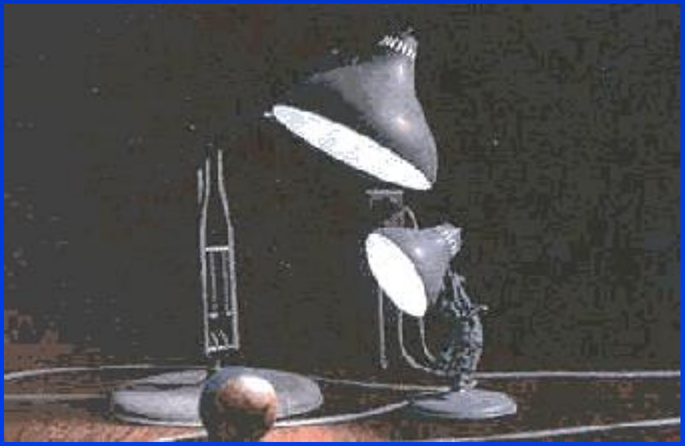
Computer Animation

Computer Animation

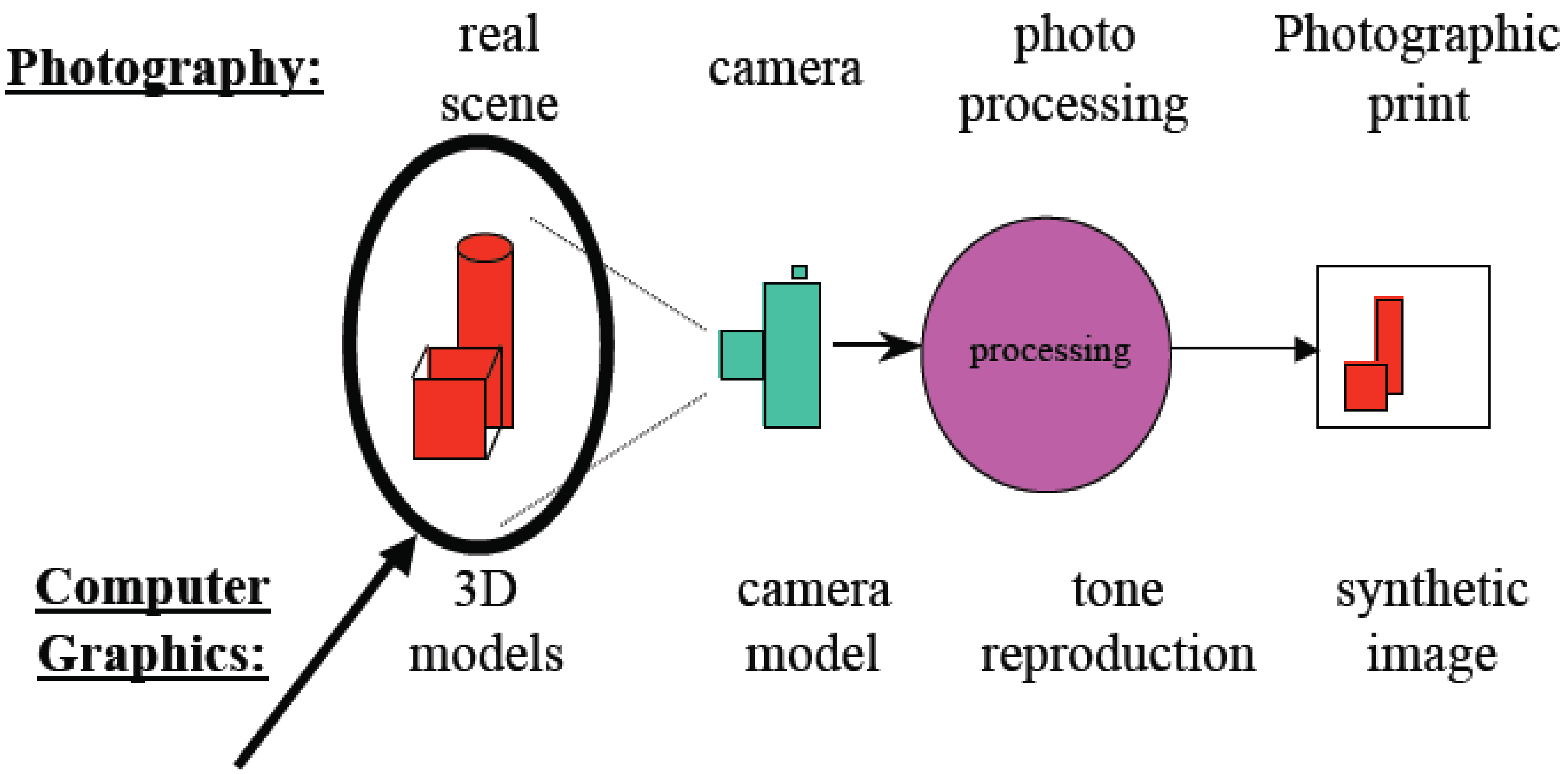
- **Animation Pipeline**
  - 3D modeling
  - Motion specification
  - Motion simulation
  - Shading, lighting, & rendering
  - Postprocessing



What is Animation

- A motion picture comprises a series of drawings/images simulating motion by means of slight progressive changes in the drawings
- Computer animation?
  - Make objects change over time according to scripted actions, and the animation consists of relative movement between rigid bodies and possibly movement of the virtual camera (or view point)
  - Using a standard renderer to produce consecutive frames

Computer Animation



Animation deals with making this move

Computer Animation

- This is completely analogous to model animation where scale models are photographed by special cameras

Simulation

- **What is simulation?**
  - Predict how objects change over time according to physical laws

Why Animation Works?

- The eye cannot register images faster than approximately 50 frames per second (30 is just about adequate)
- If a gap in the projection occurs, the eye seems to perform spatial interpolation over the gap

Computer Animation

- Creating animation sequences
  - object definition
  - path specification
  - key frames
  - in-betweening
- Parametric equations
- Displaying animation sequences
  - raster animation

Traditional Animation Techniques

- Prior to the advent of computational era…….

Overview: Traditional Animation

- **Early 2D Animation:** Used traditional techniques
- **Early 3D Animation:** Neglected traditional techniques
- Understanding of these **Fundamental principles of** traditional animation techniques is essential to producing good computer animation

Basic Principles of Traditional Animation (from Disney)

- **Squash and stretch**
- **Slow in and out**
- **Anticipation**
- **Exaggeration**
- **Follow through and overlapping action**
- **Timing**
- **Staging**
- **Straight ahead action and pose-to-pose action**
- **Arcs**
- **Secondary action**
- **Appeal**

Squash and Stretch

- Style vs. accuracy
- Teaches basic mechanics of animation
- Defines rigidity of material
- Time interpolation captures accuracy of velocity
- Squash and stretch replaces motion blur stimuli and adds life-like intent

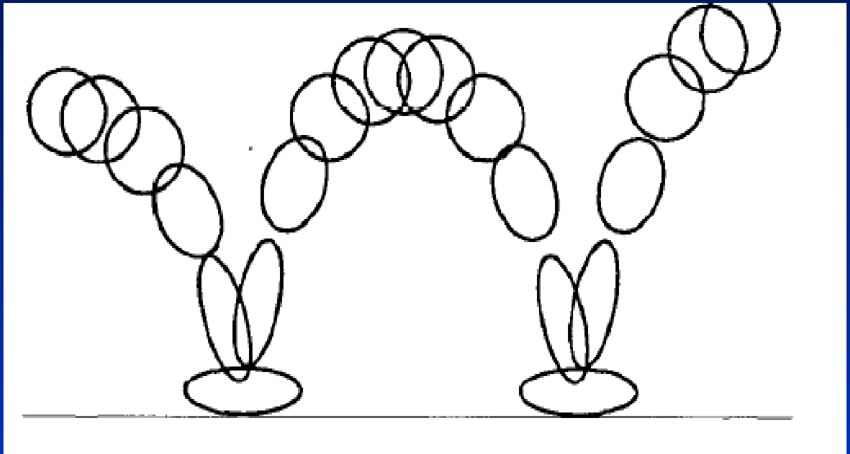
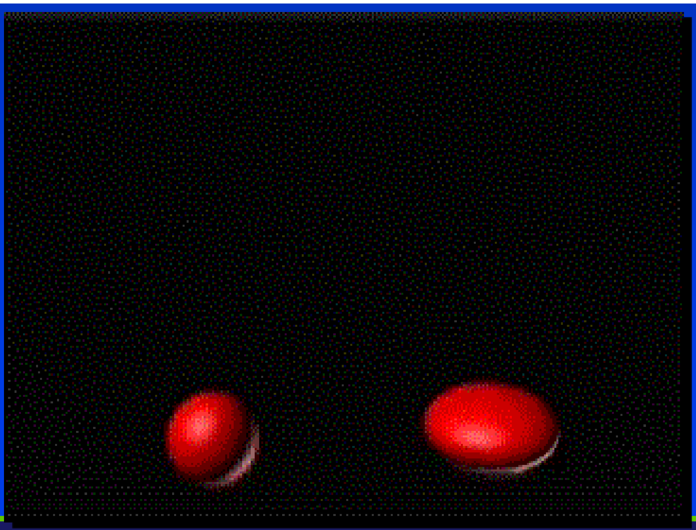
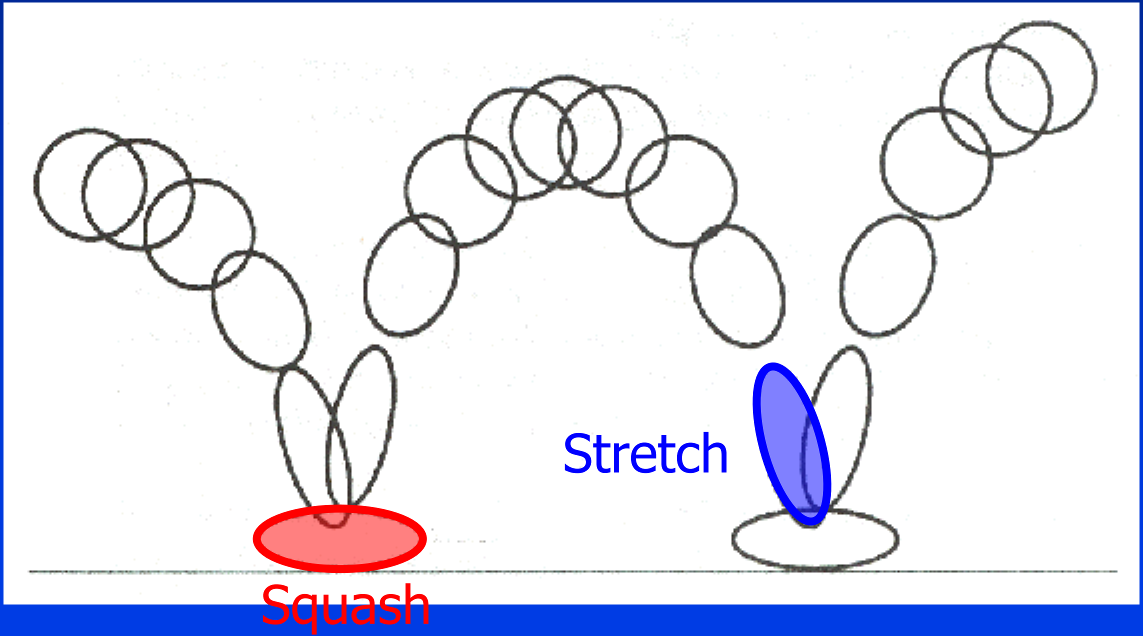


FIGURE 2. Squash & stretch in bouncing ball.



Squash and Stretch



Squash and Stretch

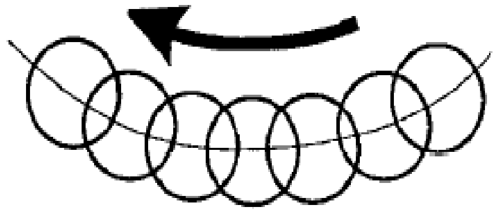


FIGURE 4a. In slow action, an object's position overlaps from frame to frame which gives the action a smooth appearance to the eye.

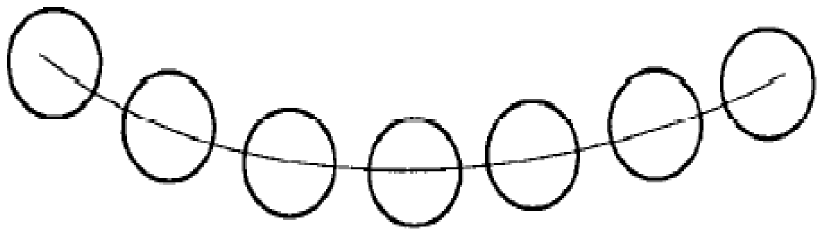


FIGURE 4b. Strobing occurs in a faster action when the object's positions do not overlap and the eye perceives seperate images.

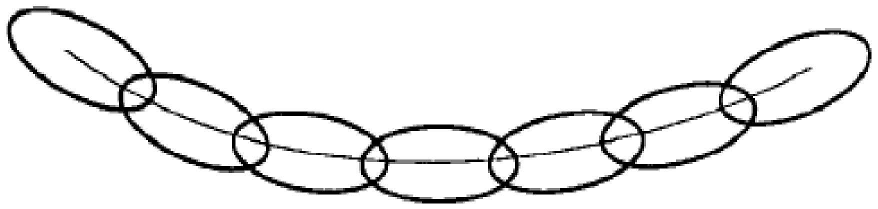
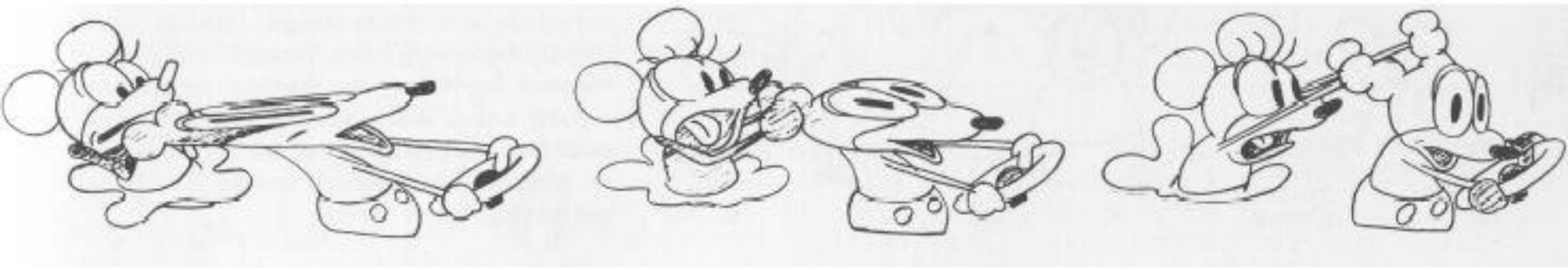
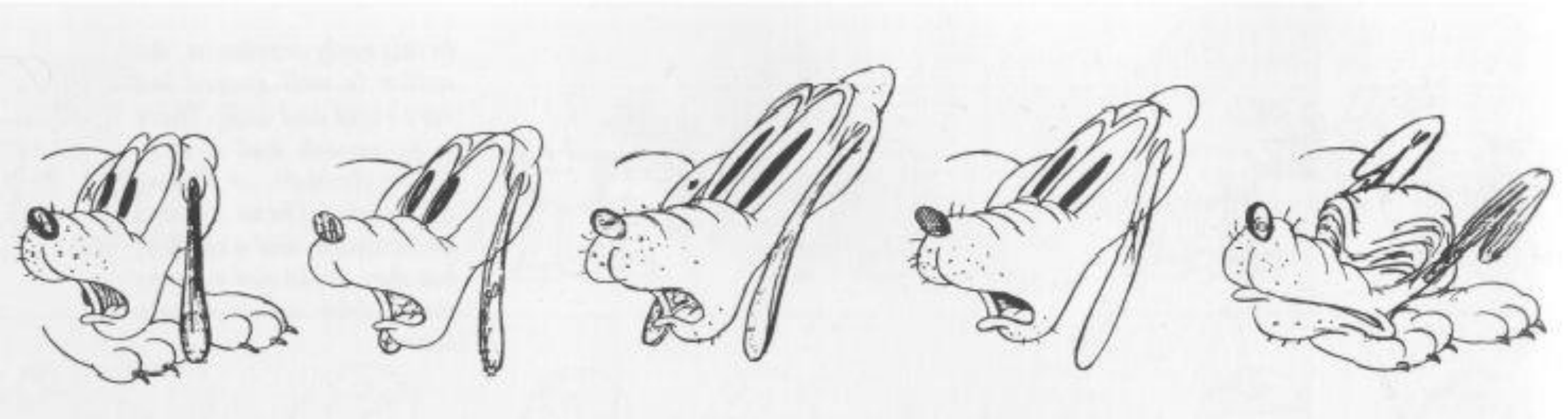


FIGURE 4c. Stretching the object so that it's positions overlap again will relieve the strobing effect.

- Can relieve the disturbing effect of strobing

Squash and Stretch





Timing and Motion

- Gives meaning to movement
- Proper timing is critical to making ideas readable
  - **•Examples:**
  - 1. Timing: tiny characters move quicker than larger ones.
  - 2. Motion: can define weights of objects.

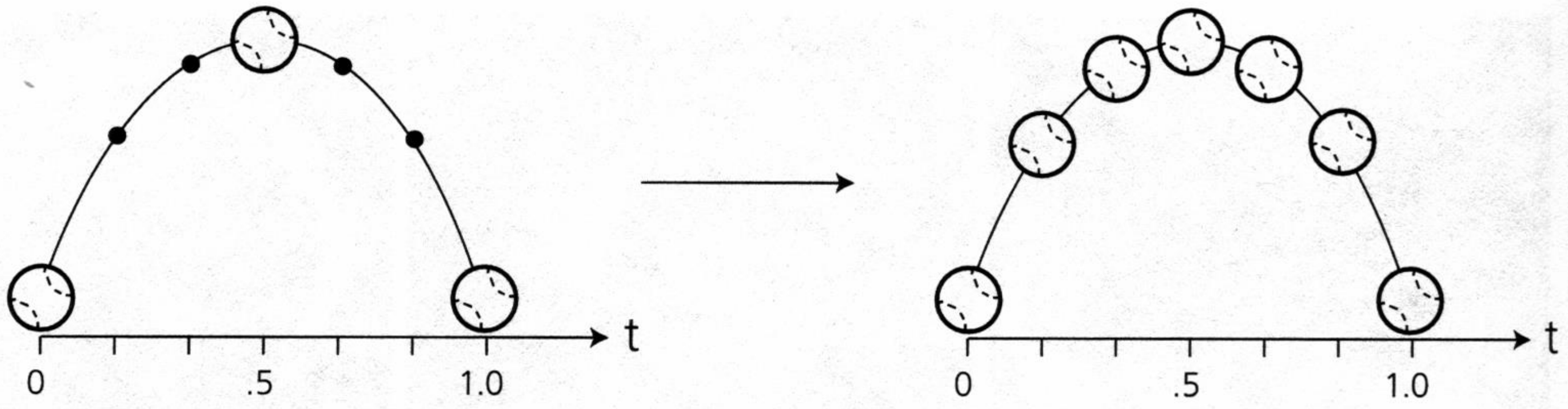


Figure 10.10 **Inbetweening with nonlinear interpolation and easing.** The ball changes speed as it approaches and leaves keyframes, so the dots indicating calculations made at equal time intervals are no longer equidistant along the path.

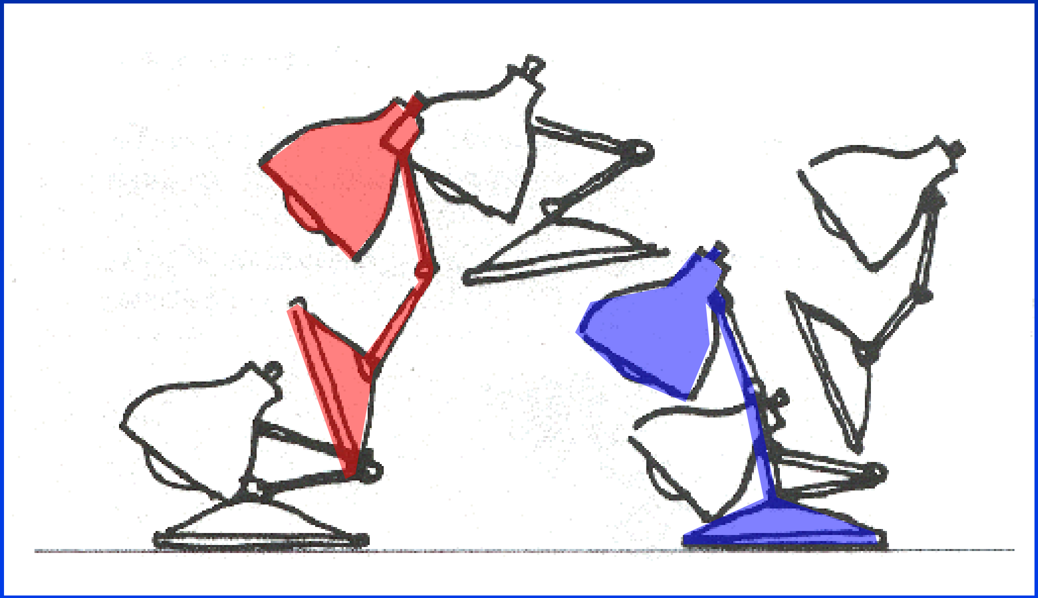
Anticipation

Preparation for an action

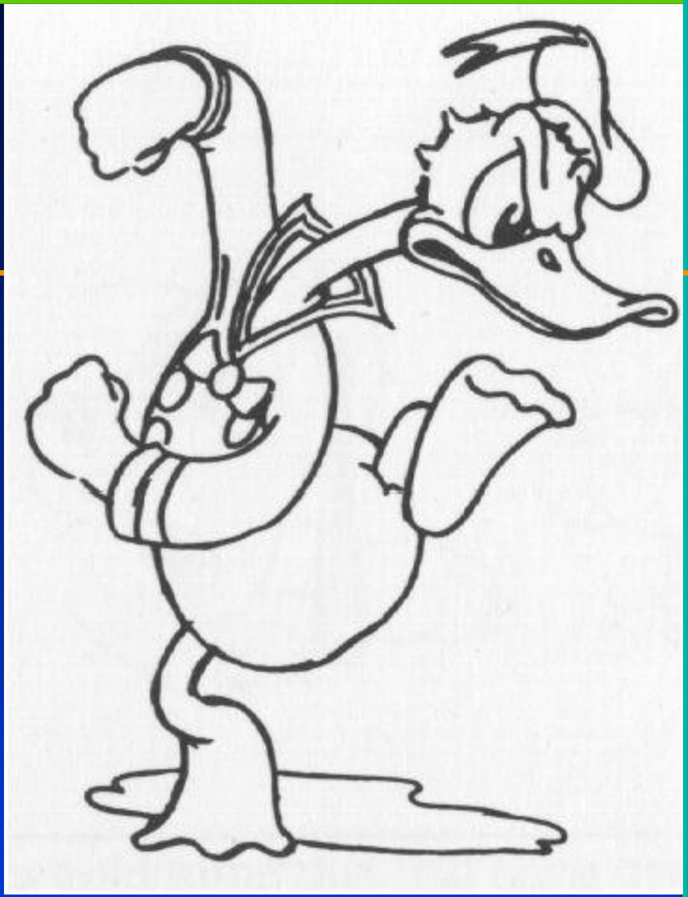
Example:

Goofy prepares to hit a baseball.

Anticipation



Staging



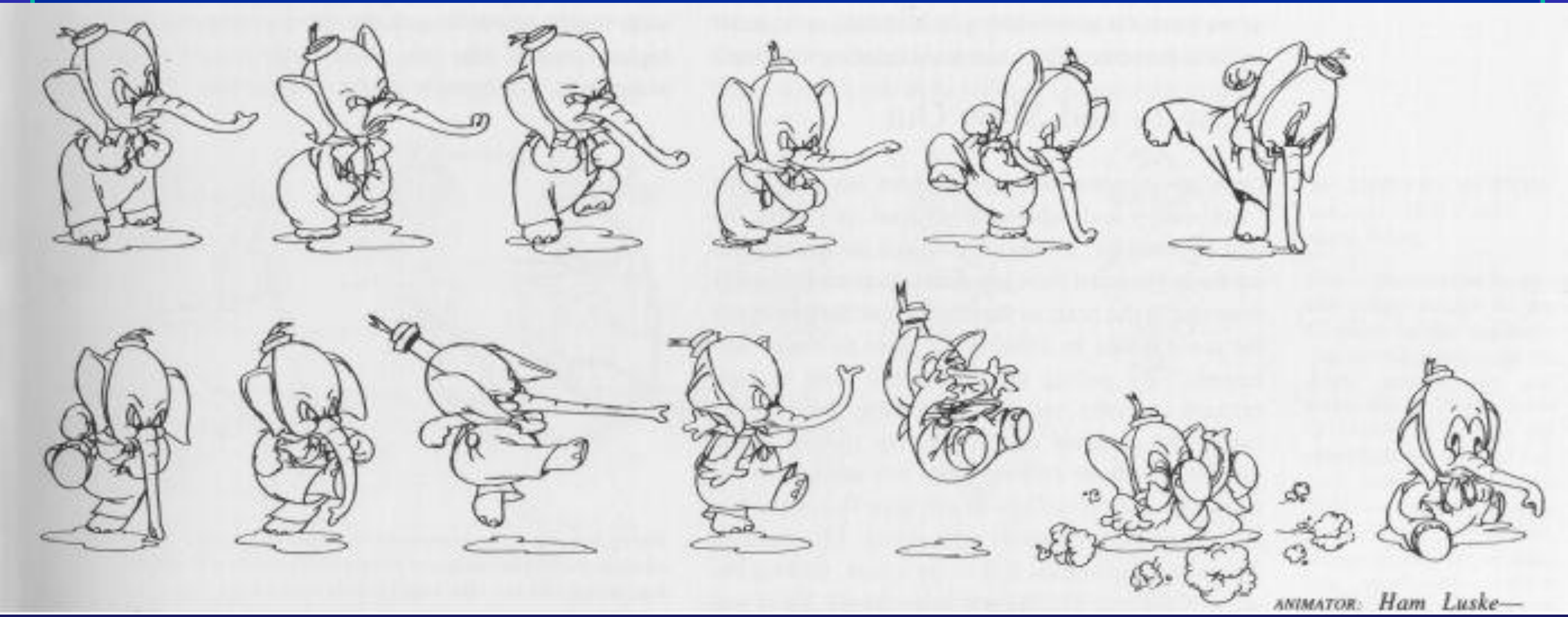
- A clear presentation of an idea
- Don’t surprise the audience
- Direct their attention to what’s important

Some Techniques:

1. **Use motion in a still scene or use of static movement in a busy scene.**
2. Use of silhouettes (to the side)

Follow Through

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



Follow Through and Overlapping Action

Follow Through

Termination part of an action

Example: after throwing a ball

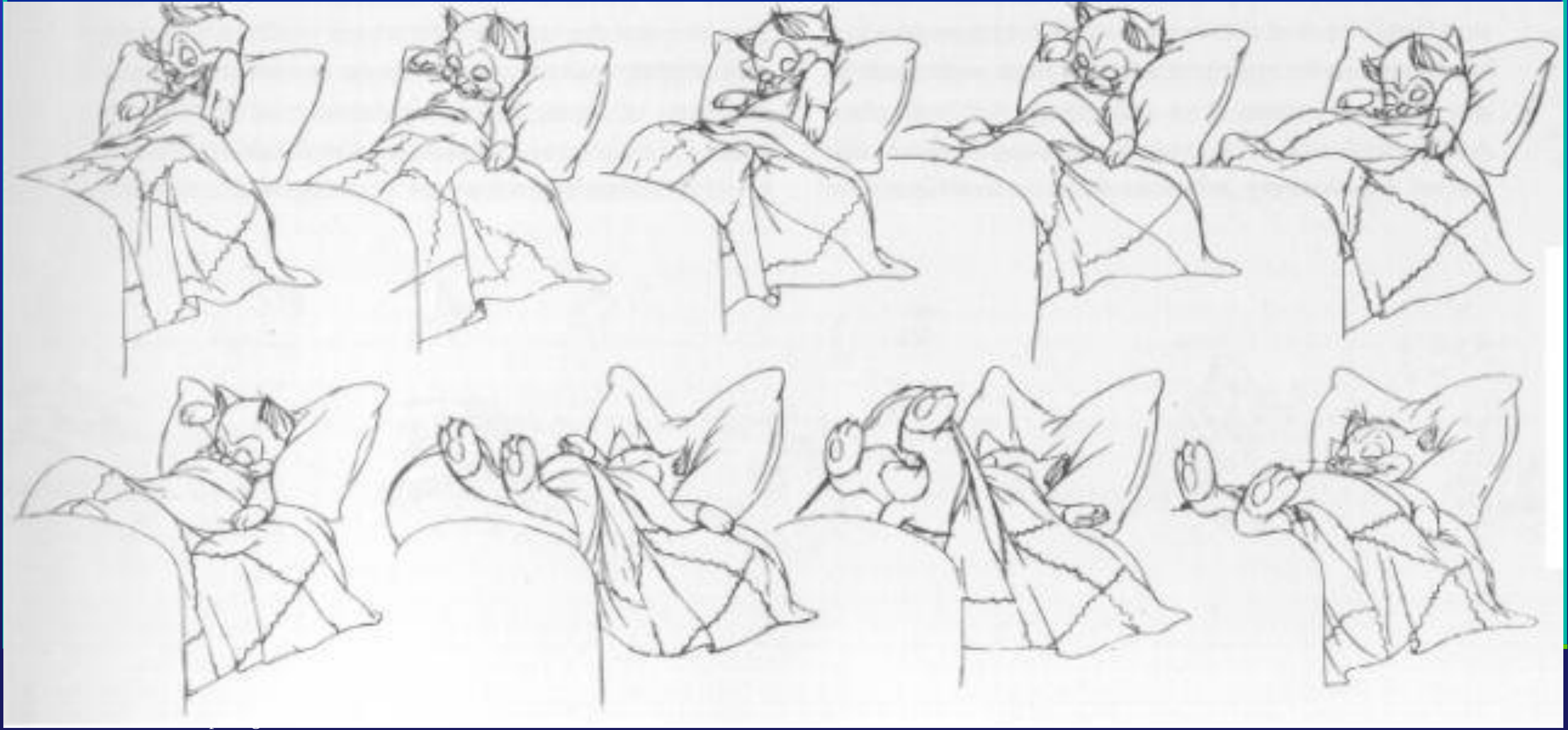
2. Overlapping Action

Starting a second action before the first has completed.

