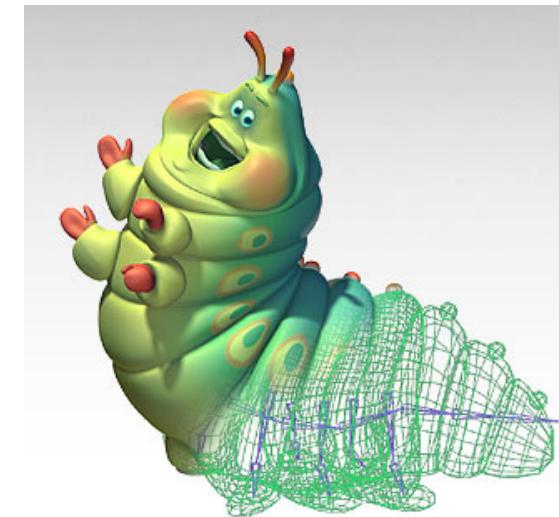
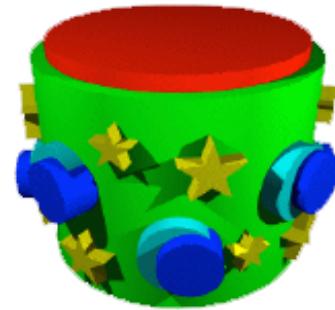


# Partitions spatiales CSG vs Brep Maillages

Fred Boudon



Inspiré des présentations de Barthe, Bessière, Cerutti, etc.

# Plan

- Partitions spatiales
  - Octree régulier et adaptatif
  - KDTree
  - Bounding Volume Hierarchy
- Modélisation géométrique de solides
  - Modèle CSG vs Modèle B-Rep
  - Les maillages
    - Définition et propriétés
    - Représentations internes
      - Représentations optimisées pour l'espace mémoire
      - Représentations optimisées pour le parcours du maillage

# Partitionnements spatiaux

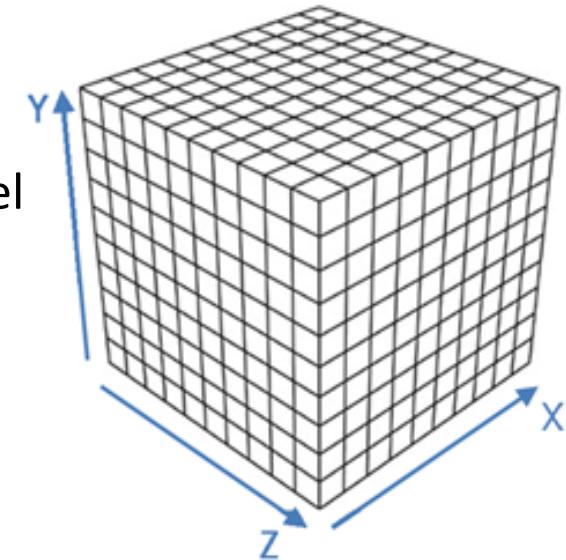
Organiser les objets dans l'espace pour:

- Localiser rapidement leur position (indexation spatiale)
  - Eliminer les objets non-visibles de la scène
  - Déetecter efficacement les collisions
- Stocker explicitement leur définition
  - Représentation géométrique par matrice (image 3D)
  - Représentation compressée par octree

# Représentation par voxels

Représentation d'un objet sur une subdivision de l'espace par une grille régulière (matrice d'énumération spatiale)

- Voxel : volume (cellule) élémentaire de taille fixe
- "Image" binaire : présence/absence de l'objet à chaque voxel
- Très présent en imagerie biomédicale

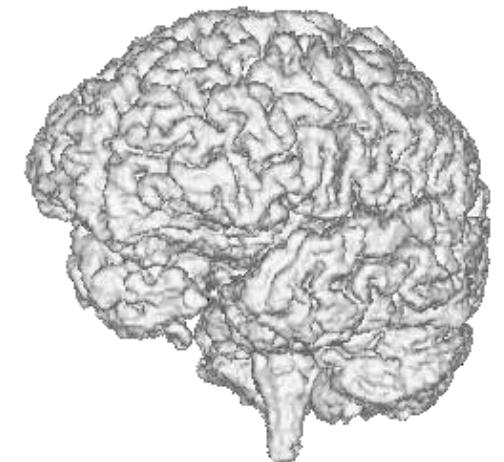


Les +

- Représentation très simple de l'objet

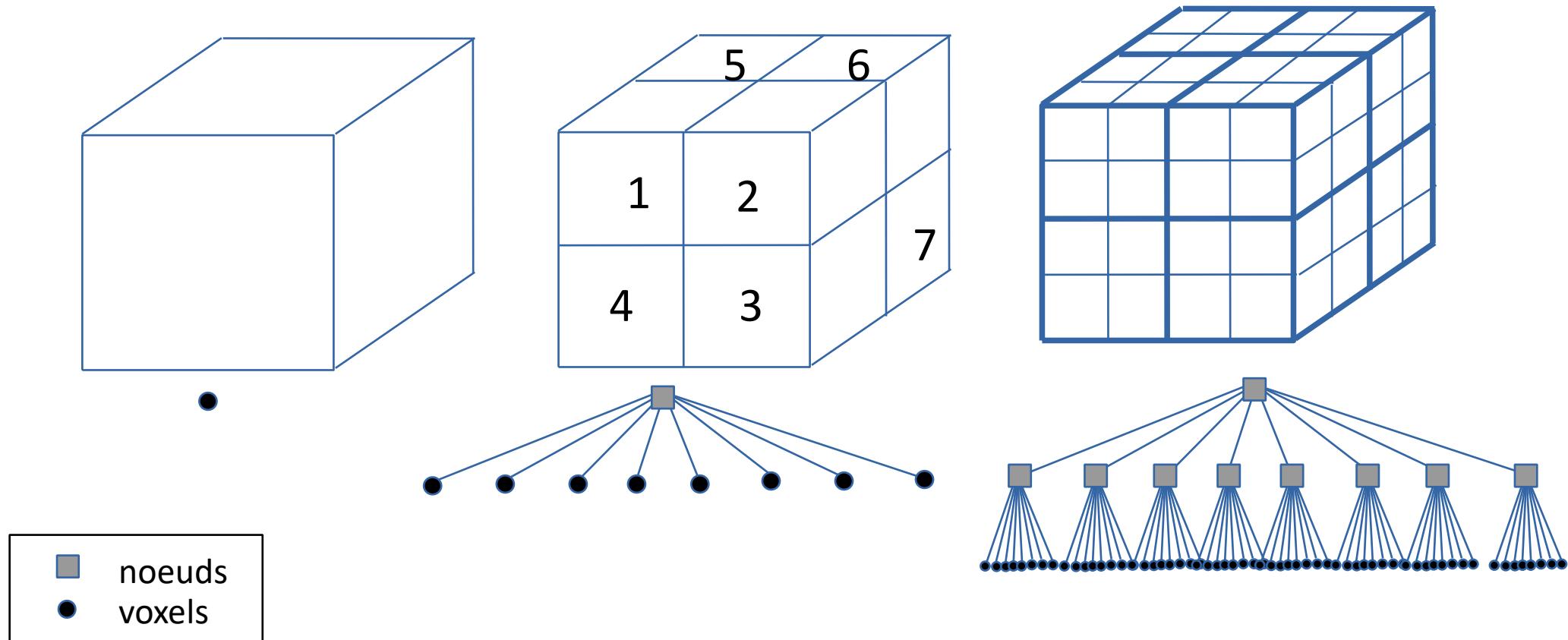
Les -

- Finesse de définition fixe (taille de voxel)
- Peu de facilités d'indexation
- Extrêmement gourmand en mémoire
- Rendu (surfique) complexe



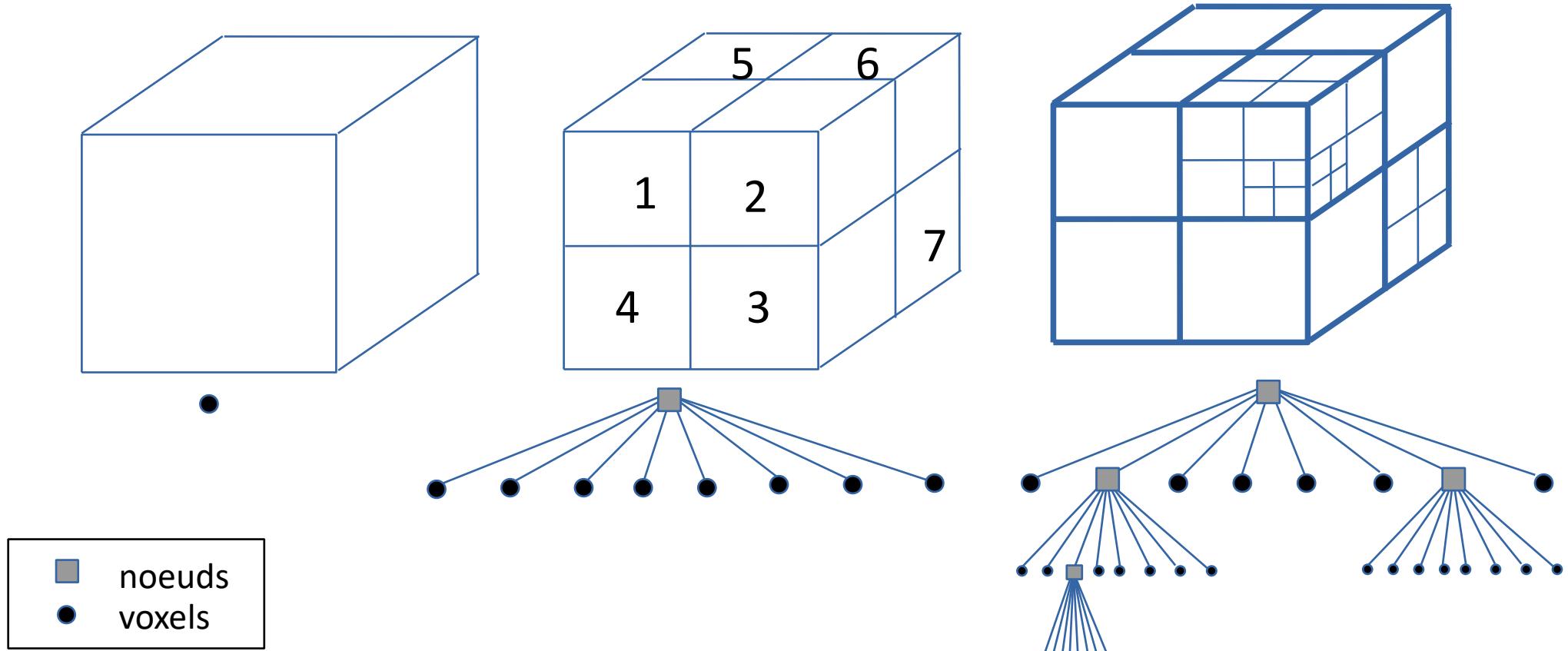
# Octree régulier

Comme son nom l'indique, un octree est un arbre à huit branches. Un octree régulier subdivise de façon récursive un volume cubique en huit sous-cubes de tailles égales. Les feuilles de l'octree sont appelées des « voxels ».



# Octree adaptatif

Dans un octree adaptatif, la profondeur de chaque branche peut être de taille différente, ceci permet de subdiviser l'espace de départ de façon irrégulière.



# Représentation surfacique par octree

**Octree régulier** : on subdivise jusqu'à la précision souhaitée et

- soit la cellule n'est pas sécante à la surface et la feuille est vide (valeur 0 par exemple),
- soit elle est sécante et la feuille est pleine (valeur 1 par exemple).

**Octree adaptatif** :

- soit la cellule n'est pas sécante à la surface : c'est une feuille vide de l'octree,
- soit la cellule est sécante à la surface : si on est au niveau de précision désiré, c'est une feuille pleine de l'octree, sinon, c'est un noeud qui va être subdivisé.

# Représentation volumique par octree

**Octree régulier** : on subdivise jusqu'à la précision souhaitée et

- soit elle est *sécante* et la feuille est pleine (valeur 0 par exemple),
- soit elle est à *l'intérieure* de l'objet (valeur 1 par exemple),
- soit elle est à *l'extérieure* de l'objet (valeur -1 par exemple).

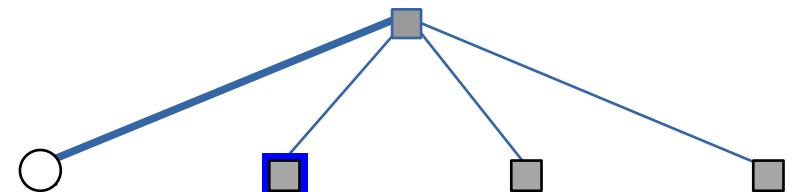
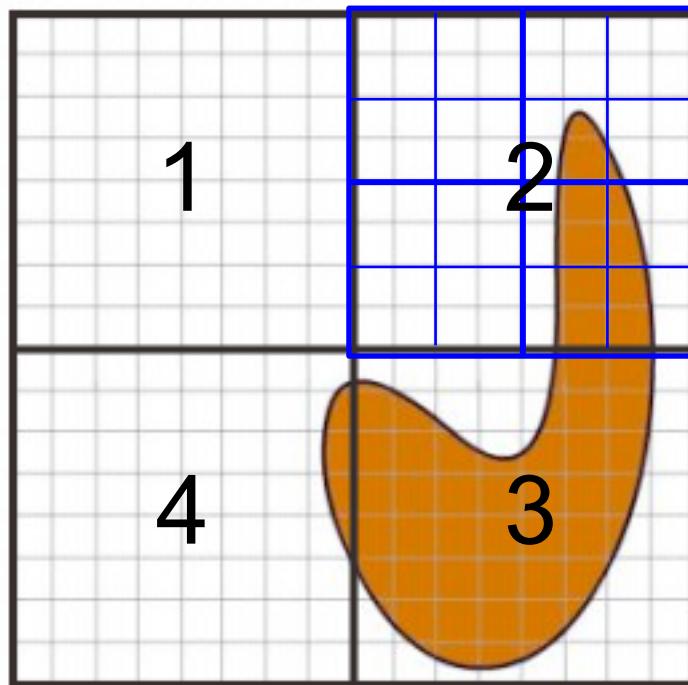
**Octree adaptatif** :

- soit la cellule est *sécante* à la surface :
  - si on est au niveau de précision désiré, c'est une feuille pleine de l'octree
  - sinon, c'est un noeud qui va être subdivisé,
- soit la cellule n'est *pas sécante* à la surface :
  - c'est soit une feuille « extérieure »,
  - soit une feuille « intérieure ».

# Illustration sur un quadtree

Un quadtree est un arbre à quatre branches. C'est l'équivalent de l'octree en deux dimensions.

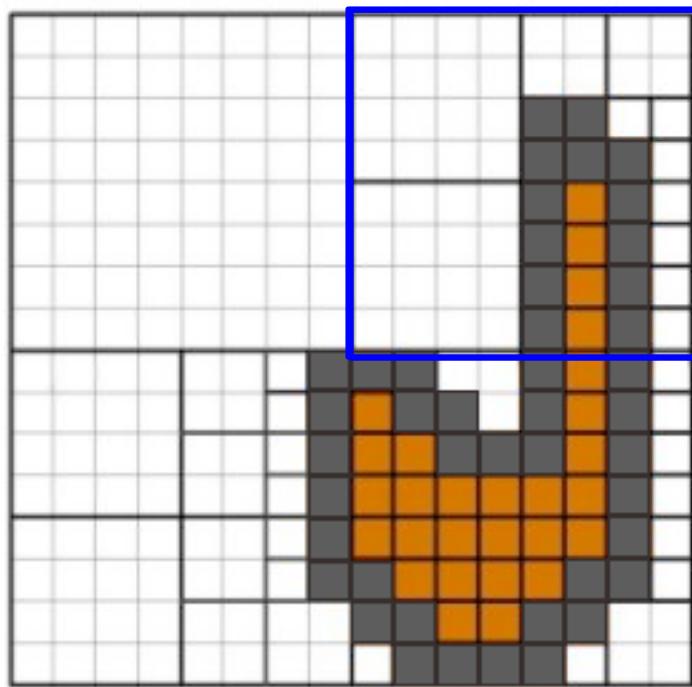
- Dessinez les feuilles du quadtree adaptatif de profondeur quatre représentant l'objet ci-dessous en choisissant une représentation volumique et ne développant que la deuxième branche



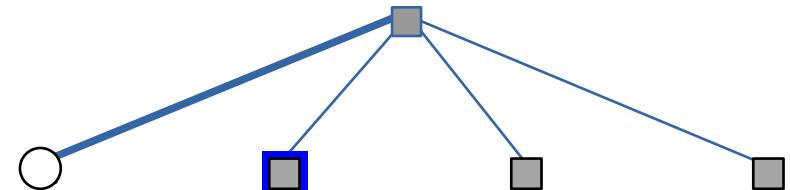
# Illustration sur un quadtree

Un quadtree est un arbre à quatre branches. C'est l'équivalent de l'octree en deux dimensions.

- Dessinez les feuilles du quadtree adaptatif de profondeur quatre représentant l'objet ci-dessous en choisissant une représentation volumique et ne développant que la deuxième branche



1	2
4	3



# Octree : +/-

## Les +

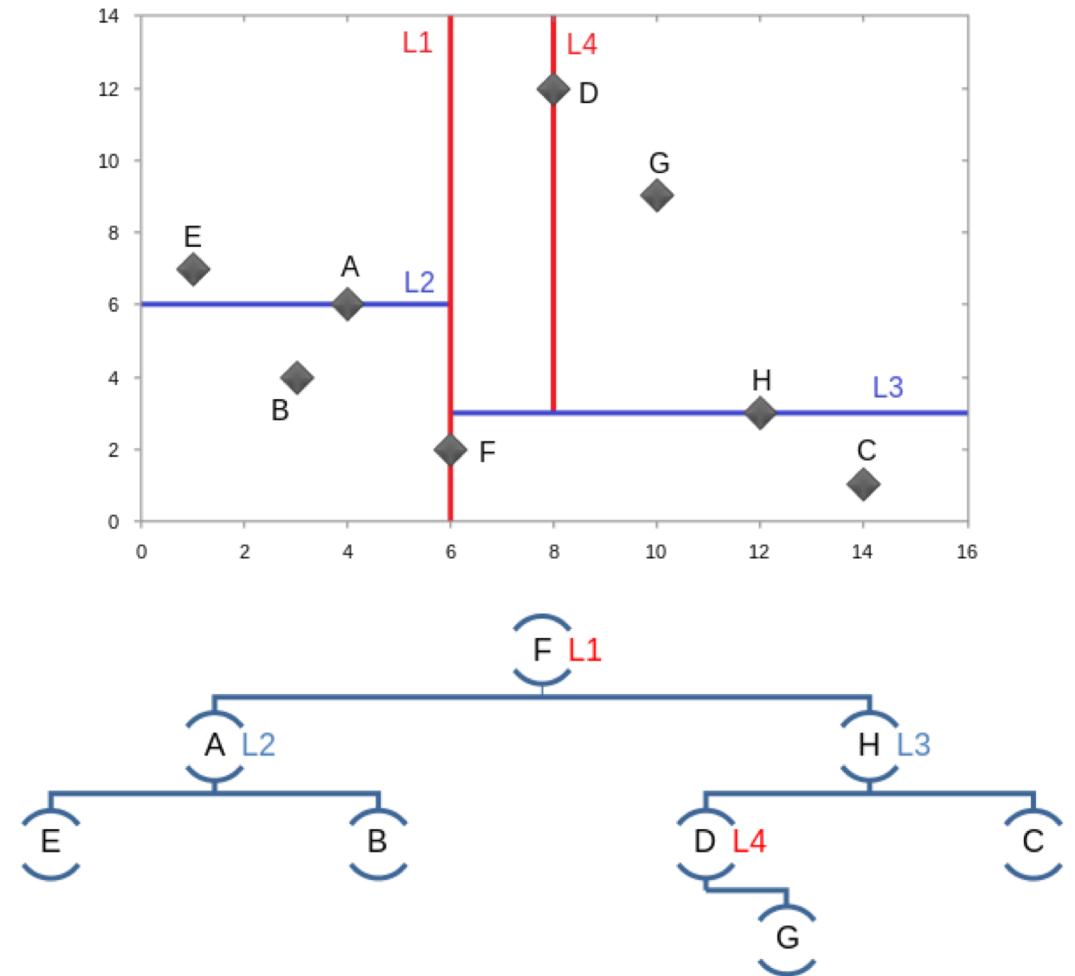
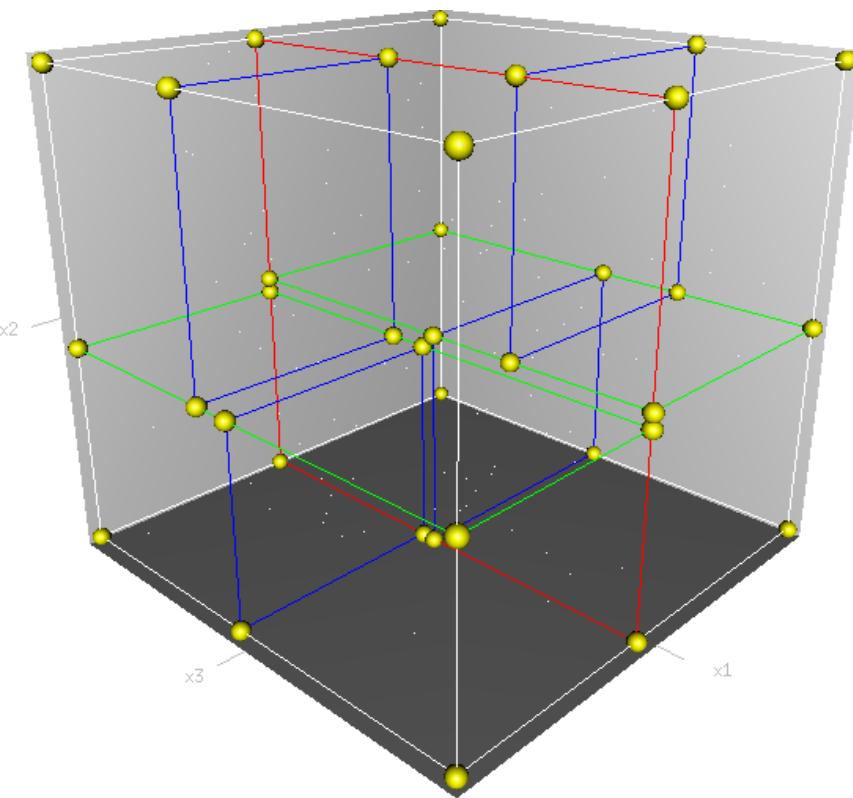
- Représentation hiérarchique de l'objet : il peut être affiché à différentes résolutions.
- Possibilité de représentation volumique.
- Simplicité de positionnement d'un volume par rapport à l'objet : sécant ou non (éventuellement intérieur/extérieur).
- Construction et parcours récursifs simples.

