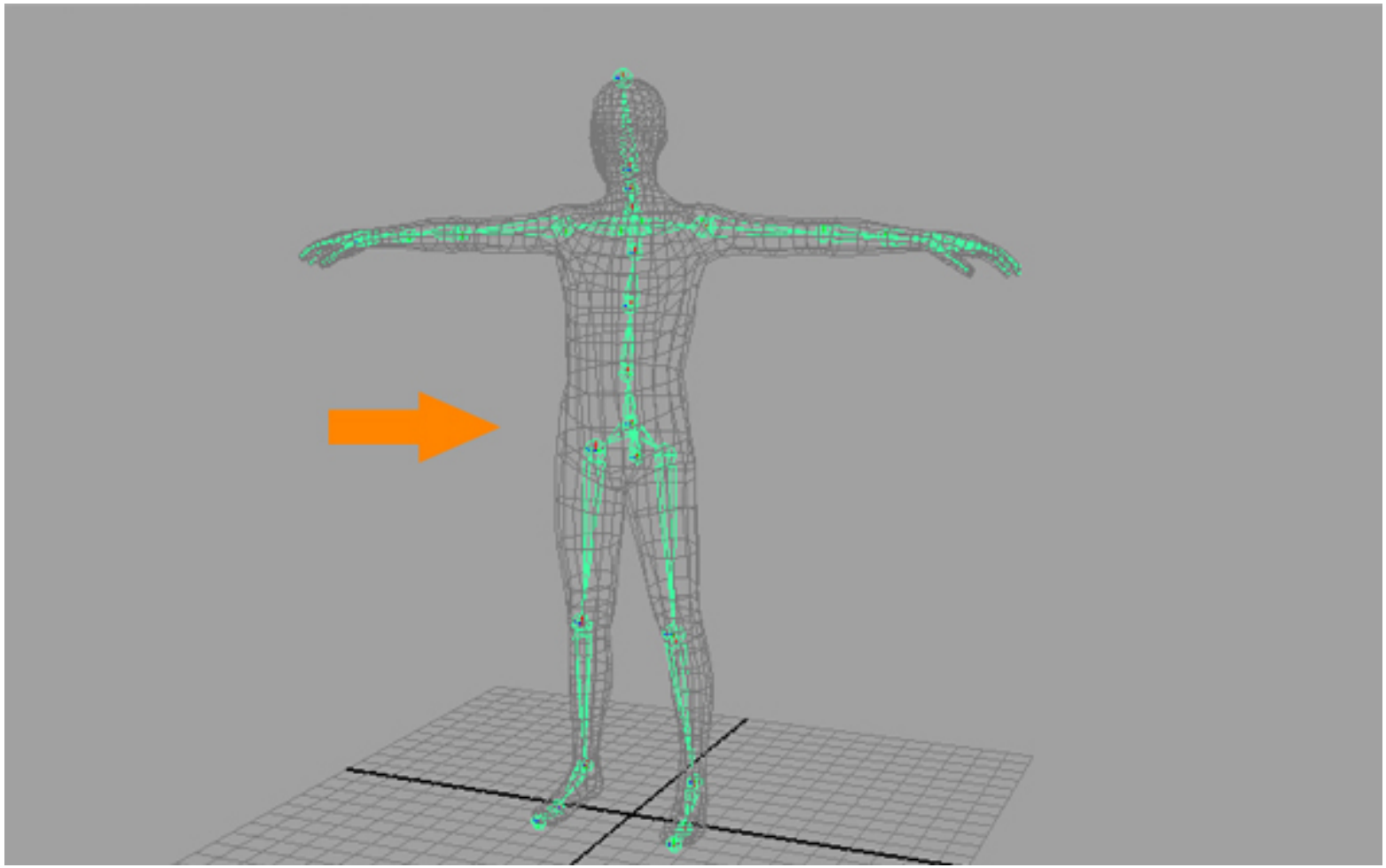
Lecture 3: Skinning

Computer Animation

Skinning



Rigid Parts

- Robots and mechanical creatures can usually be rendered with rigid parts and don't require a smooth skin
- To render rigid parts, each part is transformed by its joint matrix independently
- In this situation, every vertex of the character's geometry is transformed by exactly one matrix

$$\mathbf{v}' = \mathbf{v} \cdot \mathbf{W}$$

where v is defined in joint's local space

Skeleton Forward Kinematics

- Every joint computes a local matrix based on its DOFs and any other constants necessary (joint offsets…)

$$\mathbf{L} = \mathbf{L}_{jnt}\left(\phi_1, \phi_2, \ldots, \phi_N\right)$$

- To find the joint's world matrix, we compute the dot product of the local matrix with the parent's world matrix

$$\mathbf{W} = \mathbf{L} \cdot \mathbf{W}_{parent}$$

- Normally, we would do this in a depth-first order starting from the root, so that we can be sure that the parent's world matrix is available when its needed

Simple Skin

- A simple improvement for low-medium quality characters is to rigidly bind a skin to the skeleton. This means that every vertex of the continuous skin mesh is attached to a joint.

- In this method, as with rigid parts, every vertex is transformed exactly once and should therefore have similar performance to rendering with rigid parts.

Smooth Skin

- With the smooth skin algorithm, a vertex can be attached to more than one joint with adjustable weights that control how much each joint affects it.
- Vertices rarely need to be attached to more than three joints.
- Each vertex is transformed a few times and the results are blended.
- The smooth skin algorithm has many other names: **blended skin**, **skeletal subspace deformation (SSD)**, **multi-matrix skin**, **matrix palette skinning**…

Smooth Skin Algorithm

- The deformed vertex position is a weighted average:

$$\mathbf{v}' = w_1\left(\mathbf{v}\cdot\mathbf{M}_1\right) + w_2\left(\mathbf{v}\cdot\mathbf{M}_2\right) + \ldots w_N\left(\mathbf{v}\cdot\mathbf{M}_N\right)$$

or

$$\mathbf{v}' = \sum w_i\left(\mathbf{v}\cdot\mathbf{M}_i\right)$$

where

$$\sum w_i = 1$$

Rigging Process

- To rig a skinned character, one must have a geometric skin mesh and a skeleton
- Usually, the skin is built in a relatively neutral pose, often in a comfortable standing pose
- The skeleton, however, might be built in more of a *zero pose* where the joints DOFs are assumed to be 0, causing a very stiff, straight pose
- To attach the skin to the skeleton, the skeleton must first be posed into a ***binding pose***
- Once this is done, the vertices can be assigned to joints with appropriate weights

Binding Matrices

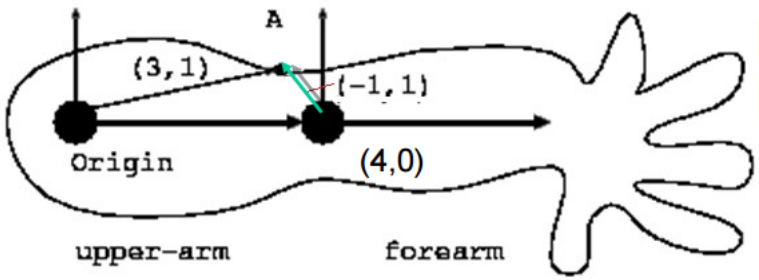
- With rigid parts or simple skin, **v** can be defined local to the joint that transforms it
- With smooth skin, several joints transform a vertex, but it can't be defined local to all of them
- Instead, we must first transform it to be local to the joint that will then transform it to the world
- To do this, we use a binding matrix **B** for each joint that defines where the joint was when the skin was attached and premultiply its inverse with the world matrix:

