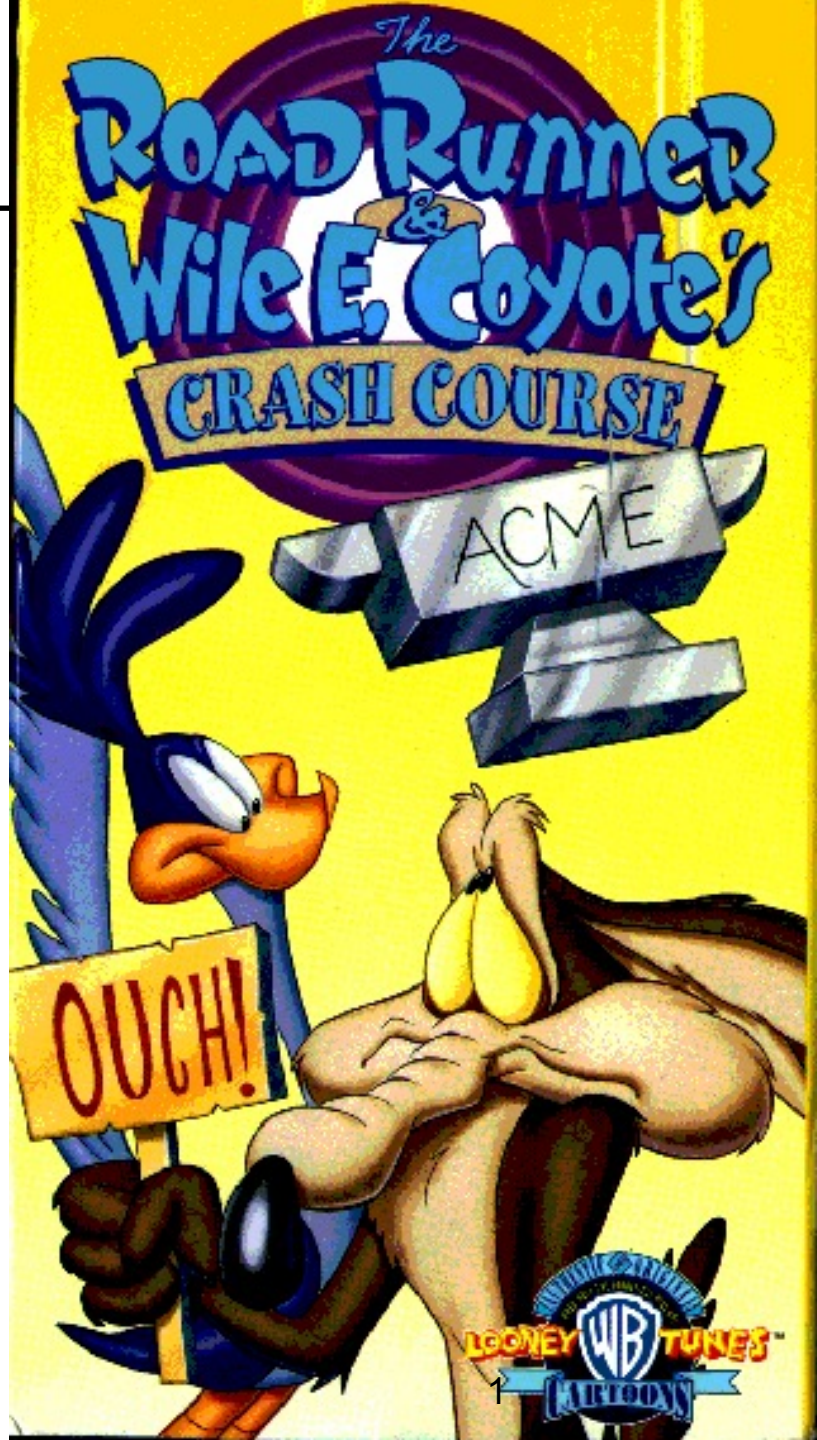
Basics of Computer Animation

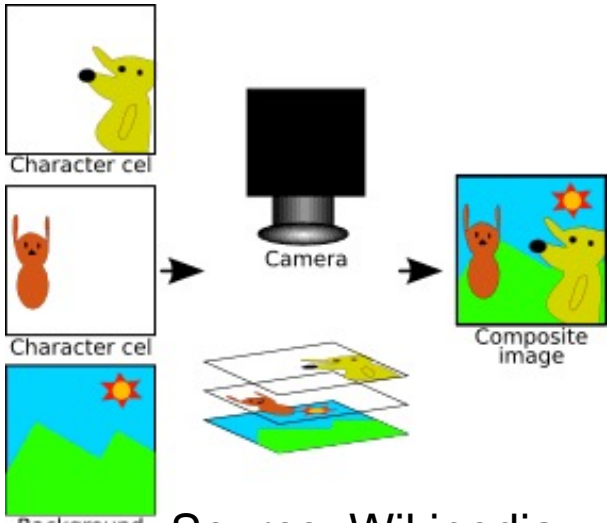
Skinning/Enveloping

Sai-Kit Yeung

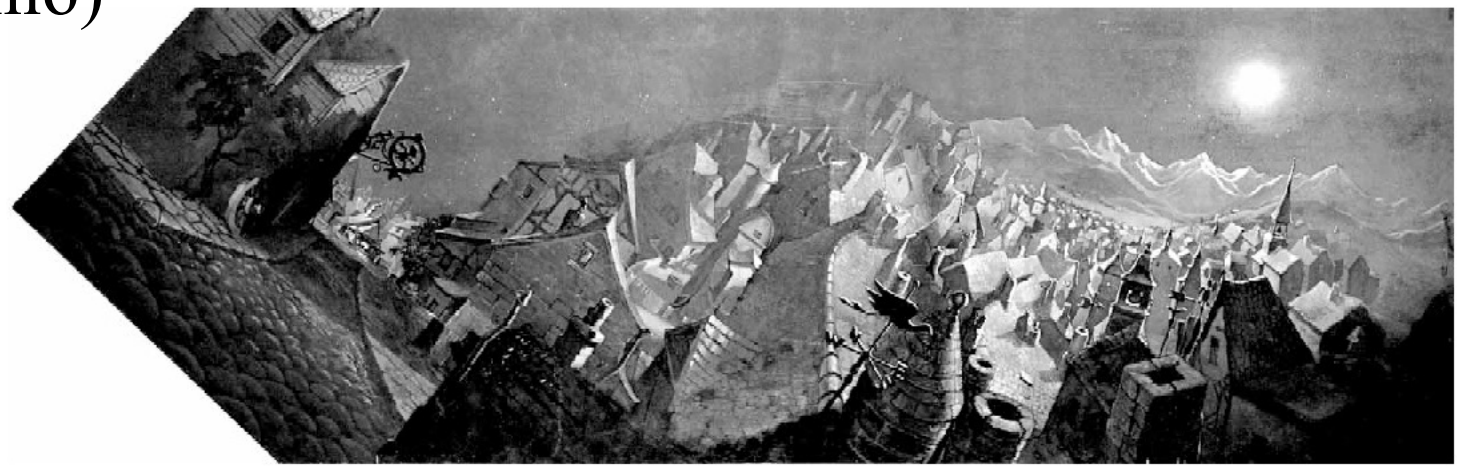


Traditional Animation

- Draw each frame by hand
  - great control, but tedious
- Reduce burden with cel animation
  - Layer, keyframe, inbetween, …
  - Example: Cel panoramas (Disney's Pinocchio)



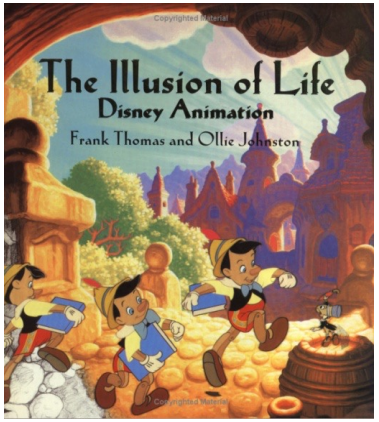
Source: Wikipedia



ACM © 1997 "Multiperspective panoramas for cel animation"

Traditional Animation Principles

- The in-betweening, was once a job for apprentice animators. Splines accomplish these tasks automatically. However, the animator still has to draw the keyframes. This is an art form and precisely why the experienced animators were spared the in-betweening work even before automatic techniques.
- The classical paper on animation by John Lasseter from Pixar surveys some the standard animation techniques:
- Principles of Traditional Animation Applied to 3D Computer Graphics, SIGGRAPH'87, pp. 35-44.
- See also



Example: Squash and Stretch

- Squash: flatten an object or character by pressure or by its own power

- Stretch: used to increase the sense of speed and emphasize the squash by contrast

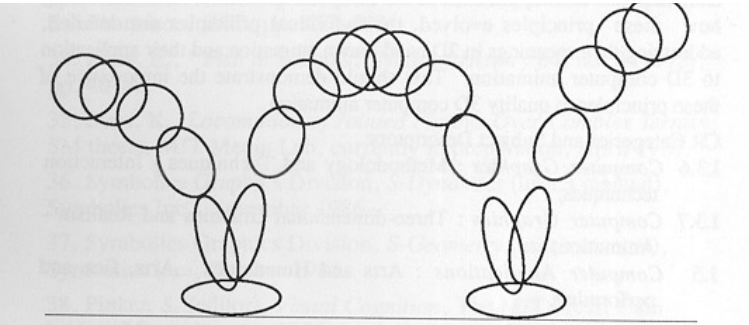


FIGURE 2. Squash & stretch in bouncing ball.

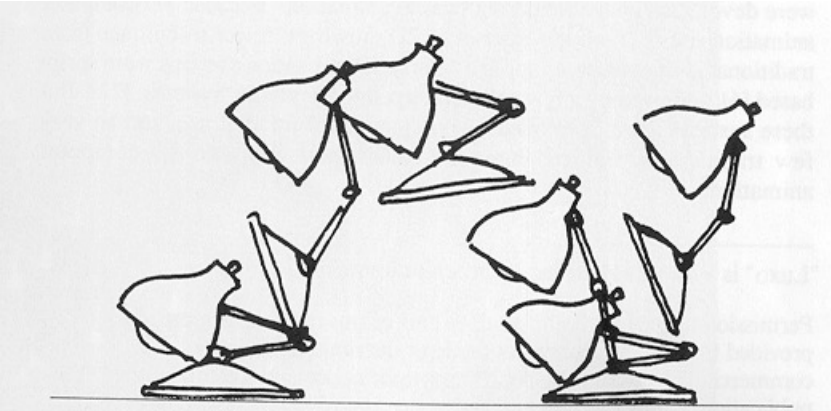


FIGURE 3. Squash & stretch in Luxo Jr.'s hop.

Example: Timing

- Timing affects weight:
  - Light object move quickly
  - Heavier objects move slower

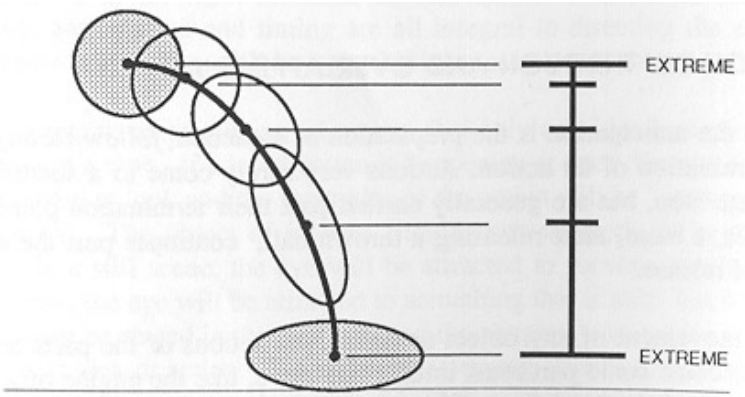
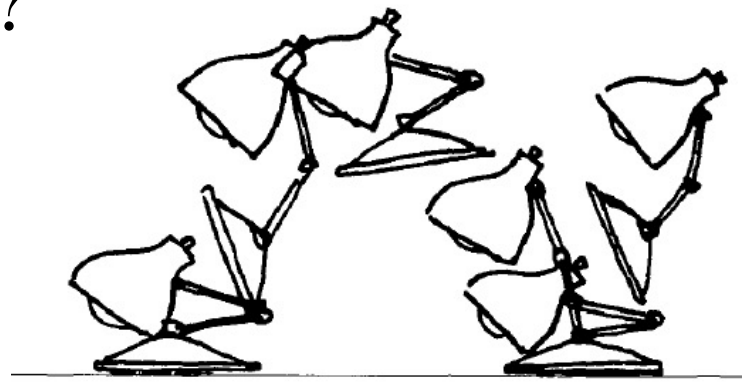


FIGURE 9. Timing chart for ball bounce.

