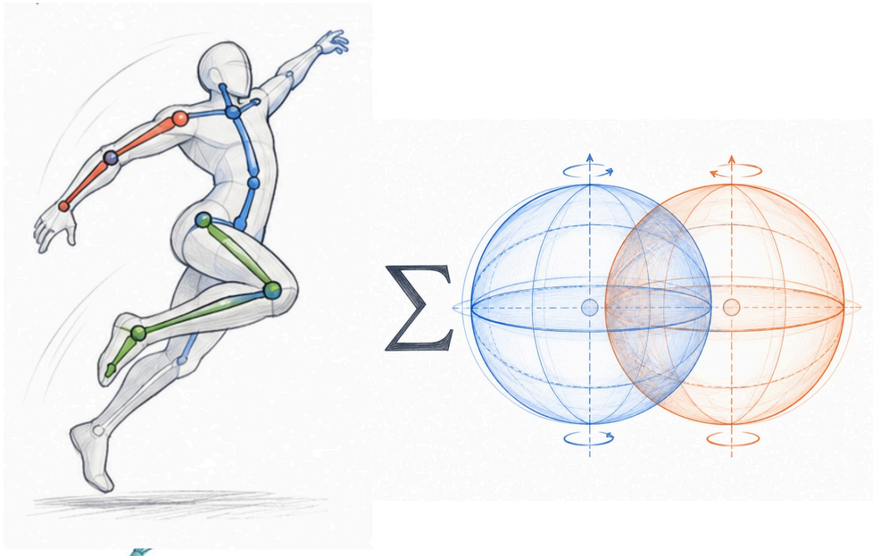
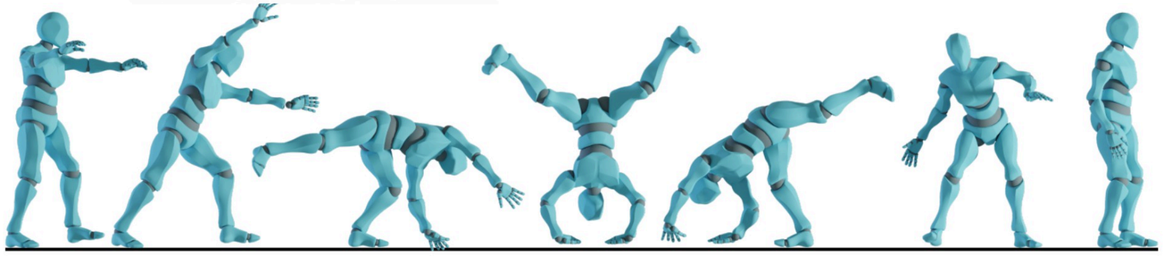
Des rotations aux personnages animés

les mathématiques de l'animation 3D





Damien Rohmer
LIX, École polytechnique
damien.rohmer@polytechnique.edu

Soirées Mathématiques de Lyon
2026

0 min

Informatique graphique

Ensemble des sciences et techniques permettant de *représenter*, *créer* et *interagir* avec des *modèles virtuels*.

1 min



Informatique graphique

Ensemble des sciences et techniques permettant de *représenter*, *créer* et *interagir* avec des *modèles virtuels*.

Avatar : la voie de l'eau, concept art (20th Century Studios, 2020)



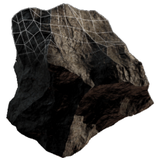
Rendu du ciel

Diffusion de Rayleigh : $I \propto 1/\lambda^4$

le bleu est plus diffusé que le rouge

Avatar : la voie de l'eau, concept art (20th Century Studios, 2020)

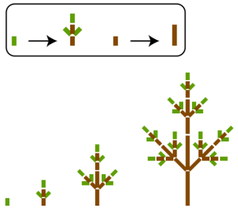
Surfaces procédurales



Bruit fractal

$P(x) = \sum_k \alpha^k f(\omega^k x)$

Végétation procédurale



L-systèmes

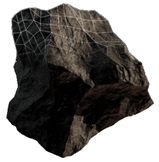
Rendu du ciel

Diffusion de Rayleigh : $I \propto 1/\lambda^4$

le bleu est plus diffusé que le rouge

Avatar : la voie de l'eau, concept art (20th Century Studios, 2020)

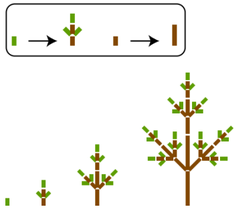
Surfaces procédurales



Bruit fractal

$P(x) = \sum_k \alpha^k f(\omega^k x)$

Végétation procédurale



L-systèmes

Rendu du ciel

Diffusion de Rayleigh : $I \propto 1/\lambda^4$

le bleu est plus diffusé que le rouge

Déformeurs interactifs



$w(p) = e^{-\|p-c\|^2/\sigma^2}$

Simulation de fluides



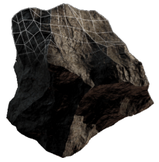
Navier-Stokes

$\nabla \cdot u = 0$

$\partial_t u + (u \cdot \nabla)u = -\frac{1}{\rho}\nabla p + \nu \Delta u + f$

Avatar : la voie de l'eau, concept art (20th Century Studios, 2020)

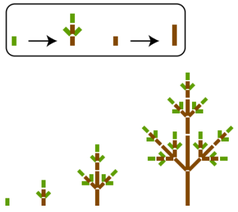
Surfaces procédurales



Bruit fractal

$P(x) = \sum_k \alpha^k f(\omega^k x)$

Végétation procédurale



L-systèmes

Animation de personnages





$p = \left(\sum_j \alpha_j \, \mathrm{T}_j \right) p^0$

Rendu du ciel

Diffusion de Rayleigh : $I \propto 1/\lambda^4$

le bleu est plus diffusé que le rouge

Déformeurs interactifs



$w(p) = e^{-\|p-c\|^2/\sigma^2}$

Simulation de fluides



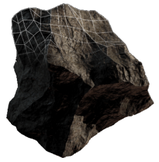
Navier-Stokes

$\nabla \cdot u = 0$

$\partial_t u + (u \cdot \nabla) u = -\frac{1}{\rho} \nabla p + \nu \Delta u + f$

Avatar : la voie de l'eau, concept art (20th Century Studios, 2020)

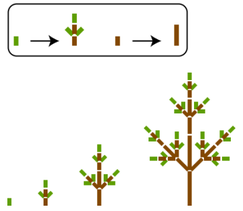
Surfaces procédurales



Bruit fractal

$P(x) = \sum_k \alpha^k f(\omega^k x)$

Végétation procédurale



L-systèmes

Animation de personnages





$p = \left(\sum_j \alpha_j \, \mathrm{T}_j \right) p^0$

Des **algorithmes** et du **calcul** (GPU, temps réel, parallélisme)

Beaucoup de **sciences** : mathématiques, physique, mécanique, optique, statistiques, perception...

Pluridisciplinaire par nature

Rendu du ciel

Diffusion de Rayleigh : $I \propto 1/\lambda^4$

le bleu est plus diffusé que le rouge

Déformeurs interactifs



$w(p) = e^{-\|p-c\|^2/\sigma^2}$

Simulation de fluides



Navier-Stokes

$\nabla \cdot u = 0$

$\partial_t u + (u \cdot \nabla) u = -\frac{1}{\rho} \nabla p + \nu \Delta u + f$

Avatar : la voie de l'eau, concept art (20th Century Studios, 2020)

Ce soir : l'animation de personnages

Rotations · squelettes · skinning · animation expressive

7 min

Plan

1. Historique de l'animation 3D — Des dessins animés aux maillages 3D
2. Interpolation de keyframes — Images clés, courbes et splines
3. Représentation de rotations — Matrices, angles d'Euler, axe-angle, quaternions
4. Interpolation de rotations — Le problème, et ce qu'apporte chaque représentation
5. Animation de personnages — Squelette et skinning
6. Animation et quaternions duaux — Nombres duaux, quaternions duaux, skinning DQS
7. Approches de recherche — Implicit skinning, retargeting, regard, animation expressive et dynamic skinning

7 min

Historique de l'animation 3D

Des dessins animés aux maillages 3D

PARTIE 1 / 7

1 **Historique de l'animation 3D**

2 Interpolation de keyframes

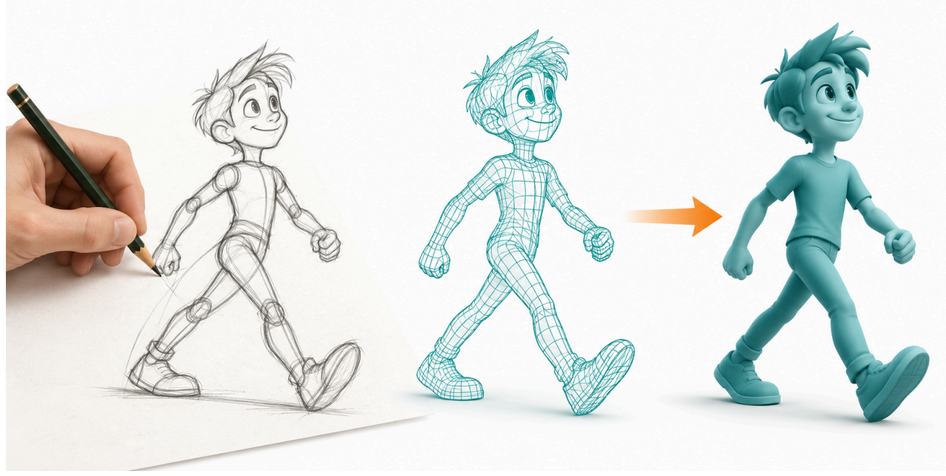
3 Représentation de rotations

4 Interpolation de rotations

5 Animation de personnages

6 Animation et quaternions duaux

7 Approches de recherche



Croquis d’animation 2D · maillage · personnage 3D

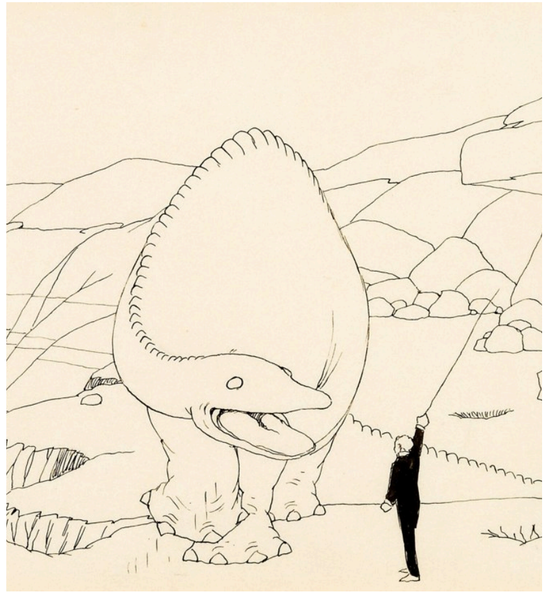
8 min

Historique, Animation 2D : les keyframes

1914

Gertie le dinosaure (W. McCay)

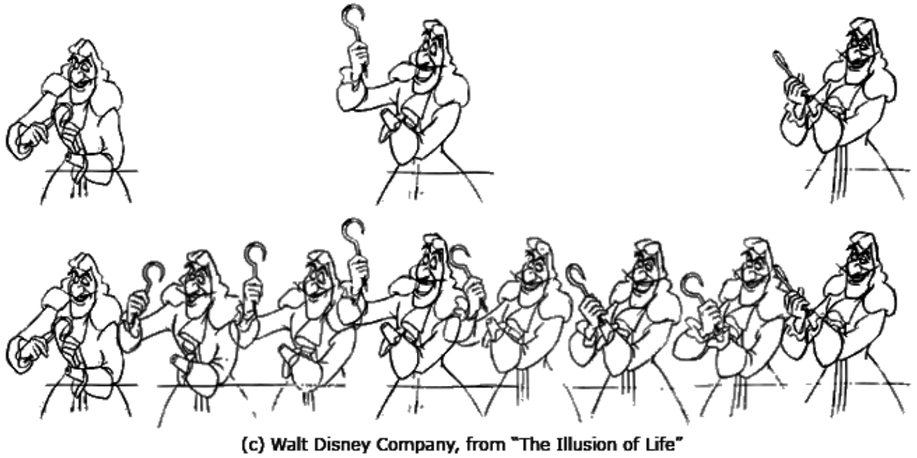
- milliers de dessins à la main
- **poses principales** d'abord
- puis intermédiaires



Années 1930

Studios Disney : division du travail

chef : **poses clés** (*keyframes*) · assistants : **intervalles** · 24 images / s



Poses clés (en haut) et séquence complète (en bas)



1937

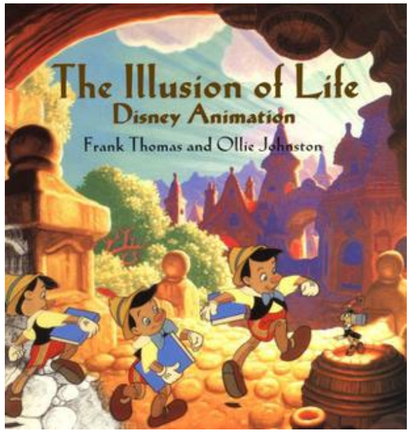
Blanche-Neige

1er long métrage de Disney

1981

The Illusion of Life

- F. Thomas, O. Johnston
- **12 principes** de l'animation



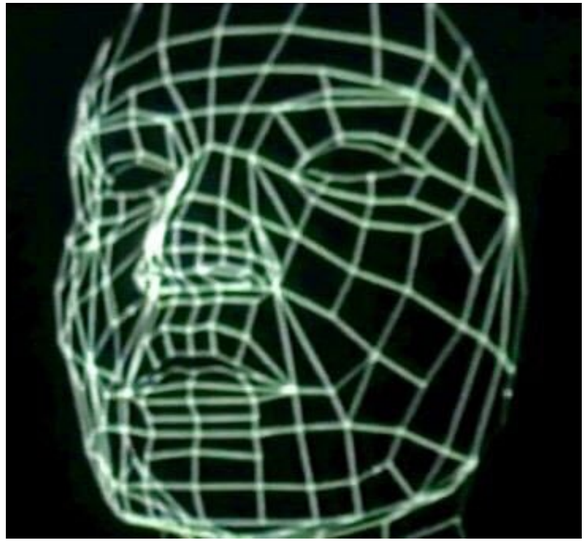
Gertie (dessin original, W. McCay, 1914) : domaine public. Blanche-Neige, The Illusion of Life : © Disney

9 min

Historique, Animation 3D : les squelettes

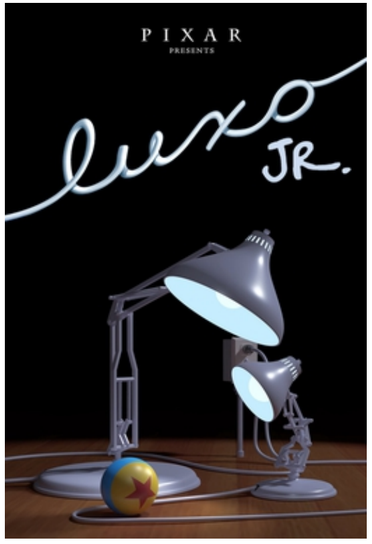
1972

E. Catmull, F. Parke (Utah)
- main et visage animés en 3D
- Catmull cofonde **Pixar**



1986-1987

J. Lasseter (Pixar)
- ancien animateur Disney
- **Luxo Jr.** : nommé aux Oscars
- principes 2D → 3D



1993

Jurassic Park (ILM)
- dinosaures numériques
- fin de l'image par image
- marionnette → **squelette 3D**



1995

Toy Story (Pixar)
- 1er long métrage 100 % 3D
- 114 240 images



⇒ Mêmes principes qu'en 2D, appliqués à des **squelettes 3D**

⇒ Poses clés par l'artiste, intermédiaires **interpolés** par l'ordinateur

Catmull et Parke 1972 ; Luxo Jr., Toy Story : © Disney / Pixar ; Jurassic Park : © Universal

