Animating Virtual Characters

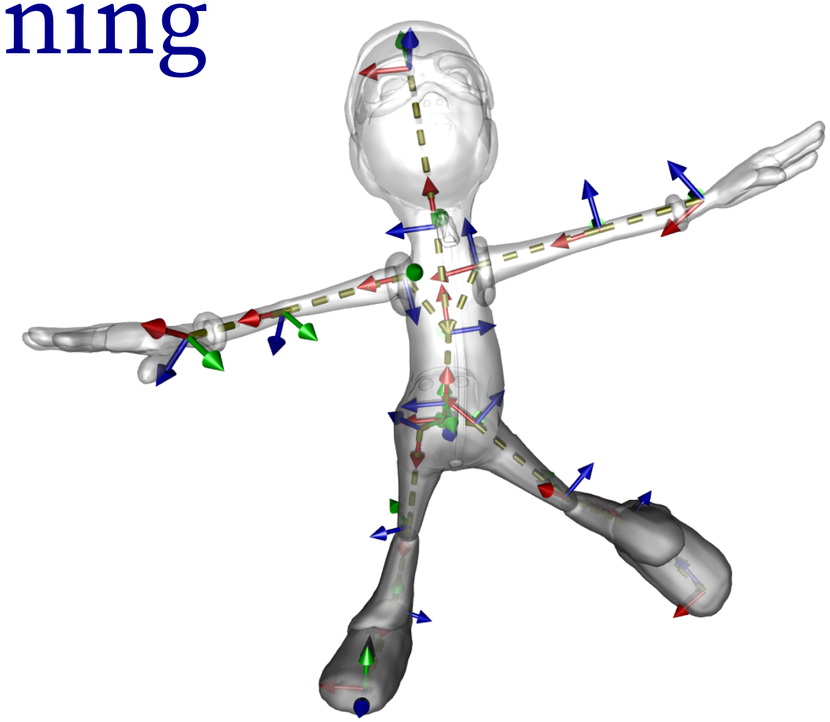
Skin deformation

Skeleton based deformation - Skinning

Objective: Deform articulated character

Idea: Use skeleton to control limbs

Articulations as rotations



Animation Skeleton

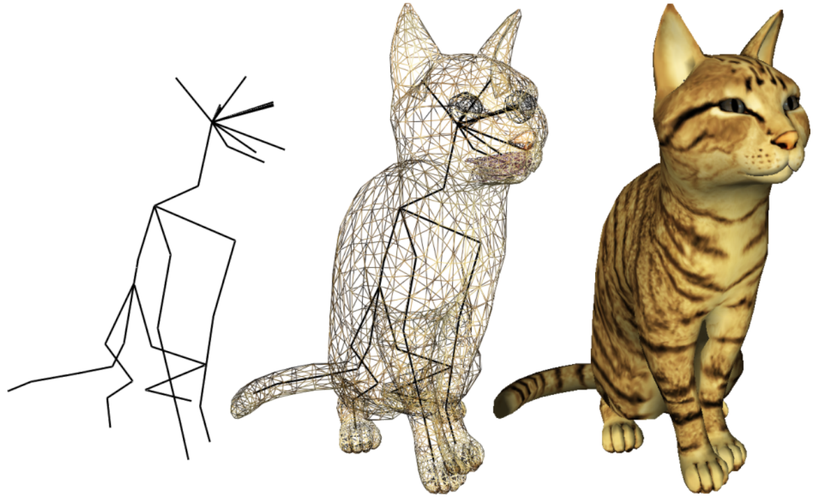
Set of frames T_i: position, orientation

Describe non-rigid parts of the character (dof)

Terminology

Joint = A frame T_i

Bone = Segment between two joints

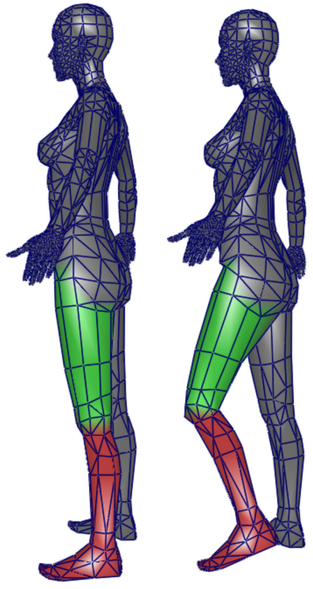


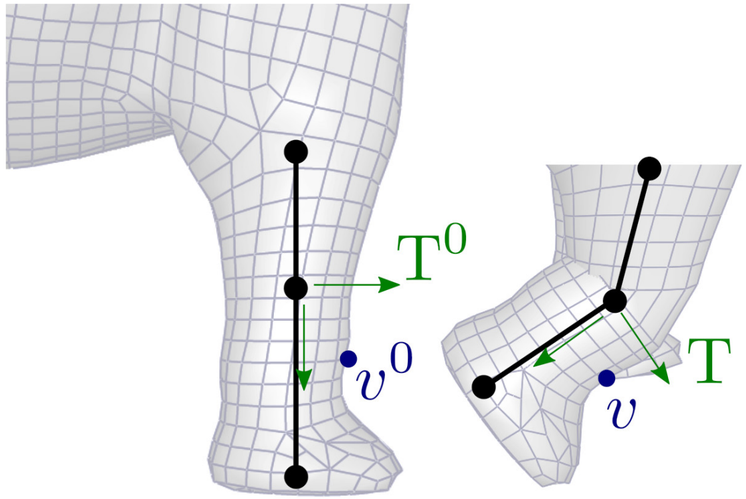
Note: Animation skeleton $\neq$ Anatomical skeleton

Rigid Skinning

Attach rigidly subset of vertices to specific bones, described by its root joint/frame.

Vertices are following rigid deformation of their associated frame.





