Animation

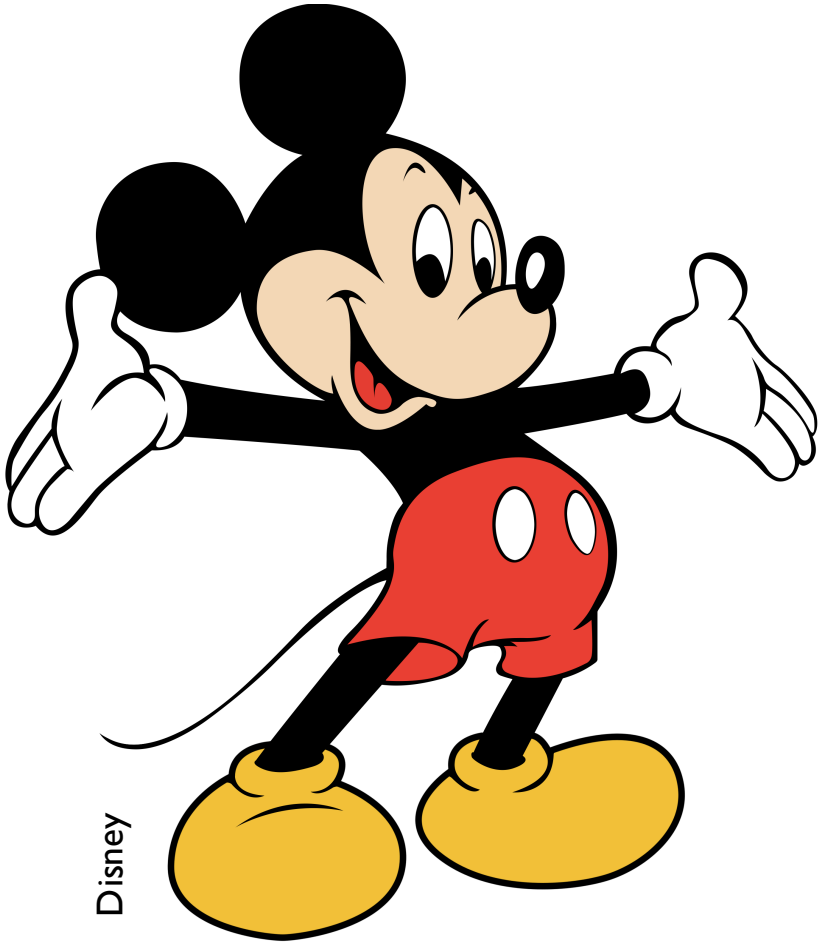
Crédits :
Estelle Duveau, Marie-Paule Cani, Lionel Revert, François Faure,
Ravi Ramamoorthi, Ronen Barzel

Plan

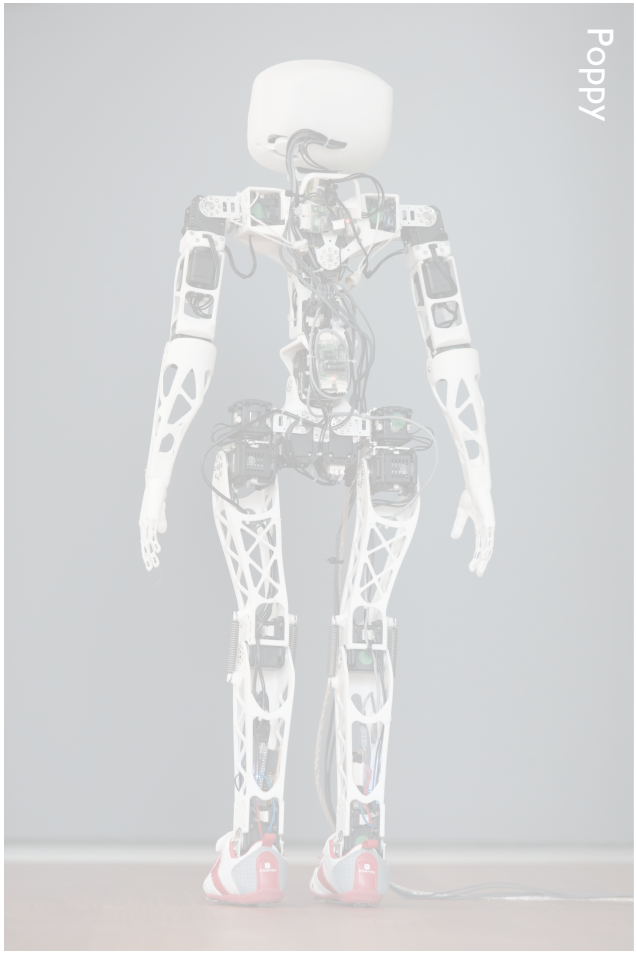
1. **Introduction**
2. Cinématique
3. Déformation de surfaces

Double héritage

Dessin animé

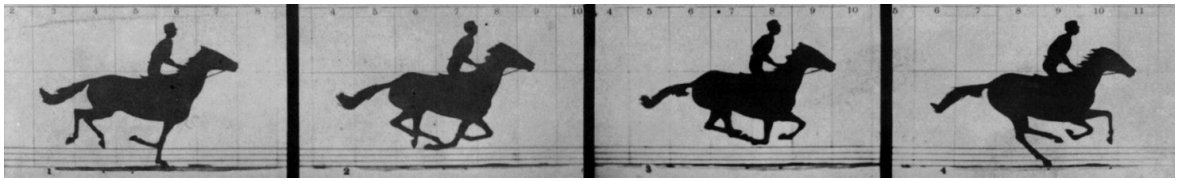


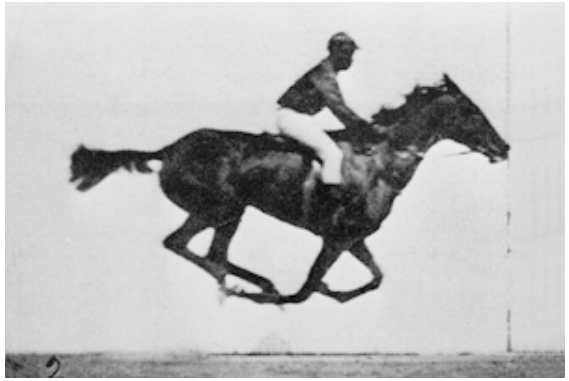
Robotique



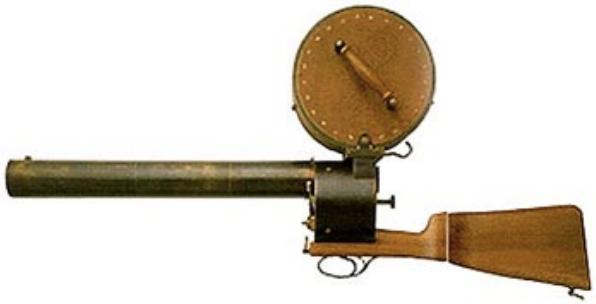
Origines de la cinématique

Eadweard Muybridge (1830-1904)





Etienne-Jules Marey (1830-1904)



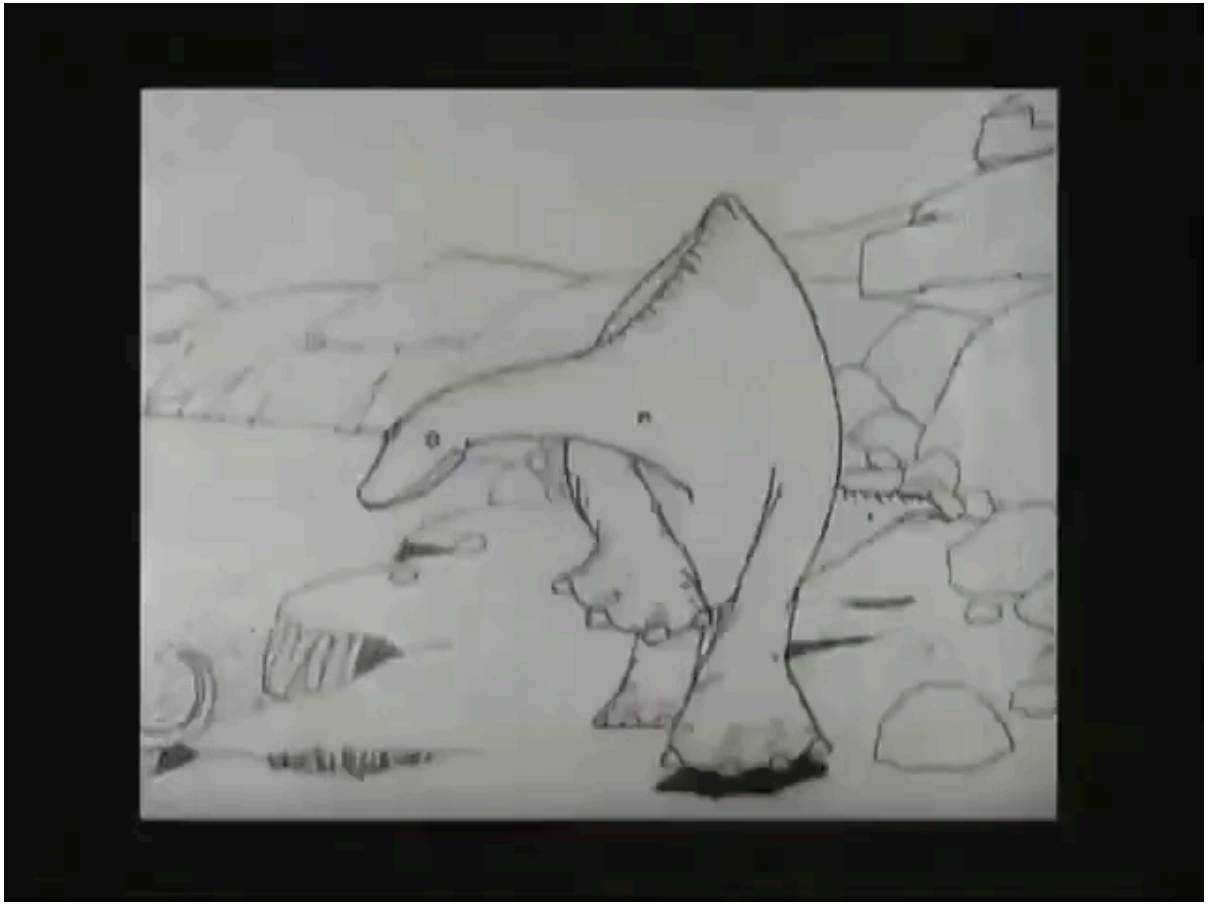
fusil photographique

Phénakistiscope (1831)

Muybridge, 1839

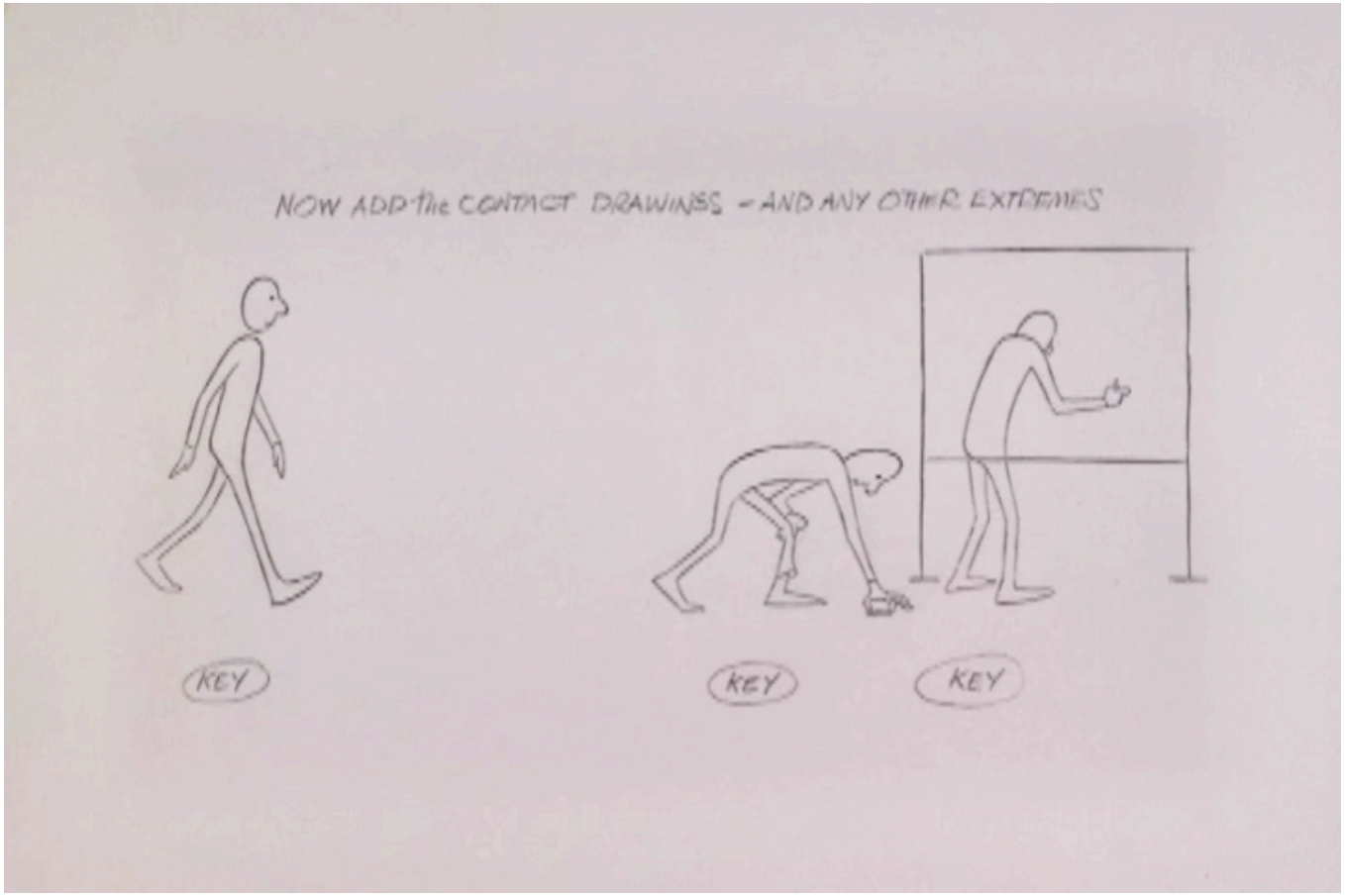
Gertie le dinosaure (1914)

Windsor McCay (1869-1934)



1 er dessin animé par image clés

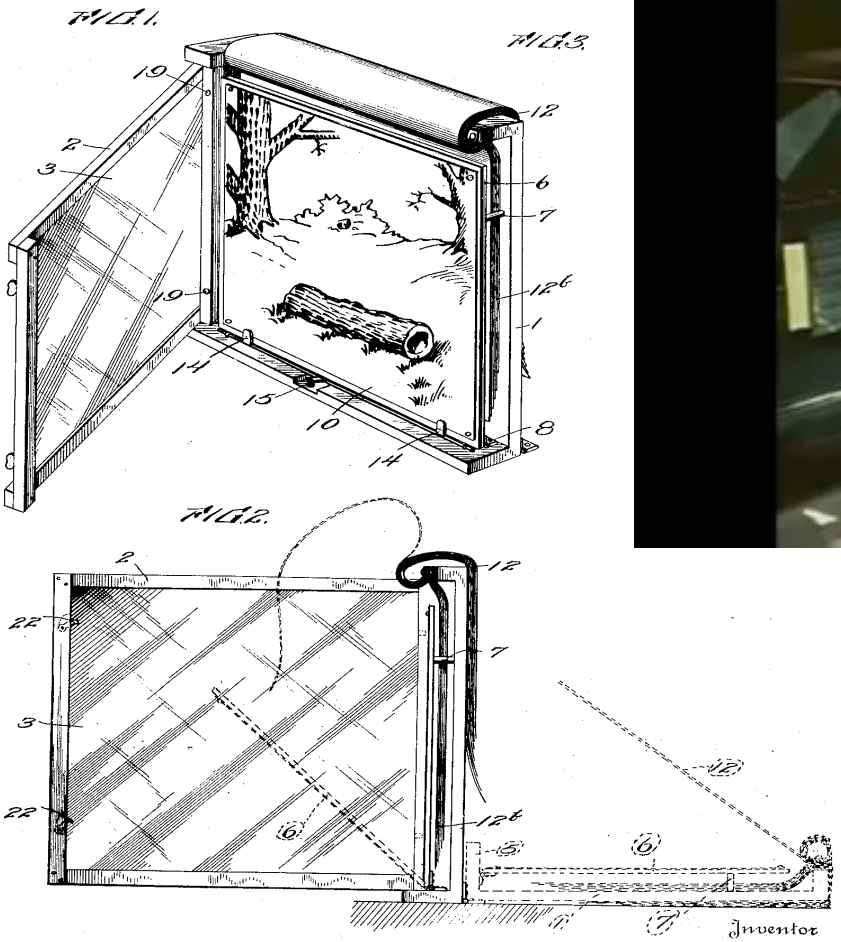
Animation par images clés



Richard Williams, *The Animator's Survival Kit*

Animation sur celluloïd (1915)

J.R. Bray & Earl Hurd
(1879-1978) (1880-1940)

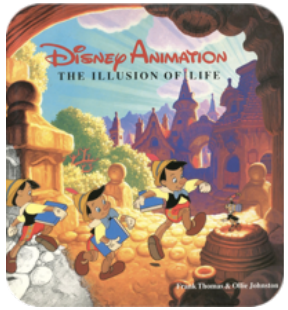


12 principes de Disney

1937 : premier long-métrage d'animation
« *Snow White and the Seven Dwarfs* »

Disney Studios, 1930s : 12 principes d'animation

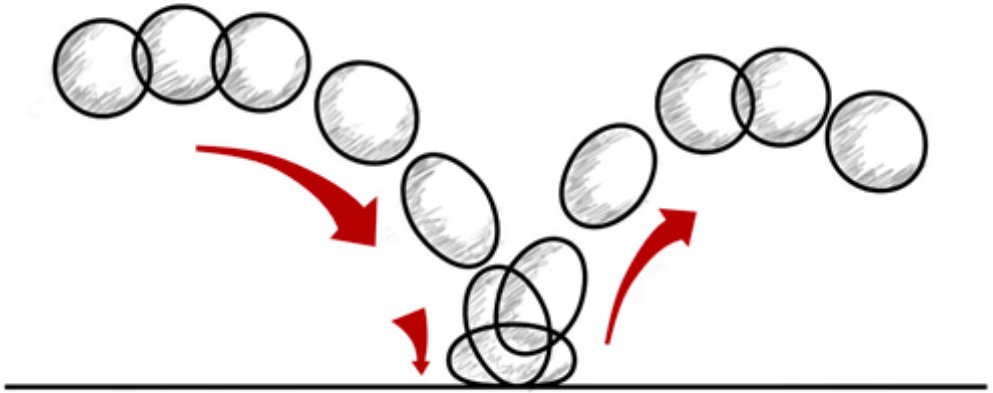
⟹ **toujours valables**

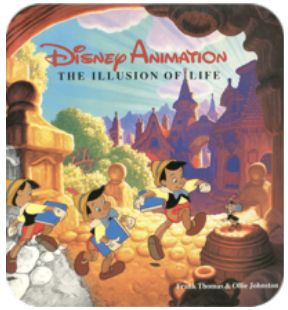


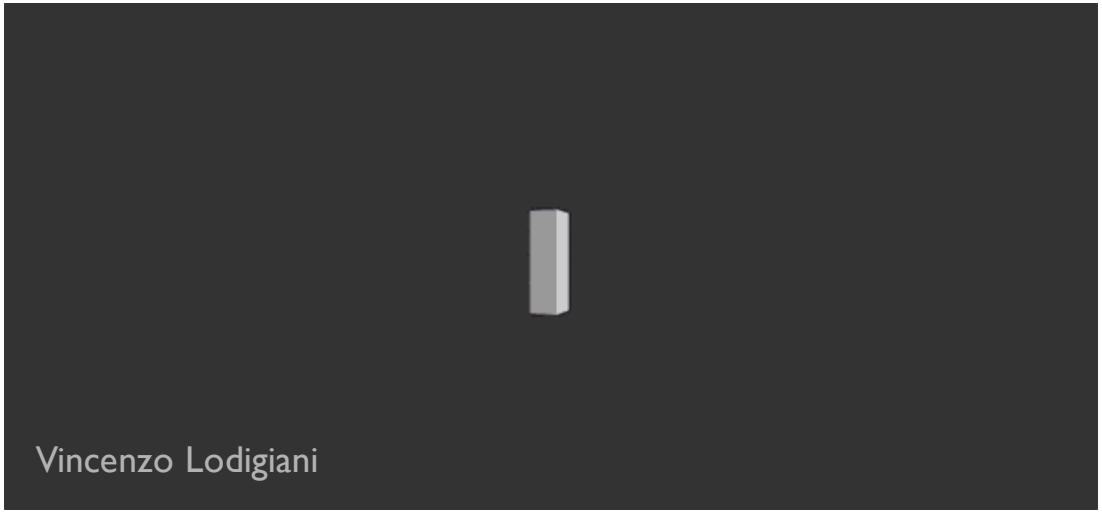
© Disney

1. Squash and stretch

Distorsion de la forme pour accentuer le mouvement
(avec préservation du volume)



