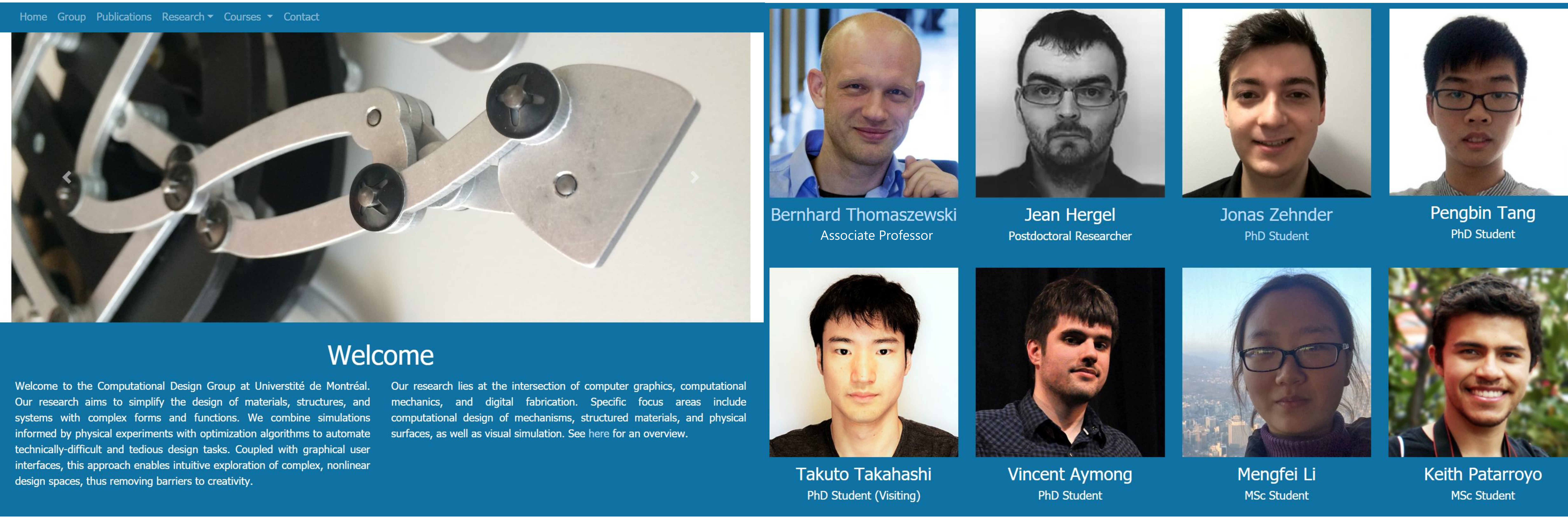
Computer animation: Rigging, FK and IK

Computational Design Group

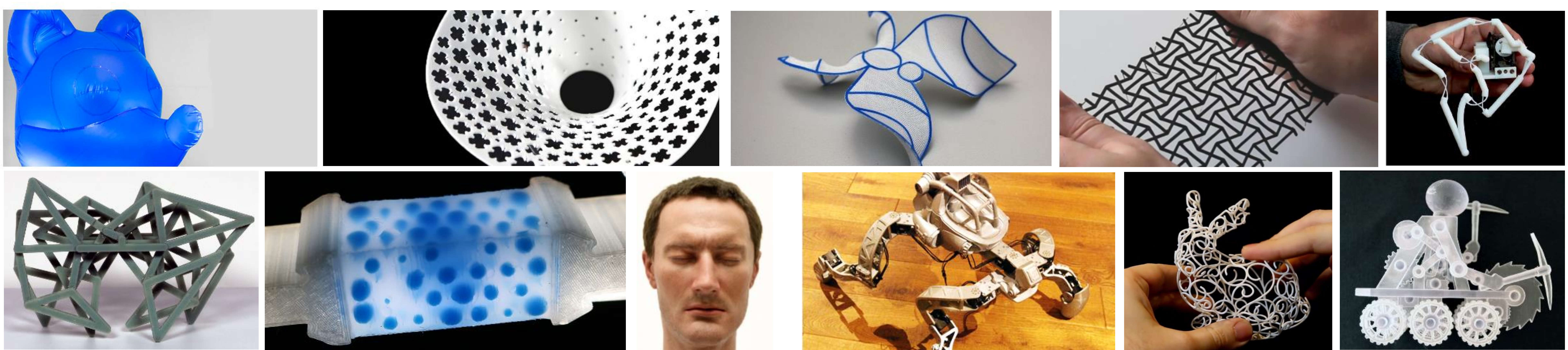
Simulation & Optimization-Driven Design

Université de Montréal



Research Themes

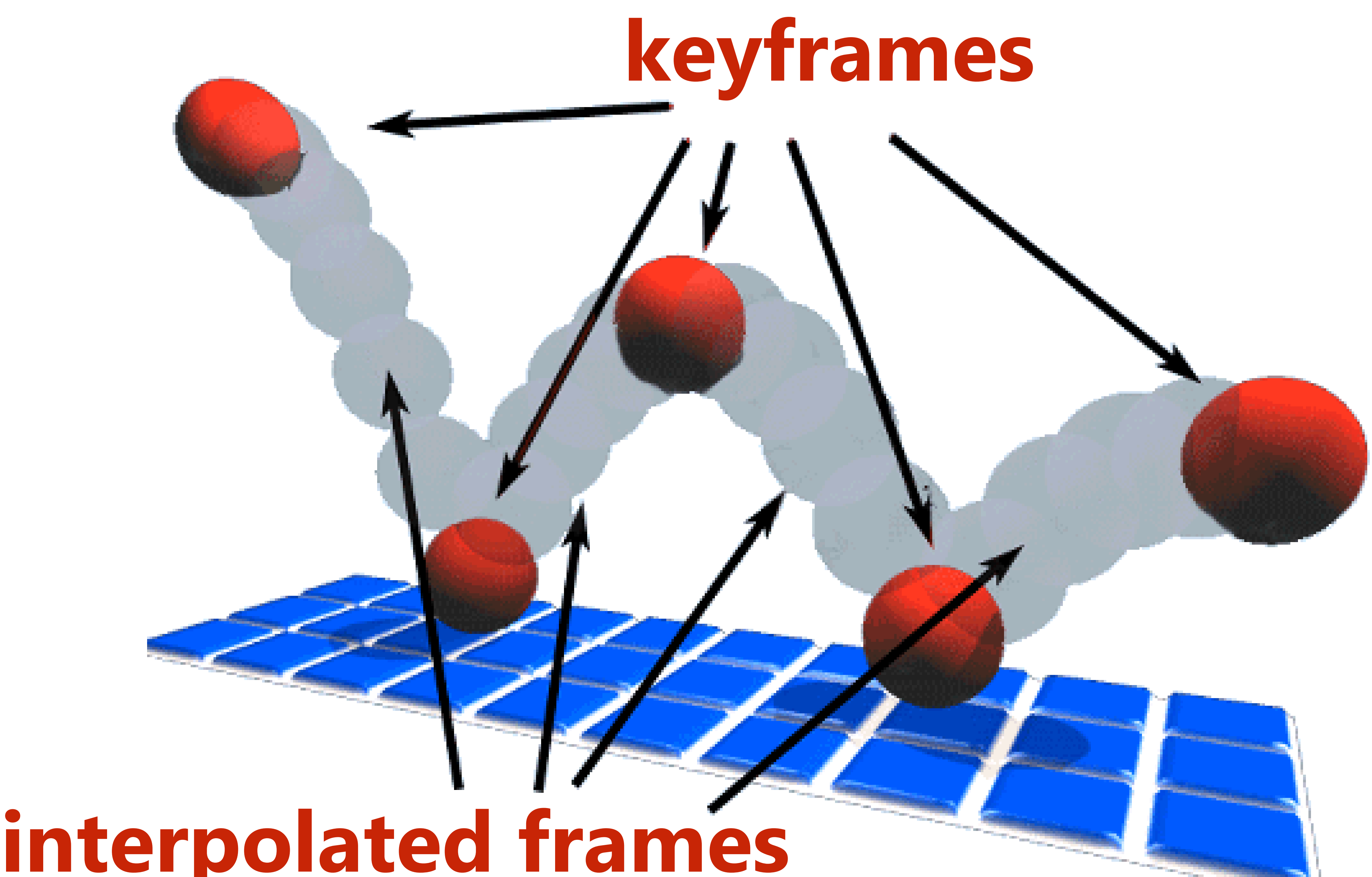
Physical Surface **Structured Materials** **Mechanism Design** **Robotics** **Visual Simulation**



Keyframing

Basic idea

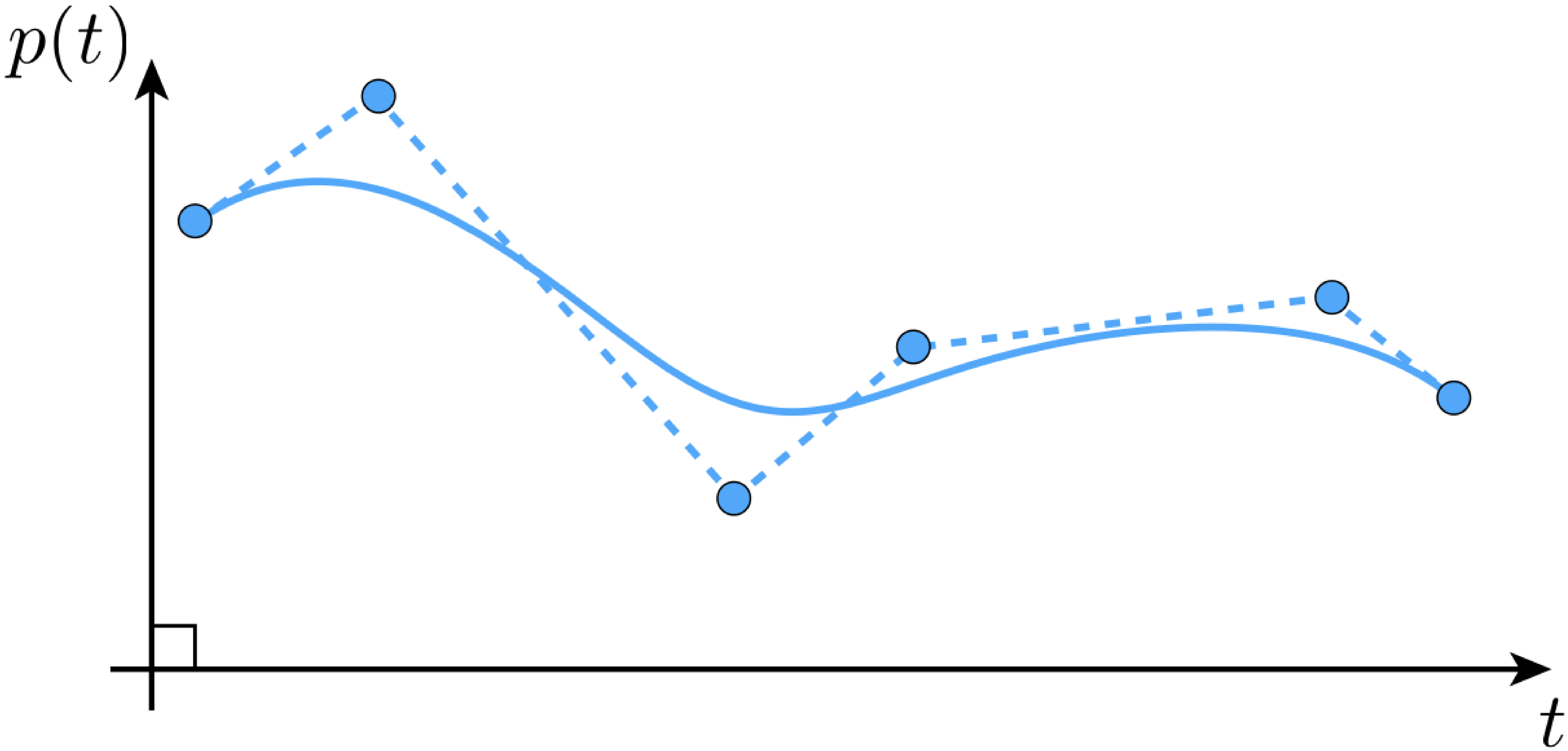
- user specifies important events (*sparse in time*)
- computer fills in the rest via interpolation or approximation



How do you interpolate data?

Splines

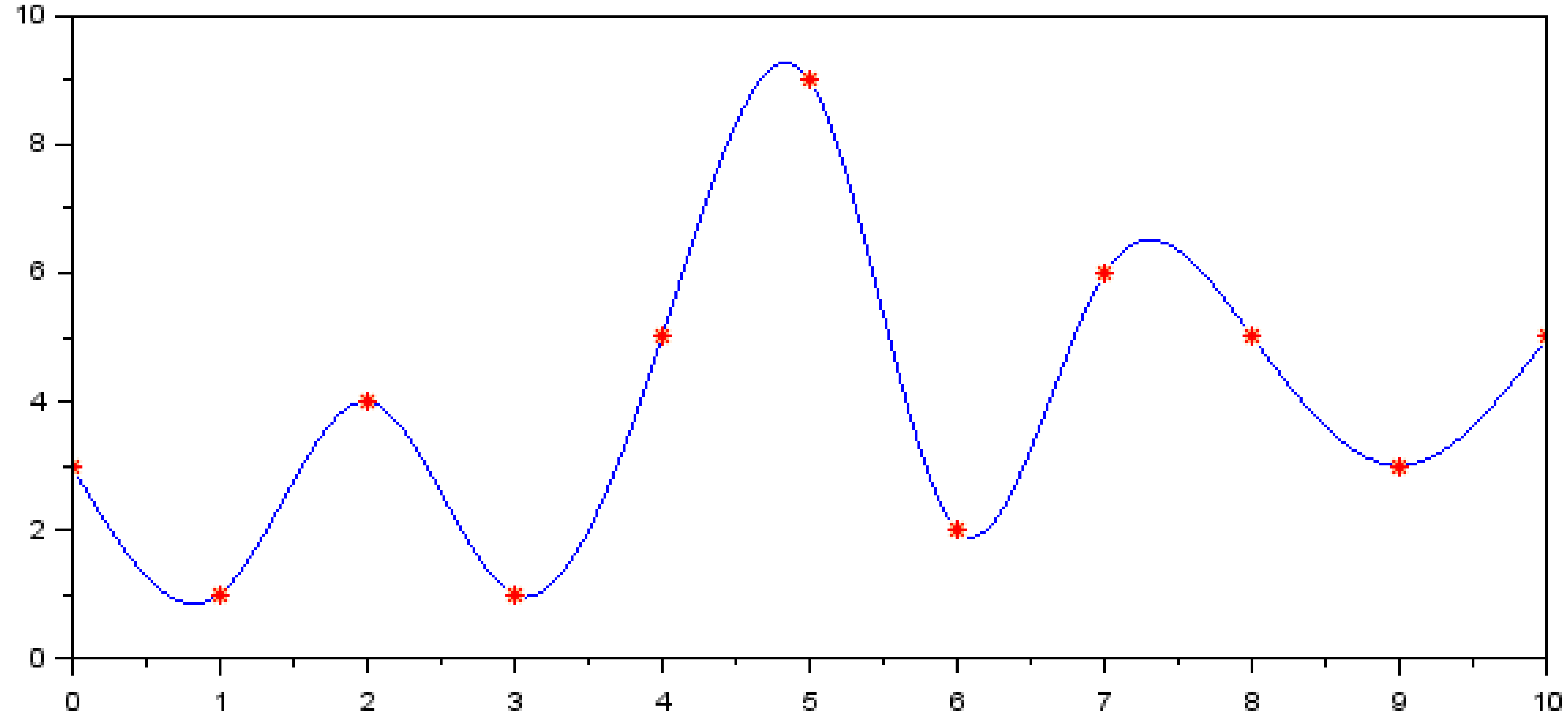
- A curve defined by piecewise polynomials
- Cubic polynomials are often used in graphics



- Q: What are the three properties that must be considered when deciding which type of spline to use?

Splines

- *S*pline defined *piecewise* from cubic polynomials p_i



- Polynomials p_i are defined on intervals $[t_i, t_{i+1}]$

$$p(t) = \begin{cases} p_1(t) & t_1 \leq t < t_2 \\ & \dots \\ p_n(t) & t_n \leq t < t_{n+1} \end{cases}$$

Splines

- Smoothness of p_i determines smoothness of p *within* segments.
- What about smoothness *across* segments?

