Modélisation Géométrique
(des surfaces)

gael.guennebaud@inria.fr

http://www.labri.fr/perso/pbenard/teaching/mondes3d

Programme

1) *Intro + Intro lancer de rayons (GG)*
2) *Modèle d'éclairage local, BRDFs (PB)*
3) *Lancer de rayon++ (PB)*
4) *OpenGL : rasterisation, Z-buffer, pipeline (GG)*
5) *Transformations de l'espace (PB)*
6) *Compléments OpenGL (PB)*
7) *Textures (GG)*
8) **Géométrie : tour d'horizon (GG)**
9) **Géométrie** : courbes paramétriques (Bézier, BSpline) (GG)
10) **Géométrie** : surfaces de subdivision / halfedge (GG)
11) Animation : intro, forward/inverse kinematics (PB)
12) Animation : skinning + déformation (PB)

Modélisation géométrique ?

- **Comment représenter des objects numériques ?**

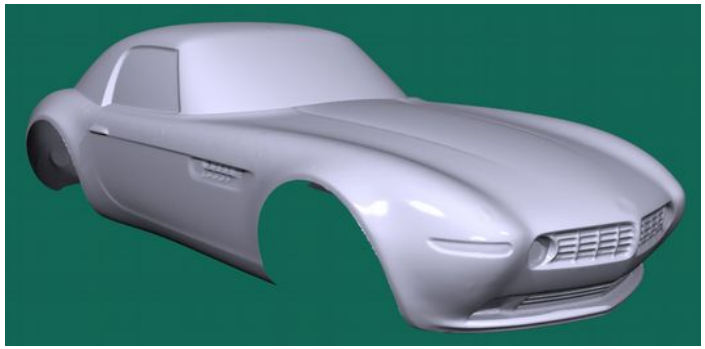
  → des surfaces ?

- **Comment les créer/manipuler ?**

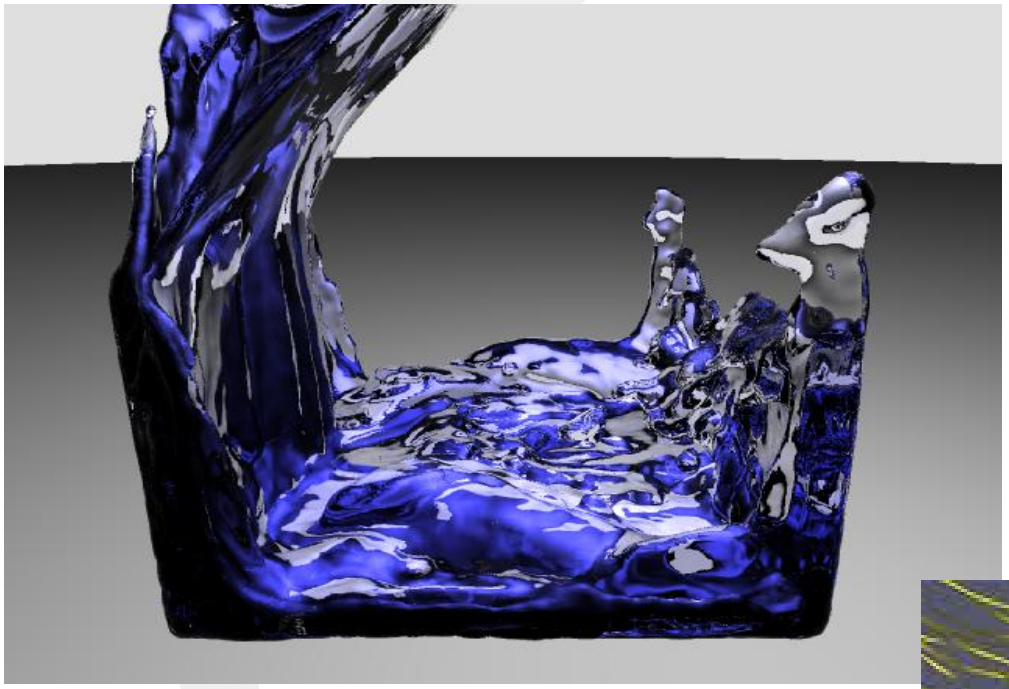
  → traitements (simplification, lissage, déformations, conversions, etc.)

Construction / Design

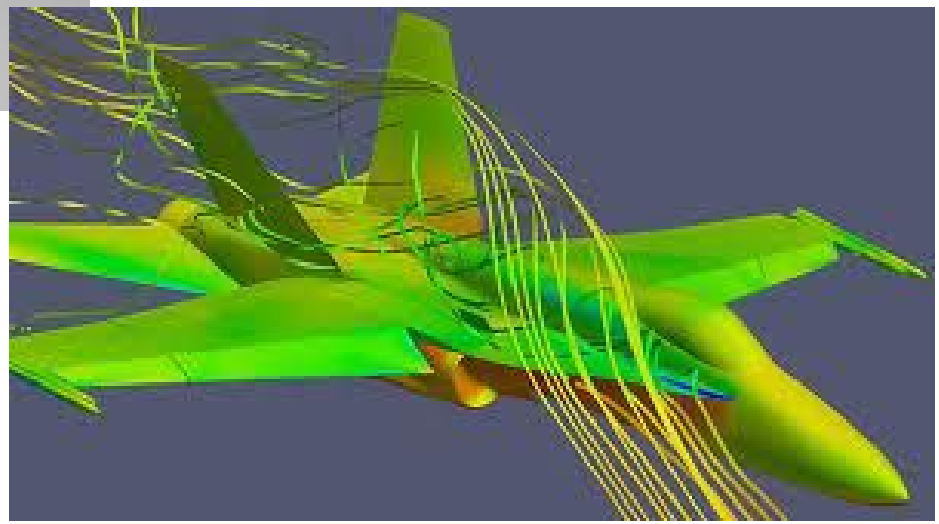
- **Maquette virtuelle**
  - Contraintes de fabrication (continuité, découpage, assemblage, résistance, …)



Simulations

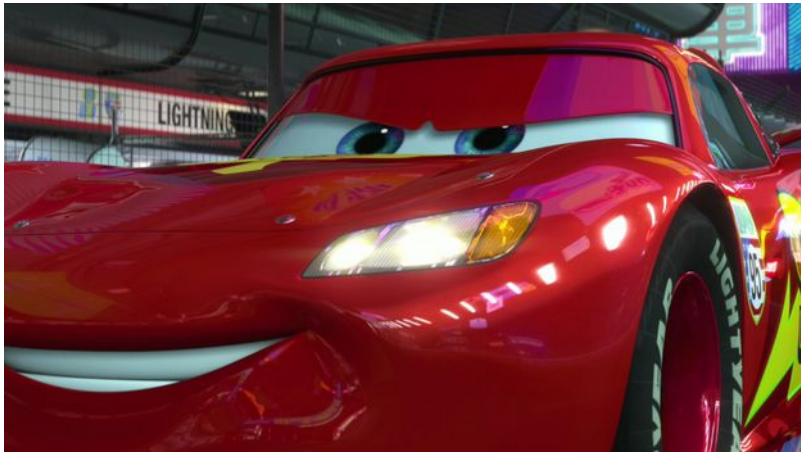


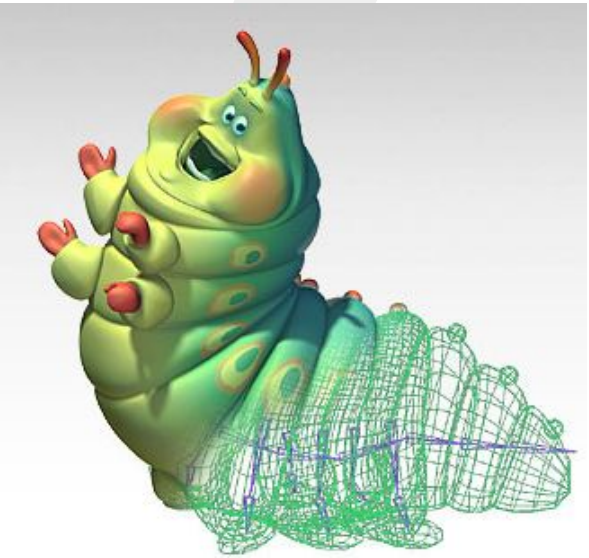
- Modèles en entrées
- Mesures pour les modèles physiques
- Extraction de surfaces



Films d'animation & Jeux vidéo

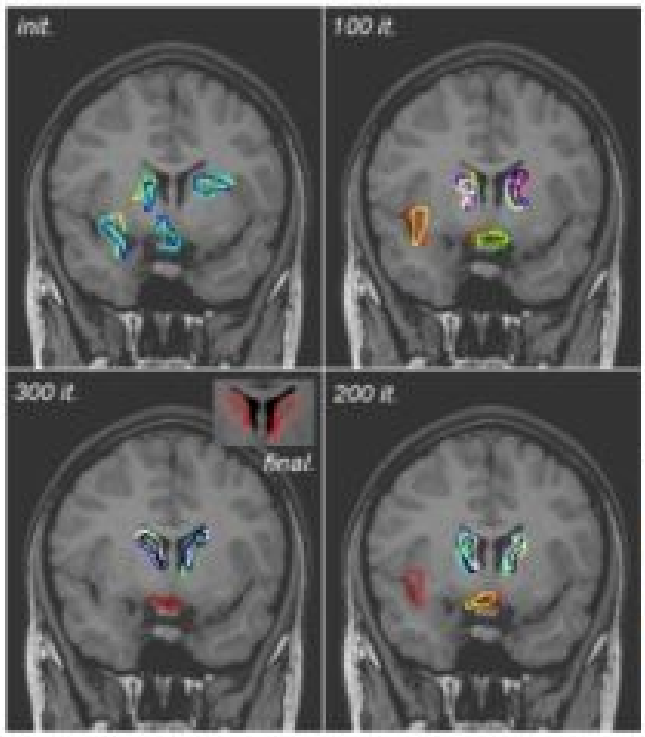
- **Modélisation de forme libre**
  - Outils intuitifs
  - Qualité visuel : détails, continuité
  - Performance de la représentation
    - rapide à afficher, compact

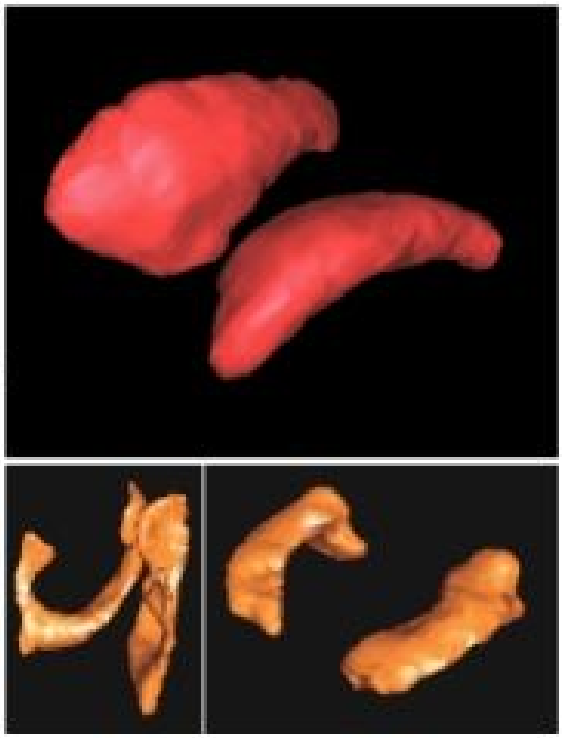




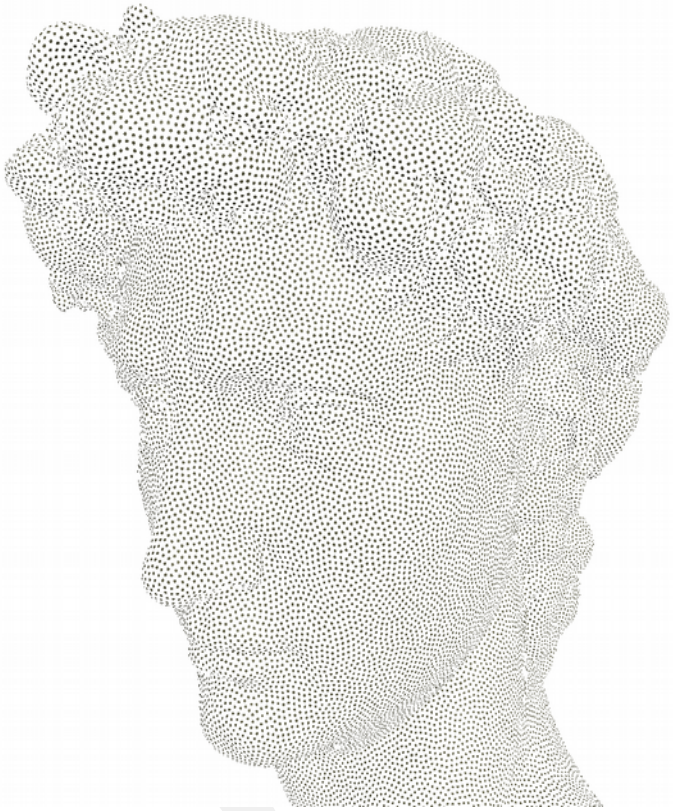
Domaine médical

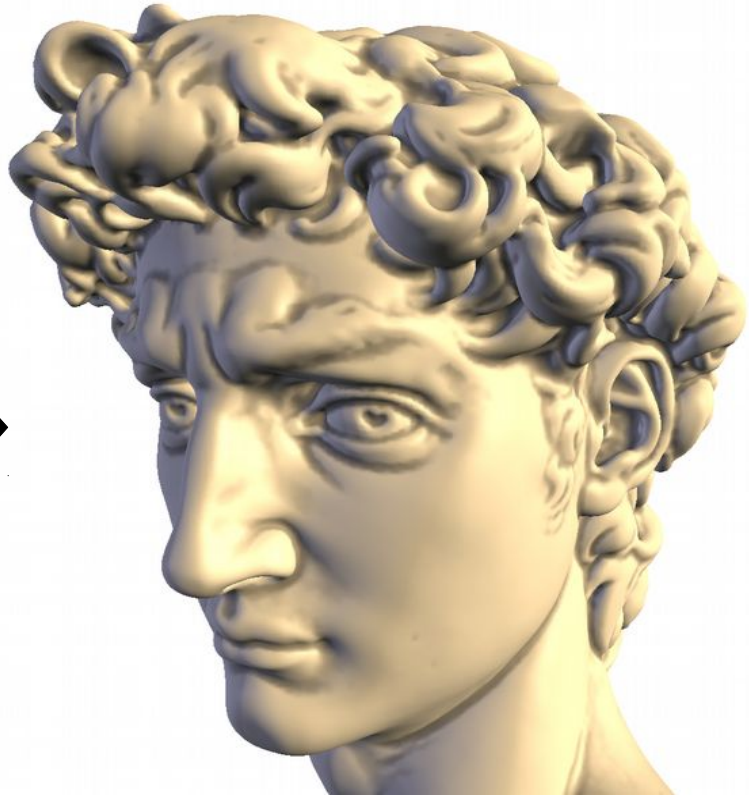
- **Visu de données issues des scanner (surfacique/volumique)**
- **Reconstruction des organes**
- **Simulation des déformations (opérations virtuelle)**





Reconstruction de surface





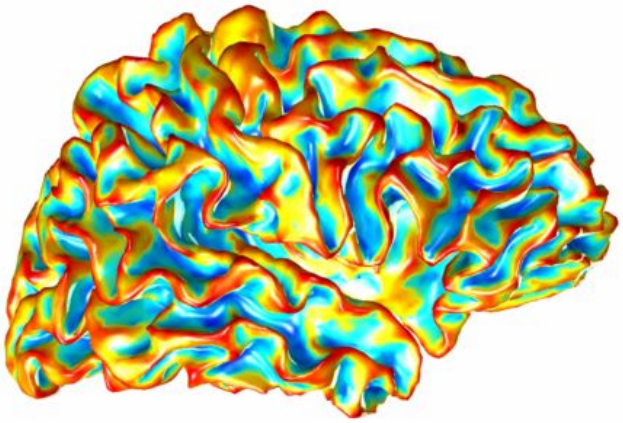
Input: a point cloud

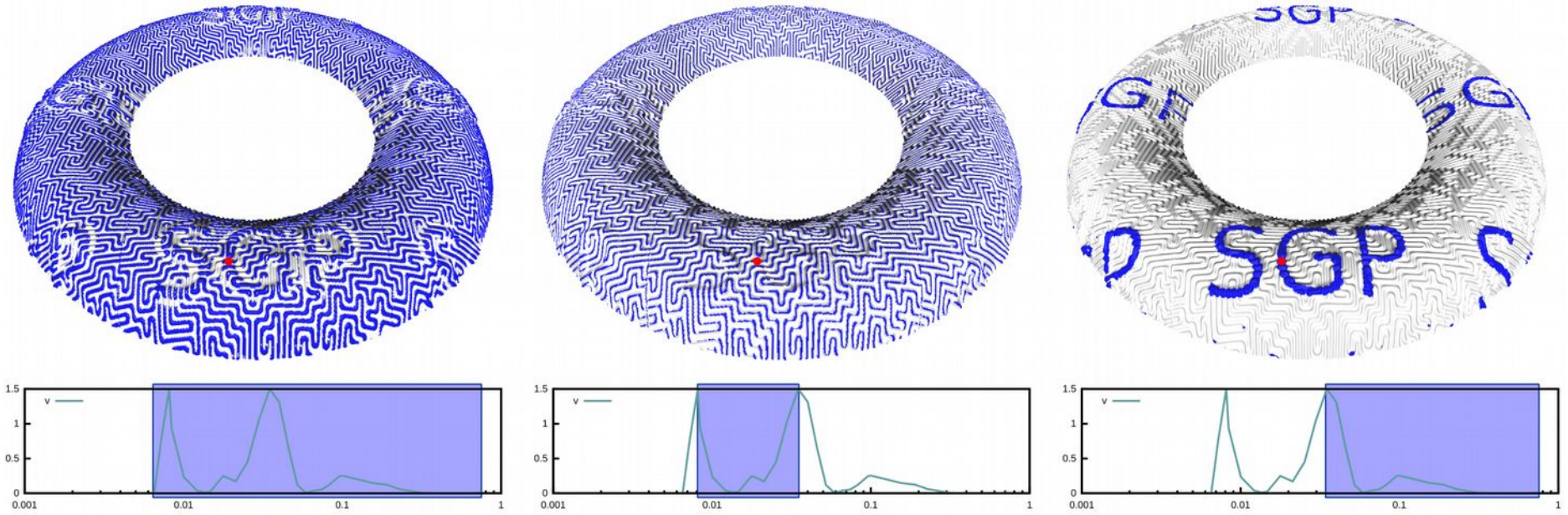
Output: a surface (continuous)

Interpolation versus approximation versus extrapolation

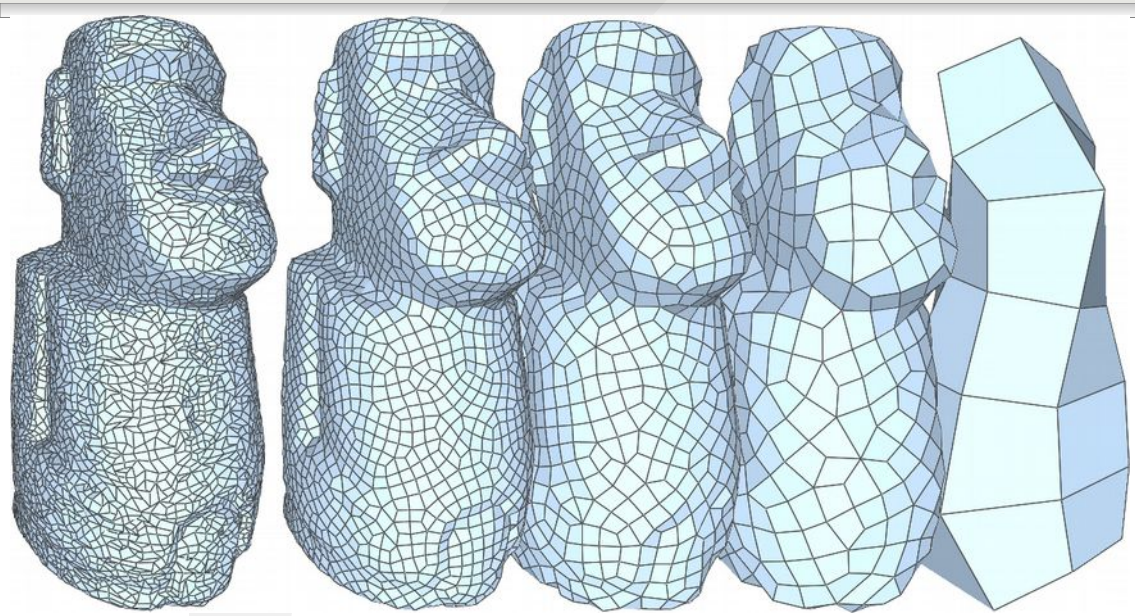
Analyse & mesure

- Mesure de longueur (géodésique) et de **courbures**
- Extraction des « features », débruitage
- Exploiter la courbure dans les simulations
- etc.





Simplification, lissage, remeshing, etc.



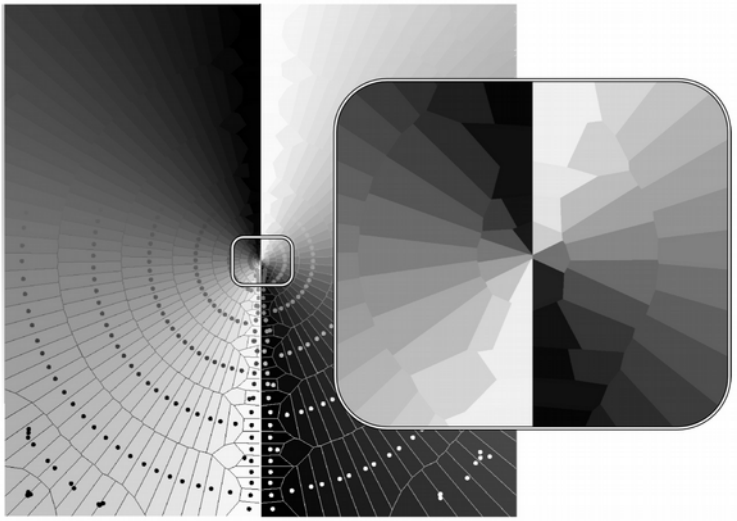
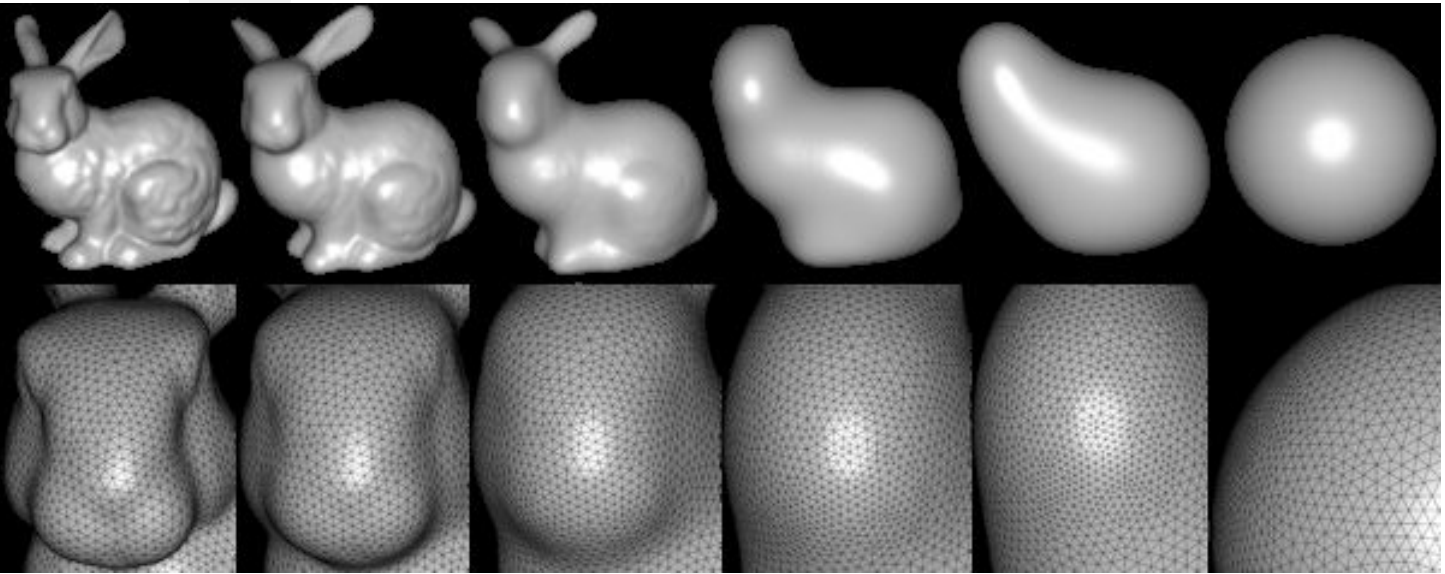


Figure 5: *Approximation of the function* $\arctan\left(\frac{y}{x}\right)$ *on* $[-1,1]^2$. *The top shows the approximation, the diagram is displayed at the bottom, and a close-up of the center is provided on the right. The samples align along the direction of the gradient of* f *and on both sides of the discontinuity, to ensure that the boundary of the Voronoï cells match the discontinuity. 500 samples, 1500 iterations for 20 sec runtime*



Modélisation Géométrique - Résumé

Programme

- **Préliminaires**
- **Tour d'horizon des représentation des surfaces**
  - Courbes & Surfaces paramétriques
  - Surfaces implicites
- **Courbes/surface paramétriques**
  - Bézier, Bspline, NURBS
  - Produit tensoriel
- **Maillages :**
  - surfaces de subdivisions
  - half-edges

Modélisation Géométrique

Généralités & préliminaires

Quelques notions

- **Dimensions**
  - Dimension de l' « espace ambiant »
    - Ex : courbe dans le plan (R^2) ou dans R^3, ou dans R^d
  - Dimension de l' « objet » (espace topologique)
    - Point : 0D
    - Droite, Segment, Courbe : 1D
    - Plan, Surface : 2D
    - Volume : 3D

- **Manifold (variété)**

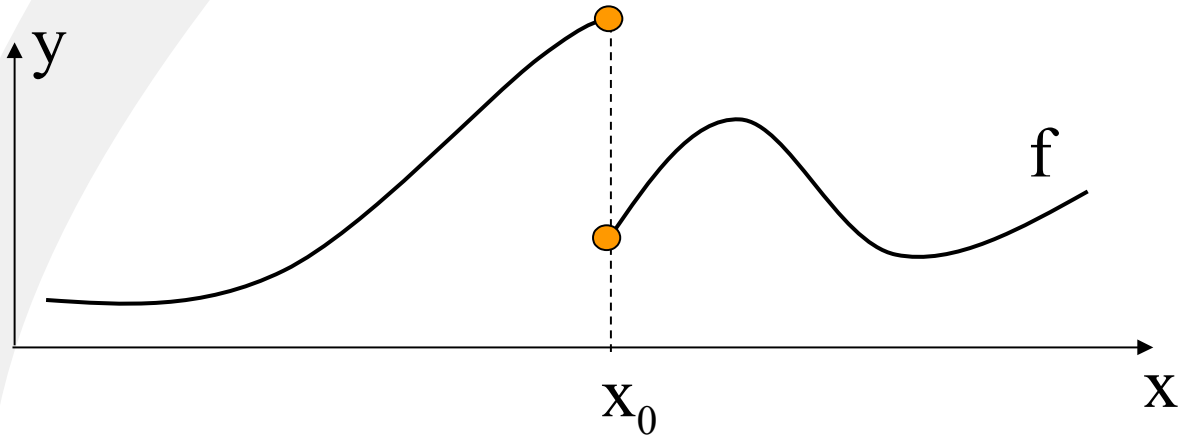
  un manifold de dimension n est un espace topologique de dimension n qui est en tout point localement homéomorphe à un espace Euclidien de dimension n

  - Notion de « (d-1) -manifold »
  - Exo : donner des exemples de courbes et surfaces manifold et non-manifold

Continuité d'une courbe

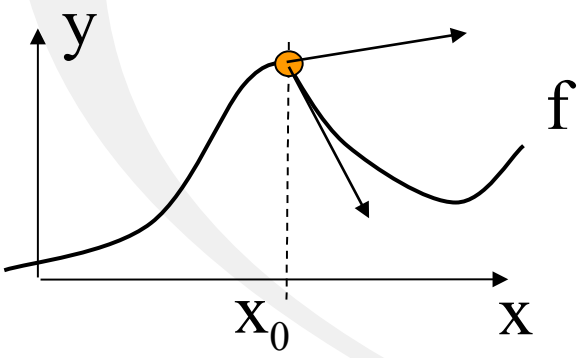
Soit une fonction f : R→ R

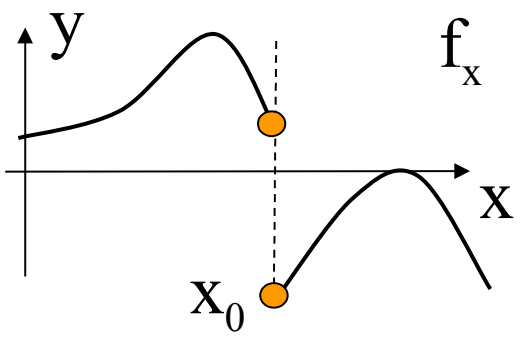
Si en $x=x_0$ $f^-(x_0) \neq f^+(x_0)$ la courbe est discontinue en x_0



« C^{-1} »

Si $f^-(x_0) = f^+(x_0)$ et $f_x^-(x_0) \neq f_x^+(x_0)$



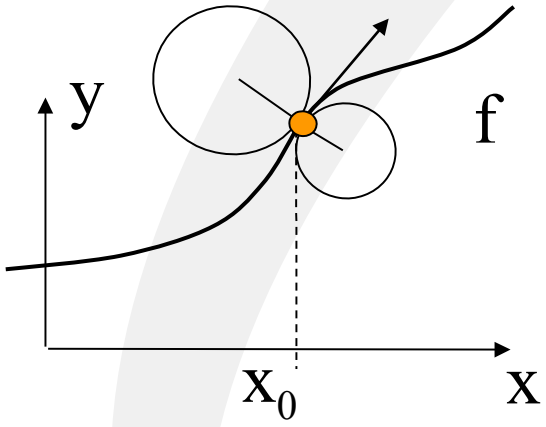


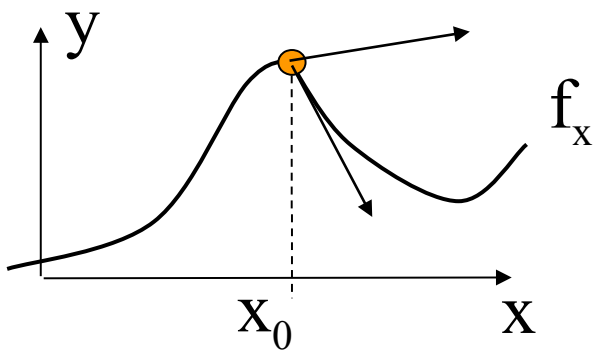
C^0

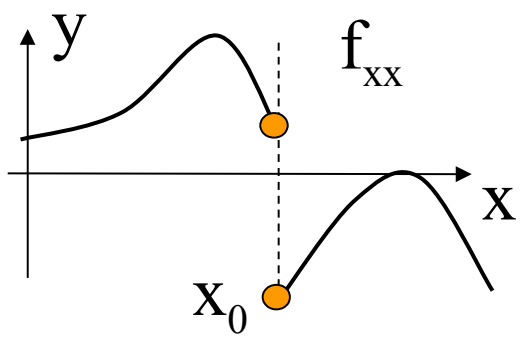
Continuité d'une courbe

Si $f^-(x_0) = f^+(x_0)$, $f_x^-(x_0) = f_x^+(x_0)$ et $f_{xx}^-(x_0) \neq f_{xx}^+(x_0)$

C^1

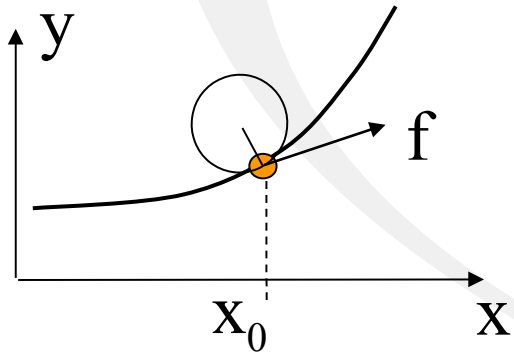


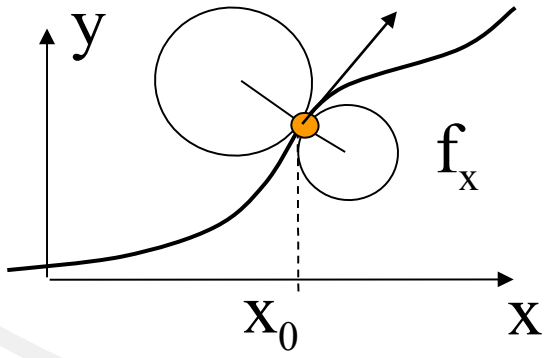


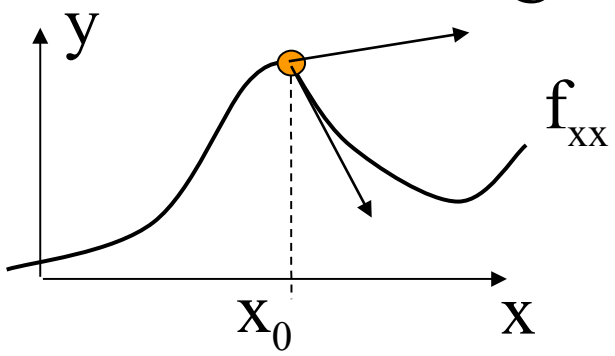


Si $f^-(x_0) = f^+(x_0)$, $f_x^-(x_0) = f_x^+(x_0)$, $f_{xx}^-(x_0) = f_{xx}^+(x_0)$ et $f_{xxx}^-(x_0) \neq f_{xxx}^+(x_0)$

C^2







Continuité

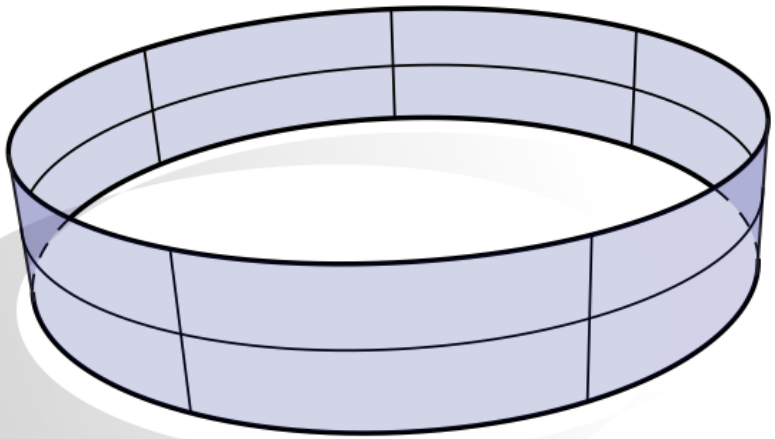
- C^-1 : trous
- C^0 : arêtes franches
- C^1 : « lisse » mais reflets avec des coins
- C^2 : reflets lisse
- C^3 : etc.

