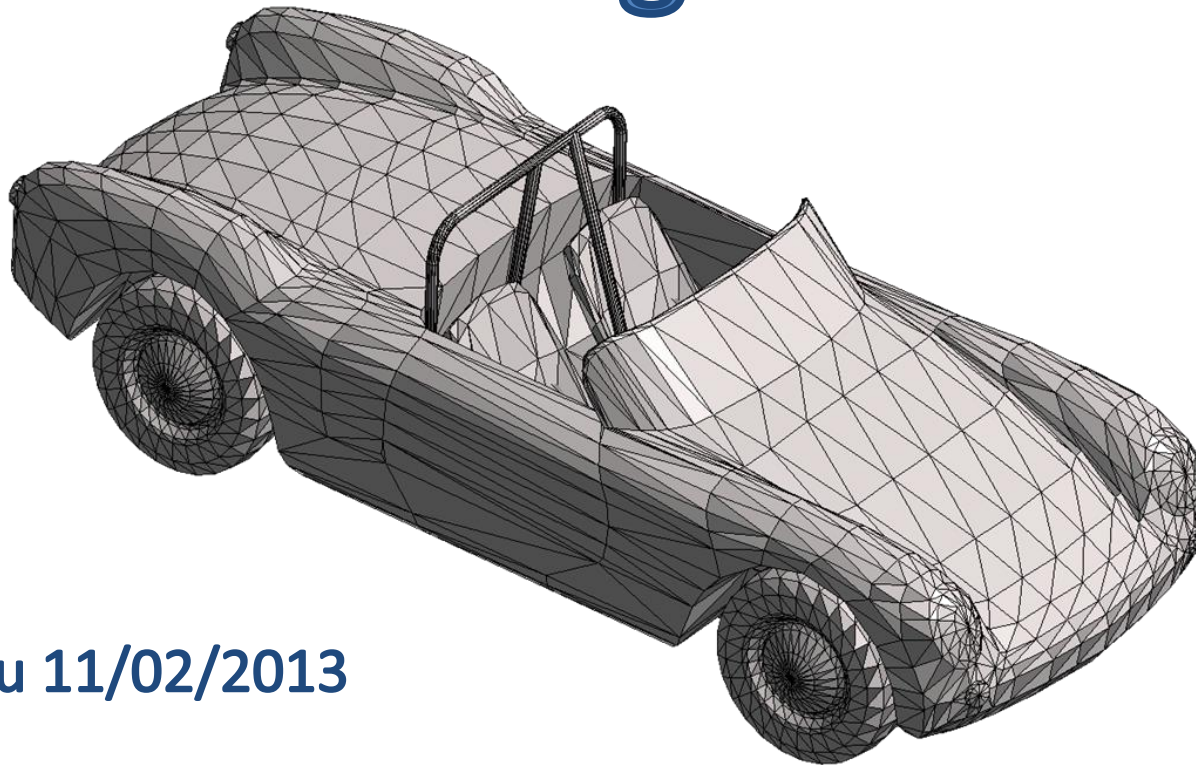


Maillages 3D



Cours du 11/02/2013

Plan

- Introduction
- Propriétés de base
- Structures de données
- Visualisation OpenGL
- Conclusion

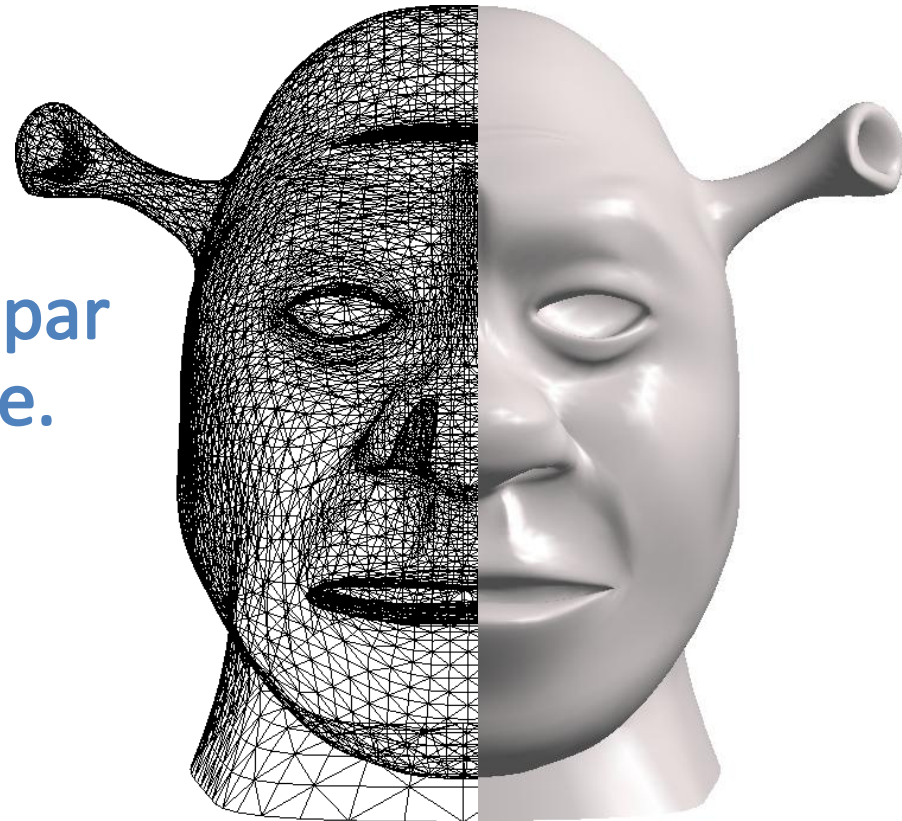
Introduction

- Une structure standard d'affichage de scènes complexes 3D.



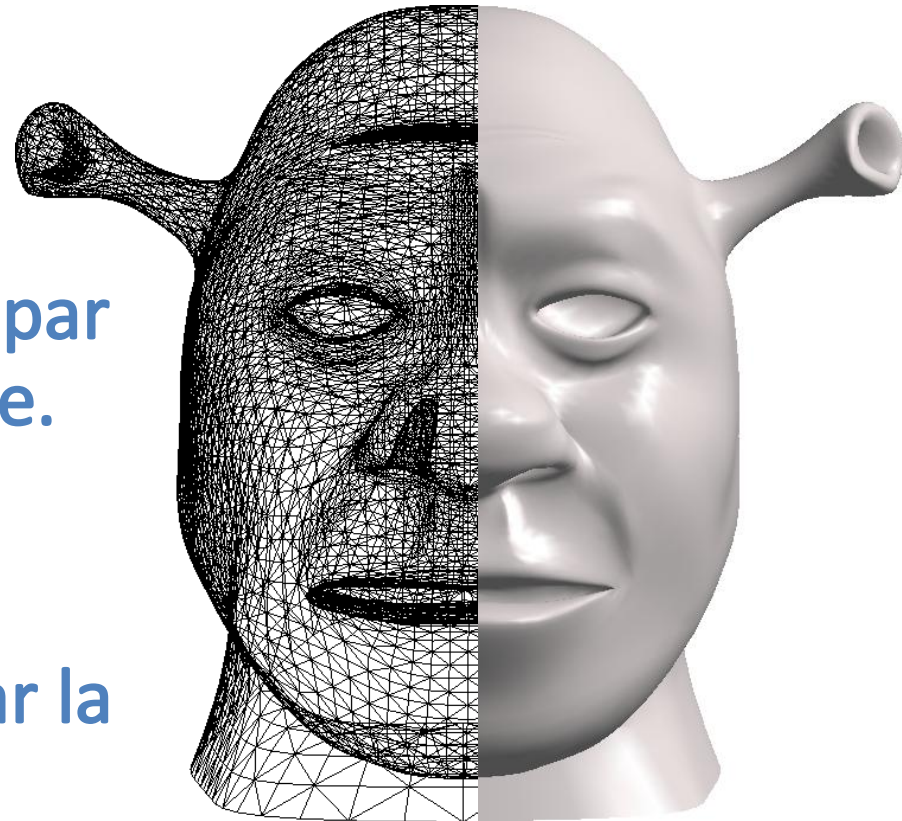
Introduction

- Une structure standard d'affichage de scènes complexes 3D.
- Représentation de la face par une ensemble de polygone.
- Souvent des triangles (simplexe pour une face).



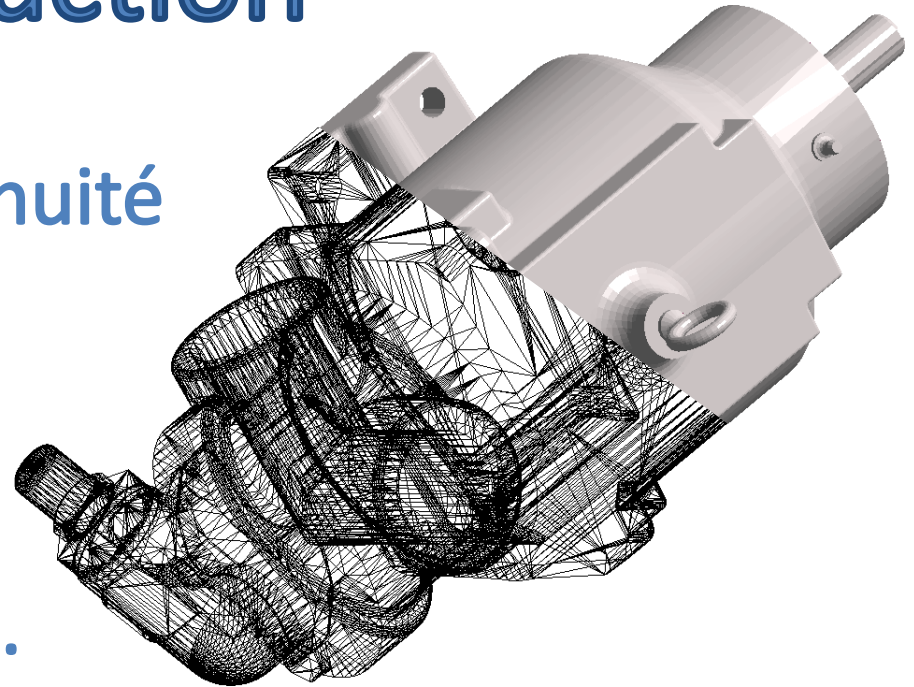
Introduction

- Une structure standard d'affichage de scènes complexes 3D.
- Représentation de la face par une ensemble de polygone.
- Souvent des triangles (simplexe pour une face).
- Visualisation optimisée par la majorité des cartes graphiques.