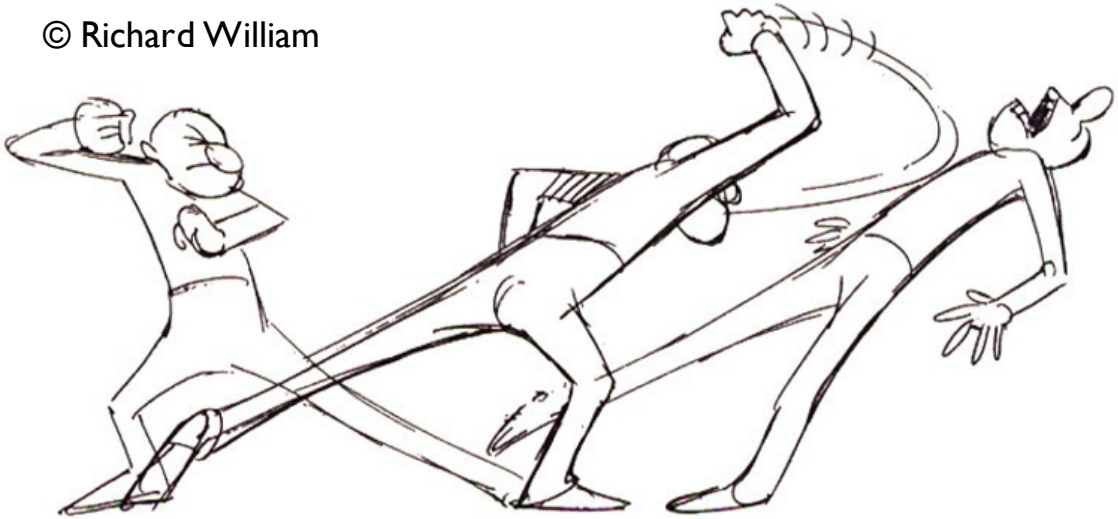


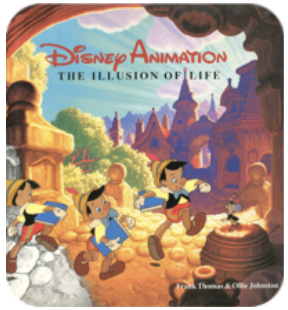


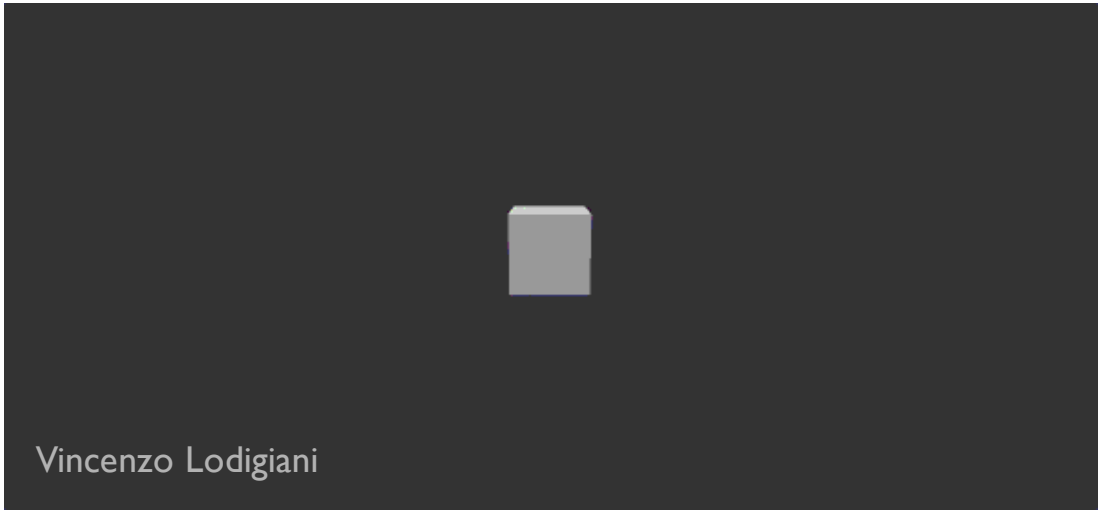
4. Anticipation

Action inverse au mouvement pour l'accentuer

© Richard William



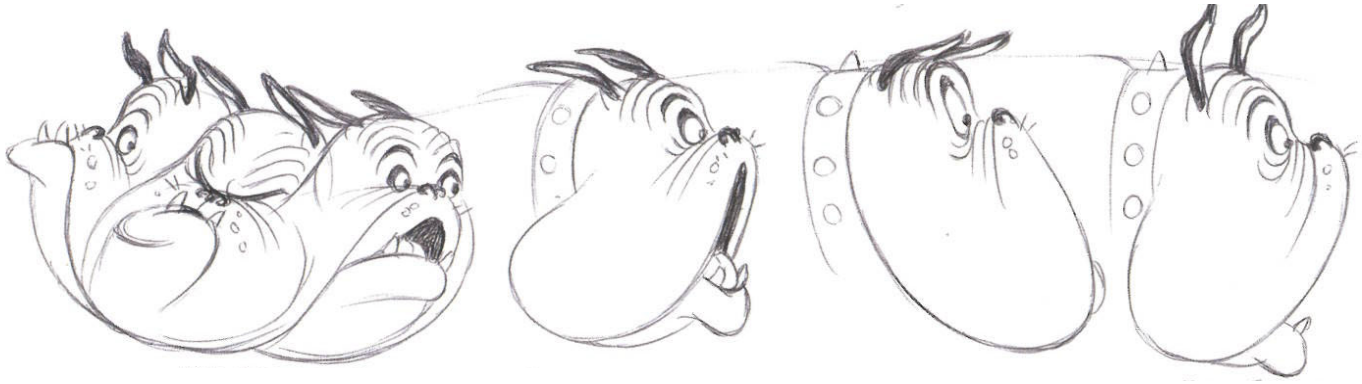


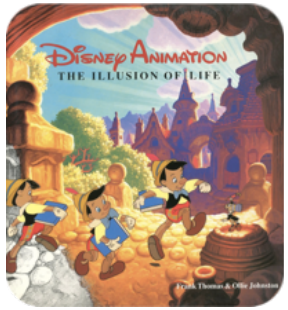


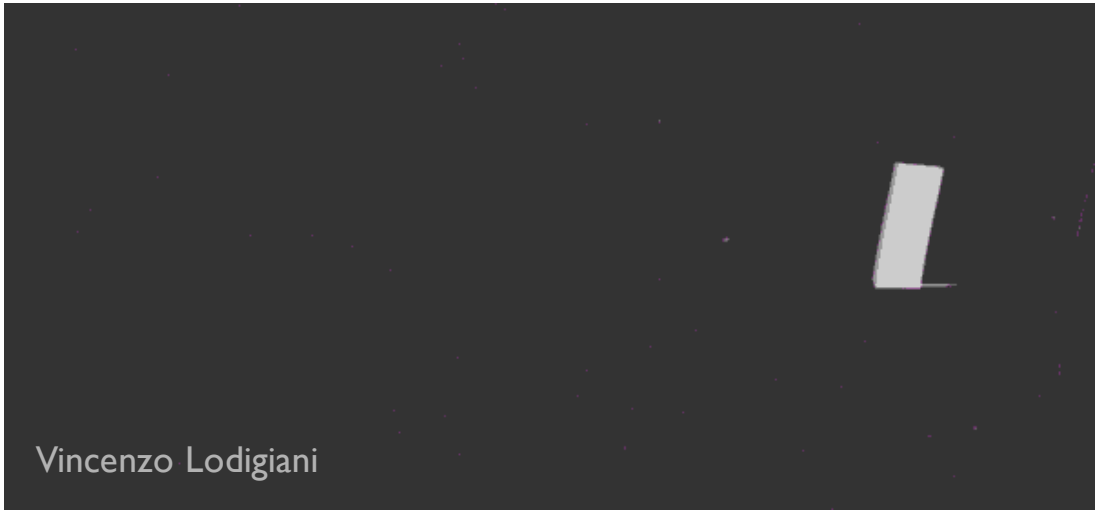
5. Follow through & Overlapping

Retards et anticipations de certaines parties de la forme

© Richard William

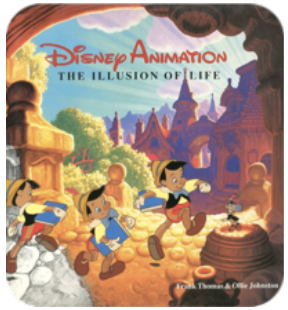


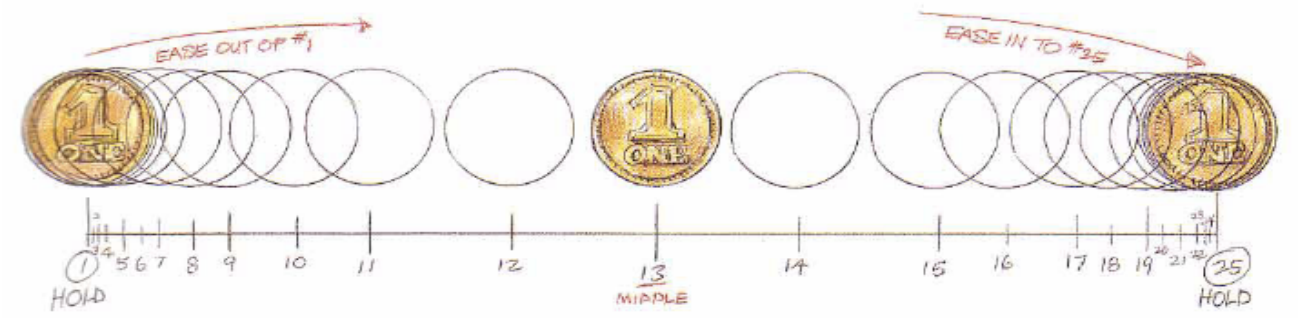




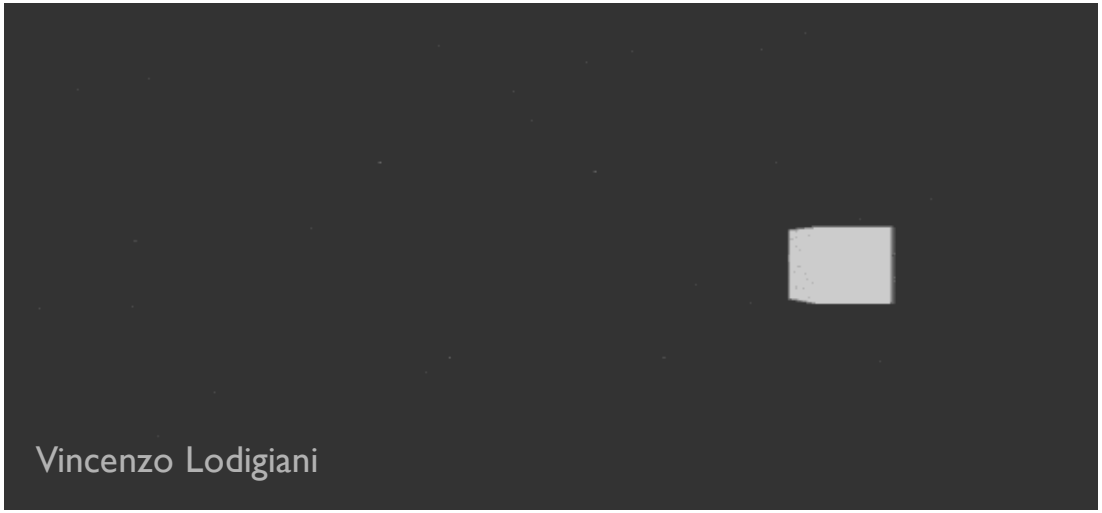
6. Slow in, slow out

Exagération de l'accélération et de la décélération aux extrémités de la trajectoire