For each segment, we want

- interpolation at both endpoints (*C^0 continuity*)

$$p_i(t_i) = f_i,\ p_i(t_{i+1}) = f_{i+1},\ i = 0, \dots, n-1$$

- tangents to agree at endpoints (*C^1 continuity*)

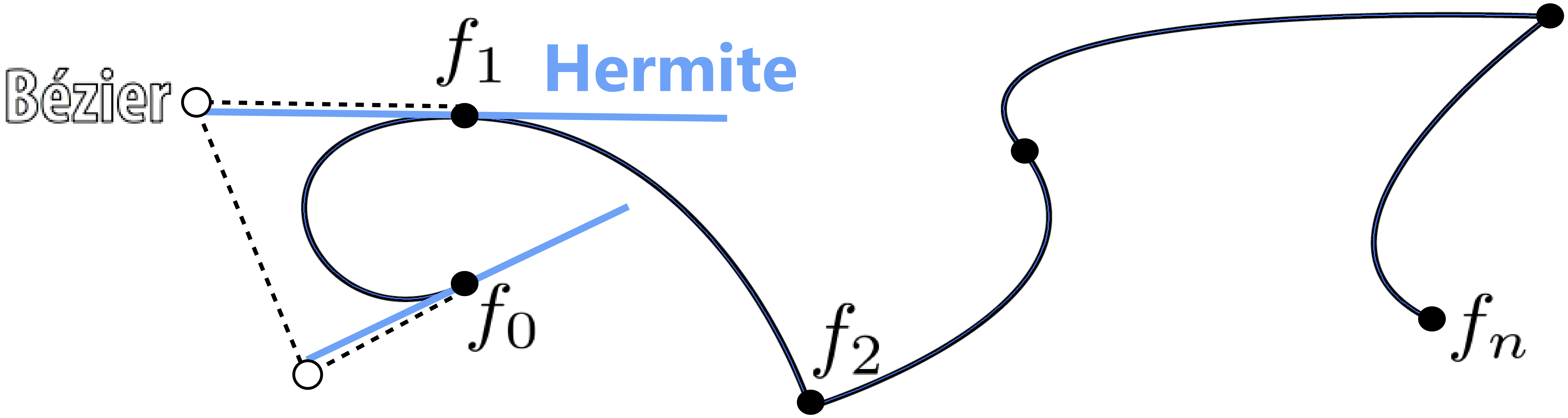
$$p_i'(t_{i+1}) = p'_{i+1}(t_{i+1}), \qquad i = 0, \dots, n-2$$

- same curvature at endpoints (?) (*C^2 continuity*)

$$p_i''(t_{i+1}) = p''_{i+1}(t_{i+1}), \qquad i = 0, \dots, n-2$$

Hermite/Bézier Splines

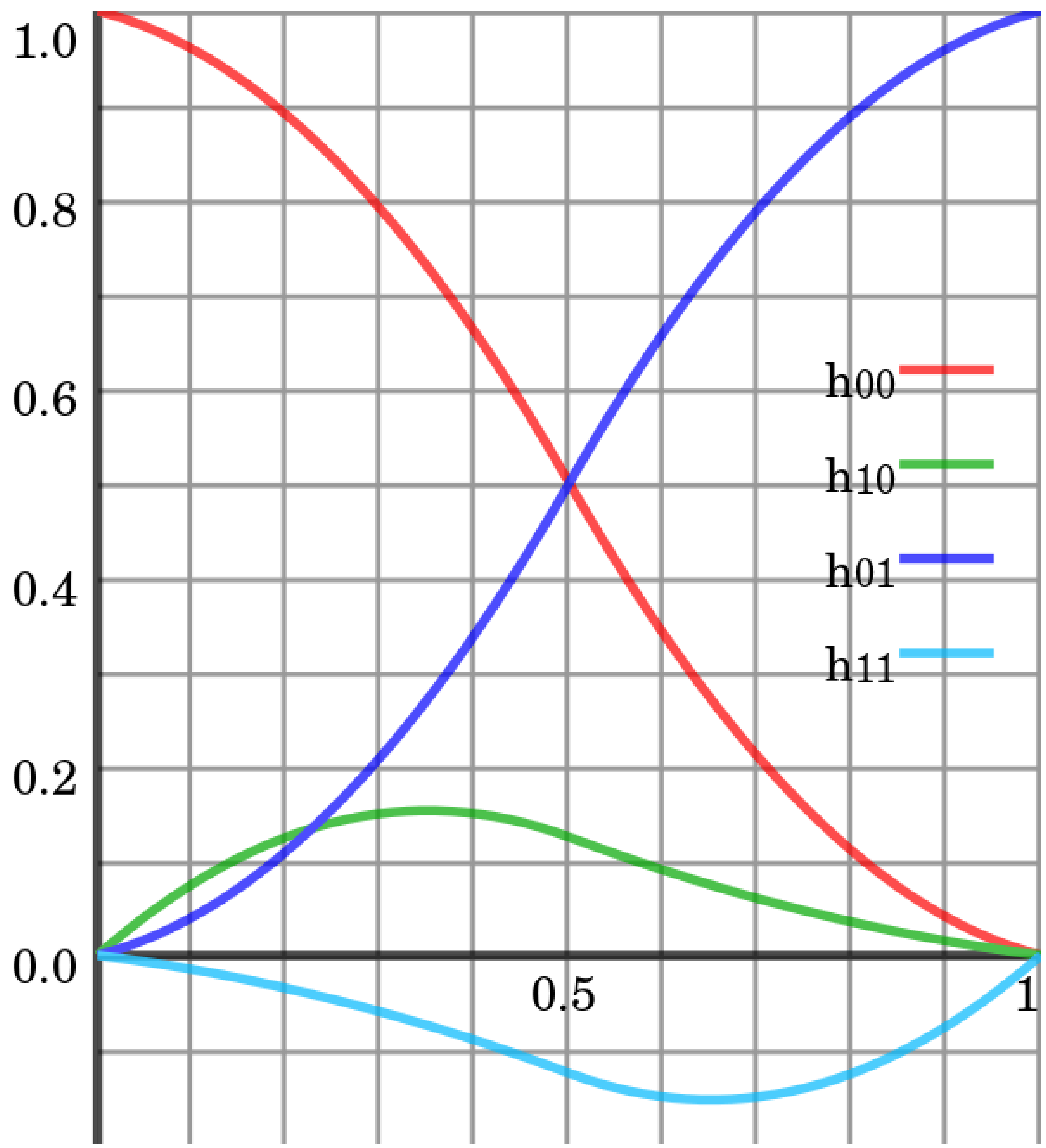
- Each cubic "piece" specified by endpoints and tangents



- Equivalently: by four points (Bézier form)
- Commonly used for 2D vector art (Illustrator, SVG, ...)
- Can we get tangent continuity?
- Sure: set tangents to same value on both sides of knot!
  - E.g., f_1 above, but not f_2

Hermite/Bézier Splines as sums of basis functions

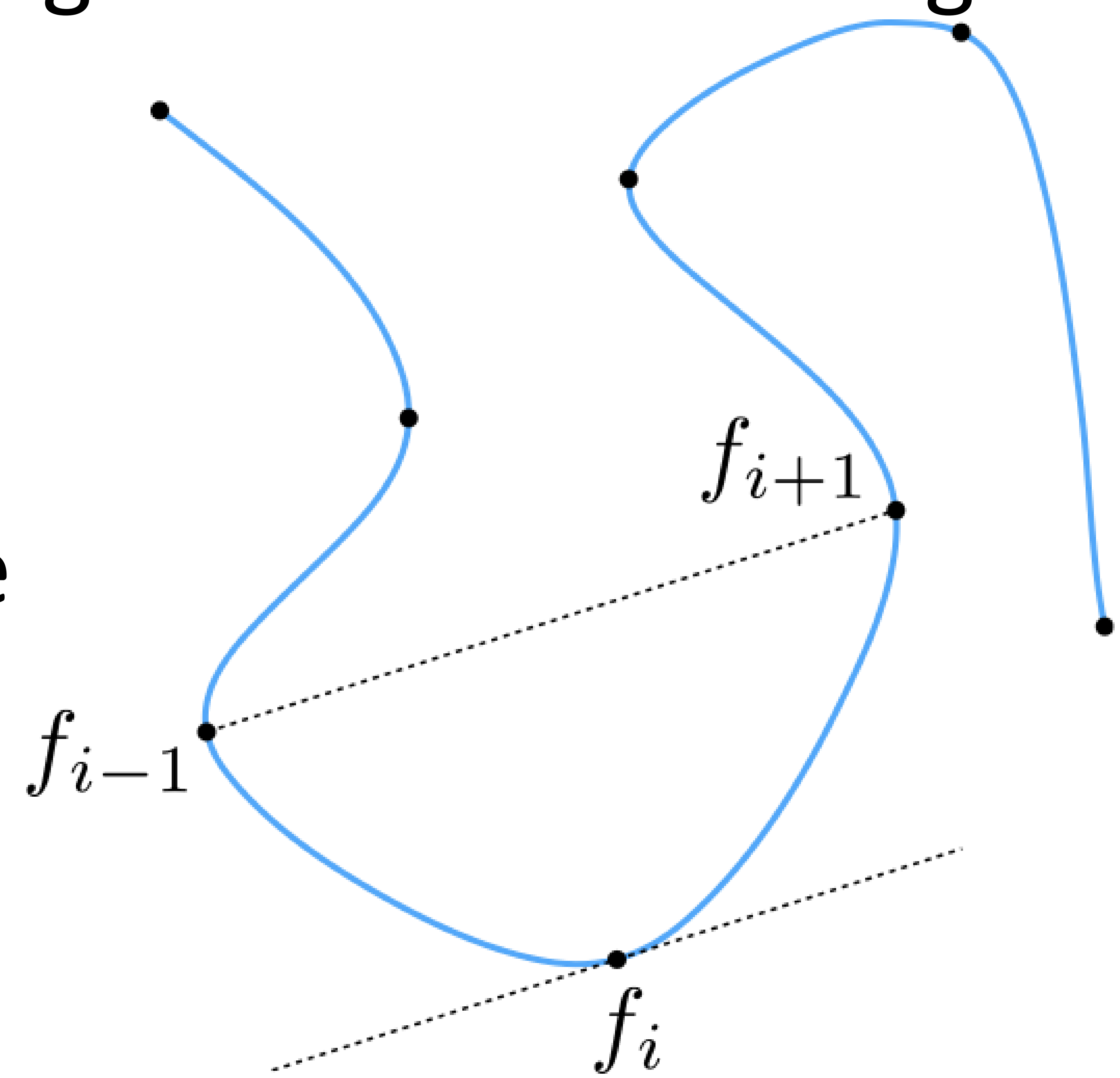
$$\boldsymbol{p}(t) = \underbrace{(2t^3 - 3t^2 + 1)}_{\color{red}{h_{00}}}\boldsymbol{p}_0 + \underbrace{(t^3 - 2t^2 + t)}_{\color{green}{h_{10}}}\boldsymbol{m}_0 + \underbrace{(-2t^3 + 3t^2)}_{\color{blue}{h_{01}}}\boldsymbol{p}_1 + \underbrace{(t^3 - t^2)}_{\color{cyan}{h_{11}}}\boldsymbol{m}_1$$



Catmull-Rom Splines

- Sometimes makes sense to specify *tangents,* but often more convenient to just specify *values*
- *Catmull-Rom Splines*: specialization of Hermite spline, determined by values alone
- Basic idea: use difference of neighbors to define tangent

$$u_i := \frac{f_{i+1} - f_{i-1}}{t_{i+1} - t_{i-1}}$$



- All the same properties as any other Hermite spline
- Commonly used to interpolate motion in computer animation.
- Many, many variants, but Catmull-Rom is usually good starting point

Spline Desiderata, Revisited

	INTERPOLATION	CONTINUITY	LOCALITY
natural	YES	YES	NO
Hermite	YES	NO	YES
???	NO	YES	YES

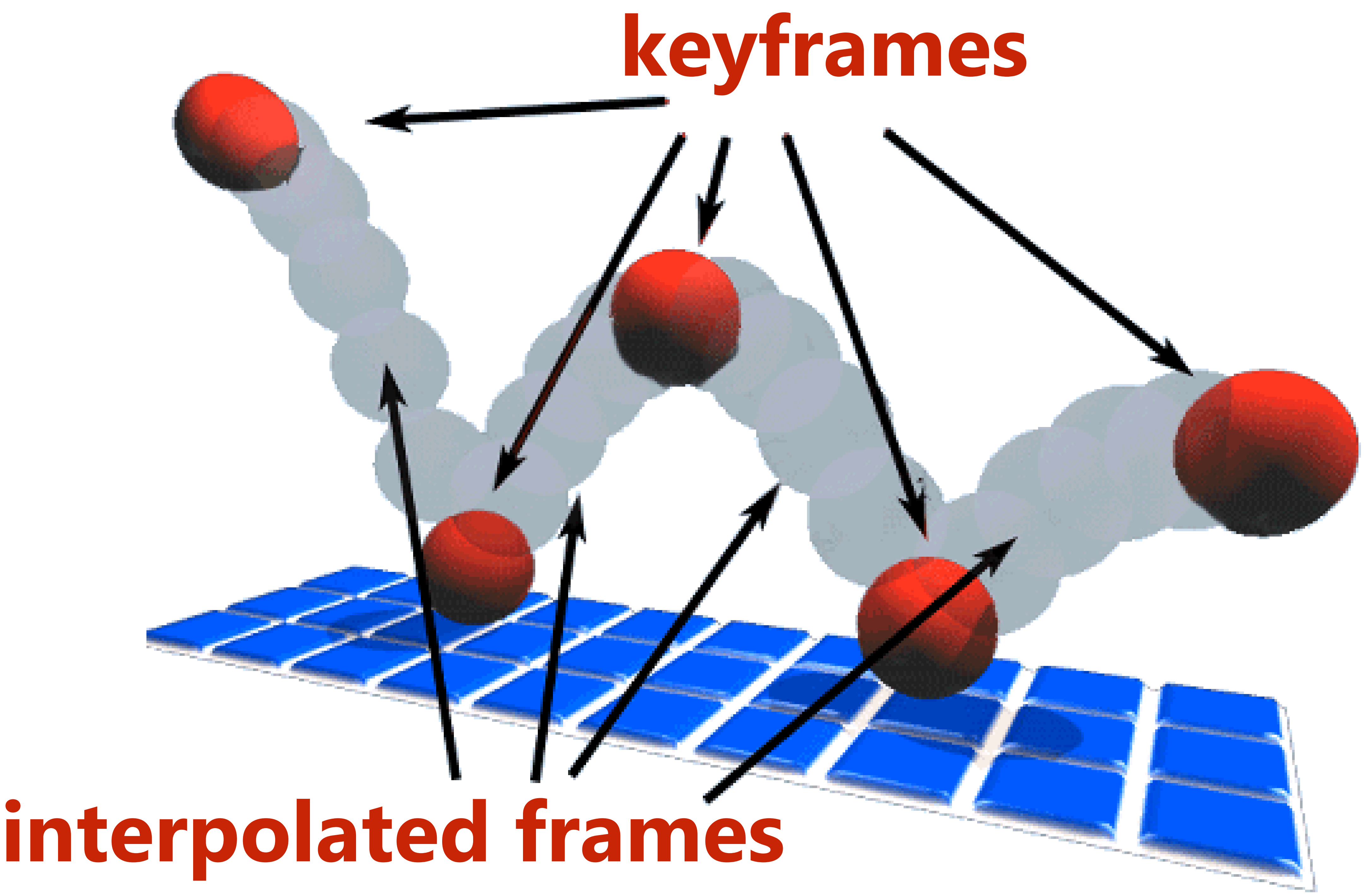
Spline Desiderata, Revisited

	INTERPOLATION	CONTINUITY	LOCALITY
natural	YES	YES	NO
Hermite	YES	NO	YES
B-Splines	NO	YES	YES

What exactly are we interpolating?