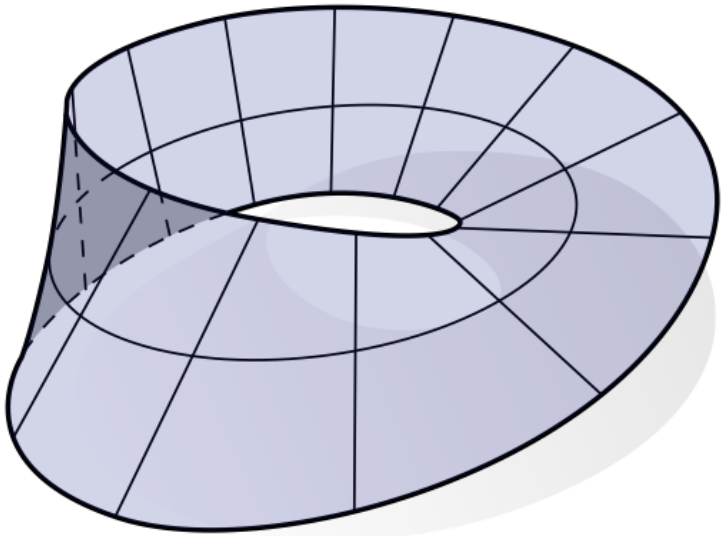
Tangentes (C^1 ou plus)

- **Pour une courbe *C* :**
  - vecteur tangent en un point *p* sur *C* :
    droite passant par *p* et *q* lorsque $q \rightarrow p$ avec *q* sur *C*
  - vecteur normal : vecteur orthogonal à la tangente
  - *en 3D : vecteur bi-normal*

- **Surfaces :**
  - plan tangent : ensemble de toutes les droites tangentes en un point
  - vecteur normal : vecteur orthogonal au plan tangent

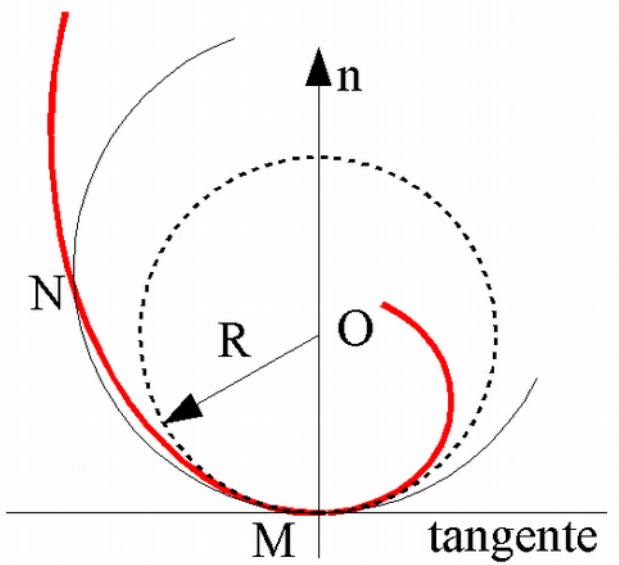
Orientabilité





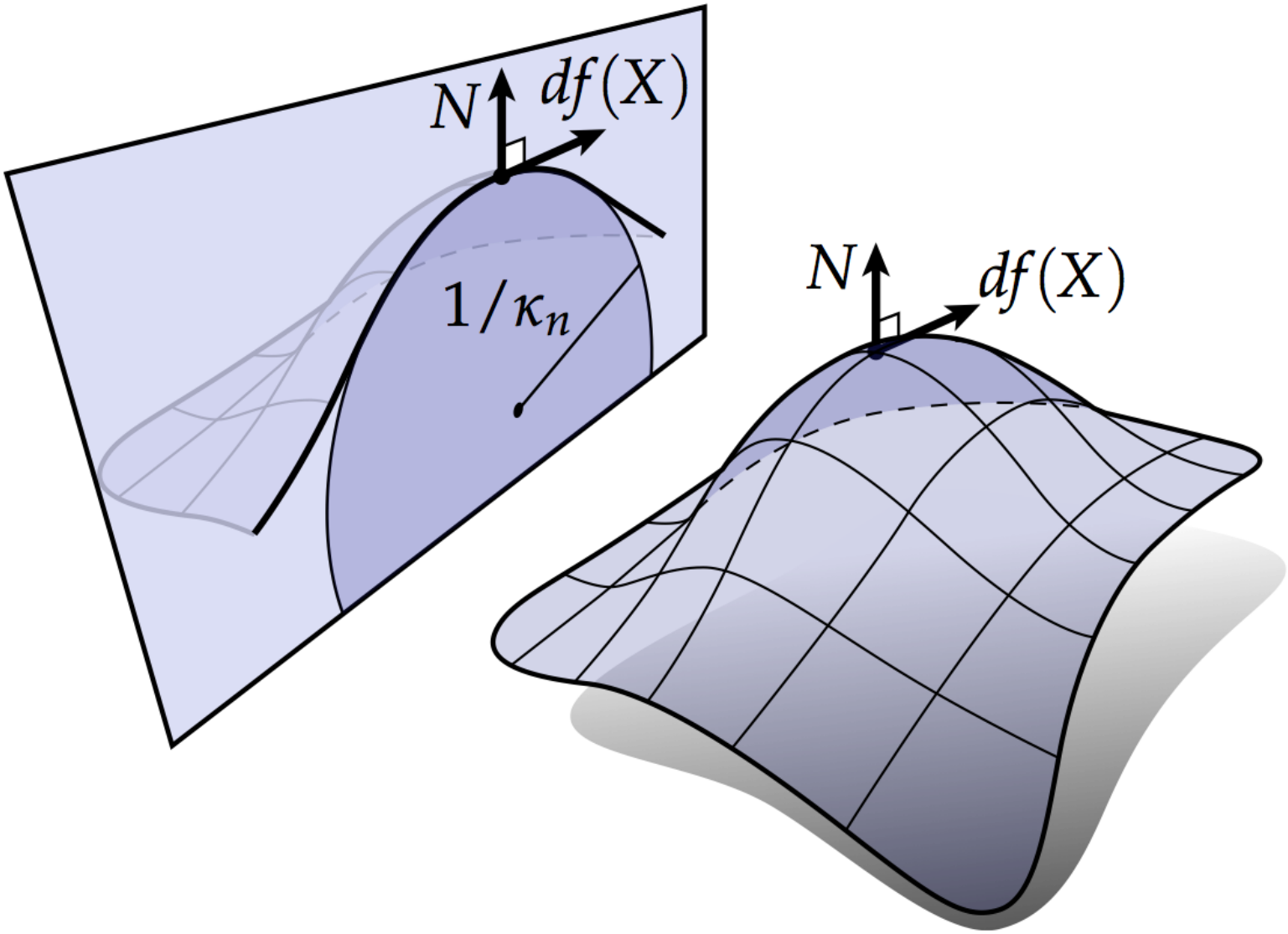
Courbure (C^2 ou plus)

- **Pour une courbe**
  - inverse du rayon du cercle osculateur (tangent)
  - orientation → courbure signée possible
    - concavité vs convexité

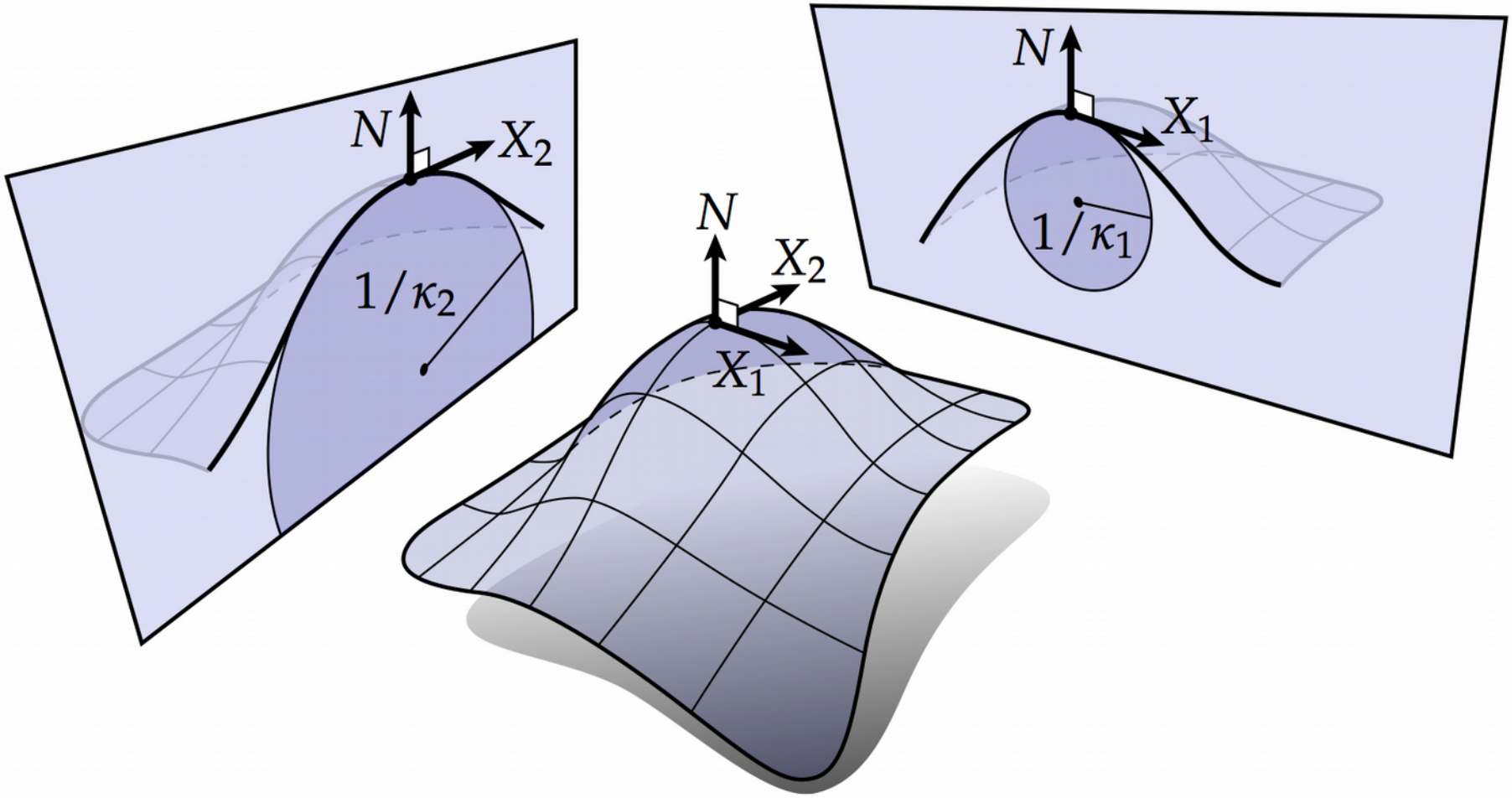


- **Pour une surface**
  - en un point, infinité de courbures
  - courbure « normale », i.e., dans une direction *t*, *kn(t)*
  - notions de courbures principales : maximale *k1* et minimale *k2*
  - courbure moyenne : *(k1+k2)/2*
  - courbure Gaussienne : *k1*k2*

Courbure normale

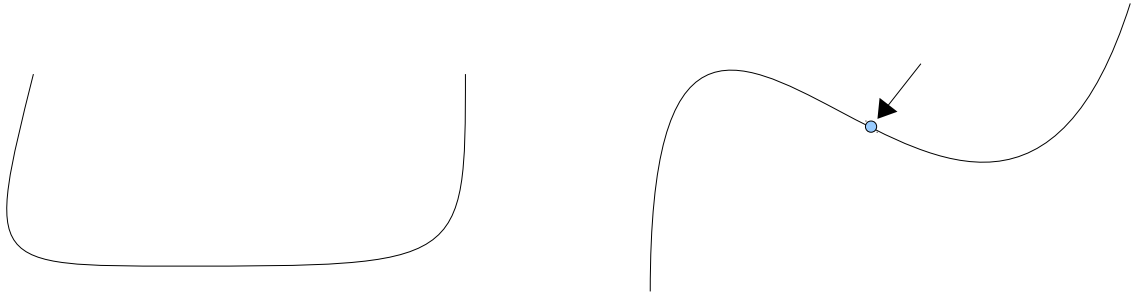


Courbures principales

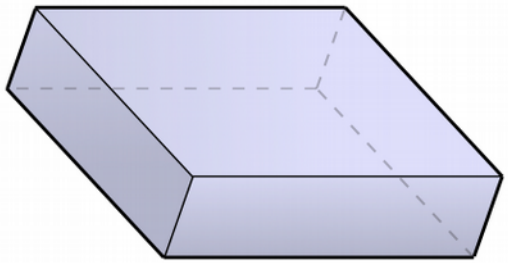


Point d'inflexion

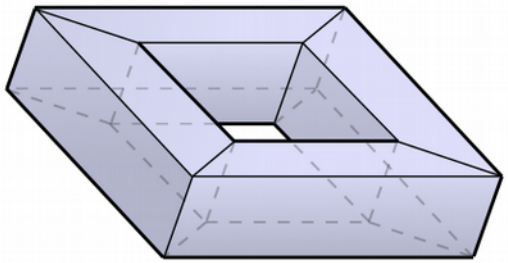
- **Pour une courbe au moins C^2:**
  - point où la courbure change de signe (courbure nulle en ce point)



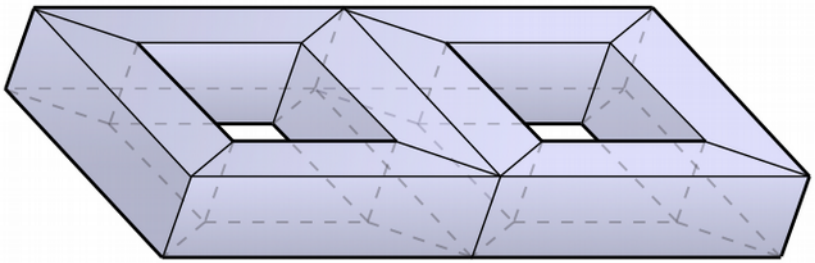
Genus



sphere
$(g = 0)$



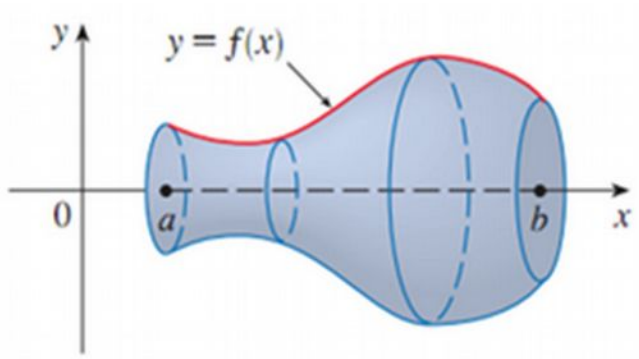
torus
$(g = 1)$



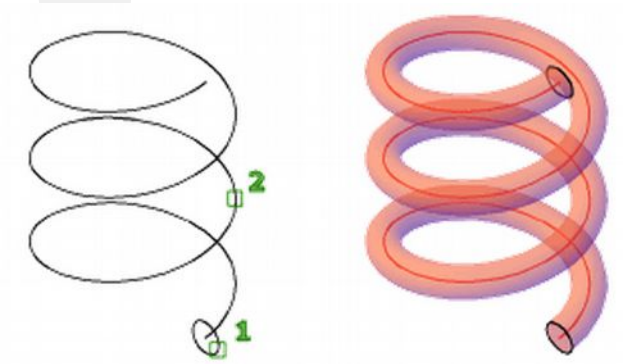
double torus
$(g = 2)$

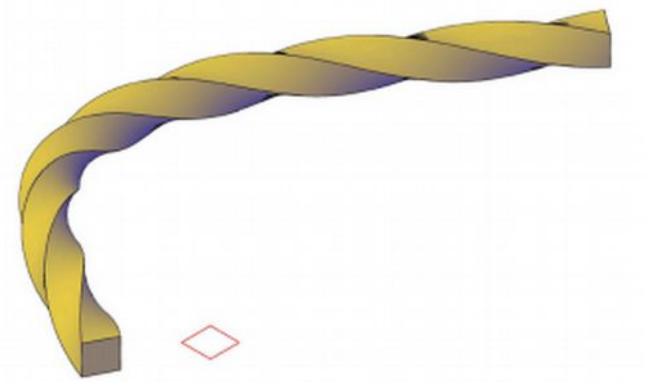
Générer une surface à partir de courbes

- **Surface de révolution**
  - courbe + révolution autour d'un axe

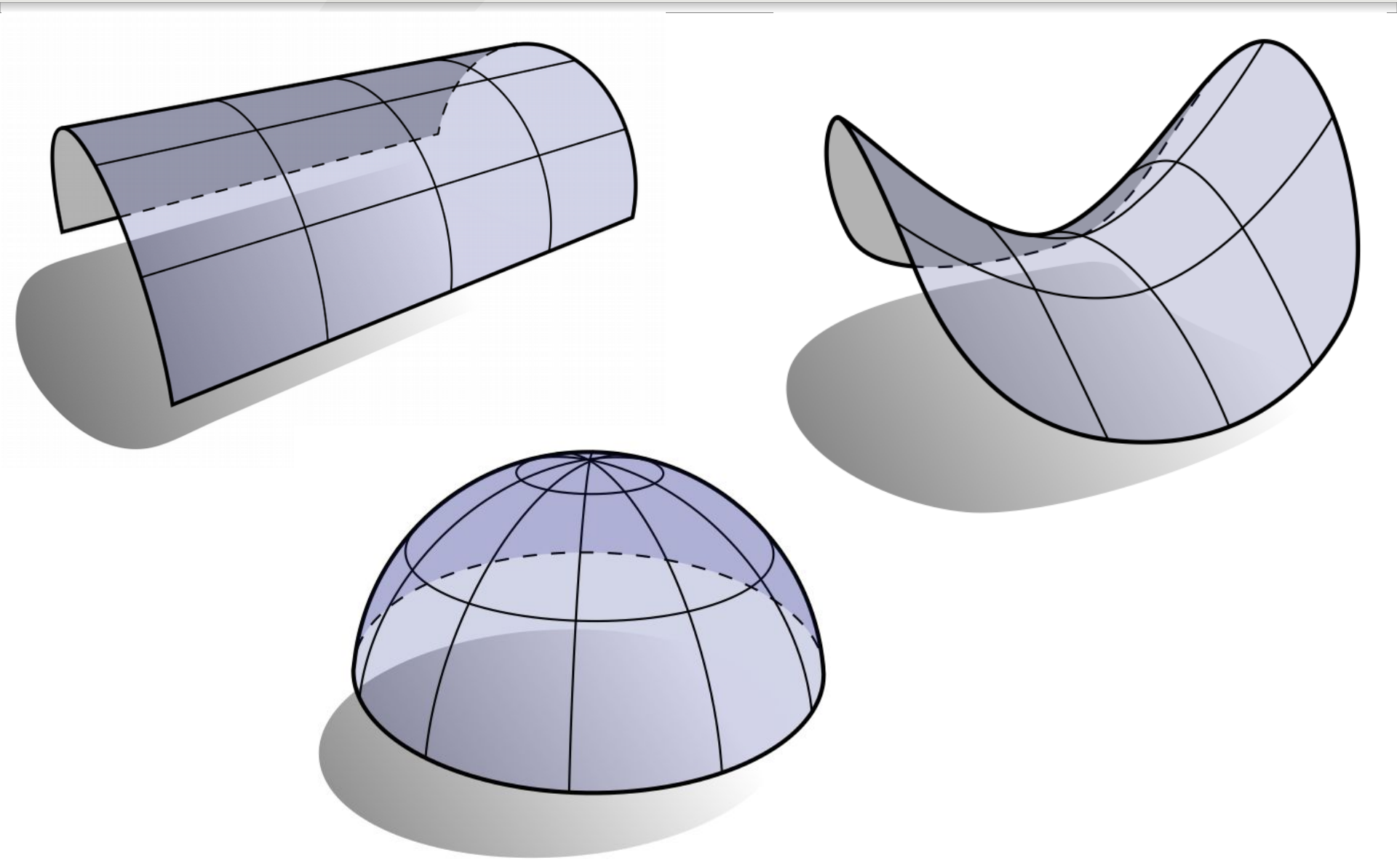


- **Surface par balayage ou extrusion**
  - balayage d'une courbe profil le long d'une courbe directrice





Formes remarquables



Intrinsèque versus Extrinsèque

- **Définitions (intuitives) :**
  - Une propriété ou mesure d'une courbe ou surface est dite **intrinsèque** si elle ne dépend pas du plongement de l'objet dans l'espace ambiant.
  - Dans le cas contraire il s'agit d'une propriété **extrinsèque**. Corollaire : Une propriété extrinsèque n'existe pas sans l'espace ambiant.
- **Exemples :**
  - Vecteurs tangents : ?
  - Vecteur normal : ?
  - Distance géodésique : ?
  - Courbure d'une courbe : ?
  - Courbures principales : ?
  - Courbure moyenne : ?
  - Courbure Gaussienne : ?
  - Déformation isométrique (qui préserve les distances) : ?

Modélisation Géométrique

Tour d'horizon des représentations de surfaces

Comment « décrire » une courbe ou une surface ?

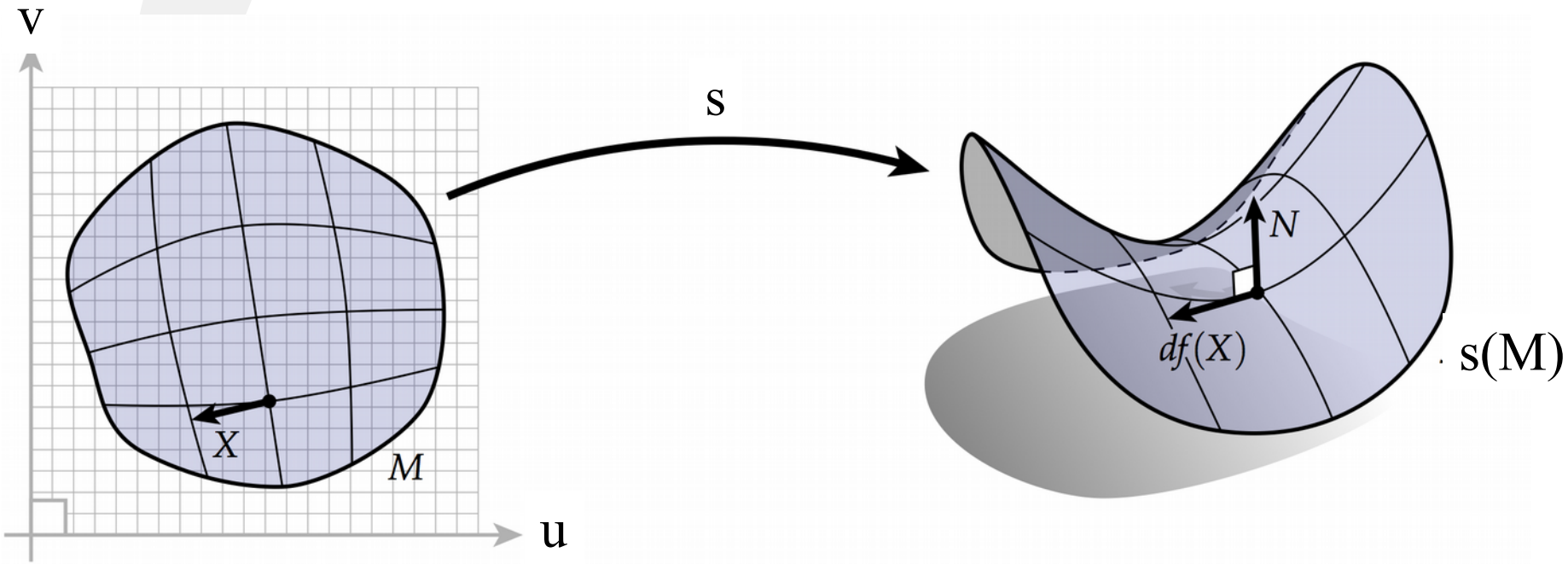
- **Différentes approches**
  - « champ de hauteur »
    - Notion de « +0.5D »
  - Forme paramétrique
    → courbe paramétrique, surface paramétrique
  - Forme implicite
    → équation cartésienne, surface implicite, iso-curve, iso-surface
    - Cas particulier : représentations volumiques discrètes
  - Discrète + équa-diff pour assurer les propriétés voulues
    → ex. : maillage + minimisation de la courbure, interpoler des bordures, etc.

Exo : droite, plan, sphère ;
formes matricielles ;
tangentes/normales ;

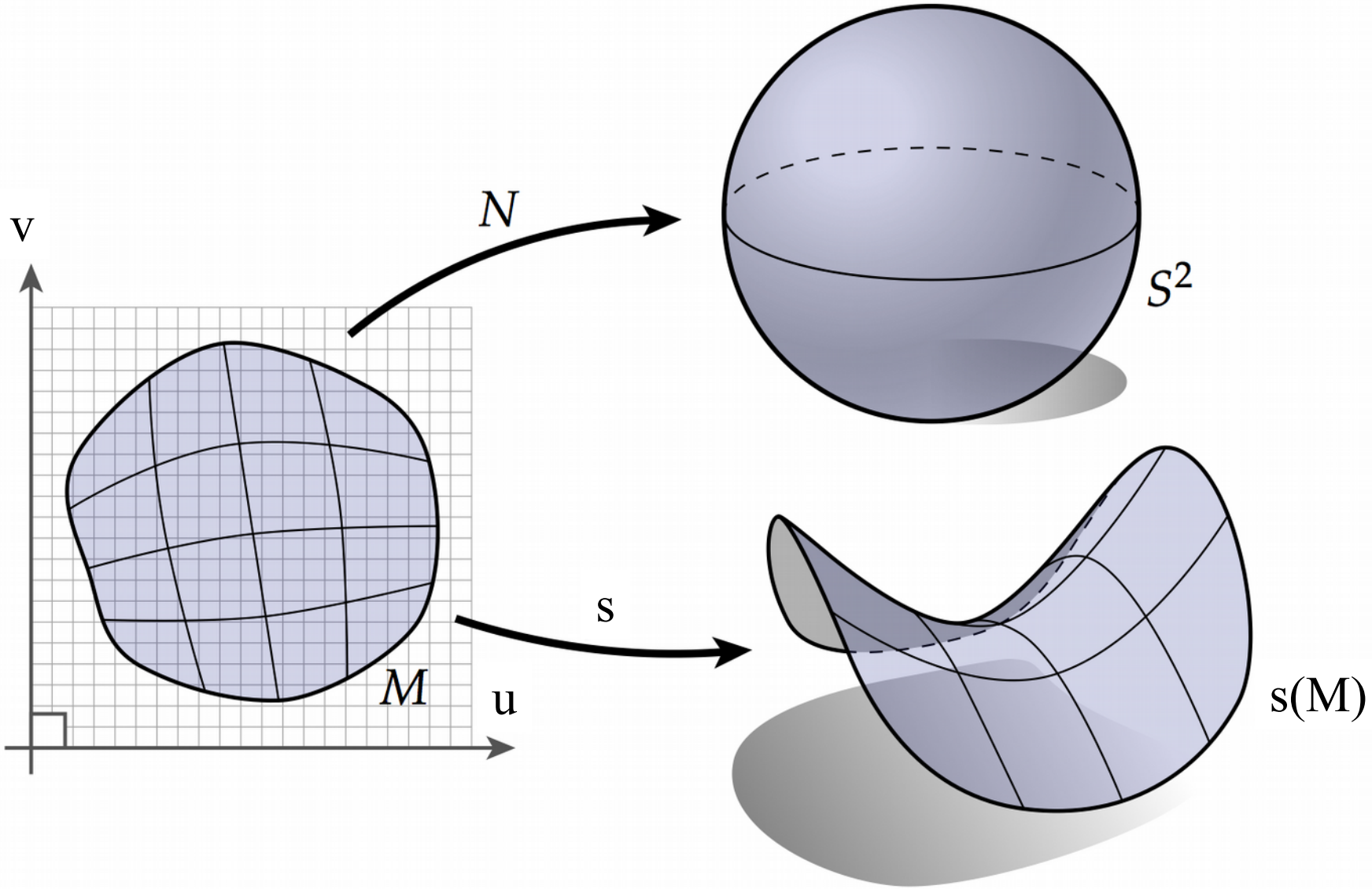
Généralisation aux dimensions supérieurs ?

Surface paramétrique

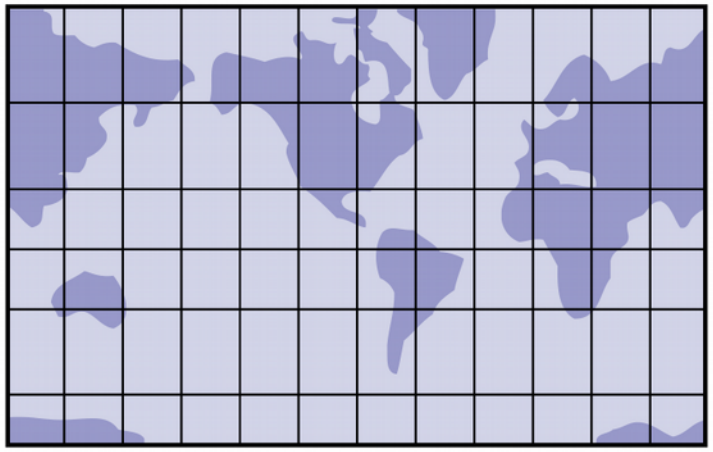
- **Courbe : c(u) = [x(u) y(u) z(u)]^T**
- **Surface : s(u,v)=[x(u,v) y(u,v) z(u,v)]^T**

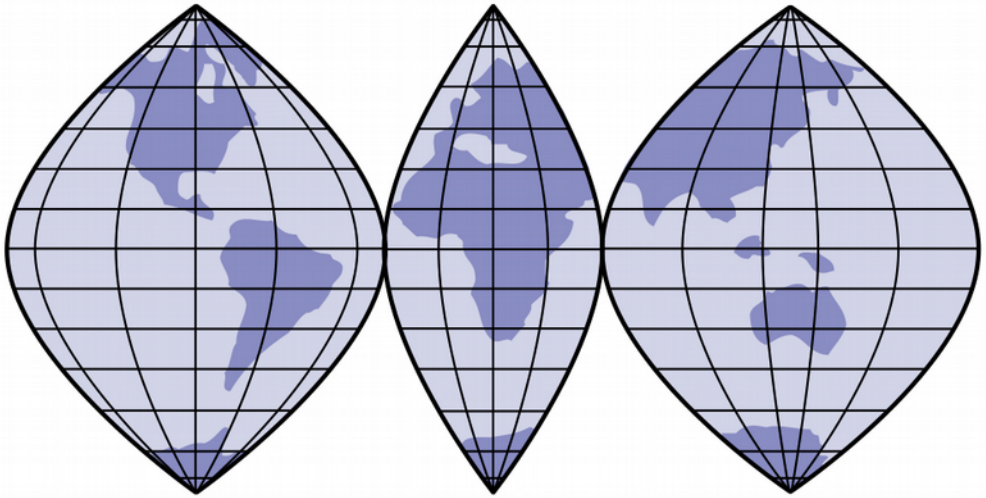


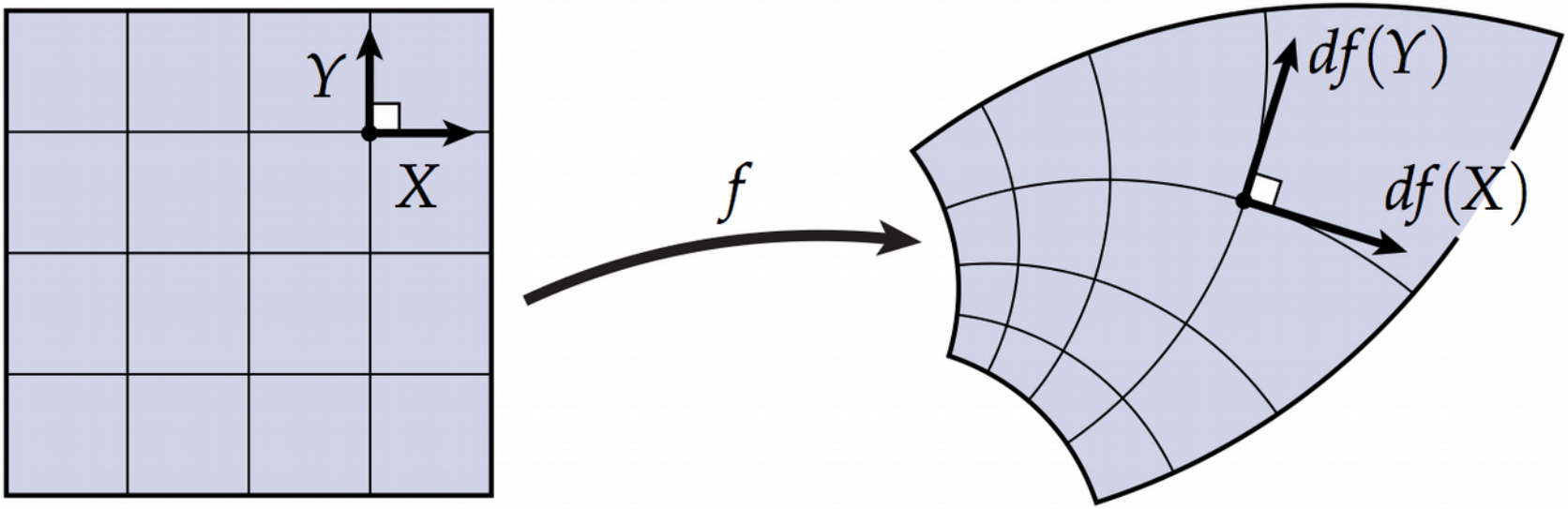
Surface paramétrique + Gauss map



Difficulté : paramétrisation ?

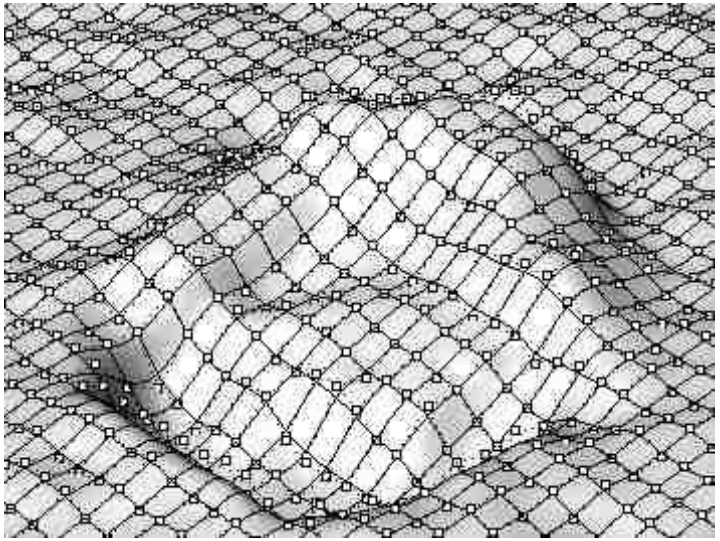






« champ de hauteur » (ou « graphe » ou « patch de Monge »)

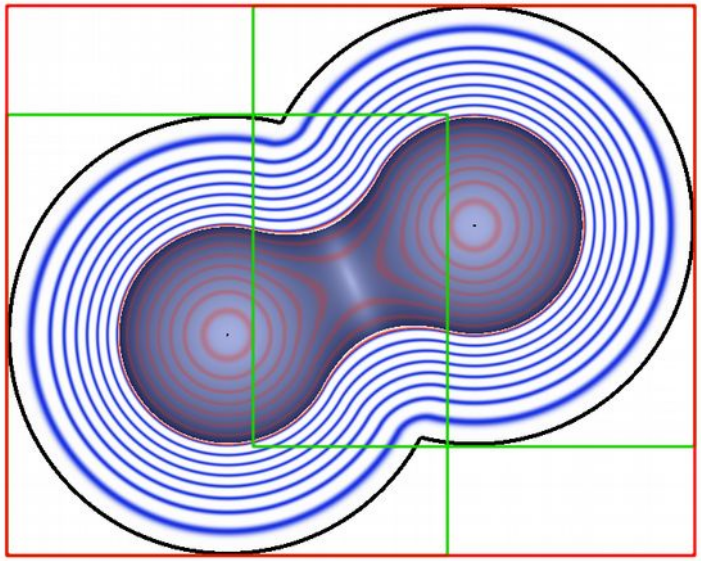
- **Courbe : y(x) = ...**
- **Surface : z(x,y) = …**
  - ex : image en niveau de gris
  - cas particulier des surfaces paramétriques, beaucoup plus simple
  - paramétrisation implicite
  - pas de repliement possible
  - Notion de « +0.5D »



Surfaces Implicites

- **Iso-contour d'un champ scalaire :**

$$S = \left\{ \mathbf{x} \in R^d \,;\, f(\mathbf{x}) = 0 \right\} \quad , \qquad f : R^d \rightarrow R$$



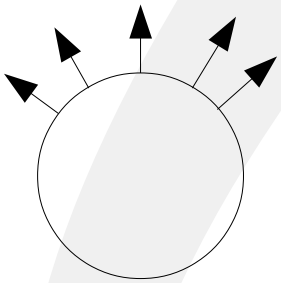
  - Courbe, exemple :
    courbes de niveaux d'un champ de hauteur
  - Notion de volume,
    connaissance de l'intérieur & extérieur
  - Cas particulier :
    f = distance à la surface

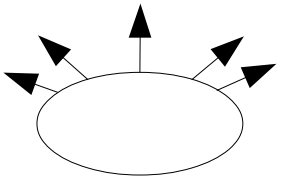
  - Difficultés :
    - où est la surface ? → comment contrôler la surface ?
    - pas de paramétrisation naturelle

  - Utilisation principale : reconstruction à partir d'un nuage de points
    → hybride explicite/implicite car la surface interpole/approche les points

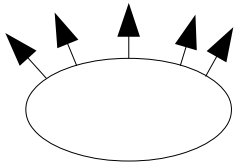
Transformation des normales

- **Problème d'une mise à l'échelle non uniforme M :**





n' = M * n
=> Faux !

