Computer Animation

Animation Methods

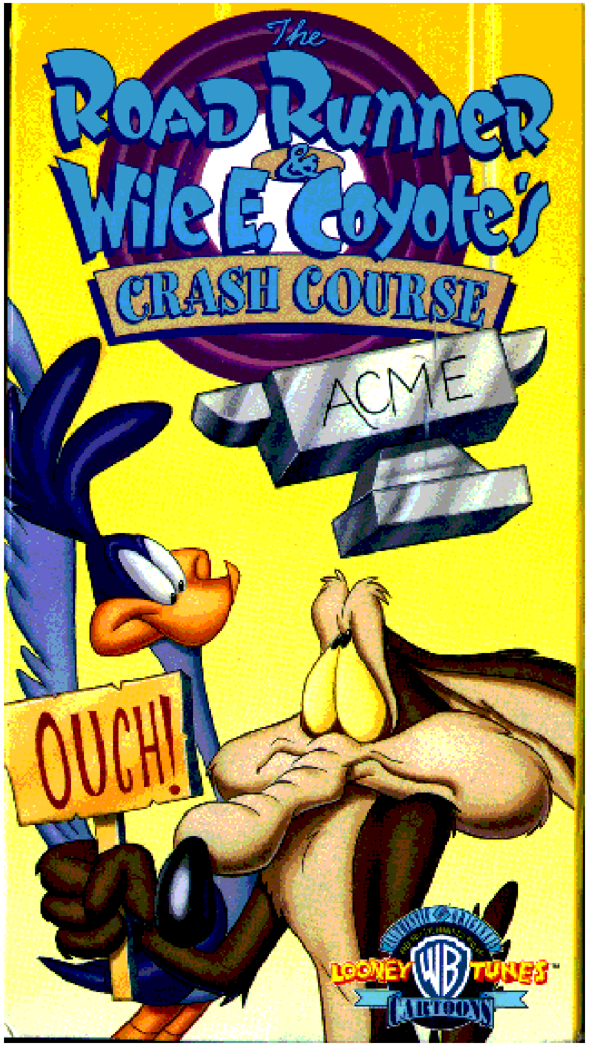
Keyframing

Interpolation

Kinematics

Inverse Kinematics

Slides courtesy of Leonard McMillan and Jovan Popovic



Administrative

Office hours

- Durand & Teller by appointment
- Ngan Thursday 4-7 in W20-575

Deadline for proposal: Friday Nov 1

Meeting with faculty & staff about proposal

- Next week
- Web page for appointment

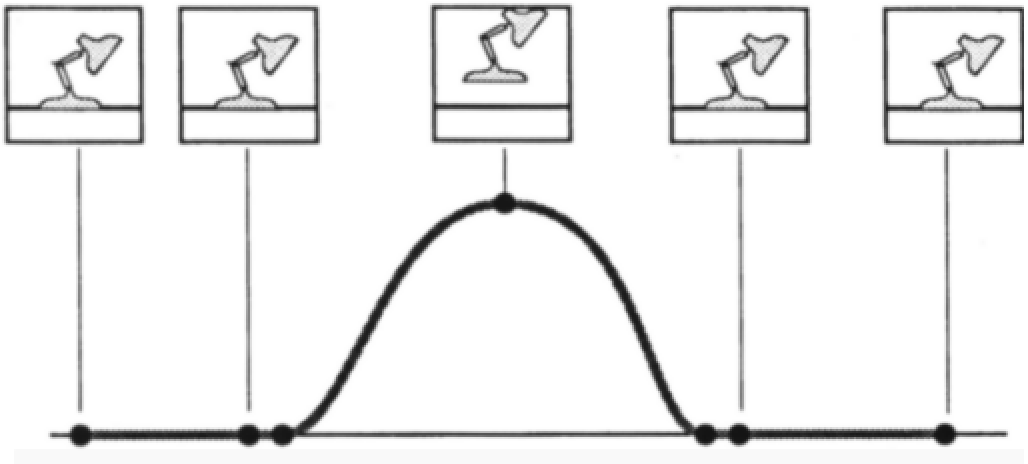
Animation

4 approaches to animation

Pros ? Cons ?