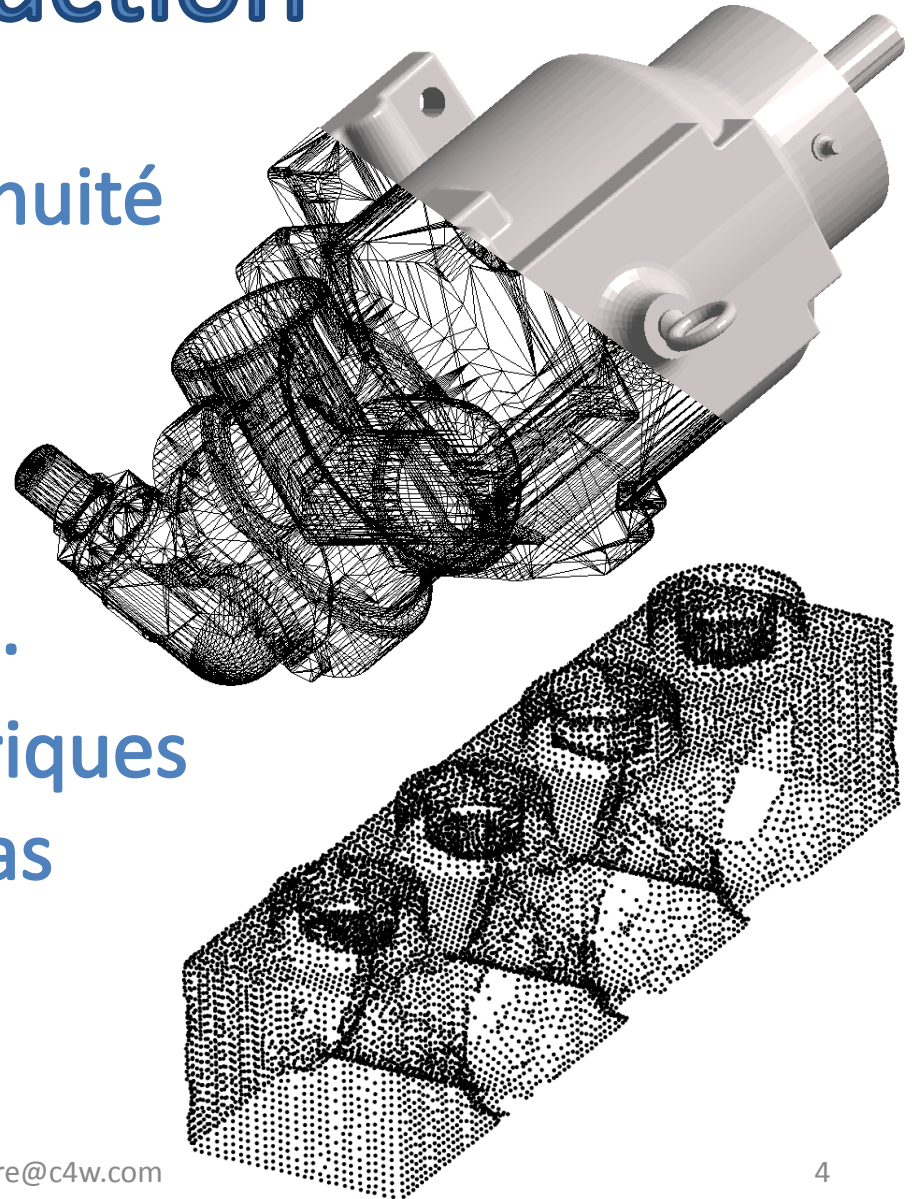
Introduction

- Continuité C^0 (discontinuité aux arêtes).
- Informations sur la géométrie et sur la topologie de la surface.



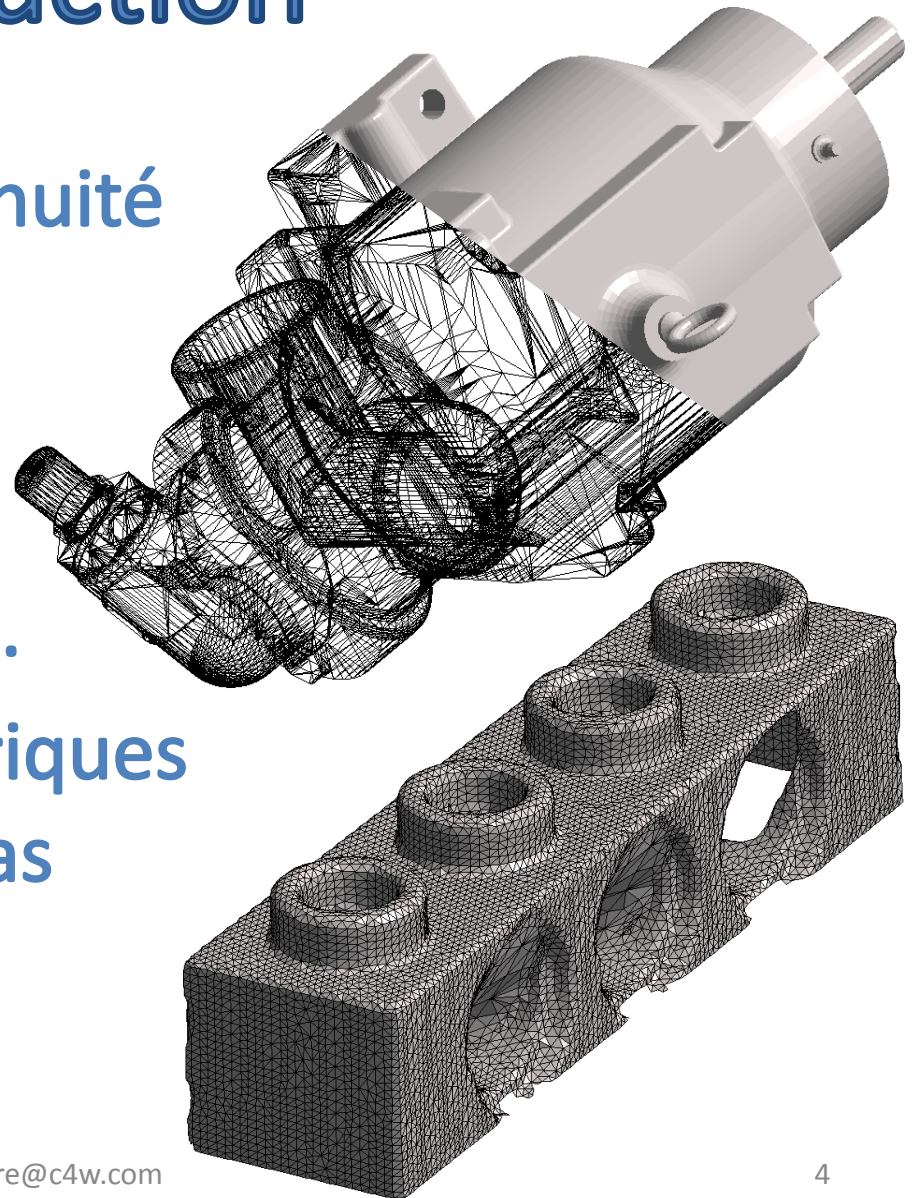
Introduction

- Continuité C^0 (discontinuité aux arêtes).
- Informations sur la géométrie et sur la topologie de la surface.
- Les équations géométriques des surfaces ne sont pas toujours disponibles (scans).



Introduction

- Continuité C^0 (discontinuité aux arêtes).
- Informations sur la géométrie et sur la topologie de la surface.
- Les équations géométriques des surfaces ne sont pas toujours disponibles (scans).



Propriétés : vocabulaire

- Entités d'un maillage :

Propriétés : vocabulaire

- Entités d'un maillage :
 - sommets (x, y, z)



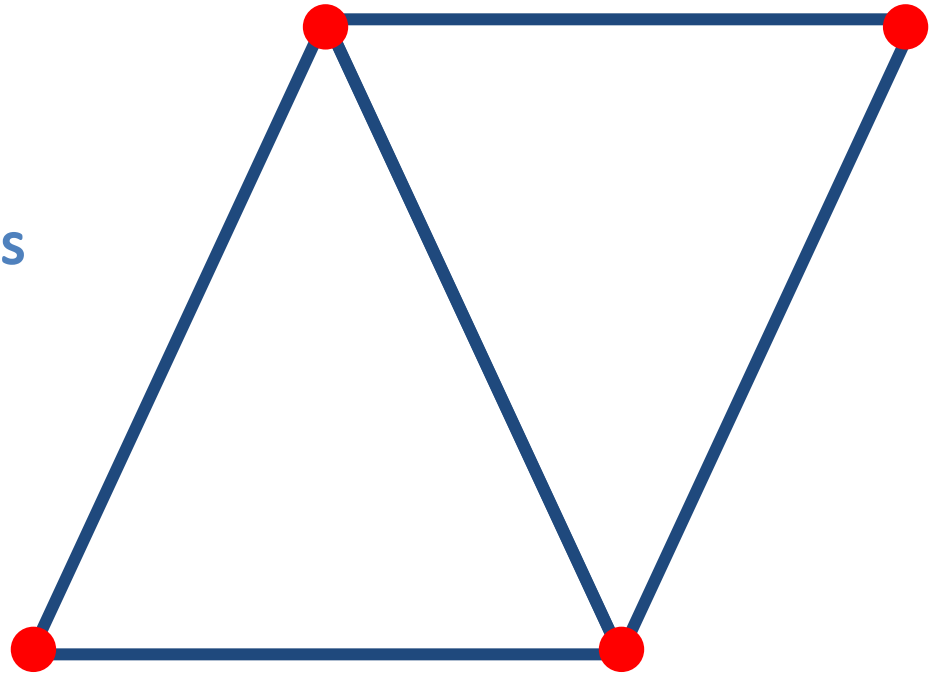
Propriétés : vocabulaire

- Entités d'un maillage :

- sommets (x, y, z)

- arêtes :

- définies par 2 sommets



Propriétés : vocabulaire

- Entités d'un maillage :

- sommets (x, y, z)

- arêtes :

- définies par 2 sommets

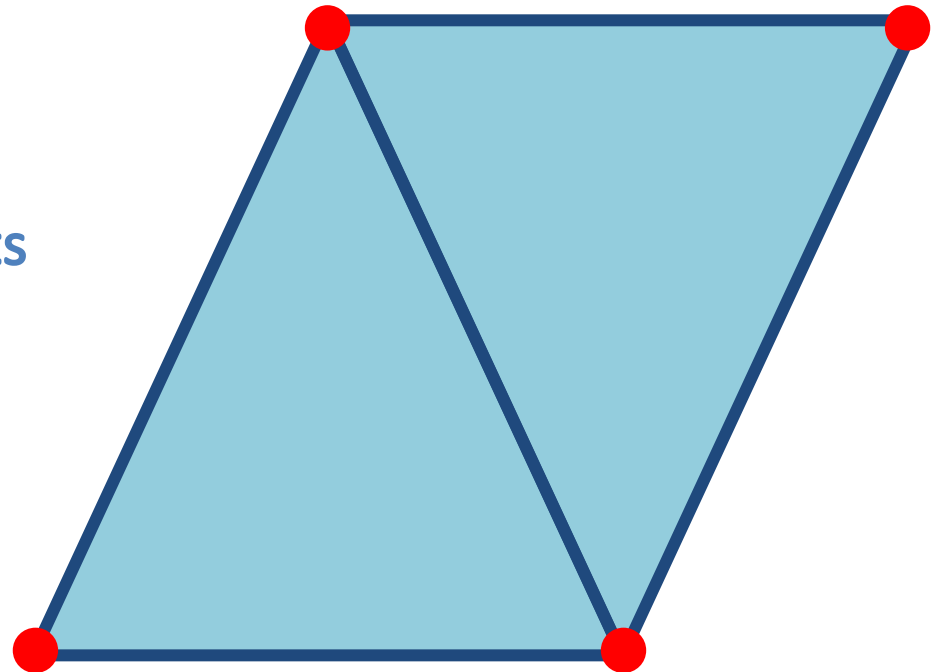
- faces :

- définis par n sommets

ou

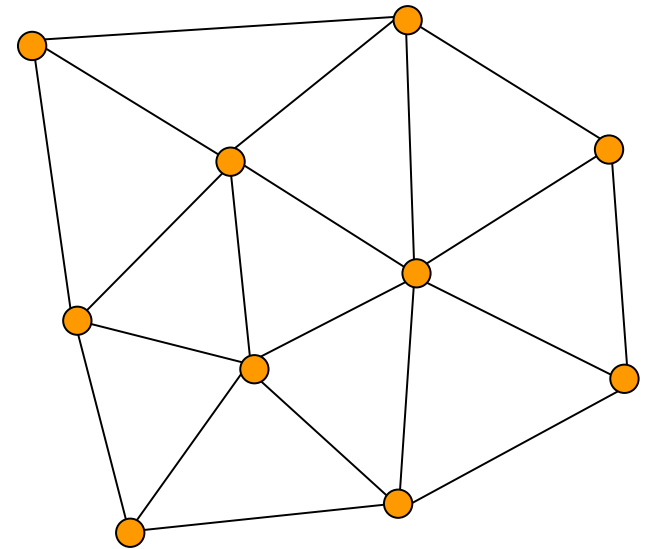
- définis par n arêtes

- en générale des triangles ($n = 3$)



