Софийски университет „Св. Климент Охридски“

ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

Фрактали

Снежинка на Кох

от Владимир Костов Димитров
ф.н. 44325

София, 2013 г.

Въведение

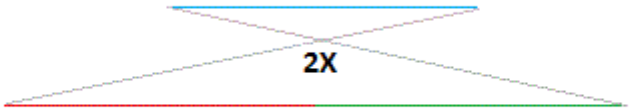
В последващите страници ще разгледаме Снежинката на Кох и една нейна реализация, но преди това нека поговорим по-подробно за това

Какво е фрактал?

Наименованието фрактал се въвежда от Манделброд и произлиза от латинската 'fractus' – разбит, за да се опишат обекти, които са твърде нестандартни за традиционната геометрия.

Фракталите са фигури с безкрайно многото детайли. Когато ги гледаме от по-близко те не стават по-прости, а се запазват толкова сложни, колкото и без увеличение. Например клон на дърво, когато се гледа с подходящо приближение прилича на цяло дърво, както и една скала прилича на цяла планина (съставена също от много скали). Свойството частите на един обект да са подобни на целият обект се нарича самоподобие (self-similarity). Има различни степени на самоподобие, понякога то не е толкова силно изразено, а е приблизително или дори статистическо. Самоподобието е характерно за всички математически фрактали.

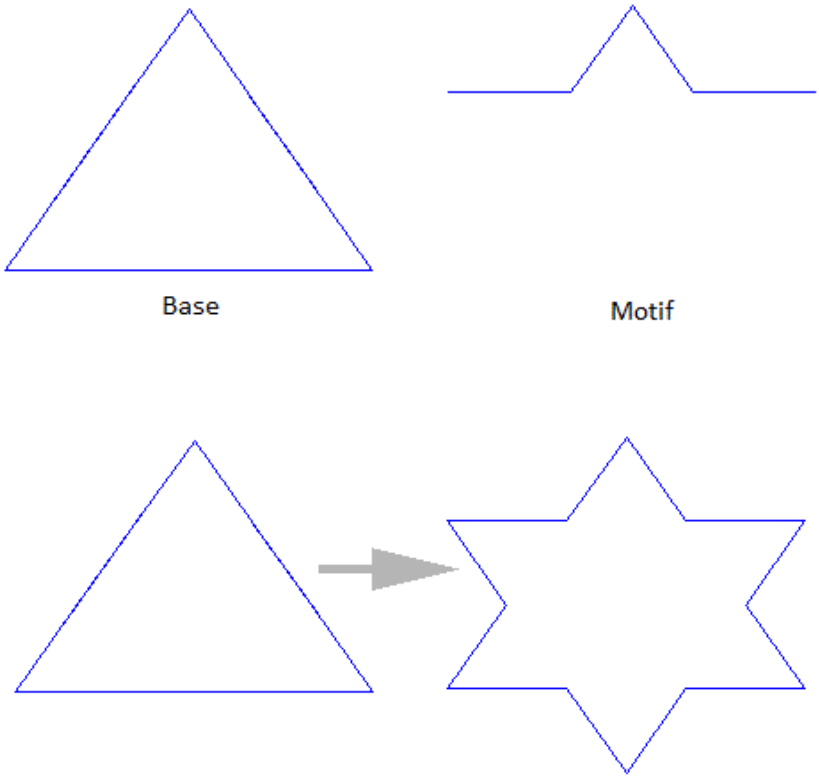
Една от основните характеристики на фракталите е размерността. Има различни дефиниции за фракталната размерност (Hausdorff, Box-counting, similarity) и те не винаги водят до еднакъв резултат. Similarity размерността е лесен вариант за пресмятане на фрактална размерност, но е приложима само за строго самоподобни фрактали. Например да предположим, че имаме отсечка ако я погледнем през двойно увеличение можем да забележим, че е съставена от две подобни на нея отсечки.



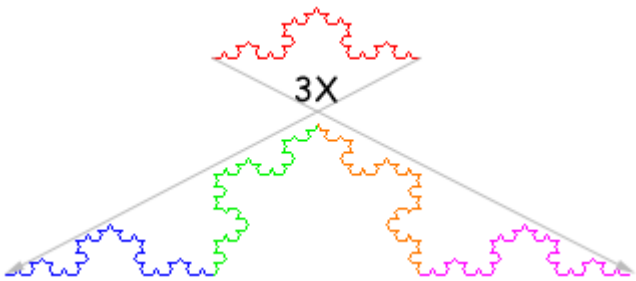
Ако similarity размерността е D, N – броят на идентичните фигури и m – кратността на приближението, то D = log N/log m.

Снежинката на Кох (Koch Snowflake) е един от най-известните фрактали. Носи името на шведският математик Helge von Koch и произлиза от така наречената крива на Кох (три криви на Кох разположени по страните на равностранен триъгълник).

Кривата и снежинката на Кох са base-motif фрактали. Въпросните base-motif фрактали се конструират като всяка отсечка на основата им (base) се замени с дадена форма състояща се пак от отсечки (motif). След което субституцията се повтаря за всички нови отсечки и т. н.



При нея се наблюдава строга себеподобност и similarity размерността се пресмята лесно чрез горната формула.



