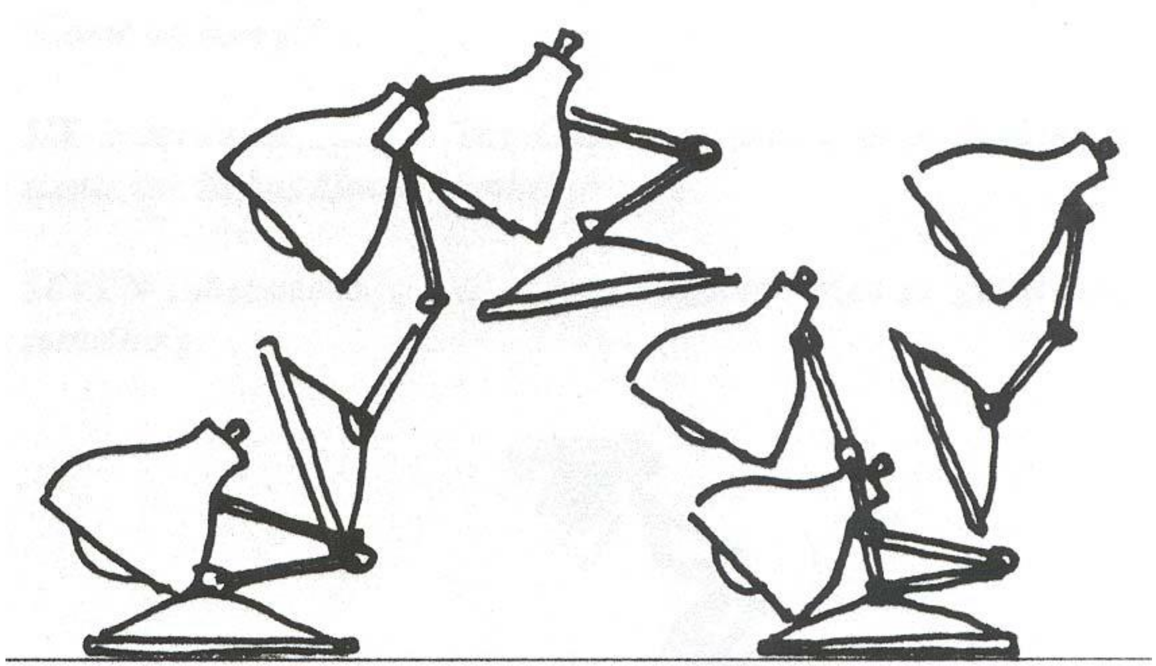
電腦動畫
Computer Animation

Shih-Ching (Luke) Yeh
shihchiy@csie.ncu.edu.tw

Department of Computer Science and Information Engineering

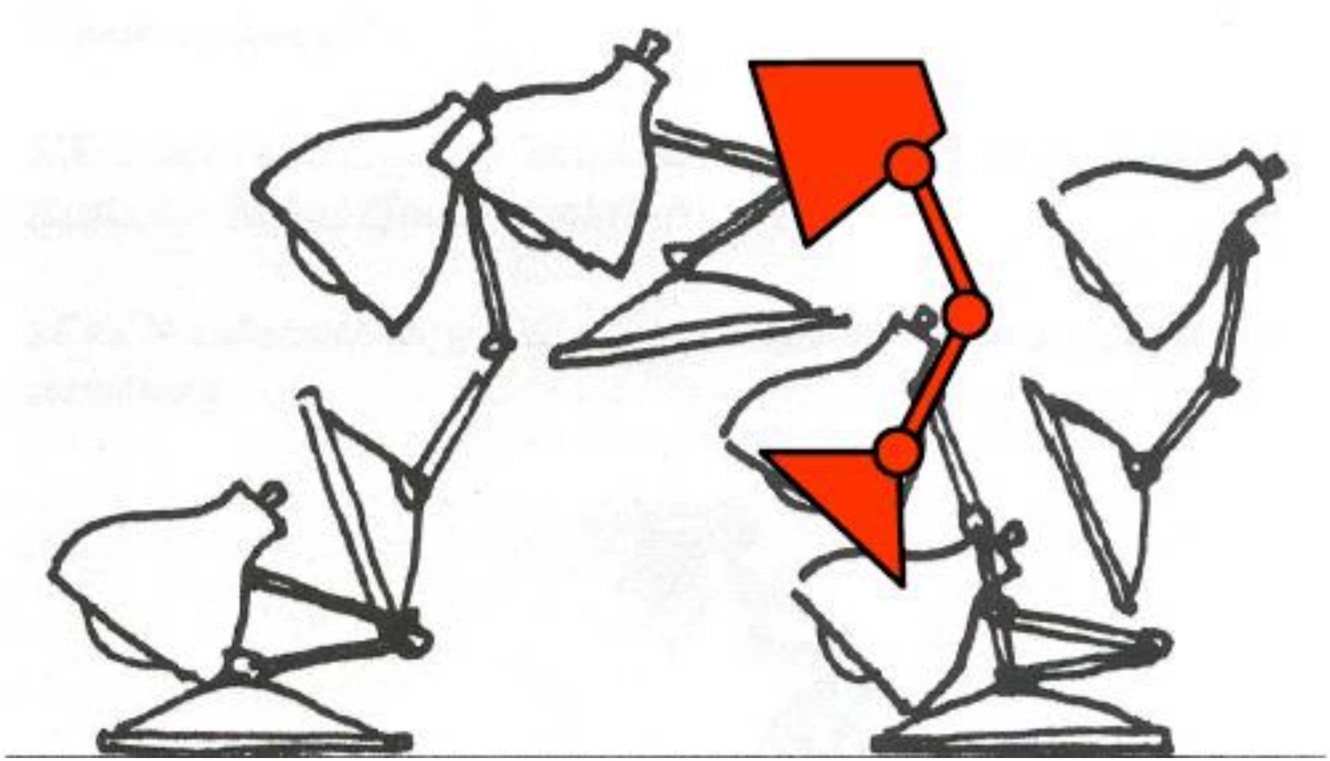
Keyframe Animation

- Define character poses at specific time steps called “keyframes”



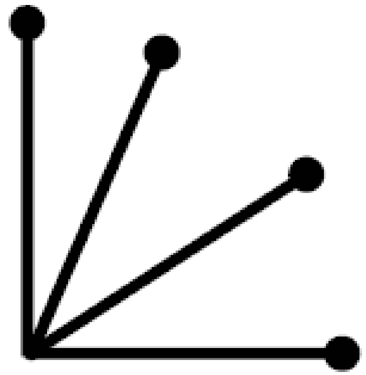
Keyframe Animation

- Interpolate variables describing keyframes to determine poses for character "in-between"

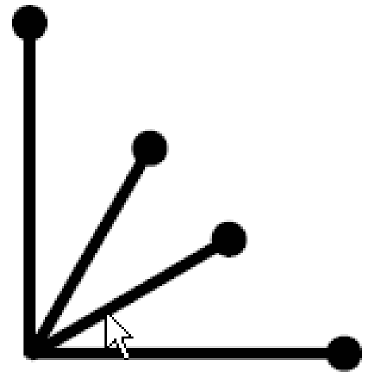


Keyframe Animation

- Specify only the important frames, interpolate the frames in-between



interpolating angle

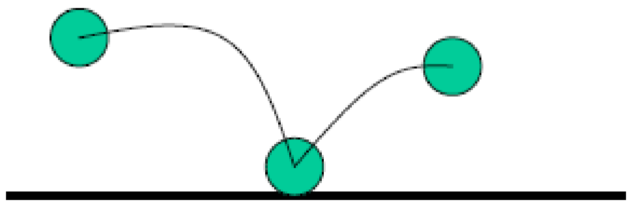


interpolating endpoints

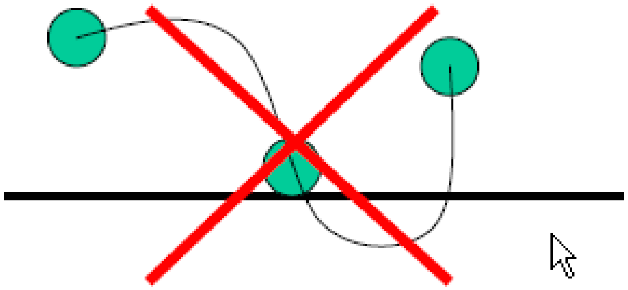
What and how to interpolate is important

Keyframe Animation

How to interpolate between keyframes?



intended result



unintended result

We need a smooth interpolation plus user control

Keyframe Animation

- How are we going to interpolate?

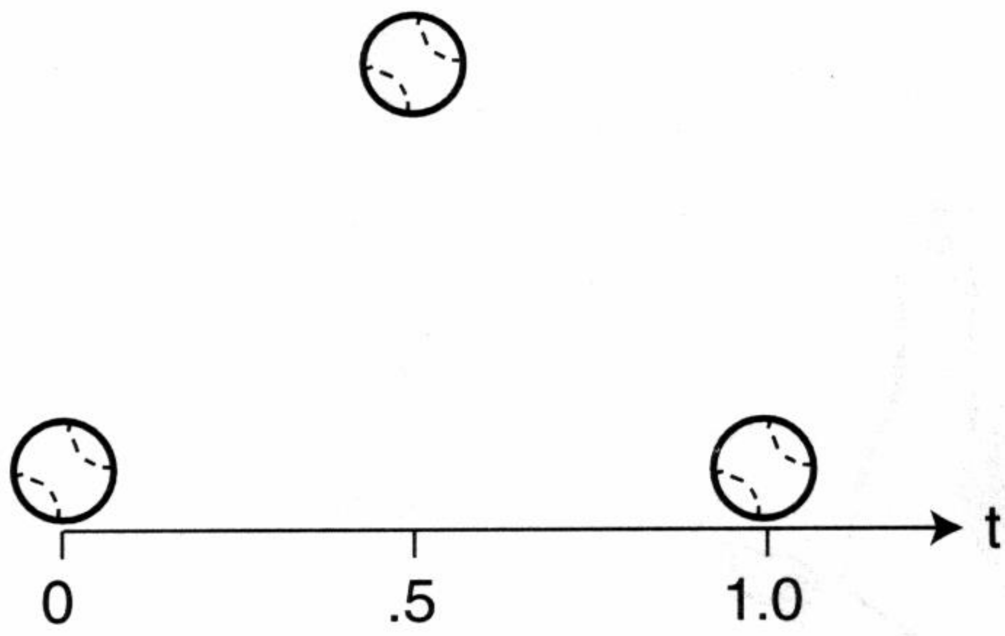
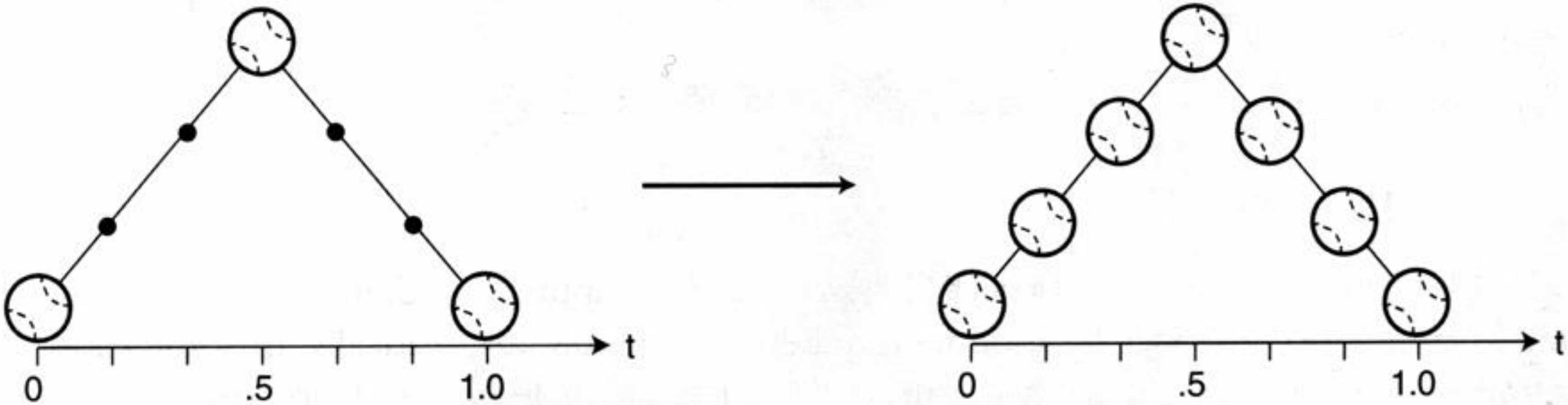


Figure 10.4 Three keyframes. Three keyframes representing a ball on the ground, at its highest point, and back on the ground.

Keyframe Animation

- Linear Interpolation
- Simple, but discontinuous velocity

Figure 10.5 Inbetweening with linear interpolation. Linear interpolation creates inbetween frames at equal intervals along straight lines. The ball moves at a constant speed. Ticks indicate the locations of inbetween frames at regular time intervals (determined by the number of frames per second chosen by the user).



Keyframe Animation

- Nonlinear Interpolation
- Smooth ball trajectory and continuous

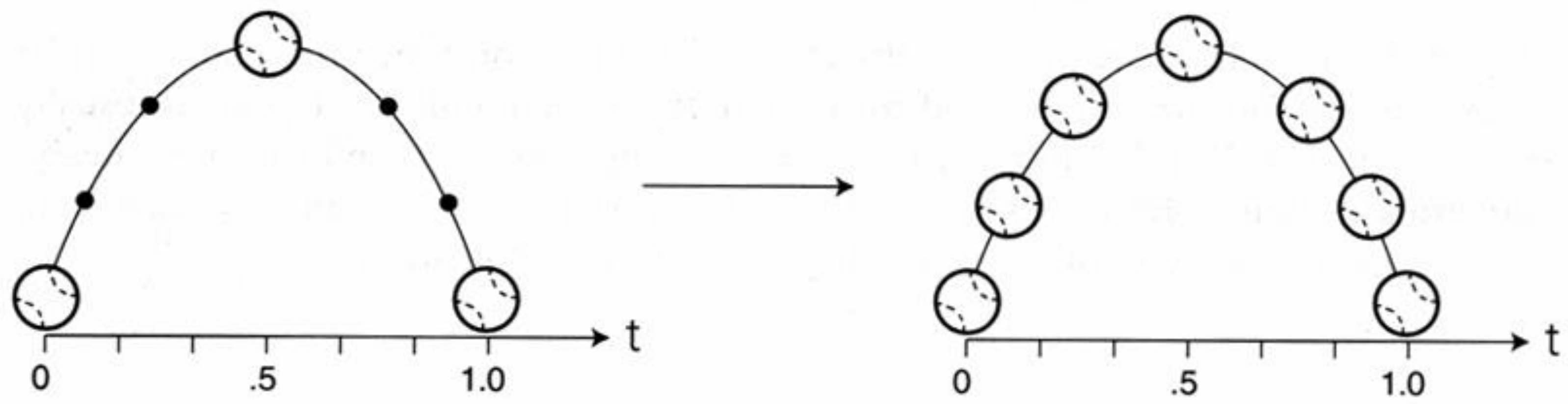


Figure 10.9 **Inbetweening with nonlinear interpolation.** Nonlinear interpolation can create equally spaced inbetween frames along curved paths. The ball still moves at a constant speed. (Note that the three keyframes used here and in Fig. 10.10 are the same as in Fig. 10.4.)

Keyframe Animation

- Easing
- Adjust the timing of the inbetween frames. Can be automated by adjusting the stepsize of parameter, t.

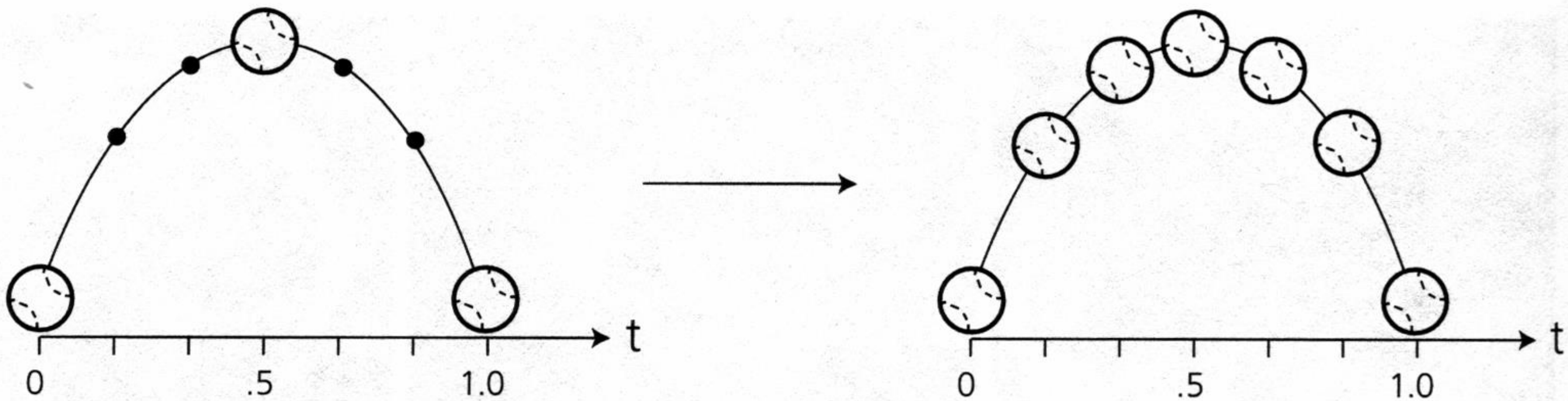


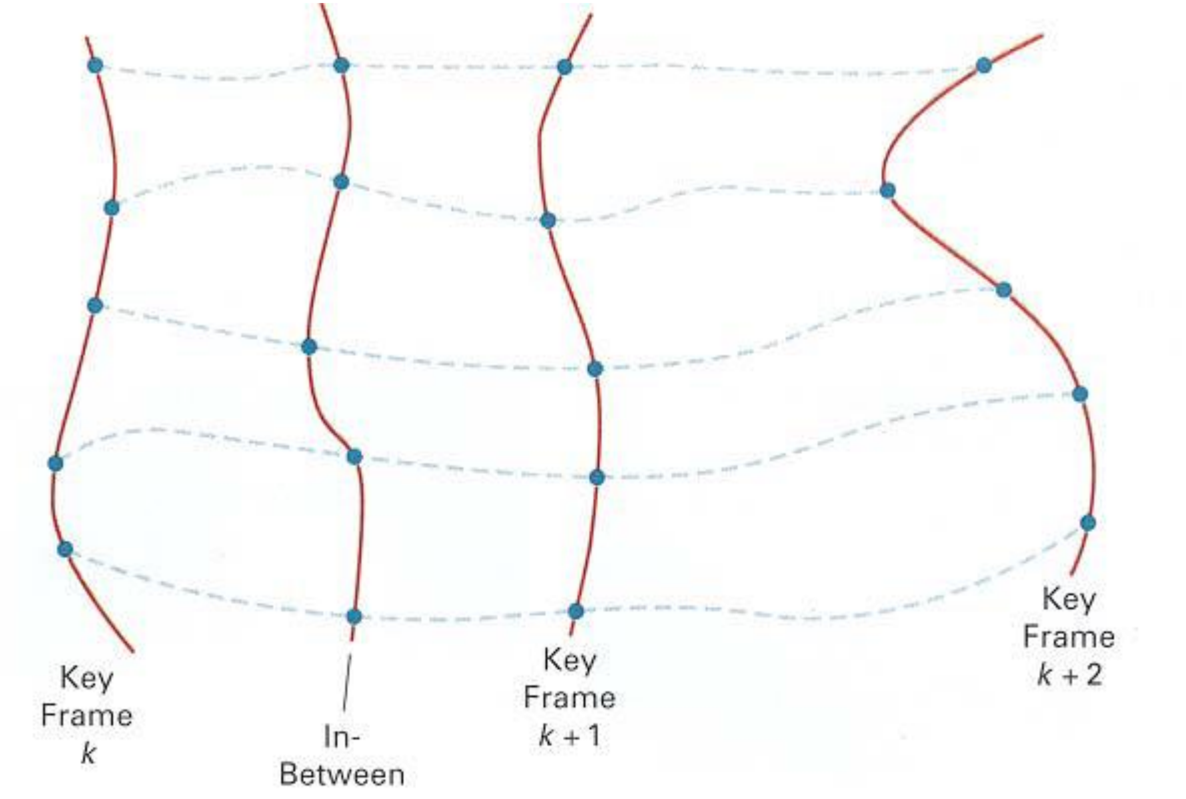
Figure 10.10 **Inbetweening with nonlinear interpolation and easing.** The ball changes speed as it approaches and leaves keyframes, so the dots indicating calculations made at equal time intervals are no longer equidistant along the path.

