Computer Graphics

Character Animation I

Mark Pauly

Geometric Computing Laboratory

Real-Time Characters

EA FIFA

Offline Characters

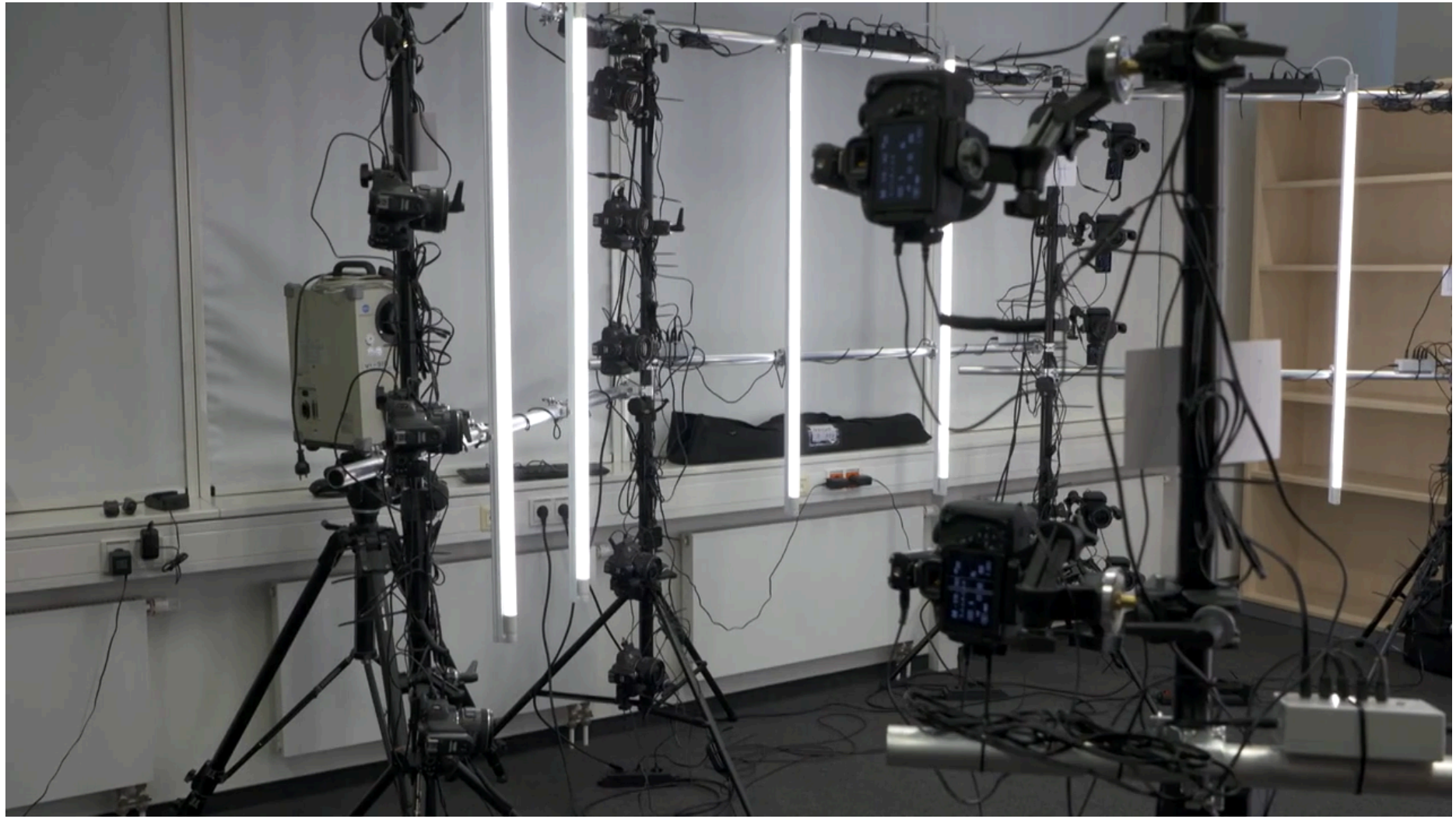
http://www.avatarmovie.com

Offline Motion Capturing

Lord of the Rings, Weta Digital

Real-Time Motion Capturing

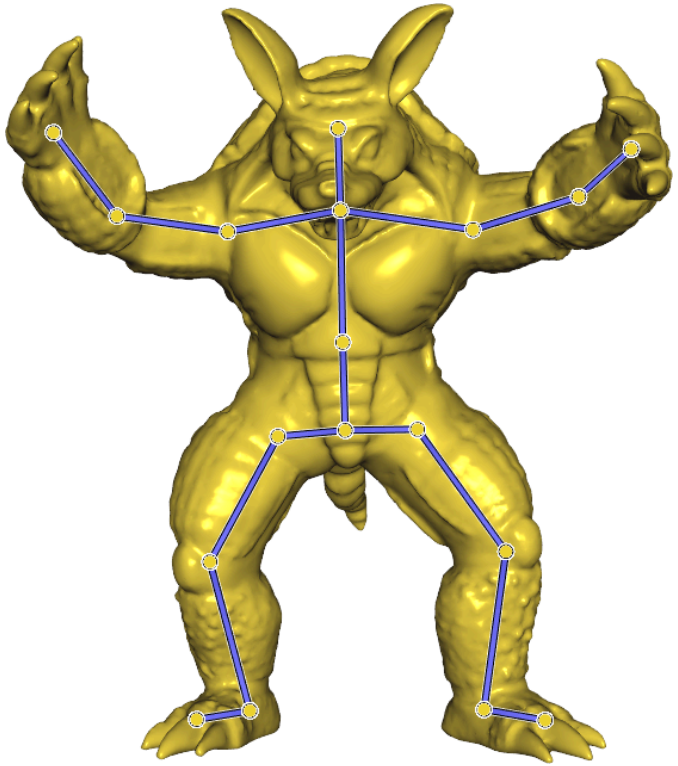