- Timing completely changes the interpretation of the motion.

Computer Animation

- How do we describe and generate motion of objects in the scene?



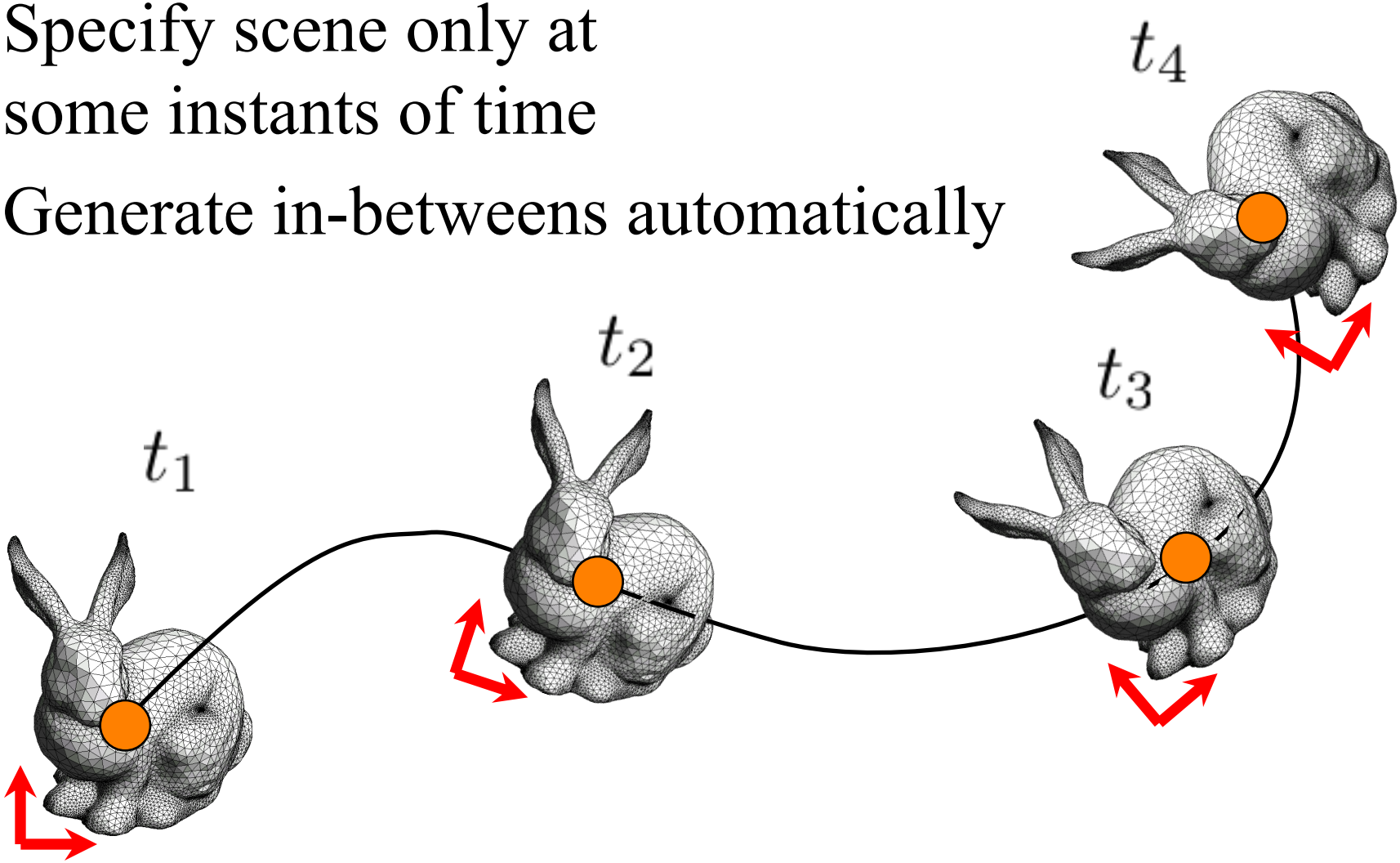
- Two very different contexts:
  - Production (offline)
    - Can be hardcoded, entire sequence know beforehand
  - Interactive (e.g. games, simulators)
    - Needs to react to user interaction, sequence not known

Plan

- Types of Animation (overview)
  - Keyframing
  - Procedural
  - Physically-based
- Animation Controls
- Character Animation
  using skinning/enveloping

Types of Animation: Keyframing

- Specify scene only at some instants of time
- Generate in-betweens automatically

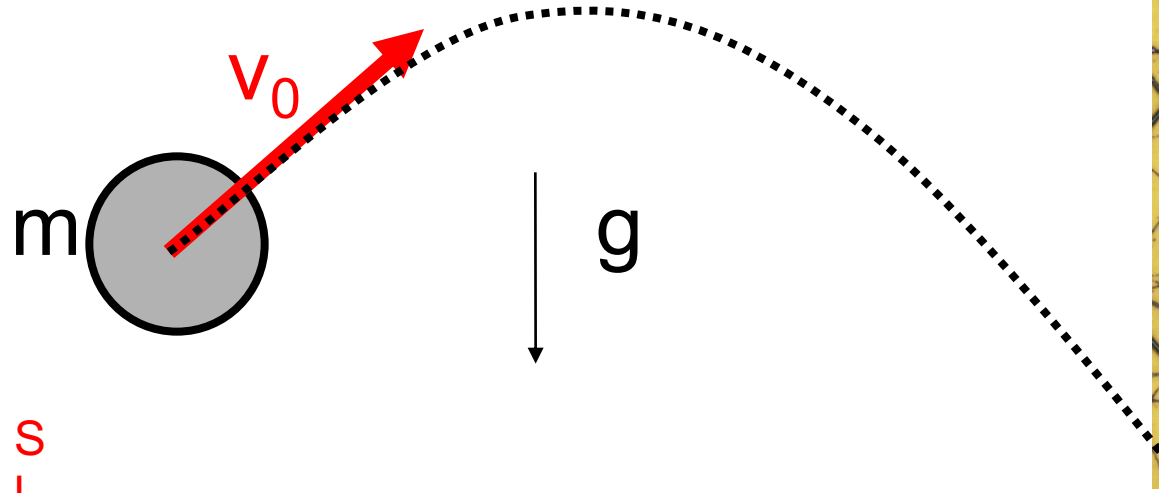


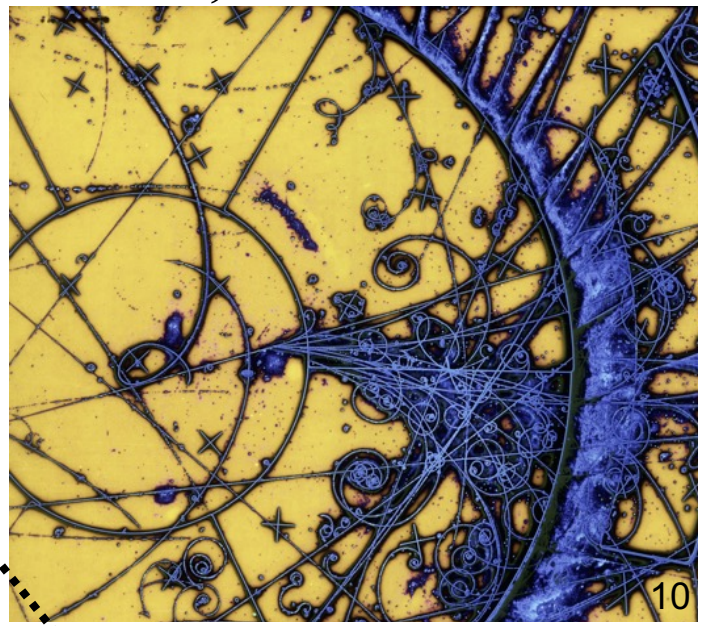
Types of Animation: Procedural

- Describes the motion algorithmically
- Express animation as a function of small number of parameters
- Example
  - a clock/watch with second, minute and hour hands
  - express the clock motions in terms of a "seconds" variable
    - the clock is animated by changing this variable
- Another example: Grass in the wind, tree canopies, etc.

Types of Animation: Physically-Based

- Assign physical properties to objects
  - Masses, forces, etc.
- Also procedural forces (like wind)
- Simulate physics by solving equations of motion
  - Rigid bodies, fluids, plastic deformation, etc.
- Realistic but difficult to control





Example: Water Simulation

Losasso, F., Talton, J., Kwatra, N. and Fedkiw, R., "Two-way Coupled SPH and Particle Level Set Fluid Simulation", IEEE TVCG 14, 797-804 (2008).

Another Example

- Physically-Based Character Animation
  - Specify keyframes, solve for physically valid motion that interpolates them by "spacetime optimization"

- Anthony C. Fang and Nancy S. Pollard, 2003. Efficient Synthesis of Physically Valid Human Motion, ACM Transactions on Graphics 22(3) 417-426, Proc. SIGGRAPH 2003.
  - http://graphics.cs.cmu.edu/nsp/projects/spacetime/spacetime.html

Plan

- Types of Animation (overview)
  - Keyframing
  - Procedural
  - Physically-based
- Animation Controls
- Character Animation
  using skinning/enveloping

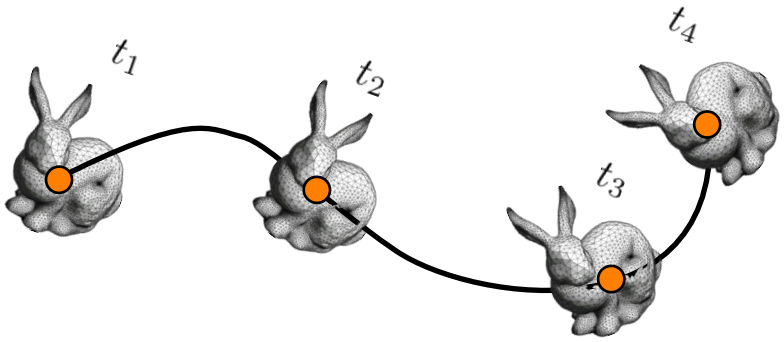
Because we are Lazy...

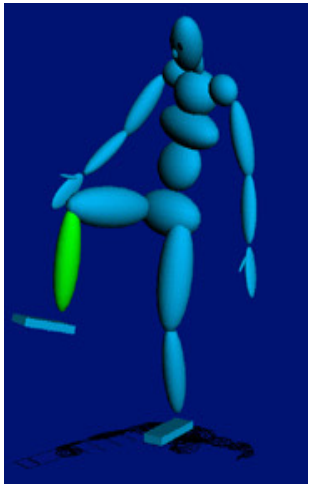
- Animation is (usually) specified using some form of low-dimensional controls as opposed to remodeling the actual geometry for each frame.

Can you think of examples?

Because we are Lazy...

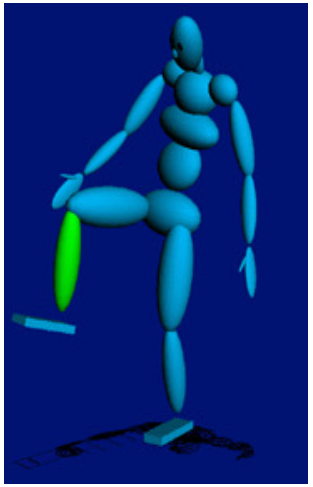
- Animation is (usually) specified using some form of low-dimensional controls as opposed to remodeling the actual geometry for each frame.
  - Example: The joint angles (bone transformations) in a hierarchical character determine the pose
  - Example: A rigid motion is represented by changing the object-to-world transformation (rotation and translation).



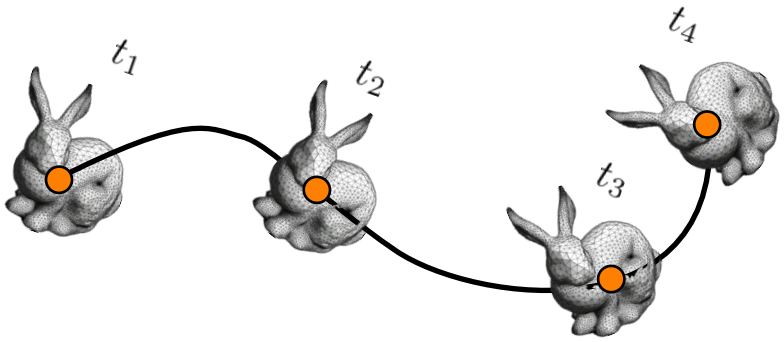


Because we are Lazy...

- Animation is (usually) specified using some form of low-dimensional controls as opposed to remodeling the actual geometry for each frame.
  - Example: The joint angles (bone transformations) in a hierarchical character determine the pose
  - Example: A rigid motion is represented by changing the object-to-world transformation (rotation and translation).



"Blendshapes" are keyframes that are just snapshots of the entire geometry.

