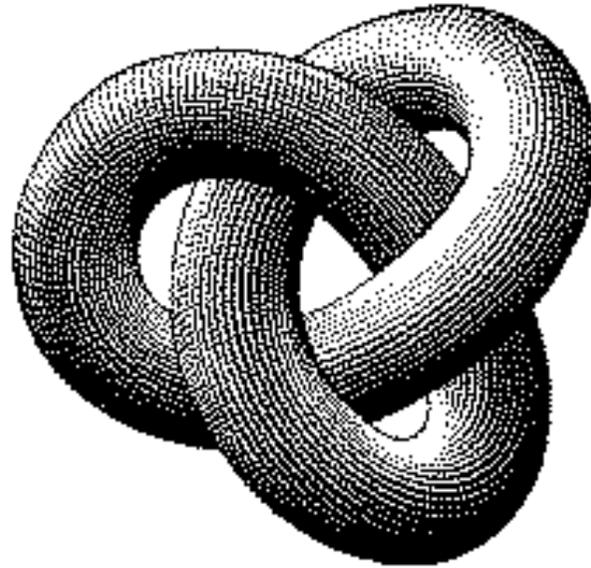


# Modélisation



*Alexis Nédélec*

Ecole Nationale d'Ingénieurs de Brest  
Technopôle Brest-Iroise, Site de la Pointe du Diable  
CP 15 29608 BREST Cedex (FRANCE)  
e-mail : nedelec@enib.fr

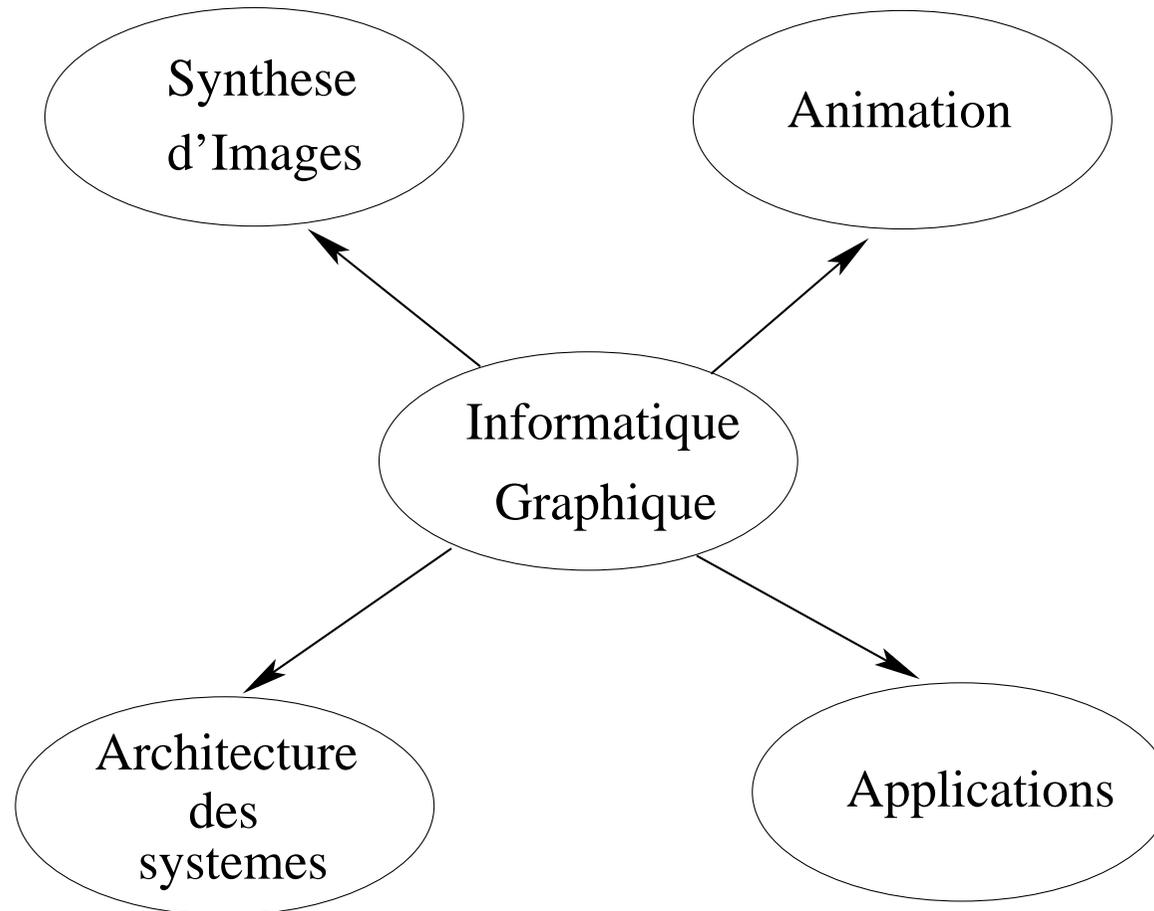
## Table des Matières

Introduction	3
Schéma de Représentation	9
Représentation Fil-De-Fer (Wire Frame)	18
Représentation par Frontières (BREPS)	21
Structure de Arête-Ailée (Baumgart)	29
Relations d'adjacences (topologie BREPS)	38
Opérateurs d'Euler (topologie BREPS)	50
Cartes Topologiques: n-G-maps	65
Représentation Volumique: CSG	74
Modélisation Fractale (Mandelbrot)	90
Modélisation de Plantes (L-Systems)	94
Conclusion	109
Bibliographie	112

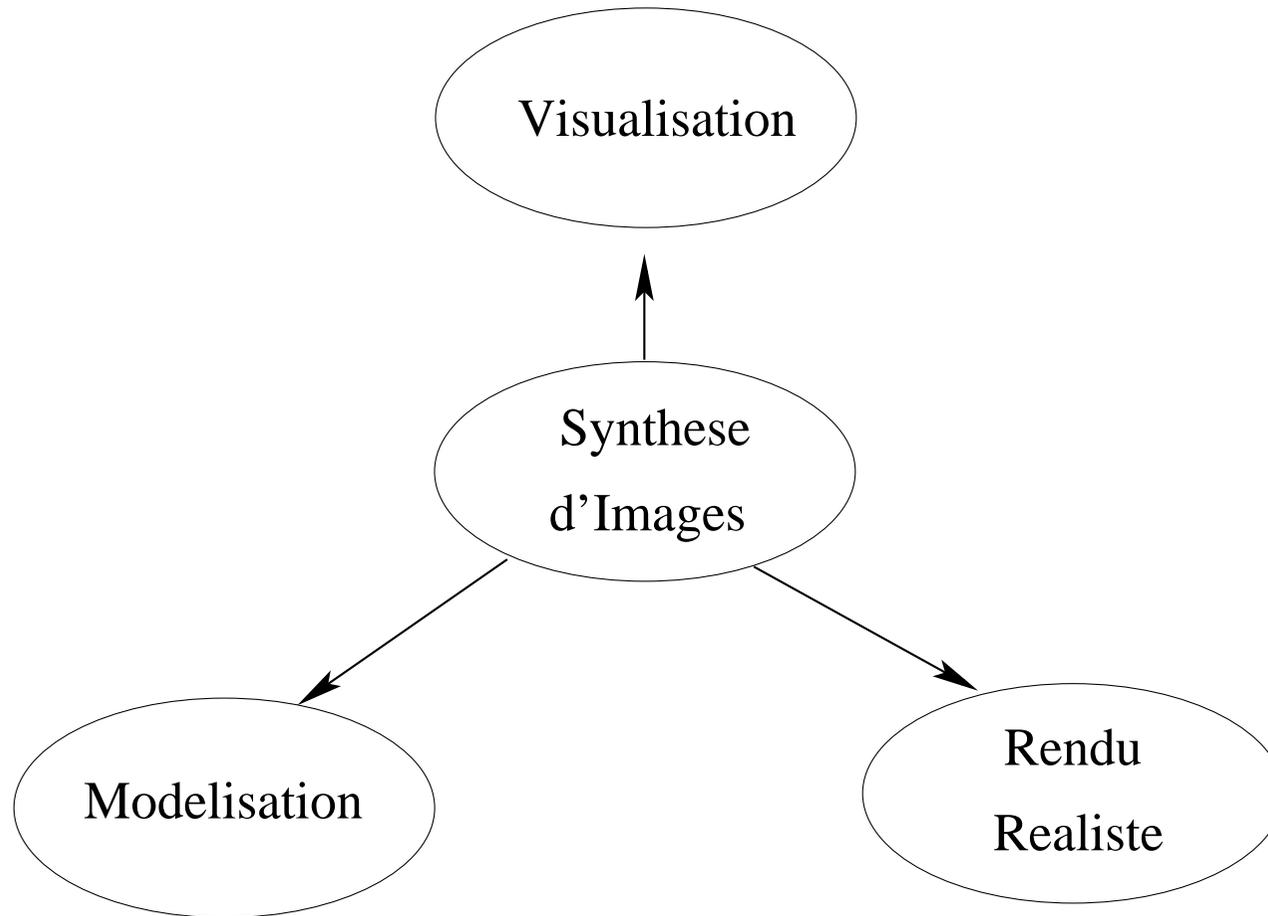
## **Introduction**

- ▷ Modélisation des objets de l'univers
- ▷ Visualisation des univers
- ▷ Transformation des objets de l'univers
- ▷ Habillages des objets (couleurs, textures)
- ▷ Rendu Réaliste (modèles d'éclairage)
- ▷ Animation de scènes
- ▷ Outils graphiques et Réalité Virtuelle

