“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.”

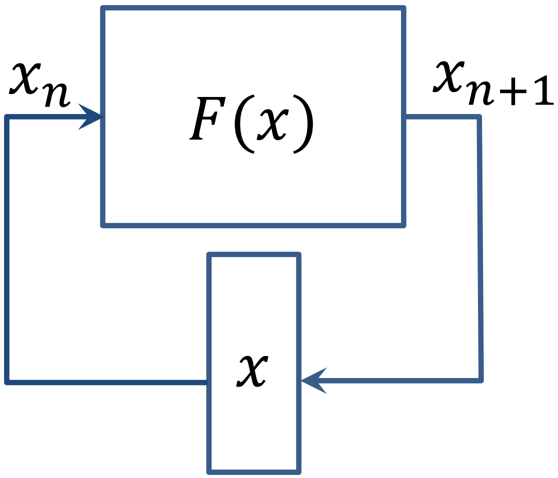
Benoit Mandelbrot

Fraktálok és káosz

1. Fraktális dimenzió

Szirmay-Kalos László

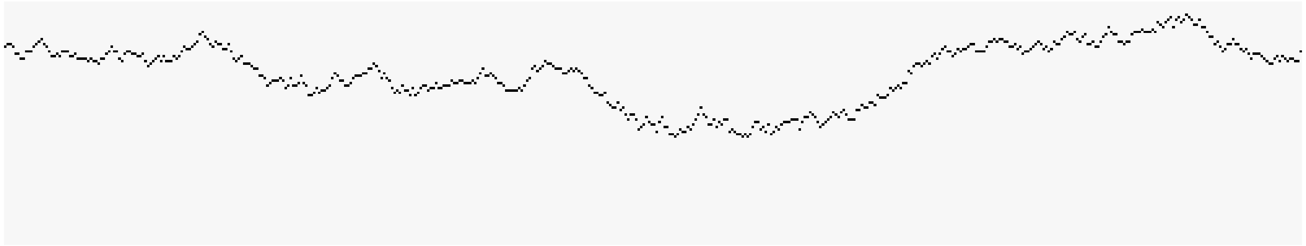
A valóság (természet) szimulálható?



~~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.~~

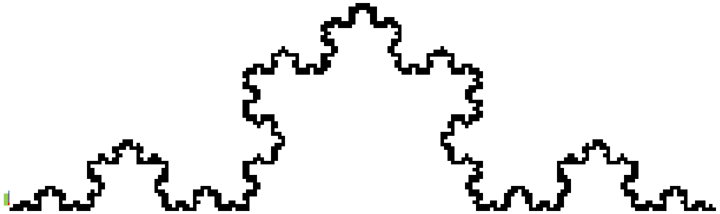
A valóság (természet) metrikus?

- **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): $\boldsymbol{r}(t)$; felület (2D): $\boldsymbol{r}(u, v)$; **milyen méret értelmes?**



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

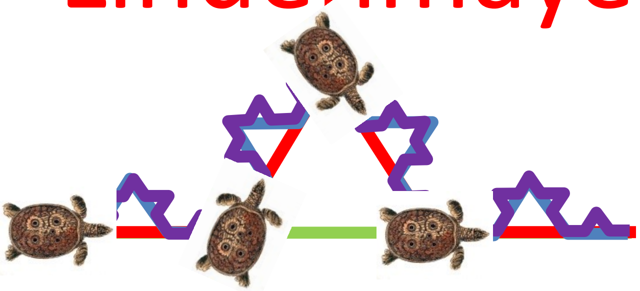
(Helge von) Koch görbe: hossz végtelen



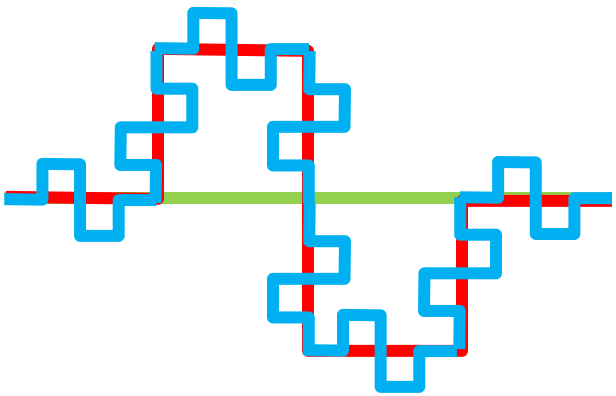
$$l_n = l_0 \left(\frac{4}{3}\right)^n \rightarrow \infty$$

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

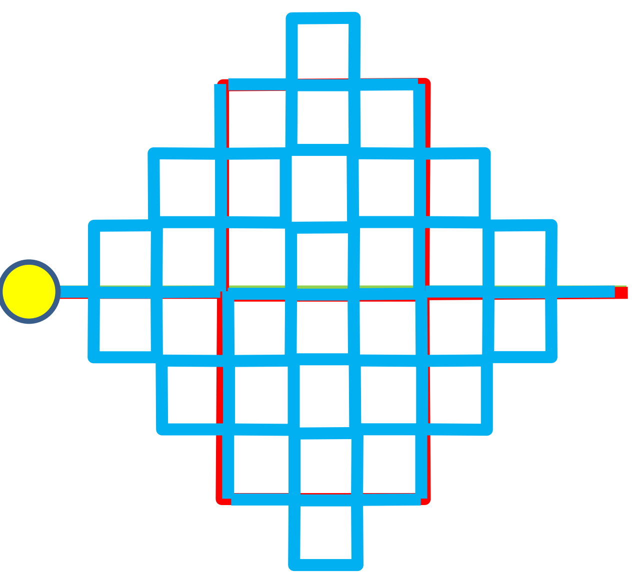
Lindenmayer (Arisztid) rendszerek



F→F+60F-120F+60F



F→F+90F-90F-90FF+90F+90F-90F



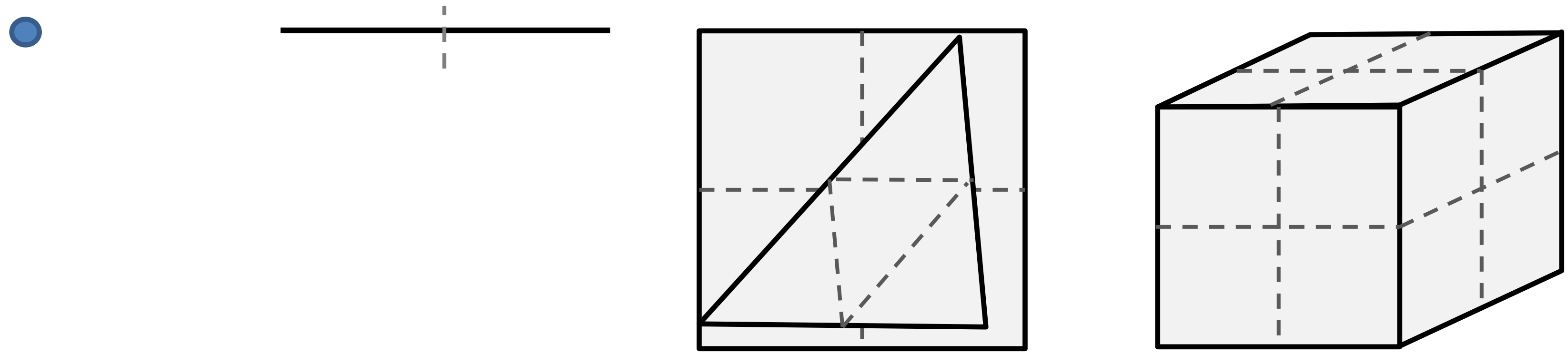
F→F+90F-90F-90F-90F+90F+90F+90F-90F

Peano/Hilbert térkitöltő görbe

(Felix) Hausdorff dimenzió önhasonló objektumokra

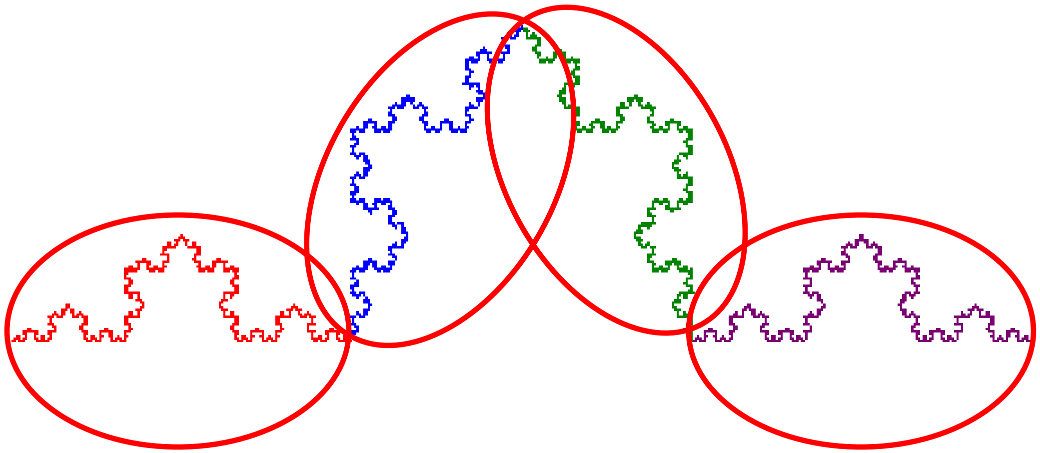
$r = 1/2$ kicsinyítés

$N = 1$ $\quad N = 2$ $\quad N = 4$ $\quad N = 8$



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

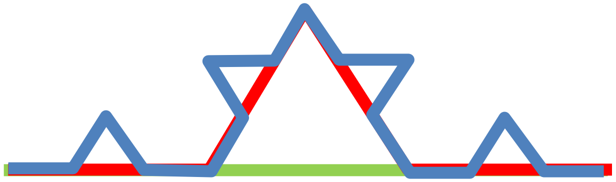
Koch görbe Hausdorff dimenziója



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

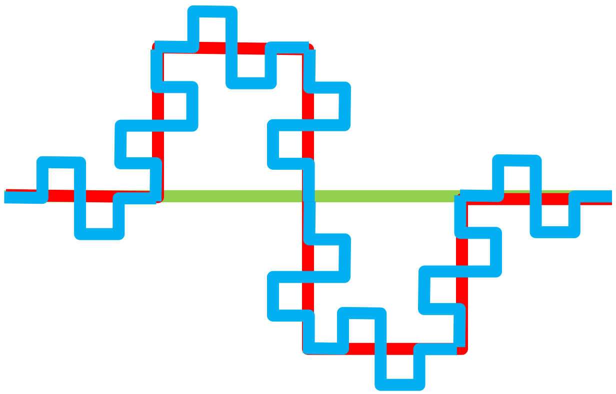
$$D = \frac{\log(4)}{\log(3)} \approx 1.26$$

Lindenmayer (Arisztid) rendszerek



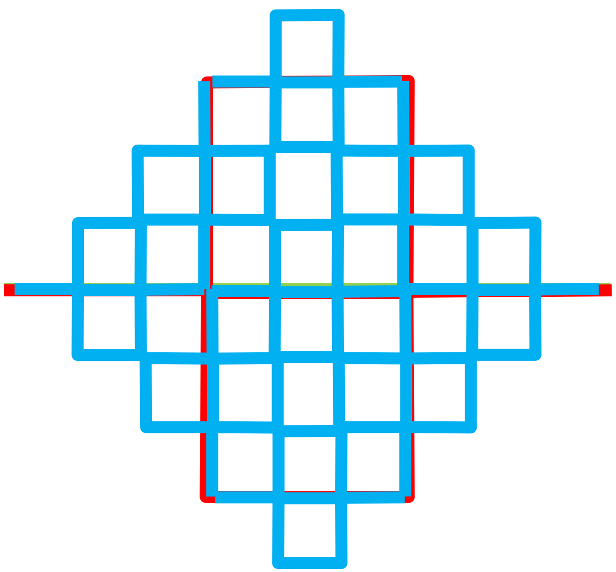
F→F+60F-120F+60F

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



F→F+90F-90F-90FF+90F+90F-90F

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



F→F+90F-90F-90F-90F+90F+90F+90F-90F

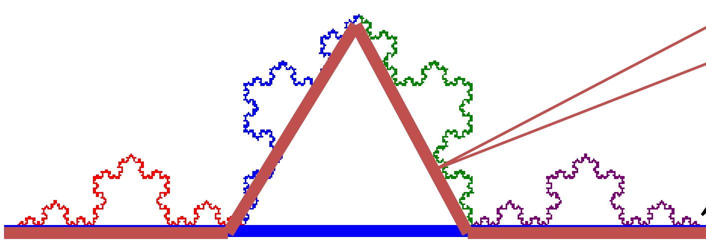
$r = 1/3, N = 9, D = \log(9)/\log(3) = 2$

Peano/Hilbert térkitöltő görbe

Nem önhasonló objektumok: vonalzó dimenzió

$$N = 1/r^D$$

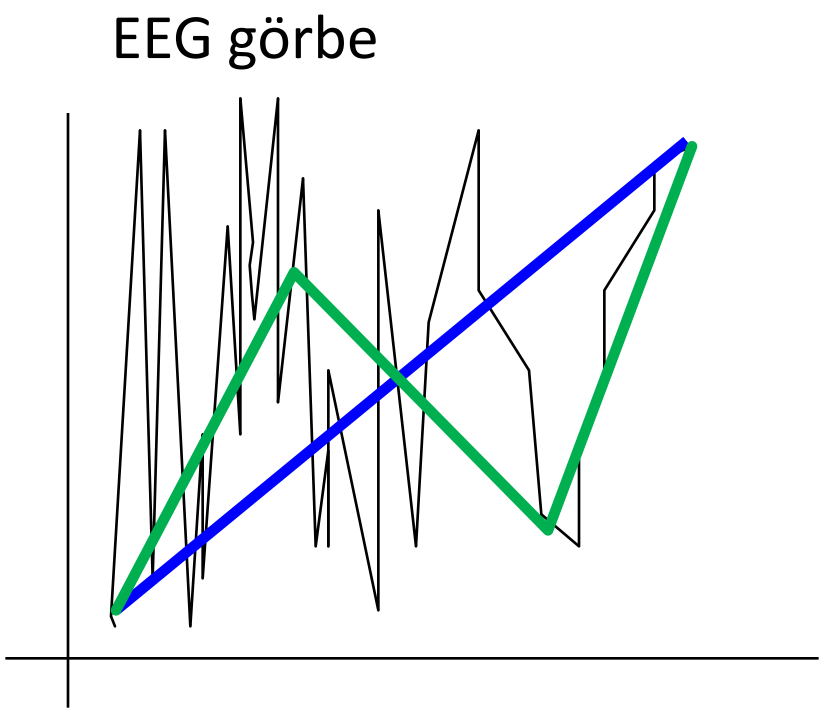
Vonalzó (l)	db
1	1
$r = 1/3$	$N = 4$
r^2	N^2
r^m	N^m

