



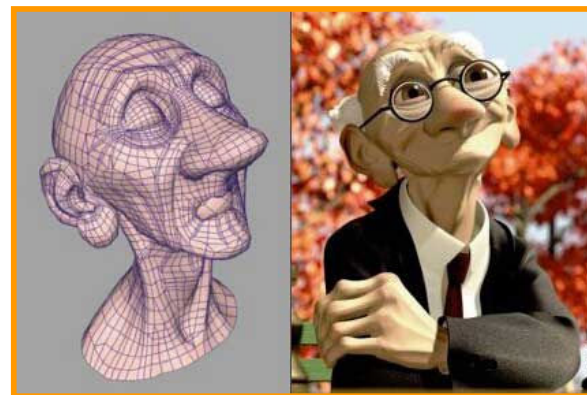
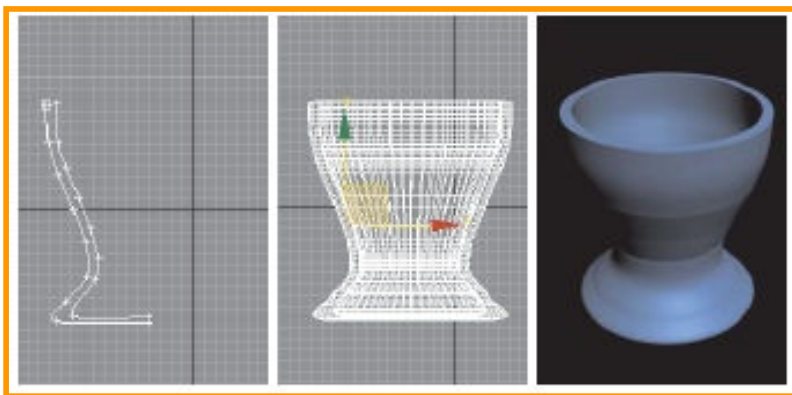
MODELAGEM DE SÓLIDOS, CURVAS E SUPERFÍCIES

Adair Santa Catarina
Curso de Ciência da Computação
Unioeste – Campus de Cascavel – PR

Jan/2021

O que é Modelagem?

Modelagem é o uso de técnicas para criar representações matemáticas ou simbólicas de objetos tridimensionais que, posteriormente, serão convertidas em imagens.



Representação de Objetos



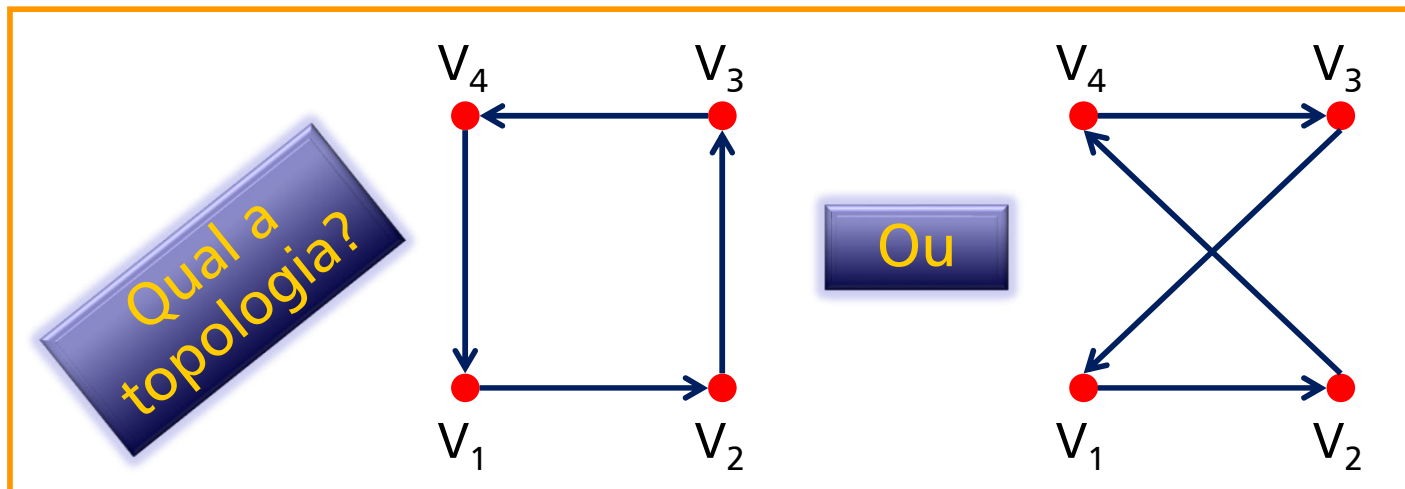
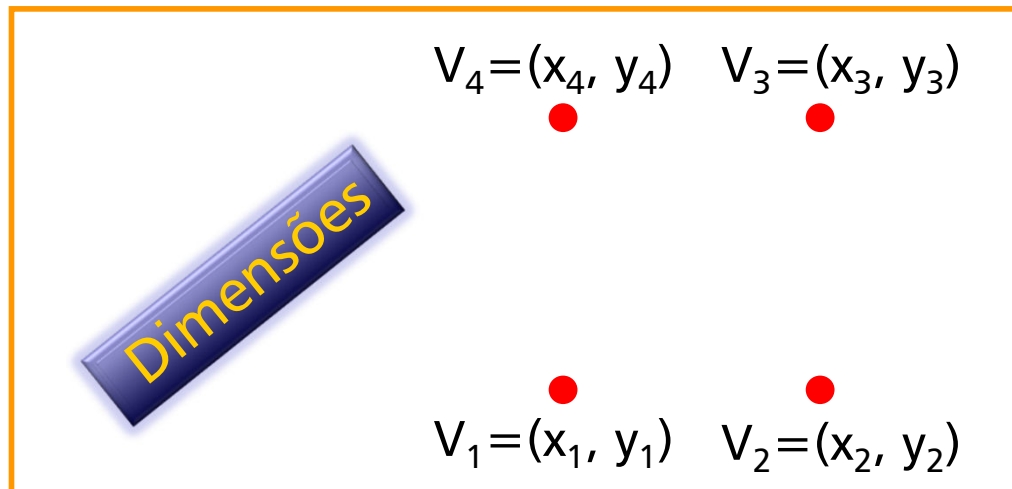
A representação matemática ou simbólica empregada na modelagem de objetos deve ser conveniente aos algoritmos gráficos utilizados.

A representação dos objetos deve incluir:

- Dimensões, por exemplo a localização dos vértices;
- Estrutura ou topologia: como os pontos são conectados para dar forma aos objetos.

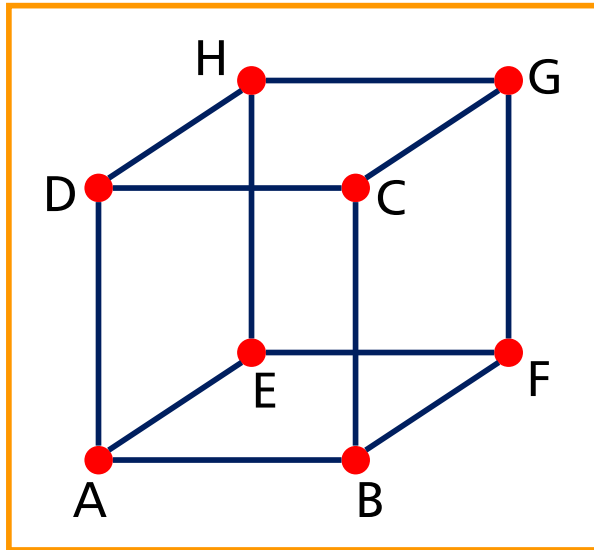


Exemplos



Exemplos

Isto é um cubo?



Sim
e
Não!

Vértices

$$A = (0, 0, 0)$$

$$B = (1, 0, 0)$$

$$C = (1, 1, 0)$$

$$D = (0, 1, 0)$$

$$E = (0, 0, 1)$$

$$F = (1, 0, 1)$$

$$G = (1, 1, 1)$$

$$H = (0, 1, 1)$$

Arestas

AB, BC,

CD, DA,

EF, FG,

GH, HE,

AE, BF,

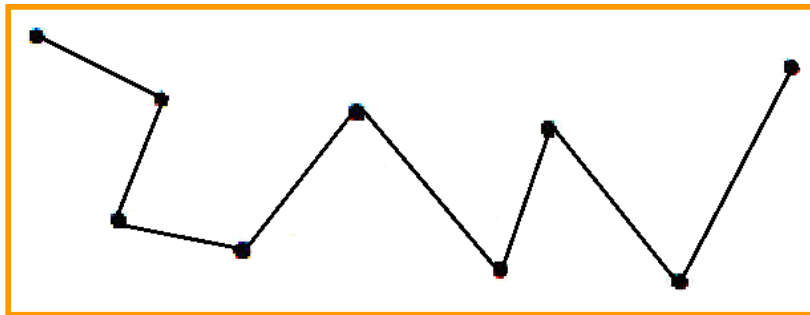
CG, DH

6 faces delimitam um volume fechado para o sólido. Então é um cubo!

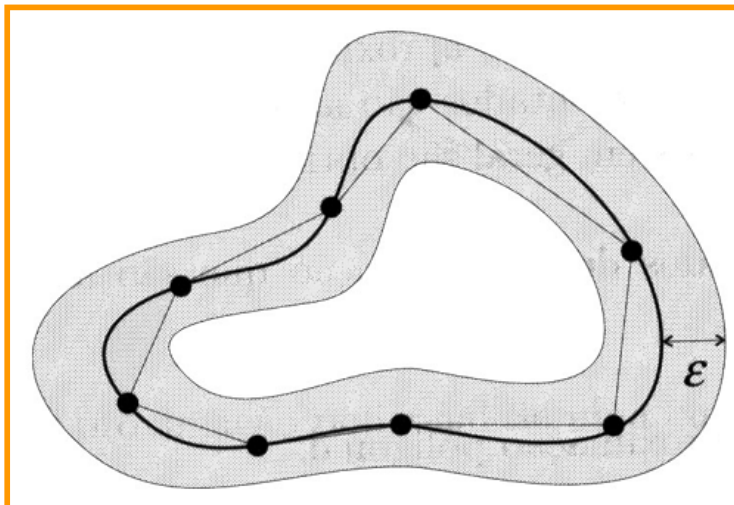


Aproximação de Curvas por *Polylines*

Resumidamente, curvas podem ser aproximadas por curvas poligonais (*polylines*)



Polyline



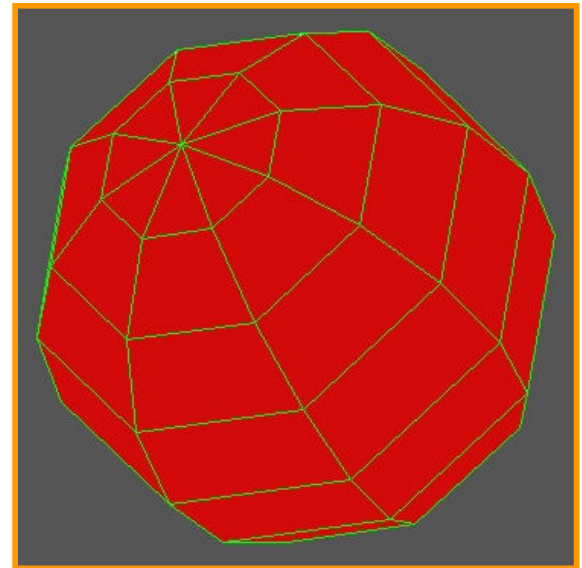
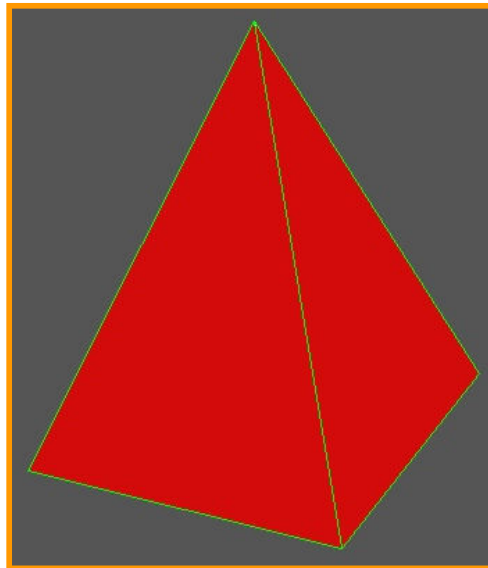
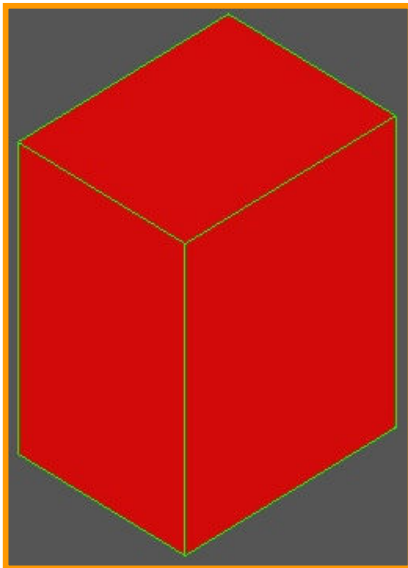
Curva
aproximada
ou
retificada



Representação Poliédrica de Objetos

Consiste em utilizar faces planas para representar uma aproximação de sólidos tridimensionais.

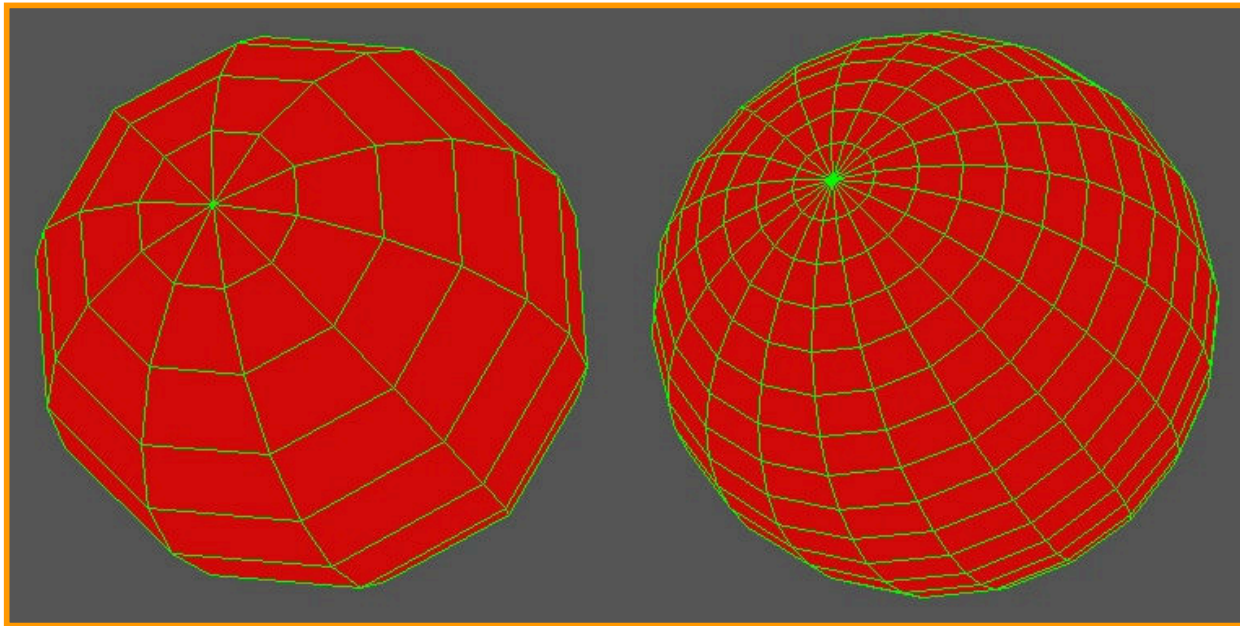
Adequada para representar objetos como cubos, paralelepípedos, prismas, cunhas, pirâmides, etc.





Representação Poliédrica de Objetos

O incremento no número de faces melhora a aproximação da representação poliédrica para superfícies curvas, porém aumenta o consumo de processamento e memória.



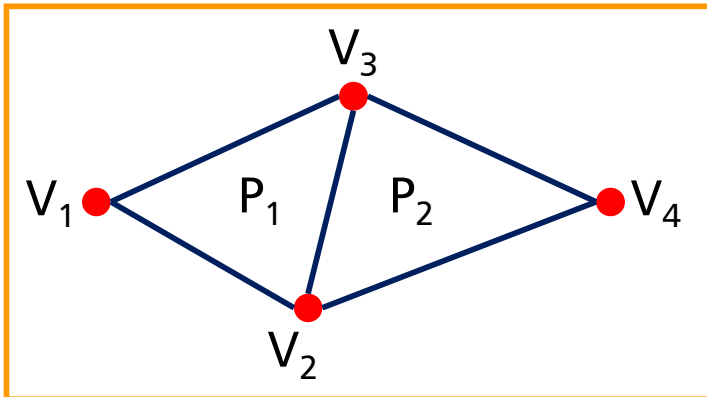
10 vértices = 100 faces

20 vértices = 400 faces

Armazenamento de Polígonos

Vértices Explícitos

Cada polígono é representado por uma lista de coordenadas de vértices.



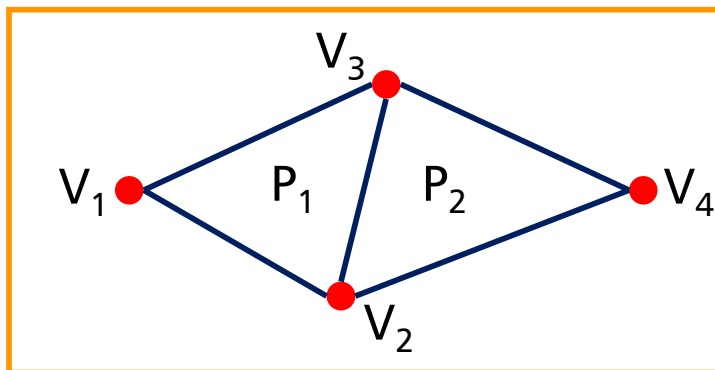
$$P_1 = \{V_1, V_2, V_3\}$$
$$P_2 = \{V_2, V_4, V_3\}$$

- Adequada para um único polígono;
- Em poliedros duplica os vértices compartilhados;
- Não explicita vértices e arestas compartilhadas.

Armazenamento de Polígonos

Ponteiros para Listas de Vértices

Os vértices são armazenados em uma lista de vértices. O polígono é representado por uma lista de ponteiros para os vértices.

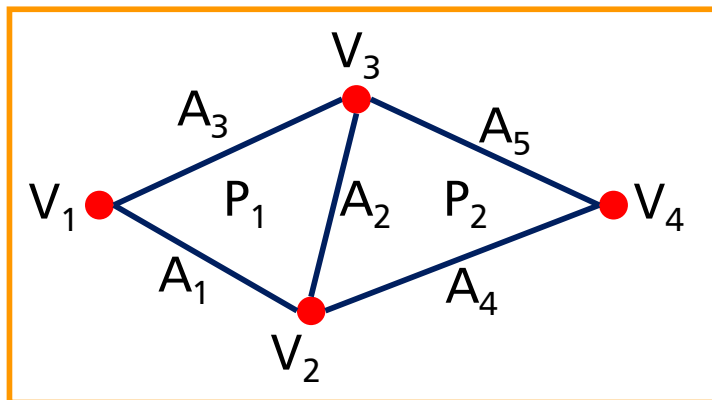

$$V = \{V_1, V_2, V_3, V_4\}$$
$$P_1 = \{\&V_1, \&V_2, \&V_3\}$$
$$P_2 = \{\&V_2, \&V_4, \&V_3\}$$

- Utiliza menos memória;
- Permite alterar os vértices mantendo a conectividade;
- É difícil identificar quais polígonos compartilham arestas;
- Arestas compartilhadas são desenhadas duas vezes.

Armazenamento de Polígonos

Arestas Explícitas

Há duas listas: de vértices e de arestas. Cada aresta tem ponteiros para dois vértices e cada polígono é formado por uma lista de ponteiros para arestas.

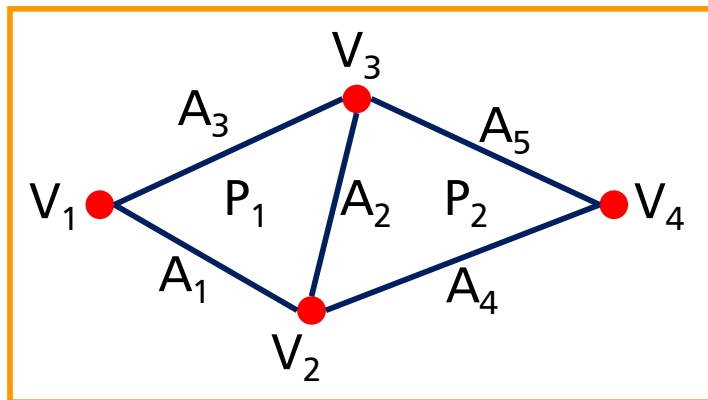


$V = \{V_1, V_2, V_3, V_4\}$
 $A = \{A_1 = (\&V_1, \&V_2), A_2 = (\&V_2, \&V_3), A_3 = (\&V_3, \&V_1), A_4 = (\&V_2, \&V_4), A_5 = (\&V_4, \&V_3)\}$
 $P_1 = \{\&A_1, \&A_2, \&A_3\}$
 $P_2 = \{\&A_2, \&A_4, \&A_5\}$

- As arestas são desenhadas percorrendo-se uma única lista;
- É fácil desenhar um único polígono;
- Ainda é difícil identificar quais faces compartilham um determinado vértice (algoritmos de sombreamento).

Armazenamento de Polígonos

Alguns processos são facilitados adicionando-se ponteiros adicionais às listas. Na estrutura abaixo é possível identificar quais polígonos compartilham uma determinada aresta.



$V = \{V_1, V_2, V_3, V_4\}$

$A = \{A_1 = (\&V_1, \&V_2, \&P_1, \phi), A_2 = (\&V_2, \&V_3, \&P_1, \&P_2), A_3 = (\&V_3, \&V_1, \&P_1, \phi), A_4 = (\&V_2, \&V_4, \&P_2, \phi), A_5 = (\&V_4, \&V_3, \&P_2, \phi)\}$

$P_1 = \{\&A_1, \&A_2, \&A_3\}$

$P_2 = \{\&A_2, \&A_4, \&A_5\}$

Armazenamento de Polígonos

Winged-Edge

Apresentada por Bruce G. Baumgart no artigo "*Winged-edge polyhedron representation for computer vision*", em 1975.



- Concentra as informações na lista de arestas, em uma estrutura de tamanho fixo;
- Permite verificar, em tempo constante, as relações de adjacência entre vértices, arestas e faces:
 - Quais vértices, arestas ou faces são adjacentes a cada face, aresta ou vértice.
- Respeito à fórmula de Euler para poliedros convexos: $F - A + V = 2$.