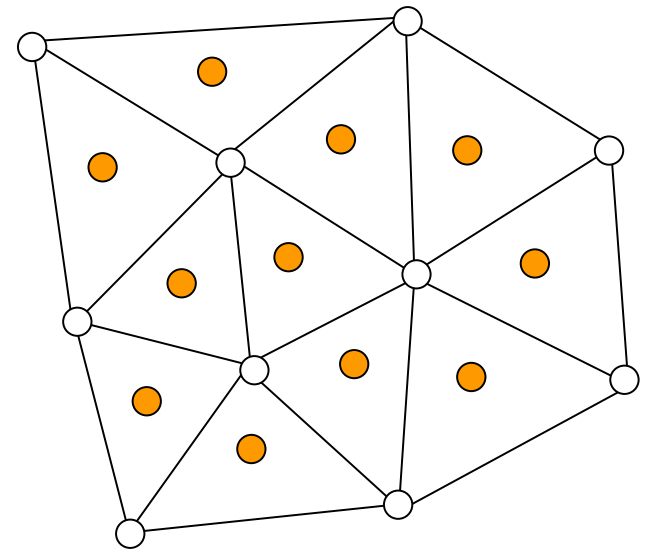
Propriétés : dualité

- Maillage dual :



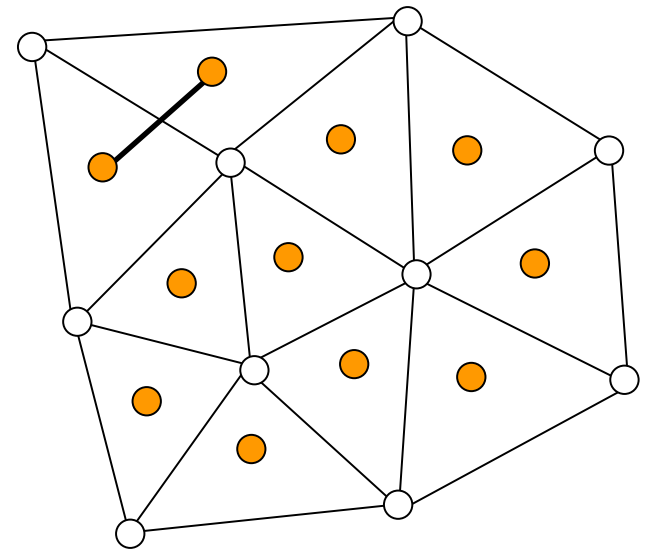
Propriétés : dualité

- Maillage dual :
 - chaque face est remplacé par un sommet → barycentre de la face,



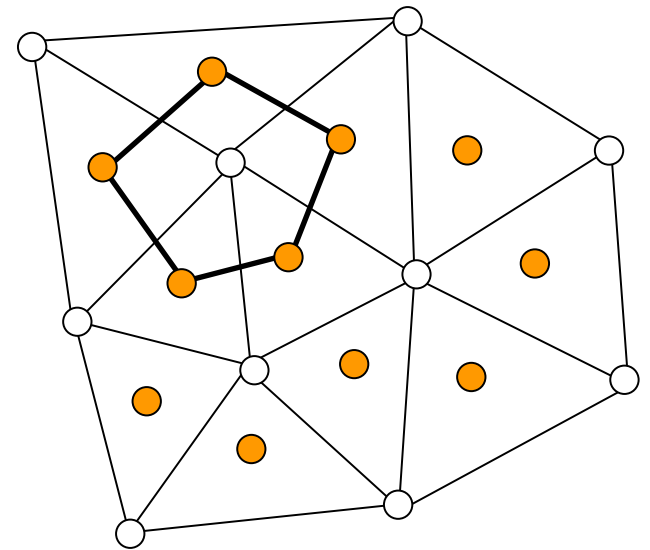
Propriétés : dualité

- Maillage dual :
 - chaque face est remplacé par un sommet → barycentre de la face,
 - une arête du dual relie deux sommets si les faces correspondantes sont voisines dans le maillage d'origine,



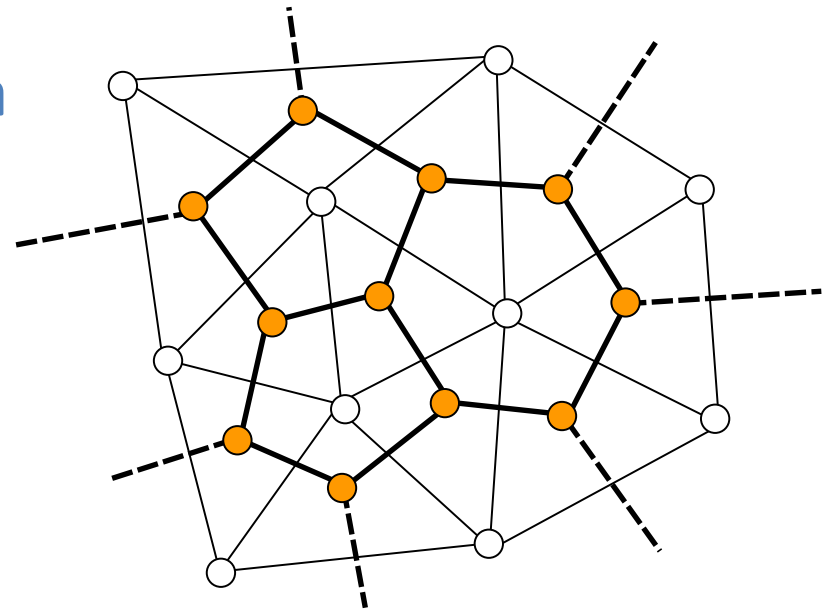
Propriétés : dualité

- Maillage dual :
 - chaque face est remplacé par un sommet → barycentre de la face,
 - une arête du dual relie deux sommets si les faces correspondantes sont voisines dans le maillage d'origine,
 - les points sont remplacés par des faces,



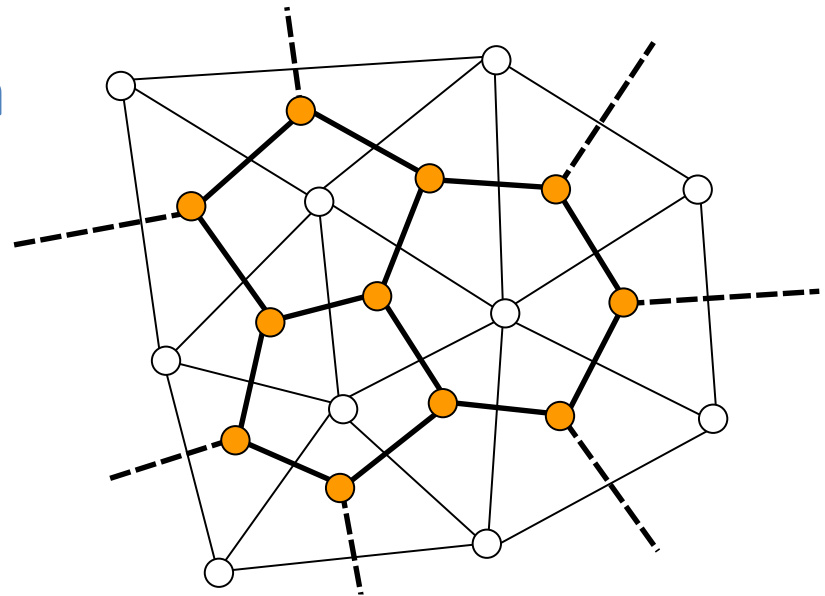
Propriétés : dualité

- Maillage dual :
 - chaque face est remplacé par un sommet → barycentre de la face,
 - une arête du dual relie deux sommets si les faces correspondantes sont voisines dans le maillage d'origine,
 - les points sont remplacés par des faces,



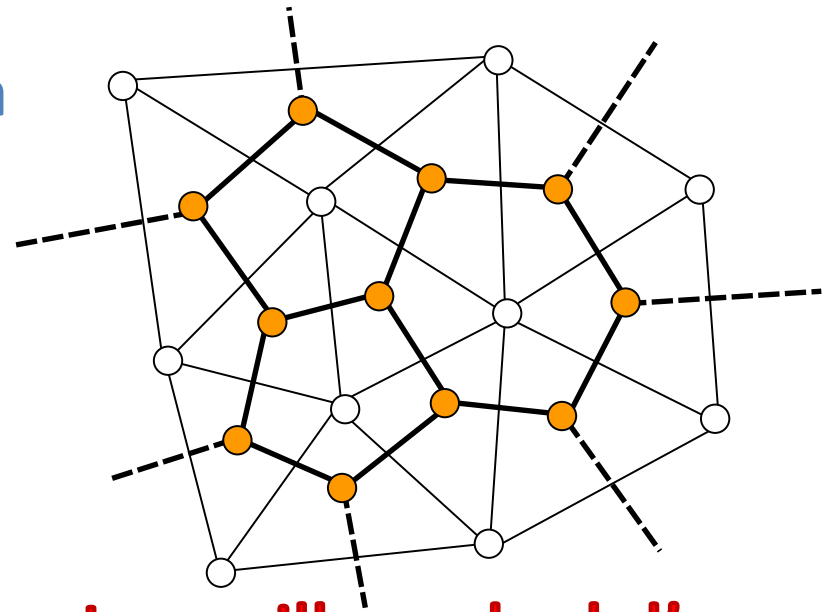
Propriétés : dualité

- Maillage dual :
 - chaque face est remplacé par un sommet → barycentre de la face,
 - une arête du dual relie deux sommets si les faces correspondantes sont voisines dans le maillage d'origine,
 - les points sont remplacés par des faces,
 - ➔ les objets de dimension k du maillage originale sont remplacés par des objets de dimension $(2-k)$ dans le dual.



Propriétés : dualité

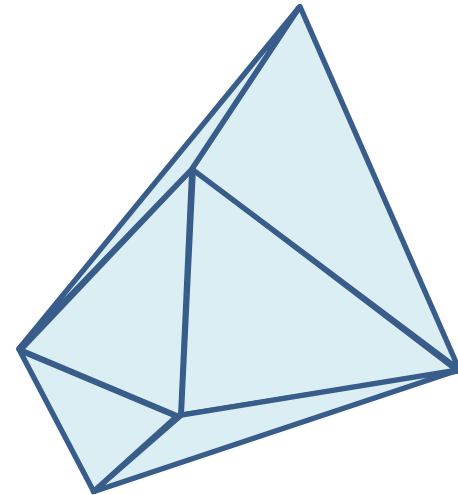
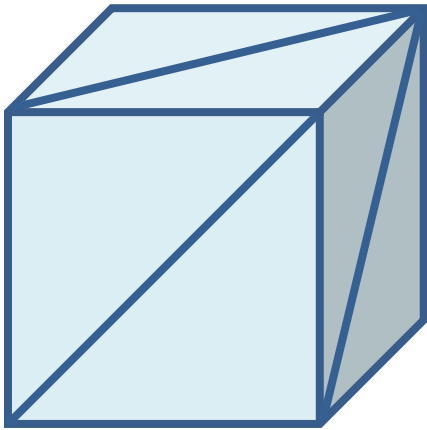
- Maillage dual :
 - chaque face est remplacé par un sommet → barycentre de la face,
 - une arête du dual relie deux sommets si les faces correspondantes sont voisines dans le maillage d'origine,
 - les points sont remplacés par des faces,
 - ➔ les objets de dimension k du maillage originale sont remplacés par des objets de dimension $(2-k)$ dans le dual.



Le maillage dual d'un maillage dual est égale au maillage original si celui-ci est fermé.