Character Scanning



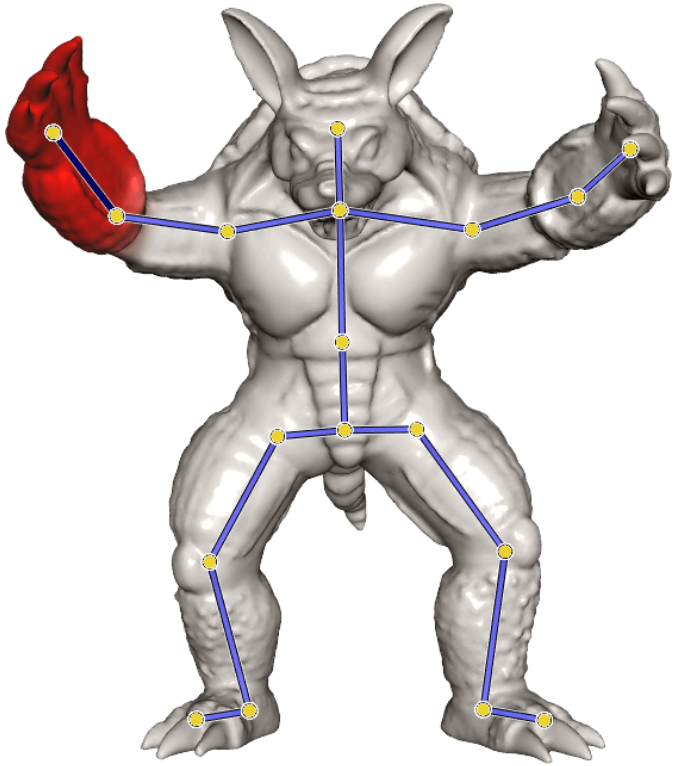
Offline Kicking

Skeleton-Based Animation

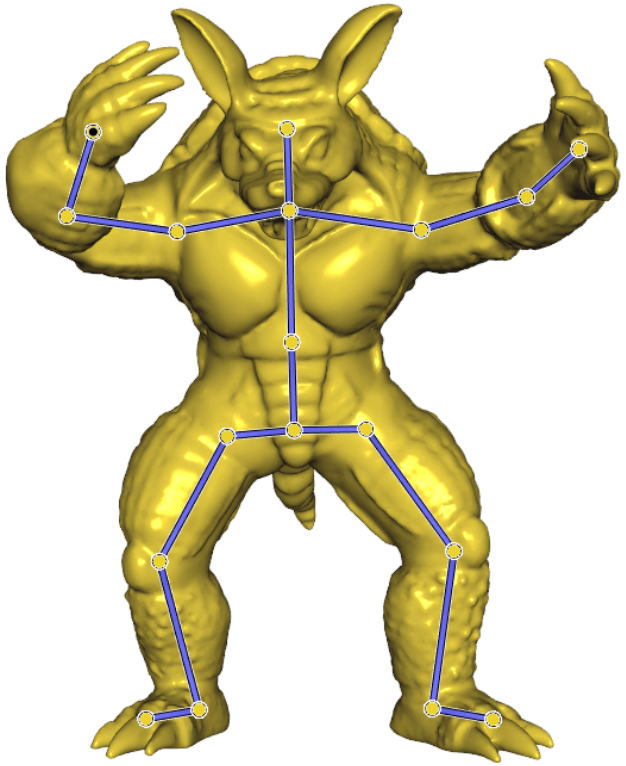
Skeleton-Based Animation



Embed skeleton



Define bone weights

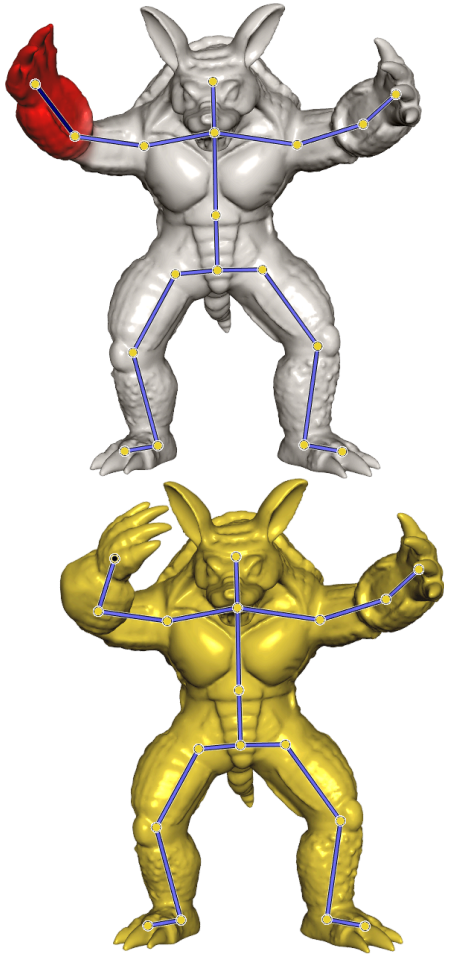


Pose skeleton, deform skin