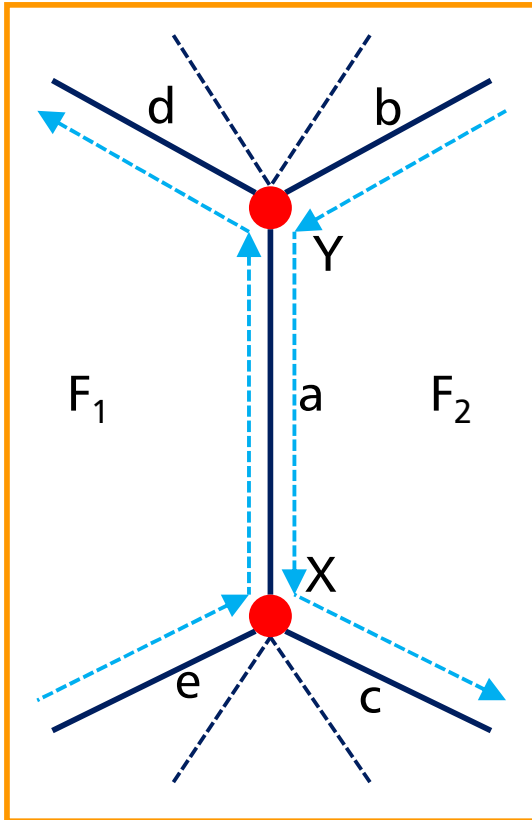


Estrutura de Dados *Winged-Edge*

A estrutura de dados *Winged-Edge* é composta por 3 listas:

- **Lista de vértices:** cada vértice v mantém suas coordenadas (x, y, z) e um ponteiro para uma aresta qualquer que incide em v ;
- **Lista de faces:** cada face f mantém um ponteiro para uma aresta qualquer da fronteira de f ;
- **Arestas:** Cada aresta possui 8 ponteiros:
 - Dois ponteiros para os vértices da aresta, cuja ordem indica a orientação da aresta;
 - Dois ponteiros para as faces que compartilham a aresta (face da esquerda e da direita);
 - Quatro ponteiros para as outras arestas conectadas.

Estrutura de Dados *Winged-Edge*



Lista de Arestas

Aresta	Vértices		Fases		Face Esq.		Face Dir.	
Nome	Ini	Fim	Esq	Dir	Pre	Suc	Pre	Suc
a	&X	&Y	&F ₁	&F ₂	&e	&d	&b	&c

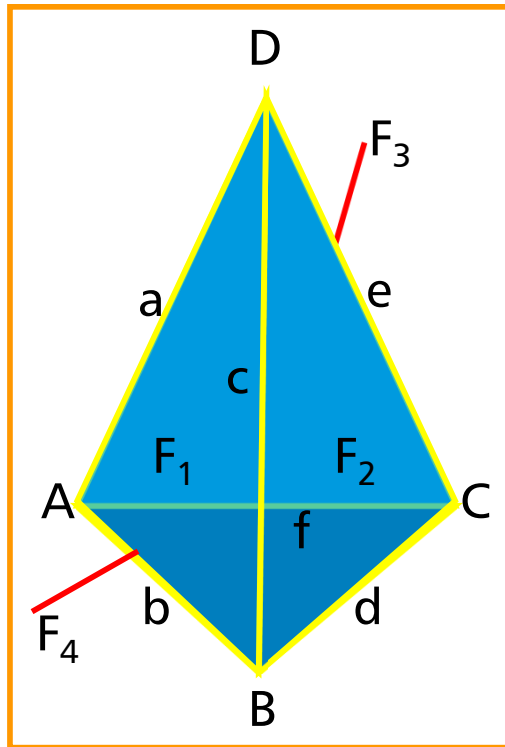
Lista de Vértices

Vértice	Coordenadas			Aresta Incidente
Nome	x	y	z	
X	X_x	X_y	X_z	&e
Y	Y_x	Y_y	Y_z	&b

Lista de Faces

Face	Aresta da Face
F_1	&d
F_2	&b

Winged-Edge – Exemplo



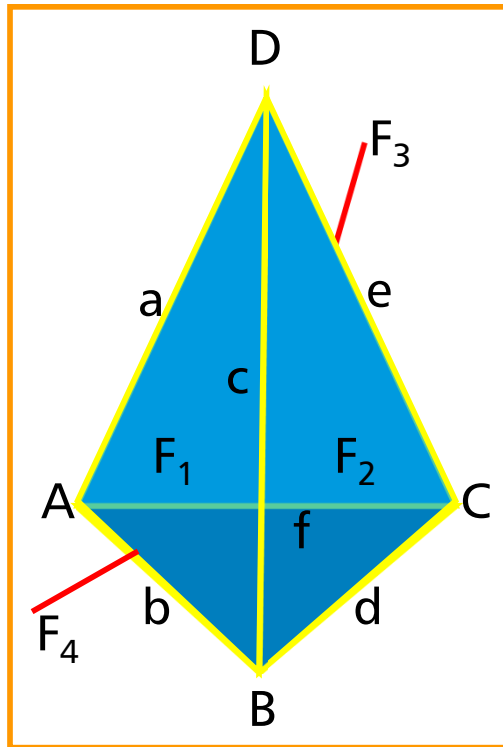
Lista de Vértices

Vértice	Coordenadas			Aresta Incidente
Nome	x	y	z	
A	A_x	A_y	A_z	&a
B	B_x	B_y	B_z	&c
C	C_x	C_y	C_z	&e
D	D_x	D_y	D_z	&a

Lista de Faces

Face	Aresta da Face
F_1	&a
F_2	&c
F_3	&e
F_4	&f

Winged-Edge – Exemplo



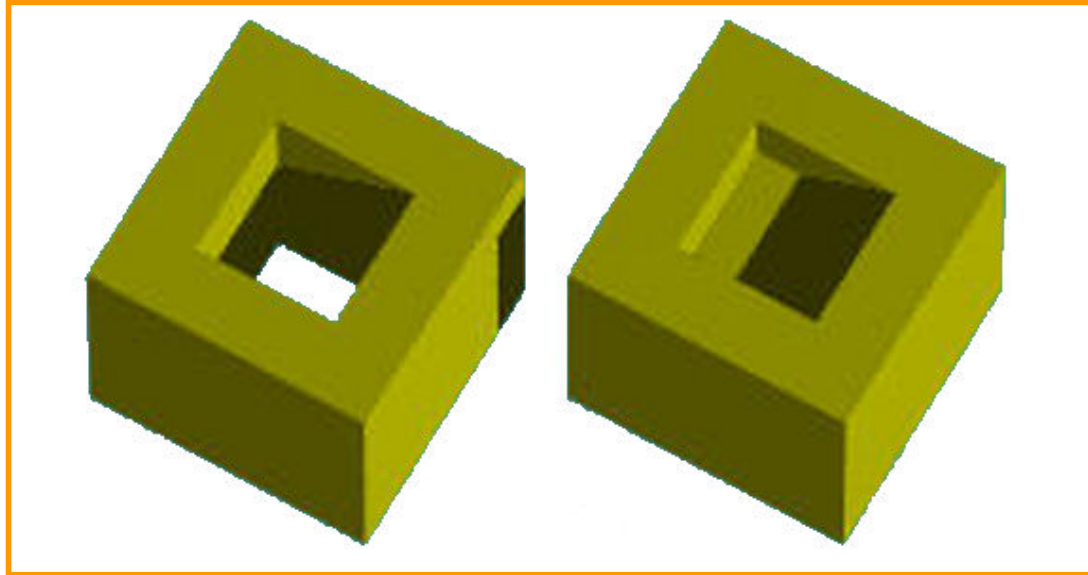
Lista de Arestas

Aresta	Vértices		Faces		Face Esq.		Face Dir.	
Nome	Ini	Fim	Esq	Dir	Pre	Suc	Pre	Suc
a	&A	&D	&F ₃	&F ₁	&f	&e	&c	&b
b	&A	&B	&F ₁	&F ₄	&a	&c	&d	&f
c	&B	&D	&F ₁	&F ₂	&b	&a	&e	&d
d	&B	&C	&F ₂	&F ₄	&c	&e	&f	&b
e	&C	&D	&F ₂	&F ₃	&d	&c	&a	&f
f	&C	&A	&F ₃	&F ₄	&e	&a	&b	&d



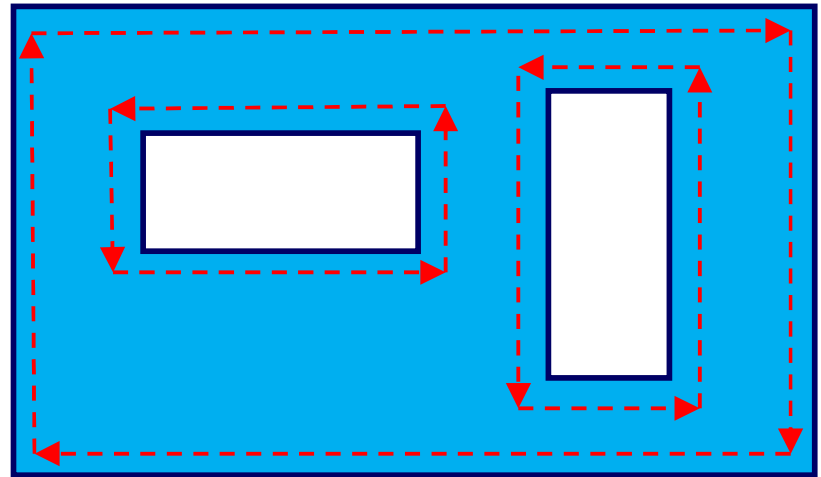
Winged-Edge – Polígonos com Buracos

E se um polígono possuir buracos internos ou chanfros? Como representá-los?

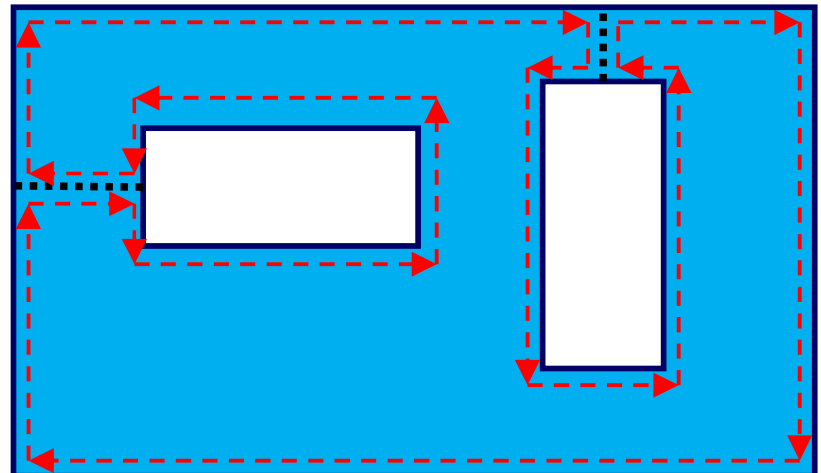


Winged-Edge – Polígonos com Buracos

Para uma face com buracos internos, a borda externa é percorrida em sentido horário enquanto os buracos internos são percorridos em sentido anti-horário.



Ou... Inserir arestas auxiliares ligando o buraco à borda externa. A face à direita e à esquerda das arestas auxiliares é a mesma, permitindo sua rápida identificação.





Representação de Sólidos por Aproximação

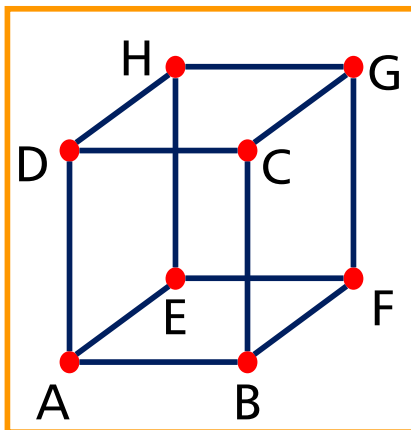
A representação de sólidos pode ser feita através da aproximação dos objetos utilizando diversas técnicas, tais como:

- *Wireframe*;
- Malhas de polígonos;
- B-Rep (*boundary representation*);
- CSG (*Constructive Solid Geometry*);
- *Sweep* (varredura);
- Enumeração da ocupação espacial ;
- *Octrees*;
- *BSP-Trees*.

Wireframe

A estrutura do objeto é representada por suas arestas, em uma estrutura aramada (*wireframe*).

São necessários dois conjuntos de dados:
Vértices (geometria) e Arestas (topologia).



Vértices

$A = (0, 0, 0)$; $B = (1, 0, 0)$; $C = (1, 1, 0)$; $D = (0, 1, 0)$;
 $E = (0, 0, 1)$; $F = (1, 0, 1)$; $G = (1, 1, 1)$; $H = (0, 1, 1)$

Arestas

AB, BC, CD, DA, EF, FG, GH, HE, AE, BF, CG, DH

Apresenta as seguintes limitações:

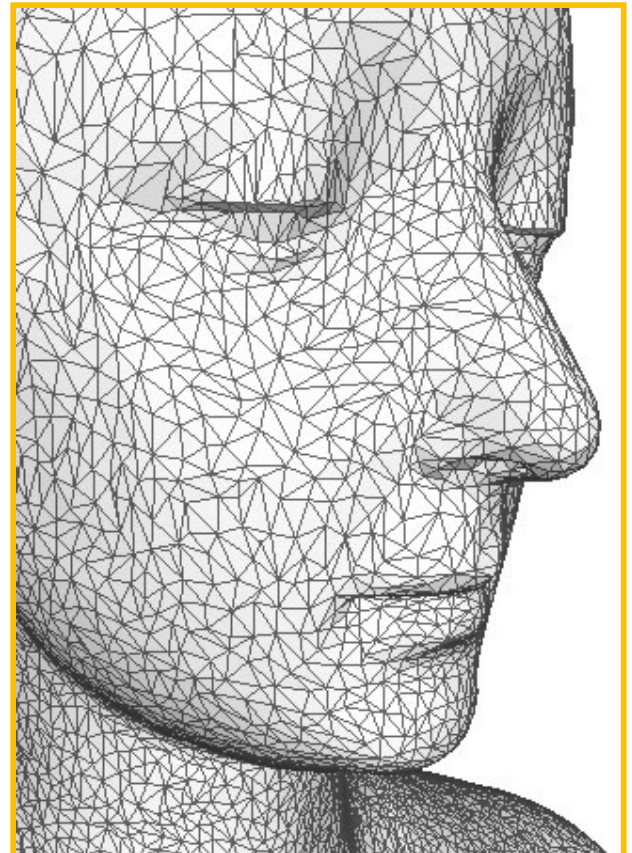
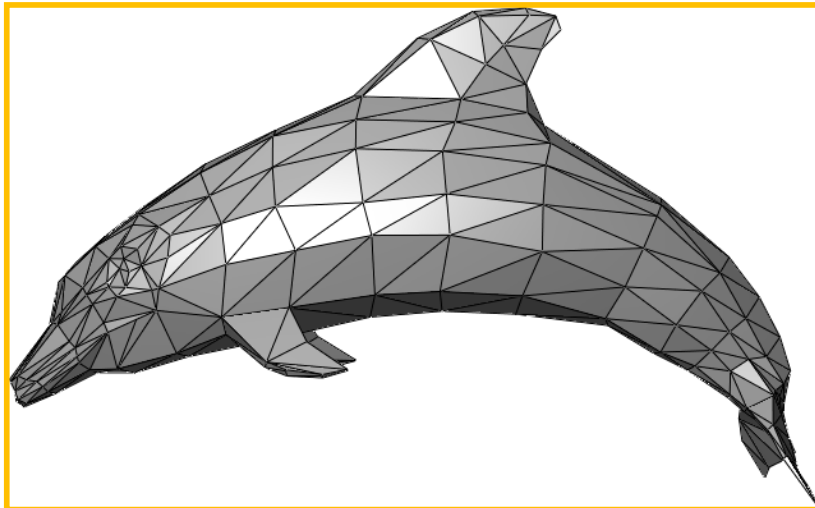
- Não é adequada para objetos vazados;
- Não armazena informações de superfície ou interior.



Malhas Poligonais (*Polygon Meshes*)

Representa uma superfície discretizada utilizando faces planas.

São necessários três conjuntos de dados: Vértices (geometria), Arestas e Faces (topologia).

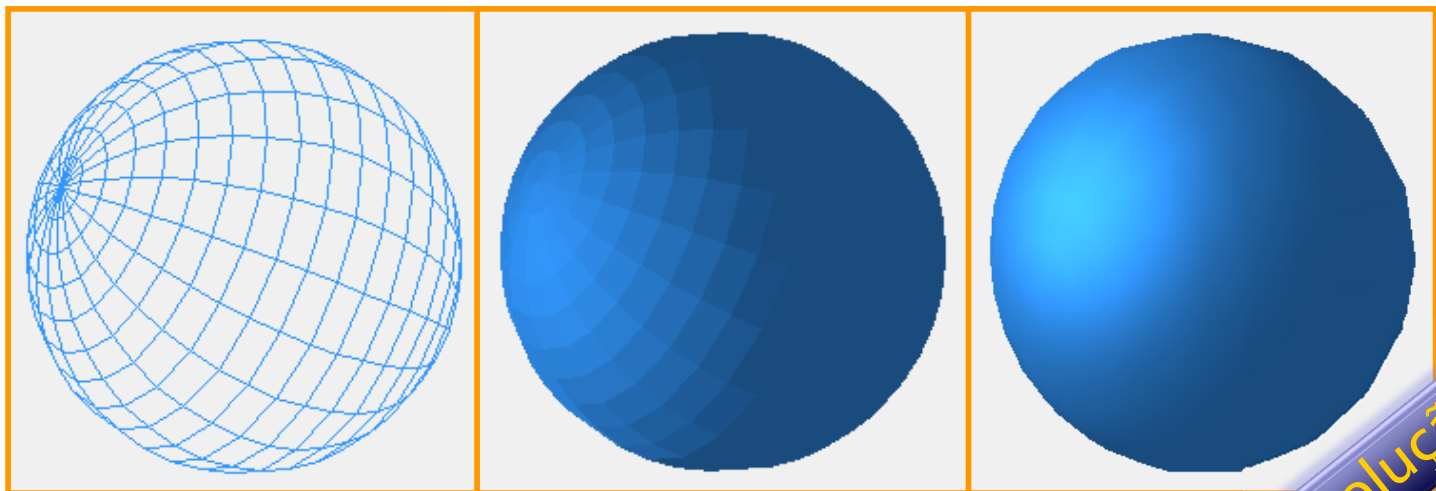


Malhas Poligonais (*Polygon Meshes*)

Estrutura de dados utilizada na representação:

- Lista de faces c/ Arestas Explícitas [vetores normais];
- *Winged-Edge*.

Apresenta problemas ao
representar objetos curvos.



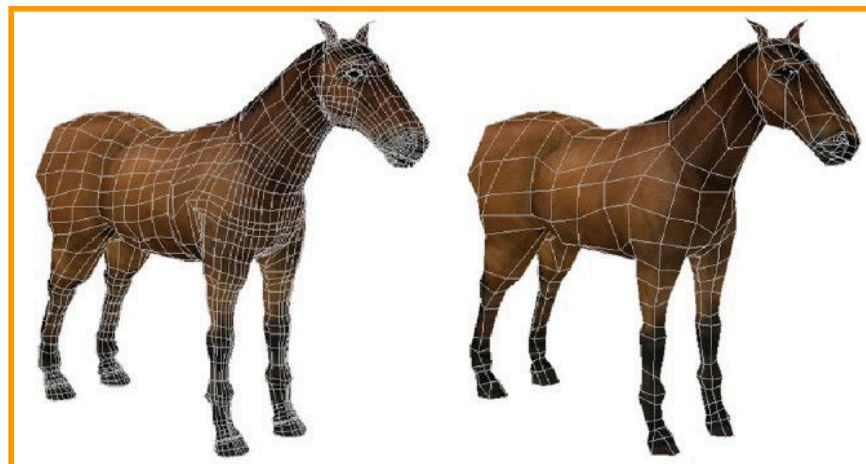
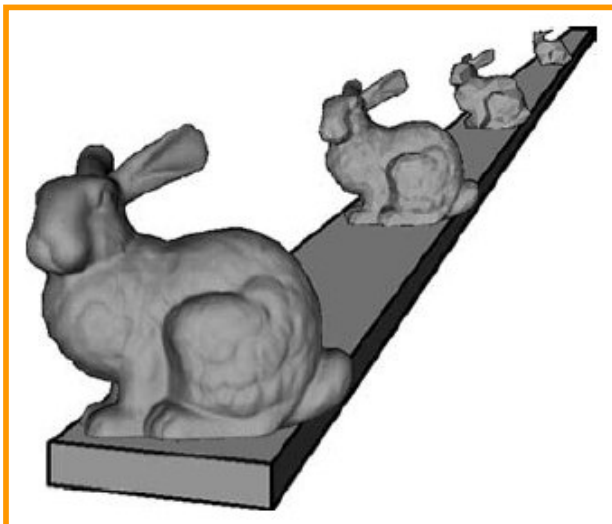
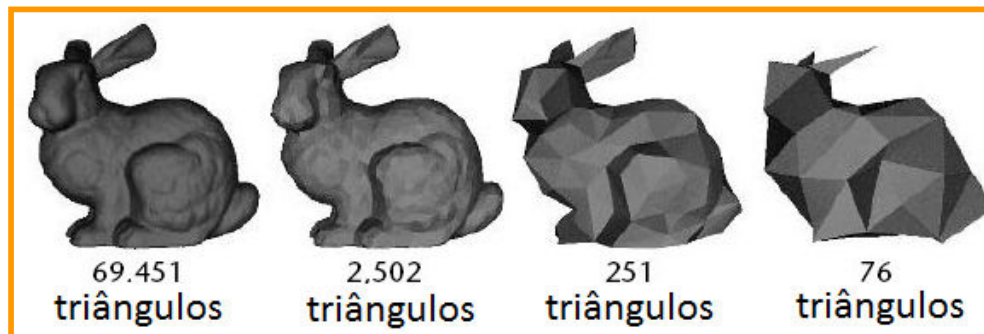
Solução

Uso de sombreamento para suavizar contornos.



Malhas Poligonais – *Level of Details* (LOD)

Processo de simplificação da malha poligonal em função da profundidade do objeto na cena.





Características das Malhas Poligonais

São flexíveis. Podem ser utilizados para representar uma ampla gama de objetos.

Quanto mais detalhada a malha, melhor a representação dos objetos e maior o consumo de memória.