Keyframe Animation

- Interpolation
  - Many parameters can be interpolated to generate animation
  - Simple interpolation techniques can only generate simple inbetweens
  - More complicated inbetweening will require a more complicated model of animated object and simulation

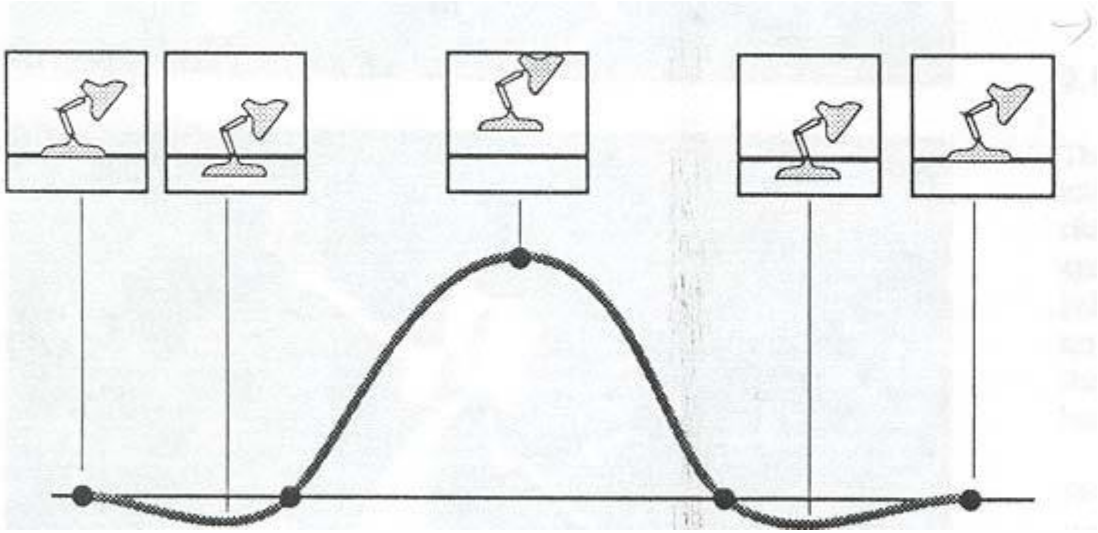
Keyframe Animation

- Inbetweening:
  - Spline interpolation - maybe good enough



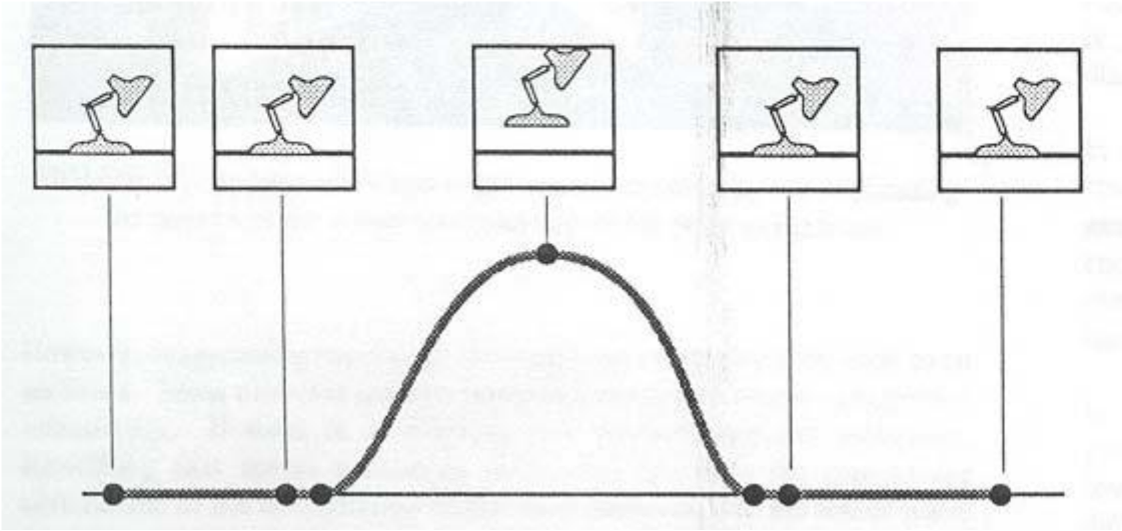
Keyframe Animation

- Inbetweening:
  - Cubic spline interpolation - maybe good enough
  - May not follow physical laws



Keyframe Animation

- Inbetweening:
  - Cubic spline interpolation - maybe good enough
  - May not follow physical laws



Keyframe Animation

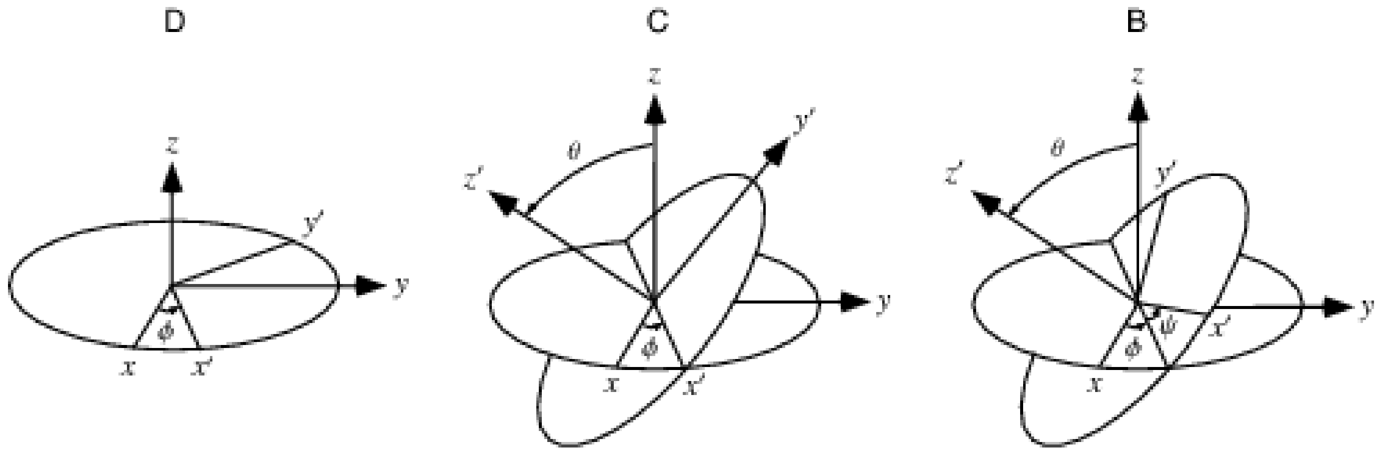
- Interpolating Rotations
  - To interpolate rotations, matrices are a bad idea.
  - Reason: redundancies in matrix coefficients
    - Non-uniqueness of an interpolation
    - Numerical issues ($\alpha R1 + (1-\alpha)R2$ is not a rotation)

We can use quaternions!!

Keyframe Animation

- Orientation
  - We will define 'orientation' to mean an object's instantaneous rotational configuration
- There are several popular options though:
  - Euler angles
  - Rotation vectors (axis/angle)
  - Quaternions
  - and more…

Euler angles



M = B C D

Euler Angles to Rotation Matrix

- Basic 3D Transformations

Rotate around Z axis:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 & 0 \\ \sin\Theta & \cos\Theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Rotate around Y axis:

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ \boldsymbol{z}' \\ \boldsymbol{w} \end{bmatrix} = \begin{bmatrix} \cos\Theta & 0 & \sin\Theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\Theta & 0 & \cos\Theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \\ \boldsymbol{w} \end{bmatrix}$$

Rotate around X axis:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\Theta & -\sin\Theta & 0 \\ 0 & \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Keyframe Animation

- Euler Angles
  - Euler angles are used in a lot of applications, but they tend to require some rather arbitrary decisions
  - They also do not interpolate in a consistent way (but this isn't always bad)
  - They can suffer from Gimbal lock and related problems
  - There is no simple way to concatenate rotations
  - Conversion to/from a matrix requires several trigonometry operations
  - They are compact (requiring only 3 numbers)

Rotation about an arbitrary line

$$\begin{bmatrix} u^2+(v^2+w^2)\cos\theta & uv(1-\cos\theta)-w\sin\theta & uw(1-\cos\theta)+v\sin\theta & (a(v^2+w^2)-u(bv+cw))(1-\cos\theta)+(bw-cv)\sin\theta \\ uv(1-\cos\theta)+w\sin\theta & v^2+(u^2+w^2)\cos\theta & vw(1-\cos\theta)-u\sin\theta & (b(u^2+w^2)-v(au+cw))(1-\cos\theta)+(cu-aw)\sin\theta \\ uw(1-\cos\theta)-v\sin\theta & vw(1-\cos\theta)+u\sin\theta & w^2+(u^2+v^2)\cos\theta & (c(u^2+v^2)-w(au+bv))(1-\cos\theta)+(bu-av)\sin\theta \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Keyframe Animation

- Quaternions
- Quaternions are an interesting mathematical concept with a deep relationship with the foundations of algebra and number theory
- Invented by W.R.Hamilton in 1843
- In practice, they are most useful to us as a means of representing orientations
- A quaternion has 4 components

$$\mathbf{q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}$$

Keyframe Animation

- Quaternions are actually an extension to complex numbers
- Of the 4 components, one is a ‘real’ scalar number, and the other 3 form a vector in imaginary ijk space!

$$\mathbf{q} = q_0 + iq_1 + jq_2 + kq_3$$

$$i^2 = j^2 = k^2 = ijk = -1$$

$$i = jk = -kj$$

$$j = ki = -ik$$

$$k = ij = -ji$$

Keyframe Animation

- Sometimes, they are written as the combination of a scalar value s and a vector value **v**

$$\mathbf{q} = \langle s, \mathbf{v} \rangle$$

- where

$$s = q_0$$
$$\mathbf{v} = \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}$$

Keyframe Animation

- A quaternion can represent a rotation by an angle θ around a unit axis **a**:

$$\mathbf{q} = \begin{bmatrix} \cos\frac{\theta}{2} & a_x \sin\frac{\theta}{2} & a_y \sin\frac{\theta}{2} & a_z \sin\frac{\theta}{2} \end{bmatrix}$$

or

$$\mathbf{q} = \left\langle \cos\frac{\theta}{2}, \mathbf{a}\sin\frac{\theta}{2} \right\rangle$$

Keyframe Animation

- Quaternion to Matrix
- To convert a quaternion to a rotation matrix:

$$\begin{bmatrix} 1-2q_2^2-2q_3^2 & 2q_1q_2+2q_0q_3 & 2q_1q_3-2q_0q_2 \\ 2q_1q_2-2q_0q_3 & 1-2q_1^2-2q_3^2 & 2q_2q_3+2q_0q_1 \\ 2q_1q_3+2q_0q_2 & 2q_2q_3-2q_0q_1 & 1-2q_1^2-2q_2^2 \end{bmatrix}$$

Matrix to Quaternion

- $qw = \sqrt{(1 + m00 + m11 + m22)}\ /2$
  $qx = (m21 - m12)/(4 * qw)$
  $qy = (m02 - m20)/(4 * qw)$
  $qz = (m10 - m01)/(4 * qw)$

Matrix to Euler Angles

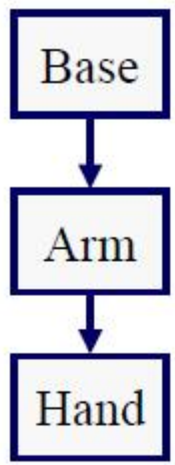
heading = atan2(-m20,m00)
attitude = asin(m10)
bank = atan2(-m12,m11)
except when M10=1 (north pole)
which gives:
heading = atan2(M02,M22)
bank = 0
and when M10=-1 (south pole)
which gives:
heading = atan2(M02,M22)
bank = 0

angle applied first	heading
angle applied second	attitude
angle applied last	bank

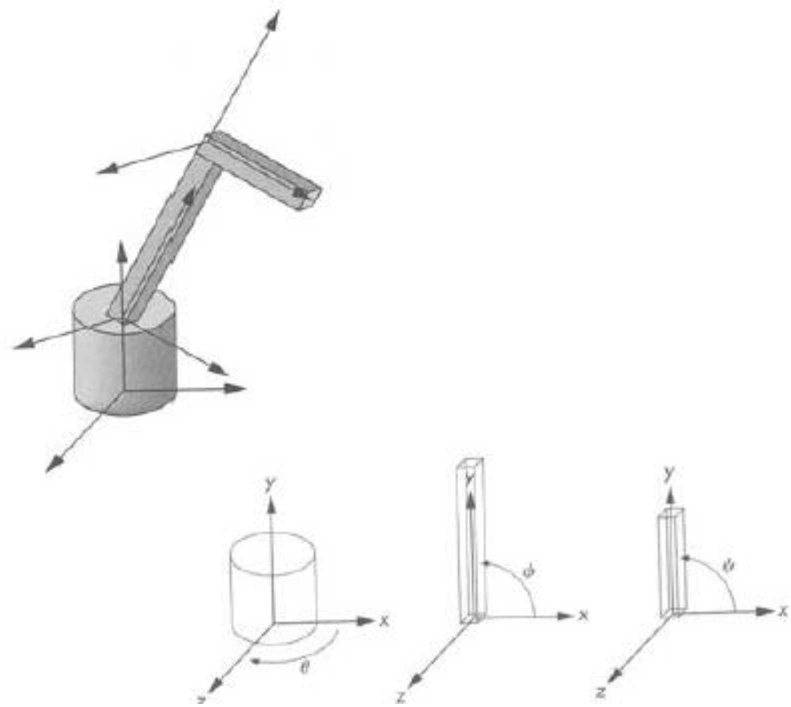
Quaternion簡介

Articulated Figures

- A figure made up of a series of links (bones) connected at joints
- Character poses described by set of rigid bodies connected by “joints”

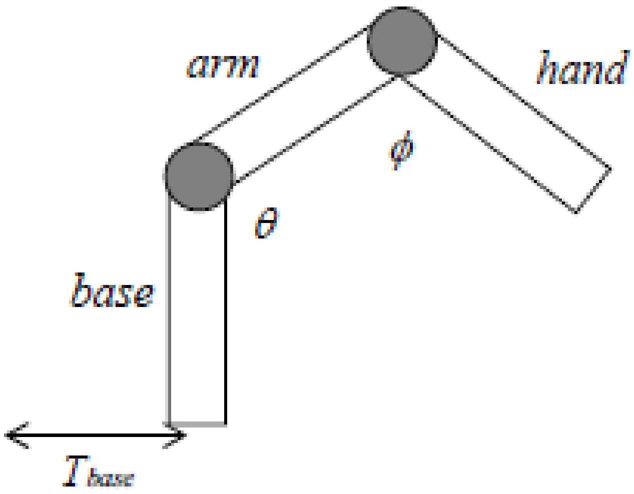


Scene Graph



Articulated Figures

- **What is a joint, exactly?**
- An ideal joint allows us to:
  - Define the **relative motion of two solids** Example: Pure rotation around an axis
  - Define a hierarchy of solids



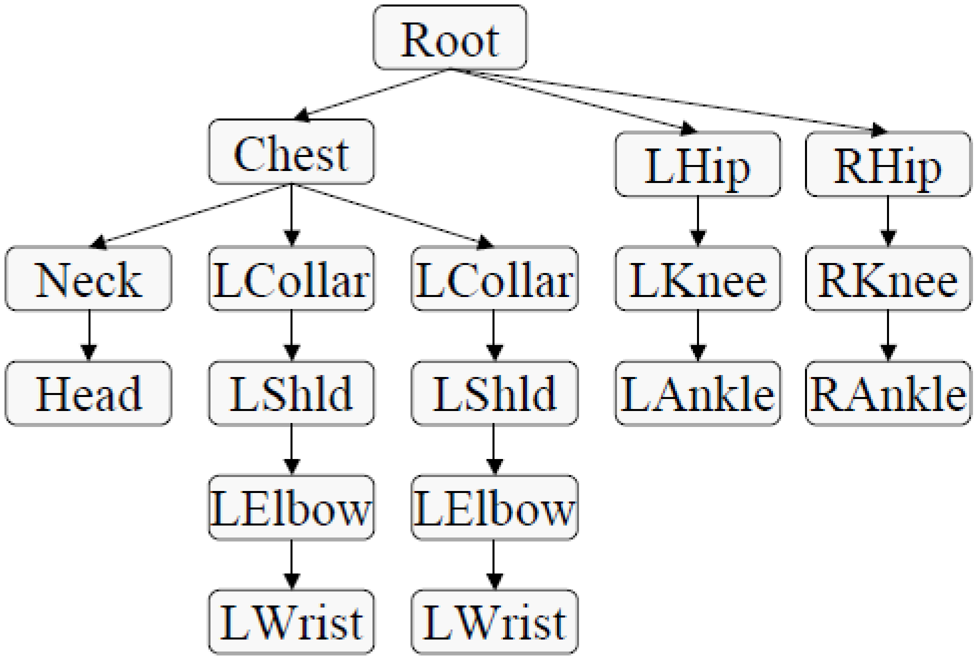
Articulated Figures

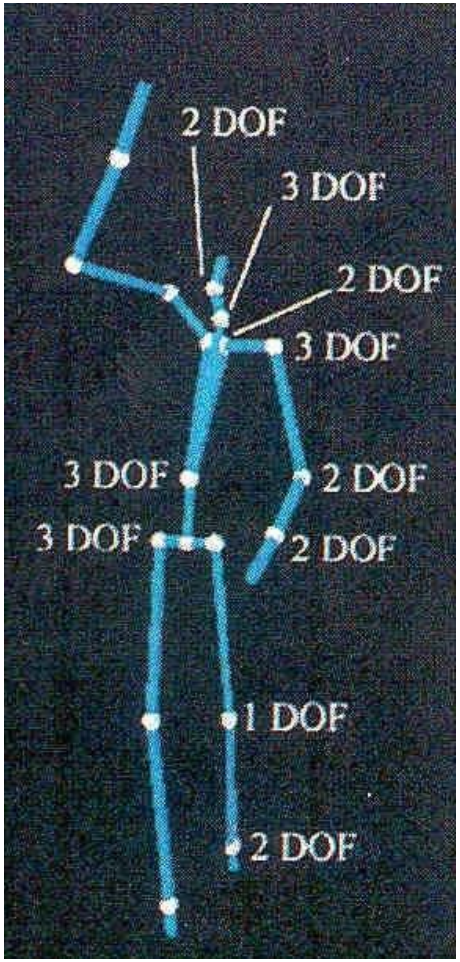
- Degrees of Freedom (DOFs)
  - **DOF of a joint:** dimensionality of independent motion between the two solids.