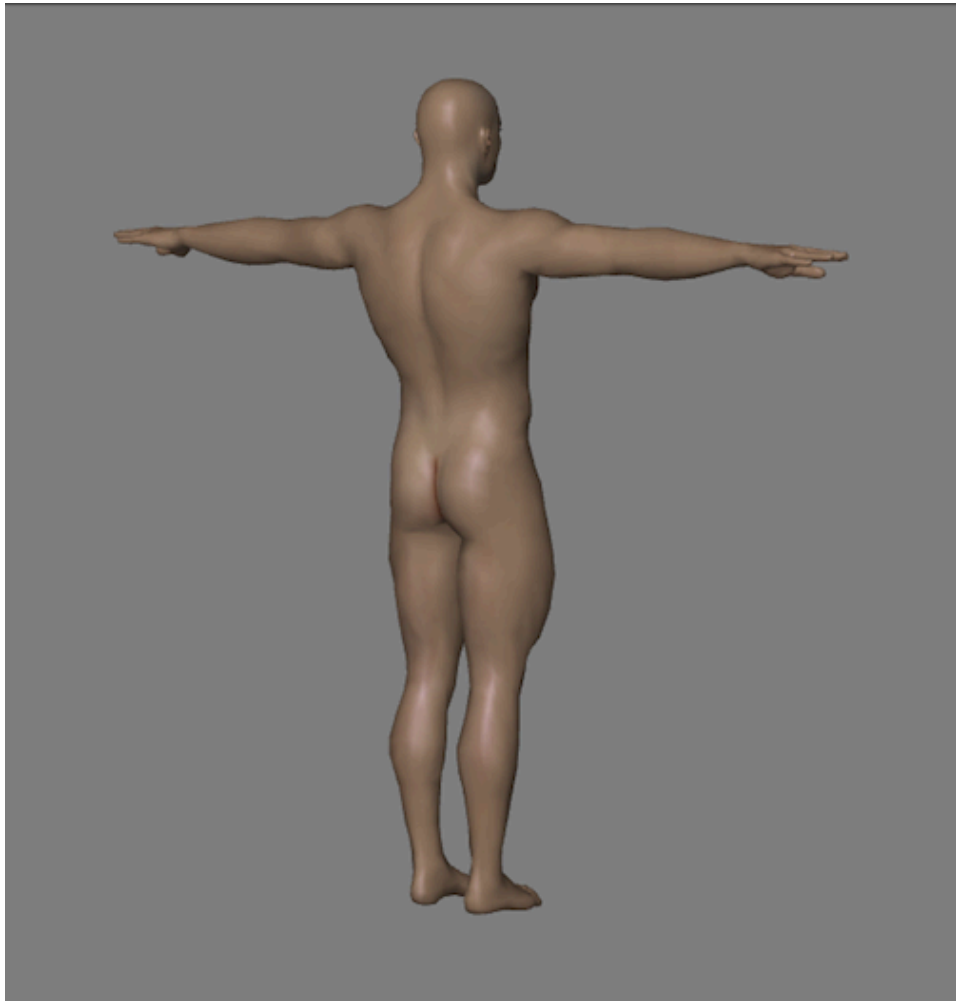
Images from Alec Jacobson

Skeleton-Based Animation

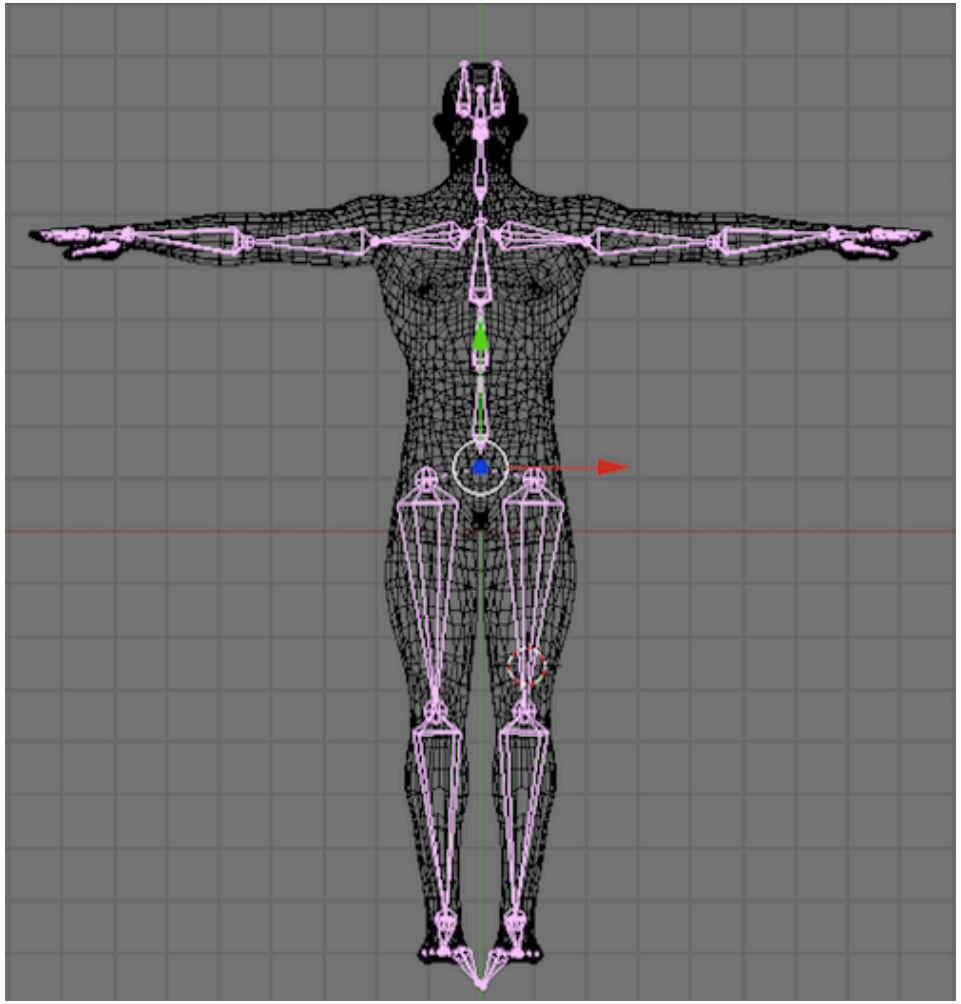
- **Rigging**
  - Embed skeleton
  - Define bone weights
- **Pose skeleton**
  - Forward Kinematics
  - Inverse Kinematics
  - Motion Capture
- **Skinning**
  - Deform skin based on deformed skeleton



