Animation

Overview

- Animation principles
- Keyframing and interpolation
- Kinematics (joints, articulated figures)
- Data driven
  - Motion capture
- Procedural animation
  - Rules and simulations automatically produce the animation
  - Physics simulation

Applications

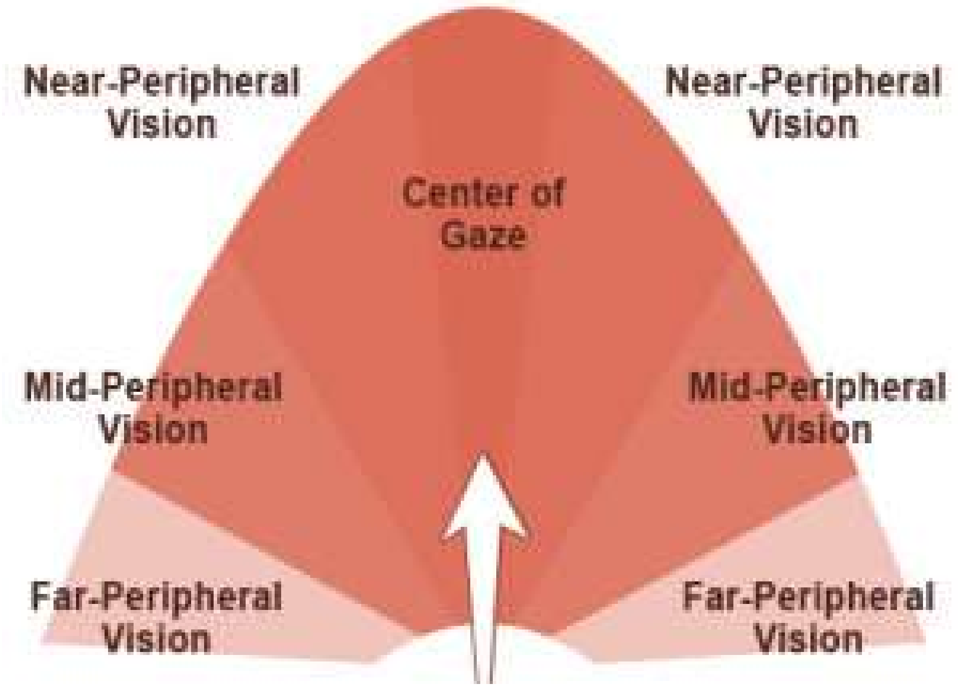
- Special Effects
  - Movies, TV
- Video Games
- Virtual Reality
- Simulation, Training
- Medical
- Robotics, Animatronics
- Visualization
- Communication

Animation Tools

- Maya
- 3D Studio
- Lightwave
- Filmbox
- Blender
- …

Animation Basics

- Persistence of vision (human eye)
  - Afterimage for 1/25 sec (compensate blinking)
  - Sequence of images shown fast enough is perceived as true motion
    - What is fast enough?
    - Foveal vs. Peripheral vision
- Frame rate (fps = frames per second)
  - Film: 24
  - TV (Computers): 60 or more



Motion Blur

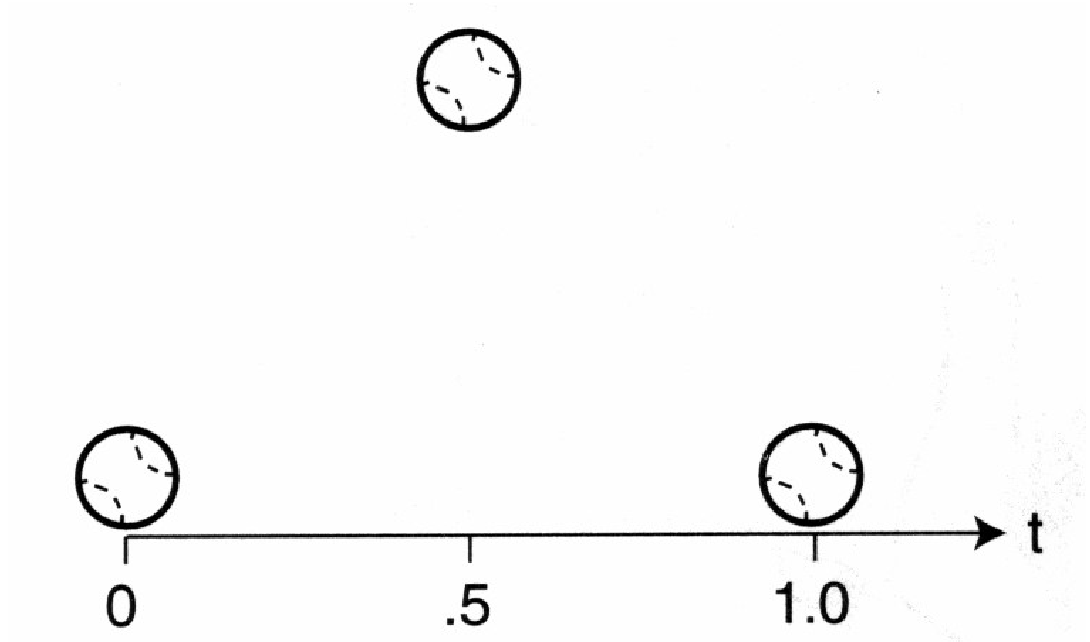
- Light persists in our vision for a while
  - Fast moving objects leave a blurred streak
- Real cameras leave shutter open for a while
  - Moving objects blurred
- Without motion blur
  - Trail of clear staggered images

Computer Animation

- Setting various animation parameters
  - Positions, angles, sizes, … in each frame
- Straight-ahead
  - Set all variables in frame 0, then frame 1, frame 2, …
- Pose-to-pose:
  - Set variables at keyframes, let the computer smoothly interpolate values in between
- Can mix the methods

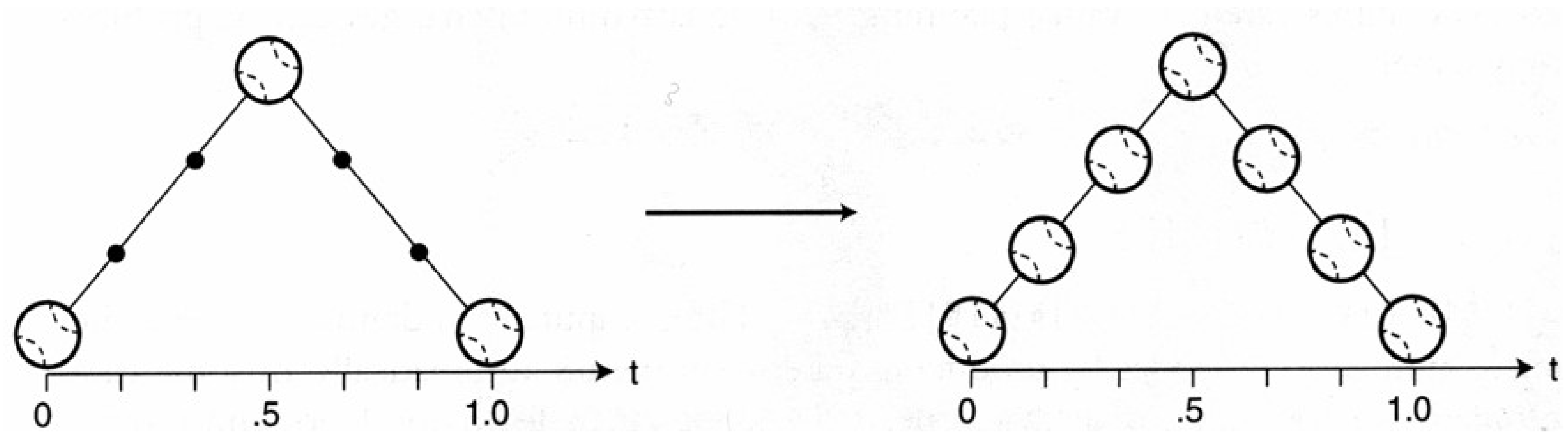
Keyframing

- Traditional animation technique
- Dependent on artist to generate 'key' frames
- Additional, 'inbetween' frames are drawn automatically by computer



Linear Interpolation

- Simple
- Constant velocity at edges
- Discontinuous velocity at corners
  - Can be unnatural

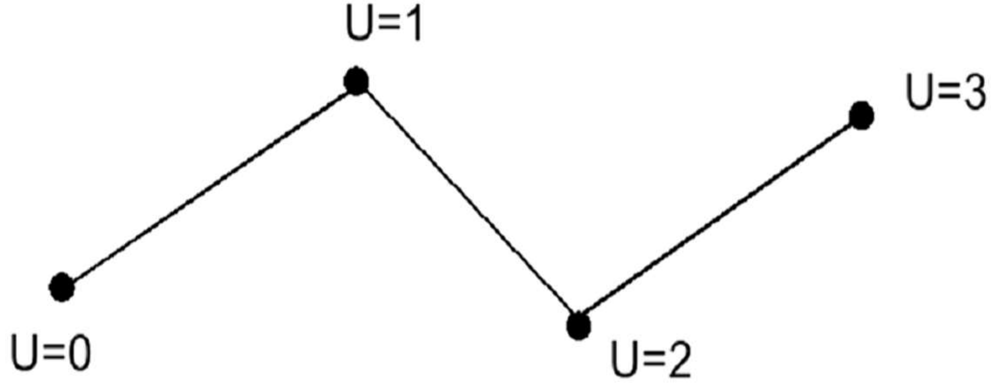


Linear Interpolation

- Given points P_0 and P_1, define curve $L(t)$:
  $L(t) = (1-t)\,P_0 + t\,P_1 \qquad t \text{ in } [0,1]$
- Weighted average of endpoints
- “Curve” is linear segment

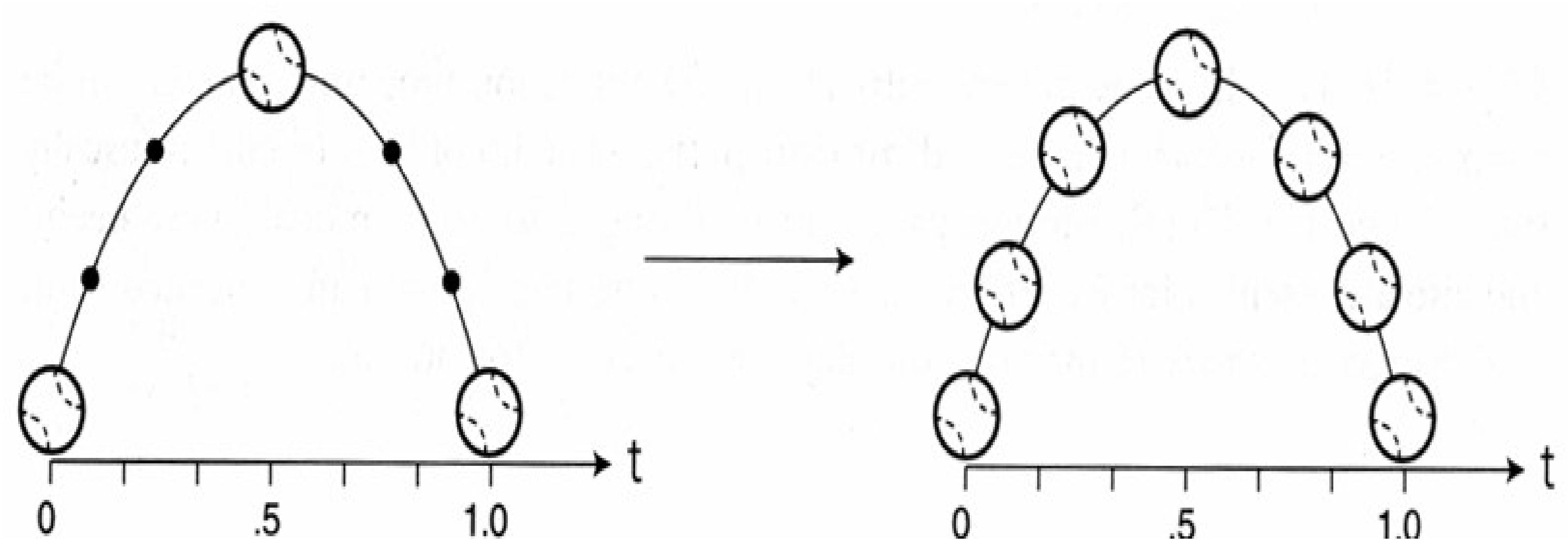
Linear Interpolation: *N* Points

- Given $P_0 \ldots P_N$ define segment:
  $L_i(s) = (1-s)\, P_i + s\, P_{i+1}$ $\quad$ s in $[0,1]$
- Then define piecewise-linear curve:
  $L(u) = L_i(fract(u))$
- Sharp discontinuities C^0 but not C^1



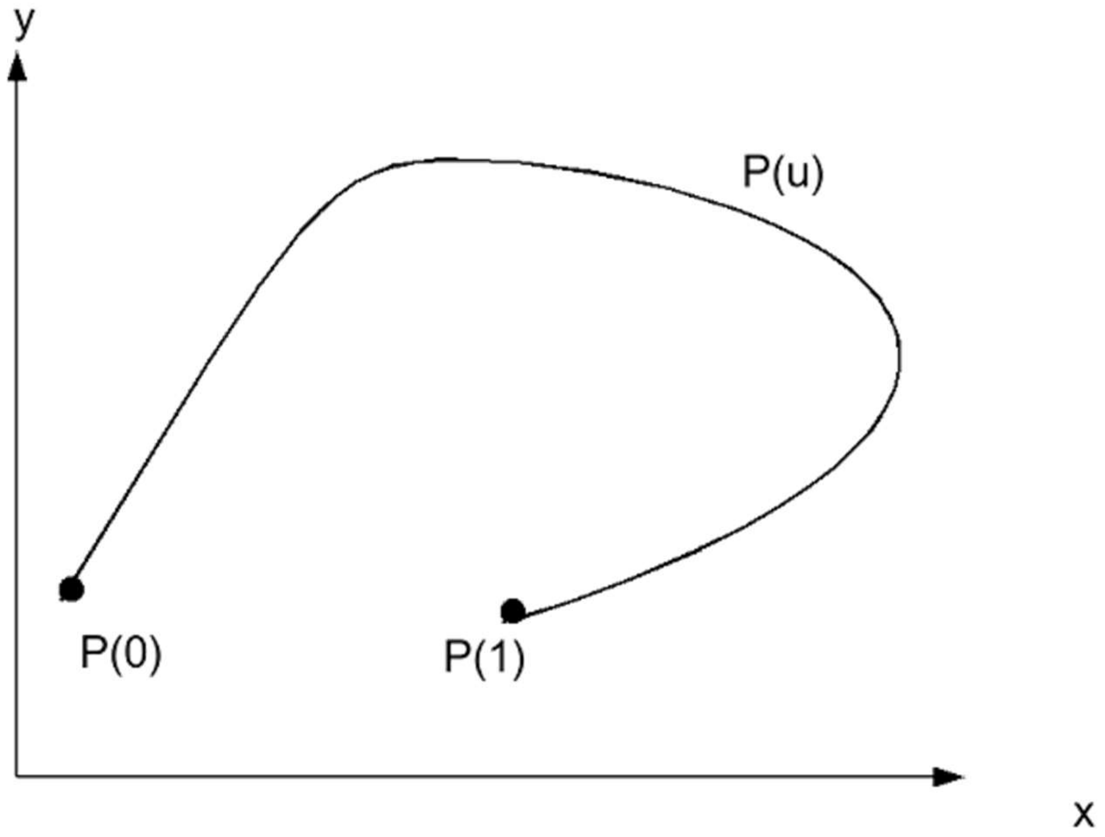
Nonlinear Interpolation

- Often Catmull-Rom-Splines
- Removes discontinuities at corners
  - C_0, C_1
  - C_2 often unnecessary (acceleration often changes discontinuous in nature)