Deformation formulation

Consider, at rest pose/initial state

- A frame T^0
- A vertex v^0 attached to this frame

After deformation

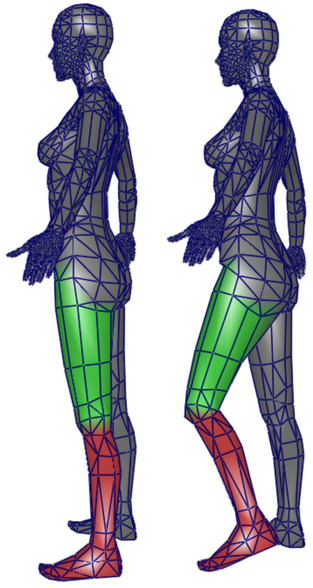
- New frame T
- New vertex position v

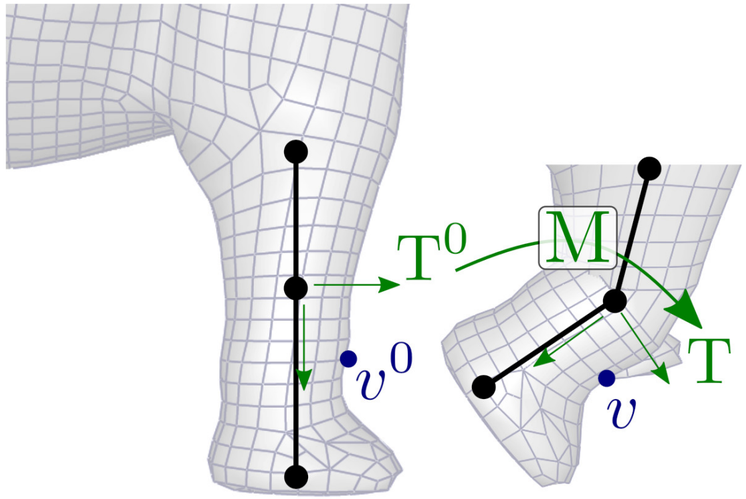
Question: What are the new coordinates v w/r T, T^0, v^0 ?

Rigid Skinning

Attach rigidly subset of vertices to specific bones, described by its root joint/frame.

Vertices are following rigid deformation of their associated frame.





Deformation formulation

Consider, at rest pose/initial state

- A frame T^0
- A vertex v^0 attached to this frame

After deformation

- New frame T
- New vertex position v

Question: What are the new coordinates v w/r T, T^0, v^0 ?

v^0 and v have similar local coordinates w/r to T^0 and T

$$\Rightarrow \mathrm{T}^{-1} v = (\mathrm{T}^0)^{-1} v_0$$

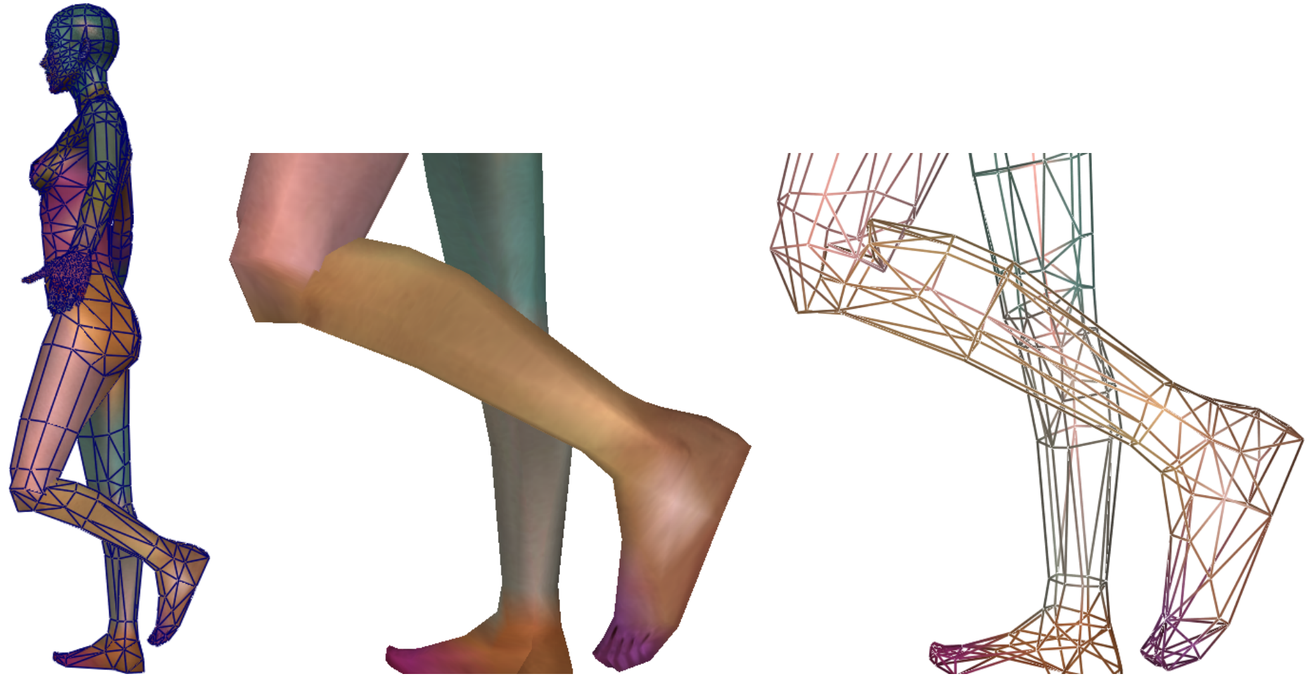
$$\Rightarrow v = \mathrm{T}\,(\mathrm{T}^0)^{-1} v_0 = \mathrm{M}\, v_0$$

