

ETI5 Majeure Image: Modélisation 3D

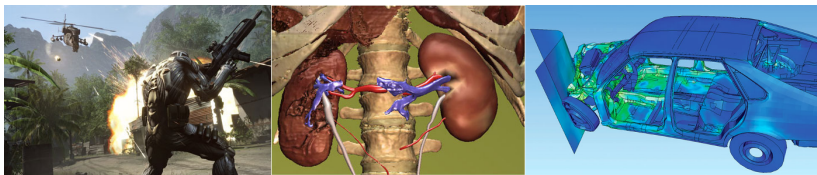
CPE Lyon
damien.rohmer@cpe.fr

4 Novembre 2010

Synthèse d'images : Domaines d'applications

- Loisir, Graphique (Entertainment) : Cinéma, Jeux vidéos, Communication,
- Calcul : Engineering, médical, ...
- CAO (CAD) : Conception, prototypage, ...

Interactions entre les domaines !



© Crysis, Sofa

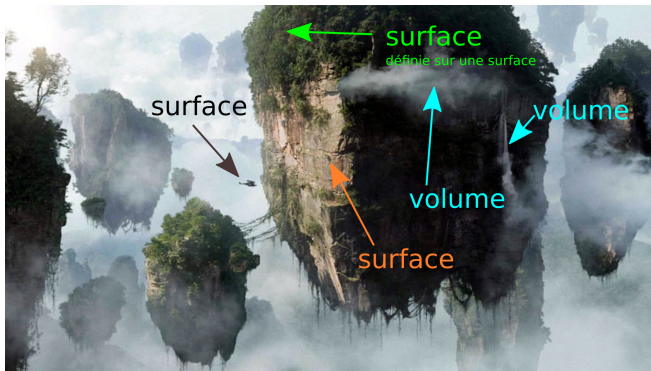
Modélisation : But du cours

- Comment modélise t'on un objet 3D, inventaire ?
- Quel modèle pour quelle application ?



■ Comment modéliser un objet 3D ?

- 1 Surface uniquement ou volume ?
- 2 Comment l'encode t'on ?



Les modèles virtuels 3D

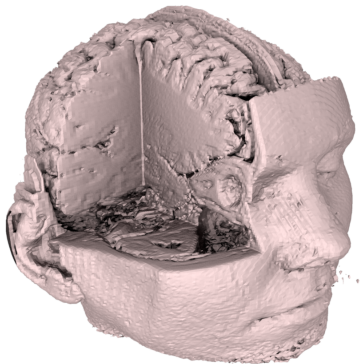
Quels sont les objets virtuels ?



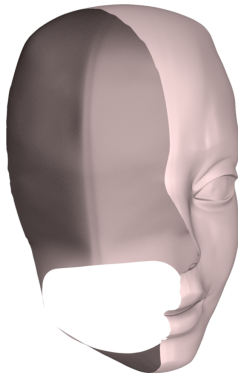
© Day After Tomorrow, © Lord of the Rings, © Titanic

Modélisation d'un objet 3D

■ Modélisation volumique



■ Modélisation surfacique



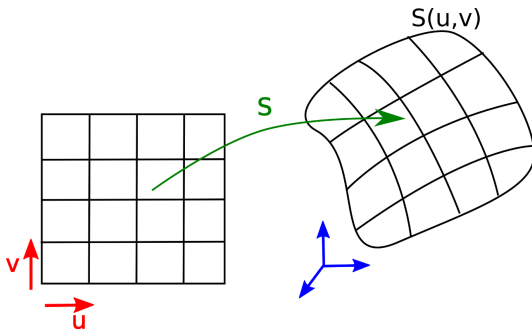
- Explicite
 - BRep : Boundary Representation
 - Maillage
 - Paramétrique
 - Subdivision
 - CSG : Constructive Solide Geometry
- Implicite
 - Voxels
 - Paramétrique
 - Squelettes
 - Analytiques
 - Points-sets
 - MLS
 - Surfels
- Fractales

Explicite BREP

- ⇒ Maillage
 - . Paramétrique
 - . Subdivision

- Explicitement $\simeq$ paramétriquement :

$$S: \begin{cases} \mathcal{D} \subset \mathbb{R}^2 & \rightarrow \mathbb{R}^3 \\ (u, v) & \mapsto S(u, v) = (S_x(u, v), S_y(u, v), S_z(u, v)) \end{cases}$$



- $S = \text{mapping} \neq \text{Surface } \Gamma = \text{trace de } S \text{ dans } \mathbb{R}^3$
- Brep $\simeq$ Estimation de S .

Maillage (triangulaire)

- Maillage triangulaire = BRep le plus simple.
- On ne connaît pas S : On l'estime localement de manière discrète

$$S = \bigcup_i S_i$$

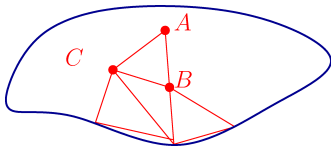
- Mapping le plus simple : Linéaire

$$S_i : \begin{cases} \mathcal{D} \subset \mathbb{R}^2 & \rightarrow \mathbb{R}^3 \\ (u, v) & \mapsto S_i(u, v) = u\vec{AB} + v\vec{AC} \end{cases}$$

$$\mathcal{D} : 0 \leq u + v \leq 1$$

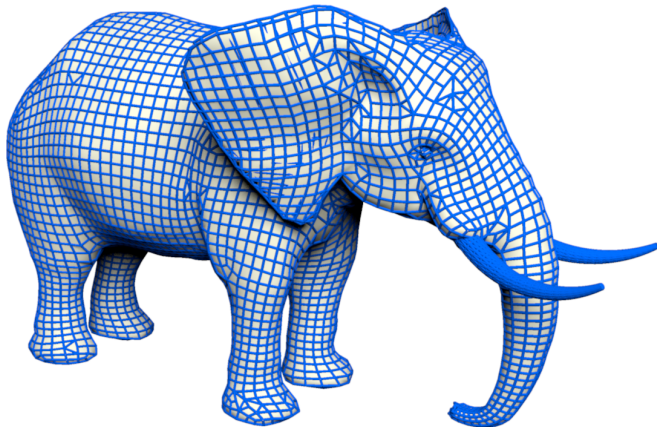
Propriétés :

- S est globalement $\mathcal{C}^0$.
- S n'est jamais $\mathcal{C}^1$ (sauf plan).
- Peut interpoler n'importe quel ensemble discret de points.

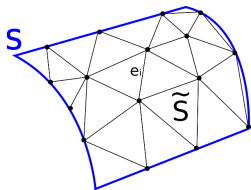


Maillage

- Cas spéciale : Un maillage peut contenir des polygones de N sommets ($N \geq 3$).
- V\'eritable polygone : N sommets coplanaires. Sinon on triangle.



Maillage : Approximation



- S : Vraie surface différentiable.
- $\mathcal{T}_S$: Surface triangulée

$$\|S - \mathcal{T}_S\| = h|\kappa_{\max}| \quad (\simeq h\|S''\|)$$

Approximation linéaire (ordre 1).

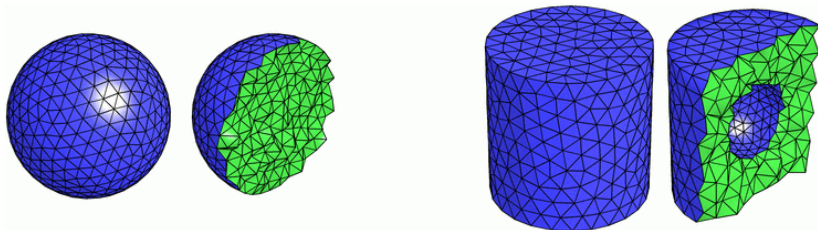
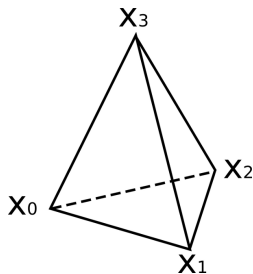
- $h = K \max_i e_i$.

$$\Rightarrow \|S - \mathcal{T}_S\| = \mathcal{O}\left(\max_i e_i |\kappa_{\max}|\right)$$

Maillage (Volume)

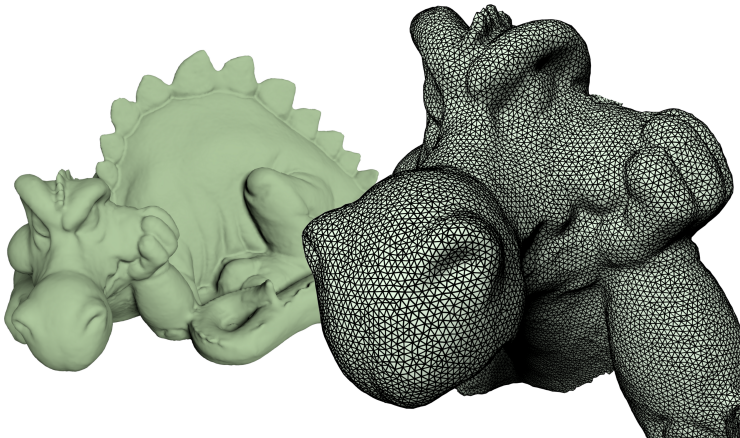
- Pour un volume :
élément linéaire = tétraèdre

$$\mathbf{x} = u\mathbf{x}_0 + v\mathbf{x}_1 + w\mathbf{x}_2 + z\mathbf{x}_3$$
$$0 < u + v + w + z < 1$$



Maillage, conclusion

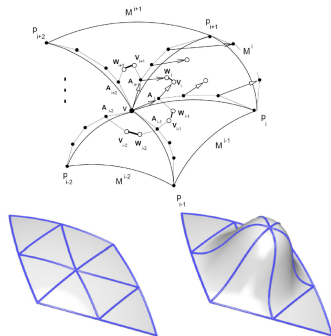
- + Le plus simple
- + Le plus polyvalent
- + Le plus répandu
- + Rendu
- La moins bonne approximation



Explicite BREP

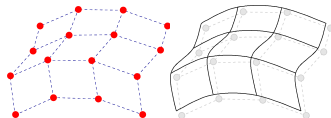
- . Maillage
- ⇒ Paramétrique
- . Subdivision

- Paramétrique = Ordre > 1
- Idée 1
 - Information de dérivées (triangles courbes)
 - Problème : Information non disponible.
 - ⇒ Peu utilisé.



[Yvart, Hahmann 01-04]

- Idée 2
 - Ajouter des sommets (patchs non triangulaires)
 - Problème : Structure des patchs.
 - Cas classique : Patchs rectangulaires uv .
 - ⇒ Très utilisé.

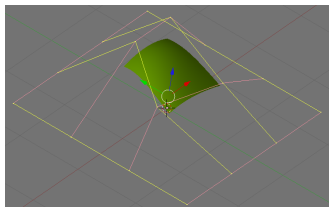


Paramétrique : Patch Splines

- Patch (4 x 4), fonctions bi-cubiques.
- ⇒ Surface paramétrique $\mathcal{C}^2$: Courbure continue.

$$S(u, v) = \sum_{i=0}^2 \sum_{j=0}^2 b_i(u) b_j(v) P_{ij}$$

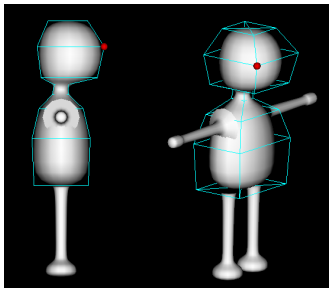
(surface produit tensoriel)



Cas particulier (vecteur de noeud uniforme)

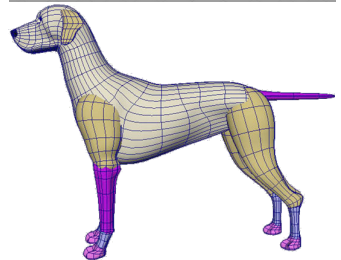
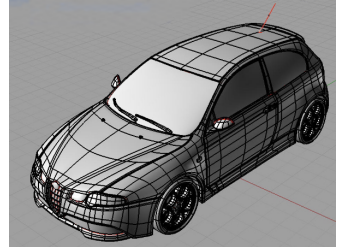
$$S(u, v) = (u^3 u^2 u \ 1) M [P_{ij}] M^T (v^3 v^2 v \ 1)^T$$

$$M = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$



Paramétrique, conclusion

- + Surface lisse (CAO).
- Structure par patches
 - modélisation manuelle
 - technique
 - jonctions



Explicite BREP

- . Maillage
- . Paramétrique
- ⇒ Subdivision

Surface de subdivisions

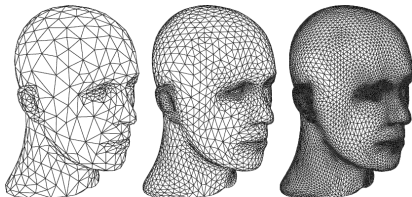
⇒ Réponse au problème :

- + Surface lisse
- + controle local
- + structure quelconque.

■ Courbes de subdivisions (principe : mask 1D)


$$\begin{pmatrix} x_n^{2k} \\ x_n^{2k+1} \end{pmatrix} = \begin{pmatrix} 3/4 & 1/4 \\ 1/4 & 3/4 \end{pmatrix} \begin{pmatrix} x_{n-1}^k \\ x_{n-1}^{k+1} \end{pmatrix}$$

■ Généralisation pour des maillages (principe : mask 2D)

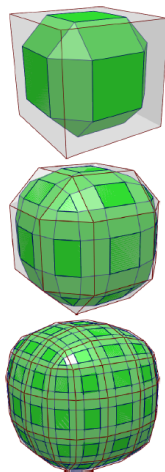
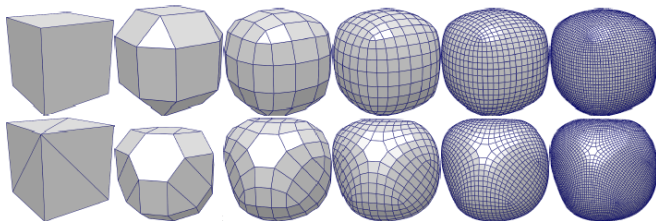


Surface de subdivisions

- étape 0 : Polygone de contrôle P^0 .
- étape 1 : Subdivision $P^0 = P^1$.
- $\vdots$
- étape i : Subdivision $P^{i-1} = P^i$.
- $\vdots$

Surface finale $S = \lim_{i \rightarrow \infty} P^i$.

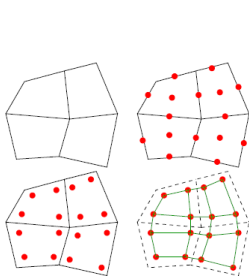
Il est possible d'avoir $S \mathcal{C}^2$ presque partout.



Surface de subdivisions

■ Schémas de subdivisions :

- **Loop** : triangles, C^2 pp, approximation.
- **Catmull Clark** : quads, C^2 pp, approximation.
- **Doo-Sabin** (corner cutting) : quads, C^2 pp, approximation.
- **Butterfly** : triangles, C^1 pp, interpolation.
- $\sqrt{3}$ -**Kobbelt** : triangles, C^2 pp, approximation.

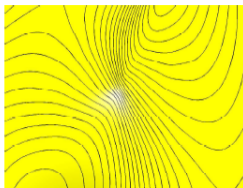
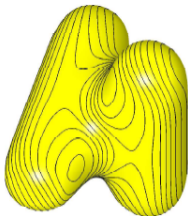
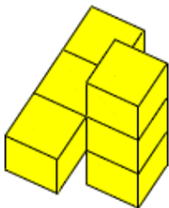
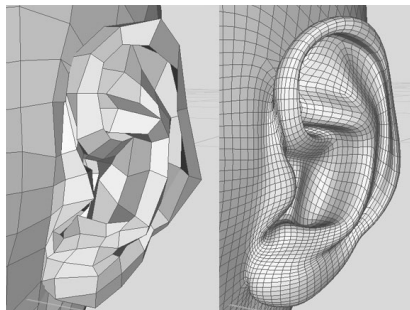


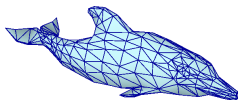
© Pixar, Geri's Game

D. Zorin, P. Schroder. **Subdivision for Modeling and Animation.** ACM SIGGRAPH Course Notes. 1999.

Surface de subdivisions : Conclusion

- + Structure quelconque.
- + Maillage lisse.
- Vraie surface inaccessible.
- Contrôle de l'aspect.
- Sommets extraordinaires.

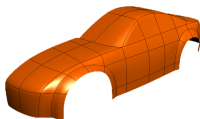




Maillage

- + Simple
- + Générique
- + génération automatique
- Non derivabilité
- *Mauvaise* approximation
- Manipulation

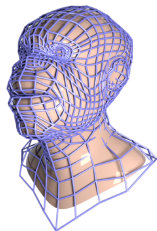
⇒ Graphique, Calcul.
Maya, 3DStudio,
Blender, ...



Paramétrique

- + Continuité
- + Informations de la paramétrisation
- Technique (modèle mathématique)
- Structure patches
- Génération manuelle nécessaire

⇒ CAD, (Graphique).
Rhino, Catia, ...



Subdivision

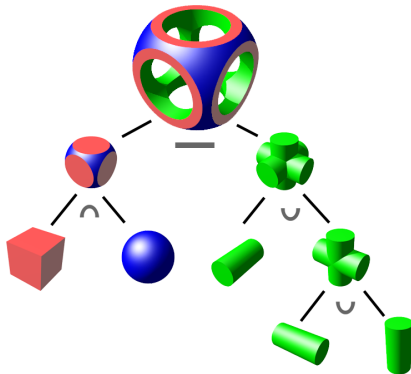
- + Apparence lisse
- +/- Pas de patches, sommets extraordinaires
- Pas/peu infos sur surface subdivisée

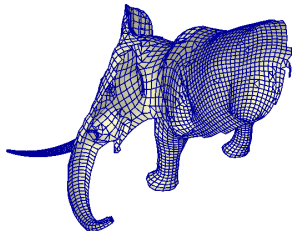
⇒ Graphique, (CAD).

Explicite CSG

- CGS = Assemblage de primitives par opérations Booléennes.

- Solide : intérieur/extérieur.
- Modéliser une chaîne d'assemblage
 - ⇒ CAO : Solid Works, AutoCAD, Catia, (PovRay), ...

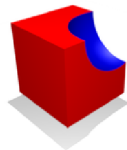
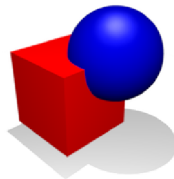




Brep

- + Modéliser objets complexes
- Approximation
- Surface uniquement
- Dependance à la discretisation

⇒ Surfaces quelconques discrètes : Graphique, Calcul et CAO.



CSG

- + Exacte
- + Méthode constructive
- Possibilitée limitées
- Lourd pour objets complexes
- Construction non unique

⇒ Objets *simples* exactes : CAO.