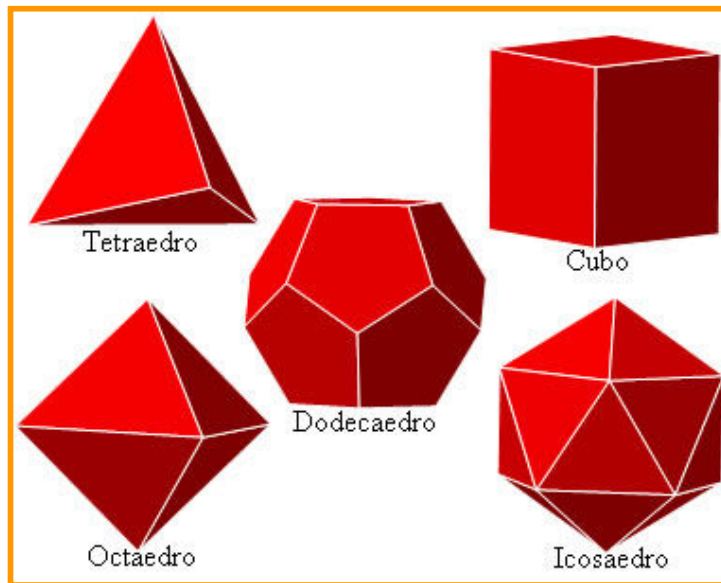
Apresentam limitações:

- Superfície não é suave;
- Não armazena informações sobre o interior do objeto, nem assegura que o objeto modelado é um sólido.

Uso de malha triangular uniformiza os algoritmos utilizados no processo de síntese de imagens.

B-Rep (*Boundary Representation*)

Técnica de representação adequada para poliedros convexos regulares.

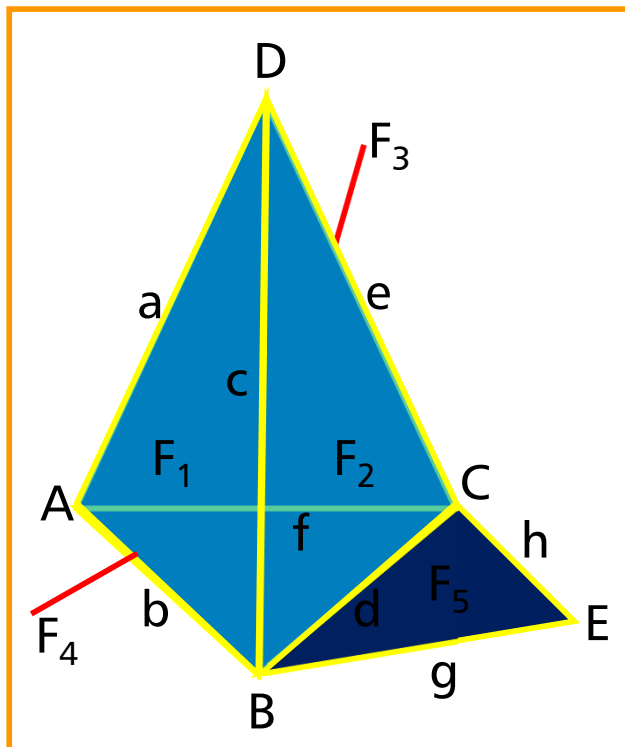


Os poliedros são descritos por suas faces, arestas e vértices.

As faces separam o interior do exterior do objeto e são representadas por polígonos planos, geralmente triângulos.

B-Rep (*Boundary Representation*)

Utilizar uma estrutura de dados baseada em faces não garante que o objeto representado atenda aos requisitos da B-Rep.



Fórmula de Euler

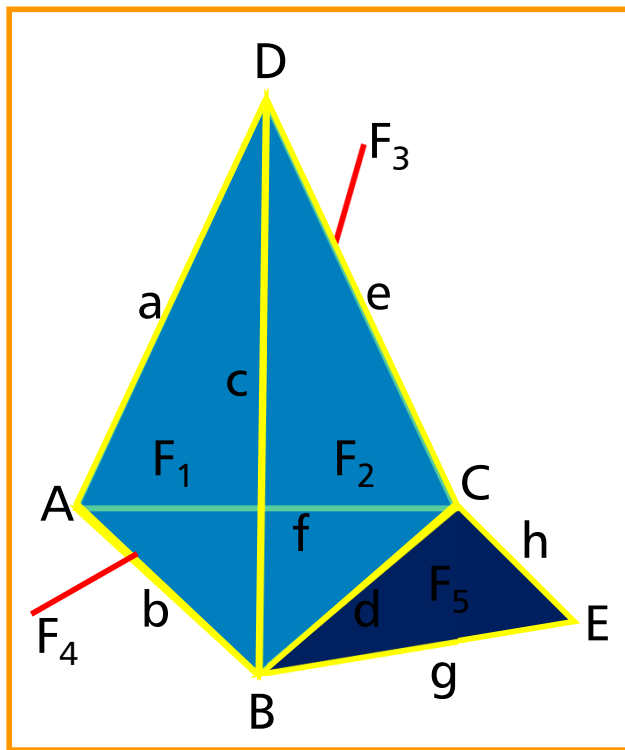
$$F - A + V = 2.$$

$$F = 5; A = 8; V = 5$$
$$5 - 8 + 5 = 2$$

Atende a Fórmula de Euler,
mas não é um poliedro
convexo fechado.

B-Rep (*Boundary Representation*)

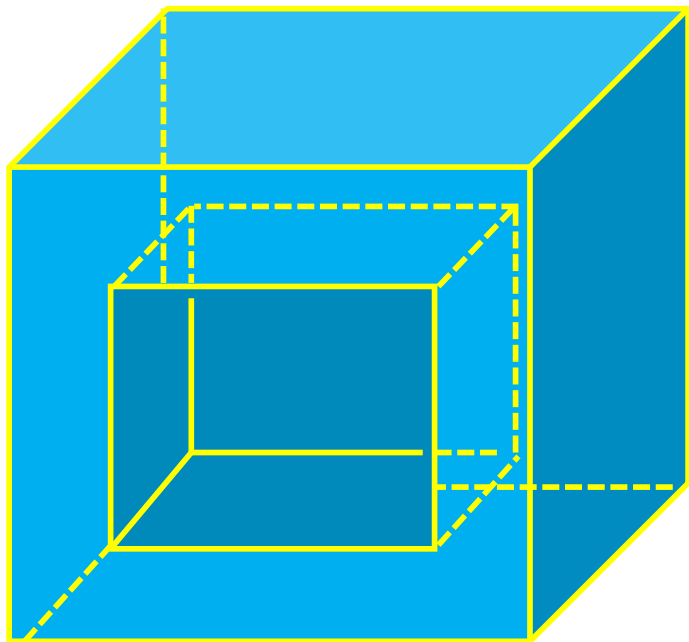
Para que um objeto atenda aos requisitos da B-Rep são necessárias restrições adicionais.



- Cada aresta deve ser compartilhada por duas faces;
- Cada aresta deve conectar dois vértices;
- Cada vértice deve ser compartilhado por 3 arestas, no mínimo.

B-Rep (*Boundary Representation*)

E se o objeto possui furos ou reentrâncias?



Fórmula de Euler
generalizada

$$F - A + V = 2 + H - 2G$$

H = Buracos nas faces;
G = Buracos no objeto.

$$F = 11; A = 24; V = 16$$

$$H = 1; G = 0$$

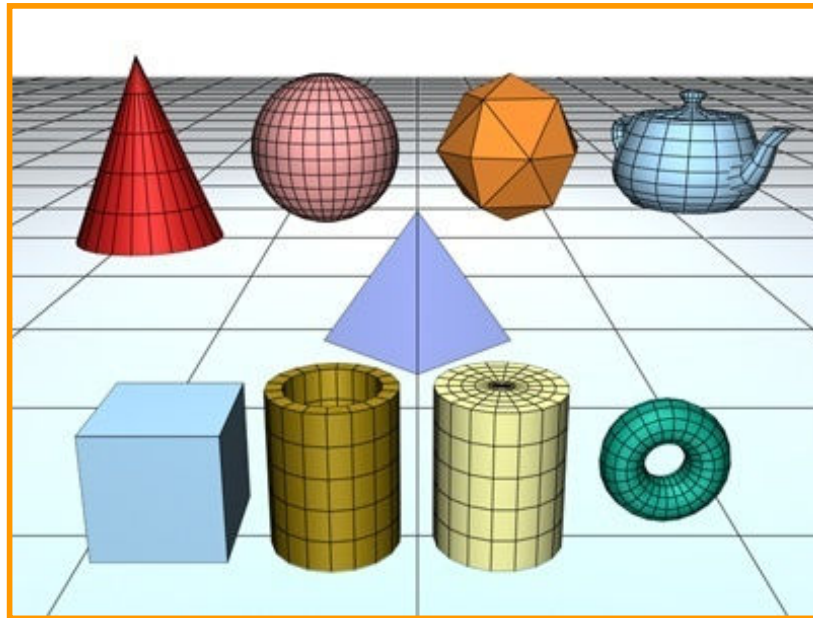
$$11 - 24 + 16 = 2 + 1 - 2 \cdot 0$$

$$3 = 3$$



CSG (*Constructive Solid Geometry*)

Utiliza um conjunto de primitivas geométricas simples: prismas, cones, cilindros, esferas, etc.



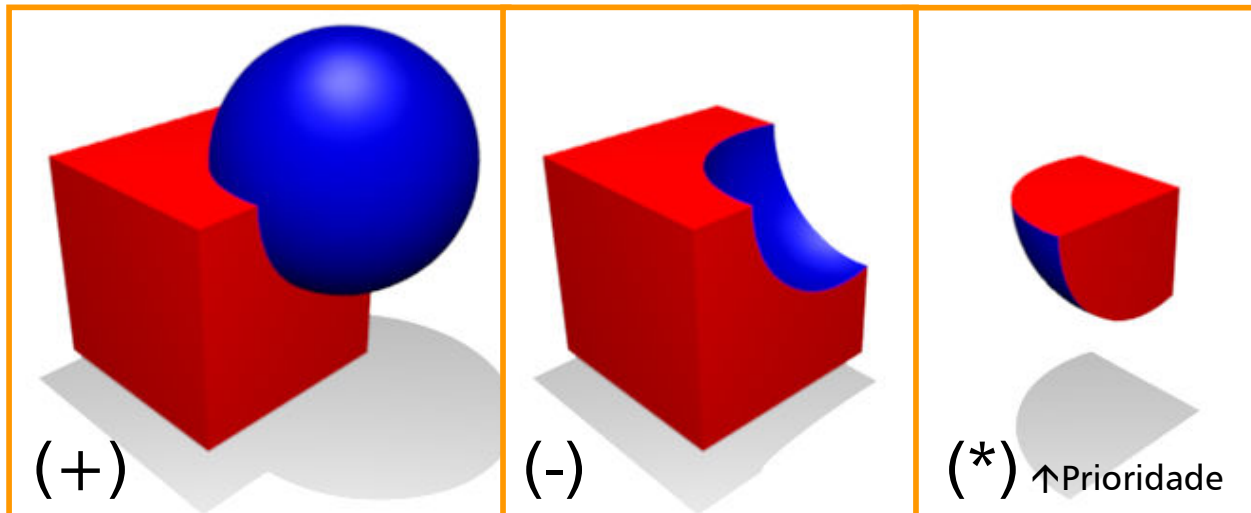
Objetos mais complexos são criados através da combinação e do posicionamento das primitivas usando operações booleanas.



CSG (*Constructive Solid Geometry*)

A CSG tem por base a teoria dos conjuntos.

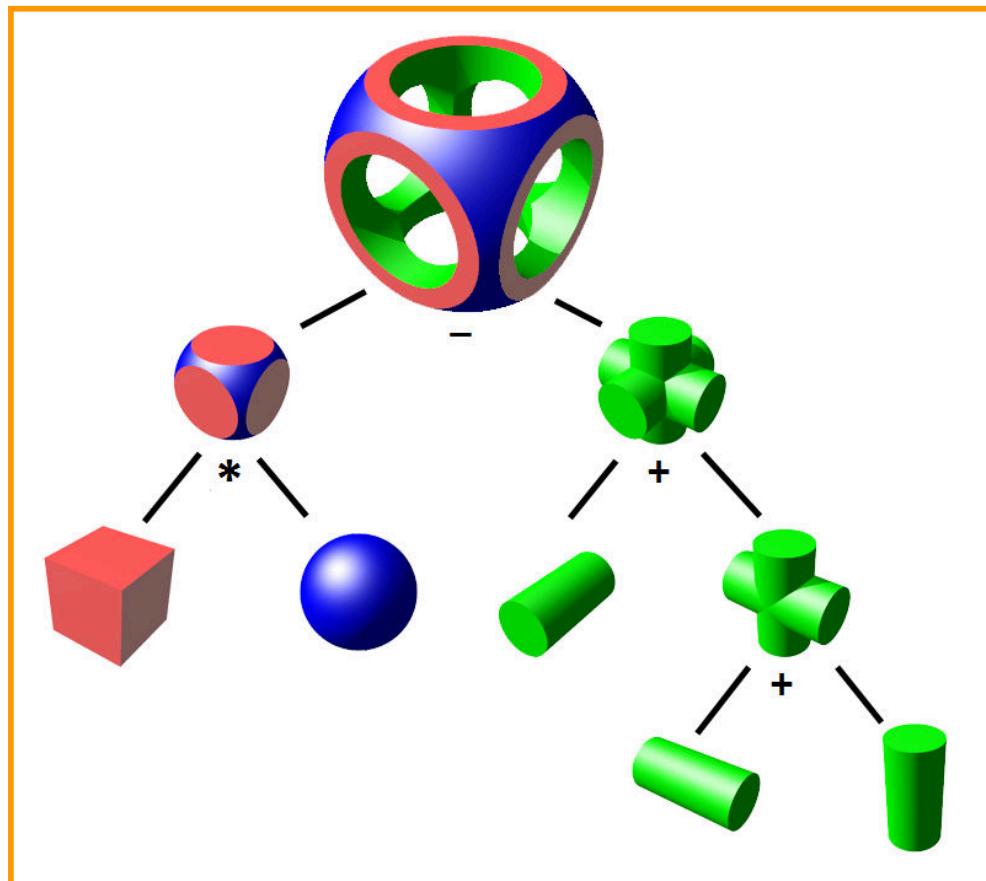
As primitivas são transformados por movimentos rígidos (translação, rotação e escala) e combinadas pelos operadores de União (+), Diferença (-) e Interseção (*).





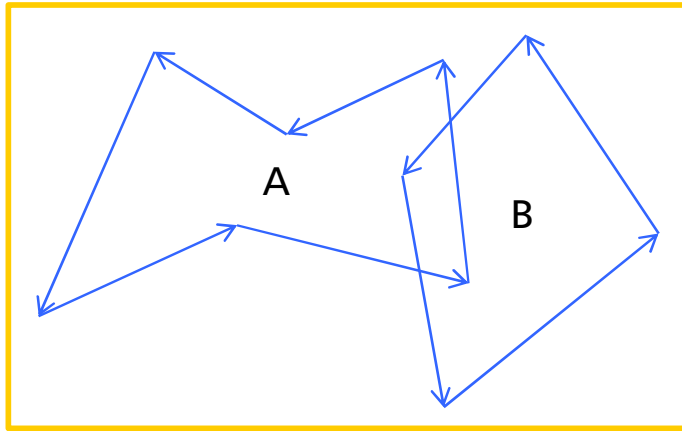
CSG (*Constructive Solid Geometry*)

Os objetos criados com CSG são representados através de árvores booleanas.

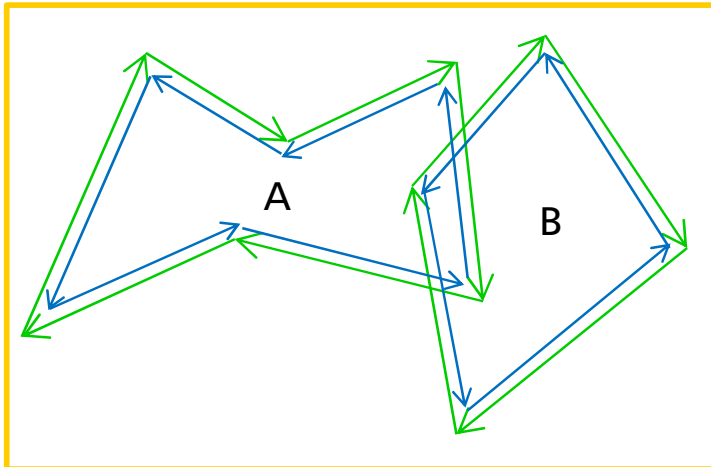




Conversão CSG \rightarrow B-Rep



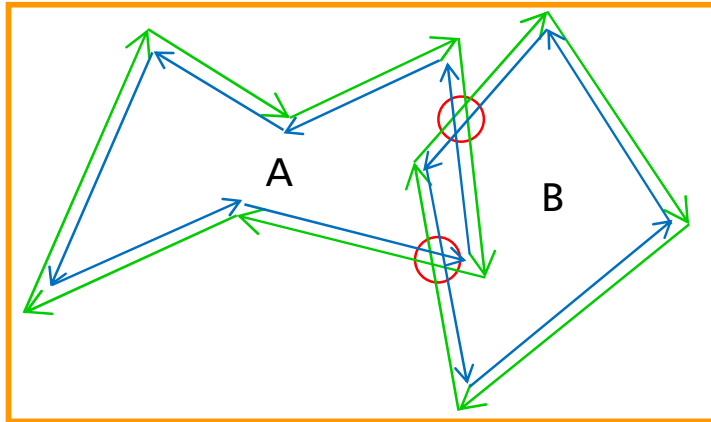
Percorrer a borda interna do polígono no sentido anti-horário



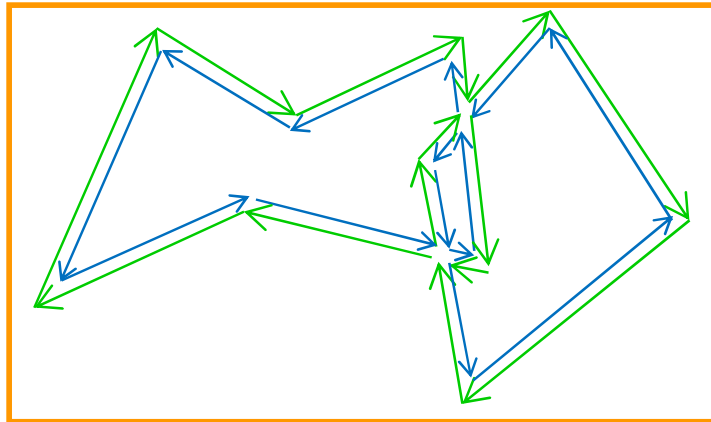
Percorrer a borda externa do polígono no sentido horário



Conversão CSG \rightarrow B-Rep



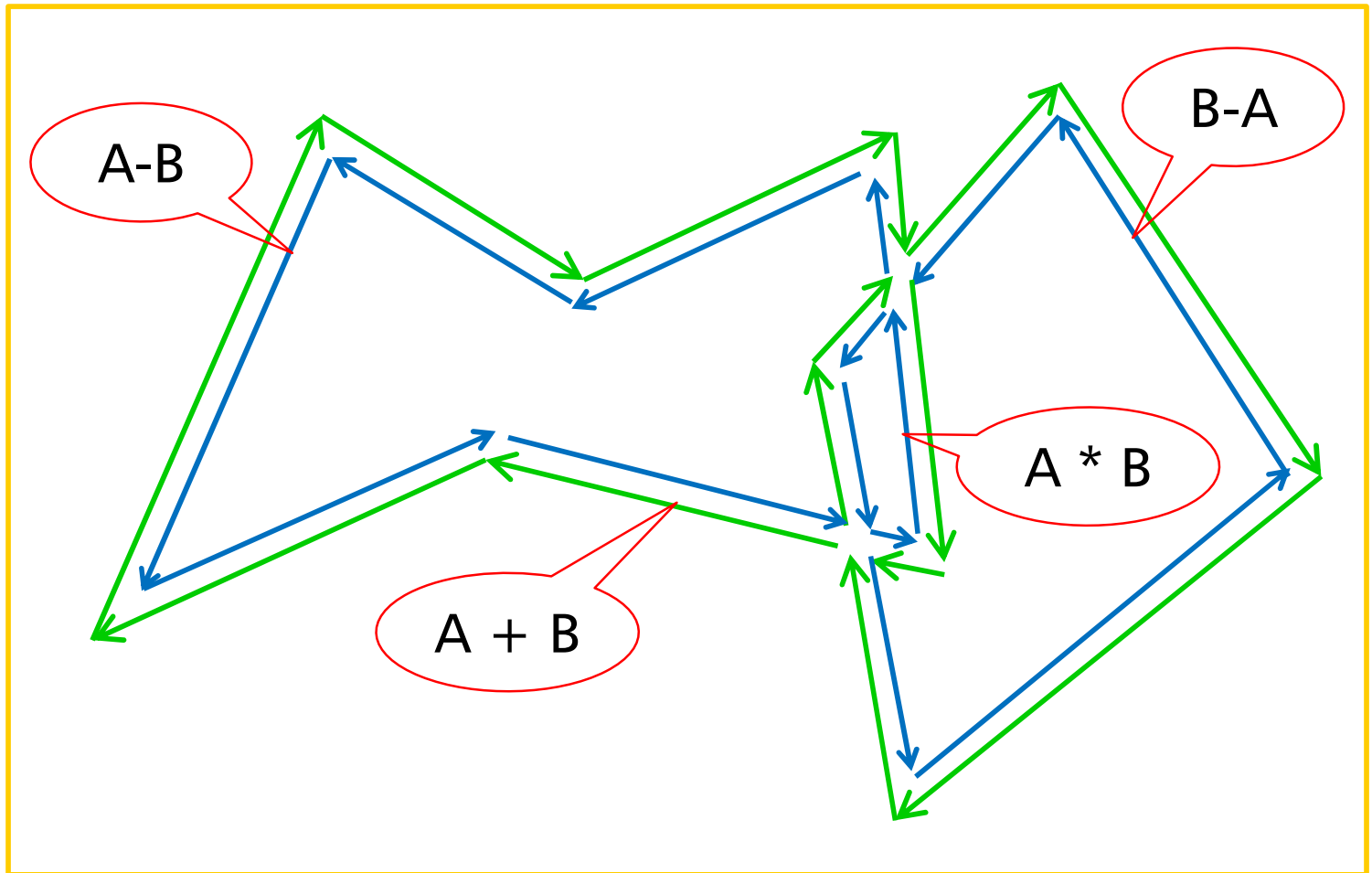
Calcular os pontos de interseção entre as arestas



Costurar adequadamente as regiões usando as interseções calculadas.