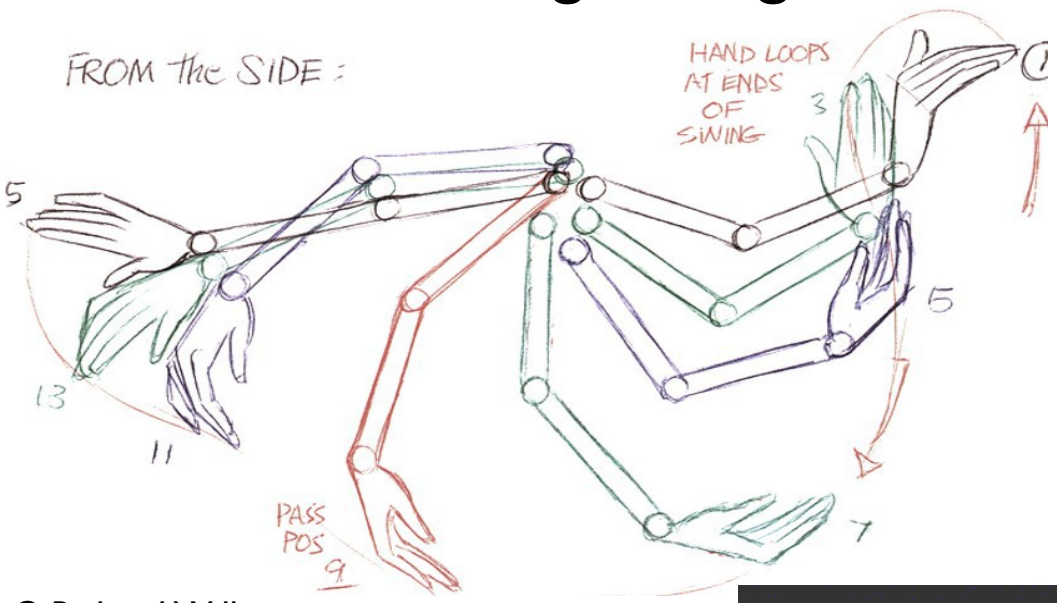


© Richard William

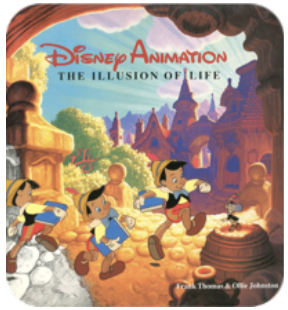


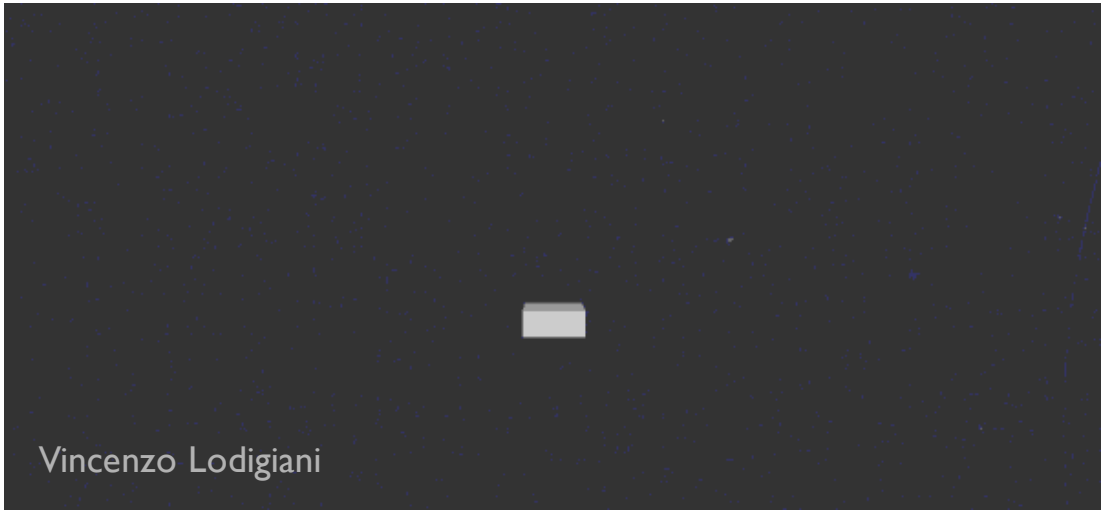
7. Arcs

« Rien » ne bouge en ligne droite mais selon un arc

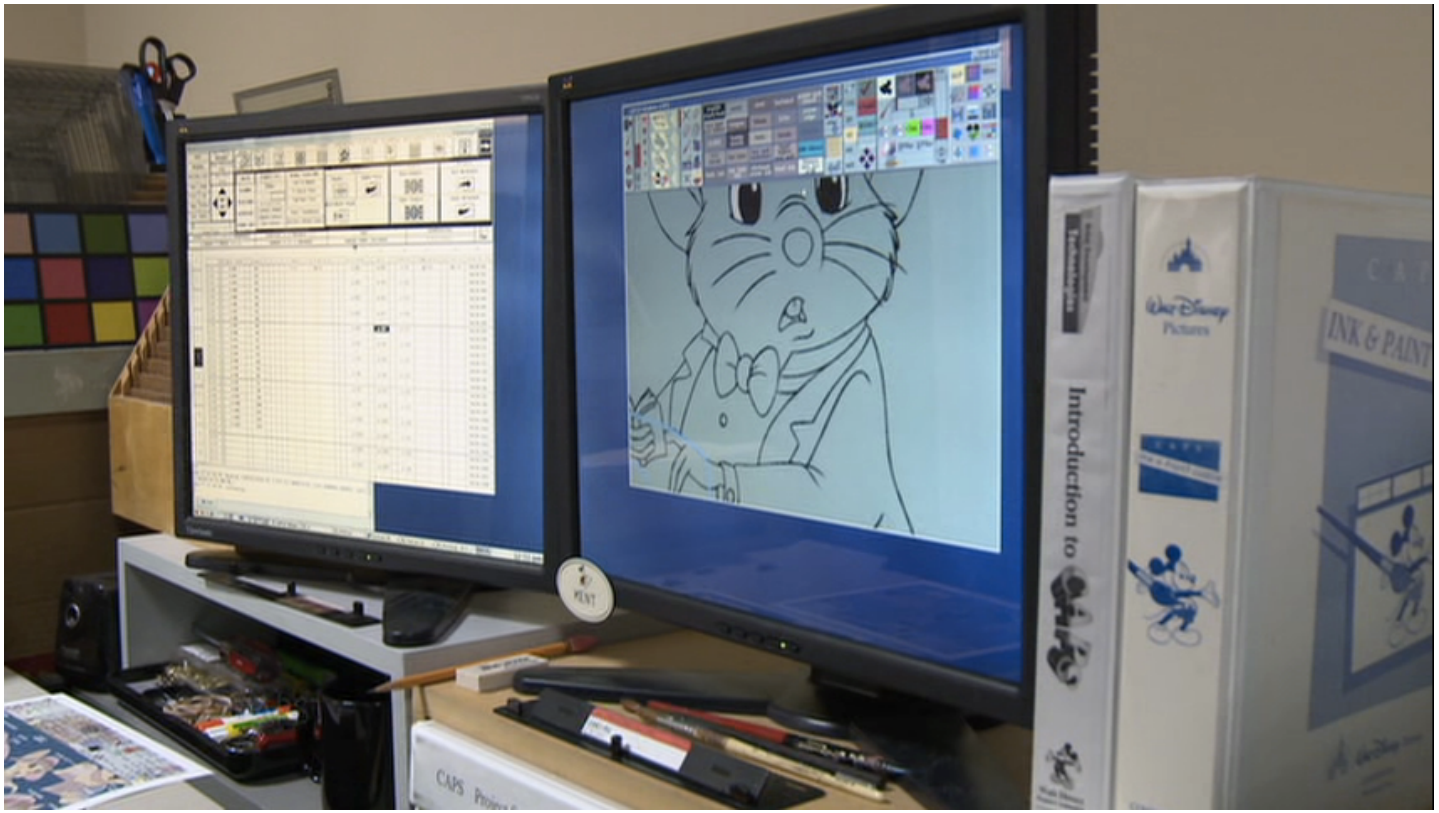


© Richard William





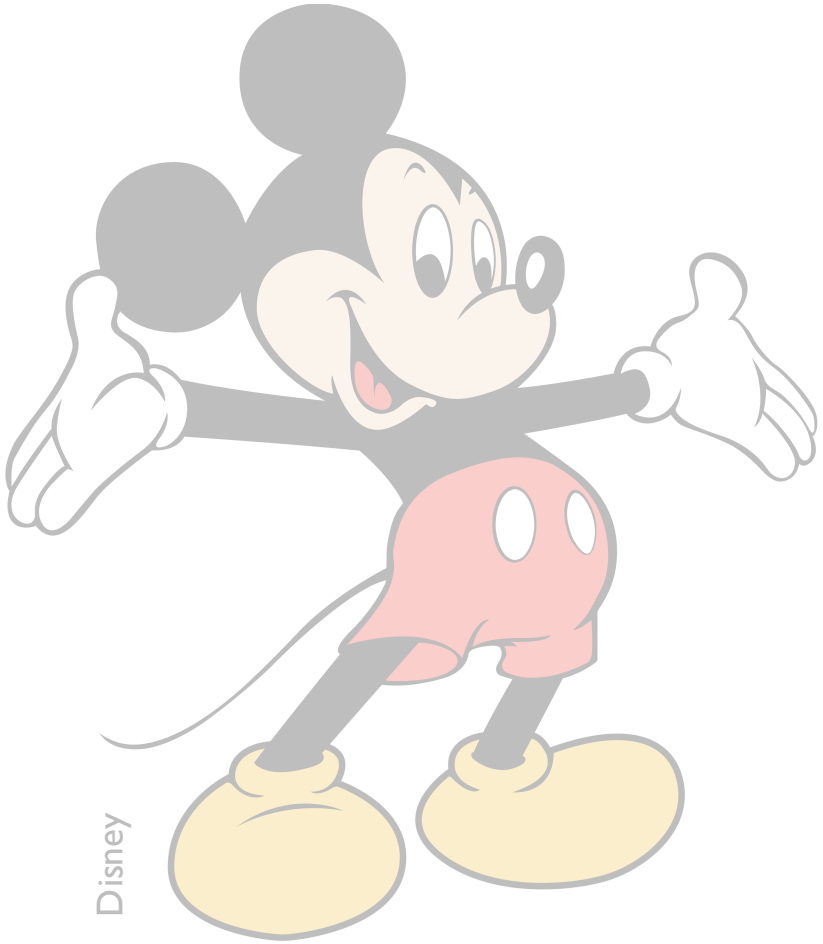
Encrage et remplissage digital



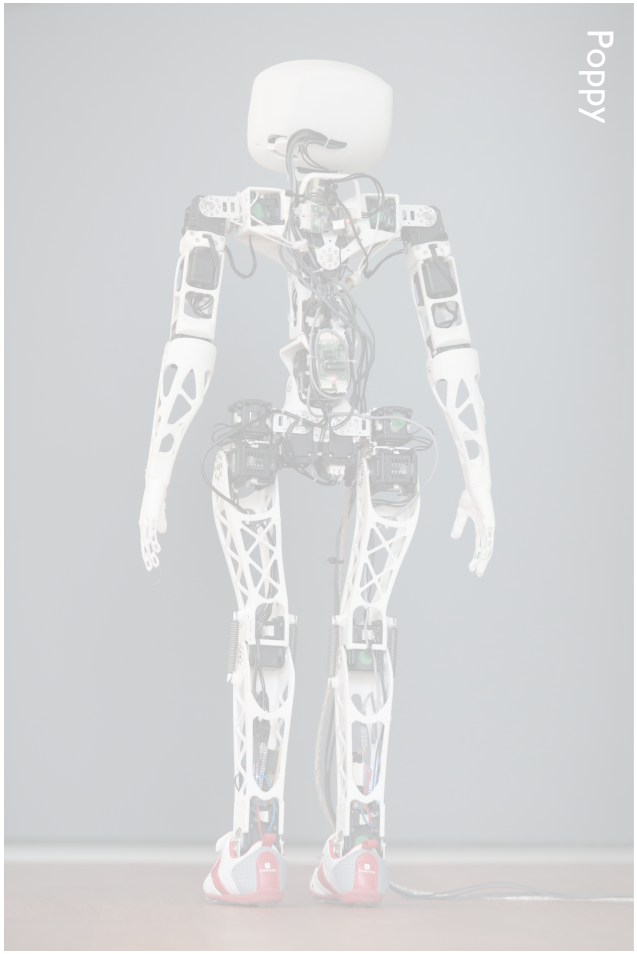
CAPS, LucasFilm - Pixar - Disney
(1989-2006)

Double héritage

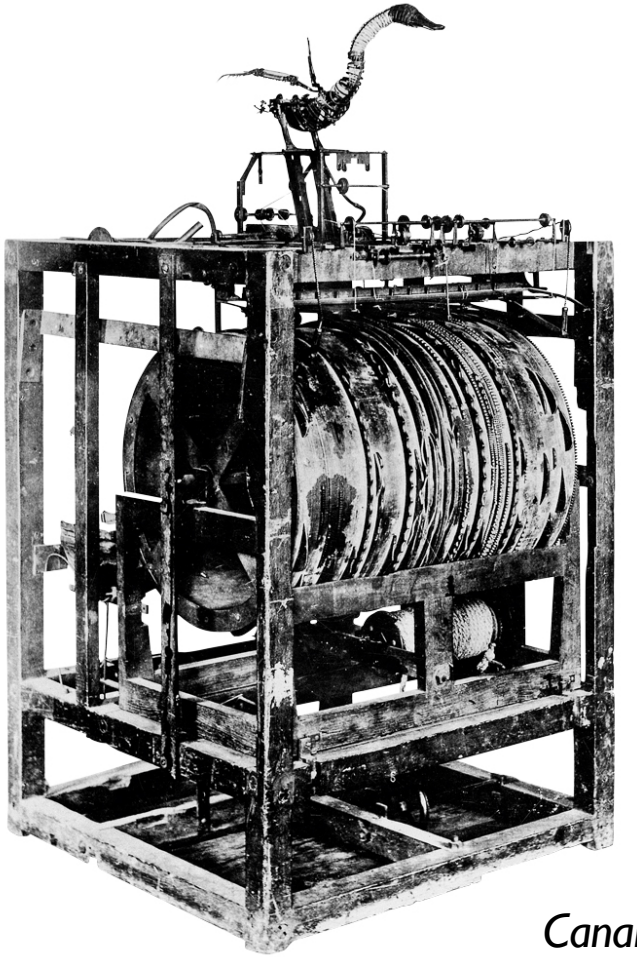
Dessin animé

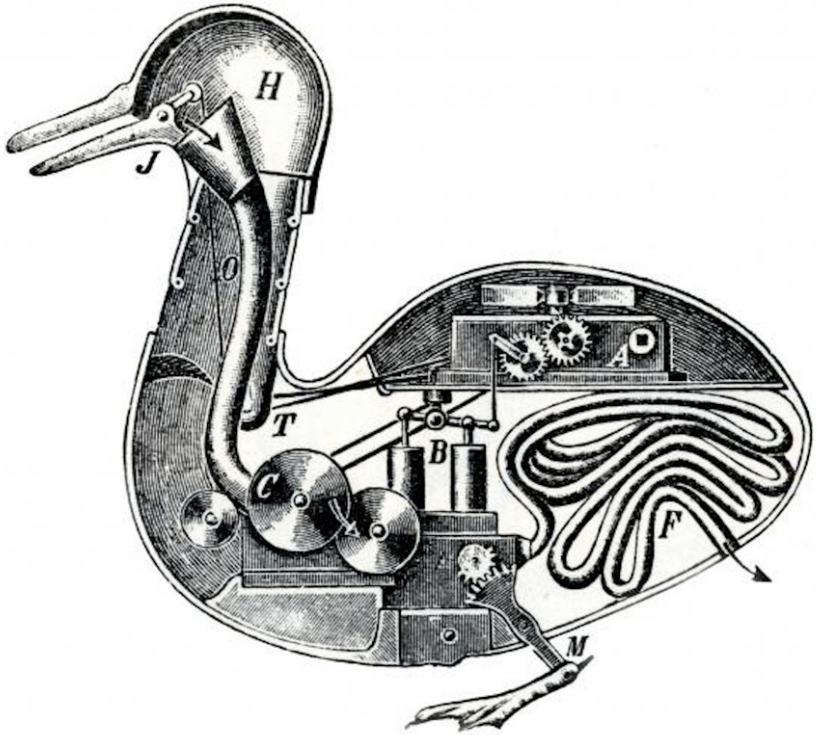


Robotique



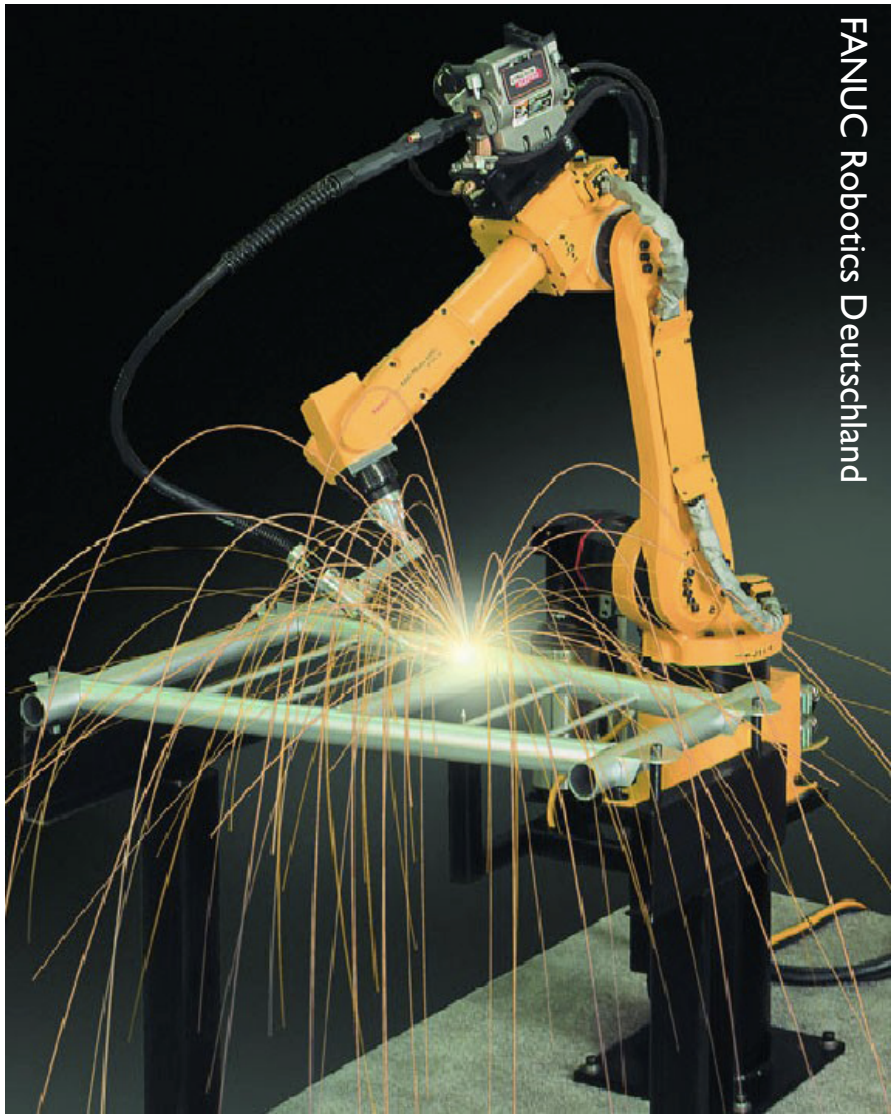
Automates





Canard digérateur (1744)
Jacques de Vaucanson

Robots industriels



Animatronique

Jurassic Park (1993)

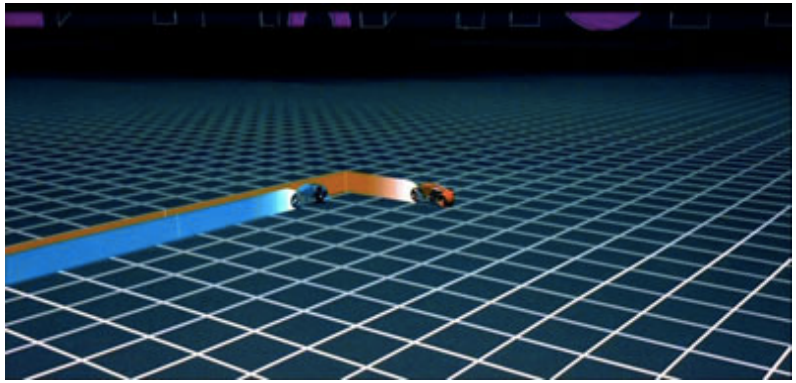
Débuts de l'animation 3D

Recherche académique

A Computer Animated Hand (1972)
Ed Catmull - Fred Parke

Débuts de l'animation 3D

Effets spéciaux



Tron (1986)

Abysse (1989)

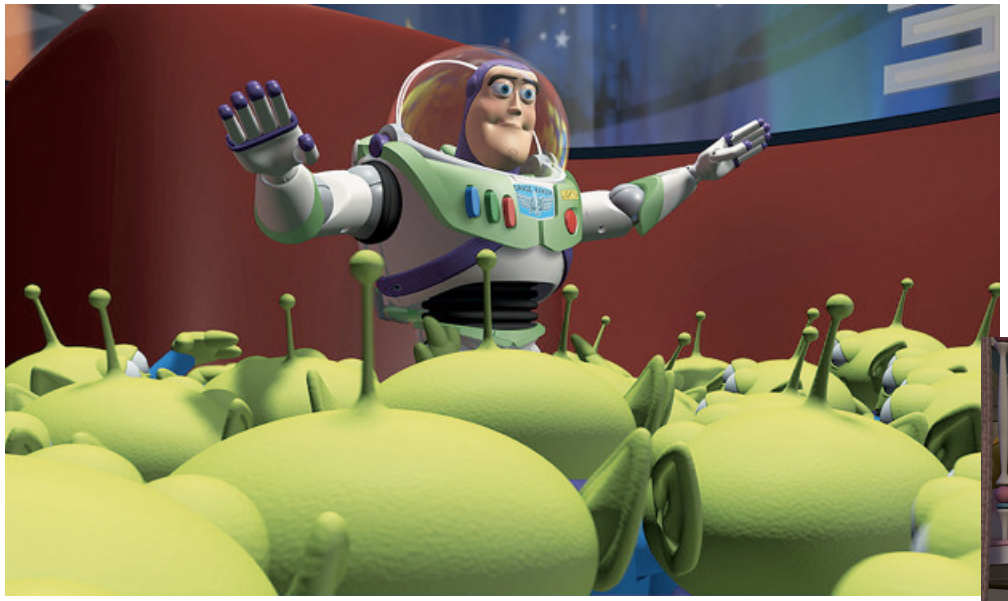
Débuts de l'animation 3D

Court métrage d'animation 3D

Luxo Jr. (Pixar 1986)

Débuts de l'animation 3D

Long métrage d'animation 3D



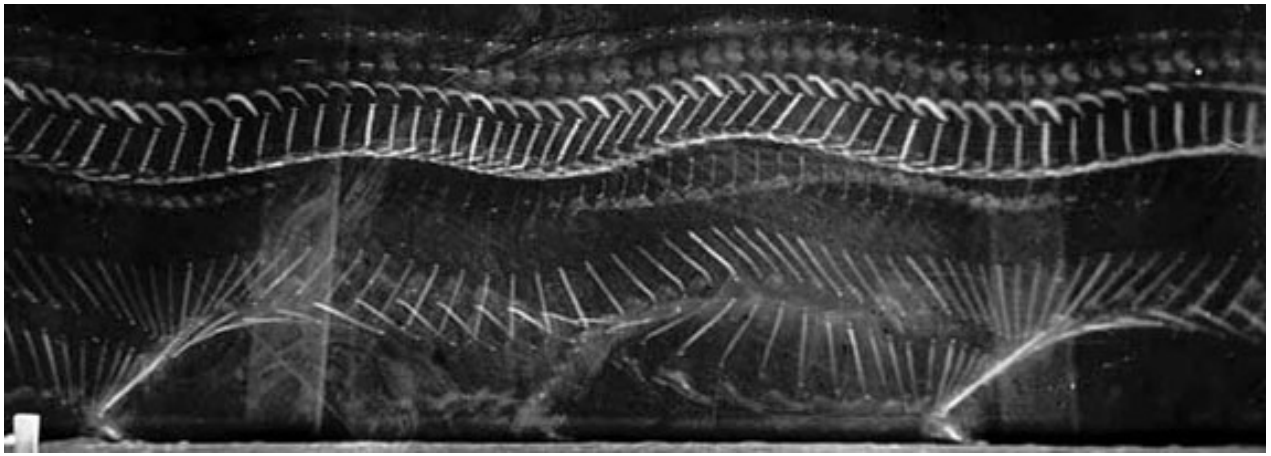
Toy Story, (Pixar 1995)

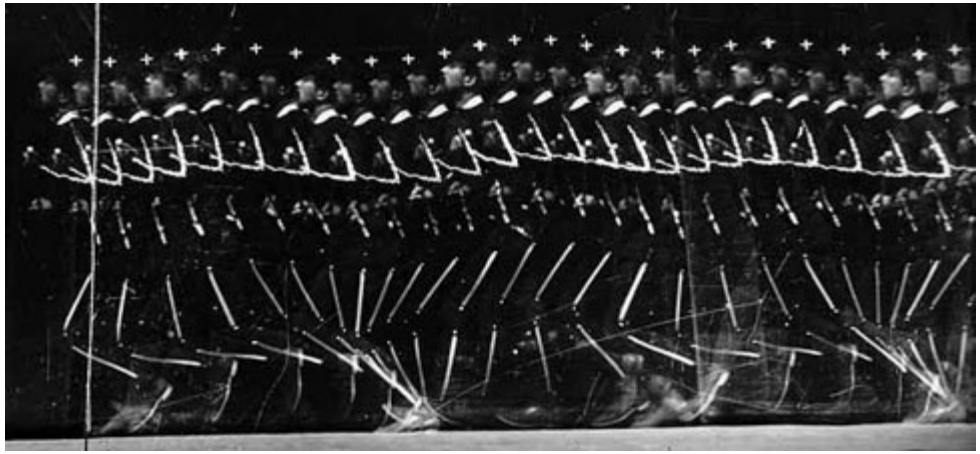
Plan

1. Introduction
2. **Cinématique**
   - Interpolation positions / vitesses
   - Modélisation hiérarchique
   - Cinématique directe
   - Cinématique inverse
3. Déformation de surfaces

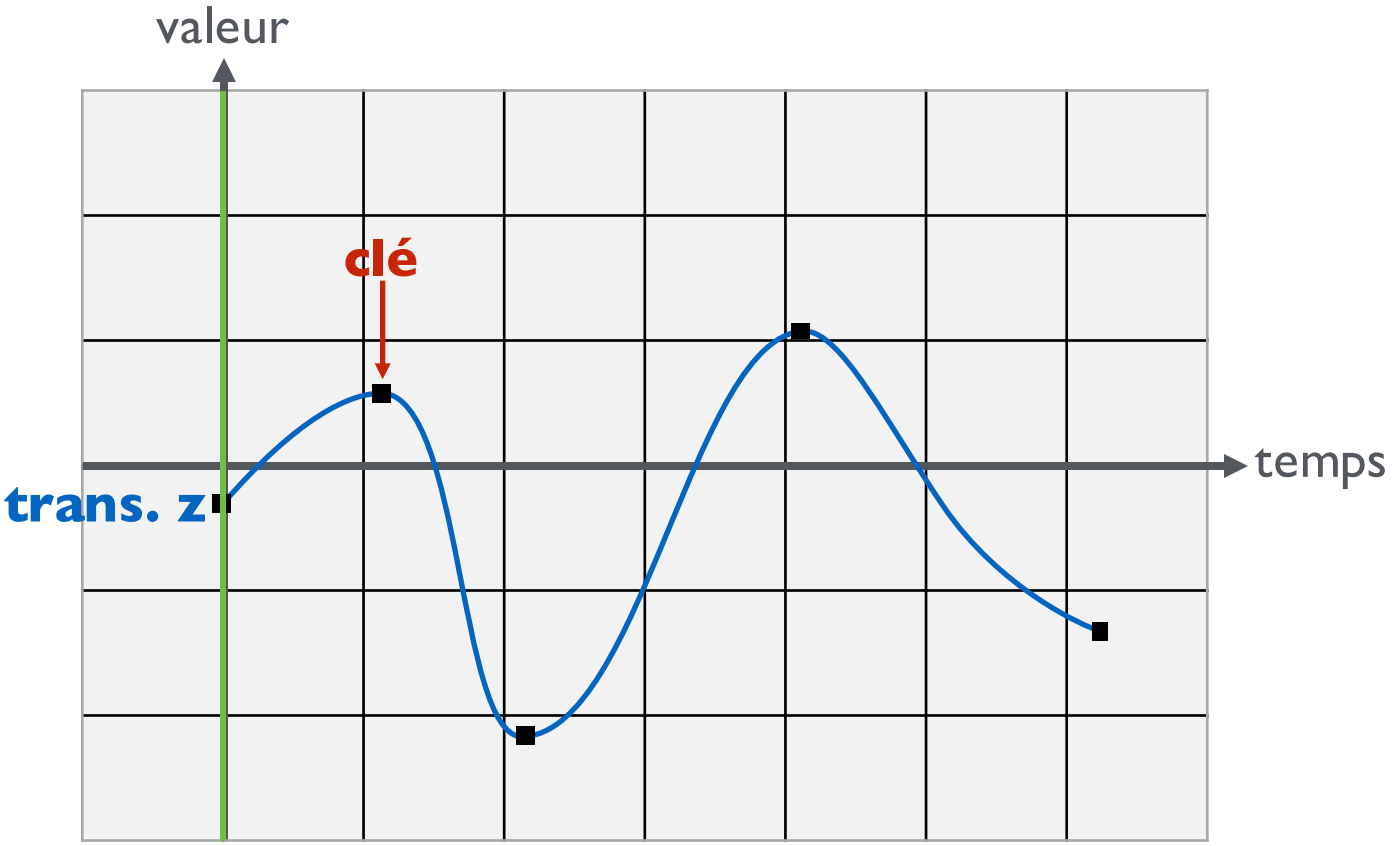
Courbe d'animation

Etienne-Jules Marey (1830-1904)





