Character Animation

Please ask questions any time!
Presentations will be 40-45 minutes!
3 chairpeople: make your choice
  - Kremer, John, Lessle
Content

- **Overview**
  - Modeling
  - Animation
- **Kinds of animation**
- **Basics of 3D transformations**
  - affine transformations
  - homogeneous coordinates
  - frame of reference
- **Articulated structures**
  - forward/inverse kinematics
Part 1

Overview:
Modeling and Animation
Ingredients

- character animation = modeling + animation
- model consists of
  - mesh (skin)
  - skeleton
  - skinning/rigging
  - morph targets
Mesh

- created by artist or 3D scan
- consists of polygons
- vertex/vertices + face(s)
- texture (e.g. bitmap file) determines faces' looks
- usually triangulated for graphics engine (automatic)
- number of polygons determine whether manageable in real time (low-poly models vs. high-poly models)
- structure of mesh highly important for skinning/deformation
Skeleton

- created after mesh
- consists of **joints**
- **bone** between two joints (usually synonymous with root joint)
- joint hierarchy
- more joints => more difficult to animate
- for this seminar, **skeleton** is the most important construct
- skeleton moves => mesh should follow (deformation)
- **bind** each vertex to a joint
- resulting mesh deformation can look awkward
- bind each vertex to **multiple joints**, using weights ("influence")
- structure of mesh important!
Morph targets

- easy technique to animate faces
- no skeleton!
- given two meshes A and B with equal structures: compute intermediate mesh

\[ M_{\text{final}} = \gamma_1 \cdot M_1 + \ldots + \gamma_N \cdot M_N \]

\[ \sum \lambda_i = 1 \]
Animation

- **Animation** = change (of a figure‘s pose) over time
- **Pose** =
  - specific configuration of skeleton (rotation of each joint + translation of root joint)
  - $\Theta := (\theta_1, \ldots, \theta_n)$
  - $P := (\Theta, t)$
- **Animation**: time $\to$ pose
- **For continuous motion**: $> 25$ fps (usually around 50 fps)
Manual Animation

- artist defines only "important" (=key) poses
  
  \[ P_1 = (t_1, \Theta_1) \text{ and } P_{10} = (t_{10}, \Theta_{10}) \]

- computer does "in-betweening"
  
  \[ P_i = (I_{\text{trans}}(t_1, t_{10}), I_{\text{rot}}(\Theta_1, \Theta_{10})) \]

- usually interpolation between joint rotations

- animator adjusts temporal dynamics

- alternatives
  - motion capture
  - procedural animation
  - physical simulation
Animation in real time

- Model + animation produced in 3D tool (3DS, Blender, Maya)
- Can be played in realtime using game/graphics engine (Unreal, Ogre3D, Crystal Space)
- Exchange format:
  - COLLADA
Real time

- Inside the graphics engine
  - usually in the main loop called $N$ times per second
  - set morph targets (linear combination)
  - set animation time
  - manipulate joints individually (e.g. gaze)

- Higher-level controls
  - play/pause animation
  - blend animations
    - mask joints
Part 2

Basics of 3D transformations
2D Transformations

Translation

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  x \\
  y
\end{bmatrix} + \begin{bmatrix}
  a \\
  b
\end{bmatrix} = \begin{bmatrix}
  x + a \\
  y + b
\end{bmatrix}
\]

\[\Leftrightarrow\]

\[p' = p + t\]
Rotation

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix}
= \begin{bmatrix}
  \cos(\alpha) & -\sin(\alpha) \\
  \sin(\alpha) & \cos(\alpha)
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

\[p' = R \cdot p\]
2D Transformations

Scale

\[
\begin{bmatrix}
    x' \\
    y'
\end{bmatrix} = \begin{bmatrix}
    s_x & 0 \\
    0 & s_y
\end{bmatrix} \cdot \begin{bmatrix}
    x \\
    y
\end{bmatrix} = \begin{bmatrix}
    s_x \cdot x \\
    s_y \cdot y
\end{bmatrix}
\]

\[\Leftrightarrow\]

\[p' = S \cdot p\]
Rigid-body transformations

- arbitrary sequence of rotation and translation
- preserves angles and lengths
- e.g., unit square remains a unit square
Affine transformations

- arbitrary sequence of rotation, translation and scale
- affine transformation preserves parallelity
- using homogenous coordinates: affine transformation as a matrix
Homogeneous coordinates

Translation

\[
\begin{bmatrix}
    x' \\
    y'
\end{bmatrix}
= \begin{bmatrix}
    x \\
    y
\end{bmatrix} + \begin{bmatrix}
    a \\
    b
\end{bmatrix}
= \begin{bmatrix}
    x + a \\
    y + b
\end{bmatrix}
\]

\[
\begin{bmatrix}
    x' \\
    y'
\end{bmatrix}
= \begin{bmatrix}
    1 & 0 & a \\
    0 & 1 & b \\
    0 & 0 & 1
\end{bmatrix} \cdot \begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
= \begin{bmatrix}
    x + a \\
    y + b \\
    1
\end{bmatrix}
\]

\[
p' = T \cdot p
\]
Homogeneous coordinates

Rotation

\[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} = \begin{bmatrix}
\cos(\alpha) & -\sin(\alpha) & 0 \\
\sin(\alpha) & \cos(\alpha) & 0 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

\( \Leftrightarrow \)

\( p' = R \cdot p \)
Homogeneous coordinates

Scale

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} =
\begin{bmatrix}
  s_x & 0 & 0 \\
  0 & s_y & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix} =
\begin{bmatrix}
  s_x \cdot x \\
  s_y \cdot y \\
  1
\end{bmatrix}
\]

\[p' = S \cdot p\]
Transformation composition

- simply multiply the matrices (order matters!)