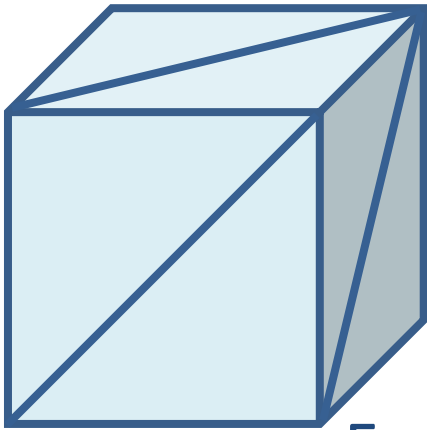
Propriétés : fermeture

- Un maillage est dit fermé si :
 - il n'a pas de bord,
 - ➔ toutes les arêtes du maillage sont au moins partagées par deux triangles

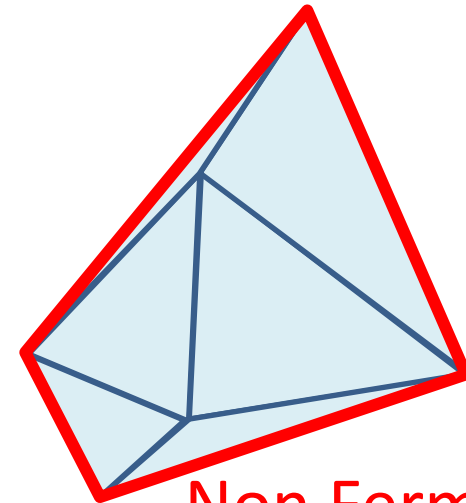


Propriétés : fermeture

- Un maillage est dit fermé si :
 - il n'a pas de bord,
 - ➔ toutes les arêtes du maillage sont au moins partagées par deux triangles



Fermé



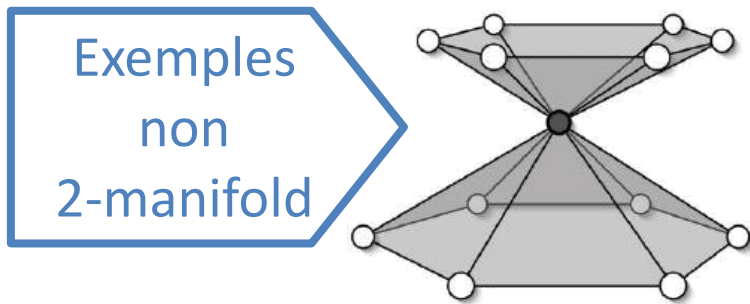
Non Fermé

Propriétés : 2-manifold

- Un maillage est 2-manifold si :
 - une sphère (rayon > 0) placée en n'importe quelle point à une intersection avec le maillage correspondante à une unique surface,

Propriétés : 2-manifold

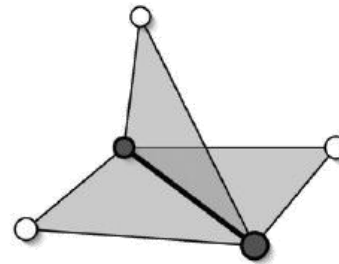
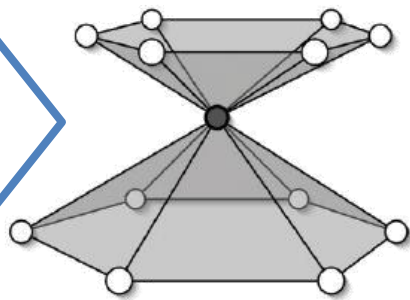
- Un maillage est 2-manifold si :
 - une sphère (rayon > 0) placée en n'importe quelle point à une intersection avec le maillage correspondante à une unique surface,



Propriétés : 2-manifold

- Un maillage est 2-manifold si :
 - une sphère (rayon > 0) placée en n'importe quelle point à une intersection avec le maillage correspondante à une unique surface,
 - il ne contient que des arêtes partagées par au plus deux triangles,

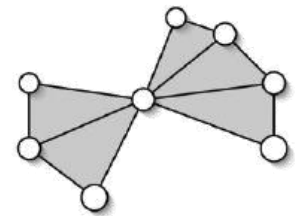
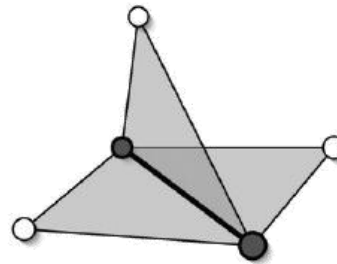
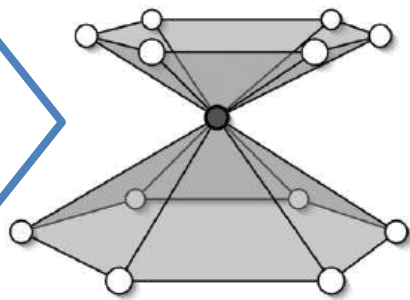
Exemples
non
2-manifold



Propriétés : 2-manifold

- Un maillage est 2-manifold si :
 - une sphère (rayon > 0) placée en n'importe quelle point à une intersection avec le maillage correspondante à une unique surface,
 - il ne contient que des arêtes partagées par au plus deux triangles,
 - il ne contient aucun sommet correspondant à au plus 2 arêtes du bord,
 - il ne contient pas d'auto-intersection.

Exemples
non
2-manifold



Propriétés : formule d'Euler

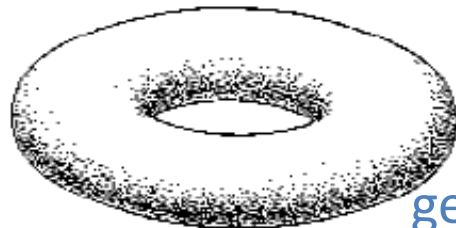
- La formule fait le lien entre le nombre d'entité de chaque groupe dans un maillage:

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

- S : nombre de sommets
- A : nombre d'arêtes
- F : nombre de faces
- C : nombre de composantes connexes
- G : genre du maillage : nombre de « trous fermé »



genre 0



genre 1



genre 2

- T : nombre de trou

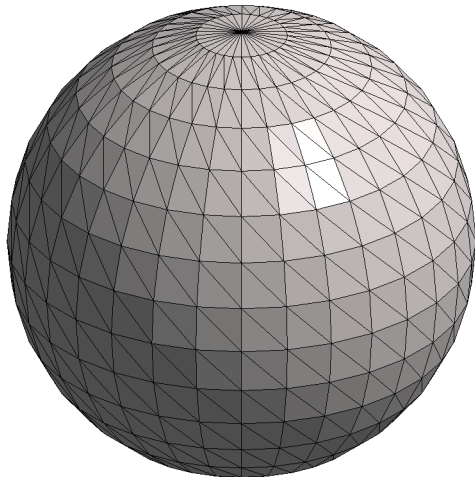
Propriétés : formule d'Euler

Exemples formule d'Euler : $S - A + F = 2C - 2G + T$

- Sphère :

- $C = 1, T=0$ et $G = 0$

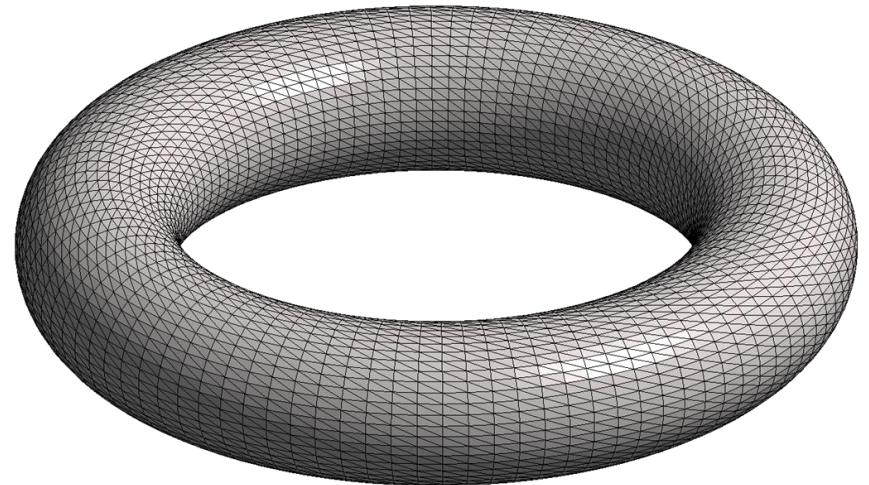
- $S - A + F = 2$



- Tore :

- $C = 1, T=0$ et $G = 1$

- $S - A + F = 0$



Propriétés : formule d'Euler

Exemples formule d'Euler : $S - A + F = 2C - 2G + T$

➤ $C = 1, T=0$ et $G = 0$

➤ $S - A + F = 2$

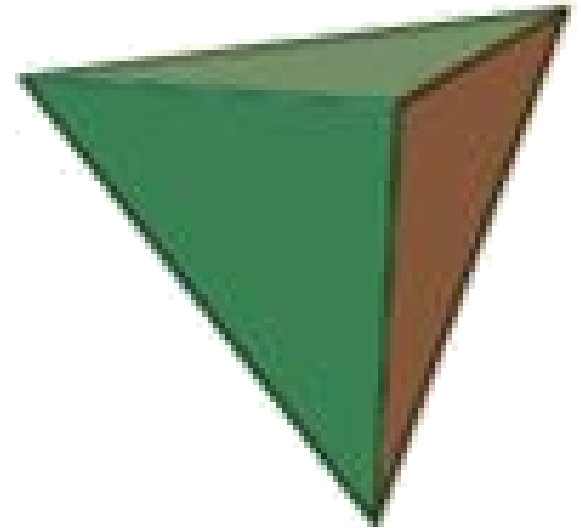
Propriétés : formule d'Euler

Exemples formule d'Euler : $S - A + F = 2C - 2G + T$

➤ $C = 1, T=0$ et $G = 0$

➤ $S - A + F = 2$

➤ Tétraèdre : $4 - 6 + 4 = 2$



Propriétés : formule d'Euler

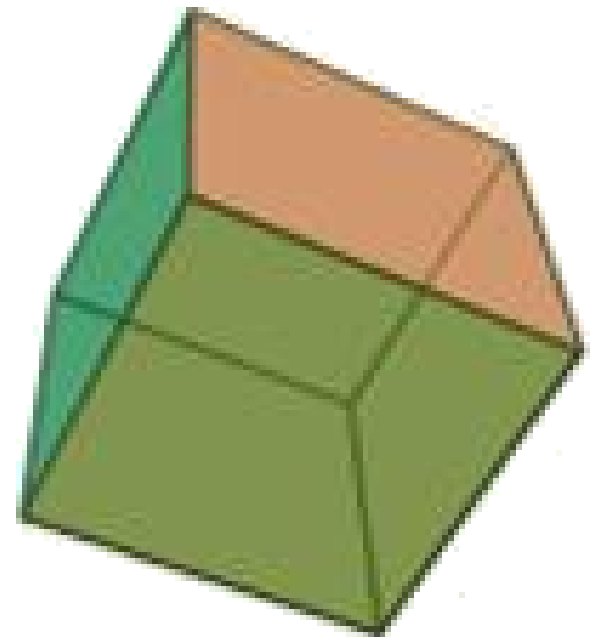
Exemples formule d'Euler : $S - A + F = 2C - 2G + T$

