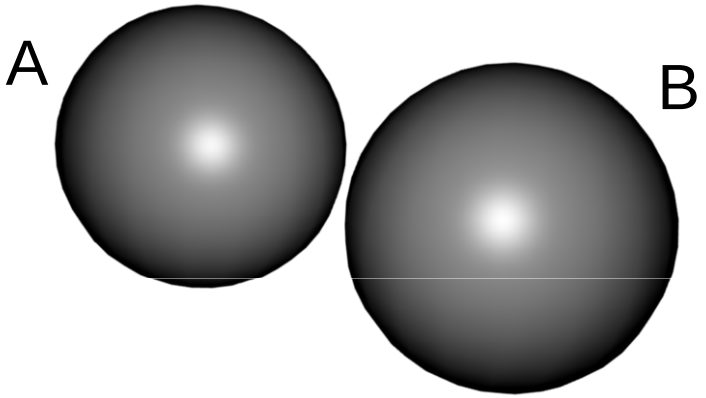
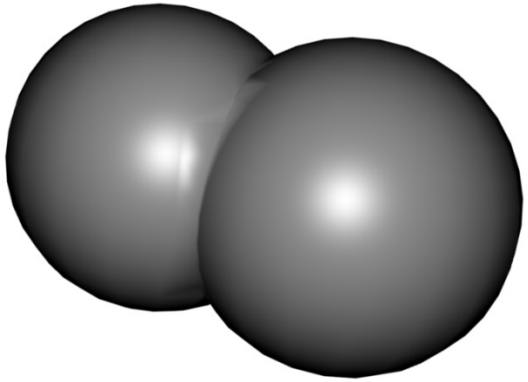
Unit 4
Boolean Operations

