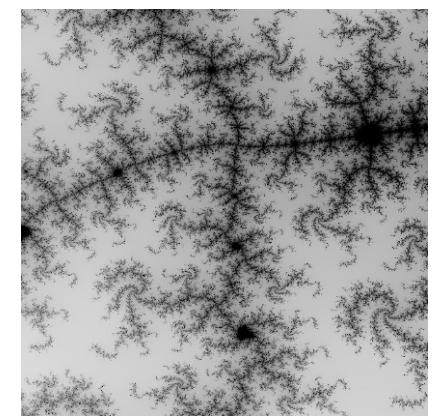
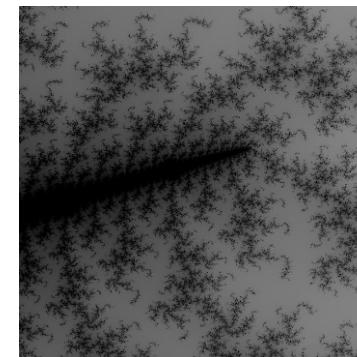
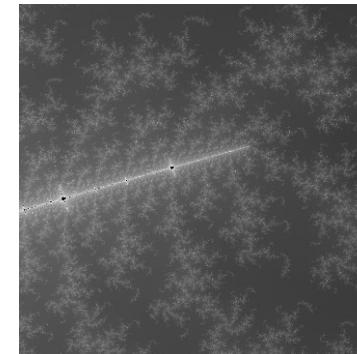
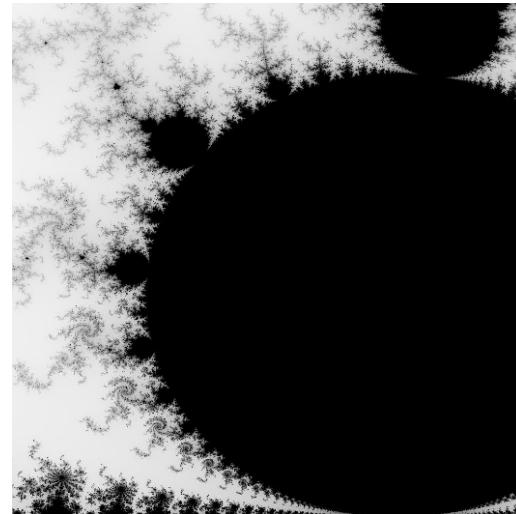
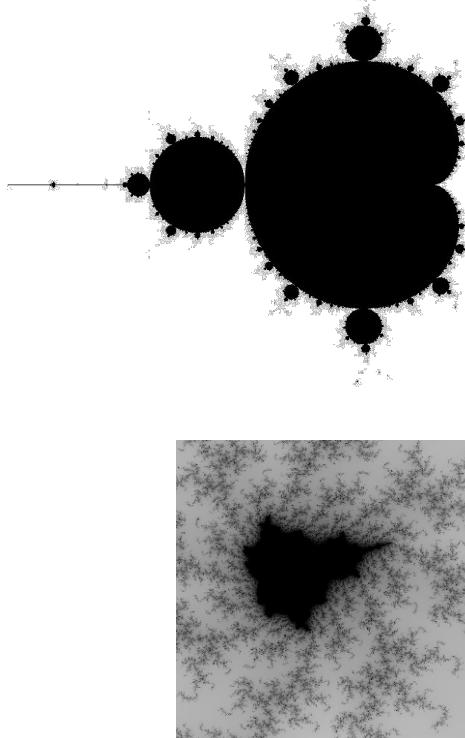


# Introducción a los fractales: el conjunto de Mandelbrot

- Los fractales son representaciones gráficas de funciones matemáticas que suelen parecerse a paisajes y formas de la naturaleza, donde pueden observarse muchos ejemplos de estructuras repetitivas que configuran hojas, dunas, etc.
- Como el objetivo no es dominar la teoría de fractales, lo que se abordará es la representación parcial de un fractal conocido como conjunto de Mandelbrot.

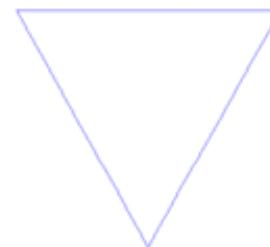


# Introducción a los fractales: el conjunto de Mandelbrot

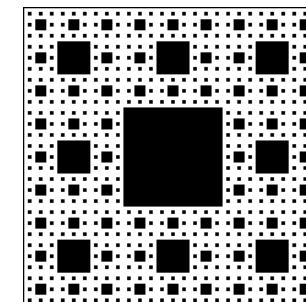
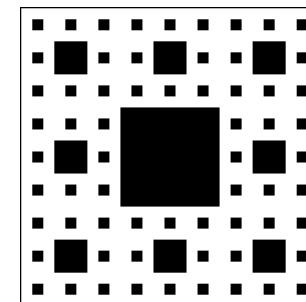
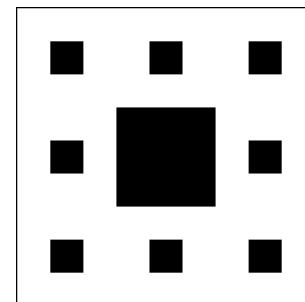
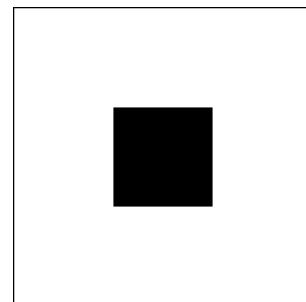
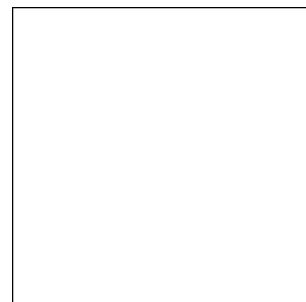
Sucesivos pasos de la construcción  
de la Curva de Koch



Tres de estas curvas unidas forman  
el copo de nieve de Koch



Construcción de la alfombra de Sierpinski



Paso 1  
(semilla)

Paso 2

Paso 3

Paso 4

Paso 5

# Introducción a los fractales: el conjunto de Mandelbrot

- El conjunto de Mandelbrot está compuesto por los números complejos,  $c = a+ib$ , para los que la relación de recurrencia:

$$z_{n+1} = z_n^2 + c \quad \text{para } n = 0, 1, 2, \dots$$

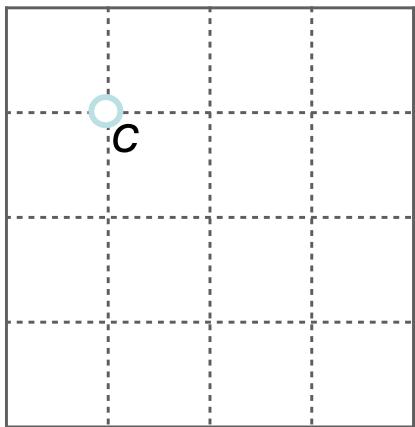
converge a un número complejo finito, siendo  $z_0 = 0$  la condición inicial.