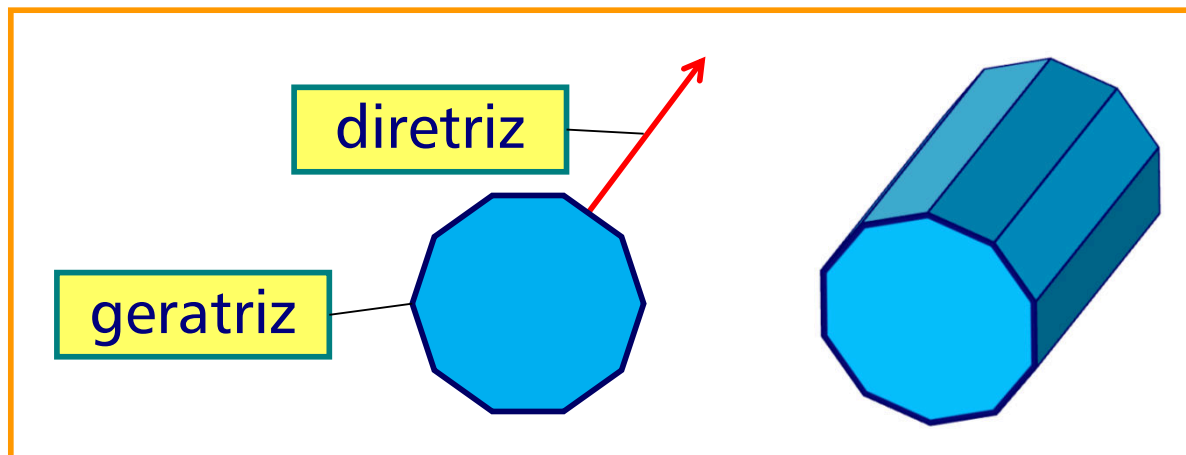
Conversão CSG \rightarrow B-Rep



Sweep

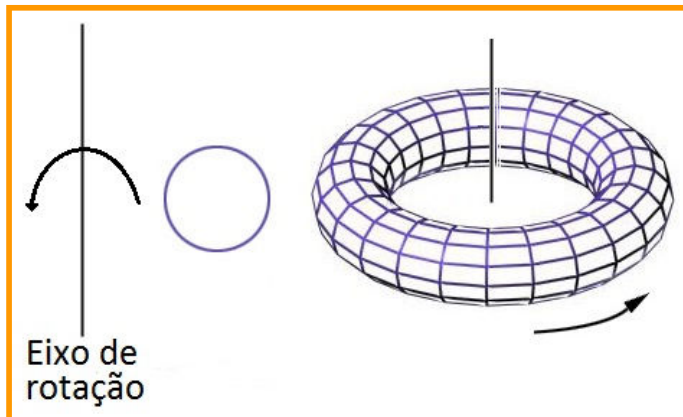
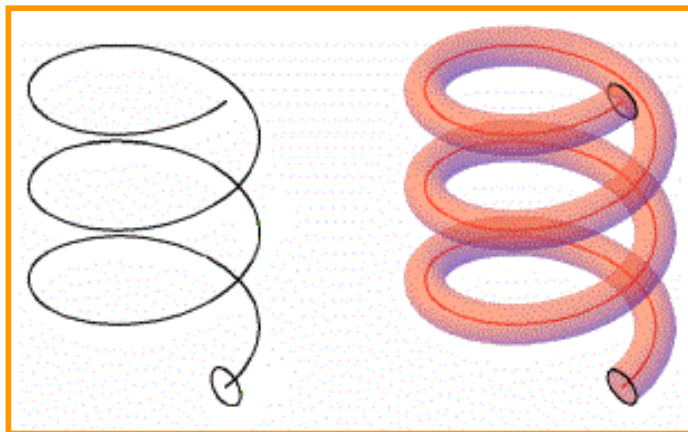
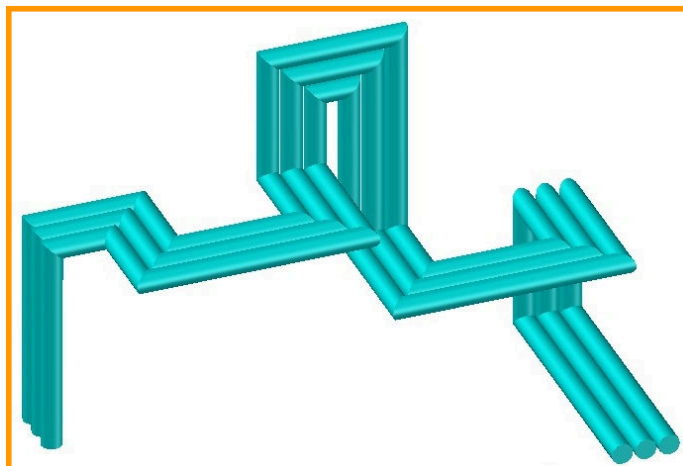
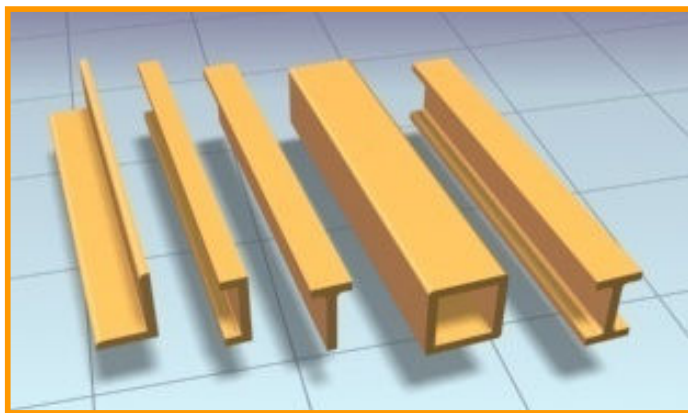
Geração de objetos 3D a partir do deslizamento de uma superfície poligonal ao longo de uma trajetória.

A superfície poligonal (uma seção plana) é chamada de **geratriz** enquanto a trajetória é chamada de **diretriz**.



Sweep

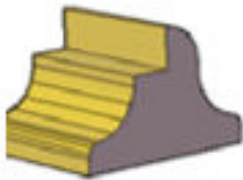
As diretrizes podem ser lineares, polylines, curvas ou de revolução.





Sweep

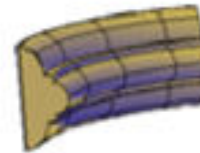
Os processos para gerar os objetos são:
Extrusão, Revolução, Deslizamento e Loft



Extrusão



Revolução



Deslizamento



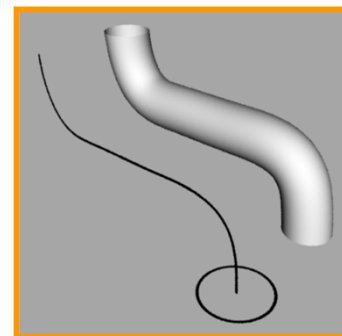
Loft



Extrusão



Revolução



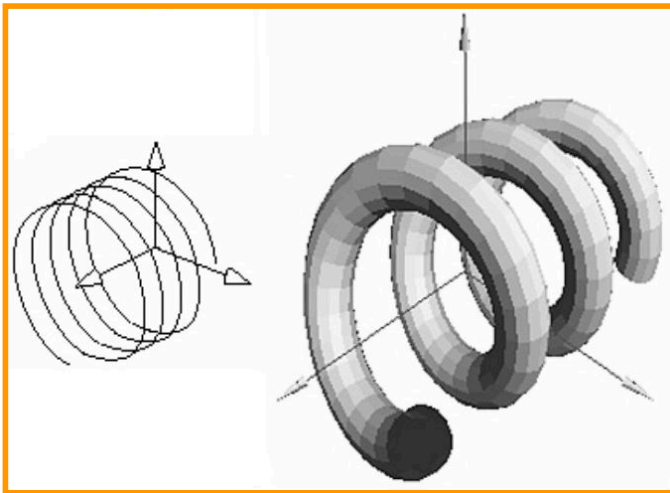
Deslizamento



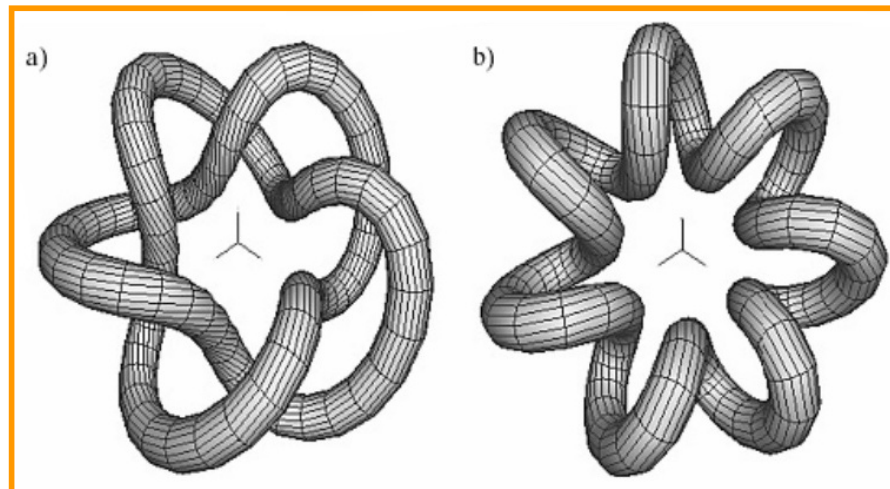
Loft



Geração de Malhas por Deslizamento



$C(t) = (\cos(t), \sin(t), b.t)$,
onde b é uma constante.



$C(t) = ((a+b.\cos(qt))\cos(pt),$
 $(a+b.\cos(qt))\sin(pt),$
 $c.\sin(qt)))$

a, b, p e q são constantes
escolhidas

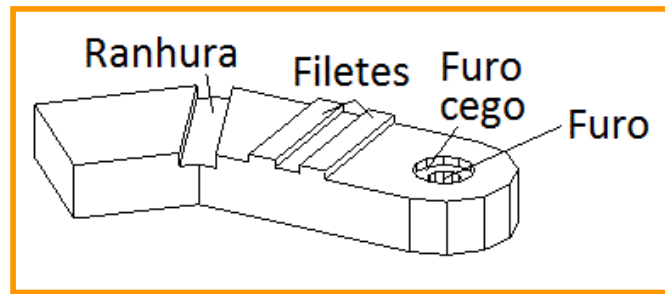
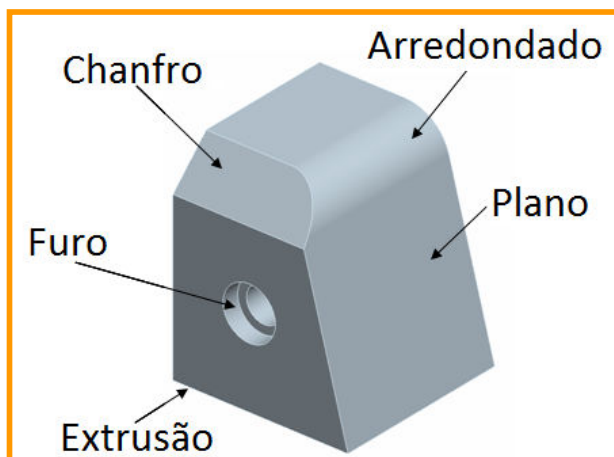
a) $p = 2, q = 5$

b) $p = 1, q = 7$

Sweep

A modelagem Sweep pode combinar aspectos da CSG e da B-Rep.

Objetos são modelados pela adição ou subtração de características a um objeto base. As características são operações de manufatura como furos, nervuras, filetes, chanfros, ranhuras, etc.

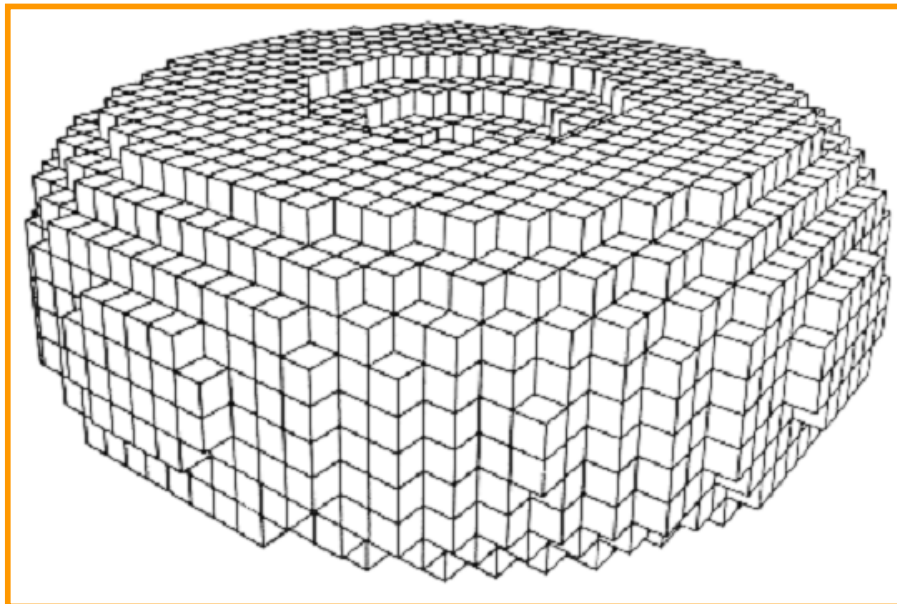




Enumeração da Ocupação Espacial

O objeto é decomposto em células idênticas (*voxels*) arranjadas num *grid* regular.

Em uma lista de células indicamos quais estão ocupadas (presença) ou não (ausência).

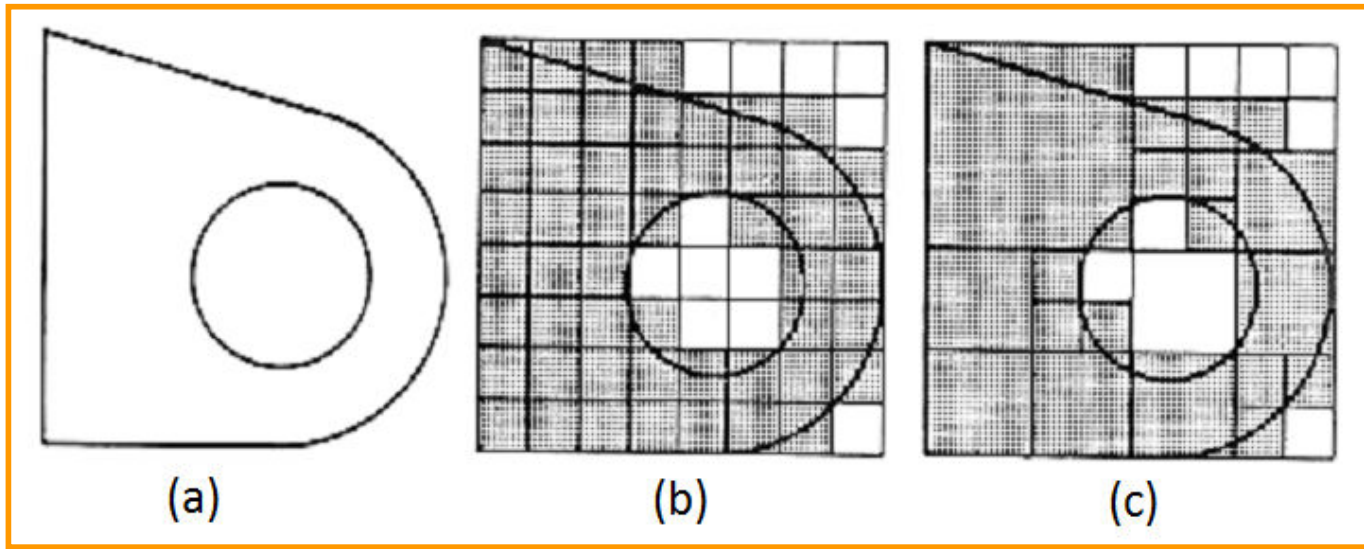




Octrees

Octrees são uma variante hierárquica da Enumeração da Ocupação Espacial.

Derivam das *Quadrees*, uma técnica empregada em armazenamento de imagens.

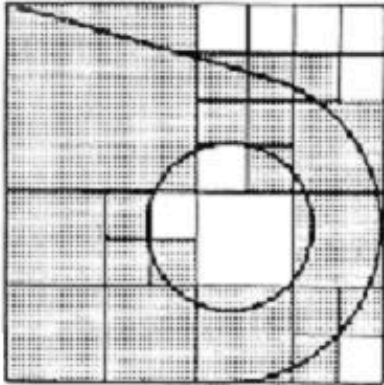


(a) Imagem Original (b) Enumeração Espacial (c) *Quadtree*.



Representação em Memória – *Quadrees*

Objeto



Numeração dos Quadrantes

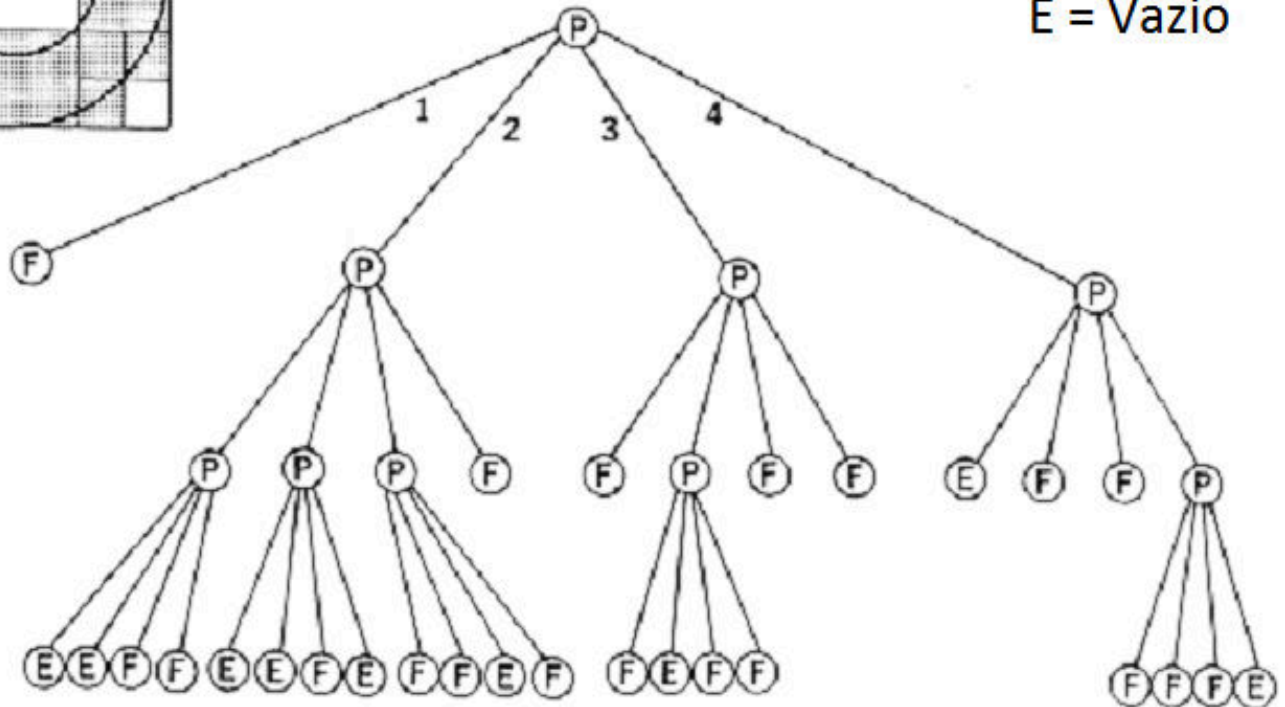
1	2
3	4

Rótulos

P = Parcial

F = Cheio

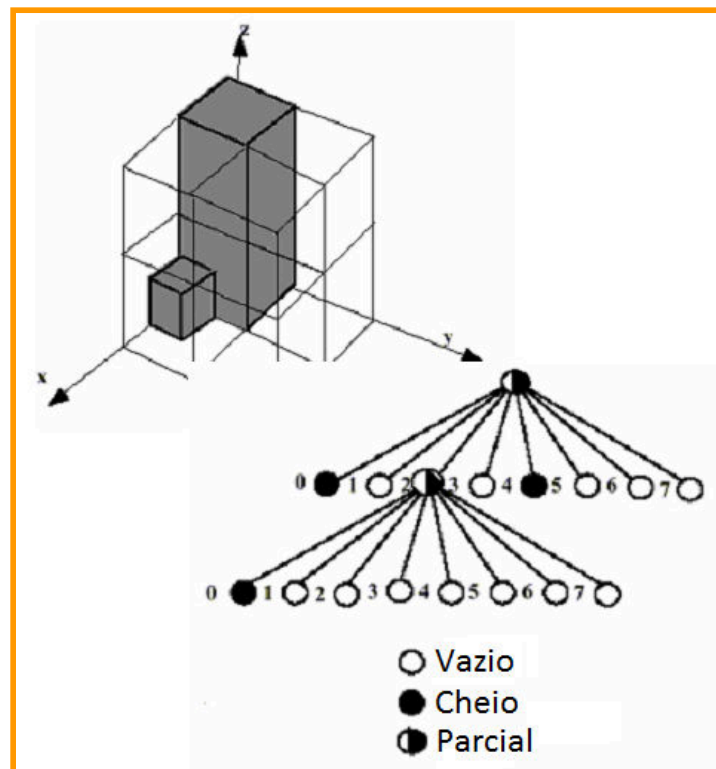
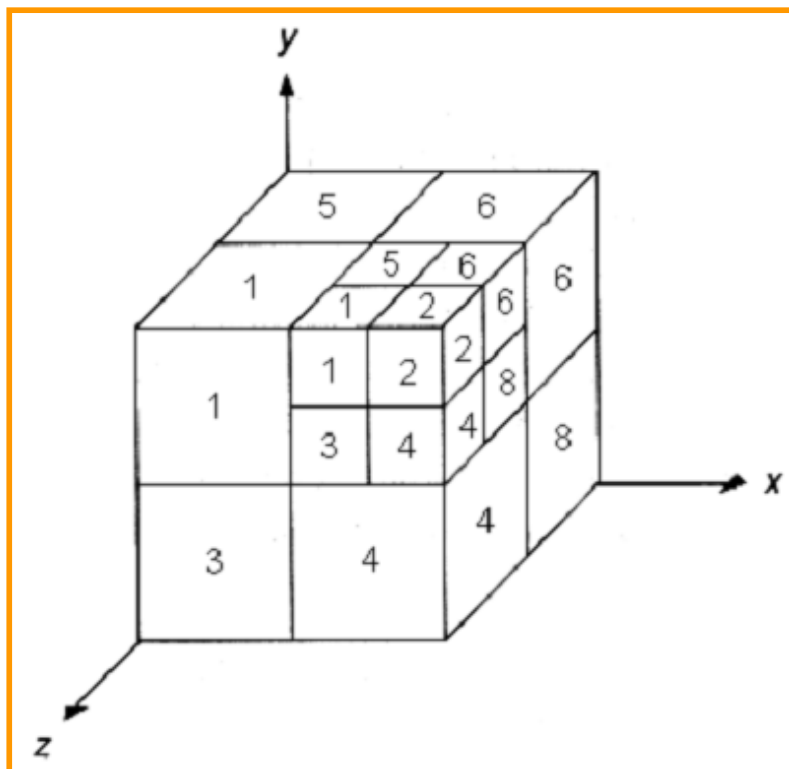
E = Vazio





Representação em Memória – *Octrees*

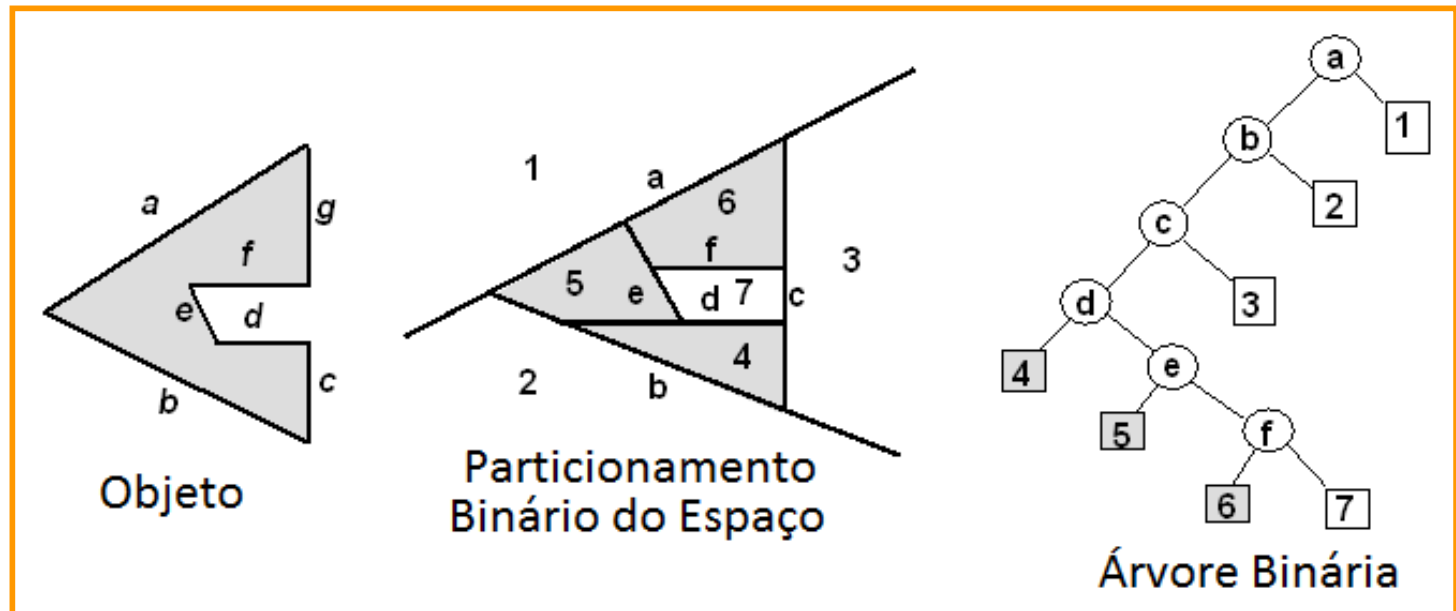
A representação é similar às *Quadtrees*. As três dimensões são recursivamente divididas em octantes numerados de 1 a 8.