➤ $C = 1, T=0$ et $G = 0$

➤ $S - A + F = 2$

➤ Tétraèdre : $4 - 6 + 4 = 2$

➤ Hexaèdre : $8 - 12 + 6 = 2$



Propriétés : formule d'Euler

Exemples formule d'Euler : $S - A + F = 2C - 2G + T$

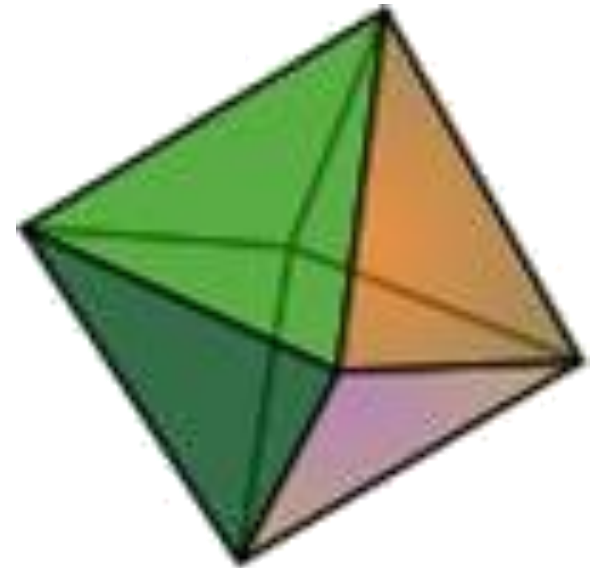
➤ $C = 1, T=0$ et $G = 0$

➤ $S - A + F = 2$

➤ Tétraèdre : $4 - 6 + 4 = 2$

➤ Hexaèdre : $8 - 12 + 6 = 2$

➤ Octaèdre : $6 - 12 + 8 = 2$



Propriétés : formule d'Euler

Exemples formule d'Euler : $S - A + F = 2C - 2G + T$

➤ $C = 1$, $T=0$ et $G = 0$

➤ $S - A + F = 2$

➤ Tétraèdre : $4 - 6 + 4 = 2$

➤ Hexaèdre : $8 - 12 + 6 = 2$

➤ Octaèdre : $6 - 12 + 8 = 2$

➤ Dodécaèdre régulier : $20 - 30 + 12 = 2$



Propriétés : formule d'Euler

Exemples formule d'Euler : $S - A + F = 2C - 2G + T$

➤ $C = 1, T=0$ et $G = 0$

➤ $S - A + F = 2$

➤ Tétraèdre : $4 - 6 + 4 = 2$

➤ Hexaèdre : $8 - 12 + 6 = 2$

➤ Octaèdre : $6 - 12 + 8 = 2$

➤ Dodécaèdre régulier : $20 - 30 + 12 = 2$

➤ Icosaèdre : $12 - 30 + 20 = 2$

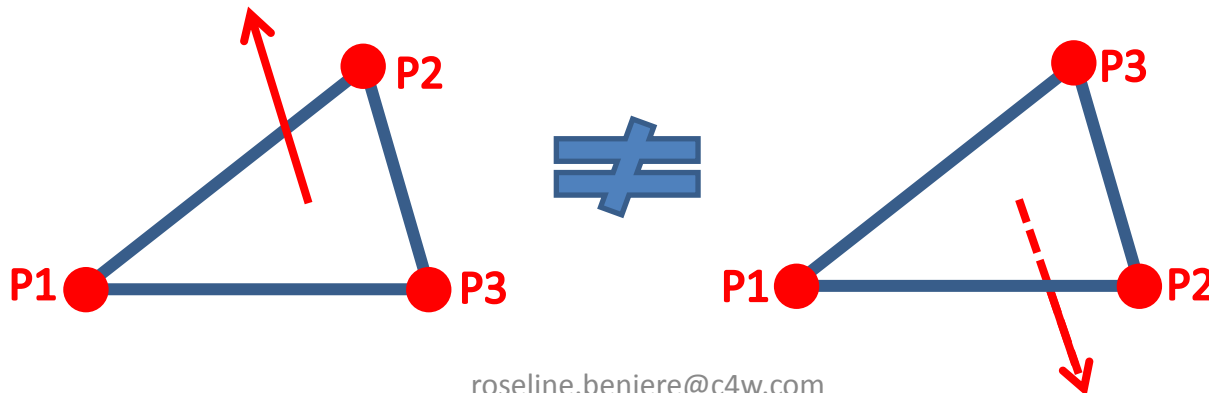


Propriétés : normales

- On peut définir une normale par face :
 - elle permet de définir l'orientation de la face
 - elle est égale au produit vectoriel des deux premières arêtes

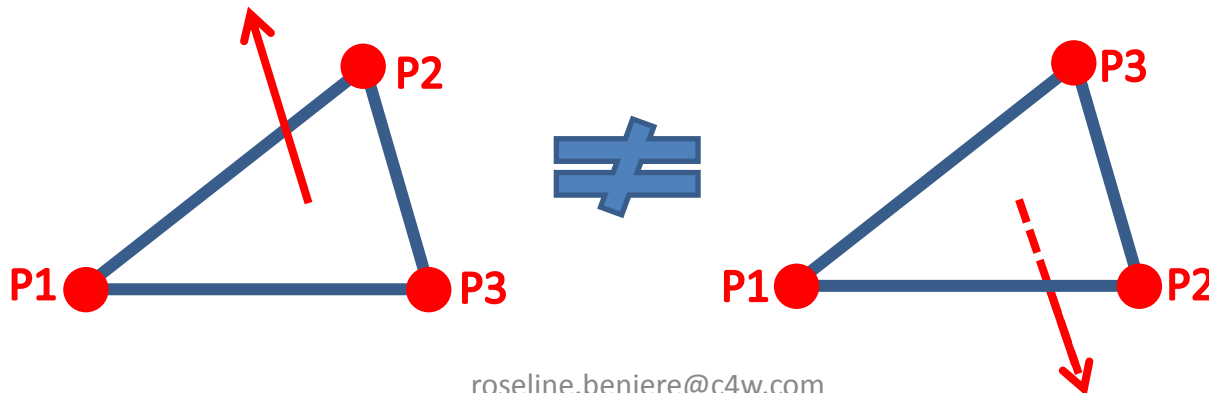
Propriétés : normales

- On peut définir une normale par face :
 - elle permet de définir l'orientation de la face
 - elle est égale au produit vectoriel des deux premières arêtes
 - l'ordre des sommets dans une face est donc important



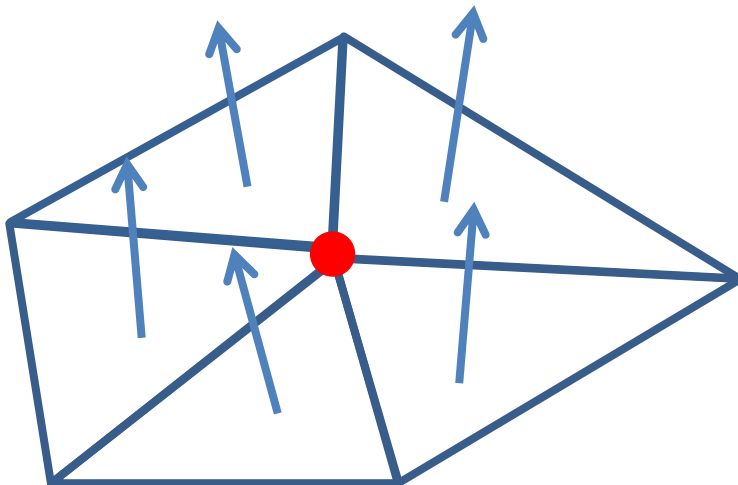
Propriétés : normales

- On peut définir une normale par face :
 - elle permet de définir l'orientation de la face
 - elle est égale au produit vectoriel des deux premières arêtes
 - l'ordre des sommets dans une face est donc important
 - elle est utilisée pour définir l'extérieur ou l'intérieur ou pour l'éclairage à l'affichage.



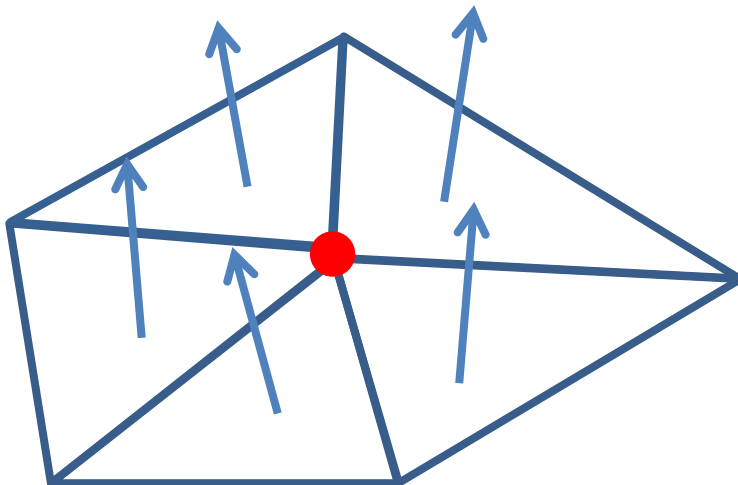
Propriétés : normales

- On peut définir une normale par sommet :
 - à partir des normales aux faces,



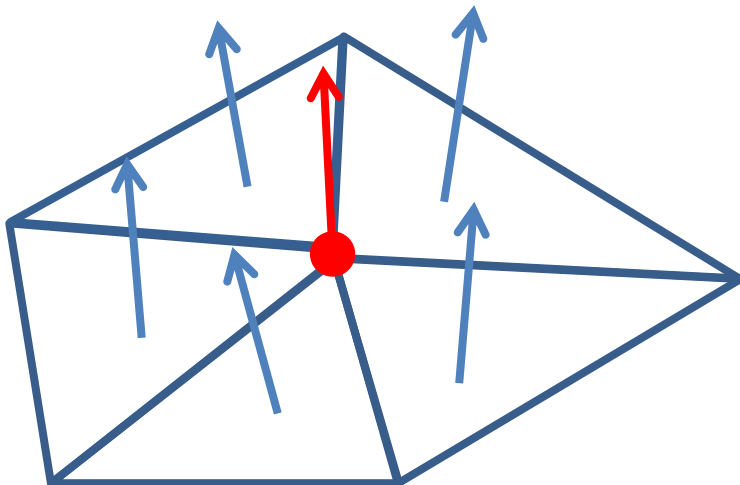
Propriétés : normales

- On peut définir une normale par sommet :
 - à partir des normales aux faces,
 - normale au sommet = moyenne des normales des faces contenant de sommets,



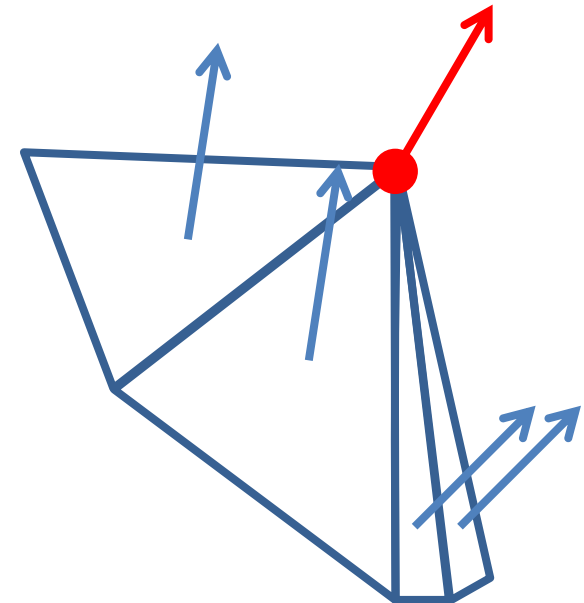
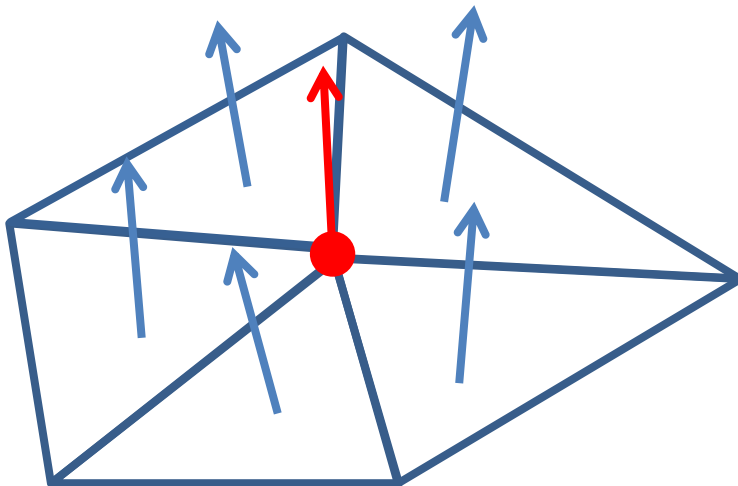
Propriétés : normales