## Les -

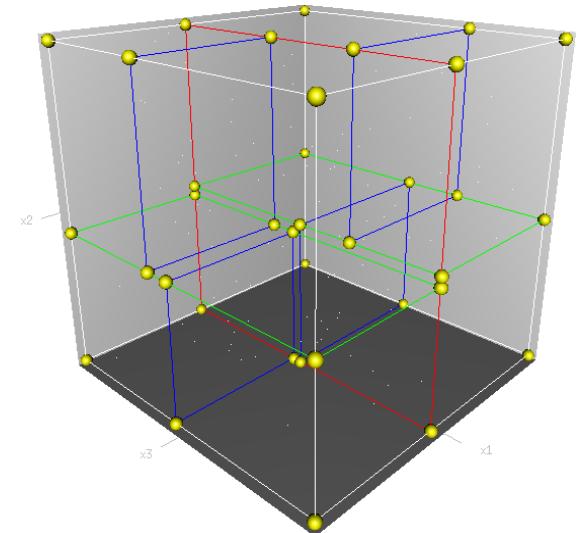
- Visualisation surfacique des voxels ?
- Rendu temps réel pour des scènes complexes ?
- Coup de stockage excessif.
- Pour objets dynamiques, nécessité de recalculer l'octree à chaque pas

# KDTree

- Arbre binaire partitionnant l'espace une dimension après l'autre pour indexer un ensemble de points

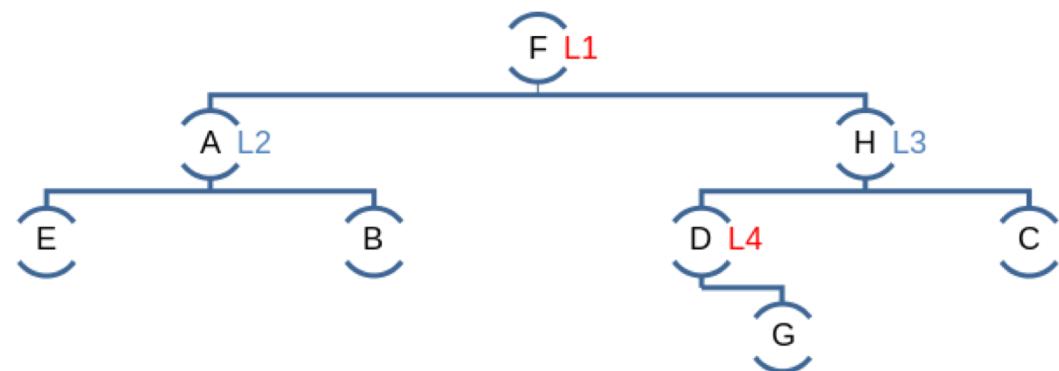
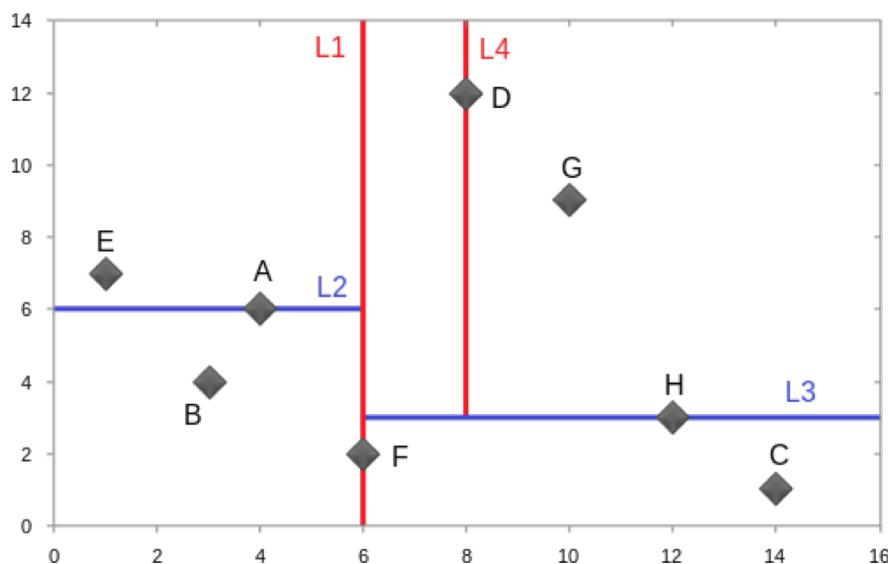


# KDTree



## Création à partir de points

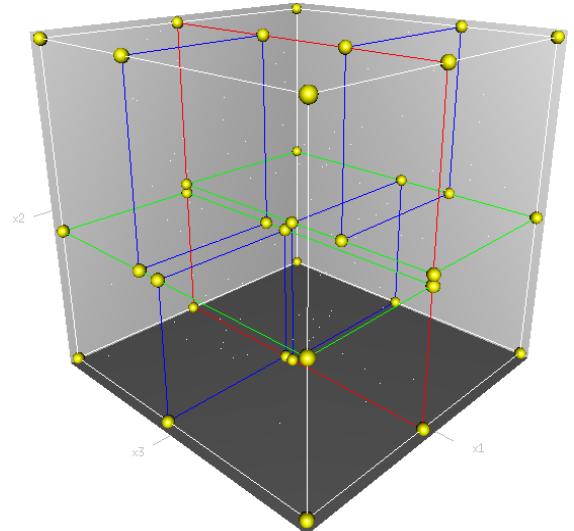
- Construction récursive, en changeant de dimension à chaque pas
- Paramètre : nombre minimum de points dans une cellule
- Choix d'un point pivot à chaque étape pour diviser les points en deux



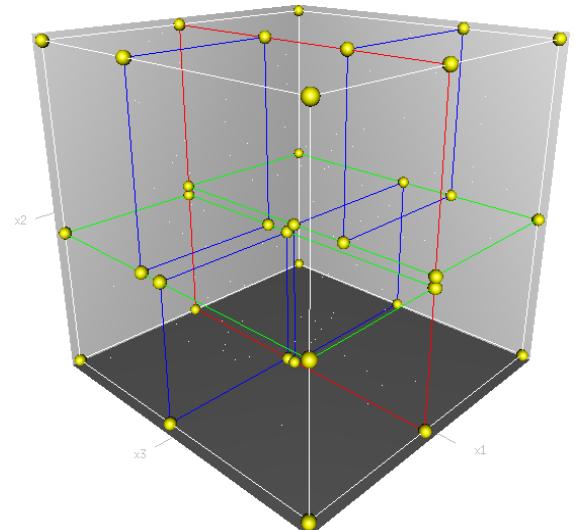
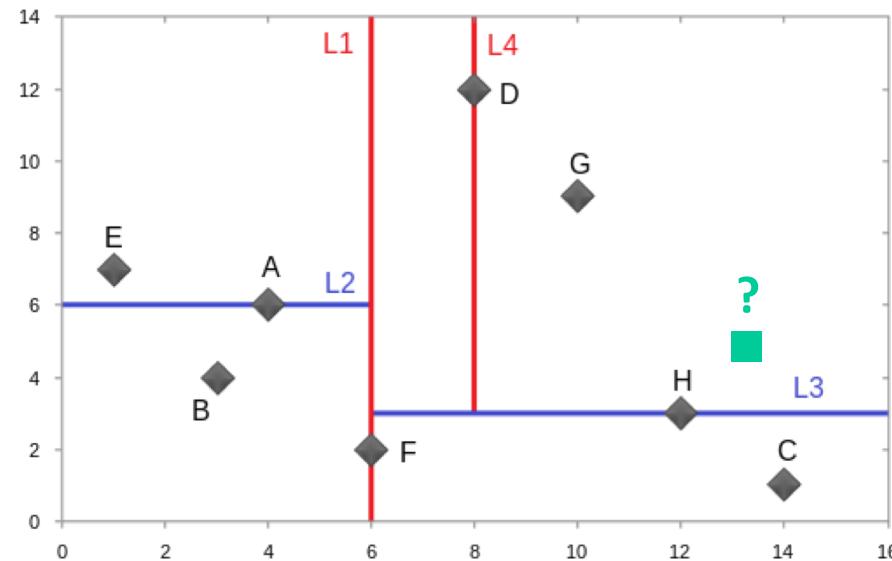
# KDTree

```
class KDNode: pass
```

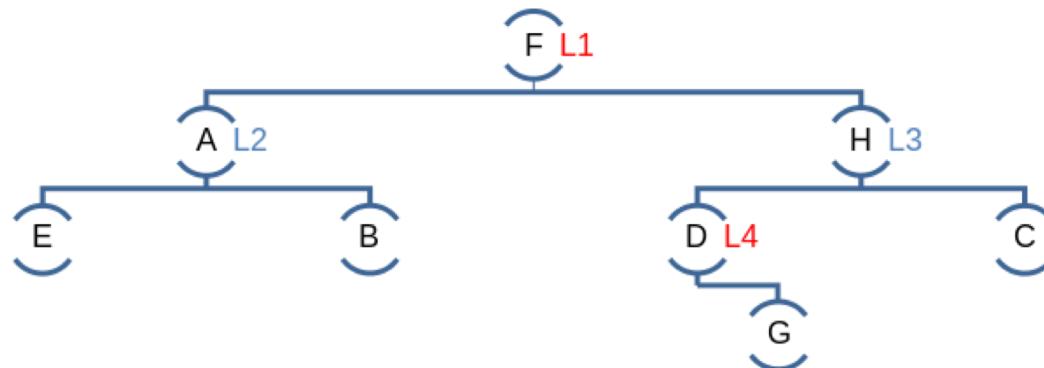
```
def create_kdtree(point_list, minbucketsize = 3, depth = 0):
    if len(point_list) > 2*minbucketsize:
        # Select axis based on depth so that axis cycles through all valid values
        axis = depth % 3
        # Sort point list and choose median as pivot element
        point_list.sort(key=lambda point: point[axis])
        median = len(point_list) // 2 # choose median
        # Create node and construct subtrees
        node = KDNode()
        node.pivot      = point_list[median]
        node.axis       = axis
        node.left_child = create_kdtree(point_list[:median], minbucketsize, depth + 1)
        node.right_child = create_kdtree(point_list[median + 1:], minbucketsize, depth + 1)
        return node
    else :
        return point_list
```



# KDTree



- Question : Donner l'algorithme pour trouver le point le plus proche d'un point donné



# KDTree

- Donner l'algorithme pour trouver le point le plus proche d'un point donné

```
function closestpoint (kdtree, point p) {
```

De la même manière qu'a la création,  
on détermine la subdivision dans laquelle le point appartient,  
on trouve le point le plus proche dans la subdivision (appelé meilleur candidat)

On remonte dans la hiérarchie en testant à chaque niveau

Si un meilleur candidat n'est pas dans la subdivision opposée ou au pivot :

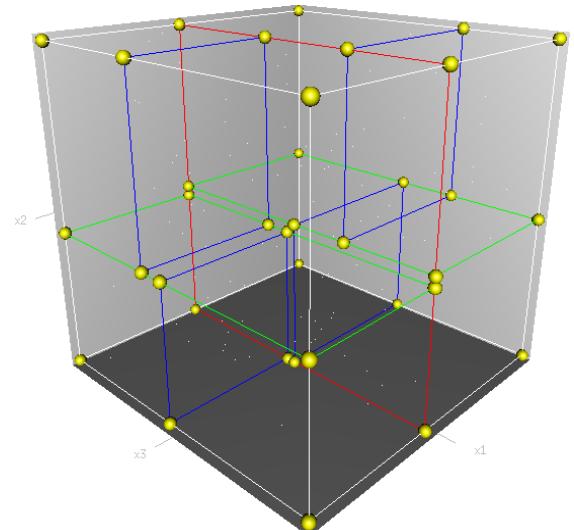
On teste cela en comparant la distance de p au meilleur candidat courant et  
la distance de p au plan de subdivision (donné par la coordonnée n du pivot)

Si la distance est trop grande:

On teste si le pivot ne fournit pas un meilleur candidat.

On teste les subdivisions opposées avec la même procédure

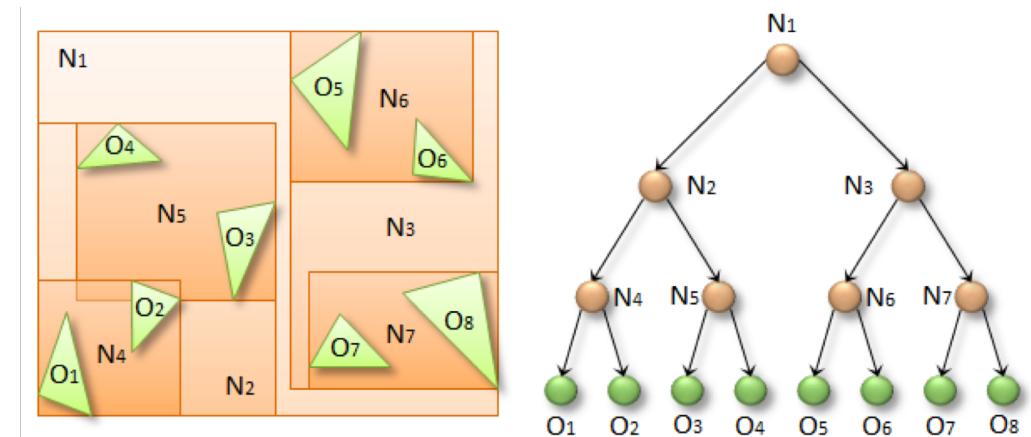
Si la distance au candidat est plus petite, on remonte



# Bounding Volume Hierarchy

Une hiérarchie de volumes englobants.

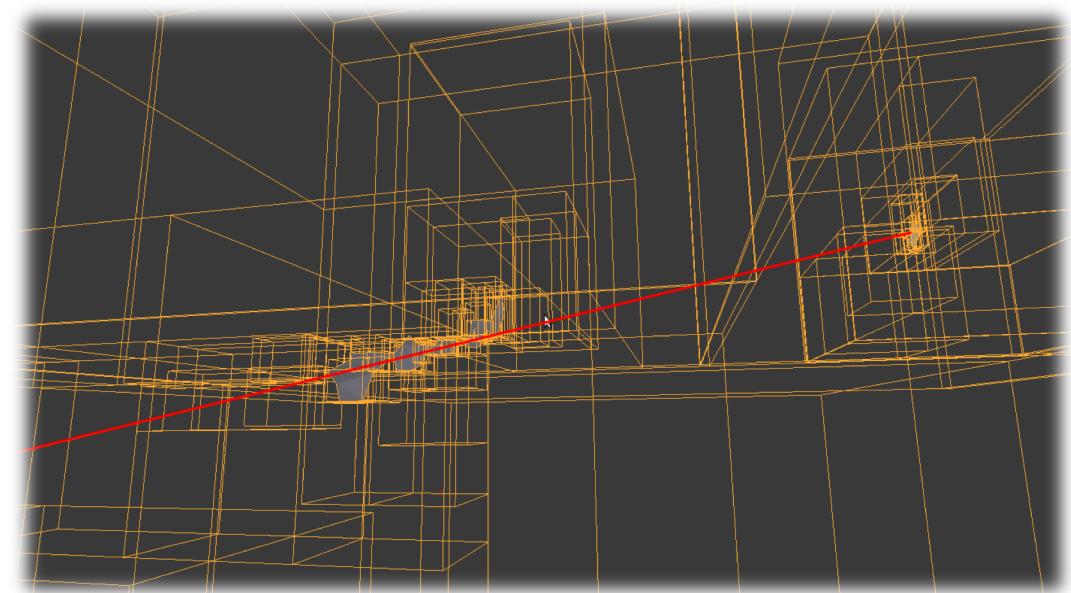
Chaque nœud de l'arbre est associé à un sous ensemble des primitives et à leur volume englobant.



# Bounding Volume Hierarchy

La BVH fournit une représentation simplifiée pour tester rapidement par exemple des test de détection de collision ou de interception de rayons.

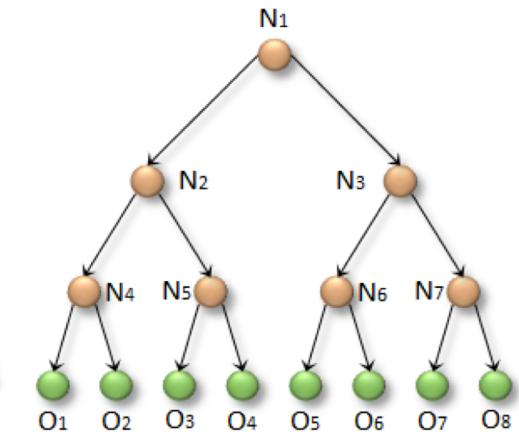
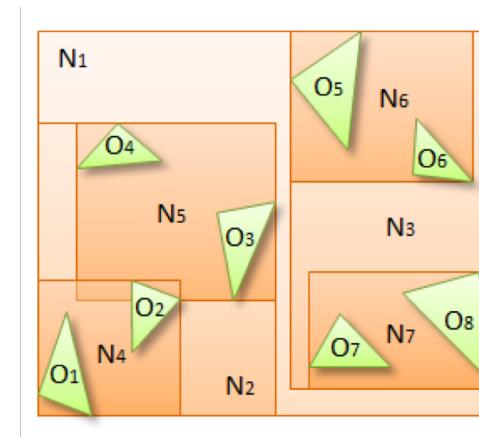
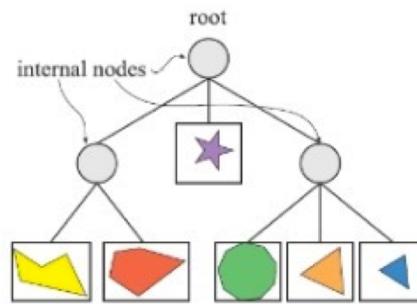
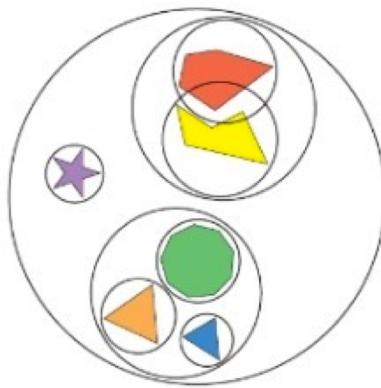
Cela permet d'éviter de tester toutes les primitives entre elles.