Implicite

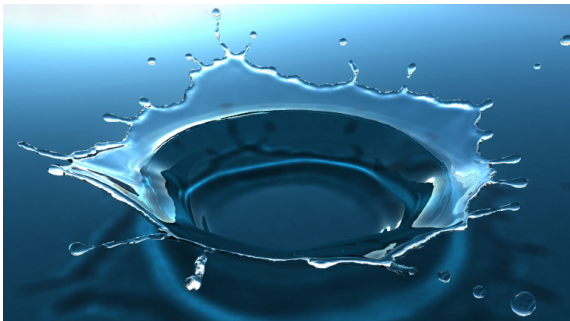
Implicite

- Problème : Modification de topologie.
- ⇒ Représentation implicite.

$$S = \{(x, y, z) \in \mathbb{R}^3 \mid \phi(x, y, z) = a\} = \phi^{-1}(a)$$

Rappel :

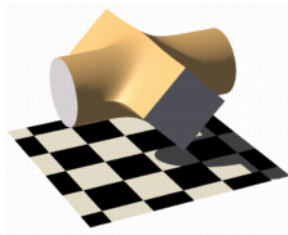
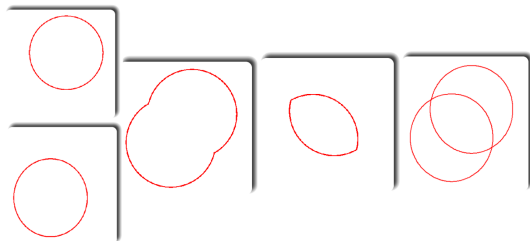
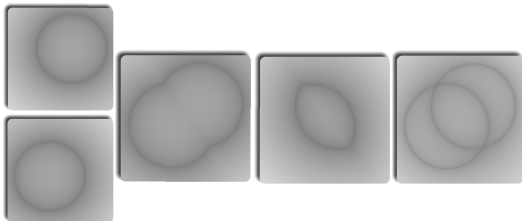
- Soit $\mathbf{x}_0 = S(u, v)$. Normale $n(u_0, v_0) = \nabla \phi(\mathbf{x}_0)$.



[Thuereu, Wojtan, Gross, Turk, SIGGRAPH 2010]

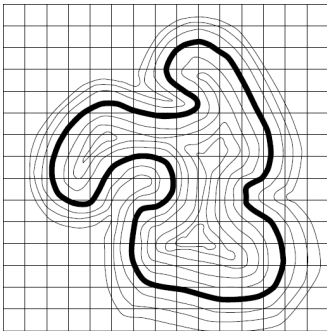
Ex. Fonction de distance :

- Opérateurs de mélange (blending).



Povray

Comment encode t'on le potentiel ?

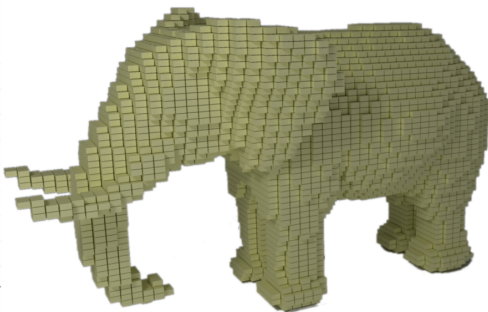
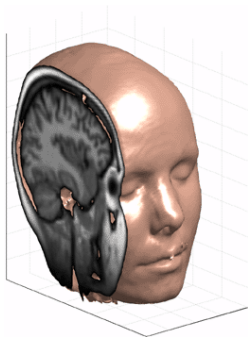


[Sethian]

Implicite : Voxels

- On discrétise l'espace en voxels.
- On stocke dans une grille $\phi(k_0, k_1, k_2)$.
- Accès par interpolation (linéaire, spline, ...)
 - + Général
 - Mémoire (ex. 1024^3 voxels : 8Go)

Imagerie scanner 3D (médical, mécanique, ...)



■ Squelettes

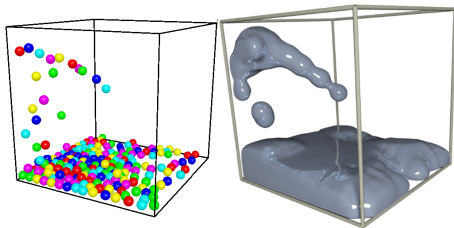
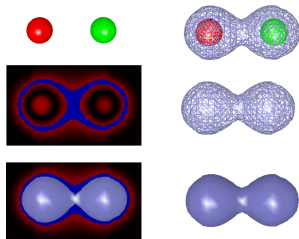
- Blobs $\phi_i = e^{-a\|x-x_i\|^2}$
- Metaballs $\phi_i = \sum_k a_k \|x - x_i\|^k$
- Convolution $\phi_i = \int \omega(y) h(\|x - y\|) dy$

Contrôle directe, méthode manuelle

⇒ Graphique.



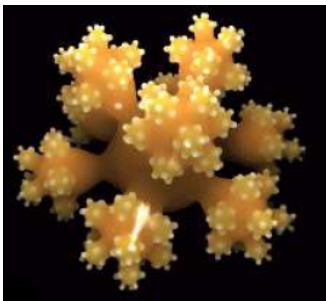
[Sherstyuk, 98-99]



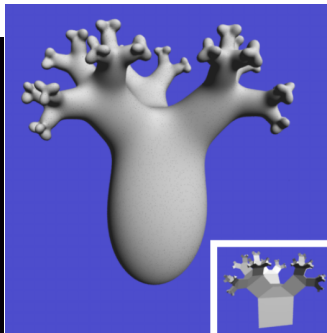
Implicite : Paramétriques

- Analytique
 - Splines
 - RBF

Nécessite une minimisation, pas de contrôle direct $\Rightarrow$ Médical.



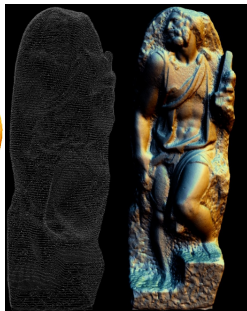
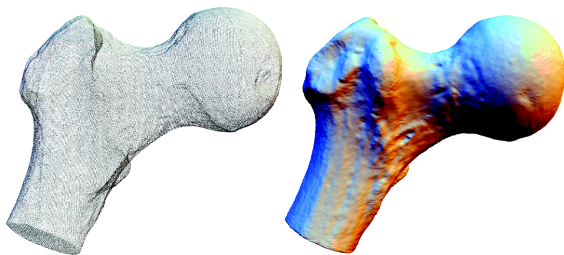
[Sherstyuk, 98]



[Turk, O'Brien, SIGGRAPH 02]

Point-sets = On ne traite en entrée que des positions discrètes de l'espace p_i (et des normales).

⇒ Données scanners.



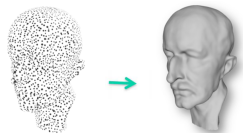
[Boubekeur]

Implicite : Point-sets

Moving Least Squares (MLS) :

- But : Trouver f fonction lisse tel que

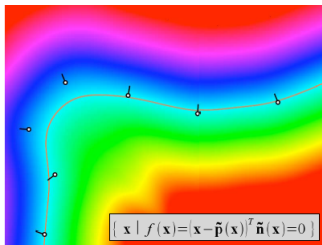
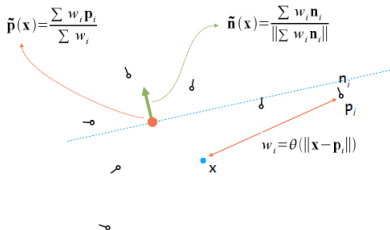
$$f = \operatorname{argmin} \left(\sum_i \psi(\|p_i - x\|) (f(p_i) - f(x))^2 \right)$$



- + Fonctions lisses approximantes
- Minimisation

[Gross]

Application : Données bruitées.

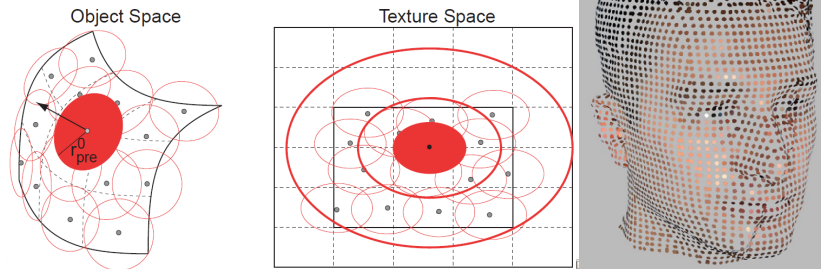


[Alexa]

Surfels

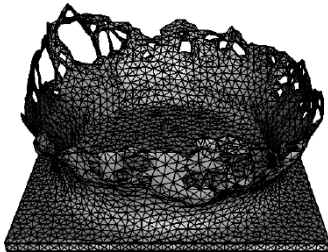
- But : Afficher une surface continue à partir de morceaux simples
- + Affichage rapide
- Pas de surface sous-jacente

Application : Grande masse de données.

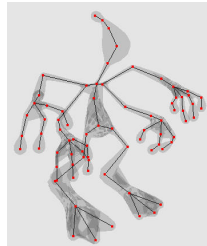


Implicite vs Explicite

- + Topologie arbitraire
- + Mélange de formes
- Manipulation
- Cout en mémoire
- Rendu + cout en temps
- Détails



[Broshu, Batty, Bridson, SCA 09]

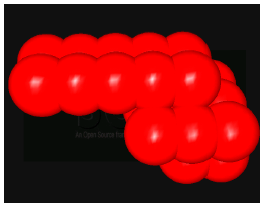
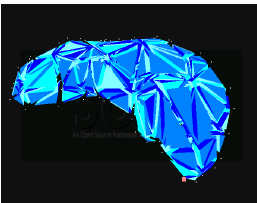
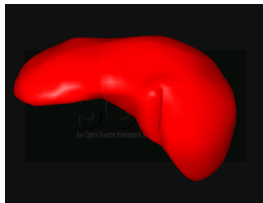
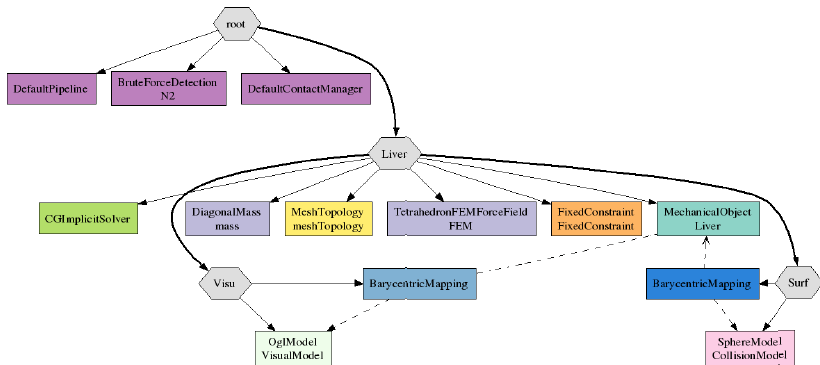


Hornus, Angelidis, Cani, Vis. Comp. 03



[Ohtake, Belyaev, Alexa, Turk, Seidel, SIGGRAPH 03]

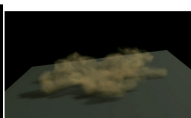
Interactions



Fractales

- Principe : Déformations reccursives convergeant vers un objet complexe.
- Utilitée : Modélisation d'**objet complexes** à partir de **règles simples** et peu nombreuses.

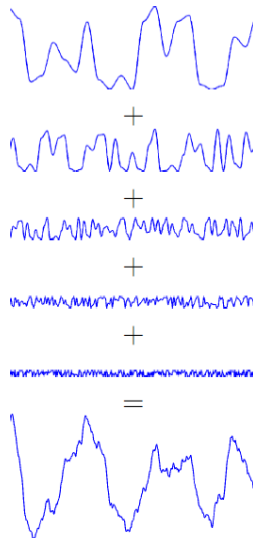
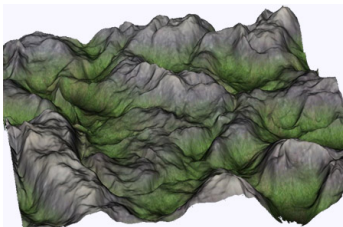
Application : Graphique (modélisation procédurale).



ex. Bruit de Perlin.

$$f(x) = \sum_{k=0}^N \frac{f(a^k x)}{b_k}$$

- N : octaves
- a : fréquence
- 1/b : persistance

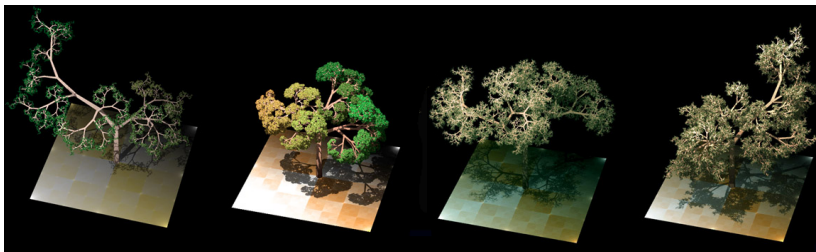
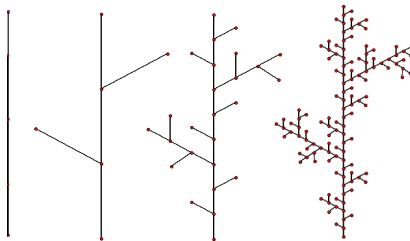


Fractales

ex. L-System.

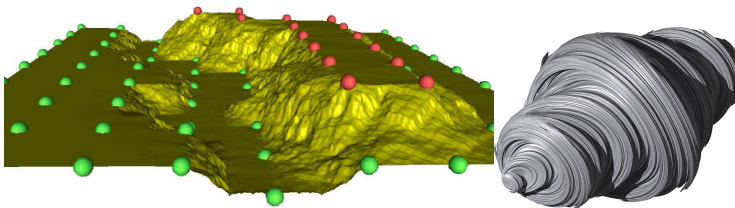
Grammaire :

$F[+F]F[-F]F, \theta = 60^\circ$

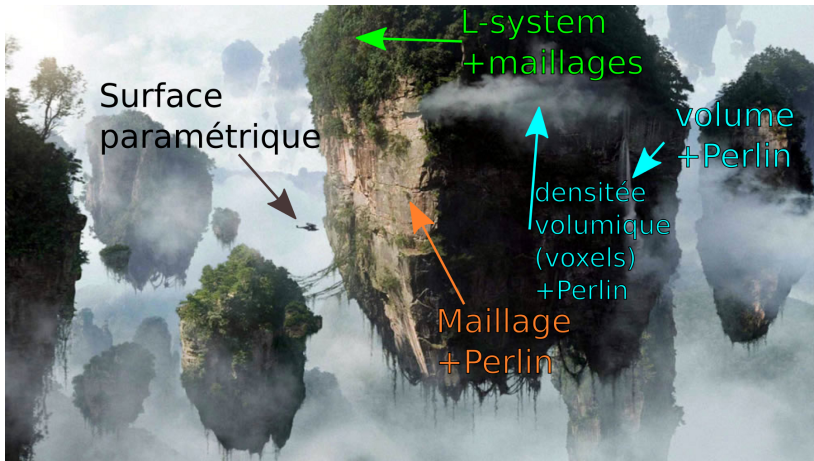


Fractales : Conclusion

- + Objets complexes à partir de règles simples
- + Aspect naturel
- Contrôle



[Hnaidi, Guérin, Akkouche, Fractals 10]



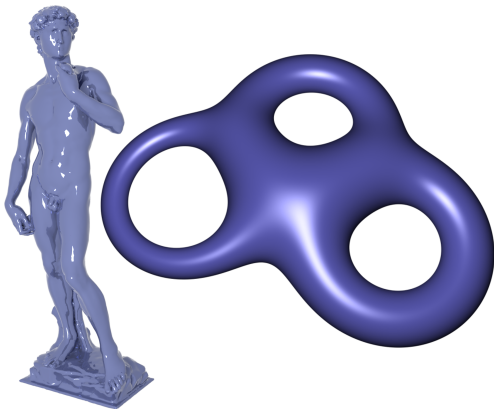
ETI5 Majeure Image: Continuité et courbure des surfaces différentiables

CPE Lyon
damien.rohmer@cpe.fr

4 Novembre 2010

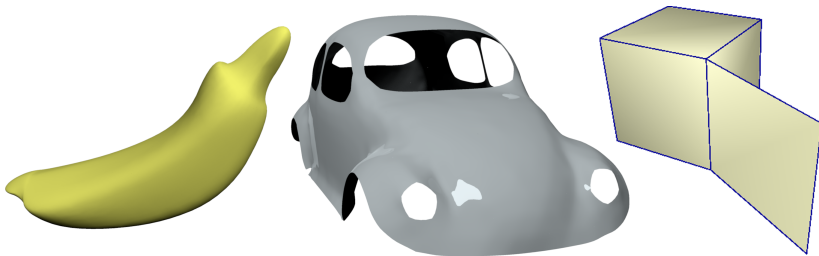
■ **Topologie** = Étude de la structure d'un espace indépendamment de sa géométrie.

- 2 surfaces sont topologiquement équivalentes si on peut transformer l'une en l'autre par des transformations continues (pas de déchirures ni soudures).



Notion de variété

- Une surface Γ est une **2-variété (manifold)** ssi tout point possède un voisinage **homéomorphe** à un (demi) disque.
- Homéomorphisme = application bijective continue + réciproque continue.



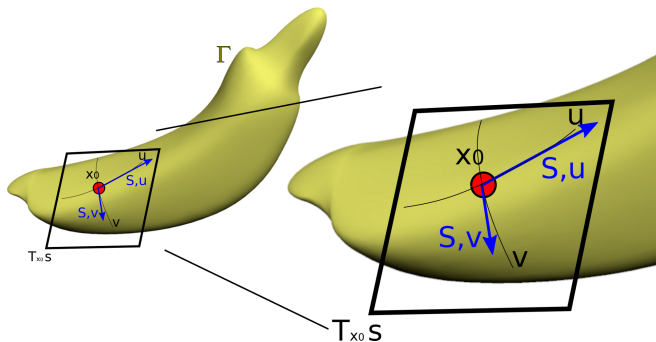
Notion de continuité

- Soit Γ la surface associée au mapping S .

$$S: \begin{cases} \mathcal{D} \subset \mathbb{R}^2 & \rightarrow \mathbb{R}^3 \\ (u, v) & \mapsto S(u, v) \end{cases}$$

$$\Gamma = tr_{\mathcal{D}}(S) = \{\mathbf{x} \in \mathbb{R}^3 \mid \forall (u, v) \in \mathcal{D}, S(u, v) = \mathbf{x}\}$$

- S est $\mathcal{C}^1$ ssi $S_{,u}$ et $S_{,v}$ sont définies et continues.
- S est $\mathcal{C}^2$ ssi $S_{,uu}$, $S_{,vv}$ et $S_{,uv}$ sont définies et continues.



- Γ est $\mathcal{G}^1$ ssi

$$\begin{aligned} \exists (S, \mathcal{D}) \subset \mathcal{C}^1 \times \mathbb{R}^2, \quad tr_{\mathcal{D}}(S) = \Gamma \\ \Leftrightarrow \text{plan tangent partout} \end{aligned}$$

- Γ est $\mathcal{G}^2$ ssi

$$\begin{aligned} \exists (S, \mathcal{D}) \subset \mathcal{C}^2 \times \mathbb{R}^2, \quad tr_{\mathcal{D}}(S) = \Gamma \\ \Leftrightarrow \text{courbure continue} \end{aligned}$$

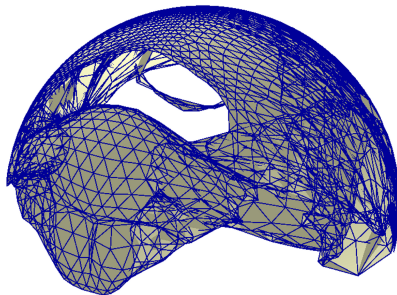
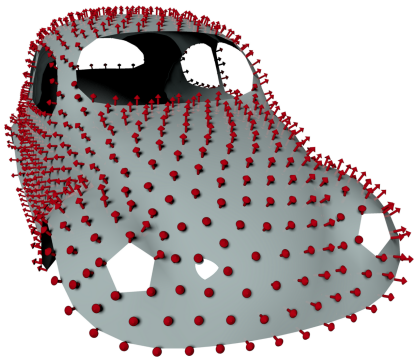
Remarque $S \in \mathcal{C}^k \Rightarrow \Gamma \in \mathcal{G}^k$.

