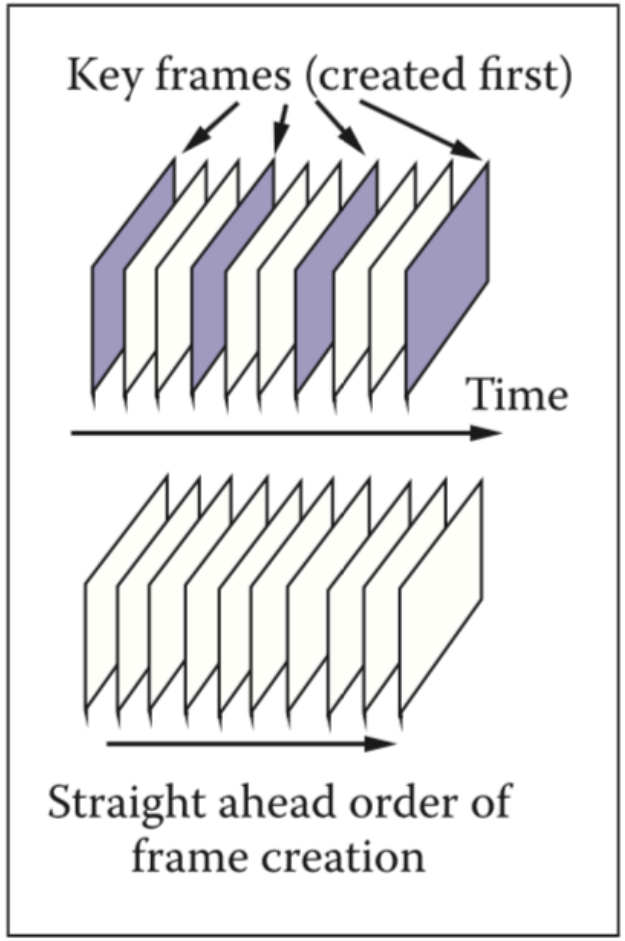
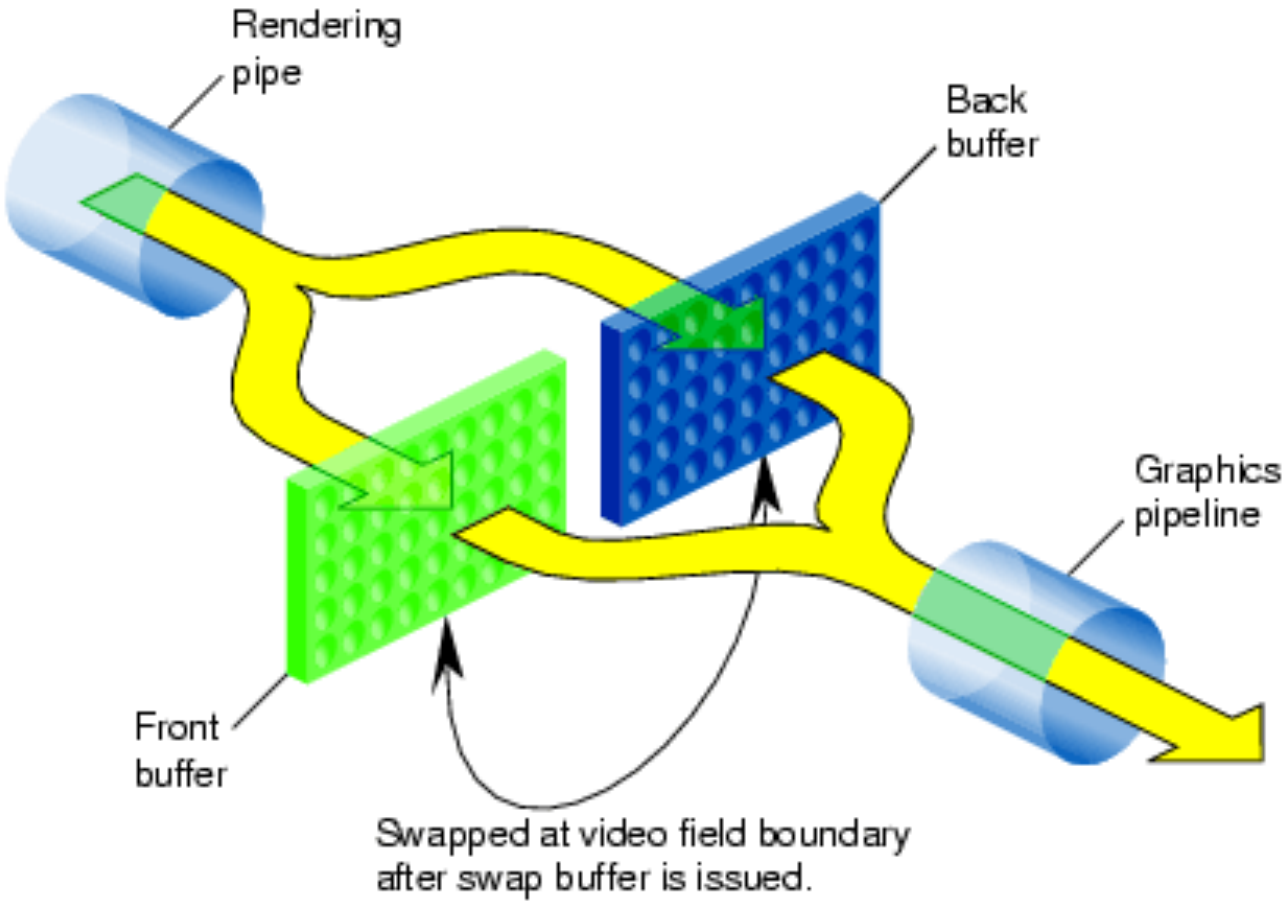
Keyframe Animation