# Bounding Volume Hierarchy

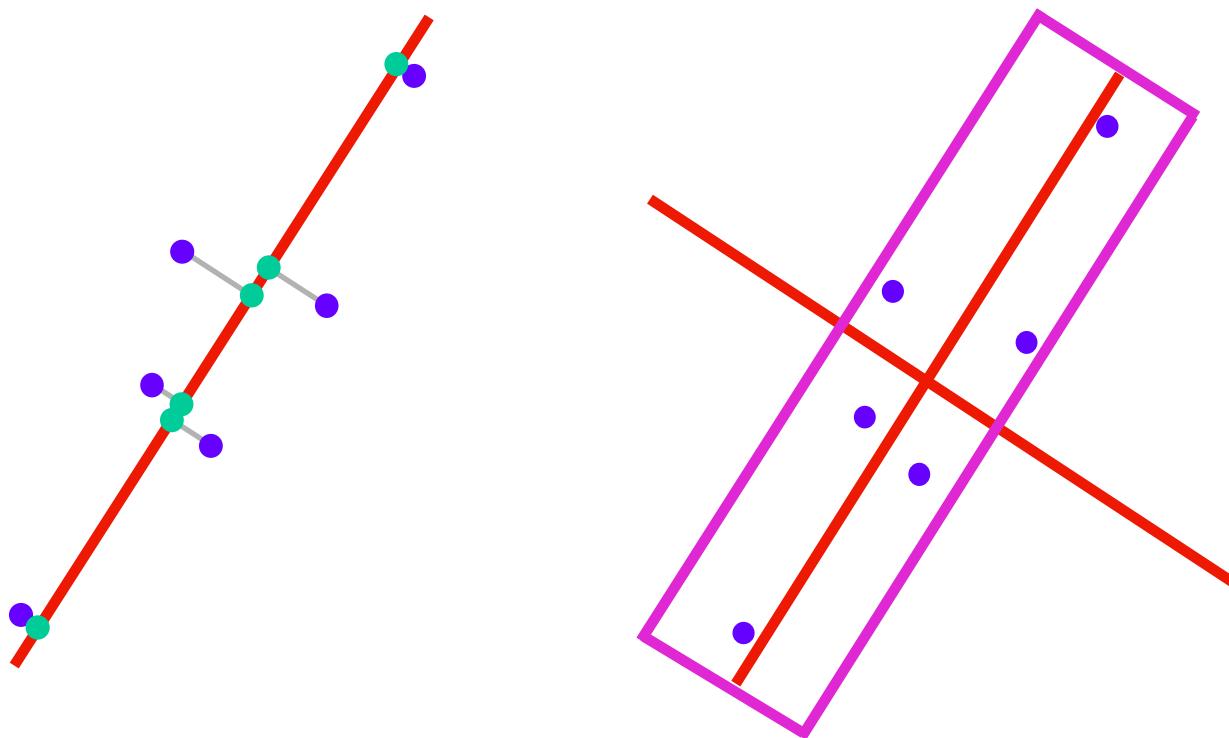
Différent types de BVH:

- Sphère
- Polytopes orientés discrets
- Boites englobantes alignées sur les axes
- Boites englobantes orientées objets



# Bounding Volume Hierarchy

Les Boites englobantes orientées objets utilisent les axes principaux de l'ensemble des sommets



# Separating Axis Theorem

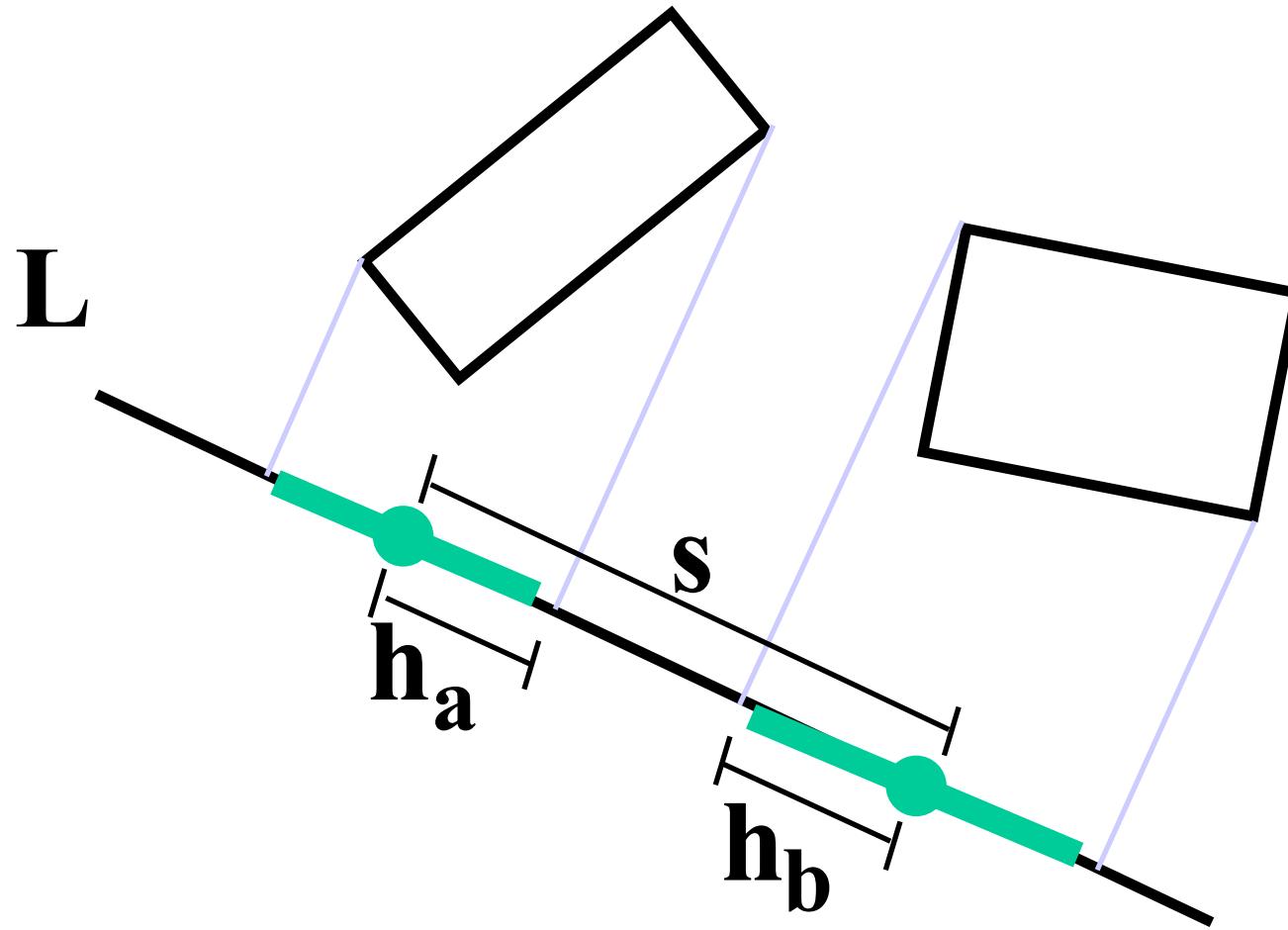
Two polytopes A and B are disjoint iff there exists a separating axis which is:

perpendicular to a face from either

or

perpendicular to an edge from each

# Test de recouvrement des OBBs



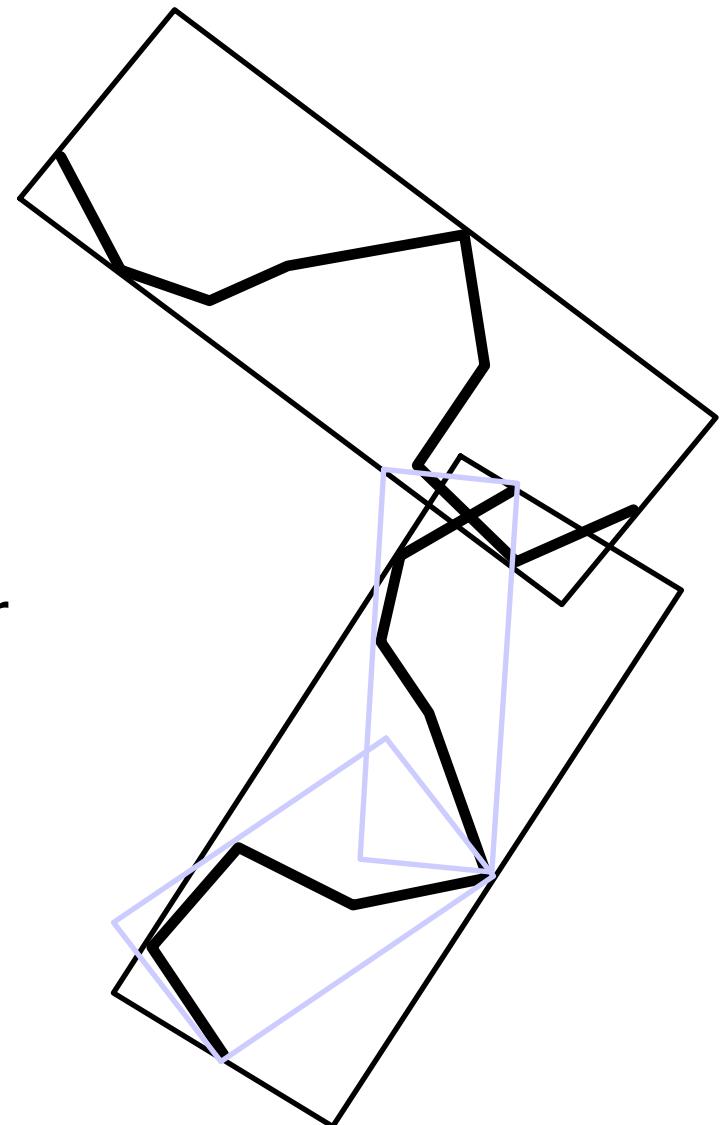
**L est un axe séparant ssi:  $s > h_a + h_b$**

4 axes à tester en dimension 2 et 15 en dimension 3

# Bounding Volume Hierarchy

## Création de la hierarchy

- Top-down
  - Minimise le volume des enfants (Zachmann, VRST02)
  - Sépare suivant la dimension la plus large (Mezger et al., WSCG03)
  - On arrête lorsqu'on a n éléments par feuilles
- Bottom-up
  - Optimise pour avoir des volumes de forme sphérique (facteur de forme) (Volino et al. CFG94)



# Modélisation Géométrique

Géométrie Constructive de Solides  
Les arbres CSG

# Modélisation à partir de primitives

Primitives géométriques solides simples:



Opérateurs de transformation géométriques:

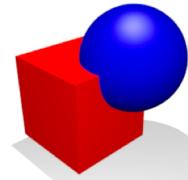
- Translations
- Rotations
- Changements d'échelle et homothéties

# Les opérateurs de composition

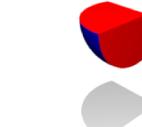
Les opérateurs de composition ensemblistes:



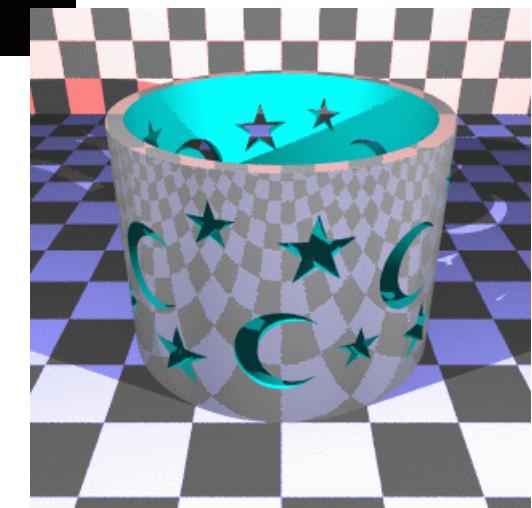
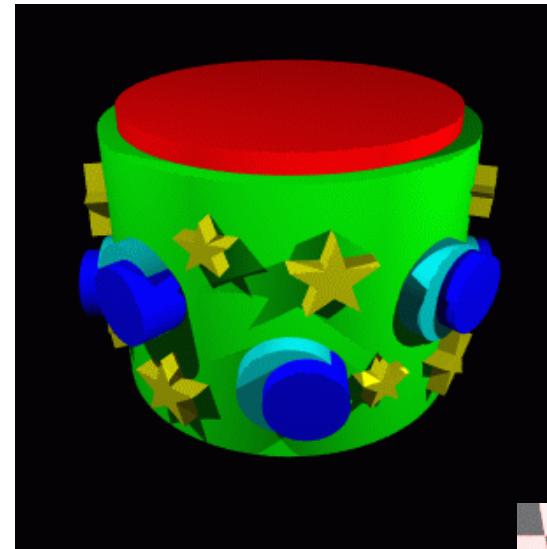
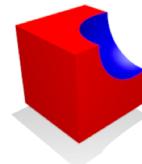
Union



Intersection



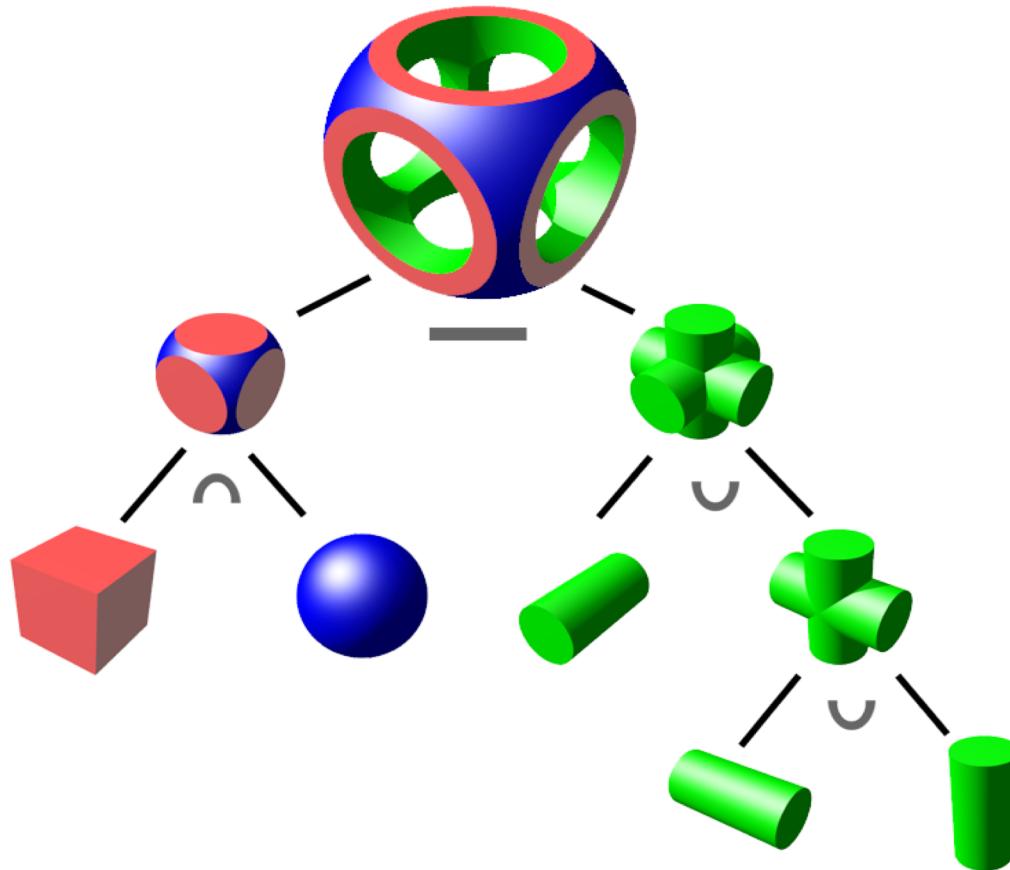
Différence



# Les arbres CSG

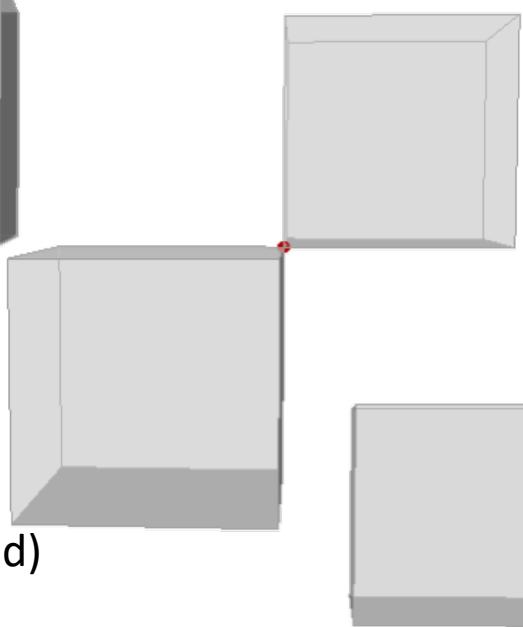
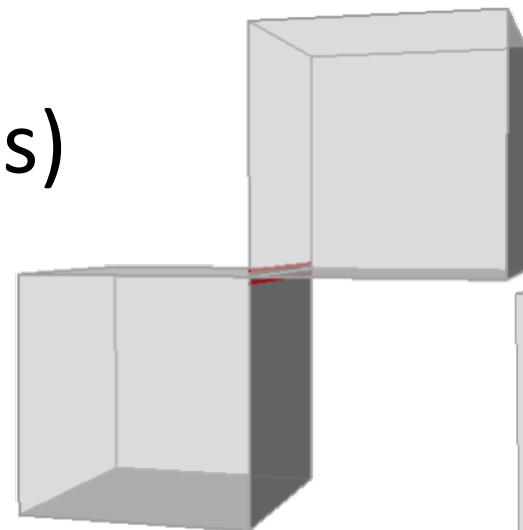
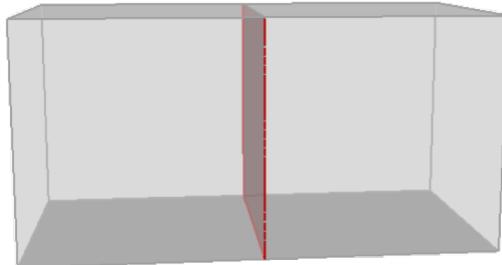
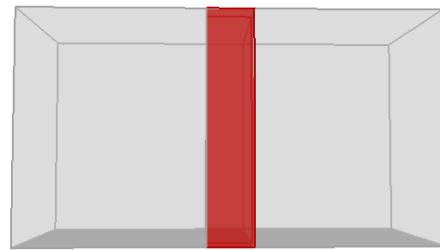
Constructive Solid Geometry : arbre de composition

- Les noeuds sont des opérateurs de composition
- Les feuilles sont des primitives géométriques (transformées)

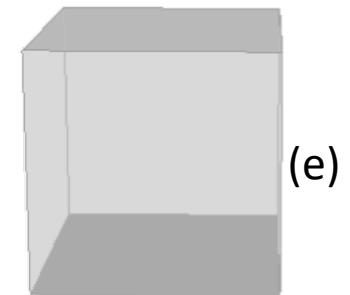


# Problème de la régularisation

- L'intersection de 2 solides ne donne pas toujours un solide  
(faces/arêtes pendantes)



L'intersection de 2 cubes peut produire (a) un solide, (b) un plan, (c) une ligne, (d) un point et (e) l'ensemble vide.



# Opérateurs régularisés

## ■ Régularité d'un ensemble

- Soit  $X$  un ensemble
- L'intérieur de  $X$ , noté  $\text{int}(X)$ , est l'ensemble des points  $p$  tels qu'il existe un voisinage (cad une boule ouverte de centre  $p$ ) inclus dans  $X$
- La fermeture de  $X$ , noté  $\delta(X)$ , est l'ensemble des points  $p$  tels que tout voisinage de  $p$  contient au moins un point de  $X$
- $X$  est régulier si  $X = \text{int}(\delta(X))$ . L'opération  $\text{int}(\delta(X))$  est appelé régularisation de  $X$ .

## ■ Opération de régularisation

- $A \cap^* B = \text{int}(\delta(A \cap B))$  (vide si  $A \cap B$  n'est pas un volume)

# Les CSG

## ■ Les -

- Non unicité : Il existe une infinité d'arbres pour une même forme.
- Validité : Les opérateurs rendent complexes les opérations sur l'arbre (intersection d'un rayon).
- Domaine insuffisant : certaines formes sont impossible (hélice)
- Nature implicite : Certaines opérations sont difficiles (calcul du volume, etc)

## ■ Les +

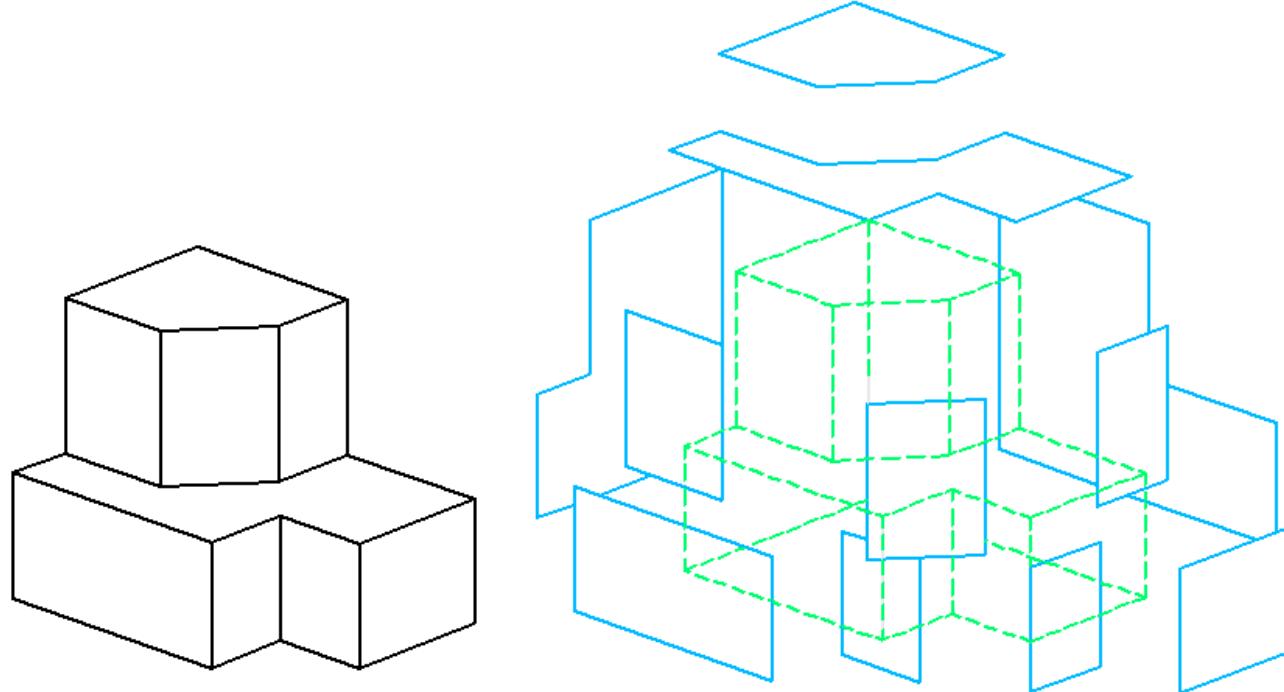
- Fournit un historique de construction
- Vers une approche fonctionnelle (opération de raccordement, offsets, balayage, etc)

# Modélisation Géométrique

Représentation par les bords  
B-Rep

# Boundary-Representation

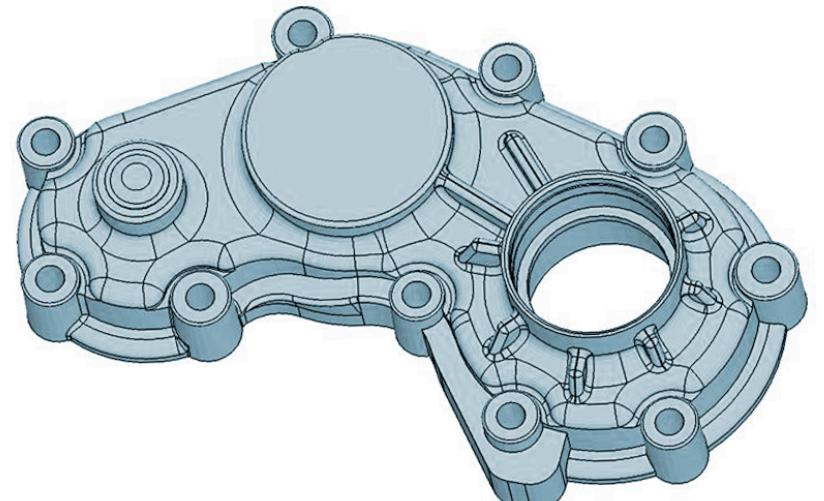
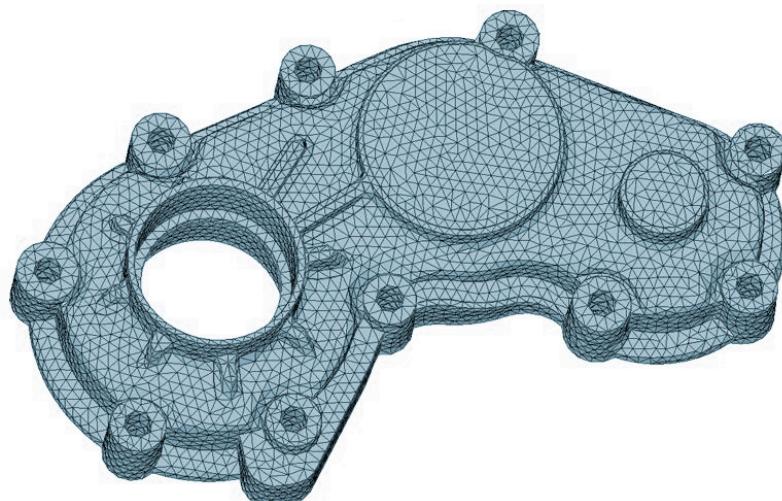
- Un modèle est représenté par ses bords (surfaces "cousues" par leurs arêtes)
- Pas de notion explicite de volume, représentation surfacique
- On peut représenter des solides ou des formes plus complexes (avec sous volumes internes, ou non bornées)