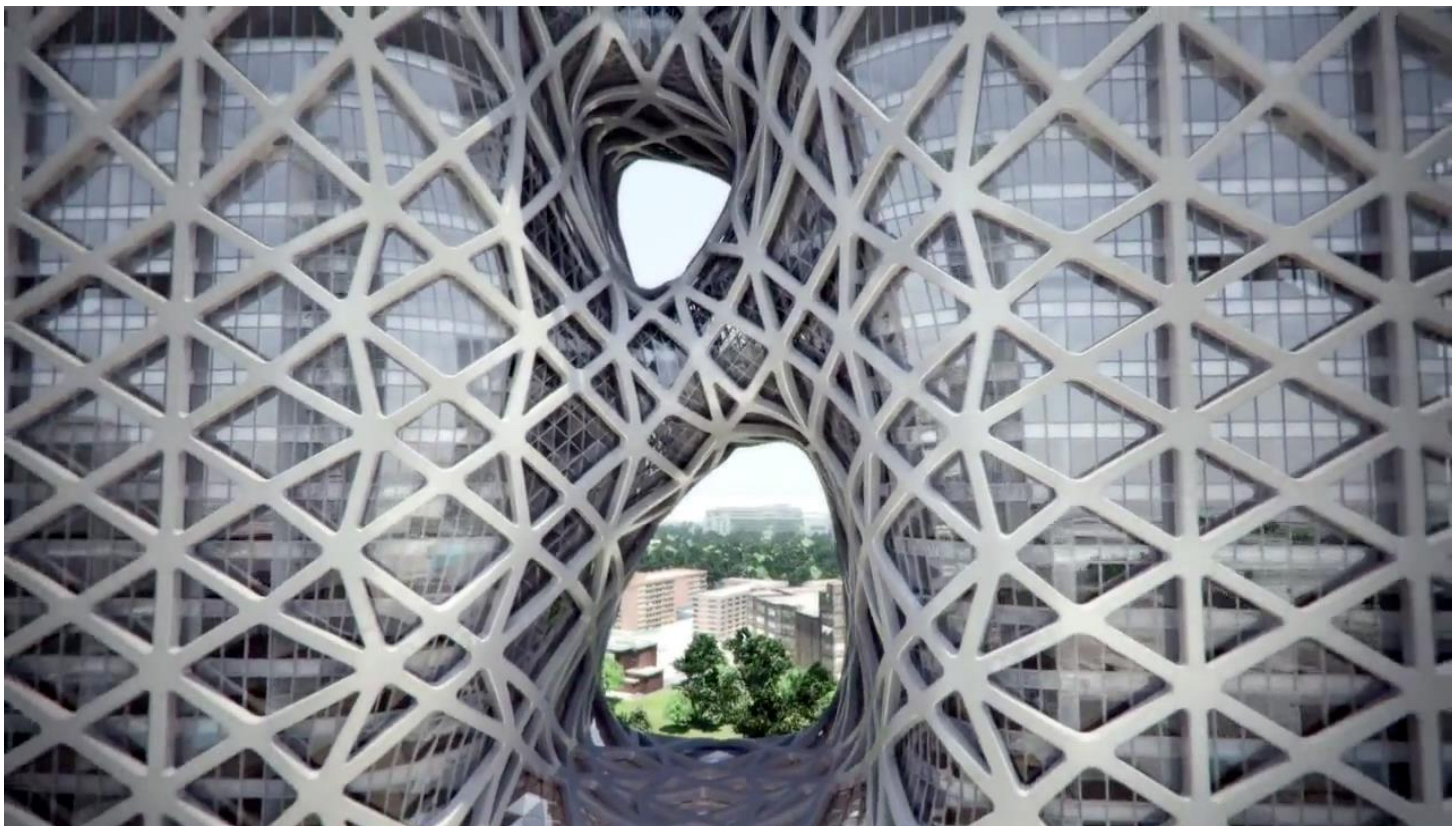
A simple example

- Position of ball over time



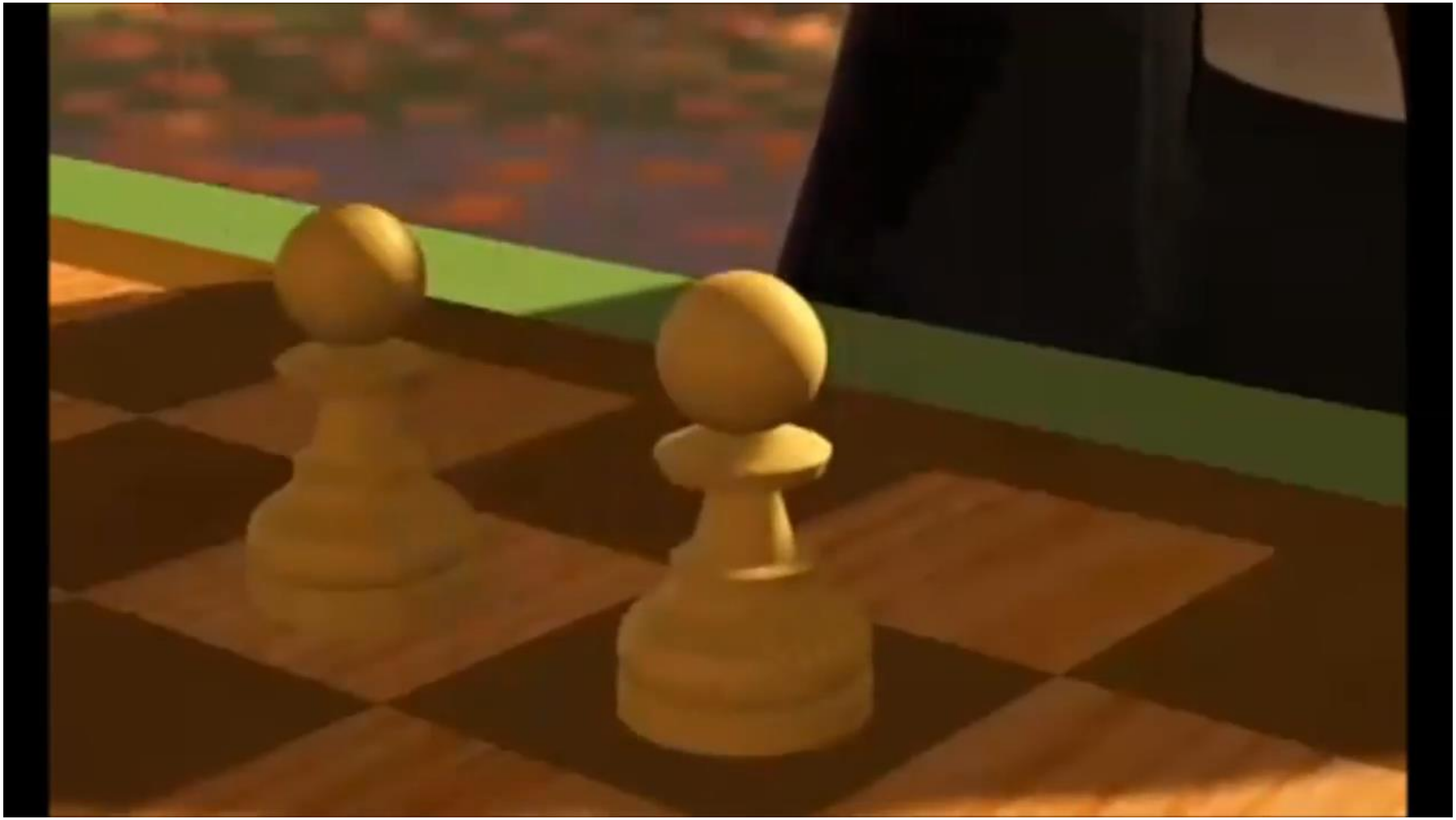
Camera paths

- Animate position, view direction, "up" direction
  - each path is a function $f(t) = (\, x(t), y(t), z(t) \,)$
  - each component (x,y,z) is a spline curve



Zaha Hadid Architects—City of Dreams Hotel Tower

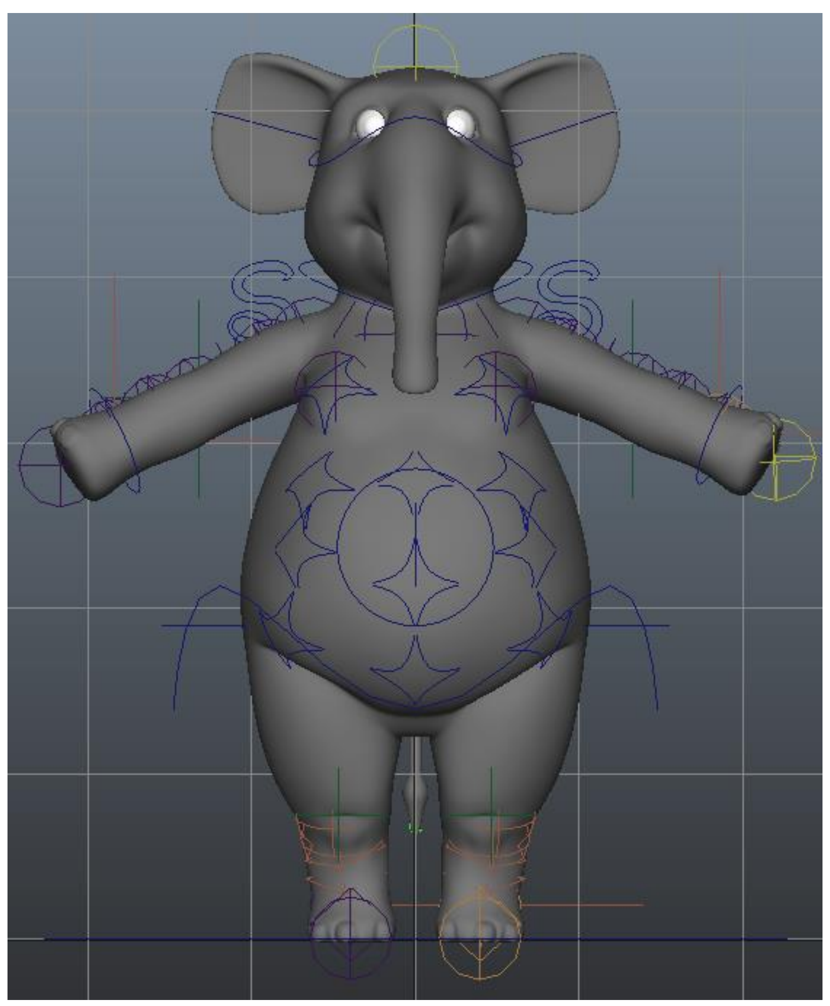
Character Animation



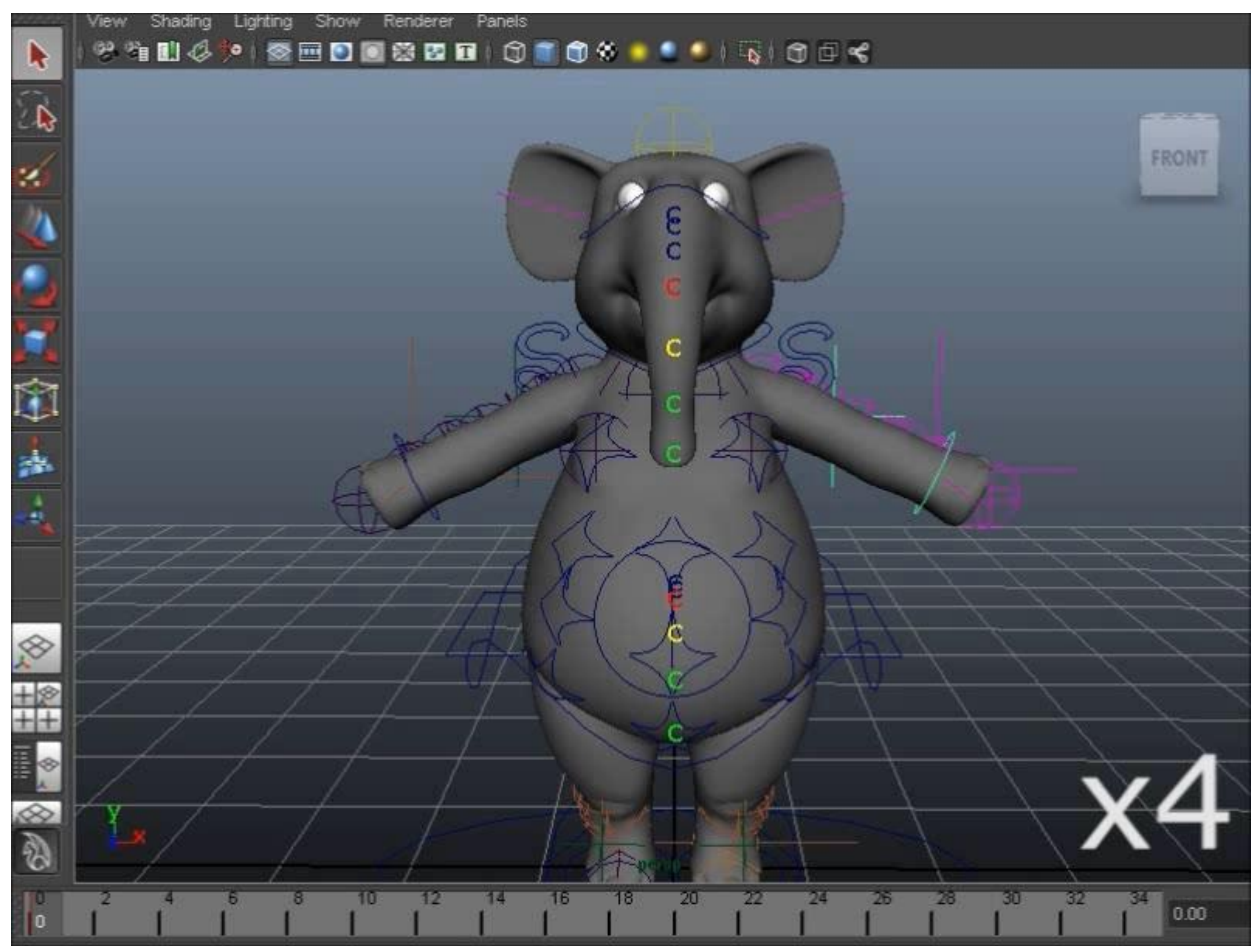
Geri's Game. Pixar, 1997

Character Animation Pipeline

Modeling

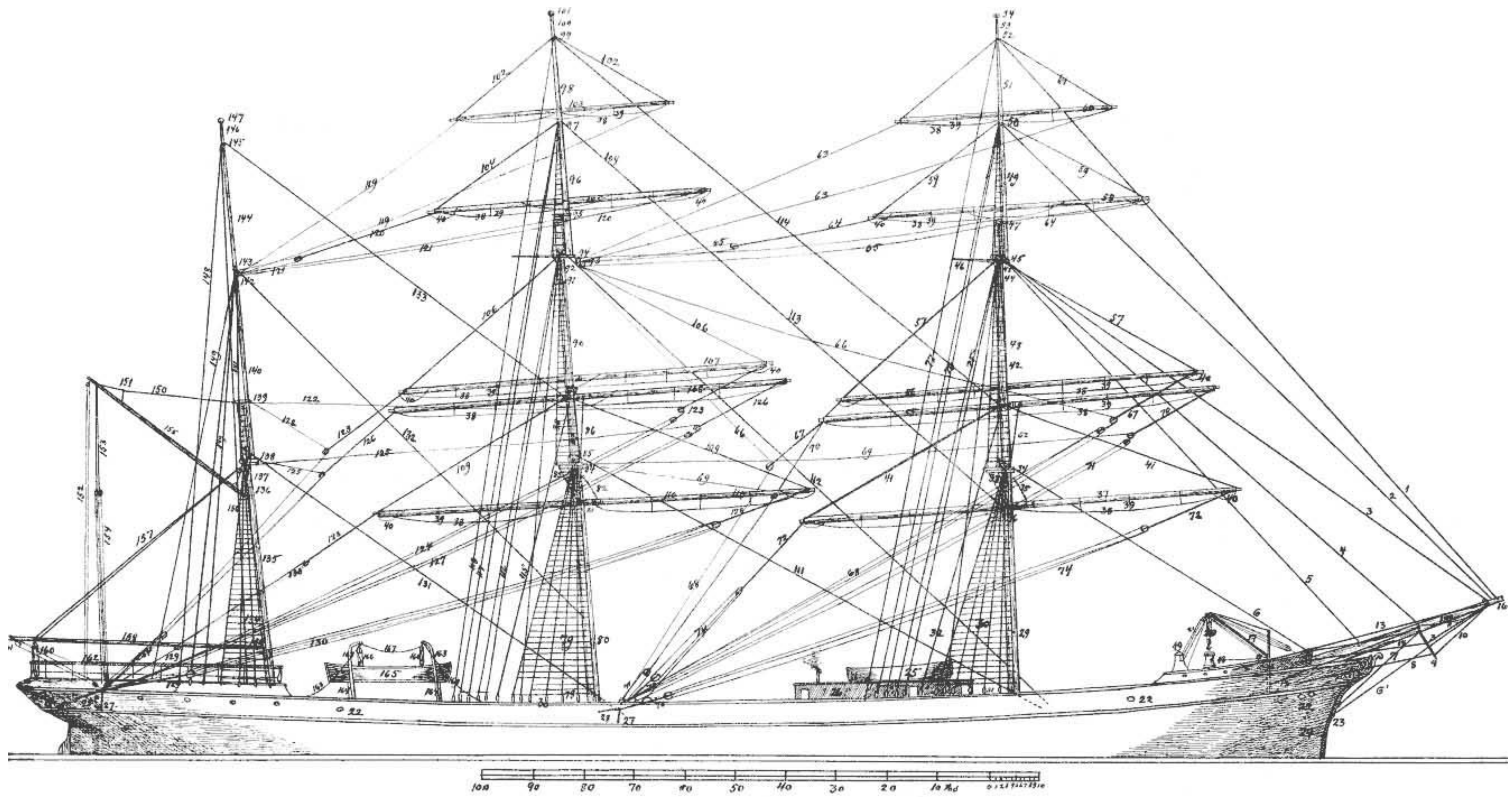


Rigging



Animation

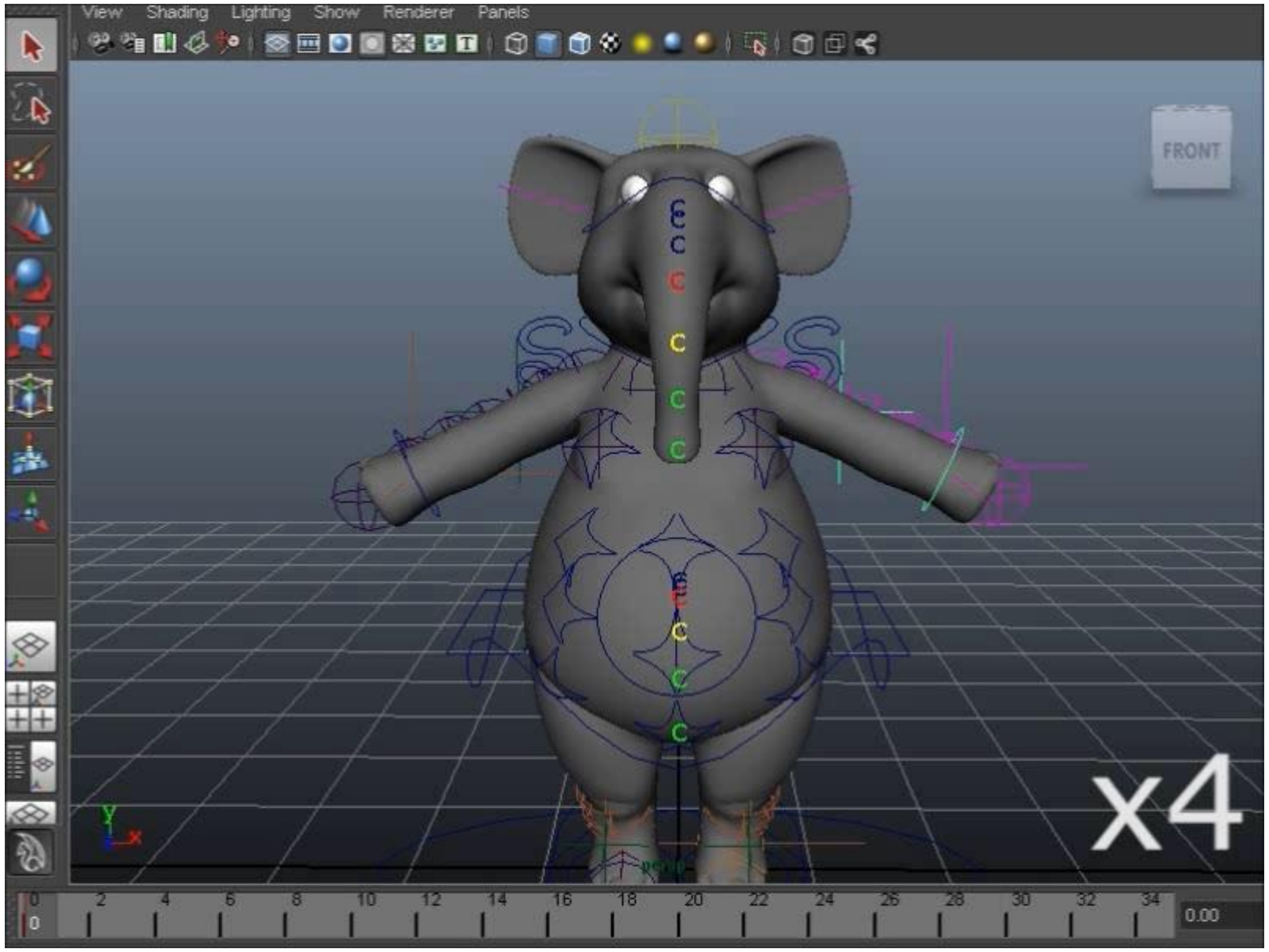
Rigging



Et Barkskibs staaende Rigning Fig. 361.