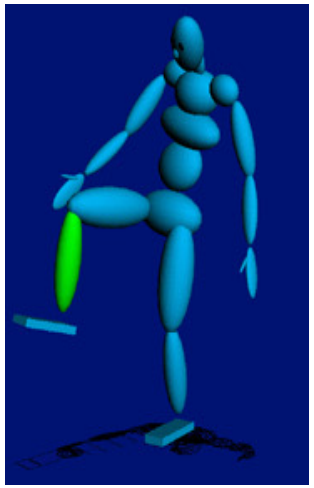


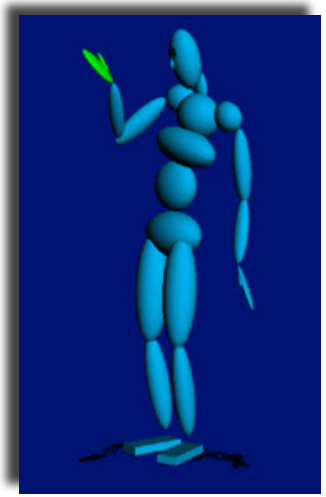
Building 3D models and their animation controls is a major component of every animation pipeline.

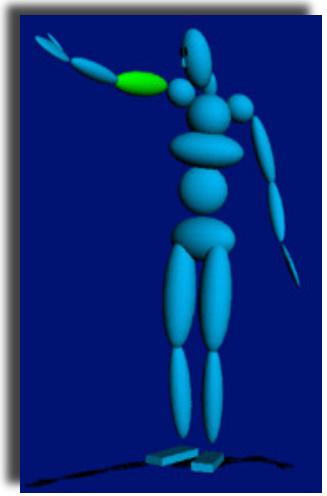
Building the controls is called “rigging”.

Articulated Character Models

- Forward kinematics describes the positions of the body parts as a function of joint angles
  - Body parts are usually called "bones"
  - Angles are the low-dimensional control.
- Inverse kinematics specifies constraint locations for bones and solves for joint angles.







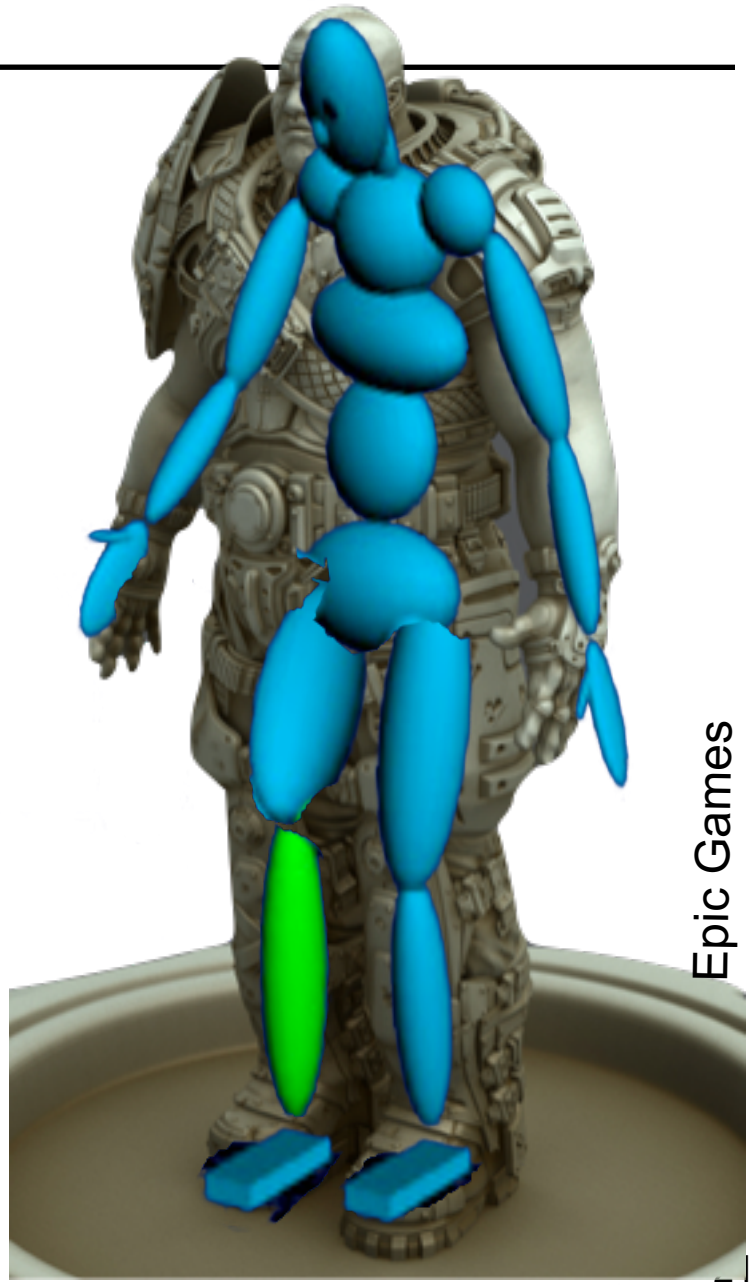
Skinning Characters

- Embed a skeleton into a detailed character mesh

Epic Games

Skinning Characters

- Embed a skeleton into a detailed character mesh
- Animate “bones”
  - Change the joint angles over time
  - Keyframing, procedural, etc.
- Bind skin vertices to bones
  - Animate skeleton, skin will move with it

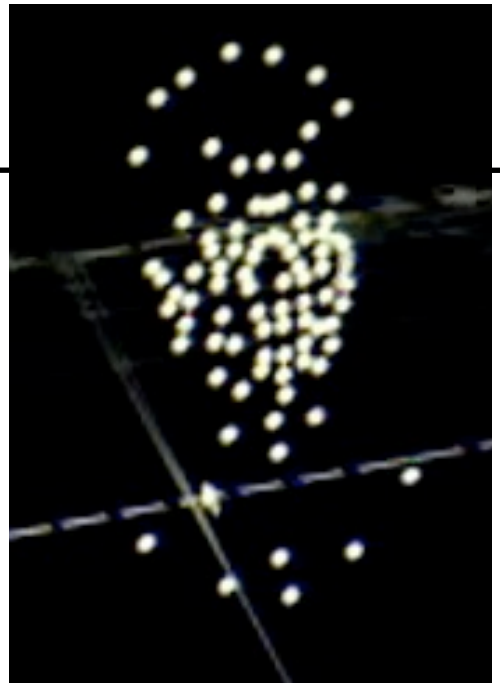


Motion Capture

- Usually uses optical markers and multiple high-speed cameras
- Triangulate to get marker 3D position
  - (Again, structure from motion and projective geometry, i.e., homogeneous coordinates)
- Captures style, subtle nuances and realism
- But need ability to record someone

Example: Facial Motion Capture

Motion Capture



?

- Motion capture records 3D marker positions
  - But character is controlled using animation controls that affect bone transformations!
- Marker positions must be translated into character controls (“retargeting”)
  - Becomes the weights for the blendshape

Markerless Motion Capture

3D Shape Regression for Real-time Facial Animation

Chen Cao[1] Yanlin Weng[1] Stephen Lin[2] Kun Zhou[1]

[1]State Key Lab of CAD&CG, Zhejiang University [2]Microsoft Research Asia

Very Fun paper

ILM / Walt Disney Pictures

ILM / Walt Disney Pictures

ILM / Walt Disney Pictures

Questions?

Plan

- Types of Animation (overview)
  - Keyframing
  - Procedural
  - Physically-based
- Animation Controls
- Character Animation using skinning/enveloping

Skinning/Enveloping

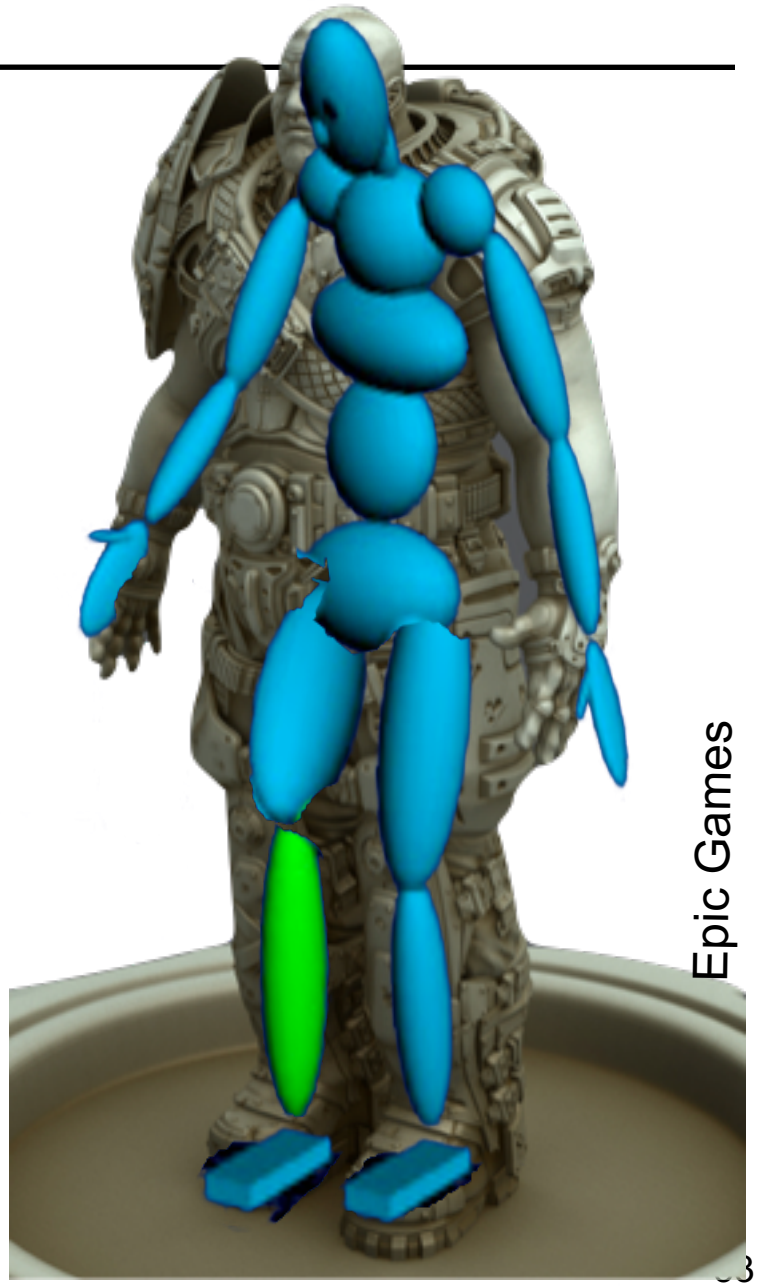
Epic Games / ign.com

Skinning

- We know how to animate a bone hierarchy
  - Change the joint angles, i.e., bone transformations, over time (keyframing)

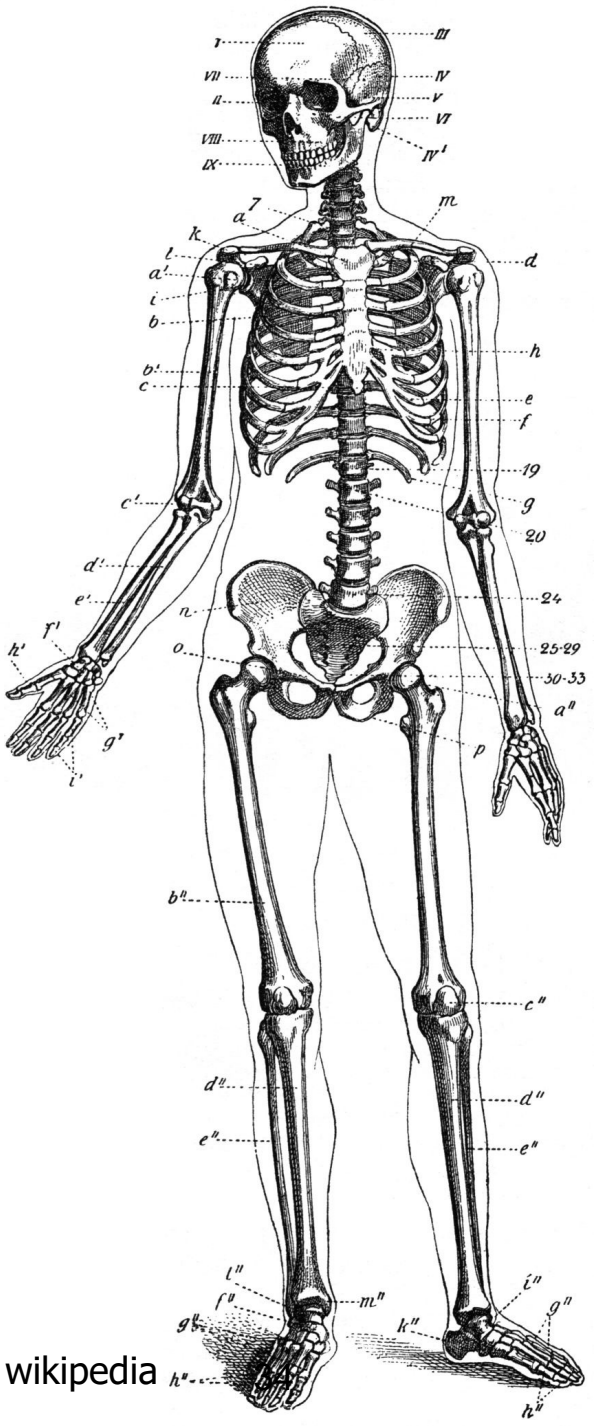
Skinning

- We know how to animate a bone hierarchy
  - Change the joint angles, i.e., bone transformations, over time (keyframing)
- Embed a skeleton into a detailed character mesh
- Bind skin vertices to bones
  - Animate skeleton, skin will move with it
  - But how?