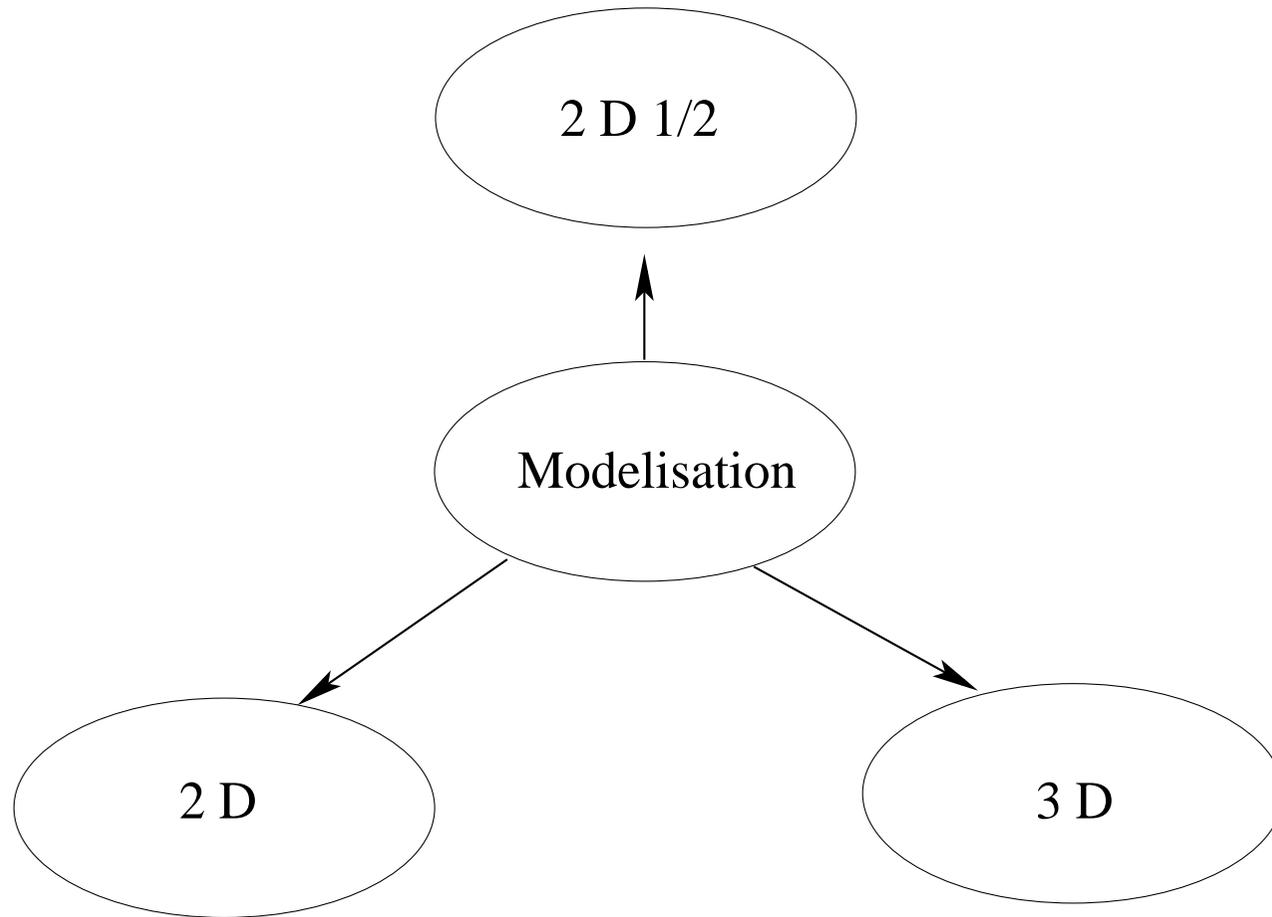
# Introduction



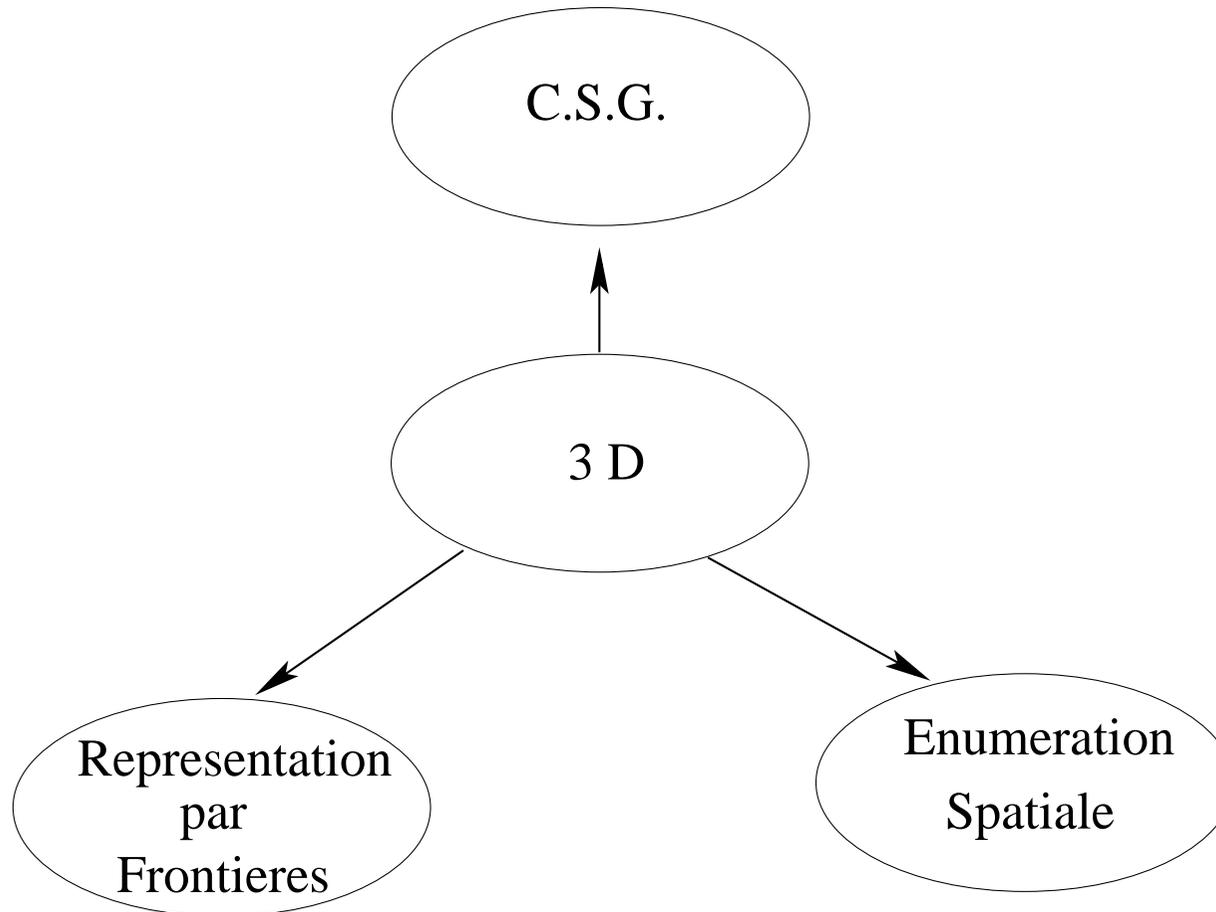
# Introduction



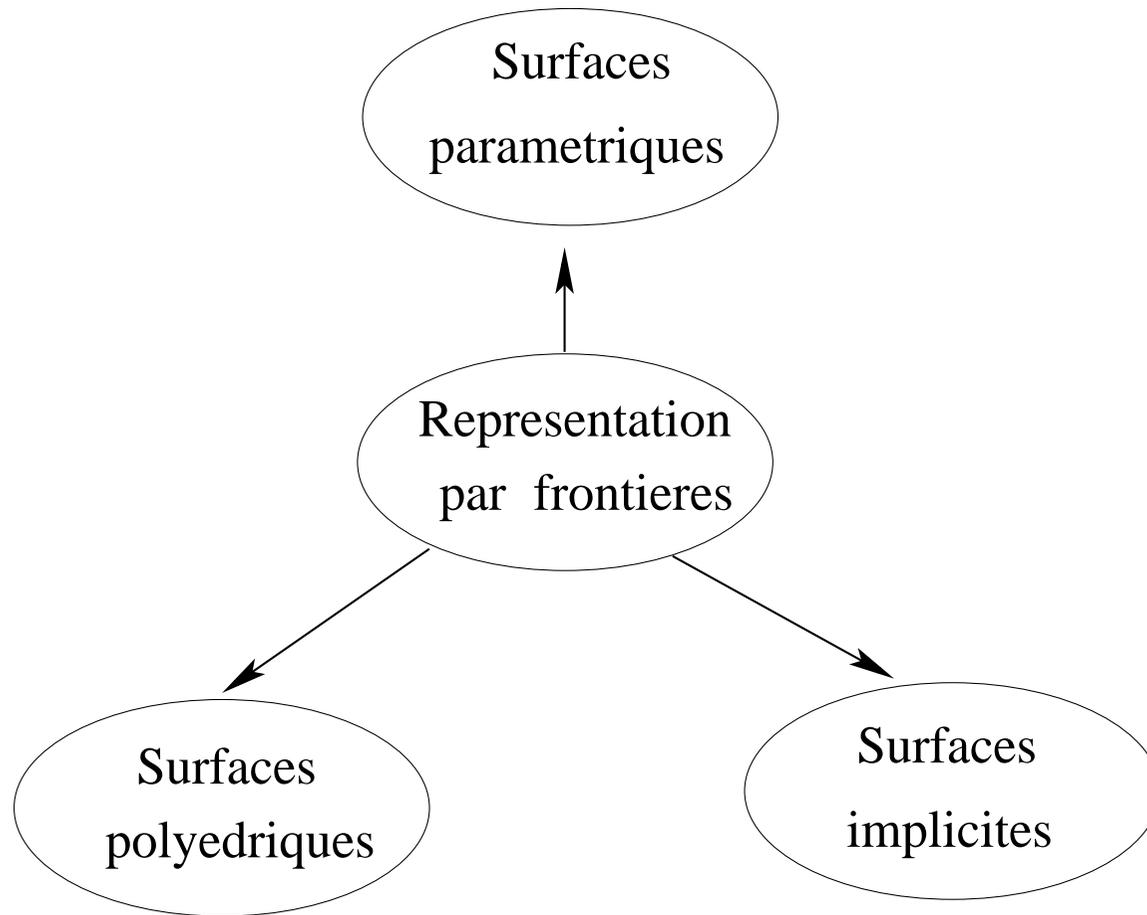
# Introduction



# Introduction



# Introduction

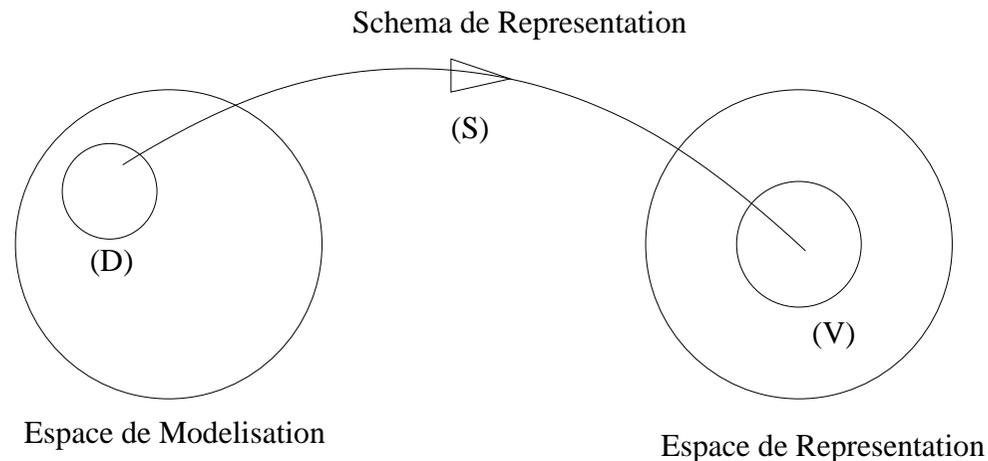


## Schéma de Représentation

Modélisation d'Univers (domaine) d'un point de vue informatique (validité)

- ▷ représentations des objets (structures de modélisation)
- ▷ traitements sur les objets (opérations sur les structures)

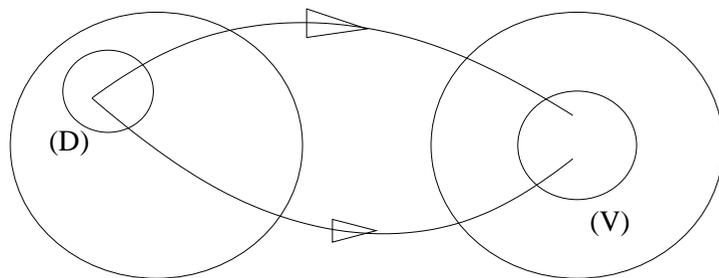
Schéma de représentation:



## Schéma de Représentation

Définition formelle de la représentation d'un solide du domaine

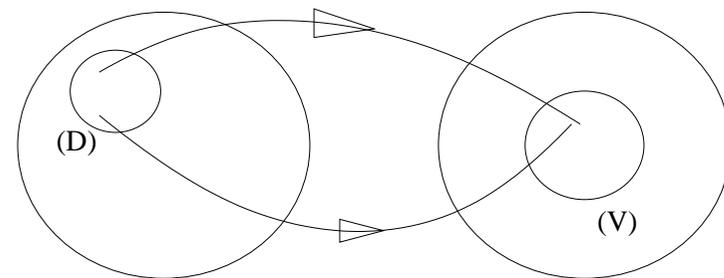
- ▷ Domaine: Pouvoir descriptif (entités représentables de l'environnement)
- ▷ Validité: Syntaxique et Sémantique (structures et signification géométrique)
- ▷ Non-ambiguïté: à UNE REPRESENTATION valide, UN SEUL SOLIDE
- ▷ Unicité: à UN SOLIDE du domaine, UNE SEULE REPRESENTATION



Modelisation

Représentation

NON-UNICITE



Modelisation

Représentation

AMBIGUITE

## Schéma de Représentation

Cas idéal de schéma de représentation:

NON-AMBIGUITE + UNICITE

On définit de manière plus pratique

- ▷ **Concision** : grande concision si non-redondance d'informations
- ▷ **Facilité de création** : liée à l'utilisateur et par conséquent à la concision
- ▷ **Efficacité** : totalement liée à l'application

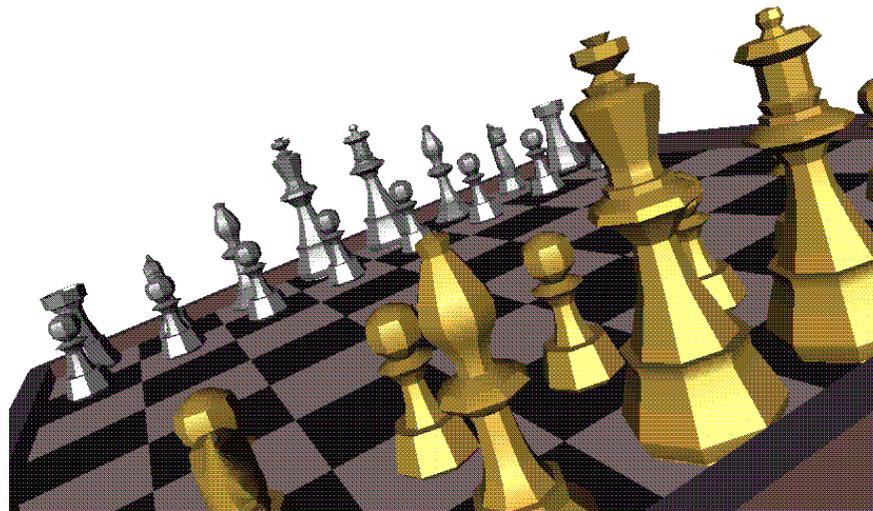
Applications en Réalité Virtuelle

- ▷ Quel domaine d'applications ?
- ▷ Quels modèles et quels types de représentation choisir?

## Classification de Modèles

On distingue les représentations

- ▷  $2D$  (segments, polygones, courbes, ...)
- ▷  $2D\frac{1}{2}$ ,  $z = f(x, y)$  (MNT)
- ▷  $3D$ ,  $\{(x, y, z) \in R^3; f(x, y, z) \leq 0\}$  (solides)



## Classification de Modèles

- ▷ Géométrie Constructive de Solides (CSG)
- ▷ Représentation par Frontières (B-REPS)
- ▷ Représentation Octree (voxels)
- ▷ Décomposition en cellules (voxel étendu)
- ▷ Modèles Topologiques (graphes d'adjacences, cartes topologiques, ...)
- ▷ Modèles de balayages
- ▷ Instantiation de primitives

## Classification de Modèles

Ces modèles sont répertoriés en trois catégories

1. représentations volumiques

- ▷ modèles constructifs (CSG)
- ▷ modèles descriptifs (Octree)

2. représentations par frontières

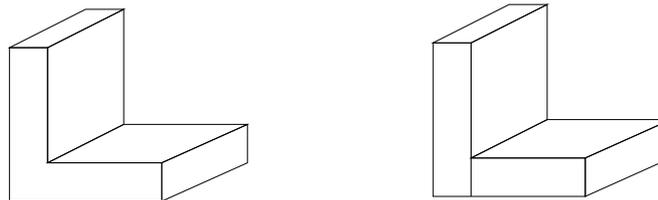
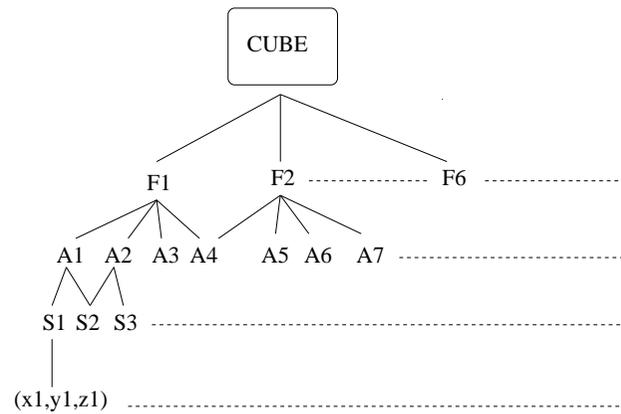
- ▷ BReps, Cartes Généralisés (n-G-maps),...

3. modèles de balayage

- ▷ simples, Hybrides, généralisés

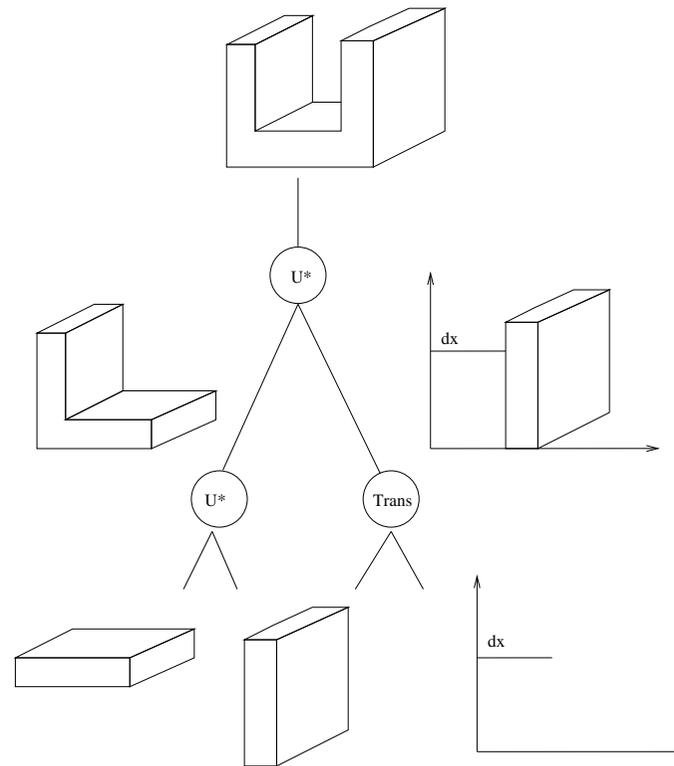
et les modèles plus récents : fractals, particules, meta-balls ...

