Solid modeling em C.G.

Cap 4 (do livro texto)

UFF - 2014

Metodologias principais

Parameterized primitive instancing
Spatial occupancy enumeration
Cell decomposition
Boundary representation
Constructive solid geometry
Sweeping
Implicit representation
Parametric and feature-based modeling

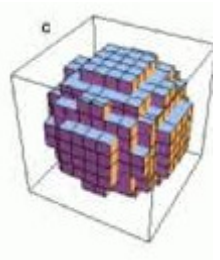
Grupo de métodos que consideram o espaço ocupado pelo sólido

Enumeração do espaço ocupado / Spatial occupancy enumeration

Decomposição em celulas/Cell decomposition

Modelos de Subdivisão : Octree, quadtree, bintree, partição binaria, divisão/fusão

Spatial Occupancy Enumeration



Representação por Enumeração da Ocupação Espacial

Sólido representado por coleções de células ou ”blocos” sem sobreposição (non-overlapping), considerados “colados” juntos para criar a geometria representada;

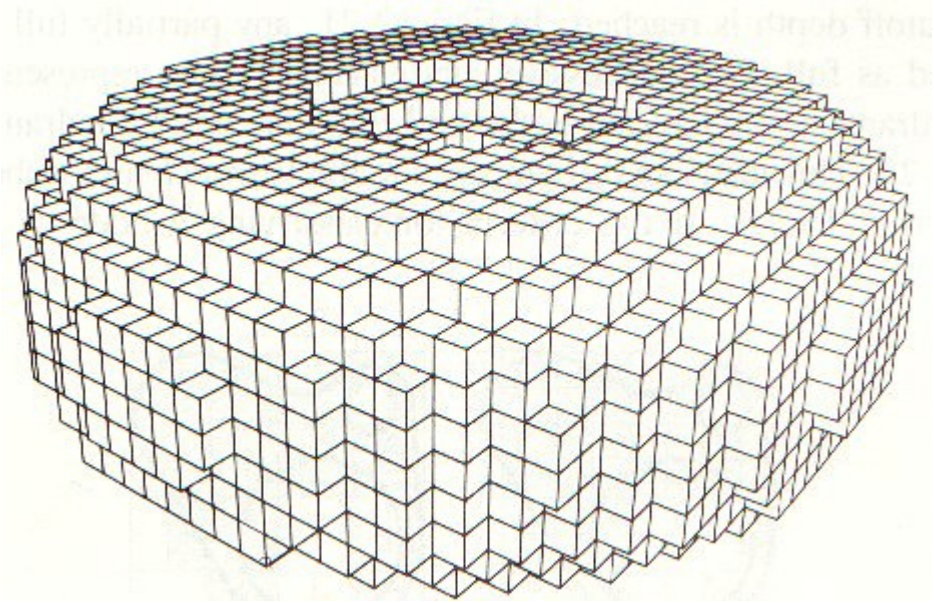
Cada é células ou ”blocos” localizado pelo seu centro no sistema de coordenadas XYZ

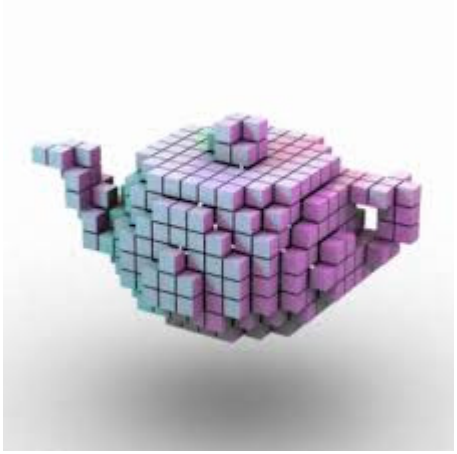
Toda células ou ”blocos” é **idêntico.**

Enumeração da Ocupação Espacial

Um sólido é visto como uma **coleção de partes *mais simples homogêneas*** *de um elemento elementar mínimo ao qual* dá-se o nome de **voxel** .

Um **cubo** é a forma mais usada (mesmo em um toro em no teapot)

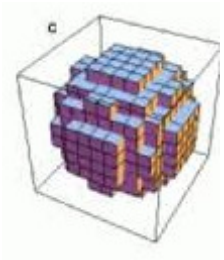




Modelos de Subdivisão (ou Decomposição) do Espaço

Formas de armazenagem e **Representação** decompõem o espaço onde o sólido está em "pedaços“ até descreve-lo precisamente.

Um sólido é visto como uma coleção de **partes *homogêneas até o elemento*** *elementar mínimo* (**voxel**).



Vantagens

É fácil determinar se um dado ponto pertence ou não ao sólido basta verificar se o ponto pertence a algum dos *voxels*

É fácil determinar se dois objetos se interceptam (se tocam ou colidem)

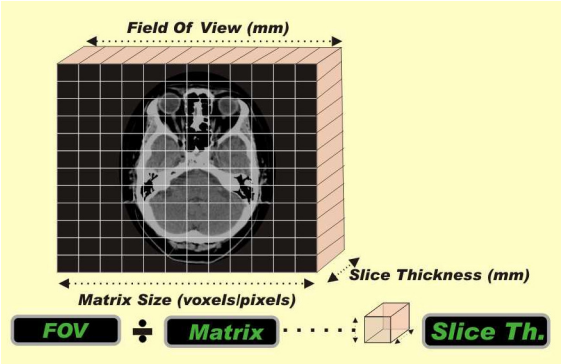
Facilita a realização de operações de união, intersecção e diferença entre sólidos.

Avaliação de: comprimentos, áreas e volumes

Principalmente usada em modelos biomédicos

CT

MRI



Spatial Occupancy Enumeration

Enumeração da Ocupação Espacial

Desvantagem

Desvantagem

Um objeto muito detalhado e complexo necessita de muito tempo e espaço

Desenvolvimento posteriores foram motivados em economizar armazenamento agregando dados que têm valores idênticos ou semelhantes.

Variações da ideia básica:

Considerados casos particulares da Enumeração de Ocupação Espacial **, deste que são considero elementos "colados":**

Octree, quadtree, bintree, partição binaria, divisão/fusão

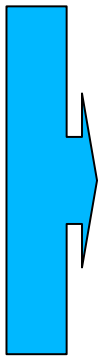
Descrevendo a classe hierárquicos da estrutura de dados do objeto.

Há diversas técnicas de ***trees*** (estruturas de dados hierárquicos usados) para representar dados de espaço:

Binary space partition

Quadtree

Octree



armazena-se essas estruturas em forma de árvore.

Binary space partitioning (BSP)

Método para a subdivisão recursiva um espaço convexo (ou não) por conjuntos de planos.

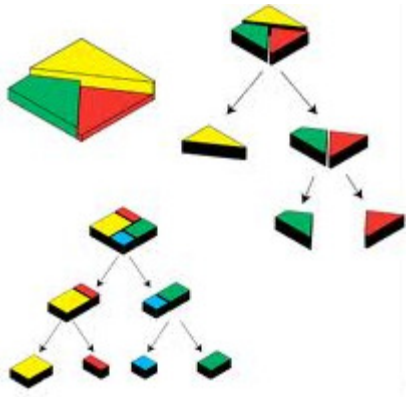
Esta subdivisão dá origem a representação de objetos associada a estrutura de dados em **árvore** conhecida como uma BSPT.

É muito aplicado em games (***first person shooters e cenas interiores***) : por permitir que as informações espaciais sobre os objetos da cena (e sua visibilidade) sejam identificadas em relação a um observador muito rapidamente mesmo em cenas muito complexas.

Outras aplicações incluem a realização de operações geométricas em geometria sólida construtiva, a detecção de colisão em robótica e jogos de vídeo 3-D, o ***ray-tracing*** e outros aplicativos envolvem a manipulação de **cenas espaciais complexos**.

Os planos são escolhido para coincidir com **os planos definidos por polígonos na cena. (Devem ser armazenados)**

BSP trees pode ser generalizado para qualquer dimensão



Usado nas *engines* mais populares de games:

Doom, Quake e posteriores.

BSPT é um método através do qual o espaço **n** dimensional é repartido por **n-1** entidades dimensionais chamados hiperplanos.

Por exemplo:

Um linha ou curva **1D** seria dividida por **pontos (0D)** ,

o plano **2D** seria dividido por entidades **1D**, ou seja, **linhas** e

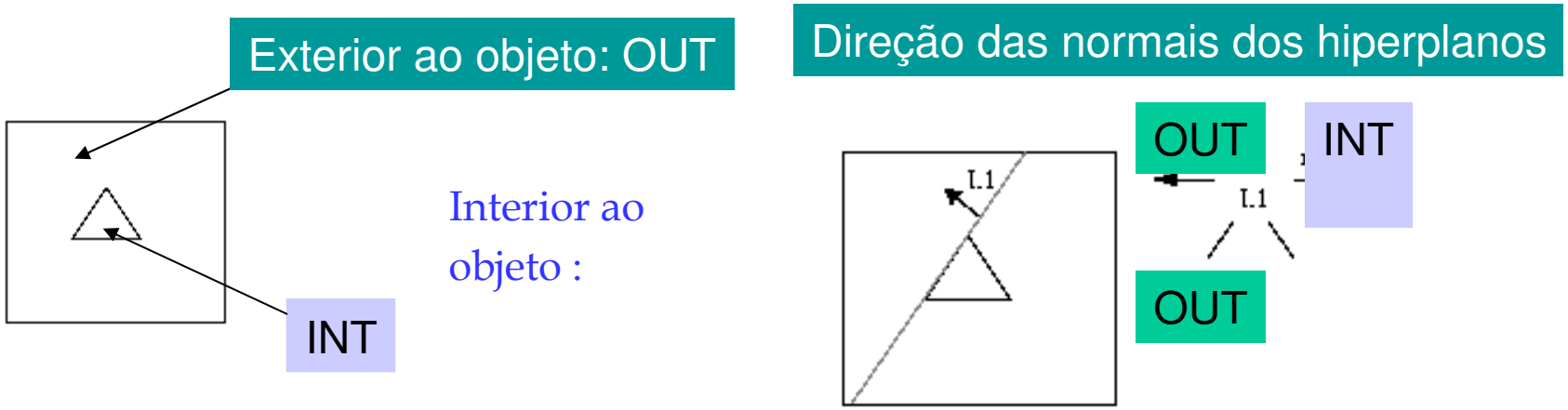
O espaço **3D** seria dividido por entidades **2D,** ou seja, **planos.**

Uma vez que o espaço, é repartido por um **hiperplano**, ele é representado por dois espaços **n dimensionais**, um de cada lado do hiperplano de particionamento.

Exemplo

Uma forma eficaz de compreender a utilização de um BSPT é a seguir um exemplo simples da sua construção, o que pode ser feito através de um exemplo **2D**.

Suponha que se deseja gerar a representação BSPT de um **triângulo** (veja a figura abaixo), EXTERIOR (OUT) ou o **universo** considerado será o **quadrado que o envolve**. Começamos com uma **BSPT vazio**.



O processo começa pela escolha de um **hiperplano** para particionar o espaço: L1.

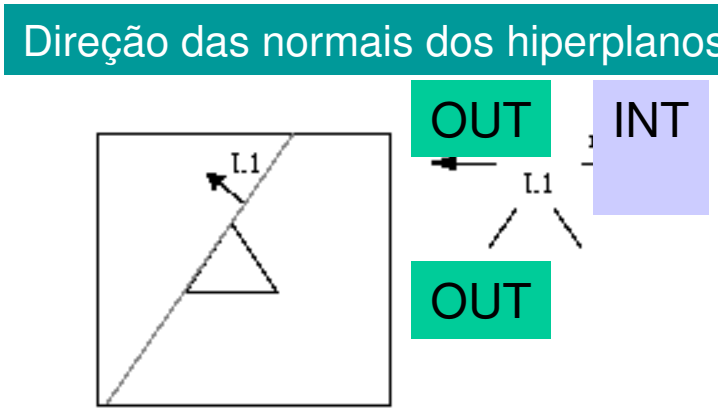
L1 particiona o espaço em 2 meios-espaços, o espaço em frente de L1 e no espaço por trás dele.

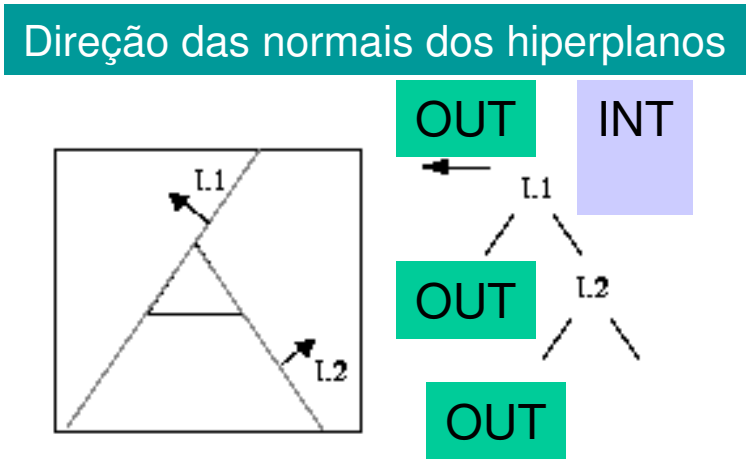
Veja que a normal que ficará associada com L1 é a que é exterior ao objeto. Os 2 meios-espaços são então repartidos de forma recursiva até que a restrição de homogeneidade for cumprida. Note que é necessário apenas voltar a particionar um semi-espaço, ele ser não é homogêneo.

Neste caso, o espaço atrás L1 não é homogêneo, por isso, iremos parti-lo.

Exemplo (cont)

Deve-se continuar este processo, até que o modelo seja completamente homogêneo. Ao classificar as regiões da árvore por completo, toda a região é descrita por uma folha na árvore, ao passo que cada nó interno descreve tanto uma aresta do modelo quanto um hiperplano.





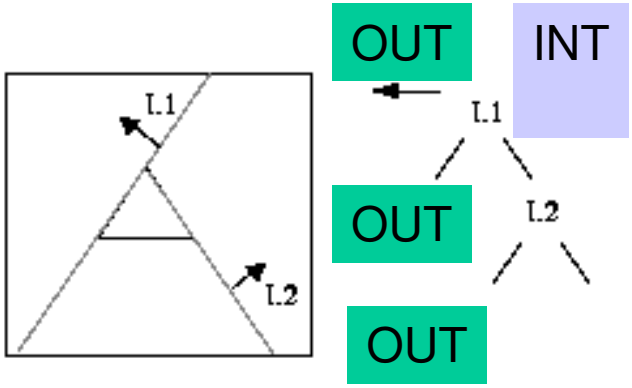
O processo continua pela escolha de um **hiperplano** para particionar o espaço restante.

Suponha que se escolheu L2 para a nova divisão deve-se considerar como sua **normal a que está apontando para fora no objeto na região que o hiperplano o corta. Esta será descrita pela direção OUT arvore** .

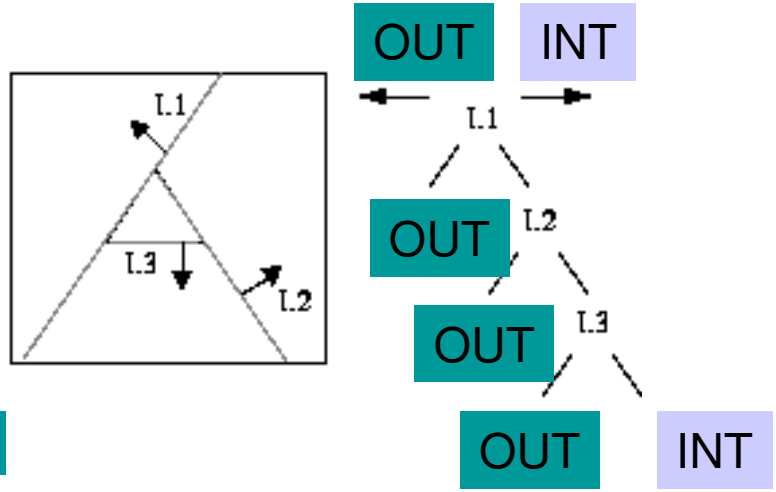
Assim com isso L2 particiona o espaço restante em 2 meios-espaços, o espaço em frente de L2 (direção da normal) e no espaço por trás dele. Esse espaço atrás ainda não é homogêneos (não tem só objeto no seu interior – INT ou só o mundo exterior - OUT).

Exemplo - cont

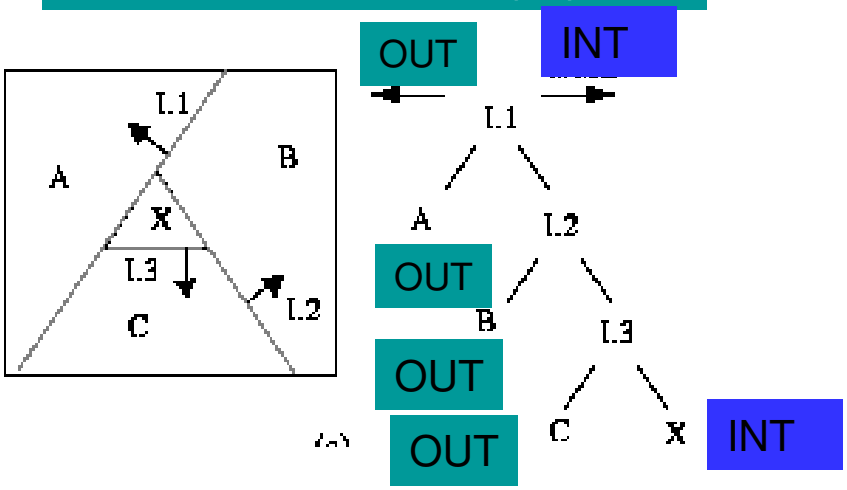
Deve-se continuar este processo, até que todas as folha do modelo seja completamente homogêneo. Assim usamos mais L3



Direção das normais dos hiperplanos



Direção das normais dos hiperplanos



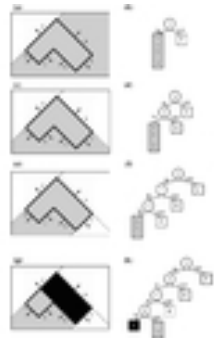
Usado para polígonos convexos

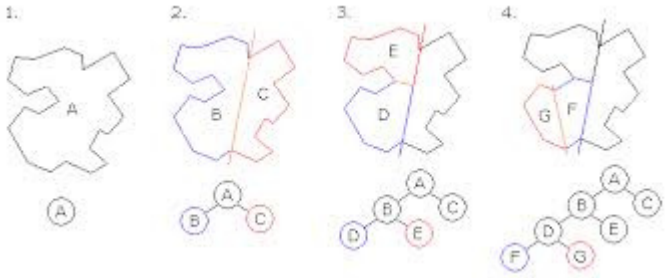
A direção da normal deve ser a mesma para toda a arvore.

Deve-se considerar em cada passo apenas o conjunto restante de poliedros até se ter uma área homogênea

A união dos subespaço com label OUT formarão o universo exterior ao objeto.

A união dos subespaço com label INT formarão o interior do objeto.





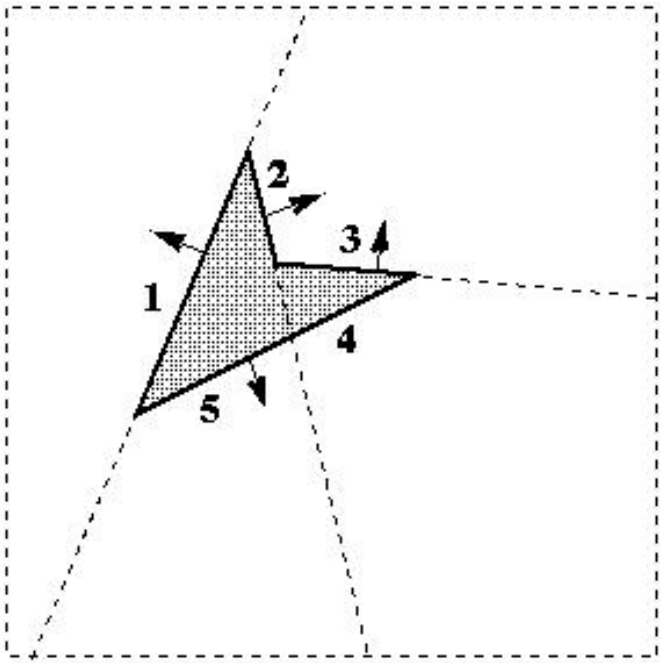
E não convexo

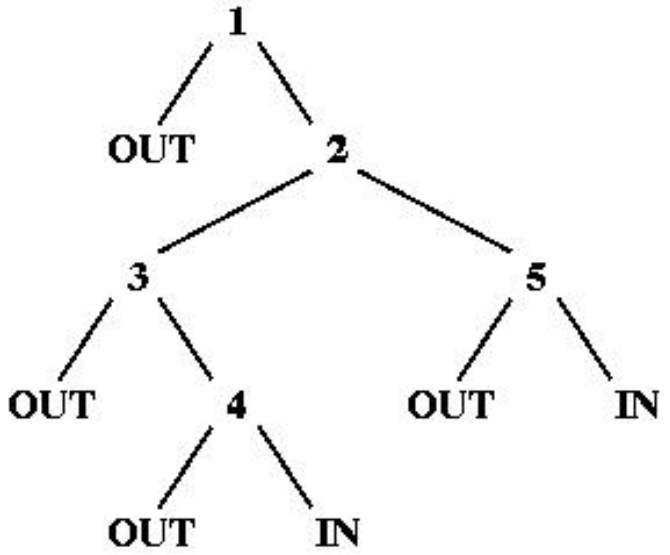
E não convexo

Considerar em cada passo apenas o conjunto restante de poliedros até só se estar no interior da figura (IN) ou exterior (OUT)

Cada hiperplano pode ser analisado mais de uma vês (como o 4 e 5 abaixo) e diversas notações podem ser usadas para isso.

A definição do interior e exterior deve usar operações mais complexas que só uniões.