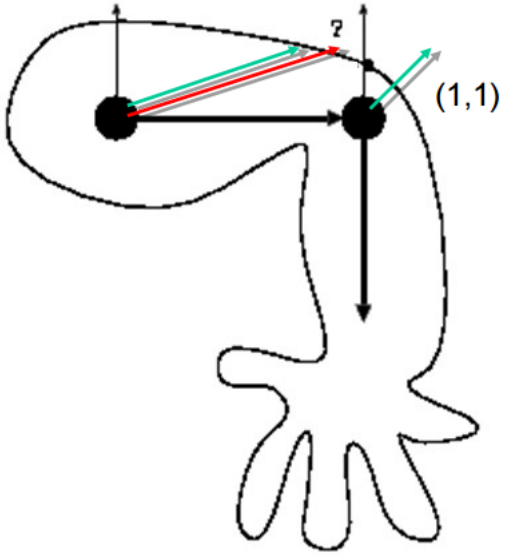
$$\mathbf{M}_i = \mathbf{B}_i^{-1} \cdot \mathbf{W}_i$$

Overall skinning algorithm

- **Step 1:** Fit the skeleton into the mesh and find the matrix that defines where the joint was when the mesh was attached to the skeleton (Binding matrix)
- **Step 2:** Convert the position of vertices to the local coordinates of each bone by inverting the binding matrix
- **Step 3:** Move the joints to the new pose, compute the global position of the vertices using the new local-to-global matrix
- **Step 4:** Blend the results between bones

Example

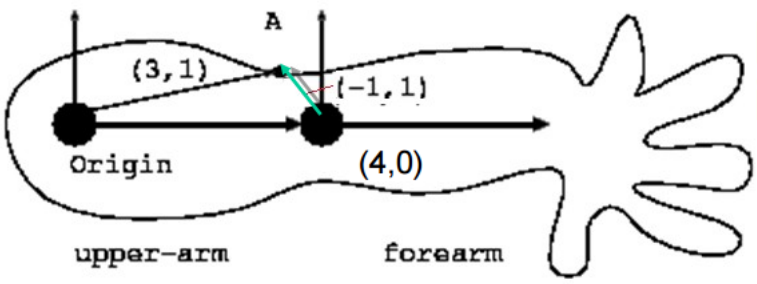


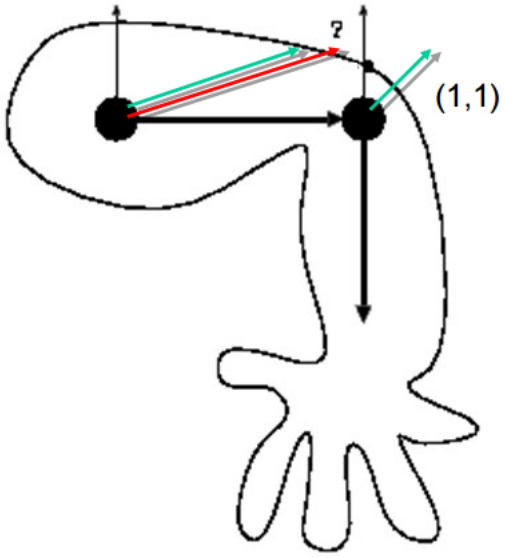


- What is the position of point A after the elbow is bent 90 degrees?
- Assuming the weight for the upper arm is 0.8 and forearm is 0.2

http://homepages.inf.ed.ac.uk/tkomura/cav/presentation4_2016.pdf

Example

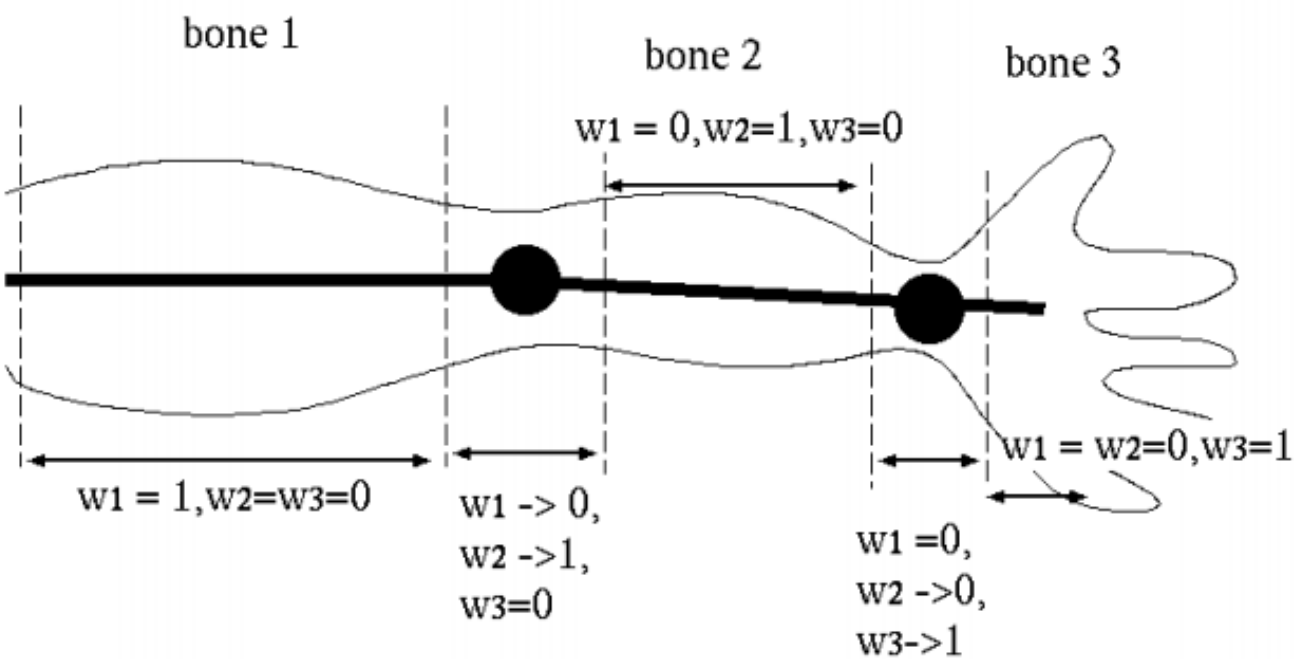




- Assuming it is a point of the upper arm, the position is (3,1)
- Assuming it is a point of the forearm, it is (5,1)
- Adding the weights
- 0.8*(3,1)+0.2*(5,1) = (3.4, 1)

How to decide the weights?

- Decide the mapping of the vertex to the bone

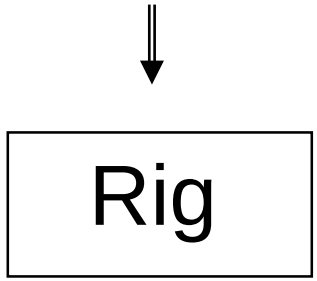


http://homepages.inf.ed.ac.uk/tkomura/cav/presentation4_2016.pdf

Rig Data Flow

- Input DOFs
- Rigging system (skeleton, skin…)
- Output renderable mesh (vertices, normals…)

$$\mathbf{\Phi} = \begin{bmatrix} \phi_1 & \phi_2 & \dots & \phi_N \end{bmatrix}$$



$$\mathbf{v}', \mathbf{n}'$$

Normals

- To compute shading, we need to transform the normals to world space also
- Because the normal is a direction vector, we don't want it to get the translation from the matrix, so we only need to multiply the normal by the upper 3x3 portion of the matrix
- For a normal bound to only one joint: $\mathbf{n}' = \mathbf{n} \cdot \mathbf{W}$
- For smooth skin, we must blend the normal as with the positions, but the normal must then be renormalized:

$$\mathbf{n}' = \frac{\sum w_i \left(\mathbf{n} \cdot \mathbf{M}_i \right)}{\left| \sum w_i \left(\mathbf{n} \cdot \mathbf{M}_i \right) \right|}$$

Algorithm Overview

Skin::Update() (view independent processing)