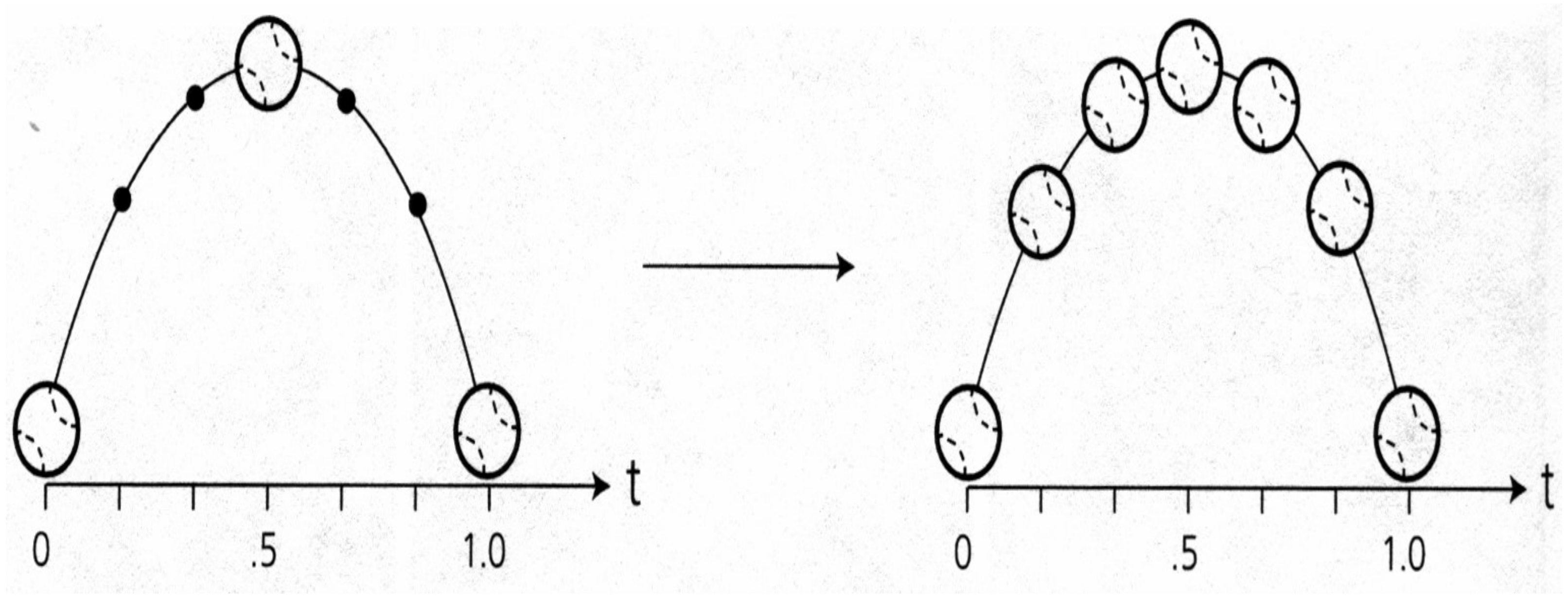
Parametric Curves

- $P(u) = (x(u), y(u), z(u)) \qquad 0 \le u \le 1$



Easing

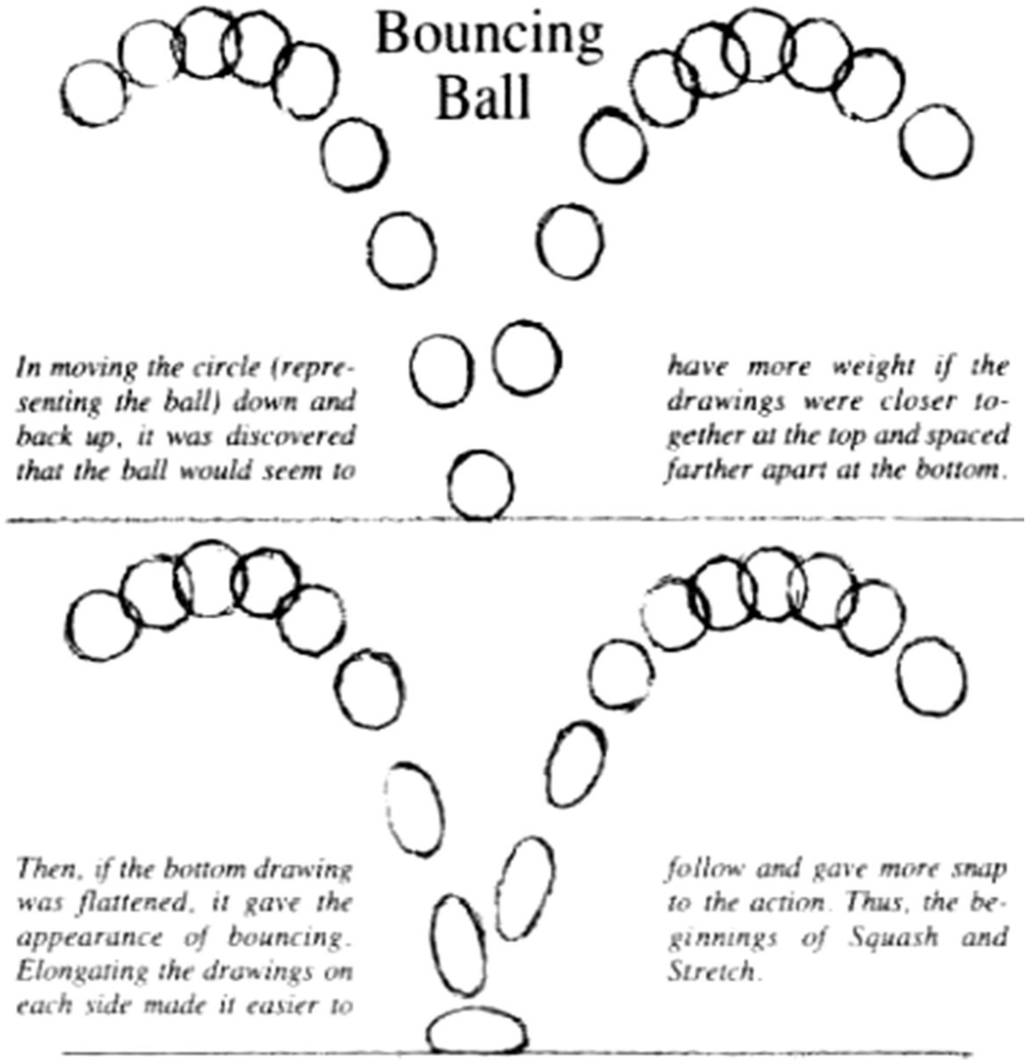
- Velocity is changed near a keyframe
  - Ball should slow down at apex



Style or Accuracy

- More weight

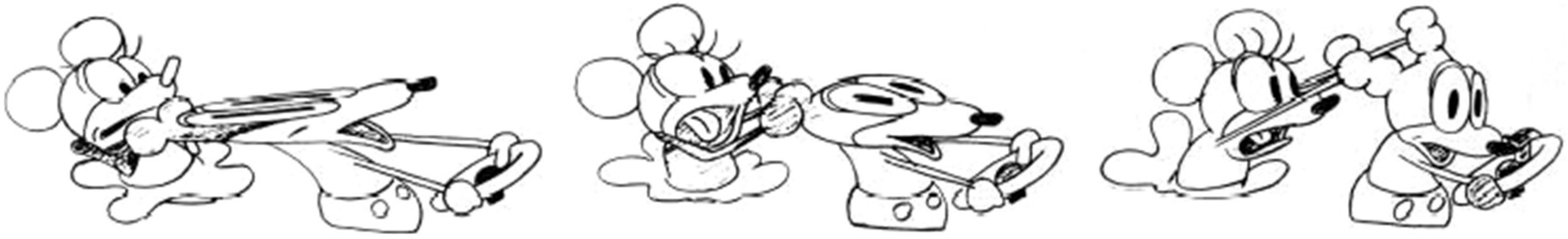
- Squash and stretch

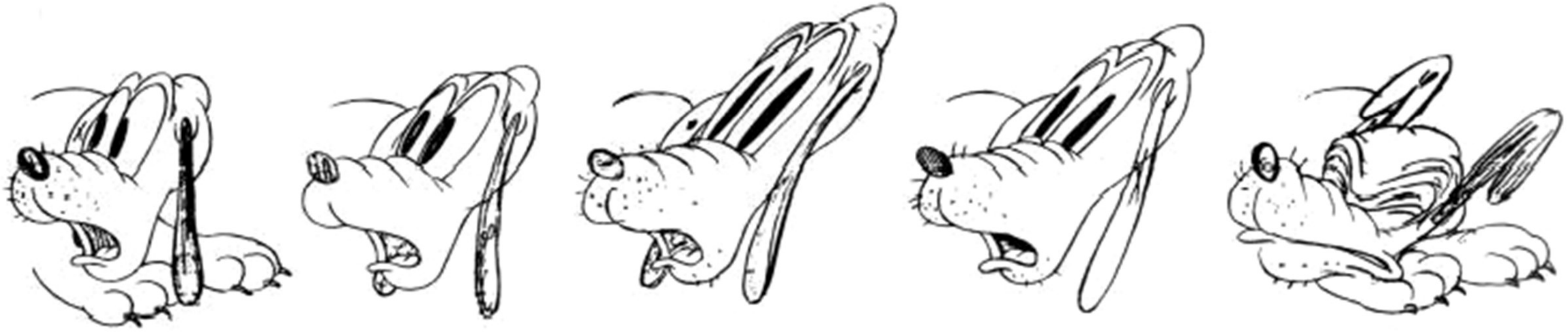


In moving the circle (representing the ball) down and back up, it was discovered that the ball would seem to have more weight if the drawings were closer together at the top and spaced farther apart at the bottom.

Then, if the bottom drawing was flattened, it gave the appearance of bouncing. Elongating the drawings on each side made it easier to follow and gave more snap to the action. Thus, the beginnings of Squash and Stretch.