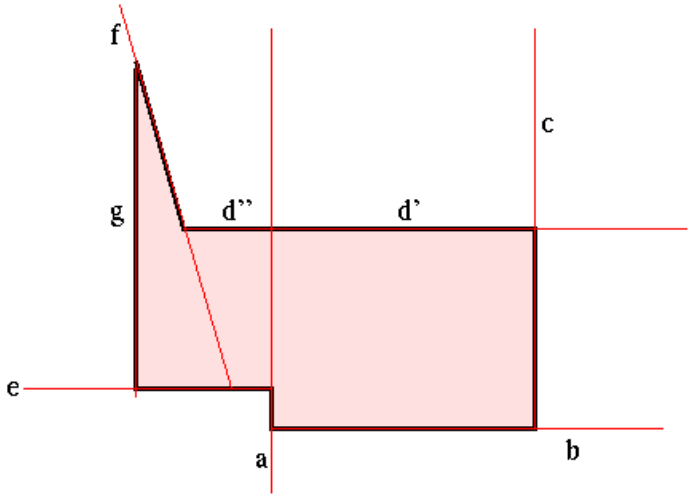


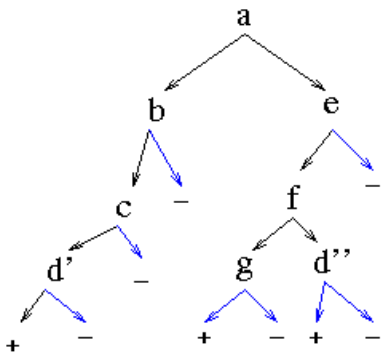


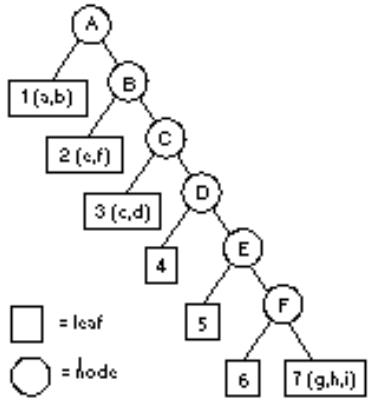
Mesmo hiperplano aqui recebe outro label

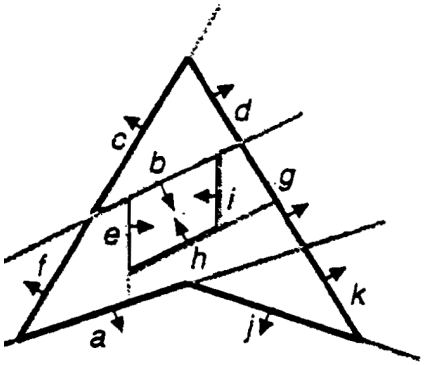
Outros exemplo

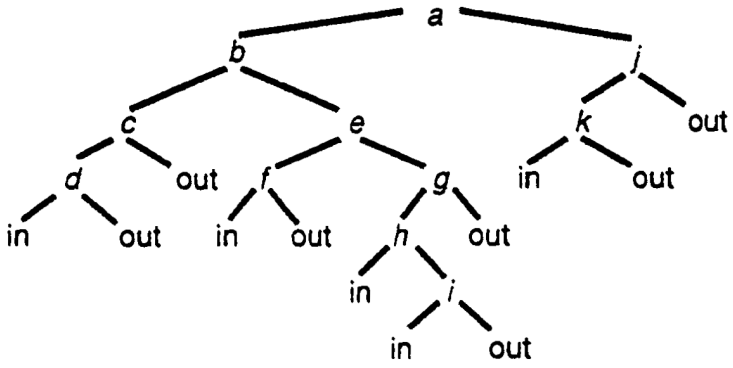
Mesmo hiperplano aqui recebe i´ ou i´´, etc..

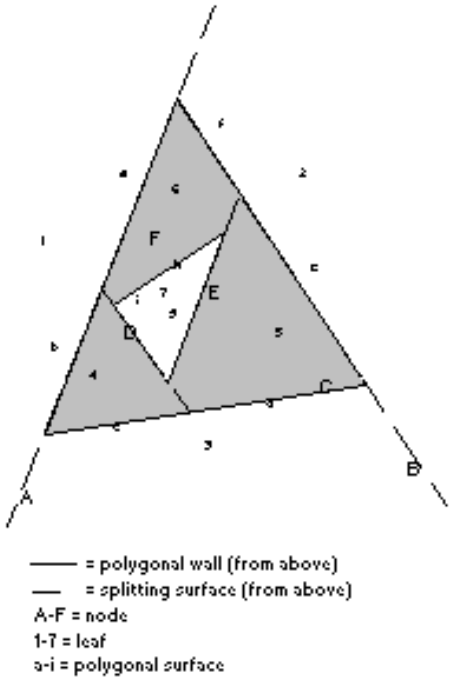








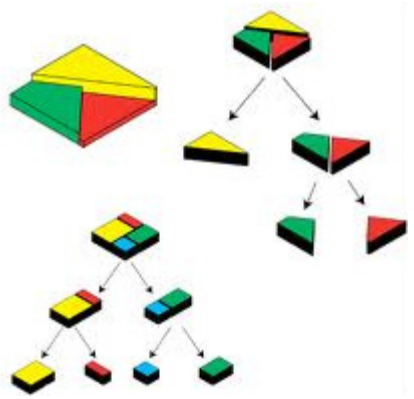




Mesmo hiperplano aqui recebe outro label

Mesmo hiperplano aqui recebe letras entre parêntesis (a,b), (c,d) , etc..

BSPT tem vantagens únicas

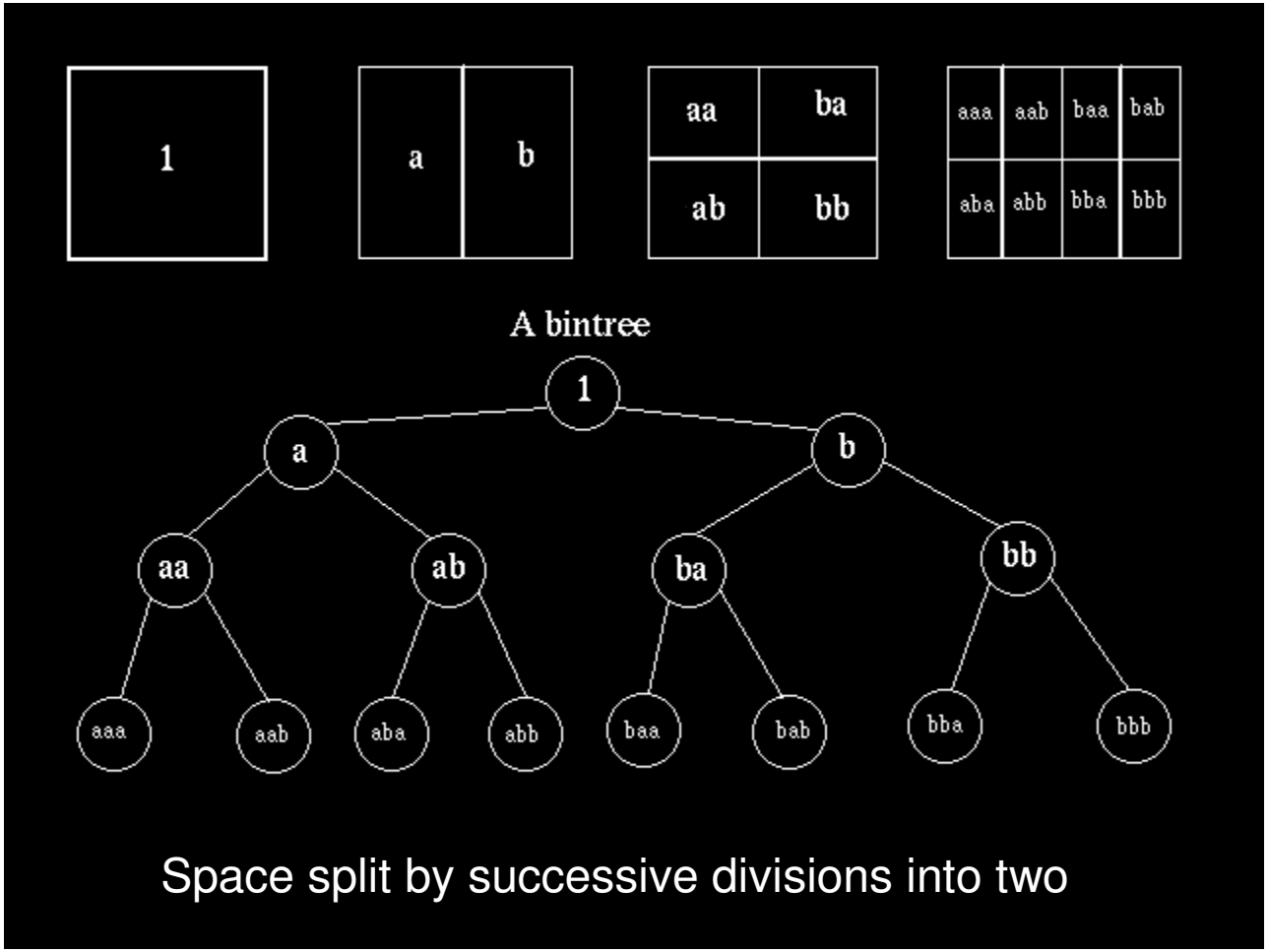


. Quando usado em visualização tem representação intermediária para os diferentes pontos de vista.

- Sua estrutura hierárquica permite algoritmos muito simples e eficientes para ser desenvolvido.
- É compacta e é numericamente robusta.
- A representação BSPT é útil em modelagem 3D, em realismo visual e no processamento de imagem.
- Há algoritmos muito rápido para mesclar modelos representados como BSPTs.
- Após a segmentação de cada região e construção da BSPT , podemos usar operações de **booleanas** para obter a area correta do interior e exterior desses modelos.
- Esta intersecção serão aquelas partes do modelo que serão visíveis de determinado ponto de vista em games.

Bintrees – árvore binária

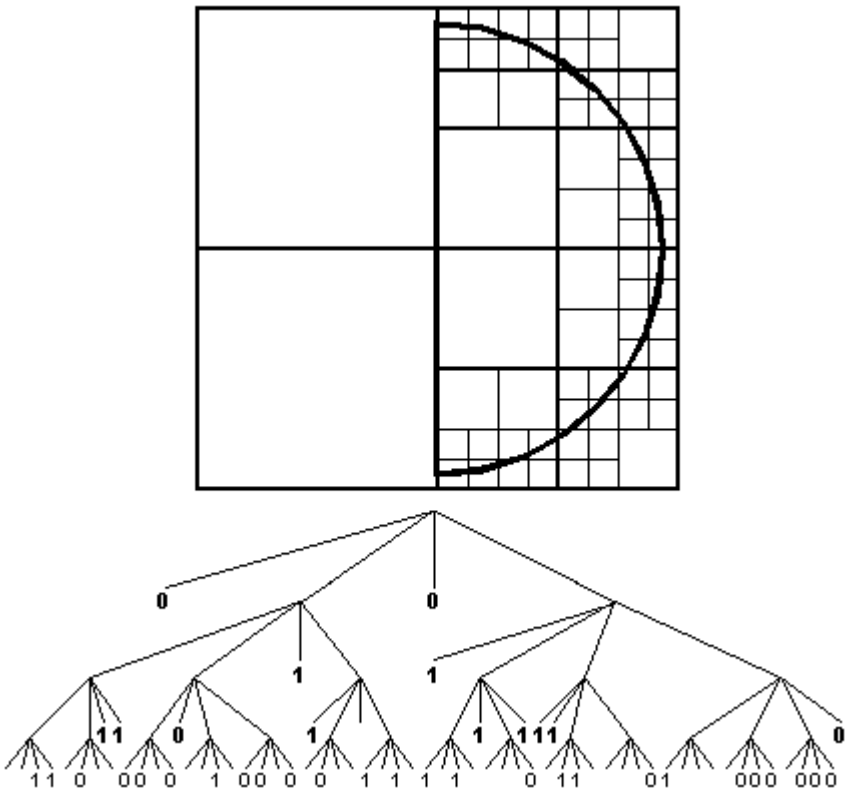
- Caso simplificado da anterior
- Com direções fixas dos planos, e seu particionamento sempre em metade.
- Também chamado de HV-tree em 2D



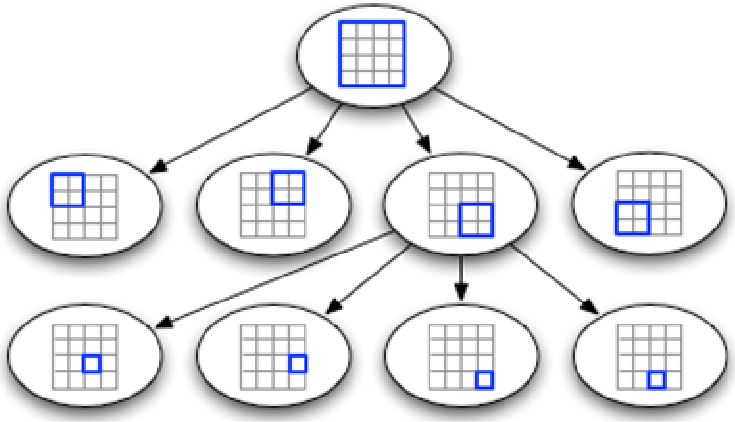
Quadtree - Octree

Q**uadtree**, regiões bi-dimensionais .

Quando tridimensionais: **octree**.



Quadtree - 4-tree



Na árvore de representação:

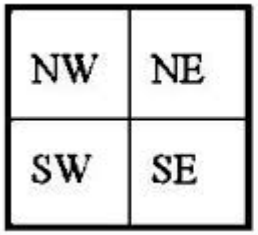
a raiz corresponde à ordem inteira.

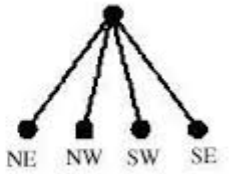
Cada filho de um nó representa um quadrante

NW (nordeste),

NE (noroeste),

SW (sudoeste),





SE (sudeste) da região no nível representada por aquele nodo.

Os nodos de folha da árvore correspondem a blocos para os quais nenhuma subdivisão adicional é necessária.

Quadtree - 4tree

O objeto é envolvido por um quadrado e em seguida é dividido em quadrados menores quadrantes**)**.

Cada um destes é então classificado em:

Cheio, caso o objeto ocupe todo o quadrante ;

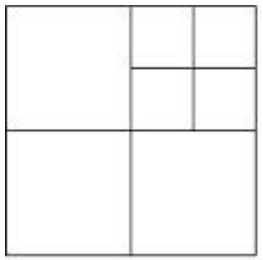
Vazio, caso o objeto não ocupe nenhuma parte do quadrante ;

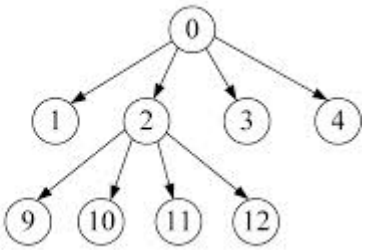
Cheio-Vazio, caso o objeto ocupe parte do quadrante ;

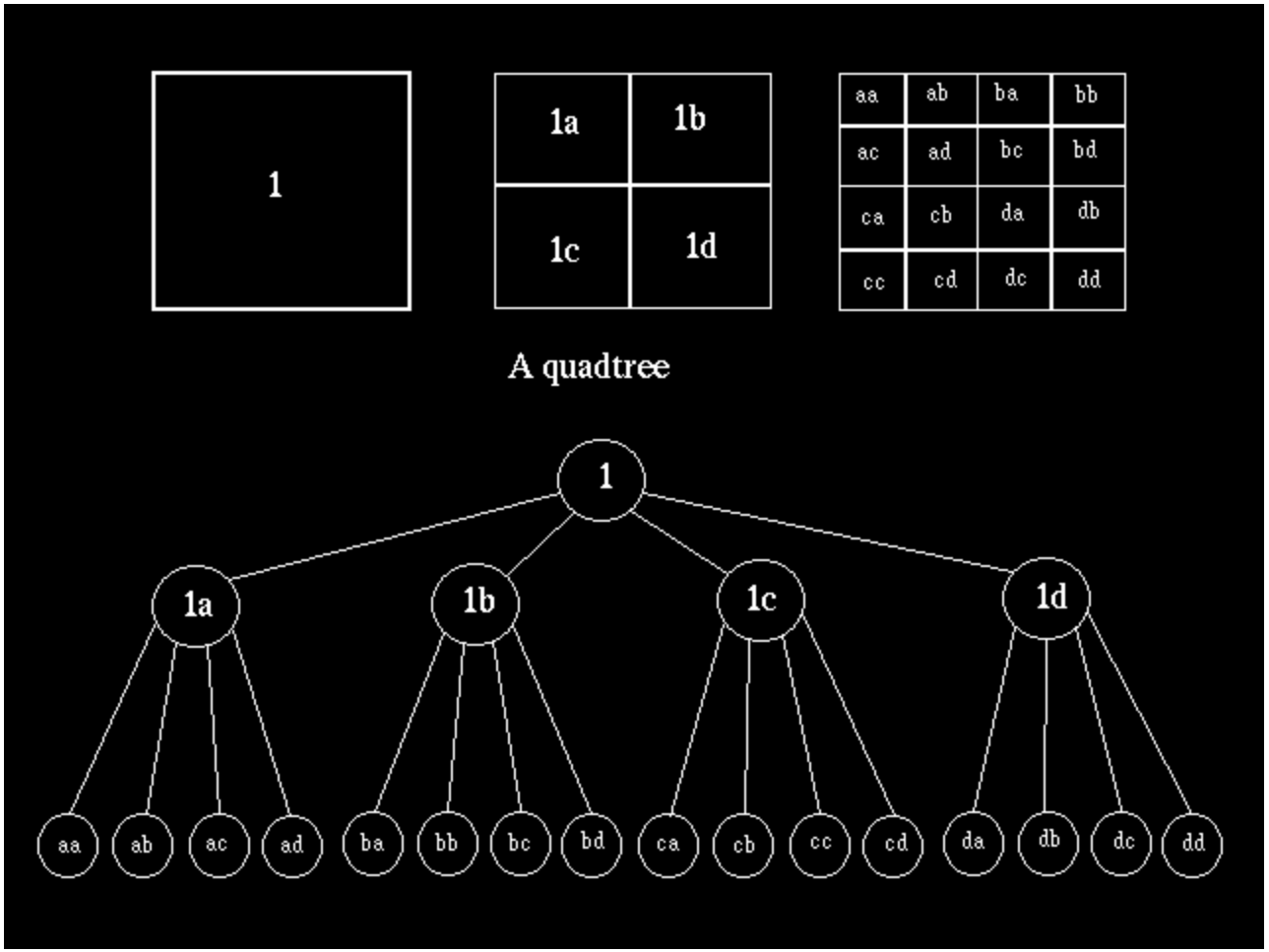
Um quadrante classificado em **"Cheio-Vazio"** é novamente dividido em 4 partes iguais e o processo de classificação é repetido para as novas partes.

O algoritmo repete-se até que só hajam quadrante com as das classes: **Cheio** e **Vazio**

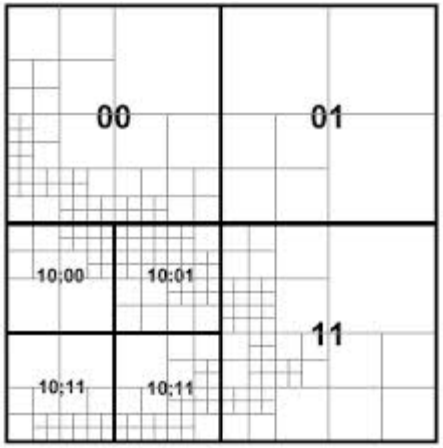
Depois usa-se alguma forma de indexação e codificação da estrutura final

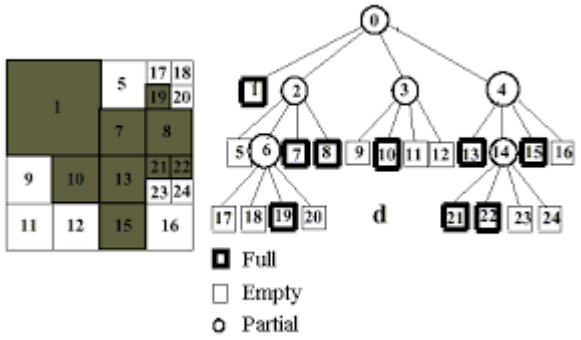






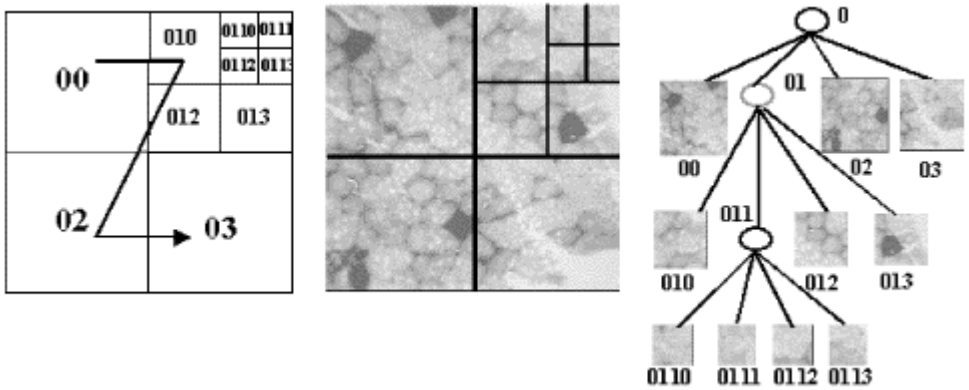
Codificação da estrutura final





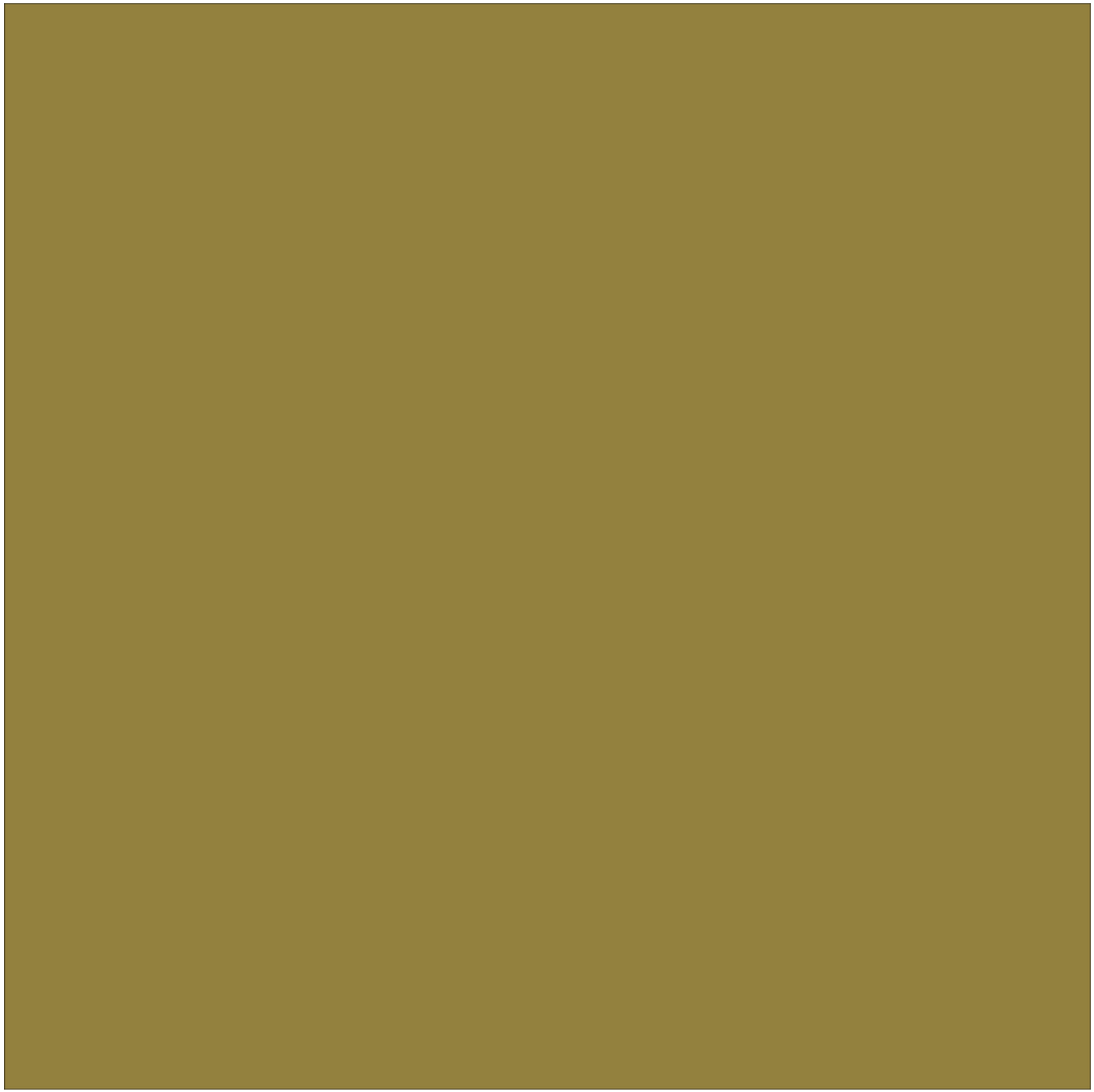
Principal uso

Descrição de partes de uma imagem ou display



OBS:

Apenas armazenada a estrutura de dados final



Octree – 8tree

O processo de subdivisão é representado por uma árvore (8 filhos) no qual o no **raiz** representa o objeto inteiro e os nos folha correspondem a boxes de qual nenhuma subdivisão adicional é necessária.

