Basics of Computer Animation

Skinning/Enveloping

Jaakko Lehtinen, MIT CSAIL

Many slides courtesy of Jovan Popovic, Ronen Barzel, and Frédo Durand
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)
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:
  - See also

![Image](image.png)
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

- Timing completely changes the interpretation of the motion.
See also


The Pixar process

There is a scene in Toy Story 2 when the old man repairing Woody tells the impatient toy collector Al, “You can’t rush art.” This is especially true at Pixar, where films go through four stages: development, creating the storyline; pre-production, addressing technical challenges; production, making the film; and post-production, “polishing” the final product. Use the lever to the left to learn about the specific steps of each stage.
Computer Animation

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

• Two very different contexts:
  – Production (offline)
  – Interactive (e.g. games, simulators)
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
To first order, Computer Animation is achieved by changing these mappings as functions of time.
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: 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 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”

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.
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.
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
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")
  – A kind of inverse kinematics!
Another Example

• Microsoft’s Project Natal tracks the player’s body in real time, enabling one to play games without any controllers.
Questions?
Simplest Type of Animation

• Rigid motion along a trajectory
  – Translation, rotation
Simplest Type of Animation

\[ M_{\text{object to world}} = M(t) \]

\[ = \begin{pmatrix}
    p_x(t) \\
    R(t) \\
    p_y(t) \\
    p_z(t) \\
    0 \\
    0 \\
    0 \\
    1
\end{pmatrix} \]
Simplest Type of Animation

\[ \mathbf{M}_{\text{object to world}} = \mathbf{M}(t) \]

\[ = \begin{pmatrix}
R(t) & p_x(t) \\
0 & p_y(t) \\
0 & p_z(t) \\
0 & 0 & 1
\end{pmatrix} \]

We saw how to animate \( R(t) \) on Tuesday.
Keyframing

- Describe motion of objects as a function of time, interpolating from a set of key object positions
  - E.g., record rotation and translation
Keyframing

• Use spline curves
  – Reuse tools from geometric modeling
  – Variable $t$ of curve $P(t)$ now describes time

• Good control, less tedious than specifying poses for every frame (duh)
  – Creating a nice animation still requires considerable skill and talent (unfortunately)
Possible Interpolation Caveats

Interpolation is not foolproof! Interpolating (vs. approximating) splines may undershoot and cause interpenetration.

MIT EECS 6.837 Fall 09 – Lehtinen
What Values are Keyframed?

Images from the Maya tutorial
What Values are Keyframed?

- Camera and object positions and orientations
  - Use quaternions for orientations
- Skeletal joint angles for skinning
  - Bone transformations
- Higher-level controls
  - Smile control, eye blinking, etc.
- Inverse kinematics
  - Endpoint positions
- Sometimes even geometry (vertex positions, normals)
  - “Blendshapes”
Questions?
Skinning/Enveloping

Gears of War 2

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
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

- Colored triangles are attached to 1 bone
- Black triangles are attached to more than 1 bone
- Note how they are near joints

James & Twigg 2005
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 $j$”
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 $B_j$. 


• 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.
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
    - This means translation independence.
    - (Again, a partition of unity – remember splines basis functions?)
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, well 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 $i$. 

MIT EECS 6.837 Fall 09 – Lehtinen

61
Computing Vertex Positions

Rest ("bind") pose

Bone 1: $T_1$

Bone 2: $T_2$

$\mathbf{p}_0$

"Skin"

- Vertex $\mathbf{p}_0$ has weights $w_{01} = 0.5$, $w_{02} = 0.5$
Computing Vertex Positions

Rest ("bind") pose

Bone 1: $T_1$

Bone 2: $T_2$

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

Transform by $T'_1$ and $T'_2$ yields $p'_{01}$, $p'_{02}$

After rotations

“Skin”
Computing Vertex Positions

Rest ("bind") pose

Bone 1: $T_1$

Bone 2: $T_2$

$\mathbf{p}_0$

"Skin"

