Modeling em C.G.
(Fractais , Partículas, etc.)

Cap 4 (do livro texto)

UFF - 2014

Até agora vimos

formas e objetos em uma dimensão, duas dimensões, e até mesmo três.

Ate hoje na sua interação com objetos você considerou medidas como :

a circunferência de um círculo?

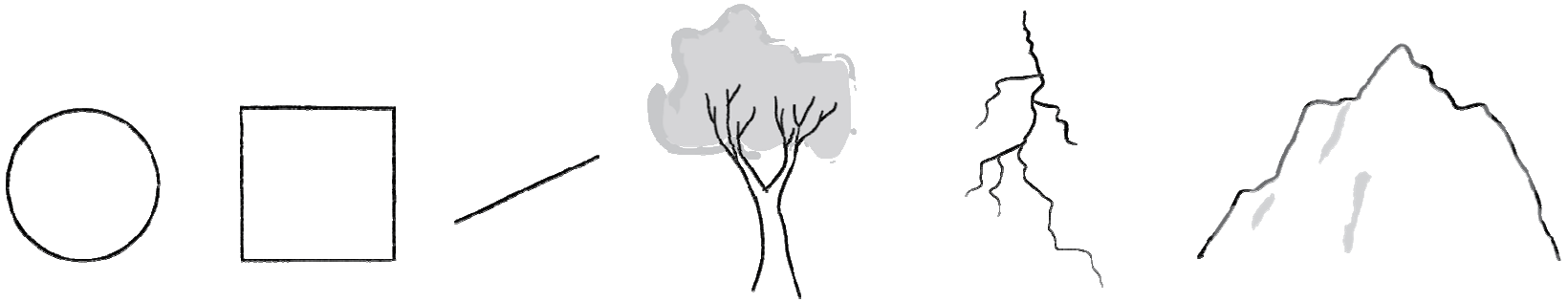
área de um retângulo?

distância entre um ponto e uma linha?

Discrição de objetos pelos seus vértices no espaço cartesiano.

Este tipo de geometria é geralmente referido como geometria euclidiana, em homenagem ao matemático grego Euclides.

Mas há mais que isso:



Agora estamos prontos para visualizar Processos utilizando recursividade.

Mas e você for olhar mais longe

vai ver as árvores da rua, as folhas que estão nas árvores, o relâmpago na tempestade, a couve-flor que comeu no almoço, os vasos sanguíneos no seu corpo, e as montanhas e as formas dos litorais....

A maioria das coisas que você encontrar na natureza não pode ser descrito pelas formas geométricas idealizadas da geometria euclidiana.

Se queremos construir modelos computacionais com padrões além das formas simples é hora de aprender sobre os conceitos por trás e técnicas para simular a geometria da natureza: os fractais.

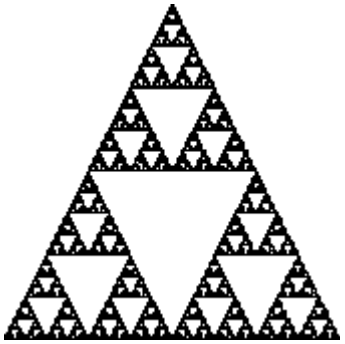
Geometria Fractal

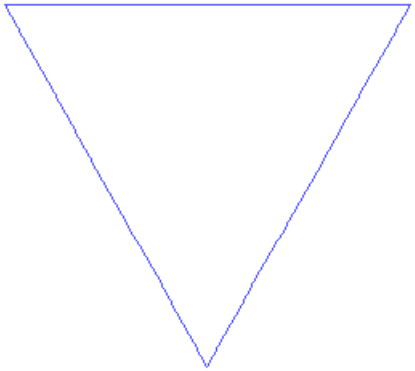
Estuda subconjuntos complexos de espaços métricos.

Na geometria de fractais determinísticos, os objetos estudados são subconjuntos gerados por transformações geométricas simples do próprio objeto nele mesmo.

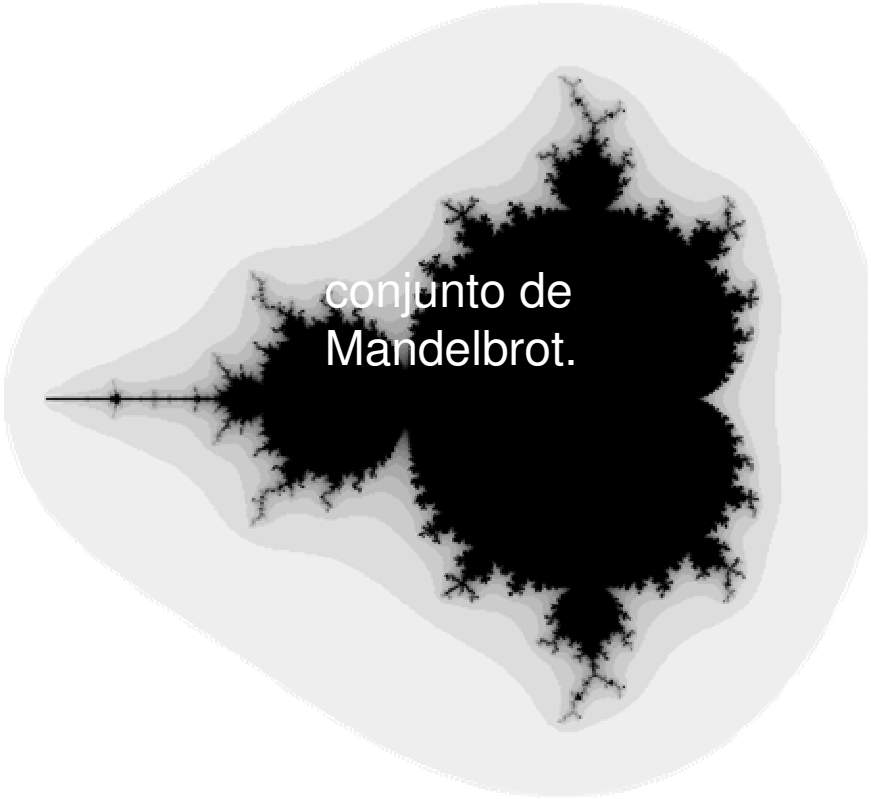
Um fractal é composto por partes reduzidas dele próprio

Fractais podem ser Determinísticas ou Randômica





Exemplo:



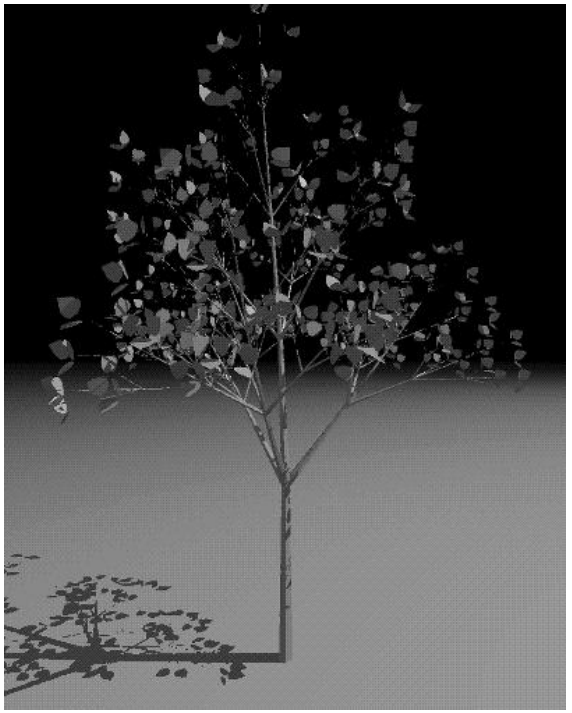
Um das mais conhecidas fractais é nomeada conjunto de Mandelbrot.

Resulta de testar se um números complexos depois multiplicado iterativamente tende para o infinito, ou ficará limitado (algoritmo "escapar-time") .

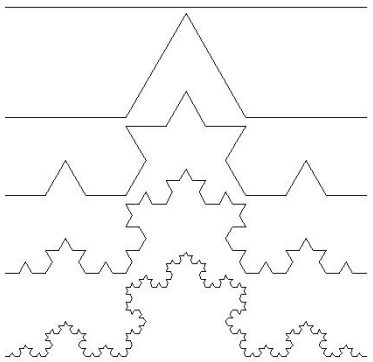
Esse conjunto é um exemplo de outro chamado conjunto de Julia.

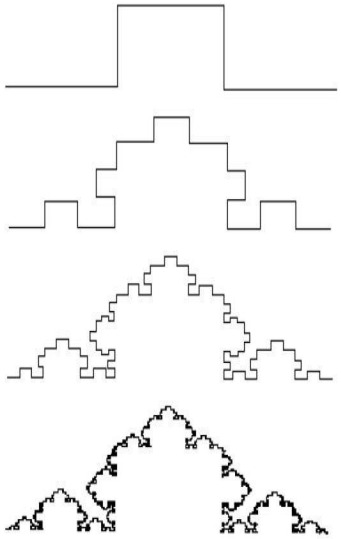
exemplos

Randômicos x determinísticos



Exemplos determinísticos





Curvas de KOCH:

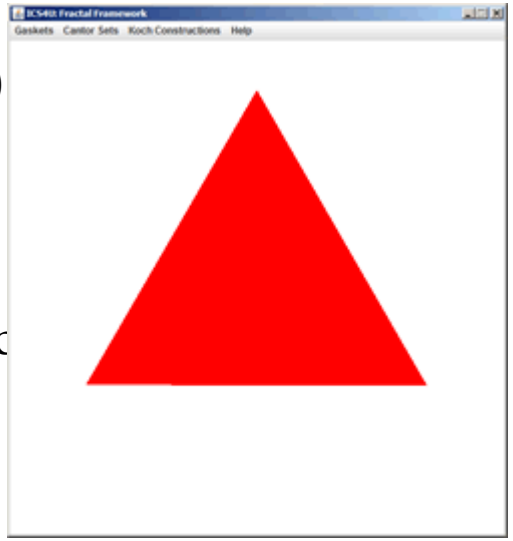
Proposta por Von Koch em 1904, tem a seguinte geração:

desenhe uma linha e a divida em 3 partes iguais
(d = 1/3 * r , onde d= escala da reta e r= comprimento inicial)
depois faça o terço central da reta ser substituindo por :

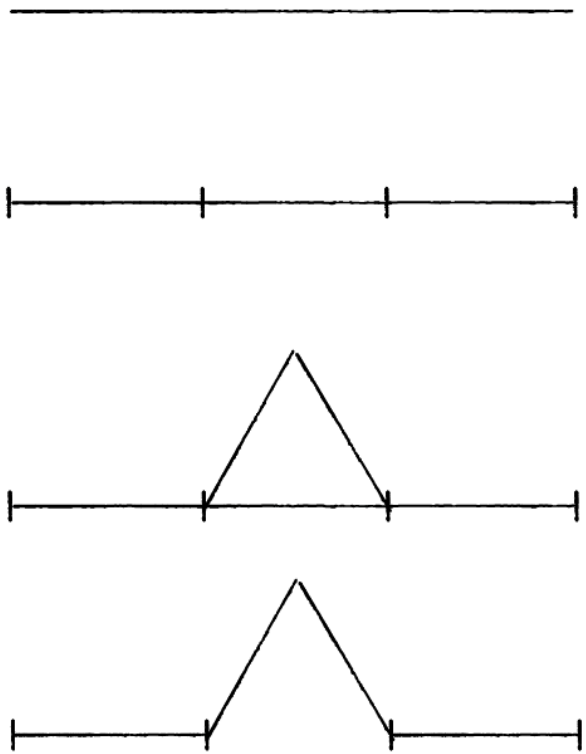
2 pedaços, repita o processo infinitamente (tridaic)

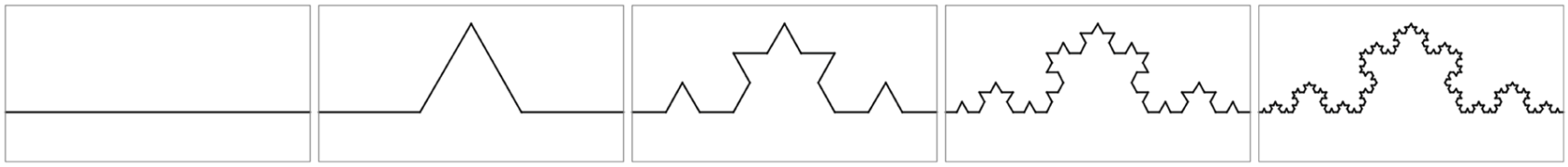
ou

3 pedaços, repita o processo infinitamente (quadric



A curva de Koch e outras fractais são chamados de "monstros matemáticos". Isso é devido a estranhos paradoxos que surgem quando se aplica a definição recursiva um número infinito de vezes. Se o comprimento da linha original é um, a primeira iteração da curva de Koch irá produzir uma linha de quatro terços do comprimento original (cada segmento é um terço do comprimento da linha inicial). Fazendo novamente você terá dezesseis nonos. À medida continua para o infinito, o comprimento da curva de Koch se aproxima do infinito. No entanto, ele se encaixa no pequeno espaço finito inicial!



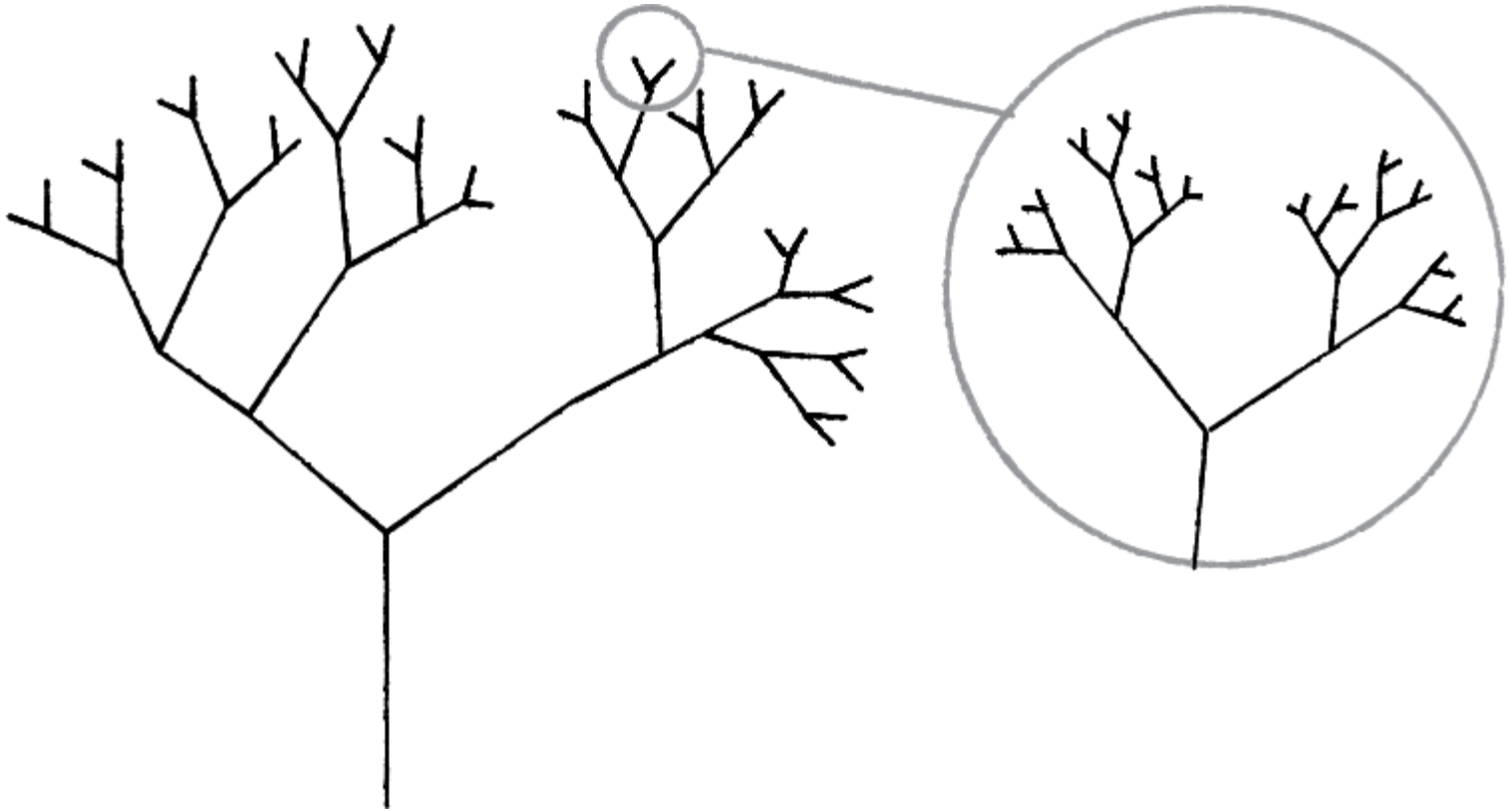


A geometria fractal

Geometria que estuda as propriedades e comportamento dos fractais.