Rigid Skinning

(+) Skeleton is easy to build

(+) Skeleton interaction is intuitive to model rigid articulation

(-) Discontinuities/Inter-penetrations



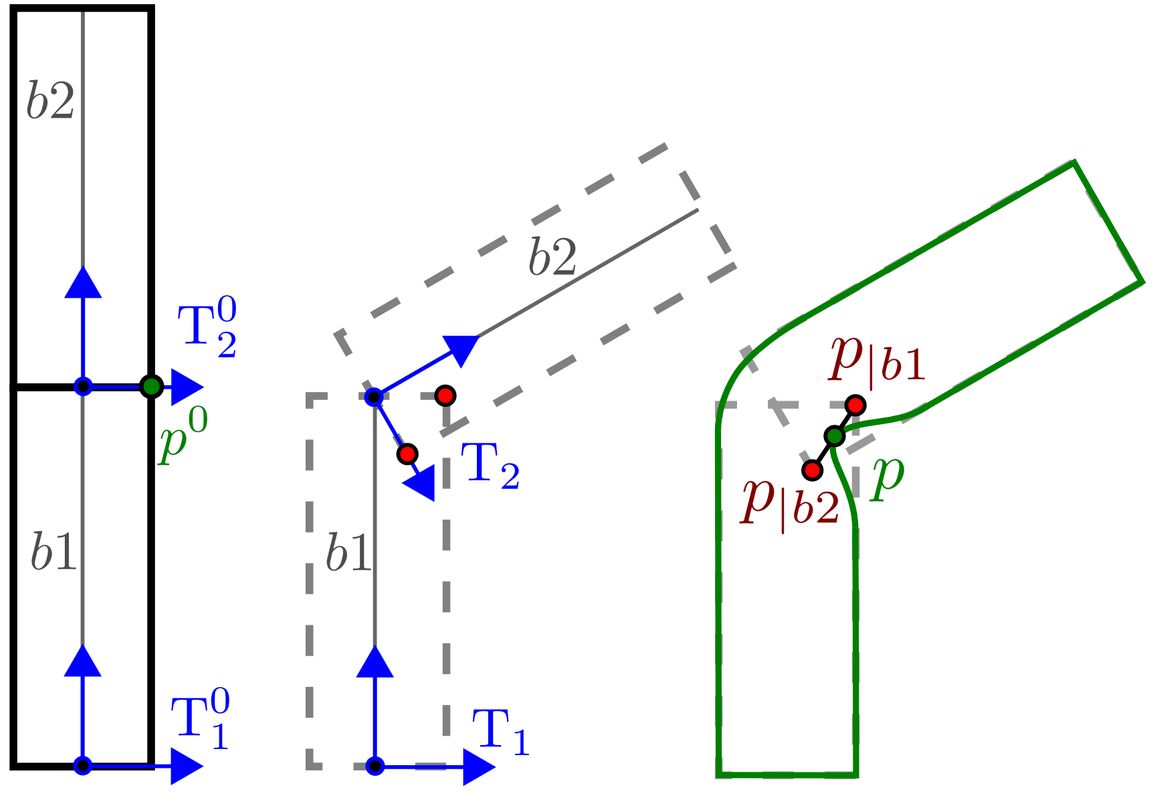
Idea of smooth skinning: Blend discontinuous transformation around articulation

Smooth skinning

Linear Blend Skinning (LBS): Linear interpolation of positions between associated frames

Example at middle vertex position of a bending cylinder

$$p = 0.5\, p_{|b1} + 0.5\, p_{|b2}$$
$$p = 0.5 \underbrace{\mathrm{T}_1 \left(\mathrm{T}_1^0\right)^{-1}}_{\mathrm{M}_1} p^0 + 0.5 \underbrace{\mathrm{T}_2 \left(\mathrm{T}_2^0\right)^{-1}}_{\mathrm{M}_2} p^0$$
$$p = \left(0.5\, \mathrm{M}_1 + 0.5\, \mathrm{M}_2\right) p^0$$

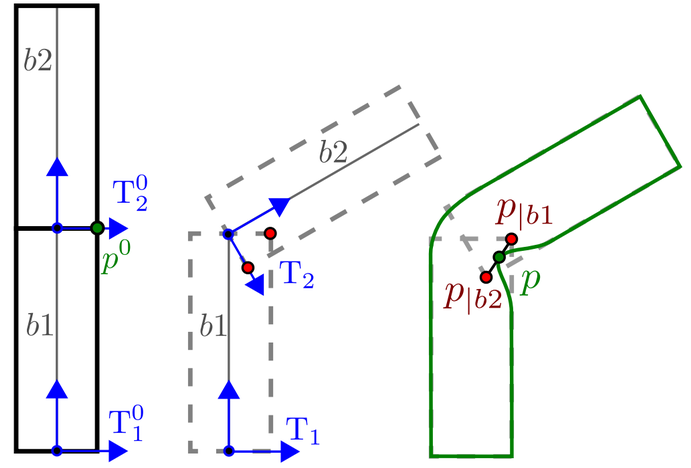


Smooth skinning

Linear Blend Skinning (LBS): Linear interpolation of positions between associated frames

Example at middle vertex position of a bending cylinder

$$p = 0.5\, p_{|b1} + 0.5\, p_{|b2}$$
$$p = 0.5 \underbrace{\mathrm{T}_1 \left(\mathrm{T}_1^0\right)^{-1}}_{\mathrm{M}_1} p^0 + 0.5 \underbrace{\mathrm{T}_2 \left(\mathrm{T}_2^0\right)^{-1}}_{\mathrm{M}_2} p^0$$
$$p = \left(0.5\, \mathrm{M}_1 + 0.5\, \mathrm{M}_2\right) p^0$$



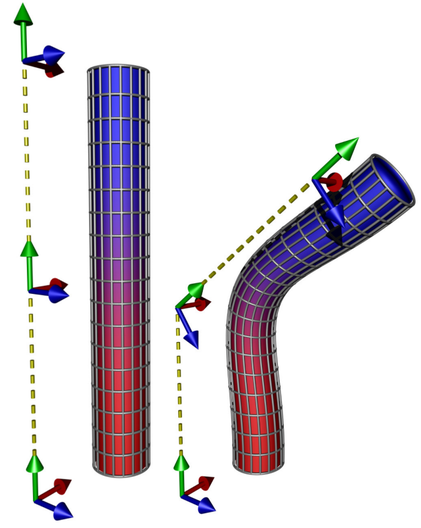
Can be generalized to arbitrary interpolation between two bones

$$p = \alpha\, p_{|b1} + (1-\alpha)\, p_{|b2} = \left(\alpha \mathrm{M}_1 + (1-\alpha)\, \mathrm{M}_2\right)\, p^0$$