- Compute skinning matrix for each joint: $\mathbf{M}=\mathbf{B}^{-1}\cdot\mathbf{W}$ (you can precompute and store $\mathbf{B}^{-1}$ instead of $\mathbf{B}$)
- Loop through vertices and compute blended position & normal

Skin::Draw() (view dependent processing)

- Loop through triangles and draw using world space positions & normal

Smooth Skin Algorithm

- The deformed vertex position is a weighted average over all of the joints that the vertex is attached to:

$$\mathbf{v}' = \sum w_i \mathbf{v} \cdot \mathbf{B}_i^{-1} \cdot \mathbf{W}_i$$

- **W** is a joint's world matrix and **B** is a joint's binding matrix that describes where it's world matrix was when it was attached to the skin model (at skin creation time)
- Each joint transforms the vertex as if it were rigidly attached, and then those results are blended based on user specified weights
- All of the weights must add up to 1: $\sum w_i = 1$
- Blending normals is essentially the same, except we transform them as direction vectors (x,y,z,0) and then renormalize the results

$$\mathbf{n}^* = \sum w_i \mathbf{n} \cdot \mathbf{B}_i^{-1} \cdot \mathbf{W}_i, \qquad \mathbf{n}' = \frac{\mathbf{n}^*}{\left|\mathbf{n}^*\right|}$$

Skinning Equations

- Skeleton

$$\mathbf{L} = \mathbf{L}_{jnt}\left(\phi_1, \phi_2, \ldots, \phi_N\right)$$

$$\mathbf{W} = \mathbf{L} \cdot \mathbf{W}_{parent}$$

- Skinning

$$\mathbf{v}' = \sum w_i \mathbf{v} \cdot \mathbf{B}_i^{-1} \cdot \mathbf{W}_i$$

$$\mathbf{n}^* = \sum w_i \mathbf{n} \cdot \mathbf{B}_i^{-1} \cdot \mathbf{W}_i$$

$$\mathbf{n}' = \frac{\mathbf{n}^*}{\left|\mathbf{n}^*\right|}$$

Skin Binding

- Attaching a skin to a skeleton is not a trivial problem and usually requires automated tools combined with interactive tuning
- Binding algorithms typically involve heuristic approaches
- Some general approaches:
  - Point-to-line mapping
  - Containment
  - Delaunay tetrahedralization

Point-to-Line Mapping

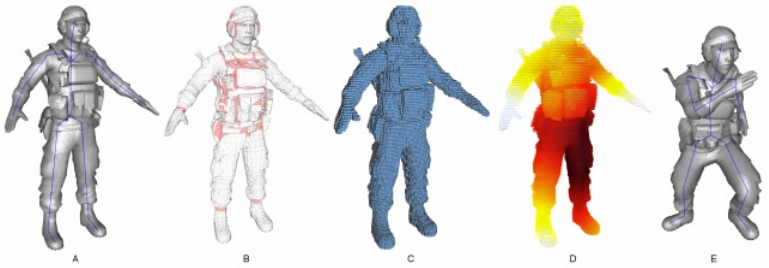
- A simple way to attach a skin is to treat each bone as one or more line segments and attach each vertex to the nearest line segment
- A bone is made from line segments connecting the joint pivot to the pivots of each child

Containment Binding

- With containment binding algorithms, the user manually approximates the body with volume primitives for each bone (cylinders, ellipsoids, spheres…)
- The algorithm then tests each vertex against the volumes and attaches it to the best fitting bone

Delaunay Tetrahedralization

- Builds a tetrahedralization of the volume within the skin using computational geometry
- The tetrahedra connect all of the skin verts and skeletal pivots in a relatively clean 'Delaunay' fashion
- The connectivity of the mesh can then be analyzed to determine the best attachment for each vertex

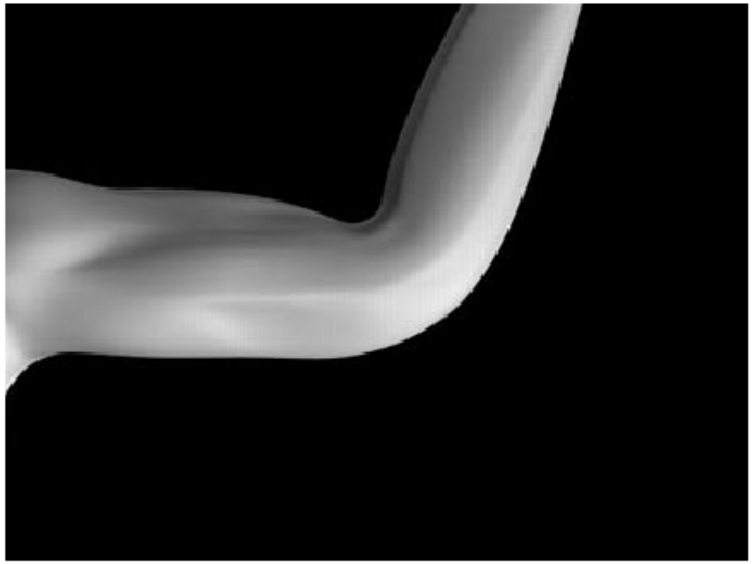


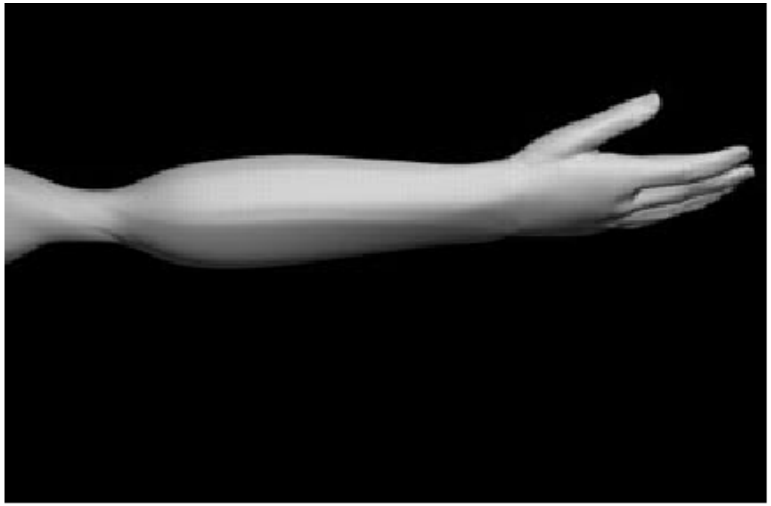
Geodesic Voxel Binding for Production Character Meshes, SCA 2013

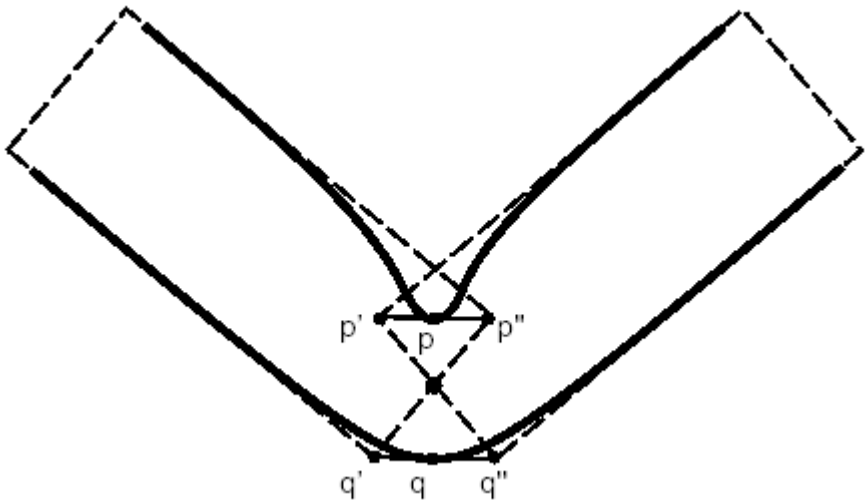
Limitations of Smooth Skin

- Smooth skin is very simple and quite fast, but its quality is limited
- The main problems are:
  - Joints tend to collapse as they bend more
  - Very difficult to get specific control
  - Unintuitive and difficult to edit
- Still, it is built in to most 3D animation packages
- If nothing else, it is a good baseline upon which more complex schemes can be built