Descreve muitas situações que não podem ser explicadas facilmente pela geometria Euclidiana, e são aplicadas em ciência, tecnologia e arte gerada por computador.

As raízes conceituais dos fractais remontam as tentativas de entender objetos para os quais as definições tradicionais baseadas na geometria euclidiana falham.



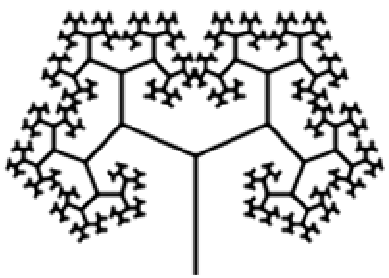
A geometria fractal

Um fractal é um objeto geométrico que pode ser dividido em partes, cada uma das quais semelhante ao objeto original.

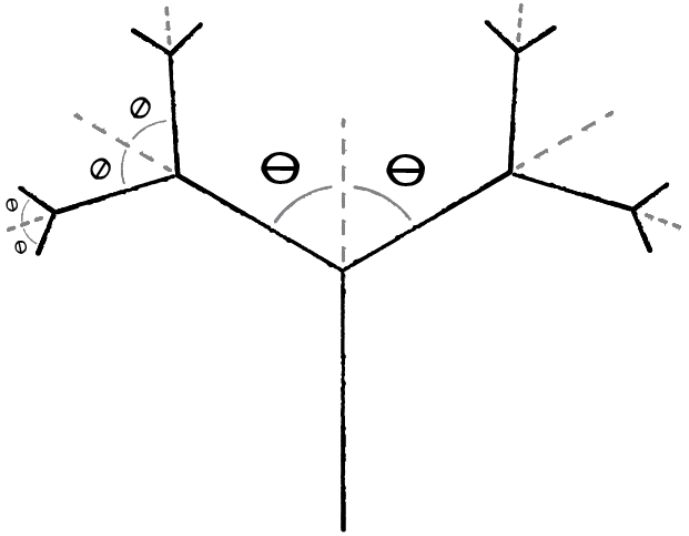
Diz-se que os fractais têm infinitos detalhes, são geralmente autossimilares e independem de escala.

Em muitos casos um fractal pode ser gerado por um padrão repetido, tipicamente um processo recursivo ou iterativo.

O termo foi criado em 1975 por Benoît Mandelbrot, matemático francês nascido na Polônia, a partir do adjetivo latino fractus, do verbo ***frangere***, que significa quebrar e fração.



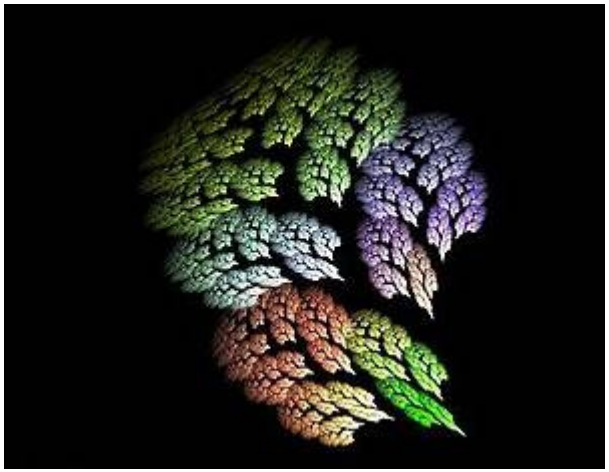
Fractal Tr

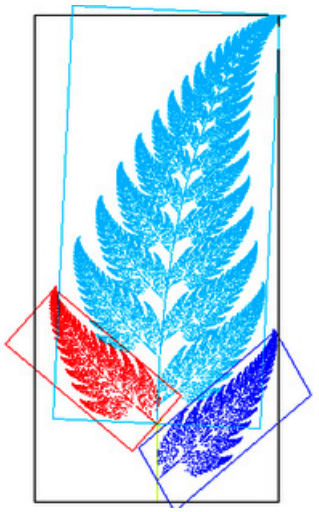


Fractais naturais

estão à nossa volta: as nuvens, as montanhas, os rios e seus afluentes, os sistemas de vasos sanguíneos, os vegetais, florestas, arvores, células e sistema nervosos, etc.

Estas figuras estão classificadas em diversas dimensões não inteiras e sim fracionarias





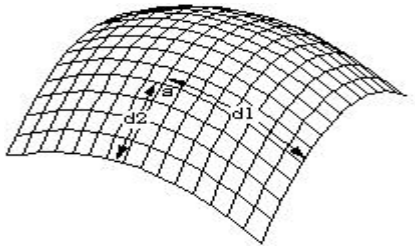
Dimensão Euclidiana – objetos euclidianos

Um objeto de 1 dimensão (por exemplo uma linha),

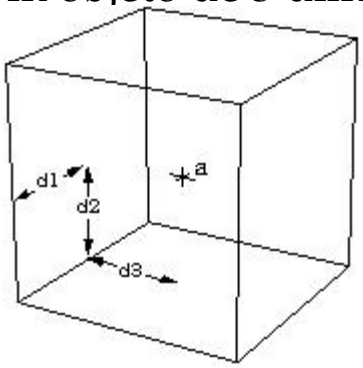
pode ser dividido em N partes , cada parte será idêntica a anterior multiplicada por um fator elevado a 1:

$$(r = 1 / N \quad e \quad N * r^{1} = 1).$$

Um objeto de 2 dimensões (por exemplo um quadrado), cada parte será idêntica a anterior multiplicada por um fator elevado a 2:



$$(r = 1 / N \quad e \quad N * r^{2} = 1).$$

- Um objeto de 3 dimensões (por exemplo um cubo), cada parte será idêntica a


al multiplicada por um fator elevado a 3:

$$(r = 1 / N \quad e \quad N * r^{3} = 1)$$

Pode-se então repensar a definição de Dimensão Euclidiana para que seja

Um objeto terá dimensão d , se ao ser dividido em N partes , cada parte será idêntica a anterior multiplicada por um fator de escala r elevado a d:

$$r = 1/N \quad e \quad N * r^{d} = 1$$

$$Ou \quad N = 1 / r^{d} = (1/ r)^{d}$$

- Aplicando log em ambos os lados:

$$\log N = \log (1/ r)^{d}$$

$$\log N = d \log (1/ r)$$

$$d = \log N / \log (1/r)$$

Dimensão fractal

$$DF = \log N / \log (1/r)$$

Mede a complexidade da fractal em relação ao espaço Euclidiano a que pertence

conjunto de Cantor

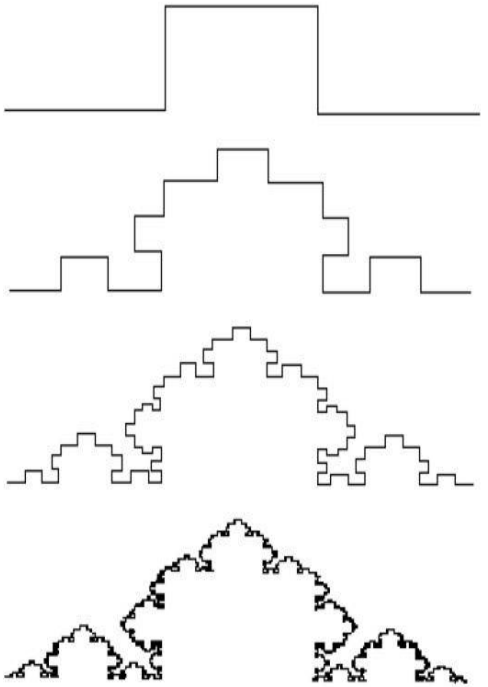
$$DF = \log 2 / \log (3) =$$

0,63092975357145743709952711434276

Dimensão fractal - *DF = log N / log (1/r)*

Mede o quanto é complexa a fractal em relação ao espaço Euclidiano a que pertence

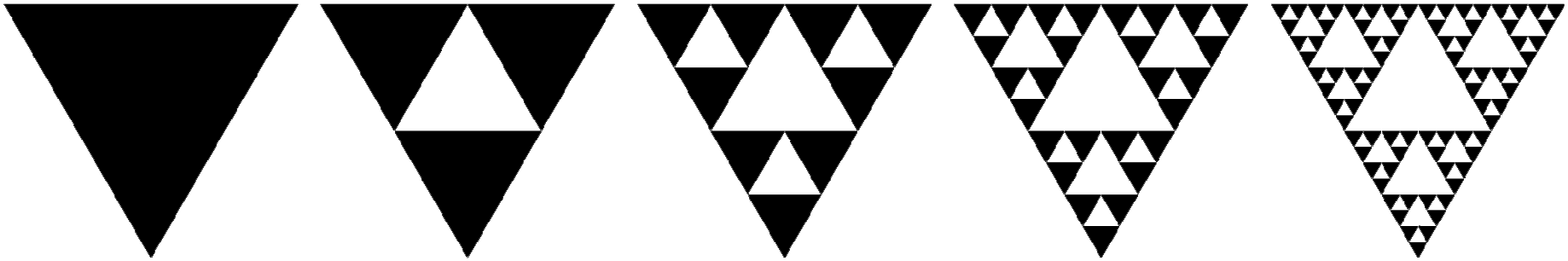
Curva quádrica de Koch



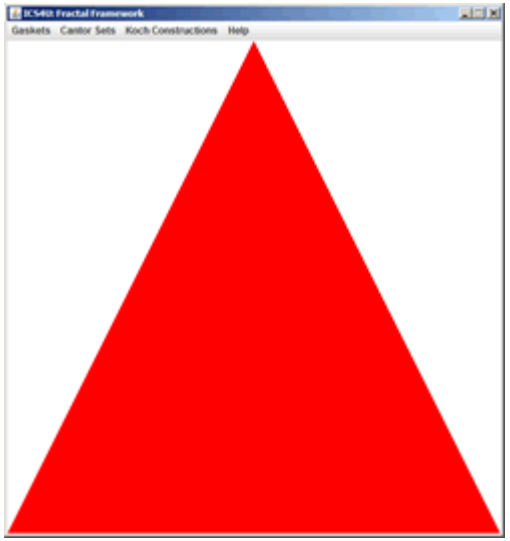
$$DF = log\ 5 / log\ (3) =$$

1,464973520717927167197040404076786

Dimensão fractal *DF = log N / log (1/r)*



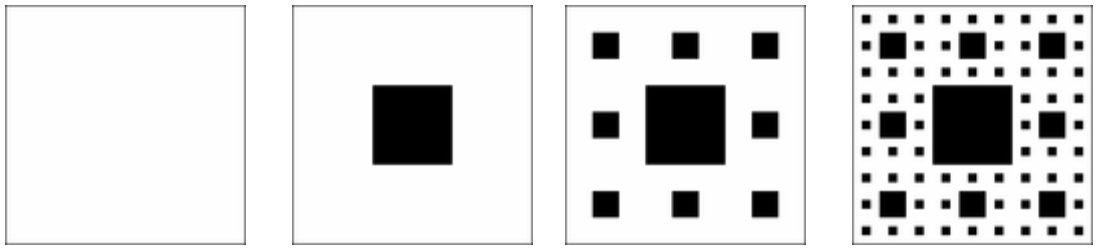
DF do Triângulo de Sierpinski:
DF = log 3 /l og2 =
1,584962500721156181453738943 9478

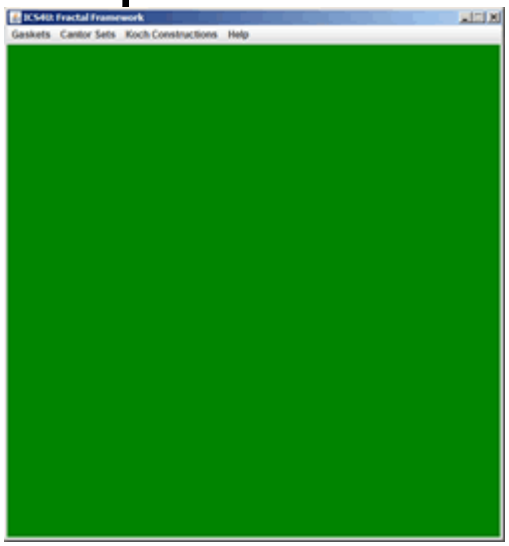


Triângulo de Sierpinski

Tapete de Sierpinski

A construção: subdivide-se cada lado um quadrado em 3 partes e remove-se a parte central. Tem-se então, oito pequenos quadrados, este processo deve ser repetido enquanto houver quadrados





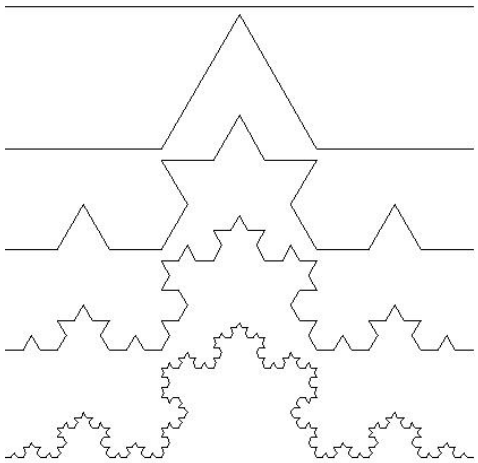
DF = log 8 /l og 3 = 1,892789260714372311298581343028 3

Dimensão fractal $DF = log\ N / log\ (1/r)$

Curva triadic de Koch

$$DF = log\ 4 / log\ (3) =$$

1,261859507142914874199054228685 5

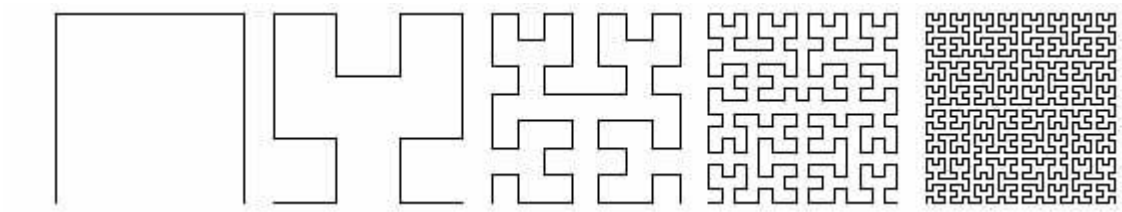


Curva de Peano.