    Example: rotation around an axis,1 DOF (a knee).
  - **DOF of an articulated body:** sum of the DOFs of every joint.

Articulated Figures

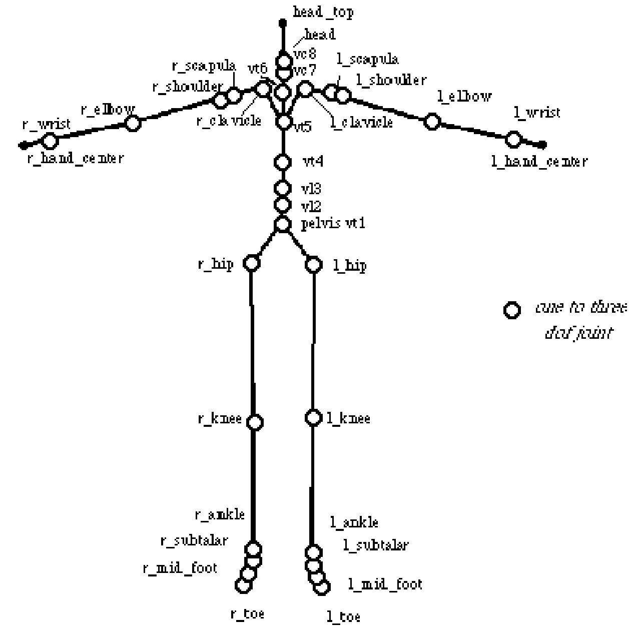
- Well-suited for humanoid characters





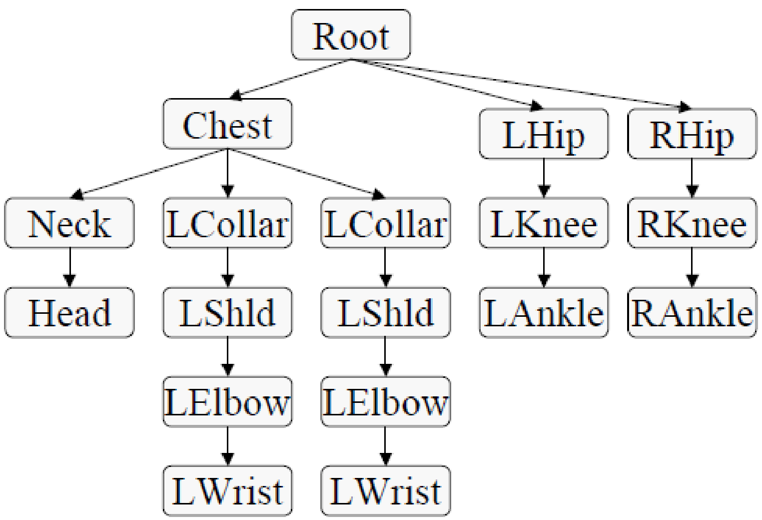
Articulated Figures

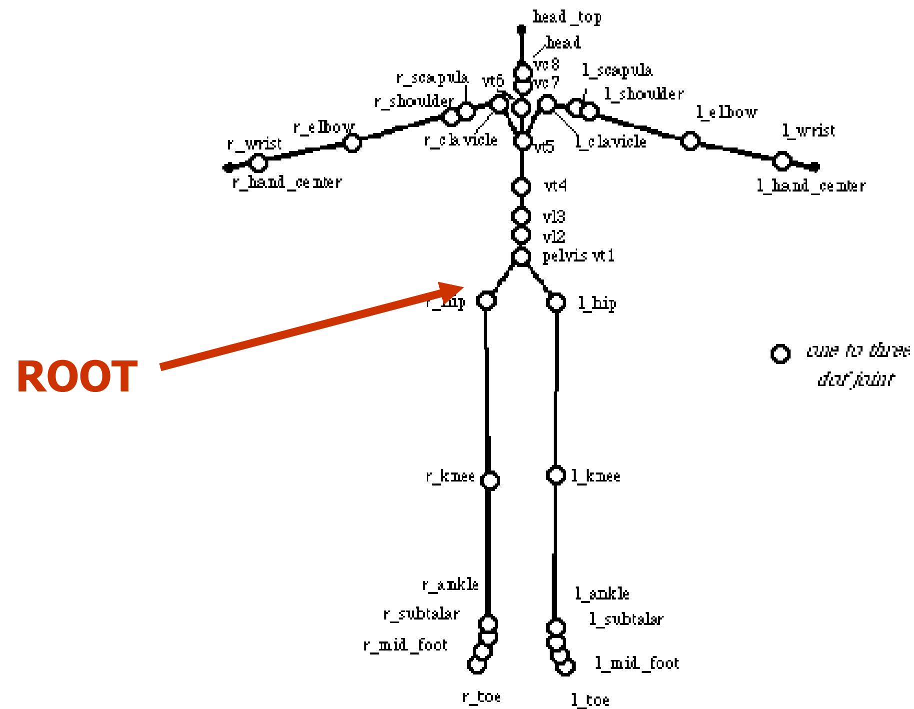
- A family of parent-child spatial relationships are functionally defined
  - Moon/Earth/Sun movements
  - Articulations of a humanoid
- Limb connectivity is built into model (joints) and animation is easier



Articulated Figures

- Model bodies (links) as nodes of a tree
- All body frames are local (relative to parent)
  - Transformations affecting root affect all children
  - Transformations affecting any node affect all its children

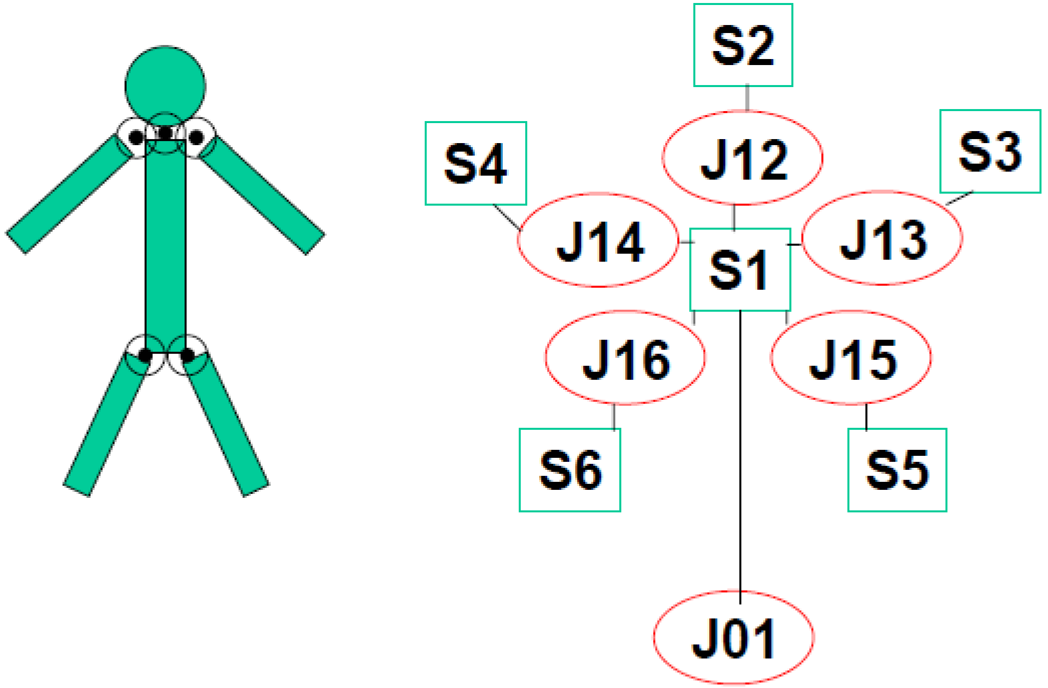




Articulated Figures

- **Kinematic graph**

  Abstract representation of an articulated body



Articulated Figures

- **Data Structure I**

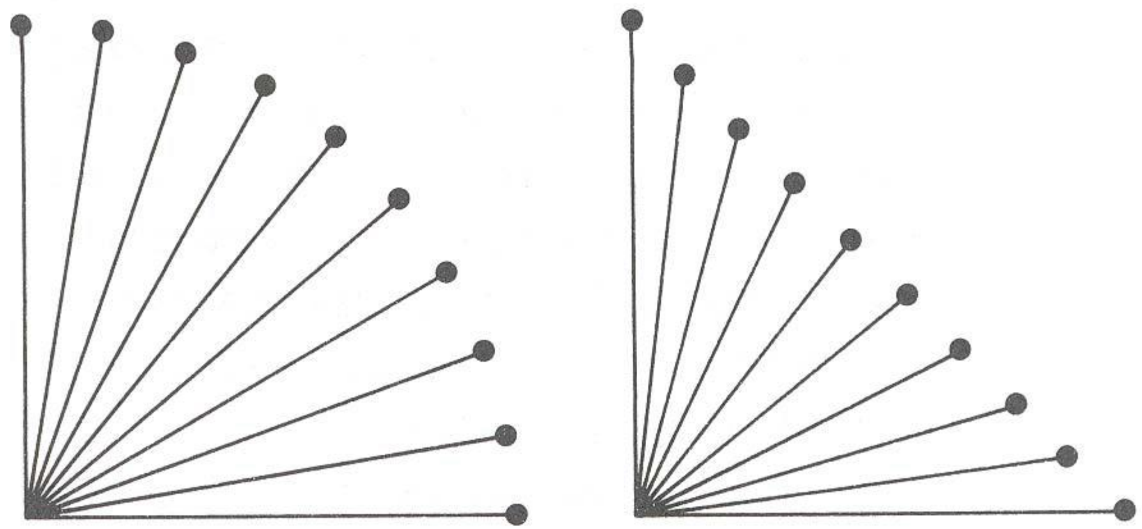
- **Graph:**
  - Root
  - Linked nodes (solids or joints)
- **Solid:**
  - Parent joint
  - (list of) child joints
  - [Transformation w.r.t. world coordinates]

Articulated Figures

- **Data Structure II**
- **Joint:**
  - Parent solid
  - Child solid
  - Transformation w.r.t. parent
  - DOF(s)
  - State variable(s) (one for each DOF)

Articulated Figures

- Inbetweening
  - Compute joint angles between keyframes

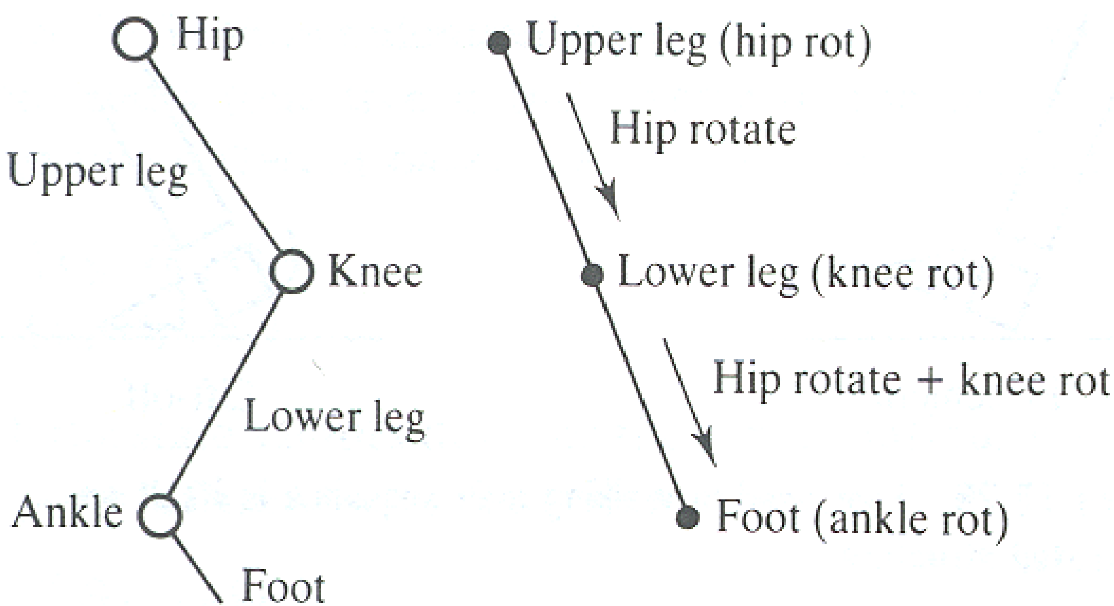


Good arm

Bad arm

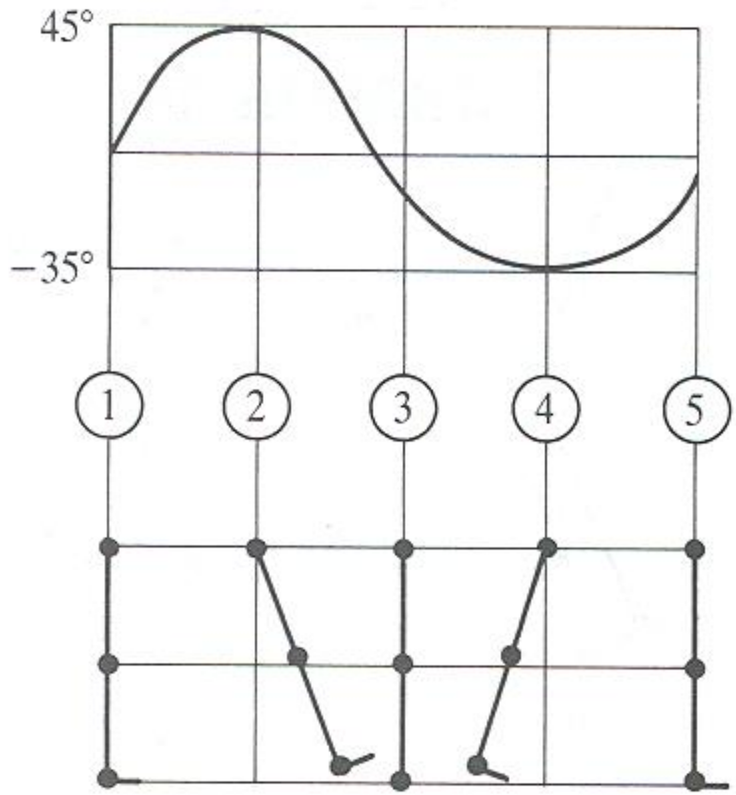
Articulated Figures

- **Example: Walk Cycle**
  - Articulated figure:



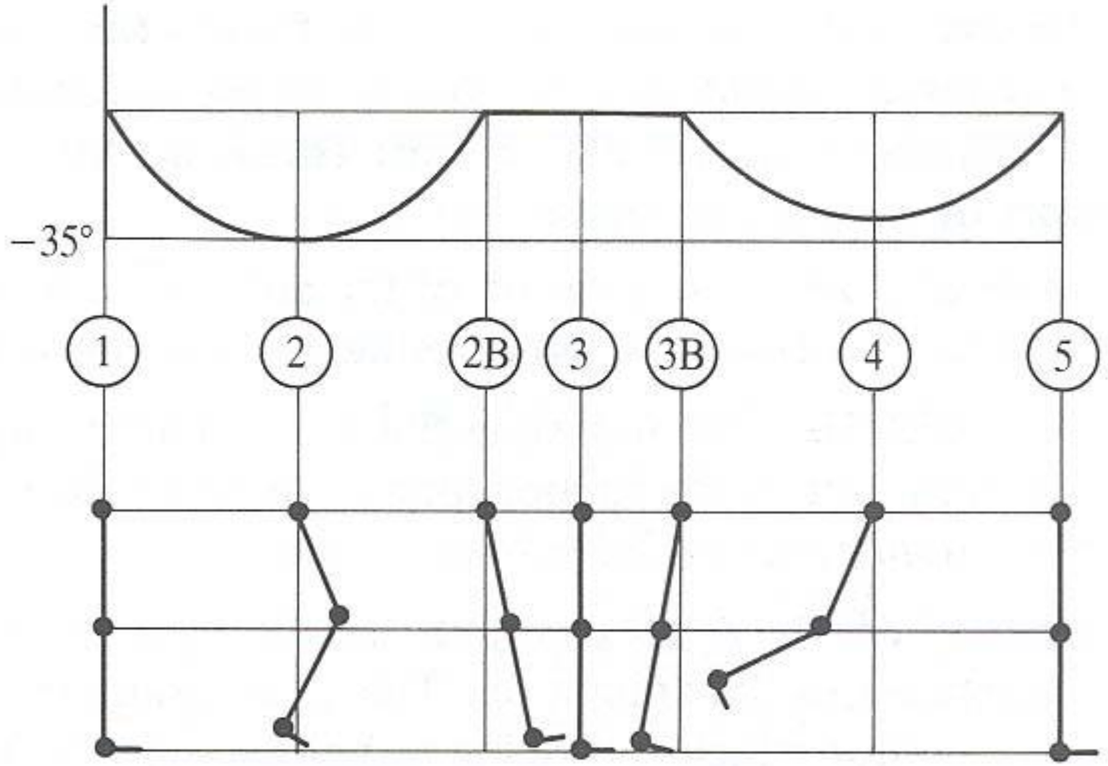
Articulated Figures

- **Example: Walk Cycle**
  - Hip joint orientation:



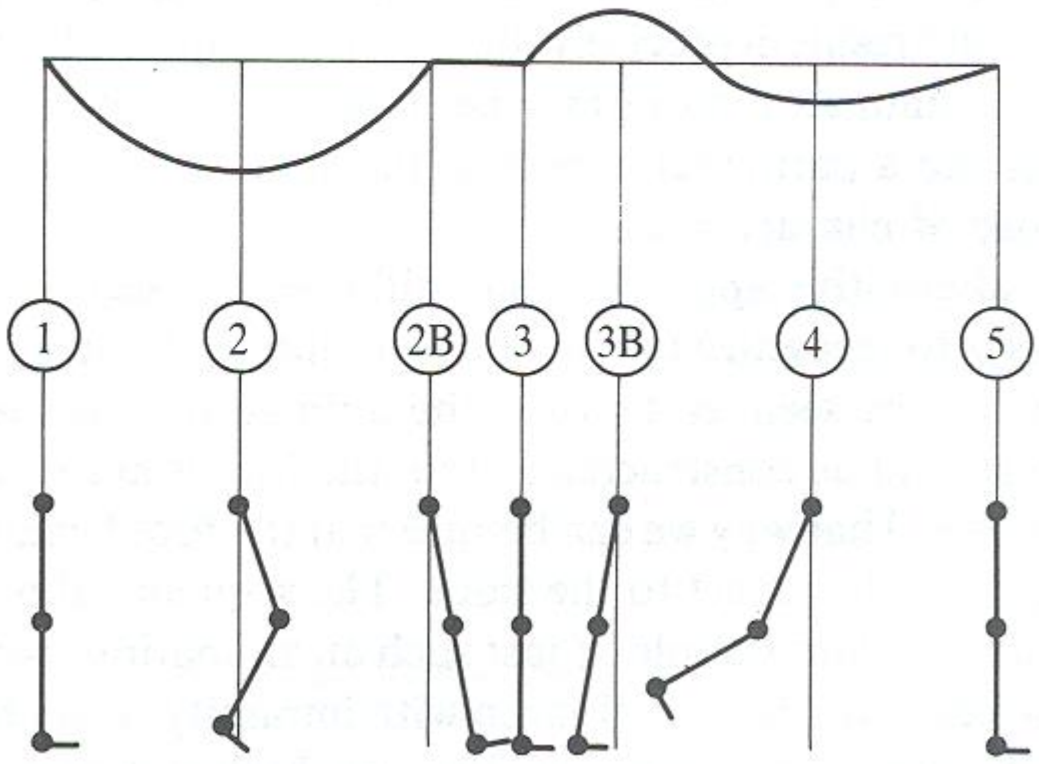
Articulated Figures

- **Example: Walk Cycle**
  - Knee joint orientation:



Articulated Figures

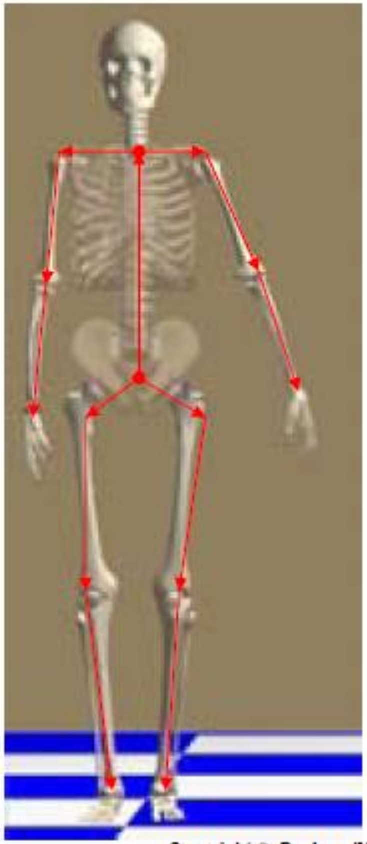
- **Example: Walk Cycle**
  - Ankle joint orientation:



Kinematics

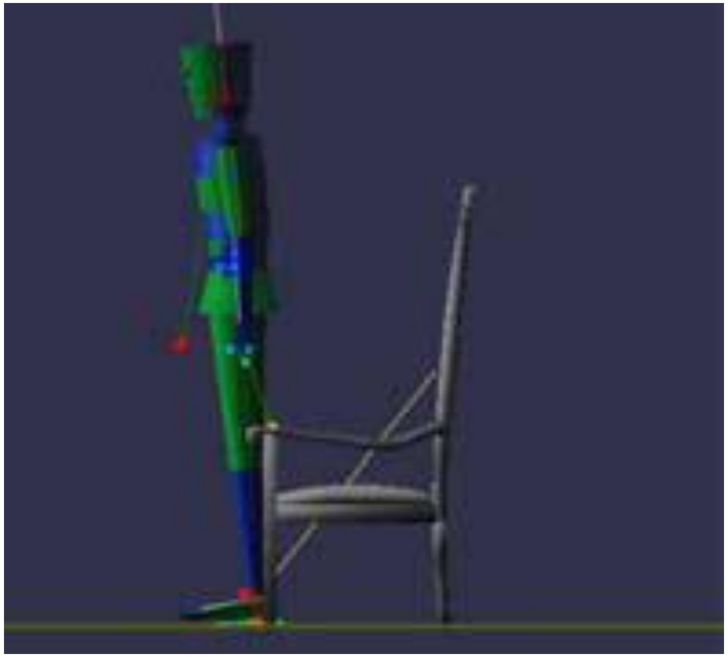
- The study or specification of motion, independent of the underlying physics that created the motion

- Considers only motion
- Determined by positions, velocities, accelerations



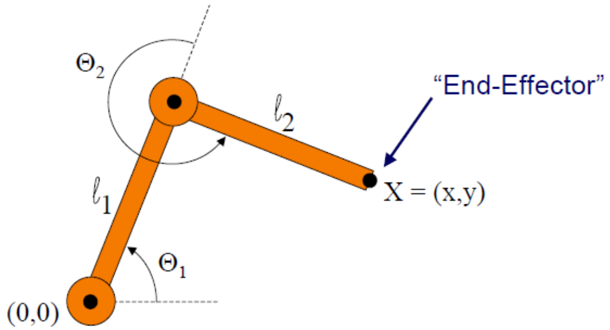
Kinematics

- Kinematic animation of articulated bodies:
  - Kinematic graph
  - Forward kinematics
  - Inverse kinematics



Kinematics

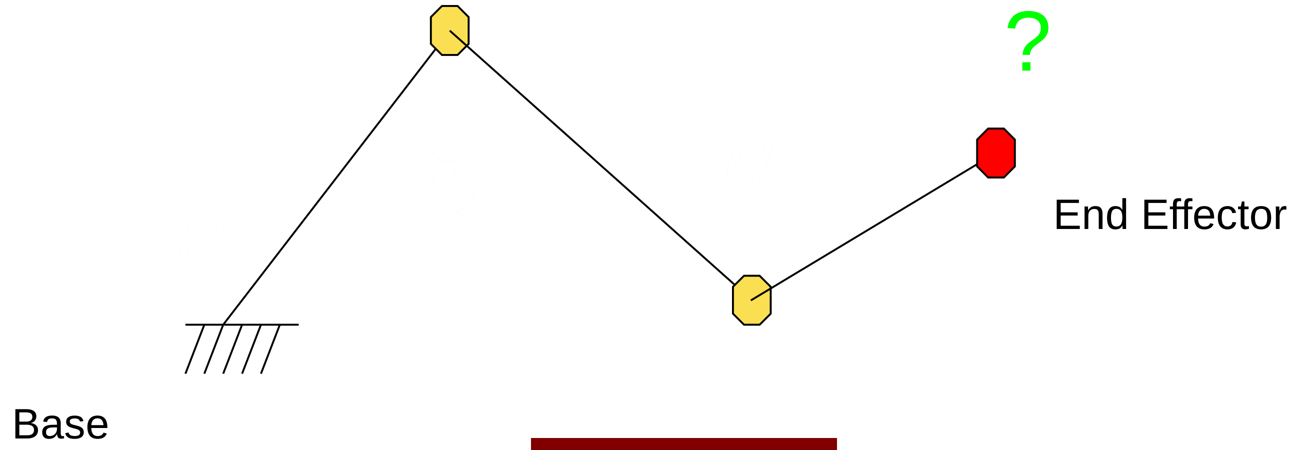
- Forward vs. Inverse Kinematics
- Forward Kinematics
  - Specify conditions (joint angles)
  - Compute positions of end-effectors
  - Good for simulation



- Inverse Kinematics
  - "Goal-directed" motion
  - Specify goal positions of end effectors
  - Compute conditions required to achieve goals
  - Good for control

Kinematics

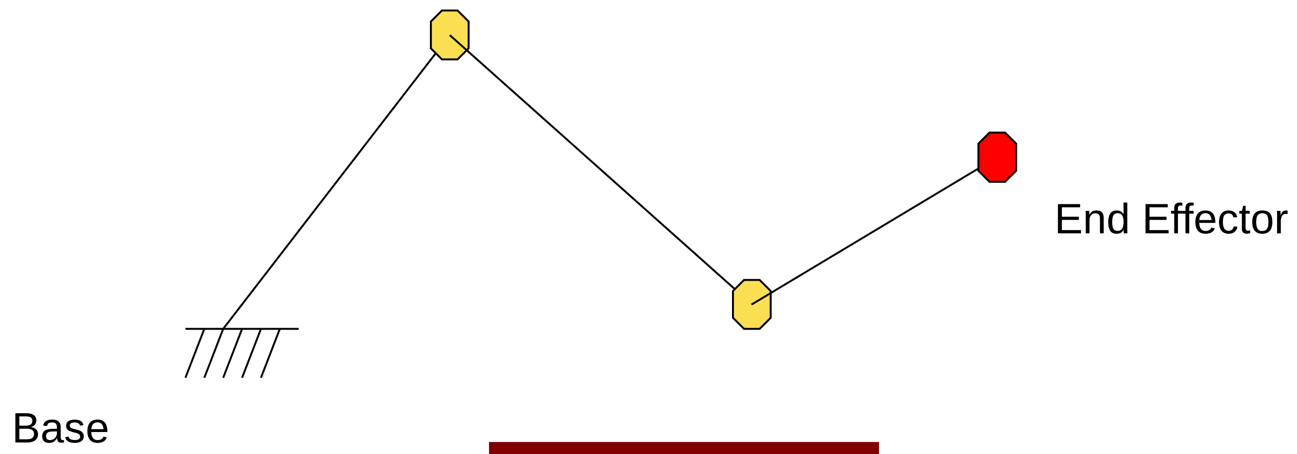
- Forward Kinematics



$$\vec{\mathbf{x}} = f(\vec{\theta})$$

Kinematics

- Inverse Kinematics



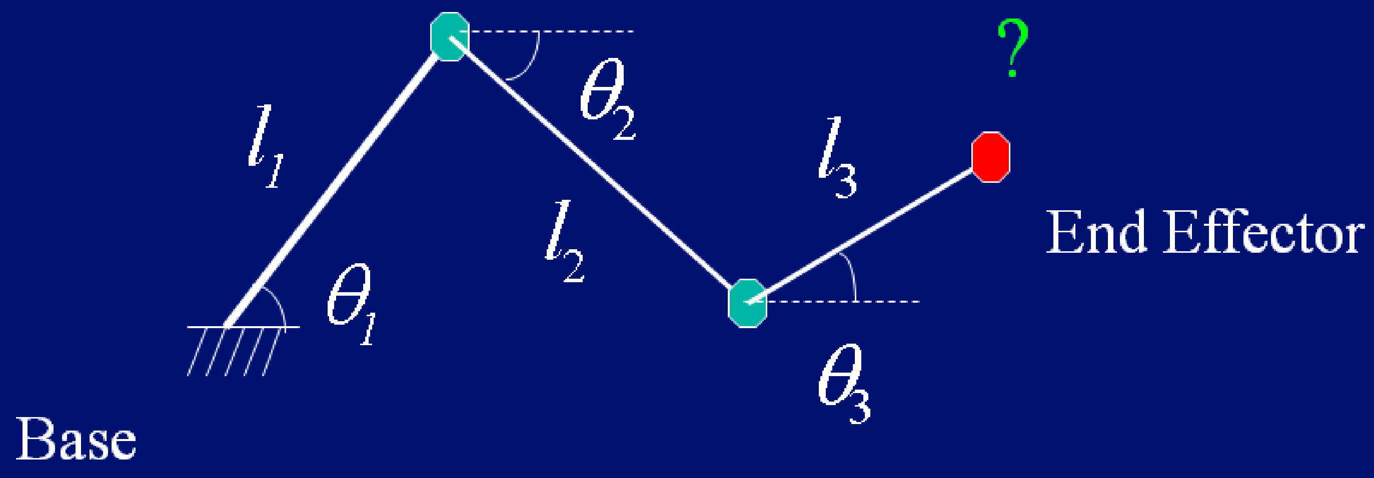
$$\vec{\theta} = f^{-1}(\vec{x})$$

Kinematics

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

Kinematics

- What does $f(\vec{\theta})$ look like?



$$x = l_1 \cos(\theta_1) + l_2 \cos(\theta_2) + l_3 \cos(\theta_3)$$

$$y = l_1 \sin(\theta_1) + l_2 \sin(\theta_2) + l_3 \sin(\theta_3)$$

Kinematics

- Solution to $\vec{\theta} = f^{-1}(\vec{x})$

$$x = l_1 \cos(\theta_1) + l_2 \cos(\theta_2) + l_3 \cos(\theta_3)$$

$$y = l_1 \sin(\theta_1) + l_2 \sin(\theta_2) + l_3 \sin(\theta_3)$$

Number of equations : 2

Unknown variables : 3

→ ***Infinite number of solutions !***

Kinematics

- Redundancy

- System DOF > End Effector DOF

- Our example

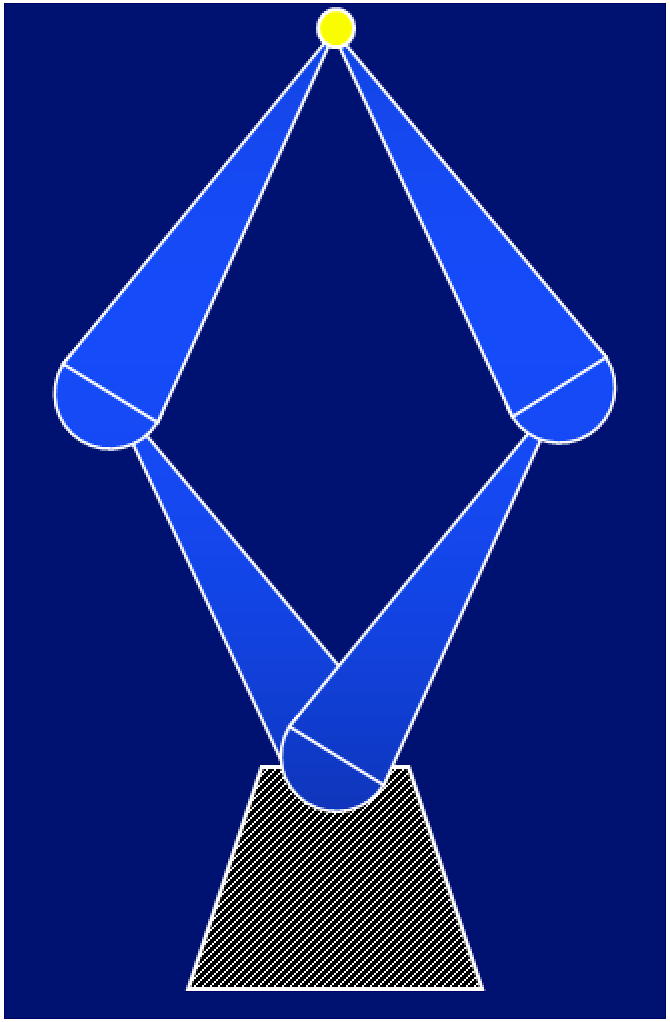
$$x = l_1 \cos(\theta_1) + l_2 \cos(\theta_2) + l_3 \cos(\theta_3)$$

$$y = l_1 \sin(\theta_1) + l_2 \sin(\theta_2) + l_3 \sin(\theta_3)$$

  - System DOF = 3
  - End Effector DOF = 2

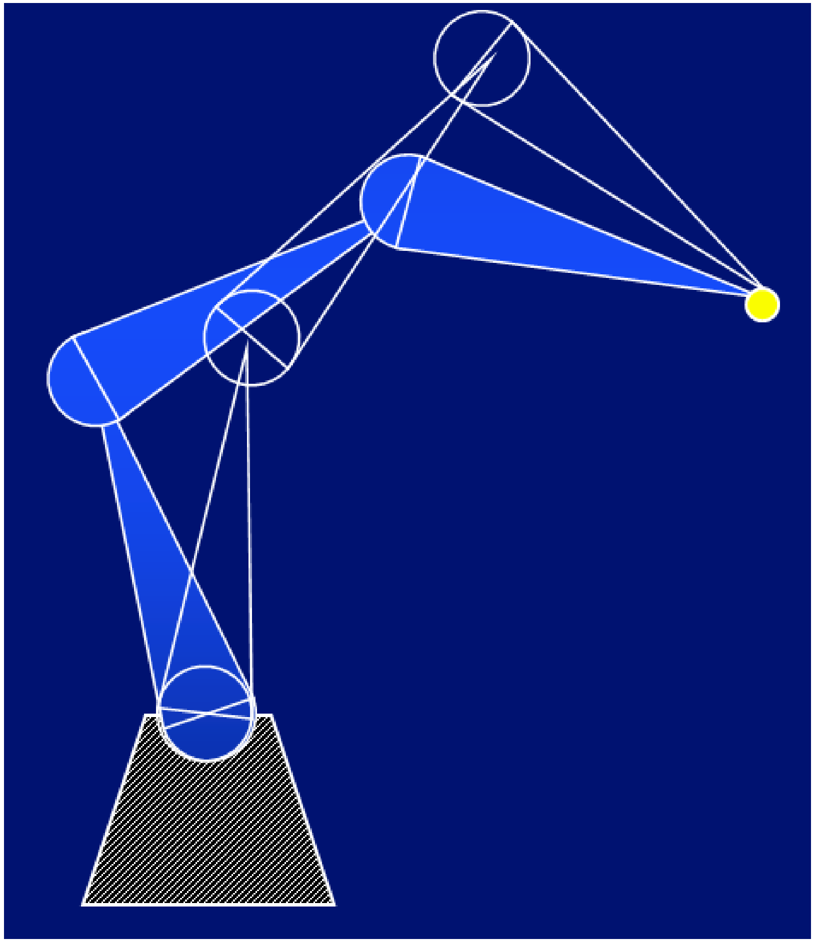
Kinematics

- Failures of simple IK
- Multiple Solutions



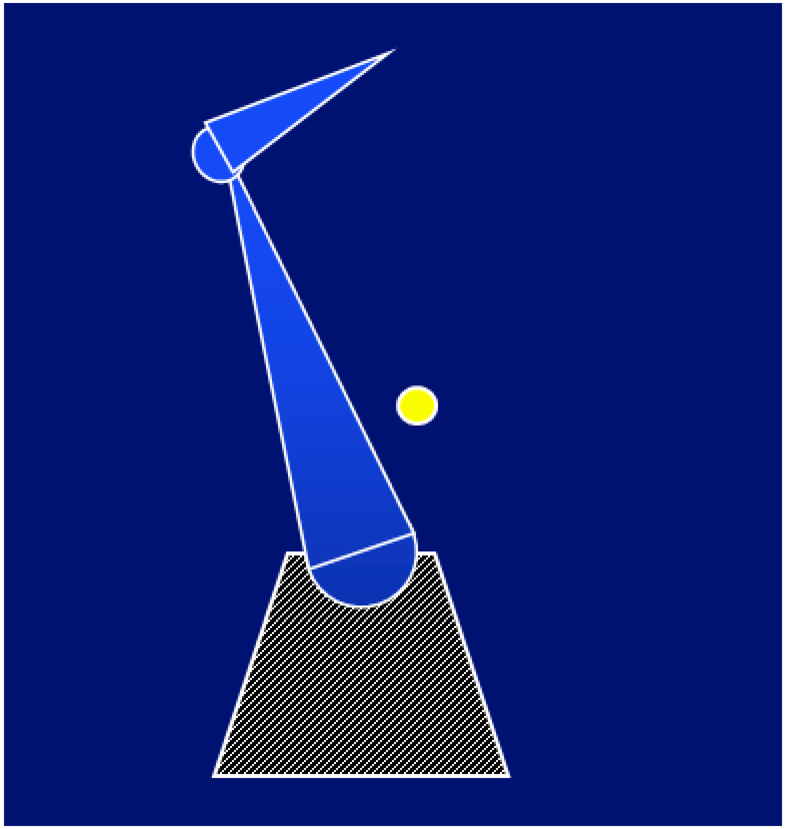
Kinematics

- Failures of simple IK
- Infinite solutions



Kinematics

- Failures of simple IK
- Solutions may not exist



Kinematics

- To solve the IK problem, consider Naturalness
  - Based on observation of natural human posture
  - Neurophysiological experiments

We will formulate the constraints as:

$$C(\theta) = 0$$

Then we need to solve for θ

Constraint Types:

- Position
- Orientation
- Pointing
- etc.

Kinematics

- Problem

Alas, there are many ways to meet constraints

(try to enumerate the number of ways you can pick up something on the floor)

Some of these solutions are pretty bad

(you can for instance pick up something while standing only on one leg, with your foot on the top of your head – don’t try this at home)

This is a GOOD thing! We can pick the one(s) we really want (minimizing energy for instance).

Kinematics

- **Minimize a "Goodness" Metric**

We can define a metric G over the set of solutions

- minimal power consumption
- least deviation from rest pose
- etc.

The problem is then:

Minimize $G(\theta)$ subject to the constraint $C(\theta) = 0$

But **C** is highly nonlinear!!

Kinematics

- **Linearization**
  - One solution is to locally linearize the problem, and iterate until convergence.
  - Numerical methods such as Lagrange Multipliers can then be applied.
  - But we need first-order derivatives!!