- Idea: Draw a subset of important frames (called **key frames**) and fill in the rest with *in-betweens*
- In hand-drawn animation, the head animator would draw the poses and the assistants would do the rest
- In computer animation, the artist draws the keys and the computer does the in-betweening
    - Interpolation is used to fill in the rest!



Double Buffering

- If you draw directly to video buffer, the user will see the drawing happen
- Particularly noticeable artifacts when doing animation



http://techpubs.sgi.com/library/dynaweb_docs/linux/SGI_Developer/books/Perf_GetStarted/sgi_html/figures/double.buffering.gif

Controlling geometry conveniently

- Manually place every control point at every keyframe?
  - labor intensive
  - hard to get smooth, consistent motion
- Animate using smaller set of meaningful *degrees of freedom*
  - modeling DOFs are inappropriate for animation

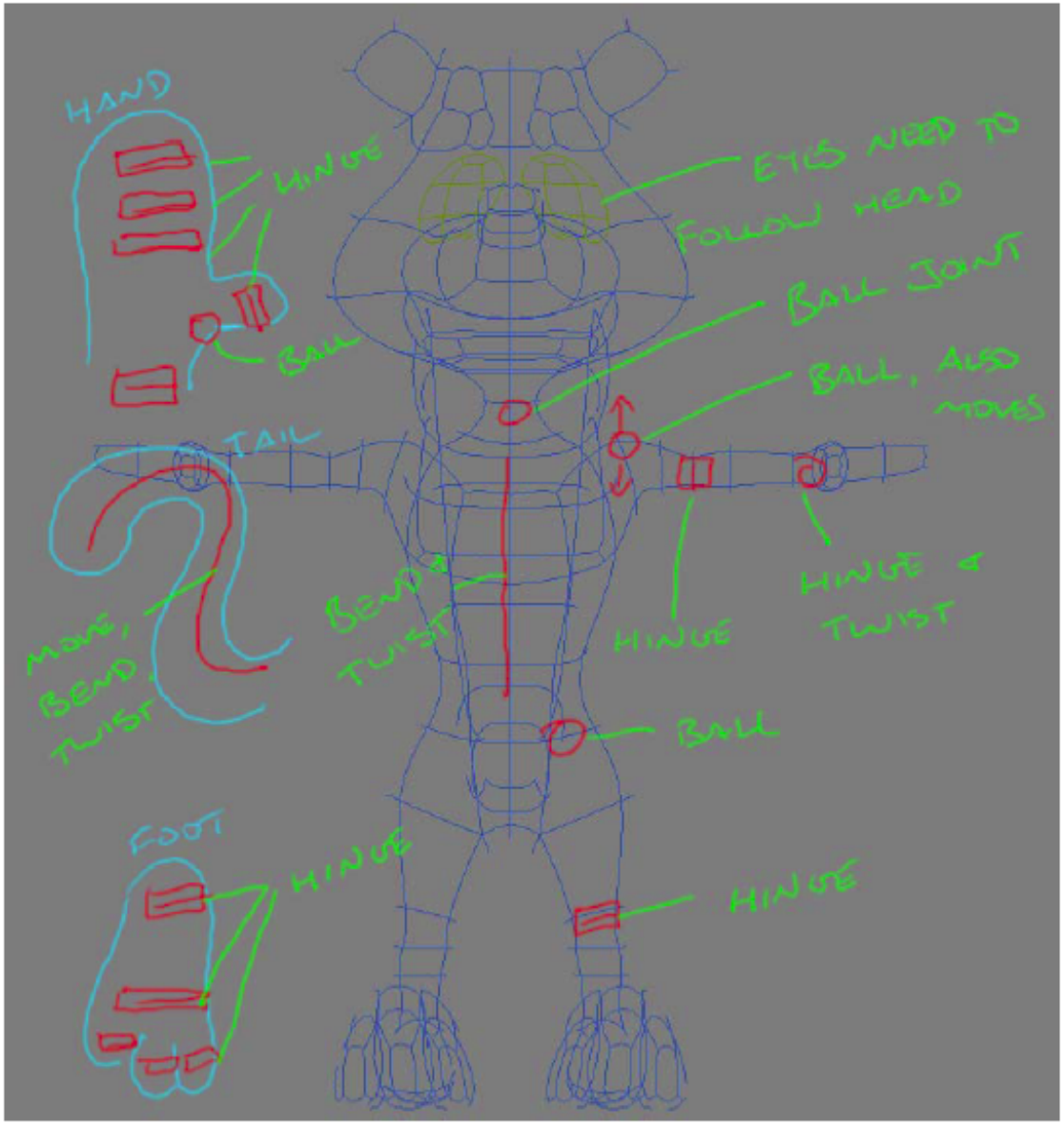
    e.g. "move one square inch of left forearm"
  - animation DOFs need to be higher level

    e.g. "bend the elbow"

Controlling shape for animation

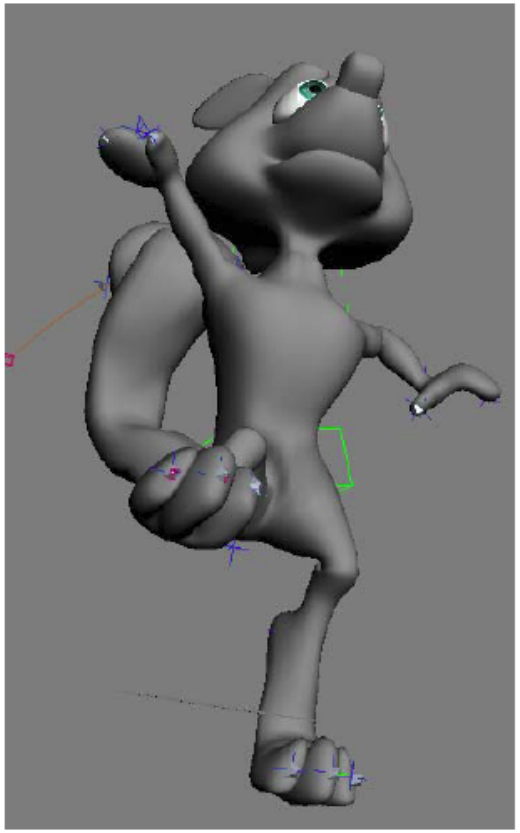
- Start with *modeling DOFs* (control points)
- *Deformations* control those DOFs at a higher level
  - Example: move first joint of second finger on left hand
- *Animation controls* control *those* DOFs at a higher level
  - Example: open/close left hand
- Both cases can be handled by the same kinds of deformers

