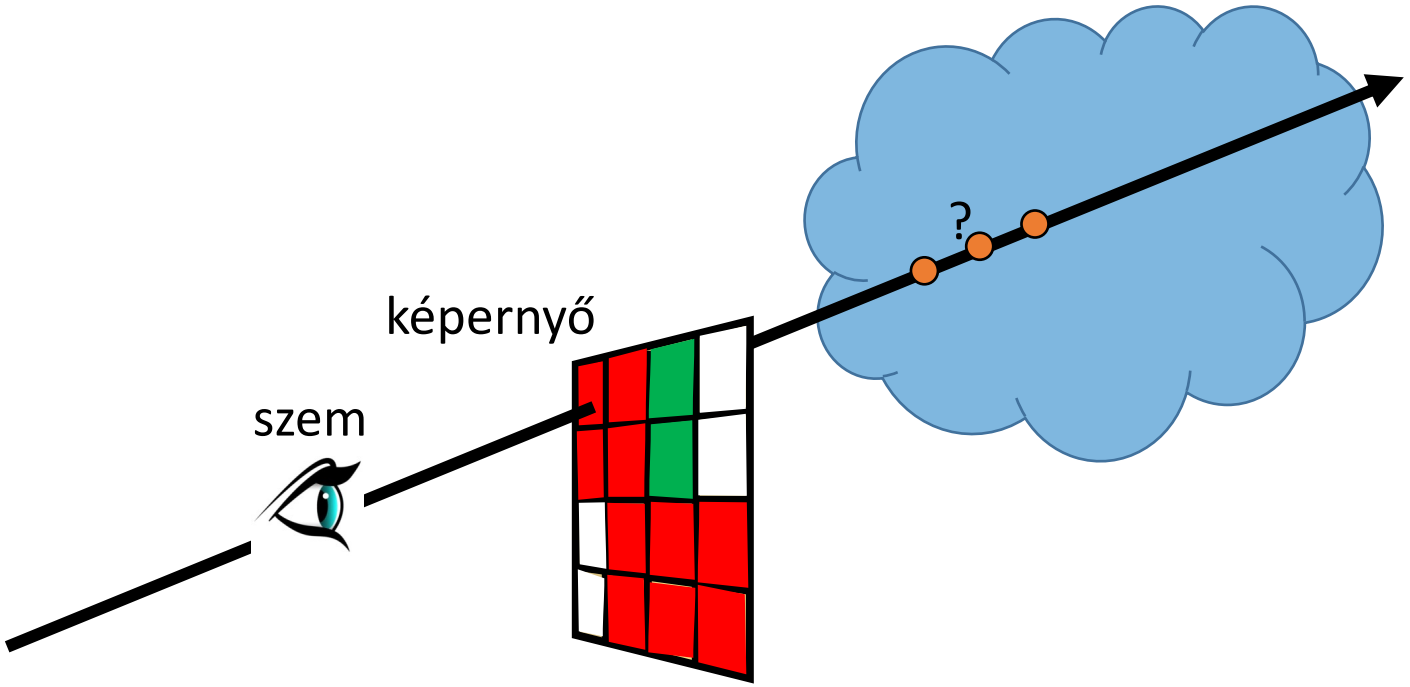
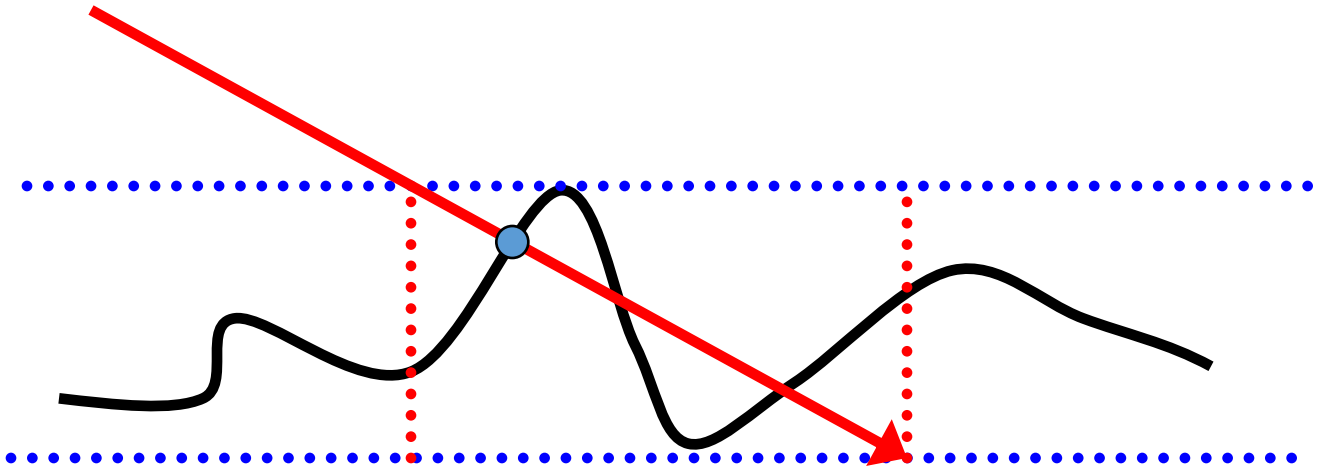
Isosurface ray casting



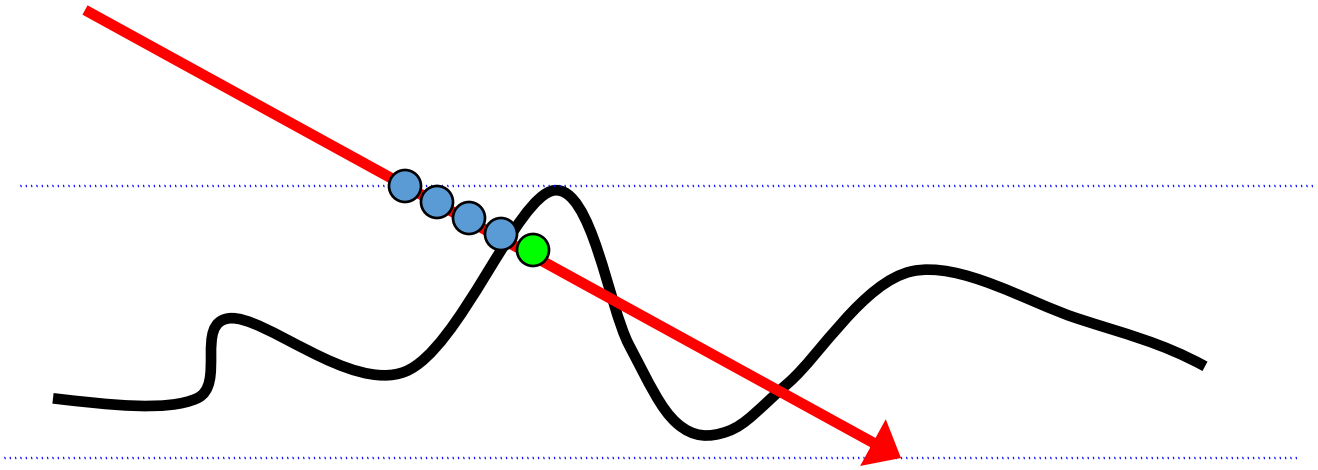
Iteratív keresés

- Skalármező szintfelületével metszéspont keresés



Lineáris keresés (ray marching)