N = 4, m = 3, D = log 4 / log 3 = 1.26. Забелязва се, че размерността не е цяло число, нещо неприсъщо за стандартната геометрия, в която имаме само едно, две, три и т. н. целочислено мерни фигури. Базата на снежинката е изградена от отсечки, според нормалната представа за размерност (топологична), отсечката е едномерна фигура, тоест топологичната размерност на фрактала е 1, докато similarity размерността е 1.26. Според Манделброд именно това е една от основните характеристики на фракталите, фракталната им размерност е строго по-голяма от топологичната.

На всяка итерация фракталът увеличава дължината си по 4/3 и така до безкрайност.

Реализация

Може да се види в приложеният диск или на http://vladko.org/~vkdimitrov/Koch-Snowflake/ използвани са HTML5, CSS3 и JavaScript, от където и идва необходимостта от съвременен browser. Приложението се състой от 3 файла index.html, layout.css и script.js. За да се стартира

се отваря index.html. Самата реализация е в script.js. Дефинирани са следните обекти

Point

```
/*
 * point object
 * @param {int} x coodrinates
 * @param {int} y coordinates
 * @returns {point}
 */
function Point(x, y)
{
	this.X = x;
	this.Y = y;
}
```

Vector

```
/*
 * vector obeject represents directed segment
 * @param {Point} pointFrom
 * @param {Point} pointTo
 * @returns {Vector}
 */
function Vector(pointFrom, pointTo)
{
	this.start	= pointFrom;
	this.end	= pointTo;
	this.len =
Math.sqrt(Math.pow((this.end.Y - this.start.Y),
2) + Math.pow((this.start.X - this.end.X), 2));
	//vector lenght
	this.ang = Get_Direction(this.start,
this.end);
```

описващи точка и отсечка.
Започва се с пресмятане и изчертаване на базата на фрактала, отсечките на която се вкарват в масив lines.

```
//init base
	var lines = new Array();
	/*
	 *	 3
	 *	 ^
	 *	/ \
	 *	.   \
	 *	1--->2  a(1,2),b(2,3),c(3,2)
	 */
	//base bottom line
	    var tmpPoint    = new Point(Math.round(center.X - side/2), Math.round(center.Y +
	                        ((Math.sqrt(3)/2)*(1/3)*side)));
	    var tmpPoint2 = new Point(Math.round(center.X + side/2), Math.round(center.Y +
	                        ((Math.sqrt(3)/2)*(1/3)*side)));
	                    var a = new Vector(tmpPoint, tmpPoint2);
	lines.push(a);

	//base right side
	     var tmpPoint3 = new Point(Math.round(center.X), Math.round(center.Y -
	                        ((Math.sqrt(3)/2)*(2/3)*side)));
	var b = new Vector(tmpPoint2, tmpPoint3);
	lines.push(b);

	//base left side
	var c = new Vector(tmpPoint3, tmpPoint);
	lines.push(c);
```

Следва обхождането и заместването им с мотива получените в резултат нови отсечки се

запазват в масив motifLines. Преди следващата итерация копираме motifLines в lines и го изчертаваме. За да е коректен резултата трябва да се изтрият основите на равнобедрените триъгълници получени след субституцията. Това става чрез изчертаването им наново с бял цвят.

```
for(i in lines)
{
        /*
         * from one line we'll make 4 others with motif
         *                                  .nextP3
         *                                 / \
         *                                /   \
         *                               /     \
         * nextP .-----.nextP2 .-----.nextP5
         *                                        nextP4
         */
        var nextP = new Point(lines[i].start.X, lines[i].start.Y);
        var nextP5 = new Point(lines[i].end.X, lines[i].end.Y);
        var nextP2 = new Point(0,0);
        var nextP4 = new Point(0,0);


        nextP2.X = Math.round((2*lines[i].start.X + lines[i].end.X) / 3);
        nextP2.Y = Math.round((2*lines[i].start.Y + lines[i].end.Y) / 3);

        nextP4.X = Math.round((lines[i].start.X + 2*lines[i].end.X) / 3);
        nextP4.Y = Math.round((lines[i].start.Y + 2*lines[i].end.Y) / 3);

        var nextP3 = new Point(nextP2.X, nextP2.Y);

        var ang = lines[i].ang + angle;
        var len = lines[i].len/3;//alert(len);

        nextP3.X += Math.round(len * Math.cos(ang));
        nextP3.Y += Math.round(len * Math.sin(ang));

        //put new vectors into array
        var motif1 = new Vector(nextP, nextP2);
        var motif2 = new Vector(nextP2, nextP3);
        var motif3 = new Vector(nextP3, nextP4);
        var motif4 = new Vector(nextP4, nextP5);
        motifLines.push(motif1);
        motifLines.push(motif2);
        motifLines.push(motif3);
        motifLines.push(motif4);
        //delete middle part
        var middle = new Vector(nextP2, nextP4);
        undraw_vector(ctx, middle);
}
lines = motifLines;
```

Използвани източници:

[1] Fractals Unleashed http://library.thinkquest.org/26242/
[2] K. Falconer Fractal Geometry, Johtn Wiley & Sons, 1990
[3] A turtle in a fractal garden http://www.math.sunysb.edu/~scott/Book331/