10 min

Représenter une surface : le maillage

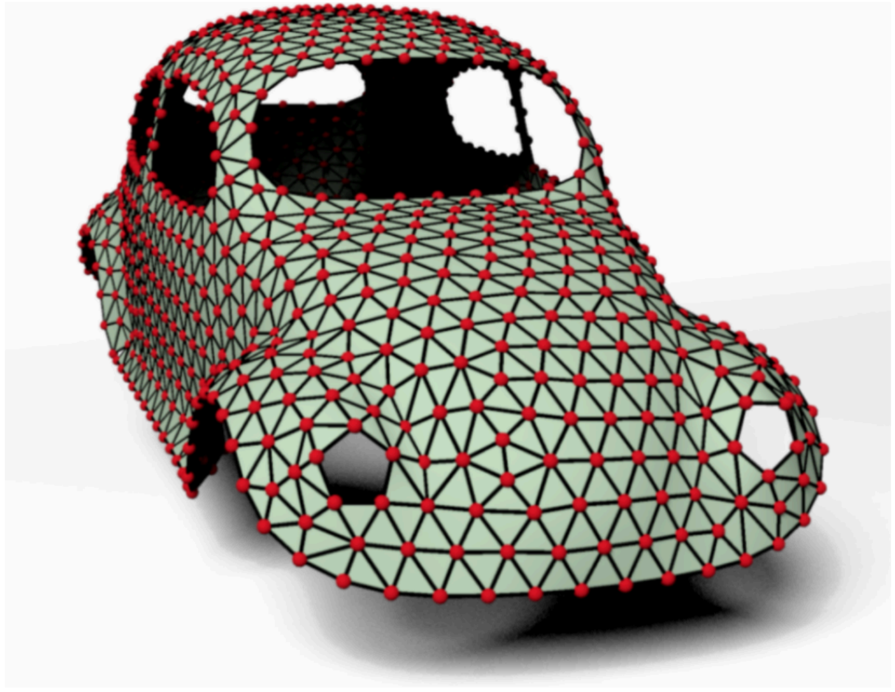
Surface continue $\approx$ ensemble de **triangles** : un *maillage*

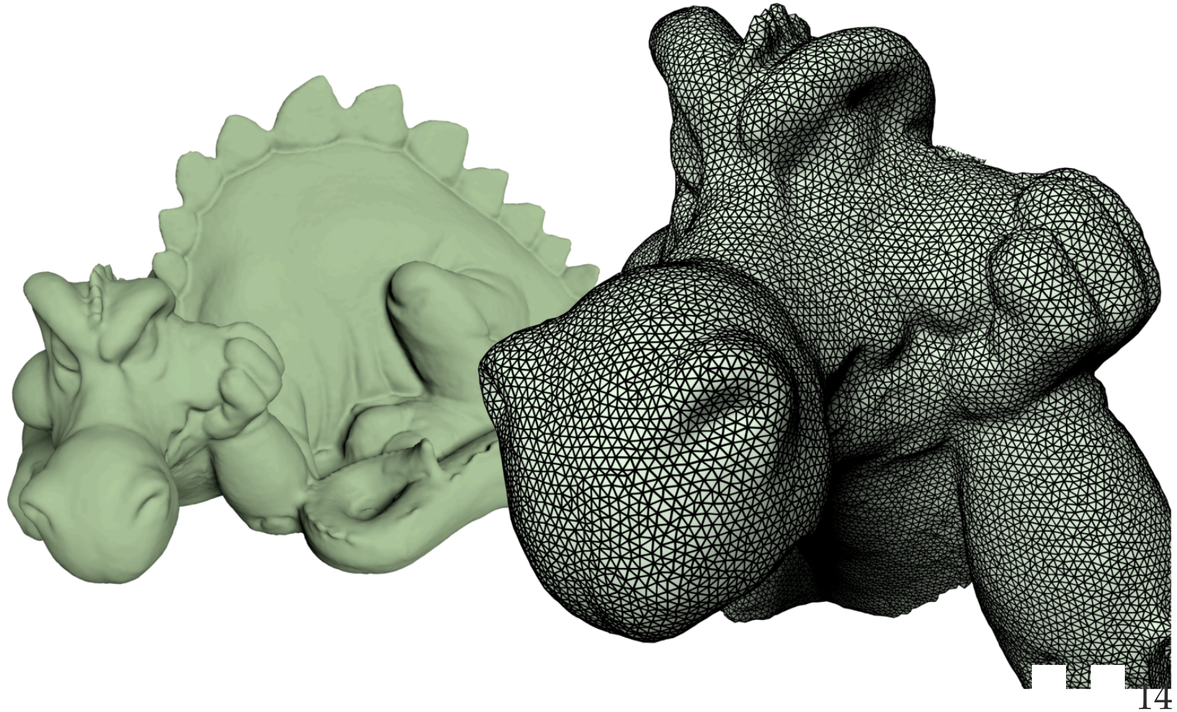
Deux ingrédients

Sommets (*vertices*) : $\mathcal{V} = (p_1, \ldots, p_N)$, $p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$ $\rightarrow$ la **géométrie**

Triangles (*faces*) : $\mathcal{F} = (f_1, \ldots, f_M)$, $f_k = (i, j, l)$ indices de 3 sommets $\rightarrow$ la **connectivité**

Plus de triangles $\Rightarrow$ surface plus fine





11 min

Animer une surface = animer coordonnées des sommets

Sommet par sommet | Par groupes

La connectivité est **fixe** : seules les positions changent, $p_i(t)$

Sommet par sommet : une trajectoire par sommet

simulation physique (tissus), capture de surfaces en mouvement...

Par groupes : même mouvement pour tout un groupe

$p_i(t) = \mathrm{T}_g(t)\, p_i^0$ pour i dans le groupe g

T_g : rotation + translation

450 sommets, 864 triangles : seules les positions bougent

12 min

Interpolation de keyframes

Images clés, courbes et splines

PARTIE 2 / 7

1 Historique de l'animation 3D

2 **Interpolation de keyframes**

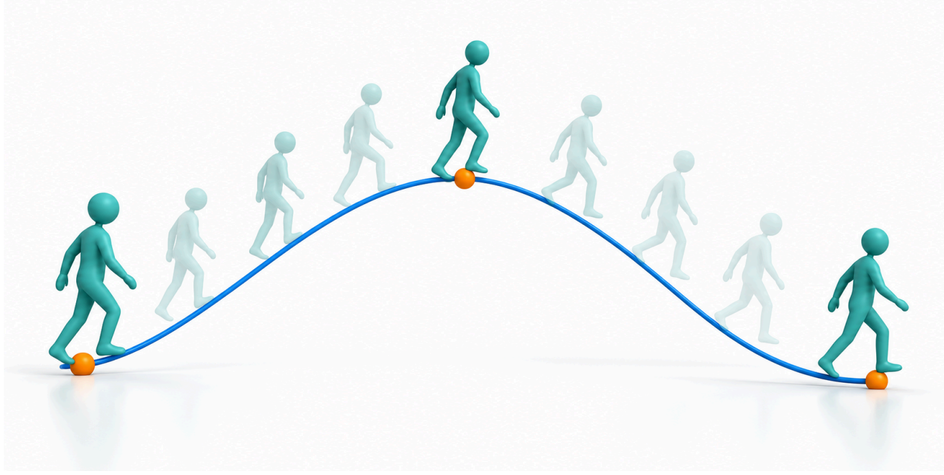
3 Représentation de rotations

4 Interpolation de rotations

5 Animation de personnages

6 Animation et quaternions duaux

7 Approches de recherche

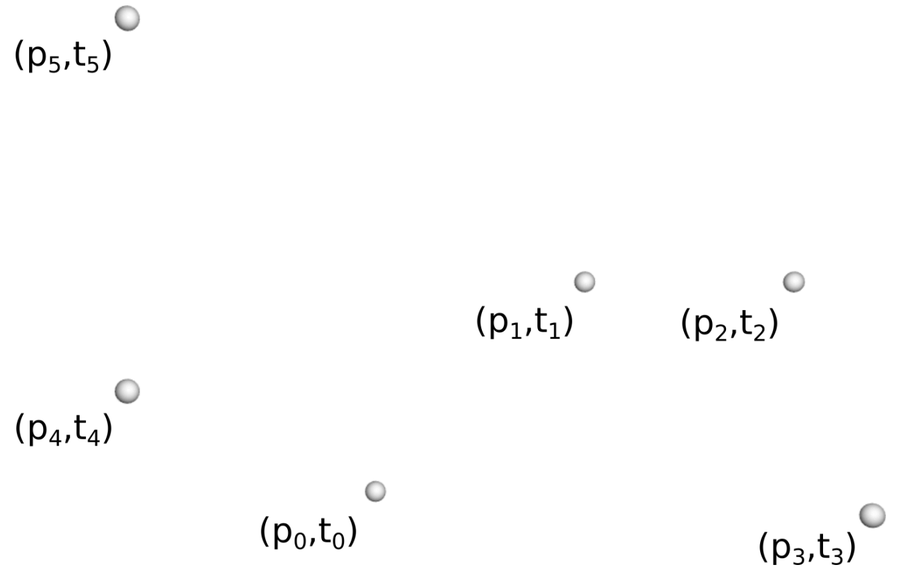


Clés choisies, poses intermédiaires calculées

12 min

Animation par images clés

- L'animateur définit des **poses clés** (position + instant), l'ordinateur calcule une **trajectoire** qui les interpole



- **Entrée** : $(p_i, t_i), i \in [0, N-1]$
- **Sortie** : une courbe espace-temps $p(t), t \in [t_0, t_{N-1}]$ telle que $p(t_i) = p_i$

13 min

Interpoler entre les images clés

Interpolation linéaire

$$p(t) = (1-\alpha)\,p_i + \alpha\,p_{i+1}$$

$$\alpha = \frac{t - t_i}{t_{i+1} - t_i} \in [0,1]$$

(+) simple, **locale**

(−) **angles** aux clés

(−) vitesse **discontinue**

Polynôme de degré $N-1$

$$p(t) = \sum_i \ell_i(t)\,p_i$$

ℓ_i : polynômes de Lagrange

(+) **lisse**

(−) **oscillations**

(−) influence **globale**

Spline (cubique)

$$p(\alpha) = a\,\alpha^3 + b\,\alpha^2 + c\,\alpha + d$$

tangentes $d_i = \dfrac{p_{i+1} - p_{i-1}}{2}$

(+) **lisse** et **locale**

$\Rightarrow$ logiciels 3D



Linéaire | Polynôme | Spline cubique | Réinitialiser

Logiciels d'animation : l'éditeur de courbes

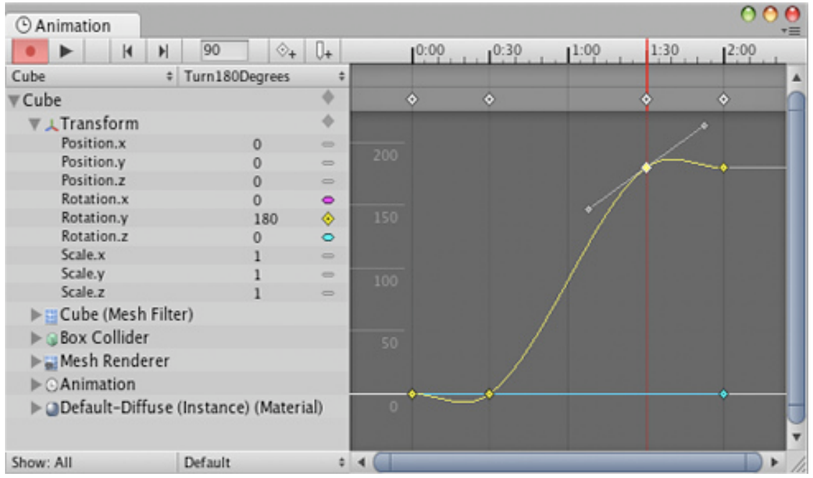
Maya, 3ds Max, Blender... : toujours un **éditeur de courbes**

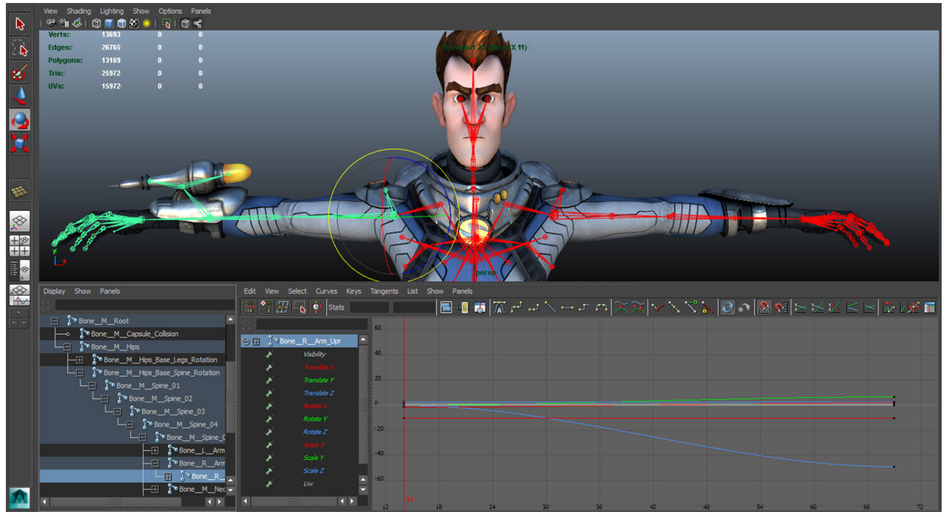
Une courbe par **paramètre scalaire**

position (x, y, z), échelle (s_x, s_y, s_z)

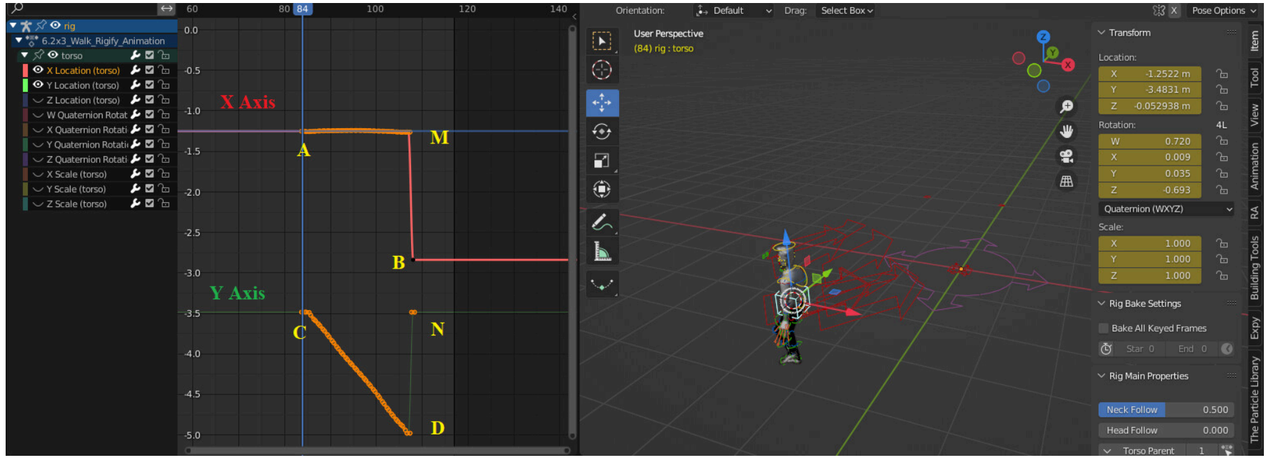
rotation, ex. quaternion (q_x, q_y, q_z, q_w)

L'artiste ajuste les **positions**, les **instants** et les **dérivées** (tangentes)





Maya



Blender

Unity, Maya : © Unity Technologies, © Autodesk. Blender : capture CC0 (Wikimedia Commons)

15 min

Personnage articulé : des blocs en rotation

Corps $\approx$ **blocs rigides** reliés par des **articulations**

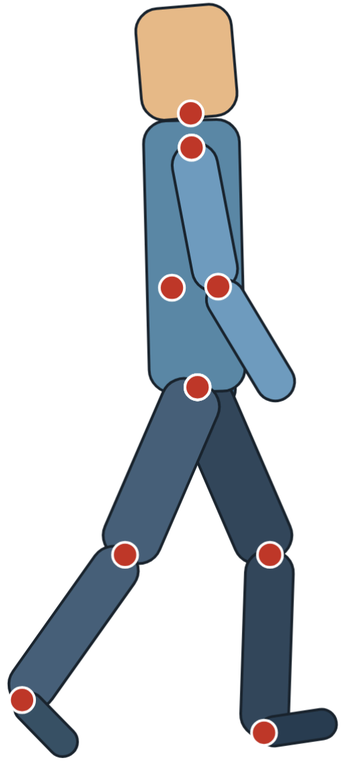
Chaque bloc **tourne** autour de son articulation

épaule, coude, hanche, genou...