- Se demuestra que si existe un  $m$  para el cual

$$|z_m| > 2,$$

la serie diverge y  $c$  no pertenece al conjunto de Mandelbrot.

- Como el  $m$ , de existir, podría ser muy grande, la primera aproximación que se hace es llegar como máximo hasta un número preestablecido de iteraciones  $z_{IterMax}$ .
- Si se alcanza dicho  $z_{IterMax}$  sin que ningún  $|z_n|$  sea mayor que 2, se asume que el número complejo  $c$  pertenece al conjunto de Mandelbrot.



$c = a + bi = (a, b)$  ¿c está en el conjunto de Mandelbrot?

$$\mathbf{z_{n+1} = z_n^2 + c, con z_0 = 0}$$

$$z_0 = 0$$

$$z_1 = c$$

$$z_2 = c^2 + c$$

$$z_3 = (c^2 + c)^2 + c$$

...

- Si  $|z_m| > 2 \rightarrow$  c **NO** está en el conjunto de Mandelbrot

- Si se llega a una iteración máxima fijada IterMax sin que se verifique que  $|z_m| > 2$



c **SI** está en el conjunto de Mandelbrot y lo pintamos de negro

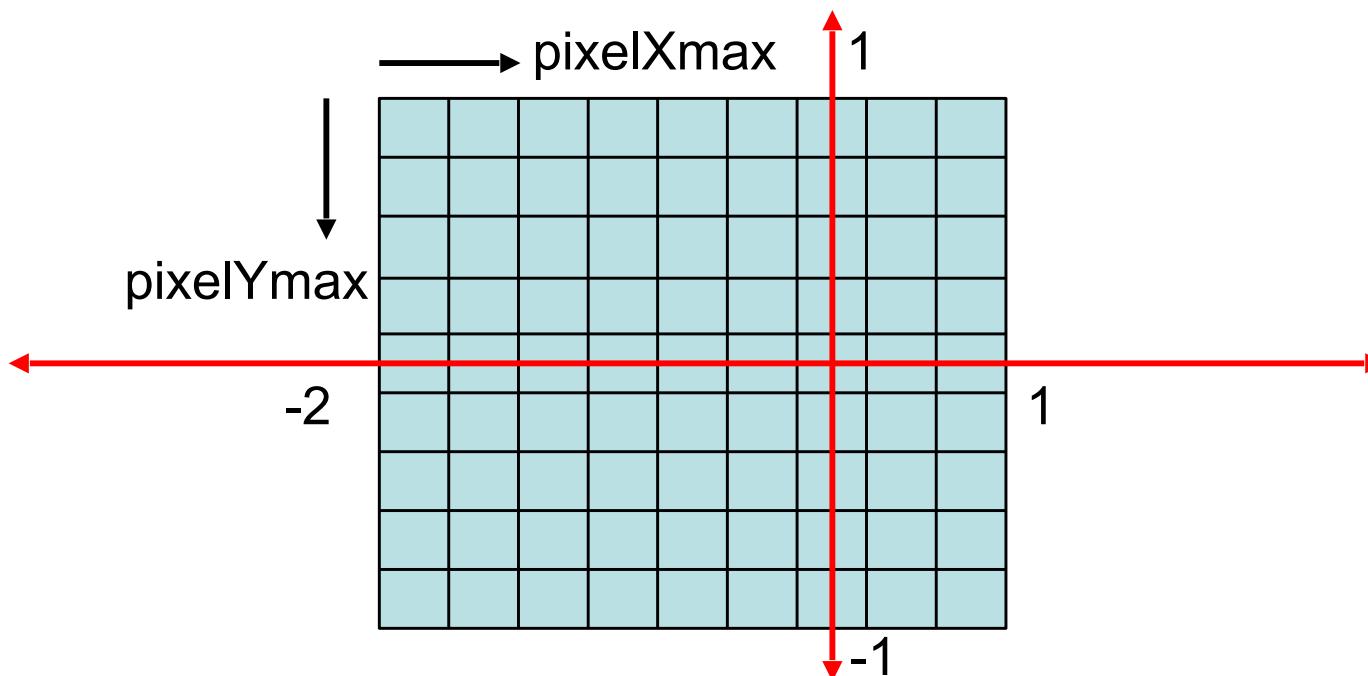
# Introducción a los fractales: el conjunto de Mandelbrot

## DIBUJANDO EL CONJUNTO DE MANDELBROT

- Establecer el tamaño de la imagen, número de píxeles horizontales y número de píxeles verticales, e.g.:

$$\text{pixelXmax} = 10, \text{pixelYmax} = 10$$

- Seleccionar un área del plano complejo donde vamos a estudiar que puntos son de Mandelbrot, es decir, establecer el rango para los valores de  $c = a+bi$ .
  - Un rango para la parte real:  $[\text{RealMin}, \text{RealMax}] \rightarrow$  e.g.,  $[-2, 1]$
  - Un rango para la parte imaginaria:  $[\text{ImMin}, \text{ImMax}] \rightarrow$  e.g.,  $[-1, 1]$



# Introducción a los fractales: el conjunto de Mandelbrot

## DIBUJANDO EL CONJUNTO DE MANDELBROT

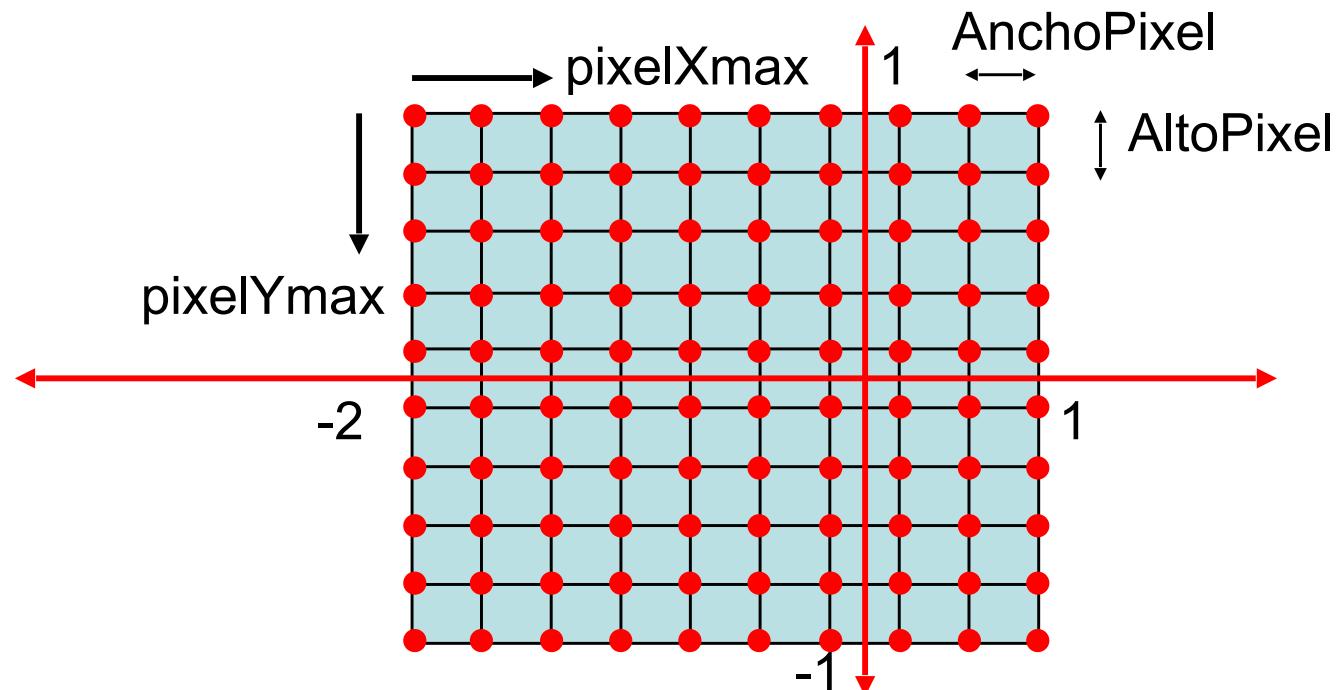
Dividir el área del plano complejo en puntos que correspondan a cada pixel, teniendo en cuenta el tamaño de la imagen.