# Boundary-Representation

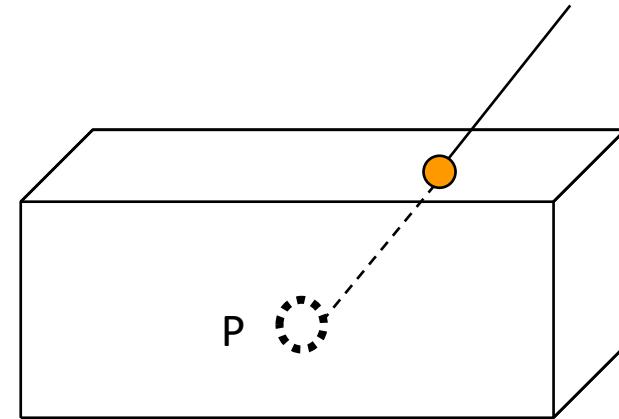
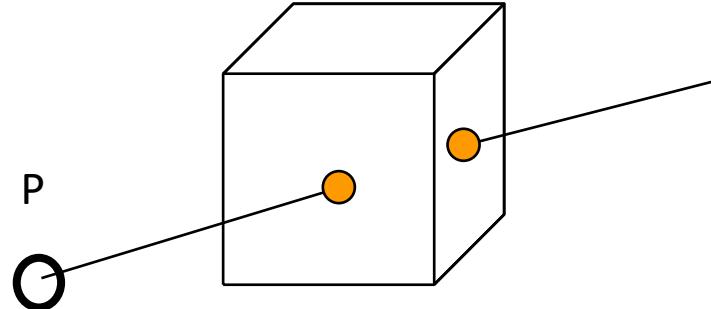
Dissociation de la topologie et de la géométrie

- Les caractéristiques géométriques des éléments (position des points, forme des surfaces) sont indépendantes de leurs relations "d'incidence" (de bord)
- On peut utiliser différents "plongements" géométrique, des polygones ou des courbes et surfaces libres (type NURBS)



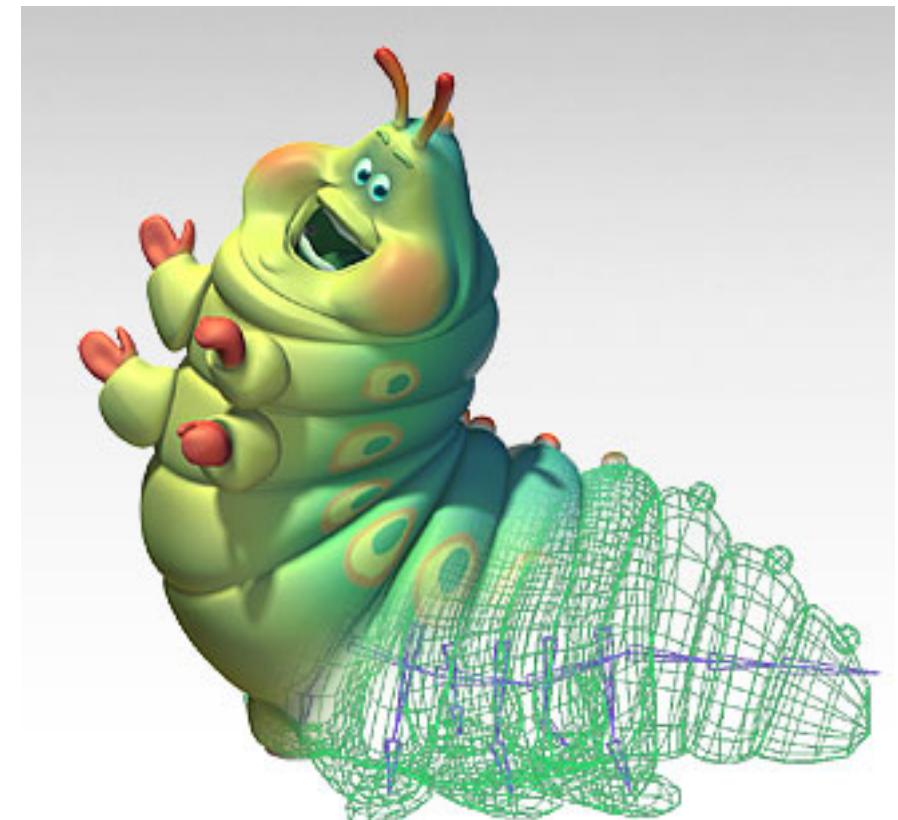
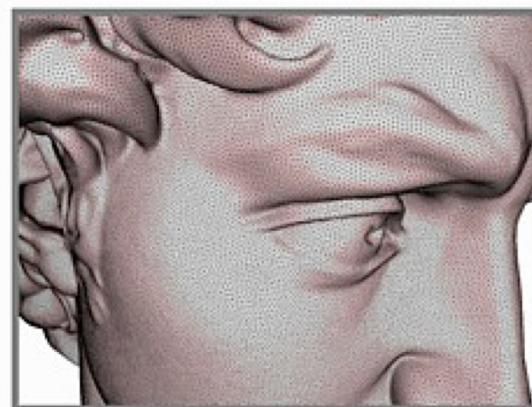
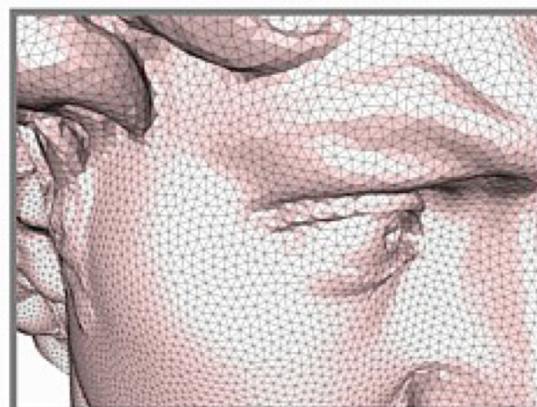
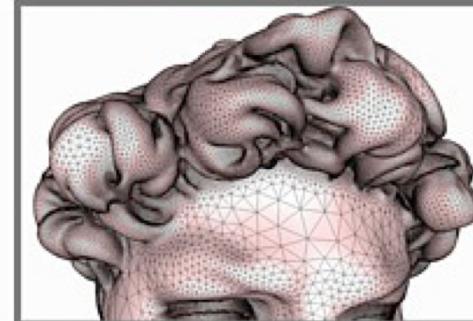
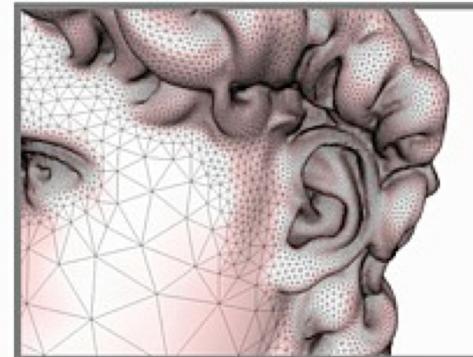
# B-Rep solide

Test intérieur / extérieur sur un solide :



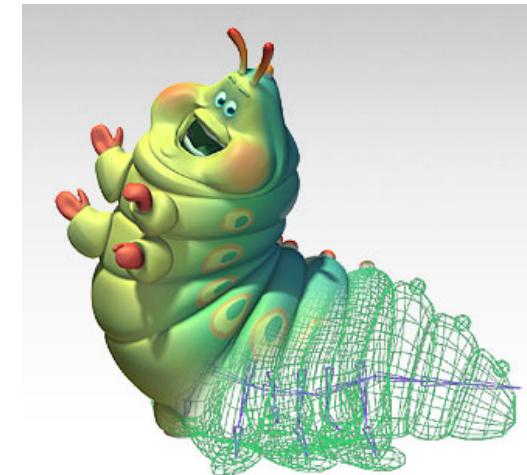
- Nombre pair d'intersections : le point P est à l'extérieur du solide
- Nombre impair d'intersections : le point P est à l'intérieur du solide

# Les maillages



# Les maillages

- B-Rep formant un complexe linéaire par morceaux : surfaces représentées par des polygones.
- Triangle souvent utilisé (simplexe pour une face) mais quad(rilatères) fréquents.
- La continuité globale est  $C^0$  (discontinuité de normales au niveau des arêtes).
- Information sur la géométrie et la topologie de la surface
- Structure standard pour afficher des scènes complexes en 3D
- Leur visualisation et leur manipulation est optimisée par la grande majorité des cartes graphiques actuelles.



# Les maillages

- Un maillage polygonal consiste en 3 types d'éléments :



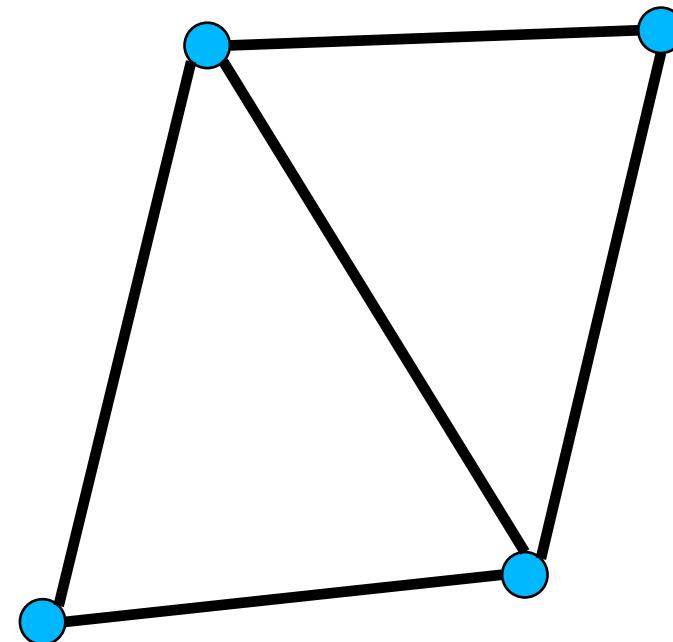
- Les sommets ( $x, y, z$ )



# Les maillages

- Un maillage polygonal consiste en 3 types d'éléments :

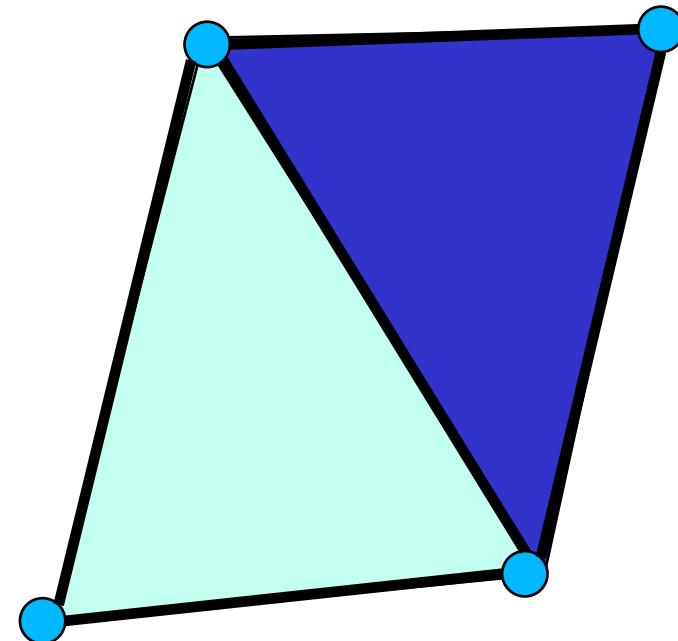
- Les sommets ( $x,y,z$ )
- Les arêtes
  - Définies par 2 sommets



# Les maillages

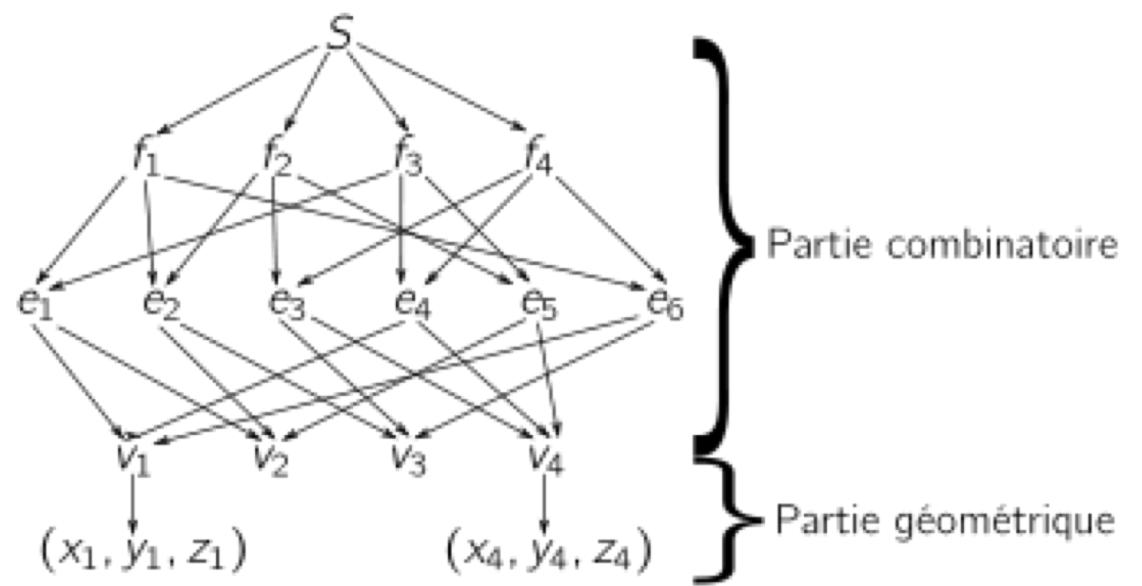
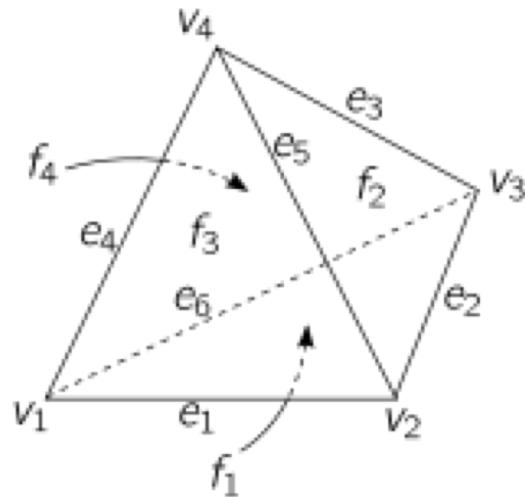
- Un maillage polygonal consiste en 3 types d'éléments :

- Les sommets ( $x,y,z$ )
- Les arêtes
  - Définies par 2 sommets
- Les faces
  - Définis par n sommets
  - Ou définis par n arêtes



# Les maillages

- Un maillage est décrit par sa *topologie* (connectivité) et sa *géométrie*
- La topologie d'un maillage décrit ses *relations d'incidences et d'adjacence* entre ses éléments (sommets adjacent, arête incident d'une face)
- La géométrie spécifie la *position* ou d'autres *caractéristiques géométriques* pour chaque élément.

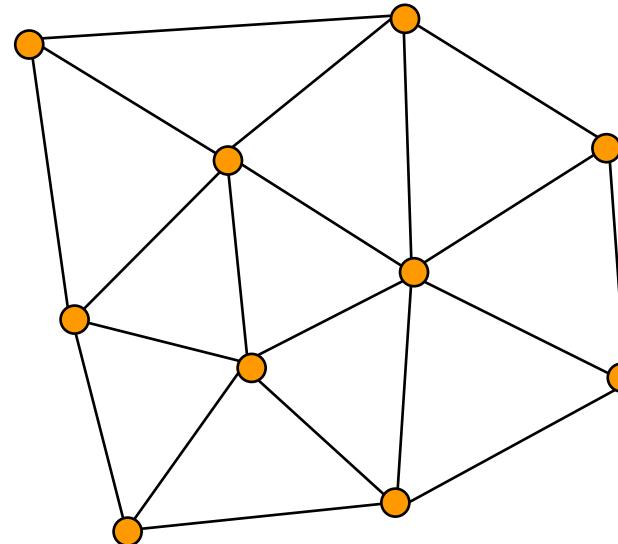


# Propriétés topologiques des maillages

- Dualité
- Fermeture
- 2-manifold
- Orientabilité
- Formule d'Euler
- Normales

# Dualité

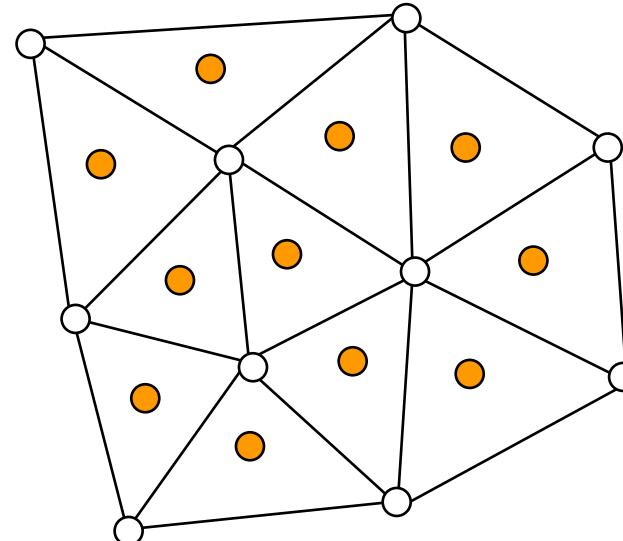
Chaque entité du maillage primal de dimension  $k$  est remplacé par une entité de dimension  $(2-k)$  dans le maillage dual:



# Dualité

Chaque entité du maillage primal de dimension  $k$  est remplacé par une entité de dimension  $(2-k)$  dans le maillage dual:

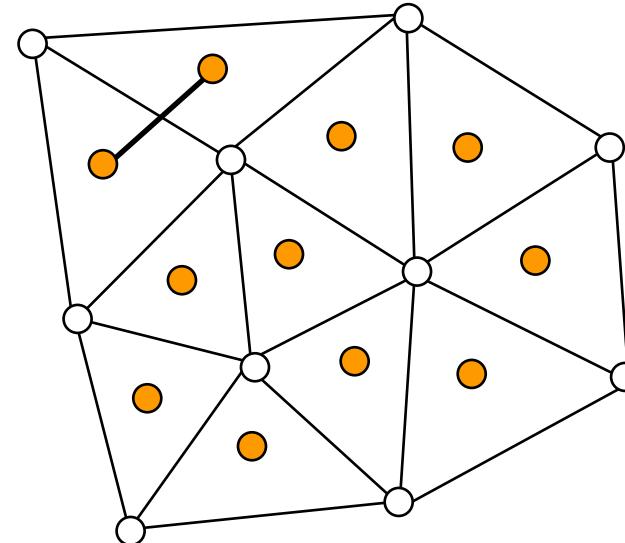
- Chaque face remplacée par un sommet avec pour coordonnées le barycentre de la face



# Dualité

Chaque entité du maillage primal de dimension  $k$  est remplacé par une entité de dimension  $(2-k)$  dans le maillage dual:

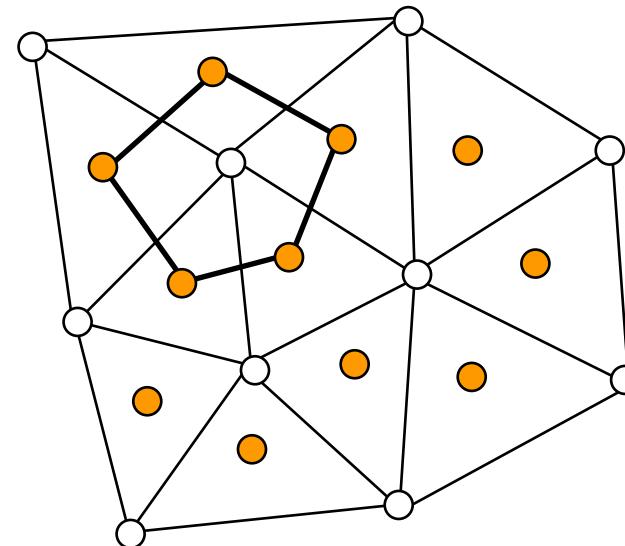
- Chaque face remplacée par un sommet avec pour coordonnées le barycentre de la face
- L'arête du dual relie deux sommets si les faces d'origines sont voisines



# Dualité

Chaque entité du maillage primal de dimension  $k$  est remplacé par une entité de dimension  $(2-k)$  dans le maillage dual:

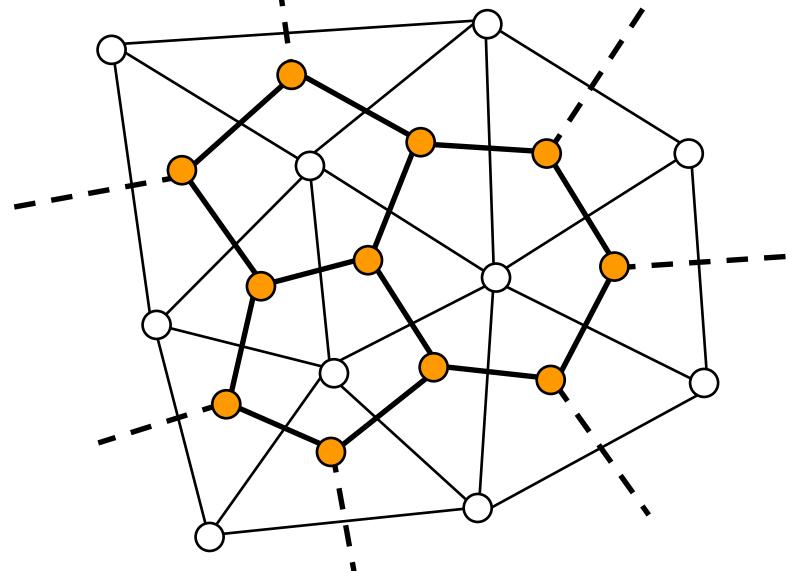
- Chaque face remplacée par un sommet avec pour coordonnées le barycentre de la face
- L'arête du dual relie deux sommets si les faces d'origines sont voisines
- Les points sont remplacés par les faces résultantes