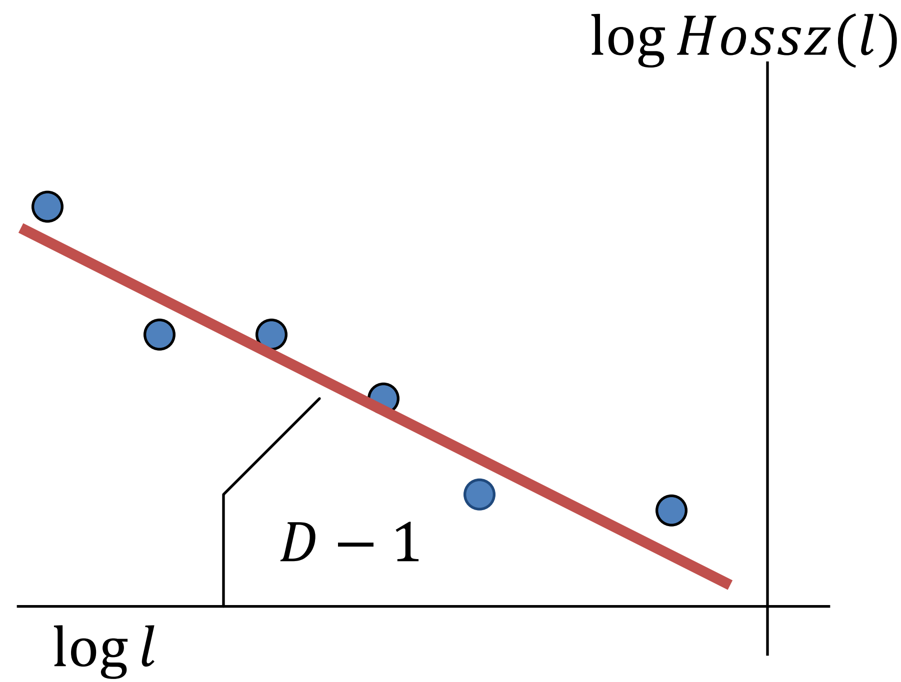


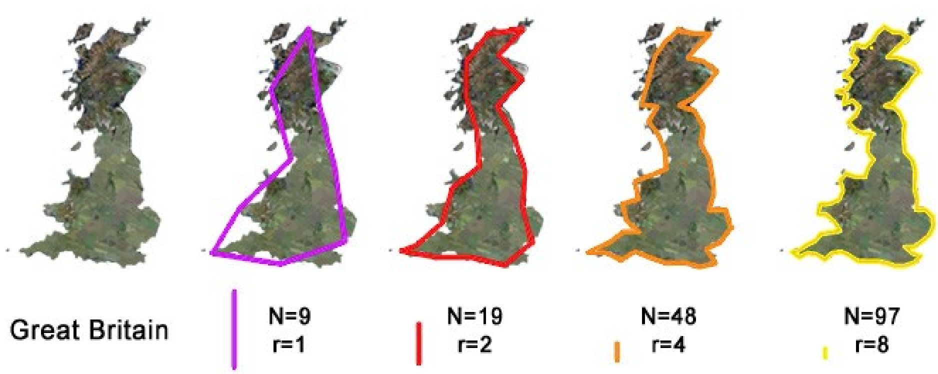
$$Hossz(l) = l\,db = l\,N^m = l(1/r^D)^m =$$
$$= l(1/r^m)^D = 1/l^{D-1}$$

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

Dimenziómérés = hosszmérés

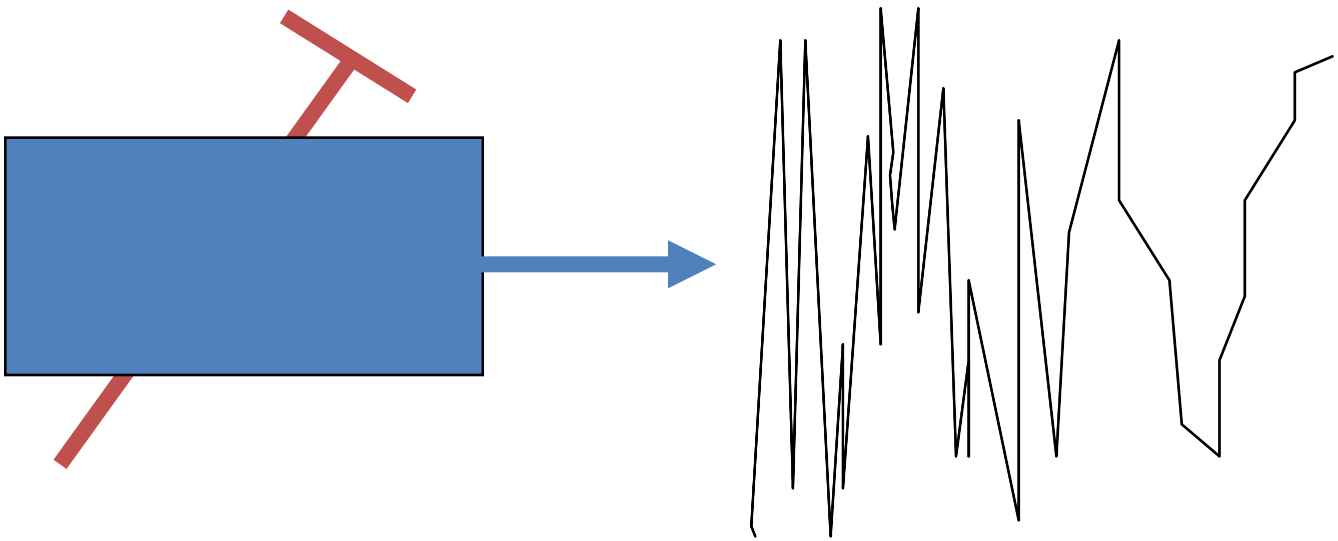






<u>Alkalmazás:</u>
Természetes objektumok elkülönítése és kategorizálása

Fraktálok előállítása

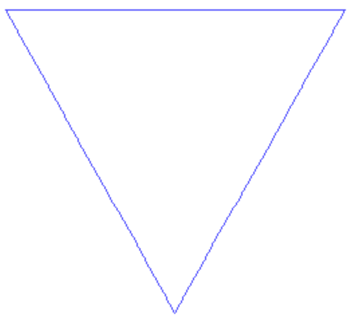


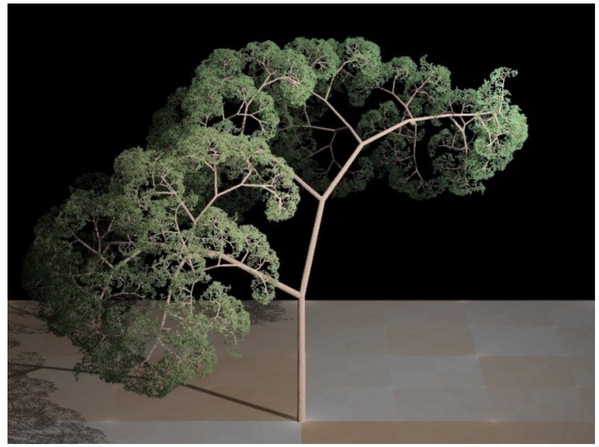
Matematikai gépek (algoritmusok):

- L-rendszerek
- Fraktális ($1/f$) zaj: Brown mozgás, Perlin zaj
- Kaotikus dinamikus rendszerek (véletlenszám generátor)

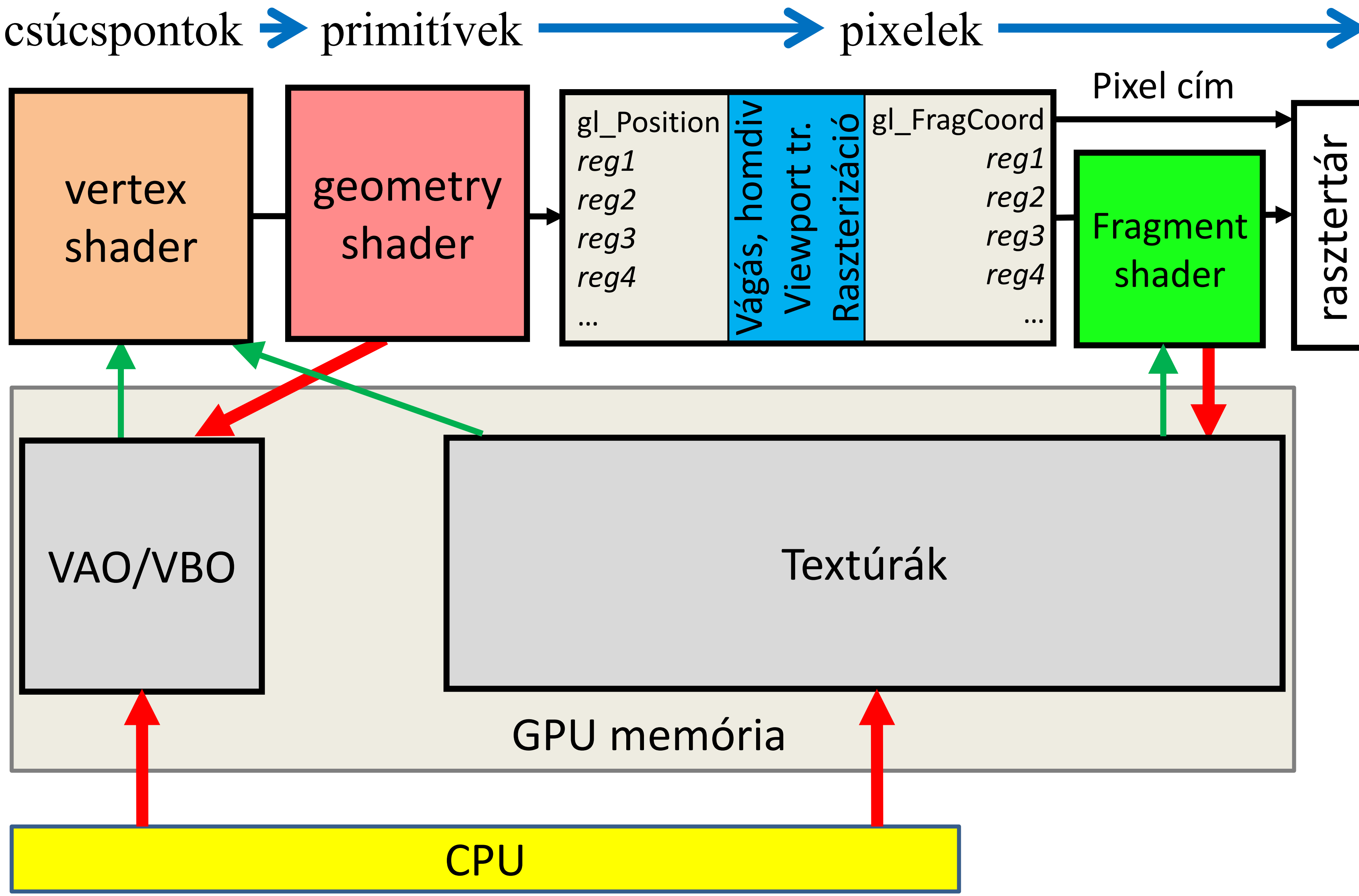
Lindenmayer rendszerek

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

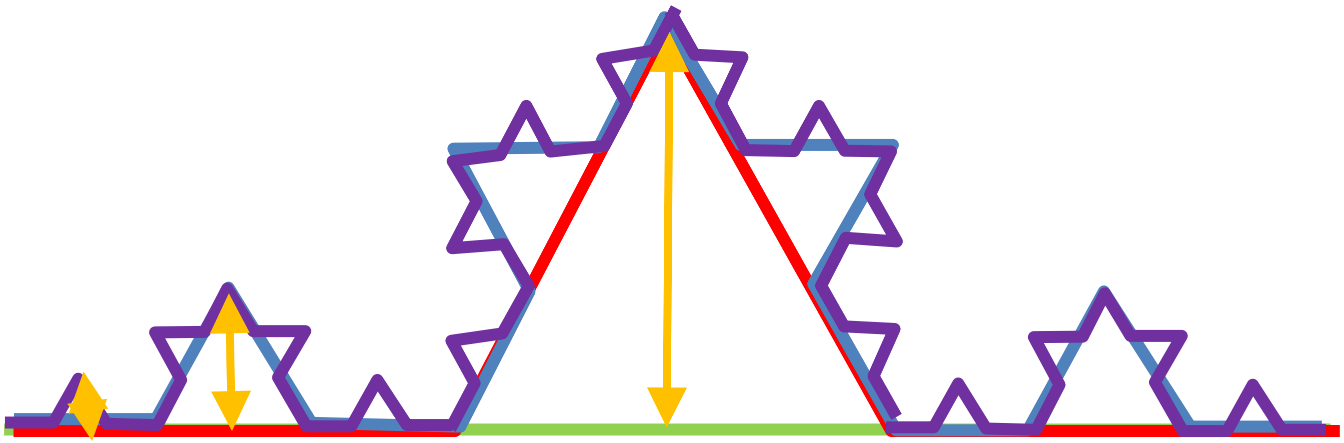




Geometry shader



Fraktális zaj

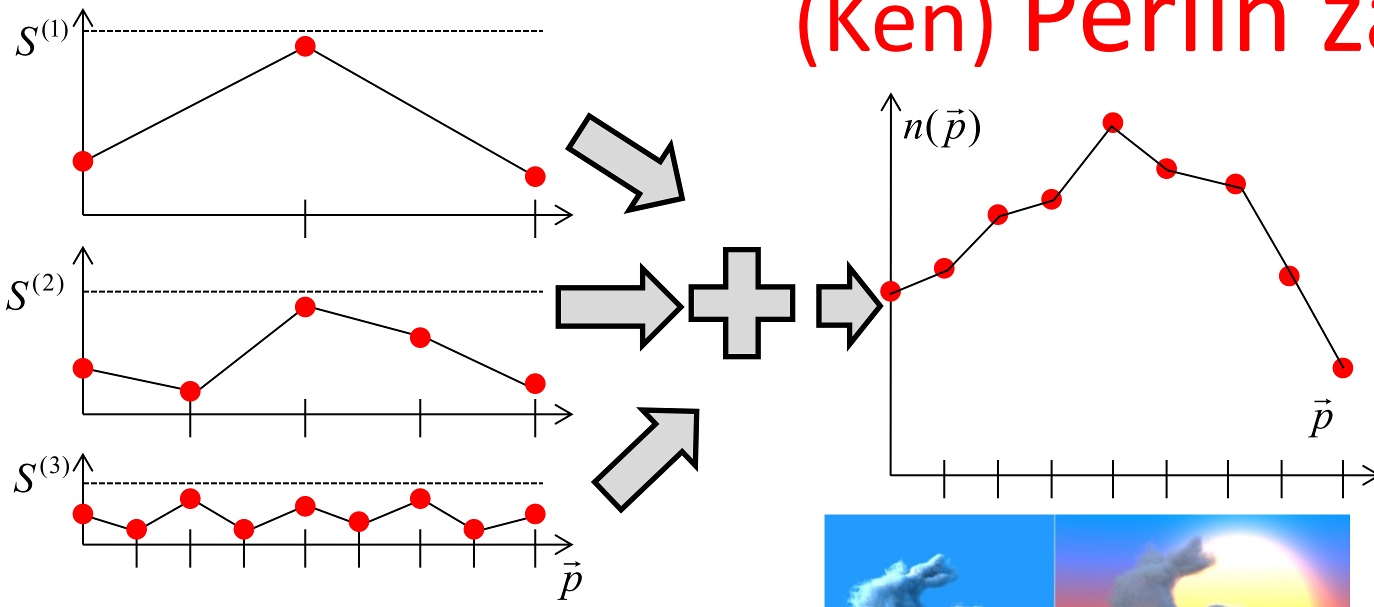


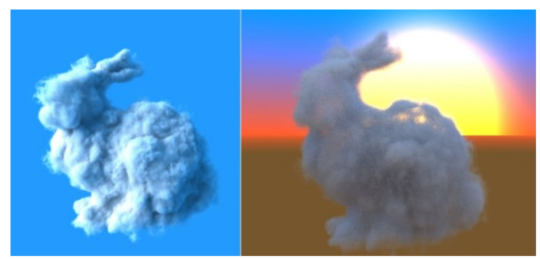
$\sigma/9$ $\sigma/3$ $\sigma = 1/(2\sqrt{3})$

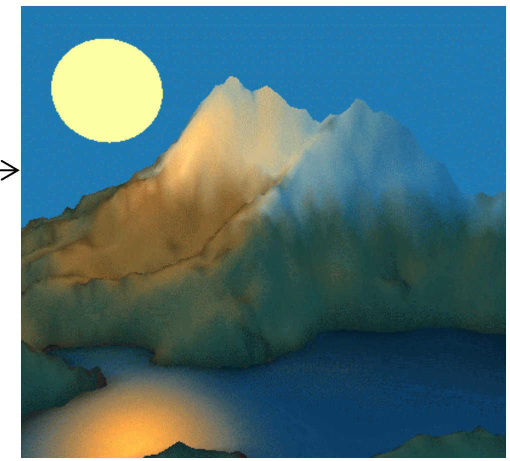
Skála	Ré[illegible]et
$N = 4$	
$N^2 = 16$	$\sigma/3$
	...
$N^m = 4^m$	$\sigma/3^{m-1}$

