

Animation et déformation

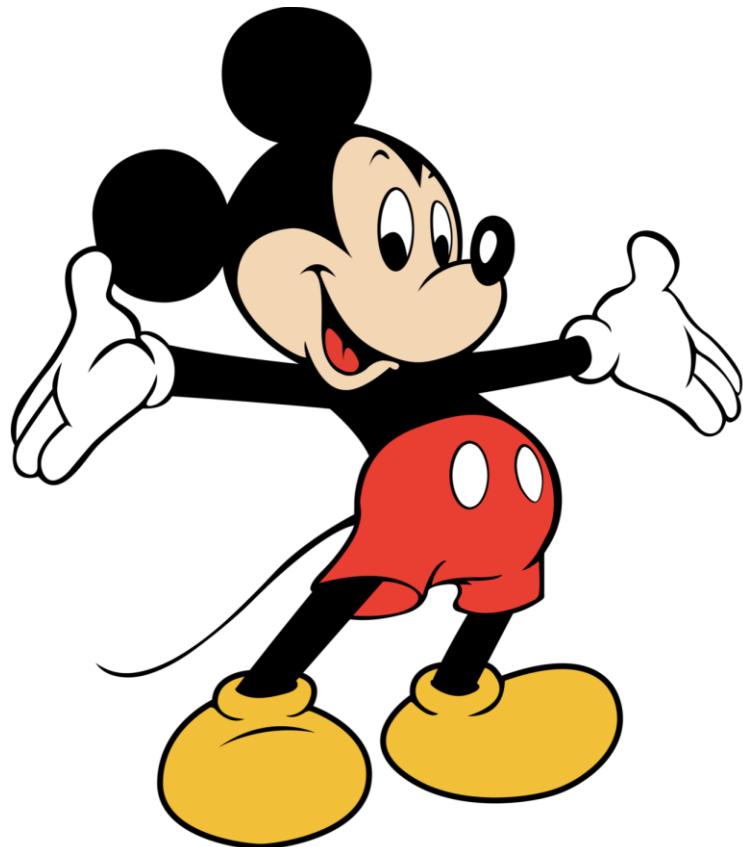
Master IMAGINA

Plan

- Introduction
- Animation
 - Images clés
 - Squelette
 - Skinning
- Deformation
 - Surface
 - Volume

Double héritage

Dessin animé



Robotique



Dessin animé

- Animation

Gertie le dinosaure (1914)



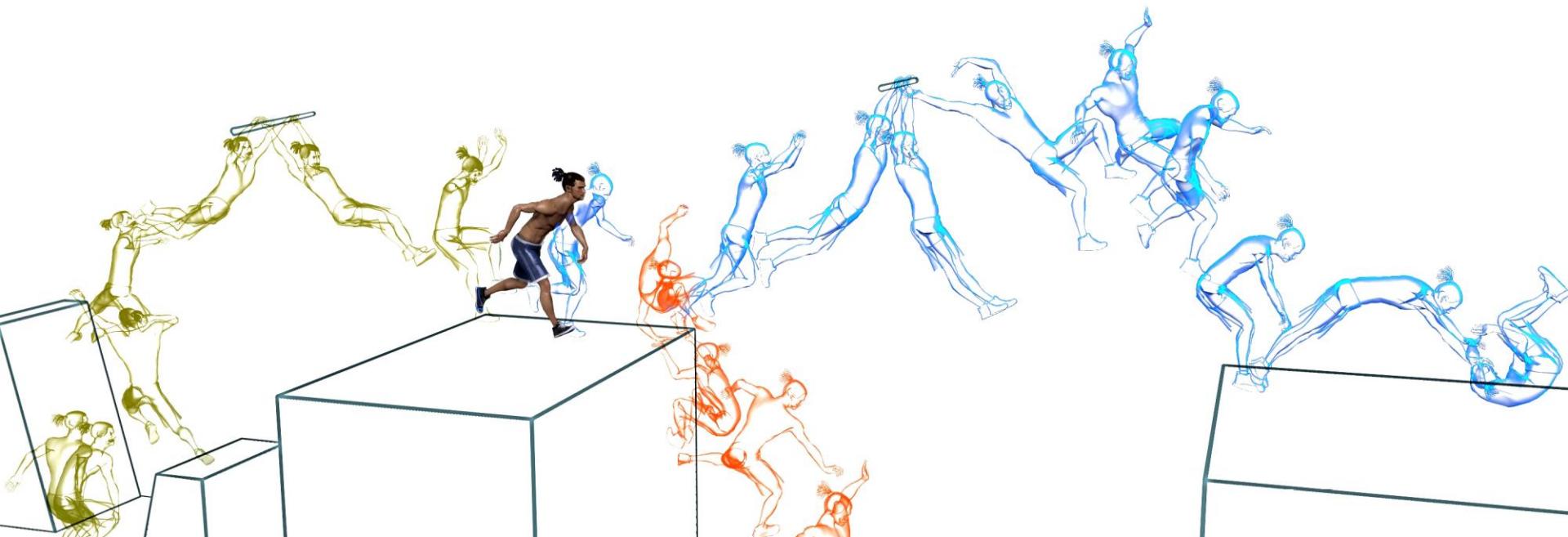
1^{ère} animation par image clés



- Données par un animateur et dessin des images intermédiaires par des « petites mains »

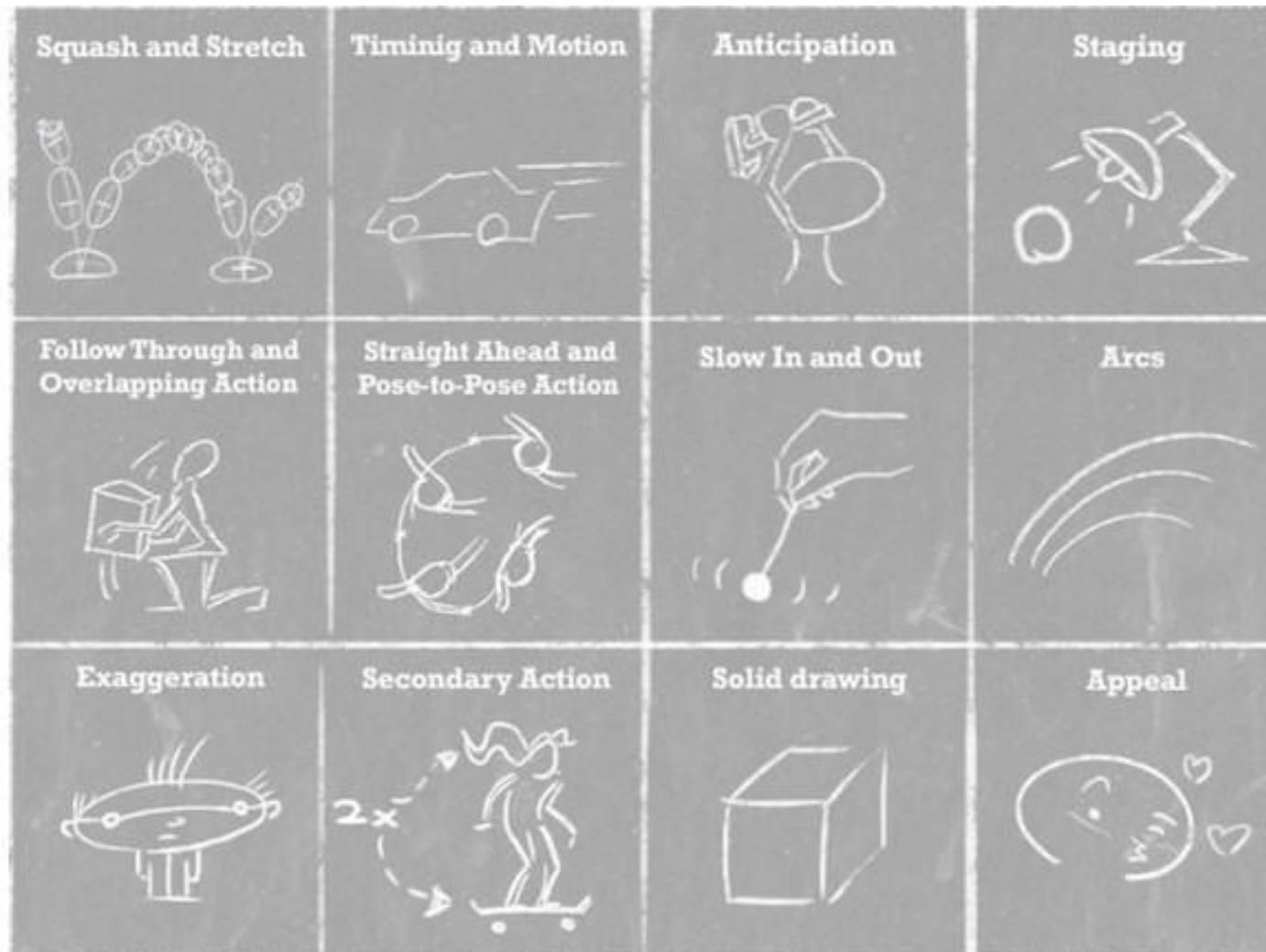
Animation par ordinateur

- Automatiser ce processus



12 principes d'animation

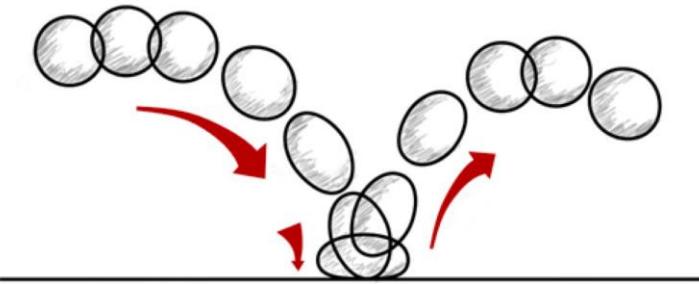
Disney Studios, 1930s



[Voir vidéo...](#)

1. Squash and stretch

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

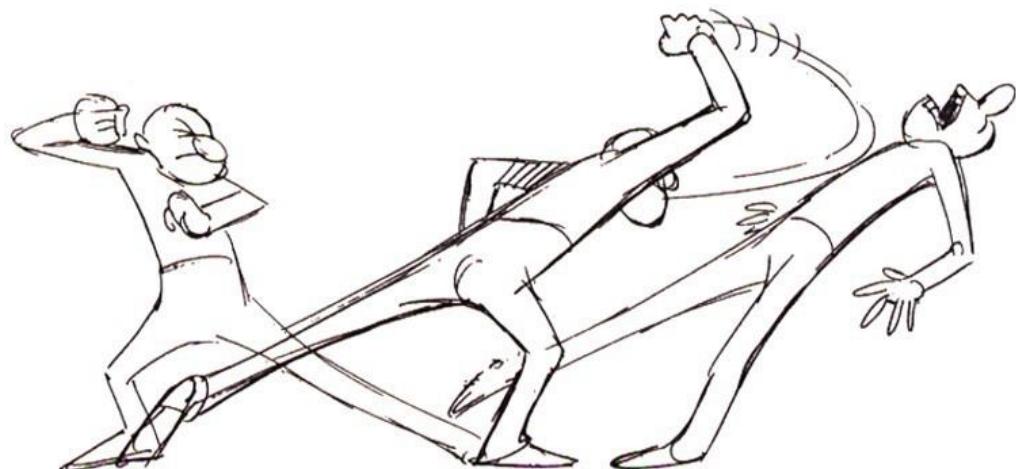


© Richard William

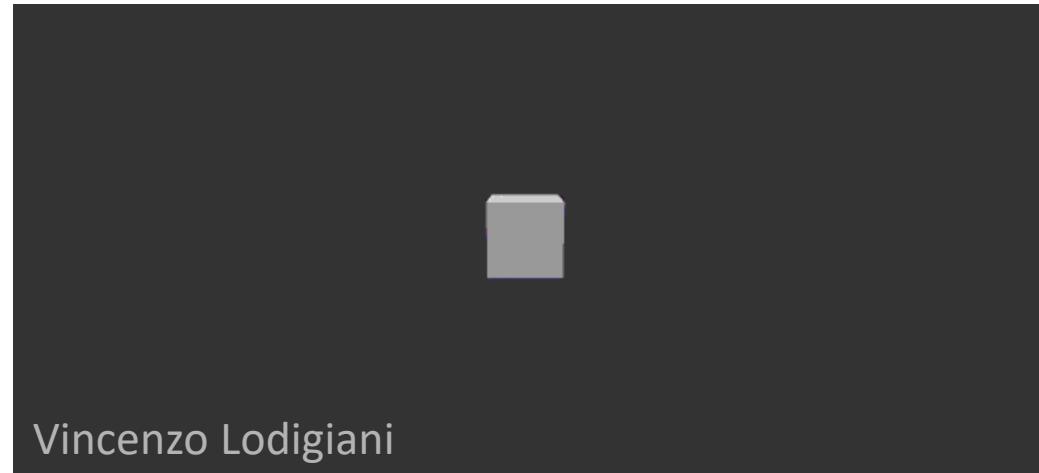


4. Anticipation

- Action inverse au mouvement pour l'accentuer



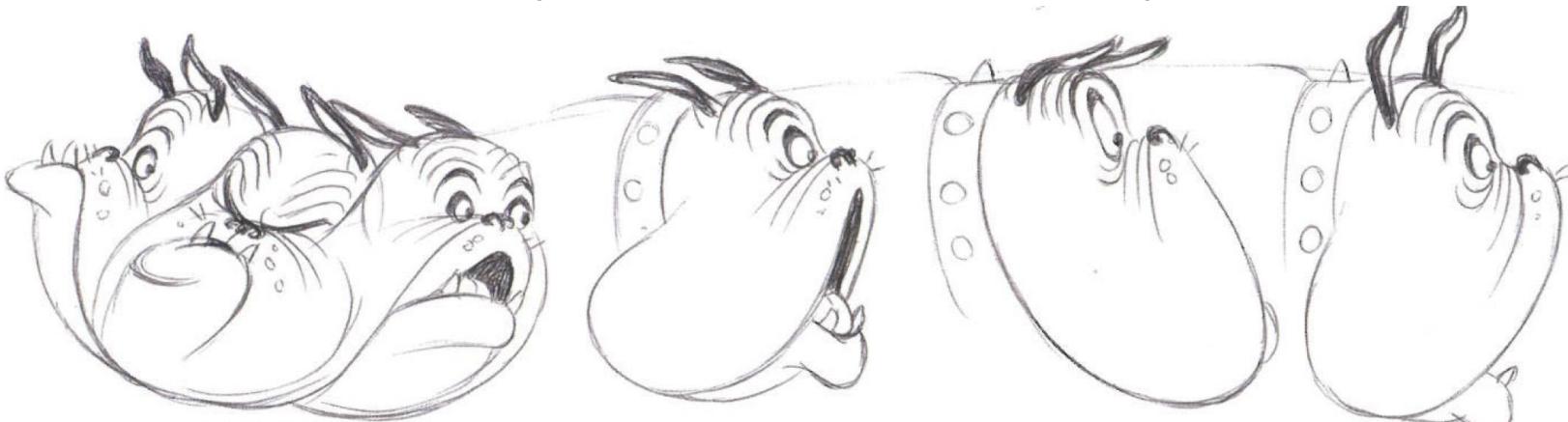
© Richard William



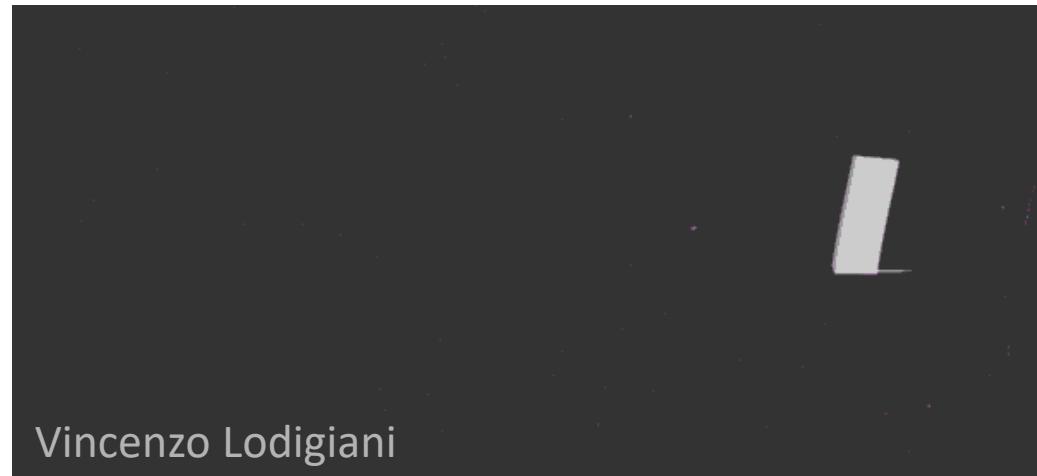
Vincenzo Lodigiani

5. Follow through & Overlapping

- Retards et anticipations de certaines parties de la forme



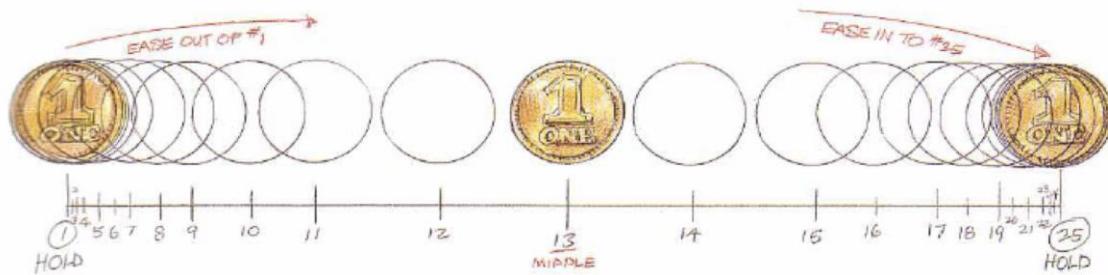
© Richard William



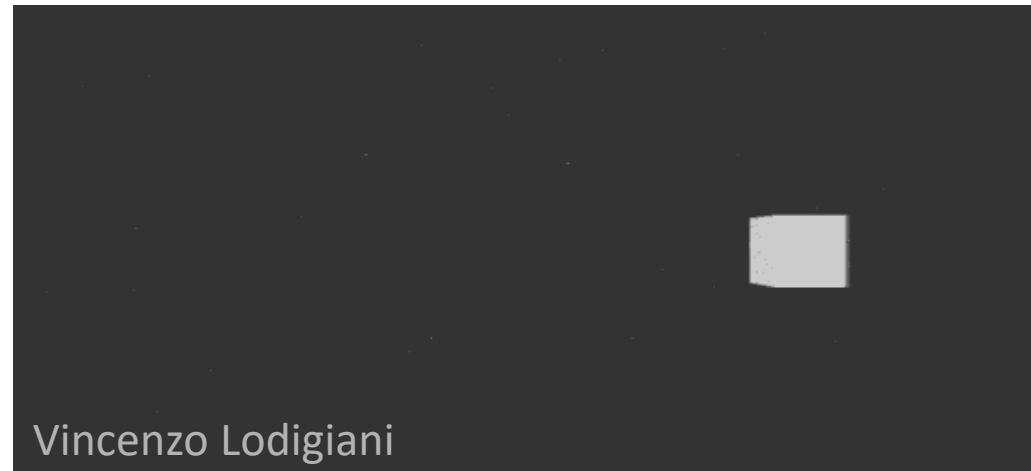
Vincenzo Lodigiani

6. Slow in, slow out

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



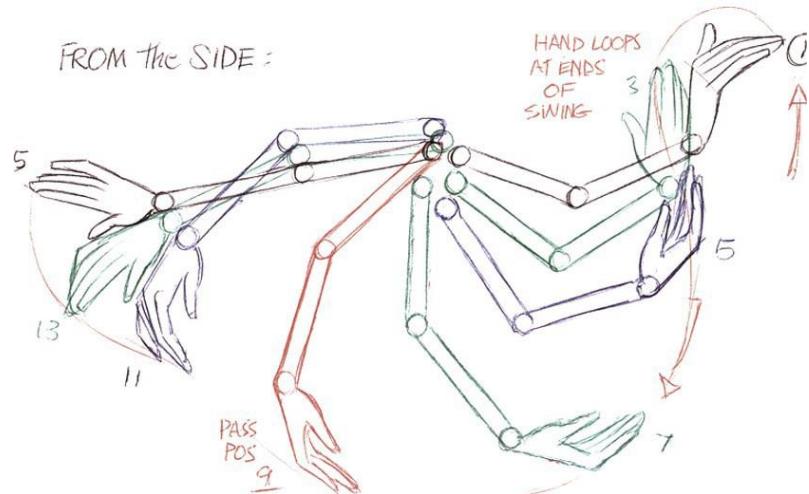
© Richard William



Vincenzo Lodigiani

7. Arcs

- Rien ne bouge en ligne droite mais selon un arc



© Richard William



Vincenzo Lodigiani

Débuts de l'animation 3D

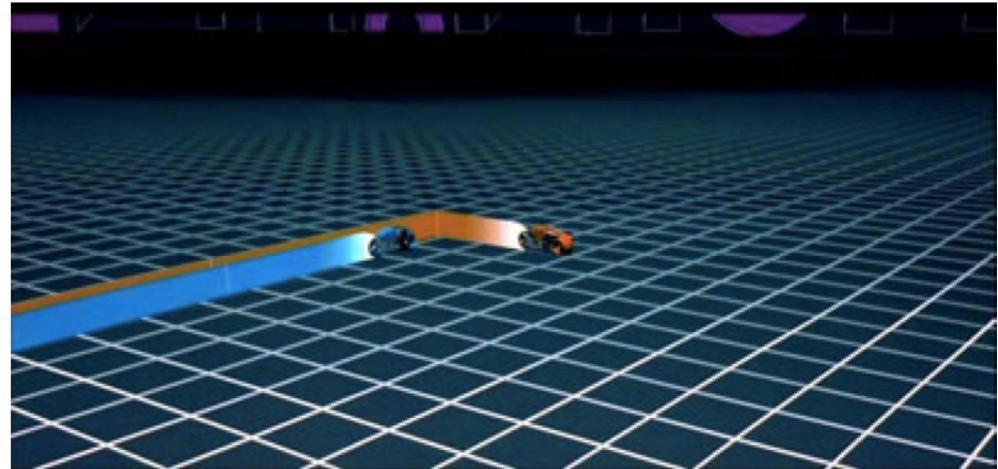
- Recherche académique



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

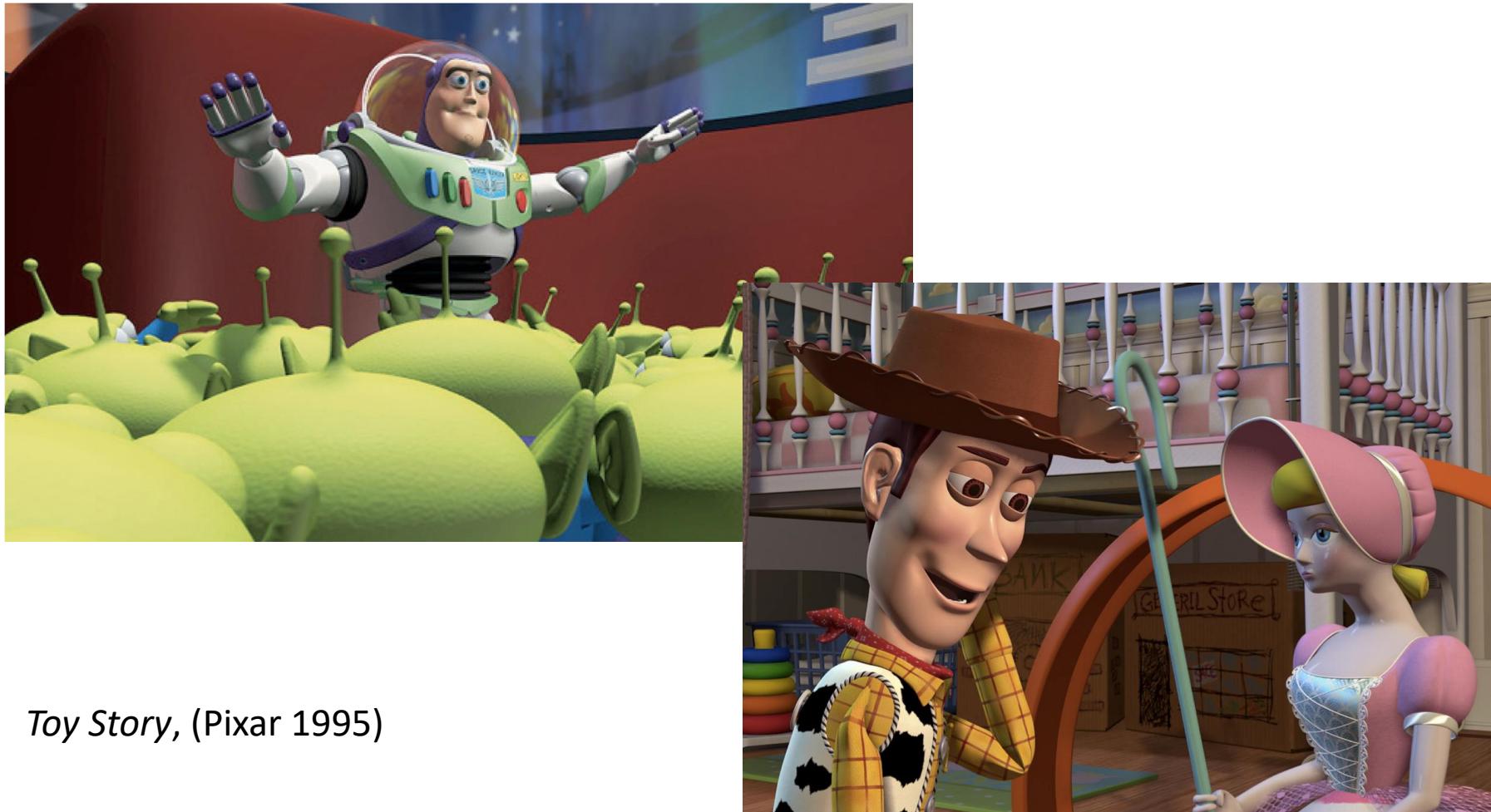
- Court métrage d'animation 3D



Luxo Jr. (Pixar 1986)