Conhecida também como "curva de Hilbert" é mostrada na figura abaixo

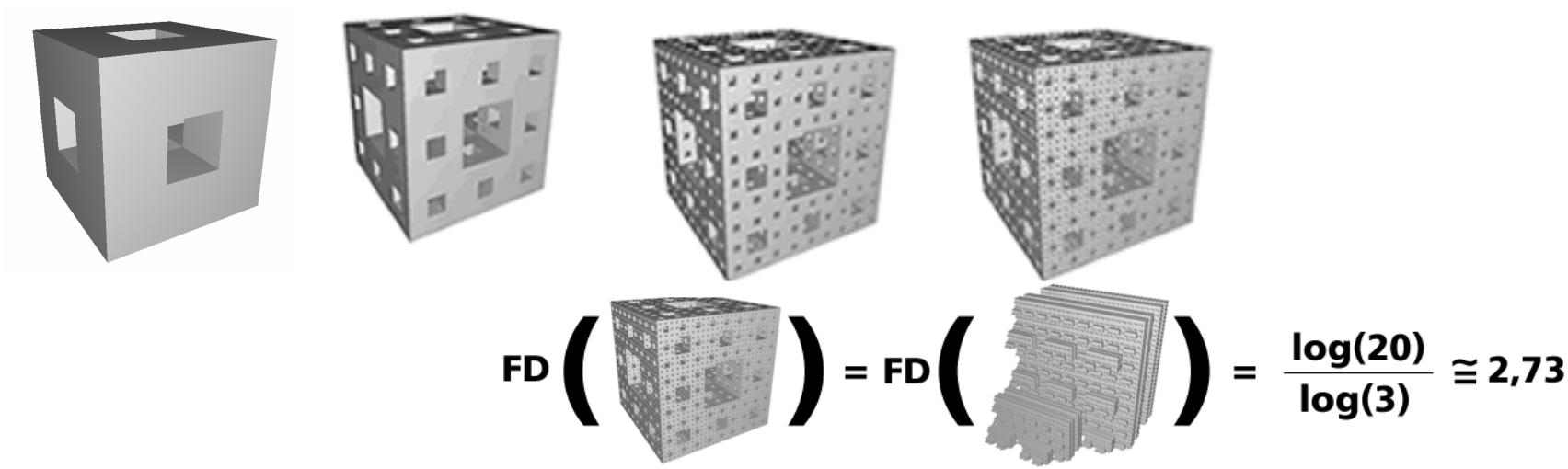
Dimensão fractal *DF = log N / log (1/r)*

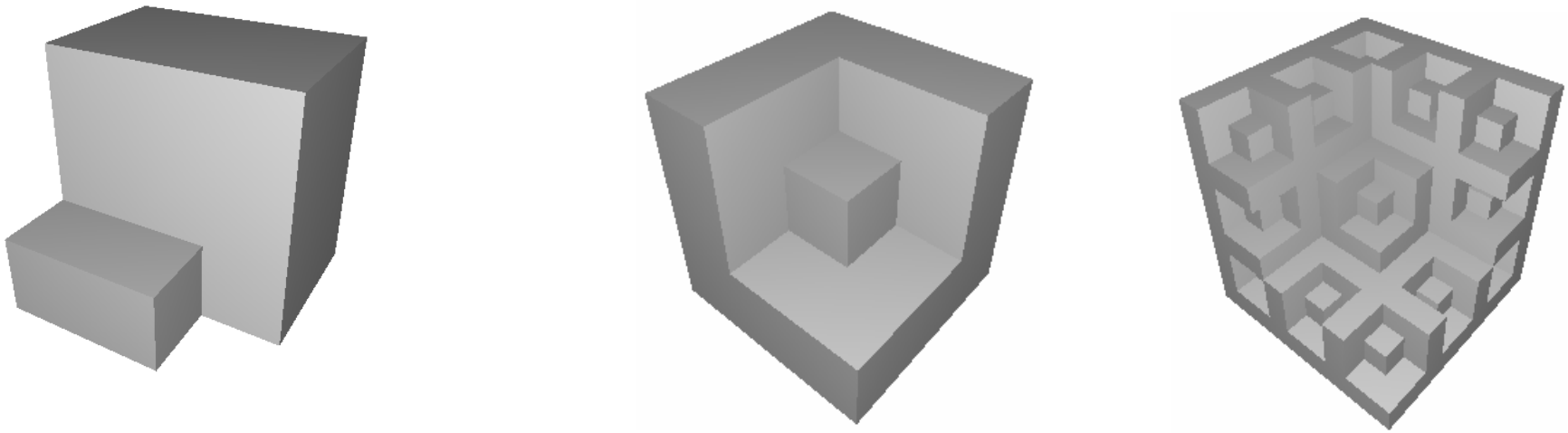
DF = log 4 / log (2) = 2



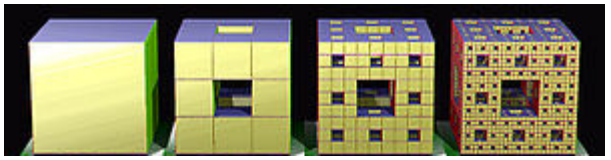
Fractais podem ser caracterizadas pela DF, mas outros ela não caracteriza unicamente um objeto fractal

Dimensão fractal = fracionaria





Esponja de Menger Sierpinski



Construção de uma esponja Menger:

1 - Comece com um cubo;

2 - Divida cada lado do cubo em 3 partes. Isso sub-divide o cubo em 27 cubos pequenos;

3 - Remova o cubo do meio de cada face e o cubo no centro, deixando 20 cubos (segunda imagem). Esta é o Nível 1;

Repita os passos 1-3 para cada cubo que ainda existir.

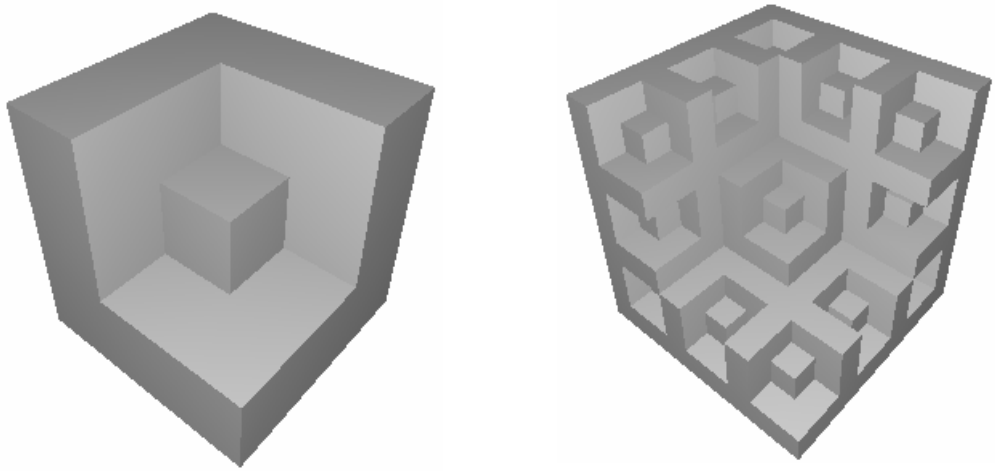
A segunda iteração dará uma esponja Nível 2, a terceira uma esponja Nível 3 , e assim por diante.

A esponja de Menger é o limite deste processo de iterações.

O número de cubos aumenta **20n em cada itera**ção.

Lacunaridade e Sucolaridade tambem podem ser preciso

Local Lacunarity (LL)

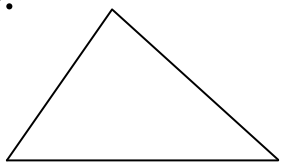


$$\Lambda(s) = \sum_{i=1}^{N} M^2\, Q(M, s) \;/\; \left(\sum_{i=1}^{N} M\, Q(M, s) \right)^2$$

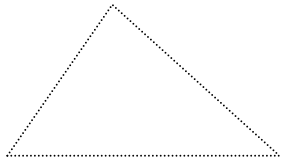
Um algoritmo simples de Montanhas

A superfície de uma montanha pode ser modelada como:

1- Desenhe um triângulo no espaço 3D.



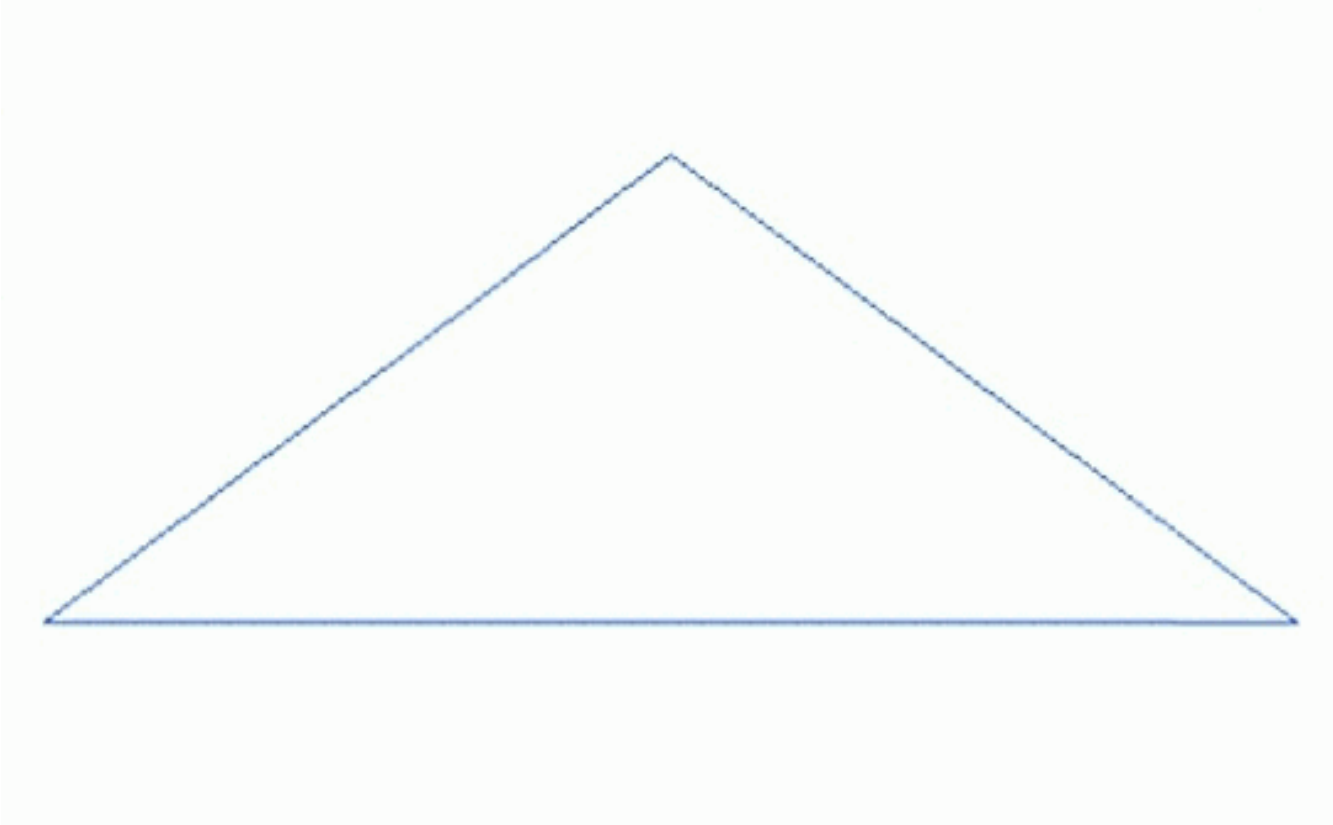
2- Ache os pontos centrais das 3 linhas que formam o triângulo e crie novos triângulos a partir desse triângulo.



3- Desloque aleatoriamente esses pontos centrais para cima ou para baixo dentro de uma gama de valores estabelecido que voce preestabeleça de modo que a superfície fique mais suave ou mais rugoso.

4- Repita 2 e 3 fazendo os deslocamentos dos pontos centrais de modo que em cada iteração é igual a metade da anterior.

Montanha fractal



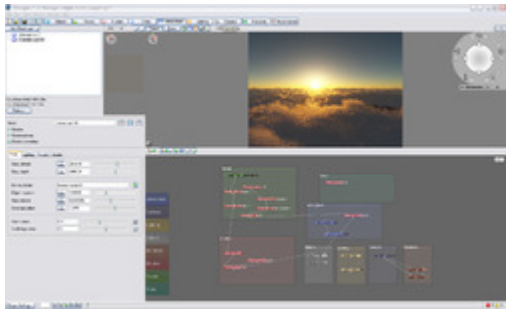
E as formas da natureza, como montanhas, arvores, nuvens, flores, frutas?

Terragen 3.1 é um exemplo de sistema free para modelar cenarios naturais usando Geo. Fractal.

http://planetside.co.uk/tg-31-release

Terragen

Terragen



Geração de cenários

Outerra engine: flora, relevo, agua atraves de algortmos revursivos: http://www.outerra.com/.

L-system ou sistema de Lindenmayer

L-sistemas foram introduzidos em 1968 por Aristid Lindenmayer, um biólogo e botânico húngaro da Universidade de Utrecht.

L-sistemas descrevem o comportamento de células, plantas e animais e também seus processos de crescimento e desenvolvimento.

Também têm sido utilizados para modelar uma variedade de processos, organismos e fractais.

A topologia dos objetos é usada pra gera-los

L-system ou sistema de Lindenmayer

é um tipo de **gramática formal, um autômato.**

Consiste de um **alfabeto de símbolos** e uma coleção de **regras de produção** que se expandem cada símbolo em uma produção maior para gerar figuras.

Seqüência inicial = "axioma"

O mecanismo de traduzir as seqüências geradas produzem as estruturas geométricas.

A ordem (iterações ou nível) de um sistema-L é o número de vezes subdivisão celular ocorre.

um sistema bem simples

Com dois tipos de células representadas pelas letras **A e B.**

A célula A divide-se em duas células representadas por: **A→AB**.

Célula B subdivide-se em duas células representadas por : **B →BA** .

A ordem dos símbolos é relevante e a inicial também.

O organismo nos modelos usando o sistema cresce por repetição.

A célula inicial ao nascer é o **axioma.**

Por exemplo : Se o axioma for a célula **A.**

Depois de uma iteração do organismo : **AB.**

Depois de duas de acordo com as regras acima,

o organismo tem quatro células dadas pela seqüência **ABBA.**

Após três, o organismo é **ABBABAAB** ,

depois de quatro o organismo tem 16 células:

ABBABAABBAABABBA

De maneira condensada pode-se escreve isso como:

ExemploRidiculo {; O nome de sistema-L, seguido de "{"

/* para sinalizar o início de uma descrição :

Axioma := A; /* seqüência de nascimento do organismo

A → AB; /* cadeia de substituição para A

B → BA; /* cadeia de substituição para B

}; Sinalizar o fim da descrição do organismo com um "}"

Turtle graphics

É um método de programação gráfica vetorial usando a idéia de um autômato 3D ("tartaruga") em um espaço ou plano cartesiano. (linguagem Logo).

A tartaruga tem três atributos: **sua localização**

sua orientação

uma caneta com atributos gráficos de cor, tipo de linha, largura, e up and down (desenhar ou não).

Os movimentos da tartaruga são gerados por comandos de **deslocamento para frente ou giro nas 3 direções possíveis em relação a um sistema de eixos cartesiano e ângulos de Euler que passe por ela em relação à sua própria posição**.

A partir desta idéia pode-se construir qualquer desenho, formas complexas, e figuras compostas.

Incluindo recursividade, fica muito útil para gerar sistema de Lindenmayer e fractais.

DAI você pode inventar um monte de símbolos e regras de substituição

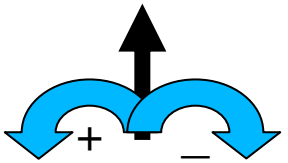
Por exemplo os símbolos **"f" , "+", "-" "!", "F", "@", "[", "]"** são símbolos usados em 2D.

"f" significa "desenhar uma linha na direção da corrente de um **comprimento** definido".

"F" significa "desloque-se em linha reta a partir da direção da corrente de um **comprimento** definido ".

"+" significa "virar à esquerda de um ângulo em graus".

"-" significa "vire à direita de um ângulo em graus".



Os símbolos "[" e "]" trabalhar em conjunto como "pilhas".

"[" Significa "amazene o estado gráficos: posição, orientação, espessura da linha atual, etc."

"]" Significa "redefina o estado de gráficos de volta ao armazenada".

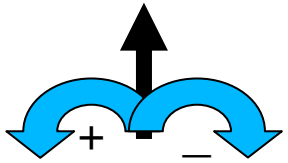
"@" significa "multiplicar o comprimento da linha atual pela quantidade seguinte". Por exemplo "@ 0,5" significa multiplicar o comprimento da linha atual por (× 0,5).

"!" significa "inverter o sentido de + e-.

"x", "y", etc. não tem qualquer significado, é apenas uma célula que se subdivide em células de acordo com uma seqüência que segue "x =", "y =", etc.