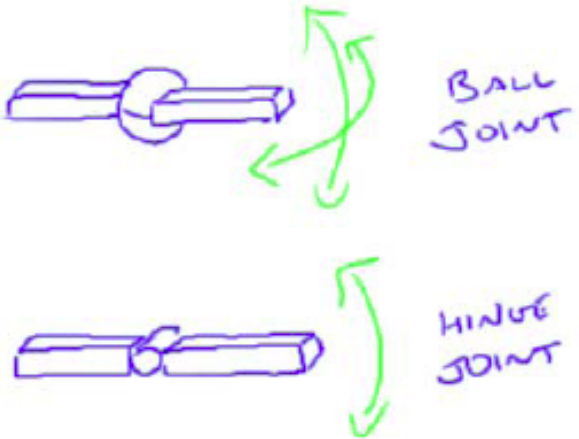
Character with DOFs



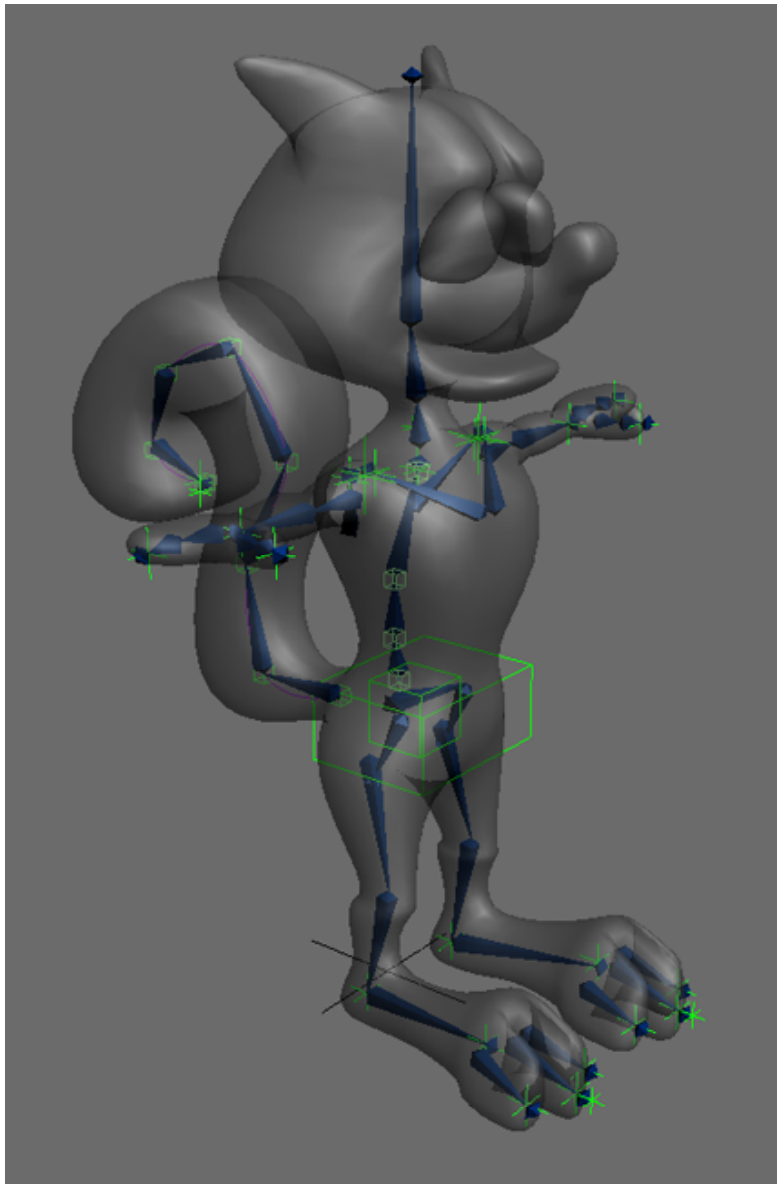
A visual description of the possible movements for the squirrel