Carlos Andújar

Feb 2014

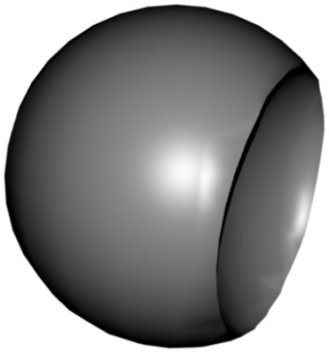
Boolean operations





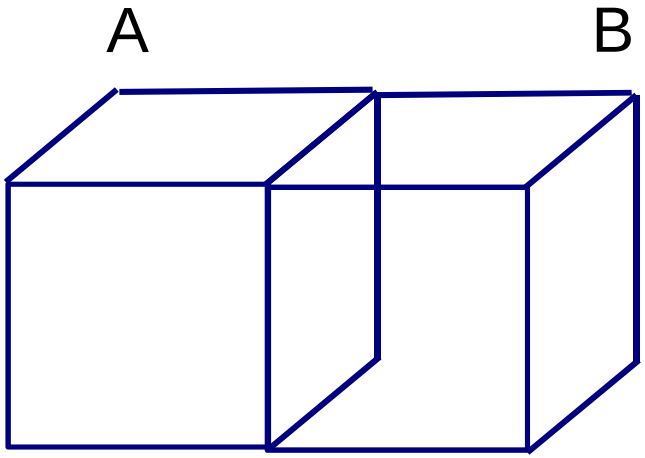
A union B

A intersection B

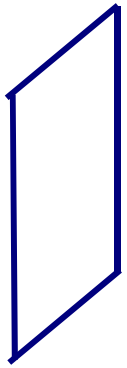


A-B

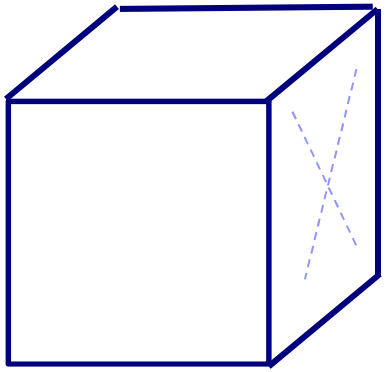
Coincidences problem



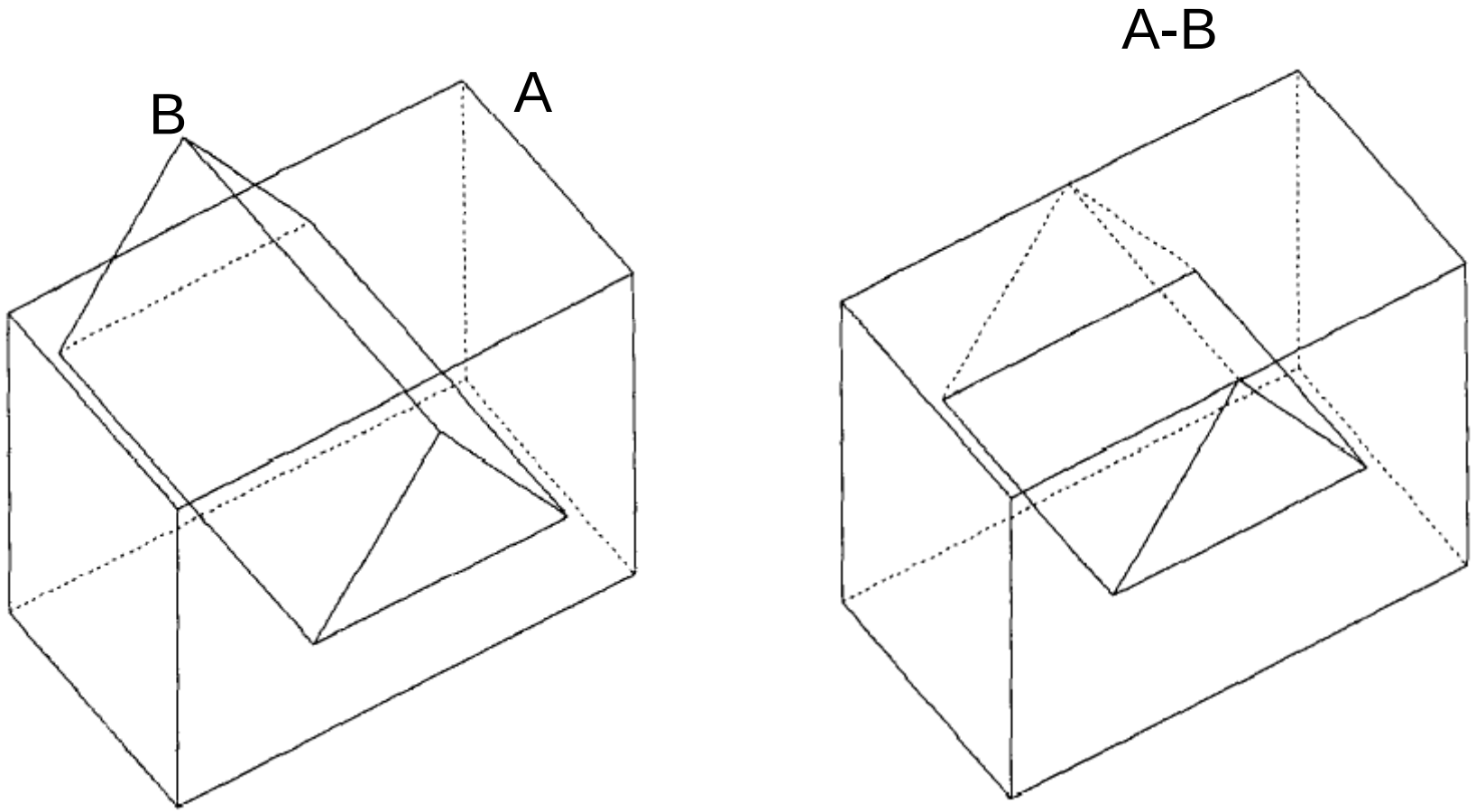
A intersection B



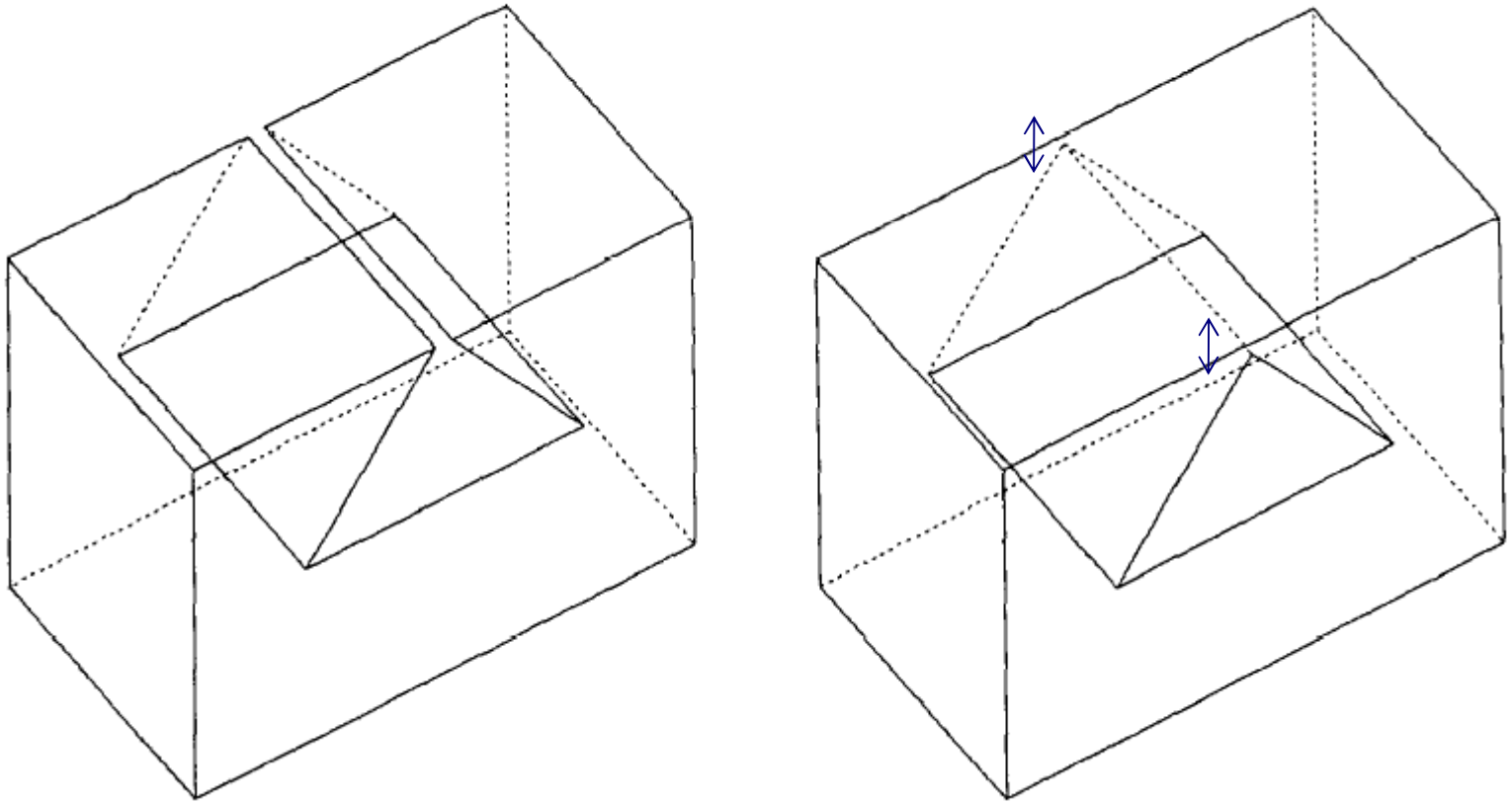
A-B



Coincidences problem



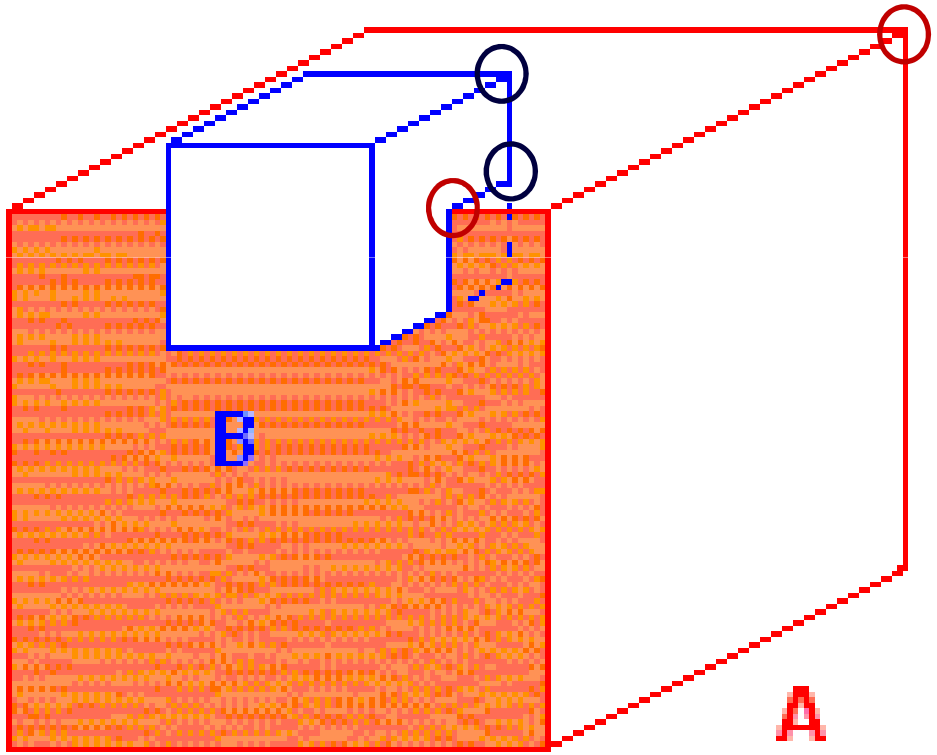
Pseudomanifolds



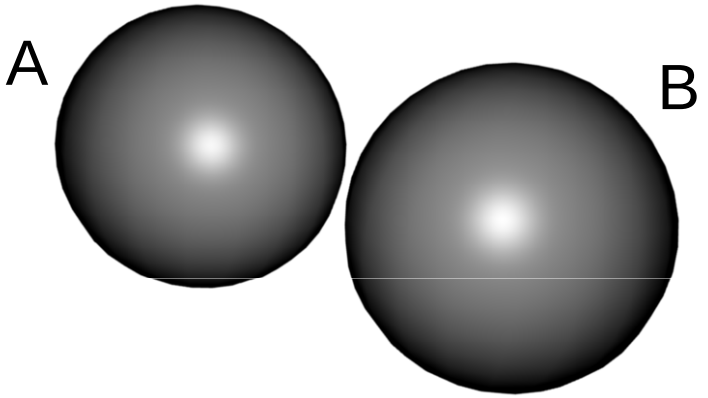
Algorithms for Boolean ops

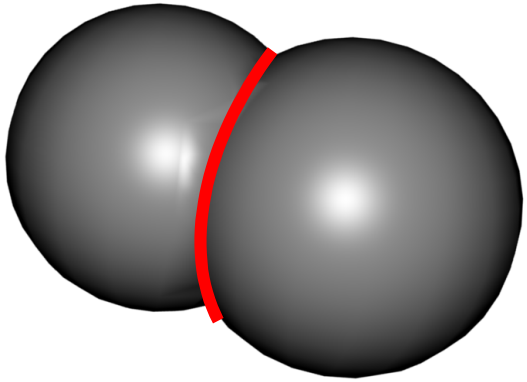
- Based on face classification → not discussed
- Based on vertex classification → discussed in depth