Rigging

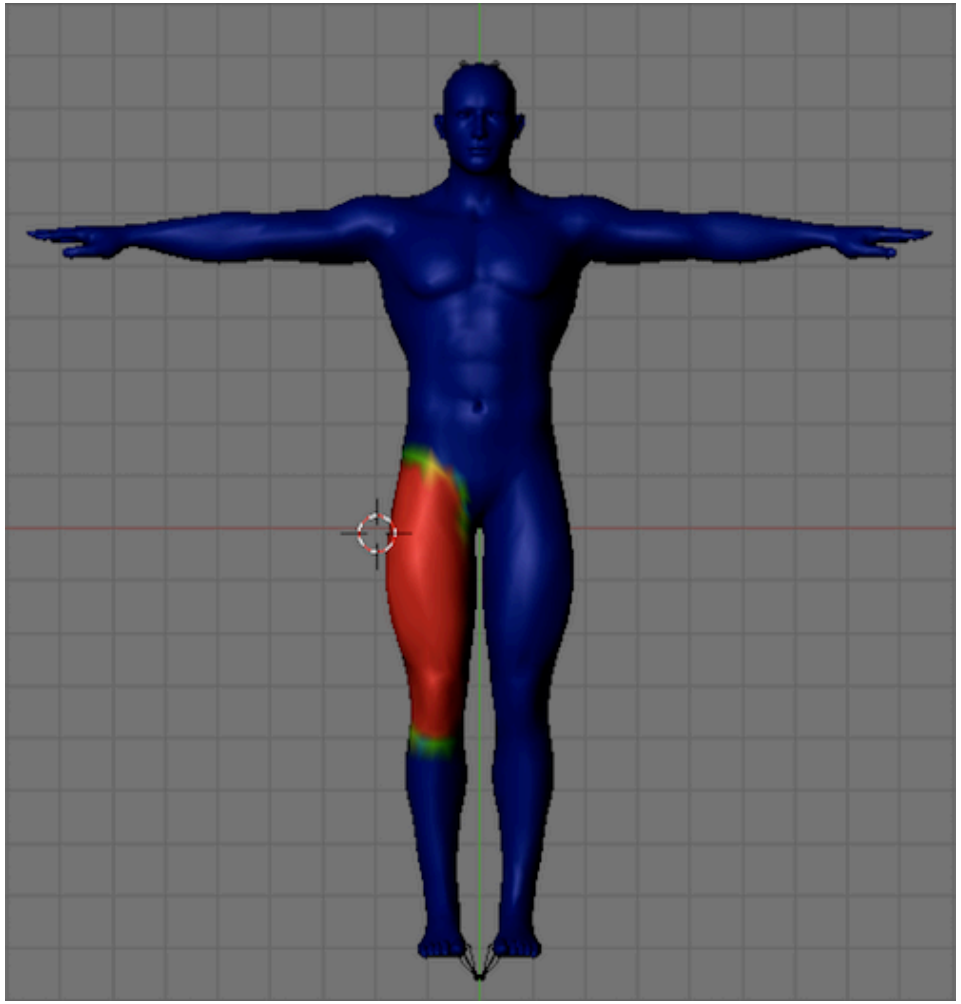


1. Design a character mesh

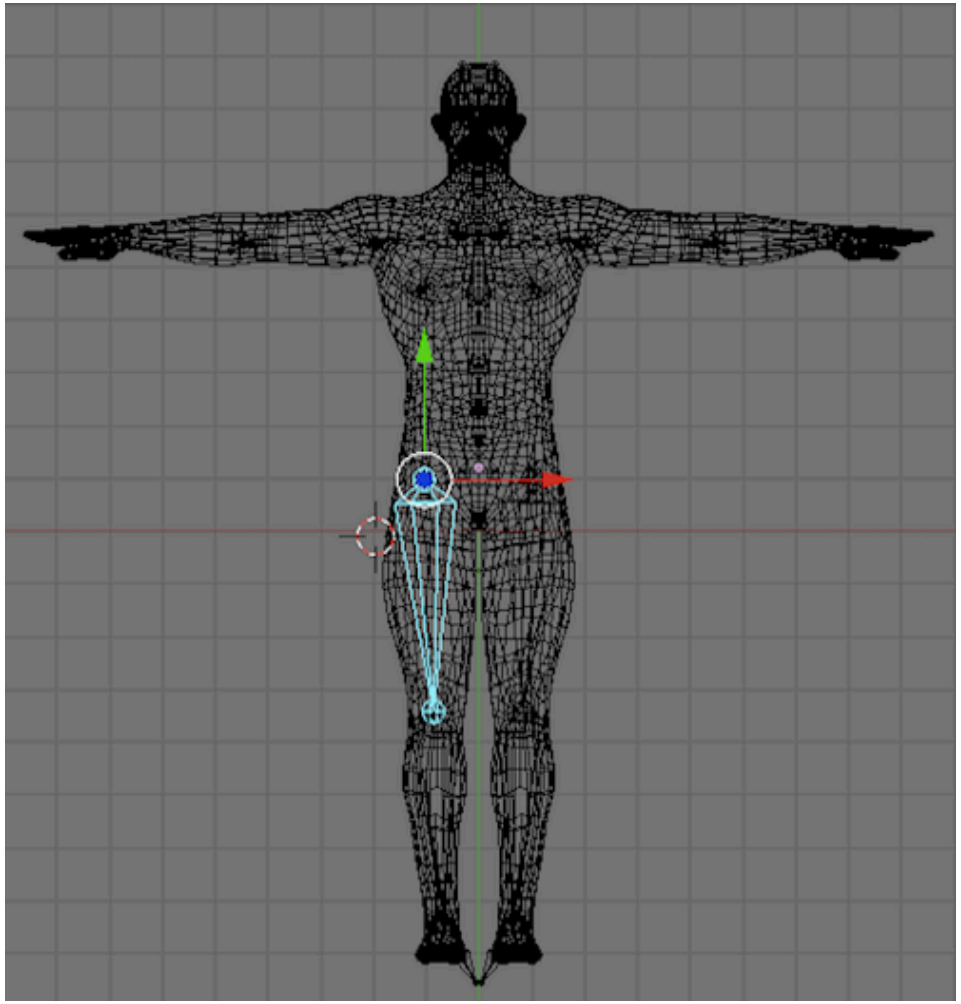


2. Embed a control skeleton

Rigging



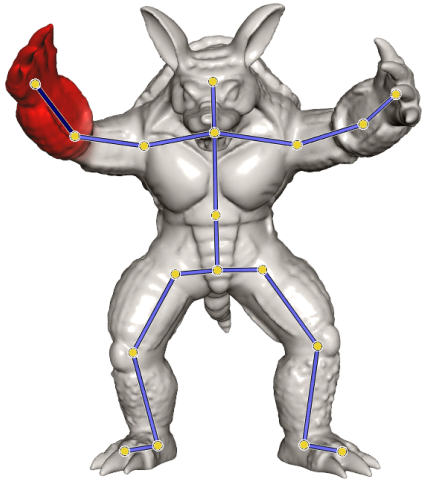
3. Assign mesh vertices…



…to skeleton bones

Rigging

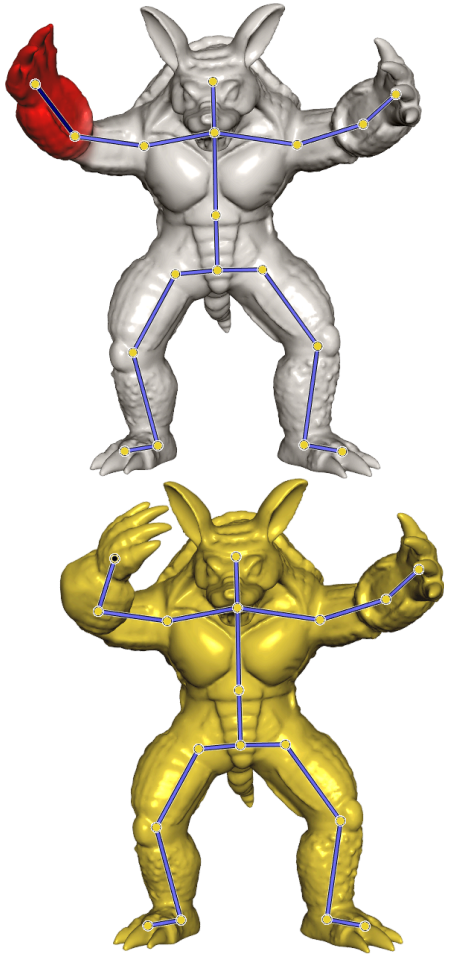
- Skinning weights (bone weights, joint weights)
  - weight $w_{i,j}$ defines the influence of bone j on vertex i
  - Weights must sum up to one: $\sum_j w_{i,j} = 1$
  - Weights should be non-negative: $w_{i,j} \geq 0$