[Greenberg/Pellacini | CIS 565]





Rigged character



[CIS 565 staff]

- Surface is deformed by a set of *bones*
- Bones are in turn controlled by a smaller set of *controls*
- The controls are useful, intuitive DOFs for an animator to use

Interpolating Rotations

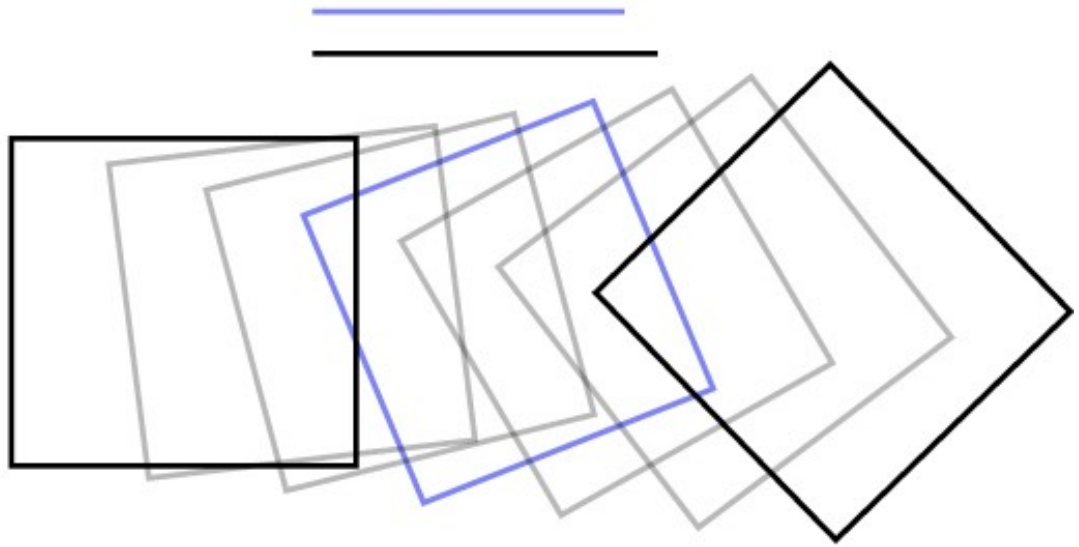
The most basic animation control

- Affine transformations position things in modeling
- Time-varying affine transformations move things around in animation
- A hierarchy of time-varying transformations is the main workhorse of animation
  - and the basic framework within which all the more sophisticated techniques are built

Interpolating transformations

- Move a set of points by applying an affine transformation
- How to animate the transformation over time?
  - interpolate the matrix entries from keyframe to keyframe?