Limitations of Smooth Skin



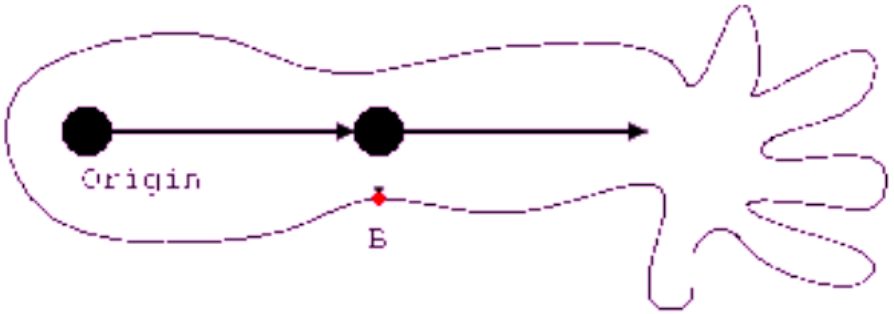


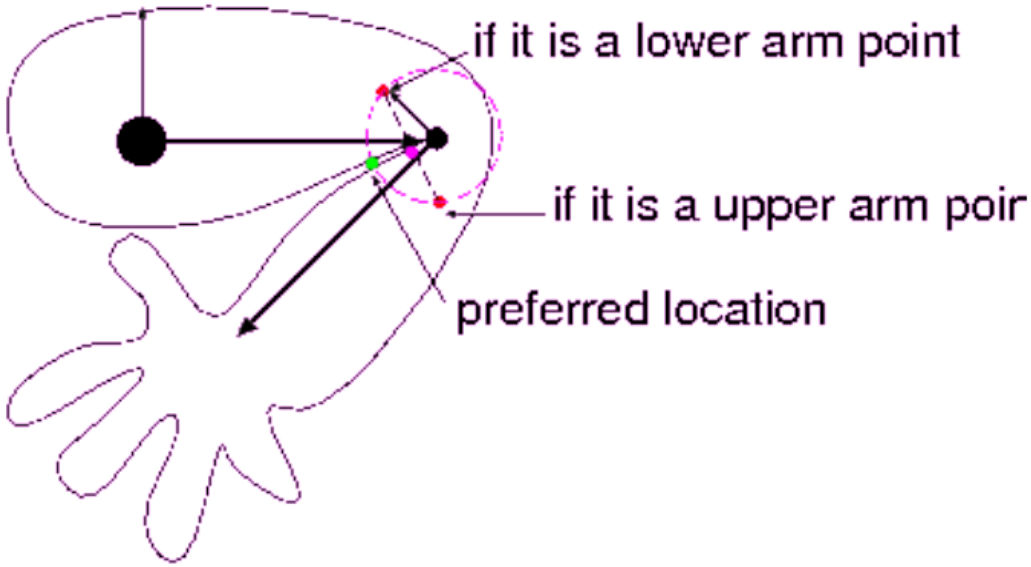


Skin collapse (bending) and twisting (candy wrapper) effect

Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation, Lewis, Cordner and Fong, SIGGRAPH 2000

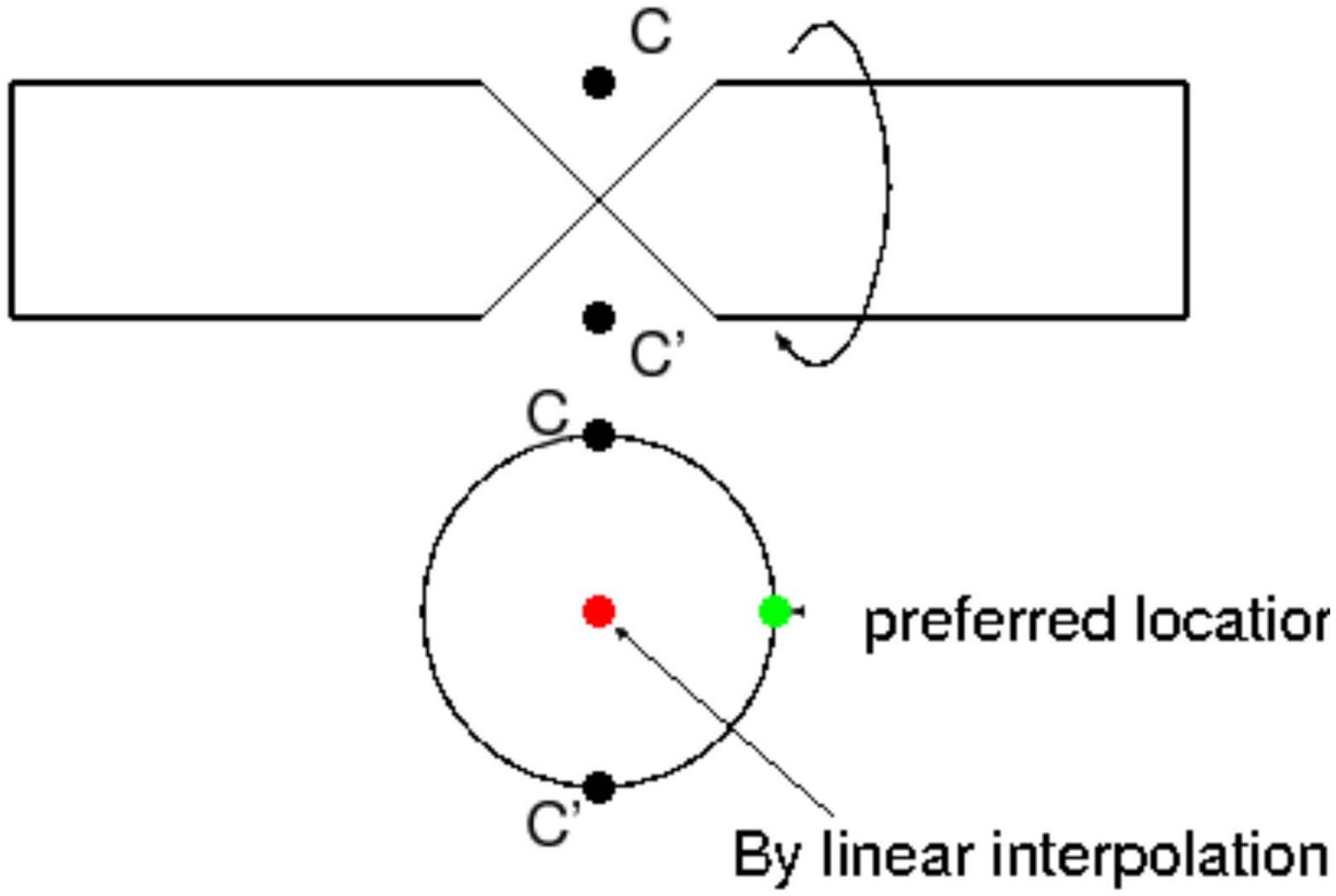
Why does it happen?





http://homepages.inf.ed.ac.uk/tkomura/cav/presentation4_2016.pdf

Why does it happen?



Bone Links

- To help with the collapsing joint problem, one option is to use bone links
- Bone links are extra joints inserted in the skeleton to assist with the skinning
- They can be automatically added based on the joint's range of motion. For example, they could be added so as to prevent any joint from rotating more than 60 degrees.
- This is a simple approach used in some real time games, but doesn't go very far in fixing the other problems with smooth skin.