Courbe d'animation



Interpolation

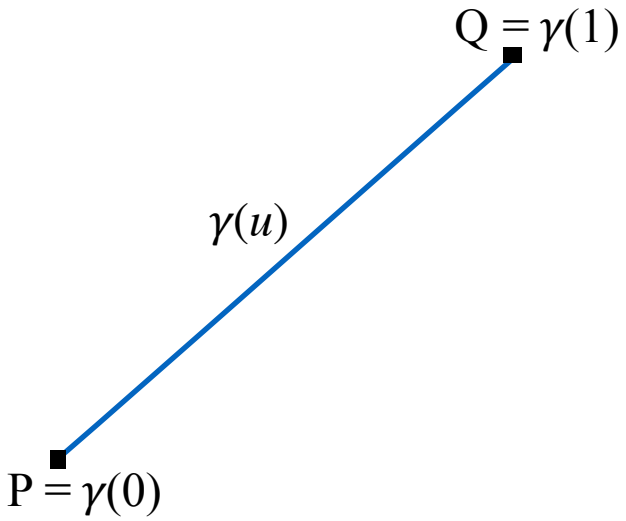
Entrées : valeurs clés (positions)

Objectif : interpoler ces valeurs clés par une fonction continue $\Longrightarrow$ courbes d'animation

- Fonction linéaire :

$$\gamma(u) = (1\text{-}u)\,\mathrm{P} + u\,\mathrm{Q}$$

- Continuité C^0 (position)



Interpolation

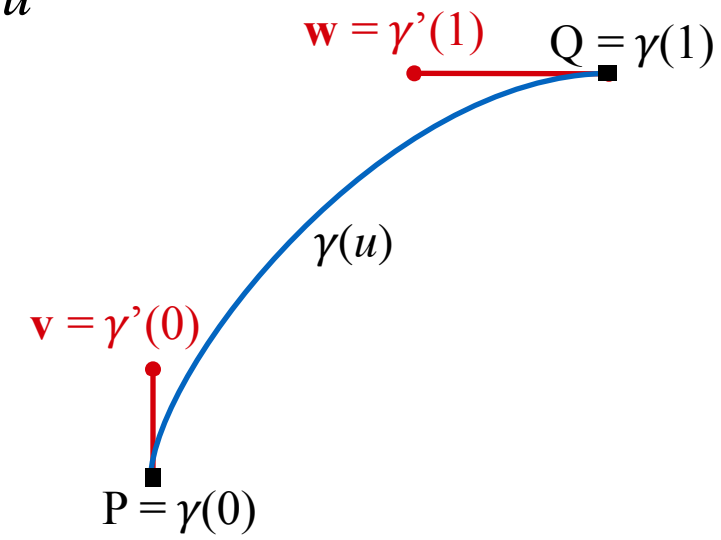
Entrées : valeurs clés (positions et vitesses)

Objectif : interpoler ces valeurs clés par une fonction continue $\Longrightarrow$ courbes d'animation

- Polynôme de degré 3 de la forme :

$$\gamma(u) = \Sigma_{n=0..3}\ a_n\ u^n$$

- Continuité C^1 entre deux clés (positions et tangentes)
- Déterminer les paramètres a_n interpolant (ou approximant) au mieux les valeurs clés
- Degré 3 $\Longrightarrow$ 4 contraintes suffisantes pour résolution exacte

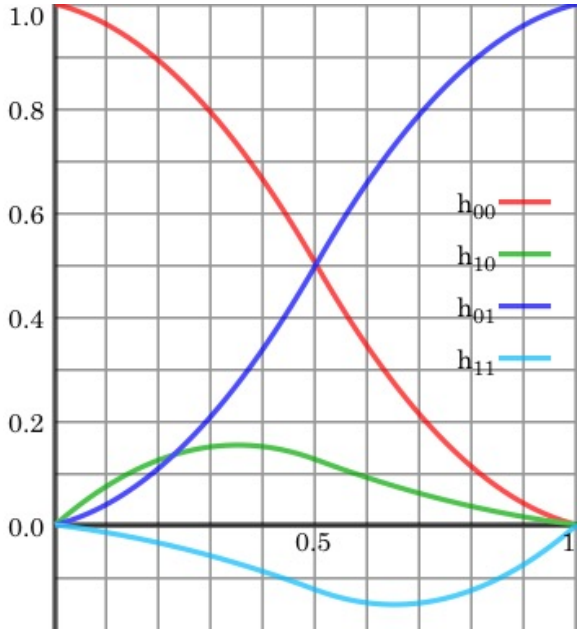


Interpolation

Polynôme d'Hermite (spline)

$$\gamma(u) = h_{00}(u)\,\mathrm{P} + h_{01}(u)\,\mathrm{Q} + h_{10}(u)\,\mathbf{v} + h_{11}(u)\,\mathbf{w}$$

avec
$$h_{00}(u) = 2u^3 - 3u^2 + 1$$
$$h_{01}(u) = -2u^3 + 3u^2$$
$$h_{10}(u) = u^3 - 2u^2 + u$$
$$h_{11}(u) = u^3 - u^2$$



polynômes de base

Notation matricielle :

$$\gamma(u) = \underbrace{[\mathrm{P}\ ;\ \mathrm{Q}\ ;\ \mathbf{v}\ ;\ \mathbf{w}]}_{\text{matrice géométrie}} \underbrace{\begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}}_{\text{matrice de base}} [1 \quad u \quad u^2 \quad u^3]^{\mathrm{T}}$$

$$\gamma(u) = \mathbf{GMT}(u)$$

Interpolation

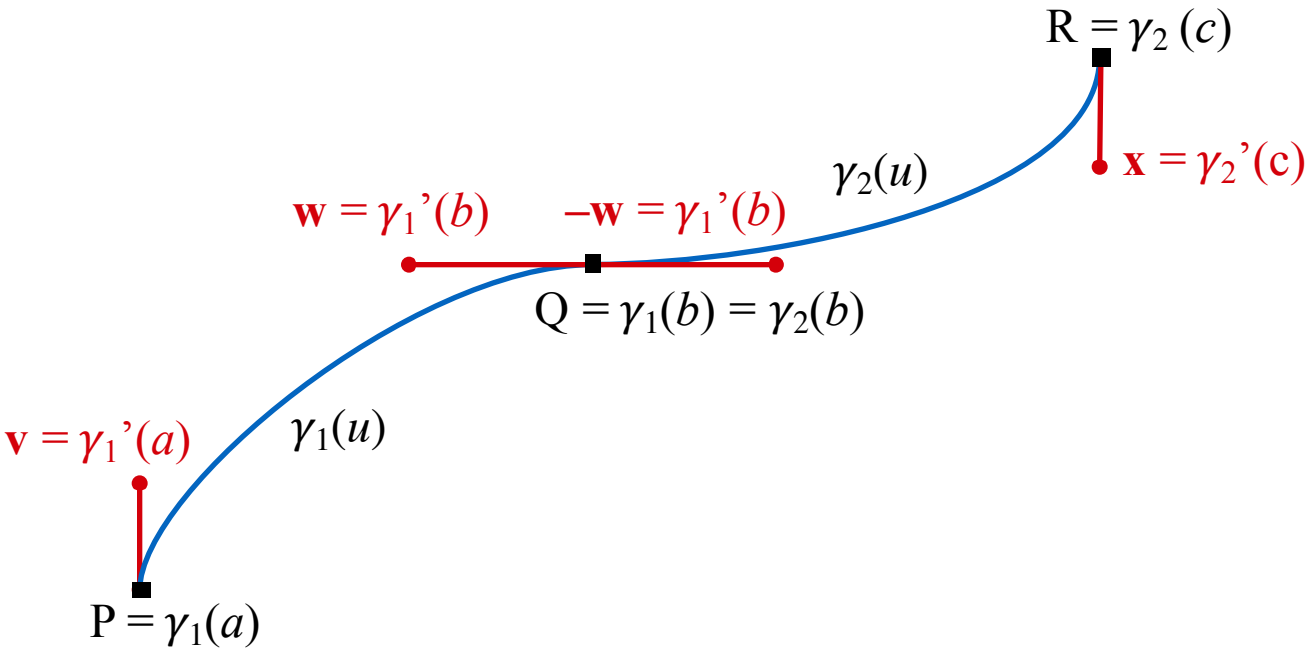
Avec 3 clés :

$$\gamma_1(u) = h_{00}(t_1)\,P + h_{01}(t_1)\,Q + h_{10}(t_1)\;\mathbf{v}\,(b-a) + h_{11}(t_1)\,\mathbf{w}\,(b-a)$$

avec $t_1 = u\text{-}a \,/\, (b\text{-}a)$

$$\gamma_2(u) = h_{00}(t_2)\,Q + h_{01}(t_2)\,R + h_{10}(t_2)\,-\mathbf{w}\,(c-b) + h_{11}(t_2)\,\mathbf{x}\,(c-b)$$

avec $t_2 = u\text{-}b \,/\, (c\text{-}b)$



Modélisation hiérarchique

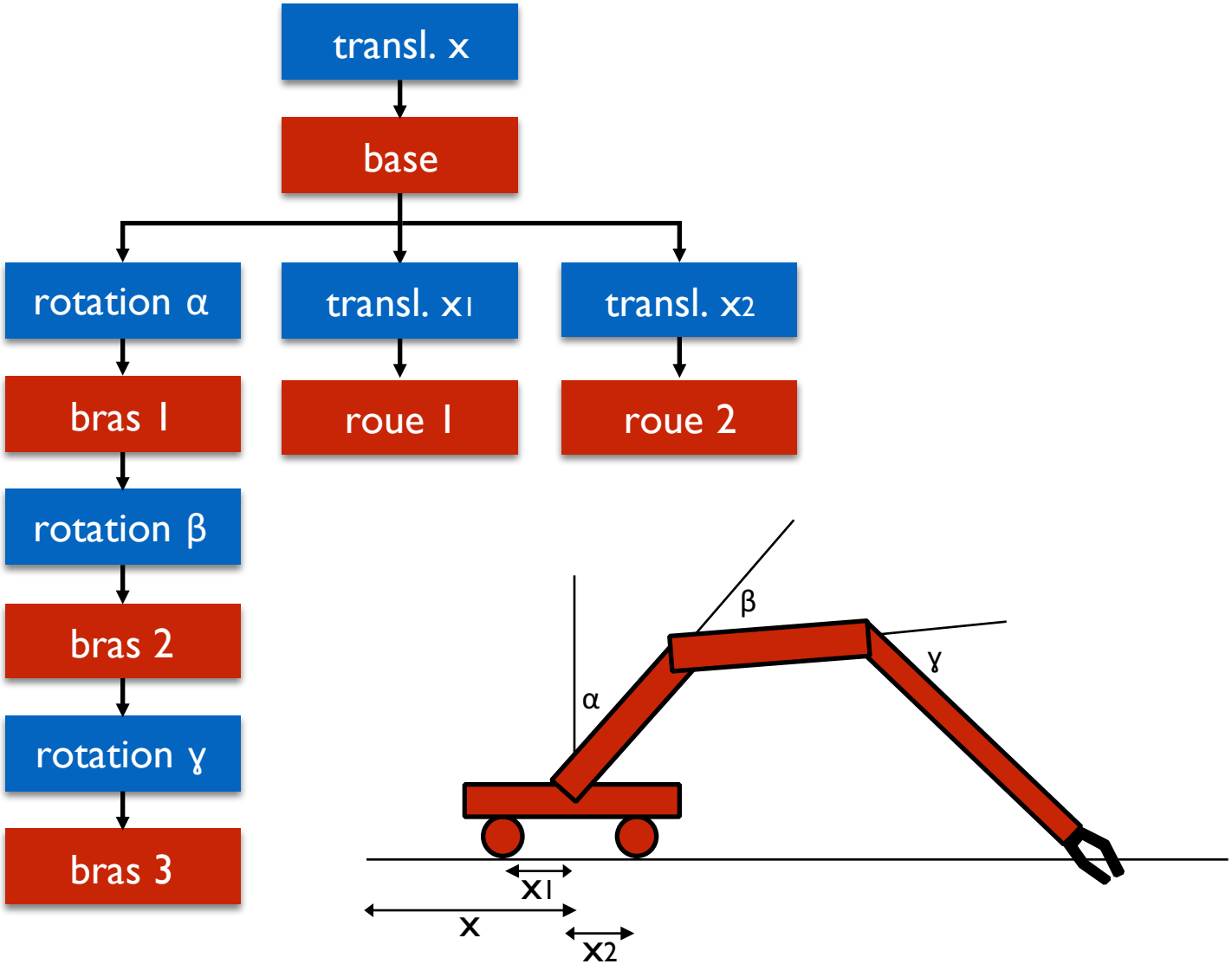
Modèle décomposé en une **hiérarchie de repères contraints**

Pas de géométrie, que des solides appelés **os** (***bones***) et des liens appelés **articulations** (***joins***)