- Első pont megtalálás legyen biztos

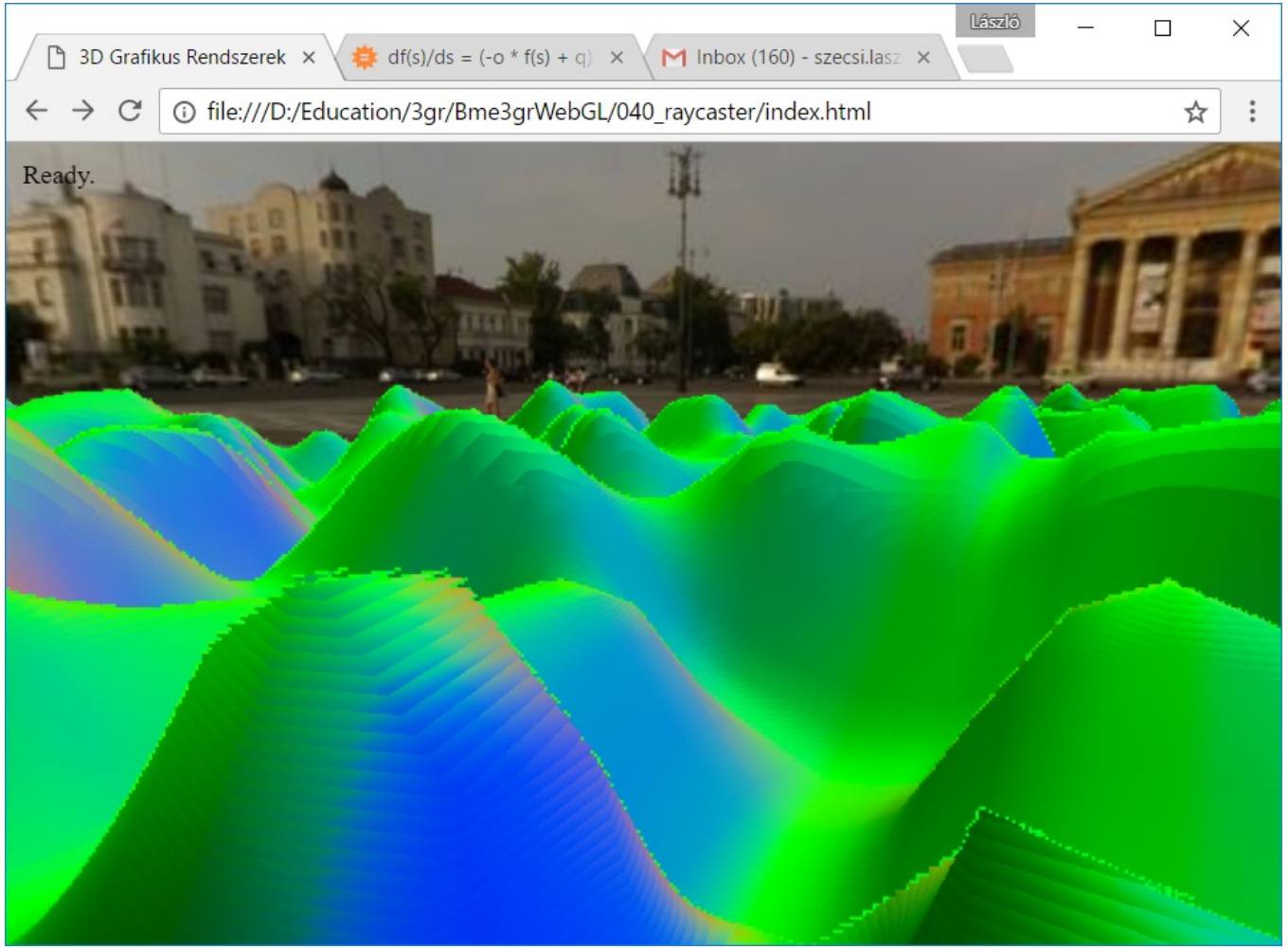


Normálvektor meghatározása árnyaláshoz

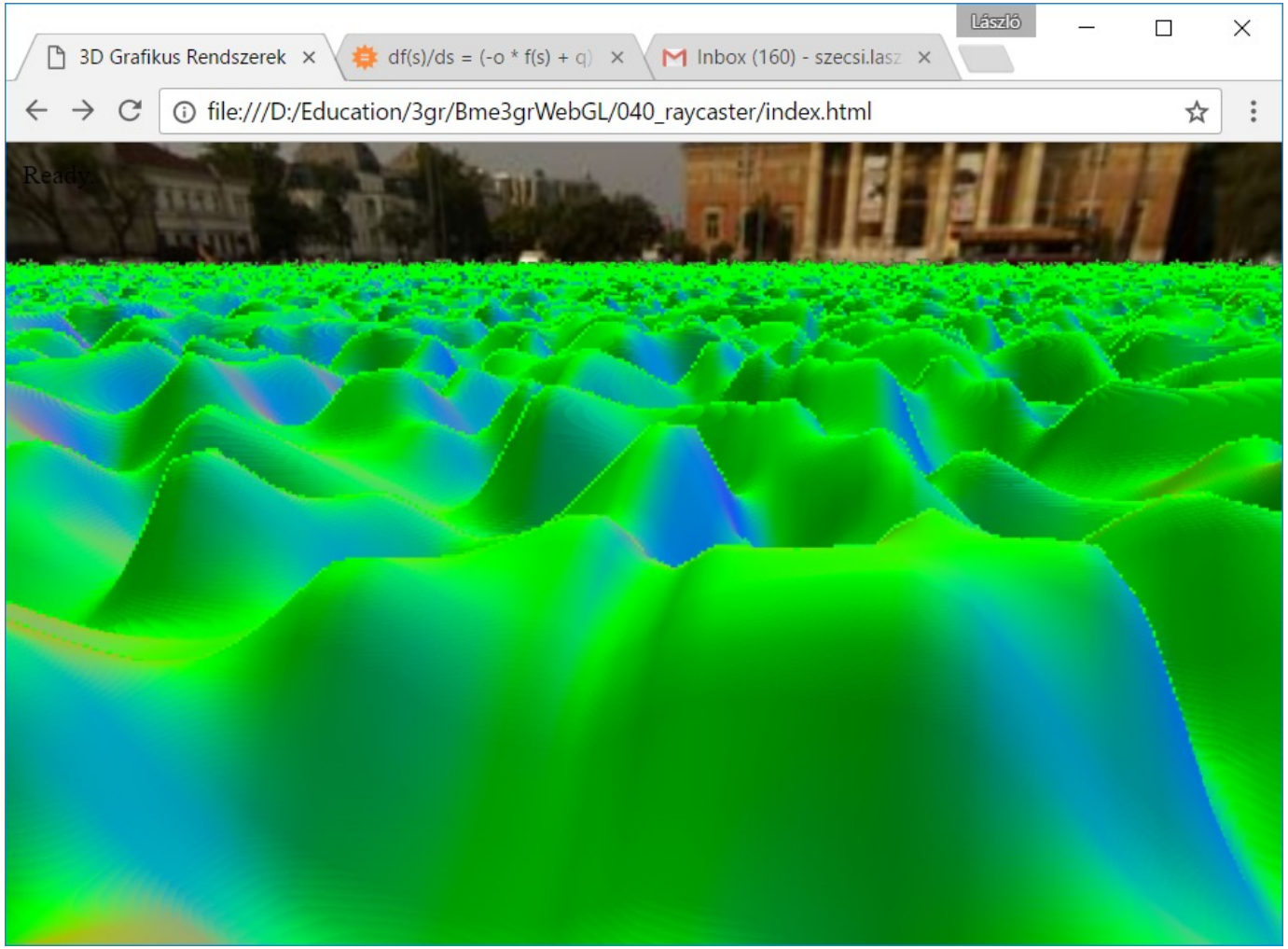
- gradiens meghatározása
  - analitikusan számolható
  - parciális deriváltak közelítése differenciákkal
    - $df(r)/dx = f(r+(\varepsilon, 0, 0)) - f(r-(\varepsilon, 0, 0))$
    - ahol ε pl. lehet függhet a távolságtól
  - normalizálni is kell