Algorithm 2 (classif. de *vèrtexs)*

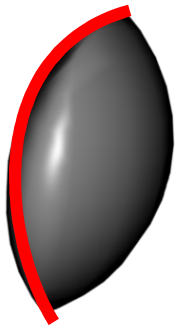


Algorithm 2 (vertex classif)

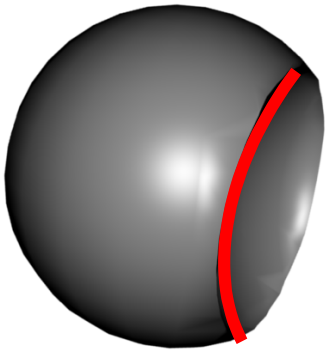




A unió B



A intersecció B



A-B

Algorithm 2

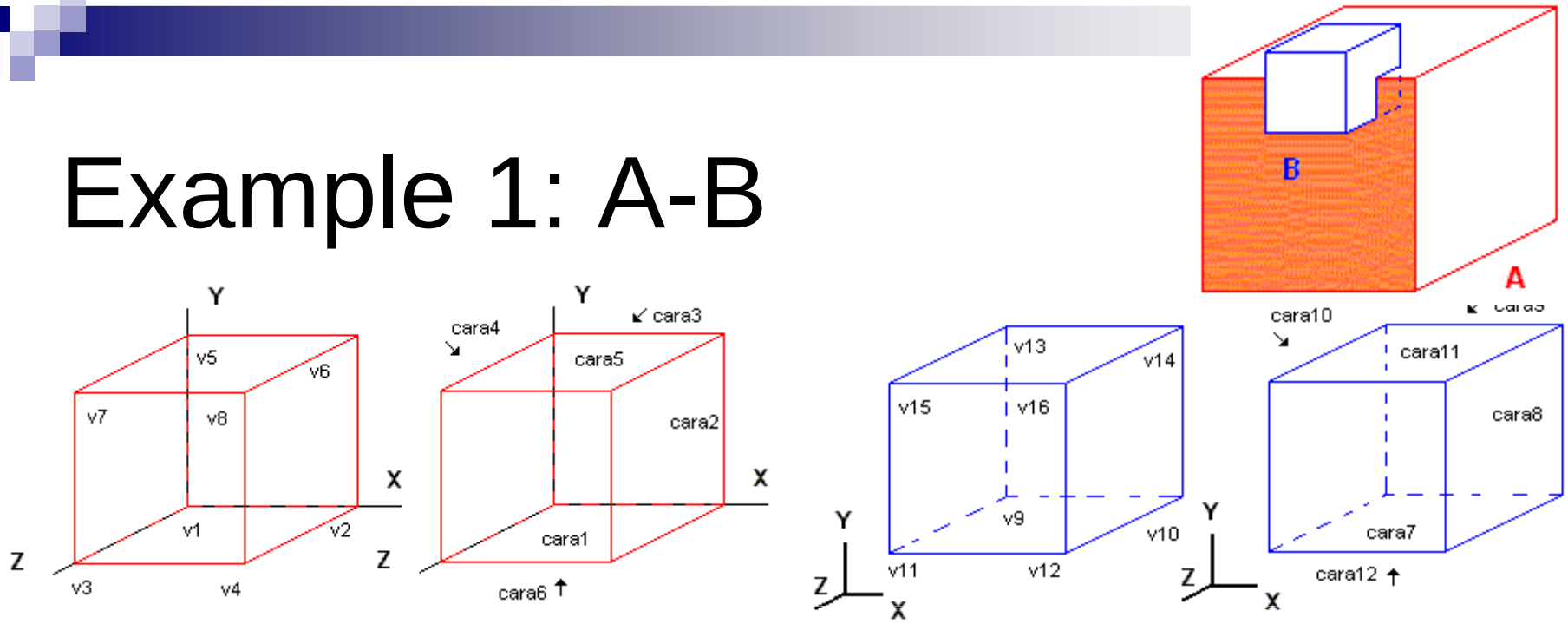
Algorithm Boolean Op (vertex classification)

```
  // 1. Classify existing vertices
  addVertices(A, B, LV); // add to LV vertices from A classified wrt B
  addVertices(B, A, LV); // add to LV vertices from B classified wrt A

  // 2. Compute new vertices
Foreach edge e from A
    foreach face f from B
      if intersect(f,e) → add( intersectionVertex (f,e),LV)

Foreach edge e from B
    foreach face f from A
      if intersect(f,e) → add( intersectionVertex (f,e),LV)
```

Algorithm 2

```
// 3. Select output vertices according to the boolean operation
foreach vertex  v in LV
  if v.type=NEW → add(result,v) otherwise
  case
    union:  if v.type = deAoutB or v.type = deBoutA → add(result,v)
    inters:  if v.type = deAinB   or v.type = deBinA → add(result,v)
    A-B :    if v.type = deAoutB or v.type = deBinA → add(result,v)
    B-A :    if v.type = deAinB   or v.type = deBoutA → add(result,v)
  end
// 4. Build F:{V}  from V:{F}
buildFaces(C)  //  change from reverse rep. to hierarchical rep.
end
```

Example 1: A-B



Objecte A

Cares	Vertexs
1:{3,4,8,7}	1:(0,0,0)
2:{2,6,8,4}	2:(3,0,0)
3:{1,5,6,2}	3:(0,0,3)
4:{1,3,7,5}	4:(3,0,3)
5:{5,7,8,6}	5:(0,3,0)
6:{1,2,4,3}	6:(3,3,0)
	7:(0,3,3)
	8:(3,3,3)

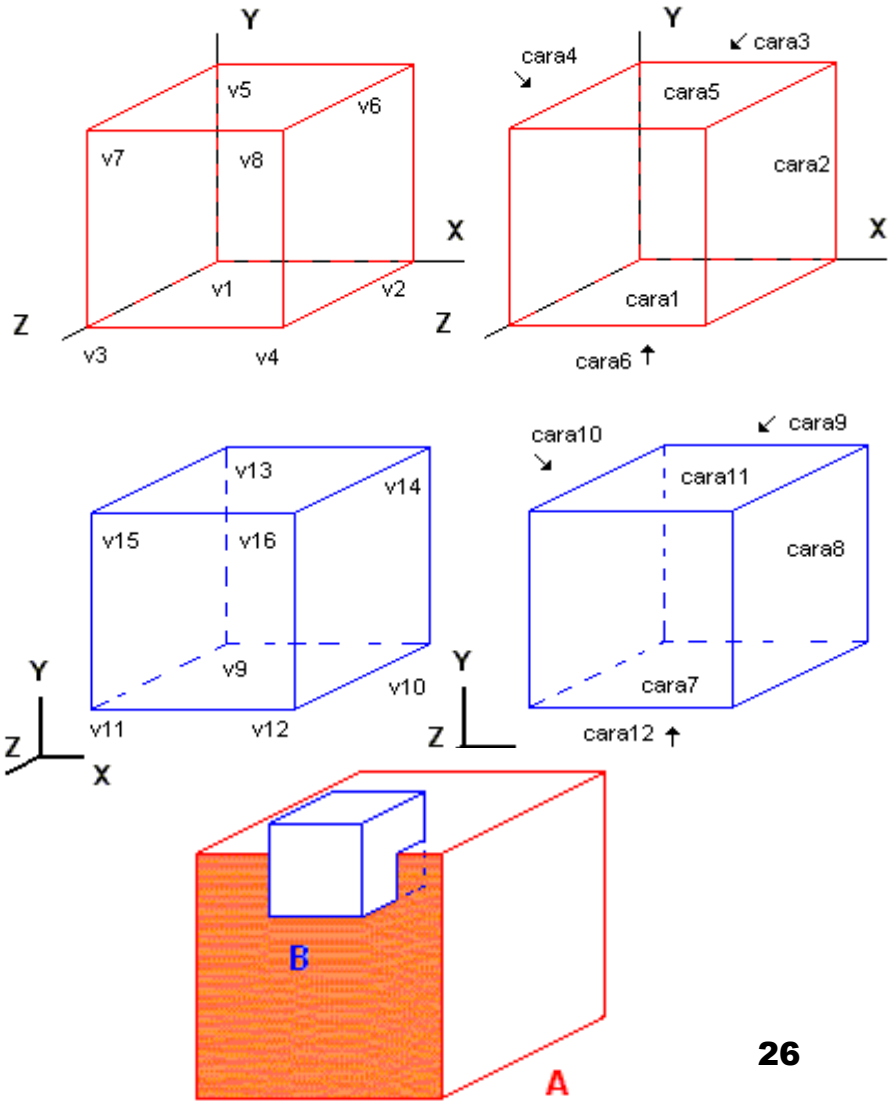
Objecte B

Cares	Vertexs
7:{11,12,16,15}	9:(1,2,2)
8:{10,14,16,12}	10:(2,2,2)
9:{ 9,13,14,10}	11:(1,2,5)
10:{ 9,11,15,13}	12:(2,2,5)
11:{13,15,16,14}	13:(1,4,2)
12:{ 9,10,12,11}	14:(2,4,2)
	15:(1,4,5)
	16:(2,4,5)