  - Weights build *convex combination* per vertex
- How to compute weights?
  - Bone Heat method [Baran & Popovic 2007]
  - Bounded Biharmonic Weights [Jacobson et al 2011]
  - Or paint them manually…



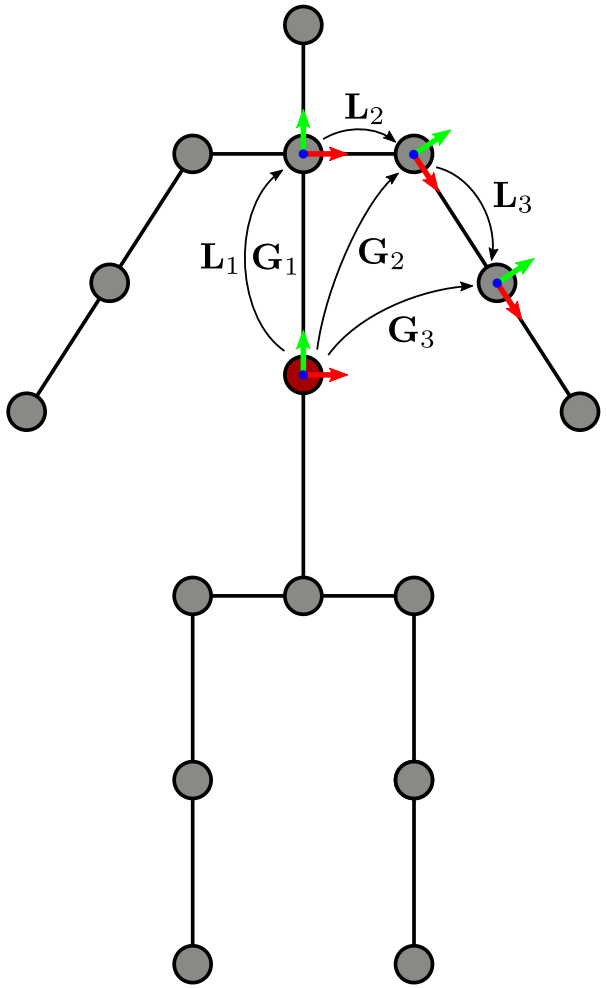
Let’s try

Skeleton-Based Animation

- **Rigging**
  - Embed skeleton
  - Define bone weights
- **Pose skeleton**
  - Forward Kinematics
  - Inverse Kinematics
  - Motion Capture
- **Skinning**
  - Deform skin based on deformed skeleton



