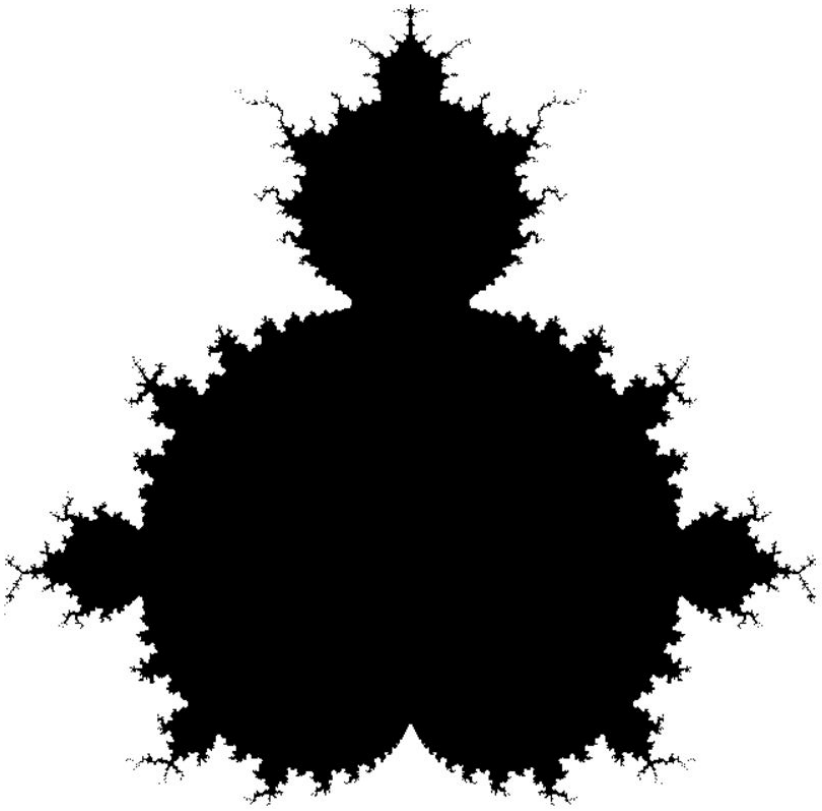
FRACTALES



Darío Rodríguez Hernández

Fundamentos de la Informática Gráfica

Escola Tècnica Superior d'Enginyeria

ÍNDICE

Introducción

Un fractal se trata de un conjunto (ya sea un dibujo, datos o cualquier otro tipo) que muestra autosimilitud a cualquier escala. Un fractal está formado por el mismo tipo de estructura sin importar a qué nivel de profundidad estemos.

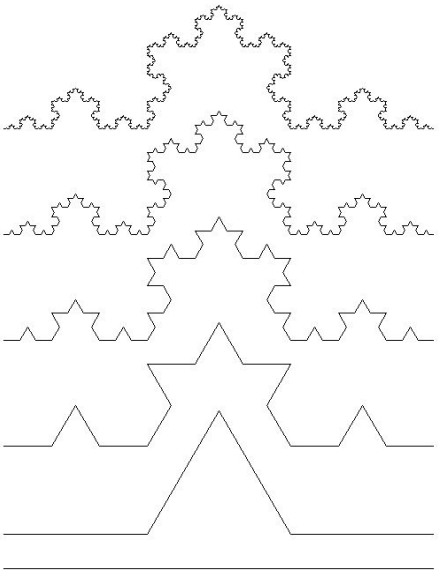
Imaginemos que nos encontramos ante un dibujo fractal ideal. Si aumentamos el dibujo una cierta escala veremos el mismo tipo de estructura. Si una vez aumentado, volvemos a realizar esta operación nos encontraremos de nuevo en la misma situación. Podríamos seguir aumentando la escala aumentando infinitas veces y el dibujo fractal seguiría mostrando el mismo tipo de estructura.