BSP-Trees

Realiza a divisão recursiva do espaço em regiões convexas utilizando hiperplanos. O processo de divisão é representado em uma árvore binária chamada *BSP-Tree*.





Comparação das Representações

Requicha (1980) propôs uma lista de propriedades desejáveis num esquema para representação de sólidos.

Domínio

Unicidade

Precisão

Validade

Eficiência



Comparação das Representações

DOMÍNIO

O domínio da representação deve ser grande o suficiente para permitir a representação de um conjunto útil de objetos.

- Sweep é limitada;
- Enumeração Espacial, *Octrees* e *BSP-Trees* permitem representar qualquer sólido, mas apenas aproximações;
- Incluir arestas e superfícies curvas aumenta o domínio de representação da B-Rep.



Comparação das Representações

UNICIDADE

Uma representação é única quando codifica o sólido de apenas uma maneira.

- Somente a Enumeração Espacial e *Octrees* são capazes de representar sólidos de modo único.

PRECISÃO

Permite representar um objeto sem aproximações.

- Enumeração Espacial e B-Rep permitem representar aproximações dos objetos;
- As outras técnicas são mais precisas, principalmente quando aumentamos a resolução.



Comparação das Representações

VALIDADE

As representações devem gerar objetos válidos.

- B-Rep é crítica: pode acontecer casos de faces, arestas e vértices inconsistentes, bem como a interseção entre faces e arestas;
- *Octrees* e CSG permitem facilmente verificar a validade do objeto representado;
- Enumeração Espacial não necessita verificação de validade.



Comparação das Representações

EFICIÊNCIA

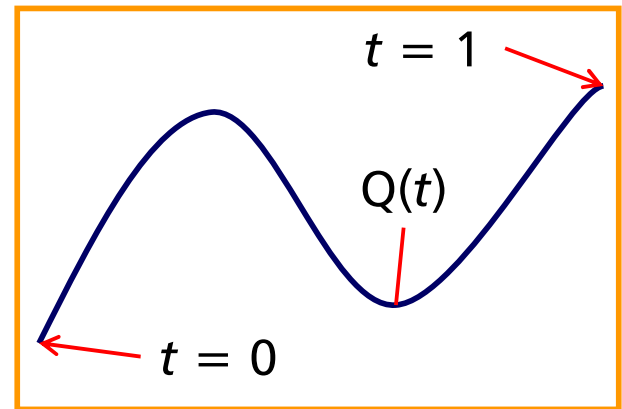
Está diretamente relacionada com o processamento realizado na interpretação dos objetos modelados.

- Técnicas que precisam processar a representação do sólido perdem em eficiência;
- *CSG*, *Octrees* e *BSP-Trees* não são eficientes;
- B-Rep e Enumeração Espacial não necessitam processar informações, portanto são consideradas mais eficientes;
- É relativo! Saber se um ponto é interno a um sólido é mais fácil em *CSG* do que em B-Rep.

Modelagem de Curvas*

Uma curva $Q(t) = [x(t) \ y(t) \ z(t)]$, com $0 \leq t \leq 1$ pode ser escrita como:

$$\begin{aligned}x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x \\y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y \\z(t) &= a_z t^3 + b_z t^2 + c_z t + d_z\end{aligned}$$



Qualquer ponto (x, y, z) ao longo da curva Q é descrito pelo conjunto de polinômios de 3º grau e do parâmetro t .

*Com informações de Esperança, C. & Cavalcanti, P. R. **Introdução à Computação Gráfica –Curvas**. Disponível em: <http://www.lcg.ufrj.br/Cursos/COS-751/curvas-ppt>. Acesso em: 31/03/2014.

Modelagem de Curvas

$$Q(t) = \begin{cases} x(t) = a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) = a_y t^3 + b_y t^2 + c_y t + d_y \\ z(t) = a_z t^3 + b_z t^2 + c_z t + d_z \end{cases}$$

Podemos reescrever $Q(t)$, como:

$$Q(t) = T \cdot C$$

$$T = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \text{ e } C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}$$

Modelagem de Curvas

Podemos reescrever C , como:

$$C = M \cdot G$$

$$C = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \cdot \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}$$

M = Matriz base

Define a mistura dos elementos da Geometria.

G = Vetor de Geometria

Elementos da curva considerados na sua modelagem.

Modelagem de Curvas

Finalmente, escrevemos $Q(t)$ como:

$$Q(t) = T \cdot M \cdot G$$

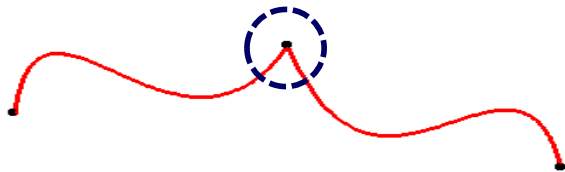
$$Q(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \cdot \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}$$

$$x(t) = (t^3 m_{11} + t^2 m_{21} + t m_{31} + m_{41}) G_1 x + (t^3 m_{12} + t^2 m_{22} + t m_{32} + m_{42}) G_2 x + (t^3 m_{13} + t^2 m_{23} + t m_{33} + m_{43}) G_3 x + (t^3 m_{14} + t^2 m_{24} + t m_{34} + m_{44}) G_4 x$$

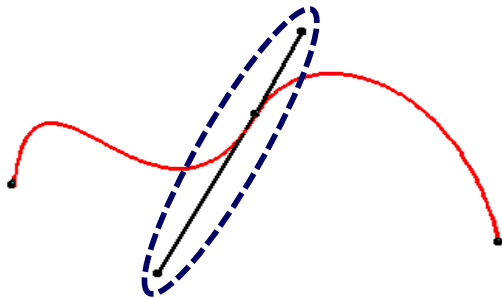
Expressões similares podem ser construídas para $y(t)$ e $z(t)$.

Continuidade das Curvas

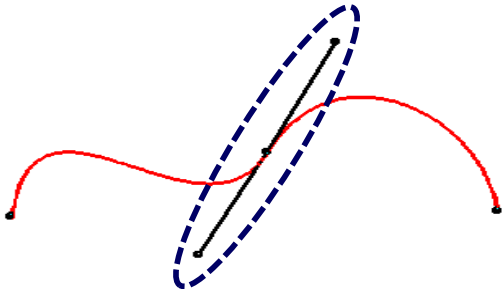
Ao desenhar curvas contínuas desejamos que as transições entre elas sejam suaves. A suavidade está associada com a continuidade algébrica das curvas.



C^0 : As duas curvas apresentam um ponto de junção.



C^1 : A direção dos vetores tangentes no ponto de junção é igual.

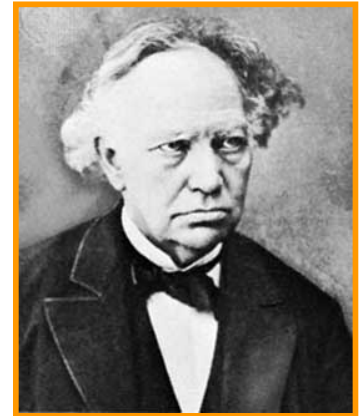


C^2 : A direção e a magnitude dos vetores tangentes no ponto de junção são iguais.

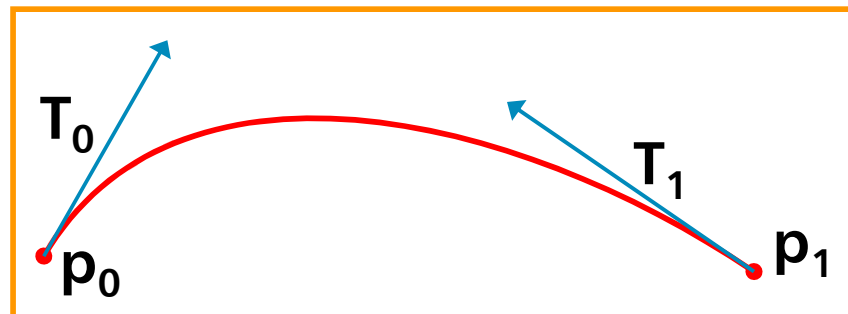


Curvas de Hermite

Charles Hermite descreveu extensamente o uso de polinômios de 3ª ordem para o ajuste de curvas. Seu trabalho é a base para os demais modelos de curvas.



A geometria proposta por Hermite propõe um interpolador local controlado por 4 fatores a cada 2 pontos: Os pontos inicial e final da curva (p_0 e p_1) e os vetores tangentes à curva nestes pontos (T_0 e T_1).





Curvas de Hermite

A Geometria de Hermite, considerando a componente x , é representada por:

$$GH_x = \begin{bmatrix} P_0 \\ P_1 \\ T_0 \\ T_1 \end{bmatrix}$$

Sabemos que componente $x(t)$ da curva $Q(t)$ é definida por:

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$
$$x(t) = T \cdot C_x = T \cdot M_H \cdot GH_x = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot M_H \cdot GH_x$$



Curvas de Hermite

As restrições impostas pela geometria de Hermite (p_0 , p_1 , T_0 e T_1) são obtidas pela substituição direta em $x(t)$ e em $x'(t)$.

$x'(t)$ é a primeira derivada de $x(t)$ e fornece a tangente à curva $Q(t)$ no ponto t .

Para p_0 ($t = 0$) e p_1 ($t = 1$) temos:

$$\begin{aligned}x(t) &= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \\x(0) &= \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \cdot M_H \cdot GH_x \\x(1) &= \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \cdot M_H \cdot GH_x\end{aligned}$$



Curvas de Hermite

Para T_0 ($t = 0$) e T_1 ($t = 1$), aplicados em $x'(t)$, temos:

$$\begin{aligned}x'(t) &= \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \\x'(0) &= \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \cdot M_H \cdot GH_x \\x'(1) &= \begin{bmatrix} 3 & 2 & 1 & 0 \end{bmatrix} \cdot M_H \cdot GH_x\end{aligned}$$

As 4 restrições podem ser reescritas matricialmente.

$$\begin{bmatrix} P_0 \\ P_1 \\ T_0 \\ T_1 \end{bmatrix}_x = G_{Hx} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot M_H \cdot G_{Hx}$$



Curvas de Hermite

Para que a igualdade anterior seja satisfeita (também para as expressões em y e z), M_H deve ser a inversa da matriz 4x4.

$$M_H = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Matriz de Hermite

Curvas de Hermite

Finalmente:

$$Q(t) = [x(t) \quad y(t) \quad z(t)] = T \cdot M_H \cdot G_H$$

$$Q(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ p_1 \\ T_0 \\ T_1 \end{bmatrix}$$

$$\text{com } \begin{bmatrix} p_0 \\ p_1 \\ T_0 \\ T_1 \end{bmatrix} = \begin{bmatrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ T_0x & T_0y & T_0z \\ T_1x & T_1y & T_1z \end{bmatrix}$$

Matriz de Hermite

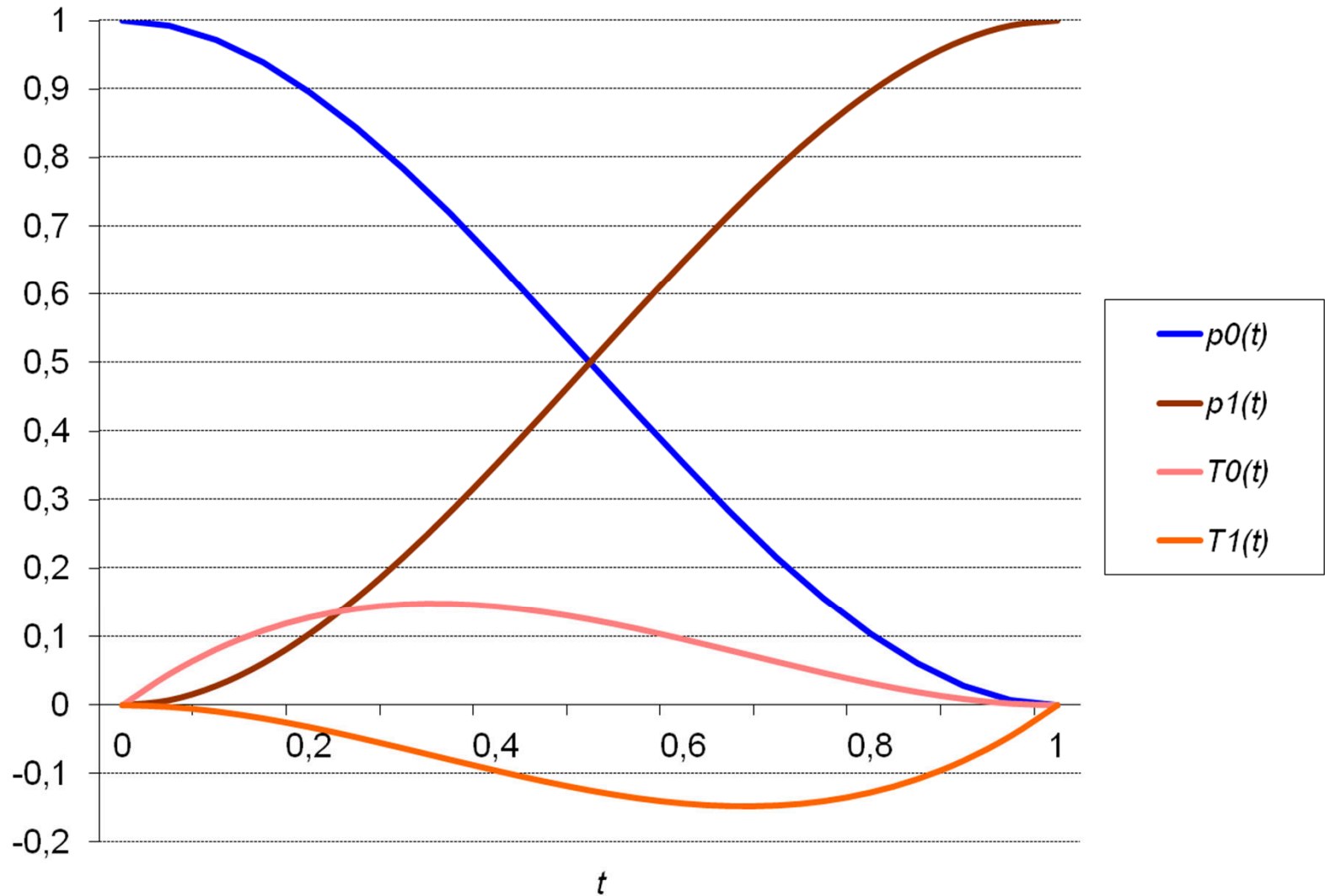
A matriz de Hermite define a contribuição de cada elemento da geometria (p_0 , p_1 , T_0 e T_1) em função do parâmetro $0 \leq t \leq 1$.

$$Q(t) = (2t^3 - 3t^2 + 1) \cdot p_0 + (-2t^3 + 3t^2) \cdot p_1 + (t^3 - 2t^2 + t) \cdot T_0 + (t^3 - t^2) \cdot T_1$$

A matriz de Hermite define os quatro polinômios acima, que ponderam os elementos da Geometria de Hermite. Estes polinômios são representados graficamente no slide seguinte.



Polinômios de Hermite

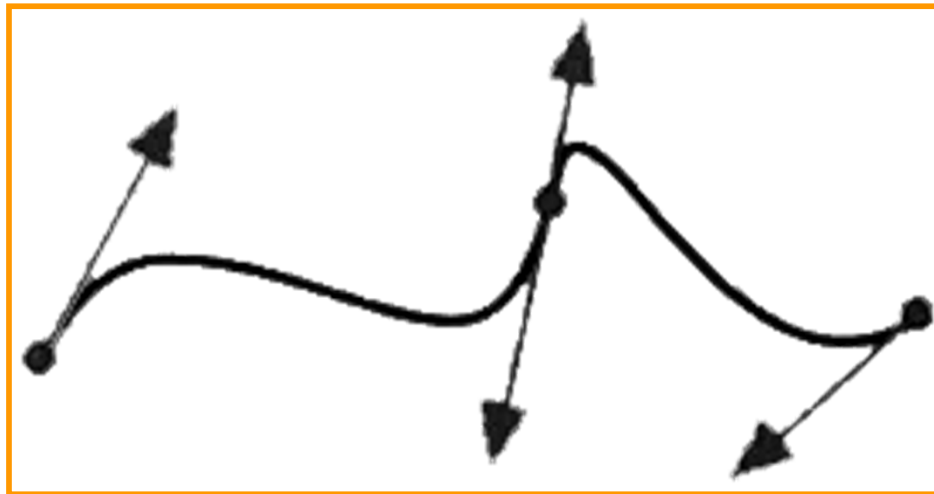




Os Vetores Tangentes

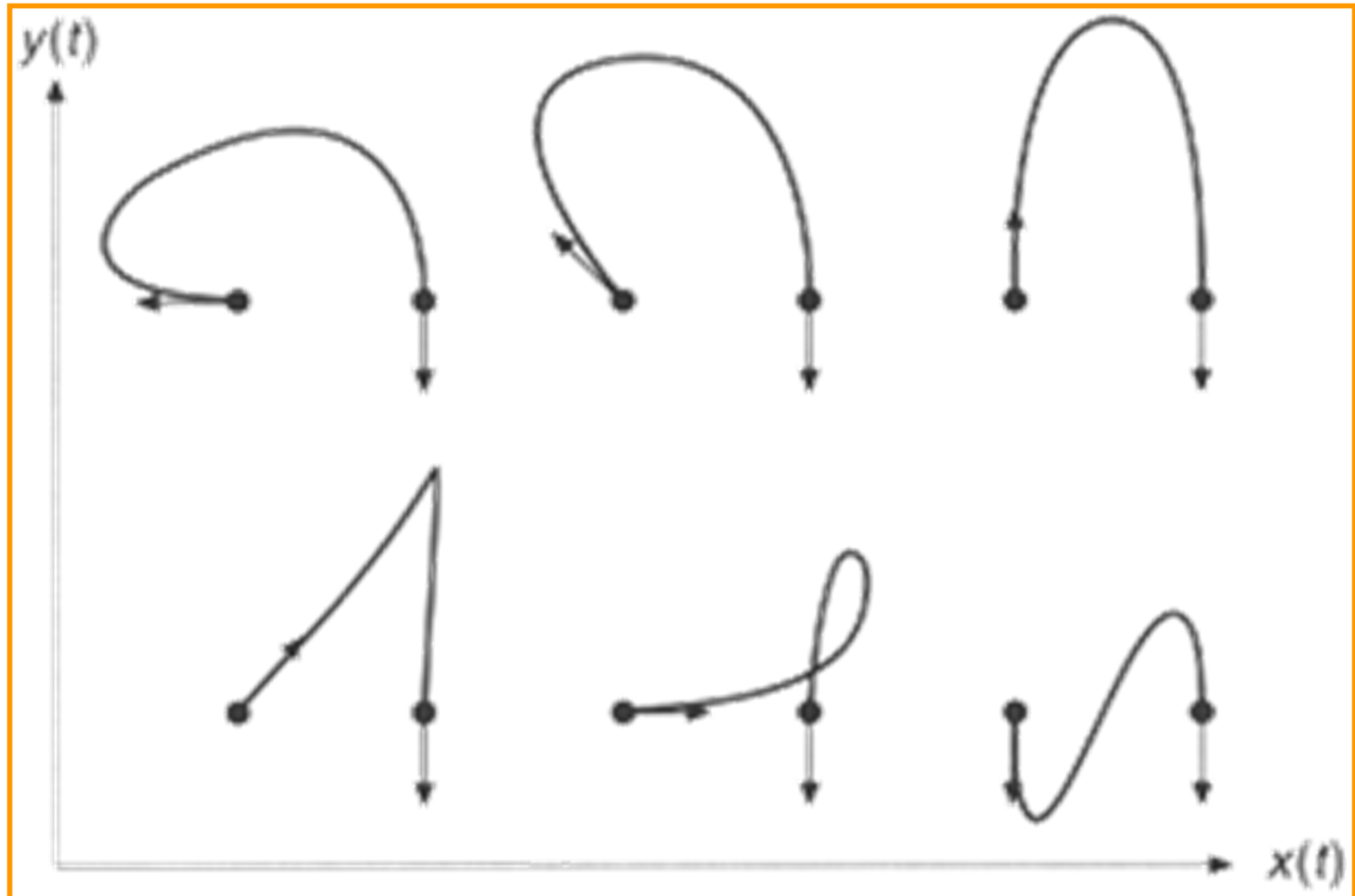
O controle dos vetores tangentes de entrada e saída das curvas influencia na “suavização” da curva total.