- szűrni is jó lenne: zajok szűrése

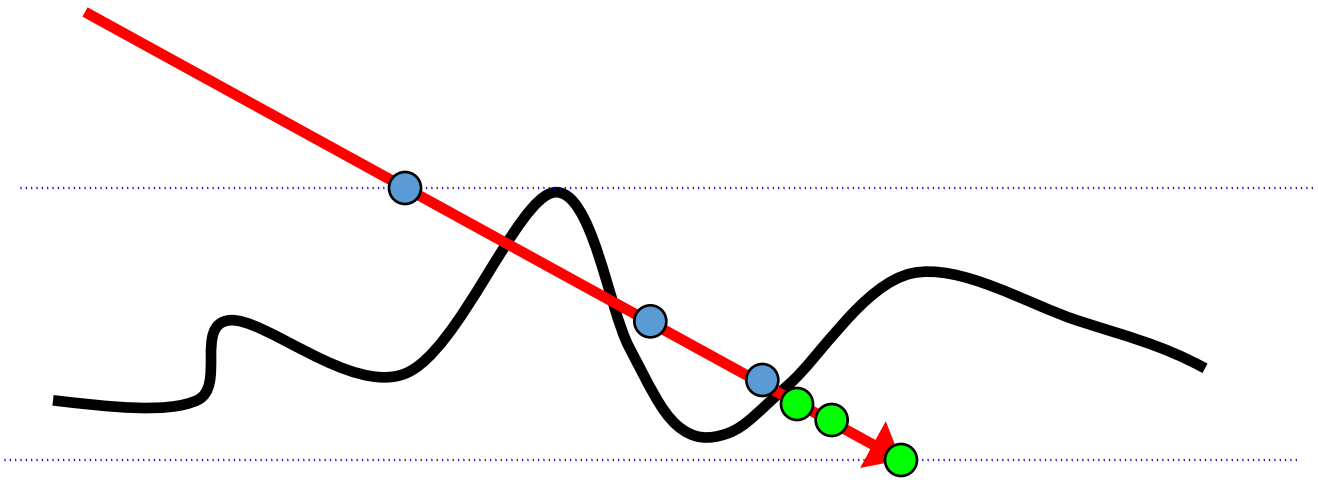
Normálvektorok megjelenítése



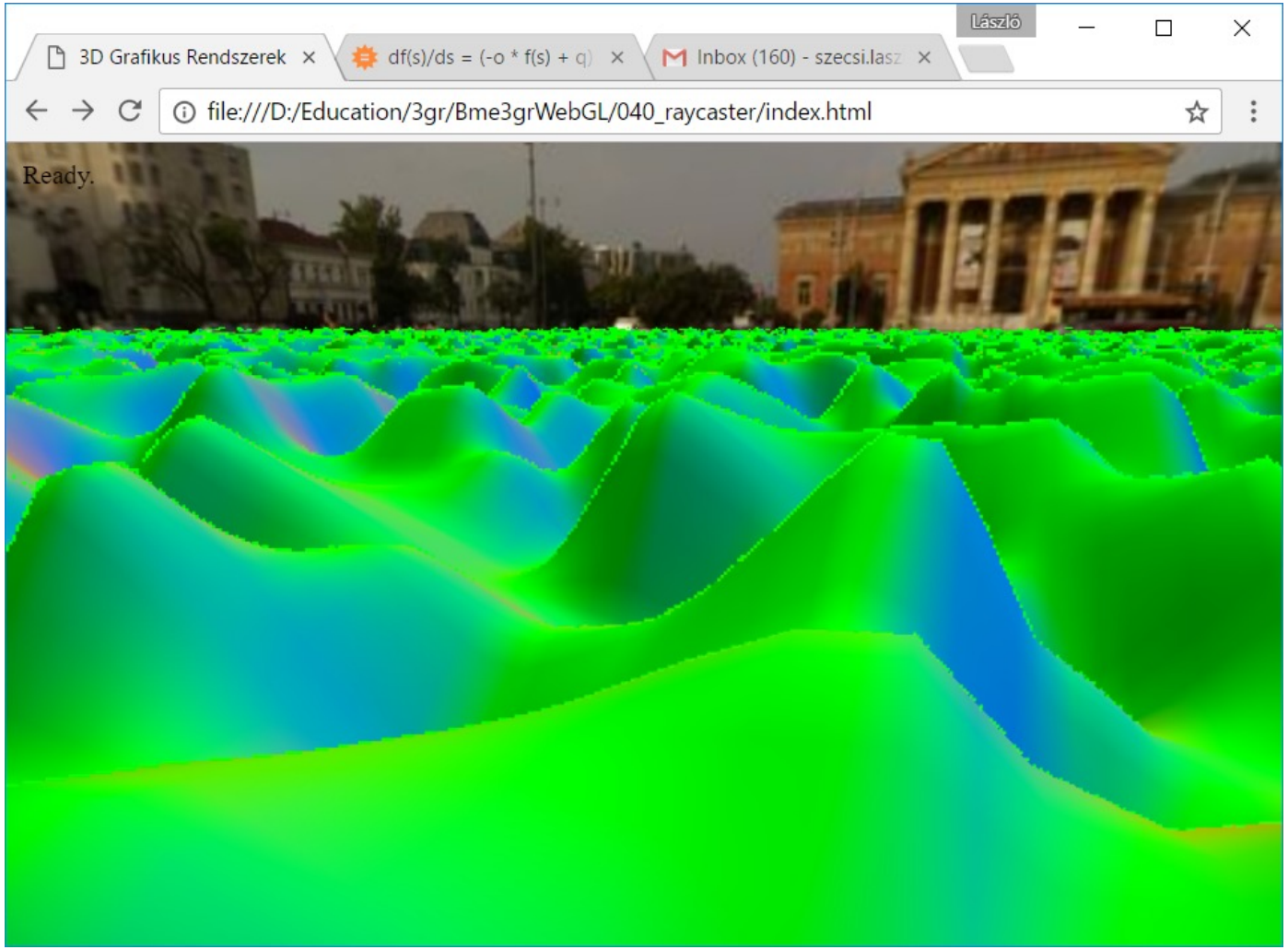
Normálvektorok megjelenítése



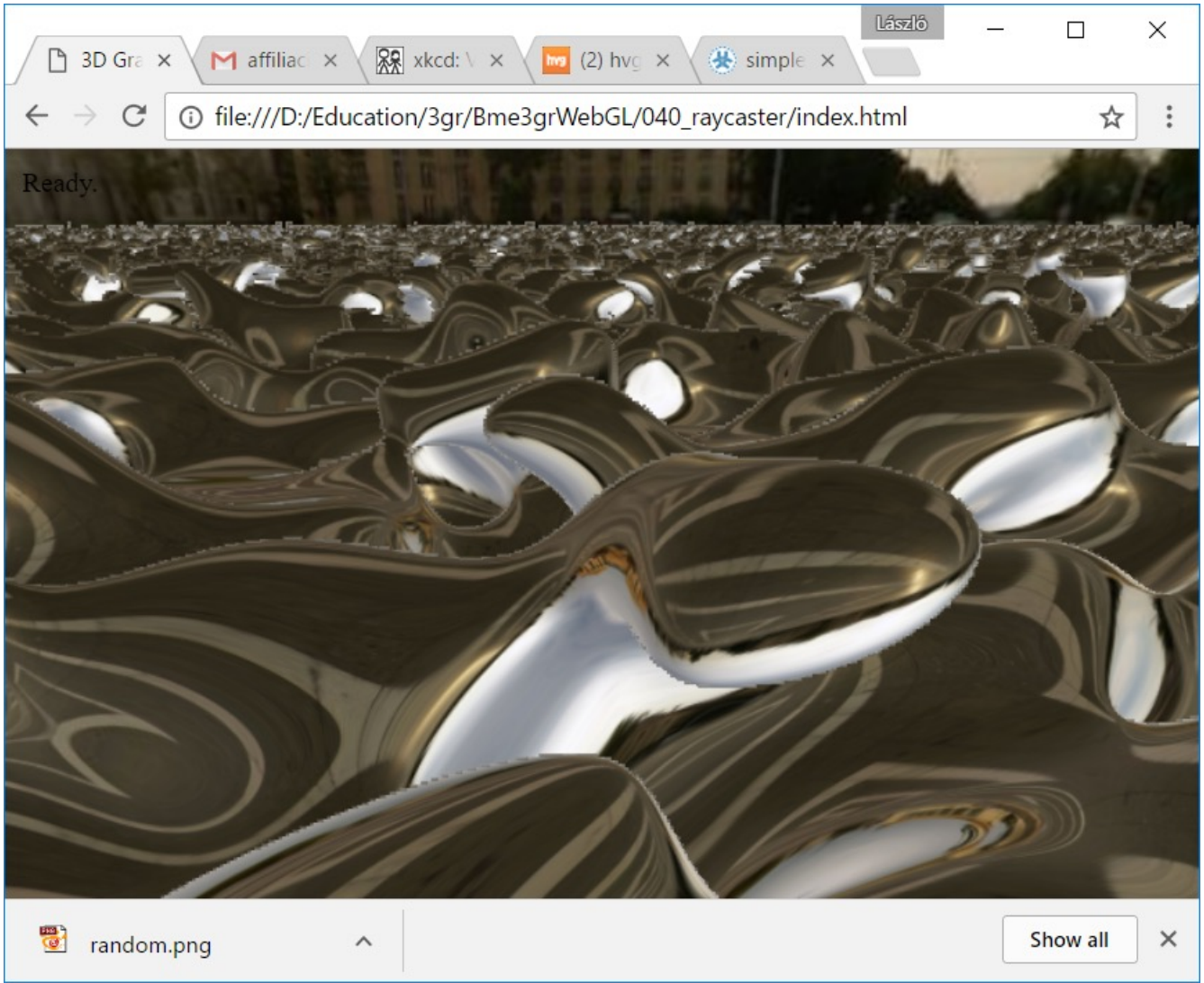
Bináris keresés