Example: Luxo Jr.'s hop with overlapping action on chord

Secondary Motion

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

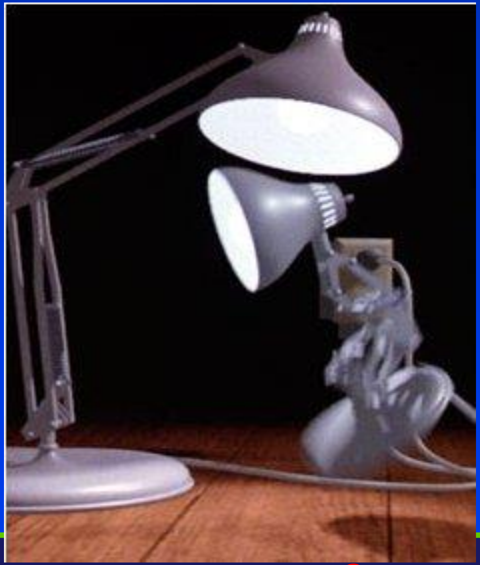


Secondary Action

- Action that results directly from another action
- Used to increase the complexity and interest of a scene

Example:

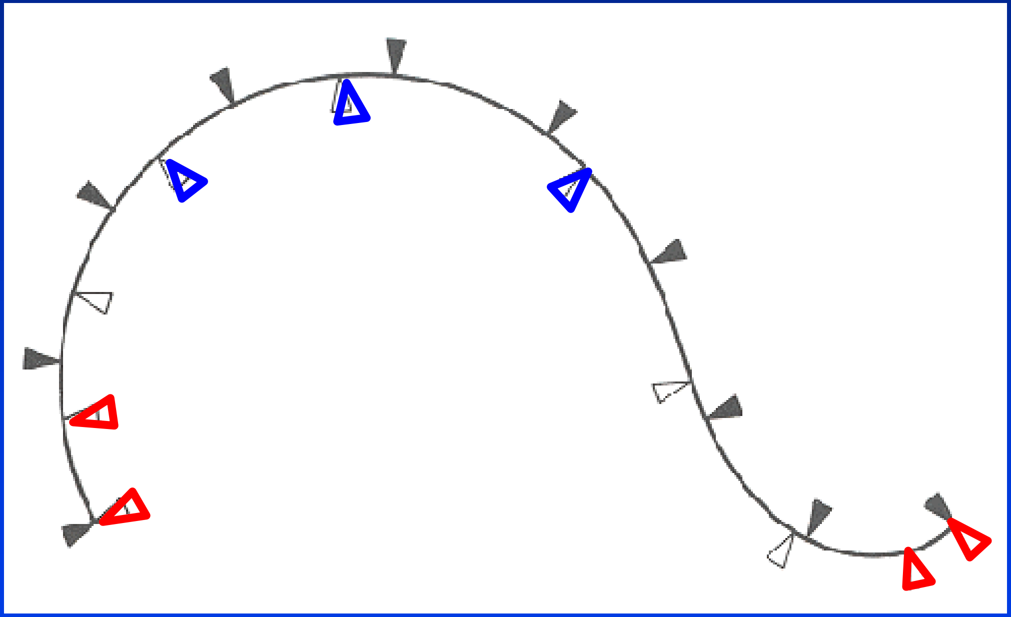
Body movement is the primary action, facial expression is the secondary action



Straight Ahead Action and Pose-to-Pose Action

- **Straight Ahead**
  - Animator starts from first drawing in the scene and draw all subsequence frames until the end of scene.
- **Pose-to-Pose**
  - Animator plans actions, draws a sequence of poses, in between frames etc.

Slow In and Out



Slow in and Out

Spacing of inbetween frames to achieve subtlety of timing and movement.

1. 3d keyframe comp. Systems uses spline interpolation to control the path of an object.
2. Has tendency to overshoot at extremes (small # of frames).

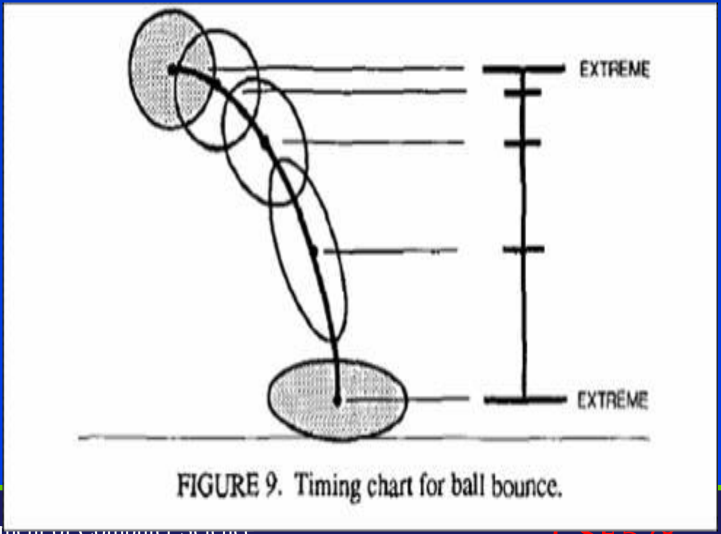
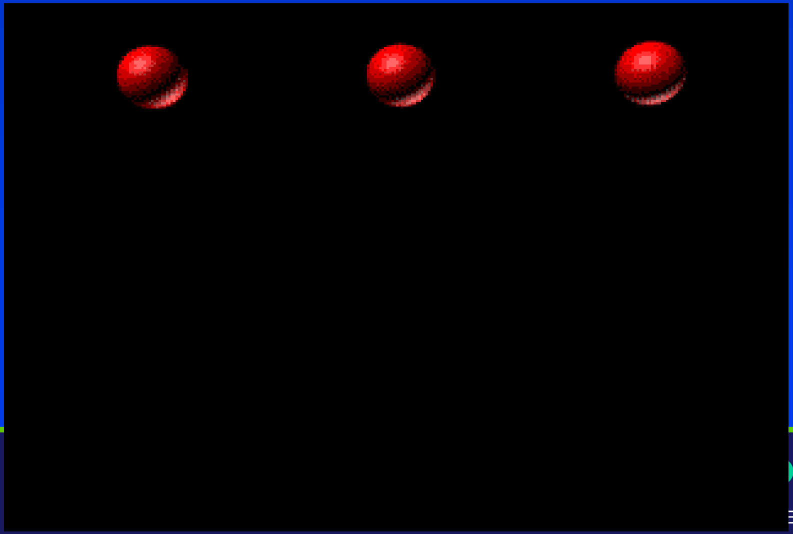


FIGURE 9. Timing chart for ball bounce.



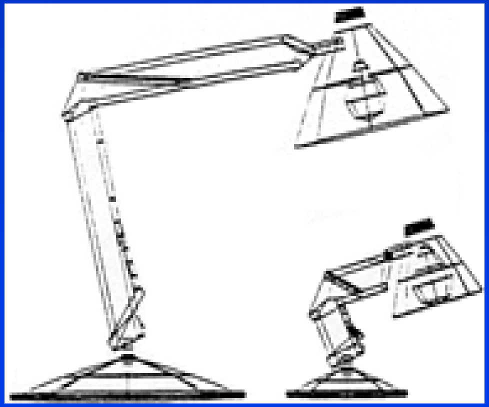
Arcs

- Visual path of action for natural movement
- Makes animation much smoother and less stiff than a straight line

Exaggeration

- Emphasizing the essence of an idea via the design and the action
- Needs to be used carefully

Example: Luxo Jr. made smaller to give idea of a child.

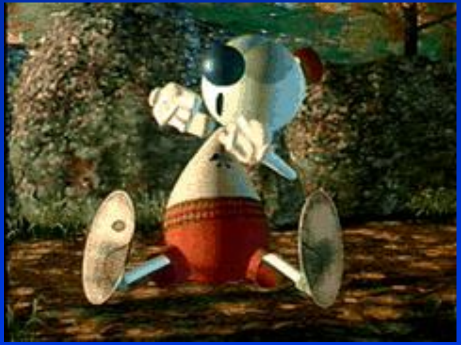


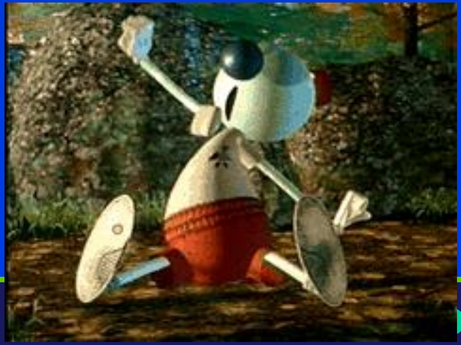
Appeal

- Refers to what an audience would like to see
- Character cannot be too simple (boring) or too complex

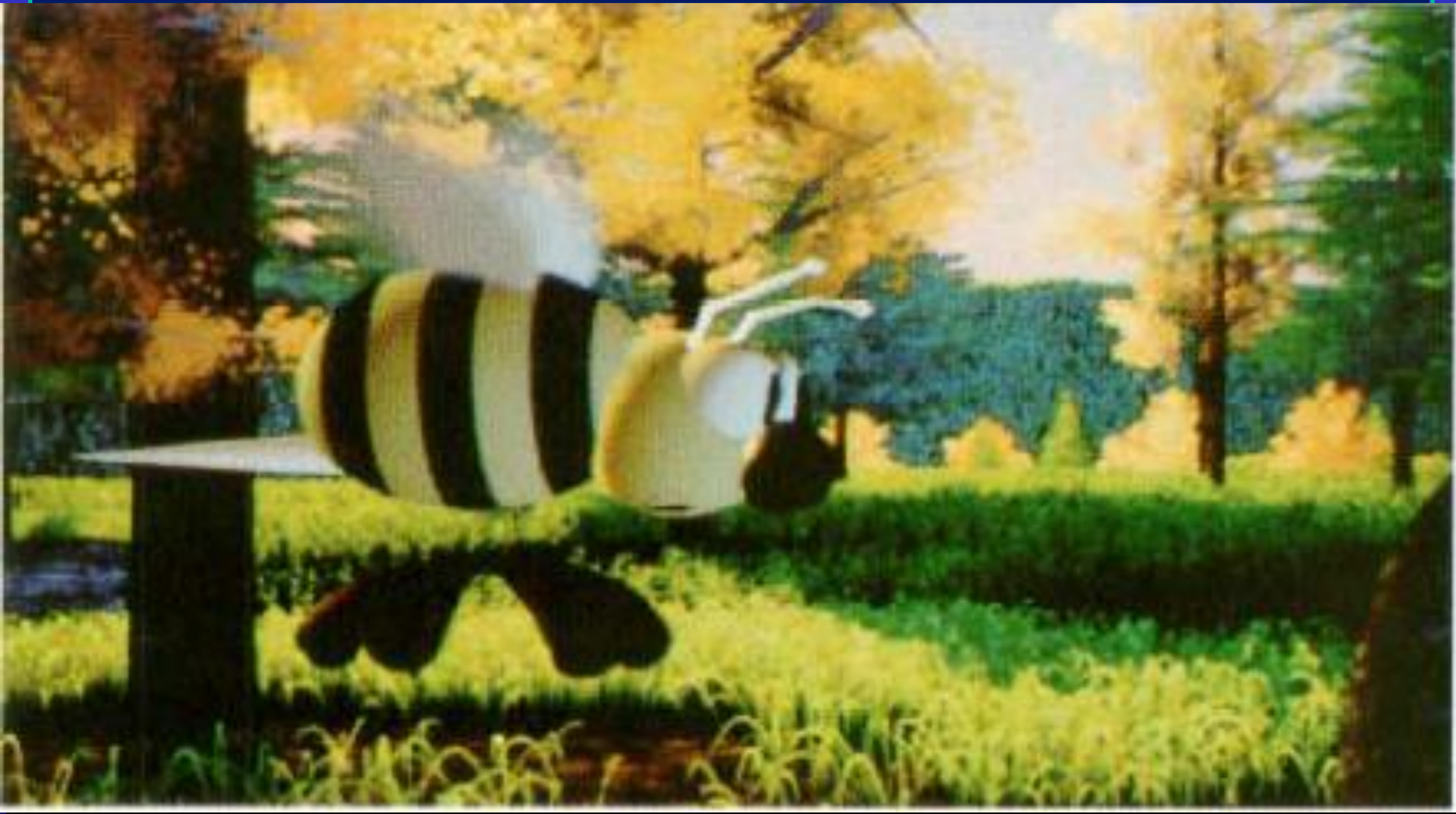
Examples:

Avoid mirror symmetry, assymmetry is interesting.





What techniques used for Wally B.?



What do you think Wally B's going to do?

The Action: Zoooooooooommmm!

Termination: Poof! He's gone!

Role of Personality

- Animator's first goal is to entertain
- Success of animation lies in the personality of the characters

Conclusion

Hardware/Software are simply not enough, these principles are just as important tools too.

Computer Animation Age

Basic Steps for a Simple Computer Animation

1. Creating animation sequences
   - Object/model definition
   - path specification (for an object or a camera)
   - key frames
   - in-betweening

- 2. Displaying the sequences

Displaying Animation Sequences

- Movies work by fooling our eyes
- A sequence of static images presented in a quick succession appears as continuous flow

Displaying Animation Sequences

- To achieve smooth animation, a sequence of images (frames) have to be presented on a screen with the speed of at least 30 per second
- Animations frames can be
  – pre-computed in advance and pre-loaded in memory
  – computed in real time (e.g. movement of the cursor)

Raster Animation

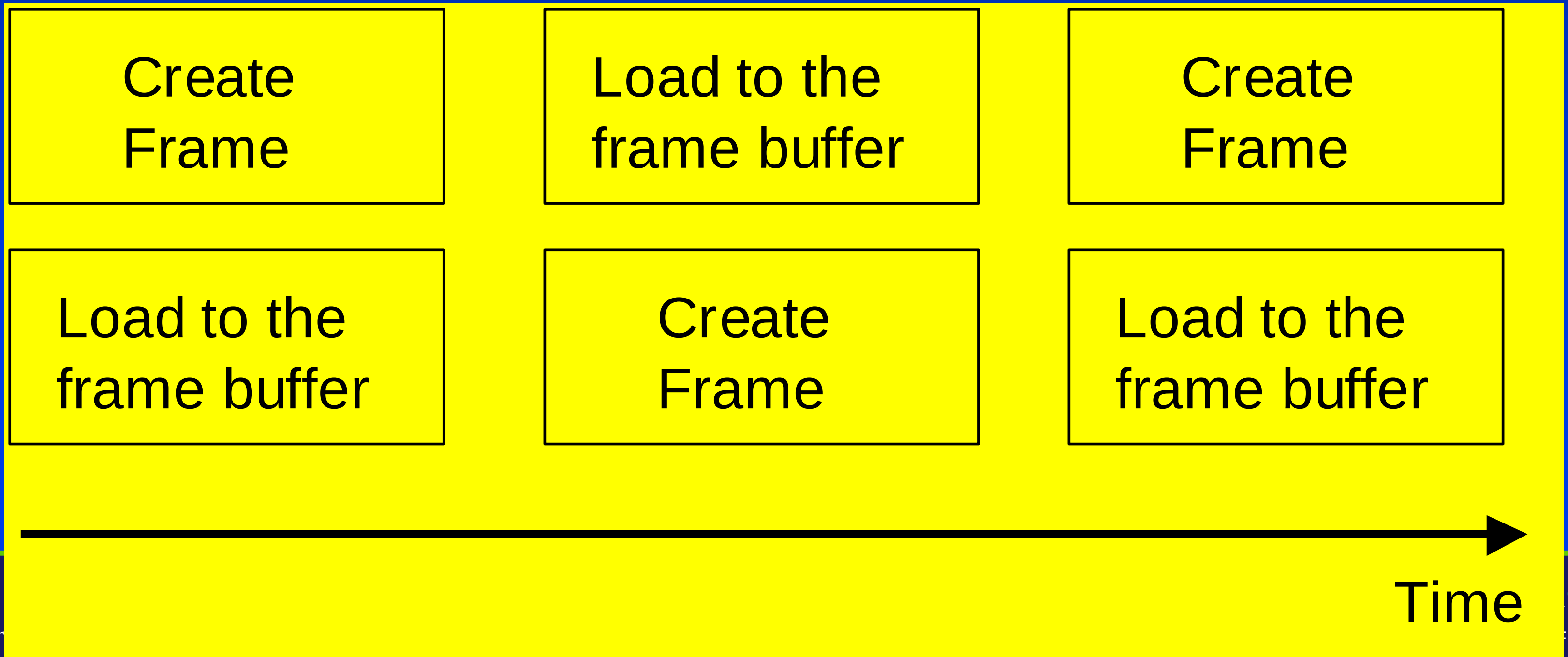
- This is the most common animation technique
- Frames are copied very fast from off-screen memory to the frame buffer
- Copying usually done with bit-block-transfer-type operations
- Copying can be applied to
  - complete frames
  - only parts of the frame which contain some movement

Raster Animation - Procedures

- A part of the frame in the frame buffer needs to be erased
- The static part of the frame is re-projected as a whole, and the animated part is over-projected.

Double Buffering

- Used to achieve smooth animation
- The next frame of animation is computed to an off-screen buffer at the same time when the current frame is transferred to the frame buffer.



Creating Animation Sequences

Object Definition

- In simple manual systems, the objects can be simply the artist drawings
- In computer-generated animations, models are used
- Examples of models:
  - a "flying logo" in a TV advertisement
  - a walking stick-man
  - a dinosaur attacking its prey in Jurassic Park

Models Can Be

- Rigid (i.e., they have no moving parts)
- Articulated (subparts are rigid, but movement is allowed between the sub-parts)
- Dynamic (using physical laws to simulate the motion)
- Particle based (animating individual particles using the statistics of behavior
- Behavior based (e.g., based on behavior of real animals)

Rigid Objects

- Simple rigid objects can be defined in terms of
  - Basic shapes such as line segments, circles, splines etc. (2D)
  - Polygon tables (3D)

- Rigid body animation is an extension of the three-dimensional viewing

Rigid Body Animation

- Rigid body animation uses standard 3D transformations
- At least 30 frames per second to achieve smooth animation
- Computing each frame would take too long

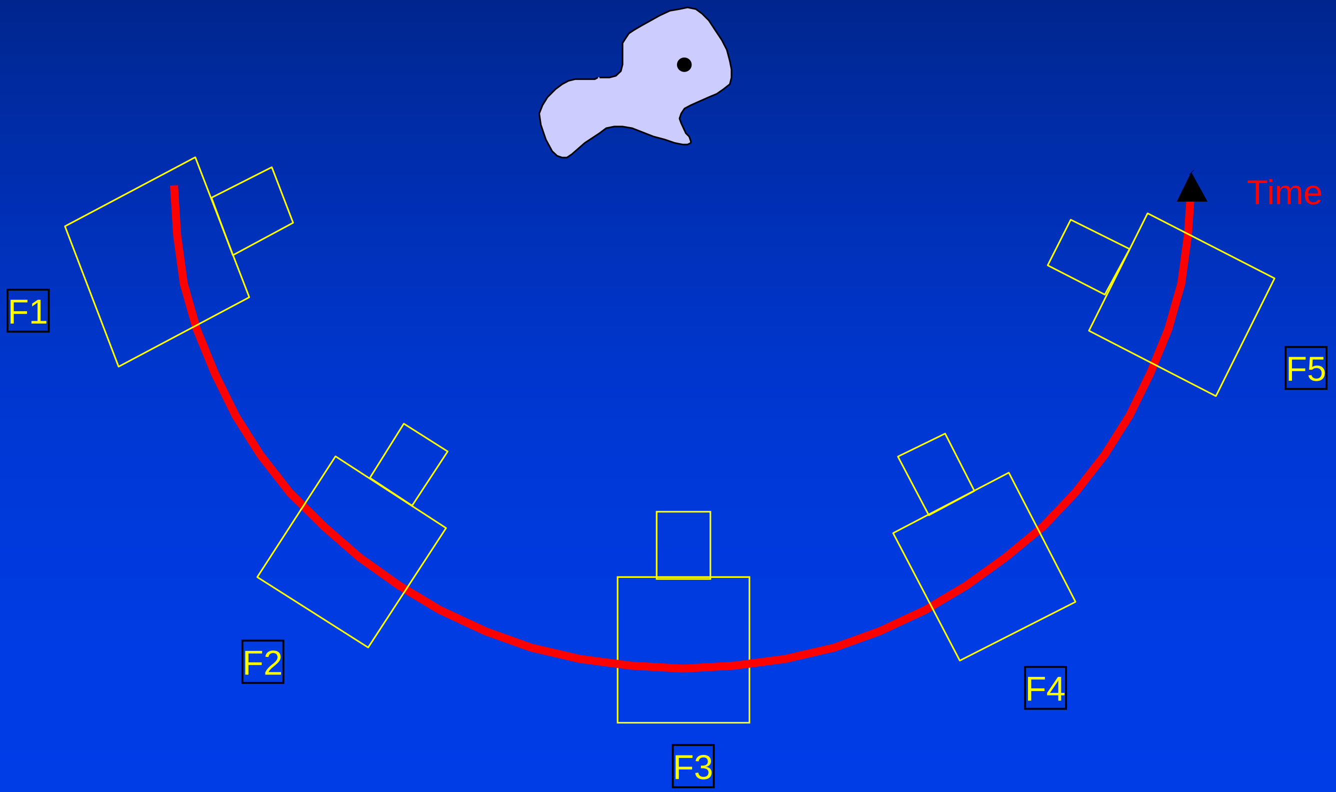
Path Specification

- Impression of movement can be created for two basic situations, or for their combination:
  - static object, moving camera
  - static camera, moving object
- The path defines the sequence of locations (for either the camera or the object) for the consecutive time frames

Static Object, Moving Camera

- The path specifies the spatial coordinates along which the camera moves
- The path is usually specified for a single point, e.g. the view reference point

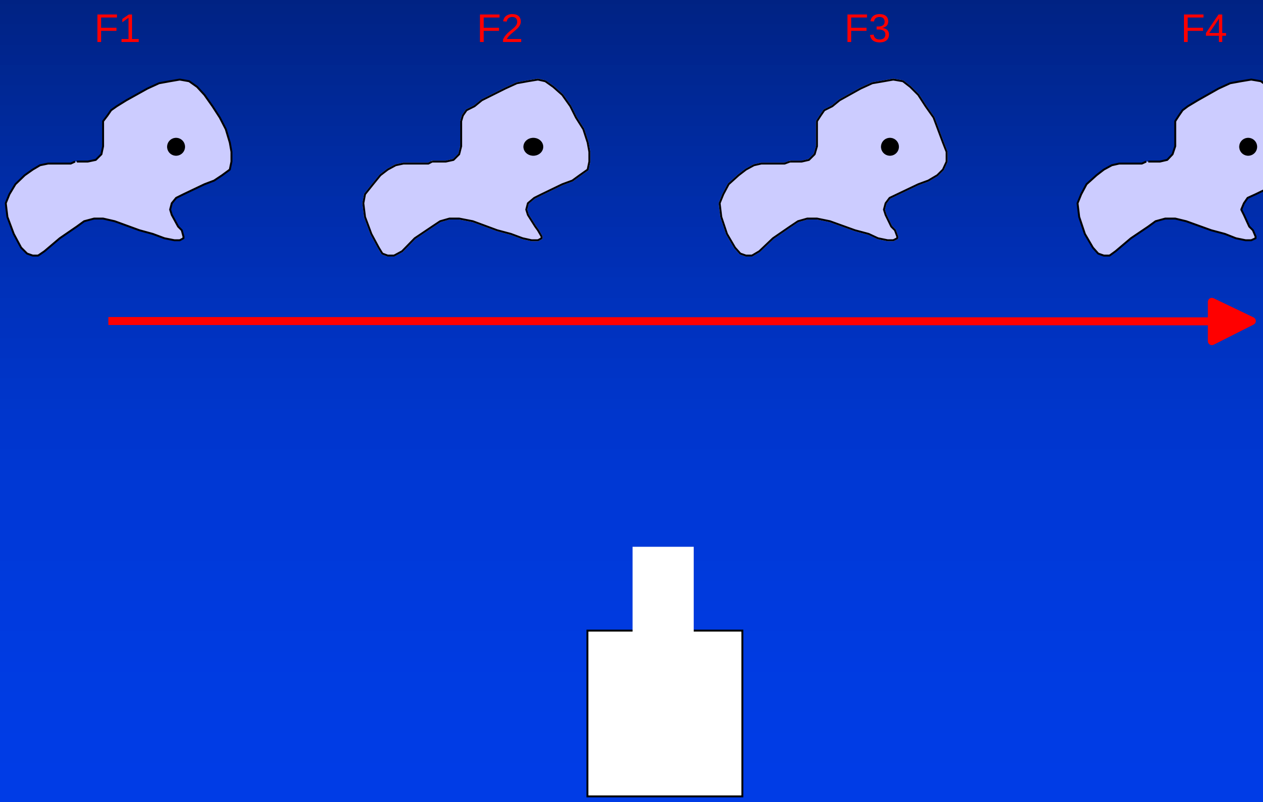
Static Object, Moving Camera



Static Object, Moving Camera

- During movement, the target point in the world coordinate system can
  - remain the same (e.g., when walking or flying around the object to see it from all directions);
  - change (e.g., standing in one location and looking round, or moving along a given path and showing the view seen by the observer while moving).