- Maritime: *Rigging comprises the system of ropes, cables and chains, which support a sailing ship's masts and which adjust the position of the vessel's sails to which they are attached.*

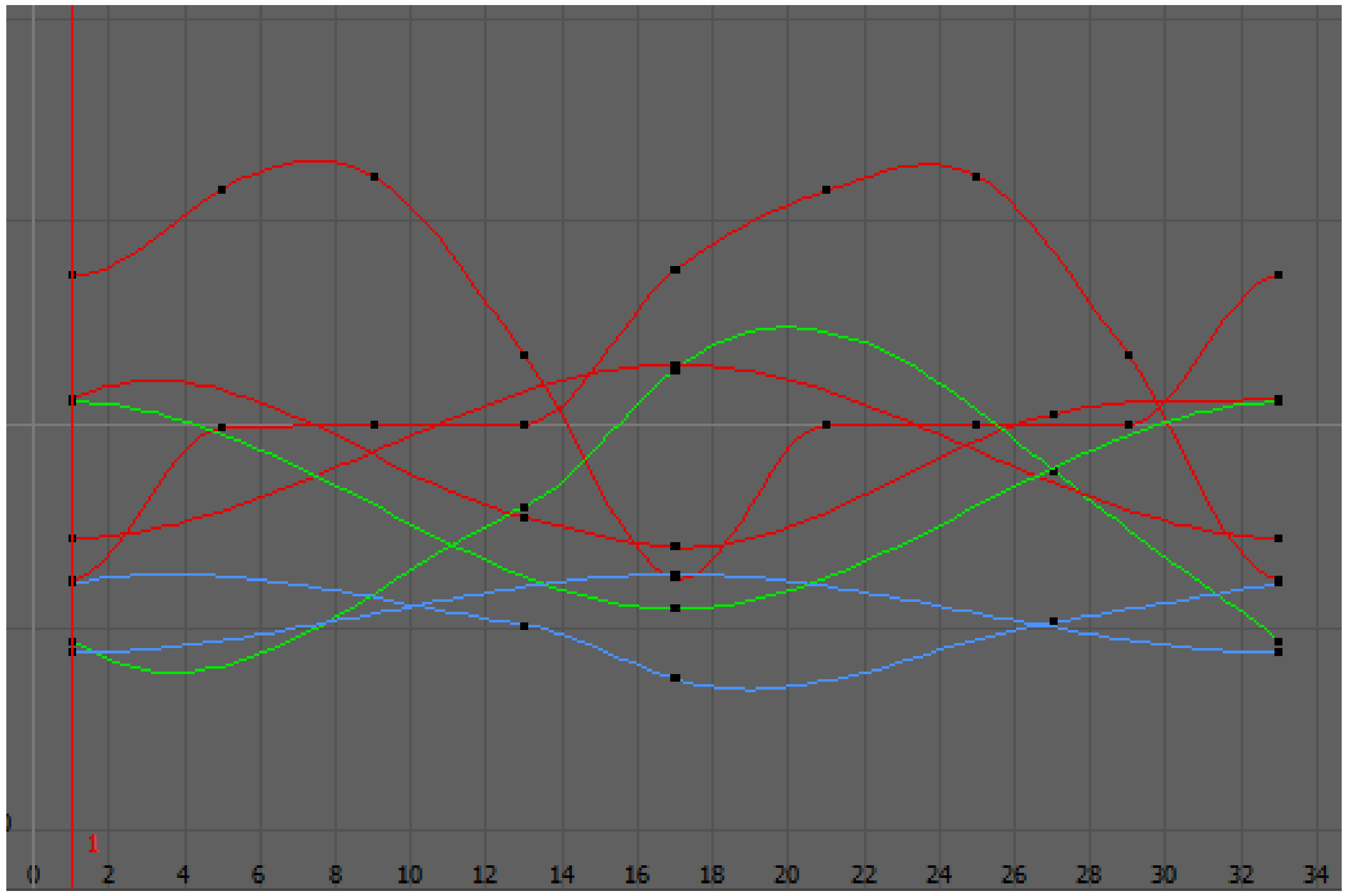
Rigging



- Animation: *A rig is a user-defined mapping between a small number of parameters and the deformations of a high-res mesh. Rigging is the process/task of creating a rig.*

Animating Rigs

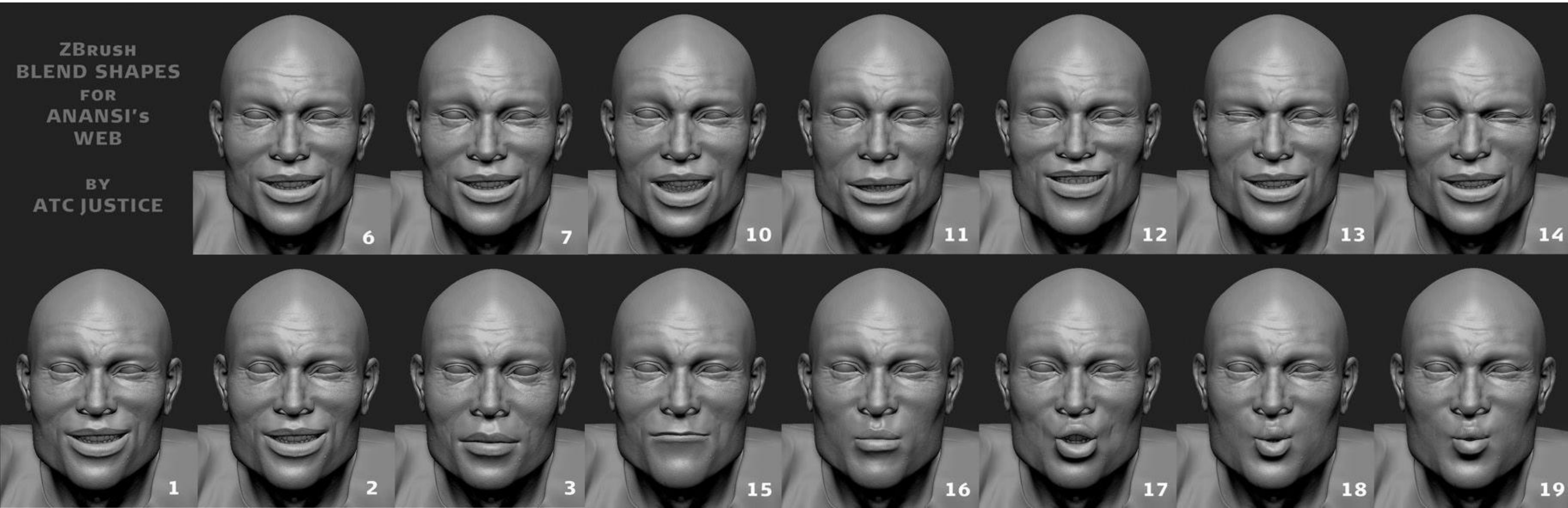
- An animation of a character is defined through a set of (spline) curves, determining how rig parameters evolve over time.



Blend Shape Rigs

- Simplest type of rig
- Input: set of meshes M_i with vertices $\boldsymbol{x}_i^j$ and blending weights $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)$
- Output: *blended* mesh M through linear combination

$$M = \sum_i \alpha_i M_i \quad \text{i.e.} \quad \boldsymbol{x}^j = \sum_i \alpha_i \boldsymbol{x}_i^j$$



If you want to reach me

Organizational affiliation

Visiting Professor at the Department of Computer Science →

Address

ETH Zürich

Prof. Dr. Bernhard Thomaszewski

Computergestützte Robotik

STD → H 23 →

Stampfenbachstrasse 48

8092 Zürich

Switzerland

+41 44 632 89 62 →

(Sec.) +41 44 632 74 01 →

bthomasz@ethz.ch →

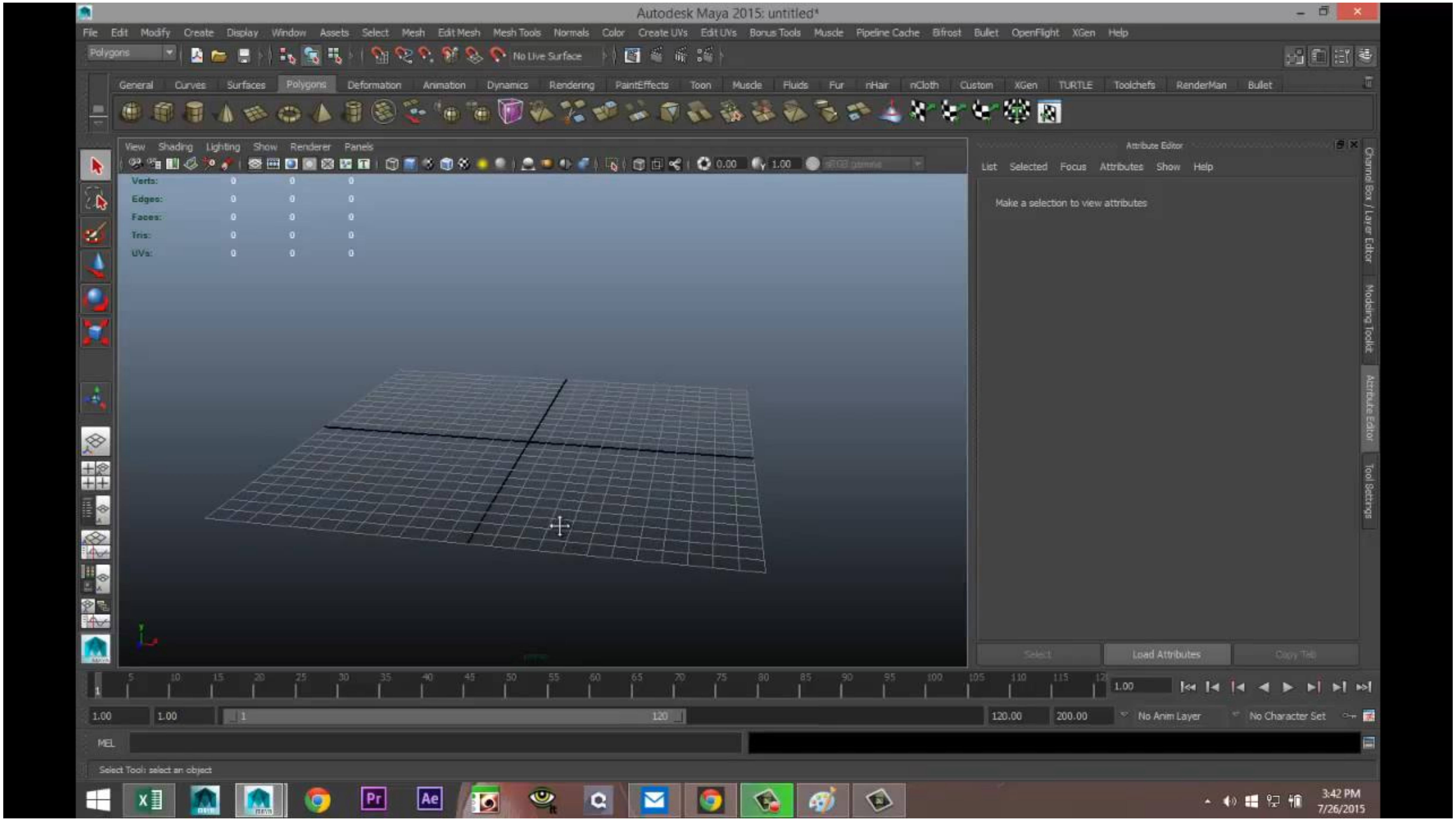
V-Card (vcf, 1kb) ↓

Print address page →

- bthomasz@inf.ethz.ch
- bernhard@iro.umontreal.ca

Blend Shape Animation

- Keyframes are blending weights $\alpha(t_i)$
- Spline used to interpolate weights over time



Blend Shape Sculpting (Modeling)

- Traditionally a manual process

https://www.youtube.com/watch?v=8Nu3znisiBg#t=01m26s

Blend Shapes

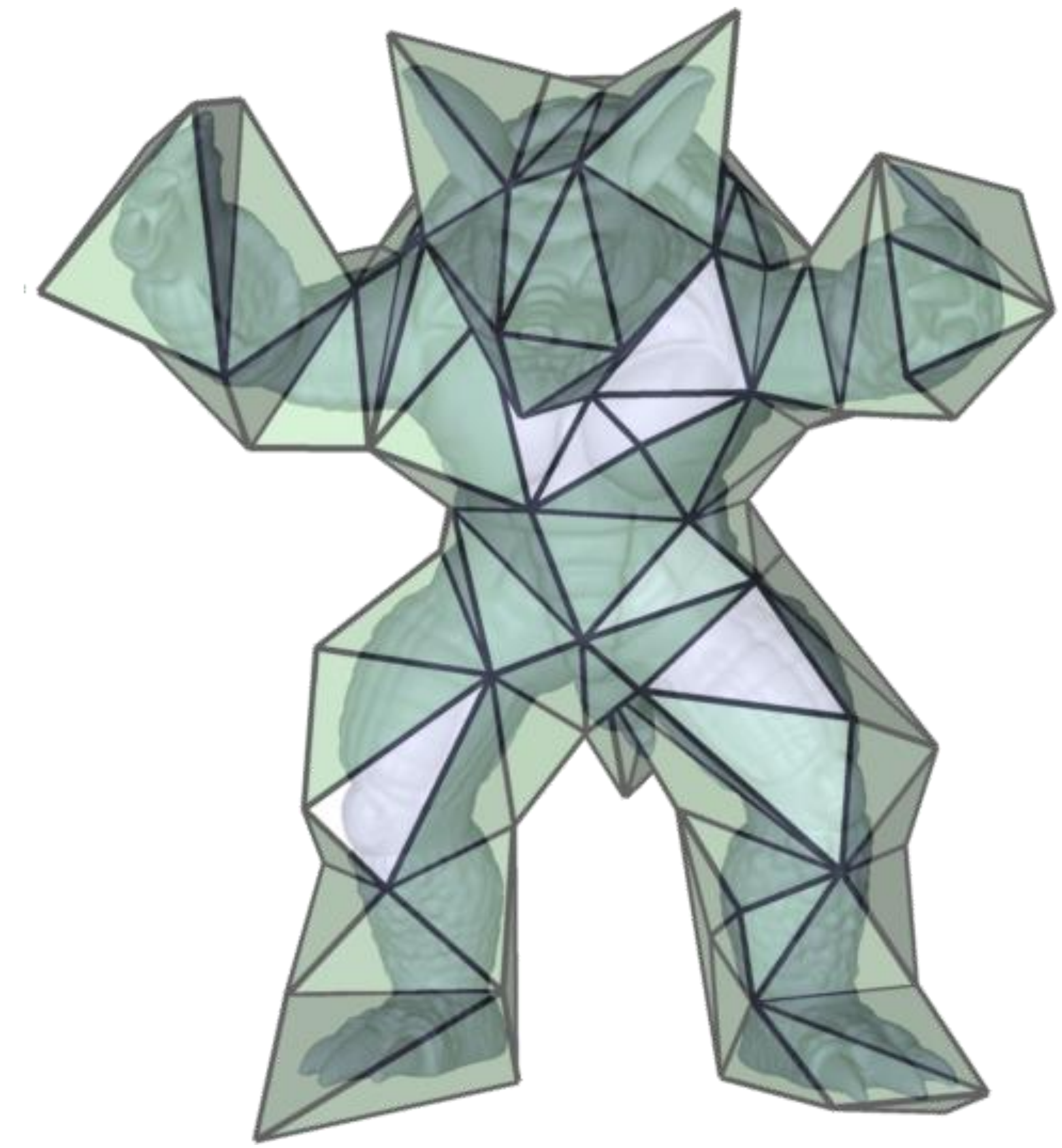
are very simple, but in the hands of a skilled animator…

courtesy Félix Ferrand

- Q: What are the pros and cons of blend shapes? When does this approach start to break down?

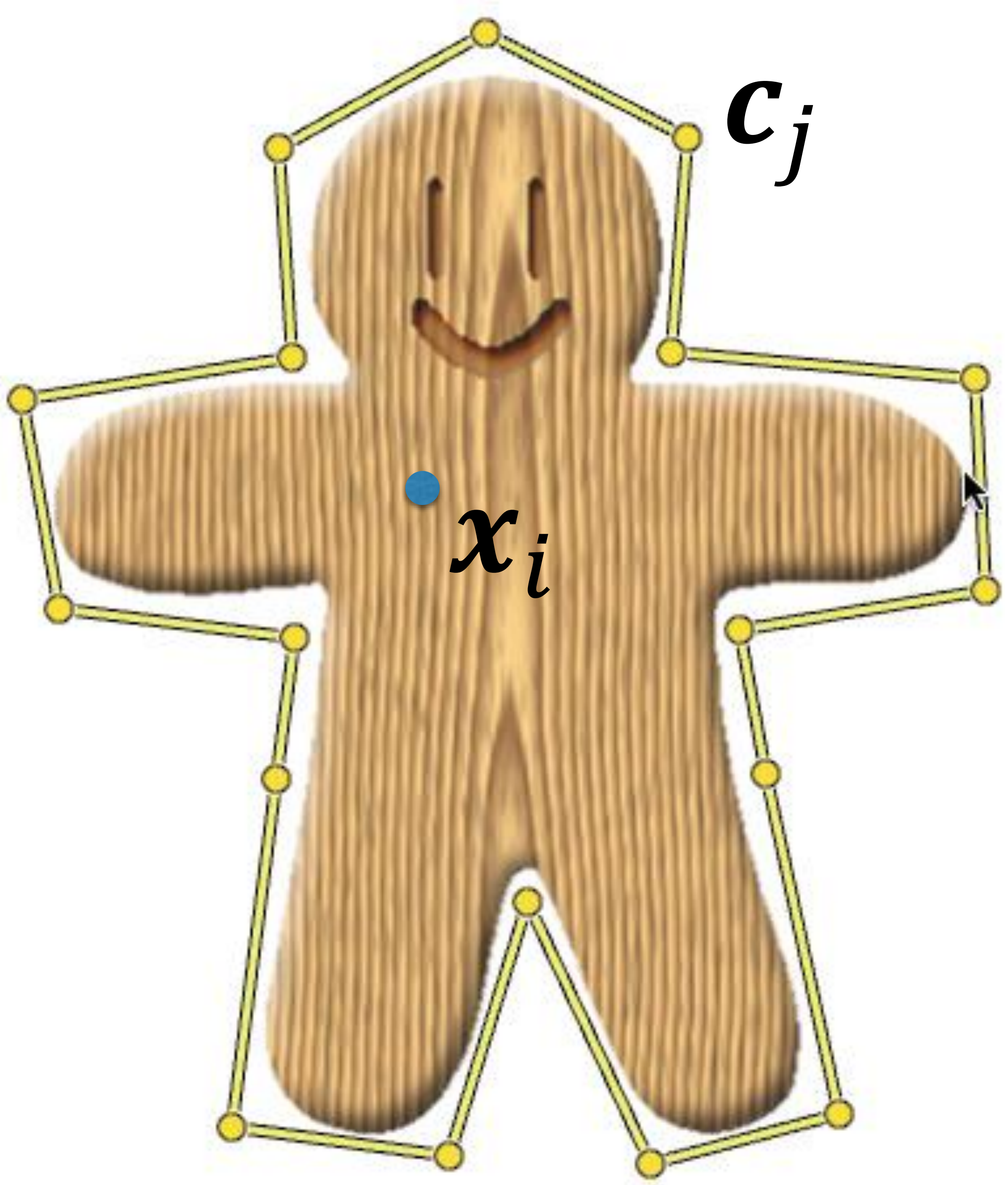
Cage-based Deformers

- Embed high-res model in a coarse mesh (*cage*)
- Deform coarse mesh, *apply* deformation to high-res model