Step 1: Classify vertices

Exemple 1

```
V 1: xyz, {4,3,6},   deAoutB
V 2: xyz, {3,2,6},   deAoutB
V 3: xyz, {1,4,6},   deAoutB
V 4: xyz, {2,1,6},   deAoutB
V 5: xyz, {5,3,4},   deAoutB
V 6: xyz, {5,2,3},   deAoutB
V 7: xyz, {1,5,4},   deAoutB
V 8: xyz, {2,5,1},   deAoutB
V 9: xyz, {9,10,12}, deBinA
V10: xyz, {9,8,12},  deBinA
V11: xyz, {7,10,12}, deBoutA
V12: xyz, {8,7,12},  deBoutA
V13: xyz, {11,9,10}, deBoutA
V14: xyz, {11,8,9},  deBoutA
V15: xyz, {7,11,10}, deBoutA
V16: xyz, {8,11,7},  deBoutA
```

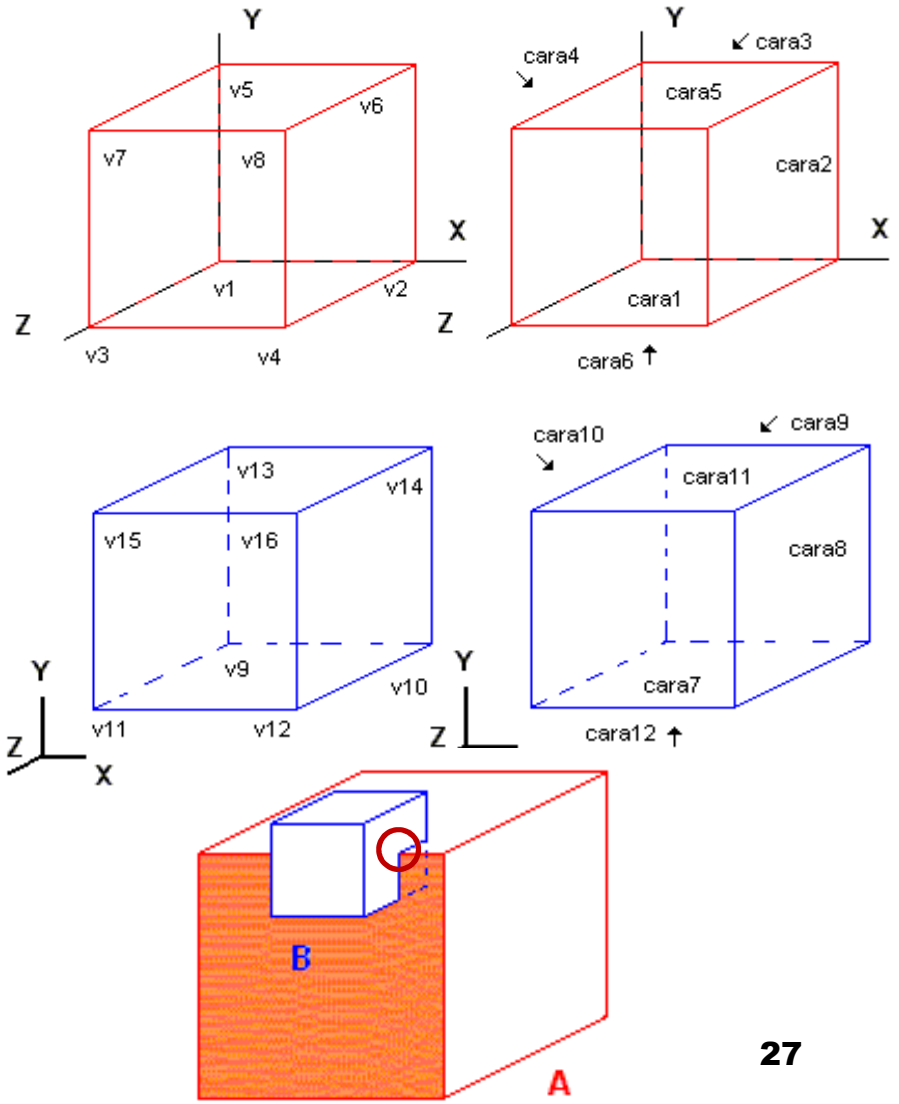


Step 2: New vertices

Exemple 1

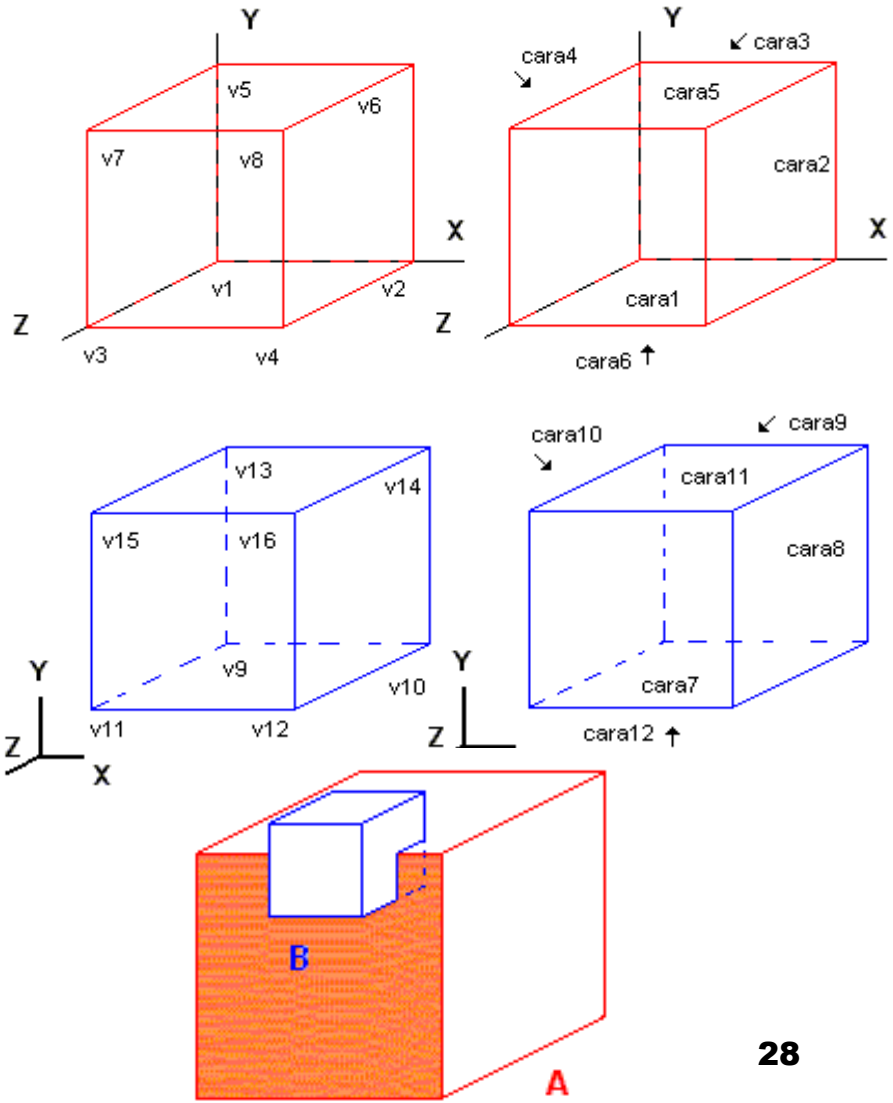
```
V17: xyz, {9,10,5}, Nou
V18: xyz, {8,9,5},  Nou
V19: xyz, {1,5,10}, Nou
V20: xyz, {1,5,8},  Nou
V21: xyz, {10,12,1},Nou
V22: xyz, {8,12,1}, Nou
```

A new vertex comes from the intersection of an edge E and a face F. The list contains **the two faces sharing E** and **the face F**.