Static Camera, Moving object



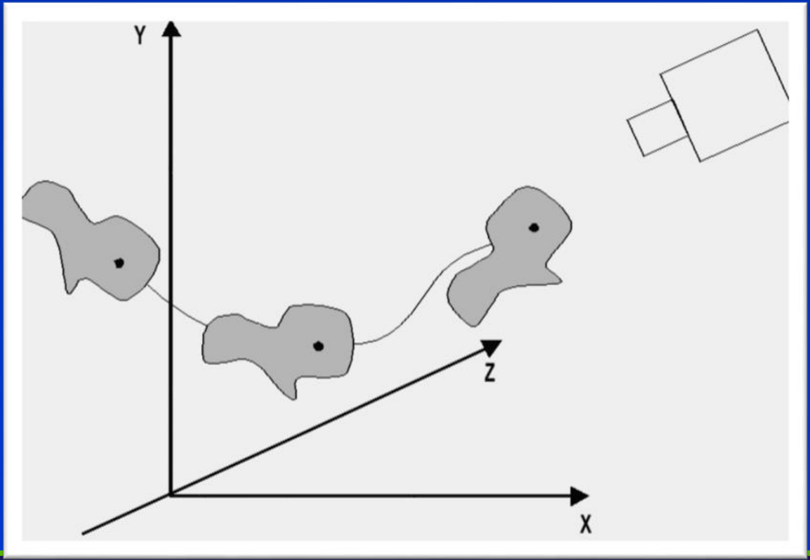
Static Camera, Moving object

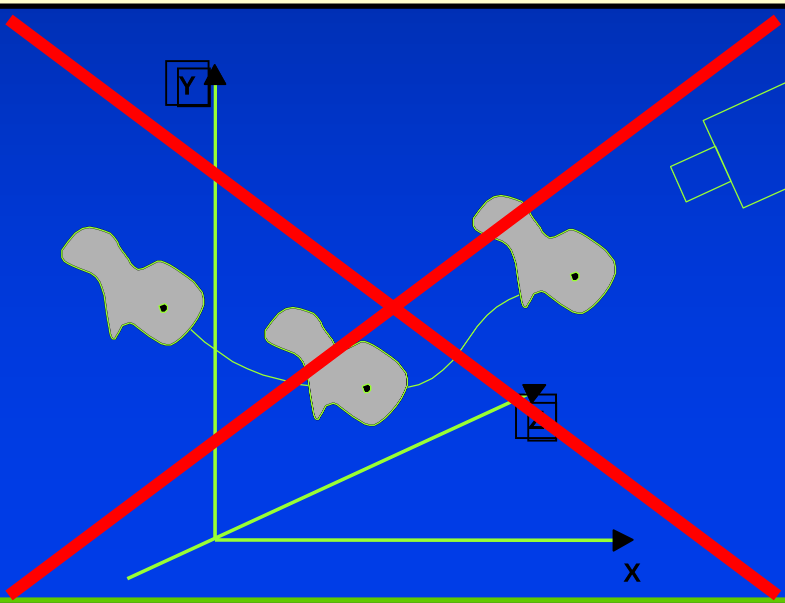
- Path specifying the object movement has to be defined
- The path is defined as the spatial coordinates along which the object moves

Static Camera, Moving Object

- Objects and their parts are defined in a local coordinate system
- Animation path is defined in the world coordinate system
- The path is specified for a single point, e.g., the center of the object's local coordinate system
- Coordinates of the actual points describing the object are calculated afterwards

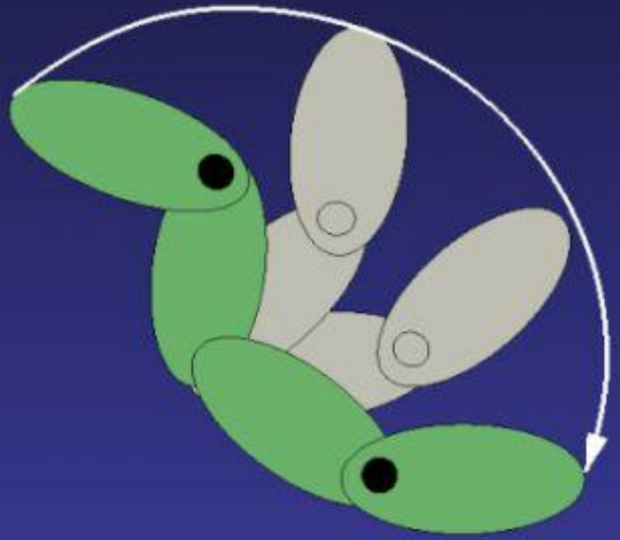
It is important to remember that when the object moves along the path, not only its position changes, but also its orientation





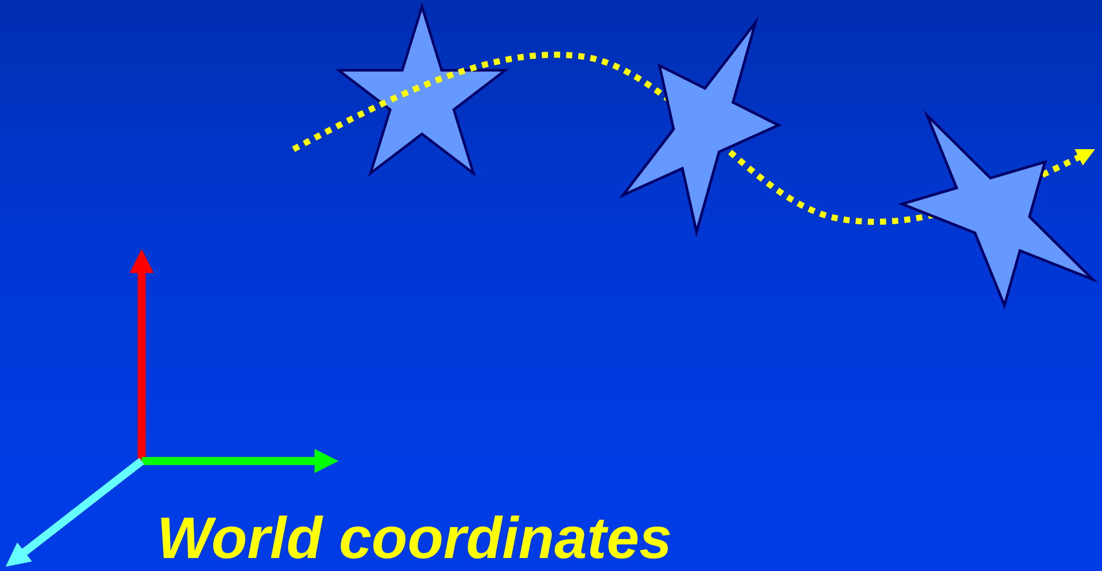
Animation Techniques

Keyframing



Motion

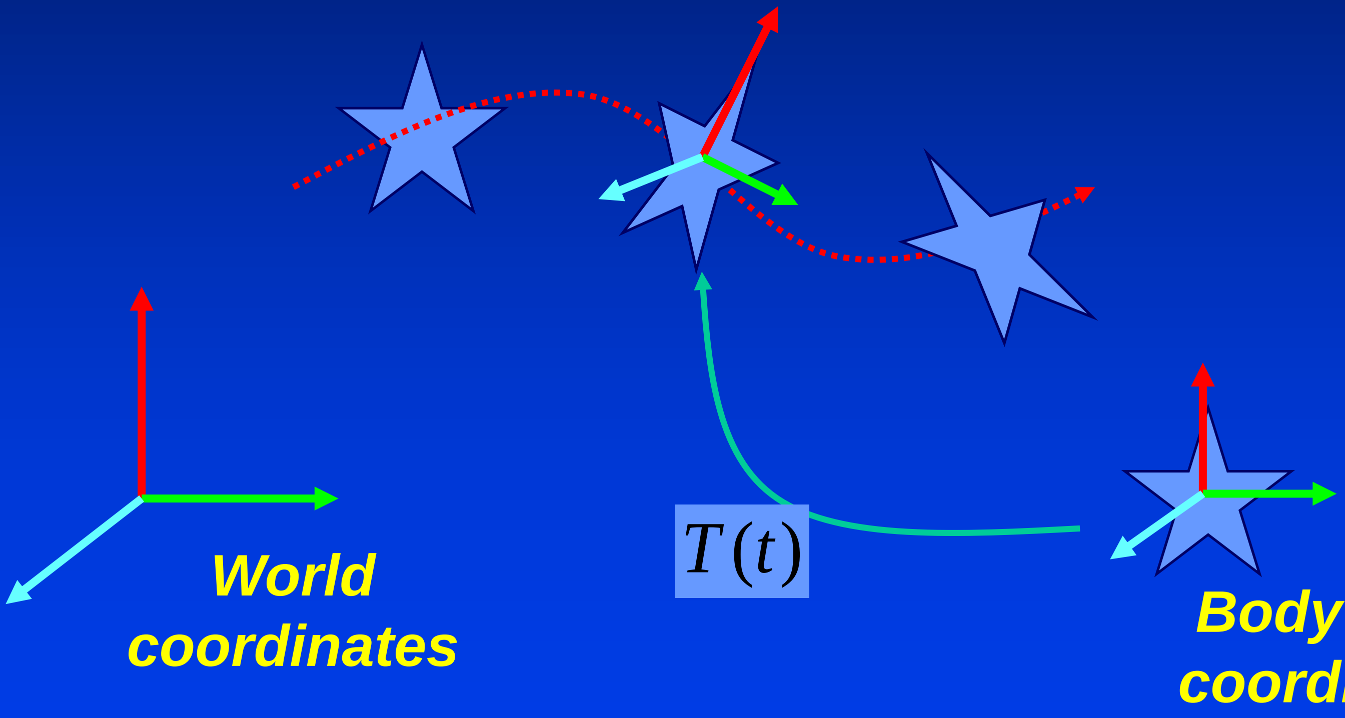
- Motion is *a time-varying transformation* from body local system to world coordinate system (in a very narrow sense)



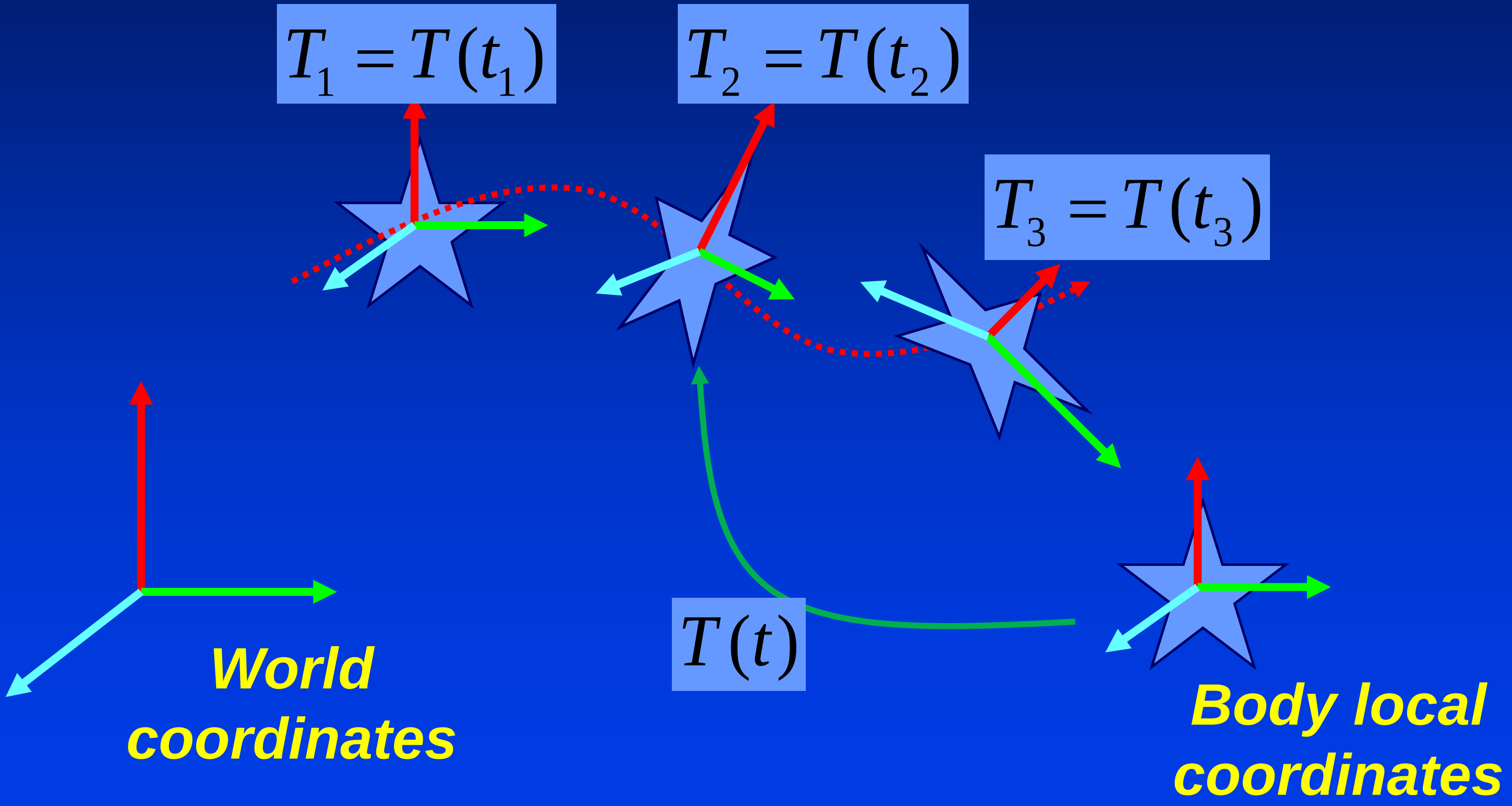
Keyframes and Inbetweening

Motion

Motion is *a time-varying transformation* from body local system to world coordinate system



Keyframing


$T_1 = T(t_1)$
$T_2 = T(t_2)$
$T_3 = T(t_3)$
$T(t)$
World coordinates
Body local coordinates

Keyframing

- *Keyframe* systems take their name from the traditional hierarchical production system first developed by Walt Disney
- Skilled animators would design or choreograph a particular sequence by drawing frames that establish the animation – these are the so-called keyframes
- The production of the complete sequence is then passed on to less skilled artists who used the keyframes to produce ‘in-between’ frames

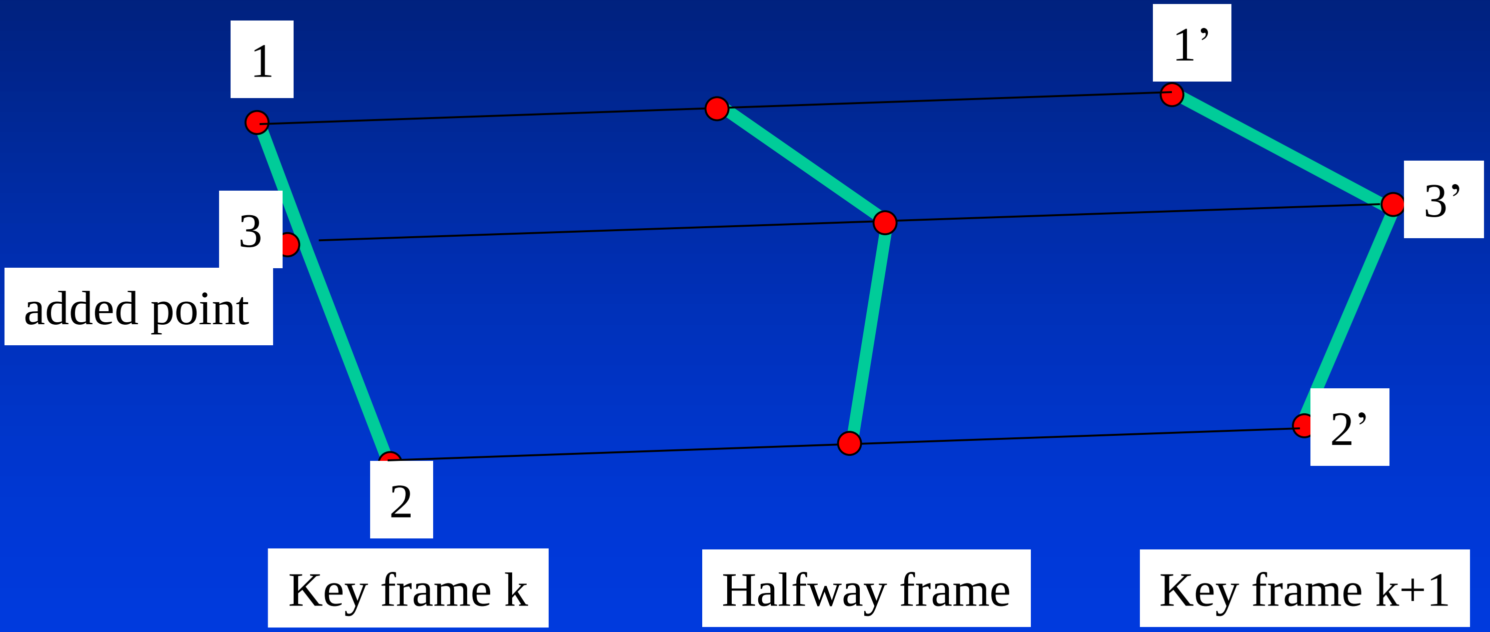
Keyframing

- The emulation of this system by the computer, whereby interpolation replaces the inbetween artist, was one of the first computer animation tools to be developed
- This technique was quickly generalized to allow for the interpolation of any parameter affecting the motion
- Special care must be taken when parameterizing the system, since interpolating naive, semantically inappropriate parameters can yield inferior motion

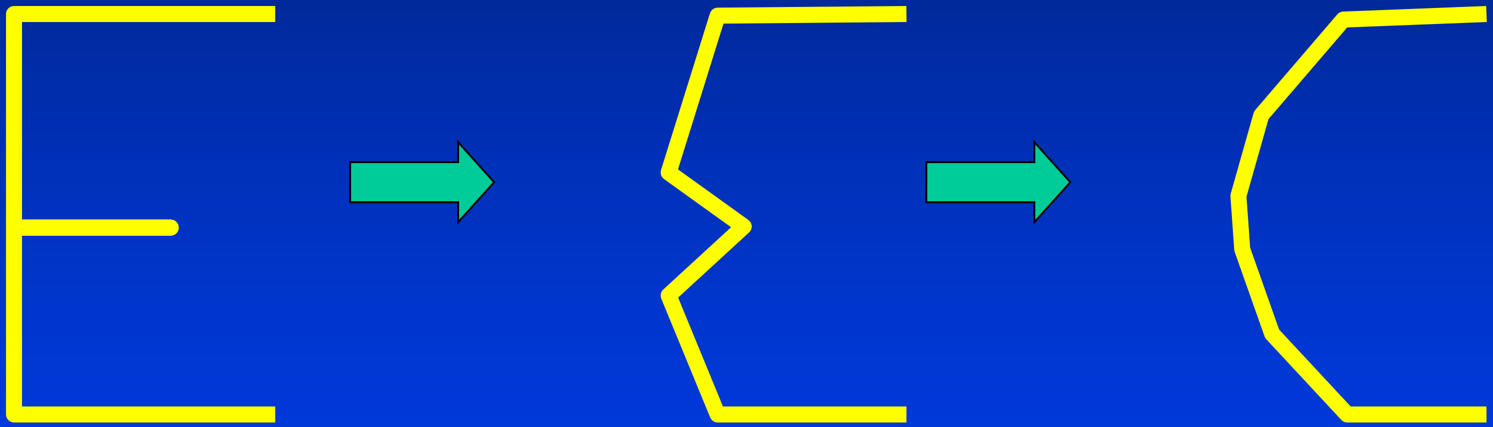
In-betweening

- The simplest method of in-betweening is linear interpolation
- Interpolation is normally applied to the projected object points

Example



Example



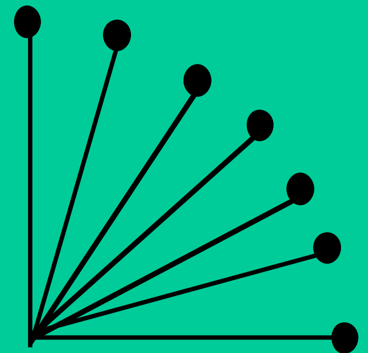
Key Frames

- First compute a small number of key frames
- Interpolate the remaining frames in-between these key frames (in-betweening)
- Key frames can be computed
  - at equal time intervals
  - according to some other rules
  - for example when the direction of the path changes rapidly

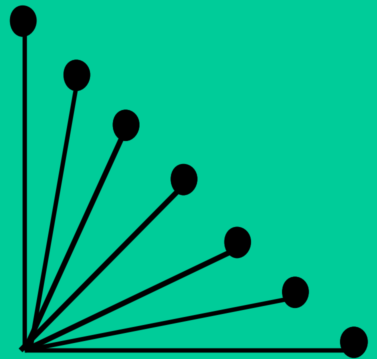
Key Frames

- The keyframing approach carries certain disadvantages
  - First, it is only really suitable for simple motion of rigid bodies
  - Second, special care must be taken to ensure that no unwanted motion excursions are introduced by the interpolant
  - Nonetheless, interpolation of key frames remains fundamental to most animation systems

Key Frame



Interpolation of rotation angle



Interpolation of end points

In-betweening

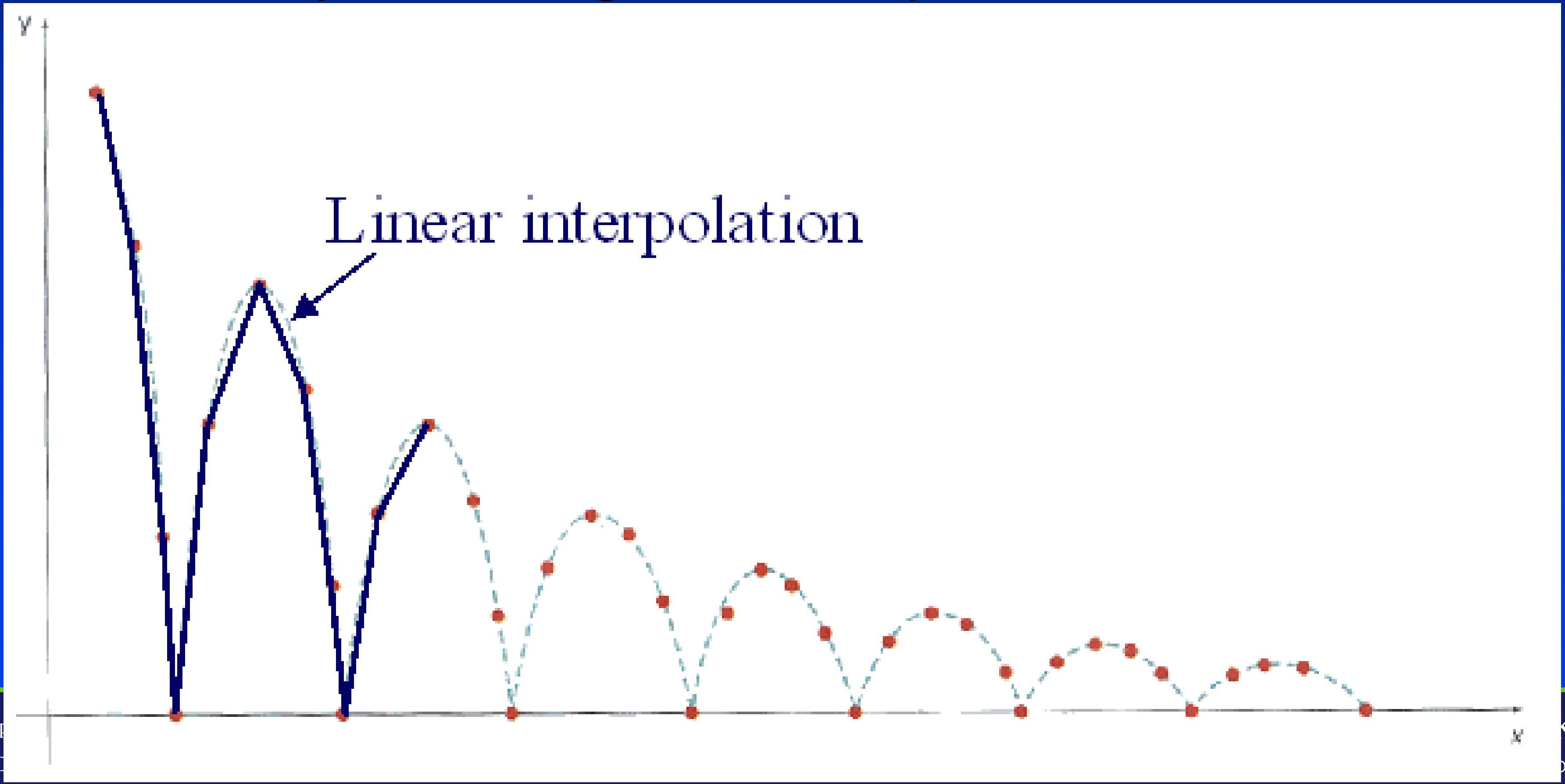
- Linear interpolation will not always produce realistic results
- Example: an animation of a bouncing ball where the best in-betweening can be achieved by dynamic animation

In-betweening

- In-betweening should use interpolation based on the nature of the path, for example:
  - straight path (linear interpolation)
  - circular path (angular interpolation)
  - irregular path linear interpolation, spline