Computer-Assisted Animation

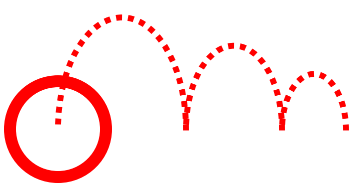
Keyframing
- automate the inbetweening
- good control
- less tedious
- creating a good animation still requires considerable skill and talent



ACM © 1987 "Principles of traditional animation applied to 3D computer animation"

Procedural animation
- describes the motion algorithmically
- express animation as a function of small number of parameteres
- Example: a clock with second, minute and hour hands
  - hands should rotate together
  - express the clock motions in terms of a "seconds" variable
  - the clock is animated by varying the seconds parameter

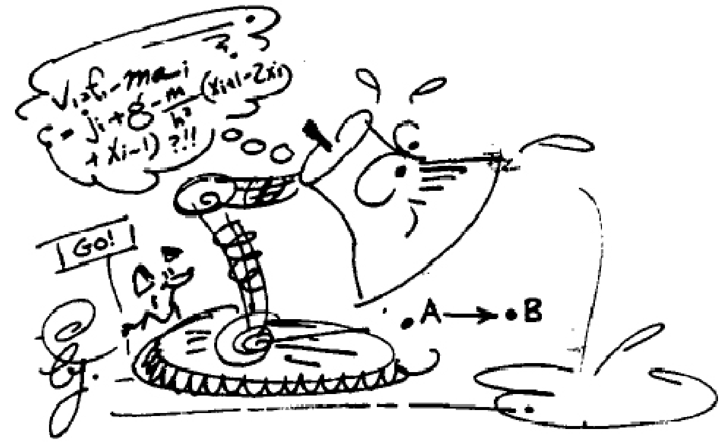
- Example 2: A bouncing ball
  - $Abs(\sin(\omega t+\theta_0))*e^{-kt}$



Computer-Assisted Animation

Physically Based Animation

- Assign physical properties to objects (masses, forces, inertial properties)
- Simulate physics by solving equations
- Realistic but difficult to control



ACM© 1988 "Spacetime Constraints"

Motion Capture

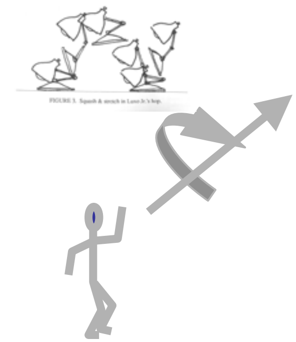
- Captures style, subtle nuances and realism
- You must observe someone do something