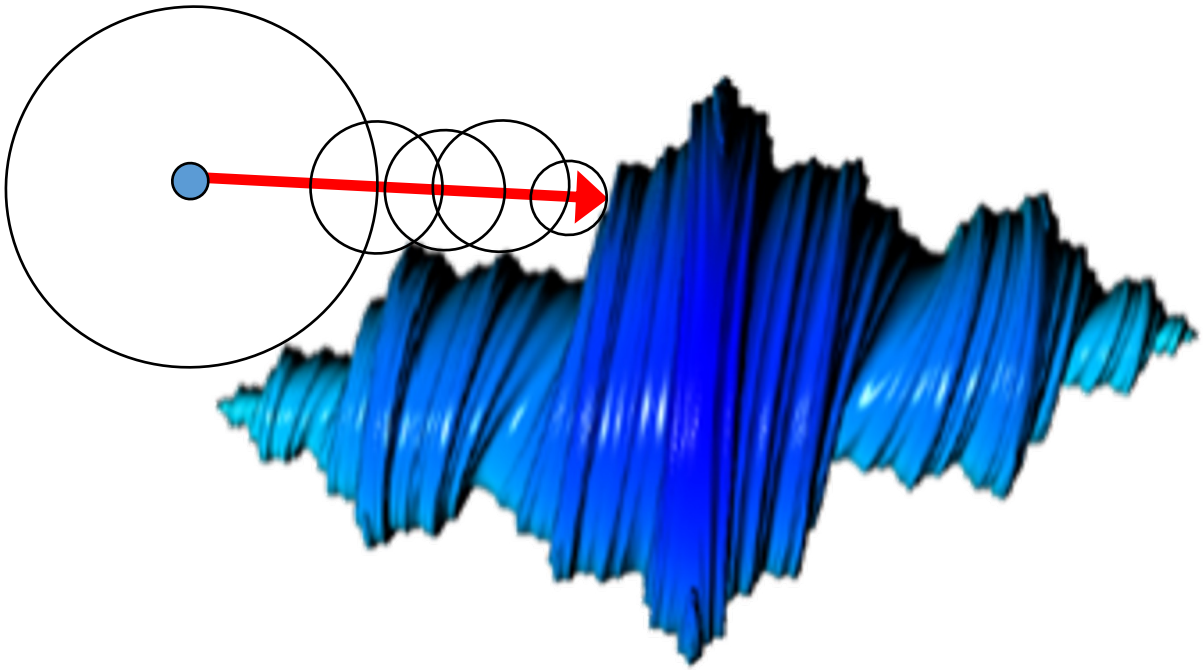
Normálvektorok megjelenítése



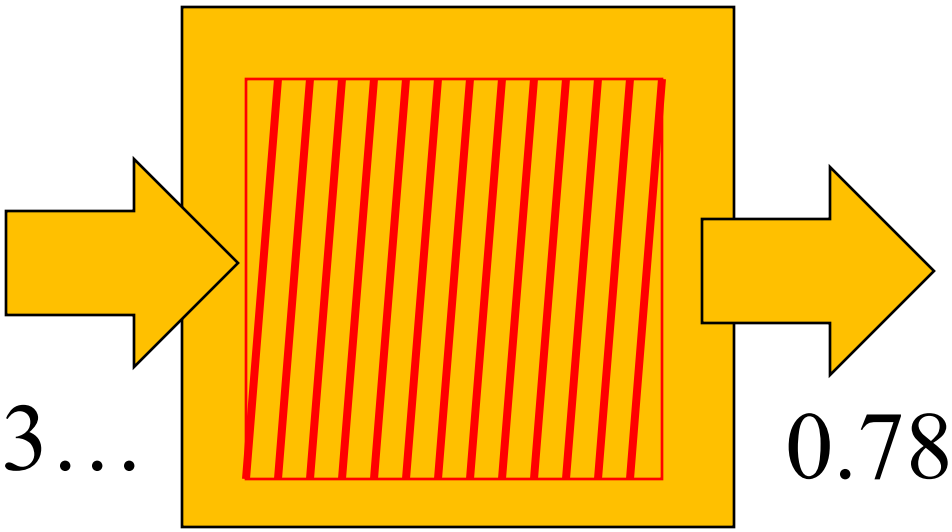
Ray casting + envmap



Sphere tracing



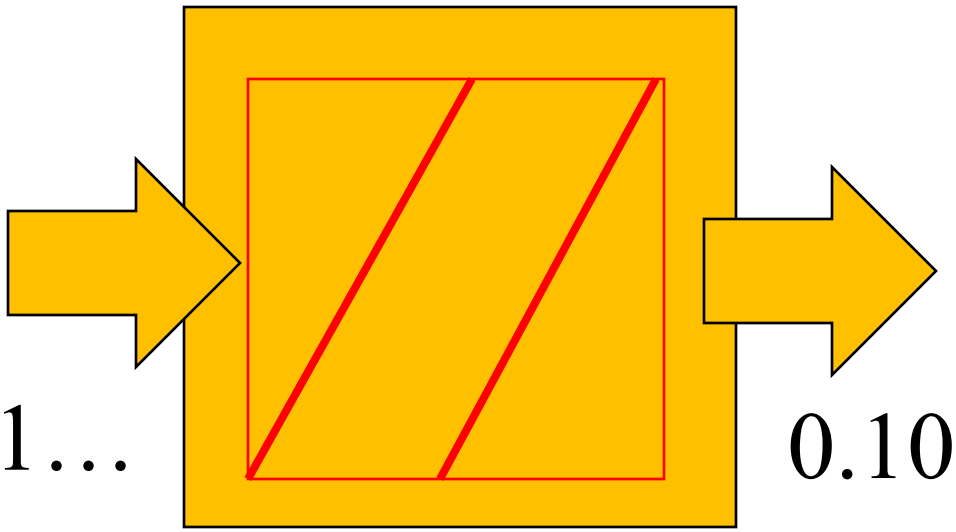
Diszkrét dinamikus rendszerek kaotikus viselkedése