→ szórás

(Ken) Perlin zaj





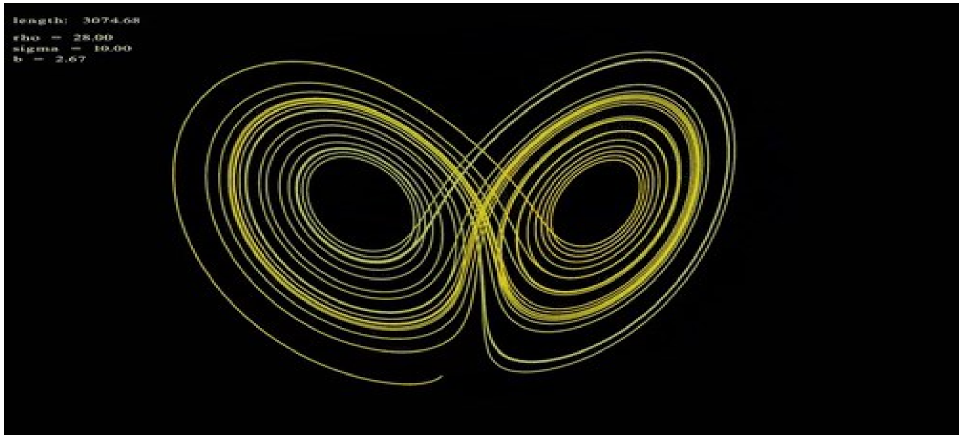


> *"Invention, it must be humbly admitted, does not consist in creating out of void but out of chaos."*
>
> *Mary Shelly*

