

*“Clouds are not spheres, mountains are not cones,
coastlines are not circles, and bark is not smooth,
nor does lightning travel in a straight line.”*

Benoit Mandelbrot

Fraktálok és káosz

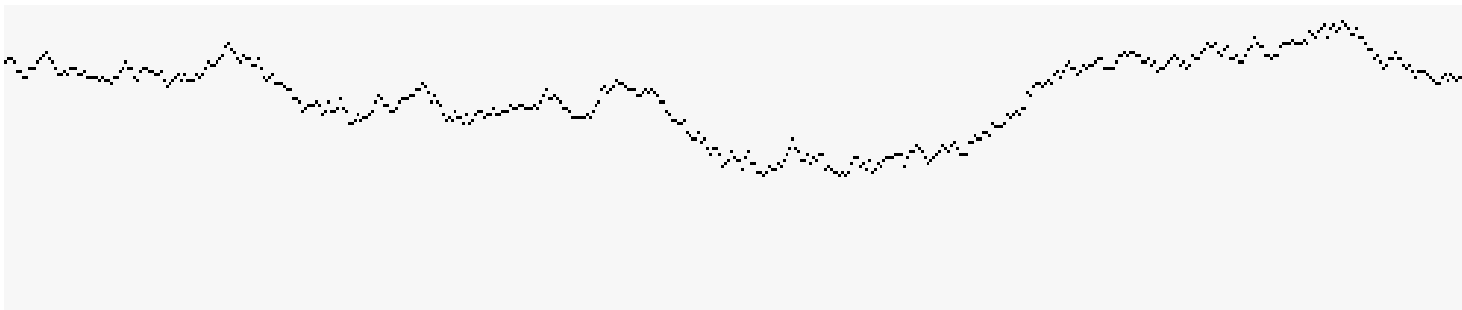
1. Fraktális dimenzió

Szirmay-Kalos László



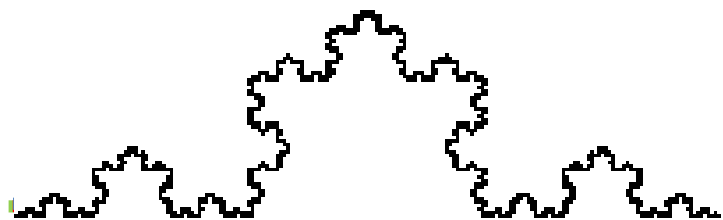
A valóság (természet) metrikus?

- **Idáig a virtuális világ = euklideszi, gömbi, hiperbolikus**
 - „Sima” egyenesre/síkra épít
 - Kicsiben mindenki lineáris: **Skálafüggőség**, differenciálható
 - Méret = lefedés: Hossz (1D) = szakasz; felület (2D) = négyzet; térfogat (3D) = kocka
 - Dimenzió: koordináták minimális száma; görbe (1D): $\mathbf{r}(t)$; felület (2D): $\mathbf{r}(u, v)$;
milyen méret értelmes?



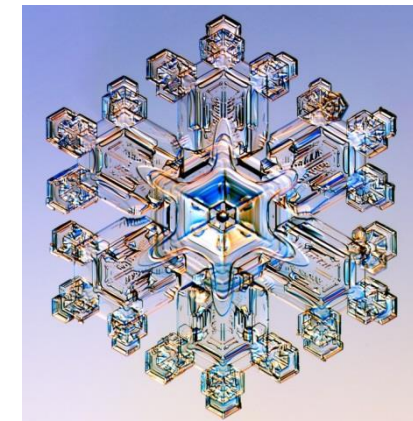
- **Természet**
 - **Skálafüggetlen**, azaz közelről olyan, mint messziről
 - Nem lesz kicsiben sem lineáris
 - Nem differenciálható
 - Szakasz, négyzet, kocka lefedés nem működik (dimenzió?)

(Helge von) Koch görbe: hossz végtelen



$$l_n = l_0 \left(\frac{4}{3}\right)^n \rightarrow \infty$$

- Véges tartományban végtelen hosszú: Dimenzió > 1
- Területe zérus: Dimenzió < 2
- Folytonos
- Sehol sem differenciálható (tüskés)
- Önhasonló





(Felix) Hausdorff dimenzió önhasznó objektumokra

$r = 1/2$ kicsinyítés

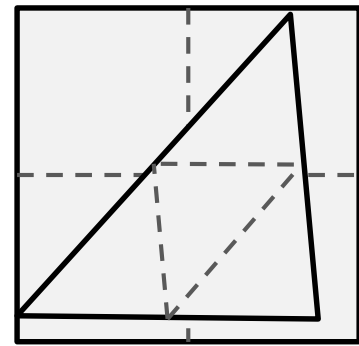
$N = 1$



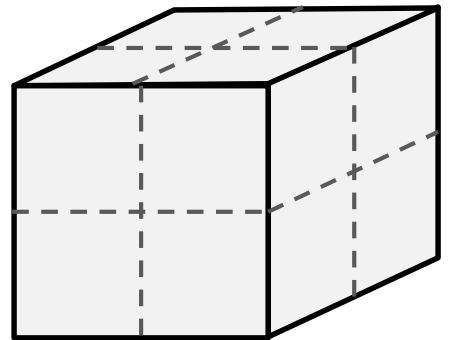
$N = 2$



$N = 4$



$N = 8$

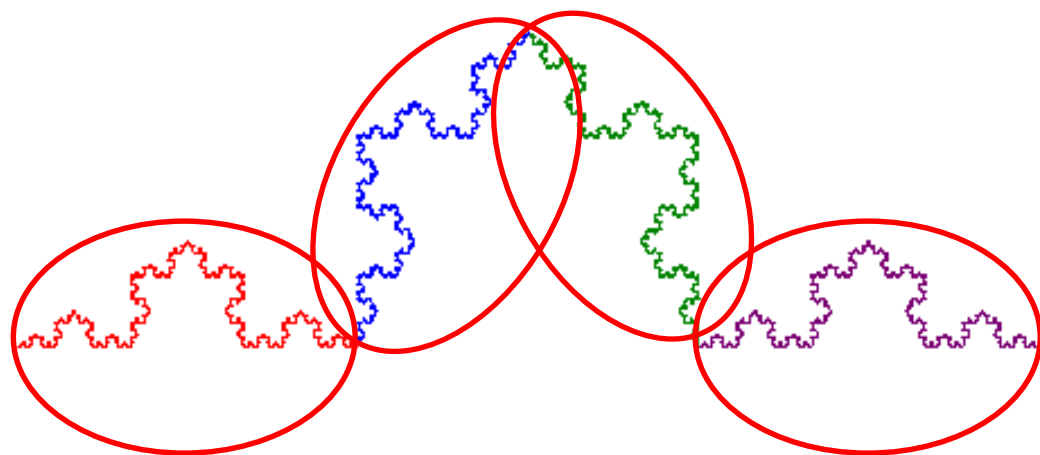


$N = 1/r^D$



$D = \frac{\log(N)}{\log(1/r)}$

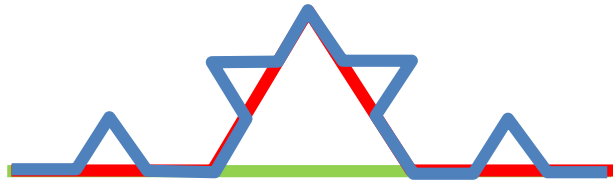
Koch görbe Hausdorff dimenziója



$$r = 1/3$$
$$N = 4$$

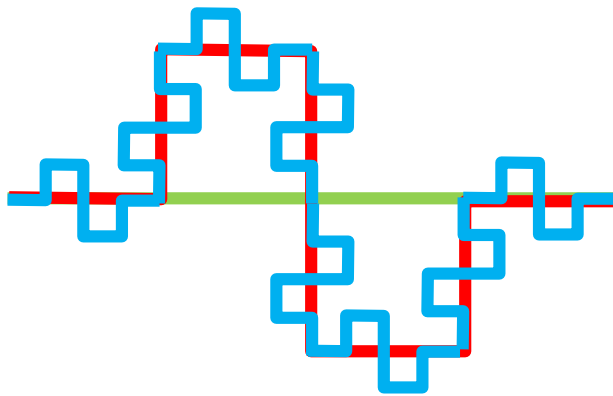
$$D = \frac{\log(4)}{\log(3)} \approx 1.26$$

Lindenmayer (Arisztid) rendszerek



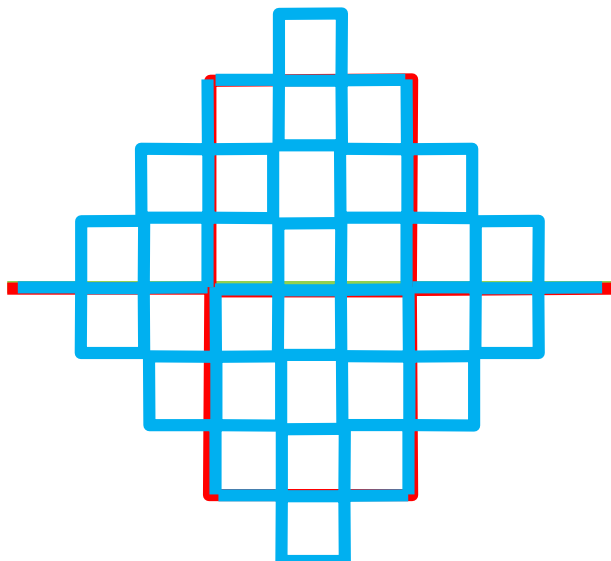
$$F \rightarrow F + 60^\circ F - 120^\circ F + 60^\circ F$$

$$r = 1/3, N = 4, D = \log(4)/\log(3) = 1.26$$



$$F \rightarrow F + 90^\circ F - 90^\circ F - 90^\circ F + 90^\circ F + 90^\circ F - 90^\circ F$$

$$r = 1/4, N = 8, D = \log(8)/\log(4) = 1.5$$

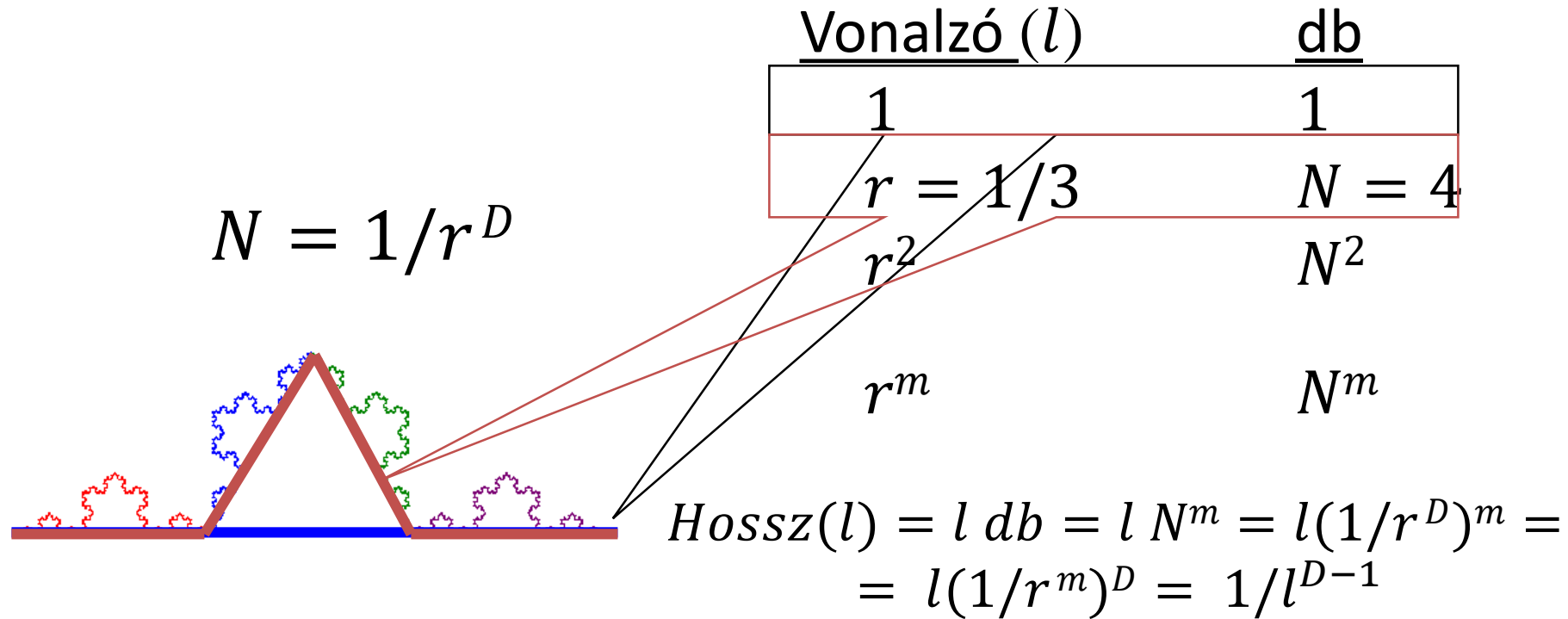


$$F \rightarrow F + 90^\circ F - 90^\circ F - 90^\circ F - 90^\circ F + 90^\circ F + 90^\circ F + 90^\circ F - 90^\circ F$$

$$r = 1/3, N = 9, D = \log(9)/\log(3) = 2$$

Peano/Hilbert térkitöltő görbe

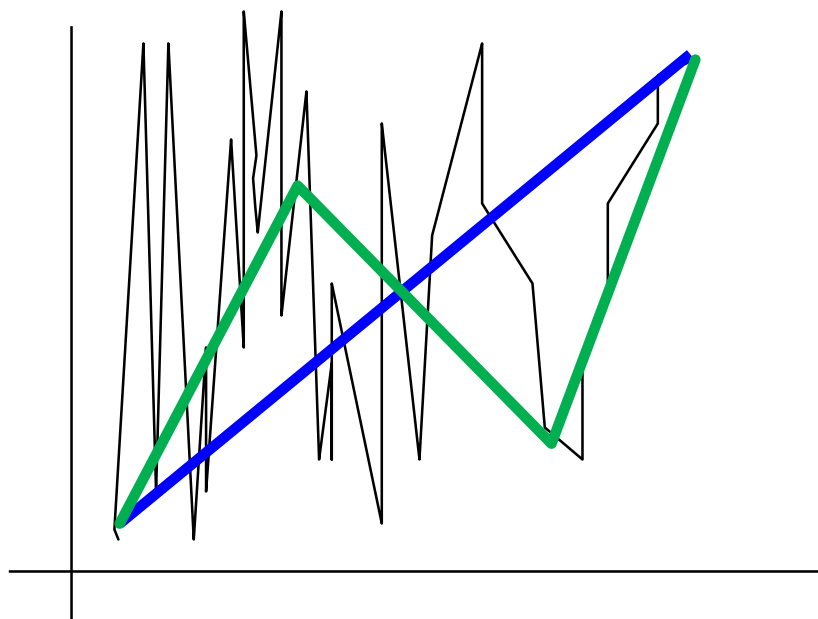
Nem önhasznó objektumok: vonalzó dimenzió



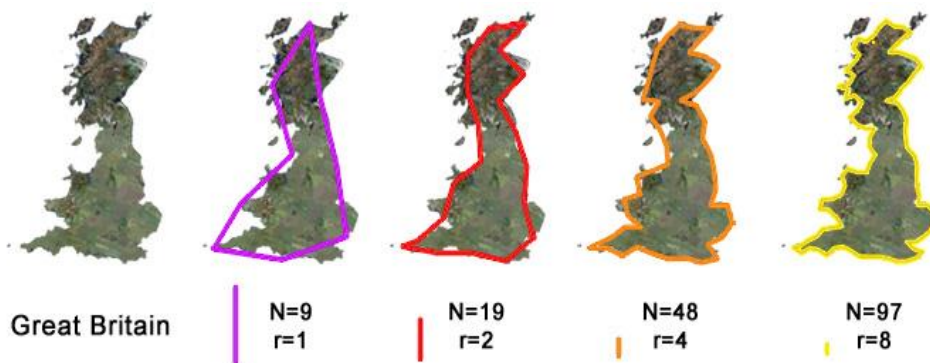
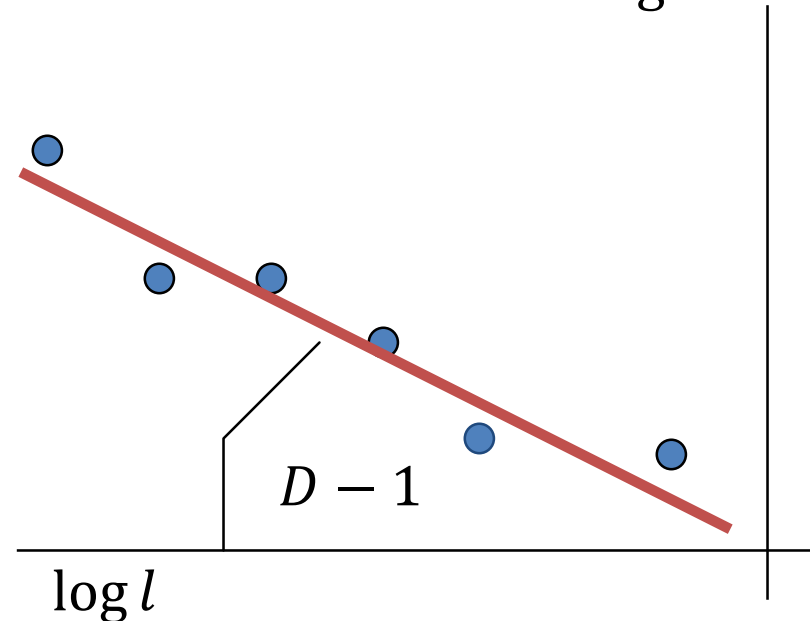
$$D = -\log(Hossz(l))/\log(l) + 1$$