Débuts de l'animation 3D

- Premier long métrage d'animation 3D



Animation par ordinateur

- Les modèles ont des paramètres :
 - Position de sommets
 - Normales
 - Points de contrôles de courbes
 - Angle de jointures
 - Couleurs,
 - Lumières...
- n paramètres définissent un espace d'état de n dimension
- Un jeu de valeurs de ces n paramètres = point dans cet espace

Animation par ordinateur

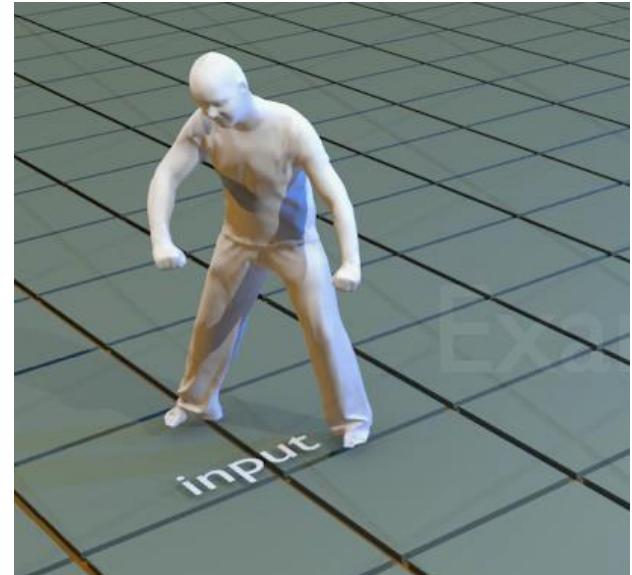
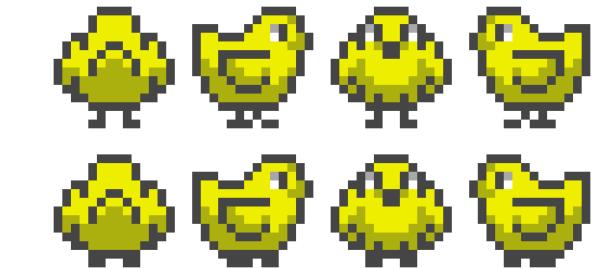
- Une animation est définie par un chemin dans cet espace d'état
- Pour produire une animation :
 1. Commencer au début du chemin
 2. Fixer les paramètres de votre modèle
 3. Rendre une image
 4. Bouger au point suivant du chemin dans l'espace d'état
 5. Aller à 2
- Le chemin est défini par un ensemble de courbe de mouvement (une pour chaque parameter)
- Animation = spécifier la trajectoire dans l'espace d'état

Techniques d'animation

- Traditionnelle (image par image)
- Par image clés (keyframing)
- Techniques procédurales
- Techniques comportementales (e.g., flocking)
- Basées sur la performance (motion capture)
- Basées sur la physique (dynamique)

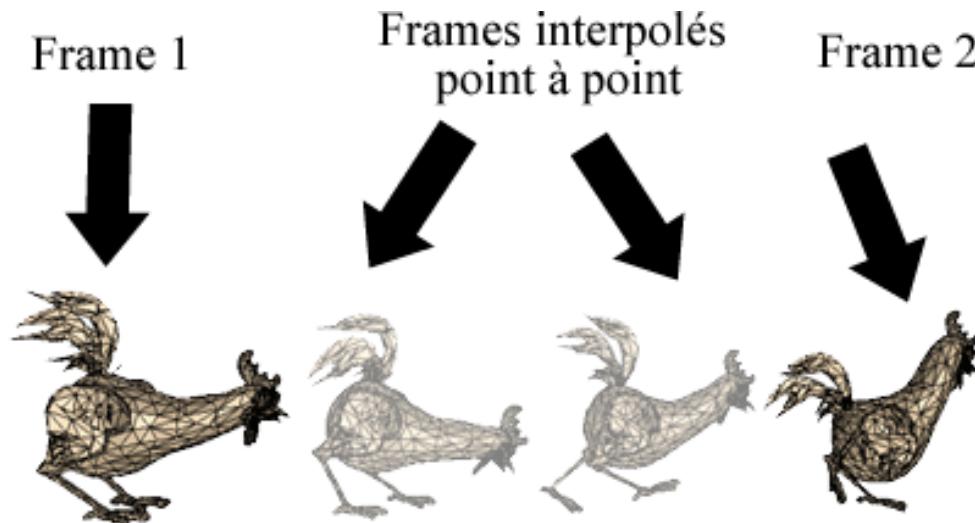
Animation classique

- Chaque image de notre animation est stockée dans une unité complète, séparée du reste.
- En 2D on parle de “Sprite”
 - Images 2D affichées en séquence.
- En 3D on parle d'une suite de maillages (mesh)
 - Suite de modèles 3D
 - rendus en séquence.



Animation par images clés

- Variation pour l'animation par **key-frame** classique en 3D :
 - Le modèle contient un nombre de frame réduit pour l'animation.
 - Chaque vertex est interpolé un à un entre les images pour combler les trous

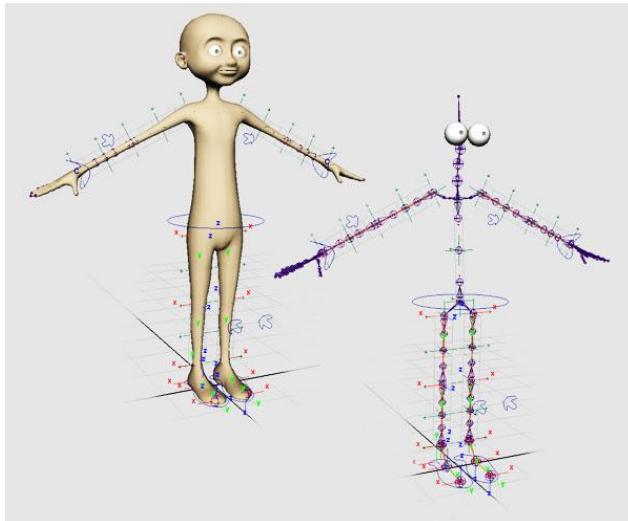


Pourquoi utiliser les squelettes ?

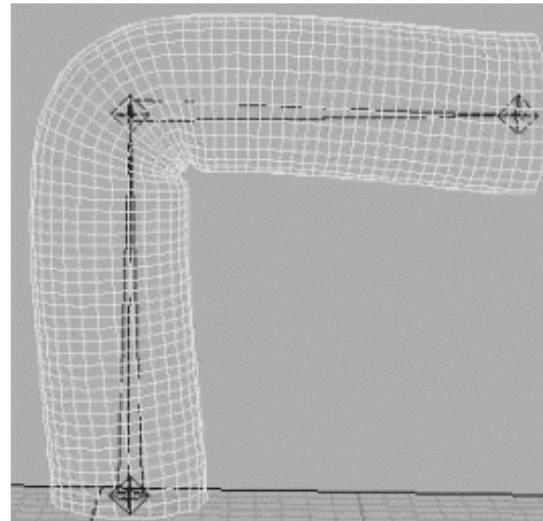
- Beaucoup plus simple à animer.
 - Manipulation d'articulations (sinon chaque vertex indépendamment)
- Beaucoup plus compact que l'animation par keyframe classique.
 - Modèle + squelette stocké 1 fois en mémoire → animation du squelette.
- Assez simple pour permettre l'interaction avec l'environnement
- Possibilité de réutiliser le même squelette pour plusieurs modèles 3D. (Mêmes animations, apparences différentes.)
- Utilisation de méthodes d'interpolation de complexité supérieure à linéaire.
 - Rendu possible puisque l'interpolation se fait sur les articulations, moins nombreuses que les points qui composent le modèle.

Squelette d'animation

- Le modèle 3D contenant la hiérarchie peut être perçu comme une **enveloppe**.
 - Les différents points (vertices) qui composent le modèle sont reliés à différents nœuds de la hiérarchie.
 - L'association d'un point à un élément de la hiérarchie est appelé “**skinning**”



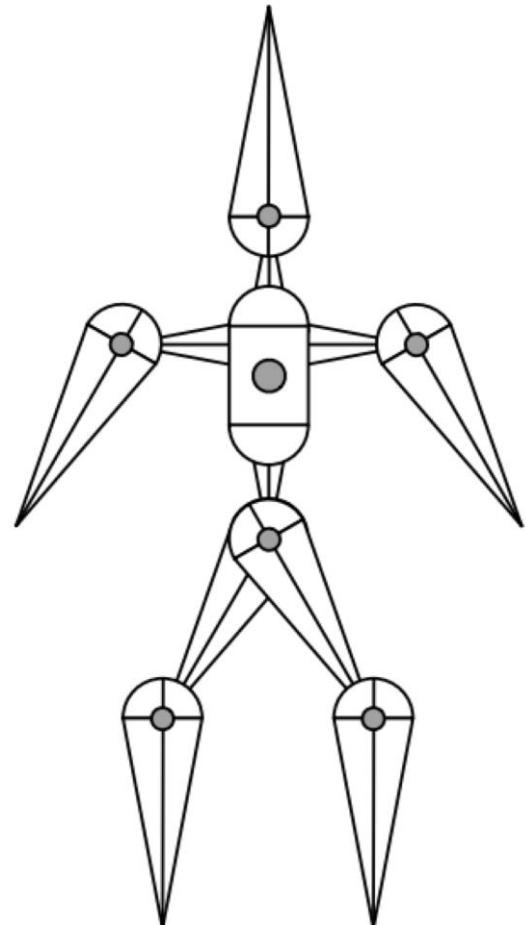
“Lucien”, Mathias Aubry, 2008



David Lanier, 2002

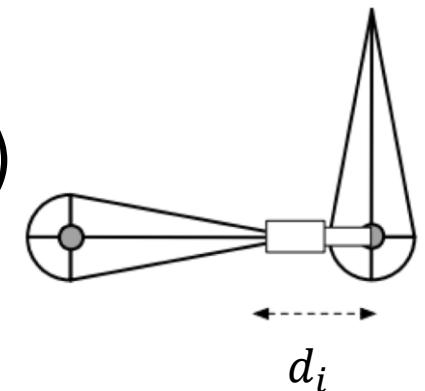
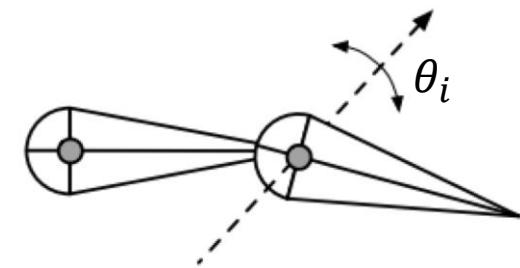
Squelette d'animation

- Modèle décomposé en une **hiérarchie de repères contraints**
- Pas de géométrie, que des solides appelés **os (*bones*)** et des liaisons appelés **articulations (*joins*)**

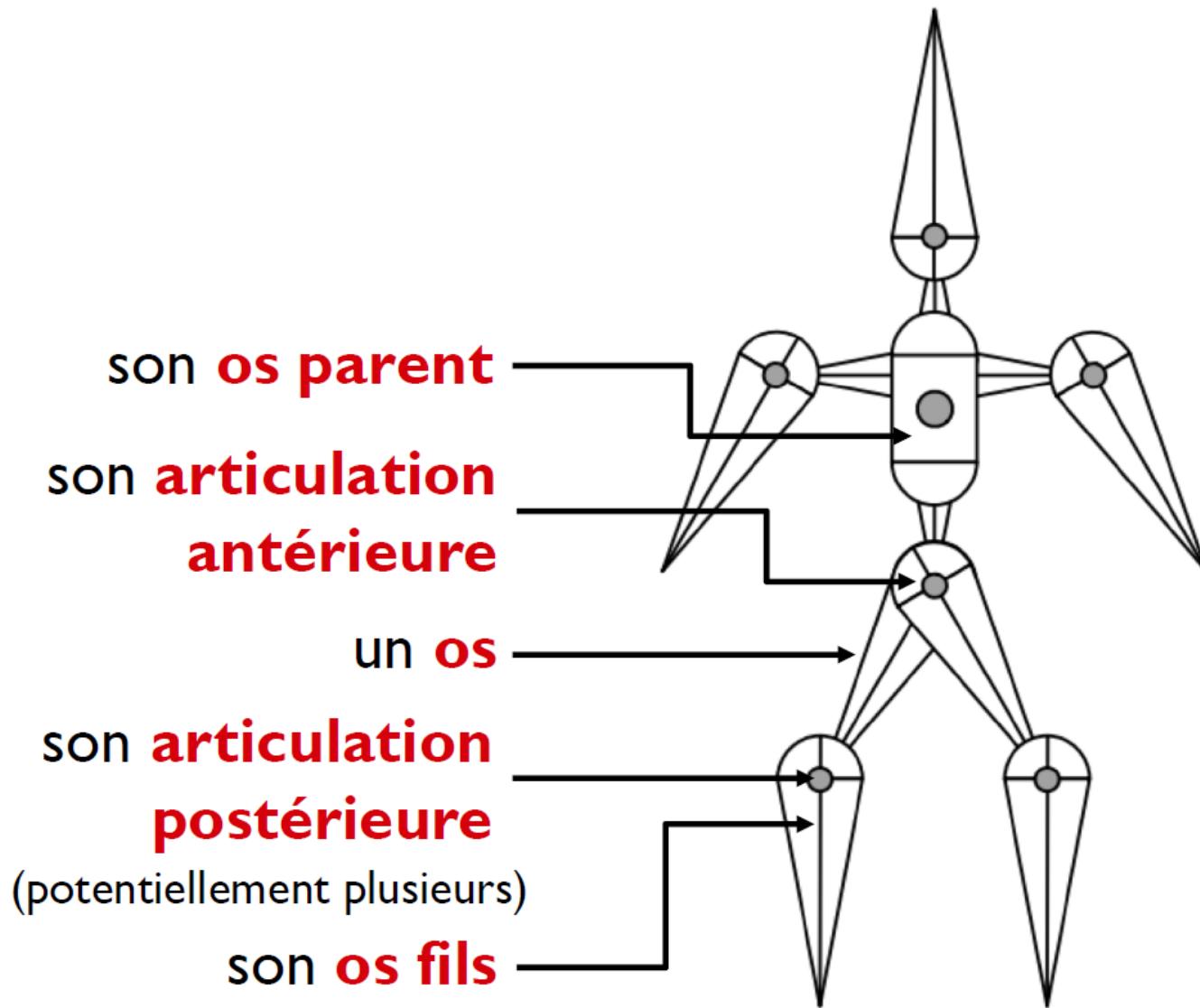


Articulations

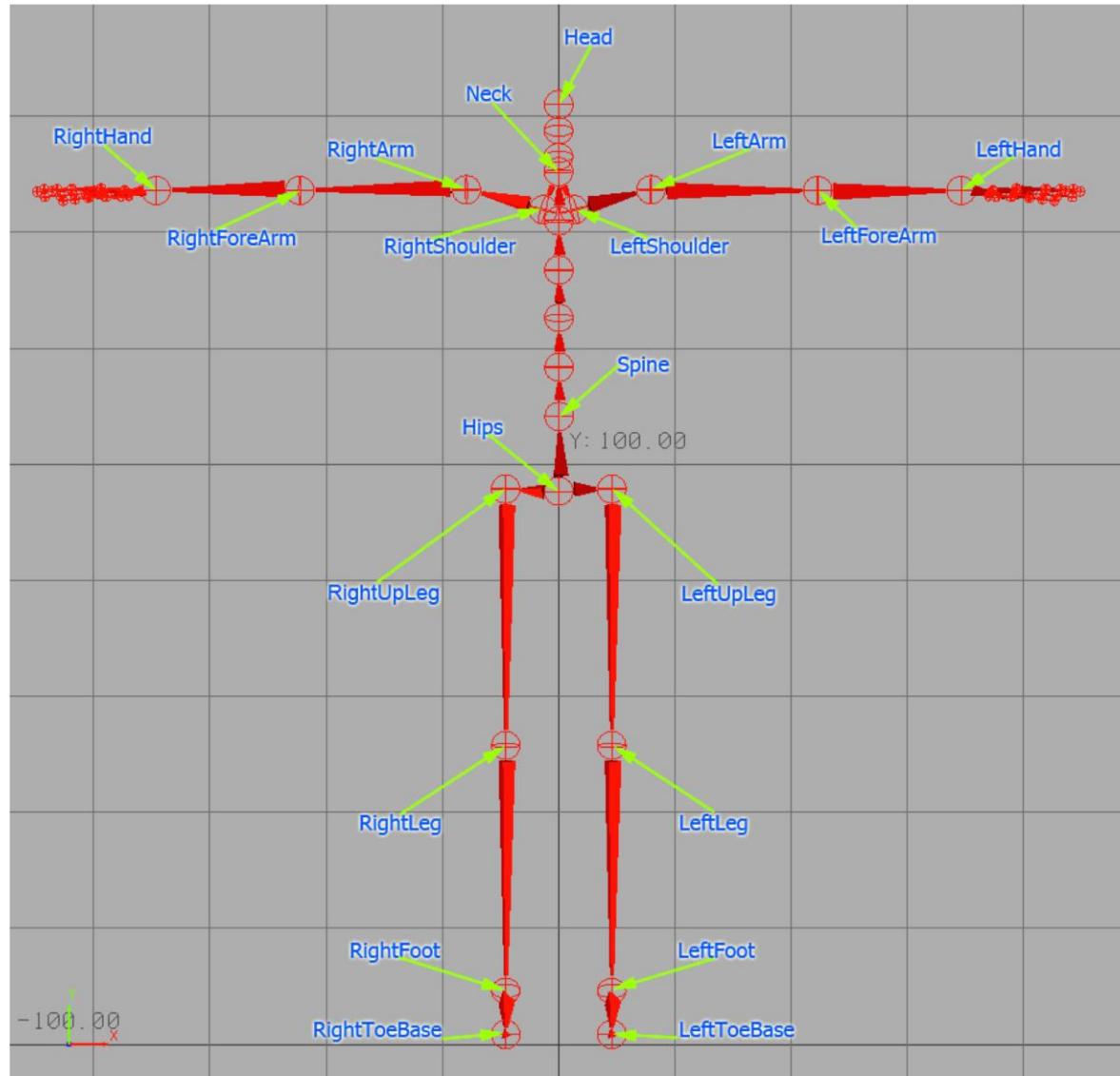
- En général **6 degrés de liberté (DOF)** :
 - 3 translations
 - 3 rotations
- En pratique, souvent des **contraintes** :
- **Pin** – rotation autour d'un axe (1 DOF)
- **Ball** – rotation arbitraire (3 DOF)
- **Prism** – translation selon un axe (1 DOF)



Chaîne articulée

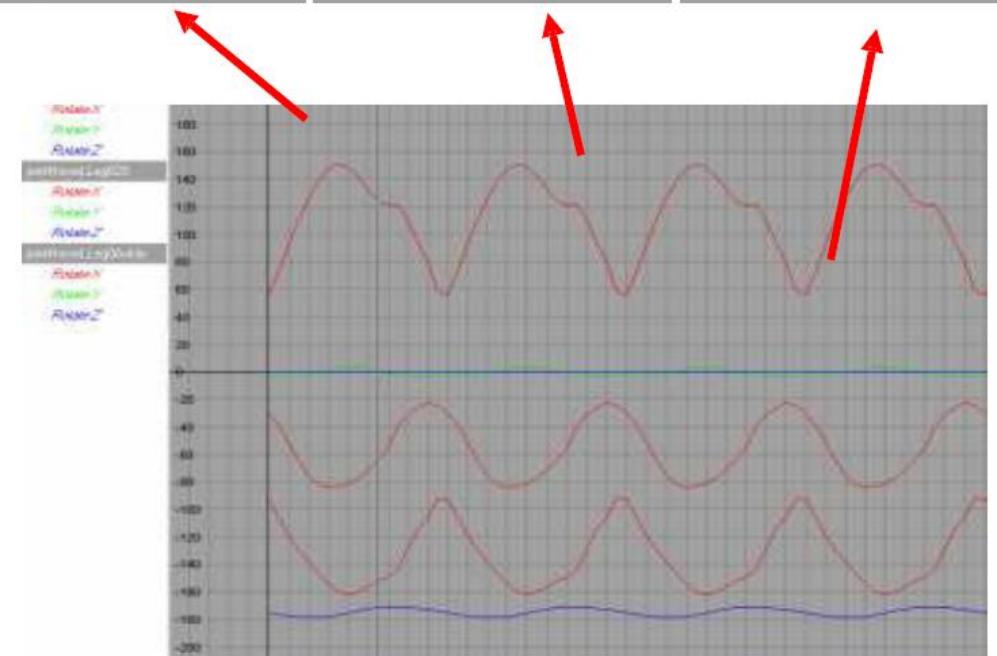
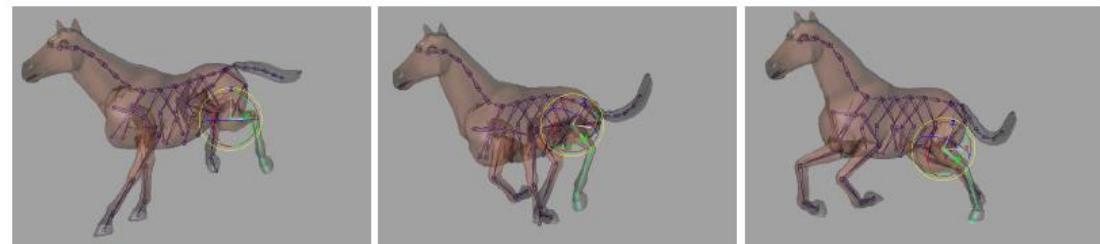
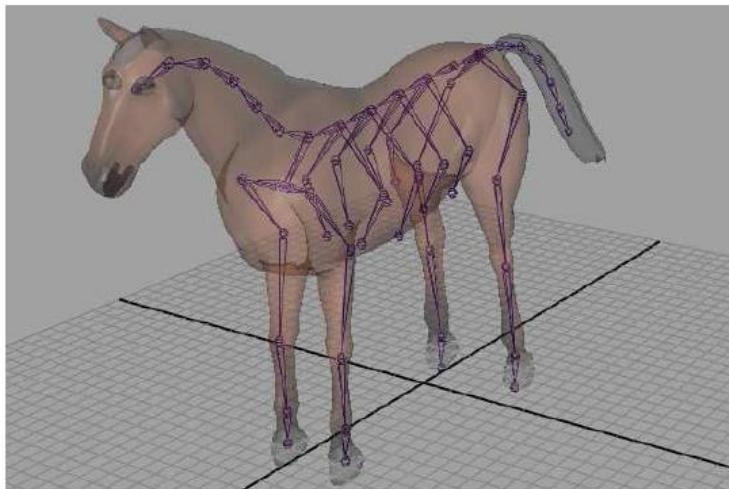


