

Els fractals

La geometria del món

Fabio Guevara

Ins Francesc Macià

Barcelona 2024



Contents

- 1 Objectius
- 2 Metodologia
- 3 Què son aquestes formes?
- 4 Origen del concepte
- 5 Característiques
- 6 Formes
- 7 Presència al món
- 8 Pràctica

Objectius

Objectius

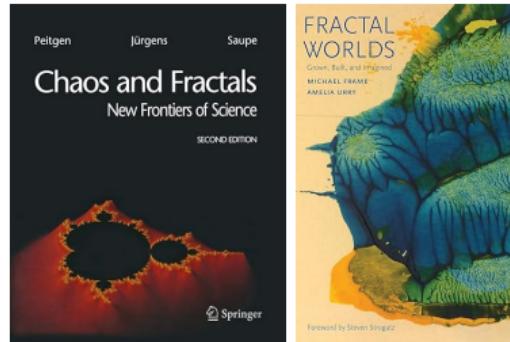
- A nivell teòric conèixer sobre el concepte de fractal i les teories matemàtiques que engloben.
- A nivell pràctic es buscava respondre les següents qüestions:
 - Es possible modelar en 3D un paisatge natural a partir d'un sol fractal?
 - Es podran obtenir més d'un tipus de paisatge?
- Aquestes qüestions es respondran o no, més endavant amb la part pràctica.

Metodología

Metodologia

Recerca i investigació sobre els fractals i tots els conceptes que engloben. Ús de diverses fonts:

- Llibres de divulgació
- Vídeos
- Pàgines web



Metodologia

- Cerca d'un llenguatge de programació on desenvolupar la pràctica. Llenguatge de programació Python.



- Elaboració de les imatges mostrades al treball en Jupyter. D'aquesta manera es va aprenent com funcionen els codis i llibreries emprats en els paisatges posteriorment.
- Després de comprendre el funcionament de l'eina, es comença l'elaboració dels paisatges naturals en 3D.

Qué son aquestes formes?

- La paraula fractal ve del llatí *fractus*, que significa trencat, fracturat o irregular
- Els fractals es defineixen com a formes geomètriques complexes, que tenen un patró que es repeteix a diferents escales i mesures.
- Matemàticament els fractals, ademés de ser semblants a diverses escales, tenen dimensió irracional, és a dir, fraccionària.

Origen del concepte

Origen del concepte:

- El primers patrons fractals coneguts datant de l'edat antiga, on eren utilitzats amb motius decoratius.
- El concepte actual es va començar a modelar durant el segle XIX, quan diversos matemàtics van estudiar els llavors coneguts 'monstres matemàtics'. Els quals serien els primers fractals.
- Finalment, el concepte va ser definit per Mandelbrot després de revisar els estudis de Julia mentre treballava a l'IBM. Mandelbrot acabaria definir el concepte de fractal i les seves característiques (autosimilitud, dimensió fractal, etc.). També construiria el seu famós conjunt de Mandelbrot.

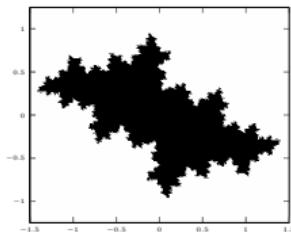
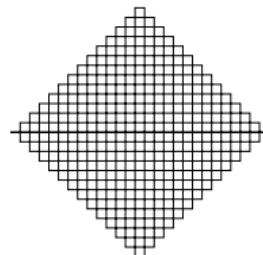
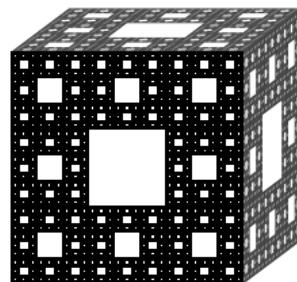
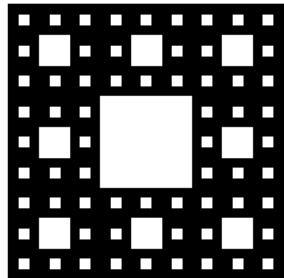
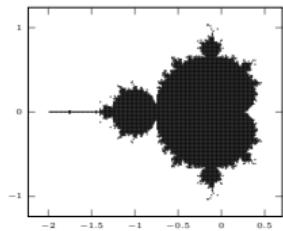
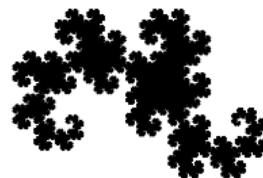
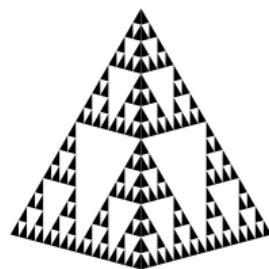
Caractéristiques

