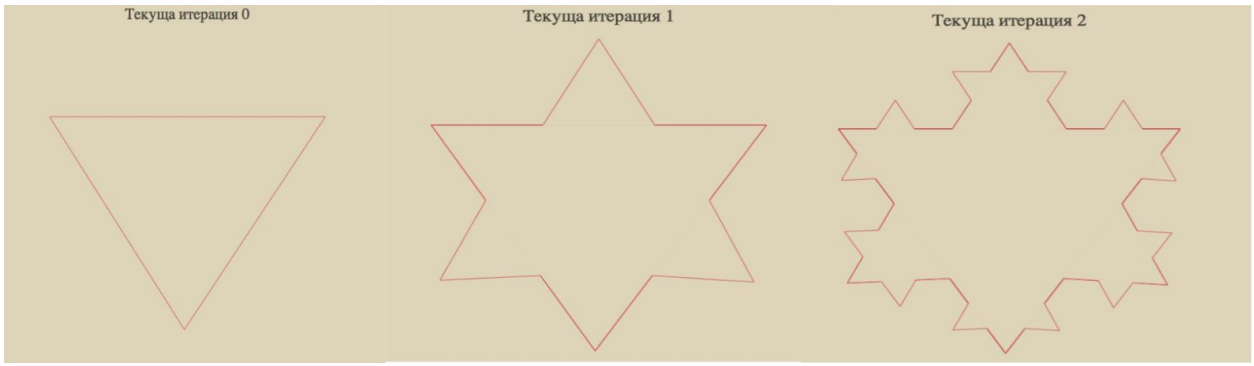
Снежинката на Кох е фрактал описан от шведския математик Нийлс Фабиан вон Кох през 1904 година в неговия труд върху теорията на числата. Снежинката на Кох е един от най-известните фрактали и е лесен за разбиране.

Крива на Кох

Кривата на Кох представлява в началото една права, която се разделя на три равни части. Средната част се използва за основа на равностранен триъгилник, след което основата на този триъгълник се премахва. Така се образуват четири Нови прави. На следващата стъпка всяка от тези прави се разделя аналогично на още 4 прави. Така кривата на Кох може да расте безкрайно.

Снежинката на Кох

Снежинката на Кох представлява в началото един равностранен триъгълник, като всяка една от страните му представлява крива на Кох.



Реализация на изчертаването на снежинката на Кох

Технологии и системни изисквания

За реализацията на фрактала са използвани съвременни уеб технологии - HTML5, JavaScript и CSS3 за оформянето на външния вид. За изчертаването на фрактала е нужно да се стартира файлът snowflake.html. За да може да се използва програмата е нужно да се отвори с последно поколение браузър като firefox, chrome или safari.

Математически изчисления

Проекта използва рекурсивен алгоритъм за изчисляване на три точки и образуване на крива на Кох. Алгоритъмът се прилага върху трите страни на равностранния триъгълник в началото на нулевата итерация, след което рекурсивно се прилага върху всяка една от новообразуваните страни. Така може да бъде прилаган безброй пъти, но поради ограничението на паметта и експоненциалното растене на функцията, съвременен браузър се справя добре с изчислението на 7 итерации. Алгоритъма започва като изчертае равностранен триъгълник. Образуването на следващите итерации става чрез навигационните бутони следваща и предишна итерация, което подтиква ново изчертаване на фрактала с дълбочина на рекурсията съответната текуща итерация. Нека да разгледаме изчисленията, които се извършват за една от страните на равностранния триъгълник (снежинката на Кох в нулевата итерация). Изчисленията са напълно аналогични за останалите страни. Нека да разгледаме страната AB. Тя се приема за права с начална точка A с координати (50, 150) и крайна точка BC Координати (500,150). Първо се намират координатите на точката C, която разделя правата AB в съотношение 1:2. От аналитичната геометрия следва че координатите на точката C, която е част от правата AB са следните:

