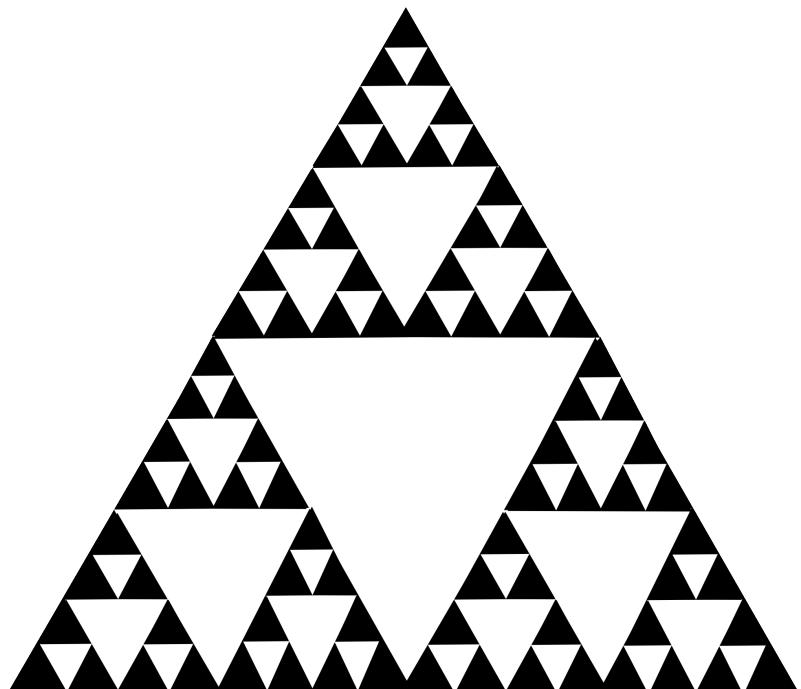




Escuela Técnica Superior de
Ingeniería Informática

SISTEMAS DE FUNCIONES ITERADAS



Matemáticas para la Computación

Autores:

Beatriz Galiana Carballido, DNI: 29569563G
Fernando Sánchez Paredes, DNI: 29501848R

Sevilla, enero de 2021

Índice general

Lista de figuras	1
1. Introducción	3
2. Estudio teórico	4
2.1. Fractales	4
2.2. Aplicaciones afines	6
2.2.1. Elementales	6
2.2.2. Contractivas	7
2.3. Sistemas de funciones iteradas	7
3. Ejemplos clásicos	9
3.1. Triángulo de Sierpinski	9
3.2. Curva de Koch	9
3.3. Helecho de Barnsley (The Barnsley Fern)	10
4. Implementación del algoritmo	12
4.1. Planteamiento del algoritmo	12
4.2. Primera parte: métodos auxiliares	12
4.2.1. Funciones Auxiliares	13
4.3. Segunda parte: calculando los puntos del fractal	15
4.3.1. Código Principal	15
5. Aplicaciones	18
6. Conclusiones	21
A. Implementación de la aplicación web	22
A.1. Aplicación web	22
A.1.1. Primera versión: ventajas y problemas	22
A.1.2. Versión definitiva: detalles de implementación y características	22
A.1.3. Código	23
B. Fractales generados mediante IFS	25
Bibliografía	25

Índice de figuras

1.0.1 Sistema de funciones iteradas	3
1.0.2 Libro sobre fractales de Barnsley	3
2.1.1 Romanesco. Un fractal natural	4
2.1.2 Fractal de Newton	5
2.1.3 Conjunto de Mandelbrot, un fractal no autosimilar	5
3.1.1 Triángulo de Sierpinski	9
3.2.1 Curva de Koch	10
3.3.1 Helecho de Barnsley	11
5.0.1 Montaña artificial creada mediante fractales	18
5.0.2 Amosita. Fósil sólido	19
5.0.3 Música fractal	19
5.0.4 Modelo forest fire con propagación	20

Capítulo 1

Introducción

En este documentos tratamos una construcción matemática empleada para la generación de fractales autosimilares, es decir, es un trabajo acerca de los sistemas de funciones iteradas.

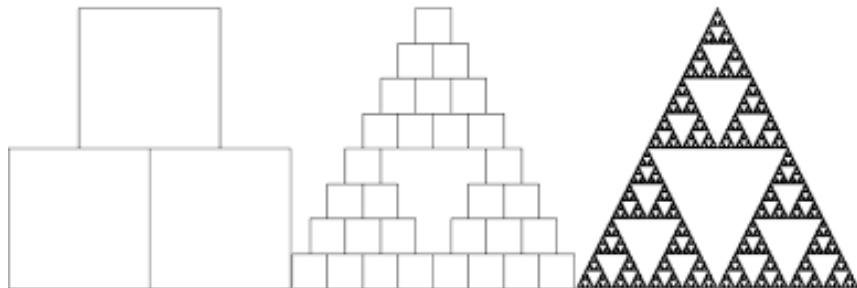


Figura 1.0.1: Sistema de funciones iteradas

Este método fue creado por Michael Fielding Barnsley, matemático, investigador y emprendedor británico que ha trabajado en la compresión de los fractales. Ha publicado múltiples artículos y su famoso libro sobre fractales *Fractals Everywhere*.

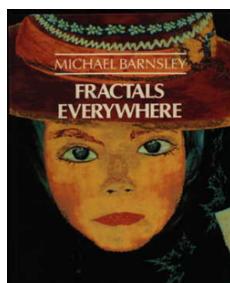


Figura 1.0.2: Libro sobre fractales de Barnsley

Vamos a definir primero algunos conceptos necesarios como son los fractales y aplicaciones afines. Explicaremos en que consiste esta construcción matemática y pondremos ejemplos de algunos fractales popularmente conocidos.

Tras realizar este estudio teórico procederemos a explicar la implementación del algoritmo y, por último, explicaremos algunas de las aplicaciones que tiene en la práctica.

Capítulo 2

Estudio teórico

En esta sección vamos a explicar en que consiste un sistema de funciones iteradas con probabilidad, para ello es necesario explicar antes qué son los fractales y las aplicaciones afines pues resulta esencial conocerlos para poder entender cómo funciona este método para generar fractales.

2.1. Fractales

Un fractal es un objeto geométrico cuya estructura básica, fragmentada o aparentemente irregular, se repite a diferentes escalas.

La palabra “fractal” proviene del latín *fractus*, que significa “fragmentado”. El término se le debe a Benoît Mandelbrot en 1977, donde aparecía en su libro *The Fractal Geometry of Nature*. Al estudio de los objetos fractales se le conoce, generalmente, como geometría fractal.

Los fractales no son siempre objetos geométricos creados por el hombre mediante ecuaciones, sino que también podemos encontrar muchos presentes en la naturaleza incluso de manera cotidiana, algunos ejemplos son hojas, corales, variedades de brócoli y un largo etcétera.



Figura 2.1.1: Romanesco. Un fractal natural

Tenemos también fractales creados mediante ecuaciones como por ejemplo el Fractal de Newton o el Conjunto de Mandelbrot.

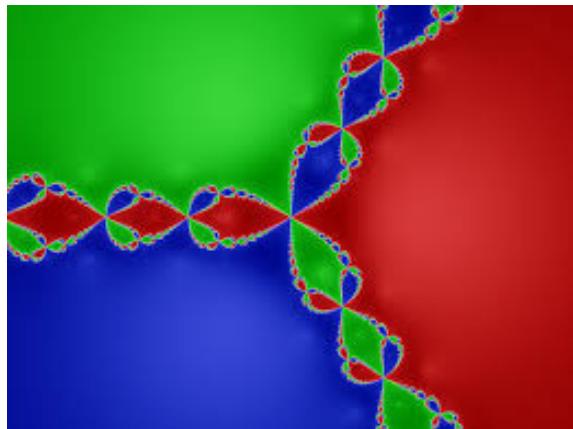


Figura 2.1.2: Fractal de Newton

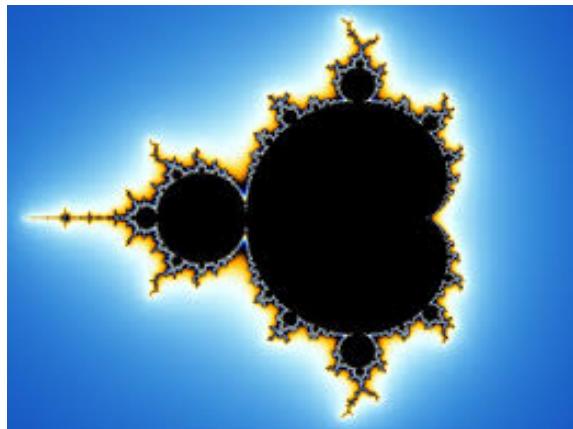


Figura 2.1.3: Conjunto de Mandelbrot, un fractal no autosimilar

Podemos enunciar algunas propiedades sobre los fractales:

- Son **autosimilares**, es decir, si miramos a mayor escala una parte de nuestro fractal vemos que no es distingible de la imagen original, es cierto que no todos son autosimilares, por ejemplo, el conjunto de Mandelbrot, pero en este trabajo nos vamos a centrar en aquellos que sí son autosemejantes tales como la curva de Koch o el triángulo de Sierpinskí.
- Su **dimensión fractal debe ser mayor que su dimensión topológica**.

Esta propiedad nos obliga a dar una definición de lo que entendemos por **dimensión** lo cual no es una tarea sencilla. Podemos definir la dimensión de un objeto geométrico como su capacidad de abarcar 'espacio', o bien como el número de coordenadas necesarias para especificar la posición de cada uno de sus elementos.

Mediante esta definición podemos decir que un punto tendría dimensión 0, una curva dimensión 1, una superficie dimensión 2, un sólido dimensión 3, etc.

Existen varias definiciones de dimensión, hemos visto lo que sería la dimensión topológica (la cual es popularmente conocida), sin embargo, la dimensión fractal puede dividirse en dos secciones:

- **Dimensión de autosemejanza.** Dimensión empleada para aquellos fractales autosimilares.
- **Recuento de cajas.** Es una definición más general que engloba a la dimensión de autosemejanza y se puede aplicar a cualquier fractal.

Al ser este un trabajo sobre fractales autosimilares vamos a centrarnos únicamente en la dimensión de autosemejanza la cual la podemos definir de la siguiente manera:

Sea un conjunto autosemejante compuesto por N copias reducidas en un factor de escala e , la dimensión d de autosemejanza del conjunto se define como el exponente que verifica

$$N = e^d$$

O lo que es equivalente

$$d = \frac{\log(N)}{\log(e)}$$

Por ejemplo, la dimensión topológica de la curva de Koch está compuesta por 4 copias de sí mismo y un factor

$$e = \frac{1}{\epsilon}$$

donde

$$\epsilon$$

conforma la tercera parte de la curva total, luego su dimensión topológica es

$$\frac{\log(4)}{\log(3)} = 1,2618\dots$$

La dimensión topológica de la curva de Koch es 1, luego se cumple la condición que indica que la dimensión topológica debe ser menor que su dimensión fractal.

- No es diferenciable en ninguno de sus puntos, es decir, no es derivable.

2.2. Aplicaciones afines

El sistema de funciones iteradas se realiza por medio de un conjunto de aplicaciones afines contractivas, vamos a recordar que eran las aplicaciones afines y las transformaciones elementales del plano.

En geometría, una transformación afín o aplicación afín (también llamada afinidad) entre dos espacios afines consiste en una transformación lineal seguida de una traslación.

2.2.1. Elementales

Hay una serie de transformaciones elementales que pueden realizarse en el plano. Estas son:

- Traslación. Incrementamos la coordenada x en un valor a y la coordenada y en un valor b .

$$f \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix} \quad (2.2.1)$$

- Rotación. Determinamos la rotación de un ángulo α alrededor del origen mediante la siguiente transformación:

$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

(2.2.2)

- Simetría respecto del eje x.

$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad (2.2.3)$$

- Simetría respecto del eje y.

$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad (2.2.4)$$

- Homotecia de razón r centrada en el origen. Mediante esta transformación elemental podemos variar a razón r las coordenadas x e y .

$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} r & 0 \\ 0 & r \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad (2.2.5)$$

2.2.2. Contractivas

Sabemos ya que una aplicación afín puede representarse de la siguiente manera

$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \quad (2.2.6)$$

Donde hemos visto los casos particulares de la transformación afín para la traslación, rotación, simetría con respecto a los ejes x e y y homotecias.

Las aplicaciones afines contractivas son, entonces, todas aquellas transformaciones que pueden obtenerse a partir de transformaciones elementales y que todos sus coeficientes sean menores que la unidad.

2.3. Sistemas de funciones iteradas

Los sistemas de funciones iteradas, también conocidas como el método de Barnsley son conjuntos de n transformaciones afines contractivas. Podemos encontrar dos tipos de algoritmos:

- Determinista. En este caso tomamos un conjunto de puntos en el plano y le aplicamos cada una de las n transformaciones afines de nuestro sistema, obtendremos n puntos, a estos n puntos le volvemos a aplicar las n transformaciones afines.
Vamos a realizar tantas iteraciones como sean necesarias hasta que obtengamos una figura que resultará ser el atractor del sistema.
- Aleatorio. Es un algoritmo similar pero en lugar de aplicar las funciones a un conjunto de puntos se aplica tan solo a uno que vamos dibujando, lo que hacemos en este caso es aplicar a cada una de las transformaciones del sistema una probabilidad, teniendo en cuenta, lógicamente, que la suma de todas las probabilidades de las transformaciones del sistema sean 1 uno.

Luego, en cada una de las iteraciones de nuestro algoritmo escogemos una de las transformaciones en función de la probabilidad asignada y dibujamos dicho punto. Iteramos tantas veces como sea necesaria para obtener el atractor del sistema.

En ambos algoritmos acabamos obteniendo el atractor del sistema que será un fractal autosimilar ya que este fractal está formado por partes más pequeñas de sí mismo.

En este trabajo nos vamos a centrar en el algoritmo aleatorio.

Capítulo 3

Ejemplos clásicos

3.1. Triángulo de Sierpinski

El triángulo de Sierpiński es un fractal que se puede construir a partir de cualquier triángulo.