Dimenziómérés = hossz mérés

EEG görbe



$\log Hossz(l)$

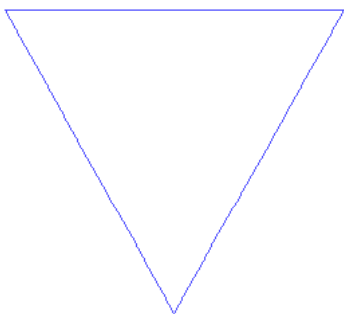


Alkalmazás:

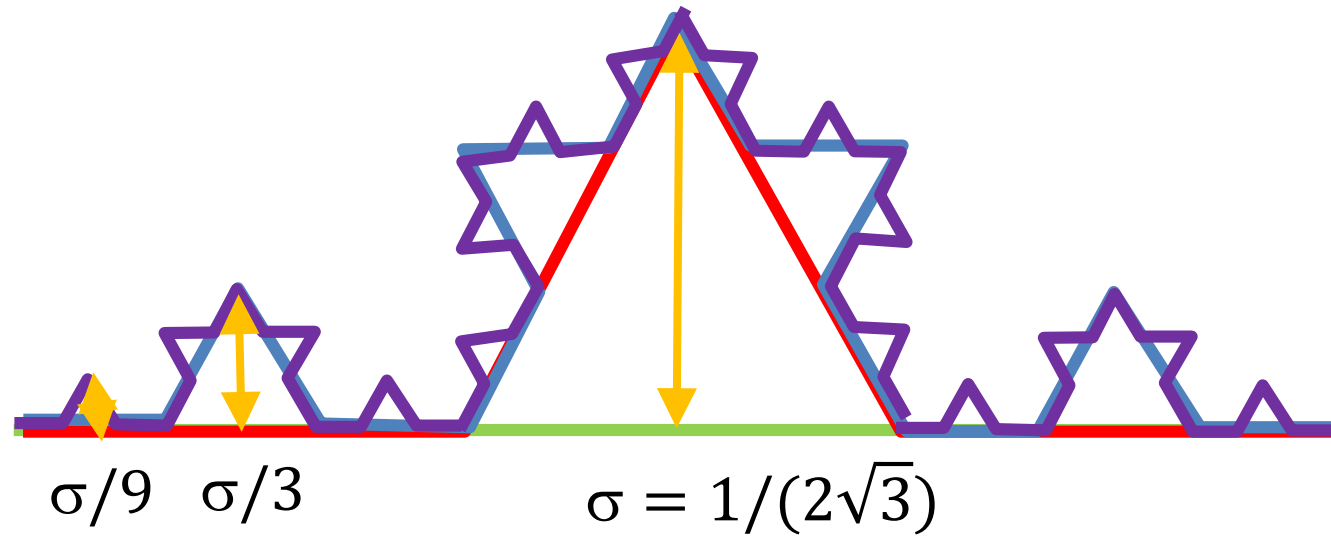
Természetes objektumok
elkülönítése és kategorizálása

Lindenmayer rendszerek

$F \rightarrow F+60F-120F+60F$



Fraktális zaj



Skála

$$N = 4$$

$$N^2 = 16$$

$$N^m = 4^m$$

Részlet

$$\sigma$$

$$\sigma/3$$

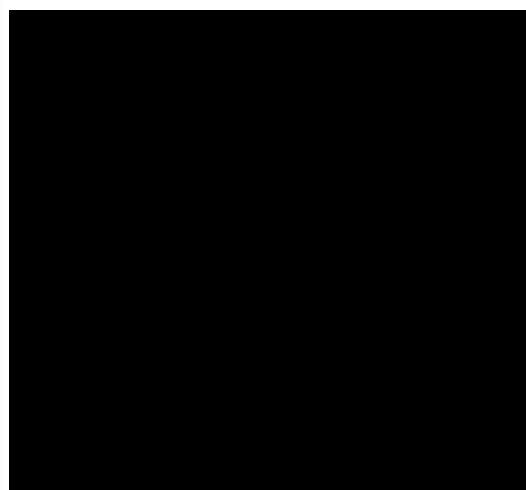
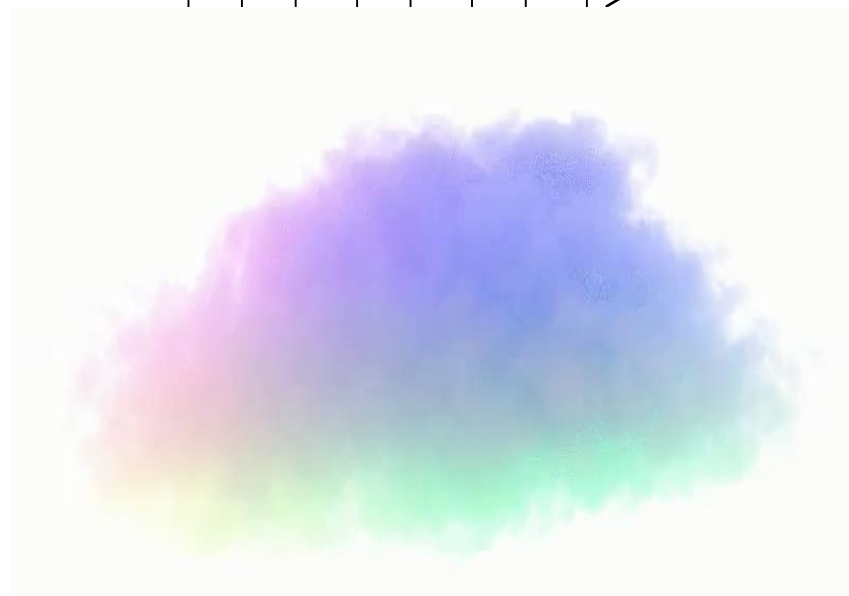
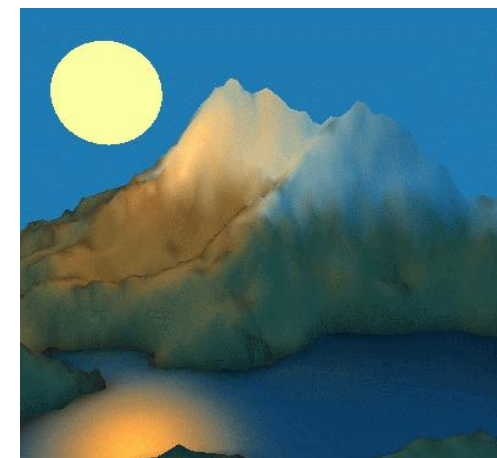
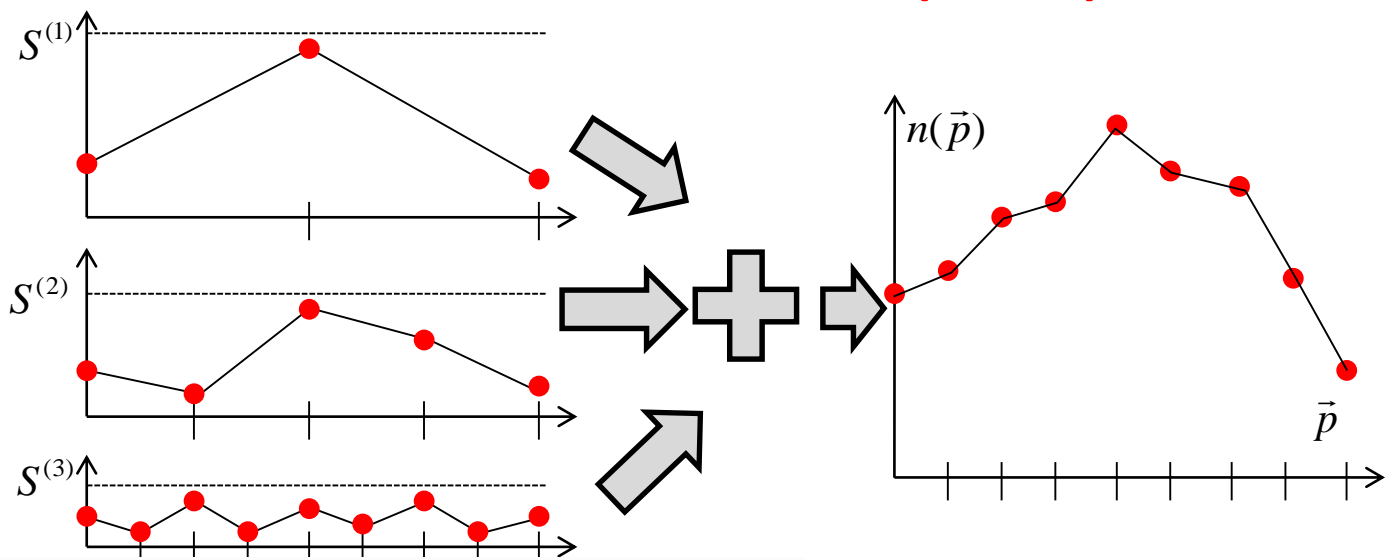
...

$$\sigma/3^{m-1}$$



szórás

(Ken) Perlin zaj



*“Invention, it must be humbly admitted,
does not consist in creating out of void
but out of chaos.”*