Squelette d'animation

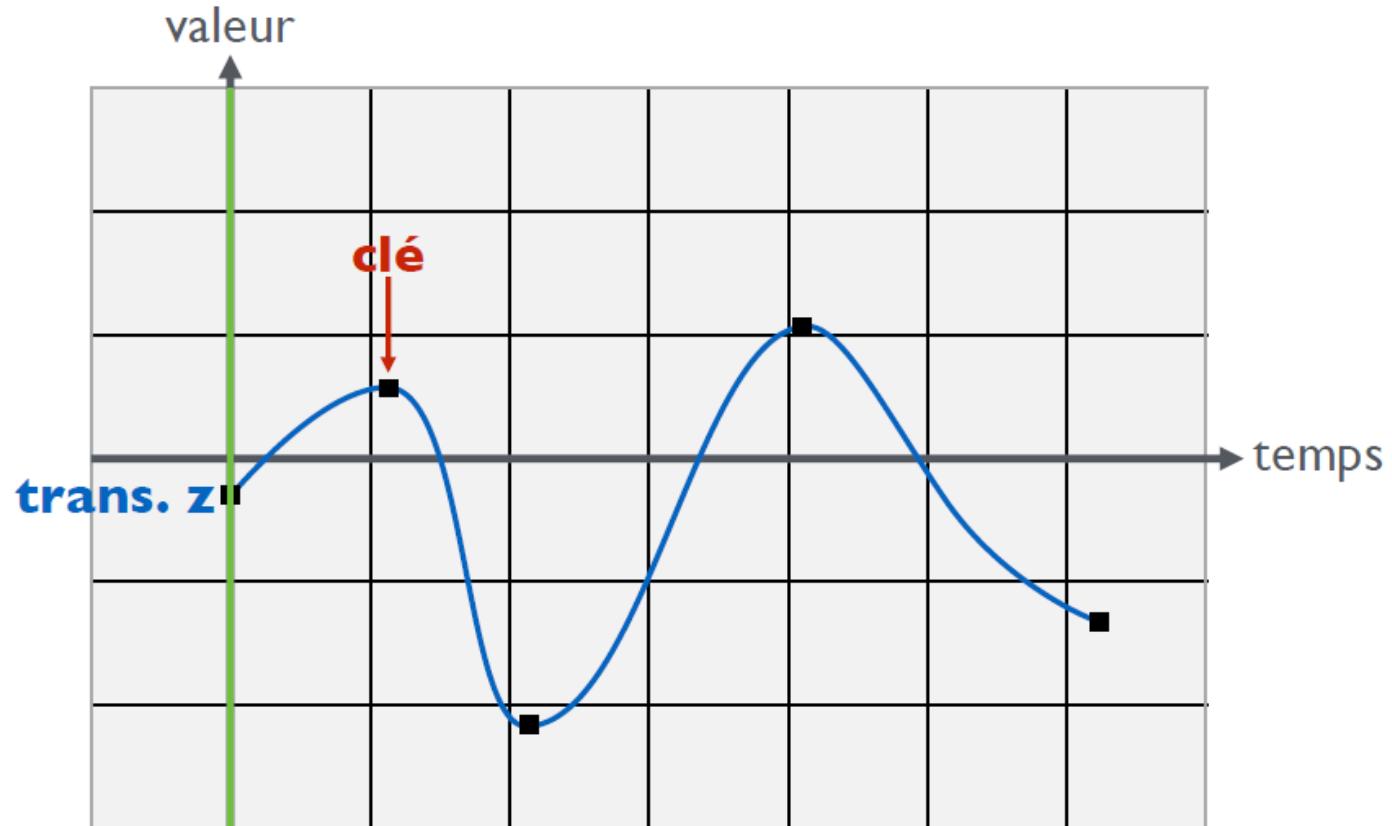


Animation 3D

- **Squelette + interpolation selon une courbe**



Courbe d'animation

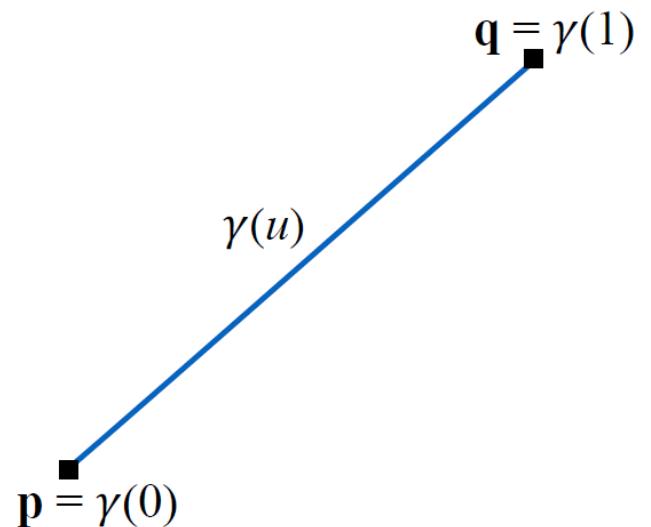


Interpolation

- **Entrées** : valeurs clés (positions)
- **Objectif** : interpoler ces valeurs clés par une fonction continue \Rightarrow courbes d'animation
- **Fonction linéaire** :

$$\gamma(u) = (1-u) \mathbf{p} + u \mathbf{q}$$

Continuité C^0
(position)

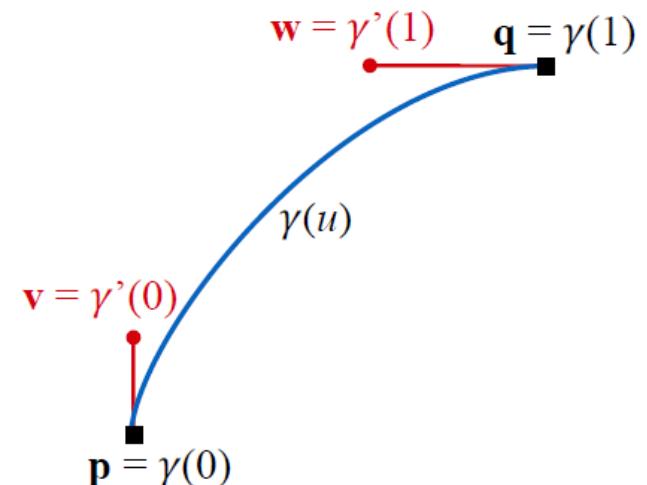


Interpolation

- **Entrées** : valeurs clés (positions et vitesses)
- **Objectif** : interpoler ces valeurs clés par une fonction continue
⇒ courbes d'animation
- **Polynôme de degré 3** de la forme :

$$\gamma(u) = \sum_{n=0 \dots 3} a_n u^n$$

Continuité C^1 entre deux clés
(positions et tangentes)



- Déterminer les paramètres en interpolant (ou approximant) au mieux les valeurs clés
- Degré 3 ⇒ 4 contraintes suffisantes pour résolution exacte

Interpolation

- Polynômes d'Hermitte
- Splines de Catmull-Rom
- Interpolation de Bézier cubique
- Comme vu aux cours précédents...

Interpolation des rotations

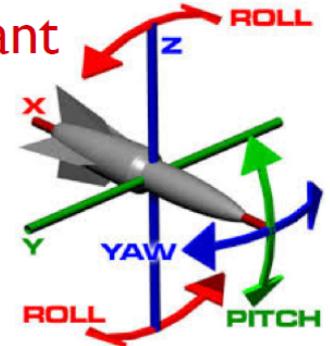
Angles d'Euler

- 1 angle par axe : $(\theta_x, \theta_y, \theta_z)$

- $M = R_{x,\theta_x} R_{y,\theta_y} R_{z,\theta_z} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$

⇒ attention, l'ordre des multiplications est important

- Interpoler chaque angle
⇒ intuitif et peu couteux



Limitation

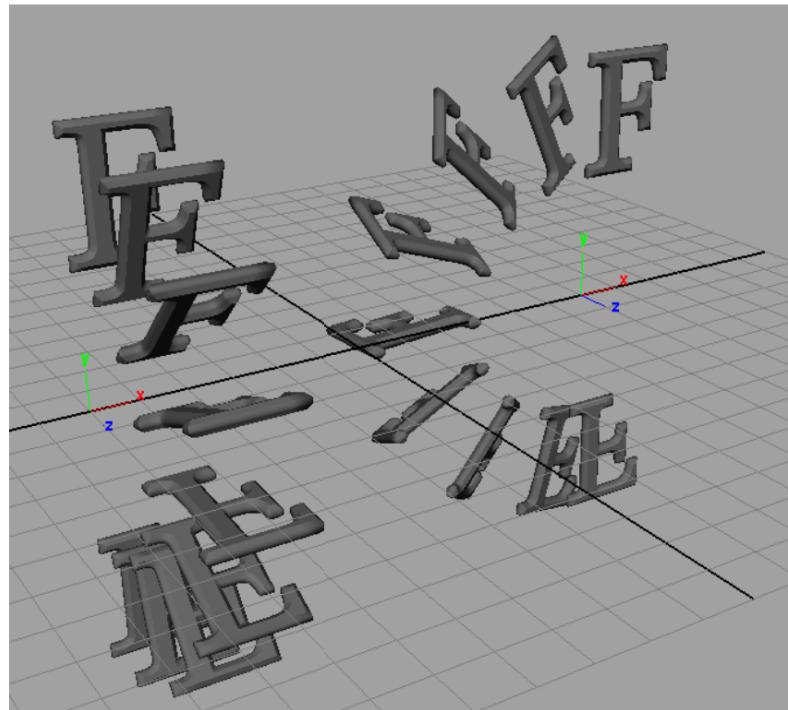
Non-unicité de la trajectoire

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



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

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



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



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

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

Quaternions

Méthode la plus populaire pour représenter une orientation dans le domaine de.

- Composé de 4 nombres, soient 1 angle et 3 nombres qui composent un vecteur.

Se note:

$$q = [s, x, y, z]$$

Ou

$$q = [s, \boldsymbol{v}], \boldsymbol{v} \text{ étant le vecteur } (x, y, z)$$

Quaternions

- Un quaternion est un **4-vecteur**

$$q = (w, x, y, z)$$

- Vu comme un scalaire et un **3-vecteur**

$$q = (s, \mathbf{v}) \quad s = w \quad \mathbf{v} = (x, y, z)$$

- notation alternative : $q = s + \mathbf{v}$

- Quaternions unitaires = **4-vecteurs unitaires** :

$$w^2 + x^2 + y^2 + z^2 = 1, \quad \text{ou} \quad s^2 + \|\mathbf{v}\|^2 = 1$$

- s et $\|\mathbf{v}\|$ vus comme **sin** et **cos** d'un angle : $q = \cos \psi + \hat{\mathbf{v}} \sin \psi$

- Représentation de rotations type angle/axe

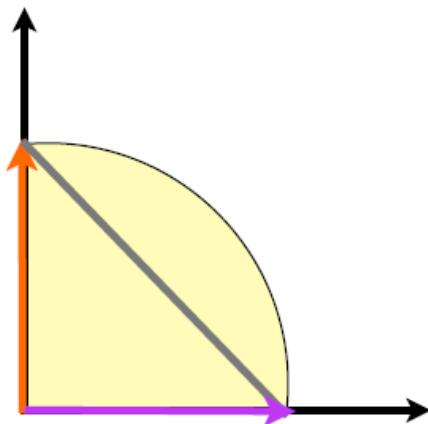
- direction de \mathbf{v} donne l'axe de rotation
 - s et $\|\mathbf{v}\|$ donnent l'angle de rotation

Pourquoi les quaternions ?

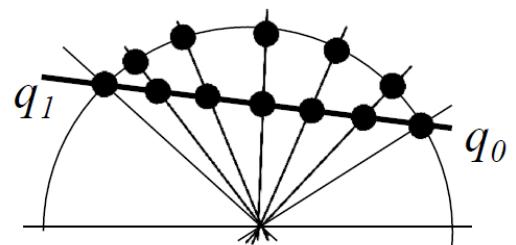
- Rapides, peu d'opérations, non redondants
- Numériquement stables aux changements incrémentaux
- Compositions de rotations faciles
- Se convertit en matrice
- Raison principale : interpolation sphérique

Interpolations entre quaternions

- Pourquoi pas une interpolation linéaire ?
 - Besoin d'être normalisé
 - N'a pas une vitesse constante d'interpolation



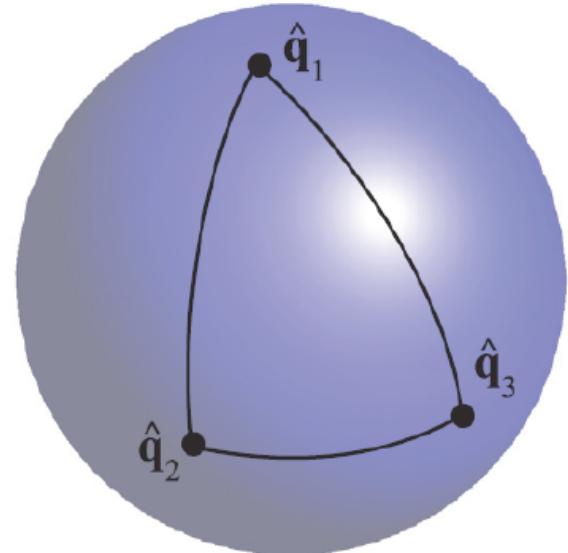
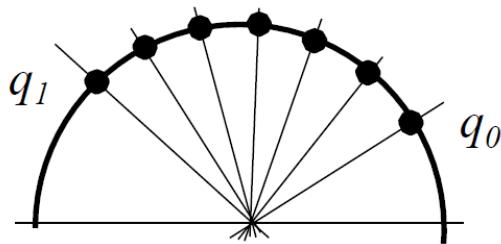
$$\frac{(1 - \alpha)x + \alpha y}{\|(1 - \alpha)x + \alpha y\|}$$



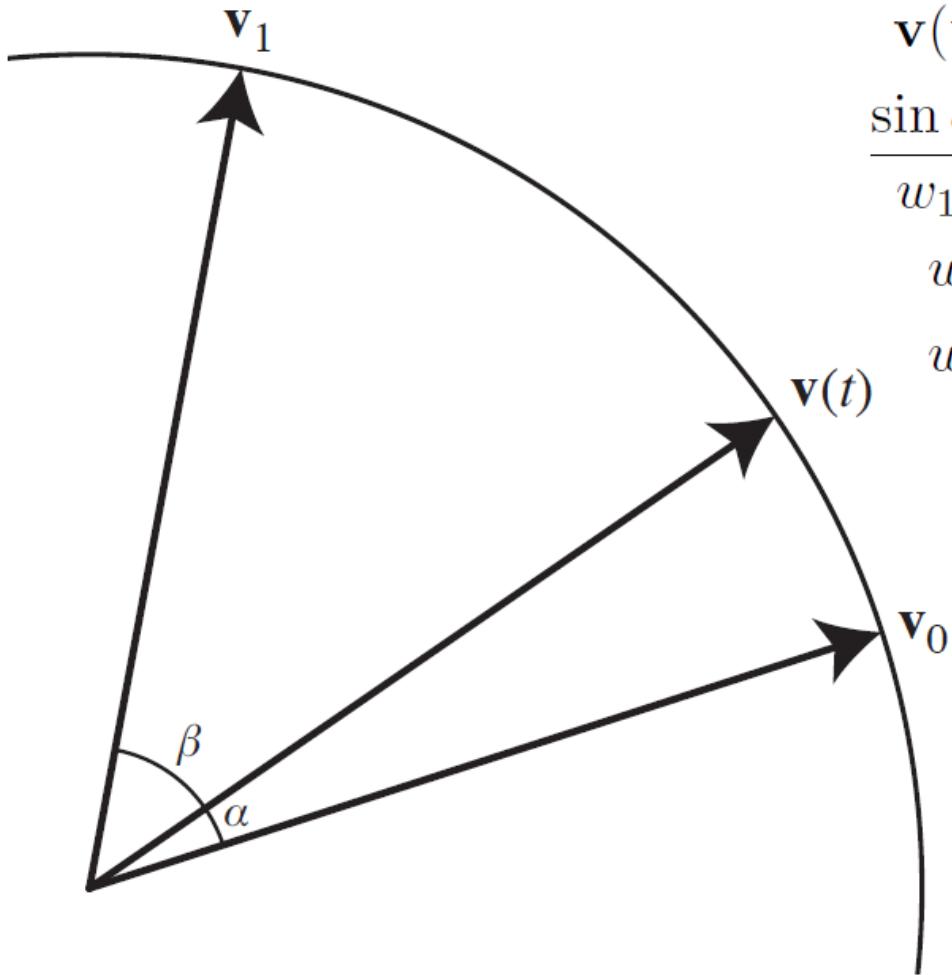
Spherical Linear Interpolation (slerp)

- Interpolation intuitive entre les orientations
 - Bien défini avec les quaternions
 - Très utile pour l'animation
 - Interpolation entre 2 quaternion = chemin le plus court entre 2 point sur une sphère

Geodesic, on Great Circle



Spherical linear interpolation (“slerp”)



$$\alpha + \beta = \psi$$

$$\mathbf{v}(t) = w_0 \mathbf{v}_0 + w_1 \mathbf{v}_1$$

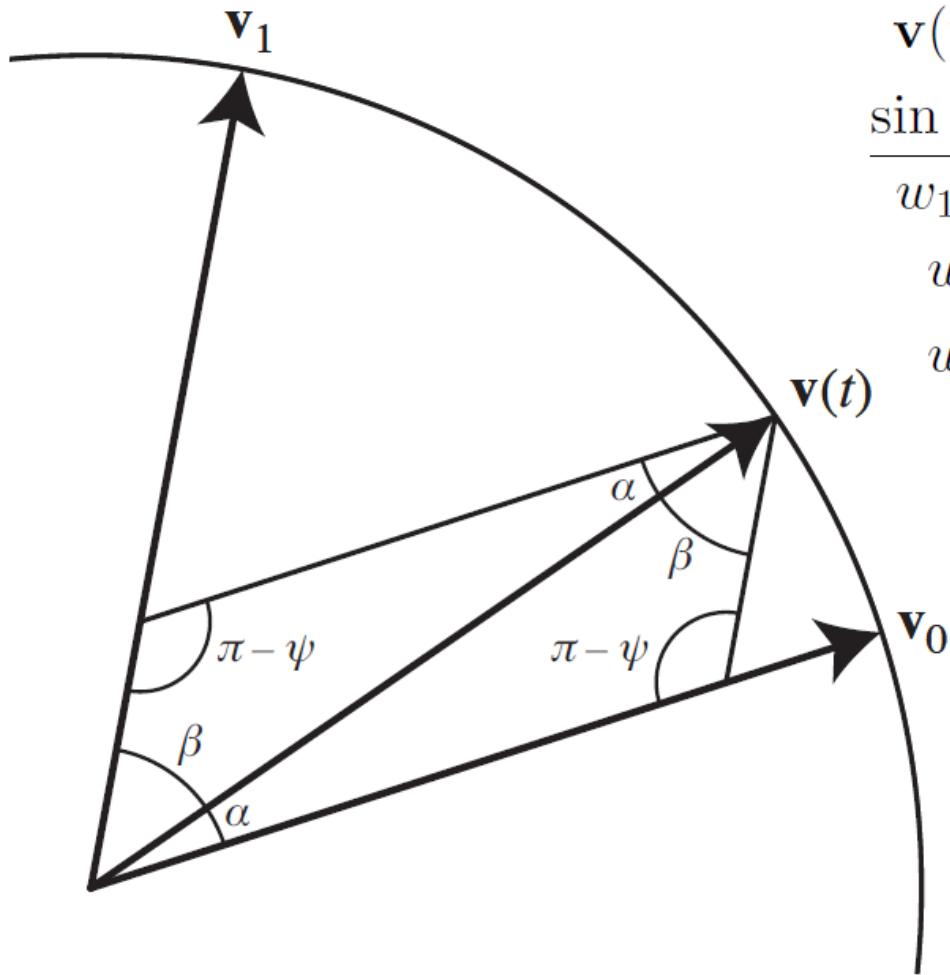
$$\frac{\sin \alpha}{w_1} = \frac{\sin \beta}{w_0} = \frac{\sin(\pi - \psi)}{1} = \sin \psi$$

$$w_0 = \sin \beta / \sin \psi$$

$$w_1 = \sin \alpha / \sin \psi$$

$$\psi = \cos^{-1}(\mathbf{v}_0 \cdot \mathbf{v}_1)$$

Spherical linear interpolation (“slerp”)



$$\alpha + \beta = \psi$$

$$\mathbf{v}(t) = w_0 \mathbf{v}_0 + w_1 \mathbf{v}_1$$

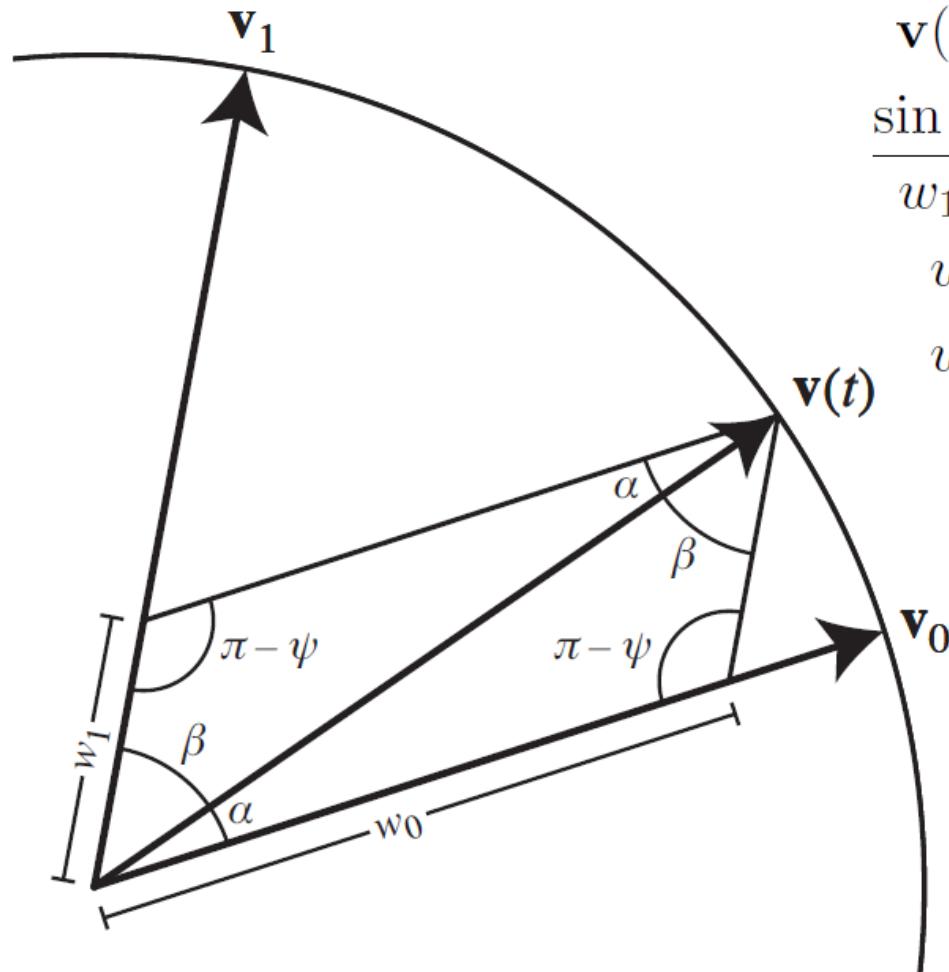
$$\frac{\sin \alpha}{w_1} = \frac{\sin \beta}{w_0} = \frac{\sin(\pi - \psi)}{1} = \sin \psi$$

$$w_0 = \sin \beta / \sin \psi$$

$$w_1 = \sin \alpha / \sin \psi$$

$$\psi = \cos^{-1}(\mathbf{v}_0 \cdot \mathbf{v}_1)$$

Spherical linear interpolation (“slerp”)



$$\alpha + \beta = \psi$$

$$\mathbf{v}(t) = w_0 \mathbf{v}_0 + w_1 \mathbf{v}_1$$

$$\frac{\sin \alpha}{w_1} = \frac{\sin \beta}{w_0} = \frac{\sin(\pi - \psi)}{1} = \sin \psi$$

$$w_0 = \sin \beta / \sin \psi$$

$$w_1 = \sin \alpha / \sin \psi$$

$$\psi = \cos^{-1}(\mathbf{v}_0 \cdot \mathbf{v}_1)$$

Interpolation de quaternions

- Spherical linear interpolation : fonctionne naturellement en toute dimension
- Traverse un arc sur la sphère unité des quaternions
 - Vitesse de rotation angulaire uniforme autour d'un axe fixe

$$\begin{aligned}\psi &= \cos^{-1}(q_0 \cdot q_1) \\ q(t) &= \frac{q_0 \sin(1-t)\psi + q_1 \sin t}{\sin}\end{aligned}$$

