8.2 The Bind Pose & Limitations

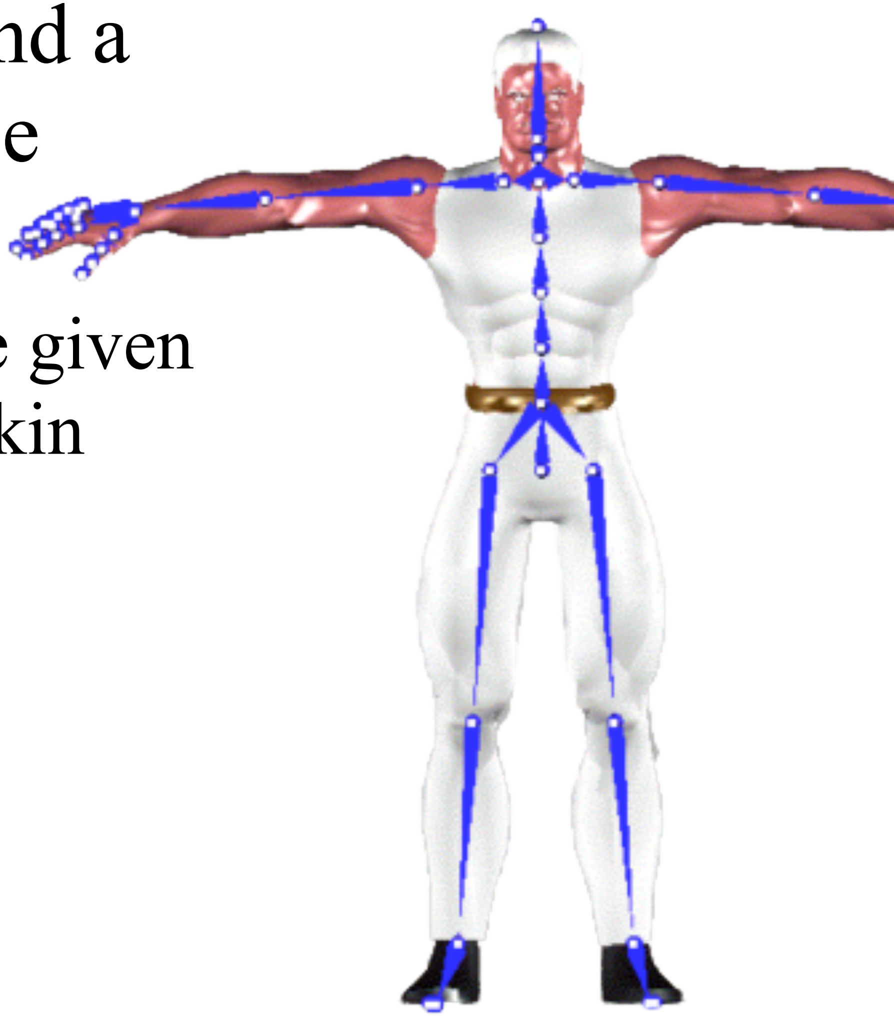
Epic Games / ign.com

In This Video

- The Bind Pose: matching the skeleton to the skin
- Skinning limitations and workarounds