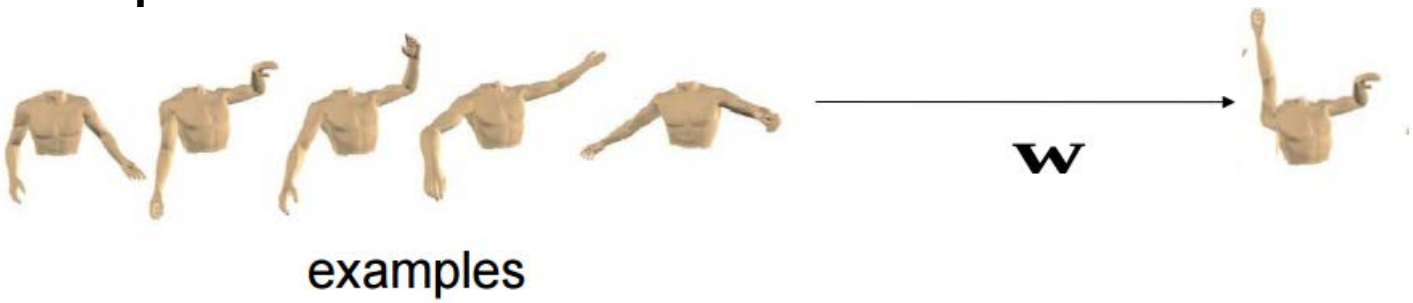
Muscles & Other Effects

- One can add custom effects such as muscle bulges as additional joints
- For example, the bicep could be a translational or scaling joint that smoothly controls some of the vertices in the upper arm. Its motion could be linked to the motion of the elbow rotation.
- With this approach, one can also use skin for muscles, fat bulges, facial expressions, and even simple clothing

Automatic methods

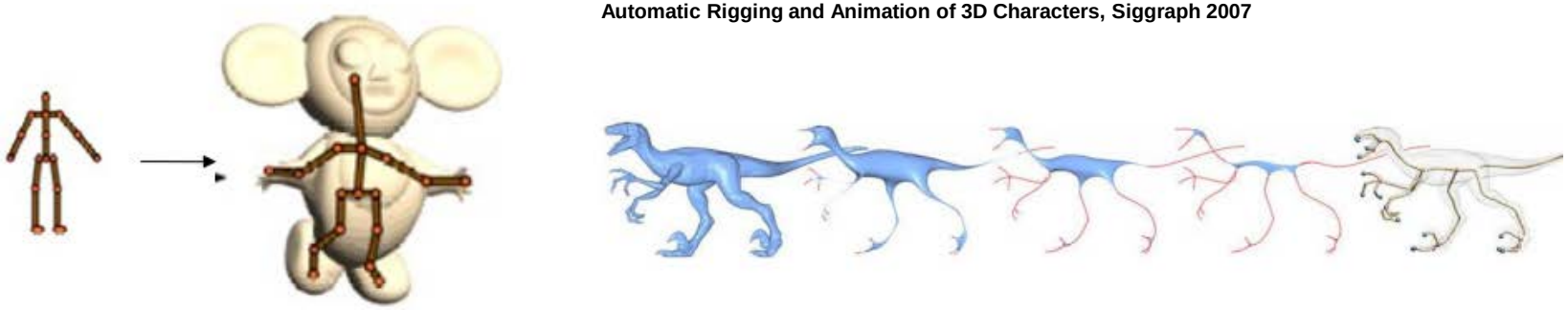
- Automatic computation of the weights from examples

Building efficient, accurate character skins from examples, Siggraph 2003



- Automatic skeleton fitting / extraction

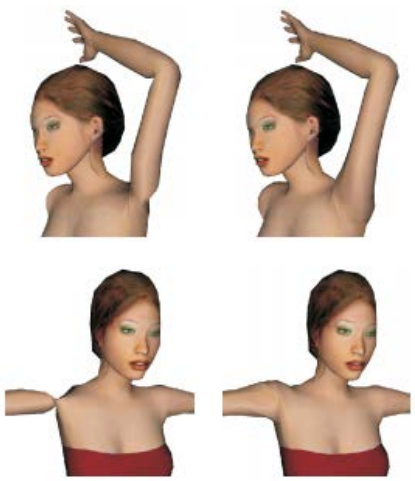
Automatic Rigging and Animation of 3D Characters, Siggraph 2007



http://homepages.inf.ed.ac.uk/tkomura/cav/presentation4_2016.pdf

Dual quaternion blending

- Proposed to overcome the bending and candy wrapper effects
- Currently, found in most 3D modelling tools, e.g. Maya, Blender



Geometric Skinning with Approximate Dual Quaternion Blending, Kavan Collins, Zara and O'Sullivan, Siggraph 2008

Supplementary material

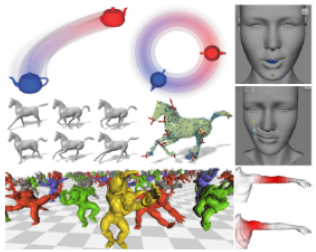
Siggraph course on skinning (2014): http://skinning.org/

Skinning: Real-time Shape Deformation

ACM SIGGRAPH 2014 Course
ACM SIGGRAPH Asia 2014 Invited Course
Symposium on Geometry Processing 2015 Invited Course
International Geometry Summit 2016 Invited Course

Alec Jacobson
Columbia University
Zhigang Deng
University of Houston
Ladislav Kavan
University of Pennsylvania
J.P. Lewis
Victoria University, Weta Digital

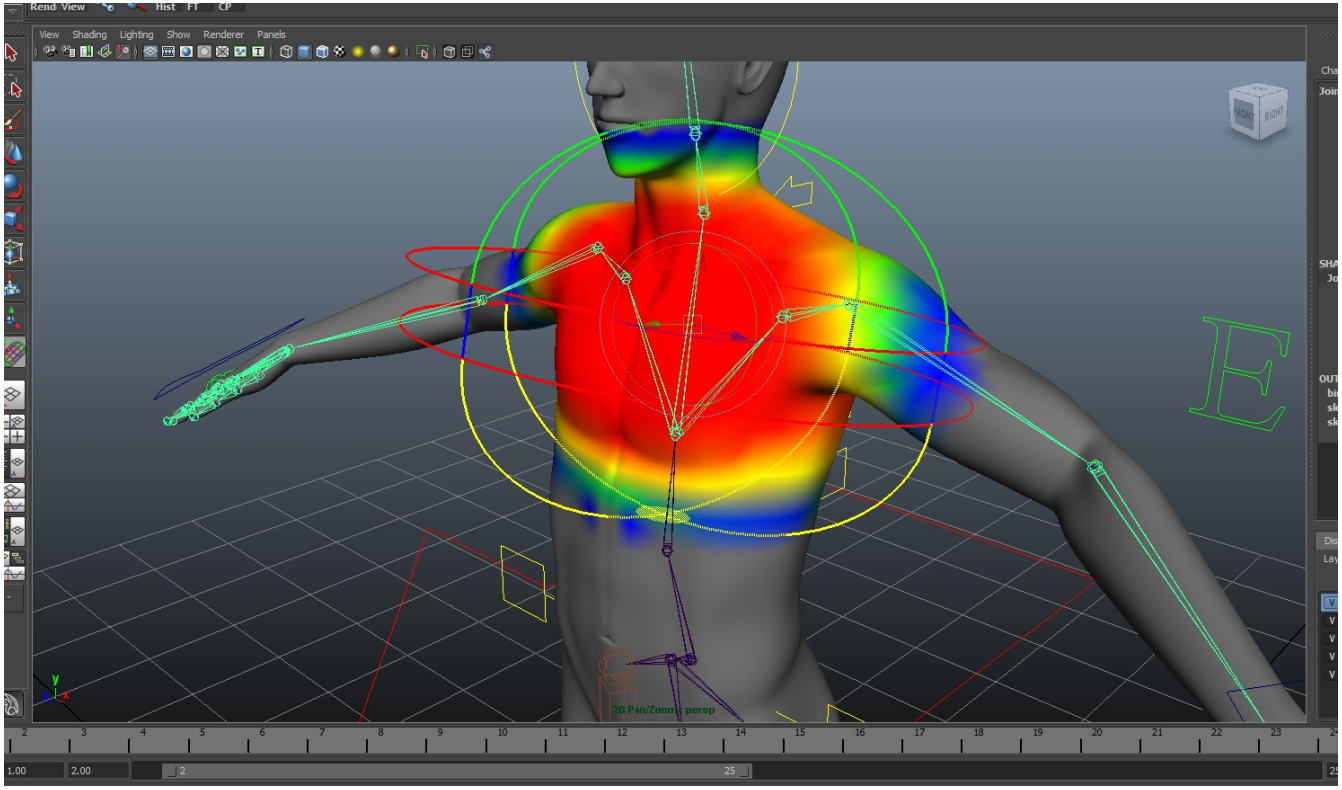
SIGGRAPH Asia lecturer: Yotam Gingold
George Mason University