# Représentation par frontières



Représentation Non-Ambigue / Non-Unique

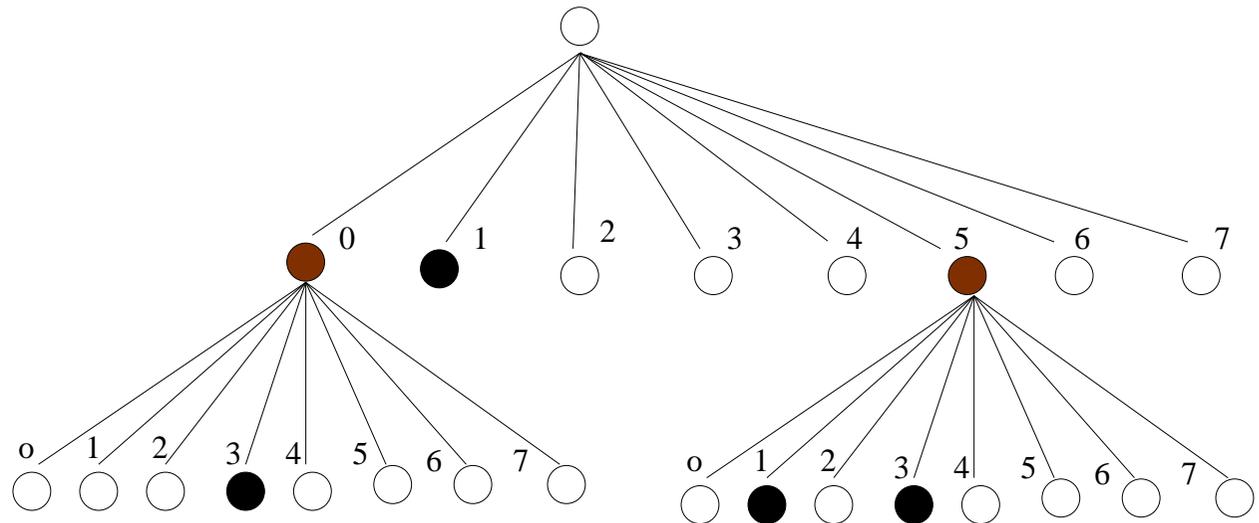
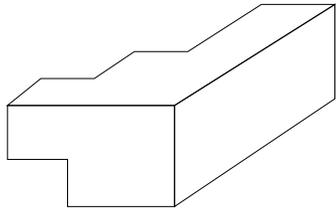
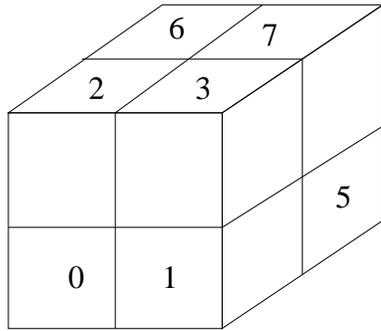
# Constructive Solid Geometry



Représentation Non-Ambigue / Non-Unique

Schéma de Représentation

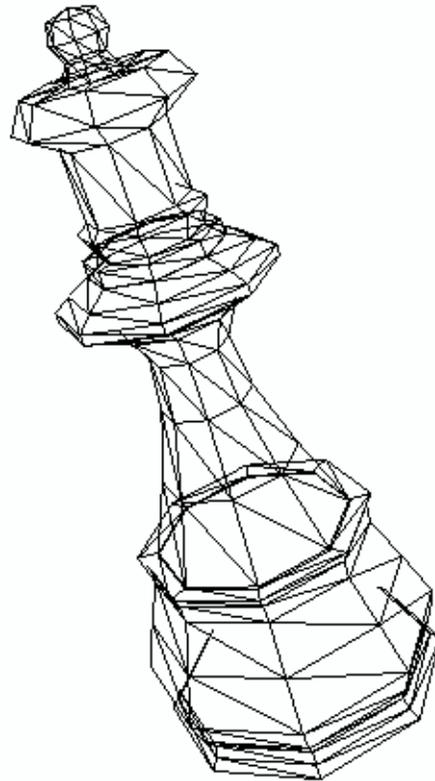
# Octree



Représentation Non-Ambigue / Unique

## Représentation Fil-de-Fer

Représentation par listes de points et de segments (Wire Frame)



## Wire Frame

Premier modèle de représentation d'objet sous forme de listes

- ▷ Points : coordonnées de sommets (géométrie)
- ▷ Arête : liaisons entre sommets (topologie)

Structure minimale dans un langage de type C

```
typedef struct {  
    int x,y;  
} Vertex;
```

Affichage d'objets peut s'avérer complexe avec les commandes de tracer

- ▷ positionnement (move), tracer entre deux points (draw)

Pour effectuer plus simplement ce traitement:

```
typedef struct {  
    Vertex p1,p2;  
} Edge;
```

## Redondance d'informations

Problèmes de ce type de structure

- ▷ redondance et mises à jour des informations

Utilisation de pointeurs sur la structure d'arête

```
typedef struct {           | typedef struct {  
    int x,y;               |     VertexList p1,p2;  
} Vertex, *VertexList;    | } Edge;
```

### Avantages

- ▷ stockage : l'information géométrique (sommet) est partagée
- ▷ modification : mise à jour des informations topologiques (arêtes)

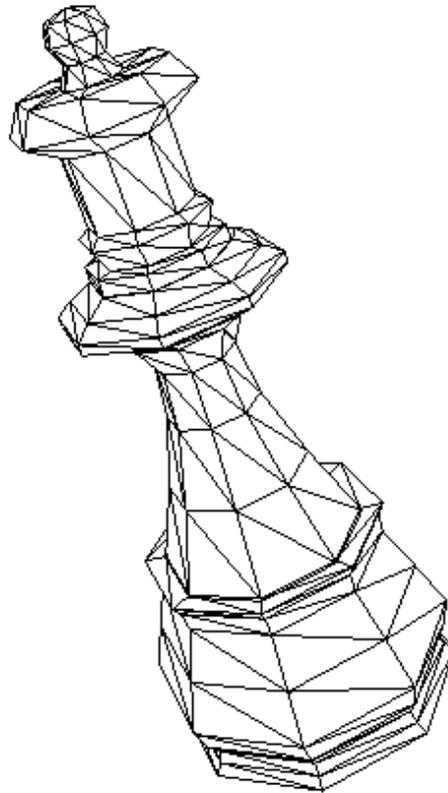
### Inconvénient

- ▷ gestion des pointeurs : s'adresser à ce qui existe

Vérifier la cohérence du modèle : d'un point de vue géométrique et topologique

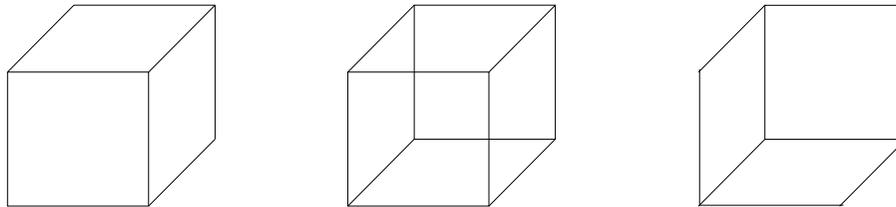
## Représentation par Frontières

Représentation par listes de points, de segments et de faces (BREPS)



## Représentation par Frontières

- ▷ Modélisation Fil-de-Fer : Pas de notion volumique



Evolution du modèle, Informations sur les frontières :

- ▷ **B**oundary **R**epresentation

Problème d'une telle représentation

- ▷ définir une structure de représentation complète:
  - ◇ géométriquement et topologiquement
- ▷ pouvoir parcourir la topologie du modèle (relations d'adjacences)
- ▷ créer des modèles par insertion d'entités (sommets, arêtes, faces)

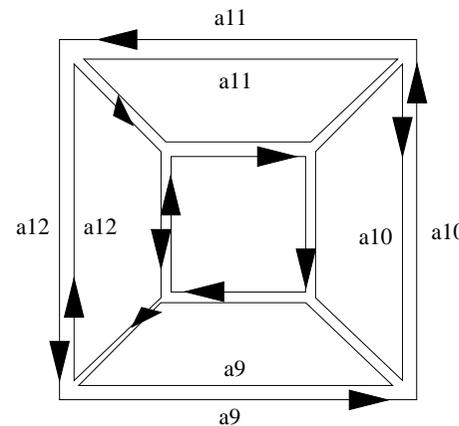
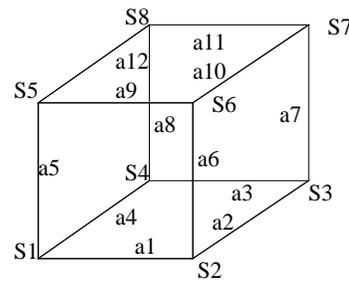
## Représentation par Frontières

**BReps** : représentation polyédrique d'objets (sommets, arêtes, faces)

Problème d'une telle modélisation

Séparer les volumes intérieur et extérieur

Solution : Surfaces orientées



## Orientabilité de Surface

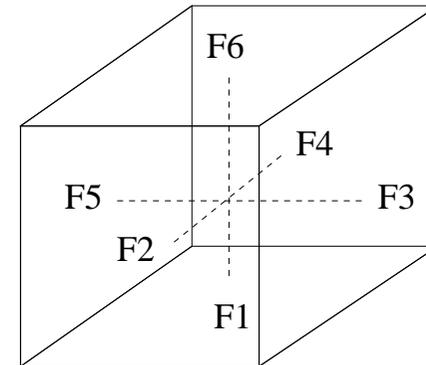
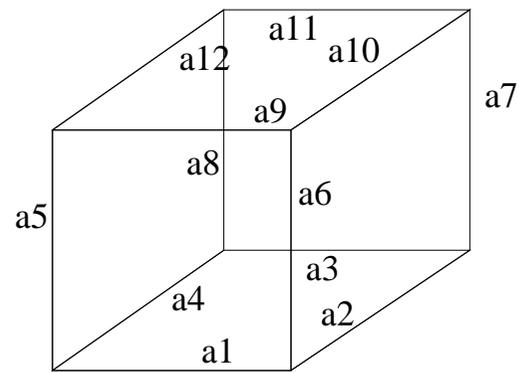
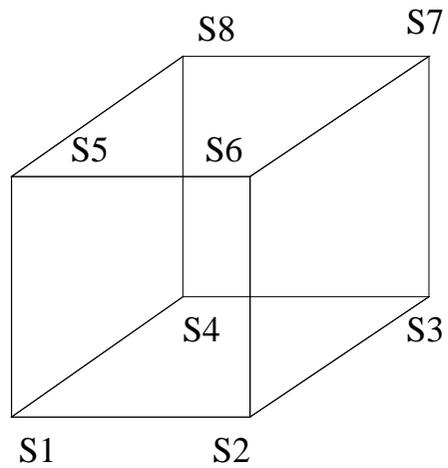
Un modèle plan sera **orientable** si chaque arête

- ▷ est parcourue 2 fois dans 2 sens opposés
- ▷ appartient à 2 faces
- ▷ est définie par 2 sommets

Définition de structure BReps de modélisation par frontières par listes de

- ▷ Sommet : informations géométriques (coordonnées)
- ▷ Arête : informations topologiques (relations d'adjacences F,A,S)
- ▷ Face : informations géométriques (normales aux surfaces)

# Topologie du Modèle



## Topologie du Modèle

Arête	Sommets	Faces	Arête
$a_1$	$(s_1, s_2)$		
$a_2$	$(s_2, s_3)$	$F_1$	$(a_1, a_4, a_3, a_2)$
$a_3$	$(s_3, s_4)$		
$a_4$	$(s_4, s_1)$	$F_2$	$(a_1, a_6, a_9, a_5)$
$a_5$	$(s_1, s_5)$		
$a_6$	$(s_2, s_6)$	$F_3$	$(a_2, a_7, a_{10}, a_6)$
$a_7$	$(s_3, s_7)$		
$a_8$	$(s_4, s_8)$	$F_4$	$(a_3, a_8, a_{11}, a_7)$
$a_9$	$(s_5, s_6)$		
$a_{10}$	$(s_6, s_7)$	$F_5$	$(a_4, a_5, a_{12}, a_8)$
$a_{11}$	$(s_7, s_8)$		
$a_{12}$	$(s_8, s_5)$	$F_6$	$(a_9, a_{10}, a_{11}, a_{12})$

## Topologie du Modèle

Sous cette représentation chaque arête

- ▷ appartient à 2 faces
- ▷ est définie par 2 sommets

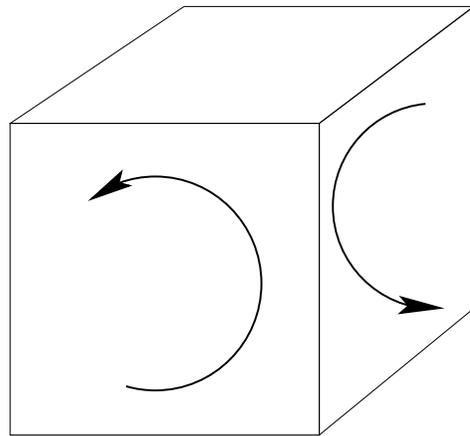
La Topologie sera complète en considérant

les **Ailes** d'une arête

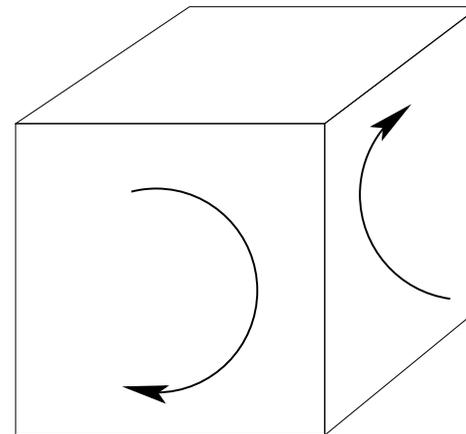
de façon à parcourir chaque arête 2 fois dans 2 sens opposés

- ▷ sens trigo (counter-clockwise)
- ▷ sens inverse (clockwise)