Skeleton Forward Kinematics

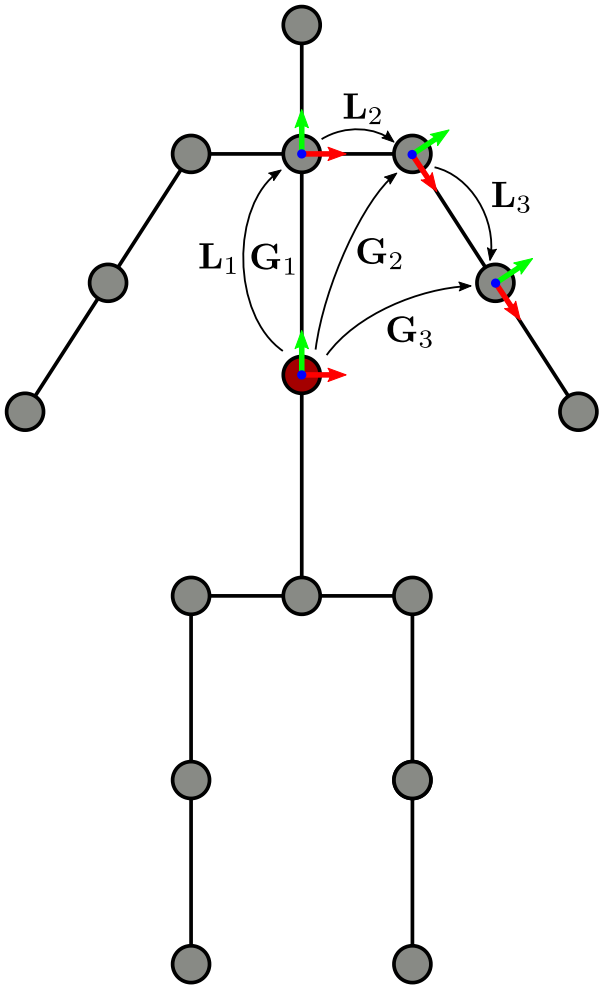


Skeleton hierarchy

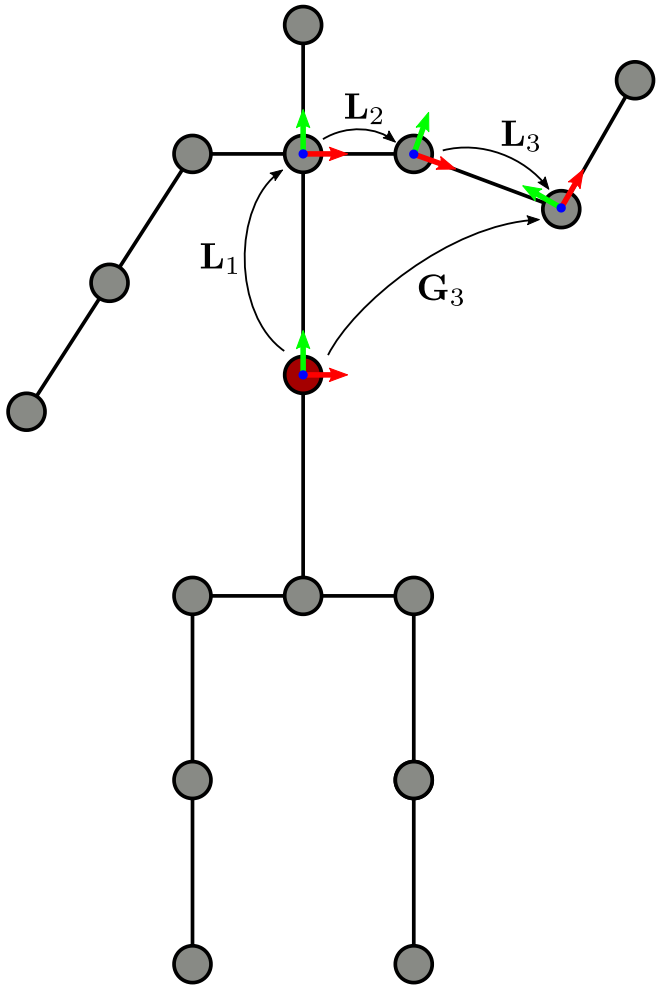
- Each joint j has a local coordinate system
- Local matrix $\mathbf{L}_j$ maps from j's coordinate system to parent's coordinates (rot+trans)
- Global matrix $\mathbf{G}_j$ maps from j's coordinate system to root coordinates (rot+trans)
- Compute global matrices through forward kinematics

$$\mathbf{G}_j = \mathbf{L}_1\mathbf{L}_2\cdots\mathbf{L}_j = \mathbf{G}_{j-1}\mathbf{L}_j$$

Skeleton Forward Kinematics



Initial bind pose

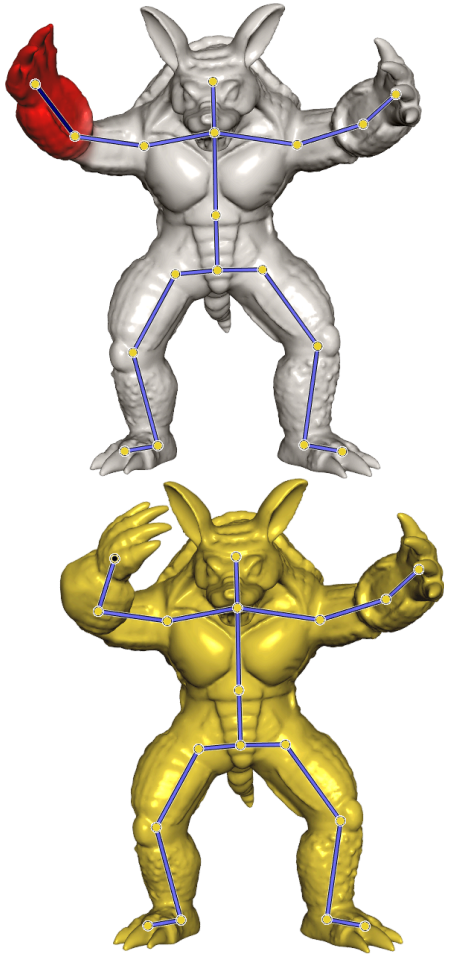


Animated pose

- Compute and store global matrix $\bar{\mathbf{G}}_j$ in initial pose
- Local matrices $\mathbf{L}_j$ change during animation and update global matrices $\mathbf{G}_j$
- Matrix $\mathbf{T}_j = \mathbf{G}_j \bar{\mathbf{G}}_j^{-1}$ maps from global initial coordinates through j's local coordinates to animated global coordinates
- Give $\mathbf{T}_j$ to skinning algorithm