Cage-based Deformers

- Embed high-res model in a coarse mesh (*cage*)
- Deform coarse mesh, *apply* deformation to high-res model



$$\boldsymbol{x}_i = \sum_{j}^{m} w_j(\boldsymbol{x}_i)\boldsymbol{c}_j$$

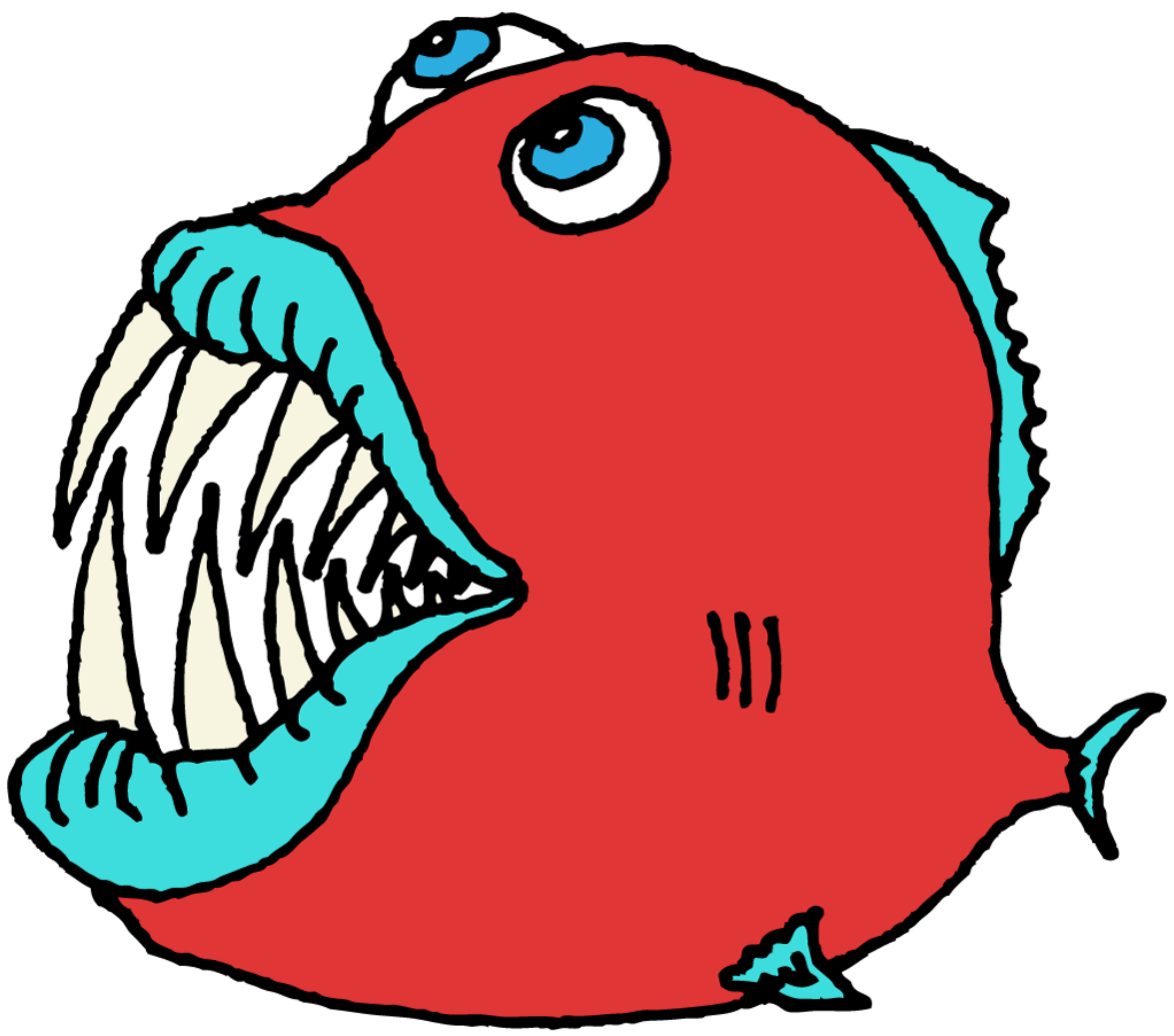
Q: where do the weights come from?

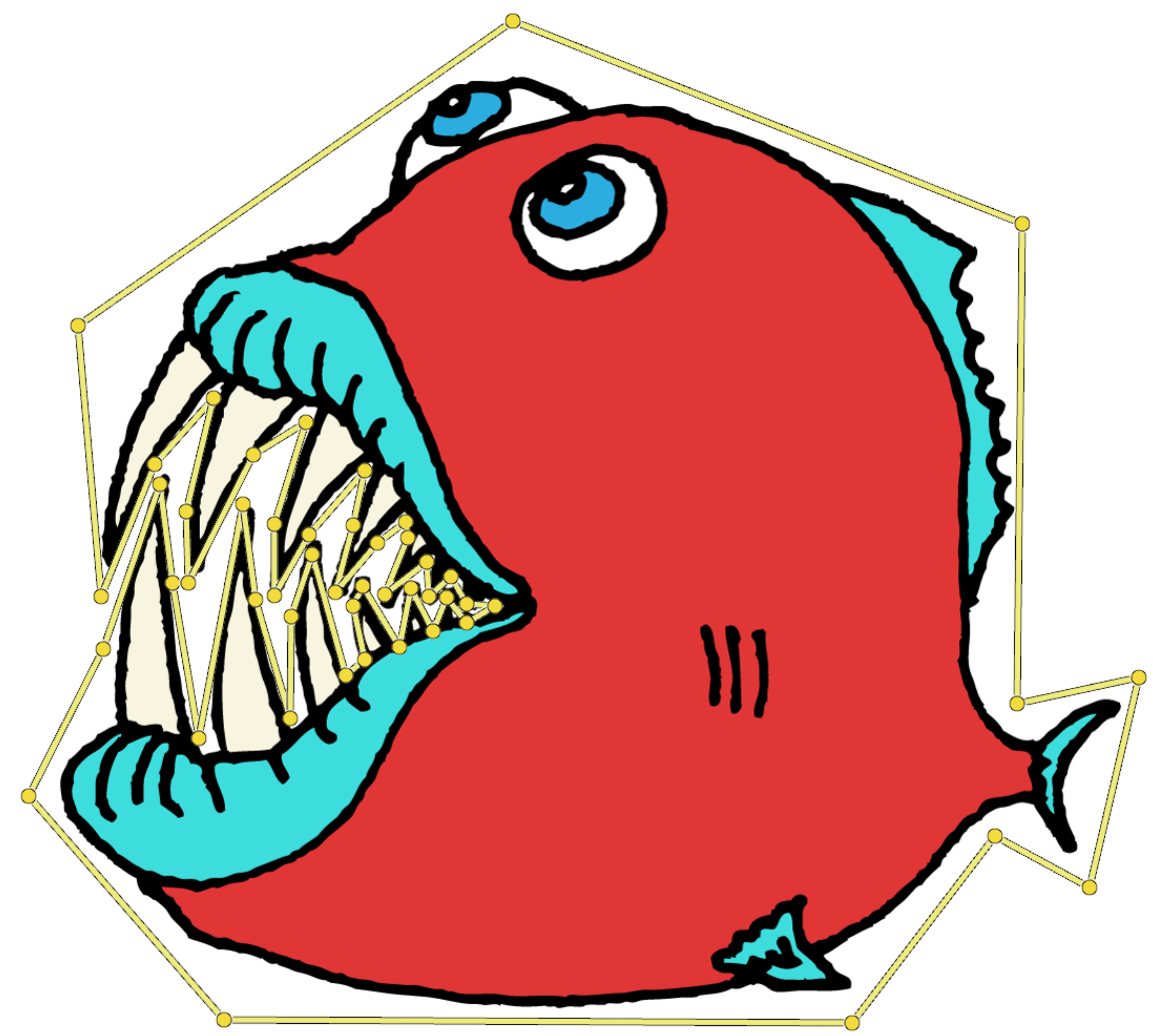
Cage-based Deformers – Zoo

- There are many ways of computing interpolation weights

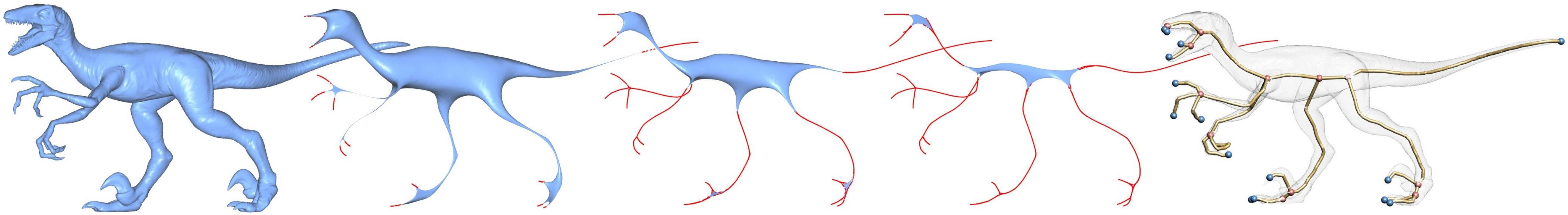
Method	N-GON	CONCAVE	SHAPE	≥ 0	C^1	LAG.	CLOSED	MONO.	OUT	LOCAL	POLY	COORD.
Barycentric	X	X	●	●	●	●	●	●	●	X	X	●
Wachspress	●	X	●	X	●	●	●	X	?	X	X	●
Natural Neighbor	●	●	X	●	X	●	●	●	●	●	●	●
Mean Value	●	●	●	X	●	●	●	X	●	X	X	●
Green and others	●	●	●	X	●	X	●	X	X	X	X	●
Positive Mean Value	●	●	●	●	X	●	X	X	X	X	X	●
Harmonic	●	●	●	●	●	●	X	●	X	X	●	●
Maximum Entropy	●	●	●	●	●	●	X	?	X	X	●	●
Const. Biharmonic	●	●	●	●	●	●	X	●	X	●	●	X

Working with cages can be too restrictive, hard to have direct control



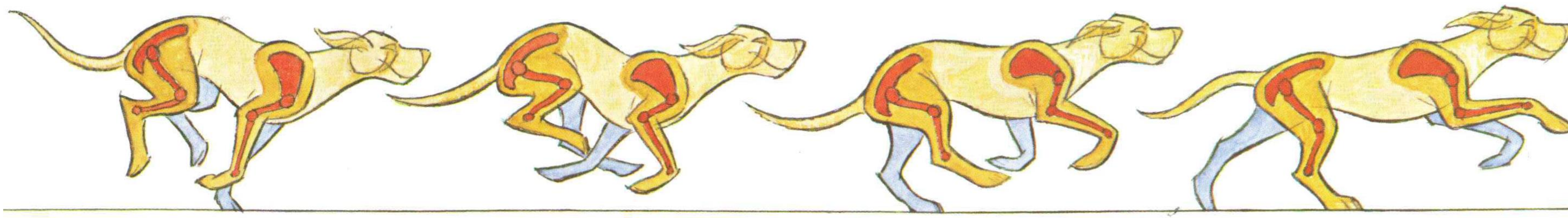


But very often, shape implies skeleton



[Au et al. 2008]

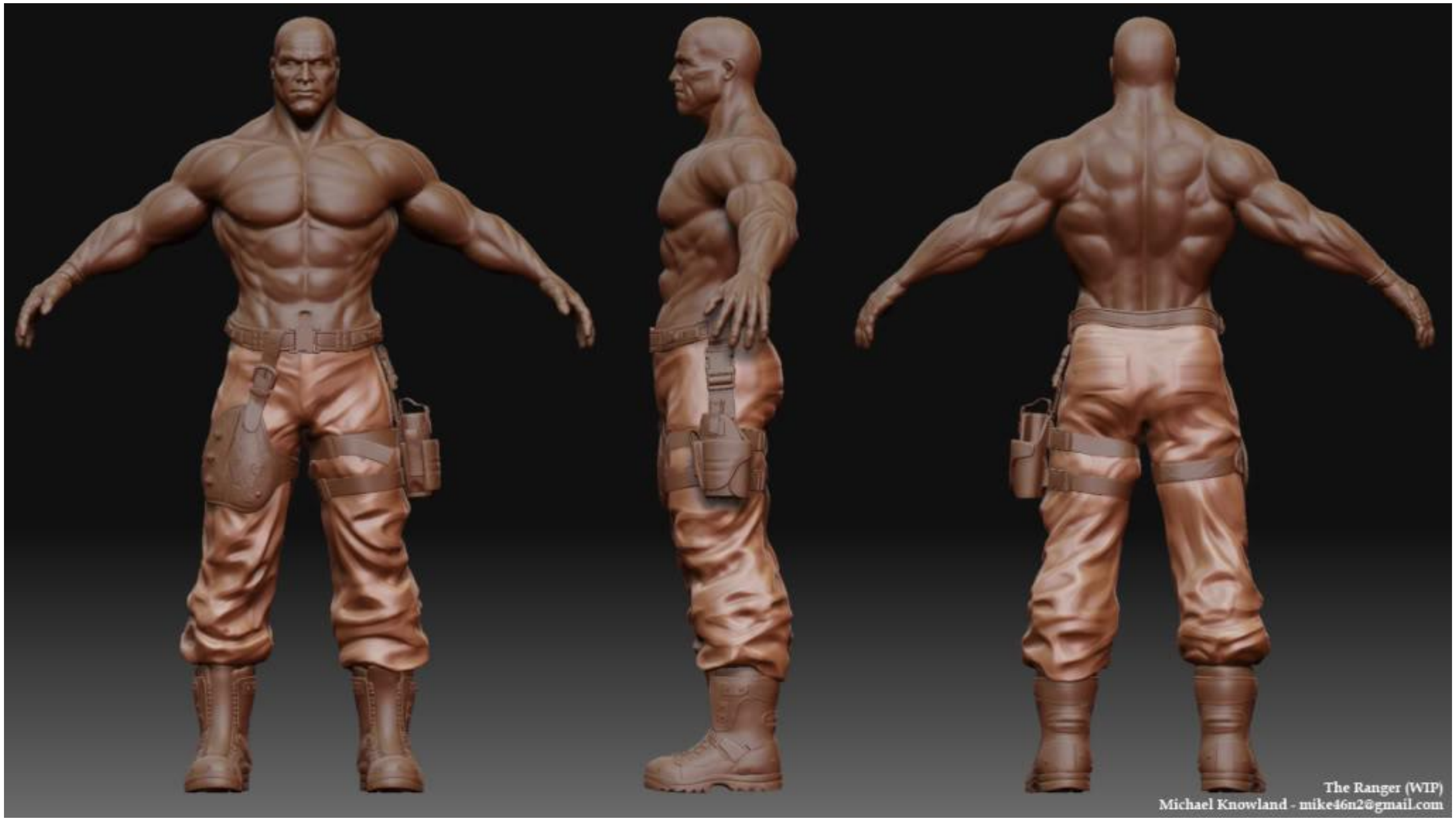
… and a skeleton imposes a lot of structure in how a character can move



[Blair 1994]

