Computer animation: rigging, FK and IK

Computer Graphics
CMU 15-462/15-662, Fall 2016
Generating Hand-Drawn Animation

- Senior artist draws keyframes
- Assistant draws inbetweens

inbetweens ("tweening")
Keyframing

- Basic idea:
  - specify important events only, computer fills in the rest via interpolation/approximation
How do you interpolate data?
Splines

- Common interpolant: piecewise cubic polynomials

Q: What are the three properties we must consider when deciding which type of spline to use?
But what exactly are we interpolating?
A simple example

- Keyframe and interpolate position of ball over time
An example: camera path

- Animate position, view direction, “up” direction
  - each path is a function $f(t) = (x(t), y(t), z(t))$
  - each component ($x,y,z$) is a spline
Animation Rigs

user-defined mappings between a small number of parameters and the deformations of a high-res mesh.
Animation Rigs

Animations are time trajectories specified for rig parameters (avars)
Simplest type of rig: Blend Shapes

- Keyframes represent surfaces (in correspondence)

- Simplest scheme: take linear combination of vertex positions

- Spline used to control choice of weights over time
Blend Shape Sculpting

- Traditionally a manual process

https://www.youtube.com/watch?v=8Nu3znisiBg#t=01m26s
Q: What are the pros and cons of blend shapes? When might it start to break down?
Cage-based deformers

- Embed model in a coarse mesh (cage)
  - deform coarse mesh, reconstruct hi-res geometry
Cage-based deformers

- Embed model in a coarse mesh (cage)
  - deform coarse mesh, reconstruct hi-res geometry
  - e.g. model vertices are weighted sum of cage vertices

\[
v = \sum_{j=1}^{m} w_j(v) c_j
\]

Q: where do the weights come from?
Many, many ways of computing interpolation weights with different properties one might care about...

<table>
<thead>
<tr>
<th>Method</th>
<th>N-gon</th>
<th>Concave</th>
<th>Shape</th>
<th>≥ 0</th>
<th>$C^1$</th>
<th>Lag.</th>
<th>Closed</th>
<th>Mono.</th>
<th>Out</th>
<th>Local</th>
<th>Poly</th>
<th>Coord.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barycentric</td>
<td>X</td>
<td>X</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>•</td>
</tr>
<tr>
<td>Wachspress</td>
<td>•</td>
<td>X</td>
<td>•</td>
<td>X</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>X</td>
<td>?</td>
<td>X</td>
<td>X</td>
<td>•</td>
</tr>
<tr>
<td>Natural Neighbor</td>
<td>•</td>
<td>•</td>
<td>X</td>
<td>•</td>
<td>X</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Mean Value</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>X</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>X</td>
<td>•</td>
<td>•</td>
<td>X</td>
<td>•</td>
</tr>
<tr>
<td>Green and others</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>X</td>
<td>•</td>
<td>•</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>•</td>
</tr>
<tr>
<td>Positive Mean Value</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>X</td>
<td>•</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>•</td>
</tr>
<tr>
<td>Harmonic</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>X</td>
<td>•</td>
<td>•</td>
<td>X</td>
<td>•</td>
</tr>
<tr>
<td>Maximum Entropy</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>X</td>
<td>?</td>
<td>X</td>
<td>X</td>
<td>•</td>
</tr>
<tr>
<td>Const. Biharmonic</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>X</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>X</td>
</tr>
</tbody>
</table>
Working with cages can be too restrictive, hard to have direct control
But very often, shape implies skeleton

[Au et al. 2008]
... and skeleton imposes a fair bit of structure in how shape can deform

[Blair 1994]
Key idea: animate just the skeleton (<< DOFs), have mesh “follow” automatically
Skeletal Animation – at one end of the spectrum

Simulate muscles, fatty tissues, skin, interactions with the environment, etc....
Skeletal Animation – at one end of the spectrum

Provides the “right” answer

Very demanding computationally

Need much faster solution for interactive applications
Linear Blend Skinning
Linear Blend Skinning
Forward Kinematics

Kinematic Skeletons
- Hierarchy of affine transformations
- Joints: local coordinate frames
- Bones: vectors between consecutive pairs of joints
- Each non-root bone defined in frame of unique parent
- Changes to parent frame affect all descendent bones
- Both skeleton and skin are designed in a rest (dress/bind) pose
Forward Kinematics