Uma curva é contínua se o vetor tangente na saída do primeiro segmento tem a mesma direção do vetor tangente na entrada do segundo segmento.





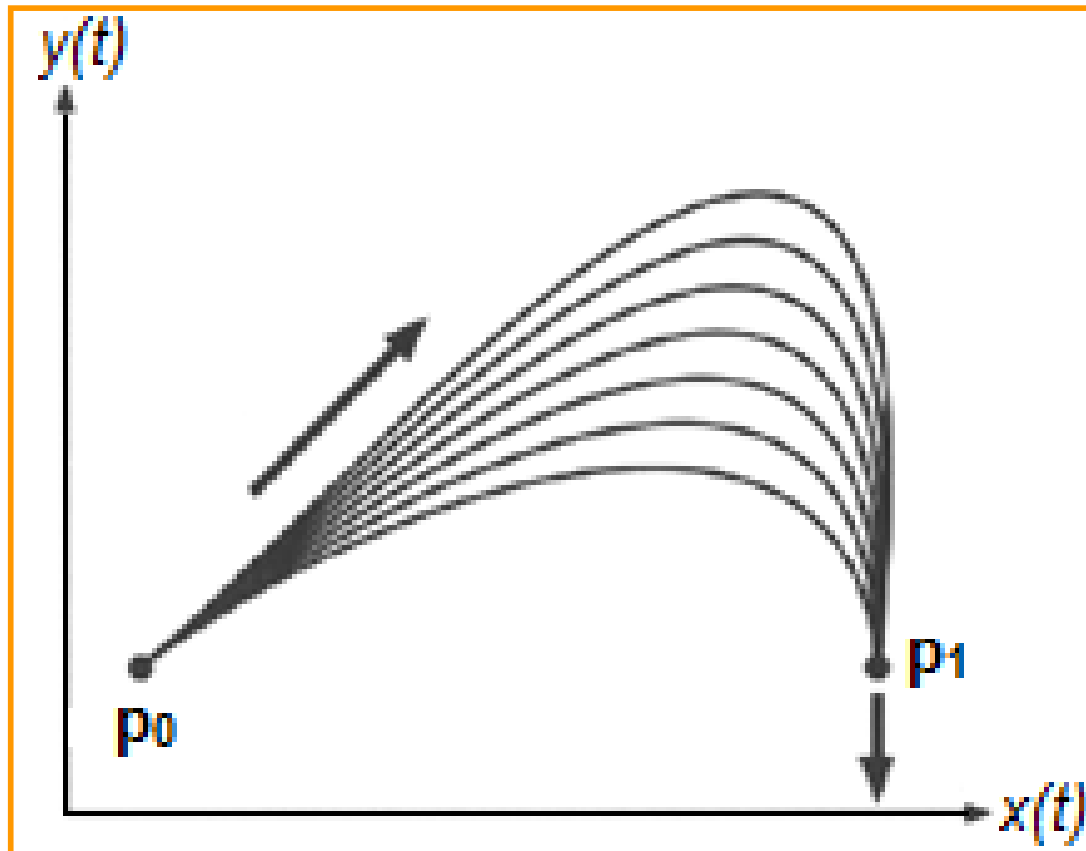
Efeito da Variação na Direção da Tangente (Hermite)





O Efeito da Magnitude do Vetor Tangente

A magnitude do vetor tangente afeta a "agressividade" da curva de Hermite.





Controle Automático dos Vetores Tangentes

Ao desenhar curvas de Hermite o controle manual das tangentes pode ser tedioso. Nestes casos um mecanismo de controle automático é desejável.

Dados 4 pontos definidos por $p_0 = (x_0, y_0)$, $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$ e $p_3 = (x_3, y_3)$ são geradas 4 tangentes necessárias à geração de 3 segmentos de curva contínuos, assim calculadas.

$$Tp_0 = (T_0x, T_0y) = (x_1 - x_0, y_1 - y_0)$$

$$Tp_1 = (T_1x, T_1y) = (x_2 - x_0, y_2 - y_0)$$

$$Tp_2 = (T_2x, T_2y) = (x_3 - x_1, y_3 - y_1)$$

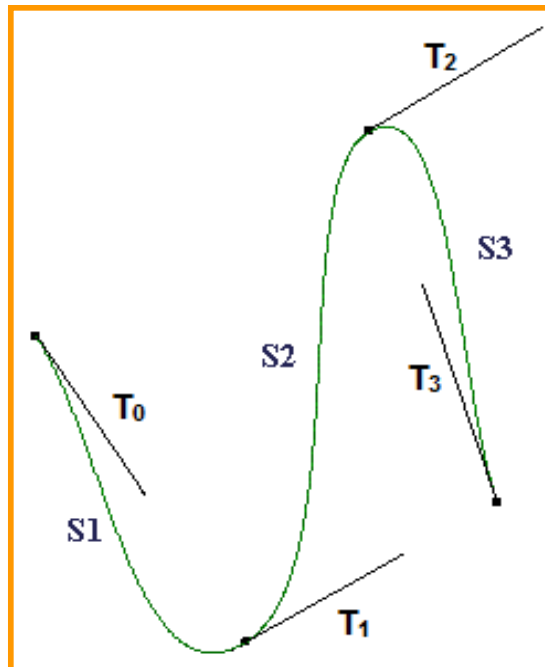
$$Tp_3 = (T_3x, T_3y) = (x_3 - x_2, y_3 - y_2)$$



Controle Automático dos Vetores Tangentes

Um fator de correção para o “peso” das tangentes para ajustar a “agressividade” da curva.

A curva a seguir foi gerada com peso igual a 50% para as tangentes.





Curvas de Hermite – Algoritmo

```
i = 0;
while(i+1 < TotMarks) { //TotMarks = número total de pontos na curva
    RangeX = fabs (X[i+1] - X[i]);
    RangeY = fabs (Y[i+1] - Y[i]);
    if(RangeX > RangeY) Step = 1.0/RangeX;
    else Step = 1.0/RangeY;

    //Determinação automática das tangentes
    if(i == 0) {
        T1X = X[i+1] - X[i];
        T2X = X[i+2] - X[i];
        T1Y = Y[i+1] - Y[i];
        T2Y = Y[i+2] - Y[i];
    }
    else if (i != 0 && i != TotMarks-2) {
        T1X = X[i+1] - X[i-1];
        T2X = X[i+2] - X[i];
        T1Y = Y[i+1] - Y[i-1];
        T2Y = Y[i+2] - Y[i];
    }
}
```

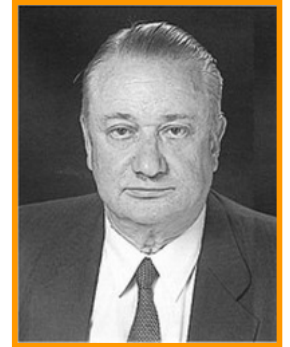


Curvas de Hermite – Algoritmo

```
else {
    T1X = X[i+1] - X[i-1];
    T2X = X[i+1] - X[i];
    T1Y = Y[i+1] - Y[i-1];
    T2Y = Y[i+1] - Y[i];
}
WG = 0.5;
for(t = 0; t <= 1; t += Step) {
    x = 0.5 + (( 2*pow(t,3) -3*pow(t,2) +0*t +1) * X[i] +
               (-2*pow(t,3) +3*pow(t,2) +0*t +0) * X[i+1] +
               ( 1*pow(t,3) -2*pow(t,2) +1*t +0) * WG*T1X +
               ( 1*pow(t,3) -1*pow(t,2) +0*t +0) * WG*T2X);
    y = 0.5 + (( 2*pow(t,3) -3*pow(t,2) +0*t +1) * Y[i] +
               (-2*pow(t,3) +3*pow(t,2) +0*t +0) * Y[i+1] +
               ( 1*pow(t,3) -2*pow(t,2) +1*t +0) * WG*T1Y +
               ( 1*pow(t,3) -1*pow(t,2) +0*t +0) * WG*T2Y);
    if(t == 0) MoveTo (hdc, x, y);
    else LineTo (hdc, x, y);
}
LineTo (hdc, X[i+1], Y[i+1]);
i++;
}
```



Algoritmo de De Casteljau

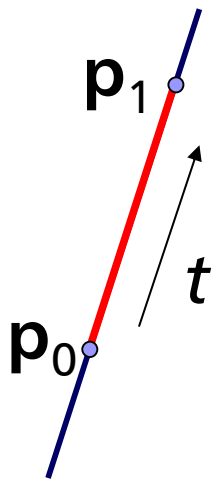


Suponha que queiramos aproximar uma curva polinomial entre os pontos p_0 e p_1 .

A solução natural é um segmento de reta entre p_0 e p_1 , parametrizado por:

$$p(t) = (1 - t).p_0 + t.p_1$$

Podemos interpretar $p(t)$ como uma média ponderada entre p_0 e p_1 .



Os polinômios $(1 - t)$ e t somam 1 para qualquer valor de t . Estes polinômios são chamados de **funções de mistura** (*blending functions*).



Algoritmo de De Casteljau

Generalizamos a ideia para 3 pontos p_0 , p_1 e p_2 , considerando os segmentos de reta p_0p_1 e p_1p_2 .

$$p_{01}(t) = (1 - t).p_0 + t.p_1$$

$$p_{12}(t) = (1 - t).p_1 + t.p_2$$

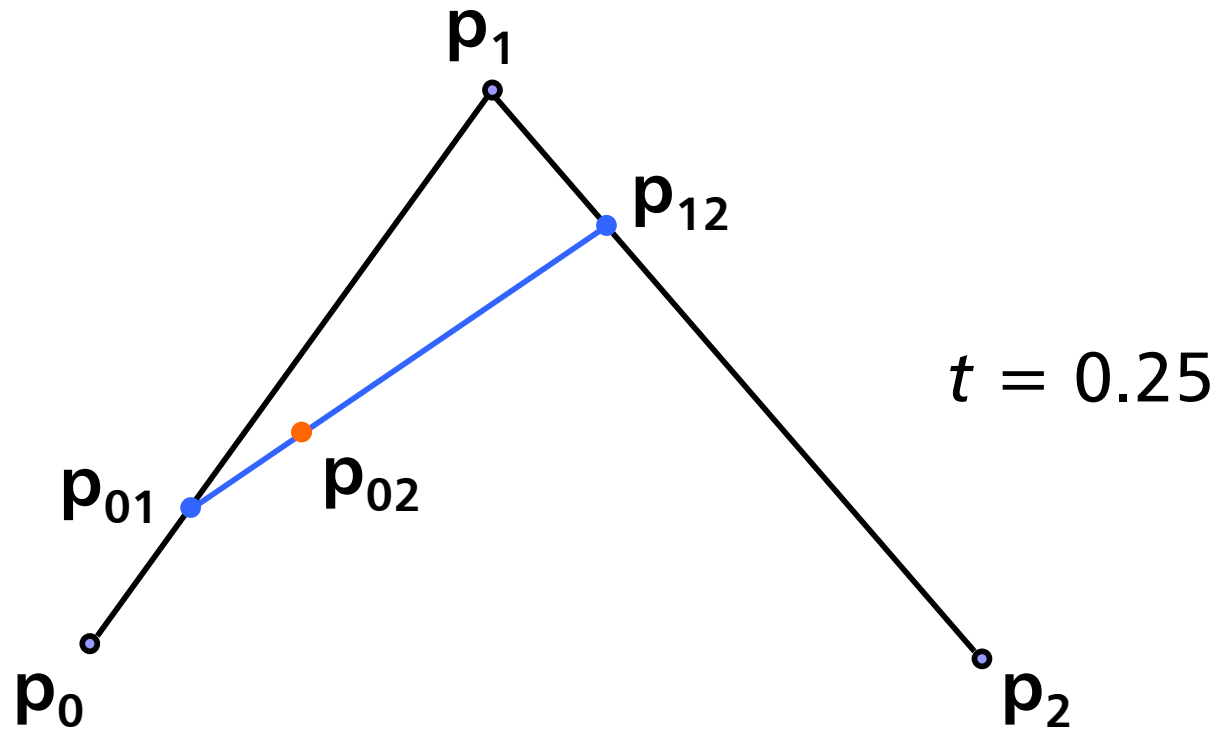
Podemos agora realizar uma interpolação entre $p_{01}(t)$ e $p_{12}(t)$

$$p_{02}(t) = (1 - t).p_{01}(t) + t.p_{12}(t)$$

$$p_{02}(t) = (1 - t)^2.p_0 + 2t.(1 - t).p_1 + t^2.p_2$$

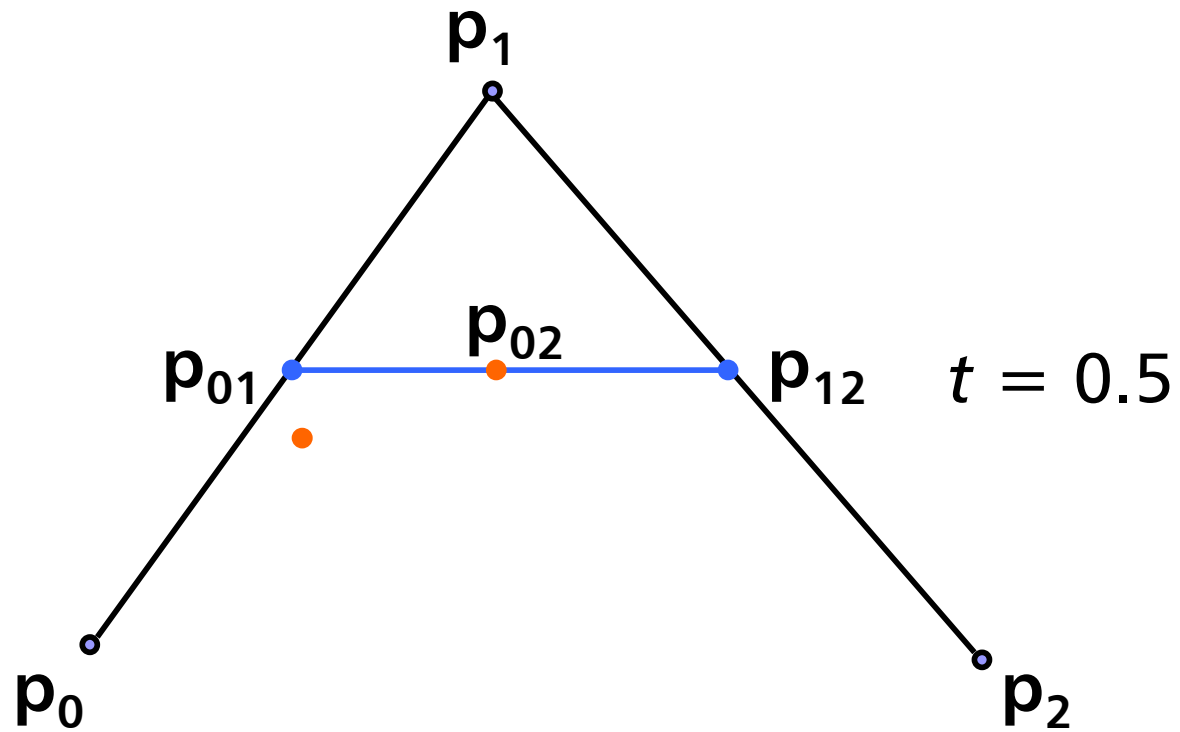


Algoritmo de De Casteljau



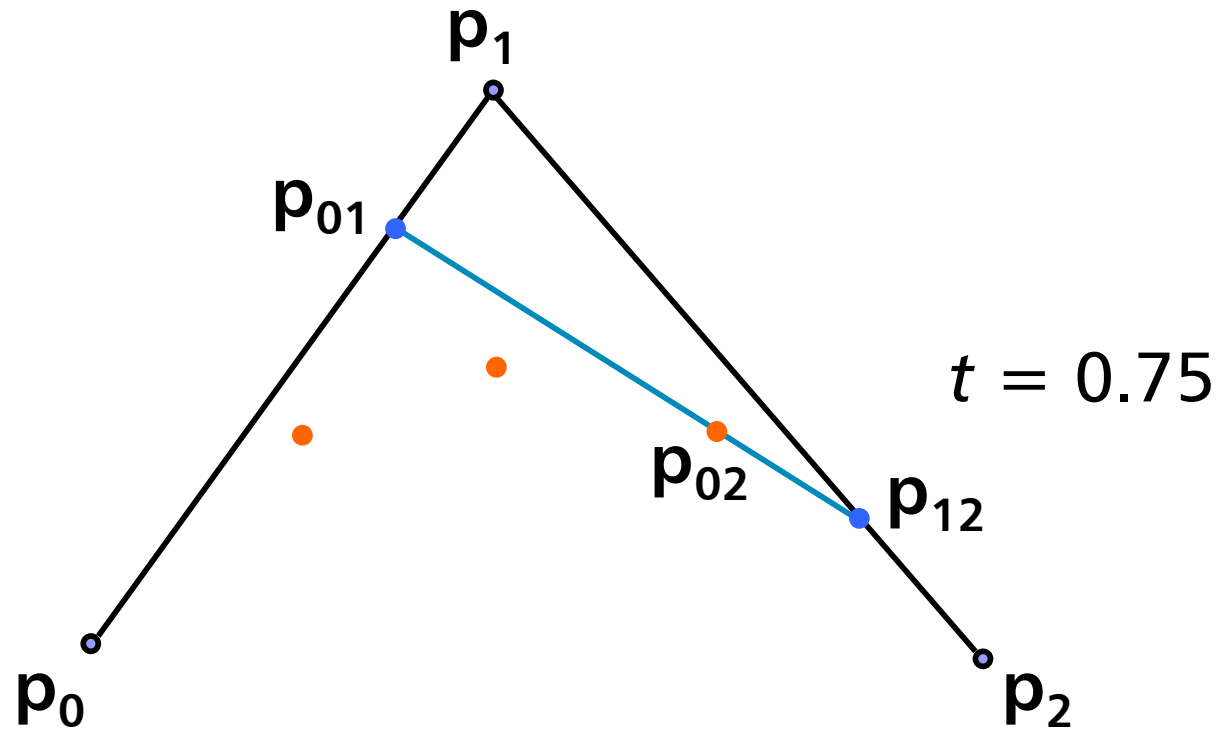


Algoritmo de De Casteljau



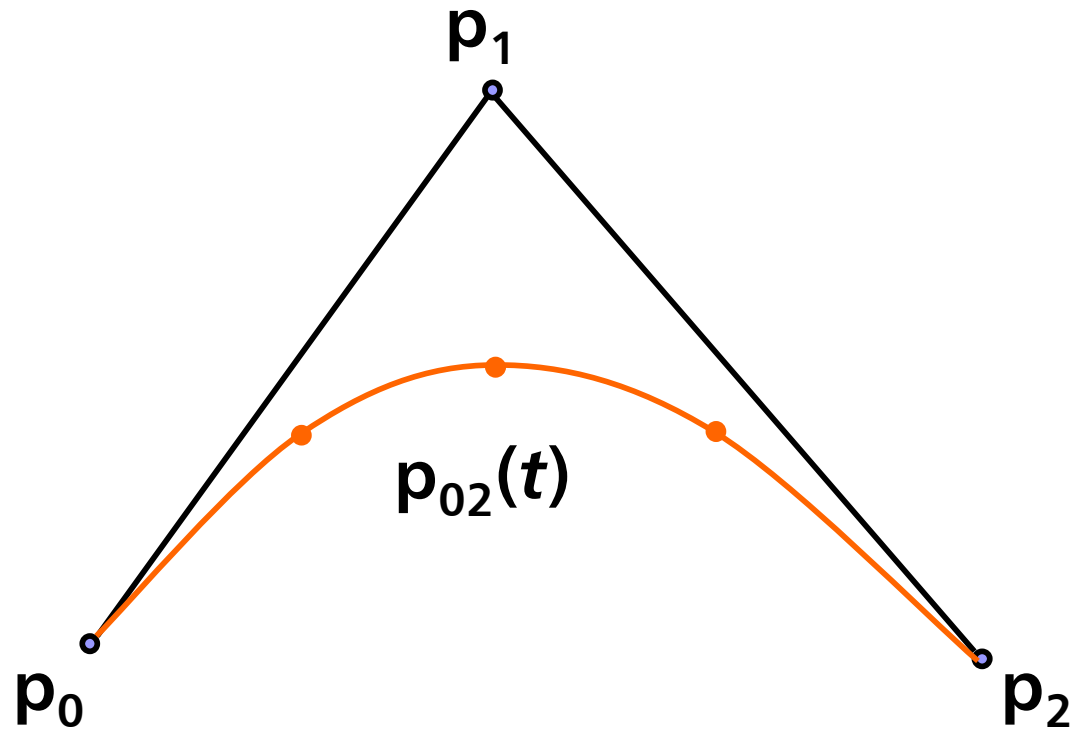


Algoritmo de De Casteljau

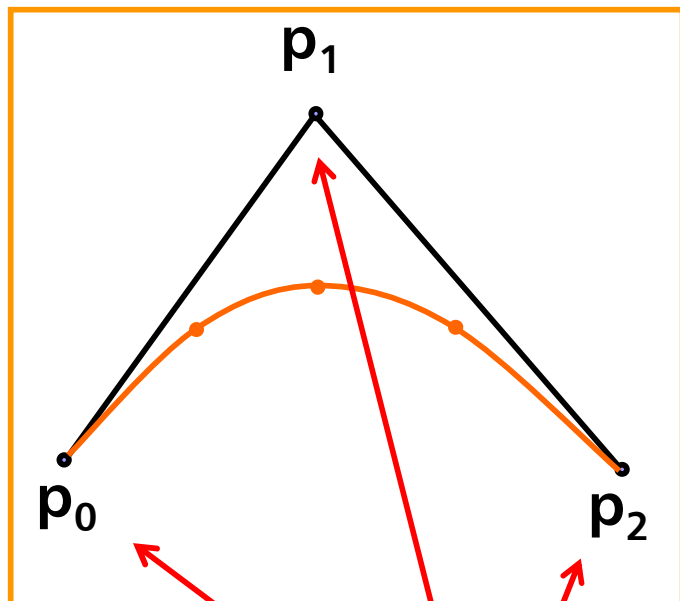




Algoritmo de De Casteljau



Algoritmo de De Castelja



Podemos dizer que a curva é obtida pela “mistura” dos pontos p_0 , p_1 e p_2 ponderadas por 3 funções quadráticas.

$$b_{02}(t) = (1 - t)^2$$

$$b_{12}(t) = 2t \cdot (1 - t)$$

$b_{22}(t) = t^2$

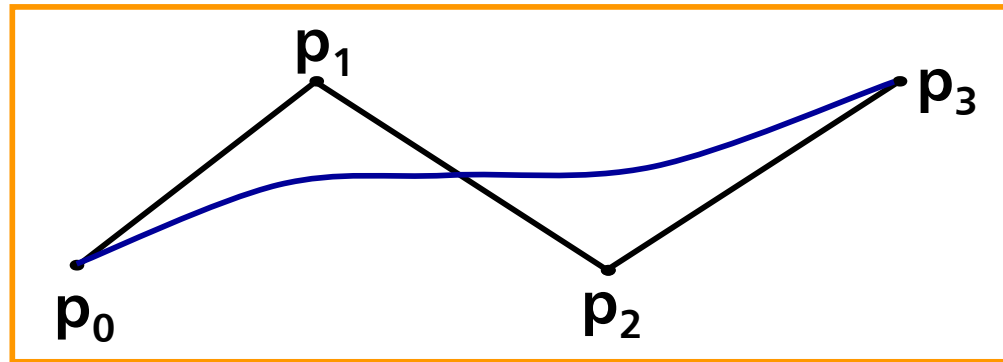
Geometria!