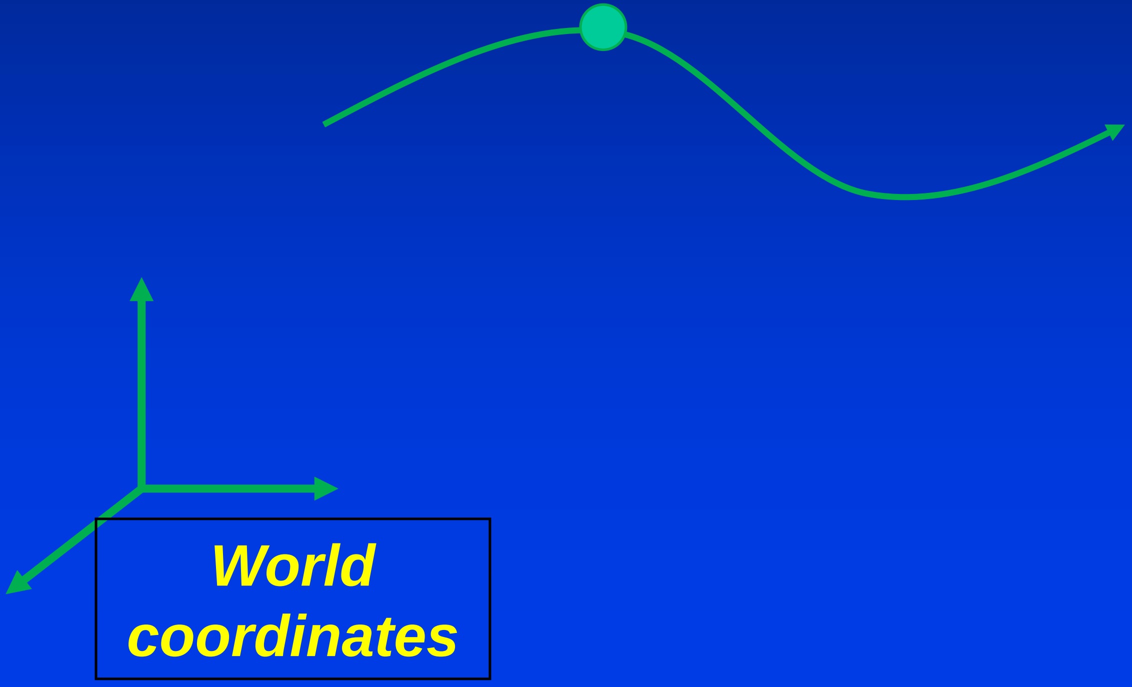
Inbetween Frames

- **Linear Interpolation**
  - Usually not enough continuity



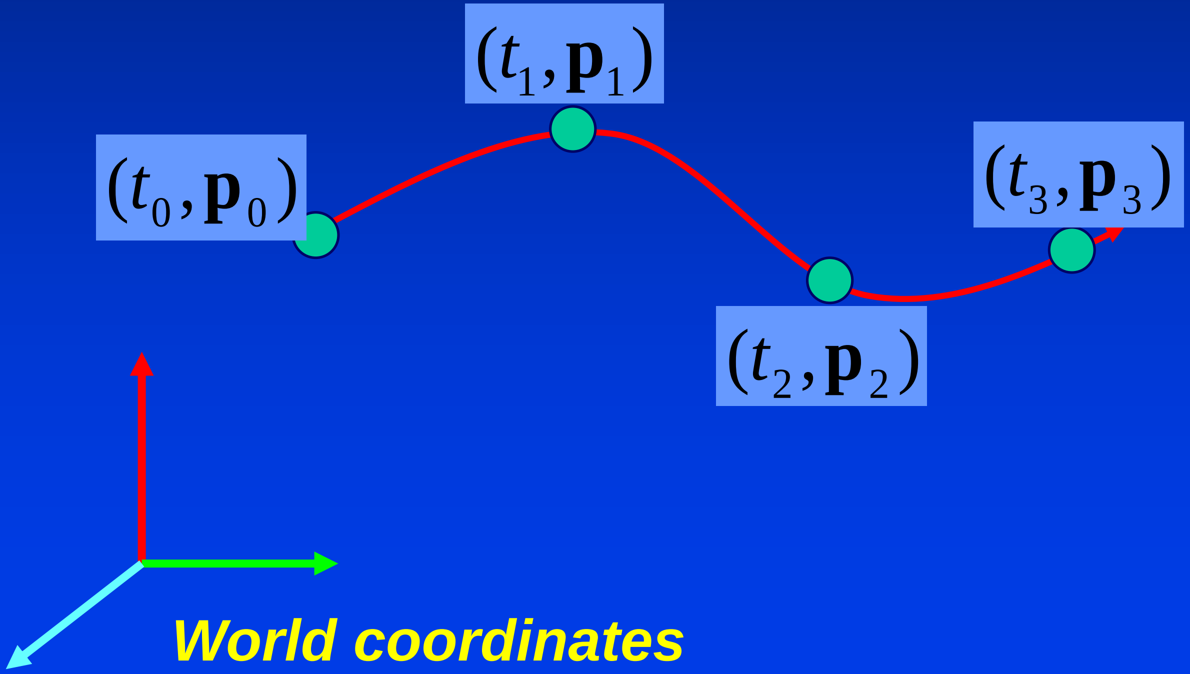
Particle Motion

$$\mathbf{p}(t) = \left(x(t), y(t), z(t)\right)$$



Keyframing Particle Motion

- Find a smooth function $\mathbf{p}(t)$ that passes through given keyframes $(t_i, \mathbf{p}_i),\ 0 \le i \le n.$



World coordinates

Image Morphing

•

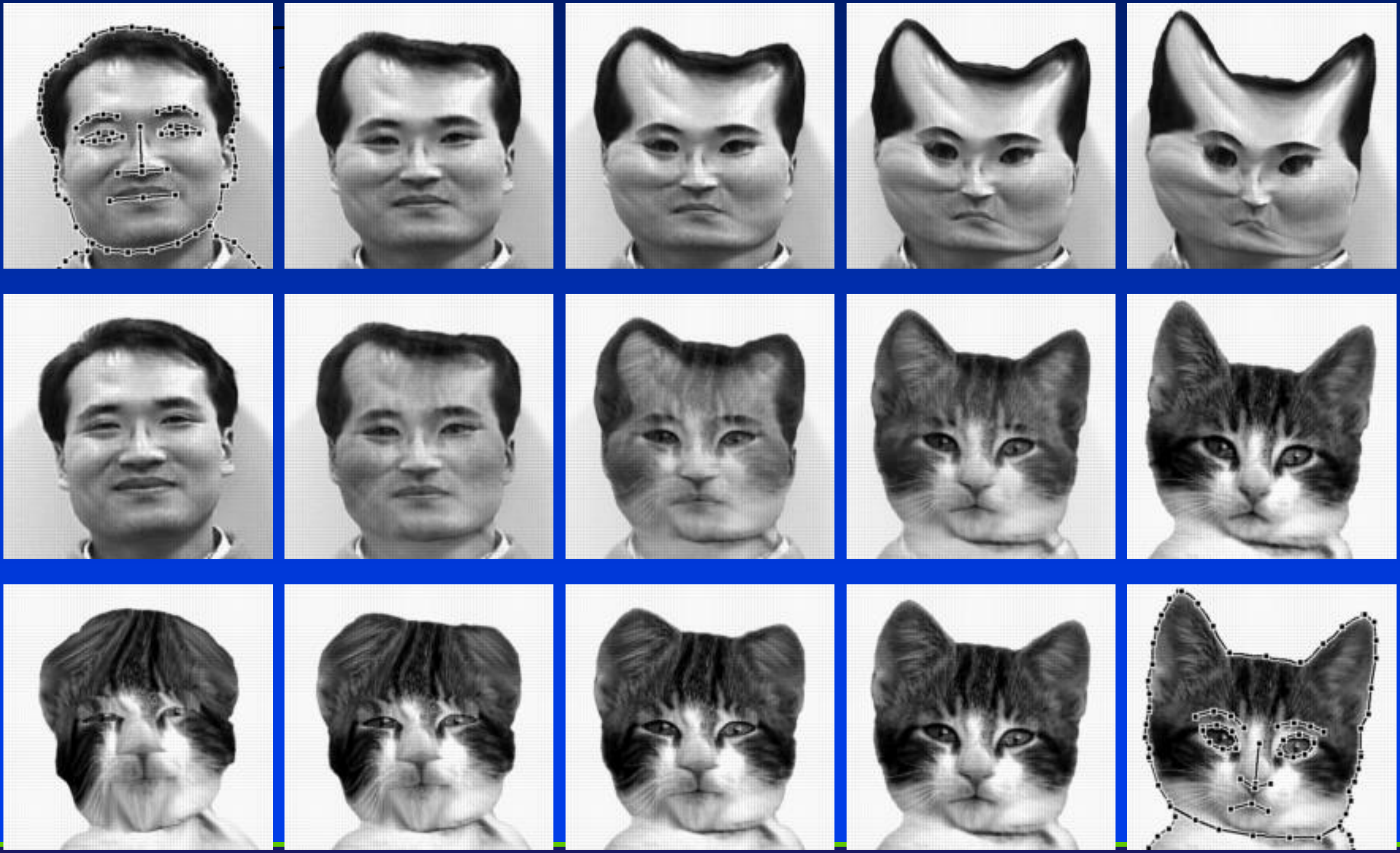
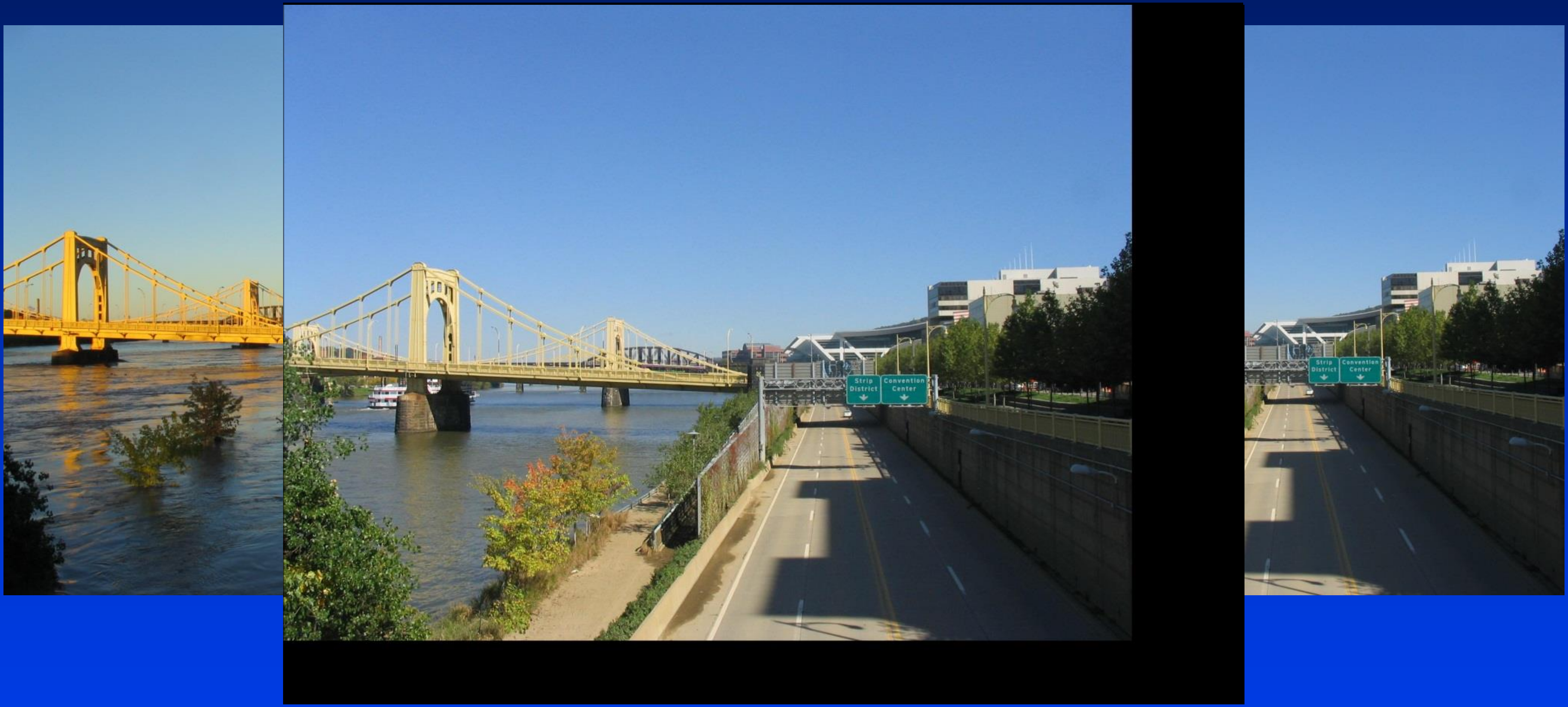
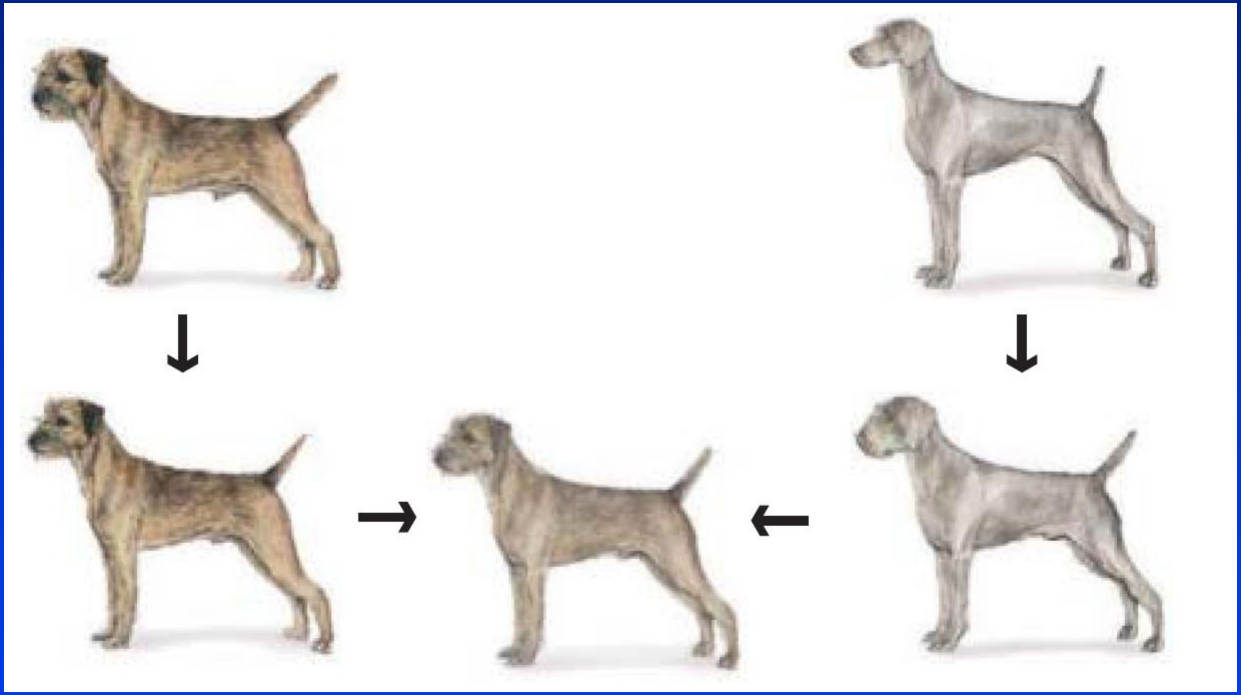


Image Morphing

Image Morphing



Animal Morphing



View Morphing

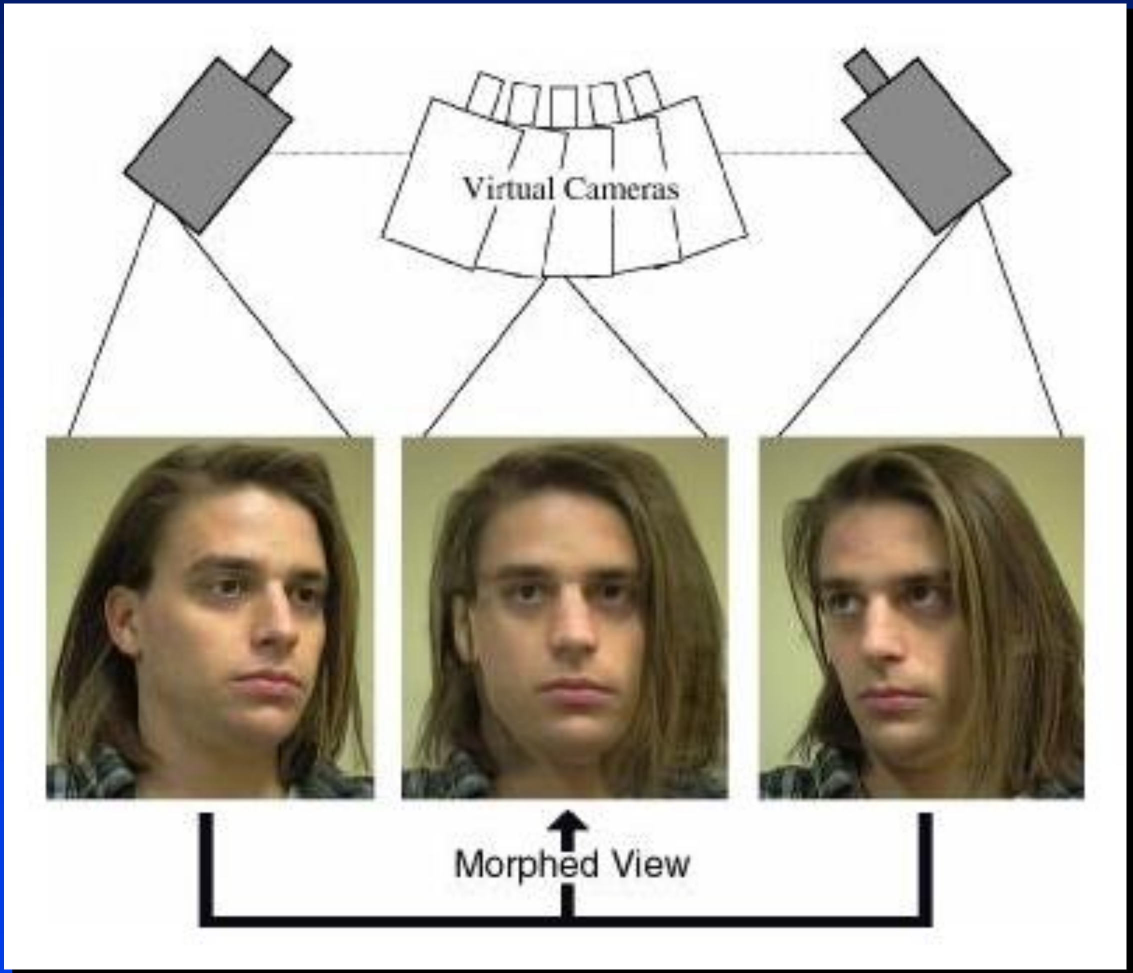
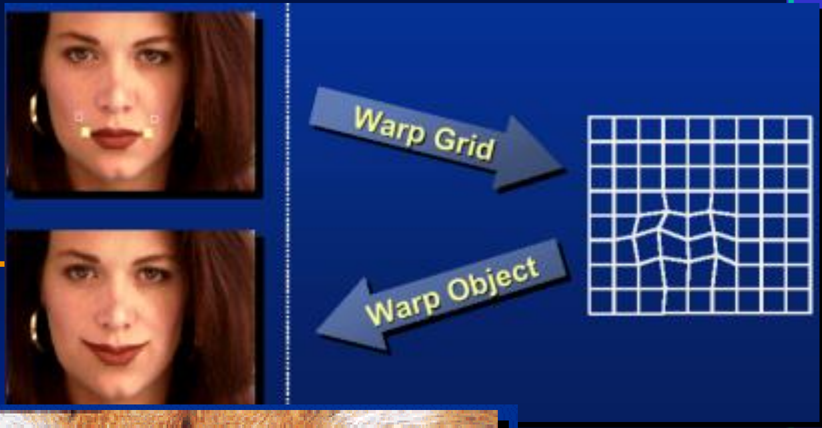
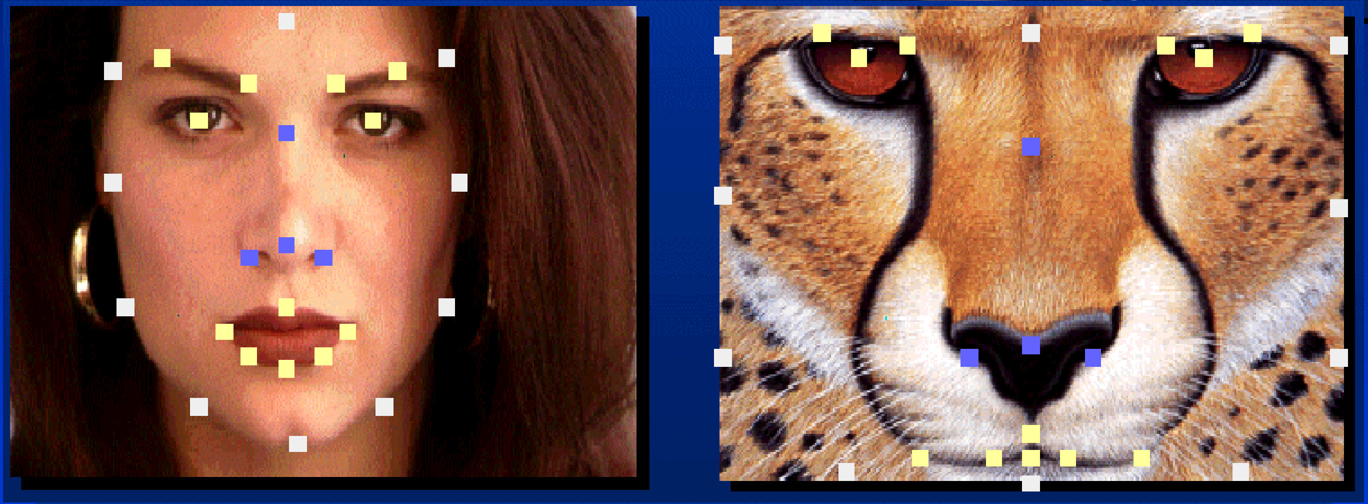
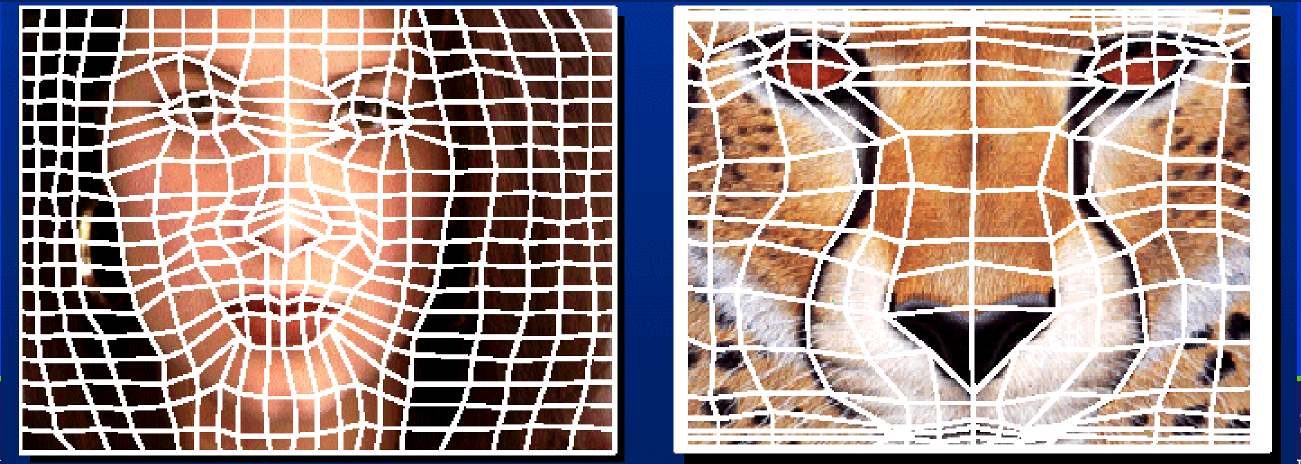


