Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Modelagem Geométrica

André Tavares da Silva

andre.silva@udesc.br

Mortenson 2006: Cap11.2 e 11.6

Foley 1996 12.5

Boundary Representation (B-rep)

Representação por Superfícies Limítrofes

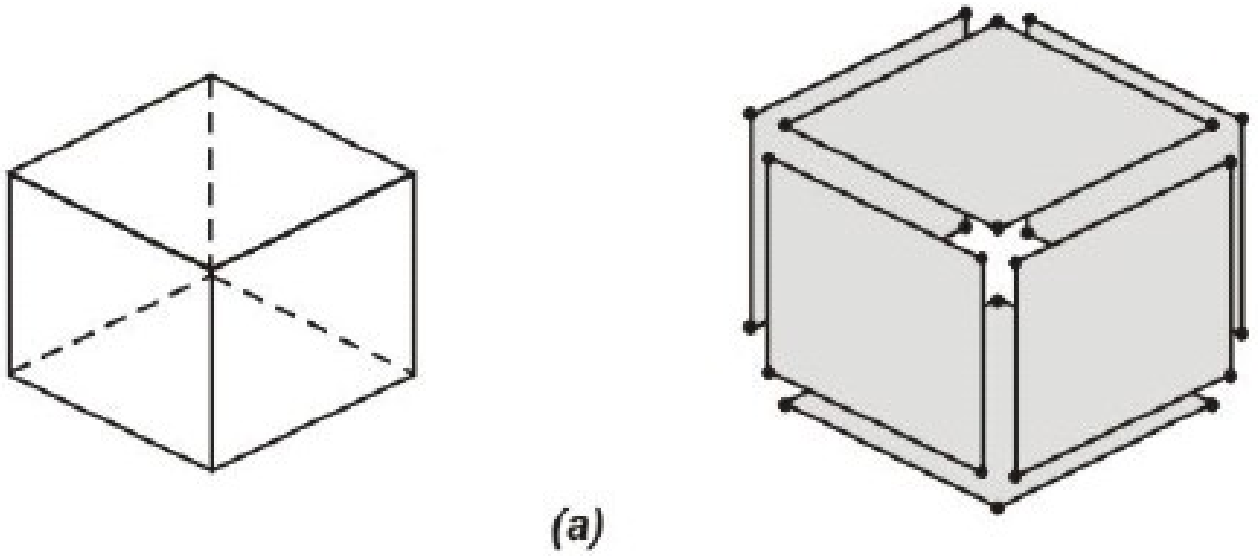
Representação por Bordos/Bordas

Representação por Fronteira

Representação por Faces (A&C:125)

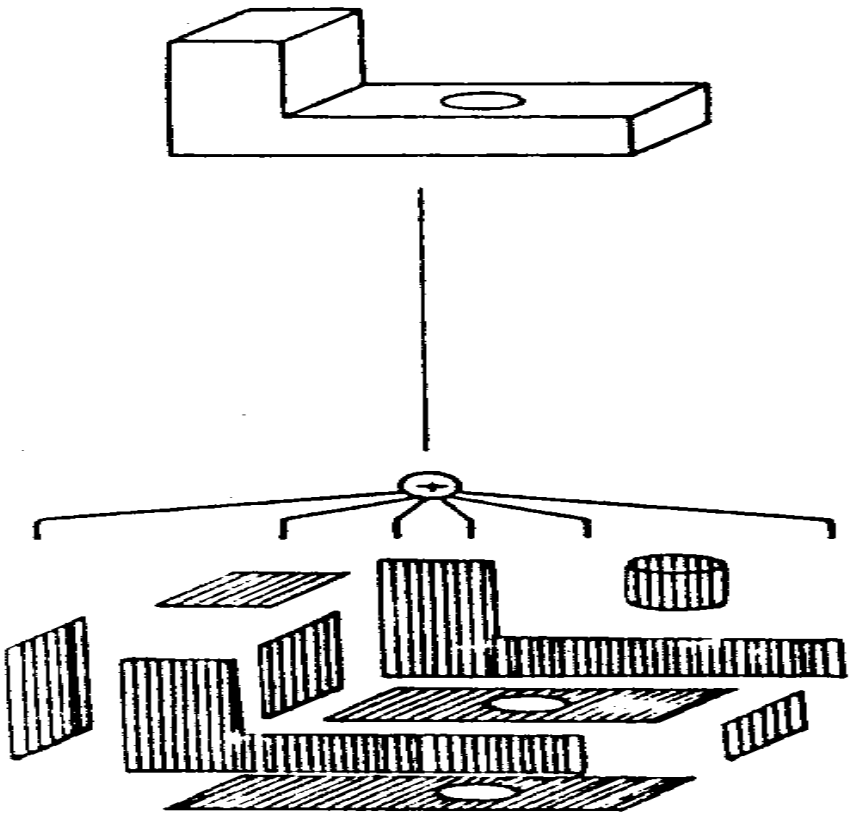
Representação por Bordos (Mortenson 2006)

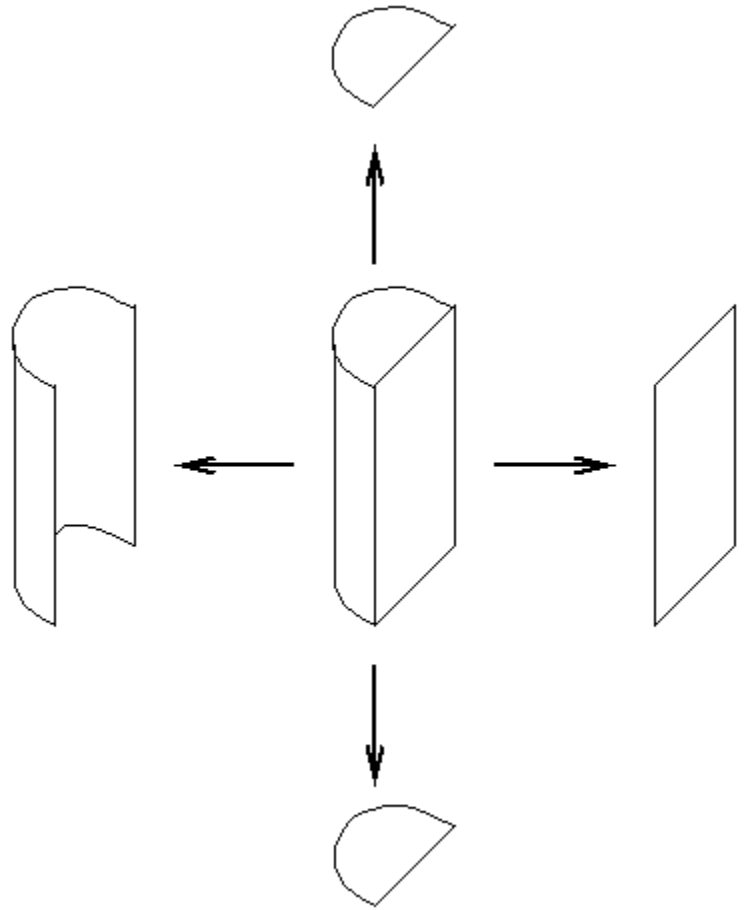
- Descreve completamente um objetos sólido como uma coleção organizada de suas **superfícies limitantes** (faces) (:377)



Representação por Bordos (Mortenson 2006)

- Representa todo o envoltório do objeto sólido (toda a "casca")
- Uma **superfície limitante** separa pontos que são internos e externos ao sólido (:379)





Representação por Bordos (Mortenson 2006)

- As **superfícies limitantes** do sólido devem atender às seguintes condições (377):
  - Fechadas por (um conjunto de) arestas
  - Não tenha singularidades (*dangling*)
  - Não auto-intersectantes
  - Todas conectadas
  - Todas limitantes do objeto
  - Cada uma orientada (com a mesma convenção)

Características do B-rep

- Limitações
  - Conceito de Superfície/Face nem sempre é claro e facilmente representável
  - Não unicidade construtiva
  - A determinação da inclusão de um ponto em um sólido exige algoritmos não-triviais.
  - A avaliação de operações booleanas é algoritmicamente trabalhosa.

Características do B-rep

- Vantagens
  - Dados *Wireframe* podem ser facilmente derivados dos modelos B-rep baseados em grafos
  - Pode-se obter B-rep de várias outras formas de representação e criação
  - A intersecção de superfícies é armazenada explicitamente.
  - A exibição de um ponto da superfície é fácil.

Esquemas de Representação Brep Sólidos Poliédricos

B-rep Poliédrico *Foley 1996 12.5.1*

- Um B-rep Poliédrico é um B-rep simplificado onde :
  - O objeto é limitado por superfícies planares poligonais (simplificação do conceito de **superfície limitante**)
  - As faces são limitadas por arestas retas
  - As arestas são limitadas por dois vértices no R3

Poliedros (Simples) Platônicos

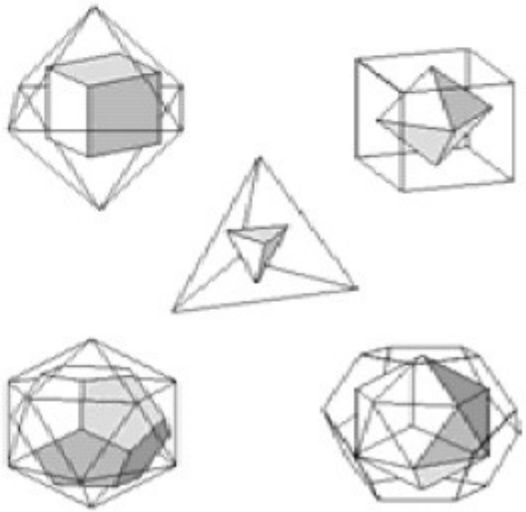
Tetraedro (4 = pirâmide)

Hexaedro (6 = cubo)

Octaedro (8)

Icosaedro (10)

Dodecaedro (12)