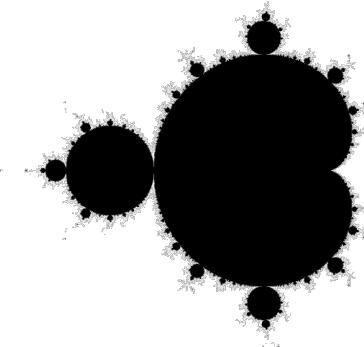
$$\text{AnchoPixel} = (\text{RealMax} - \text{RealMin}) / (\text{pixelXmax} - 1) = (1 - (-2)) / (10 - 1) = 3/9 = 1/3$$

$$\text{AltoPixel} = (\text{ImMax} - \text{ImMin}) / (\text{pixelYmax} - 1) = (1 - (-1)) / (10 - 1) = 2/9$$

```
For (pixelY=0; pixelY<pixelYmax; pixelY++)
    Cimg = ImMin + pixelY*AltoPixel;
    for(pixelX=0;pixelX<pixelXmax;pixelX++)
        Creal = RealMin + pixelX*AnchoPixel;
```

RealMax = 1  
 RealMin = -2  
 ImMax = 1  
 ImMin = -1  
 pixelXmax = 10  
 pixelYmax = 10





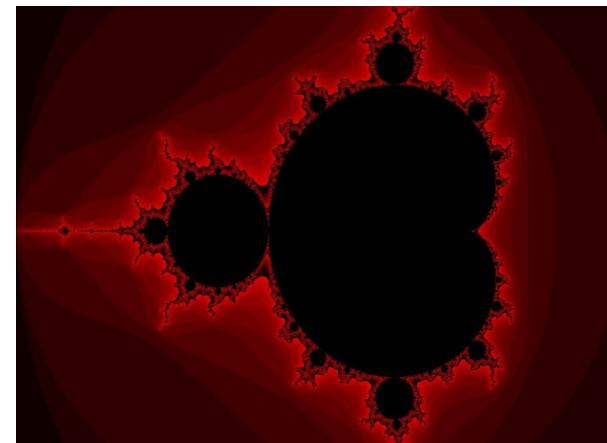
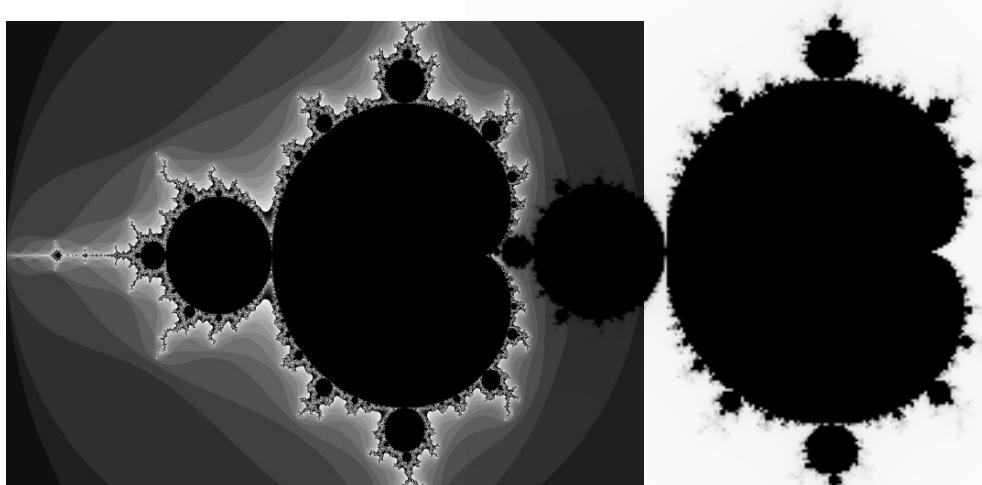
# Introducción a los fractales: el conjunto de Mandelbrot

## DIBUJANDO EL CONJUNTO DE MANDELBROT

- Seleccionar una paleta de colores y un número máximo de iteraciones.
- Para cada pixel en la imagen, aplicar la ecuación recursiva de Mandelbrot al valor correspondiente de  $c$ :

$$c = C_{real} + i \cdot C_{img}.$$

- Si el punto  $c$  resulta estar en el **conjunto** (para un número máximo de iteraciones preestablecidas), colorear el pixel como **negro**.
- En otro caso, seleccionar un **color** de la paleta de colores basándose en el **número de iteraciones** necesarias para que  $|z_m| > 2$ .