## Topologie du Modèle



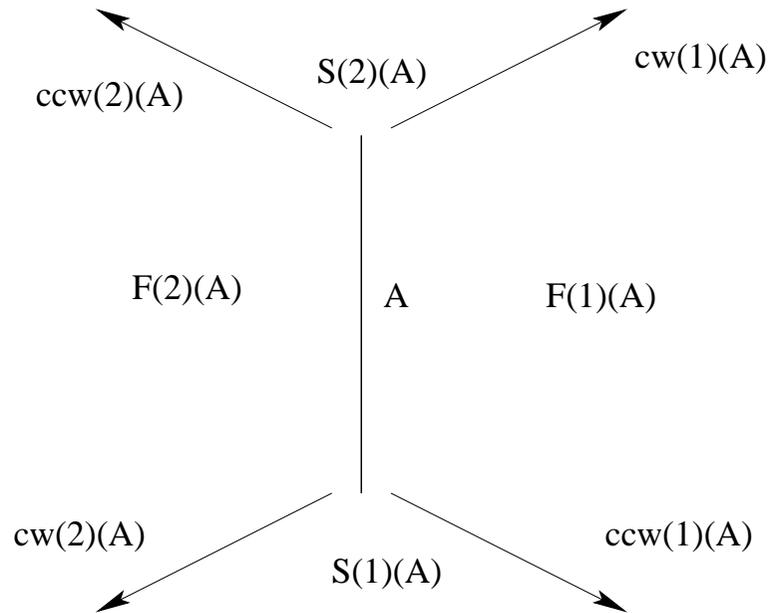
Counter-clockwise



Clockwise

## Structure Arête-Ailée

Winged-Edge Data Structure: Structure introduite par Baumgart (1972)



## Structure Arête-Ailée

Une **Arête** devra pointer sur

- ▷ 2 faces, 2 sommets, 4 arêtes (ailes)

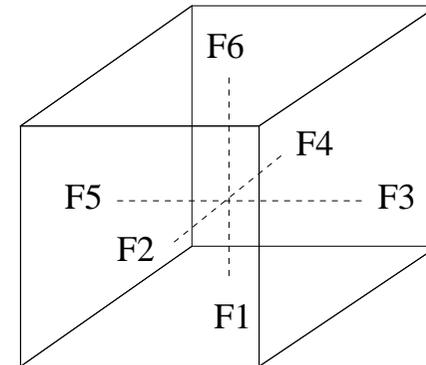
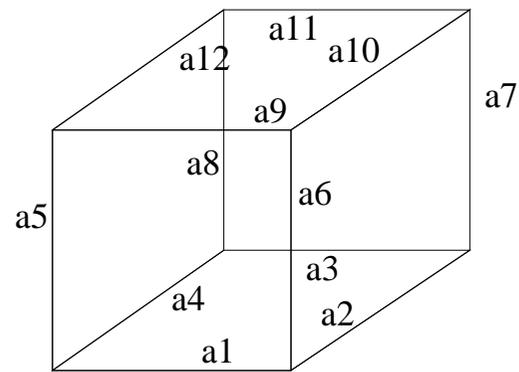
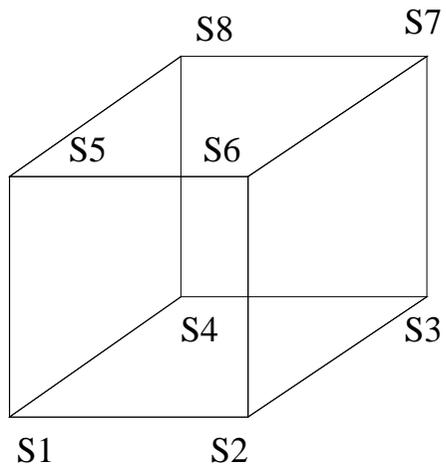
Une **Face**

- ▷ doit pointer sur une de ses arêtes incidentes
- ▷ contenir des informations géométriques (normales)

Un **Sommet**

- ▷ doit pointer sur une de ses arêtes incidentes
- ▷ contenir des informations géométriques (coordonnées)

# Topologie du Modèle



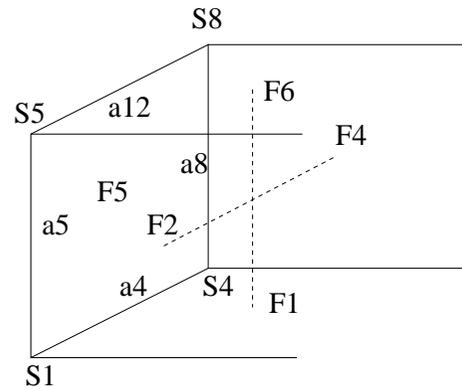
## Topologie du Modèle

Arête (a)	S(1) (a)	S(2) (a)	F(1) (a)	F(2) (a)
$a_1$	$S_1$	$S_2$	$F_1$	$F_2$
$a_2$	$S_2$	$S_3$	$F_1$	$F_3$
$a_3$	$S_3$	$S_4$	$F_1$	$F_4$
$a_4$	$S_4$	$S_1$	$F_1$	$F_5$
$a_5$	$S_1$	$S_5$	$F_2$	$F_5$
$a_6$	$S_2$	$S_6$	$F_3$	$F_2$
$a_7$	$S_3$	$S_7$	$F_4$	$F_3$
$a_8$	$S_4$	$S_8$	$F_5$	$F_4$
$a_9$	$S_5$	$S_6$	$F_2$	$F_6$
$a_{10}$	$S_6$	$S_7$	$F_3$	$F_6$
$a_{11}$	$S_7$	$S_8$	$F_4$	$F_6$
$a_{12}$	$S_8$	$S_5$	$F_5$	$F_6$

## Topologie du Modèle

Arête (a)	cw(1) (a)	ccw(1) (a)	cw(2) (a)	ccw(2) (a)
$a_1$	$a_2$	$a_4$	$a_5$	$a_6$
$a_2$	$a_3$	$a_1$	$a_6$	$a_7$
$a_3$	$a_4$	$a_2$	$a_7$	$a_8$
$a_4$	$a_1$	$a_3$	$a_8$	$a_5$
$a_5$	$a_9$	$a_1$	$a_4$	$a_{12}$
$a_6$	$a_{10}$	$a_2$	$a_1$	$a_9$
$a_7$	$a_{11}$	$a_3$	$a_2$	$a_{10}$
$a_8$	$a_{12}$	$a_4$	$a_3$	$a_{11}$
$a_9$	$a_6$	$a_5$	$a_{12}$	$a_{10}$
$a_{10}$	$a_7$	$a_6$	$a_9$	$a_{11}$
$a_{11}$	$a_8$	$a_7$	$a_{10}$	$a_{12}$
$a_{12}$	$a_5$	$a_8$	$a_{11}$	$a_9$

# Topologie du Modèle



a	S(1)(a)	S(2)(a)
$a_5$	$S_1$	$S_5$
$a_4$	$S_4$	$S_1$
$a_8$	$S_4$	$S_8$
$a_{12}$	$S_8$	$S_5$

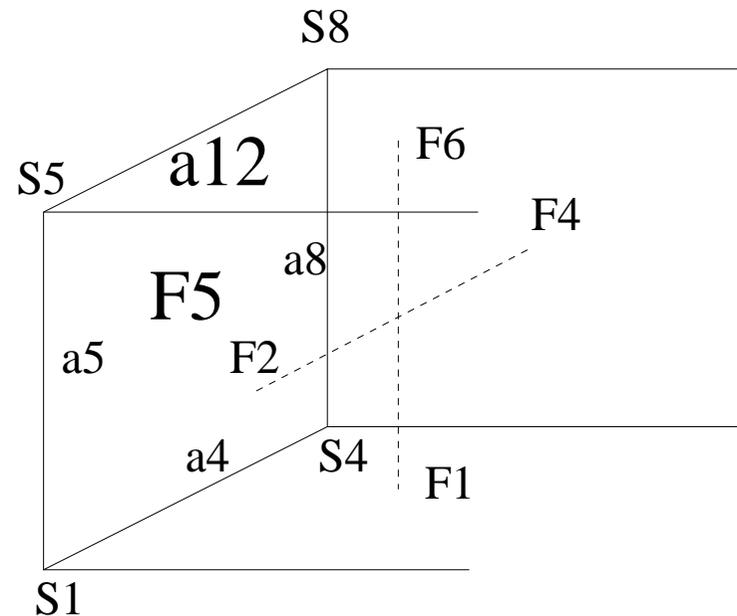
a	F(1)(a)	F(2)(a)
$a_5$	$F_2$	$F_5$
$a_4$	$F_1$	$F_5$
$a_8$	$F_5$	$F_4$
$a_{12}$	$F_5$	$F_6$

a	cw(1)(a)	ccw(1)(a)	cw(2)(a)	ccw(2)(a)
$a_5$	$a_9$	$a_1$	$a_4$	$a_{12}$
$a_4$	$a_1$	$a_3$	$a_8$	$a_5$
$a_8$	$a_{12}$	$a_4$	$a_3$	$a_{11}$
$a_{12}$	$a_5$	$a_8$	$a_{11}$	$a_9$

## Topologie du Modèle

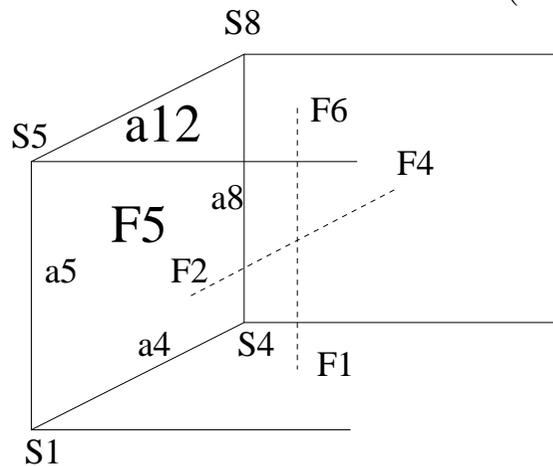
### Hypothèses :

- ▷ On se donne une face ( $F_5$ )
- ▷ On se donne une arête ( $a_{12}$ )



## Topologie du Modèle

Liste des arêtes dans le sens (*cw*)



$a_{12}$

$$F(1)(a_{12}) = F_5$$

$$cw(1)(a_{12}) = a_5$$

$a_5$

$$F(2)(a_5) = F_5$$

$$cw(2)(a_5) = a_4$$

$a_4$

$$F(2)(a_4) = F_5$$

$$cw(2)(a_4) = a_8$$

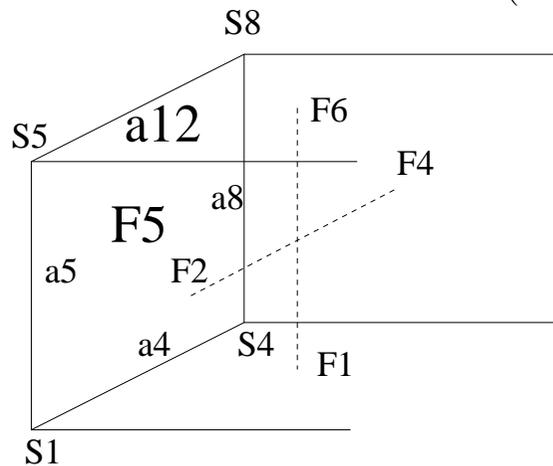
$a_8$

$$F(1)(a_8) = F_5$$

$$cw(1)(a_8) = a_{12}$$

## Topologie du Modèle

Liste des arêtes dans le sens (*ccw*)



$a_{12}$

$$F(1)(a_{12}) = F_5$$

$$ccw(1)(a_{12}) = a_8$$

$a_8$

$$F(1)(a_8) = F_5$$

$$ccw(1)(a_8) = a_4$$

$a_4$

$$F(2)(a_4) = F_5$$

$$ccw(2)(a_4) = a_5$$

$a_5$

$$F(2)(a_5) = F_5$$

$$ccw(2)(a_5) = a_{12}$$

## Relations d'Adjacences

### Hypothèses

- ▷ Une face pointe sur une de ses arêtes adjacentes
- ▷ un sommet pointe sur une de ses arêtes incidentes
- ▷ une arête possède la structure Baumgart

**Problème** : Retrouver les 9 relations d'Adjacences

$$\begin{aligned} S_d\{S\}, A_d\{S\}, F_d\{S\}, \\ S_d\{A\}, A_d\{A\}, F_d\{A\}, \\ S_d\{F\}, A_d\{F\}, F_d\{F\}. \end{aligned}$$

## Adjacence de sommet : $S_d\{S\}$

▷ Un sommet donné :  $S_d$

▷ pointant sur une de ces arêtes incidentes :  $a(S_d)$

Parcours des sommets adjacents dans le sens trigonométrique (*ccw*)

```
 $a \leftarrow a(S_d)$   
si  $(S_d = S(1)(a))$   
  alors  $prem \leftarrow 1$   
         $dern \leftarrow 2$   
  sinon  $prem \leftarrow 2$   
         $dern \leftarrow 1$   
finsi  
   $\{S\} \leftarrow S(dern)(a)$   
   $a \leftarrow ccw(prem)(a)$ 
```

## Adjacence de sommet : $S_d\{A\}$

▷ Un sommet donné :  $S_d$

▷ pointant sur une de ces arêtes incidentes :  $a(S_d)$

Parcours des arêtes incidentes dans le sens trigonométrique

```
{A} ← a(Sd)
si   (Sd = S(1)(a))
      alors   prem ← 1
      sinon   prem ← 2
finsi
      {A} ← ccw(prem)(a)
      a ← ccw(prem)(a)
```

## Adjacence de sommet : $S_d\{F\}$

▷ Un sommet donné :  $S_d$

▷ pointant sur une de ces arêtes incidentes :  $a(S_d)$

Parcours des faces adjacentes dans le sens trigonométrique

```
 $a \leftarrow a(S_d)$   
si  $(S_d = S(1)(a))$   
    alors  $\text{prem} \leftarrow 1$   
    sinon  $\text{prem} \leftarrow 2$   
finsi  
     $\{F\} \leftarrow F(\text{prem})(a)$   
     $a \leftarrow ccw(\text{prem})(a)$ 
```

## **Adjacence d'Arête : $A_d\{S\}$ , $A_d\{A\}$ , $A_d\{F\}$**

La structure d'arête représentant la topologie du modèle

- ▷ pointeur sur les 4 ailes
- ▷ pointeur sur les 2 sommets incidents
- ▷ pointeur sur les 2 face incidentes

l'accès aux informations adjacentes est immédiat

## Adjacence de Face : $F_d\{S\}$

▷ Une face donnée :  $F_d$

▷ pointant sur une de ces arêtes incidentes :  $a(F_d)$

Parcours des sommets adjacents dans le sens trigonométrique

```
 $a \leftarrow a(F_d)$   
si  $(F_d = F(1)(a))$   
  alors   prem  $\leftarrow 1$   
          dern  $\leftarrow 2$   
  sinon  prem  $\leftarrow 2$   
          dern  $\leftarrow 1$   
  
finsi  
   $\{S\} \leftarrow S(\text{dern})(a)$   
   $a \leftarrow ccw(\text{prem})(a)$ 
```