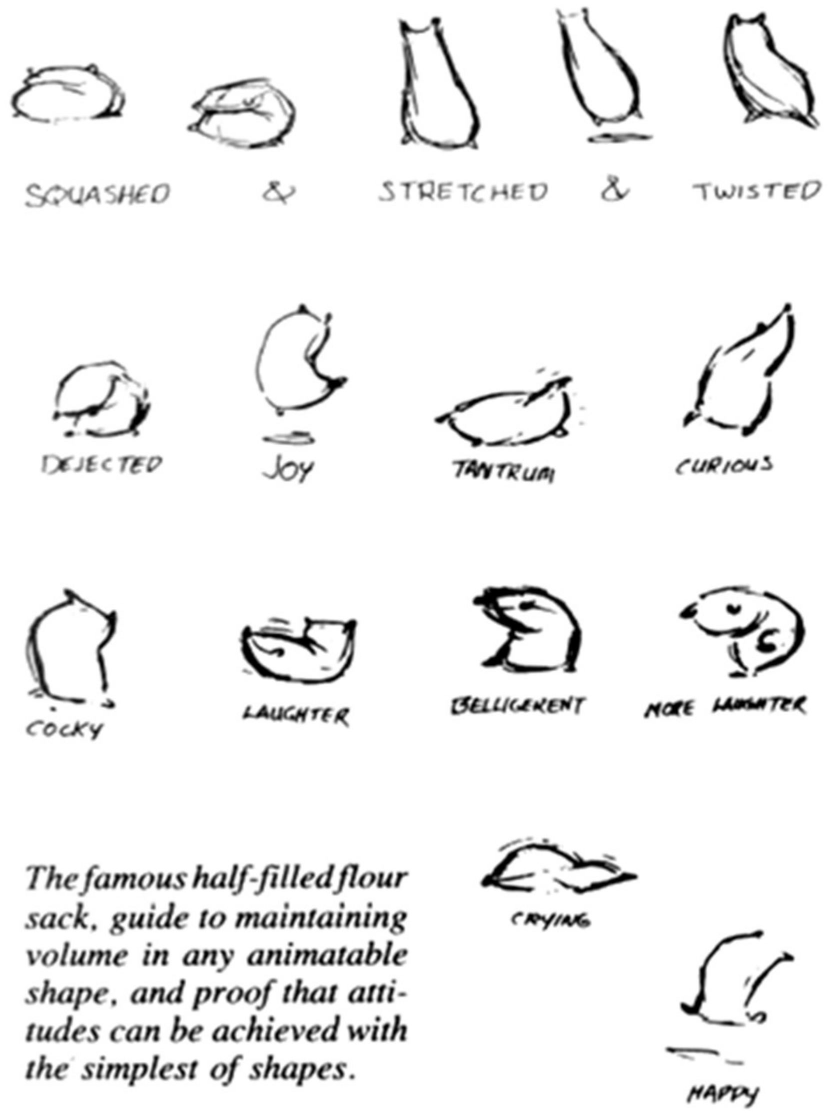
More Squash and Stretch





Traditional Motivation

- A keyframe should contain all informationto understand a situation



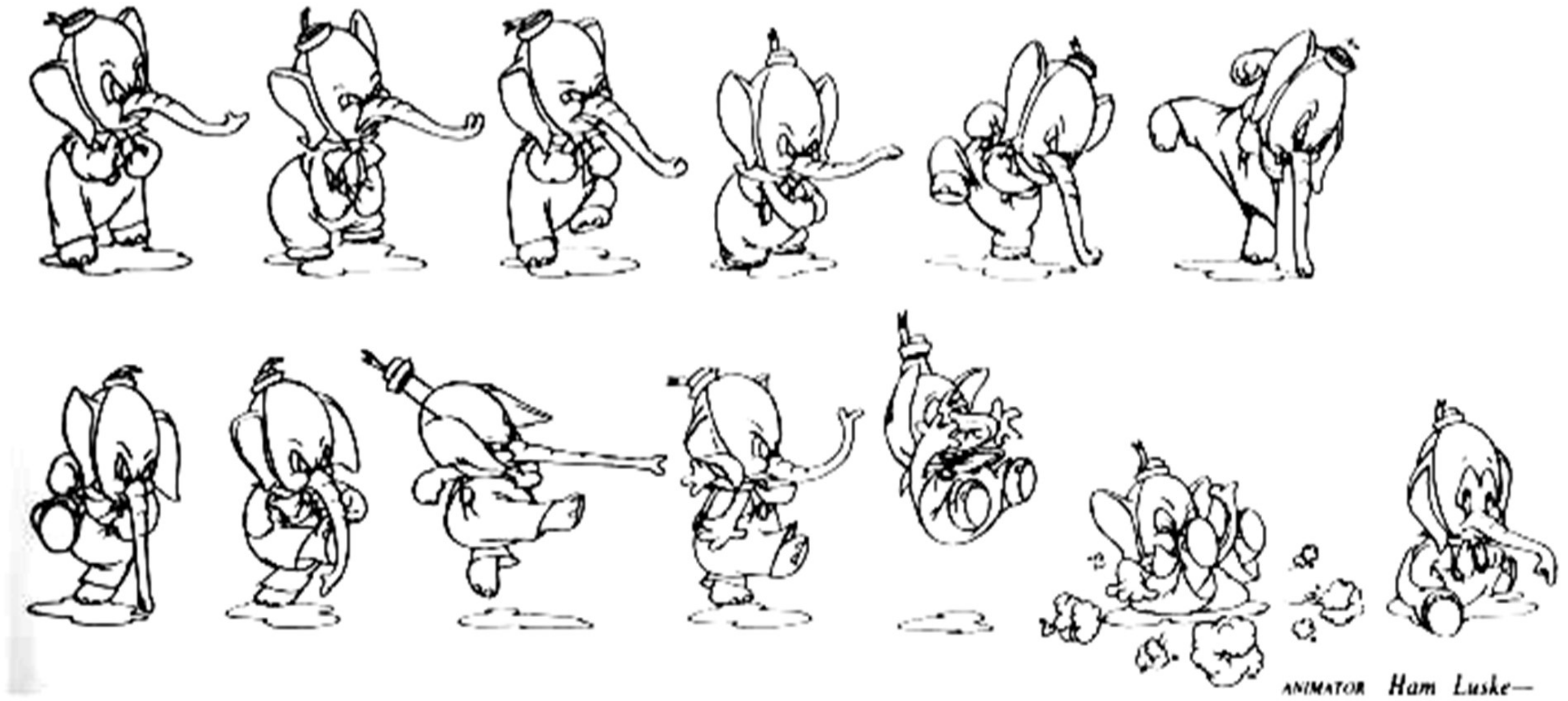
The famous half-filled flour sack, guide to maintaining volume in any animatable shape, and proof that attitudes can be achieved with the simplest of shapes.

Anticipation and Staging

- Don’t surprise the audience
- Direct their attention to what’s important

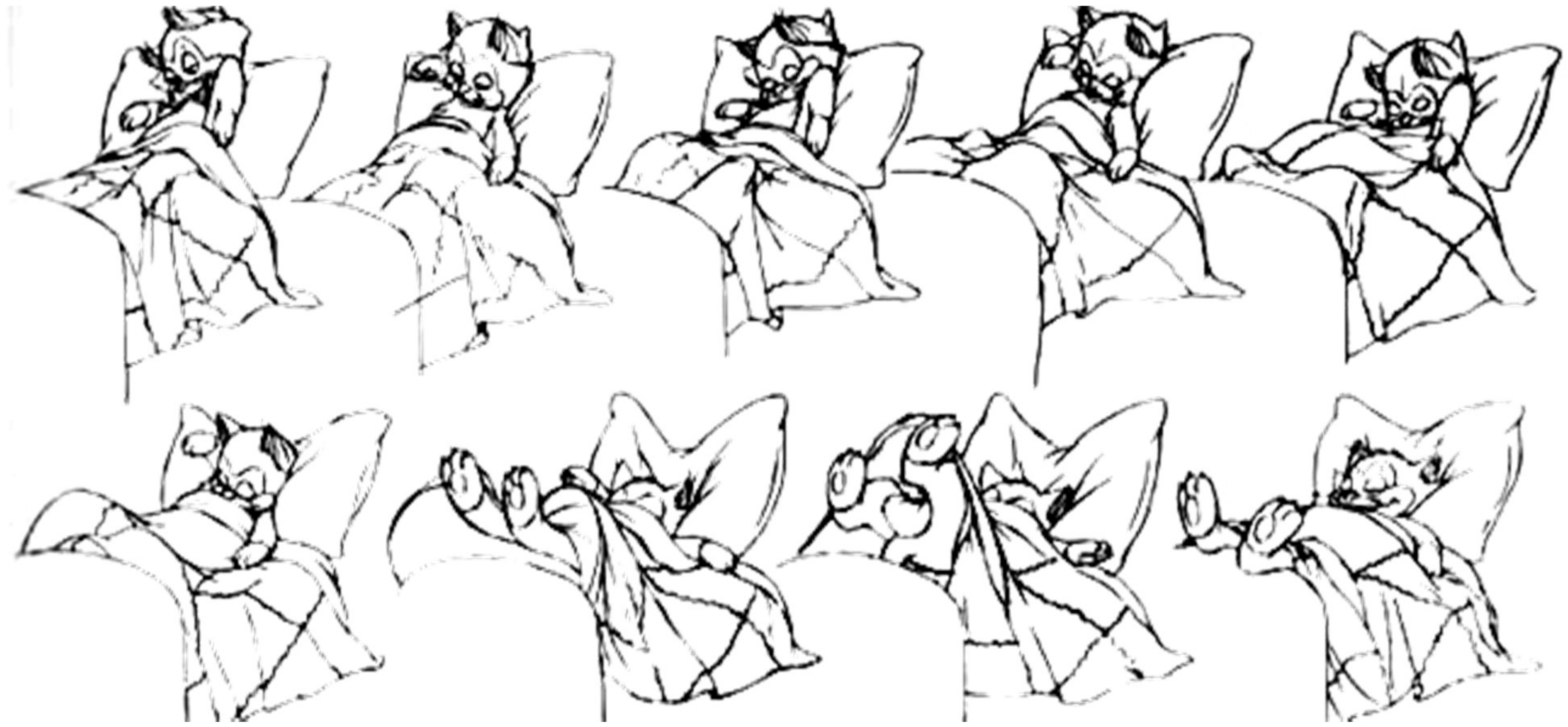
Follow Through

- Audience likes to see resolution of action
- Discontinuities are unsettling



Secondary Motion

- Characters should exist in a real environment
- Extra movements should not detract

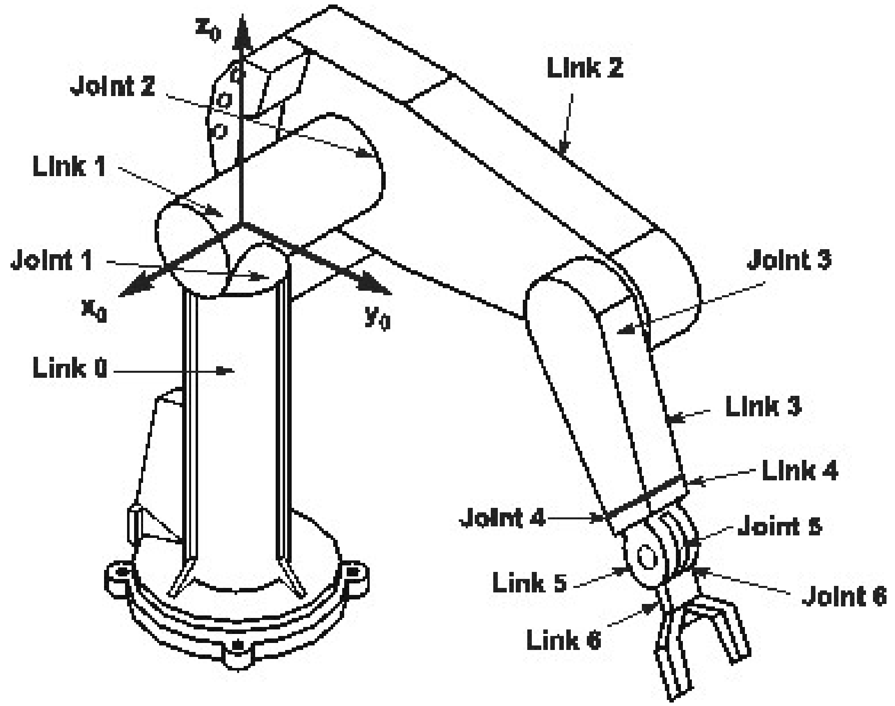


Animation

Kinematics

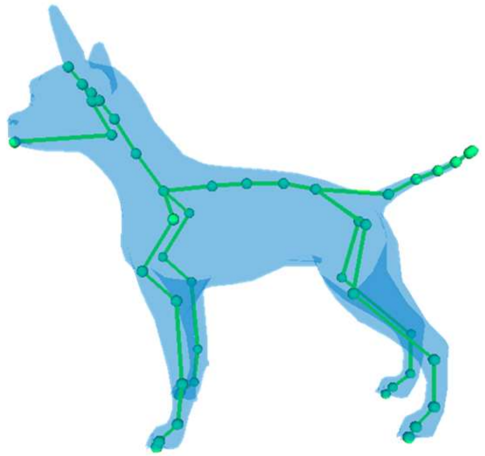
Kinematics

- The study of how things move (without considering the causes)
- Describing the motion of articulated rigid figures
  - Things made up of rigid "links"
  - Attached to each other at movable joints (articulation)
- Many mathematical approaches

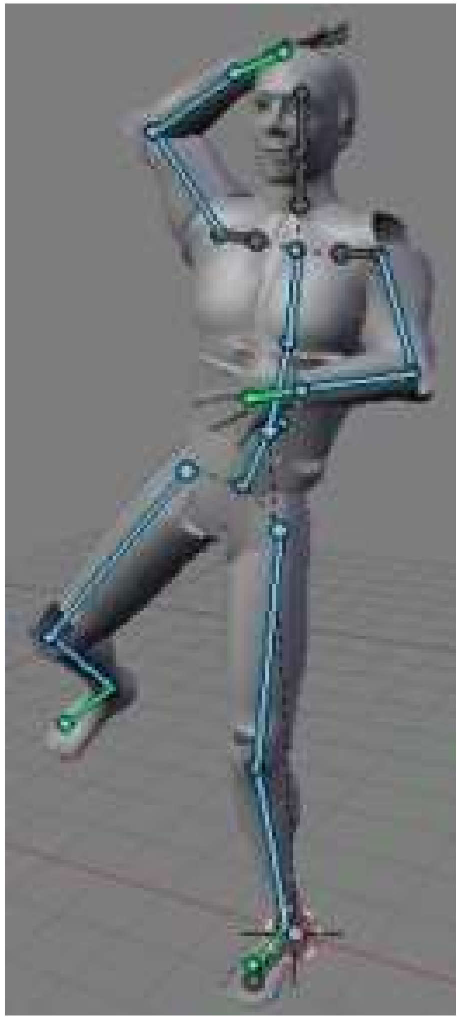


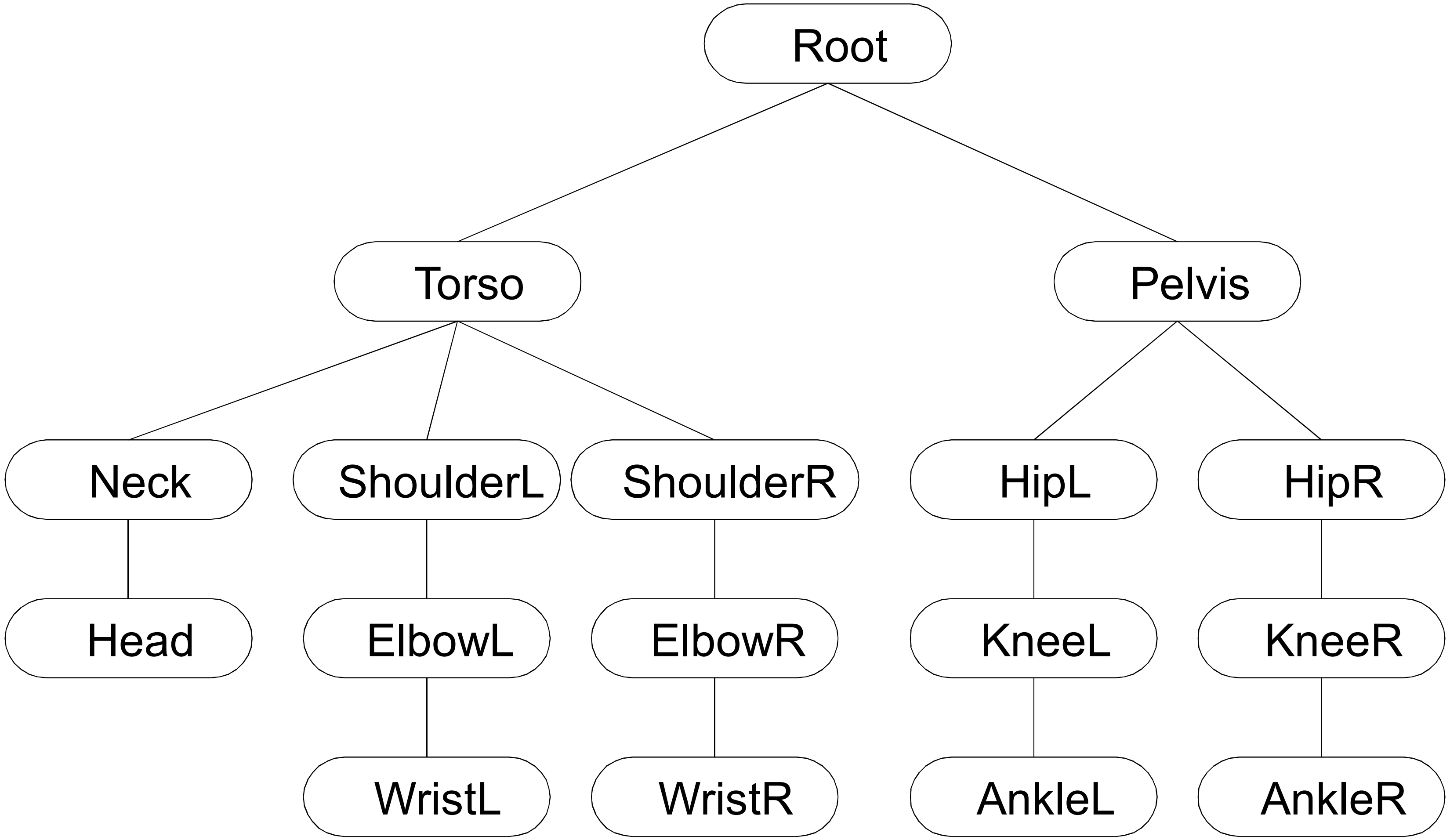
Skeletons

- Skeleton
  - Pose-able framework of joints
  - Arranged in a tree structure
  - Invisible armature
- Joint (Bone)
  - Allows relative movement
  - Essentially 4x4 matrix transformations
  - Rotational, translational, …
- Rigging = building the skeleton