Esta definición de fractal es útil para comprender el concepto, pero en la realidad encontramos gran diversidad de estructuras fractales ya sean naturales (flores, nubes, minerales…) o artificiales (arte, estructuras de datos o incluso algoritmos de compresión) y todas ellas son finitas.

Computacionalmente se pueden representar gráficos fractales finitos. Es posible crear fractales que al aumentar la escala aparentemente sean infinitos desarrollando el fractal conforme se aumenta una zona específica pero realmente estamos tratando con fractales finitos con un nivel de profundidad determinado *n*. El trabajo de este documento se enfoca en el uso de fractales en gráficos, para ello se tratan especialmente tres de los algoritmos más significativos y conocidos como son los algoritmos para desarrollar la *curva de Koch*, el *triángulo de Sierpinski* y el algoritmo de *Mandelbrot.* No obstante cabe destacar el algoritmos del *fractal de Julia (conjuntos de Julia)* el cual guarda mucha similitud con *Mandelbrot.*

Curva de Koch

La curva de Koch es un fractal muy simple y sencillo de representar. Está formado a partir de un segmento dividido en 3 partes iguales. La parte central del segmento es la base de un triángulo equilátero, pero esta base no es representada. Por lo tanto la estructura resultante es una recta, un pico y una recta.



Si repetimos este proceso en cada parte del segmento dividido sucesivamente crearemos una estructura más compleja con el mismo criterio estructural.

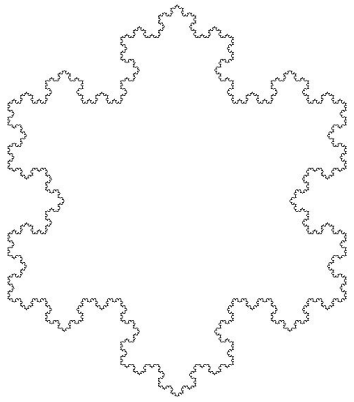
Si queremos crear un algoritmo que dibuje este fractal, lo primero será crear la estructura base. Supongamos que *A* (avanzar) dibuja una línea recta de ⅓ de la longitud del segmento total, *I* (izquierda) gira la dirección de dibujo en 60º y *D* (derecha) en -60º. La secuencia que deberíamos seguir es *AIADDAIA*.

En esta situación ya tenemos la estructura base pero para crear un fractal, en cada nuevo segmento debe haber la misma estructura que acabamos de crear. Si nos fijamos en la secuencia, cada vez que llamamos a *A* se dibuja un segmento, por lo tanto, deberíamos sustituir cada *A* de la secuencia por la secuencia entera. Quedaría *AIADDAIAIAIADDAIADDAIADDAIAIA*.

Hacer este proceso iterativamente sería un trabajo muy tedioso. Así que la forma más eficiente es usar un método recursivo en el cual cada llamada a *A* vuelve a llamar a la secuencia completa. Obviamente, al tratarse de un método recursivo debemos crear una condición de parada para no crear un bucle infinito (que idealmente así debería de ser para crear un fractal ideal matemático, pero no es lo que pretendemos ya que computacionalmente es imposible), esta condición será el nivel de profundidad que se desee que tenga el fractal.

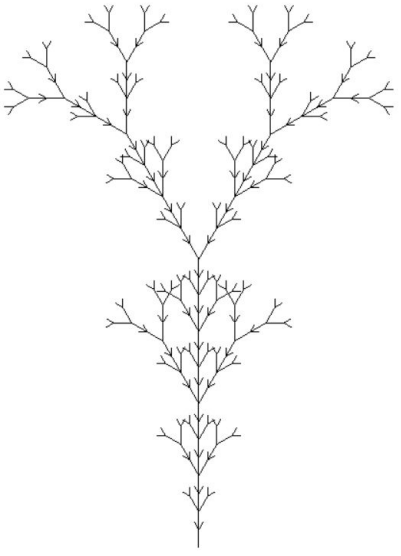
En pseudocódigo el algoritmo es tal que así, siendo *cursor* el objeto que guarda la posición y dirección de dibujado, *long* la longitud del segmento y *n* el nivel de profundidad:

```
dibujar_koch(cursor, long, n)
si n = 0
      avanzar(cursor, long) // Solo al llegar al caso base dibuja una línea.
finsi
sino
      dibujar_koch(direccion, long / 3, n - 1) // Recursividad.
      cursor.angulo ← cursor.angulo + 60
      dibujar_koch(direccion, long / 3, n - 1)
      cursor.angulo ← cursor.angulo - 120
      dibujar_koch(direccion, long / 3, n - 1)
      cursor.angulo ← cursor.angulo + 60
      dibujar_koch(direccion, long / 3, n - 1)
finsino
```