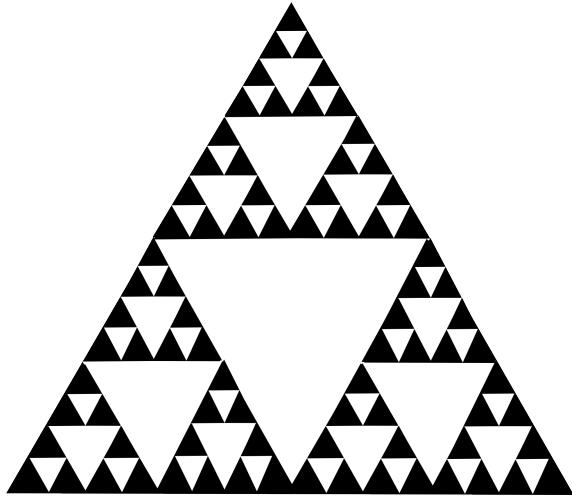


Figura 3.1.1: Triángulo de Sierpinski

Para formar este fractal mediante los sistemas de funciones iteradas con probabilidad necesitaremos 3 funciones, todas ellas equiprobables.

$$f_1 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \left(\frac{x}{2}, \frac{y}{2} \right) \quad (3.1.1)$$

$$f_2 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{1}{2} \\ 0 \end{pmatrix} = \left(\frac{x+1}{2}, \frac{y}{2} \right) \quad (3.1.2)$$

$$f_3 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \cos 60 \\ \sin 60 \end{pmatrix} = \left(\frac{x+\cos 60}{2}, \frac{y+\sin 60}{2} \right) \quad (3.1.3)$$

Todas ellas con $\frac{1}{3}$ de probabilidad.

3.2. Curva de Koch

Este fractal es una curva cerrada continua pero no diferenciable en ningún punto descrita por el matemático sueco Helge von Koch. Para construir este fractal mediante sistemas de funcio-

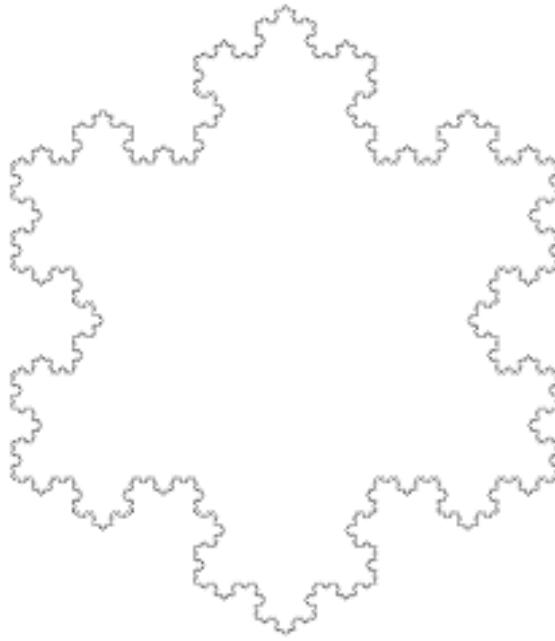


Figura 3.2.1: Curva de Koch

nes iteradas necesitamos 4 funciones todas ellas equiprobables.

$$f_1 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (3.2.1)$$

$$f_2 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{6} & 0,2\widehat{8} \\ 0,2\widehat{8} & \frac{1}{6} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{1}{3} \\ 0 \end{pmatrix} \quad (3.2.2)$$

$$f_3 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{6} & 0,2\widehat{8} \\ 0,2\widehat{8} & \frac{1}{6} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{2}{3} \\ 0 \end{pmatrix} \quad (3.2.3)$$

$$f_4 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{2}{3} \\ 0 \end{pmatrix} \quad (3.2.4)$$

Todas ellas con $\frac{1}{4}$ de probabilidad.

3.3. Helecho de Barnsley (The Barnsley Fern)

El helecho de Barnsley es un fractal que lleva el nombre del matemático británico Michael Barnsley, quien lo describió por primera vez en su libro *Fractals Everywhere*.

Para construir este fractal mediante sistemas de funciones iteradas necesitamos 4 funciones.

La primera función se da con probabilidad 0,01

$$f_1 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & \frac{4}{25} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (3.3.1)$$

La segunda función se da con probabilidad 0,85

$$f_2 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{17}{20} & \frac{1}{25} \\ -\frac{1}{25} & \frac{17}{20} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{8}{5} \end{pmatrix} \quad (3.3.2)$$



Figura 3.3.1: Helecho de Barnsley

La tercera función se da con probabilidad 0,07

$$f_3 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{5} & -\frac{13}{50} \\ \frac{23}{100} & \frac{11}{50} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{8}{5} \end{pmatrix} \quad (3.3.3)$$

La cuarta función se da con probabilidad 0,07

$$f_4 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -\frac{3}{20} & \frac{7}{25} \\ \frac{13}{50} & \frac{6}{25} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{11}{25} \end{pmatrix} \quad (3.3.4)$$

Capítulo 4

Implementación del algoritmo

4.1. Planteamiento del algoritmo

Para programar el algoritmo hemos decidido usar el lenguaje JavaScript (a partir de ahora JS). El motivo principal es que el generador de fractales va a integrarse dentro de una sencilla aplicación web, y JS nos permite realizar esta integración de manera cómoda y eficiente.

Además, utilizaremos el elemento Canvas de HTML5, que permite la renderización interpretada dinámica de gráficos 2D y mapas de bits, así como animaciones con estos gráficos, a través del *scripting*.

Es extremadamente interesante el desarrollo utilizando HTML5 Canvas y JavaScript pero dado que los detalles de implementación del programa *per se* se encuentran fuera de los objetivos del trabajo y de la asignatura, vamos a centrarnos exclusivamente en la implementación de los sistemas de funciones iterados.

En caso de que resulte de interés ampliar un poco más sobre detalles técnicos del programa, dejaremos un apartado en la bibliografía con algunos recursos útiles para profundizar un poco más en las herramientas que han sido utilizadas.

Como se ha mencionado anteriormente, hemos decidido no entrar en detalle en la implementación de aspectos meramente relacionados con la sintaxis y las interfaces de programación de las herramientas utilizadas. Sin embargo, ya que queremos que se comprenda la funcionalidad el código en su totalidad, vamos a ubicar los métodos y mencionar ligeramente qué hacen.

4.2. Primera parte: métodos auxiliares

Podemos ubicar cinco funciones auxiliares en el código:

- `readIfs()`
- `plot()`
- `findBounds()`
- `newPicture()`
- `DrawSomePoints()`

4.2.1. Funciones Auxiliares

La siguiente función se encarga de leer las funciones pertenecientes a un fractal y añadirlas a una variable global *fractal* para usarse posteriormente para el cálculo de puntos.

```
1 function readIfs() {
2     values = document.getElementById("values").value.trim().split(/\s+/);
3     // Transforma tipo texto en tipo NaN
4     for (var i = 0; i < values.length; i++) values[i]=Number(values[i]);
5     if (!(values.length > 0 && values.length % 7 == 0)) {
6         alert("Invalid IFS");
7     } else {
8         fractal = [];
9         for (var i = 0; i < values.length; i += 7) {
10             fractal.push(values.slice(i, i + 7));
11         }
12     }
13 }
```

La siguiente función se encarga de dibujar los pixeles correspondientes a los puntos de la iteración en la que se encuentre el algoritmo, para ello ajustamos los puntos *x* e *y* (nuevos puntos de la iteración ya calculados) y lo ajustamos a la pantalla multiplicándolo por un factor que se explicará en la siguiente función, una vez tenga el tamaño deseado le sumamos un desplazamiento para situarlo en la parte que se desee.

La variable *c* contiene la función del fractal escogida, el vector *colors* contiene una serie de colores **RGB**, una vez tengamos los colores y las coordenadas asignamos estos valores a los pixeles correspondientes.

```
1 function plot(c) {
2     var px = scaleX * x + offsetX,
3         py = scaleY * y + offsetY,
4         base = (Math.floor(py) * width + Math.floor(px))*4;
5
6     if(!c) elige_color_probabilidad=colors[0]; else
7         elige_color_probabilidad = colors[c % 8];
8
9     pixels[base] = elige_color_probabilidad[0];
10    pixels[base+1] = elige_color_probabilidad[1];
11    pixels[base+2] = elige_color_probabilidad[2];
12    pixels[base+3] = 255;
13 }
```

La siguiente función se encarga de obtener los valores necesarios para que el fractal se pueda ver con un tamaño adecuado y en una posición visible, estos valores son los que se usan en la función *plot(c)*, esta función carece de importancia en este contexto donde tratamos de explicar los sistemas de funciones iteradas.

```
1 function findBounds() {
2     var left = 0, right = 0, top = 0, bottom = 0;
3     x = 0;
4     y = 0;
5     for (var i = 0; i < 10000; i += 1) {
6         next();
```

```

7         // Reasignar limites del canvas
8         if (x < left) left = x;
9         if (x > right) right = x;
10        if (y < bottom) bottom = y;
11        if (y > top) top = y;
12    }
13    // Ajusta a 1:1
14    if (top - bottom > right - left) {
15        left = (left + right - top + bottom) / 2;
16        right = left + top - bottom;
17    } else {
18        bottom = (bottom + top - right + left) / 2;
19        top = bottom + right - left;
20    }
21    // Cuadra elementos dentro del canvas
22    scaleX = width / (right - left);
23    scaleY = height / (bottom - top);
24    offsetX = (width * left) / (left - right);
25    offsetY = (height * top) / (top - bottom);
26}

```

La siguiente función se encarga de inicializar el *canvas* correspondiente, leemos los valores del fractal seleccionado y empezamos a dibujar los puntos en el plano.

```

1 function newPicture(){
2     // Clear pixel data - Colorea el fondo
3     for (var i = 0; i < width * height * 4; i += 4) {
4         pixels[i] = 0; //R
5         pixels[i+1] = 0; //G
6         pixels[i+2] = 0; //B
7         pixels[i+3] = 0; //Transparencia
8     }
9     ctx.putImageData(imageData, 0, 0);
10
11    readIffs();
12
13    findBounds();
14}

```

La siguiente función se encarga de dibujar los puntos en el plano, si se ha seleccionado un fractal nuevo inicializa todo de nuevo, sino, calcula los puntos con la función *next* la cual es la fundamental en este código y una vez hayamos calculado los puntos los almacenamos en el vector *pixels* y actualizamos el canvas con dicho vector. Esta función se llama recursivamente para ir mostrando los puntos calculados.

```

1 function drawSomePoints() {
2     if (needRefresh) {
3         newPicture();
4         needRefresh = false;
5     }
6
7     for (var i = 0; i < 500; i += 1) {
8         var c = next();
9         plot(c);
10    }
11
12    // Vuelca el array de pixeles en el canvas
13    ctx.putImageData(imageData, 0, 0);

```

```

14         setTimeout(drawSomePoints, 500);
15     }
16 }
```

4.3. Segunda parte: calculando los puntos del fractal

Vamos a ver los detalles del código encargado de realizar las transformaciones que se han visto a lo largo de esta memoria para poder calcular los puntos del trazado del fractal resultante.

Esta parte del código se puede dividir de la siguiente manera:

- Declaración de variables globales
- Creación de desplegables y cuadro de texto para parámetros del IFS
- Botón Cargar
- Selección de fractal del desplegable
- Temporizador
- Función `next()` - aplica el sistema de funciones iteradas

4.3.1. Código Principal

Aquí podemos ver las variables globales declaradas que han sido usada previamente:

- *canvas*. El recuadro donde aparecerá el fractal deseado.
- *width*. Anchura del canvas
- *height*. Altura del canvas
- *ctx*. Indicamos que el canvas será en dos dimensiones
- *imageData*. Creamos la imagen mediante la anchura y altura del canvas
- *fractal*. Pila que almacenará el fractal deseado.
- *x*. Coordenada *x* del fractal deseado.
- *y*. Coordenada *y* del fractal deseado.
- *scaleX*. Factor por el que se multiplicará la coordenada *x*.
- *scaleY*. Factor por el que se multiplicará la coordenada *y*.
- *offsetX*. Factor de desplazamiento que se aplicará a la coordenada *x*.
- *offsetY*. Factor de desplazamiento que se aplicará a la coordenada *y*.
- *pixels*. Contiene los pixeles del canvas creado.
- *colores*. Lista con una serie de colores para dibujar la función escogida con probabilidad *p* del fractal deseado.
- *needRefresh*. Variable que indicará si debemos refrescar el canvas con un fractal nuevo.
- *select*. Referencia al botón que contiene el desplegable de todos los fractales.
- *values*. Referencia al elemento HTML que contiene los valores de la función del fractal seleccionado.
- *refresh*. Referencia al botón *refresh* para indicar que hemos modificado algún parametro de una función.
- *data*. Referencia al elemento HTML que contiene las funciones de todos los fractales.

Tras declarar las variables a emplear guardaremos en *node* todos los fractales disponibles. Cada vez que se pulse en el botón de refrescar pondremos la variable *refresh* a verdadero y se inicializará de nuevo el código.

```

1 var canvas = document.getElementById("ifs"),
2 width = canvas.width,
3 height = canvas.height,
```

```

4   ctx = canvas.getContext("2d"),
5   imageData = ctx.createImageData(width, height),
6   fractal = [],
7   x = 0,
8   y = 0,
9   scaleX = 1,
10  scaleY = 1,
11  offsetX = 0,
12  offsetY = 0,
13  pixels = imageData.data,
14  colors = [[249,7,255],[2,216, 99],[249, 56, 0],[0, 0, 153],[153, 102,
15   51],
16   [255, 0, 102],[153, 0, 153],[0,0,0]];
17  needRefresh = false;
18
19 select = document.getElementById("ifs-select"), // Desplegable fractales
20 values = document.getElementById("values"), // Cuadro de texto con valores
21   del IFS
22 refresh = document.getElementById("refresh"), // boton cargar
23 data = document.getElementById("data").childNodes; // divs conteniendo
24   diferentes fractales
25
26 // Create el desplegable para la seleccion de fractales
27 for (var i = 0, n = data.length; i < n; i++) {
28   node = data[i];
29   if (node.nodeType === 1) {
30     select.options.add(new Option(node.id, node.innerHTML));
31   }
32 }
33
34 // Pulsar boton cargar
35 refresh.onclick = function () {
36   needRefresh = true;
37 }
38
39 // Seleccionar fractal de desplegable
40 select.onchange = function () {
41   document.getElementById("values").value = select.options[select.
42     selectedIndex].value;
43   needRefresh = true;
44 }
45
46 // Dibuja un conjunto de puntos cada 500ms.
47 setTimeout(drawSomePoints, 500);

```

Esta es la función más importante de todo el código, en esta función es donde aplicamos el sistema de funciones iteradas calculando las coordeandas x e y de la siguiente iteración.