Skinning/Enveloping

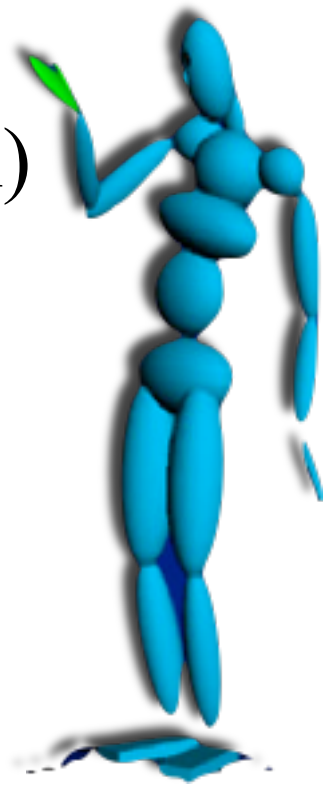
- Need to infer how skin deforms from bone transformations.
- Most popular technique: Skeletal Subspace Deformation (SSD), or simply Skinning
  - Other aliases
    - vertex blending
    - matrix palette skinning
    - linear blend skinning



From wikipedia

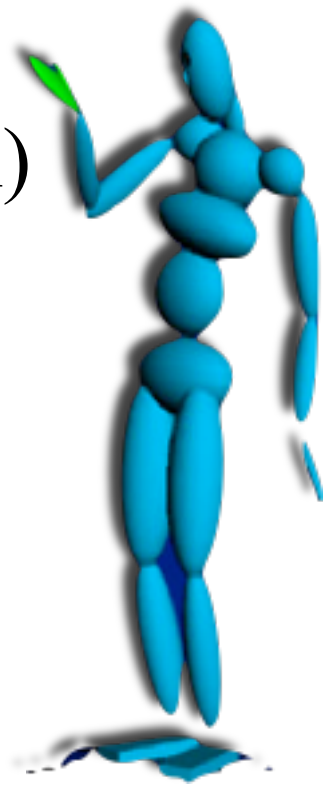
SSD / Skinning

- Each bone has a deformation of the space around it (rotation, translation)



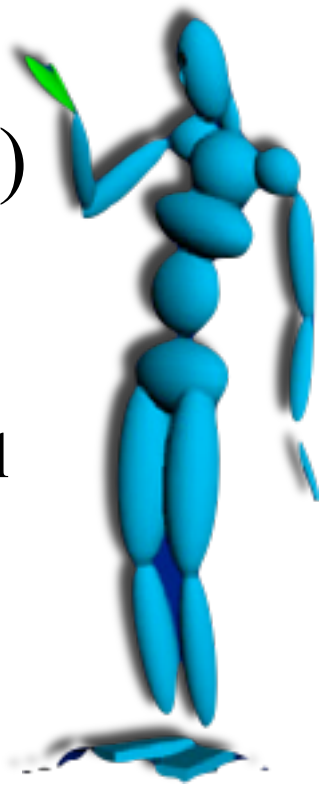
SSD / Skinning

- Each bone has a deformation of the space around it (rotation, translation)
  - What if we attach each vertex of the skin to a single bone?



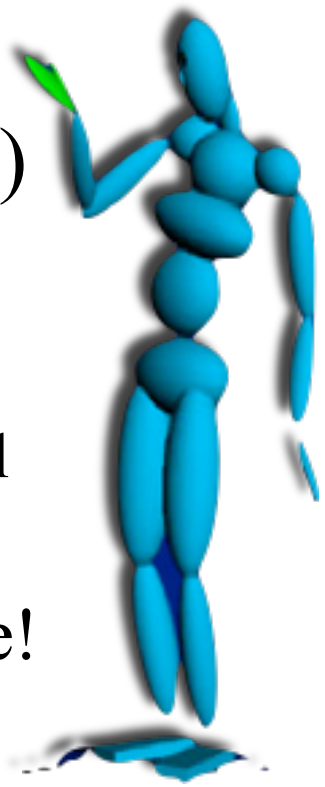
SSD / Skinning

- Each bone has a deformation of the space around it (rotation, translation)
  - What if we attach each vertex of the skin to a single bone?
    - Skin will be rigid, except at joints where it will stretch badly



SSD / Skinning

- Each bone has a deformation of the space around it (rotation, translation)
  - What if we attach each vertex of the skin to a single bone?
    - Skin will be rigid, except at joints where it will stretch badly
  - Let's attach a vertex to many bones at once!
    - In the middle of a limb, the skin points follow the bone rotation (near-rigidly)
    - At a joint, skin is deformed according to a "weighted combination" of the bones



Examples

Colored triangles are attached to 1 bone

Black triangles are attached to more than 1

Note how they are near joints

James & Twigg 2005

Examples

James & Twigg 2005

Colored triangles are attached to 1 bone

Black triangles are attached to more than 1

Note how they are near joints

Vertex Weights

- We'll assign a weight w_{ij} for each vertex p_i for each bone B_j.
  - "How much vertex i should move with bone j"
  - $w_{ij} = 1$ means p_i is rigidly attached to bone j.

Vertex Weights

- We'll assign a weight w_{ij} for each vertex p_i for each bone B_j.
  - "How much vertex i should move with bone j"
  - $w_{ij} = 1$ means p_i is rigidly attached to bone j.

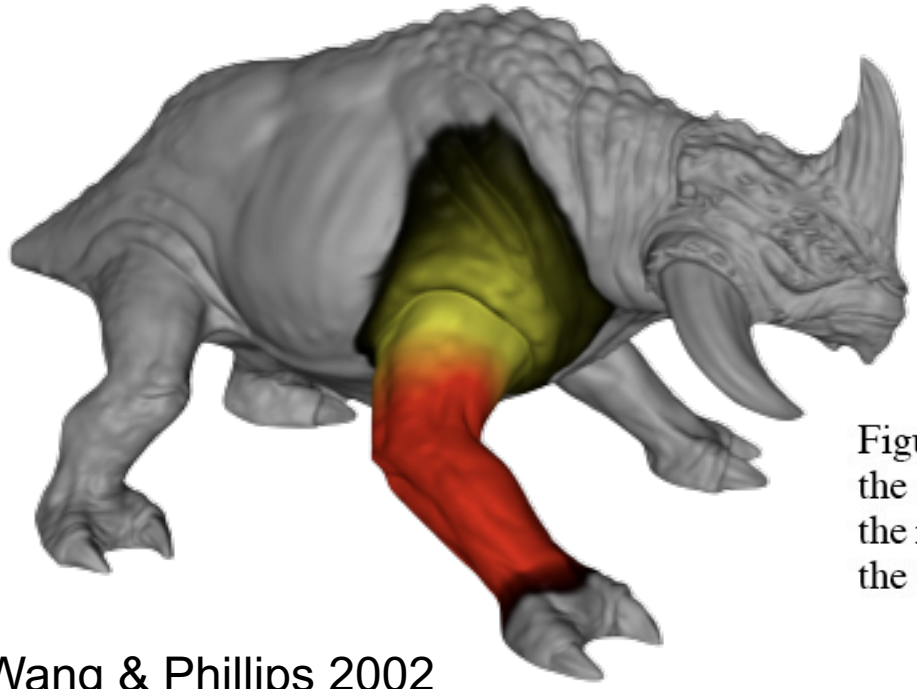


Figure 4: Color coded influences maps from two bones on the animal creature. The yellow area for the upper leg and the red area for the lower leg. The darker the area, the smaller the influence.

Wang & Phillips 2002

Vertex Weights

- We'll assign a weight w_{ij} for each vertex p_i for each bone B_j.
  - "How much vertex i should move with bone j"
  - $w_{ij} = 1$ means p_i is rigidly attached to bone j.
- Weight properties
  - Usually want weights to be non-negative

Vertex Weights

- We'll assign a weight w_{ij} for each vertex p_i for each bone B_j.
  - "How much vertex i should move with bone j"
  - $w_{ij} = 1$ means p_i is rigidly attached to bone j.
- Weight properties
  - Usually want weights to be non-negative
  - Also, want the sum over all bones to be 1 for each vertex

Vertex Weights cont'd

- We'll assign a weight w_{ij} for each vertex p_i for each bone B_j.
  - "How much vertex i should move with bone j"
  - $w_{ij} = 1$ means p_i is rigidly attached to bone j.
- We'll limit the number of bones N that can influence a single vertex
  - N=4 bones/vertex is a usual choice
  - Why?

Vertex Weights cont'd

- We'll assign a weight w_{ij} for each vertex p_i for each bone B_j.
  - "How much vertex i should move with bone j"
  - $w_{ij} = 1$ means p_i is rigidly attached to bone j.
- We'll limit the number of bones N that can influence a single vertex
  - N=4 bones/vertex is a usual choice
  - Why? You most often don't need very many.
  - Also, storage space is an issue.
  - In practice, we'll store N (bone index j, weight w_{ij}) pairs per vertex.

How to compute vertex positions?

Linear Blend Skinning

- Basic Idea 1: Transform each vertex p_i with each bone as if it was tied to it rigidly.

Linear Blend Skinning

- Basic Idea 1: Transform each vertex p_i with each bone as if it was tied to it rigidly.
- Basic Idea 2: Then blend the results using the weights.

Computing Vertex Positions

- Basic Idea 1: Transform each vertex p_i with each bone as if it was tied to it rigidly.
- Basic Idea 2: Then blend the results using the weights.

“

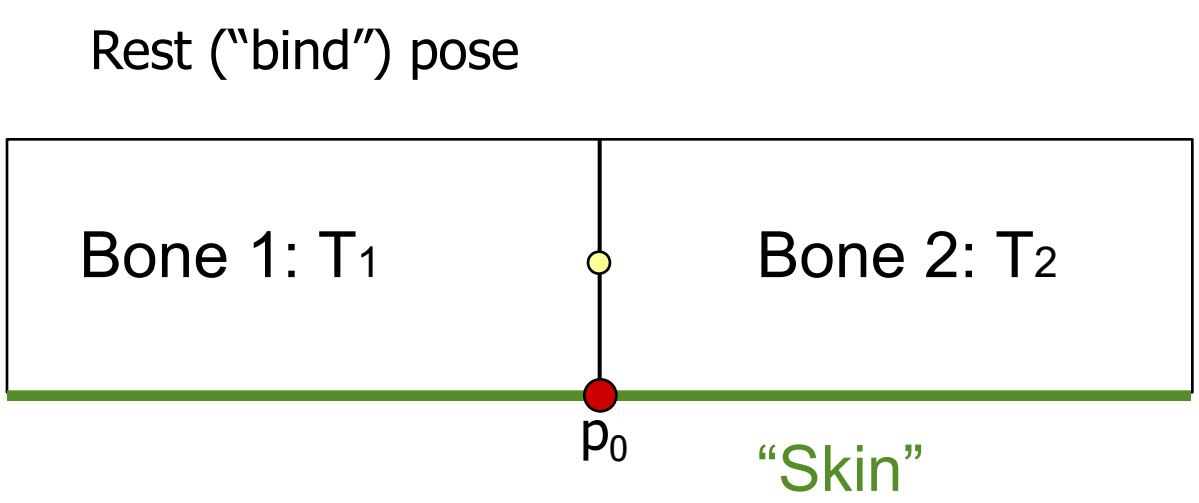
$$\boldsymbol{p}'_{ij} = \boldsymbol{T}_j \, \boldsymbol{p}_i$$

$$\boldsymbol{p}'_i = \sum_j w_{ij} \boldsymbol{p}'_{ij}$$

”