x = (2xa + xb) 13, y = (2ya + yb) 13, където xa и xb са съответно абсцисите на точките A и B, ya и yb са ординатите съответно на точките A и B. Следва намирането на Координатите на точката D, която разделя правата AB в отношение 2:1. Правата AB Може да се приеме и за вектор, като точката D разделя вектора BA също в Отношение 1:2, което означава че може да се приложат същите изчисления като за Точката C, само че с обратния вектор. Програмното реализиране е чрез използването на прототип в езика javaScript за задаването на обект представляваш Точка с координати Хиуи помощни функции за умножаване на координатите със скалар, събиране на координатите и деленето и на двете координати на скалар.

```
var C = divide(add(multiply(A, 2), B), 3);

function multiply(v, num){
return { x: v.x * num, y: v.y * num};
};

function add(a, b){
return { x: a.x + b.x, y: a.y + b.y };
};
```

```
function divide(v, num){
return { x: v.x/num, y: v.y / num };
};
```

И съответно за точката D се извършва същото изчисление, но с обратния вектор BA

```
var D = divide(add(multiply(B, 2), A), 3);
```

Следва намирането на точката F, която служи за връх на равностранен триъгълник със основа CD.

Точката F лежи върху права, която е перпендикулярна на AB и минава през средата на AB. Тоест точката F лежи върху нормализирания вектор на AB. Ако Me средата на вектора AB, то дължината на MF може да се изрази като височина в равностранен триъгълник със страна 1/3 от AB, което е AB√(3/6).

Следователно за намирането на координатите на точката F имаме следната поредица от действия:

Намираме средата на AB точката M, която дели отсечката в отношение 1:1

```
var M = divide(add(A, B), 2)
```

Намираме единичния вектор AB

```
var E = divide(minus(M, A), length(M, A));
```

Където дължината се изчислява по формулата за разстояние между две точки.

```
function length(a, b){
return Math.sqrt(Math.pow(a.x - b.x, 2) + Math.pow(a.y - b.y, 2));
};
```

След което намираме нормализирания вектор на единичния вектор E

```
var N = normala(E);
```

където нормализирания вектор се намира по формулата:

```
function normala(v){
return { x: v.y, y: -v.x }
}
```

Точката F се намира върху вече намерения нормализиран вектор N, но както вече уточнихме трябва да е в точката Ми да е с дължина AB√(3/6), затова имаме следното изчисление:

```
var F = add(multiply(N, Math.sqrt(3)/6 * length(B, A)), M);
```

За красивата форма на фрактала остана само да се изчертаят четирите линии определяни от точките C, D и F и да се премахне основата на равностранния триъгълник правата CD. Изчертаването на четирите прави става рекурсивно, ако имаме още итерации всяка от тези прави ще бъде разделена на три равни части и ще се намери върха на равностранния триъгълник. Премахването на основата в конкретната реализация е осъществена чрез начертаването на права с цвят съвпадащ с фона.

За повече интерактивност е добавена и анимация, по време на изчертаването на фрактала. Цялото изчисление е осъществено вътре в клоужър, който връща функцията, която изчертава фрактала. Целта е да се предотврати извикването на помощните изчислителни функции извън файла с кода на фрактала. Кода за изчертаването на снежинката на Кох се намира във файла fractal.js

Свойства на Снежинката на Кох

Кривите на Кох, които изграждат фрактала на негово име са известни като безкрайни криви – на всяка итерация кривата се разделя на 4 прави всяка с дължина 1/3 от дължината на оригиналната крива, следователно новополучената дължина се е увеличила с 1/3. Периметърът на Снежинката на Кох за стъпка k е $(4/3)*P_{k-1}$

, където P_{k-1} е периметъра от предишната стъпка.

Фракталната размерност е log4/log3 = 1.26

В началото фрактала представлява равностранен триъгълник с лице √3/4*s^2, където е страната на този равностранен триъгълник На следващата итерация към фрактала се добавят нови три равностранни триъгълника със страна s/3 Лицето на новополучената фигура е очевидно сбора на старото лице и новополучените три триъгълника:

$$\sqrt{3}/4*s^2 + \sqrt{3}/4*(s/3)^2.$$

След преобразуване се получава:

$$\sqrt{3}/4*s^2 * (1+⅓).$$

Ако направим абсолютно аналогичните разсъждения и изчисления ще намерим

лицето на фигурата в следващата итерация:

$$\sqrt{3}/4*s^2 * (1 + ⅓ + 3*4/9^2)$$

Лесно е да забележим, че имаме геометрична прогресия относно лицата на малките равностранни триъгълничета.