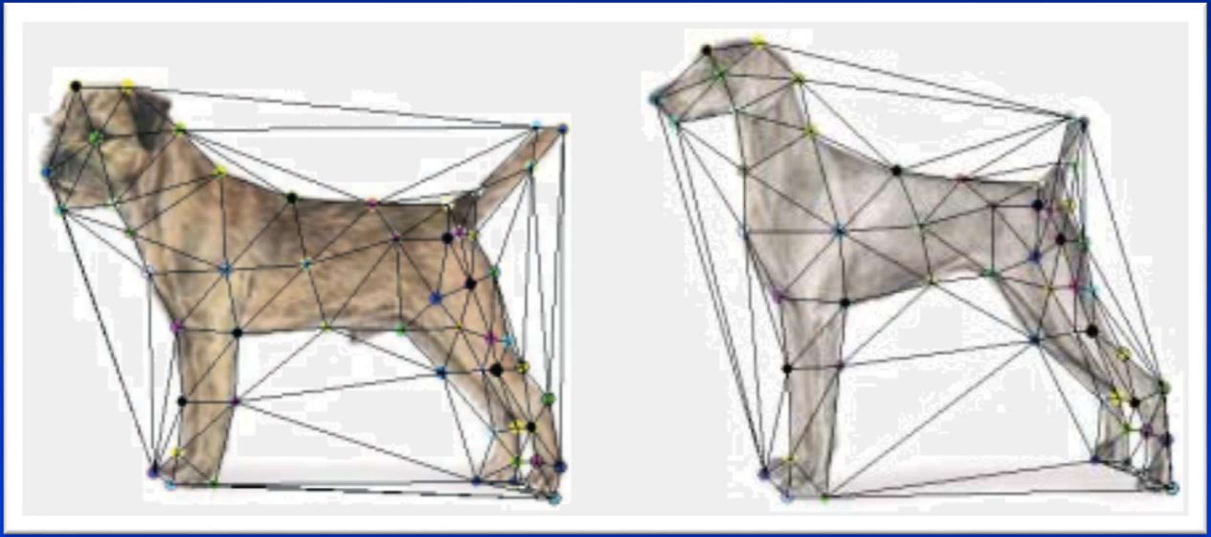
Image Warping

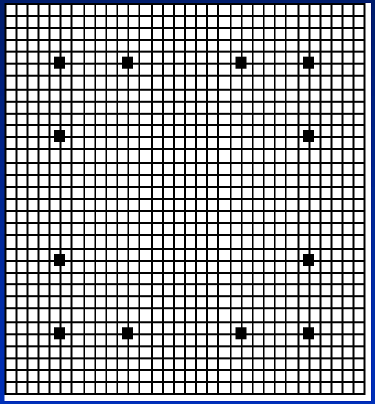


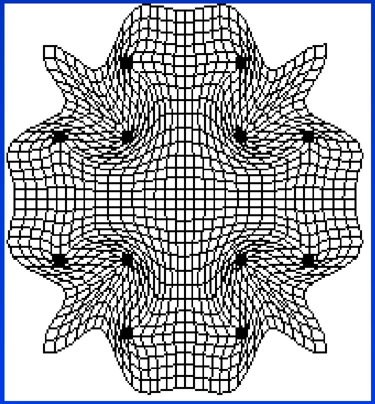




Warp Interpolation

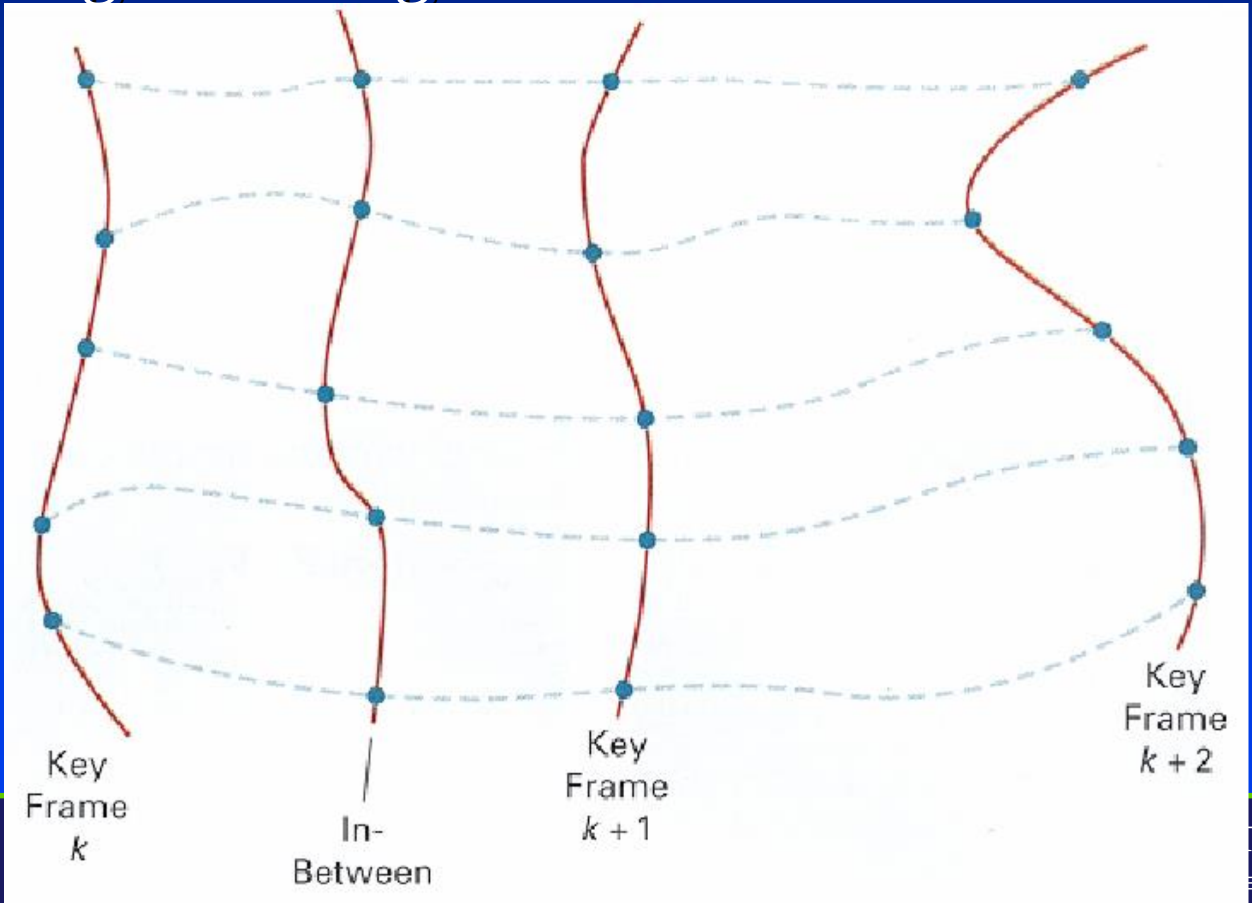






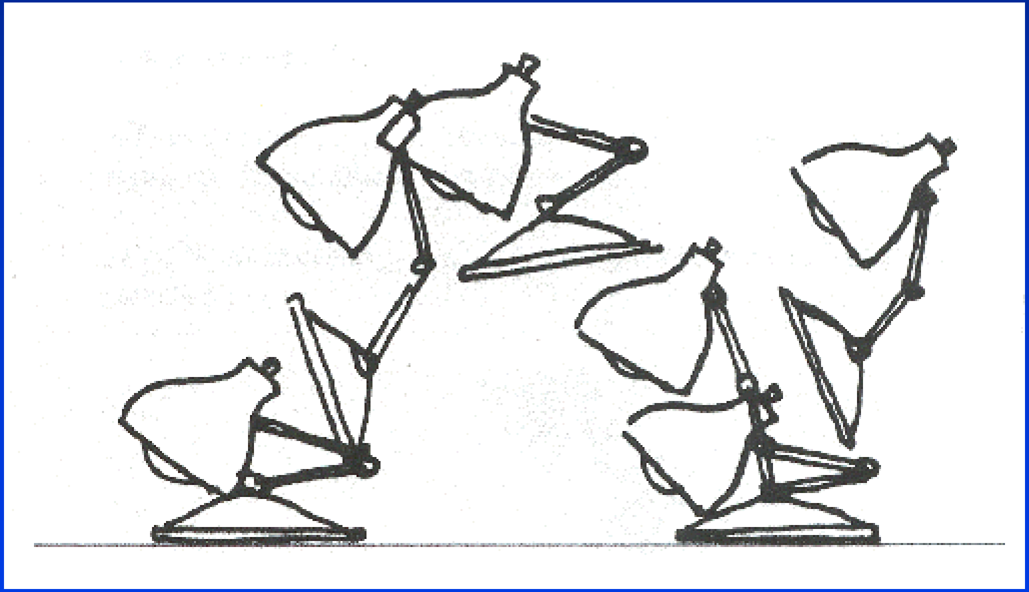
Inbetween Frame

- **Spline Interpolation**
  - Maybe good enough



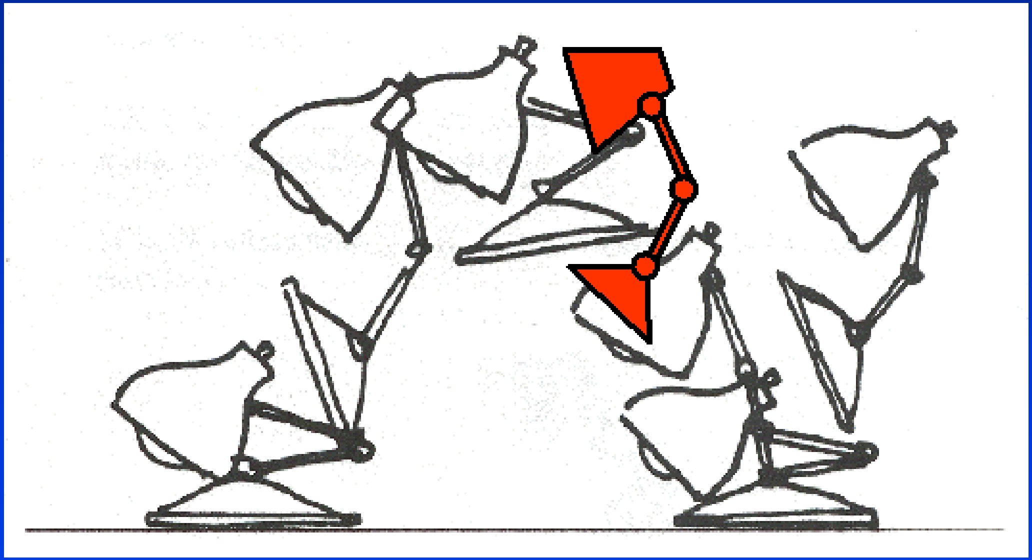
Keyframe Animation

- **Define Character Poses at Specific Time Steps Called “Keyframes”**



Keyframe Animation

- **Interpolate variables that are describing keyframes to determine poses for character in between**



Spline-Driven Animation

- *Spline-driven animation* means the explicit specification of the motion characteristic of an object by using cubic splines
- Cubic B-splines are composite curves made up of several curve segments
- The curve possesses second-order continuity
- These cubic splines are commonly used in computer graphics

Spline-Driven Animation

- Consider a single segment of the curve defined over the interval $0 \leq u \leq 1$
- The curve is a cubic polynomial which can be specified interactively by defining four control points
- The particular curve that passes through these points is constrained by the need for second-order continuity at the end points of the curve segments
  - Adjacent curve segments share three control points

Spline-Driven Animation

- Using this information we can derive mathematically the exact form of each of the curve segments as follows:
  - $Q_i(u)$ = sum from k=0 to 3 $p_{i+k}B_k(u)$

where the p_i's are the control points, and the B_i's are defined as follows:

  - $B_0(u) = (1+u)^3/6$
  - $B_1(u) = (3u^3-6u^2+4)/6$
  - $B_2(u) = (-3u^3+3u^2+3u+1)/6$
  - $B_3(u) = u^3/6$

Spline-Driven Animation

- Then the curve is re-parameterized in terms of a global variable u
- If the ends of the curve segments occur at equal intervals with respect to the curve parameter, the curve is known as a uniform B-spline

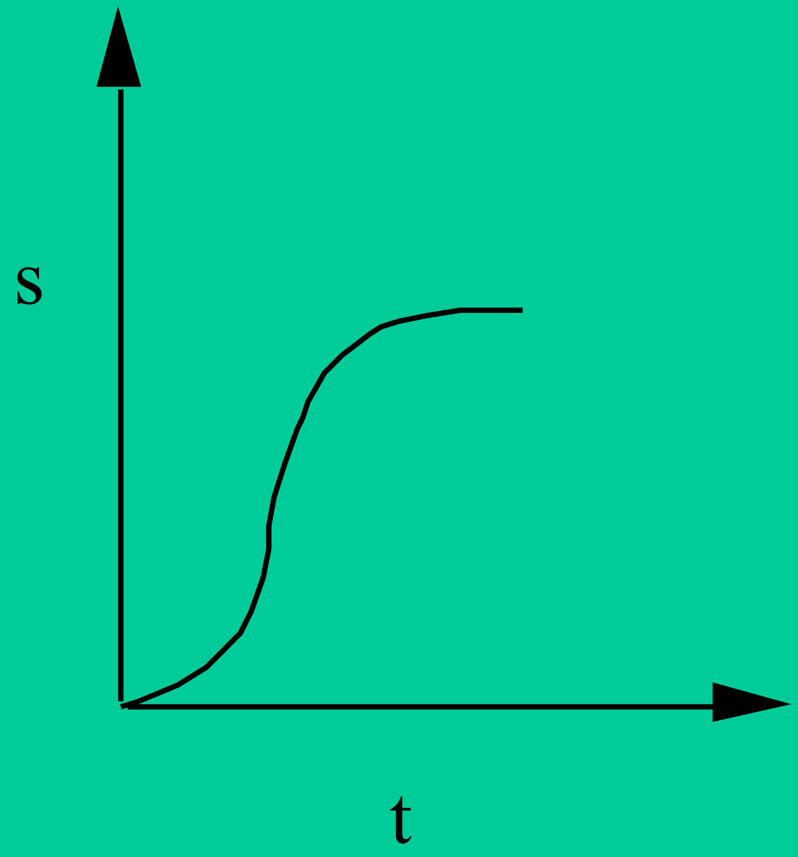
Spline-Driven Animation

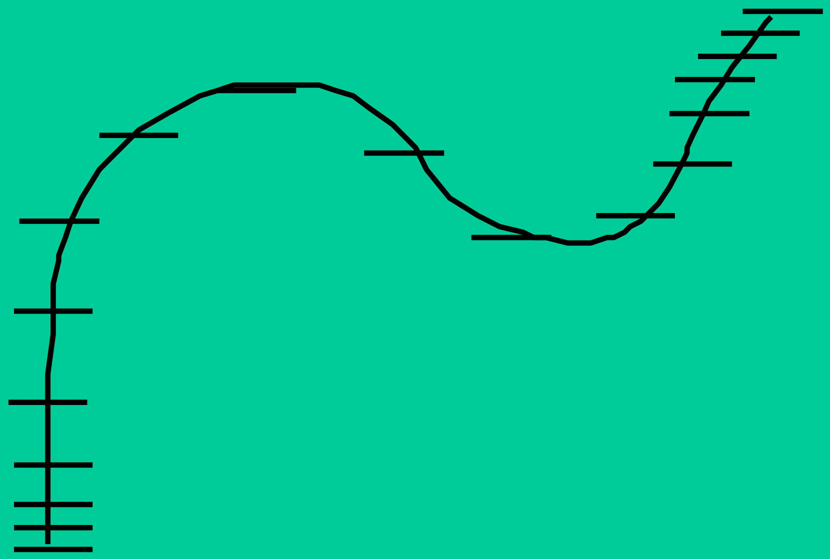
- Suppose we have interactively specified a spline q(u) (by giving four control points) that we wish to use as the path for the motion of an object
- To generate an animation sequence, we need to find the position of the object along the path at equal intervals in time

Spline-Driven Animation

- In order to do this, we need to re-parameterize the curve in terms of arclength parameterization
  - Without the arclength parameter, it is not possible to have an object move with uniform speed along a spline
  - The re-parameterization is nontrivial and will not be given here
  - Once this has been done, an object positioned on a curve q(u) can be driven by a velocity curve
    - v(u) = (t(u), s(u)) that plots the arclength s, or distance traveled, against time

Ease-in, Ease-out Velocity Curve with Space Curve



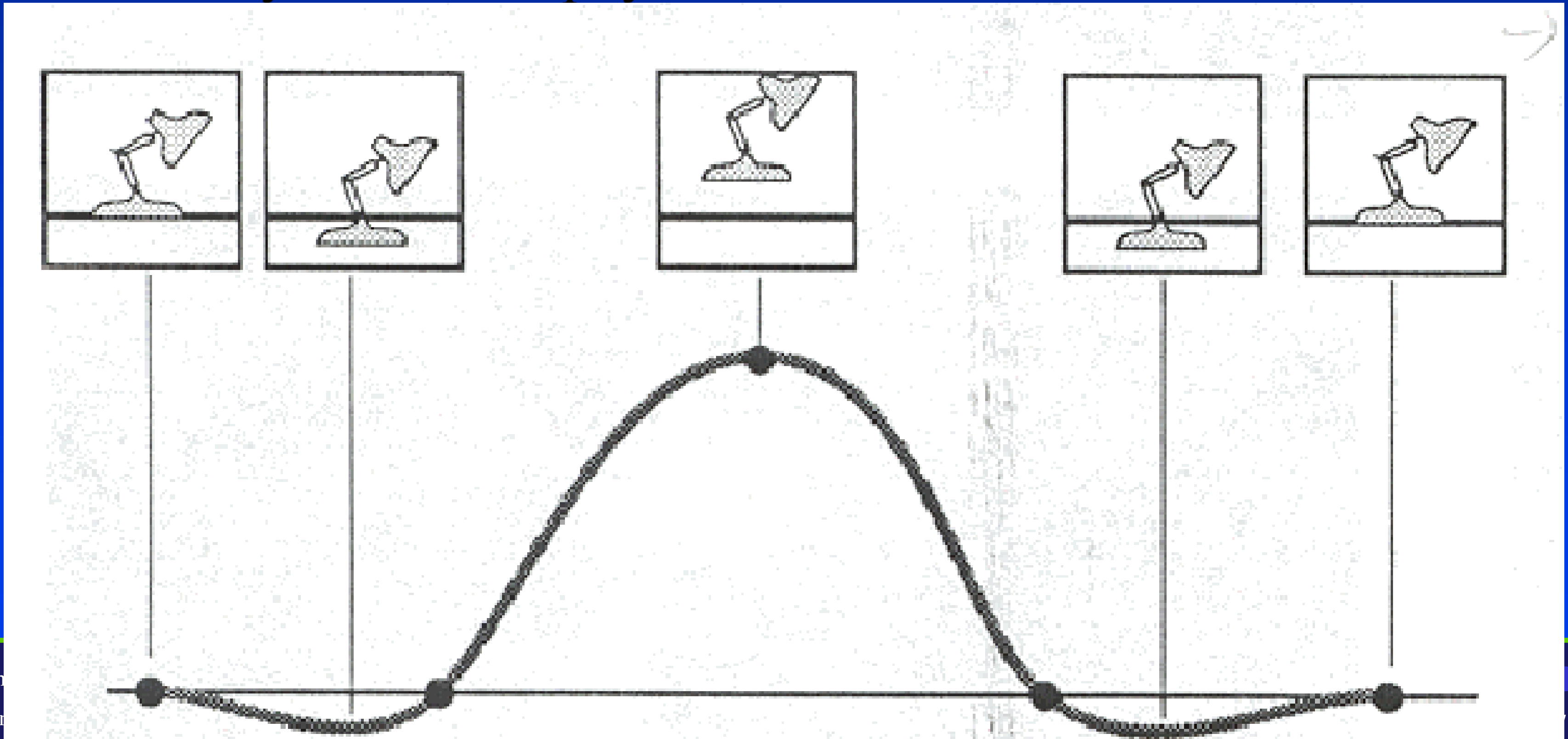


Spline-Driven Animation

- The velocity curve can be generalized to drive any motion parameter
  – The term ‘motion parameter’ then encompasses anything that moves in the animation sequence apart from the usual kinetic variables such as position and orientation
  – Movement could also include color and transparency, for example
  – This methodology is known as general kinetic control

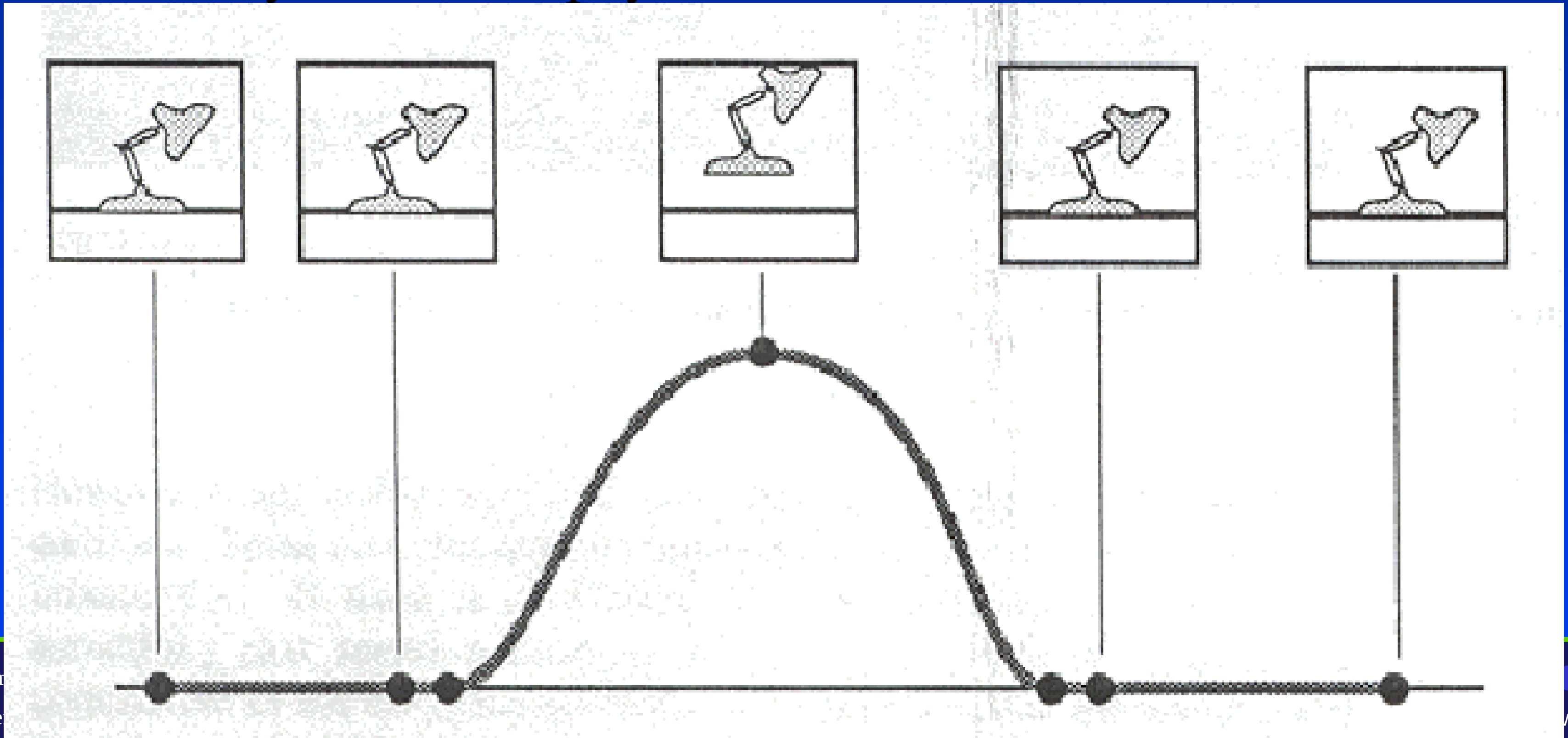
Inbetween Frames based on Spline

- **Spline Interpolation**
  - Maybe good enough
    - May not follow physical laws



Inbetween Frames

- **Spline Interpolation**
  - Maybe good enough
    - May not follow physical laws



Animation Challenges

- Animating one rigid object with 6 degrees of freedom for 5 seconds at 30 frames per second requires 9000 numbers to be interactively specified
- A fully defined human figure will have more than 200 degrees of freedom
- A control hierarchy reduces the number of degrees of freedom (DOFs) that the animator has to specify
  - High-level constructs are mapped to lower-level data

Outline

- **Principles of animation**
- **Keyframe animation**
- **Articulated figures**

Animation Control

Computer Animation Control

- An animation system may be high-level, low-level, or somewhere in between
  - High-level animation systems allow the animator to specify the motion in general, abstract terms
  - High-level commands describe behavior implicitly in terms of events and relationships
  - Low-level systems requires the animator to specify individual moving parameters

Medium-Level Animation Control

- **Medium-level** animation techniques may generally be placed in one or more of the following categories
- *Representational animation*
  - not only can an object move through space, but the shape of the object itself may change
- *Procedural animation*
  - control over motion specification achieved through use of procedures that explicitly define the movement as a function of time

Medium-Level Animation

- There are two subsections of this category:
  - The animation of *articulated objects*
    - An articulated object is made up of connected segments or links whose motion relative to each other is somewhat restricted
  - *Soft object animation*
    - This includes the more general techniques for deforming and animating the deformation of objects

Medium-Level Animation Control

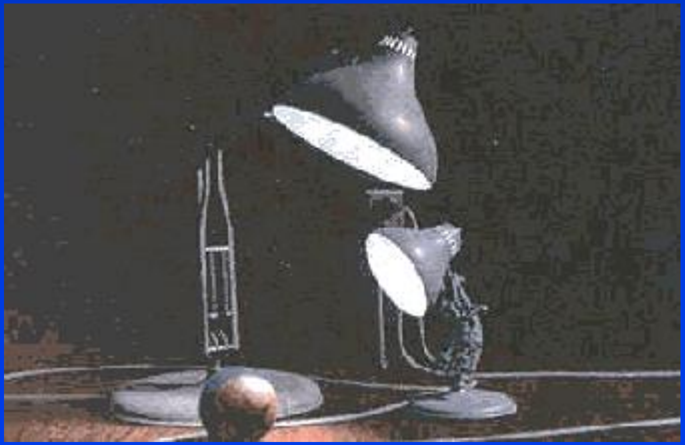
- *Stochastic animation* controls the general features of the animation by invoking stochastic processes that generate large amounts of low-level detail
  - This approach is particularly suited to particle systems
- In *behavioral animation*, the animator exerts control by defining how objects behave or interact with their environment

Low-Level Control

- We now examine some of the techniques that, under the paradigm of animation as hierarchy of control, corresponds to the different ways of imposing the first level of abstraction on the task of motion control

Computer Animation

- **Animation pipeline**
  - 3D modeling
  - Motion specification
  - Motion simulation
  - Shading, lighting, & rendering
  - Postprocessing

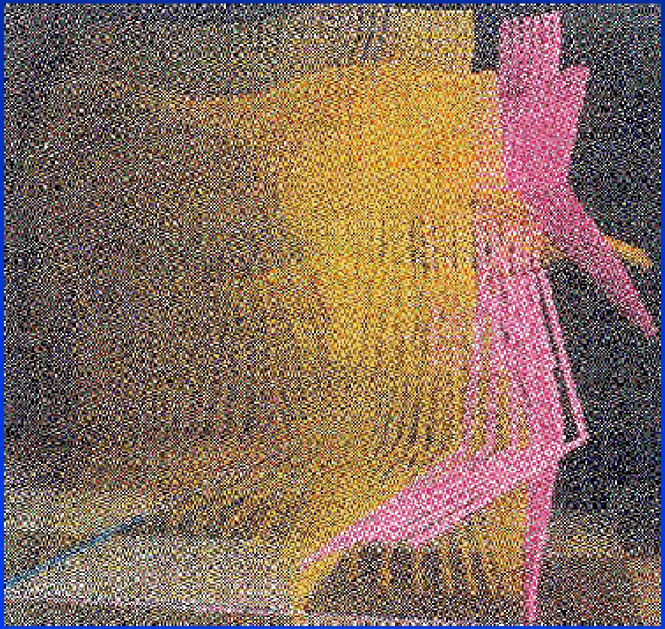


Outline

- Principles of Animation
- Keyframe Animation
- Articulated Figures

Inbetween Frames

- **Inverse kinematics or dynamics**
- **Articulated figures**



Animating Articulated Figures

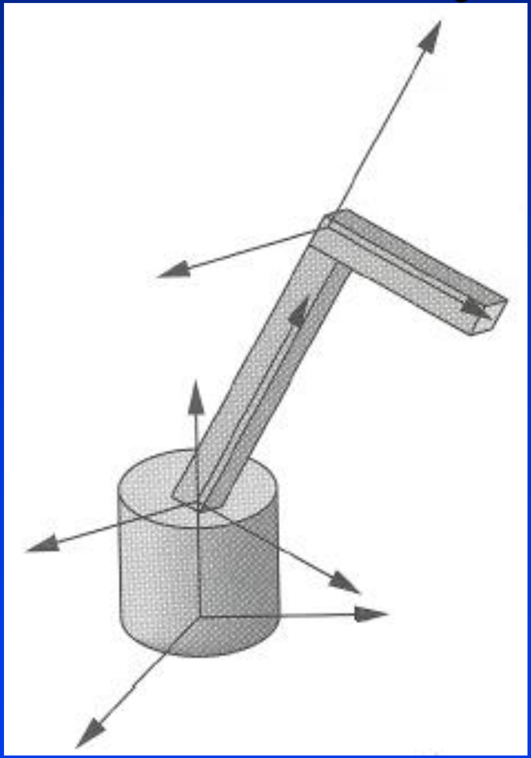
- Older animation systems are keyframe-based
- Newer animation systems use *forward kinematics* and *inverse kinematics* to specify and control motion
- The characters themselves are constructed out of skeletons which resemble the articulated structures similar in robotics

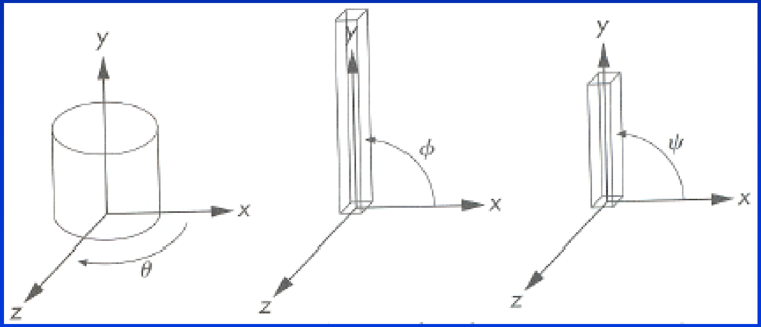
Articulated Figures

- **Character poses are described by a set of rigid bodies connected by “joints”**

$$\begin{tikzcd}\boxed{\text{Base}} \arrow[d] \\ \boxed{\text{Arm}} \arrow[d] \\ \boxed{\text{Hand}}\end{tikzcd}$$