Fraktálok és káosz

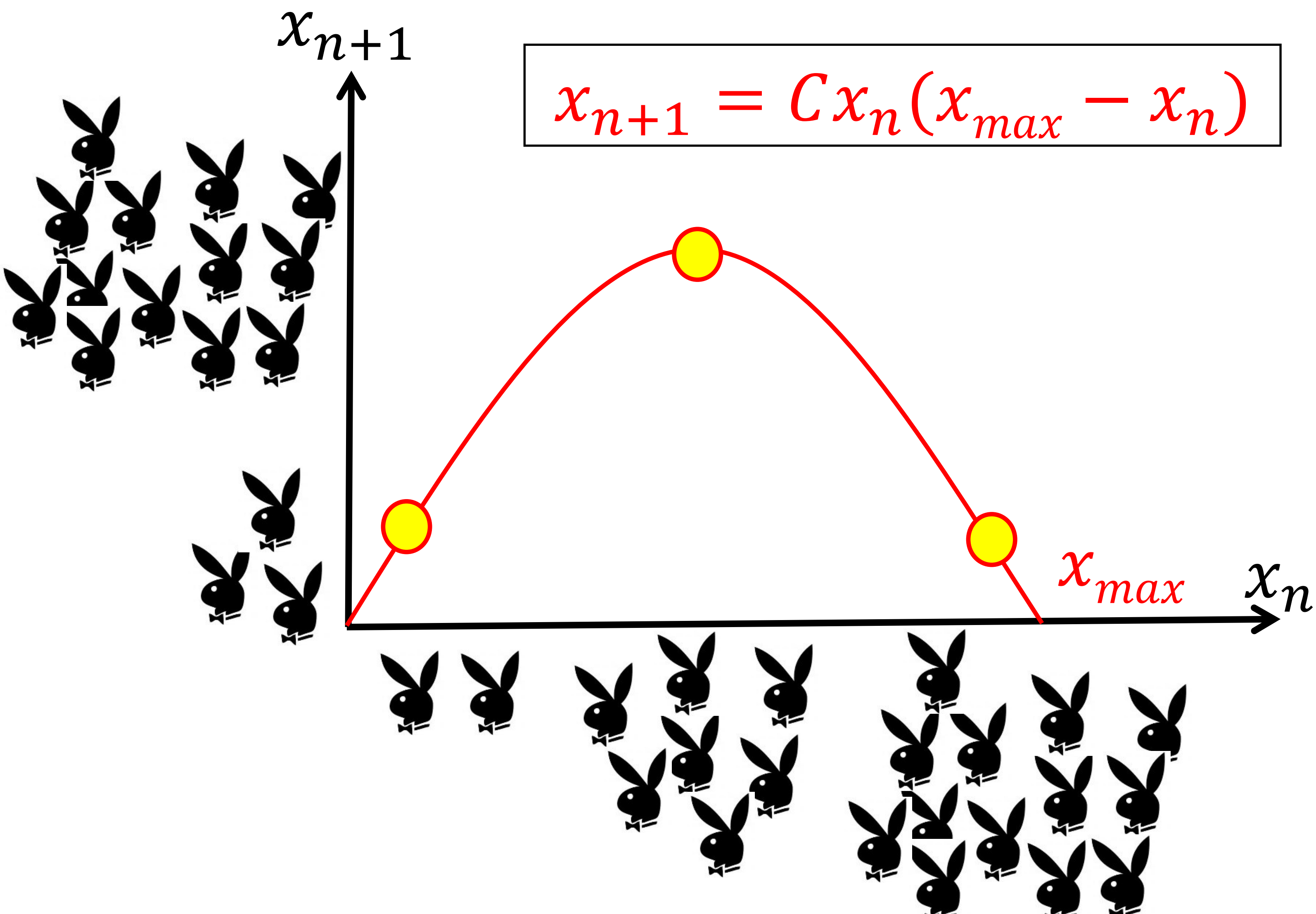
2. Káosz

Szirmay-Kalos László

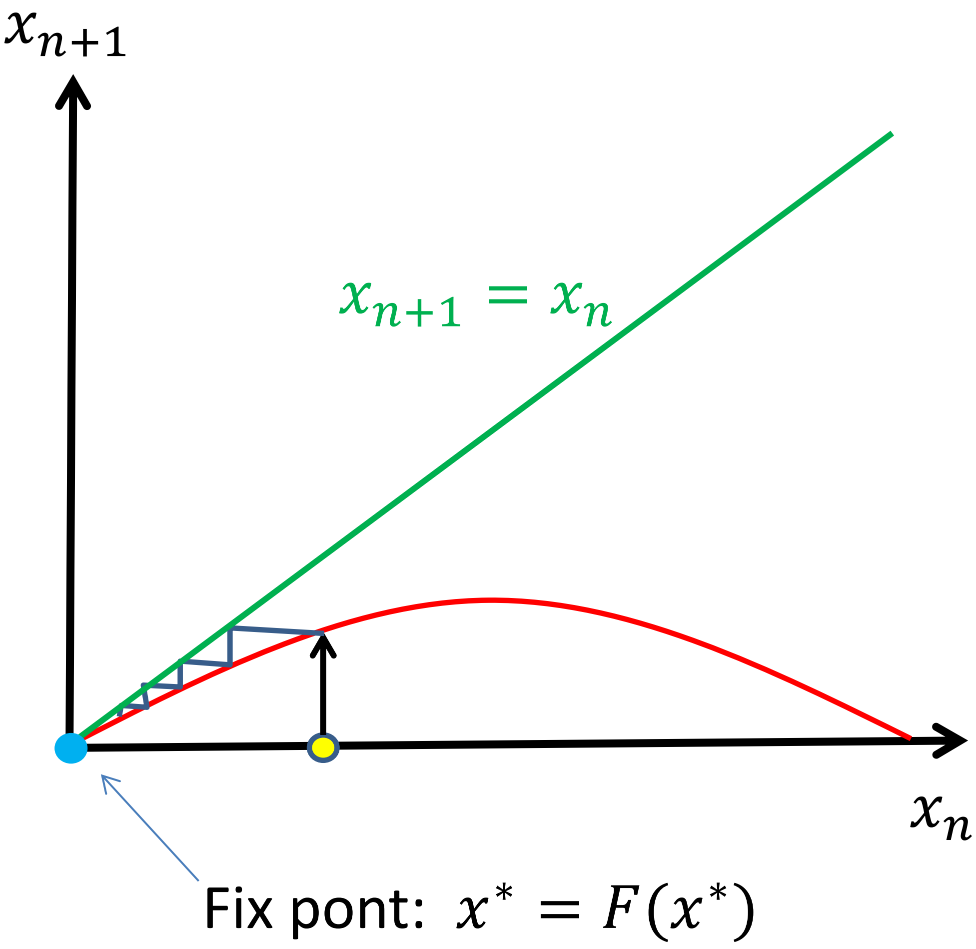


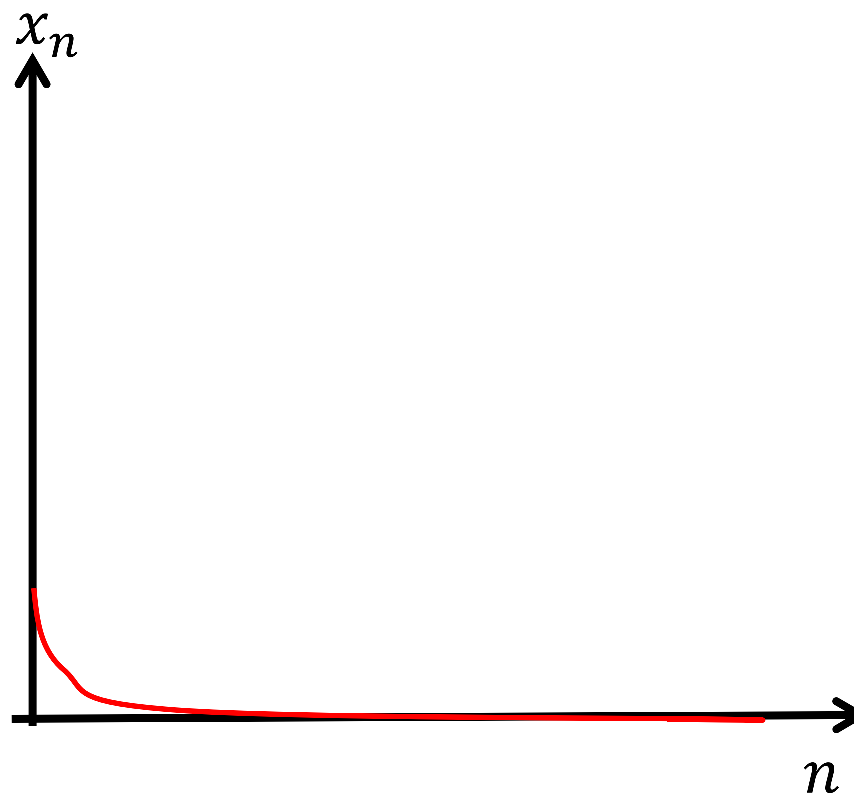
Nyulak szigete

$$x_{n+1} = Cx_n(x_{max} - x_n)$$

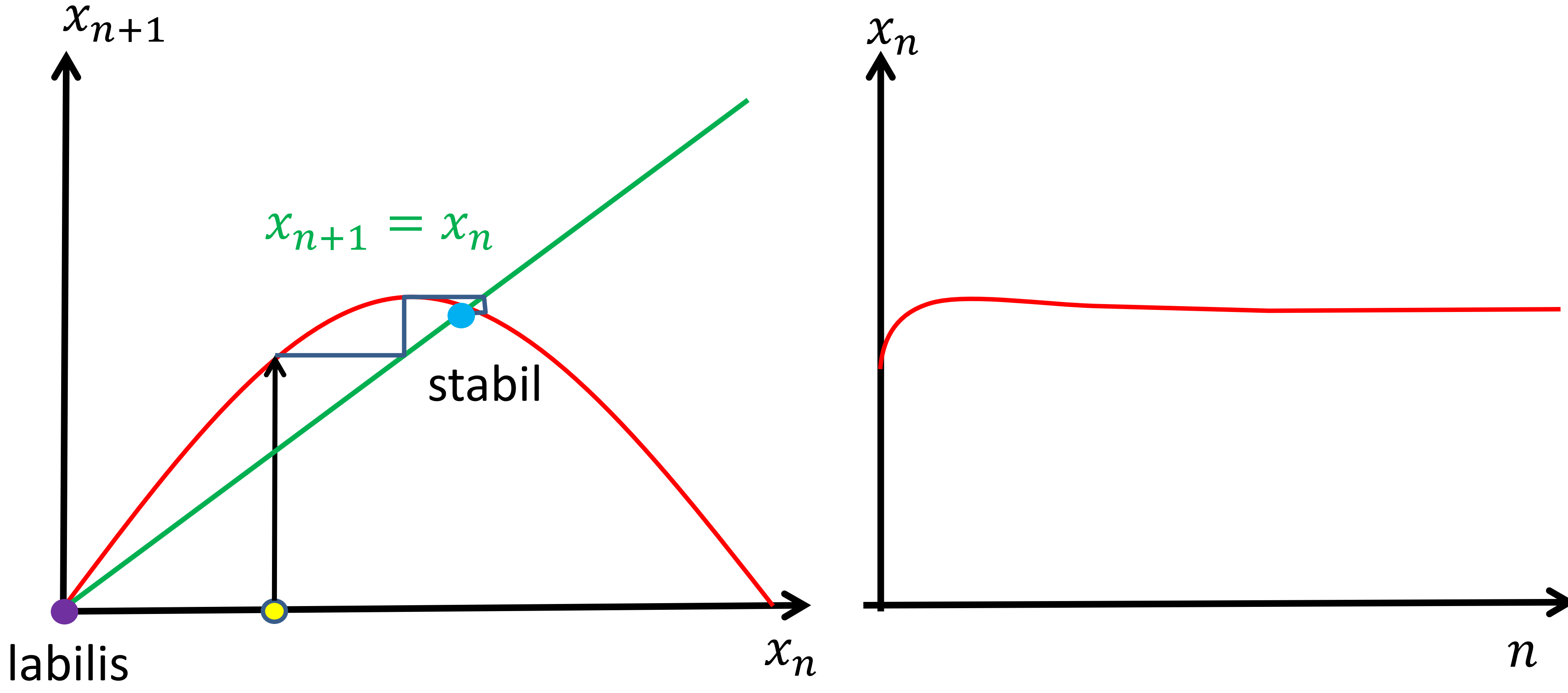


Nyulak kis *C* értékre: $x_{n+1} = Cx_n(x_{max} - x_n)$

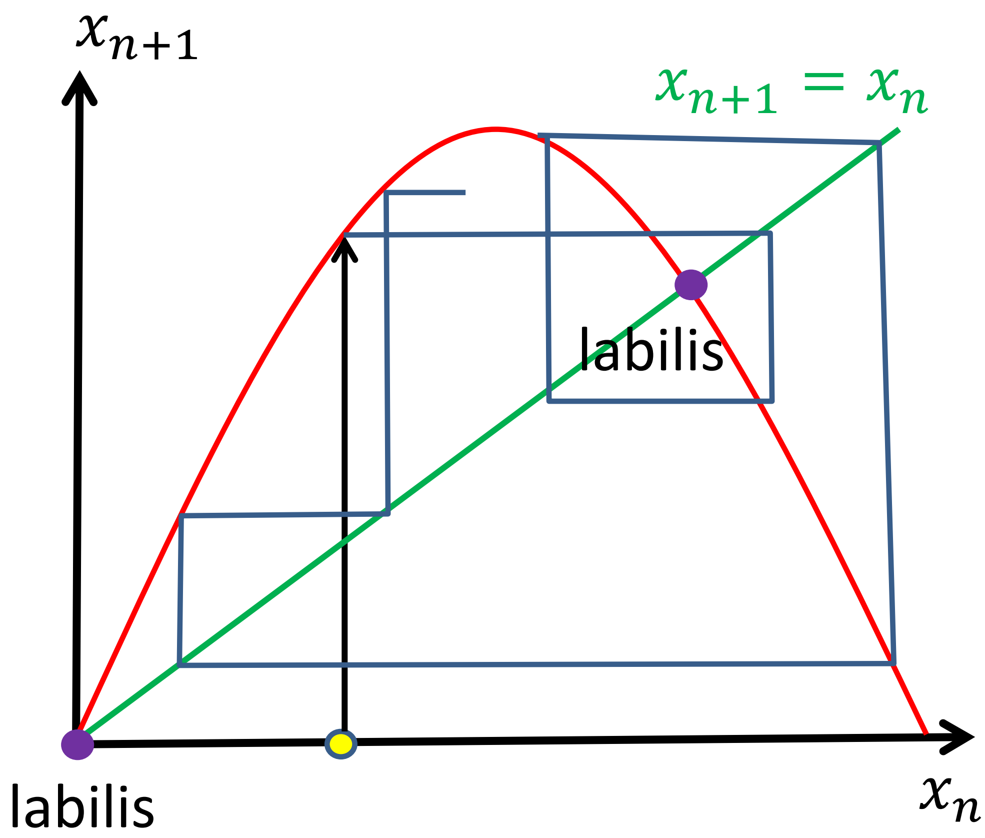


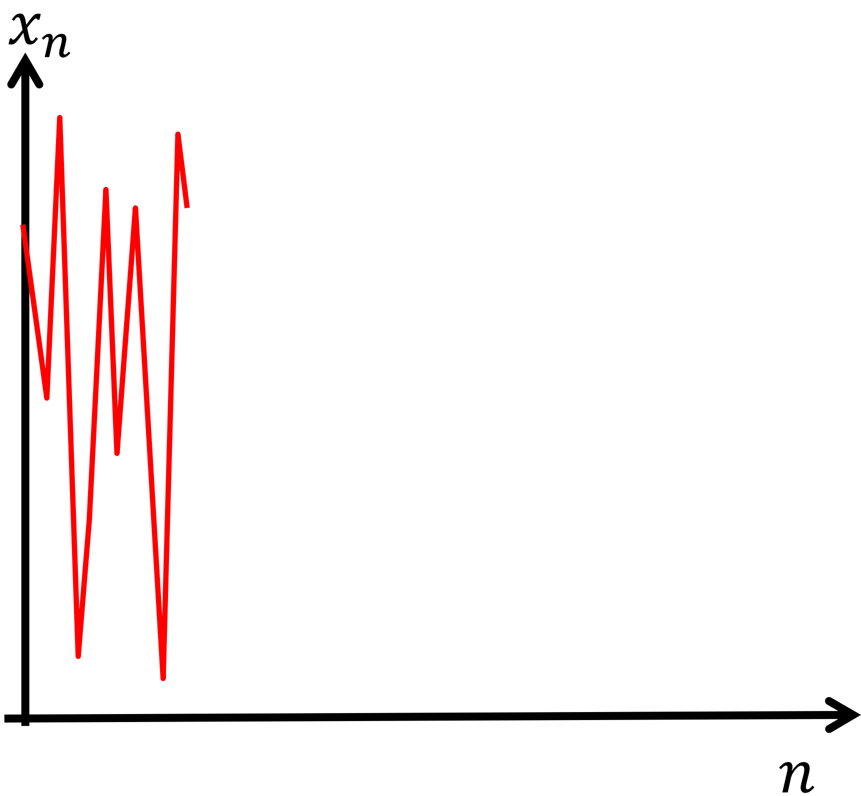


Közepes C értékre: $x_{n+1} = Cx_n(x_{max} - x_n)$

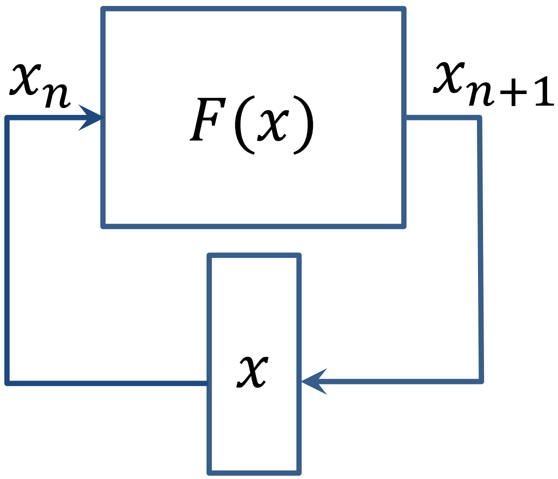


Nagy C értékre: $x_{n+1} = Cx_n(x_{max} - x_n)$





Iterált függvények, fix pont



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

$$\downarrow \quad \downarrow$$

$$\boxed{x^* = F(x^*)}$$

Fix pont

<u>Viselkedés (trajektória):</u>
- $F(x)$
- x_0

$$x_1 = F(x_0)$$
$$x_2 = F(x_1) = F(F(x_0))$$
$$x_3 = F(x_2) = F(F(F(x_0)))$$
$$\dots$$

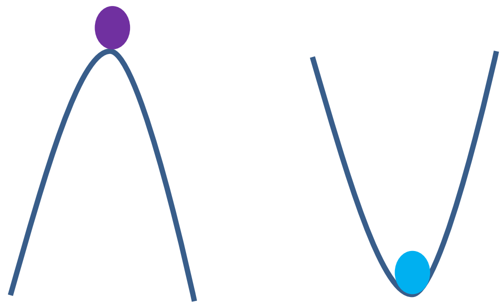
Iterált függvények, fix pont stabilitás

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

$$\downarrow \qquad \downarrow$$

$$\boxed{x^* = F(x^*)}$$

Fix pont

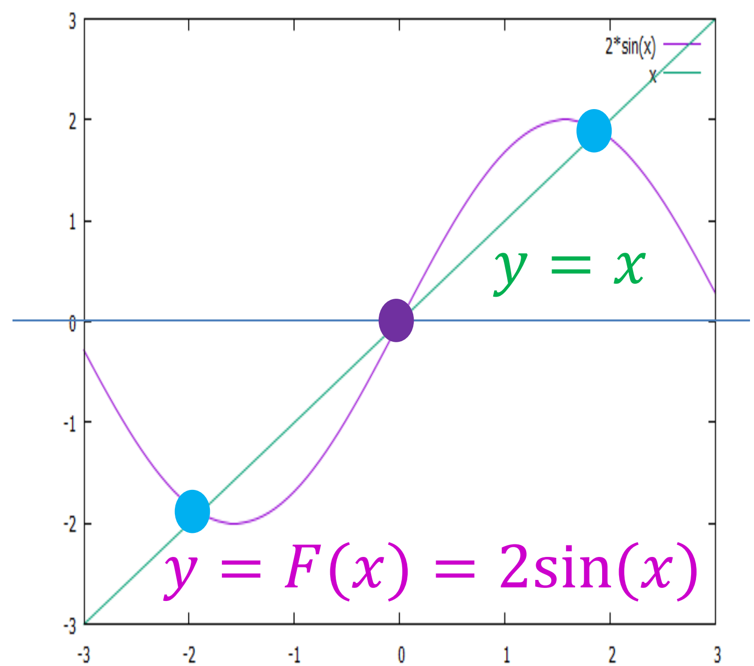


Labilis Stabil

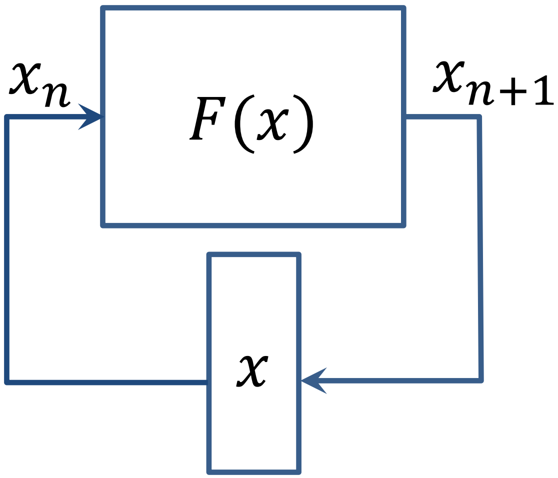
$$\cancel{x^*} + \Delta x_{n+1} = F(x^* + \Delta x_n)$$
$$\approx F(\cancel{x^*}) + F'(x^*)\cdot\Delta x_n$$

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

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?

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

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

$$\dots$$

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

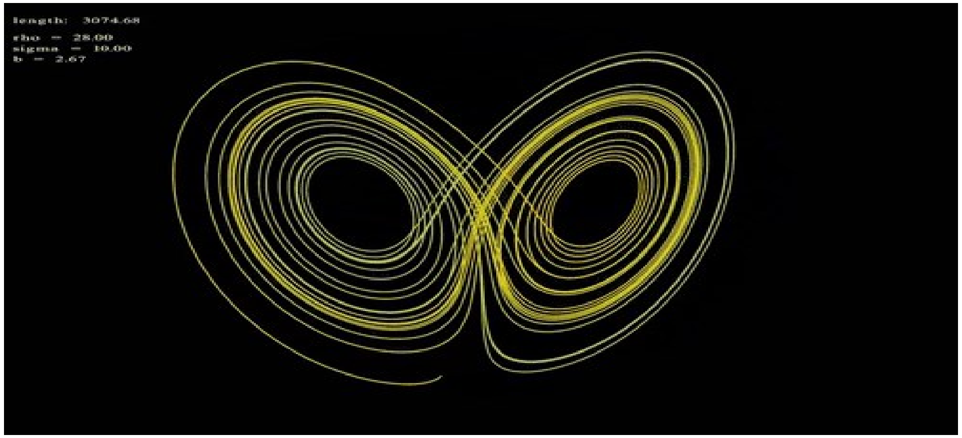
Akkumulált hiba:
Ha $|\tilde{F}'| > 1$, akkor divergál, hiába tart Δx_0 és ΔF zérushoz

Káosz

Edward Lorenz

<u>A rendszer determinisztikus, de</u>

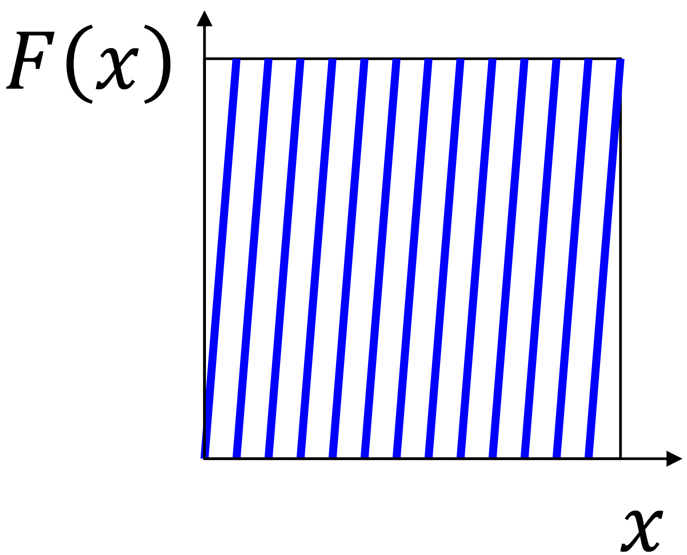
- Kezdeti állapot hatása eltűnik
  - Kis perturbáció nagyon eltérő viselkedéshez vezet
- Auto-korrelációs függvény zérushoz tart
  - Korábbi állapot gyengén befolyásolja a sokkal későbbit
  - Megjósolhatatlanság
- Teljesítmény sűrűség spektrum nem tart zérushoz
  - Nagy frekvencia



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



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

- $\{z\} = z$ tört része
- g nagy

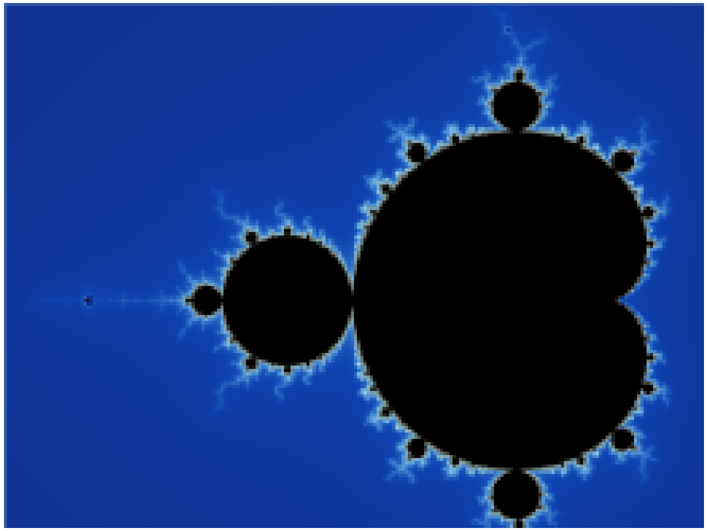
“There's no sense in being precise when you don't even know what you're talking about.”