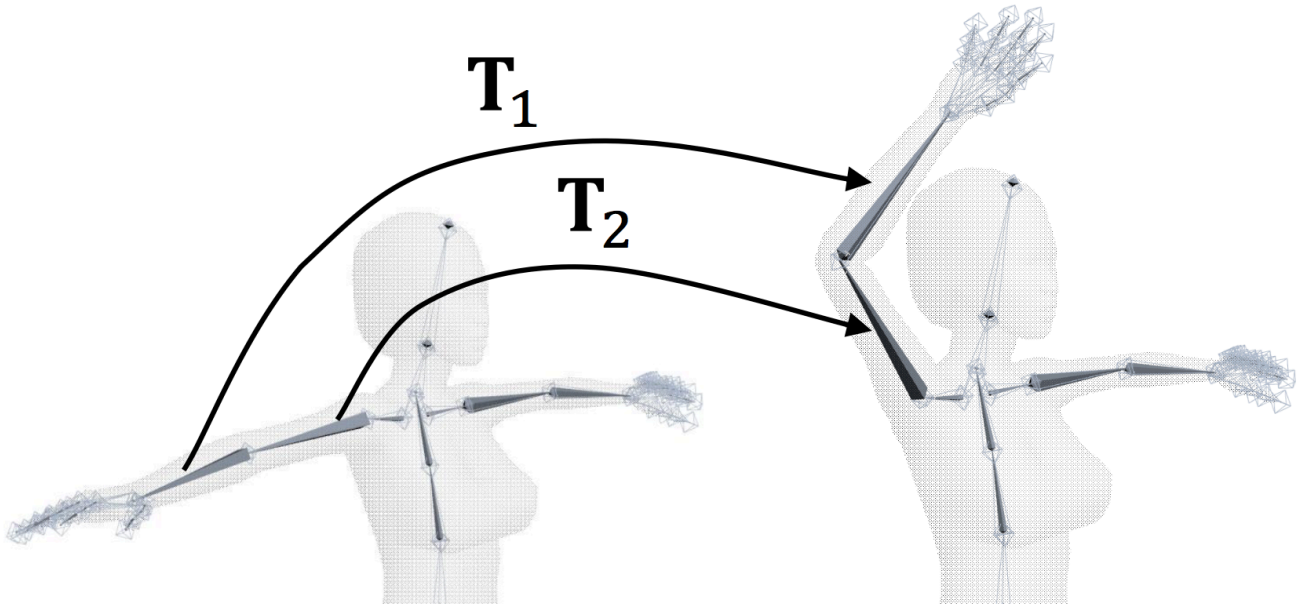
Skeleton-Based Animation

- **Rigging**
  - Embed skeleton
  - Define bone weights
- **Pose skeleton**
  - Forward Kinematics
  - Inverse Kinematics
  - Motion Capture
- **Skinning**
  - Deform skin based on deformed skeleton



Skinning

- Deform skin mesh to follow embedded skeleton
- Each bone j has an associated transformation matrix $\mathbf{T}_j$, which consist of rotation and translation (*rigid motion*)

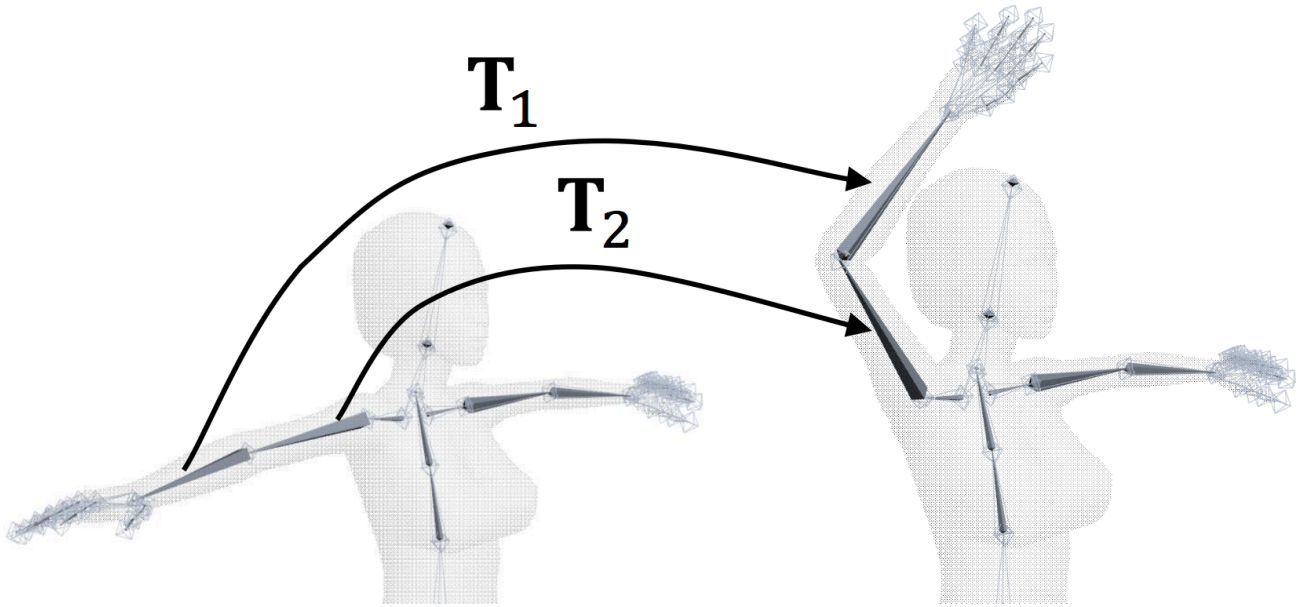


Images from Ladislav Kavan

“Rigid Skinning”

- Transform a vertex $\mathbf{x}_i$ that depends on **only one** bone j

$$\begin{pmatrix} \mathbf{x}_i' \\ 1 \end{pmatrix} = \mathbf{T}_j \begin{pmatrix} \mathbf{x}_i \\ 1 \end{pmatrix}$$