$\mathcal{G}^2$ nécessaire pour reflets.

Plan tangent

- Surface de $\mathbb{R}^3$: $S(u, v) = (S_x(u, v), S_y(u, v), S_z(u, v))$
- Normale : $n^S(u, v) = (S_{,u} \times S_{,v}) / \|S_{,u} \times S_{,v}\| \in \mathbb{S}^2$
- Espace tangent de S en x_0 :
 $T_{x_0} S = \text{Im}(DS(u, v)) = \{S_{,u}(u, v)h_u + S_{,v}(u, v)h_v | (h_u, h_v) \in \mathbb{R}^2\}.$
- Application de Gauss :

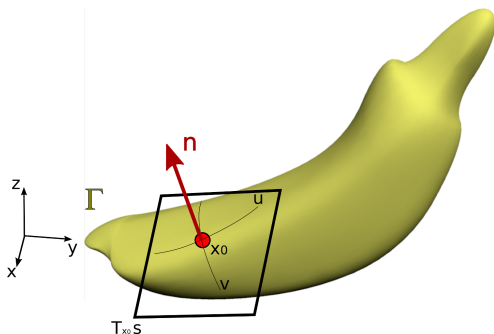
$$N : \begin{cases} \Gamma & \rightarrow \mathbb{S}^2 \\ x_0 = S(u, v) & \mapsto N(x_0) = n(u, v) \end{cases}$$



Propriétés intégrales

Propriétés intégrales :

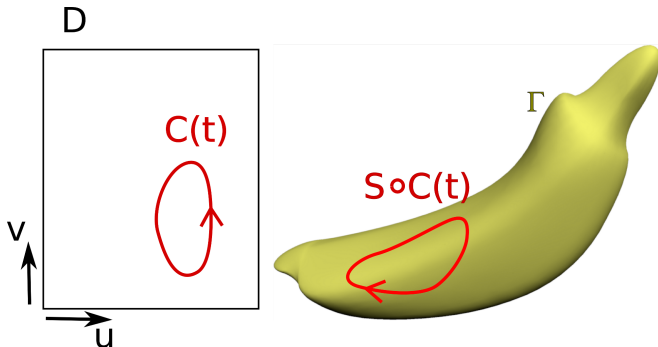
- Aire de Γ : $\int \int_{(u,v) \in \mathcal{D}} \|S_{,u} \times S_{,v}\| du dv$.
- Volume domaine défini par Γ : $\int \int_{(u,v) \in \mathcal{D}} S_z(u, v) n_z^S(u, v) du dv$



Première forme fondamentale

- Courbe $C \subset \mathcal{D}$ de longueur $L = \int_t \langle C'(t), C'(t) \rangle^{1/2} dt$.
- Longueur de $C_S = S(C_x, C_y)$

$$\begin{aligned} L_S &= \int_t \langle (S \circ C)'(t), (S \circ C)'(t) \rangle^{1/2} dt \\ &= \int_t (C'^T(t) I_S(t) C'(t))^{1/2} dt \end{aligned}$$



- I_S : Première forme fondamentale / tenseur métrique

$$I_S = \begin{pmatrix} S_{,u}^2 & \langle S_{,u}, S_{,v} \rangle \\ \langle S_{,u}, S_{,v} \rangle & S_{,v}^2 \end{pmatrix}$$

- I_S : forme quadratique associée à $\langle dS, dS \rangle$

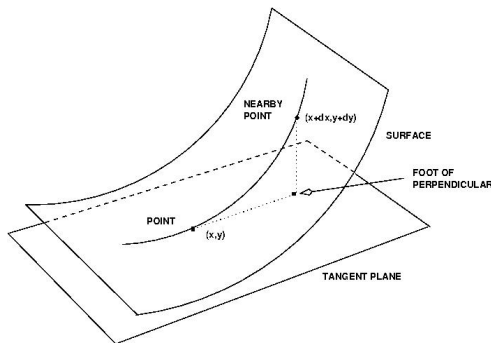
- $\sqrt{\det(I_S)}$ = variation d'aire infinitesimale
 $\Rightarrow$ Aire de $\Gamma = \int \int_{(u,v) \in \mathcal{D}} \sqrt{\det(I)} \, du \, dv$

Seconde forme fondamentale

- II forme quadratique associée à $-\langle dS, dn \rangle$
= développement de Taylor de la surface dans le plan tangent.

$$\text{II}_S = - \begin{pmatrix} \langle n, u, S, u \rangle & \langle n, u, S, v \rangle \\ \langle n, v, S, u \rangle & \langle n, v, S, v \rangle \end{pmatrix}$$

$$\Leftrightarrow \text{II}_S = \begin{pmatrix} \langle S, uu, n \rangle & \langle S, uv, n \rangle \\ \langle S, uv, n \rangle & \langle S, vv, n \rangle \end{pmatrix}$$



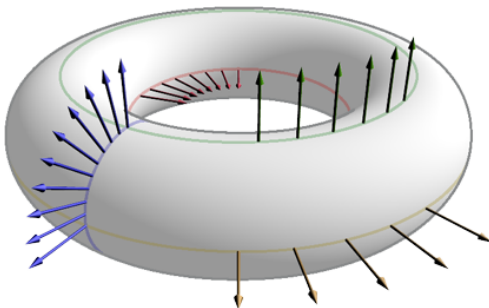
Application de Weingarten

- Application de Weingarten = Différentielle de l'application de Gauss (n) :
- $n_{,u}$ et $n_{,v}$ sont dans le plan tangent.

$$\begin{cases} \det(I) n_{,u} = (I_1 II_1 - I_2 II_0) S_{,u} + (I_1 II_0 - I_0 II_1) S_{,v} \\ \det(I) n_{,v} = (I_1 II_2 - I_2 II_1) S_{,u} + (I_1 II_1 - I_0 II_2) S_{,v} \end{cases}$$

- Shape operator S :

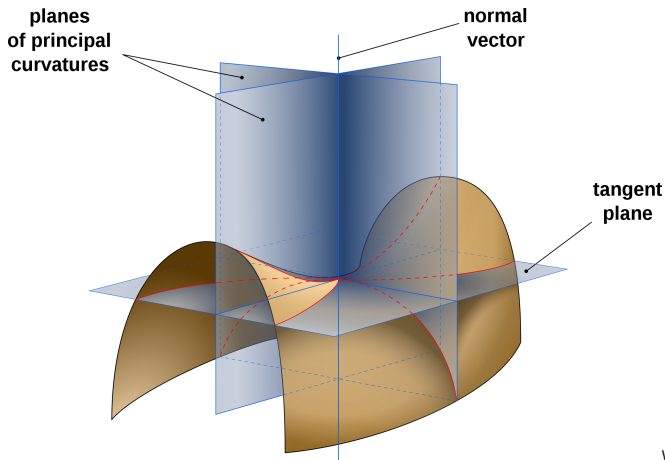
$$S = \frac{1}{\det(I)} \begin{pmatrix} I_1 II_1 - I_2 II_0 & I_1 II_0 - I_0 II_1 \\ I_1 II_2 - I_2 II_1 & I_1 II_1 - I_0 II_2 \end{pmatrix}$$



Wikipedia

Courbure

- Courbure = variation de la normale $\Leftrightarrow$ Weingarten
- S diagonalisable : $S = T^T \text{diag}(\kappa_1, \kappa_2) T$.
- (κ_1, κ_2) = courbures principales.
- T = directions des courbures principales.

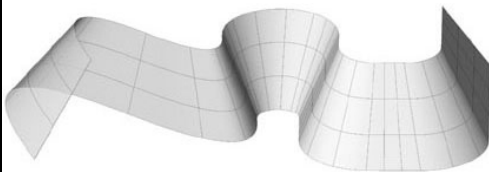
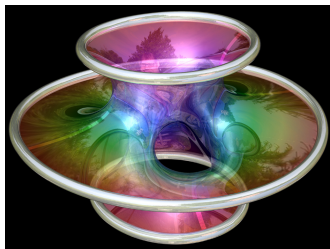


Wikipedia

- Courbure Gaussienne : $K = \kappa_1 \kappa_2 = \det(S) = \frac{\det(\text{II})}{\det(\text{I})}$
- Courbure moyenne :

$$H = \frac{1}{2}(\kappa_1 + \kappa_2) = \frac{1}{2}\text{tr}(S) = \frac{1}{2\det(\text{I})}(\text{II}_0\text{I}_2 + \text{I}_0\text{II}_2 - 2\text{I}_1\text{II}_1)$$

- $H = 0 \Rightarrow$ surface minimale
- 2 Surfaces isométriques on le même K
- $K = 0$ Surface développable



Relation intégrale

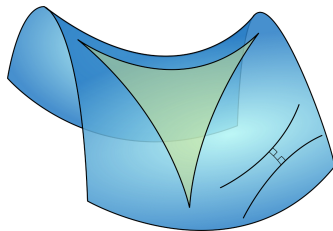
Théorème de Gauss-Bonnet :

$$\int_S K dA + \int_{\partial S} k_g ds = 2\pi\chi(S)$$

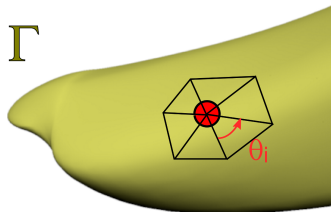
- k_g : courbure géodésique.
- χ : Caractéristique d'Euler (invariant topologique).

Application pour un polygone au voisinage d'un sommet :

$$\frac{1}{A} \left(\sum_i \theta_i - 2\pi \right) \simeq K$$



Wikipedia



ETI5 Majeure Image: Maillage

CPE Lyon
damien.rohmer@cpe.fr

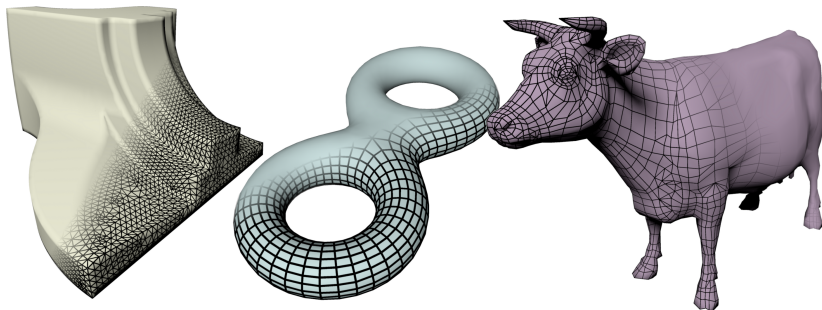
18 Novembre 2010

Plan du cours

- 1 Introduction Maillages polygonaux
- 2 Élement de base : Triangle
- 3 Description d'un maillage
- 4 Textures
- 5 Softwares

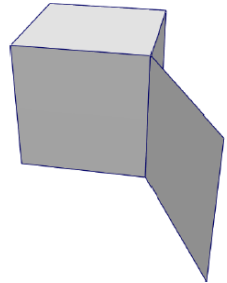
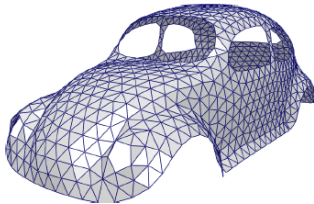
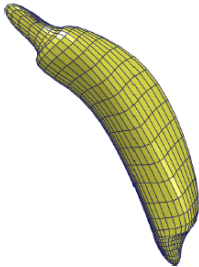
Maillage

- Maillage (Mesh) = Ensemble de **polygones** partageants certains sommets
- N_f **faces**, N_s **sommets** (vertices), N_e **arêtes** (edges).
- **Triangulation** : toutes les faces sont des triangles.
- **Quad-mesh** : toutes les faces sont des quads.
- Poly-mesh : mélange de types.



Topologie

- Rappel : Surface variété (Manifold) ssi le voisinage de tout point est homéomorphe à un (demi) disque.
⇒ Toute arête est partagée par au plus 2 faces (connectivité) + non auto-intersection (plongement).



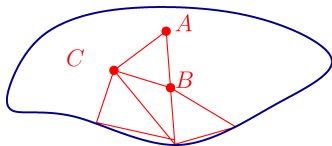
Maillage

- Poly-mesh = cas particulier de triangulation
- Rappel : Triangulation = Mapping linéaire S

$$S_i : \begin{cases} \mathcal{D} \subset \mathbb{R}^2 & \rightarrow \mathbb{R}^3 \\ (u, v) & \mapsto S_i(u, v) = u\vec{AB} + v\vec{AC} \end{cases}$$
$$\mathcal{D} : 0 \leq u + v \leq 1$$

Propriétés :

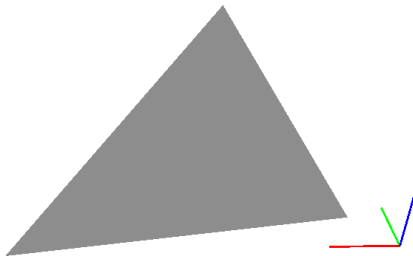
- Surface globalement $\mathcal{G}^0$.
- Surface jamais $\mathcal{G}^1$ (sauf plan).
- **Interpolation linéaire** de $\mathbb{R}^2$ vers $\mathbb{R}^3$ (normales, couleurs, textures).



Triangles en OpenGL

■ Syntaxe OpenGL (mode immédiat)

```
glBegin(GL_TRIANGLES);  
glNormal3d(0,0,1);  
glVertex3d(0,0,0);  
glVertex3d(1,0,0);  
glVertex3d(0,1,0);  
glEnd();
```



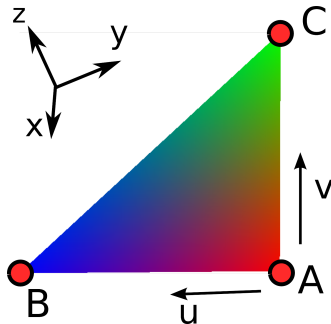
Interpolation linéaire

■ Interpolation de couleurs

$$\begin{cases} r(u, v) &= (1 - u - v)r_A + ur_B + vr_C \\ g(u, v) &= (1 - u - v)g_A + ug_B + vg_C \\ b(u, v) &= (1 - u - v)b_A + ub_B + vb_C \end{cases}$$

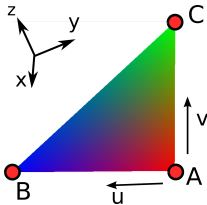
■ Dans le cas général pour une fonction f

$$f(u, v) = (1 - u - v)f_A + uf_B + vf_C$$



Interpolation de couleurs

```
glBegin(GL_TRIANGLES);  
glNormal3d(0,0,1);  
glColor3d(1,0,0);  
glVertex3d(0,0,0);  
glColor3d(0,1,0);  
glVertex3d(1,0,0);  
glColor3d(0,0,1);  
glVertex3d(0,1,0);  
glEnd();
```

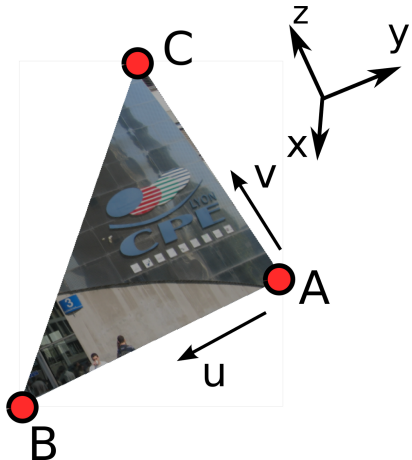
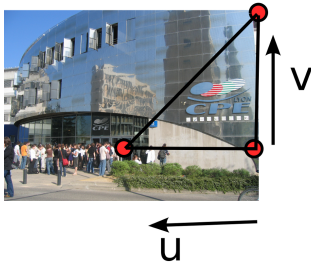


Interpolation linéaire

- On peut interpoler des coordonnées !

⇒ Textures

$$\begin{cases} t_x = (1 - u - v)t_x(A) + ut_x(B) + vt_x(C) \\ t_y = (1 - u - v)t_y(A) + ut_y(B) + vt_y(C) \end{cases}$$



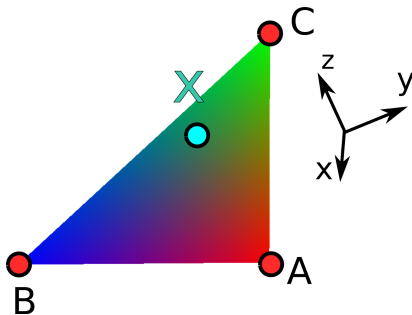
Coordonnées barycentriques

- Étant donné un point $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$: connaître (α, β, γ) tel que $\mathbf{x} = \alpha\mathbf{x}_A + \beta\mathbf{x}_B + \gamma\mathbf{x}_C$, $(\alpha + \beta + \gamma = 1)$.
 $\Rightarrow$ coordonnées barycentriques.

$$\begin{cases} A = \text{aire}(\mathbf{x}_B - \mathbf{x}_A, \mathbf{x}_C - \mathbf{x}_A) \\ A_1 = \text{aire}(\mathbf{x}_C - \mathbf{x}_B, \mathbf{x} - \mathbf{x}_B) \\ A_2 = \text{aire}(\mathbf{x}_A - \mathbf{x}_C, \mathbf{x} - \mathbf{x}_C) \\ A_3 = \text{aire}(\mathbf{x}_B - \mathbf{x}_A, \mathbf{x} - \mathbf{x}_A) \end{cases}$$

avec $\text{aire}(\mathbf{v}_0, \mathbf{v}_1) = 1/2 \|\mathbf{v}_0 \times \mathbf{v}_1\|$

$$\Rightarrow \begin{cases} \alpha = A_1/A \\ \beta = A_2/A \\ \gamma = A_3/A \end{cases}$$

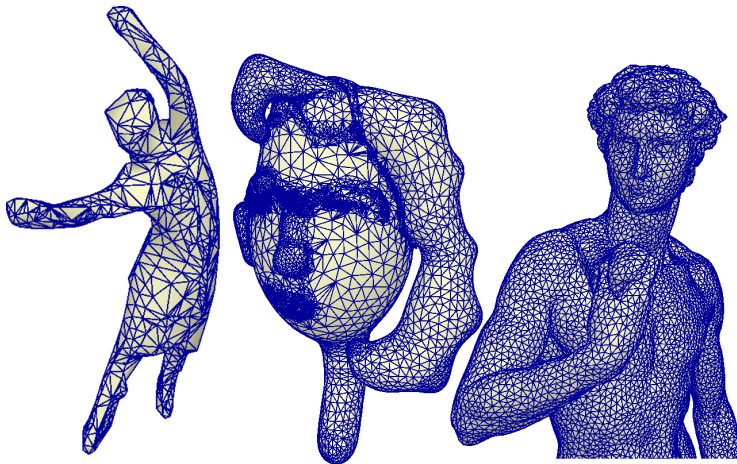


Qualité d'un maillage

- Triangulation : $\theta_{\min} \simeq 30^\circ$

- Quads : $\theta_{\min} \simeq 45^\circ$

Application : Calculs (FEM), (Rendu)



Maillage

Structure de données :

■ *Ex. Représenter un tétraèdre :*

Idée 1 :

```
(0.0,0.0,0.0) , (1.0,0.0,0.0) , (0.0,0.0,1.0)
(0.0,0.0,0.0) , (0.0,0.0,1.0) , (0.0,1.0,0.0)
(0.0,0.0,0.0) , (0.0,1.0,0.0) , (1.0,0.0,0.0)
(0.0,1.0,0.0) , (0.0,0.0,1.0) , (1.0,0.0,0.0)
```



Il y a mieux :

coordonnees:

```
(0,0,0) , (1,0,0) , (0,1,0) , (0,0,1)
```

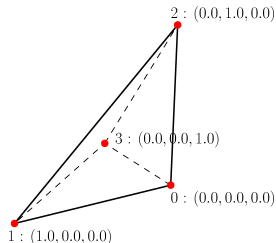
Connectivite

```
(0,1,3)
```

```
(0,3,2)
```

```
(0,2,1)
```

```
(1,2,3)
```



Format off

Exemple de format d'echange. Format off.