Neumann János

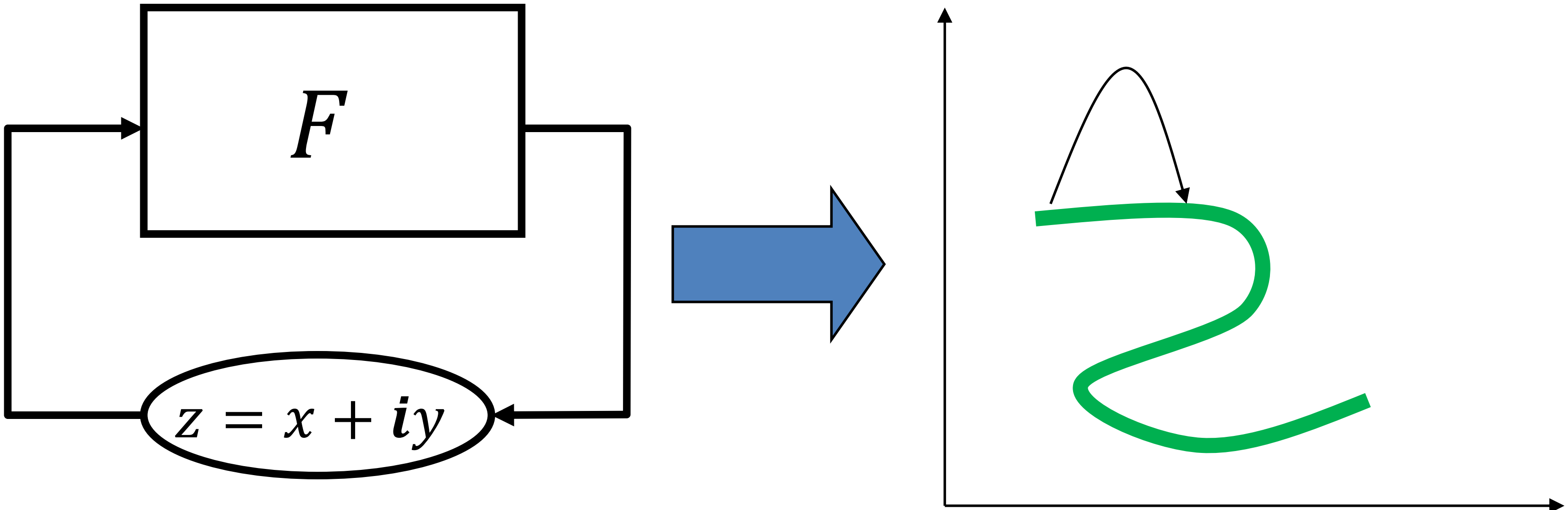
Fraktálok és káosz

3. Kaotikus rendszerek a síkon

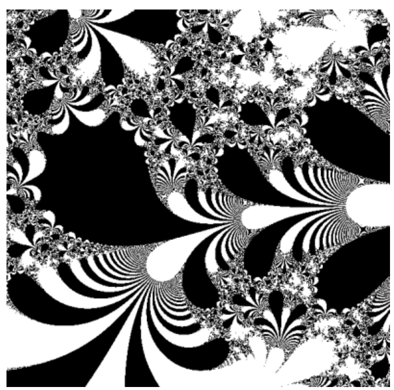
Szirmay-Kalos László



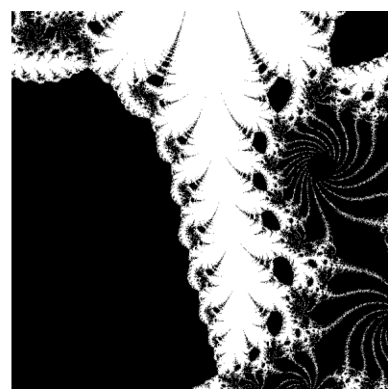
Kaotikus rendszerek a síkon



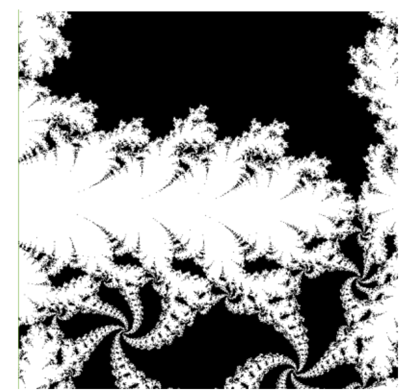
$F(z) = z^2 + c$



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

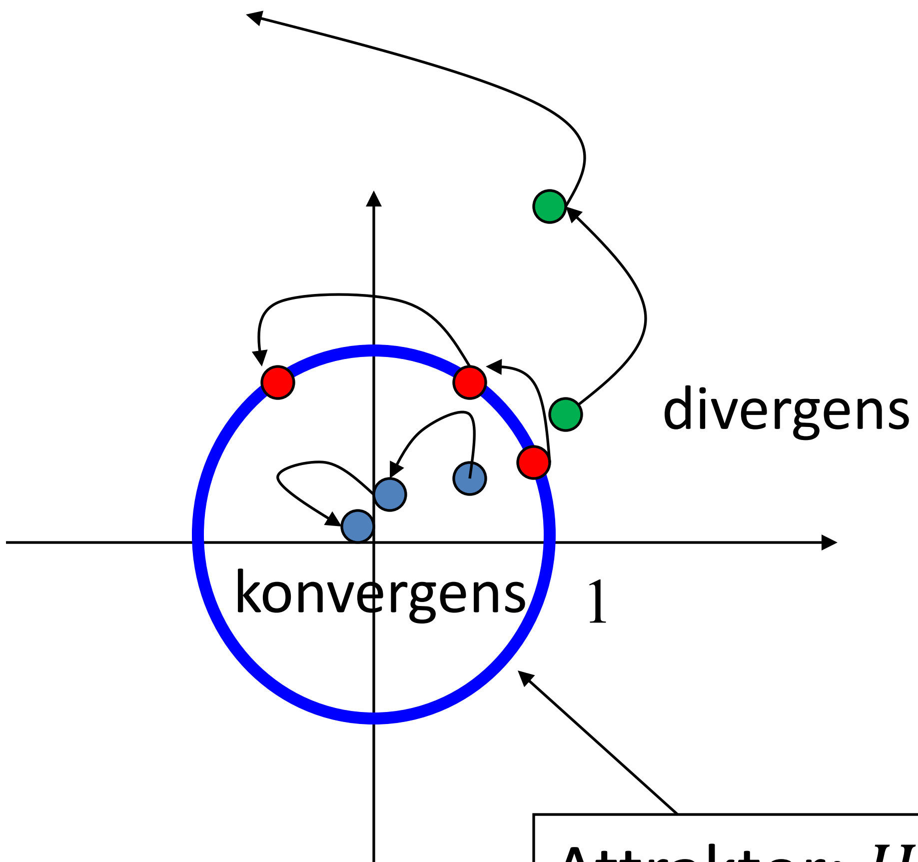


$F(z) = \cos(z + c)$



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

$$F\colon z \to z^2$$

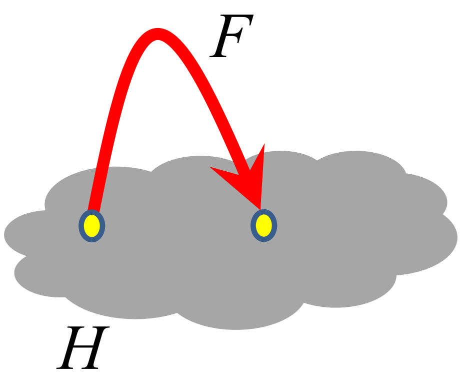


$$z = re^{i\varphi}$$

$$r \to r^2$$

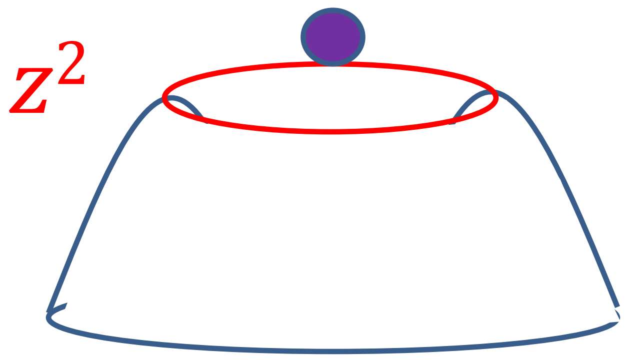
$$\varphi \to 2\varphi$$

Attraktor: $H = F(H)$

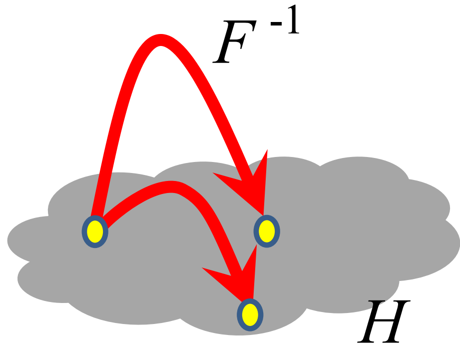


Attraktor felrajzolása

- 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
  - $z_{n+1} = {z_n}^2$ <u>attraktora labilis</u>



Inverz iterációs módszer

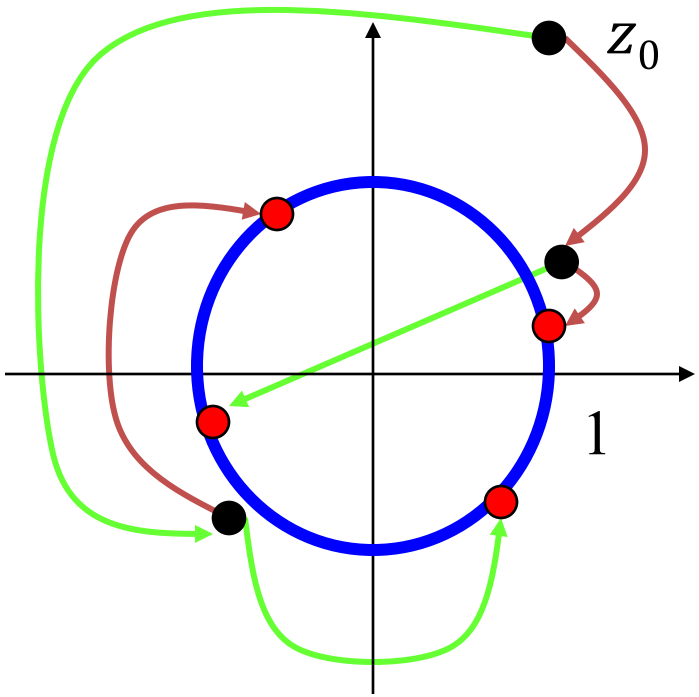


$$H = F(H) \quad \rightarrow \quad H = F^{-1}(H)$$

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

$$z_{n+1} = \pm\sqrt{z_n}$$

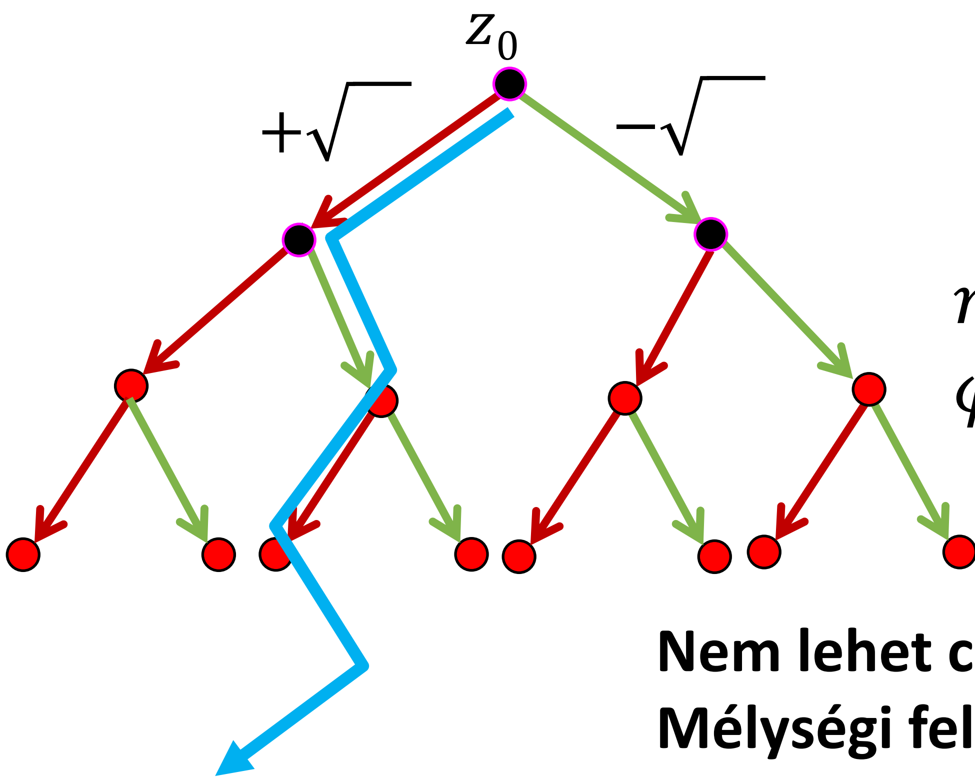
$$r_{n+1} = \sqrt{r_n}$$
$$\varphi_{n+1} = \varphi_n/2 + \pi\{0|1\}$$



Ha n nagy:

$$r_n = \sqrt[2^n]{r_0} \approx 1$$
$$\varphi_n = \frac{\varphi_0}{2^n} + \pi\{0|1\}.\{0|1\}\{0|1\}\ldots$$
$$\approx \underset{n}{\pi\{0|1\}}.\underset{n-1}{\{0|1\}}\underset{n-2}{\{0|1\}}\ldots$$

Többértékű leképzés: Bolyongás



$$r_n \approx 1$$

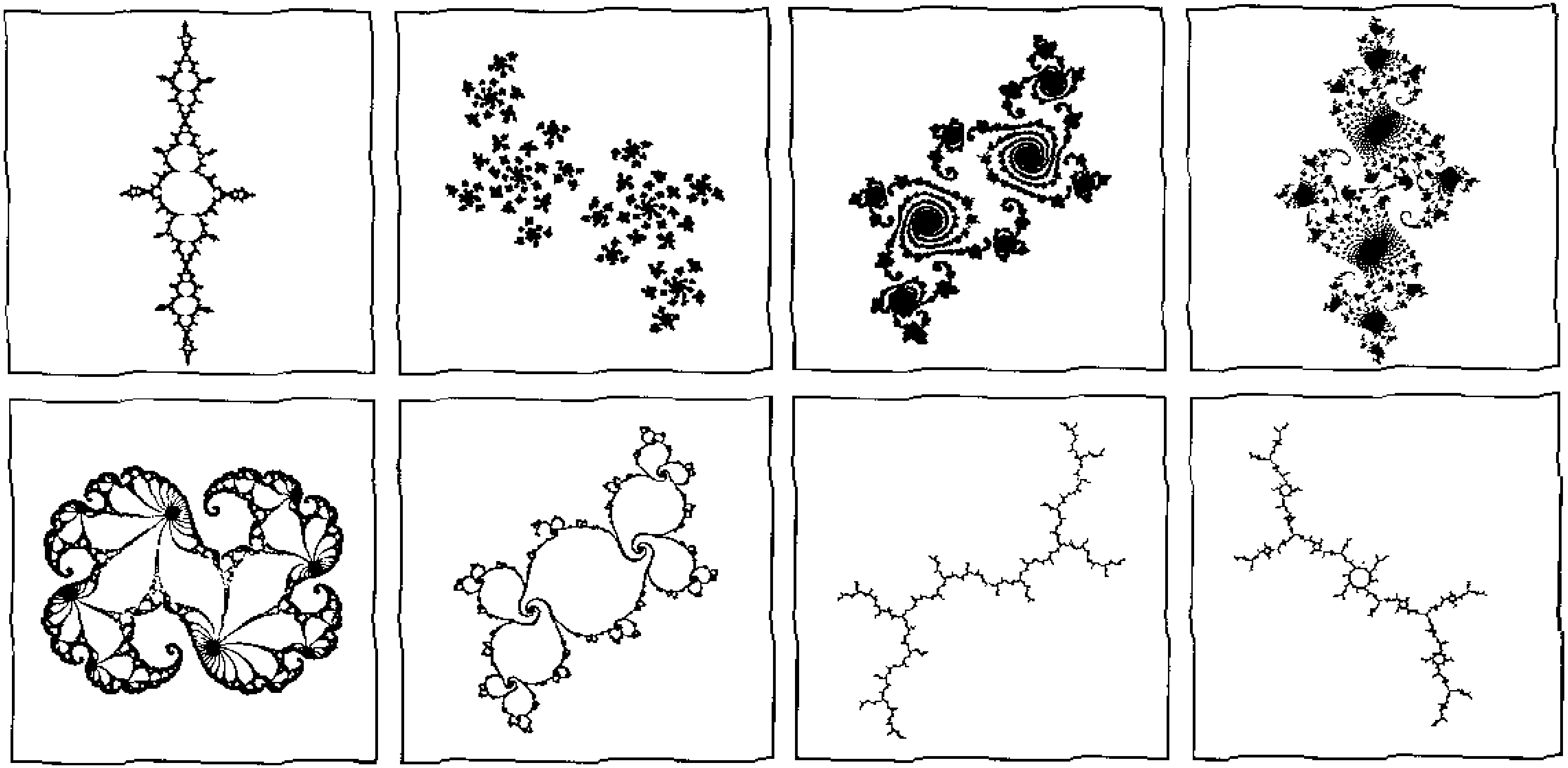
$$\varphi_n \approx \pi\underset{n}{\{0|1\}}.\underset{n-1}{\{0|1\}}\underset{n-2}{\{0|1\}}\ldots$$

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

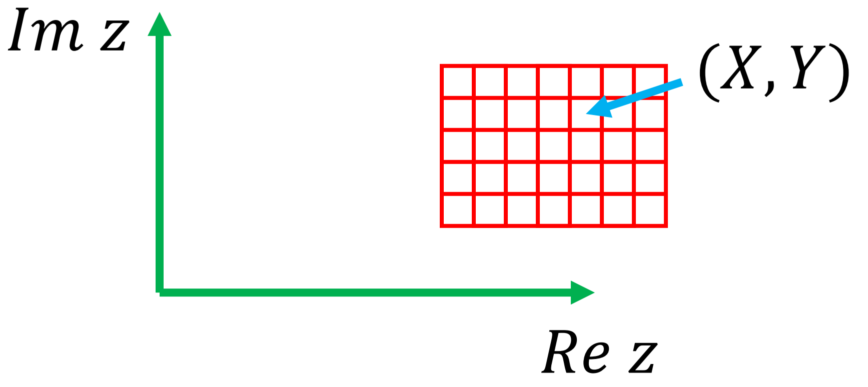
- Csak a $+$: $\varphi_n \approx 0.00\ldots\cdot\pi = 0$
- Csak a $-$: $\varphi_n \approx 1.1\ldots\cdot\pi = 2\pi \sim 0$
- Felváltva: $\varphi_{2n} \approx 1.010\ldots\cdot\pi = \frac{4\pi}{3}$; $\varphi_{2n+1} \approx 0.1010\ldots\cdot\pi = \frac{2\pi}{3}$
- **Véletlenszerűen!**

(Gaston) Julia halmaz

$$F\colon z \to z^2 + c$$



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 = z^2 + c
            image[Y][X] = (|z| < infinity) ? black : white;
        }
    }
}
```

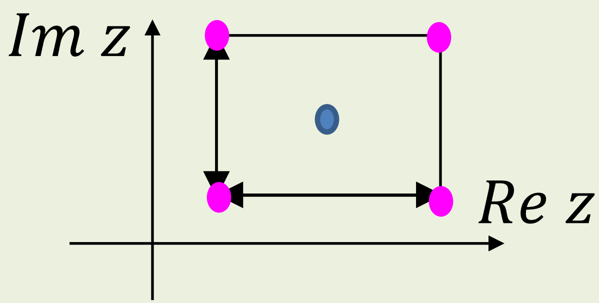
GPU implementáció

CPU program