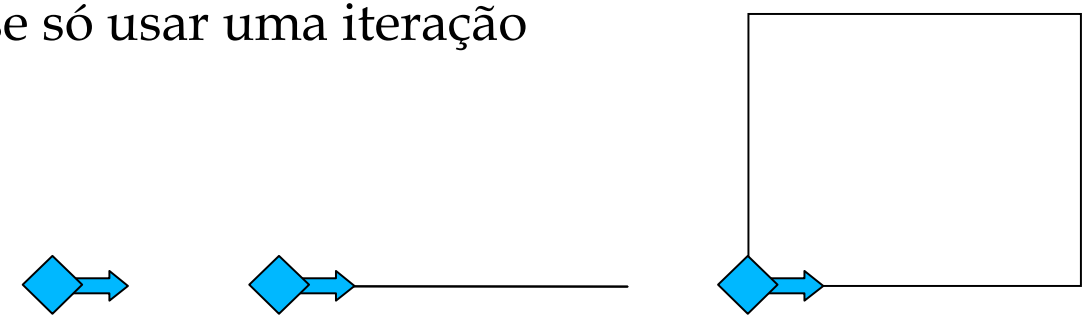
Com isso você pode gerar desenhos, ou texturas procedurais, ou objetos:

Você pode desenhar um quadrado como

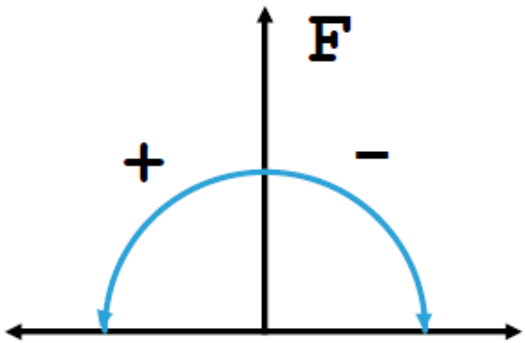


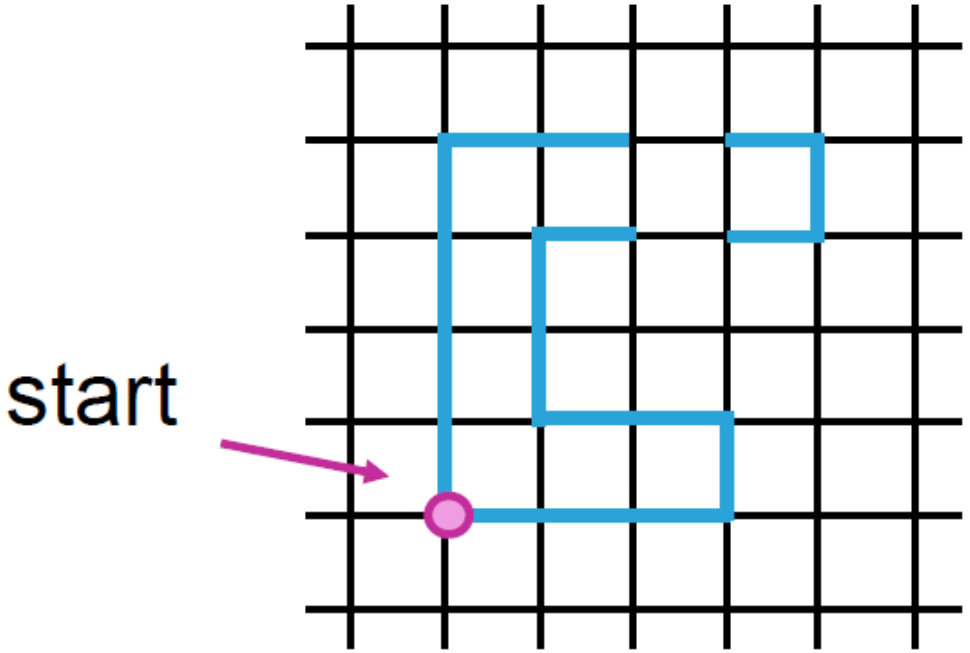

```
quadrado {
  angulo:= 90;  /* giros de 90  graus
  Axioma :=f;  /*  seqüência inicial neste caso se supormos que estamos
    na horizontal e com orientação para da direita para a esquerda
  f ⟶ f + f + f + f;  /* seqüência de substituição para f
        }
```

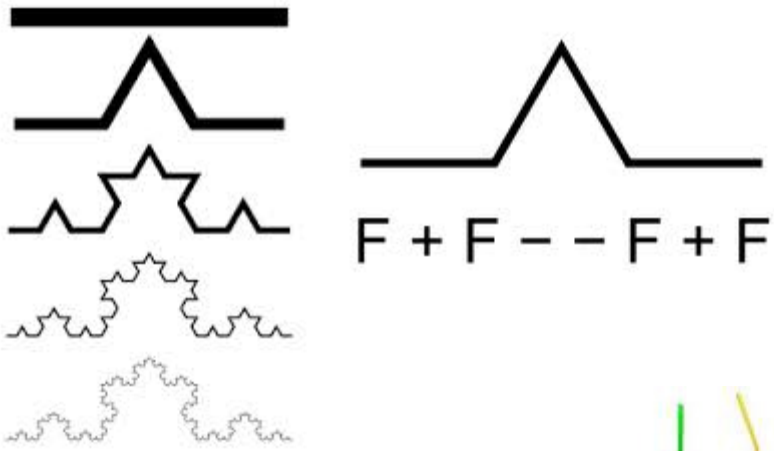
se só usar uma iteração

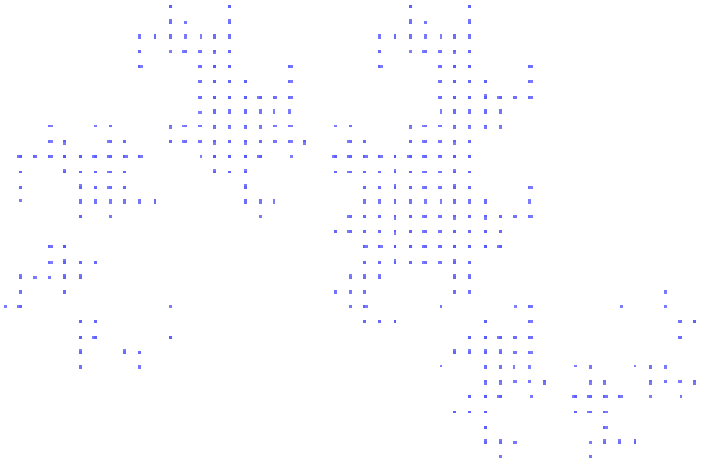


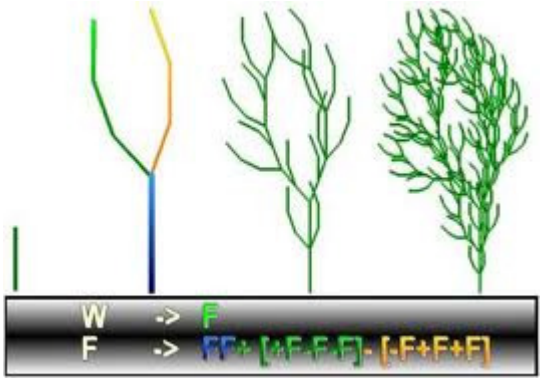
Example: **FFFF-FFfF-F-FfF+FF+FF-F-FFF**,
d=1, δ =90°

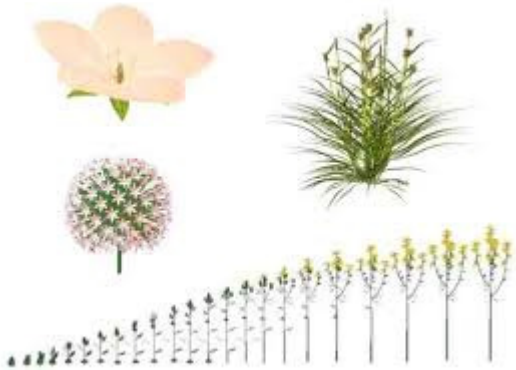


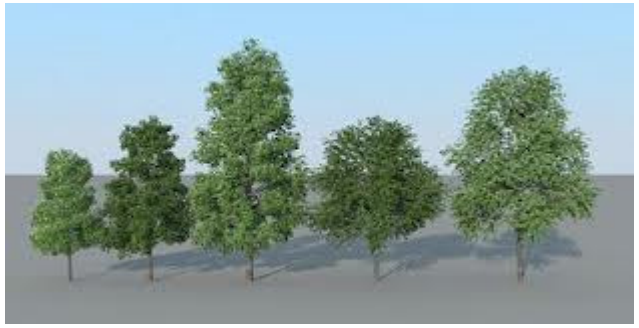



F + F − − F + F

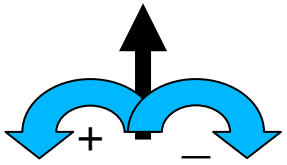



W -> F
F -> FF+ [+F-F-F]- [-F+F+F]





Usar o quadrado como axioma e
gerar a fractal ilha quádrica da Koch, e a desenha-la ate um certo nível de substituição.



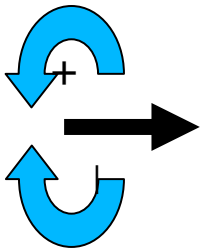
ilhaFractal {

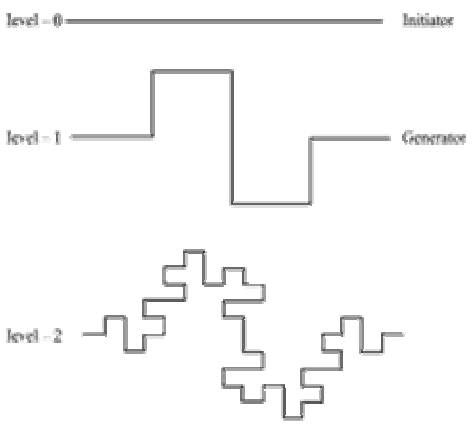
angulo:= 90; /* giros de 90 graus

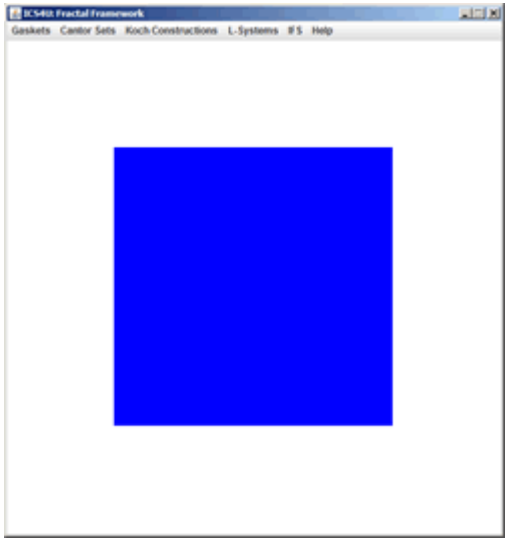
Axioma:= f-f-f-f; /* seqüência inicial neste caso se supormos que estamos na horizontal e com orientação para a direita

f → @ 0,5 f +f - f - f f + f + f - f; /* seqüência de substituição

}





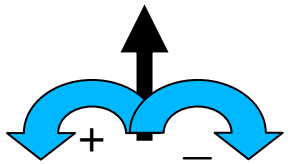


Com isso você pode gerar desenhos, ou texturas procedurais, ou objetos:

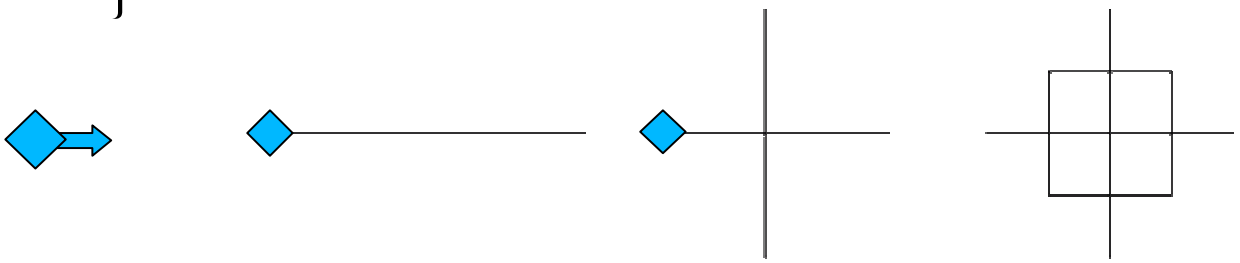
Você pode pensar em um quadrado como uma célula que se subdivide em quatro quadrados.

Cada um desses quadrados se subdivide em quatro quadrados e assim por diante.

Vamos olhar para o seguinte L-system.




```
grade {
  angulo:= 90;  /* giros de 90  graus
  Axioma f;  /*  seqüência inicial neste caso se supormos que estamos
    na horizontal e com orientação para da direita para a esquerda
  f → @0,5 f [+ f] [- f] f;  /* seqüência de substituição para f
        }
```

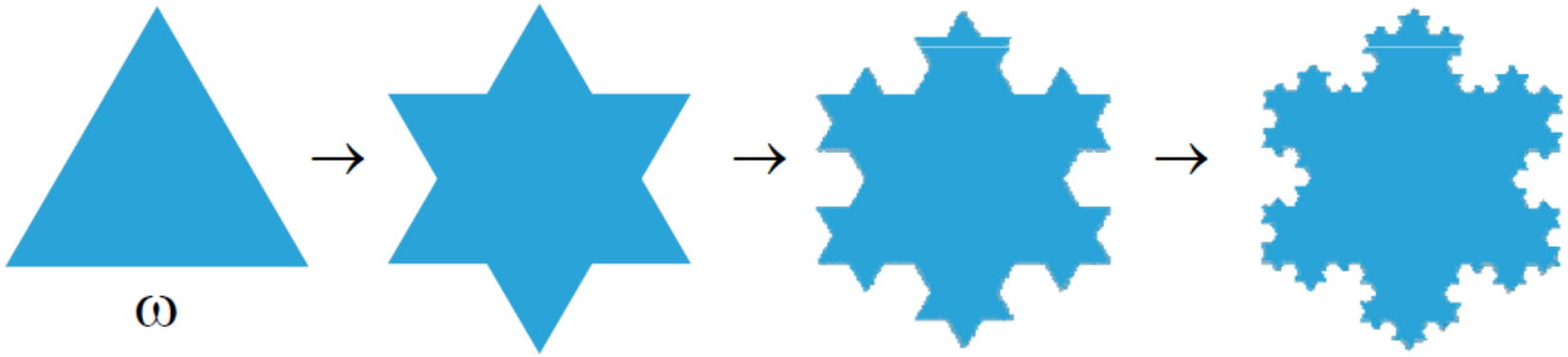


- Construction of Koch's island:

  δ = 60°, d depends on derivation length

  ω = **F--F--F** //Initiator

  F $\rightarrow$ **F+F--F+F** //Generator



E do mesmo modo as demais fractais que já vimos neste capitulo. Observe pode-se usar qualquer número de regra de substituição

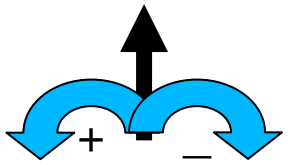
Por exemplo a curva de Hilbert é feita dividindo-se um quadrado de lado L em 4 outros de lado L/2. Junta-se os centros de quadrados adjacentes com um segmento de linha, de modo que as linhas não se cruzem.

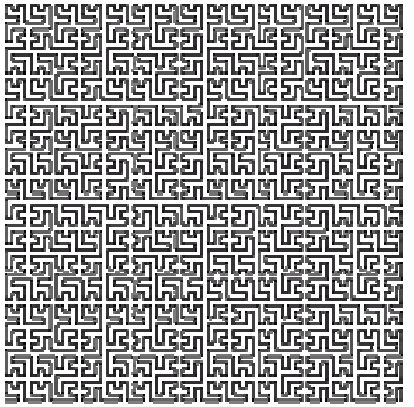
Hilbert { angulo:= 90; axioma := X;

X → @0,5 -YF+XFX+FY- ;

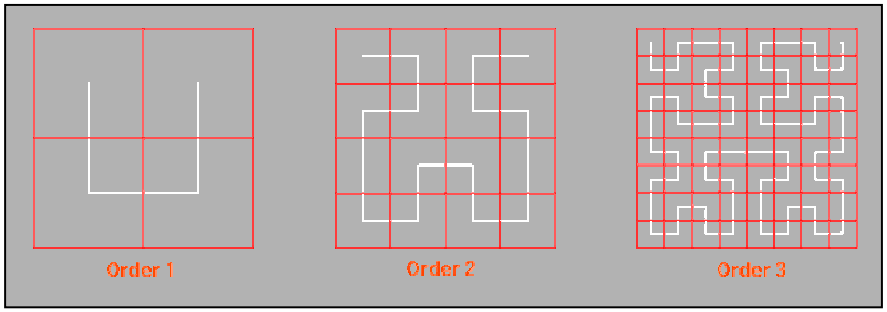
Y → +XF-YFY-FX+ ;

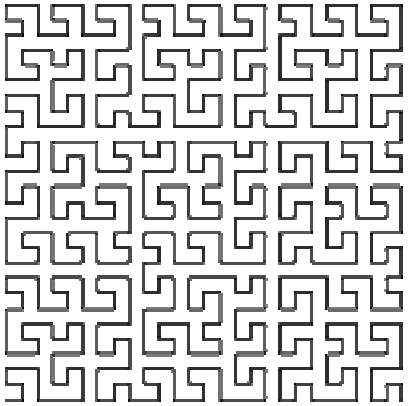
}



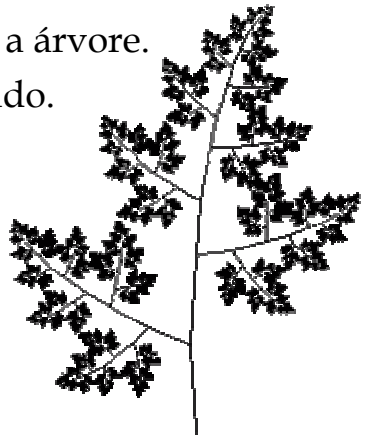


(curva de Hilbert de ordens 1, 2 e 3 com o quadrado,e depois só as linhas)





Plantas :



Olhe para uma árvore. Um ramo da árvore se parece com uma versão menor de toda a árvore.