    this is fine for translations but bad for rotations

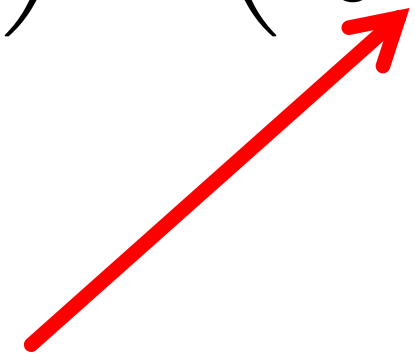


Interpolating Rotations

$$\frac{1}{2}\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

90° CW

90° CCW



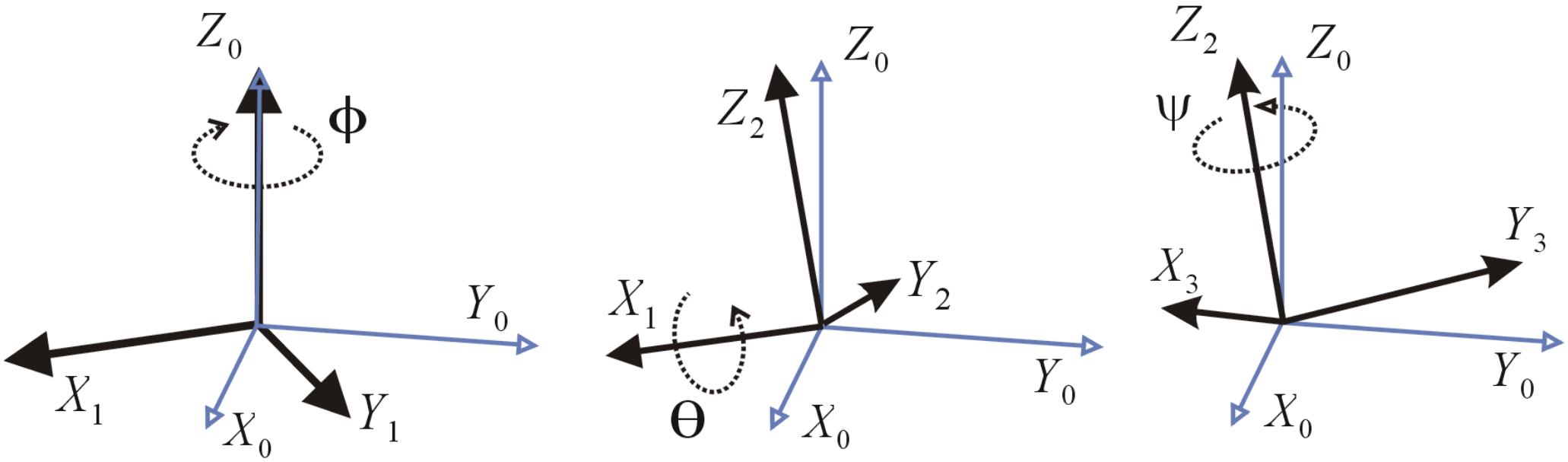
Not a rotation matrix!