Primero escogemos un número r aleatorio entre 0 y 1 que indicará la probabilidad, iteramos las funciones del fractal seleccionado y si la probabilidad de dicha función es mayor o igual a r entonces será esa función la que se utilizará, sino, acumulamos la probabilidad y en la siguiente iteración comprobamos la siguiente función.

Una vez tengamos la función a emplear calculamos los nuevos valores de x e y tal y como se ha explicado en el estudio teórico, mediante las aplicaciones afines contractivas. La función devuelve la posición de función utilizada para, en función de este valor, dibujar dicho punto de

un color u otro.

```
1 function next() {
2     var r = Math.random(), probabilityThreshold = 0.0;
3     for (var i = 0; i < fractal.length; i += 1) {
4         var t = fractal[i];
5         if (r <= (probabilityThreshold += t[6])) {
6             var oldx = x;
7             // Nuevos valores de X e Y
8             x = t[0] * x + t[1] * y + t[4];
9             y = t[2] * oldx + t[3] * y + t[5];
10            return i; // Rompemos el bucle y almacenamos indice para
11                seleccionar color en plot
12        }
13    }
```

Capítulo 5

Aplicaciones

Se suele decir que los fractales son mágicos e incluso místicos, lo que se puede asegurar es que son muy especiales. Más allá de descubrir una nueva dimensión geométrica de gran belleza en las formas y los procesos de la naturaleza, Benoit Mandelbrot sabía que los fractales serían de gran utilidad para muchos campos de la ciencia.

Está claro que son uno de los descubrimientos matemáticos más populares del siglo XX. Están en la naturaleza, y tienen aplicaciones científicas y tecnológicas insospechadas. En la medicina, en geografía, en bioquímica, arqueología, etc. Es decir, las aplicaciones de la geometría fractal en nuestra vida son muchísimas, desde el cambio climático al cáncer.

Un ejemplo de uso de los fractales lo encontramos cuando la productora 'LucasFilm' comenzó a hacer animaciones por ordenador a finales de los años 70, no sabían cómo simular la forma y textura de las montañas hasta que Loren Carpenter, de la división de animación digital, descubrió que podía representar esa textura, repitiendo triángulos de manera fractal, para luego cubrirlas con un color afín a la montaña que se quería representar.

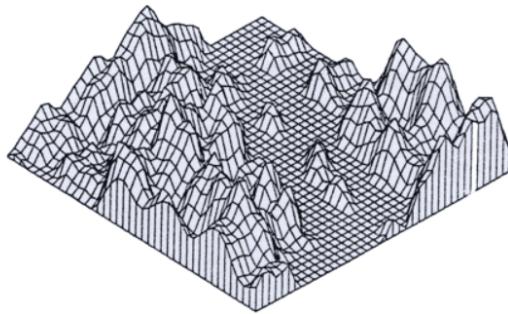


Figura 5.0.1: Montaña artificial creada mediante fractales

Podríamos decir que Pixar y Dreamworks le deben su existencia a los fractales... O al menos les facilitó el desarrollo.

También son de utilidad en arqueología, donde los patrones que aparecen al analizar los hallazgos son justamente fractales. En sismología también se ha descubierto que valores como la

frecuencia de sismos y la intensidad de las réplicas obedecen a patrones fractales.



Figura 5.0.2: Amosita. Fósil sólido

Los fractales no tienen utilidad solo en la ciencia, sino en prácticamente cualquier aspecto de nuestra vida. Se han desarrollado sistemas para crear música usando fractales, como Aural Fractals.

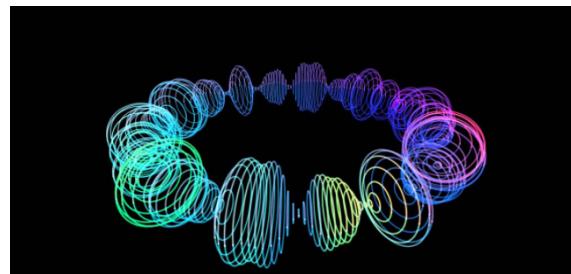


Figura 5.0.3: Música fractal

Si se identifica una estructura esencial en la naturaleza y se aplican los principios de la geometría fractal para descomponerla, se pueden hacer predicciones sobre cómo se comportará dicha estructura en el futuro gracias a que los fractales cumplen con el principio matemático de autosimilitud.

Sería posible explicar muchos fenómenos complejos de la naturaleza si la interacción entre los átomos se producen en función del principio de autosimilitud de la geometría fractal. En términos matemáticos esto significa que los fractales no pueden predecir como serán los eventos en sistemas caóticos pero si pueden decírnos que van a ocurrir (que no es poco).

Durante muchísimos años se pensaba que el corazón humano latía de forma regular y lineal, a día de hoy sabemos que esto no es así, existe un patrón fractal determinado para los latidos de un corazón sano. Investigadores de la Harvard Medical School han demostrado que las alteraciones en la escala fractal pueden ser la base de alteraciones fisiopatológicas, incluido el síndrome de muerte súbita cardíaca.

Podría ocurrir que la geometría fractal una especie de principio de diseño biológico que nos programa para funcionar de la manera más eficaz posible, si esto fuese así, una de sus aplicaciones más prometedoras es detectar cuándo va a fallar esa programación óptima, convirtiéndose en una potente herramienta de diagnóstico, mucho más que cualquier máquina y, además, de aplicación universal. Así, en el tratamiento del cáncer, por ejemplo, la geometría fractal es útil para desvelar la arquitectura patológica de los tumores y sus mecanismos de crecimiento.

Los fractales también tienen muchas aplicaciones en ecología como, por ejemplo, estudiar cómo se extiende un fuego en un incendio forestal (mediante autómatas celulares).

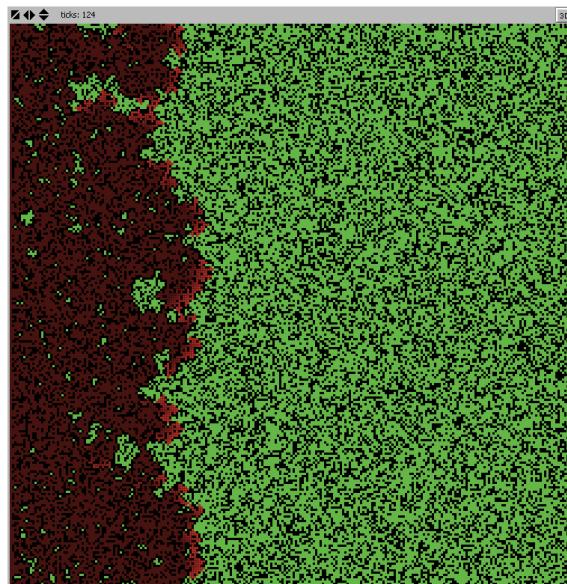


Figura 5.0.4: Modelo forest fire con propagación

Capítulo 6

Conclusiones

Durante la realización de este trabajo hemos aprendido más en profundidad qué son los fractales, que características posee y cuáles son algunas de sus aplicaciones. Hemos visto que los fractales son objetos aún muy desconocidos llegando a parecer 'mágicos' por la infinidad de aplicaciones que tienen en nuestra vida y que aún desconocemos.

Los fractales están presente en cualquier campo de estudio y en la naturaleza en elementos tales como algunas verduras, animales e incluso los órganos de nuestro cuerpo contienen patrones fractales.

No solo se ha aprendido más acerca de los fractales sino que también se ha llevado a cabo la implementación de un algoritmo de sistemas de funciones iteradas con probabilidad el cual nos ha servido para poder crear fractales autosimilares a partir de una serie de funciones y transformaciones elementales pudiendo observar de manera experimental fractales famosos tales como la curva de Koch, el helecho de Barnsley... y también crear nuestros propios fractales de una manera muy intuitiva y sencilla.

Apéndice A

Implementación de la aplicación web

A.1. Aplicación web

En este anexo voy a describir brevemente el proceso que se ha seguido para construir la aplicación web donde se generan los fractales.

Todo el código utilizado para el desarrollo de esta aplicación, tanto la implementación del algoritmo como de la aplicación web, se encuentra disponible en un repositorio público en GitHub al que se puede acceder clicando aquí y que también estará disponible en la bibliografía de esta memoria.

A.1.1. Primera versión: ventajas y problemas

En un primer lugar se creó una aplicación muy sencilla utilizando simplemente una página HTML complementado con un fichero CSS para estilizar un poco su apariencia y con JavaScript para poder implementar las funciones que resultan en el generador de fractales *per se*, las cuales se vieron ya en el apartado de implementación de esta memoria.

Sin embargo, los datos necesarios para cargar en la web los diferentes sistemas de funciones iterados se encontraban directamente incluidos dentro del HTML lo cual daba lugar a un monstroso fichero de nada más y nada menos 543 líneas de código.

La principal ventaja que tenía esto era la simplicidad para la ejecución de la aplicación, ya que puede ejecutarse y funcionar a la perfección solamente abriendo el fichero en cualquier navegador.

Por otro lado, existía un inconveniente a tener muy en cuenta, teníamos un fichero de aproximadamente 500 líneas de las que realmente sólo se consultaban unas 50. Por cuestiones de legibilidad, decidimos externalizar en otro fichero toda la parte del código que guardaba los datos correspondientes al trazo de fractales, lo que nos llevó de un fichero principal de 494 líneas a uno de tan sólo 44.

A.1.2. Versión definitiva: detalles de implementación y características

Tras crear un nuevo fichero fractals.html contenido sólo los datos necesarios para cargar fractales, nos encontramos con la dificultad de introducir un fichero HTML en otro.

Probadas diversas formas, la mayoría tan sumamente engorrosas que definitivamente no solucionaban nuestro problema de legibilidad, encontramos una forma de hacerlo utilizando PHP, de manera que esta inclusión se haría del lado del servidor.

```
<?php include "fractals.html" ?>
```

Con una línea de código y apenas 30 caracteres tenemos una solución elegante y legible, que nos permite importar todos los datos contenidos en fractals.html sin necesidad de cambiar nada más en nuestra aplicación.

Un pequeño tutorial: ejecutando la aplicación

El único posible inconveniente que nos trae el uso de PHP es que para poder ejecutar la aplicación necesitamos correr un servidor. Hemos decidido incluir aquí un sencillo tutorial de como correr el servidor integrado de PHP.

En caso de que el usuario probando la aplicación no quisiera instalar PHP en su máquina o tuviera dificultad para seguir este tutorial, el fichero original HTML, aunque menos legible y algo más pesado, sigue estando disponible en el repositorio y puede ejecutarse perfectamente con simplemente abrirse en cualquier navegador.

```
1 $ cd ~/directorio-contenedor-de-ficheros
2 $ php -S localhost:8000
```

Listing A.1: Ejecutando la app web en el servidor integrado de PHP.

Ahora sólo tenemos que acceder a <http://localhost:8000> en nuestro navegador y ya tenemos la aplicación web desplegada y funcionando.

A.1.3. Código

Aquí puedes echar un vistazo al código de la aplicación web.

En la línea 16 observamos que se incluye el fichero fractals.html y en la línea 36 el fichero script.js, estos dos ficheros incluirán, respectivamente, los datos y las funciones necesarias para renderizar los fractales utilizando sistemas de funciones iteradas.

El cuerpo consta principalmente de:

- En su parte izquierda: título, todos los textos, cuadros de texto y desplegables de selección que se pueden observar en la aplicación.
- Parte derecha: se incluye también el canvas donde se dibujarán los fractales.

```
1 <!doctype html>
2
3 <html>
4   <head>
5     <meta charset="utf-8">
6     <title>Sistemas de Funciones Iteradas</title>
7     <meta name="viewport" content="width=device-width, initial-scale=1.0">
8     <link rel="stylesheet" type="text/css" href="index.css">
9     <link rel="stylesheet" type="text/css" href="index.css">
10    <link href="https://fonts.googleapis.com/css?family=
11      Roboto|Lato&display=swap" rel="stylesheet">
12  </head>
13  <body>
14    <div id="data" style="display:none">
```

```

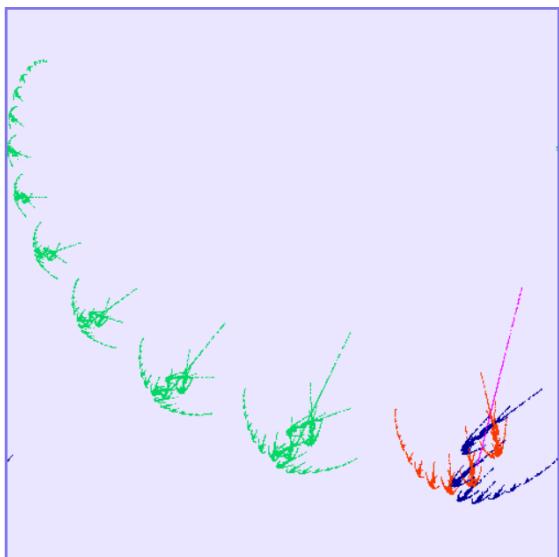
15     <?php include "fractals.html" ?>
16 </div>
17
18 <h2>Renderizador de Sistemas de Funciones Iteradas</h2>
19 <div style="display: flex;">
20
21     <div id="menu">
22         <p style="padding: 5%; font-size: medium;">Elige uno de los
23             siguientes ejemplos clasicos o modifica los parametros
24             tu mismo:
25         <select id="ifs-select"></select></p>
26         <p style="text-align: center;">
27             <textarea id="values" rows="6" cols="80"></textarea></p>
28             <button id="refresh"> Cargar !</button>
29             <p style="padding: 10%; font-size: medium;">Echale un vistazo al
30                 <a href="http://www.github.com/beagaliana/mc2021">repositorio</a>
31                 del codigo en GitHub
32         </div>
33
34         <div style="padding-left: 5%;">
35             <canvas id="ifs" height="480" width="500"
36                 style="outline: rgb(127, 119, 235) 3px solid;">
37                 Vaya ! Parece que tu navegador no soporta Canvas
38             </canvas>
39         </div>
40
41         <script src="script.js"></script>
42     </div>
43 </body>
44 </html>

```

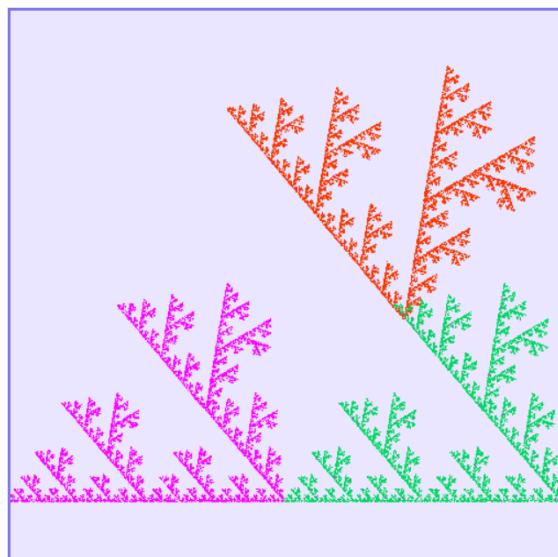
Listing A.2: Ejecutando la app web en el servidor integrado de PHP.