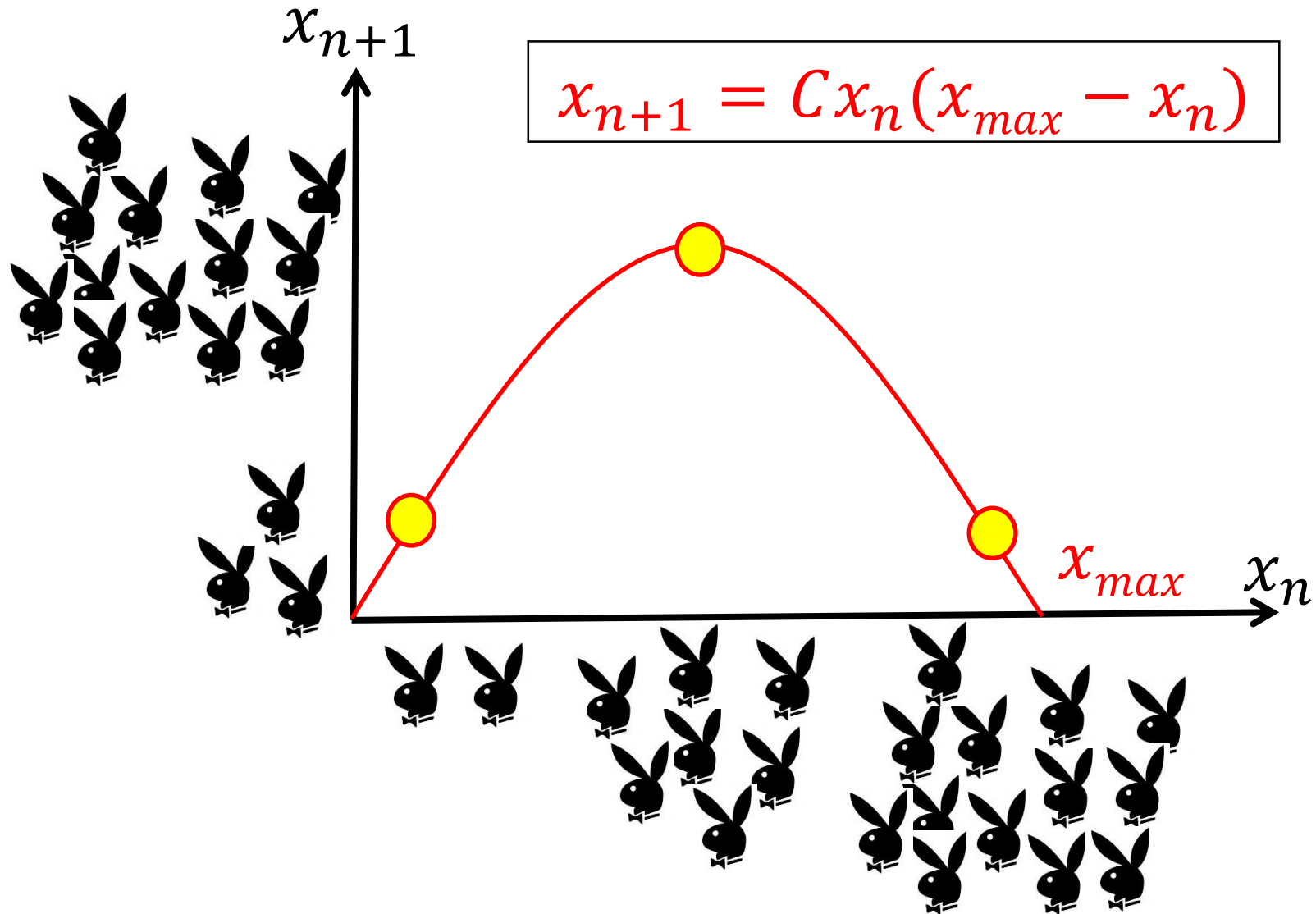
Mary Shelly

Fraktálok és káosz

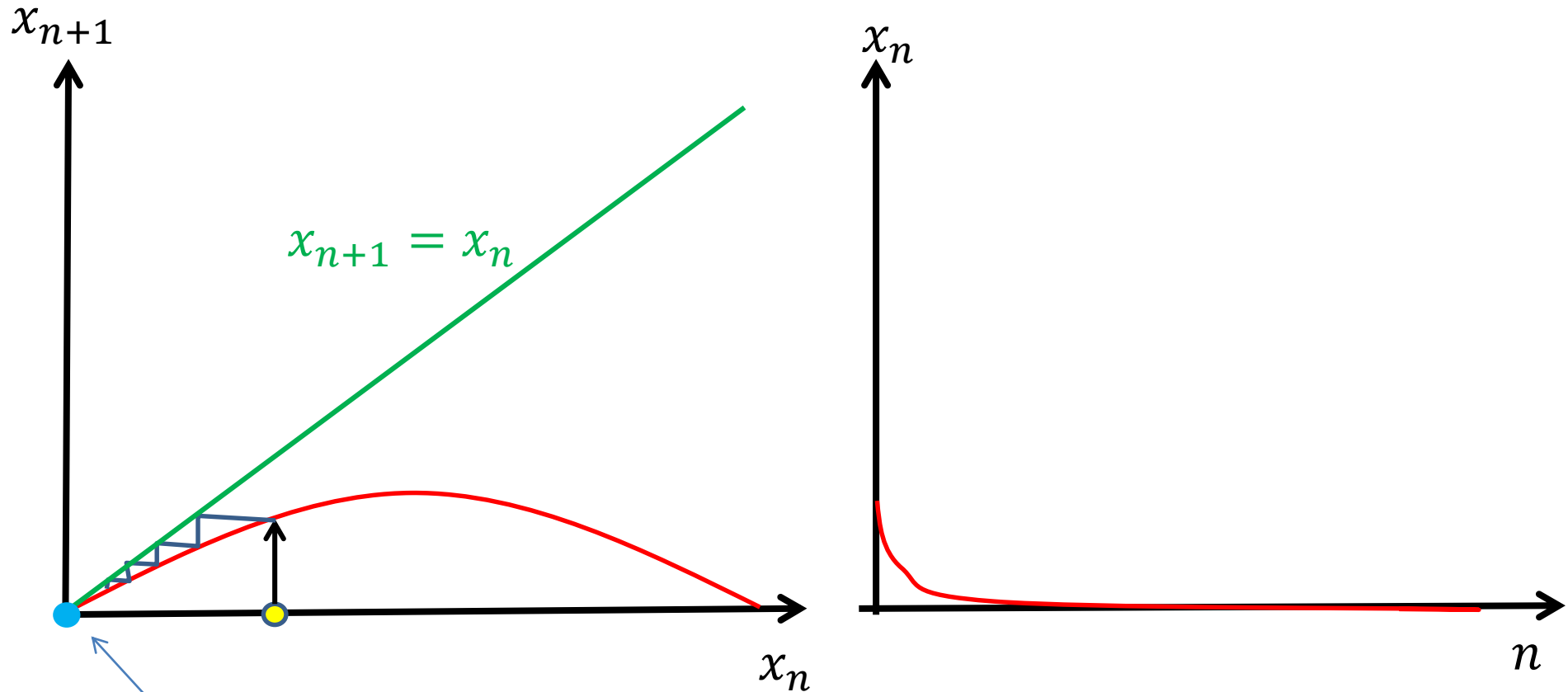
2. Káosz

Szirmay-Kalos László

Nyulak szigete



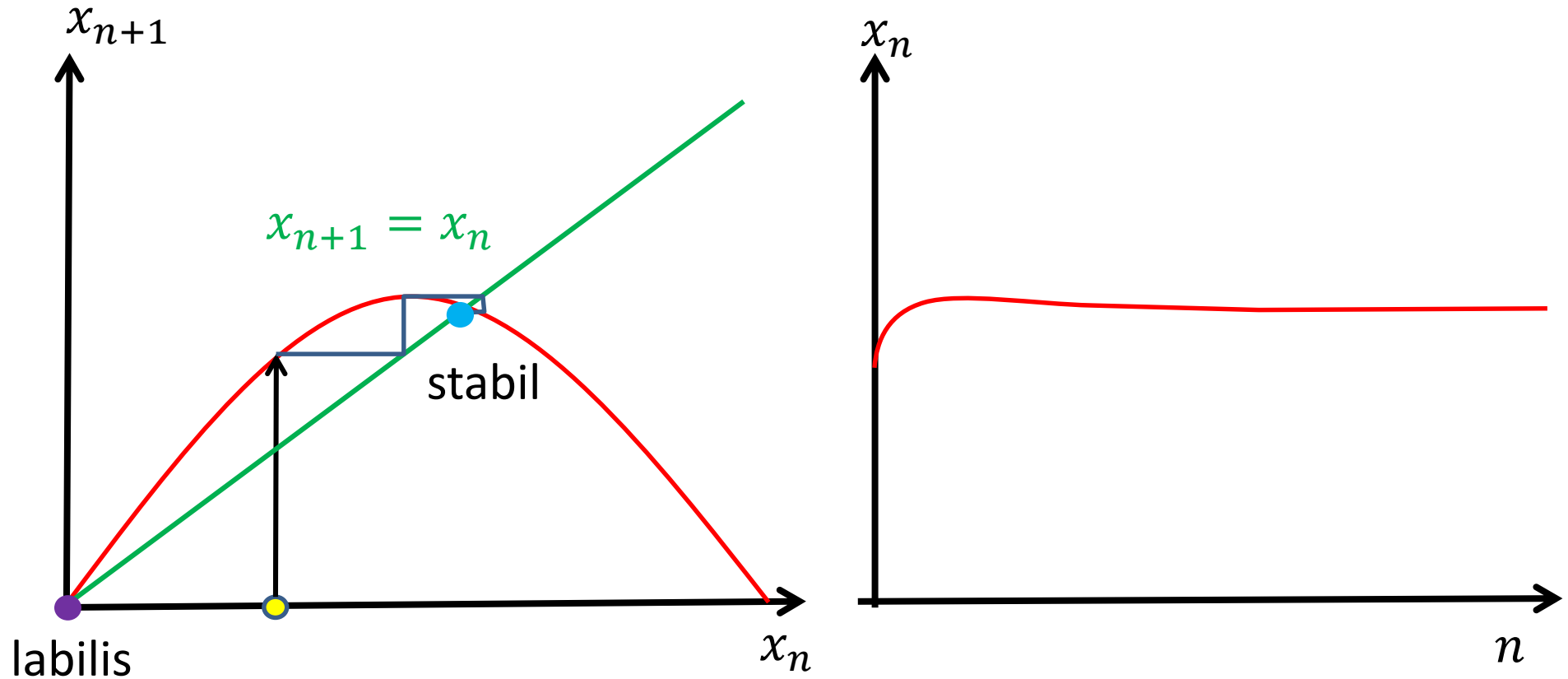
Nyulak kis C értékre: $x_{n+1} = Cx_n(x_{max} - x_n)$



Fix pont: $x^* = F(x^*) = Cx^*(x_{max} - x^*) \Rightarrow x^* = 0$

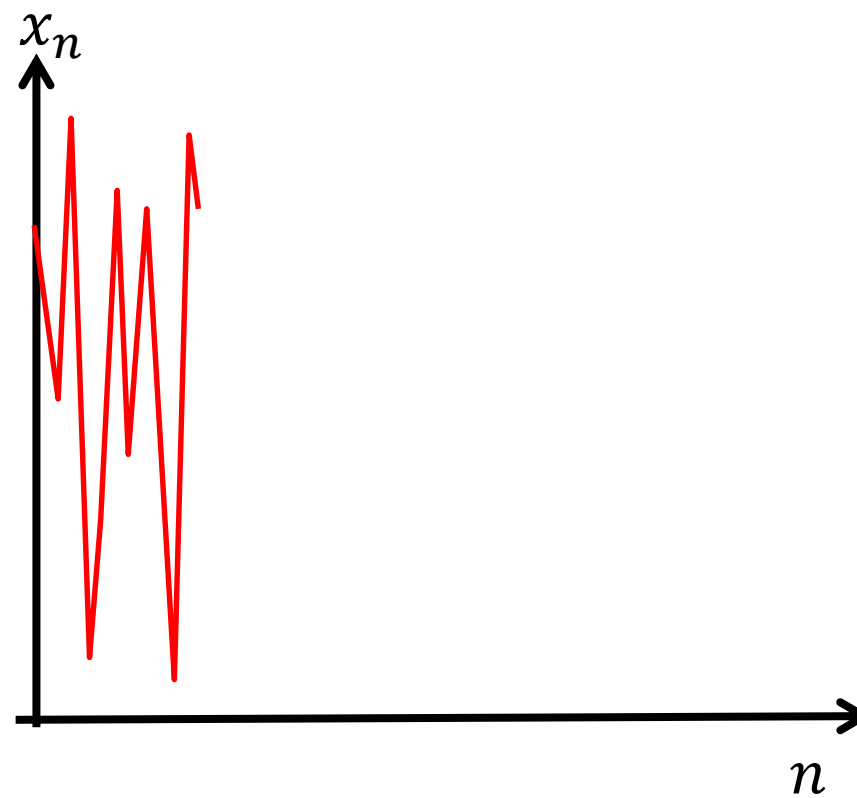
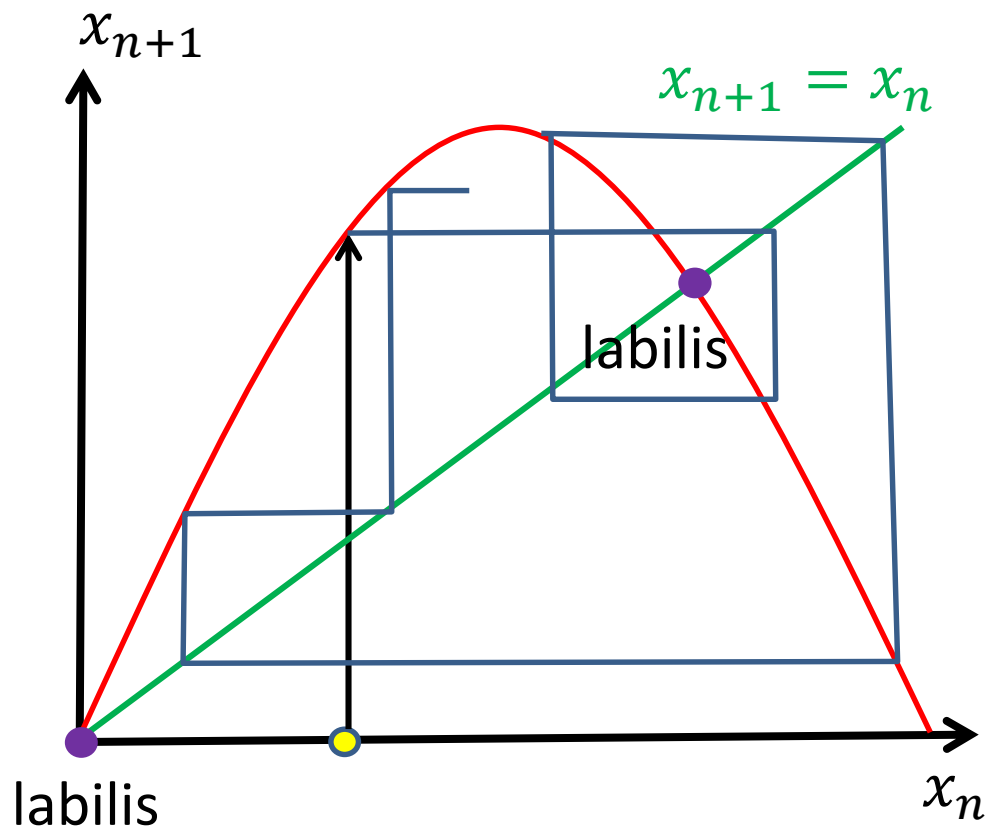
Stabil: $|F'(x^*)| = |C(x_{max} - 2x^*)| < 1 \Rightarrow C < 1/x_{max}$

Közepes C értékre: $x_{n+1} = Cx_n(x_{max} - x_n)$

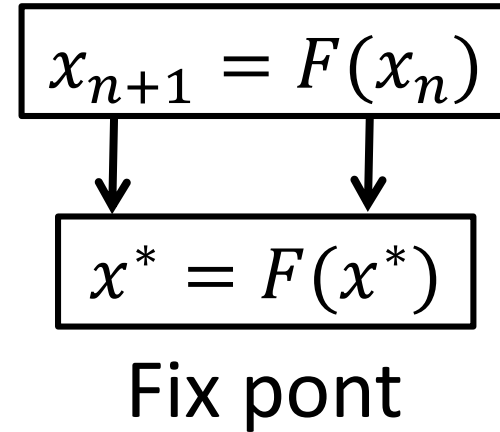
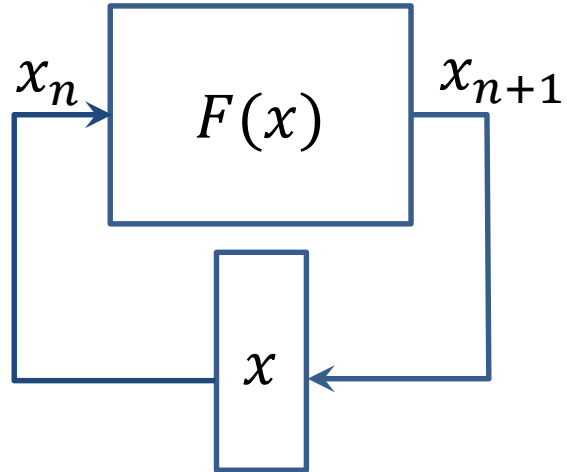


Fix pont: $x^* = F(x^*) = Cx^*(x_{max} - x^*) \Rightarrow x^* = x_{max} - 1/C$
Stabil: $|F'(x^*)| = |C(x_{max} - 2x^*)| < 1 \Rightarrow C < 3/x_{max}$

Nagy C értékre: $x_{n+1} = Cx_n(x_{max} - x_n)$



Iterált függvények, fix pont



Viselkedés (trajektória):

- $F(x)$
- x_0

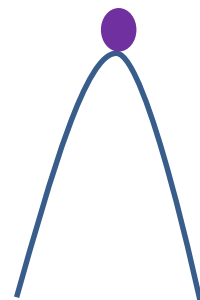
$$\begin{aligned} x_1 &= F(x_0) \\ x_2 &= F(x_1) = F(F(x_0)) \\ x_3 &= F(x_2) = F(F(F(x_0))) \\ &\vdots \end{aligned}$$

Iterált függvények, fix pont stabilitás

$$x_{n+1} = F(x_n)$$

$$x^* = F(x^*)$$

Fix pont



Labilis

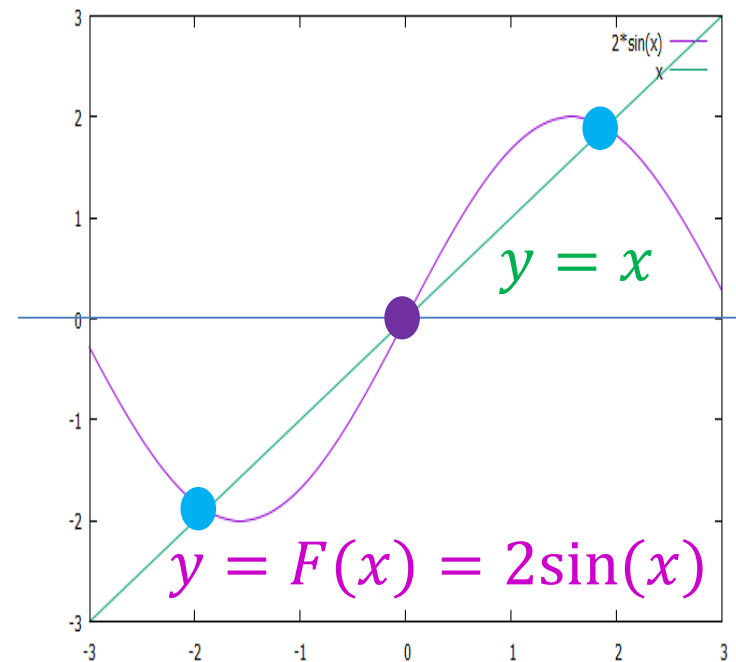


Stabil

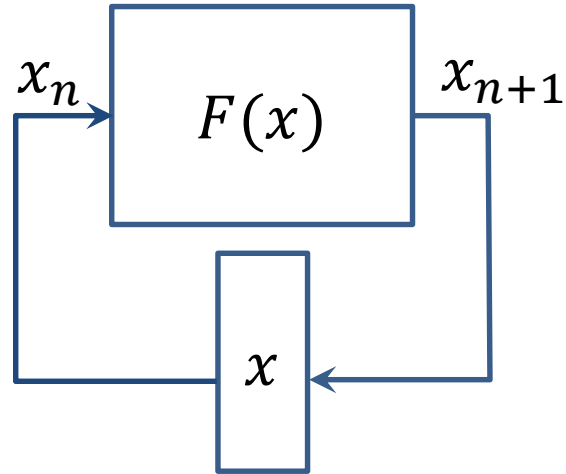
$$\cancel{x^*} + \Delta x_{n+1} = F(\cancel{x^*} + \Delta x_n) \\ \approx F(\cancel{x^*}) + F'(\cancel{x^*}) \cdot \Delta x_n$$

$$|\Delta x_{n+1}| \approx |F'(x^*)| \cdot |\Delta x_n|$$

$$\text{Stabilitás: } |F'(x^*)| < 1$$



A valóság (természet) szimulálható?



Ha F és x_0 csak közelítőleg ismert, akkor x_n is csak közelítés lesz, de ha pontosítjuk F -t és x_0 -t, akkor a hiba zérushoz tart?

$$x_1 = F(x_0) = \tilde{F}(\tilde{x}_0) + \tilde{F}'(\tilde{x}_0)\Delta x_0 + \Delta F(\tilde{x}_0)$$

$$x_2 = F(x_1) = \tilde{F}(\tilde{F}(\tilde{x}_0)) + \tilde{F}'(\tilde{x}_1)\tilde{F}'(\tilde{x}_0)\Delta x_0 + \tilde{F}'(\tilde{x}_1)\Delta F(\tilde{x}_0) + \Delta F(\tilde{x}_1)$$

...

$$x_n = F(x_{n-1}) = \tilde{F}(\tilde{F} \dots (\tilde{x}_0)) + \tilde{F}'(\tilde{x}_{n-1}) \dots \tilde{F}'(\tilde{x}_1)\tilde{F}'(\tilde{x}_0)\Delta x_0 + \tilde{F}'(\tilde{x}_{n-1}) \dots \tilde{F}'(\tilde{x}_0)\Delta F(\tilde{x}_0) + \tilde{F}'(\tilde{x}_{n-1}) \dots \Delta F(\tilde{x}_1) + \dots \Delta F(\tilde{x}_{n-1})$$



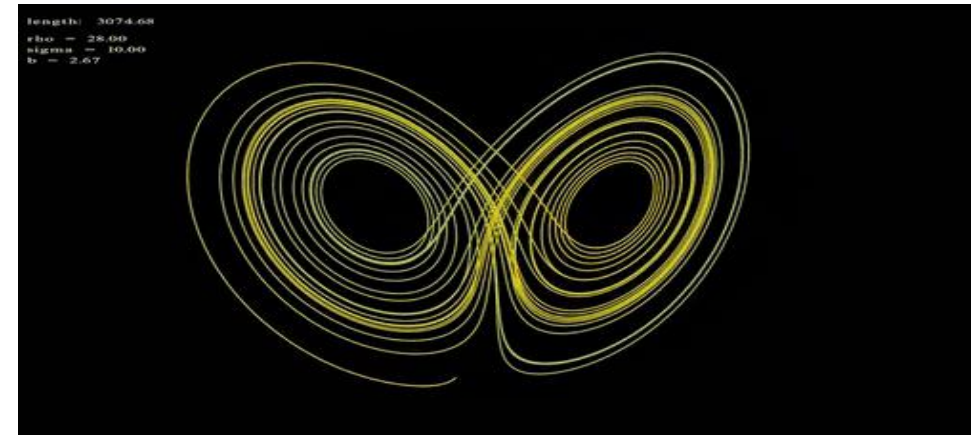
Akkumulált hiba:
Ha $|\tilde{F}'| > 1$, akkor divergál,
hiába tart Δx_0 és ΔF zérushoz