Característiques:

Els fractals presenten qualitats o característiques úniques, aquestes són:

- **Autosimilitud:** és la propietat de mantenir-se semblant a diverses escales.
 - Exacta
 - Aproximada
 - Estadística
- **Recursivitat:** es poden construir mitjançant la repetició de si mateix indefinidament.
- **Dimensió fractal:** els fractals són formes geomètriques de dimensió irracional.

Formes:



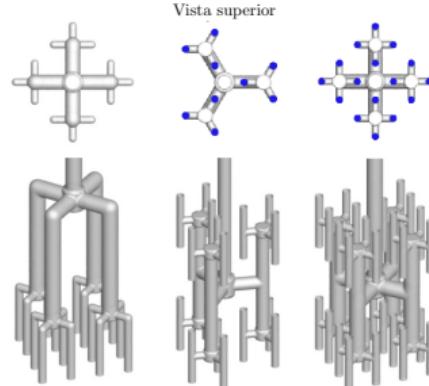
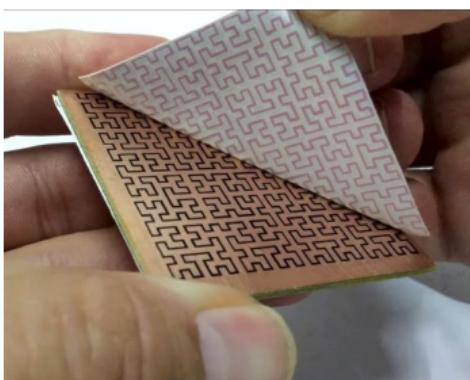
Presència al món:

Els fractals es manifesten al món de dues formes:

- A la natura:



- A la ciència i tecnologia:

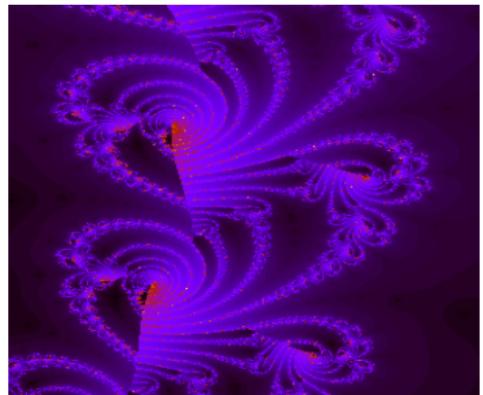
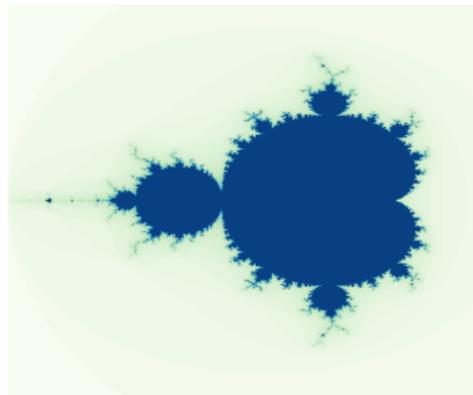
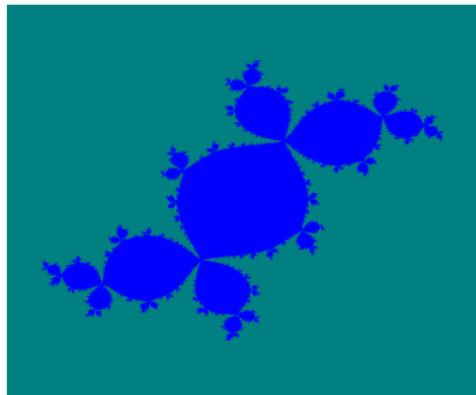


