Articulated Characters 1
Skeleton

- A skeleton is a framework of rigid bones connected by articulated joints and arranged in a hierarchical data structure.
- Used as an invisible armature to position and orient geometry (surface triangles).
Joints

• Connect rigid bodies, while allowing for relative motion between them

• Different types: hinge, ball-and-socket, saddle joint, sliding...

• A Joint has 0-6 degrees of freedom (DoF)
  – A 0 DOF joint rigidly connects two bodies into a single rigid body
  – A full 6 DOF joint doesn’t do anything, and each of the bodies are free to move entirely independently
Joints

• Rotational Joint
  – 1-DoF Rotation: a rotation matrix $\text{Rot}_{4x4}(\mathbf{n}, \theta)$ defined by axis $\mathbf{n}$ and angle $\theta$
  – 2-DoF Rotation: multiplication of two sequential rotation matrices about different axes
  – 3-DoF Rotation: multiplication of 3 sequential rotation matrices about 3 different axes
    • Like Euler Angles, has the same problem of Gimbal lock
    • Instead, specify a 3D rotation about an arbitrary axis

• Translational Joint
  – Similar to rotational joints, translational joints can be specified to translate along any axis (1-DoF, 2-DoF, 3-DoF)
  – Translation matrix $\text{T}_{4x4}(\mathbf{v})$ is defined by the translation vector $\mathbf{v}$

• Compound Joint
  – Combine rotational and translational joints together, i.e. 6-DoF joint
Joints: examples...

- **Hinge**
  - DoF=1
  - rotation along one axis

- **Saddle**
  - DoF=2
  - back and forth & up and down motion
  - No “rotation”

- **Ball-and-socket**
  - DoF=3
  - rotation along all 3 axes

- **Sliding**
  - DoF=2
  - sliding in a plane

Image from http://www.shockfamily.net/skeleton/JOINTS.HTML
Joints in Character Animation

- Joints are organized in a hierarchy
- The **root** is the position of the “base” of the skeleton
  - typically the backbone or pelvis
  - the root has all 6 DoF so it can be placed anywhere, with any orientation
- Typically, other joints have only rotational DoFs
  - but in reality they have prismatic (translational) components as well

3 translational and 48 rotational DoFs

1 DoF: Knee

2 DoFs: Wrist

3 DoFs: Shoulder

Image from http://www.cc.gatech.edu/classes/AY2012/cs4496_spring/
State Vector

\[ q = x, y, z, \theta, \phi, \sigma, \theta_{th}, \phi_{th}, \sigma_{th}, \theta_{kn}, \ldots \]

Root Joint

\[ x, y, z, \theta, \phi, \sigma \]

\[ \theta_{thigh}, \phi_{thigh}, \sigma_{thigh} \]

\[ \theta_{knee} \]

\[ \theta_{ankle}, \phi_{ankle} \]
Joint Hierarchy

- The articulated skeleton can be described by a tree
  - Nodes: joints
  - Edges: bones

- The root of the tree can be any joint, but is usually near the center of the character (e.g. spine or pelvis)
- The transformation of a joint is defined relative to its parent joint in the hierarchy
Joint Parameters

• Offset
  – A fixed translational offset position in the space of the parent body/joint, which acts as a pivot point for the joint’s movement.

• Orientation
  – Orientation of a joint’s local coordinate system defined in the space of its parent body/joint (a matrix or quaternion)
  – If using homogeneous coordinates, offset and orientation can be defined together in one 4x4 matrix

• Joint Limits
  – The minimum and maximum limits for each DOF that can be enabled or disabled independently
  – E.g. the human elbow can bend to about +150 degrees and hyperextend back as much as –10 degrees
Joint Parameters

J1
Offset: (0,6)
Orientation: Rot(-45)
Limit: None

J2
Offset: (5,0)
Orientation: Rot(50)
Limit: [0,90]

J3
Offset: (4,0)
Orientation: Rot(30)
Limit: [-90,90]
Forward Kinematics

- Specify the base position plus joint angles to prescribe motion
- To compute the final position and orientation of a joint in the world coordinate system, the transformations of all parent joints need to be combined together
  - Changing a parent joint affects *all* of its child bodies/joints
  - Changing a child joint does not affect any of its parent bodies/joints
Forward Kinematics

**Goal:** Compute the position of a local point \( P \) in the world space.

\[
P_{\text{world}} = T(0, 6)R(-45)T(5, 0)R(50)T(4, 0)R(30)P_{\text{local}}
\]
Transformations in World & Object Space

**World Space**: Read right to left

- T(1,1)

- T(1,1)

- R(45) T(1,1)

**Object Space**: Read left to right

- R(45)

- R(45)