Káosz



A rendszer determinisztikus, de

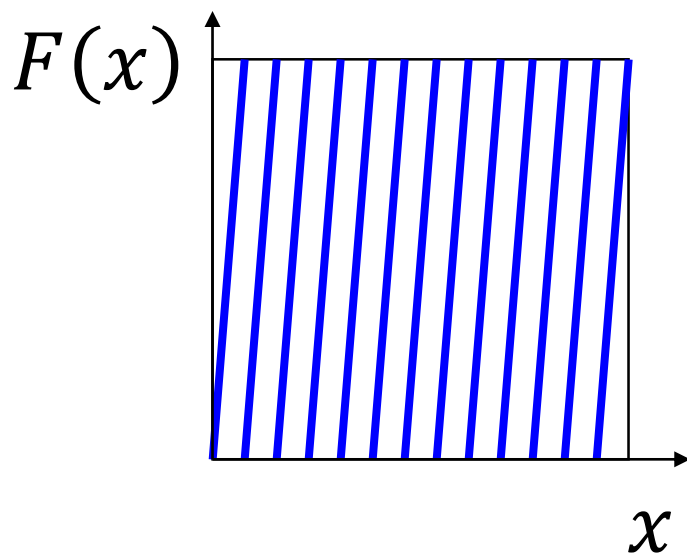
- Kezdeti állapot hatása eltűnik
 - Kis perturbáció nagyon eltérő viselkedéshez vezet
- Auto-korrelációs függvény zérushoz tart
 - Korábbi állapot gyengén befolyásolja a sokkal későbbit
 - Megjósolhatatlanság
- Teljesítmény sűrűség spektrum nem tart zérushoz
 - Nagy frekvencia



Pszeudó véletlenszám generátor

```
static uint x = 3;
void seed(uint s) { x = s; }
uint rand( ) {
    x = F(x);
    return x;
}
```

- $x_{n+1} = F(x_n)$
- $|F'(x)| > 1$,
nagy és állandó



$$F(x) = \{g \cdot x + c\}$$

- $\{z\} = z$ tört része
- g nagy

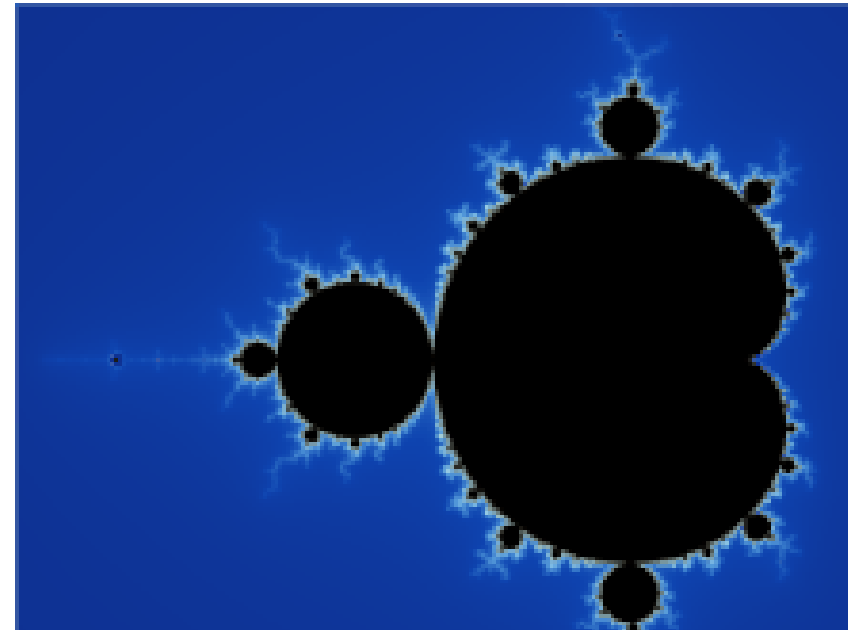
“There's no sense in being precise when you don't even know what you're talking about.”

Neumann János

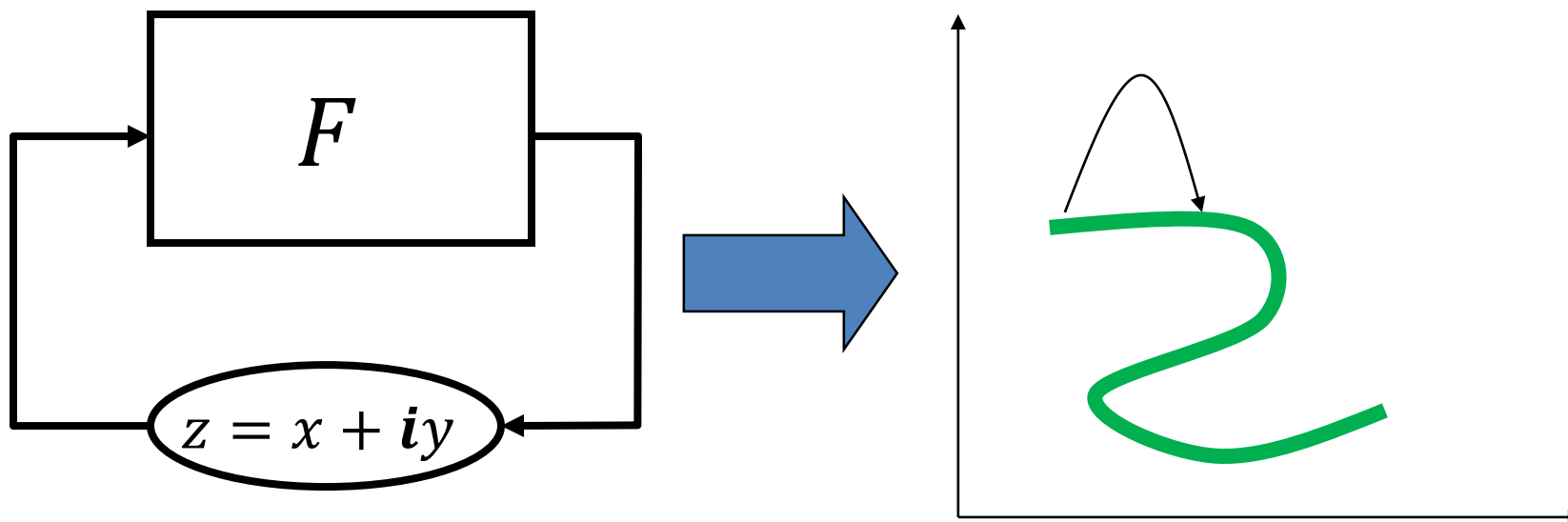
Fraktálok és káosz

3. Kaotikus rendszerek a síkon

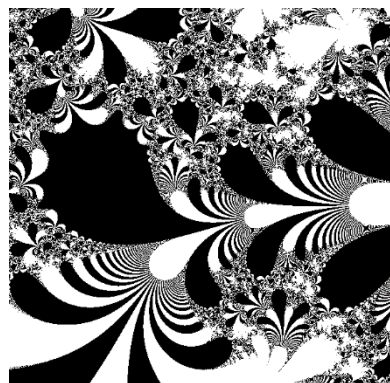
Szirmay-Kalos László



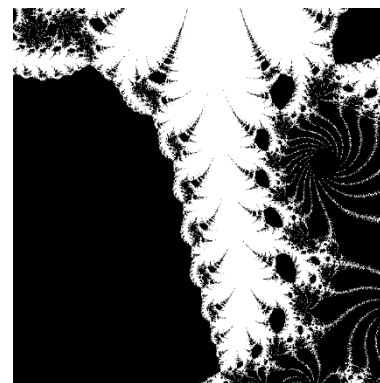
Kaotikus rendszerek a síkon



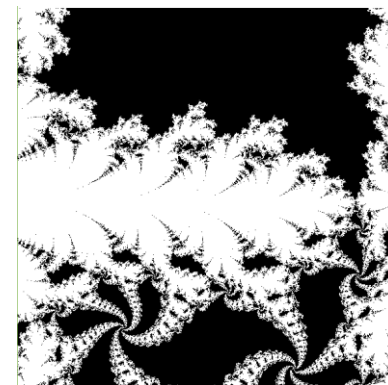
$$F(z) = z^2 + c$$



$$F(z) = e^z + c$$

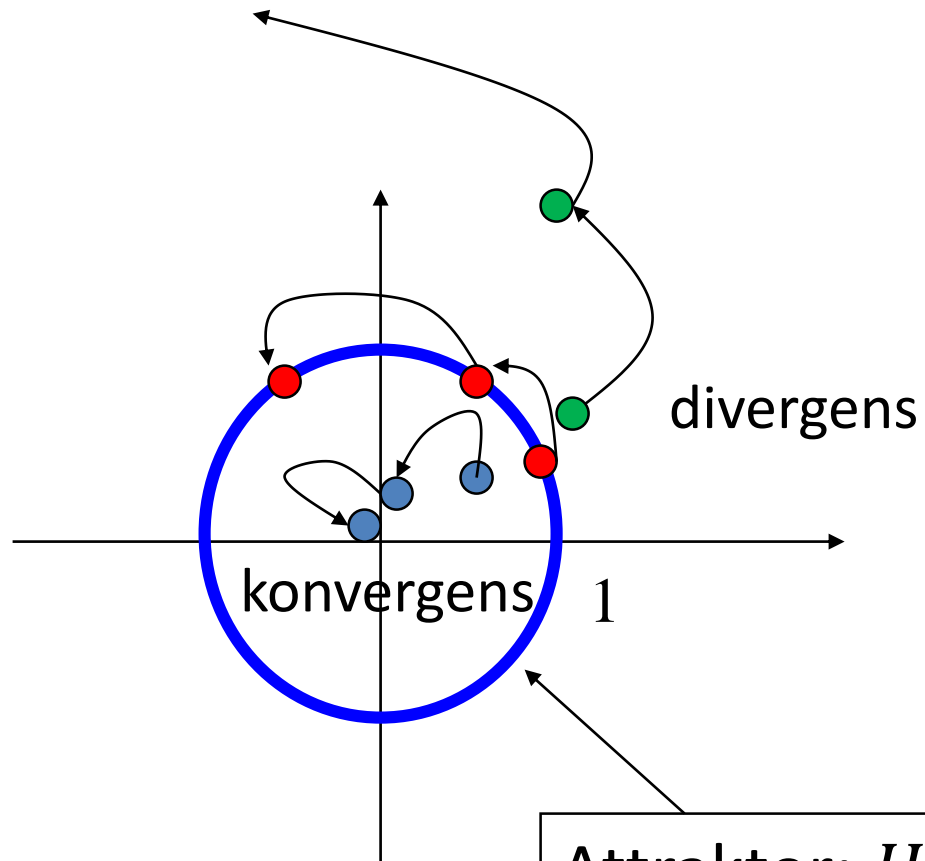


$$F(z) = \cos(z + c)$$



$$F(z) = \sinh(z) + c$$

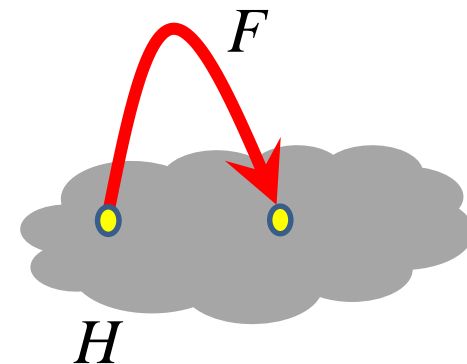
$$F: z \rightarrow z^2$$



$$z = r e^{i\varphi}$$

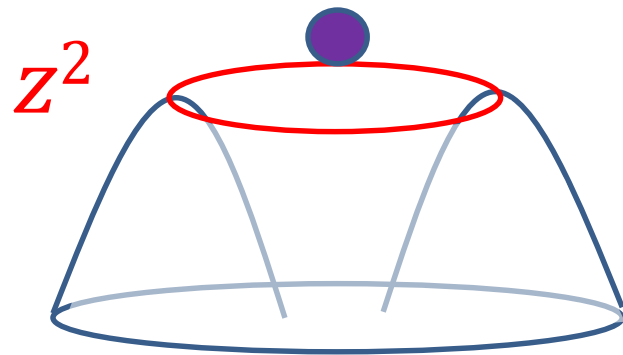
$$r \rightarrow r^2$$

$$\varphi \rightarrow 2\varphi$$



Attraktor felrajzolása

- Attraktor a divergens és konvergens határa: kitöltött attraktor = nem divergens pontok
 - $z_{n+1} = z_n^2$: ha $|z_\infty| < \infty$ akkor fekete
- Attraktorhoz konvergálunk, ha az stabil
 - $z_{n+1} = z_n^2$ attraktora labilis



Inverz iterációs módszer

$$H = F(H)$$

$$z_{n+1} = z_n^2$$

$$\rightarrow \frac{H = F^{-1}(H)}{z_{n+1} = \pm\sqrt{z_n}}$$

$$r_{n+1} = \sqrt{r_n}$$

$$\varphi_{n+1} = \varphi_n/2 + \pi\{0|1\}$$

$$\varphi_{n+1} = \varphi_n/2 + \pi\{0|1\}$$

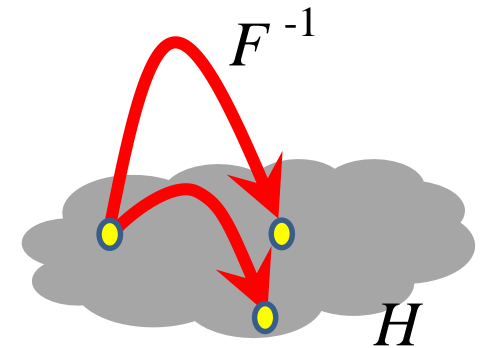
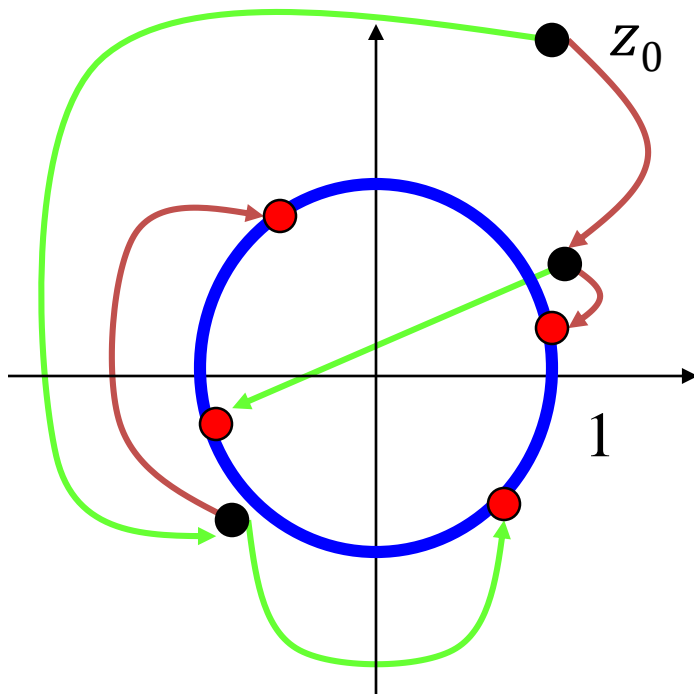
Ha n nagy:

$$r_n = 2^n \sqrt{r_0} \approx 1$$

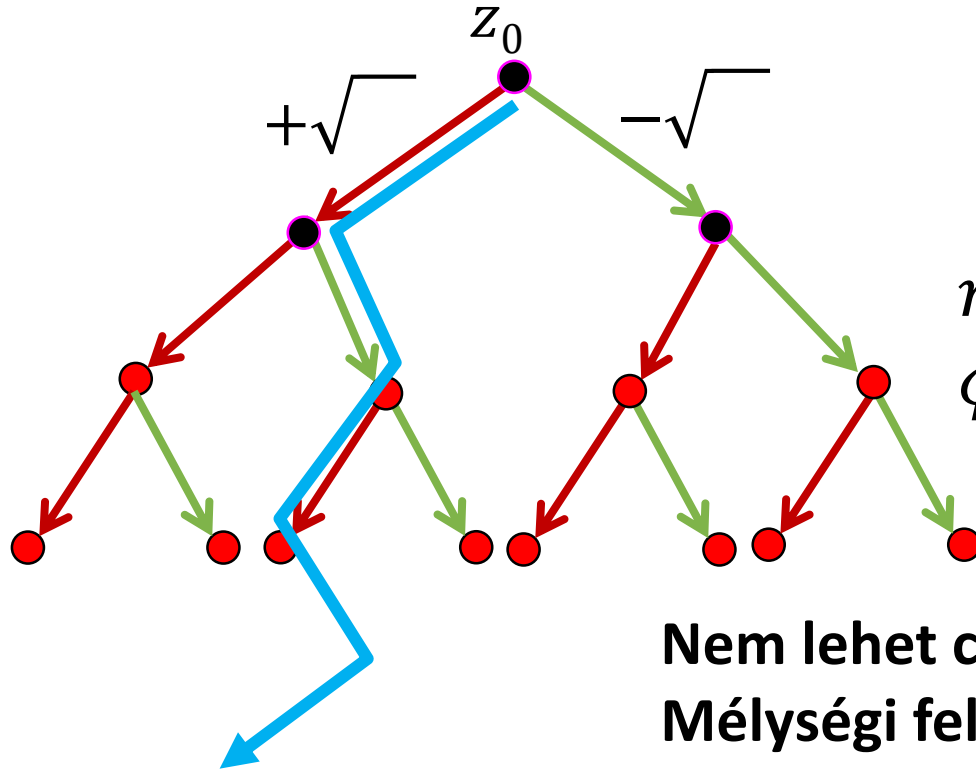
$$\varphi_n = \frac{\varphi_0}{2^n} + \pi\{0|1\} \cdot \{0|1\}\{0|1\} \dots$$