Pose = ensemble d'**angles** ; animation = angles au cours du temps

Le mouvement d'un bloc entraîne les suivants

$\Rightarrow$ Animer un personnage, c'est d'abord animer des **rotations**



16 min

Représentation de rotations

Matrices, angles d'Euler, axe-angle, quaternions

PARTIE 3 / 7

1 Historique de l'animation 3D

2 Interpolation de keyframes

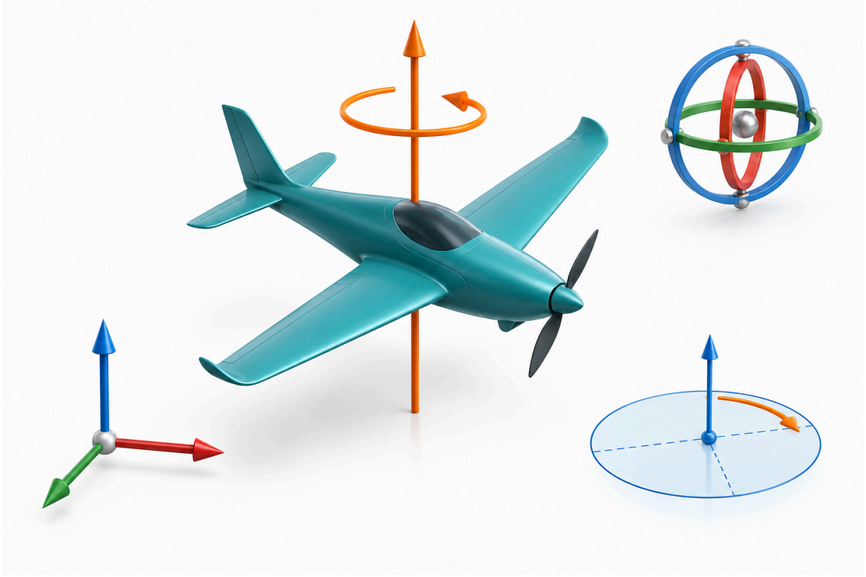
3 **Représentation de rotations**

4 Interpolation de rotations

5 Animation de personnages

6 Animation et quaternions duaux

7 Approches de recherche



Une rotation, plusieurs représentations

16 min

Encoder une rotation

Une bonne représentation doit être

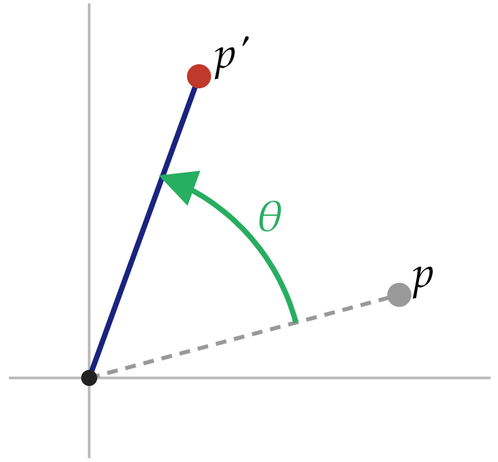
- **simple à comprendre** (pour l'animateur)
- **simple et concise** à stocker
- **interpolable** entre images clés

En 2D : un seul angle θ

$$p' = \mathrm{R}(\theta)\, p, \quad \mathrm{R}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

Interpolation : $\theta(t) = (1-\alpha)\,\theta_0 + \alpha\,\theta_1$

$\Rightarrow$ Et **en 3D** ? Comment représenter une rotation ?



17 min

Représenter une rotation 3D

Une rotation 3D a 3 degrés de liberté, mais il n'existe pas de représentation unique.

Matrice (Rep. 1)

$$R = \begin{pmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{pmatrix}$$

$$R^T R = \mathrm{I}$$

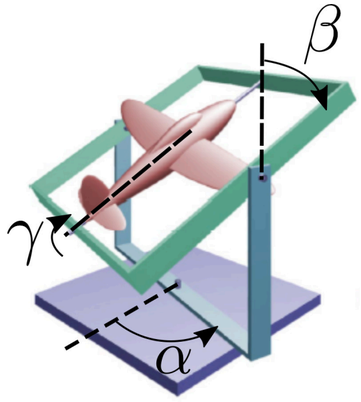
$$\det(R) = 1$$

Angles d'Euler (Rep. 2)

3 angles: (α, β, γ)

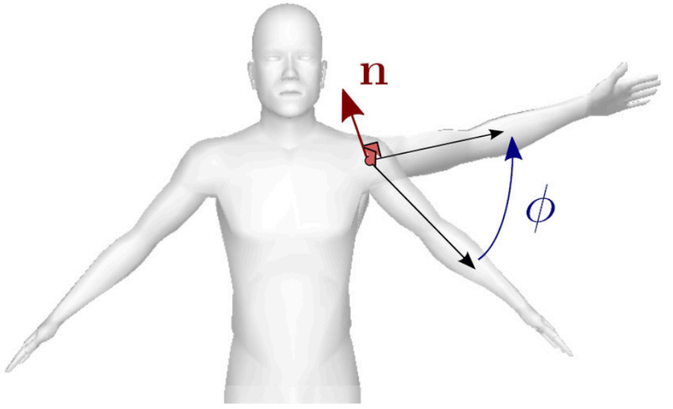
Composition de rotation autour d'axes de bases

Conventions multiples



Axe angle (Rep. 3)

$(\mathbf{n}, \theta)$



Quaternion (Rep. 4)

$$q = (x, y, z, w)$$
$$= (\mathbf{n} \sin \tfrac{\theta}{2}, \cos \tfrac{\theta}{2})$$

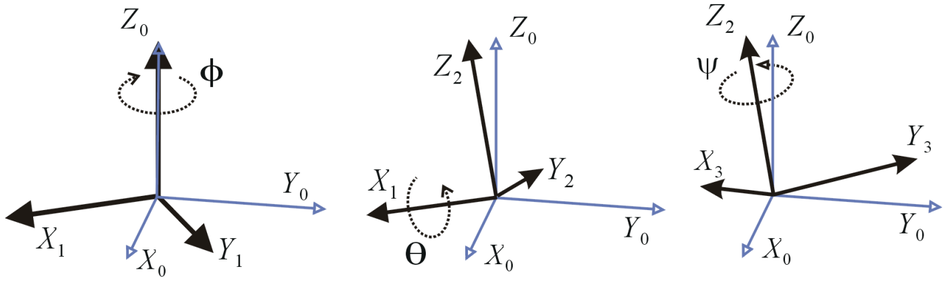
18 min

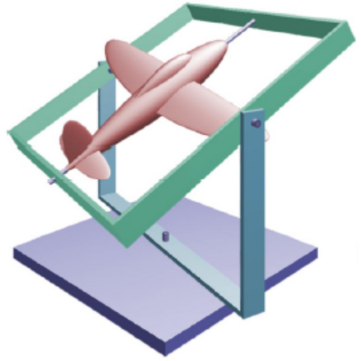
Angles d'Euler (Rep. 2)

Trois rotations successives : autour de x, puis y', puis z''

$$\mathrm{R} = \mathrm{R}_x(\alpha)\,\mathrm{R}_y(\beta)\,\mathrm{R}_z(\gamma)$$

$$\mathrm{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \quad \mathrm{R}_y = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \quad \mathrm{R}_z = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$$





Pro

3 paramètres compréhensibles

Les animateurs éditent une courbe par angle

Largement utilisé en robotique

Attention : conventions multiples

Tait-Bryan : x-y-z, z-y-x, ... Euler : z-x-z, ...

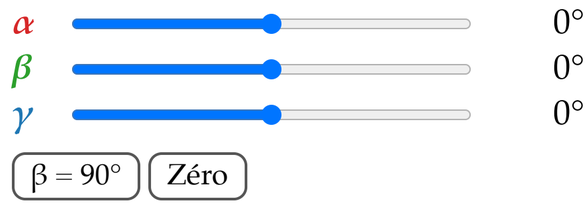
Export/import entre logiciels

19 min

Problème Gimbal lock / blocage de cardan (Rep. 2)

$\mathrm{R} = \mathrm{R}_x(\alpha)\,\mathrm{R}_y(\beta)\,\mathrm{R}_z(\gamma)$

trois anneaux emboîtés



$\beta = 90^\circ$: axes de α et γ alignés

$\Rightarrow$ **un degré de liberté perdu**

Cardan : suspension à anneaux, G. Cardano (XVI^e s.)

1969, Apollo 11, M. Collins :

« How about sending me a fourth gimbal for Christmas »

20 min

Représentation axe-angle (Rep. 3)

Rotation 3D = un axe unitaire n et un angle θ

- Concise : 3 degrés de liberté
- Paramètres compréhensibles

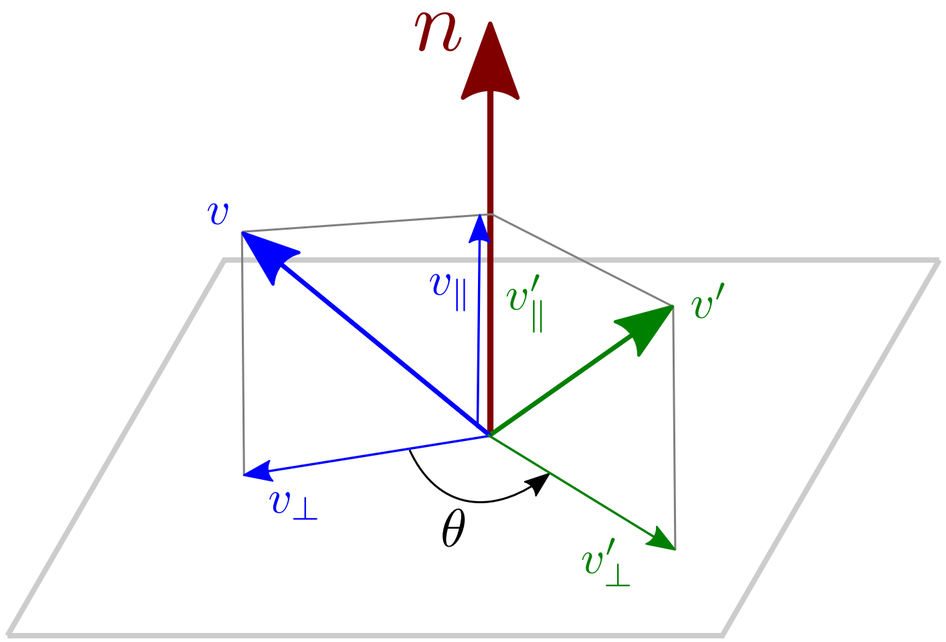
Appliquer la rotation (n, θ) au vecteur v

$$v = v_{\parallel} + v_{\perp}, \quad v_{\parallel} = (v \cdot n)\, n$$

$$v' = v_{\parallel} + \cos(\theta)\, v_{\perp} + \sin(\theta)\, n \times v_{\perp}$$

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

Formule de **Rodrigues**



21 min

Quaternions (Rep. 4)

Généralisation des nombres complexes (W. R. Hamilton, 1843)

$q = x\,\mathbf{i} + y\,\mathbf{j} + z\,\mathbf{k} + w$ noté $q = (x, y, z, w)$

w partie réelle, (x, y, z) partie imaginaire ; quaternion pur si $w = 0$

$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1, \quad \mathbf{ij} = -\mathbf{ji} = \mathbf{k}$, ...

Conjugué $q^\star = (-x, -y, -z, w)$, norme $\|q\| = \sqrt{x^2 + y^2 + z^2 + w^2}$

Produit $q_1\ q_2$: associatif, **non commutatif** (comme les matrices)

Quaternions et rotations (Rep. 4)

Rotation d'angle θ autour de l'axe $n = (n_x, n_y, n_z)$, $\|n\| = 1$

$$q = \left(n_x \sin\tfrac{\theta}{2},\ n_y \sin\tfrac{\theta}{2},\ n_z \sin\tfrac{\theta}{2},\ \cos\tfrac{\theta}{2}\right) = \left(n \sin\tfrac{\theta}{2},\ \cos\tfrac{\theta}{2}\right)$$

Appliquer la rotation à un vecteur v (q unitaire)

$v \to$ quaternion pur $q_v = (v_x, v_y, v_z, 0)$

$q\ q_v\ q^\star = (v'_x, v'_y, v'_z, 0) \to v'$: vecteur tourné

$q\ q_v\ q^\star$ redonne la formule de **Rodrigues** (Rep. 3)

22 min

Synthèse : quaternions et rotations (Rep. 4)

	Espace 3D	**Quaternions**
Vecteur	$v = (v_x, v_y, v_z)$	$q_v = (v_x, v_y, v_z, 0)$
Rotation	R (matrice 3×3)	$q = (x, y, z, w)$, $\|q\| = 1$
Appliquer à un vecteur	$R\,v$	$q\ q_v\ q^\star$
Composer deux rotations	$R_1\,R_2$	$q_1\,q_2$

Axe n, angle $\theta \Rightarrow q = (n \sin\frac{\theta}{2},\ \cos\frac{\theta}{2})$

$$q = (x, y, z, w) \Rightarrow R = \begin{pmatrix} 1 - 2(y^2 + z^2) & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & 1 - 2(x^2 + z^2) & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & 1 - 2(x^2 + y^2) \end{pmatrix}$$

Valable uniquement pour des quaternions unitaires

23 min

Interpolation de rotations

Le problème, et ce qu'apporte chaque représentation

PARTIE 4 / 7

1 Historique de l'animation 3D

2 Interpolation de keyframes

3 Représentation de rotations

4 Interpolation de rotations

5 Animation de personnages

6 Animation et quaternions duaux

7 Approches de recherche



Interpoler les orientations — la sphère est une analogie géométrique

24 min

Interpoler des rotations : le problème

Deux images clés : orientations $\mathbf{R}_1$ et $\mathbf{R}_2$

Chercher $\mathbf{R}(t)$, $t \in [0, 1]$, avec $\mathbf{R}(0) = \mathbf{R}_1$, $\mathbf{R}(1) = \mathbf{R}_2$

- une **rotation** pour tout t
- le chemin **le plus court**, à **vitesse angulaire constante**

En 2D : interpoler l'angle θ ; en 3D : dépend de la représentation

R_1 (départ) *?* *R_2 (arrivée)*

25 min

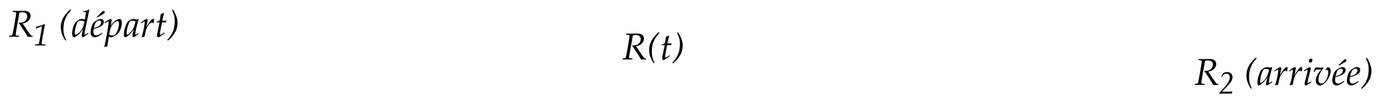
Interpoler des matrices (Rep. 1)

Interpolation coefficient par coefficient : $\mathrm{M}(t) = (1-t)\,\mathrm{R}_1 + t\,\mathrm{R}_2$

$\mathrm{M}(t)$ n'est **pas une rotation** : $\mathrm{M}\,\mathrm{M}^T \neq \mathrm{I}$, $\det \mathrm{M} \leq 1 \Rightarrow$ échelle, cisaillement

$\Rightarrow$ Ne **pas** interpoler une matrice de rotation coefficient par coefficient

Matrices
Euler
Axe-angle
Quaternions LERP
Quaternions SLERP



det = 0.15 : ce n'est plus une rotation (l'aigle se déforme)

25 min

Interpoler des angles d'Euler ou en axe-angle (Rep. 2, 3)

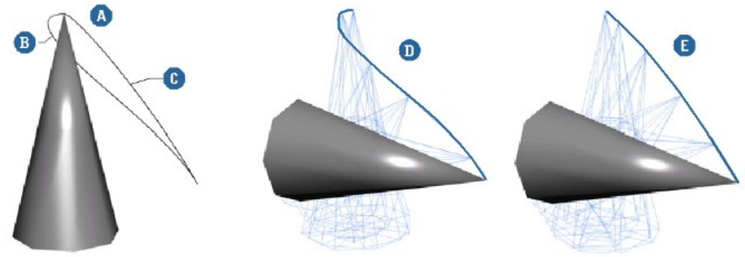
Interpolation linéaire des 3 angles : $\big(\alpha(t), \beta(t), \gamma(t)\big)$

(+) toujours une rotation, simple

(−) **fait un détour**, vitesse non constante

(−) dépend de la convention, blocage de cardan

Axe-angle : interpoler n et θ, **même détour**



- Matrices
- Euler
- Axe-angle
- Quaternions LERP
- Quaternions SLERP

R_1 *(départ)* $R(t)$ R_2 *(arrivée)*

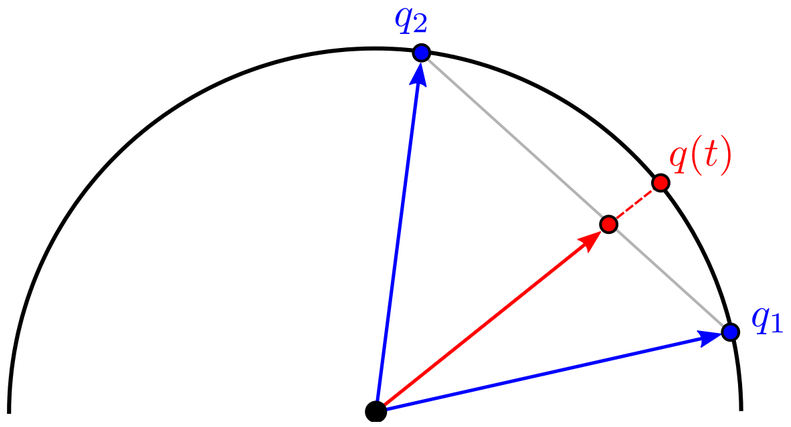
26 min

Interpolation de quaternions : LERP (Rep. 4)

Interpolation linéaire (avec normalisation) dans l'espace des quaternions (**LERP**)

- Considérons deux rotations avec les quaternions correspondants q_1, q_2.
- Le quaternion *interpolé linéairement* au paramètre $t \in [0, 1]$ est

$$q(t) = \frac{(1-t)\, q_1 + t\, q_2}{\|(1-t)\, q_1 + t\, q_2\|}$$



(+) Chemin le plus court (grand cercle de la sphère 4D)
(+) Se généralise aux courbes (splines...)