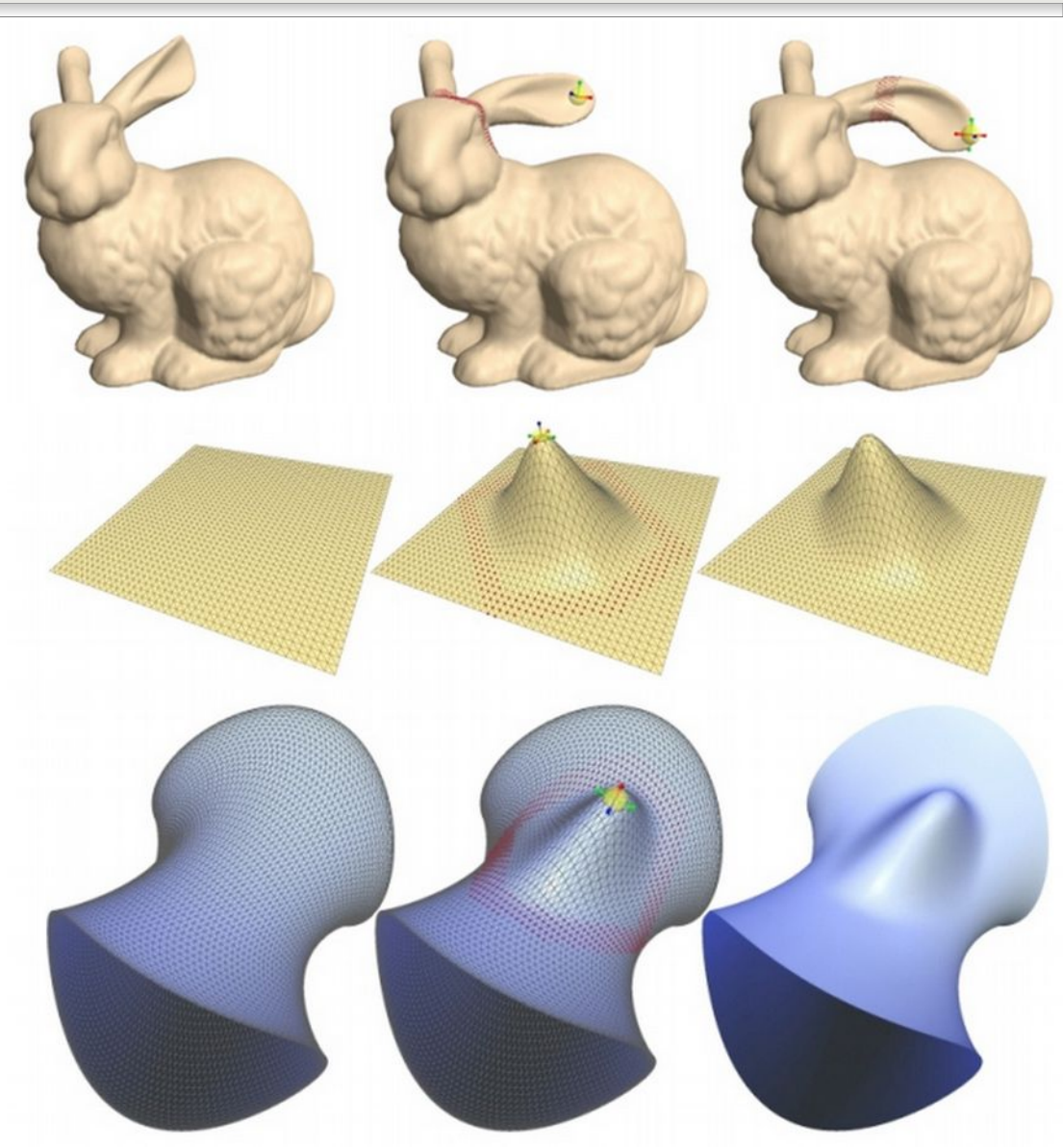


correct

- **Règle: utiliser la transposée de l'inverse:**
  - Soit M la matrice 3x3 de transformation courante:
    $v' = M * v + T$
  - Alors :
    $n' = (M^{-1})^t * n$

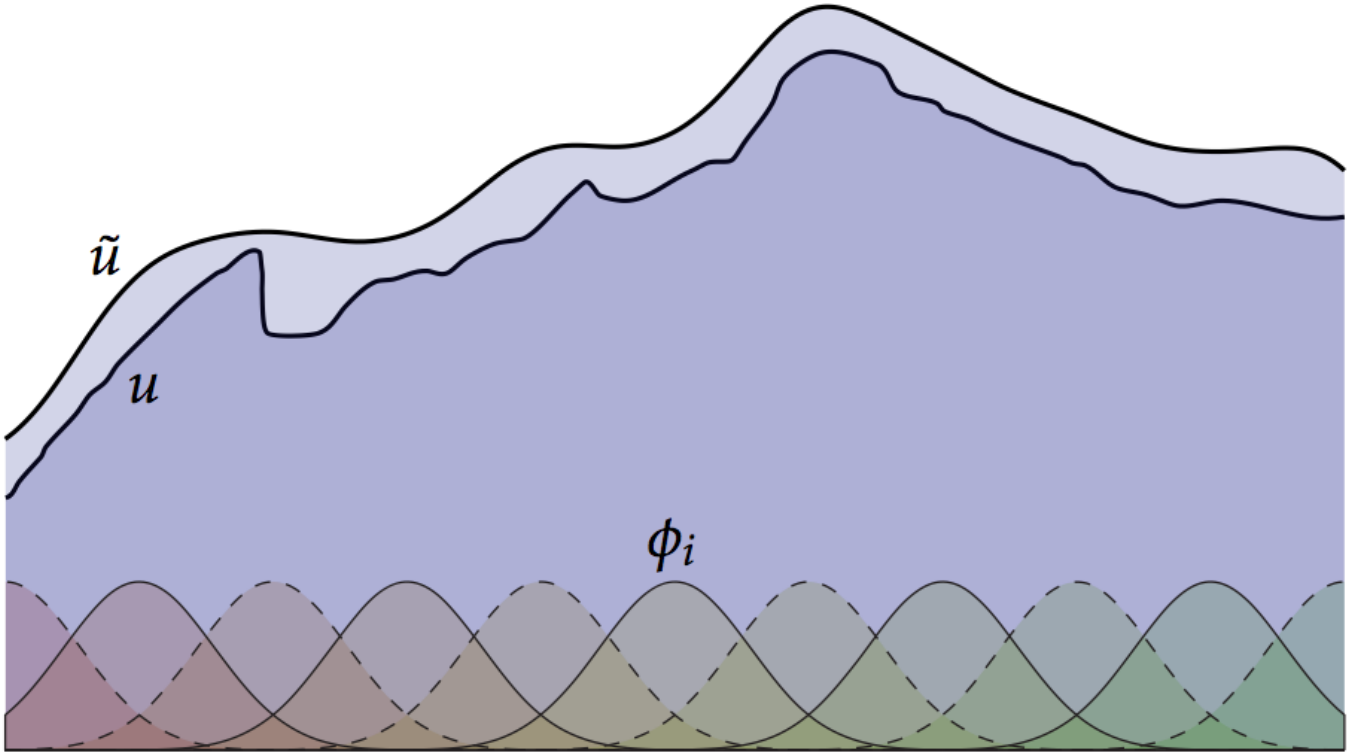
Approches variationnelles [moderne]

- **Représentation explicite discrète (ex. maillage dense)**
- **+ contraintes & processus d'optimisation**
- **Exploite la puissance de calcul des processeurs**



Discrétisation : Base de fonctions

- **Exemples : constant, linéaire, Gaussienne, Monomes, Bernstein, Legendre, etc.**



Modélisation Géométrique

Courbes paramétriques

Intro géométrie différentielle

- **Définir un segment, demi-droite, une droite**
  → interpolation linéaire
  - Distinction du sous-espace topologique et de la courbe paramétrique
- **Abscisse curviligne**
  - Vecteur tangent (unitaire) (T)
  - Vecteur normal principal (N)
  → binormal B → Frenet
  → plan osculateur et plan tangent

$$T = \frac{\dot{p}}{\|\dot{p}\|} \qquad B = \frac{\dot{p} \wedge \ddot{p}}{\|\dot{p} \wedge \ddot{p}\|} \qquad N = B \wedge T$$

- **Courbure**

$$\frac{d^2 p}{ds^2} = \frac{dT}{ds} = kN \qquad k = \frac{\|\dot{p} \wedge \ddot{p}\|}{\|\dot{p}\|^3}$$

- **Torsion**

$$\frac{dB}{ds} = \tau N \qquad \tau = \frac{\dot{p} . (\ddot{p} \wedge \dddot{p})}{\|\dot{p} \wedge \ddot{p}\|^2}$$

  - Si torsion=0 ? Si courbure = 0 ?

Qques preuves

- **Dérivé d'un vecteur unitaire ?**
  - orthogonal à lui même, i.e., $\frac{d\,\mathbf{v}(u)}{du}\cdot\mathbf{v}(u)=0$
  - 1ère approche, calculer

$$d\left(\frac{\mathbf{v}}{\|\mathbf{v}\|}\right)\cdot\mathbf{v}=\ldots$$

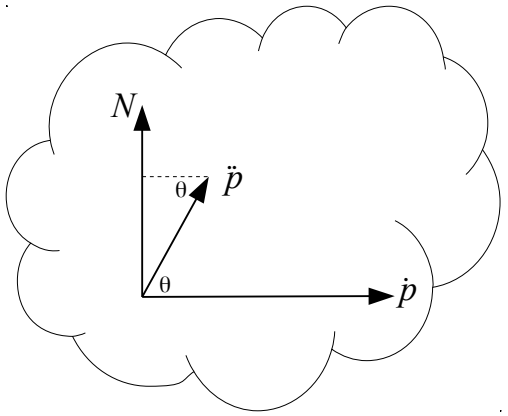
  - 2ème approche, observer que $\mathbf{v}/\|\mathbf{v}\|$ « vit » sur la sphère unitaire
    => dérivée tangente à la sphère, cqfd.

- D'où, $\frac{dT}{ds}$ est orthogonal au vecteur tangent *T*, et par définition, est aligné avec *N*

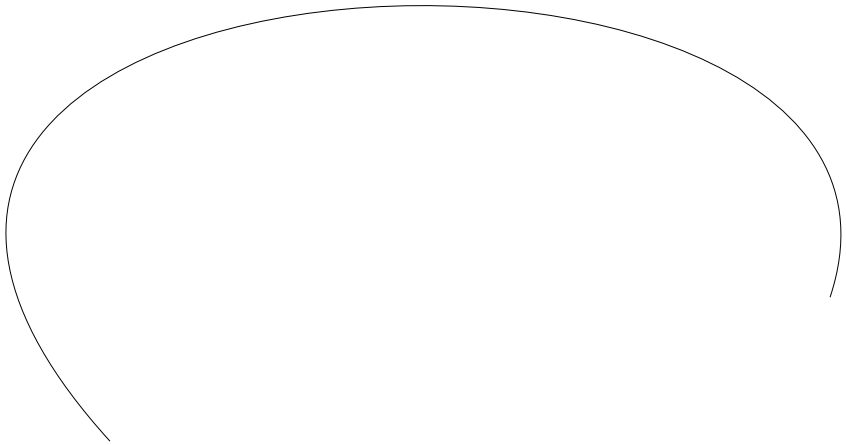
et

$$\kappa=\frac{dT}{ds}\cdot N=\frac{d}{du}\frac{\dot{p}}{\|\dot{p}\|^2}\cdot N=\frac{\|\dot{p}\|^2\,\ddot{p}-\dot{p}\frac{d}{du}\|\dot{p}\|^2}{\|\dot{p}\|^4}\cdot N$$

$$=\frac{\ddot{p}\cdot N}{\|\dot{p}\|^2}=\frac{\sin\theta\,\|\dot{p}\|\,\|\ddot{p}\|}{\|\dot{p}\|^3}=\frac{\|\dot{p}\wedge\ddot{p}\|}{\|\dot{p}\|^3}$$



Cubique d'Hermite



$$\mathbf{p}(u) = \begin{pmatrix} x(u) \\ y(u) \\ z(u) \end{pmatrix} = \begin{pmatrix} a_0 + b_0 u + c_0 u^2 + d_0 u^3 \\ a_1 + b_1 u + c_1 u^2 + d_1 u^3 \\ a_2 + b_2 u + c_2 u^2 + d_2 u^3 \end{pmatrix}$$

Cubique d'Hermite

- **Exercices sur les cubiques d'Hermite**
  - Retrouver l'équation
    - différente formes : somme de fonctions de bases, forme matricielle
    - tangente = ?
  - Matlab : tracer des fonctions de bases, tracer pour un jeux de données

Courbes de Bézier

- **Courbes de Bézier**
  - Objectifs : généraliser aux degrés supérieurs, manipulations plus simple
  - Somme de fonctions de bases
    → combinaison affine/centre de masse,
    → enveloppe convexe si poids > 0
  - Reformulation de la forme :

$$p(u)=\sum_{i=0}^{n} B_i^n(u)\,P_i \qquad u\in[a,b]$$

$$\sum_{i=0}^{n} B_i^d(u)=1 \qquad \forall\, u\in[a,b]$$

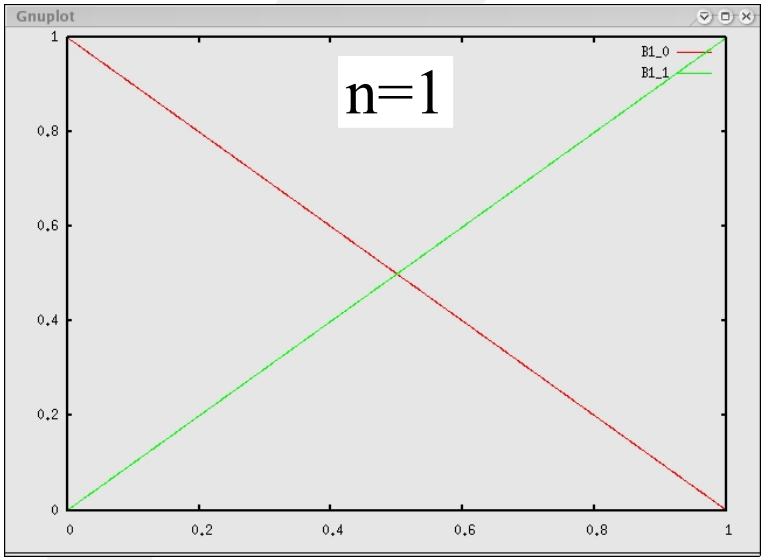
  - *B* = Polynomes de Bernstein :

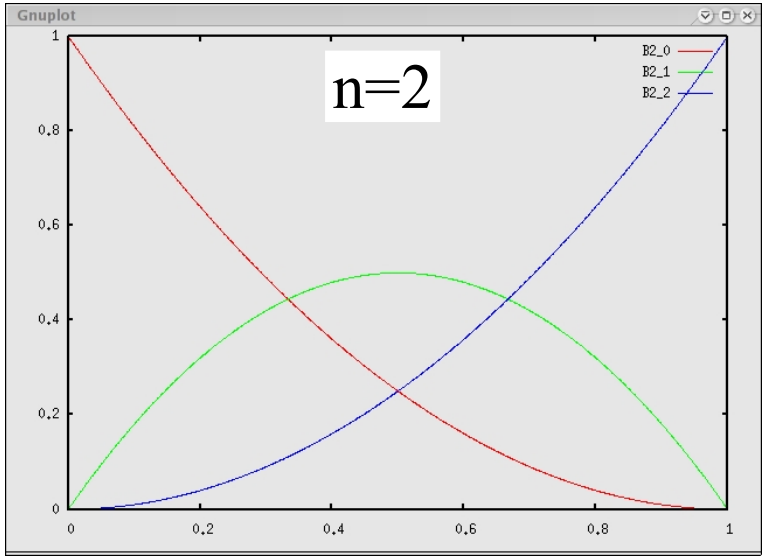
$$B_i^n(u)=\binom{n}{i}u^i(1-u)^{n-i}\,, \qquad i=0,\ldots,n \qquad \binom{n}{i}=\frac{n!}{i!(n-i)!}$$

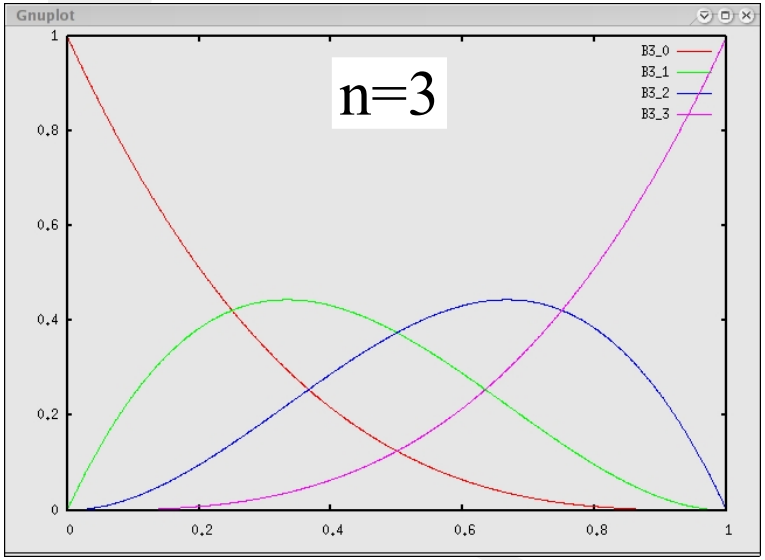
Courbes de Béziers

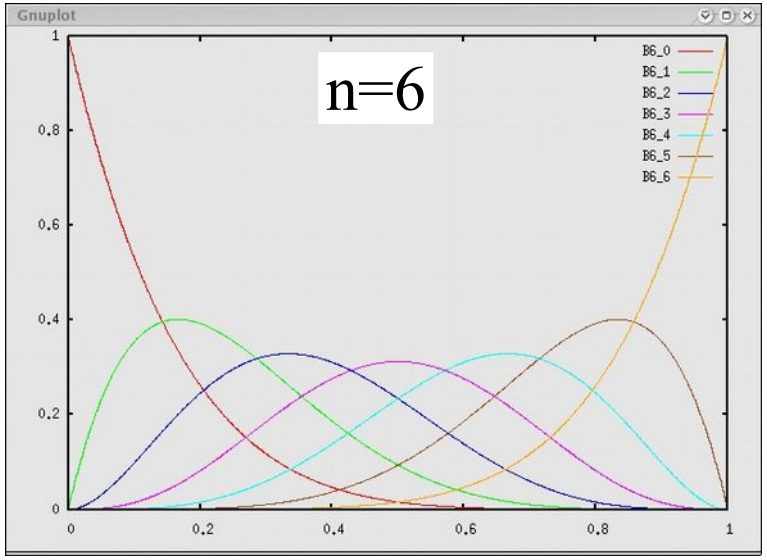
- **Propriété**
  - partition de l'unité : $1 = \left(u + \left(1-u\right)\right)^n = \sum_{i=0}^{n} \binom{n}{i} u^i \left(1-u\right)^{n-i} = \sum_{i=0}^{n} B_i^n\left(u\right)$
  - symétrique
  - positif sur [0:1]
  - combinaison affine de n+1 points (degré=n, ordre=n+1)
  - portée des fonctions de bases

Graphe des polynômes de Bernstein

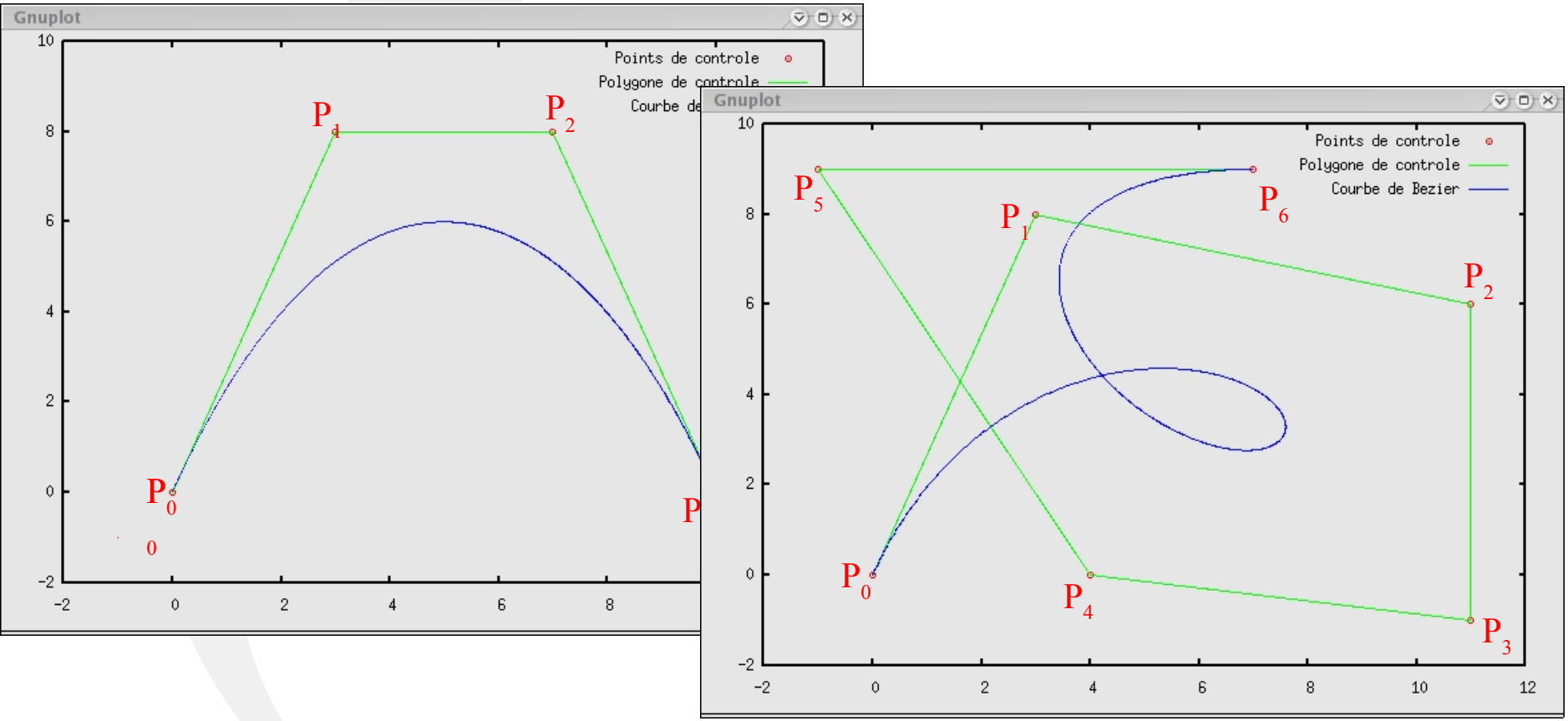






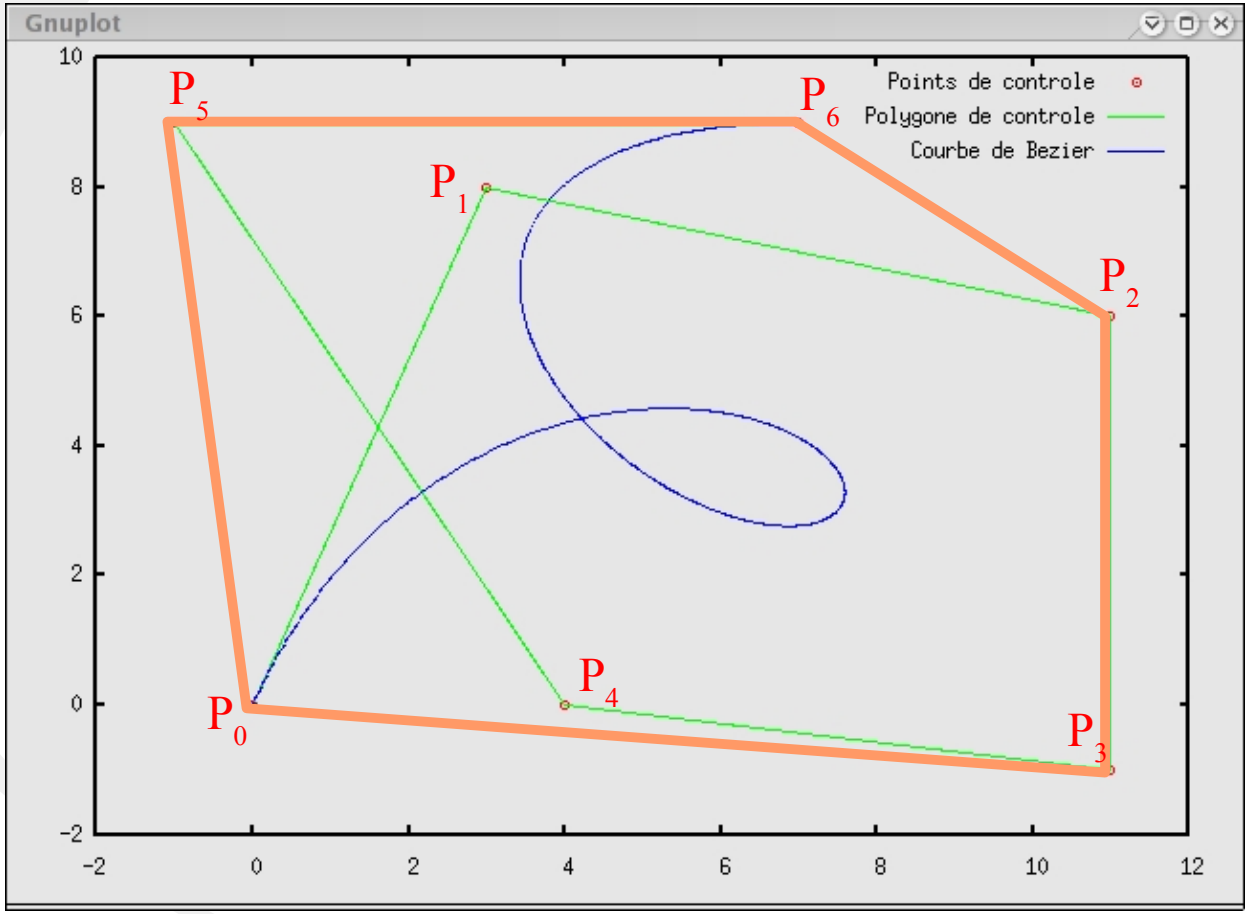


Exemples de courbes de Bézier



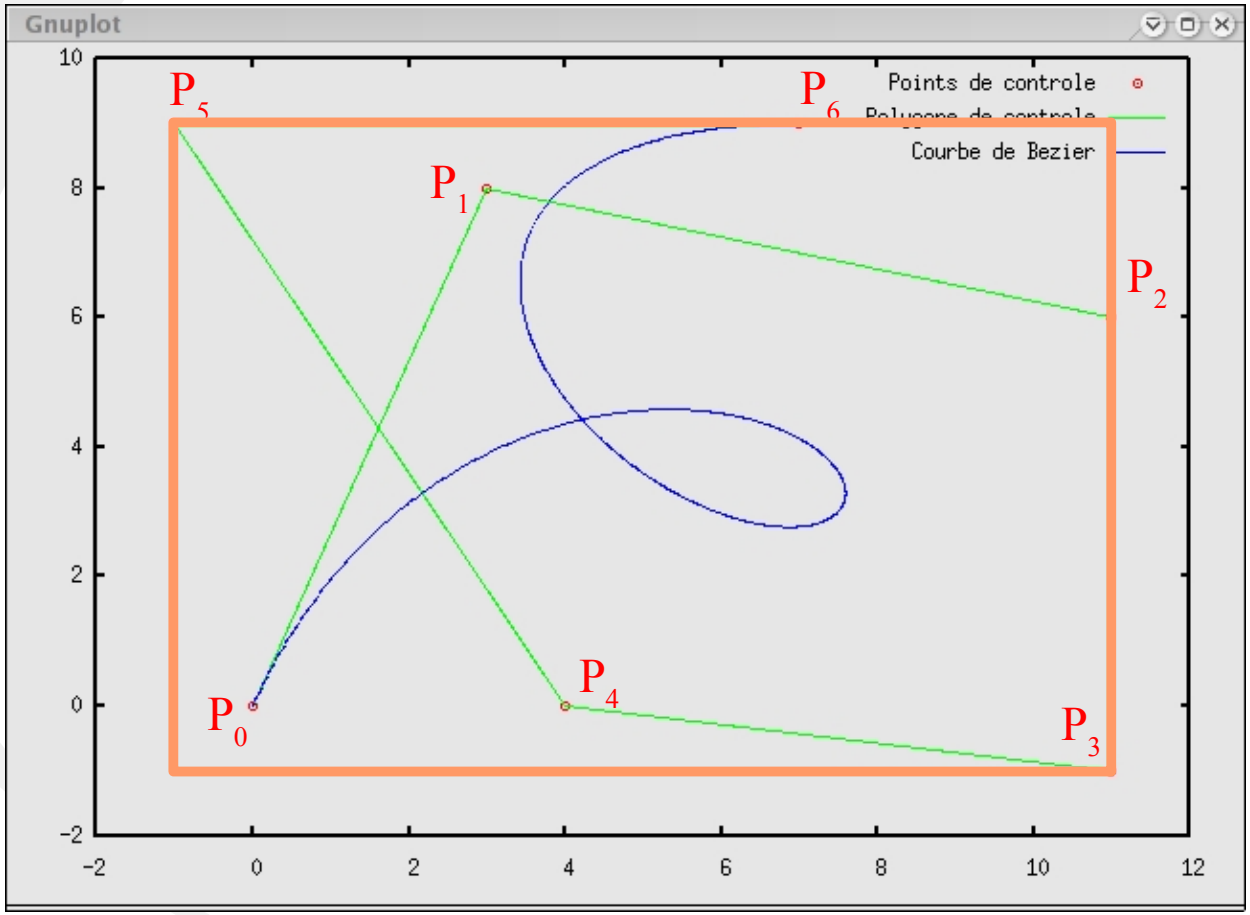
Enveloppe convexe

La courbe est inclue dans l'enveloppe convexe de son polygone de contrôle (car les polynômes de Bernstein sont positifs sur [0,1]).



Boîte englobante

En prenant individuellement le min et le max de chaque coordonnée des points de contrôle, on obtient une boîte englobante de la courbe qui est parallèle aux axes :



Bezier

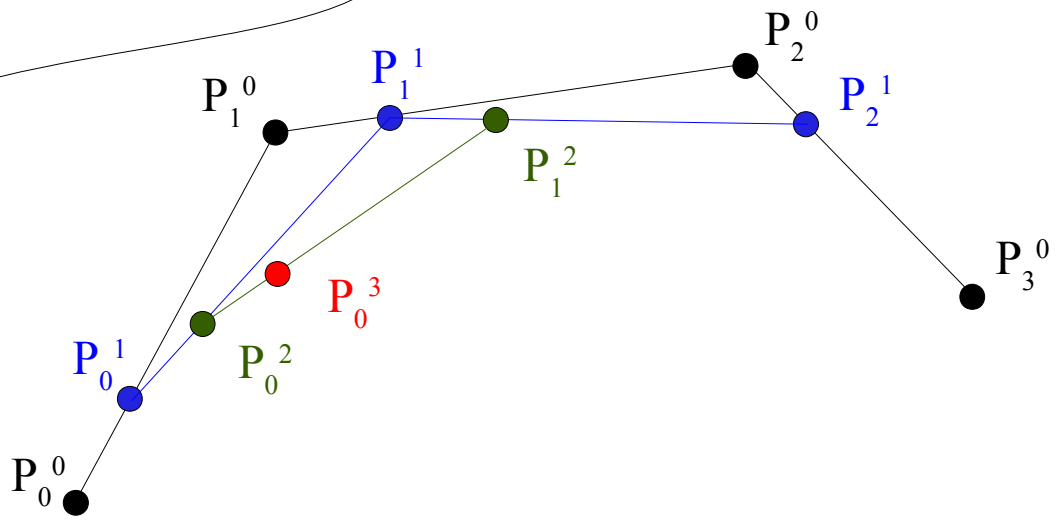
- **Algorithme de De Casteljau**
  - Récursivité : $\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}$

0: 1
1: 1 1
2: 1 2 1
3: 1 3 3 1
4: 1 4 6 4 1
5: 1 5 10 10 5 1
6: 1 6 15 20 15 6 1
7: 1 7 21 35 35 21 7 1
8: 1 8 28 56 70 56 28 8 1

    - D'ou : $B_i^{n+1}(u) = u\,B_{i-1}^{n}(u) + (1-u)\,B_i^n(u)$
    - D'ou :

$$p(u) = \sum_{i=0}^{n} B_i^n(u)\,P_i^0 = \sum_{i=0}^{n-1} B_i^{(n-1)}(u)\,P_i^1 = \ldots = \sum_{i=0}^{0} B_i^0(u)\,P_i^n = P_0^n$$

$$P_i^{k+1} = (1-u)\,P_i^k + u\,P_{i+1}^k$$

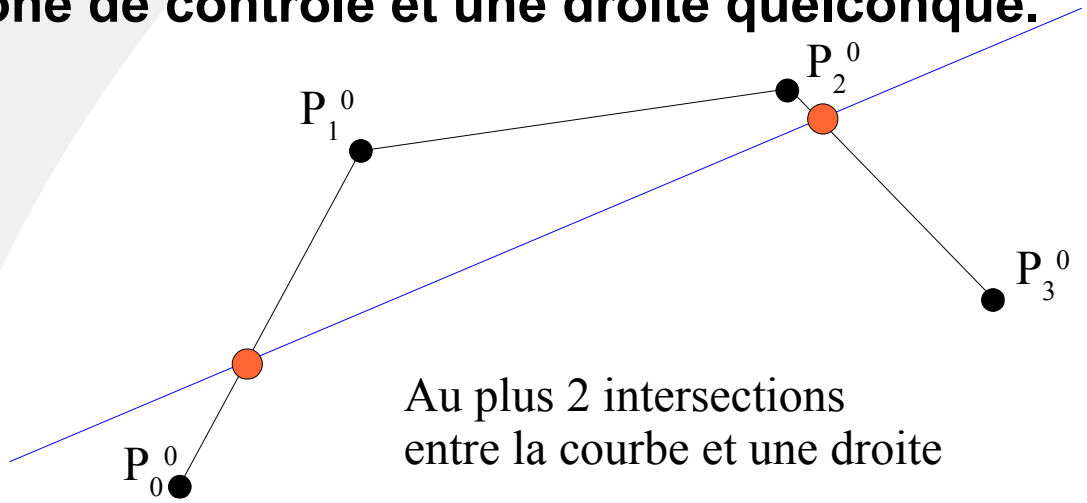


Bezier

- **Algorithme de De Casteljau**
  - Propriétés différentiels : tangent et plan osculateur
  - Applications :
    - discrétisation, visualisation
    - subdivision

Propriété de variation

Une courbe de Bézier ne peut pas avoir plus d'intersections avec une droite que le maximum d'intersection possible entre son polygone de contrôle et une droite quelconque.



Exercices :

Quel impact a cette propriété sur les oscillations de la courbe ?

Que peut-on dire des oscillations des morceaux de courbe lorsque l'on subdivise ?