# Introducción a los fractales: el conjunto de Mandelbrot

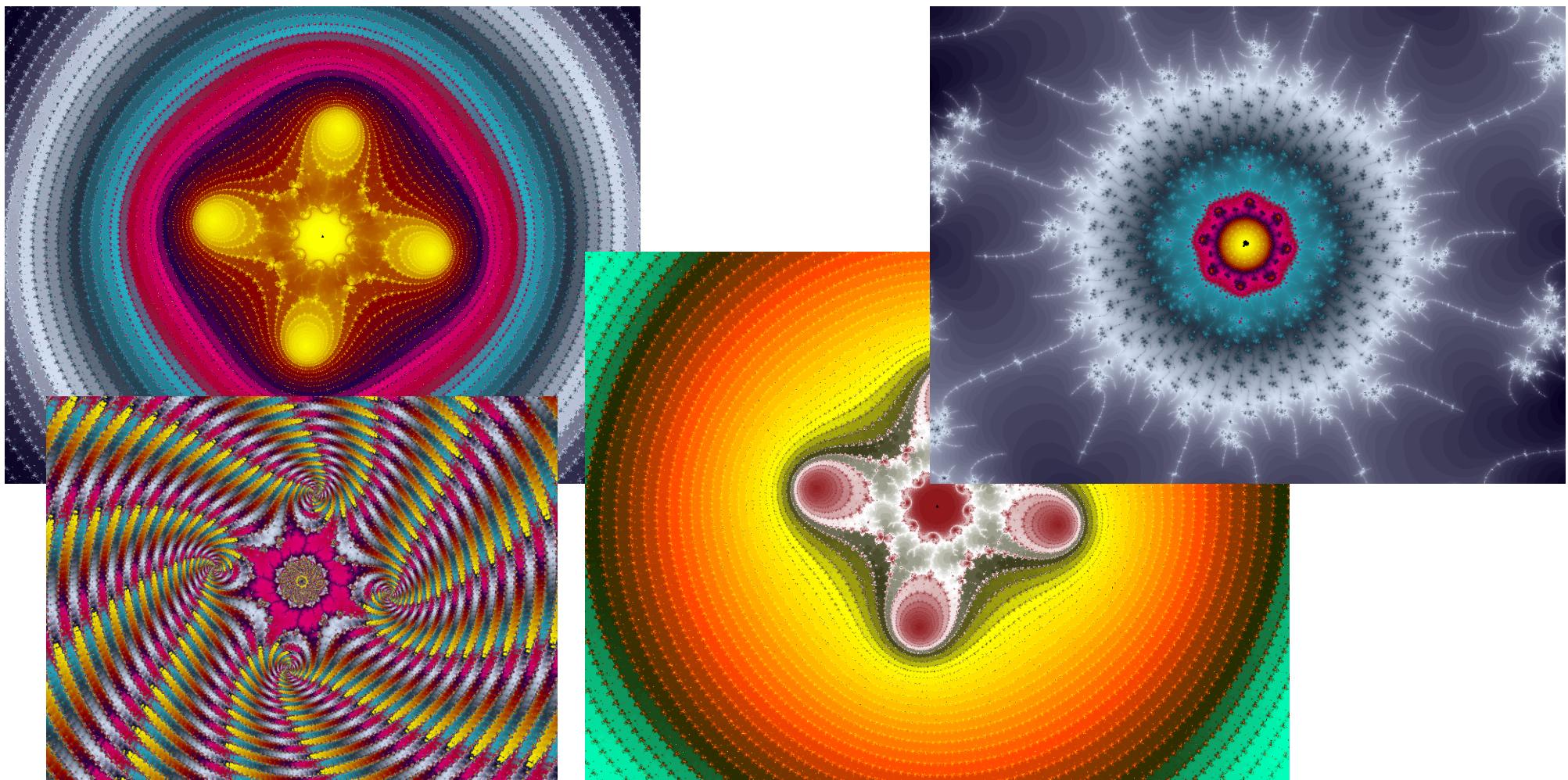
## Dibujando el conjunto de Mandelbrot

- La complejidad y, por tanto, el tiempo de cómputo depende de dos variables:
  - La ampliación de imagen que queramos realizar.
    - Depende del dominio del plano complejo que tomemos.
    - Cuanto más pequeño, más precisión necesitamos.
    - El tipo de dato double sólo permite ampliación del orden de  $10^{14}$ .
    - Ampliaciones del orden de  $10^{120}$  sólo pueden conseguirse con librerías específicas como [GNU Multiple Precision Arithmetic Library](#).
  - El máximo número de iteraciones necesarias para determinar si un punto está en el conjunto de Mandelbrot o no.
    - Dependiendo de la complejidad de la zona del plano complejo que estemos analizando, este número de iteraciones puede variar desde unas pocas decenas hasta 1.000.000.000 (mil millones) de iteraciones (o incluso billones).
- Todo ello hace que, en algunos casos, el tiempo de cómputo sea de varios meses.

# Introducción a los fractales: el conjunto de Mandelbrot

## Dibujando el conjunto de Mandelbrot

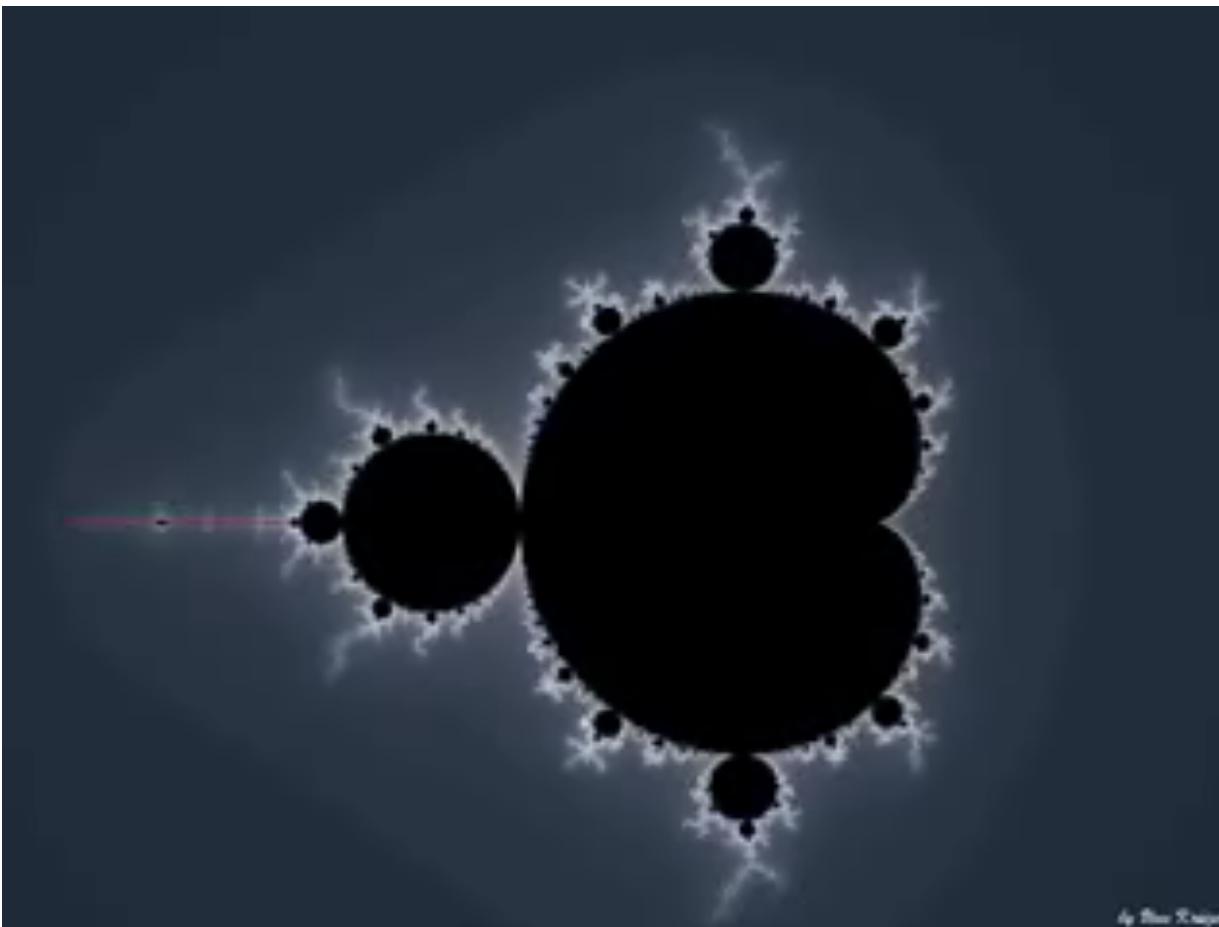
Todo ello hace necesaria la inclusión del **parallelismo** para poder tratar este tipo de problemas y poder conseguir imágenes como, por ejemplo, estas:



# Introducción a los fractales: el conjunto de Mandelbrot

## DIBUJANDO EL CONJUNTO DE MANDELBROT

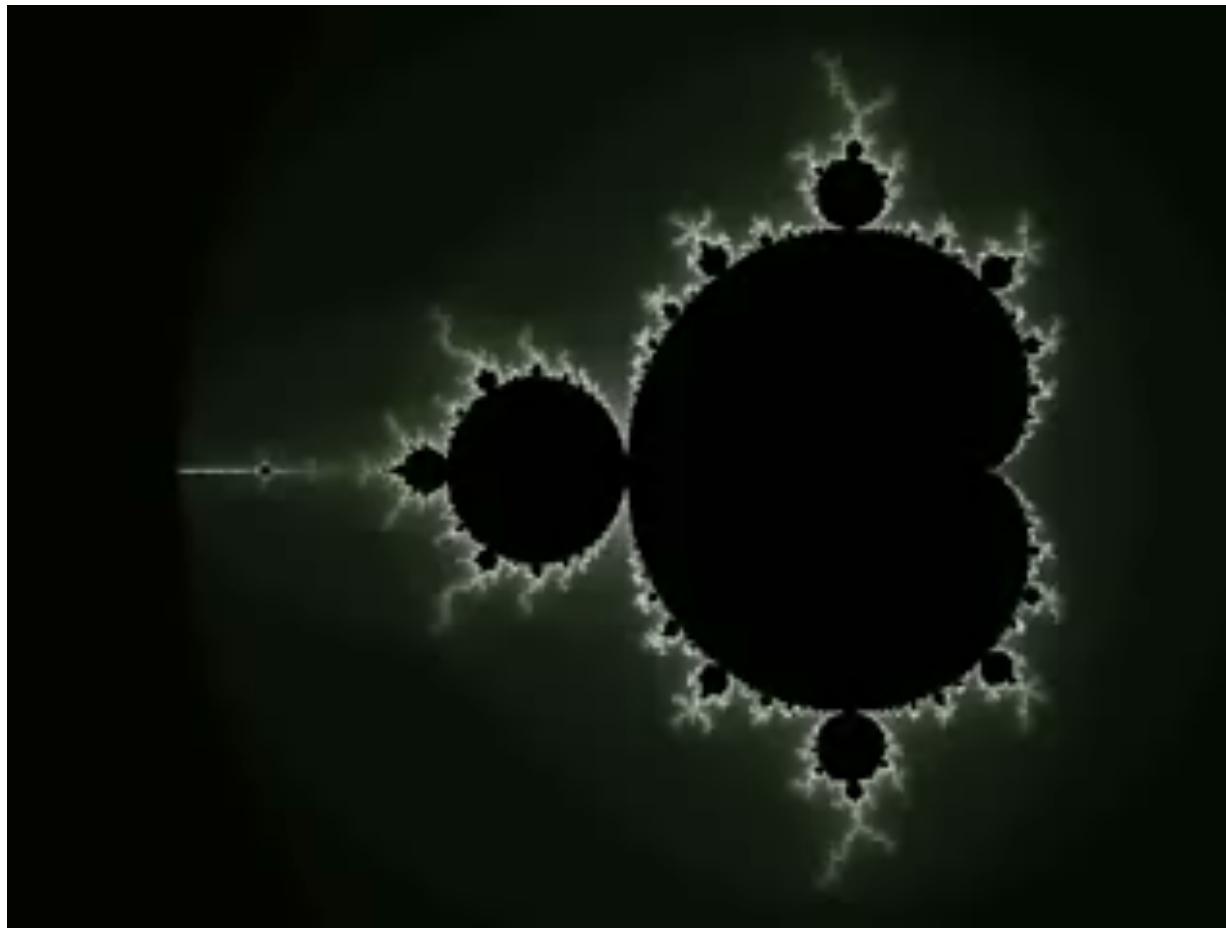
Todo ello hace necesaria la inclusión del **paralelismo** para poder tratar este tipo de problemas y poder conseguir imágenes como, por ejemplo, estas:



# Introducción a los fractales: el conjunto de Mandelbrot

## DIBUJANDO EL CONJUNTO DE MANDELBROT

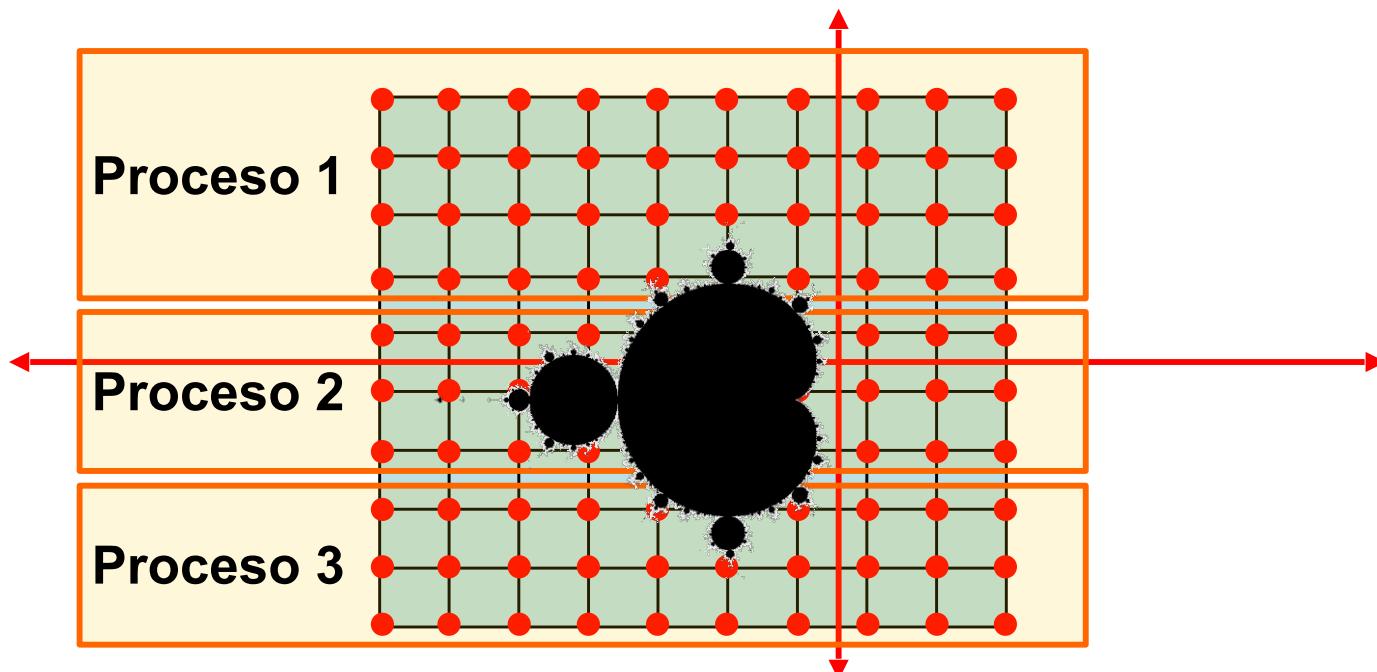
Todo ello hace necesaria la inclusión del **paralelismo** para poder tratar este tipo de problemas y poder conseguir imágenes como, por ejemplo, estas:



# Introducción a los fractales: el conjunto de Mandelbrot

## PARALELIZACIÓN DEL ALGORITMO

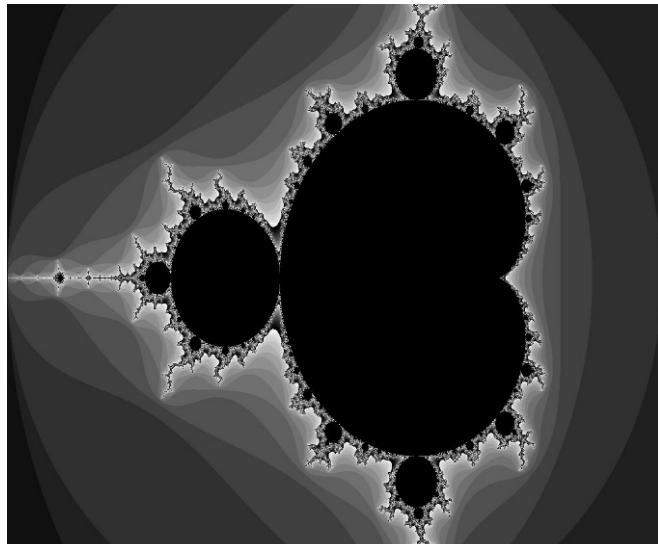
- Una primera estrategia consistiría en dividir por filas consecutivas los pixeles de la imagen en función del número de procesos.
- PROBLEMA. Puede que la carga de trabajo entre procesos no esté equilibrada: En la imagen, el proceso 2 tendría mucho más trabajo que los otros.



# Introducción a los fractales: el conjunto de Mandelbrot

## Paralelización del algoritmo

- Otra estrategia se basaría en la idea de establecer un *pool* de procesos.
- La idea radica en definir una serie de tareas (muchas más que procesos) e **ir asignando tareas a los procesos conforme quedan inactivos**.
- En nuestro caso, podríamos definir una **tarea como el procesado de una fila de píxeles de la imagen**. Cada una de estas filas se iría asignando a cada proceso en el momento que quedan inactivos.



Proceso 1

Proceso 2

Proceso 3

# Introducción a los fractales: el conjunto de Mandelbrot

## ¿CÓMO TRABAJAR CON UN POOL DE PROCESOS CON MPI?

```

int main(int argc, char **argv) {
    int myrank, ntasks, rank;
#define ETIQUETA_CONTINUAR 1
#define ETIQUETA_TERMINAR 99
    MPI_Status status;
    MPI_Init(&argc, &argv); MPI_Comm_rank(MPI_COMM_WORLD, &myrank); MPI_Comm_size(MPI_COMM_WORLD, &ntasks);
    if (myrank == 0) {
        for (rank = 1; rank < ntasks; ++rank) { // Inicializar el trabajo de los hijos. Se le manda una unidad de trabajo a cada hijo
            work = obtener_nuevo_trabajo(); // Se obtiene un nuevo trabajo para un hijo
            MPI_Send(&work, 1, MPI_INT, rank, ETIQUETA_CONTINUAR, MPI_COMM_WORLD); } // Se manda el trabajo al hijo
        work = obtener_nuevo_trabajo(); // Vamos obteniendo nuevos trabajos y asignándolos a los hijos hasta que se agoten
        while (work != NULL) {
            // Recibimos un resultado de un hijo cualquiera (MPI_ANY_SOURCE) y le mandamos trabajo a ese mismo hijo
            // identificandolo por "status.MPI_SOURCE"
            MPI_Recv(&result, 1, MPI_DOUBLE, MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &status);
            MPI_Send(&work, 1, MPI_INT, status.MPI_SOURCE, ETIQUETA_CONTINUAR, MPI_COMM_WORLD);
            work = obtener_nuevo_trabajo(); }

        // Si no hay mas trabajos que mandar a los hijos, recogemos los resultados que queden del total de hijos
        for (rank = 1; rank < ntasks; ++rank)
            MPI_Recv(&result, 1, MPI_DOUBLE, MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &status);
        // Indicamos a los hijos que deben dejar de trabajar mediante el envio de la etiqueta "ETIQUETA TERMINAR"
        for (rank = 1; rank < ntasks; ++rank)
            MPI_Send(0, 0, MPI_INT, rank, ETIQUETA_TERMINAR, MPI_COMM_WORLD);
    }
}

```

# Introducción a los fractales: el conjunto de Mandelbrot

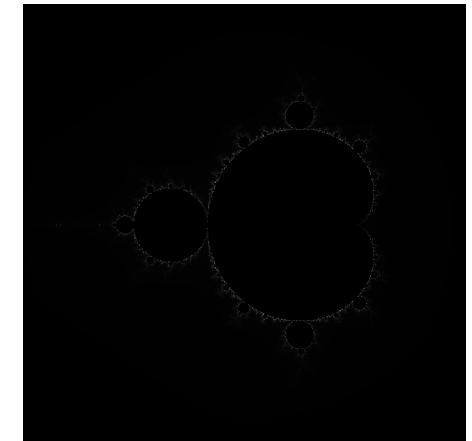
## ¿CÓMO TRABAJAR CON UN POOL DE PROCESOS CON MPI?

```
else {  
    while (1) {  
        // Recibir asignacion de trabajo del padre  
        MPI_Recv(&work, 1, MPI_INT, 0, MPI_ANY_TAG, MPI_COMM_WORLD, &status);  
        // Comprobar si la etiqueta de envio me indica que debo parar  
        if (status.MPI_TAG == ETIQUETA_TERMINAR) {  
            MPI_Finalize();  
            return 0; }  
        result = realizar_trabajo_asignado(work);  
        // Enviar resultados al padre  
        MPI_Send(&result, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD); }  
    }  
    MPI_Finalize();  
    return 0;  
}
```