Kinematics

- **Jacobian Matrix**

First-order derivatives = Jacobian matrix

Jacobian of the constraint C:

$$J = \begin{bmatrix} \frac{\partial C_1}{\partial \theta_1} & \cdots & \frac{\partial C_1}{\partial \theta_n} \\ \vdots & & \vdots \\ \frac{\partial C_m}{\partial \theta_1} & \cdots & \frac{\partial C_m}{\partial \theta_n} \end{bmatrix}$$

Tells us how the constraints move when the state vector is slightly changed.

Kinematics

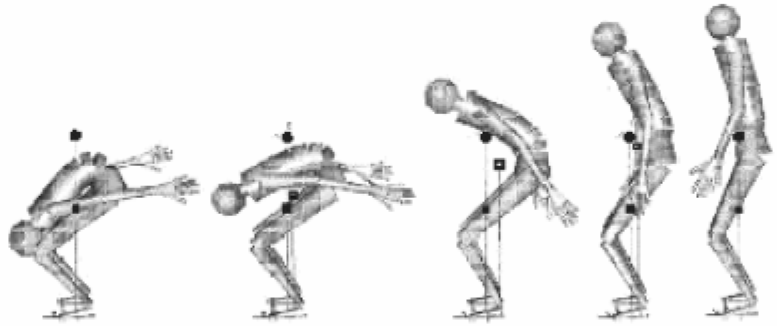
- **Examples of Metrics**

All the solutions lie in what we call the *null space* of J (set of possible moves that have no influence on the constraints)

- Restricting the set to "comfortable" solutions:

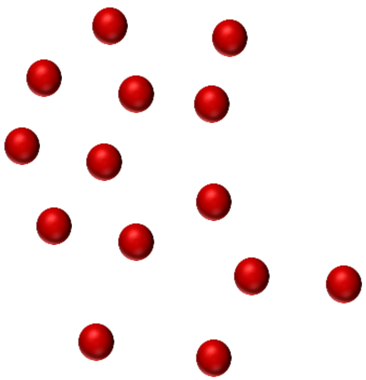
$$G(\theta) = \left\| \theta - \theta_{comfortable} \right\|$$

- Maintaining the center of mass above the feet



Particles

- Particles are objects modeled as point masses
- Particle properties:
  - Mass
  - position
  - velocity
  - force accumulator
  - age, lifespan
  - rendering properties



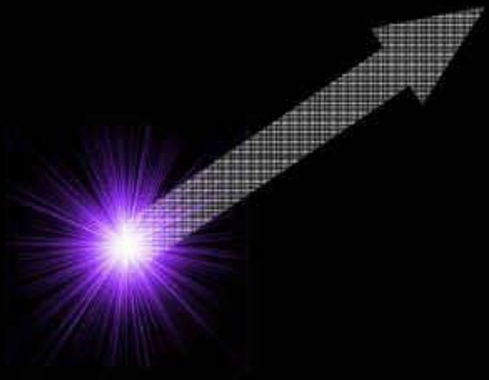
Applications

Particle Systems

- Particle systems are collections of particles
- Particle systems can represent:
  - fire
  - smoke/clouds
  - water
  - debris/shrapnel
  - soft bodies
  - flocks/crowds
  - etc.

Particle System Attributes

- Creation—number, initial condiions
  - position/velocity
    - randomness
    - surface of emitter shape
    - vertex of polygonal object
  - size
  - color
  - transparency
  - shape
  - lifetime
- Deletion
- Update of position/velocity
  - translation
  - vortex
- Rendering style – motion blur, compositing



What control handles do we want/need?

Particle Systems

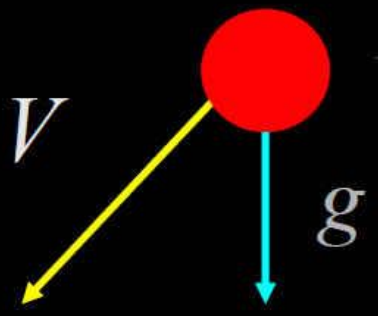
- Particles respond to forces

$$P = (x, y, z)$$

$$A = g$$

$$V' = V + A\Delta t$$

$$P' = P + \frac{V + V'}{2}\Delta t$$



Integration: accuracy improves as step size decreases but never a perfect match

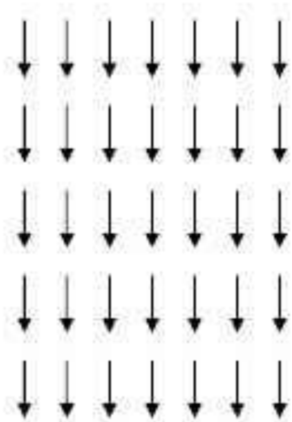
Particle System Forces

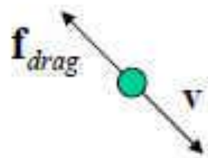
- Force fields
  - Gravity, wind, pressure
- • Viscosity/damping
  - Liquids, drag
- • Collisions
  - Environment
  - Other particles
- • Other particles
  - Springs between neighboring particles (mesh)
  - Useful for cloth

Particle System Forces

Unary forces -

forces that only depend on 1 particle



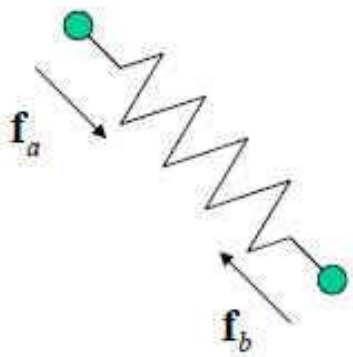


Gravity	Dampening	Wind Fields
$\mathbf{f} = m\mathbf{g}$	$\mathbf{f}_{drag} = -k_d \mathbf{v}$	$\mathbf{f} = k\mathbf{v}_{wind}$

Particle System Forces

Binary forces -

forces that only depend on 2 particles



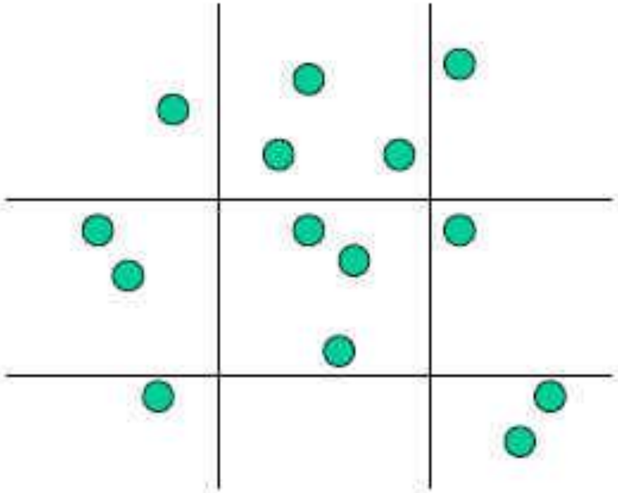
Springs

$$\mathbf{f}_a = -k_s(|\mathbf{x}_a - \mathbf{x}_b| - l_0)\frac{\mathbf{x}_a - \mathbf{x}_b}{|\mathbf{x}_a - \mathbf{x}_b|} - k_d\left(\frac{(\mathbf{v}_a - \mathbf{v}_b)\cdot(\mathbf{x}_a - \mathbf{x}_b)}{|\mathbf{x}_a - \mathbf{x}_b|}\right)\frac{\mathbf{x}_a - \mathbf{x}_b}{|\mathbf{x}_a - \mathbf{x}_b|}$$

Particle System Forces

Spatial forces -

forces that depend on local particles

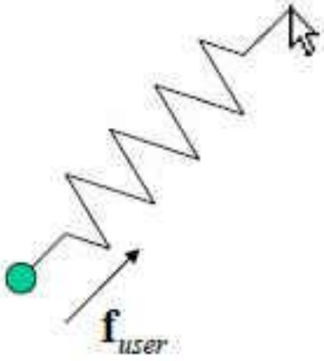


Gravity, Lennard-Jones and electric potentials

Particle System Forces

User Interaction forces -

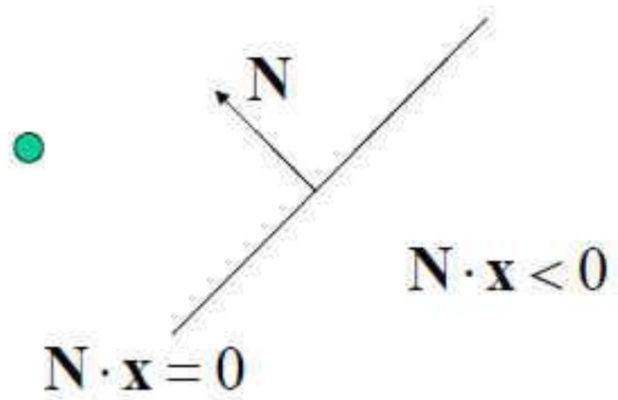
forces applied to particles by the user



Usually just use springs

Collision Detection

Determine when a particle has collided



Particle has collided iff $\mathbf{N}\cdot\mathbf{x}<0$

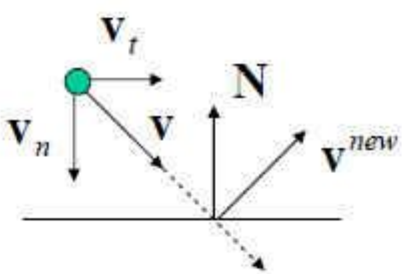
Use raytracing for more complex shapes

Collision Response

What should we do when a particle has collided?

The correct thing to do is rollback the simulation to the exact point of contact

Easier to just modify positions and velocities



After the collision:

$$\mathbf{v}^{new} = -\varepsilon \mathbf{v}_n + \mathbf{v}_t$$

coefficient of restitution

Contact Forces

When the particle is on the collision surface a contact force resists penetration

$$\mathbf{f}^c = -(\mathbf{N}\cdot\mathbf{f})\,\mathbf{f} \qquad (\mathbf{N}\cdot\mathbf{f}) < 0$$

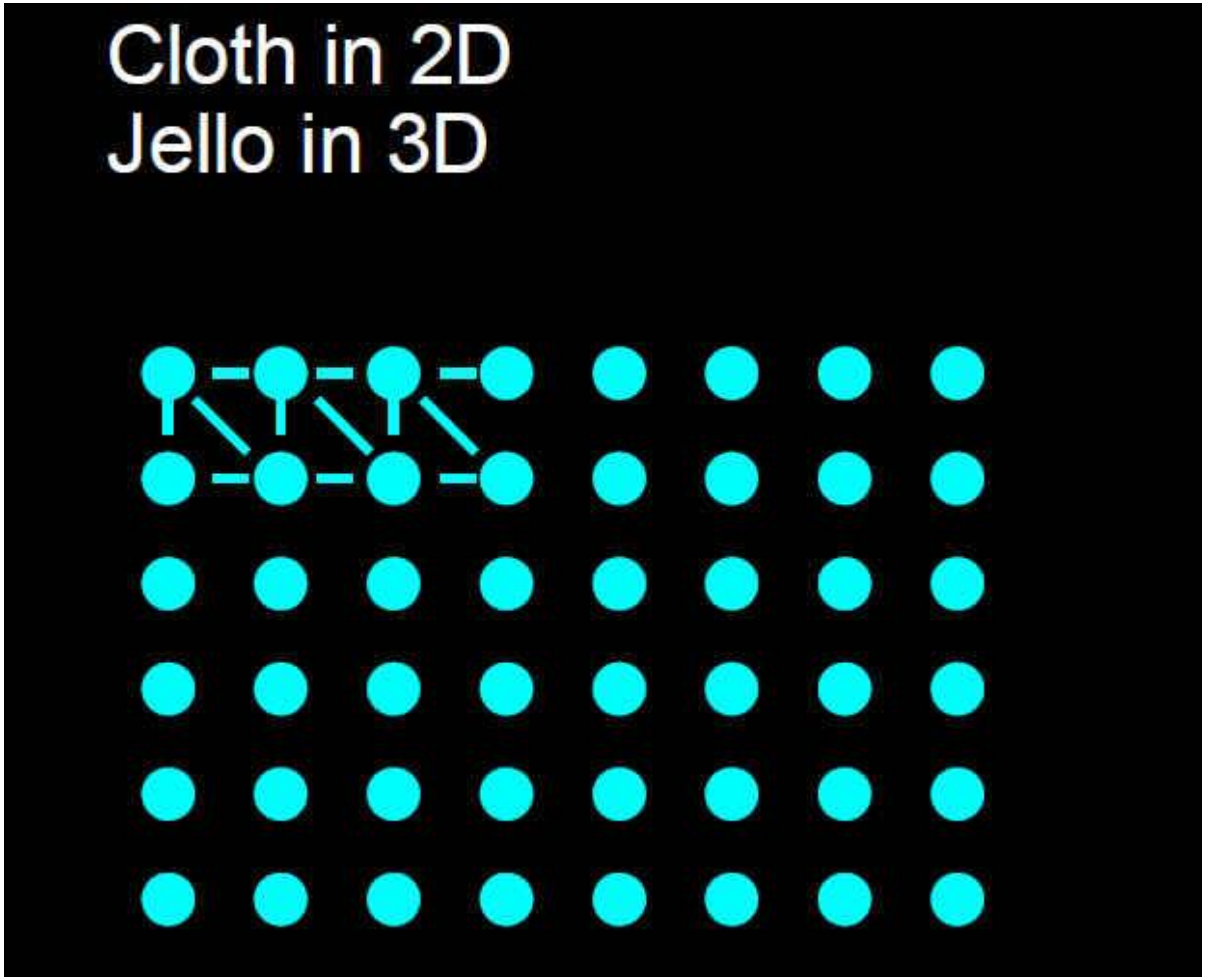
Contact forces do not resist leaving the surface

$$\mathbf{f}^c = 0 \qquad (\mathbf{N}\cdot\mathbf{f}) > 0$$

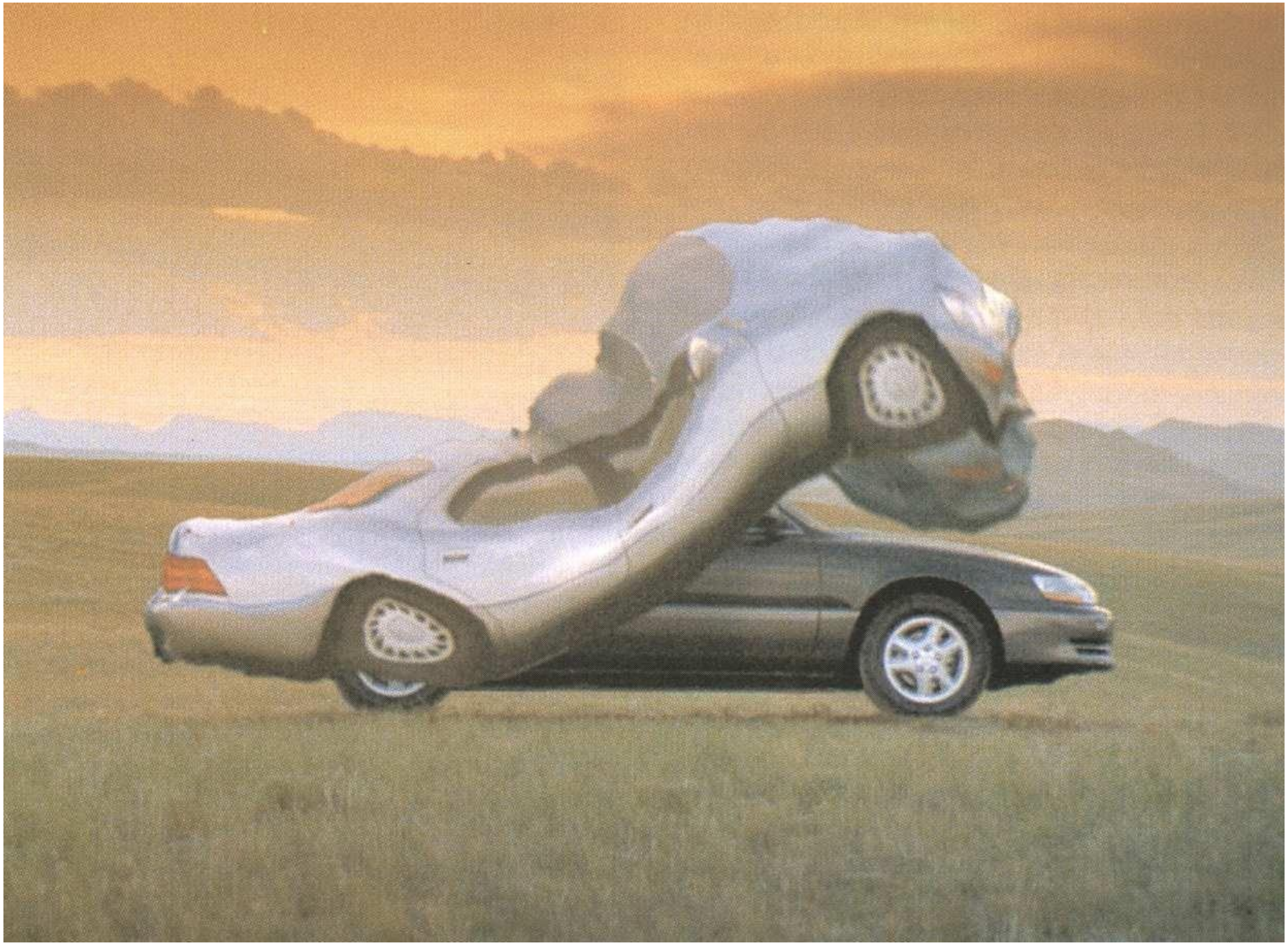
Simple friction can be modeled

$$\mathbf{f}^f = -k_f(-\mathbf{N}\cdot\mathbf{f})\,v_t \qquad (\mathbf{N}\cdot\mathbf{f}) < 0$$