Step 3: Select output vertices

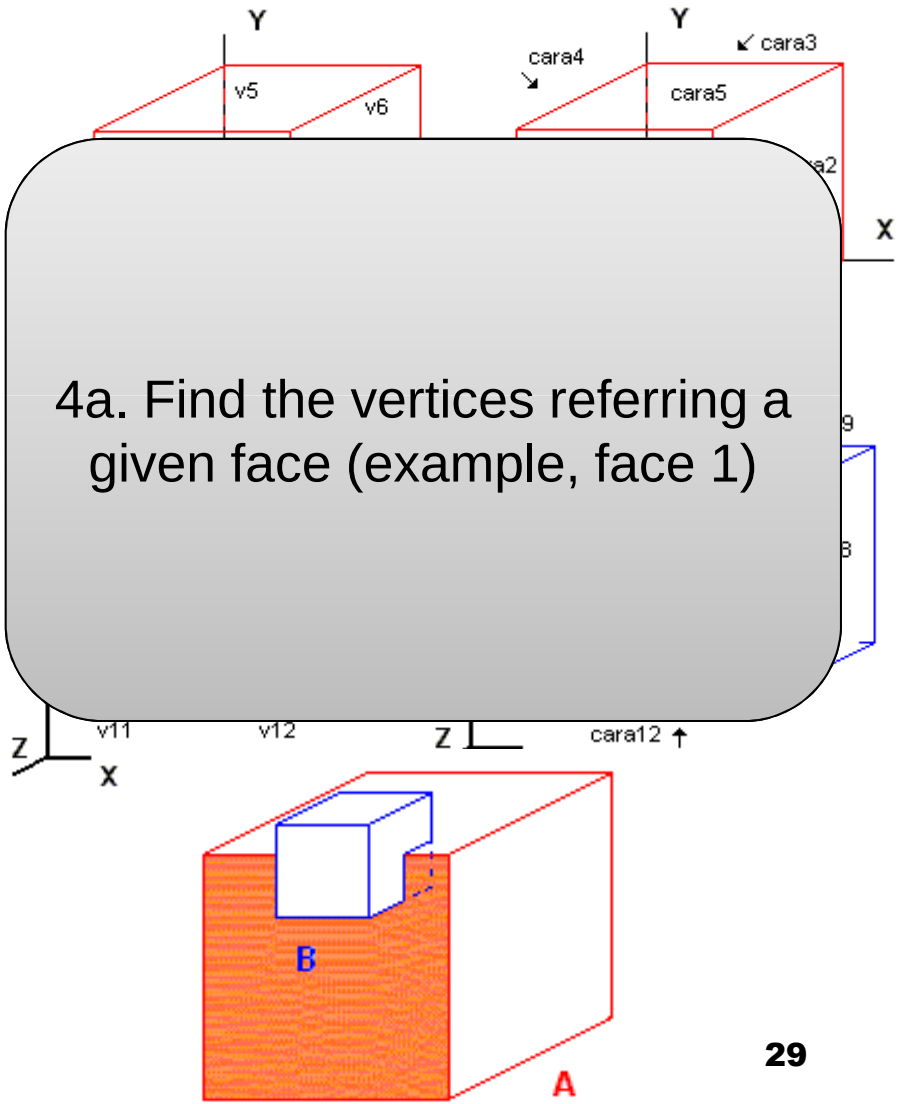
```
V 1: xyz, {4,3,6},    deAoutB
V 2: xyz, {3,2,6},    deAoutB
V 3: xyz, {1,4,6},    deAoutB
V 4: xyz, {2,1,6},    deAoutB
V 5: xyz, {5,3,4},    deAoutB
V 6: xyz, {5,2,3},    deAoutB
V 7: xyz, {1,5,4},    deAoutB
V 8: xyz, {2,5,1},    deAoutB
V 9: xyz, {9,10,12}, deBinA
V10: xyz, {9,8,12},   deBinA
V11: xyz, {7,10,12}, deBoutA
V12: xyz, {8,7,12},   deBoutA
V13: xyz, {11,9,10}, deBoutA
V14: xyz, {11,8,9},   deBoutA
V15: xyz, {7,11,10}, deBoutA
V16: xyz, {8,11,7},   deBoutA
V17: xyz, {9,10,5}, Nou
V18: xyz, {8,9,5},  Nou
V19: xyz, {1,5,10}, Nou
V20: xyz, {1,5,8},  Nou
V21: xyz, {10,12,1},Nou
V22: xyz, {8,12,1}, Nou
```



Step 4: Build faces

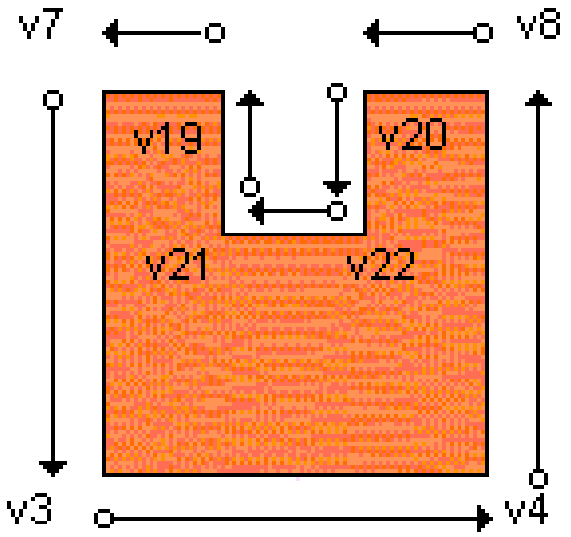
```
V 1: xyz, {4,3,6},    deAoutB
V 2: xyz, {3,2,6},    deAoutB
V 3: xyz, {1,4,6},    deAoutB
V 4: xyz, {2,1,6},    deAoutB
V 5: xyz, {5,3,4},    deAoutB
V 6: xyz, {5,2,3},    deAoutB
V 7: xyz, {1,5,4},    deAoutB
V 8: xyz, {2,5,1},    deAoutB
V 9: xyz, {9,10,12}, deBinA
V10: xyz, {9,8,12},   deBinA
V11: xyz, {7,10,12}, deBoutA
V12: xyz, {8,7,12},   deBoutA
V13: xyz, {11,9,10}, deBoutA
V14: xyz, {11,8,9},   deBoutA
V15: xyz, {7,11,10}, deBoutA
V16: xyz, {8,11,7},   deBoutA
V17: xyz, {9,10,5},  Nou
V18: xyz, {8,9,5},    Nou
V19: xyz, {1,5,10},  Nou
V20: xyz, {1,5,8},    Nou
V21: xyz, {10,12,1},Nou
V22: xyz, {8,12,1},  Nou
```

4a. Find the vertices referring a given face (example, face 1)