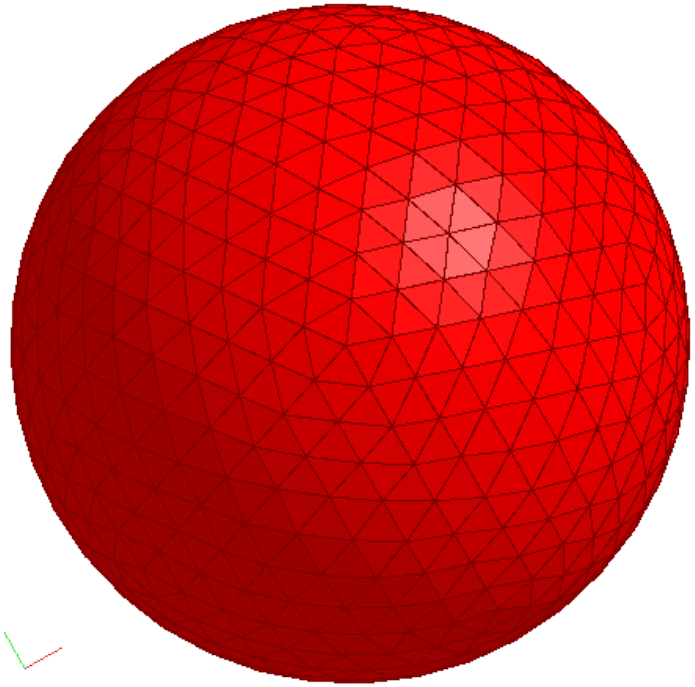


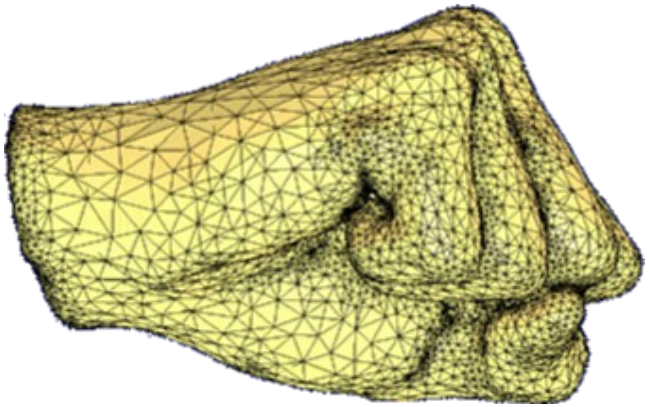
Cada poliedro platônico tem um dual onde os vértices de um é o centro da face do outro!

B-rep Poliédrico

- Objetos Complexos podem ser divididos em vários poliedros simples
- Os polígonos são a pele/casca do objeto
  - Objetos Ocos (não se representa interior)
  - Mas, polígonos têm face frontal e traseira
- Poligonalização de Objetos !!!!

Quantos triângulos?

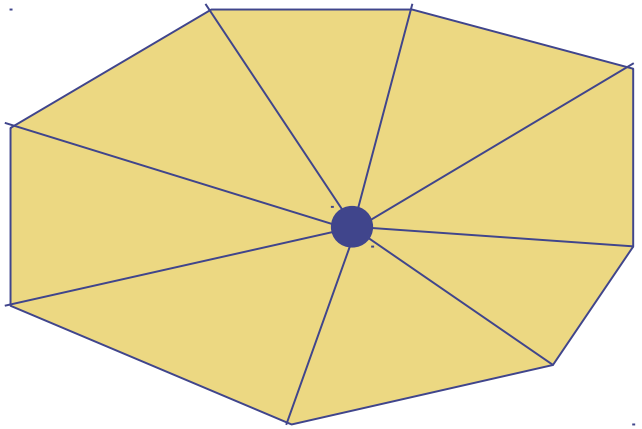




Operações Importantes sobre Malhas Poligonais

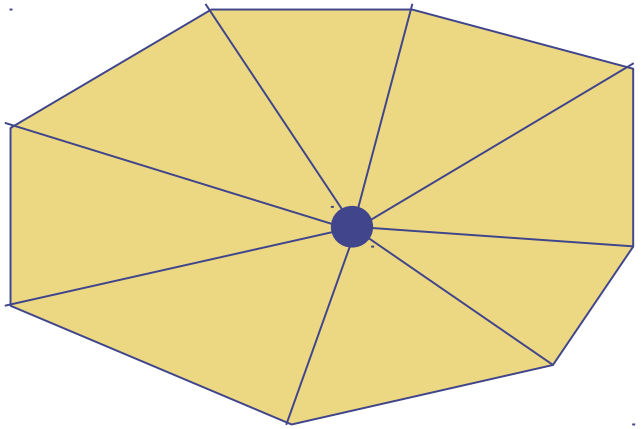
- **As GRANDES Vantagens (Velocidade) do B-rep Poligonal estão em :**
- **Visualizar**
- Identificar **adjacências**

  Quais ??????

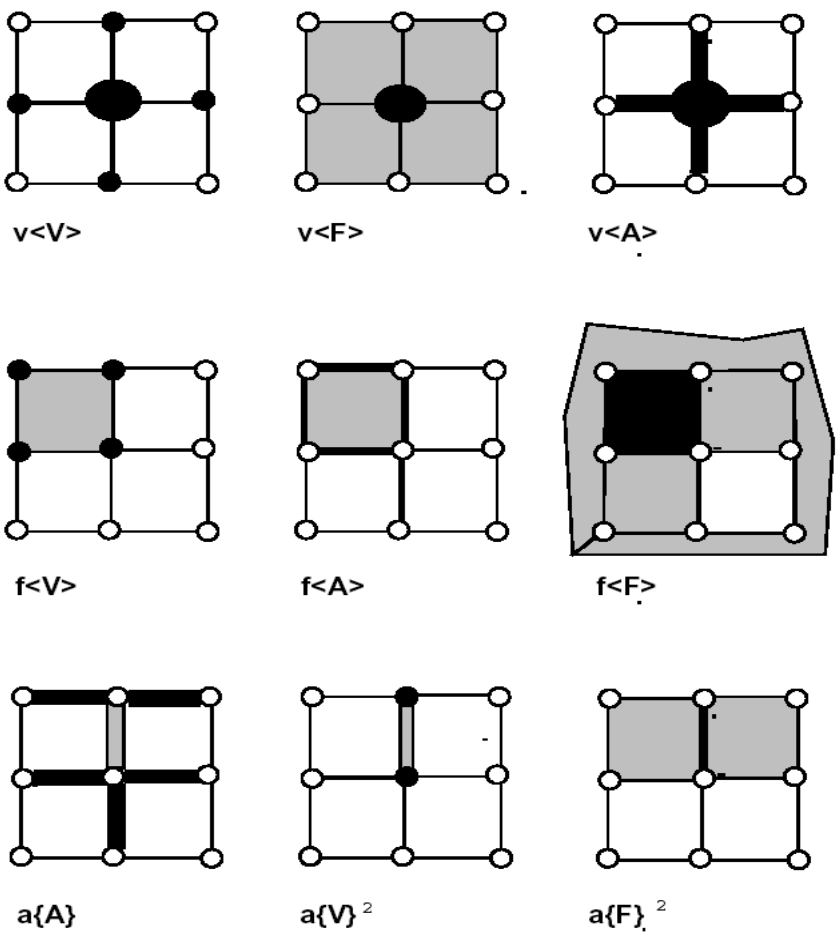


Operações Importantes sobre Malhas Poligonais

- **Visualizar** a malha de polígonos (WF)
- Identificar **adjacências**
  - Achar todas as arestas que incidem em um vértice.
  - Achar as faces que incidem numa aresta ou vértice.
  - Achar as arestas na fronteira de uma face.
  - ...



9 tipos de adjacências



Estruturas de Dados

Estrutura de Dados para B-rep

"A melhor estrutura de dados para representar um B-rep é um grafo hierárquico" (Mortenson 2006: 380)

Lembrete: "árvore e listas são casos simplificados de grafos"

B-rep Sólidos 2-*manifold*

- Codificação Explícita (CE)
- Lista de Vértices (LV)
- Lista de Arestas
  - Simples (LAS)
  - Melhorada (LAM)
- Winged-Edge (WE)
- Half-Edge (HE)

Codificação Explícita

Codificação Explícita (CE)

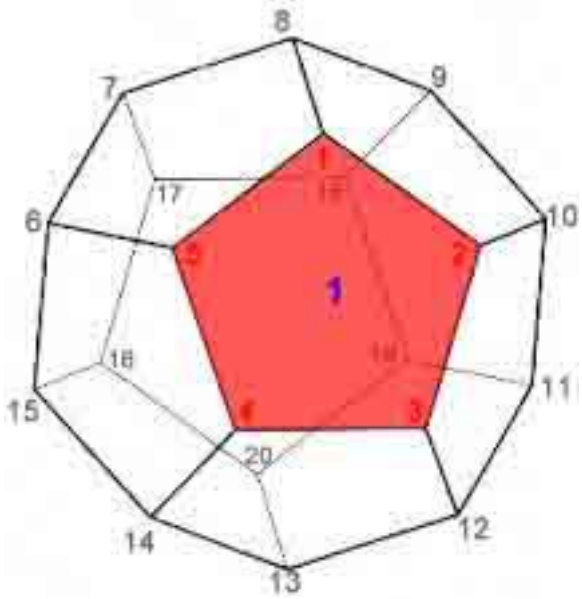
- Codifica explicitamente os polígonos da superfície fornecendo uma lista de vértices **com** suas coordenadas

Codificação explícita
$f_1 = ((x_1,y_1,z_1),(x_5,y_5,z_5),(x_2,y_2,z_2))$
$f_2 = ((x_3,y_3,z_3),(x_2,y_2,z_2),(x_5,y_5,z_5))$
$f_3 = ((x_3,y_3,z_3),(x_4,y_4,z_4),(x_5,y_5,z_5))$
$f_4 = ((x_1,y_1,z_1),(x_4,y_4,z_4),(x_5,y_5,z_5))$
$f_5 = ((x_1,y_1,z_1),(x_2,y_2,z_2),(x_3,y_3,z_3),(x_4,y_4,z_4))$

Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

DODECAEDRO
Codificação Explícita

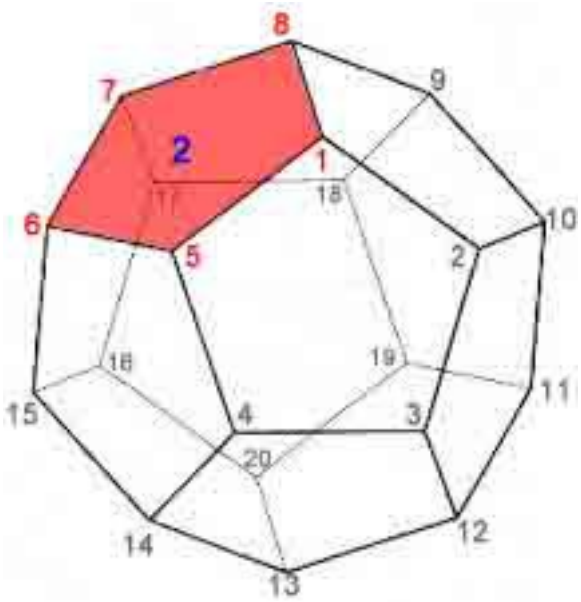


f1 = ((x1,y1,z1); (x2,y2,z2); (x3,y3,z3); (x4,y4,z4); (x5,y5,z5))
f2 = ((x7,y7,z7); (x8,y8,z8); (x1,y1,z1); (x5,y5,z5); (x6,y6,z6))
f3 = ((x9,y9,z9); (x10,y10,z10); (x2,y2,z2); (x1,y1,z1); (x8,y8,z8))
f4 = ((x11,y11,z11); (x12,y12,z12); (x3,y3,z3); (x2,y2,z2); (x10,y10,z10))
f5 = ((x13,y13,z13); (x14,y14,z14); (x4,y4,z4); (x3,y3,z3); (x12,y12,z12))
f6 = ((x15,y15,z15); (x6,y6,z6); (x5,y5,z5); (x4,y4,z4); (x14,y14,z14))
f7 = ((x6,y6,z6); (x7,y7,z7); (x17,y17,z17); (x16,y16,z16); (x15,y15,z15))
f8 = ((x8,y8,z8); (x9,y9,z9); (x18,y18,z18); (x17,y17,z17); (x7,y7,z7))
f9 = ((x10,y10,z10); (x11,y11,z11); (x19,y19,z19); (x18,y18,z18); (x9,y9,z9))
f10 = ((x12,y12,z12); (x13,y13,z13); (x20,y20,z20); (x19,y19,z19); (x11,y11,z11))
f11 = ((x14,y14,z14); (x15,y15,z15); (x16,y16,z16); (x20,y20,z20); (x13,y13,z13))
f12 = ((x20,y20,z20); (x16,y16,z16); (x17,y17,z17); (x18,y18,z18); (x19,y19,z19))

Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

DODECAEDRO
Codificação Explícita



f1 = ((x1,y1,z1); (x2,y2,z3); (x3,y3,z3); (x4,y4,z4); (x5,y5,z5))

f2 = ((x7,y7,z7); (x8,y8,z8); (x1,y1,z1); (x5,y5,z5); (x6,y6,z6))

f3 = ((x9,y9,z9); (x10,y10,z10); (x2,y2,z2); (x1,y1,z1); (x8,y8,z8))

f4 = ((x11,y11,z11); (x12,y12,z12); (x3,y3,z3); (x2,y2,z2); (x10,y10,z10))

f5 = ((x13,y13,z13); (x14,y14,z14); (x4,y4,z4); (x3,y3,z3); (x12,y12,z12))

f6 = ((x15,y15,z15); (x6,y6,z6); (x5,y5,z5); (x4,y4,z4); (x14,y14,z14))

f7 = ((x6,y6,z6); (x7,y7,z7); (x17,y17,z17); (x16,y16,z16); (x15,y15,z15))

f8 = ((x8,y8,z8); (x9,y9,z9); (x18,y18,z18); (x17,y17,z17); (x7,y7,z7))

f9 = ((x10,y10,z10); (x11,y11,z11); (x19,y19,z19); (x18,y18,z18); (x9,y9,z9))

f10 = ((x12,y12,z12); (x13,y13,z13); (x20,y20,z20); (x19,y19,z19); (x11,y11,z11))

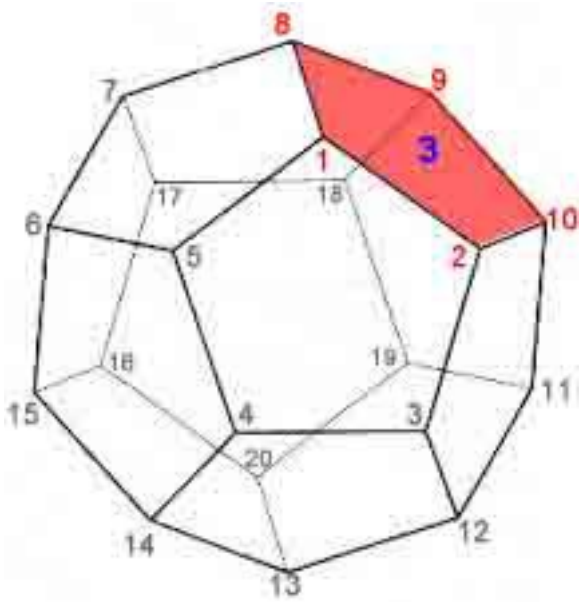
f11 = ((x14,y14,z14); (x15,y15,z15); (x16,y16,z16); (x20,y20,z20); (x13,y13,z13))

f12 = ((x20,y20,z20); (x16,y16,z16); (x17,y17,z17); (x18,y18,z18); (x19,y19,z19))

Departamento de Ciência da COMPUTAÇÃO

DODECAEDRO
Codificação Explícita

UDESC Joinville

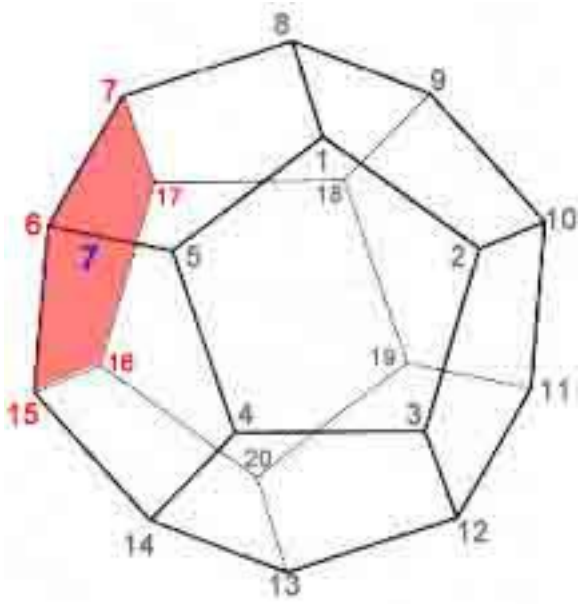


f1 = ((x1,y1,z1); (x2,y2,z3); (x3,y3,z3); (x4,y4,z4); (x5,y5,z5))
f2 = ((x7,y7,z7); (x8,y8,z8); (x1,y1,z1); (x5,y5,z5); (x6,y6,z6))
f3 = ((x9,y9,z9); (x10,y10,z10); (x2,y2,z2); (x1,y1,z1); (x8,y8,z8))
f4 = ((x11,y11,z11); (x12,y12,z12); (x3,y3,z3); (x2,y2,z2); (x10,y10,z10))
f5 = ((x13,y13,z13); (x14,y14,z14); (x4,y4,z4); (x3,y3,z3); (x12,y12,z12))
f6 = ((x15,y15,z15); (x6,y6,z6); (x5,y5,z5); (x4,y4,z4); (x14,y14,z14))
f7 = ((x6,y6,z6); (x7,y7,z7); (x17,y17,z17); (x16,y16,z16); (x15,y15,z15))
f8 = ((x8,y8,z8); (x9,y9,z9); (x18,y18,z18); (x17,y17,z17); (x7,y7,z7))
f9 = ((x10,y10,z10); (x11,y11,z11); (x19,y19,z19); (x18,y18,z18); (x9,y9,z9))
f10 = ((x12,y12,z12); (x13,y13,z13); (x20,y20,z20); (x19,y19,z19); (x11,y11,z11))
f11 = ((x14,y14,z14); (x15,y15,z15); (x16,y16,z16); (x20,y20,z20); (x13,y13,z13))
f12 = ((x20,y20,z20); (x16,y16,z16); (x17,y17,z17); (x18,y18,z18); (x19,y19,z19))

Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

DODECAEDRO
Codificação Explícita



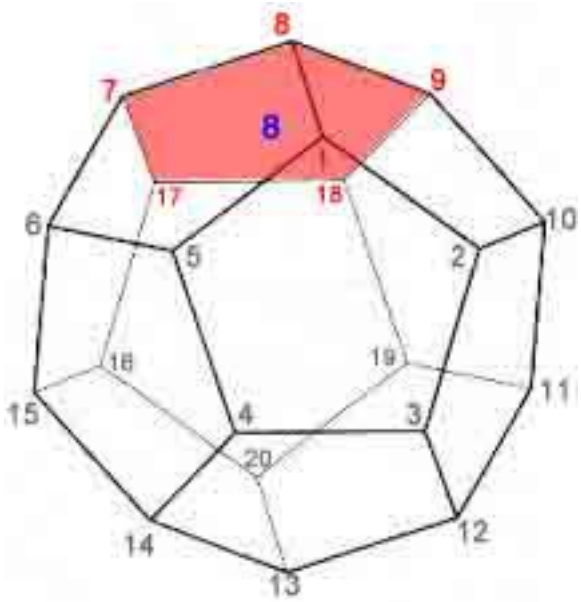
f1 = ((x1,y1,z1); (x2,y2,z3); (x3,y3,z3); (x4,y4,z4); (x5,y5,z5))
f2 = ((x7,y7,z7); (x8,y8,z8); (x1,y1,z1); (x5,y5,z5); (x6,y6,z6))
f3 = ((x9,y9,z9); (x10,y10,z10); (x2,y2,z2); (x1,y1,z1); (x8,y8,z8))
f4 = ((x11,y11,z11); (x12,y12,z12); (x3,y3,z3); (x2,y2,z2); (x10,y10,z10))
f5 = ((x13,y13,z13); (x14,y14,z14); (x4,y4,z4); (x3,y3,z3); (x12,y12,z12))
f6 = ((x15,y15,z15); (x6,y6,z6); (x5,y5,z5); (x4,y4,z4); (x14,y14,z14))
f7 = ((x6,y6,z6); (x7,y7,z7); (x17,y17,z17); (x16,y16,z16); (x15,y15,z15))
f8 = ((x8,y8,z8); (x9,y9,z9); (x18,y18,z18); (x17,y17,z17); (x7,y7,z7))
f9 = ((x10,y10,z10); (x11,y11,z11); (x19,y19,z19); (x18,y18,z18); (x9,y9,z9))
f10 = ((x12,y12,z12); (x13,y13,z13); (x20,y20,z20); (x19,y19,z19); (x11,y11,z11))
f11 = ((x14,y14,z14); (x15,y15,z15); (x16,y16,z16); (x20,y20,z20); (x13,y13,z13))
f12 = ((x20,y20,z20); (x16,y16,z16); (x17,y17,z17); (x18,y18,z18); (x19,y19,z19))

PORQUE TÁ ERRADO ???

Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

DODECAEDRO
Codificação Explícita

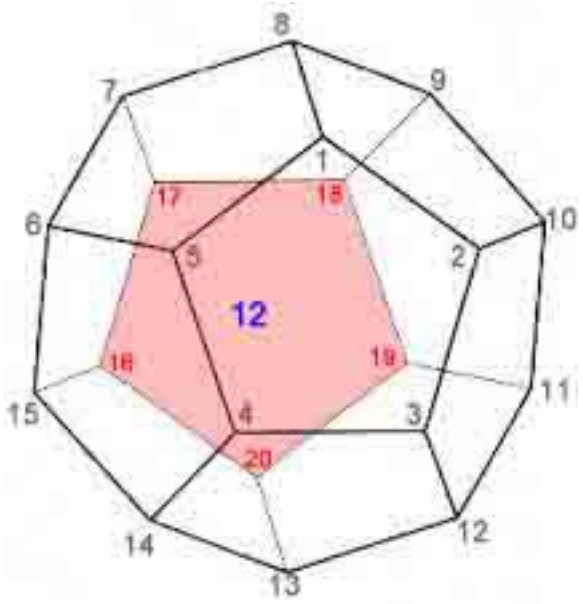


f1 = ((x1,y1,z1); (x2,y2,z3); (x3,y3,z3); (x4,y4,z4); (x5,y5,z5))
f2 = ((x7,y7,z7); (x8,y8,z8); (x1,y1,z1); (x5,y5,z5); (x6,y6,z6))
f3 = ((x9,y9,z9); (x10,y10,z10); (x2,y2,z2); (x1,y1,z1); (x8,y8,z8))
f4 = ((x11,y11,z11); (x12,y12,z12); (x3,y3,z3); (x2,y2,z2); (x10,y10,z10))
f5 = ((x13,y13,z13); (x14,y14,z14); (x4,y4,z4); (x3,y3,z3); (x12,y12,z12))
f6 = ((x15,y15,z15); (x6,y6,z6); (x5,y5,z5); (x4,y4,z4); (x14,y14,z14))
f7 = ((x6,y6,z6); (x7,y7,z7); (x17,y17,z17); (x16,y16,z16); (x15,y15,z15))
f8 = ((x8,y8,z8); (x9,y9,z9); (x18,y18,z18); (x17,y17,z17); (x7,y7,z7))
f9 = ((x10,y10,z10); (x11,y11,z11); (x19,y19,z19); (x18,y18,z18); (x9,y9,z9))
f10 = ((x12,y12,z12); (x13,y13,z13); (x20,y20,z20); (x19,y19,z19); (x11,y11,z11))
f11 = ((x14,y14,z14); (x15,y15,z15); (x16,y16,z16); (x20,y20,z20); (x13,y13,z13))
f12 = ((x20,y20,z20); (x16,y16,z16); (x17,y17,z17); (x18,y18,z18); (x19,y19,z19))

Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

DODECAEDRO
Codificação Explícita



f1 = ((x1,y1,z1); (x2,y2,z3); (x3,y3,z3); (x4,y4,z4); (x5,y5,z5))
f2 = ((x7,y7,z7); (x8,y8,z8); (x1,y1,z1); (x5,y5,z5); (x6,y6,z6))
f3 = ((x9,y9,z9); (x10,y10,z10); (x2,y2,z2); (x1,y1,z1); (x8,y8,z8))
f4 = ((x11,y11,z11); (x12,y12,z12); (x3,y3,z3); (x2,y2,z2);
(x10,y10,z10))
f5 = ((x13,y13,z13); (x14,y14,z14); (x4,y4,z4); (x3,y3,z3);
(x12,y12,z12))
f6 = ((x15,y15,z15); (x6,y6,z6); (x5,y5,z5); (x4,y4,z4);
(x14,y14,z14))
f7 = ((x6,y6,z6); (x7,y7,z7); (x17,y17,z17); (x16,y16,z16);
(x15,y15,z15))
f8 = ((x8,y8,z8); (x9,y9,z9); (x18,y18,z18); (x17,y17,z17);
(x7,y7,z7))
f9 = ((x10,y10,z10); (x11,y11,z11); (x19,y19,z19); (x18,y18,z18);
(x9,y9,z9))
f10 = ((x12,y12,z12); (x13,y13,z13); (x20,y20,z20);
(x19,y19,z19); (x11,y11,z11))
f11 = ((x14,y14,z14); (x15,y15,z15); (x16,y16,z16);
(x20,y20,z20); (x13,y13,z13))
**f12 = ((x20,y20,z20); (x19,y19,z19); (x18,y18,z18);
(x17,y17,z17); (x16,y16,z16))**

Codificação Explícita (CE)

- Vantagens:
  - Extremamente simples
- Desvantagens:
  - Tem redundância na geometria
    - Ocupa espaço de armazenamento desnecessário.
    - Operações geométricas introduzem erros numéricos independentes nas coordenadas dos vértices.
  - Nenhuma Redundância na topologia
    - não considera que os vértices são compartilhados
    - Ineficiência (cada aresta é desenhada duas vezes na visualização).
    - Não armazena informações extras quanto as adjacências

Como melhorar a CE

- Para solucionar os problemas encontrados na CE devemos eliminar os seguintes problemas
  - Evitar a replicação de vértices.
  - Codificar informações de adjacência.

Lista de Vértices
(A&C: 131)

Lista de Vértices (LV)

- Definição
  - Há uma ***lista de vértices*** armazenados separadamente (geometria)
  - Faces ***listam os vértices*** que as compõem (topologia)

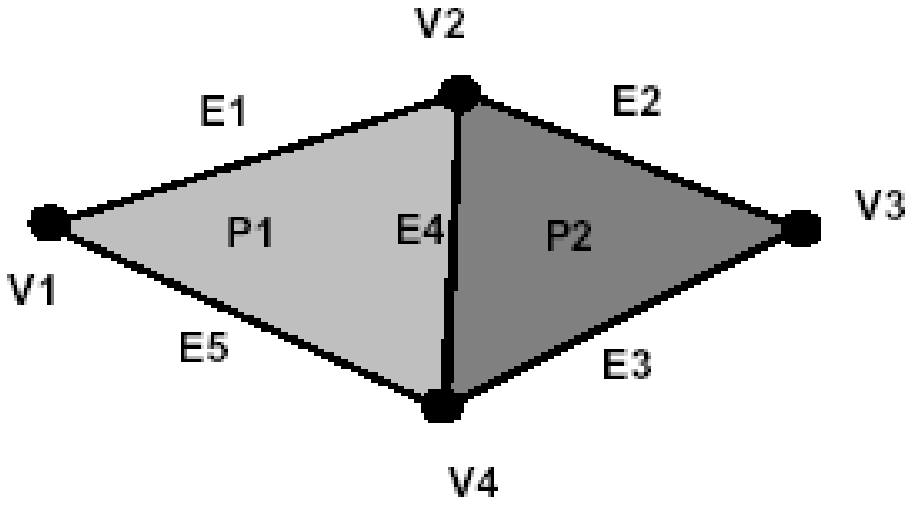
Lista de Vértices (LV)

- Numa lista de vértices cada polígono da face/superfície é definido por referência aos vértices da lista

Lista de vértices
$v_1 = (x_1, y_1, z_1)$
$v_2 = (x_2, y_2, z_2)$
$v_3 = (x_3, y_3, z_3)$
$v_4 = (x_4, y_4, z_4)$
$v_5 = (x_4, y_4, z_4)$

Lista de faces
$f_1 = (v_1, v_5, v_2)$
$f_2 = (v_3, v_2, v_5)$
$f_3 = (v_3, v_4, v_5)$
$f_4 = (v_1, v_4, v_5)$
$f_5 = (v_1, v_2, v_3, v_4)$

Lista de Vértices (LV)



- $V = \{V_1 = (x_1, y_1, z_1), V_2 = (x_2, y_2, z_2), V_3 = (x_3, y_3, z_3), V_4 = (x_4, y_4, z_4)\}$;
- $P_1 = \{V_1, V_2, V_4\}$;
- $P_2 = \{V_4, V_2, V_3\}$.

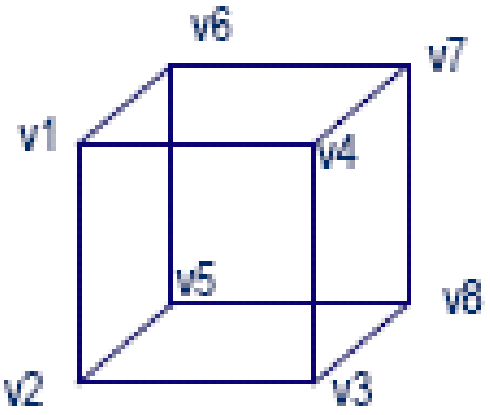
P1 e P2 estão orientados no sentido horário !

Lista de Vértices

versus

Wireframe

Wireframe x Lista de Vértices



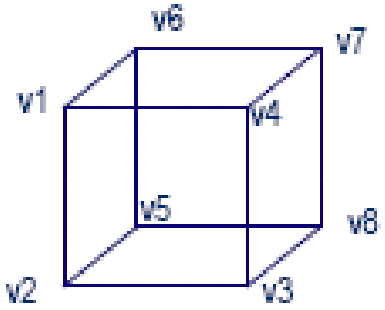
WF

Vértices={ v1,...,v8}
Arestas={v1-v2,v2-v3,...,v1-v6}

LV

Vértices={ v1,...,v8}
Faces={(v1,v2,v3,v4), ...,(v2,v5,v8,v3)}

Wireframe x Lista de Vértices



WF

Vértices={ v1,...,v8}
Arestas={v1-v2,v2-v3,...,v1-v6}

LV

Vértices={ v1,...,v8}
Faces={(v1,v2,v3,v4), ...,(v2,v5,v8,v3)}

WF = tem as arestas prontamente listadas mas não reconhece o nível da face

LV = Incorpora o conceito de face na estrutura mas, não armazena/considera as arestas (não estão prontamente disponíveis)

B-rep Poliédricos em **VRML** é uma Lista de Vértices (LV)

```
Shape { appearance Appearance{
                material Material { } }
        geometry IndexedFaceSet {
          coord Coordinate {
            point [ -1.0  1.0   1.0,    1.0   1.0  1.0,
                     1.0  1.0  -1.0,   -1.0   1.0 -1.0,
                    -1.0 -1.0   1.0,    1.0  -1.0  1.0,
                     1.0  -1.0 -1.0,   -1.0  -1.0 -1.0] }
          coordIndex [ 0, 1, 2, 3, -1,  7, 6, 5, 4, -1,
                       0, 4, 5, 1, -1, 1, 5, 6, 2, -1,
                       2, 6, 7, 3, -1, 3, 7, 4, 0] } }
```

Lista de Vértices (LV) Vantagens

- Proporciona **maior economia de memória** evitando redundância dos valores geométricos dos vértices (que o esquema anterior)
- (Ainda) é um esquema **simples e rápido**
- Ao alterar as coordenadas de um vértice, todos os polígonos nele incidentes são alterados automaticamente