α: skinning weights

Can be generalized to any number of bones

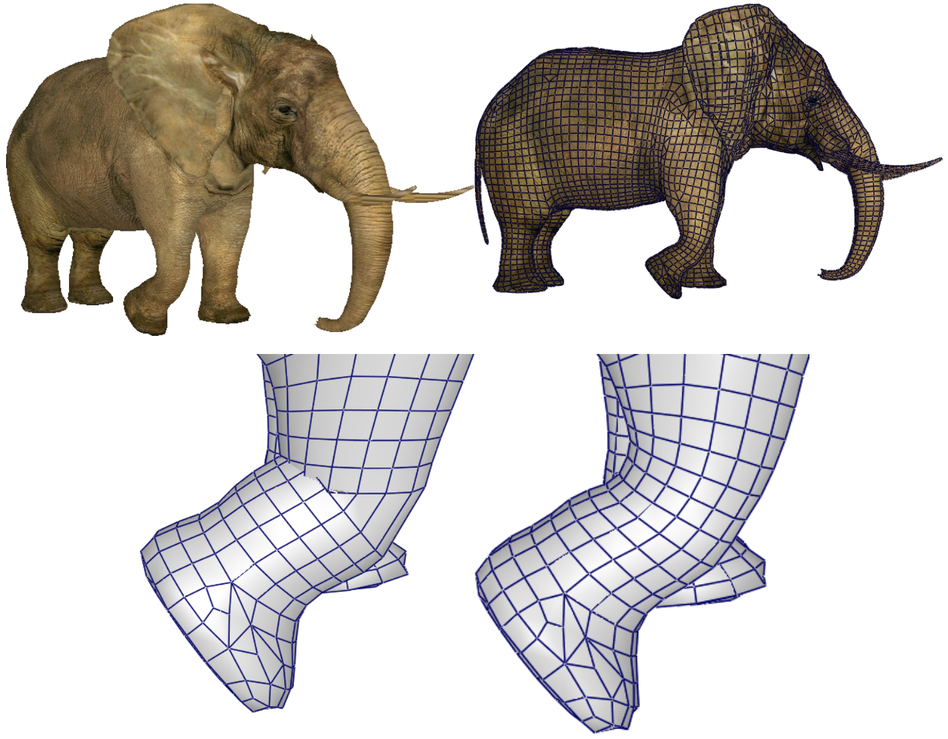
$$p = \sum_{k=0}^{N-1} \alpha_k\, p_{|bk} = \left(\sum_{k=0}^{N-1} \alpha_k\, \mathrm{M}_k\right)\, p^0 \,, \quad \sum_k \alpha_k = 1$$



Smooth skinning - Summary

$$p = \left(\sum_{k=0}^{N-1} \alpha_k \, \mathrm{M}_k \right) p^0 = \left(\sum_{k=0}^{N-1} \alpha_k \; \mathrm{T}_k \, (\mathrm{T}_k^0)^{-1} \right) p^0$$

$$\forall k \, , \;\; \alpha_k \in [0,1], \text{ and } \sum_{k=0}^{N-1} \alpha_k = 1$$



The current **standard** for almost all articulated character deformations

- Intuitive deformation
- Controlable shape (*through weights*)
- Fast to compute (GPU compatible)

  matrix average, multiplication matrix-vector

$\Rightarrow$ Heavily used in Animation cinema & Video Game

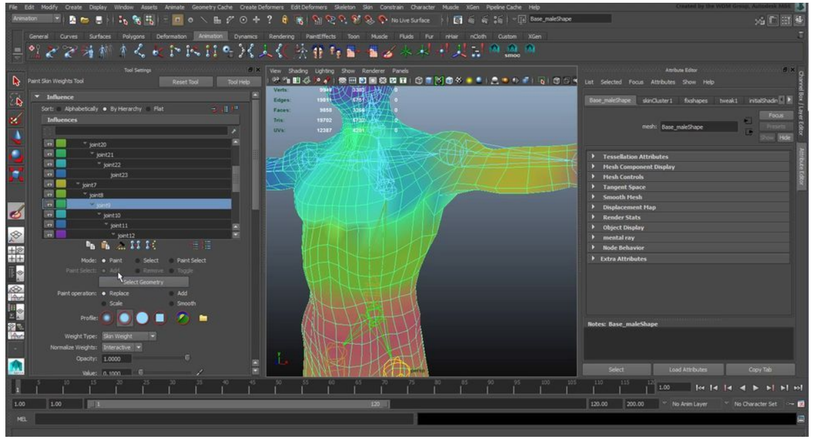
[Joint-Dependent Local Deformations for Hand Animation and Object Grasping. Nadia Magnenat-Thalmann, Rochard Laperière, Daniel Thalmann. Graphics Interface, 1988]

[Over My Dead, Polygonal Body. Jeff Lander. Game Developer Magazine, 1998]

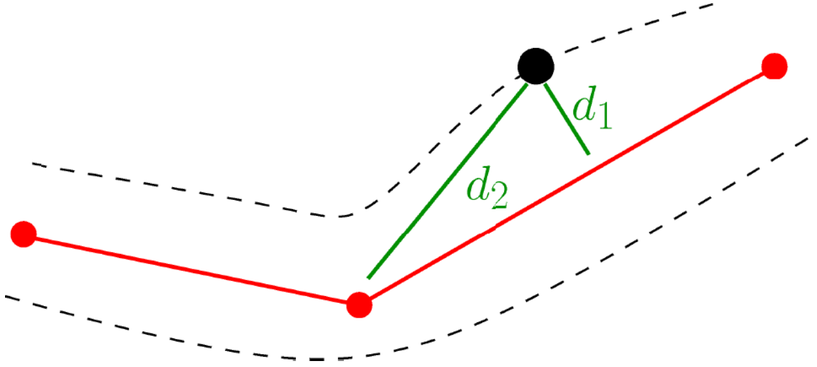
Skinning weights

How to generate skinning weights ?