After rotations

Bone 1: $T_1'$

Bone 2: $T_2'$

$\mathbf{p}'_{01}$ $\mathbf{p}'_0$ $\mathbf{p}'_{02}$

• Vertex $\mathbf{p}_0$ has weights $w_{01}=0.5, w_{02}=0.5$

• Transform by $T'_1$ and $T'_2$ yields $\mathbf{p}'_{01}, \mathbf{p}'_{02}$

the new position is $\mathbf{p}'_0 = 0.5*\mathbf{p}'_1 + 0.5*\mathbf{p}'_2$
Computing Vertex Positions

Rest ("bind") pose

Bone 1: $T_1$

Bone 2: $T_2$

After rotations

$\mathbf{p}_0$

"Skin"

Bone 1: $T'_1$

Bone 2: $T_2$

Vertex $\mathbf{p}_0$ has weights $w_{01}=0.5$, $w_{02}=0.5$

Transform by $T'_1$ and $T'_2$ yields $\mathbf{p}'_{01}$, $\mathbf{p}'_{02}$ the new position is $\mathbf{p}'_0 = 0.5*\mathbf{p}'_1 + 0.5*\mathbf{p}'_2$
SSD is Not Perfect

After rotations
Questions?

- Next, explanation of the mysterious quotes..
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
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.
Bind Pose cont’d

• In the rigging phase, we line the skeleton up with the undeformed skin.
  – This gives some “rest pose” bone transformations $B_j$
Bind Pose cont’d

• In the rigging phase, we line the skeleton up with the undeformed skin.
  – This gives some “rest pose” bone transformations $B_j$

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

• When we animate the model, the bone transformations $T_j$ change.
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.
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 $j$.
  – This is where the bind pose bone transformations $B_j$ come in.
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.

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 $T_j$. 

$$p_{ij}' = T_j B_j^{-1} p_i$$
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 \( B_j^{-1} \), 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.
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.

What is the transformation when the model is still in bind pose? The identity!
Questions?
Skinning Pseudocode

- Do the usual forward kinematics
  - maybe quaternion interpolation for rotations
  - get a matrix $T_j(t)$ per bone

- 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
  - maybe quaternion interpolation for rotations
  - get a matrix $T_j(t)$ per bone
- 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
  – maybe quaternion interpolation for rotations
  – get a matrix $T_j(t)$ per bone

• 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
  – maybe quaternion interpolation for rotations
  – get a matrix $T_j(t)$ per bone

• For each skin vertex $p_i$

$$p_i' = \sum_{j} w_{ij} T_j(t) B_j^{-1} p_i$$

• 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)
Questions?
Hmmh...

- This is what we do to get deformed positions

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

- 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...

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

- 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 exactly what I warned you of on Tuesday: 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
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: **Automatic Linearization of Nonlinear Skinning, I3D 2009**
People Have Thought of This...

Skinning with Dual Quaternions

L. Kavan, S. Collins, J. Zara, C. O'Sullivan

Trinity College Dublin
Czech Technical University in Prague

http://isg.cs.tcd.ie/projects/DualQuaternions/
Real-time enveloping with rotational regression
Wang, Pulli, Popovic
We learn a fast model from exported examples.

Black Box Simulation  ➔  Exported Examples (skeleton-mesh pairs)  ➔  Fast Model

\[(q^*, y^*)\]

Real-Time Enveloping with Rotational Regression
Robert Y. Wang\(^1\) Kari Pulli\(^{1,2}\) Jovan Popovic\(^1\)
\(^1\)Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology
\(^2\)Nokia Research Center

Slide from Rob Wang
Figuring out the Weights

- Usual approach: Paint them on the skin.
- Can also find them by optimization from example poses and deformed skins.
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!
  - http://www.mit.edu/~ibaran/autorig/
Pinocchio

Automatic Rigging and Animation of 3D Characters

SIGGRAPH 2007

papers_0030
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

*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.
Skinning Mesh Animations

Doug L. James
Christopher D. Twigg

Carnegie Mellon University
That’s All for Today!

• Further reading

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