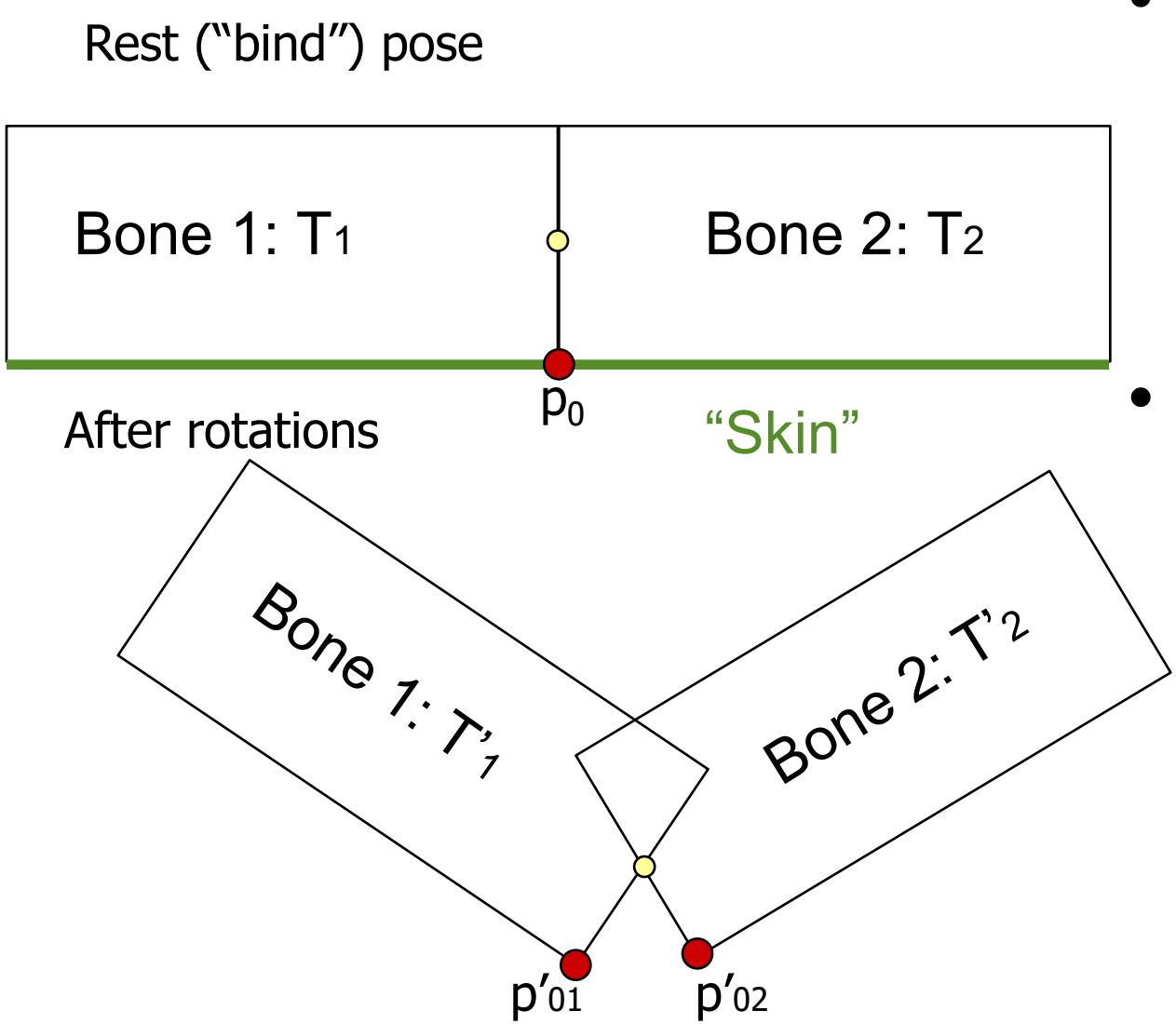
p'_{ij} is the vertex i transformed using bone j.
T_j is the current transformation of bone j.
p'_i is the new skinned position of vertex i.

Computing Vertex Positions



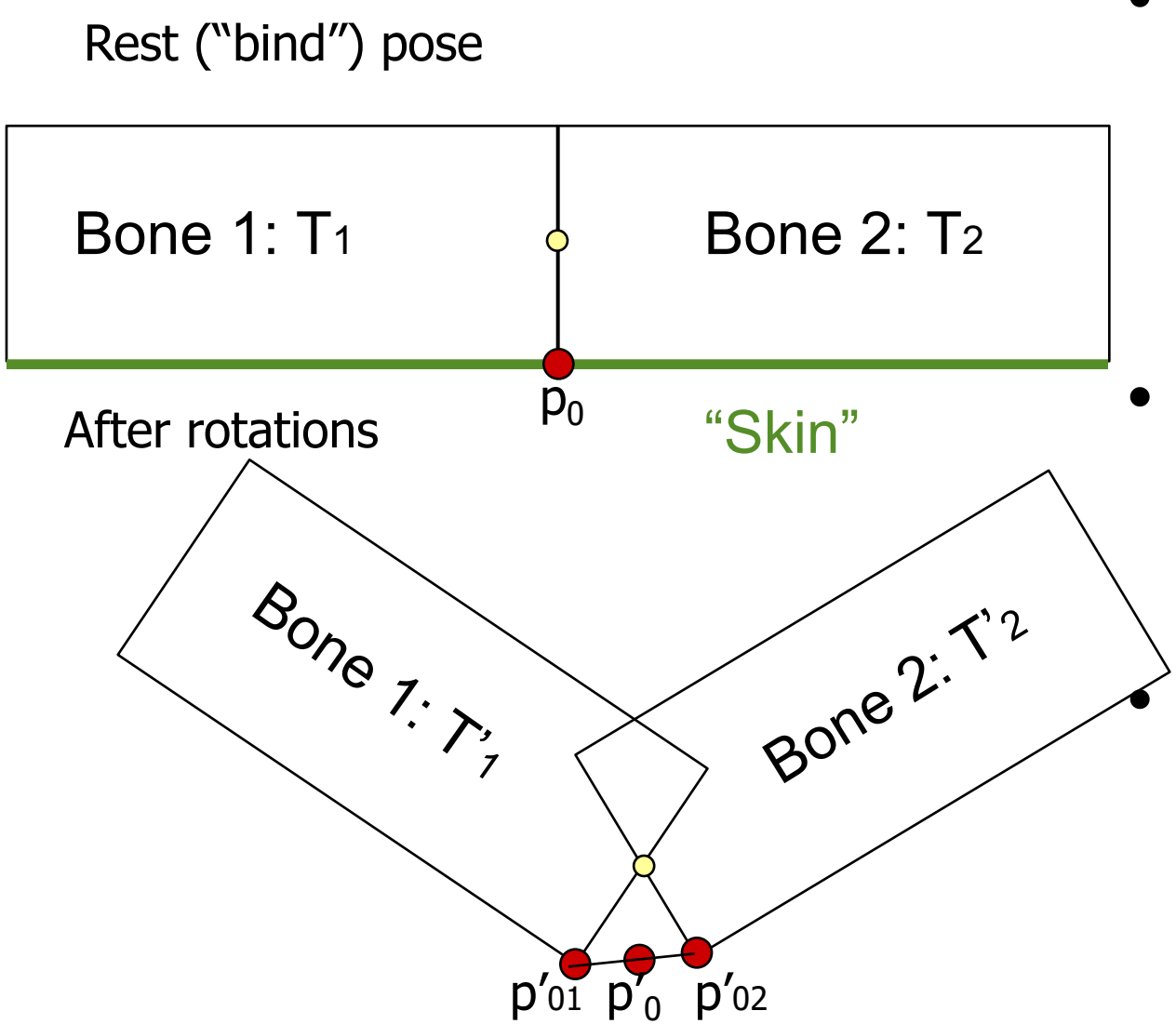
- Vertex p_0 has weights $w_{01}=0.5$, $w_{02}=0.5$

Computing Vertex Positions



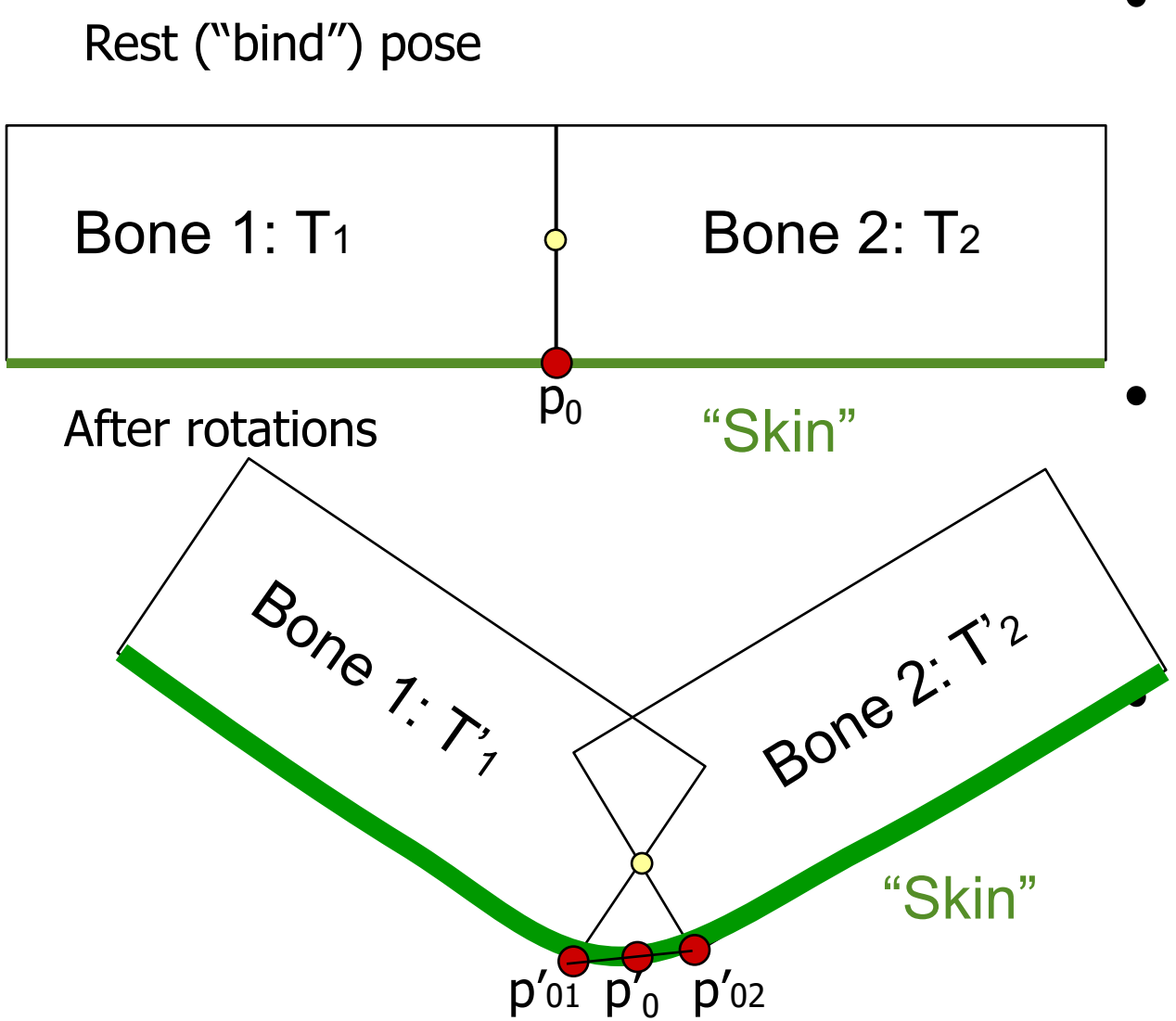
- Vertex p_0 has weights $w_{01}=0.5$, $w_{02}=0.5$
- Transform by T'_1 and T'_2 yields p'_{01}, p'_{02}