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

Issues occurs when the source and target angles are not close to each other

Could Instead Decompose Rotation by Euler Angles

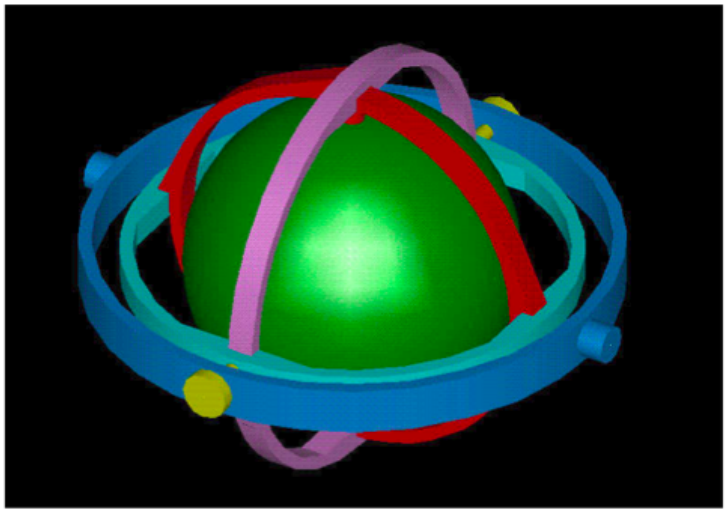


http://upload.wikimedia.org/wikipedia/commons/7/73/EulerG.png

Parameterizing rotations

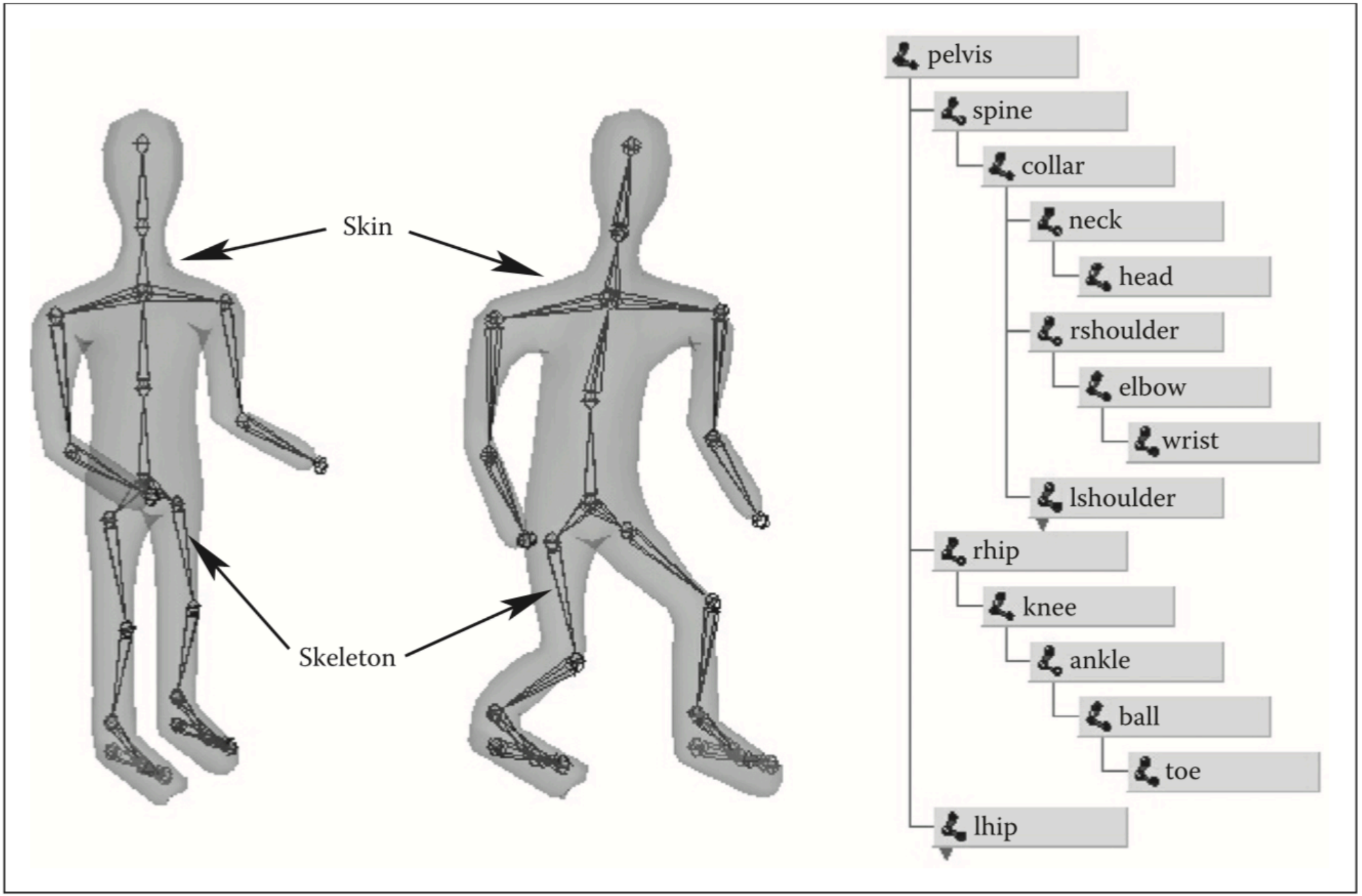
- Euler angles
  - rotate around x, then y, then z
  - nice and simple