# Dualité

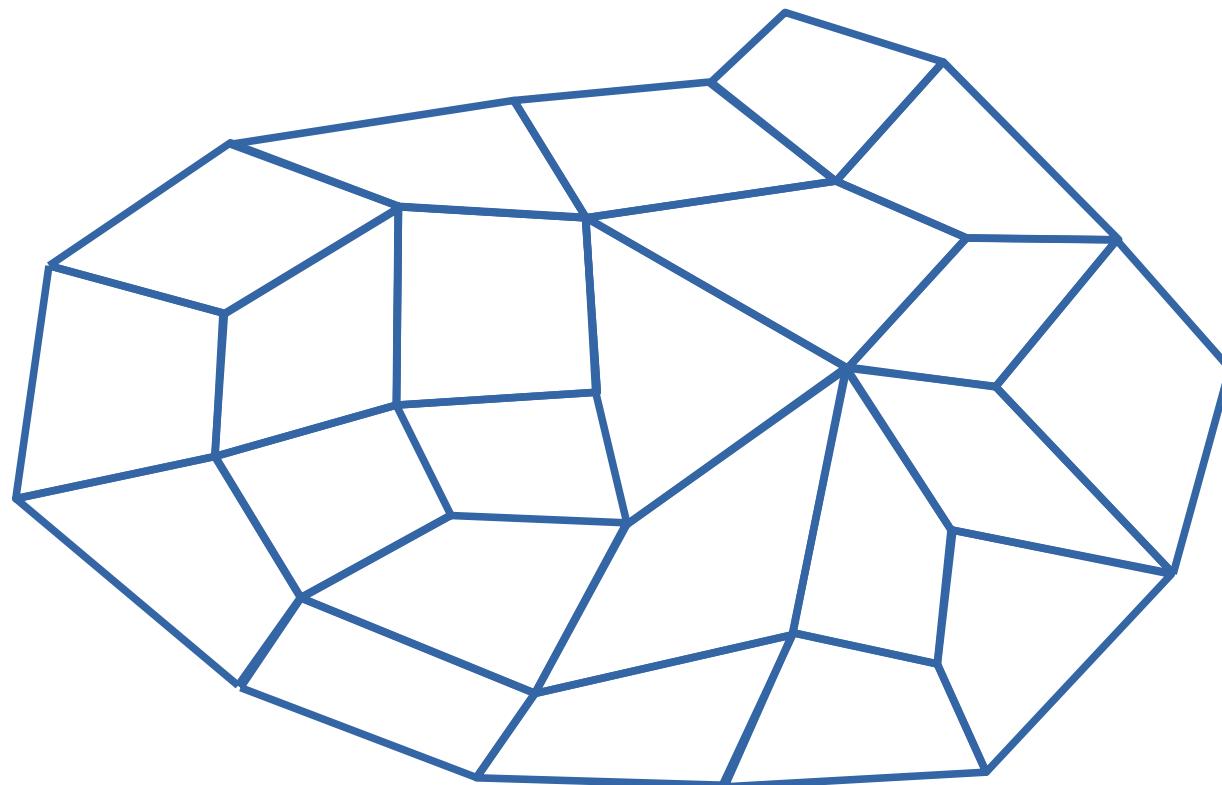
Chaque entité du maillage primal de dimension  $k$  est remplacé par une entité de dimension  $(2-k)$  dans le maillage dual:

- Chaque face remplacée par un sommet avec pour coordonnées le barycentre de la face
- L'arête du dual relie deux sommets si les faces d'origines sont voisines
- Les points sont remplacés par les faces résultantes



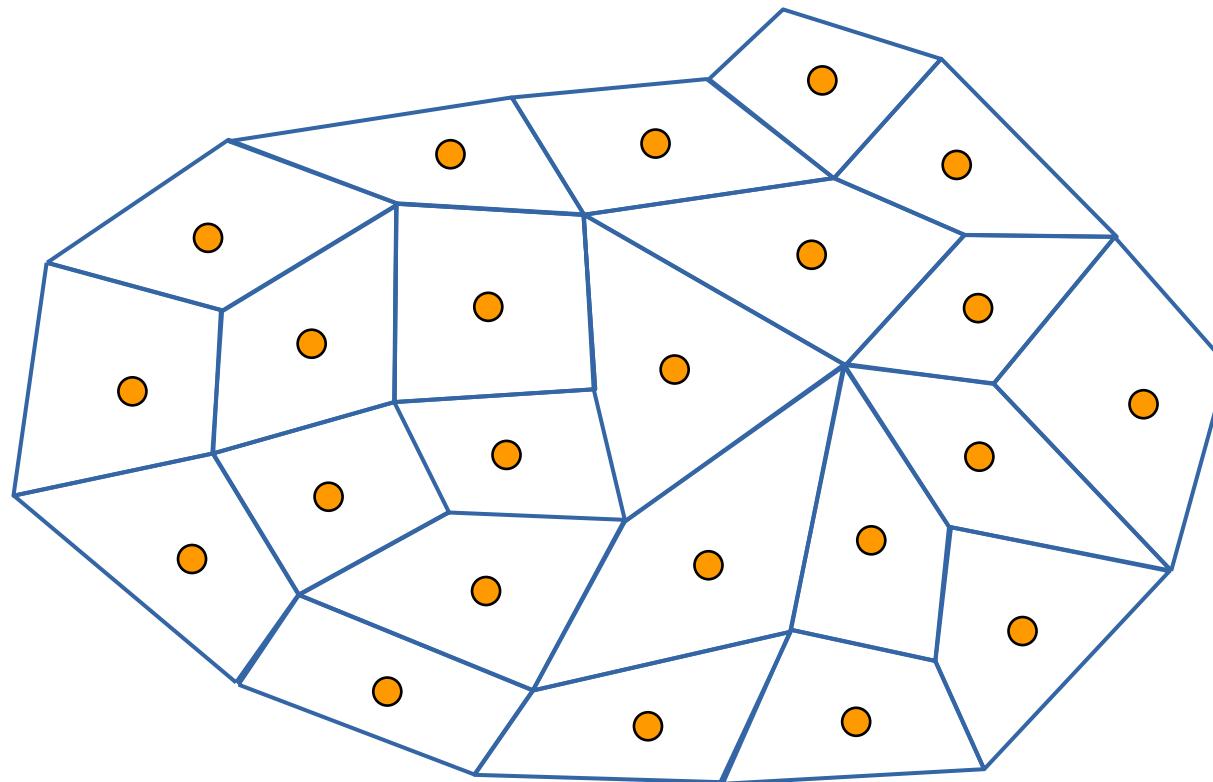
# Exercice sur le dual

Tracez le dual du maillage suivant :



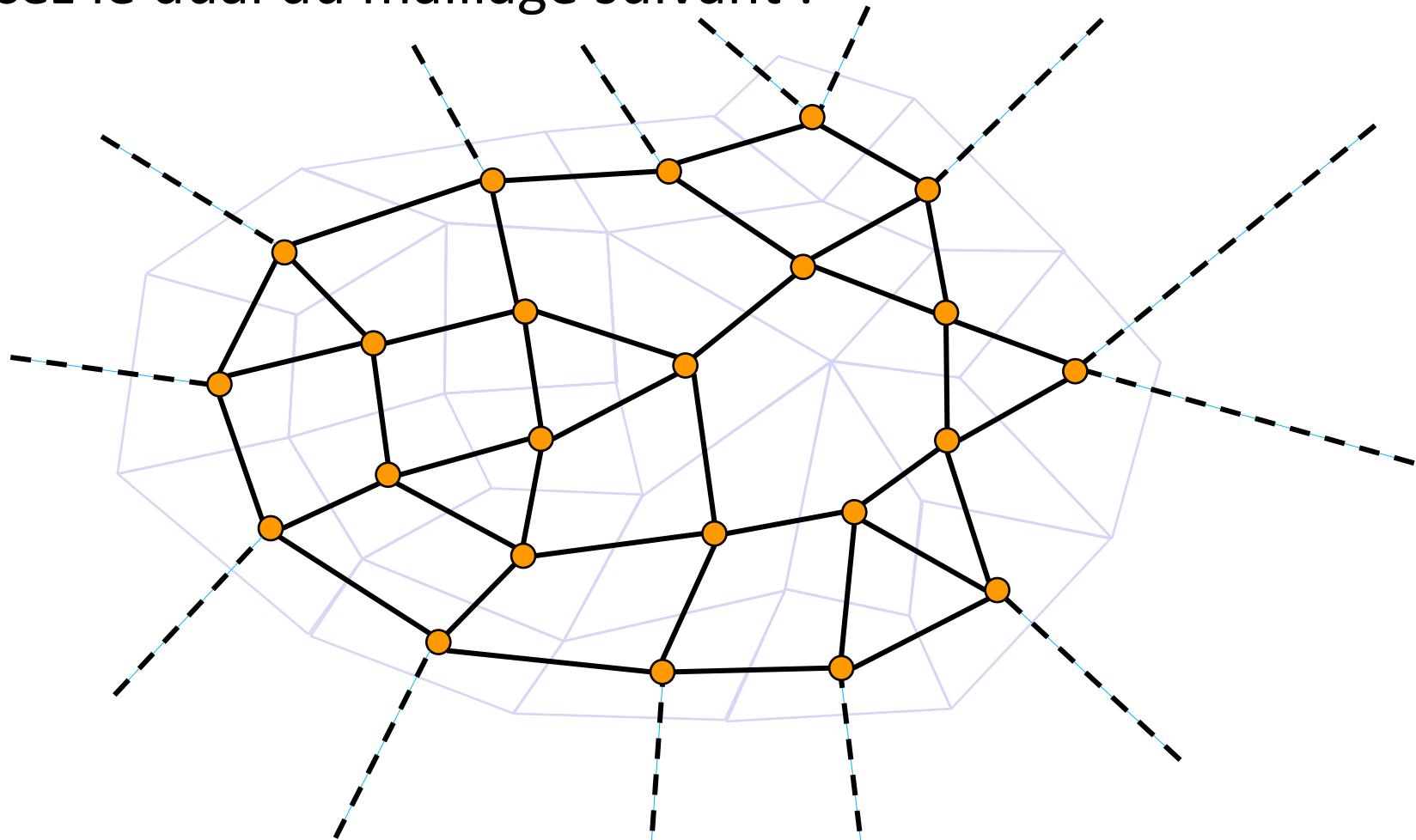
# Exercice sur le dual

Tracez le dual du maillage suivant :



# Exercice sur le dual

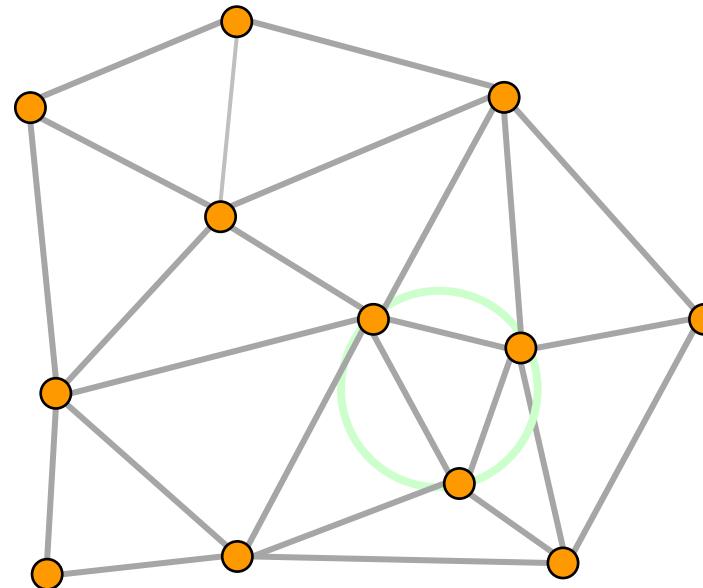
Tracez le dual du maillage suivant :



- Que remarquez vous au niveau des faces ?
- Que dire sur le dual du dual ?

# Application : Diagramme de Voronoi

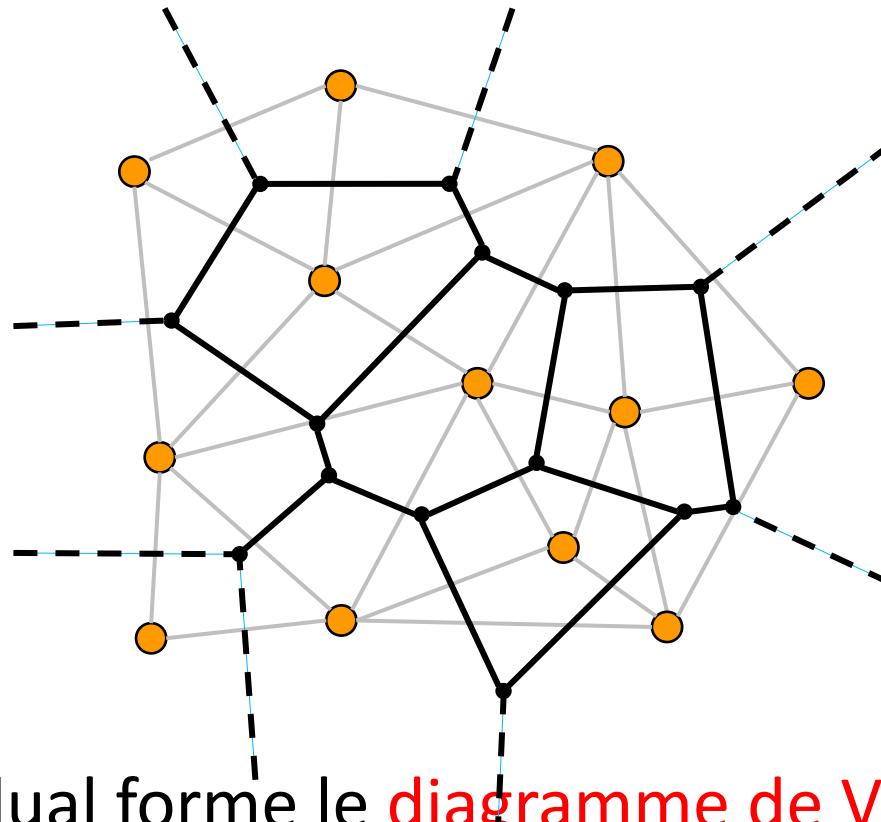
A partir d'un ensemble de points on peut construire (facilement) leur **triangulation de Delaunay**



- Maillage triangulaire (unique) tel que aucun sommet ne soit à l'intérieur du cercle circonscrit à une face

# Application : Diagramme de Voronoi

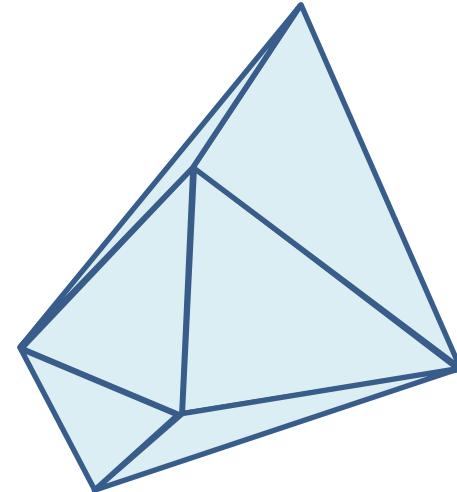
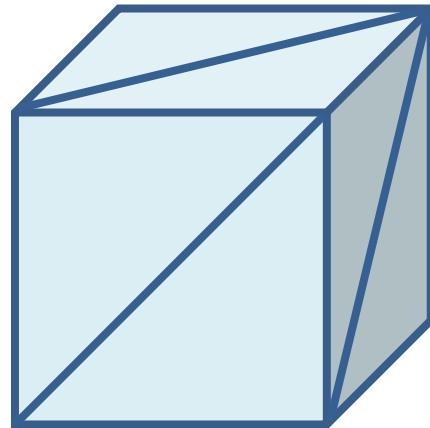
A partir d'un ensemble de points on peut construire (facilement) leur **triangulation de Delaunay**



- Le maillage dual forme le **diagramme de Voronoi** des points : partition de l'espace où chaque face (duale) représente l'ensemble des plus proches voisins du point (primal)

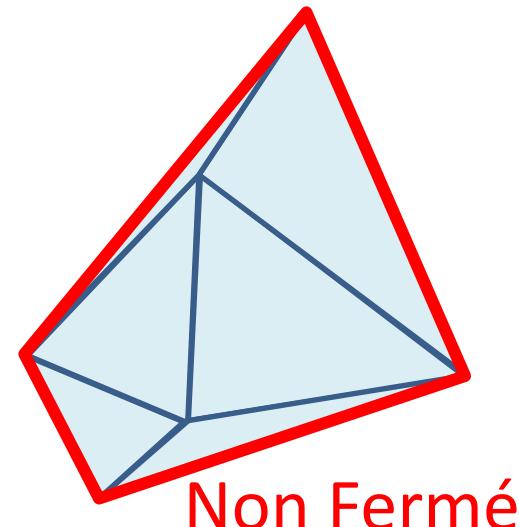
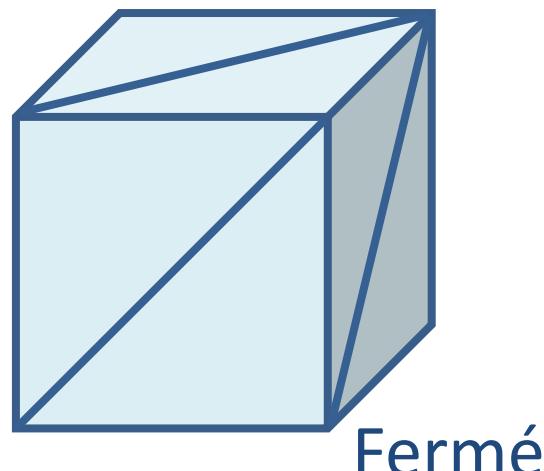
# Fermeture

- Un maillage est dit fermé s'il n'a pas de bord.
  - Toutes les arêtes du maillage sont partagées par 2 triangles



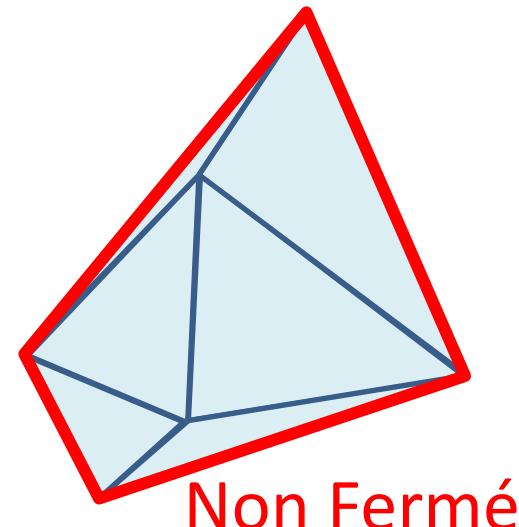
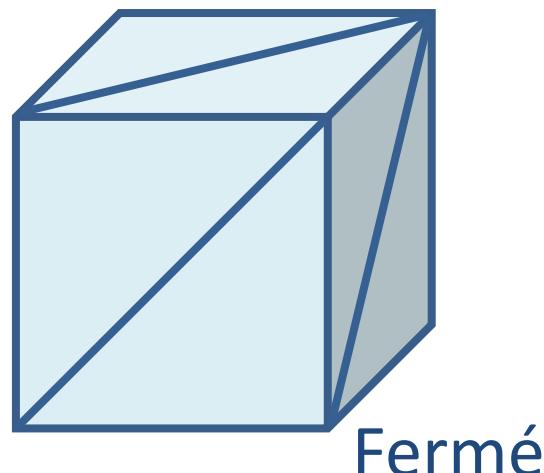
# Fermeture

- Un maillage est dit fermé s'il n'a pas de bord.
  - Toutes les arêtes du maillage sont partagées par 2 triangles



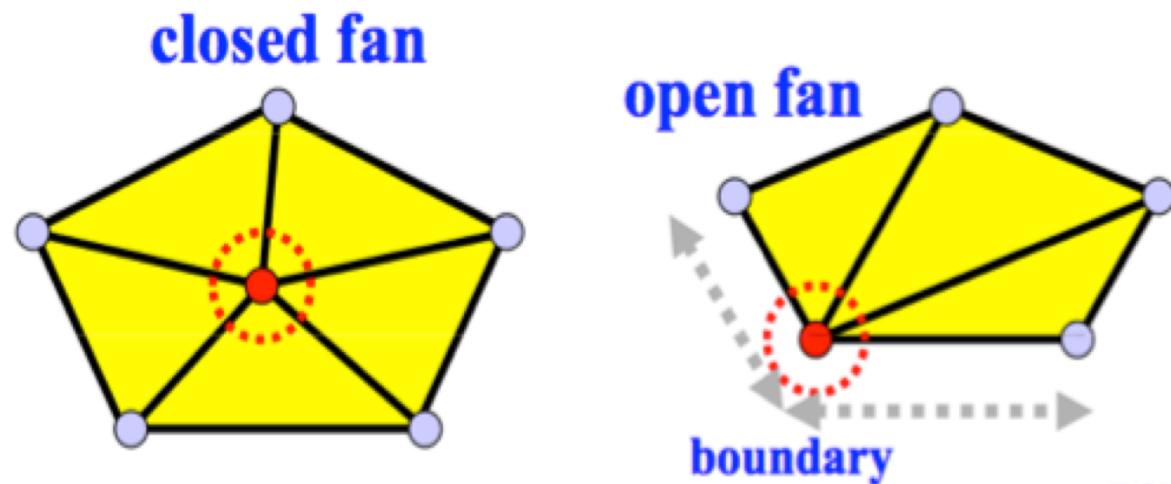
# Fermeture

- Un maillage est dit fermé s'il n'a pas de bord.
  - Toutes les arêtes du maillage sont partagées par 2 triangles
- Pour un maillage ouvert, les arêtes incidentes à seulement une face forment la frontière.