\[ p' = S \cdot R \cdot T \cdot p \]
\[ p' = M \cdot p \]
Going 3D

- translation + scale analogously
- rotation: 3 rotations around x, y, z axis

\[
\begin{bmatrix}
1 & 0 & a \\
0 & 1 & b \\
0 & 0 & 1 \\
\end{bmatrix} \Rightarrow
\begin{bmatrix}
1 & 0 & 0 & a \\
0 & 1 & 0 & b \\
0 & 0 & 1 & c \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
s_x & 0 & 0 \\
0 & s_y & 0 \\
0 & 0 & s_z \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
cos(\alpha) & -sin(\alpha) & 0 & 0 \\
sin(\alpha) & cos(\alpha) & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
cos(\alpha) & -sin(\alpha) & 0 & 0 \\
sin(\alpha) & cos(\alpha) & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix}
\]
Part 3

Articulated Structures
FK + IK
Quaternions
Articulated Structures

- Most used method to control human-like structures for animation
- Kinematics
  - study of motion independent of underlying forces (no physics!): position, velocity, accel.
- Articulated figure
  - consists of series of rigid links connected at (rotary) joints
More terminology

- **Degrees of freedom (DOF)**
  - number of indep. variables to specify the state of the structure

- **State vector:**
  - set of parameters defining the DOF
  - $\Theta = (\theta_1, \ldots, \theta_n)$
  - e.g., for a 6 DOF position:
    $\Theta = (x, y, z, rx, ry, rz)$
Example: 2 DOF

Forward Kinematics (FK): \( X = f(\theta) \)

Inverse Kinematics (IK): \( \theta = f^{-1}(\theta) \)
Example: 2 DOF

Forward Kinematics (FK): \( X = f(\theta) \)

\[
x = l_1 \cos \theta_1 + l_2 \cos (\theta_1 + \theta_2) \\
y = l_1 \sin \theta_1 + l_2 \sin (\theta_1 + \theta_2)
\]
Example: 2 DOF

Inverse Kinematics (IK): $\theta = f^{-1}(X)$

\[
\theta_2 = \cos^{-1} \left( \frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1l_2} \right)
\]

\[
\theta_1 = \frac{-(l_2 \sin \theta_2)x + (l_1 + l_2 \cos \theta_2)y}{(l_2 \sin \theta_2)y + (l_1 + l_2 \cos \theta_2)x}
\]
Inverse Kinematics

- convenient for arm animation (picking, pointing, gesture) and walking

- drawbacks
  - becomes harder with increasing DOF
  - multiple solutions exist
    - solution: introduce constraints
    - however: no precise control

- animation usually relies on both FK + IK
Joints

Revolute joint (1 DOF)

Prismatic joint (1 DOF)

Ball-and-socket joint (3 DOF)
Representation

- use tree of nodes and arcs
- highest node = root node
- every node positioned relative to frame-of-reference of parent node
- every arc represents an affine transformation
Frames of reference

- head
- chest
- up_arm_L
- lo_arm_L
- up_arm_R
- lo_arm_R
- chest
- up_arm_L
- lo_arm_R
Changing frame of reference

$M_{A \leftarrow B} = T(2,1) \ S(0.5) \ R(45^\circ)$

$p^A = M_{A \leftarrow B} \ p^B$

What if we have another frame of reference $C$ inside of $B$:

Wanted: $M_{A \leftarrow C}$

Given: $M_{A \leftarrow B}$ and $M_{B \leftarrow C}$

$p^A = M_{A \leftarrow B} \ p^B$

$p^B = M_{B \leftarrow C} \ p^C$

$p^A = M_{A \leftarrow B} \ M_{B \leftarrow C} \ p^C$
Frames of reference

\[ p = M_c \ M_{\text{ual}} \ M_{\text{al}} \ p^{\text{al}} \]
Rotations are the most important operation in skeletal animation.

Typical problem: interpolate between two joint positions.

So far: matrix representation.

Alternatives:
- Euler angles
- Quaternion
Euler angles

- any rotation can be expressed using x/y/z rotations: \( R = (r_x, r_y, r_z) \)

- advantage:
  - easy to understand / visualize
  - minimal: 3 parameters for 3 DOF

- disadvantages:
  - ambiguous: \( R(45°,90°,0)=R(0,0,45°) \)
    → depending on your choice you get 2 different interpolation paths! (VIDEO)
  - linear interpolation results in nonlinear motion
  - gimbal lock
Quaternions

- Origin: extending complex/imaginary numbers (Hamilton 1866)
- Intuition: represent a rotation by an axis of rotation and a rotation angle
- Notation:
  - $Q = w + x\, i + y\, j + z\, k$
  - $q = (w, \, v) \quad v = (x, \, y, \, z)$
Quaternions

- Multiplication:
  - $q_1 q_2 = (w_1 w_2 - v_1 \cdot v_2, \ v_1 \times v_2 + w_1 v_2 + w_2 v_1)$

- Important property: composition by multiplication
  - $q = q_1 q_2$

- Fast transformation, e.g. Euler -> Quat
  - $q_x = (\cos(r_x/2), (\sin(r_x/2), 0, 0))$
  - $q = q_z q_y q_x$
Quaternions

- Most importantly: algorithm for smooth interpolation without gimbal lock
  - SLERP: next talk
- Advantages
  - interpolation without singularities
  - computationally efficient
- Disadvantage
  - non-intuitive maths
That’s it for today

- Questions?
- FAQ for matrices etc.: http://www.gamedev.net/reference/articles/article1691.asp