Scene Graph





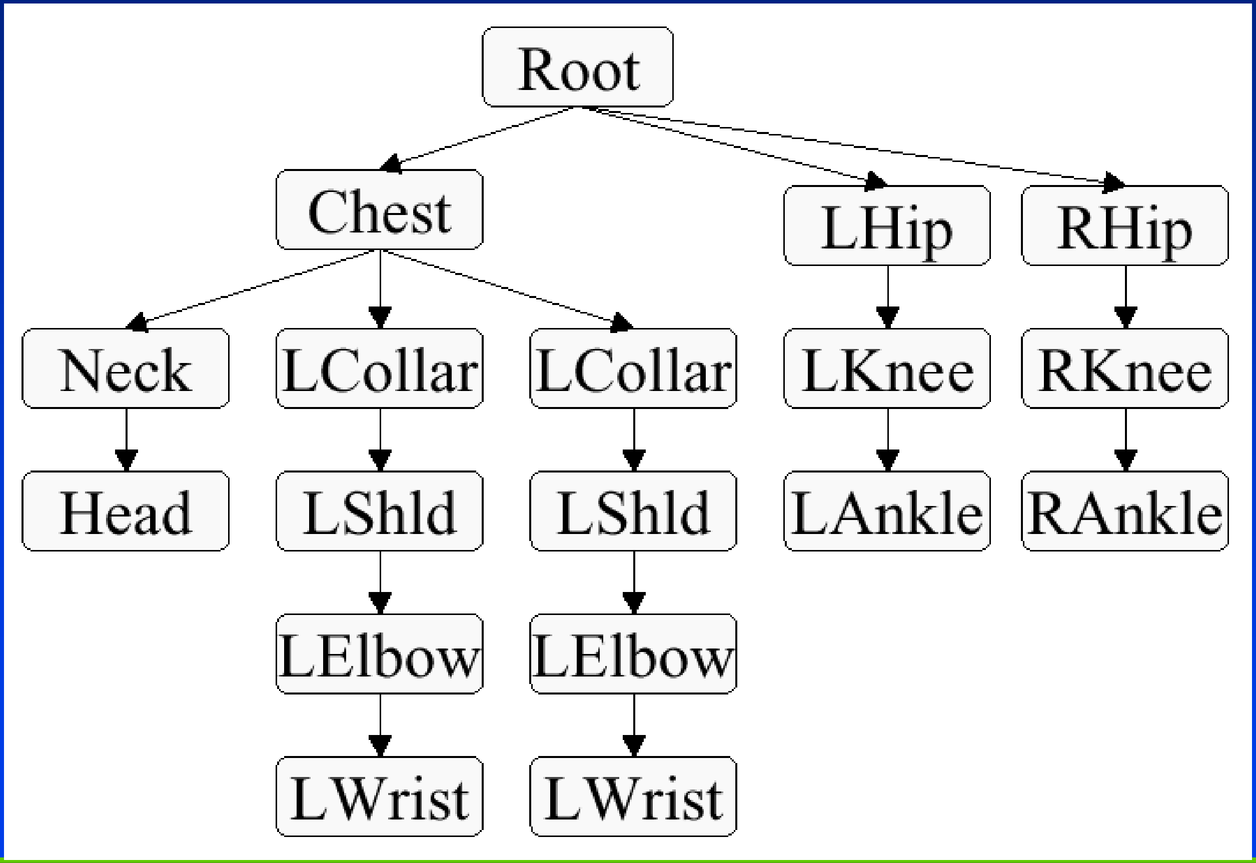
Animating Articulated Figures

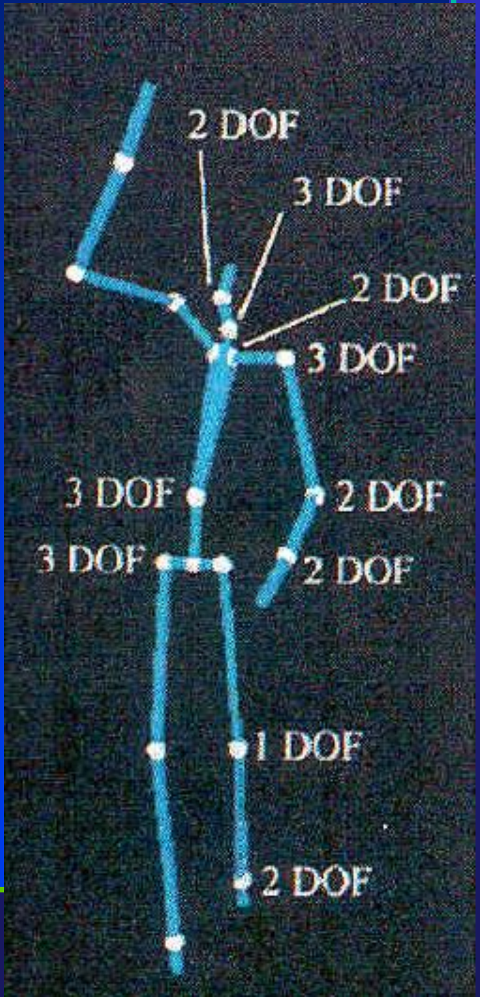
Some definitions

- **Kinematics:** The study of motion independent of forces producing the motion
- **Articulated figure:** A structure consisting of rigid links connected at joints
- **Degrees of freedom (DOFs):** The number of independent joint variables specifying the state of the structure
- **End effector:** The end of a chain of links, i.e. a hand or a foot
- **State vector:** The set of independent parameters which define a particular state of the articulated structure, thus the state vector Q is (q1, q2, ..., qn) where it has n degrees of freedom

Articulated Figures

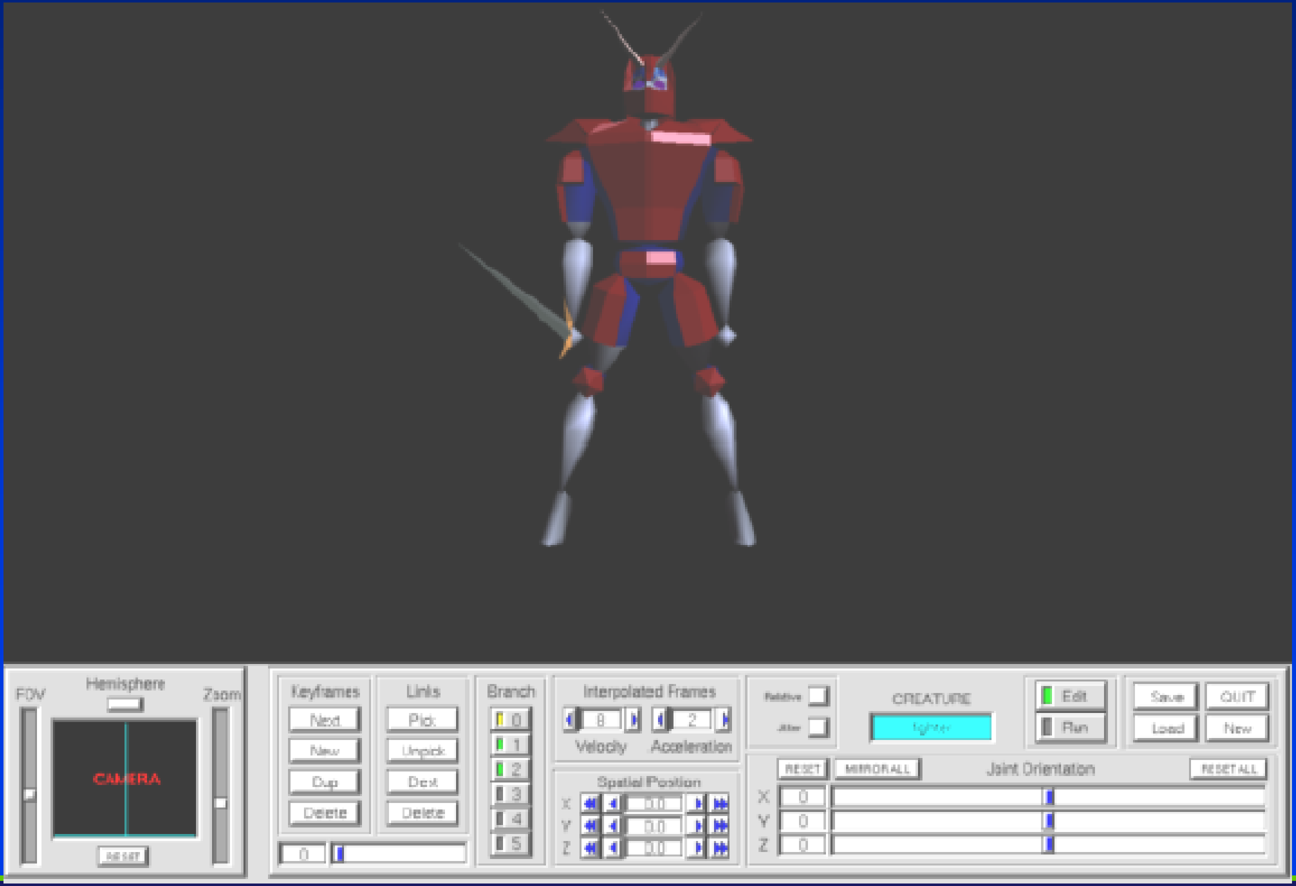
- **Well-suited for humanoid characters**





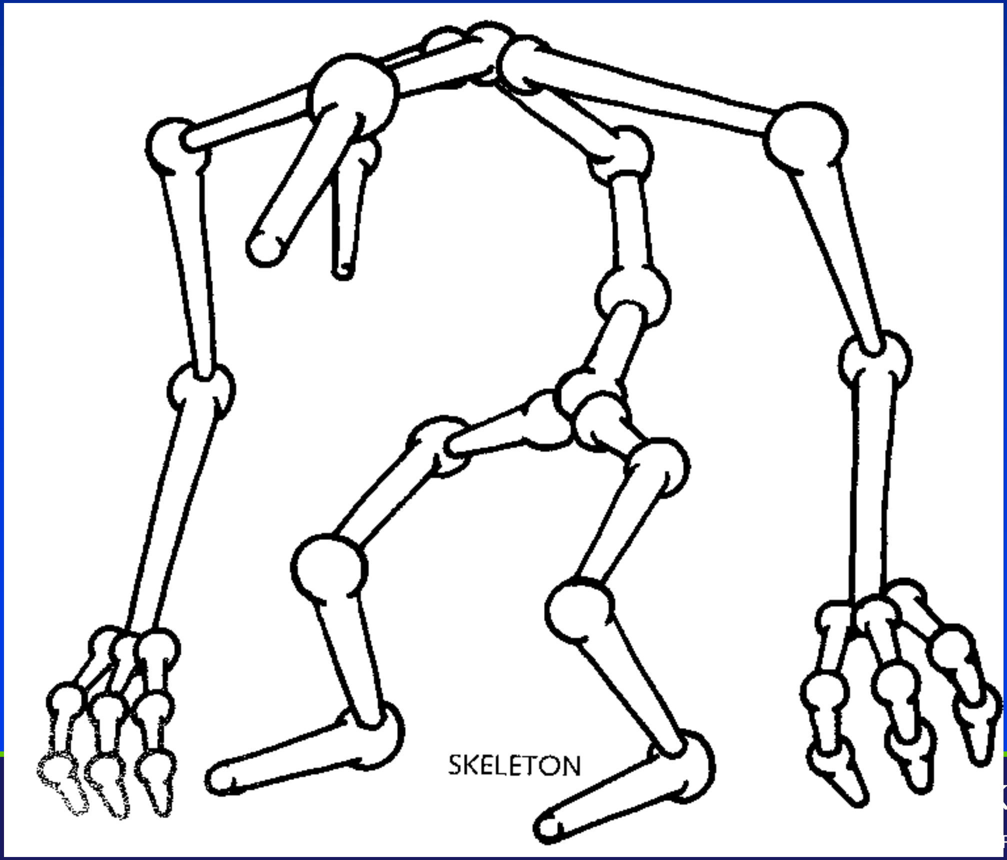
Articulated Figures

- **Joints provide handles for moving articulated figures**

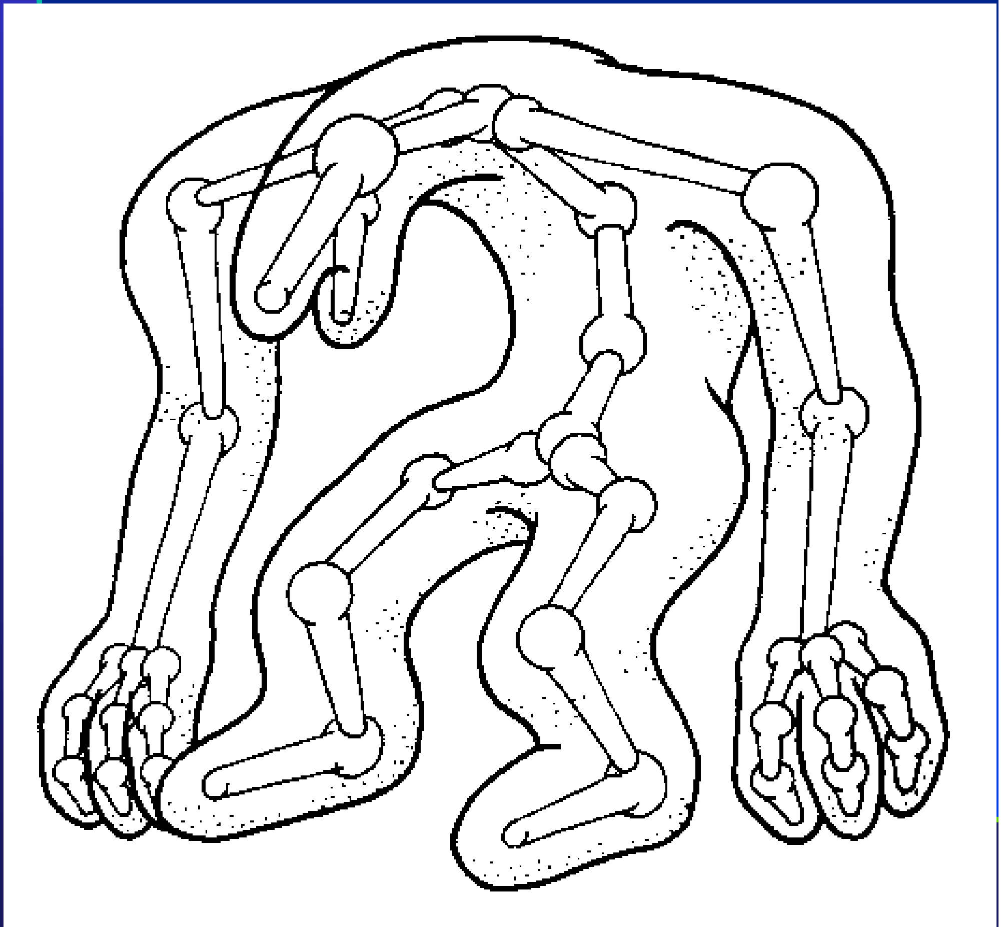


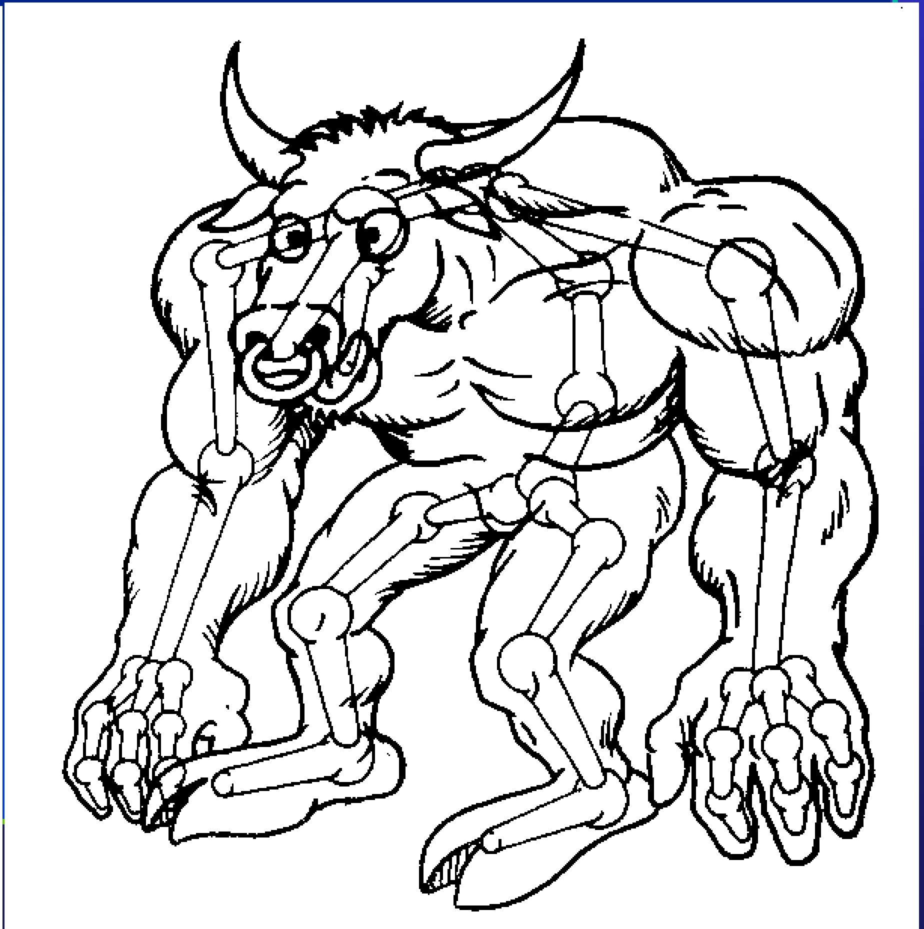
Skin and Bones

- Skeleton with joined "bones"
- Can add "skin" on top of bones
- Automatic or hand-tuned skinning



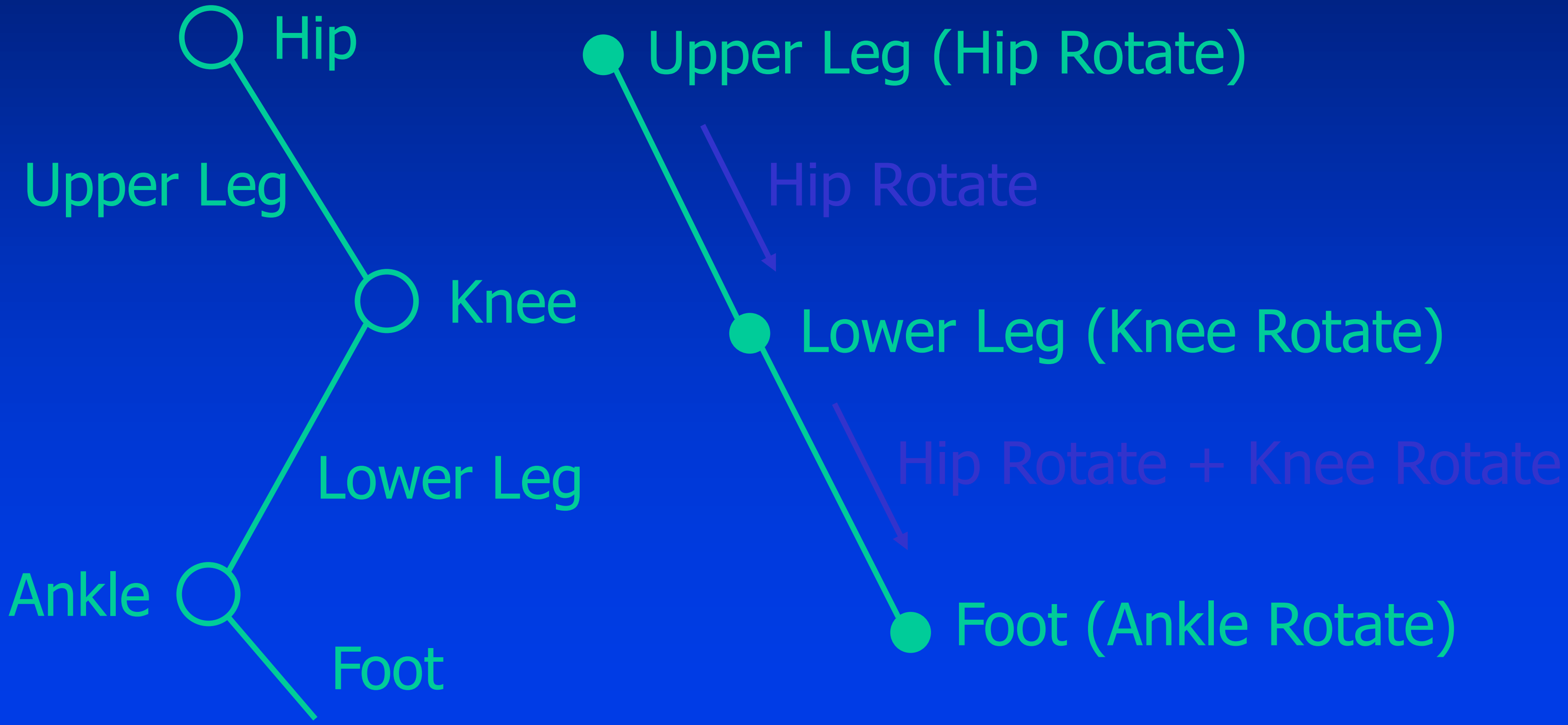
Skeletons





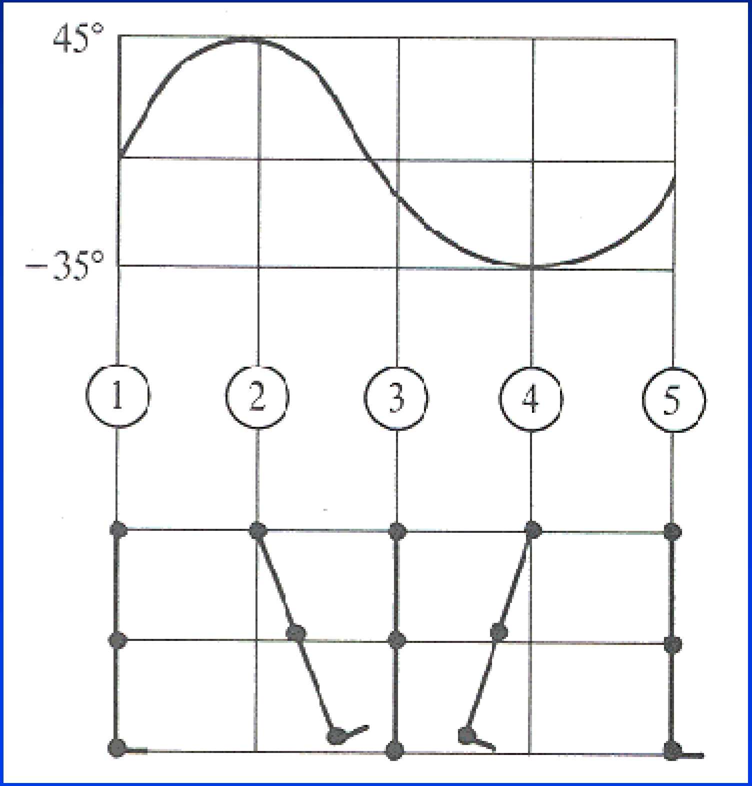
Example: Walk Cycle

- **Articulated figure:**



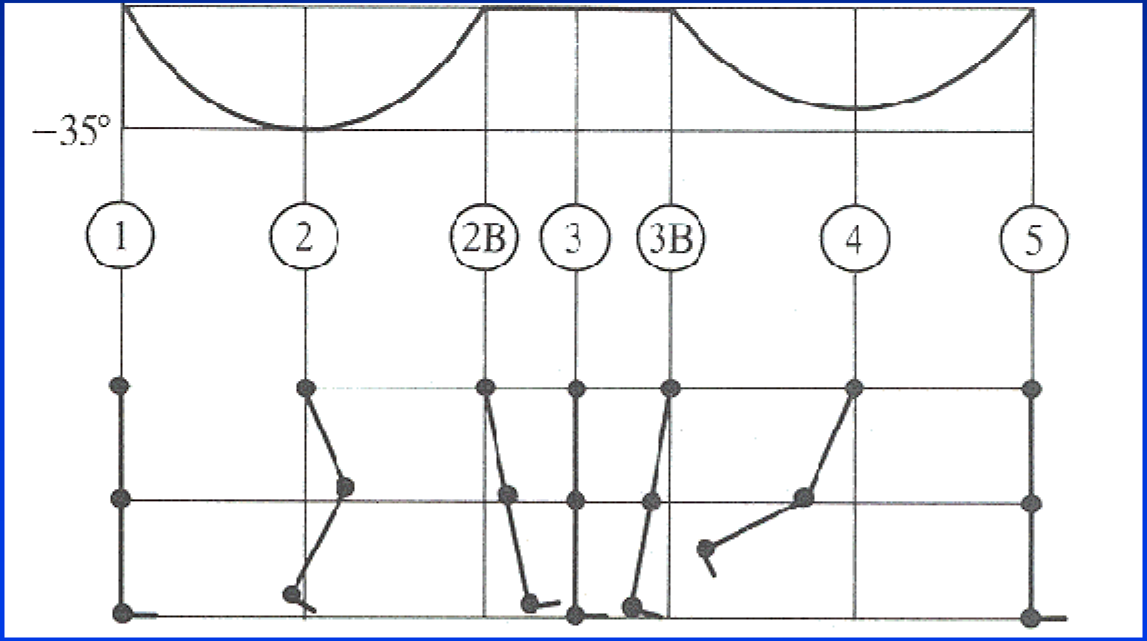
Example: Walk Cycle

- **Hip joint orientation:**



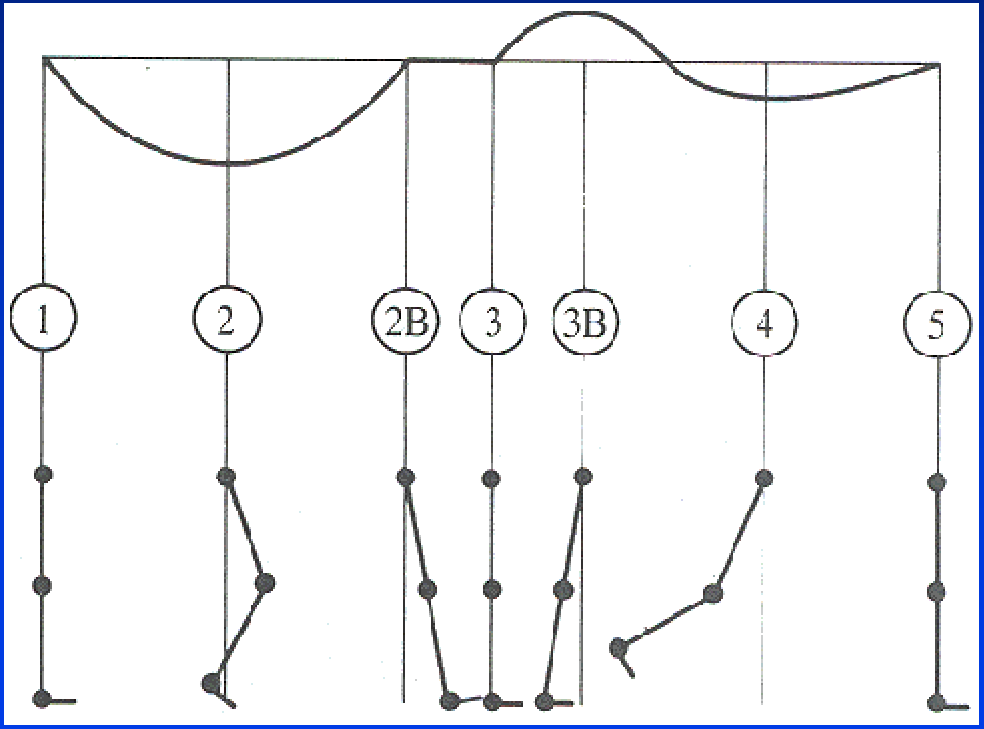
Example: Walk Cycle

- **Knee joint orientation:**



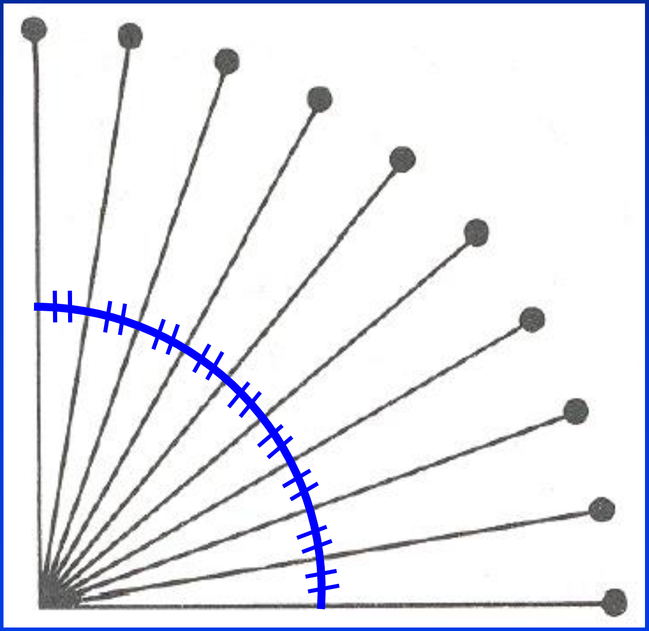
Example: Walk Cycle

- **Ankle joint orientation:**

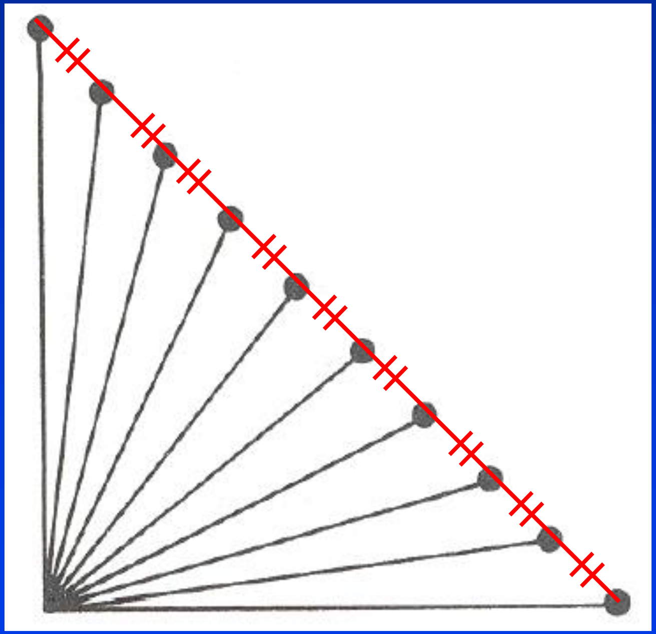


Inbetweening

- **Compute joint angles between keyframes**
  - **Consider the length constancy**