Raccordement de deux courbes

Exercice :

soient deux courbes de Bézier :

p(u) de degré n, u dans [0,1], points de contrôle P_i

q(v) de degré m, v dans [0,1], points de contrôle Q_j

Donnez les conditions sur les polygones de contrôle pour que les courbes soient raccordées en u=1 et v=0 avec :

une continuité C^0

une continuité G^1

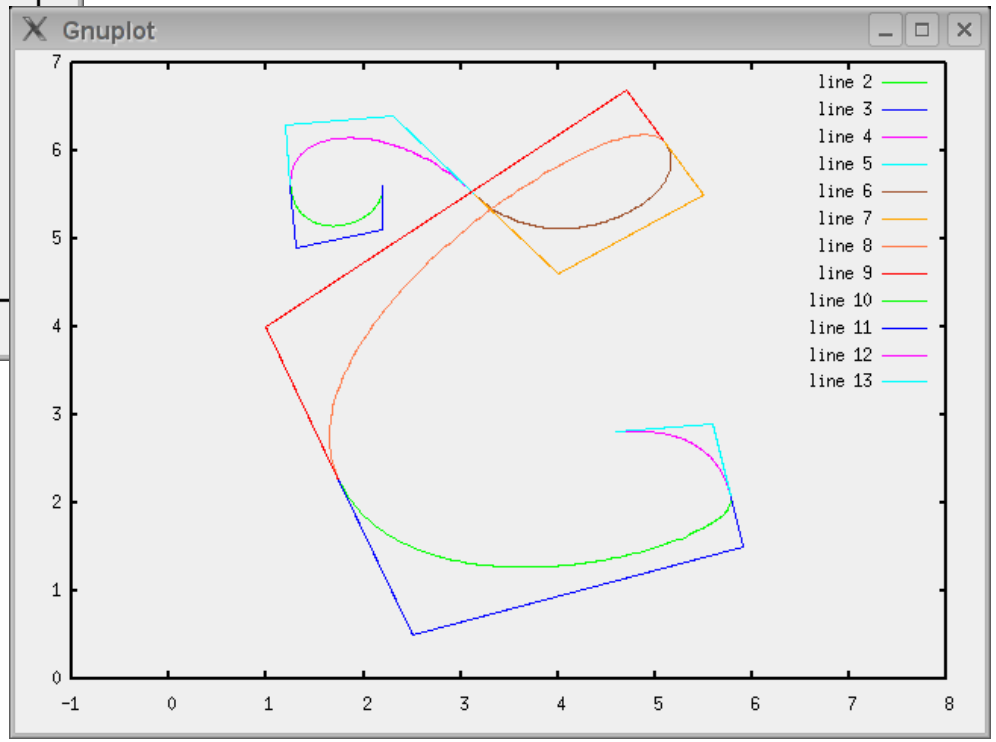
une continuité C^1

une continuité C^2

Exemple de modélisation de courbe complexe

Une unique courbe de degré 12

6 courbes de Bézier de degré 3 (et 2 à la fin) raccordées par continuité C^1



Avantages et inconvénients

- **Avantages :**
  - contrôle intuitif
  - courbe incluse dans le polygone convexe
  - simplicité de mise en oeuvre

- **Désavantages :**
  - support global
  - degré lié au nombre de points de contrôle

Les courbes B-spline

Les courbes B-spline sont plus « souples » que les courbes de Bézier. Elles permettent notamment un contrôle local de la courbe à partir des points de contrôle. Ici encore, chaque point de contrôle est associé à une fonction de base :

$$p(u)=\sum_{i=0}^{n} N_i^k(u)\,P_i \quad , \qquad u_{min} \leqslant u < u_{max}$$

Il y a **n+1 points de contrôle** : P_0, P_1, ... , P_n

Les N_i^k sont les **fonctions de base** d'ordre **k** et le **degré** de la courbe est **k-1**.

l'ordre k doit être choisi dans l'intervalle : 2 <= k <= n+1

Les courbes B-spline

Une B-spline est une fonction définie par morceaux

Il est nécessaire de définir un vecteur nodal composé de noeuds :

$$\left(u_0, u_1, \ldots, u_{k+n}\right) \qquad u_j \leqslant u_{j+1} \; \forall \; j$$

Il détermine les valeurs de *u* pour lesquelles les fonctions de bases sont définis.

La courbe est alors définie pour u dans [u_{k-1}, u_{n+1}].

Les fonctions de base N_i^k dépendent uniquement de la valeur de k et du vecteur nodal. Elles sont définies récursivement de la façon suivante (formule de Cox-de-Boor) :

$$N_i^1(u) = \begin{cases} 1 & , \quad u_i \leqslant u < u_{i+1} \\ 0 & , \quad sinon \end{cases}$$

$$N_i^k(u) = \frac{u - u_i}{u_{i+k-1} - u_i} N_i^{k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1}^{k-1}(u)$$

Comment contrôler une B-spline ?

- **Exemples :**
  - Déplacer un point de contrôle
  - Ajouter un point de contrôle
  - Utiliser un point multiple
  - Changer l'ordre k
  - Changer le type de vecteur nodal (uniforme, ouvert uniforme, ...)
  - Changer l'espacement des noeuds uj dans le vecteur nodal
  - Utiliser des noeuds multiples dans le vecteur nodal

- **Exercices :**
  - Comment créer un « coin » ?
  - Comment définir une courbe fermée ?
  - Application aux champs de hauteurs ?

NURBS

Modélisation Géométrique

Surfaces paramétriques

Définition générale

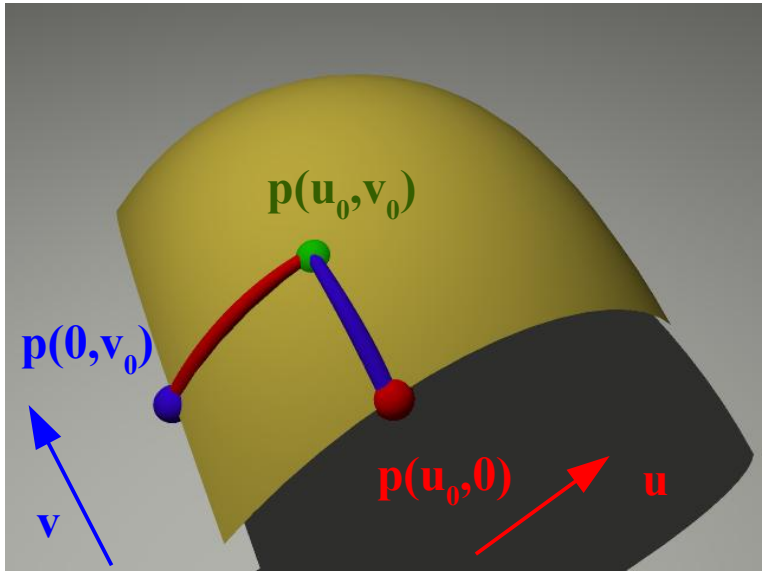
Une surface paramétrique dans l'espace R^3 est définie par une fonction f :

$$f: \quad D \times E \quad \rightarrow \quad \mathbb{R}^3$$

$$u, v \quad \rightarrow \quad p(u,v) = \begin{cases} x(u,v) = f_x(u,v) \\ y(u,v) = f_y(u,v) \\ z(u,v) = f_z(u,v) \end{cases}$$

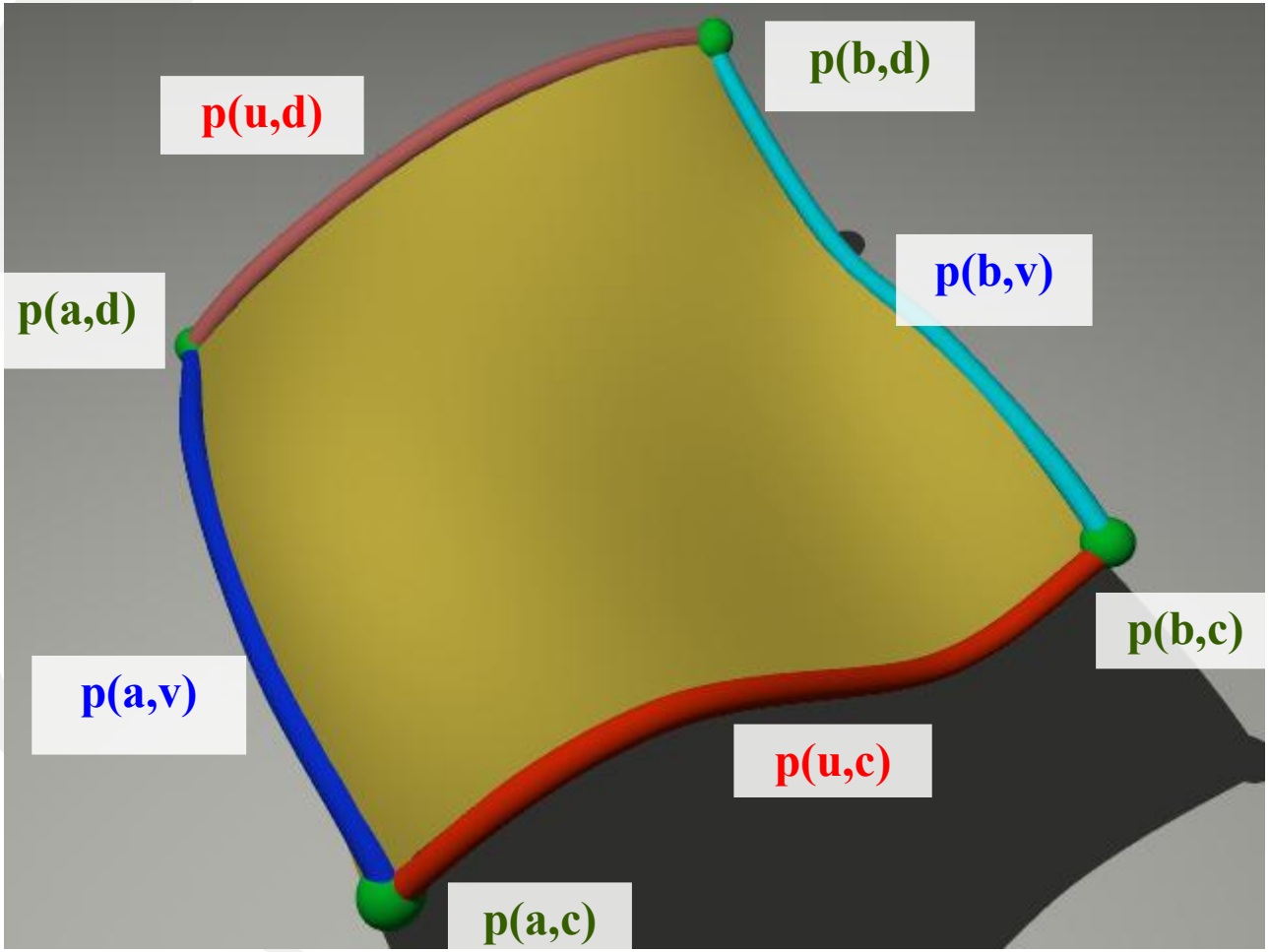
Ainsi quand u parcours D et v parcours E, le point p(u,v) parcours la surface

Le point $p(u_0,v_0)$ est l'intersection entre 2 courbes iso-paramétriques : celle à u constant ($u=u_0$ en bleu) et celle à v constant ($v=v_0$ en rouge).



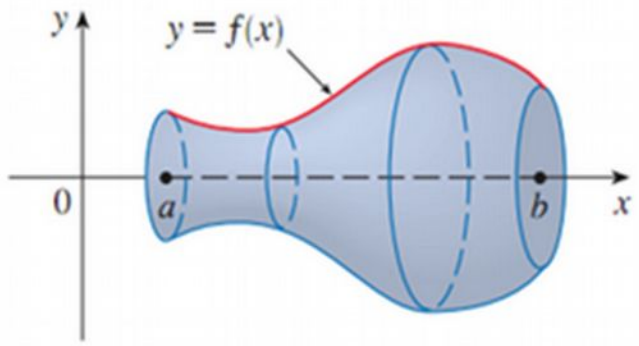
Coins et bords

Soient un (u,v) dans [a,b]x[c,d]

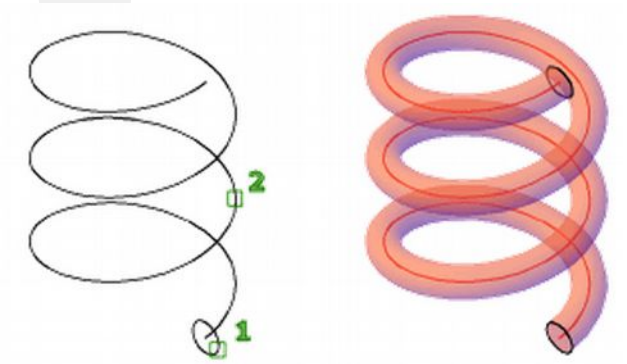


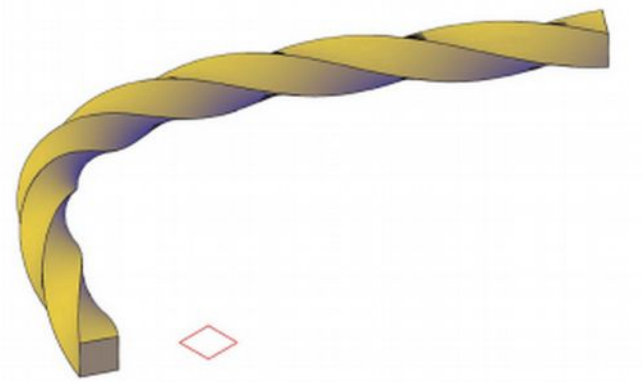
Exemple : surface paramétrique à partir de 2 courbes

- **Surface de révolution**
  - courbe + révolution autour d'un axe



- **Surface par balayage ou extrusion**
  - balayage d'une courbe profil le long d'une courbe directrice





Carreau bi-cubique

Un carreau bi-cubique est obtenu en faisant le produit tensoriel de fonctions polynomiales de degrés 3 (l'une suivant le paramètre u et l'autre suivant le paramètre v) :

$$p(u,v)=\sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}\,u^i\,v^j \quad , \quad a_{ij}\in\mathbb{R}$$

$$p(u,v)=a_{00}+a_{01}\,v+a_{02}\,v^2+a_{03}\,v^3+a_{10}\,u+a_{11}\,u\,v+a_{12}\,u\,v^2+a_{13}\,u\,v^3+\ldots+\ldots+a_{33}\,u^3\,v^3$$

Ceci nous amène à la formulation matricielle suivante :

$$p(u,v)=U\,A\,V^T=\begin{bmatrix}u^3 & u^2 & u & 1\end{bmatrix}\begin{bmatrix}a_{33} & a_{32} & a_{31} & a_{30}\\ a_{23} & a_{22} & a_{21} & a_{20}\\ a_{13} & a_{12} & a_{11} & a_{10}\\ a_{03} & a_{02} & a_{01} & a_{00}\end{bmatrix}\begin{bmatrix}v^3\\ v^2\\ v\\ 1\end{bmatrix}$$

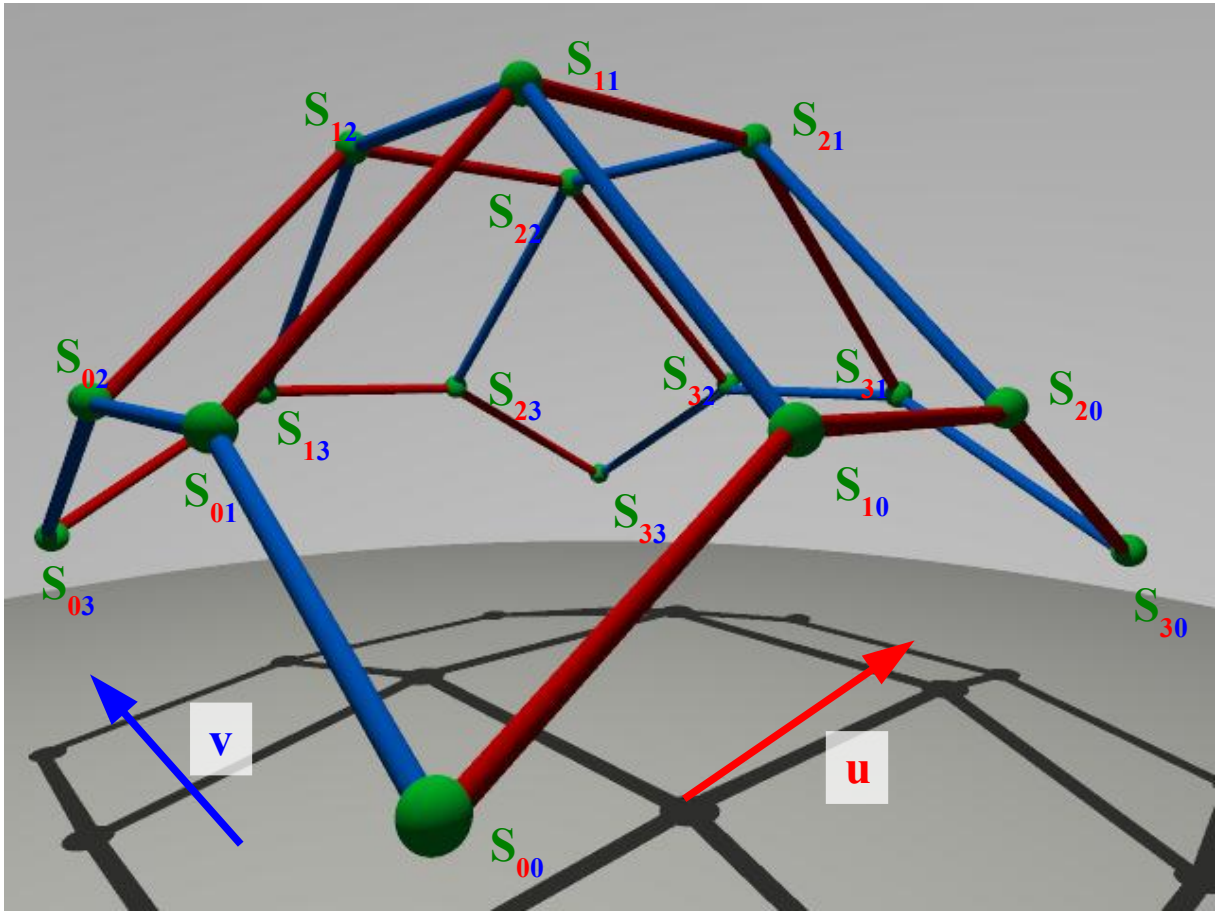
Comme pour les courbes cubiques, le contrôle de la forme de la surface par les coefficients a_{ij} n'est absolument pas intuitif => carreaux de Bézier

Carreaux : maillage de contrôle

$(u,v) \in [0,1]^2$

La surface est contrôlée à partir d'un maillage régulier composé de quadrilatères

Les points de contrôles sont les sommets S_{ij} du maillages et ils sont numérotés dans les directions de u et de v



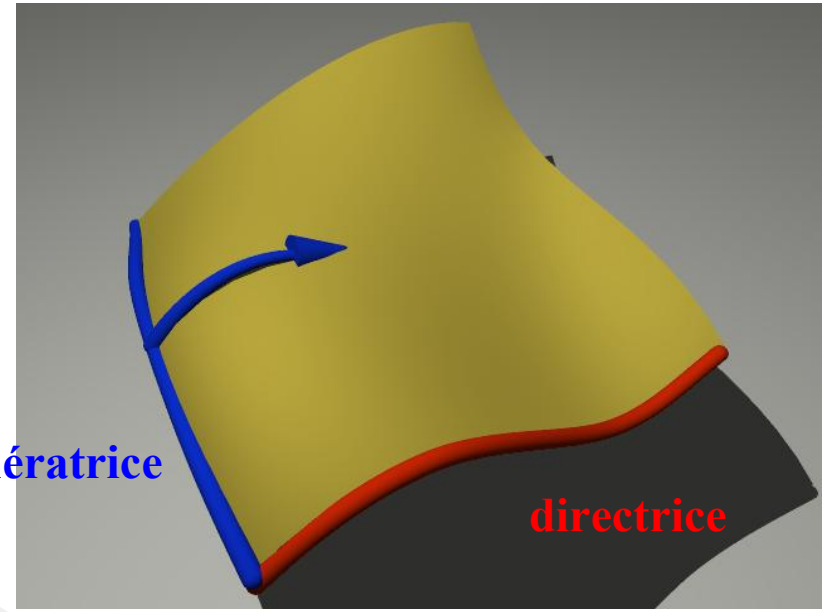
Les **(n+1)** points de contrôle en **u** donnent le degré **n** des courbes en u. Ici n = 3.

Les **(m+1)** points de contrôle en **v** donnent le degré **m** des courbes en v. Ici m = 3.

Produit tensoriel : principe

Une façon de construire surface paramétrique est de faire le produit tensoriel de deux courbes paramétriques. Une courbe $f_d(u)$ est appelée courbe directrice et l'autre courbe $f_g(v)$ est appelée courbe génératrice.

La surface est obtenue en déplaçant et déformant la courbe génératrice le long de la courbe directrice.



Carreaux de Bézier : produit tensoriel

Pour évaluer un point $p(u_0,v_0)$ sur la courbe, on effectue un produit tensoriel :

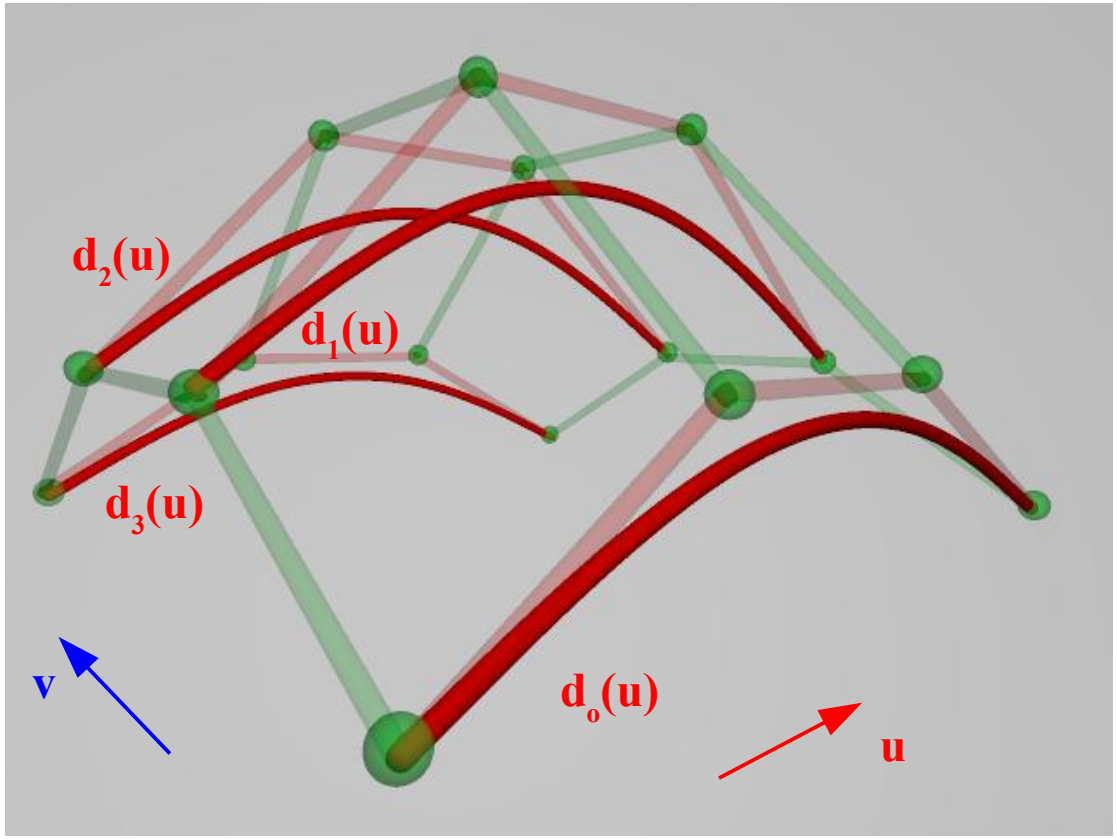
Si nous choisissons les directrices dans la direction de **u**

Il y a une directrice $d_j(u)$, (j=0..m) par polygone de contrôle dans la direction des u

Le polygone de contrôle de la directrice $d_j(u)$ est défini par les points de contrôle S_{ij} (i=0..n)

Une directrice est une courbe de Bézier définie par :

$$d_j(u) = \sum_{i=0}^{n} B_i^n(u) S_{ij}$$



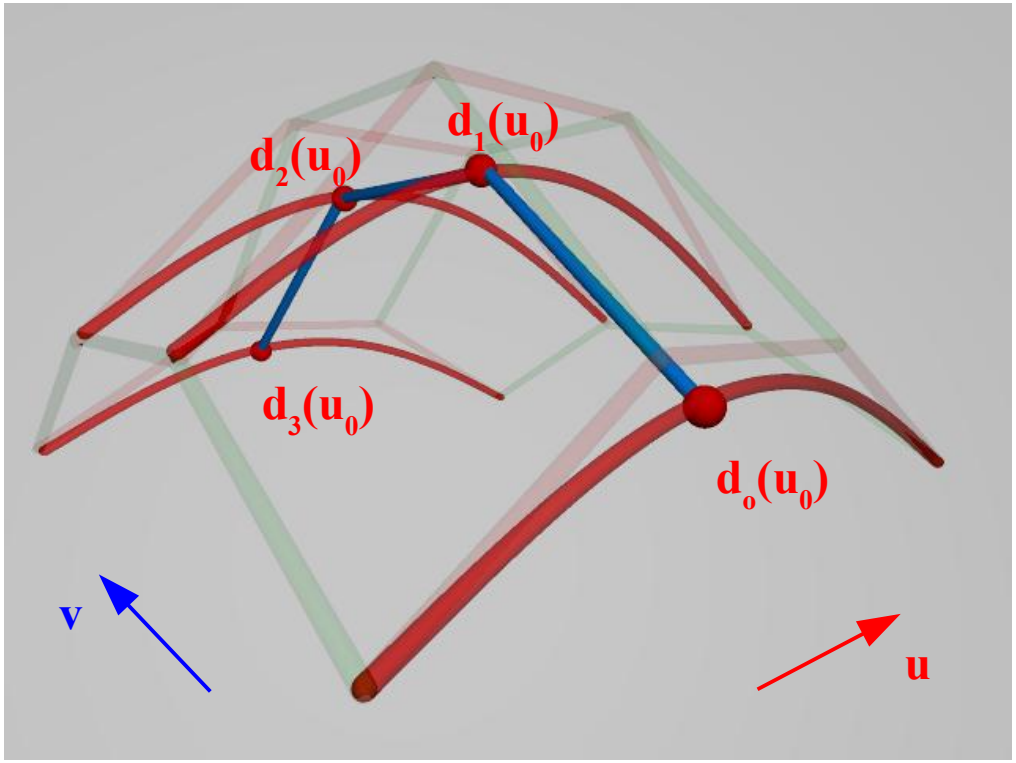
Carreaux de Bézier : produit tensoriel

Pour chaque directrice (en rouge), on évalue le point $d_j(u_0)$:

$$d_j(u_0) = \sum_{i=0}^{n} B_i^n(u_0)\, S_{ij} \quad , \qquad j = 0..m$$

Les $d_j(u_0)$ sont les sommets du polygone de contrôle (en bleu) de la courbe génératrice g(v) :

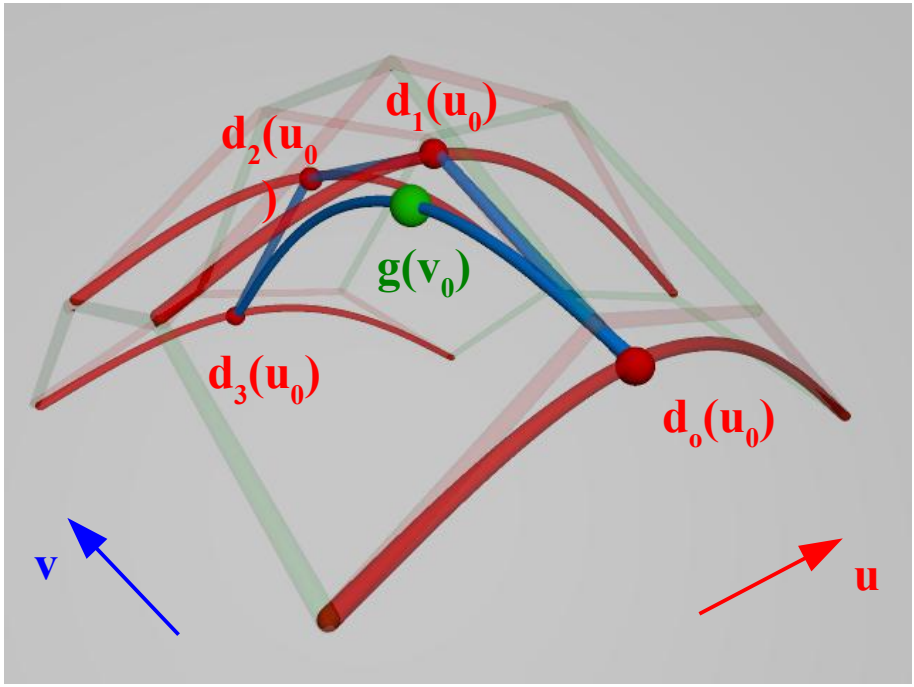
$$g(v) = \sum_{j=0}^{m} B_j^m(v)\, d_j(u_0)$$

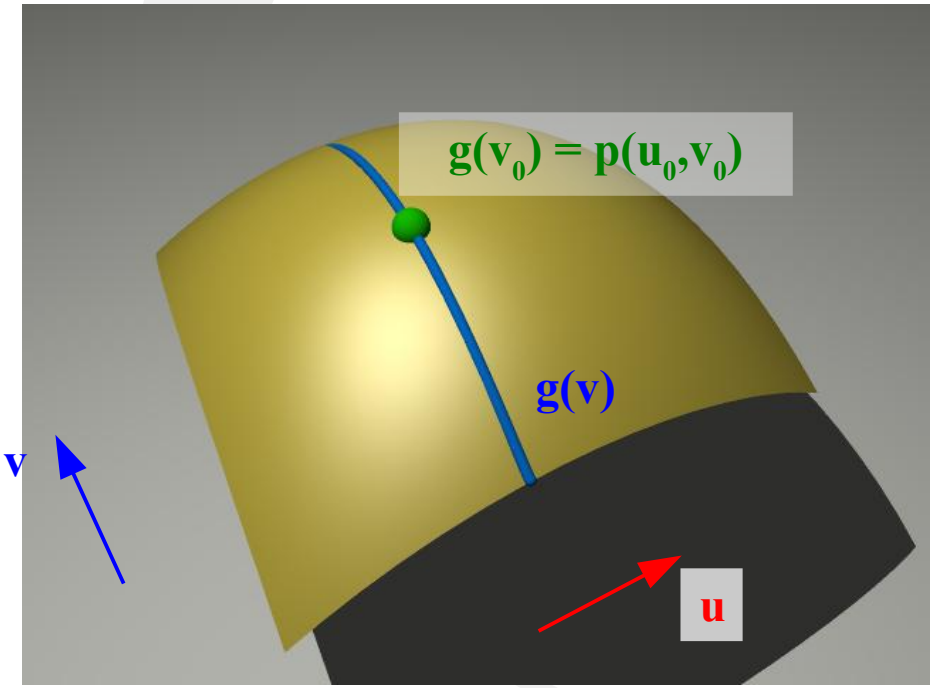


Carreaux de Bézier : produit tensoriel

La génératrice est sur la surface et le point de la surface $p(u_0,v_0)$ est alors celui de la courbe génératrice g(v) en $v=v_0$

$$g(v_0)=\sum_{j=0}^{m} B_j^m(v_0)\, d_j(u_0)$$





En reportant l'équation des $d_j(u)$ dans l'équation de g(v), on obtient l'équation du carreau de Bézier :

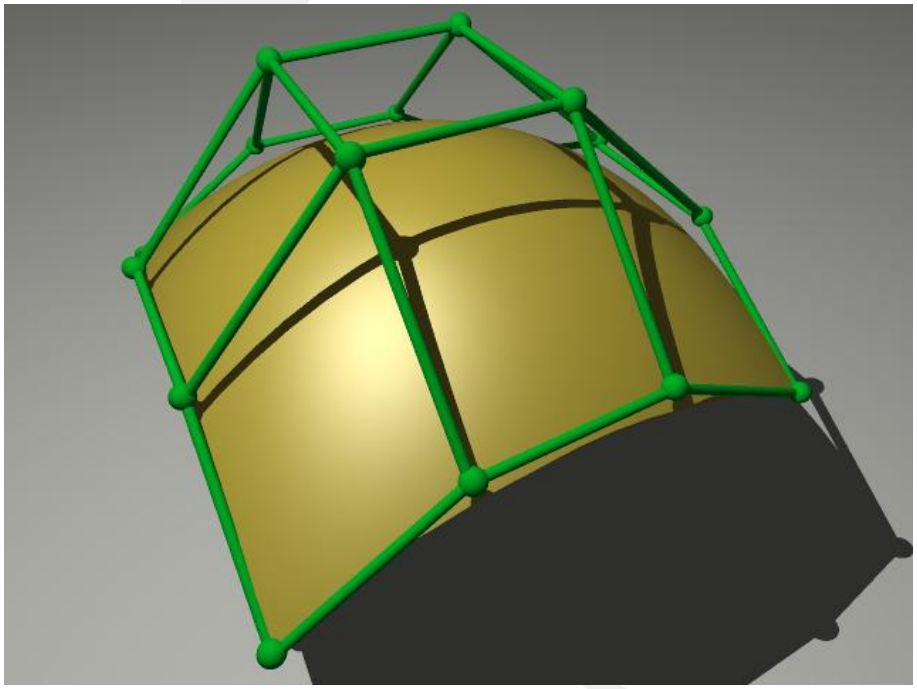
$$p(u,v)=g(v)=\sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u)\, B_j^m(v)\, S_{ij}$$

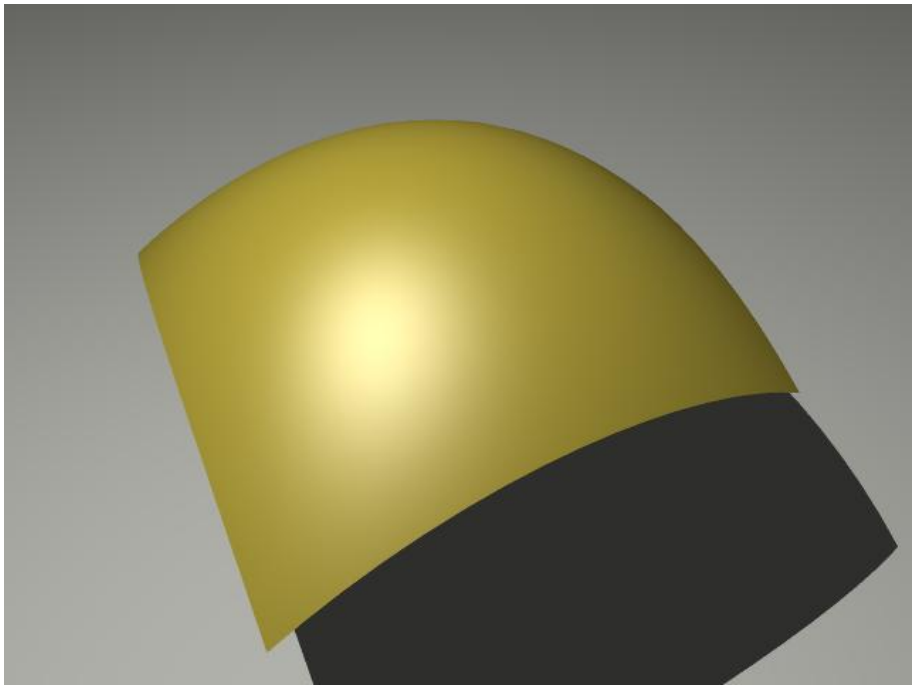
Carreaux de Bézier : définition

Un carreau de Bézier est défini à partir d'un maillage de contrôle et des polynômes de Bernstein de la façon suivante :

$$p(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u)\,B_j^m(v)\,S_{ij}$$

Exemple de carreau bi-cubique = produit tensoriel de deux courbes de degré 3 (ordre 4)





Les fonctions de base

Les carreaux de Bézier sont construits par produit tensoriel de courbes de Bézier.