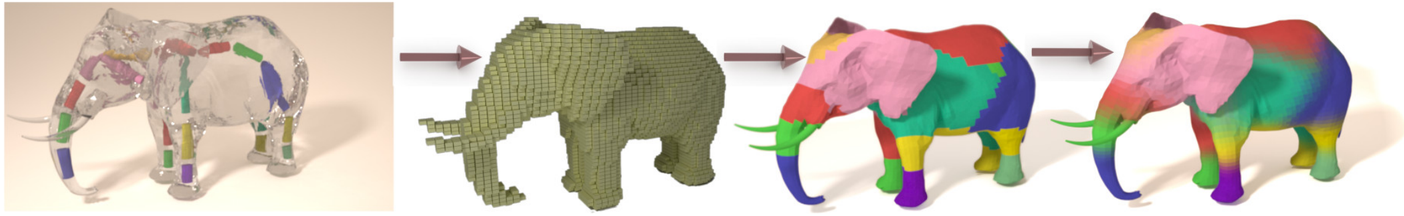
- Paint them manually
- Automatic computation



ex. Using cartesian distances: $\alpha_1 = d_1^{-1}/(d_1^{-1} + d_2^{-1})$



Or using diffusion on the volume/surface



Rigging: Associating bones and skinning weights (or any animation handle) to mesh parts

Skinning: File Format

Unfortunately few standard open format to store skinning animation data

- Main open format: Collada (XML), glTF (JSON)
- Common software related formats: FBX, Blend, 3DS, ...

krissie Maya 7.0 | ColladaMaya v2.04 | FCollada v1.14 Collada Maya Export Options:
bakeTransforms=0;exportPolygonMeshes=1;bakeLighting=0;isSampling=0;
curveConstrainSampling=0;exportCameraAsLookat=0; exportLights=0;exportCameras=1;exportJointsAndSkin=1;
exportAnimations=1;exportTriangles=0;exportInvisibleNodes=0;
exportNormals=1;exportTexCoords=1;exportVertexColors=0;exportTangents=0;
exportTexTangents=0;exportConstraints=0;exportPhysics=0;exportXRefs=0;
dereferenceXRefs=0;cameraXFov=0;cameraYFov=1 file:///E:/maya_projets/girafe_skin/scenes/giranitex_33_bak3.ma 2008-04-16T21:56:42Z 2008-04-16T21:56:42Z Y_UP 0.040000 0.080000 0.120000 0.160000 0.200000 0.240000 0.280000 0.320000 0.360000 0.400000 0.440000 0.480000 0.520000 0.560000 0.600000 0.640000 0.680000 0.720000 0.760000 0.800000 0.840000 0.880000 0.920000 0.960000 1.000000 1.040000 1.080000 1.120000 1.160000 1.200000 1.240000 1.280000 1.320000 1.360000 1.400000 1.440000 1.480000 1.520000 1.560000 1.600000 1.640000 1.680000 1.720000 1.760000 1.800000 1.840000 1.880000 1.920000 1.960000 2.000000 2.040000 2.080000 2.120000 2.160000 2.200000 2.240000 2.280000 2.320000 2.360000 2.400000 2.440000 2.480000 2.520000 2.560000 2.600000 2.640000 2.680000 2.720000 2.760000 2.800000 2.840000 2.880000 2.920000 2.960000 3.000000 3.040000 3.080000 3.120000 3.160000 3.200000 3.240000 3.280000 3.320000 3.360000 3.400000 3.440000 3.480000 3.520000 3.560000 3.600000 3.640000 3.680000 3.720000 3.760000 3.800000 3.840000 3.880000 3.920000 3.960000 4.000000 4.040000 4.080000 4.120000 4.160000 4.200000 4.240000 4.280000 4.320000 4.360000 4.400000 4.440000 4.480000 4.520000 4.560000 4.600000 4.640000 4.680000 4.720000 4.760000 4.800000 4.840000 4.880000 4.920000 4.960000 5.000000 5.040000 5.080000 5.120000 5.160000 5.200000 5.240000 5.280000 5.320000 5.360000 5.400000 5.440000 5.480000 5.520000 5.560000 5.600000 5.640000 5.680000 5.720000 5.760000 5.800000 5.840000 5.880000 5.920000 5.960000 6.000000 6.040000 6.080000 6.120000 6.160000 6.200000 6.240000 6.280000 6.320000 6.360000 6.400000 6.440000 6.480000 6.520000 6.560000 6.600000 6.640000 6.680000 6.720000 6.760000 6.800000 6.840000 6.880000 6.920000 6.960000 7.000000 7.040000 7.080000 7.120000 7.160000 7.200000 7.240000 7.280000 7.320000 7.360000 7.400000 7.440000 7.480000 7.520000 7.560000 7.600000 7.640000 7.680000 7.720000 7.760000 7.800000 7.840000 7.880000 7.920000 7.960000 8.000000 8.040000 8.080000 8.120000 8.160000 8.200000 8.240000 8.280000 8.320000 8.360000 8.400000 8.440000 8.480000 8.520000 8.560000 8.600000 8.640000 8.680000 8.720000 8.760000 8.800000 8.840000 8.880000 8.920000 8.960000 9.000000 9.040000 9.080000 9.120000 9.160000 9.200000 9.240000 9.280000 9.320000 9.360000 9.400000 9.440000 9.480000 9.520000 9.560000 9.600000 9.640000 9.680000 9.720000 9.760000 9.800000 9.840000 9.880000 9.920000 9.960000 10.000000 10.040000 10.080000 10.120000 10.160000 10.200000 10.240000 10.280000 10.320000 10.360000 10.400000 10.440000 10.480000 10.520000 10.560000 10.600000 10.640000 10.680000 10.720000 10.760000 10.800000 10.840000 10.880000 10.920000 10.960000 11.000000 0.016364 -0.049779 1.881053 0.017034 -0.052046 1.877202 0.018112 -0.055759 1.865879 0.019565 -0.060865 1.847425 0.021359 -0.067310 1.822185 0.023464 -0.075041 1.790502 0.025845 -0.084004 1.752719 0.028471 -0.094146 1.709178 0.031309 -0.105413 1.660224 0.034326 -0.117752