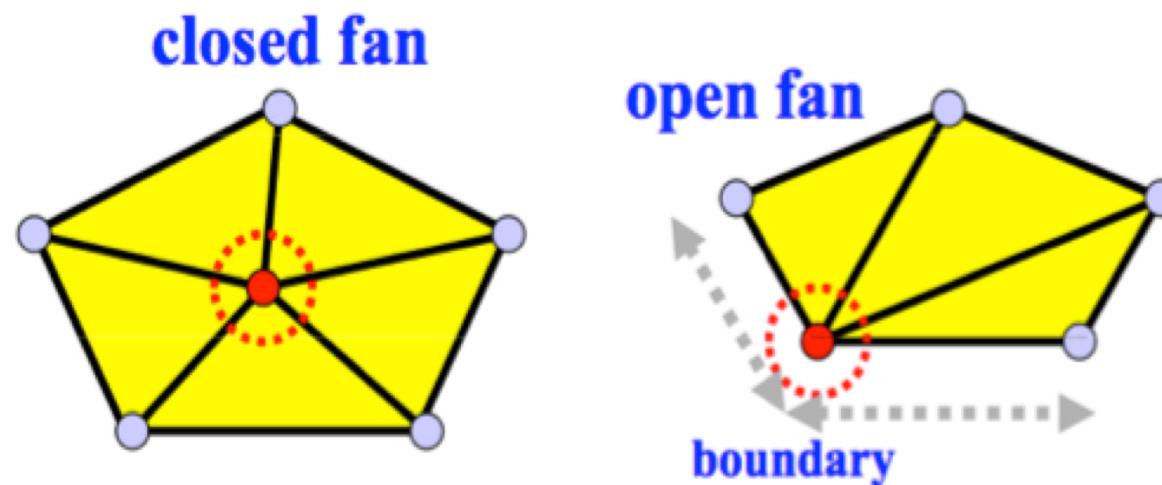
# Fermeture

- Pour un maillage fermé, les polygones et les arêtes autour d'un sommet forment une orbite.



# Manifold - Définitions

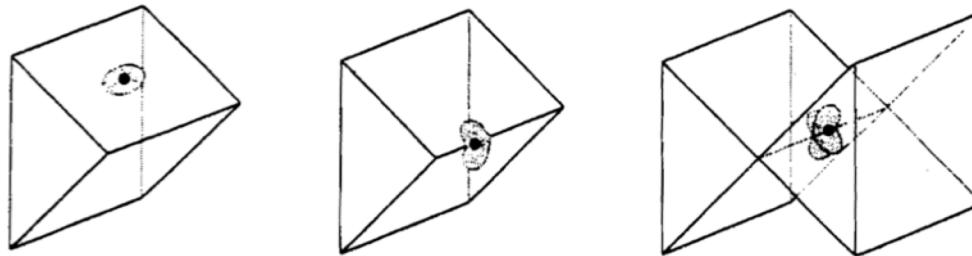
- Un maillage est **manifold** si
  - Chaque arête est incidente à seulement 1 ou 2 faces
  - Chaque sommet n'est incident qu'à 0 ou 2 arêtes du bord.



- De façon informelle, si un maillage fermé et 2-manifold (ou manifold) alors l'objet qu'il représente est « usable ».

# Définition alternative

- Un maillage est 2-manifold si:
  - En chaque point du maillage, il existe une sphère de rayon  $r > 0$  telle que l'intersection entre la sphère et le maillage est *homothétique à un disque*

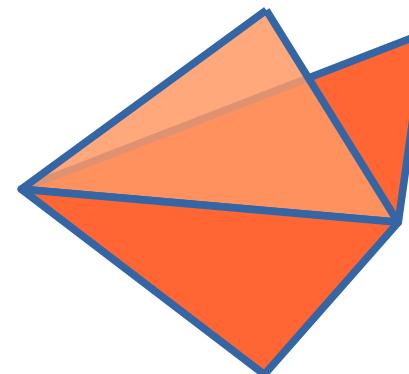


- Pour un maillage 2-manifold ouvert, à la bordure, l'intersection entre la sphère et le maillage doit être topologiquement équivalente à un **demi-disque**.

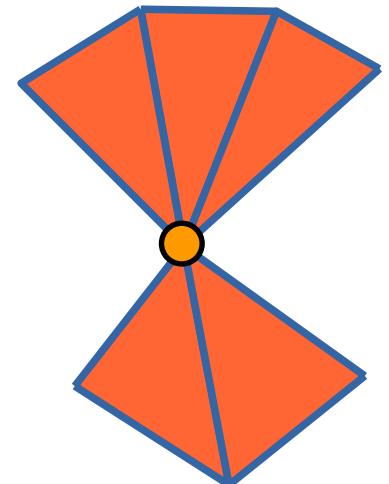
# Exercice : manifold / non-manifold ?

Sur des maillages triangulaires:

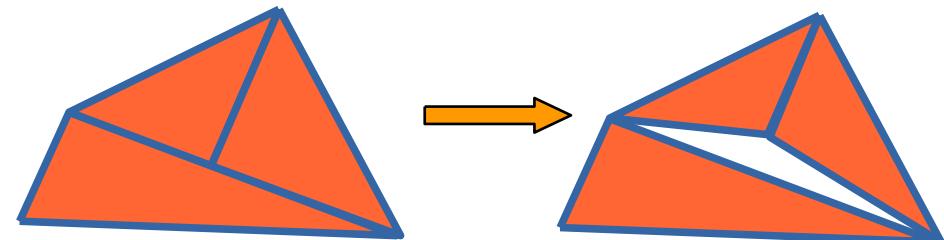
Une arête partagée par plus de 2 triangles ?



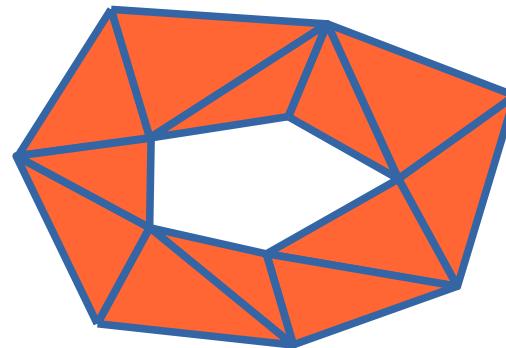
Un sommet partagé par deux ensembles de facettes non connectés ?



Une jonction en T ? Problème de Crack ?

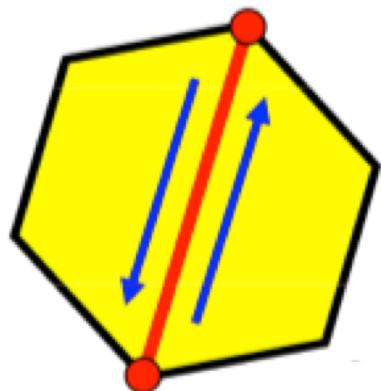


Un trou dans le maillage ?



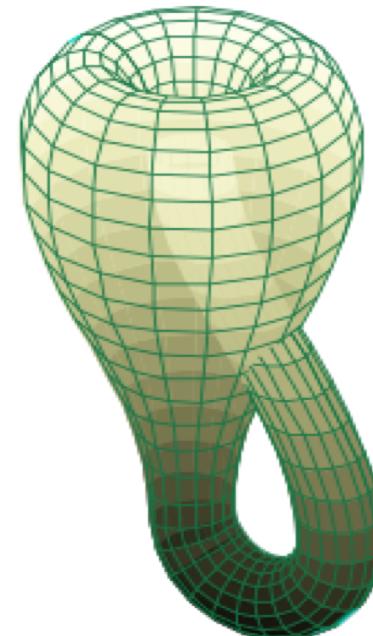
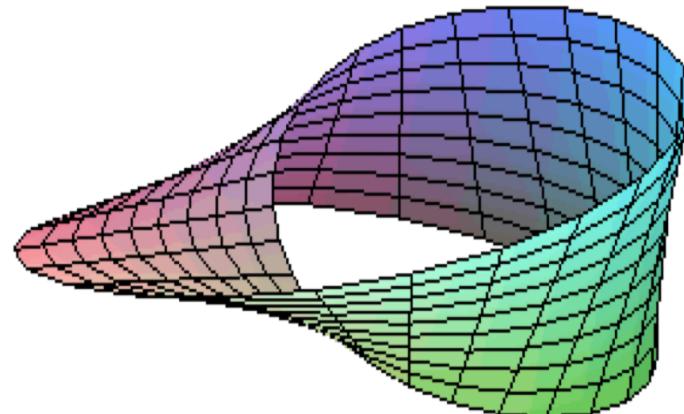
# Orientation des faces

- L'orientation d'une face est donné par l' ordre cyclique de ces sommets incidents
- L'orientation de deux faces adjacentes est compatible si les 2 sommets de leurs arêtes communes sont dans un ordre opposé
- Un maillage manifold est orientable si chaque paires de faces adjacentes ont des orientation compatible



# Manifolds non orientables

- Tous les maillages manifold ne sont pas orientables. Les plus connus sont le ruban de Möbius et la bouteille de Klein.
- Le ruban de Möbius n'a qu'un coté. La bouteille de Klein n'a pas d'intérieur et d'extérieur.



# Formule d'Euler-Poincaré (1752)

Correspondance entre le nombre de composant de chaque entité du maillage  
(sommets, arêtes, faces)

$$S - A + F = X = 2(1-g)$$

- $S$  : nombre de sommets,
- $A$  : nombre d'arêtes
- $F$  : nombre de faces.
- $g$  est le genre (génus) de l'objet = le nombre de trous (poignées) dans un objet fermé



genre 0



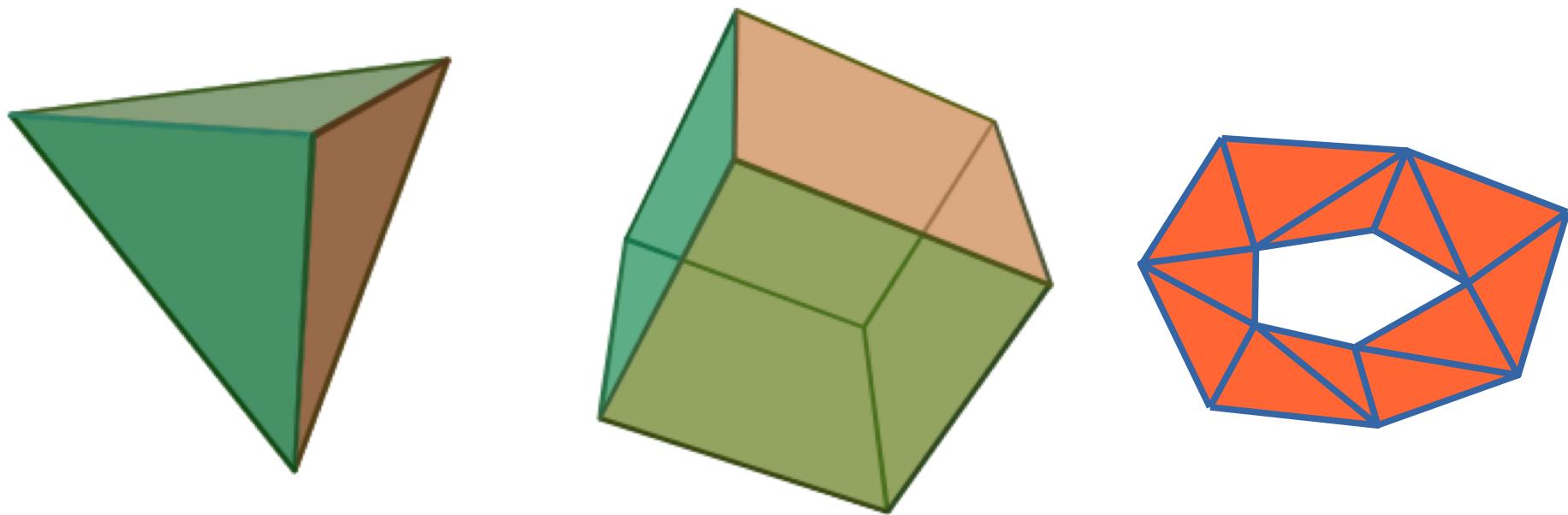
genre 1



genre 2

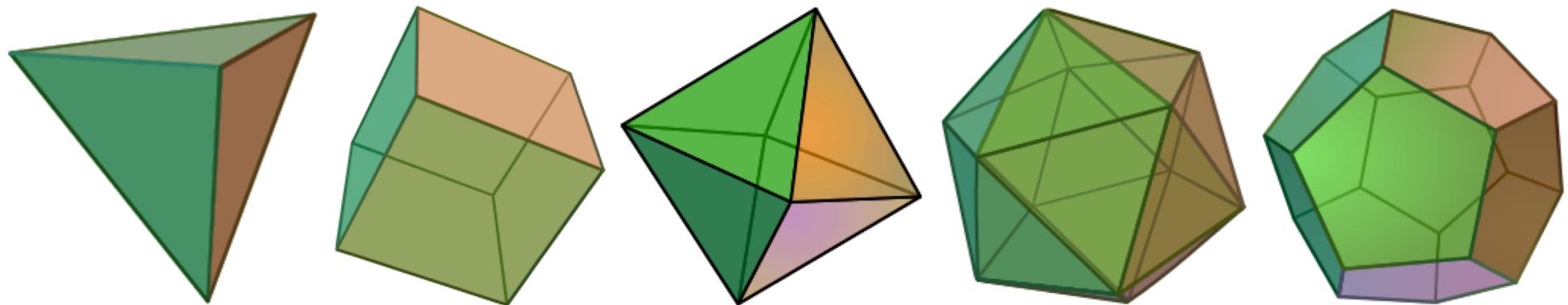
# Formule d'Euler-Poincaré (1752)

$$S - A + F = X = 2(1-g)$$



# Formule d'Euler-Poincaré (1752)

Solide platoniciens



Tétraèdre :  $4 - 6 + 4 = 2$

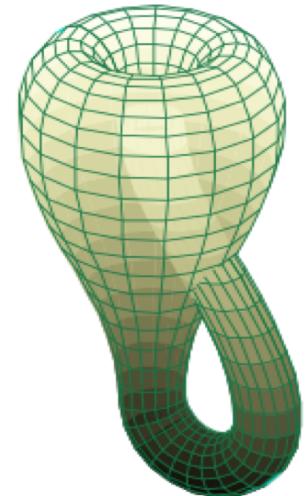
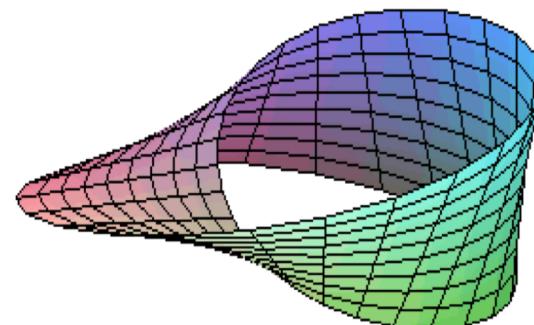
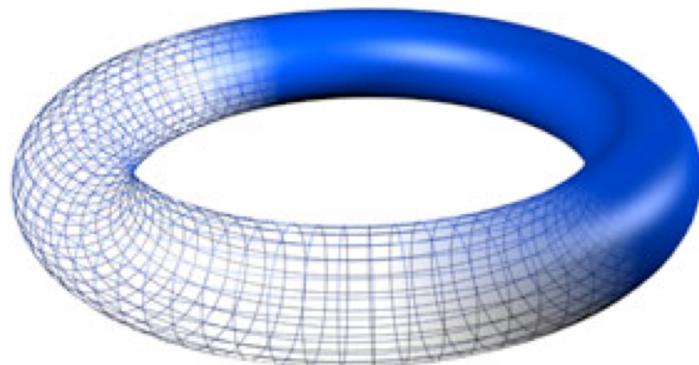
Hexaèdre (cube) :  $8 - 12 + 6 = 2$

Octaèdre :  $6 - 12 + 8 = 2$

Dodécaèdre :  $20 - 30 + 12 = 2$

Icosaèdre :  $12 - 30 + 20 = 2$

# Formule d'Euler-Poincaré (1752)



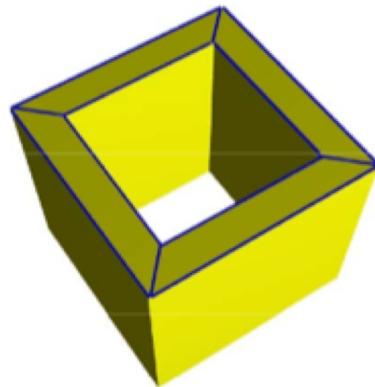
Tore: 0

Ruban de Möbius : 0

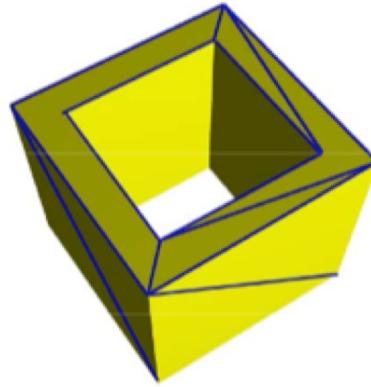
Bouteille de Klein : 0

# Formule d'Euler-Poincaré

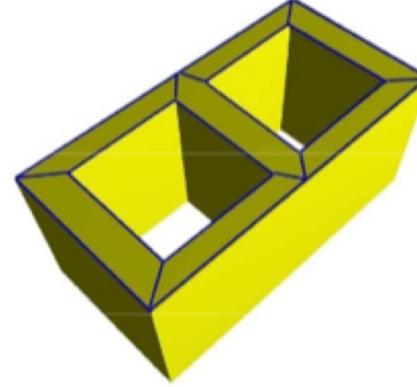
- Indépendant de la tessellation



$$\begin{aligned}V &= 16, E = 32, F = 16 \\ \chi(\mathbf{M}) &= V - E + F = 0\end{aligned}$$

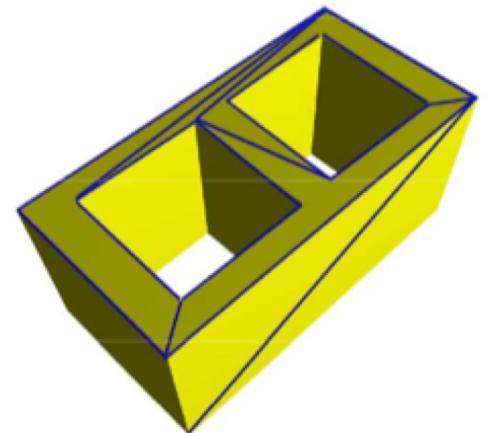


$$\begin{aligned}V &= 16, E = 36, F = 20 \\ \chi(\mathbf{M}) &= V - E + F = 0\end{aligned}$$



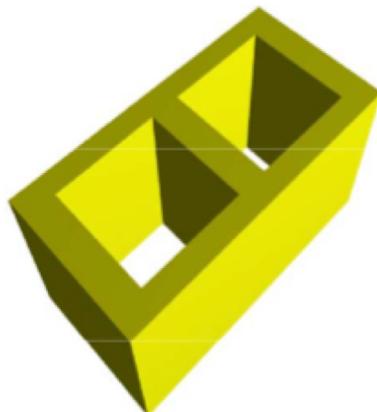
$$\begin{aligned}V &= 28, E = 56, F = 26 \\ \chi(\mathbf{M}) &= V - E + F = -2\end{aligned}$$

$$\begin{aligned}V &= 24, E = 48, F = 22 \\ \chi(\mathbf{M}) &= V - E + F = -2\end{aligned}$$

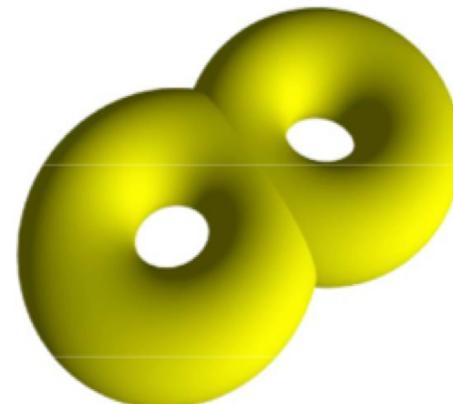


# Homéomorphisme

- Deux maillage manifold A et B sont homéomorphe si leur surfaces peuvent être transformé l'une en l'autre en les déformant.
- Deux maillages manifold fermé sont homéomorphe s'il ont la même caractéristique Euler-Poincaré



is homeomorphic to

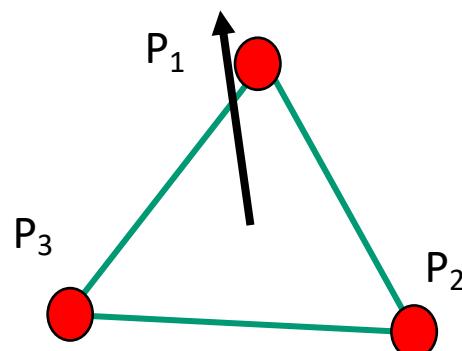


# Normale à une face

La normale a une face

- Elle permet de définir l'orientation d'une face
- Elle est utilisé pour défini l'extérieur/intérieur de la forme ou pour l'éclairage
- Elle est obtenue par le produit vectoriel de sa première arête avec la deuxième. L'ordre des sommets est donc important

$$N = \frac{(P_2 - P_1) \wedge (P_3 - P_2)}{\|(P_2 - P_1) \wedge (P_3 - P_2)\|}$$