Cinématique

Mouvement non-constraint

- Hucher la tête, faire un signe de la main...
⇒ Cinématique directe (FK)

Mouvement constraint

- Attraper un objet, marcher sur le sol...
⇒ Cinématique inverse (IK)

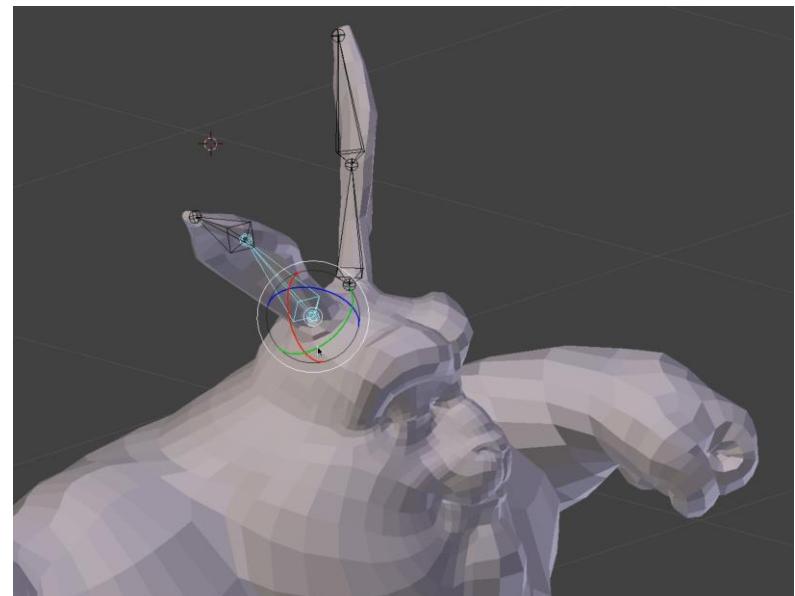
Cinématique directe

Entrées : configuration de la chaîne = valeurs des paramètres des articulations à un temps donnée :

$$\mathbf{q} = (\theta_1, \theta_2, \theta_3, \dots, d_1, d_2, \dots)$$

Sortie : position \mathbf{p} d'une extrémité de la chaîne (coordonnées cartésiennes + orientation, 3 ou 6D)

On calcule : $\mathbf{p} = f(\mathbf{q})$



Cinématique inverse

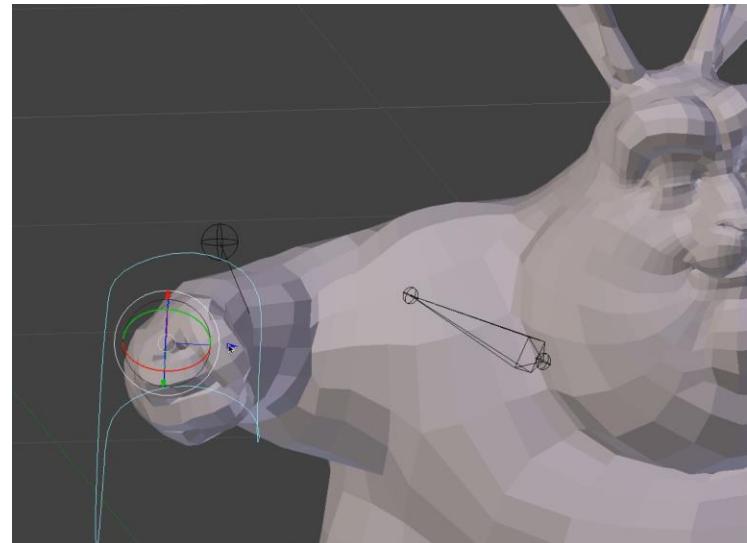
Entrée : position à atteindre \mathbf{p}

Sortie : valeurs des paramètres des articulations

$$\mathbf{q} = (\theta_1, \theta_2, \theta_3, \dots, d_1, d_2, \dots)$$

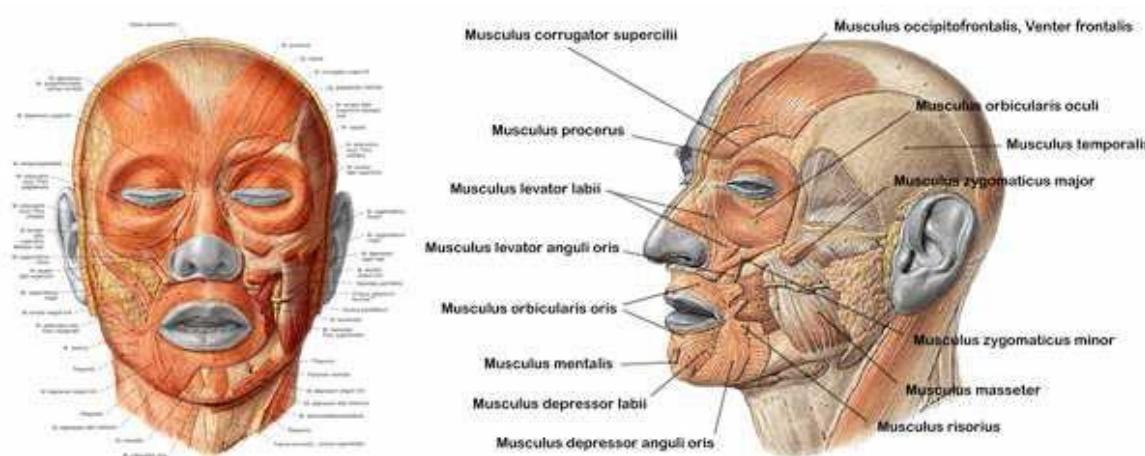
On veut trouver \mathbf{q} tel que

$$f(\mathbf{q}) = \mathbf{p} \text{ c'est à dire } \mathbf{q} = f^{-1}(\mathbf{p})$$



Déformation de surface

- Squelette d'animation vs. modèle visuel
- En réalité, forme visible composée de tissus organiques



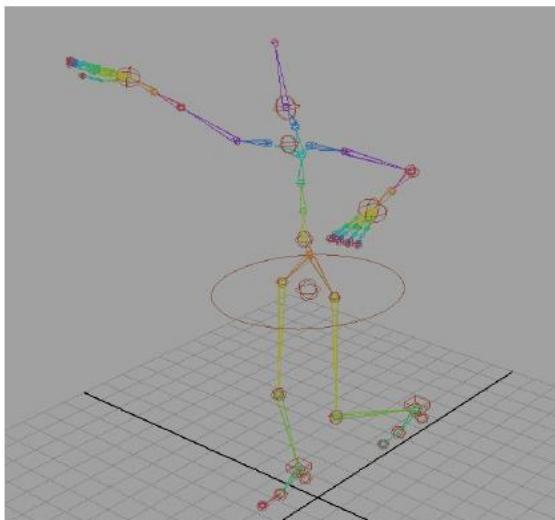
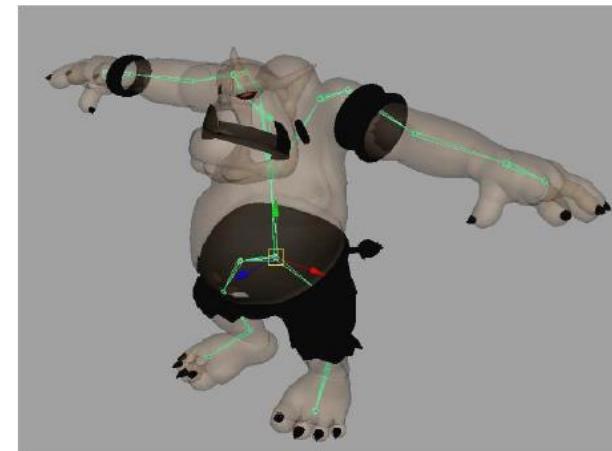
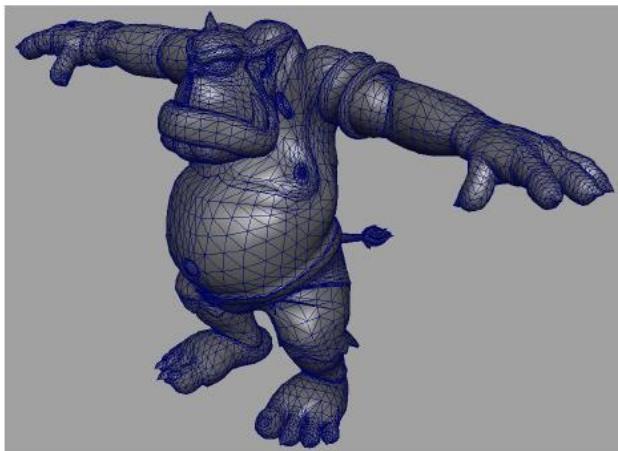
Modèle visuel

- Le niveau de réalisme dépend de l'application

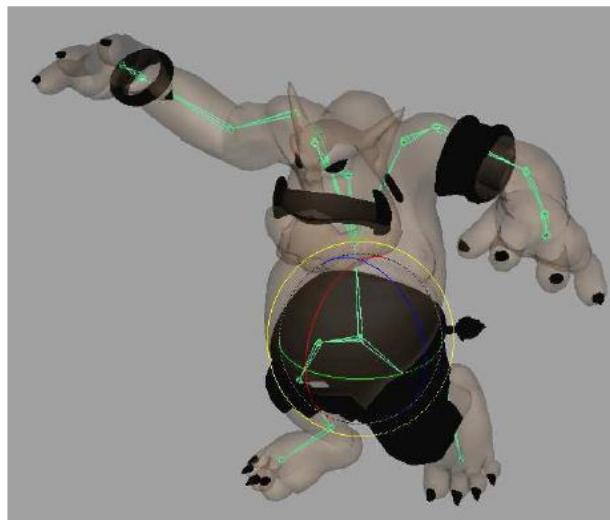


Pipeline 3D

**Squelettage ou rigging
+ calcul des poids de skinning**



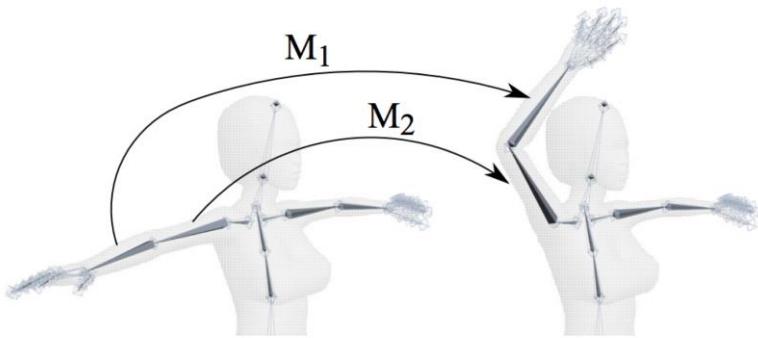
Animation



Calcul des positions
des sommets du maillage



Rigging



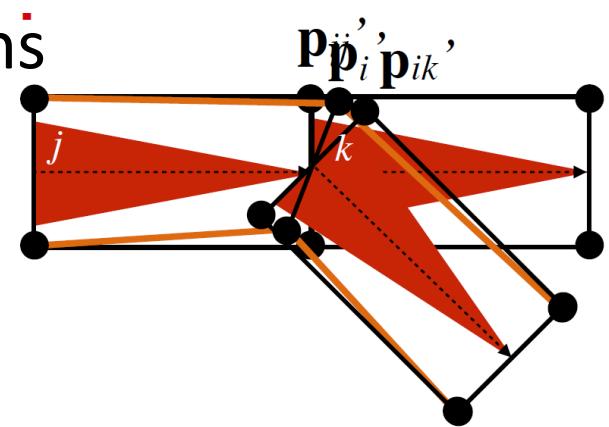
- **Entrées** : sommets p_i de la surface 3D (*skin*) et squelette dans une pose de base (*bind/rest pose*)
 - **Sortie** : positions p_i' après mouvement du squelette
 - On connaît p_i dans le repère monde (ou objet)
 - On connaît la matrice B_j de passage de l'os j au repos au repère monde
- ⇒ nouvelles coordonnées : $p_i' = T_j B_j^{-1} p_i = M_j p_i$
- T_j la matrice de transformation de l'os j

Skinning linéaire

Linear Blend Skinning [Alexa et al. 02]

- Sommets influencés par **plusieurs os**
⇒ **interpolation linéaire** des positions
- Si p_i lié aux os j et k :
$$p_i' = w p_{ij}' + (1-w) p_{ik}'$$

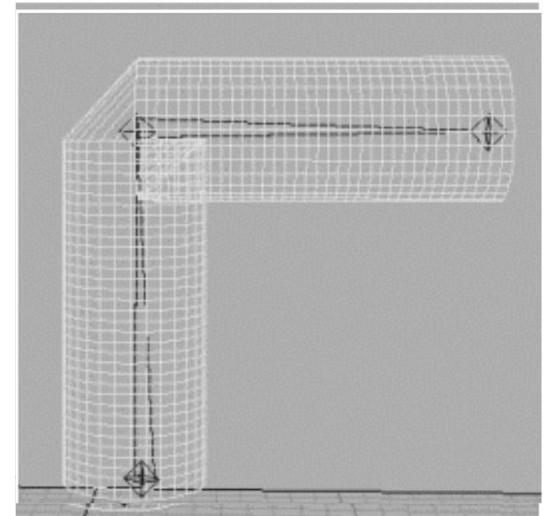
 $w \in [0,1] \text{ skin weight}$



Cas général

$$\begin{aligned} p_i' &= \sum_j w_{ij} T_j B_j^{-1} p_i \\ &= \sum_j w_{ij} M_j p_i \end{aligned}$$

avec $\sum_j w_{ij} = 1$



David Lanier, 2002

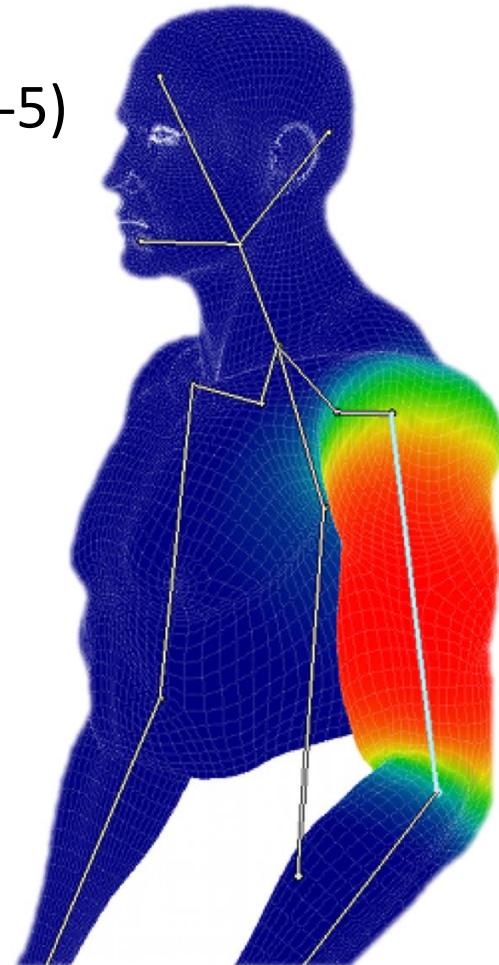
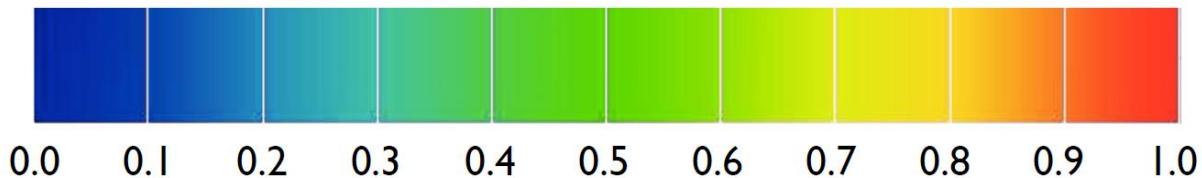
Poids de skinning

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

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

et/ou calculés automatiquement

(distance euclidienne, harmonique, géodésique)



Skinning linéaire

1. *Rigging* – pré-calcul

⇒ matrices de pose : B_j

⇒ poids de *skinning* : w_{ij}

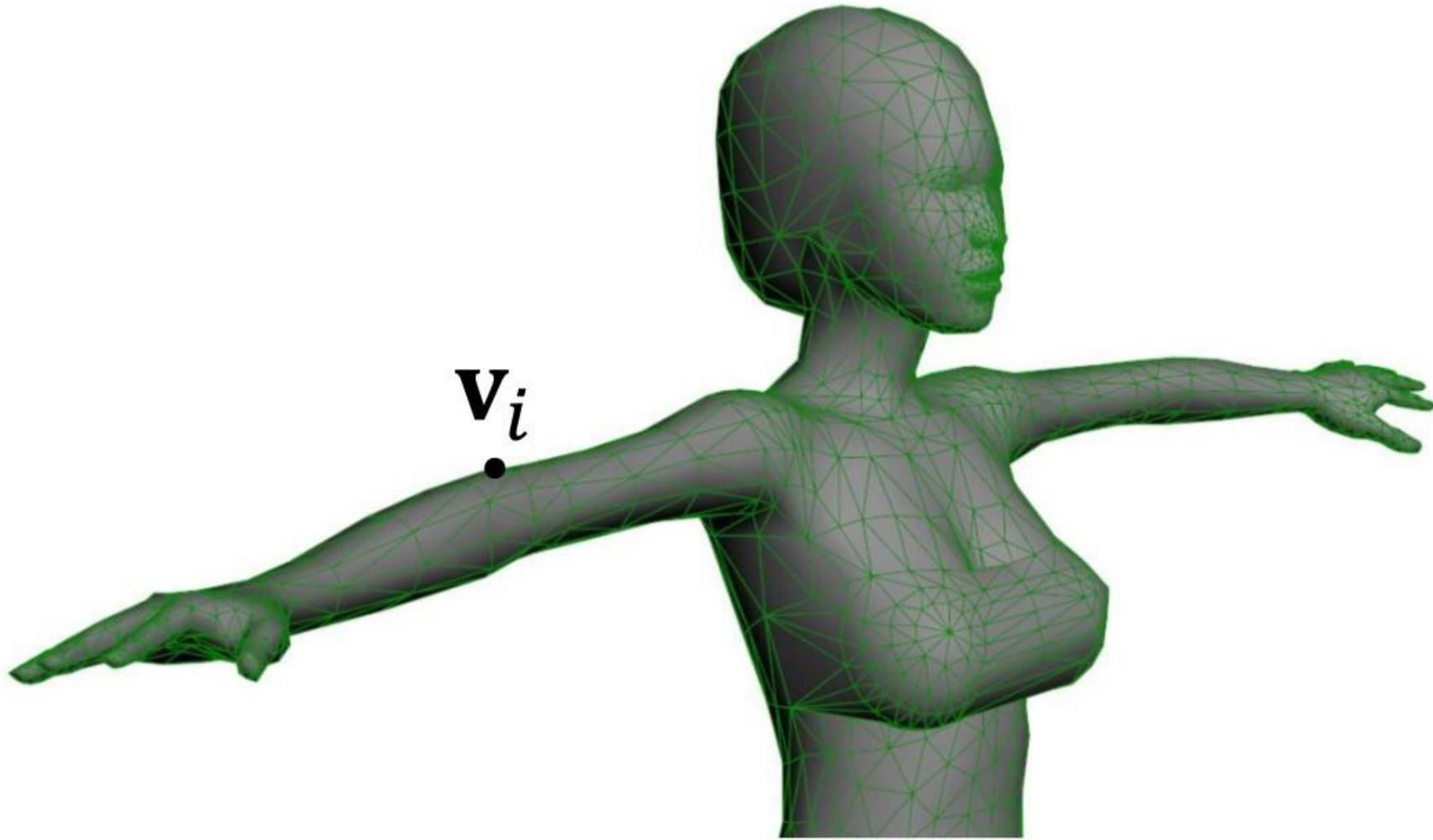
2. Cinématique direct ou inverse – à chaque temps t