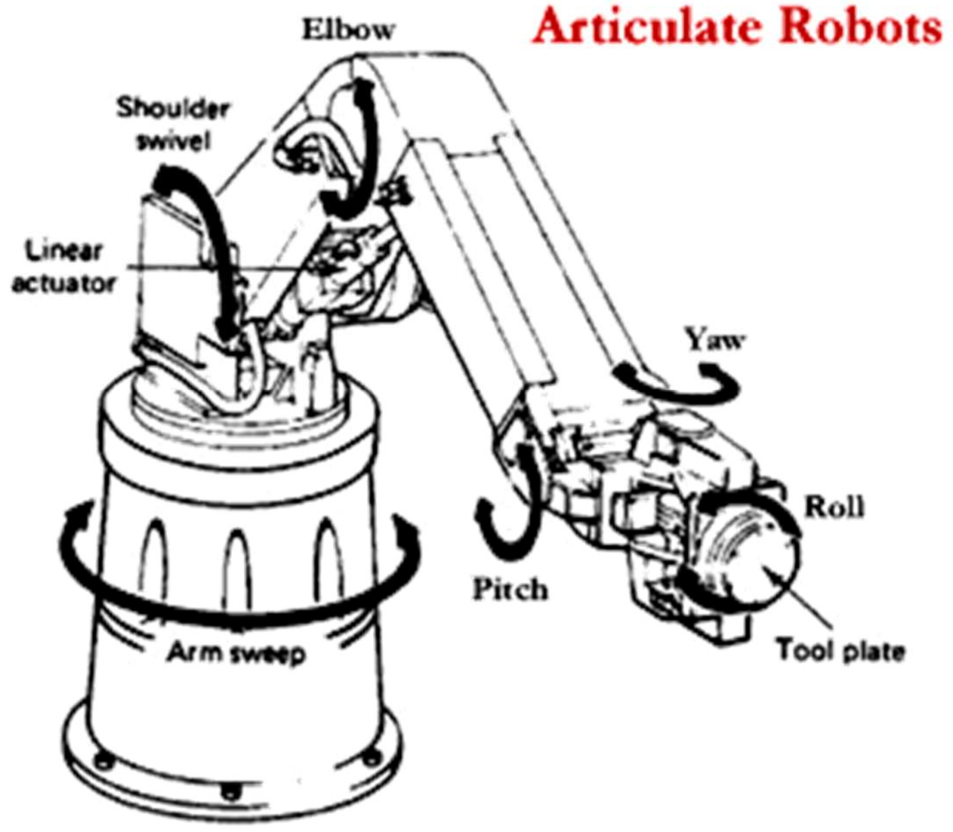
Example Joint Hierarchy





Degree of Freedom (DOF)

- A variable ϕ describing a particular axis or dimension of movement within a joint
- Joints typically have around 1-6 DOFs ($\phi_1 \ldots \phi_N$)

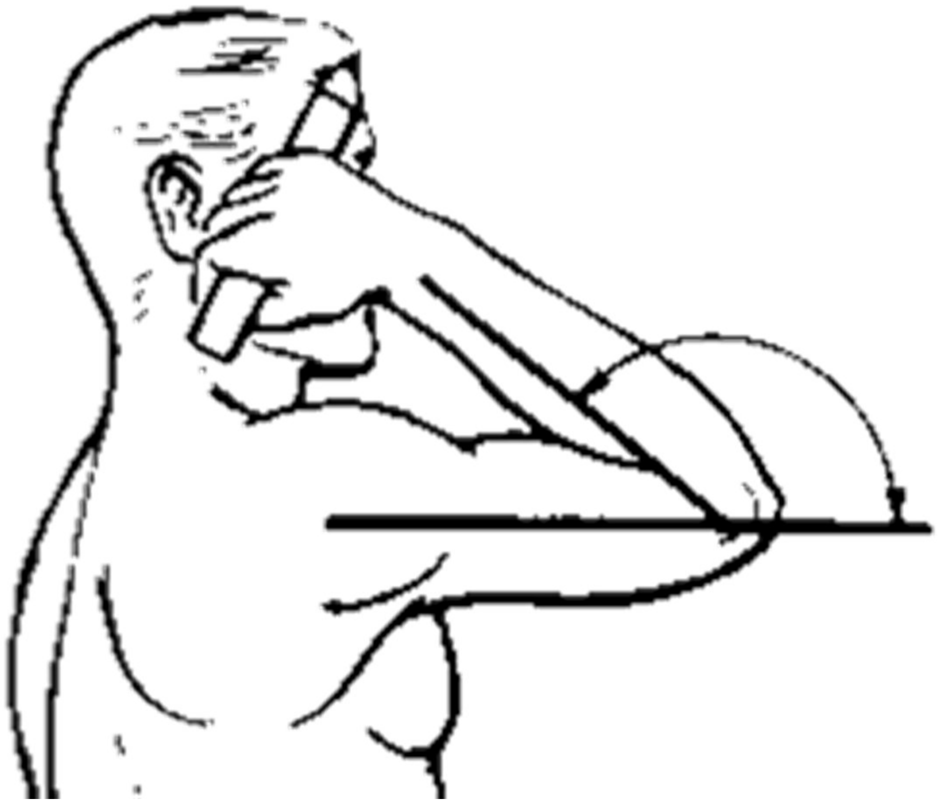


Degree of Freedom (DOF)

- Changing DOF values over time results in animation of skeleton
- Example
  - Free rigid body has 6 DOFs
    - 3 for position
    - 3 for rotation

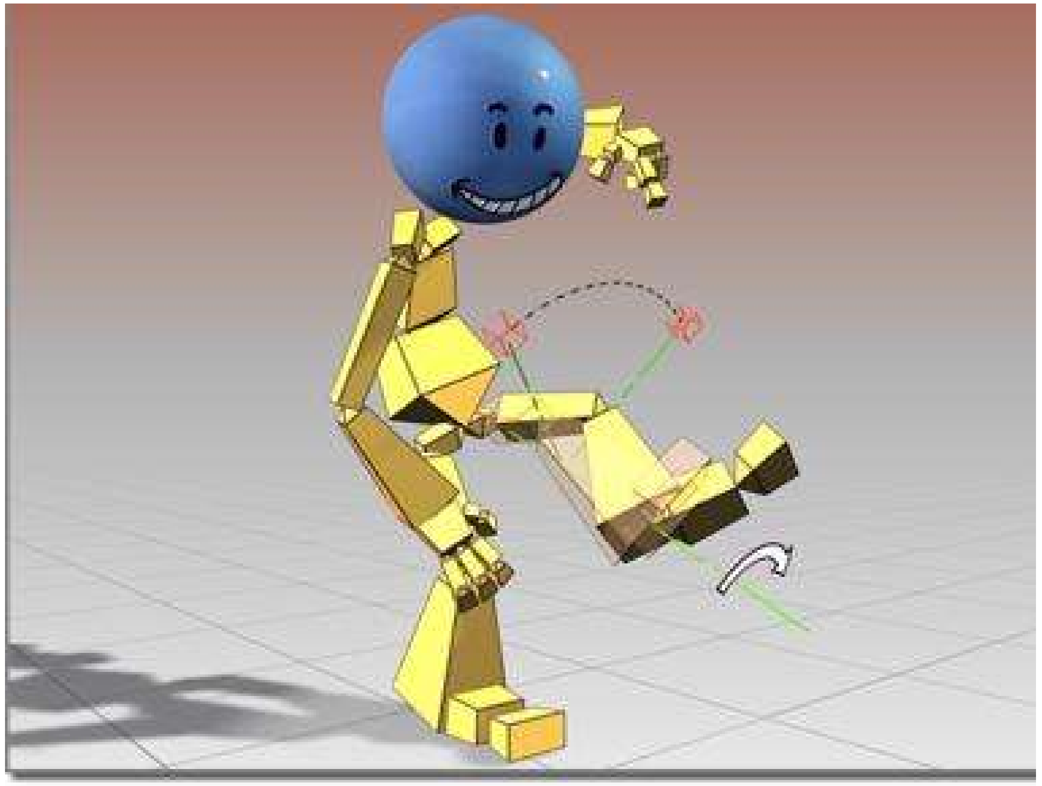
DOF Limits

- Limit DOF to some range
  - E.x. limit elbow from 0º to 150º
- Realistic character
  - All DOFs will be limited
  - Except ones controlling root



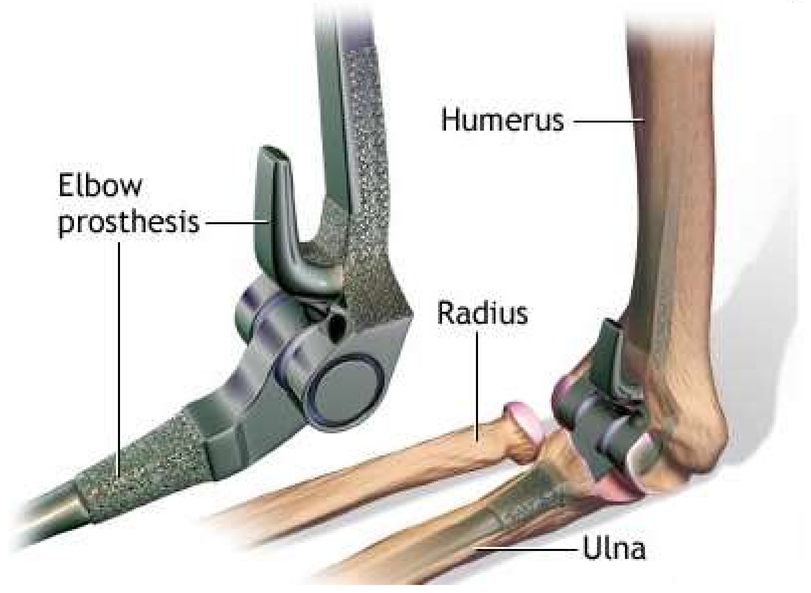
Joints

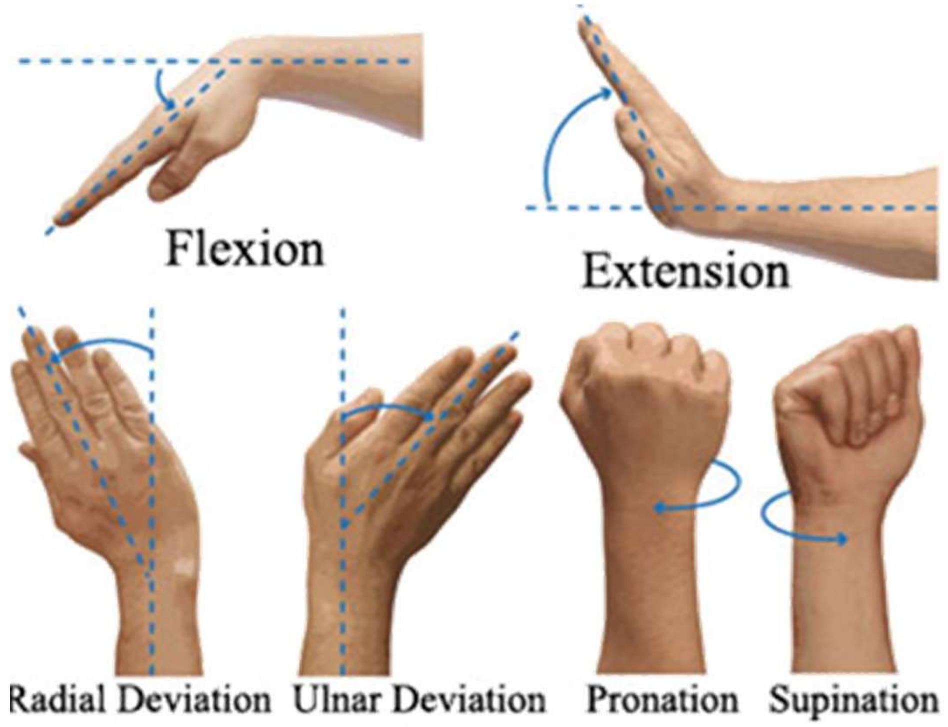
- Core joint data
  - DOFs (**n** floats)
  - Local matrix: **L**
  - World matrix: **W**
- Additional Data
  - DOF limits (min/max value)
  - Type-specific data (rotation/translation axes, constants…)
  - Tree data (pointers to children, siblings, parent…)



Reality Check

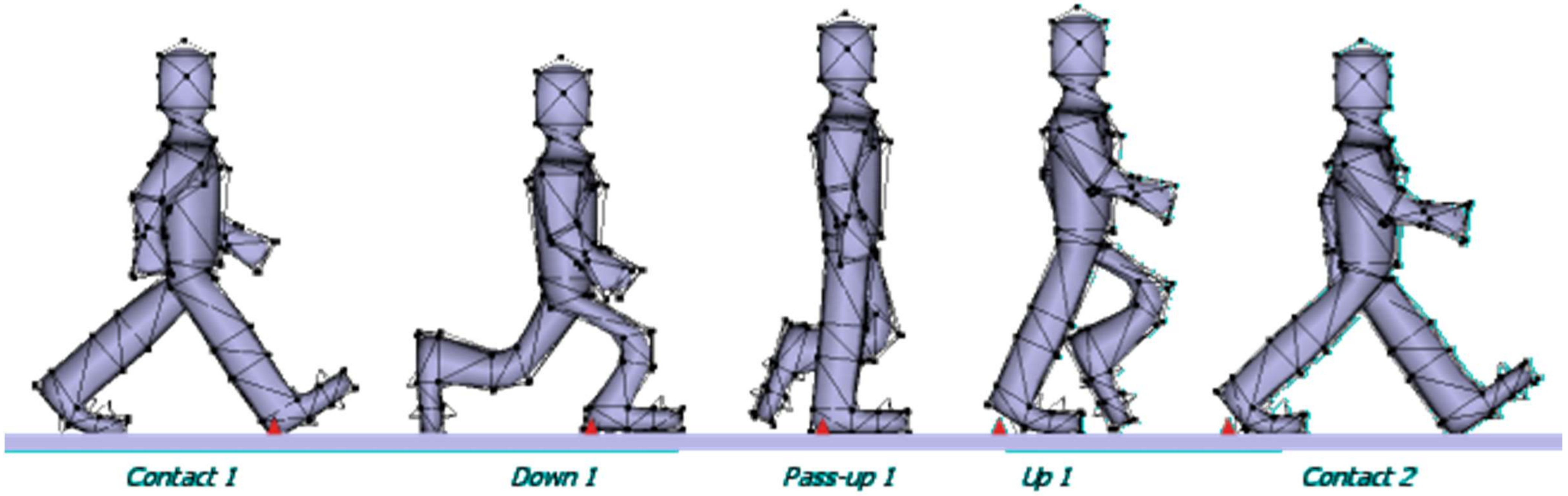
- Elbow has 1 DOF: the angle the forearm makes with the upper arm (rotation in plane)
- Wrist has 3 DOF
- …