0.478564783…
0.785647835…
0.856478359…

0.785647835…
0.856478359…
0.564783591…

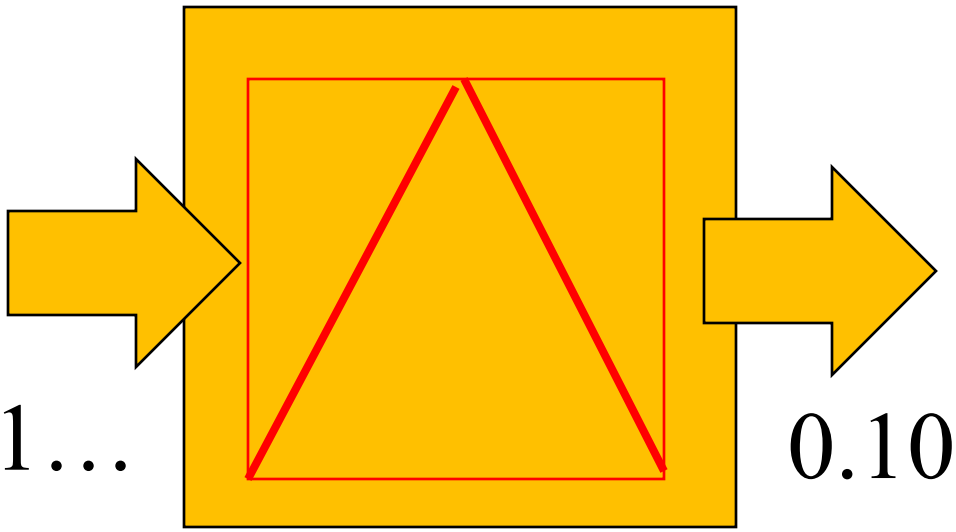
Diszkrét dinamikus rendszerek kaotikus viselkedése



0.010111011…
0.101110111…
0.011101110…

0.101110111…
0.011101110…
0.111011101…

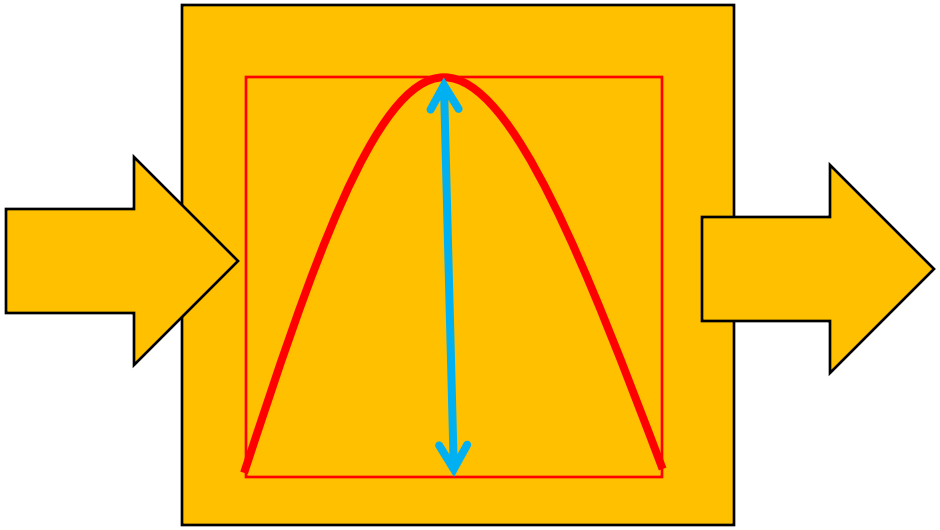
Diszkrét dinamikus rendszerek kaotikus viselkedése