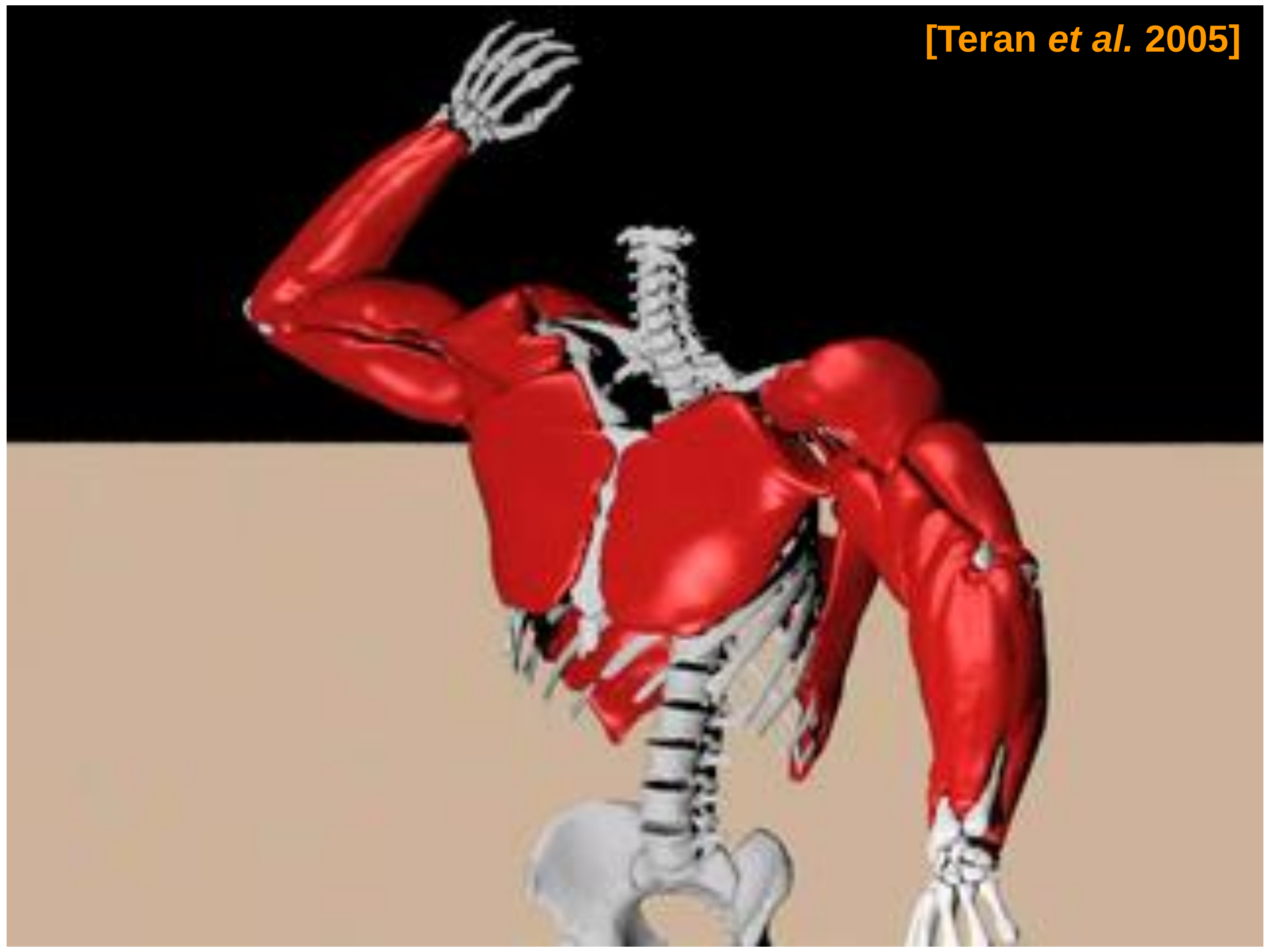
Skeletal Animation

Key idea: animate just the skeleton (<< DOFs), have mesh "follow" automatically



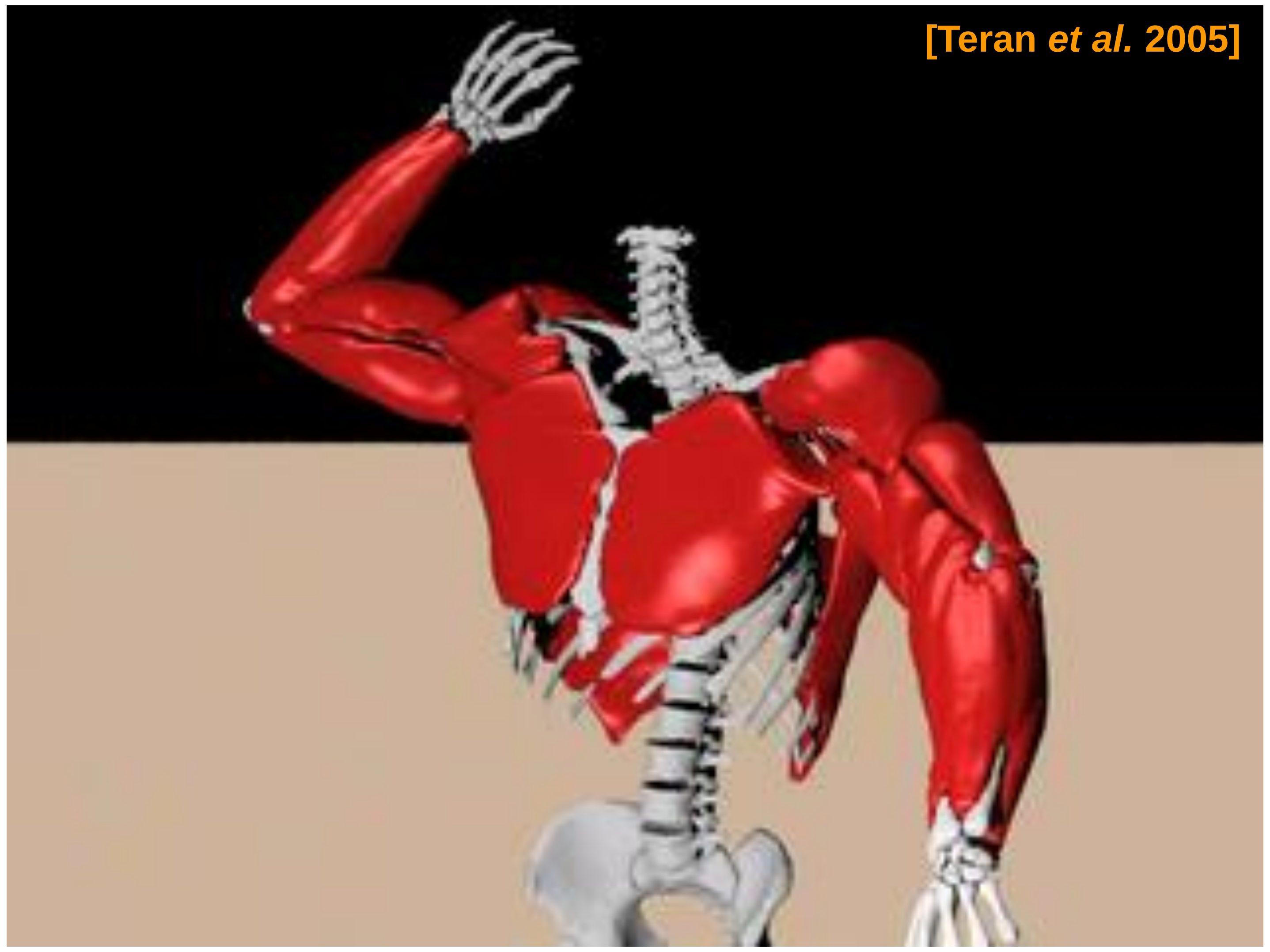
Skeletal Animation – at one end of the spectrum

- Animate just the skeleton
- Simulate muscles, fatty tissues, skin, interactions with the environment, etc….



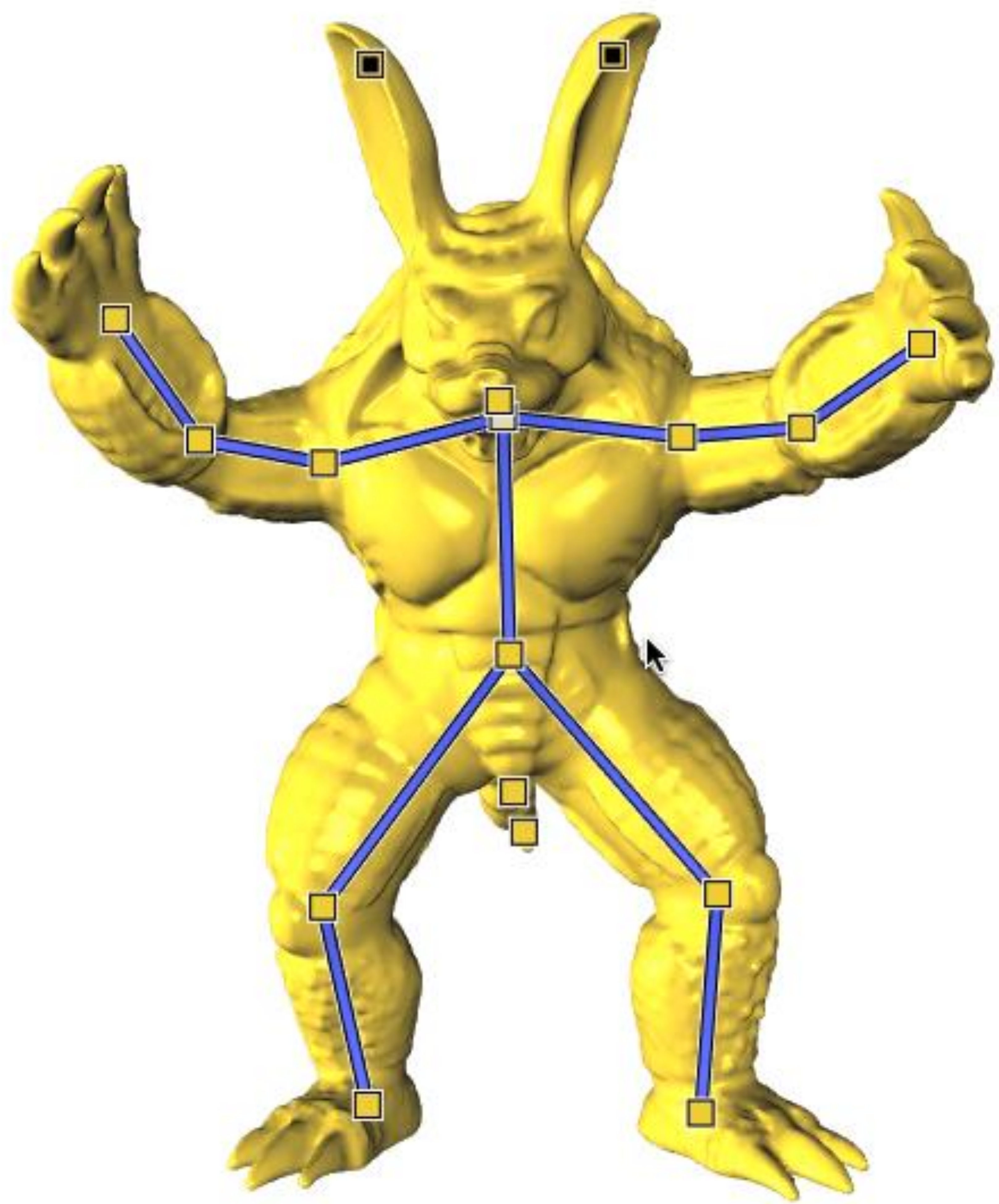
Skeletal Animation – at one end of the spectrum

- Automatically provides the *right* answer
- Very demanding computationally
- Need much faster solution for interactive applications

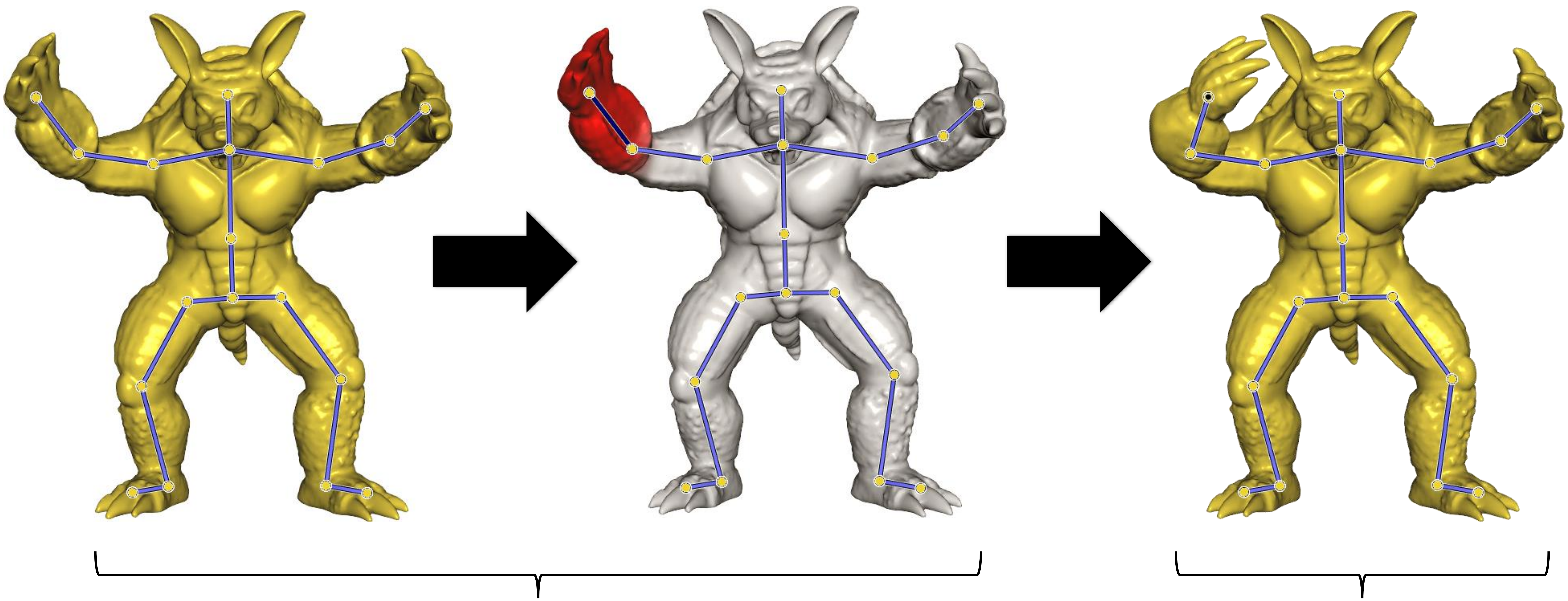


Linear Blend Skinning

- Couple deformation of surface mesh to skeletal pose



Linear Blend Skinning

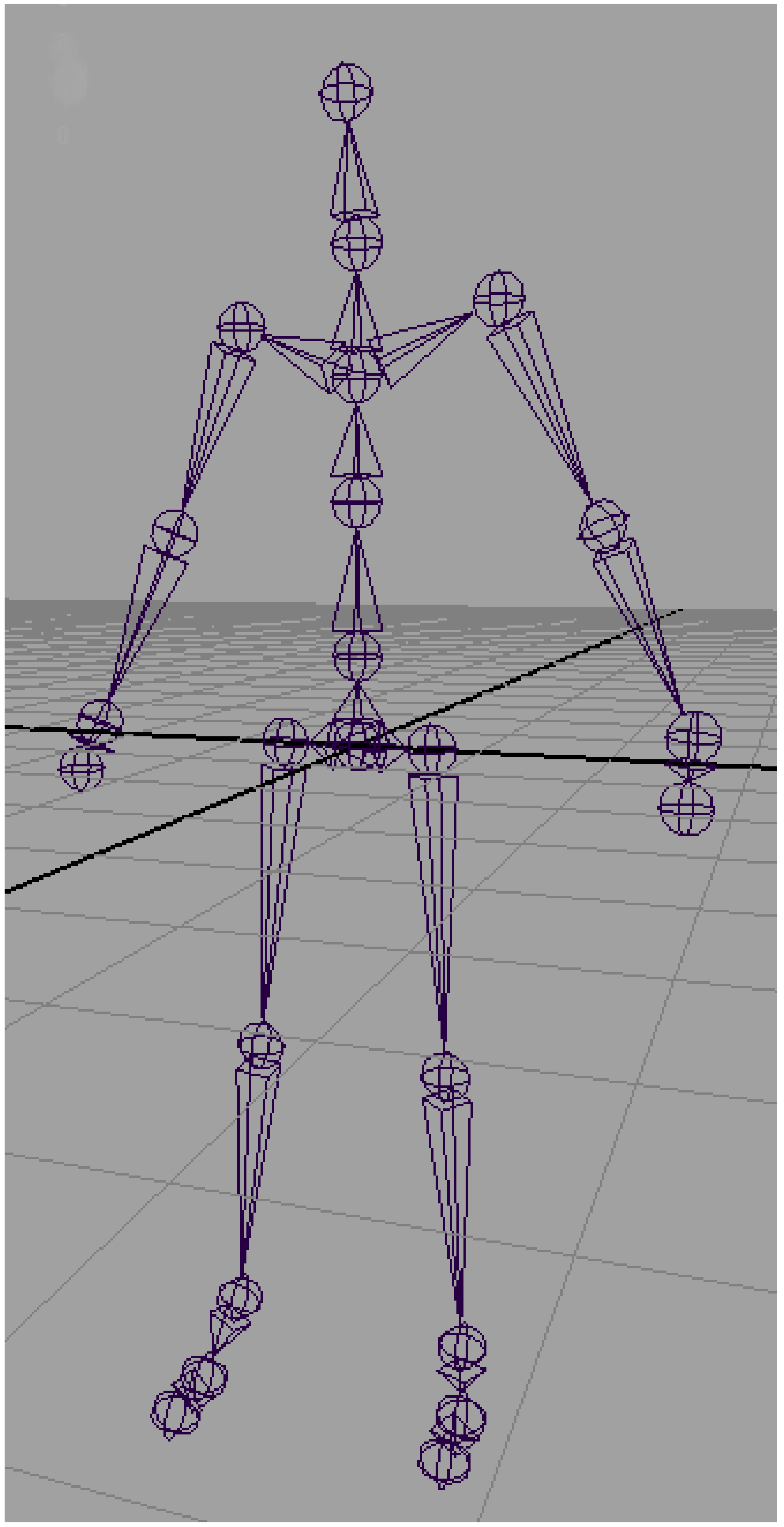


Rigging

Animation

Forward Kinematics

- Kinematic Skeletons
  - Hierarchy of affine transformations
  - *Joints*: local coordinate frames
  - *Bones*: vectors between consecutive pairs of joints
  - Each non-root bone defined in frame of unique parent
  - Changes to parent frame affect all descendent bones
  - Both skeleton and skin are designed in a rest (*bind*) pose



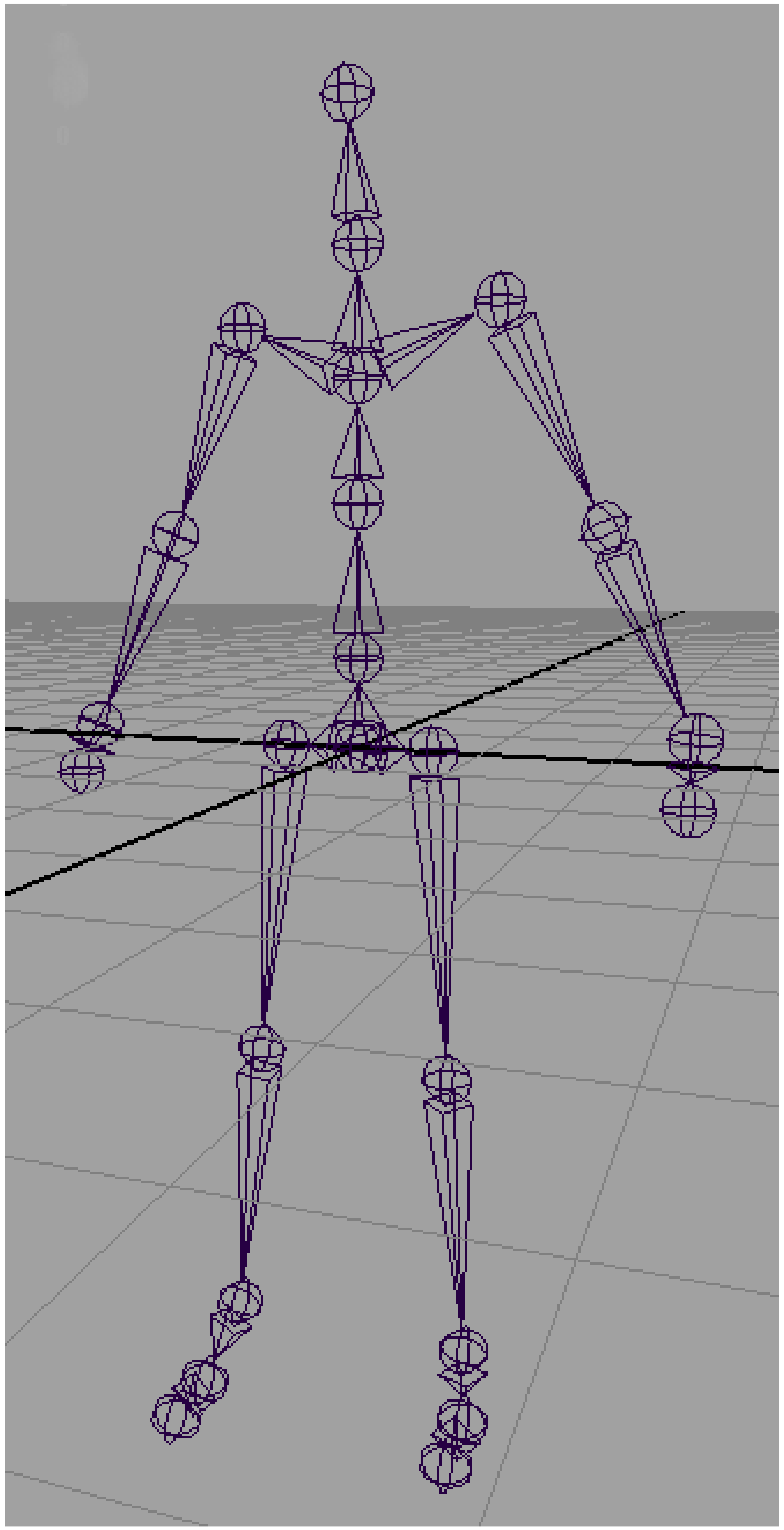
Forward Kinematics

Assume n+1 joints: 0, 1, ..., n (0 == root)