Bind Pose

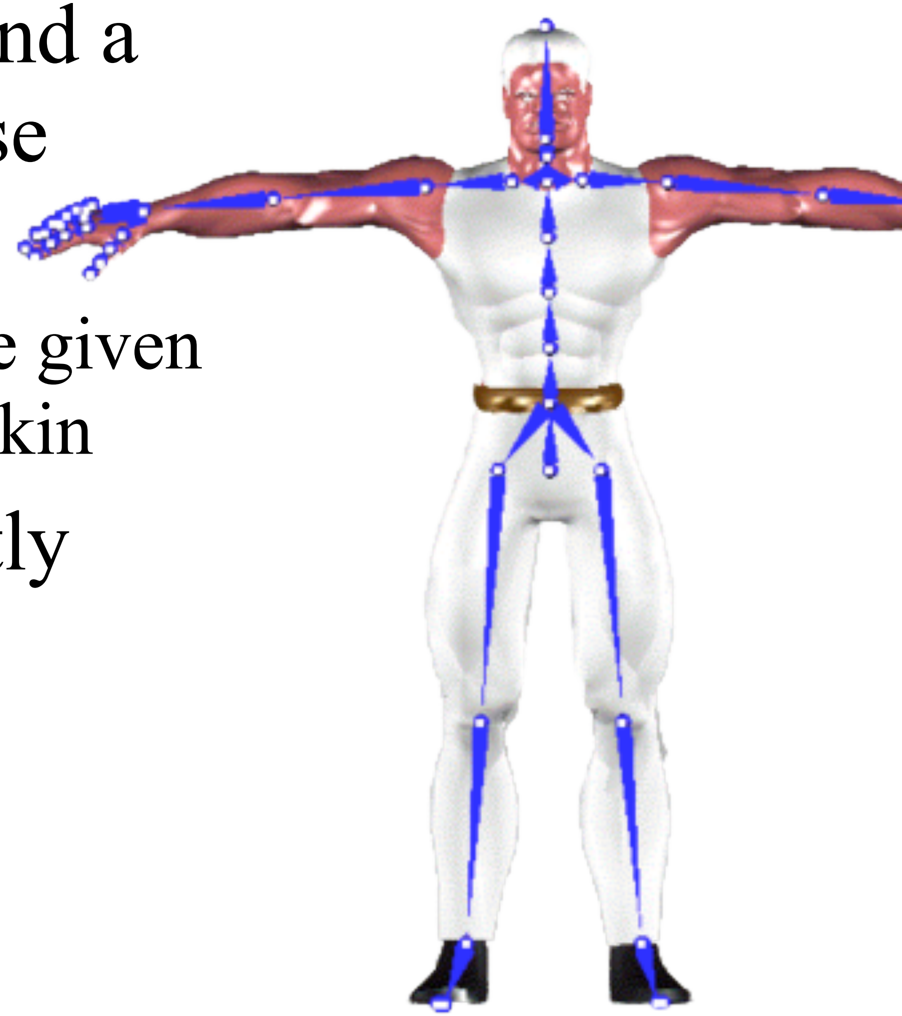
- We are given a skeleton and a skin mesh in a default pose
  - Called "bind pose"
  - Undeformed vertices $\mathbf{p}_i$ are given in the object space of the skin



okino.com

Bind Pose

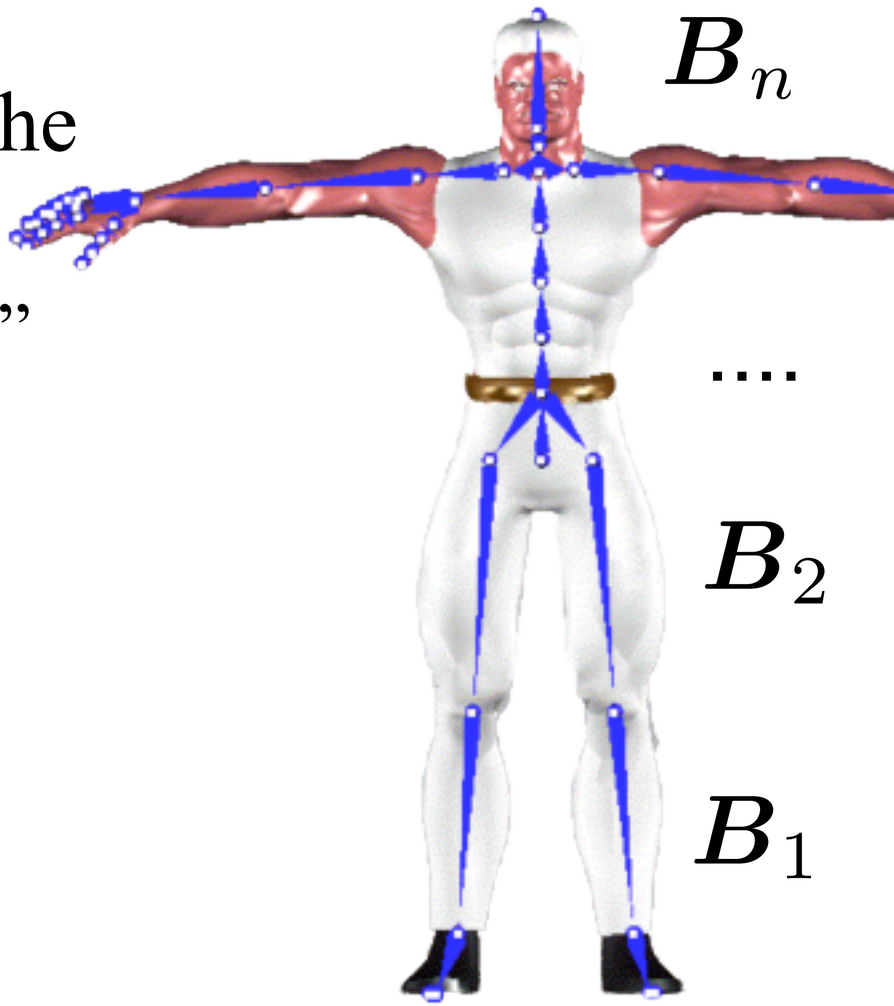
- We are given a skeleton and a skin mesh in a default pose
  - Called "bind pose"
  - Undeformed vertices $\mathbf{p}_i$ are given in the object space of the skin
- Previously we conveniently forgot that in order for $\mathbf{p}'_{ij} = \mathbf{T}_j\ \mathbf{p}_i$ to make sense, coordinate systems must match up.