collada.dae


```
{
    "images":[{
        "url":"MarineCv2_color.jpg",
        "uuid":"1F3E9A1C-DDD2-3F4D-80C1-BC5F609DC23B",
        "name":"MarineCv2_color.jpg"
    }],
    "geometries":[{
        "type":"Geometry",
        "uuid":"4181C04A-B75A-3FDC-9A89-89A1D2BCE0AF",
        "data":{
            "uvs":
[[0.270354,0.313478,0.271947,0.319847,0.275729,0.309655,0.274243,0.3087,0.272832,0.307868,0.269884
,0.310585,0.226132,0.380806,0.222211,0.383513,0.22838,0.387803,0.230469,0.384471,0.220584,0.387519
,0.227778,0.393438,0.236551,0.373639,0.235651,0.37412,0.236463,0.379508,0.237612,0.379488,0.239952
,0.371796,0.240752,0.379751,0.245983,0.382396,0.244072,0.372847,0.029897,0.380277,0.026492,0.37821
7,0.024403,0.381866,0.027378,0.383736,0.23588,0.388524,0.23554,0.386494,0.234758,0.383916,0.050509
,0.560407,0.047071,0.557888,0.045167,0.558824,0.048244,0.563377,0.064365,0.566106,0.055853,0.56226
3,0.05439,0.567096,0.064598,0.572284,0.04127,0.560623,0.045026,0.567068,0.051922,0.572939,0.058767
,0.576525,0.037322,0.561994,0.040429,0.570751,0.047131,0.578106,0.034887,0.563887,0.036649,0.57368
8,0.041729,0.581221,0.051832,0.587997,0.054417,0.583009,0.033166,0.564334,0.033893,0.574381,0.0382
76,0.583055,0.046687,0.590068,0.029228,0.564585,0.030582,0.575915,0.035745,0.587145,0.043417,0.594
797,0.012878,0.575167,0.025786,0.576822,0.024107,0.563407,0.011428,0.55775,0.019835,0.589523,0.031
426,0.590171,0.056273,0.542792,0.059073,0.545878,0.065176,0.5447,0.063035,0.542068,0.048447,0.5468
91,0.053936,0.549727,0.04472,0.552324,0.048873,0.553322,0.055773,0.539823,0.044725,0.545943,0.0644
16,0.538785,0.040454,0.553447,0.054384,0.536842,0.041786,0.545212,0.065807,0.534961,0.037489,0.554
075,0.049928,0.535922,0.039805,0.54519,0.035552,0.554559,0.046737,0.533753,0.037925,0.544124,0.033
877,0.554064,0.043337,0.529892,0.034539,0.543152,0.030422,0.55413,0.04709,0.510318,0.042159,0.5067
9,0.026639,0.514508,0.031141,0.519839,0.038545,0.525958,0.02404,0.532227,0.030178,0.540512,0.01661
9,0.542977,0.02613,0.552544,0.019243,0.52681,0.072806,0.54418,0.068036,0.541536,0.069568,0.544295,
0.072537,0.546332,0.074457,0.542001,0.070694,0.540527,0.075113,0.538587,0.072083,0.53632,0.070949,
0.565762,0.071836,0.562397,0.067553,0.560436,0.064941,0.562816,0.080185,0.557515,0.078931,0.555483
,0.076466,0.562489,0.076226,0.565704,0.058861,0.594811,0.064195,0.589419,0.069791,0.550726,0.06658
1,0.555646,0.071219,0.555536,0.07257,0.547845,0.061829,0.558444,0.058373,0.559956,0.058247,0.55486
4,0.053888,0.556612,0.529304,0.679074,0.528585,0.691852,0.525233,0.680917,0.521958,0.67057,0.00535
1,0.580703,0.020611,0.59504,0.002847,0.554083,0.040637,0.597389,0.01118,0.540222,0.054475,0.599187
,0.046209,0.601951,0.073623,0.5477,0.073842,0.556482,0.06595,0.597735,0.05175,0.602881,0.062732,0.
603138,0.513246,0.652152,0.507764,0.643571,0.505233,0.633772,0.522766,0.650073,0.06824,0.506816,0.
075348,0.508814,0.073568,0.50543,0.068519,0.503123,0.062065,0.55231,0.067781,0.548426,0.055314,0.5
17723,0.050992,0.513294,0.062031,0.506743,0.064039,0.51368,0.068682,0.511233,0.061823,0.58071,0.08
```


.json

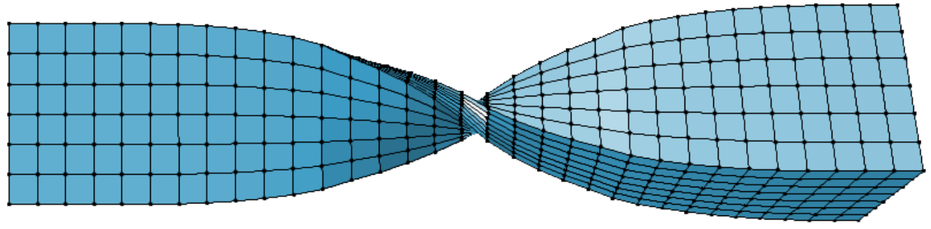
Linear Blend Skinning - Limitations

- Non-trivial rigging settings
- Artifacts for large rotations: Candy wrapper, Collapsing ellbow

Candy wrapper

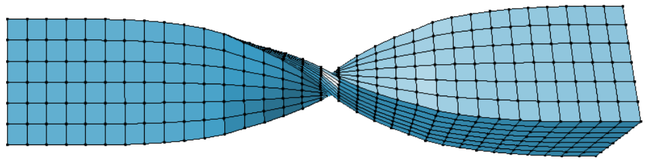
Collapsing elbow

Linear blending between rigid transformation matrices



Skinning improvement

Avoid linear blending b/w affine transform matrices

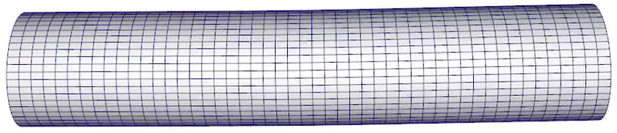


First idea: Split rigid transformation matrices in

- rotation part: blend using quaternions (solve candy wrapper)
- translation part: blend linearily

But doesn't work for general deformation: Arbitrary center of rotation

Rotation and translations are treated separately



Possibility to compute an optimal center of rotation at each frame

[Spherical Blend Skinning: A Real-time Deformation of Articulated Models. L. Kavan, J. Zara. I3D 2005.]