$$\approx \pi\{0|1\} \cdot \{0|1\}\{0|1\} \dots$$

$$n \quad n-1 \quad n-2$$



Többértékű leképezés: Bolyongás



$$r_n \approx 1$$

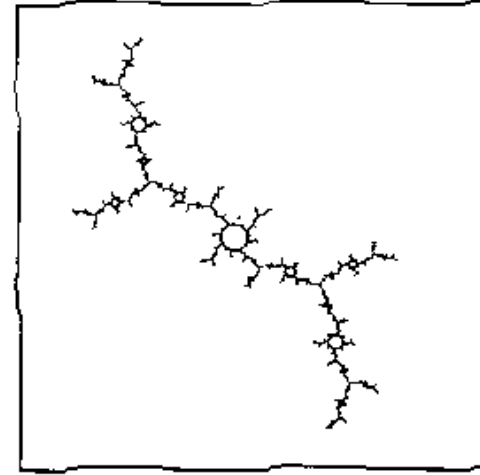
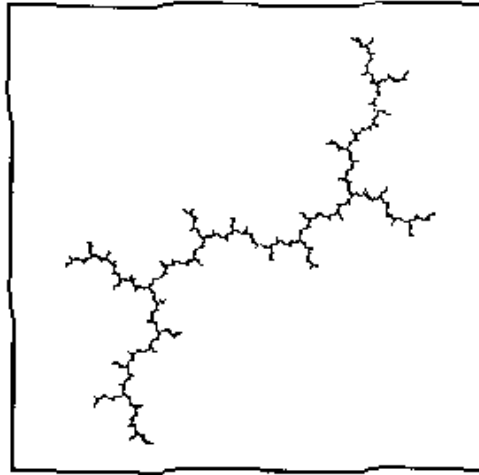
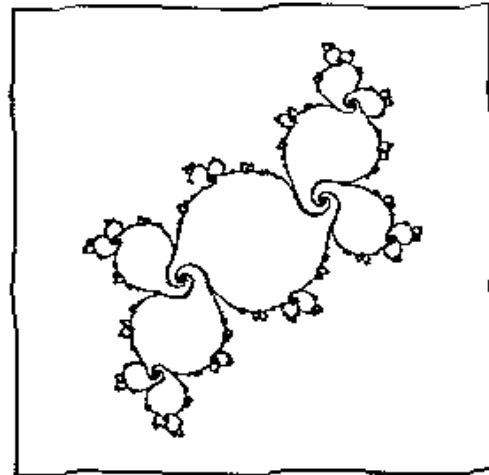
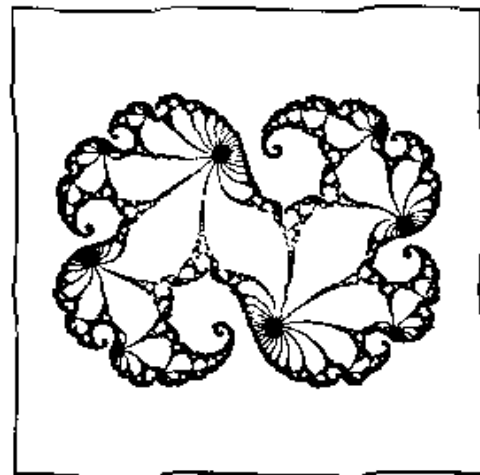
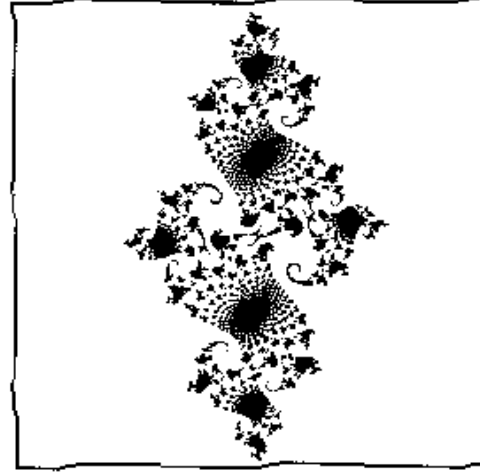
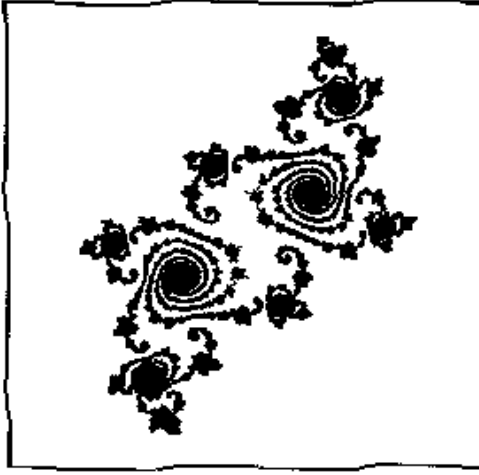
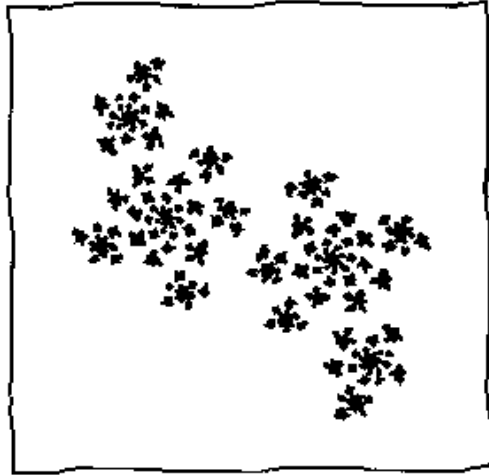
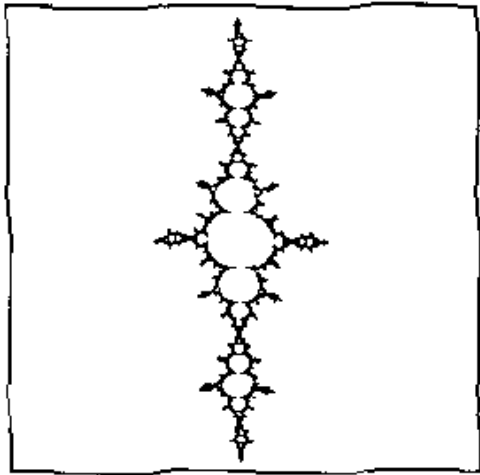
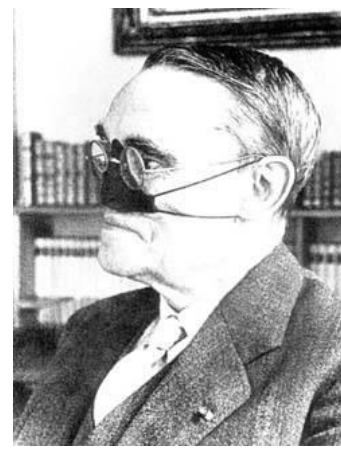
$$\varphi_n \approx \pi \{0|1\} \cdot \{0|1\} \{0|1\} \dots$$

$n \quad n-1 \quad n-2$

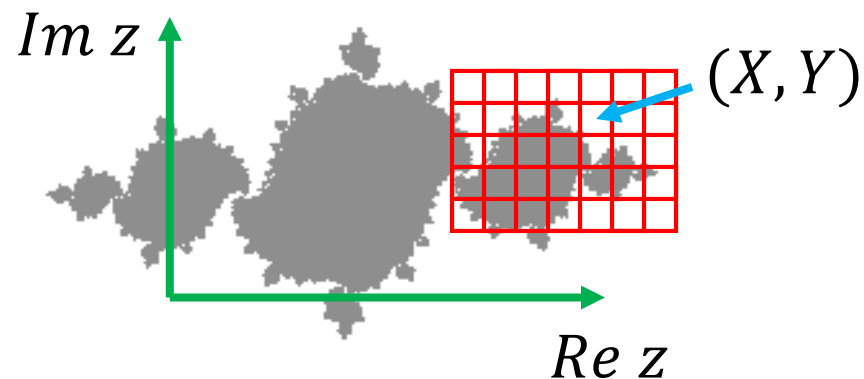
Nem lehet csak egy értékkel dolgozni ???
Mélyégi feltárás szélességi helyett???

- Csak a +: $\varphi_n \approx 0.00 \dots \cdot \pi = 0$
- Csak a -: $\varphi_n \approx 1.1 \dots \cdot \pi = 2\pi \sim 0$
- Felváltva: $\varphi_{2n} \approx 1.010 \dots \cdot \pi = \frac{4\pi}{3}$; $\varphi_{2n+1} \approx 0.1010 \dots \cdot \pi = \frac{2\pi}{3}$
- **Véletlenszerűen!**

(Gaston) Julia halmaz: $F: z \rightarrow z^2 + c$



Kitöltött Julia halmaz: algoritmus



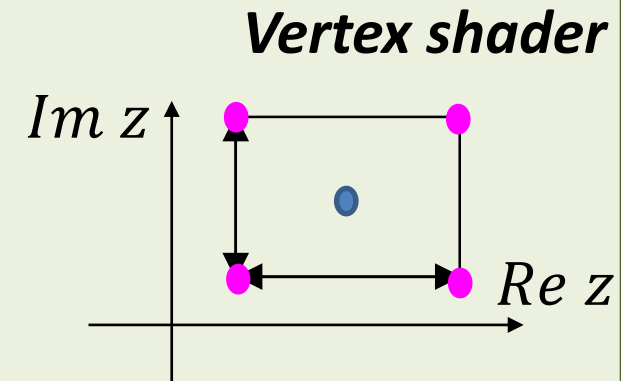
```
FilledJulia( Complex c ) {  
  for(Y = 0; Y < Ymax; ++Y) {  
    for(X = 0; X < Xmax; ++X) {  
      Complex z = ViewportWindow(X, Y);  
      for(n = 0; n < infinity; ++n) z = z2 + c  
      image[Y][X] = (|z| < infinity) ? black : white;  
    }  
  }  
}
```

GPU implementáció

```
float cVtx[] = { -1, -1, 1, -1, 1, 1, -1, 1 };  
glBufferData(GL_ARRAY_BUFFER, sizeof(cVtx), cVtx, GL_STATIC_DRAW);  
...  
glDrawArrays(GL_TRIANGLE_FAN, 0, 4);
```

CPU program

```
uniform vec2 cameraCenter, cameraSize;  
layout(location = 0) in vec2 cVertex;  
out vec2 z0;  
void main() {  
    gl_Position = vec4(cVertex, 0, 1);  
    z0 = cVertex * cameraSize/2 + cameraCenter;  
}
```



```
uniform vec2 c;  
in vec2 z0;  
out vec4 fragCol;  
void main() {  
    vec2 z = z0;  
    for(int i=0; i<1000; i++) z = vec2(z.x*z.x-z.y*z.y, 2*z.x*z.y) + c;  
    fragCol = (dot(z,z) < 100) ? vec4(0, 0, 0, 1) : vec4(1, 1, 1, 1);  
}
```

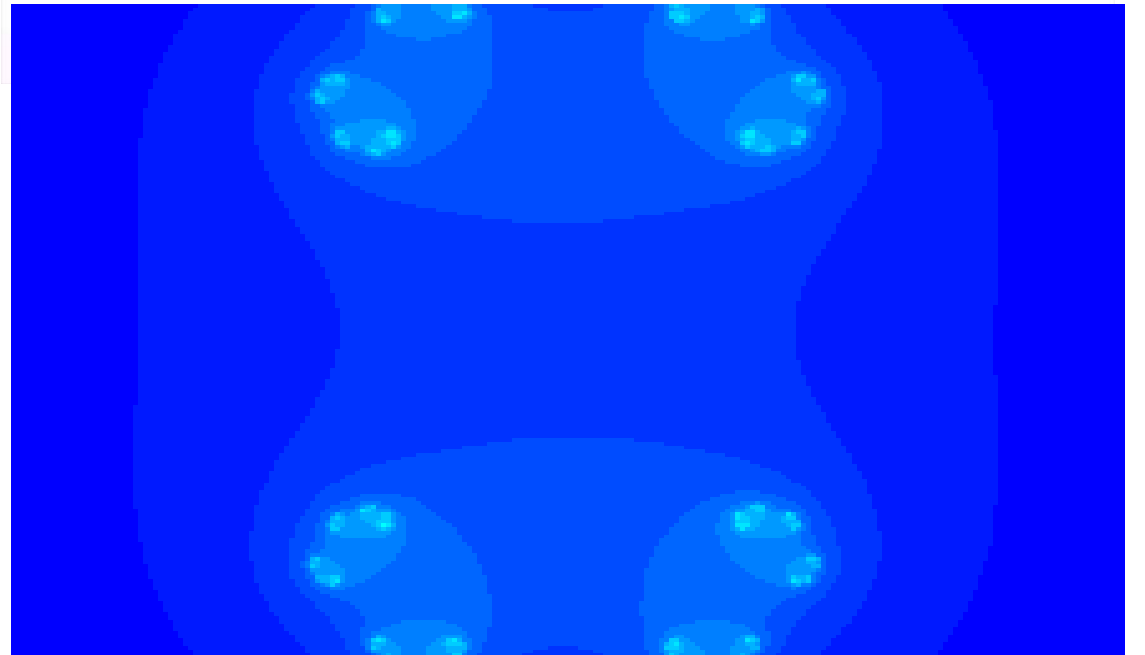
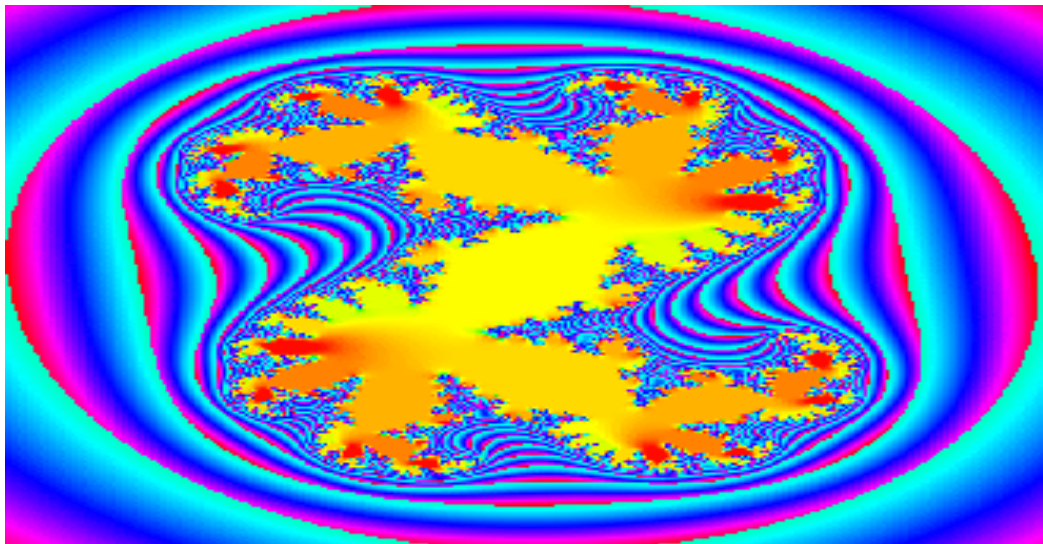
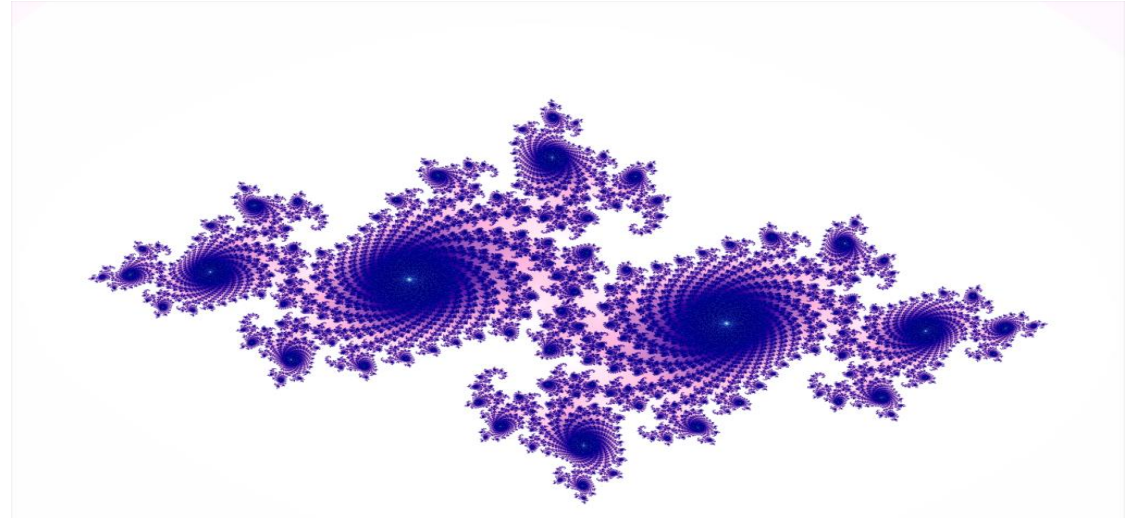
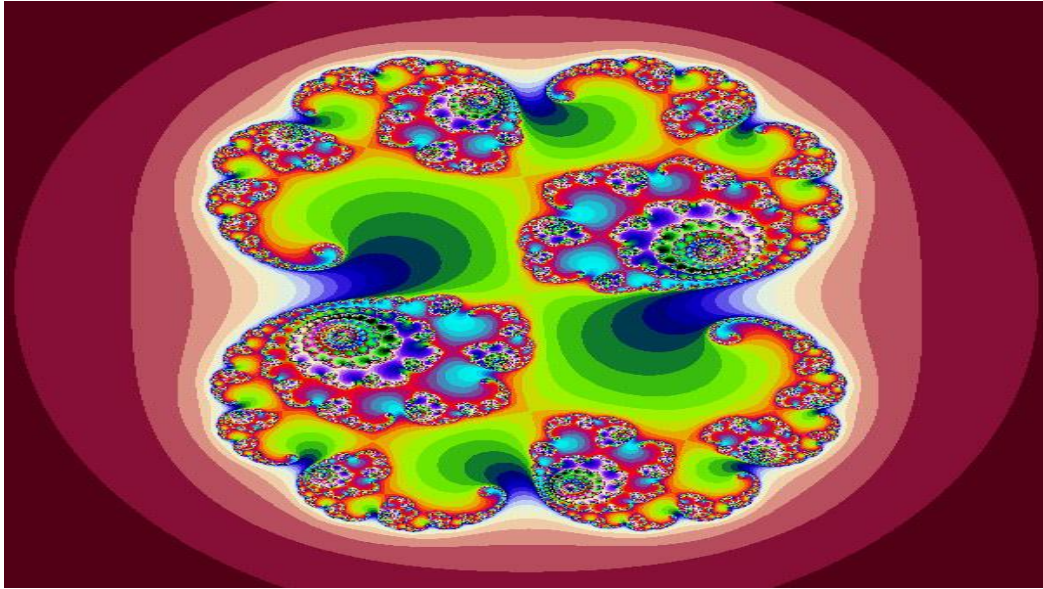
Fragment shader

