Animation

CS 4620 Lecture 33
Announcements

• Grading A5 (and A6) on Monday after TG

• 4621: one-on-one sessions with TA this Friday
Quaternions

• Remember that
  – Orientations can be expressed as rotation
    • Why?
      – Start in a default position (say aligned with z axis)
      – New orientation is rotation from default position
    – Rotations can be expressed as (axis, angle)

• Quaternions let you express (axis, angle)
Quaternion for Rotation

- Rotate about axis $\vec{a}$ by angle $\theta$

$$q = (s, v) = (s, v_1, v_2, v_3)$$

$$s = \cos \left( \frac{\theta}{2} \right)$$

$$v = \sin \left( \frac{\theta}{2} \right) \hat{a}$$
Rotation Using Quaternion

• A point in space is a quaternion with 0 scalar

\[ X = (0, \vec{x}) \]

• Rotation is computed as follows

\[ x_{rotated} = qXq^{-1} = qXq' \]

• See Buss 3D CG: A mathematical introduction with OpenGL, Chapter 7
Why Quaternions?

• Fast, few operations, not redundant
• Numerically stable for incremental changes
• Composes rotations nicely
• Convert to matrices at the end
• Biggest reason: spherical interpolation
Interpolating between quaternions

• Why not linear interpolation?
  • Need to be normalized
  • Does not have a constant rate of rotation

\[
\frac{(1 - \alpha)x + \alpha y}{\| (1 - \alpha)x + \alpha y \|}
\]
Spherical Linear Interpolation

- Intuitive interpolation between different orientations
  - Nicely represented through quaternions
  - Useful for animation
  - Given two quaternions, interpolate between them

- Shortest path between two points on sphere
  - Geodesic, on Great Circle
Spherical linear interpolation ("slerp")

\[ \alpha + \beta = \psi \]
\[ \mathbf{v}(t) = w_0 \mathbf{v}_0 + w_1 \mathbf{v}_1 \]
\[ \frac{\sin \alpha}{w_1} = \frac{\sin \beta}{w_0} = \frac{\sin(\pi - \psi)}{1} = \sin \psi \]
\[ w_0 = \frac{\sin \beta}{\sin \psi} \]
\[ w_1 = \frac{\sin \alpha}{\sin \psi} \]
\[ \psi = \cos^{-1}(\mathbf{v}_0 \cdot \mathbf{v}_1) \]
Quaternion Interpolation

- Spherical linear interpolation naturally works in any dimension
- Traverses a great arc on the sphere of unit quaternions
  Uniform angular rotation velocity about a fixed axis

\[
\psi = \cos^{-1}(q_0 \cdot q_1)
\]

\[
q(t) = \frac{q_0 \sin(1 - t)\psi + q_1 \sin t\psi}{\sin \psi}
\]
Practical issues

• When angle gets close to zero, use small angle approximation
  – degenerate to linear interpolation
• When angle close to 180, there is no shortest geodesic, but can pick one
• $q$ is same rotation as $-q$
  – if $q_1$ and $q_2$ angle < 90, slerp between them
  – else, slerp between $q_1$ and $-q_2$
Interpolating transformations

• Linear interpolation of matrices is not effective
  • leads to shrinkage when interpolating rotations
• One approach: always keep transformations in a canonical form (e.g. translate-rotate-scale)
  • then the pieces can be interpolated separately
  • rotations stay rotations, scales stay scales, all is good
• But you might be faced with just a matrix. What then?
Decomposing transformations

• A product $M = TRS$ is not hard to take apart
  – translation sits in the top right

• If we allow $S$ to be a scale along arbitrary axes

• $M = TRS$ where
  • $T$ is a translation
  • $R$ is a rotation
  • $S$ is a symmetric matrix (positive definite if no reflection)
  • Linear algebra name
    – Polar decomposition (at least the $A = RS$ part)
Parameterizing rotations

• Unit quaternions
  A 4D representation (like 3D unit vectors for 2D sphere)
  Good choice for interpolating rotations

• These are first examples of motion control
  Matrix = deformation
  Angles/quaternion = animation controls
The artistic process of animation

• What are animators trying to do?