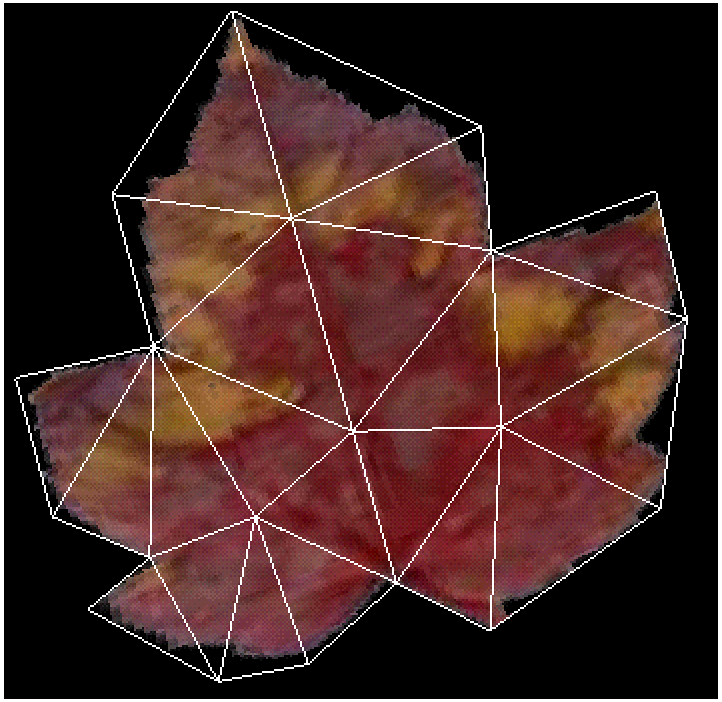
Spring-Mass Systems

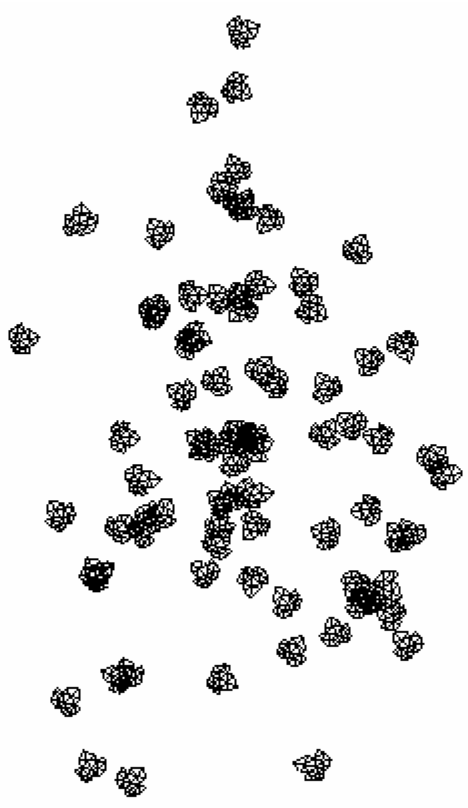


Spring-Mass Systems

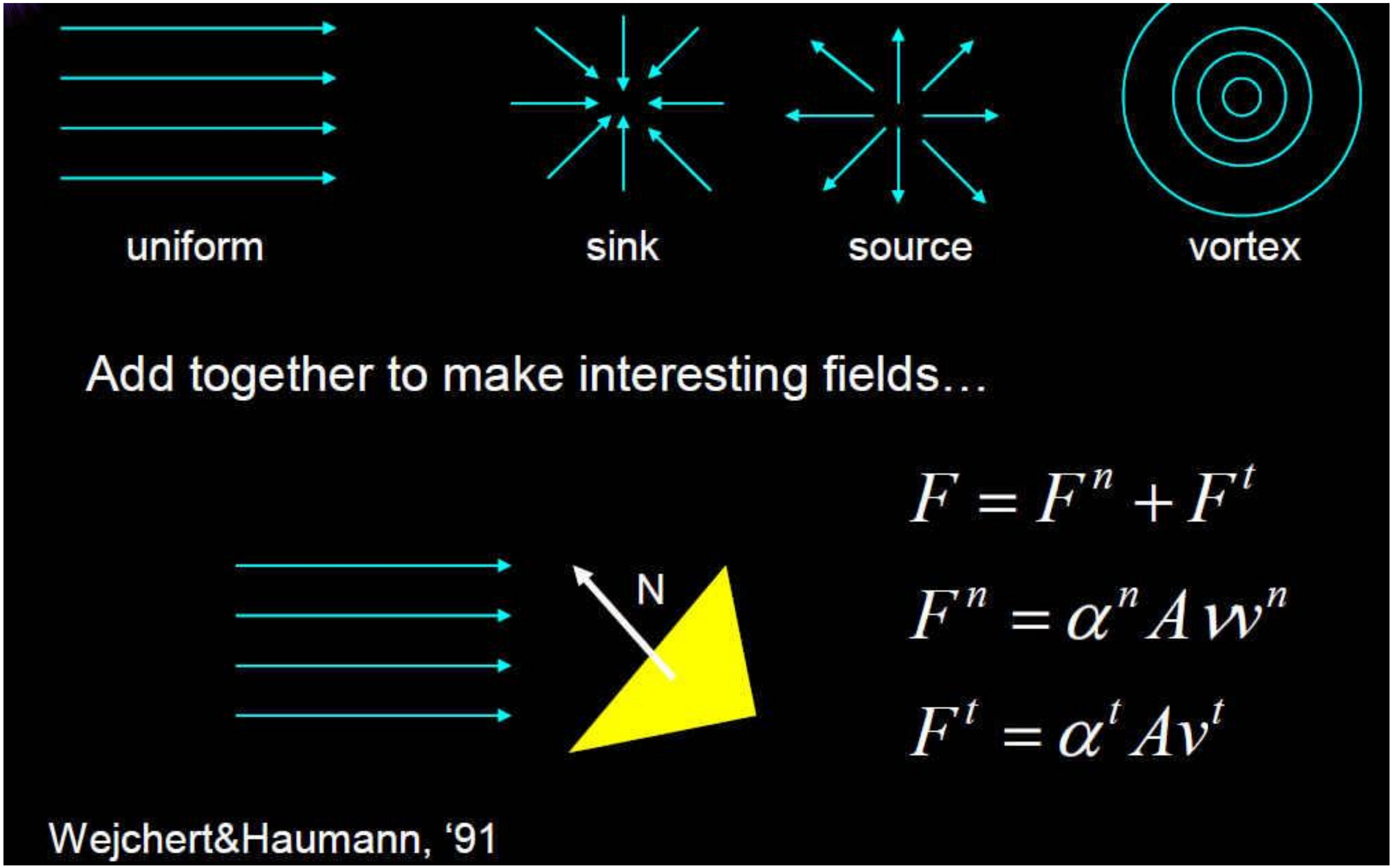


Leaves in the Wind…





Leaves in the Wind…



Demo

- Fire

http://www.youtube.com/watch?v=trOOGZhVC2E
http://youtu.be/mc0x8lGkBy0
http://youtu.be/HACFaHXe6Hg
http://youtu.be/ZgoDypGMV50

- Smoke

http://www.youtube.com/watch?v=O2Vlie5s4t4
http://youtu.be/V42sX_wz0qg
http://youtu.be/otmbEl3hopc

- Leaves

https://youtu.be/yvKGduel4CQ

- Wind

http://www.youtube.com/watch?v=zSvOS06hpxk

- Explosion

http://www.youtube.com/watch?v=vzsAU_K_7qE

Cloth Animation

Feynman (1986)

Energy minimization

- Strain Energy

$$E_s = \frac{E}{1-\nu^2}\left(u_{xx}^2 - u_{yy}^2\right) + \frac{2\nu E}{1-\nu^2} u_{xx} u_{yy}$$

- Bending Energy

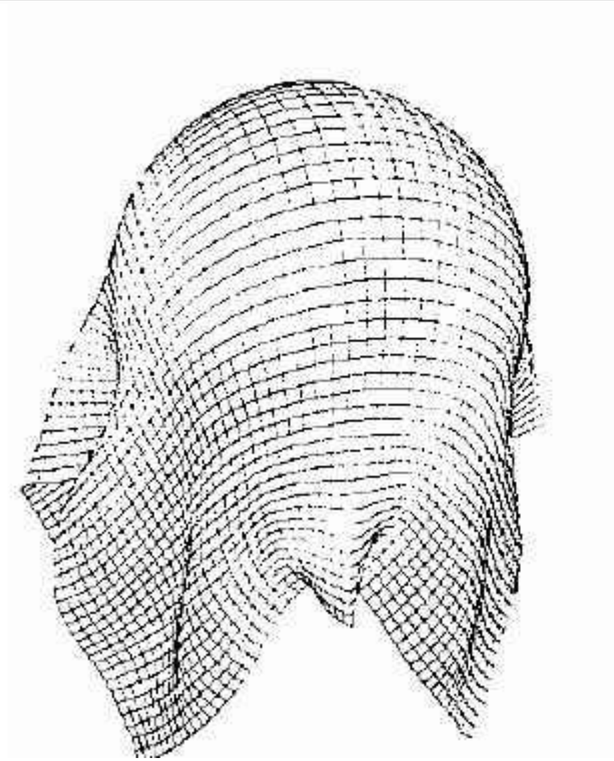
$$E_b = \iint c_1 \kappa^2 \, du dv$$

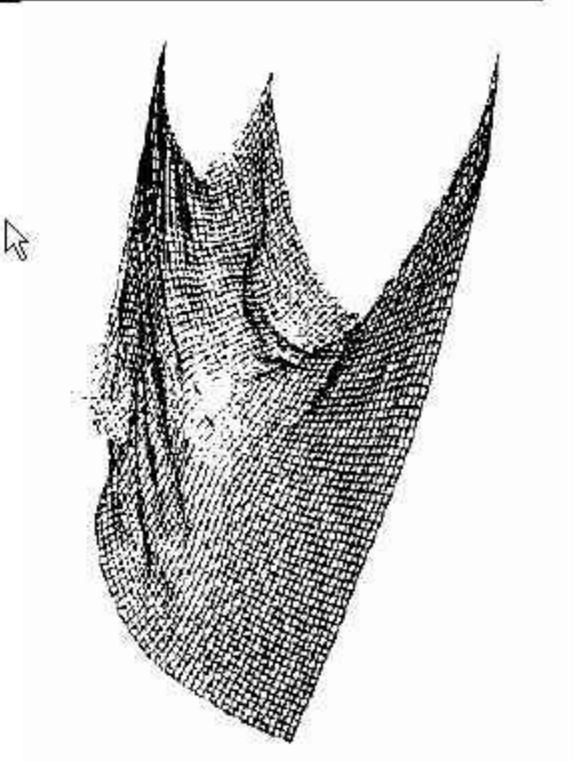
Multigrid relaxation method

Added constraints & collision detection

Cloth Animation

Feynman Results





Cloth Animation

Terzopoulos et al. (1987)

- Based on elasticity theory

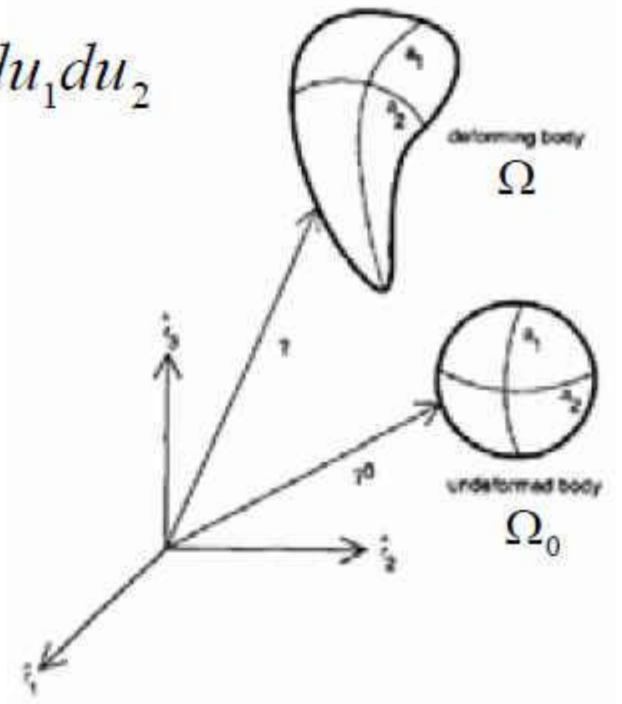
$$\varepsilon = \int_{\Omega} \left| G - G^{o} \right|^{2} + \left| B - B^{o} \right|^{2} du_1 du_2$$

- G – 1st fundamental form

$$G_{ij} = \frac{\partial \mathbf{r}}{\partial u_i} \cdot \frac{\partial \mathbf{r}}{\partial u_j}$$

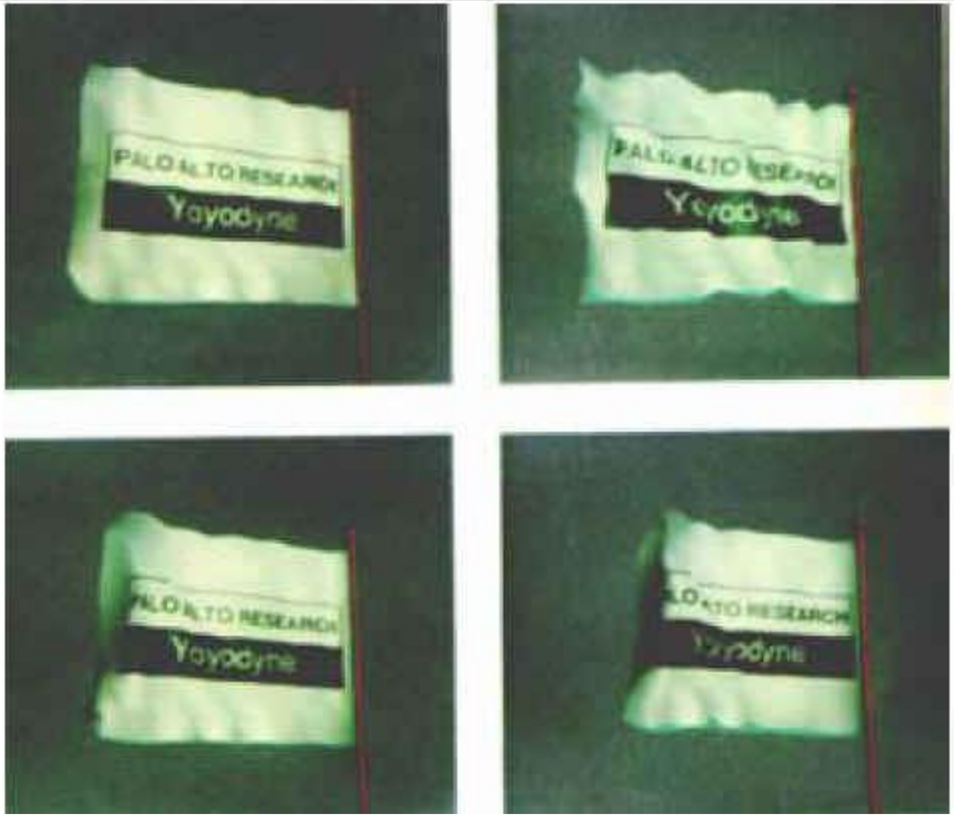
- B – 2nd fundamental form

$$B_{ij} = \mathbf{n} \cdot \frac{\partial^2 \mathbf{r}}{\partial u_i \partial u_j}$$



Cloth Animation

Terzopoulos et al. Results



Cloth Animation

Thalmann (1990 - Present)

- Extend Terzopoulos model
- Enhanced computational techniques
  - Collision detection and response
- Designing a complete set of clothing
  - User interface
  - Data structures
- Focused on clothing virtual actors

Cloth Animation

Thalmann Results

Cloth Animation

Breen, House & Wozny (1991-94)

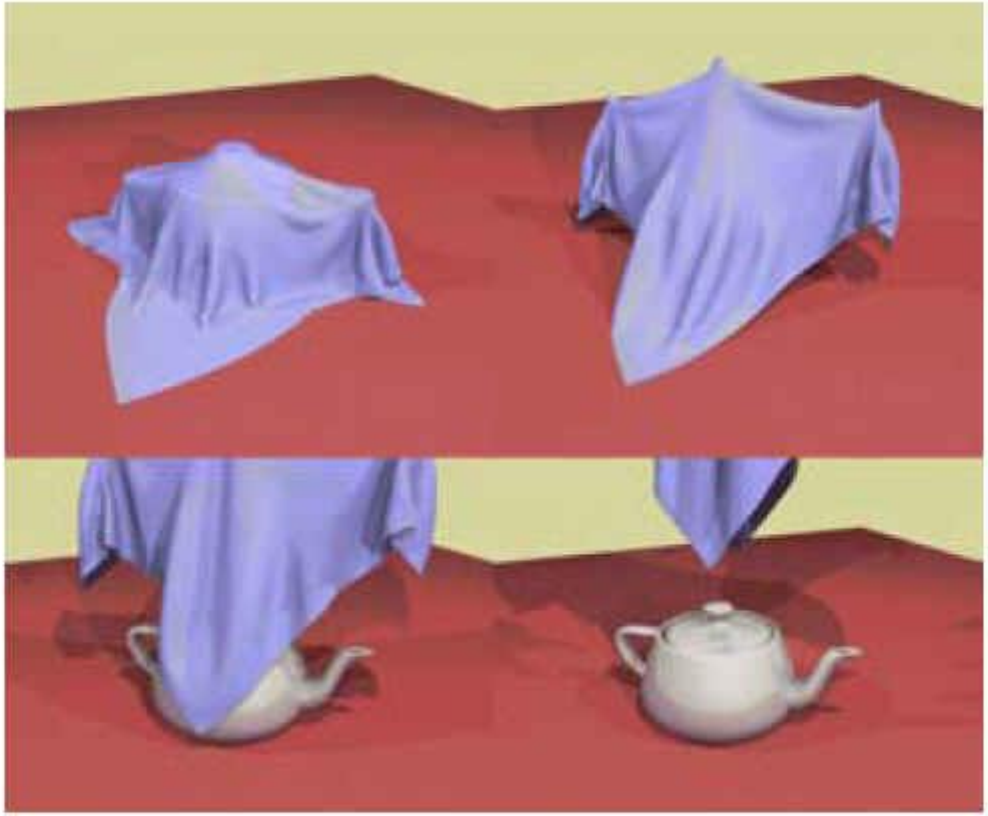
- Particle-based method
  - Macroscopic behavior arises from modeling microscopic structure
- Particles based on thread-level interactions

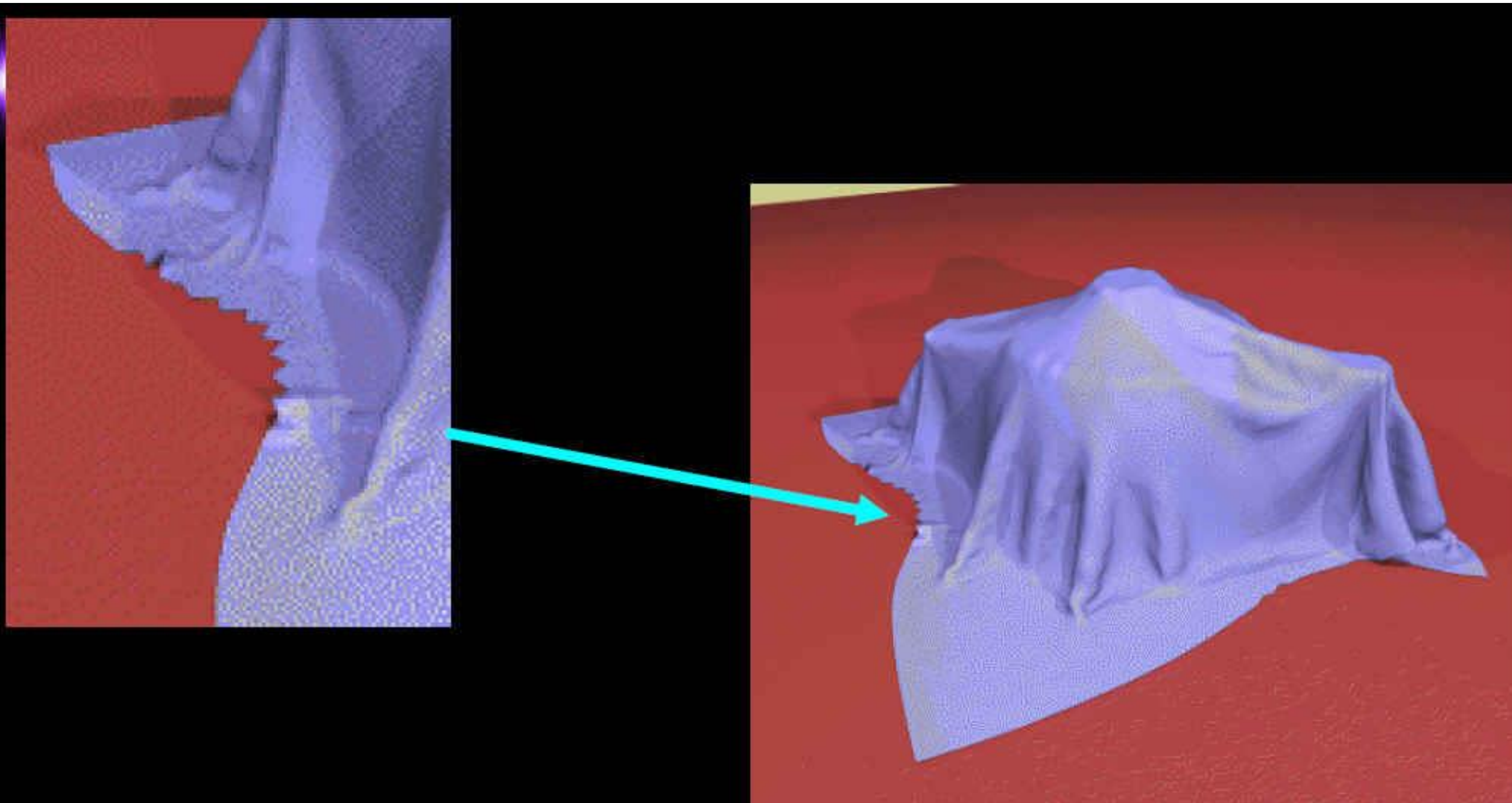
$$E = E_{stretch} + E_{shear} + E_{bend}$$