OFF

8 6 12

0 0 0

1 0 0

0 1 0

0 0 1

1 1 0

1 0 1

0 1 1

1 1 1

4 0 1 4 2

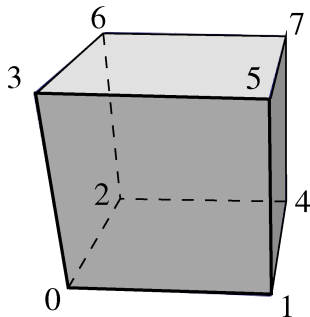
4 1 5 7 4

4 3 6 7 5

4 2 6 3 0

4 2 4 7 6

4 0 3 5 1



Lecture fichier

```
while(k_vertex<N_vertex)
{
    fscanf(fid,"%f %f %f",X,X+1,X+2);
    add_vertex(X[0],X[1],X[2]);
    k_vertex++;
}
for(k_poly=0;k_poly<N_poly;k_poly++)
{
    fscanf(fid,"%d",&size_poly);
    std::vector v_poly;
    for(k=0;k<size_poly;k++)
    {
        fscanf(fid,"%d",&temp);
        v_poly.push_back(temp);
    }
    add_polygon(v_poly);
}
```

Structure de données

- Vecteurs contigus dans la mémoire : Affichage rapide en OpenGL.

```
// (x0,y0,z0,x1,y1,z1,...)
std::vector <double> vertex

// (i00,i01,i02,i10,i11,i12,...)
std::vector <int> connectivity

std::vector <double> normal, color, texture ...
```

- Accès à la coordonnée y du sommet k .
`vertex[3*k+1]`
- Accès à la coordonnée y du sommet $s(1,2 \text{ ou } 3)$ du triangle t .
`vertex[3*connectivity[3*t+s]+1]`

Normale d'un maillage

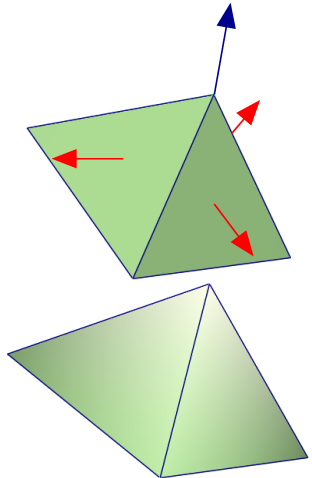
- Aspect lisse
⇒ 1 normale par sommet.
- Moyenne de normales (faux mais répandue)

$$\mathbf{n}_k = \frac{\sum_{i \in \mathcal{V}(k)} \mathbf{n}_i}{\left\| \sum_{i \in \mathcal{V}(k)} \mathbf{n}_i \right\|}$$

k : indice sommet

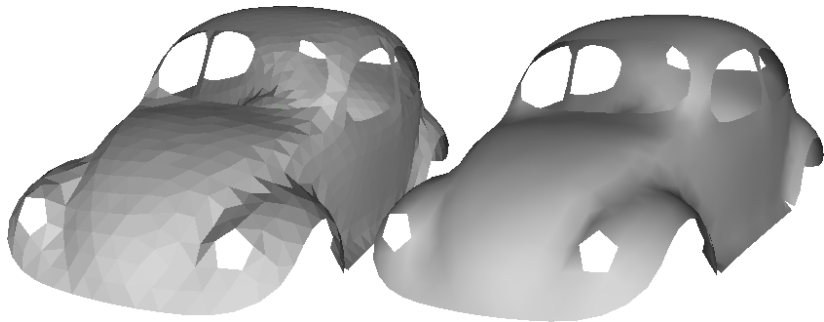
i : indice face

$\mathcal{V}(k)$: faces voisines du sommet k



Normales

- En OpenGL : 1 Normale interpolé par sommets
⇒ Normale par polygone = Plusieurs normales par sommet



Version lente

```
glBegin(GL_TRIANGLES);  
  for(k_tri=0;k_tri<N_tri;k_tri++)  
    for(k_vertex=0;k_vertex<3;k_vertex++)  
      for(k_dim=0;k_dim<3;k_dim++)  
      {  
        x[k_dim] = vertex[3*connectivity  
                               [3*k_tri+k_vertex]+k_dim];  
        n[k_dim] = normal[3*connectivity  
                               [3*k_tri+k_vertex]+k_dim];  
  
        glNormal3d(n[0],n[1],n[2]);  
        glVertex3d(x[0],x[1],x[2]);  
      }  
glEnd();
```

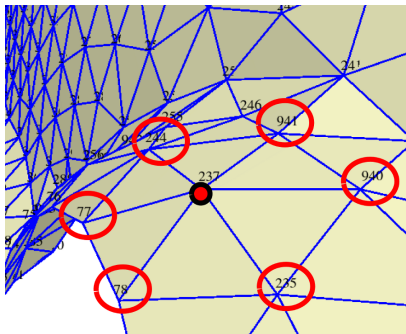
Version Rapide

```
glEnableClientState(GL_VERTEX_ARRAY);  
glVertexPointer(3, GL_DOUBLE, 0, &vertex[0]);  
  
glEnableClientState(GL_NORMAL_ARRAY);  
glNormalPointer(GL_DOUBLE, 0, &normal[0]);  
  
glDrawElements(GL_TRIANGLES, 3*N_tri,  
               GL_UNSIGNED_INT, &connectivity[0]);  
  
glDisableClientState(GL_VERTEX_ARRAY);  
glDisableClientState(GL_NORMAL_ARRAY);
```

Structure de données : Voisinage

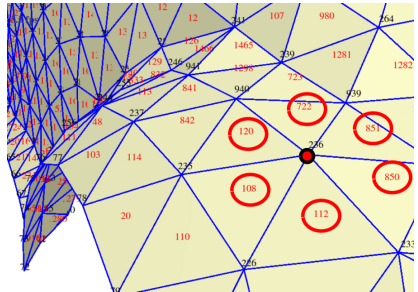
■ 1-Voisinage = Sommets voisins d'un sommet donné

```
std::vector <std::vector <int>> one_ring  
  
// exemple pour le cube:  
one_ring[0] = [1,2,3]  
one_ring[1] = [5,4,0]  
...
```



Voisinage

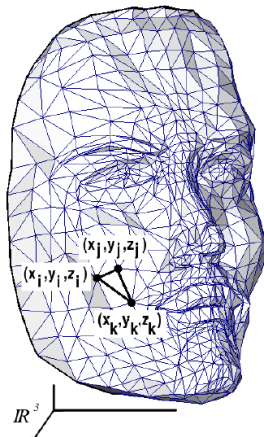
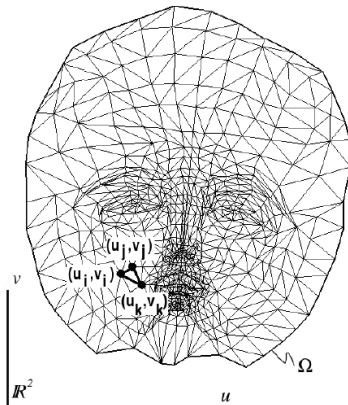
- Triangles voisins d'un autre
- Triangles voisins d'un point : étoilé (1-star)
 - ⇒ calcul des normales !



! Attention aux structures de données.
Compromis entre : temps accès / temps recherche / espace
memoire / facilité ...

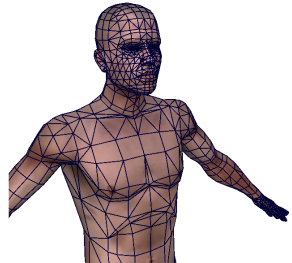
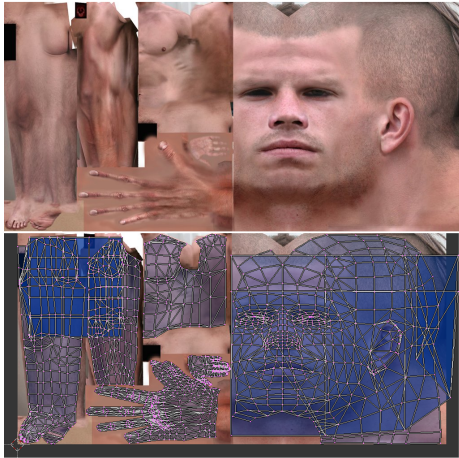
Parametrisation / Textures

- Paramétrisation d'un maillage = Construction de S (par morceaux) étant donné Γ .

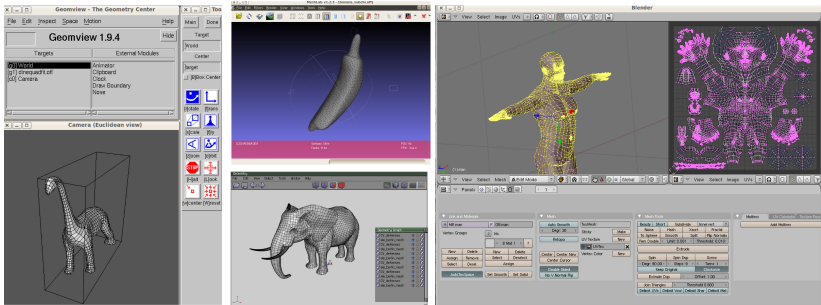


Textures

- Morceaux se recouvrants = atlas (charts).



Softwares



- Geomview (Viewer)
- Meshlab (Mesh Processing)
- Wings3D (Subdivision)
- Blender (Artiste)

ETI5 Majeure Image: Maillage avancé

CPE Lyon
damien.rohmer@cpe.fr

18 Novembre 2010

- Caractéristiques topologiques.
- Structure de données avancées : Halfedge.
- Subdivision de maillage.
- Notion de calcul sur variété : Lissage Laplacien.

Caractéristique topologique

Caractéristique d'Euler-Poincaré

- Maillage à N_f faces, N_s sommets, N_a arêtes.

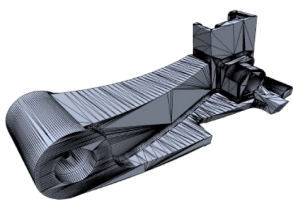
$$N_s - N_a + N_f = \chi = 2(c - g) - b$$

χ : Caractéristique d'Euler (Rappel : Gauss-Bonnet).

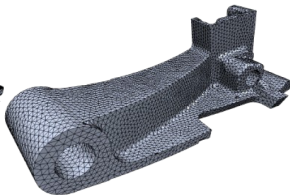
c : Nbr composante connexes

g : Nbr de trous (genre topologique)

b : Nbr de bords



Hetroy



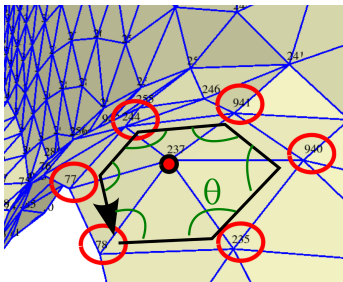
Wikipedia

Structure de données

Limitation encodage indexé

Encodage géométrie + connectivité indexée

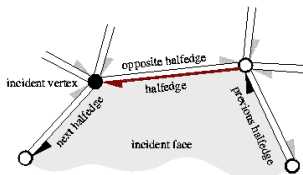
- + Affichage rapide (si contigu)
- + Générique
- + Simple
 - Voisinage non encodé
 - Ajout/suppression en $\mathcal{O}(N)$.



```
OFF
40 95 75
-0.175114 -0.047799 -0.045492
-0.199566 0.730914 -0.064795
-0.010689 0.674496 0.008900
-0.015538 0.153071 0.107408
-0.070148 0.767894 -0.116107
-0.053836 -0.792815 0.109714
-0.162416 -0.785481 0.088014
-0.112365 -0.792492 0.135482
-0.240928 0.031451 0.031966
-0.259289 0.200557 0.035420
0.296891 -0.707385 0.143375
-0.190129 -0.069002 0.109358
-0.010148 0.024179 -0.067283
-0.112968 -0.089127 0.092391
-0.185828 0.377372 -0.111155
3 20 4 1
3 34 11 13
3 12 30 0
3 30 13 17
3 23 22 21
3 29 38 17
3 32 0 13
3 14 0 37
3 24 4 21
3 14 32 1
3 24 2 22
3 3 12 25
3 4 24 15
3 21 15 26
3 35 34 13
3 19 32 13
3 19 13 27
```

Encodage Halfedge

- Encodage Demi-Arêtes (Halfedge)
- Encodage des arêtes : On retrouve les faces par parcours (variété).
- Ajout/suppression en $\mathcal{O}(1)$.



Vertex
Halfedge_handle halfedge()

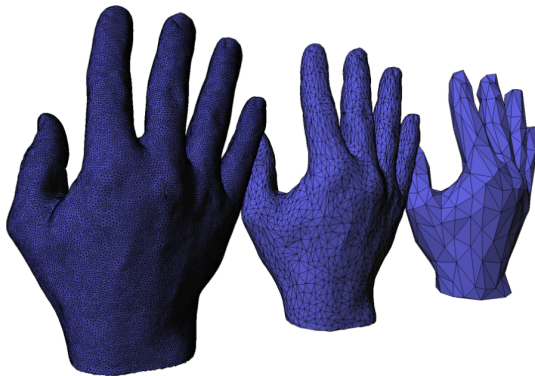
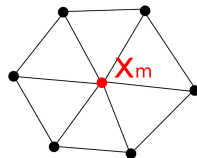
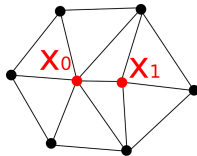
Halfedge
Halfedge_handle opposite()
Halfedge_handle next()
Halfedge_handle prev()
Vertex_handle vertex()
Face_handle face()

Face
Halfedge_handle halfedge()

CGAL

Edge Collapse

- Suppression d'arête (edge collapse) = Base de la simplification de maillage



CGAL

Encodage Halfedge comparaison

Choisir la structure la plus adaptée :

■ Tableaux d'indices (vecteurs contigus)

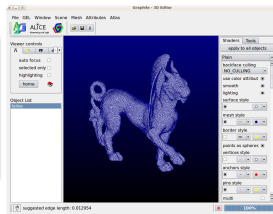
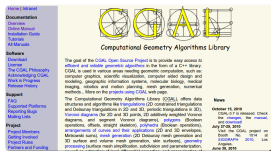
- + Simple, général, adapté GPU.
- + Accès aléatoire $\mathcal{O}(1)$.
- Parcours voisinage en $\mathcal{O}(N)$.
- Ajout/Suppression $\mathcal{O}(N)$.

■ Halfedge

- + Parcours voisinage en $\mathcal{O}(1)$.
- + Ajout/Suppression en $\mathcal{O}(1)$.
- Technique, variété uniquement, encodage non contigu.
- Accès aléatoire en $\mathcal{O}(N)$.

```
Halfedge_handle g = h->next()->opposite()->next();
P.split_edge( h->next());
P.split_edge( g->next());
P.split_edge( g);
h->next()->vertex()->point()    = Point( 1, 0, 1);
g->next()->vertex()->point()    = Point( 0, 1, 1);
g->opposite()->vertex()->point() = Point( 1, 1, 0);
Halfedge_handle f = P.split_facet( g->next(),
                                   g->next()->next()->next());
Halfedge_handle e = P.split_edge( f);
e->vertex()->point() = Point( 1, 1, 1);
P.split_facet( e, f->next()->next());
```

Libraries implémentant le Halfedge

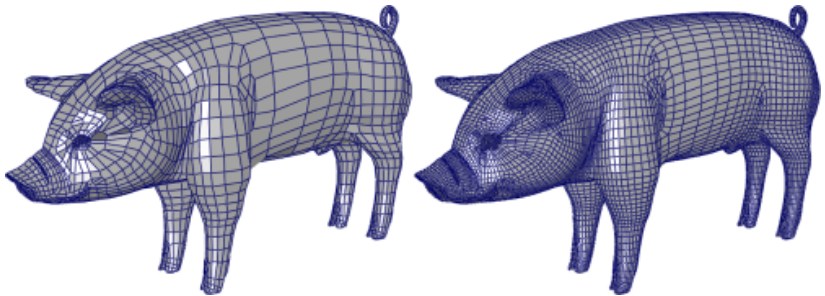


- CGAL (C++ complexe, calcul exact, beaucoup d'algos)
- Graphite (remaillage, parametrisation, GUI)
- OpenMesh (plus simple que CGAL, moins complet)

Subdivision

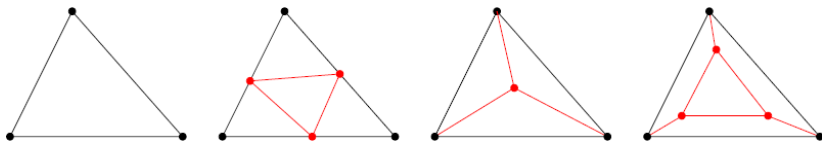
Subdivision de maillage

- Raffiner pour : rendu, calcul, déformer, ...
- Voir cours SIGGRAPH



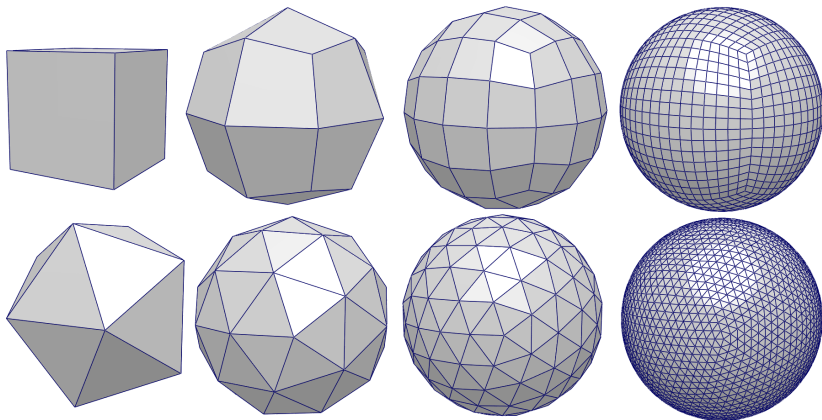
Subdivision de maillage

- Subdivision de la connectivité.
- Plusieurs possibilités de subdivision.