⇒ matrices de transformation $M_j(t) = T_j(t)B_j^{-1}$

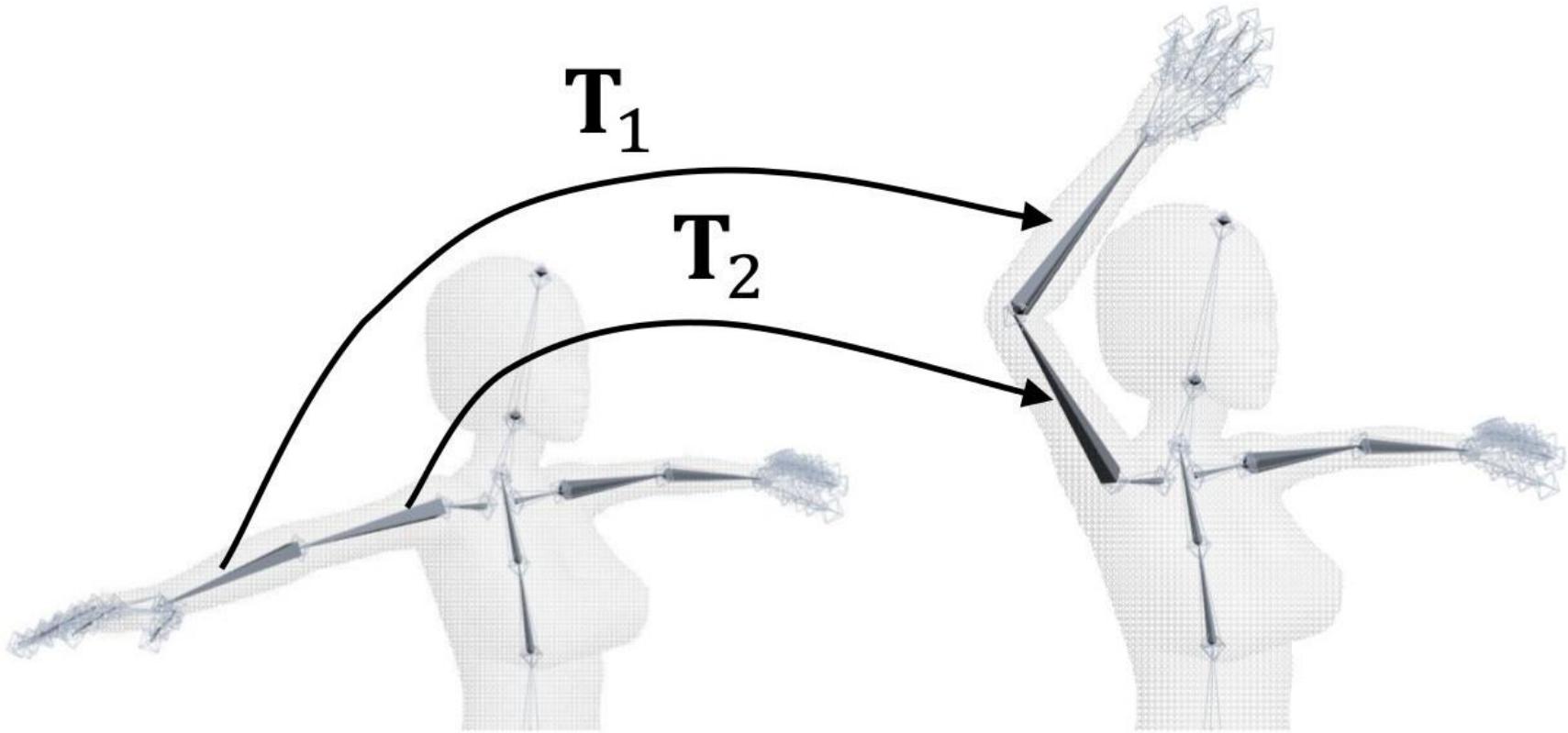
3. Skinning linéaire – *vertex shader*

⇒ $p_i' = \sum_j w_{ij} M_j(t) p_i$

TP :Linear blend skinning

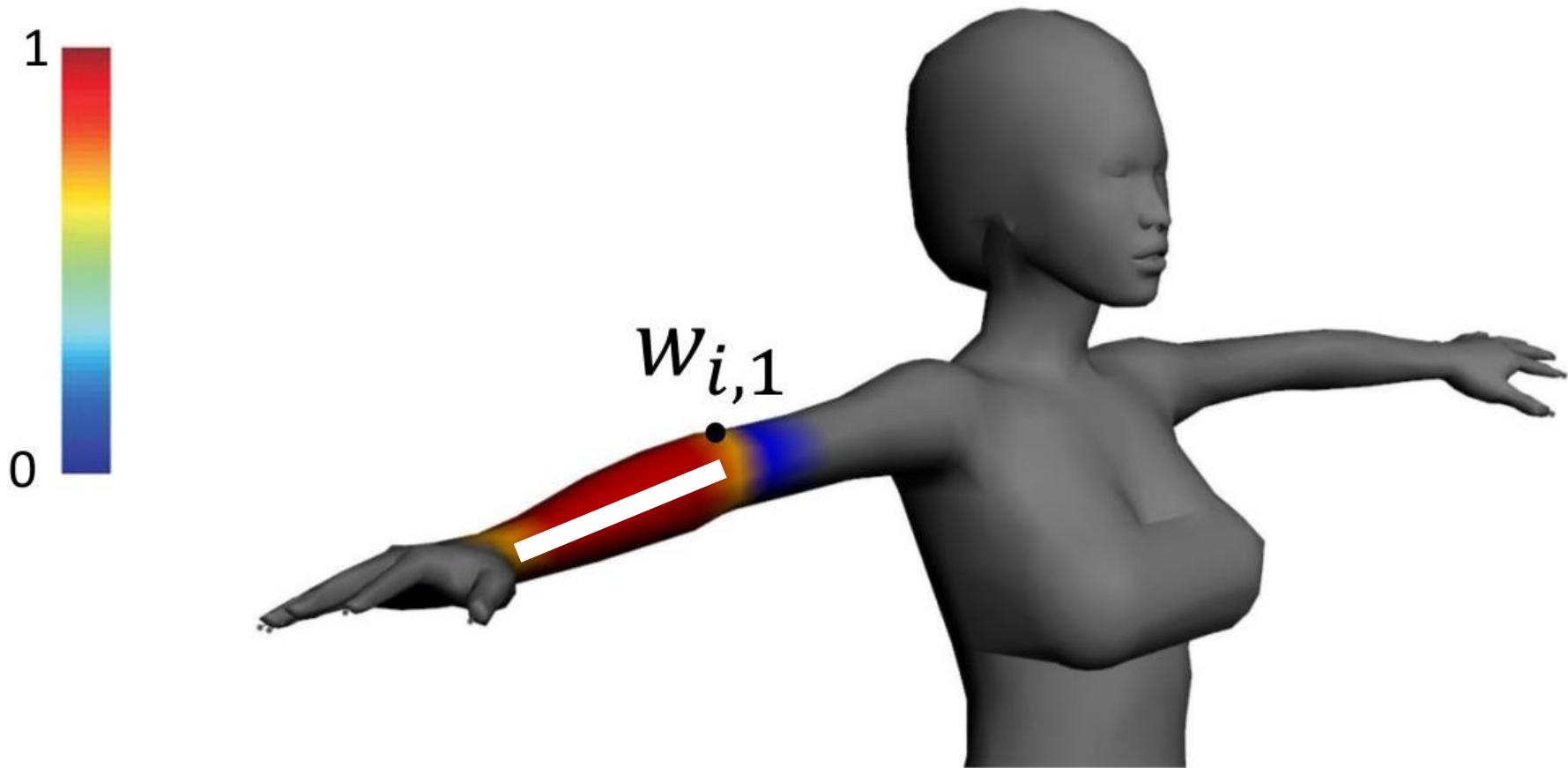


Linear blend skinning

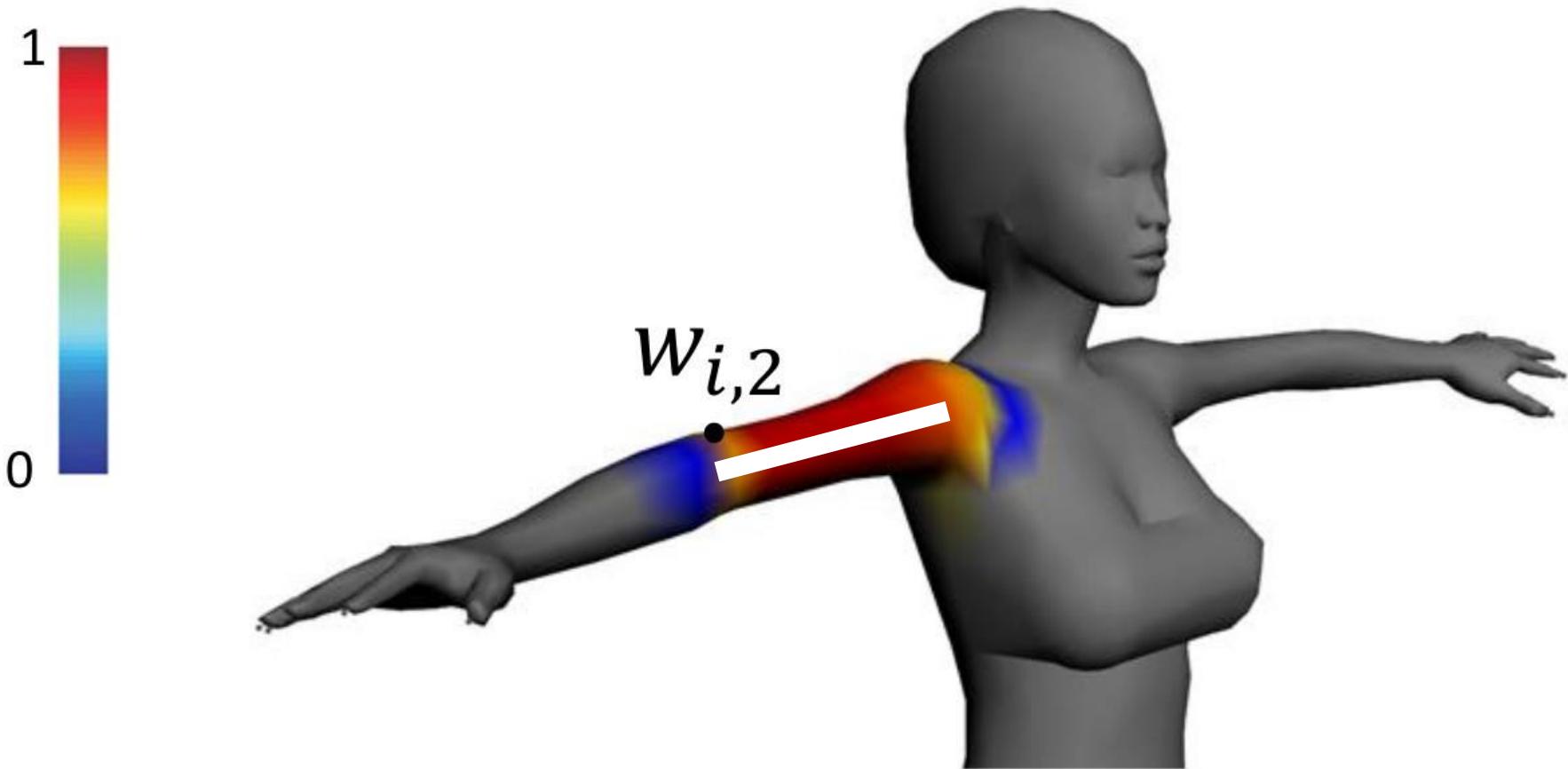


Nous devons définir l'influence des os sur les somets du maillage

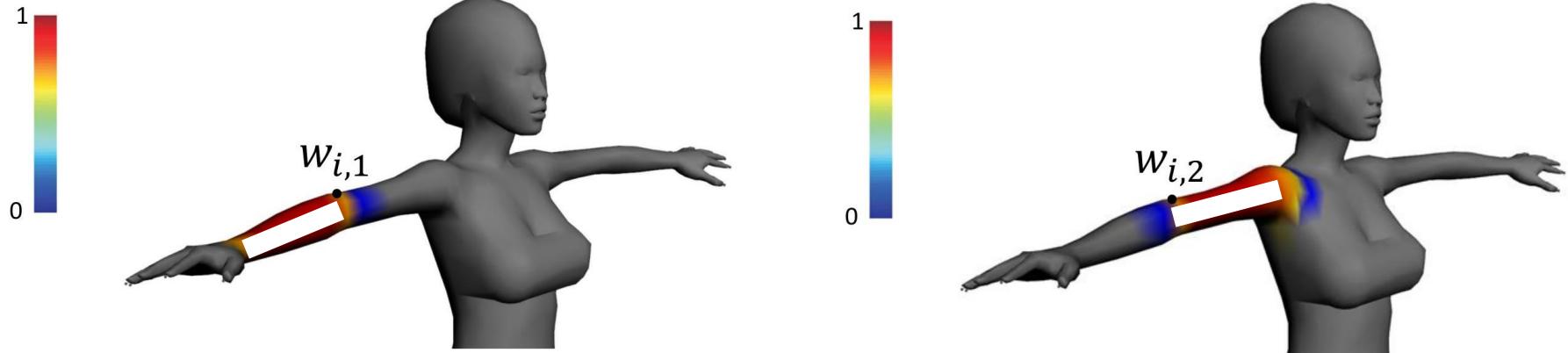
Linear blend skinning



Linear blend skinning



Linear blend skinning



$$f: v_i \rightarrow \sum_j w_{ij} (R_j \cdot v_i + T_j)$$

Sommet i

Poids de l'os j sur le sommet i

Fonction de déformation

Os influençant le sommet i

Rotation de l'os j

Translation de l'os j

The diagram illustrates the linear blend skinning formula. It shows a function f mapping a vertex v_i to a weighted sum of transformed vertices. The weight w_{ij} represents the influence of bone j on vertex i . The transformation is composed of a rotation R_j and a translation T_j . Arrows point from the text labels to their corresponding components in the formula.

Propriété des poids de skinning

$$f: v_i \rightarrow \sum_j w_{ij} (R_j \cdot v_i + T_j)$$

Poids basés sur la distance euclidienne :

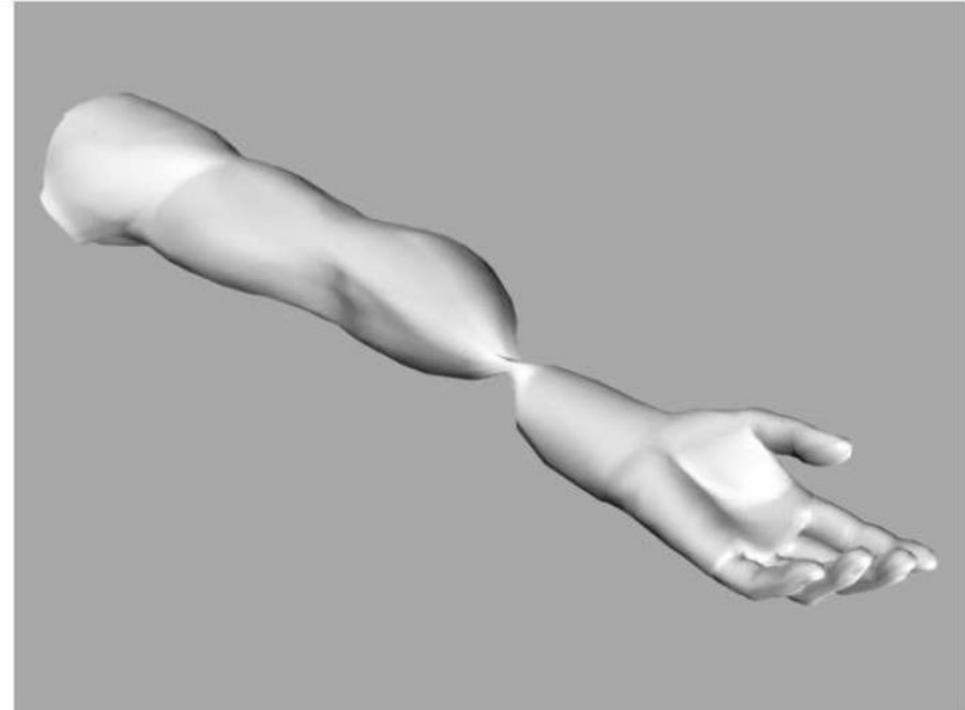
$$w_{ij} = \left(\frac{1}{d_{ij}} \right)^n \text{ avec } n \geq 1$$

- Positifs $w_{ij} \geq 0$
- Affines $\sum_j w_{ij} = 1 \quad \rightarrow \quad w_{ij} = \frac{w_{ij}}{\sum w_{ij}}$

Skinning linéaire

- **Limitation : rotations**

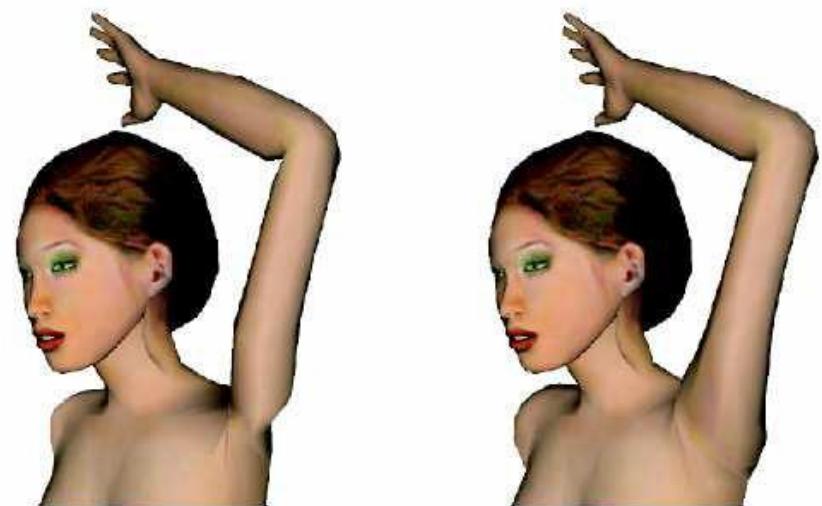
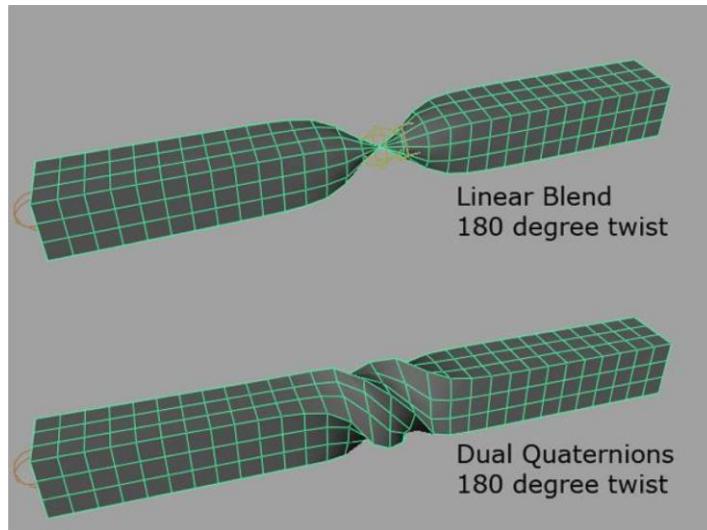
(la moyenne de 2 rotations n'est pas une rotation !)



Skinning + quaternions duaux

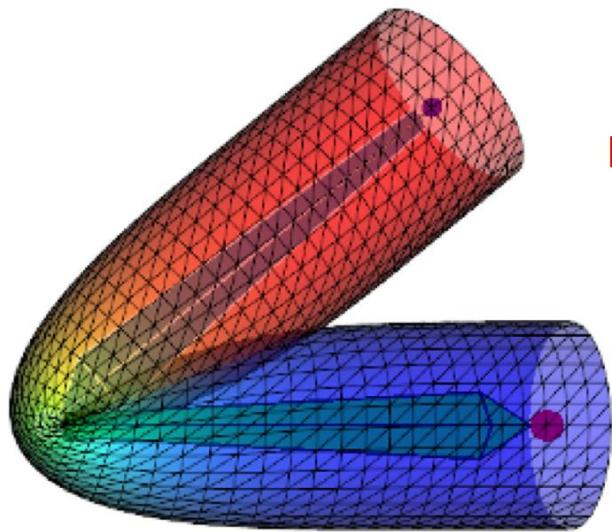
[Kavan et al. 08]

Interpolation des matrices de transformation à l'aide de
quaternions duaux (encodant rotation et translation)



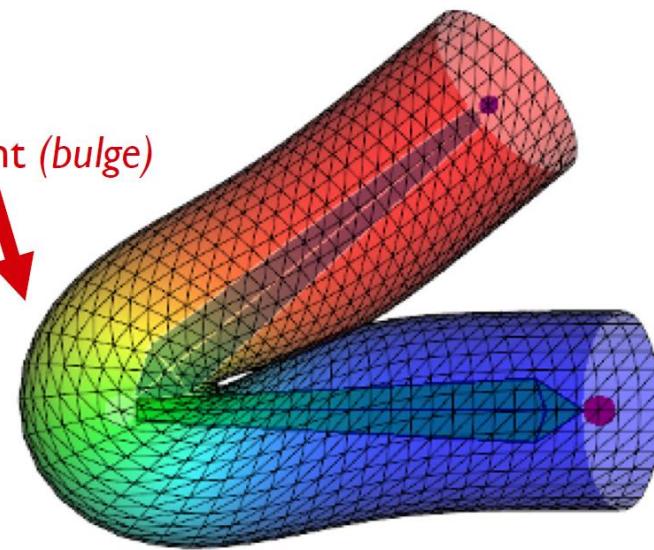
Skinning + quaternions duaux

[Kavan et al. 08]

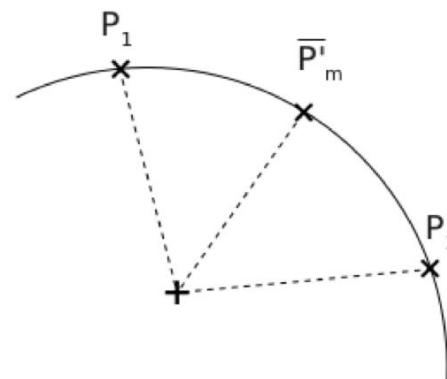
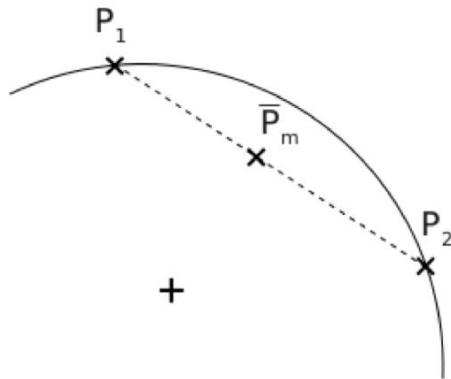


Skinning linéaire

Renflement (*bulge*)



Dual quaternions



Recalcule des normales

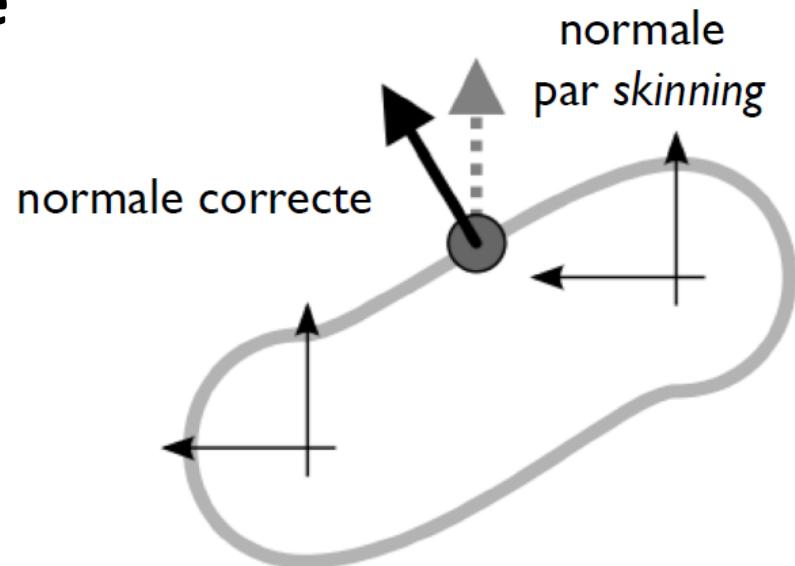
Calculer la transformation linéaire moyenne pour le sommet i : $C_i = \sum_j w_{ij} M_j$