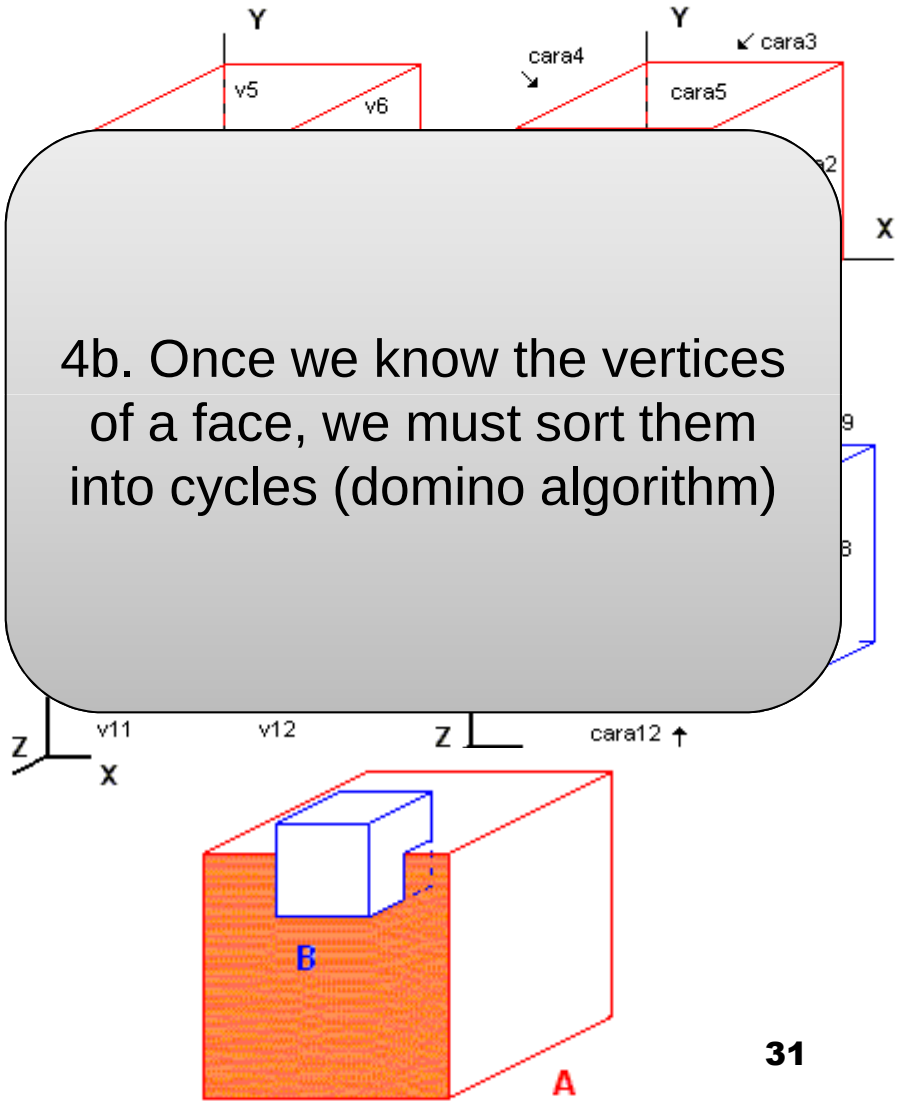


Step 4: Build faces

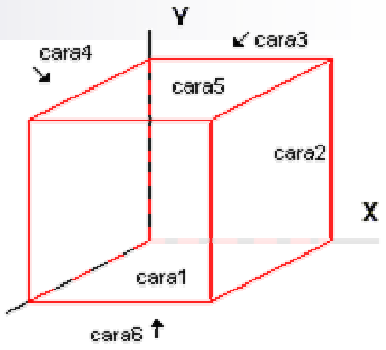
```
V 3: xyz, {4,1,6}, deAoutB
V 4: xyz, {2,1,6}, deAoutB
V 7: xyz, {4,1,5}, deAoutB
V 8: xyz, {5,1,2}, deAoutB

V19: xyz, {10,1,5}, Nou
V20: xyz, {8,1,5},  Nou
V21: xyz, {12,1,10},Nou
V22: xyz, {8,1,12}, Nou
```



4b. Once we know the vertices of a face, we must sort them into cycles (domino algorithm)

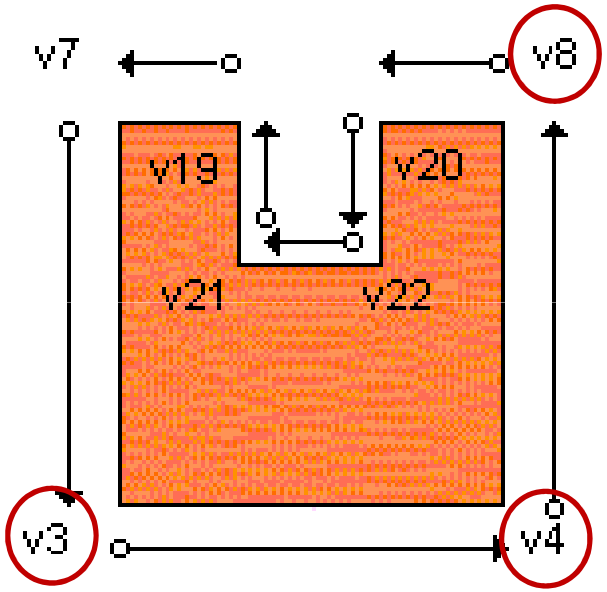


Step 4: Build faces



❶ V 3: xyz, {4,1,6}, deAoutB
❷ V 4: xyz, {2,1,6}, deAoutB
V 7: xyz, {4,1,8}, deAoutB
❸ V 8: xyz, {5,1,2}, deAoutB

V19: xyz, {10,1,5}, Nou
V20: xyz, {8,1,5}, Nou
V21: xyz, {12,1,10},Nou
V22: xyz, {8,1,12}, Nou

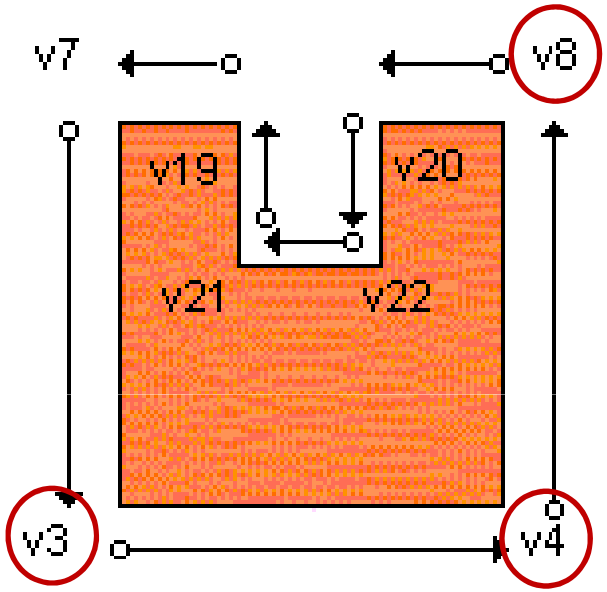


V3 ------> V4 ------> V8 ------> ?
Cara 6 Cara 2 Cara 5

Step 4: Build faces

To solve the indetermination:

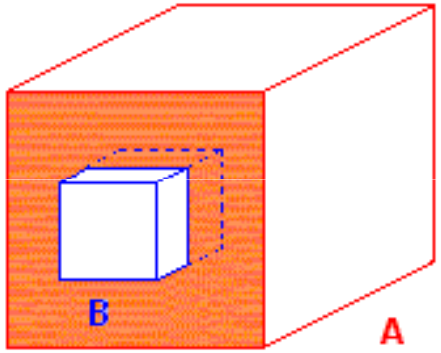
1. **Sort the vertices** involved according to the parameter of the supporting line: V8, V20, V19, V7
2. **Group forming pairs** (will become edges of the result): (V8, V20) (V19, V7).