Right



Wrong

Kinematics & Dynamics

Overview

- **Kinematics**
  - Consider only motion
  - Motion is determined by positions, velocities, accelerations
- **Dynamics**
  - Consider underlying forces
  - Compute motion from initial conditions and physics

Forward and Inverse Kinematics

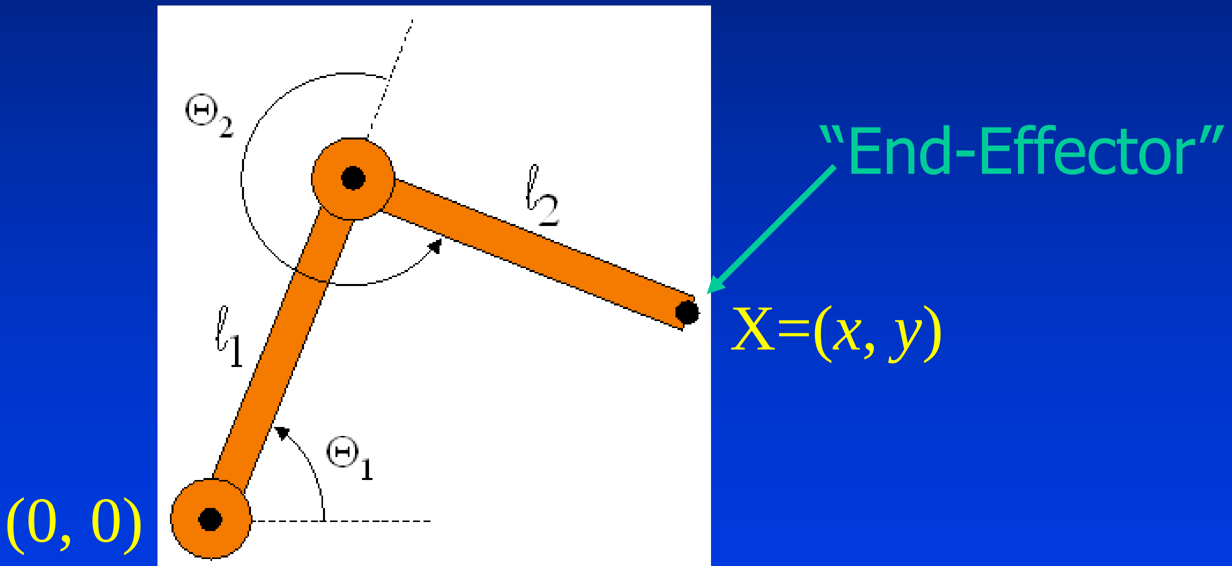
- In forward kinematics, the motion of all the joints in the structure are explicitly specified which yields the end effector position
- The end effector position X is a function of the state vector of the structure: X=f(Q)

Forward and Inverse Kinematics

- In inverse kinematics (also known as "goal-oriented motion") the end effector's position is all that is defined
- Given the end effector position, we must derive the state vector of the structure which produced that end effector position
- Thus the state vector is given by
  - $Q = f^{-1}(X)$

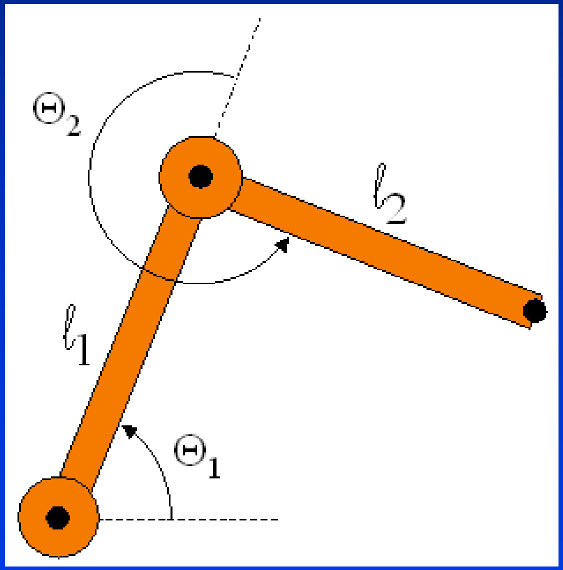
Example: 2-Link Structure

- **Two links connected by rotational joints**



Forward Kinematics

- **Animator specifies joint angles: Θ_1 and Θ_2**
- **Computer finds positions of end-effector: X**



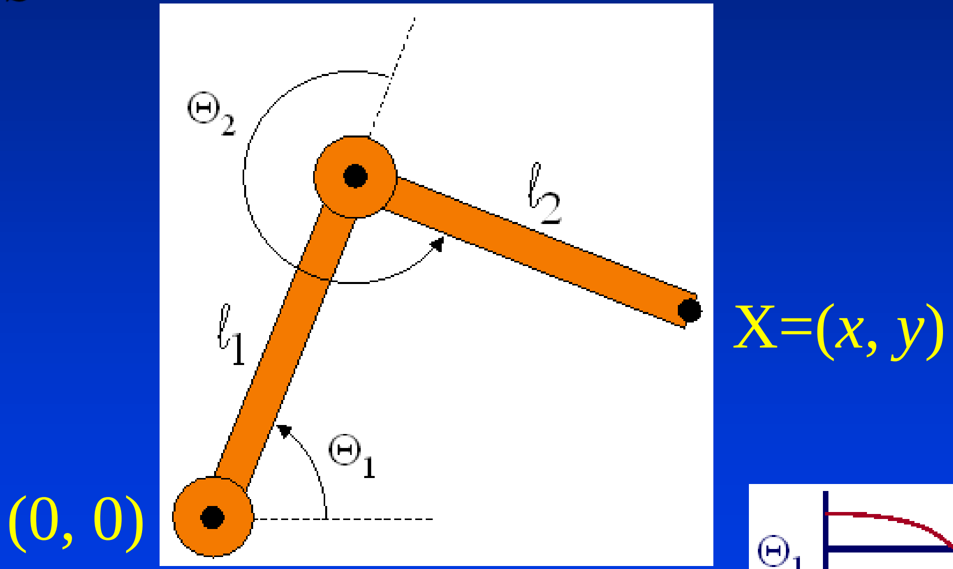
X=(x, y)

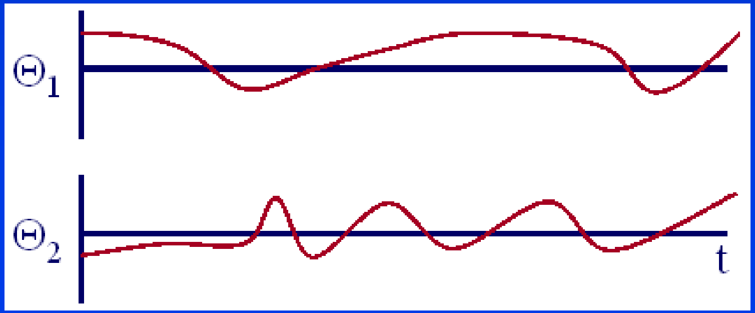
(0, 0)

$$X=(l_1\cos\Theta_1+ l_2\cos(\Theta_1+\Theta_2), l_1\sin\Theta_1+ l_2\sin(\Theta_1+\Theta_2))$$

Forward Kinematics

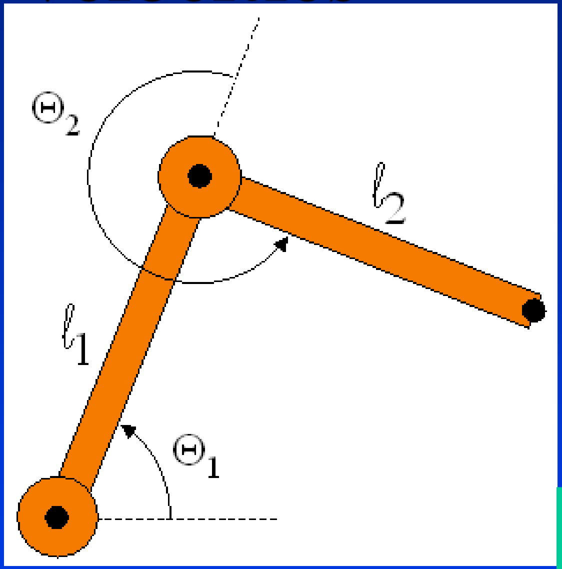
- **Joint motions can be specified by spline curves**





Forward Kinematics

- **Joint motions can be specified by initial conditions and velocities**



X=(*x*, *y*)

(0, 0)

$$\Theta_1(0) = 60° \qquad \Theta_2(0) = 250°$$

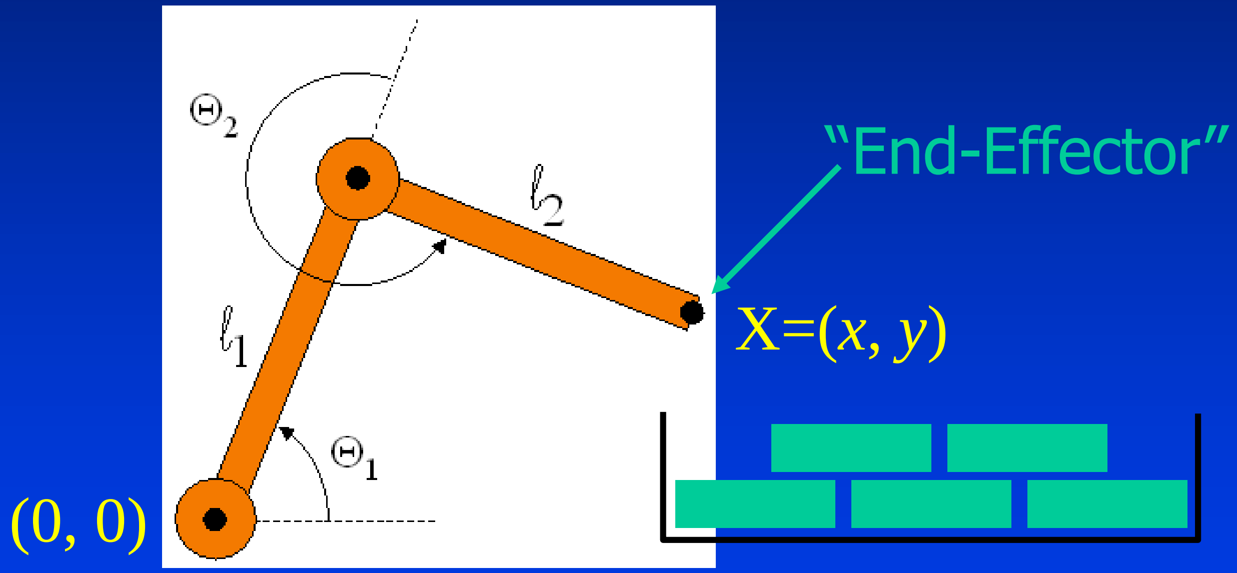
$$\frac{d\Theta_1}{dt} = 1.2 \qquad \frac{d\Theta_2}{dt} = -0.1$$

Forward Kinematics

- The figure on the next slide shows a hierarchy of two links where the links can only move in the plane of the page
- The end effector position is given as X(x,y) and the two joint angles are Θ1 and Θ2
  - Note that Θ2 is relative to the orientation of link L1
- Using geometric means (projecting each link onto the x and y axes) we can show that:
- $X = (l_1 \cos \Theta_1 + l_2 \cos (\Theta_1 + \Theta_2), \quad l_1 \sin \Theta_1 + l_2 \sin (\Theta_1 + \Theta_2))$

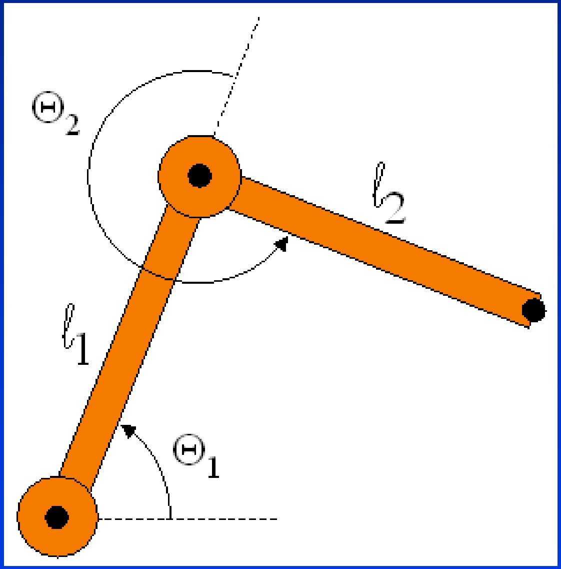
Example: 2-Link Structure

- **What if animator knows position of "end-effector"**



Inverse Kinematics

- **Animator specifies end-effector positions: X**
- **Computer finds joint angles: Θ_1 and Θ_2**



X=(x, y)

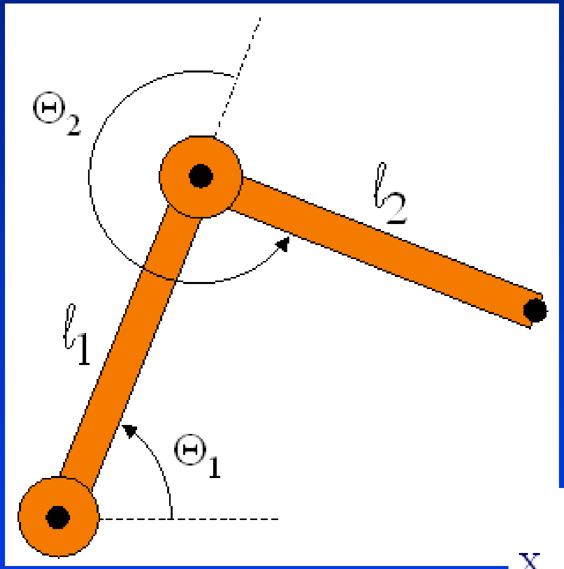
(0, 0)

$$\Theta_2 = \cos^{-1}\left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2}\right)$$

$$\Theta_1 = \frac{-(l_2 \sin\Theta_2)x + (l_1 + l_2\cos\Theta_2)y}{(l_2 \sin\Theta_2)y + (l_1 + l_2\cos\Theta_2)x}$$

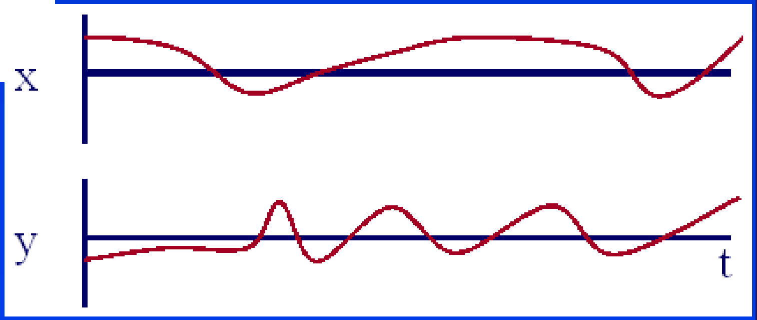
Inverse Kinematics

- **End-effector postions can be specified by spline curves**



(0, 0)

X=(*x*, *y*)

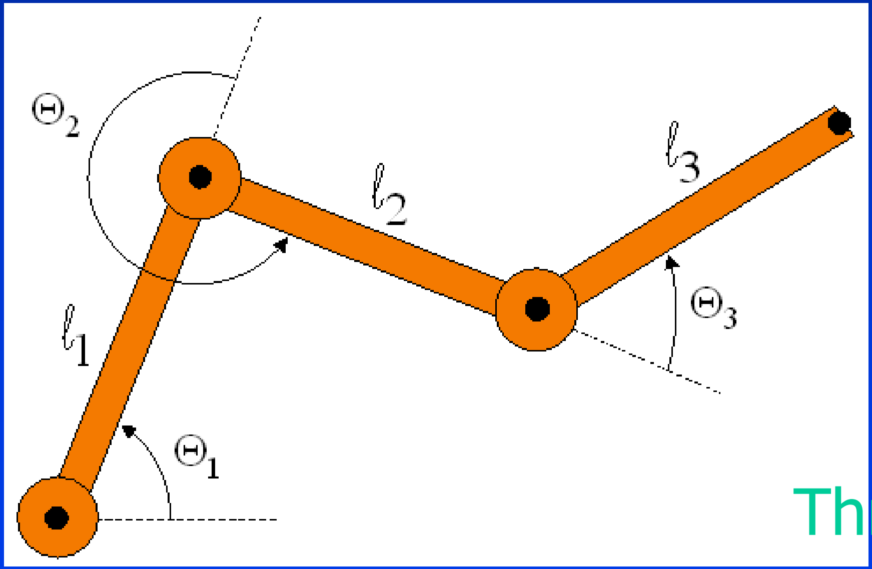


Inverse Kinematics

- Given the end-effector position (x,y) we can find the joint angles Θ1 and Θ2
  - Once again use simple geometry
- Increasing degrees of freedom (DOFs) allows more motion, but makes the geometry more difficult (for inverse kinematics, there will be multiple solutions)

Inverse Kinematics

- **Problem for more complex structures**
  - System of equations is usually under-defined
  - Multiple solutions



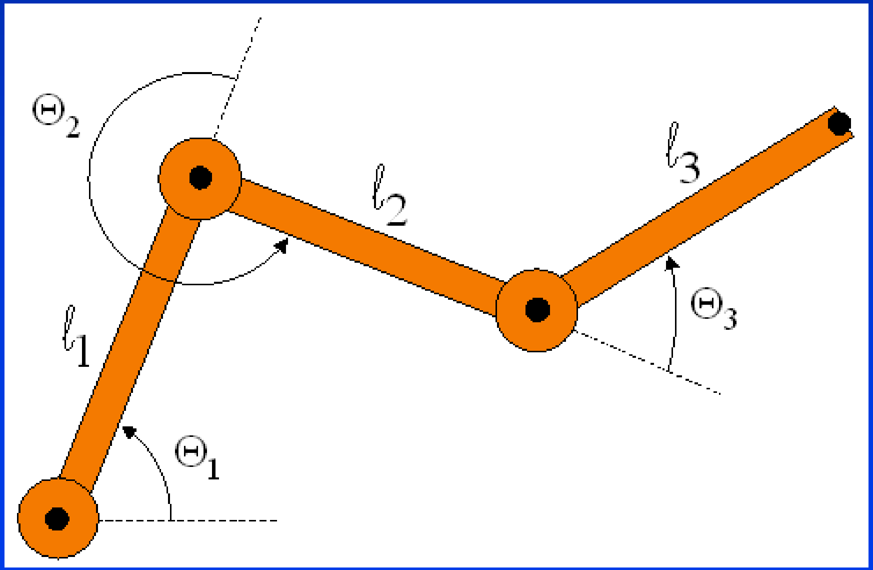
X=(x, y)

(0, 0)

Three unknowns: $\Theta_1, \Theta_2, \Theta_3$
Two equations: x, y

Inverse Kinematics

- **Solution for more complex structures**
  - Find best solution (e.g., minimize energy in motion)
  - Non-linear optimization



X=(x, y)

(0, 0)

The Jacobian Matrix

- Given X = f(Q) where X is of dimension n and Q is of dimension m, the Jacobian is the n x m matrix of partial derivatives relating differential changes of Q(dQ) to differential changes in X(dX)
  - dX = J(Q) dQ
- For use in computer graphics, we usually divide everything by dt
  - dX/dt = J(Q) dQ/dt

The Jacobian Matrix

- Where dotX is velocity of the end effector which is itself a vector of six dimensions that include linear velocity and angular velocity, and where dotQ is time derivative of the state vector
- Thus the Jacobian maps velocities in state space to velocities in cartesian space
- Thus at any time these quantities are related via the linear transformation J which itself changes through time as Q changes

The Jacobian Matrix

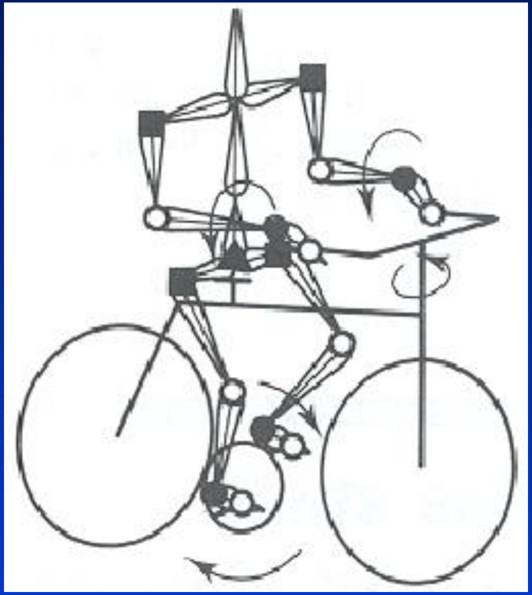
- Recall our inverse kinematics statement
- If we localize around the current operating position and invert the Jacobian we obtain
  - $dQ = J^{-1} (dX)$
- Thus, we can iterate toward the goal over a series of incremental steps as shown in the next slide

The Jacobian Matrix

- Rather than doing the actual differentiation we need another way to construct the Jacobian
- This is done by developing a system for referencing the chain of links (not developed here)

Summary

- **Forward kinematics**
  - Specify conditions (joint angles)
  - Compute positions of end-effectors
- **Inverse kinematics**
  - "Goal-directed" motion
  - Specify goal positions of end effectors
  - Compute conditions required to achieve goals



Inverse kinematics provides easier specification for many animation tasks, but it is computationally more difficult

Kinematics vs. Dynamics

- **Kinematics**
  - Consider only motion
  - Motion is determined by positions, velocities, accelerations
- **Dynamics**
  - Consider underlying forces
  - Compute motion from initial conditions and physics

Challenge of Animation

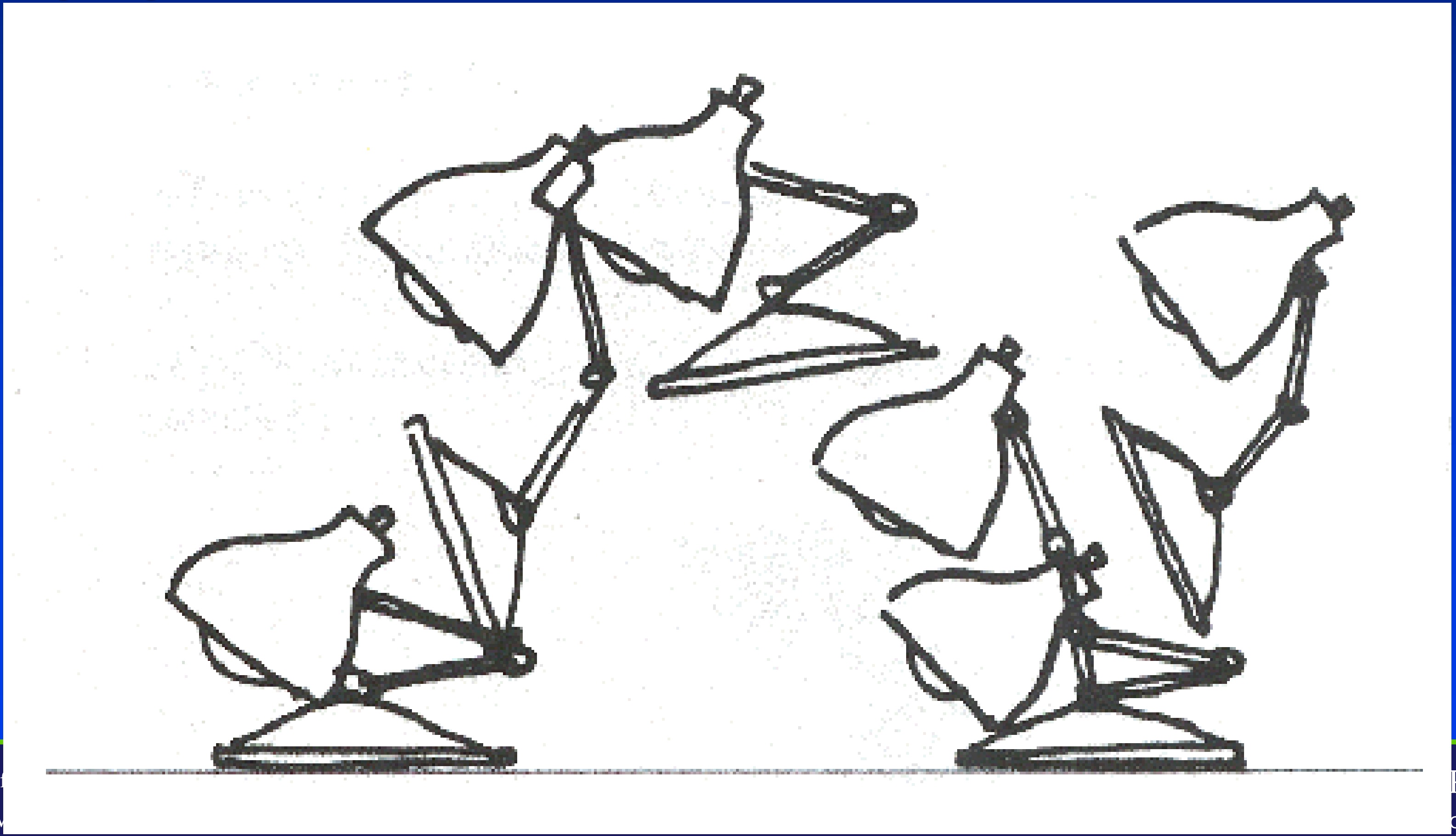
- **Temporal aliasing**
  - Motion blur

Summary

- **Animation Requires ...**
  - Modeling
  - Scripting
  - Inbetweening
  - Lighting, shading
  - Rendering
  - Image processing

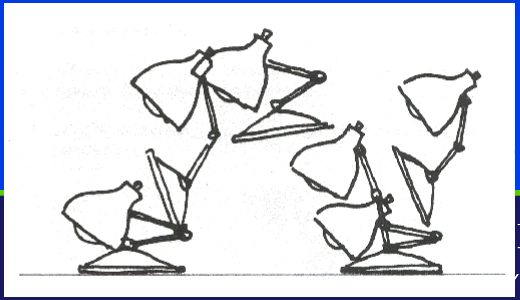
Dynamics

- **Simulation of physics ensures realism of motion**



Space Time Constraints

- **Animator specifies constraints**
  - What the character's physical structure is
    - e.g., articulated figure
  - What the character has to do
    - e.g., jump from here to there within time *t*
  - What other physical structures are present
    - e.g., floor to push off and land
  - How the motion should be performed
    - e.g., minimize energy



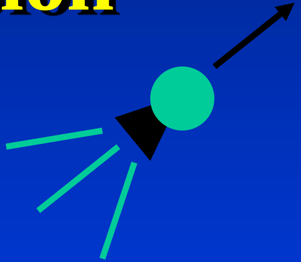
Space Time Constraints

- **Computer Finds the "Best" Physical Motion**
  - Satisfying constraints
- **Example: Particle with Jet Propulsion**
  - *x*(*t*) is position of particle at time *t*
  - ***f***(*t*) is force of jet propulsion at time *t*
  - Particle's equation of motion is:

$$m x'' - f - m g = 0$$

  - Suppose we want to move from *a* to *b* within t_0 to t_1 with minimum jet fuel:

$$\text{Minimize } \int_{t_0}^{t_1} |f(t)|^2 dt \text{ subject to } x(t_0) = a \text{ and } x(t_1) = b$$