...et transformer sa normale par $(C_i^{-1})^T$

⇒ **approximation grossière**



Pose de base

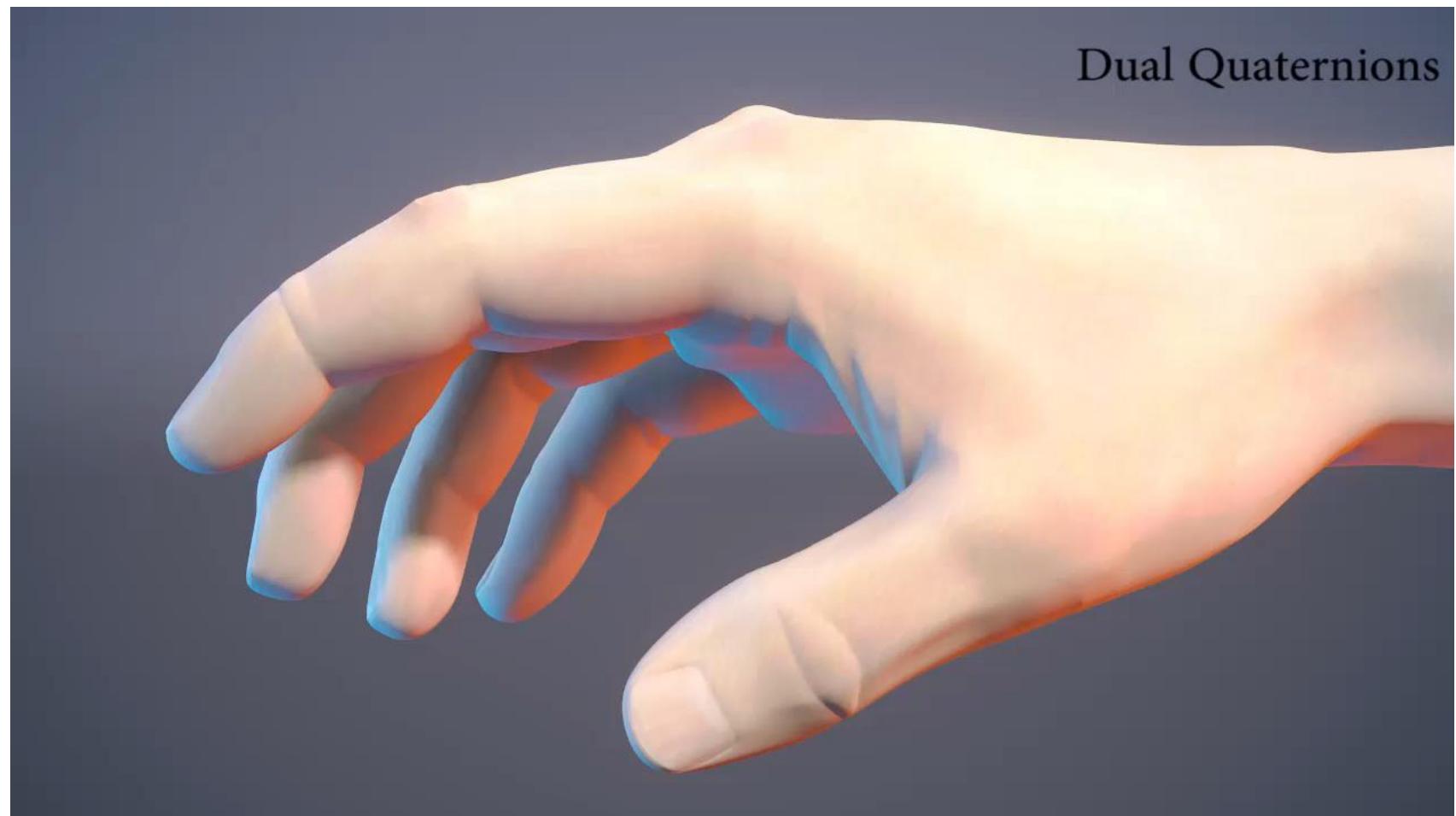


Pose déformée

Skinning + surface implicites

[Vaillant et al. 13 & 14]

Modélisation des **contacts**



Blending transformations

$$f: v_i \rightarrow \sum_j w_{ij} (R_j \cdot v_i + T_j)$$

Blend « the transformed vertices »

Blending transformations

$$f: v_i \rightarrow \left(\sum_j w_{ij} R_j \right) \cdot v_i + \left(\sum_j w_{ij} T_j \right)$$


?

Blend « the transformations »

$$1/2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} +$$

$$1/2 \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} =$$

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Rotation
by 0

Rotation
by π

Not a
rotation

Blending transformations

$$f: v_i \rightarrow \left(\sum_j w_{ij} R_j \right) \cdot v_i + \left(\sum_j w_{ij} T_j \right)$$


?

Blend « the transformations »

$$1/2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} +$$

Rotation
by 0

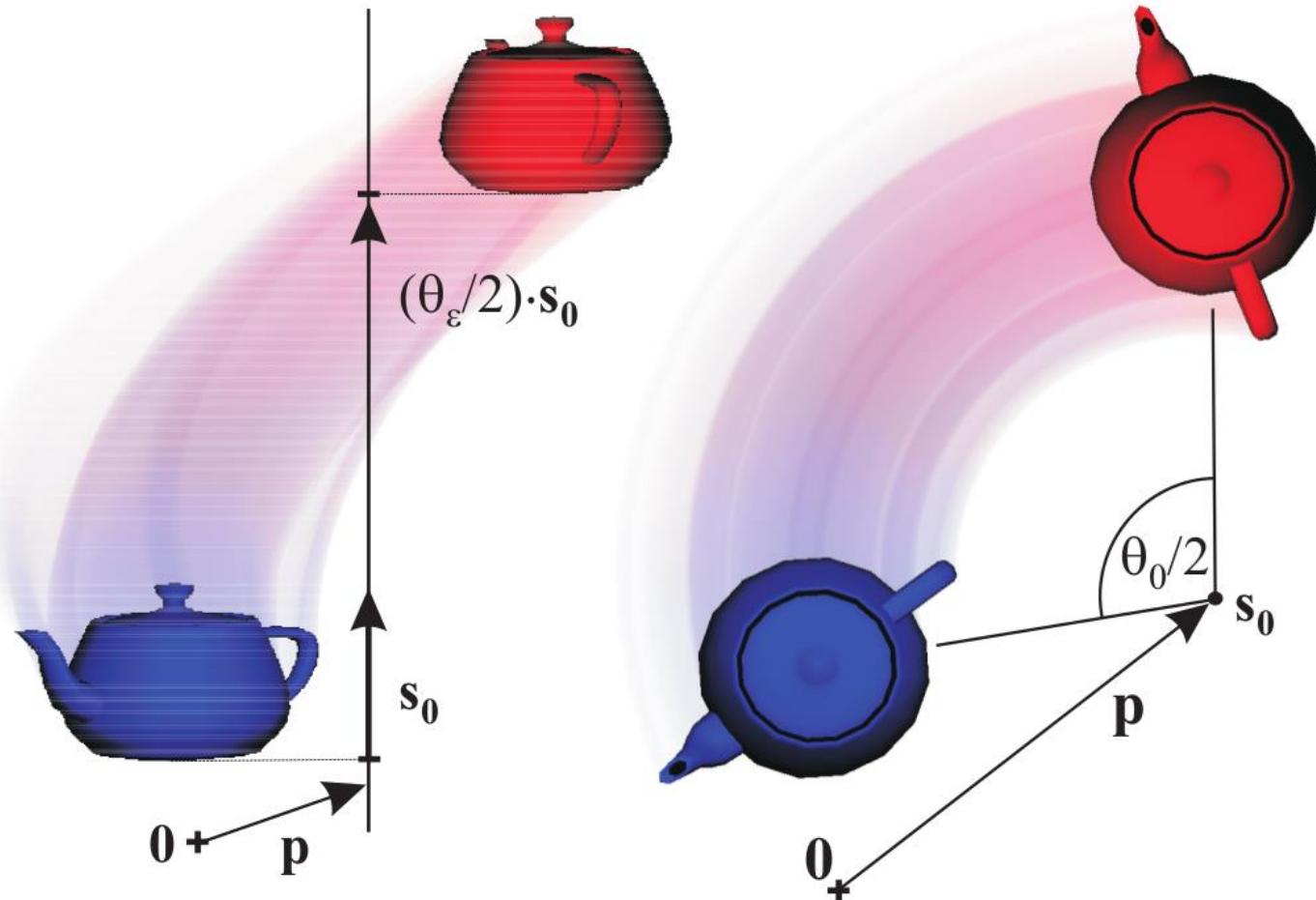
$$1/2 \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \ll = \gg$$

Rotation
by π

$$\sqrt{2}/2 \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

Rotation
by $\pi/2$

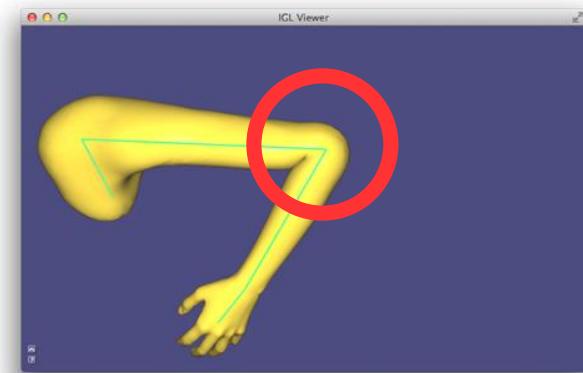
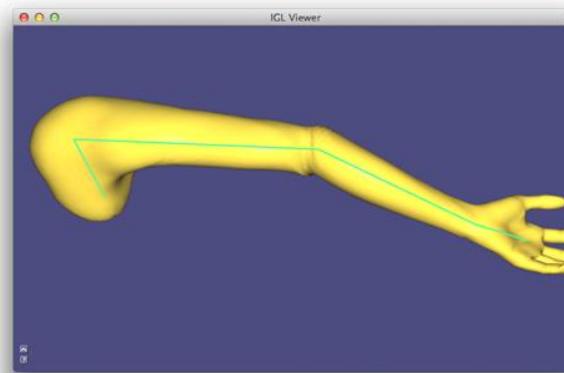
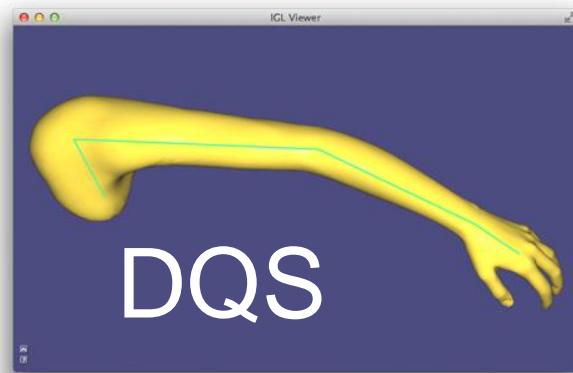
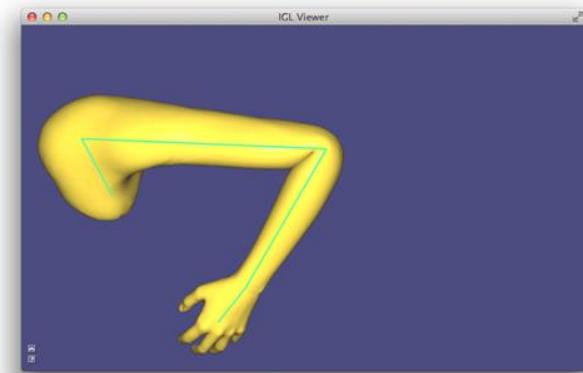
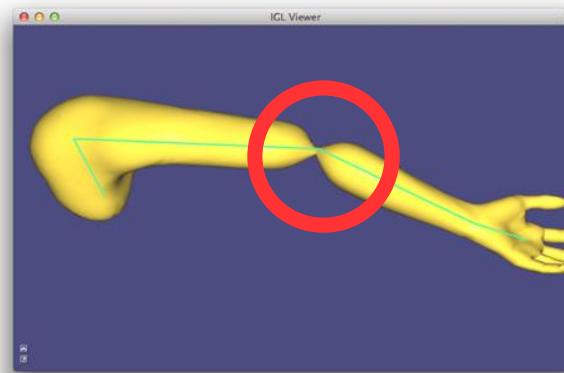
Dual quaternion Skinning



[Kavan et al.] : Skinning with Dual Quaternions

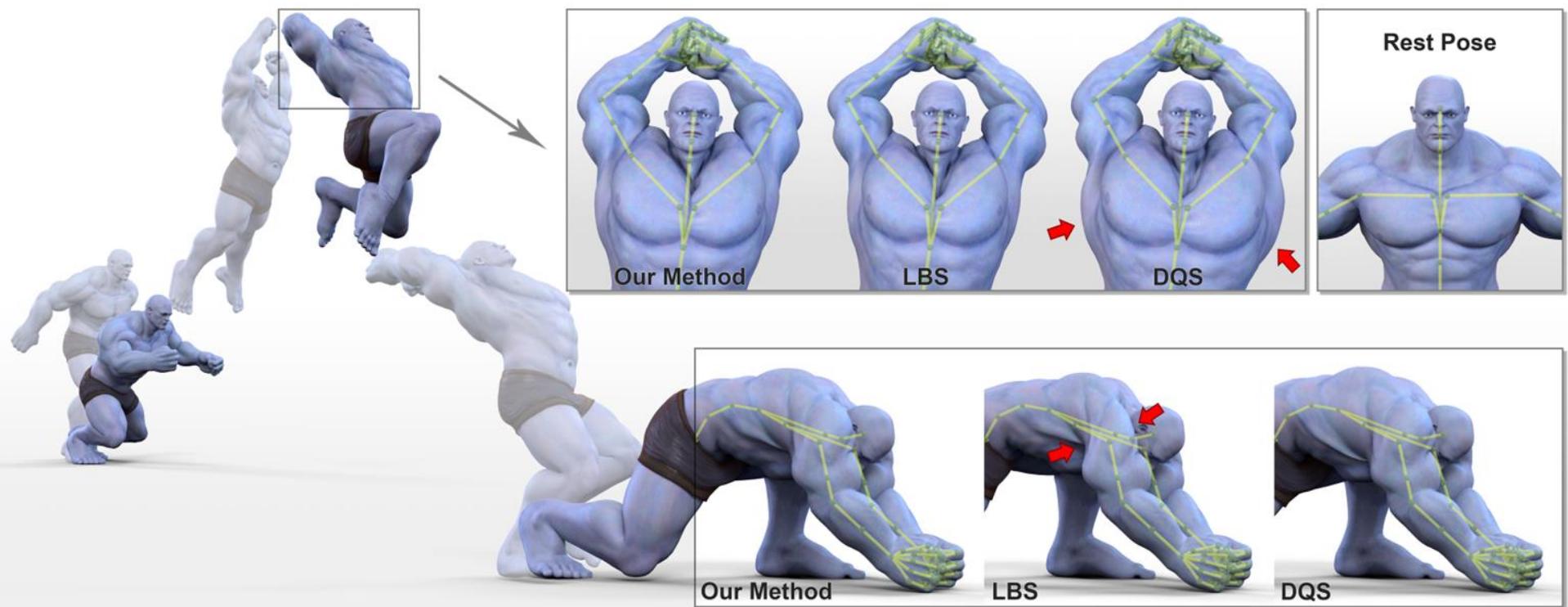
Dual quaternion Skinning

« candy-wrapper »



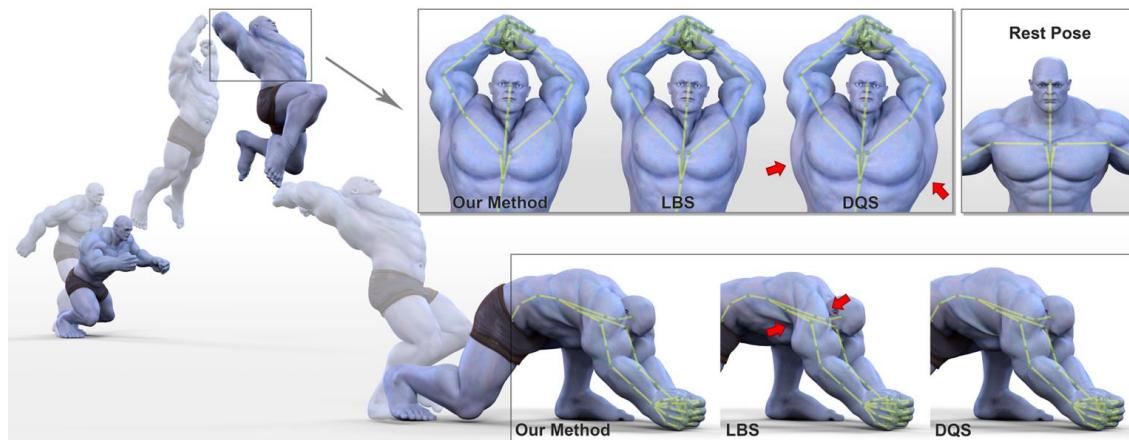
« bulge »

Disney's CoRs



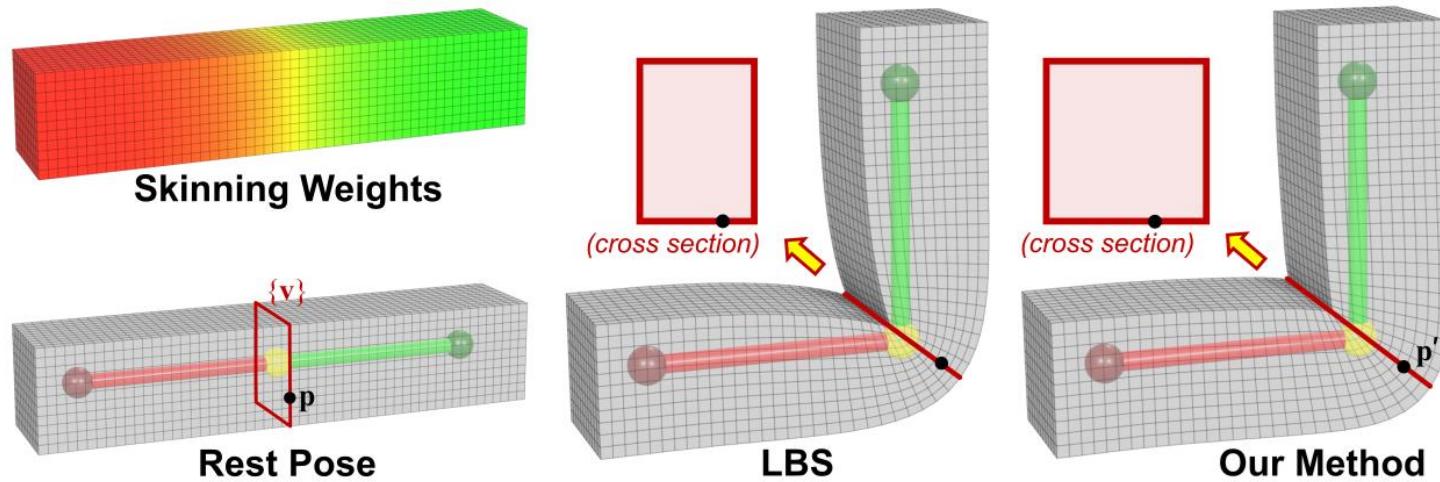
Key ideas

- LBS and DQS have « orthogonal » problems
- DQS is good at blending the rotations
- The bulge effect is due to a non-optimized translation (or a non-optimized center of rotation)



What is a good CoR ?

- Vertices with similar weights will have a similar rotation
- They are organized as cross sections orthogonal to the bones, and ideally should be transformed rigidly.



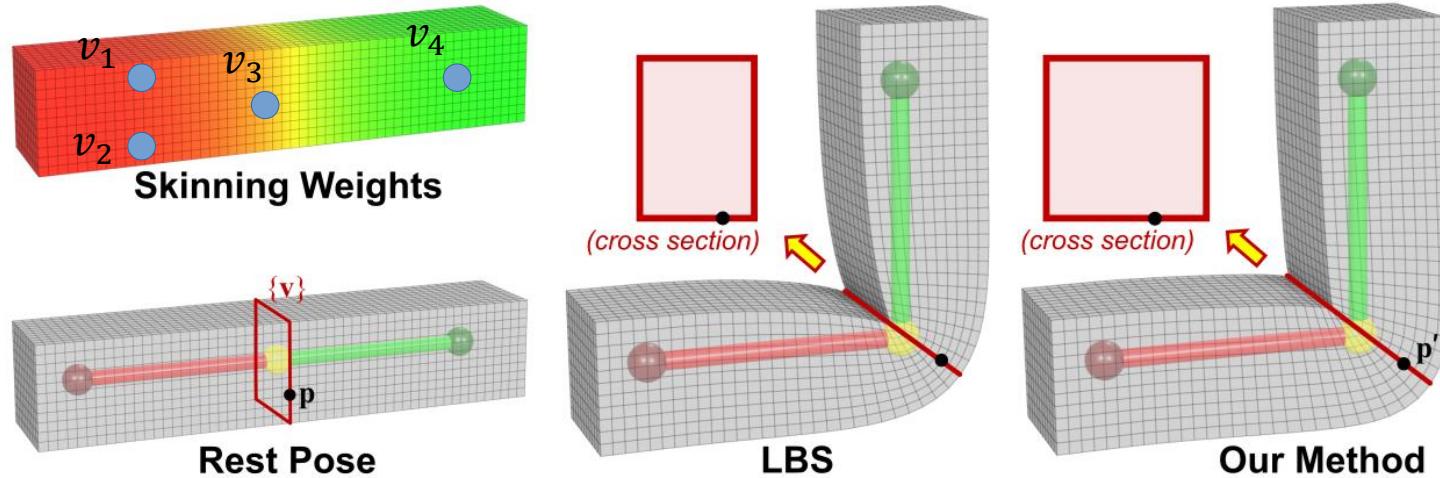
What is a good CoR ?

- Vertices with similar weights will have a similar rotation
- They are organized as cross sections orthogonal to the bones, and ideally should be transformed rigidly.
- Requires a similarity function between weights

$$s(w_1, w_2) = 1$$

$$s(w_1, w_3) = 0.01$$

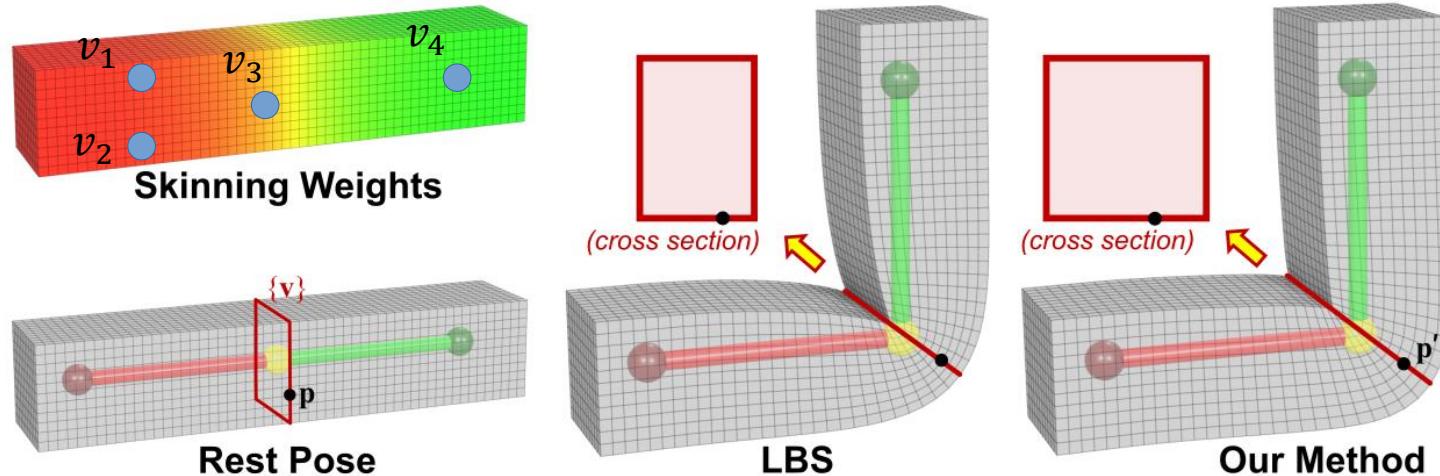
$$s(w_1, w_4) = 0$$



What is a good CoR ?