- each joint corresponds to a frame
- p(j): the parent of joint j (root special: p(0) = -1)
- frame of joint j expressed w.r.t. to frame of p(j)

$$
{}_{p(j)}R_j = \begin{pmatrix} r_{11}(j) & r_{12}(j) & r_{13}(j) & t_1(j) \\ r_{21}(j) & r_{22}(j) & r_{23}(j) & t_2(j) \\ r_{31}(j) & r_{32}(j) & r_{33}(j) & t_3(j) \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
= \begin{pmatrix} Rot(j) & \mathbf{t}(j) \\ 0 & 1 \end{pmatrix}
$$



Forward Kinematics

Assume n+1 joints: 0, 1, ..., n (0 == root)

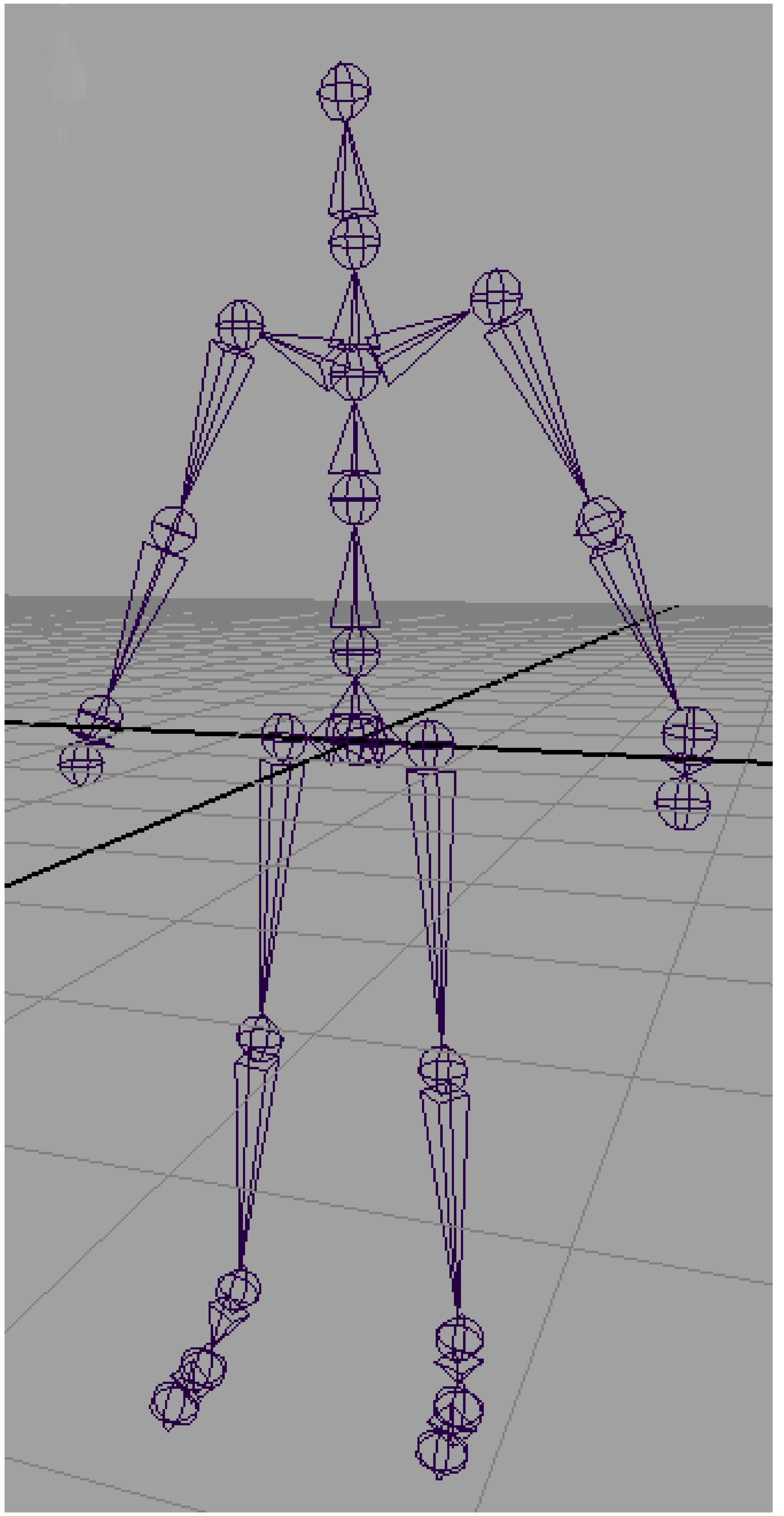
- each joint corresponds to a frame
- p(j): the parent of joint j (root special: p(0) = -1)
- frame of joint j expressed w.r.t. to frame of p(j)

What does this transform correspond to?

- **t**_(j)_ typically constant during animation

- _Rot(j)_ typically varies during animation

$$_{p(j)}R_j = \begin{pmatrix} Rot(j) & \mathbf{t}(j) \\ 0 & 1 \end{pmatrix}$$



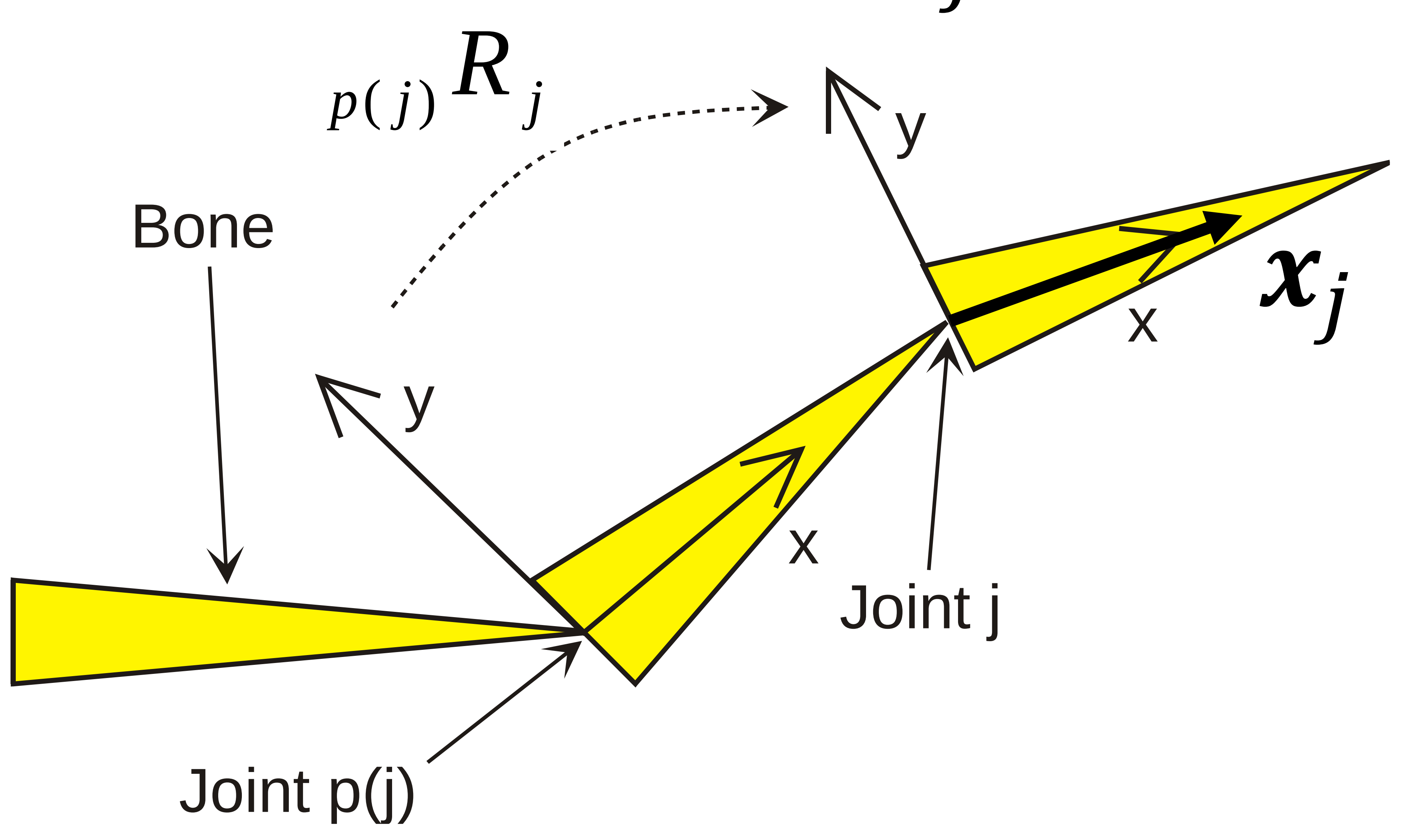
Forward Kinematics

- The transformation from frame *j* to world is:

$$
\begin{aligned}
{}_wR_j &= {}_wR_0 \ldots {}_{p(p(j))}R_{p(j)}\,{}_{p(j)}R_j \\
&= T(0)Rot(0)\ldots T(p(j))Rot(p(j))T(j)Rot(j)
\end{aligned}
$$

- Each joint rotation $Rot(j)$ has up to 3-DOFs
  - Translation and scale also possible, but less common
- Determining the world-space position of a point $\boldsymbol{x}_j$ in local coordinates on bone *j*

$$\boldsymbol{x}_w = {}_wR_0 \;\ldots\; {}_{p(j)}R_j \boldsymbol{x}_j$$

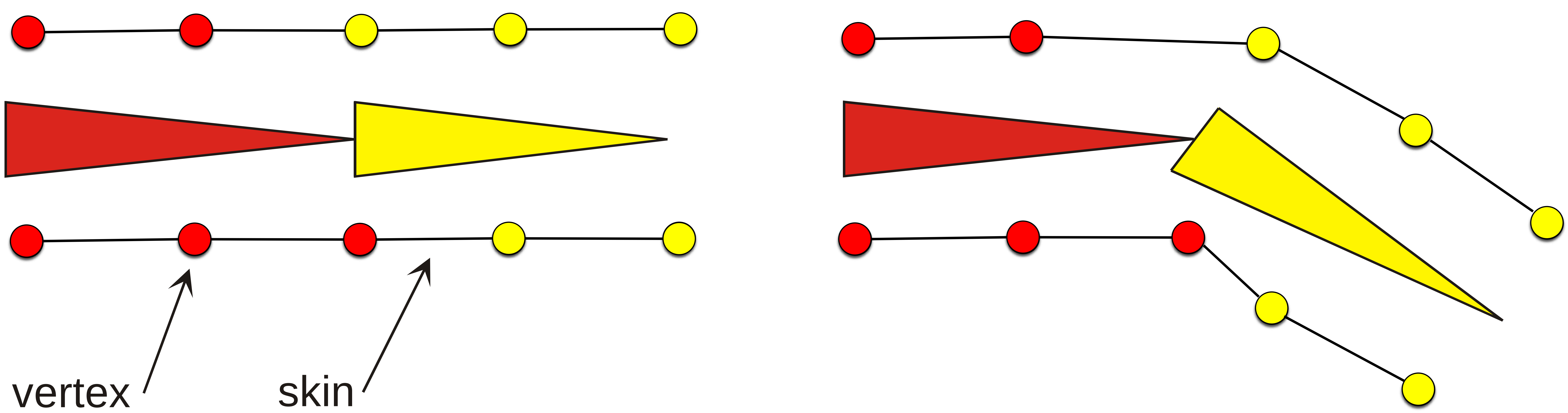


Skinning

Given joint angles, we compute configuration of the skeleton using forward kinematics. What do we do about the surface mesh?

- **Move vertices along with the bones!**
- **Attempt 1: assign each vertex to closest bone, compute world coordinates according to bone's transformation**

$$v = {}_{w}R_{j}\,{}_{w}\overline{R}_{j}^{-1}v'$$



Skinning

Given joint angles, we compute configuration of the skeleton using forward kinematics. What do we do about the surface mesh?

- **Move vertices along with the bones!**
- **Attempt 1: assign each vertex to closest bone, compute world coordinates according to bone's transformation**

$$\underbrace{v}_{\text{Skinned vertex coordinates}} = {}_w R_j \underbrace{{}_w\bar{R}_j^{-1} \underbrace{v'}_{\text{Vertex coordinates in rest pose}}}_{\text{Vertex coordinates in coordinate frame of bone j}}$$

- **Rigid skinning: fine if vertices are close to bone, fine for small rotations, used in older video games**
- **Can do better with only slight increase in complexity**

Linear Blend Skinning

Given joint angles, we compute configuration of the skeleton using forward kinematics. What do we do about the surface mesh?

- **Attempt 2: assign each vertex to multiple bones, compute world coordinates as *convex combination***
- **Weights: influence of each bone on the vertex**
- **Leads to smoother* deformations of the skin**

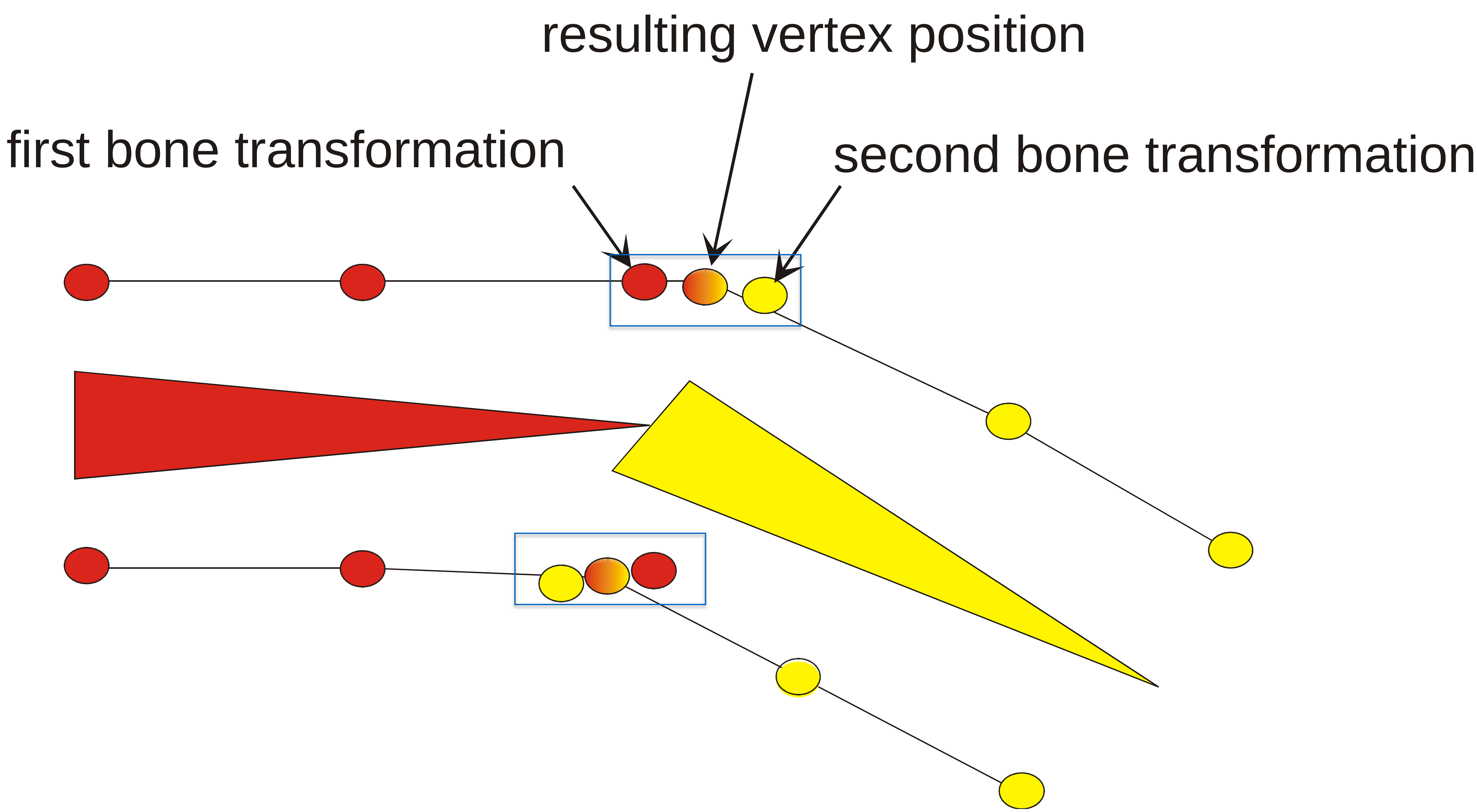
$$v = \sum_j \alpha_j \, {}_{w}R_j \, \underbrace{{}_{\bar{w}}\bar{R}_j^{-1} v'}_{\text{Vertex coordinates in coordinate frame of bone j}}$$