okino.com

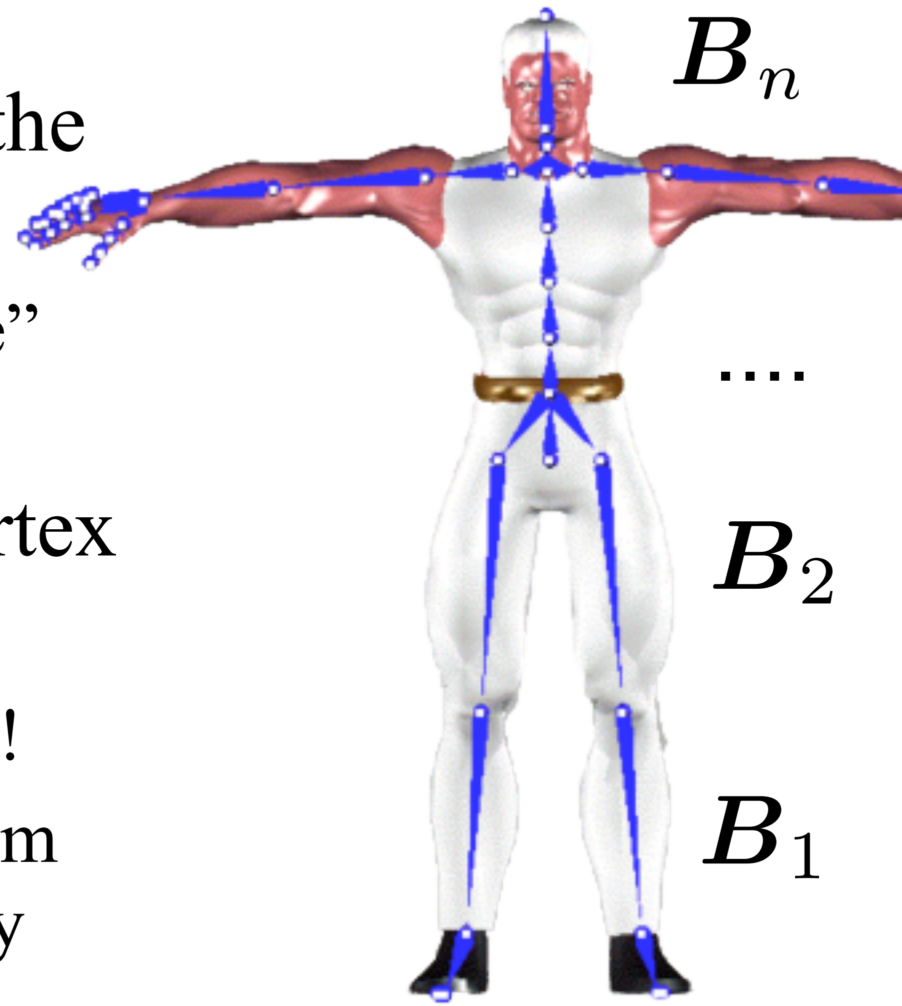
Bind Pose cont'd

- In the rigging phase, we line the skeleton up with the undeformed skin.
  - This gives some "rest pose" bone transformations $\mathbf{B}_j$