Funções
de mistura



Algoritmo de De Casteljau

Aplicando novamente a ideia podemos definir uma cúbica por 4 pontos (p_0 , p_1 , p_2 e p_3).



$$p_{02}(t) = (1 - t)^2 \cdot p_0 + 2t(1 - t) \cdot p_1 + t^2 \cdot p_2$$

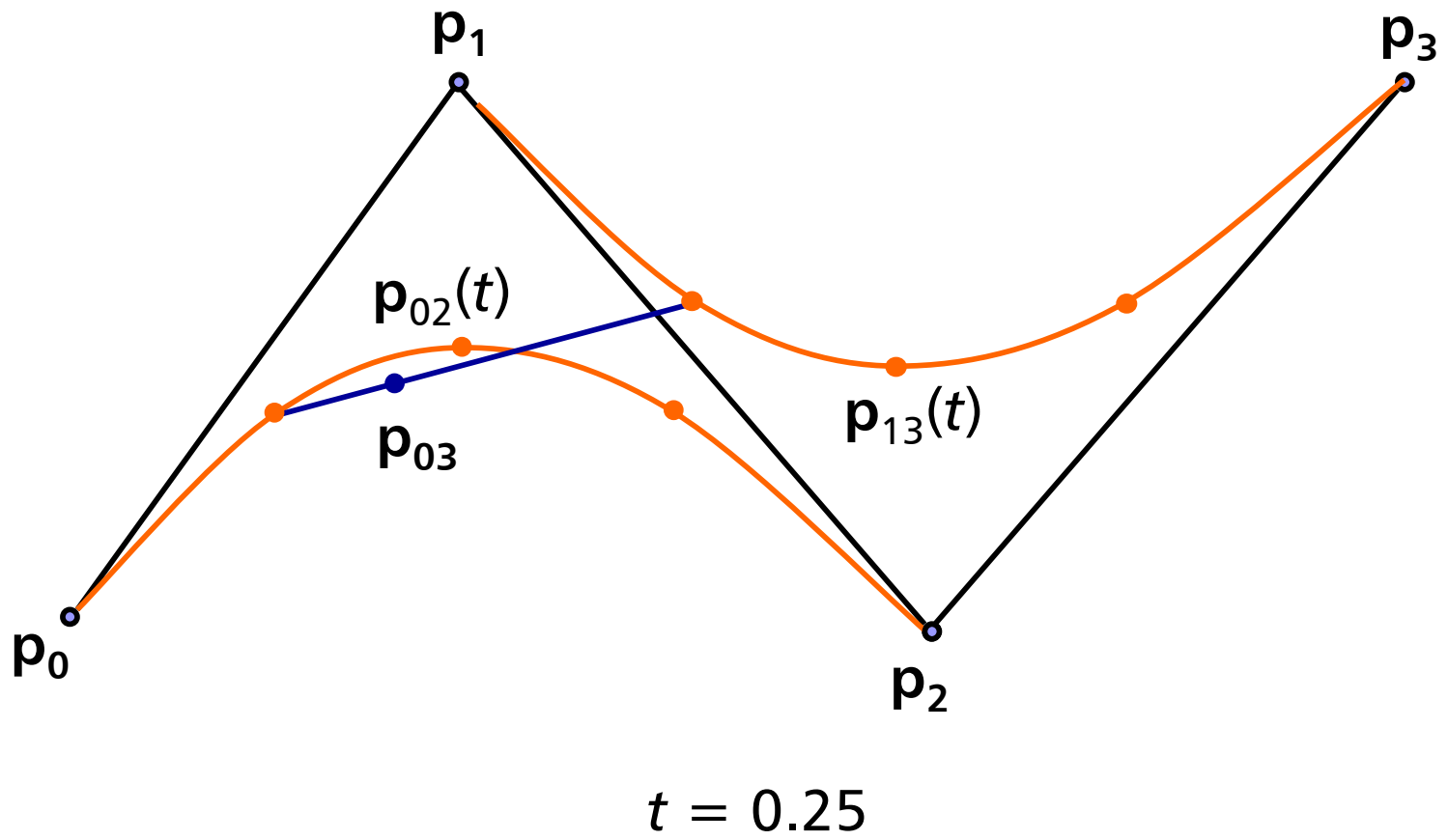
$$p_{13}(t) = (1 - t)^2 \cdot p_1 + 2t(1 - t) \cdot p_2 + t^2 \cdot p_3$$

$$p_{03}(t) = (1 - t) \cdot p_{02}(t) + t \cdot p_{13}(t)$$

$$p_{03}(t) = (1 - t)^3 \cdot p_0 + 3t(1 - t)^2 \cdot p_1 + 3t^2(1 - t) \cdot p_2 + t^3 \cdot p_3$$

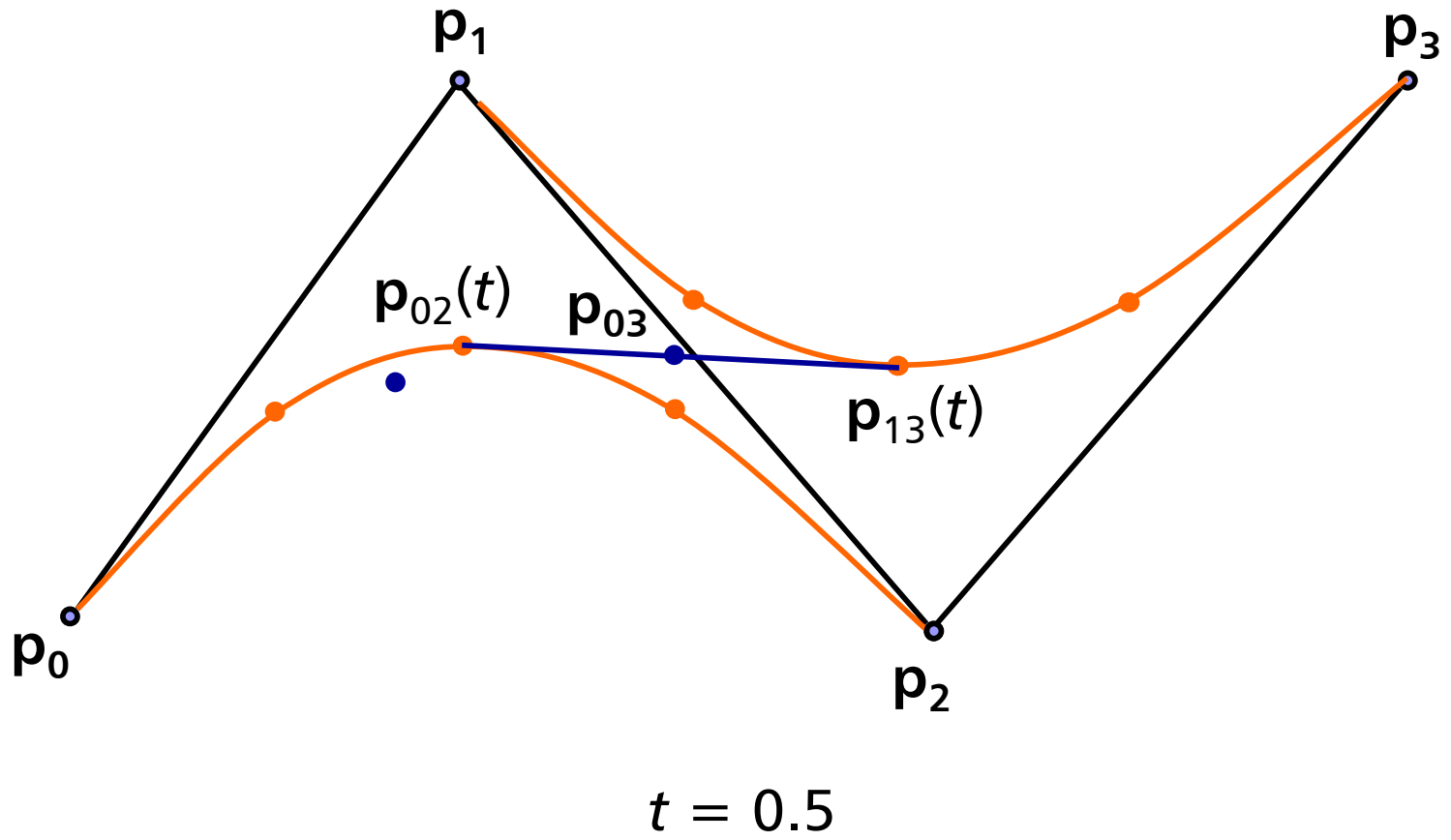


Algoritmo de De Casteljau



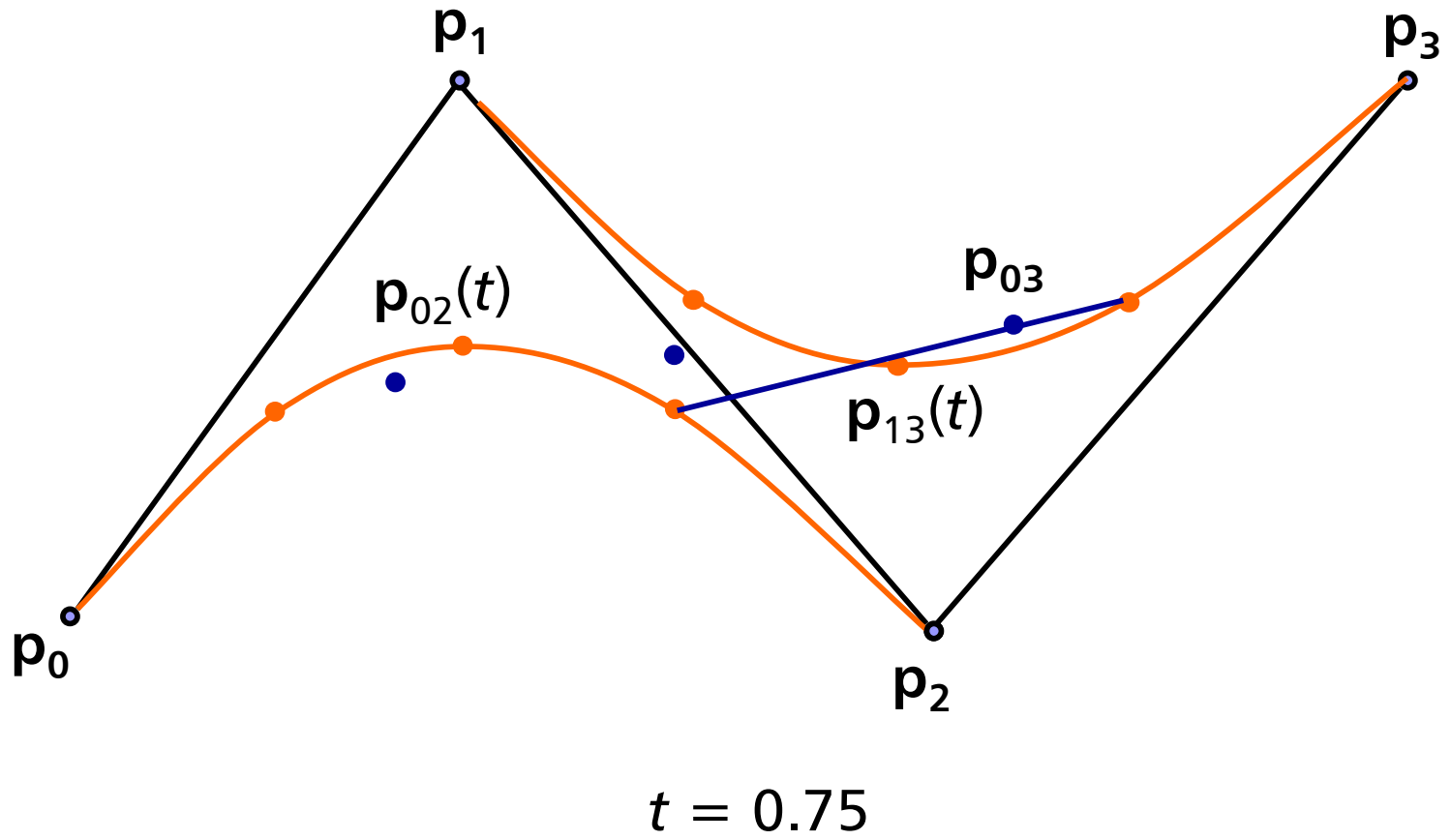


Algoritmo de De Casteljau



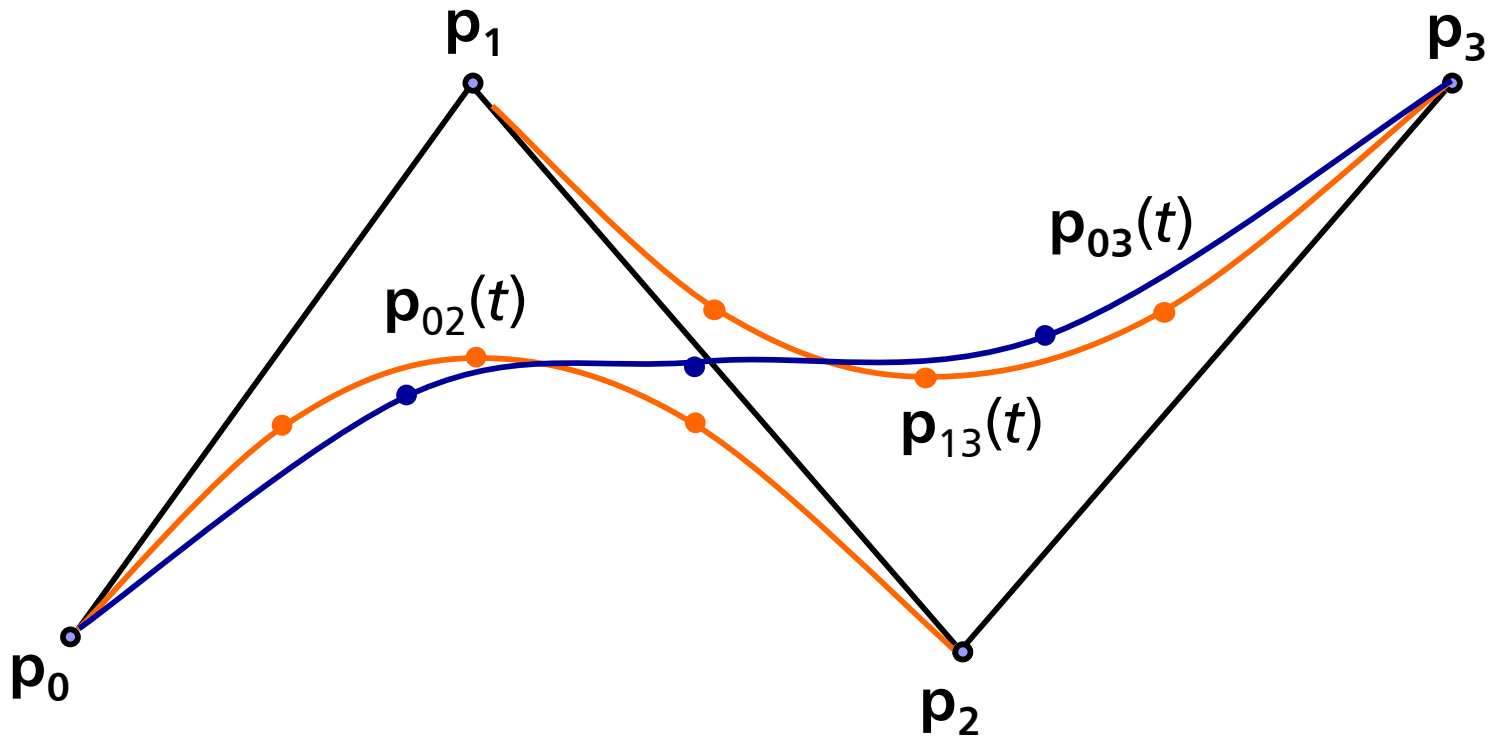


Algoritmo de De Casteljau

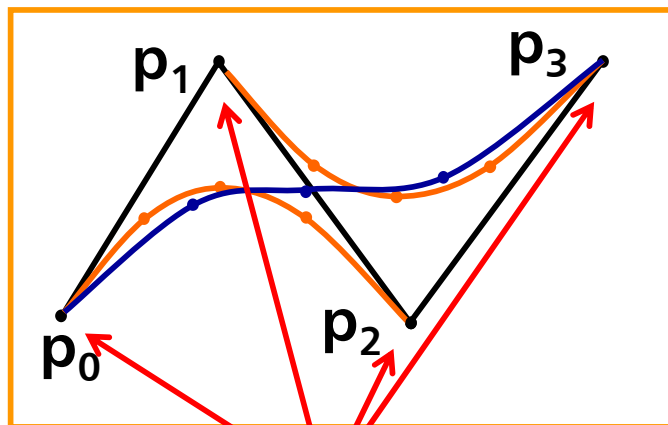




Algoritmo de De Casteljau



Algoritmo de De Casteljau



A curva é obtida por 4 funções de mistura (agora cúbicas) operadas sobre os pontos p_0 , p_1 , p_2 e p_3 .

Geometria!

Generalizando

$$\mathbf{p}_{0n}(t) = \sum_{j=0}^n b_{jn}(t) \mathbf{p}_j$$

$$\begin{aligned} b_{03}(t) &= (1-t)^3 \\ b_{13}(t) &= 3t.(1-t)^2 \\ b_{23}(t) &= 3t^2.(1-t) \\ b_{33}(t) &= t^3 \end{aligned}$$

Funções de mistura



Curvas de Bézier e Polinômios de Bernstein

As curvas construídas pelo Algoritmo de De Casteljau são conhecidas como *curvas de Bézier* e as funções de mistura são chamadas de *base de Bézier* ou *Polinômios de Bernstein*.

Os Polinômios de Bernstein de grau n têm como forma geral $b_{in}(t) = c_i \cdot t^i \cdot (1 - t)^{n-i}$

Escrevendo as constantes c_i para os diversos polinômios teremos:

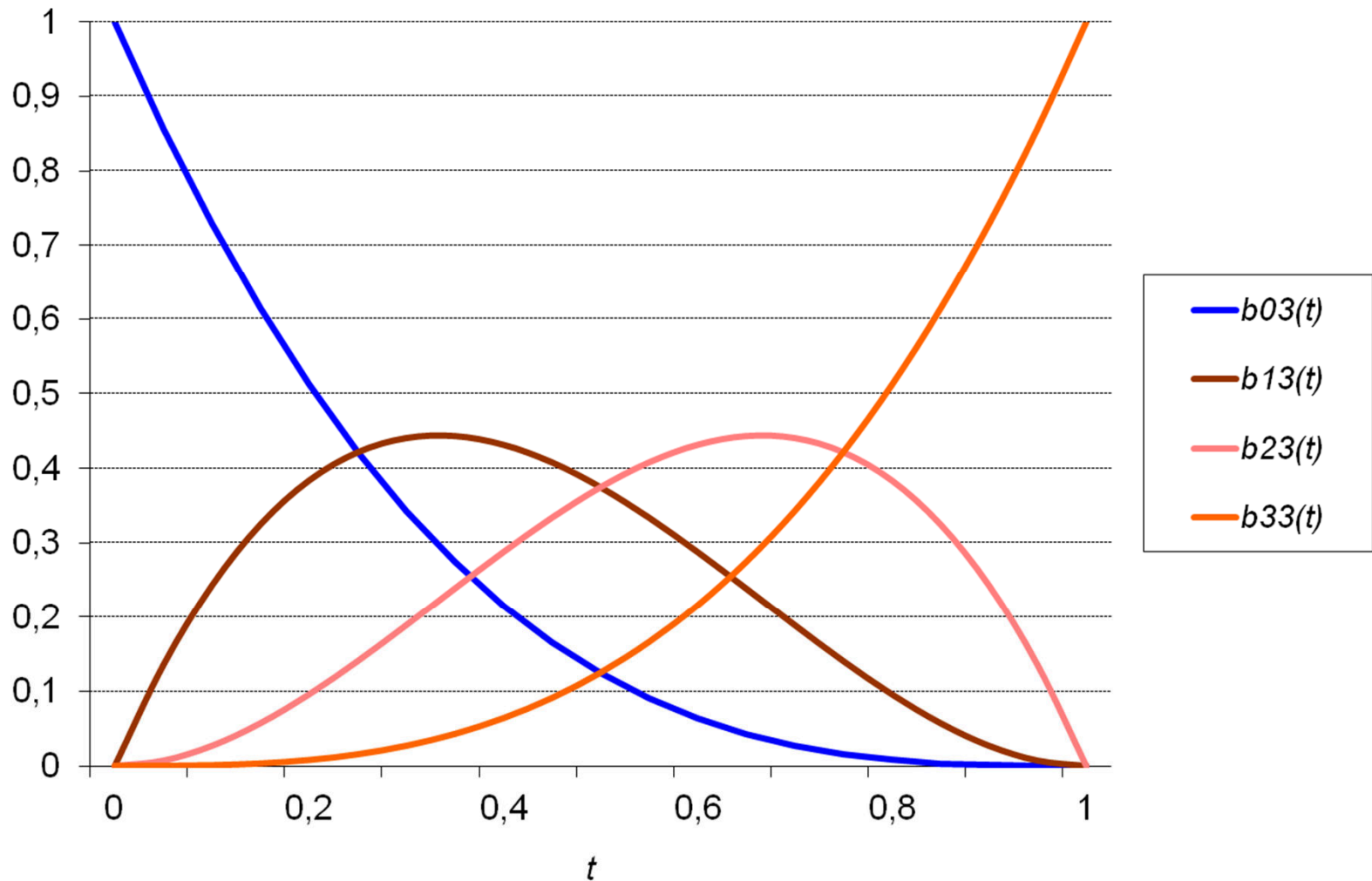
- 1º grau: 1 1
- 2º grau: 1 2 1
- 3º grau: 1 3 3 1
- 4º grau: 1 4 6 4 1



$$b_{in}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$



Polinômios de Bernstein de Grau 3



Matriz de Bézier


Sabendo que o polinômio de Bézier para uma curva $Q(t)$ definida pelos pontos p_0, p_1, p_2 e p_3 é:

$$Q(t) = (1 - t)^3.p_0 + 3t.(1 - t)^2.p_1 + 3t^2.(1 - t).p_2 + t^3.p_3$$

Expandido $Q(t)$ temos:

$$Q(t) = (-t^3 + 3t^2 - 3t + 1).p_0 + (3t^3 - 6t^2 + 3t).p_1 + (-3t^3 + 3t^2).p_2 + t^3.p_3$$

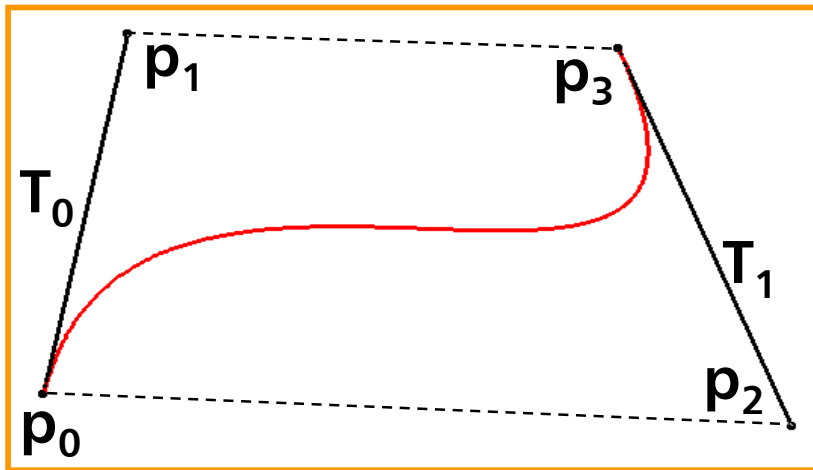
Matricialmente escrevemos $Q(t)$ como:


$$Q(t) = T \cdot M_B \cdot G = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$



Curvas de Bézier

Foram desenvolvidas pelo engenheiro Pierre Bézier para desenhar carros para a Renault. Devido a sua versatilidade tornaram-se padrão nos programas de desenho.