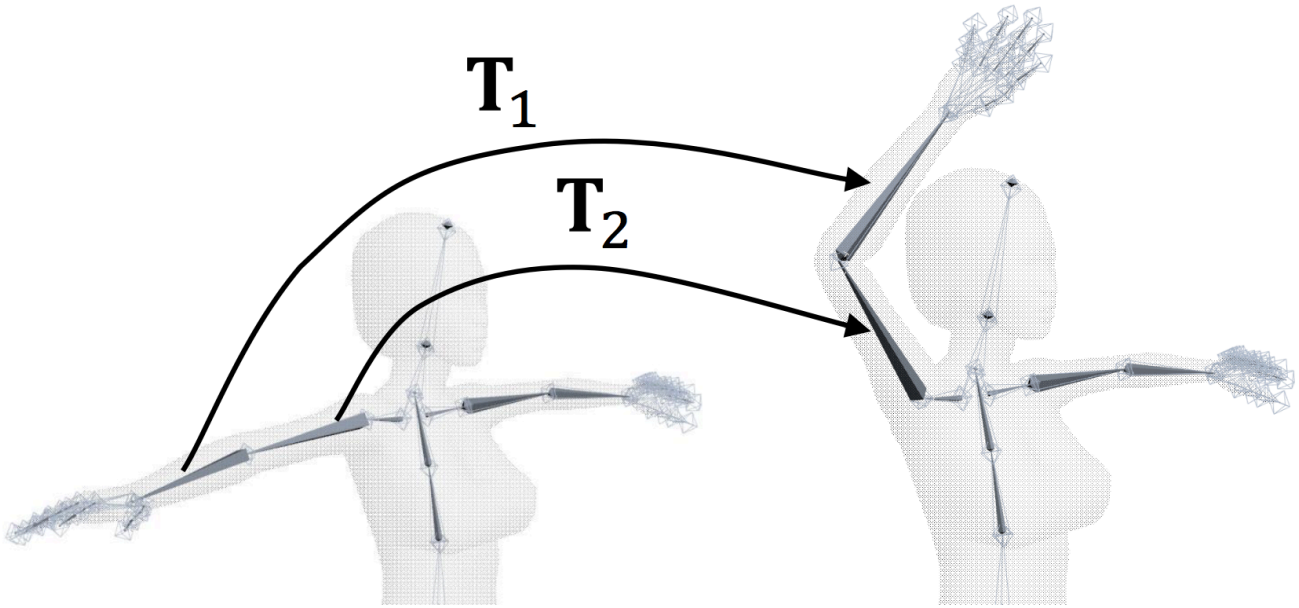


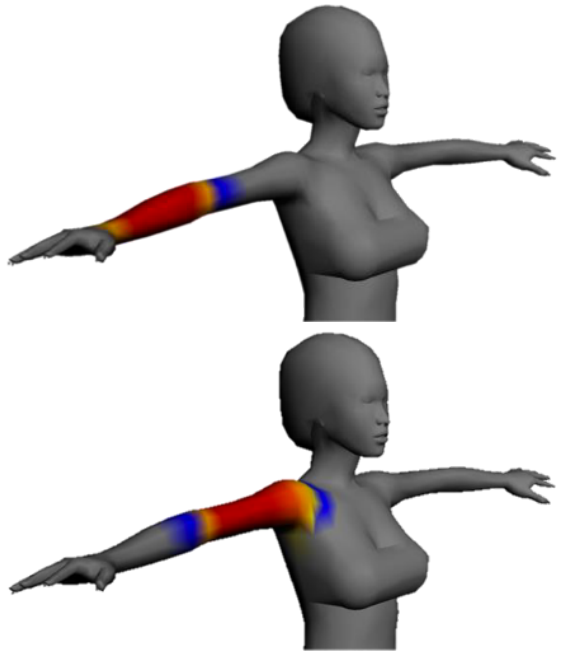
Images from Ladislav Kavan

Linear Blend Skinning

- Transform a vertex $\mathbf{x}_i$ that depends on **several** bones

$$\begin{pmatrix} \mathbf{x}_i' \\ 1 \end{pmatrix} = \sum_{j=1}^{m} w_{i,j} \mathbf{T}_j \begin{pmatrix} \mathbf{x}_i \\ 1 \end{pmatrix}$$





Images from Ladislav Kavan

Let’s try

Linear Blend Skinning

- Transform a vertex $\mathbf{x}_i$ that depends on several bones

$$\begin{pmatrix} \mathbf{x}_i' \\ 1 \end{pmatrix} = \sum_{j=1}^{m} w_{i,j} \mathbf{T}_j \begin{pmatrix} \mathbf{x}_i \\ 1 \end{pmatrix}$$

- Leads to well-known problems at joints

"candy wrapper"

"collapsing joint"

Where do these problems come from?

Images from Ladislav Kavan

Linear Blend Skinning

- Extreme 2D example: Blend between identity and 180° rotation:

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

“candy wrapper”

- Linear combination of rigid motions

$$\sum_{j=1}^{m} w_{i,j}\,\mathbf{T}_j$$

is not a rigid motion anymore!

How to blend rigid motions? Dual Quaternion Skinning!

Summary Skeleton-Based Animation

- **Rigging**
  - Bone weights define influence of bones on vertices
  - Weights must sum up to one and should be non-negative
- **Skeleton Forward Kinematics**
  - Local matrices map from joint coordinates to parent coordinates
  - Global matrices map from joint coordinates to root coordinates
  - Define 4×4 rigid transformation $\mathbf{T}_j$ for each joint j.
- **Linear Blend Skinning**
  - Convex combination of rigid transformations: $\mathbf{T} = \sum_j w_j \mathbf{T}_j$
  - Resulting transformation $\mathbf{T}$ is **not rigid** anymore
  - Artifacts: candy-wrapper and joint collapse 😢
  - Advanced Method: Dual Quaternion Skinning

Let’s try

Literature

- **Skeleton Animation**
  - Kavan et al, *Skinning with Dual Quaternions*, Symposium on Interactive 3D Graphics and Games 2007
  - Jacobson et al, *Skinning: Real-time Shape Deformation*, Course at SIGGRAPH 2014

Quiz: Linear Blend Skinning

What is the reason for the skinning artifacts?
What does it have to do with the blended matrix $\mathbf{T} := \sum_j w_{i,j}\mathbf{T}_j$?

A: $\mathbf{T}$ is not symmetric anymore

B: $\mathbf{T}$ is not orthogonal anymore

C: $\mathbf{T}$ is not rigid motion anymore

D: $\mathbf{T}$ is not invertible anymore