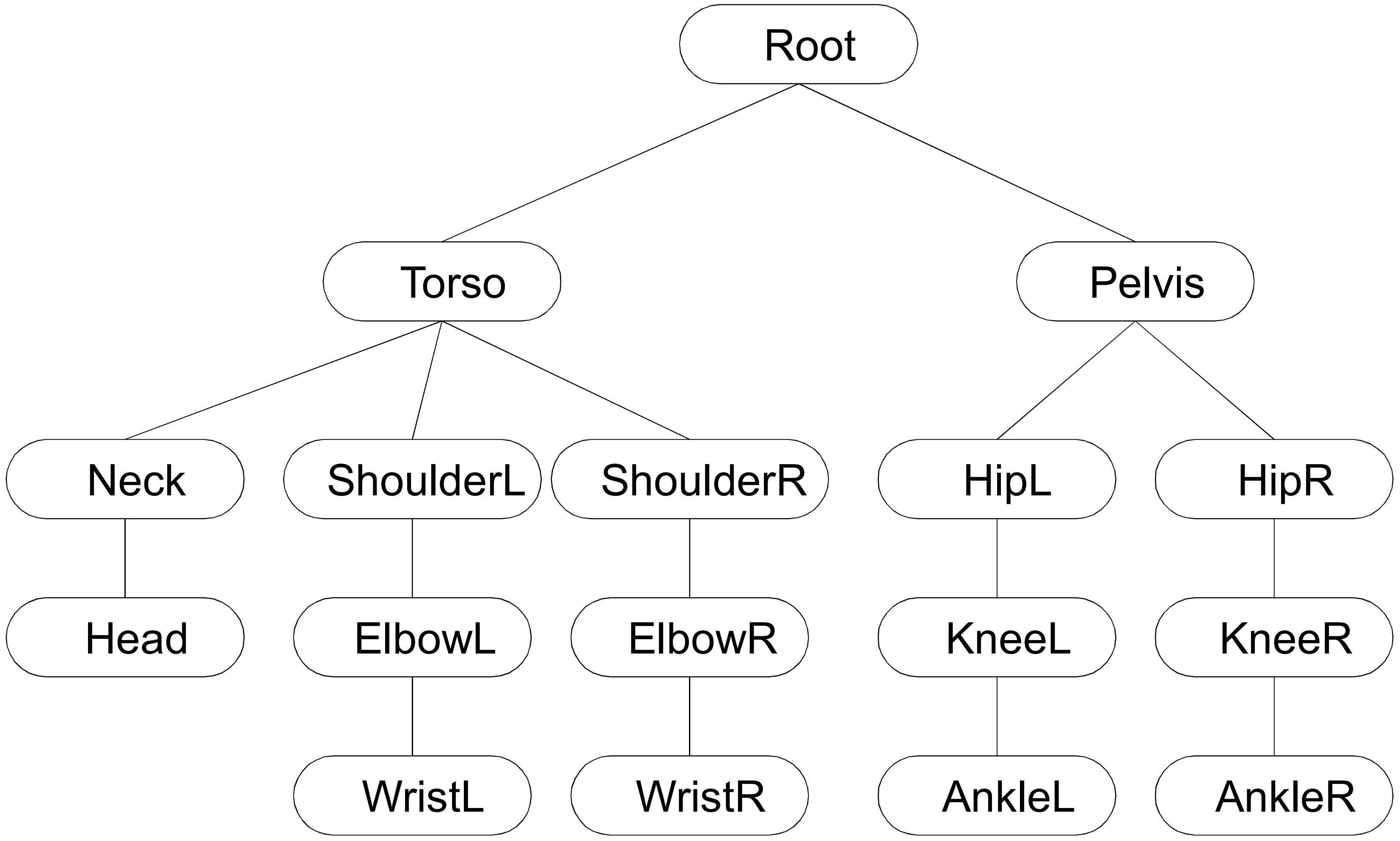


Pose

- Setting of all DOFs (specify in Blender, Maya)
  $\Phi = (\phi_1\ \phi_2\ \ldots\ \phi_N)$



Forward Kinematics



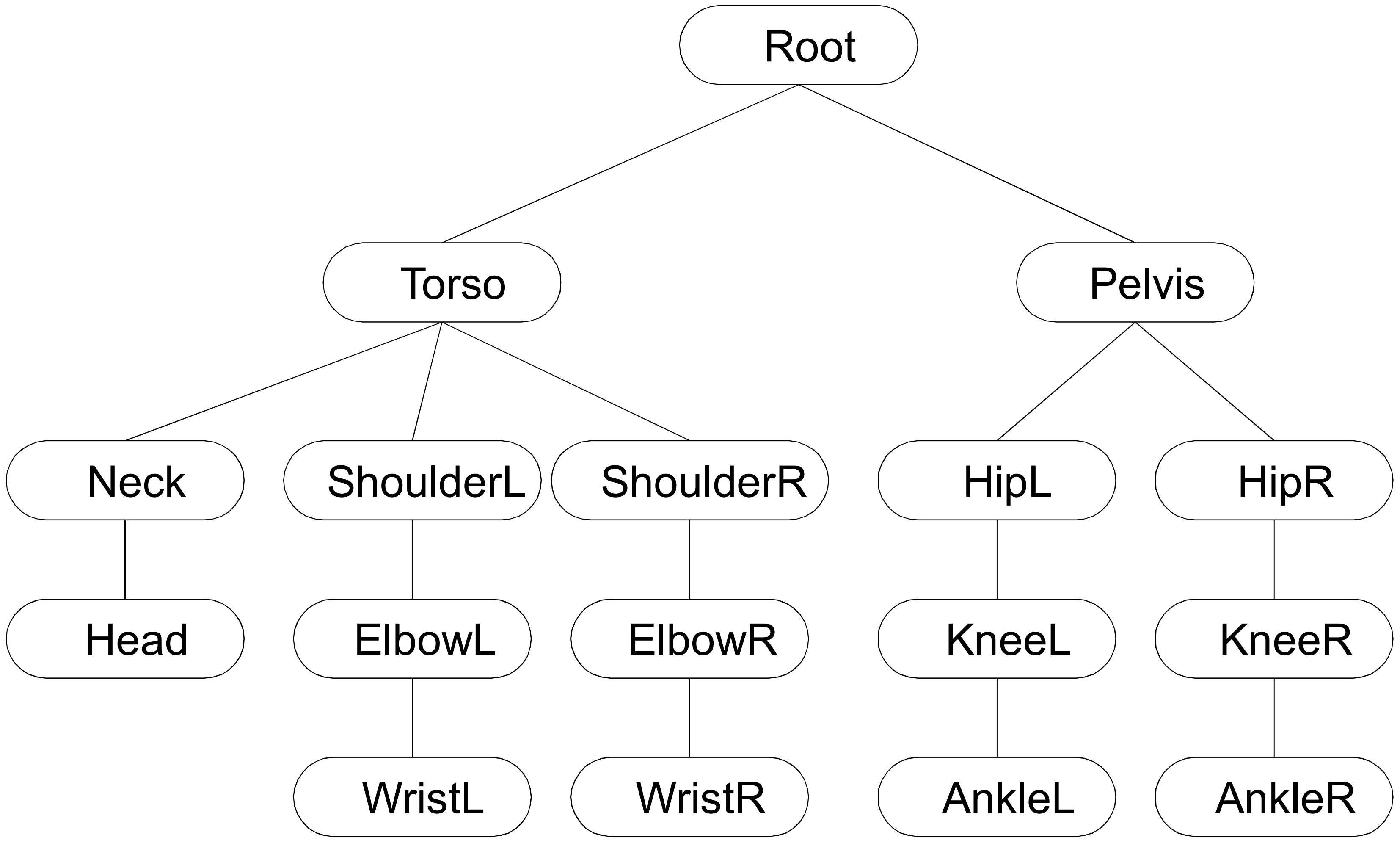
Forward Kinematics

- Top down recursive tree traversal
- Each joint computes local matrix **L**
  - Based on the values of DOFs (ϕ_1 ϕ_2 … ϕ_N)
  - And joint type (rotational, translational, …)
  - Local matrix $\mathbf{L} = \mathbf{L}_{joint}(\phi_1,\phi_2,\ldots,\phi_N)$
- Each joint computes world matrix **W**
  - Multiply **L** with world matrix of parent joint
  - World matrix $\mathbf{W} = \mathbf{W}_{parent} \cdot \mathbf{L}$

Pros and Cons

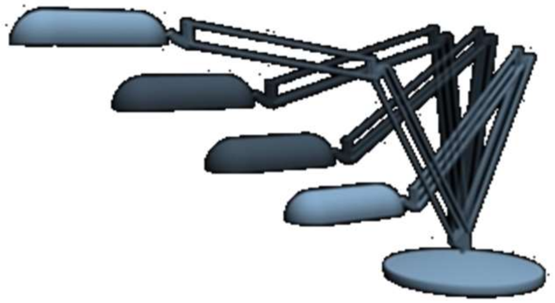
- A simple layered approach
  - Get the root link moving first (e.g. the pelvis)
  - Fix the pose outward, link by link (the back and the legs next, then the head, the arms, the hands, the fingers, ...)
- Great for certain types of motion
  - General acting, moving in free space, ...
- Problems when interacting with other objects
  - Fingers grabbing a doorknob
  - Feet stay on the ground

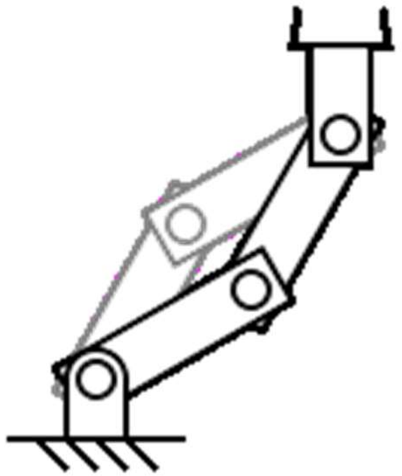
Inverse Kinematics (IK)



Inverse Kinematics (IK)

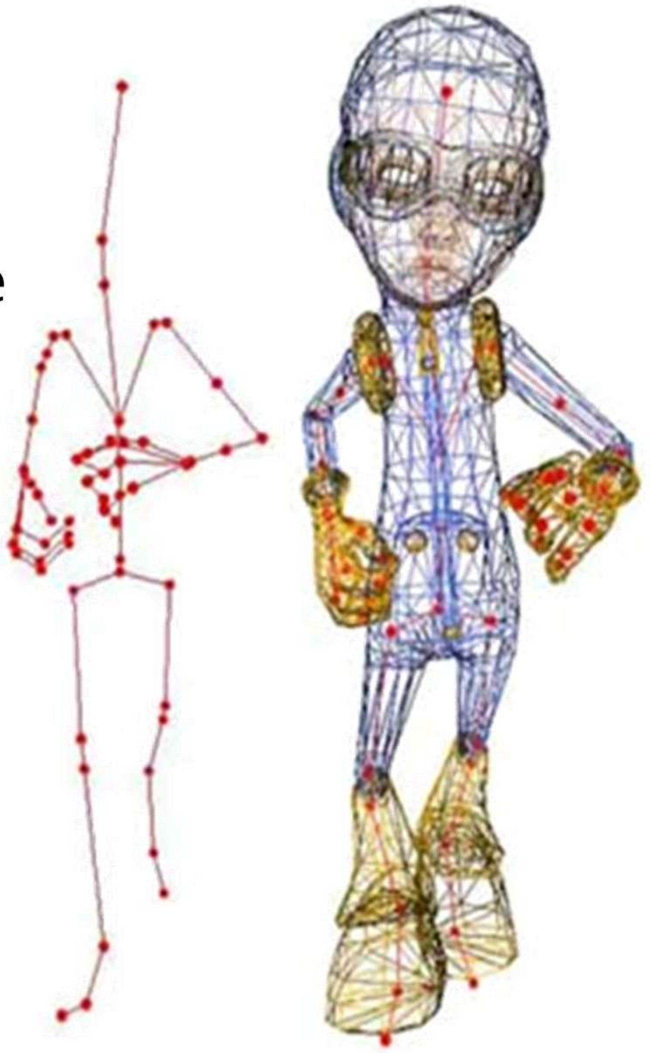
- Given the desired displacement of a point
  - Hand on doorknob, foot on the ground, …
- How to compute the necessary joint motions?
  - Solve joint chain (chain of transformation matrices!)
  - Solve equation system





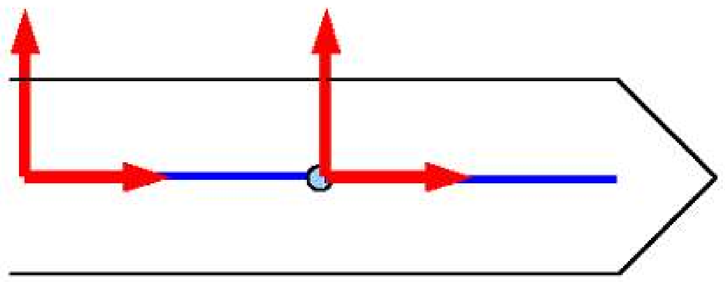
Skinning

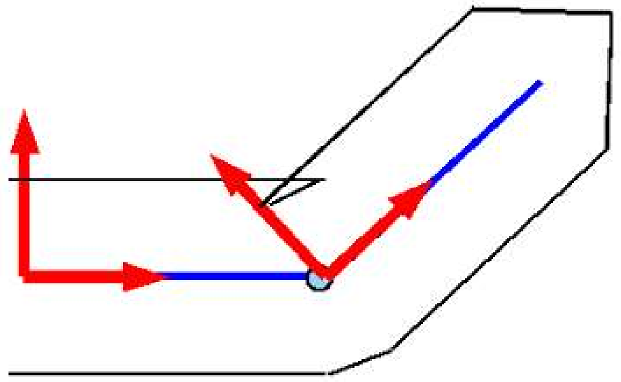
- Till now only animated skeleton
- Rigid geometry (at joints) usually looks very strange (maybe OK for robots)
- Need to wrap skin (a mesh), flesh, clothes … around the animated skeleton
- Continuous mesh deformations