Lista de Vértices (LV) Desvantagens

- Achar **adjacências** é complicado
- As arestas são desenhadas **duas** vezes
- É difícil determinar os polígonos que compartilham uma aresta a menos que por um mecanismo exaustivo de **busca**

Lista de Arestas Simples

(A&C: 132)

Lista de Arestas Simples (LAS)

- Acrescentamos uma lista de arestas definida por pares de referências à lista de vértices.
- A lista de faces é definida por referências às arestas que as definem, descritas na lista de arestas.
- Sem nenhuma redundância

Lista de vértices
$v_1 = (x_1, y_1, z_1)$
$v_2 = (x_2, y_2, z_2)$
$v_3 = (x_3, y_3, z_3)$
$v_4 = (x_4, y_4, z_4)$
$v_5 = (x_4, y_4, z_4)$

Lista de arestas
$e_1 = v_1, v_2$
$e_2 = v_2, v_3$
$e_3 = v_3, v_4$
$e_4 = v_4, v_1$
$e_5 = v_1, v_5$
$e_6 = v_2, v_5$
$e_7 = v_3, v_5$
$e_8 = v_4, v_5$

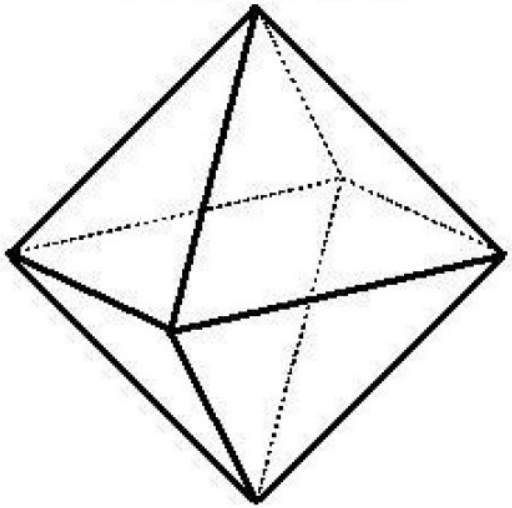
Lista de faces
$f_1 = e_1, e_5, e_6$
$f_2 = e_2, e_6, e_7$
$f_3 = e_3, e_7, e_8$
$f_4 = e_4, e_8, e_5$
$f_5 = e_1, e_2, e_3, e_4$

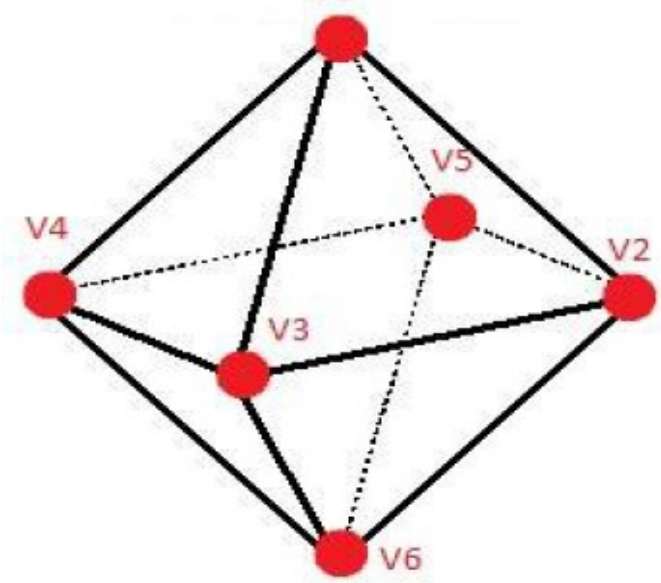
Octaedro

- 6 Vertices

-12 Arestas

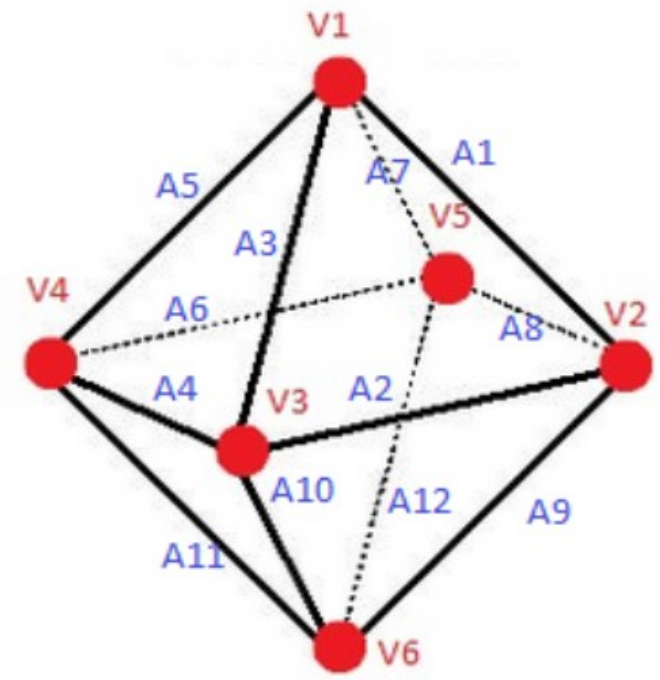
- 8 Faces





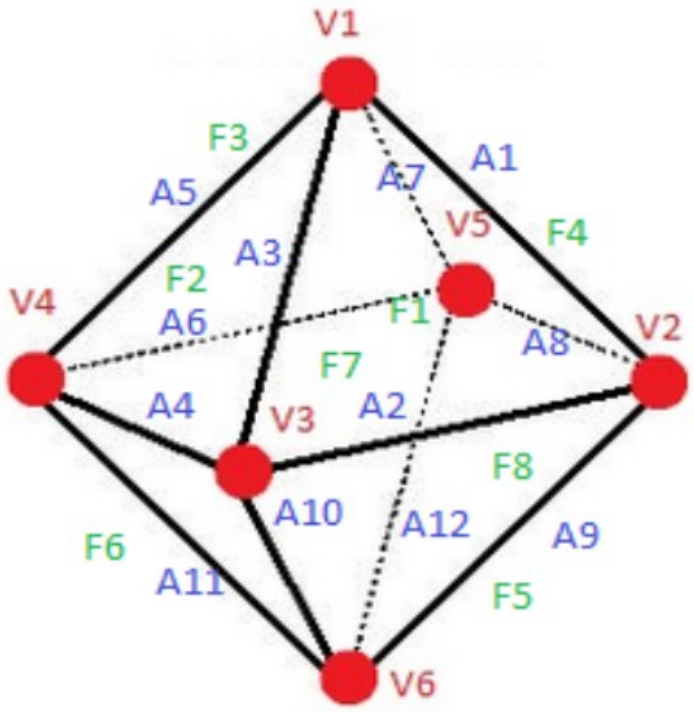
VERTICES	COORDENADAS
V1	X1, Y1, Z1
V2	X2, Y2, Z2
V3	X3, Y3, Z3
V4	X4, Y4, Z4
V5	X5, Y5, Z5
V6	X6, Y6, Z6

Octaedro em LAS



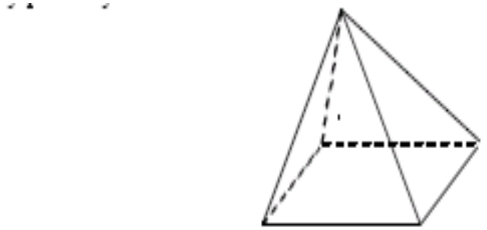
ARESTAS	VERTICES
A1	V1, V2
A2	V2, V3
A3	V3, V1
A4	V3, V4
A5	V4, V1
A6	V4, V5
A7	V5, V1
A8	V5, V2
A9	V2, V6
A10	V6, V3
A11	V6, V4
A12	V6, V5

Octaedro em LAS

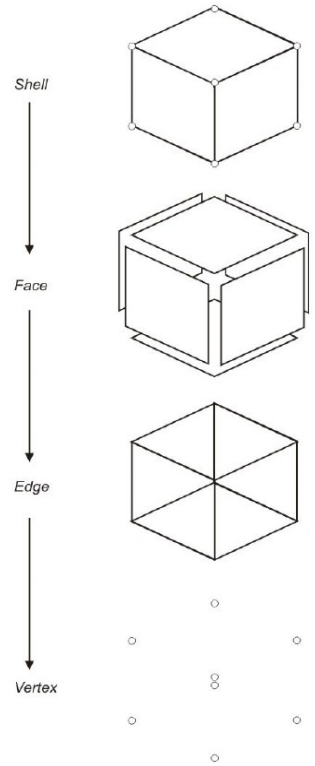


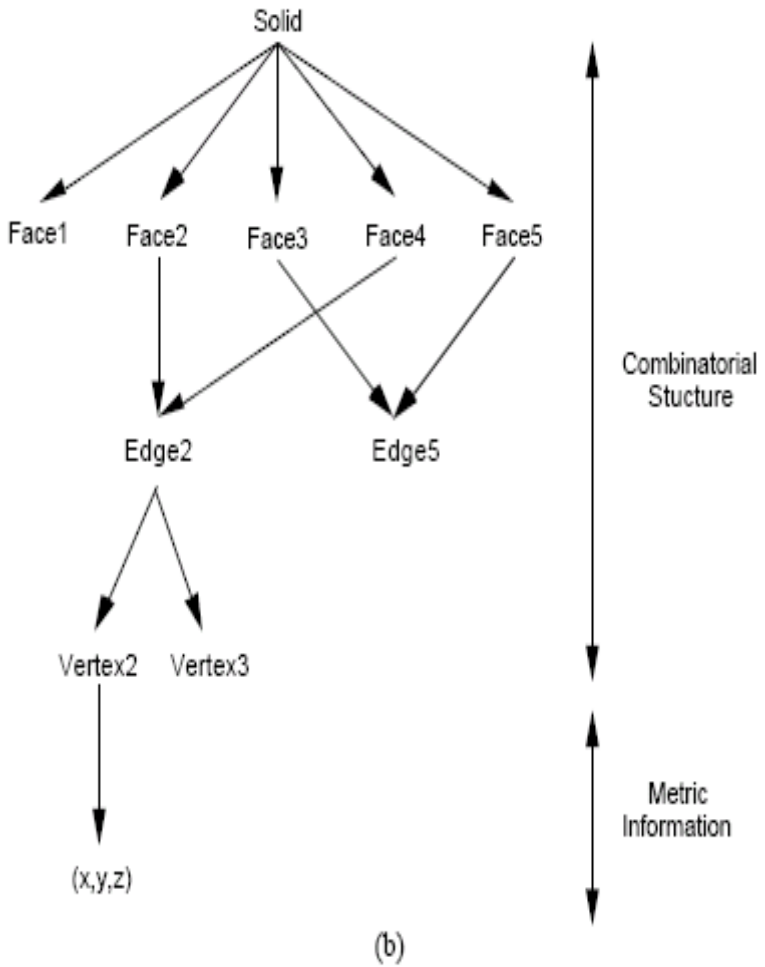
FACES	ARESTAS
F1	A1, A2, A3
F2	A3, A4, A5
F3	A5, A6, A7
F4	A7, A8, A1
F5	A9, A10, A2
F6	A10, A11, A4
F7	A11, A12, A6
F8	A12, A9, A8