O exemplo abaixo usa essa idéia para modelar uma **avenca** ou uma **erva** como a ao lado.

Avenca { ângulo := 7,2^{o} ; Axioma := x

x → f [@ 0,5 +++++++++ x] -f [@0,4 - - - - - - - - - - -! x] @. 6x

}

Agora vamos traduzir estes para comandos:

O símbolo "x" representa a erva.

Axioma: como ela começa , neste caso se supormos que estamos no ponto que a queremos desenhar e estamos com orientação para cima o **axioma não faz nada**.

O texto de substituição para a erva **x** diz desenhar uma linha (f), o caule da planta.

Em seguida, armazenar essa posição na memória.

Depois diminuir o comprimento a metade (**×**0,5), vire à esquerda 9x7,2=64,8 graus, substituir pela "erva"=x, e voltar para a posição de memória anterior = haste . Assim ([@ 0,5 +++++++++ x]) gera uma haste menor da erva.

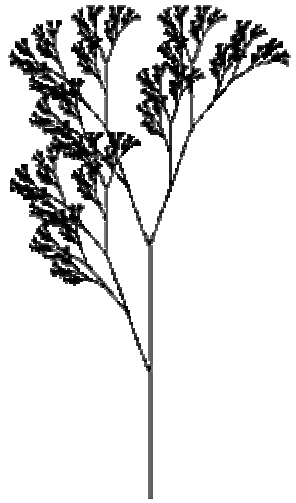
Em seguida, vire à direita ligeiramente (7,2 graus) e desenha outra parte da haste (-f).

Encolhe-se uma pouco o comprimento da erva, por 0,4, vira-se à direita 11 * 7.2 = 79.2 graus, inverter o sentido de direita e esquerda, substituir pela erva e retornar para a última posição : ([@ 0,4 ---------- -! x]).

Finalmente, muda-se o comprimento de 0,6 (@ .6x) e substituir pela erva .

A figura ao lado foi gerada, usando como critério de parada chegar o cpmprimento chegar a 1 pixel

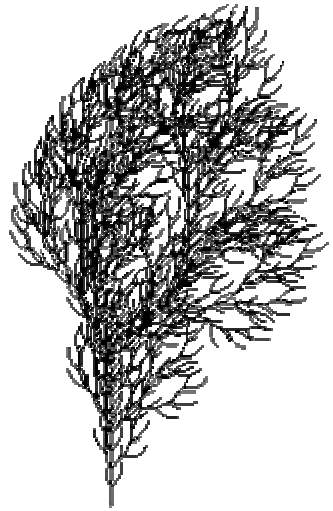
Como Lindenmayer (1925-1989) era botânico, com isso ele registrou o crescimento e forma de uma infinidade de espécies.



Plantinha {angulo=: 20º; axioma:= X;

X → F[+X]F[-X]+X F=FF

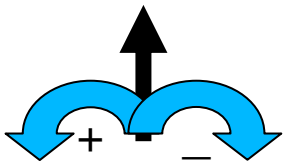
}



Arbusto {angulo:= 22,5º ; Axioma:=F ;

F⟶FF-[-F+F+F]+[+F-F-F]

}



Mas os L systemas permitem muito mais que só isso.

até agora tudo o que se viu foram construções determinísticas e que oceoreem sempre da mesma forma, isto é em quqlquer contexto. Os chamados:

D0L-sistemas, pronunciado "dee-Zero-ell-sistemas", são a classe mais simples de L-sistemas.

Os símbolos "0D" indicam “de livre de contexto" e "D" denota que é determinista.

Stochastic L-systems

Context sensitive L-systems

Parametric L-systems

Stochastic OL-system -

Todas as plantas geradas pelo mesmo sistema L determinista são idênticos.

Uma tentativa de usa-las no mesmo cenário iria produzir um impressionante regularidade artificial.

A fim de evitar este efeito, é necessário introduzir variações que vão preservar os aspectos gerais de uma planta, mas irá modificar seus detalhes.

A variação pode ser conseguida por randomização no sistema-L.

Mas aleatoriedade sozinha tem um efeito limitado.

Enquanto os aspectos geométricos de uma planta - tais como as hastes de comprimentos e ângulos de ramificação - são modificados, a topologia subjacente permanece inalterada.

Mas a estocástica pode afetar a topologia e a geometria da planta.

Um sistema de-OL estocástico é definido como G = (V, W, P, W).

O alfabeto V, o W axioma e o conjunto de produções P são definidos como em um sistema-OL

Mas se inclui uma distribuição de probabilidade, para qualquer produção. A soma das probabilidades de todas as produções é igual a 1.

Assim, diferentes produções com o mesmo antecessor pode se

w: P1: F $\xrightarrow{1/3}$ F [+ F] F [- F] F

P2: F $\xrightarrow{1/3}$ F [+ F] F

P3: F $\xrightarrow{1/3}$ F [- F] F

Cada produção pode ser selecionada

com a mesma probabilidade de 1/3.

Mas depois de 5 iterações as possibilidades de repetição

São muito pequenas como a=na figura ao lado

Stochastic L-Systems

Stochastic L-Systems

Context-sensitive L-systems : 1L e 2L systems

A produção também pode depender do contexto do antecessor ou sucessor.

Este efeito é útil para simular as interações entre as partes da planta, por exemplo, devido ao fluxo de nutrientes.

Sistemas de 2L usam produções de forma $e < a > d \rightarrow X$,

onde a letra **a** (o predecessor) pode produzir **X** se e somente se um é precedido por ***e***

e seguido por ***d***

As produções sensíveis ao contexto são assumidos ter precedência sobre produções livres de contexto com o mesmo predecessor estrito.

Consequentemente, uma produção contextual e outro não tanto se aplicam a uma determinada letra, a uma sensível ao contexto deve ser selecionada.

O exemplo a seguir 1L faz uso do contexto para simular a propagação de sinal em toda a cadeia de símbolos:

w: baaaaaaaa

P1 : b<a → b

P2 : b → a

baaaaaaaaaa
abaaaaaaaaa
aabaaaaaaaa
aaabaaaaaaa
aaaabaaaaaa
aaaaabaaaaa
aaaaaabaaaa
aaaaaaabaaa
aaaaaaaabaa

Exemplos:

a n=30,δ=22.5°
#ignore: +-F
F1F1F1
0 < 0 > 0 → 0
0 < 0 > 1 → 1[+F1F1]
0 < 1 > 0 → 1
0 < 1 > 1 → 1
1 < 0 > 0 → 0
1 < 0 > 1 → 1F1
1 < 1 > 0 → 0
1 < 1 > 1 → 0
* < + > * → -
* < - > * → +

b n=30,δ=22.5°
#ignore: +-F
F1F1F1
0 < 0 > 0 → 1
0 < 0 > 1 → 1[-F1F1]
0 < 1 > 0 → 1
0 < 1 > 1 → 1
1 < 0 > 0 → 0
1 < 0 > 1 → 1F1
1 < 1 > 0 → 1
1 < 1 > 1 → 0
* < + > * → -
* < - > * → +

c n=26, δ=25.75°
#ignore: +-F
F1F1F1
0 < 0 > 0 → 0
0 < 0 > 1 → 1
0 < 1 > 0 → 0
0 < 1 > 1 → 1[+F1F1]
1 < 0 > 0 → 0
1 < 0 > 1 → 1F1
1 < 1 > 0 → 0
1 < 1 > 1 → 0
* < - > * → +
* < + > * → -

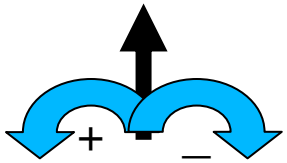
d n=24, δ=25.75°
#ignore: +-F
F0F1F1
0 < 0 > 0 → 1
0 < 0 > 1 → 0
0 < 1 > 0 → 0
0 < 1 > 1 → 1F1
1 < 0 > 0 → 1
1 < 0 > 1 → 1[+F1F1]
1 < 1 > 0 → 1
1 < 1 > 1 → 0
* < + > * → -
* < - > * → +

Context Sensitive L-Systems

Context-sensitive L-systems -: 1L e 2L systems

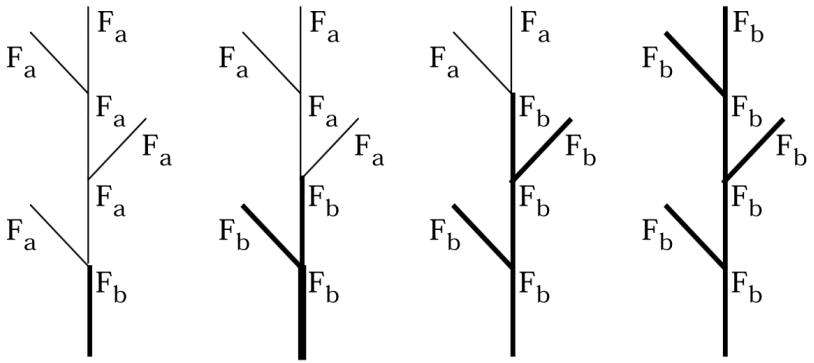
Por exemplo, o seguinte sistema de 1L-simula a propagação de um sinal de em uma estrutura ramificada que não cresce.

Aparece um novo simbolo o **# ignore : + -**



#ignore : +−

ω : $F_b[+F_a]F_a[-F_a]F_a[+F_a]F_a$
p_1 : $F_b < F_a \rightarrow F_b$

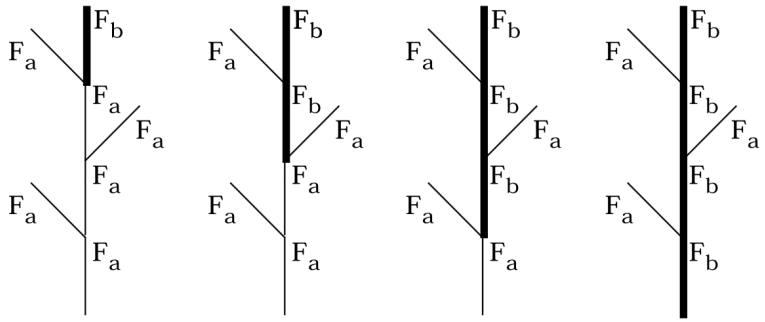


Signal propagation in a branching structure: acropetal,

#ignore : +−

ω : $F_b[+F_a]F_a[-F_a]F_a[+F_a]F_a$
p_1 : $F_b < F_a \rightarrow F_b$

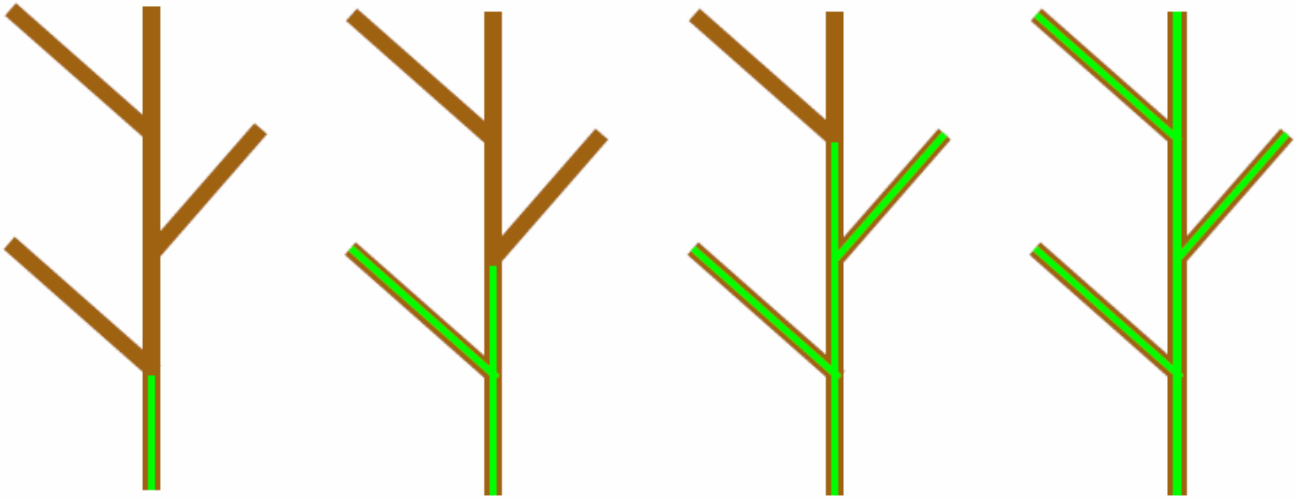
basipetal



Context Sensitive L-Systems

Mudando a cor!

```
# ignore +-
ω:  Fb[+Fa]Fa[-Fa]F[+Fa]Fa
p:  Fb< Fa → Fb
```



PL-systems

Parametric L- systems

Qualquer paramento também pode aparecer nas **produções os nas condições (lógicas ou aritméticas)** a direita e esquerda.

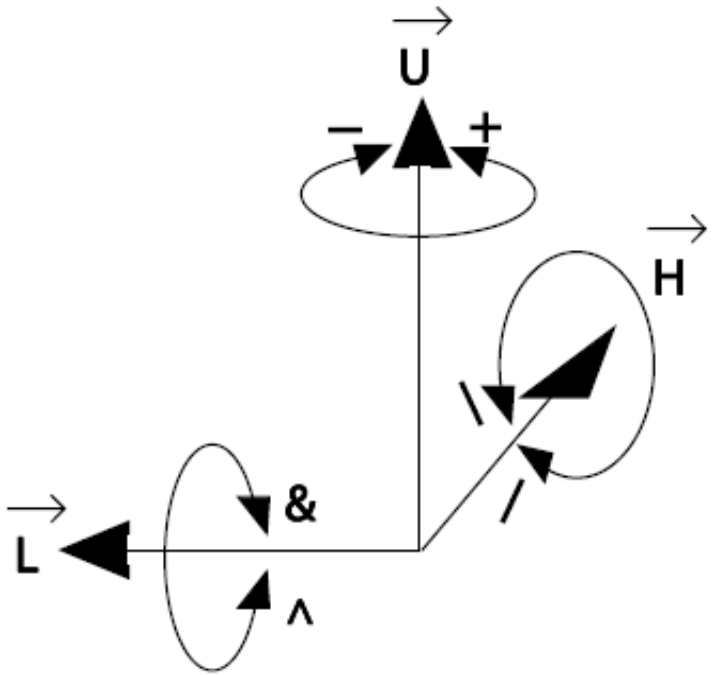
Neste caso a produção só ocorre se a condição total for satisfeita

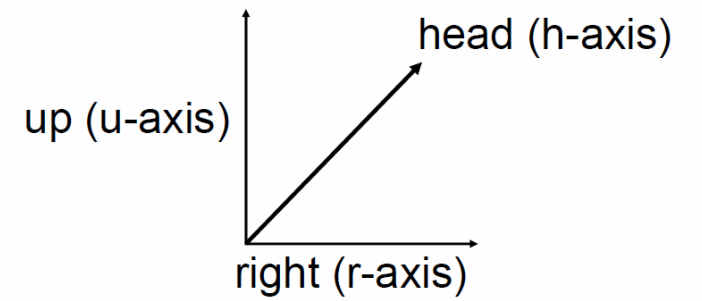
$$A(t) : t > 5 \rightarrow B(t+1)CD(t \wedge 0.5, t-2)$$