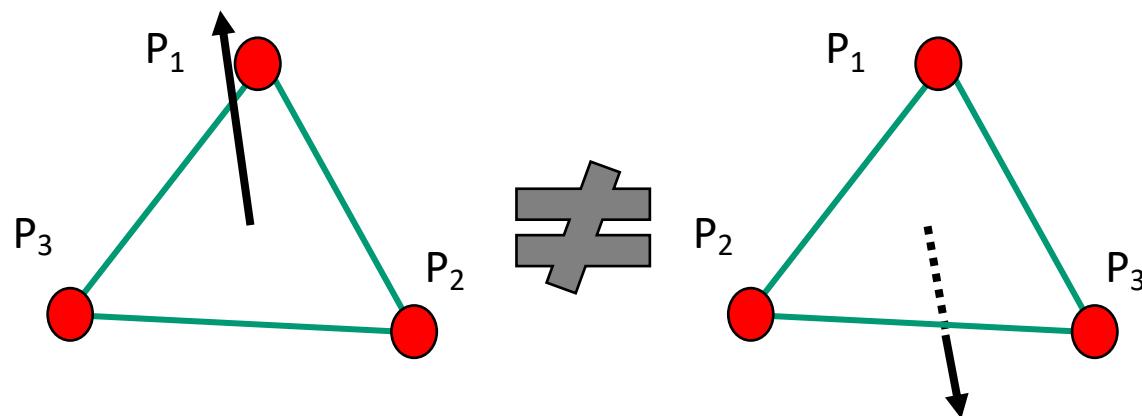


# Normale à une face

La normale a une face

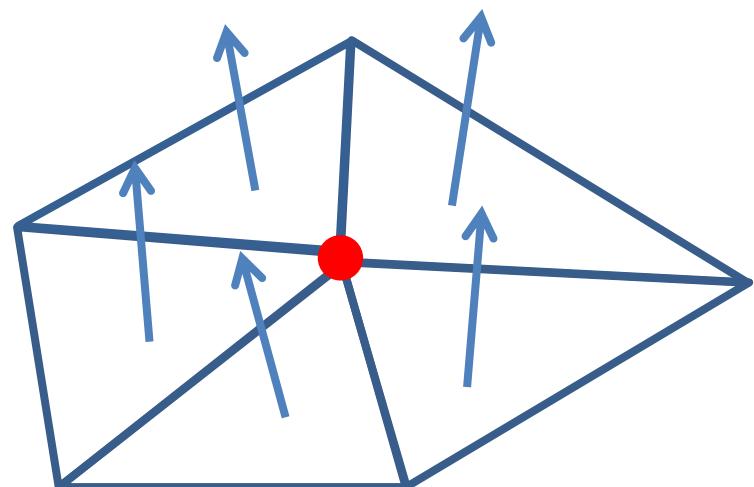
- Elle permet de définir l'orientation d'une face
- Elle est utilisé pour défini l'extérieur/intérieur de la forme ou pour l'éclairage
- Elle est obtenue par le produit vectoriel de sa première arête avec la deuxième. L'ordre des sommets est donc important

$$N = \frac{(P_2 - P_1) \wedge (P_3 - P_2)}{\|(P_2 - P_1) \wedge (P_3 - P_2)\|}$$



# Normales aux sommets

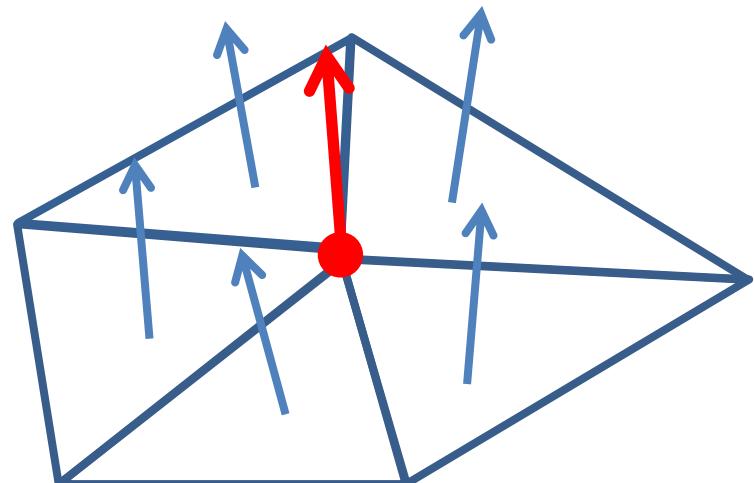
- On peut définir des normales pour chaque sommets
  - A partir des normales aux faces



# Normales aux sommets

- On peut définir des normales pour chaque sommets
  - A partir des normales aux faces
  - Approche naïve: somme normée des normales unitaires des faces adjacentes

$$N^s = \frac{\sum N^f_i}{\| \sum N^f_i \|}$$

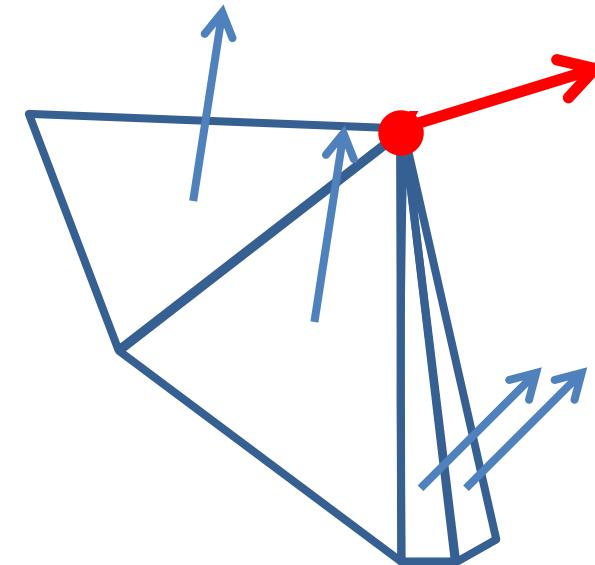
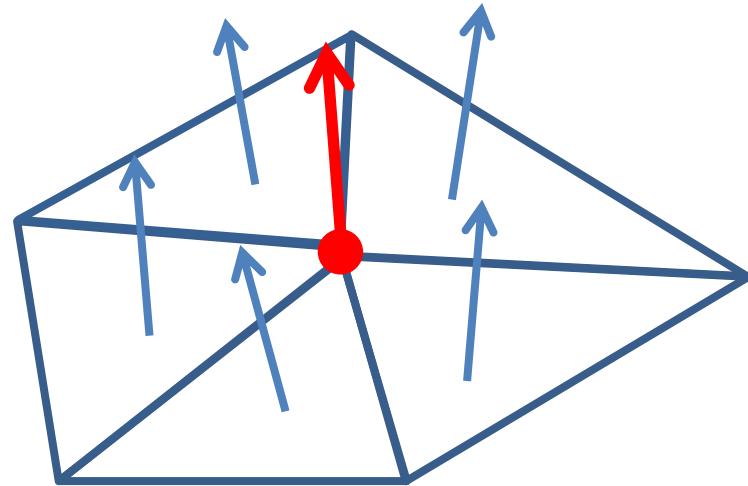


# Normales aux sommets

- On peut définir des normales pour chaque sommets
  - A partir des normales aux faces
  - Approche naïve: somme normée des normales unitaires des faces adjacentes

$$N^s = \frac{\sum N_i^f}{\| \sum N_i^f \|}$$

Solutions plus intelligentes?

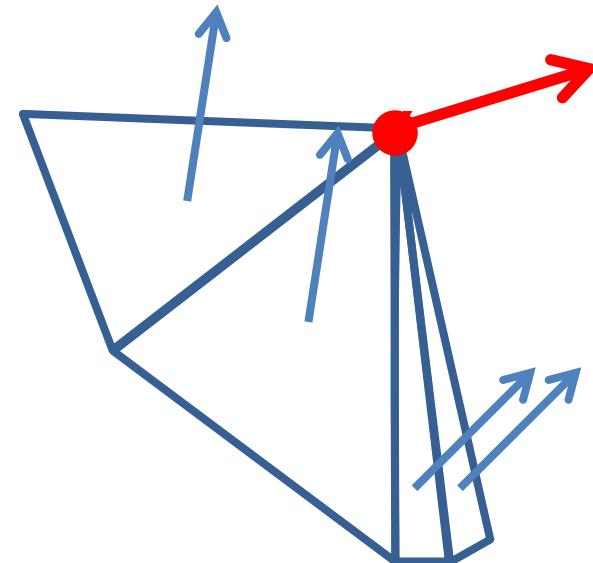
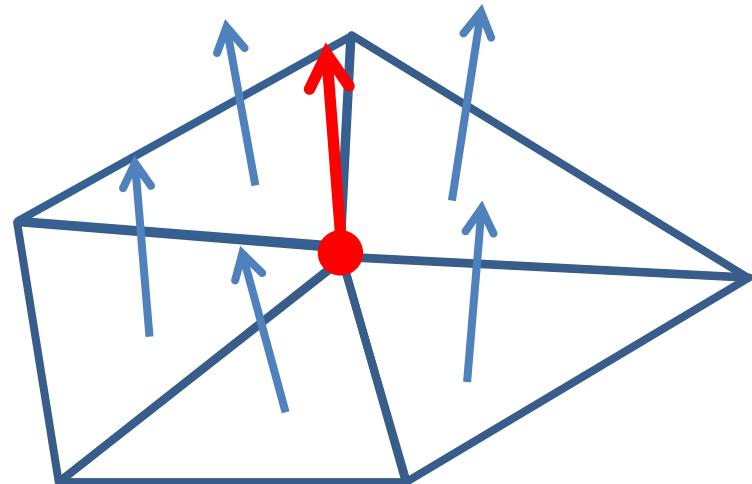


# Normales aux sommets

- On peut définir des normales pour chaque sommets
  - A partir des normales aux faces
  - Approches plus avancées : les normales sont pondérées

$$N^s = \frac{\sum \alpha_i N^f_i}{\| \sum \alpha_i N^f_i \|}$$

- En utilisant les angles que forment chaque facette au point
- En utilisant l'aire des facettes adjacentes

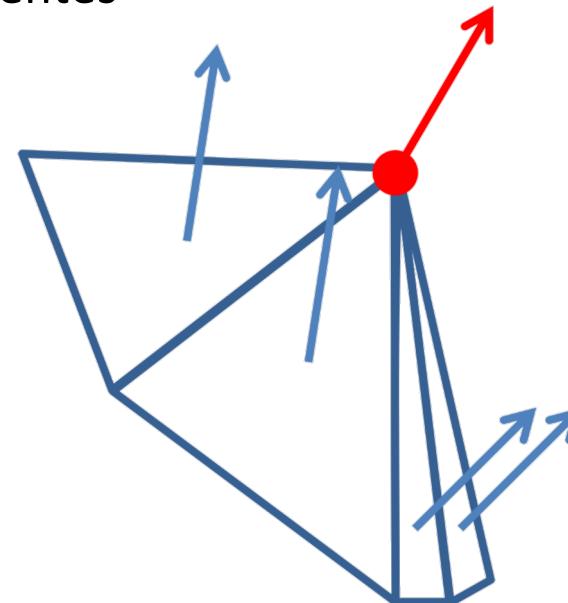
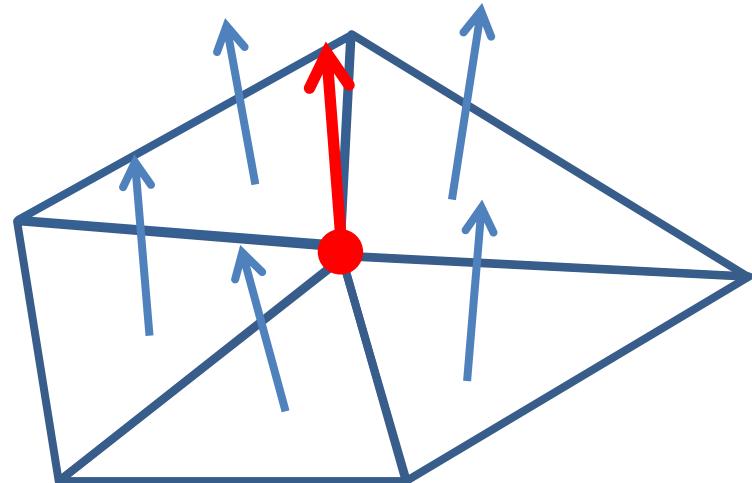


# Normales aux sommets

- On peut définir des normales pour chaque sommets
  - A partir des normales aux faces
  - Approches plus avancées : les normales sont pondérées

$$N^s = \frac{\sum \alpha_i N^f_i}{\| \sum \alpha_i N^f_i \|}$$

- En utilisant les angles que forment chaque facette au point
- En utilisant l'aire des facettes adjacentes



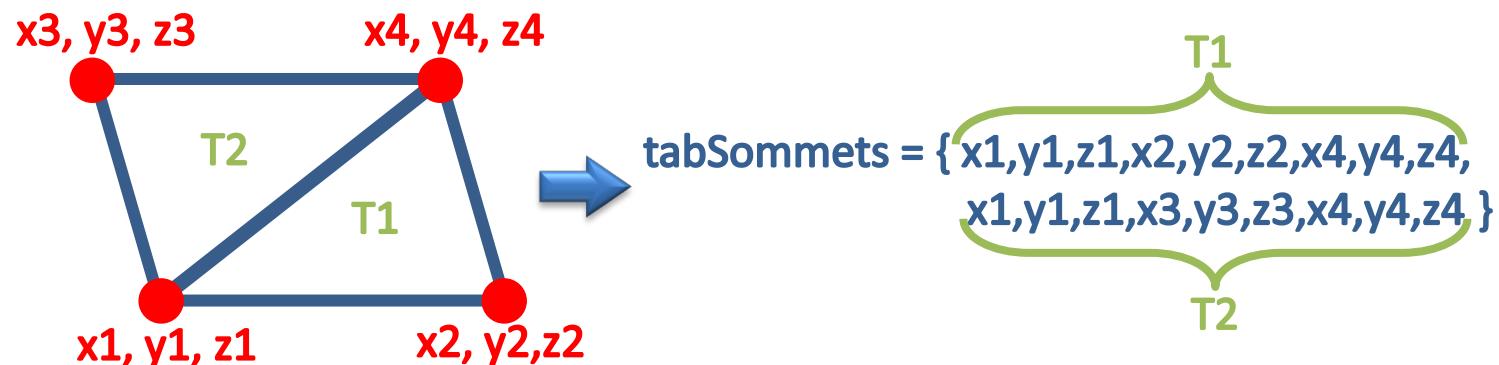
# Structures de données pour les maillages

- Ce qu'il faut représenter:
  - Les entités (sommets, arêtes, faces)
  - Leurs organisations (adjacences, incidences)
  - Leurs propriétés (positions, normales, couleurs, textures)
- Choisir une représentation adaptée aux opérations que l'on souhaite effectuer sur le maillage :
  - Représentation compacte
    - Transmission efficace à la GPU, sur le réseau, ...
  - Représentation optimisée pour le parcours
    - Pour passer d'un sommet à ses voisins, d'une face à ses voisines, d'une faces à ses arêtes, etc.

# Structure de données

- Approche naïve:
  - Maillage représenté par un unique tableau de sommet (maillage non indexé)
    - 4 octets par coordonnée (un flottant)
    - 9 coordonnées par faces
    - 2 fois plus de faces que de sommets

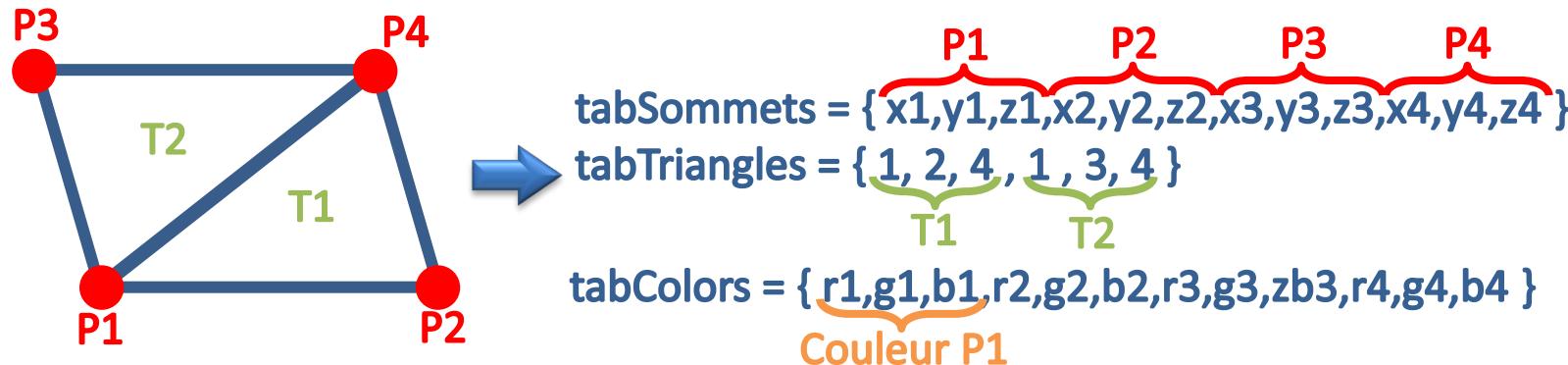
Pour un maillage composé de  $S$  sommets, on a donc besoin de  $4*9*2*S = 72*S$  octets.



# Structure de données

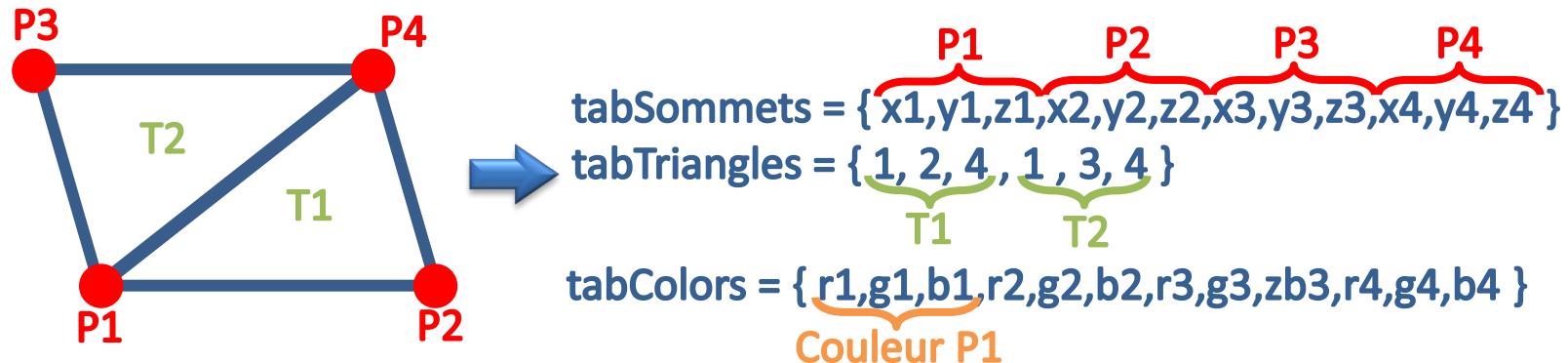
## Sommets partagés

- Approche classique
  - Maillage représenté par plusieurs tableaux: un pour les sommets, un pour les faces, un pour les couleurs. Maillage indexé.
  - les facettes sont définies en donnant les index des sommets qui les composent.
  - Les coordonnées des sommets ne sont plus répétées.
  - En général, le fichier commence par le nombre de sommets et le nombre de facettes



# Structure de données Sommets partagés

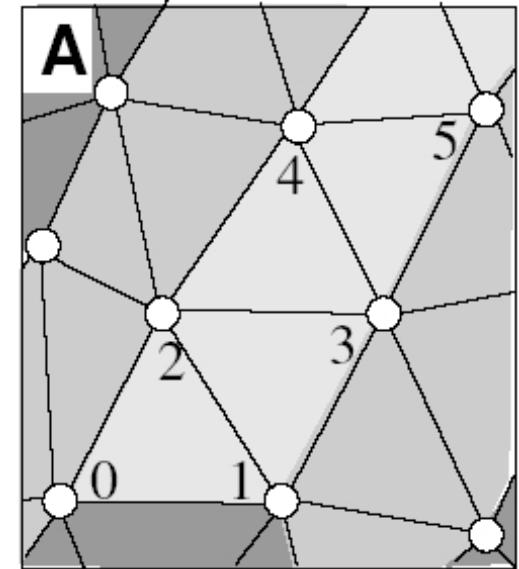
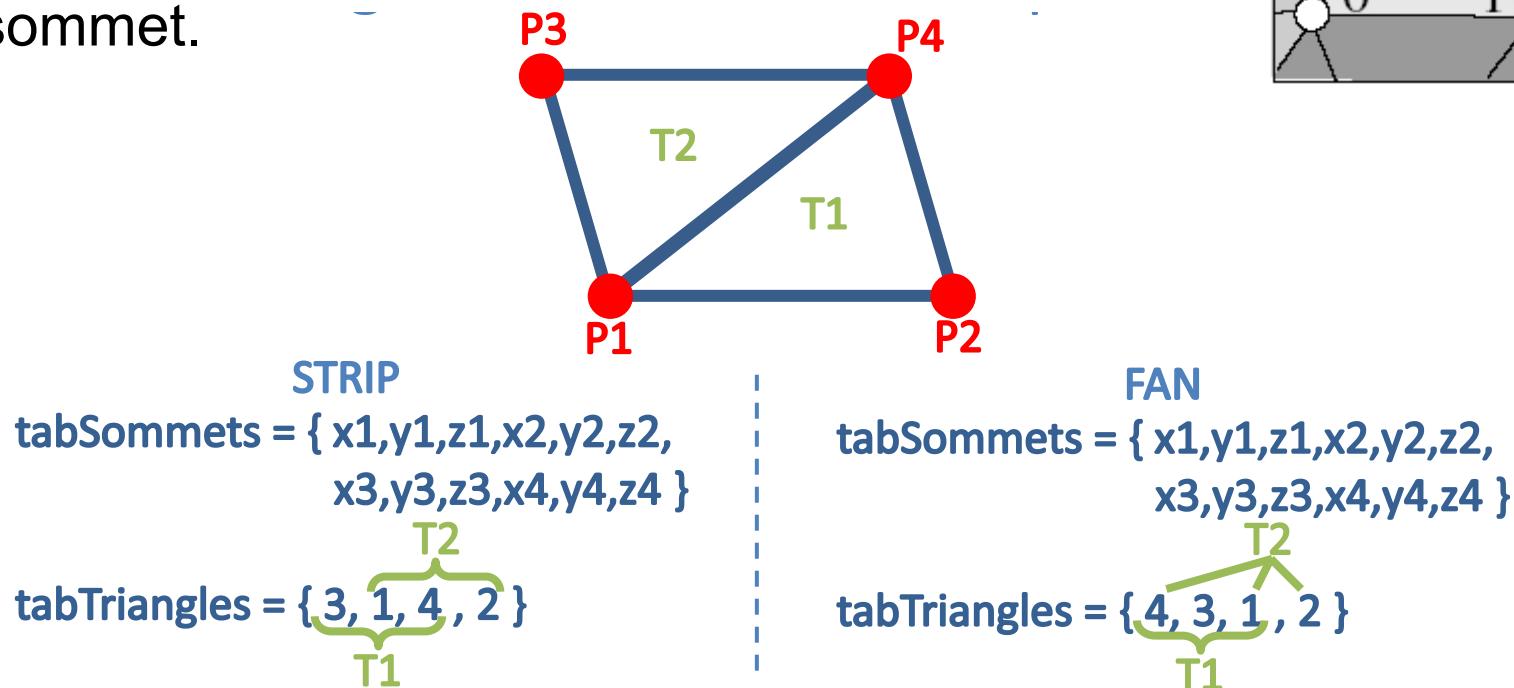
- Approche classique
  - Maillage représenté par plusieurs tableaux: un pour les sommets, un pour les faces, un pour les couleurs. Maillage indexé.
  - les facettes sont définies en donnant les index des sommets qui les composent.
  - Les coordonnées des sommets ne sont plus répétées.
  - En général, le fichier commence par le nombre de sommets et le nombre de facettes



➤ Pas pratique pour les parcours topologiques (accéder aux voisins, etc) !!!!