Course Materials

- Part I: Direct methods (Ladislav Kavan)
  Course notes | Slides
- Part II: Automatic methods (Alec Jacobson)
  Course notes | Course notes (low resolution) | Slides | Slides (167MB .pptx with videos)
- Part III: Example-based methods (JP Lewis)
  Course notes

Painting weights



https://www.youtube.com/watch?v=Ah-Jk7d3oks

Shape Interpolation

- Another extension to the smooth skinning algorithm is to allow the vertices to be modeled at key values along the joints motion
- For an elbow, for example, one could model it straight, then model it fully bent
- These shapes are interpolated local to the bones before the skinning is applied

Shape Interpolation Algorithm

- To compute a blended vertex position:

$$\mathbf{v}' = \mathbf{v}_{base} + \sum \phi_i \cdot \left(\mathbf{v}_i - \mathbf{v}_{base} \right)$$

- The blended position is the base position plus a contribution from each target whose DOF value is greater than o

- To blend the normals, we use a similar equation:

$$\mathbf{n}' = \mathbf{n}_{base} + \sum \phi_i \cdot \left(\mathbf{n}_i - \mathbf{n}_{base} \right)$$

- We won't normalize them now, as that will happen later in the skinning phase

Layered Approach

- We use a simple layered approach
  - Skeleton Kinematics
  - Shape Interpolation
  - Smooth Skinning
- Most character rigging systems are based on some sort of layered system approach combined with general purpose data flow to allow for customization

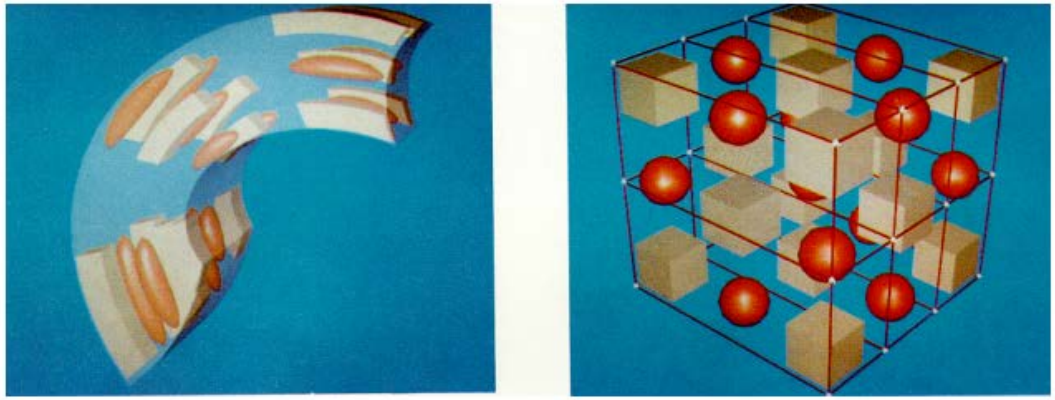
Free-Form Deformations

- FFDs are a class of deformations where a low detail control mesh is used to deform a higher detail skin
- There are a lot of variations on FFDs based on the topology of the control mesh

Thomas W. Sederberg and Scott R. Parry. 1986. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '86)

Lattice FFDs

- The original type of FFD uses a simple regular lattice placed around a region of space
- The lattice is divided up into a regular grid
- When the lattice points are then moved, they describe smooth deformation in their vicinity



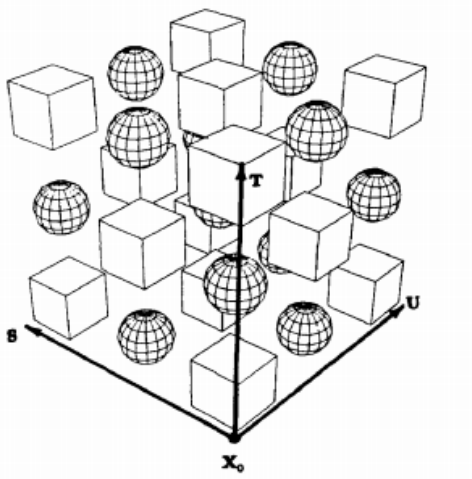
Lattice FFDs

- We start by defining the undeformed lattice space:

$$\mathbf{x}(s,t,u) = \mathbf{x}_0 + s \cdot \mathbf{s}_0 + t \cdot \mathbf{t}_0 + u \cdot \mathbf{u}_0$$

$$0 \le s \le 1 \quad 0 \le t \le 1 \quad 0 \le u \le 1$$

$$\mathbf{M} = \begin{bmatrix} s_x & s_y & s_z & 0 \\ t_x & t_y & t_z & 0 \\ u_x & u_y & u_z & 0 \\ x_{0x} & x_{0y} & x_{0z} & 1 \end{bmatrix}$$



Lattice FFDs

- We then define the number of sections in the 3 lattice dimensions:

  1 <= l,m,n <= 3

- And then set the initial lattice positions: $\mathbf{p}_{ijk}$

$$\mathbf{p}_{ijk} = \mathbf{x}_0 + \frac{i}{l} \cdot \mathbf{s}_0 + \frac{j}{m} \cdot \mathbf{t}_0 + \frac{k}{n} \cdot \mathbf{u}_0$$

Lattice FFDs

- To deform a point x, we first find the (s,t,u) coordinates:

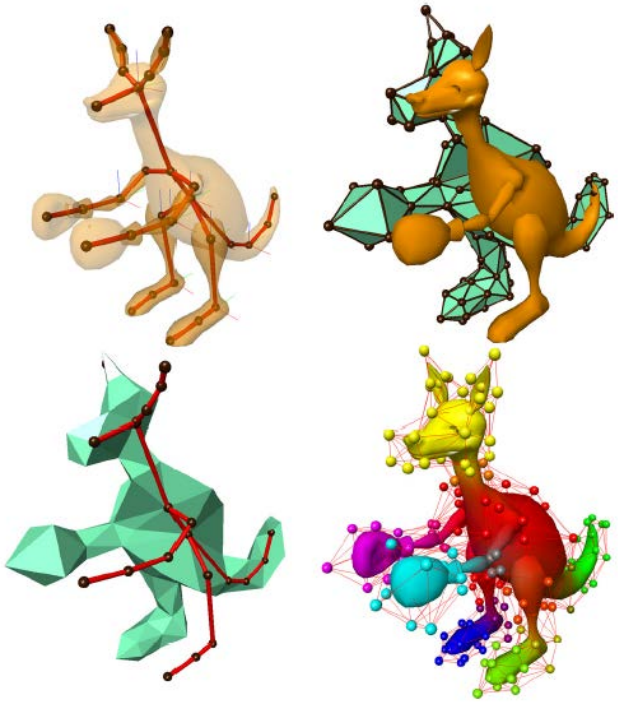
$$\begin{bmatrix} s & t & u & 1 \end{bmatrix} = \mathbf{x} \cdot \mathbf{M}^{-1}$$