Skinned vertex coordinates

Skinning weights

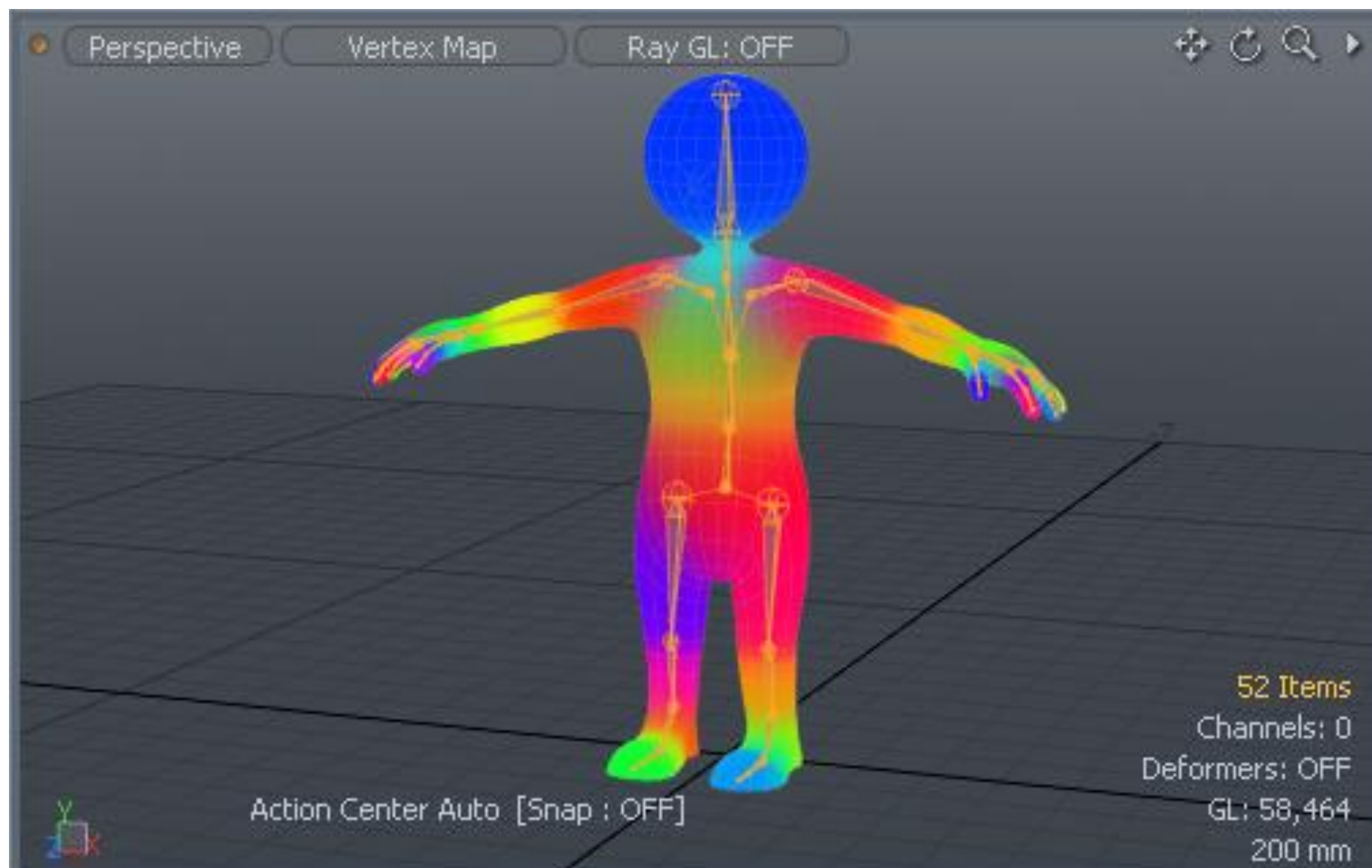
Vertex coordinates in rest pose

Linear Blend Skinning



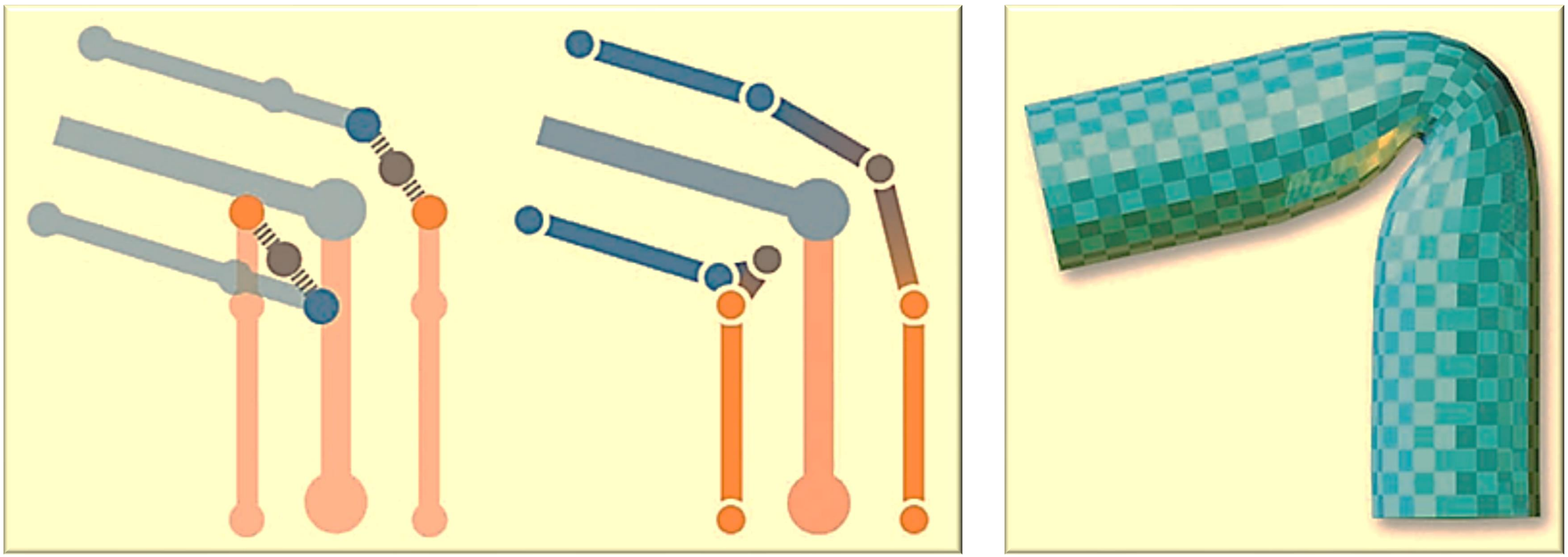
How are skinning weights generated?

- Often they are manually painted
- Can be computed automatically (active research area)
    - e.g. based on distance to bones - how would you do this? What might go wrong?



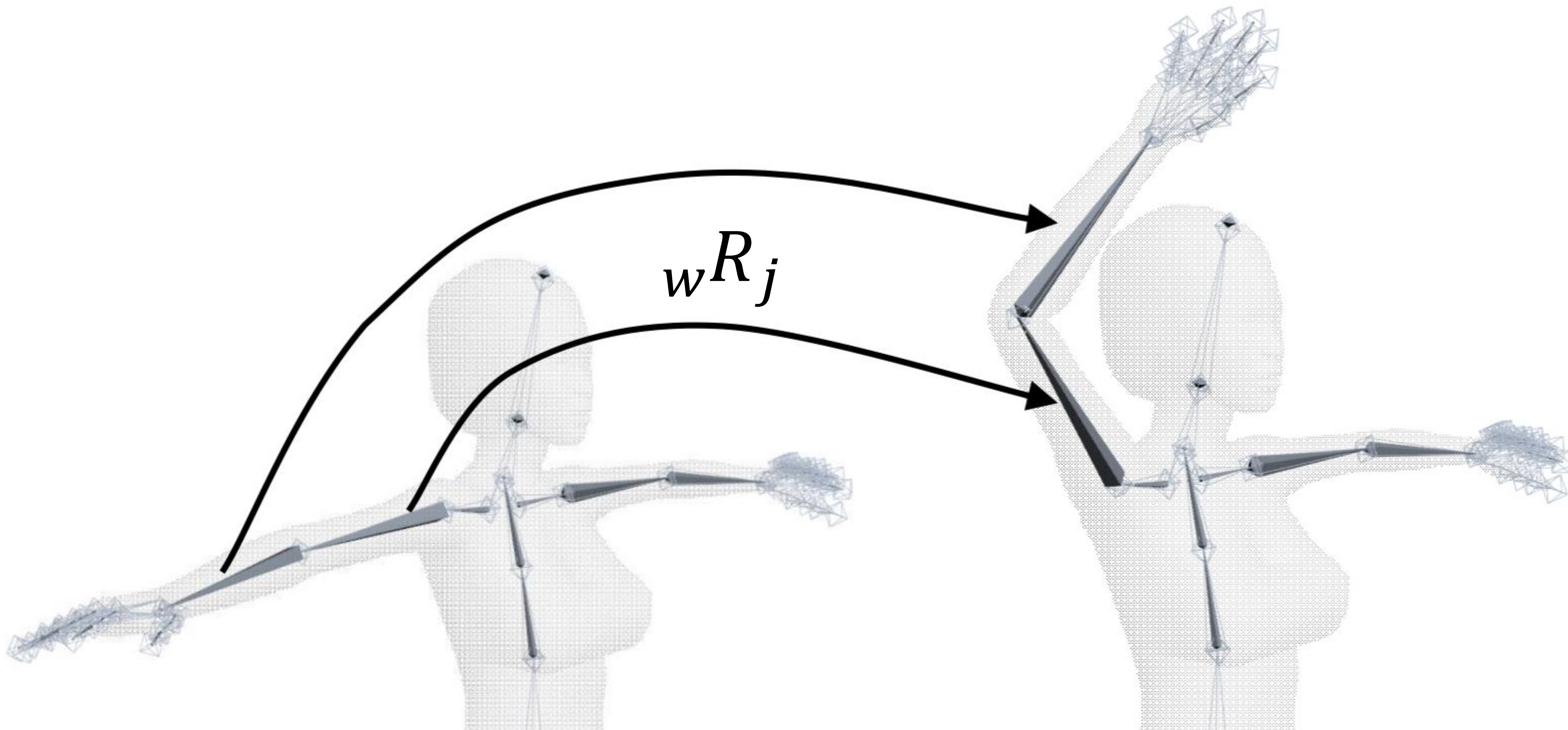
Linear Blend Skinning artifacts

- What's causing this?



- How to fix candy-wrapper effects: better weights, introduce auxiliary joints/bones, employ better interpolation schemes for transformations, pose-space deformers (PSD)

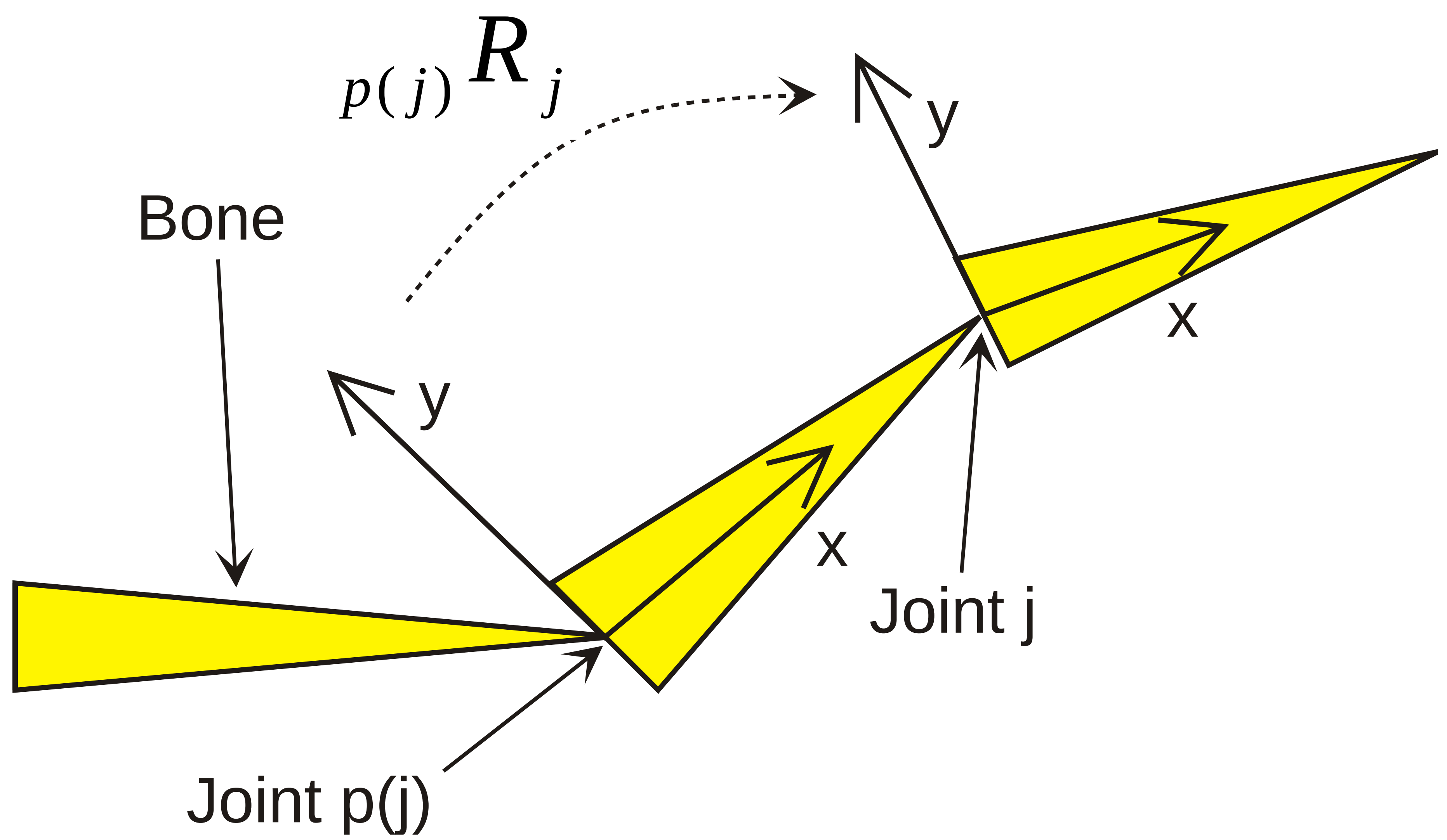
Linear Blend Skinning



Widely used in industry!

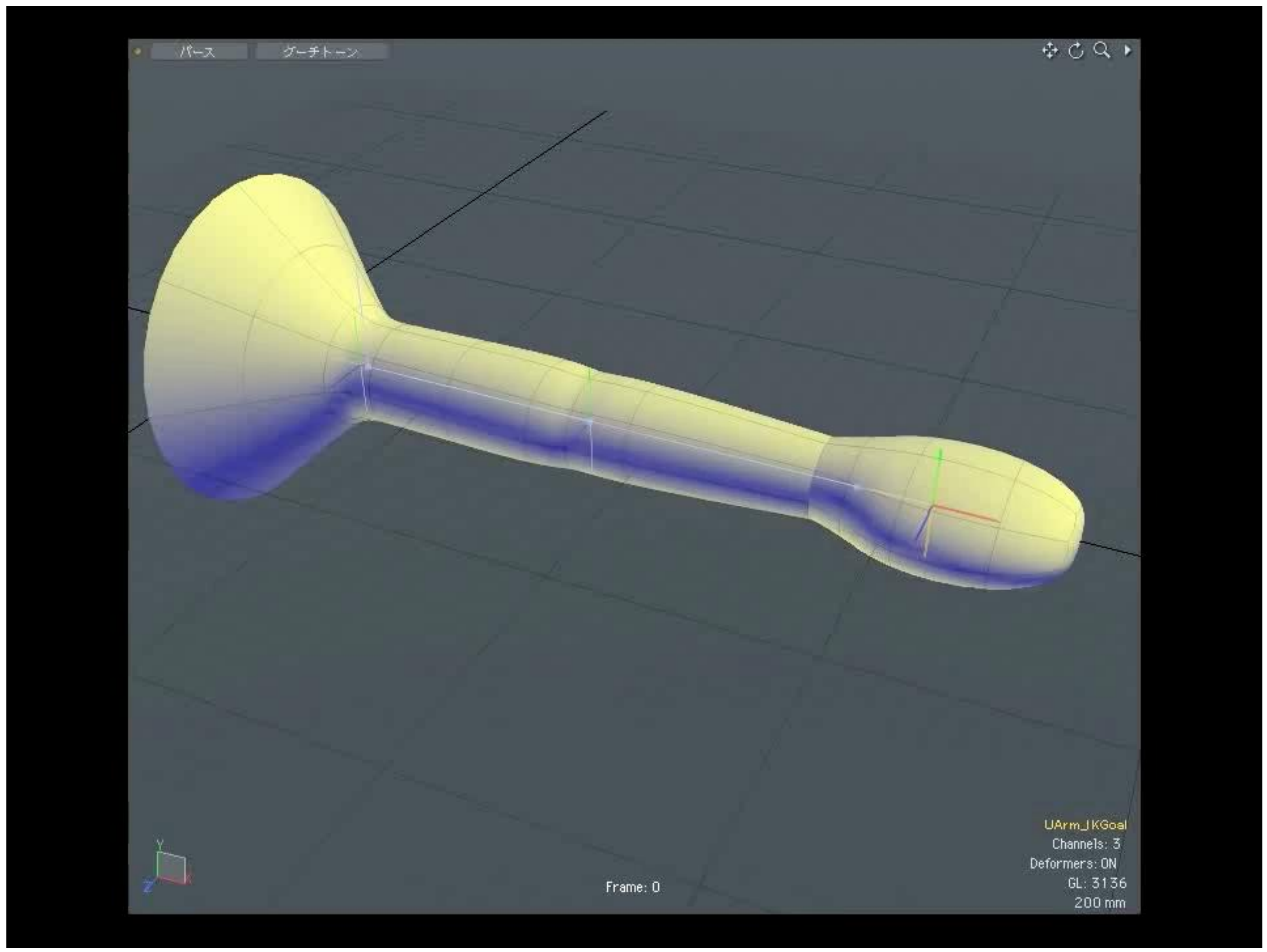
Forward Kinematics

- Given joint angles, compute configuration of the skeleton



Inverse Kinematics (IK)

- Given goal(s) for “end effector” compute joint angles
- Very important technique in animation and robotics!



- Many algorithms: analytic formulations for specific cases, energy-based methods, etc