- R(45) T(1,1)
In the View of World Space

\[ P_{\text{world}} = T(0,6)R(-45)T(5,0)R(50)T(4,0)R(30)P_{\text{local}} \]
In the View of World Space

\[ P_{\text{world}} = T(0,6)R(-45)T(5,0)R(50)T(4,0)R(30)P_{\text{local}} \]
In the View of World Space

\[ P_{\text{world}} = T(0,6)R(-45)T(5,0)R(50)T(4,0)R(30)P_{\text{local}} \]
In the View of World Space

\[ P_{\text{world}} = T(0,6)R(-45)T(5,0)R(50)T(4,0)R(30)P_{\text{local}} \]
In the View of World Space

\[ P_{\text{world}} = T(0,6)R(-45)T(5,0)R(50)T(4,0)R(30)P_{\text{local}} \]
In the View of World Space

\[ P_{\text{world}} = T(0,6)R(-45)T(5,0)R(50)T(4,0)R(30)P_{\text{local}} \]
In the View of World Space

$P_{\text{world}} = T(0,6) R(-45) T(5,0) R(50) T(4,0) R(30) P_{\text{local}}$
In the View of Object Space

\[ P_{\text{world}} = T(0,6)R(-45)T(5,0)R(50)T(4,0)R(30)P_{\text{local}} \]
In the View of Object Space

\[ P_{\text{world}} = T(0,6)R(-45)T(5,0)R(50)T(4,0)R(30)P_{\text{local}} \]
In the View of Object Space

\[ \mathbf{P}_{\text{world}} = T(0, 6) R(-45) T(5, 0) R(50) T(4, 0) R(30) \mathbf{P}_{\text{local}} \]
In the View of Object Space

\[ P_{\text{world}} = T(0, 6) R(-45) T(5, 0) R(50) T(4, 0) R(30) P_{\text{local}} \]
In the View of Object Space

\[ P_{\text{world}} = \text{T}(0, 6)\text{R}(-45)\text{T}(5, 0)\text{R}(50)\text{T}(4, 0)\text{R}(30)P_{\text{local}} \]
In the View of Object Space

\[ P_{\text{world}} = T(0, 6)R(-45)T(5, 0)R(50)T(4, 0)R(30)P_{\text{local}} \]
In the View of Object Space

\[ P_{\text{world}} = T(0,6)R(-45)T(5,0)R(50)T(4,0)R(30)P_{\text{local}} \]
Pros and Cons

• The hierarchy is simple and efficient
  – Specify the motion of the root node first
  – Then specify the poses of children one layer at a time

• Especially suitable for modeling motions in open space (without constraints)
  – E.g. flying birds, swimming fish...

• Has difficulties with interactions with the environment
  – Making a foot stay on the ground, fingers stay in contact with a cup, etc.
Forward Kinematics

• Given values for the joint DoF
  \[ \theta = [\theta_1, \theta_2, \ldots, \theta_n]^T \]

• Compute the end effectors in world space
  \[ e = [e_1, e_2, \ldots, e_m]^T \]

• Forward Kinematics defines and uses the function
  \[ e = F(\theta) \]

Given \( \theta = [\theta_1, \theta_2, \theta_3]^T \)

Find \( e = [e_1, e_2]^T \)

\((e_1, e_2) = F(\theta_1, \theta_2, \theta_3)\)
Inverse Kinematics

• Given the values for the end effectors in world space
  \[ \mathbf{e} = [e_1, e_2, \ldots, e_m]^T \]

• Compute the joint angles
  \[ \mathbf{\theta} = [\theta_1, \theta_2, \ldots, \theta_n]^T \]

• Inverse Kinematics defines:
  \[ \mathbf{\theta} = G(\mathbf{e}) \]

Given \( \mathbf{e} = [e_1, e_2]^T \)
Find \( \mathbf{\theta} = [\theta_1, \theta_2, \theta_3]^T \)
\( (\theta_1, \theta_2, \theta_3) = G(e_1, e_2) \)
Inverse Kinematics

- Finding a solution for IK can be hard
  - one unique solution
  - infinite solutions (underconstrained)
  - no solution (overconstrained)
- Cannot be solved analytically in most cases
- Usually requires numerical methods
  - Jacobian iterative method
  - Optimization based methods

Two solutions $[\theta_1, \theta_2, \theta_3]^T$ for constrained $[e_1, e_2]^T$
Linearization

• Use the secant approximation

\[ e - e_0 = \frac{dF}{d\theta} (\theta - \theta_0) \]

• Here \( J = \frac{dF}{d\theta} \) is the **Jacobian matrix** of partial derivatives of \( e = F(\theta) \)

• \( J \) defines the instantaneous changes in the end effectors \( e \) relative to changes in the angles \( \theta \)

• since \( e = F(\theta) \) is nonlinear, \( J \) is only valid as an approximation near the current configuration \( \theta_0 \)