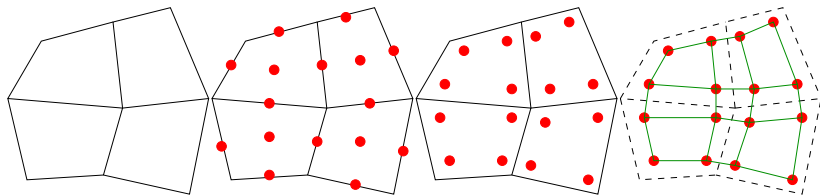
- Présences de polygones quelconques ...
- Structure de données !
- Géométrie
 - Schémas interpolants
 - Schémas approximants

- Application à la génération d'une sphère par projection :



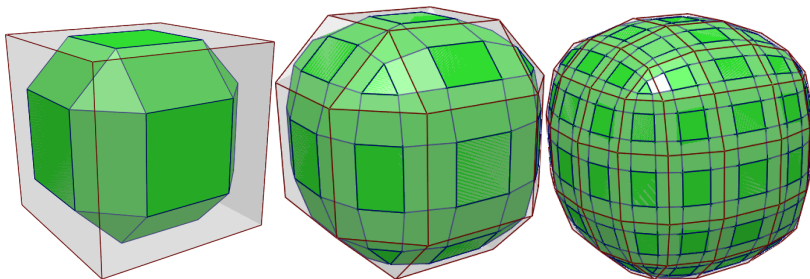
Exemple : Doo-Sabin

- 1 Soit une face formée par $(x_i)_{i=\llbracket 0, N-1 \rrbracket}$.
- 2 Construit sommets milieux $m_i = (x_i + x_{i+1})/2$.
- 3 Barycentre de la face $b = (\sum_i x_i)/N$.
- 4 Nouveaux sommets = $n_i = (x_i + m_i + m_{i-1} + b)/4$.

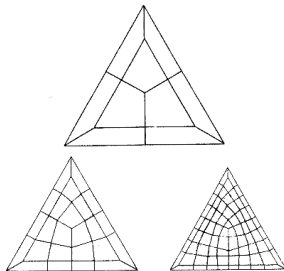


Subdivision

Exemple : Doo-Sabin

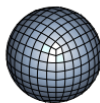
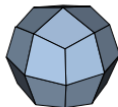
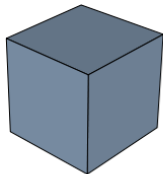


- Un sommet de face = barycentre de l'ancienne face
 - Un sommet d'arête = moyenne entre le milieu des anciens sommets et le milieu des faces partageant l'arête
 - Nouvelle position du sommet = $\frac{Q+2R+S(n-3)}{n}$
-
- Q : Moyenne des sommets de face adjacents
 - R : Moyenne des points milieux des arêtes incidentes
 - S : Anciennes coordonnées
 - n : valence



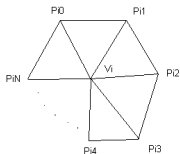
Catmull-Clark Resultats

- C^2 sauf au sommets extra-ordinaires
- Schema d'approximation
- Subdivision de face
- Basé de préférence sur des quads



Maillage Triangulaire

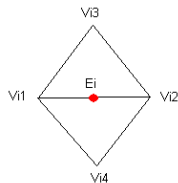
■ Nouvelle Position



$$V^{i+1} = (1 - n\alpha)V^i + \alpha \sum_{k=0}^n P_k$$

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

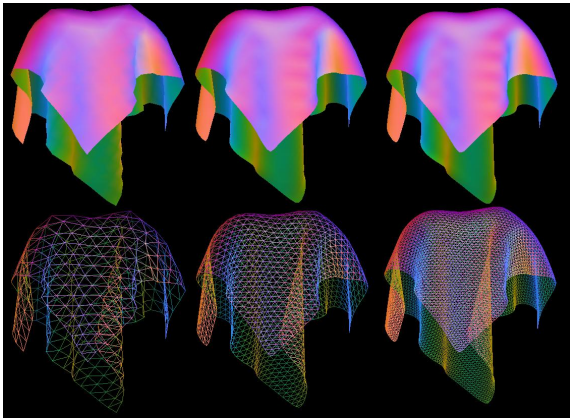
■ Sommets d'arêtes



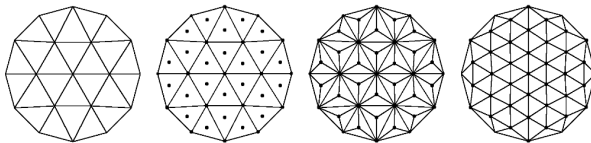
$$E^{i+1} = \frac{3}{8}(V_1 + V_2) + \frac{1}{8}(V_3 + V_4)$$

Loop

- C^2 sauf au sommets extra-ordinaires
- Schema d'approximation
- Subdivision de face
- Subdivision de triangles



Maillage Triangulaire

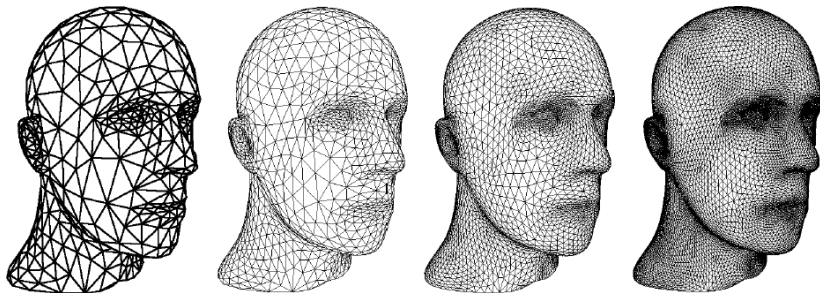


- Nouveaux sommets : barycentre de l'ancienne face
- Nouvelle position du sommet i :

$$(1 - \alpha_n)p_i + \frac{\alpha_n}{n} \sum_{j \in \mathcal{V}_i} p_j \quad (n=\text{valence}, \mathcal{V}=\text{voisinage})$$

$$\text{et } \alpha_n = \frac{4 - 2 \cos\left(\frac{2\pi}{n}\right)}{9}$$

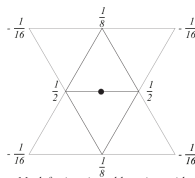
- C^2 sauf au sommets extra-ordinaires
- Schema d'approximation
- Subdivision de face
- Subdivision de triangles



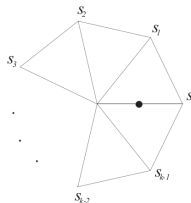
[Kobbelt, SIGGRAPH 00]

Maillage Triangulaire

■ Sommet régulier



■ Sommet non régulier



■ Ajout d'un point par coté

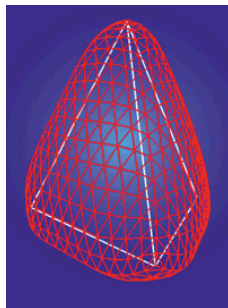
■ Dans le cas d'un sommet non régulier :

$$s_i = \frac{1}{k} \left(\frac{1}{4} + \cos \left(\frac{2\pi}{k} \right) + \frac{1}{2} \cos \left(\frac{4\pi}{k} \right) \right) \quad \text{pour } k > 5$$

$$s_0 = \frac{5}{12}, s_{1,2} = -\frac{1}{12} \quad \text{pour } k = 3$$

$$s_0 = \frac{3}{8}, s_2 = -\frac{1}{8}, s_{1,3} = 0 \quad \text{pour } k = 4.$$

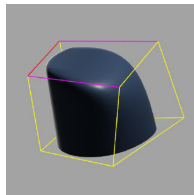
- C^1 sauf au sommets extra-ordinaires
- Schema d'interpolation
- Subdivision de face
- Subdivision de triangles



Subdivision : Arêtes vives

- Ne pas lisser toutes les arêtes :

[De Rose, Kass, Truong.
**Subdivision Surfaces in
Character Animation.** *ACM
SIGGRAPH*. 1998]

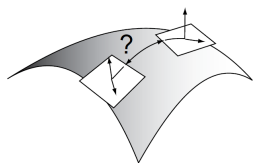


Lissage Laplacien

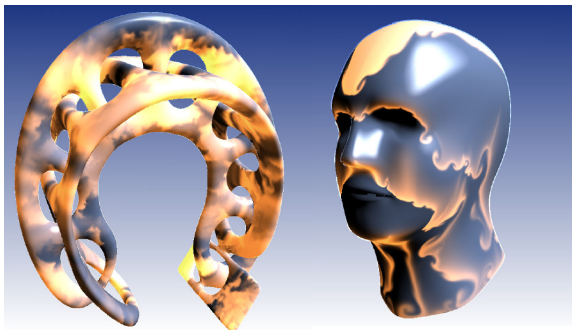
Lissage Laplacien

- Soit f définie sur une variété différentielle (surface lisse).

$$f : \begin{cases} \Gamma & \rightarrow \mathbb{R}^N \\ (\xi_1, \xi_2) & \mapsto f(\xi_1, \xi_2) \end{cases}$$



- Calcul différentiel ?
- Problème : Comparer vecteurs dans différents espaces



- Opérateur de Laplace-Beltrami = Laplacien sur une variété

$$\Delta = \frac{1}{\sqrt{\det(I_{\Gamma})}} \sum_i \frac{\partial}{\partial \xi_i} \left(\sqrt{\det(I_{\Gamma})} \sum_j I_{\Gamma}^{ij} \frac{\partial}{\partial \xi_j} \right)$$

- Cas particulier : Laplacien des propres coordonnées !
 $f = \mathbf{x}(\xi_1, \xi_2) = (x(\xi_1, \xi_2), y(\xi_1, \xi_2), z(\xi_1, \xi_2))$
- $\text{Sp}(\Delta \mathbf{x}) = \text{modes de vibrations propres} = \text{Base de Fourier}$
Théorie Spectrale



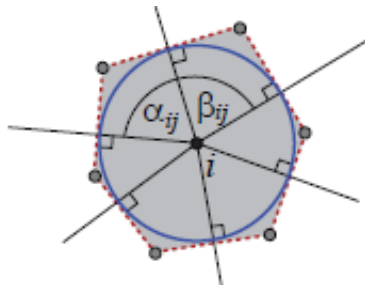
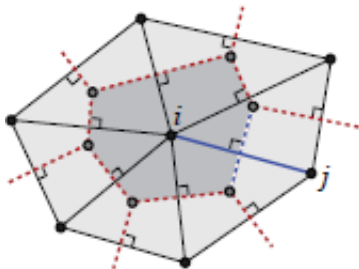
Lissage Laplacien

- Filtre passe bas = Convolution par filtre Gaussien en 2D = Solution de l'équation de diffusion (voir *Scale Space theory*).

$$\frac{\partial x}{\partial t} = \Delta \mathbf{x}$$

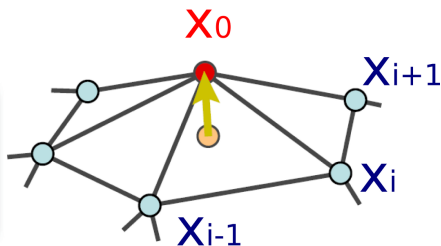
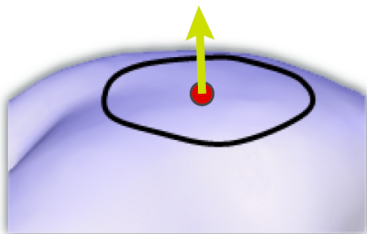
- Comment approximer Δ sur un maillage ?
⇒ Plusieurs solutions, aucune parfaite.

[Wardetzky, Mathur, Kalberer, Grinspun. **Discrete Laplace Operators : No Free Lunch.** *SGP 07*]



- Approximation la plus simple :

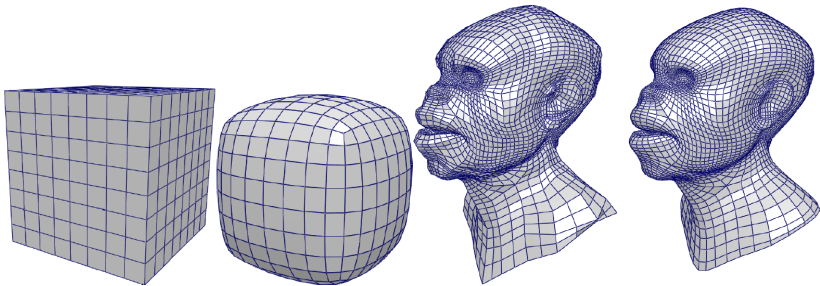
$$\Delta \mathbf{x}(\mathbf{x}_0) \simeq \frac{1}{N} \sum_i (\mathbf{x}_i - \mathbf{x}_0) = \text{bar} - \mathbf{x}_0$$



[Sorkine, Eurographics 05]

- Algorithme du lissage laplacien :

$$\mathbf{x}^{k+1} = \mu \mathbf{\bar{x}} + (1 - \mu) \mathbf{x}^k$$



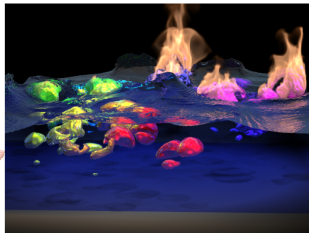
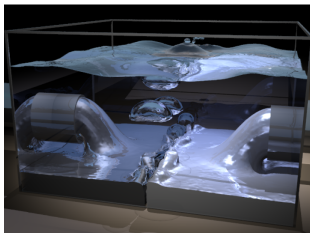
ETI5 Majeure Image: Animation physique

CPE Lyon
damien.rohmer@cpe.fr

26 Novembre 2010

Animation physique

- Quand on ne sait pas le déformer *à la main*.
- Quand il faut modéliser des phénomènes physiques.



[Fedkiw SIGGRAPH 06],[Grinspun SIGGRAPH 07]

Principe générale

- 1 Défini un système S_0 (positions, vitesses, forces, ...) à l'instant $t = 0$.
- 2 On modélise son évolution temporelle par une équation aux dérivées partielles (EDP / PDE)

$$\mathcal{F} \left(S(t), \frac{\partial^n S}{\partial t^n}, \frac{\partial^m S}{\partial x^m} \right) = 0$$

- Généralement on pourra écrire

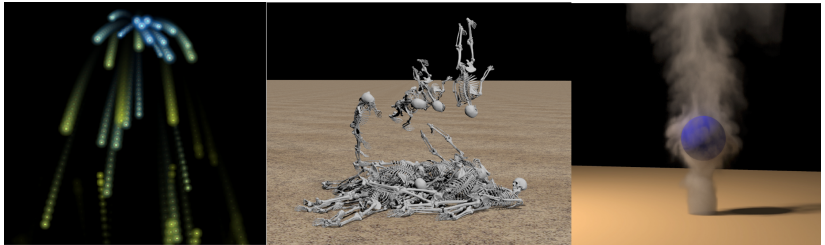
$$\frac{\partial S}{\partial t} = AS + \mathcal{H} \left(S(t), \frac{\partial^m S}{\partial x^m} \right)$$

- 3 On résoud numériquement le système pour avancer temporellement suivant un pas Δt .

Approximation

Différents niveaux d'approximations

- 1 **Méthodes particulières** (particle system) = on réduit tout à un point (pas de rotation + EDO) : le + simple, le moins fidèle.
- 2 **Mécanique du solide** (rigid bodies) = paramètres identiques pour tout le solide.
- 3 **Mécanique des milieux continus** (continuum mechanics) = paramètres variables dans la forme : EDP (FEM, FV, ...).



[Fedkiw IEEE TVCG 06],[Fedkiw SIGGRAPH 01]

Système de particules

1-Description du système :

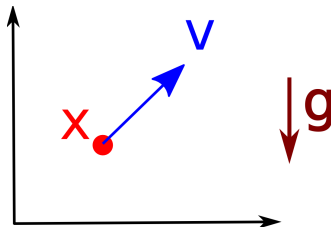
- Particule de masse m tombant en chute libre. Systeme décrit par $u = (x, v)$.
 - Position initiale $x(t = 0) = x_0$.
 - Vitesse initiale $v(t = 0) = v_0$.

2-Equation différentielle :

- Équation de la dynamique :

$$ma = \sum F = mg$$

$$\Rightarrow \begin{cases} x'(t) = v(t) \\ v'(t) = g \end{cases}$$



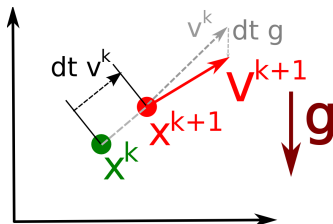
Résolution numérique directe :

$$\begin{cases} v^{k+1} = v^k + (\Delta t)g \\ x^{k+1} = x^k + (\Delta t)v^k \end{cases}$$

Implémentation triviale

```
x=x0;  
v=v0  
for (k=0; k<N; ++k)  
{  
    x=x+dt*v;  
    v=v+dt*g;  
}
```

Quel erreur commet-on en résolvant numériquement ?



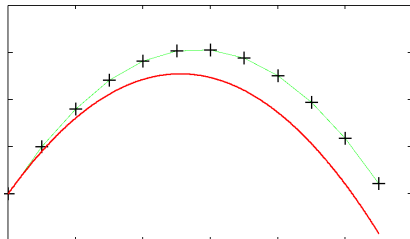
Equation résolue

Equation numérique :

$$\begin{cases} v^{k+1} = v^k + (\Delta t)g \\ x^{k+1} = x^k + (\Delta t)v^k \end{cases}$$

Réccursion sur x^k :

$$\Rightarrow \begin{cases} x^{k+2} = 2x^{k+1} - x^k + (\Delta t)^2 g \\ x^0 = x_0 \text{ et } x^1 = x_0 + \Delta t v_0 \end{cases}$$



Par récurrence :

$$\Rightarrow x(t = k\Delta t) = x_0 + (k\Delta t) v_0 + \frac{k(k-1)}{2}(\Delta t)^2 g$$

Or solution exacte de $\frac{\partial x}{\partial t}(t) = g$: $\tilde{x}(t) = 1/2gt^2 + v_0t + x_0$.

$$\tilde{x}(t = k\Delta t) = x_0 + (k\Delta t)v_0 + \frac{1}{2}(k\Delta t)^2 g$$

Précision

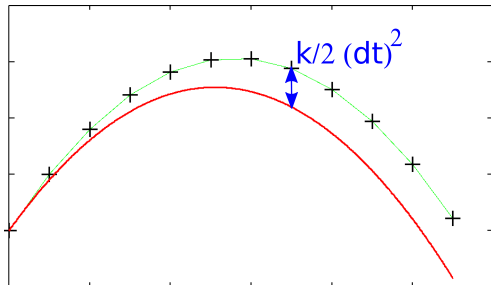
- Erreur :

$$\|x(k\Delta t) - \tilde{x}(k\Delta t)\| = \frac{k}{2}(\Delta t)^2$$

- Précision (accuracy) d'une méthode d'intégration d'ordre h :

$$\|x(k\Delta t) - \tilde{x}(k\Delta t)\| = \mathcal{O}((\Delta t)^{h+1})$$

- Notre schéma d'intégration est d'ordre 1. On ne résout pas l'équation continue !