Néanmoins, il est important de bien voir qu'ils sont obtenus par combinaison barycentrique de leurs points de contrôle :

$$p(u,v)=\sum_{i=0,j=0}^{n,m} B_{ij}^{n,m}(u,v)\,S_{ij} \quad , \qquad (u,v)\in[0,1]^2$$

Exercice : Soit un carreau de Bézier de degré 3 en u et de degré 2 en v

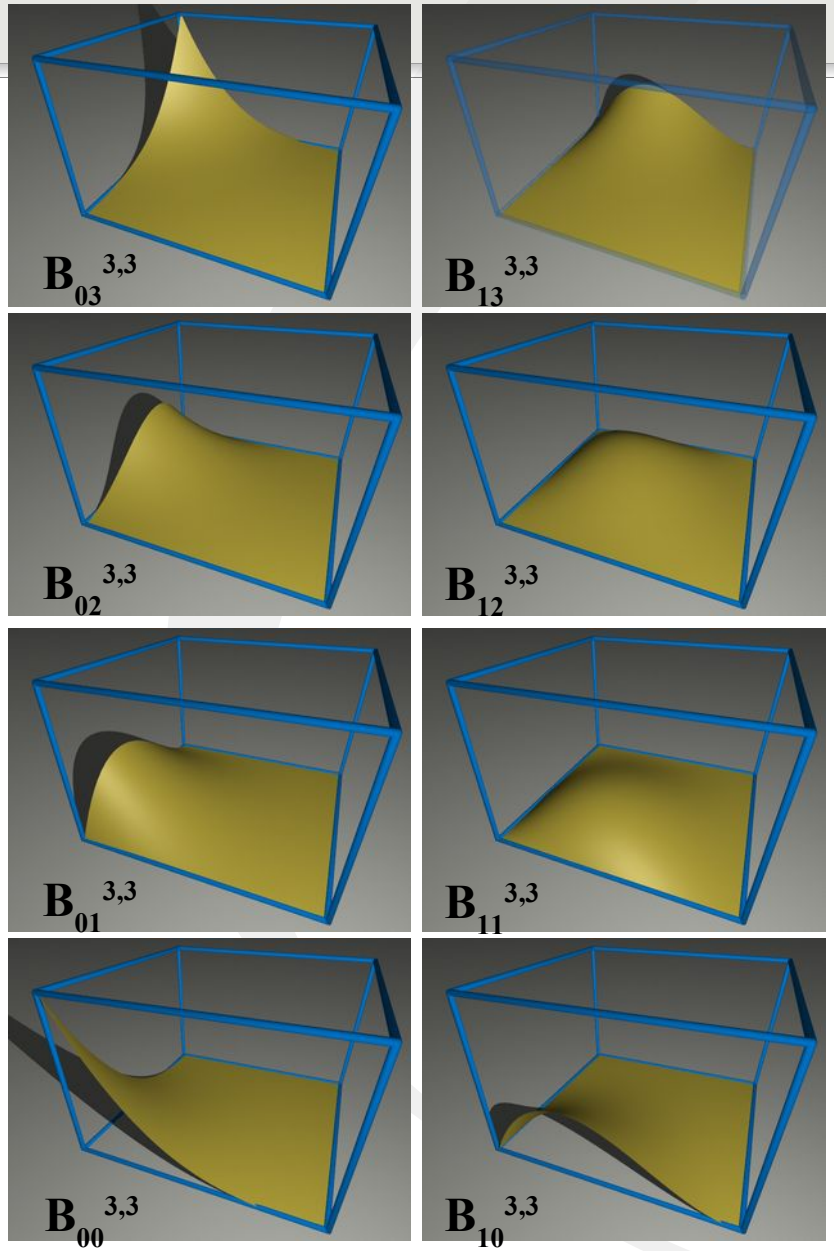
Combien de points de contrôle sont nécessaires pour définir ce carreau de Bézier ?

Combien de fonctions de base sont nécessaire ?

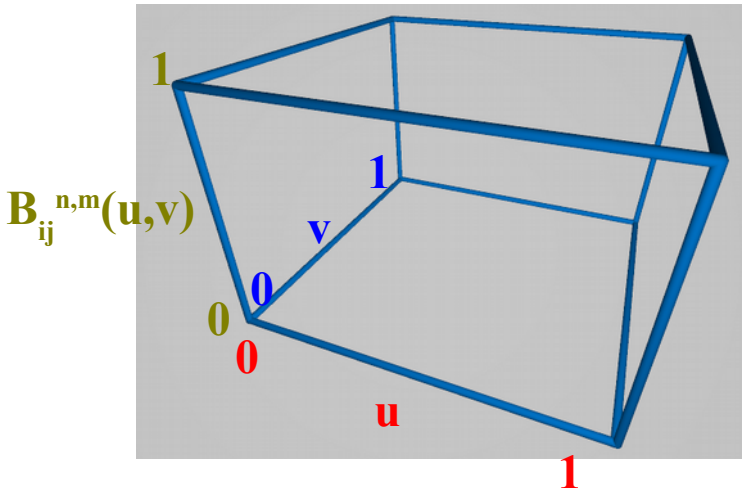
Exprimez $B_{ij}^{n,m}(u,v)$ en fonction de $B_i^n(u)$ et $B_j^m(v)$

Donnez l'équation de la fonction de base associée au point de contrôle S_{12}.

Fonctions de base pour un carreau de Bézier bi-cubique



Les 8 autres fonctions de base $B_{2j}^{3,3}$ et $B_{3j}^{3,3}$ sont obtenues par symétrie.



Modélisation Géométrique

Surfaces de subdivision

gael.guennebaud@inria.fr

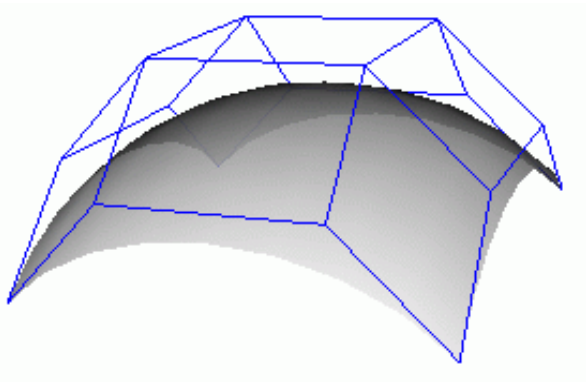
http://www.labri.fr/perso/guenneba/mg_2016

Surfaces paramétriques

Les carreaux de Bézier (1960)

Les surfaces splines (B-spline, NURBS,...)

standard pour la construction et l'animation



$$p(u,v)=\sum_{i=1}^{m+1}\sum_{j=1}^{n+1} N_i^k(u)\,N_j^l(v)\,P_{i,j}$$

\+

Modélisation par carreaux type couture

Chaque carreau est défini par un polygone de contrôle

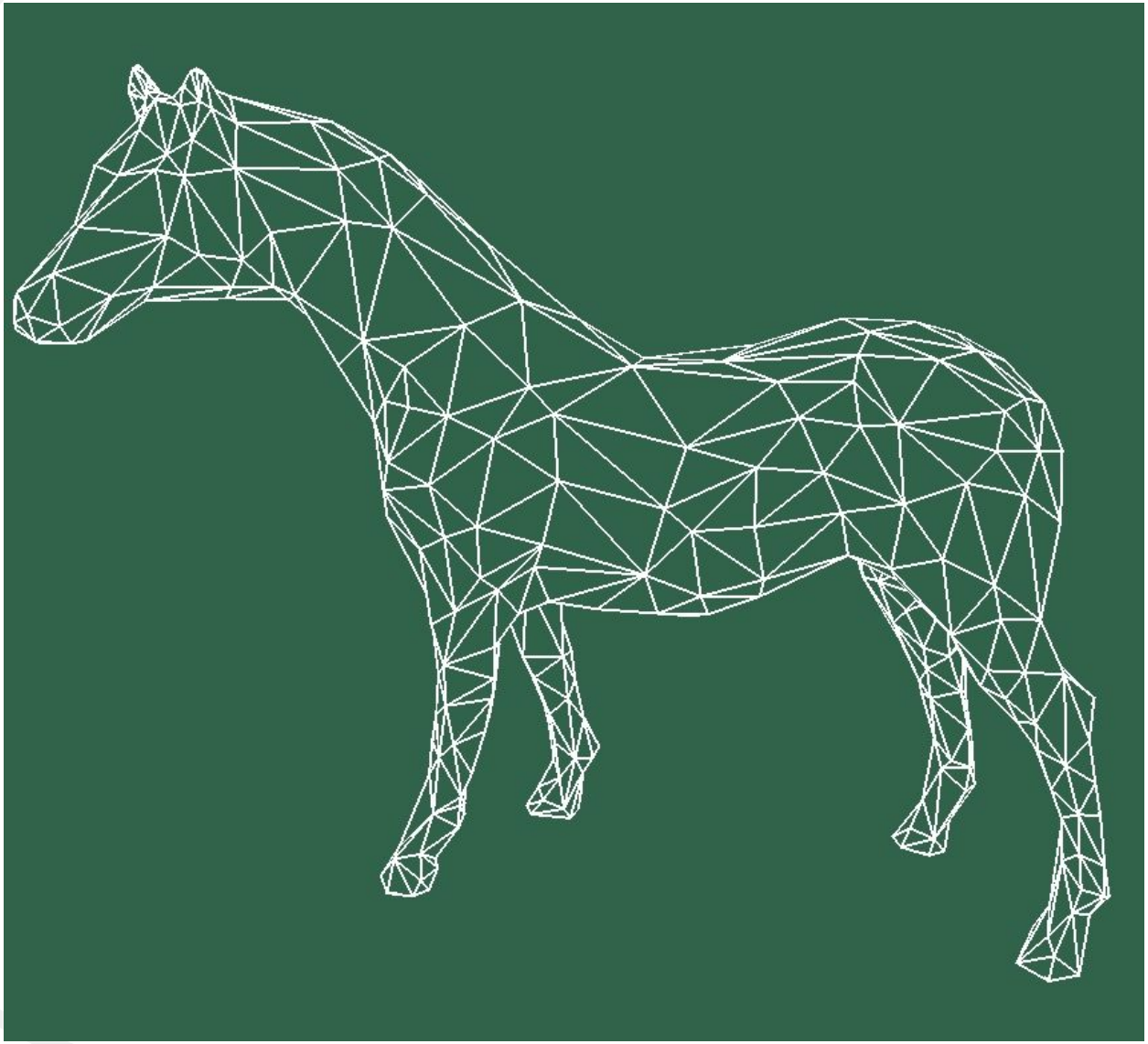
\-

Difficulté de contrôle des connections inter-carreaux

Difficulté d'ajuster la surface à un polygone de contrôle quelconque

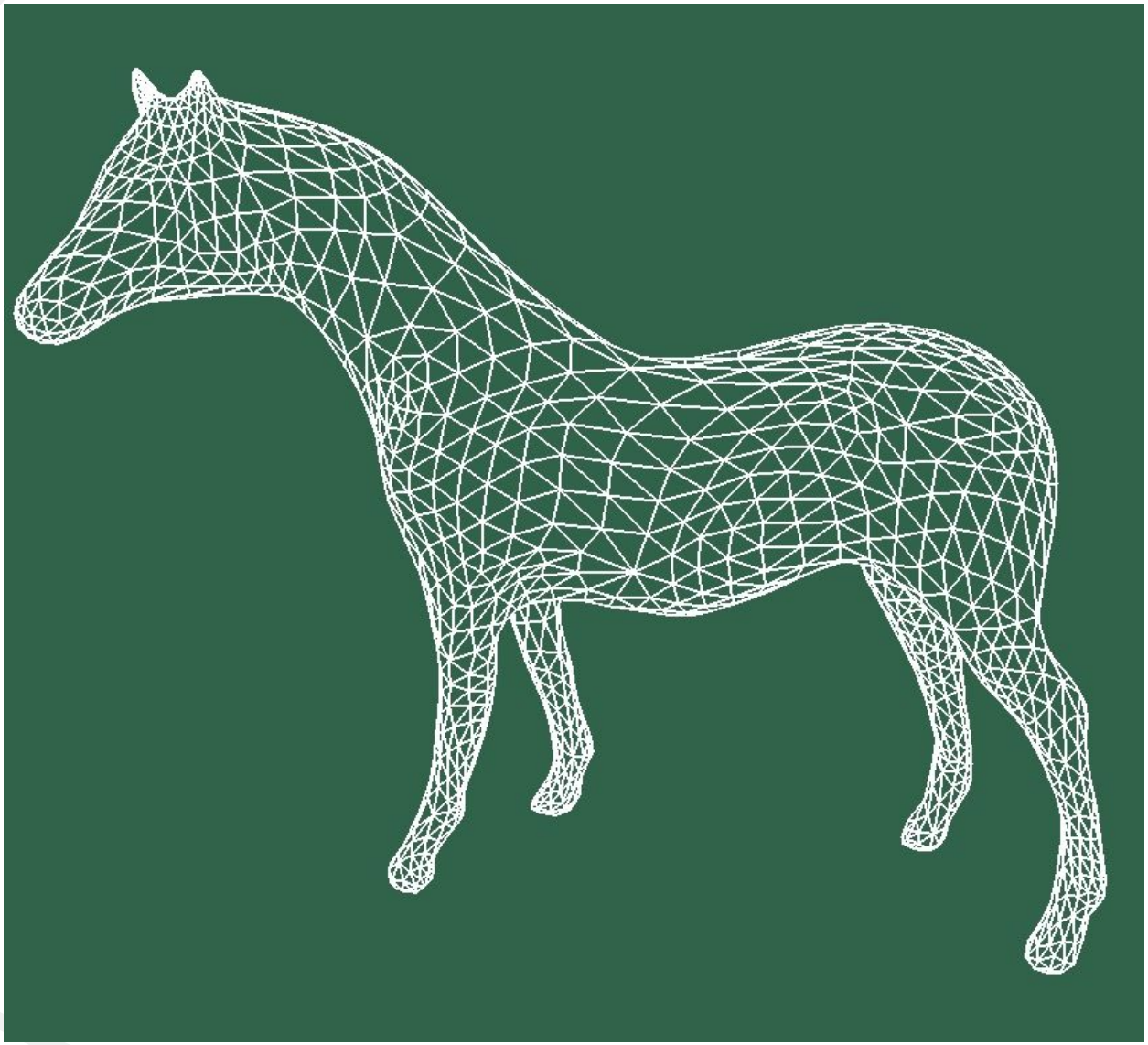
Exemple de subdivision

- **Maillage initial**



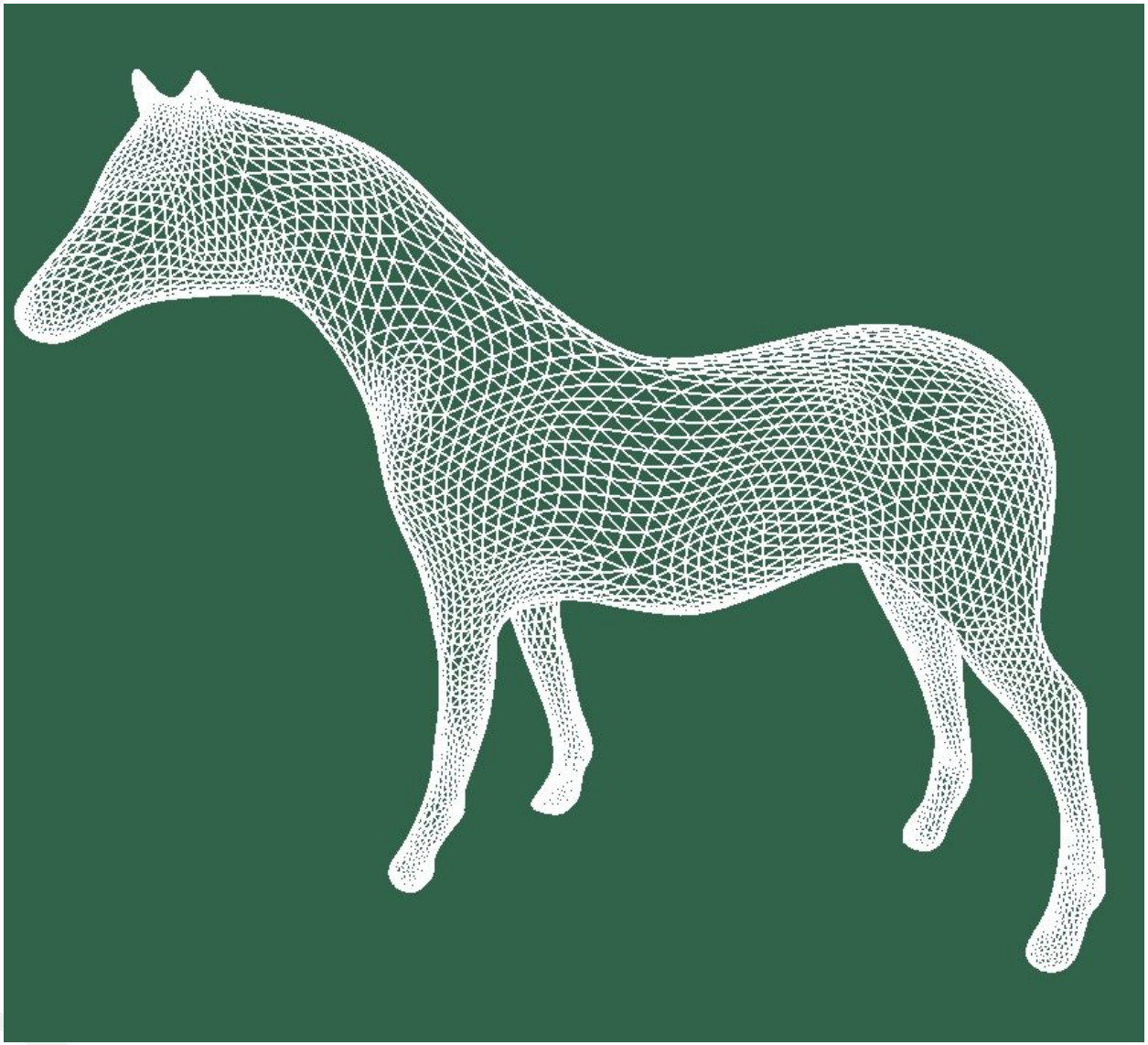
Exemple de subdivision

- **Le maillage après un pas de subdivision**



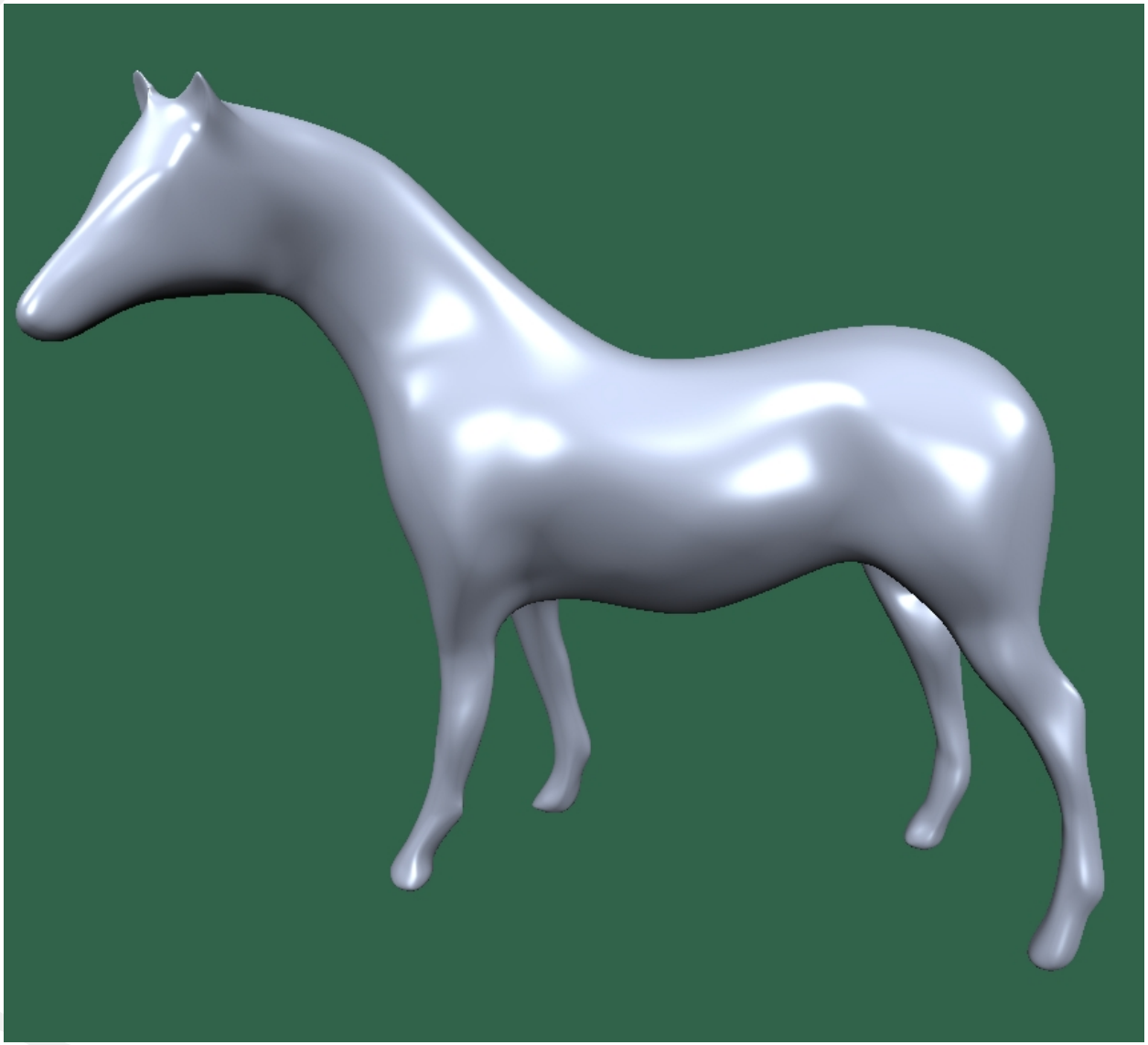
Exemple de subdivision

- **Le maillage après deux pas de subdivision**



Exemple de subdivision

- **La surface limite**



Un standard pour l'animation

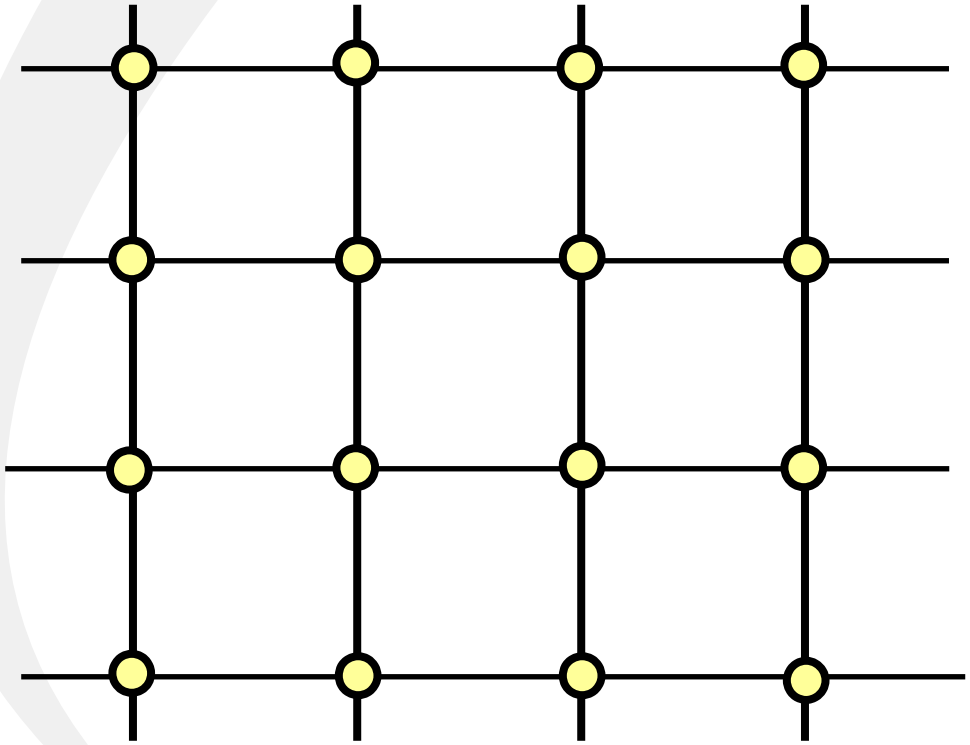
Pixar 1997 : Geri's game

- **Les NURBS sont abandonnées au profit des surfaces de subdivision**
  - Un maillage grossier est directement subdivisé pour produire une surface lisse
  - La multi-résolution est « gratuite » : un objet peut être représenté/modifié à différents niveaux de subdivision

- **Nombreuses autres utilisations**
  - représentation d'une fonction multivaluée sur un domaine de topologie non triviale
  - ex : résolution d'équa. diff. par la méthode des éléments finis

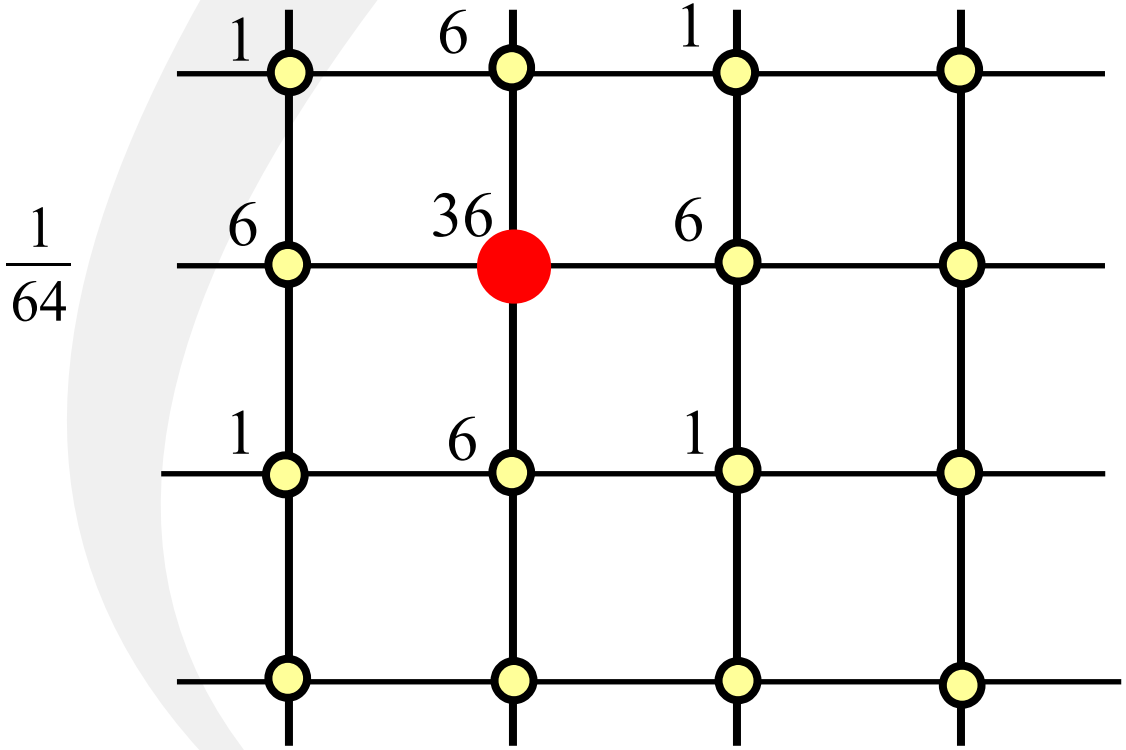
Principe de subdivision

- **Généralisation des surfaces paramétriques à des polygones de contrôle quelconque (1978)**



Principe de subdivision

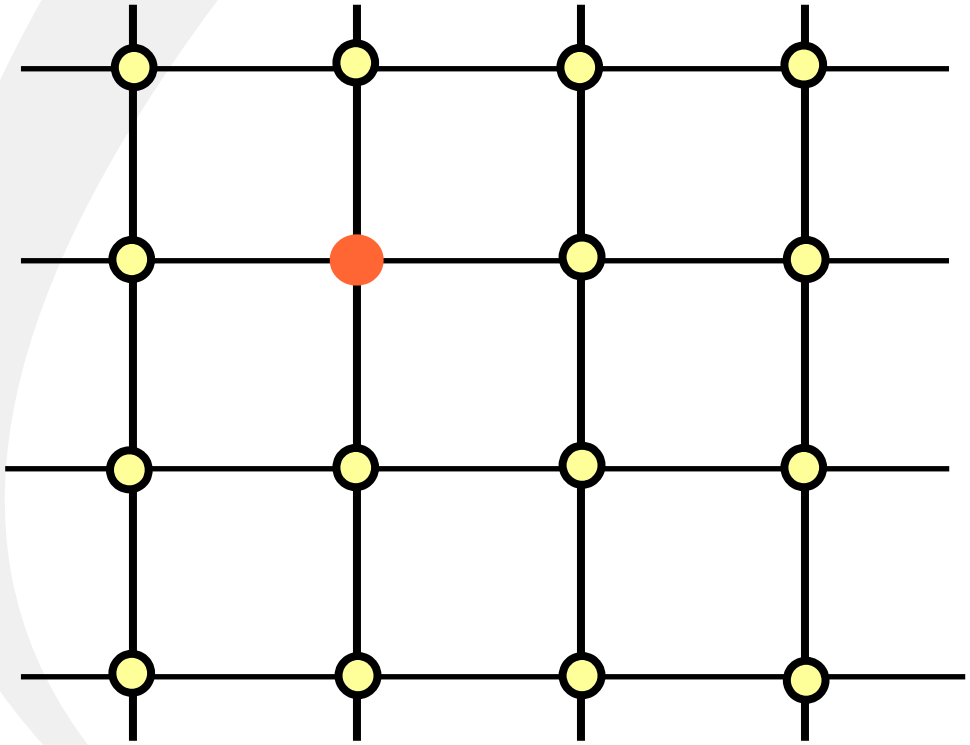
- **Généralisation des surfaces paramétriques à des polygones de contrôle quelconque (1978)**



- **Les sommets sont déplacés par une combinaison affine des sommets du maillage**

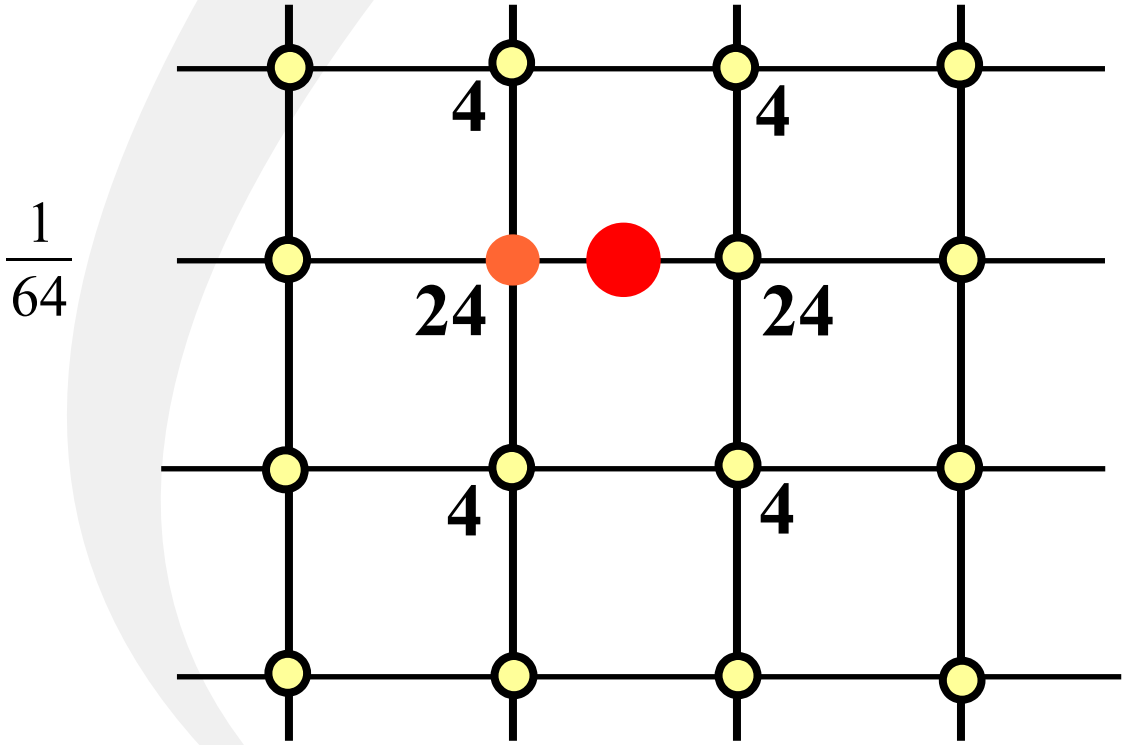
Principe de subdivision

- **Généralisation des surfaces paramétriques à des polygones de contrôle quelconque (1978)**



Principe de subdivision

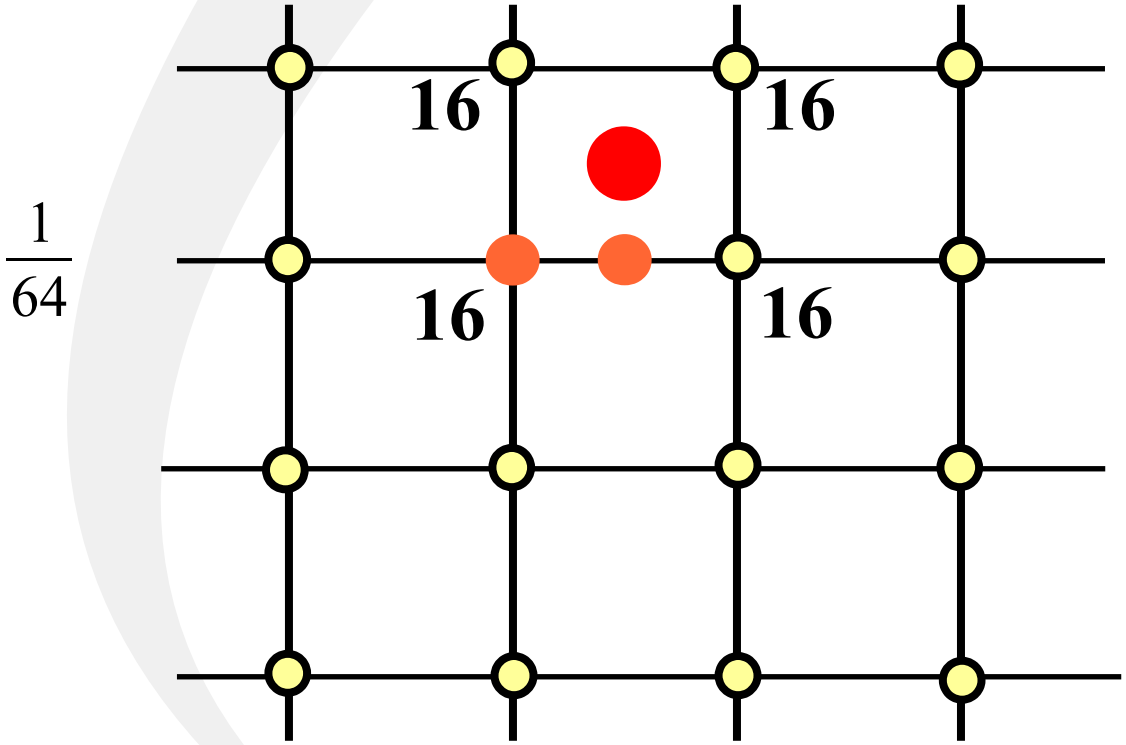
- **Généralisation des surfaces paramétriques à des polygones de contrôle quelconque (1978)**



- **Des nouveaux sommets sont ajoutés par une combinaison affine des sommets du maillage**