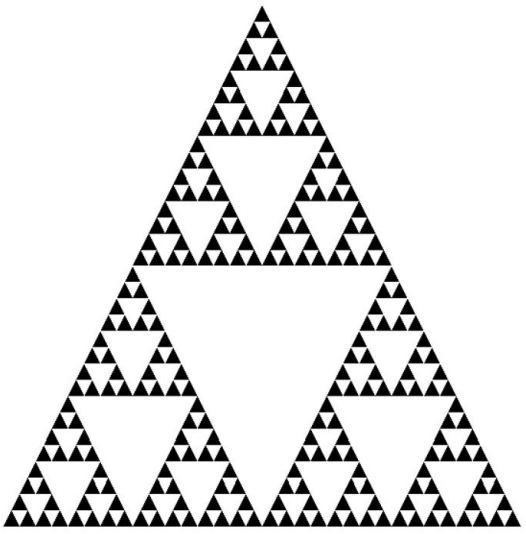
El ejemplo más popular de figura creada a partir de este algoritmo es el *Copo de Nieve de Koch* resultante de aplicar este algoritmo a cada uno de los segmentos de un triángulo equilátero.

Es posible crear infinidad de figuras diferentes cambiando levemente el algoritmo. Por ejemplo, podemos añadir al algoritmo la capacidad de recordar posiciones antiguas para volver a ellas y seguir dibujando desde esa posición. Usando esta nueva implementación podríamos crear estructuras parecidas a árboles. Considerando que *P1* agrega los valores del cursor a la una pila y *P2* lee el valor de la cima de la pila y borra el elemento, para crear el árbol comentado deberíamos seguir el siguiente patrón: *A A P1 I A P2 P1 D A P2*. Con un ángulo inicial de 90º para que el *árbol* crezca hacia arriba, con valores de giro *I* de 30º y *D* de -30º y dividiendo el segmento en ½ en lugar de en ⅓.



Triángulo de Sierpinski

Este fractal fue introducido por el matemático polaco Waclaw Sierpinski en 1919. La estructura básica de este se crea a partir de un triángulo sobre el cual se toman los puntos medios de cada lado del triángulo para crear tres triángulos más pequeños en su interior dejando en el centro un *hueco* formando un triángulo invertido.



Una vez realizado este proceso, se vuelve a hacer con cada uno de los tres triángulos creados. Obviamente, para implementar este proceso seguiremos un método recursivo.