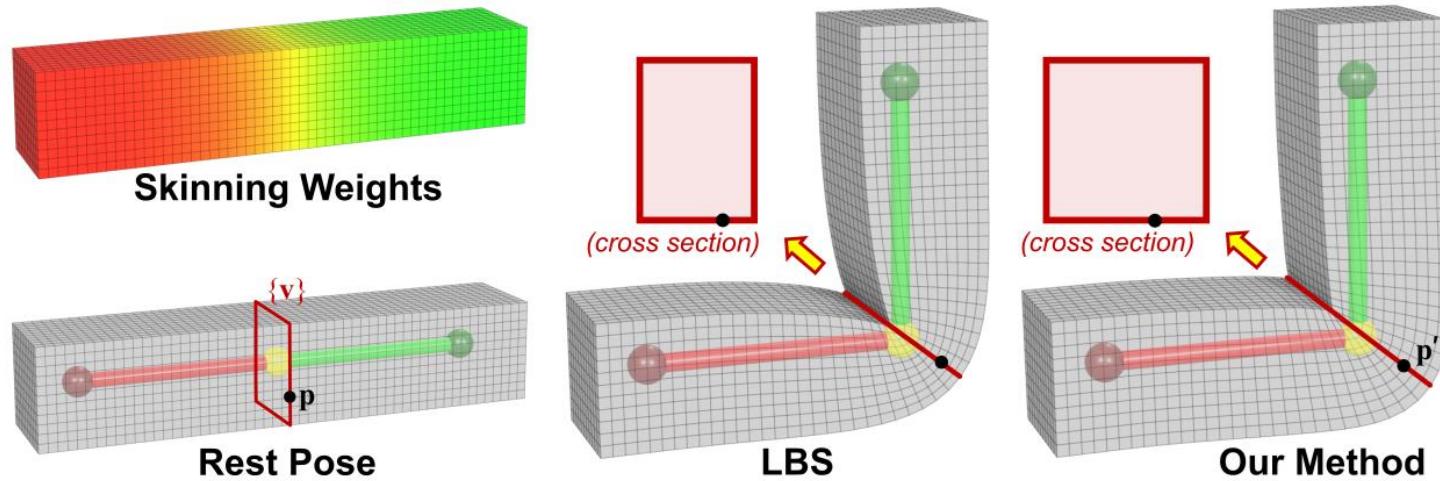
- Vertices with similar weights will have a similar rotation
- They are organized as cross sections orthogonal to the bones, and ideally should be transformed rigidly.
- Requires a similarity function between weights

$$s(\mathbf{w}_p, \mathbf{w}_v) = \sum_{\forall j \neq k} \mathbf{w}_{pj} \mathbf{w}_{pk} \mathbf{w}_{vj} \mathbf{w}_{vk} e^{-\frac{(\mathbf{w}_{pj} \mathbf{w}_{vk} - \mathbf{w}_{pk} \mathbf{w}_{vj})^2}{\sigma^2}}$$



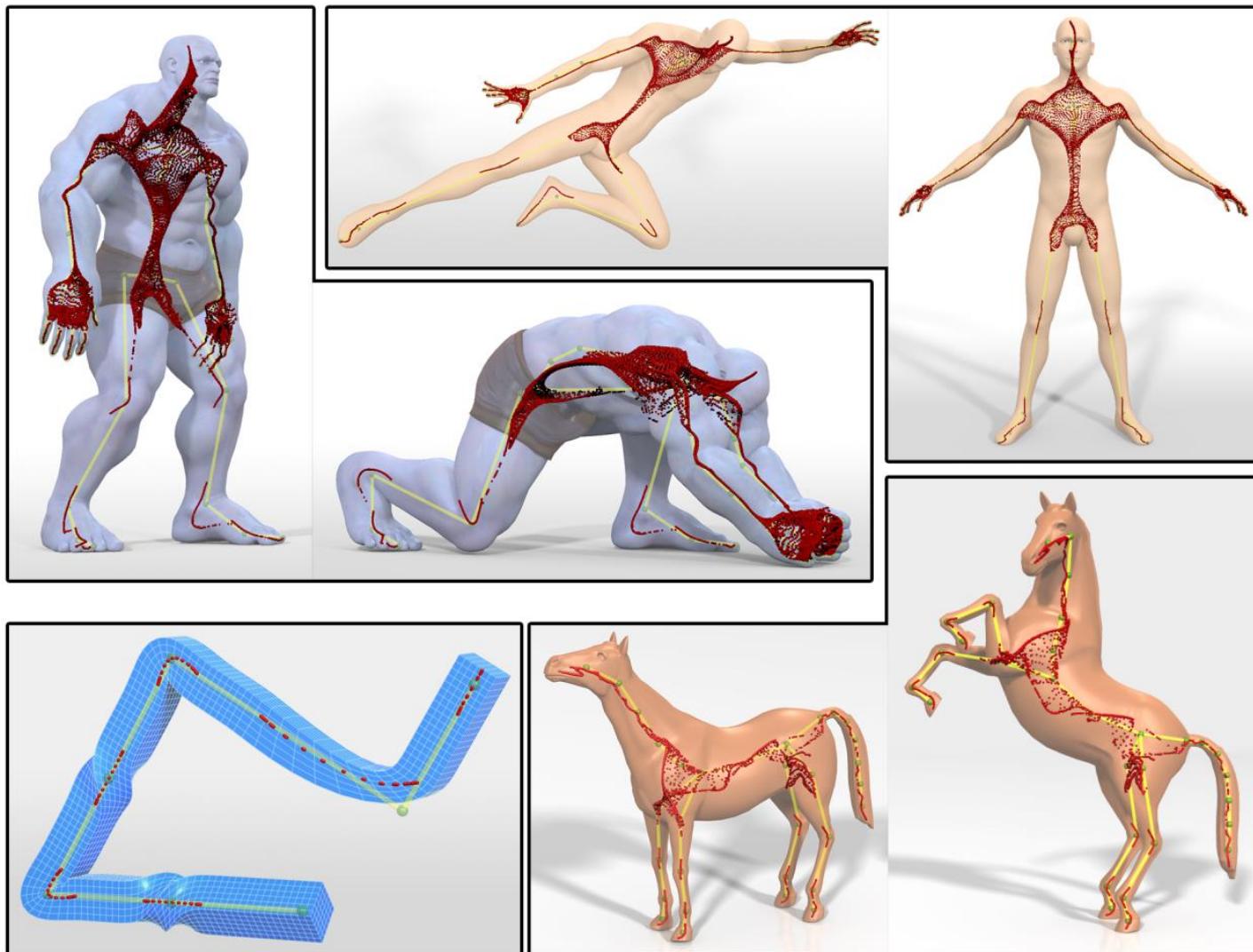
Idea

- Consider the LBS transformation of the mesh
- Use the DQS rotation for each vertex
- Optimize per-vertex translation to fit the LBS deformation while enforcing rigid sections.



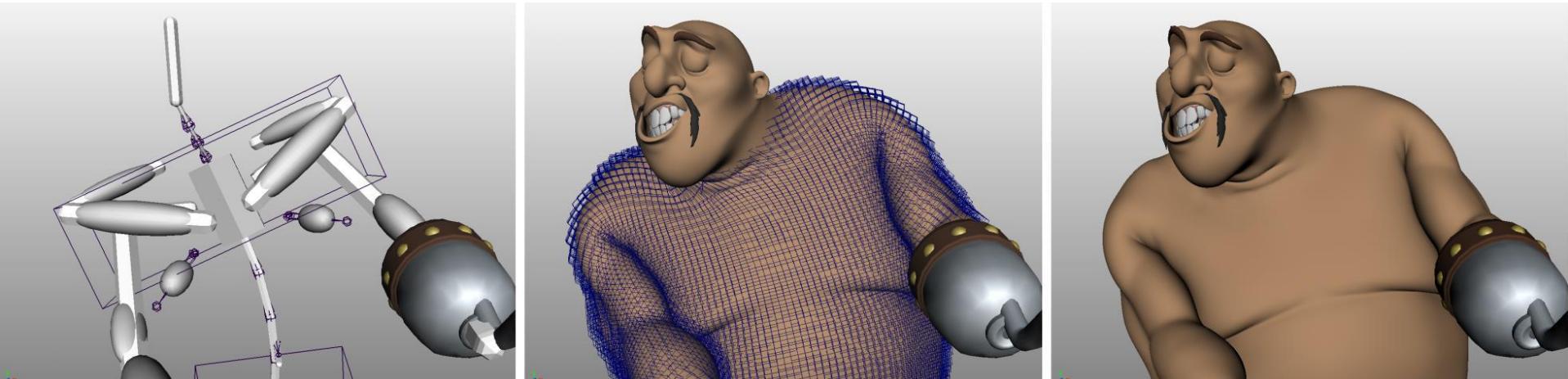
$$\mathbf{p}^* = \frac{\int_{\mathbf{v} \in \Omega} s(\mathbf{w}_p, \mathbf{w}_v) \mathbf{v} d\mathbf{v}}{\int_{\mathbf{v} \in \Omega} s(\mathbf{w}_p, \mathbf{w}_v) d\mathbf{v}}$$

CoRs



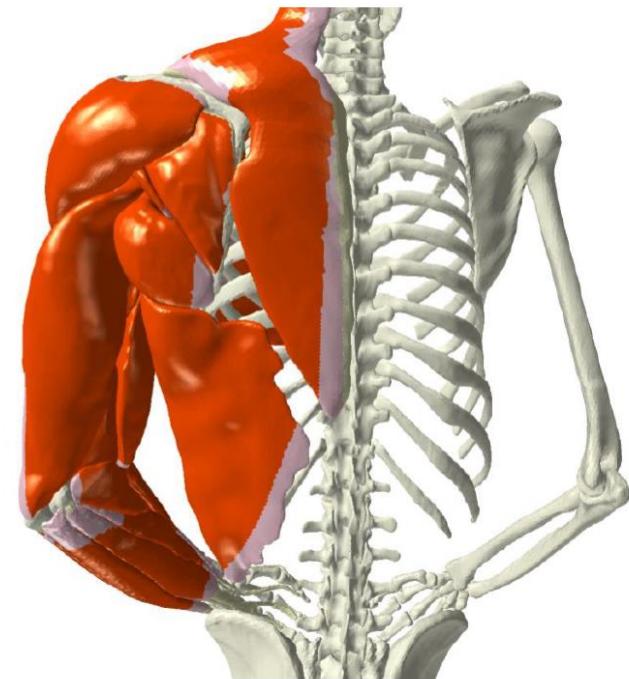
Skinning basé-physique

- Inclure le squelette dans un modèle volumique (e.g. maillage tétraédrique). Simulation de matière élastique (chaire, muscle) guidé par le squelette.



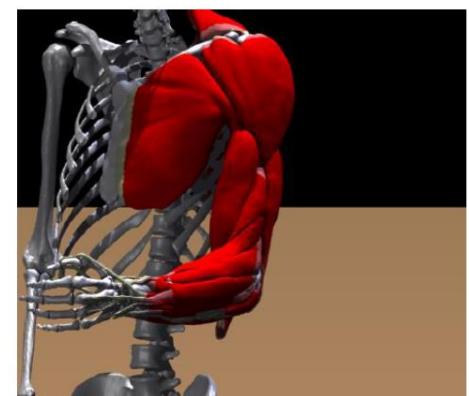
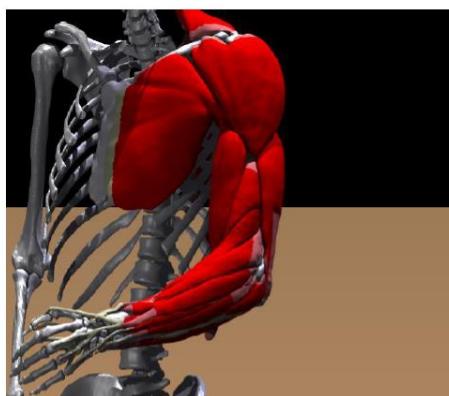
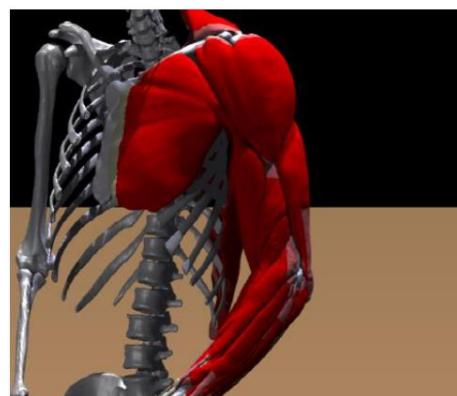
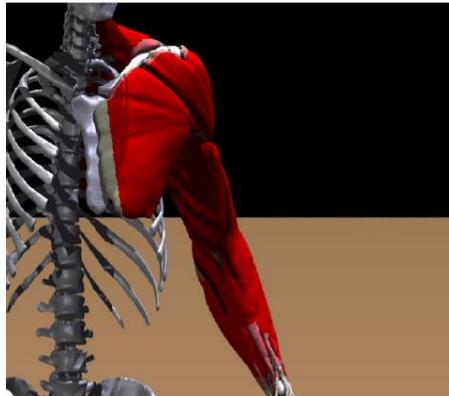
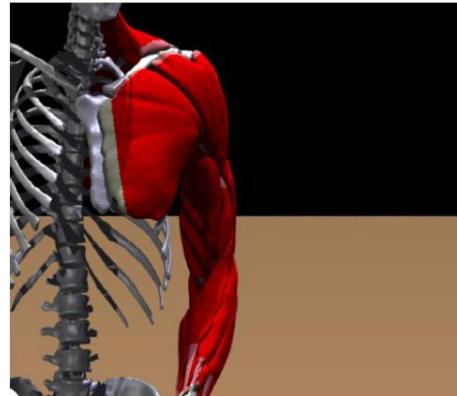
Muscles

- Amélioration du réalisme en ajoutant des muscles se contractant quand ils sont activés et exercent des forces sur les tissus mous (chair)
- Données réelles pour la forme des muscles

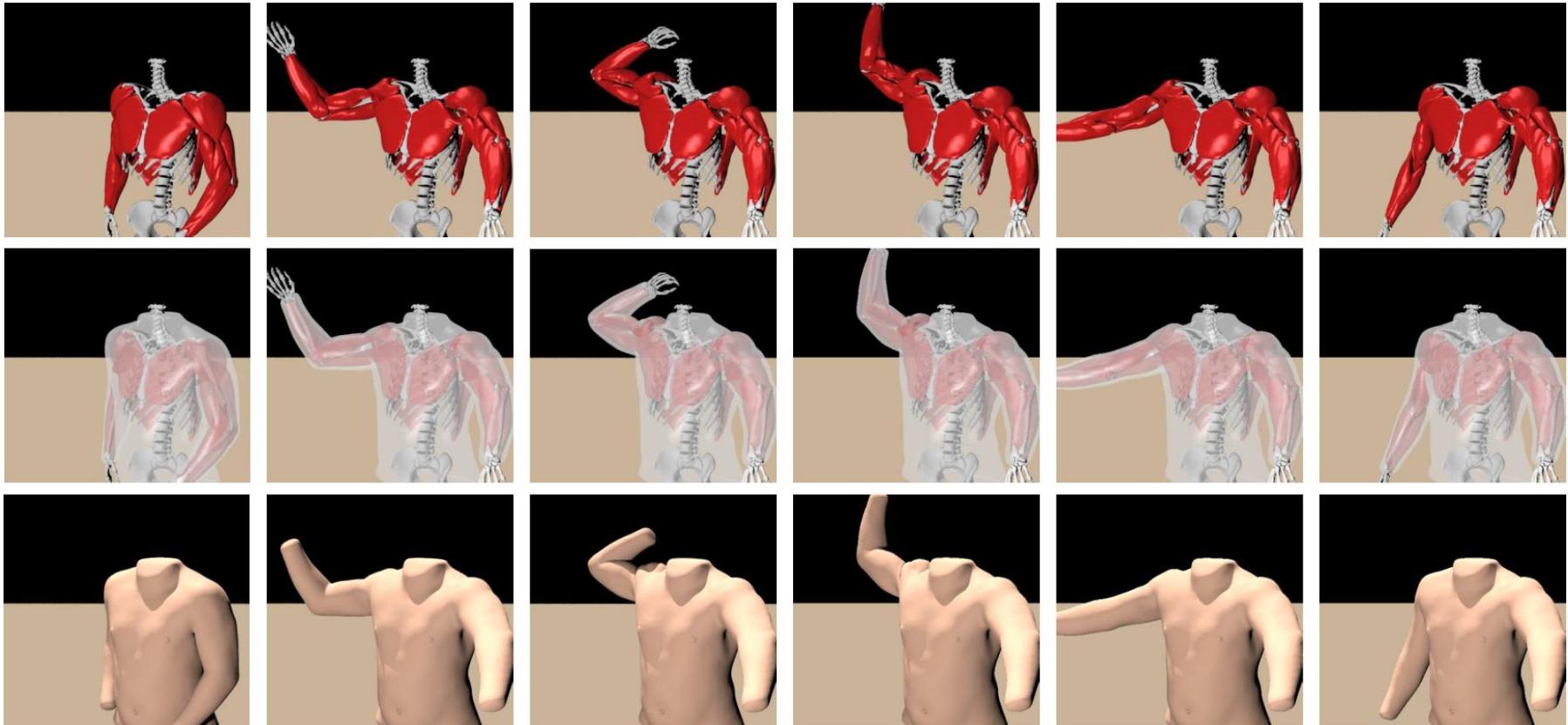


Muscles

Résoudre un problème inverse pour déduire quels muscles doivent s'activés d'après le mouvement des os



Muscles

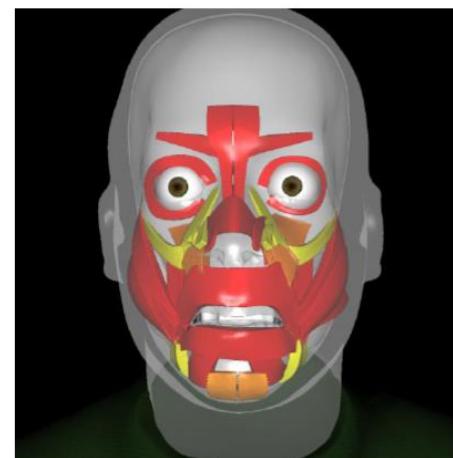
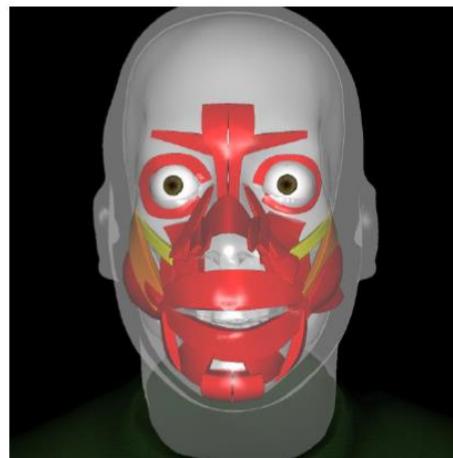


Modèle anatomique du visage

Même principe pour obtenir des expressions réalistes du visage

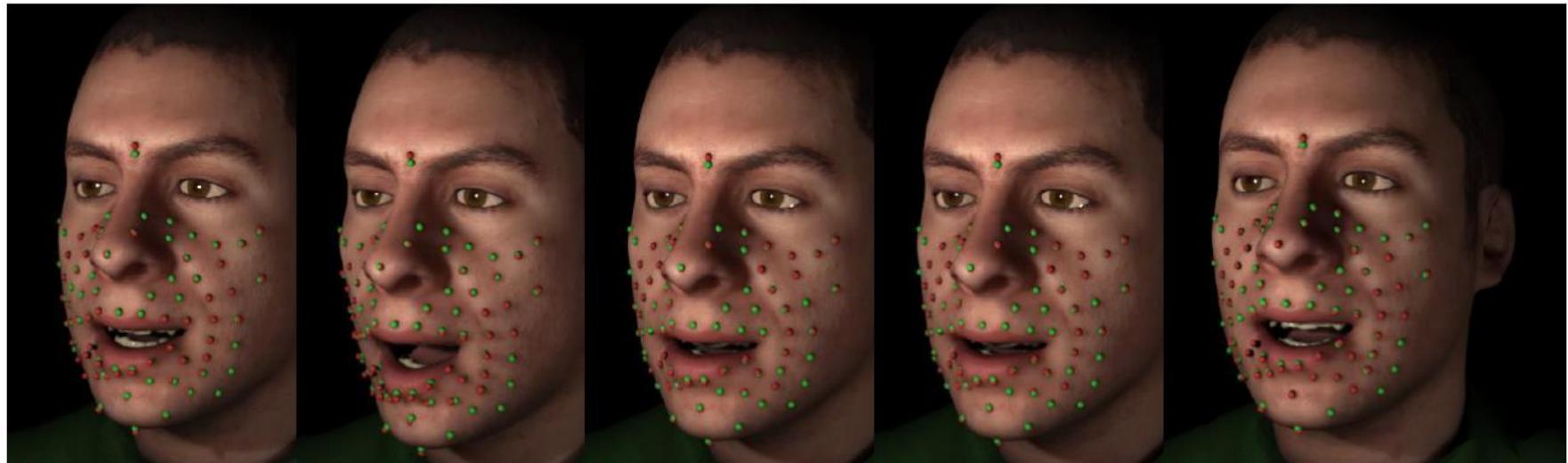
Activation :

- Jaune : complètement
- Rouge : inactif



Activation des muscles

- Utilisation de la capture de mouvement avec des marqueurs à matcher avec des positions du maillage
- Modification des expressions en interpolant les angles et les activations entre les keyframes



Capture de mouvements



Discovery, "Avatar: Motion Capture Mirrors Emotions",
<https://youtu.be/1wK1lxr-UmM>