```
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);
```

Vertex shader

```
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;
}
```

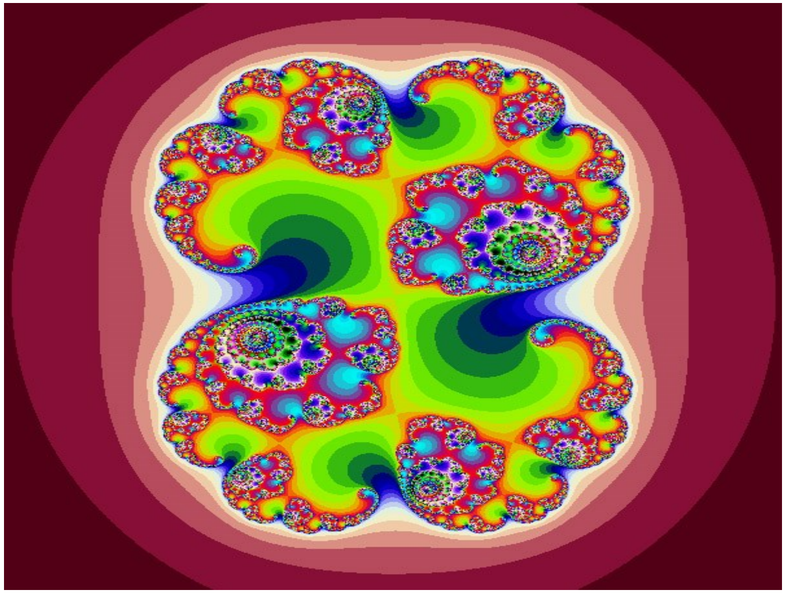


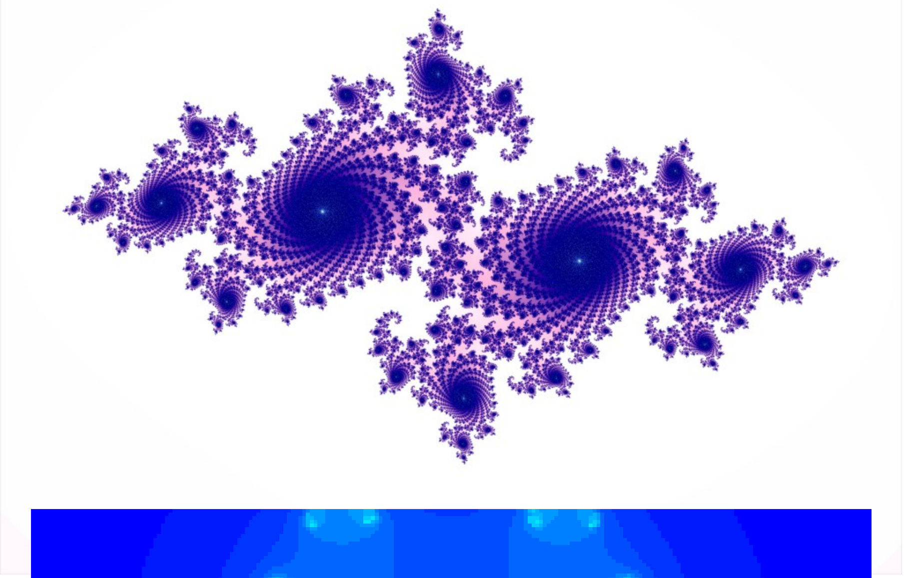
Fragment shader

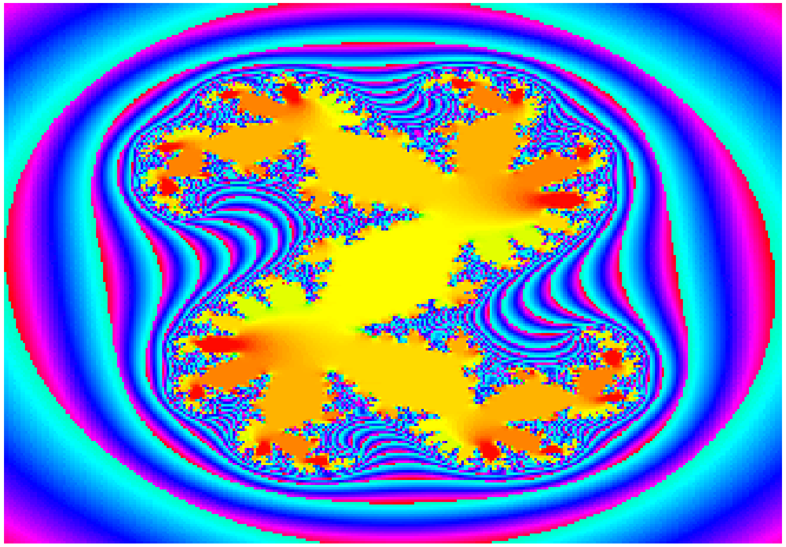
```
uniform vec2 c;
in vec2 z0;
out vec4 fragCol;

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);
}
```

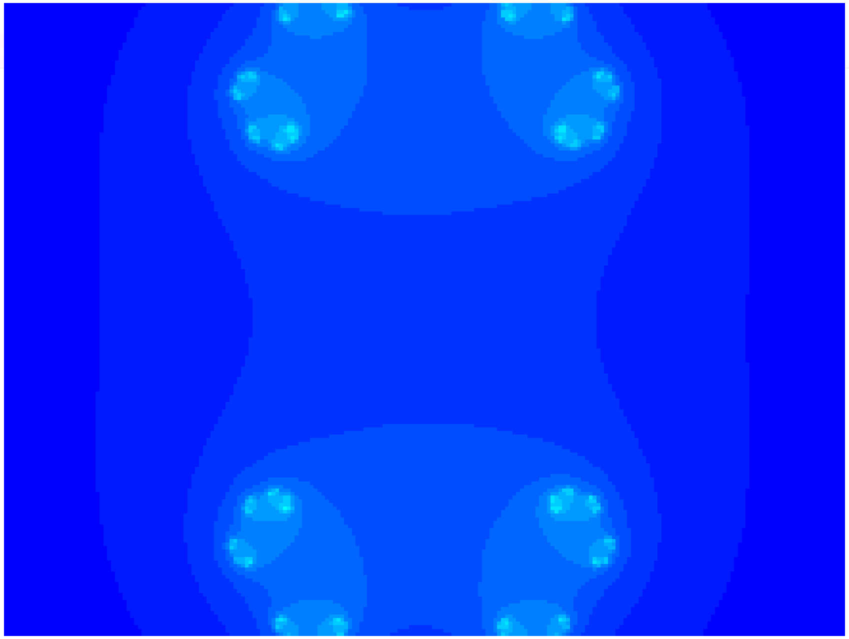
$$z = z^2 + c$$

Kitöltött Julia halmaz: kép

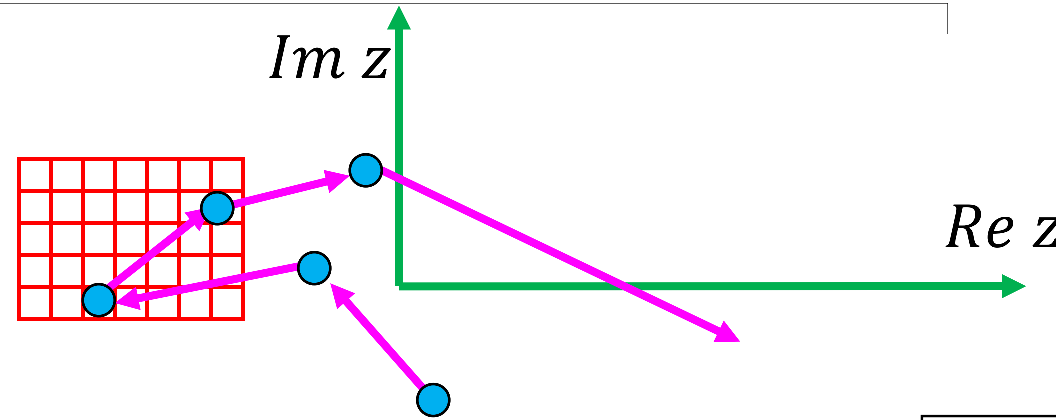








Julia halmaz inverz iterációval




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

Kezdeti z érték:

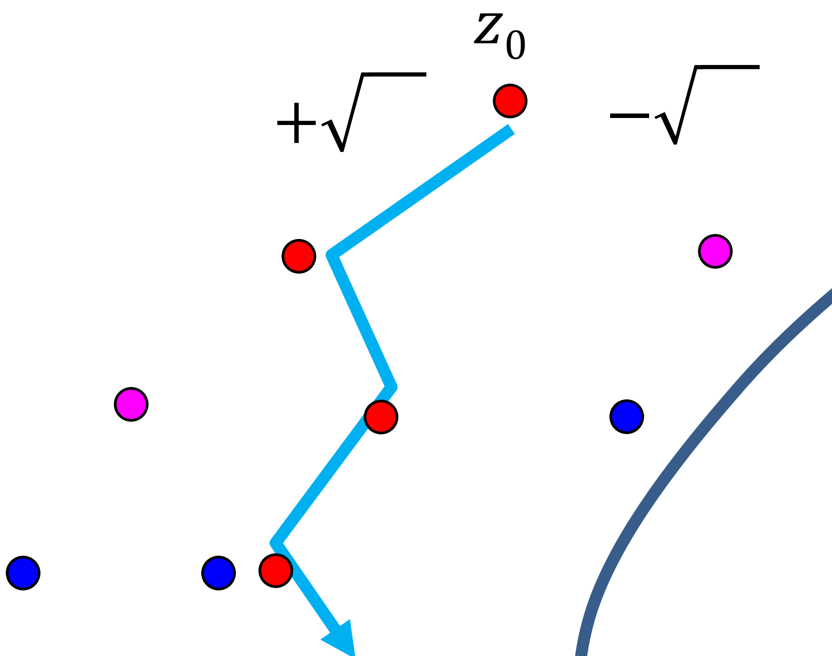
$z = z^2 + c$ gyöke

Inverz:

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

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

GPU implementáció

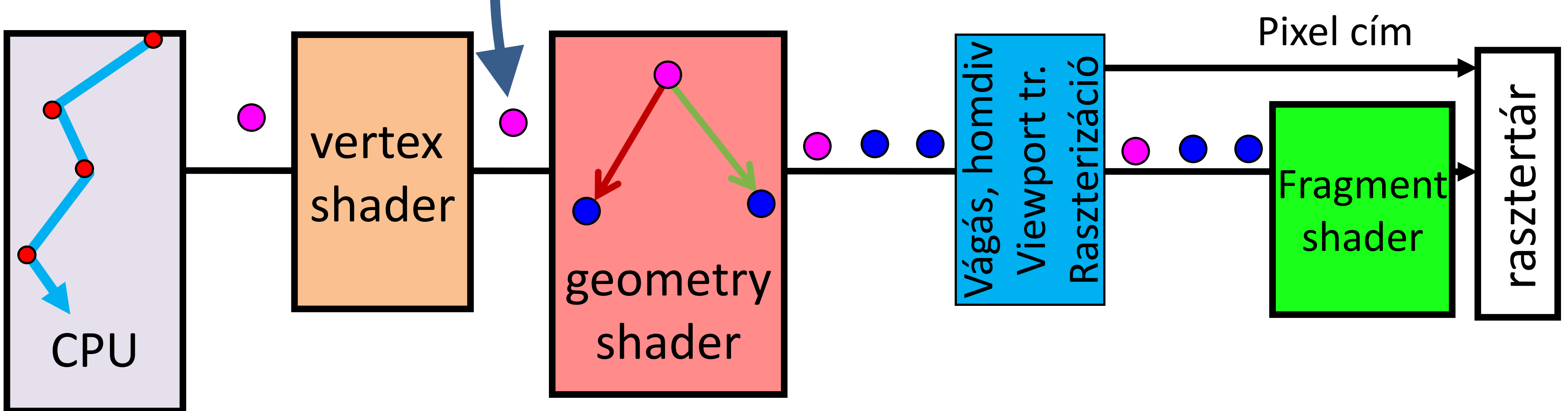


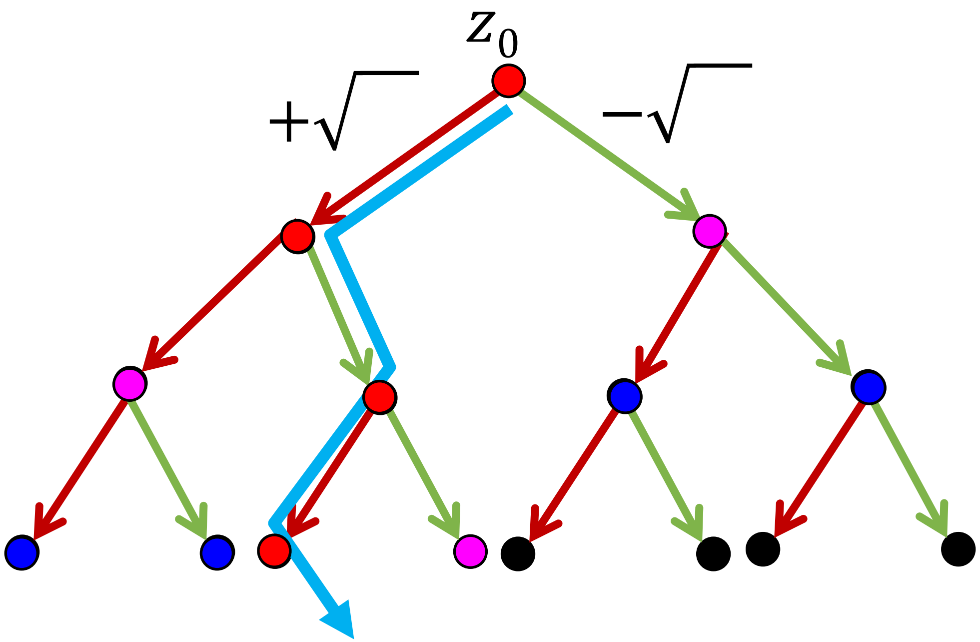
Vertex shader

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

Fragment shader

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





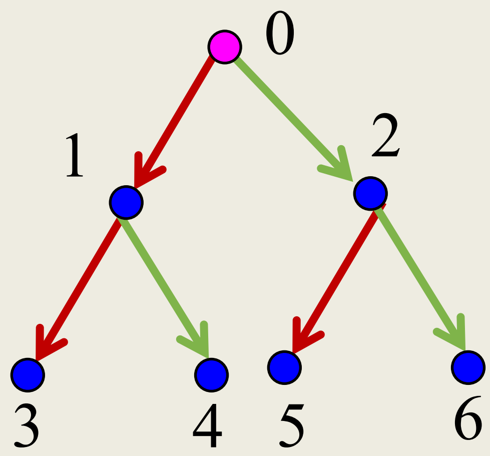
CPU program

Kezdeti *z* érték:
$z^2 = z - c$ **gyöke**

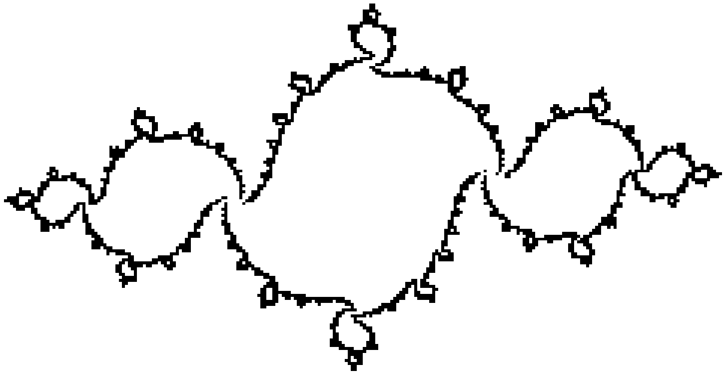
```
vec2 z = vec2(0.5, 0) + sqrtComplex(vec2(0.25 - c.x, -c.y));
for (int p = 0; p < nPackets; p++) {
   vec2 vtx[nSeeds];
   for (int i = 0; i < nSeeds; i++) {
       z = sqrtComplex(z - c) * (rand() & 1 ? 1 : -1);
       vtx[i] = -z;
   }
   glBufferData(GL_ARRAY_BUFFER, sizeof(vtx), vtx,
        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