Uma octree é uma estrutura de árvore de dados baseada em uma célula com oito "crianças".

Cada célula de uma octree representa um cubo em espaço físico.

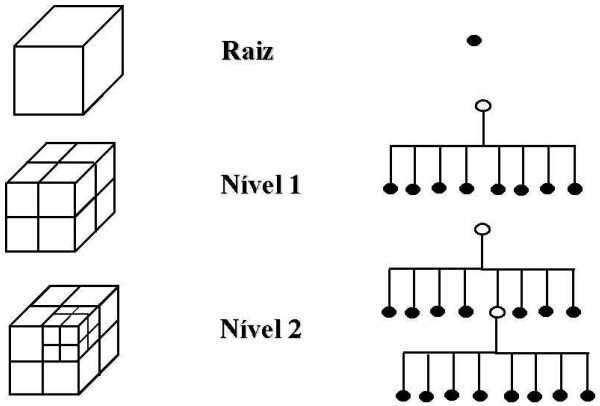
Cada criança representa uma "octant" de seu pai.

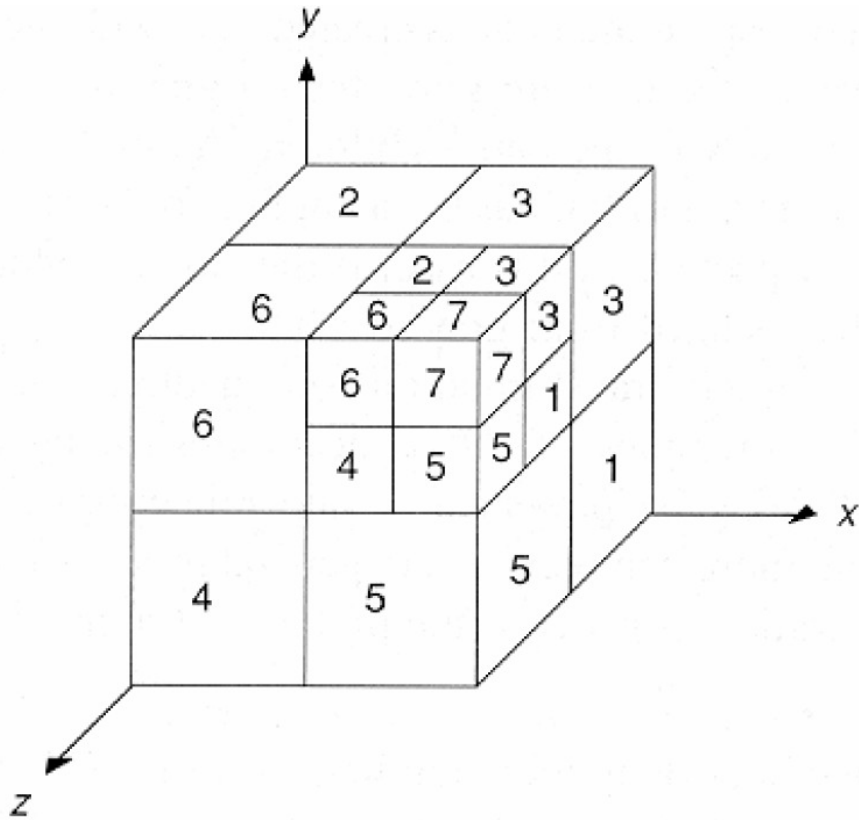
Nas folhas FINAIS da árvore estão Voxcels.

Exemplo de enumeração da 8Tree

Pode ocorrer denominações mais mnemônicas como:

Left-L, Right – R, Up- U, down – D, Front- F, Back -B

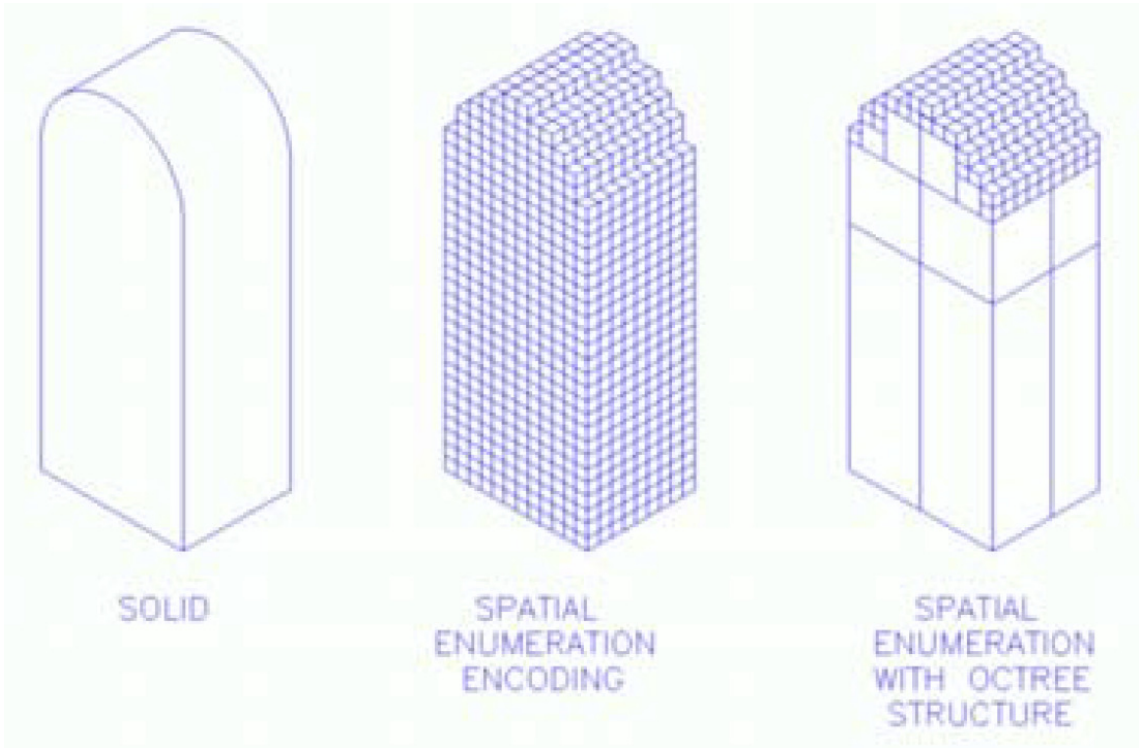




Comparando:

um pequeno comprimento de 1000 pixels, acaba ocupando

1 bilhão de voxels : 1000x1000x1000= 1.000.000.000 !



"representação por OCTREE"
(ou árvore com 8 filhos)

O objeto é envolvido por um box e em seguida é dividido em 8 box menores **OCTANTES)**.

Cada um destes é então classificado em:

Cheio, caso o objeto ocupe todo o box;

Vazio, caso o objeto não ocupe nenhuma parte do box;

Cheio-Vazio, caso o objeto ocupe parte do box;

Um octante classificado em **"Cheio-Vazio"** é novamente dividido em 8 partes iguais e o processo de classificação é refeito para as novas partes.

O algoritmo repete-se até que só hajam box com as das classes: **Cheio** e **Vazio**

Polygonal Map Octree – PM octree

Representação hibrida que combina octree ou quadtree com B-rep para obter um contorno mais preciso.

No exemplo abaixo os nós podem ser (além de cheios e vazios) com vértice, aresta ou face

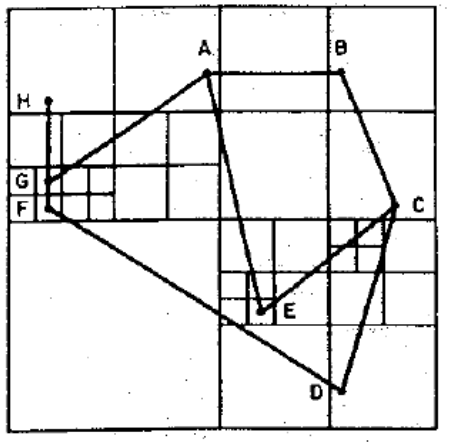


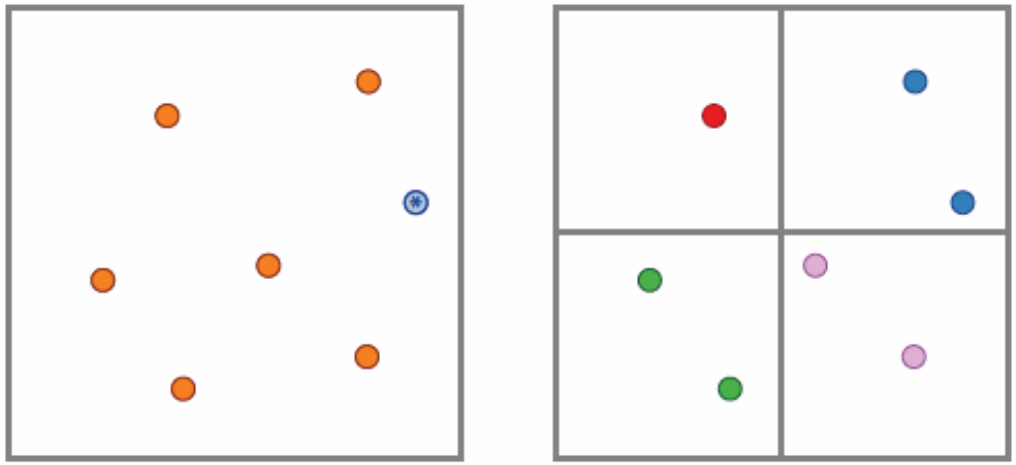
Figure 8. A PM quadtree.

Point Region Octree – PR Octree

É uma Octree ou quadtree onde a subdivisão acaba quando o numero de partes da região alcança um numero mínimo pré definido.

É principalmente usada em GIS, definição de mobiles, sistemas de partículas, identificação de elementos por area em games, etc.

No exemplo abaixo cada região precisa ter pelo menos 1 ponto.



Subdivisão e Fusão (spliit & merge)

Divide-se o espaço ou o plano onde está o objeto em partes iguais (usando 2Tree, 4Tree ou 8Tree como nas técnicas anteriores, até e classifica-se cada parte de forma homogênea como objeto ou universo.

Mas une as partes homogêneas vizinhas em qualquer posição que estejam.

Elementos pertencentes a figura (pixels) em duas dimensões são **pretos.**

Branco => universo

Exemplo de como ficaria com uma representação inicial do objeto por Quadtree

(a)Forma do objeto original

(b) Representação binária da área ocupada:

0 = Universo

1=Objeto

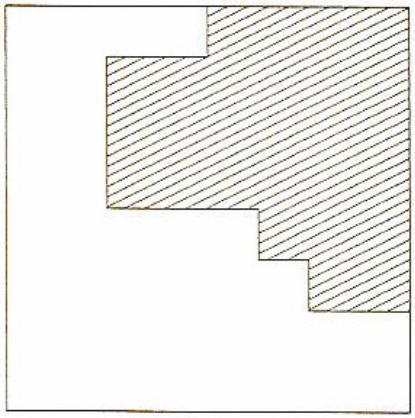
(c) Depois de divisão em 4tree e fusão

(d) Estrutura da

Em arvore : nós

Não homogêneos=●

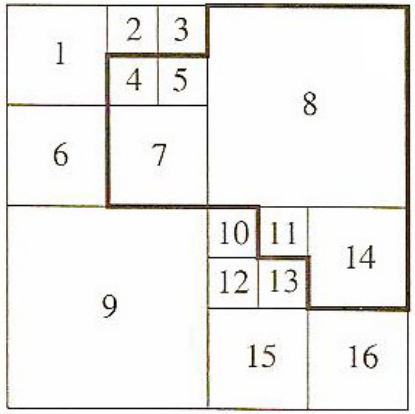
Homogeneos = ■



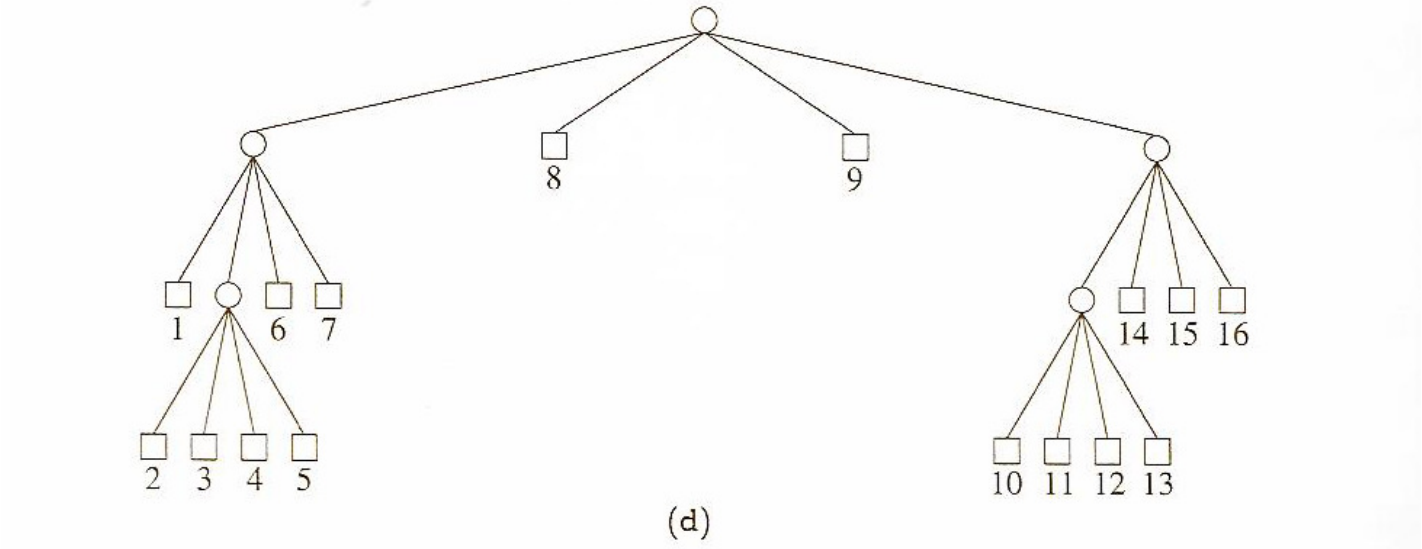
(a)

0	0	0	0	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

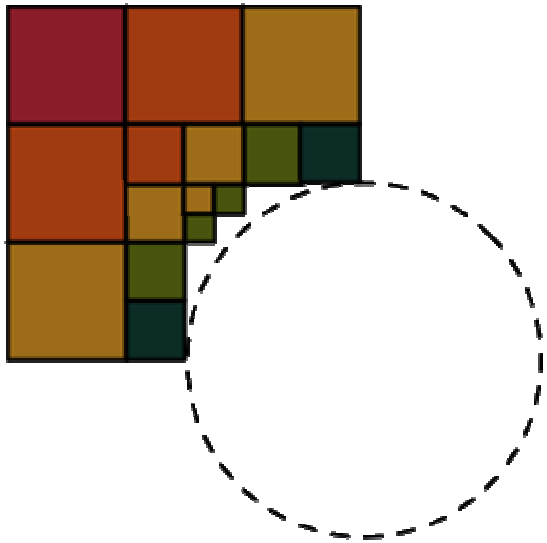
(b)

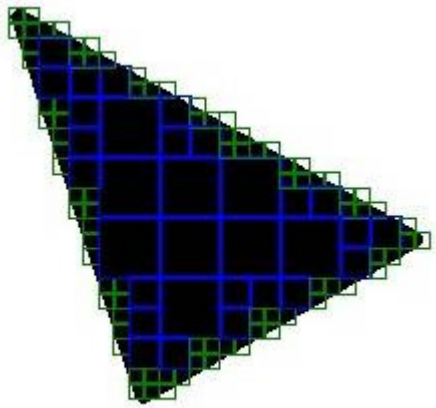


(c)



(d)



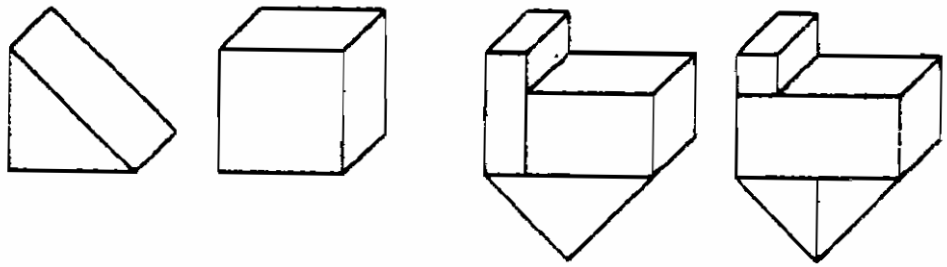


Variações da idéia básica:

Decomposição em células ou partes diferentes:

Cell Decomposition.

Forma genérica de decomposição onde as células podem ser de tipos diversos e parametrizadas, incluindo elementos não planos combinados em uma operação simples de colagem.



Primitivas Parametrizáveis.

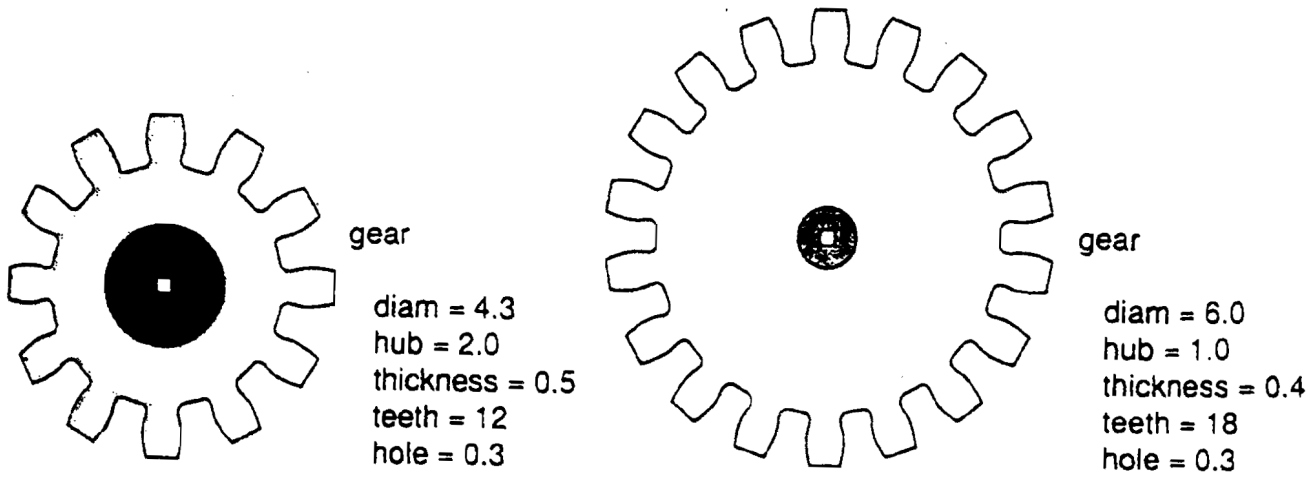
Forma simples de criar objetos transformando objetos de um biblioteca.