- Then deform that into world space (similar to Bezier curves and Bernstein polynomials) :

$$\mathbf{x}_{ffd} = \sum_{i=0}^{l} \binom{l}{i} (1-s)^{l-i} s^{i} \cdot \left(\sum_{j=0}^{m} \binom{m}{j} (1-t)^{m-j} t^{j} \cdot \left(\sum_{k=0}^{n} \binom{n}{k} (1-u)^{n-k} u^{k} \cdot \mathbf{p}_{ijk} \right) \right)$$

Arbitrary Topology FFDs

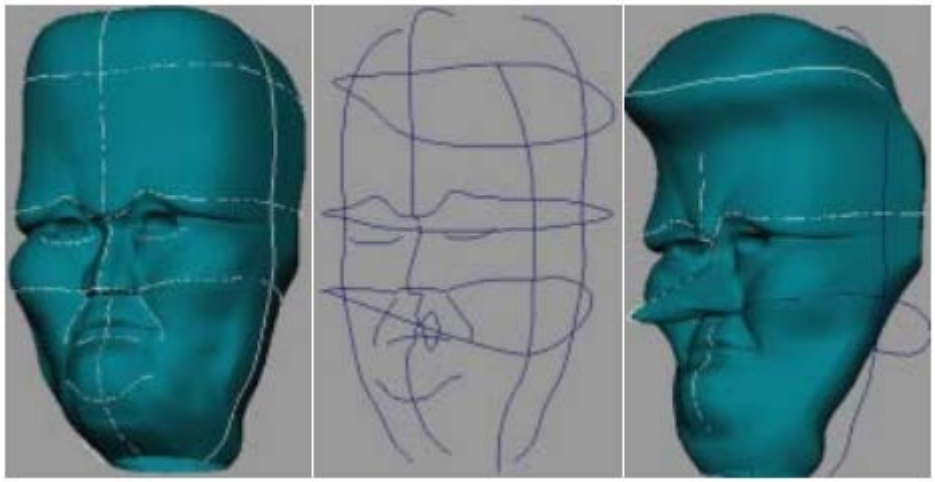
- The concept of FFDs was later extended to allow an arbitrary topology control volume to be used



Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. 2002. Interactive skeleton-driven dynamic deformations. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '02).

Axial Deformations & WIRES

- Another type of deformation allows the user to place lines or curves within a skin
- When the lines or curves are moved, they distort the space around them
- Multiple lines & curves can be placed near each other and will properly interact



Karan Singh and Eugene Fiume. 1998. Wires: a geometric deformation technique. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '98).

Anatomical Modeling

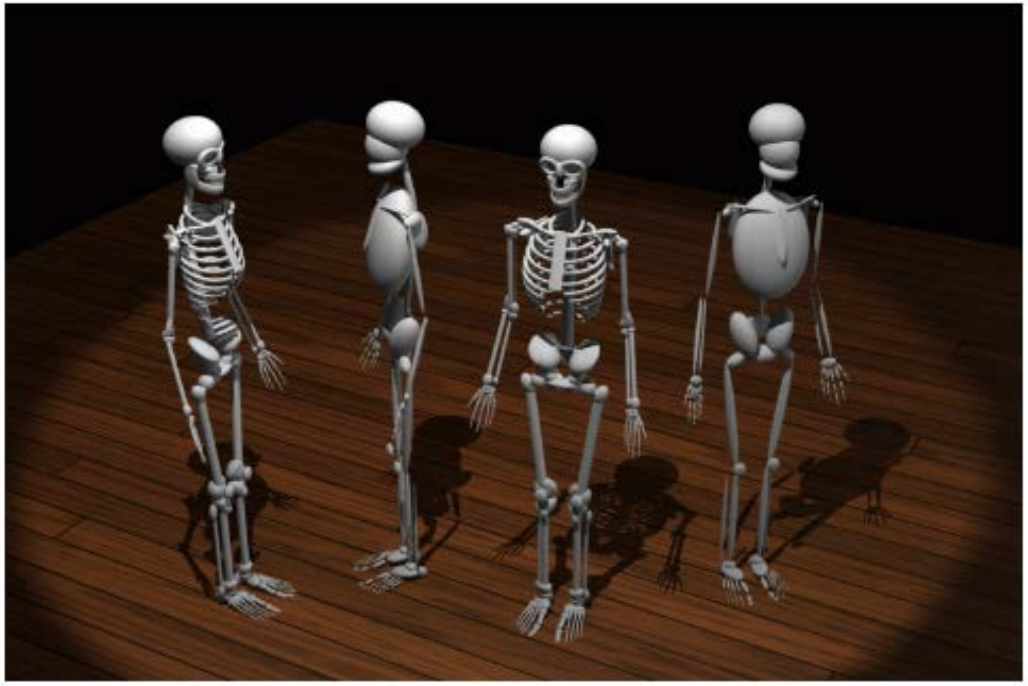
- The motion of the skin is based on the motion of the underlying muscle and bones. Therefore, in an anatomical simulation, the tissue beneath the skin must be accounted for
- One can model the bones, muscle, and skin tissue as deformable bodies and then use physical simulation to compute their motion
- Various approaches exist ranging from simple approximations using basic primitives to detailed anatomical simulations

Skin & Muscle Simulation

- Bones are essentially rigid
- Muscles occupy almost all of the space between bone & skin
- Although they can change shape, muscles have essentially constant volume
- The rest of the space between the bone & skin is filled with fat & connective tissues
- Skin is connected to fatty tissue and can usually slide freely over muscle
- Skin is anisotropic as wrinkles tend to have specific orientations

Simple Anatomical Models

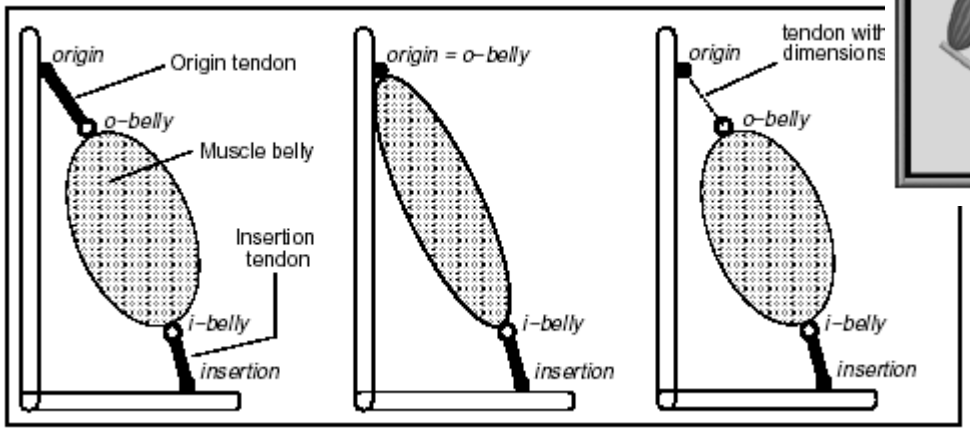
- Some simplified anatomical models use ellipsoids to model bones and muscles

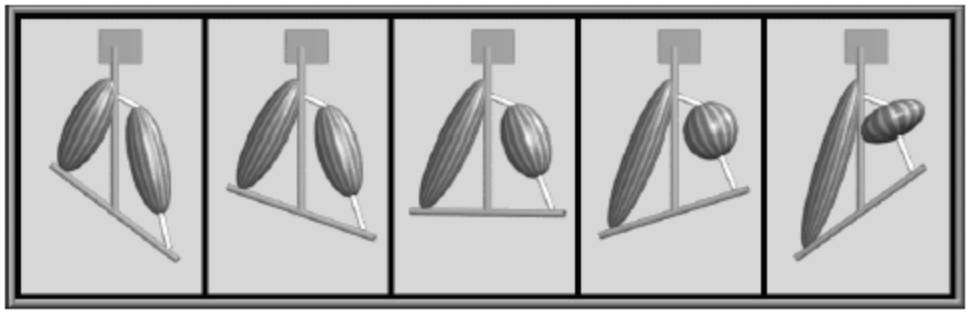


Ferdi Scheepers, Richard E. Parent, Wayne E. Carlson, and Stephen F. May. 1997. Anatomy-based modeling of the human musculature. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '97)