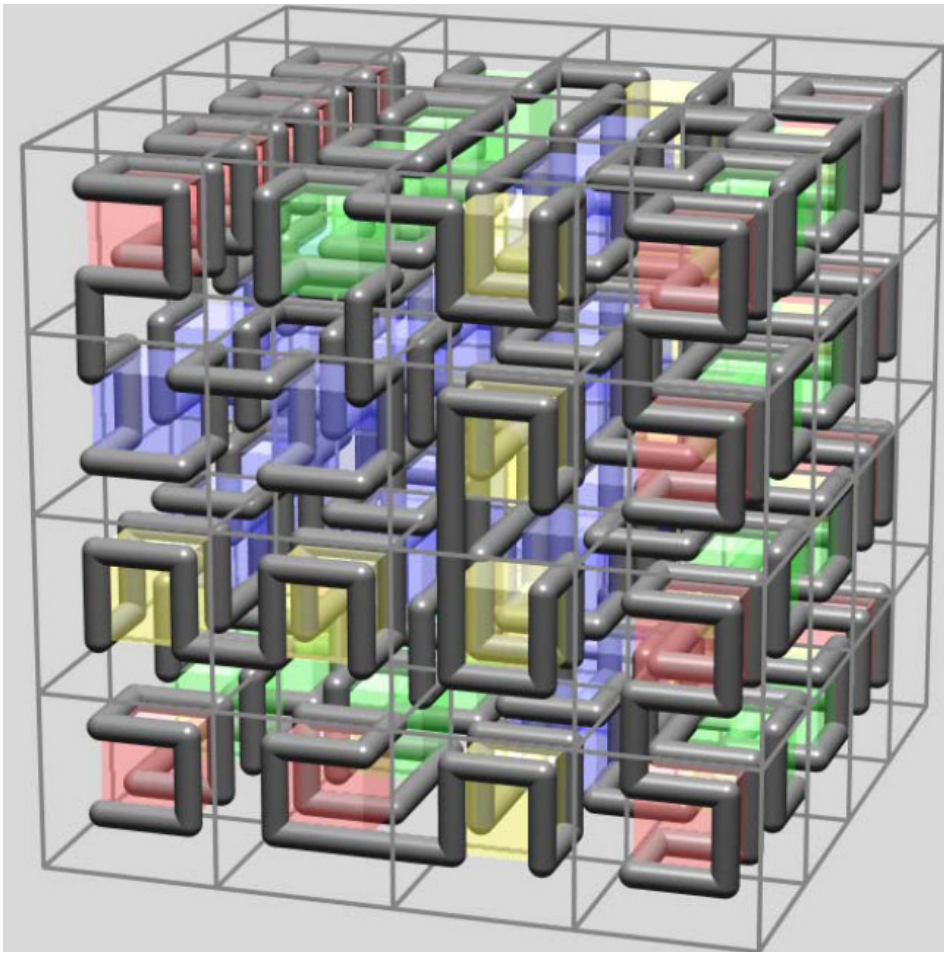
$$A(x) < B(y) > C(z) : x + y + z > 10 \rightarrow E((x+y)/2)F((y+z)/2)$$

$$\cdots A(4)B(5)C(6) \cdots \qquad\qquad E(4.5)F(5.5)$$

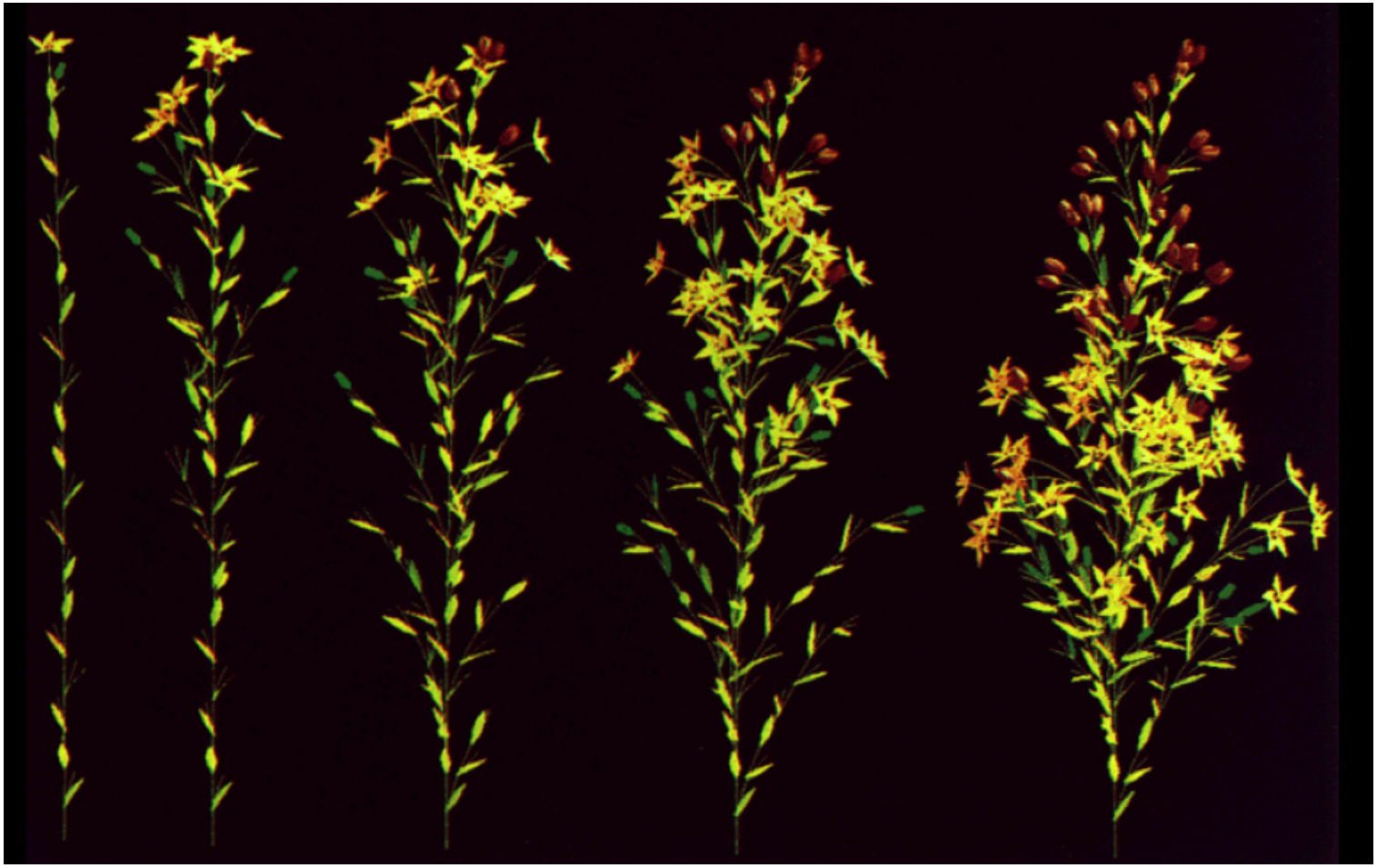
The 3D-Turtle



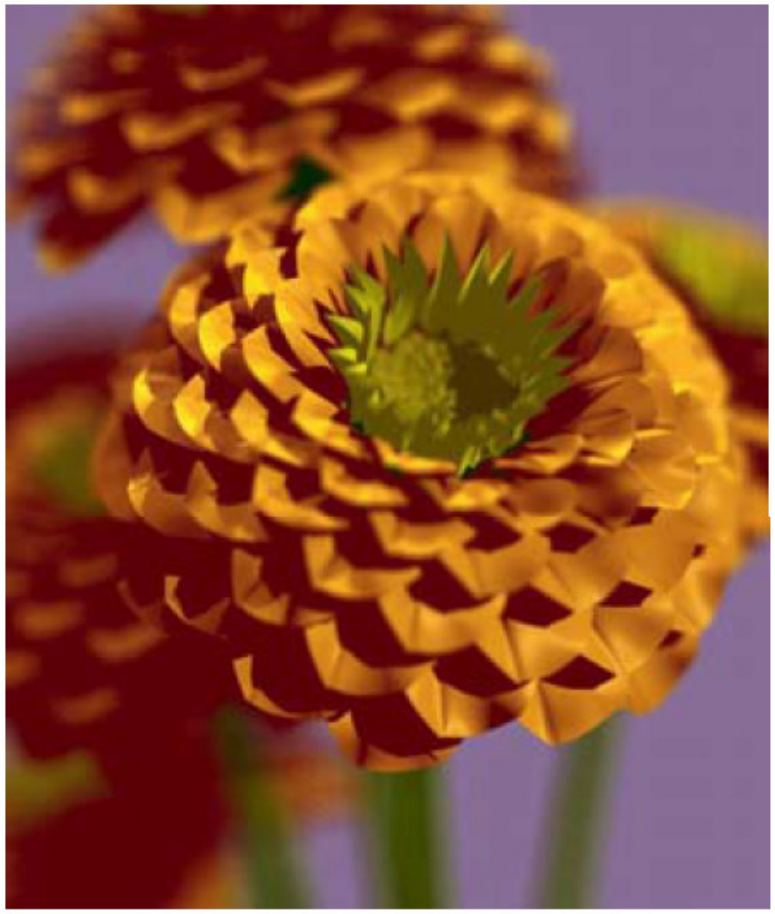




Desenvolvimento de florecencias



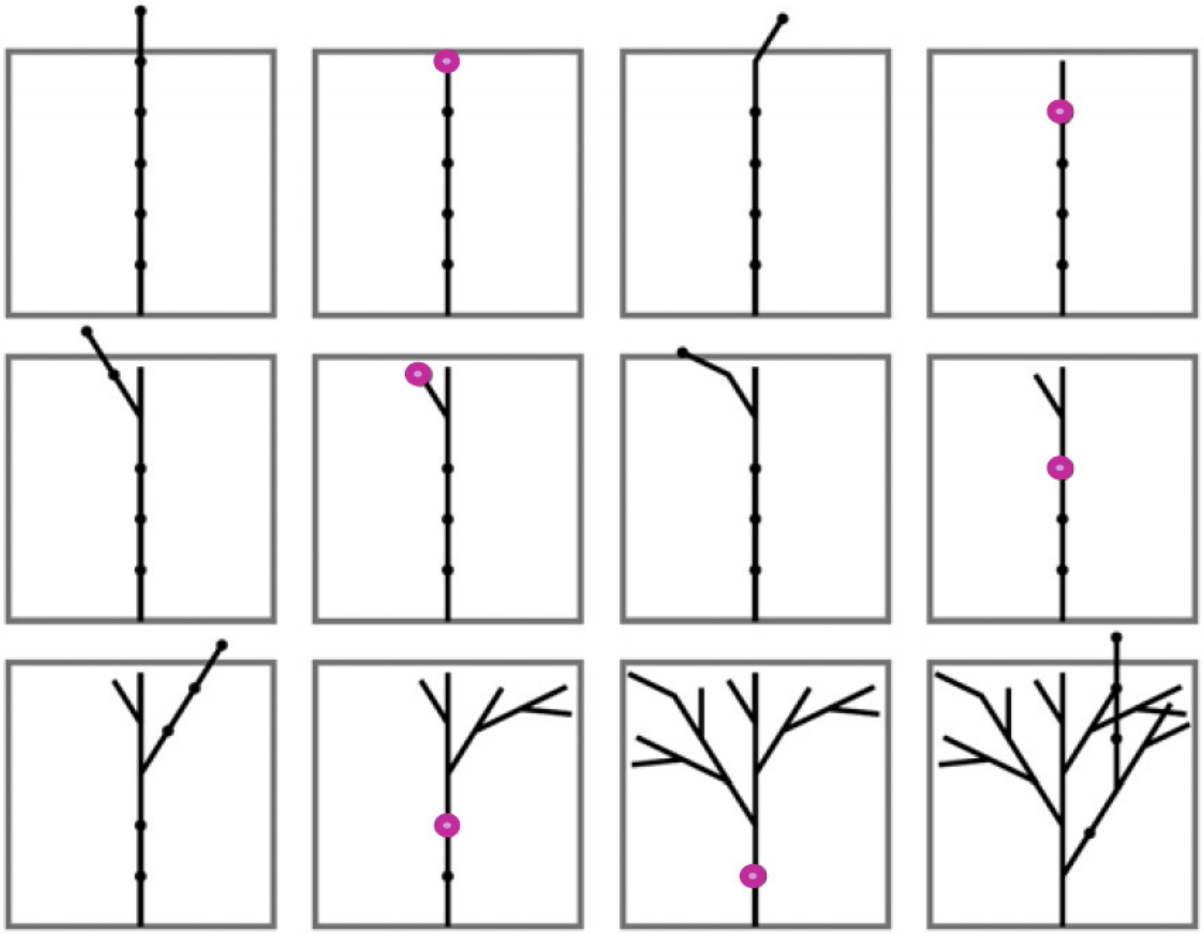
Padrão de crescimento de pétalas, folhas, sementes: seqüência de Fibonacci e Lucas



Exemplo combinados

Produções sensíveis ao ambiente

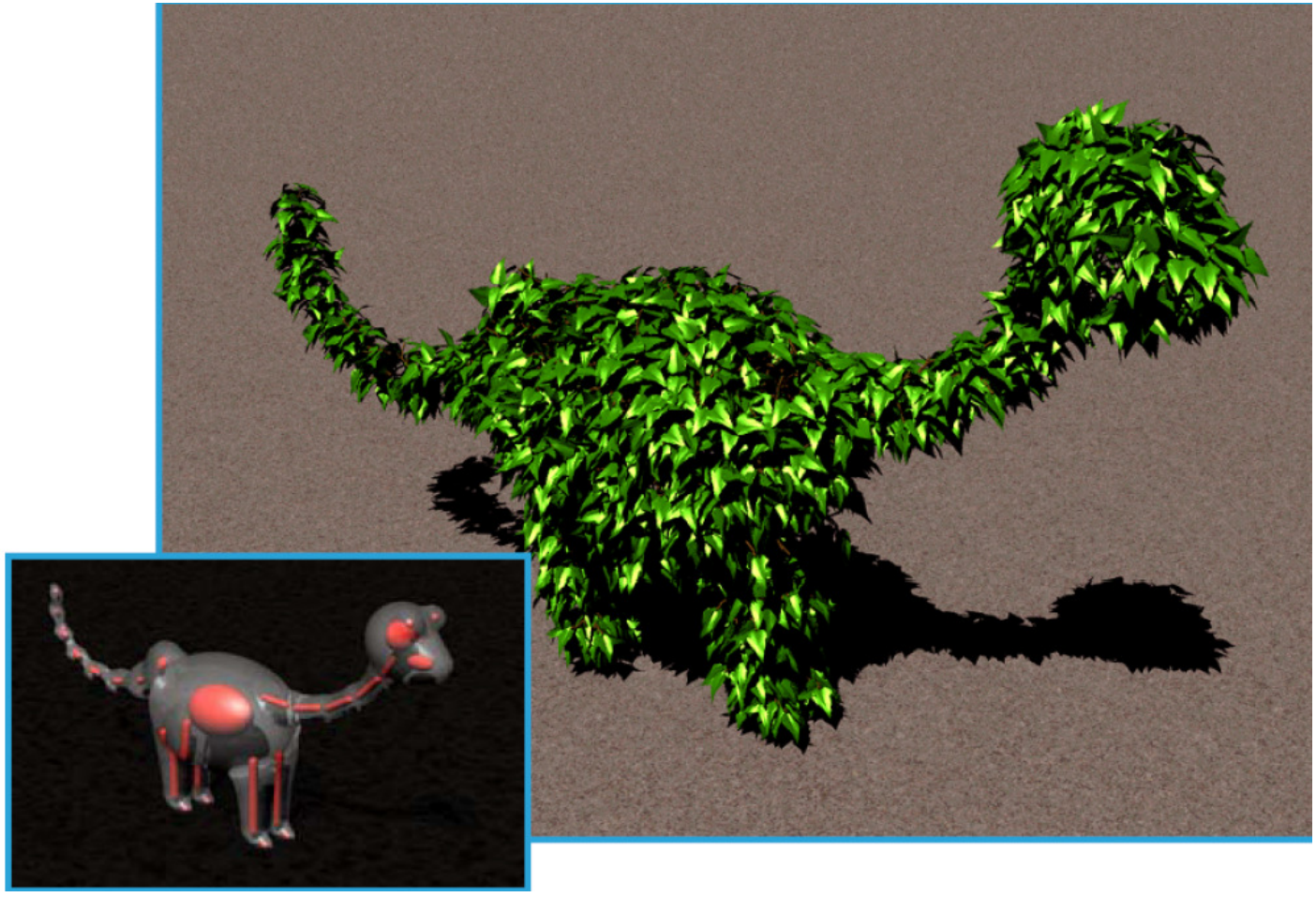
Por exemplo verifica se chegou ao teto da estufa, se sim ramifica