Geometry shader

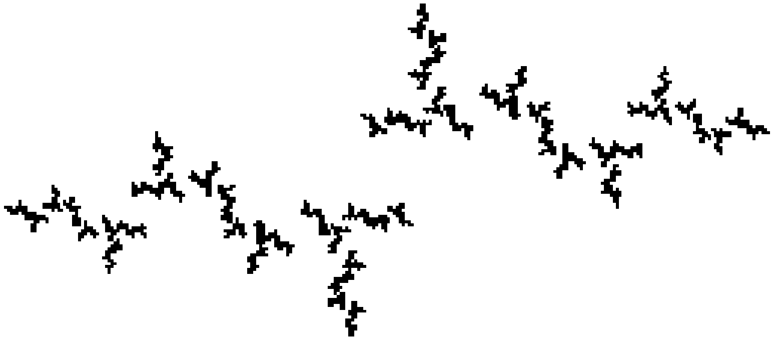
```
uniform vec2 cameraCenter, cameraSize, c;

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



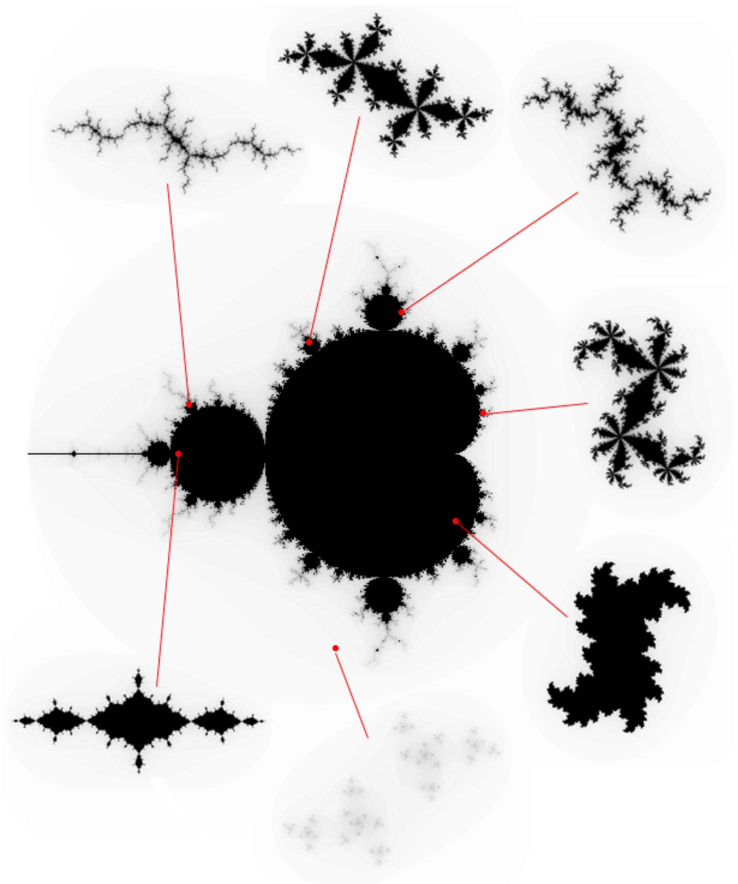
Összefüggő

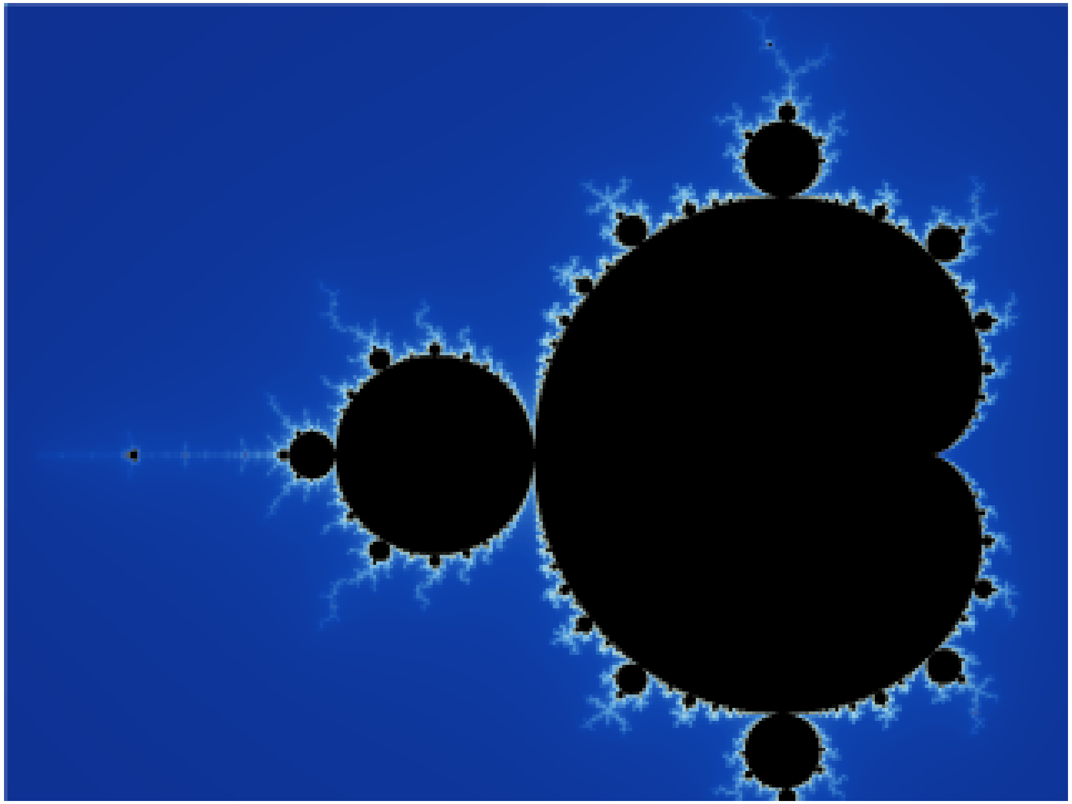


Nem összefüggő,
Cantor féle halmaz

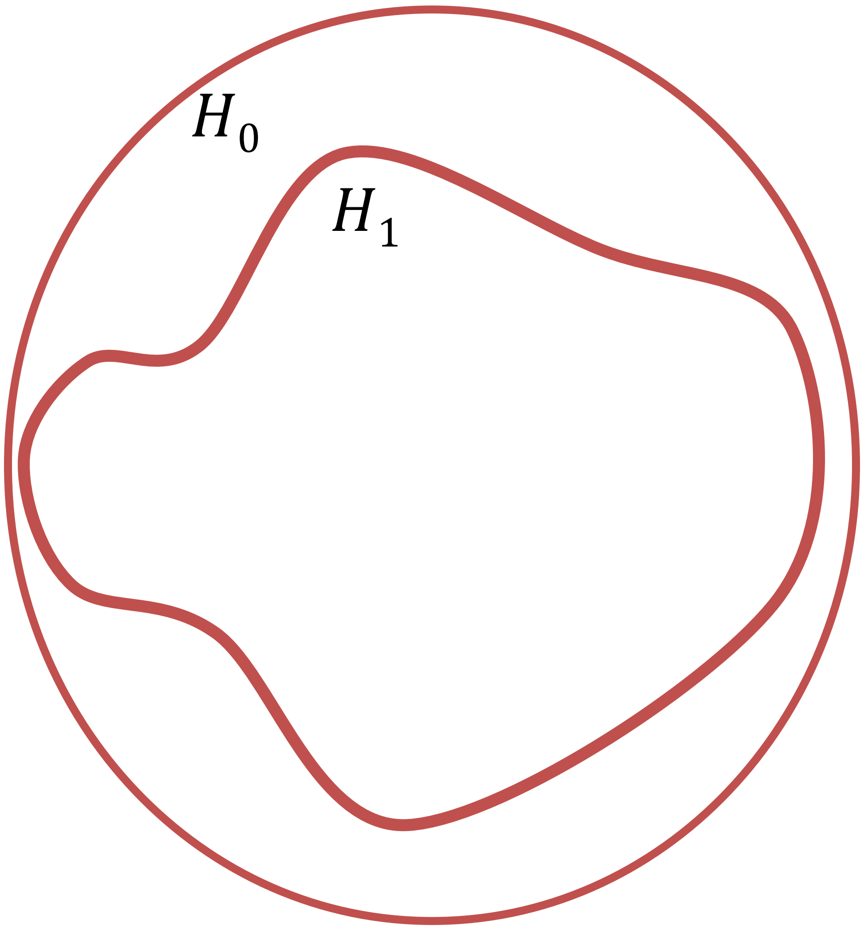
(Benoit) Mandelbrot halmaz

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



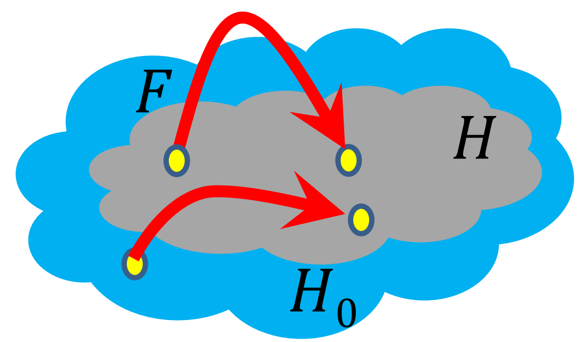


Julia halmaz összefüggősége



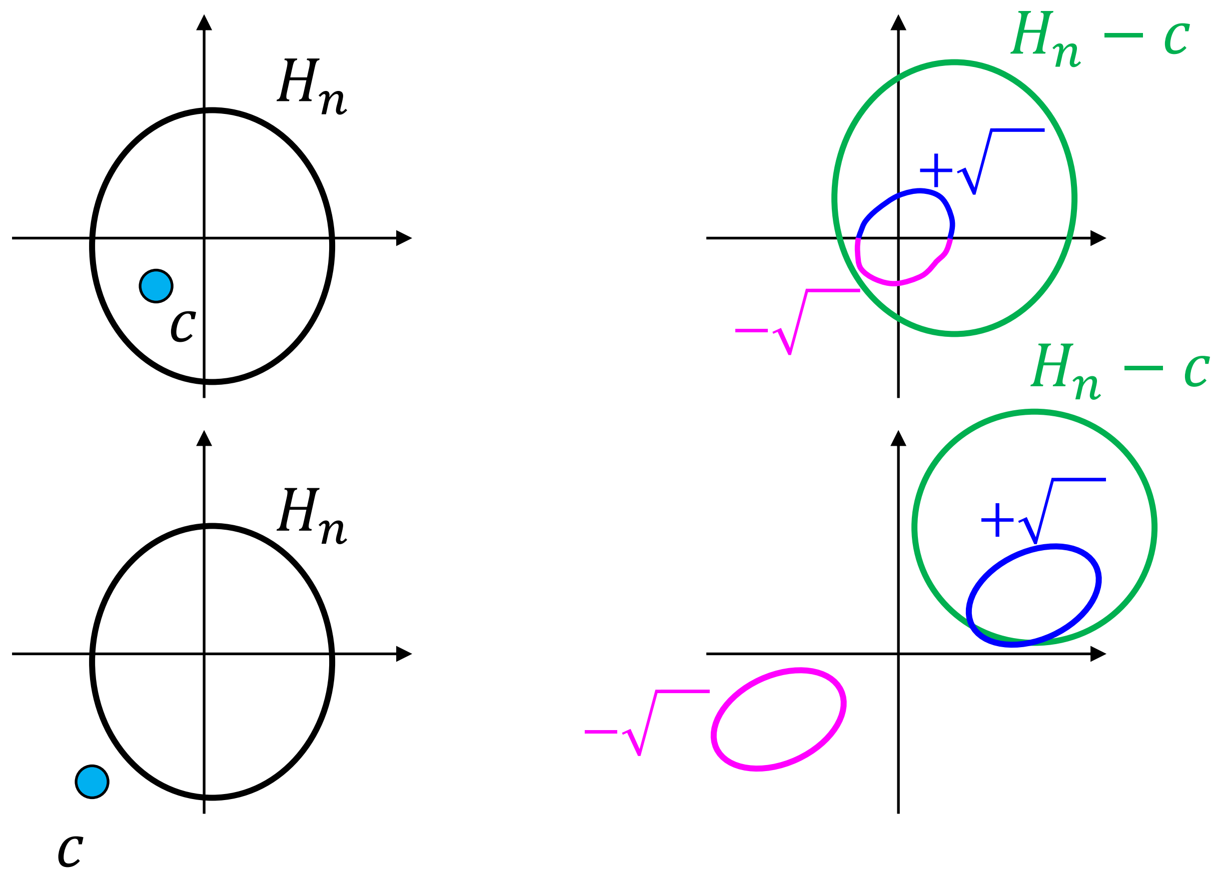
$$F(z) = \pm\sqrt{z-c}$$

$$\boxed{H = F(H)}$$



$$H_1 = F(H_0)$$
$$H_2 = F(H_1)$$
$$H_3 = F(H_2)$$
$$\ldots$$

Julia halmaz összefüggősége



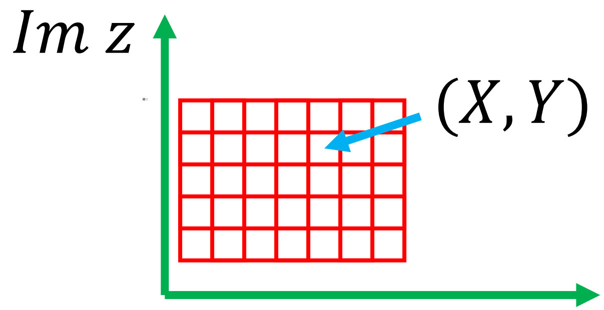
$$H_{n+1} = \pm\sqrt{H_n - c}$$

(Benoit) Mandelbrot halmaz

Azon *c* komplex számok halmaza, amelyekre:

- A $z \rightarrow z^2 + c$ Julia halmaza összefüggő.
- A c a Julia halmaz attraktorában vagy konvergens tartományában van.
- A $z_{n+1} = {z_n}^2 + c$ 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 = z^2 + c
            image[Y][X] = (|z| < infinity) ? black : white;
        }
    }
}
```