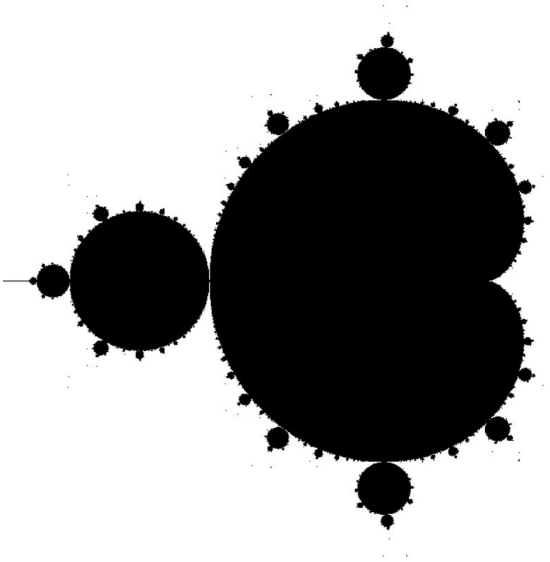
Si esta idea la traspasamos a pseudocódigo siendo *a*, *b* y *c* los puntos de los vértices del triángulo con sus coordenadas *x* e *y*, y *n* el nivel de profundidad del fractal, sería tal que así:

```
dibujar_sierpinski (a, b, c, n)
	si n = 0
		dibujar triangulo (a, b, c) // Dibuja un triángulo con esos vértices.
	finsi
	sino
		ab ← punto medio de a y b.
		bc ← punto medio de b y c.
		ca ← punto medio de c y a.
		dibujar_sierpinski (a, ab, ca, n - 1)
		dibujar_sierpinski (ab, b, bc, n - 1)
		dibujar_sierpinski (ca, bc, c, n - 1)
	finsino
```

Este algoritmo se puede utilizar para diferentes figuras geométricas modificando levemente la implementación pudiendo así crear fractales cuadrados o pentagonales entre infinidad de figuras.

Mandelbrot

Posiblemente, el fractal de Mandelbrot, y sus variantes, ha sido el más estudiado de todos. El desarrollo de este fractal se basa en números complejos y un proceso iterativo. El matemático francés Benoît Mandelbrot partió de la función compleja $Z_n = Z^2_{n-1} + C$ siendo Z y C números complejos. Lo que representa el fractal no es la función mencionada, sino un conjunto de valores de dicha función, concretamente los valores que tienen un módulo (radio) menor o igual a 2 conocidos como conjunto de Mandelbrot ($|Z_n| \leq 2$), únicamente serán representados estos valores.



C representa cada valor del plano, será necesario comprobar si cada valor, es decir, cada punto, pertenece al conjunto de Mandelbrot. Para ello, se realiza un proceso iterativo el cual cuantas más veces se repita, más seguros podremos estar de que dicho punto pertenece al conjunto. Inicialmente, $Z_0 = 0$, la iteración comienza con Z_1 ya que para calcular Z_n siempre necesitaremos saber Z_{n-1} según la función compleja.

Lo ideal, sería repetir la iteración $Z_n = Z^2_{n-1} + C$, siendo *n* el número de iteraciones, para cada C un número infinito de veces lo cual, como hemos comentado con anterioridad, es imposible computacionalmente. No obstante, dependiendo de la resolución del dispositivo en el que se muestra el fractal, a partir de un número de iteraciones se obtiene un fractal lo suficientemente definido para no apreciarse irregularidades. A partir de 100 iteraciones se puede observar la forma de este sin problema pero con bastante *ruido* apreciable. Partiendo de aquí, puedes utilizar un número indefinido de iteraciones.

Además de limitar el número de iteraciones, también es necesario limitar el plano el cual se va a representar ya que no podemos representar infinitos puntos. Comúnmente se escoge el vertice superior (-2+2i), (2+2i) y el inferior (-2-2i) (2-2i) como rango.

A continuación, para representar computacionalmente el fractal de Mandelbrot seguiremos el siguiente algoritmo:

mi_fractal() inicializa variables y constantes necesarias para Mandelbrot y se llama a la función mandelbrot para cada pixel.

```
mi_fractal()
      x_ui, x_uf, y_ui, y_uf // Rango de representación (plano).
      x_pi, x_pf, y_pi, y_pf // Resolución del dispositivo donde se representa
                             //  el fractal.
      num_max // Número máximo de iteraciones.
      radio // Radio de representación (comúnmente 2).
      C // Punto a representar (número complejo en cartesianas x e y).
      booleano dibujar //Pixel pertenece o no al conjunto

      altura_funcion ← |y_uf - y_ui|
      anchura_funcion ← |x_uf - x_ui|
      altura_dispositivo ← |y_pf - y_pi + 1|
      anchura_dispositivo ← |x_pf - x_pi + 1|

      //Avance de la función por cada pixel del dispositivo en el eje X.
      dx_p ← anchura_funcion / anchura_dispositivo
      //Avance de la función por cada pixel del dispositivo en el eje Y.
      dy_p ← altura_funcion / altura_dispositivo
      //Bucles que recorren cada pixel del dispositivo
      Para x_p ← x_pi hasta x_pf con incremento en 1
            Para y_p ← y_pi hasta y_pf con incremento en 1
                  C.x ← x_ui + x_p * dx_p
                  C.y ← y_ui + y_p * dy_p
                  dibujar ← mandelbrot(C, num_max, radio)
                  PintarPixel(xp, yp)
            fin_para
      fin_para
```