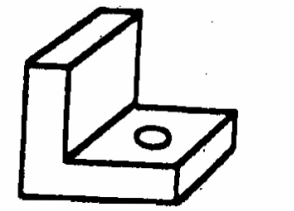
Cubos = > Paralelepípedos =>Placas =>Laminas

Elipsóides => Esferas =>Círculos=> Pontos

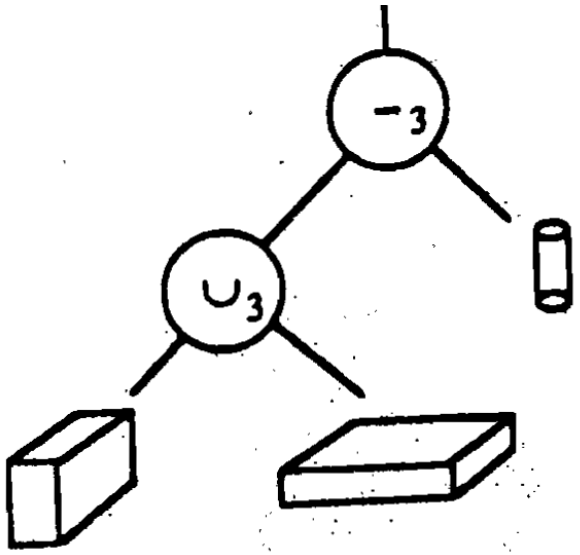
Instanciamento de primitivas

Por exemplo: Uma fabrica de engrenagens (gears) pode ter um programa especifico para gerar todas as possibilidades que produz.





SOLID

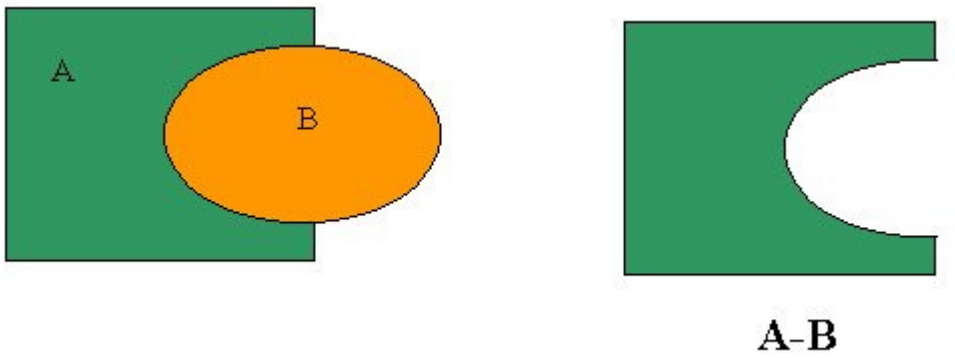


Constructive Solid Geometry
CSG

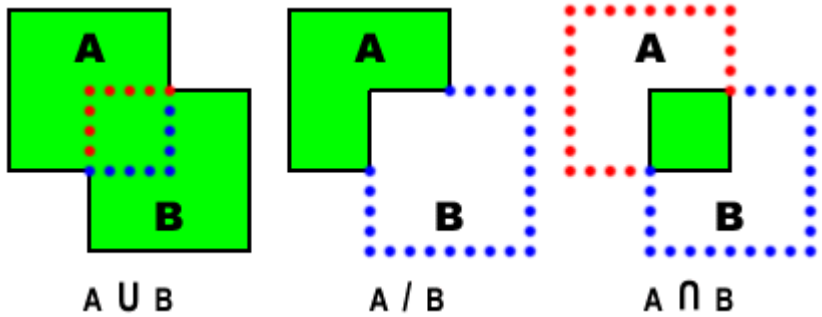
Geometria Sólida Construtiva (***CSG-Contrutive Solid Geometry***)

Usa Instanciamento de Primitivas e *operação com esses sólidos primitivos).*

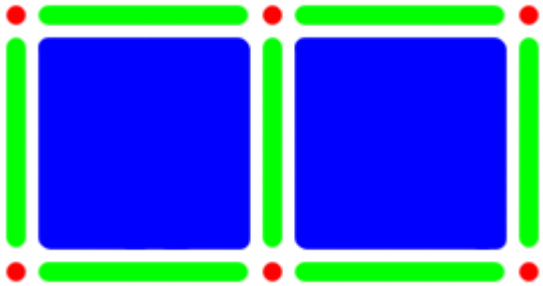
Esta técnica de modelagem cria novos objetos através de transformações, reposicionamento e ***operações básicas nas primitivas***

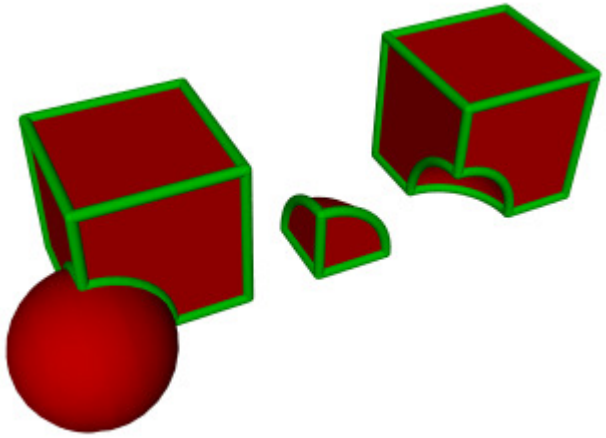


Operações Booleanas Regularizadas=
geram sempre sólidos nas subtrações / e interseçõesꓵ



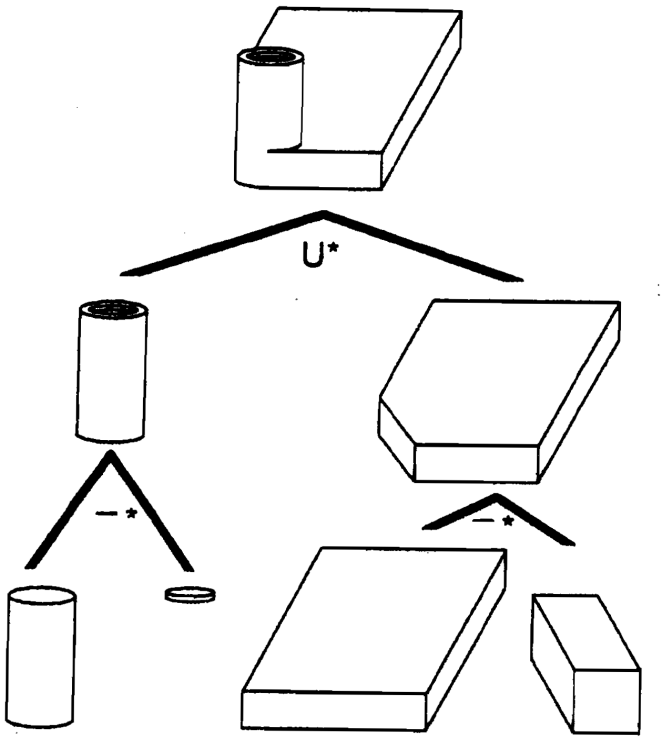
Quando dois elementos tem um limite (aresta, vértice ou face) em comum nas operações de subtrações / e interseções ∩ o limite ficará com o sólido restante de modo a deixa-lo sempre com todos os contorno fechados.



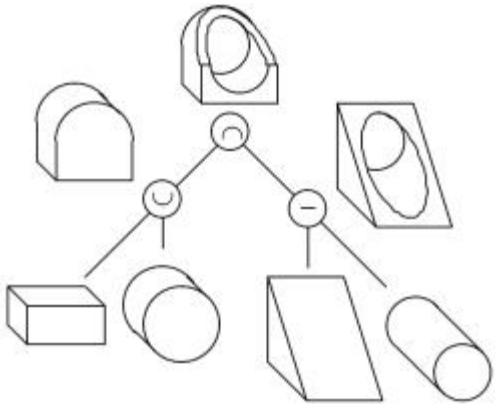


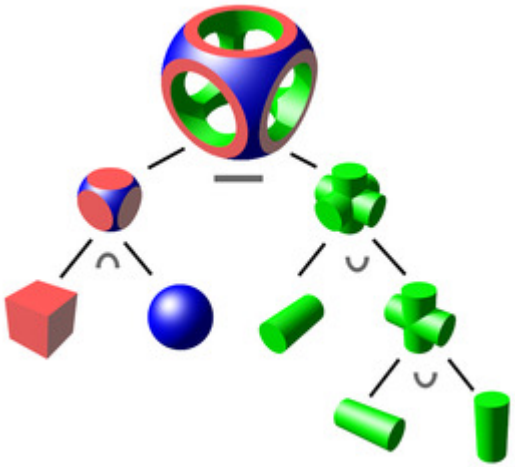
A modelagem através do uso de CSG possui muitas aplicações como:
Simulação por elementos finitos, e
é claro jogos (presente na engine do Unreal e no Editor do Quake.
Uma otima ideia é usar a biblioteca : OpenCSG que usa OpenGL e C++.
http://www.opencsg.org/

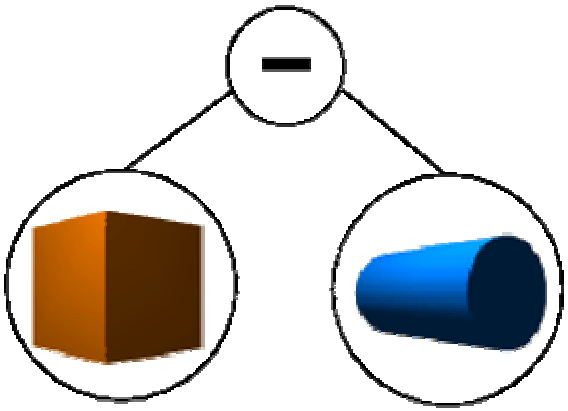
Example of the CSG tree



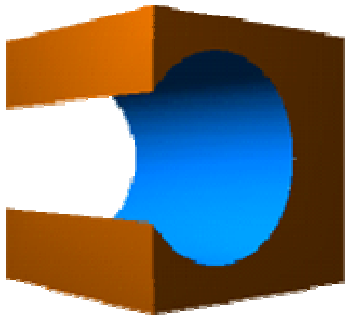
As operações da CSG para gerar um solido não são comutativas e nem unicas





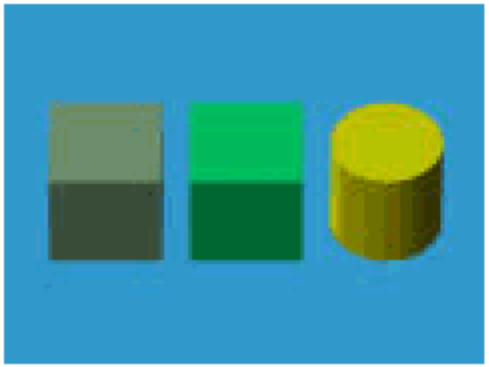


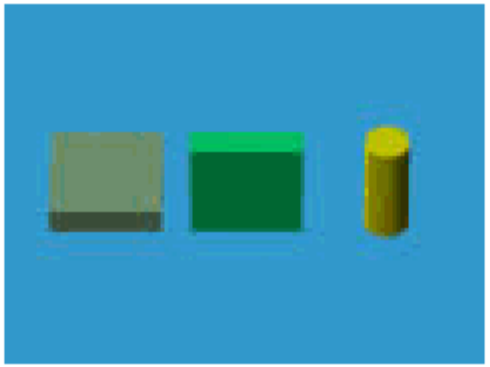
CSG Tree

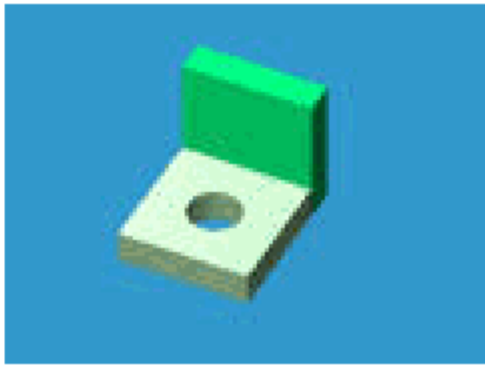


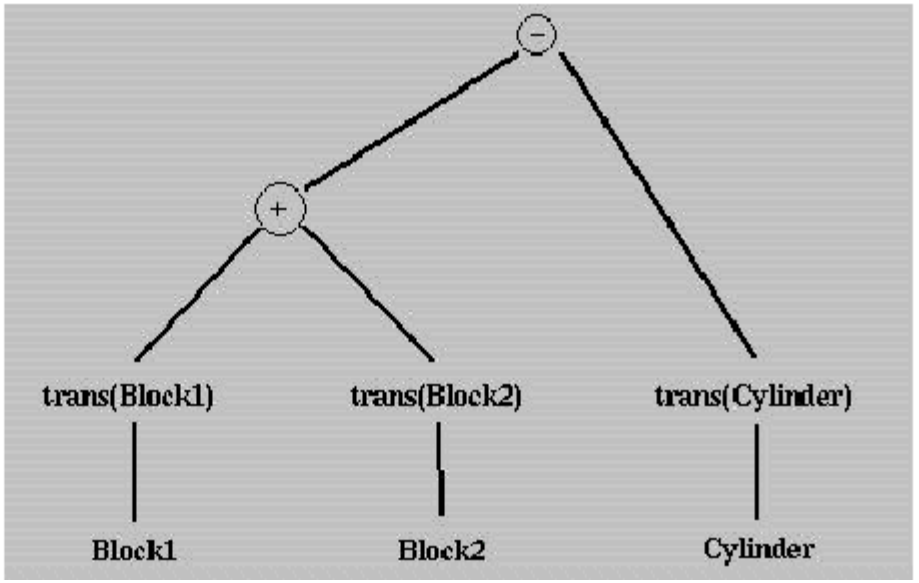
The Resulting Solid

Mais exemplos









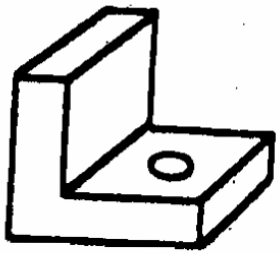
Representação dos limites do sólido

Boundary Representation – Brep

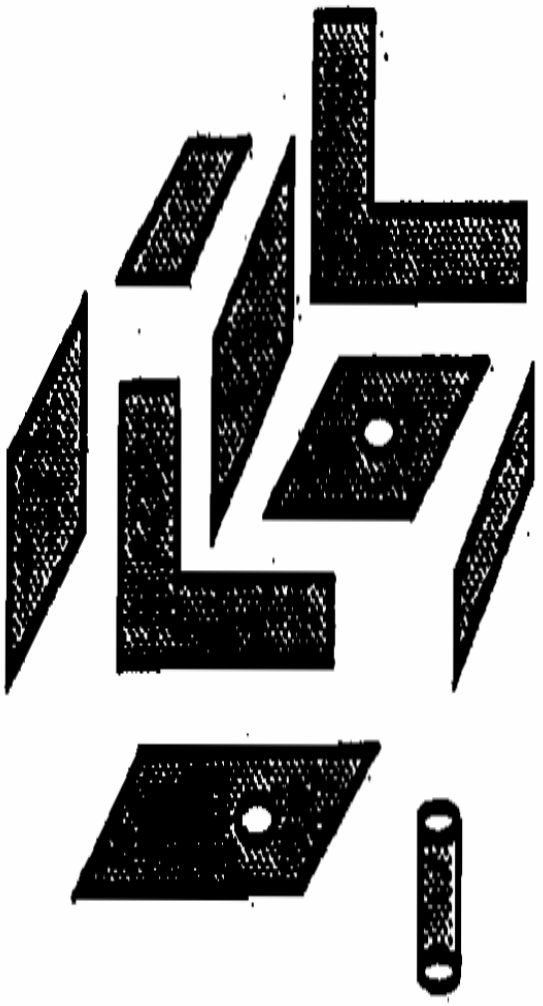
É a forma mais usada

Muito vezes confundida com a modelagem de superfícies, mas agora toda a topologia é considerada para garantir que o objeto seja realizável e continue após as operações que serão realizadas nele.

Agora a topologia deve ser validada não só a geometria gerada (Equação de Euler)



SOLID



Boundary representation

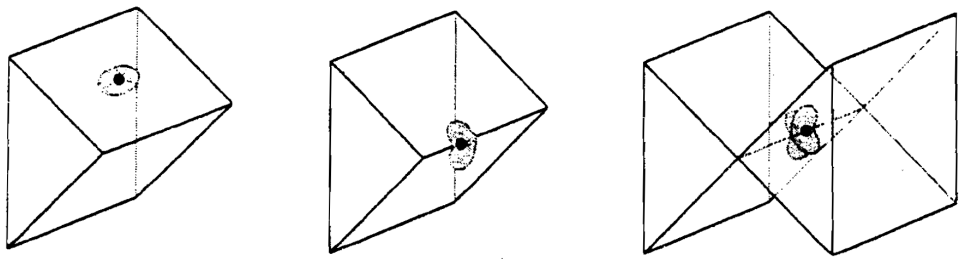
B-rep

Sólidos cujos limites seja topologicamanete equivalentes a um disco

Limitações da maioria dos modeladores baseado em

Limites (que é o que ocorrem em modelos físicos reais e não abstrações matemáticas)

2-manifold.

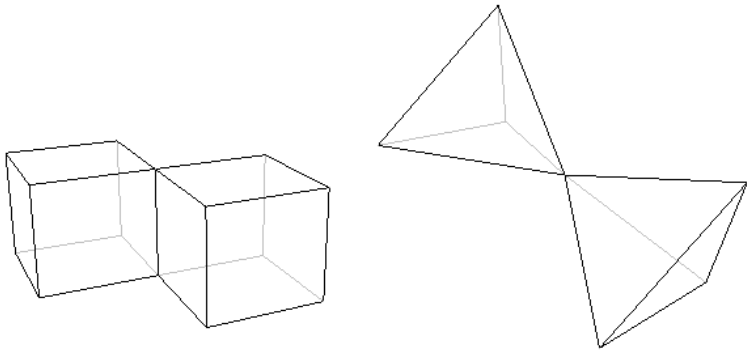


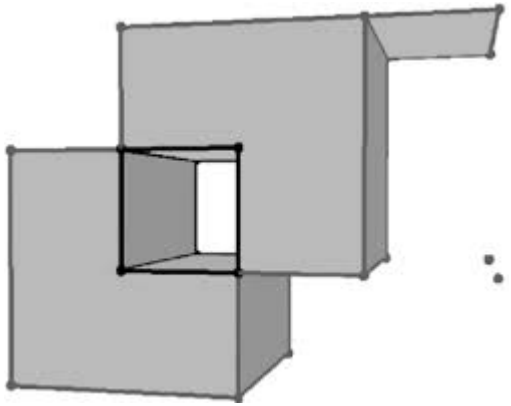
Sólido com superfícies non manifold

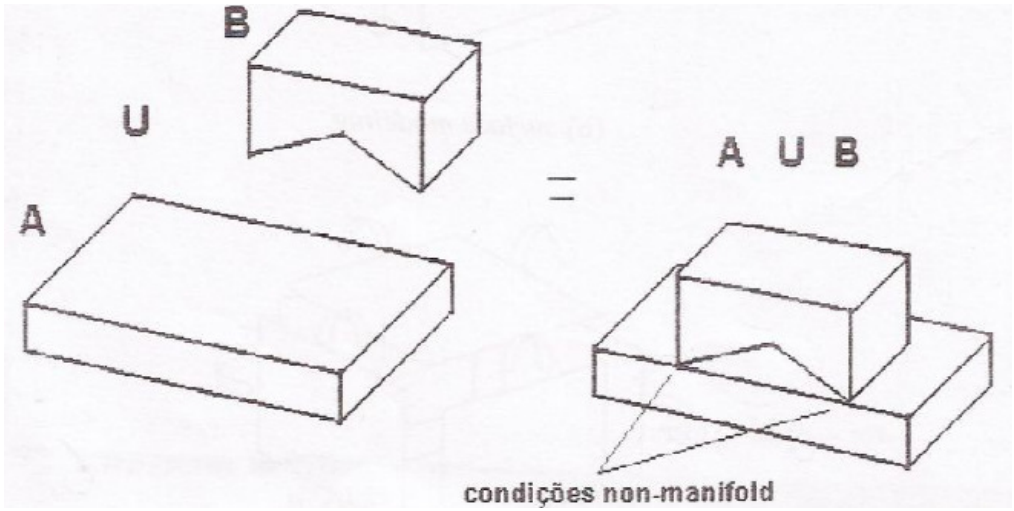
Superfícies non manifold x 2 manifold

O conceito de *manifold* permite caracterizar de forma rigorosa uma importante classe de objetos geométricos de grande interesse no contexto da modelagem geométrica. Uma superfície *manifold* ou *2-manifold* é um espaço topológico onde cada ponto possui uma vizinhança aberta equivalente a um disco bidimensional. Isto quer dizer que, se analisada localmente numa área pequena o suficiente no entorno de um ponto dado, uma superfície existente num espaço tridimensional pode ser considerada "chata" ou plana. Pode-se dizer que deformando a superfície localmente para um plano, ela não rasga ou passa a possuir pontos coincidentes. Num poliedro *manifold*, cada aresta pertence a exatamente a duas faces. Esta propriedade é fundamental para a utilização de algumas estruturas de dados topológicas, como a *Winged-Edge* e a *Half-Edge*, que serão estudadas nas seções seguintes.

um objeto *non-manifold* obtido como resultado da operação booleana de união de dois objetos *manifold*.





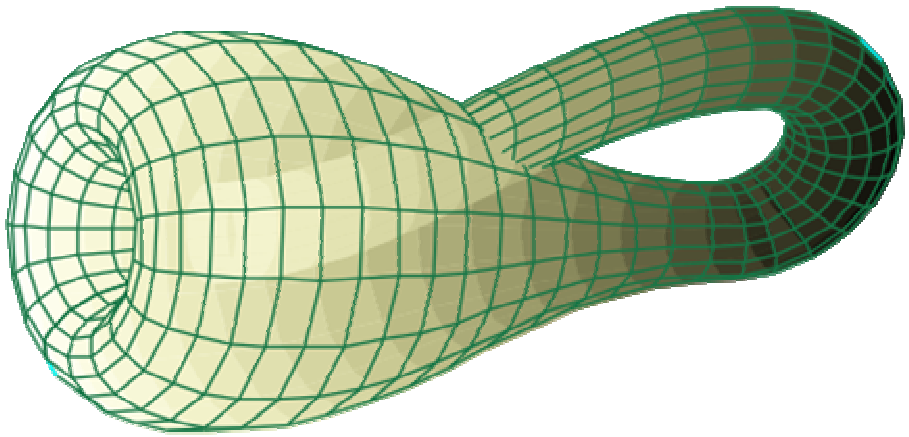


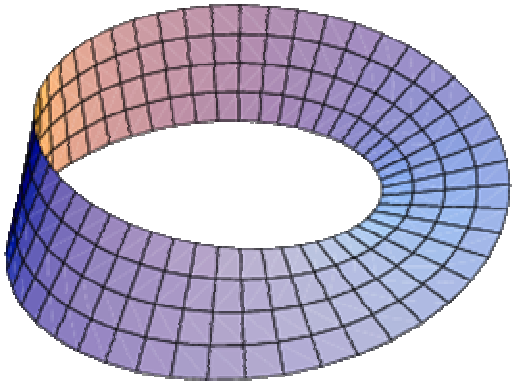
Modelos realizáveis e não realizáveis

A **garrafa de Klein** é um exemplo de uma as estruturas não orientáveis como a **faixa de Möbius**.