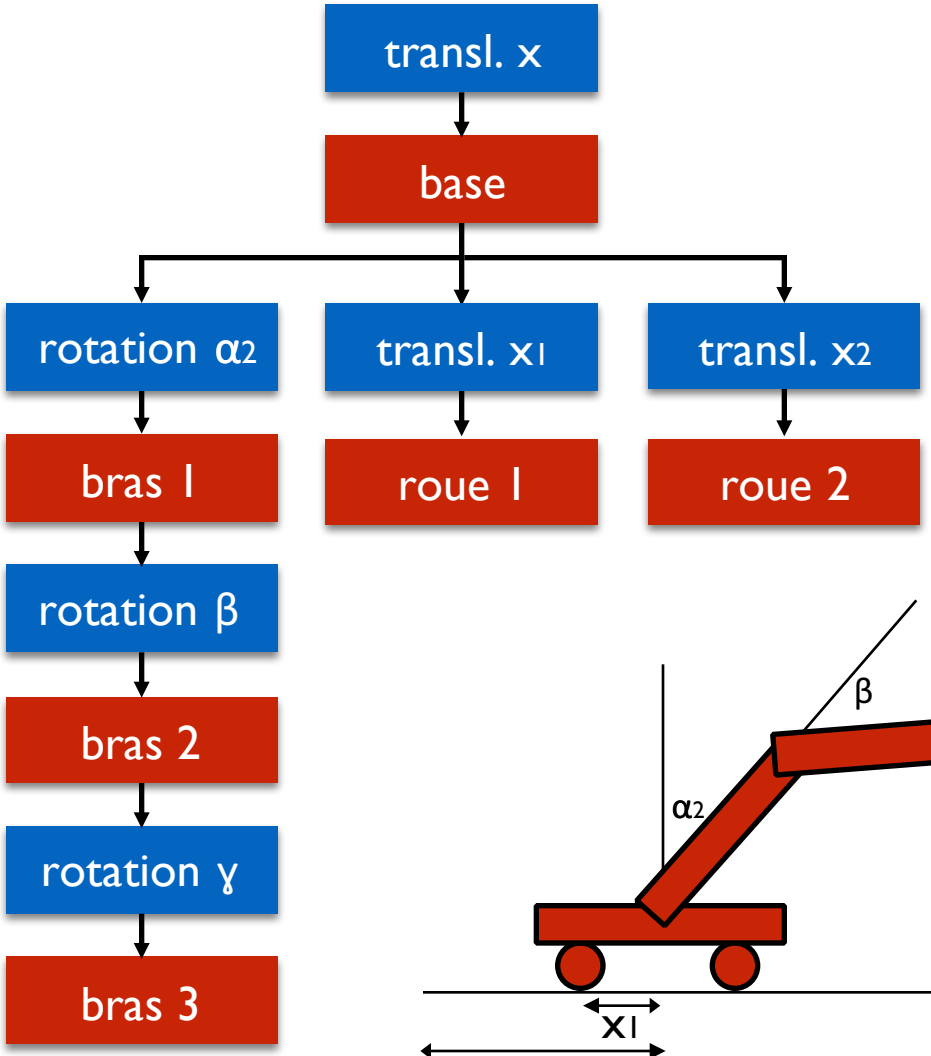
Articulation = 6 DOF

- 3 translations
- 3 rotations

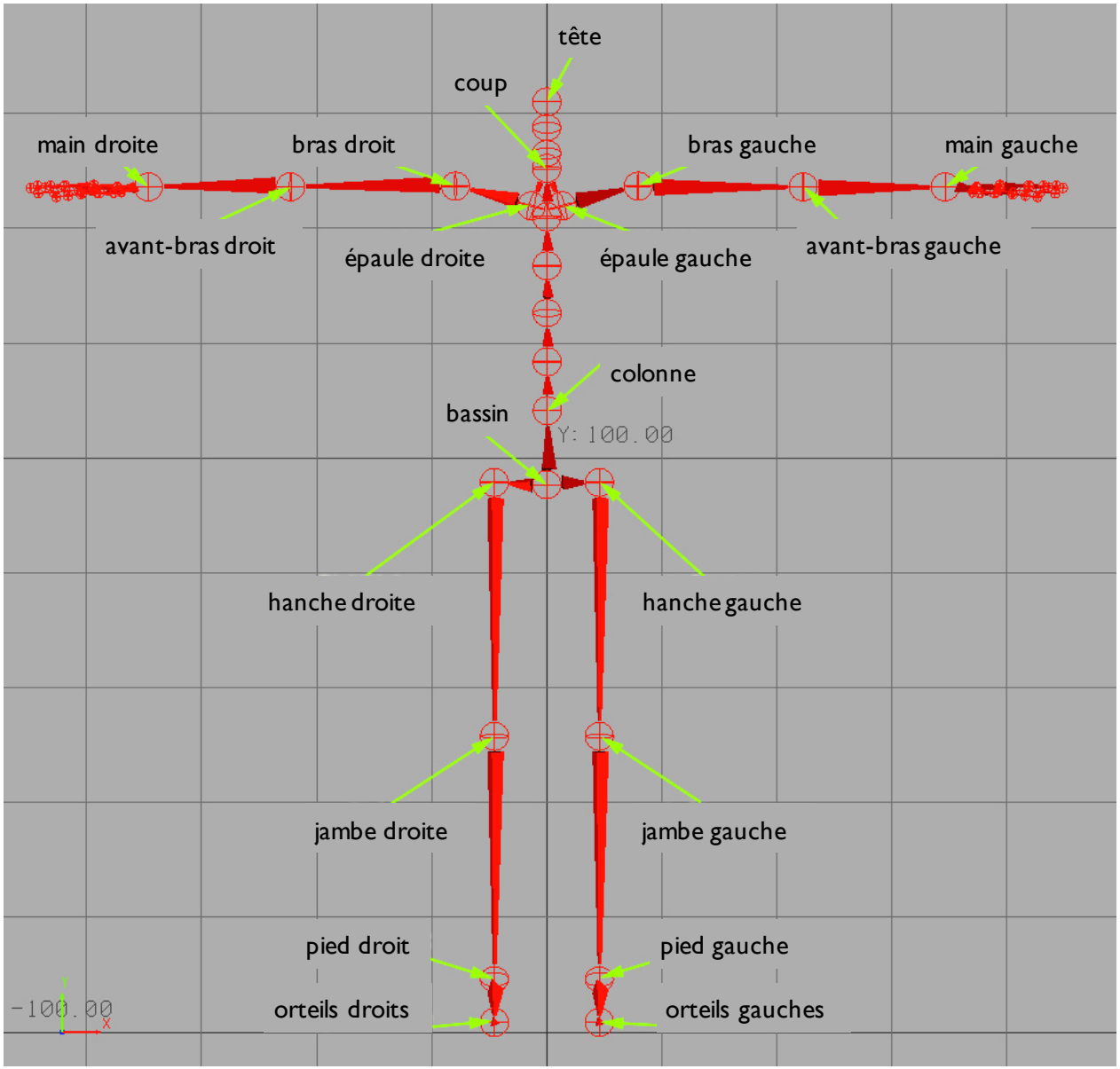
Chaîne articulée



Chaîne articulée

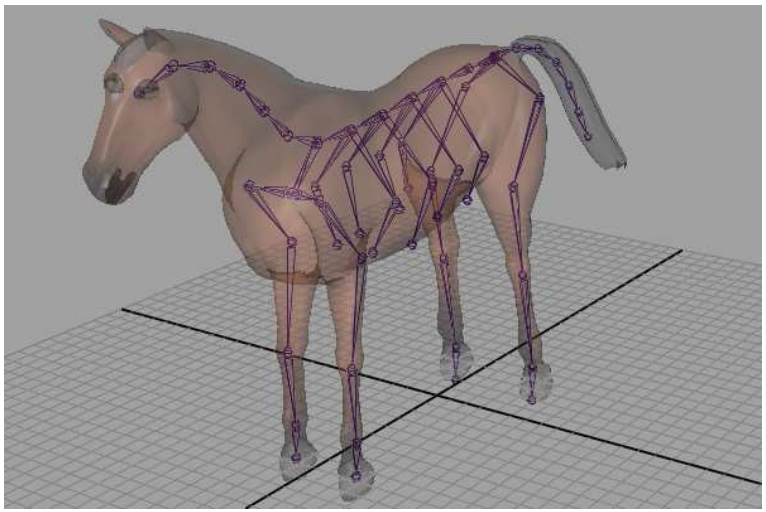


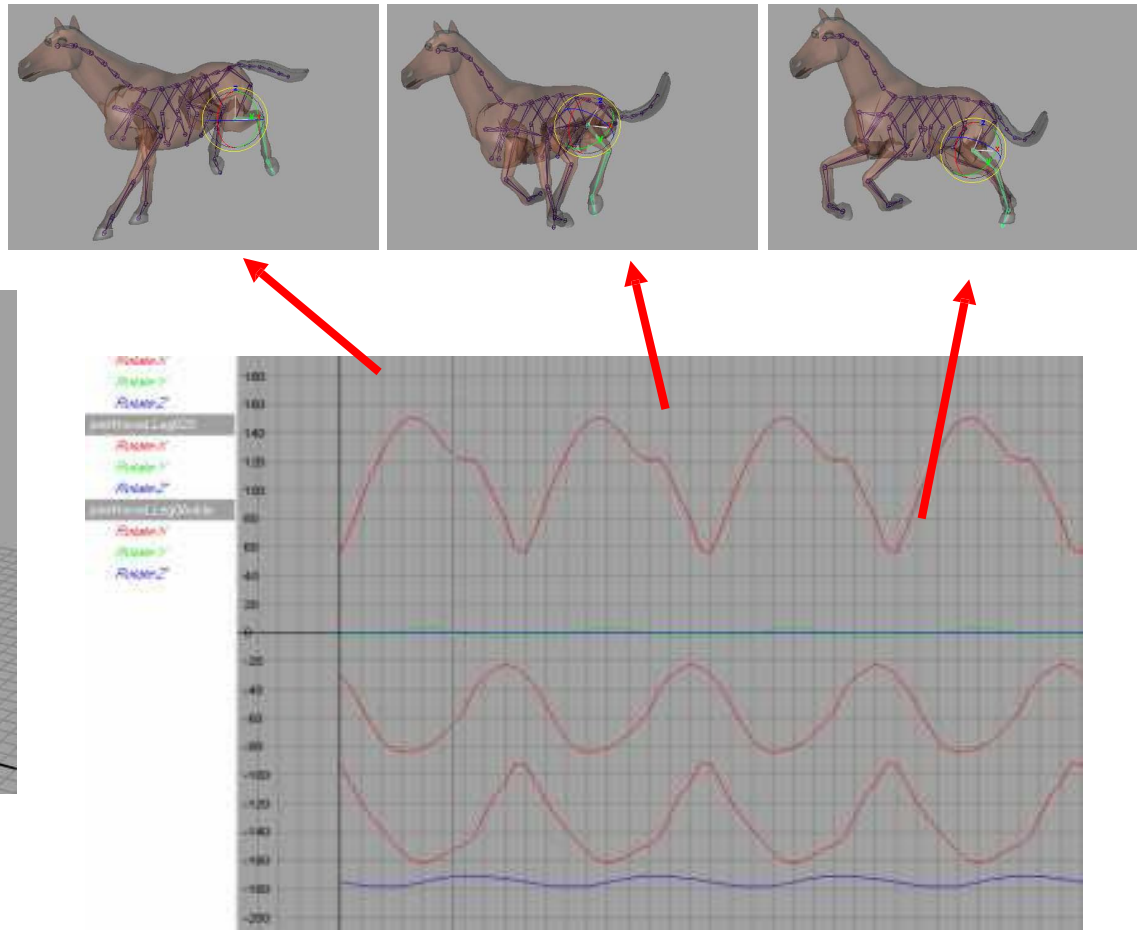
Squelette d'animation



Animation 3D

Squelette + interpolation selon une courbe





Interpolation des rotations

Représentation canonique

- Matrices orthonormales $SO(3)$
- Problème : $M_0, M_1 \in SO(3) \not\Rightarrow (1-\alpha)M_0 + \alpha M_1 \in SO(3)$
- Interpoler les coefficients puis re-orthonormaliser

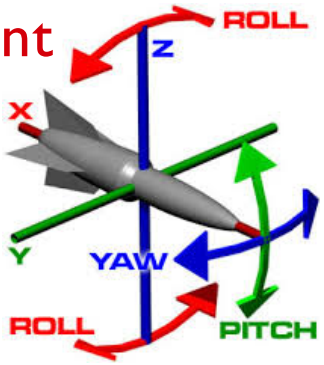
  $\Longrightarrow$ trop couteux

Angles d'Euler

- 1 angle par axe : $(\theta_x\ ,\ \theta_y\ ,\ \theta_z)$
- $M = R_{x,\theta x}\ R_{y,\theta y}\ R_{z,\theta z} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta_x & -sin\theta_x \\ 0 & sin\theta_x & cos\theta_x \end{bmatrix} \begin{bmatrix} cos\theta_y & 0 & sin\theta_y \\ 0 & 1 & 0 \\ -sin\theta_y & 0 & cos\theta_y \end{bmatrix} \begin{bmatrix} cos\theta_z & -sin\theta_z & 0 \\ sin\theta_z & cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$

  $\Longrightarrow$ attention, l'ordre des multiplications est important
- Interpoler chaque angle

  $\Longrightarrow$ intuitif et peu couteux



Interpolation des rotations

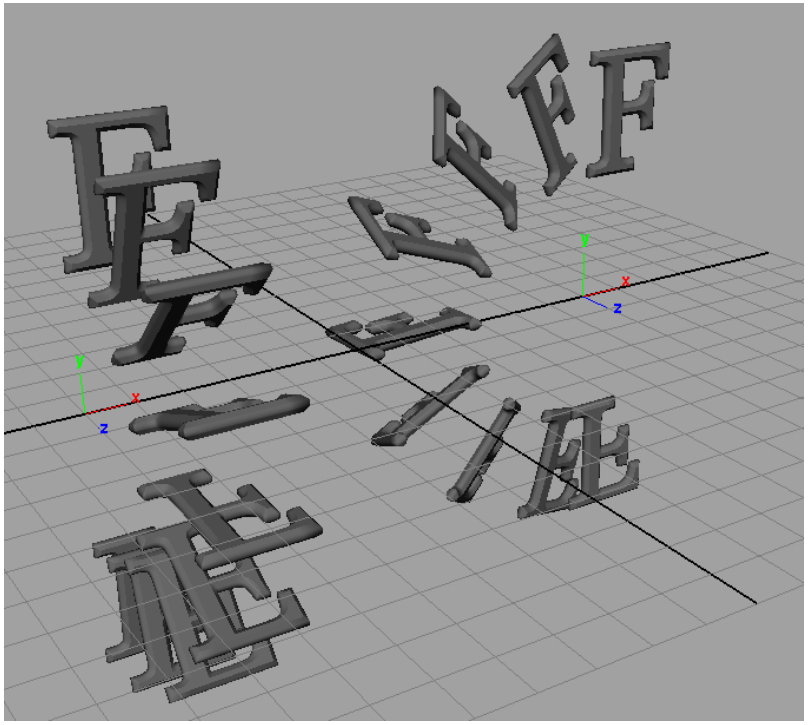
Limitations des angles d'Euler

Non-unicité de la trajectoire

$[\theta_x,\theta_y,\theta_z] = [0,0,0]$

$[\theta_x,\theta_y,\theta_z] = [\pi,0,0]$

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$



$[\theta_x,\theta_y,\theta_z] = [0,0,0]$

$[\theta_x,\theta_y,\theta_z] = [0,\pi,\pi]$

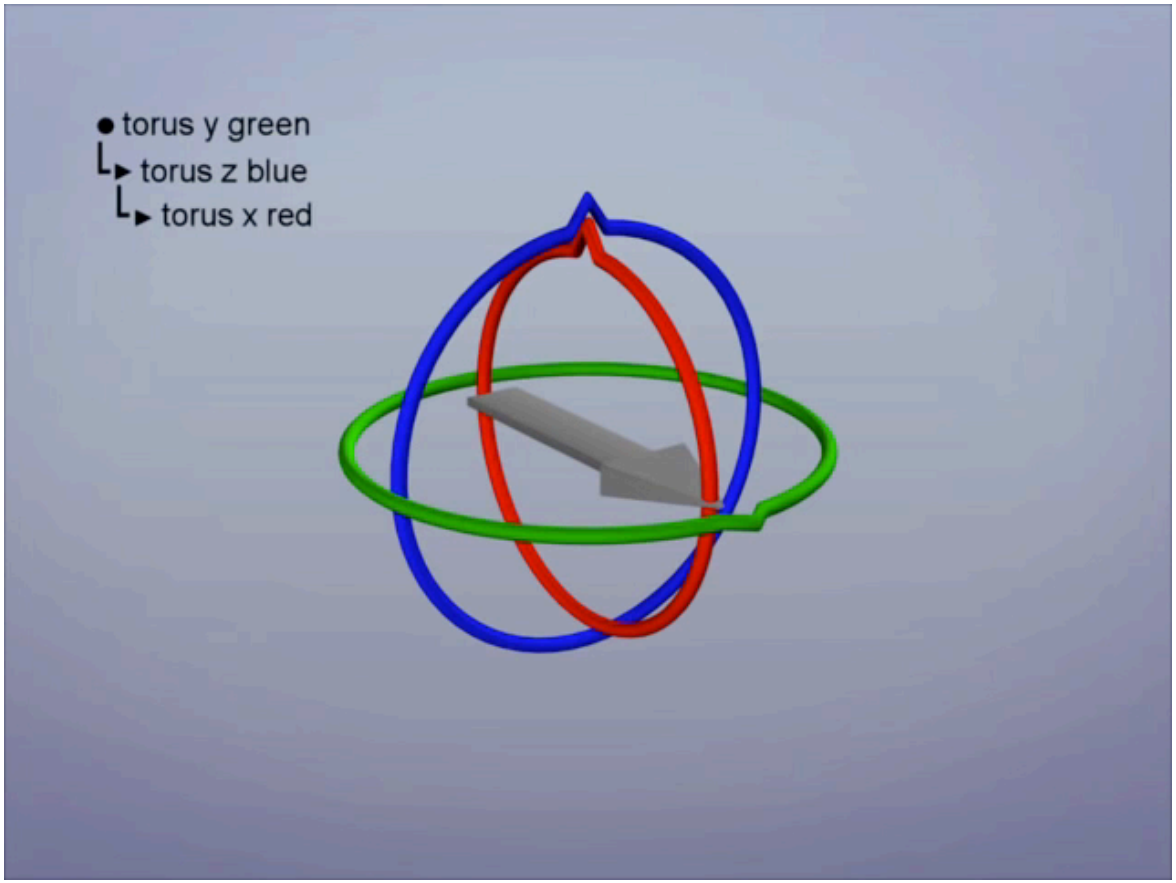
$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Interpolation des rotations

Limitations des angles d'Euler

"Gimbal lock" : perte d'un degré de liberté

$\Rightarrow$ **rotation en θ_x $\Leftrightarrow$ rotation en θ_z**



http://www.youtube.com/watch?v=zc8b2Jo7mno

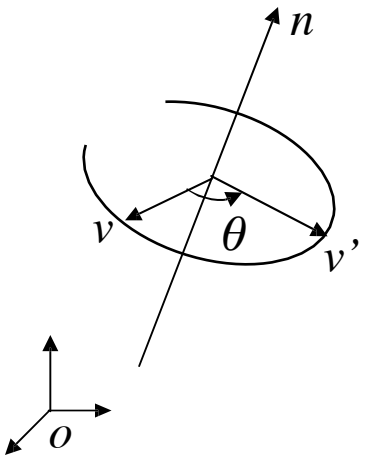
Interpolation des rotations

Axe et angle

- Toute rotation de $\mathbb{R}^3$ est une rotation planaire autour d'un axe
- Lien fort avec les quaternions

$$v' = R_{\theta,n} v = cos\theta\ v + sin\theta\ n \times v + (1 - cos\theta)(v \cdot n)\ n$$

Rodrigues' rotation formula



Interpolation des rotations

Quaternions

- Représente un **axe** et un **angle**
- Extension des nombres complexes à la 3D :

  $q = [\, s, x, y, z \,] = s + ix + jy + kz = [s, \boldsymbol{n}]$

  avec $i^2 = j^2 = k^2 = ijk = -1$
- Comme les nombres complexes :
  - Conjugué : $q^* = [s, -x, -y, -z]$
  - Norme : $||q|| = qq^* = s^2 + x^2 + y^2 + z^2$
  - Inverse : $q^{-1} = q^*/q$
- Multiplication :

  $q_1 q_2 = [s_1, \boldsymbol{n_1}]\ [s_2, \boldsymbol{n_2}] = [s_1 s_2 - \boldsymbol{n_1} \cdot \boldsymbol{n_2},\ s_1 \boldsymbol{n_2} + s_2 \boldsymbol{n_1} + \boldsymbol{n_1} \times \boldsymbol{n_2}]$

Interpolation des rotations

Quaternions

Toute rotation 3D peut être représentée par un quaternion :

$$x' = R_{\theta,n}\, x \Leftrightarrow [0, x'] = q[0, x]q^{-1}$$

avec $q = [cos(\theta/2), sin(\theta/2)\boldsymbol{n}]$

et $q^{-1} = [cos(\theta/2), -sin(\theta/2)\boldsymbol{n}]$

Composition de rotations = **multiplication de quaternions**

si q_1 représente la rotation R_1 et q_2 représente R_2,
alors $q_1 q_2$ représente la rotation $R_1 R_2$

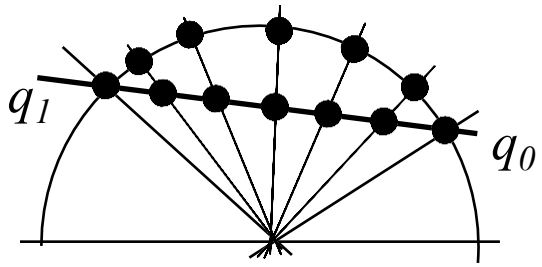
Interpolation des rotations

Quaternions

Interpolation linéaire (lerp) incorrecte

$$lerp(q_0, q_1, t) = (1\text{-}t)q_0 + tq_1$$

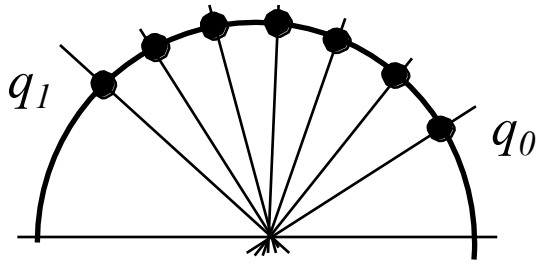
$\Longrightarrow$ ne préserve pas la vitesse angulaire



Interpolation sphérique (slerp)

[Ken Shoemake 85]

$$slerp(q_0, q_1, t) = q_0(q_0^{-1}q_1)^t$$
$$= (q_1q_0^{-1})^tq_0$$



Cinématique

Mouvement non-contraint

Hocher la tête, faire un signe de la main…

$\Longrightarrow$ Cinématique directe (FK)

Mouvement contraint

Attraper un objet, marcher sur le sol…

$\Longrightarrow$ Cinématique inverse (IK)

Cinématique directe

Entrées : valeurs des paramètres des articulations à un temps donnée :

$$\Theta = (\theta_1, \theta_2, \theta_3, \ldots, t_1, t_2, \ldots)$$

Sortie : position à atteindre M

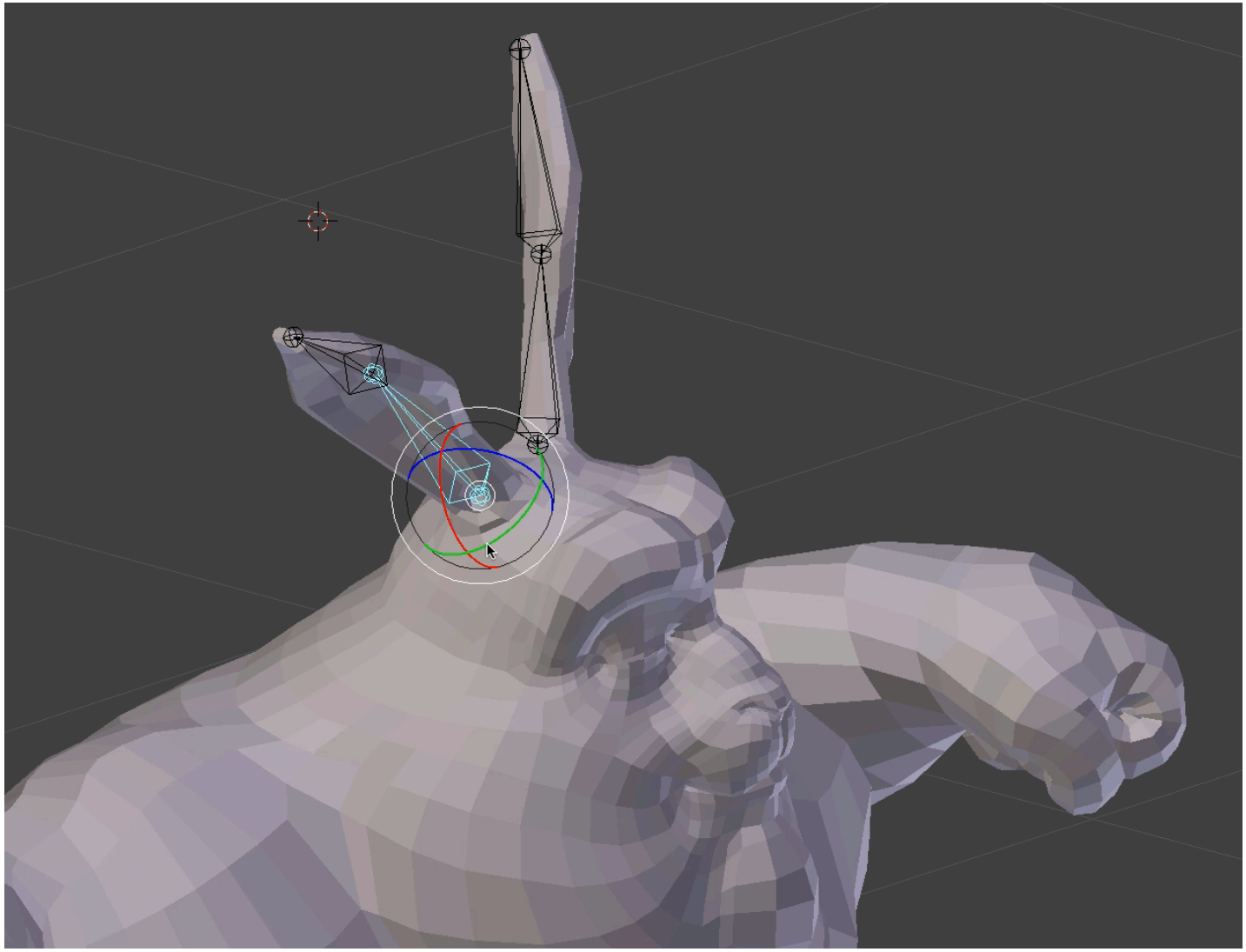
On **calcule** : $f(\Theta) = \mathrm{M} = R_1(\theta_1)T_1R_2(\theta_2)T_2 \ldots \mathrm{M}_0$
où M_0 est la position initiale