Apéndice B

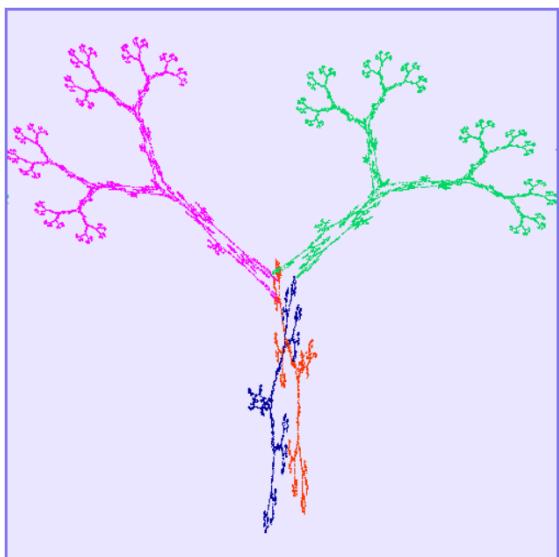
Fractales generados mediante IFS



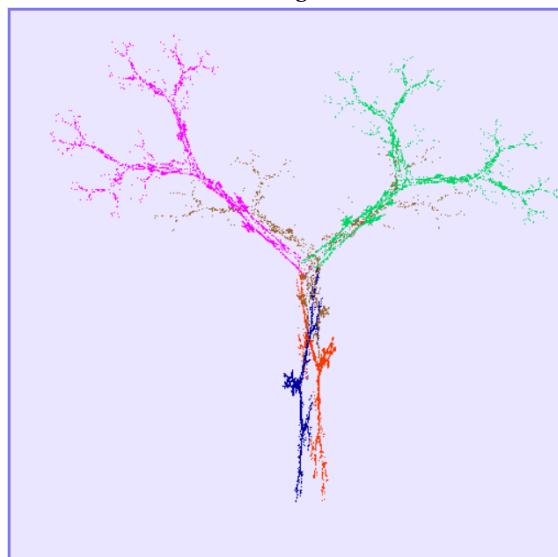
(a) Anclas



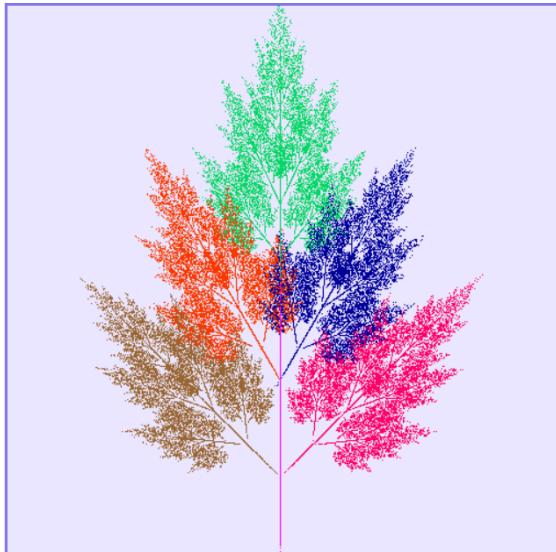
(b) Ángulo



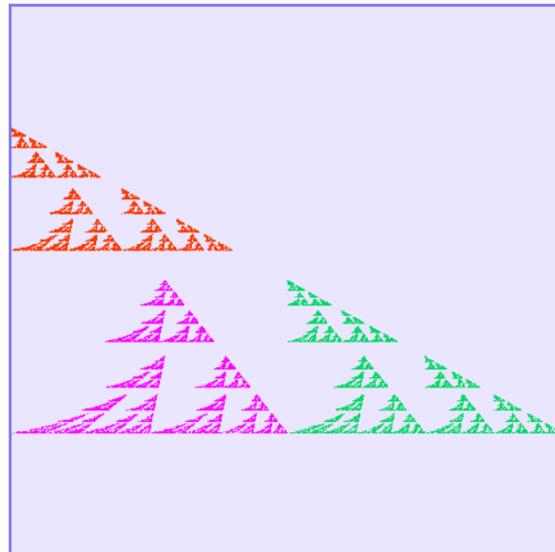
(c) Árbol 1



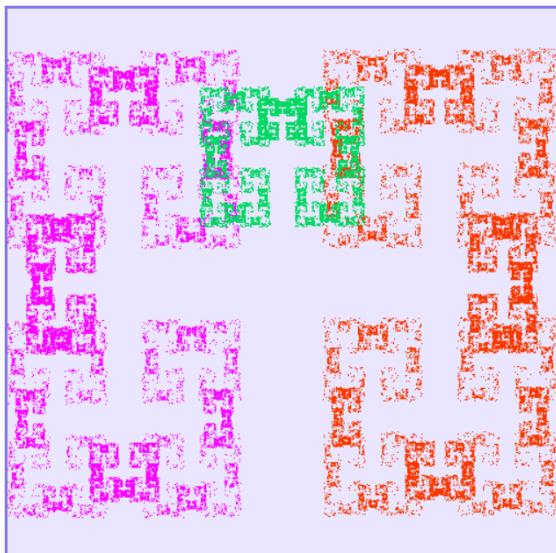
(d) Árbol 2



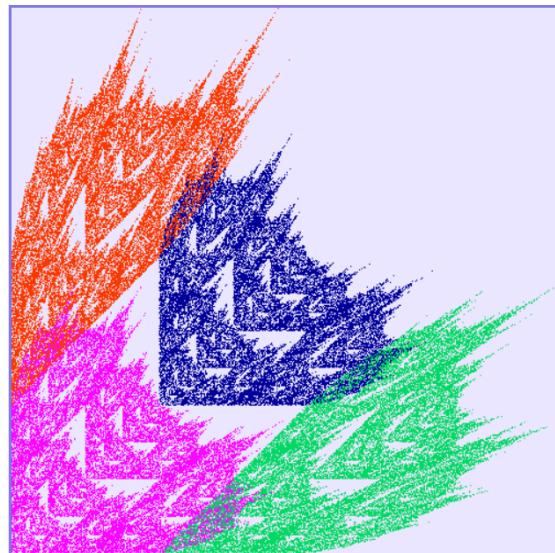
(a) Árbol 3



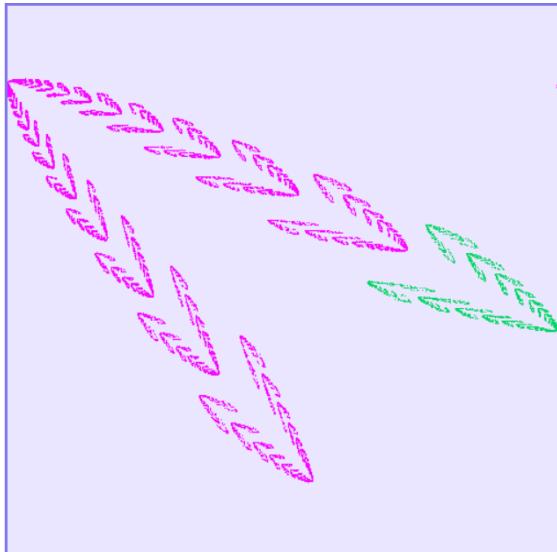
(b) Viento



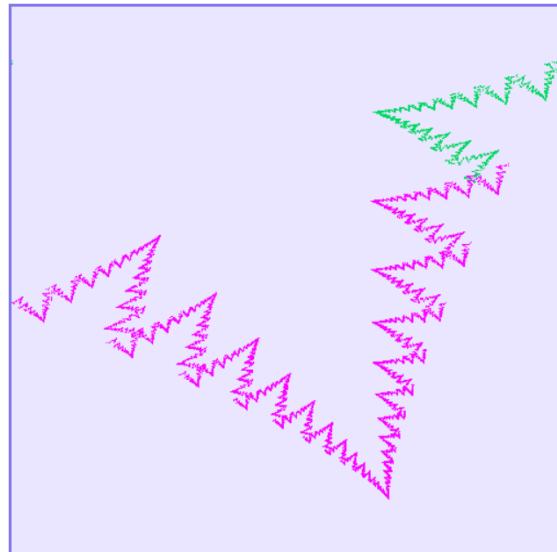
(c) Babilonia



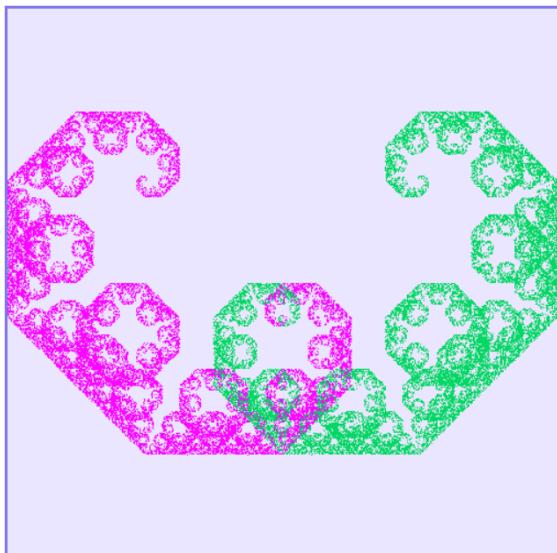
(d) Batman



(a) Boomerang



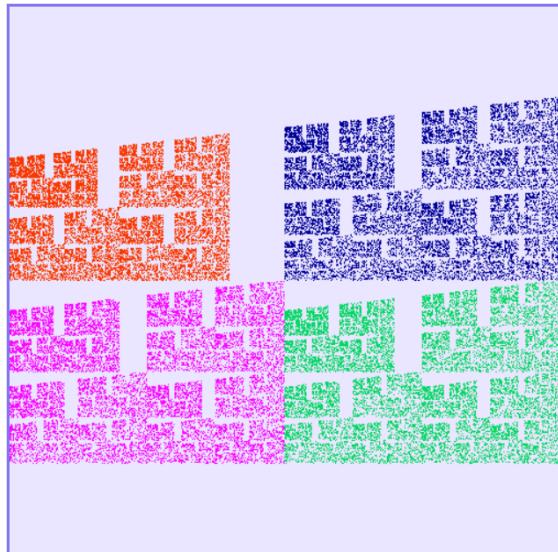
(b) Bosque



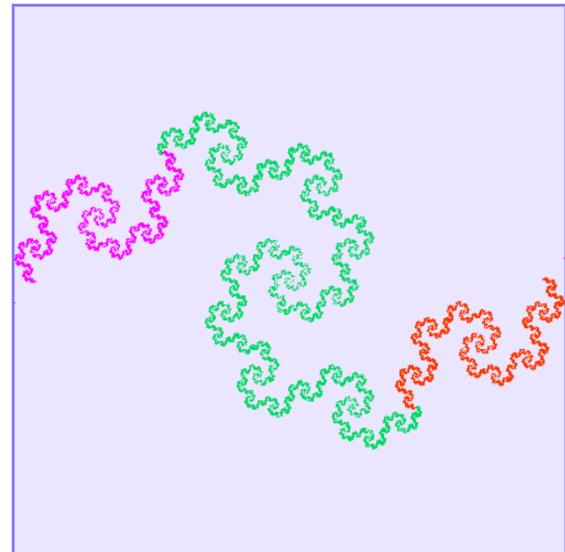
(c) C



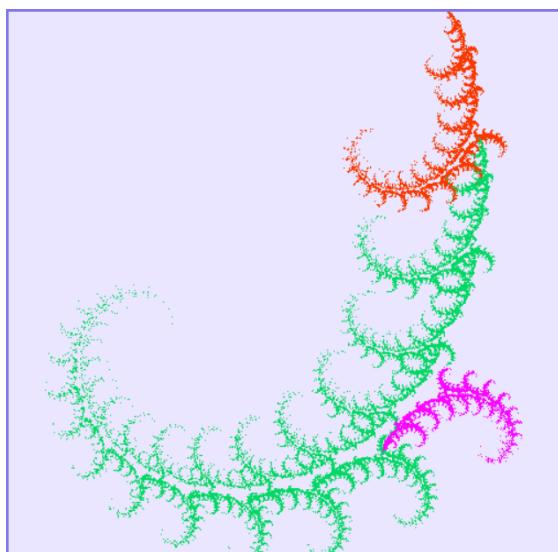
(d) Cadenas



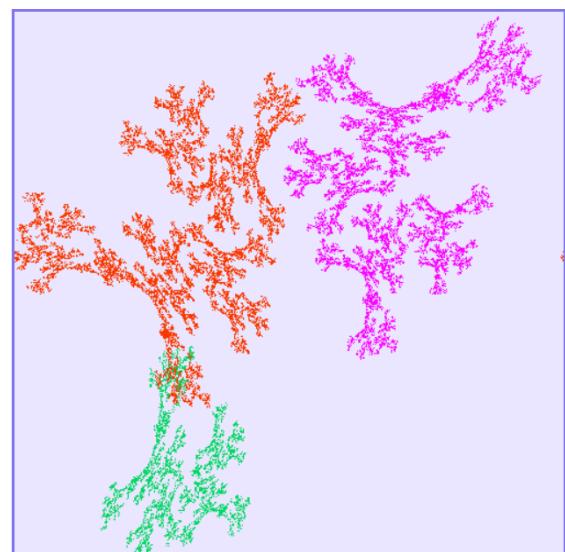
(a) Castillo



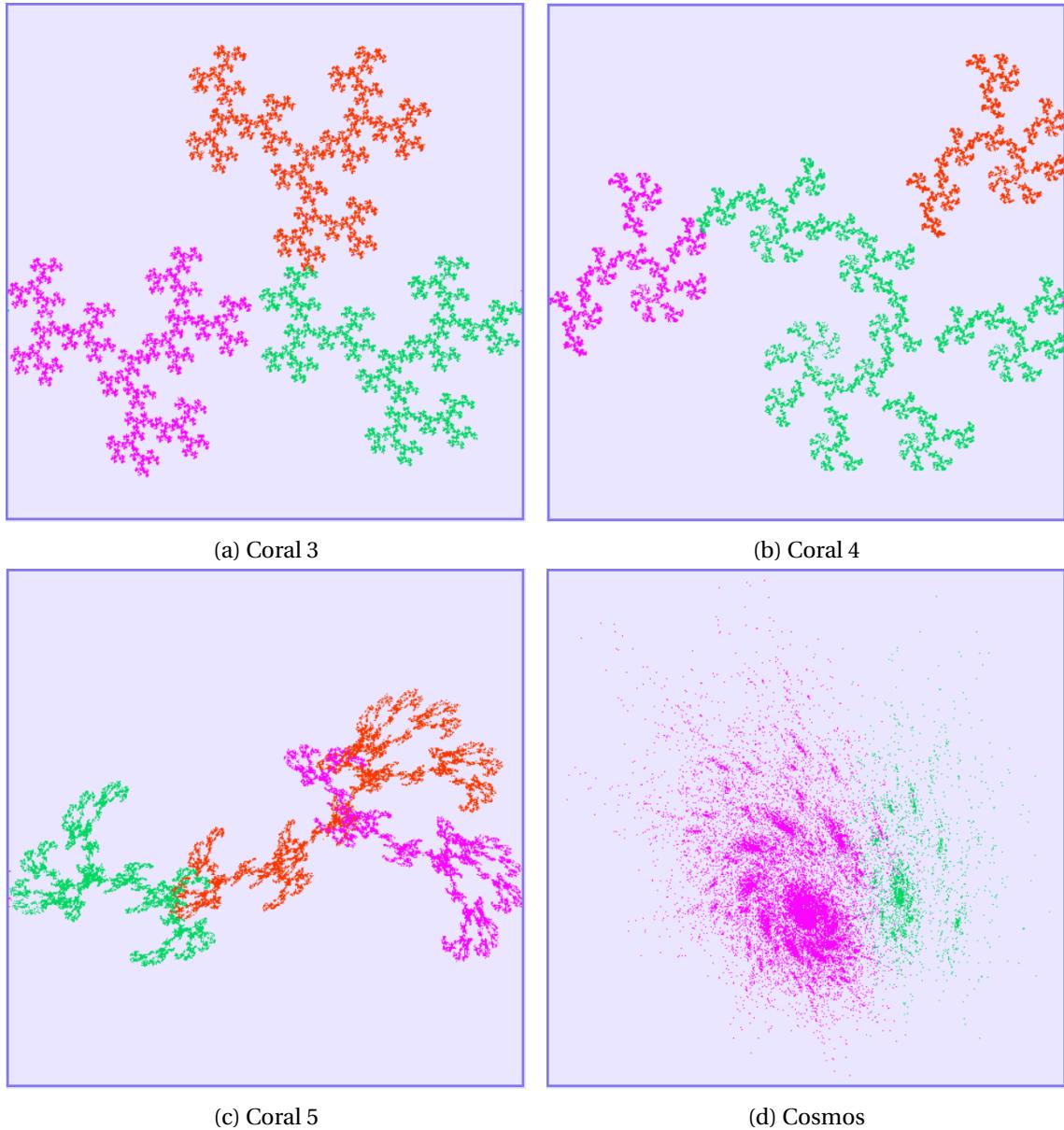
(b) Cinta



(c) Coral 1



(d) Coral 2

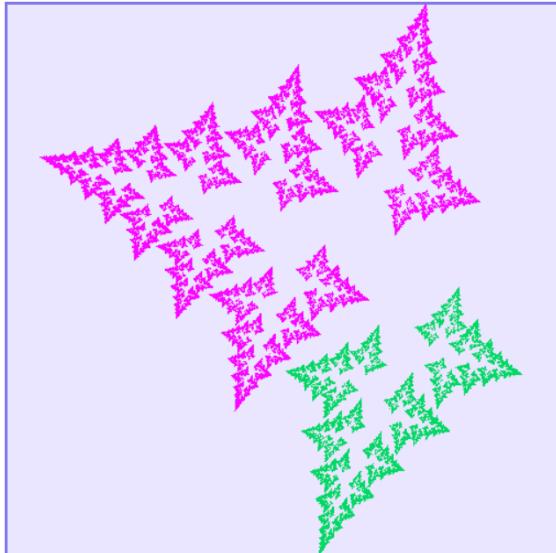


(a) Coral 3

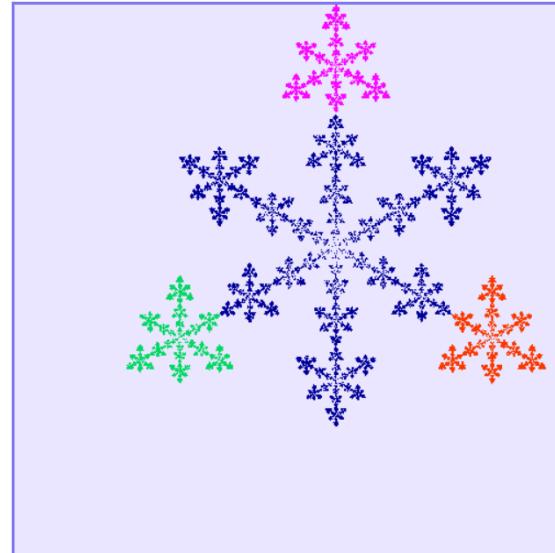
(b) Coral 4

(c) Coral 5

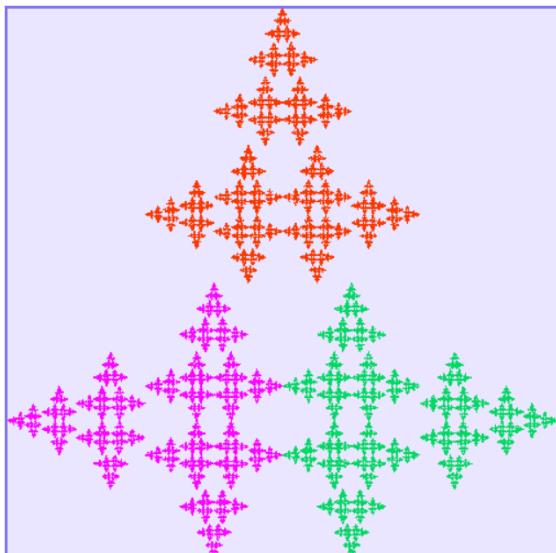
(d) Cosmos



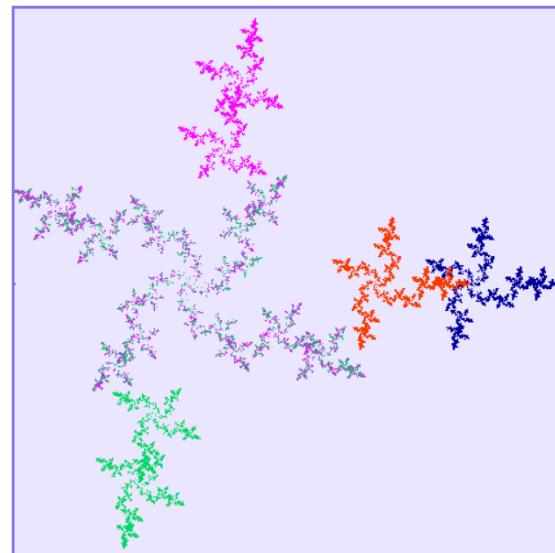
(a) Cristales 1



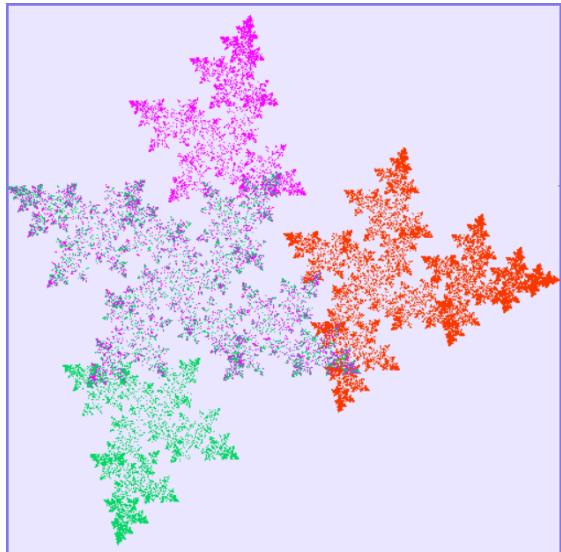
(b) Cristales 2



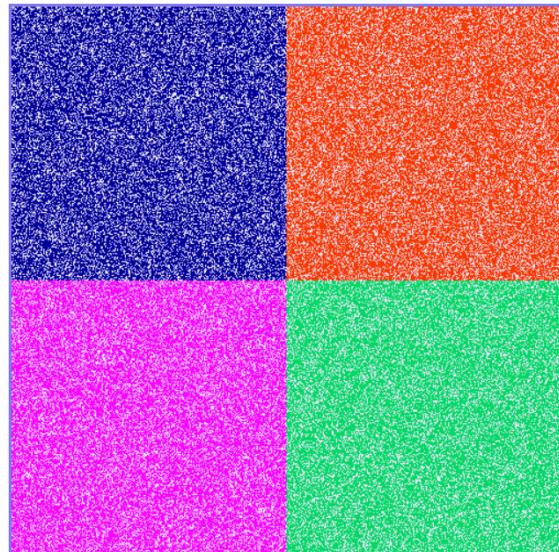
(c) Cristales 3