Seguido, el núcleo para representar Mandelbrot, dentro de la función mandelbrot se usan algunas funciones adicionales como *modulo(complejo)* que devuelve el módulo de un número complejo y *z_siguiente(complejo, complejo)* que realiza la función compleja de Mandelbrot y y devuelve Z_{n+1}.

```
booleano mandelbrot(C, max_iter, radio)
	b_mandelbrot ← verdadero
	z ← C
	z_sig ← z_siguiente(z, c)
	i ← 1 // Iterador
	mientras b_mandelbrot = verdadero e i < max_iter
		si modulo(z_sig) < radio
			z ← z_dig
			z_sig = z_siguiente(z, c)
			i ← i + 1
		sino
			b_mandelbrot = falso
		fin_si
	devolver b_mandelbrot
```

Conclusión

Hemos tratado tres tipos diferentes de fractales, pero sin duda alguna, los fractales son un ámbito de las matemáticas y de la informática gráfica en nuestro caso, que tiene una profundidad de estudio enorme, tal es así que matemáticos a lo largo de los años han dedicado parte de su carrera en ellos. Porque los fractales se encuentran en todos lados y su estudio es sinónimo del estudio de la naturaleza. Cabe decir por lo tanto, que hemos dando una pincelada al mundo fractal, siendo este un lienzo gigantesco.

Me gustaría incluir además en esta conclusión, que personalmente, me siento contento de haber elegido el tema de fractales ya que me ha permitido ver una parte de la informática gráfica que no conocía en *profundidad* y este trabajo, posiblemente, me de pie para seguir investigando el tema porque me parece muy curioso y, por encima de todo, muy bonito.

Como documento auxiliar al trabajo, adjunto a continuación cuatro enlaces los cuales contienen código fuente de programas que he realizado mediante OpenGL en Microsoft Visual Studio. Cada programa te permite representar gráficamente los siguientes fractales:
Curva de Koch: https://figdrh.blogspot.com/2018/12/curva-de-koch-opengl.html
Árbol de Koch: https://figdrh.blogspot.com/2018/12/arbol-de-koch-opengl.html
Triángulo de Sierpinski: https://figdrh.blogspot.com/2018/12/triangulo-de-sierpinski-opengl.html
Fractal de Mandelbrot: https://figdrh.blogspot.com/2018/12/fractal-de-mandelbrot.html

Bibliografía

Steven R. Davidson (Junio de 2004). Gráficos con clase. Lugar de publicación: conclase.net. Disponible en: http://graficos.conclase.net/curso/?cap=003#inicio .

Yango (18 de Marzo de 2005). Fractales: Una nueva geometría. Lugar de publicación: Universidad de Granada. Disponible en: https://www.ugr.es/~pgomez/docencia/tc/documentos/Fractales-una-nueva-geometria.pdf.

Schmidtke, M. (2018). Copo de nieve de Koch. [Imagen] Disponible en: http://enciclopedia.us.es/index.php/Archivo:Copo_de_nieve_de_Koch.png [Aceso el 2 Dec. 2018].

Schmidtke, M. (2018). Copo de nieve de Koch 2. [Imagen] Disponible en: http://enciclopedia.us.es/index.php/Archivo:Copo_de_nieve_de_Koch_2.png [Aceso el 2 Dec. 2018].