Computing Vertex Positions



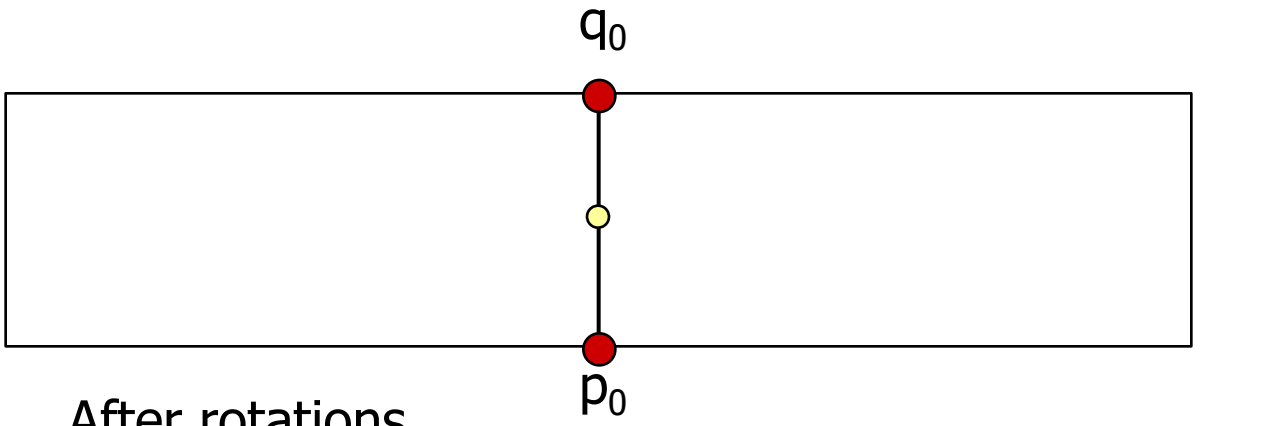
- Vertex p_0 has weights $w_{01}=0.5$, $w_{02}=0.5$
- Transform by T'_1 and T'_2 yields p'_{01}, p'_{02}
- the new position is $p'_0 = 0.5*p'_1 + 0.5*p'_2$

Computing Vertex Positions

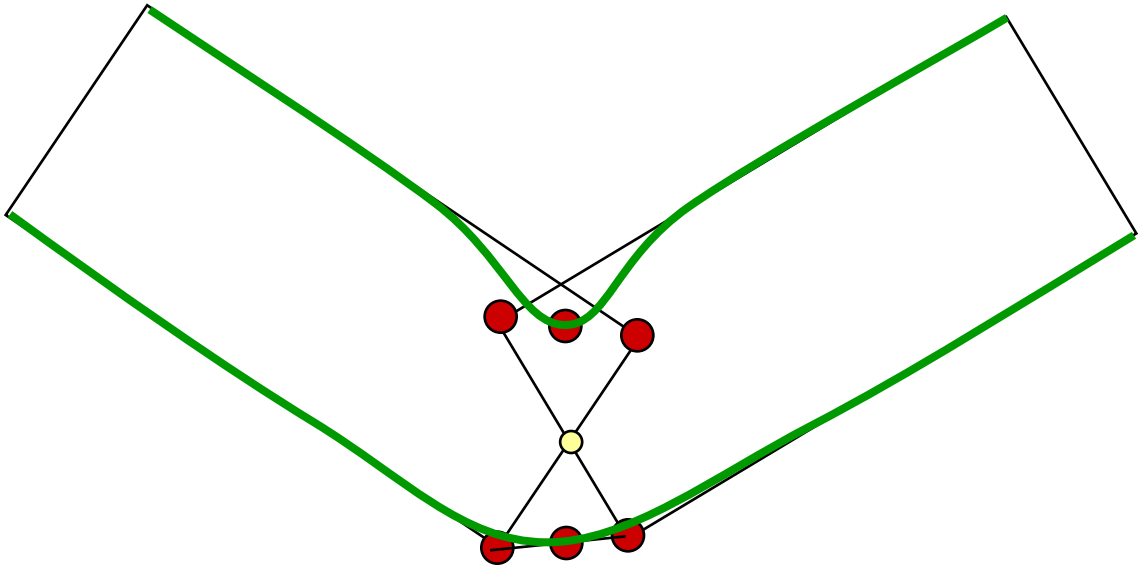


- Vertex p_0 has weights $w_{01}=0.5$, $w_{02}=0.5$
- Transform by T'_1 and T'_2 yields p'_{01}, p'_{02}
- the new position is $p'_0 = 0.5*p'_1 + 0.5*p'_2$

SSD is Not Perfect

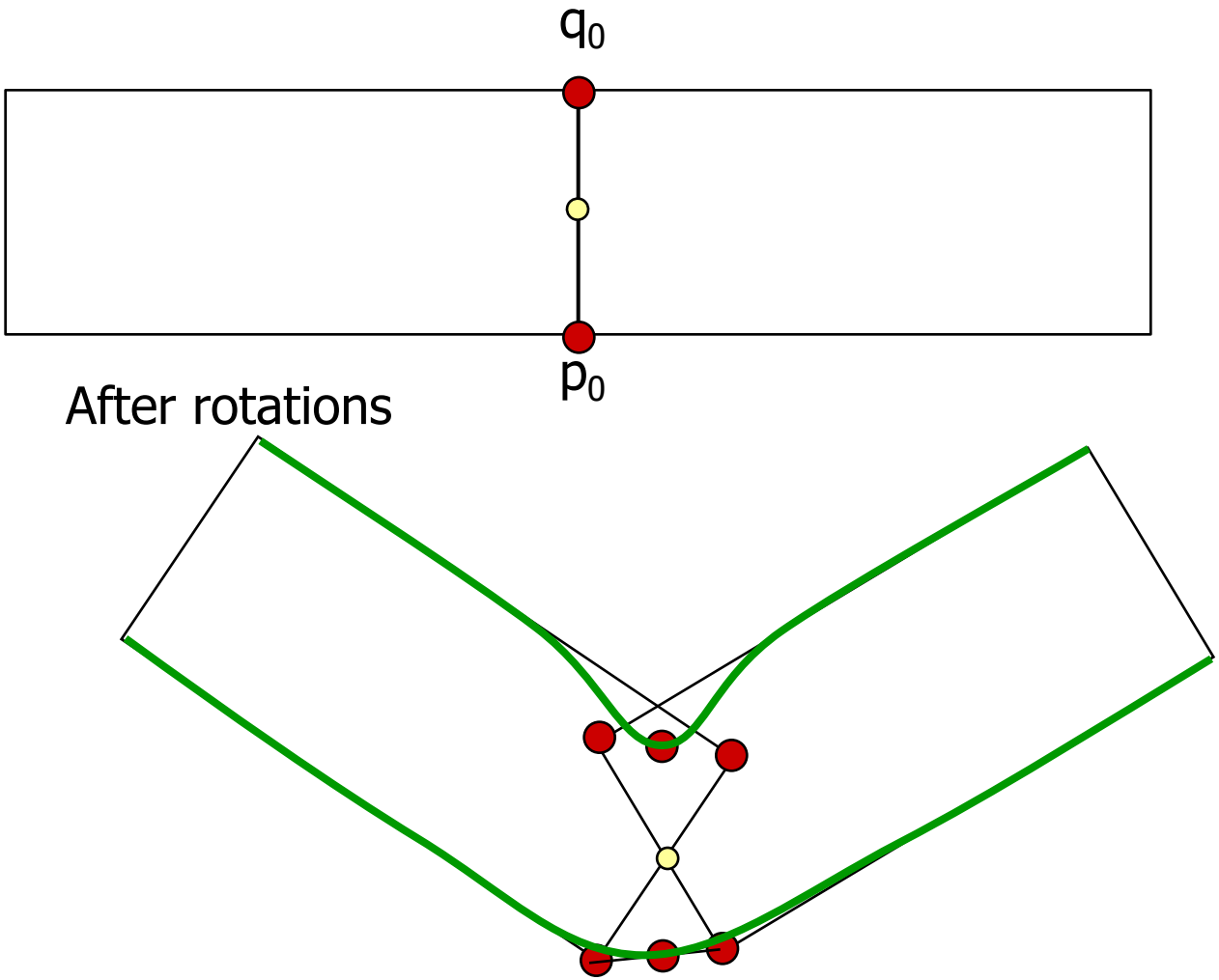


After rotations



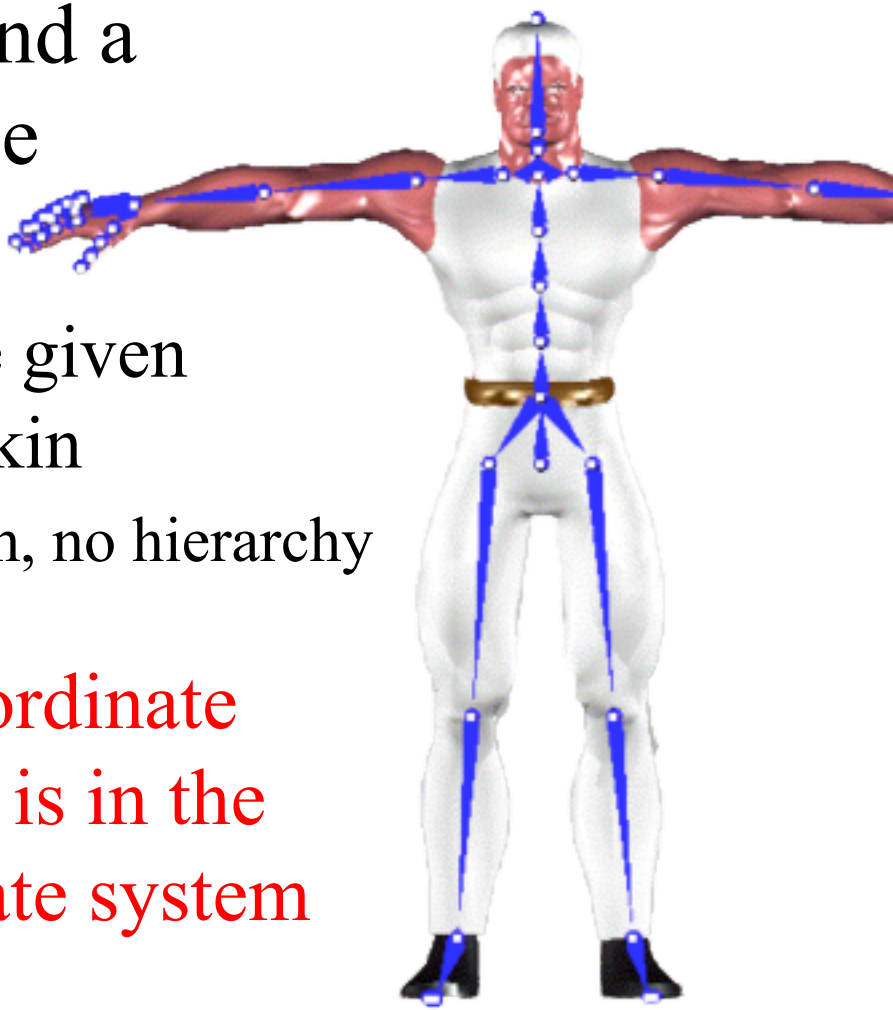
SSD is Not Perfect

Questions?



Bind Pose

- We are given a skeleton and a skin mesh in a default pose
  - Called “bind pose”
  - Undeformed vertices p_i are given in the object space of the skin
    - a “global” coordinate system, no hierarchy

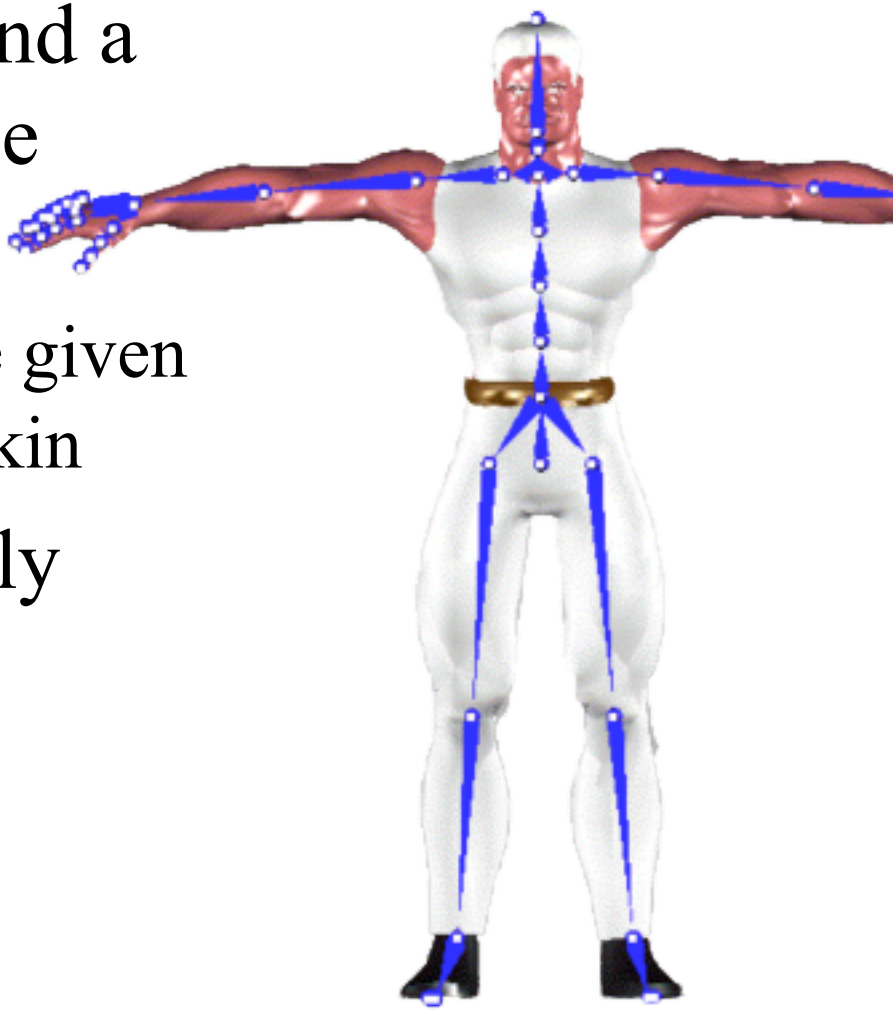


p_i are not in the local coordinate system of each bone – it is in the global or world coordinate system