- On peut définir une normale par sommet :
 - à partir des normales aux faces,
 - normale au sommet = moyenne des normales des faces contenant de sommets,



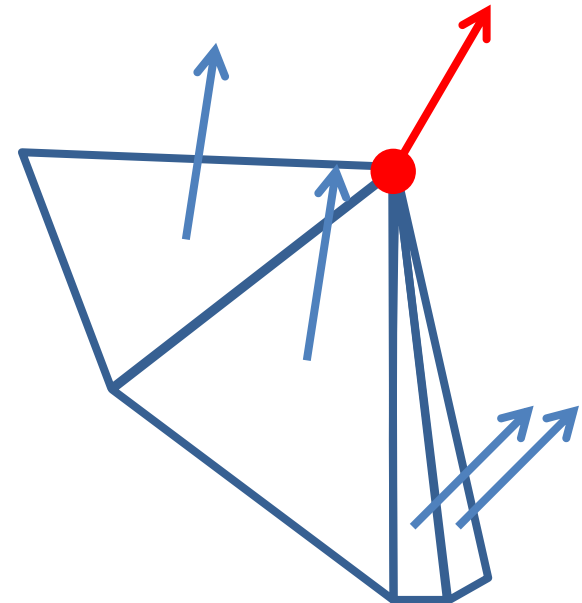
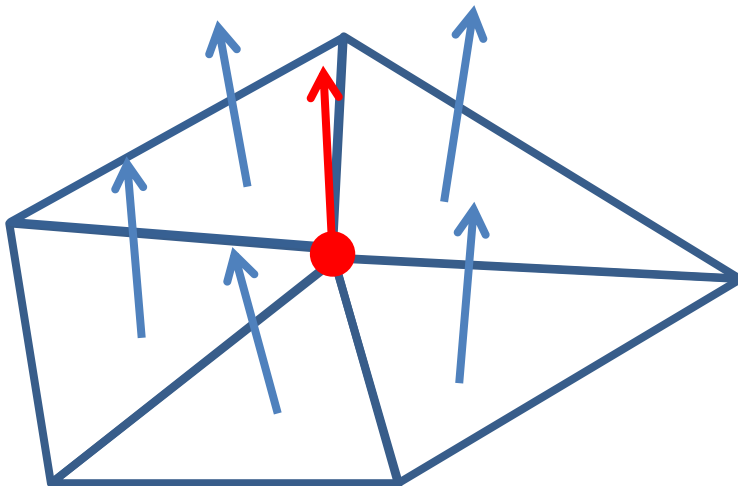
Propriétés : normales

- On peut définir une normale par sommet :
 - à partir des normales aux faces,
 - normale au sommet = moyenne des normales des faces contenant de sommets,



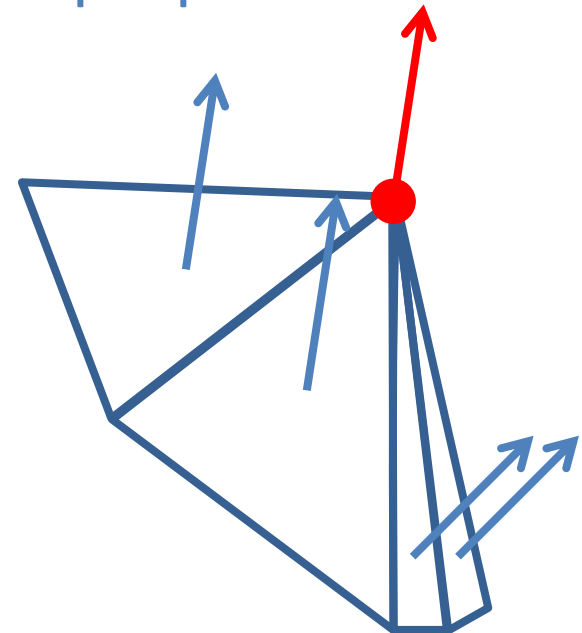
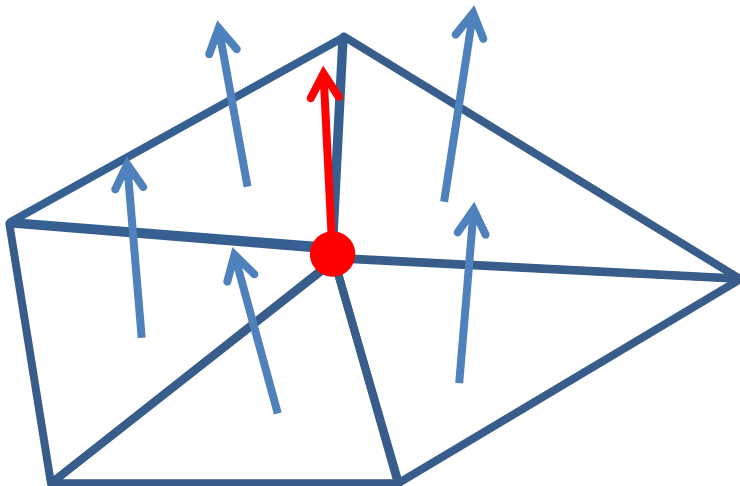
Propriétés : normales

- On peut définir une normale par sommet :
 - à partir des normales aux faces,
 - normale au sommet = moyenne des normales des faces contenant de sommets,
 - mieux si on pondère par une propriété du triangle (ex : aire).



Propriétés : normales

- On peut définir une normale par sommet :
 - à partir des normales aux faces,
 - normale au sommet = moyenne des normales des faces contenant de sommets,
 - mieux si on pondère par une propriété du triangle (ex : aire).



Structure de données

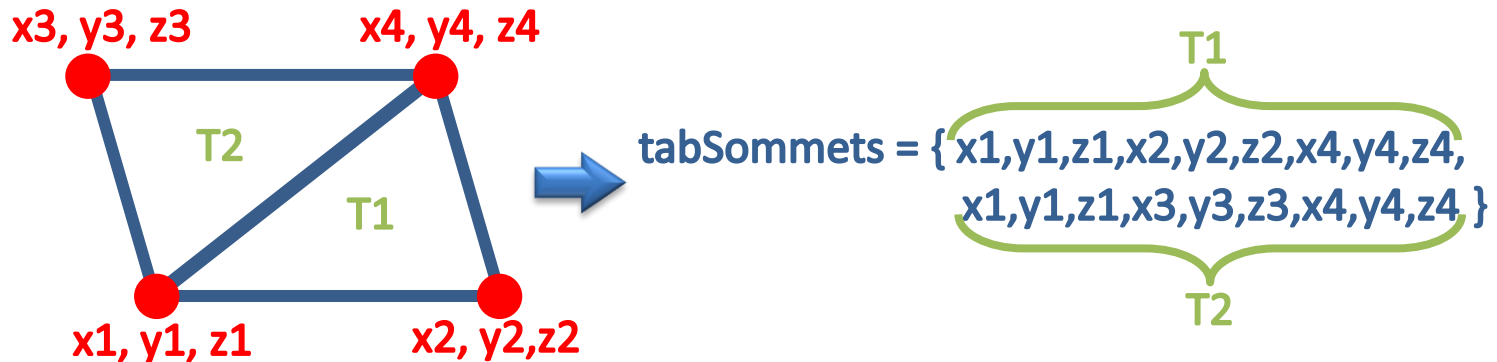
- Ce qu'il y a à stocker :
 - les entités : sommets, arêtes, faces;
 - les normales (par sommet ou face);
 - les couleurs (par sommet ou face), ou les textures ...
 - ...

Structure de données

- **Ce qu'il y a à stocker :**
 - les entités : sommets, arêtes, faces;
 - les normales (par sommet ou face);
 - les couleurs (par sommet ou face), ou les textures ...
 - ...
- **Pour stocker un maillage il faut choisir entre :**
 - minimiser la taille mémoire,
 - répéter le moins possible les coordonnées des points, ...
 - faciliter le parcours dans le maillage,
 - pour passer d'un sommet à l'autre, ...
 - permettre d'extraire les informations de topologie.
 - pour connaître les sommets liés à un autre sommet , les arêtes liées à un sommet, ...

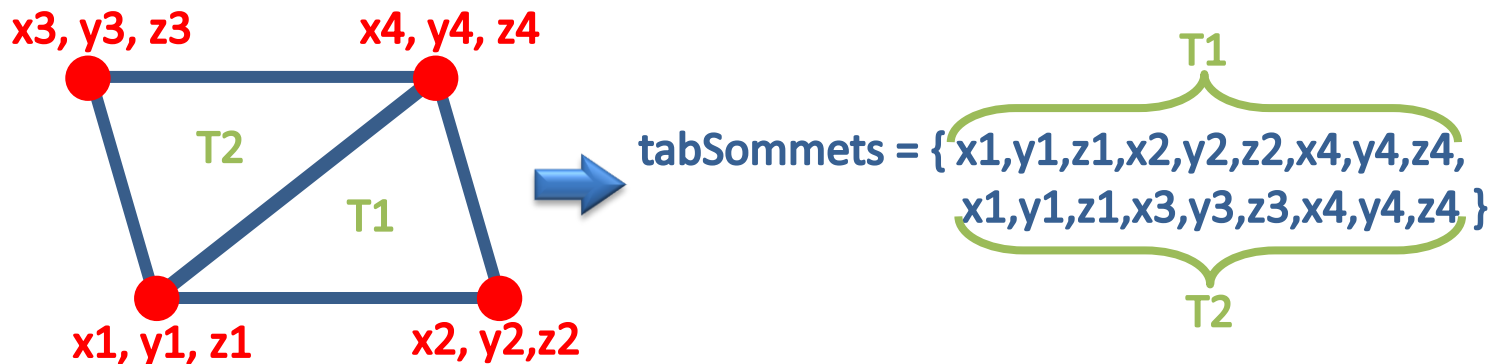
Structure de données

- Approche naïve :
 - maillage représenté par un unique tableau de sommet ➔ maillage *non indexé*,
 - les coordonnées des sommets sont répétées autant de fois qu'ils y a de faces qui les contiennent.



Structure de données

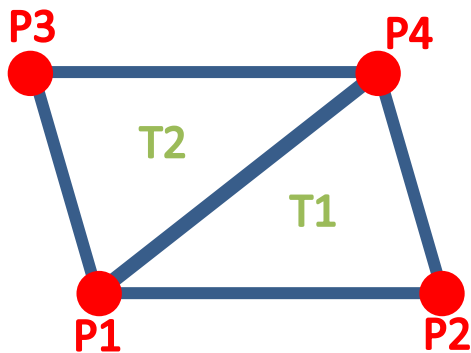
- Approche naïve : ➡ **Prend beaucoup de place**
 - maillage représenté par un unique tableau de sommet ➡ maillage *non indexé*,
 - les coordonnées des sommets sont répétées autant de fois qu'ils y a de faces qui les contiennent.



Structure de données

- Approche classique :

- maillage représenté un ensemble de tableau : un pour les sommets, un pour les faces, un pour les couleurs ... ➔ maillage *indexé*,
- les coordonnées des sommets ne sont plus répétées.



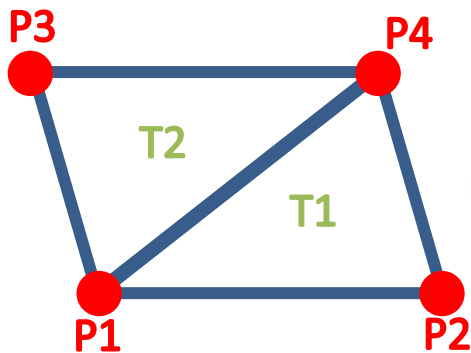
➔

```
tabSommets = { x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4 }
tabTriangles = { 1, 2, 4, 1, 3, 4 }
tabColors = { r1,g1,b1,r2,g2,b2,r3,g3,zb3,r4,g4,b4 }
```

Couleur P1

Structure de données

- Approche classique : ➡ **Pas pratique pour la topologie**
 - maillage représenté un ensemble de tableau : un pour les sommets, un pour les faces, un pour les couleurs ... ➡ maillage *indexé*,
 - les coordonnées des sommets ne sont plus répétées.