Why trivial methods do not work

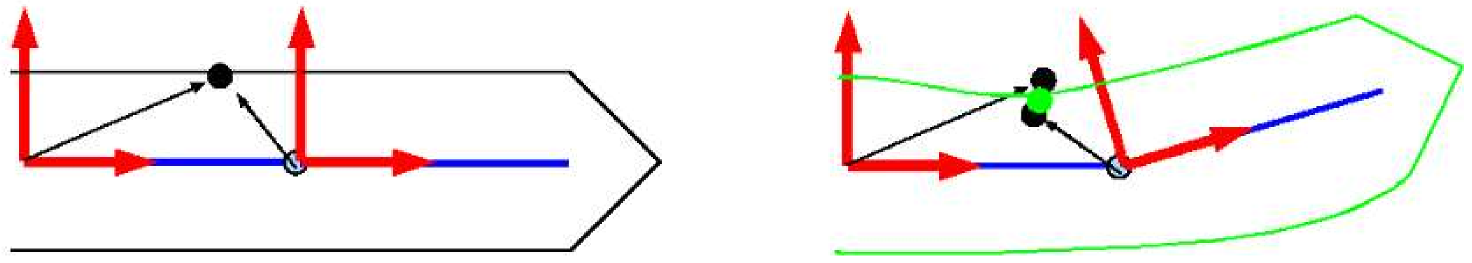
- Attach each vertex to the closest solid
  - Discontinuities
  - Self-intersections





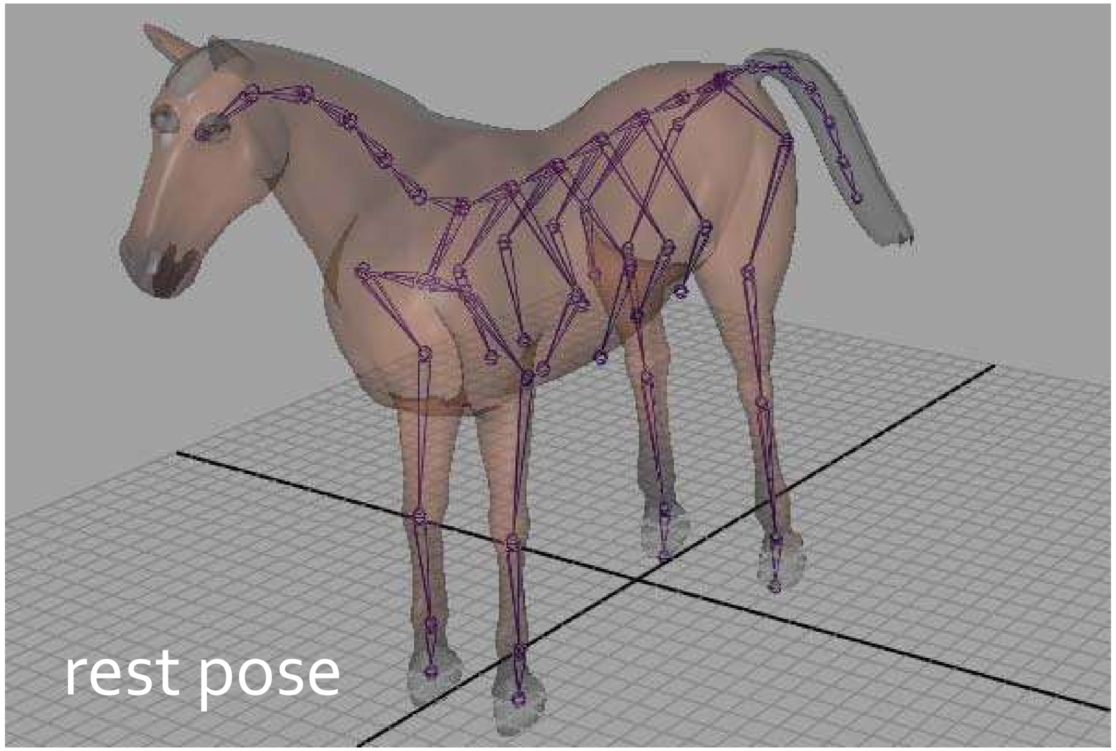
Skinning: Linear Blending

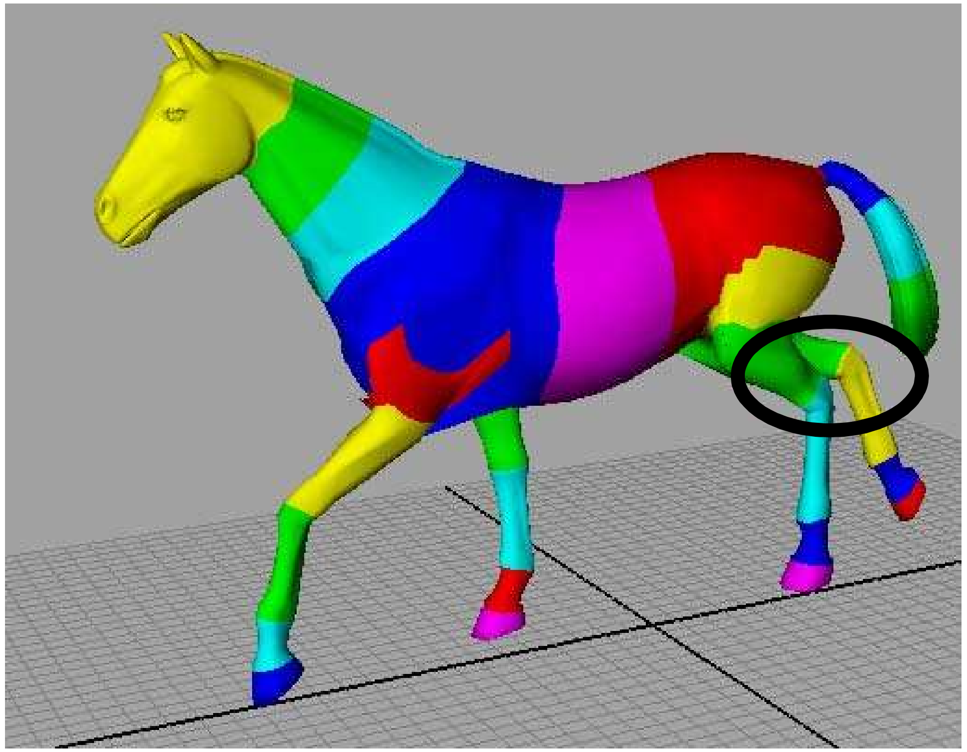
- Basic idea
  - Record vertex position in the closest solids
  - Apply a weighted sum



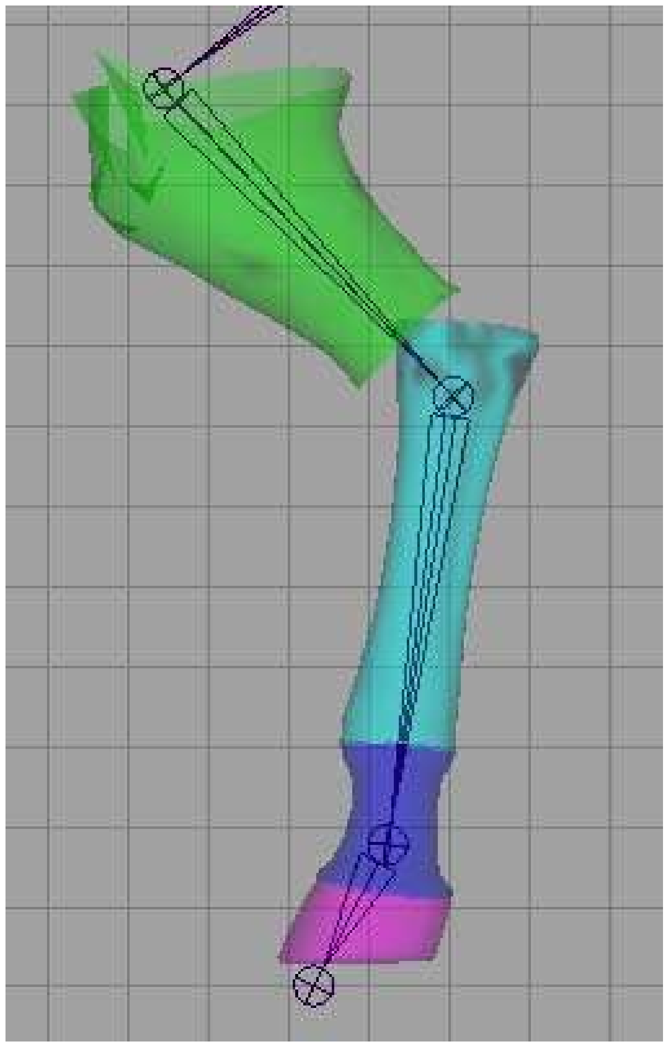
- Difficulties
  - Which solids to use?
  - Which weights?
  - Chosen by artist

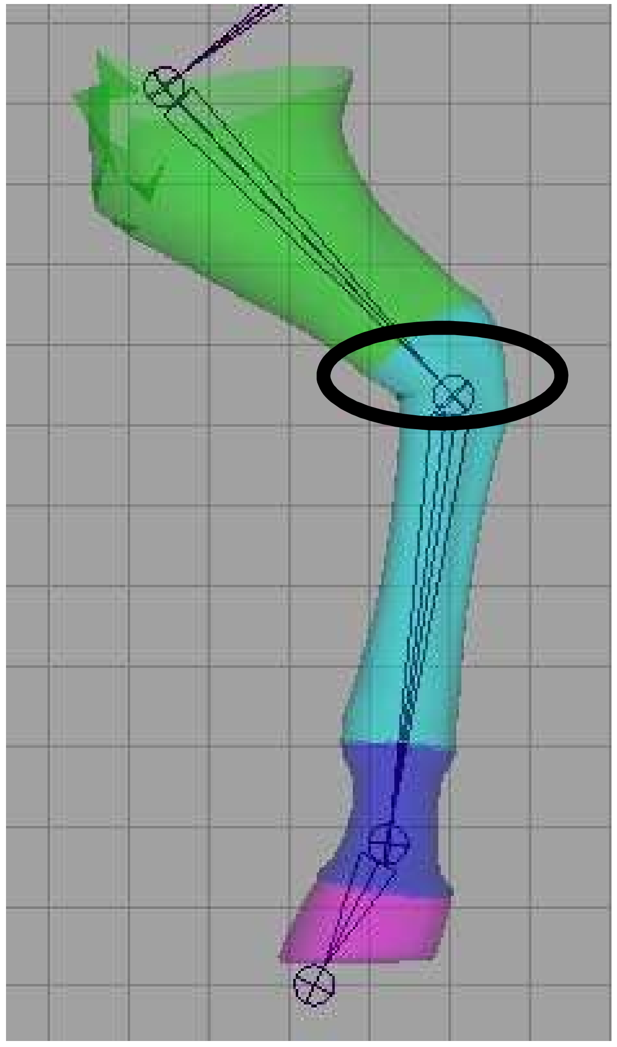
Skinning: Example



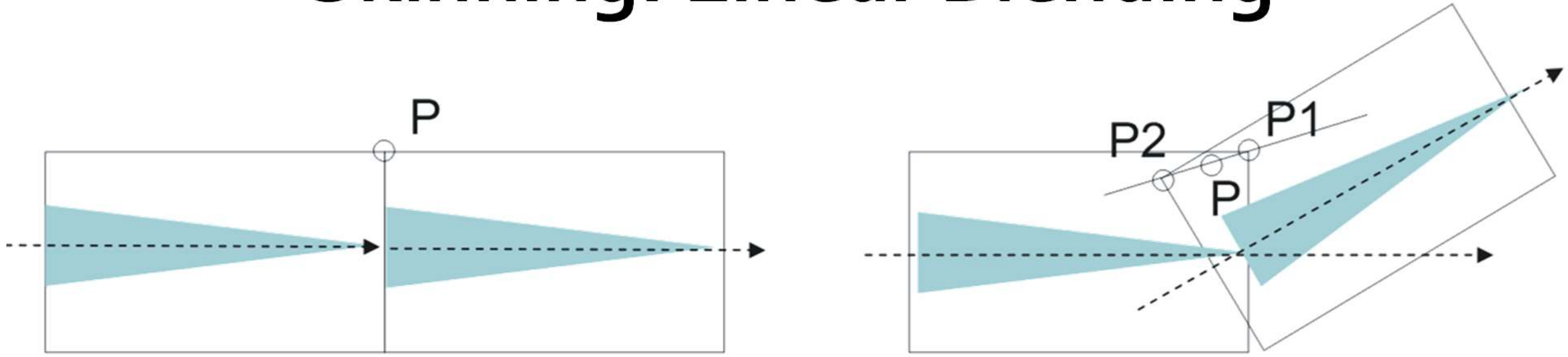


Skinning: Example



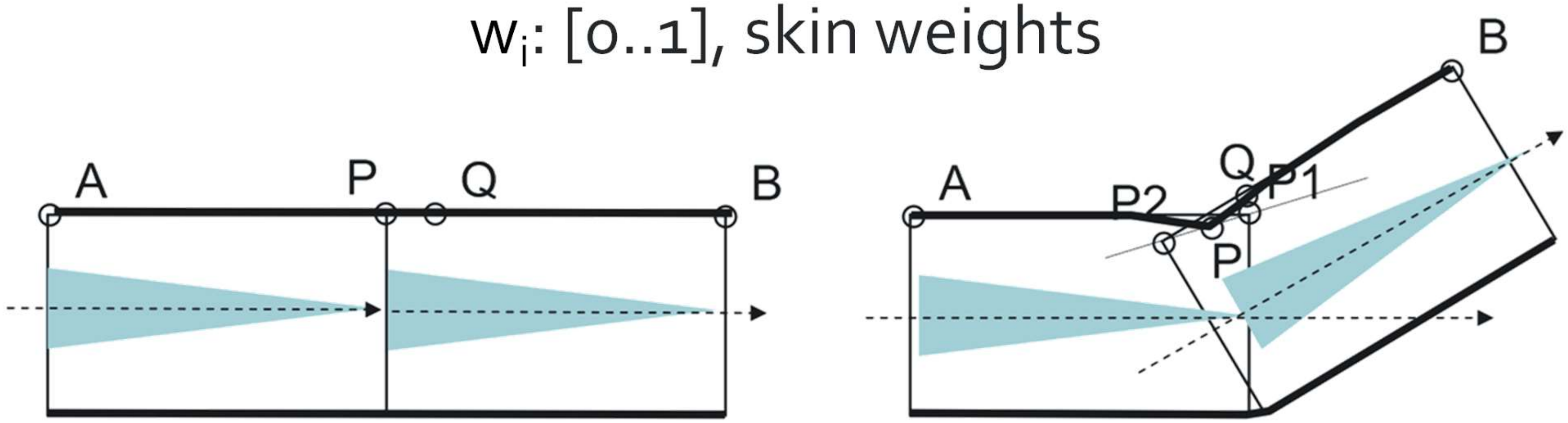


Skinning: Linear Blending



$$\mathbf{P} = w_1 * \mathbf{P_1} + w_2 * \mathbf{P_2}$$

w_i: [0..1], skin weights



Skeletal Subspace Deformation (SSD)