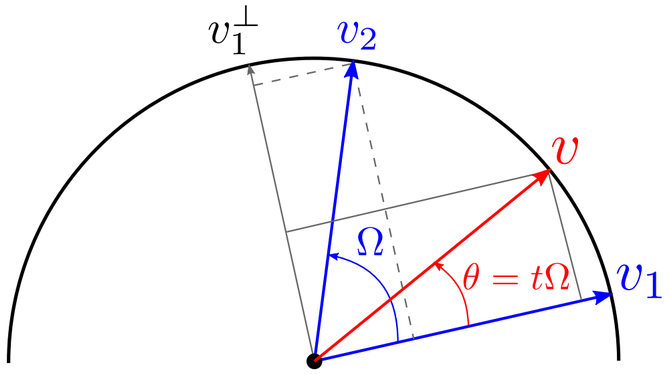
(—) Vitesse angulaire non constante
d'autant plus que l'angle entre les clés est grand

Interpolation de quaternions : SLERP (Rep. 4)

q_1, q_2 unitaires : points de la sphère unité de $\mathbb{R}^4$

Interpolation linéaire sphérique (**SLERP**), $t \in [0, 1]$

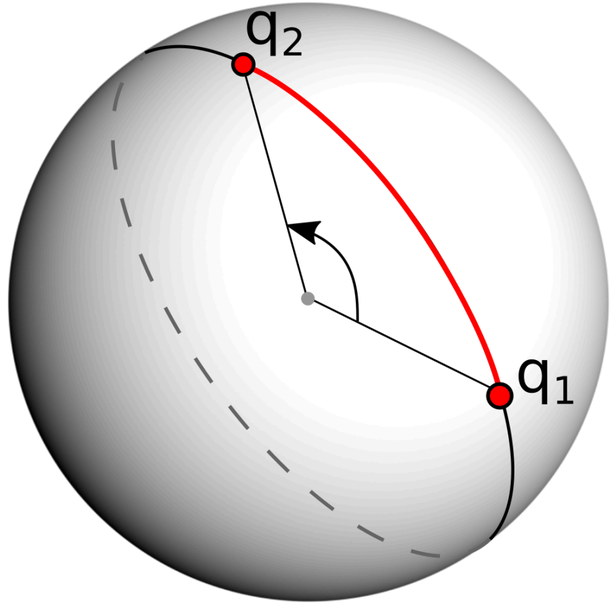
$$q(t) = \frac{\sin((1-t)\Omega)}{\sin(\Omega)}\, q_1 + \frac{\sin(t\Omega)}{\sin(\Omega)}\, q_2 \qquad \cos(\Omega) = q_1 \cdot q_2$$



Pourquoi ? q à l'angle $\theta = \Omega t$ dans le plan (q_1, q_2)

$$q = q_1 \cos\theta + q_1^\perp \sin\theta, \qquad q_1^\perp = \frac{q_2 - \cos(\Omega)\, q_1}{\sin(\Omega)}$$

$$\Rightarrow q = \frac{\sin(\Omega - \theta)}{\sin(\Omega)}\, q_1 + \frac{\sin\theta}{\sin(\Omega)}\, q_2$$



(+) Chemin le plus court (grand cercle de la sphère 4D)

(+) Vitesse angulaire constante

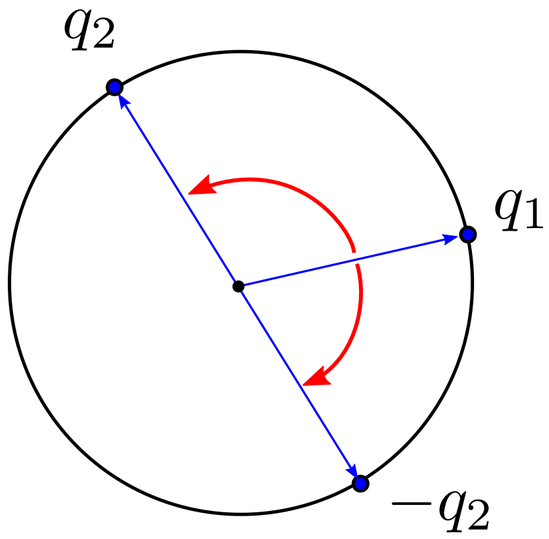
(−) Seulement entre deux quaternions

Attention à −q (Rep. 4)

$+q$ et $-q$ correspondent à la même rotation ($n \to -n, \theta \to 2\pi - \theta$)

Attention $-q$ ***ne correspond pas*** *à la matrice de rotation* $-R$.

Mais à un **chemin différent** lors de l'interpolation en 4D



$\Rightarrow$ le chemin $q_1 \to -q_2$ est plus court que $q_1 \to q_2$ lorsque $q_1 \cdot q_2 < 0$.

En pratique : choisir le chemin le plus court avant SLERP

Algorithme

```
if( dot(q1,q2)<0 )
    q2 = -q2

q(t) = SLERP(q1,q2,t)
```



29 min

Interpoler des rotations : bilan

Matrice
(Rep. 1)

$$R = \begin{pmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{pmatrix}$$

(+) Calcul simple

(−) Redondant

(−) Interpolation : pas une rotation

Angles d'Euler
(Rep. 2)

3 angles: (α, β, γ)

(+) Intuitifs

(−) Gimbal-lock

(−) Interpolation : détour

Axe angle
(Rep. 3)

$(\mathbf{n}, \theta)$

(+) Intuitifs

(−) Composition difficile

(−) Interpolation : détour

Quaternion
(Rep. 4)

$q = (x, y, z, w)$
$= (\mathbf{n} \sin \frac{\theta}{2}, \cos \frac{\theta}{2})$

(+) Composition

(+) SLERP : chemin court, vitesse constante

(−) Peu intuitifs

(−) q et $-q$: même rotation

30 min

Animation de personnages

Squelette et skinning

PARTIE 5 / 7

1 Historique de l'animation 3D

2 Interpolation de keyframes

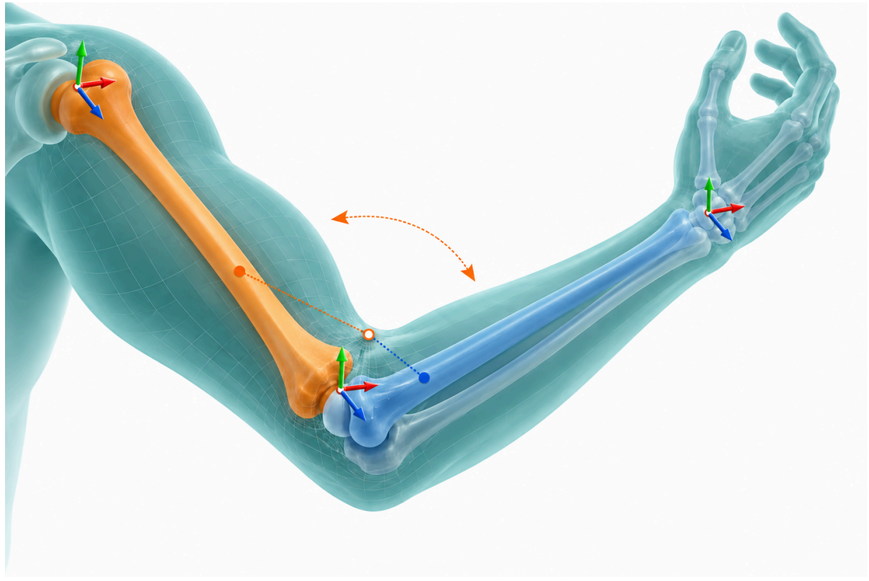
3 Représentation de rotations

4 Interpolation de rotations

5 **Animation de personnages**

6 Animation et quaternions duaux

7 Approches de recherche



Repères aux articulations · influence sur les sommets

30 min

Animer avec un squelette

Objectif : Déformer un personnage articulé

Idée : Utiliser un squelette pour contrôler les membres

Articulations comme rotations

Squelette d'animation

Ensemble de repères $\mathbf{M}_i$: position, orientation

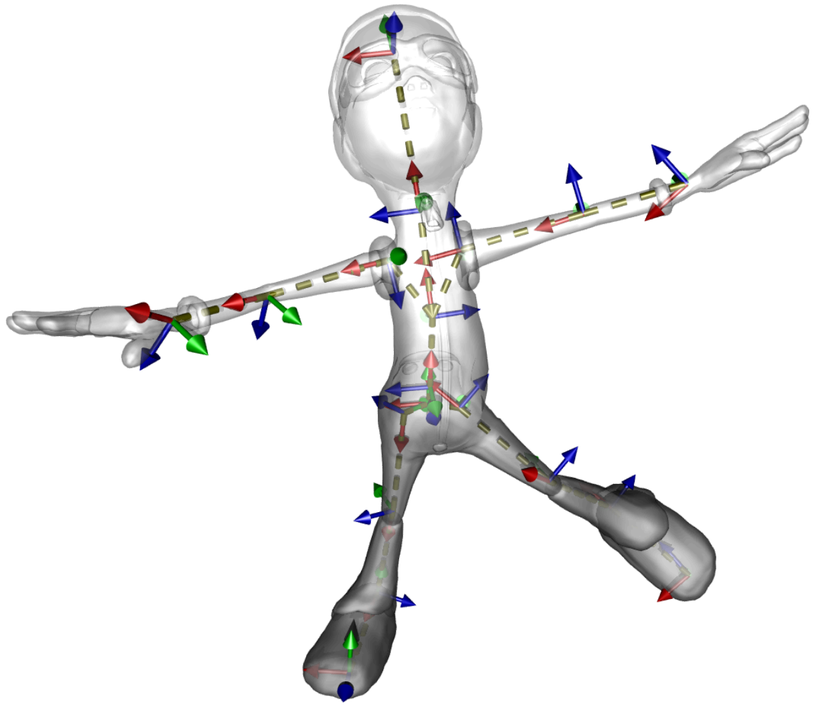
Décrire les parties non rigides du personnage (degrés de liberté)

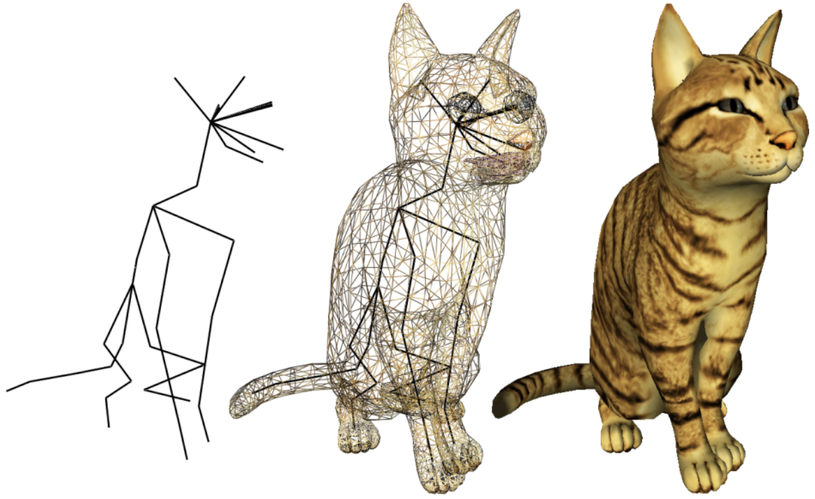
Terminologie

Articulation = Un repère $\mathbf{M}_i$

Os = Segment entre deux articulations

Note : Squelette d'animation $\neq$ Squelette anatomique

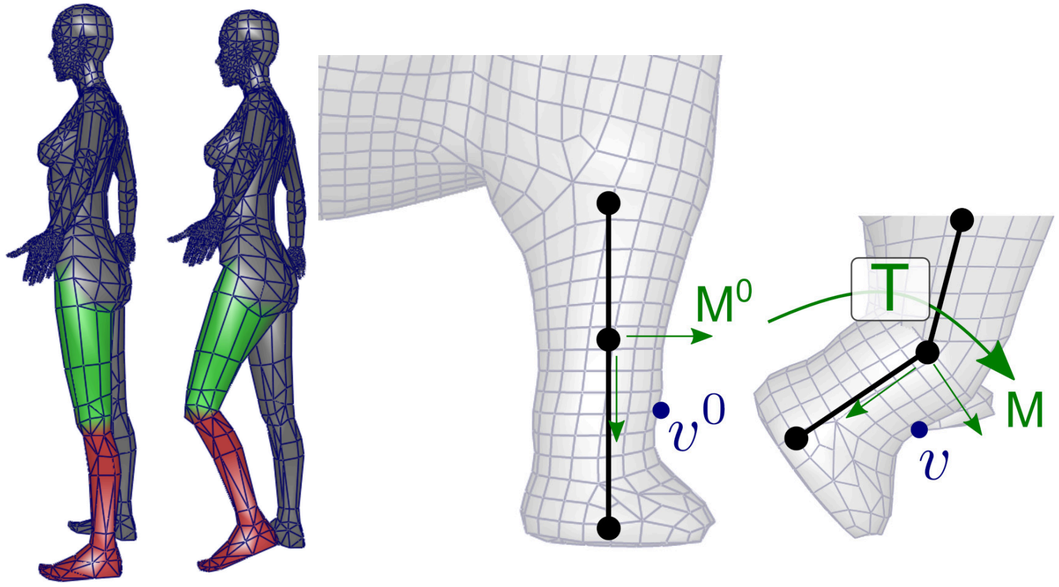




Skinning rigide

Attache rigide d'un sous-ensemble de sommets à des os spécifiques, décrits par le repère de l'articulation.

Les sommets suivent la déformation rigide de leur repère.