São exemplos de sólidos não realizável.

Operadores de Euler = grupo pequeno de **operações** (e suas **inversas**) que permitam a modificação dos modelos B-Rep, seguras o bastante **para não produzirem modelos não orientáveis** ou **não realizáveis**.

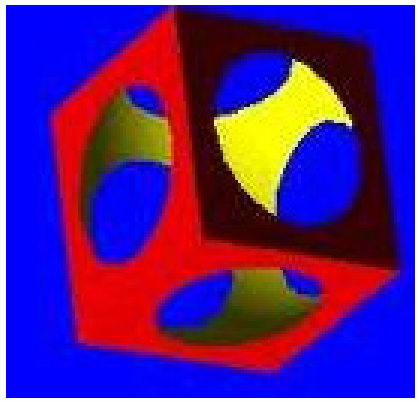


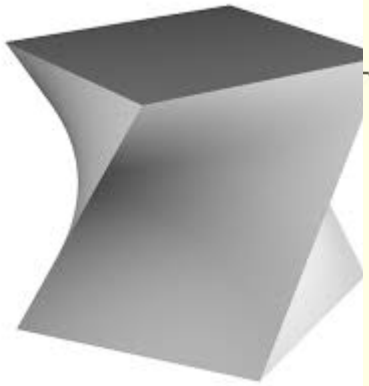


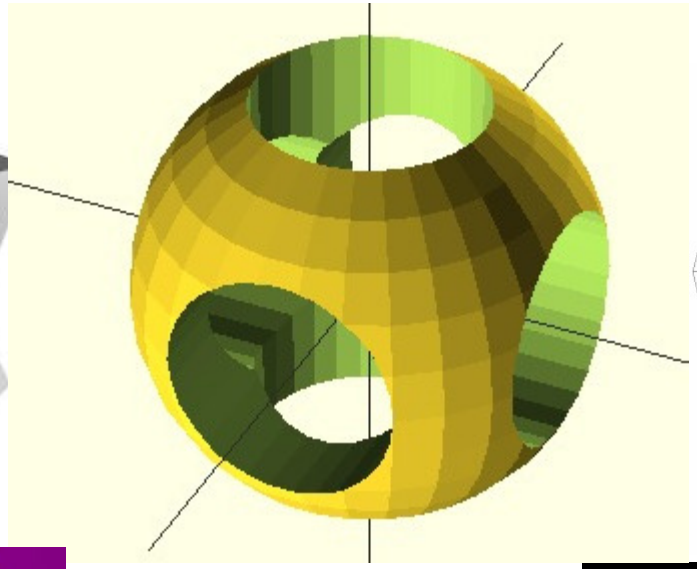
Topologicamente equivalentes:

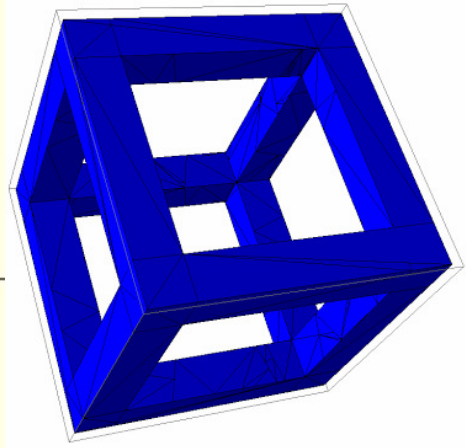
Rubber sheet deformation

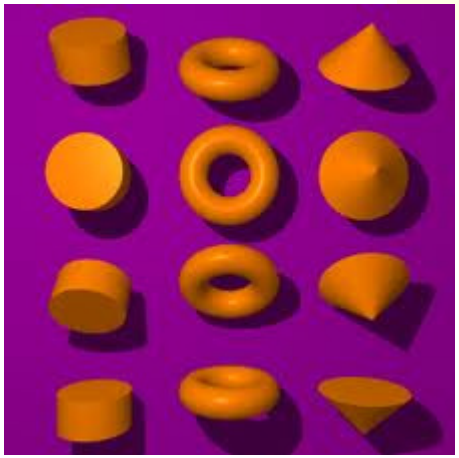
Massinha de modelar = "Play-Doh"

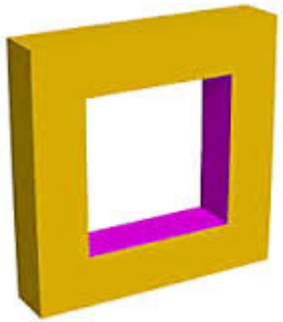








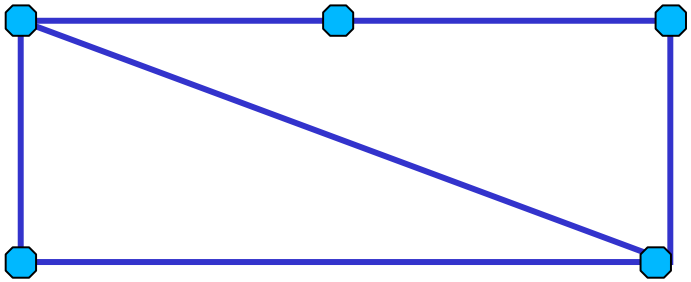




Exemplo de
Operações Topologicas Locais

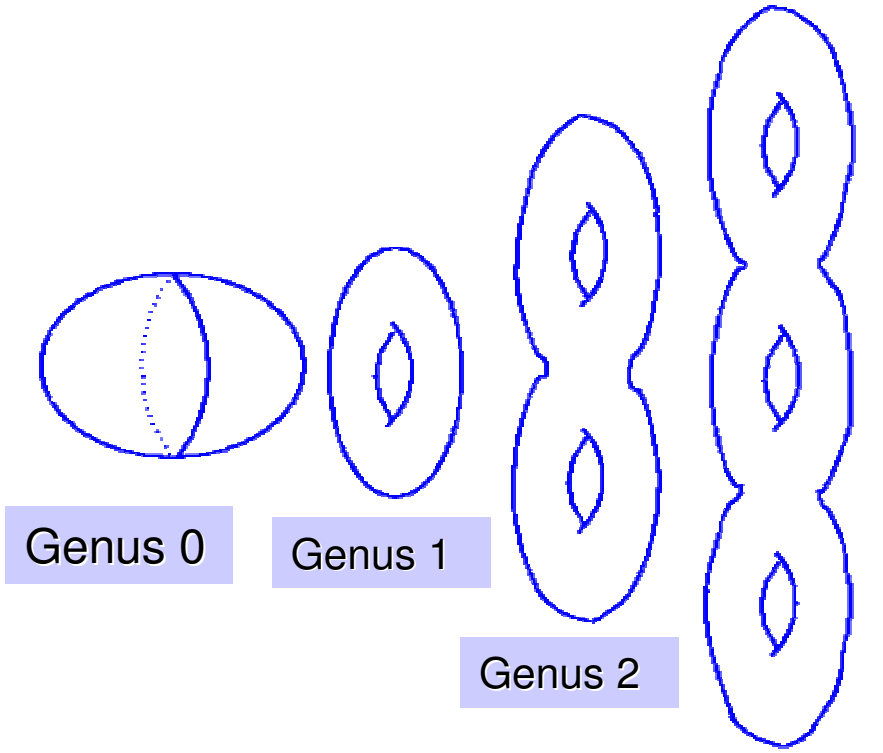
As propriedades topológicas de um modelo não são alteradas por **dividir** uma face em duas adicionando um lado , ou por **colar** duas faces em um lado comum.

Se vértices ou **lados** forem subdivididos ou unidos.

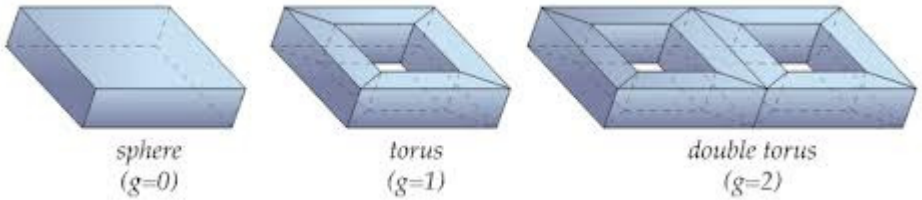


Topologicamente equivalentes:
Rubber sheet deformation
Massinha de modelar = "Play-Doh"

Op. Topológicas Locais

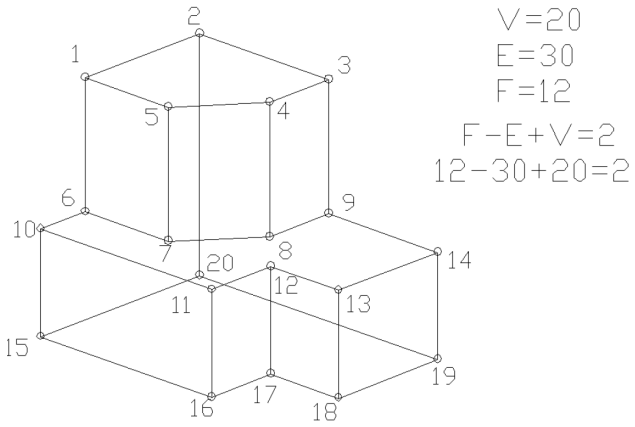


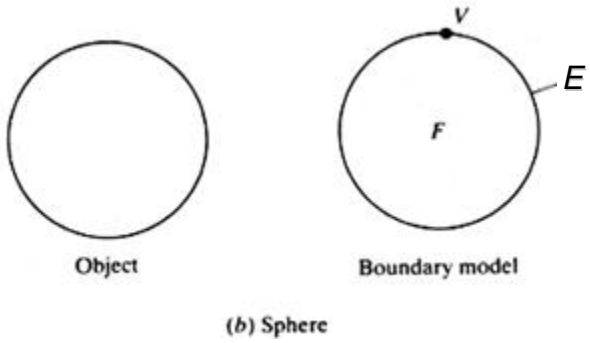
Op. Topologicas Globais = mudam o genus . Como a soma de dois torus

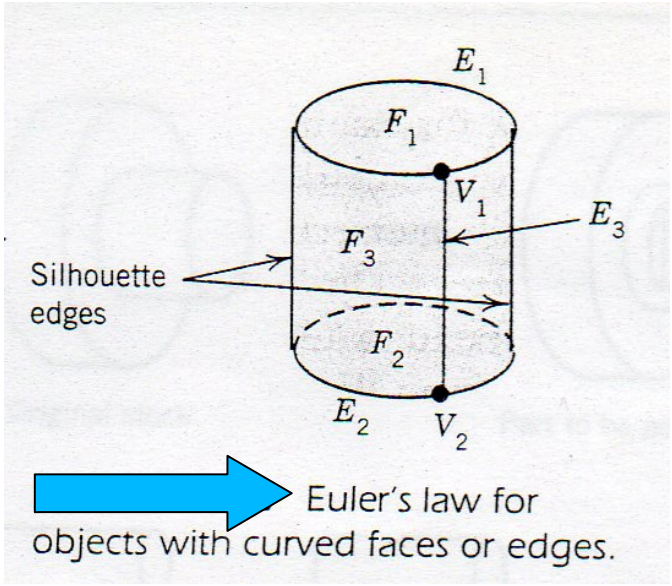


Leonhard Euler

(1707-1783)







Euler's Law

Se o objeto tem componentes múltiplos S ou C ≠ 1 ,
G (genus) é a soma dos Gs de cada objeto

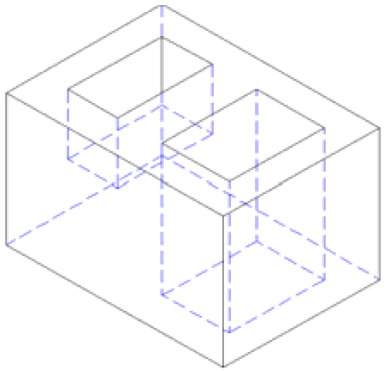
Euler-Poincare Law:

V - E + F - L = 2(S - G)

L = # of inner face loops (a loop contained entirely within another face loop)

S = # of shell bodies (sometimes "C")

G = # thru holes, A.K.A genus (# of passage features)



V - E + F - L = 2 (S – G)
24 - 36 + 15 - 3 = 2 (1 - 1)

Jules Henri [illegible] 1912).
considered to be one of the founders of the field of Topologia

Euler Operators

Model is topologically valid if satisfies Euler-Poincare

Validity relationship allows for definition of a set of operators:

- known as ***Euler Operators***
- allow faces, edges and vertices to be added and removed from model while retaining validity

Data structure

- Polygon-based (Face list)
- Vertex-based
- Edge-based
  - Radial edge data structure (for non-manifold)
  - Wing-edge data structure (for manifold, but support non-manifold with some modifications)
  - Half-edge data structure (for manifold only)

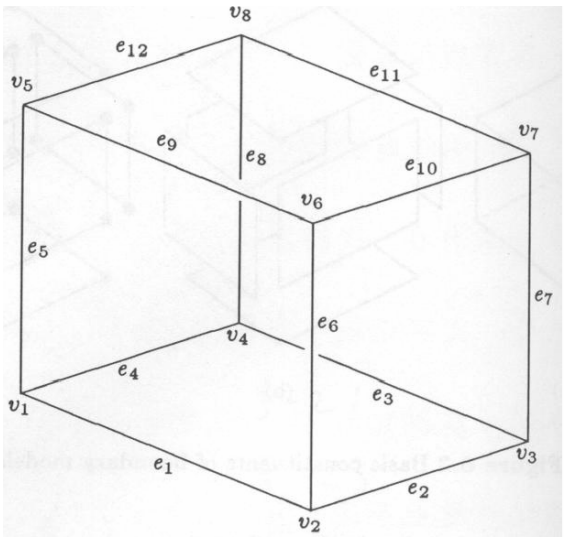
Adjacency queries

- Which faces use this vertex?
- Which edges use this vertex?
- Which faces border this edge?
- Which edges border this face?
- Which faces are adjacent to this face?

Estrutura de dados baseada em Vértice

vertex	*coordinates*
v_1	$x_1\ y_1\ z_1$
v_2	$x_2\ y_2\ z_2$
v_3	$x_3\ y_3\ z_3$
v_4	$x_4\ y_4\ z_4$
v_5	$x_5\ y_5\ z_5$
v_6	$x_6\ y_6\ z_6$
v_7	$x_7\ y_7\ z_7$
v_8	$x_8\ y_8\ z_8$

face	*vertices*
f_1	$v_1\ v_2\ v_3\ v_4$
f_2	$v_6\ v_2\ v_1\ v_5$
f_3	$v_7\ v_3\ v_2\ v_6$
f_4	$v_8\ v_4\ v_3\ v_7$
f_5	$v_5\ v_1\ v_4\ v_8$
f_6	$v_8\ v_7\ v_6\ v_5$

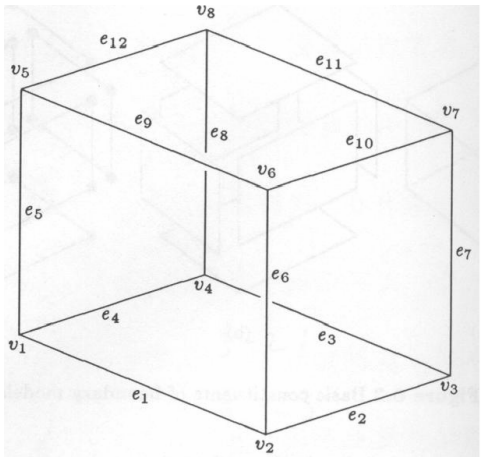


Baseada em lados (edges)

Lados são considerados orientados.

Cada lado pertence a duas faces.

Faces são consideradas orientadas, positivas se sua lista de lados apontar para fora se for no sentido horário



edge	*vertices*
e_1	$v_1\ v_2$
e_2	$v_2\ v_3$
e_3	$v_3\ v_4$
e_4	$v_4\ v_1$
e_5	$v_1\ v_5$
e_6	$v_2\ v_6$
e_7	$v_3\ v_7$
e_8	$v_4\ v_8$
e_9	$v_5\ v_6$
e_{10}	$v_6\ v_7$
e_{11}	$v_7\ v_8$
e_{12}	$v_8\ v_5$

vertex	*coordinates*
v_1	$x_1\ y_1\ z_1$
v_2	$x_2\ y_2\ z_2$
v_3	$x_3\ y_3\ z_3$
v_4	$x_4\ y_4\ z_4$
v_5	$x_5\ y_5\ z_5$
v_6	$x_6\ y_6\ z_6$
v_7	$x_7\ y_7\ z_7$
v_8	$x_8\ y_8\ z_8$

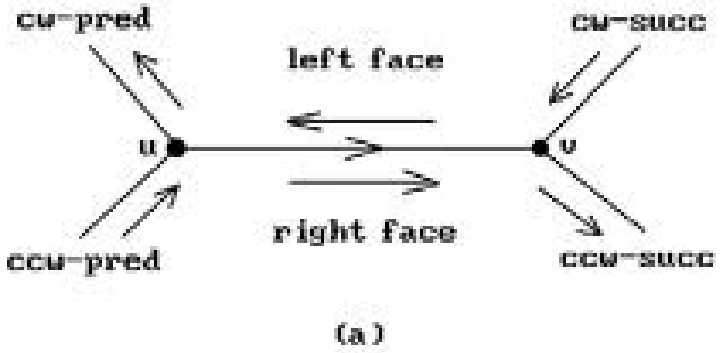
face	*edges*
f_1	$e_1\ e_2\ e_3\ e_4$
f_2	$e_9\ e_6\ e_1\ e_5$
f_3	$e_{10}\ e_7\ e_2\ e_6$
f_4	$e_{11}\ e_8\ e_3\ e_7$
f_5	$e_{12}\ e_5\ e_4\ e_8$
f_6	$e_{12}\ e_{11}\ e_{10}\ e_9$

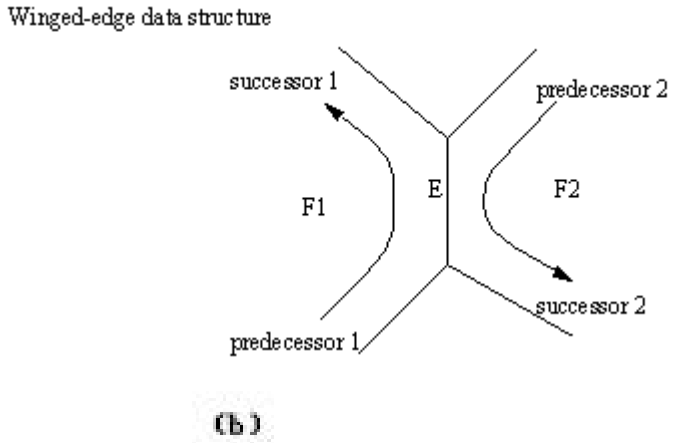
Winged edge data struture

A estrutura de lados anterior é adicionada a informação do vértice inicial e final cada lado.

Cada lado pertence a duas faces. Cada face tem os lados orientados no sentido dos ponteiros do relógio (cw=clockwise) .