## Adjacence de Face : $F_d\{A\}$

▷ Une face donnée :  $F_d$

▷ pointant sur une de ces arêtes incidentes :  $a(F_d)$

Parcours des arêtes incidentes dans le sens trigonométrique

$\{A\} \leftarrow a(F_d)$

**si**  $(F_d = F(1)(a))$

**alors**  $\text{prem} \leftarrow 1$

**sinon**  $\text{prem} \leftarrow 2$

**finsi**

$\{A\} \leftarrow ccw(\text{prem})(a)$

## Adjacence de Face : $F_d\{F\}$

▷ Une face donnée :  $F_d$

▷ pointant sur une de ces arêtes incidentes :  $a(F_d)$

Parcours des faces incidentes dans le sens trigonométrique

```
 $a \leftarrow a(F_d)$   
si  $(F_d = F(1)(a))$   
  alors  prem  $\leftarrow 1$   
          dern  $\leftarrow 2$   
  sinon  prem  $\leftarrow 2$   
          dern  $\leftarrow 1$   
finsi  
   $\{F\} \leftarrow F(\text{dern})(a)$   
   $a \leftarrow ccw(\text{prem})(a)$ 
```

## Connexion d'Ailes

Toute la topologie du modèle repose sur les ailes

**Hypothèses** : les adjacences d'une arête  $A$  sont connues

- ▷ faces :  $F(1)(A)$  ,  $F(2)(A)$
- ▷ sommets :  $S(1)(A)$  ,  $S(2)(A)$

**Problème** : connexion d'arêtes  $A_i, A_j$  suivant que chaque sommet face est

- ▷ directs :  $S(1)(A)$  ,  $F(1)(A)$
- ▷ indirects :  $S(2)(A)$  ,  $F(2)(A)$  à chacune des deux arêtes  $A_i, A_j$  à connecter

Il existe huit possibilités de connexion

## Connexion d'Ailes

Ailes ( $A_i, A_j$  : aretes)

**si** ( $S(2)(A_i) = S(2)(A_j)$ ) **alors**  
  **si** ( $F(1)(A_i) = F(2)(A_j)$ ) **alors**  
     $cw(1)(A_i) \leftarrow A_j$   
     $ccw(2)(A_j) \leftarrow A_i$   
  **sinon si** ( $F(2)(A_i) = F(1)(A_j)$ ) **alors**  
     $ccw(2)(A_i) \leftarrow A_j$   
     $cw(1)(A_j) \leftarrow A_i$   
**si** ( $S(1)(A_i) = S(1)(A_j)$ ) **alors**  
  **si** ( $F(1)(A_i) = F(2)(A_j)$ ) **alors**  
     $ccw(1)(A_i) \leftarrow A_j$   
     $cw(2)(A_j) \leftarrow A_i$   
  **sinon si** ( $F(2)(A_i) = F(1)(A_j)$ ) **alors**  
     $cw(2)(A_i) \leftarrow A_j$   
     $ccw(1)(A_j) \leftarrow A_i$

## Connexion d'Ailes

**si**  $(S(1)(A_i) = S(2)(A_j))$  **alors**  
  **si**  $(F(1)(A_i) = F(1)(A_j))$  **alors**  
     $ccw(1)(A_i) \leftarrow A_j$   
     $cw(1)(A_j) \leftarrow A_i$   
  **sinon si**  $(F(2)(A_i) = F(2)(A_j))$  **alors**  
     $cw(2)(A_i) \leftarrow A_j$   
     $ccw(2)(A_j) \leftarrow A_i$   
**si**  $(S(2)(A_i) = S(1)(A_j))$  **alors**  
  **si**  $(F(1)(A_i) = F(1)(A_j))$  **alors**  
     $cw(1)(A_i) \leftarrow A_j$   
     $ccw(1)(A_j) \leftarrow A_i$   
  **sinon si**  $(F(2)(A_i) = F(2)(A_j))$  **alors**  
     $ccw(2)(A_i) \leftarrow A_j$   
     $cw(2)(A_j) \leftarrow A_i$

## Structure Arête-Ailée

Structure Baumgart adaptée à une représentation de polyèdres orientés

- ▷ modèle topologique complet (ailes d'arêtes)
- ▷ prise en compte de la géométrie (coordonnées, normales)

parcours topologique des modèles par

- ▷ les neuf relations d'adjacences
- ▷ et la connexion d'ailes suivant les huit cas existants

Problème de construction construction du modèle:

- ▷ opérateurs atomiques : opérateurs d'Euler

## Opérateurs d'Euler

### Problèmes

- ▷ comment créer le modèle ?
- ▷ comment insérer un élément ?
- ▷ comment supprimer un élément ?

en vérifiant la cohérence volumique du modèle

### Solution

- ▷ Opérateurs “atomiques” vérifiant la cohérence topologique des polyèdres

## Première Equation

Pour les modèles de genre zéro, homéomorphe à une sphère, les opérateurs doivent vérifier la relation d' Euler

$$F - A + S = 2$$

- ▷  $F$  : nombre de faces
- ▷  $A$  : nombre d'arêtes
- ▷  $S$  : nombre de sommets

Exemple du cube :  $F = 6$  ,  $A = 12$  ,  $S = 8$

## Deuxième Equation

Pour les modèles plus complexes

$$F - A + S = 2(C - G) + T$$

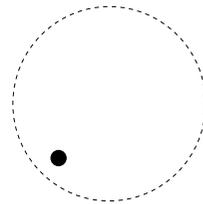
- ▷  $C$  : nombre de composantes connexes
- ▷  $G$  : genre du modèle
- ▷  $T$  : nombre de trous du modèle

En anglais :  $F - E + V = 2(S - G) + H$

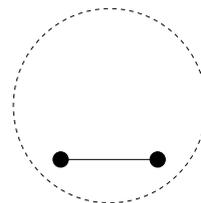
- ▷ **F**ace : **F**ace
- ▷ **E**dge : **A**rête
- ▷ **V**ertex : **S**ommet
- ▷ **S**hell : **C**onnexe
- ▷ **G**enus : **G**enre
- ▷ **H**ole : **T**rou

# Modèles simples

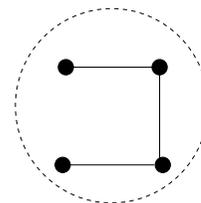
$$F - E + V = 2$$



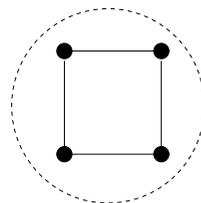
mvsf()



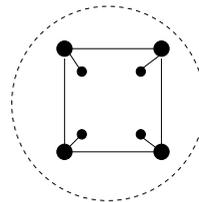
mev()



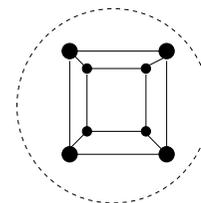
mev()  
mev()



mef()



mev()  
mev()  
mev()  
mev()



mef()  
mef()  
mef()  
mef()

## Modèles simples

Modèles de genre zéro : Trois opérateurs de construction

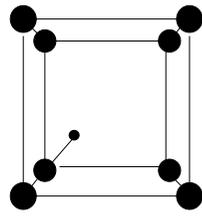
1. **mvsf** () : Créer un Sommet, une Composante, une Face.
2. **mev** () : Créer un Sommet, une Arête.
3. **mef** () : Créer une Arête, une Face.

et les opérateurs inverses

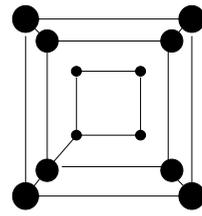
1. **kvsf** () : Détruire un Sommet, une Composante, une Face.
2. **kev** () : Détruire un Sommet, une Arête.
3. **kef** () : Détruire une Arête, une Face.

# Modèles complexes

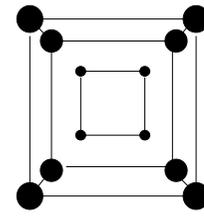
$$F - E + V = 2(S - G) + H$$



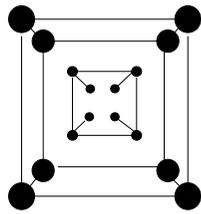
mev()



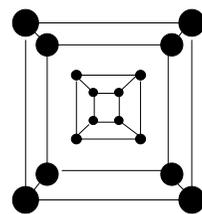
mev()  
mev()  
mev()  
mef()



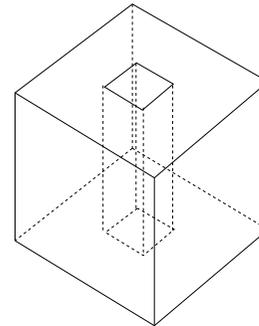
kemh()



mev()  
mev()  
mev()  
mev()



mef()  
mef()  
mef()  
mef()  
kfmgh()



## Modèles complexes

Deux opérateurs supplémentaires

1. **kemh()** : Détruire une Arête, Créer un Trou
2. **kfmgh()** : Détruire une Face, Créer un Trou, un Genre

et leur opposés

1. **mekh()** : Créer une Arête, Détruire un Trou
2. **mfkgh()** : Créer une Face, Détruire un Trou, un Genre

Un opérateur supplémentaire est souvent utilisé

- ▷ **semv()** : Couper une arête en deux, créer un Sommet
  - ▷ **jekv()** : Joindre deux Arêtes, Détruire un Sommet
- équivalent topologique de l'opérateur **mev()**

## Exemple d'opérateur

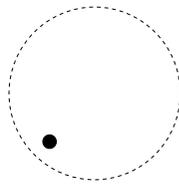
```
semv(Edge edge)
  Edge  new_e;
  Vertex new_v;
  new_v <- mv( s(2)(edge) );
  new_e <- me( edge ) ;
  s(1)(new_e) <- new_v ;
  s(2)(new_e) <- s(2)(edge) ;
  f(1)(new_e) <- f(1)(edge) ;
  f(2)(new_e) <- f(2)(edge) ;

  if ( a(s(2)(edge)) = edge ) then a(s(2)(edge)) <- new_e ;
  s(2)(edge) <- new_v ;
  a(new_v) <- new_e ;

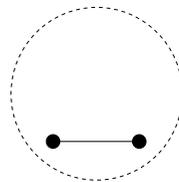
  ailes( cw(1)(edge) , new_e ) ;
  ailes( ccw(2)(edge) , new_e ) ;
  cw(1)(edge) <- new_e ;
  ccw(2)(edge) <- new_e ;
  cw(2)(new_e) <- edge ;
  ccw(1)(new_e) <- edge ;
```

## Deuxième Equation

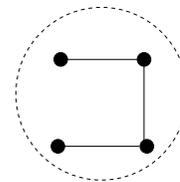
$$F - A + S = 2(C - G) + T$$



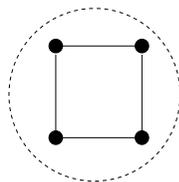
mvsf()



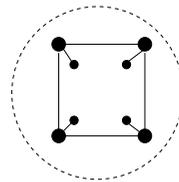
mev()



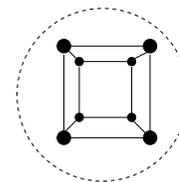
mef()  
mev()



mef()



mev()  
mev()  
mev()  
mev()

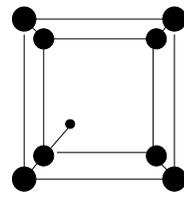


mef()  
mef()  
mef()  
mef()

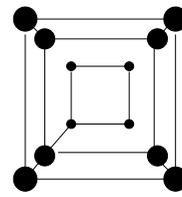
$$F = 6, A = 12, S = 8, C = 1, G = 0, T = 0$$

## Deuxième Equation

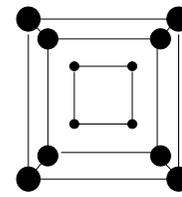
$$F - A + S = 2(C - G) + T$$



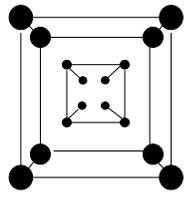
mev()



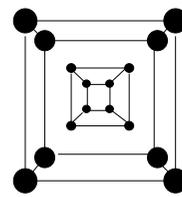
mev()  
mev()  
mev()  
mef()



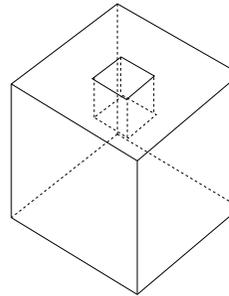
kemh()



mev()  
mev()  
mev()  
mev()



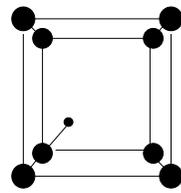
mef()  
mef()  
mef()  
mef()



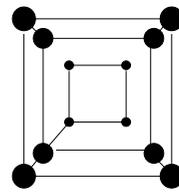
$$F = 11, A = 24, S = 16, C = 1, G = 0, T = 1$$

## Deuxième Equation

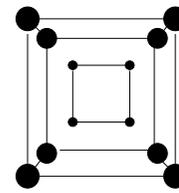
$$F - A + S = 2(C - G) + T$$



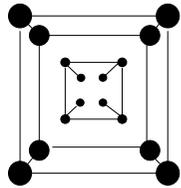
mev()



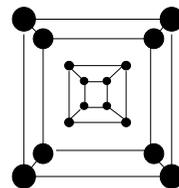
mev()  
mev()  
mev()  
mef()



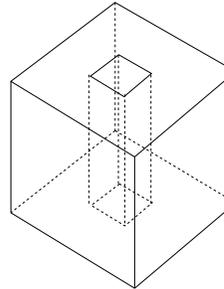
kemh()



mev()  
mev()  
mev()  
mev()



mef()  
mef()  
mef()  
mef()  
kfingh()



$$F = 10, A = 24, S = 16, C = 1, G = 1, T = 2$$

## Conclusion

Les opérateurs d' Euler ont pour but de créer des

▷ entités élémentaires

de manière à réaliser des modèles volumiques qui soient des

▷ polyèdres orientés

Tout opérateur doit vérifier la relation d' Euler

$$F - A + S = 2 (C - G) + T$$

## Conclusion

Pour créer des modèles de genre zéro, trois opérateurs suffisent

1. **mvsf**() : Sommet, Composante, Face
2. **mev**() : Sommet, Arête
3. **mef**() : Arête, Face

Pour les modèles de genre supérieur, deux opérateurs supplémentaires

- ▷ **kemh**() : Détruire Arête, Créer Trou
- ▷ **kfmgh**() : Détruire Face, Créer Trou, Genre

**N.B.** : **semv**() représente le même opérateur (au sens eulérien) que **mev**()

## Conclusion

Objectif premier de la modélisation BReps

- ▷ création de Polyèdres Orientés

Trois structures nécessaires pour la réalisation de ces modèles

- ▷ Faces, Arêtes, Sommets

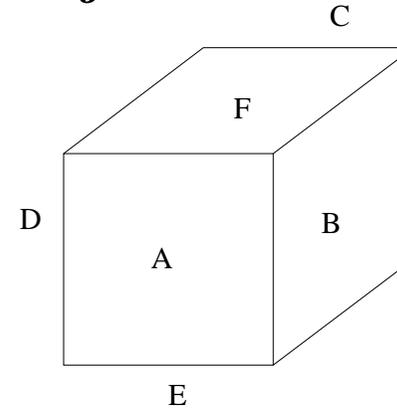
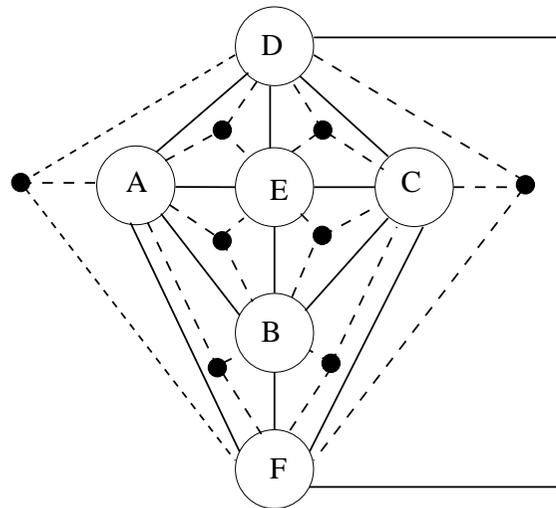
Ces structures doivent définir complètement le modèle d'un point de vue

- ▷ géométrique (coordonnées de sommets, normales de faces, ...)
- ▷ topologique (relations d'adjacences, ...)