0.010111011…
0.101110111…
0.100010001…

0.101110111…
0.100010001…
0.000100010…

A nyulak szigete



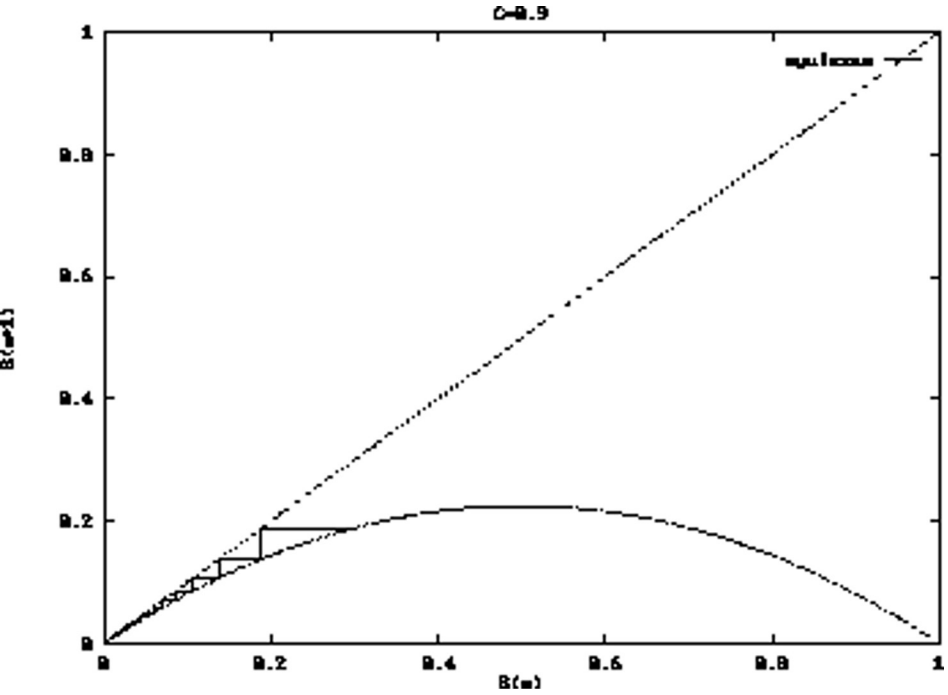
nyulak száma idén

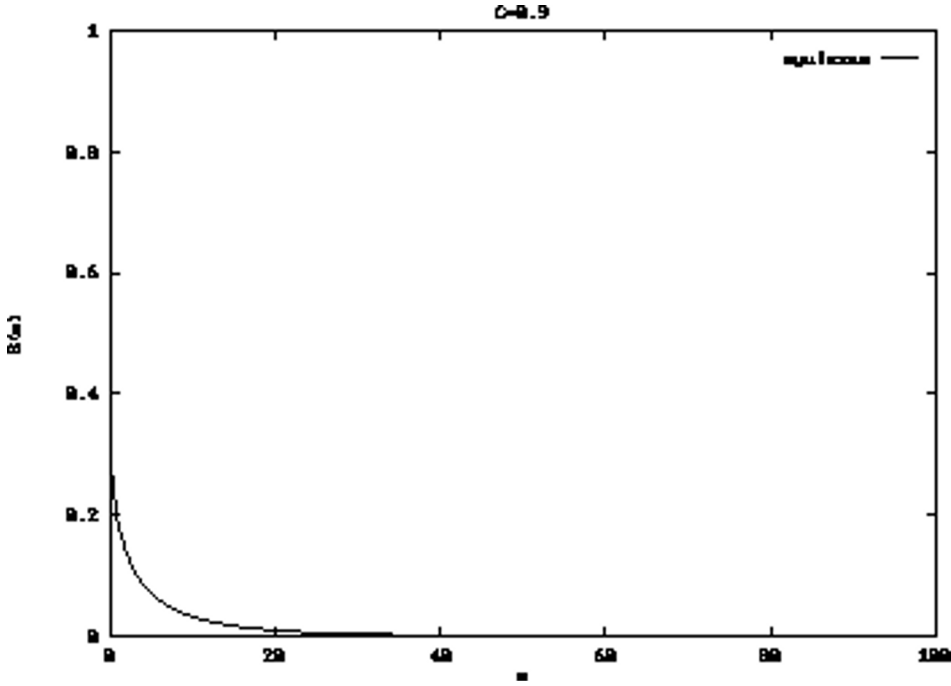
nyulak száma jövőre

$$S_{n+1} = C\, S_n\, (1-S_n)$$

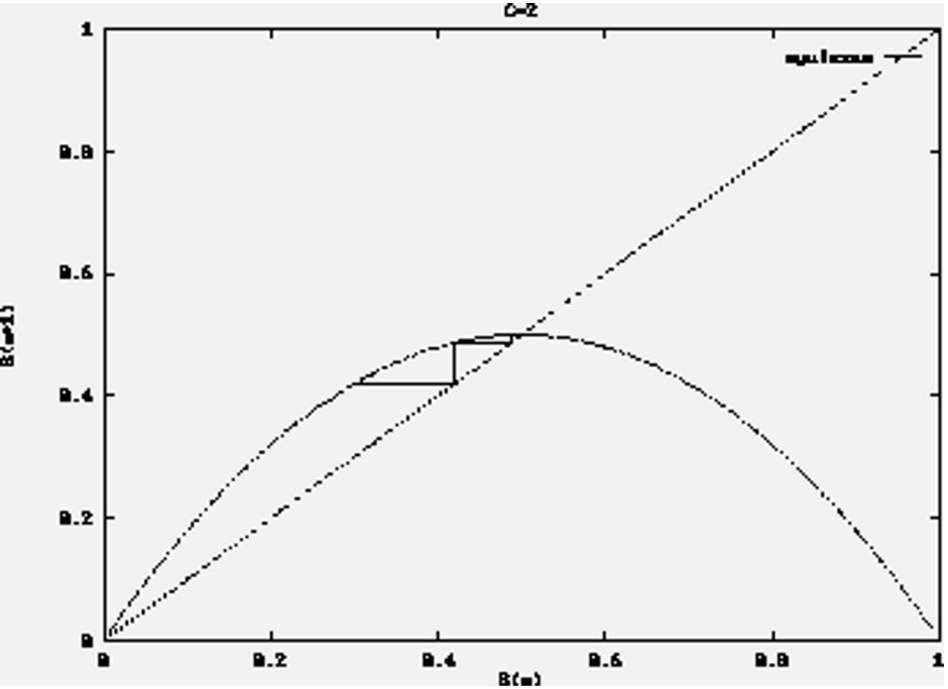
Alacsony C

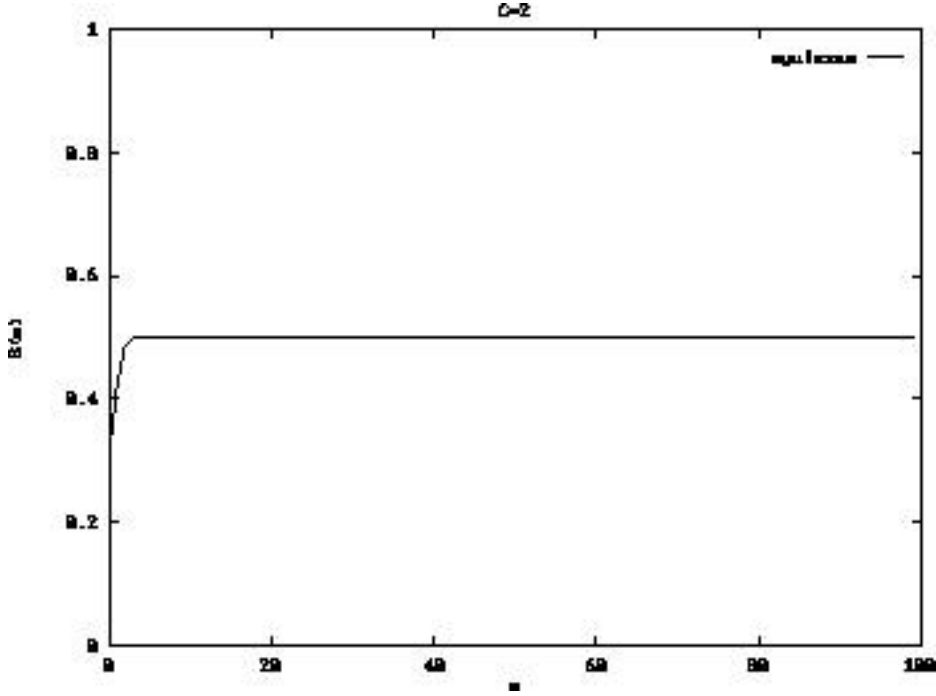
$$S_{n+1} = C\, S_n\, (1-S_n)$$



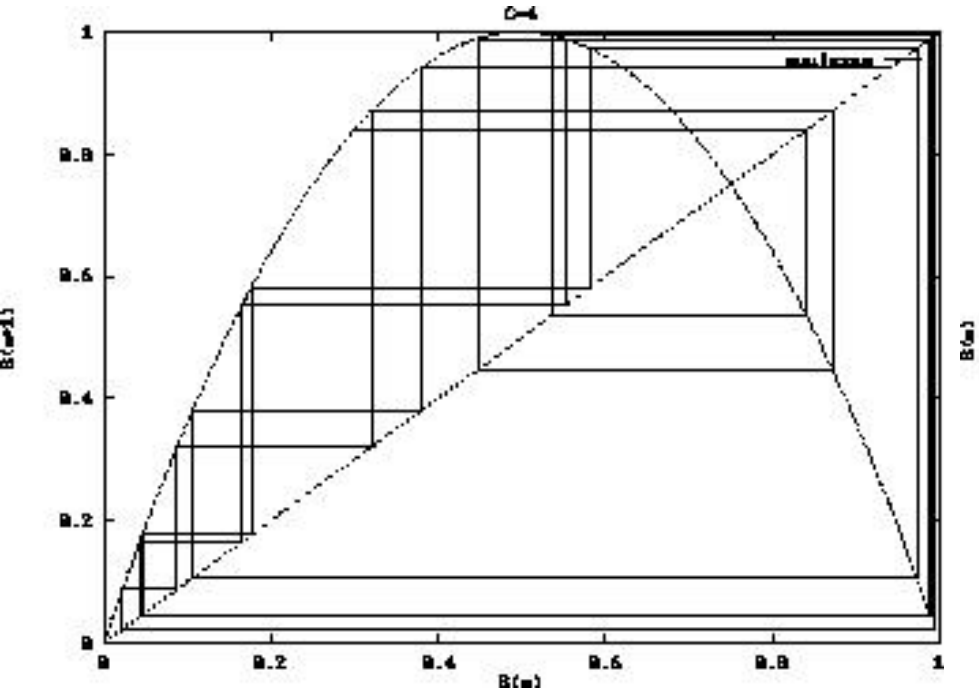


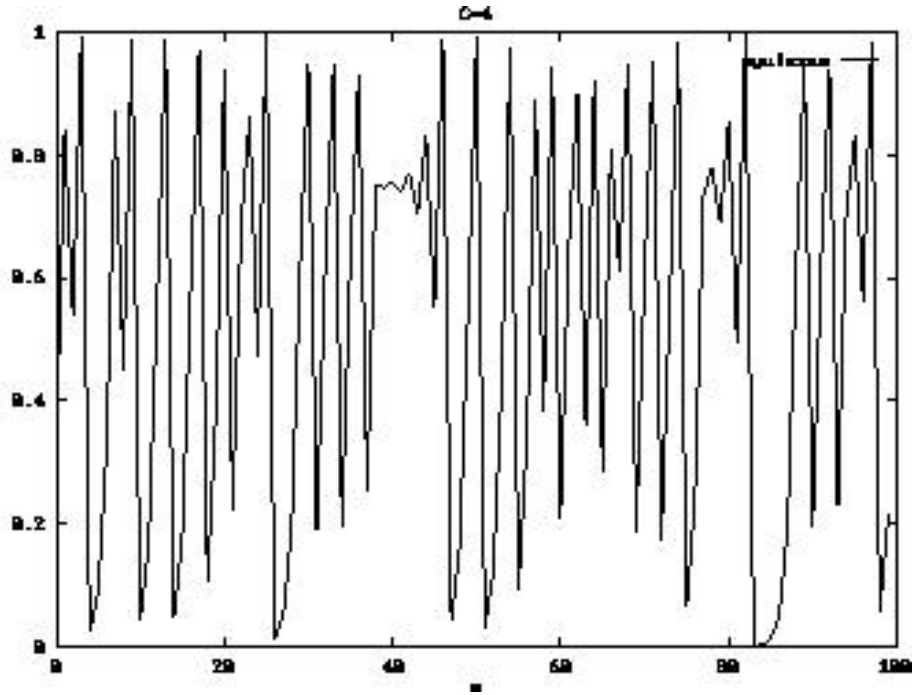
Közepes C



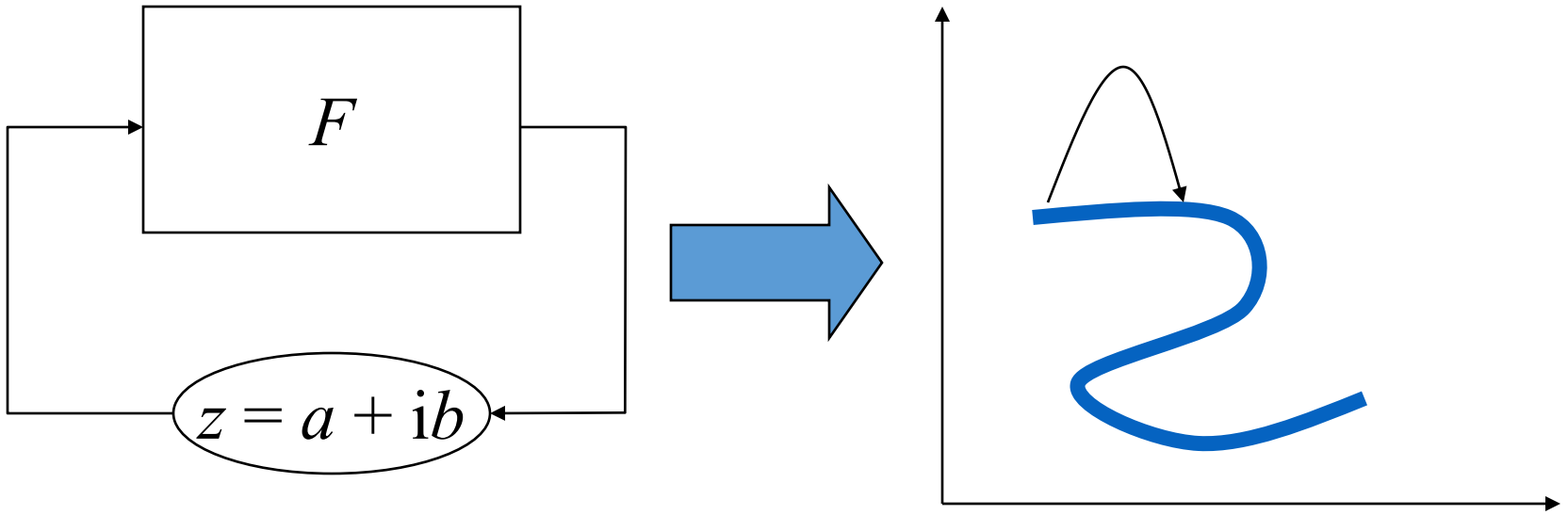


Magas C

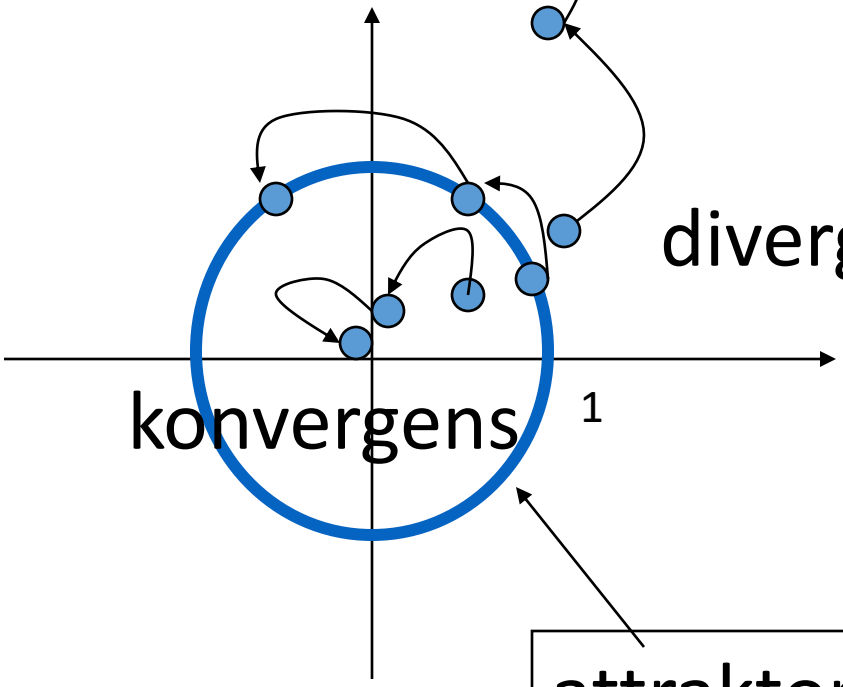