Formulation de la déformation

Considérons, en pose de repos (bind pose)

- Un repère M^0
- Un sommet p_i^0 attaché à ce repère

Après déformation

- Nouveau repère M
- Nouvelle position du sommet p_i

Question : Quelles sont les nouvelles coordonnées p_i par rapport à M, M^0, p_i^0 ?

p_i^0 et p_i ont des coordonnées locales identiques par rapport à M^0 et M

$$\Rightarrow \mathrm{M}^{-1} p_i = (\mathrm{M}^0)^{-1} p_i^0$$

$$\Rightarrow p_i = \mathrm{M}\,(\mathrm{M}^0)^{-1} p_i^0 = \mathrm{T}\, p_i^0$$

Aparté : coordonnées homogènes

Repère M : rotation R (3×3) et translation t

$v' = R\,v + t$: un produit et une somme

Ajouter un 1 aux vecteurs : $v = (x, y, z) \;\rightarrow\; (x, y, z, 1)$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \left(\begin{array}{ccc|c} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \qquad \left(\begin{array}{c|c} R & t \\ \hline 0 & 1 \end{array}\right) \begin{pmatrix} v \\ 1 \end{pmatrix} = \begin{pmatrix} R\,v + t \\ 1 \end{pmatrix}$$

Une seule matrice 4×4, un seul produit matrice-vecteur

$\mathrm{T} = \mathrm{M}\,(\mathrm{M}^0)^{-1}$: aussi une matrice 4×4

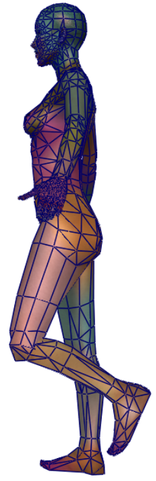
Skinning rigide : limites

(+) Le squelette est facile à construire

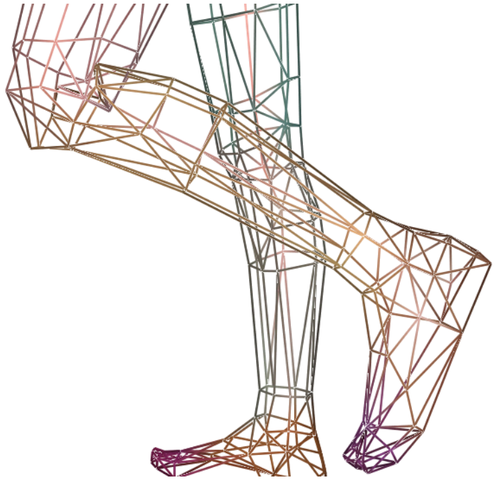
(+) L'interaction du squelette est intuitive pour modéliser une articulation rigide

(—) Discontinuités / Interpénétrations

Jeux vidéo des années 1990 : membres rigides, articulations visibles











Virtua Fighter (Sega, 1993) — Tomb Raider (Core Design / Eidos, 1996)

Idée du skinning lisse : mélanger les transformations discontinues autour de l'articulation

Linear Blend Skinning (LBS)

$$p_i = \left(\sum_{j=0}^{N-1} \alpha_{ij} \, \mathrm{T}_j \right) p_i^0 = \left(\sum_{j=0}^{N-1} \alpha_{ij} \, \mathrm{M}_j \, (\mathrm{M}_j^0)^{-1} \right) p_i^0$$

$\forall j \ , \ \alpha_{ij} \in [0,1]$, et $\sum_{j=0}^{N-1} \alpha_{ij} = 1$

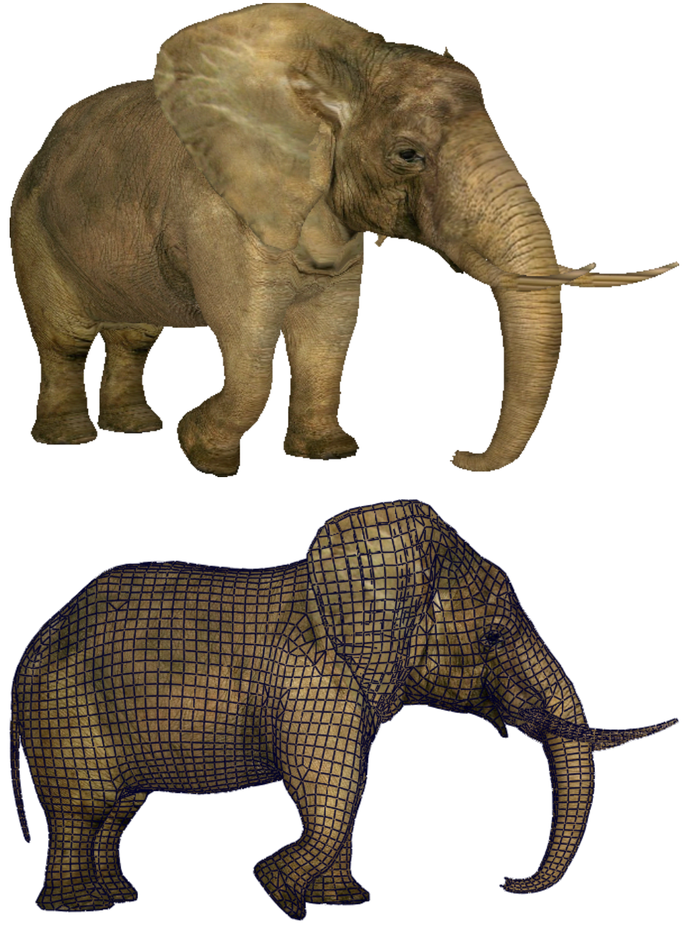
M_j^0 : repère de l'articulation j dans la pose de repos (bind pose)

La **norme** pour déformer les personnages articulés

- Déformation intuitive
- Forme contrôlable (*via les poids*)
- Rapide à calculer (compatible GPU)

  moyenne de matrices, multiplication matrice-vecteur

$\Rightarrow$ Cinéma & jeux vidéo



[Joint-Dependent Local Deformations for Hand Animation and Object Grasping. N. Magnenat-Thalmann, R. Laperrière, D. Thalmann. Graphics Interface, 1988]

[Over My Dead, Polygonal Body. J. Lander. Game Developer Magazine, 1999]

LBS : le volume s'écrase aux plis et en torsion

35 min

Linear Blend Skinning : limites

- Paramètres de rigging non triviaux
- Artéfacts pour de grandes rotations : Candy wrapper, Collapsing elbow



Candy wrapper

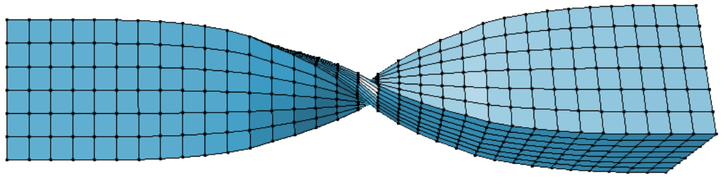


Coude qui s'effondre

Question : d'où viennent ces artéfacts ?

LBS : oups...

Interpolation de rotations : coefficient par coefficient ? **Surtout pas !**



Skinning LBS avec 2 os : $p_i = (\alpha_1 \, \mathrm{T}_1 + \alpha_2 \, \mathrm{T}_2) \, p_i^0$

$$\alpha_1 \underbrace{\left(\begin{array}{c|c} \mathrm{R}_1 & t_1 \\ \hline 0 & 1 \end{array}\right)}_{\mathrm{T}_1} + \alpha_2 \underbrace{\left(\begin{array}{c|c} \mathrm{R}_2 & t_2 \\ \hline 0 & 1 \end{array}\right)}_{\mathrm{T}_2} = \left(\begin{array}{c|c} \alpha_1 \, \mathrm{R}_1 + \alpha_2 \, \mathrm{R}_2 & \alpha_1 \, t_1 + \alpha_2 \, t_2 \\ \hline 0 & 1 \end{array}\right)$$

$\alpha_1 \, \mathrm{R}_1 + \alpha_2 \, \mathrm{R}_2$: moyenne **coefficient par coefficient** de rotations $\Rightarrow$ **pas une rotation**

On vient de faire **exactement** ce qu'il ne fallait pas faire

. . . et c'est la norme au cinéma et dans le jeu vidéo depuis 30 ans

Animation et quaternions duaux

Nombres duaux, quaternions duaux, skinning DQS

PARTIE 6 / 7

1 Historique de l'animation 3D

2 Interpolation de keyframes

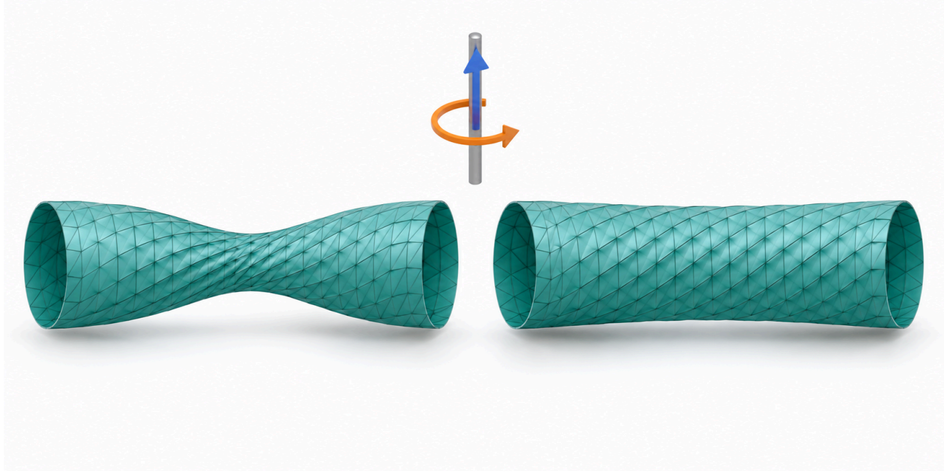
3 Représentation de rotations

4 Interpolation de rotations

5 Animation de personnages

6 **Animation et quaternions duaux**

7 Approches de recherche



À gauche : LBS · à droite : DQS — exemple de torsion

38 min

Idée naïve : séparer rotation et translation

$$\mathrm{T}_j = \left(\begin{array}{c|c} \mathrm{R}_j & t_j \\ \hline 0 & 1 \end{array}\right) \longrightarrow q_j \text{ (quaternion)}, \quad t_j$$

Translation : mélange linéaire, $t = \sum_j \alpha_{ij}\, t_j$

Rotation : mélange de quaternions, $q = \dfrac{\sum_j \alpha_{ij}\, q_j}{\left\| \sum_j \alpha_{ij}\, q_j \right\|}$ (LERP normalisé)

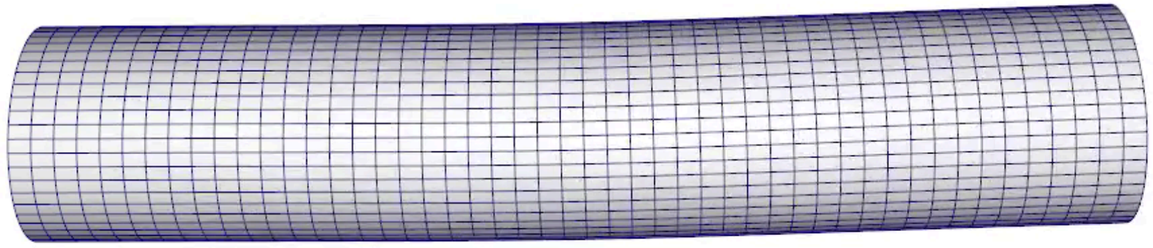
On recompose : $\mathrm{T} = \left(\begin{array}{c|c} \mathrm{R}(q) & t \\ \hline 0 & 1 \end{array}\right)$: une vraie **transformation rigide**

$\Rightarrow$ Plus de candy wrapper ?

Le résultat dépend du centre de rotation choisi

Rotation et translation mélangées **séparément** $\Rightarrow$ résultat dépendant d'un **centre de rotation** arbitraire

$\Rightarrow$ Déformations aberrantes : le remède est pire que le mal





[Correctif possible : centre de rotation optimal à chaque image. Spherical Blend Skinning. L. Kavan, J. Zara. I3D 2005.*]*

Améliorer le skinning : quaternions duaux

Deuxième idée : Utilisation du **quaternion dual**

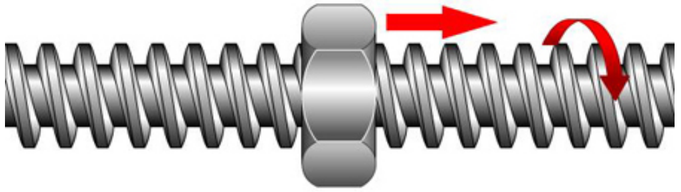
$\hat{q} = q_0 + \epsilon\, q_\epsilon$, ϵ élément dual.

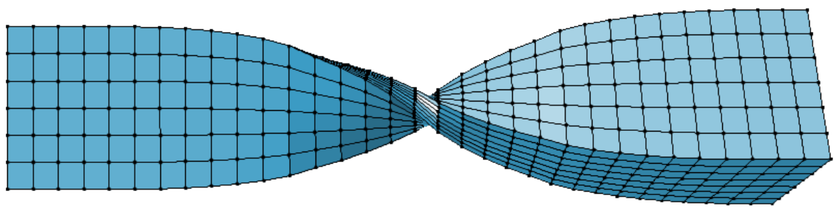
Quaternion dual = généralisation du quaternion pour gérer la *transformation rigide*

Rotation + translation

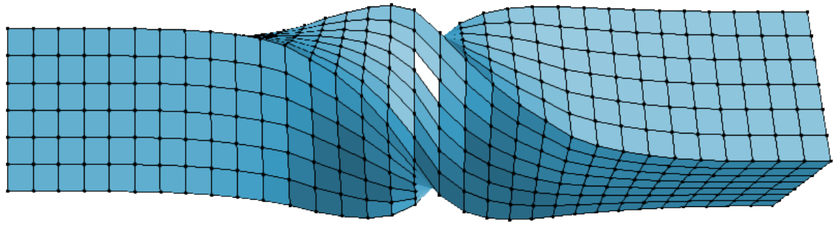
Le quaternion dual unitaire représente un **mouvement hélicoïdal**

Tourner et avancer le long d'un même axe.





$\Rightarrow$



LBS *DQS*

Nombres duaux

W. K. Clifford (1873) : biquaternions, $\omega^2 = 0$ *; E. Study (1891-1903) : algèbre et géométrie des nombres duaux*

Comme les nombres complexes $a_0 + i\,a_1$ avec $i^2 = -1$, mais avec un élément ϵ

$\epsilon \neq 0$ et $\epsilon^2 = 0$: un « infiniment petit » dont le carré disparaît

nombre dual $a = a_0 + \epsilon\,a_\epsilon$: partie réelle a_0, partie duale a_ϵ

Produit

$$(a_0 + \epsilon\,a_\epsilon)\,(b_0 + \epsilon\,b_\epsilon) = a_0\,b_0 + \epsilon\,(a_0\,b_\epsilon + a_\epsilon\,b_0)$$

Division, $b_0 \neq 0$

$$\frac{a_0 + \epsilon\,a_\epsilon}{b_0 + \epsilon\,b_\epsilon} = \frac{(a_0 + \epsilon\,a_\epsilon)\,(b_0 - \epsilon\,b_\epsilon)}{b_0^2} = \frac{a_0}{b_0} + \epsilon\,\frac{a_\epsilon\,b_0 - a_0\,b_\epsilon}{b_0^2}$$

Avec $a = u + \epsilon\,u'$ et $b = v + \epsilon\,v'$

$a\,b = u\,v + \epsilon\,(u'\,v + u\,v')$: dérivée d'un produit

$\frac{a}{b} = \frac{u}{v} + \epsilon\,\frac{u'\,v - u\,v'}{v^2}$: dérivée d'un quotient

$f(x + \epsilon) = f(x) + \epsilon\,f'(x)$: la partie duale porte la dérivée (différentiation automati e

Quaternions duaux

Quaternion à coefficients duaux : $\hat{q} = q_0 + \epsilon\, q_\epsilon$ (2 quaternions, 8 composantes)

Conjugué : $\hat{q}^\star = q_0^\star + \epsilon\, q_\epsilon^\star$

Norme : $\|\hat{q}\|^2 = \hat{q}\,\hat{q}^\star = \|q_0\|^2 + 2\,\epsilon\,(q_0 \cdot q_\epsilon)$: un nombre dual

$$\|\hat{q}\| = \|q_0\| + \epsilon\,\frac{q_0 \cdot q_\epsilon}{\|q_0\|} \qquad \text{car } \sqrt{a + \epsilon\, b} = \sqrt{a} + \epsilon\,\tfrac{b}{2\sqrt{a}} \text{ (dérivée)}$$

Norme 1 : $\|\hat{q}\| = 1 \;\Leftrightarrow\; \|q_0\| = 1$ et $q_0 \cdot q_\epsilon = 0$