• Algorithm: replace \( e \) with \( e_{\text{target}} \) and solve for \( \theta \)
Jacobian Matrix

For $\mathbf{e}=[\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_m]^T$ and $\mathbf{\theta}=[\theta_1, \theta_2, \ldots, \theta_n]^T$,

$$ J = \begin{bmatrix} \frac{\partial F_1}{\partial \theta_1} & \frac{\partial F_1}{\partial \theta_2} & \cdots & \frac{\partial F_1}{\partial \theta_n} \\ \frac{\partial F_2}{\partial \theta_1} & \frac{\partial F_2}{\partial \theta_2} & \cdots & \frac{\partial F_2}{\partial \theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial \theta_1} & \frac{\partial F_m}{\partial \theta_2} & \cdots & \frac{\partial F_m}{\partial \theta_n} \end{bmatrix} $$

The column of the $j$th joint $\mathbf{J}_j$ can be computed numerically as:

$$ \mathbf{J}_j = \frac{F(\mathbf{\theta} + b_j \delta \mathbf{\theta}) - F(\mathbf{\theta})}{\delta \mathbf{\theta}} $$

where $b_j = [0,0,\ldots, 1 ,\ldots,0,0]^T$

If the $j$th joint is a rotational joint with a single degree of freedom then the elements corresponding to the $i$th end effector in the $j$th column of $\mathbf{J}$ can be computed analytically as:

$$ \mathbf{J}_{ij} = \mathbf{v}_j \times (\mathbf{s}_i - \mathbf{p}_j) $$

where $\mathbf{v}_j$ is the unit vector pointing along the axis of rotation, $\mathbf{p}_j$ is the position of the joint, and $\mathbf{s}_i$ is the position of the $i$th end effector.
Iterative Solver

Input: \( e_0 \) – current end effector positions
\( e_t \) – target end effector positions
\( \theta_0 \) – current DoFs

Algorithm:
while(\(|e_t - e_0| > \text{threshold}|\){{
    compute \( J \); // Take many small steps and recompute \( J \) at each step
    \( \delta e = e_t - e_0 \); // Can scale this down to aim for smaller steps
    Solve \( J \delta \theta = \delta e \) to find \( \delta \theta \);
    update the DoFs with a small step of \( \alpha \delta \theta \): i.e., \( \theta_0 += \alpha \delta \theta \);
    update end effectors: \( e_0 = F(\theta_0) \);
}}


Solving $J \delta \theta = \delta e$

- $J$ is not guaranteed to be invertible, and is typically not even a square matrix.
- If $J$ is **overdetermined**, multiply both sides by $J^T$, solve $J^T J \delta \theta = J^T \delta e$ for the least-squares solution (Householder is better!)
  - this gives the **unique solution** too, if it exists (of course, there are better methods if the solution is unique)
- If $J$ is **underdetermined**, use pseudo inverse $J^+$, solve $\delta \theta = J^+ \delta e$ for the minimum norm solution
  - The general solution of $\delta \theta = J^+ \delta e$ can be written as
    $\delta \theta = J^+ \delta e + (I - J^+J)\gamma$
  - The second term $(I - J^+J)$ represents the orthogonal projection to the null space of $J$
  - For any $\beta = (I - J^+J)\gamma$ we have $J \beta = 0$, which means $\beta$ will only affect the interior joints and causes no motion for the end effectors
  - This null space term can be used for some secondary goals, e.g. finding the most natural positions for the joints, balance, etc.
    - think about all the “styles” of walking
- Take CS205!
Pros and Cons

• Modeling poses of characters interacting with other objects
  – E.g., make sure fingers are exactly in contact with a cup

• Cannot get a solution if the target position is impossible (over-constrained)
  – Try to find a solution as close as possible in some sense, but this could look bad

• Hard to pick the best solution for an under-constrained system (null space matters!)
  – Requires additional constraints or optimizing some quantities
Puppeteering

• Specifying animation curves and splines for every degree of freedom (root and all angles) by hand can be tedious
  – And hard to make look realistic
• Inverse kinematics can help
  – But still leaves the null space degrees of freedom unspecified
    • E.g. motion style
• It would be better if one could input motion in a higher level fashion
  – similar to how a puppeteer *guides* a puppet
• One solution to this is motion capture, where human movements are captured by various types of cameras
Motion Capture

• Attach a number of markers to a person
• Light is emitted from a number of cameras and reflected back to the cameras
• Compute the 3D location of the markers by inferring depth values
• From the marker locations, determine rigid body positions and joint angles
  – and thus the character motion

http://www.hulu.com/watch/358363
Cameras

- Multiple cameras are set up to capture the performance space...
Motion Capture Stage

• Stages can be quite extensive...
Facial Motion Capture

- A special camera attached to the actors head is often used
Facial Motion Capture

- Extensive secondary effects, including simulation, can be incorporated on top of the motion capture