Bind Pose cont'd

- In the rigging phase, we line the skeleton up with the undeformed skin.
  - This gives some "rest pose" bone transformations $\mathbf{B}_j$
- We then figure out the vertex weights w_{ij}.
  - How? Often paint by hand!
  - Pinocchio was the algorithm that does this automatically



okino.com

Bind Pose cont'd

- When we animate the model, the bone transformations $\mathbf{T}_j$ change.

okino.com

Bind Pose cont'd

- When we animate the model, the bone transformations $\mathbf{T}_j$ change.
  - What is $\mathbf{T}_j$? It maps from the local coordinate system of bone *j* to object space.
    - Remember hierarchical modeling!

Bind Pose cont'd

- When we animate the model, the bone transformations $\mathbf{T}_j$ change.
  - What is $\mathbf{T}_j$? It maps from the local coordinate system of bone j to object space.
- To be able to deform $\mathbf{p}_i$ according to $\mathbf{T}_j$, we must first express $\mathbf{p}_i$ in the local coordinate system of bone j.
  - This is where the bind pose bone transformations $\mathbf{B}_j$ come in.

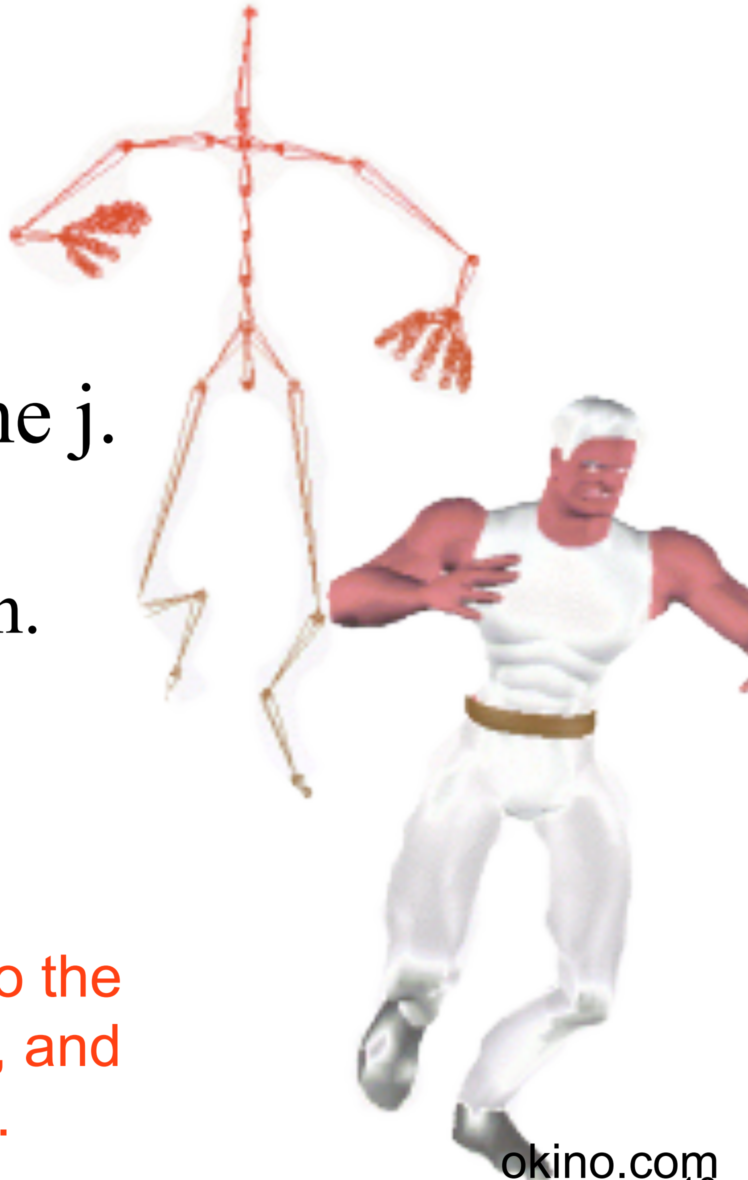
okino.com

Bind Pose cont'd

- To be able to deform $\mathbf{p}_i$ according to $\mathbf{T}_j$, we must first express $\mathbf{p}_i$ in the local coordinate system of bone j.
  - This is where the bind pose bone transformations $\mathbf{B}_j$ come in.