Paisatges fractals

- La part pràctica del meu TR es va basar en la programació de paisatges naturals a partir d'un sol fractal.
- Primer, vaig escollir un llenguatge de programació per fer-ho, que va ser Python. Després vaig buscar quins codis necessitaria per realitzar els paisatges.
- Abans d'elaborar els paisatges, vaig posar en pràctica tot el que havia après de programació dibuixant primer formes fractals conegudes. Les imatges obtingudes les vaig utilitzar com a exemples en el meu treball.

Paisatges fractals

Per produir els paisatges vaig utilitzar els següents fractals:



Conjunt de Julia

El conjunt de julia o fractal de julia es defineix de la següent funció:

$$f_c(z) = z^2 + c. \quad (1)$$

On utilitzem un nombre complex c concret per fer iteracions d'aquesta funció:

$$z_0 = z$$

$$z_1 = f_c(z_0) = z_0^2 + c$$

$$z_2 = f_c(z_1) = z_1^2 + c$$

$$\vdots$$

$$z_n = f_c(z_{n-1}) = z_{n-1}^2 + c$$

Conjunt de Mandelbrot

El conjunt de Mandelbrot s'obté de la següent funció:

$$f(z) = z^2 + c, \quad z, c \in \mathbb{C}. \quad (2)$$

Primer es fixa un nombre complex c qualsevol. A partir de c , es construeix una successió recursiva:

$$\begin{cases} z_0 = 0, & \text{termini inicial} \\ z_{n+1} = z_n^2 + c, & \text{successió recursiva.} \end{cases}$$

En general, els punts c pertanyen al subconjunt del pla complex $[-2, 2] \times [-2, 2]$.

Fractal de Newton

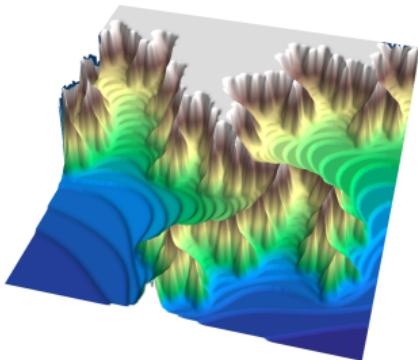
El fractal de Newton es genera a partir del mètode de **Newton-Raphson**. Aquest, troba les arrels d'una funció prenen una estimació inicial d'una arrel, x_0 , i cerca successivament millors aproximacions de la mateixa de la manera següent:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, \dots$$

En cada iteració, l'arrel s'aproxima a x_{n+1} , la intercepció en l'eix x de la tangent a la gràfica en $f(x_n)$. Quan s'aplica a funcions de variable complexa z , es poden crear fractals.