Kaotikus rendszerek a síkon



$z \rightarrow z^2$



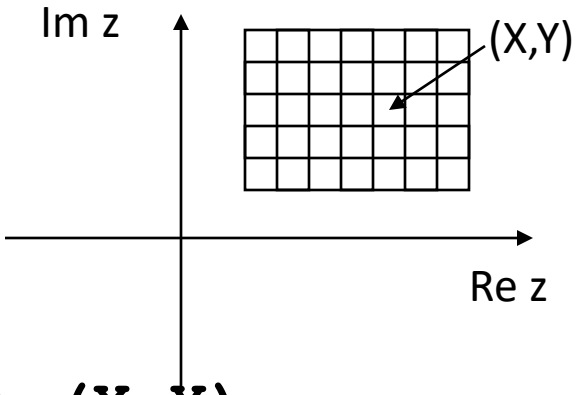
$z = r\, e^{i\phi}$

$r \rightarrow r^2$

$\phi \rightarrow 2\phi$

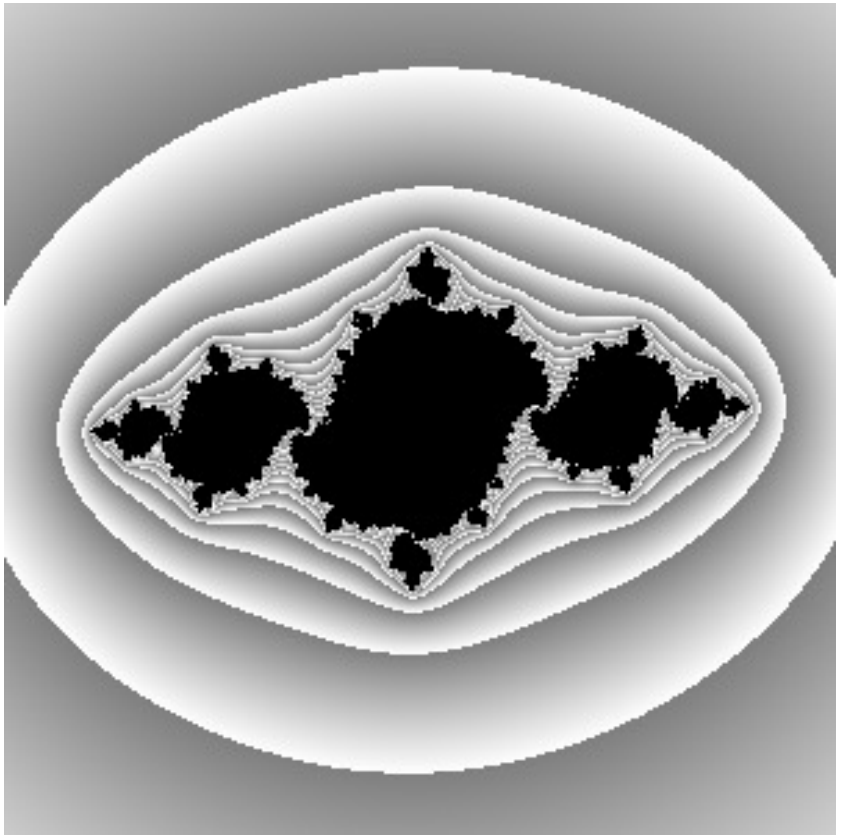
attraktor: $H = F(H)$

A konvergens rész kitöltése Julia halmaz: $z \rightarrow z^2+c$




```
filledJuliaDraw ( )
  FOR Y = 0 TO Ymax DO
    FOR X = 0 TO Xmax DO
      (x, y) = viewport2Window(X,Y)
      z = x + j y
      FOR i = 0 TO n DO z = z² + c
      IF |z| > ∞ THEN WRITE(X,Y, white)
      ELSE WRITE(X,Y, black)
    ENDFOR
  ENDFOR
END
```