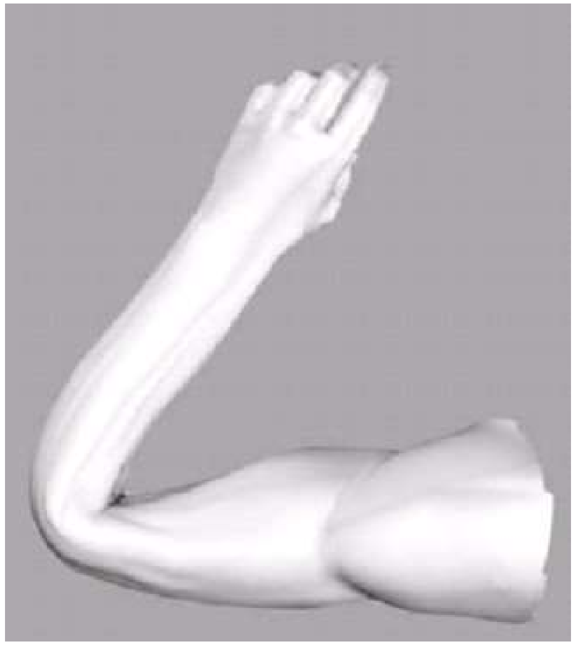
- Deformation used for skinning
- Define skin geometry around skeleton in rest pose
- For each vertex
  - Figure out weights for each bone
    - based on distance
    - painted on by artist
  - Weights sum to 1
  - New position is weighted average of positions indicated by nearby bones

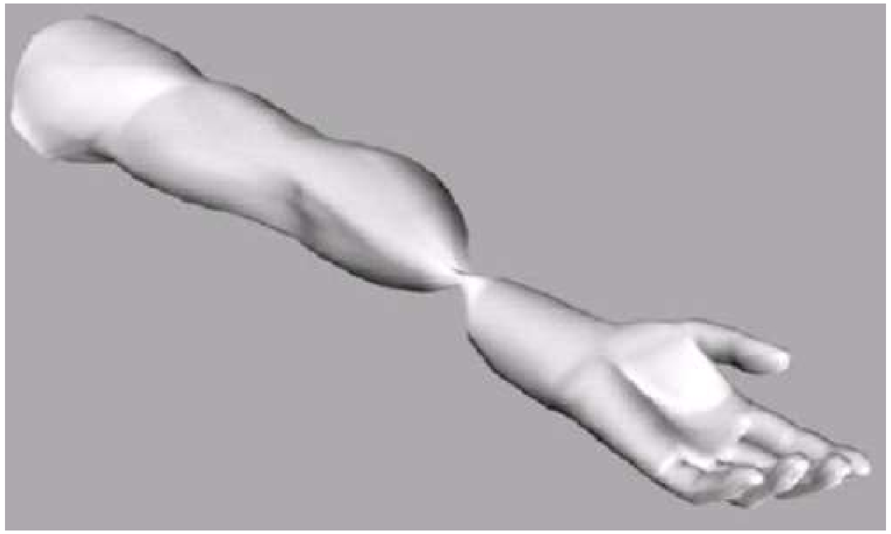
Controlling SSD’s

- Note: rigid parts, will want to have all but one weight equal to zero
  - “Skin” moves rigidly with bone
- Flexible parts near joints
  - Smoothly change weights from emphasizing one bone to the other
  - Skin will stay smooth as skeleton moves

SSD Problems

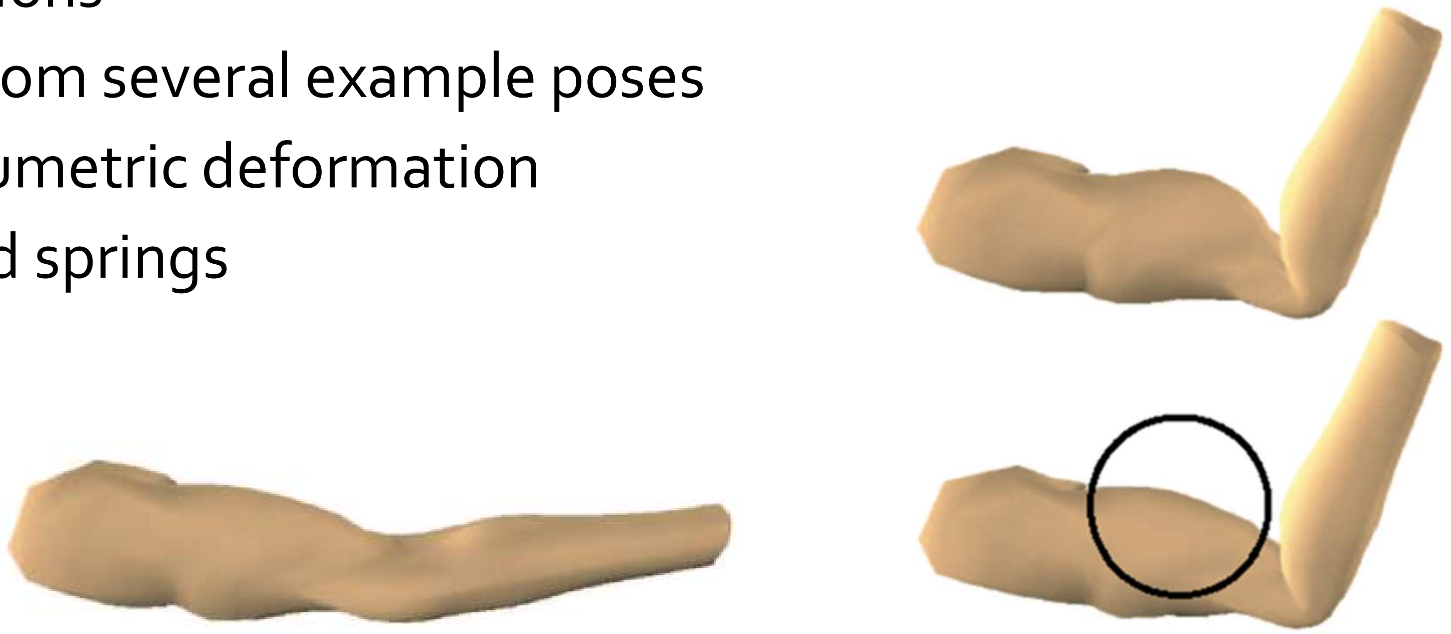
- Joint pinching
  - Large rotations at a joint
  - Skin deflating
  - Folding over itself, …





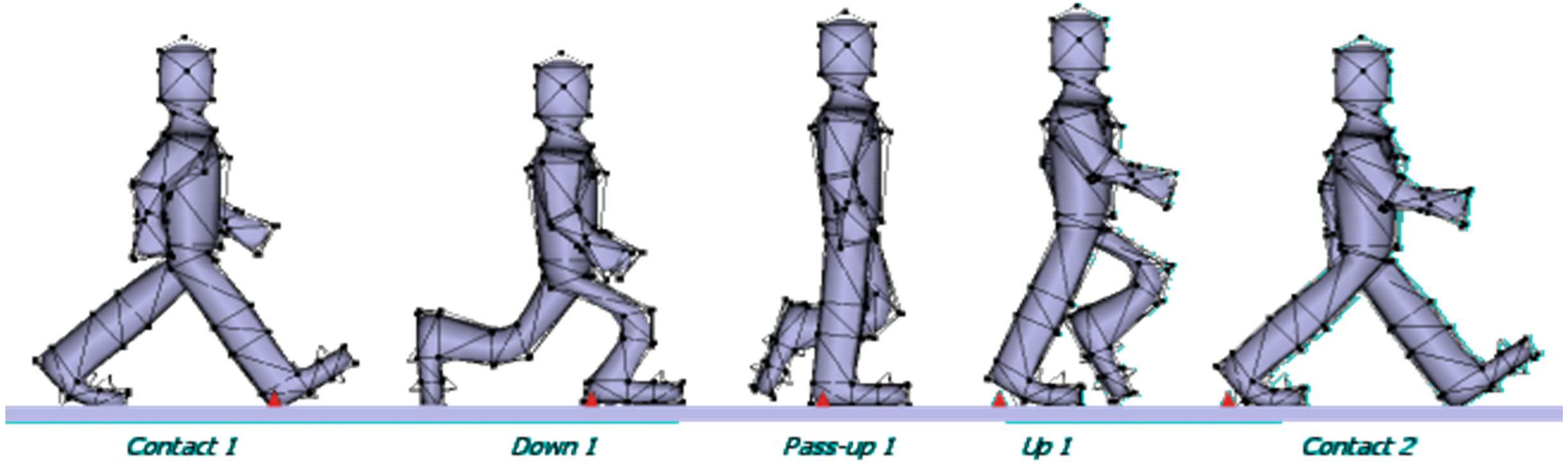
SSD Problems

- Missing effect of underlying anatomy
  - Muscles bulging, wrinkles, …
- Advanced solutions
  - Interpolate from several example poses
  - Simulate volumetric deformation
    - Masses and springs
  - …



Skeleton Posing Process

1. Specify DOF values for skeleton in animation system
2. Traverse through the hierarchy starting at the root down to the leaves (forward kinematics) and compute the world matrices
3. Use world matrices to deform skin & render



Animation

Data Acquisition

Motion Capture

- Human motion very subtle
  - E.g. shifting balance, complex joints, personality…

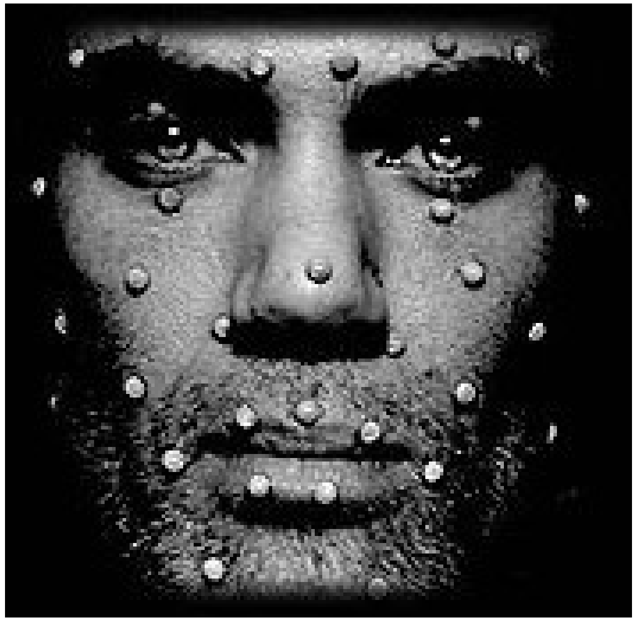
- Motion capture (mocap) records real motion from actors
  - E.g. Gollum, Polar Express, Beowulf, a lot of TV shows, plenty of games

- Technical difficulties:
  - How do you record?
  - What do you do with the data?

Mocap Methods

- Most common: “marker-based”
  - E.g. draw dots on the actor’s face, or dress in black and attach retro-reflective balls in key places
  - Film from one or more cameras (preferably calibrated and synchronized, preferably with a strobe light)
  - Reconstruct 3D positions of markers in each frame
- Some “markerless” systems: rely on good computer vision algorithms
- Some use direct or electromagnetic measurement

Motion Capture

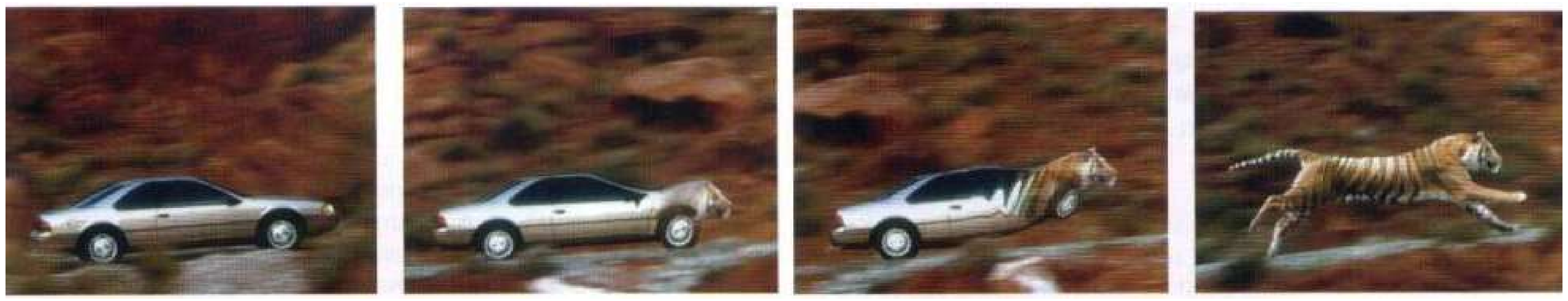


Move trees

- Standard videogame solution
- Design a graph corresponding to available player actions
  - E.g. walk forward, turn, jump, …
- Design and record corresponding actions with mocap
- Warp/retime/edit to make clips easily transition where needed
- Note: in playback need to keep separate track of global position/orientation

Morphing

- Closely related family of effects
- Warp two images or models to roughly match each other's geometry
  - Often based on artist-selected features
  - Modern computer vision and/or geometry algorithms getting better at automatically finding matches



Morphing

- Closely related family of effects
- Warp two images or models to roughly match each other’s geometry
  - Often based on artist-selected features
  - Modern computer vision and/or geometry algorithms getting better at automatically finding matches
- Cross-fade between the two to get in-between frames
  - For images, just average pixels
  - For 3D geometry, helps to have a common parameterization…

Animation

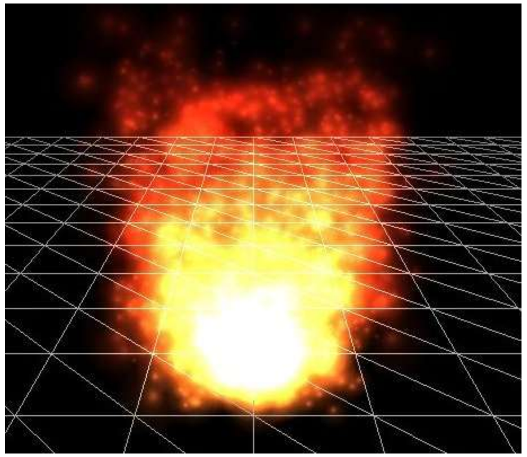
Physics Simulation

Physics Simulation

- Particles
- Rigid bodies
  - Collisions, contact, stacking, …
- Articulated bodies
  - Hinges, constraints
- Deformable bodies (solid mecha.)
  - Elasticity, plasticity, viscosity
  - Fracture
  - Cloth
- Fluid dynamics
  - Fluid flow (liquids & gasses)
  - Combustion (fire, smoke, explosions…)
  - Phase changes (melting, freezing, boiling…)
- Vehicle dynamics
  - Cars, boats, airplanes, helicopters, motorcycles…
- Character dynamics
  - Body motion, skin & muscle, hair, clothing