8 composantes, 2 contraintes $\Rightarrow$ 6 degrés de liberté : 3 rotation + 3 translation

Quaternion dual de norme 1 $\Leftrightarrow$ transformation rigide

mouvement hélicoïdal : rotation autour d'un axe + translation le long de cet axe

Rotation + translation en quaternion dual

Rotation q_0 (quaternion unitaire)

Translation $t \to q_t = (t_x, t_y, t_z, 0)$

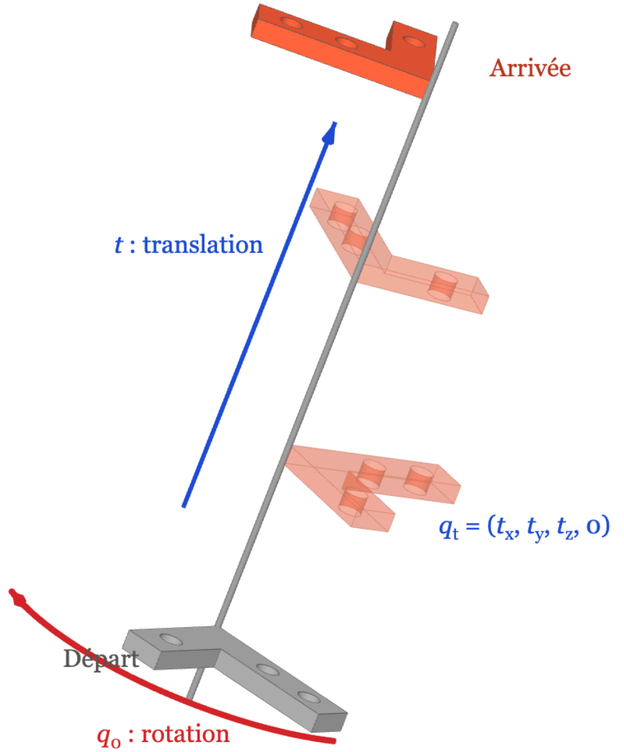
$$\hat{q} = q_0 + \frac{\epsilon}{2}\, q_t\, q_0$$

$q_\epsilon = \frac{1}{2}\, q_t\, q_0$: quaternion qui encode la translation

Norme 1 : $\|q_0\| = 1$ et $q_0 \cdot q_\epsilon = 0$ ✓

$\Rightarrow$ transformation rigide

Translation retrouvée : $q_t = 2\, q_\epsilon\, q_0^\star$



43 min

Appliquer un quaternion dual à un point

$\hat{q} = q_0 + \epsilon\, q_\epsilon,\ \ q_\epsilon = \frac{1}{2}\, t\, q_0$

Point : $\hat{p} = 1 + \epsilon\, p$ (p quaternion pur)

Double conjugué : $\hat{q}^\dagger = q_0^\star - \epsilon\, q_\epsilon^\star$

$$\hat{q}\,\hat{p}\,\hat{q}^\dagger = 1 + \epsilon\left(q_0\, p\, q_0^\star + q_\epsilon\, q_0^\star - q_0\, q_\epsilon^\star\right)\ \ (\epsilon^2 = 0)$$

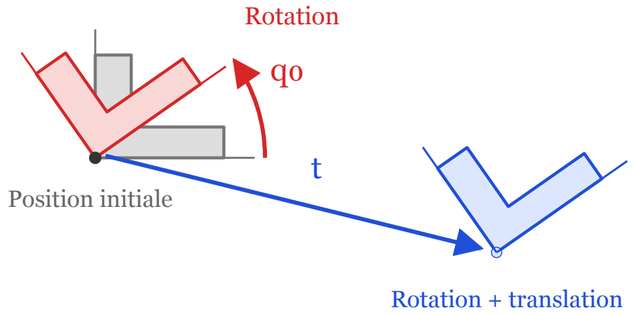
$q_0\, p\, q_0^\star = R\, p$: rotation

$q_\epsilon\, q_0^\star = \frac{1}{2}\, t\, q_0 q_0^\star = \frac{1}{2}\, t$

$q_0\, q_\epsilon^\star = \frac{1}{2}\, q_0 q_0^\star\, t^\star = -\frac{1}{2}\, t$ (t pur)

$$\boxed{\hat{q}\,\hat{p}\,\hat{q}^\dagger = 1 + \epsilon\left(R\, p + t\right)}$$

Un même objet, trois positions



$q_t = (t_x, t_y, t_z, 0)$

$q_\varepsilon = ½\ q_t\ q_0$

La partie duale combine translation et rotation.

$2q_\varepsilon\ q_0^* = q_t$

Le conjugué permet de retrouver la translation.

Skinning par quaternions duaux (DQS)

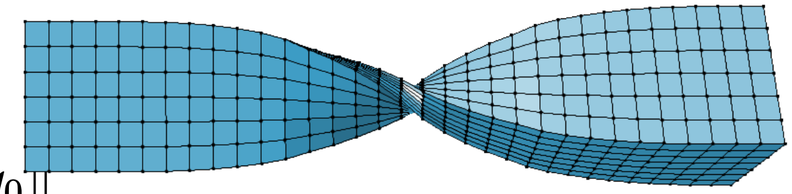
1- **Encoder** chaque os $j : \mathrm{T}_j = (\mathrm{R}_j,\, t_j) \to \hat{q}_j = q_j + \dfrac{\epsilon}{2}\, q_{t_j}\, q_j$

2- **Mélanger et normaliser** : $\hat{q} = \sum_j \alpha_{ij}\, \hat{q}_j = q_0 + \epsilon\, q_\epsilon,\ \ \hat{q} \leftarrow \hat{q} / \|q_0\|$

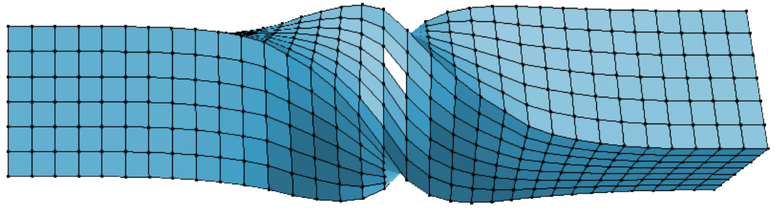
signe de chaque $\hat{q}_j$ aligné (q et $-q$: même rotation)

3- **Revenir** à $\mathrm{T} = (\mathrm{R},\, t) : q_0 \to \mathrm{R},\ \ t$: partie vectorielle de $2\, q_\epsilon\, q_0^\star$

$p = \mathrm{T}\, p^0$

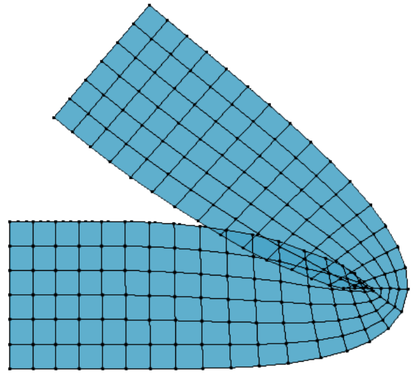


LBS

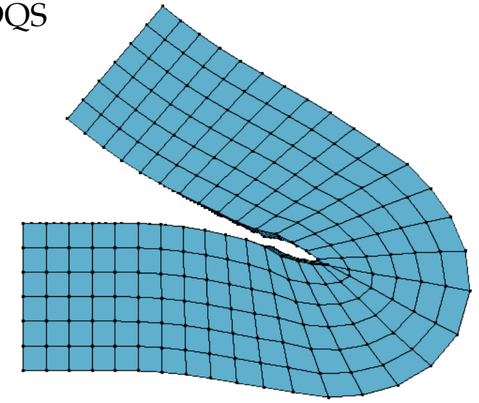


DQS

LBS : moyenne de **matrices**
DQS : mélange **normalisé** de **transformations rigides**



LBS



DQS

[Skinning with Dual Quaternions. Kavan et al. I3D, 2007]

LBS ou DQS : DQS conserve le volume en torsion

45 min

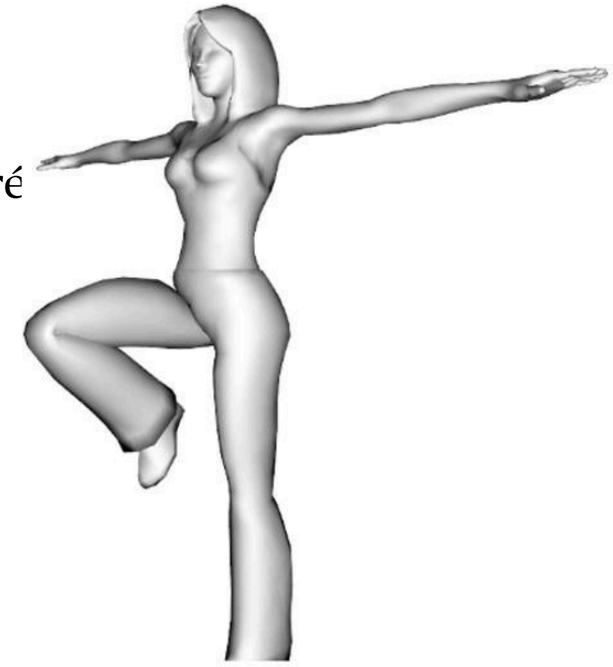
Quaternions duaux vs LBS

Quaternion dual

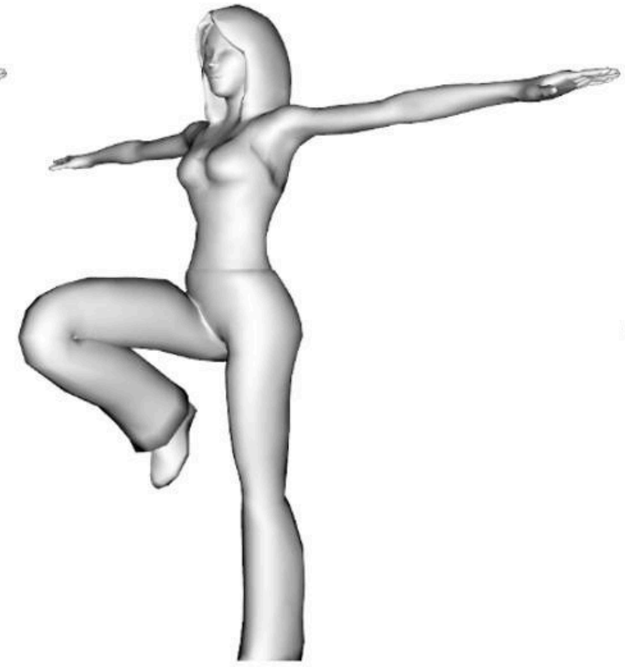
- (+) Résout le *candy wrapper* (torsion $< 180°$)
- (+) Presque aussi efficace que LBS
- (—) Peut créer un renflement artificiel/non désiré

Pas toujours préféré à LBS

Les deux solutions (LBS, DQS) sont proposées dans les outils standards



LBS



DQS

Approches de recherche

Implicit skinning, retargeting, regard, animation expressive et dynamic skinning

PARTIE 7 / 7

1 Historique de l'animation 3D

2 Interpolation de keyframes

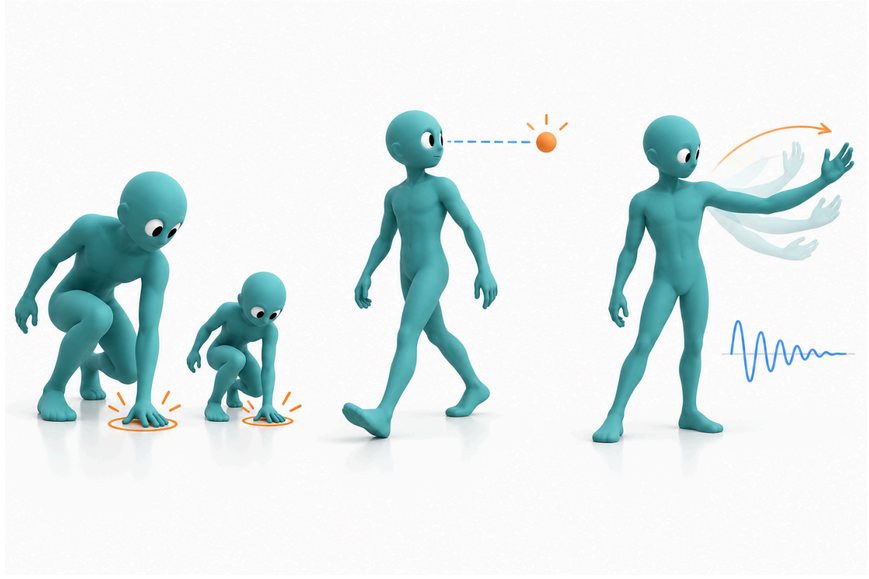
3 Représentation de rotations

4 Interpolation de rotations

5 Animation de personnages

6 Animation et quaternions duaux

7 **Approches de recherche**

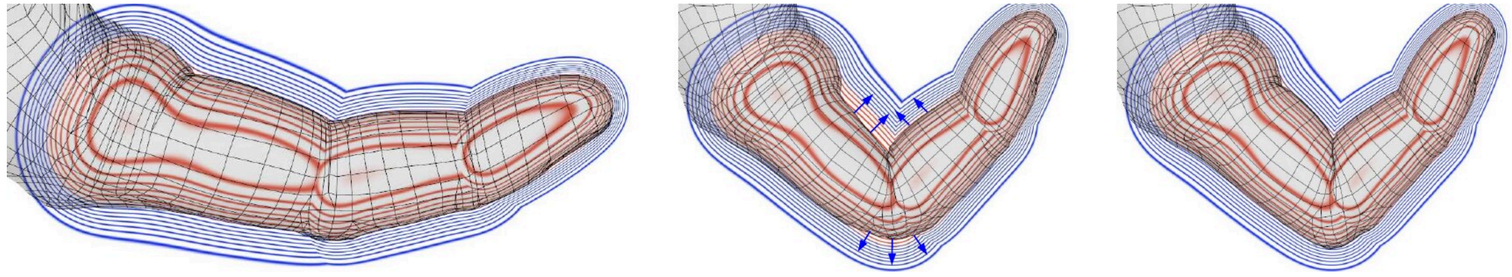


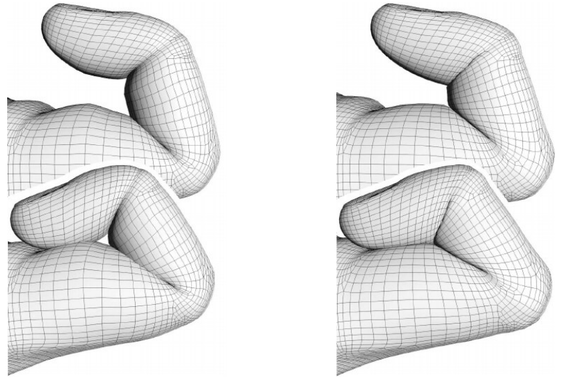
Contacts · regard · effets secondaires du mouvement

47 min

Implicit skinning : contact et volume

- LBS / DQS au pli : perte de volume, auto-intersections
- Chaque partie (os i) $\approx$ iso-surface $f_i(x) = 0.5$ d'un champ scalaire
- Union des champs $f = \max_i f_i$: os rigide, **contact** dans le pli
- À chaque image : DQS, puis chaque sommet remonte ∇f jusqu'à son iso-valeur
- Temps réel (GPU), sans détection de collision





LBS Implicit skinning

[Implicit Skinning: Real-Time Skin Deformation with Contact Modeling.
R. Vaillant, L. Barthe, G. Guennebaud, M.-P. Cani, D. Rohmer, B. Wyvill, O. Gourmel, M. Paulin. ACM SIGGRAPH, 2013]