okino.com

Bind Pose

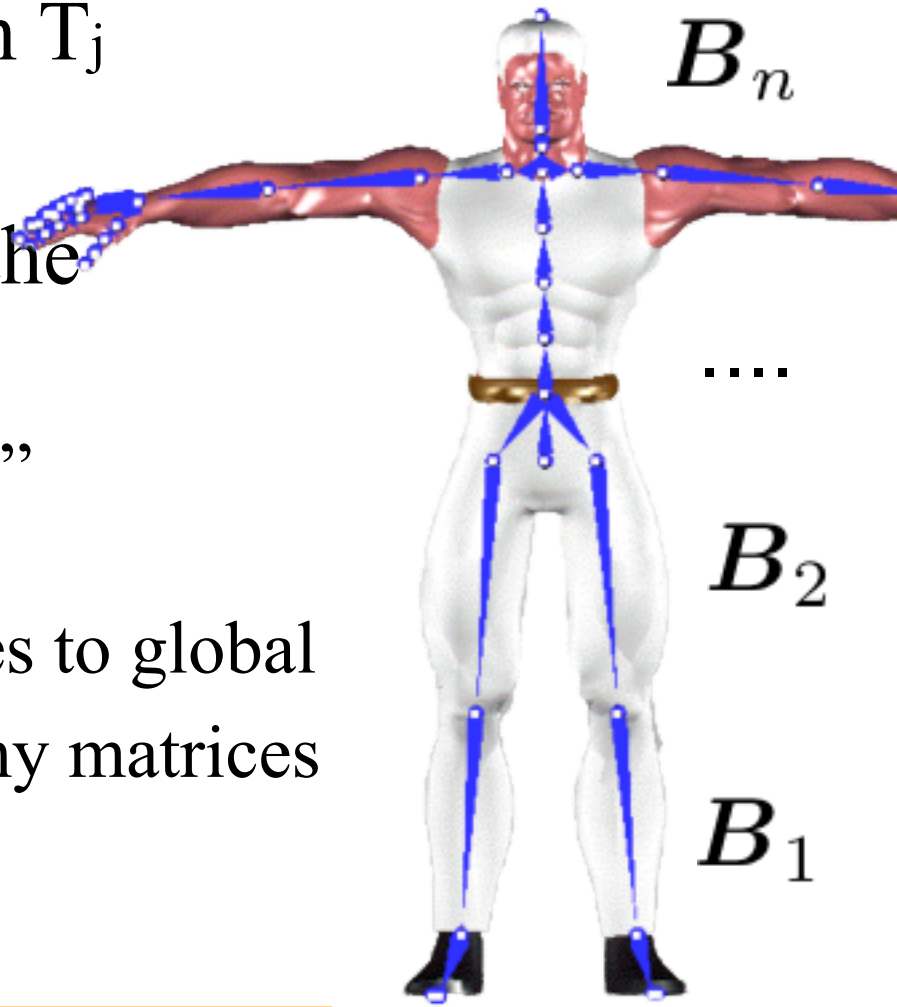
- We are given a skeleton and a skin mesh in a default pose
  - Called “bind pose”
  - Undeformed vertices p_i are given in the object space of the skin
- Previously we conveniently forgot that in order for $p'_{ij} = T_j p_i$ to make sense, coordinate systems must match up.



okino.com

Bind Pose cont'd

- Before any transformation T_j
- In the rigging phase, we line the skeleton up with the undeformed skin.
  - This gives some "rest pose" bone transformations B_j from local bone coordinates to global
  - B_j concatenates all hierarchy matrices from bone j up to the root



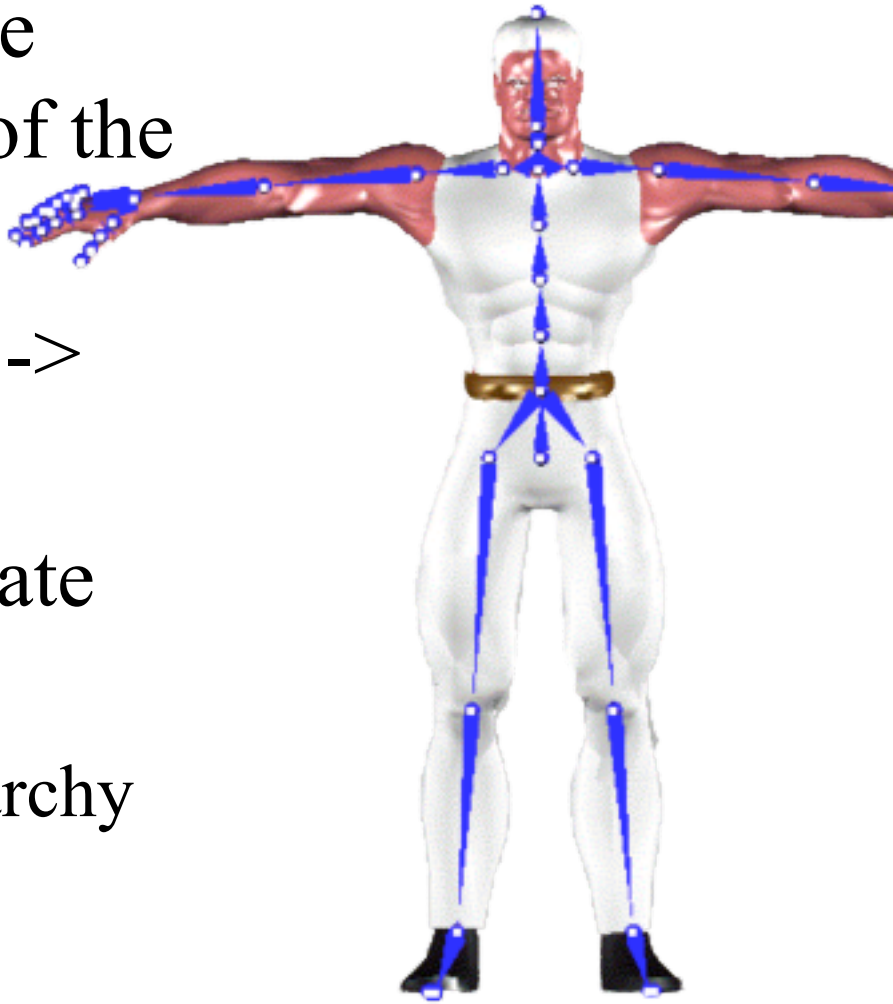
Remember this?

$$\mathbf{v}_w = \mathbf{T}(x_h, y_h, z_h)\mathbf{R}(q_h, f_h, s_h)\ \mathbf{TR}(q_t, f_t, s_t)\,\mathbf{TR}(q_c)\,\mathbf{TR}(q_f, f_f)\,\mathbf{v}_s$$

okino.com

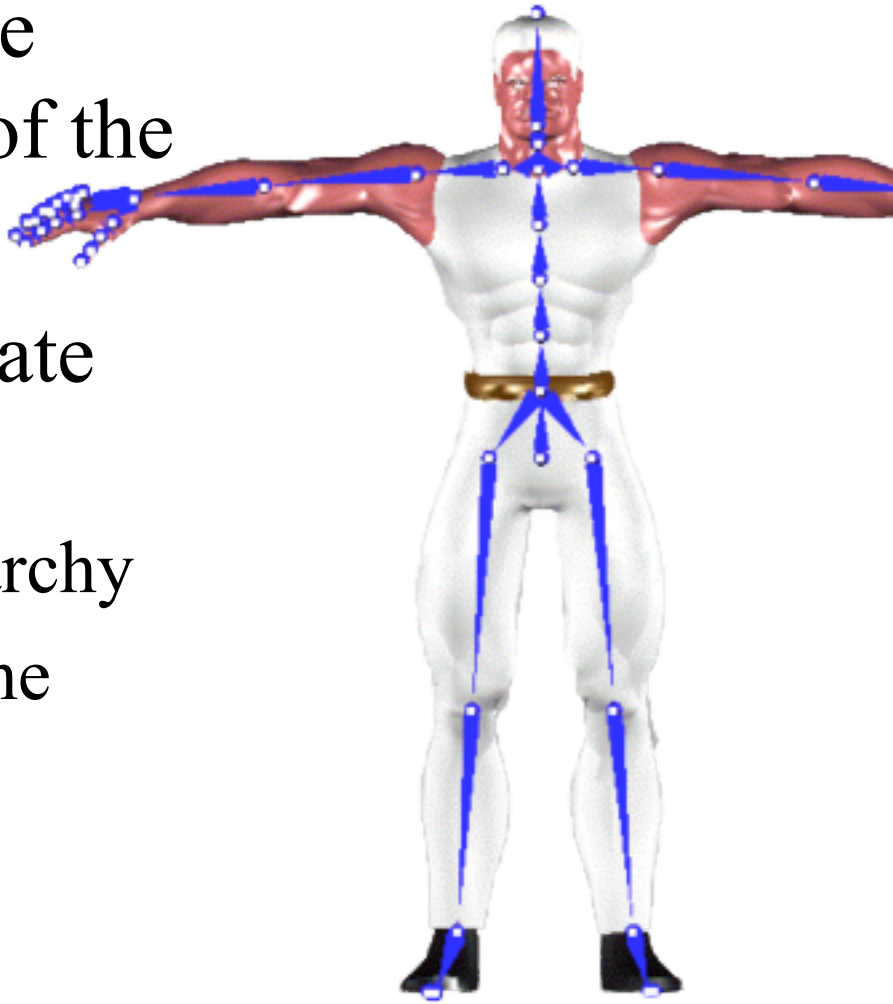
Coordinate Systems

- Undeformed vertices p_i are given in the object space of the skin (global)
- B_j – local bone coordinate -> global coordinate
- T_j is in local bone coordinate system
  - according to skeleton hierarchy



Coordinate Systems

- Undeformed vertices p_i are given in the object space of the skin (global)
- T_j is in local bone coordinate system
  - according to skeleton hierarchy
  - What is T_j? It maps from the local coordinate system of bone j to world space.



Bind Pose cont'd

- When we animate the model, the bone transformations T_j change.

okino.com

Bind Pose cont'd

- When we animate the model, the bone transformations T_j change.
  - What is T_j? It maps from the local coordinate system of bone j to world space.
  - again, concatenates hierarchy matrices

Bind Pose cont'd

- When we animate the model, the bone transformations T_j change.
  - What is T_j? It maps from the local coordinate system of bone j to world space.
- To be able to deform p_i according to T_j, we must first express p_i in the local coordinate system of bone j.

Bind Pose cont'd

- T_j - apply on local bone coordinate -> output is world coordinate
- p_i – in world coordinate
- p_{ij} – also in world coordinate

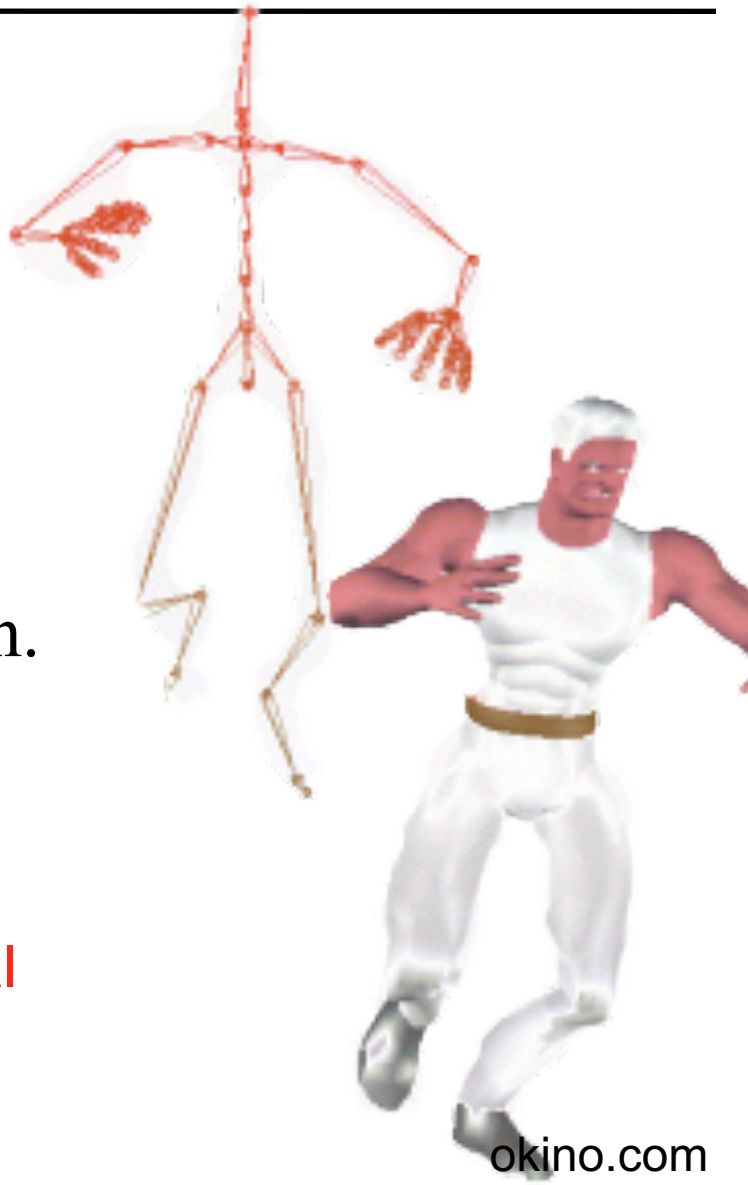
- Bring p_i to the local bone coordinate before applying T_j
  - This is where the bind pose bone transformations B_j come in.