Approche matricielle :

- On ne traite qu'avec une équation d'ordre 1 en posant $u = (x, v)$:

$$\frac{\partial u}{\partial t}(t) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} u(t) + \begin{pmatrix} 0 \\ g \end{pmatrix}$$

$$\Leftrightarrow u_{,t} = Au + b$$

- 2 Possibilités de discrétisation temporelle :
 - $(u^{k+1} - u^k)/\Delta t = Au^k + b$: Euler explicite
 - $(u^{k+1} - u^k)/\Delta t = Au^{k+1} + b$: Euler implicite

- Euler explicite :

$$u^{k+1} = (I + \Delta t A)u^k + \Delta t b$$

$$\begin{pmatrix} x \\ v \end{pmatrix}^{k+1} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix}^k + \Delta t \begin{pmatrix} 0 \\ g \end{pmatrix}$$

Même résultats que précédemment

$$\Rightarrow \begin{cases} x^{k+2} = 2x^{k+1} - x^k + (\Delta t)^2 g \\ x^0 = x_0 \text{ et } x^1 = x_0 + \Delta t v_0 \end{cases}$$

$$\Rightarrow x(k\Delta t) = x_0 + (k\Delta t) v_0 + \frac{k(k-1)}{2} (\Delta t)^2 g$$

- Euler implicite :

$$u^{k+1} = (I - \Delta t A)^{-1}(u^k + \Delta t b)$$

$$\begin{pmatrix} x \\ v \end{pmatrix}^{k+1} = \begin{pmatrix} 1 & -\Delta t \\ 0 & 1 \end{pmatrix}^{-1} \left[\begin{pmatrix} x \\ v \end{pmatrix}^k + \Delta t \begin{pmatrix} 0 \\ g \end{pmatrix} \right]$$

$$\Rightarrow \begin{cases} x^{k+2} = 2x^k - x^k + (\Delta t)^2 g \\ x^0 = x_0 \text{ et } x^1 = x_0 + \Delta t v_0 + (\Delta t)^2 g \end{cases}$$

$$\Rightarrow x(k\Delta t) = x_0 + (k\Delta t) v_0 + \frac{k(k+1)}{2} (\Delta t)^2 g$$

On ne résout toujours pas la bonne équation, et précision d'ordre 1 :

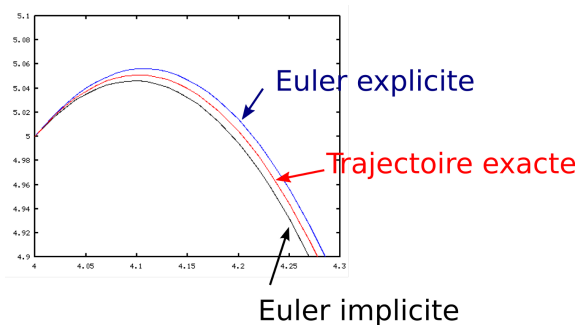
$$\|\tilde{x}(k\Delta t) - x(k\Delta t)\| = \frac{k}{2} (\Delta t)^2.$$

Euler explicite/implicite

- Euler explicite : $u^{k+1} = u^k + \Delta t F^k$
- Euler implicite : $u^{k+1} = u^k + \Delta t F^{k+1}$

Remarque :

- Euler explicite sur-estime $\tilde{x}$.
- Euler implicite sous-estime $\tilde{x}$.



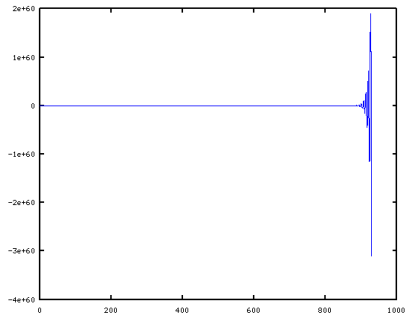
Implicite vs Explicite

Euler explicite

- + Trivial
- Potentiellement divergeant

Euler implicite

- + Inconditionnelement stable
- Inversion d'un système matriciel



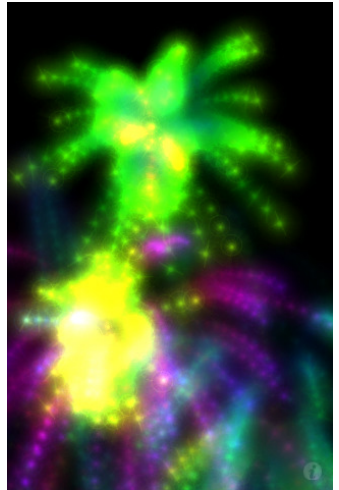
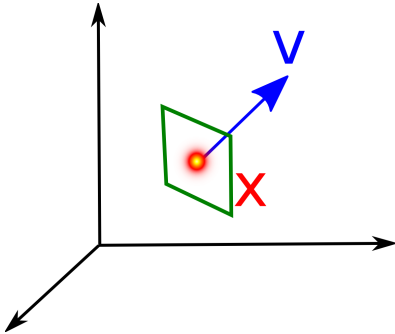
- Méthodes explicites précises : Runge-Kutta pour résoudre $u'(t) = F(u, t)$.
- Ordre 4, estimation d'erreur d'ordre 5 : Dormand-Prince (ode45)

$$\begin{aligned}k_1 &= F(t_n, u_n) \\k_2 &= F\left(t_n + \frac{1}{5}\Delta t, u_n + \frac{1}{5}\Delta t k_1\right) \\k_3 &= F\left(t_n + \frac{3}{10}\Delta t, u_n + \left(\frac{3}{40}k_1 + \frac{9}{40}k_2\right)\Delta t\right) \\k_4 &= F\left(t_n + \frac{3}{5}\Delta t, u_n + \left(\frac{3}{10}k_1 - \frac{9}{10}k_2 + \frac{6}{5}k_3\right)\Delta t\right) \\k_5 &= F\left(t_n + \Delta t, u_n + \left(-\frac{11}{54}k_1 + \frac{5}{2}k_2 - \frac{70}{27}k_3 + \frac{35}{27}k_4\right)\Delta t\right) \\k_6 &= F\left(t_n + \frac{7}{8}\Delta t, u_n + \left(\frac{1631}{55296}k_1 + \frac{175}{512}k_2 - \frac{575}{13824}k_3 + \frac{44275}{110592}k_4 + \frac{253}{4096}k_5\right)\Delta t\right) \\u_{n+1}^4 &= u_n + \Delta t \left(\frac{2825}{27648}k_1 + \frac{18575}{48384}k_3 + \frac{13525}{55296}k_4 + \frac{277}{14336}k_5 + \frac{1}{4}k_6\right) \\u_{n+1}^5 &= u_n + \Delta t \left(\frac{37}{378}k_1 + \frac{250}{621}k_3 + \frac{125}{594}k_4 + \frac{512}{1771}k_6\right)\end{aligned}$$

Application : Sprites

Sprites=

- Particule tombant sous gravité
- Durée de vie limitée
- Plaqué une texture transparente animée

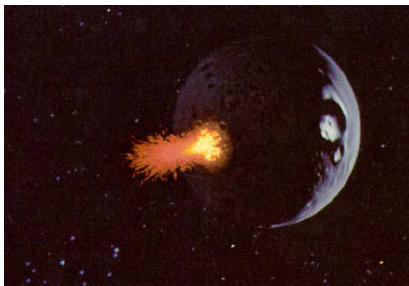


©Apple

Particules + trajectoires

- On peut afficher la trajectoire

[William T. Reeves. **Particle Systems. A Technique for Modeling a Class of Fuzzy Objects.** *ACM Transaction on Graphics*, 17(3). 1983]



© Lucasfilm, Star Trek II



[Reeves, TOG 83]

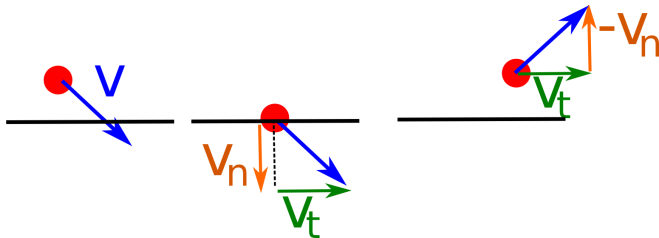
Système de particules + collision

- Les particules peuvent entrer en collision avec un plan :
 $\mathcal{P} : \langle x - p_0, n \rangle = 0$
- Détection : $\langle x - p_0, n \rangle < 0$
- Séparation : vitesse tangentielle v_t , vitesse normale v_n .

$$\begin{cases} v_n = \langle v, n \rangle \\ v_t = v - \langle v, n \rangle n \end{cases}$$

- Après collision, perte d'énergie : amortissement par μ =coeff de restitution

$$v^{k+1} = -\mu \langle v^k, n \rangle n + (v^k - \langle v^k, n \rangle n)$$



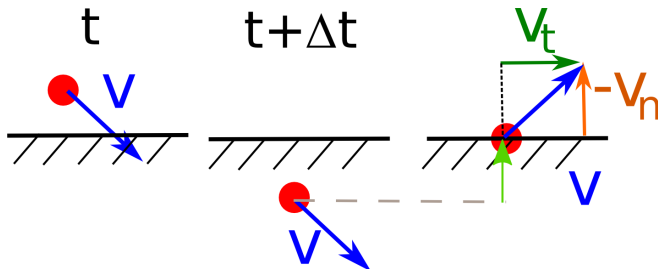
Système de particules + collision

- Attention à la discrétisation temporelle !

1 Projection : Simple, faux (instabilités).

$$x^{k+1} = x^k - \langle x^k - p_0, n \rangle n$$

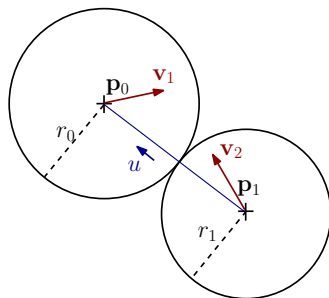
2 Recherche arrière : Moins simple, moins faux.



Sphères dures

- Particules = Sphères masse m , centre x , rayon r .
- Particules en collision si

$$\|x_1 - x_2\| < r_1 + r_2$$



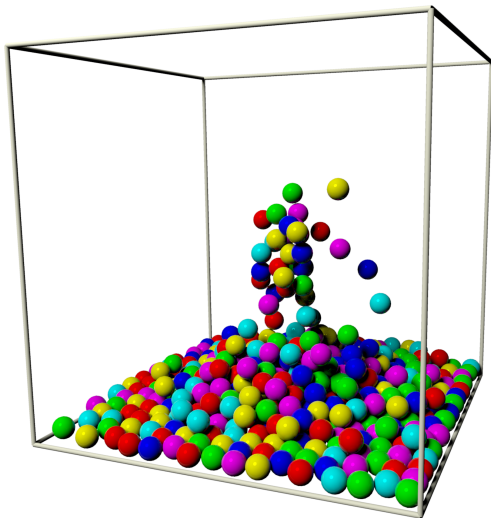
- Nouvelle vitesse en choc élastique

$$\begin{cases} v_1^{k+1} = v_1^k + \frac{1}{m_1+m_2} \left[m_2 \langle v_2^k, u \rangle - \frac{1}{2}(m_1 + 3m_2) \langle v_1^k, u \rangle \right] u \\ v_2^{k+1} = v_2^k + \frac{1}{m_1+m_2} \left[m_1 \langle v_1^k, u \rangle - \frac{1}{2}(m_2 + 3m_1) \langle v_2^k, u \rangle \right] u \\ u = x_1 - x_0 \end{cases}$$

- Penser à reprojeter sur la surface de contact

Sphères dures

- On génère beaucoup de sphères

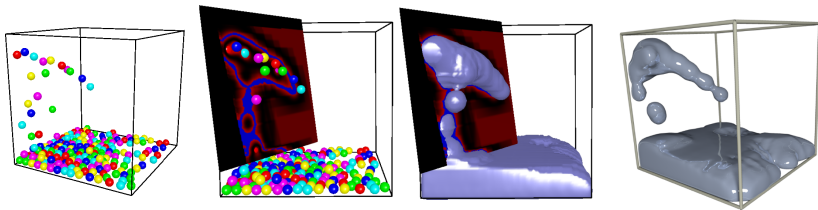
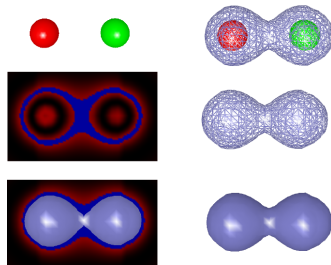


Sphère : Potentiel implicite

- Affecte potentiel de l'espace à chaque particule (ex. blobs)

$$f(x) = \sum_i \exp(-a\|x - x_i\|^n)$$

⇒ Simulation de fluides.



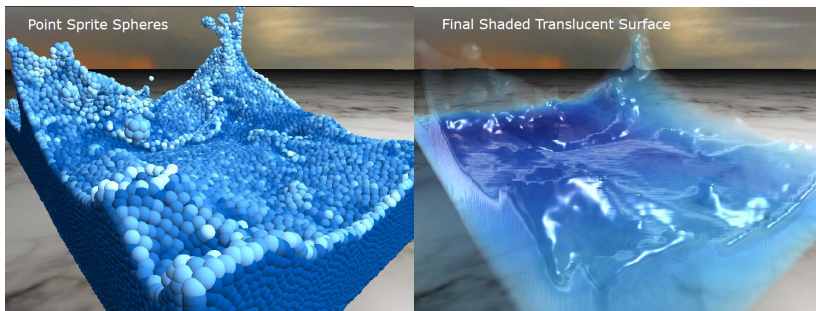
Detection efficace de collision

- Algorithme force brute :

```
for (i=0; i<N; ++i)
  for (j=i+1; j<N; ++j)
    if ( norm(xi-xj)<r1+r2 )
      CollisionResponse(i, j)
```

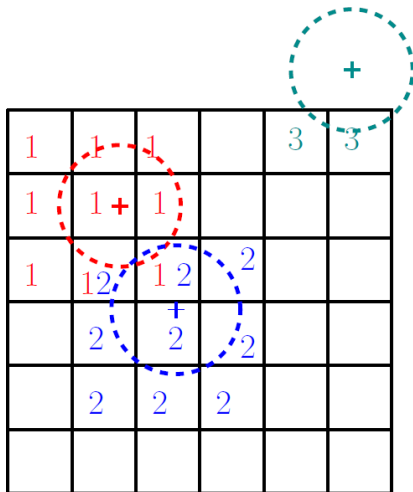
- Complexité $\mathcal{O}(N^2)$

- Impossible pour 10^{3-6} particules en temps réel.



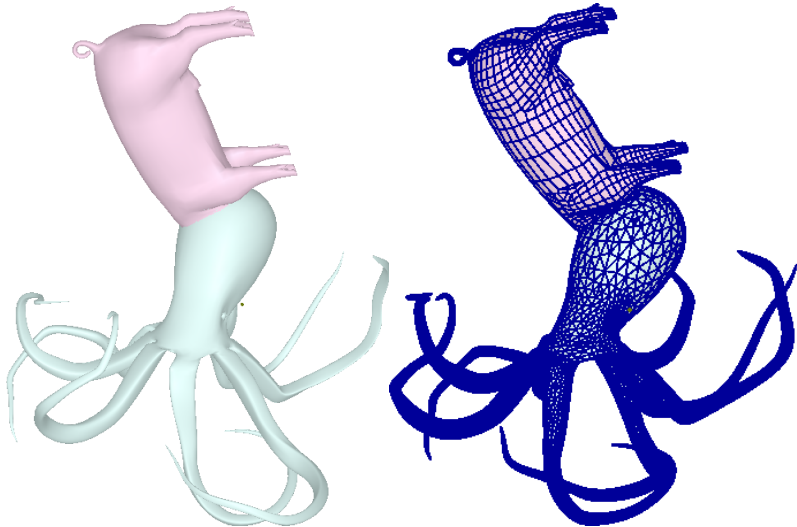
Grilles acceleratrices

- Grille régulière
- Recherche en $\mathcal{O}(1)$
- + Simple, recherche spatial efficace
- + Ok pour échantillonnage uniforme
- Utilisation mémoire



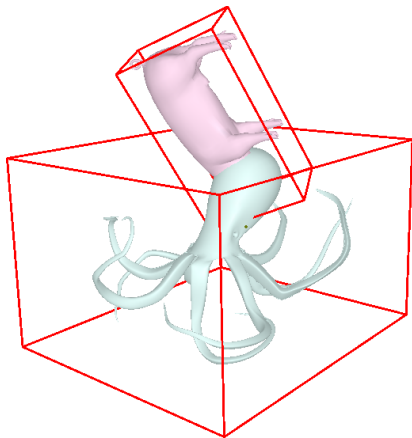
Détection de collision objets

- Recherche triangles/triangles en $\mathcal{O}(N_{T1} N_{T2})$.
- Zones vides.

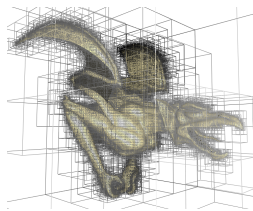


Bounding Box

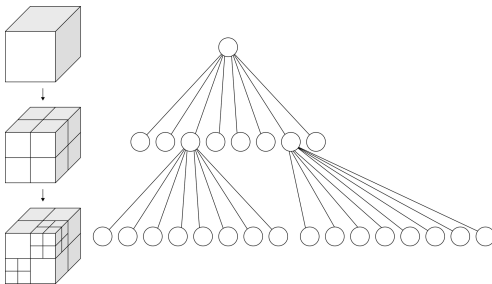
- Boîte englobante = Bounding Box (BB)
- Le plus simple : AABB (Axis Aligned Bounding Box)
- Sphère englobantes
- + Détection de Non-Collision en $\mathcal{O}(N_{\text{obj}}^2)$



- Grille auto-adaptable : **Octree**
- + Géométrie complexe
- + Recherche en $\mathcal{O}(\log(M))$, M taille arbre.



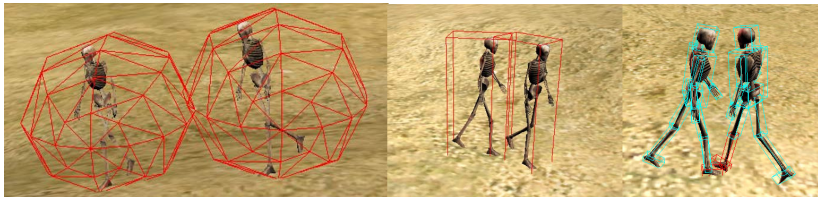
[Lefebvre GPU Gems 2, 2004]



■ En pratique : Niveaux de détails de boites englobantes

ex. Spheres dans BB dans Octree

ex. OBB dans AABB dans Spheres



Ditchburn

1 Test grossier

2 Test fin

3 ...

4 Optionnel : Calcul de la vraie collision

⇒ Tests de Non-collision rapide

⇒ Détection de collision lent

Masse-ressorts

- Ressort 1D : $F(t) = K(L_0 - x(t))$
- Equation : $x''(t) = K/m(L_0 - x(t))$

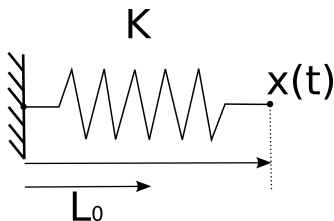
$$u' = \begin{pmatrix} 0 & 1 \\ -K/m & 0 \end{pmatrix} u + \begin{pmatrix} 0 \\ kL_0/m \end{pmatrix}$$

- Vraie solution : oscillations
- Euler explicite :

$$x^{k+2} = 2x^{k+1} - \left(1 + (\Delta t)^2 \frac{K}{m}\right) x^k + (\Delta t)^2 \frac{K}{m} L_0$$

Stable si $(1 + (\Delta t)^2 \frac{K}{m}) < 1 \Rightarrow$ Euler explicite diverge forcément !