Skinning improvement: Dual Quaternion

Second idea: Use of **dual quaternion**

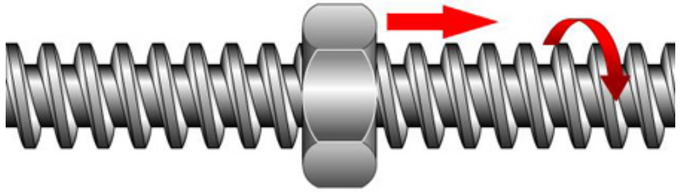
$\hat{q} = q_0 + \epsilon\, q_\epsilon$, ϵ dual element.

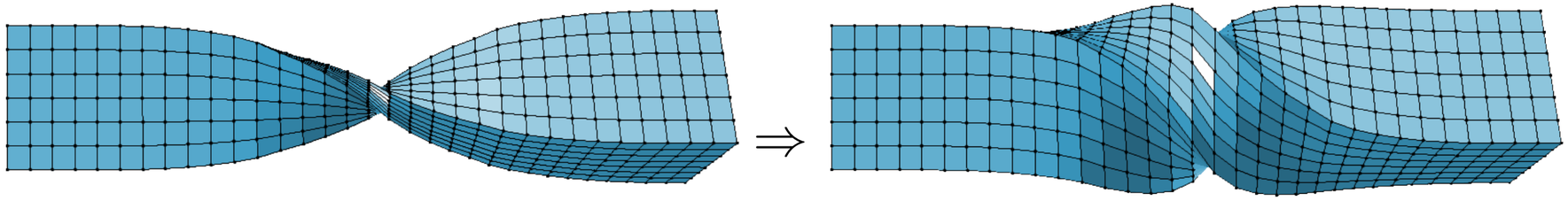
Dual quaternion = generalization of quaternion to handle *rigid transformation*

Rotation + translation

Unit dual quaternion models rigid transformation as **screw motions**

= *rotation about an axis followed by a translation in the direction of this axis*





LBS *DQS*

Dual number and dual quaternion

Dual number $a = a_0 + \epsilon a_\epsilon$

Dual element ϵ is nilpotent : $\epsilon \neq 0, \epsilon^2 = 0$

ϵ commonly used to model infinitesimal quantity (ex. automatic differentiation)

Dual quaternion $\hat{q}$ = Generalization of quaternion to dual numbers

$\hat{q} = q_0 + \epsilon\, q_\epsilon$

- q_0: pure rotation component
- q_ϵ: encodes translation component (dual part)

Given a unit quaternion q_0, and a translation $t = (t_x, t_y, t_z)$, the associated unit dual quaternion is

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

Unit dual quaternion ($\|\hat{q}\| = 1$) describe the set of rigid transformation as screw motion.

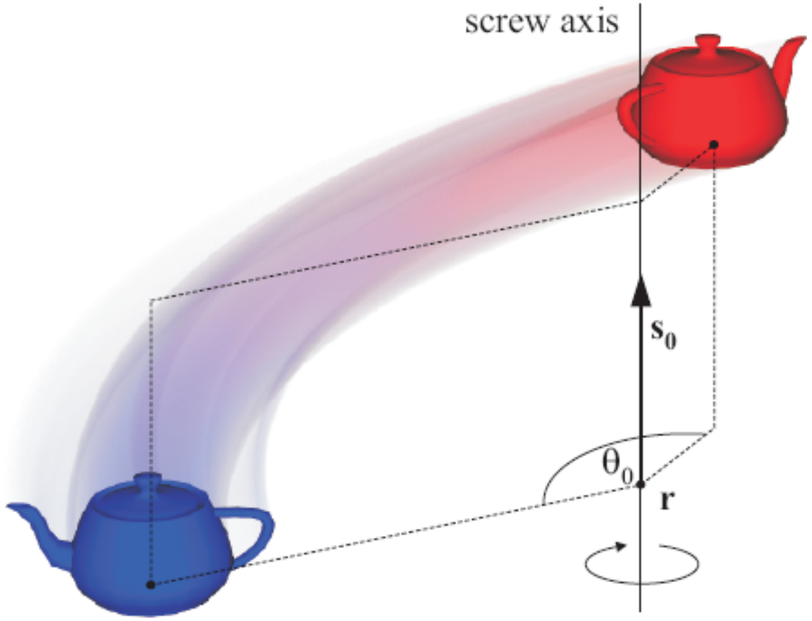
Axis-angle representation with dual quaternion

Similarily to quaternion: "angle/axis" correspondance

- $\hat{q} = \cos(\hat{\theta}/2) + \hat{n}\ \sin(\hat{\theta}/2)$

- $\hat{\theta} = \theta_0 + \epsilon\theta_\epsilon$
  - θ_0: angle of rotation
  - $\theta_\epsilon = t \cdot n_0$: quantity of translation along n_0

- $\hat{n} = n_0 + \epsilon\, n_\epsilon$
  - n_0: axis of rotation
  - $n_\epsilon = \frac{1}{2}\left((n_0 \times t)\, cotan(\theta_0/2) + t\right) \times n_0$: called the moment of the rotation axis.



Convert dual quaterion to rotation/translation

Given a non-unit dual quaternion as input $\hat{q}' = q'_0 + \epsilon q'_\epsilon$

How to compute the components of $\hat{q}'/\|\hat{q}'\| = \hat{q} = q_0 + \epsilon q_\epsilon$?

Express the parameterization in rotation-translation:

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

First, normalize the non dual component: $\hat{q}'/\|q'_0\|$

$$\Rightarrow \boxed{q_0 = q'_0/\|q'_0\|}$$

Second, enforce the parameterization of the dual component: $\frac{1}{2} q_t \, q_0 = q'_\epsilon/\|q'_0\|$

$$\Rightarrow \boxed{q_t = 2\, q'_\epsilon \, q_0^\star/\|q'_0\|}$$

Dual Quaternion Skinning (DQS)

1- Encode rigid transformation (q^i, t^i) into dual quaternion

$$\hat{q}^i = q_0^i + \frac{\epsilon}{2} q_t^i \, q_0^i \,, \quad q_t^i = (t_x^i, t_y^i, t_z^i, 0)$$