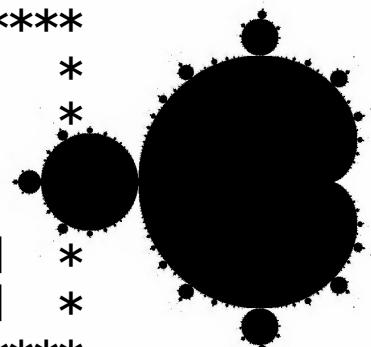
## Introducción a los fractales: Código secuencial mandelbrot.c

**./mandelbrot**

```
***** POSIBLES PARÁMETROS *****
* Ancho imagen: pixelXmax
* Alto imagen: pixelYmax
* Número de iteraciones: IterMax
* Nombre archivo imagen de salida: ArchivoImagen
* Intervalo a considerar: dominio
*****
```

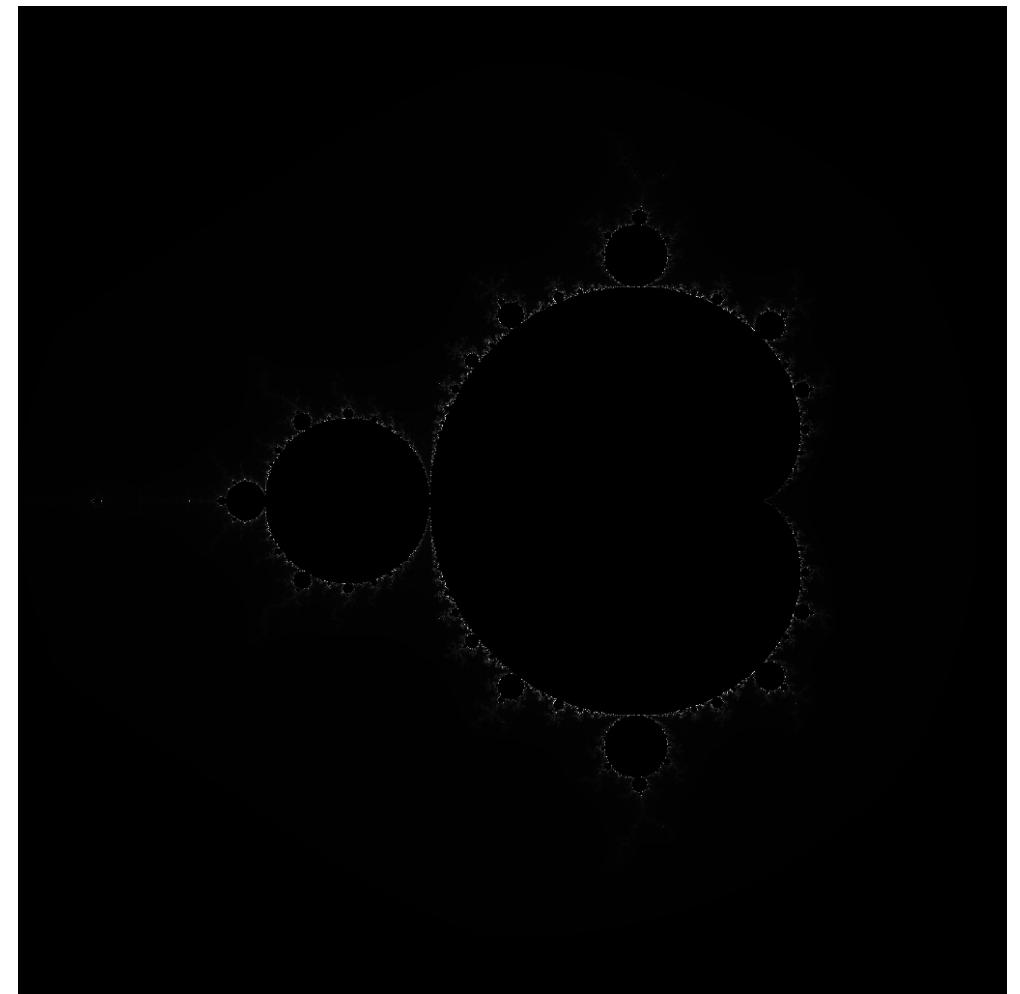
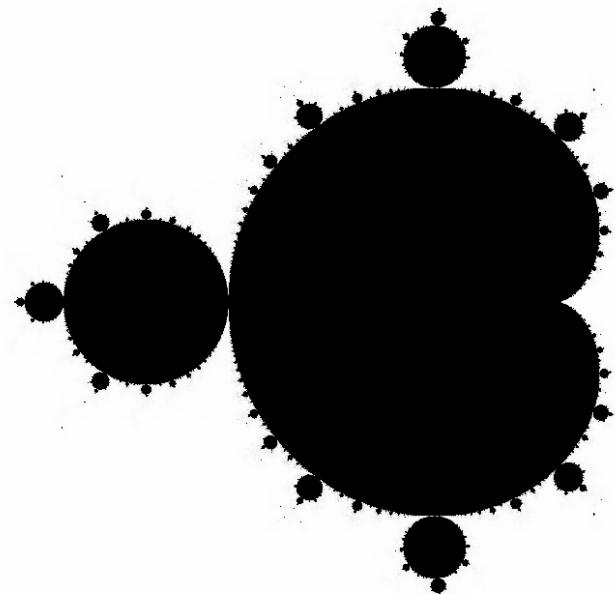


```
***** DATOS DE LA EJECUCIÓN *****
* pixelXmax = 1024, pixelYmax = 1024
* IterMax = 1000
* ArchivoImagen = imgA.pgm imgB.pgm
* Dominio = 0
* Intervalo para X: [ -2.0000000000000000, 1.0000000000000000 ]
* Intervalo para Y: [ -1.5000000000000000, 1.5000000000000000 ]
*****
```



Tiempo: 3.343509 segundos  
 Mínimo = 0.000000, Máximo = 253980.000000  
 Mínimo = 0.000000, Máximo = 254.000000