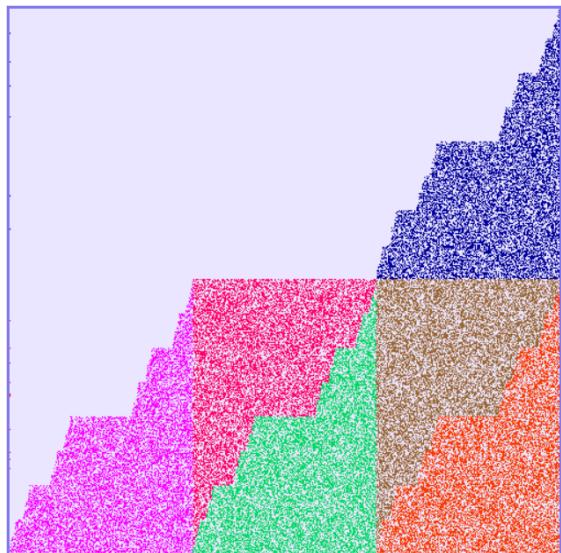
(d) Cristales 4



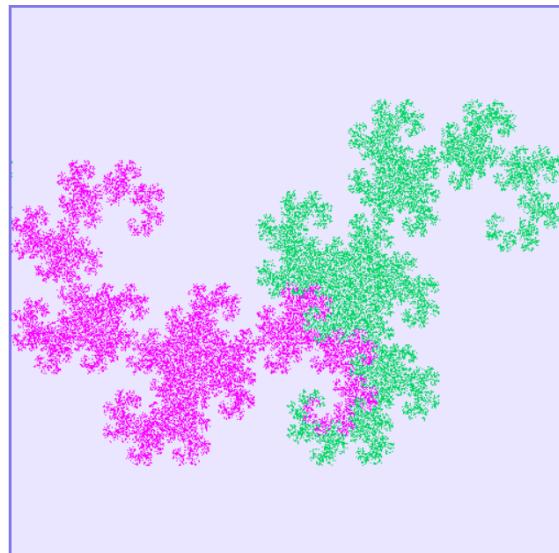
(a) Cristales 5



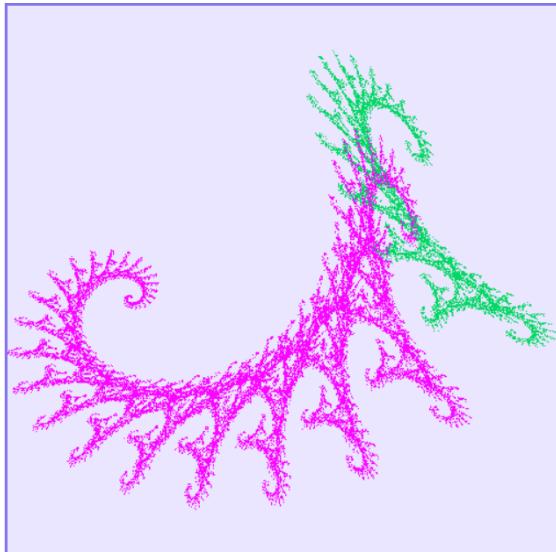
(b) Cuadrado



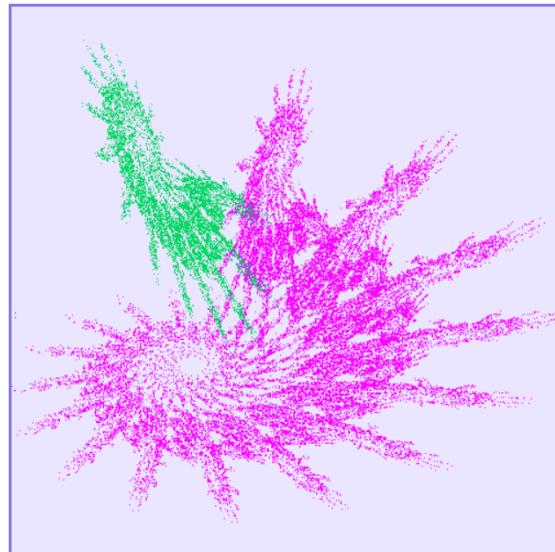
(c) Diablo



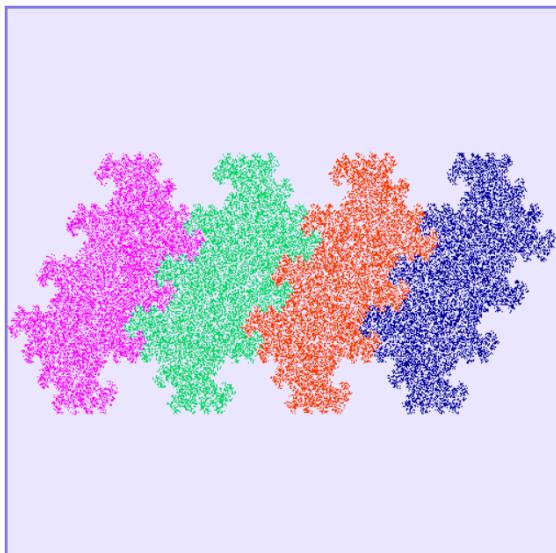
(d) Dragon 1



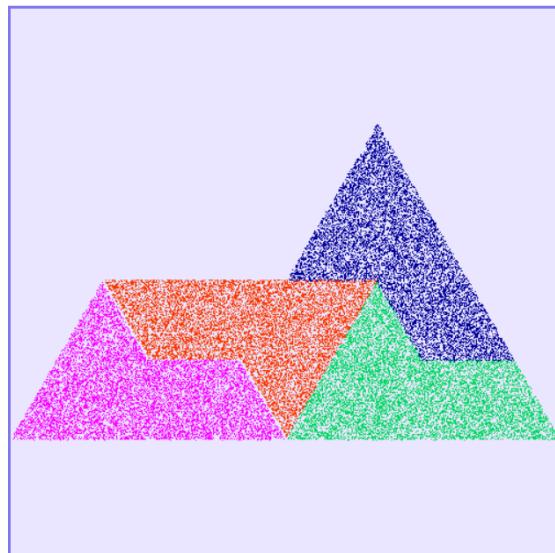
(a) Dragon 2



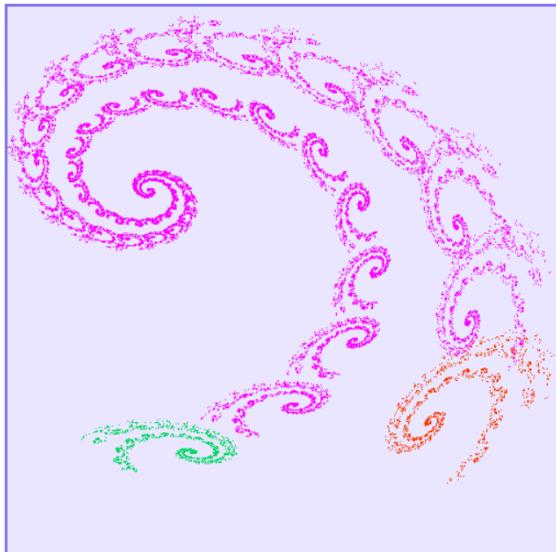
(b) Dragon 3



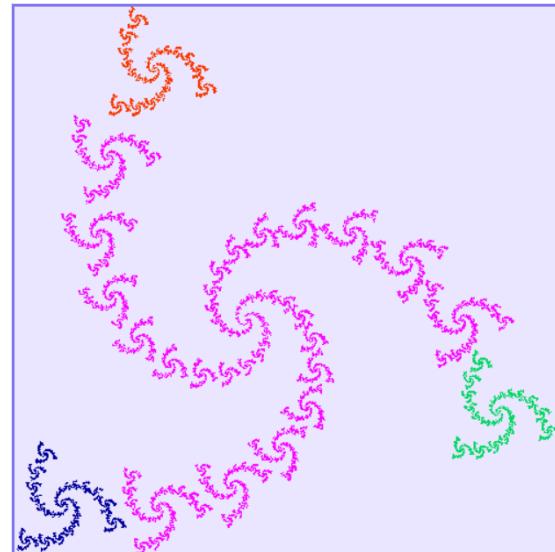
(c) Dragones



(d) Esfinge



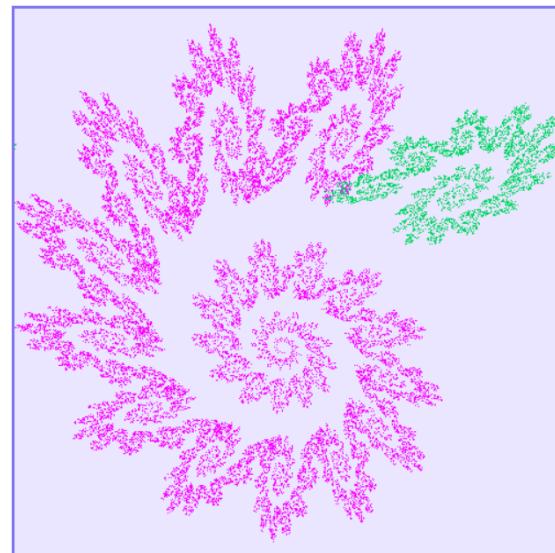
(a) Espiral 1



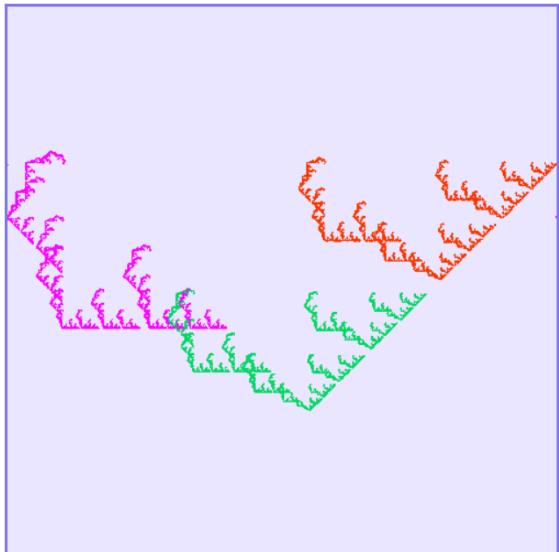
(b) Espiral 2



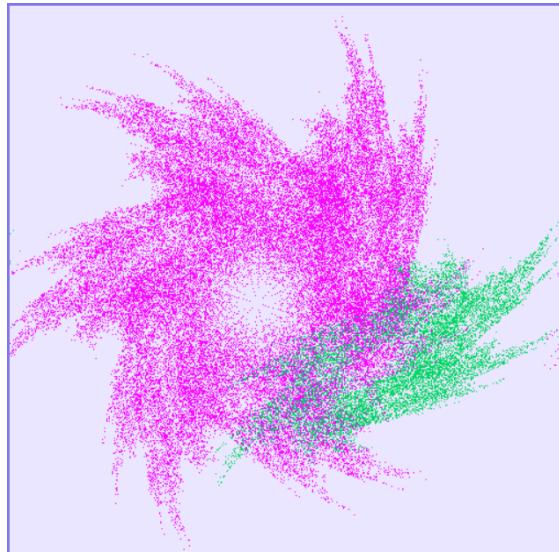
(c) Espiral 3



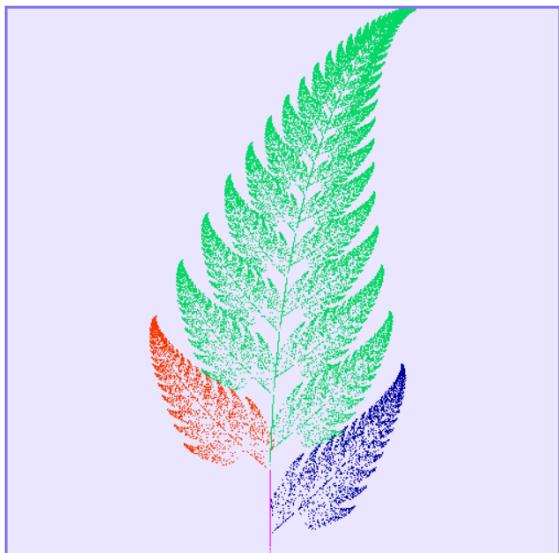
(d) Espiral 4



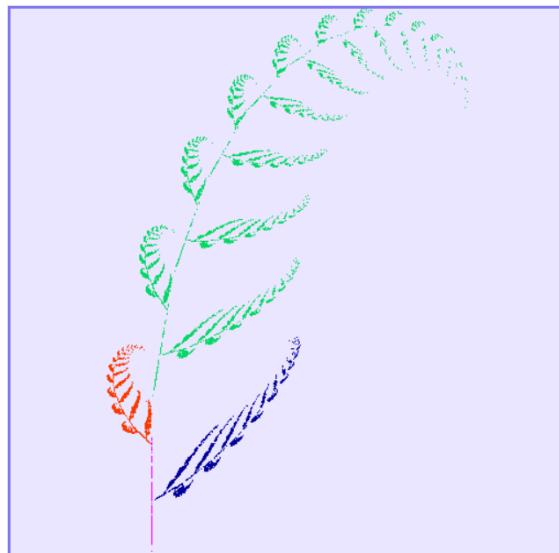
(a) Garra



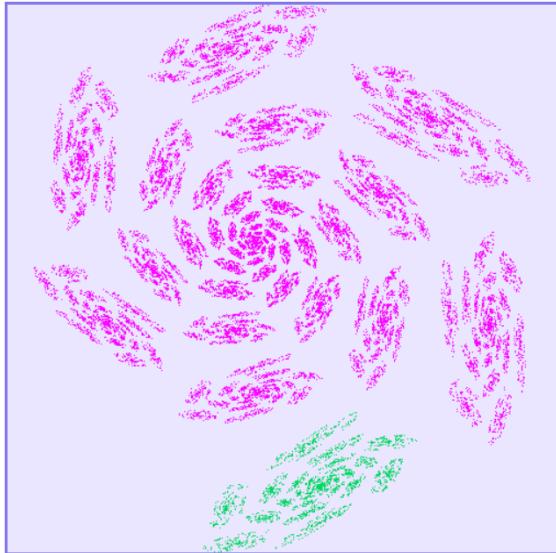
(b) Guirnalda



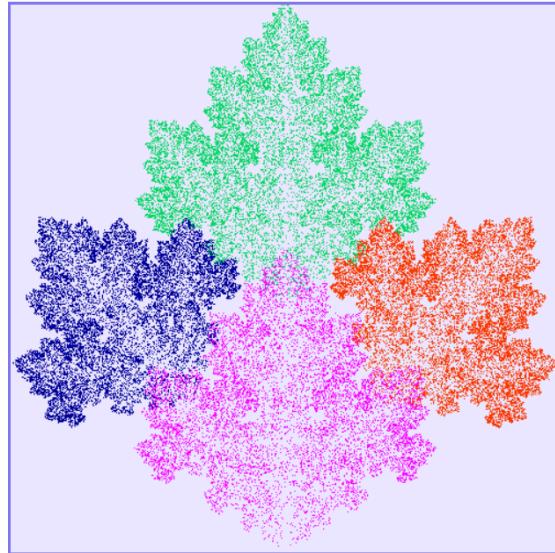
(c) Helecho de Barnsley



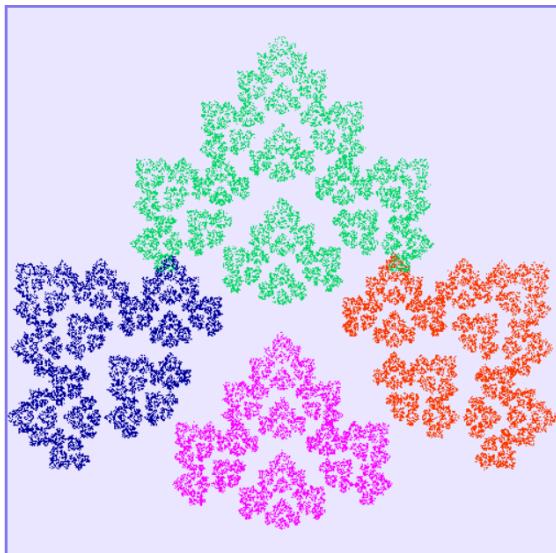
(d) Helecho de Barnsley 2



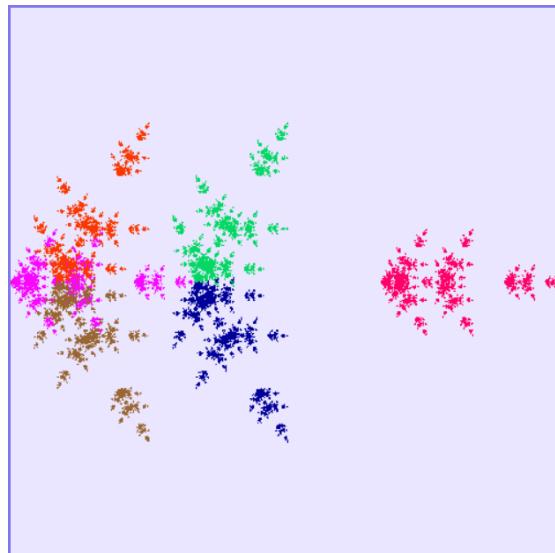
(a) Vórtice



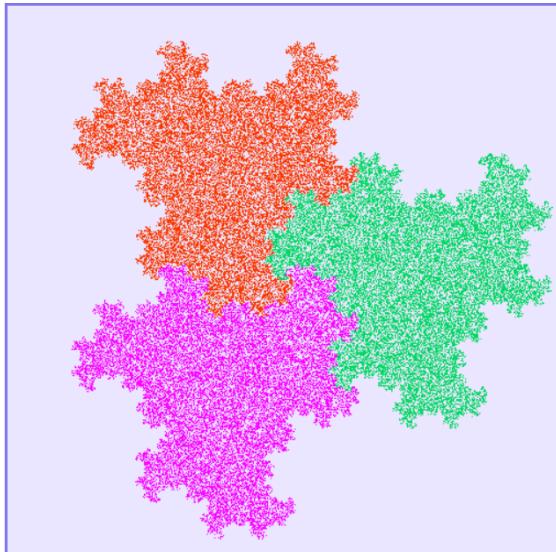
(b) Hoja 1



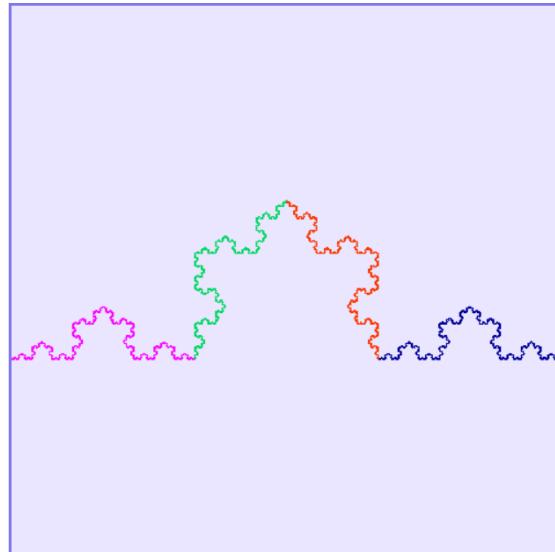
(c) Hoja 2



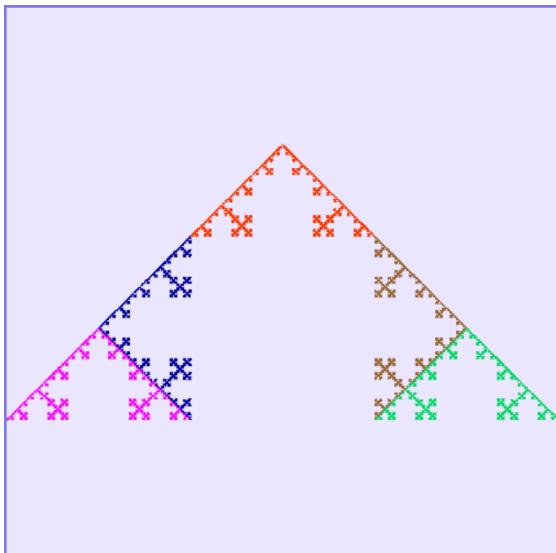
(d) Hoja 3



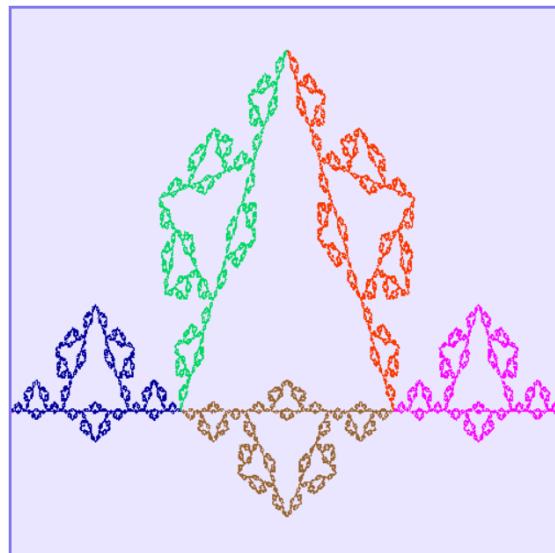
(a) Isla