➡

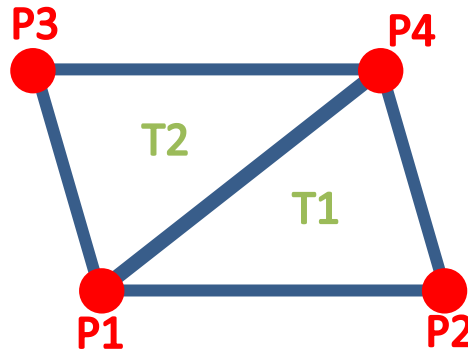
```
tabSommets = { x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4 }
tabTriangles = { 1, 2, 4, 1, 3, 4 }
tabColors = { r1,g1,b1,r2,g2,b2,r3,g3,zb3,r4,g4,b4 }
```

Couleur P1

Structure de données

- Approche *Strip* ou *Fan*:

- STRIP : maillage représenté par une bande,
- FAN : maillage défini autour d'un premier sommet.



STRIP

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

tabTriangles = { $\underbrace{3, 1, 4}_{T2}, 2 \}_{T1}$

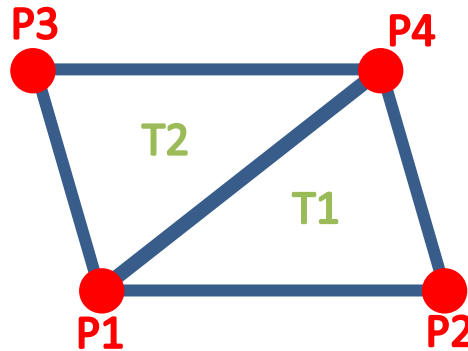
FAN

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

tabTriangles = { $\underbrace{4, 3, 1}_{T2}, 2 \}_{T1}$

Structure de données

- Approche *Strip* ou *Fan*:  Pas adapté à tous les maillages et prob topologie
 - STRIP : maillage représenté par une bande,
 - FAN : maillage défini autour d'un premier sommet.

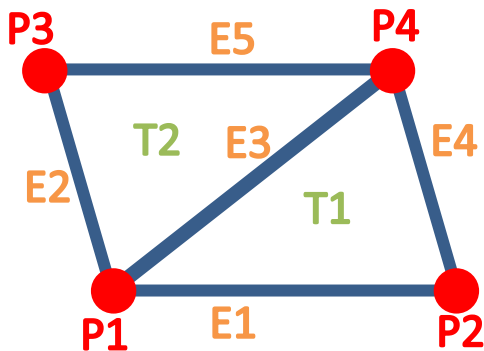


STRIP
tabSommets = { x1,y1,z1,x2,y2,z2,
x3,y3,z3,x4,y4,z4 }
tabTriangles = { $\underbrace{3, 1, 4}_{T2}, 2 \}_{T1}$

FAN
tabSommets = { x1,y1,z1,x2,y2,z2,
x3,y3,z3,x4,y4,z4 }
tabTriangles = { $\underbrace{4, 3, 1}_{T1}, 2 \}_{T2}$

Structure de données

- Approche par arête :
 - maillage représenté par
 - des sommets définis par 3 coordonnées,
 - des arêtes définies par 2 sommets et deux faces,
 - des faces définies par 3 arêtes.




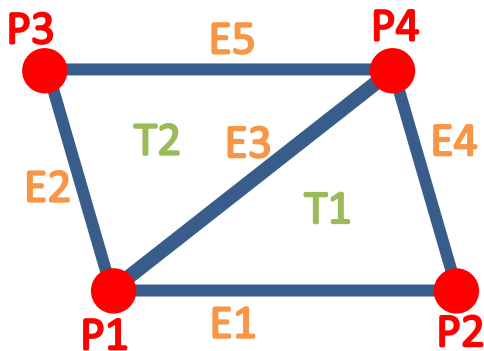
tabSommets = { $\underbrace{x_1, y_1, z_1}_{P1}, x_2, y_2, z_2, x_3, y_3, z_3, x_4, y_4, z_4$ }

tabEdges = { $\underbrace{P1, P2, T1, \emptyset}_{E1}, P1, P3, T2, \emptyset, P1, P4, T1, T2, \dots$ }

tabElements = { $\underbrace{E1, E3, E4}_{T1}, E2, E3, E5$ }

Structure de données

- Approche par arête :  **Prend beaucoup de place mais topologie simple**
 - maillage représenté par
 - des sommets définis par 3 coordonnées,
 - des arêtes définies par 2 sommets et deux faces,
 - des faces définies par 3 arêtes.



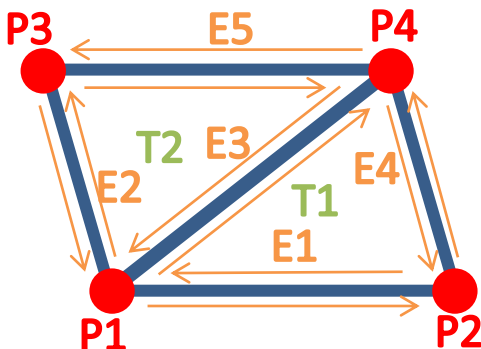
tabSommets = { $\underbrace{x_1, y_1, z_1}_{P1}, x_2, y_2, z_2, x_3, y_3, z_3, x_4, y_4, z_4$ }

tabEdges = { $\underbrace{P1, P2, T1, \emptyset}_{E1}, P1, P3, T2, \emptyset, P1, P4, T1, T2, \dots$ }

tabElements = { $\underbrace{E1, E3, E4}_{T1}, E2, E3, E5$ }

Structure de données

- Approche par demi - arête :
 - une arête donne deux demi-arête définie par
 - la seconde demi-arête de l'arête,
 - l'arête suivante dans la face,
 - la face que borde l'arête
 - le sommet extrémité.



$\text{tabSommets} = \{ \underbrace{x1, y1, z1}_{P1}, x2, y2, z2, x3, y3, z3, x4, y4, z4 \}$

$\text{tabEdges} = \{ \underbrace{E1', E3', T1, P2}_{E1}, \underbrace{E2', E5', T2, P3}_{E2}, E3', E2, T2, P1, \dots \}$

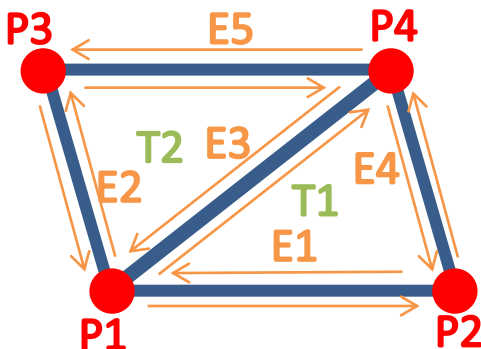
$\text{tabElement} = \{E1, E2\}$

Structure de données

- Approche par demi - arête :

- une arête donne deux demi-arête définie par
 - la seconde demi-arête de l'arête,
 - l'arête suivante dans la face,
 - la face que borde l'arête
 - le sommet extrémité.

-prend de la place mais
-topologie/parcours simple
-suppression/ajout simple



tabSommets = { $\underbrace{x1, y1, z1}_{P1}, x2, y2, z2, x3, y3, z3, x4, y4, z4$ }

tabEdges = { $\underbrace{E1', E3', T1, P2}_{E1}, \underbrace{E2', E5', T2, P3}_{E2}, E3', E2, T2, P1, \dots$ }

tabElement = {E1, E2}

Structure de données

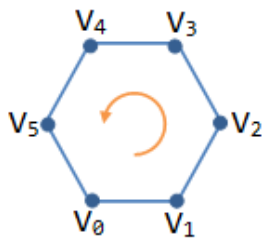
- **Formats de fichier :**
 - **Soit indexé**
 - OFF
 - OBJ
 - **Soit non indexé**
 - STL

Visualisation OpenGL

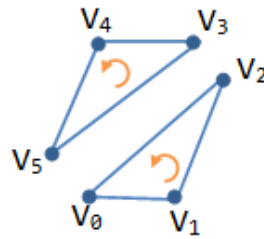
- Rendu optimisé par VA ou VBO :
 - on ne donne plus la liste de sommets les uns après les autres, mais des tableaux.
 - **VA** = ``Vertex Array'', buffers stockés sur la RAM .
 - **VBO** = ``Vertex Buffer Object'' buffers stockés sur la carte graphique ➡ évite de renvoyer des données à la carte à chaque rafraichissement de la vue.
➡ les VBO ne sont pas supportés sur toutes les cartes graphiques

Visualisation OpenGL

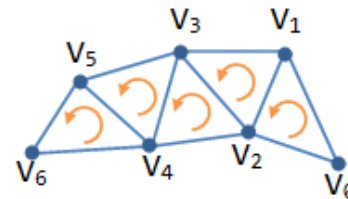
- Type de face :
 - triangles : *GL_TRIANGLES*
 - quadrangles : *GL_QUADS*
 - polygones : *GL_POLYGON*
 - ...



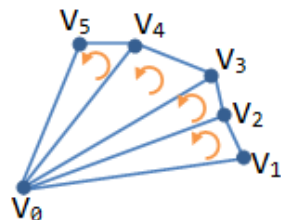
GL_POLYGON



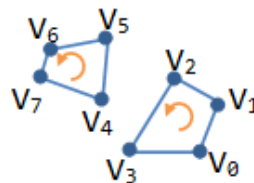
GL_TRIANGLES



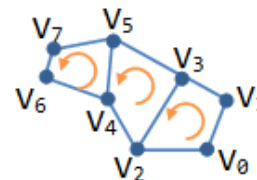
GL_TRIANGLE_STRIP



GL_TRIANGLE_FAN



GL_QUADS



GL_QUAD_STRIP

Visualisation OpenGL

- on active les tableaux
 - tableau de points
 - ➔ glEnableClientState (GL_VERTEX_ARRAY);
 - tableau de normales :
 - ➔ glEnableClientState (GL_NORMAL_ARRAY);
 - ...

Visualisation OpenGL

- **on active les tableaux**

- tableau de points

-  glEnableClientState (GL_VERTEX_ARRAY);

- tableau de normales :

-  glEnableClientState (GL_NORMAL_ARRAY);

- ...

- **on lie les tableaux**

- tableau de points

-  glVertexPointer (3, GL_FLOAT, 0, TabVertices);

- tableau de normales

-  glNormalPointer (GL_FLOAT, 0, TabNorm);

- ...

Visualisation OpenGL

- **on active les tableaux**

- tableau de points

- ➔ `glEnableClientState (GL_VERTEX_ARRAY);`

- tableau de normales :

- ➔ `glEnableClientState (GL_NORMAL_ARRAY);`

- ...

- **on lie les tableaux**

- tableau de points

- ➔ `glVertexPointer (3, GL_FLOAT, 0, TabVertices);`

- tableau de normales

- ➔ `glNormalPointer (GL_FLOAT, 0, TabNorm);`

- ...

- **on trace le maillage**

- ➔ `glDrawElements (GL_TRIANGLES, nb_index, GL_UNSIGNED_INT, TabElement);`

Visualisation OpenGL

- on active les tableaux

- tableau de points

- ➔ `glEnableClientState (GL_VERTEX_ARRAY);`

- tableau de normales :

- ➔ `glEnableClientState (GL_NORMAL_ARRAY);`

- ...

- on lie les tableaux

- tableau de points

- ➔ `glVertexPointer (3, GL_FLOAT, 0, TabVertices);`

- tableau de normales

- ➔ `glNormalPointer (GL_FLOAT, 0, TabNorm);`

- ...

- on trace le maillage

- ➔ `glDrawElements (GL_TRIANGLES, nb_index, GL_UNSIGNED_INT, TabElement);`

- on désactive les tableaux

- ...


