Basics of Computer Animation
Skinning/Enveloping

Many slides courtesy of Jovan Popovic, Ronen Barzel, and Jaakko Lehtinen

Courtesy of Blender Foundation. License CC-BY. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.
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)

Image courtesy of Garrett Albright on Wikimedia Commons. License: CC-BY-SA. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.
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 of the standard animation techniques:

- See also The Illusion of Life: Disney Animation, by Frank Thomas and Ollie Johnston.
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

© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see [http://ocw.mit.edu/help/faq-fair-use/](http://ocw.mit.edu/help/faq-fair-use/).

Example: Timing

• Timing affects weight:
  – Light object move quickly
  – Heavier objects move slower

• Timing completely changes the interpretation of the motion.
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
Another Example

• Physically-Based Character Animation
  – Specify keyframes, solve for physically valid motion that interpolates them by “spacetime optimization”

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?
• 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.
Example of Higher-Level Controls

- Ken Perlin’s facial expression applet [http://mrl.nyu.edu/~perlin/experiments/facedemo/](http://mrl.nyu.edu/~perlin/experiments/facedemo/)

- Lower-level controls are mapped to semantically meaningful higher-level ones – “Frown/smile” etc.
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.

Courtesy Robert C. Duvall, Duke University. License CC BY-NC-SA.
Skinning Characters

• Embed a skeleton into a detailed character mesh

Courtesy of Blender Foundation. License CC-BY. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use.
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

Courtesy of Blender Foundation. License CC-BY. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use.
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

Courtesy Robert C. Duvall, Duke University. License CC BY-NC-SA.
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”)
Questions?
Plan

• Types of Animation (overview)
  – Keyframing
  – Procedural
  – Physically-based

• Animation Controls

• Character Animation using skinning/enveloping
Skinning/Enveloping

Courtesy of Blender Foundation. License CC-BY. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use.
Skinning

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

Courtesy of Blender Foundation. License CC-BY. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use.
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

This image is in the public domain. Source: Wikimedia Commons.
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
Example

Colored triangles are attached to 1 bone

Black triangles are attached to more than 1

Note how they are near joints
Example

Colored triangles are attached to 1 bone

Black triangles are attached to more than 1

Note how they are near joints

© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.
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 $j$”
  – $wij = 1$ 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 8: Top: heat equilibrium for two bones. Bottom: the result of rotating the right bone with the heat-based attachment.
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 $wij$ for each vertex $p_i$ for each bone $B_j$.
  – “How much vertex $i$ should move with bone $j$”
  – $wij = 1$ 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?
• 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.

$$p'_{ij} = T_j p_i$$

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

$p'_{ij}$ 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

Rest ("bind") pose

Bone 1: $T_1$

Bone 2: $T_2$

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

Rest ("bind") pose

Bone 1: $T_1$

Bone 2: $T_2$

After rotations

"Skin"

Bone 1: $T'1$

Bone 2: $T'2$

- 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

Rest ("bind") pose

- 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?

After rotations
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

GNU Free Documentation License. Some rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.
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.
Coordinate Systems

• Undeformed vertices $p_i$ are given in the object space of the skin
• $T_j$ is in local bone coordinate system – according to skeleton hierarchy
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 node j up to the root
Bind Pose cont’d

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

- 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.

$$p_{ij} = T_j B^{-1}_j p_i$$

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 $T_j$. 
Bind Pose cont’d

\[ p'_{ij} = T_j B_j^{-1} p_i \]

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

What is \( T_j B^{-1} \)? It is the relative change between the bone transformations between the current and the bind pose.
Bind Pose cont’d

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

What is $T_j B_j^{-1}$? It is the relative change between the bone transformations between the current and the bind pose.
Bind Pose cont’d

\[ p'_{ij} = T_j B^{-1}_j p_i \]

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

\[ p'_i = \sum_j w_{ij} T_j(t) B_j^{-1} p_i \]
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 $p_i$

$$p'_i = \sum_j w_{ij} T_j(t) B_j^{-1} p_i$$

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 $p_i$

\[ p'_i = \sum_j w_{ij} T_j(t) B_j^{-1} p_i \]

• Inverse transpose for normals!

\[ n'_i = \left( \sum_j w_{ij} T_j(t) B_j^{-1} \right)^{-T} n_i \]
Skinning Pseudocode

- Do the usual forward kinematics
- For each skin vertex $p_i$
  $$p'_i = \sum_{j} w_{ij} T_j(t) B_j^{-1} p_i$$
- 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)
Hmmmh...

• This is what we do to get deformed positions

\[ p_i' = \sum_j w_{i,j} T_j(t) B_j^{-1} p_i \]
Hmmm...

- This is what we do to get deformed positions

\[ p'_i = \sum_j w_{ij} T_j(t) B_j^{-1} p_i \]

- But wait...

\[ p'_i = \left( \sum_j w_{ij} T_j(t) B_j^{-1} \right) p_i \]
• This is what we do to get deformed positions

\[ p'_i = \sum_j w_{ij} T_j(t) B_j^{-1} p_i \]

• But wait...

\[ p'_i = \left( \sum_j w_{ij} T_j(t) B_j^{-1} \right) p_i \]

• Rotations are not handled correctly (!!!)
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.

© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/. 

- From: Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation, J. P. Lewis, Matt Cordner, Nickson Fong
Real-time enveloping with rotational regression
Wang, Pulli, Popovic
We learn a fast model from exported examples.
Figuring out the Weights

- Usual approach: Paint them on the skin.
- Can also find them by optimization from example poses and deformed skins.

Figure 8: Top: heat equilibrium for two bones. Bottom: the result of rotating the right bone with the heat-based attachment
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
Super Cool: Automatic Rigging

From Automatic Rigging and Animation of 3D Characters by Baran and Popovic, used with permission from ACM, Inc.

© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.
The Other Direction

Skinning Mesh Animations

Doug L. James       Christopher D. Twigg
Carnegie Mellon University

Figure 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.

© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

From Skinning Mesh Animations.
That’s All for Today!

• Further reading

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