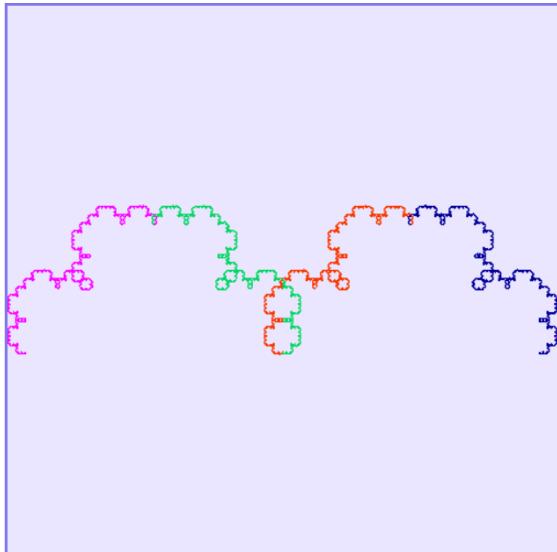
(b) Curva de Koch



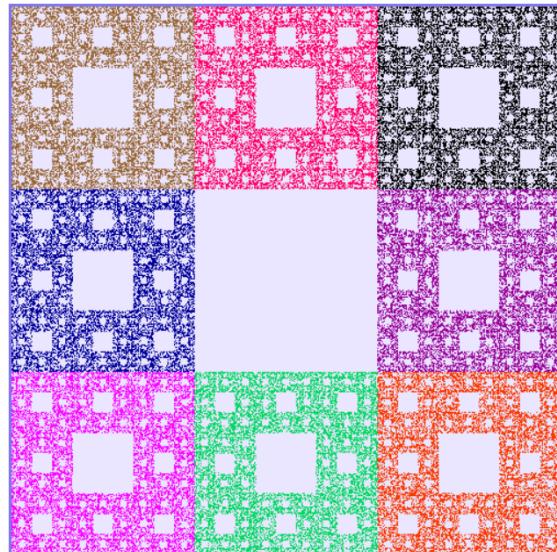
(c) Curva de Koch 2



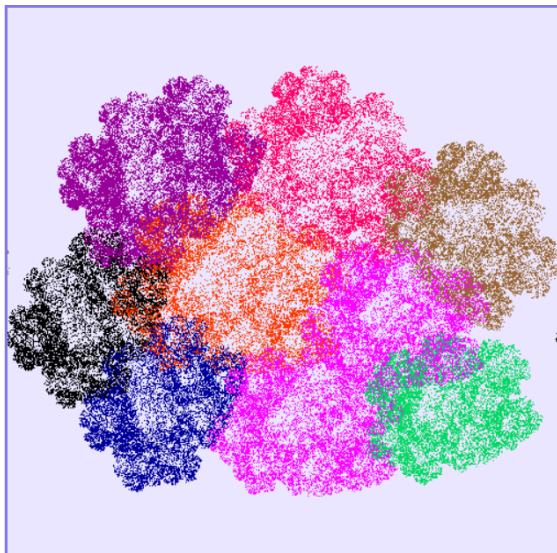
(d) Curva de Koch 3



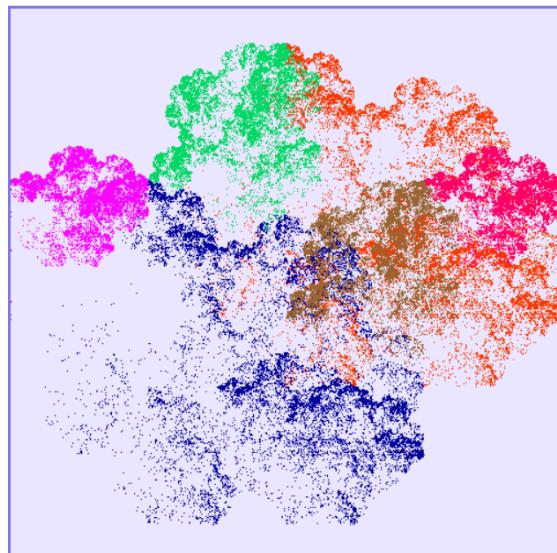
(a) M



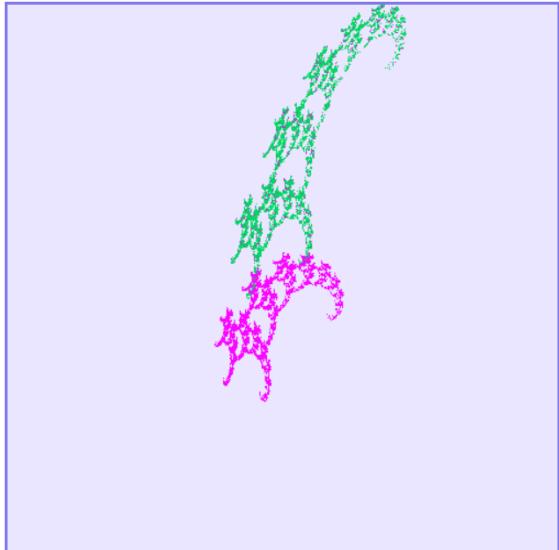
(b) Alfombra de Sierpinski



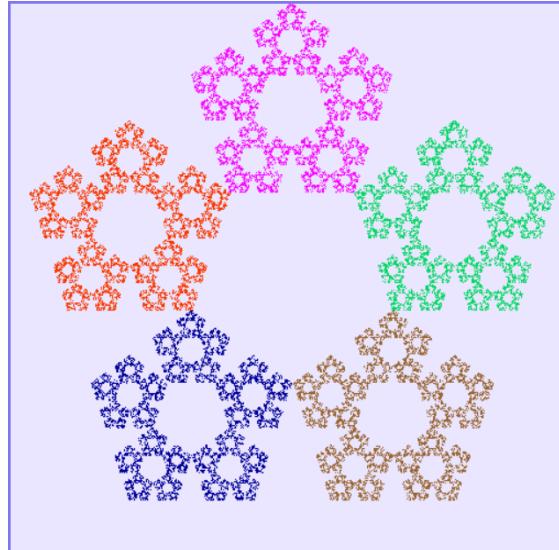
(c) Nubes 1



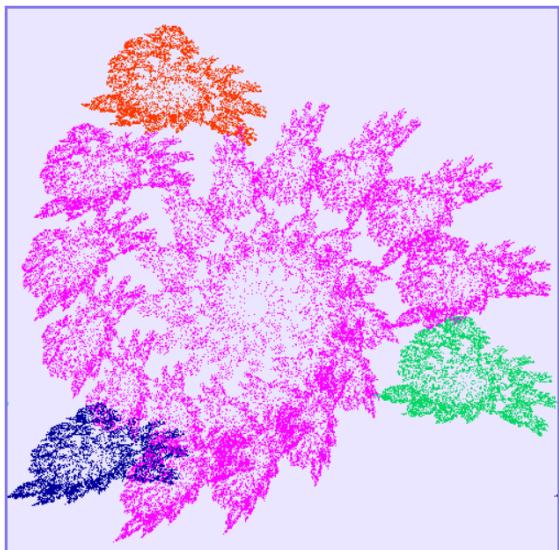
(d) Nubes 2



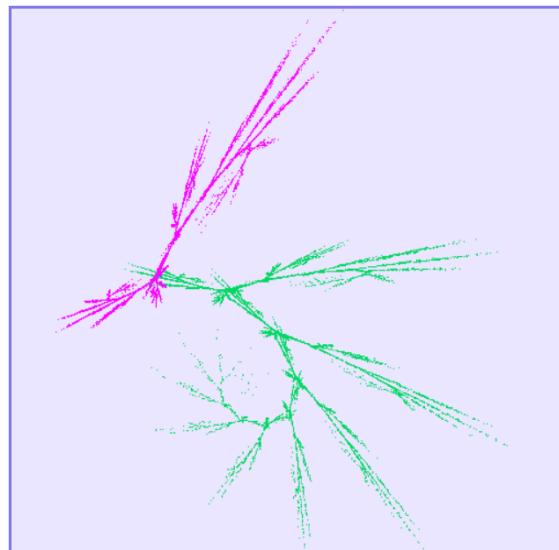
(a) Pata



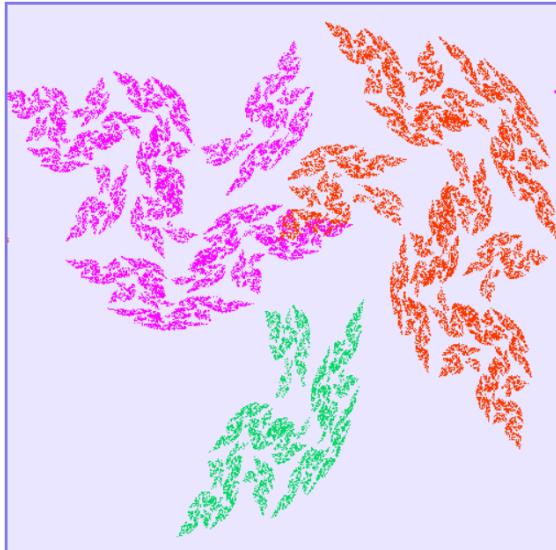
(b) Pentagono



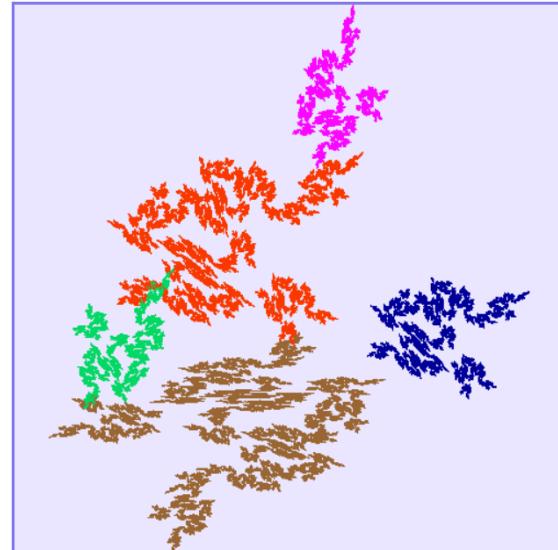
(c) Perros



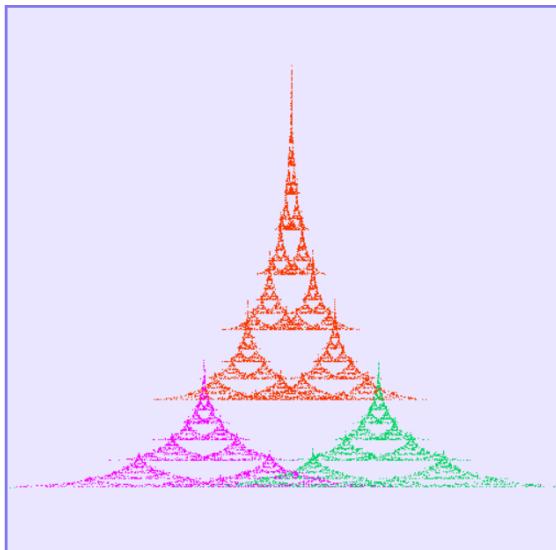
(d) Pez volador



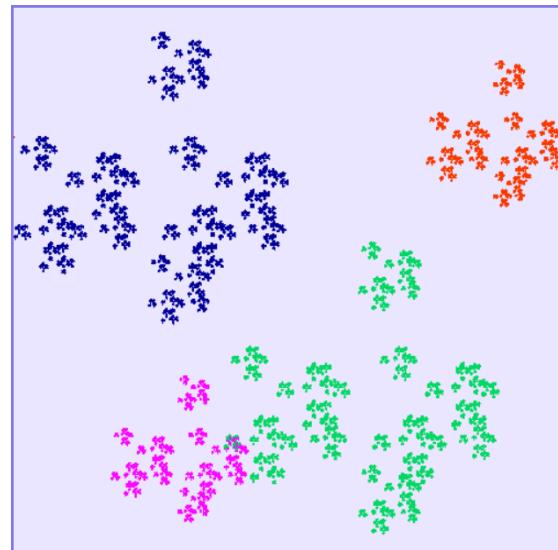
(a) Petalos 1



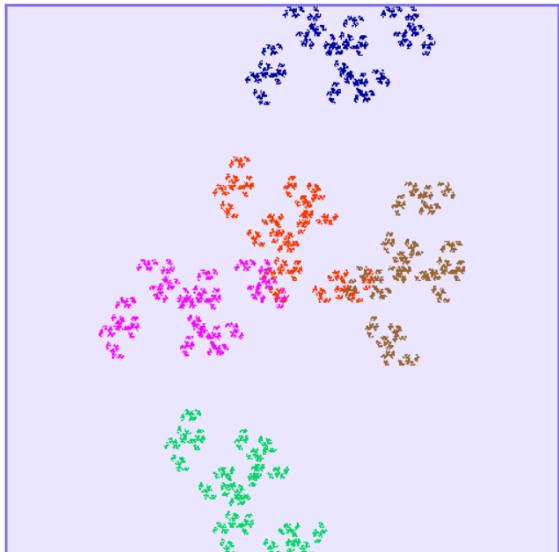
(b) Pétalos 2



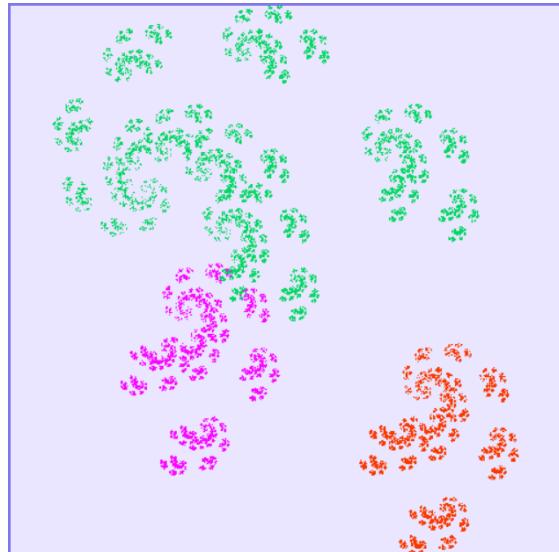
(c) Torre Eiffel



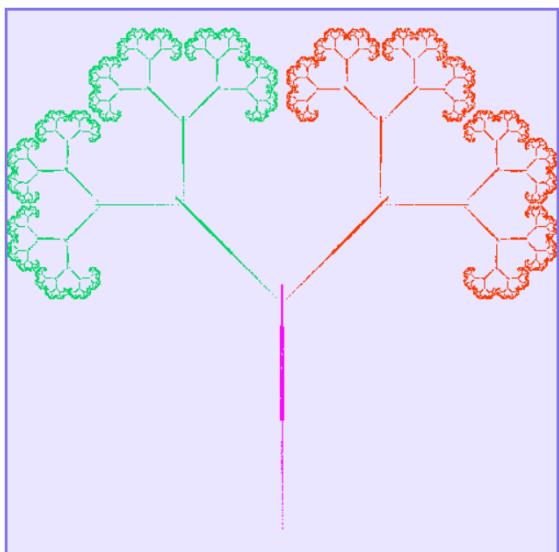
(d) Posies 1



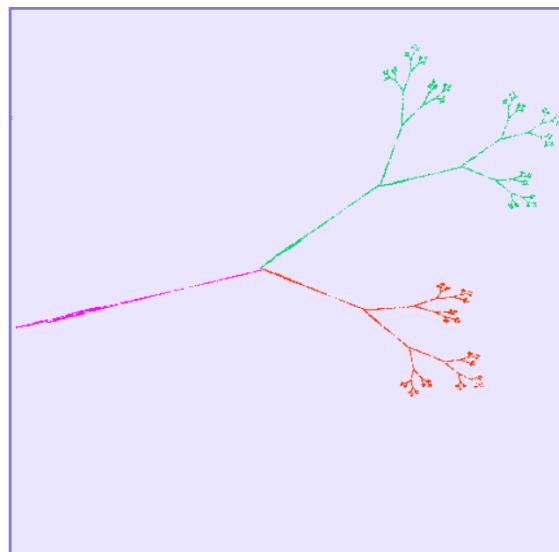
(a) Posies 2



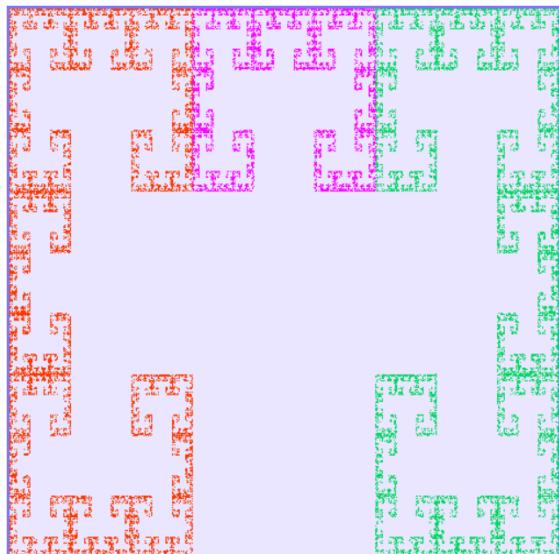
(b) Posies 3



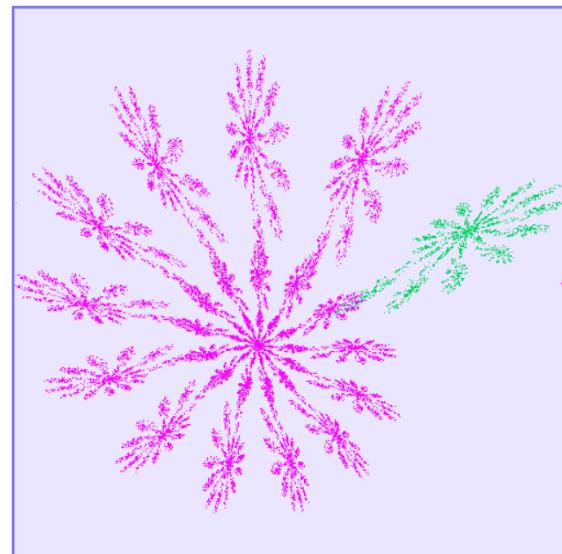
(c) Ramas



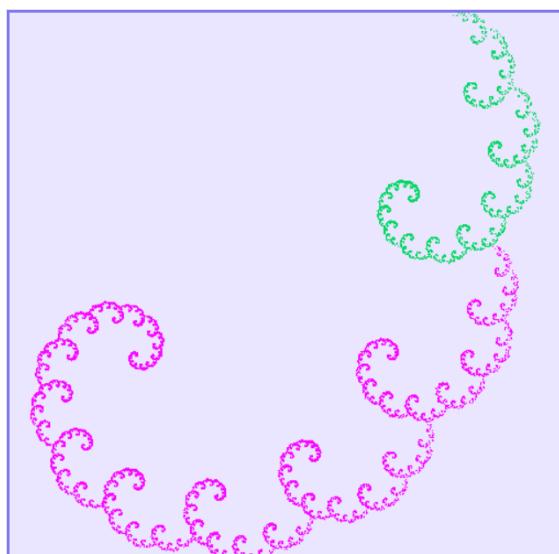
(d) Ramita



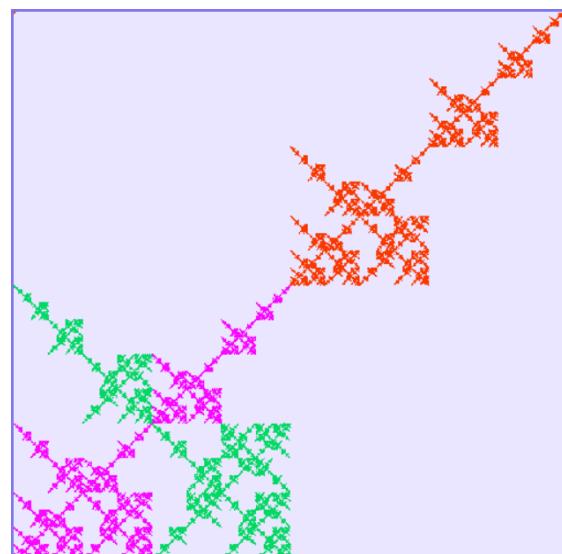
(a) Suelo 3



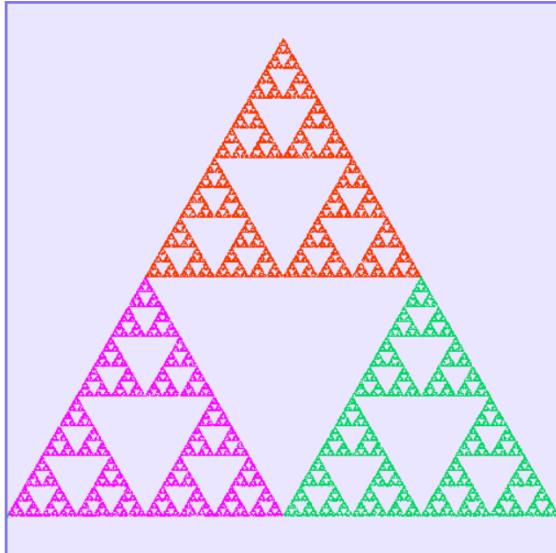
(b) Remolino



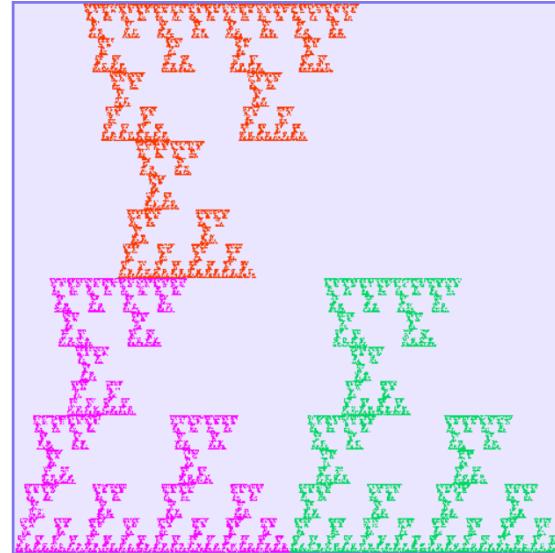