Mais 2 lados são incluídos na estrutura os próximos lados da face vizinha cuja orientação dos vértices do lado é no sentido horário (cw) e no anti-horário (counter clock wise – ccw)

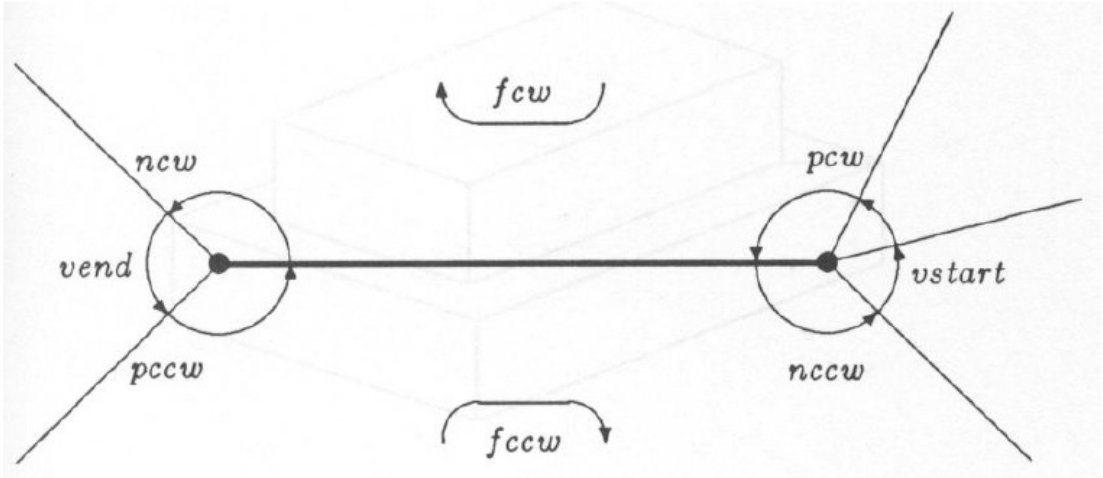


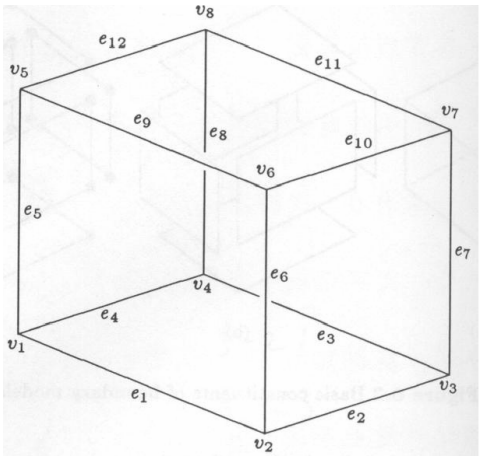


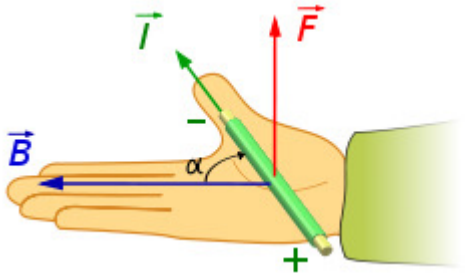
Estruturas de dados para permitir as o uso dos operadores de Euler

Winged edge data struture

fcw: face in clockwise
fccw: face in counterclockwise
ncw: next clockwise
nccw: next counterclockwise
pcw: previous clockwise
pccw: previous counterclockwise







edge	vstart	vend	ncw	nccw
e_1	v_1	v_2	e_2	e_5
e_2	v_2	v_3	e_3	e_6
e_3	v_3	v_4	e_4	e_7
e_4	v_4	v_1	e_1	e_8
e_5	v_1	v_5	e_9	e_4
e_6	v_2	v_6	e_{10}	e_1
e_7	v_3	v_7	e_{11}	e_2
e_8	v_4	v_8	e_{12}	e_3
e_9	v_5	v_6	e_6	e_{12}
e_{10}	v_6	v_7	e_7	e_9
e_{11}	v_7	v_8	e_8	e_{10}
e_{12}	v_8	v_5	e_5	e_{11}

vertex	coordinates
v_1	$x_1\ y_1\ z_1$
v_2	$x_2\ y_2\ z_2$
v_3	$x_3\ y_3\ z_3$
v_4	$x_4\ y_4\ z_4$
v_5	$x_5\ y_5\ z_5$
v_6	$x_6\ y_6\ z_6$
v_7	$x_7\ y_7\ z_7$
v_8	$x_8\ y_8\ z_8$

face	first edge	sign
f_1	e_1	+
f_2	e_9	+
f_3	e_6	+
f_4	e_7	+
f_5	e_{12}	+
f_6	e_9	−

For orientation of the edge

Face loop :

+ : follow ncw - : follow nccw

Loop of v_1 : e_1

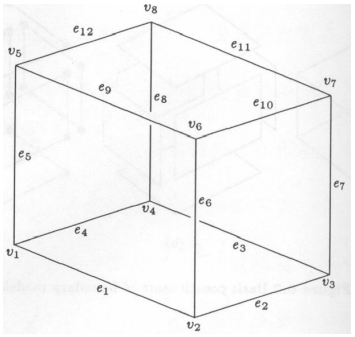
$e_5 = \mathrm{nccw}(e_1)$

$e_4 = \mathrm{nccw}(e_5)$

Full Winged edge data struture

Inclui informações do lado predecessor e das faces que dividem o lado.

edge	*vstart*	*vend*	*fcw*	*fccw*	*ncw*	*pcw*	*nccw*	*pccw*
e_1	v_1	v_2	f_1	f_2	e_2	e_4	e_5	e_6
e_2	v_2	v_3	f_1	f_3	e_3	e_1	e_6	e_7
e_3	v_3	v_4	f_1	f_4	e_4	e_2	e_7	e_8
e_4	v_4	v_1	f_1	f_5	e_1	e_3	e_8	e_5
e_5	v_1	v_5	f_2	f_5	e_9	e_1	e_4	e_{12}
e_6	v_2	v_6	f_3	f_2	e_{10}	e_2	e_1	e_9
e_7	v_3	v_7	f_4	f_3	e_{11}	e_3	e_2	e_{10}
e_8	v_4	v_8	f_5	f_4	e_{12}	e_4	e_3	e_{11}
e_9	v_5	v_6	f_2	f_6	e_6	e_5	e_{12}	e_{10}
e_{10}	v_6	v_7	f_3	f_6	e_7	e_6	e_9	e_{11}
e_{11}	v_7	v_8	f_4	f_6	e_8	e_7	e_{10}	e_{12}
e_{12}	v_8	v_5	f_5	f_6	e_5	e_8	e_{11}	e_9



vertex	*first edge*	*coordinates*
v_1	e_1	$x_1\ y_1\ z_1$
v_2	e_2	$x_2\ y_2\ z_2$
v_3	e_3	$x_3\ y_3\ z_3$
v_4	e_4	$x_4\ y_4\ z_4$
v_5	e_9	$x_5\ y_5\ z_5$
v_6	e_{10}	$x_6\ y_6\ z_6$
v_7	e_{11}	$x_7\ y_7\ z_7$
v_8	e_{12}	$x_8\ y_8\ z_8$

face	*first edge*
f_1	e_1
f_2	e_9
f_3	e_6
f_4	e_7
f_5	e_{12}
f_6	e_9

Por razoes históricas os operadores são descritos pela primeira letras dos seus nomes mnemônicos

M → *make* (criar)

K → *kill* (deletar)

S → *split* (separar)

J → *join* (juntar)

V → *vertex* (vértice)

E → *edge* (aresta)

F → *face* (face)

S → *solid* (sólido)

H → *hole* (furo)

R→ *ring* (anel)

Conceitos básicos:

Outro termo importante é o *loop* que pode ser descrito como mais uma entidade topológica e representa o ciclo que as arestas fazem em torno de uma face.

Por exemplo, o operador "*mev*" significa *Make Edge, Vertex*

operadores de Euler terão operadores inversos correspondentes, que podem desfazer o efeito do operador inicial. O inverso do MVFS é o KVFS.

exemplos:

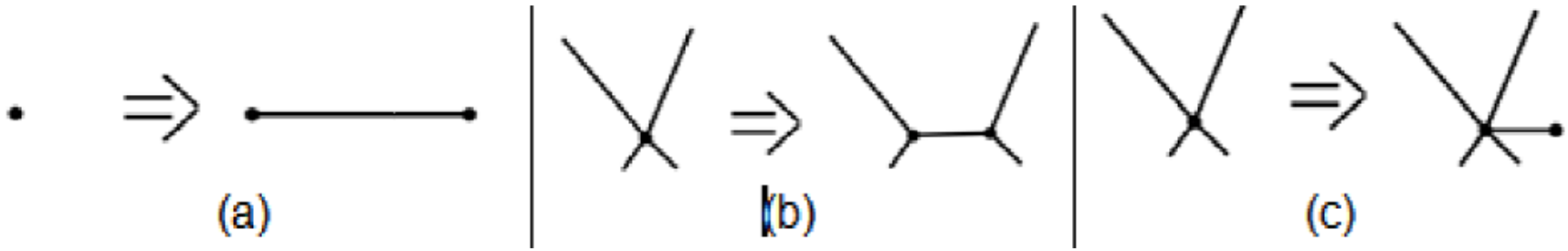
- **MEV (Make Edge Vertex) e KEV (Kill Edge Vertex)**

Este operador subdivide um vértice em dois, unindo-os por uma nova aresta, tendo como efeito a adição de um vértice e uma aresta à estrutura de dados.

três situações distintas,

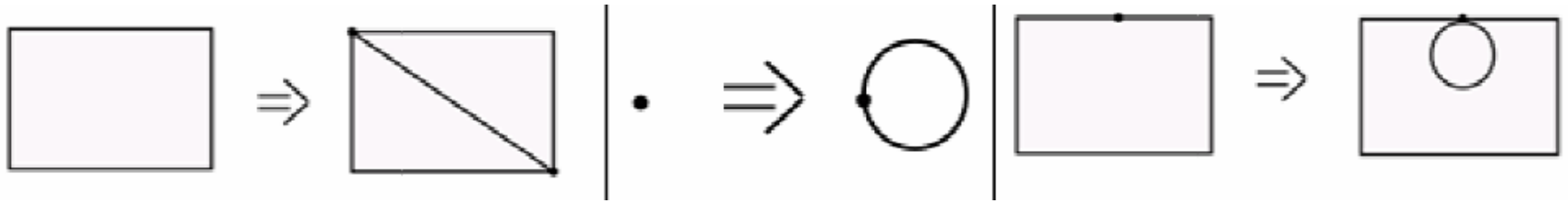
o vértice sem aresta é dividido

em dois, que são unidos pela aresta criada; (b) subdividindo o ciclo de arestas de um vértice em dois ciclos através da divisão de um vértice em dois, juntando-os com uma nova aresta; (c) gerando um novo vértice que é unido a um vértice já existente, através de uma nova aresta criada.

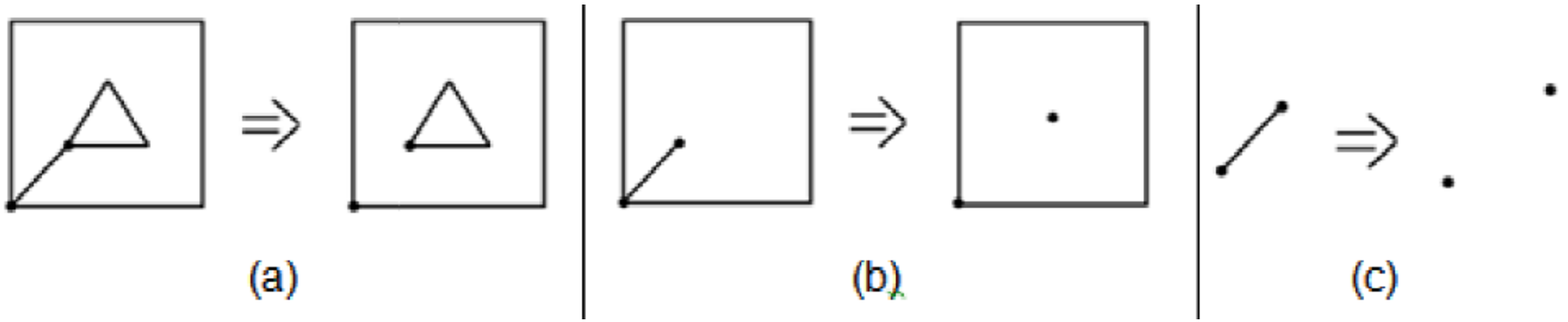


MEF (Make Edge Face) e KEF (Kill Edge Face)

Este operador subdivide um *loop*, fazendo a ligação de dois vértices por uma nova aresta, ou seja, adiciona uma nova aresta e uma nova face à estrutura de dados.



KEMR (Kill Edge Make Ring) e MEKR (Make Edge Kill Ring)

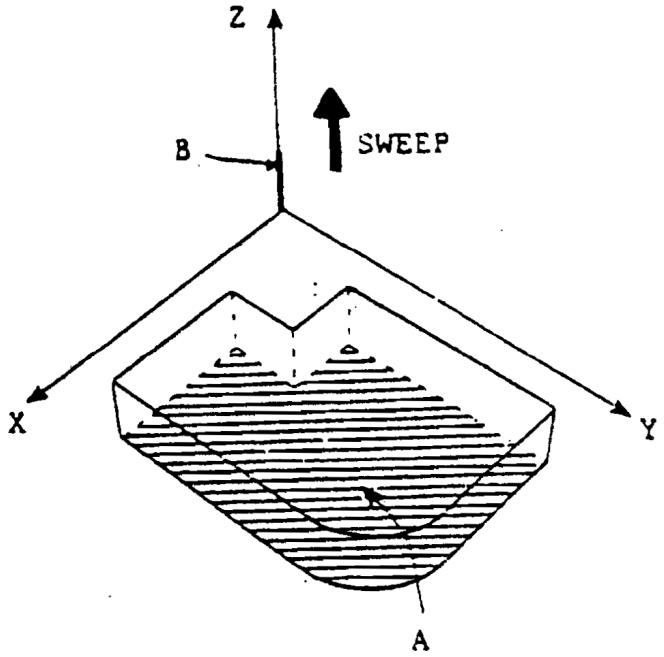


Sample Euler Operators

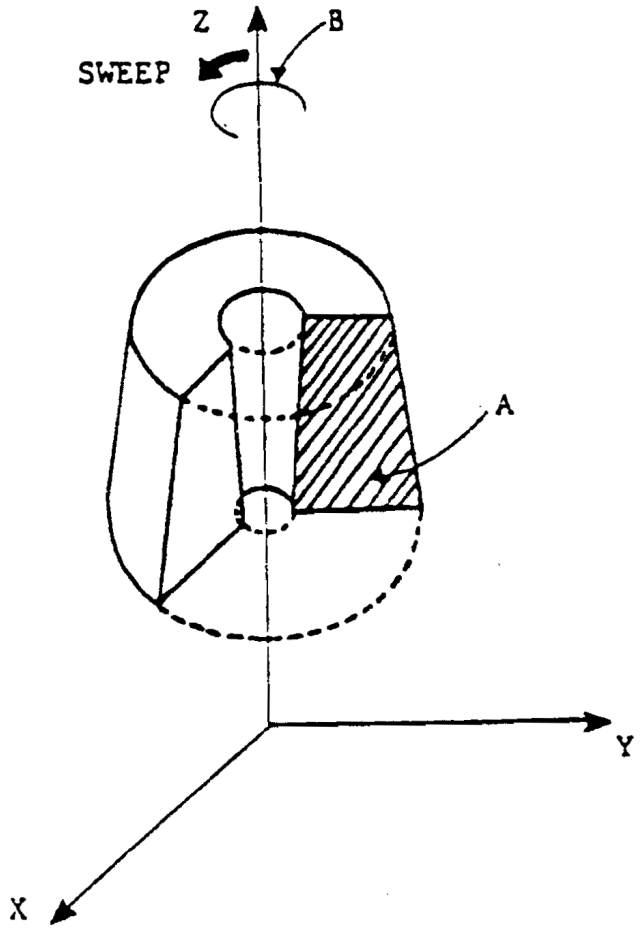
As letras correspondem aos elementos da fórmula e os números aos elementos que

	v e f g s l	
mvfs	1 0 1 0 1 0	make vertex,face,body
mev	1 1 0 0 0 0	make edge,vertex
mef	0 1 1 0 0 0	make edge,face
kemr	0-1 0 0 0 1	kill edge,make ring
kev	-1-1 0 0 0 0	kill edge,vertex
kef	0-1-1 0 0 0	kill edge,face
mekr	0 1 0 0 0-1	make edge,kill ring

Sweep representation



Translational sweeping.



Rotational sweeping.

Function representation (F-rep)

- A geometric object is defined by a single real function of several variables $f(x_1, x_2, ..., x_n) \geq 0$.

- A function can be defined analytically, with an evaluation algorithm, or with tabulated values and an appropriate interpolation procedure.

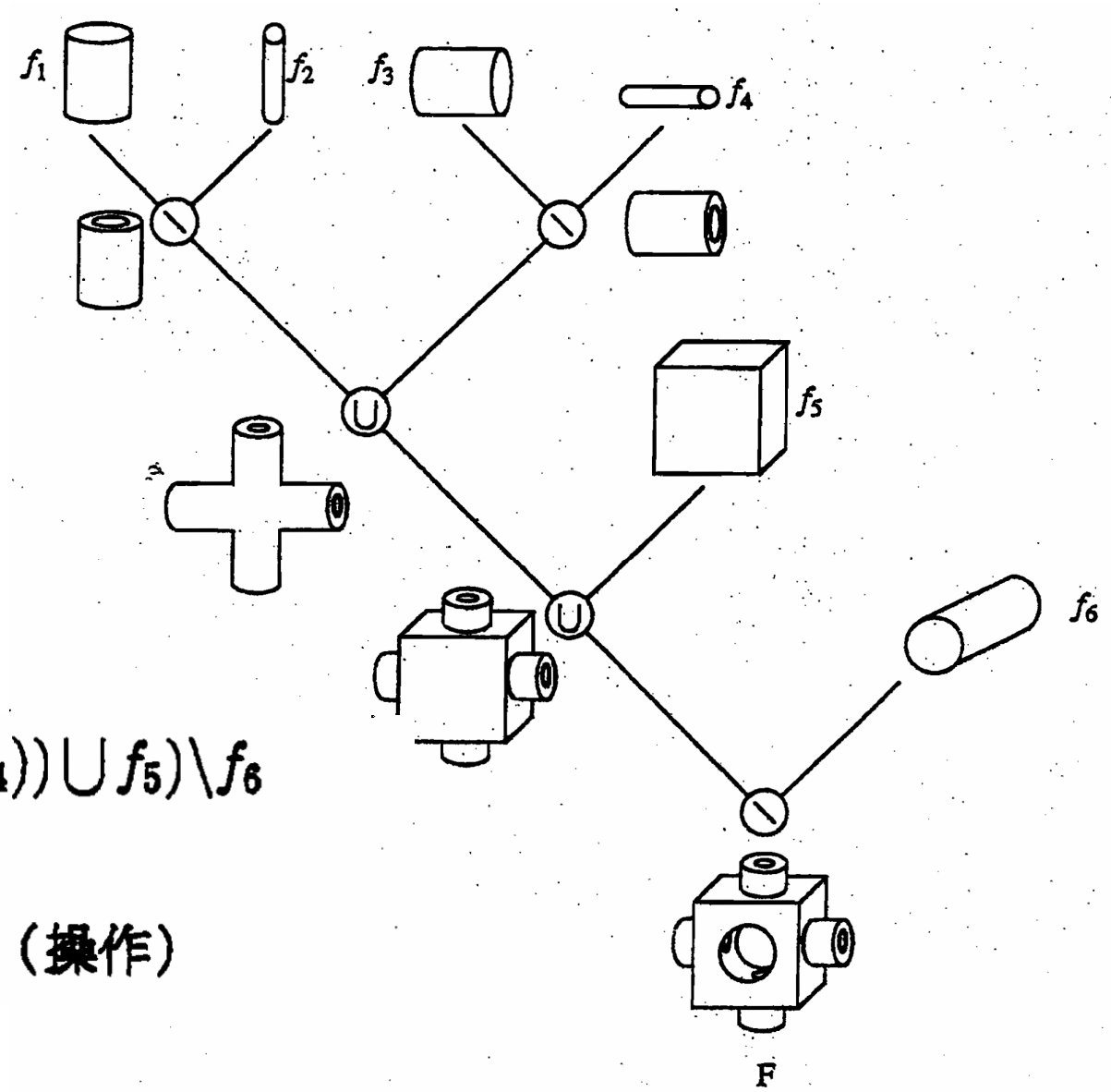
- An abstraction level higher than those of other known representations is provided. Combinations of the following modeling styles are supported: CSG, sweeping, blobby objects, volumetric objects. Dimension independent representation.

- Not closely related with B-rep.

- Time-consuming and requires parallel/distributed processing or hardware solutions.

Usado hoje na maioria dos modeladores

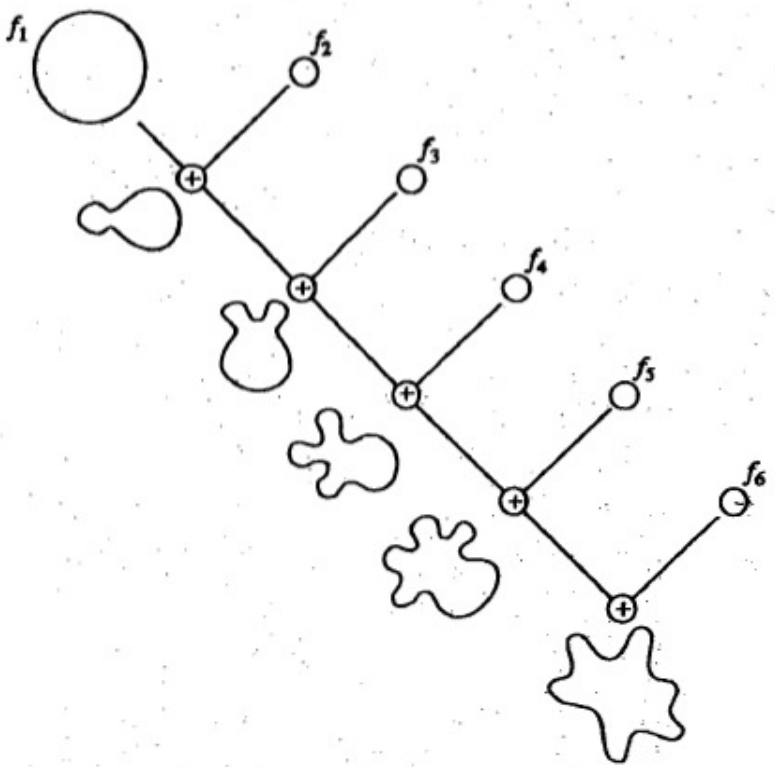
- F-rep is closed under the following operations:
  - set-theoretic operations with R-functions;
  - blending set-theoretic operations;
  - offsetting;
  - Cartesian product;
  - bijective mapping;
  - projection;
  - deformations;
  - metamorphosis.



$$F = ((f_1 \backslash f_2) \cup (f_3 \backslash f_4)) \cup f_5) \backslash f_6$$

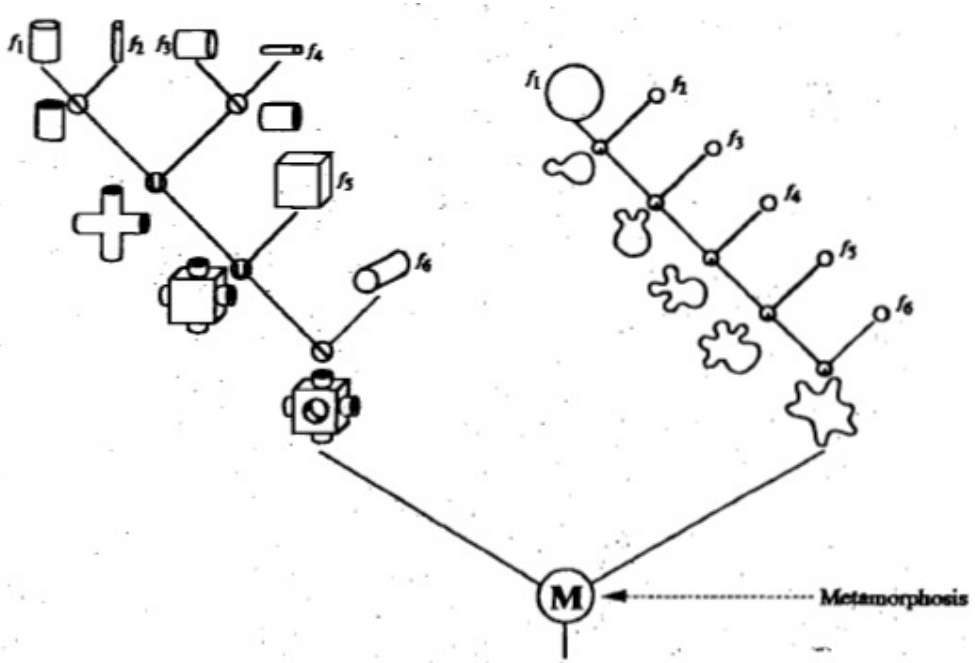
$\backslash, \cup$: R function （操作）

Shape Modeling and Computer Graphics with Real Functions
http://www.u-aizu.ac.jp/public/www/labs/sw-sm/FrepWWW/F-rep.html



$$f_i = b_i e^{-a_i r_i^2}$$

$$F_2 = ((((f_1 + f_2) + f_3) + f_4) + f_5) + f_6 = \sum_{i=1}^{6} f_i$$



$$M(t) = F_1 \cdot t + F_2(1 + t)$$

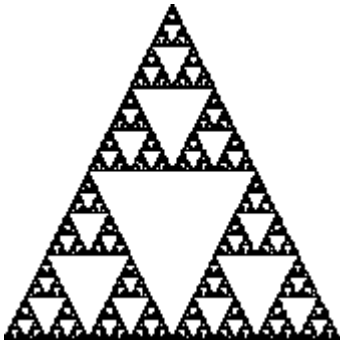
Geometria Fractal

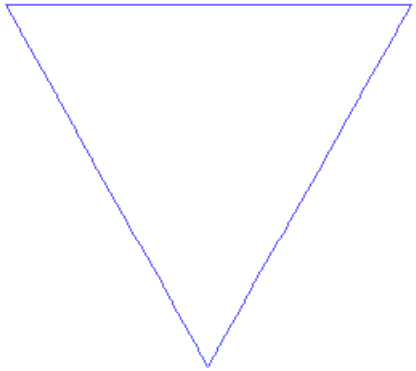
Estuda subconjuntos complexos de espaços métricos.