$\Longrightarrow$ **interpolation des orientations clés**

Problèmes

- Contrôle des extrémités
- Accumulation des erreurs

Cinématique directe



Cinématique inverse

Entrée : position à atteindre M

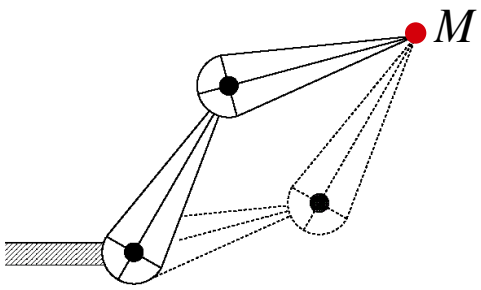
Sortie : valeurs des paramètres des articulations

$$\Theta = (\theta_1, \theta_2, \theta_3, \ldots, t_1, t_2, \ldots)$$

On veut **trouver** Θ tel que $f(\Theta) = M$

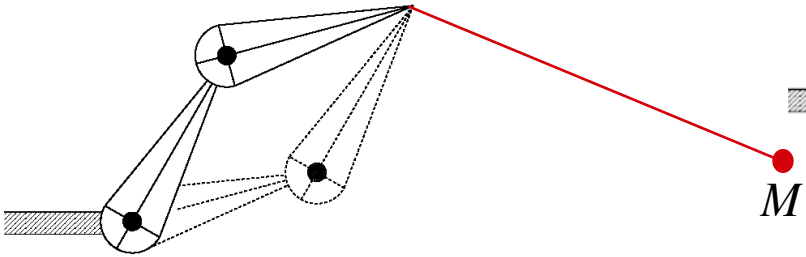
Difficultés :

- Solution analytique seulement pour 2 à 3 rotations
- Non-linéarité à cause des rotations (*cos*/*sin*)
- Non-unicité de la solution

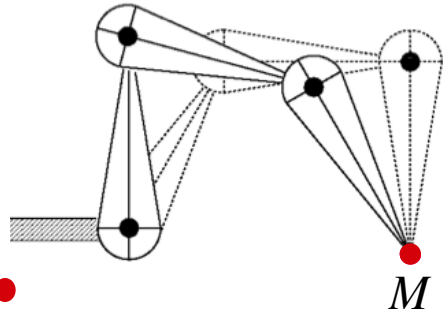


2 solutions

pas de solution



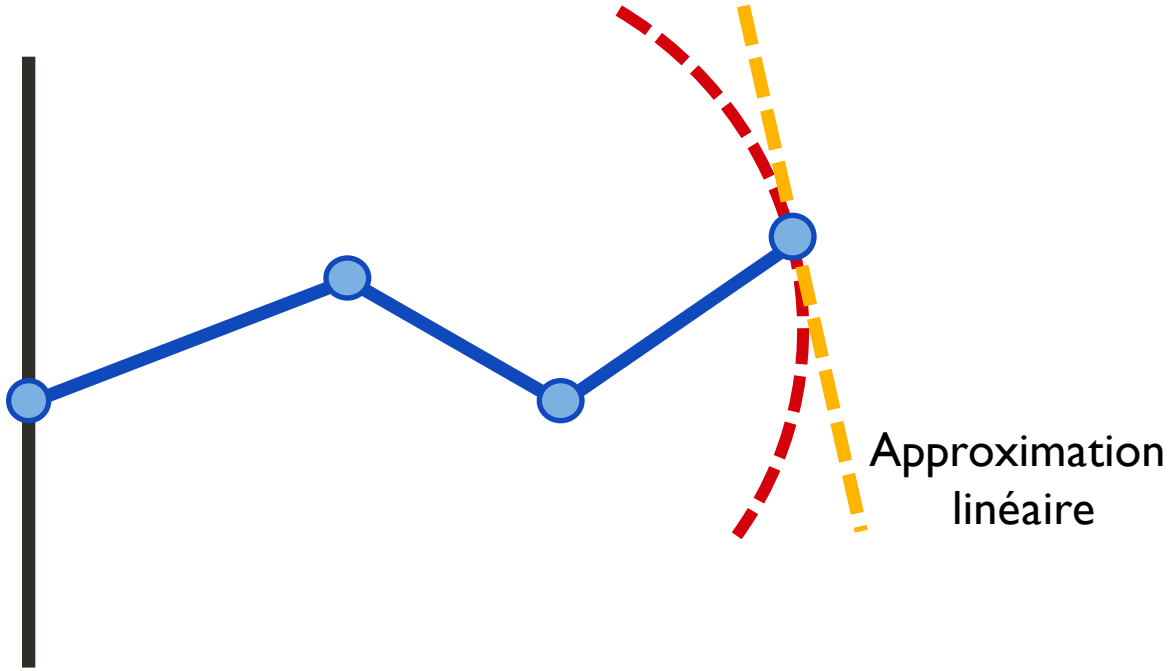
une infinité de solutions



Cinématique inverse

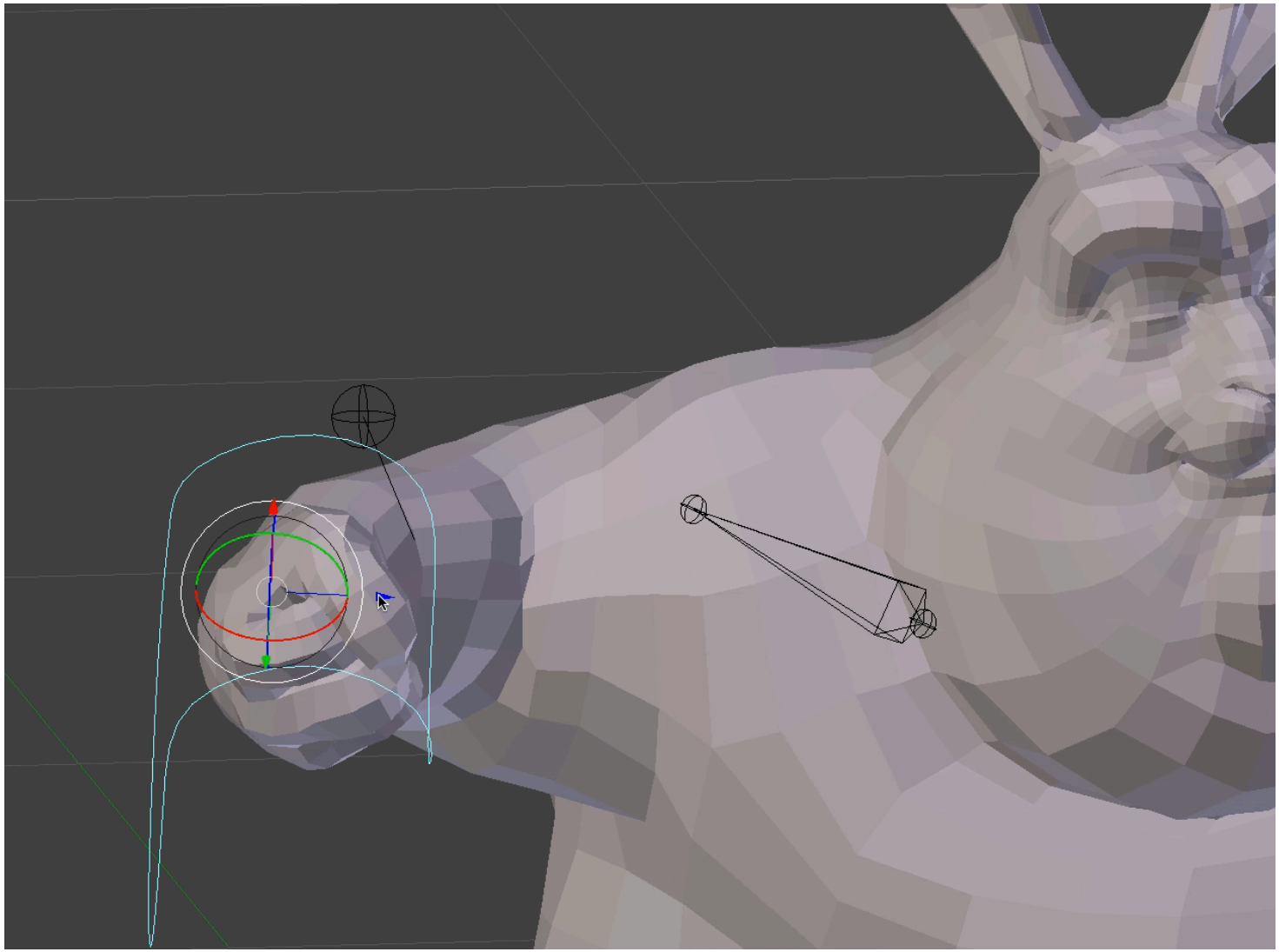
Objectif : trouver Θ tel que $f(\Theta) = \mathrm{M}$

Linéarisation du problème



...et minimisation de la distance $\| f(\Theta_0 + \Lambda) - \mathrm{M} \|$

Cinématique inverse



Linéarisation du problème

Objectif : trouver Θ tel que $f(\Theta) = \mathrm{M}$

Linéarisation du problème :

- Depuis une configuration initiale Θ_0
- On cherche Λ tel que ll $f(\Theta_0 + \Lambda) - \mathrm{M}$ ll est minimal
- Si Λ est petit, $\mathrm{M} \approx \Theta_0 + f'(\Theta)\Lambda$
- En 1D, série de Taylor :

$$f(\Theta + \Lambda) = f(\Theta) + f'(\Theta)\Lambda + f''(\Theta)\Lambda^2 + \ldots$$

- En dimensions supérieures :

$$f(\Theta + \Lambda) = f(\Theta) + J(\Theta)\Lambda + \Lambda^t H(\Theta)\Lambda + \ldots$$

avec J = **Jacobien** de f, forme linéaire

H = **Hessien** de f, forme quadratique

Algorithme

Approximation linéaire (petits déplacements) :

$$J(\Theta)\Lambda = \mathrm{M} - \Theta_0$$

Résolution du système linéaire $\Longrightarrow \Lambda$

calculer la pseudo-inverse de J : $J^+ = J^\top(JJ^\top)^{-1}$ si $\mathrm{n} > 3$

J matrice $3 \times \mathrm{n}$ $= (J^\top J)^{-1}J^\top$ si $\mathrm{n} < 3$

Mise-à-jour : $\Theta = \Theta_0 + \Lambda$

…et on recommence

itérations = **série de petits déplacements pour atteindre la cible** M

Limite aux articulations

Important pour le réalisme :

respect des **contraintes morphologiques**

Ex. : l'angle du coude varie sur $[0, \pi]$

Modifications de l'algorithme

- Tester si dépassement pour paramètre i
- Annuler paramètre i
  (en pratique, enlever la colonne correspondante dans J)
- Recalculer J, J^+ et Θ sans i
- Vérifier les autres paramètres

Plus d'infos : http://billbaxter.com/courses/290/html/

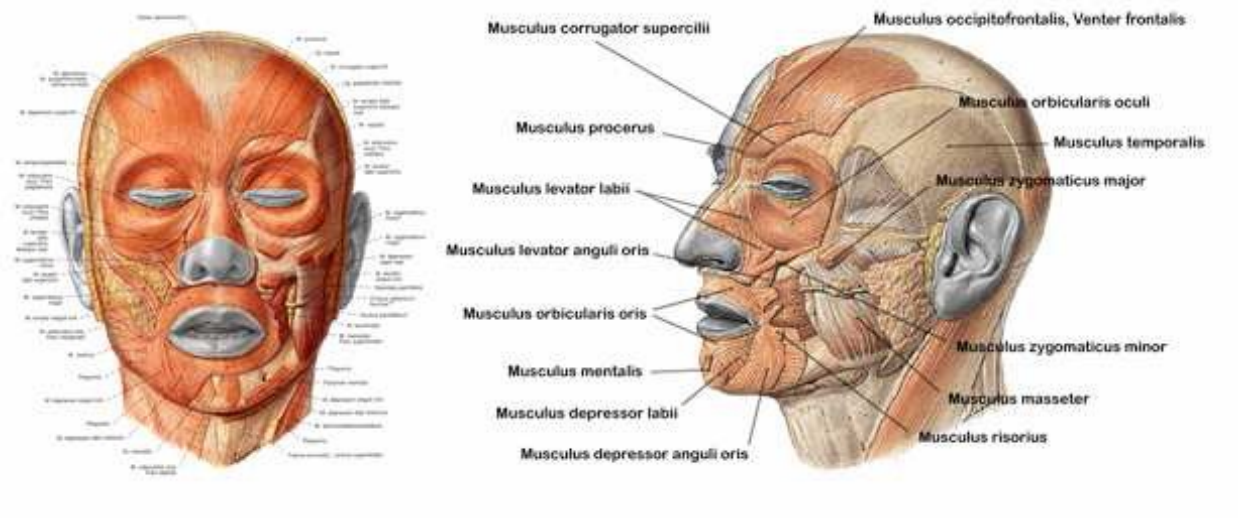
Plan

1. Introduction
2. Cinématique
3. **Déformation de surfaces**
   - Skinning
   - Déformeurs non-linéaires
   - Morphing

Déformation de surfaces

Squelette d'animation vs. modèle visuel

En réalité, forme visible composée de **tissus organiques**



De quel niveau de détails se satisfaire dans l'animation ?

Modèle visuel

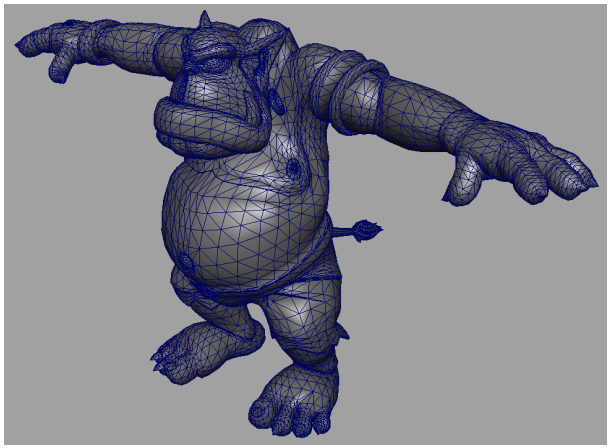
Le niveau de réalisme dépend de l'application

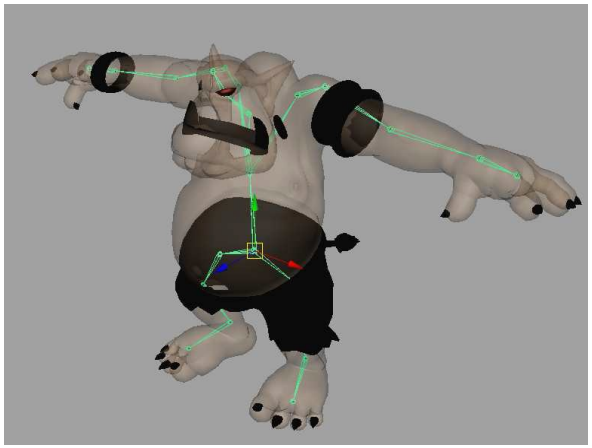
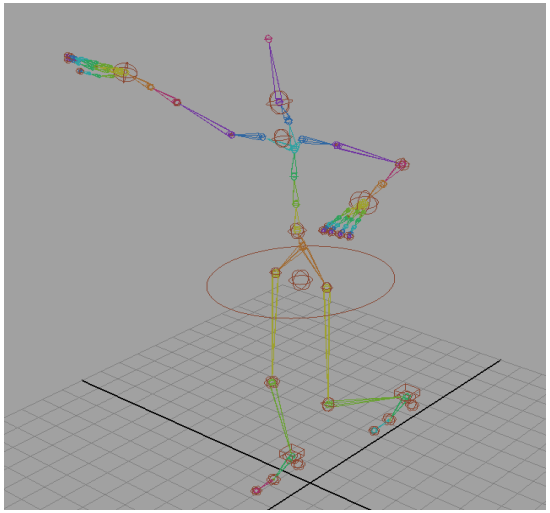
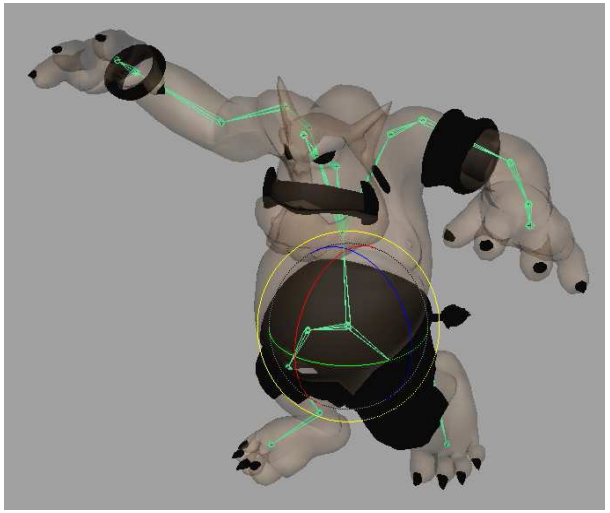
Déformation de surfaces

Souvent, on se satisfait de **l'approximation par une surface 3D**

- ✓ rapport direct avec la modélisation (forme et texture)
- ✓ structure légère facile à animer (en temps réel)
- ✗ approximation grossière de la réalité

Pipeline 3D




Rigging

Entrées : sommets p_i de la surface 3D (*skin*) et squelette dans une pose de base (*bind pose*)

Sortie : positions p_i' après mouvement du squelette

- On connaît p_i dans le repère monde (ou objet)
- On connaît la matrice B_j de passage de l'os j au repos au repère monde

$\Longrightarrow$ nouvelles coordonnées : $p_i' = T_j B_j^{-1} p_i$
avec T_j la matrice de transformation de l'os j

Skinning linéaire

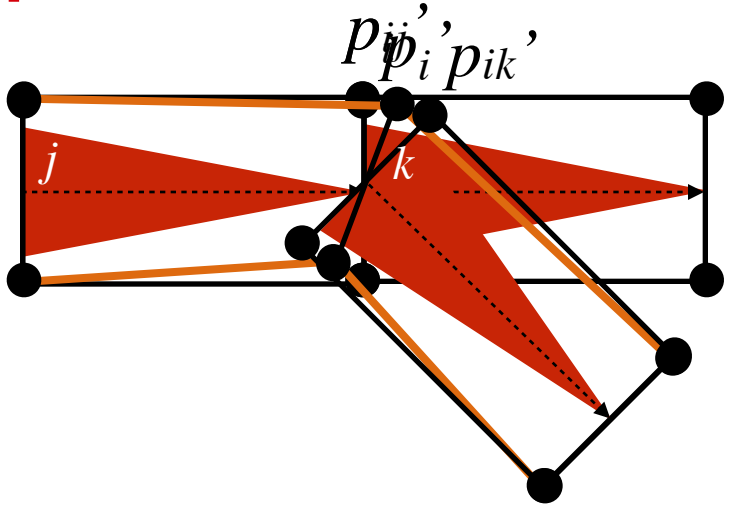
Sommet influencés par plusieurs os

$\Longrightarrow$ interpolation linéaire des positions

Si p_i lié aux os j et k :

$$p_i' = w\, p_{ij}' + (1\text{-}w)\, p_{ik}'$$

w : [0,1] *skin weight*



Cas général

$$p_i' = \Sigma_j\, w_{ij}\, T_j B_j^{-1}\, p_i$$

$$= (\Sigma_j\, w_{ij}\, M_j)\, p_i$$

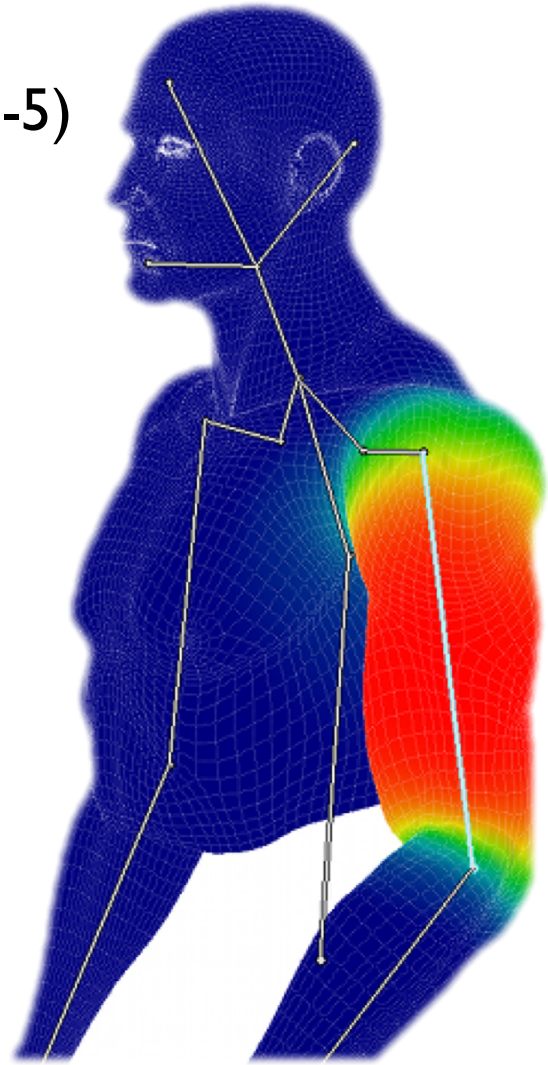
avec $\Sigma_j\, w_{ij} = 1$

Poids de skinning

Spécifiés (« peints ») par un artiste os par os

- $w_{ij} = 1 \Longrightarrow$ mouvement rigide avec l'os
- Limiter le nombre de poids par sommet (4-5)
  $\Longrightarrow$ coût de la sommation
  $\Longrightarrow$ stockage mémoire

et/ou calculés automatiquement
(distance euclidienne, harmonique, géodésique)