Julia halmaz



Quaternion Julia távolságfüggvény

```
vec4 quatMult( vec4 q1, vec4 q2 ) {
   vec4 r;

   r.x   = q1.x * q2.x - dot( q1.yzw, q2.yzw );
   r.yzw = q1.x * q2.yzw + q2.x * q1.yzw + cross( q1.yzw, q2.yzw );

   return r;
}

vec4 quatSq( vec4 q ) {
   vec4 r;

   r.x   = q.x * q.x - dot( q.yzw, q.yzw );
   r.yzw = 2.0 * q.x * q.yzw;

   return r;
}
```

Érdekesebb távolságfüggvény: quaternion Julia

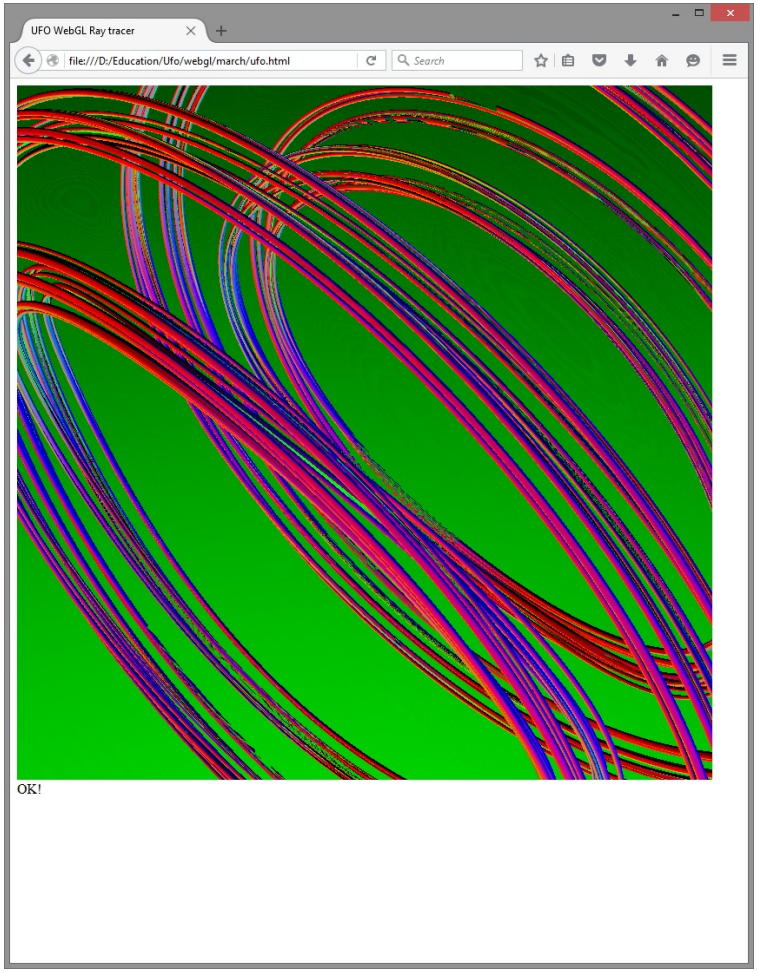
```
void iterateIntersect( inout vec4 q, inout vec4 qp) {
   for( int i = 0; i < 10; i++ ) {
      qp = 2.0 * quatMult(q, qp);
      q = quatSq(q) + vec4(1, 0.5, -0.1, 0.3);

      if( dot( q, q ) > 7.0 ) {
         break;
      }
   }
}
```

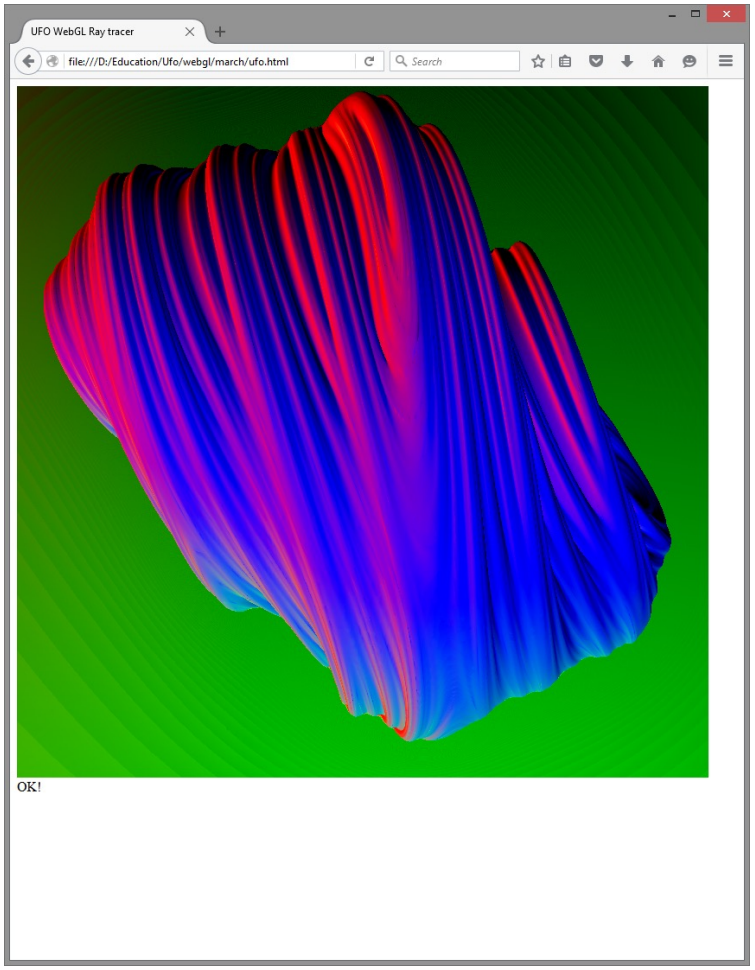
Érdekesebb távolságfüggvény: quaternion Julia

```
float dist(vec3 p)
{
  vec4 z = vec4( p, 0.0 );
  vec4 zp = vec4( 1, 0.0, 0.0, 0.0 );
  iterateIntersect( z, zp );
  float normZ = length( z );
  return 0.5 * normZ * log( normZ ) / length(
zp );
}
```

Webgl sphere tracer

- sphere tracing ciklus
  - webgl1: konkrét lépésszám kell (pl. 150)
  - pozíció a szemből indul
  - távolságfüggvény kiértékelése
  - léptetés a távolsággal
  - kilépés a ciklusból (break;), ha a távolság kisebb mint epszilon
    - epszilon függhet a teljes megtett távolságtól: távolabb nagyobb hiba is oké
- ha nem konvergált az iteráció, a hátteret látjuk


UFO WebGL Ray tracer
file:///D:/Education/Ufo/webgl/march/ufo.html
Search
OK!


UFO WebGL Ray tracer
file:///D:/Education/Ufo/webgl/march/ufo.html
Search
OK!