Simple Anatomical Models

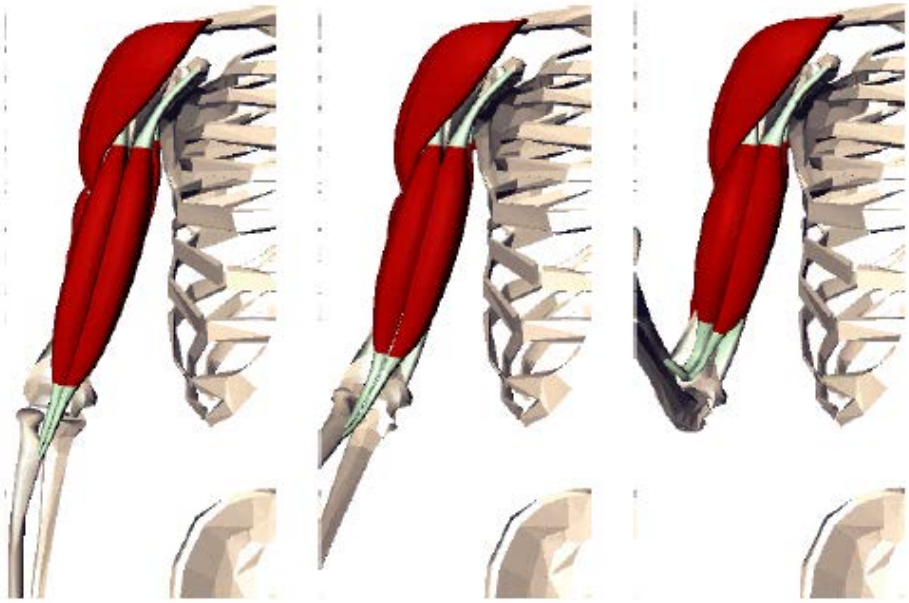
- Muscles are attached to bones, sometimes with tendons as well
- The muscles contract in a volume preserving way, thus getting wider as they get shorter

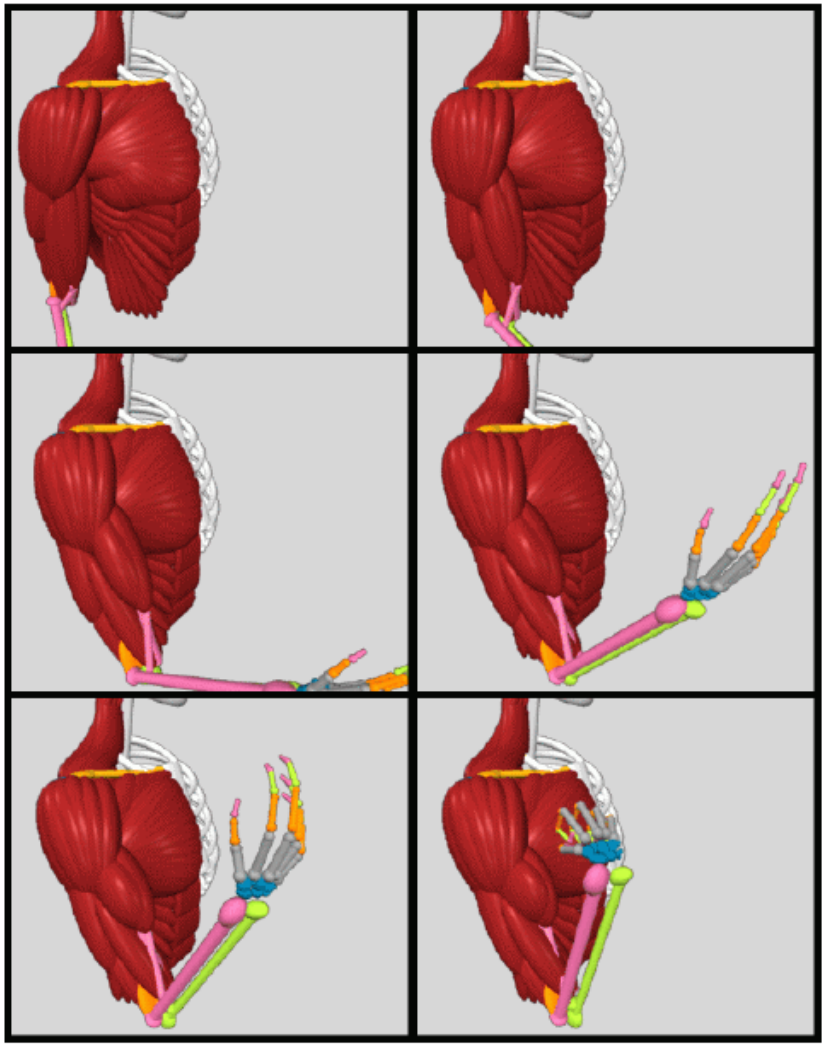




Simple Anatomical Models

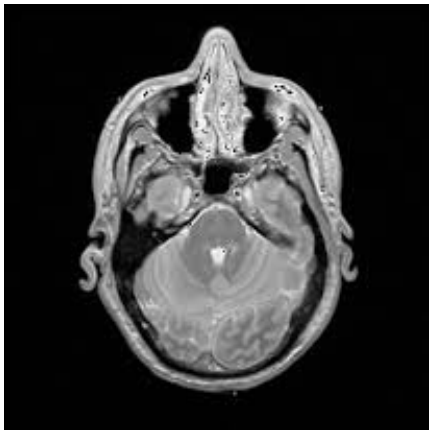
- Complex musculature can be built up from lots of simple primitives





Detailed Anatomical Models

- One can also do detailed simulations that accurately model bone & muscle geometry, as well as physical properties



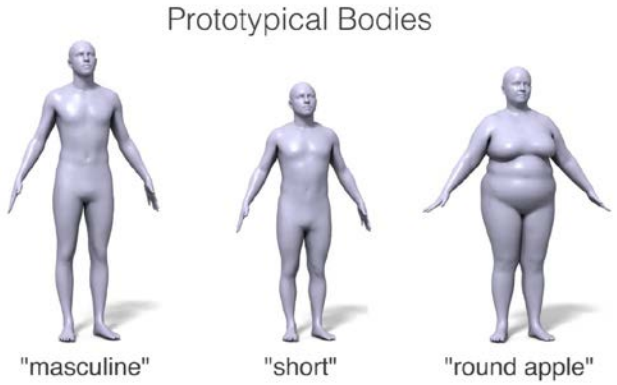
- e.g. Visible Human Project

Body Scanning

- Data input has become an important issue for the various types of data used in computer graphics
- Examples:
  - Geometry: Laser scanners
  - Motion: Optical motion capture
  - Materials: Gonioreflectometer
  - Faces: Computer vision
- Recently, people have been researching techniques for directly scanning human bodies and skin deformations

Body Scanning

- Practical approaches tend to use either a 3D model scanner (like a laser) or a 2D image based approach (computer vision)
- The skin is scanned at various key poses and some sort of 3D model is constructed
- Some techniques attempt to fit this back onto a standardized mesh, so that all poses share the same topology. This is difficult, but it makes the interpolation process much easier.
- Other techniques interpolate between different topologies. This is difficult also.



http://graphics.cs.cmu.edu/projects/muscle/
http://dyna.is.tue.mpg.de/

- Next lecture “Interpolation and Blending” on 9 May