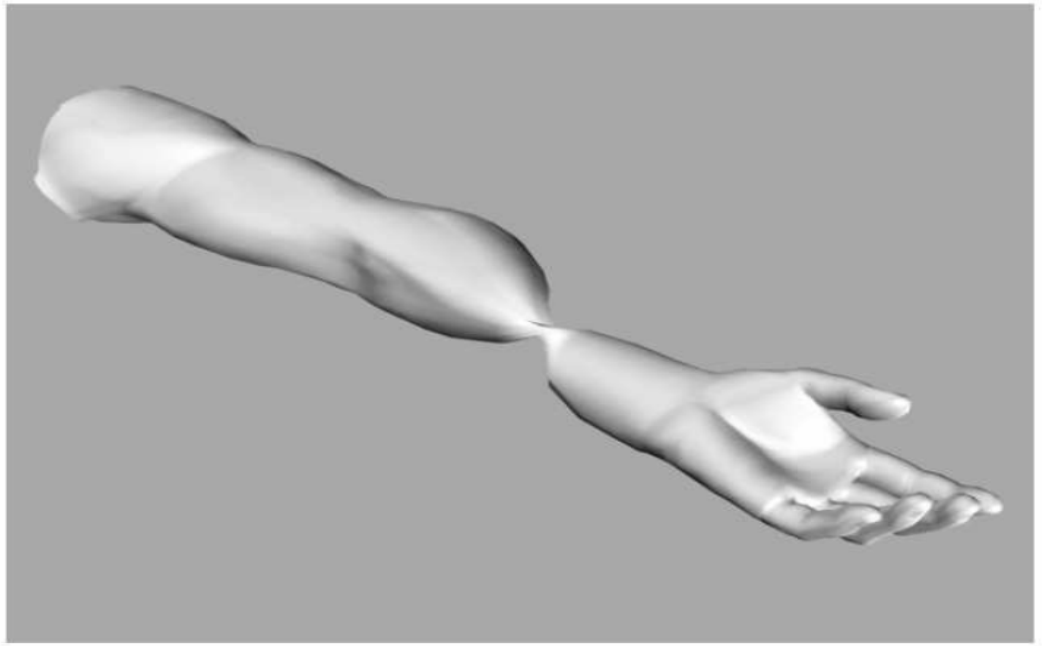
Skinning linéaire

1. **Rigging – pre-process**
   $\Longrightarrow$ matrices de pose : B_j
   $\Longrightarrow$ poids de skinning : w_{ij}
2. **Cinématique direct ou inverse – à chaque temps *t***
   $\Longrightarrow$ matrices de transformation $M_j(t) = T_j(t)\ B_j^{-1}$
3. **Skinning linéaire – *vertex shader***
   $\Longrightarrow p_i' = (\Sigma_j\ w_{ij} M_j(t))\ p_i$

Skinning linéaire

Limitation : rotations

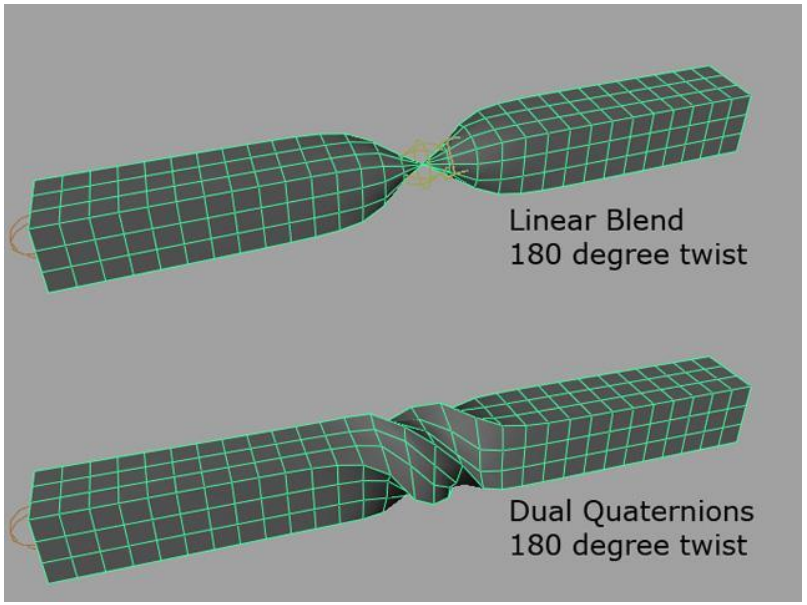
(la moyenne de 2 rotations n'est pas une rotation !)



Skinning + quaternions duaux

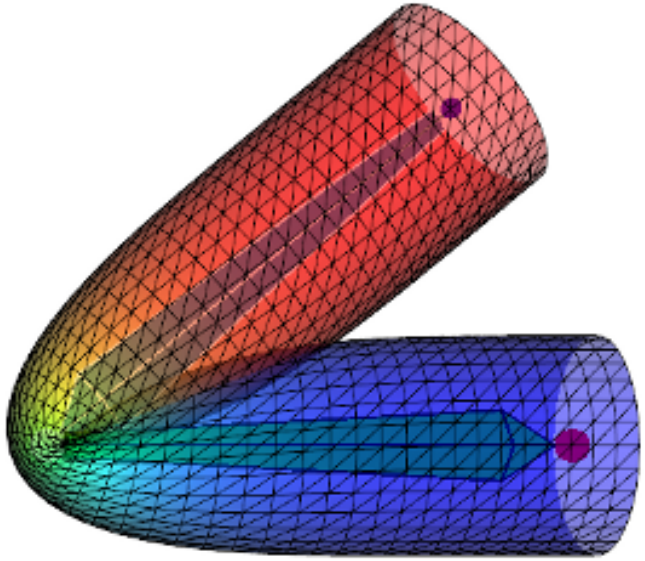
[Kavan et al. 08]

Calculer l'interpolation de matrice en maintenant des rotations correctes à l'aide de quaternions duaux

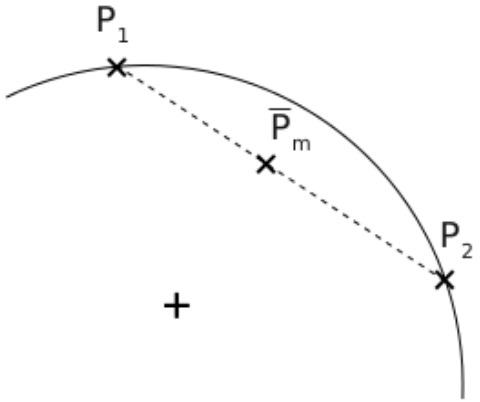


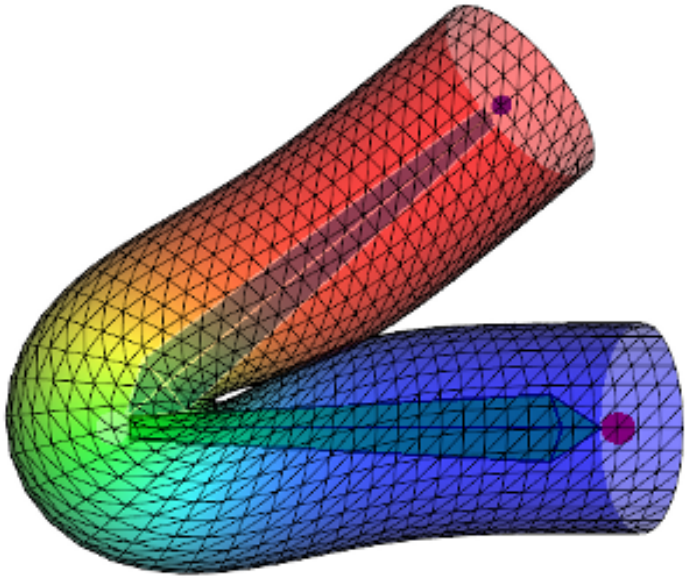
Skinning + quaternions duaux

[Kavan et al. 07]

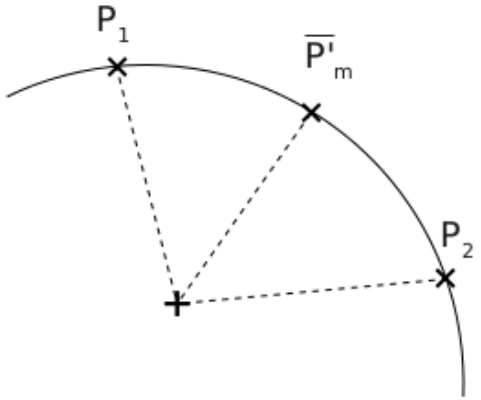


Skinning linéaire





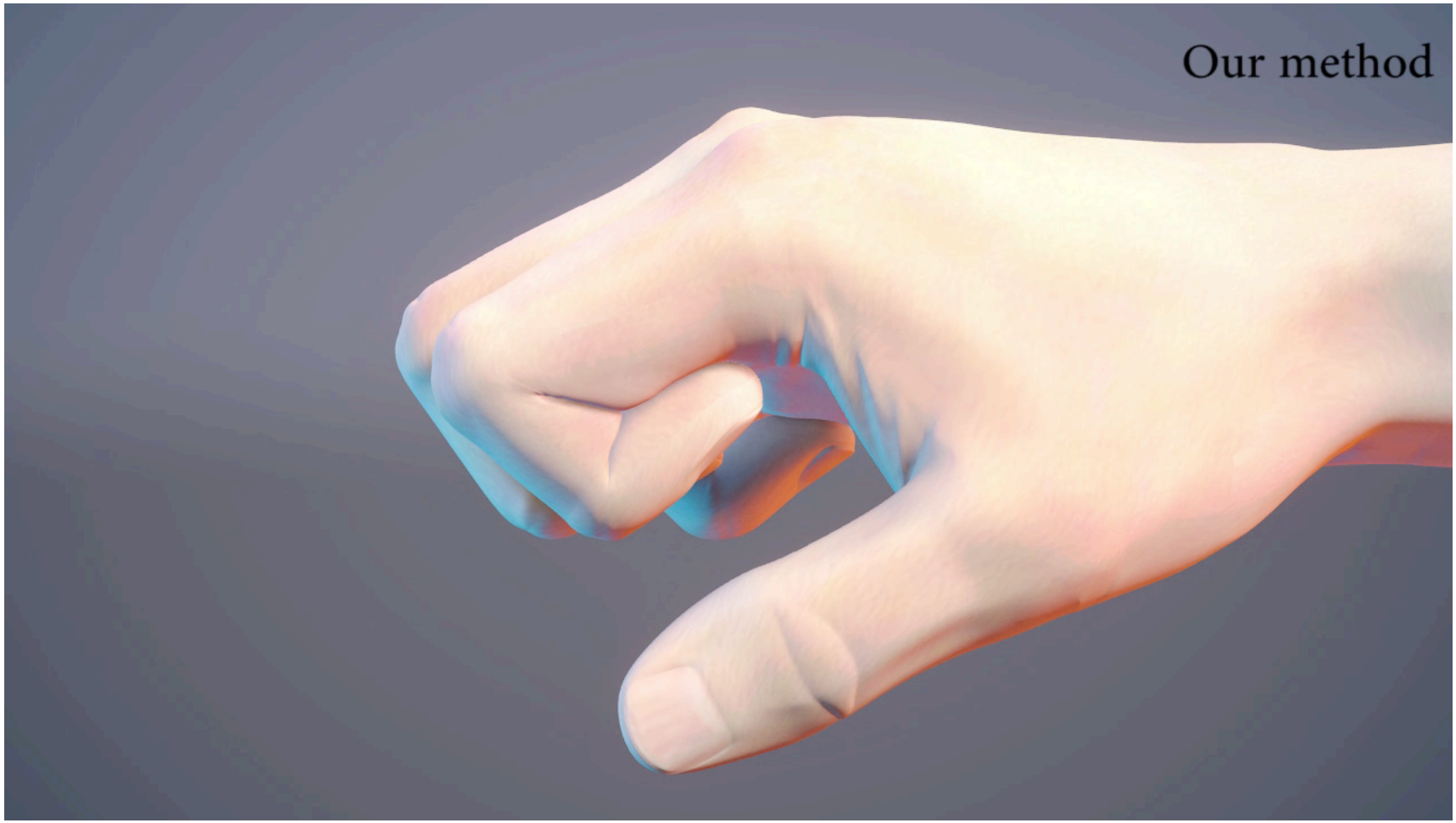
Dual quaternions



Skinning + surfaces implicites

[Vaillant et al. 13]

Prise en compte des **contacts**



Deformers non-linéaire

Modification globale de l'espace 3D

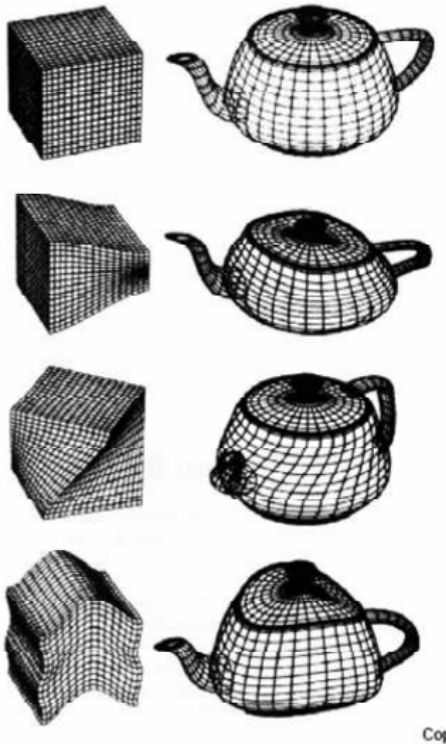
Définir une **fonction dans l'espace** :

$$p \in \mathbb{R}^3 \longrightarrow \mathrm{M}(p)$$

où $\mathrm{M}(p)$ est une matrice de transformation

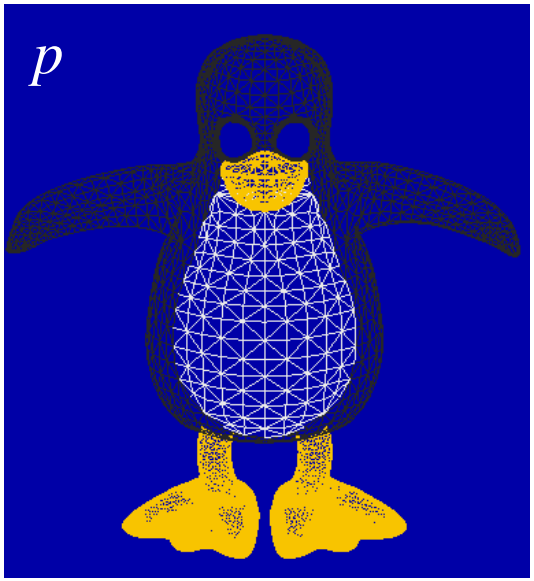
Action sur un point p

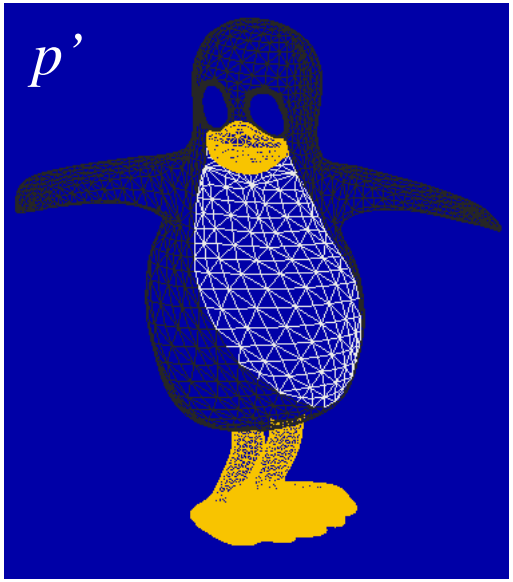
- Évaluer M au point p : $\mathrm{M}(p)$
- Faire agir M sur P : $p' = \mathrm{M}(p)\mathrm{p}$



Twist (rotation non-uniforme)

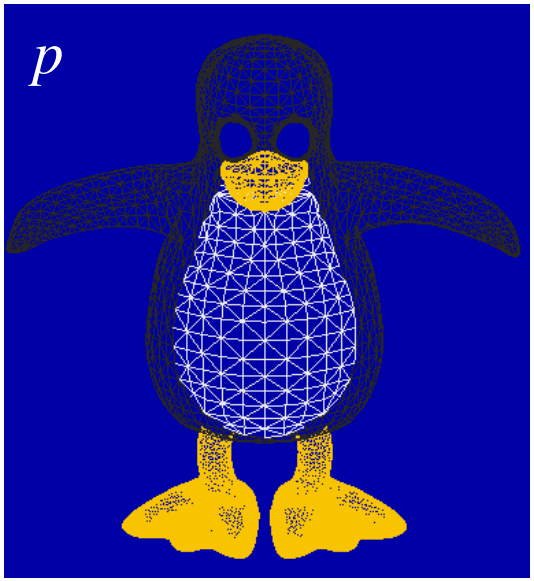
$$r(z) = \begin{cases} 0 & z \leq z_0 \\ \frac{z - z_0}{z_1 - z_0}\theta_{\max} & z_0 \leq z \leq z_1 \\ \theta_{\max} & z_1 \leq z_0 \end{cases}$$

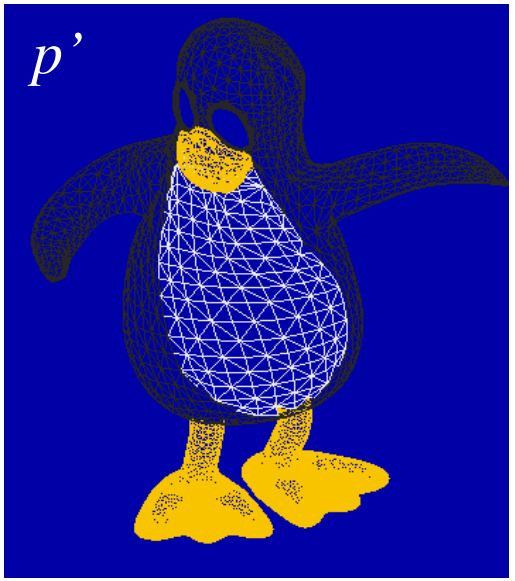




$$p' = \begin{bmatrix} \cos(r(p_z)) & -\sin(r(p_z)) & 0 \\ \sin(r(p_z)) & \cos(r(p_z)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

Vortex





$$r(z) = \begin{cases} 0 & z \le z_0 \\ \frac{z-z_0}{z_1-z_0}\theta_{\max} & z_0 \le z \le z_1 \\ \theta_{\max} & z_1 \le z_0 \end{cases}$$

$$\alpha(p) = r(p_z)e^{-(p_x^2+p_y^2)}$$

$$p' = \begin{bmatrix} \cos(\alpha(p)) & -\sin(\alpha(p)) & 0 \\ \sin(\alpha(p)) & \cos(\alpha(p)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