A curva é definida por 2 pontos extremos (p_0 e p_3) e outros dois pontos (p_1 e p_2) que controlam os extremos dos vetores tangentes (T_0 e T_1).

$$T_0x = 3(p_1x - p_0x) \text{ e } T_0y = 3(p_1y - p_0y)$$

$$T_1x = 3(p_2x - p_3x) \text{ e } T_1y = 3(p_2y - p_3y)$$



Curvas de Bézier – Algoritmo

```
i = 0;
while(i+3 < TotMarks) { //TotMarks = número total de pontos na curva
    RangeX = fabs (X[i+3] - X[i]);
    RangeY = fabs (Y[i+3] - Y[i]);
    if(RangeX > RangeY)
        Step = 1.0/RangeX;
    else
        Step = 1.0/RangeY;
    for (t = 0; t <= 1; t += Step) {
        x = ((-1*pow(t,3) +3*pow(t,2) -3*t +1)*X[i]      +
              ( 3*pow(t,3) -6*pow(t,2) +3*t +0)*X[i+1] +
              (-3*pow(t,3) +3*pow(t,2) +0*t +0)*X[i+2] +
              ( 1*pow(t,3) +0*pow(t,2) +0*t +0)*X[i+3]);

        y = ((-1*pow(t,3) +3*pow(t,2) -3*t +1)*Y[i]      +
              ( 3*pow(t,3) -6*pow(t,2) +3*t +0)*Y[i+1] +
              (-3*pow(t,3) +3*pow(t,2) +0*t +0)*Y[i+2] +
              ( 1*pow(t,3) +0*pow(t,2) +0*t +0)*Y[i+3]);

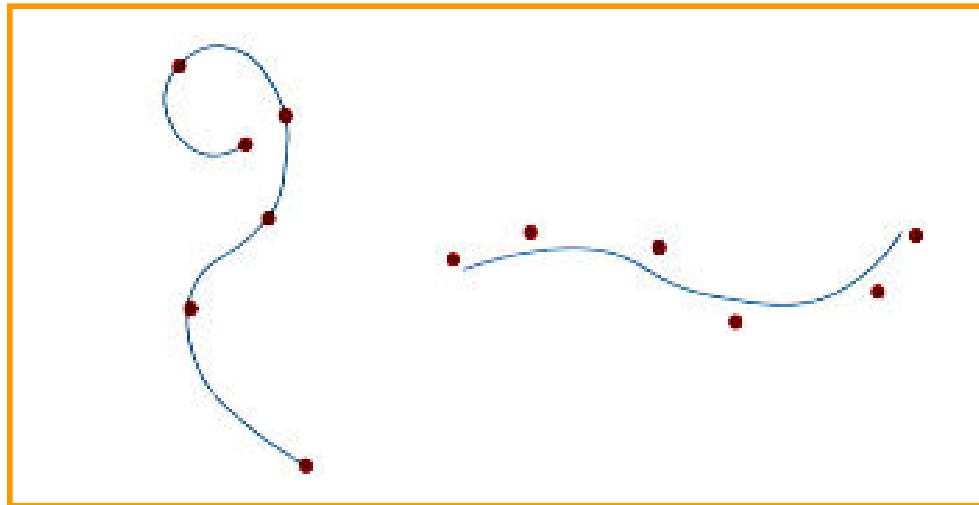
        if(t == 0) MoveTo (hdc, x, y);
        else LineTo (hdc, x, y);
    }
    i += 3;
}
```




Splines

Uma *spline* é uma linha flexível usada para produzir uma curva suavizada ao longo de uma série de pontos de controle.

Existem vários tipos de *splines*, cuja amostragem varia de acordo com a fórmula matemática utilizada na sua construção. Elas podem ser interpoladas ou aproximadas.



Splines

A forma de uma curva *spline* depende da distribuição e dos pesos dos pontos de controle. Manipulando os pontos e os pesos ajustam-se a curvatura do *spline*.

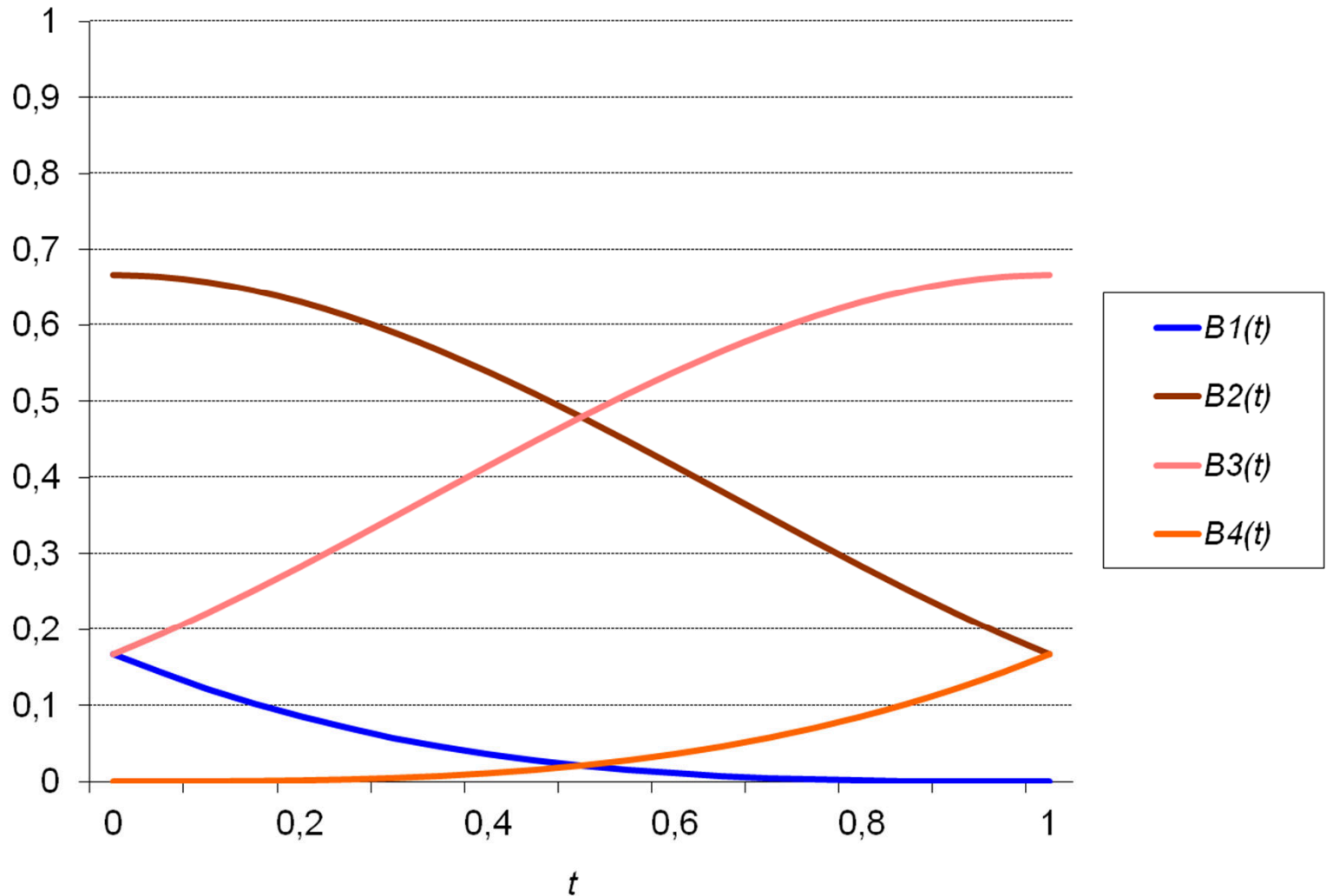
Matricialmente escrevemos um *B-spline* $Q(t)$ como:

$$Q(t) = T \cdot M_B \cdot G = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

Matriz *B-Spline*



Funções de Mistura para B-Spline





Curvas *B-Spline* – Algoritmo

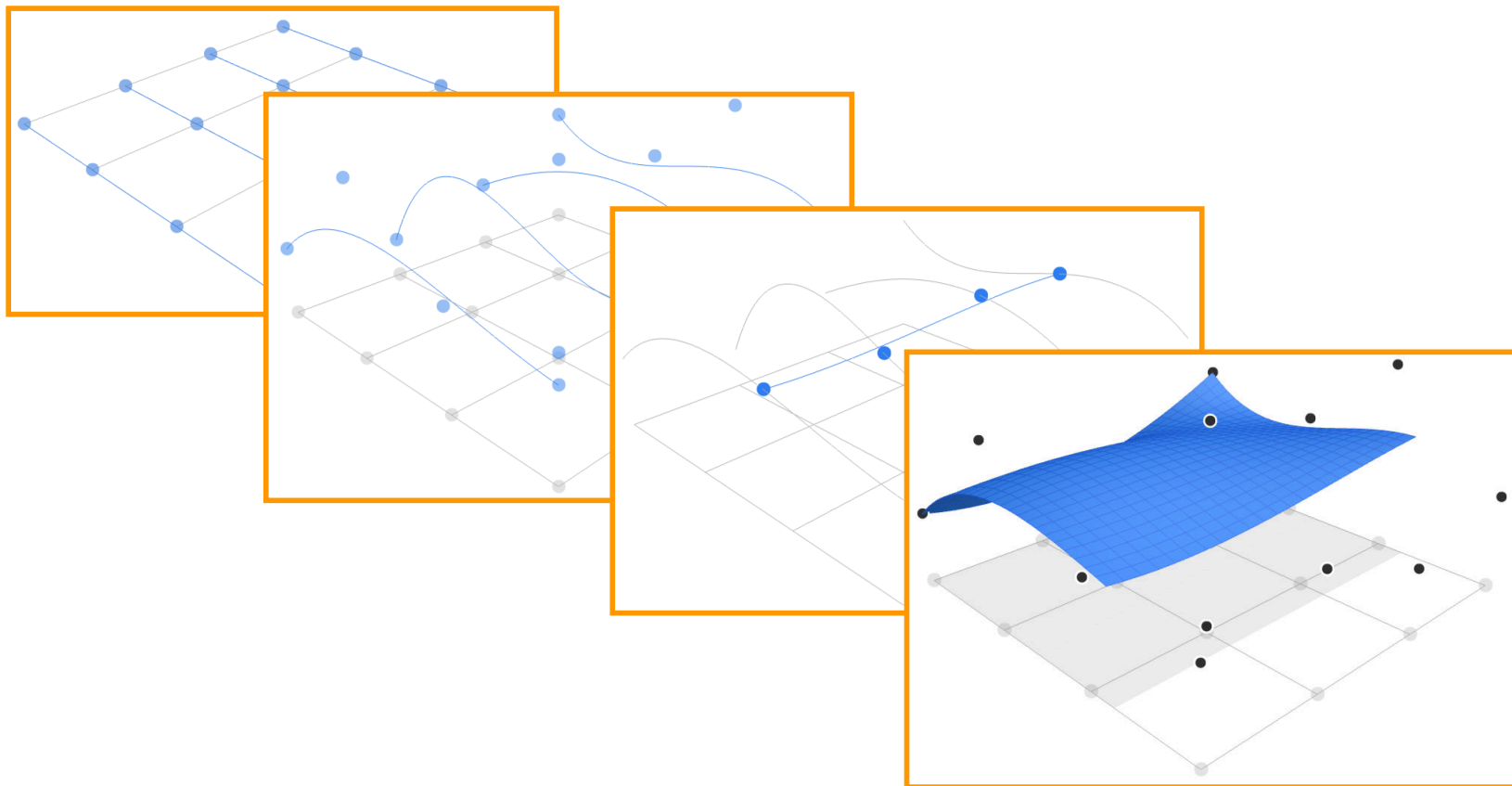
```
i = 0;
while(i+3 < TotMarks) { //TotMarks = número total de pontos na curva
    RangeX = fabs (X[i+2] - X[i+1]);
    RangeY = fabs (Y[i+2] - Y[i+1]);
    if(RangeX > RangeY) Step = 1.0/RangeX;
    else Step = 1.0/RangeY;

    for(t = 0; t <= 1; t += Step) {
        x = (((-1*pow(t,3) +3*pow(t,2) -3*t +1)*X[i]      +
              ( 3*pow(t,3) -6*pow(t,2) +0*t +4)*X[i+1] +
              (-3*pow(t,3) +3*pow(t,2) +3*t +1)*X[i+2] +
              ( 1*pow(t,3) +0*pow(t,2) +0*t +0)*X[i+3])/6);
        y = (((-1*pow(t,3) +3*pow(t,2) -3*t +1)*Y[i]      +
              ( 3*pow(t,3) -6*pow(t,2) +0*t +4)*Y[i+1] +
              (-3*pow(t,3) +3*pow(t,2) +3*t +1)*Y[i+2] +
              ( 1*pow(t,3) +0*pow(t,2) +0*t +0)*Y[i+3])/6);
        if(t == 0) MoveTo (hdc, x, y);
        else LineTo (hdc, x, y);
    }
    i++;
}
```



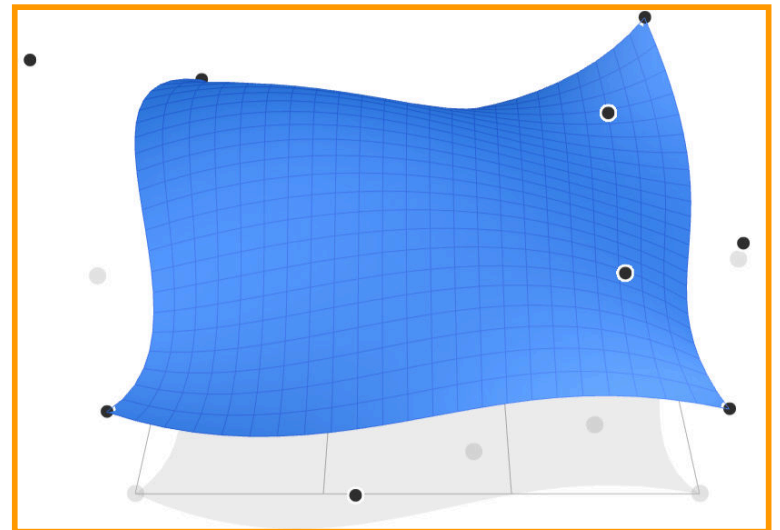
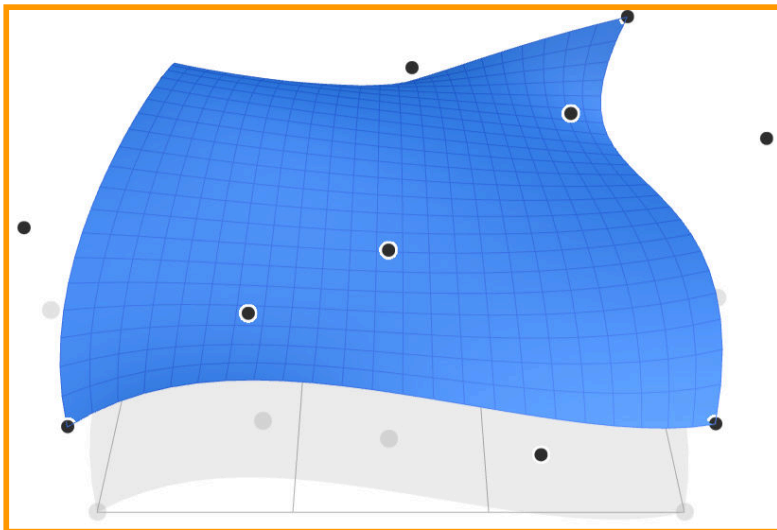
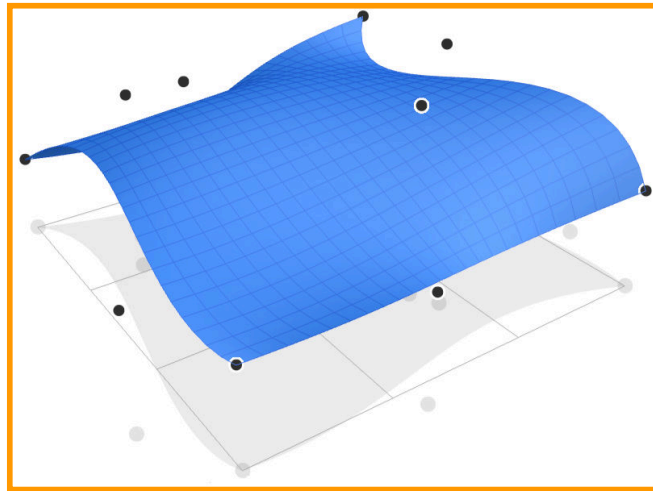
Superfícies de Bézier

Duas curvas de Bézier podem ser utilizadas para formar uma superfície de Bézier.





Superfícies de Bézier





Superfícies de Bézier

A superfície de Bézier é formada pelo produto cartesiano das funções das duas curvas geratrizes.

$$P(u, v) = \sum_{j=0}^m \sum_{k=0}^n p_{j,k} BEZ_{j,m}(v) BEZ_{k,n}(u)$$

$$BEZ_{j,m}(v) = C(m, j) v^j (1 - v)^{m-j}$$

$$BEZ_{k,n}(u) = C(n, k) u^k (1 - u)^{n-k}$$

$$C(m, j) = \frac{m!}{j!(m-j)!}$$

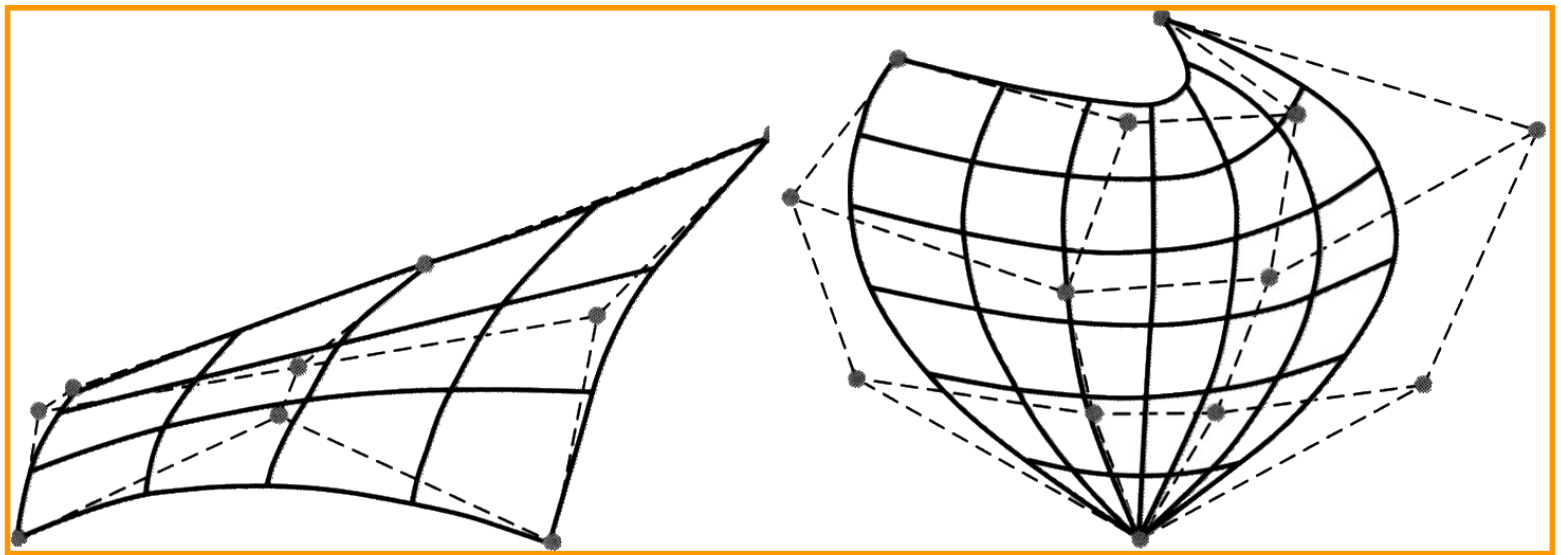
$$C(n, k) = \frac{n!}{k!(n-k)!}$$

$p_{j,k}$ são os $(m + 1)$ por $(n + 1)$ pontos de controle.



Superfícies de Bézier

A seguir dois exemplos de superfícies geradas com $m = 3, n = 3$ e $m = 4, n = 4$ pontos de controle.





Curvas e Superfícies de Bézier

Wittens, Steven. **Making things with Maths**. Disponível em:
<http://acko.net/files/fullfrontal/fullfrontal/wdcode/online.html>.

Acesso em 04/04/2014.



Linear interpolation

$$\text{lerp}(\vec{a}, \vec{b}, t) = \vec{a} + t \cdot (\vec{b} - \vec{a})$$