- Energy fitted to Kawabata measurements

Cloth Animation

Breen et al. Results





Breen ‘95

Increased Resolution of Mesh

+Possible Shapes

+ Smoothness

- Simulation time

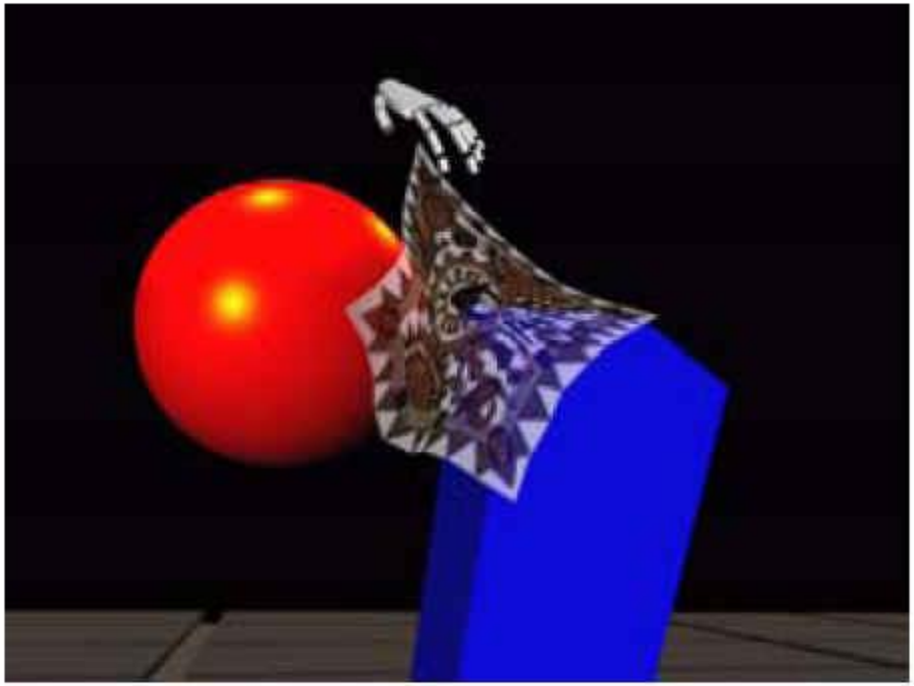
Cloth Animation

Desbrun et al. (2000)

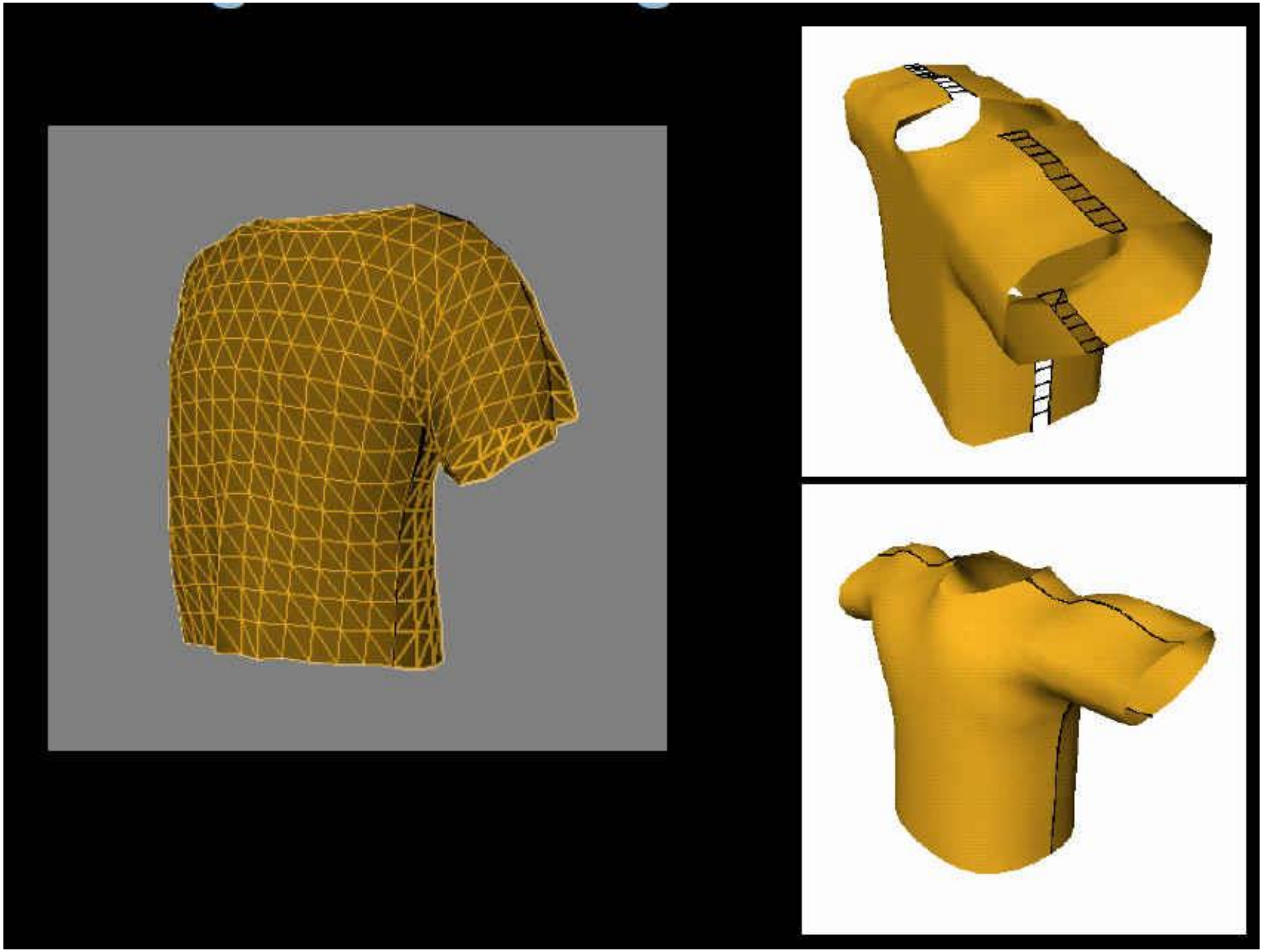
- Designed for Virtual Reality:
  - Interactive rates
  - Unconditionally stable/robust
- Mass-spring system
- Predictor/Corrector Implicit Integration
  - Addition of artificial viscosity
  - Filters the force field
  - Correct to maintain momenta
- Post-step relaxation

Cloth Animation

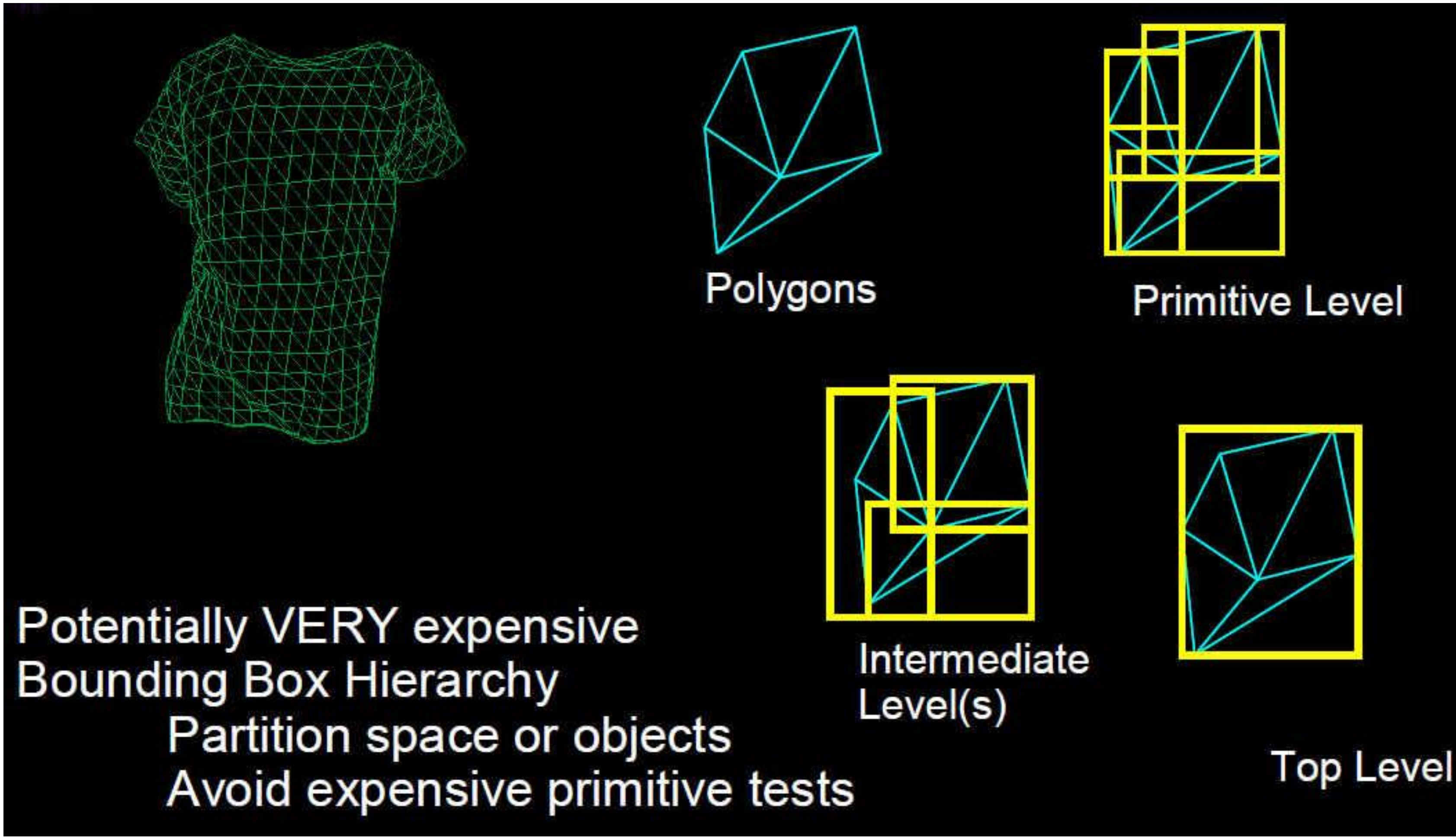
Desbrun et al. Results



Modeling for Clothing



Collisions for Clothing



Clothes Simulation

DEMO 1

DEMO 2

Fluids Animation

What is a Fluid?

A *fluid* is a substance which deforms continuously under the application of a shear stress

Examples of fluids
- liquids
- gases

A fluid
- cannot resist shear stress
- can resist normal stress (pressure)

Fluids Animation

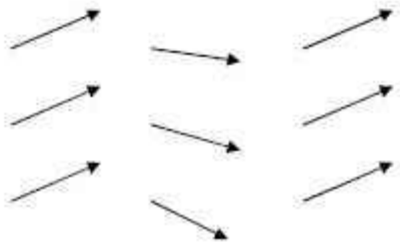
Representation of Fluids

Lagrangian

Individual particles are observed through time

Eulerian

Fixed positions are observed through time



Fluids Animation

Total Acceleration

For an Eulerian representation, acceleration is:

$$\mathbf{a} = \frac{d\mathbf{v}}{dt} = \frac{\partial \mathbf{v}}{\partial t} + \frac{\partial \mathbf{v}}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial \mathbf{v}}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial \mathbf{v}}{\partial z}\frac{\partial z}{\partial t}$$

$$\mathbf{a} = \frac{d\mathbf{v}}{dt} = \frac{\partial \mathbf{v}}{\partial t} + \frac{\partial \mathbf{v}}{\partial x}v_x + \frac{\partial \mathbf{v}}{\partial y}v_y + \frac{\partial \mathbf{v}}{\partial z}v_z$$

$$\mathbf{a} = \frac{d\mathbf{v}}{dt} = \underbrace{\frac{\partial \mathbf{v}}{\partial t}}_{\text{local acceleration}} + \underbrace{\mathbf{v}\cdot\nabla\mathbf{v}}_{\text{advective acceleration}}$$

Fluids Animation

Fluid Properties

Viscosity

a measure of friction in the fluid

$$\mathbf{F} = \mu \nabla^2 \mathbf{v}$$

Surface Tension

a skin-like stress at the fluid's surface

Pressure

a normal stress (force/area)

$$\mathbf{F} = -\nabla p$$

Fluids Animation

Navier Stokes

A fluid responds to several forces:

$$\rho\,\mathbf{a} = \mathbf{F}_{\text{body}} + \mathbf{F}_{\text{pressure}} + \mathbf{F}_{\text{viscosity}}$$

$$\rho\,\mathbf{a} = \mathbf{F}_{\text{body}} - \nabla p + \mu \nabla^2 \mathbf{v}$$

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v}\right) = \mathbf{F}_{\text{body}} - \nabla p + \mu \nabla^2 \mathbf{v}$$

$$\boxed{\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v} = \mathbf{a}_{\text{body}} - \frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2 \mathbf{v}}$$

Fluids Animation

Simplified Navier Stokes

Assuming uniform pressure and no body forces:

$$\frac{\partial \mathbf{v}}{\partial t} = -(\mathbf{v} \cdot \nabla)\mathbf{v} + \frac{\mu}{\rho} \nabla^2 \mathbf{v}$$

Similar equations can be derived for other quantities:

$$\frac{\partial p}{\partial t} = -(\mathbf{v} \cdot \nabla)p + \frac{\kappa}{\rho} \nabla^2 p$$

The velocity field should also conserve mass:

$$\nabla \cdot \mathbf{v} = 0$$

Fluids Animation

Computer Graphics Fluids

Early models were extremely coarse
- particles simulate spray
- height field for fluid surface

Waves using Fourier synthesis
- generate noise
- multiply by power spectrum

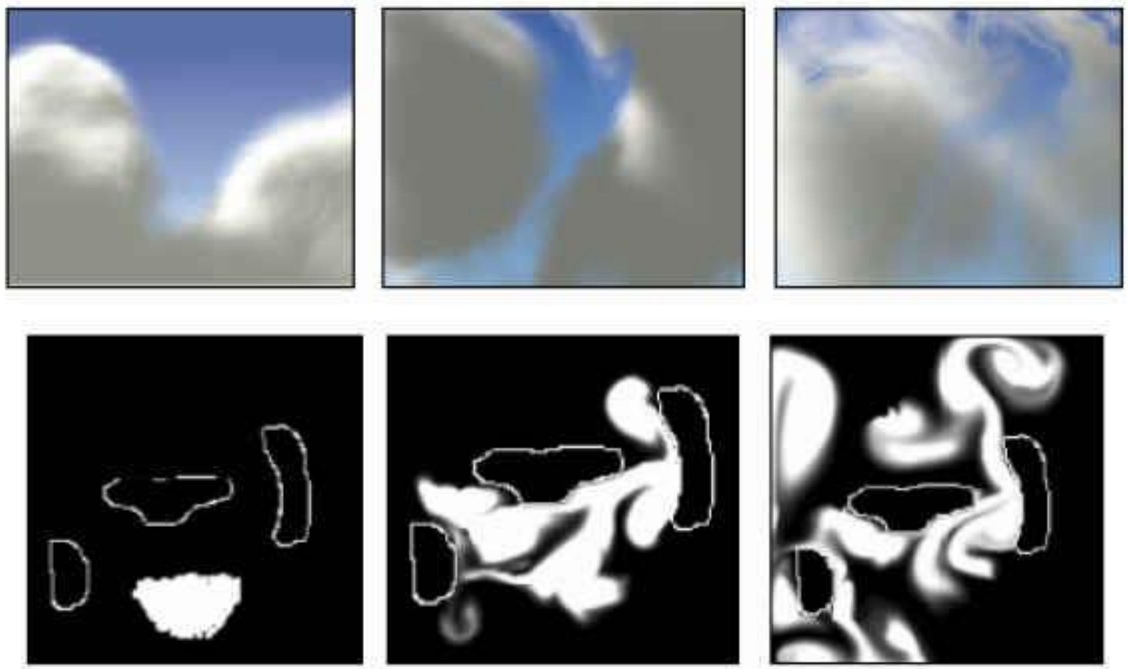
Shallow water approximation [Kass and Miller '92]
- diffusion process – good for ripples

First Navier Stokes model [Foster and Metaxas '93-'95]

Fluids Animation

Stable Fluids [Stam '99]

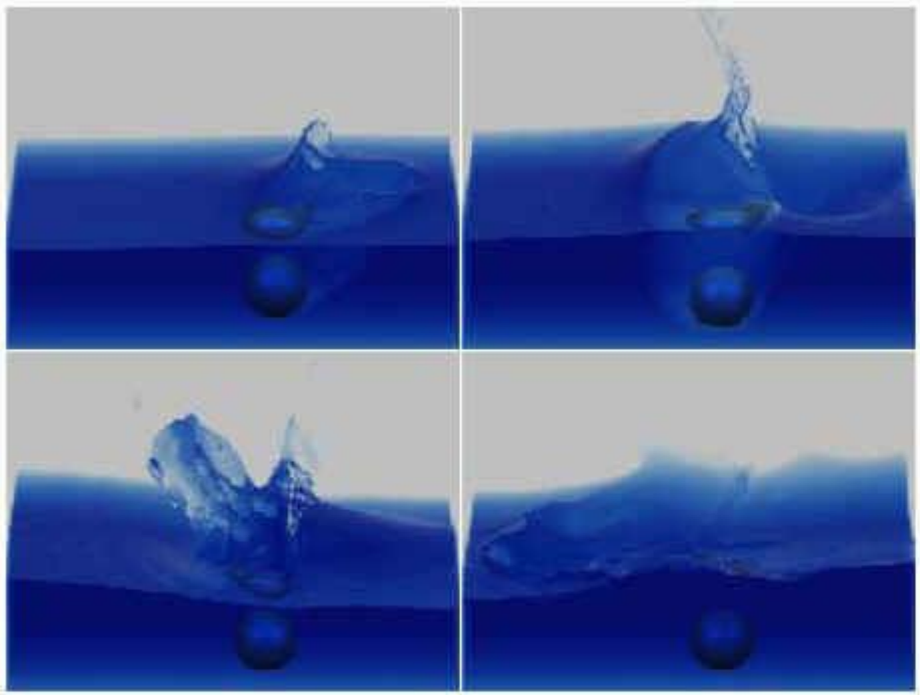
Stable semi-lagrangian Navier Stokes simulation



Fluids Animation

Free Boundary Fluids [Foster and Fedkiw '01]