Overview

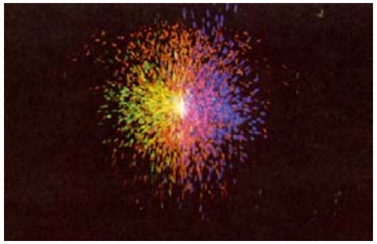
Keyframing and interpolation

Interpolation of rotations, quaternions

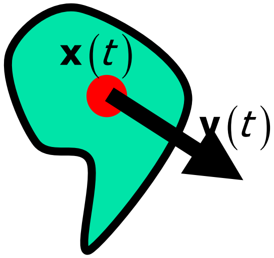
Kimematrics, articulation



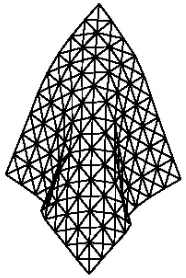
Particles



Rigid bodies



Deformable objects, clothes, fluids



Kinematics vs. Dynamics

Kinematics

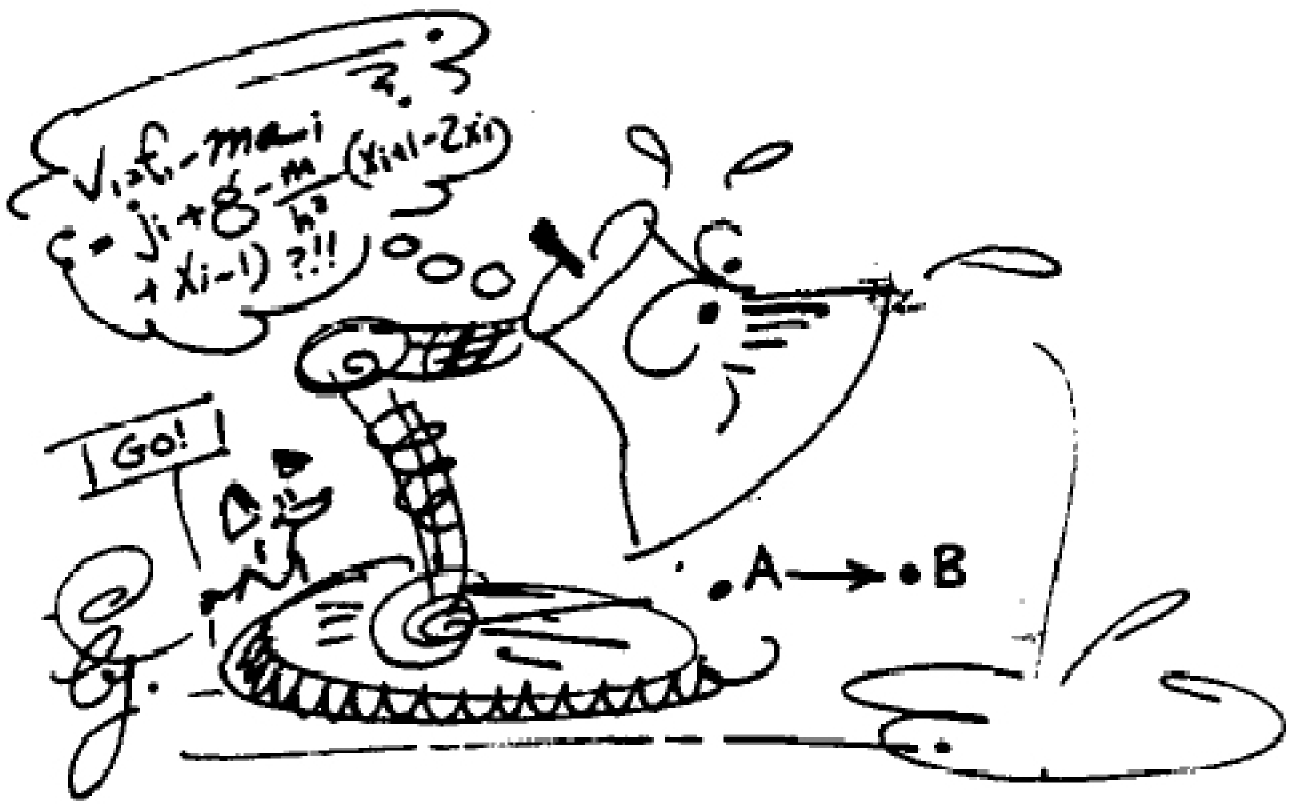
Describes the positions of the body parts as a function of the joint angles.

Dynamics

Describes the positions of the body parts as a function of the applied forces.