Hierarquia dos Elementos Topológicos



(a)





(b)

Lista de Arestas Melhorada

Lista de Arestas Melhorada (LAM)

- Acrescentar na lista de arestas informações sobre as faces adjacentes a uma aresta (em um número fixo de 2)

Lista de arestas
$e_1 = v_1, v_2, f_1, f_5$
$e_2 = v_2, v_3, f_3, f_5$
$e_3 = v_3, v_4, f_2, f_5$
$e_4 = v_4, v_1, f_4, f_5$
$e_5 = v_1, v_5, f_1, f_4$
$e_6 = v_2, v_5, f_1, f_3$
$e_7 = v_3, v_5, f_2, f_5$
$e_8 = v_4, v_5, f_2, f_4$

Lista de Arestas Melhorada (LAM)

Definição:

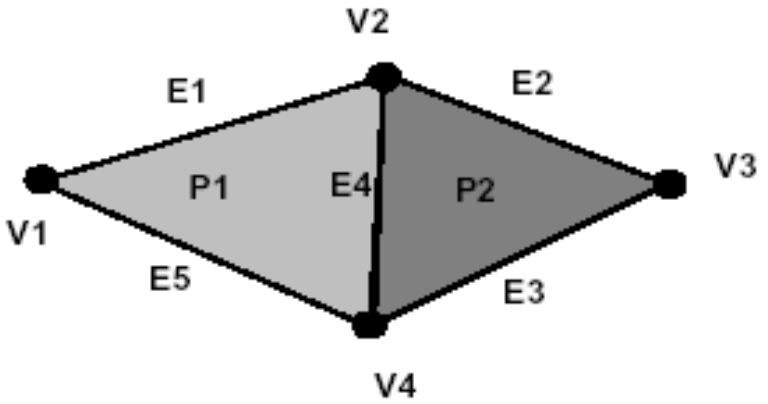
- Tem-se uma ***lista de arestas*** (sem repeti-las) que indicam os vértices que as compõem

Faces apontam para a ***lista de arestas*** e cada aresta inclui referência (de volta) para as duas faces que compartilham uma aresta (redundância)

- Facilita entretanto, a determinação das duas faces incidentes na aresta
- Arestas são prontamente desenhadas percorrendo-se a lista de arestas (WF embutido)

Lista de Arestas Melhorada (LAM)

- $V = \{V_1 = (x_1, y_1, z_1), V_2 = (x_2, y_2, z_2), V_3 = (x_3, y_3, z_3), V_4 = (x_4, y_4, z_4)\}$;
- $E_1 = \{V_1, V_2, P_1, \lambda\}$;
- $E_2 = \{V_2, V_3, P_2, \lambda\}$;
- $E_3 = \{V_3, V_4, P_2, \lambda\}$;
- $E_4 = \{V_2, V_4, P_1, P_2\}$;
- $E_5 = \{V_4, V_1, P_1, \lambda\}$;
- $P_1 = \{E_1, E_4, E_5\}$;
- $P_2 = \{E_2, E_3, E_4\}$.



CONVENÇÕES
- A aresta é orientada
- O primeiro índice indica face à direita da aresta, o segundo, à esquerda

Lista de Arestas Melhorada (LAM)

- Propriedades
  - Tem-se acesso a todas as arestas sem precisar percorrer as fronteiras dos polígonos.
  - As arestas que incidem em um vértice podem ser obtidas através de uma combinação de algoritmos geométricos e de **busca**

Aresta Alada

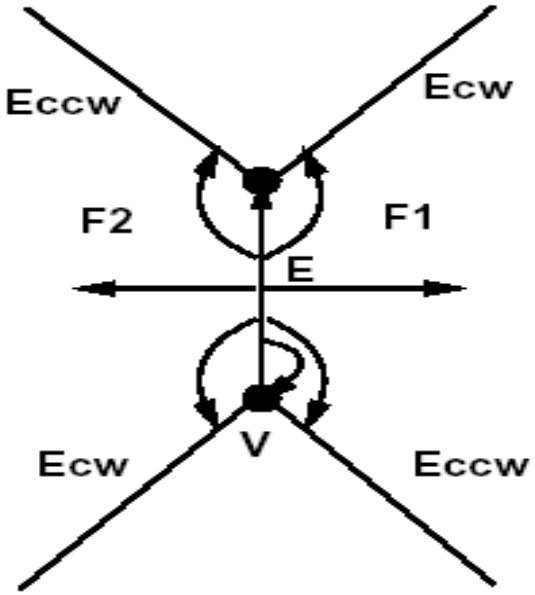
Winged-Edge

(A&C: 133)

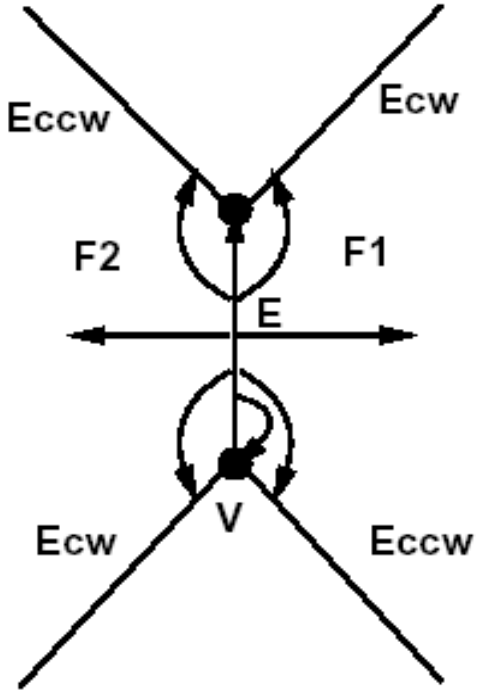
(Foley: seção 12.5.2)

Winged-Edge (WE)

- Criada entre 1972 e 1975 por Baumgart
- Esquema lembra uma asa (*wing*)
- É um marco na representação B-rep



A ArestaAlada (*Winged-Edge*)



Usa **8** índices (tamanho fixo, veja 8 setas na figura ao lado):

1- Vértice inicial (Vi)

2- Vértice final (Vf)

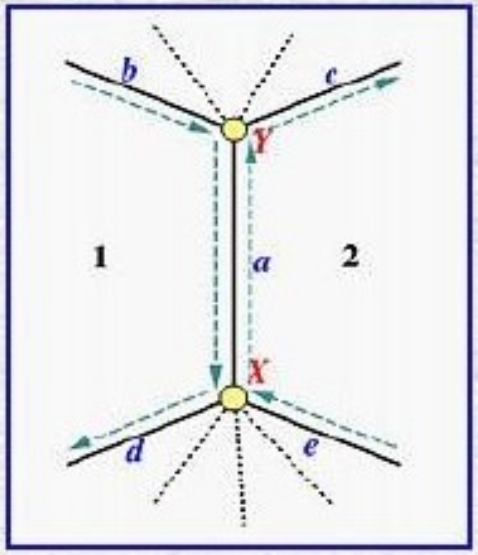
3- Face 1 (à direita)

4- Face 2 (à esquerda)

5,6- As duas WE que emanam de Vi

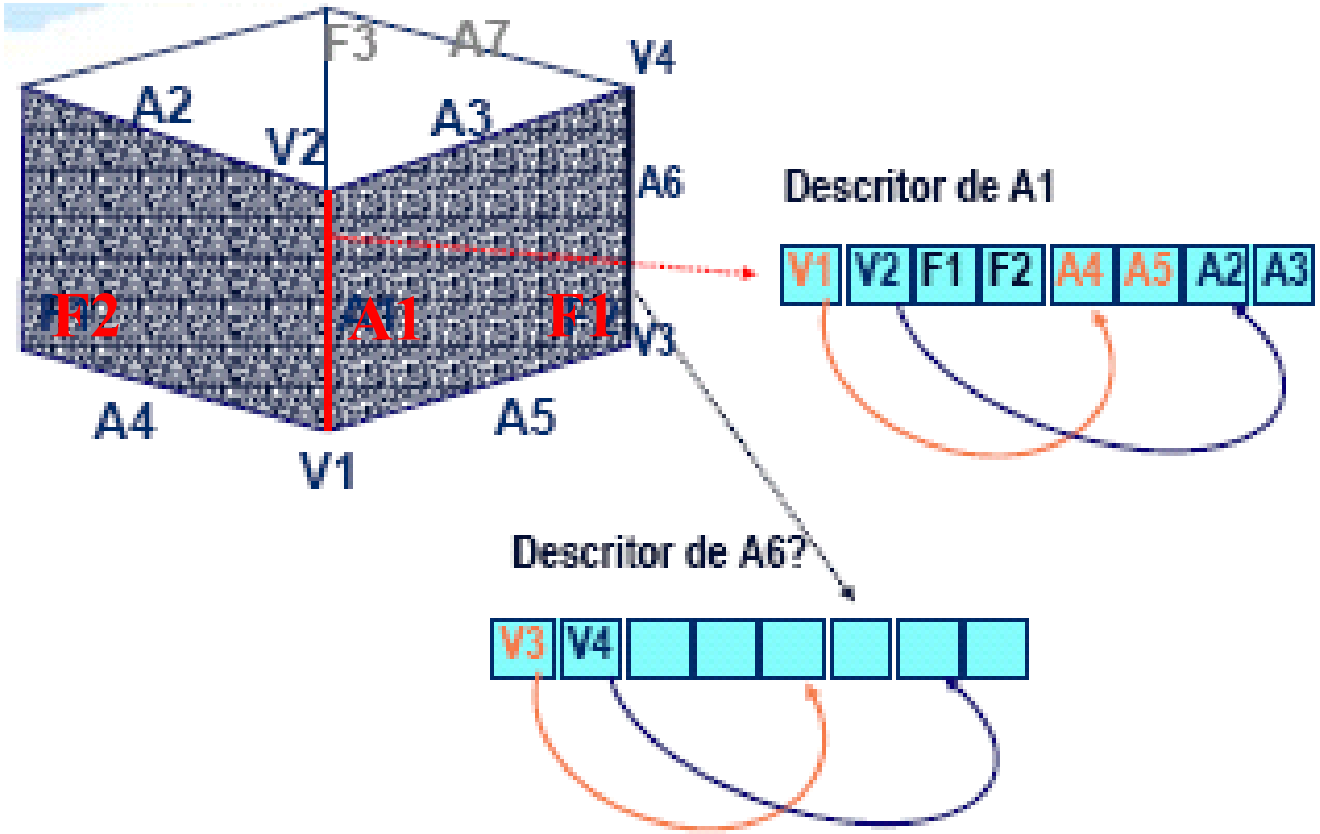
7,8-As duas WE que emanam de Vf

Exemplo de *Winged-Edge*



Edge	Vertices		Faces		Left Traverse		Right Traverse	
Name	Start	End	Left	Right	Pred	Succ	Pred	Succ
a	X	Y	1	2	b	d	e	c