$$\boxed{\boldsymbol{p}'_{ij} = \boldsymbol{T}_j \boldsymbol{B}_j^{-1} \boldsymbol{p}_i}$$

This maps $\mathbf{p}_i$ from bind pose object space to the local coordinate system of bone j using $\mathbf{B}^{-1}_j$, and then to deformed object space using $\mathbf{T}_j$.



okino.com

Bind Pose cont'd

$$\boxed{\boldsymbol{p}'_{ij} = \boldsymbol{T}_j \boldsymbol{B}_j^{-1} \boldsymbol{p}_i}$$

This maps $\mathbf{p}_i$ from bind pose object space to the local coordinate system of bone j using $\mathbf{B}^{-1}_j$, and then to deformed object space using $\mathbf{T}_j$.

What is $\mathbf{T}_j\ \mathbf{B}^{-1}_j$? It is the relative change between the bone transformations between the current and the bind pose.

Bind Pose cont'd

What is the transformation when the model is still in bind pose?

$$\boxed{\boldsymbol{p}'_{ij} = \boldsymbol{T}_j \boldsymbol{B}_j^{-1} \boldsymbol{p}_i}$$

This maps $\mathbf{p}_i$ from bind pose object space to the local coordinate system of bone j using $\mathbf{B}^{-1}_j$, and then to deformed object space using $\mathbf{T}_j$.

What is $\mathbf{T}_j\ \mathbf{B}^{-1}_j$? It is the relative change between the bone transformations between the current and the bind pose.

Bind Pose cont'd

What is the transformation when the model is still in bind pose?

The identity!

$$\boxed{\boldsymbol{p}'_{ij} = \boldsymbol{T}_j \boldsymbol{B}_j^{-1} \boldsymbol{p}_i}$$

This maps $\mathbf{p}_i$ from bind pose object space to the local coordinate system of bone j using $\mathbf{B}^{-1}_j$, and then to deformed object space using $\mathbf{T}_j$.

What is $\mathbf{T}_j\ \mathbf{B}^{-1}_j$? It is the relative change between the bone transformations between the current and the bind pose.

Skinning Pseudocode

- Do the usual forward kinematics
  - maybe quaternion interpolation for rotations
  - get a matrix $\mathbf{T}_j(t)$ per bone
- For each skin vertex $\mathbf{p}_i$

$$\boldsymbol{p}'_i = \sum_j w_{ij} \boldsymbol{T}_j(t) \boldsymbol{B}_j^{-1} \boldsymbol{p}_i$$

Skinning Pseudocode

- Do the usual forward kinematics
  - maybe quaternion interpolation for rotations
  - get a matrix $\mathbf{T}_j(t)$ per bone
- For each skin vertex $\mathbf{p}_i$

$$\boldsymbol{p}'_i = \sum_j w_{ij} \boldsymbol{T}_j(t) \boldsymbol{B}_j^{-1} \boldsymbol{p}_i$$

Remember from Lecture 2:
Normals must be treated differently!

Skinning Pseudocode

- Do the usual forward kinematics
  - maybe quaternion interpolation for rotations
  - get a matrix $\mathbf{T}_j(t)$ per bone
- For each skin vertex $\mathbf{p}_i$

$$\boldsymbol{p}_i' = \sum_j w_{ij} \boldsymbol{T}_j(t) \boldsymbol{B}_j^{-1} \boldsymbol{p}_i$$

- Inverse transpose for normals!

$$\boldsymbol{n}_i' = \left(\sum_j w_{ij} \boldsymbol{T}_j(t) \boldsymbol{B}_j^{-1} \right)^{-\mathrm{T}} \boldsymbol{n}_i$$

Skinning Pseudocode

- Do the usual forward kinematics
  - maybe quaternion interpolation for rotations
  - get a matrix $\mathbf{T}_j(t)$ per bone
- For each skin vertex $\mathbf{p}_i$

$$\boldsymbol{p}'_i = \sum_j w_{ij} \boldsymbol{T}_j(t) \boldsymbol{B}_j^{-1} \boldsymbol{p}_i$$

- Note that the weights are constant over time
  - Only a small number of matrices change
  - This enables implementation on GPU "vertex shaders" (little information to update for each frame)

Hmmh...