• "Principles of Traditional Animation Applied to 3D Computer Graphics," SIGGRAPH'87, by John Lasseter

• Widely cited set of principles laid out by Frank Thomas and Ollie Johnston in The Illusion of Life (1981)

• The following slides follow Michael Comet’s examples: www.comet-cartoons.com
Animation principles: timing

• Speed of an action is crucial to the impression it makes

  examples with same keyframes, different times:

  60 fr: looking around  30 fr: “no”  5 fr: just been hit
Animation principles: ease in/out

- Real objects do not start and stop suddenly
  animation parameters shouldn’t either

straight linear interp.  
 ease in/out

a little goes a long way (just a few frames acceleration or deceleration for “snappy” motions)
Animation principles: moving in arcs

- Real objects also don’t move in straight lines; generally curves are more graceful and realistic.
Animation principles: anticipation

- Most actions are preceded by some kind of “wind-up”
Animation principles: exaggeration

- Animation is not about exactly modeling reality
- Exaggeration is very often used for emphasis
Animation principles: squash & stretch

- Objects do not remain perfectly rigid as they move.
- Adding stretch with motion and squash with impact: models deformation of soft objects.
  
  Indicates motion by simulating exaggerated “motion blur”.

[www.animdesk.com]
Animation principles: follow through

- We’ve seen that objects don’t start suddenly
- They also don’t stop on a dime
Anim. principles: overlapping action

- Usually many actions are happening at once
Animation principles: staging

- Want to produce clear, good-looking 2D images
  need good camera angles, set design, and character positions
Principles at work: weight
Extended example: Luxo, Jr.
Computer-generated motion

• Interesting aside: many principles of character animation follow indirectly from physics

• Anticipation, follow-through, and many other effects can be produced by simply minimizing physical energy

• Seminal paper: “Spacetime Constraints” by Witkin and Kass in SIGGRAPH 1988
Forward Kinematics

Inverse Kinematics
• **Forward kinematics**
  – Describe positions of body parts as fn of joint angles
  – Body parts: bones

• **Inverse kinematics**
  – Constrain locations for bones and solve for joint angles
Forward Kinematics

- Articulated body
  - Hierarchical transforms
  - Comes from robotics
Rigid Links and Joint Structure

- Links connected by joints
  Joints are purely rotational (single DOF)
  Links form a tree (no loops)
  End links have end effectors
Basic surface deformation methods

• Mesh skinning: deform a mesh based on an underlying skeleton
• Blend shapes: make a mesh by combining several meshes
• Both use simple linear algebra
  Easy to implement—first thing to try
  Fast to run—used in games
• The simplest tools in the offline animation toolbox
Mesh skinning

• A simple way to deform a surface to follow a skeleton
Skinning

- Embed a skeleton into a character mesh
- Animate “bones”
  - Change joint angles over time
  - Key framing, etc.
- Bind skin vertices to bones
  - Animate skeleton
  - Skin will move with it
Mesh skinning math: setup

- Surface has control points $p_i$:
  Triangle vertices, spline control points, subdiv base vertices
- Each bone has a transformation matrix $M_j$:
  Normally a rigid motion
- Every point–bone pair has a weight $w_{ij}$:
  In practice only nonzero for small # of nearby bones
  The weights are provided by the user
Colored tris attached to one bone

Black to > one bone

James & Twigg, Skinning Mesh Animations, 2005, used with permission from ACM, Inc.
Mesh skinning math

- Deformed position of a point is a weighted sum of the positions determined by each bone’s transform alone weighted by that vertex’s weight for that bone

\[ w_{ij} \text{: How much should vertex } i \text{ move with bone } j \]

\[ p'_i = \sum_j w_{ij} M_j p_i \]
Mesh skinning

• Simple and fast to compute
  Can even compute in the vertex stage of a graphics pipeline
• Used heavily in games
• One piece of the toolbox for offline animation
  Many other deformers also available
Mesh skinning: classic problems

- Surface collapses on the inside of bends and in the presence of strong twists
  
  Average of two rotations is not a rotation!

  Add more bones to keep adjacent bones from being too different, or change the blending rules.

[Image: Images of a human arm in different poses]
Blend shapes

• Another very simple surface control scheme
• Based on interpolating among several key poses
  Aka. blend shapes or morph targets
Blend shapes math

- Simple setup
  User provides key shapes—that is, a position for every control point in every shape: $p_{ij}$ for point $i$, shape $j$
  Per frame: user provides a weight $w_j$ for each key shape
  - Must sum to 1.0
- Computation of deformed shape
  $p'_i = \sum_j w_j p_{ij}$
  - Works well for relatively small motions
    Often used for facial animation
    Runs in real time; popular for games
Animation

- Key frame
- Motion capture
- Physics-based