$$z = z^2 + c$$

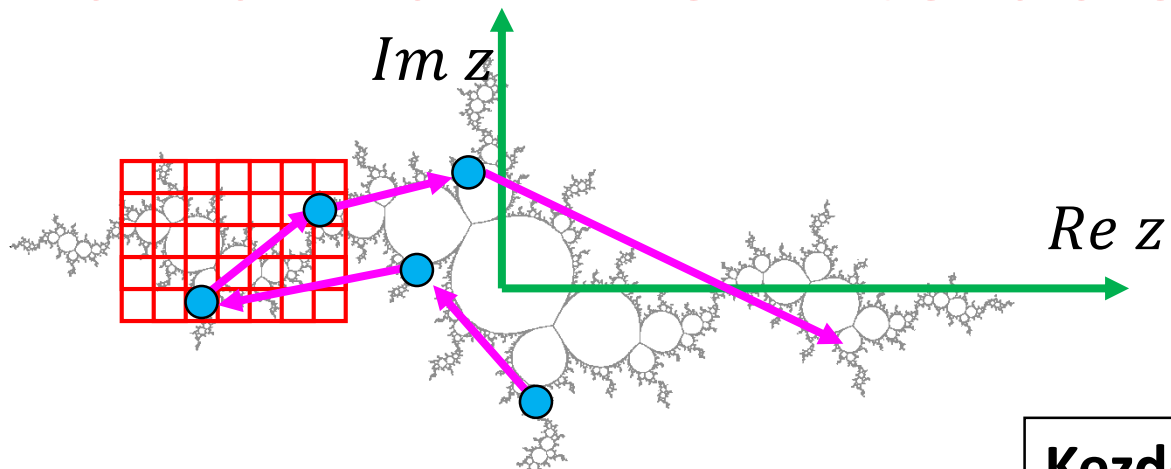
↓



Kitöltött Julia halmaz: kép



Julia halmaz inverz iterációval



```
Julia( Complex c ) {  
  z = Kezdeti érték választás  
  for(n = 0; n < infinity; ++n) {  
    if (InWindow( z )) {  
      pixel = WindowViewport( z );  
      image[pixel] = black;  
    }  
    z =  $\sqrt{z - c}$   
    if (random( ) > 0.5) z = -z;  
  }  
}
```

Kezdeti z érték:

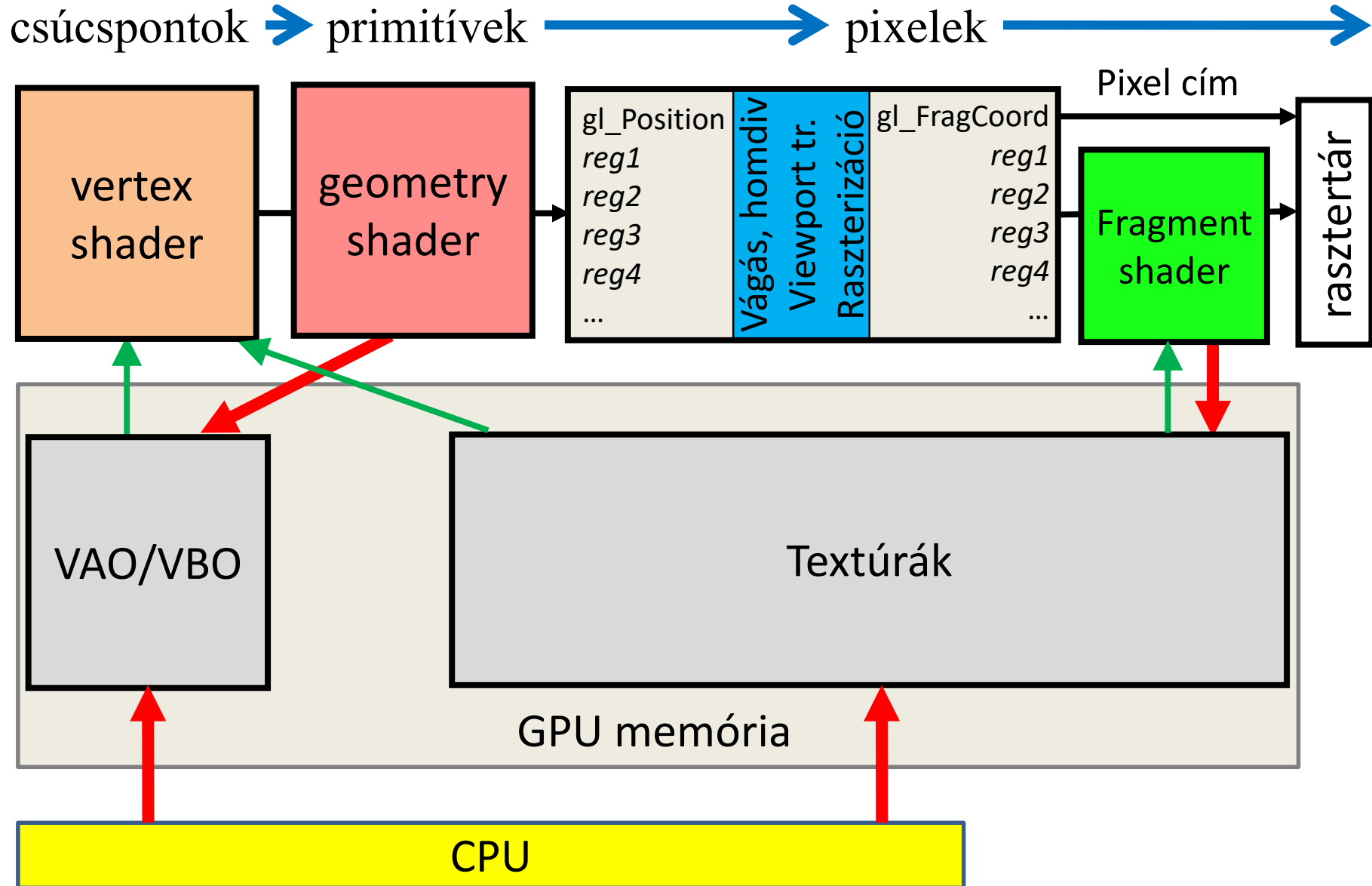
$$z = z^2 + c \text{ gyöke}$$

Inverz:

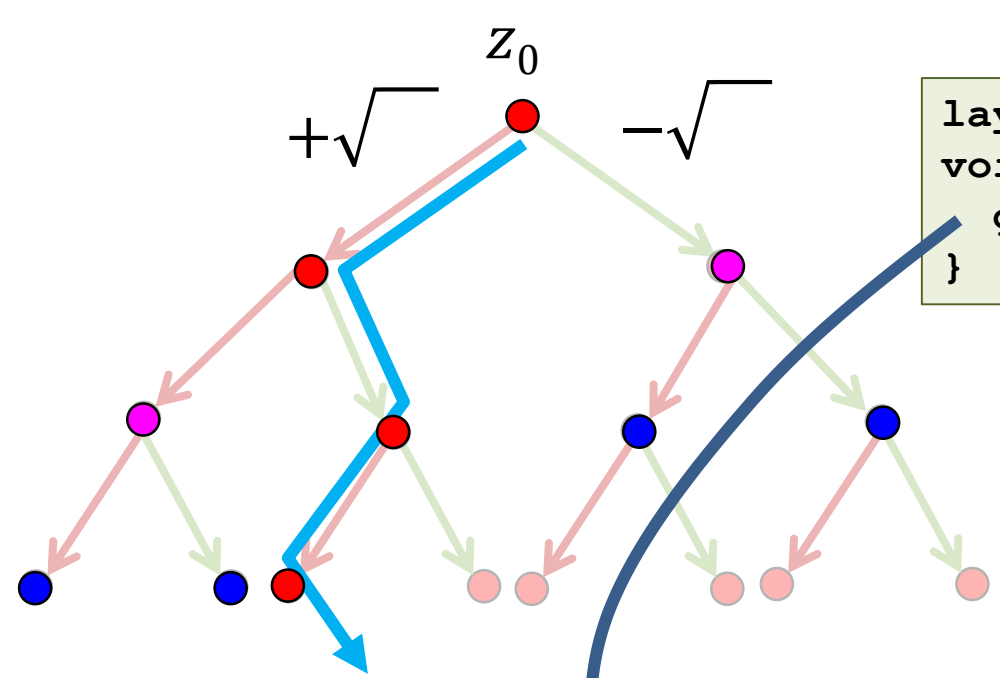
$$F(z) = z^2 + c$$

$$F^{-1}(z) = \pm\sqrt{z - c}$$

Geometry shader



GPU implementáció

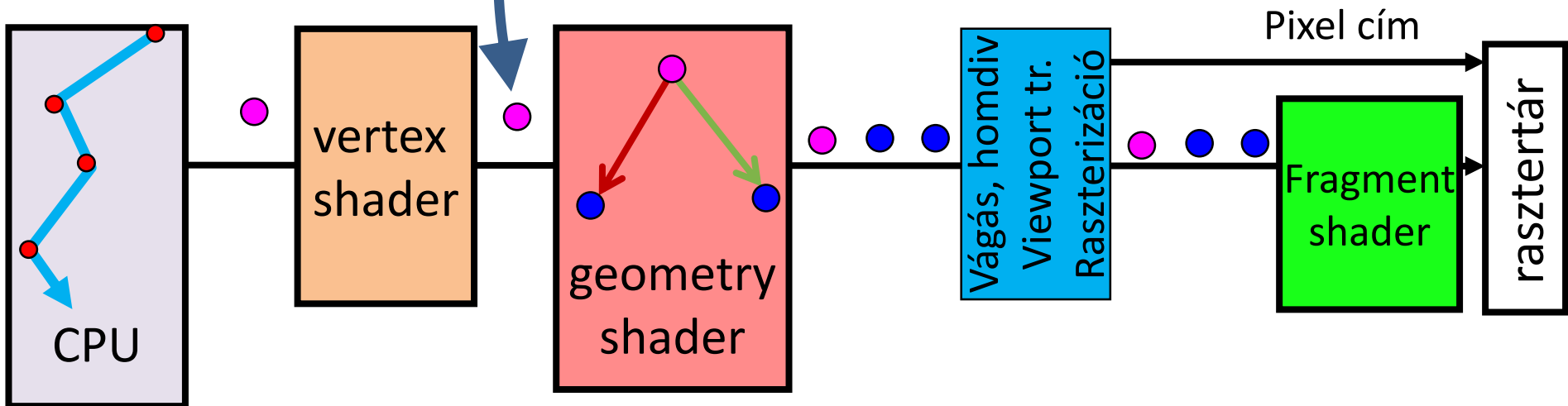


Vertex shader

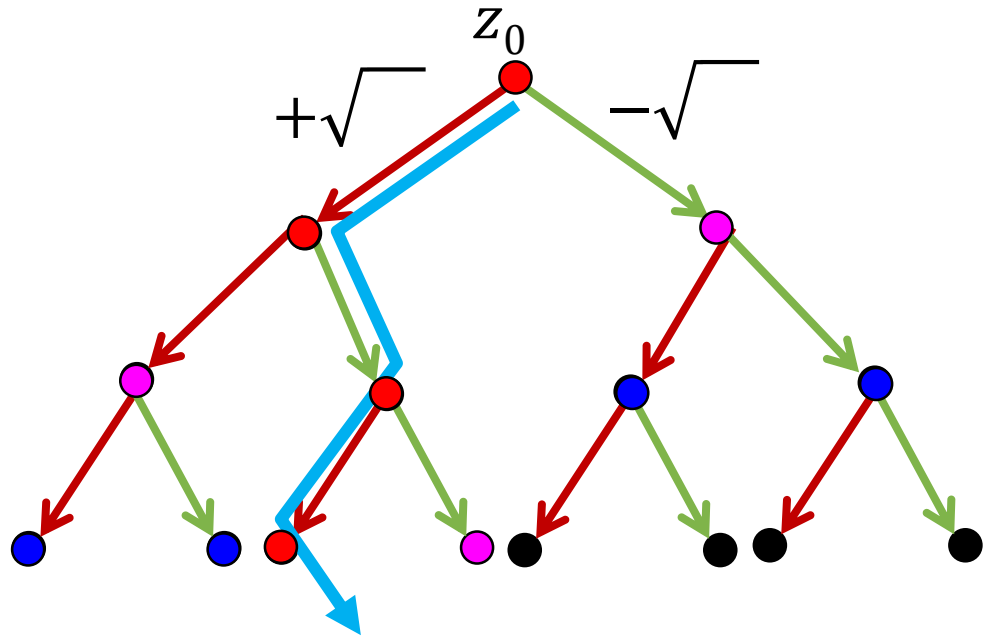
```
layout(location = 0) in vec2 zRoot;  
void main() {  
    gl_Position = vec4(zRoot, 0, 1);  
}
```

Fragment shader

```
void main() {  
    fragColor = vec4(0, 0, 0, 1);  
}
```



CPU program



Kezdeti z érték:
 $z^2 = z - c$ gyöke

```
vec2 z = vec2(0.5, 0) + sqrtComplex(vec2(0.25 - c.x, -c.y));  
for (int p = 0; p < nPackets; p++) {  
    vec2 vtx[nSeeds];  
    for (int i = 0; i < nSeeds; i++) {  
        z = sqrtComplex(z - c) * (rand() & 1 ? 1 : -1);  
        vtx[i] = -z;  
    }  
    glBufferData(GL_ARRAY_BUFFER, sizeof(vtx), vtx, GL_DYNAMIC_DRAW);  
    glDrawArrays(GL_POINTS, 0, nSeeds);  
}
```