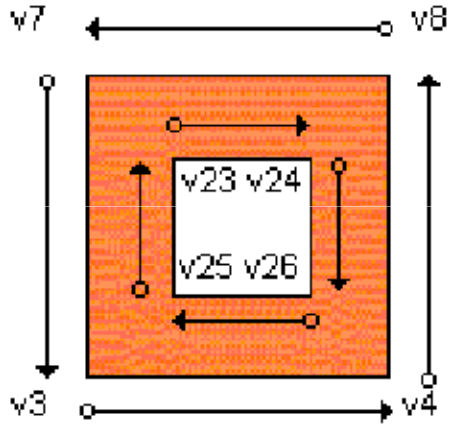



```
V8 ------> V20 ------> V22 ------> V21 -------> V19
   Cara 5      Cara 8     Cara 12      Cara 10   |
                                                 |
                                                 | Cara
                                                 |
                                                 |
                                V3 <------ V7 <--
                                    Cara 4
```

Example 2. Still A-B

The domino algorithm can detect more than one cycle (faces with internal loops)





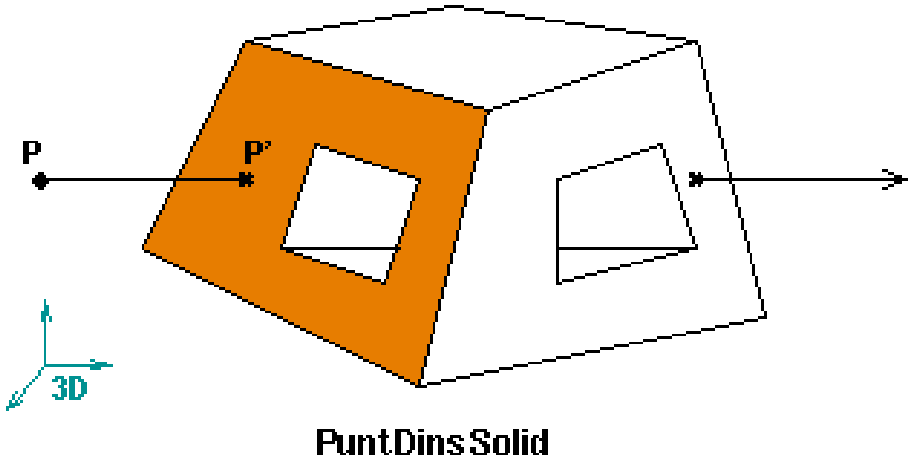

```
V3 ------> V4 ------> V8 ------> V7 ------> V3
   Cara 6     Cara 2     Cara 5     Cara 4
```

```
V23 -------> V24 ------> V26 -------> V25 -------> V23
    Cara 11      Cara 8      Cara 12      Cara 10
```

Geometric tests

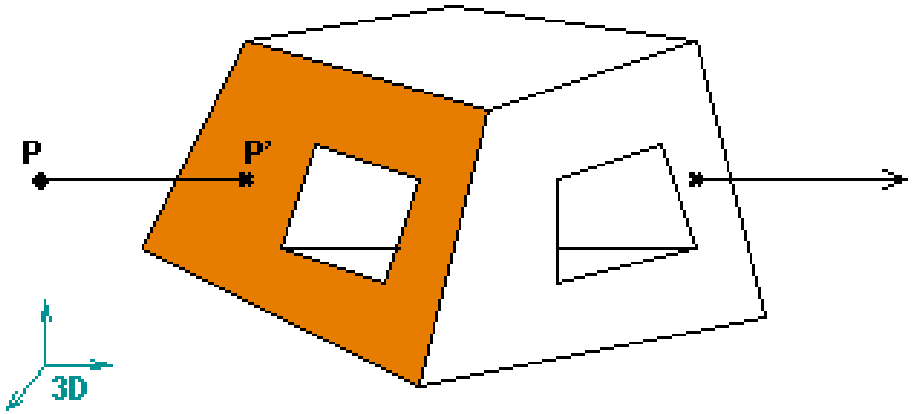
- Point inside solid
- Convexity of an egde
- Sorting faces around a vertex
- Classify cycles as interior/exterior