- This is what we do to get deformed positions

$$\boldsymbol{p}'_i = \sum_j w_{ij} \boldsymbol{T}_j(t) \boldsymbol{B}_j^{-1} \boldsymbol{p}_i$$

Hmmh...

- This is what we do to get deformed positions

$$\boldsymbol{p}_i' = \sum_j w_{ij} \boldsymbol{T}_j(t) \boldsymbol{B}_j^{-1} \boldsymbol{p}_i$$

- But wait...

$$\boldsymbol{p}_i' = \left(\sum_j w_{ij} \boldsymbol{T}_j(t) \boldsymbol{B}_j^{-1} \right) \boldsymbol{p}_i$$

Hmmh...

- This is what we do to get deformed positions

$$\boldsymbol{p}_i' = \sum_j w_{ij} \boldsymbol{T}_j(t) \boldsymbol{B}_j^{-1} \boldsymbol{p}_i$$

- But wait...

$$\boldsymbol{p}_i' = \left(\sum_j w_{ij} \boldsymbol{T}_j(t) \boldsymbol{B}_j^{-1} \right) \boldsymbol{p}_i$$

- This is *exactly* what I warned you of earlier: blending matrices entry-by-entry (!!!)

Indeed... Limitations

- Rotations really need to be combined differently (quaternions!)

Figure 2: The 'collapsing elbow' in action, c.f. Figure 1.

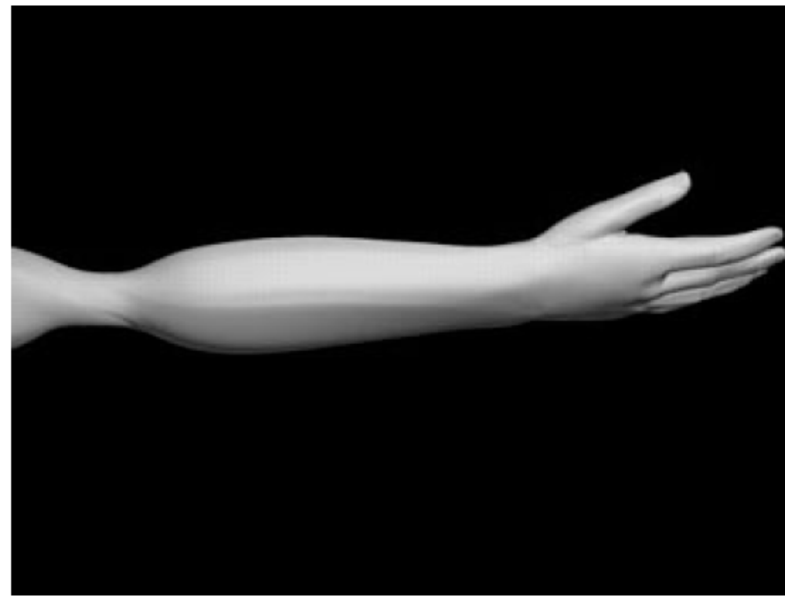


Figure 3: The forearm in the 'twist' pose, as in turning a door handle, computed by SSD. As the twist approaches 180° the arm collapses.

- From: Pose Space Deformation: A Unified Approach to Shape Interpolation and
- Skeleton-Driven Deformation
- J. P. Lewis, Matt Cordner, Nickson Fong

Dual Quaternion Skinning

The common skin deformation technique (linear blend skinning) sometimes produces non-natural results, such as below.