## **Introducción a los fractales: Código secuencial mandelbrot.c**

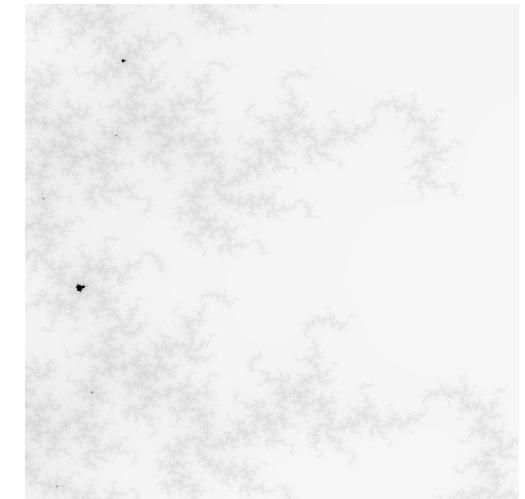


## Introducción a los fractales: Código secuencial mandelbrot.c

```
./mandelbrot 2048 2048 1000 A2 2
```

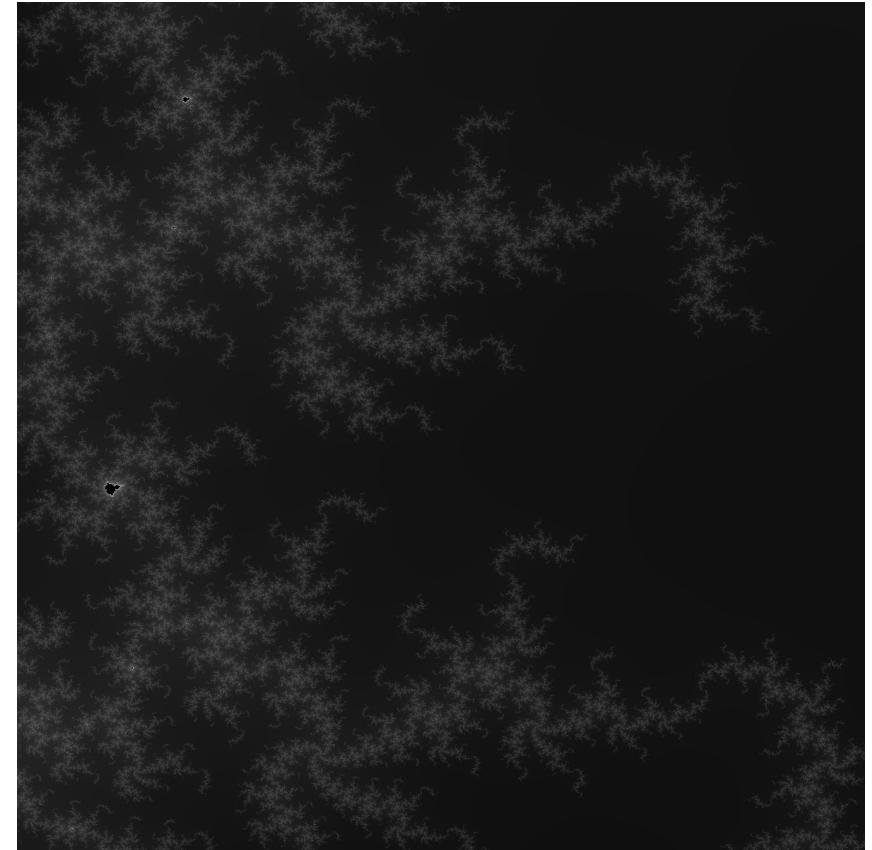
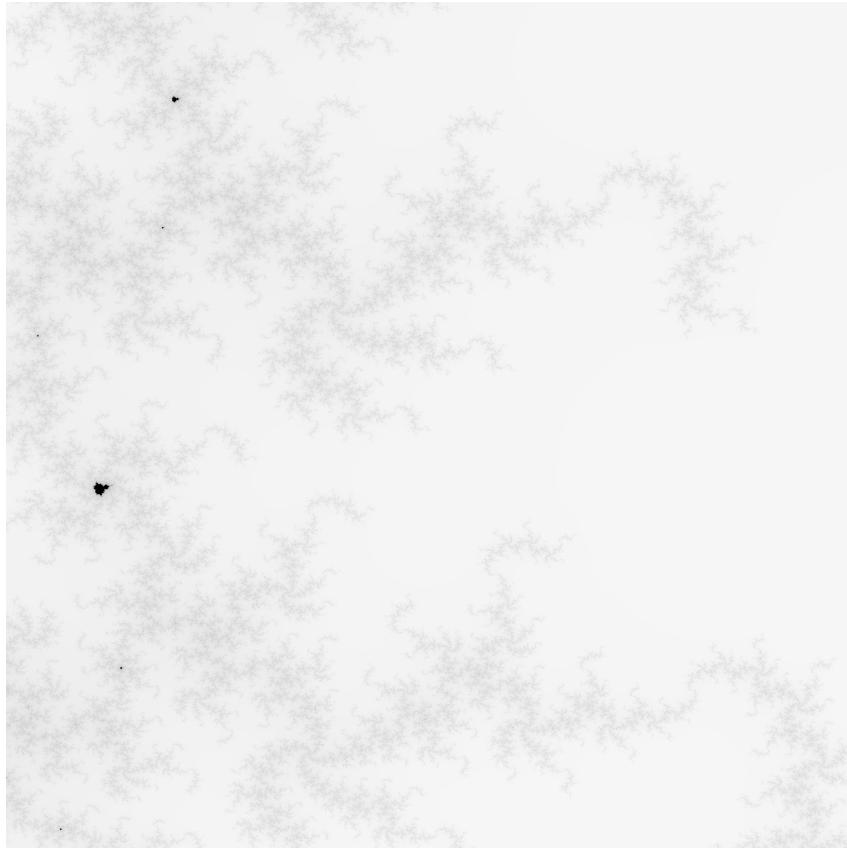
```
***** POSIBLES PARÁMETROS *****
* Ancho imagen: pixelXmax *
* Alto imagen: pixelYmax *
* Numero de iteraciones: IterMax *
* Nombre archivo imagen de salida: ArchivoImagen *
* Intervalo a considerar: dominio *
*****
```

```
***** DATOS DE LA EJECUCIÓN *****
* pixelXmax = 2048, pixelYmax = 2048 *
* IterMax = 1000 *
* ArchivoImagen =           A2A.pgm          A2B.pgm *
* Dominio = 2 *
* Intervalo para X: [ -1.017579000000000, -1.01696800000000 ] *
* Intervalo para Y: [ -0.274444000000000, -0.27375800000000 ] *
*****
```



Tiempo: 8.331326 segundos  
Minimo = 0.000000, Maximo = 254235.000000  
Minimo = 0.000000, Maximo = 255.000000

## **Introducción a los fractales: Código secuencial mandelbrot.c**



## Introducción a los fractales: Código secuencial mandelbrot.c

```
./mandelbrot 2048 2048 1000 A3 3
```

\*\*\*\*\* POSIBLES PARÁMETROS \*\*\*\*\*

- \* Ancho imagen: pixelXmax \*
- \* Alto imagen: pixelYmax \*
- \* Número de iteraciones: IterMax \*
- \* Nombre archivo imagen de salida: ArchivoImagen \*
- \* Intervalo a considerar: dominio \*

\*\*\*\*\*

\*\*\*\*\* DATOS DE LA EJECUCIÓN \*\*\*\*\*

- \* pixelXmax = 2048, pixelYmax = 2048 \*
- \* IterMax = 1000 \*
- \* ArchivoImagen = A3A.pgm A3B.pgm \*
- \* Dominio = 3 \*
- \* Intervalo para X: [ -1.017523000000000, -1.017493000000000 ] \*
- \* Intervalo para Y: [ -0.274065000000000, -0.274032000000000 ] \*

Tiempo: 22.784786 segundos

Minimo = 0.000000, Maximo = 255000.000000

Minimo = 0.000000, Maximo = 255.000000

## **Introducción a los fractales: Código secuencial mandelbrot.c**