Resultats

Exposició dels paisatges obtinguts



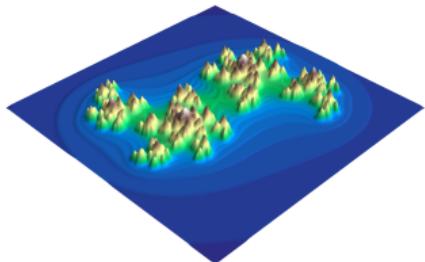
```

def mandelbrot(c,ite):
    # Usen com a punt d'inici el nombre complex z=0+0i
    z=complex(0,0)
    # Amb el següent bucle es crea una successió de nombres complexos c(0), c(1), c(2), etc. Aquest procés s'executarà
    # fins que es trobi la iteració c(k). Tal que
    # |c(k)|>2.
    for k in range(ite):
        z=z*z+c
        if abs(z)>2:
            break
    return k

# Definim una altra funció, que servirà per dibuixar regions del conjunt de Mandelbrot, utilitzant un nombre complex 'z'
# concret.
# z = nombres complexos
# dx = rang de capacitat pels valors reals, per obtenir la regió a graficar.
# dy = rang de capacitat pels valors imaginaris, per obtenir la regió a graficar.
# n = nombre de punts en la regió en els eixos x i y.
def plot_mandelbrot3D(z,dx,dy,n):
    # Construïm la regió de [real(z)-dx,real(z)+dx]X[imag(z)-dy,imag(z)+dy].
    x=np.linspace(np.real(z)-dx,np.real(z)+dx,n)
    y=np.linspace(np.imag(z)-dy,np.imag(z)+dy,n)
    # Fem una matriu de zeros de dimensió nxn.
    a=np.zeros((n,n))
    # Amb dos bucles for, construïm una malla de nombres complexos 'c' amb els valors 'x' i 'y'.
    for k in range(n):
        for j in range(n):
            c=complex(x[k],y[j])
            # Actualitzem la matriu, substituint cada zero pel valor de 'k' que retorna la funció mandelbrot().
            # c = tots el nombres complexos.
            a[k,j]=mandelbrot(c,40)
            # La matriu s'actualitzarà amb els valors de 'a' obtinguts en cada nova iteració.
    mayavi.mlab.figure(bgcolor=(1, 1, 1))
    smoothed_atlas = scipy.ndimage.gaussian_filter(a.T, 2)
    mayavi.mlab.surf(smoothed_atlas, warp_scale=5.5,colormap = 'terrain')
    mayavi.mlab.show()

```

Resultats



```

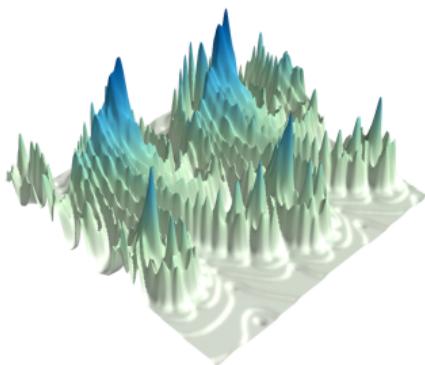
def julia(c,z,ite):
    for k in range(ite):
        z=z*z+c
        if abs(z)>4:
            break
        pass
    pass
    return k

# Valors de la malla.
x_min = -1.5
x_max = 1.5
y_min = -1.5
y_max = 1.5

# A continuació definim una altra funció, la qual, anomenarem plot_julia3D(). Aquesta funció ens servirà per crear el paisatge 3D.
def plot_julia3D():
    # Construïm dos eixos de [x_min, x_max]X[y_min, y_max]. També farem una matriu de zeros de dimensió nxn.
    x=np.linspace(x_min,x_max,n)
    y=np.linspace(y_min,y_max,n)
    a=np.zeros((n,n))
    # Amb dos bucles for, construïm una malla de nombres complexos 'z' amb els valors 'x' i 'y'.
    for i in range(n):
        for j in range(n):
            z=complex(x[i],y[j])
            a[i,j]=julia(c,z,40)
            #print(a,c)
            pass
    pass
mayavi.mlab.figure(bgcolor=(1, 1, 1))
mayavi.mlab.surf(a.T, warp_scale=2.5,colormap = 'YlGn')
smoothed_atlas = scipy.ndimage.gaussian_filter(a.T, 2)
mayavi.mlab.surf(smoothed_atlas, warp_scale=2.5,colormap = 'terrain')
mayavi.mlab.show()