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

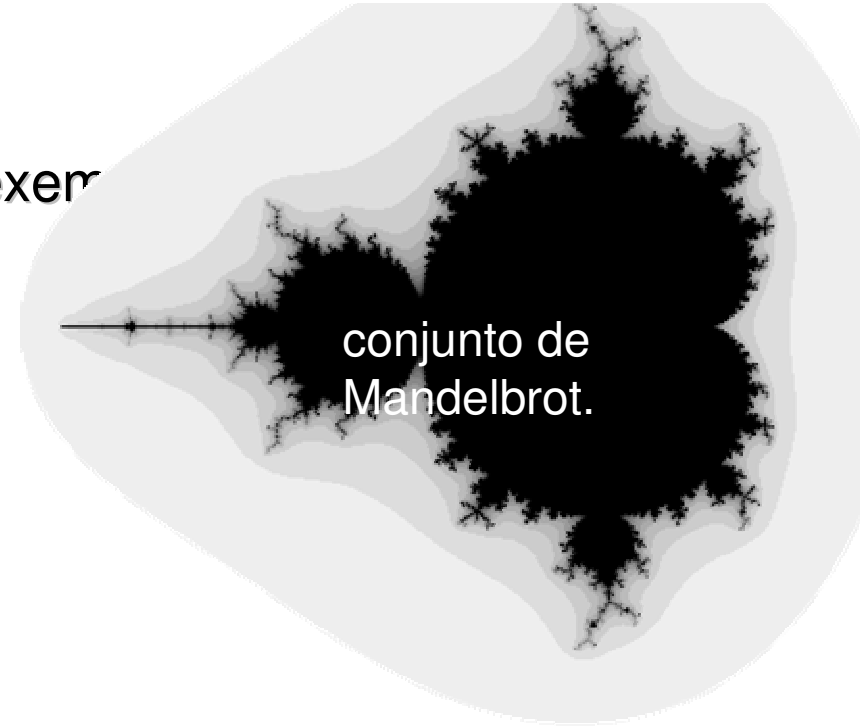
Um fractal é composto por partes reduzidas dele próprio

Fractais podem ser Determinísticas ou Randômica



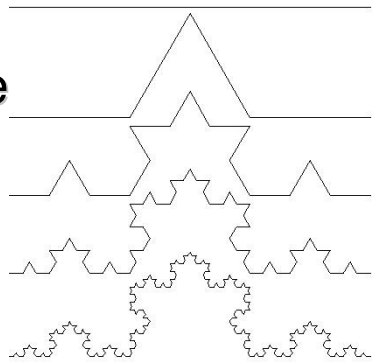


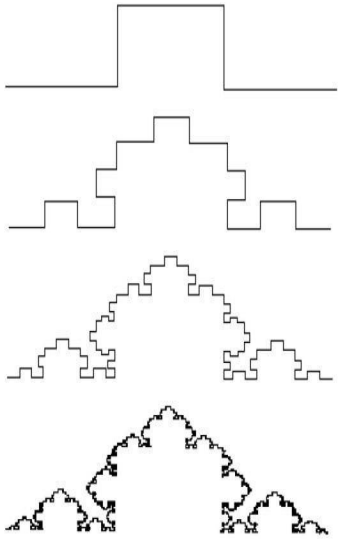
exem



Randômicos x determinísticos

Exemplos de





Curvas de KOCH:

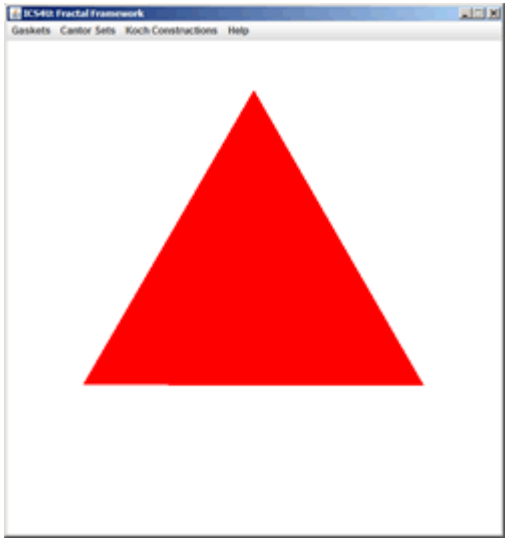
Proposta por Von Koch em 1904, tem a seguinte geração:

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 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.

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.

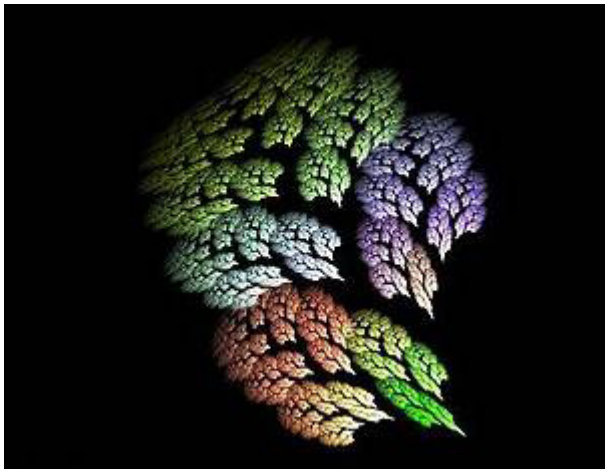
Em muitos casos um fractal pode ser gerado por um padrão repetido, tipicamente um processo recursivo ou iterativo.

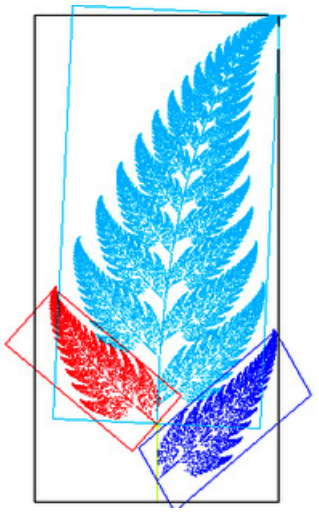
O termo foi criado em 1975 por Benoît Mandelbrot, matemático francês nascido na Polónia, a partir do adjetivo latino fractus, do verbo frangere, que significa quebrar e fração.

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





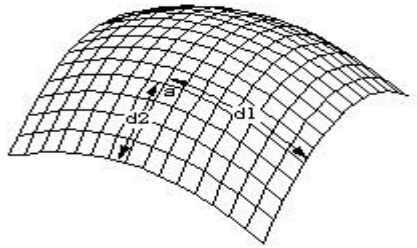
Dimensão Euclidiana – objetos euclidianos

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

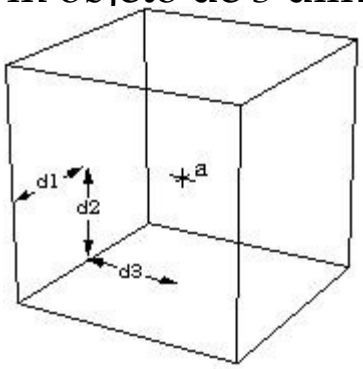
pode ser dividido em N partes , cada parte será idêntica a anterior multiplicada por um fator elevado a 1:

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

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:

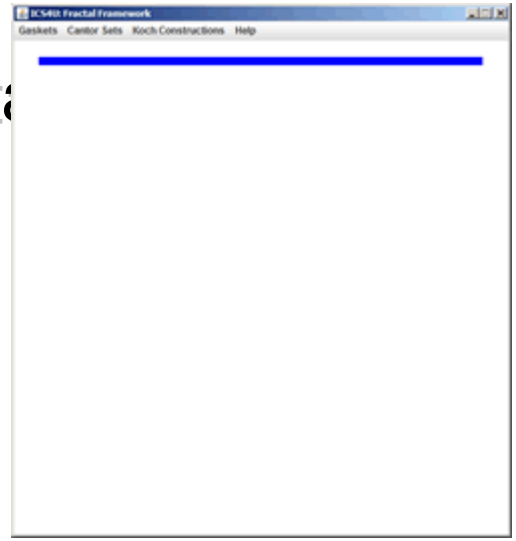
$$\log N = \log (1/ r)^{d}$$

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

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

Dimensão fracta

DF = log N / log



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

conjunto de Cantor

DF = log 2 / log (3) =

0,63092975357145743709952711434276

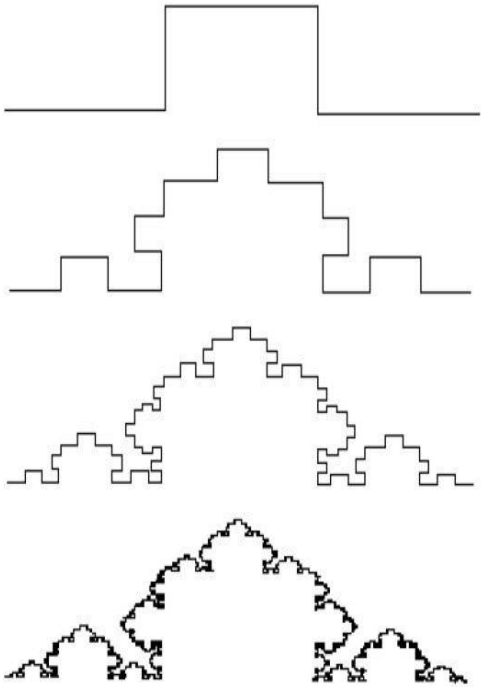
Dimensão fractal - *DF = log N / log (1/r)*

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

Curva quádrica de Koch

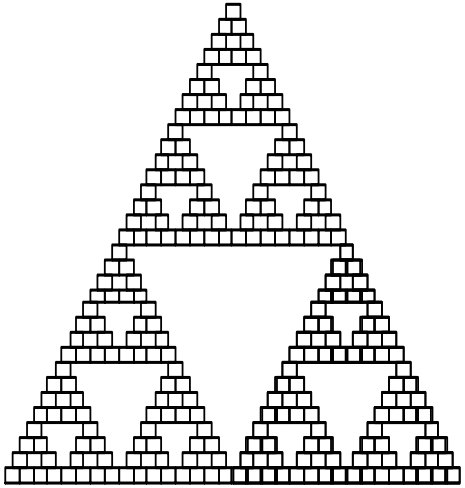
$$DF = log\ 5 / log\ (3) =$$

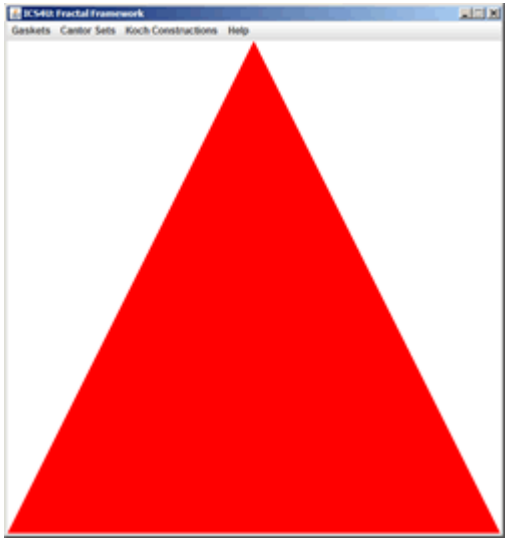
1,464973520717927167197040404076786



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

Triângulo de Sierpinski

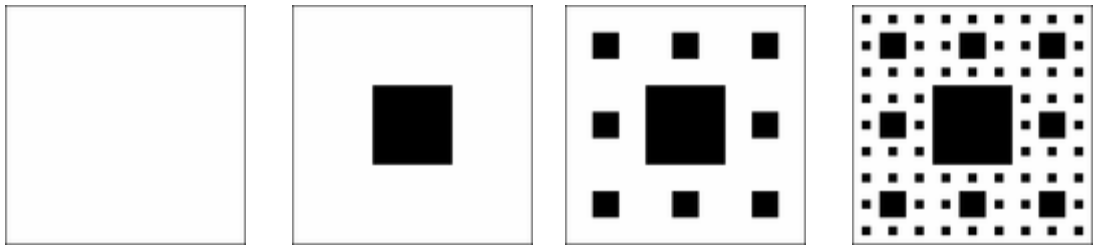


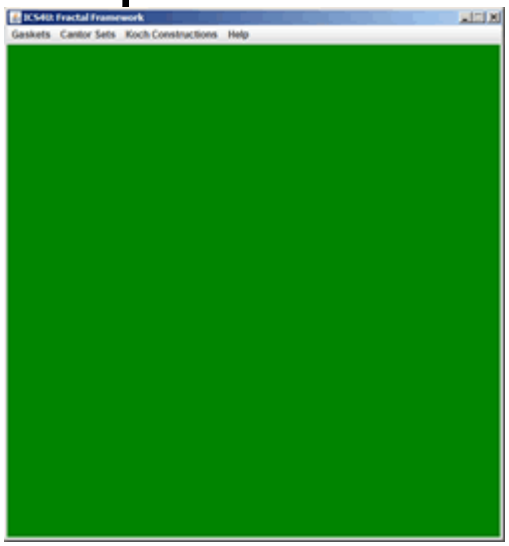


DF do Triângulo de Sierpinski:
DF = log 3 /l og2 =
1,584962500721156181453738943 9478

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





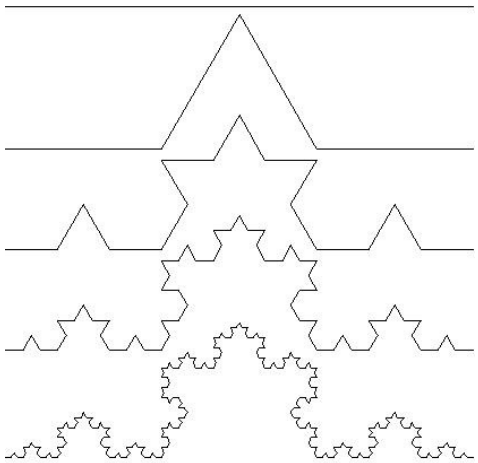
DF = log 8 /l og 3 = 1,892789260714372311298581343028 3

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

Curva triadic de Koch

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

1,261859507142914874199054228685 5

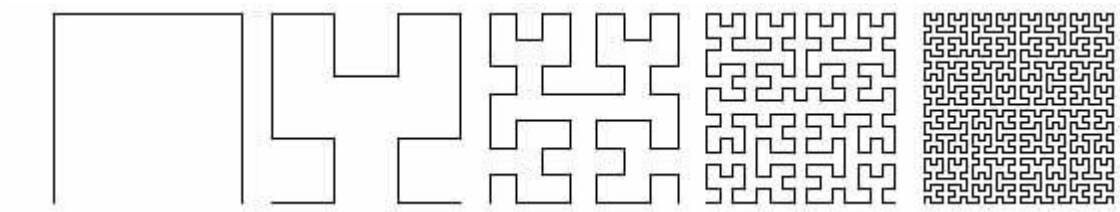


Curva de Peano.

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

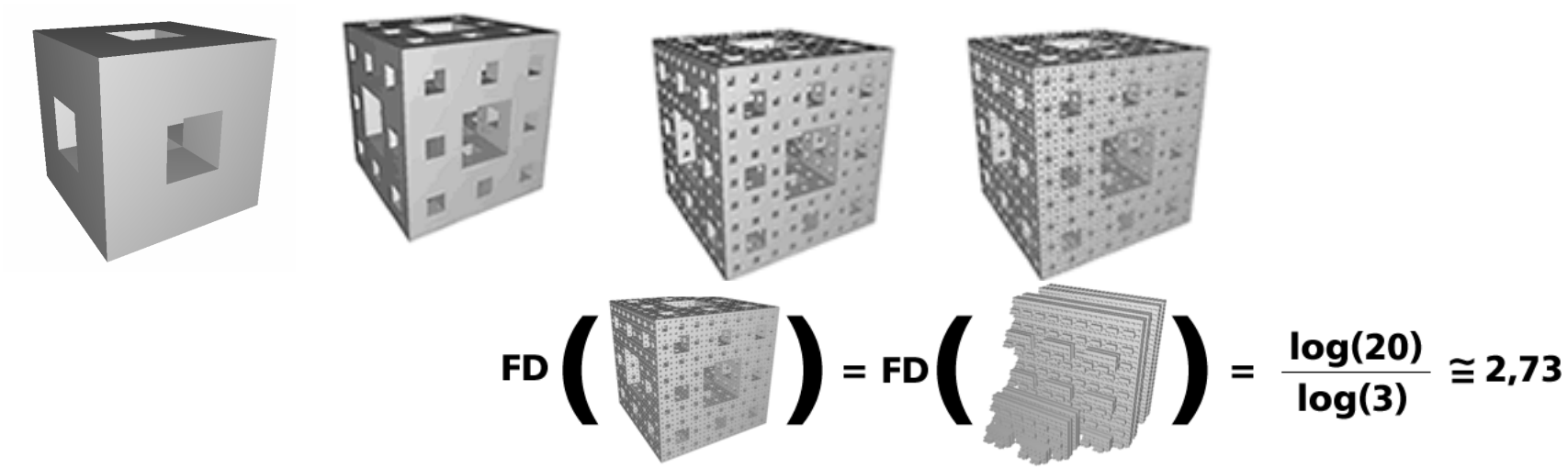
Dimensão fractal *DF = log N / log (1/r)*

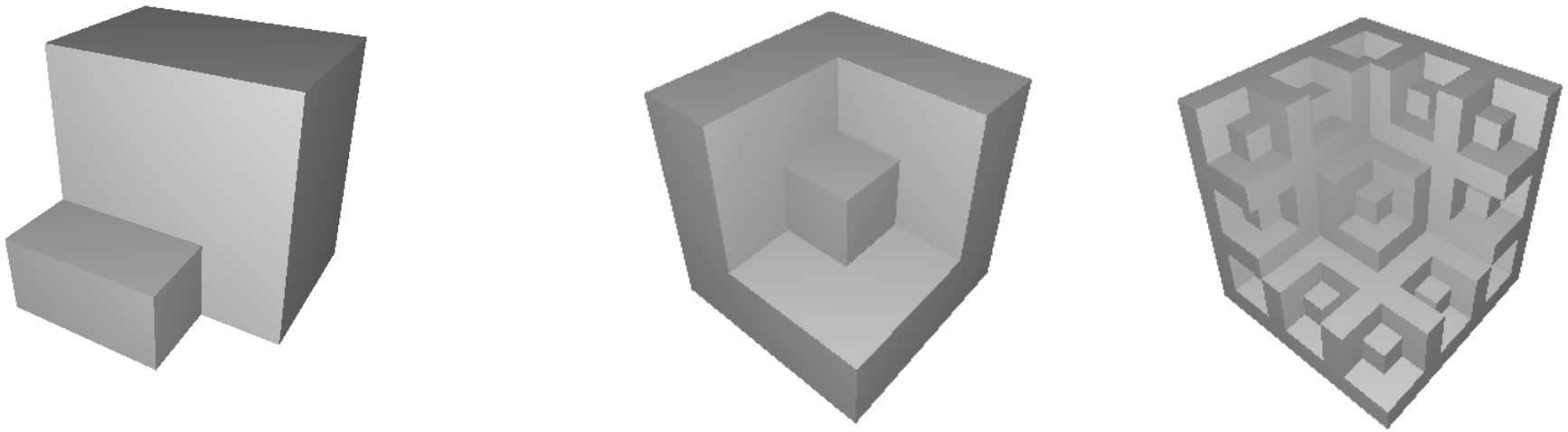
DF = log 4 / log (2) = 2



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

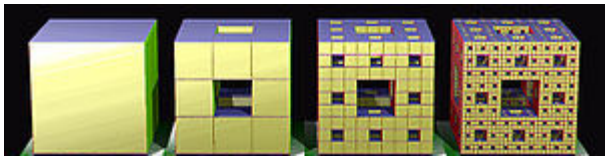
Dimensão fractal = fracionaria





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.

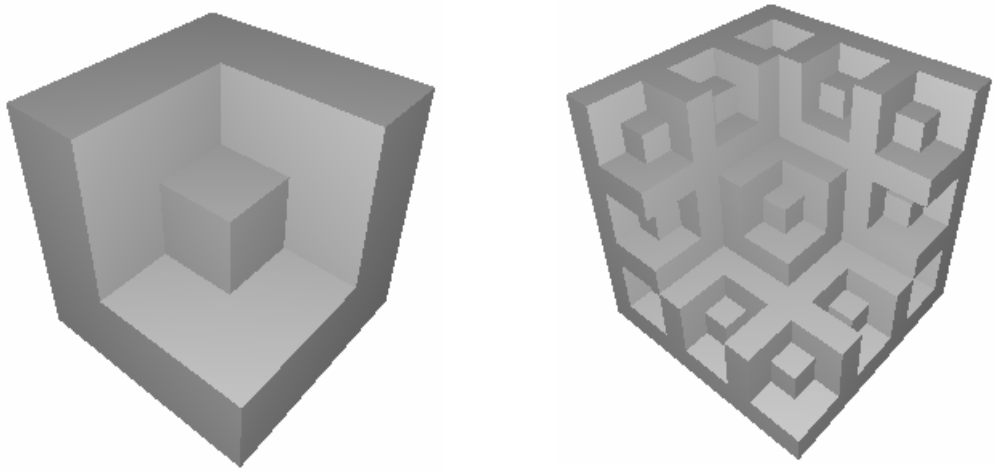
A segunda iteração dará uma esponja Nível 2, a terceira uma esponja Nível 3 , e assim por diante.