Point inside solid

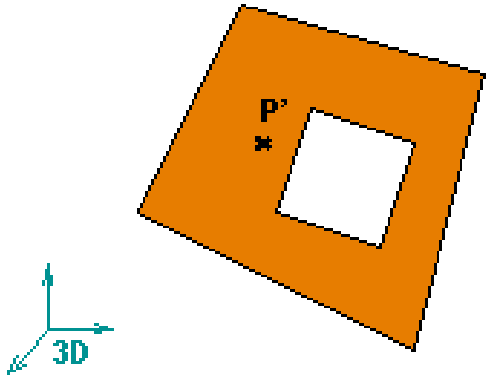


PuntDinsSolid

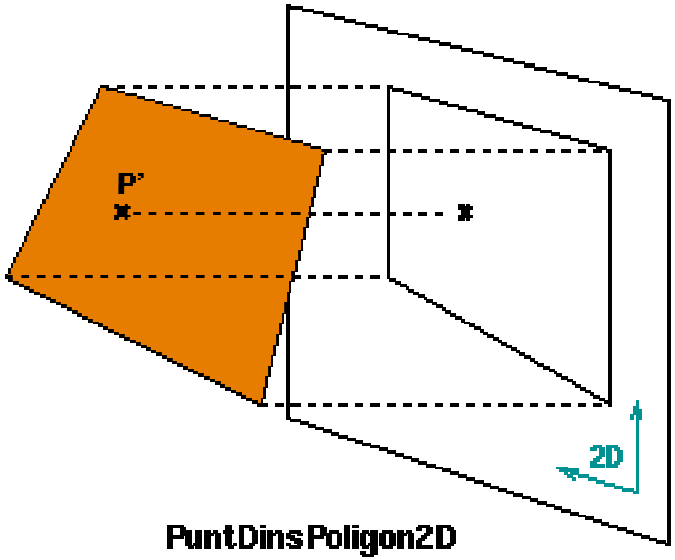
Point inside solid



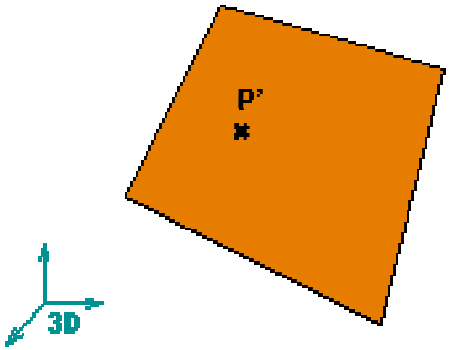
PuntDinsSolid



PuntDinsCara3D

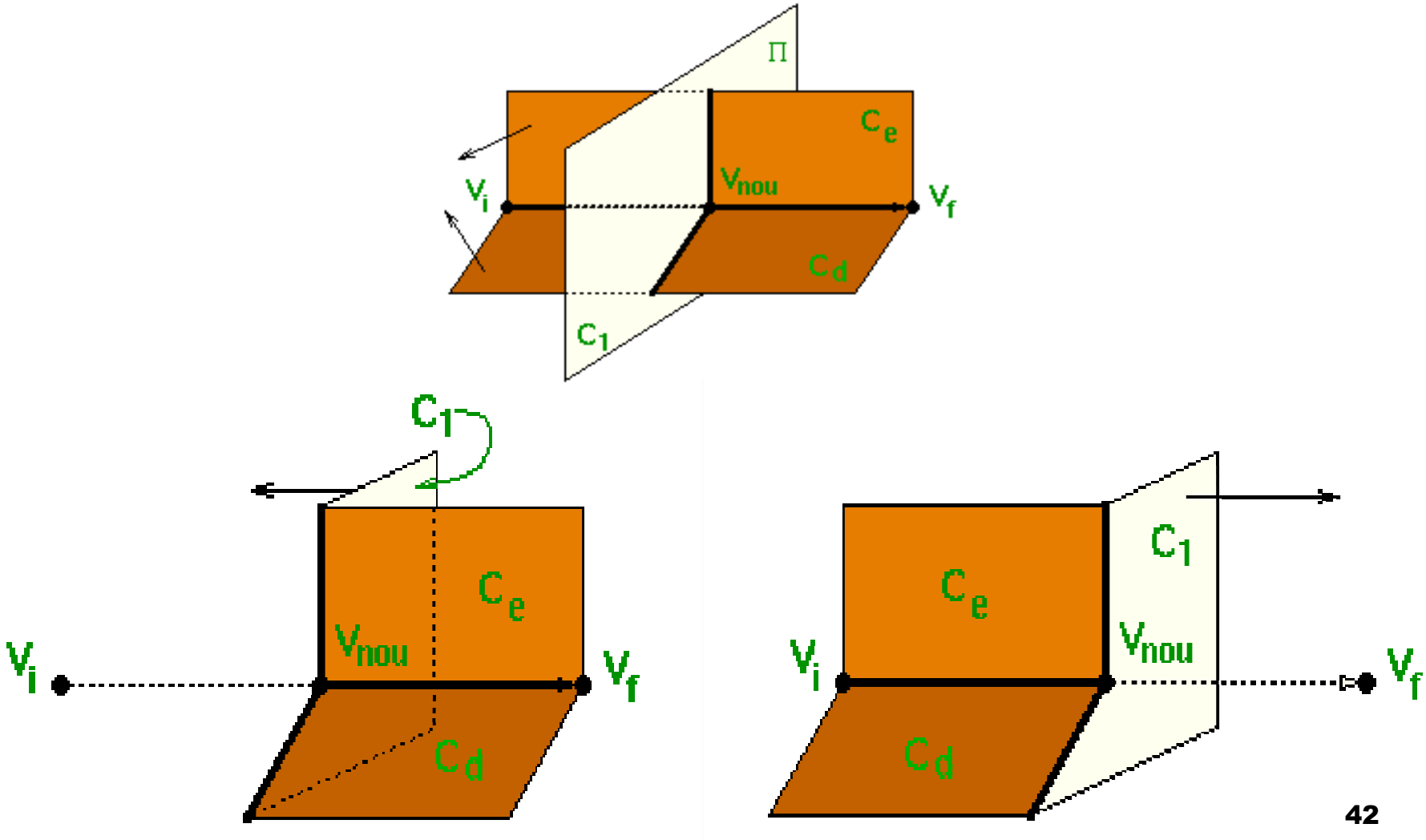


PuntDinsPoligon2D

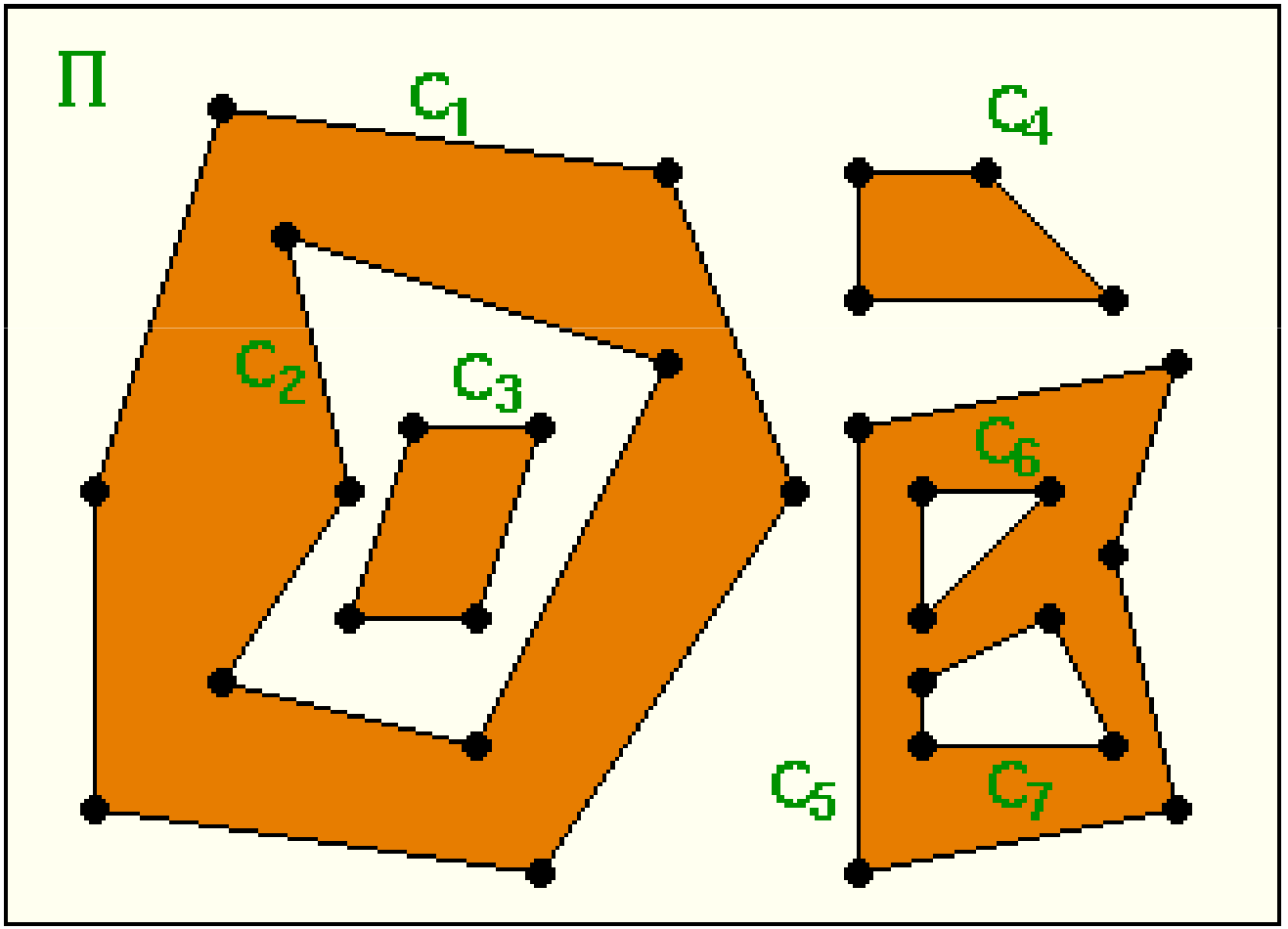


PuntDinsPoligon3D

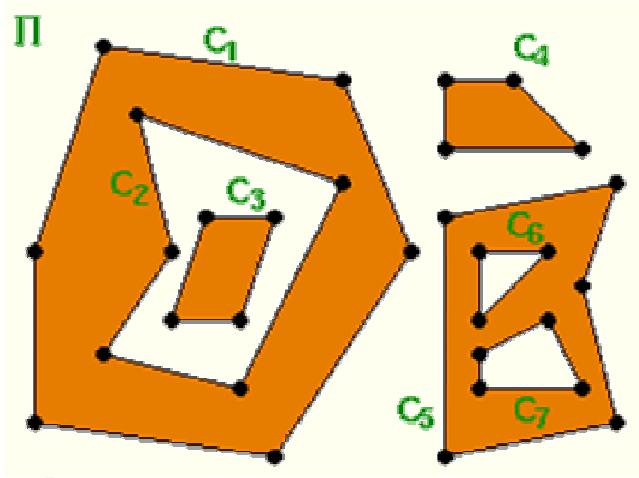
Sorting faces around a vertex



Classify cycles as in/out



Classify loops as in/out



parity=**true**
C:=set of loops
while C is not empty **do**
    D := Ø
    for each loop cx in C **fer**
      if cx is inside to some loop cy in C **then**
          classify cx as an internal loop of cy
        else D := D + {cx}
    if parity **then** loops in D are exterior loops of faces
    else the loops in D are interior loops
    parity:=**not** parity
    C:=C-D
end