La structure Baumgart prend en compte la topologie au niveau de l'arête.

# Conclusion

Topologie au niveau des faces : **Grphe d'Adjacence de Faces**

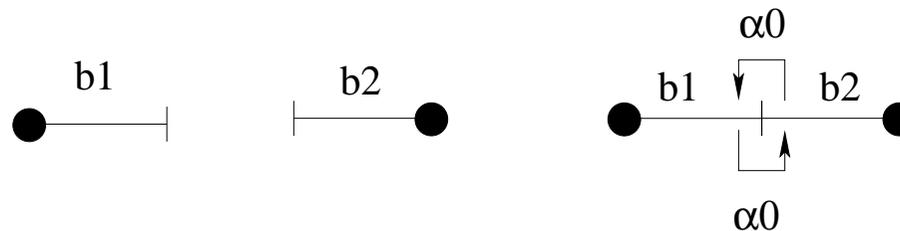


-  Noeud : Face
-  Arc : face induite par aretes
-  Hyperarc : face induite par sommets

## Cartes Topologiques: n-G-maps

▷ Cartes généralisées de dimension N

Element de base : le **Brin**, représente une demi-arête d'un graphe



Tout autre élément d'une carte

Arête, Sommet, Face, Volume, ...

s'en déduit par **orbite** de brin

## Définitions

Une carte généralisée de dimension  $n$  est un  $(n+2)$ -uplet :  $(B, \alpha_0, \alpha_1, \alpha_2, \dots, \alpha_n)$

- ▷  $B$  : ensemble non-vide de brins
- ▷  $\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_{n-1}$  : involutions sans point fixe
- ▷  $\forall i \in \{0, 1, \dots, n-2\}, \forall j \in \{i+2, \dots, n\}, (\alpha_i \alpha_j)$  est une involution
- ▷  $\alpha_n$  est une involution

Une fonction  $f$  est une **involution** sur un ensemble  $B$  ssi :

$$\forall b \in B, f^2(b) = b$$

Une fonction  $f$  est une involution **sans point fixe** (sans bords) ssi :

$$\forall b \in B, f^2(b) = b$$

$$\forall b \in B, f(b) \neq b$$

## 0-G-maps, 1-G-maps

Structure de données de brin d'une 0-G-maps :  $(B, \alpha_0)$

- ▷ pointeur  $(\alpha_0)$  sur un brin
- ▷ pointeur d'immersion 3D ( coordonnées)

Orbite de brin d'une 0-G-map

- ▷ ensemble des brins atteignables à partir d'un brin  $(b)$  en appliquant  $\alpha_0$

Notation :  $\langle \alpha_0 \rangle (b)$  ou  $(b\alpha_0^*)$

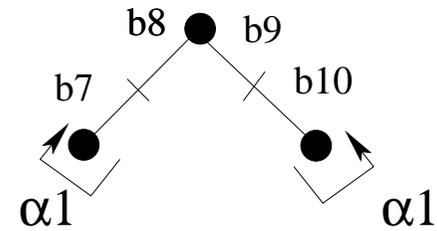
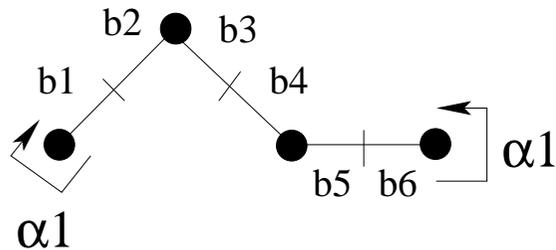
Carte généralisée de dimension 1 :  $(B, \alpha_0, \alpha_1)$

- ▷  $(B, \alpha_0)$  : 0-G-carte sans bords
- ▷  $\alpha_1$  : opération de couture entre brins

Structure de données de brin d'une 1-G-maps

- ▷ celle d'une 0-G-maps
- ▷ pointeur supplémentaire,  $(\alpha_1)$ , sur un brin

## Opération de Couture



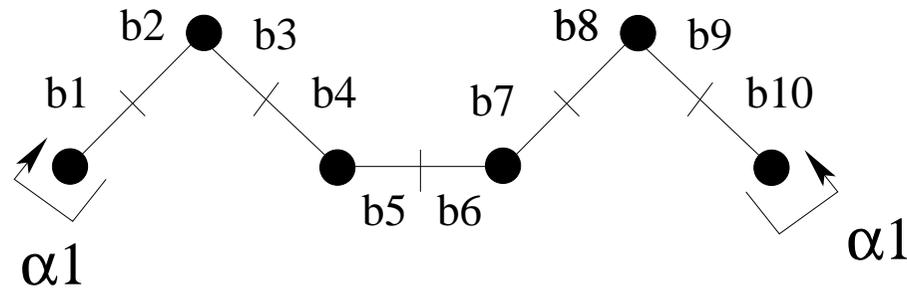
Avant couture des brins  $b_6, b_7$

▷  $\langle \alpha_0 \rangle (b_6) : \{b_6, b_5\}$ ,  $\langle \alpha_0 \rangle (b_7) : \{b_7, b_8\}$

▷  $\langle \alpha_1 \rangle (b_6) : \{b_6\}$ ,  $\langle \alpha_1 \rangle (b_7) : \{b_7\}$

▷  $\langle \alpha_0, \alpha_1 \rangle (b_6) : \{b_6, b_5, b_4, b_3, b_2, b_1\}$ ,  $\langle \alpha_0, \alpha_1 \rangle (b_7) : \{b_7, b_8, b_9, b_{10}\}$

## Opération de Couture

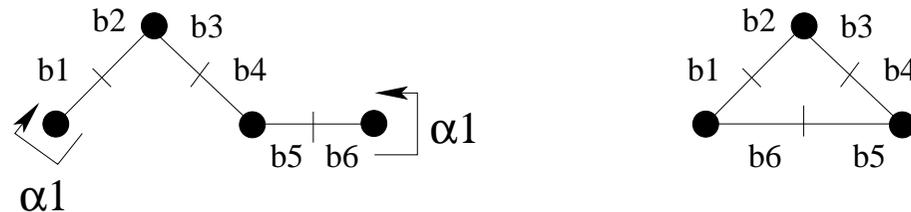


Après couture des brins  $b_6, b_7$

- ▷  $\langle \alpha_0 \rangle (b_6) : \{b_6, b_5\}, \langle \alpha_0 \rangle (b_7) : \{b_7, b_8\}$
- ▷  $\langle \alpha_1 \rangle (b_6) : \{b_6, b_7\} = \langle \alpha_1 \rangle (b_7) : \{b_7, b_6\}$
- ▷  $\langle \alpha_0, \alpha_1 \rangle (b_6) = \langle \alpha_0, \alpha_1 \rangle (b_7) : \{b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}\}$

## Opération de couture

Obtention d'un cycle élémentaire



Avant couture des brins  $b_1, b_6$

$$\triangleright \langle \alpha_1 \rangle (b_1) : \{b_1\}, \langle \alpha_1 \rangle (b_6) : \{b_6\}$$

$$\triangleright \langle \alpha_0, \alpha_1 \rangle (b_1) = \langle \alpha_0, \alpha_1 \rangle (b_6) : \{b_1, b_2, b_3, b_4, b_5, b_6\}$$

Après couture

$$\triangleright \langle \alpha_1 \rangle (b_1) = \langle \alpha_1 \rangle (b_6) : \{b_1, b_6\},$$

$$\triangleright \langle \alpha_0, \alpha_1 \rangle (b_1) = \langle \alpha_0, \alpha_1 \rangle (b_6) : \{b_6, b_5, b_4, b_3, b_2, b_1\}$$

## Opération de couture

Avant couture des brins  $b_1, b_6$

$$\triangleright \alpha_0 : \{(b_1, b_2), (b_3, b_4), (b_5, b_6)\}$$

$$\triangleright \alpha_1 : \{(b_1), (b_2, b_3), (b_4, b_5), (b_6)\}$$

$\alpha_1$  : involution

Après couture

$$\triangleright \alpha_0 : \{(b_1, b_2), (b_3, b_4), (b_5, b_6)\}$$

$$\triangleright \alpha_1 : \{(b_1, b_6), (b_2, b_3), (b_4, b_5)\}$$

$\alpha_1$  : involution sans point fixe

Tout orbite de brin de la 1-G-map définit alors une cellule de dimension 2

## 2-G-maps

Carte généralisée de dimension 2 :  $(B, \alpha_0, \alpha_1, \alpha_2)$

- ▷  $(B, \alpha_0, \alpha_1)$  : 1-G-carte sans bords
- ▷  $\forall b, \langle \alpha_0, \alpha_1 \rangle (b)$  : cellule 2D (faces)
- ▷  $\alpha_2$  : involution telle que
  - ◇  $\alpha_0\alpha_2 = \alpha_2\alpha_0$

$\alpha_2$  : opération de couture de faces par leurs arêtes ( $\alpha_0\alpha_2$ )

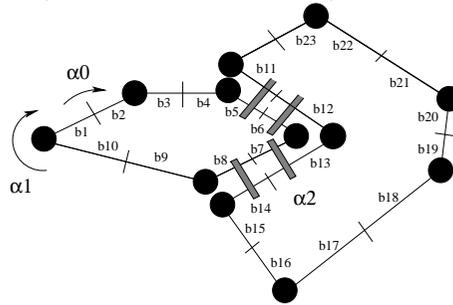
Les 2-G-maps modélisent les subdivisions de surfaces 3D

Association Orbite / Géométrie pour un brin ( $b$ )

- ▷  $\langle \alpha_0, \alpha_1 \rangle (b)$  : Face incidente
- ▷  $\langle \alpha_0 \rangle (b)$  : Arête incidente
- ▷  $\langle \alpha_1, \alpha_2 \rangle (b)$  : Sommet incident

## 2-G-maps

Exemple de carte généralisée  $(B, \alpha_0, \alpha_1, \alpha_2)$



$$\triangleright \langle \alpha_0 \rangle (b_4) = \{b_4, b_3\}$$

$$\triangleright \langle \alpha_0, \alpha_1 \rangle (b_5) = \{b_5, b_6, b_7, b_8, b_9, b_{10}, b_1, b_2, b_3, b_4\}$$

$$\triangleright \langle \alpha_1, \alpha_2 \rangle (b_6) = \{b_6, b_7, b_{13}, b_{12}\}$$

Les 3-G-maps suffisent à modéliser

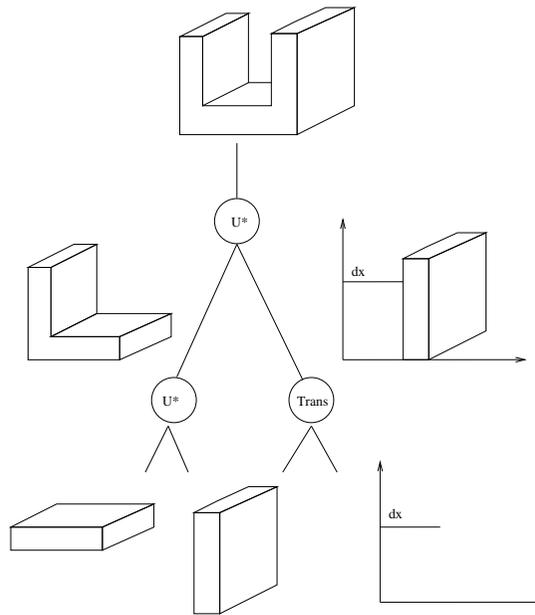
▷ des solides de dimension 3

▷ avec ou sans bords

▷ orientés ou non (anneau de Möbius, Bouteille de Klein, ...)

# Représentation volumique: CSG

Constructive Solid Geometry



## Introduction

**Noeud** : informations sur les transformations et opérations booléennes

```
typedef struct _noeud {
    char* opbool ;
    float mat[4][4] ;
    struct _noeud *fd;
    struct _noeud *fg ;
    Feuille* feuille ;
} Arbre ;
```

**Feuille** : informations sur la géométrie des primitives (solides simples))

```
typedef struct {
    float vecteur[4] ; /* definition de point */
    ...                /* autres informations */
} Feuille ;
```

## Demi-Espaces

Définition d'une primitive : Intersection de demi-espaces

$$\frac{1}{2} \text{ Espace} = \{ \vec{P}(x, y, z) \quad / \quad f(P) \geq 0 \}$$

Exemple du cube

$$E = E_1 \cap E_2 \cap E_3 \cap E_4 \cap E_5 \cap E_6$$

Où :

$$E_1 = \{ P/x \geq 0 \}$$

$$E_2 = \{ P/x \leq 1 \}$$

$$E_3 = \{ P/y \geq 0 \}$$

$$E_4 = \{ P/y \leq 1 \}$$

$$E_5 = \{ P/z \geq 0 \}$$

$$E_6 = \{ P/z \leq 1 \}$$

## Primitives Bornées

Surface délimitant une région fermée de l'espace.