(c) Rizo



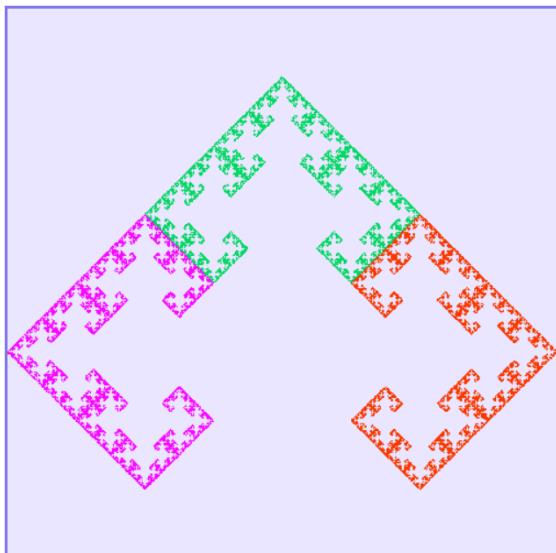
(d) Satélite



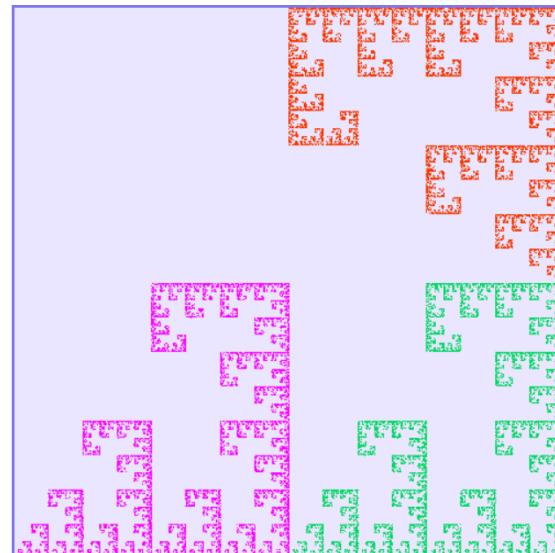
(a) Triángulo Sierpinski



(b) Sigma



(c) Suelo 1



(d) Suelo 2

Bibliografía

- [1] Wikipedia, fractales <https://es.wikipedia.org/wiki/Fractal#:~:text=Un%20fractal%20es%20un%20objeto,naturales%20son%20de%20tipo%20fractal>
- [2] Fractales, um. https://webs.um.es/jmz/DiseGrafSimula/alumnos_08_09/german_rios/index%.files/fractal1_Intro
- [3] MOOC, caos, sistemas de funciones iteradas. UPM. <https://www.youtube.com/watch?v=yQ7n8V1KZCg&t=212s>
- [4] Dibujar fractales con sistemas de funciones iteradas. <http://software-tecnico-libre.es/es/articulo-por-tema/todas-las-secciones/todos-los-temas/todos-los-articulos/dibujar-con-sistemas-de-funciones-iteradas>
- [5] Wikipedia, Transformación afín. https://es.wikipedia.org/wiki/Transformaci%C3%B3n_af%C3%A1n
- [6] PHP, Built-in web server. <https://www.php.net/manual/en/features.commandline.webserver.php>
- [7] Wikipedia, curva de Koch. https://es.wikipedia.org/wiki/Copo_de_nieve_de_Koch
- [8] Wikipedia, tirángulo de Sierpinski. https://es.wikipedia.org/wiki/Tri%C3%A1ngulo_de_Sierpinski
- [9] Wikipedia, Barnsley Fern. https://en.wikipedia.org/wiki/Barnsley_fern