Geometry shader



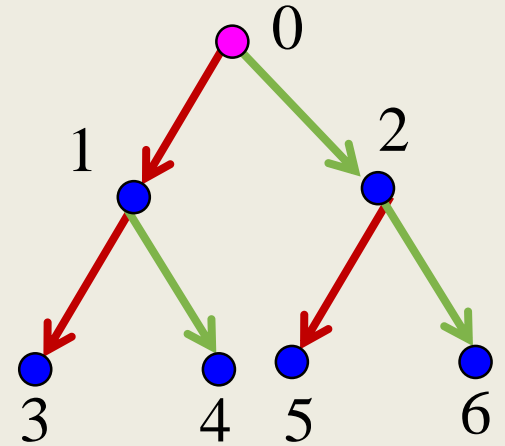
```
uniform vec2 cameraCenter, cameraSize, c;

layout(points) in;
layout(points, max_vertices = 63) out;

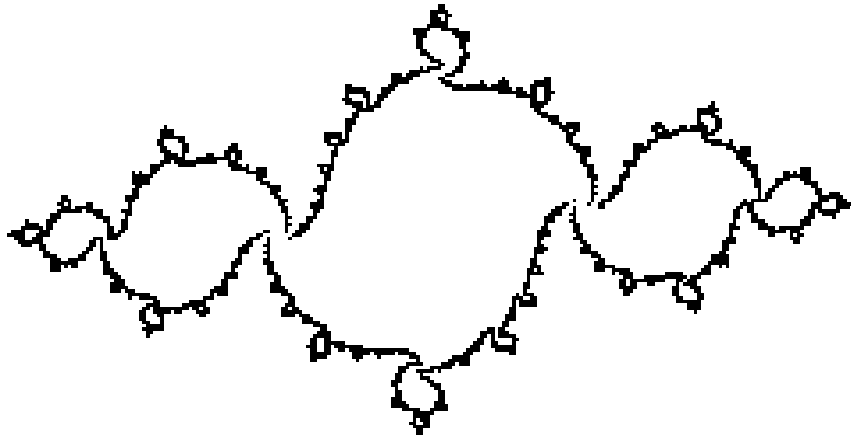
vec2 sqrtComplex(vec2 z) {
    float r = length(z), phi = atan(z.y, z.x);
    return vec2(cos(phi/2), sin(phi/2)) * sqrt(r);
}

void main() {
    vec2 zs[63];
    zs[0] = gl_in[0].gl_Position.xy;
    gl_Position = vec4((zs[0]-cameraCenter)/(cameraSize/2),0,1);
    EmitVertex();

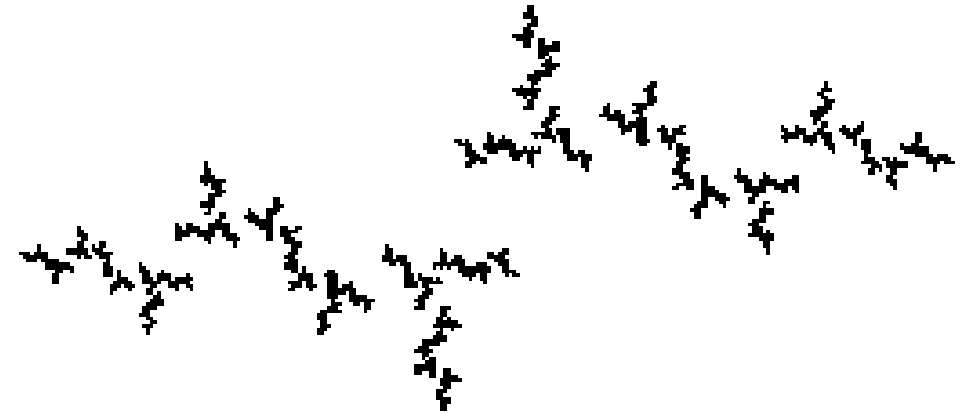
    for(int i = 0; i < 63/2; i++) {
        vec2 z = sqrtComplex(zs[i] - c);
        for(int j = 1; j <= 2; j++) {
            zs[2 * i + j] = z;
            gl_Position = vec4((z-cameraCenter)/(cameraSize/2),0,1);
            EmitVertex();
            z = -z;
        }
    }
    EndPrimitive();
}
```



Julia halmaz összefüggése



Összefüggő

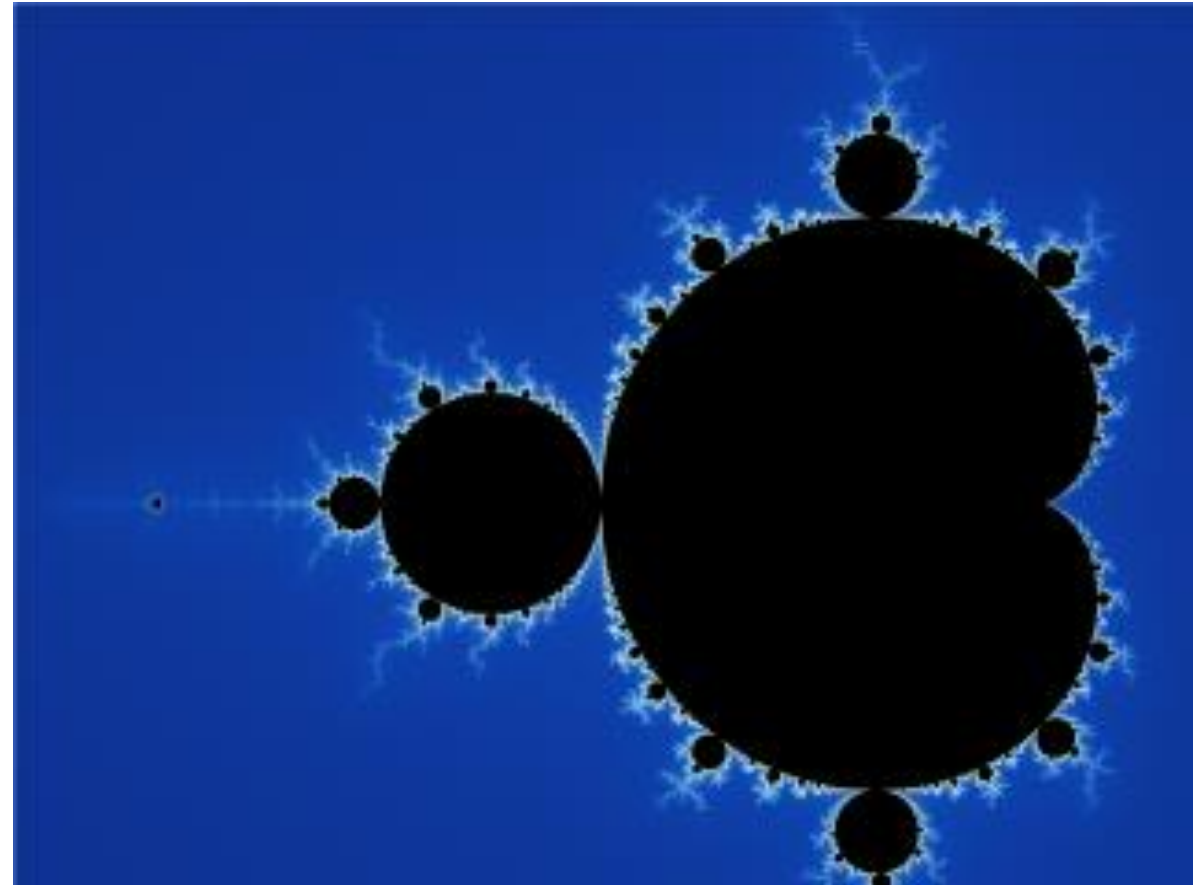
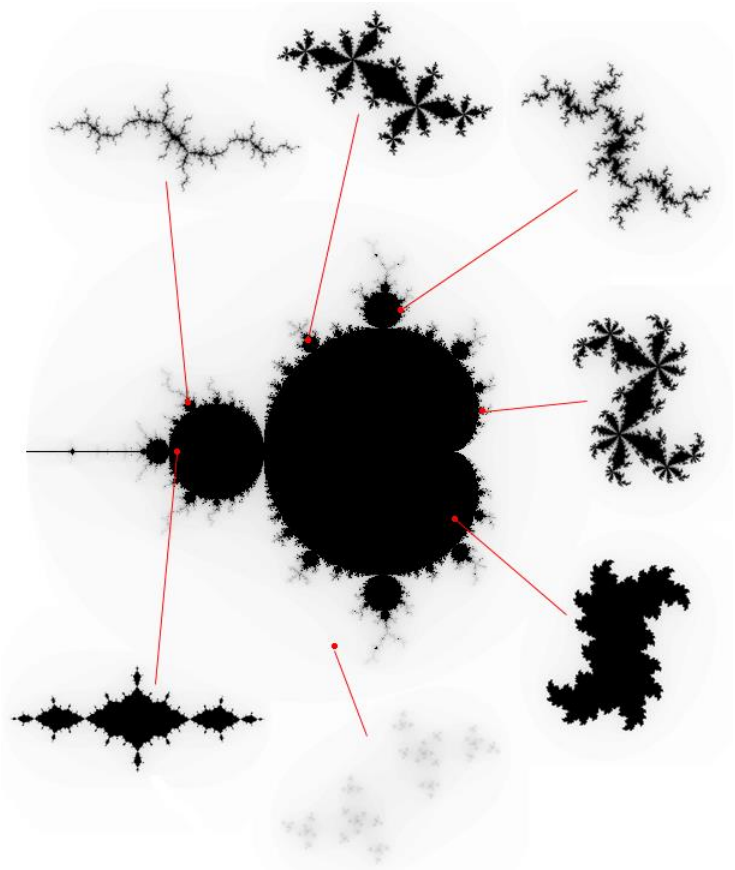


Nem összefüggő,
Cantor féle halmaz

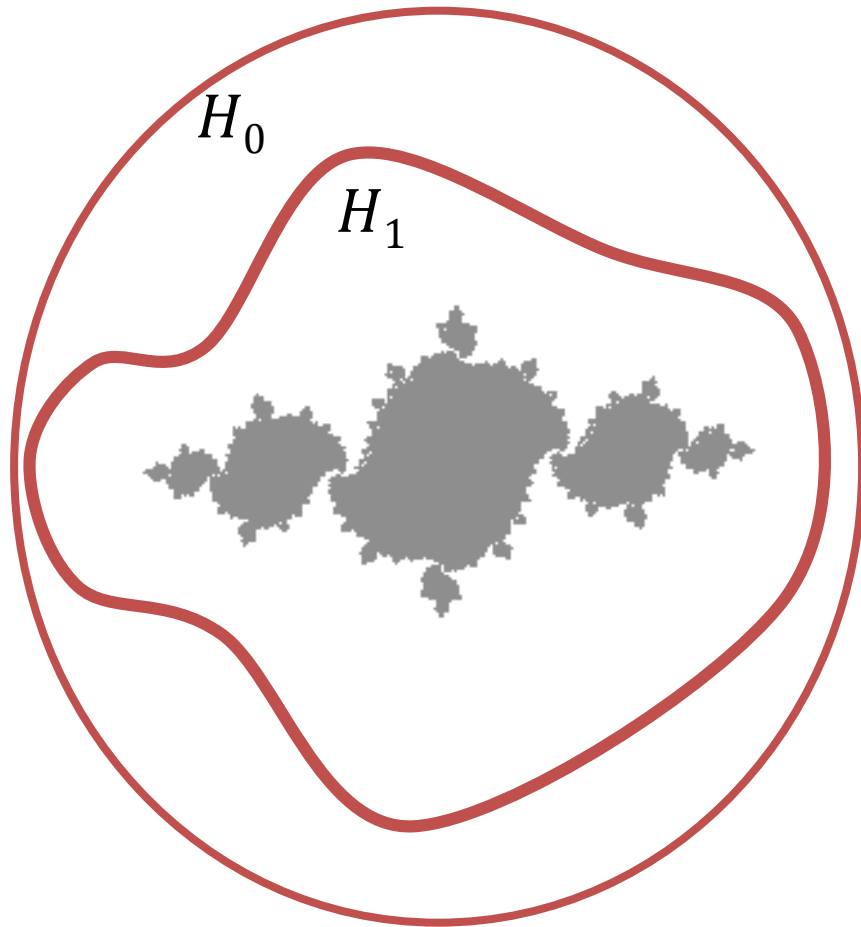
(Benoit) Mandelbrot halmaza



Azon c komplex számok, amelyekre a $z \rightarrow z^2 + c$ Julia halmaza összefüggő.



Julia halmaz összefüggőssége



$$H_1 = F(H_0)$$

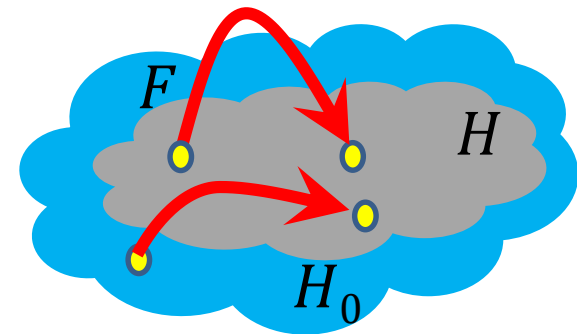
$$H_2 = F(H_1)$$

$$H_3 = F(H_2)$$

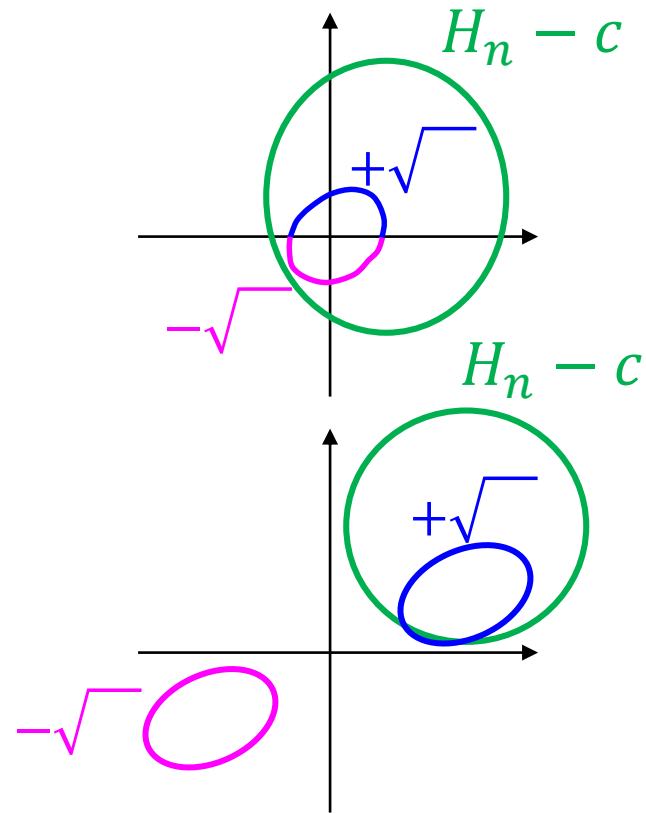
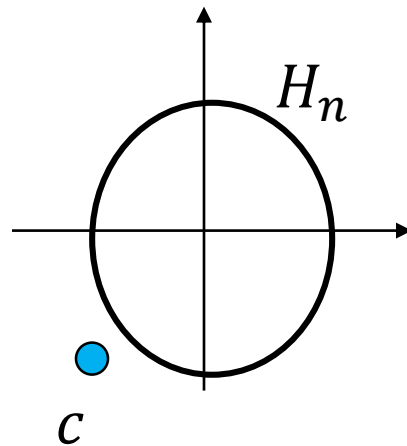
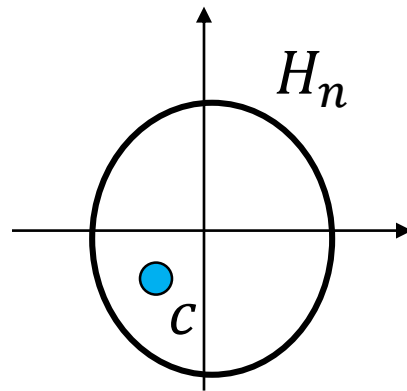
...

$$F(z) = \pm\sqrt{z - c}$$

$$H = F(H)$$



Julia halmaz összefüggősége



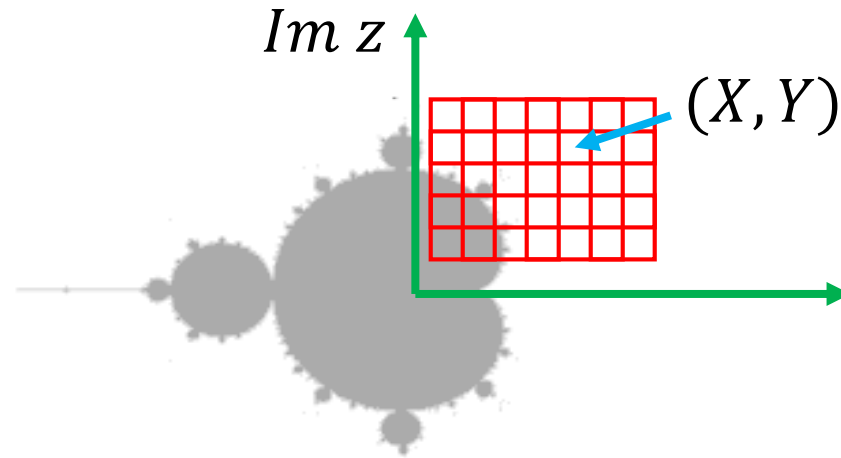
$$H_{n+1} = \pm\sqrt{H_n - c}$$

(Benoit) Mandelbrot halmaz

Azon c komplex számok halmaza, amelyekre:

- $z \rightarrow z^2 + c$ Julia halmaza összefüggő.
- c a Julia halmaz attraktorában vagy konvergens tartományában.
- $z_{n+1} = z_n^2 + c$ iteráció a c -ből indítva nem divergens.