- Euler implicite converge vers 0.



- Ajout d'un terme de frottement fluide : $F(t) = -\mu v(t)$.

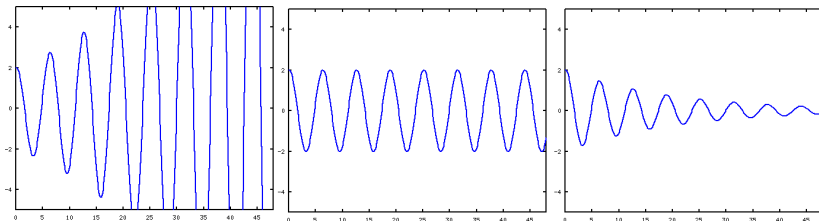
$$u' = \begin{pmatrix} 0 & 1 \\ -K/m & -\mu/m \end{pmatrix} u + \begin{pmatrix} 0 \\ kL_0/m \end{pmatrix}$$

- Nouvelle récurrence pour Euler explicite :

$$x^{k+2} = (2 - \mu\Delta t)x^{k+1} - \left(1 + \frac{(\Delta t)^2 K - \mu}{m}\right) x^k + (\Delta t)^2 \frac{K}{m} L_0$$

- Stable si $\Delta t < \sqrt{\frac{\mu}{K}}$
- Ressorts raides : K grand $\Rightarrow$ Système raide (stiff).

! Précision $\neq$ Stabilité



- Schémas explicite $\Rightarrow$ petits pas de temps / rapide.
- Schémas implicites $\Rightarrow$ grands pas de temps / inversion système.

■ Calcul automatique du pas de temps Δt :

1 Calcul $u_1^{n+1} = F(\Delta t, u^n)$

2 Calcul $u_2^{n+1/2} = F(\frac{\Delta t}{2}, u^n)$;
 $u_2^{n+1} = F(\frac{\Delta t}{2}, u^{n+1/2})$

3 $e = \|u_1^{n+1} - u_2^{n+1}\|$

⇒ $e > \text{threshold}_1 : (\Delta t)_{\text{new}} = \frac{\Delta t}{2}$

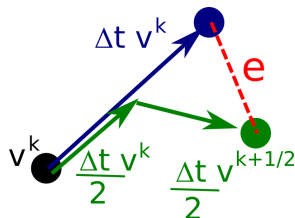
⇒ $e < \text{threshold}_2 : (\Delta t)_{\text{new}} = 2\Delta t$

■ 3 évaluations

■ Avec des ordres différents (Dormant-Prince !)

■ $e = \|u_{\text{order}h} - u_{\text{order}h+1}\|$

■ 2 évaluations.



Faire

$$u1 = F(dt, u)$$

$$u2 = F(dt/2, u);$$

$$u2 = F(dt/2, u2);$$

$$e = \text{norm}(u1 - u2)$$

$$\text{si } e < E_{\min}$$

$$dt *= 2;$$

$$\text{si } e > E_{\max}$$

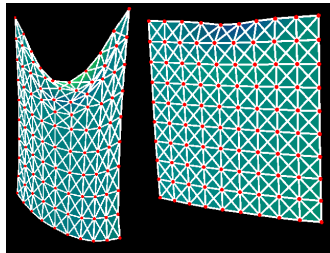
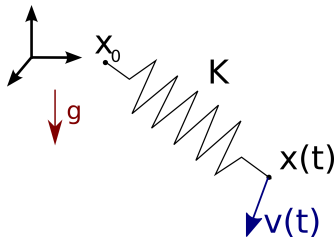
$$dt /= 2;$$

tant que

$$!(E_{\min} < e < E_{\max})$$

- En 3D :

$$F(t) = K (L_0 - \|x - x_0\|) \frac{x - x_0}{\|x - x_0\|}$$



M. Fisher

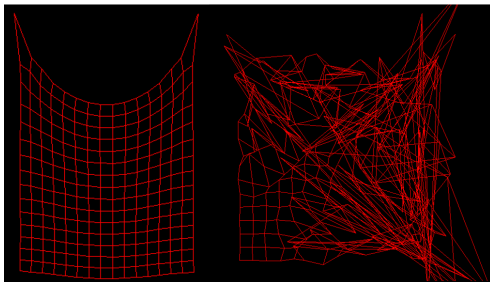
Masses-ressorts : Animation de tissus

- Modélisation d'un tissu par des ressorts couplés.
- Force pour un sommet i de voisins $\mathcal{V}_i$

$$F(x_i, t) = \sum_{j \in \mathcal{V}_i} K^{ij} \left(L_0^{ij} - \|x_i - x_j\| \right) \frac{x_i - x_j}{\|x_i - x_j\|} + g$$

$$\forall i, \begin{cases} x'_i(t) = v_i(t) \\ v'_i(t) = \frac{1}{m_i} \sum_j K^{ij} \left(L_0^{ij} - \|x_i - x_j\| \right) \frac{x_i - x_j}{\|x_i - x_j\|} + g \end{cases}$$

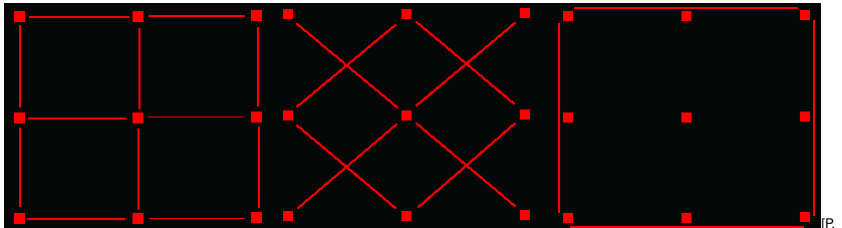
⇒ Schéma d'intégration favori



[P. Jacobs]

Masses-ressorts, animation de tissu

- Ressorts d'elongation (stiffness) : K_1
- Ressorts de cisaillement (shear) K_2
- Ressorts de courbure (bending) K_3



Jacobs]

[P.

Algorithme Euler explicite

```
//calcul des forces
Pour tout i
  Pour j: 4 voisins directes (structure) K=K1
          4 voisins diagonaux (shear) K=K2
          8 voisins (bend): K=K3
    u=x[i]-x[j]
    F[i] += K (L0-norm(u))*u/norm(u)
  fin Pour
fin Pour

//MAJ
Pour tout i
  v[i] += dt*F[i]
  x[i] += dt*v[i]
fin Pour
```

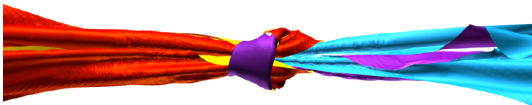
- Forme vectorielle :

$$u(t) = (x_0(t), x_1(t), \dots, x_{N-1}(t), v_0(t), \dots, v_{N-1}(t))$$

$$u'(t) = \mathcal{F}(u(t))$$

- ⇒ Système non linéaire à N inconnues : On ne sait pas inverser.
- ⇒ Linéarisation en $x(t) - x_0 = \Delta L v(t) + \mathcal{O}((\Delta L)^2)$
 - On se ramène à un système linéaire.
 - Perte de la stabilité inconditionnelle : En pratique très stable.

- Vêtement = Tissu + collisions complexes



[Grinspun, SIGGRAPH 09]



© DAZ3D, Dynamic Clothing

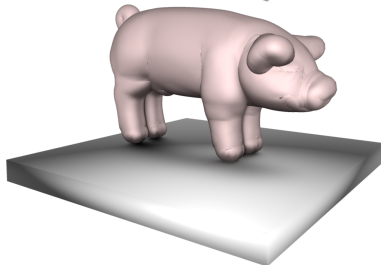
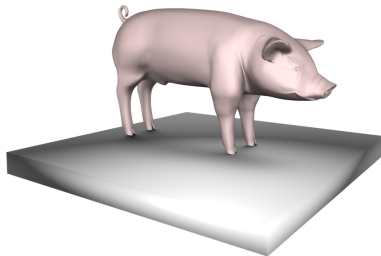
ETI5 Majeure Image: Déformations de surfaces

CPE Lyon
damien.rohmer@cpe.fr

01 Decembre 2010

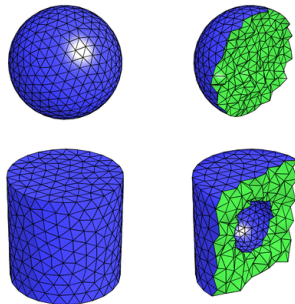
- On cherche à modifier les coordonnées
- Pas la connectivité
⇒ Changement de Topologie
= Complexe

```
OFF
40 95 75
-0.175114 -0.047799 -0.046492
-0.199566 0.738914 -0.064795
-0.010689 0.674496 0.008908
-0.015538 0.153071 0.107408
-0.070148 0.767894 -0.116107
-0.053836 -0.782815 0.109714
-0.162416 -0.785481 0.088014
-0.112365 -0.782492 0.135482
-0.240928 0.031451 0.031966
-0.259289 0.208557 0.035420
0.296891 -0.707385 0.143375
-0.190129 -0.069002 0.109358
-0.010148 0.024179 -0.067283
-0.112968 -0.089127 0.092391
-0.185028 0.377372 -0.111155
3 20 4 1
3 34 11 13
3 12 30 0
3 30 13 17
3 23 22 21
3 29 38 17
3 32 0 13
3 14 0 37
3 24 4 21
3 14 32 1
3 24 2 22
3 3 12 25
3 4 24 15
3 21 15 26
3 35 34 13
3 19 32 13
3 19 13 27
```



Déformations : Methodes physiques ?

- On exprime les equations de la physique sur des éléments volumiques (il faut les construire)= equations différentielles sur des tetraèdres linéaires.
- On applique les conditions initiales (forces, paramètres) sur le maillage volumique
- On résout les equations par méthode numérique = Inversion de larges matrices creuses.
- On attend ...
- On réitère sur les paramètres



Déformations : Methode non physique

- ⊕ On déplace comme on souhaite ce que l'on souhaite.
- ⊕ Orienté Résultat.
- ⊕ On ne traite que ce qui est visualisé.
- ⊖ On introduit des procédés non réalistes physiquement.
⇒ Contraintes

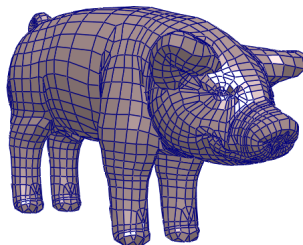
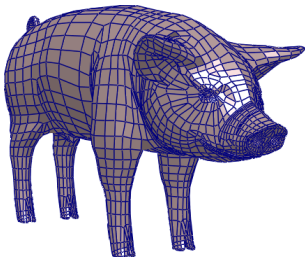


Déformation d'un maillage

- Animation - Déformation de surface
= trouver f tel que

$$(x'_0, x'_1, x'_2, \dots, x'_n) = f(x_0, x_1, x_2, \dots, x_n)$$

On affiche le maillage formé par $(x'_0, \dots, x'_n)$ avec l'ancienne connectivité.



- Comment choisir f ?

Fonction de Deformations

- On maîtrise assez peu de transformations f

$$f : \mathbf{x} \in \mathbb{R}^3 \mapsto \mathbf{x}' \in \mathbb{R}^3$$

- On prend f linéaire : $f = M$.

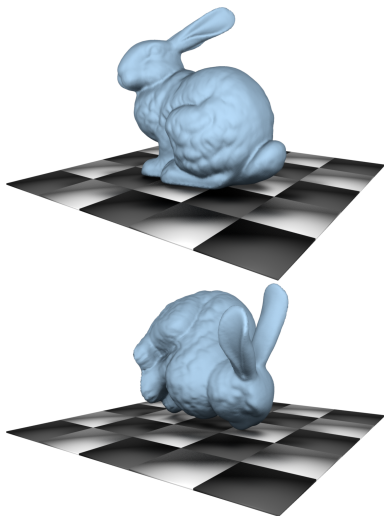
- Isométries

- Translation
- Symétries
- Rotations

$$\mathbf{x}' = M\mathbf{x} \quad \text{et} \quad |\det(M)| = 1$$

- Les homothéties

$$\mathbf{x}' = \text{diag}(s_1, s_2, s_3)\mathbf{x}$$



Important !

Rotation autour d'un axe n par un angle ϕ !

$$R(\mathbf{n}, \phi) = \begin{pmatrix} \cos(\phi) + n_x^2(1 - \cos(\phi)) & n_x n_y(1 - \cos(\phi)) - n_z \sin(\phi) & n_y \sin(\phi) + n_x n_z(1 - \cos(\phi)) \\ n_z \sin(\phi) + n_x n_y(1 - \cos(\phi)) & \cos(\phi) + n_y^2(1 - \cos(\phi)) & -n_x \sin(\phi) + n_y n_z(1 - \cos(\phi)) \\ -n_x \sin(\phi) + n_x n_z(1 - \cos(\phi)) & n_x \sin(\phi) + n_y n_z(1 - \cos(\phi)) & \cos(\phi) + n_z^2(1 - \cos(\phi)) \end{pmatrix}$$

- Rotation Globale

$$\forall i, \mathbf{x}'_i = R \mathbf{x}_i$$

- Revient au même qu'une expression par quaternion.
(interpolations)

Rotation

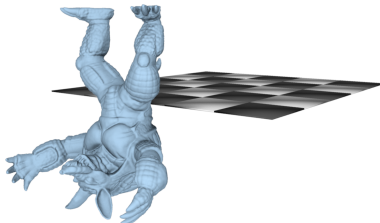
Ne pas oublier de centrer la rotation

$$\mathbf{x}' = R(\mathbf{x} - \mathbf{x}_0) + \mathbf{x}_0$$

Ou sous forme matricielle

$$\mathbf{x}' = T R T^{-1} \mathbf{x}$$

T est une simple translation, ou un changement de base locale !
(Inversion de matrice 3×3 est connu explicitement)



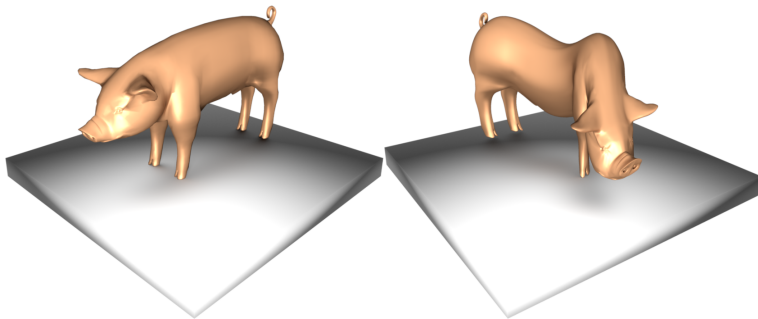
Transformations rigides

Utilisation supplémentaire :

- On peut faire varier les paramètres de la transformation dans l'espace !

$$f : \mathbf{x} \in \mathbb{R}^3 \mapsto \mathbf{x}' = M(\mathbf{x}) \mathbf{x}$$

Ex. Translation dépendant de la position :



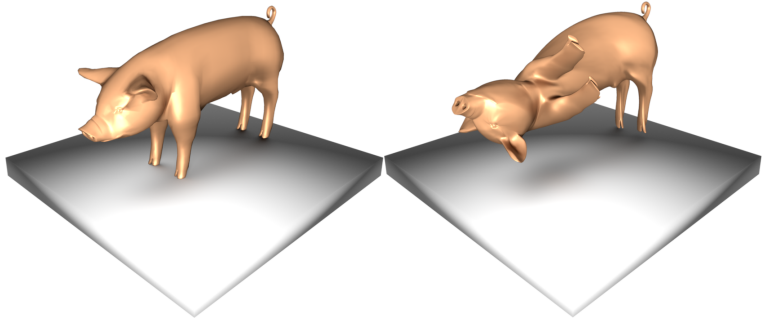
- Angle de rotation qui varie

$$\forall i, \mathbf{x}_i' = R\left(\mathbf{n}, \frac{z}{z_{max}}\right) \mathbf{x}_i$$

```
for (k=0; k<N; k++)  
{  
    Vec x = mesh.get_vertex(k);  
    double angle = x[0] * PI * time;  
    Matrix R = Matrix::rotation(Vec(1,0,0), angle);  
    mesh.set_vertex(k, R*x);  
}
```

Transformations rigides : Exemple

Exemple de Rotation d'angle variable



- Interpolation de matrices pas ok

$$M = \alpha R_1 + (1 - \alpha) R_2 \notin SO(3)$$

- Problème : Interpolation dans espace non Cartésien (Sphère unité = variété).
- ⇒ Généralisation de l'interpolation linéaire dans un **groupe de Lie** :

$$R = \left(R_2 R_1^{-1} \right)^\alpha R_1$$

- Exponentielle de matrices

- Quaternions : Rotation axe n , angle θ .

$$q = (n_x \sin(\theta), n_y \sin(\theta), n_z \sin(\theta), \cos(\theta)) \in \mathbb{R}^4$$

- ⇒ Représentation concise rotation (quaternion unitaires).
- ⇒ Généralisation nombres complexes : $q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$.
- Produit de rotation : $q_1 q_2 \Leftrightarrow R_1 R_2$ (non commutatif)

Algebre

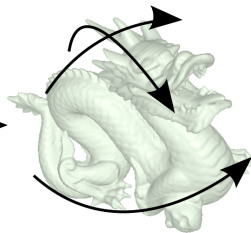
$\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$



Lie
Group
 $SO(3)$



rotations



- Quaternion $q = (q_0, q_1, q_2, q_3) \Rightarrow$ Matrice R :

$$R = \begin{pmatrix} 1 - 2(q_1^2 + q_2^2) & 2(q_0 q_1 - q_3 q_2) & 2(q_0 q_2 + q_3 q_1) \\ 2(q_0 q_1 + q_3 q_2) & 1 - 2(q_0^2 + q_2^2) & 2(q_1 q_2 - q_3 q_0) \\ 2(q_0 q_2 - q_3 q_1) & 2(q_1 q_2 + q_3 q_0) & 1 - 2(q_0^2 + q_1^2) \end{pmatrix}$$

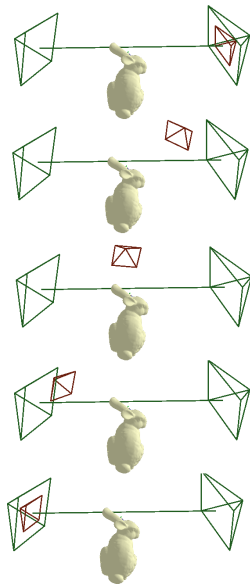
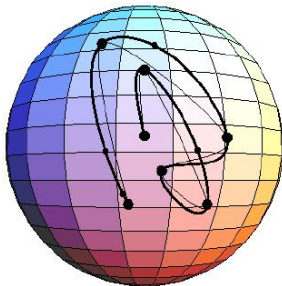
- Représentation concise multiplications : $q = (\mathbf{v}, w)$

$$q_1 q_2 = (w_0 \mathbf{v}_1 + w_1 \mathbf{v}_0 - \mathbf{v}_0 \times \mathbf{v}_1, w_0 w_1 - \mathbf{v}_0 \cdot \mathbf{v}_1)$$

Interpolation de rotations

- Interpolation sans exponentielle de matrices : (SLERP) *Spherical Linear intERPolation*

$$\left\{ \begin{array}{l} SLERP(q_0, q_1, \alpha) = \\ \frac{\sin(\alpha(1-\theta))}{\sin(\theta)} q_0 + \frac{\sin(\alpha\theta)}{\sin(\theta)} q_1 \Leftrightarrow (R_1 R_0)^\alpha R_1 \\ \cos(\theta) = q_0 \cdot q_1 \end{array} \right.$$



- ⊖ Transformations non locales.
- ⊖ Axe et angle variable des rotations : complexe à manipuler
- ⇒ On utilise des transformations constantes
- ⇒ On les rend locales artificiellement morceaux par morceaux

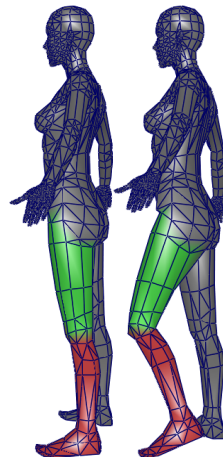
$$\forall i, \mathbf{x}'_i = M(\mathcal{E}(i)) \mathbf{x}$$

Et $M(\mathcal{E}(i)) = I$ pour les ensembles non déformés.