ReConForM : transfert de mouvement

Transférer une animation vers une autre morphologie

copier les rotations : interpénétrations, contacts perdus

Décrire la pose par les **relations entre sommets clés**

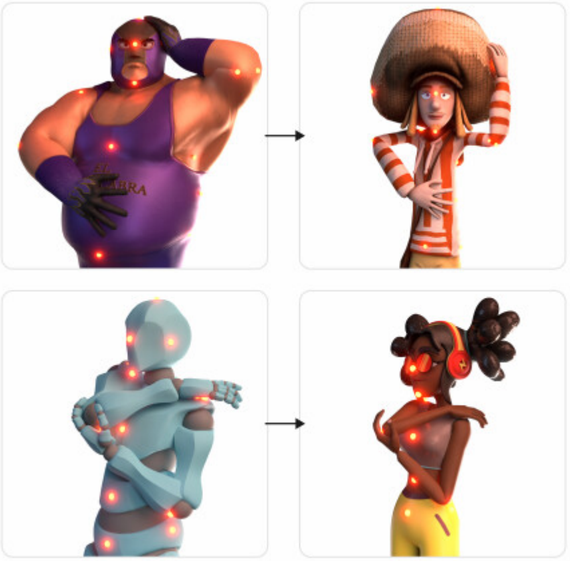
distances, directions, pénétration, hauteur au sol

Ne garder que ce qui compte : paires proches (contacts)

Trouver la pose cible qui reproduit ces relations

$$\min_{\text{pose cible}} \sum W \left\| M^{source} - M^{cible} \right\|^2$$

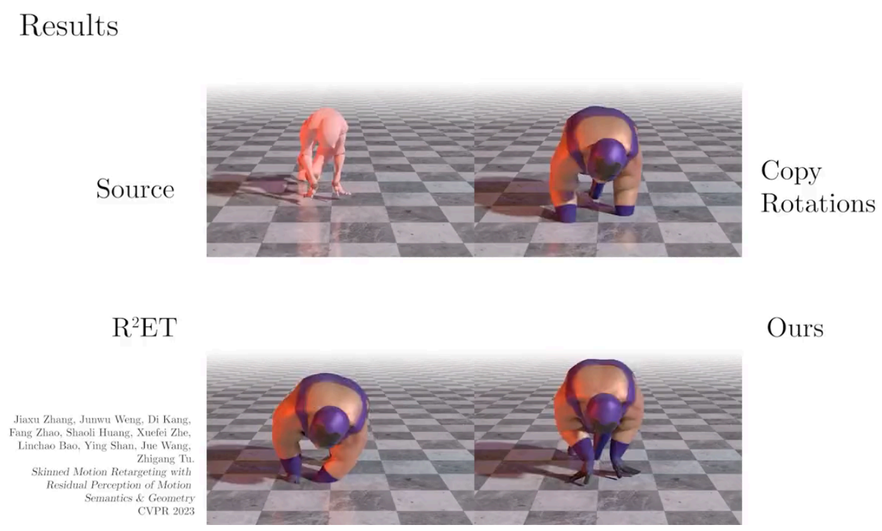
Peu de contraintes actives $\Rightarrow$ temps réel

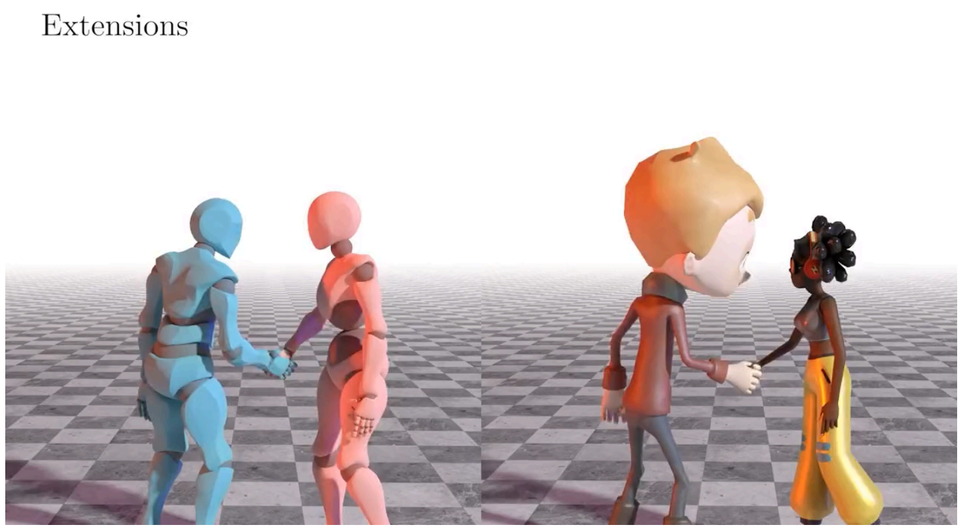


[ReConForM: Real-time Contact-aware Motion Retargeting for more Diverse Character Morphologies. T. Cheynel, T. Rossi, B. Bellot-Gurlet, D. Rohmer, M.-P. Cani. Eurographics, 2025]

ReConForM : résultats

- Contacts préservés : auto-contacts, sol, entre personnages, terrain non plat







Vidéo et images : Cheynel et al. ur gr blics 2025

Regard réactif pendant la marche

- Personnage qui marche, regard figé droit devant : semble absent
- Où regarder ? chemin, pente du terrain, formes, couleurs, mouvements
- Saillance (réseau de neurones) + décision issue des sciences cognitives
- Temps réel, comportement réglable : orienté but ou exploratoire





[Reactive Gaze during Locomotion in Natural Environments.
J. K. Melgaré, D. Rohmer, S. R. Musse, M.-P. Cani. Symposium on Computer Animation (SCA), 2024]

Les 12 principes de l'animation

- Interpoler entre des poses réalistes ne suffit pas pour une animation expressive
- *12 principes* des animateurs de Disney (F. Thomas, O. Johnston, *The Illusion of Life*, 1981)

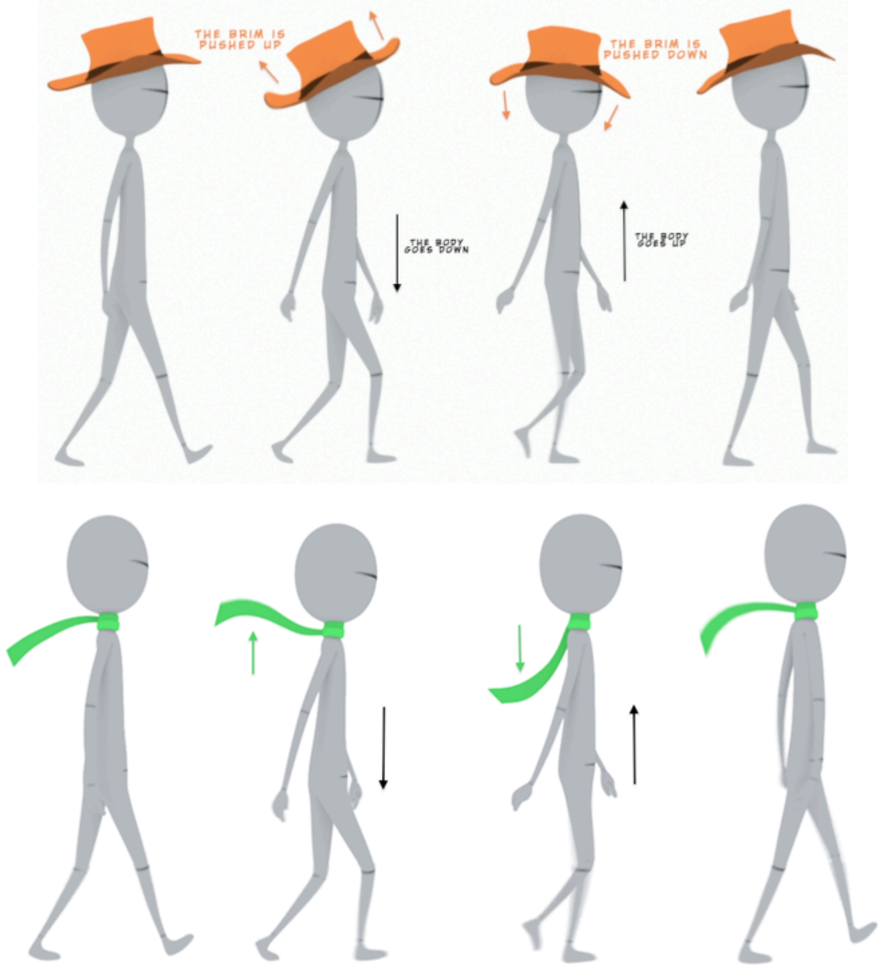
1. Timing (rythme)
2. Dessin en volume (solid drawing)
3. Ralentis en début et fin (slow-in, slow-out)
4. **Écrasement et étirement (squash & stretch)**
5. **Anticipation**
6. **Continuité du mouvement (follow through)**
7. Action secondaire
8. Exagération
9. Attrait (appeal)
10. Trajectoires en arcs
11. Mise en scène (staging)
12. Animation directe / pose à pose

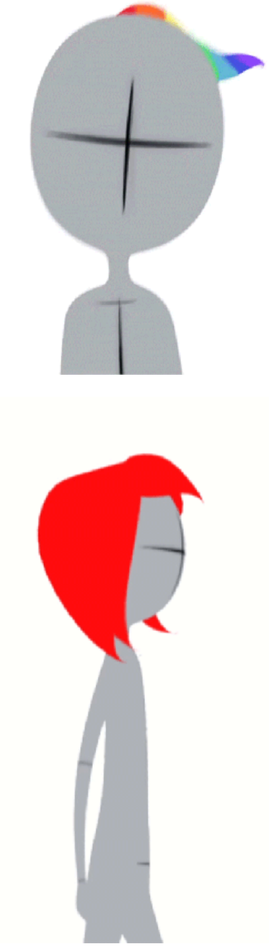


(C) Chiara Porri

Continuité du mouvement (follow-through)

Les parties souples suivent le mouvement avec retard





(C) Chiara Porri

Effets dynamiques : des déformations exagérées

Mouvement rapide : la forme s'**étire**, **traîne** derrière le mouvement, se dédouble

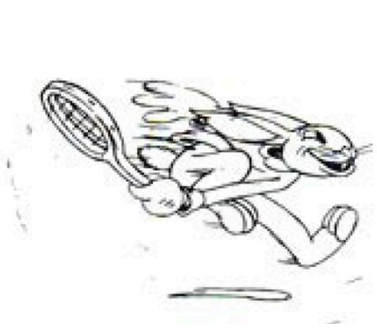
Donne l'énergie, le poids, la souplesse du personnage, **visible même sur une image fixe**

$\neq$ animation réaliste








Raiponce (Disney, 2010) ; Moi, moche et méchant 2 (Illumination, 2013) ; The Tortoise and the Hare (Disney, 1935)

53 min

Dynamic skinning : du skinning à la vitesse

Skinning (LBS) : $p_i = \sum_j \alpha_{ij}\, p_{i/j}$

$p_{i/j} = T_j\, p_i^0$: sommet i transporté rigidement par j

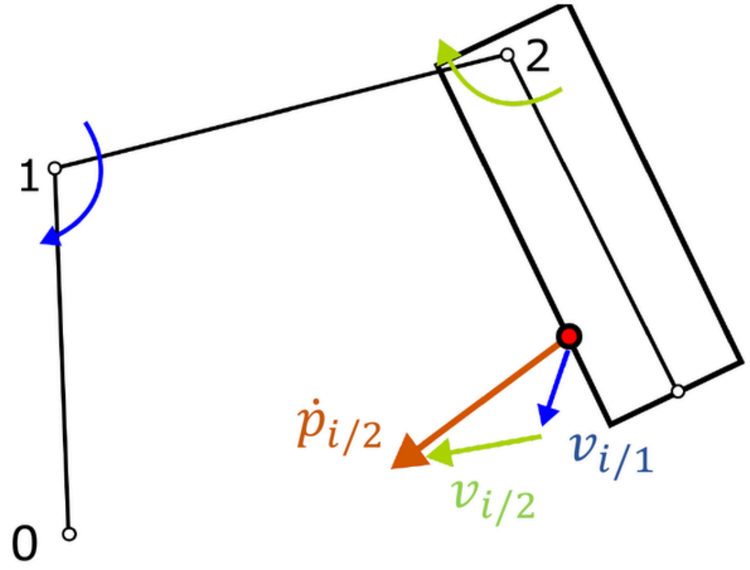
Dériver en temps : $\dot{p}_i = \sum_j \alpha_{ij}\, \dot{p}_{i/j}$

j est entraînée par ses ancêtres $\mathcal{A}(j)$ (j inclus)

$\dot{p}_{i/j} = \sum_{k \in \mathcal{A}(j)} v_{i/k}$

$v_{i/k} = L_k + \Omega_k \times (p_i - c_k)$: vitesse rigide propre de k

Double somme : $\dot{p}_i = \sum_j \alpha_{ij} \sum_{k \in \mathcal{A}(j)} v_{i/k}$



Vitesse d'un point : somme des contributions des articulations **en amont**

[Dynamic Skinning: Kinematics-Driven Cartoon Effects for Articulated Characters. D. Rohmer, K. Salem, N. Kalyanasundaram, V. Zordan. IEEE TVCG, 2026]

Dynamic skinning : poids hiérarchiques

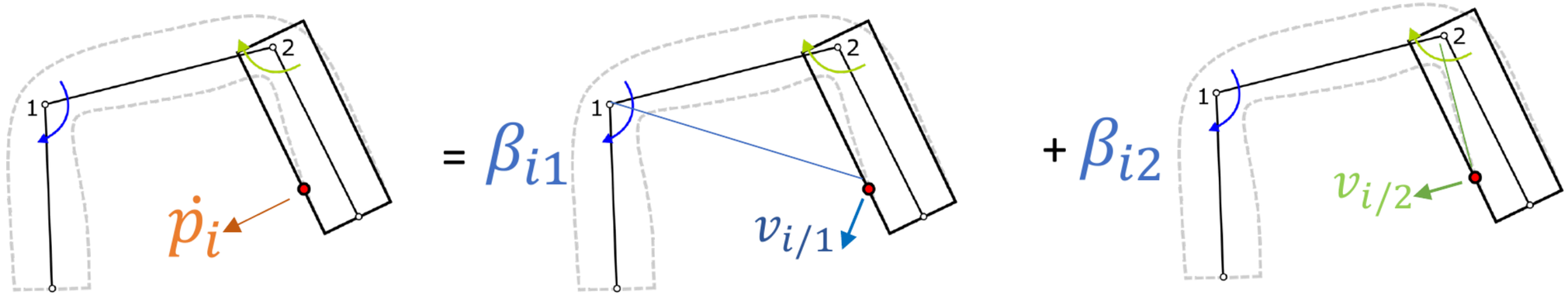
Échanger les sommes : regrouper par articulation k

k intervient pour chaque descendant $j \in \mathcal{D}(k)$ (k inclus)

$$\dot{p}_i = \sum_k \beta_{ik}\, v_{i/k} \qquad \beta_{ik} = \sum_{j \in \mathcal{D}(k)} \alpha_{ij}$$

β_{ik} : **poids hiérarchiques**, influence du mouvement propre de k sur i

Même forme que le skinning : $\sum_j \alpha_{ij}\, p_{i/j} \leftrightarrow \sum_k \beta_{ik}\, v_{i/k}$



54 min

Déformation guidée par la vitesse

Même forme que le skinning

$$p_i = \sum_j \alpha_{ij}\, \mathrm{T}_j\, p_i^0 \quad \leftrightarrow \quad \dot{p}_i = \sum_j \beta_{ij}\, v_{i/j}$$

Idée : remplacer $v_{i/j}$ par une déformation procédurale ψ

$$p_i = p_i^{LBS} + \sum_j \beta_{ij}\, \psi\left(\Omega_j,\, c_j,\, p_i^{LBS}\right)$$

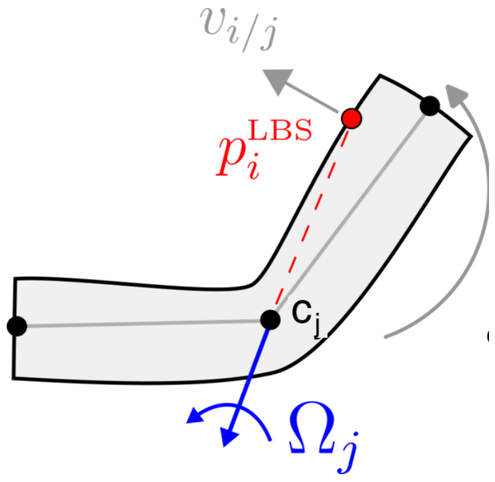
ψ exprimée dans le **repère local** de l'articulation