Bind Pose cont'd

- To be able to deform p_i according to T_j, we must first express p_i in the local coordinate system of bone j.
  - This is where the bind pose bone transformations B_j come in.

$$p'_{ij} = T_j B_j^{-1} p_i$$

This maps p_i from bind pose to the local coordinate system of bone j using B^{-1}_j, and then to world space using T_j.



okino.com

Recall frame

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

This maps p_i from bind pose to the local coordinate system of bone j using B^{-1}_j, and then to world space using T_j.

- An object frame has coordinates O in the world (of course O is also our 4x4 matrix)

$$\vec{\mathbf{o}}^t = \vec{\mathbf{w}}^t O$$

- Then we are given coordinates c in the object frame

$$\vec{\mathbf{o}}^t \mathbf{c} = \vec{\mathbf{w}}^t O \mathbf{c}$$

Bind Pose cont'd

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

This maps p_i from bind pose to the local coordinate system of bone j using B^{-1}_j, and then to world space using T_j.

What is T_j 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?

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

This maps p_i from bind pose to the local coordinate system of bone j using B^{-1}_j, and then to world space using T_j.

What is T_j 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!

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

This maps p_i from bind pose to the local coordinate system of bone j using B^{-1}_j, and then to world space using T_j.

What is T_j 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!

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

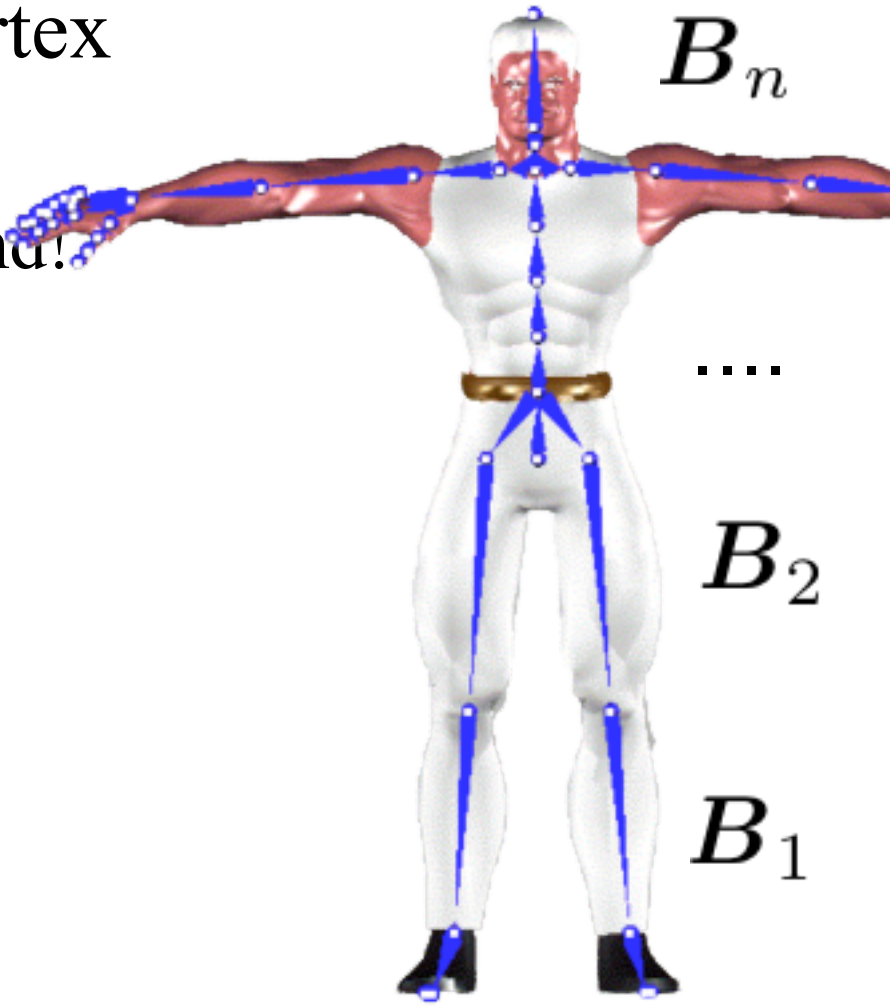
This maps p_i from bind pose to the local coordinate system of bone j using B^{-1}_j, and then to world space using T_j.

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

Questions?

Bind Pose & Weights

- We then figure out the vertex weights w_{ij}.
  - How? Usually paint by hand!
  - We'll look at much cooler methods in a while.



okino.com

Skinning Pseudocode

- Do the usual forward kinematics
  - get a matrix $T_j(t)$ per bone
    (full transformation from local to world)
- For each skin vertex 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
  - get a matrix $T_j(t)$ per bone
    (full transformation from local to world)
- For each skin vertex p_i

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

Do you remember how to treat normals?

Skinning Pseudocode

- Do the usual forward kinematics
  - get a matrix $T_j(t)$ per bone
    (full transformation from local to world)
- For each skin vertex 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
- For each skin vertex p_i

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

- Note that the weights & bind pose vertices are constant over time
  - Only matrices change (small number of them, one per bone)
  - 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...

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

- Rotations are not handled correctly (!!!) (matrices are linearly blended.!)

Indeed... Limitations

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

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

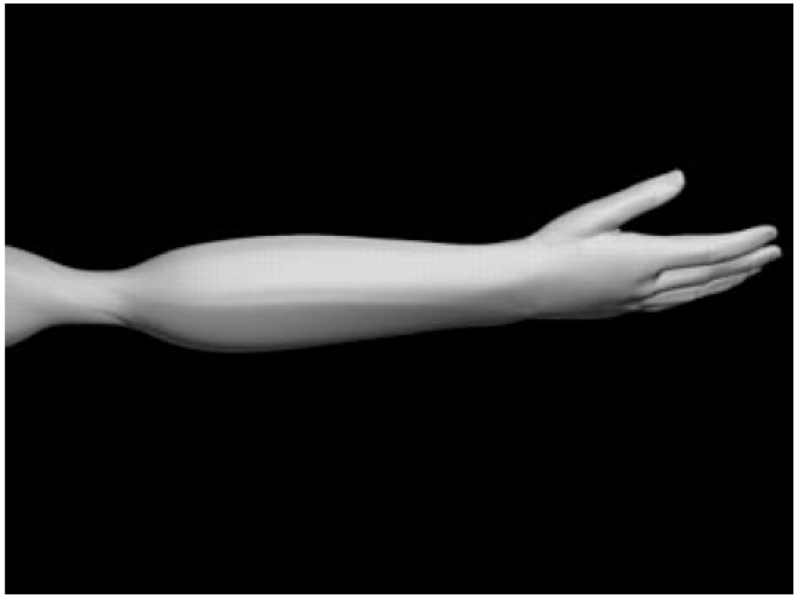
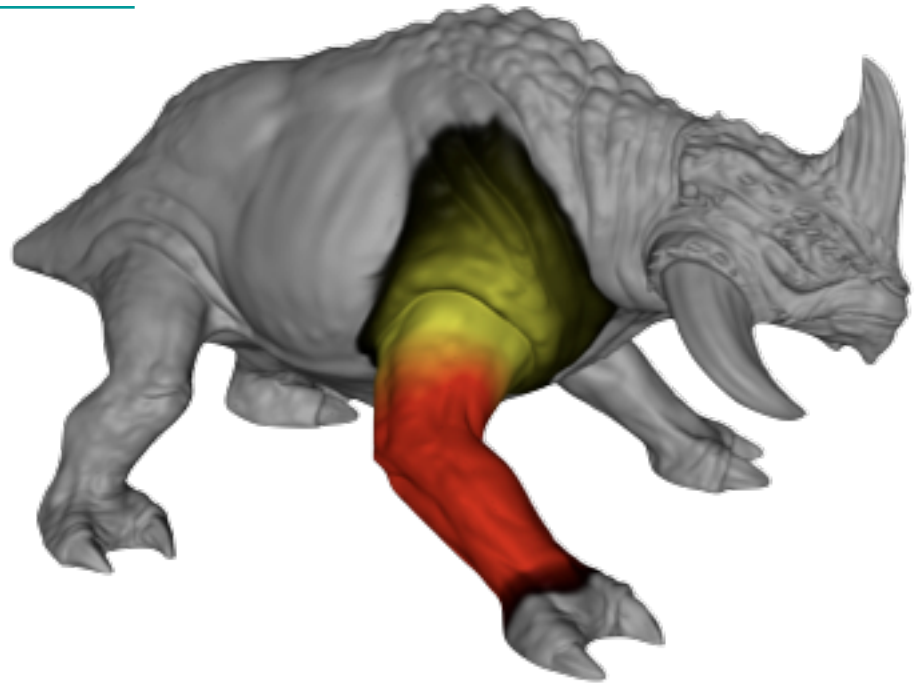


Figure 3: The forearm in the 'twist' pose, as in turning a door handle, computed by SSD. As the twist approaches 180° the arm collapses.

- From: Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation, J. P. Lewis, Matt Cordner, Nickson Fong
- Spherical Blend Skinning: A Real-time Deformation of Articulated Models, Ladislav Kavan and Jiří Žára
- Animating rotation with Quaternion Curves. Ken Shoemake

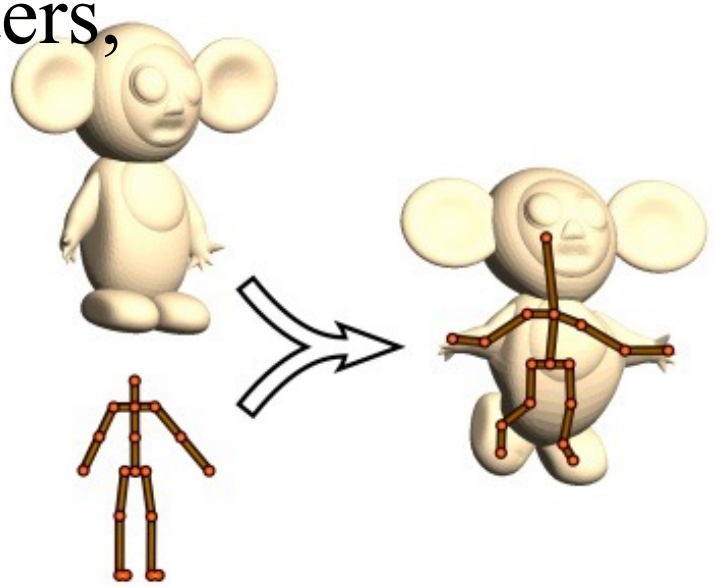
Figuring out the Weights

- 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



First Fully Automatic Rigging Algorithm

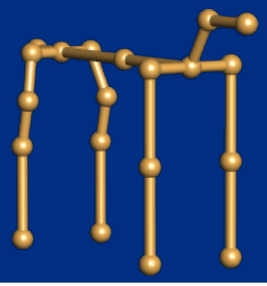
Input

Processing

Output

3D Character

Generic Skeleton



Pinocchio

Rigged Character

Slide from Ilya Baran

Pinocchio

Automatic Rigging and Animation of 3D Characters

Ilya Baran and Jovan Popović

SIGGRAPH 2007

papers_0030

The Other Direction

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.*

Skinning Mesh Animations

Doug L. James
Christopher D. Twigg

Carnegie Mellon University

Additional materials

- 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.