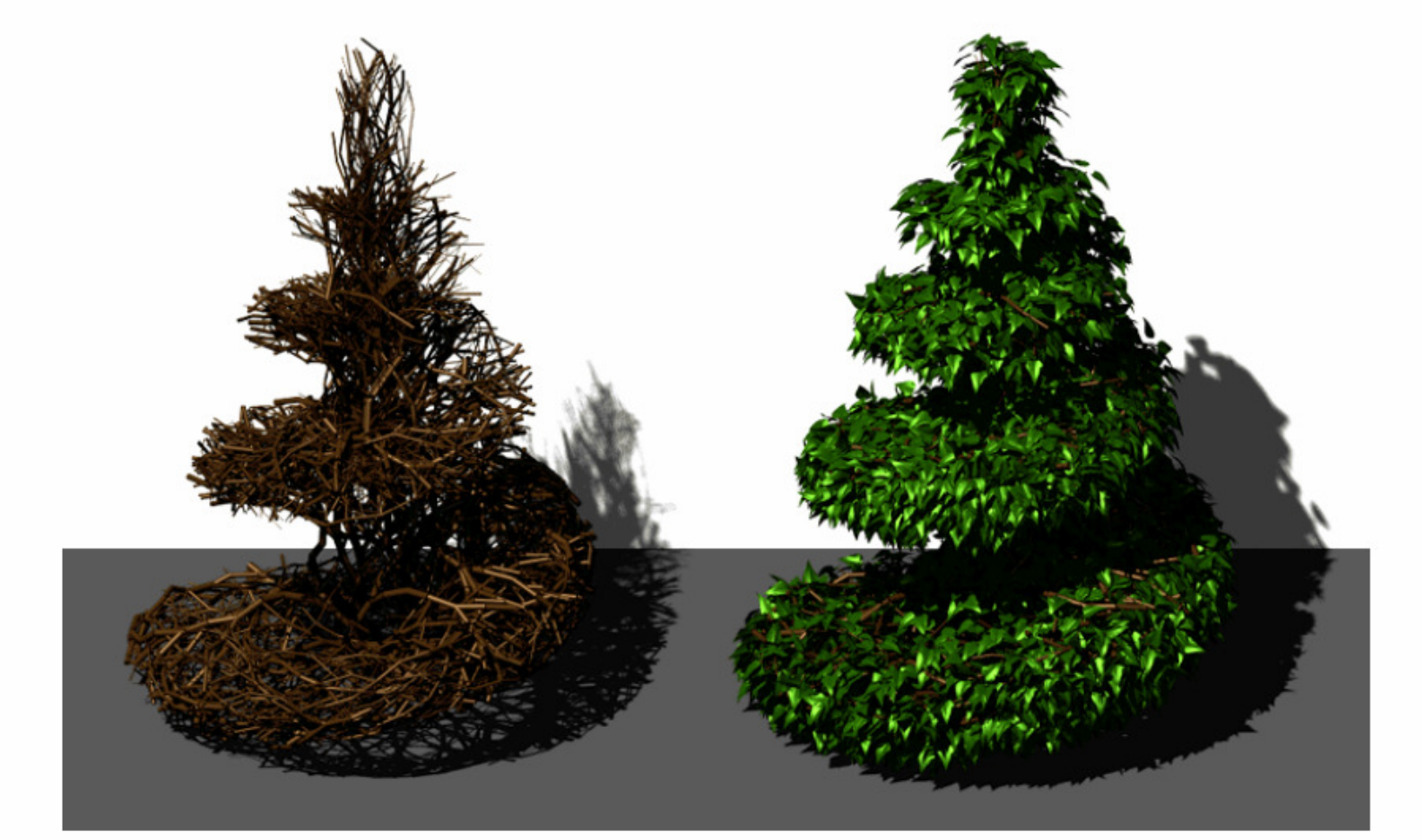
Synthetic Topiary

Synthetic Topiary

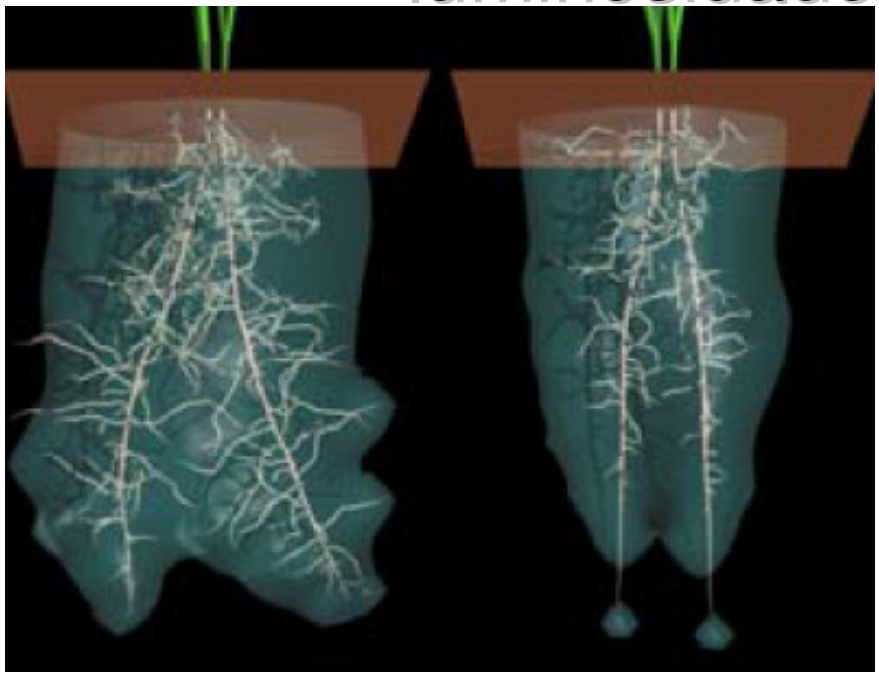
Synthetic Topiary

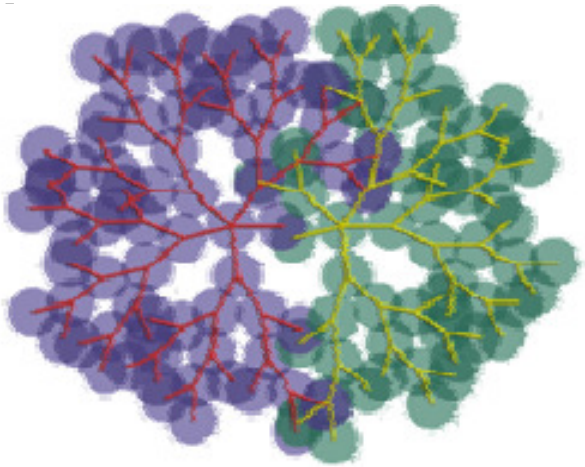


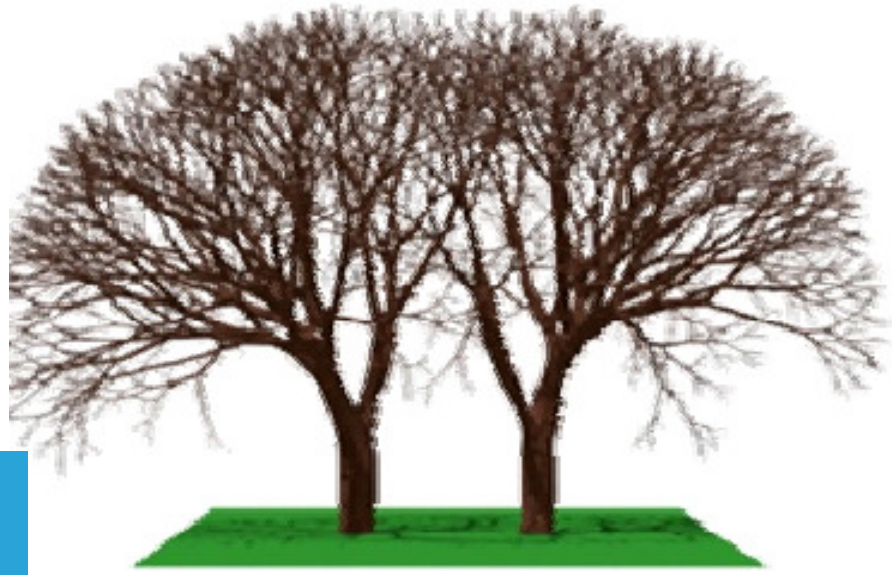
A produção verifica se algo ocorre não ambiente , depois de cada iteração , o que é usado para selecionar produções ou definir estados.



Podem checar condições biológicas do terreno (umidade, composição do solo) ou do ambiente (vento, luminosidade, etc.) :







Open L-Systems

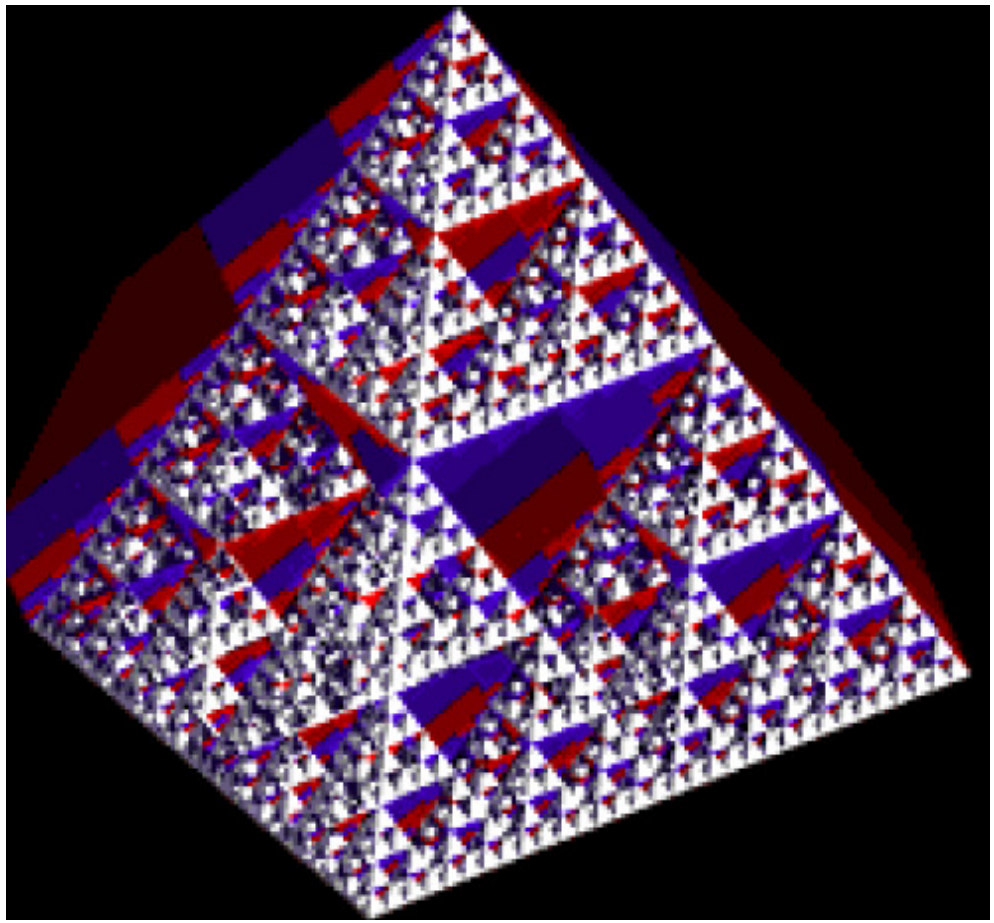
Open L-Systems

Efficient Rendering of pL-Systems

O rendering das fractais pode ser complexo;

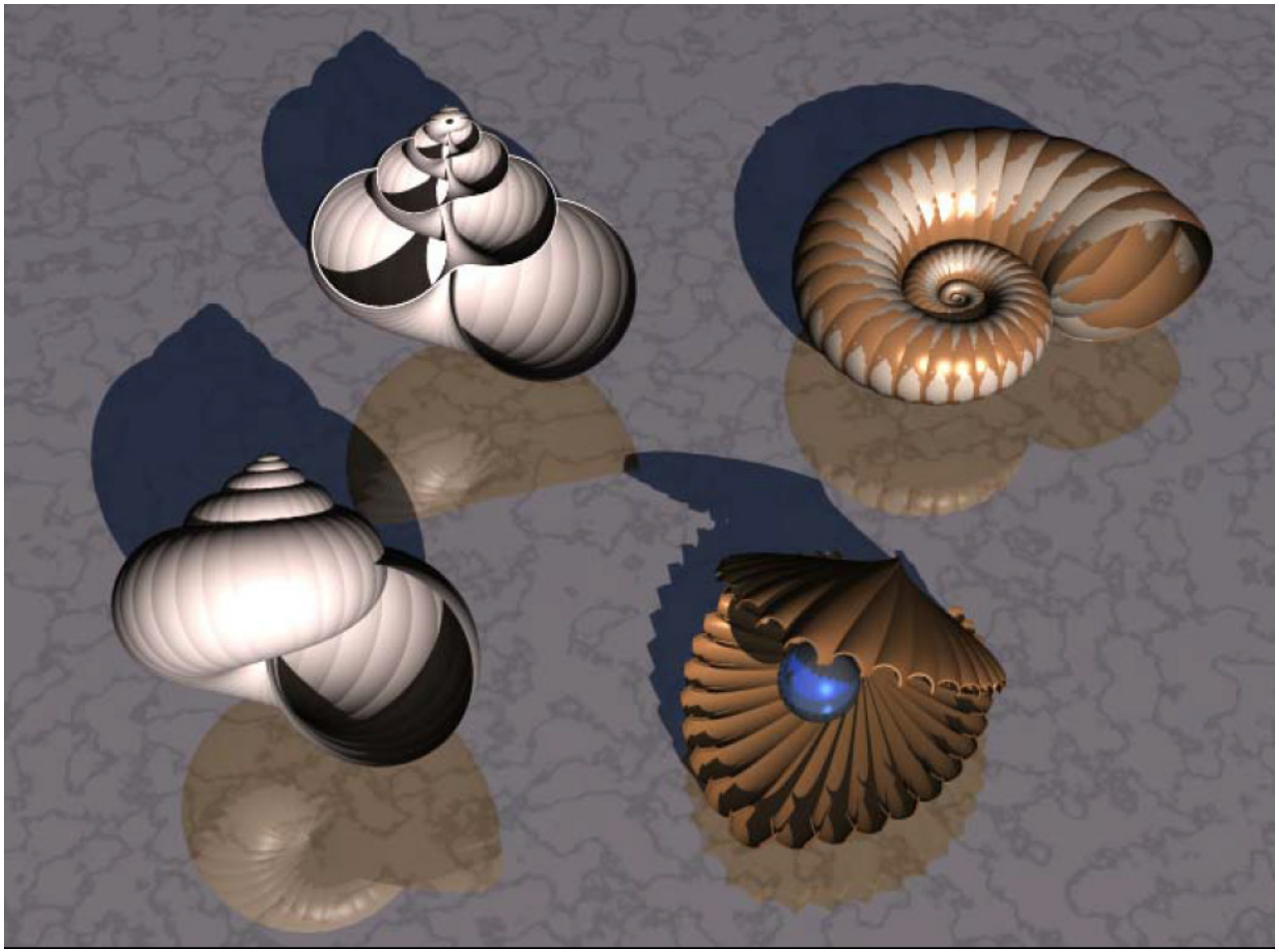
Uma forma pode ser usar bounding box de texturas ou cores e transformá-lo juntos com a criação do objeto.

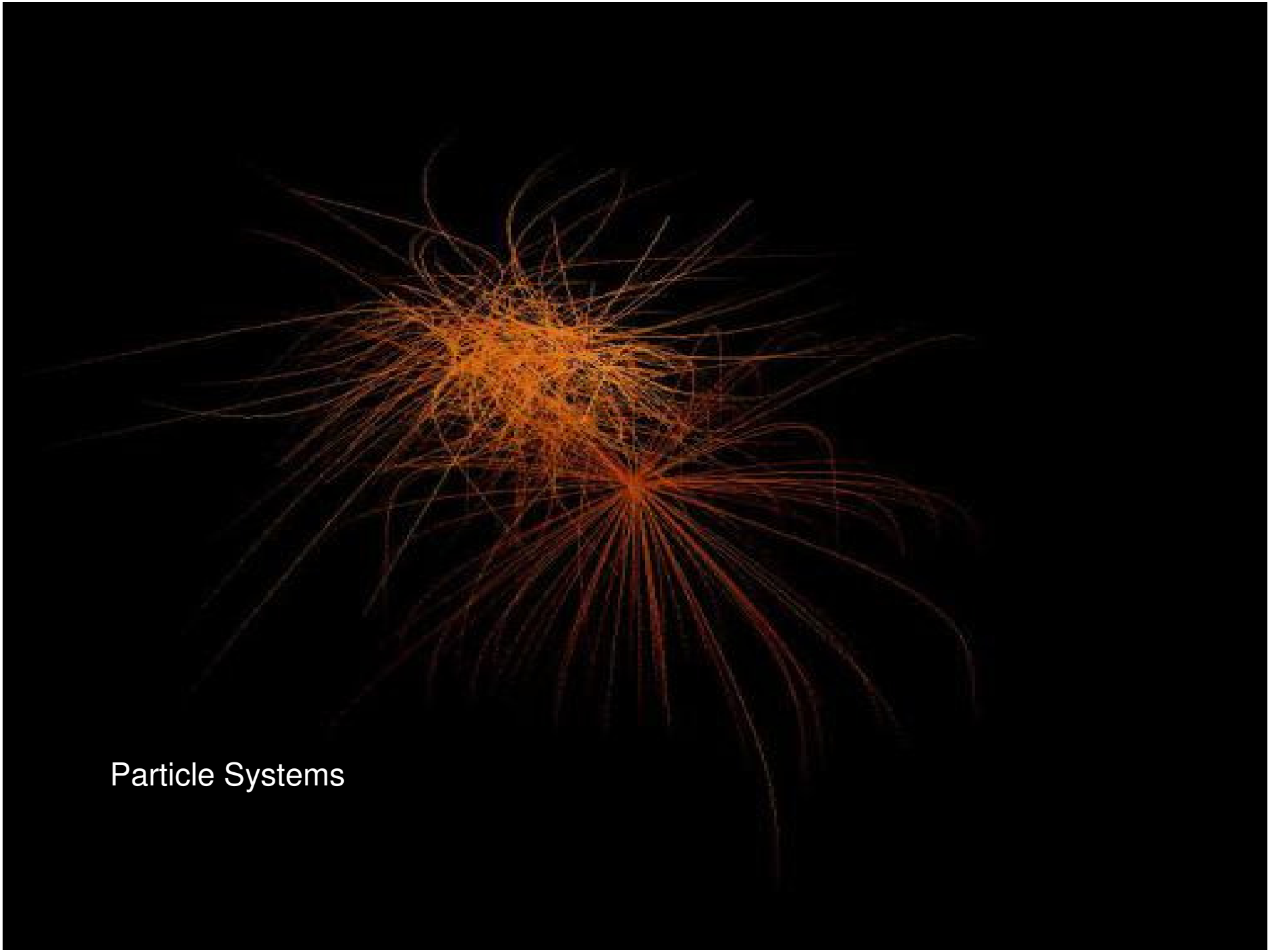
Esse foi usado no exemplo:



Só cor e forma triangular transformada!