Principe de subdivision

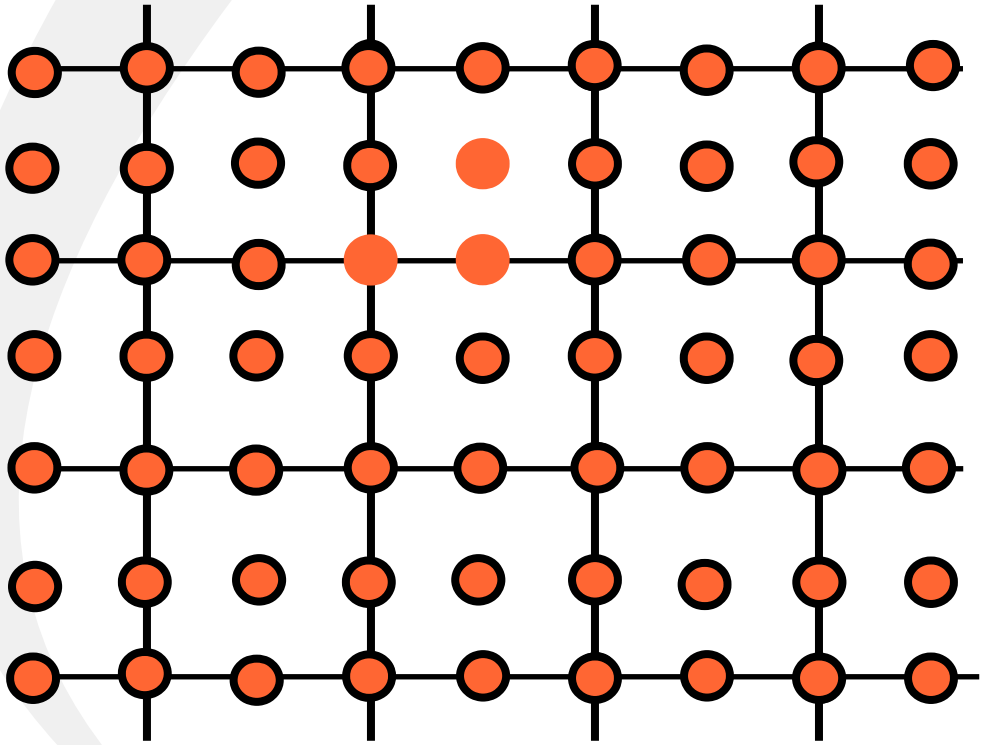
- **Généralisation des surfaces paramétriques à des polygones de contrôle quelconque (1978)**



- **Des nouveaux sommets sont ajoutés par une combinaison affine des sommets du maillage**

Principe de subdivision

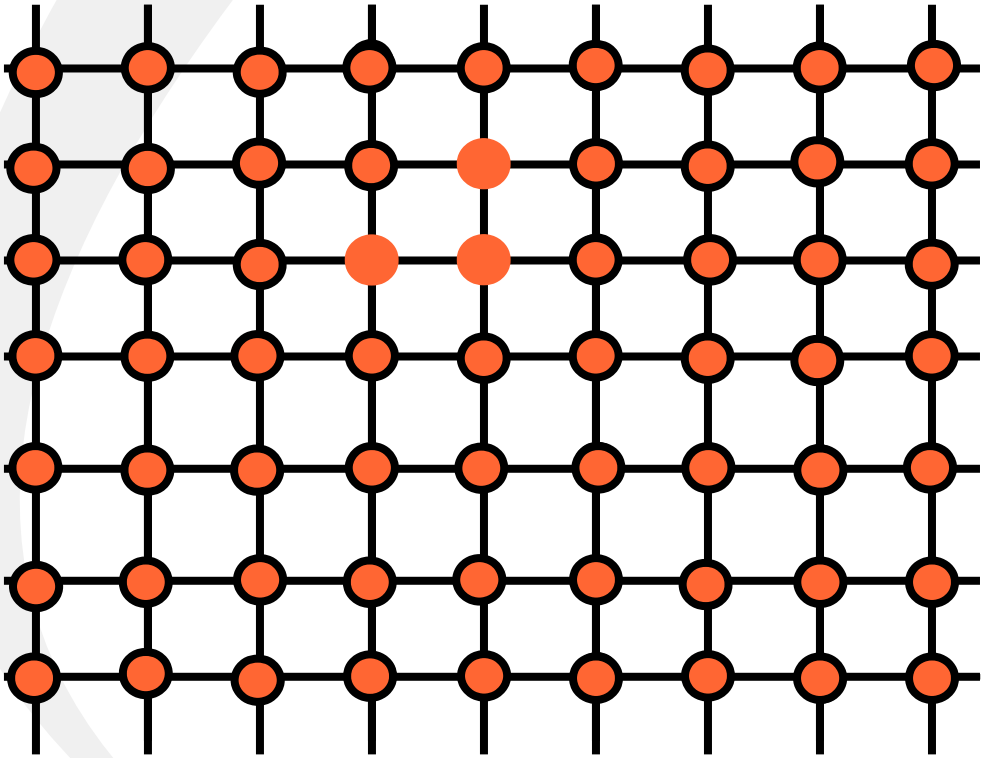
- **Généralisation des surfaces paramétriques à des polygones de contrôle quelconque (1978)**



- **Le processus de déplacement / insertion est répété sur l'ensemble du polygone initial**

Principe de subdivision

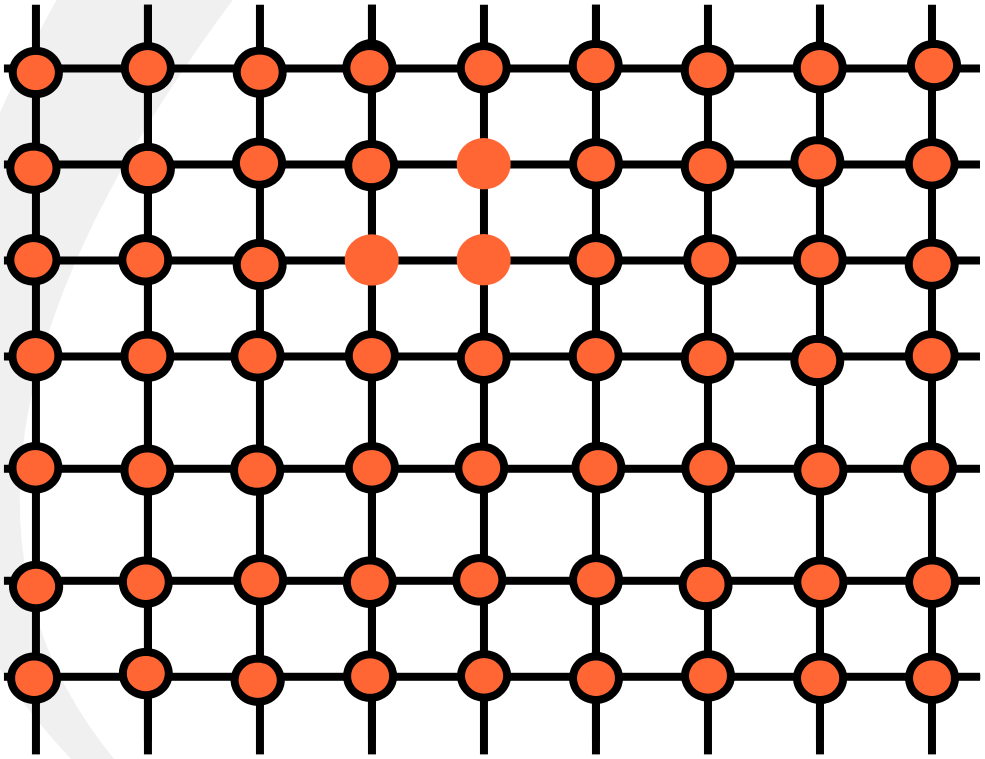
- **Généralisation des surfaces paramétriques à des polygones de contrôle quelconque (1978)**

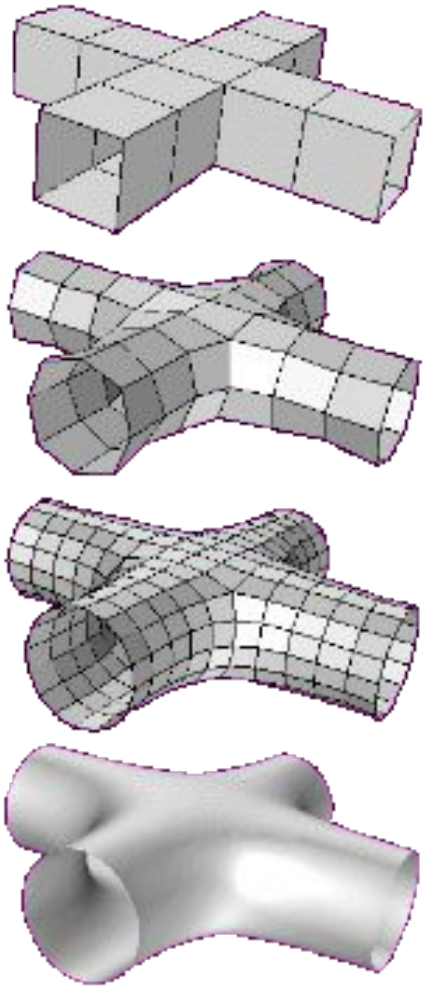


- **Les nouveaux points sont maillés**

Principe de subdivision

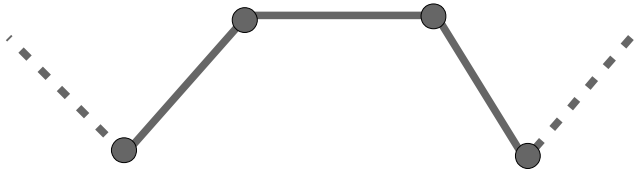
- **Généralisation des surfaces paramétriques à des polygones de contrôle quelconque (1978)**



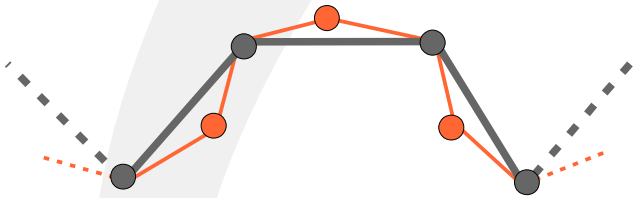


Exemple de subdivision

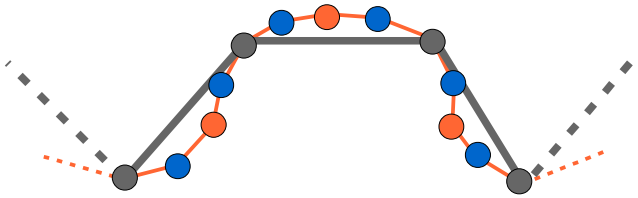
Polygone de contrôle



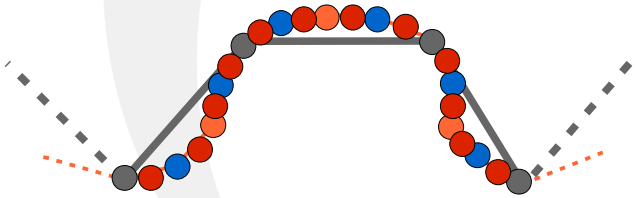
1 pas de subdivision :



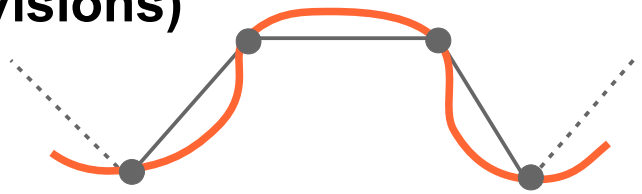
2 pas de subdivision



3 pas de subdivision



Courbe limite (après une infinité de subdivisions)



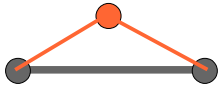
Exemple de subdivision

On peut dire :

A chaque pas de subdivision, un nouveau polygone ayant 2 fois plus de sommets que le précédent est créé.

Pour réaliser un pas de subdivision, chaque segment est coupé en deux puis les nouveaux points sont déplacés

insertion d'un nouveau point



déplacement du point inséré

Après une infinité de subdivisions les polygones convergent vers une **courbe limite**

Le polygone initial est un **polygone de contrôle** de la courbe limite

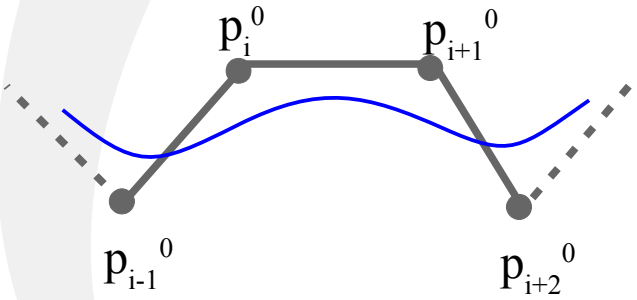
Chaque nouveau polygone est un polygone de contrôle de la courbe limite

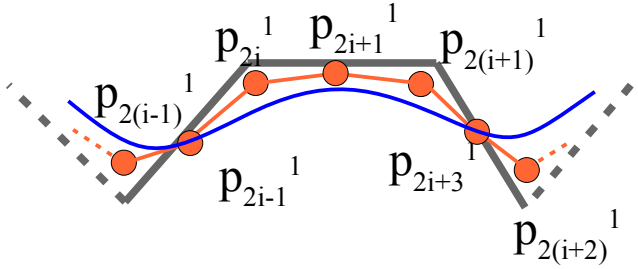
Les Box-spline

Dans le cas des courbes (un seul paramètres, ou spline univariée), les Box-spline ont les mêmes fonctions de base que les B-spline uniformes. Ainsi la seule différence entre courbes Box-spline et courbes B-spline, c'est que le vecteur nodal des courbes Box-spline est obligatoirement uniforme alors que celui des courbes B-spline peut être quelconque.

Subdivision et Box-spline

Une courbe Box-spline est définie par son polygone de contrôle. Appliquer un pas de subdivision dyadic revient créer un nouveau polygone de contrôle définissant la même courbe et ayant le double de points de contrôle.





A la limite, le polygone de contrôle subdivisé converge vers la courbe Box-spline.

Subdivision et Box-spline - Exemples

- **k=3**

$$p(u) = \sum_{i=1}^{n} N_i^3(u)\, p_i^0 = \ldots$$

$$= \sum_{i=1}^{n} N_{2i}^3(2u) \underbrace{\left(\quad ? \quad \right)}_{p_{2i}^1} + N_{2i+1}^3(2u) \underbrace{\left(\quad ? \quad \right)}_{p_{2i+1}^1}$$

$$N_i^k(u) = \frac{1}{2^{k-1}} \sum_{l=0}^{k} \binom{k}{l} N_{2i+l}^k(2u)$$

0:				1				
1:				1	1			
2:			1	2	1			
3:			1	3	3	1		
4:		1	4	6	4	1		
5:		1	5	10	10	5	1	
6:	1	6	15	20	15	6	1	
7:	1	7	21	35	35	21	7	1
8:	1	8	28	56	70	56	28	8	1

Subdivision et Box-spline - Exemples

- **k=3**

$$p(u) = \sum_{i=1}^{n} N_i^3(u)\, p_i^0 = \ldots$$

$$= \sum_{i=1}^{n} N_{2i}^3(2u)\frac{3\,p_{i-1}^0 + p_i^0}{4} + N_{2i+1}^3(2u)\frac{p_{i-1}^0 + 3\,p_i^0}{4}$$

$$N_i^k(u) = \frac{1}{2^{k-1}} \sum_{l=0}^{k} \binom{k}{l} N_{2i+l}^k(2u)$$

```
0:                    1
1:                  1   1
2:                1   2   1
3:              1   3   3   1
4:            1   4   6   4   1
5:          1   5  10  10   5   1
6:        1   6  15  20  15   6   1
7:      1   7  21  35  35  21   7   1
8:    1   8  28  56  70  56  28   8   1
```

Subdivision et Box-spline - Exemples

- **k=3**

$$p(u) = \sum_{i=1}^{n} N_i^3(u)\, p_i^0 = \ldots$$

$$= \sum_{i=1}^{n} N_{2i}^3(2u)\frac{3\,p_{i-1}^0 + p_i^0}{4} + N_{2i+1}^3(2u)\frac{p_{i-1}^0 + 3\,p_i^0}{4}$$

$$N_i^k(u) = \frac{1}{2^{k-1}} \sum_{l=0}^{k} \binom{k}{l} N_{2i+l}^k(2u)$$

0:				1				
1:				1	1			
2:			1	2	1			
3:			1	3	3	1		
4:		1	4	6	4	1		
5:		1	5	10	10	5	1	
6:	1	6	15	20	15	6	1	
7:	1	7	21	35	35	21	7	1
8:	1	8	28	56	70	56	28	8	1

- **k=4**

$$p(u) = \sum_{i=1}^{n} N_i^4(u)\, p_i^0 = \ldots$$

$$= \sum_{i=1}^{n} N_{2i}^4(2u)\frac{p_{i-1}^0 + 6\,p_i^0 + p_{i+1}^0}{8} + N_{2i+1}^4(2u)\frac{4\,p_i^0 + 4\,p_{i+1}^0}{8}$$

Des courbes aux surfaces
- les schémas les plus utilisés -

Catmull-Clark : maillages de quadrilatères ou quelconques, approximant, continuité C^2

Loop : maillages triangulaires, approximant, continuité C^2

Butterfly : maillages triangulaires, interpolant, continuité C^1

Aux sommets irréguliers, tous ces schémas sont de continuité C^1

Schéma de Catmull-Clark

C'est le produit tensoriel du schéma de subdivision convergeant vers la Box-spline univariée de degré 3. C'est aussi le schéma convergeant vers la Box-spline bi-directionnelle à 8 directions (4 sur chaque axe) sur un maillage quadrilatéral. Enfin c'est le schéma convergeant vers la B-spline uniforme, cubique, bivariée (= bidirectionnelle). C'est un schéma dyadic (continuité C^2).

Masque de subdivision de la Box-spline univariée de degré 3 :

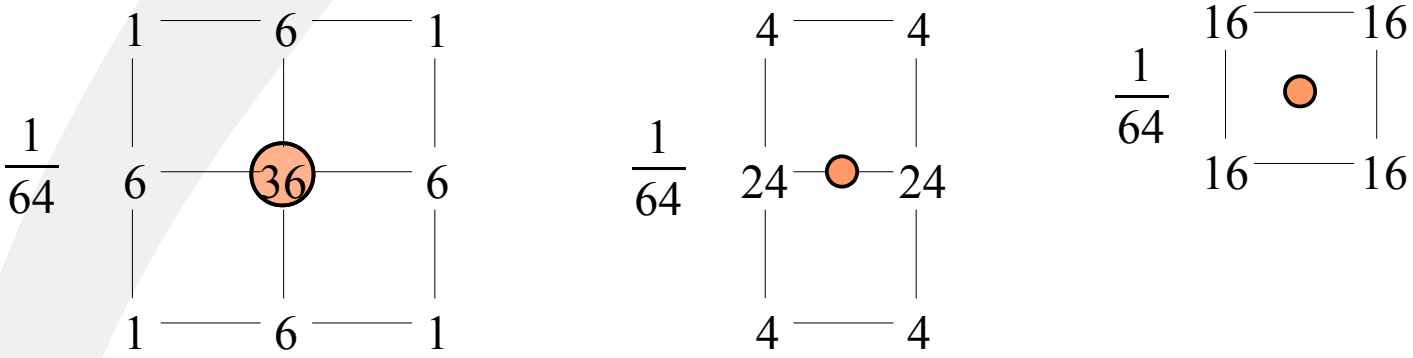
$$\frac{1}{8}\begin{bmatrix}1 & 4 & 6 & 4 & 1\end{bmatrix}$$

Masque du produit tensoriel :

$$\frac{1}{64}\begin{bmatrix}1 & 4 & 6 & 4 & 1\\ 4 & 16 & 24 & 16 & 4\\ 6 & 24 & 36 & 24 & 6\\ 4 & 16 & 24 & 16 & 4\\ 1 & 4 & 6 & 4 & 1\end{bmatrix}$$

Catmull-Clark : règles de subdivision

D'où les règles de subdivision (stencils) pour le cas régulier :



Afin de s'adapter à un maillage quelconque, le schéma est appliqué avec la formule suivante :

Les points de face sont les équibarycentres des sommets de chaque face originelle

Les points d'arête sont les équibarycentres des deux sommets de l'arête et des deux nouveaux points de face des deux faces adjacentes à l'arête

Les anciens sommets de valence **n** sont replacés selon l'équation suivante :

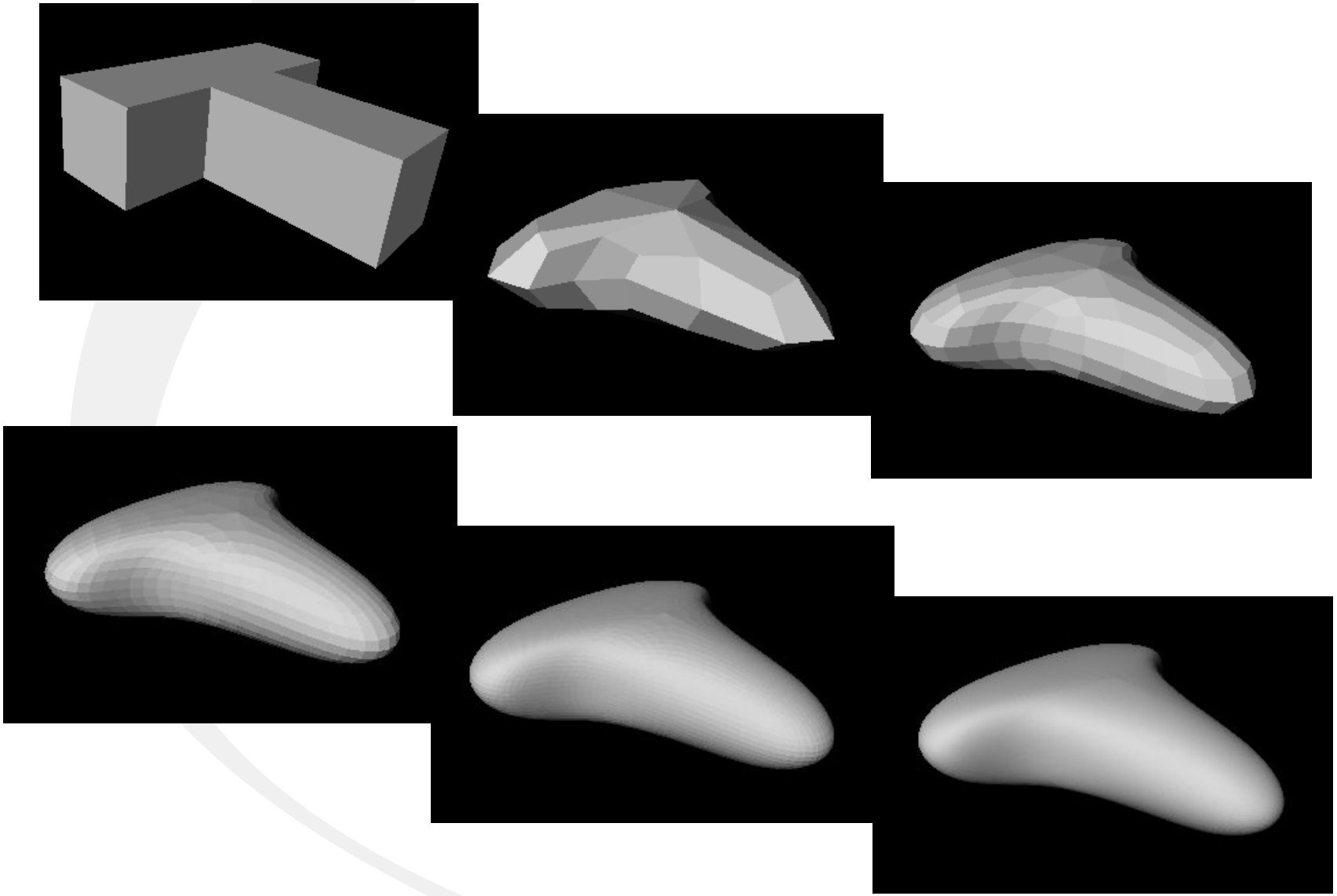
$$\frac{1}{n}\left(Q+2\mathrm{R}+S(n-3)\right)$$

Q est l'équibarycentre des nouveaux points de face des faces passant par l'ancien sommet

R est l'équibarycentre des milieux des arêtes passant par l'ancien sommet

S est l'ancien sommet

Catmull-Clark : exemple



Exercices

Montrez que les règles dépendantes du nombre *n* d'arêtes sortantes du sommet subdivisé (valence) sont équivalentes aux règles de subdivision classiques quand *n=4*.

Appliquez Catmull-Clark sur le maillage suivant :

Que remarquez vous sur le nombre de

côté des faces après un pas de subdivision ?

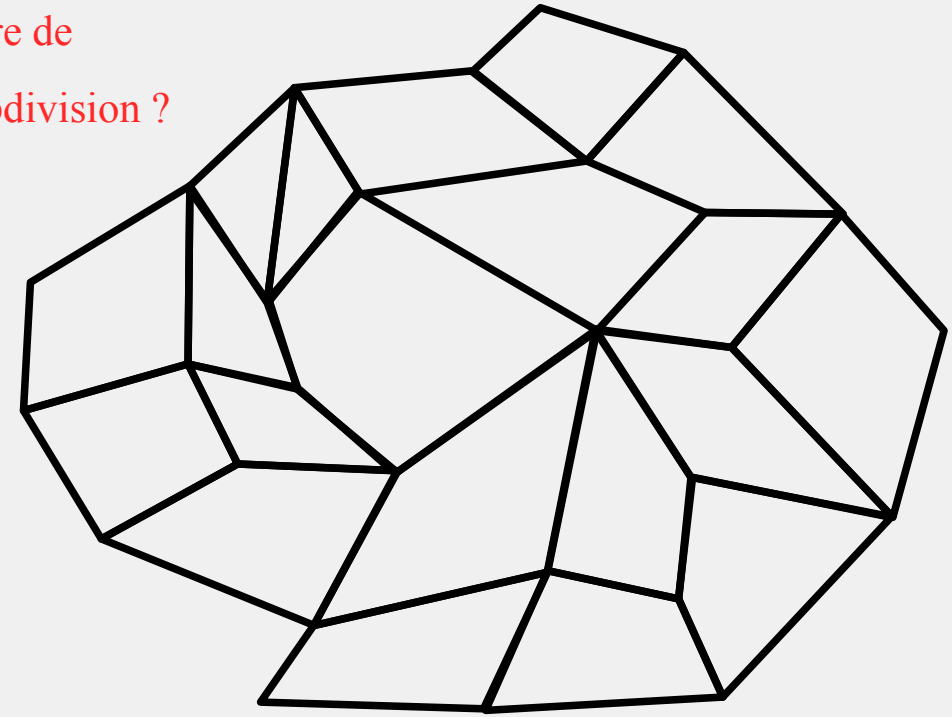
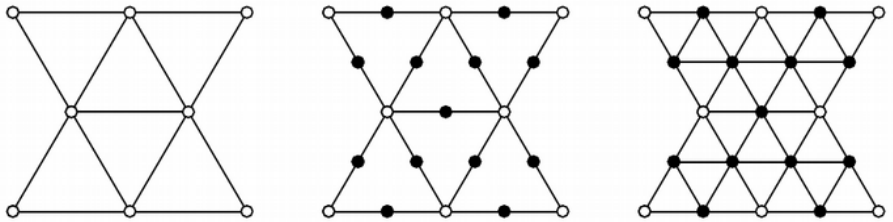
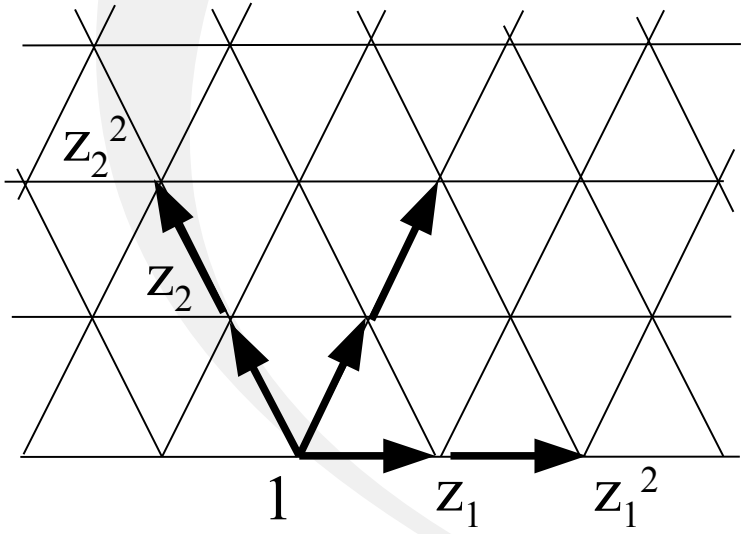


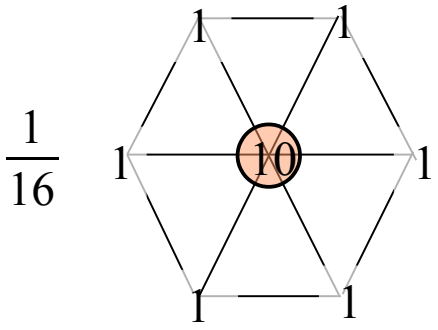
Schéma de Loop

- **Pour les maillages triangulaire**
  - schéma dyadique et approximant

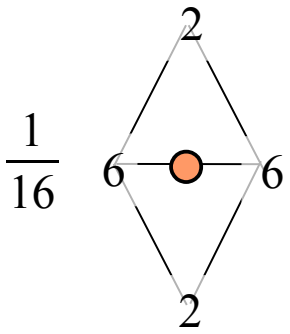


  - Box-spline cubique à trois directions (continuité C^2)





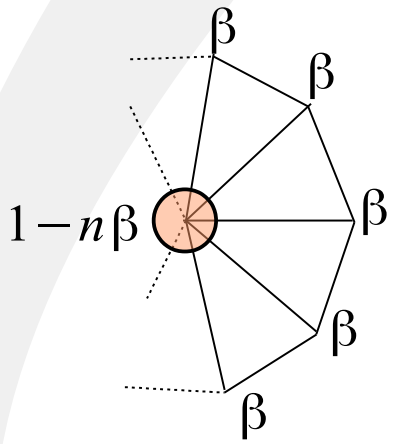
anciens sommets



insertion

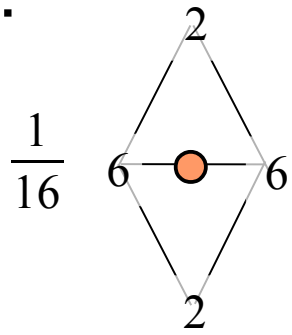
Loop aux sommets irréguliers

Déplacement d'un sommet de valence n :



$$\beta = \frac{1}{n}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\left(\frac{2\pi}{n}\right)\right)^2\right)$$

La règle d'insertion reste identique :

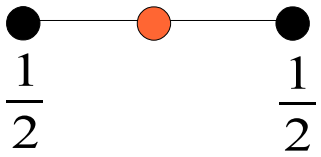


Arêtes franches pour Loop et Catmull-Clark

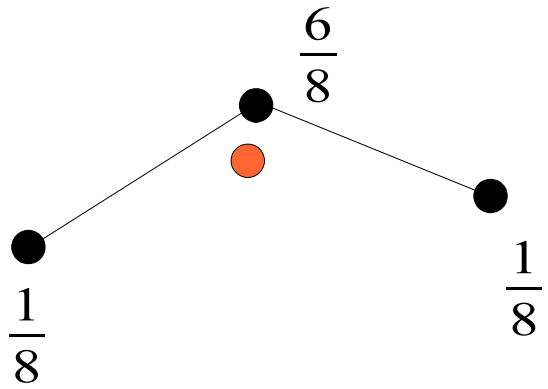
Arêtes : plis ou bords

On applique le schéma de la Box-spline cubique univariée dont les règles de subdivision sont les suivantes :

Insertion



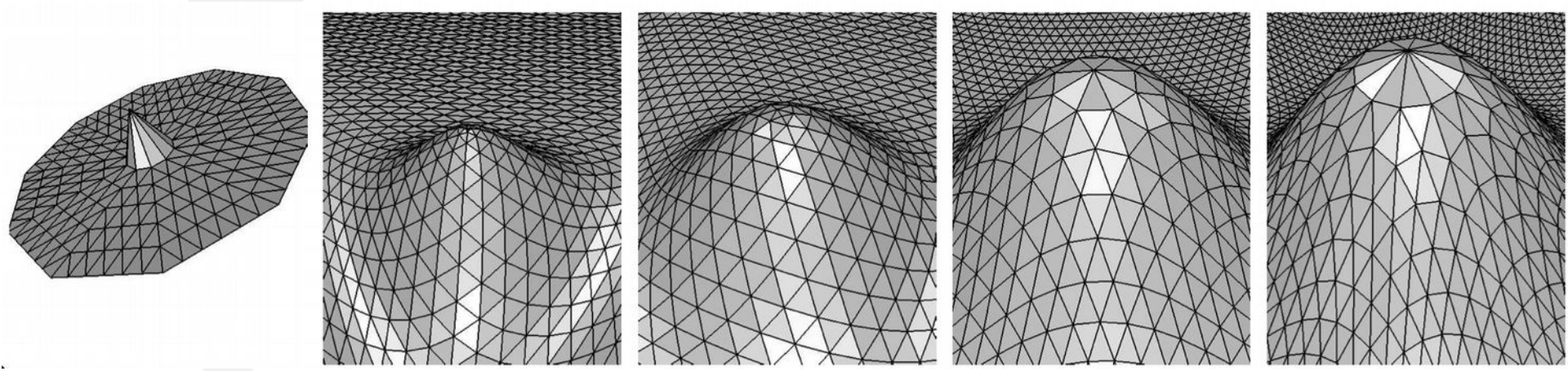
Déplacement



Exercice

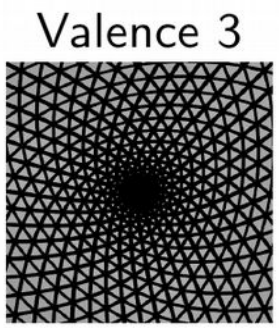
- **Comment évaluer/visualiser les fonctions de bases sous-jacente ?**

  → démo Blender

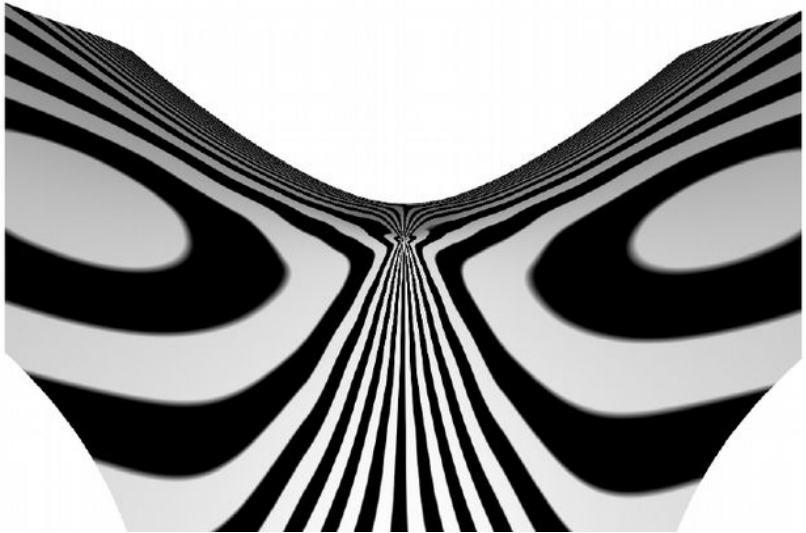


Sommets extraordinaires

- **Règles pour les sommets réguliers uniquement**
  - pour les sommets non réguliers → heuristiques
  - => nombreux artéfacts