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

O número de cubos aumenta **20n em cada iteração**.

Lacunaridade e Sucolaridade tambem podem ser preciso

Local Lacunarity (LL)

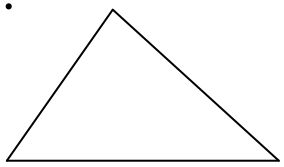


$$\Lambda(s) = \sum_{i=1}^{N} M^2\, Q(M, s) \;/\; \left(\sum_{i=1}^{N} M\, Q(M, s) \right)^2$$

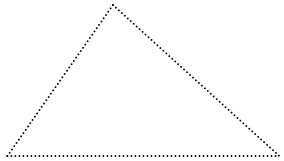
Um algoritmo simples de Montanhas

A superfície de uma montanha pode ser modelada como:

1- Desenhe um triângulo no espaço 3D.



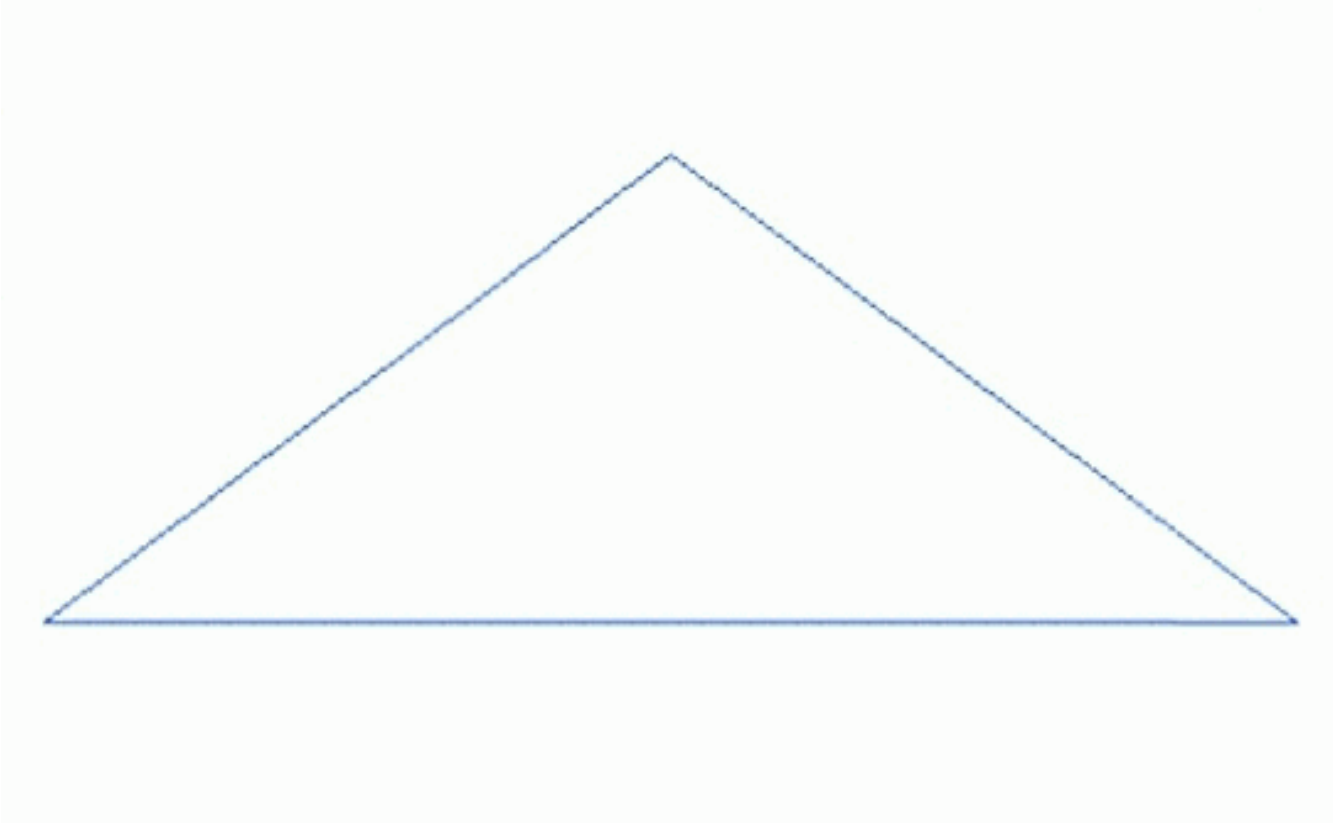
2- Ache os pontos centrais das 3 linhas que formam o triângulo e crie novos triângulos a partir desse triângulo.



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



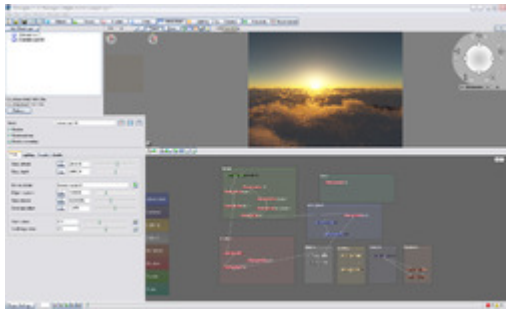
E as formas da natureza, como montanhas, arvores, nuvens, flores, frutas?

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.

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

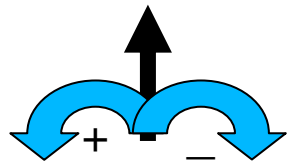
Por exemplo os símbolos **"f" , "+", "-" "!", "F", "@", "[", "]"** são símbolos usados em 2D.

"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."

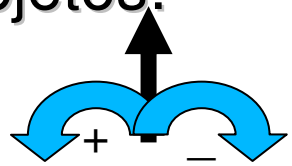
"]" Significa "redefina o estado de gráficos de volta ao armazenada".

"@" 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.

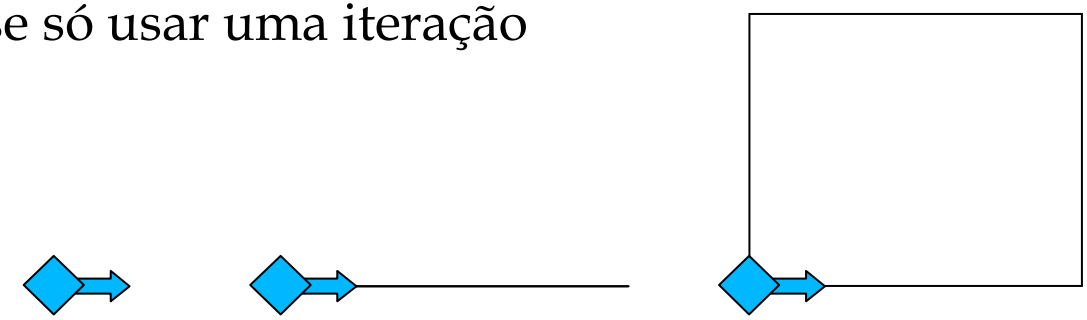
Com isso você pode gerar desenhos, ou texturas procedurais, ou objetos:

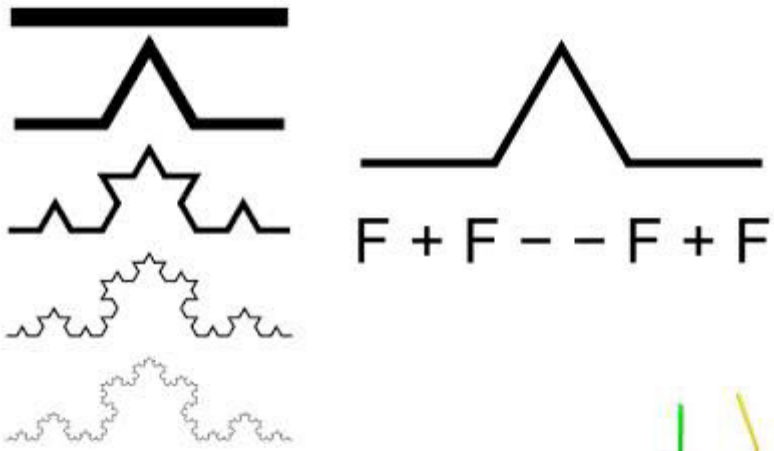


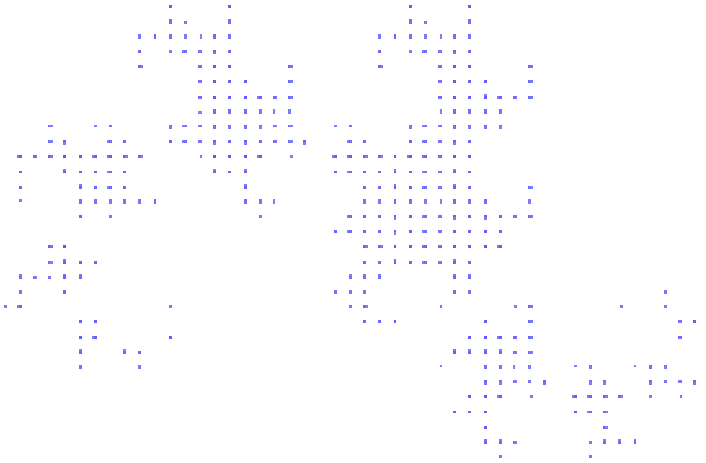
Você pode desenhar um quadrado como

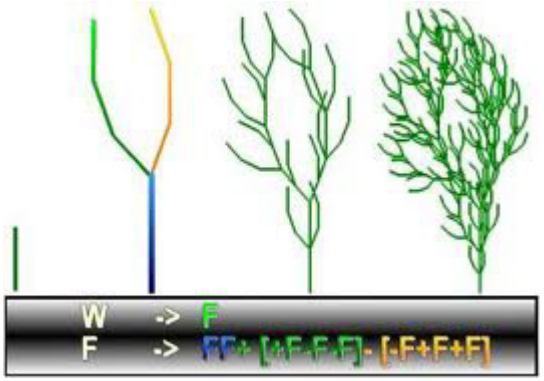
quadrado {
  angulo:= 90; /* giros de 90 graus
  Axioma :=f; /* seqüência inicial neste caso se supormos que estamos na horizontal e com orientação para da direita para a esquerda
  f ⟶ f + f + f + f; /* seqüência de substituição para f
  }

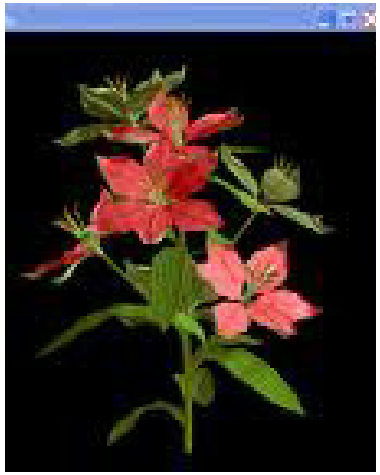
se só usar uma iteração




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.

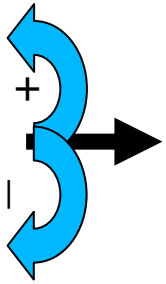
ilhaFractal {

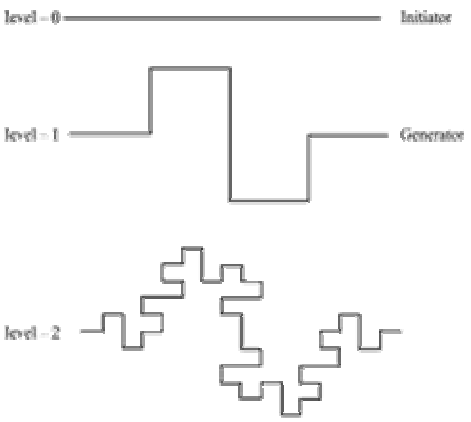
angulo:= 90; /* giros de 90 graus

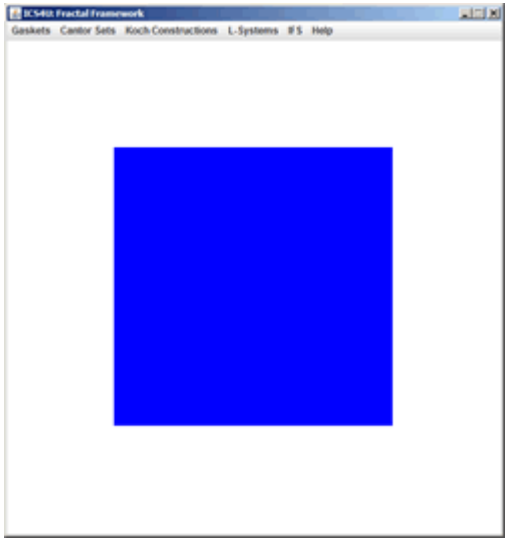
Axioma:= f-f-f-f; /* seqüência inicial neste caso se supormos que estamos na horizontal e com orientação para a direita

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

}





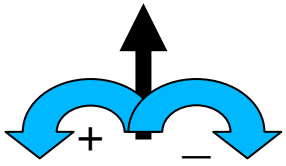


Com isso você pode gerar desenhos, ou texturas procedurais, ou objetos:

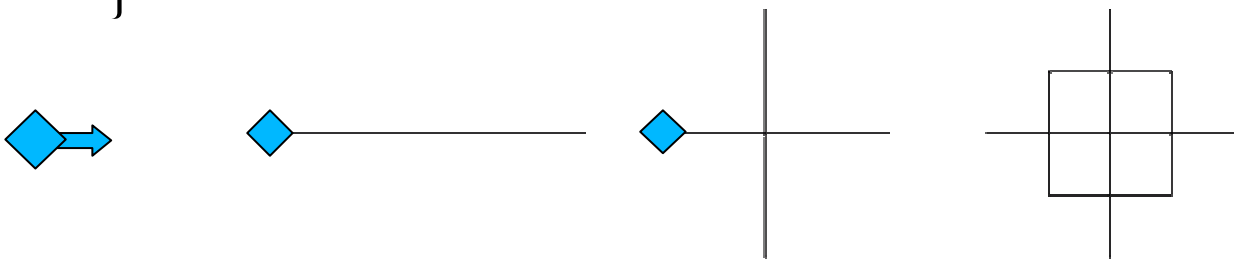
Você pode pensar em um quadrado como uma célula que se subdivide em quatro quadrados.

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

Vamos olhar para o seguinte L-system.

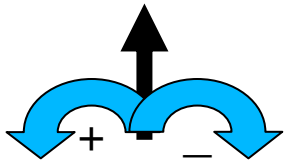



```
grade {
  angulo:= 90;  /* giros de 90  graus
  Axioma f;  /*  seqüência inicial neste caso se supormos que estamos
    na horizontal e com orientação para da direita para a esquerda
  f → @0,5 f [+ f] [- f] f;  /* seqüência de substituição para f
        }
```



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.

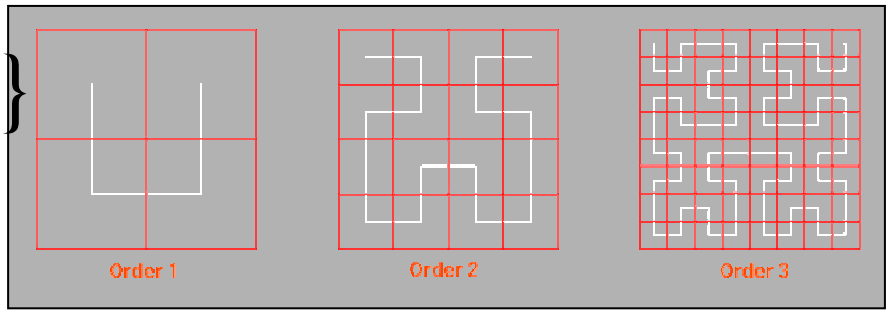


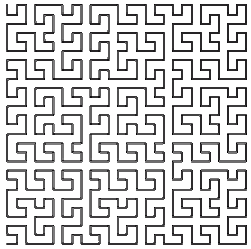
Hilbert { angulo:= 90; axioma := X;

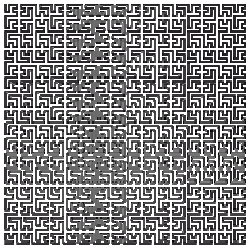
X → @0,5 -YF+XFX+FY- ;

Y → +XF-YFY-FX+ ;

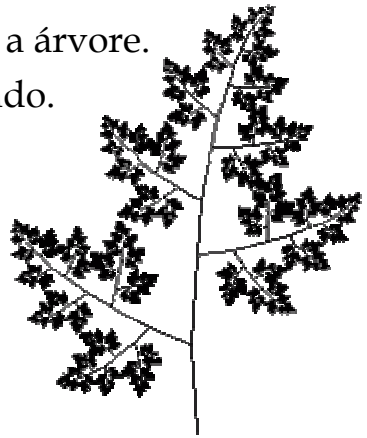
(curva de Hilbert de ordens 1, 2 e 3 com o quadrado,e depois só as linhas)







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.

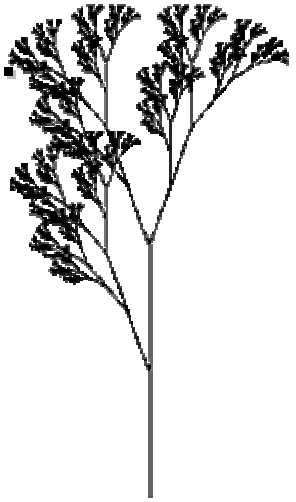
Em seguida, vire à direita ligeiramente (7,2 graus) e desenha outra parte da haste (-f).

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

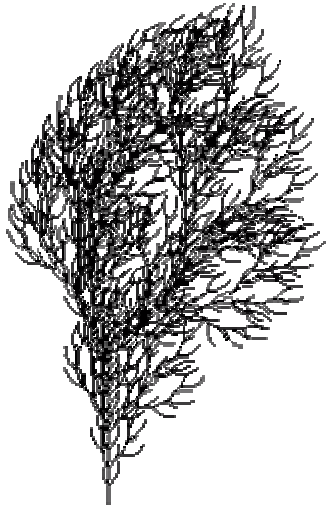
Como Lindenmayer (1925-1989) era botânico, com isso ele registrou o crescimento e forma de uma infinidade de espécies.



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 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

Context-sensitive L-systems

Produções em sistemas OL são livres de contexto; isto é aplicável independentemente do contexto em que o antecessor aparece. No entanto, a aplicação de produção também pode depender do contexto do antecessor. Este efeito é útil para simular as interações entre as partes da planta, por exemplo, devido ao fluxo de nutrientes ou hormonas. Várias extensões sensíveis ao contexto de L-sistemas têm sido propostos e estudados cuidadosamente no passado [62, 90, 128]. Sistemas de 2L usar produções de forma a <a> a ~ - ~ X, onde a letra a (o chamado predecessor estrito) pode produzir palavra X se e somente se um é precedido por carta pelo e seguido por um .. Assim, letras e na a. formam o contexto certo de um nesta produção e esquerdo. Produções em Il-sistemas têm unilateral único contexto; conseqüentemente, eles são ou do formulário em <a - ~ X ou a> ar - ~ X.-sistemas OL, IL-sistemas e 2L-sistemas pertencem a uma classe mais ampla de IL-sistemas, também chamados de (k, l) -Sistemas. Em um (k, l) -sistema, o contexto esquerda é uma palavra de comprimento k eo contexto direito é uma palavra de comprimento l. A fim de manter as especificações de L-sistemas curtos, a noção habitual de IL-sistemas foi modificado aqui, permitindo Vroductions contexto diferente com comprimentos de coexistir dentro de um único sistema. Além disso, as produções sensíveis ao contexto são assumidos ter precedência sobre produções livres de contexto com o mesmo predecessor estrito. Consequentemente, se um livre de contexto e uma produção contextual tanto se aplicam a uma determinada letra, a uma sensível ao contexto deve ser selecionada. Se nenhuma produção se aplica, esta letra é substituída por si só, como anteriormente assumido para sistemas OL. O exemplo a seguir pelo sistema IL faz uso do contexto para simular a propagação de sinal em toda a cadeia de símbolos: w: baaaaaaaa p ~: BG [H] M pode ser aplicado ao símbolo S na cadeia ABC [DE] [SG [HI [JK] L] MNO], que envolve pular sobre símbolos [de] na busca de contexto para a esquerda, e eu L [JK] na busca de contexto certo. Dentro do formalismo de L-sistemas enquadradas, o contexto esquerda pode ser utilizado para simul ~ te sinais de controlo que se propagam acropetally, ou seja, a partir da raiz ou folhas basais para os ápices da planta modelada, enquanto a direita representa contexto sinais que se propagam basipetally , ou seja, a partir dos ápices no sentido da raiz. Por exemplo, o seguinte sistema de IL-simula a propagação de um sinal de acrópeto em uma estrutura ramificada que não cresce: # ignore-: + - w: F ~ [~ + F] F ~ [-F ~] ~ F [F + ~] F ~ Pl:. ~ FbFb "Fb A operação de L-sistemas sensíveis ao contexto é analisado usando mais ex & mples obtido por Hogeweg e Hesper [64] Em 1974, eles publicaram os resultados de um estudo exaustivo de 3.584 padrões gerados por um classe de sistemas 2L enquadradas definidas sobre o alfabeto (0,1}. Alguns desses padrões tinham formas semelhantes a plantas. Posteriormente, Smith melhorou significativamente a qualidade das imagens geradas usando técnicas de imagem de computador state-of-the-art [136 , 137]. Exemplos de estruturas geradas por L-sistemas semelhantes ao proposto por Hogeweg e por Hes são mostrados na Figura 1.31. as diferenças estão relacionadas com a interpretação geométrica das cadeias resultantes. de acordo com a interpretação original, ramos consecutivos são emitidos alternadamente à esquerda e nd direito, enquanto interpretação tartaruga requer especificação explícita de ramificação ângulos dentro do sistema L.

Bibliografia:

M.A. Mortenson. Geometric modeling , Wiley, 1985

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

J.D.Foley,A.van Dam,S.K.Feiner,J.F.Hughes. Computer Graphics- Principles and Practice, Addison-Wesley, Reading, 1990.

M. Mäntylä, An introduction to solid modeling ,CS Press, 1988.

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