Ray tracing e também usado freqüentemente como nestes:




Particle Systems

sistemas de partículas

O sistema de partículas são incrivelmente comum e presentes em todos os modelador. É uma técnica muito útil em computação gráfica.

Desde o início da década de 1980, têm sido usados em jogos, animações, arte digital, e para modelar vários tipos de fenômenos naturais, como fogo, fumaça, cachoeiras, nevoeiro, grama, bolhas, etc.

Exemplos de utilização de sistemas de partículas incluem incêndios, explosões, água em movimento (como uma cachoeira), faíscas, neve, poeira, meteoros, fogos de artifício, estrelas, galáxias, e magias.

Os sistemas de partículas pode ser 2D ou 3D.

Criadas em 1982 por W. Reeves , da Lucasfilm que trabalhava no *Star Trek II:The Wrath of Khan*

"Um sistema de partículas é uma coleção de muitas minúsculas partículas que, juntas, representam um objeto difuso. Ao longo de um período de tempo, as partículas são geradas num sistema, se movimentam e interagem dentro do sistema, e morrem. “

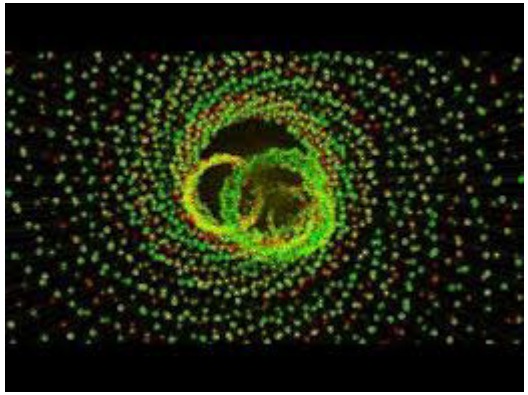
William Reeves, ""*Particle Systems – A Technique for Modeling a Class of Fuzzy Objects* ", ACM Transactions on Graphics 2: 2 (abril 1983), 92.

Importância

Um sistema de partículas embora muitas vezes representado por formas simples (um ponto) permite a modelagem de fenômenos complexos (explosões, sistemas da natureza, formas de vida interagindo: peixes saltando, aves se reunindo, ecossistemas em evolução) e todos os tipos de coisas no plural.

Sistemas de partículas podem ser ou animado ou estático;

Se todo o ciclo de vida de cada partícula é processado simultaneamente, o resultado é partículas estáticas



Particularidades:

1- lidar com quantidades flexíveis de elementos: Às vezes 0 - zero coisas, às vezes 1 - uma coisa, às vezes 10 - dez coisas, e às vezes 10.000 - dez mil coisas.

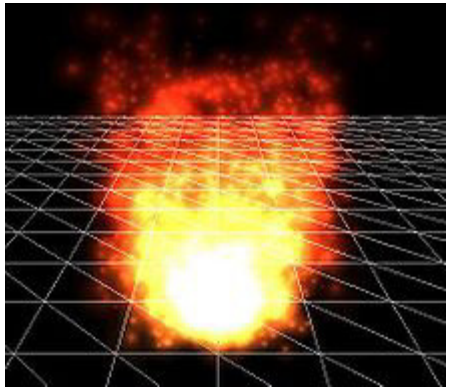
2 - abordagem orientada a objetos sofisticados: Em vez de simplesmente escrever uma classe para descrever uma única partícula, nós também vamos querer escrever uma classe que descreve a coleção de partículas do próprio sistema de partículas.

3 – Descrever a evolução e a iteração do bando e dos elementos

sistemas de partículas

também é um uso dos conceitos (de programação orientada a objetos): herança e polimorfismo.

Uso de *arrays* de objetos de diferentes tipos. Desta forma, um sistema de partículas não necessita de ser apenas um sistema de um único tipo de partícula.



Manter o controle de um sistema de muitos elementos

No que segue iniciamos com conceitos simples mas isso não deve limitar a sua imaginação. Só porque inicialmente iniciamos com fogos de artifício que só andam para a frente, e caem com a gravidade não significa que essas são as únicas características possíveis.

Uma partícula é

um corpo independente que se move na tela.

Tem pelo menos os atributos básicos de:

- localização,
- velocidade e
- aceleração.

Funções básicas de:

- **construtor** (inicializa as variáveis) ;
- exibição - **display( )** , e
- atualização - **update ( )** em localização e propriedades.

A partir deste básico a partícula pode:

Mover em qualquer direção;

Receber forcas se incluindo função como

applyForce ( ) para afetar o comportamento da partícula.

Acrescentar variáveis para descrever a cor e forma, ou receber imagens e ter desenhos complexos.

Ter um inicio , uma duração e uma vida útil: ***emitter*** e ***lifespan***

Emitter = > emissor

é a fonte das partículas: controla as configurações iniciais: a localização, a velocidade, etc.

Um emissor pode emitir uma única rajada de partículas, ou um fluxo contínuo de partículas, ou ambos.

Em uma implementação típica, uma partícula nasce no emissor, mas não viver para sempre.

Se fosse viver para sempre, o **Processamento acabaria por ficar paralisado**: número de partículas aumentaria para um número de difícil controle ao longo do tempo.

À medida que novas partículas nascem, precisamos de formas (idade) para morrer.

Isso cria a ilusão de um fluxo infinito de partículas, e o desempenho do nosso programa não diminui.

Há muitas maneiras para decidir quando uma partícula morre.

Poderia entrar em contato com outro objeto (passar a fazer parte dele),

Pode simplesmente deixar a tela.

Pode-se simplesmente adicionar uma variável de tempo de vida. Por exemplo um timer começando em 255, em contagem regressiva para 0, quando a partícula será considerada "morta"

Array de partículas

Depois de gerenciar o comportamento de uma deve-se controlar um grupo para então ter um sistema de partículas.

Ao invés de ter um número fixo de partículas é interessante que seu array tenha a possibilidade de se:

- expandir ( ),
- contrair (   ),
- Subdividir - splice ( ) ou criar
- subconjuntos ( ),
- e outros métodos para redimensionamento.

Dicas simples:

O motivo para começar em 255 e fazer contagem regressiva a 0 é por conveniência.

Com estes valores, o ***tempo de vida*** para agir como transparência (alfa). Quando a partícula "morre" ela desaparece da tela.

Adição da variável tempo de vida, pode ser feita com um booleano de estado que pode ser consultado (como verdadeiro ou falso) para saber se a partícula está vivo ou morto.

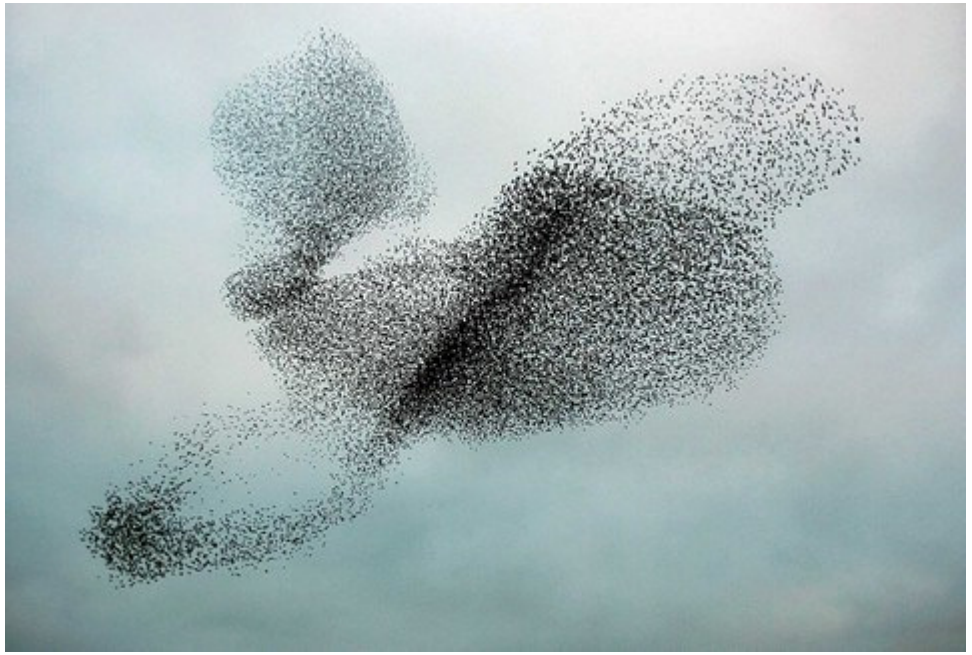
Assim é útil na classe Sistema de Partículas, incluir uma função para verificar e ver se o valor do tempo de vida é menor que 0.

Flocks (bandos)

Introduzido por Reynolds 1987 - http://www.red3d.com/cwr/boids/

C. Reynold (1987). "Flocks, herds and schools: A distributed behavioral model.". *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* ACM: 25–34.

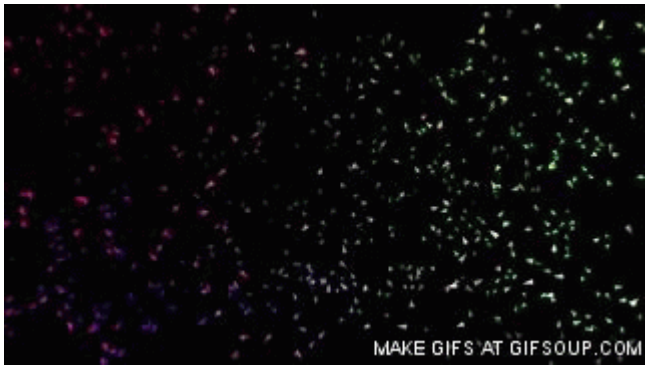
iconic work of 1980s. "Stanley and Stella in Breaking the Ice" (this is the original version). The film was especially novel at the time for using algorithmically-generated flocking and schooling behaviors.



https://www.youtube.com/watch?v=3bTqWsVqyzE

boids

O modelo básico dos bandos é composto por três comportamentos de direção simples que descrevem como um indivíduo faz manobras com base nas posições e velocidades seus colegas próximos:



Separação, coesão e outros

Orientação para evitar aglomeração os membros e para se mover para a posição média (centro de massa) dos membros, **desvio de obstáculos e busca de metas.**

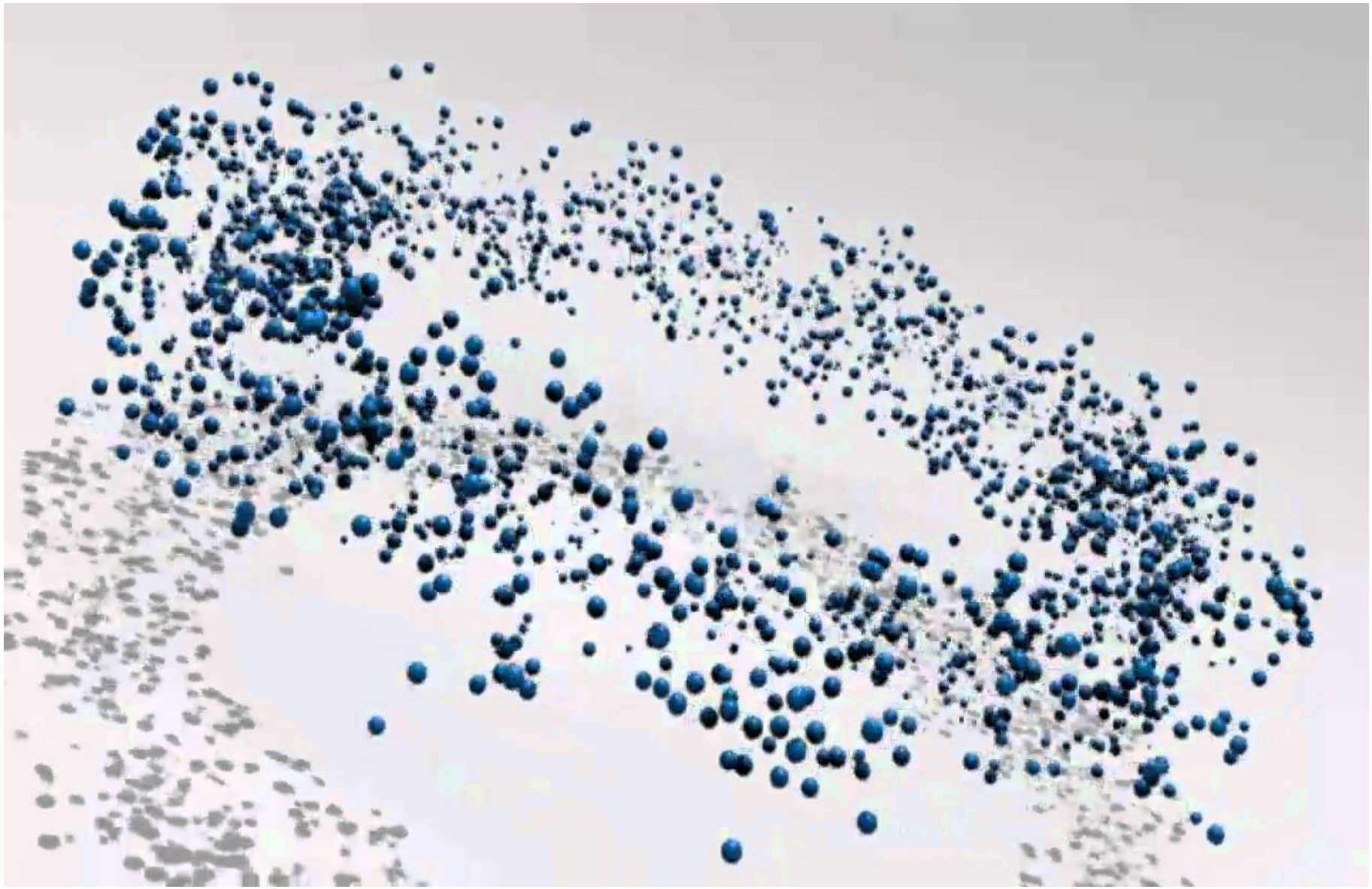
O modelo básico foi estendido de várias formas : como incorporar medo, olfato (através de feromônios modelados como partículas), mudança de liderança.

O movimento de Boids podem ser caracterizados como caóticos (grupos de divisão e comportamento selvagem) ou ordenada. Comportamentos inesperados, como a divisão rebanhos e reunindo depois de evitar obstáculos, pode ser considerado emergente.

A abordagem de Reynold representou um gigantesco passo em comparação com as técnicas tradicionais usadas na animação por computador para filmes.

A primeira animação criada com o modelo foi Stanley e Stella em: Quebrando o Gelo (1987), seguido por uma estréia no cinema no filme Batman Returns de Tim Burton (1992) com gerada por computador enxames de morcegos e exércitos de pingüins marcham pelas ruas de Gotham City.

Boids



Bibliografia:

M. Magdics , Real-time generation of L-system scene models for rendering and interaction, SCCG '09 Proceedings of the 25th Spring Conference on Computer GraphicsPages 67-74 , ACM, New York, 2009

E. Azevedo, A. Conci, Computação Gráfica: teoria e prática, Campus ; - Rio de Janeiro, 2003

http://www.kevs3d.co.uk/dev/lsystems/.

http://natureofcode.com/book/chapter-4-particle-systems/.

http://www.avatar.com.au/courses/Lsystems/