Navier Stokes plus implicit surface and particles



- Water

http://www.youtube.com/watch?v=3rlbnqayGZs

http://youtu.be/-AiLyQWXjIg

http://youtu.be/Yi3LW5riHfc

http://youtu.be/Iv5vppgQOr0

- Ocean

https://youtu.be/Ccq-NgFv7_w

https://youtu.be/QahKrkTbls4

https://youtu.be/3lAM933H9Hc

Hair Simulation

DEMO 1

DEMO 2

Facial Animation

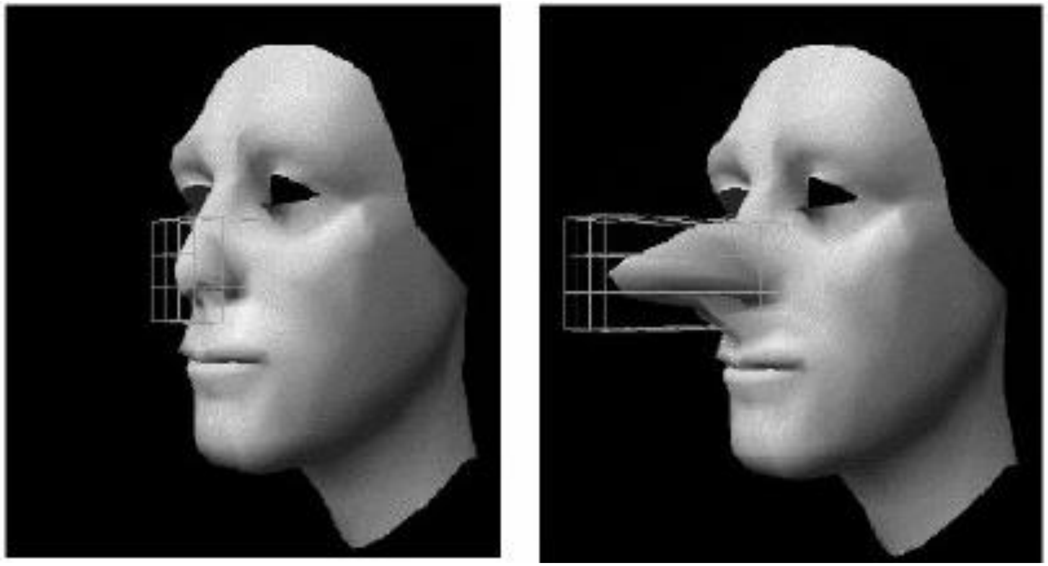
- Why is it useful ?
  - Human Computer Interaction
  - Digital actors (Xmen, Harry Potter, Lord of the Rings …)

Facial animation techniques

- Key-framing
- Motion capture
- Physically-based techniques

Keyframe Animation

- Keyframe specification
- Free form deformation



Keyframe Animation

- Problems with FFD
  - Too much freedom: nothing constraints the face to a space of probable expressions
  - Only permit limited deformations, those that correspond to changes in facial expression

Keyframe Animation

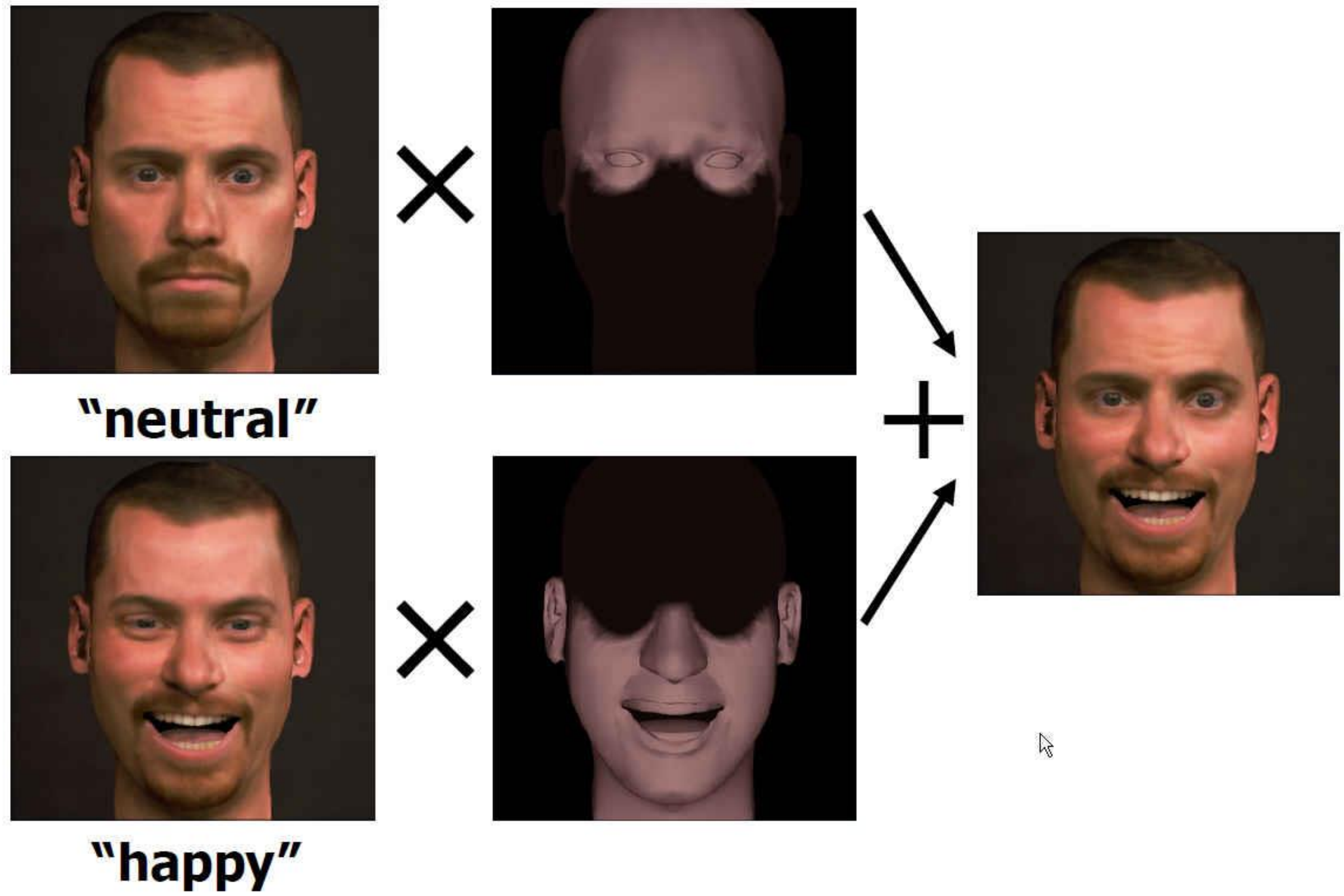
- Blend shapes
  - Manually build a set of facial expressions (blend shapes)
  - Express facial expressions as a combination of blend shapes

Keyframe Animation

- Blend shapes

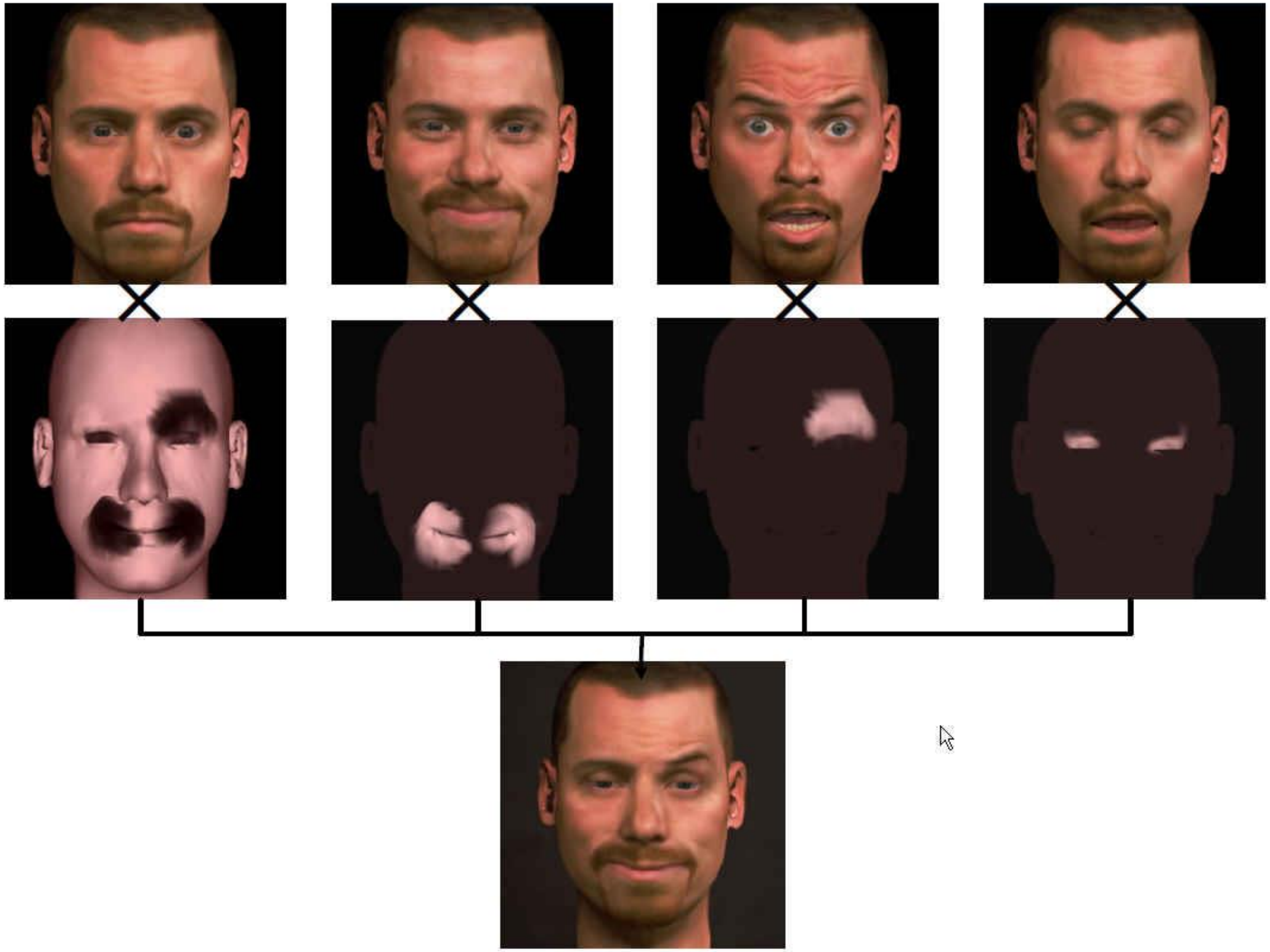
Keyframe Animation

- Blend shapes

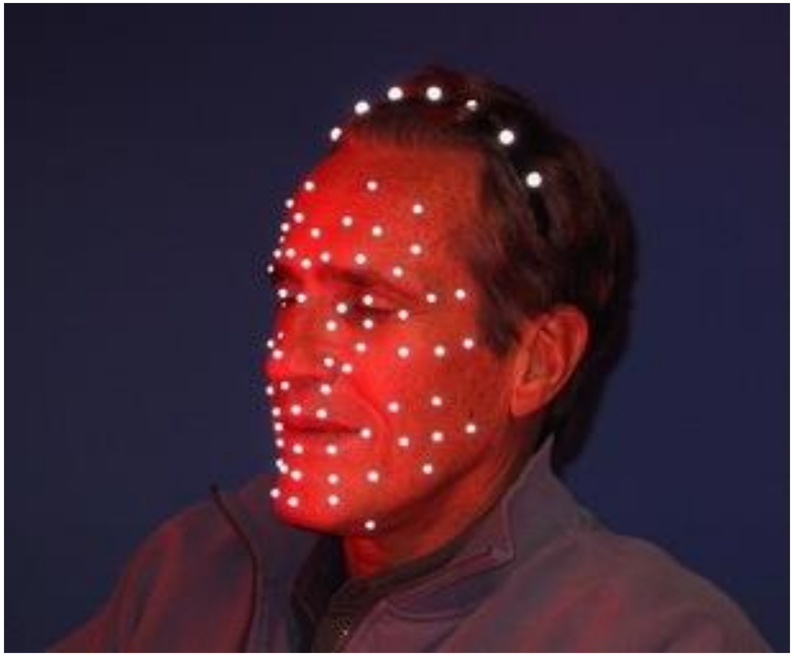


Keyframe Animation

- Blend shapes

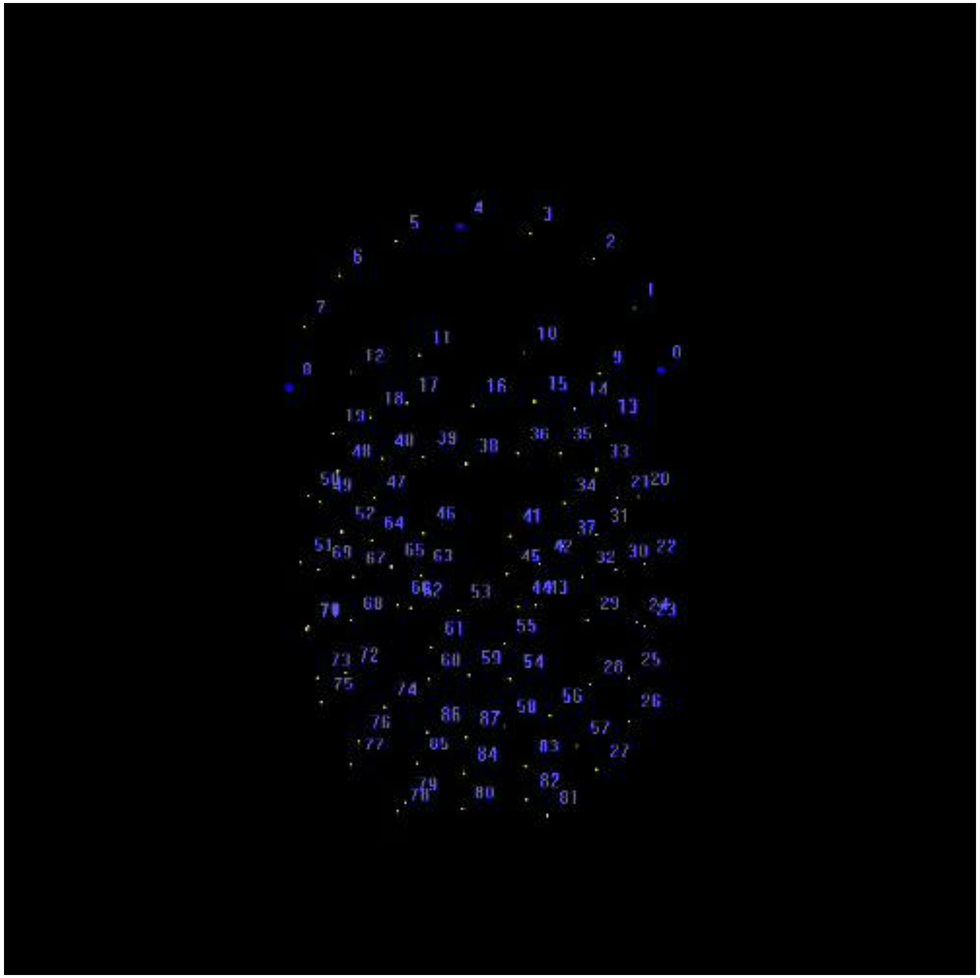


Motion-capture Animation



Motion-capture Animation

- Tracked motion

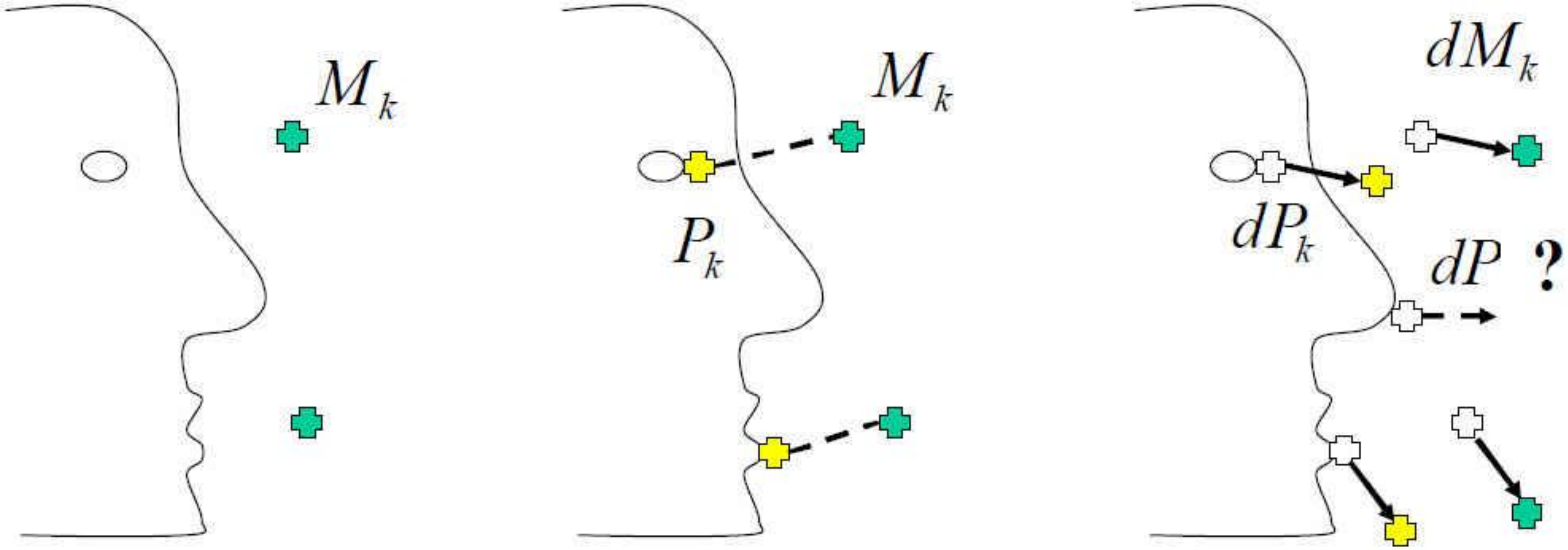


Motion-capture Animation

- Mapping motion onto face model
  - Associate markers to points on the model
  - Interpolate motion over the whole surface of the face

Motion-capture Animation

- Mapping motion onto face model



Facial Animation

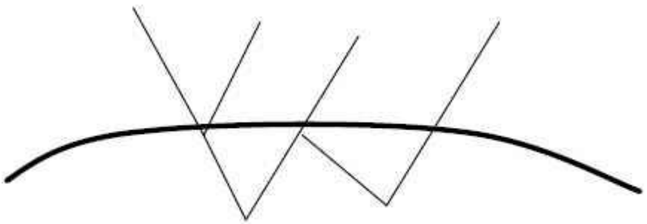
- Practical systems
  - Most practical animation systems combine multiple techniques; e.g.
    - Motion capture and physical skin model
    - Keyframing on top of motion capture

Facial Animation

- Issues in facial animation
  - Speech animation
  - Motion retargeting
  - Rendering

Facial Animation

- Rendering
  - The reflectance of the skin is not well approximated by simple reflectance model (e.g. Phong shading)
    - Fresnel reflection (shiny at grazing angles)
    - Subsurface scattering



Facial Animation

- Image based rendering
  - Capture (sample) all possible lighting conditions

- High Resolution Face Scanning for Digital Emily

http://gl.ict.usc.edu/Research/DigitalEmily/

- Live 3D Teleconferencing

http://gl.ict.usc.edu/Research/3DTeleconferencing/

光學式動作捕捉-電腦視覺

- Facial Motion Capture

DEMO

DEMO

CG Model Animation

Happy Sad Angry Fear

Emotion Detection