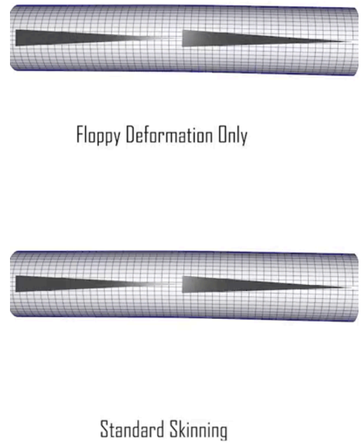
centre c_j, vitesse angulaire Ω_j

flexion : rotation d'axe Ω_j, d'angle $-s\, \|\Omega_j \times (p_i - c_j)\|$

la peau « traîne » derrière le mouvement

Calcul direct par sommet : temps réel (GPU)





[Velocity Skinning for Real-time Stylized Skeletal Animation.
D. Rohmer, M. Tarini, N. Kalyanasundaram, F. Moshfeghifar, M.-P. Cani, V. Zordan. Eurographics, 2021]

Déformation guidée par la vitesse : démonstration

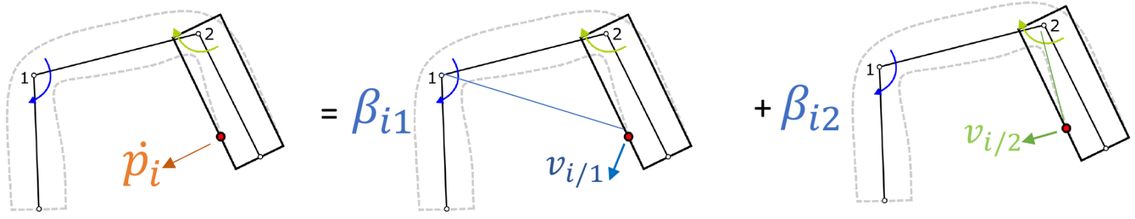
Ajouter le temps : contrôle spatial et temporel

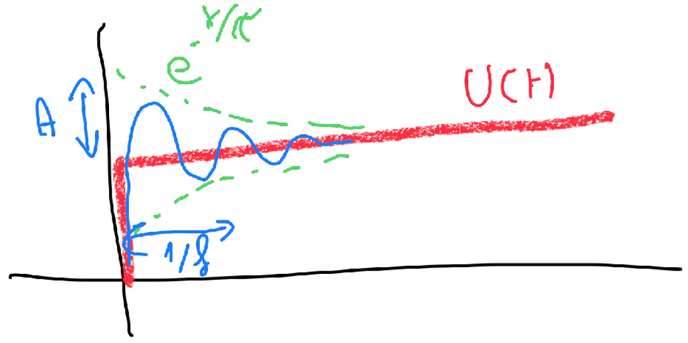
Pas de rebond $\Rightarrow$ filtrer le **signal** de vitesse $\Omega_j(t)$

$$p_i = p_i^{LBS} + \sum_j \underbrace{\beta_{ij}}_{\text{espace}} \psi\Big(\underbrace{\mathcal{F}(\Omega_j, t)}_{\text{temps}}, c_j, p_i^{LBS}\Big)$$

Espace : poids hiérarchiques β_{ij}
issus de la dérivation en vitesse
où la peau se déforme

Temps : filtre $\mathcal{F}$ sur $\Omega_j(t)$
comment elle réagit au cours du temps
défini par sa **réponse à un échelon** de vitesse





Contrôle temporel : une convolution

1. **Choisir** la réponse h à un échelon $U(t)$ de vitesse
   $h(t) = \big(1 + \kappa(t)\big)\, U(t), \quad \kappa(t) = A\, e^{-t/\tau} \sin(2\pi f\, t)$
   $\kappa(0) = 0, \kappa(W) \approx 0\ (W \gg \tau)$

2. **Réponse impulsionnelle** : sa dérivée
   $\dot{h}(t) = \delta(t) + \dot{\kappa}(t)\, U(t)$

3. **Filtre linéaire** $\Rightarrow$ convolution

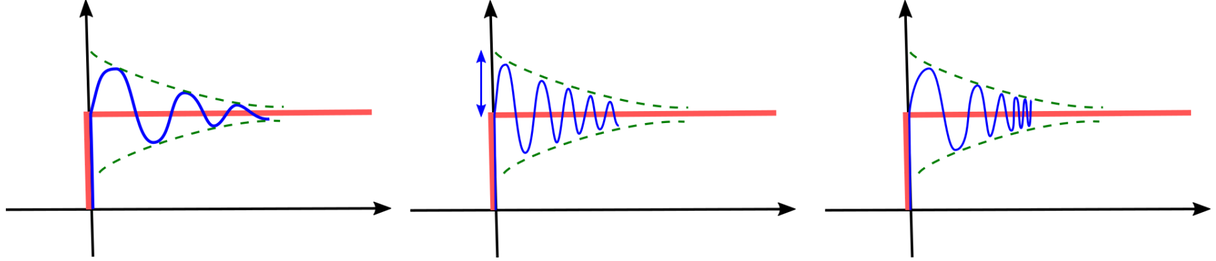
$$\mathcal{F}(\Omega_j, t) = (\dot{h} \star \Omega_j)(t) = \Omega_j(t) + \int_0^W \kappa(u)\, \dot{\Omega}_j(t - u)\, du$$

chaque variation de vitesse $\dot{\Omega}_j$ déclenche une oscillation κ

A, τ, f **indépendants**

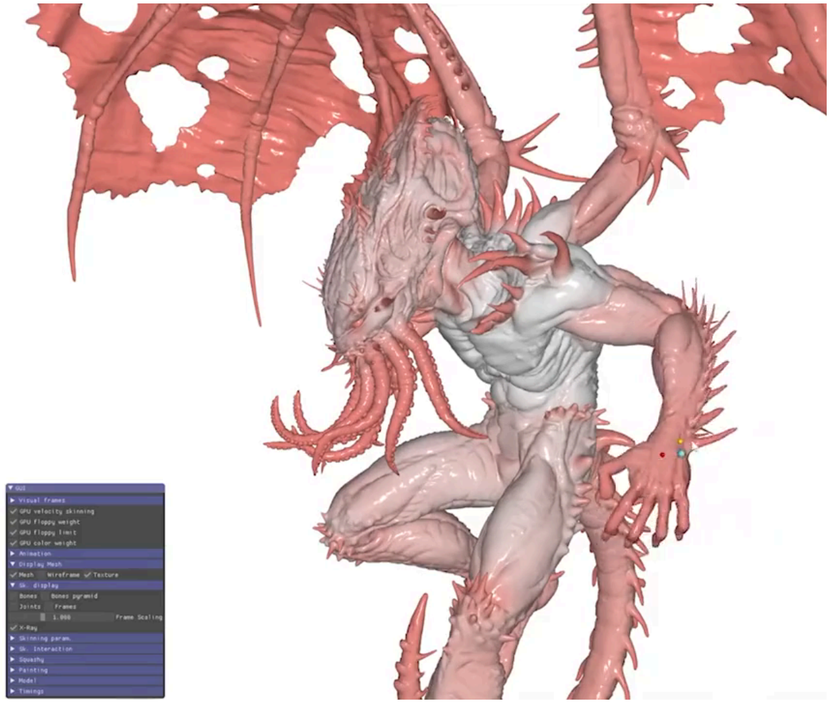
Pas d'intégration numérique : stable, temps réel



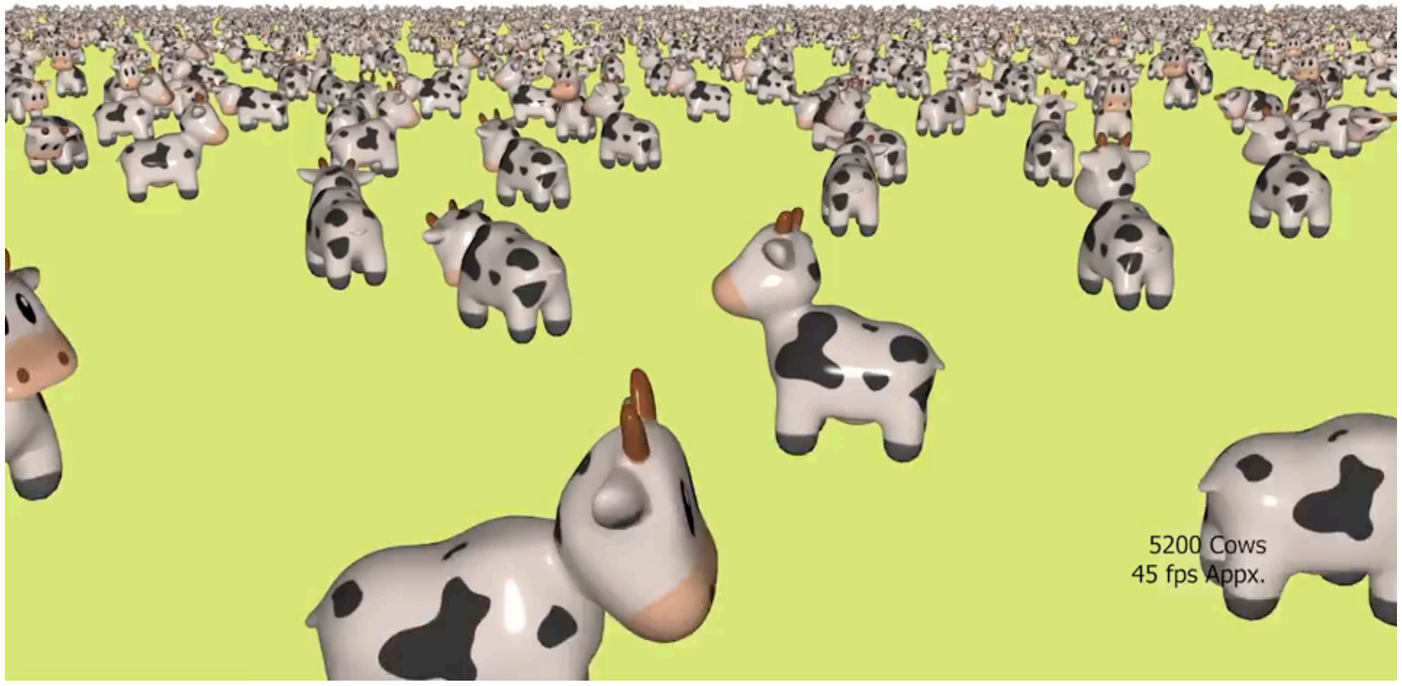


Chaque sommet se calcule indépendamment : temps réel

- Calcul **procédural** à chaque instant : pas de simulation physique
- Chaque sommet **indépendant** des autres ⇒ parallèle, sur GPU (vertex shader)
- Coût du même ordre que LBS



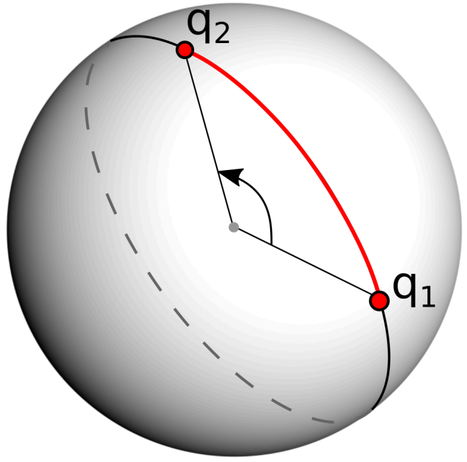
1,5 million de polygones



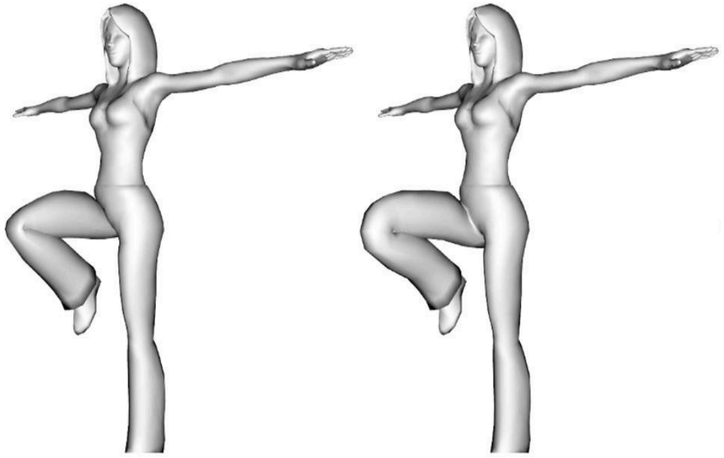
5 200 vaches, 45 images/s

Dynamic skinning : démonstration

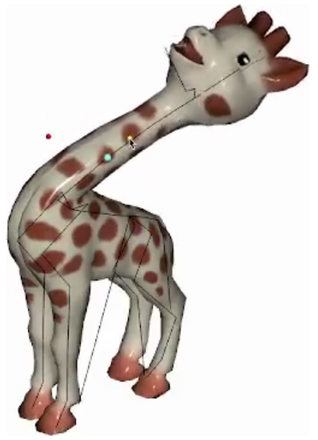
Conclusion : représenter, mélanger, enrichir



1. Représenter le mouvement
matrices, Euler, axe-angle, quaternions



2. Mélanger les transformations
skinning : matrices (LBS) ou quaternions duaux (DQS)



3. Enrichir la déformation
vitesse, filtres $\Rightarrow$ personnage expressif

Et au-delà : simulation physique (muscles, tissus, cheveux), apprentissage à partir de capture de mouvement

60 min

Merci pour votre attention

Annexes

Détails mathématiques complémentaires

Quaternion et formule de Rodrigues

Demi-angle $\phi = \theta/2$, $\|n\| = 1$: $q = (n \sin\phi,\, \cos\phi)$, $q^\star = (-n \sin\phi,\, \cos\phi)$, $q_v = (v,\, 0)$

Produit : $(s_1, w_1)\,(s_2, w_2) = (s_1\, w_2 + s_2\, w_1 + s_1 \times s_2\,,\; w_1\, w_2 - s_1 \cdot s_2)$

1. $q\, q_v = (n \sin\phi,\, \cos\phi)\,(v,\, 0) = (\underbrace{\cos\phi\, v + \sin\phi\; n \times v}_{a}\;,\; \underbrace{-\sin\phi\,(n \cdot v)}_{b})$

2. $(a,\, b)\,(-n \sin\phi,\, \cos\phi) = (\cos\phi\; a - \sin\phi\; b\, n - \sin\phi\; a \times n\,,\; \cos\phi\; b + \sin\phi\; a \cdot n)$

   Partie réelle : $-\sin\phi\cos\phi\,(n \cdot v) + \sin\phi\cos\phi\,(v \cdot n) + \sin^2\phi\; \underbrace{(n \times v) \cdot n}_{0} = 0$: quaternion pur ✓

   Partie vectorielle, avec $v \times n = -\,n \times v$ et $(n \times v) \times n = v - (n \cdot v)\, n$:

$$\cos^2\phi\, v + \sin\phi\cos\phi\; n \times v + \sin^2\phi\,(n \cdot v)\, n + \sin\phi\cos\phi\; n \times v - \sin^2\phi\,\big(v - (n \cdot v)\, n\big)$$

$$= (\cos^2\phi - \sin^2\phi)\, v + 2\sin\phi\cos\phi\; n \times v + 2\sin^2\phi\,(n \cdot v)\, n$$

$$q\, q_v\, q^\star = \big(\cos\theta\, v + \sin\theta\; n \times v + (1 - \cos\theta)\,(n \cdot v)\, n\,,\; 0\big)$$: formule de **Rodrigues**

Angle double : $\cos^2\phi - \sin^2\phi = \cos\theta$, $2\sin\phi\cos\phi = \sin\theta$, $2\sin^2\phi = 1 - \cos\theta$. Retour

Quaternion dual d'une transformation rigide : norme 1

$$\hat{q} = q_0 + \frac{\epsilon}{2}\, q_t\, q_0, \;\; q_0 \text{ unitaire, } q_t = (t_x, t_y, t_z, 0)$$

$$\|\hat{q}\| = 1 \;\Leftrightarrow\; \|q_0\| = 1 \text{ et } q_0 \cdot q_\epsilon = 0$$

$\|q_0\| = 1$ ✓

$q_0 = (x, y, z, w)$: $q_0 \cdot q_\epsilon = q_0 \cdot \left(\frac{1}{2}\, q_t\, q_0\right) = \frac{1}{2}\left(x\,(t_x w + t_y z - t_z y) + y\,(t_y w + t_z x - t_x z) + \ldots\right) = 0$ ✓