2- Compute blending in the dual quaternion space (ScLERP)

$$\hat{q}' = \sum_i \omega_i \, \hat{q}^i = q'_0 + \epsilon \, q'_\epsilon$$

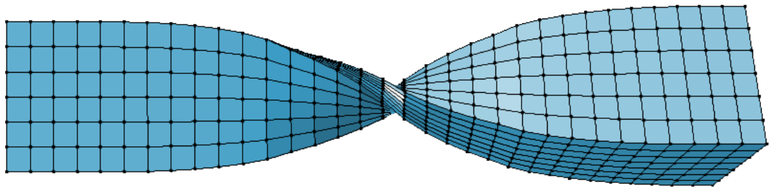
3- Extract components (q, t) from $\hat{q}'$

- $q = q'_0 / \|q'_0\|$
- $(t_x, t_y, t_z, 0) = 2 \, q'_\epsilon \, q^\star$

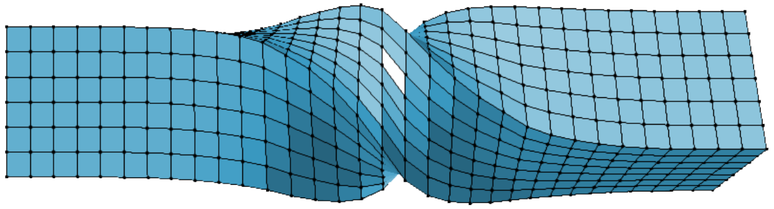
$q^\star$: conjugate of q.

4- Apply finaly the transformation (q, t) to position p_0

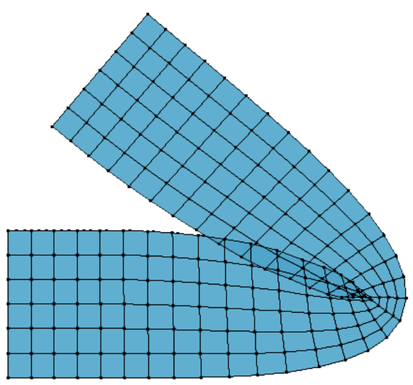
[Skinning with Dual Quaternions. Kavan et al. I3D, 2007]



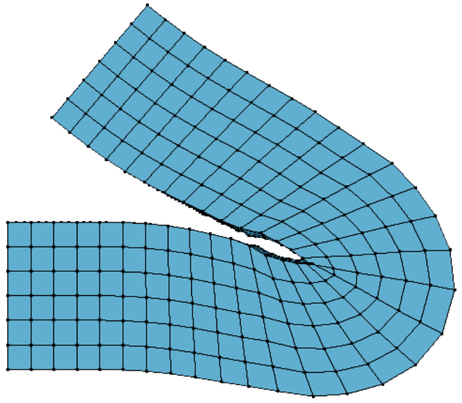
LBS



DQS



LBS



DQS

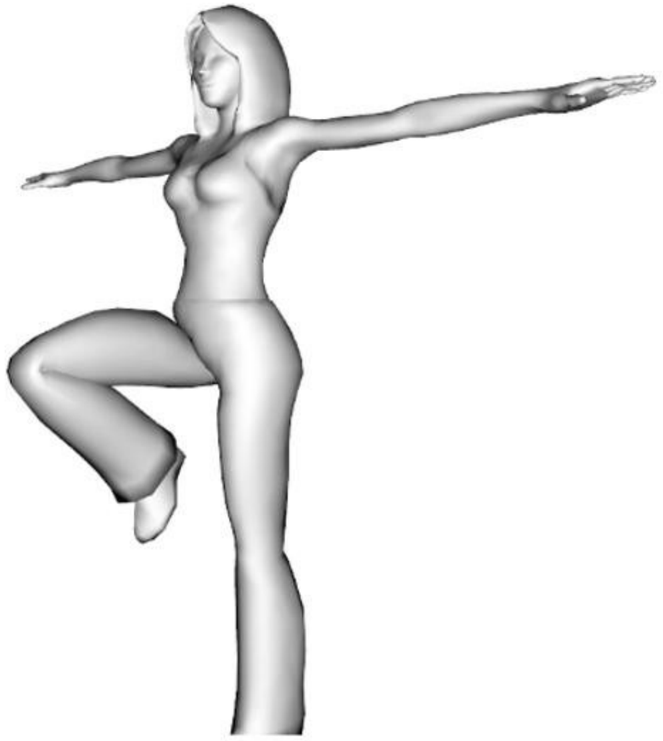
Dual Quaternion VS LBS

Dual quaternion

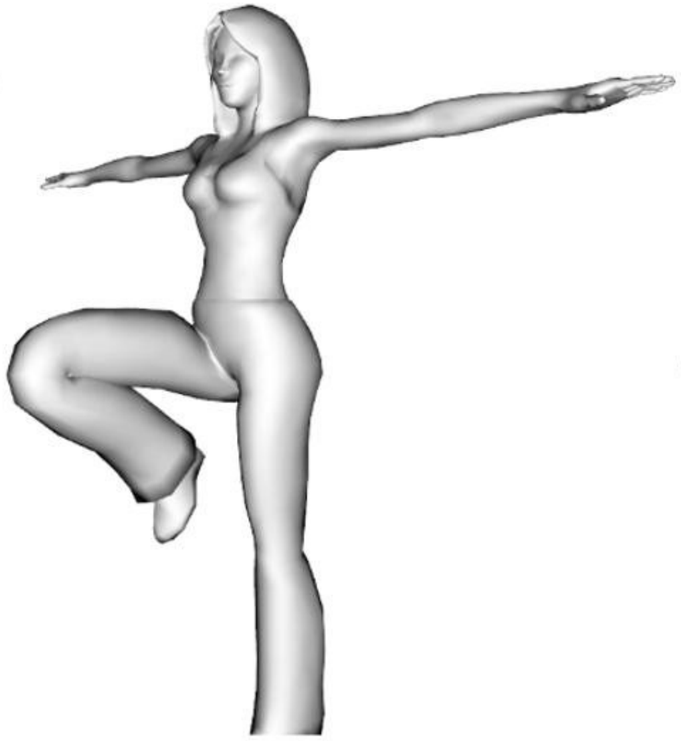
- (+) Fully solves *candy wrapper* artifact
- (+) Almost as efficient as LBS
- (-) May create artificial/unwanted bulge

Not always prefered to LBS

Both solutions (LBS, DQS) are proposed in standard tools



LBS



DQS