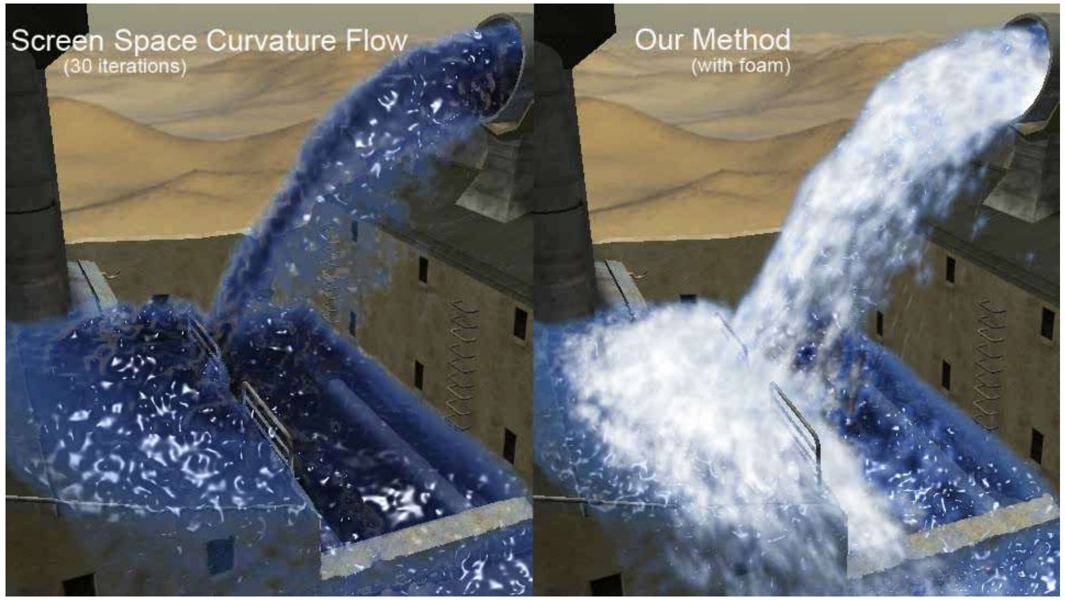
Particle Systems

- Break up complex phenomena into component parts
  - Particles
- For fuzzily defined phenomena
- Highly complex motion
- No animation of each part by hand
- Provide overall rules for animation



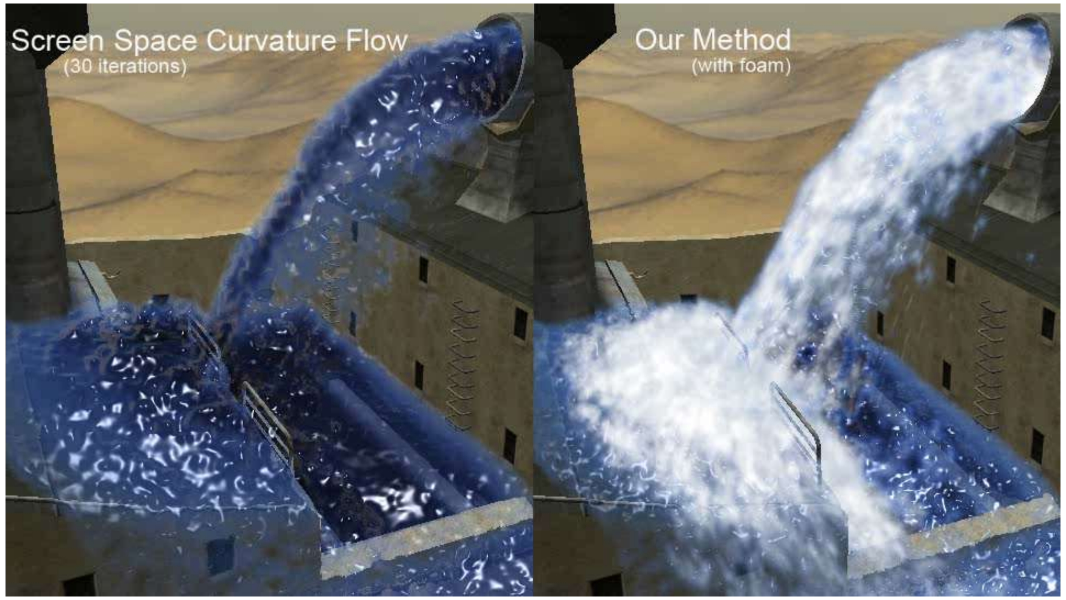
Particle Systems

- Dust, sparks, fireworks, leaves, flocks, water spray…
- Also phenomena with many DOF
  - Fluids (water, mud, …), fire, explosions, hair, fur, grass, clothing, …



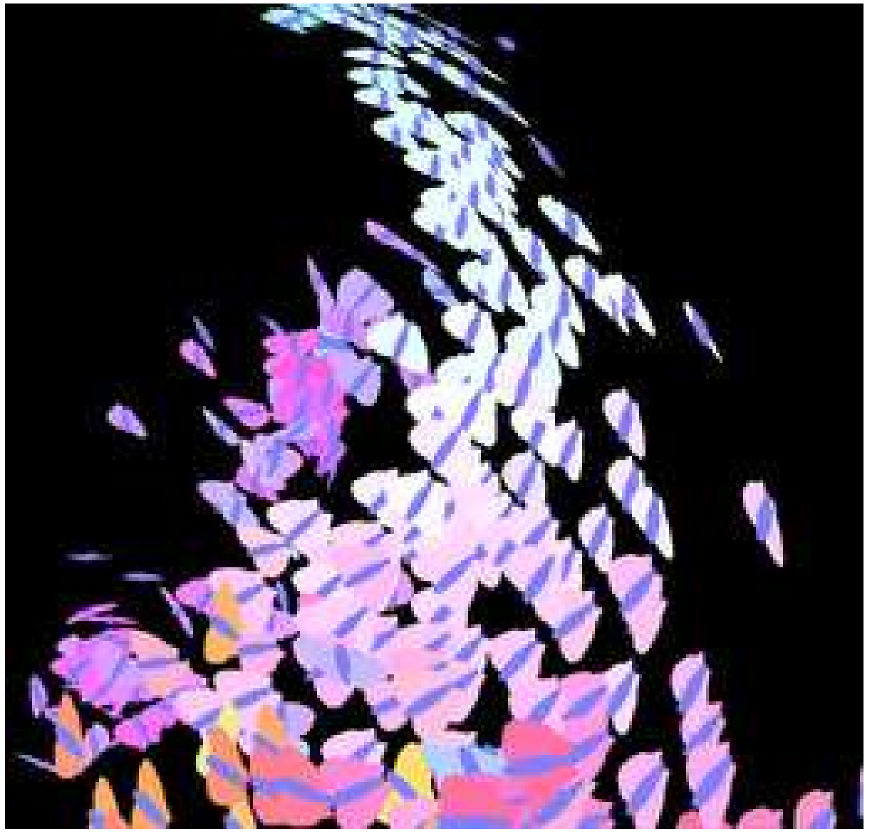
Particle Systems

- Three things to consider:
  - When and where particles start/end
  - Rules that govern motion (attached variables, e.g. color)
  - How to render the particles



What is a Particle?

- Particles have a position
- Usually add other attributes
  - Age, colour, radius, orientation, velocity, mass, temperature, type
- The sky is the limit
  - e.g. AI models of agent behaviour e.x.: flocking behaviour



Particle Seeding

- Need to add (or seed) particles to the scene
- Where?
  - Randomly within a shaped volume or on a surface
    - Uniform, jittered grid, …
  - At a source (waterfall, …)
  - Where there aren’t many particles currently
- When?
  - At the start
  - Several per frame
  - When there aren’t enough particles somewhere
- Need to figure out other attributes, not just position
  - E.g. velocity pointing outwards in an explosion

Basic Animation

- Specify a velocity field v(x,t) for any point in space x, any time t
- Break time into steps
  - E.g. per frame Δt = (1/fps)th of a second
  - Or several steps per frame
- Change each particle's position xi by "integrating" over the time step (Forward Euler)

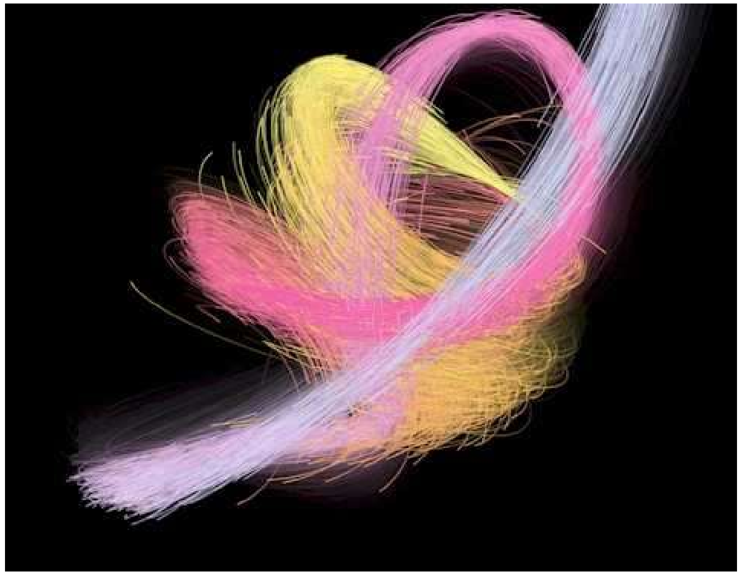
$$x_i^{new} = x_i + \Delta t v(x_i, t)$$

Basic Rendering

- Draw a dot for each particle
- But what do you do with several particles per pixel?
  - No special handling
  - Add: models each point emitting (but not absorbing) light -- good for sparks, fire, …
  - Compute depth order, do alpha-compositing (and worry about shadows etc.)
- Anti-aliasing
  - Blur edges of particle, make sure blurred to cover at least a pixel
- Particle with radius: kernel function

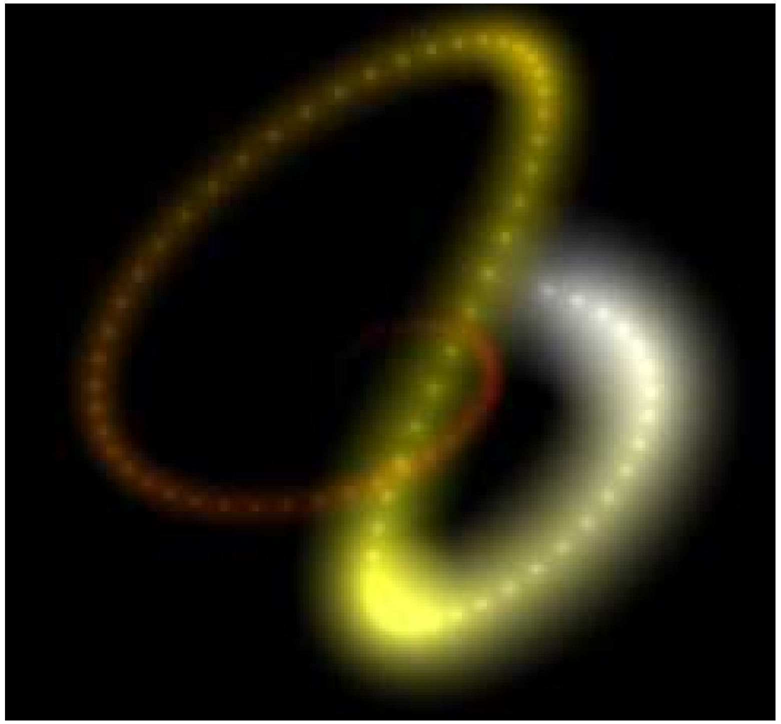
Motion blur

- One case where you can actually do exact solution instead of sampling
- Really easy for simple particles
  - Instead of a dot, draw a line
    (from old position to new position - the shutter time)



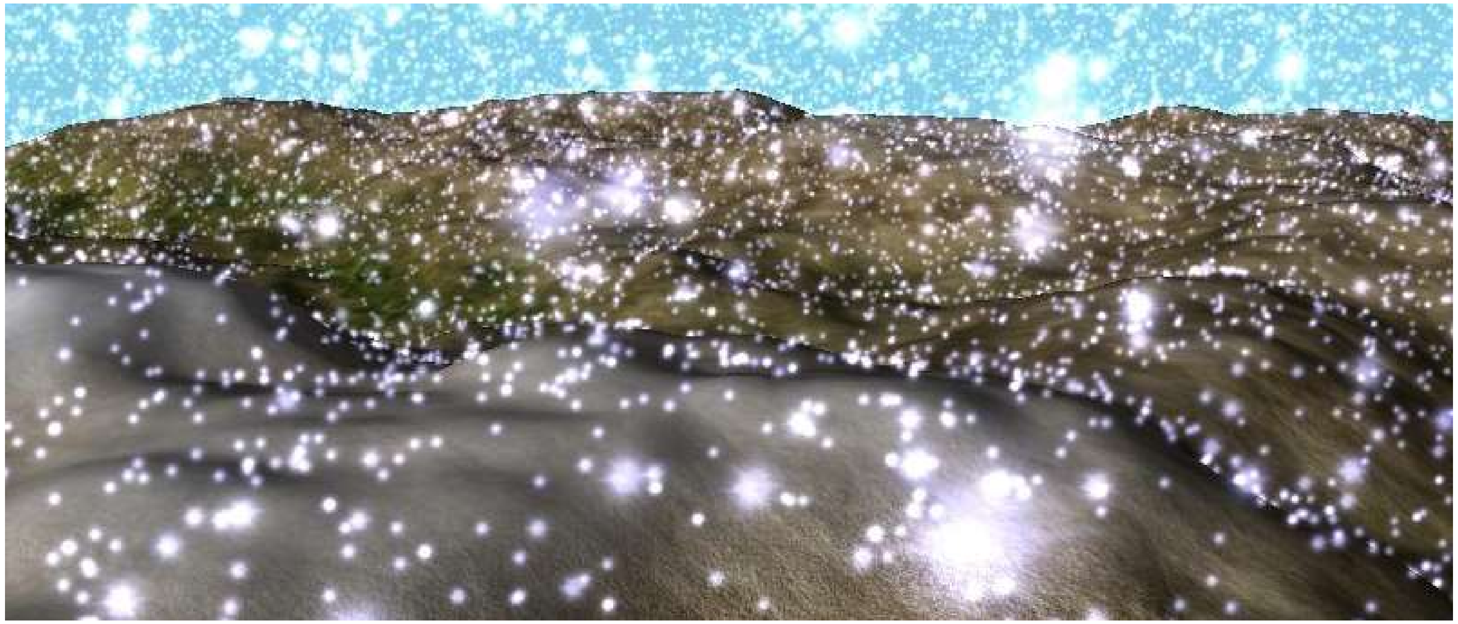
Motion blur

- May involve decrease in alpha
- More accurately, draw a spline curve
- May need to take into account radius as well…



More Detailed Particle Rendering

- Stick a texture (or even a little movie) on each particle: “sprites” or “billboards”
  - E.g. a noise function
  - E.g. a video of real flames



More Detailed Particle Rendering

- Stick a texture (or even a little movie) on each particle: “sprites” or “billboards”
  - E.g. a noise function
  - E.g. a video of real flames
- Draw a little object for each particle
  - Need to keep track of orientation as well, unless spherical
- Draw between particles
  - curve (hair), surface (cloth)
- Implicit surface wrapped around virtual particles (e.g. water)

Physics Simulation