Assume n+1 joints: 0, 1, ..., n (0 == root)
- each joint corresponds to a frame
- \( p(j) \): the parent of joint \( j \) (root special: \( p(0) = -1 \))
- frame of joint \( j \) expressed w.r.t. to frame of \( p(j) \)

\[
p(j) R_j = \begin{pmatrix} r_{11}(j) & r_{12}(j) & r_{13}(j) & t_1(j) \\ r_{21}(j) & r_{22}(j) & r_{23}(j) & t_2(j) \\ r_{31}(j) & r_{32}(j) & r_{33}(j) & t_3(j) \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} Rot(j) & t(j) \\ 0 & 1 \end{pmatrix}
\]
Forward Kinematics

Assume n+1 joints: 0, 1, ..., n (0 == root)
- each joint corresponds to a frame
- \( p(j) \): the parent of joint j (root special: \( p(0) = -1 \))
- frame of joint j expressed w.r.t. to frame of \( p(j) \)

- What does this transform correspond to?
- \( t(j) \) typically comes from bind pose
- \( Rot(j) \) typically comes from animation

\[
p(j) R_j = \begin{pmatrix} Rot(j) & t(j) \\ 0 & 1 \end{pmatrix}
\]
Forward Kinematics

- The transformation from frame \( j \) to world is:

\[
\begin{align*}
  wR_j &= wR_0 \cdots \begin{pmatrix}
    p(p(j)) & R_{p(j)p(j)} & R_j
  \end{pmatrix} \\
  &= T(0) Rot(0) \cdots T(p(j)) Rot(p(j)) T(j) Rot(j)
\end{align*}
\]

- Each joint rotation \( Rot(j) \) has up to 3-DOFs
  - Translation & scale also possible, but less common
  - \( w \overline{R}_j \) is defined the same way as \( wR_j \), but in rest pose (e.g. no joint rotations)
Skinning

Given joint angles, we compute configuration of the skeleton using forward kinematics. What do we do about the surface mesh?

- Move vertices along with the bones!
- Attempt 1: assign each vertex to closest bone, compute world coordinates according to bone’s transformation

\[ v = \sum_w R_j w \bar{R}_j^{-1} v' \]

vertex skin
Skinning

Given joint angles, we compute configuration of the skeleton using forward kinematics. What do we do about the surface mesh?

- **Move vertices along with the bones!**

- **Attempt 1:** assign each vertex to closest bone, compute world coordinates according to bone’s transformation

  \[ v = \overline{R}_j R_j^{-1} v' \]

  - Skinned vertex coordinates
  - Vertex coordinates in rest pose
  - Vertex coordinates in coordinate frame of bone j

- **Rigid skinning:** fine if vertices are close to bone, fine for small rotations, used in older video games

- **Can do better with only slight increase in complexity**
Linear Blend Skinning

Given joint angles, we compute configuration of the skeleton using forward kinematics. What do we do about the surface mesh?

- Attempt 2: assign each vertex to multiple bones, compute world coordinates as convex combination
- Weights: influence of each bone on the vertex
- Leads to smoother* deformations of the skin

\[ v = \sum_j \alpha_{jw} R_{jw} \bar{R}_j^{-1} v' \]

- Skinned vertex coordinates
- Vertex coordinates in rest pose
- Vertex coordinates in coordinate frame of bone j
- Skinning weights
Linear Blend Skinning

resulting vertex position

first bone transformation

second bone transformation
How are skinning weights generated?

- Often they are manually painted
- Can be computed automatically (active research area)
  - e.g. based on distance to bones - how would you do this? What might go wrong?
Linear Blend Skinning artifacts

- What’s causing this?

- How to fix candy-wrapper effects: better weights, introduce auxiliary joints/bones, employ better interpolation schemes for transformations, pose-space deformers (PSD)
Linear Blend Skinning

Widely used in industry!
Forward Kinematics

- Given joint angles, compute configuration of the skeleton
Inverse Kinematics (IK)

- Given goal(s) for “end effector” compute joint angles
- Very important technique in animation & robotics!

- Many, many algorithms: analytic formulations for specific cases, cyclic coordinate descent, energy-based methods, etc.
Pitfalls of hand-crafted animation: the Uncanny Valley
Full-body motion capture

https://www.youtube.com/watch?v=zQPfxcQKr0Q
Is Motion Capture the solution to the Uncanny Valley?

https://www.youtube.com/watch?v=piJ4Zke7EUw