Exemplo de *Winged-Edge*

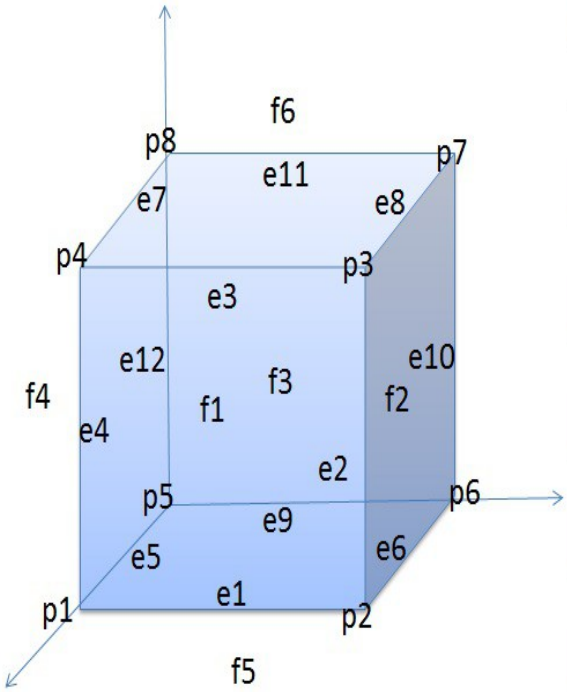


Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Aluno: Carlos Alberto Felippi

Winged-Edge ex: Hexaedro



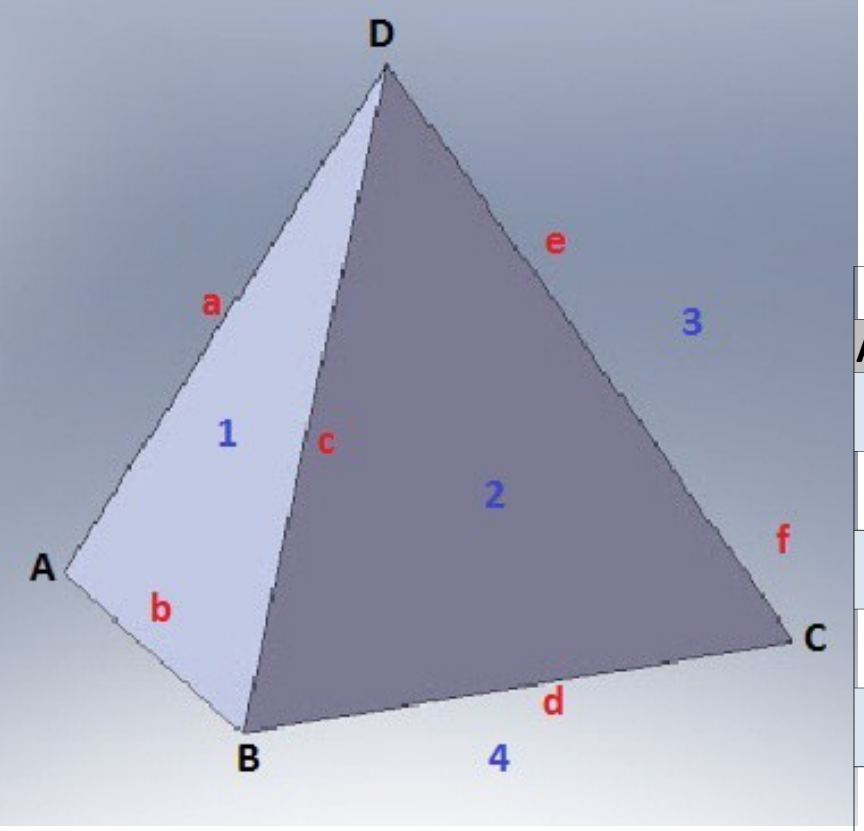
Aresta	Vértices		Faces		Esquerda		Direita	
ID	Inicio	Fim	Esq.	Dir.	Pred	Suc	Pred	Suc
e1	p1	p2	f1	f5	e4	e2	e6	e5
e2	p2	p3	f1	f2	e1	e3	e7	e6
e3	p3	p4	f1	f6	e2	e4	e8	e7
e4	p4	p1	f1	f4	e3	e1	e5	e8
e5	p1	p5	f5	f4	e1	e9	e12	e4
e6	p2	p6	f2	f5	e2	e10	e9	e1
e7	p3	p7	f6	f2	e3	e11	e10	e2
e8	p4	p8	f4	f6	e4	e12	e11	e3
e9	p5	p6	f5	f3	e5	e6	e10	e12
e10	p6	p7	f2	f3	e6	e7	e11	e9
e11	p7	p8	f6	f3	e7	e8	e12	e10
e12	p8	p5	f4	f3	e8	e5	e9	e11

Face	af
f1	e1
f2	e2
f3	e10
f4	e12
f5	e1
f6	e8

Vértice	(x,y,z)	av
p1	(1,0,0)	e1
p2	(1,1,0)	e2
p3	(1,1,1)	e3
p4	(1,0,1)	e4
p5	(0,0,0)	e9
p6	(0,1,0)	e10
p7	(0,1,1)	e11
p8	(0,0,1)	e12

Estrutura de Dados WE

- É composta de 3 elementos
  - A **Aresta** Alada (como vista antes)
  - Uma lista de **vértices** que têm um ponteiro para uma de suas ArestasAladas
  - Uma lista de **faces** que têm um ponteiro para uma de suas ArestasAladas
- Convenção: Primeiro índice da face é a da direita



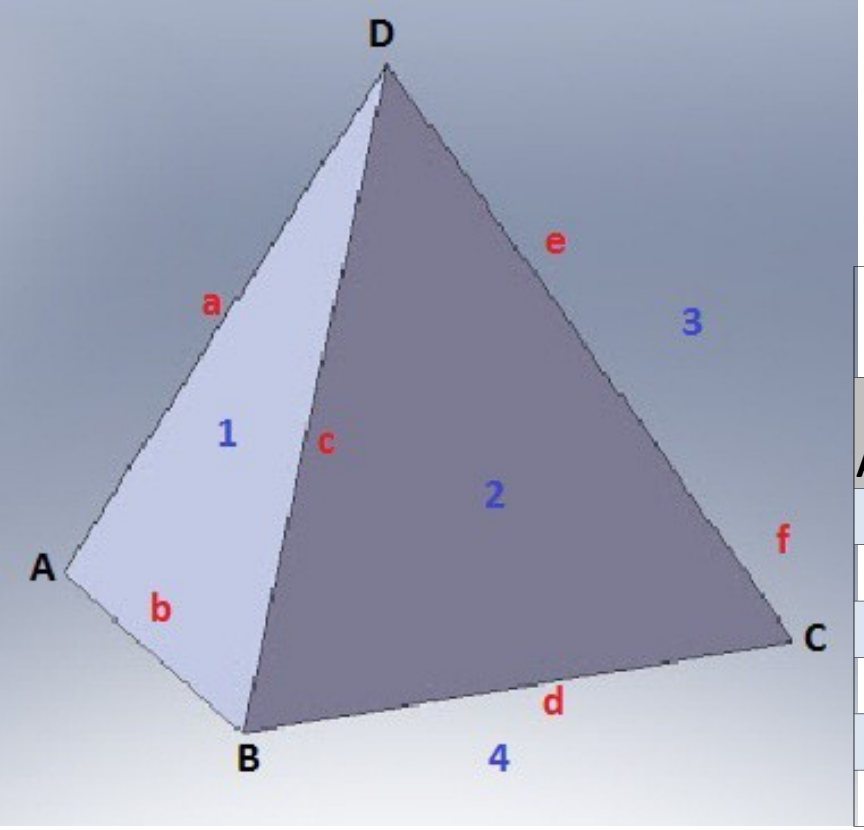
UDESC Joinville

Lista de Arestas

	Vértices		Faces		Inicial		Final	
Aresta	Início	Fim	Dir	Esq	Dir	Esq	Dir	Esq
a								
b								
c								
d								
e								
f								

Lista de vértices	
Vértice	Posição (x,y,z)
A	(1,1,1)
B	(2,3,5)
C	(3,4,5)
D	(4,5,6)

Lista de faces	
Face	Arestas
1	acb
2	ced
3	eaf
4	fbd



UDESC Joinville

Lista de Arestas

	Vértices		Faces		Inicial		Final	
Aresta	Início	Fim	Dir	Esq	Dir	Esq	Dir	Esq
a	A	D	1	3	b	f	c	e
b	A	B	4	1	f	a	d	c
c	B	D	2	1	d	b	e	a
d	B	C	4	2	b	c	f	e
e	C	D	3	2	f	d	a	c
f	A	C	3	4	a	b	e	d

Lista de vértices	
Vértice	Posição (x,y,z)
A	(1,1,1)
B	(2,3,5)
C	(3,4,5)
D	(4,5,6)

Lista de faces	
Face	Arestas
1	acb
2	ced
3	eaf
4	fbd

Winged Edge

- Armazena mais informações topológicas
- Facilita verificação de consistência (validação topológica)
- Armazena informação na estrutura associada às arestas com número pequeno de campos (compacto) e fixo
- Permite obter todos os 9 tipos de adjacências entre vértices, arestas e faces