```

Resultats



```

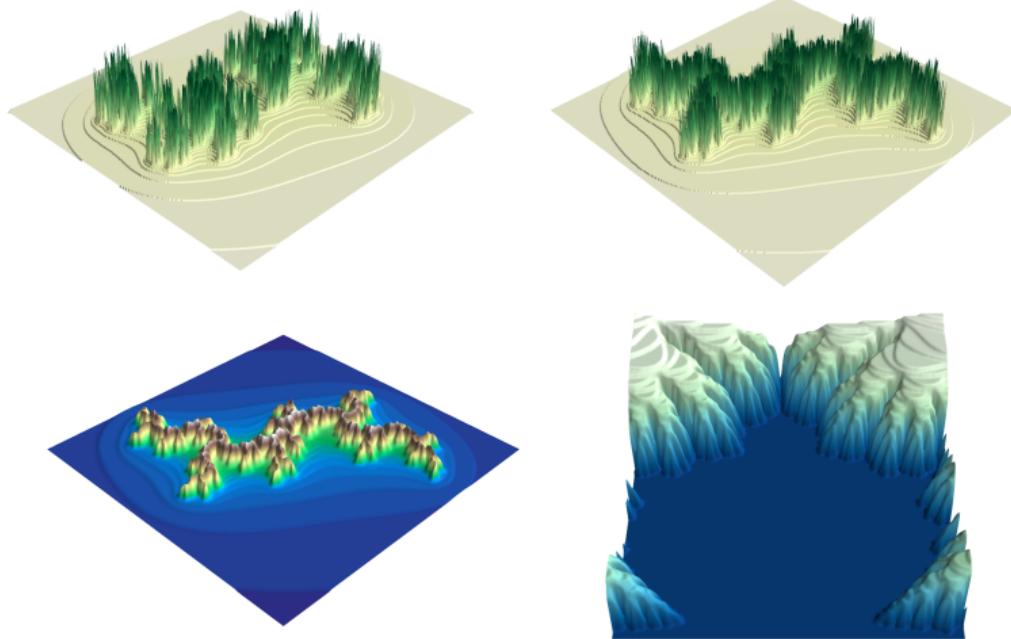
def newton_fractal(f,df,z0,ite):
    # tol = La tolerància: un petit valor que serveix com a valor límit, és a dir, només agafem els valors que siguin
    # majors a la tolerància.
    tol = 1e-8
    z=z0
    for k in range(ite):
        dz = f(z)/df(z)
        if abs(dz)<tol:
            return k
        z = z - 1.25*dz

f=lambda z : z***(4 + 3j) - 1
df=lambda z : (4 + 3j)*z***(3 + 3j)
x_min = 0.4
x_max = 0.9
y_min = -0.9
y_max = -0.3

n_space = 300
x_values=np.linspace(x_min,x_max,n_space)
y_values=np.linspace(y_min,y_max,n_space)
n=len(x_values)
m=len(y_values)
a=np.zeros((n,m))
for k in range(n):
    for j in range(m):
        z=complex(x_values[k],y_values[j])
        # Actualitzem la matriu, substituint cada zero pel valor del mòdul de 'z' que retorna la funció newton_fractal
        #().
        # z = tots el nombres complexos.
        a[k,j]=newton_fractal(f,df,z,200)
        # La matriu s'actualitzarà amb els valors de 'a' obtinguts en cada nova iteració.
        pass
    pass
mayavi.mlab.figure(bgcolor=(1, 1, 1))
smoothed_atlas = scipy.ndimage.gaussian_filter(a.T, 1.5)
mayavi.mlab.surf(smoothed_atlas, warp_scale=2.5,colormap = 'GnBu')
mayavi.mlab.show()

```

Resultats



Conclusions:

- S'ha obtingut un ampli coneixement sobre els fractals.
- S'ha après a utilitzar un llenguatge de programació (Python).
- S'ha aconseguit generar paisatges a partir d'un sol fractal.
- S'ha aconseguit generar més d'un sol tipus de paisatge.