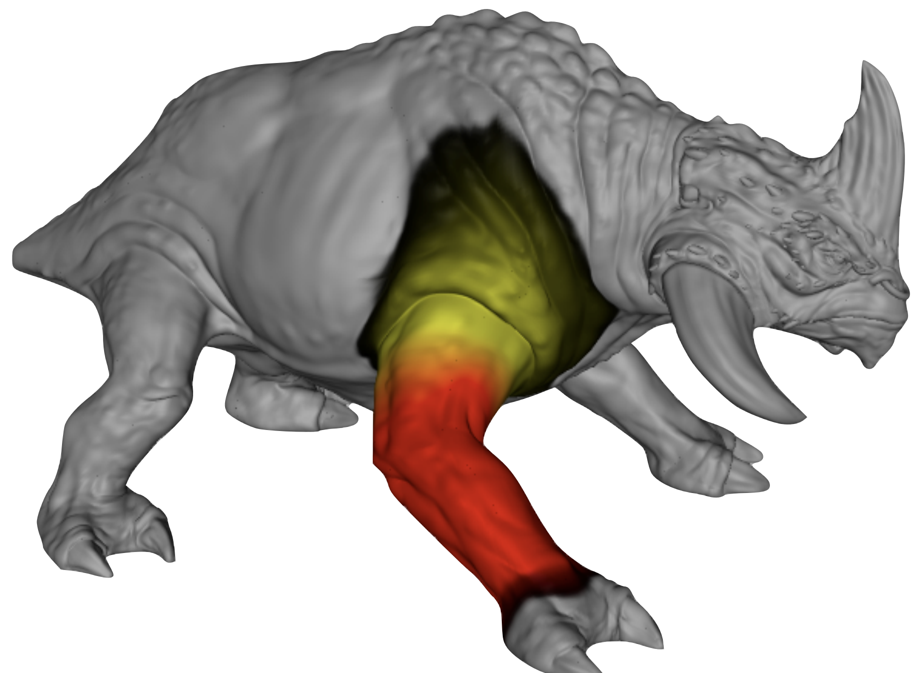
Several more advanced methods exist, but lead to slower run-time performance and require a more expensive character setup. Our approach, based on dual quaternions, removes the skinning artifacts at a small cost in computational efficiency and no cost at all in character setup. The result of our method is pictured below (using the same model files as above).

Usual Solution

- Have the artists deal with it O:-)
- In practice, build a rig that has extra bones near joints that move in sync to counter the artifacts.
  - Tedious, but this is what people often do.
  - Need sophisticated animation controls to drive this.
- Cool paper that does this automatically: Kavan, Collins, O'Sullivan: <u>Automatic Linearization of Nonlinear Skinning, I3D 2009</u>

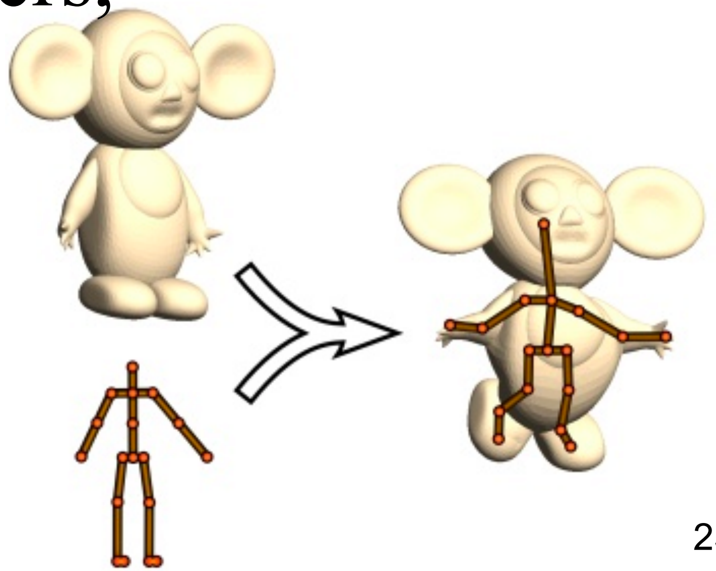
Figuring out the Weights

- Usual approach: Paint them on the skin.
- Can also find them by optimization from example poses and deformed skins.
  - Wang & Phillips, SCA 2002



Super Cool: Automatic Rigging

- When you just have some reference skeleton animation (perhaps from motion capture) and a skin mesh, figure out the bone transformations and vertex weights!
- Ilya Baran, Jovan Popovic: Automatic Rigging and Animation of 3D Characters, SIGGRAPH 2007
  - http://www.mit.edu/~ibaran/autorig/
- You saw this earlier



The Other Direction

- When you have no skeleton, but a source animation for the full mesh (not so common)