$\mathcal{E}(i)$ = segmentation du maillage.

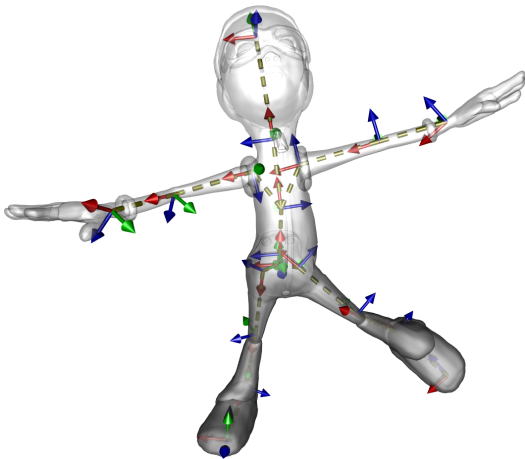
Transformation locales

- On définit des régions $\mathcal{E}$
 - On applique une transformation différente sur chaque partie $M(\mathcal{E})$
 - On utilise principalement des rotations. $R_{\mathcal{E}}$
- ⇒ Comment définir les paramètres de la transformation ?



Squelette d'animation

- Objets sont articulés autour d'un squelette d'animation.
- Squelette = Ensemble de repères hiérarchiques.
- Orientation par rotations successives.



Squelette d'animation

- Transformations hiérarchiques.

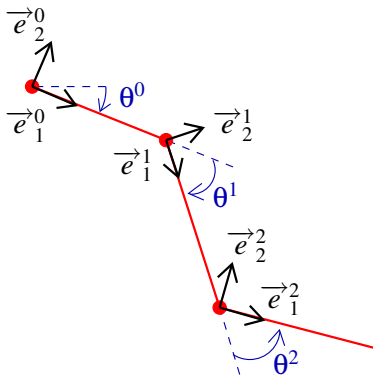
$$T_0 = R_0$$

$$T_1 = T_0 M_1 R_1 M_1^{-1} = R_0 M_1 R_1 M_1^{-1}$$

$$T_2 = T_1 M_2 R_2 T_2^{-1} = \dots$$

⋮

$$T_i = \prod_{k=0}^i M_k R_k M_k^{-1}$$

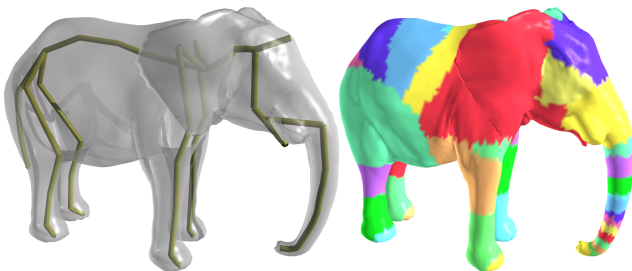


Skinning Rigide

- On lie un morceau local de la surface à un repère du squelette.
- Transformation du repère appliquée sur tous les points du morceau.

$$\mathbf{x}_i(t) = T_{\mathcal{E}(i)}(t) T_{\mathcal{E}(i)}^{-1}(0) \mathbf{x}_i(0)$$

$T(0)$ = Bind Pose.



```
class Frame
{
    Matrix R;
    Matrix Bind;
    Joint *father;
    std::list <Joint*> son;
}
```

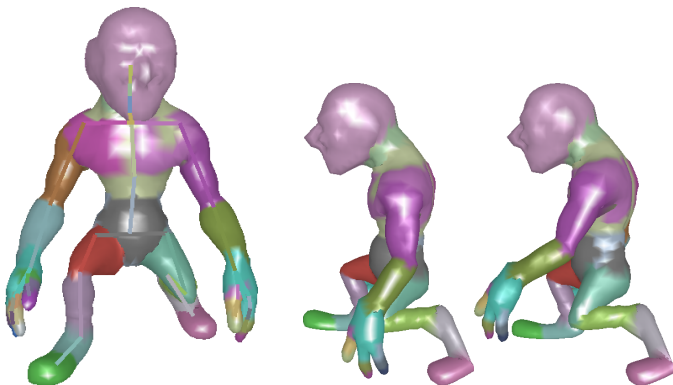
```
for(k_vertex=0;k_vertex<N_vertex;k_vertex++)
{
    int bone_dependency =
        skinning.bone_dependency(k_vertex);
    Matrix T = skeleton.get_matrix(bone_dependency);
    Matrix B =
        (skeleton.get_bind_pose(bone_dependency)).invert();
    deformed_mesh(k_vertex) = T*B * mesh(k_vertex);
}
```

Skinning Rigide : Exemple

■ Action Hierarchique du Squelette

$$\mathbf{x}(t) = \mathbf{M} \mathbf{x}(0)$$

Ex. Une rotation $R_3 = R((1, 0, 0), 45)$



Le squelette est

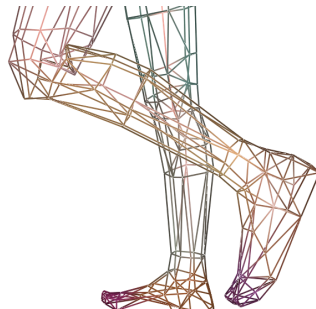
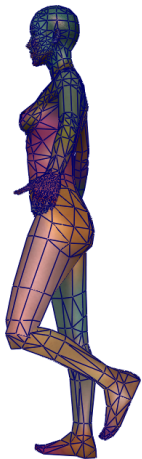
- ⊕ Facile à construire
- ⊕ Facile à animer
- ⊕ Intuitif pour animer un personnage.



Inconvénients Skinning

Mais :

⊖ Discontinuités



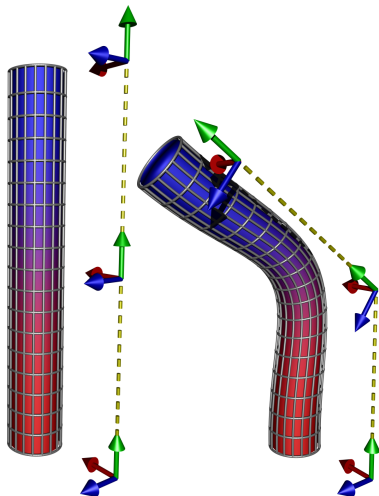
Interpolation des Transformations

Idee : Il faudrait interpoler les transformations entre les repères.

- Le plus simple :

$$T = \omega T_0 + (1 - \omega) T_1$$

- Comment définir les ω ?

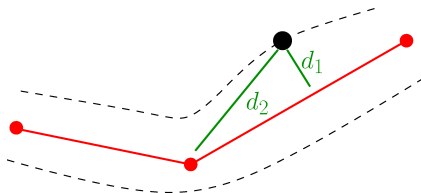
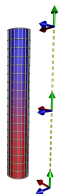


Poids de Skinning

Méthode Possible :

- Fonction de la distance au squelette puis on normalise.

$$\alpha_1 = \frac{1}{d_1}$$
$$\alpha_2 = \frac{1}{d_2}$$
$$\omega_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2}$$
$$\omega_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2}$$



Distance cylindrique

Distance d'un point $\mathbf{x}$ à un segment $[AB]$.

$$\mathbf{u}_1 = \mathbf{x} - \mathbf{x}_A, \quad \mathbf{u}_2 = \mathbf{x}_B - \mathbf{x}_A$$

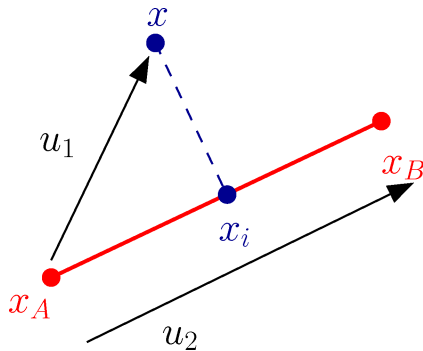
$$p = \frac{\mathbf{u}_1 \cdot \mathbf{u}_2}{\|\mathbf{u}_2\|^2}$$

Si $p < 0$ $\mathbf{x}_i = \mathbf{x}_A$

Si $p > 1$ $\mathbf{x}_i = \mathbf{x}_B$

Sinon $\mathbf{x}_i = \mathbf{x}_A + p\mathbf{u}_2$.

$$d = \|\mathbf{x} - \mathbf{x}_i\|$$

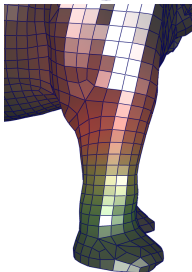
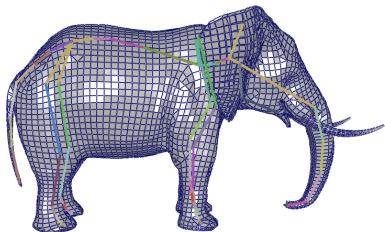


- Pour tout sommet i , on associe les couples (os,poids) :
 $(b_k, \omega_k)_{k \in \mathcal{E}(i)}$.

```
std::vector <std::vector <int>> bone_dependencies;  
std::vector <std::vector <double>> skinning_weights;  
  
for(k_vertex=0;k_vertex<N_vertex;k_vertex++)  
{  
    N_dep = bone_dependencies[k_vertex].size();  
    for(k_dep=0;k_dep<N_dep;k_dep++)  
    {  
        bone = bone_dependencies[k_vertex][k_dep];  
        weight = skinning_weight[k_vertex][k_dep];  
    }  
}
```

```
//skinning weights
for(k_vertex=0;k_vertex<N_vertex;k_vertex++)
{
    for(k_bone=0;k_bone<N_bone;k_bone++)
    {
        alpha = push_back(1/distance(k_vertex,k_bone));
        if(alpha>epsilon)
        {
            skinning_weights[k_vertex].push_back(alpha);
            bone_dependencies[k_vertex].push_back(k_bone);
        }
    }
    norm_skinning_weights();
}
```

Exemples

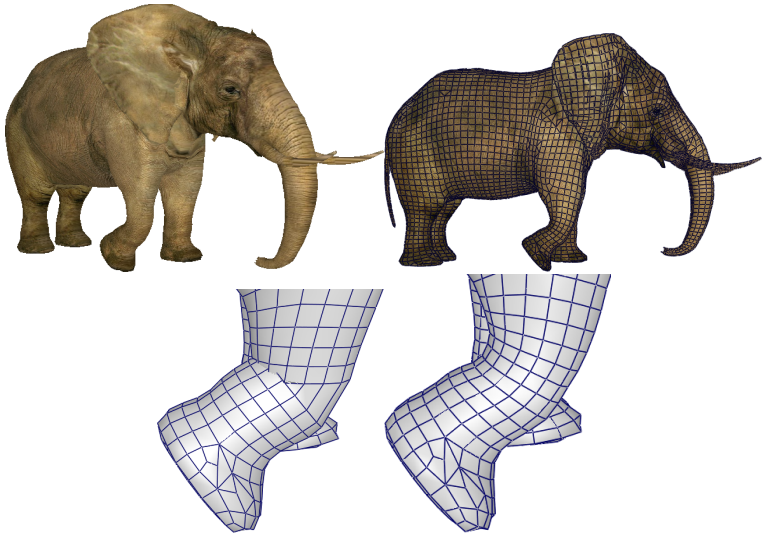


$$\mathbf{x} = \left(\sum_i \omega_i \mathbf{M}_i \right) \mathbf{x}_0$$

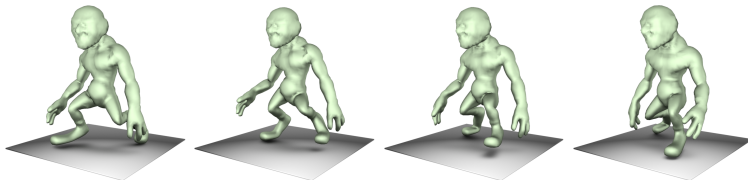


```
for(k_vertex=0;k_vertex<N_vertex;k_vertex++)
{ int N_dep = bone_dependencies[k_vertex].size();
  Matrix D;
  for(k_dep=0;k_dep<N_dep;k_dep++)
  { int k_bone = bone_dependency[k_vertex][k_dep];
    double weight = skinning_weights[k_vertex][k_dep];
    D += weight*skeleton.matrix(k_bone); }
  deformed_mesh(k_vertex) = D * mesh(k_vertex);
}
```

Exemples



- Animation = Les angles des rotations de M dépendent du temps.



```
+ <animation id="ele_R_shoulder_rotateY"></animation>
+ <animation id="ele_R_shoulder_rotateZ"></animation>
+ <animation id="ele_R_elbow_rotateX"></animation>
- <animation id="ele_R_elbow_rotateY">
  - <source id="ele_R_elbow_rotateY_ele_R_elbow_rotateY_ANGLE-input">
    - <float_array id="ele_R_elbow_rotateY_ele_R_elbow_rotateY_ANGLE-input-array" count="134">
      0.040000 0.080000 0.120000 0.160000 0.200000 0.240000 0.280000 0.320000 0.360000 0.400000 0.440000 0.480000 0.520000
      0.560000 0.600000 0.640000 0.680000 0.720000 0.760000 0.800000 0.840000 0.880000 0.920000 0.960000 1.000000 1.040000
      1.080000 1.120000
    </float_array>
  </source>
  - <source id="ele_R_elbow_rotateY_ele_R_elbow_rotateY_ANGLE-output">
    - <float_array id="ele_R_elbow_rotateY_ele_R_elbow_rotateY_ANGLE-output-array" count="134">
      0 -0.072670 -0.223444 -0.389830 -0.549435 -0.692583 -0.812962 -0.905247 -0.964565 -0.985765 -0.985721 -0.985668 -0.985607
      -0.985540 -0.961561 -0.887361 -0.755854 -0.560878 -0.285142 -0.201935 -0.173549 -0.175789 -0.182313 -0.192826 -0.207032
      -0.224636
    </float_array>
```

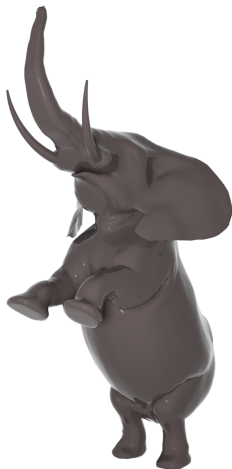
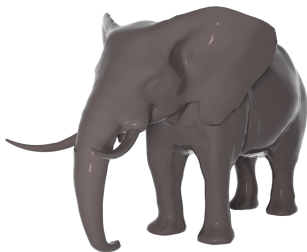
Skinning Lisse : Conclusion

Avantages

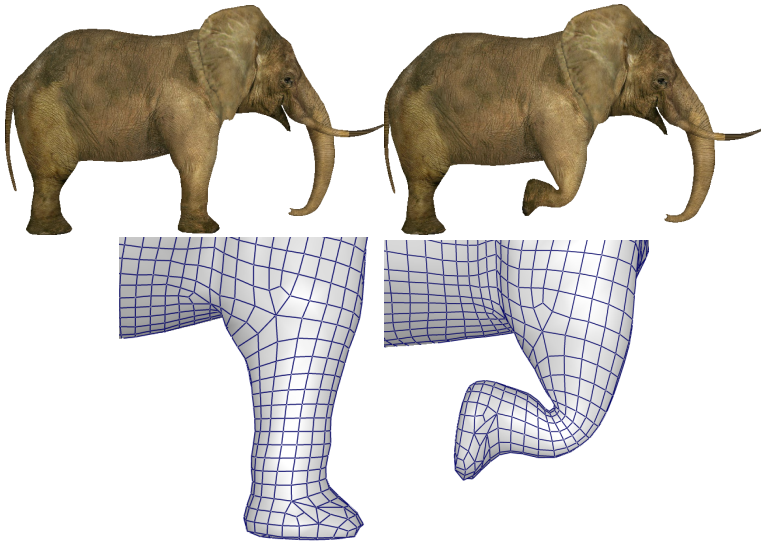
- ⊕ Calcul Rapide.
- ⊕ Squelette intuitif à paramétrer.

inconvénients

- ⊖ Artefacts pour des angles importants.

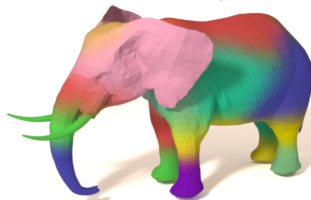
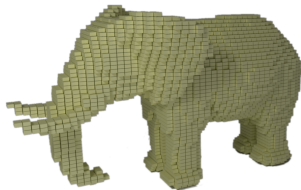
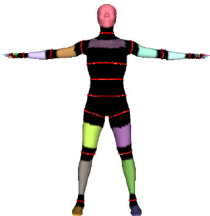


Artefact : Collapsing Elbow



Poids de skinning : Problème de distance cartésienne.

- Méthode 2 : Un artiste peint directement sur le maillage
- Méthode 3 : Plus intelligent (distance curviligne, geodesique, ...)



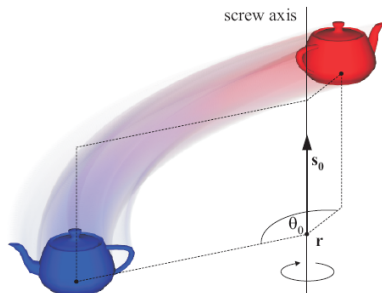
Améliorations Possibles

Interpolation : Interpolation linéaire n'est pas correcte.

$$\mathcal{R}_1 + \mathcal{R}_2 \neq \mathcal{R}.$$

⇒ Comment interpoler une matrice de transformation ?

- **Exponential map** : Problème de temps de calcul.
- **Quaternions** : Ici on a un axe de rotation qui se translate.
- **Dual Quaternions** : OK



Skinning = **S**keleton **S**ubspace **D**eformation

Methode la plus adaptée pour les mouvements articulés autour d'un squelette.

- Déformer un visage = interpolation de formes
- Déformation “molles” = FFD
- Déformations hierarchiques = Multiresolution
- Déformations libres = Inversion de matrices + contraintes