$|z| < 2$

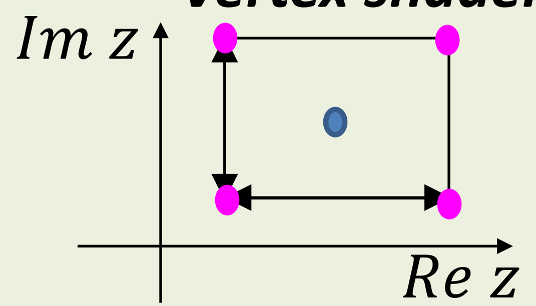
GPU implementáció

CPU program

```
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);
```

Vertex shader

```
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;
}
```

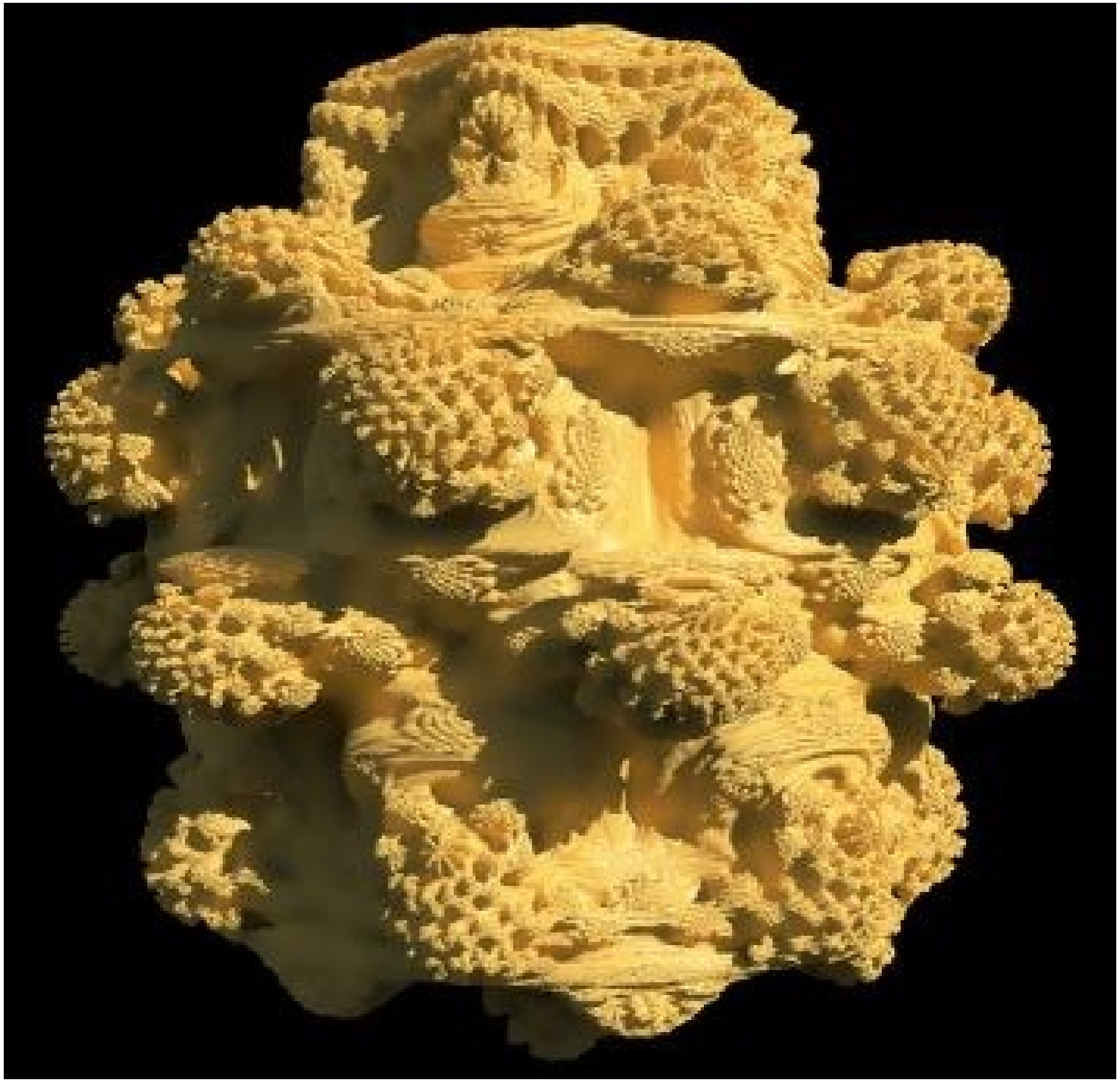


Fragment shader

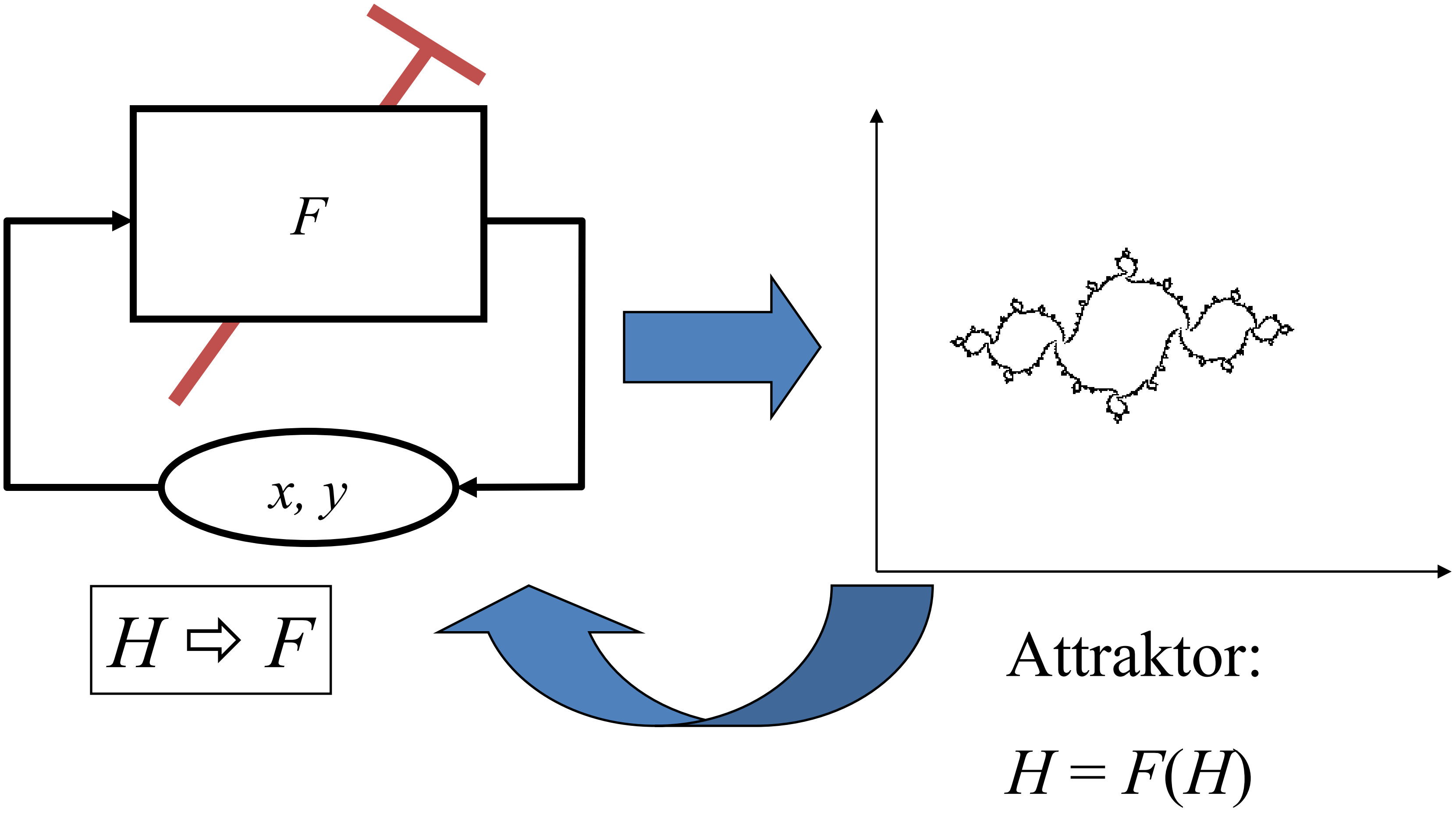
```
in vec2 c;
out vec4 fragCol;
const int nIteration = 1000;

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);
}
```

$$z = z^2 + c$$

Mandelbulb



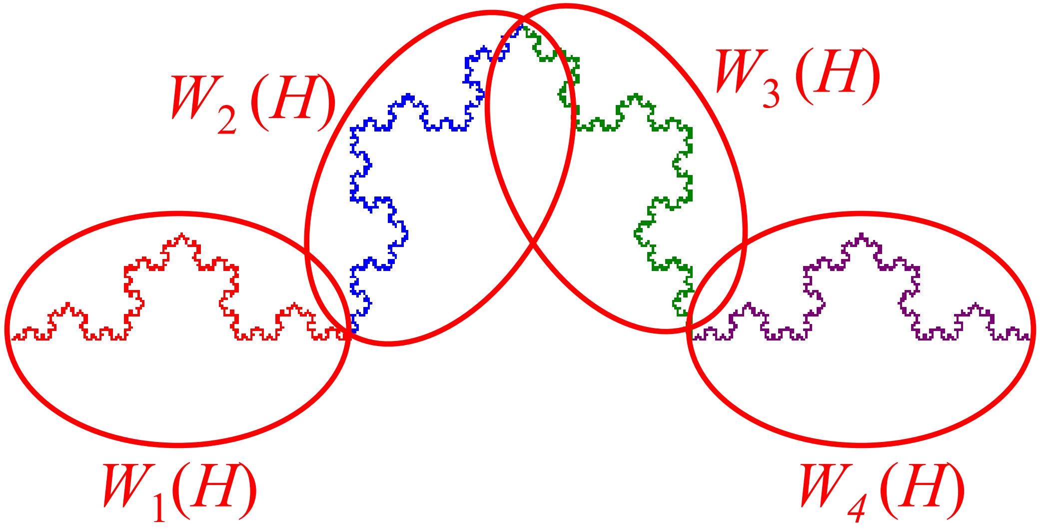
Inverz feladat: IFS modellezés



F: szabadon vezérelhető, legyen stabil attraktora

Melyik függvény attraktora a Koch görbe?

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

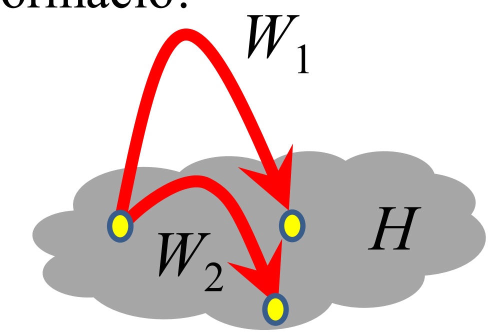


$$\boxed{W_k(x,y) = [x,y] \cdot \boldsymbol{A}_k + \boldsymbol{q}_k}$$

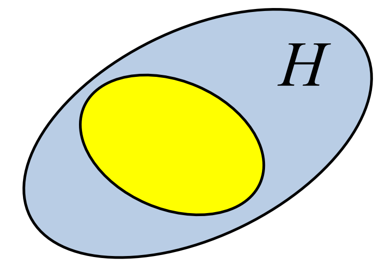
F: többértékű lineáris leképzés

Nem feltétlenül hasonlósági transzformáció!
Nem csak önhasonló objektumok.

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

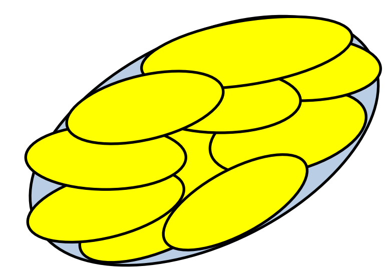


$$\boxed{W_k(x,y) = [x,y]\cdot \boldsymbol{A}_k + \boldsymbol{q}_k}$$



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

$$\boxed{H = W_1(H) \cup W_2(H) \cup \ldots \cup W_m(H)}$$



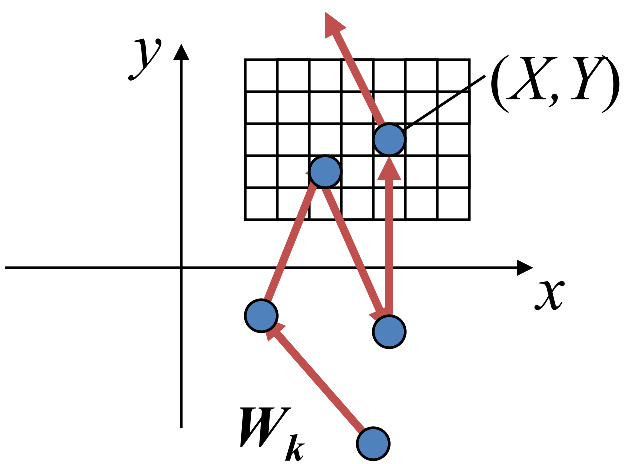
$$H = F(H)$$

<u>Kollázs:</u>
lefedés a kicsinyített változatokkal
Nem feltétlenül diszkjunkt

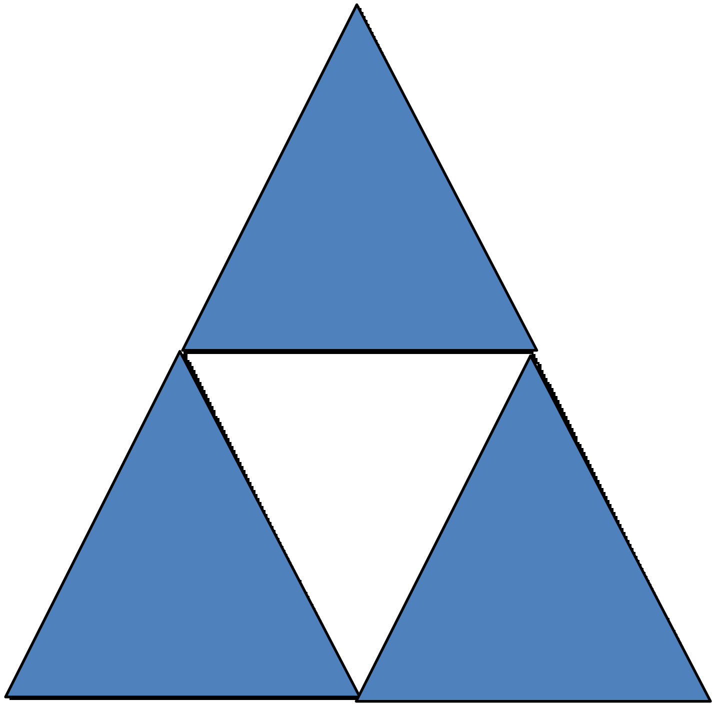
IFS rajzolás: iterációs algoritmus

<u>IFSDraw ( )</u>

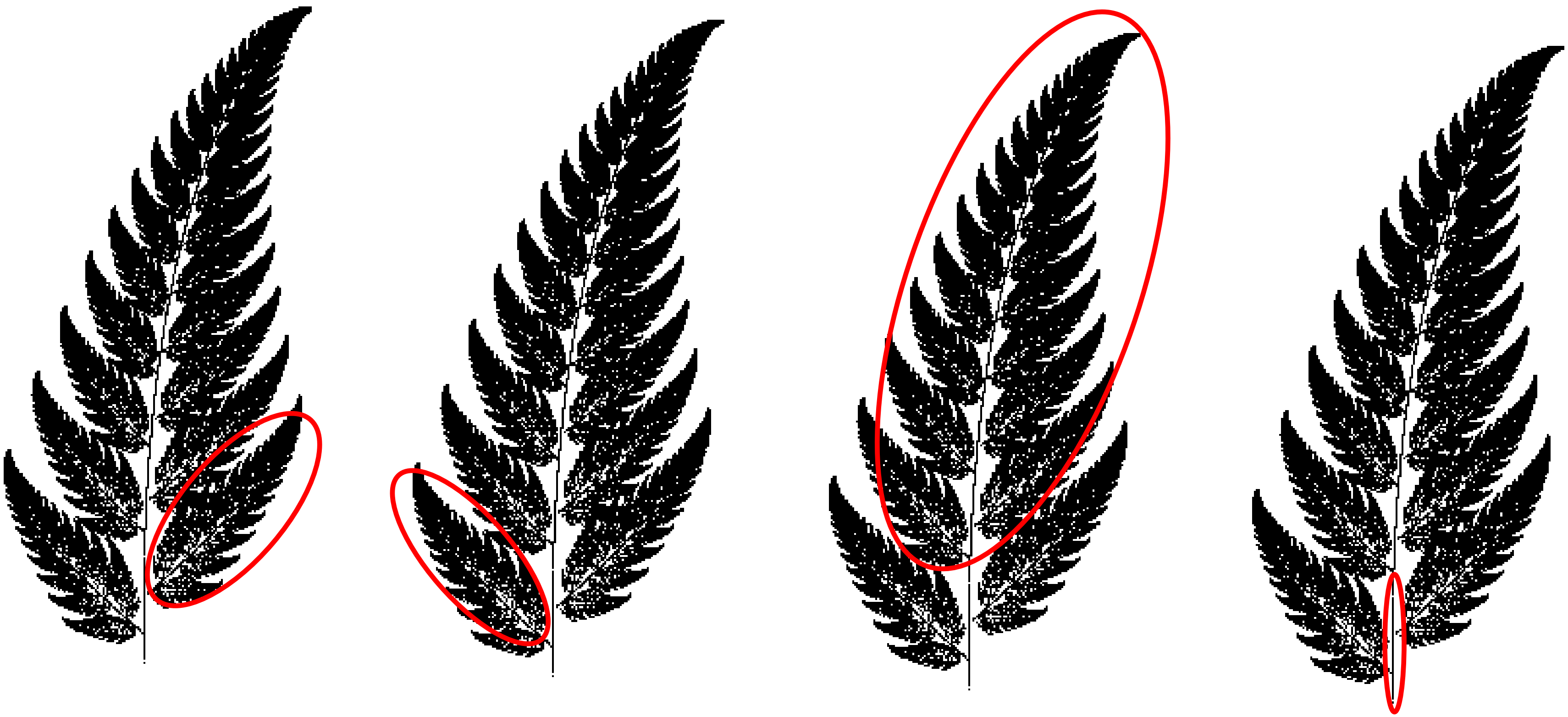
Legyen [x,y] = [x,y] A_1 + q_1 megoldása a kezdő [x,y]
FOR i = 0 TO "infinity" DO
IF InWindow(x, y)
WindowViewport(x, y ⇨ X, Y)
Write(X, Y, color);
ENDIF
Válassz k-t p_k valószínűséggel
[x,y] = [x,y] A_k + q_k
ENDFOR
END



Egyszerű IFS-ek



IFS modellezés



IFS képek