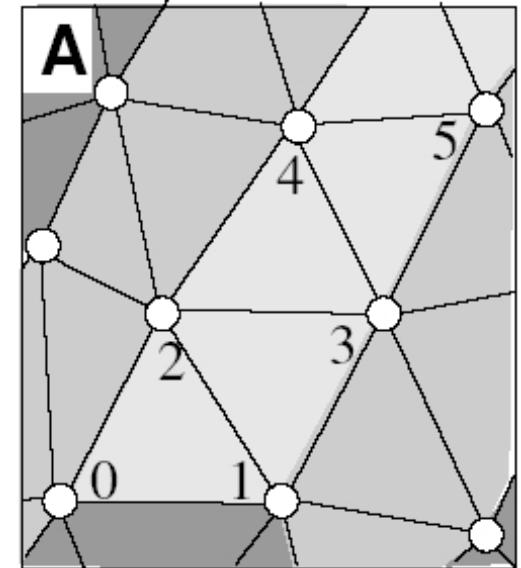
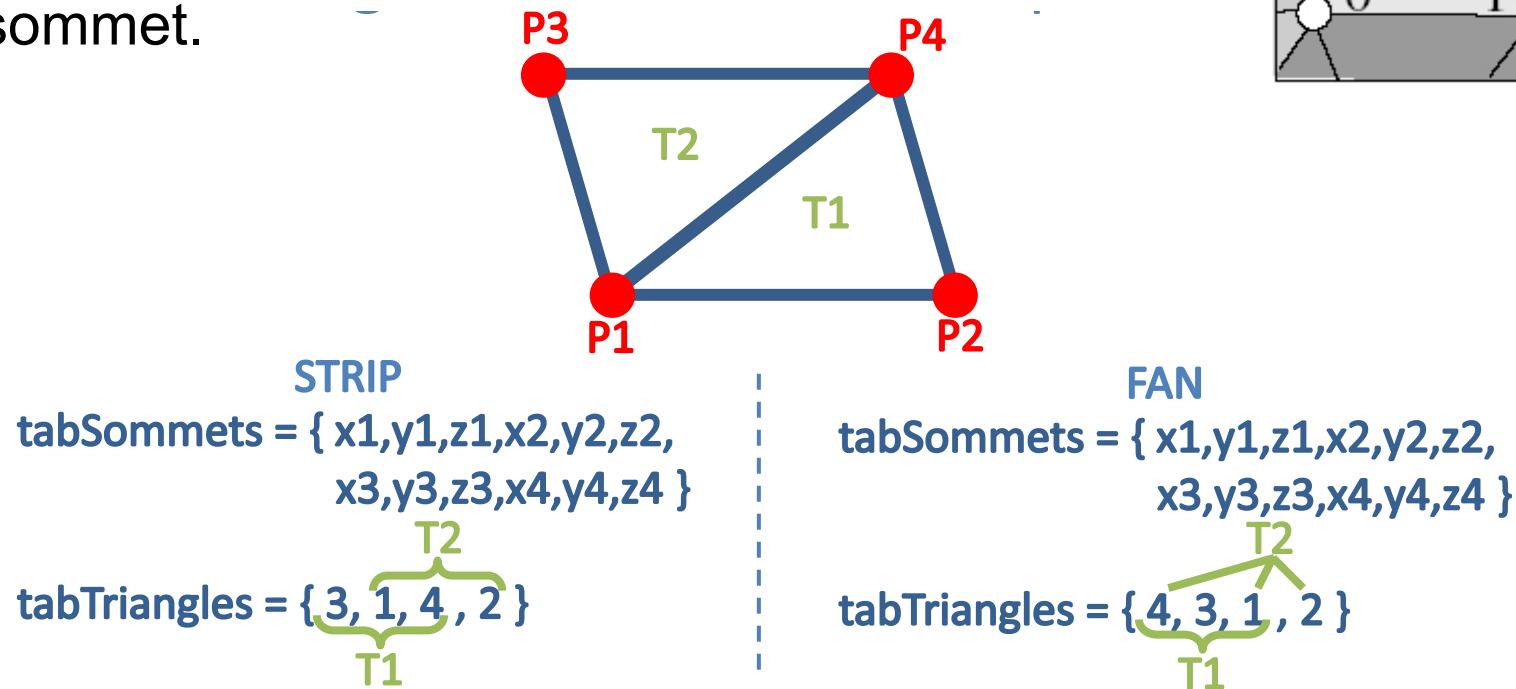
# Structure de données

- Approche Strip ou Fan
  - La topologie est codée de façon implicite dans l'ordre des sommets
  - Chaque sommet est donc visité deux fois
- Strip : Maillage représenté par une bande
- Fan : Maillage défini autour d'un premier sommet.



# Structure de données

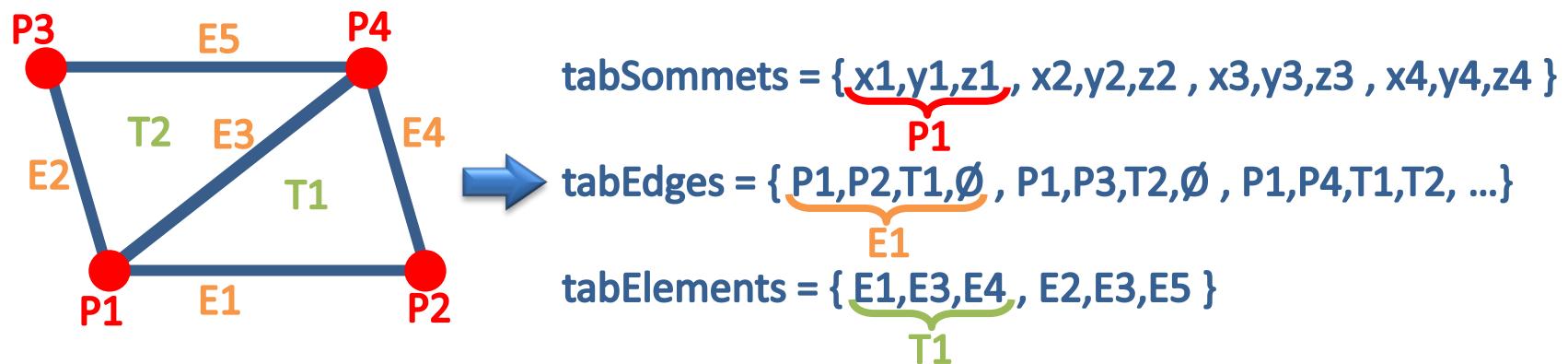
- Approche Strip ou Fan
  - La topologie est codée de façon implicite dans l'ordre des sommets
  - Chaque sommet est donc visité deux fois
- Strip : Maillage représenté par une bande
- Fan : Maillage défini autour d'un premier sommet.



➤ Pas adapté à tous les maillages et pas pratique encore pour la topologie !!!!

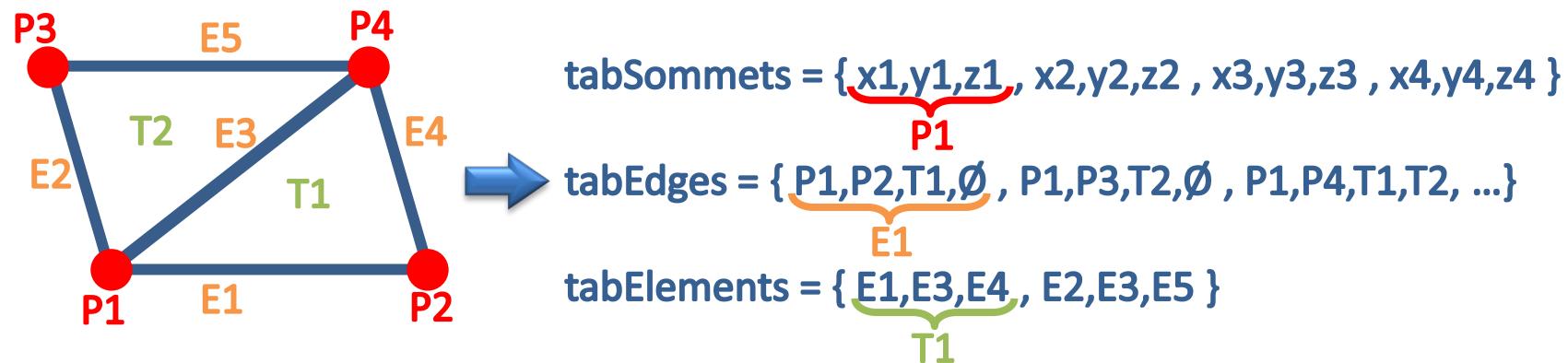
# Structure de données

- Approche par arêtes
  - Maillage représenté par
    - Des sommets définis par 3 coordonnées,
    - Des arêtes définies par 2 sommets et 2 faces
    - Des faces définies par 3 arêtes



# Structure de données

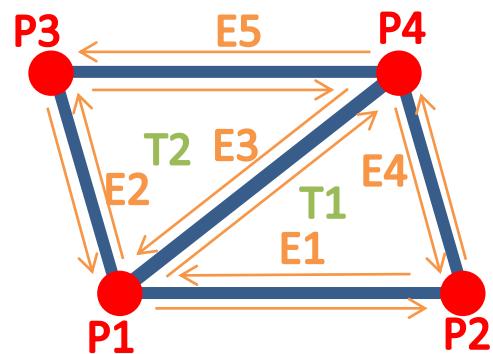
- Approche par arêtes
  - Maillage représenté par
    - Des sommets définis par 3 coordonnées,
    - Des arêtes définies par 2 sommets et 2 faces
    - Des faces définies par 3 arêtes



➤ Couteux en mémoire. Topologie simple. N'encode pas l'orientation !!!

# Structure de données

- Approche par demi-arêtes
  - Chaque arête est représenté par 2 demi-arêtes d'orientation opposée
  - Chaque demi-arête est définie par
    - La demi-arête opposée,
    - L'arête suivante dans la face
    - La face que borde l'arête
    - Le sommet extrémité



tabSommets = { x1,y1,z1, x2,y2,z2 , x3,y3,z3 , x4,y4,z4 }

P1

tabEdges = { E1',E3',T1,P2 , E2',E5',T2,P3 , E3',E2,T2,P1 , ... }

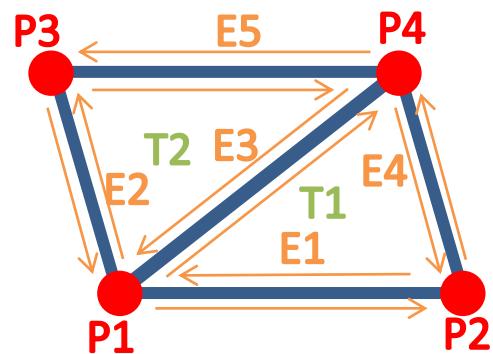
E1

E2

tabElement = {E1, E2}

# Structure de données

- Approche par demi-arêtes
  - Chaque arête est représenté par 2 demi-arêtes d'orientation opposée
  - Chaque demi-arête est définie par
    - La demi-arête opposée,
    - L'arête suivante dans la face
    - La face que borde l'arête
    - Le sommet extrémité



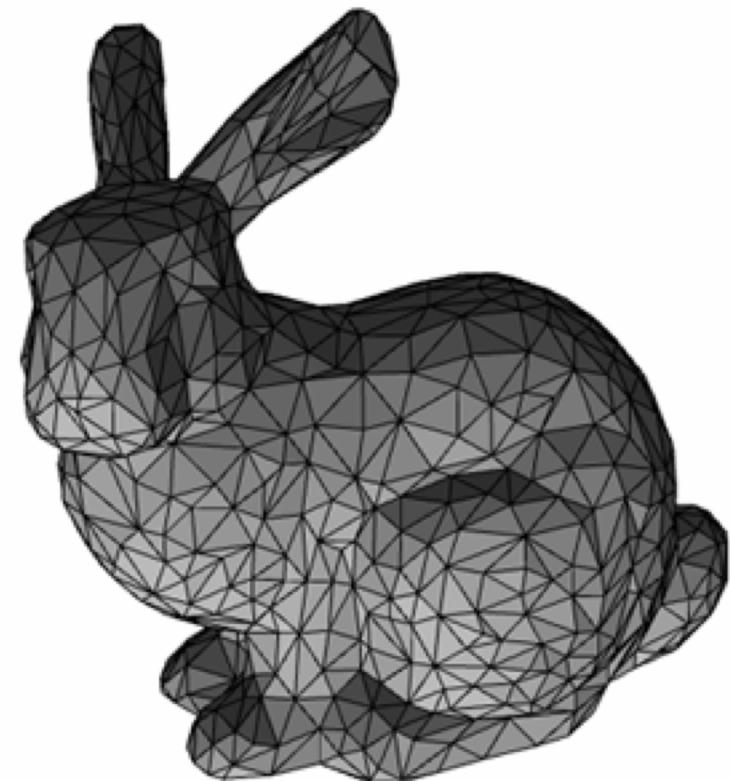
tabSommets = {x1,y1,z1, x2,y2,z2 , x3,y3,z3 , x4,y4,z4 }  
tabEdges = {E1',E3',T1,P2, E2',E5',T2,P3, E3',E2,T2,P1 , ...}  
tabElement = {E1, E2}

# Implémentation des demi-arêtes

- Mise en œuvre
  - Utilisation aisée
  - Implémentation robuste des opérateurs assez complexes
- Bibliothèque open-sources
  - OpenMesh ([www.openmesh.org](http://www.openmesh.org))
  - VCG-Lib dans MeshLab ([vcg.sf.net](http://vcg.sf.net))
  - CGAL [www.cgal.org](http://www.cgal.org)

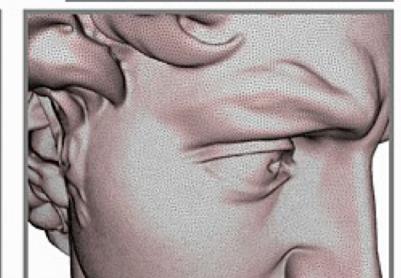
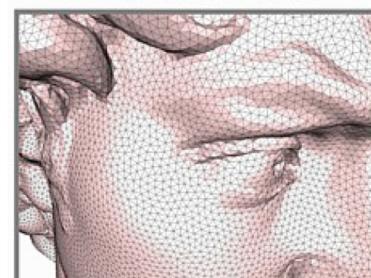
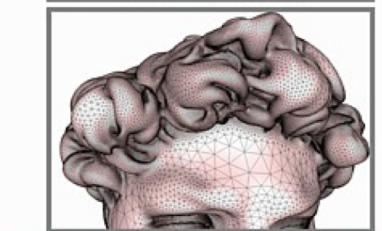
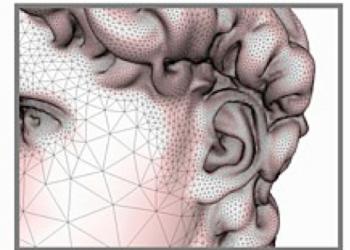
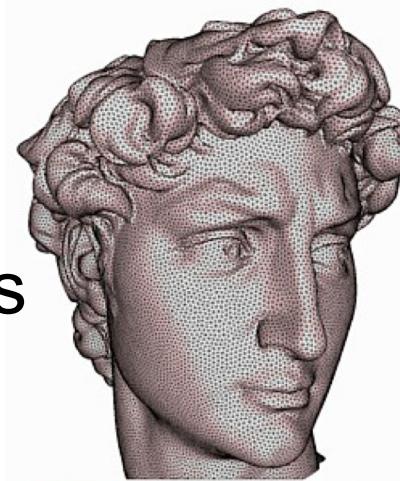
# Formats de fichier standards

- Indexé
  - OFF
  - OBJ
  - PLY
- Non Indexé
  - STL



# Conclusion

- Représentation par maillage
  - Très utilisé pour le rendu
  - Différentes représentations optimisant le stockage ou la manipulation
- La suite
  - Représentation de complexes cellulaires
  - Algorithmie sur les maillages
    - Subdivision
    - Filtres de régularisation, etc



# Visualisation OpenGL d'un maillage

## Construction de l'objet :

Sa liste de sommets

```
GLfloat sommets[] = {x0, y0, z0, ... xn, yn, zn};
```

Ses attributs, par exemple les normales

```
GLfloat normales[] = {nx0, ny0, nz0, ..., nxn, nyn, nzn};
```

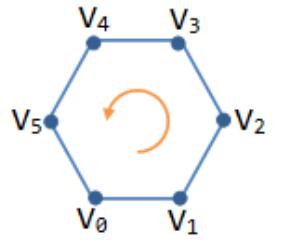
Sa liste d'index

```
GLuint index[] = {0, 1, 2, 5, 0, 4, ...};
```

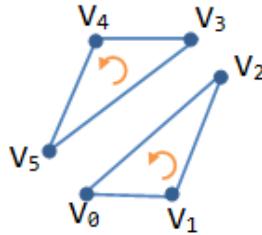
# Visualisation OpenGL d'un maillage

Types de représentation pour OpenGL :

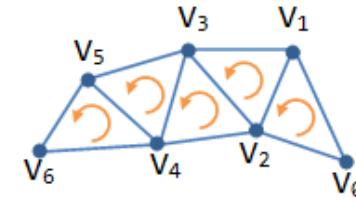
- `GL_TRIANGLES` : il faut donner les index des facettes triangulaires,
- `GL_QUADS` : il faut donner les index des quadrilatères,
- `GL_TRIANGLE_STRIP` : la liste ordonnée des index pointant sur les sommets des bandes de triangles. Dans le tableau d'index, il est possible d'utiliser un séparateur pour séparer les différentes bandes de triangles,
- `GL_QUAD_STRIP` : idem, mais liste de quadrilatères,
- ...



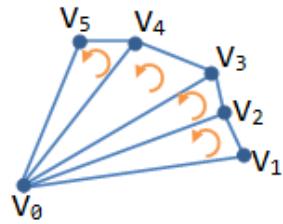
`GL_POLYGON`



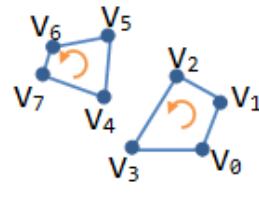
`GL_TRIANGLES`



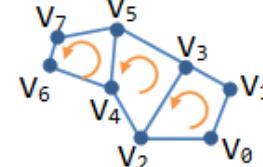
`GL_TRIANGLE_STRIP`



`GL_TRIANGLE_FAN`



`GL_QUADS`



`GL_QUAD_STRIP`

# Rendu OpenGL

```
// Méthode simple mais pas rapide  
glBegin( GL_TRIANGLES); // on trace l'ensemble des triangles  
  
for (int i = 0; i < nb_index; ++i){  
    for (int j = 0; j < 3; ++j){  
        glNormal3fv(normal[index[i]]);  
        glVertex3fv(sommet[index[i][j]]);  
    }  
}  
glEnd();
```

# Visualisation OpenGL

## Les VertexArray

// Activation du mode de tracé par tableaux de sommets

```
glEnableClientState (GL_VERTEX_ARRAY);
```

```
glEnableClientState (GL_NORMAL_ARRAY); // si on utilise les normales
```

```
glEnableClientState (...); // si on utilise d'autres attributs (couleur, coordonnées texture, ...)
```

// Affectation des différents tableaux

```
glVertexPointer (3, GL_FLOAT, 0, sommets);
```

```
glNormalPointer (GL_FLOAT, 0, normales); // si on utilise les normales
```

```
// ... + autres tableaux s'il y a (couleur,...)
```

// Tracé du maillage triangulaire

```
glDrawElements (GL_TRIANGLES, nb_index, GL_UNSIGNED_INT, index);
```

// Désactivation du mode tracé

```
glDisableClientState ( .....); // un pour chaque glEnableClientState
```

# Visualisation OpenGL

## Les Vertex Buffer Objects (VBO)

// Initialisation des tableaux

```
Gliunt TabIdentBuffer[nbIdentBuffer];
```

```
glGenBuffer(nbIdentBuffer)
```

// Envoie des tableaux

```
glBindBuffer(GL_ARRAY_BUFFER, TabIdentBuffer[identPoint]);
```

```
glBufferData(GL_ARRAY_BUFFER, tailleTab*sizeof(float), TabPoint, GL_STATIC_DRAW);
```

```
glBindBuffer(GL_ARRAY_BUFFER, 0);
```

// peut le recuperer

```
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, TabIdentBuffer[identElem]);
```

```
TabElem = glMapBuffer(GL_ELEMENT_ARRAY_BUFFER, GL_READ_ONLY);
```

```
glUnmapBuffer(GL_ELEMENT_ARRAY_BUFFER);
```

// On le dessine

```
glBindBuffer(GL_ARRAY_BUFFER, TabIdentBuffer[identColor]);
```

```
glColorPointer(4, GL_FLOAT, 0, BUFFER_OFFSET(0));
```