- [10] El definido, Los enigmáticos fractales y sus insospechados usos. <https://eldefinido.cl/actualidad/plazapublica/7240/Los-enigmaticos-fractales-y-sus-insospechados-usos/>
- [11] BBVA, OpenMind, Aplicaciones de la geometría fractal: del cambio climático al cáncer. <https://www.bbvaopenmind.com/ciencia/matematicas/aplicaciones-la-geometria-fractal-del-cambio-climatico-al-cancer/#:~:text=Adem%C3%A1s%20de%20los%20fen%C3%B3menos%20que,valores%20o%20la%20composici%C3%B3n%20musical.>
- [12] w3Schools, HTML5 Canvas, ImageData property. https://www.w3schools.com/jsref/canvas_imagedata_data.asp
- [13] Wikipedia, Michael Barnsley. https://en.wikipedia.org/wiki/Michael_Barnsley
- [14] MOOC, caos, Dimensión fractal. UPM. <https://www.youtube.com/watch?v=-36sHrtrqKw&feature=youtu.be>

Documentación adicional sobre HTML5, CSS y JavaScript

- [15] HTML5 Canvas: a tutorial for beginners <https://www.w3resource.com/html5-canvas/>
- [16] Quick CSS Guide https://www.tutorialspoint.com/css/css_quick_guide.htm

[17] A JavaScript Beginner's Essentials: cheat sheet <https://websitesetup.org/wp-content/uploads/2020/09/Javascript-Cheat-Sheet.pdf>

Repositorio del proyecto

[18] <https://github.com/beagaliana/mc2021>