$$R(\theta_x, \theta_y, \theta_z) = R_z(\theta_z) R_y(\theta_y) R_x(\theta_x)$$

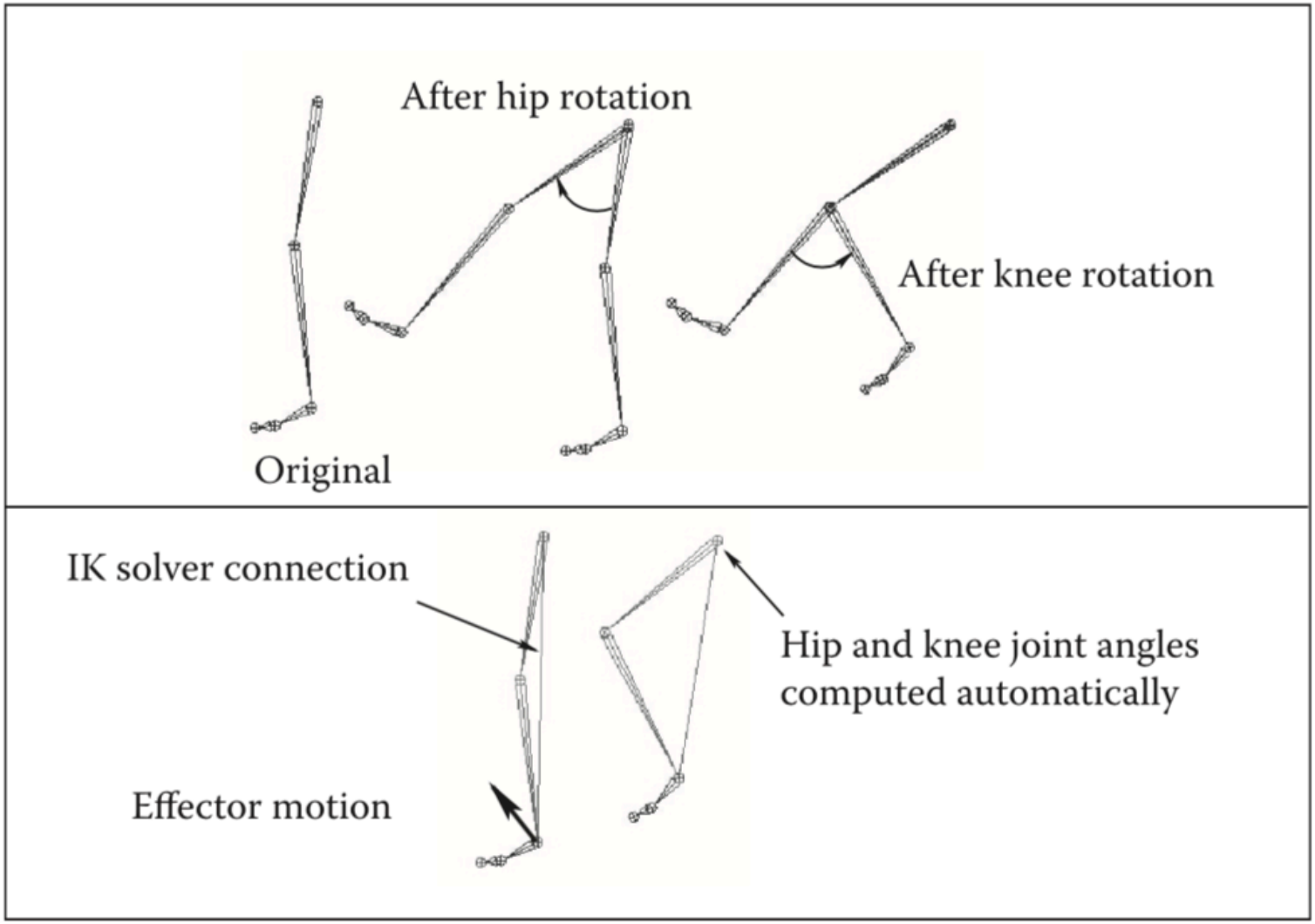


Character Animation

Animating w/ Skeletal Hierarchies



Forward vs. Inverse Kinematics



Inverse Kinematics Solves for all Intermediate Constraints