Winged Edge

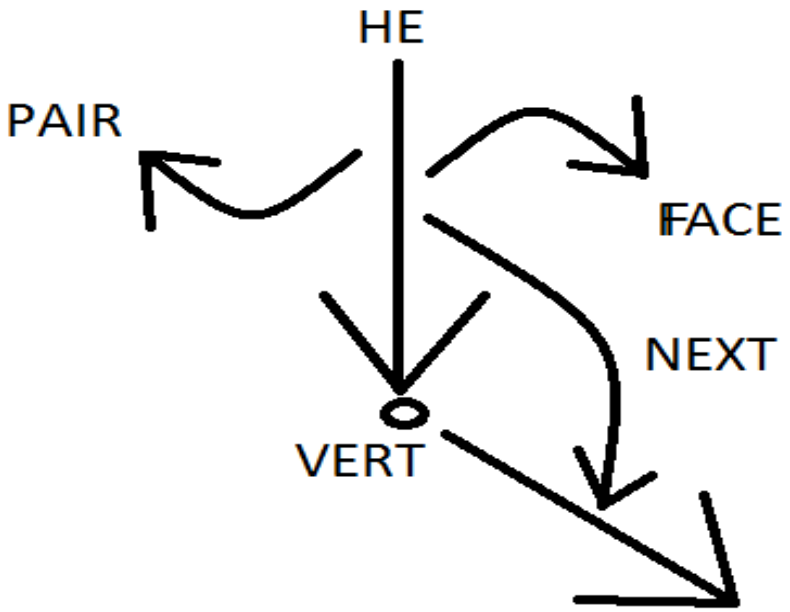
- Permite determinar quais faces ou vértices estão adjacentes à aresta em tempo constante *mas, outros tipos de adjacências podem requerer um processamento maior*
- Atualizada com o uso de operadores de Euler
- Foi concebida para faces que não têm buracos, *a priori.*

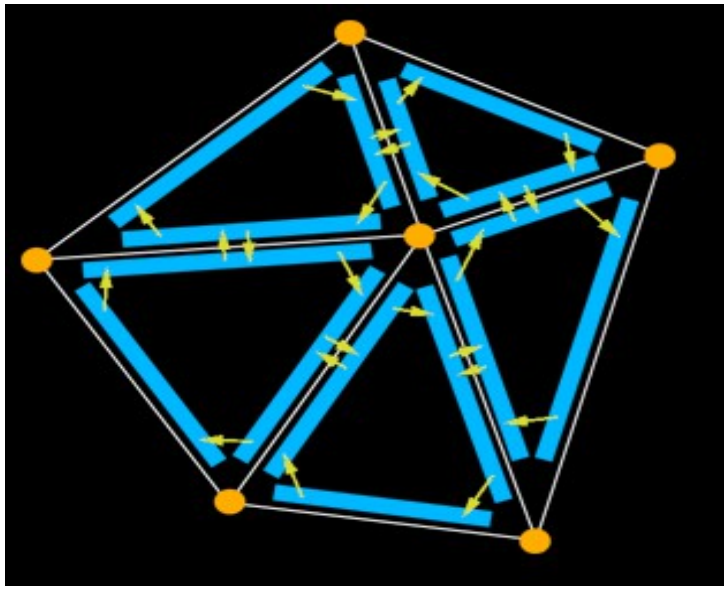
Meia Aresta
Half-Edge
(A&C: 133)

Half-Edge

- Definida por Marti Mantylla, 1988
- Mais sofisticada que WE
- Permite todas as 9 adjacências serem feitas em tempo constante
  - Tempo constante para cada pedaço de informação: quando calculando todas as arestas adjacentes a um vértice, a operação cresce linear com o número de arestas adjacentes ao vértice mas, um tempo constante por aresta
- Tamanho finito para as estruturas de faces, arestas e vértices, e compacto
- Contempla algumas redundâncias

Half-Edge



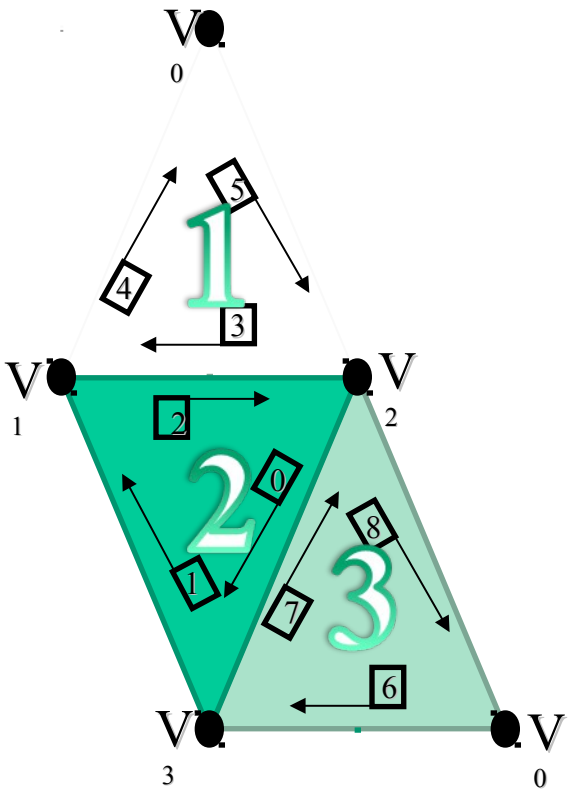


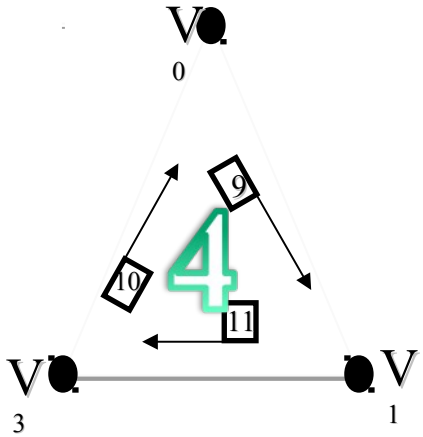
4 apontadores para cada HE

- Vértice onde chega
- Próxima HE
- Face a que pertence no *loop* (ver convenção de direção)
- Par

Composição

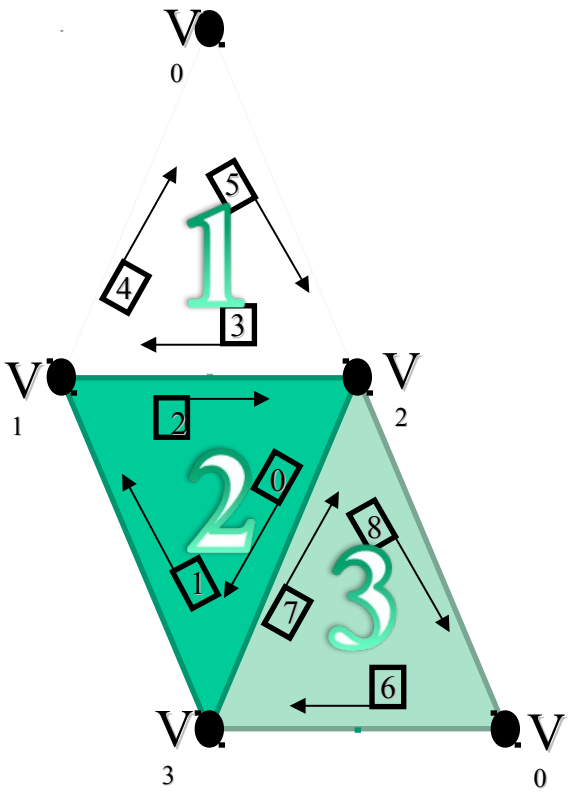
Parte Superior
Base





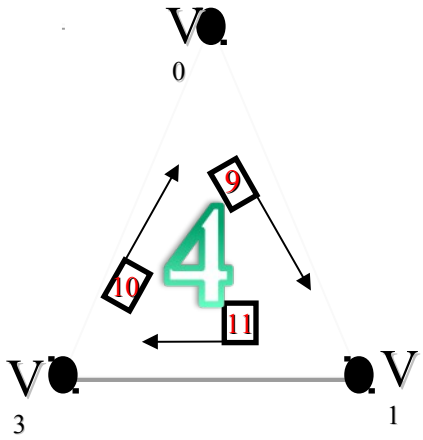
Face	HalfEdge
1	A3, A4, A5
2	A0,A1,A2
3	A6,A7,A8
4	A1,A10,A9

Parte Superior



Aresta Chega	Face	Vértice	Aresta Próximo	Par

Base



Aresta Chega	Face	Vértice	Aresta Próximo	Par

Estrutura de Dados HE

- É composta de 3 elementos
  - A **Half-Edge** (como vista antes)
  - Uma lista de **vértices** que têm um ponteiro para uma de suas Meia-Arestas
  - Uma lista de **faces** que têm um ponteiro para uma de suas Half-Edge
- Convenção: Primeiro índice da face é a da direita

Conclusões sobre B-rep

Prós e Contras de Representações B-rep

- É fácil exibir/determinar um ponto sobre a superfície do objeto
- Numa B-rep as interseções (operações) estão representadas explicitamente (não se armazenam as intenções de projeto)
- É difícil determinar, dado um ponto, se ele está no interior, fronteira ou exterior do objeto
- Operações *booleanas* são complicadas

Característica da Modelagem B-rep (Poliédrica)

- Domínio: **amplo**
- Precisão**: aproximado**
  - **se poliédrico que é o mais comum**
- **Finito**
  - poucos elementos/dados de caracterização
- **Fechado** (com esforço computacional !)
- **Não** ambíguo

Característica da Modelagem B-rep (Poliédrica)

- Eficiência
  - Para visualizar – OK
  - Para garantir validade – NÃO OK
- **Não** Conciso (*verbose*)
- Forma de Interação – **não** intuitiva
- **Não** único

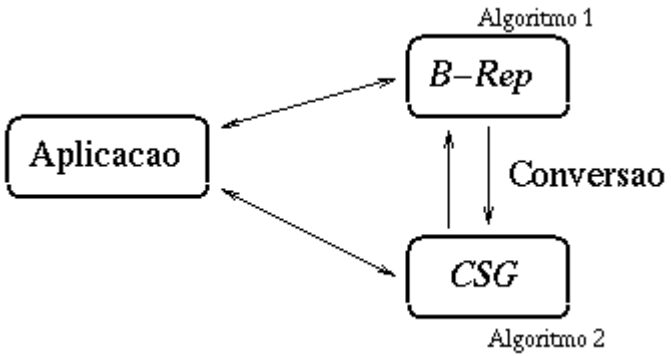
Conversão entre Representações

Conversão entre Representações

Alguns algoritmos são mais facilmente implementados em uma certa representação.

Caso exista a conversão entre representações, cada algoritmo pode se valer da representação mais apropriada.

A conversão se encarrega de manter ambas as representações atualizadas.



Detalhes Conclusivos

- O método de modelagem (interface) não restringe o modelo de representação e vice-versa.
- Um determinado *software* pode ter uma interface rica em meios de modelagem e usar uma única modelagem de representação interna.