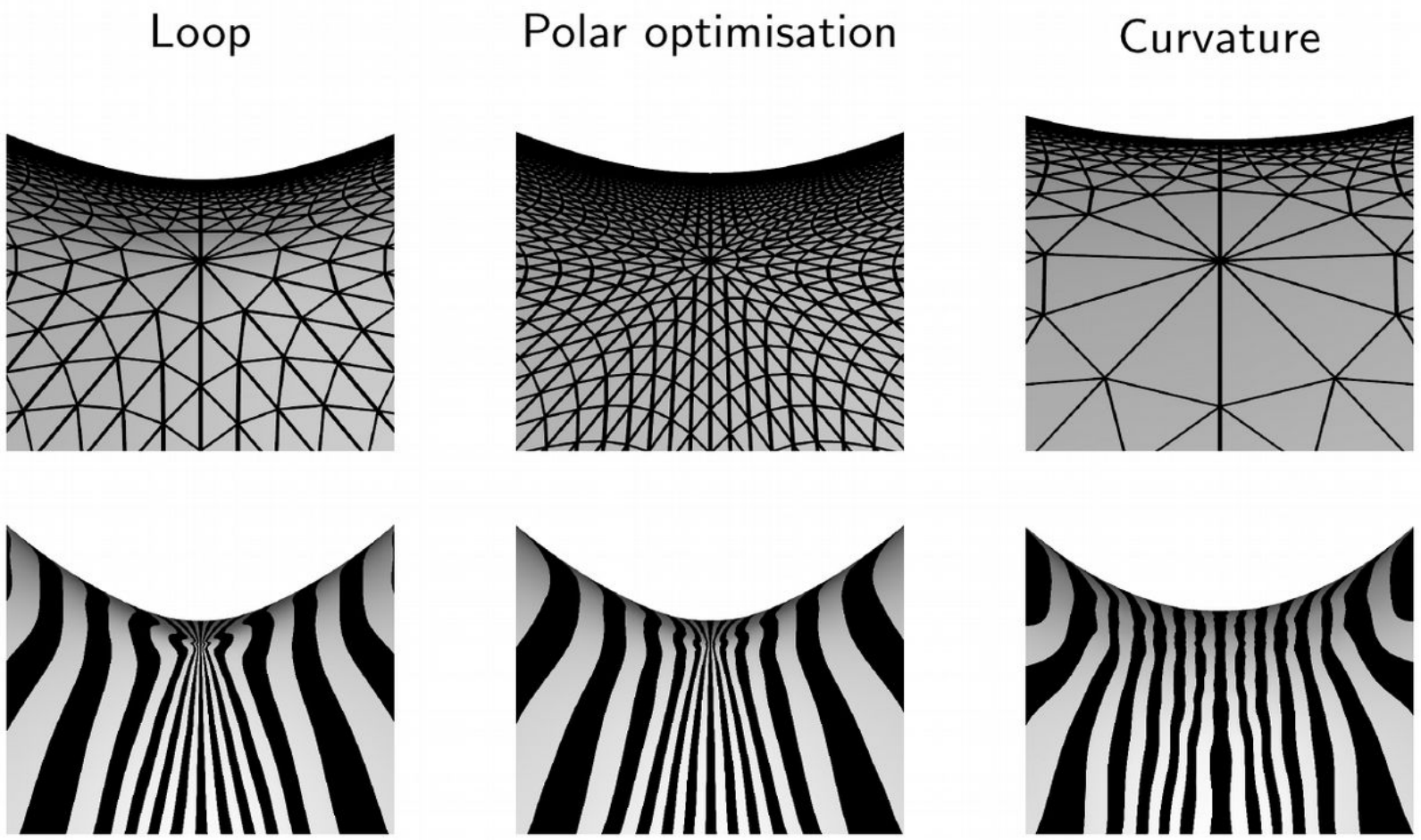


Saddle with reflexion lines valence 12



Sommets extraordinaires

- **Heuristiques :**



LOD
(Level Of Details)

LOD : Objectif

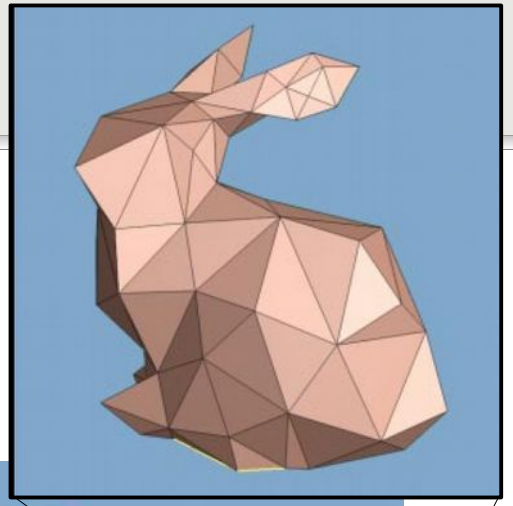
- **Observations :**
  - objet détaillé → énormément de polygones
  - coût de rendu = fonction du nombre de polygones
  - taille des polygones en espace image = fonction de la distance à l'observateur
    → objet lointain → nombreux micro polygones se projetant sur un même pixel

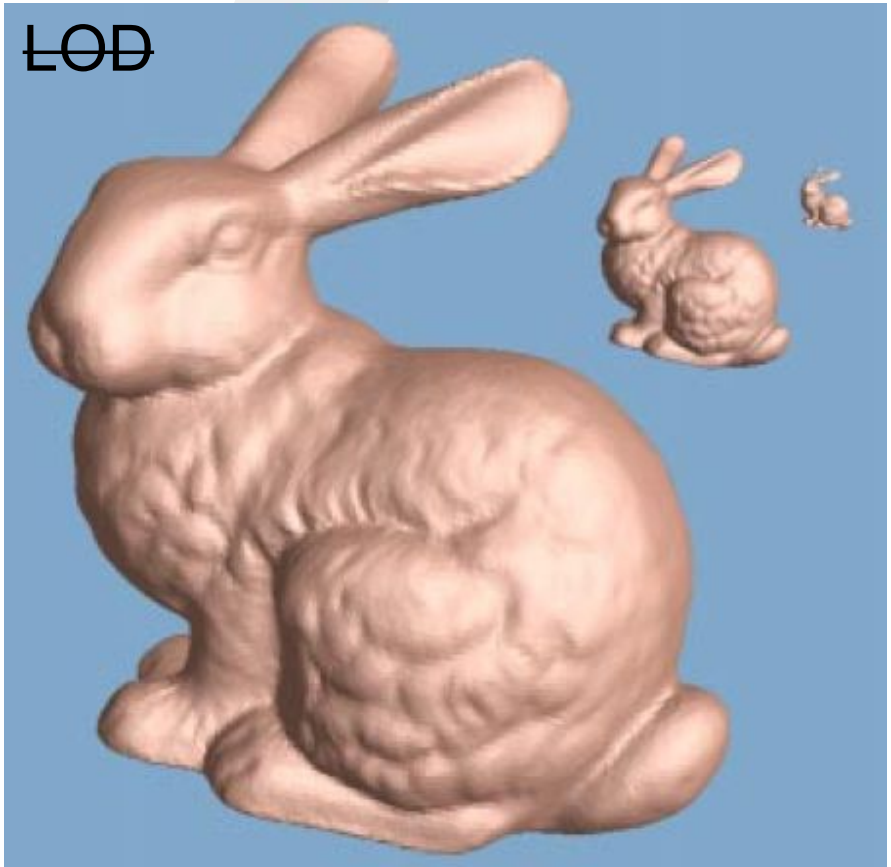
- **Idée :**
  - adapter la résolution (le nombre de polygones) du maillage en fonction du point de vue

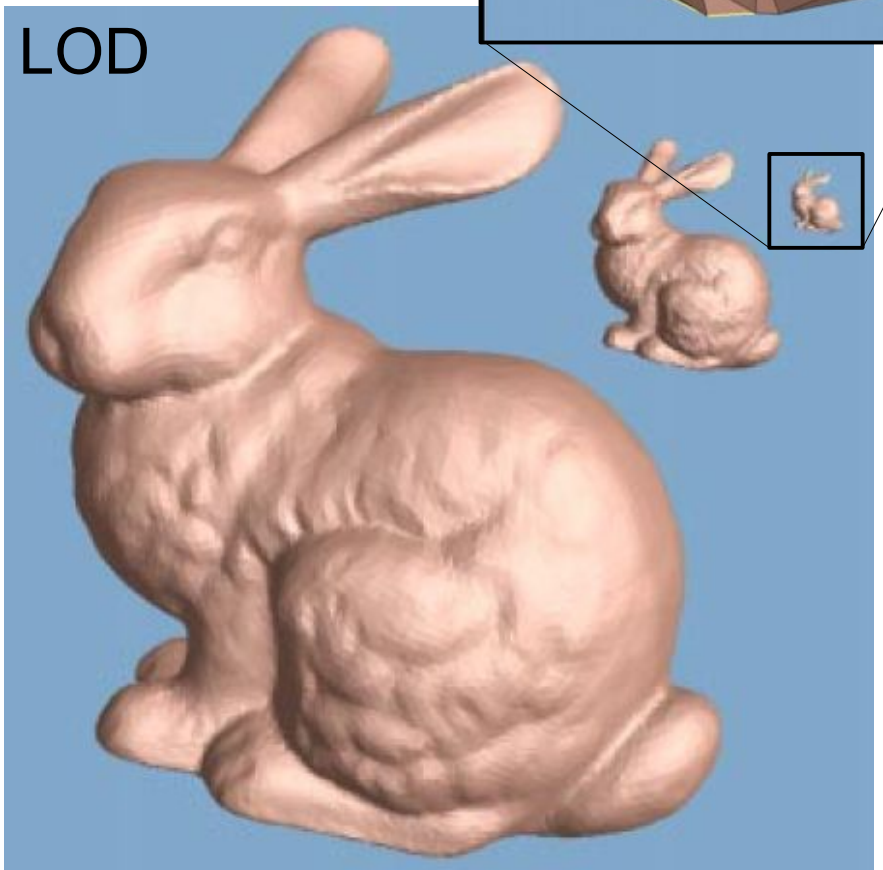
- **Défis :**
  - comment calculer des versions simplifiées du maillage ?
  - comment choisir la résolution adaptée ?
  - comment rendre cela efficace du point de vu du GPU ?

LOD discret

- **Différentes représentations du même objet sont calculées, chacune à un niveau de détail différent. Pendant l'exécution de l'application, une représentation de l'objet est sélectionnée et visualisée.**

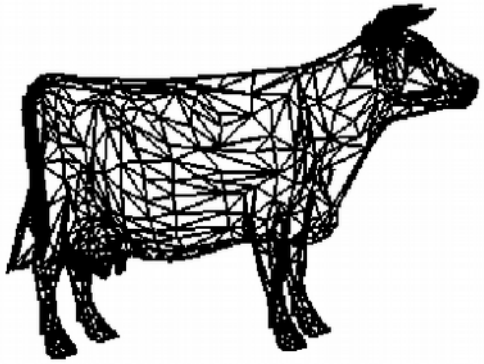




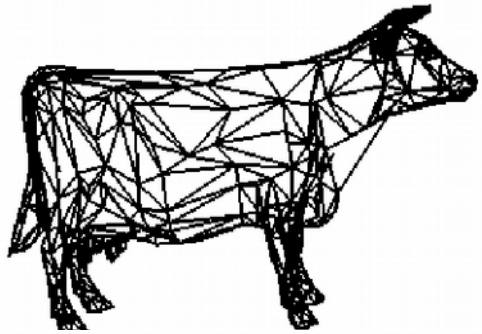


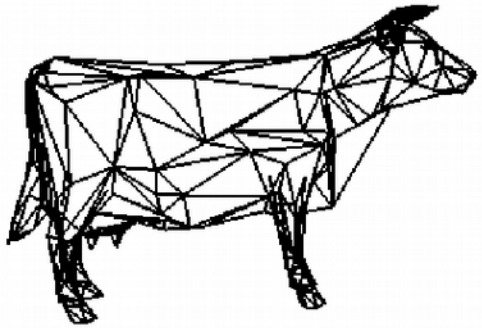
LOD discret

- **Précalculer les niveaux de détails**
  - surfaces de subdivision : trivial !
  - maillage quelconque :
    → simplification de maillage



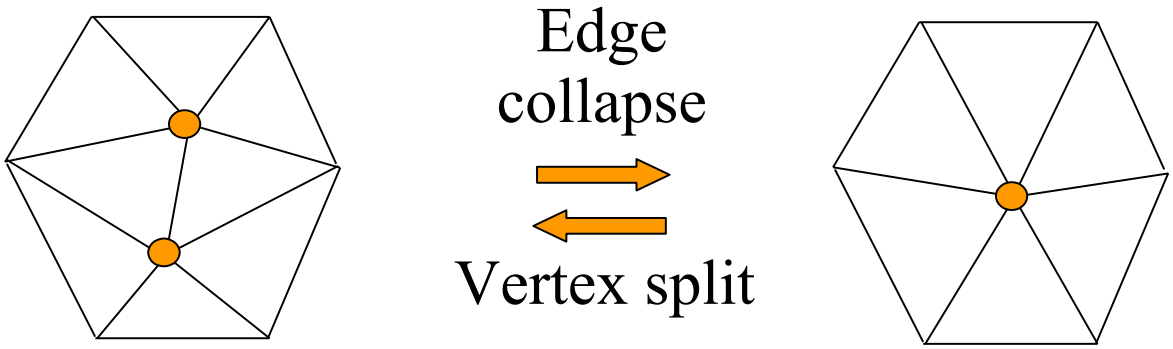
- **Critères ?**
  - Minimiser la distance entre le maillage simplifié et le maillage original
  - Distance de Hausdorff
    - difficile en pratique
  - Heuristiques...





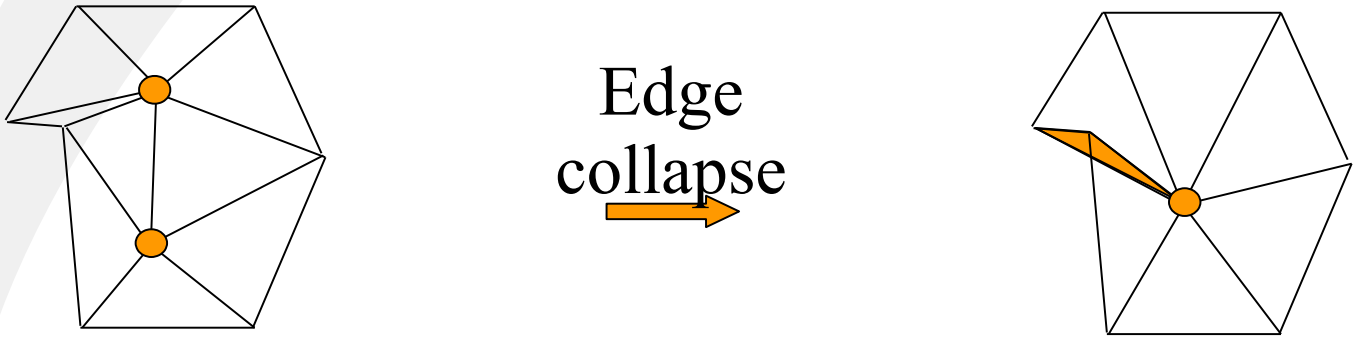
Edge collapse

- Cet opérateur transforme une arête en un sommet. L'opérateur inverse appelé "vertex split" ajoute une arête et le triangle qui lui est adjacent.

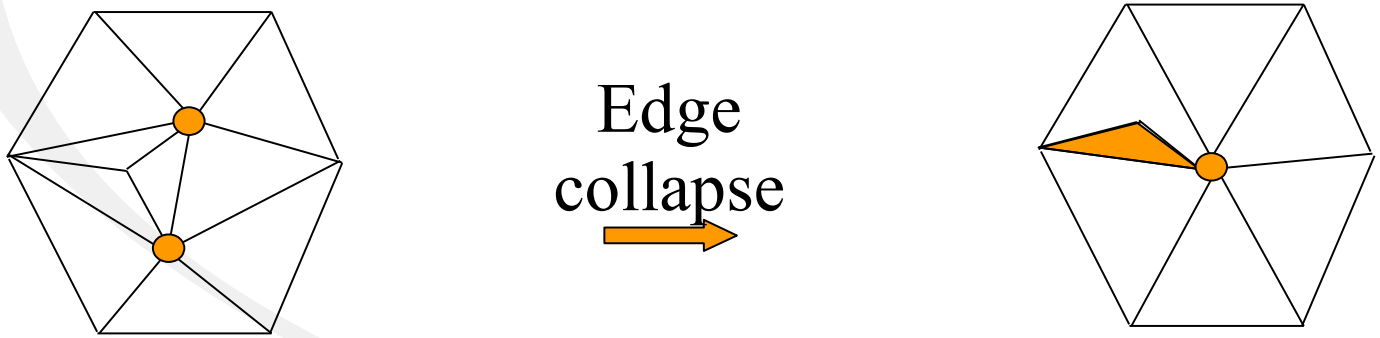


Quand ne pas utiliser “Edge collapse”

- **Attention au problème de recouvrement:**

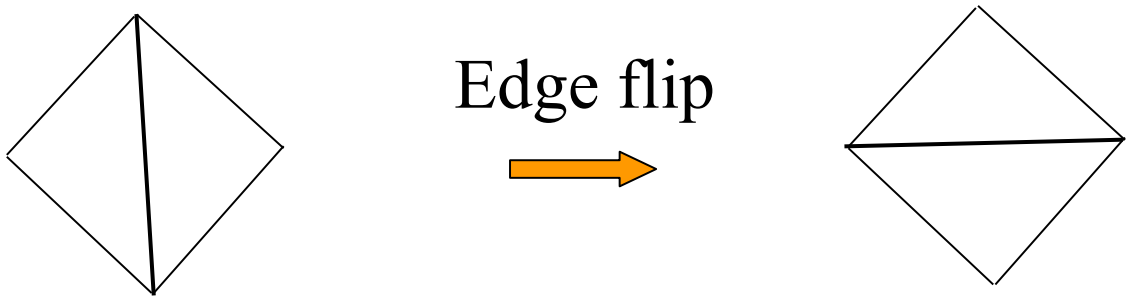


  - Le maillage reste 2-manifold, mais le résultat est indésirable
- **Le maillage peut aussi devenir non-manifold:**



Edge flip

- **Edge flip**
  - uniformiser les valences
  - aligner les arêtes avec les angles saillants



- Il a été montré que toutes les opérations de simplification de la géométrie d'un maillage peuvent être réalisée à l'aide des opérateurs "edge flip" et "edge collapse".

Simplification par edge collapse

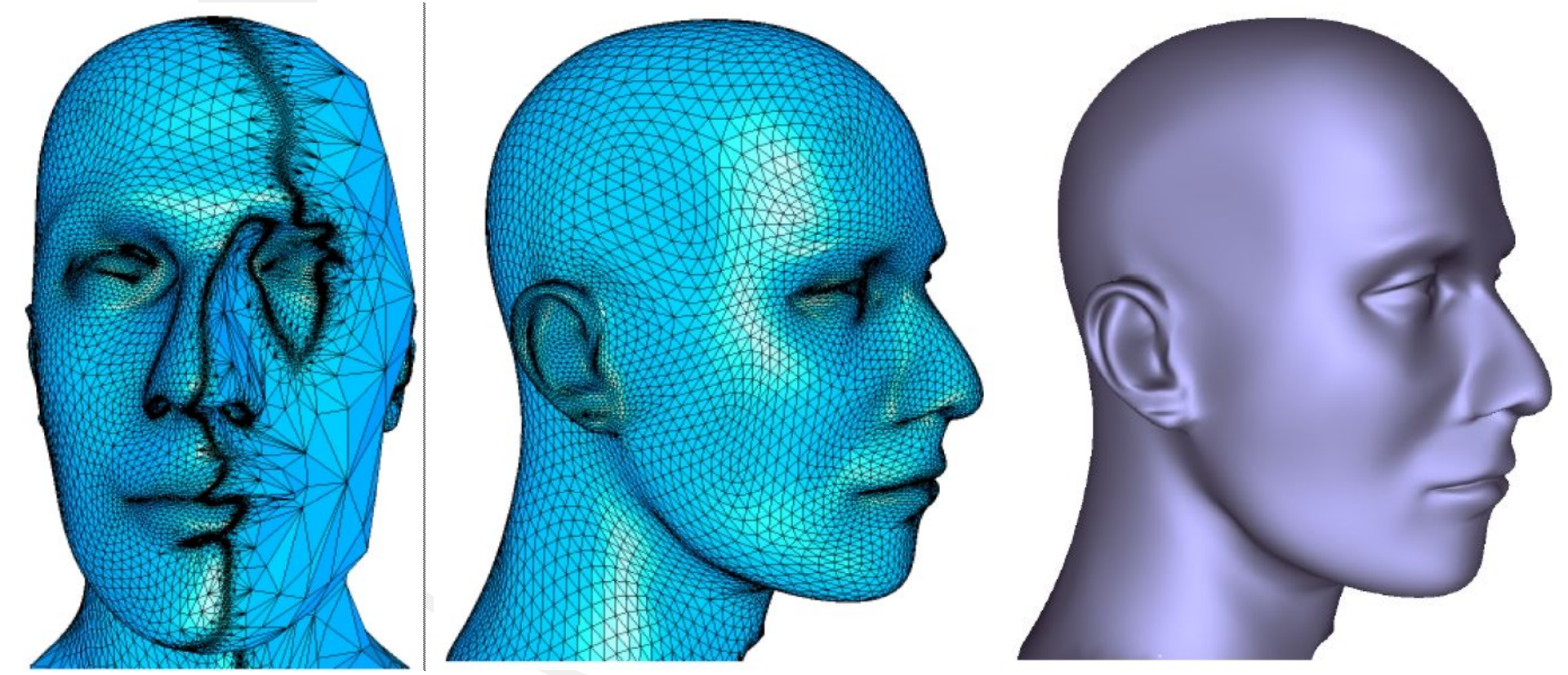
- **Calculer une erreur pour chaque arête**
  - représente l'erreur introduite par la contraction de l'arête
- **Tant que #triangles > seuil**
  - contracter l'arête avec l'erreur minimale
  - mettre à jour les erreurs des arêtes voisines
- **Améliorations :**
  - contracter plusieurs arêtes en même temps
  - passe de edge-flip pour régulariser le maillage
  - prise en compte de différents critères dans le calcul de l'erreur (géométrie, normales, texture, forme des triangles, etc.)

LOD continu

- **Adapter la résolution de l'objet au polygone prêt**
  - une seule structure représentant l'objet avec une représentation continue des détails
  - utiliser le nombre optimal de polygones pour représenter l'objet à la définition souhaitée
  - modification de la géométrie au cours de l'exécution de l'application
    - approche « top-down »
      - supprimer les triangles un à un → très difficile en pratique !!!
    - approche « bottom-up »
      - partir d'une version extrêmement simplifiée
        - insérer les détails stockés dans une représentation multi-resolution
      - ex :
        - surface de subdivision
        - surface de subdivision + offsets
        - progressive mesh
        - quatree pour les terrains

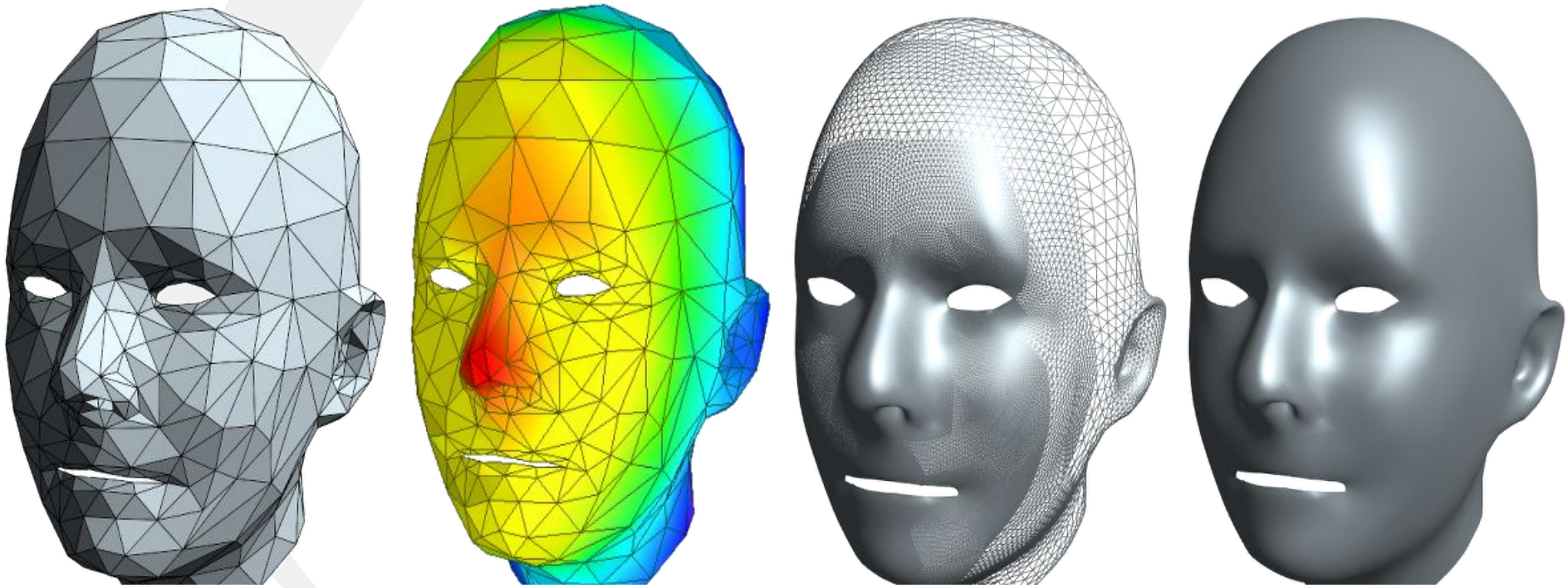
LOD dépendant du point de vue

- **Extension des LOD continus**
  - critère de simplification qui est dépendant du point de vue
  - représentation anisotrope : différentes zones du même objet sont visualisées à des niveaux de détail différents



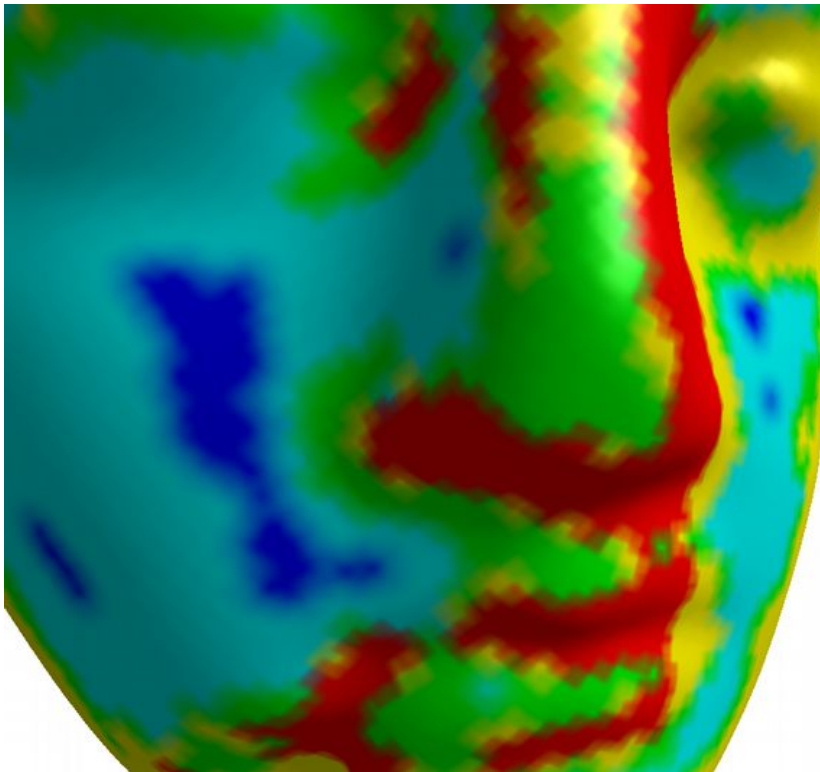
Raffinement à la volé

- **Résolution dépendante du point de vue**



LOD dépendant du point de vue

- **Métrique d'erreur**
  - Dépend de nombreux facteurs
    - distance à l'observateur, résolution de l'image,
    - orientation, courbure,
    - attributs, éclairage, effets de masquage,
    - texture, etc.



Représentations des maillages

- **Pour le rendu : « la soupe de polygones »**
  - liste de sommets (avec attributs)
  - liste de faces
- **Requêtes :**
  - comment lister les arêtes de manière unique ?
  - quelles sont les faces voisines d'une arêtes ? d'une face ? d'un sommet ?
  - quels sont les sommets voisins d'un sommet ?
  - etc.
- **Opérations de bases :**
  - insertion/suppression de sommets, faces, et arêtes
  - contraction/split d'arêtes
  - edge flip

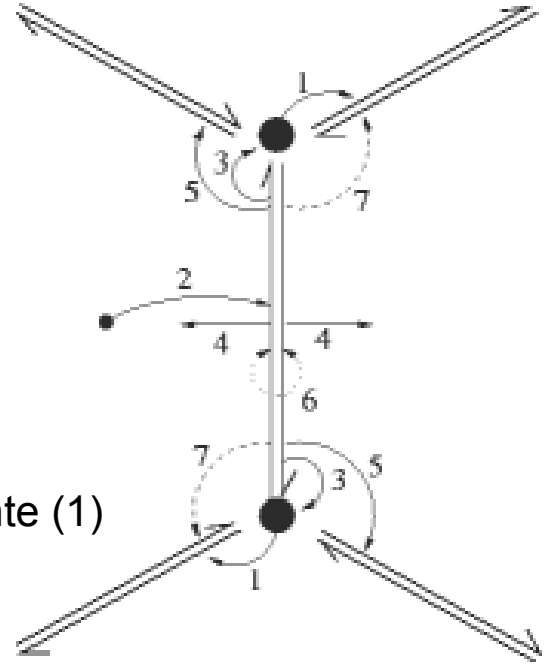
→ ces opérations doivent maintenir la validité du maillage

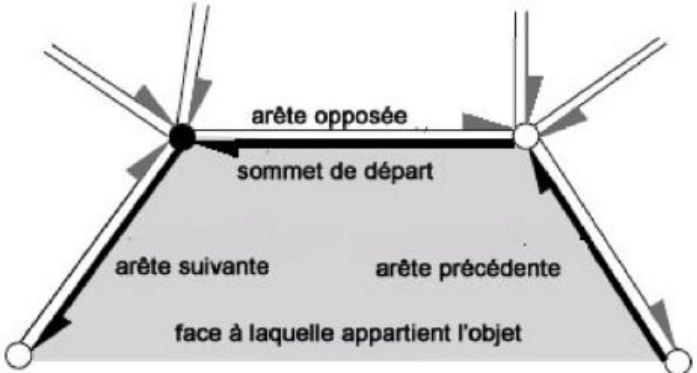
→ besoin de structures de données plus sophistiquées

- le standard : les halfedges (demies-arêtes)

Halfedges

- **Objectifs**
  - Accès rapide aux informations géométriques et topologiques
  - Représentation compact (faible cout mémoire)
- **Principe**
  - une arêtes est décomposée en deux demi-arêtes orientées
    - pour chaque demi-arête, on mémorise :
      - *la demi-arête opposée (6)*
      - le sommet vers lequel elle pointe (3)
      - la face à laquelle elle appartient (4)
      - la demi-arête suivante (5)
      - *la demi-arête précédente (7)*
  - chaque sommet contient un pointeur sur une demi-arête sortante (1)
  - chaque face, un pointeur vers une de ses demi-arête (2)

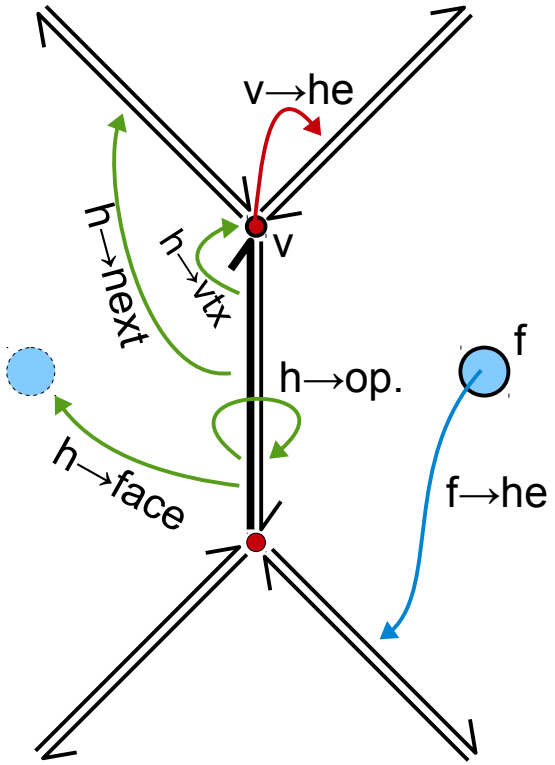




Halfedges

- **Implémentation(naive):**

```
struct HalfEdge {
  HalfEdge    *next ;
  HalfEdge    *opposite ;
  Face        *face ;
  Vertex      *vertex;
  /* attributes */
} ;

struct Face {
  HalfEdge    *he;
  /* attributes */
} ;

struct Vertex {
  HalfEdge    *he;
  /* attributes */
} ;

struct Mesh {
  std::vector<HalfEdge>    halfedges ;
  std::vector<Vertex>      vertices;
  std::vector<Face>        faces;
} ;
```



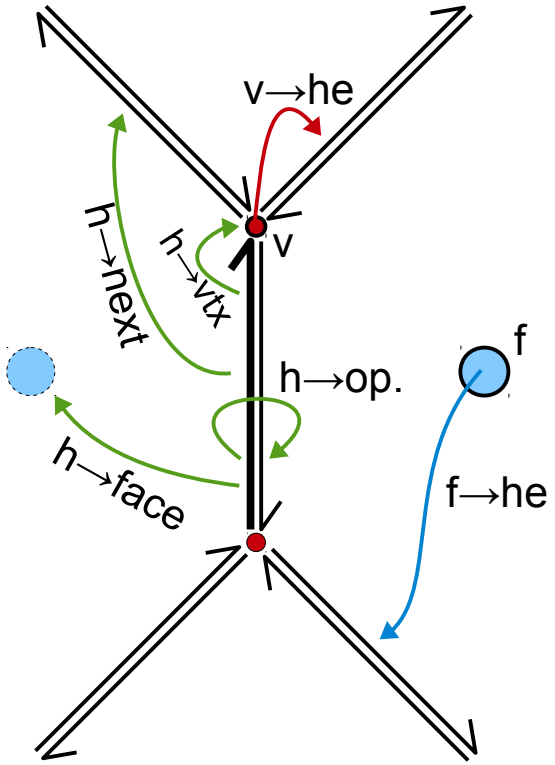
Halfedges

```
struct HalfEdge {
  HalfEdge     *next ;
  HalfEdge     *opposite ;
  Face         *face ;
  Vertex       *vertex;
};
struct Face { HalfEdge *he; };
```

```
struct Vertex { HalfEdge *he; };

struct Mesh {
  vector<HalfEdge>    halfedges ;
  vector<Vertex>      vertices;
  vector<Face>              faces;
} ;
```

- **Exercice : parcours des sommets d'une face**

```
Face &f = mesh.faces[i];
Vec3 cog = 0;

...
```



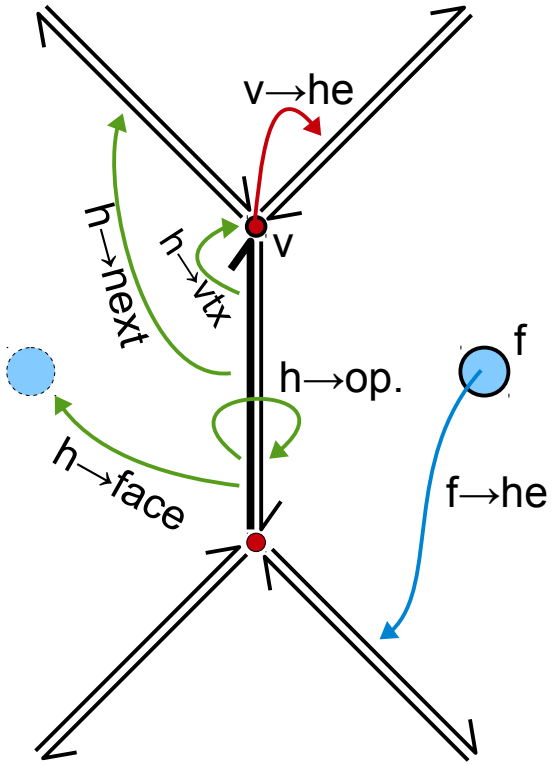
Halfedges

```
struct HalfEdge {
  HalfEdge     *next ;
  HalfEdge     *opposite ;
  Face         *face ;
  Vertex       *vertex;
};
struct Face { HalfEdge *he; };
```

```
struct Vertex { HalfEdge *he; };

struct Mesh {
  vector<HalfEdge>    halfedges ;
  vector<Vertex>      vertices;
  vector<Face>              faces;
} ;
```