Blend shapes

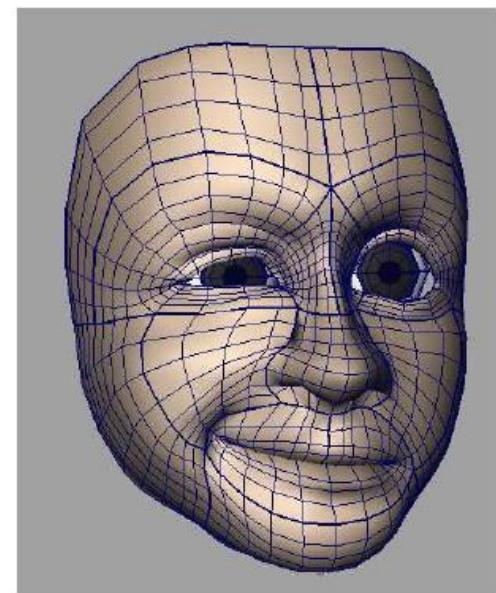
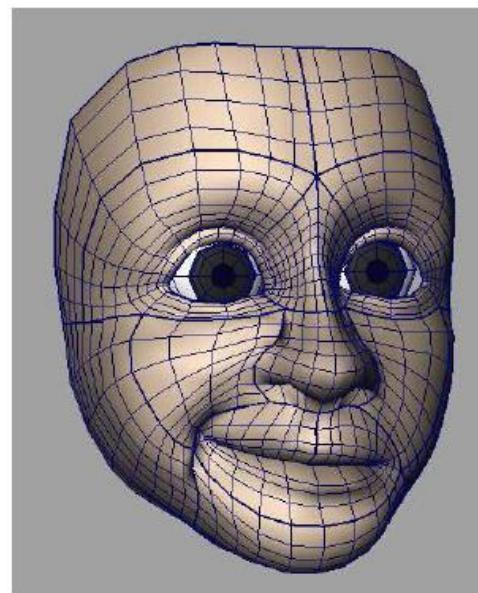
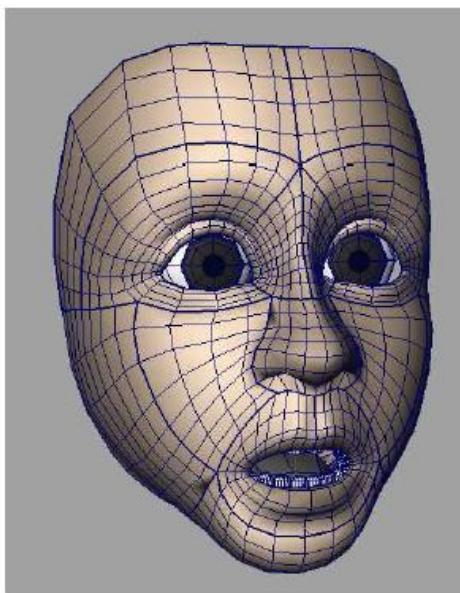
Interpolation de **formes clés**

- Interpolation linéaire de la position des sommets

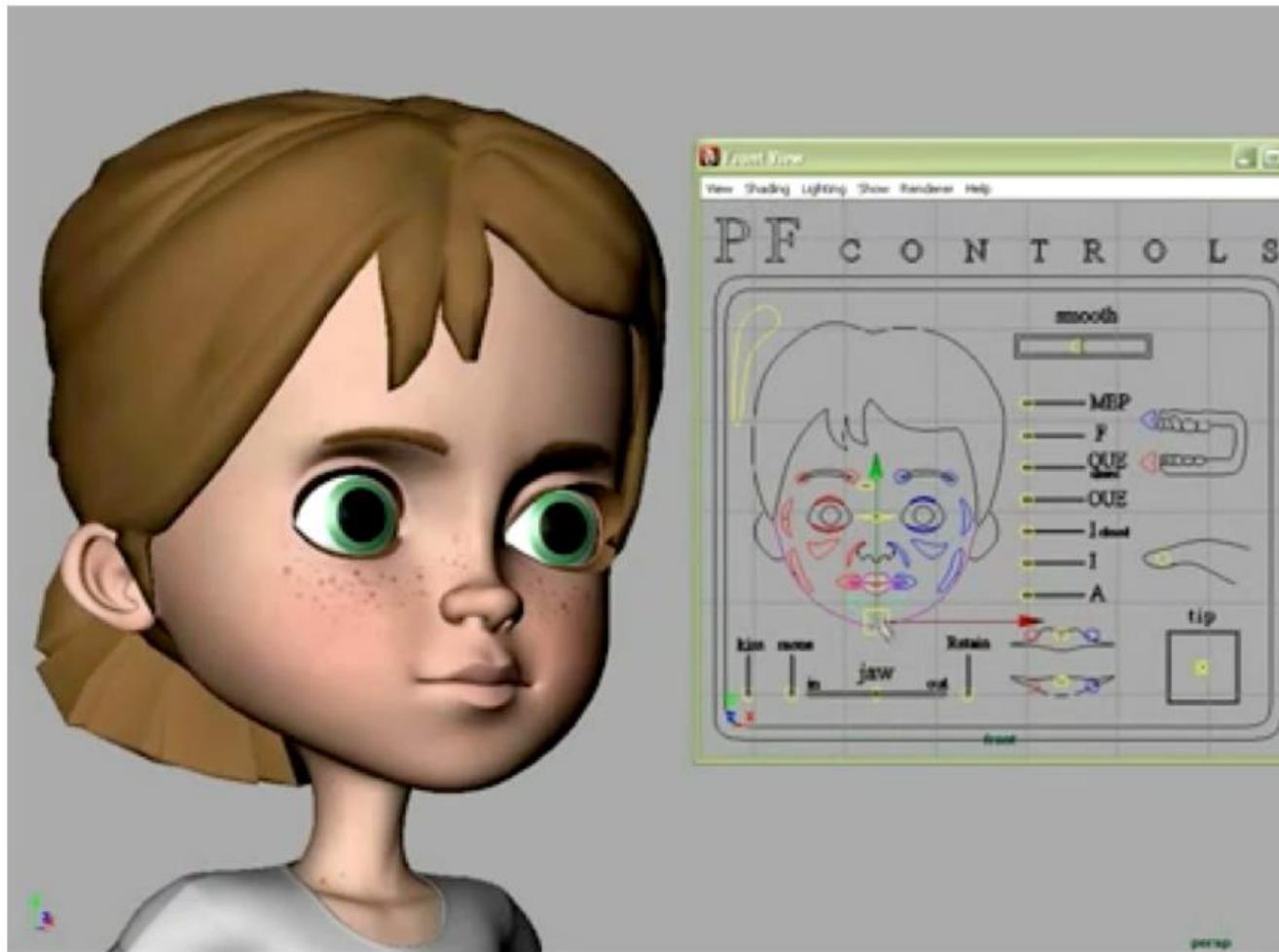
$$p' = p_0 + \sum_i w_i (p_i - p_0)$$

- Contrôle de l'animation par les poids w_i

Surtout pour l'**animation faciale**



Blend Shapes (198 expressions)



Remy Terreaux

Cinématique

Mouvement non-constraint

- Hucher la tête, faire un signe de la main...
⇒ Cinématique directe (FK)

Mouvement constraint

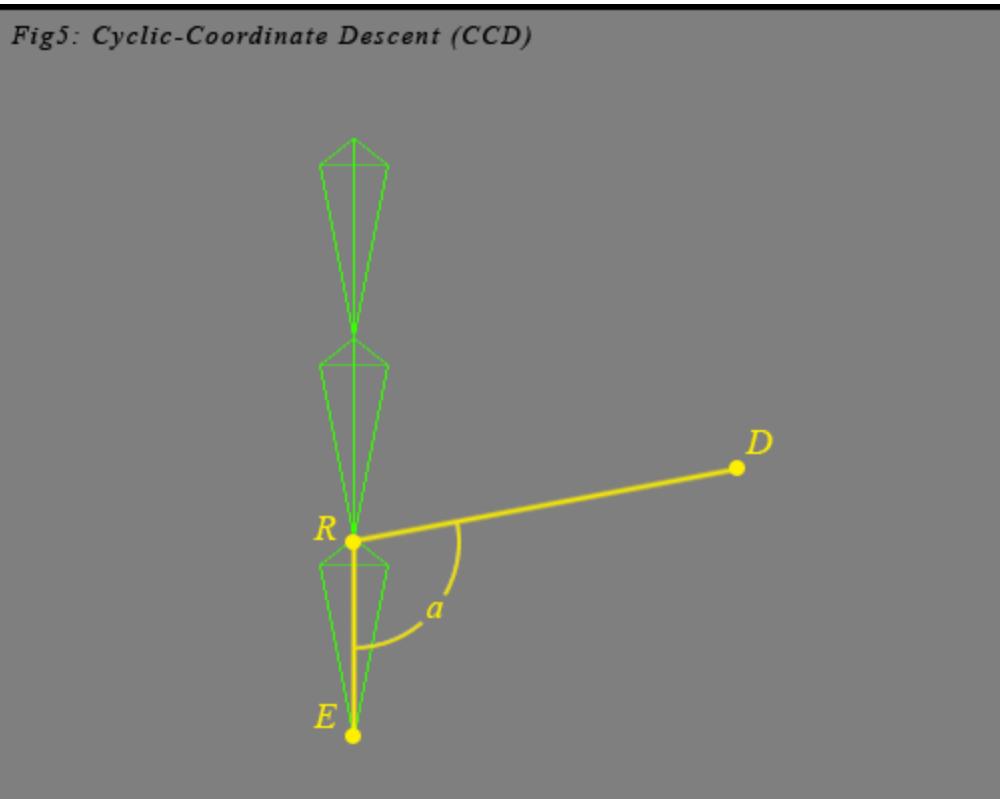
- Attraper un objet, marcher sur le sol...
⇒ Cinématique inverse (IK)

Une solution de cinématique inverse

Cyclic-Coordinate Descent

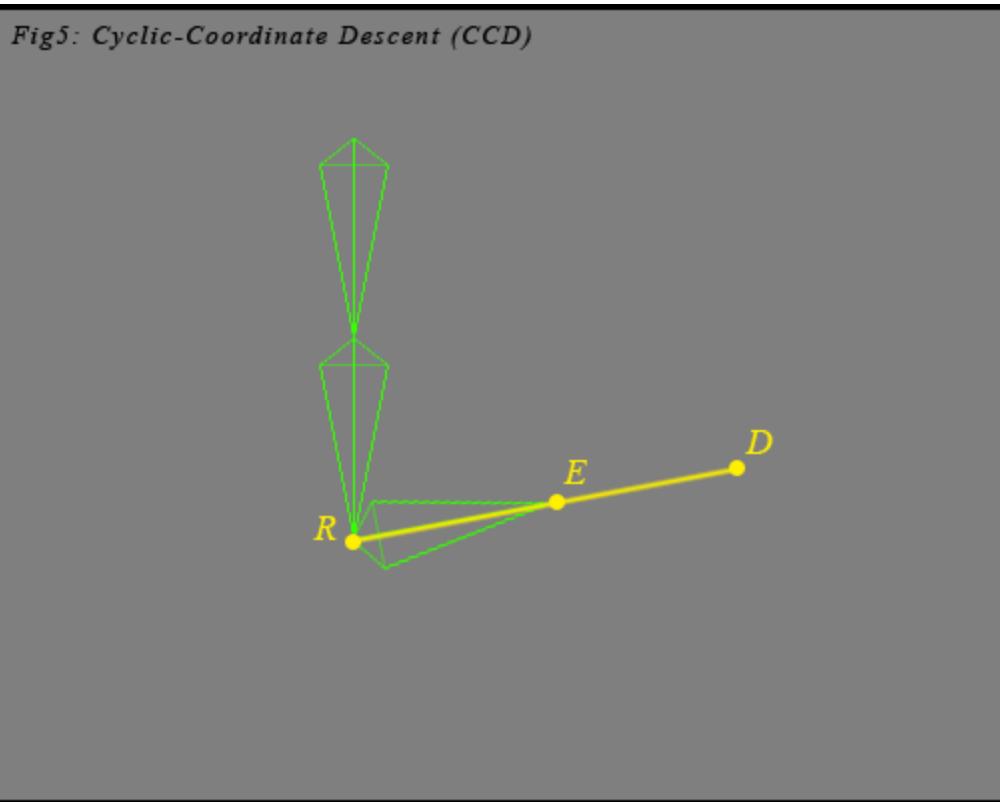
Objectif = E aussi proche de D

- Trouver l'articulation R contenant E
- Dessiner un vecteur de R vers D
- Déduire l'angle du produit scalaire (inv. Cos) : $\cos(a) = \mathbf{RD} \bullet \mathbf{RE}$



Cyclic-coordinate Descent

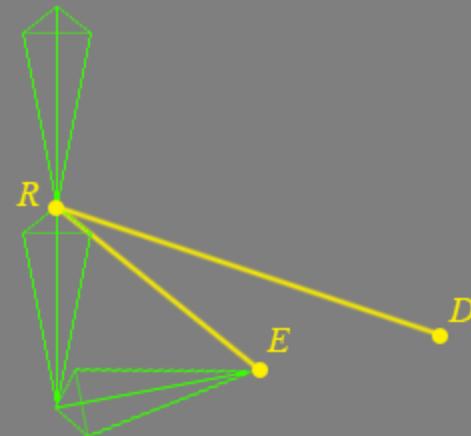
Trouver RE sur RD



Cyclic-coordinate Descent

Monter dans la chaîne et répéter

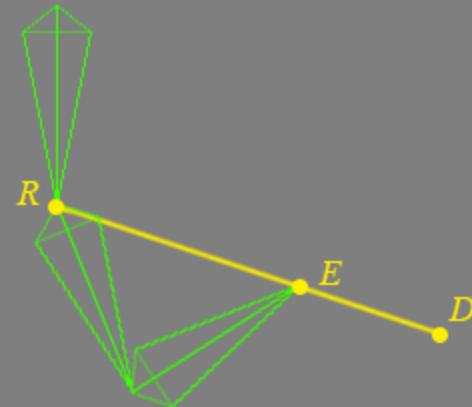
Fig5: Cyclic-Coordinate Descent (CCD)



Cyclic-coordinate Descent

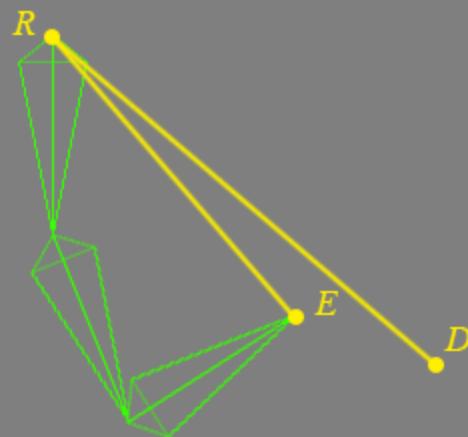
Répéter jusqu'à atteindre la racine,
Itérer jusqu'à ce que E soit suffisement proche de D

Fig5: Cyclic-Coordinate Descent (CCD)



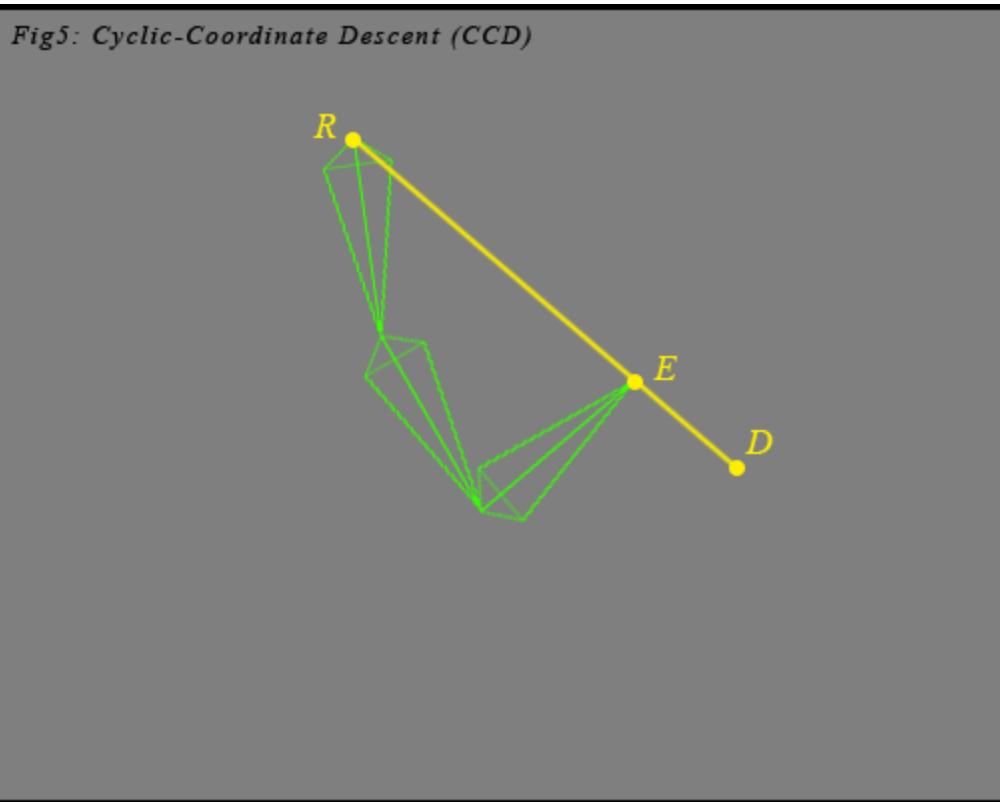
Cyclic-coordinate Descent

Fig5: Cyclic-Coordinate Descent (CCD)



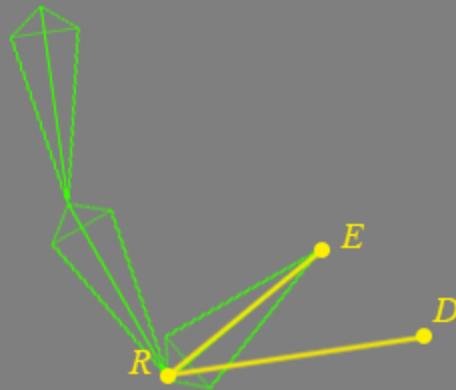
Cyclic-coordinate Descent

On a atteind la racine. Répéter le processus.



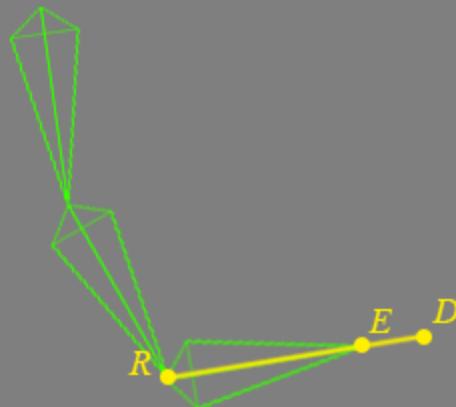
Cyclic-coordinate Descent

Fig5: Cyclic-Coordinate Descent (CCD)



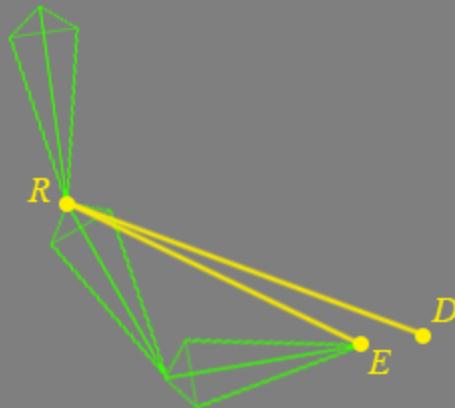
Cyclic-coordinate Descent

Fig5: Cyclic-Coordinate Descent (CCD)



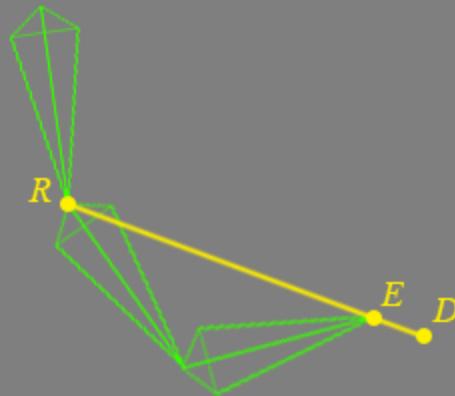
Cyclic-coordinate Descent

Fig5: Cyclic-Coordinate Descent (CCD)



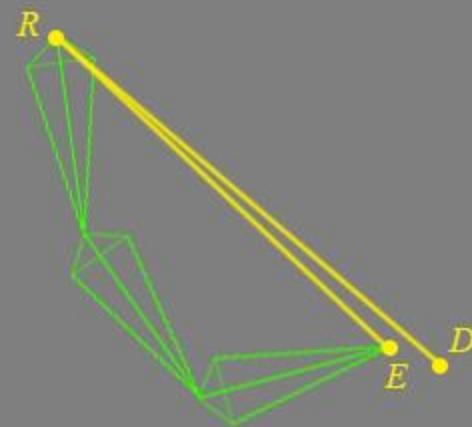
Cyclic-coordinate Descent

Fig5: Cyclic-Coordinate Descent (CCD)



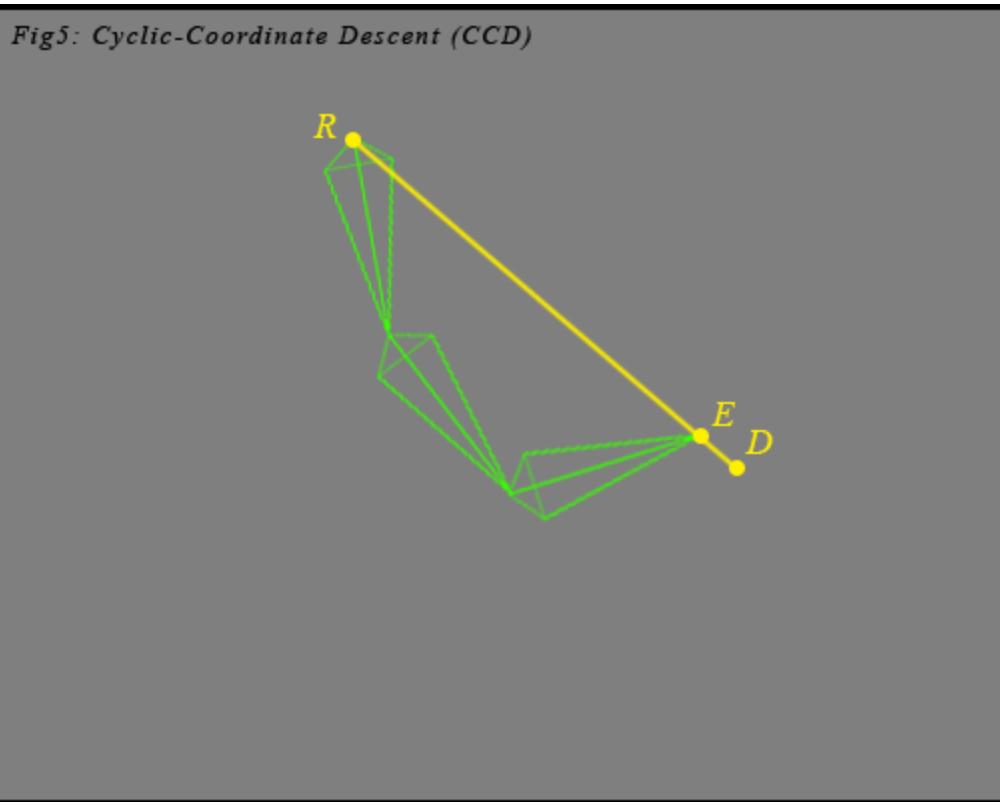
Cyclic-coordinate Descent

Fig5: Cyclic-Coordinate Descent (CCD)



Cyclic-coordinate Descent

Racine de nouveau. Répéter jusqu'à la solution.



Using IK in Game Development

- Examples of IK in action:
 - Character Animation Demo (Softimage XSI 5.0, Blender, Maya, everywhere)
 - Real-Time calculations: E3 2003 Demo Footage of Half-Life 2

Crédits

- Pierre Benard, Jean-Marc Thiery, Steve Marschner, Olivier Vaillancourt, Olivier Godin, Estelle Duveau, Marie-Paule Cani, Lionel Revert, François Faure, Ravi Ramamoorthi, Ronen Barzel, Bill Baxter...