Combinaison

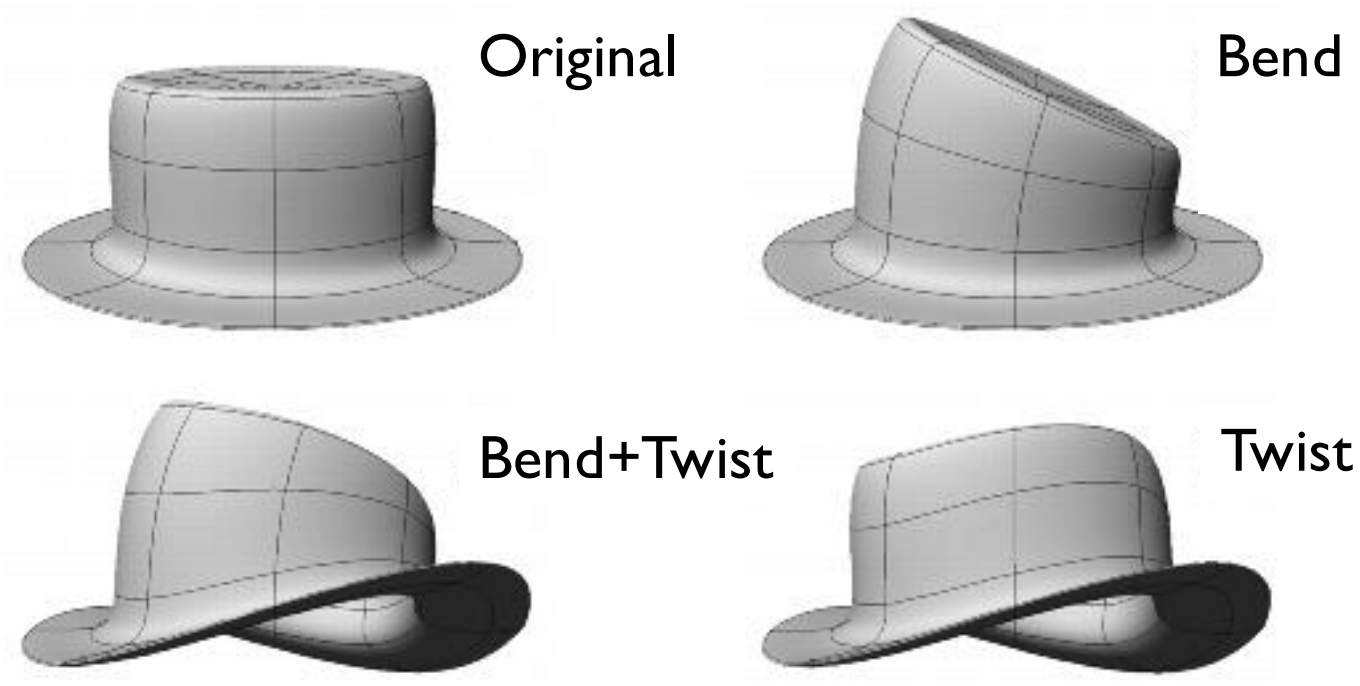
Avantage :

✓ Simple

Inconvénients :

✗ pas de contrôle fin des déformations

✗ le modèle peut s'auto-intersecter



Free-Form Deformations (FFD)

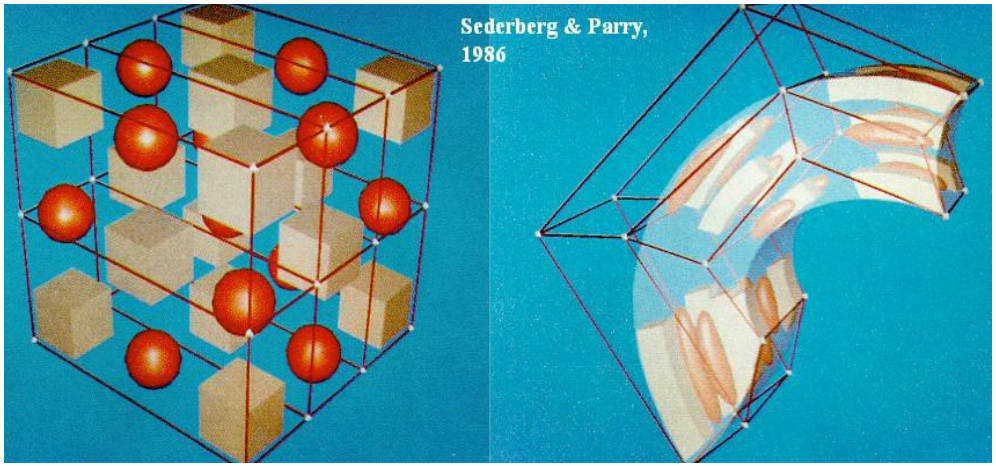
Treillis de points de contrôle (maillage régulier)

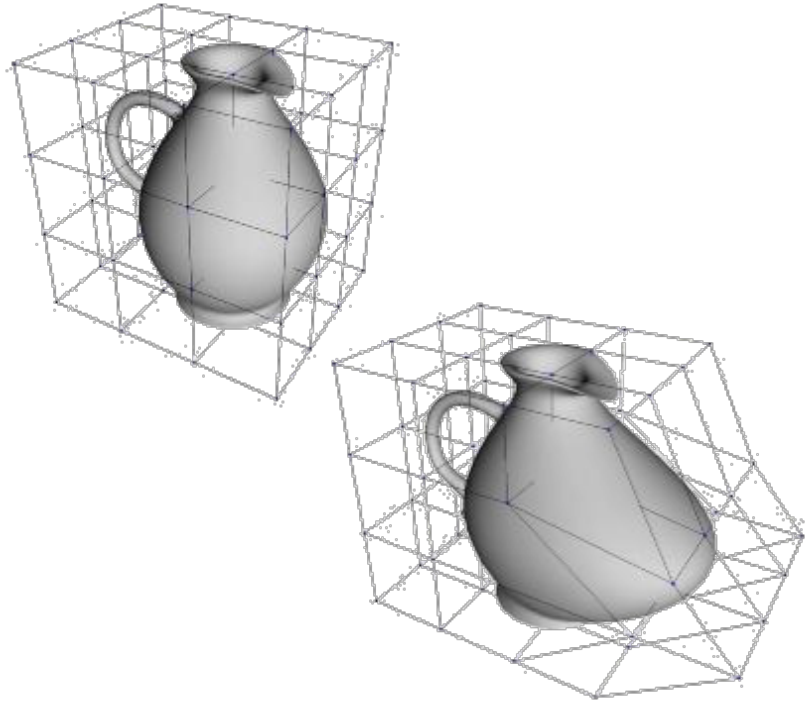
le modèle est plongé dans du « caoutchouc » 3D

Déformer le treillis

$\Longrightarrow$ l'espace « suit » le maillage

$\Longrightarrow$ interpolation dans la grille par Bézier (ou autre)





Free-Form Deformations (FFD)

Grille :

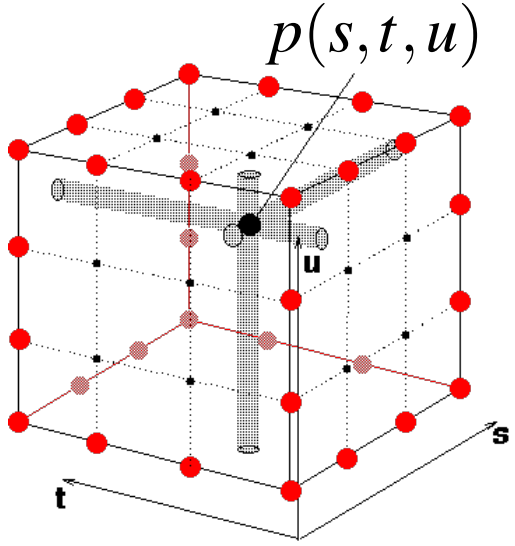
- Origine = q, axes orthogonaux = $(\mathbf{u},\mathbf{v},\mathbf{w})$, #cellules = l,m,n
- Points de la grille :

$$g_{ijk} = q + \frac{i}{l}\mathbf{u} + \frac{j}{m}\mathbf{v} + \frac{k}{n}\mathbf{w}$$

Point p à déformer :

$$p(s,t,u) = q + s\mathbf{u} + t\mathbf{v} + u\mathbf{w}$$

avec :
$$\begin{aligned} s &= (p-q)\cdot\mathbf{u}/\|\mathbf{u}\| \\ t &= (p-q)\cdot\mathbf{v}/\|\mathbf{v}\| \\ u &= (p-q)\cdot\mathbf{w}/\|\mathbf{w}\| \end{aligned}$$



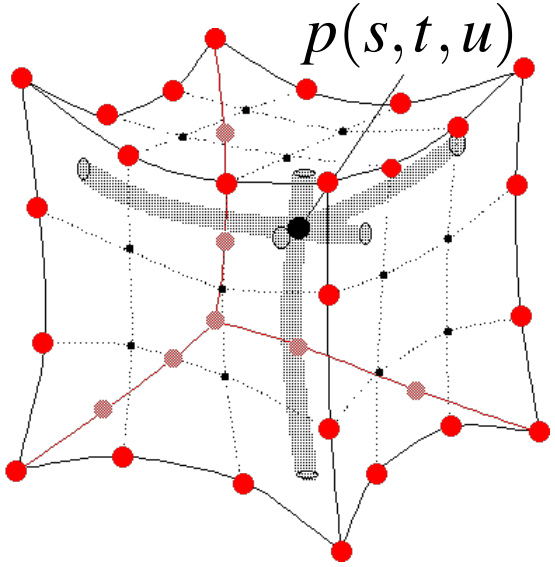
Free-Form Deformations (FFD)

Interpolation

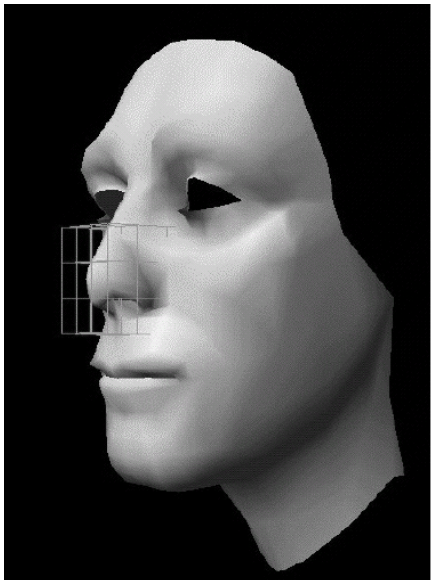
- Points de la grille déplacés en g'_{ijk}
- Interpolation dans la grille par Bézier (ou autre) :

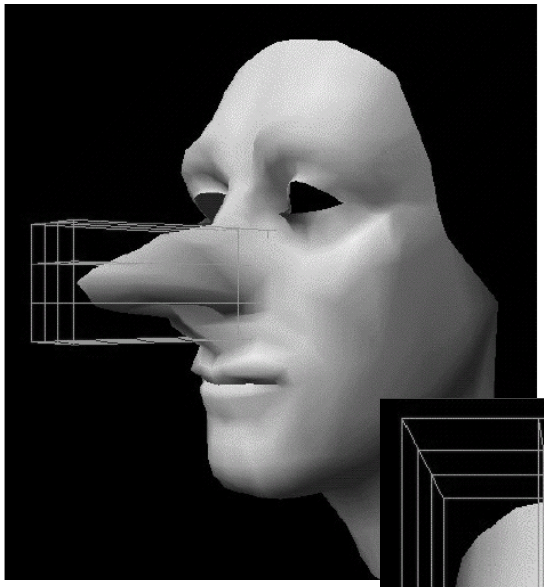
$$p'(s,t,u) = \sum_{i=0}^{l} \sum_{j=0}^{m} \sum_{k=0}^{n} g'_{ijk} B_i^l(s) B_j^m(t) B_k^n(u)$$

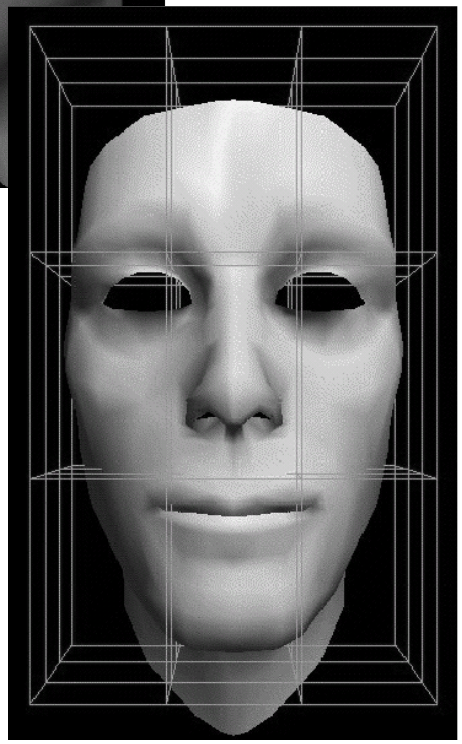
avec $B_i^l(s) = \begin{pmatrix} l \\ i \end{pmatrix} s^i (1-s)^{l-i}$

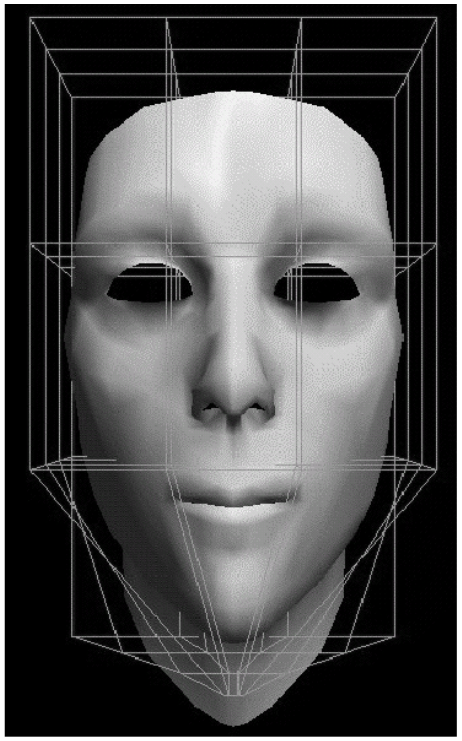


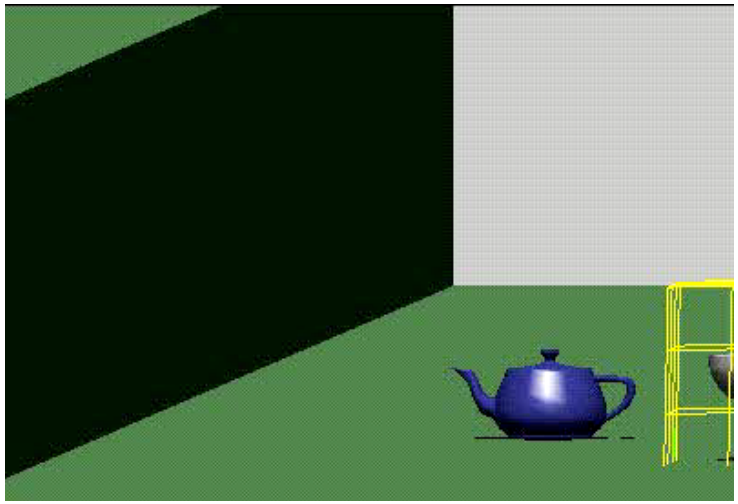
Free-Form Deformations (FFD)











Blend shapes

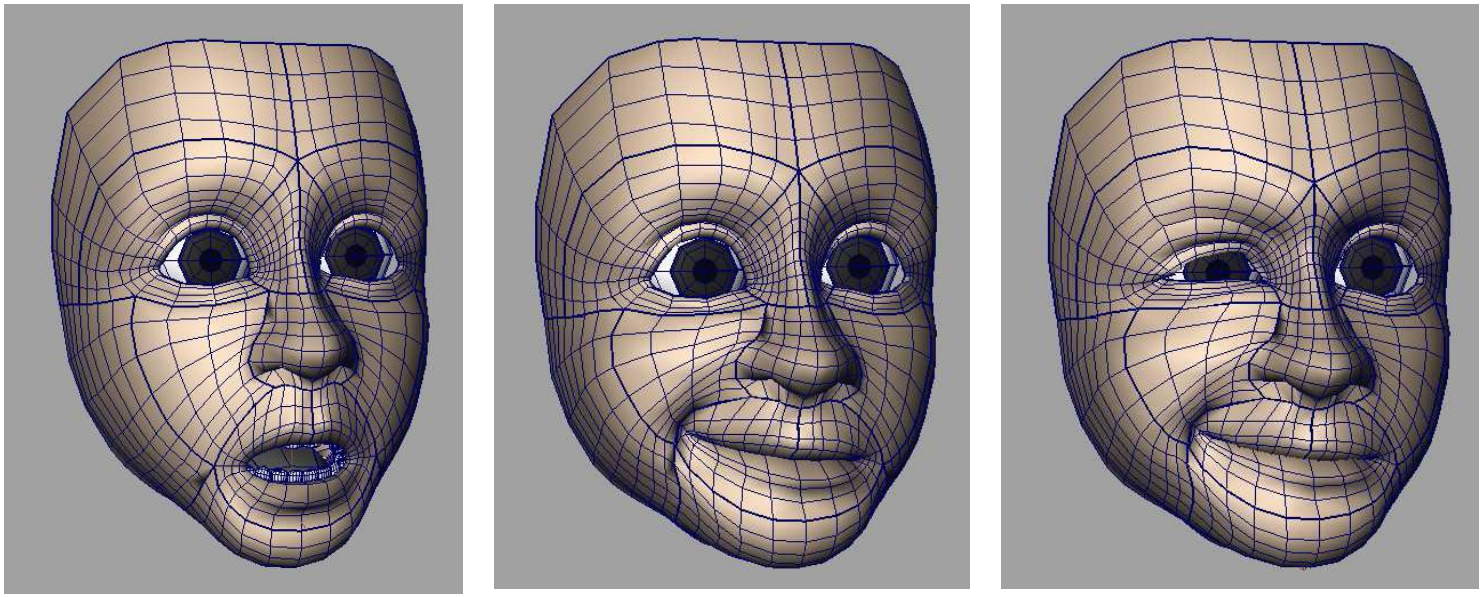
Interpolation de **formes clés**

- Interpolation linéaire de la position des sommets

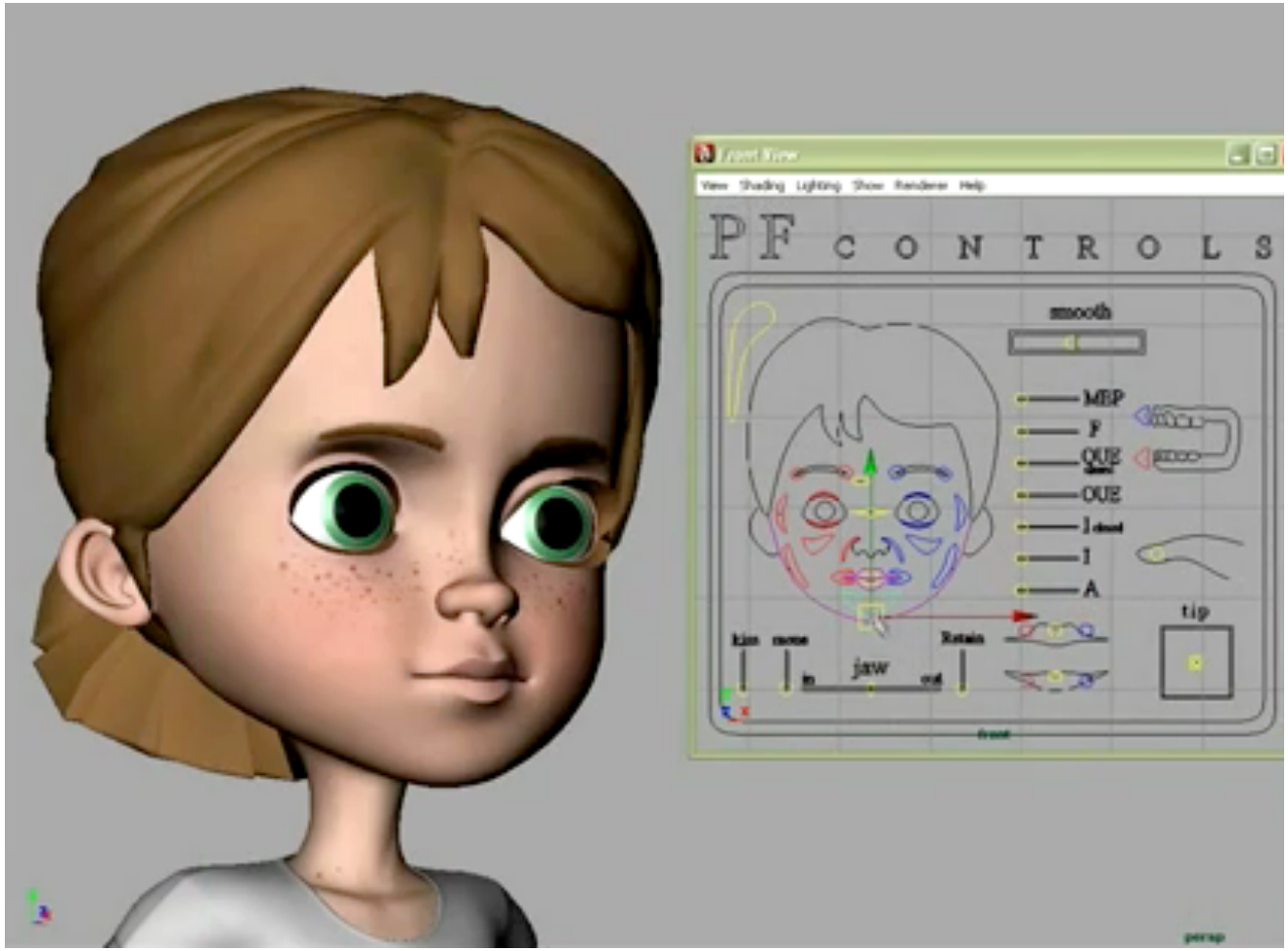
$$p' = p_0 + \Sigma_i w_i (p_i - p_0)$$

- Contrôle de l'animation par les poids w_i

Surtout pour l'**animation faciale**



Blend Shapes (198 expressions)



Remy Terreaux