- **Exercice : parcours des sommets d'une face**

```
Face &f = mesh.faces[i];
Vec3 cog = 0;

HalfEdge *h = f.he;

do {

  cog += h->vertex->position;

  h = h->next ;

} while (h!=f.he) ;
```



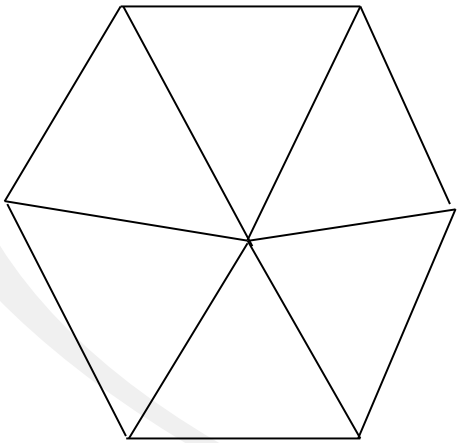
Halfedges

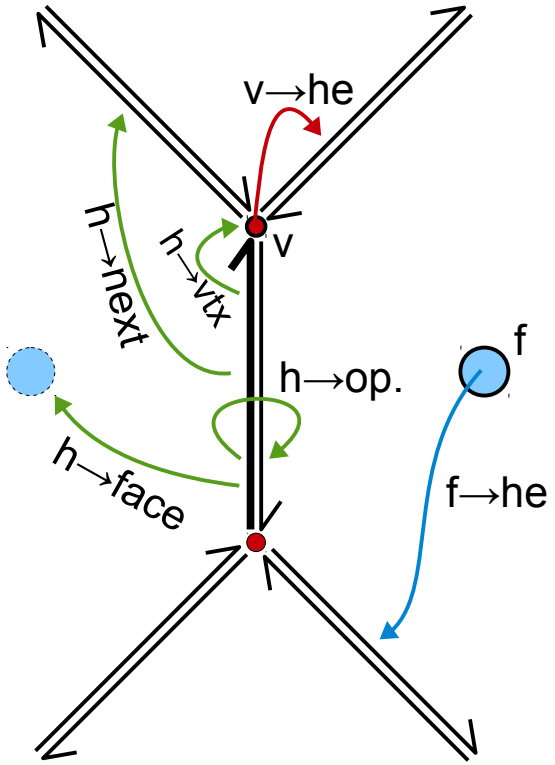
```
struct HalfEdge {
  HalfEdge     *next ;
  HalfEdge     *opposite ;
  Face         *face ;
  Vertex       *vertex;
};
struct Face { HalfEdge *he; };
```

```
struct Vertex { HalfEdge *he; };

struct Mesh {
  vector<HalfEdge>    halfedges ;
  vector<Vertex>      vertices;
  vector<Face>              faces;
} ;
```

- **Exercice : parcours du 1er anneau**

```
Vertex &vi = mesh.vertices[i];
Vec3 cog = 0;

...
```





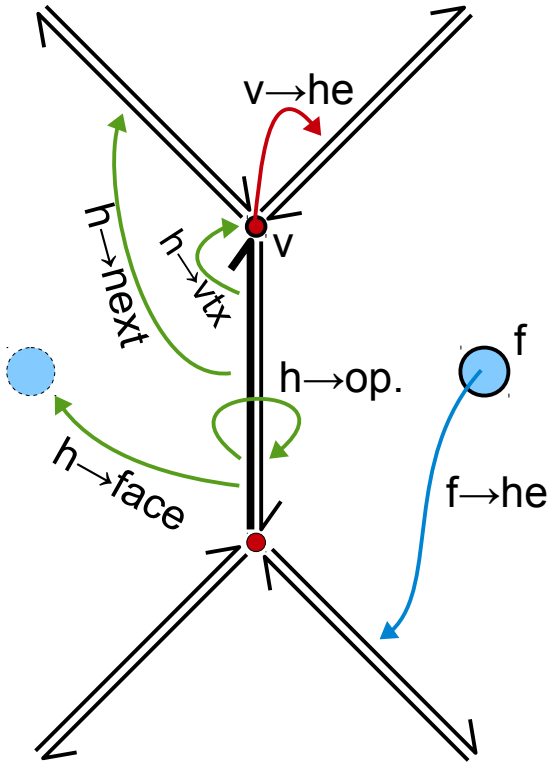
Halfedges

```
struct HalfEdge {
  HalfEdge     *next ;
  HalfEdge     *opposite ;
  Face         *face ;
  Vertex       *vertex;
};
struct Face { HalfEdge *he; };
```

```
struct Vertex { HalfEdge *he; };

struct Mesh {
  vector<HalfEdge>    halfedges ;
  vector<Vertex>      vertices;
  vector<Face>              faces;
} ;
```

- **Exercice : parcours du 1er anneau**

```
Vertex &vi = mesh.vertices[i];
Vec3 cog = 0;

HalfEdge *h = v.he;

do {

  cog += h->vertex->position;

  h = ????

} while (h!=v.he) ;
```



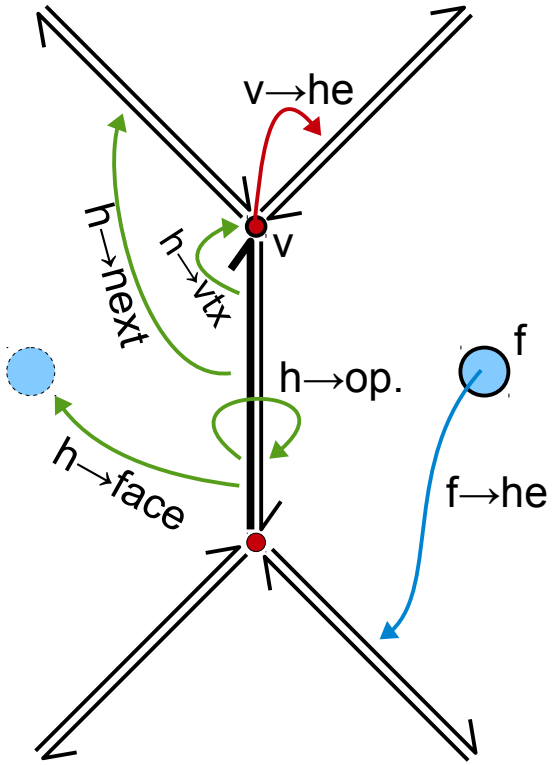
Halfedges

```
struct HalfEdge {
  HalfEdge     *next ;
  HalfEdge     *opposite ;
  Face         *face ;
  Vertex       *vertex;
};
struct Face { HalfEdge *he; };
```

```
struct Vertex { HalfEdge *he; };

struct Mesh {
  vector<HalfEdge>    halfedges ;
  vector<Vertex>      vertices;
  vector<Face>              faces;
} ;
```

- **Exercice : parcours du 1er anneau**

```
Vertex &vi = mesh.vertices[i];
Vec3 cog = 0;

HalfEdge *h = v.he;

do {

  cog += h->vertex->position;

  h = h->opposite->next ;

} while (h!=v.he) ;
```



Halfedges

- **Exercice : parcours du 2ème anneau**
  - Idée :
    - récursion sur chacun des voisins
    - plus masque binaire pour éviter les doublons (marquer les sommets déjà visités)

- **Exercice : parcours de N anneaux**
  - Idem !

Halfedge

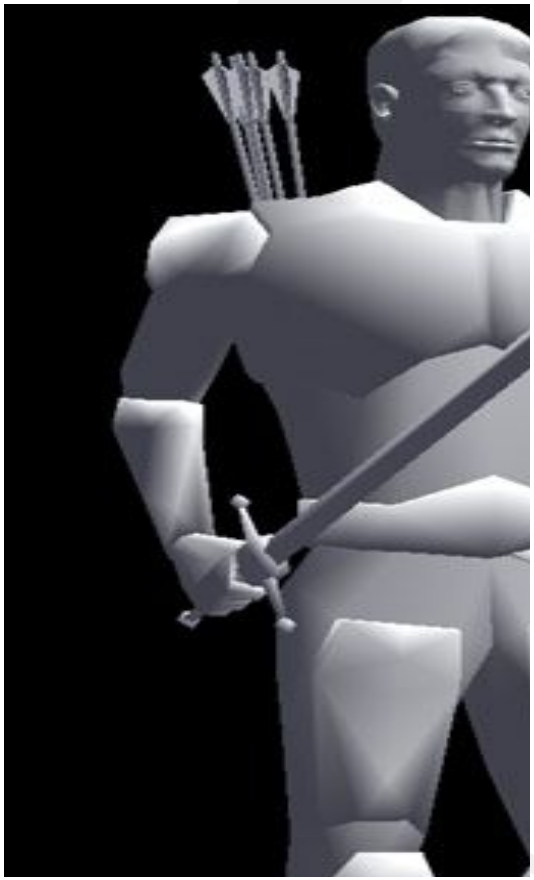
- **Mise en œuvre**
  - utilisation aisée
  - mise en oeuvre robuste des opérateurs assez complexe

- **Bibliothèques open-sources**
  - **SurfaceMesh** (http://graphics.uni-bielefeld.de/publications/imr11/)
  - OpenMesh (http://www.openmesh.org)
  - VCG-lib (http://vcg.sf.net - MeshLab)
  - CGAL – http://www.cgal.org

et le GPU....

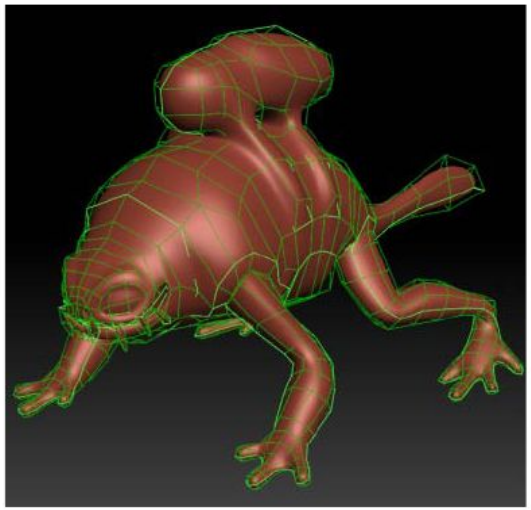
LOD hybride

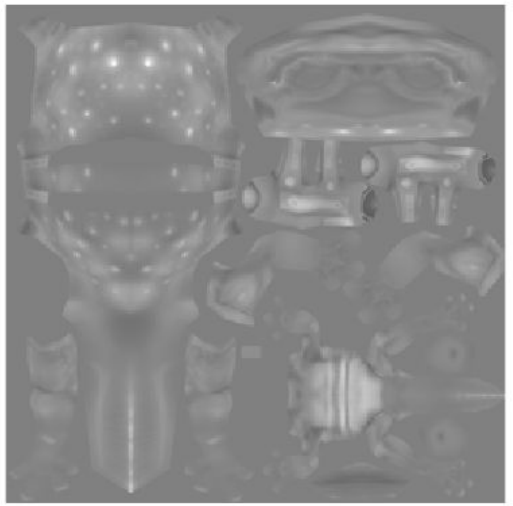
- **Simplification de maillage + normal mapping**
  - transfère des détails sous la forme de carte de normales



Displacement Maps

- **Simplification de maillage**
  - + displacement map (offset le long de la normale, ou vecteur 3D)
  - + *normal map*
- **Compact : pas de coordonnées 3D, pas**



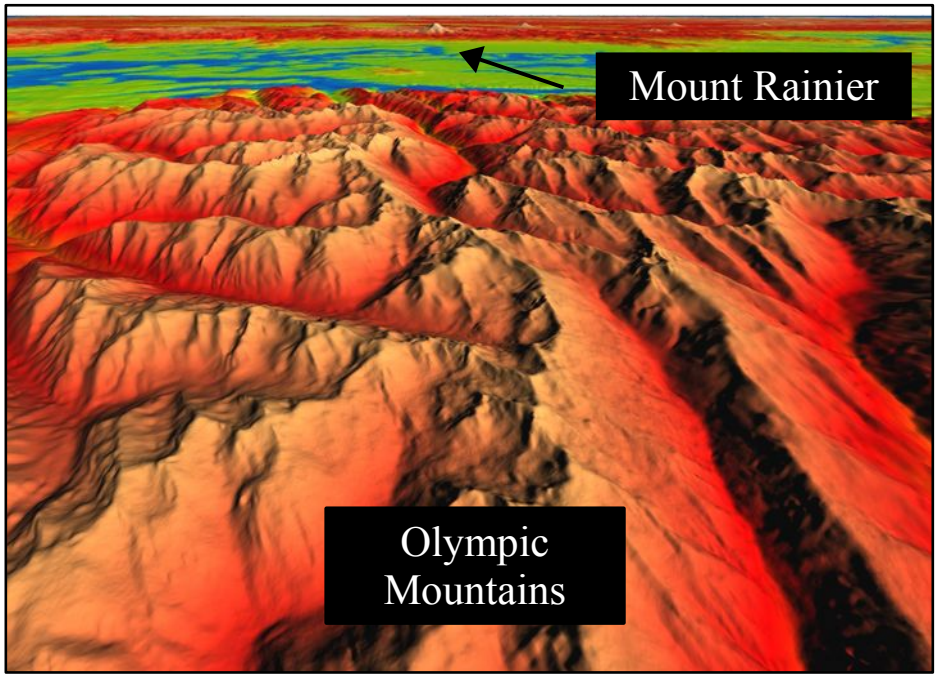


© Bay Raitt

	Level 8	Level 16	Level 32	Level 64
Regular Triangle Mesh	16MB	59MB	236MB	943MB
Displaced Subdivision Surface	1.9MB	7.5MB	30MB	118MB

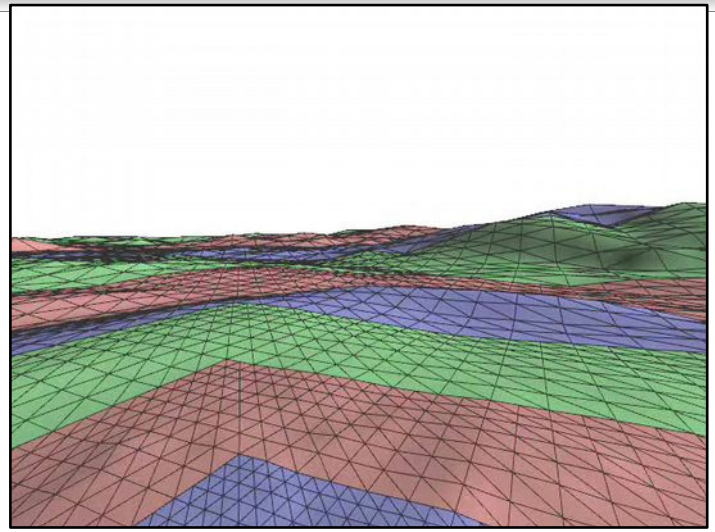
Cas particulier : terrain

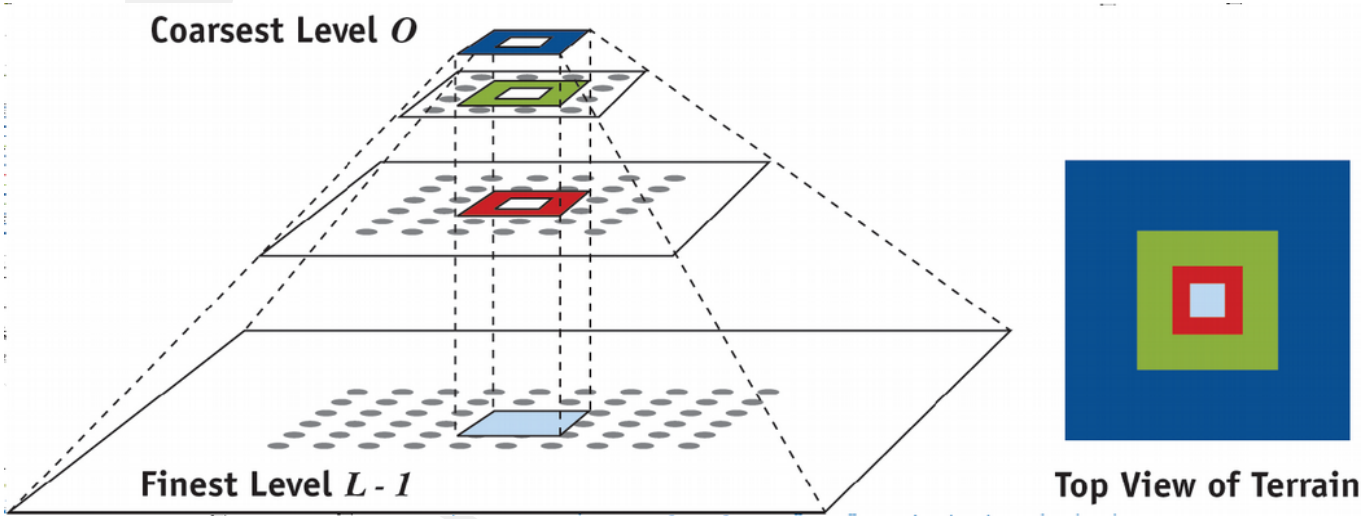
- **Problème : terrain = infini !**
  - => géométrie multi-résolution spécialisée
  - ex. : « *geometry clipmaps* » [Losasso04]



Geometry clipmaps [Losasso04]

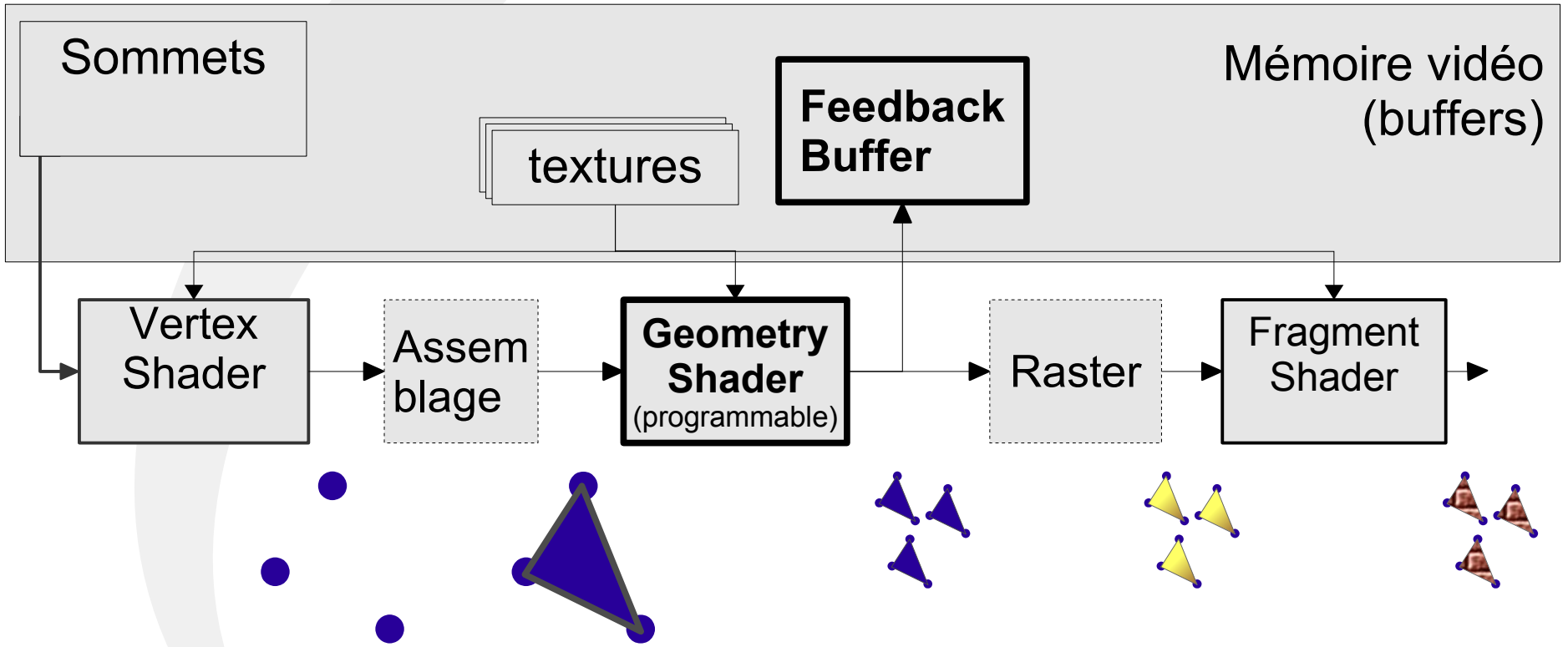
- = clipped mimap hierarchy
  - construction en fonction du point de vue (distance uniquement)
  - mise à jour incrémentale
- permet la compression, synthèse
- => simple mais efficace !





Top View of Terrain

Geometry Shader



Geometry Shader

- **En entrée**
  - une primitive (POINT, LINE, TRIANGLE)
    - = tableau des sommets (+ attributs) en sortie du vertex shader
- **En sortie**
  - gl_Position + variables « out »
    - EmitVertex() pour générer un sommet
    - EndPrimitive() pour finaliser une primitive
      - POINTS, LINE_STRIP, TRIANGLE_STRIP
      - peut être différent du type en entrée (POINT → TRIANGLE)
  - variables en sorties (out) interpolées par le raster et deviennent entrées du fragment shader
  - on peut générer plusieurs primitives (ou zéro → culling)
- **Feedback-buffer**
  - les sommets en sortie peuvent être empilés (sauvegardés) dans un VBO
  - permet de mettre en cache la géométrie ainsi générée

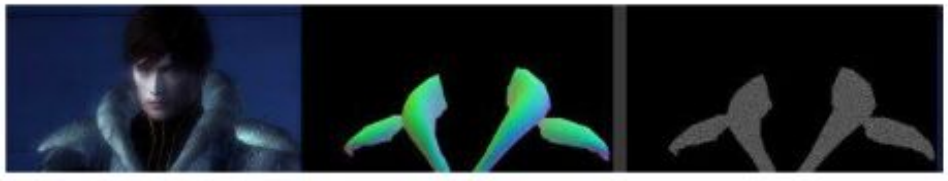
Geometry Shader

```
layout (triangles) in;
layout (triangle_strip, max_vertices=6) out;
uniform mat4 mat_mvp;
in vec3 in_normals[3];
in vec3 in_positions[3];
out vec3 out_normal;

void main() {

  //Pass through the original vertex
  for(int i=0; i<3; i++)  {
    gl_PerVertexOut.gl_Position = gl_PerVertexIn[i].gl_Position;
    out_normal = in_normals[i];
    EmitVertex();
  }
  EndPrimitive();

  //Push the vertex out a little using the normal
  for(int i=0; i<3; i++) {
    vec3 p = in_positions[i] + 0.1 * in_normals[i];
    gl_PerVertexOut.gl_Position = mat_mvp * vec4(p, 1.0);
    out_normal = in_normals[i];
    EmitVertex();
  }
  EndPrimitive();
}
```

Geometry Shader

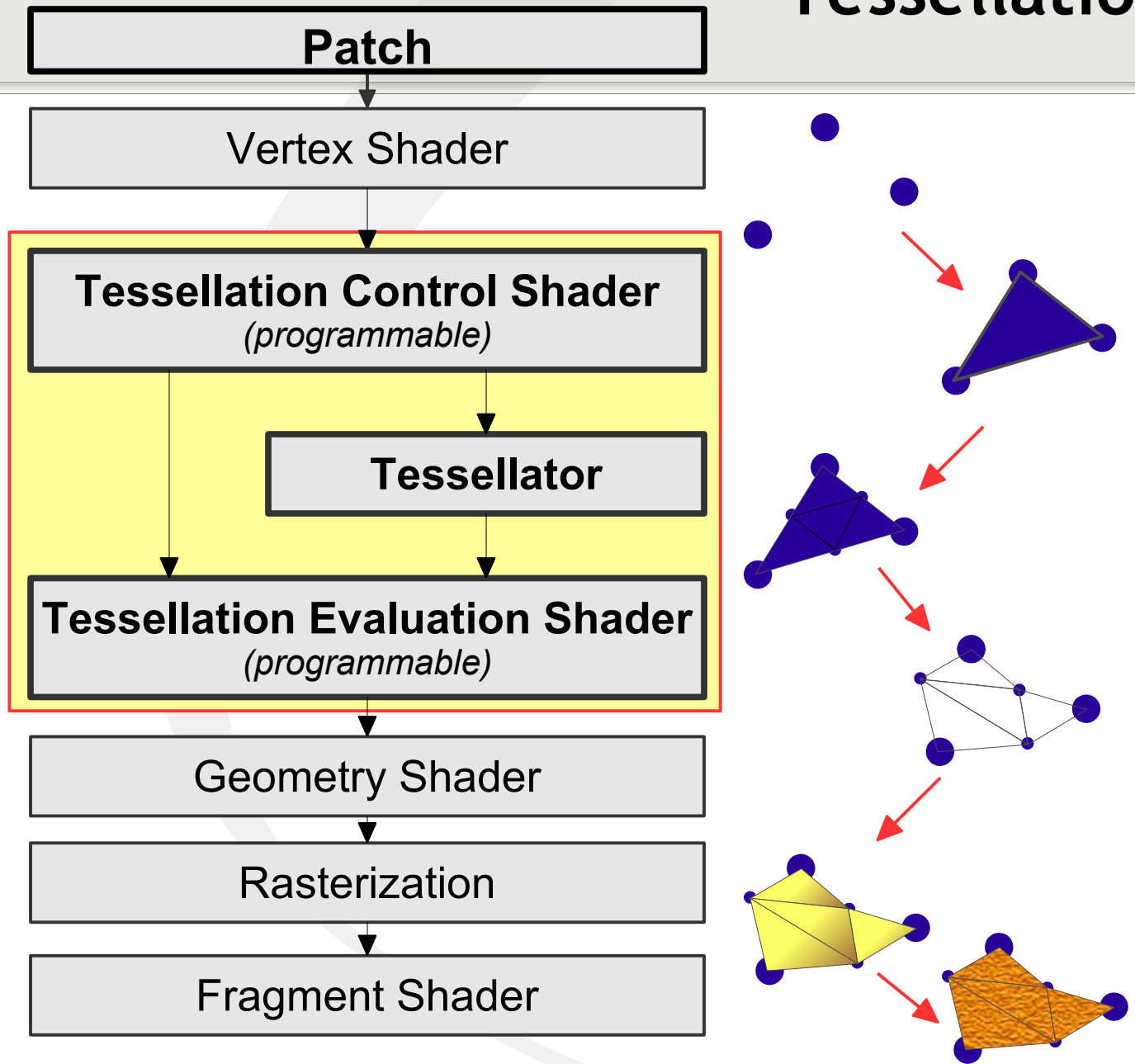
- **Limitations**
  - efficace uniquement si le nombre de primitives générées est ~ constant
    - en caricaturant :
      - cost = #input_primitives * #max_output_vertices
  - peu efficace pour faire du raffinement, LOD, …

- **Applications**
  - Occlusion culling
  - Rendu dans les 6 faces d'une cube map en une seule passe
  - Shadow volume : génération des faces des volumes d'ombres
  - Silhouette drawing, Line drawing (outline)
  - Fourrure



  - etc.

Tessellation Engine



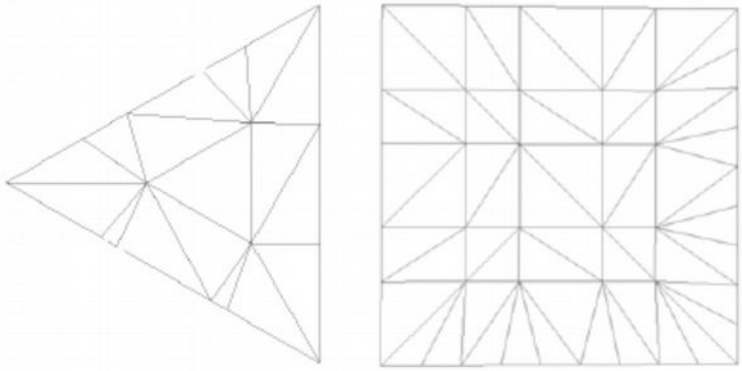
Tessellation Engine

- **En entrée**
  - Nouveau type de primitive : « patch »
    - nombre arbitraire de sommets (points de contrôle), pas de topologie
      - ex. : NURBS

- **Vertex Shader**
  - transformation des points de contrôle
    - ex. : animation, déformation, skinning, etc.
    - → calculs couteux réalisés à une plus faible résolution

Tessellation Engine

- **Tessellation Control Shader**
  - Patch → Base (triangle/quad)
    - Exécuté N fois où N = nombre de sommets en sortie
  - Niveau de subdivision par arête et par patch
    - réel (continue)
  - Précalcul des info par patch

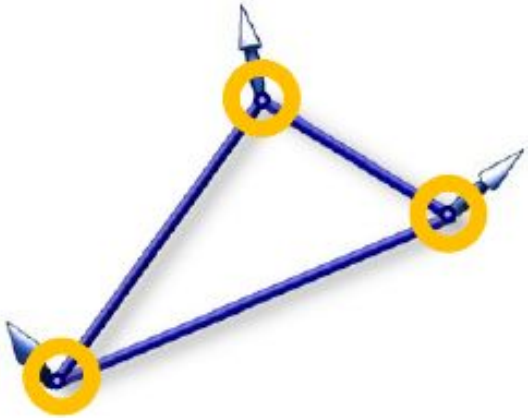


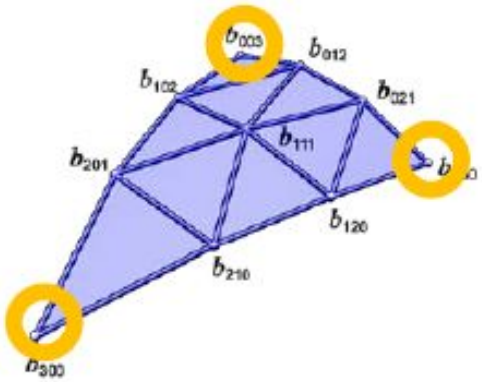
- **Tessellator**
  - Base → Triangles + uv(w) (coordonnées barycentriques)
  - Non programmable
  - Symétrique

- **Tessellation Evaluation Shader**
  - coordonnées paramétriques uv(w) → 1 sommet
  - Plus info issues du Tess. Control Shader
  - Interpolation des attributs, application d'une carte de déplacement

Exemple PN - triangles

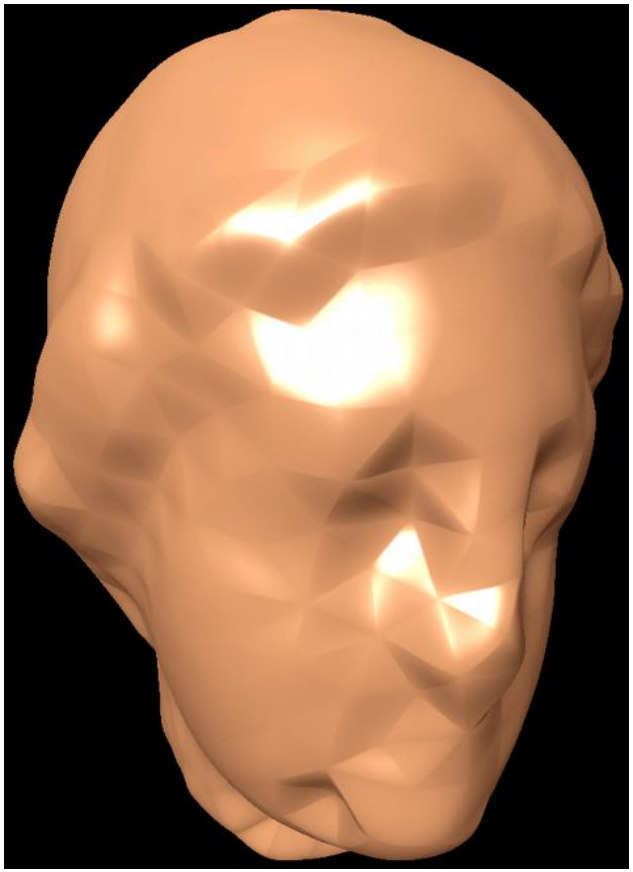
- **Idée**
  - Remplacer chaque triangle par un triangle de Bézier
  - Points de contrôles dépendent des normales





Input Triangles

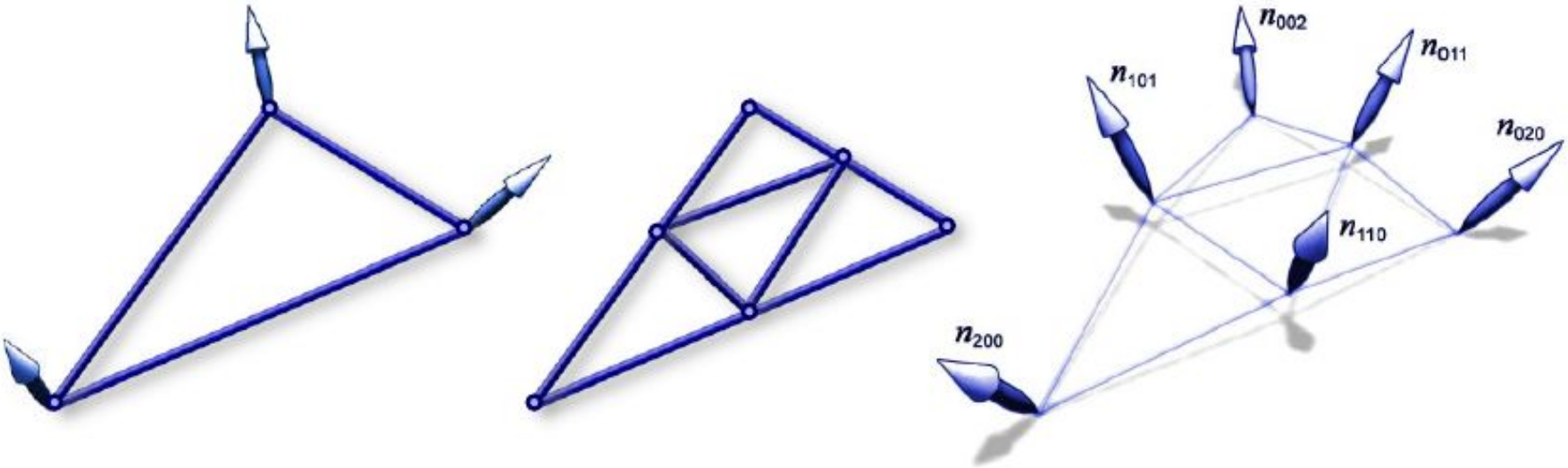
Output
Curved PN triangles



C^0 seulement !

PN – triangles

- **+ interpolation quadratique des normales :**

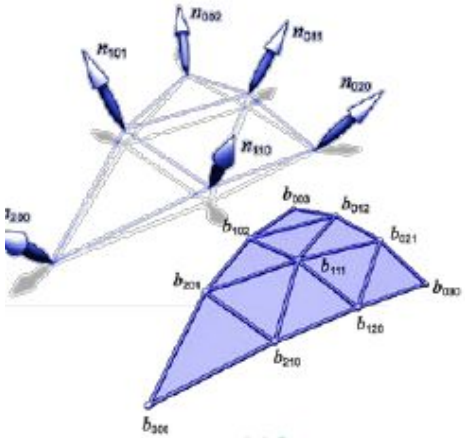


Exemple PN - triangles

Entrée :
* points de contrôles
(triangle grossier)

Tessellation Control

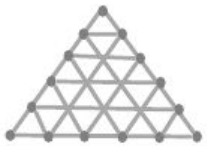
Sortie :
* points de contrôles
du triangle de Bézier



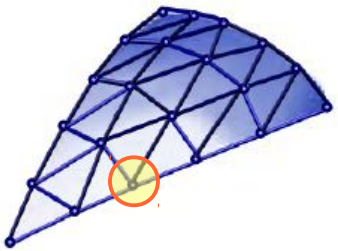
Entrée :
* niveaux de tessellation

Tessellator

Sortie:
* coordonnées u,v,w



Tessellation Evaluation



Sortie:
* un sommet interpolé