Skinning Mesh Animations

Doug L. James Christopher D. Twigg

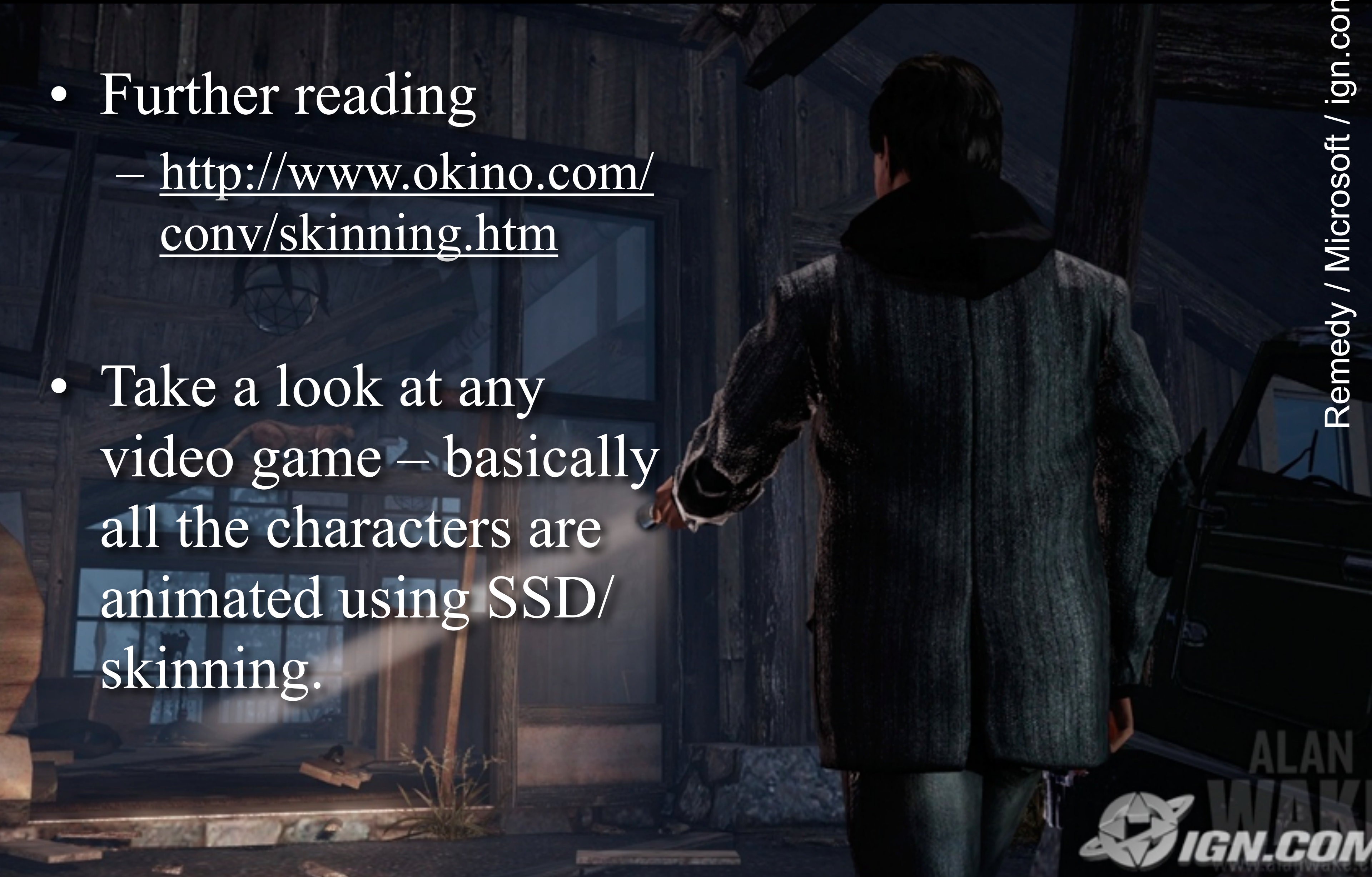
Carnegie Mellon University

igure 1: **Stampede!** *Ten thousand skinned mesh animations (SMAs) synthesized in graphics hardware at interactive rates. All SMAs are eformed using only traditional matrix palette skinning with well-chosen nonrigid bone transforms. Distant SMAs are simplified.*

That's All!

- Further reading
  - http://www.okino.com/conv/skinning.htm

- Take a look at any video game – basically all the characters are animated using SSD/skinning.



Remedy / Microsoft / ign.com