Mandelbrot halmaz rajzolás



```
Mandelbrot ( ) {  
  for(Y = 0; Y < Ymax; ++Y) {  
    for(X = 0; X < Xmax; ++X) {  
      Complex c = ViewportWindow(X, Y);  
      Complex z = c;  
      for(n = 0; n < infinity; ++n) z = z2 + c  
      image[Y][X] = (|z| < infinity) ? black : white;  
    }  
  }  
}
```

$|z| < 2$

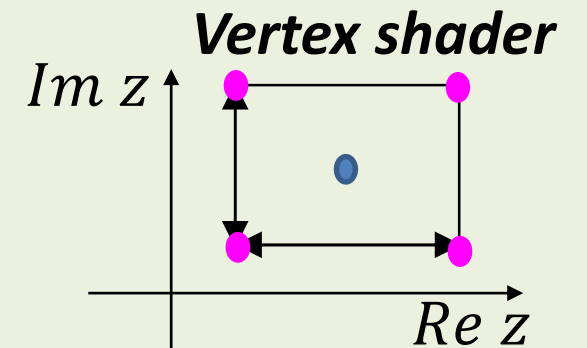
GPU implementáció



```
float cVtx[] = { -1, -1, 1, -1, 1, 1, -1, 1 };
glBufferData(GL_ARRAY_BUFFER, sizeof(cVtx), cVtx, GL_STATIC_DRAW);
...
glDrawArrays(GL_TRIANGLE_FAN, 0, 4);
```

CPU program

```
uniform vec2 cameraCenter, cameraSize;
layout(location = 0) in vec2 cVertex;
out vec2 c;
void main() {
    gl_Position = vec4(cVertex, 0, 1);
    c = cVertex * cameraSize/2 + cameraCenter;
}
```

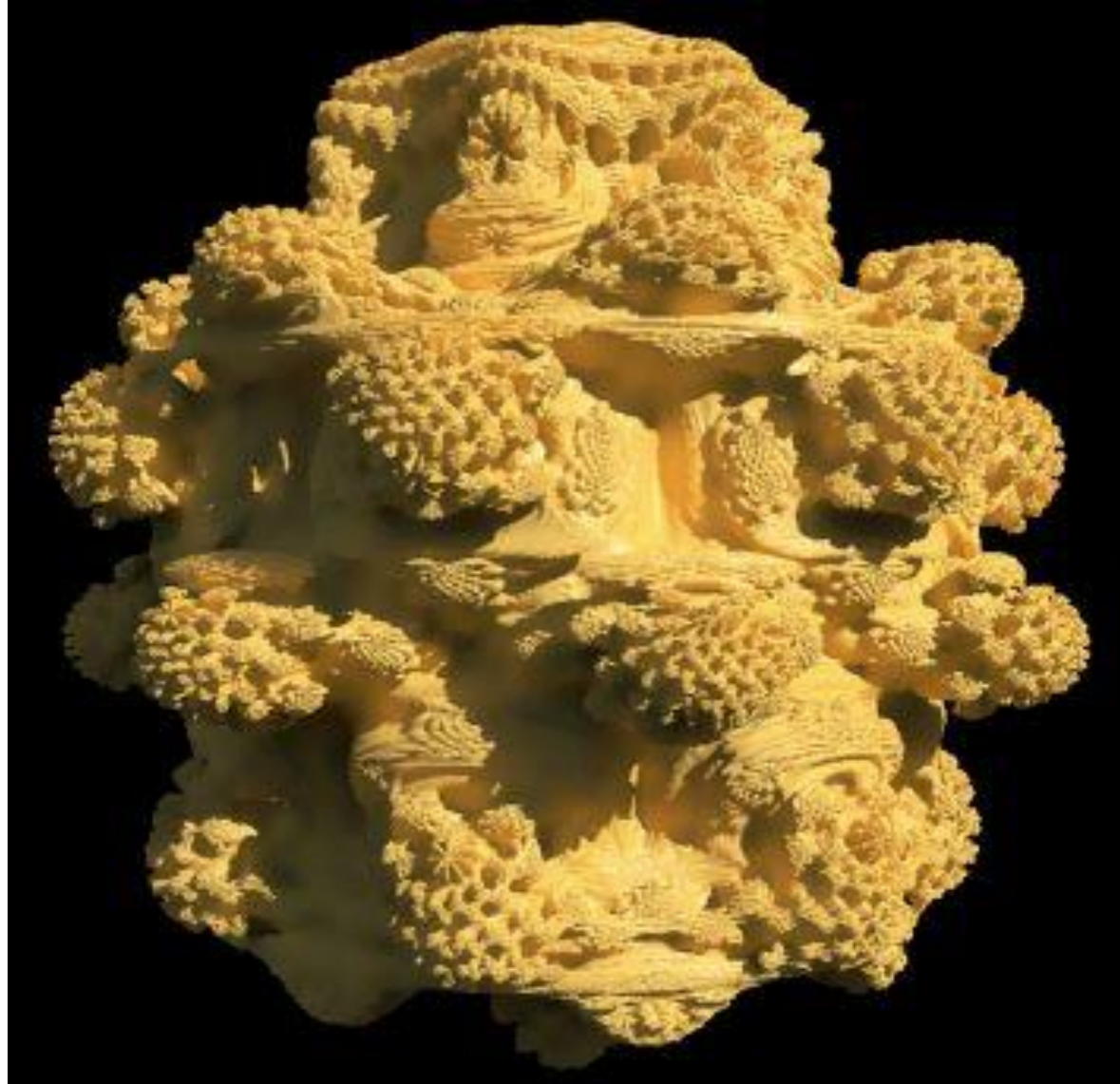


```
in vec2 c;
out vec4 fragCol;
const int nIteration = 1000;
void main() {
    vec2 z = c;
    for(int i = 0; i < nIteration; i++) {
        z = vec2(z.x * z.x - z.y * z.y, 2 * z.x * z.y) + c;
        if (dot(z, z) > 4) break;
    }
    fragCol = (i == nIteration) ? vec4(0, 0, 0, 1) : vec4(1, 1, 1, 1);
}
```

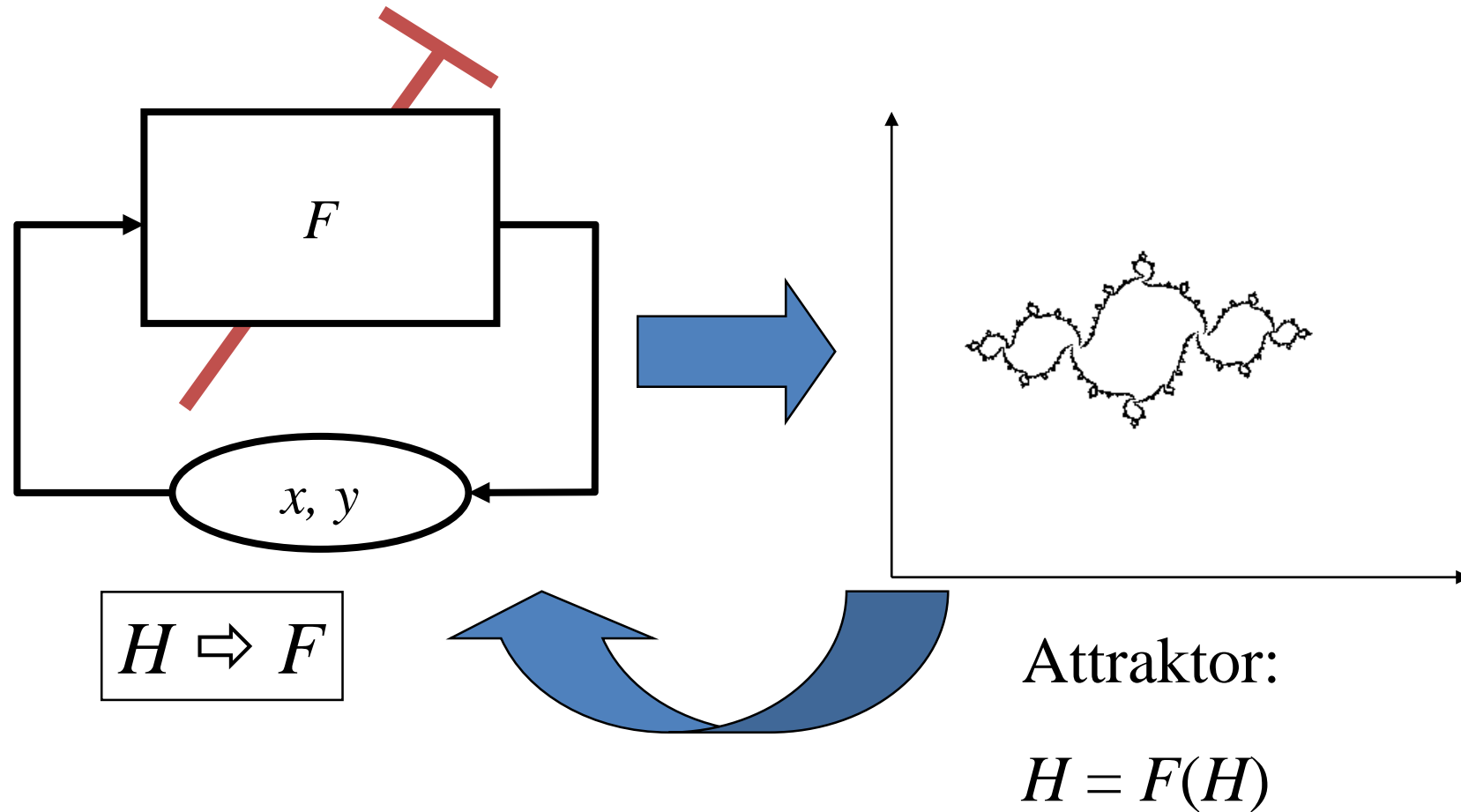
$$z = z^2 + c$$

Fragment shader

Mandelbulb



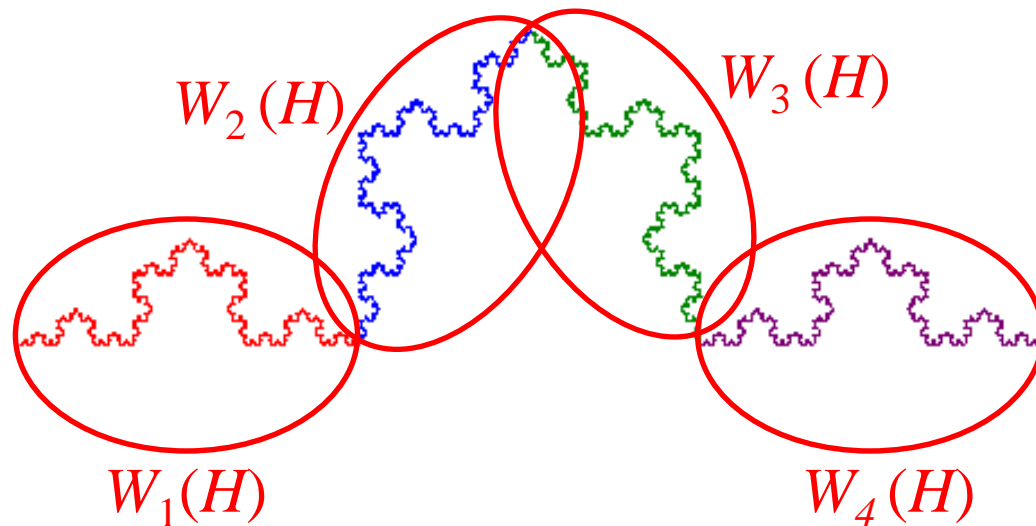
Inverz feladat: IFS modellezés



F : szabadon vezérelhető, legyen stabil attraktora

Melyik függvény attraktora a Koch görbe?

Attraktor: $H = F(H) = W_1(H) \cup W_2(H) \cup W_3(H) \cup W_4(H)$



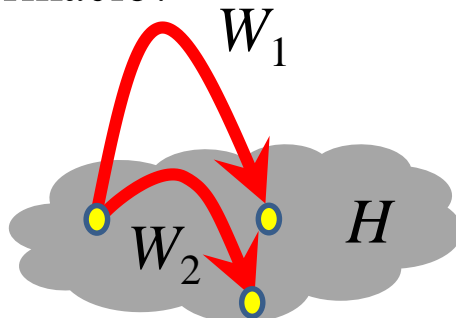
$$W_k(x,y) = [x,y] \cdot \mathbf{A}_k + \mathbf{q}_k$$

F : többértékű lineáris leképezés

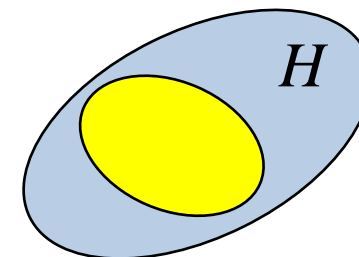
Nem feltétlenül hasonlósági transzformáció!

Nem csak önazonos objektumok.

$$F = W_1 \vee W_2 \vee \dots \vee W_m$$



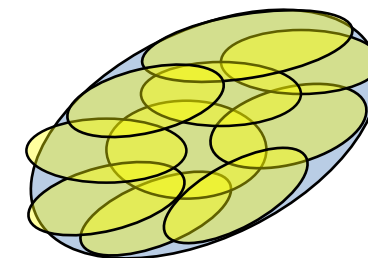
$$W_k(x, y) = [x, y] \cdot \mathbf{A}_k + \mathbf{q}_k$$



Stabilitás = kontrakció
 $|\mathbf{A}$ sajátértékei < 1

$$H = W_1(H) \cup W_2(H) \cup \dots \cup W_m(H)$$

$$H = F(H)$$



Kollázs:

lefedés a kicsinyített változatokkal

Nem feltétlenül diszjunkt

IFS rajzolás: iterációs algoritmus

IFSDraw ()

Legyen $[x,y] = [x,y] A_1 + q_1$ megoldása a kezdő $[x,y]$

FOR $i = 0$ TO “infinity” DO

IF InWindow(x, y)

WindowViewport(x, y \Rightarrow X, Y)

Write(X, Y, color);

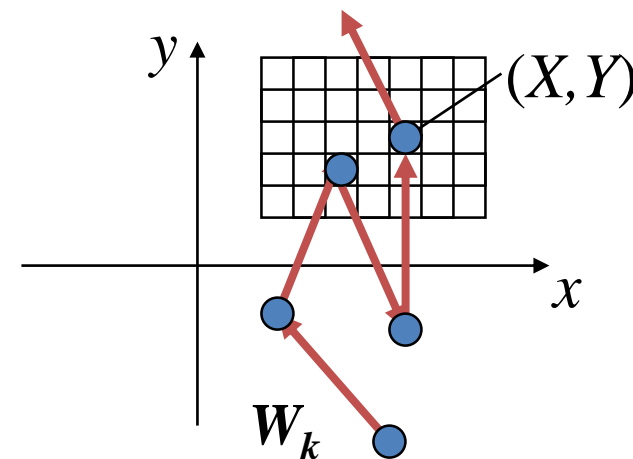
ENDIF

Válassz k -t p_k valószínűséggel

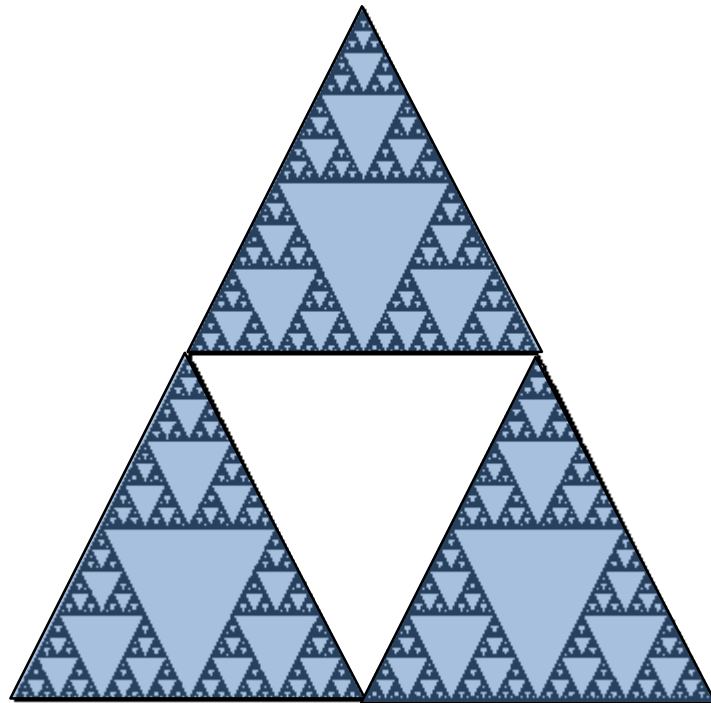
$[x,y] = [x,y] A_k + q_k$

ENDFOR

END



Egyszerű IFS-ek



IFS modellezés



IFS képek