За лицето в итерация n имаме:

$$\frac{\sqrt{3}}{4}s^2\left(1+\sum_{k=1}^{n}\frac{3\cdot 4^{k-1}}{9^k}\right)$$

N клони към безкрайност, следователно получаваме:

$$\lim_{n\to\infty}\frac{\sqrt{3}}{4}s^2\left(1+\sum_{k=1}^{n}\frac{3\cdot 4^{k-1}}{9^k}\right)$$

След изчисление на израза намираме, че лицето на снежинката на Кох е:

$$\frac{\sqrt{3}}{4}s^2\left(1+\frac{3/9}{1-4/9}\right)$$

Което означава след преобразуване на израза, че лицето е: ((2 * √3) / 5) * s^2.

Изводът е, че снежинката на Кох има безкраен периметър, но крайно лице.

Среда за програмиране

За изпълнението на заданието са използвани: javaScript, HTML, CSS.

Логика:

На всяка стъпка текущата линия се разделя на три като се използва рекурсивното правило, разгледано в началото на доклада.

Размерност на фрактала: $\log_3 4$

Програмен код

```
var iteration = 0;

function next_iteration(step)
{
var iterationCounter = document.getElementById("iterationCounter");
iteration = iteration + step < 0 ? 0 : iteration + step ;
iterationCounter.innerHTML = iteration;
init(iteration);
}

var init = (function(){

var W = 550;
var H = 510;
var timeout = 100;
var strokeColor = "#ce534d";
var bgColor = "#ded4b9";
var canvas;

return function(iteration){
        canvas = document.getElementById("imageView");

   context = canvas.getContext('2d');
   context.beginPath();
   context.stroke();
   context.closePath();

        context.clearRect(0, 0, W, H);

   kochCurve({x: 50,  y: 150}, {x: 500, y: 150}, iteration);
   kochCurve({x: 500, y: 150}, {x: 270, y: 490}, iteration);
```

```
    kochCurve({x: 270, y: 490}, {x: 50,  y: 150}, iteration);

};

function kochCurve(A, B, iteration){

    if (iteration < 0){
        return null;
    }

    var C = divide(add(multiply(A, 2), B), 3);
    var D = divide(add(multiply(B, 2), A), 3);
    var M = divide(add(A, B), 2);

    var E = divide(minus(M, A), length(M, A));
    var N = normala(E);

    var F = add(multiply(N, Math.sqrt(3)/6 * length(B, A)), M);


    setTimeout(function(){
        line(A, B, strokeColor);

        if (iteration !=0){
            setTimeout(function(){
                for (var i = 0; i < 7; i++)
                    line(C, D, bgColor);
            }, timeout);
        };

        kochCurve(A, C, iteration-1);
        kochCurve(C, F, iteration-1);
        kochCurve(F, D, iteration-1);
        kochCurve(D, B, iteration-1);

    },timeout);


};

function normala(v){
    return { x: v.y, y: -v.x }
}
```

```
function multiply(v, num){
    return { x: v.x * num, y: v.y * num};
};

function divide(v, num){
    return { x: v.x/num, y: v.y / num };
};

function add(a, b){
    return { x: a.x + b.x, y: a.y + b.y };
};

function minus(a, b){
    return {
        x: a.x - b.x, y: a.y - b.y };
};

function length(a, b){
    return Math.sqrt(Math.pow(a.x - b.x, 2) +
                    Math.pow(a.y - b.y, 2));
};

function line(a, b, c){
    context.beginPath();
    context.strokeStyle = c;
    context.moveTo(a.x, a.y);
    context.lineTo(b.x, b.y);
    context.stroke();
    context.closePath();
};

})();
```

Използвана литература.

- Wikipedia.org
- Github.com
- YouTube.com
- KhanAcademy
- Mathworld.wolfram.com