Primitives bornées les plus courantes : **Surfaces Quadriques**

$$f(x, y, z) = [x \ y \ z \ 1] \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Exemple d'une sphère de rayon  $R$

$$f(x, y, z) = [x \ y \ z \ 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -R \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

## Primitives Bornées

**Surface de Révolution** : balayage de courbe autour d'un axe

Dans un repère  $(O, \vec{i}, \vec{j}, \vec{k})$

▷ courbe paramétrique dans le plan  $(O, \vec{i}, \vec{k})$

$$\overrightarrow{OP_1}(u) = p(u) \vec{i} + q(u) \vec{k}$$

▷ axe de rotation :  $\vec{k}$

$$\overrightarrow{OP}(u, v) = \begin{bmatrix} \cos v & -\sin v & 0 & 0 \\ \sin v & \cos v & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p(u) \\ 0 \\ q(u) \\ 1 \end{bmatrix}$$

## Primitives Bornées

**Surface d'Extrusion** : balayage de courbe (fermée) sur une courbe ouverte.

▷ courbe paramétrique dans le plan  $(O, \vec{i}, \vec{j})$

$$\overrightarrow{OP_1}(u) = p(u) \vec{i} + q(u) \vec{j}$$

▷ courbe paramétrique dans le plan  $(O, \vec{j}, \vec{k})$

$$\overrightarrow{OP_2}(v) = q'(v) \vec{j} + r'(v) \vec{k}$$

$$\overrightarrow{OP}(u, v) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & q'(v) \\ 0 & 0 & 1 & r'(v) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p(u) \\ q(u) \\ 0 \\ 1 \end{bmatrix}$$

## Principes de calcul

### Création de modèle

- ▷ primitives bornées (feuilles)
- ▷ transformations géométrique (nœuds)
- ▷ opérations ensemblistes (nœuds)

### Traitement sur le modèle

- ▷ paradigme diviser pour régner (Divide & Conquer)
- ▷ récursivité sur un arbre binaire

## Récurtivité

```
Prop* prop_arbre( Arbre* arbre, void* arg ) {  
    if ( arbre -> opbool == NULL )  
    /* primitive */  
        return prop_prim( arbre, arg );  
  
    /* operation booleenne */  
    else  
        return div_regne( prop_arbre( arbre -> filsd, arg ),  
                          prop_arbre( arbre -> filsg, arg ), arbre -> opbool );  
}
```

Deux fonctions à implémenter

- ▷ Traitement géométrique (Prop\* prop\_prim())
- ▷ Traitement booléen (Prop\* div\_regne())

## Récurtivité

Intérêt pour la synthèse d'image

- ▷ Classification d'Appartenance (modélisation géométrique 3D)
- ▷ Suivi de rayon (Ray-Tracing en Rendu Réaliste)

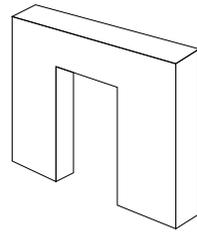
Opérations booléennes usuelles

- ▷ Union ( $\cup$ ,  $+$ )
- ▷ Intersection ( $\cap$ ,  $\&$ )
- ▷ Différence ( $-$ )

Opérations booléennes régulières :  $\cup^*$ ,  $\cap^*$ ,  $-^*$

vérifier la cohérence volumique du modèle

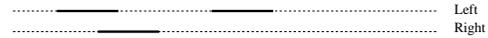
# Classification d'Appartenance



Left



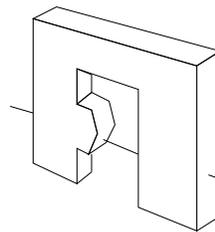
Right



Left  
Right



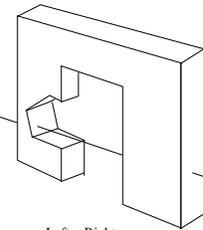
Left + Right  
Left & Right  
Left - Right



Left + Right



Left & Right



Left - Right

## Classification d'Appartenance

- ▷ Traitement géométrique sur les primitives (fonction `prop_prim()`)
  - ◇ Calculs d'intersection : Segment de droite / surface primitive
- ▷ Traitement Booléen (fonction `div_regne()`)  
déterminer les parties du segment ( $S$ ) par rapport aux
  - ◇ fils gauche ( $L$ ) :  $S_{onL}$  ,  $S_{inL}$  ,  $S_{outL}$
  - ◇ fils droit ( $R$ ) :  $S_{onR}$  ,  $S_{inR}$  ,  $S_{outR}$

## Traitement Géométrique

Exemple d'intersection de droite / sphère

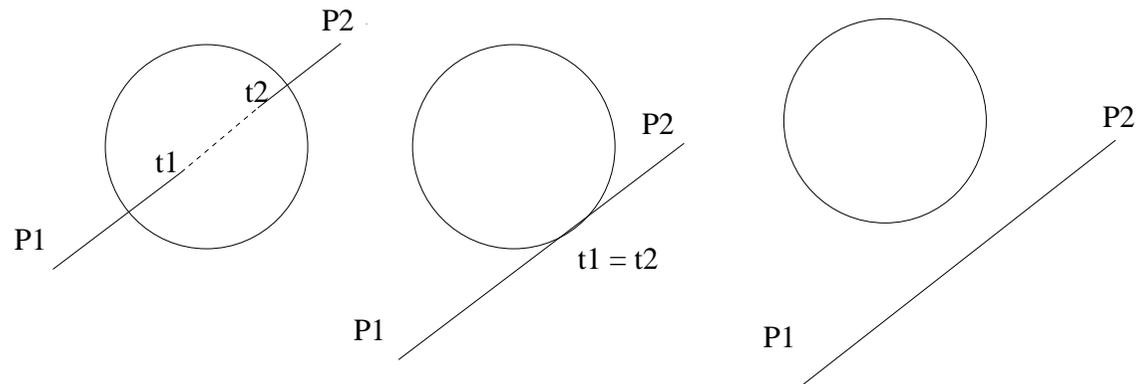
▷ Equation analytique de sphère

$$x^2 + y^2 + z^2 - R^2 = 0$$

▷ Equation paramétrique de segment de droite

$$d(t) = P_1 + t(P_2 - P_1)$$

avec :  $t \in [0, 1]$



## Traitement Géométrique

Système à résoudre

$$\begin{aligned}x^2 + y^2 + z^2 &= R^2 \\x &= x_1 + t(x_2 - x_1) \\y &= y_1 + t(y_2 - y_1) \\z &= z_1 + t(z_2 - z_1)\end{aligned}$$

Solutions :

- ▷ Aucune racine réelle : pas d'intersection
- ▷ Racine double ( $t$ ) : droite tangente
- ▷ Deux racines distinctes ( $t_1, t_2$ ) : intersection

## Traitement Booléen

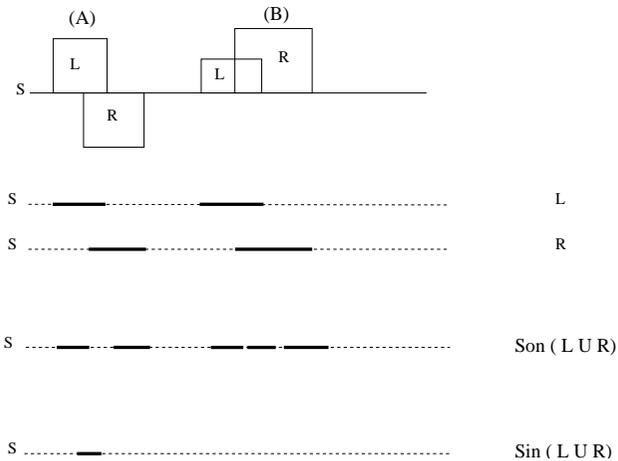
Sur les deux sous-arbres droit ( $R$ ) et gauche ( $L$ )

▷  $S_{onL}, S_{inL}, S_{outL}$

▷  $S_{onR}, S_{inR}, S_{outR}$

SetOperator	Left	Right	Composite
<i>Union</i> ( $\cup^*$ )	$S_{inL}$	$S_{inR}$	$S_{in}(L \cup^* R)$
	$S_{inL}$	$S_{outR}$	$S_{in}(L \cup^* R)$
	$S_{outL}$	$S_{inR}$	$S_{in}(L \cup^* R)$
	$S_{outL}$	$S_{outR}$	$S_{out}(L \cup^* R)$
<i>Intersection</i> ( $\cap^*$ )	$S_{inL}$	$S_{inR}$	$S_{in}(L \cap^* R)$
	$S_{inL}$	$S_{outR}$	$S_{out}(L \cap^* R)$
	$S_{outL}$	$S_{inR}$	$S_{out}(L \cap^* R)$
	$S_{outL}$	$S_{outR}$	$S_{out}(L \cap^* R)$
<i>Difference</i> ( $-^*$ )	$S_{inL}$	$S_{inR}$	$S_{out}(L -^* R)$
	$S_{inL}$	$S_{outR}$	$S_{in}(L -^* R)$
	$S_{outL}$	$S_{inR}$	$S_{out}(L -^* R)$
	$S_{outL}$	$S_{outR}$	$S_{out}(L -^* R)$

# Indétermination ON/ON



▷  $L, R$  non-disjoints :  $L_{on_s}R(R_{on_s}L)$ ,

▷  $L, R$  disjoints :  $L_{on_{op}}R(R_{on_{op}}L)$

Opérations booléennes régulières

$$L \cup^* R = L_{out}R + R_{out}L + L_{on_s}R$$

$$L \cap^* R = L_{in}R + R_{in}L + L_{on_s}R$$

$$L -^* R = L_{out}R + R_{in}L + L_{on_{op}}R$$

## Conclusion

La modélisation de type C.S.G. regroupe les notions de

- ▷ structure d'arbre binaire
- ▷ transformations disponibles
- ▷ demi-espaces / primitives bornées
- ▷ régularité d'opérations booléennes

Les problèmes d'une telle représentation sont

- ▷ quelles primitives ?
- ▷ transformations, déformations ?
- ▷ calculs d'intersections (droite/primitive, courbe/surface, ...)
- ▷ indétermination ON / ON ?

## Modélisation Fractale

Modèle Fractale : objets auto-similaires (Mandelbrot)

- ▷ mouvement brownien
- ▷ côtes de Bretagne, montagnes
- ▷ poumons, reins
- ▷ nuages, choux-fleurs

L-Systèmes : modèles graphicaux

- ▷ fractals
- ▷ croissance de plantes

## Dimension Fractale

Dimension Euclidienne entière

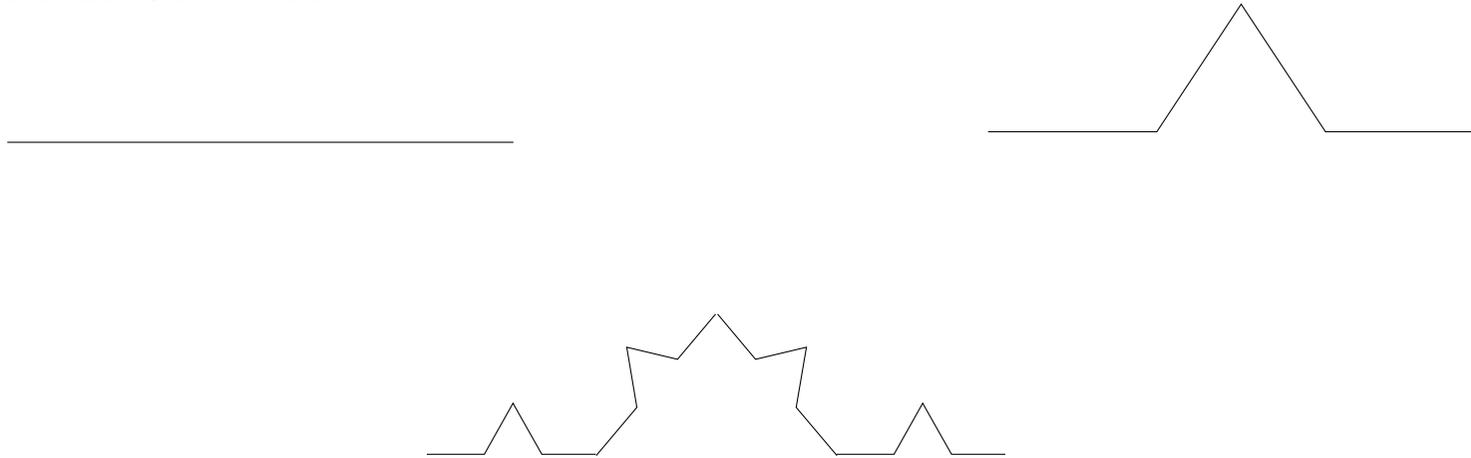
- ▷ ligne : dimension 1
- ▷ rectangle : dimension 2
- ▷ cube : dimension 3

Dimension fractale:  $N = (\frac{1}{L})^d$

- ▷  $N$  : nombre de pas nécessaires pour parcourir la courbe
- ▷  $L$  : longueur (résolution) fixée pour le pas
- ▷  $d$  : dimension fractale

## Dimension Fractale

Courbe de Van Koch



diviser la longueur  $L$  par 3 revient à multiplier le pas  $N$  par quatre

$$4\left(\frac{1}{L}\right)^d = \left(\frac{L}{3}\right)^{-d}$$

la dimension fractale est donc :  $d = \frac{\log(4)}{\log(3)} = 1.2618\dots$

## Dimension Fractale

Un carré de dimension 2 ( $d$ ) est auto-similaire à

▷ quatre ( $N$ ) copies

▷ deux fois ( $L$ ) plus petites que lui

$$4\left(\frac{1}{L}\right)^d = \left(\frac{L}{2}\right)^{-d}$$

La dimension (fractale) est :  $d = \frac{\log(4)}{\log(2)} = 2$

Cube de dimension 3 (auto-similaire à 8 copies 2 fois plus petites)

$$8\left(\frac{1}{L}\right)^d = \left(\frac{L}{2}\right)^{-d}$$

La dimension (fractale) est :  $d = \frac{\log(8)}{\log(2)} = 3$

## Modélisation de Plantes

L-Systems: Aristid Lindenmayer et Przemyslaw Prusinkiewicz

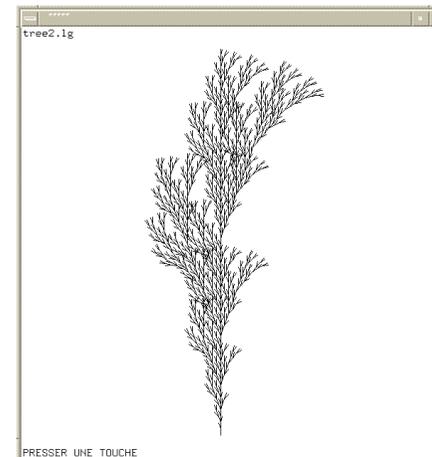
- ▷ un motif
- ▷ un générateur de motif

Développement de plantes représenté par:

- ▷ une grammaire
- ▷ des règles de réécriture

```
depth 5 ;  
angle 20 ;
```

```
F ;  
F --> F [ + F ] F [ - F ] [ F ] ;
```





## Définition

- ▷ L-Système :  $G = \langle V, \omega, P \rangle$
- ▷  $V$  : alphabet
- ▷  $V^*$  : ensemble de tous les mots sur  $V$
- ▷  $V^+$  : ensemble non-vidé de mots sur  $V$
- ▷  $\omega \in V^+$  : axiome (motif)
- ▷  $P \subset V \times V^*$  : ensemble de production (générateurs)
- ▷ une production  $(a, \chi) \in P$ , notée  $a \rightarrow \chi$

## Modèle de la Tortue

### Interprétation

- ▷ état de la tortue :  $(x, y, \alpha)$ 
  - ◇  $(x, y)$  : position
  - ◇  $\alpha$  : direction
- ▷ pas de la tortue :  $d$
- ▷ virage :  $\delta$

## Modèle de la Tortue

### Comportement

- ▷  $F$  : avancer d'un pas  $(x', y', \alpha)$ 
  - ◇  $x' = x + d\cos\alpha$
  - ◇  $y' = y + d\sin\alpha$
  - ◇ ligne entre  $(x, y)$  et  $(x', y')$
- ▷  $f$  : avancer d'un pas sans tracer de ligne
- ▷  $+$  : tourner à gauche  $(x, y, \alpha + \delta)$
- ▷  $-$  : tourner à droite  $(x, y, \alpha - \delta)$

## Modèle de la Tortue

Exemple : Ile quadratique de KOCH

$$\triangleright \omega : F - F - F - F$$

$$\triangleright p : F \rightarrow F - F + F + FF - F - F + F$$

avec

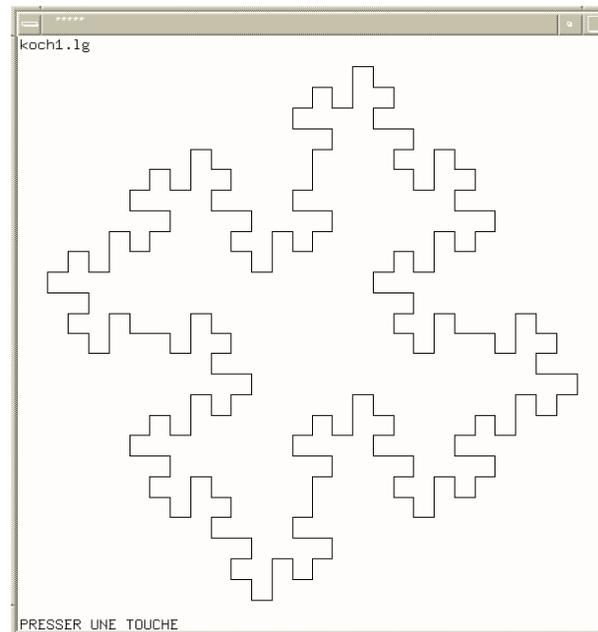
$$\triangleright \delta = 90^\circ$$

$$\triangleright d \leftarrow \frac{d}{4} \text{ à chaque dérivation}$$

## Modèle de la Tortue

```
depth 2 ;  
angle 90 ;
```

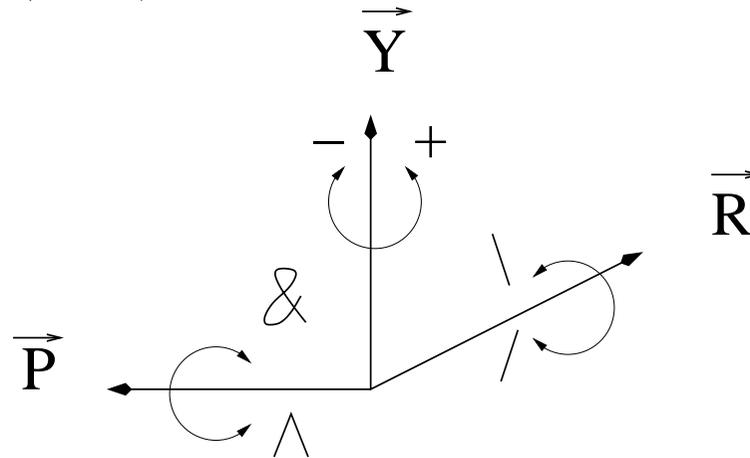
```
F - F - F - F ;  
F --> F - F + F + F F - F - F + F ;
```



## Modèle de la Tortue

Extension 3D du modèle de la tortue : repère  $\vec{R}$ ,  $\vec{P}$ ,  $\vec{Y}$

- ▷  $\vec{R}$  : direction d'avancée (roulis/roll)
- ▷  $\vec{P}$  : virages (tanguage/pitch)
- ▷  $\vec{Y}$  : altitude (lacet/yaw)



## **Extension 3D**

Rotation autour de l'axe de roulis (Roll)

$$[R_R(\alpha)] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix}$$

Rotation autour de l'axe de tangage (Pitch)

$$[R_P(\alpha)] = \begin{bmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{bmatrix}$$

Rotation autour de l'axe des lacets (Yaw)

$$[R_Y(\alpha)] = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

## Symboles

1. Lacet (Yaw) :

(a) + : tourner à gauche d'un angle  $\delta$  ( $R_Y(\delta)$ )

(b) - : tourner à droite d'un angle  $\delta$  ( $R_Y(-\delta)$ )

2. Tanguage (Pitch):

(a) & : tourner en hauteur d'un angle  $\delta$  ( $R_P(\delta)$ )

(b)  $\Lambda$  : tourner en bas d'un angle  $\delta$  ( $R_P(-\delta)$ )

3. Roulis (Roll) :

(a) \ : tourner autour du sens de déplacement d'un angle  $\delta$  ( $R_R(\delta)$ )

(b) / : tourner autour du sens de déplacement d'un angle  $-\delta$  ( $R_R(-\delta)$ )

4. | : virage à 180 degré ( $R_Y(180^\circ)$ )

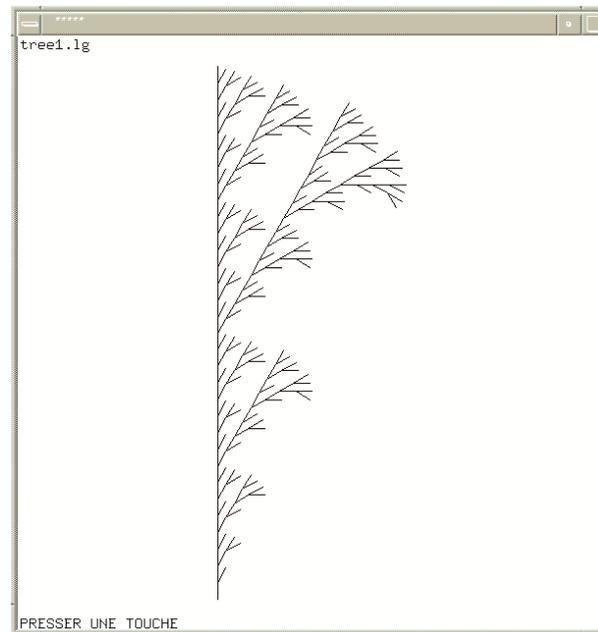
## Croissance d'arbre

Structure d'arbre : deux nouveaux symboles nécessaires

- ▷ [ : mémorisation des caractéristiques courantes
  - ◇ position
  - ◇ orientation
  - ◇ couleur
  - ◇ largeur de ligne
  - ◇ ...
- ▷ ] : restauration de l'état mémorisé

## Croissance d'arbre

```
depth 5 ;  
angle 30 ;  
  
F ;  
F --> F [ - F ] F ;
```



## L-Système Contextuel

Prise en compte de l'environnement :

▷ symboles :  $\langle , \rangle$

▷ production :  $a_l \langle a \rangle a_r \rightarrow \chi$

◇  $a \rightarrow \chi$

◇ **si et seulement si**  $a$  est précédé par  $a_l$  et suivi par  $a_r$

Exemple

▷  $\omega$  : *baaaaaa*

▷  $p_1$  :  $b \langle a \rangle \rightarrow b$

▷  $p_2$  :  $b \rightarrow a$

## L-Système Contextuel

Ignorer en fonction du contexte

▷ **#ignore:** + -

L-Systèmes Propagation de la racine aux sommet

▷ # ignore: + -

▷  $\omega$  :  $F_b[+F_a]F_a[-F_a]F_a[+F_a]F_a$

▷  $F_b < F_a \rightarrow F_b$

Propagation du sommet à la racine

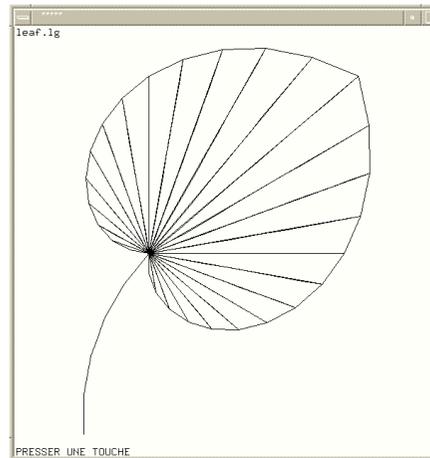
▷ # ignore: +-

▷  $\omega$  :  $F_a[+F_a]F_a[-F_a]F_a[+F_a]F_b$

▷  $F_a > F_b \rightarrow F_b$

## Autres Symboles

- ▷ structure de feuille : { et } (frontières)
- ▷ diminution de la taille de segments : ! (diminution)
- ▷ couleur : ' (incrémentation index)
- ▷ fonctions de croissances
- ▷ ...



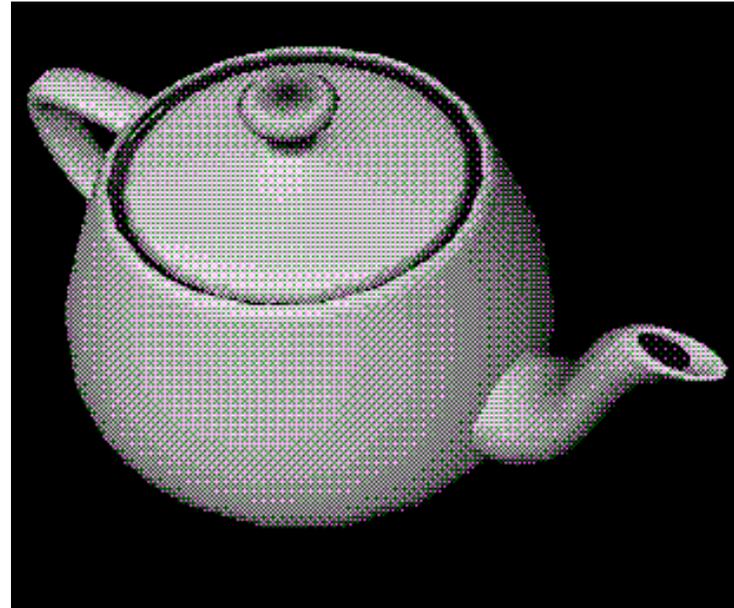
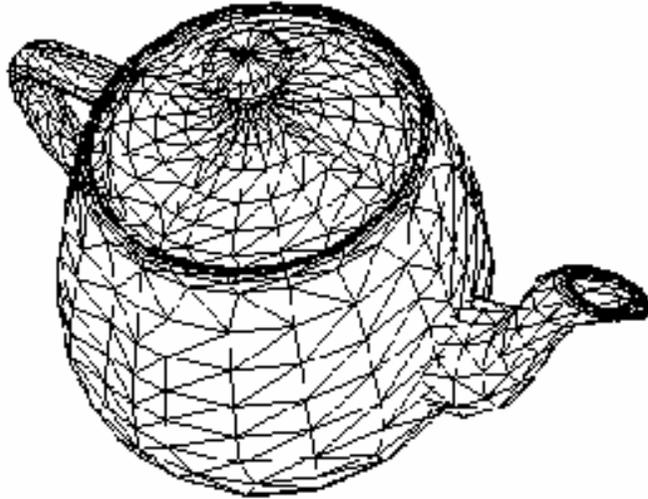
## **Modélisation**

- ▷ Polyèdres : représentation de solides par frontières (Breps)
- ▷ Modèles Topologiques
  - ◇ Baumgart (adjacence d'arête)
  - ◇ Graphe d'Adjacence de Face
  - ◇ Cartes Généralisés de dimension n (n-G-maps)
- ▷ Représentation sous forme d'arbres CSG
- ▷ Modèles de décomposition
  - ◇ voxels, Octree (équivalent 2D : pixels, Quadtree)
  - ◇ cellules (voxels de formes différentes)

## **Modélisation**

- ▷ Modèles de Balayage
  - ◇ simples (extrusion, révolution)
  - ◇ hybrides (simple + CSG)
  - ◇ généralisé (simple + déformations)
- ▷ autres représentations
  - ◇ courbes & surfaces (Bézier, splines, B-Splines, NURBS, ...)
  - ◇ fractals (objets auto-similaires)
  - ◇ graphals, L-Systems
  - ◇ systèmes de particules
  - ◇ meta-balls

## Conclusion



## Modélisation

### Ouvrages

- ▷ **Couwenbergh J.P.**: "La Synthèse d'image", Marabout 1998 ([www.marabout.com](http://www.marabout.com))
- ▷ **Cambray B.**: "Modélisation 3D: Etat de l'art", Rapport MASI Octobre.92  
Institut Blaise Pascal PARIS VI  
<http://fermivista.math.jussieu.fr/cgi-bin/fv-query>
- ▷ **Prusinkiewicz P., Lindenmayer A.**: "The algorithmic Beauty of plants", Springer-Verlag 1990

### Et les bonnes adresses

- ▷ <http://www.swin.edu.au/astronomy/pbourke>
- ▷ <http://www.linux3D.org>
- ▷ <http://www.opencascade.com>
- ▷ <http://www.blender.nl>
- ▷ <http://www.hsak.ac.at/OpenGeometry> : bibliothèque C++ pour modélisation 3D
- ▷ <http://www.ktx.com> : logiciel 3D Studio Max
- ▷ <http://www.cirad.fr/amap> : logiciel de modélisation de plantes
- ▷ <http://www.cpsc.ucalgary.ca/projects/bmv/vlab>: les L-systems