Space Time Constraints

- **Discretize time steps**

$$x' = \frac{x_i - x_{i-1}}{h}$$

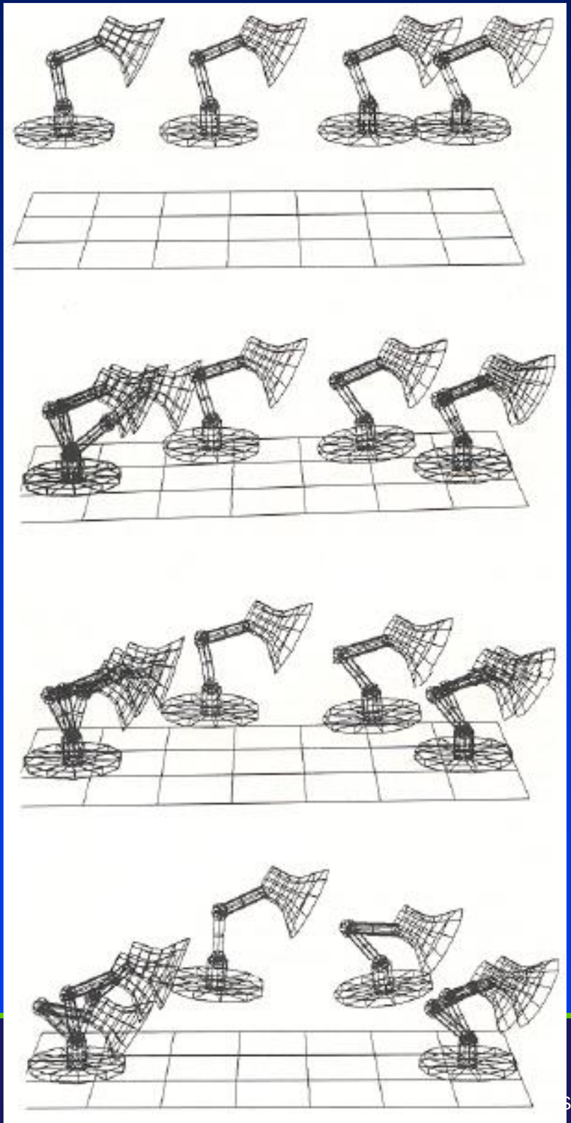
$$x'' = \frac{x_{i+1} - 2x_i + x_{i-1}}{h^2}$$

$$m\left(x'' = \frac{x_{i+1} - 2x_i + x_{i-1}}{h^2}\right) - f_i - mg = 0$$

$$\text{Minimize} \quad h\sum_i |f_i|^2 \text{ subject to } x_0 = a \text{ and } x_1 = b$$

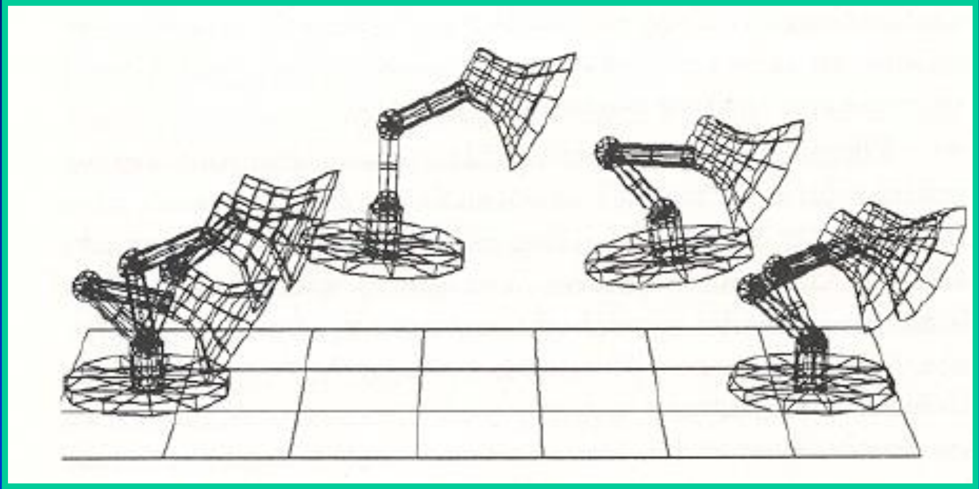
Space Time Constraints

- **Solve with iterative optimization methods**

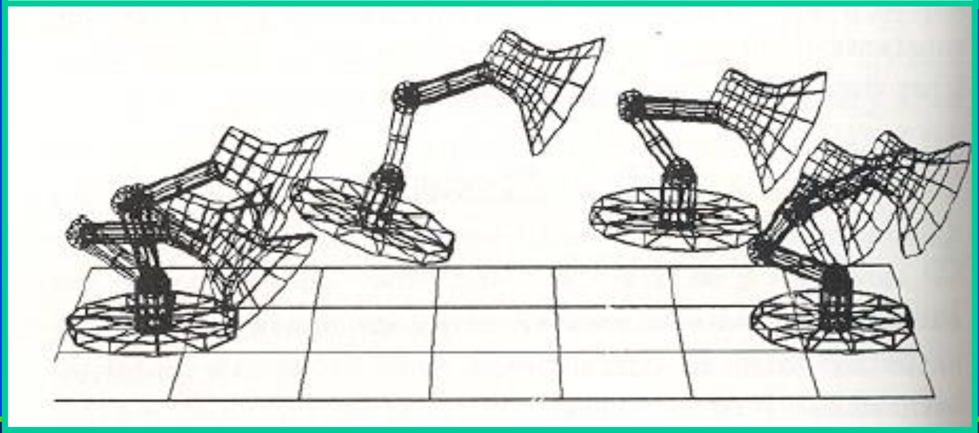


Space Time Constraints

- **Adapting motion**



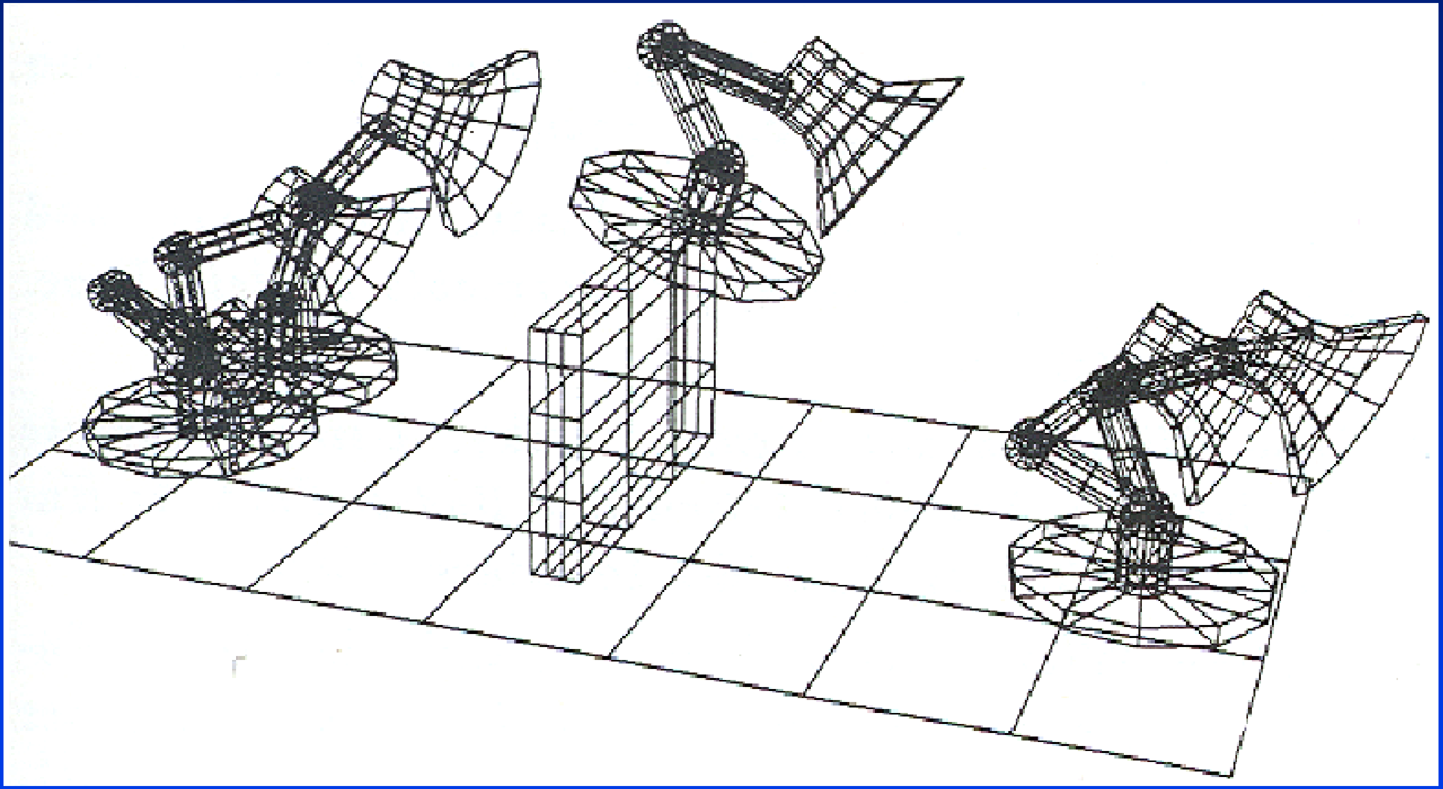
Original Jump



Heavier Base

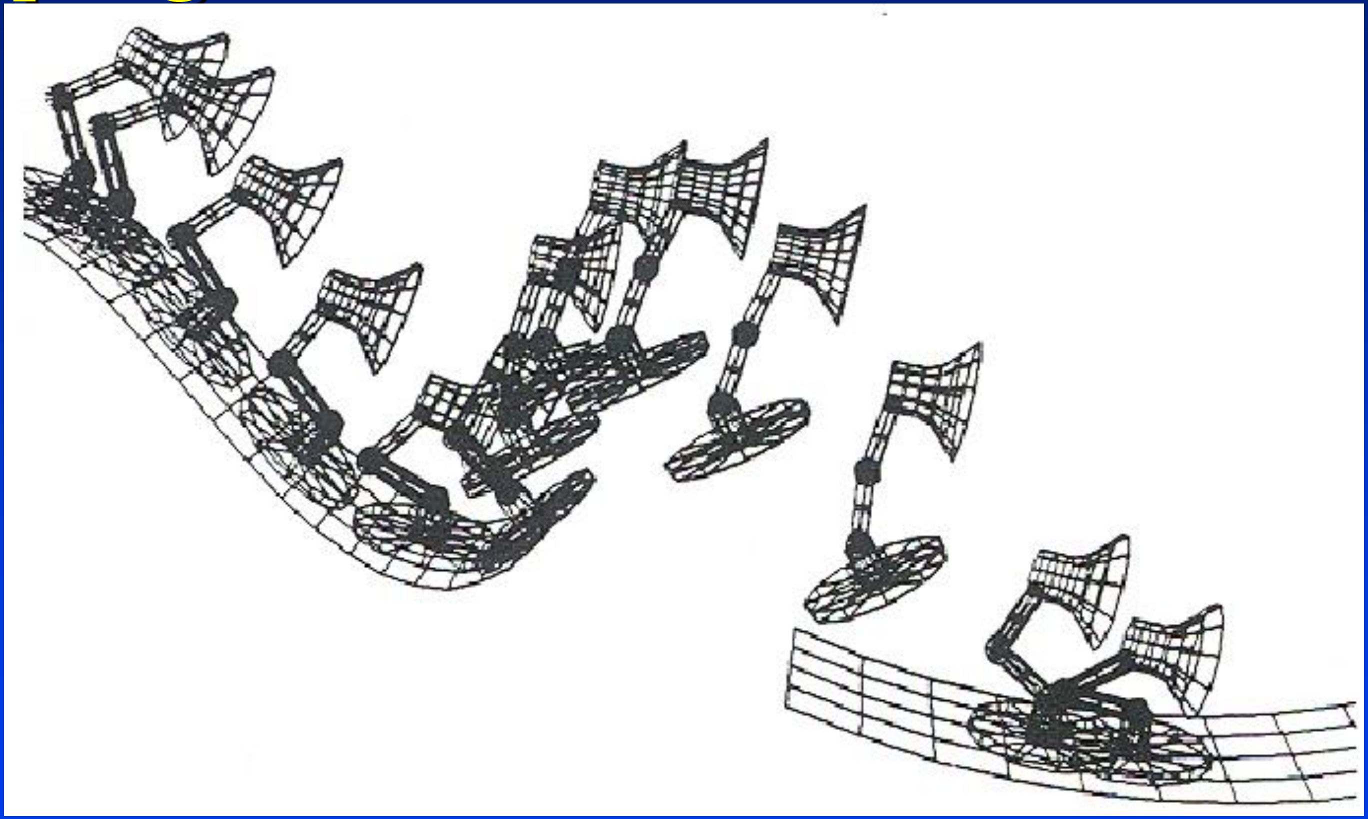
Space Time Constraints

- **Adapting motion**


Hurdle

Space Time Constraints

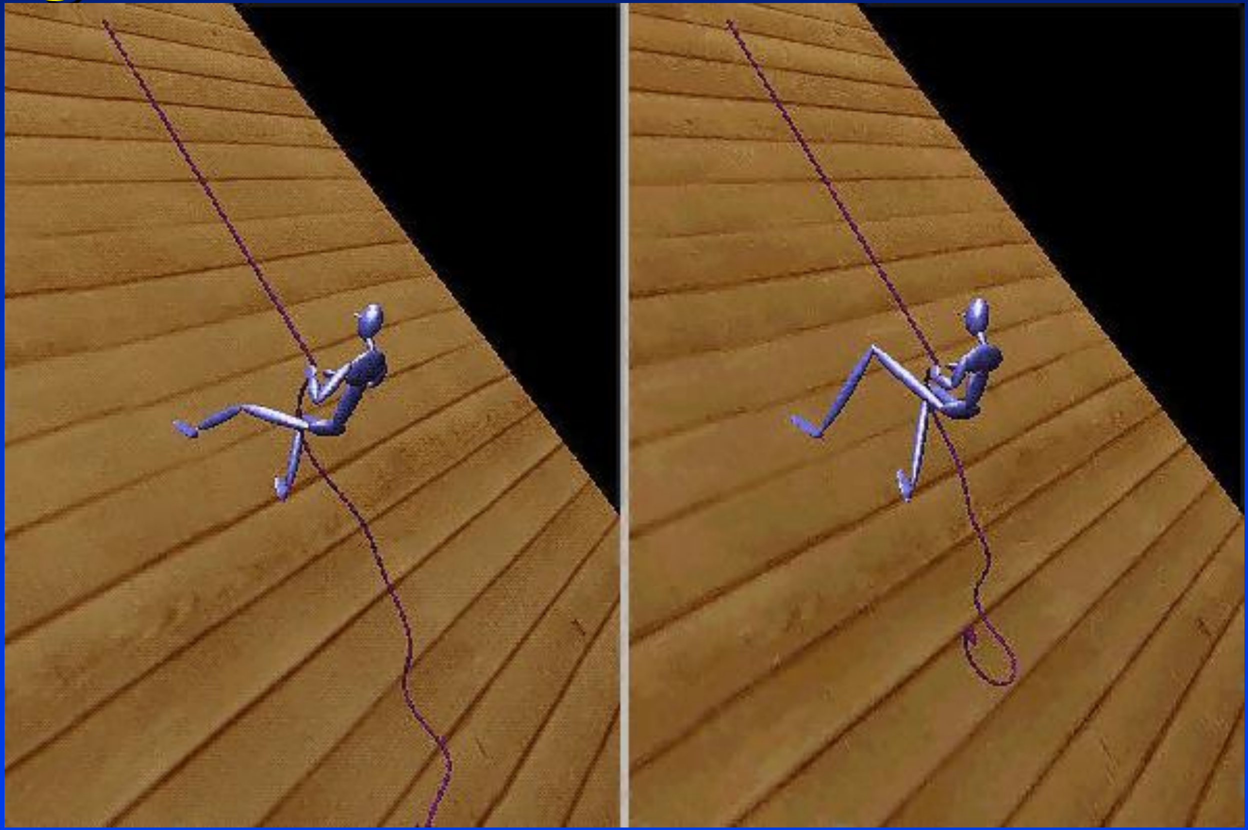
- **Adapting motion**



Ski Jump

Space Time Constraints

- **Editing motion**



Original Adapted

Space Time Constraints

- **Motion morphing**

The female character morphs into a smaller character during her spine

Space Time Constraints

- **Advantages**
  - Free animator from having to specify details of physically realistic motion with spline curves
  - Easy to vary motions due to new parameters and/or new constraints

- **Challenges**
  - Specifying constraints and objective functions
  - Avoiding local minima during optimization

Soft Object Animation

- Free Form Deformation (FFD) is part of the computer graphics literature on **soft objects**
  - The definition of a soft object is an object that can be deformed by the user or during the process of animation
- Soft object deformation is used for many purposes:
  - Shape distortion to highlight dynamic interaction with the environment
    - For instance, an animator may want to create a basketball that will deform when it bounces on the ground

Soft Object Animation

- Another use would be to deform the shape of a car during a collision in a racing simulation
- Realistic deformation of an object that has a highly elastic and flexible shape
  - Examples include the facial expressions, motion of the human body, and cartoon animation
  - In movies like **Luxo Jr.** and **Toy Story**, the character shapes are deformed when they walk, talk, or hit another object
- Deformation of an object occurs by moving the vertices of a polygonal object or the control points of a parametric curve

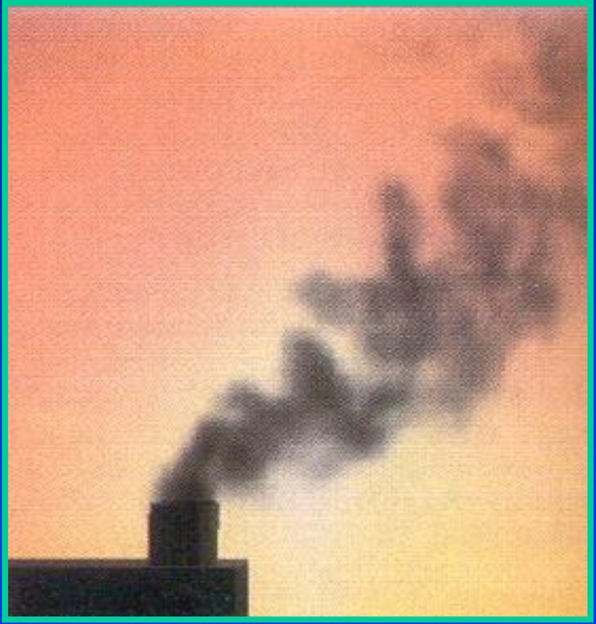
Soft Object Animation

- Deformation of polygonal objects in problematic since it can cause aliasing effects to occur
- More typical representations for deformation are
  – Bezier patch representation
  – B-spline patch representation

Dynamics

- **Other physical simulations**
  - Rigid bodies
  - Soft bodies
  - Cloth
  - Liquids
  - Gases
  - etc.

Cloth


Hot Gases

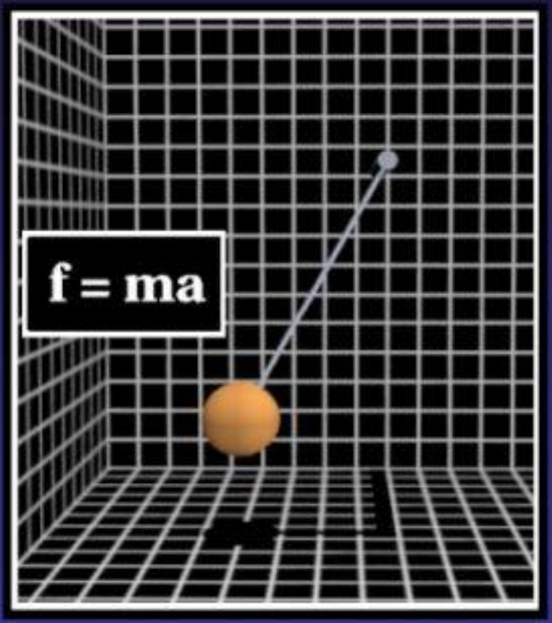
Particles

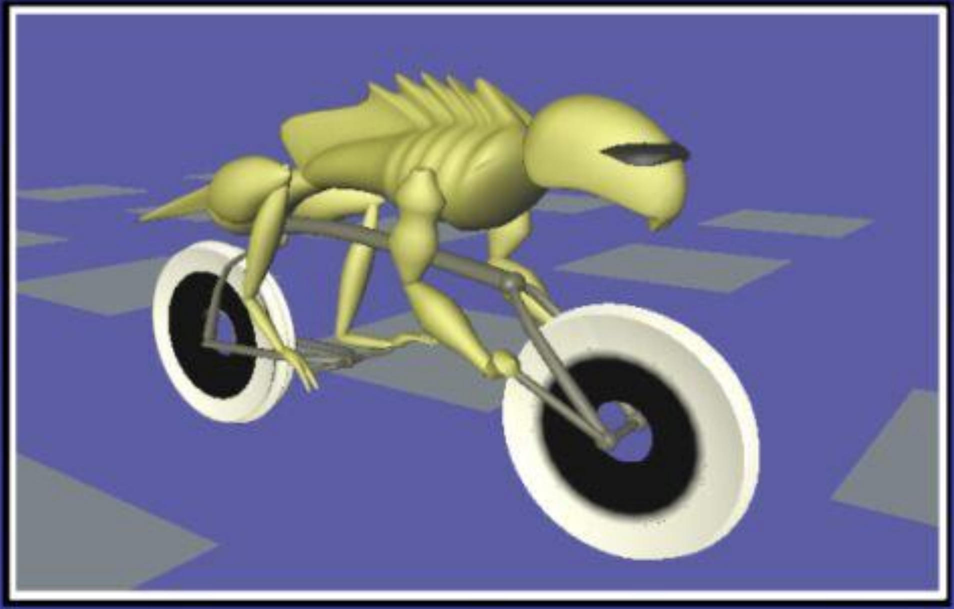
Kinematics vs. Dynamics

- **Kinematics**
  - Forward kinematics
    - Animator specifies joints (hard)
    - Compute end-effectors (easy)
  - Inverse kinematics
    - Animator specifies end-effectors (easier)
    - Solve for joints (harder)
- **Dynamics**
  - Space-time constraints
    - Animator specifies structures & constraints (easiest)
    - Solve for motion (hardest)
  - Also other physical simulations

Animation Techniques

Procedural Techniques





Procedural Animation

- Very general term for a technique that puts more complex algorithms behind the scenes
- Technique attempts to consolidate artistic efforts in algorithms and heuristics
- Allows for optimization and physical simulation

Procedural Animation Strength

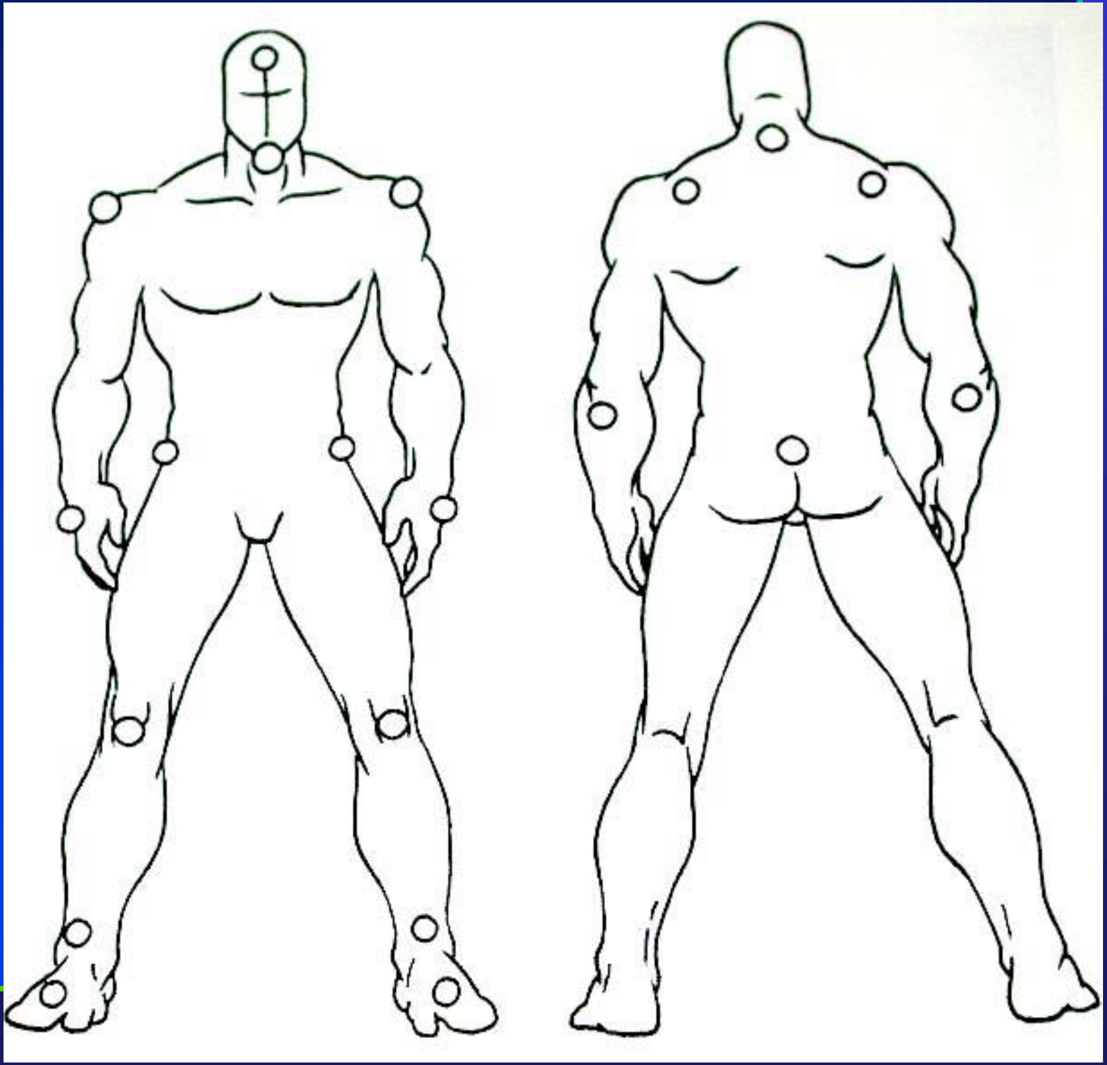
- Animation can be generated ‘on the fly’
- Dynamic response to user
- Write-once, use-often
- Algorithms provide accuracy and exhaustive search that animators cannot

Procedural Animation Weakness

- We are not great at boiling human skill down to algorithms
  – How do we move when juggling?
- Difficult to generate
- Expensive to compute
- Difficult to force system to generate a particular solution
  – Bicycles will fall down

Motion Capture

- More realistic motion sequences can be generated by *Motion Capture*
- Attach joint position indicators to real actors
- Record live action



Animation Techniques

Motion Capture

Microsoft Motion Capture Group

Motion Analysis

Motion Capture Strength

- Exactly captures the motions of the actor
  - Michael Jordan's video game character will capture his style
- Easy to capture data

Motion Capture Weakness

- Noise, noise, noise!
- Magnetic system interference
- Visual system occlusions
- Mechanical system mass
- Tethered (wireless is available now)

Motion Capture Weakness

- Aligning motion data with CG character
  - Limb lengths
  - Idealized perfect joints
- Reusing motion data
  - Difficult to scale in size (must also scale in time)
  - Changing one part of motion

Motion Capture Weakness

- Blending segments
  - Motion clips are short (due to range and tethers)
  - Dynamic motion generation requires blending at run time
  - Difficult to manage smooth transition