- tableau de normales :

- ➔ `glDisableClientState (GL_NORMAL_ARRAY); //1 par glEnableClientState`

- tableau de sommets :

- ➔ `glDisableClientState (GL_VERTEX_ARRAY);`

Visualisation OpenGL

- Gérer l'affichage en VBO : (utiliser la librairie glew)
- on demande des pointeurs pour les tableaux :
 - commencer par `glewInit();`
 - `GLuint TabIdentBuffer[nbIdentBuffer];`
 -  ➤ `glGenBuffers(nbIdentBuffer);`

Visualisation OpenGL

- Gérer l'affichage en VBO : (utiliser la librairie glew)
- on demande des pointeurs pour les tableaux :
 - commencer par `glewInit()`;
 - `GLuint TabIdentBuffer[nbIdentBuffer]`;
 - ➔ `glGenBuffers(nbIdentBuffer)`;
- on envoie les tableaux à la carte graphique :
 - `glBindBuffer(GL_ARRAY_BUFFER, TabIdentBuffer[identPoint])`;
 - ➔ `glBufferData(GL_ARRAY_BUFFER, tailleTab*sizeof(float), TabPoint, GL_STATIC_DRAW)`;
 - `glBindBuffer(GL_ARRAY_BUFFER, 0)`;

Visualisation OpenGL

- Gérer l'affichage en VBO : (utiliser la librairie glew)
- on demande des pointeurs pour les tableaux :
 - commencer par `glewInit()`;
 - `GLuint TabIdentBuffer[nbIdentBuffer]`;
 - ➔ `glGenBuffers(nbIdentBuffer)`;
- on envoie les tableaux à la carte graphique :
 - `glBindBuffer(GL_ARRAY_BUFFER, TabIdentBuffer[identPoint])`;
 - ➔ `glBufferData(GL_ARRAY_BUFFER, tailleTab*sizeof(float), TabPoint, GL_STATIC_DRAW)`;
 - `glBindBuffer(GL_ARRAY_BUFFER, 0)`;
- on peut le récupérer :
 - `glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, TabIdentBuffer[identElem])`;
 - ➔ `TabElem = glMapBuffer(GL_ELEMENT_ARRAY_BUFFER, GL_READ_ONLY)`;
 - `glUnmapBuffer(GL_ELEMENT_ARRAY_BUFFER)`;

Visualisation OpenGL

- Gérer l'affichage en VBO : (utiliser la librairie glew)
- on demande des pointeurs pour les tableaux :
 - commencer par `glewInit()`;
 - `GLuint TabIdentBuffer[nbIdentBuffer]`;
 - ➔ `glGenBuffers(nbIdentBuffer)`;
- on envoie les tableaux à la carte graphique :
 - `glBindBuffer(GL_ARRAY_BUFFER, TabIdentBuffer[identPoint])`;
 - ➔ `glBufferData(GL_ARRAY_BUFFER, tailleTab*sizeof(float), TabPoint, GL_STATIC_DRAW)`;
 - `glBindBuffer(GL_ARRAY_BUFFER, 0)`;
- on peut le récupérer :
 - `glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, TabIdentBuffer[identElem])`;
 - ➔ `TabElem = glMapBuffer(GL_ELEMENT_ARRAY_BUFFER, GL_READ_ONLY)`;
 - `glUnmapBuffer(GL_ELEMENT_ARRAY_BUFFER)`;
- on les dessine :
 - `glBindBuffer(GL_ARRAY_BUFFER, TabIdentBuffer[identColor])`;
 - ➔ `glColorPointer(4, GL_FLOAT, 0, BUFFER_OFFSET(0))`;

Visualisation OpenGL

- Gérer l'affichage en VBO : (utiliser la librairie glew)
- on demande des pointeurs pour les tableaux :
 - commencer par `glewInit()`;
 - `GLuint TabIdentBuffer[nbIdentBuffer]`;
 - ➔ `glGenBuffers(nbIdentBuffer)`;
- on envoie les tableaux à la carte graphique :
 - `glBindBuffer(GL_ARRAY_BUFFER, TabIdentBuffer[identPoint])`;
 - ➔ `glBufferData(GL_ARRAY_BUFFER, tailleTab*sizeof(float), TabPoint, GL_STATIC_DRAW)`;
 - `glBindBuffer(GL_ARRAY_BUFFER, 0)`;
- on peut le récupérer :
 - `glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, TabIdentBuffer[identElem])`;
 - ➔ `TabElem = glMapBuffer(GL_ELEMENT_ARRAY_BUFFER, GL_READ_ONLY)`;
 - `glUnmapBuffer(GL_ELEMENT_ARRAY_BUFFER)`;
- on les dessine :
 - `glBindBuffer(GL_ARRAY_BUFFER, TabIdentBuffer[identColor])`;
 - ➔ `glColorPointer(4, GL_FLOAT, 0, BUFFER_OFFSET(0))`;
- on les supprime :
 - ➔ `glDeleteBuffers(nbIdentBuffer, TabIdentBuffer)`;

Visualisation OpenGL

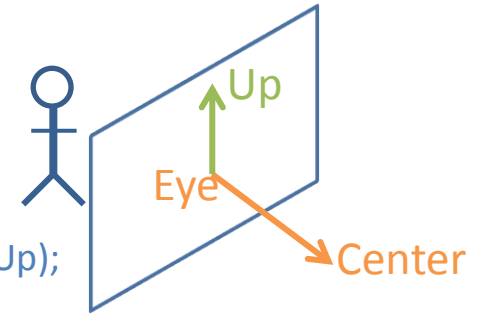
- Placer la scène:

- placer les « plans clipping »

- ➔ `glOrtho(Left, Right, Bottom, Top, Near, Far);`

- placer la camera

- ➔ `gluLookAt(xEye, yEye, zEye, xCenter, yCenter, zCenter, xUp, yUp, zUp);`



Visualisation OpenGL

- **Placer la scène:**

- placer les « plans clipping »

- ➔ `glOrtho(Left, Right, Bottom, Top, Near, Far);`

- placer la camera

- ➔ `gluLookAt(xEye, yEye, zEye, xCenter, yCenter, zCenter, xUp, yUp, zUp);`

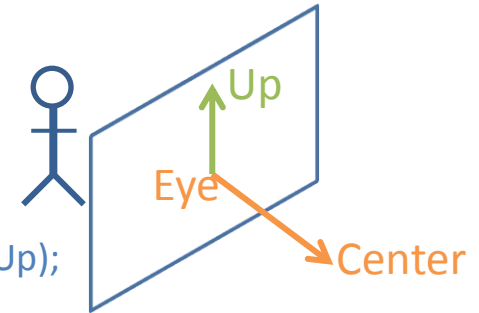
- **Gérer la lumière:**

- pour l'allumer ou l'éteindre

- ➔ `glEnable(GL_LIGHTING); / glDisable(GL_LIGHTING);`

- pour paramétrer une des 8 lumières possibles :

- ➔ `glLightfv(GL_LIGHT0, GL_POSITION, TabPosition);`



Visualisation OpenGL

- **Placer la scène:**

- placer les « plans clipping »

➔ `glOrtho(Left, Right, Bottom, Top, Near, Far);`

- placer la camera

➔ `gluLookAt(xEye, yEye, zEye, xCenter, yCenter, zCenter, xUp, yUp, zUp);`

- **Gérer la lumière:**

- pour l'allumer ou l'éteindre

➔ `glEnable(GL_LIGHTING); / glDisable(GL_LIGHTING);`

- pour paramétrer une des 8 lumières possibles :

➔ `glLightfv(GL_LIGHT0, GL_POSITION, TabPosition);`

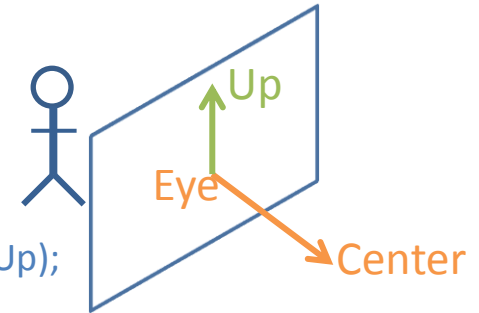
- **Paramètres d'affichage :**

- Type de représentation (face plane ou juste les arêtes)

➔ `glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);` ou `GL_LINE`

- Style de représentation (facette ou lissé) :

➔ `glShapeMode(GL_FLAT);` ou `GL_SMOOTH`



Visualisation OpenGL

- **Placer la scène:**

- placer les « plans clipping »

➔ `glOrtho(Left, Right, Bottom, Top, Near, Far);`

- placer la camera

➔ `gluLookAt(xEye, yEye, zEye, xCenter, yCenter, zCenter, xUp, yUp, zUp);`

- **Gérer la lumière:**

- pour l'allumer ou l'éteindre

➔ `glEnable(GL_LIGHTING); / glDisable(GL_LIGHTING);`

- pour paramétrer une des 8 lumières possibles :

➔ `glLightfv(GL_LIGHT0, GL_POSITION, TabPosition);`

- **Paramètres d'affichage :**

- Type de représentation (face plane ou juste les arêtes)

➔ `glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);` ou `GL_LINE`

- Style de représentation (facette ou lissé) :

➔ `glShapeMode(GL_FLAT);` ou `GL_SMOOTH`

- Affichage des « back faces » d'une couleur différente :

➔ `glEnable(GL_CULL_FACE)`

`//Dessin des front faces`

`glCullFace(GL_BACK);`

`glColor4fv(ColorFrontFace);`

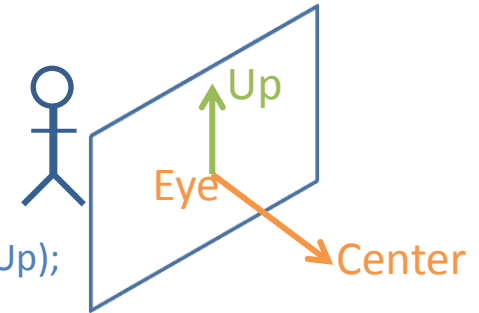
`// Dessiner le maillage`

`//Dessin des back faces`

`glCullFace(GL_FRONT);`

`glColor4fv(ColorBackFace);`

`// Dessiner le maillage`



Conclusion

- **Représentation par maillage :**
 - un ensemble de sommets, un d'arêtes et un de faces,
 - plus les autres propriétés : normales, couleurs ...

Conclusion

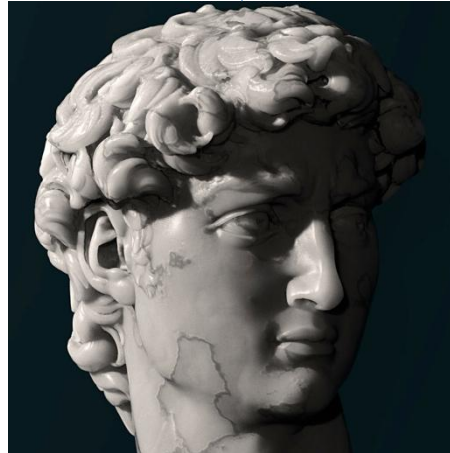
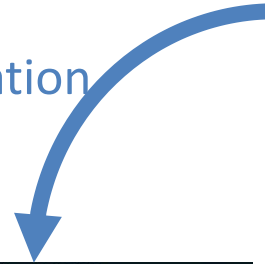
- **Représentation par maillage :**
 - un ensemble de sommets, un d'arêtes et un de faces,
 - plus les autres propriétés : normales, couleurs ...
- **Plusieurs représentations possibles :**
 - les arêtes ou les faces ne sont pas forcément stockées de manière explicite,
 - selon la représentation les liaisons : sommets/faces, sommets/sommets, arêtes/faces ... ne sont pas toujours les même,
 - il faut choisir entre taille en mémoire, parcours dans le maillage et extraction de la topologie.

FIN



13 millions de triangles
<http://www.cs.unc.edu/~walk/models/>

numérisation



2 milliards de polygones
<http://graphics.stanford.edu/projects/mich/>