Now

Dynamics



ACM© 1988 "Spacetime Constraints"

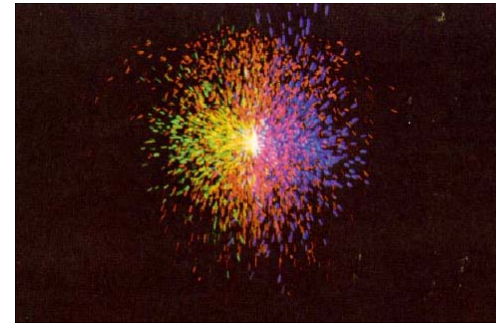
Particle

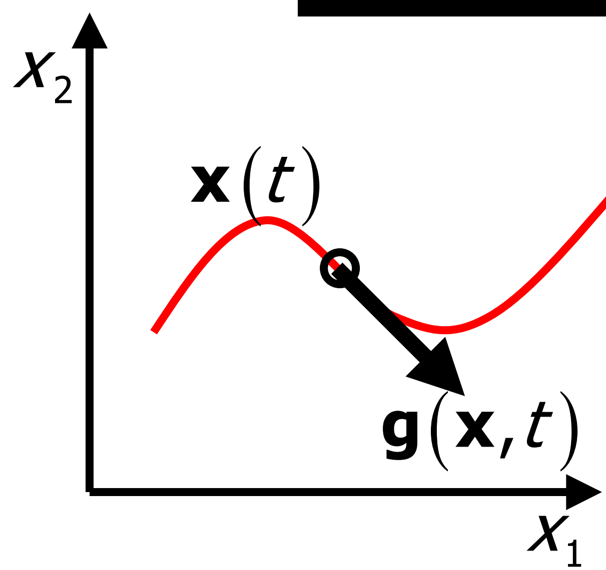
A single particle in 2-D moving in a flow field

- Position $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

- Velocity $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \mathbf{v} = \frac{d\mathbf{x}}{dt}$

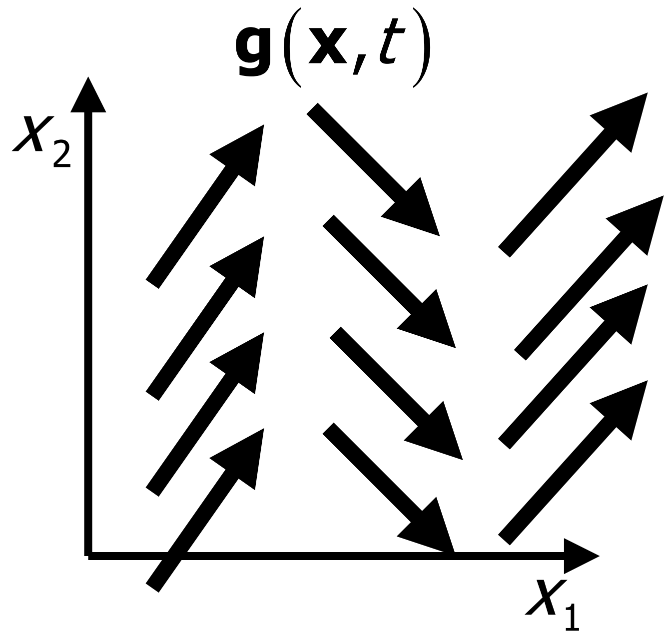
- The flow field function dictates particle velocity $\mathbf{v} = \mathbf{g}(\mathbf{x}, t)$





Vector Field

The flow field $\mathbf{g}(\mathbf{x},t)$ is a vector field that defines a vector for any particle position $\mathbf{x}$ at any time t.



How would a particle move in this vector field?

Differential Equations

The equation **v** = **g**(**x**, *t*) is a first order differential equation:

$$\frac{d\mathbf{x}}{dt} = \mathbf{g}(\mathbf{x}, t)$$

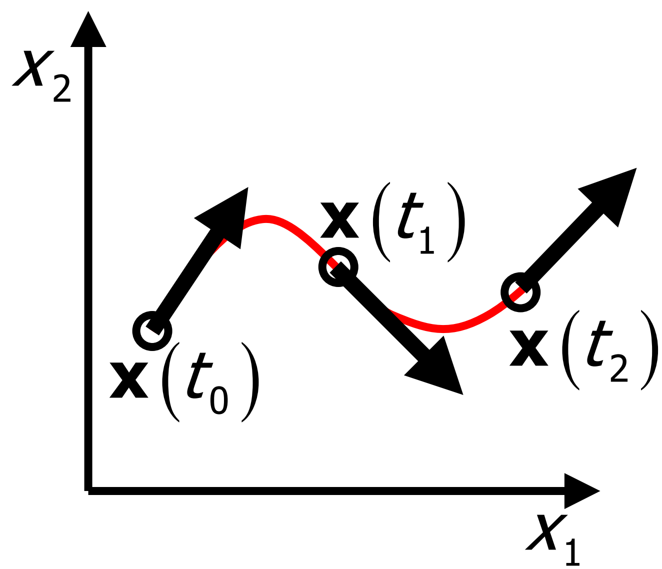
The position of the particle is computed by integrating the differential equation:

$$\mathbf{x}(t) = \mathbf{x}(t_0) + \int_{t_0}^{t} \mathbf{g}(\mathbf{x}, t)\, dt$$

For most interesting cases, this integral cannot be computed analytically.

Numeric Integration

Instead we compute the particle's position by numeric integration: starting at some initial point $\mathbf{x}(t_0)$ we step along the vector field to compute the position at each subsequent time instant. This type of a problem is called an **initial value problem**.



Euler's Method

Euler's method is the simplest solution to an initial value problem. Euler's method starts from the initial value and takes small time steps along the flow:

$$\mathbf{x}(t+\Delta t)=\mathbf{x}(t)+\Delta t\,\mathbf{g}(\mathbf{x},t)$$

Why does this work?

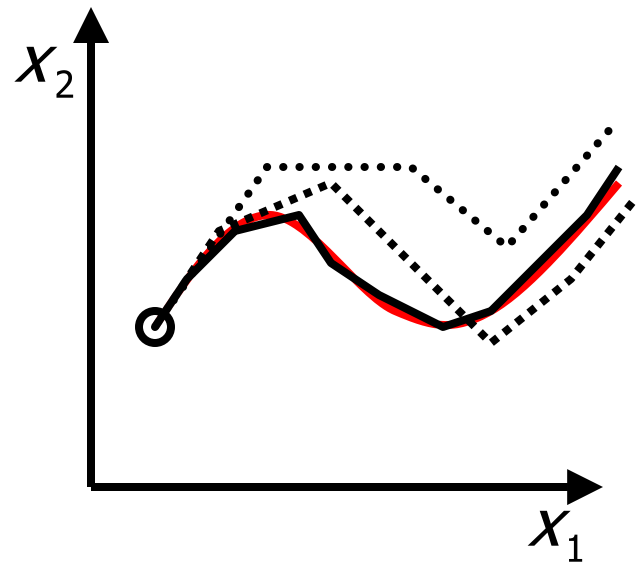
Let's look at a Taylor series expansion of function $\mathbf{x}(t)$:

$$\mathbf{x}(t+\Delta t)=\mathbf{x}(t)+\Delta t\frac{d\mathbf{x}}{dt}+\frac{\Delta t^2}{2}\frac{d^2\mathbf{x}}{dt^2}+\cdots$$

Disregarding higher-order terms and replacing the first derivative with the flow field function yields the equation for the Euler's method.

Other Methods

Euler's method is the simplest numerical method. The error is proportional to Δt^2. For most cases, the Euler's method is inaccurate and unstable requiring very small steps.



Other methods:

- Midpoint (2nd order Runge-Kutta)
- Higher order Runge-Kutta (4th order, 6th order)
- Adams
- Adaptive Stepsize

Particle in a Force Field

What is a motion of a particle in a force field?

The particle moves according to Newton's Law:

$$\frac{d^2\mathbf{x}}{dt^2} = \frac{\mathbf{f}}{m} \qquad \left(\mathbf{f} = ma\right)$$

The mass m of a particle describes the particle's inertial properties: heavier particles are easier to move than lighter particles. In general, the force field $\mathbf{f}(\mathbf{x}, \mathbf{v}, t)$ may depend on the time t and particle's position $\mathbf{x}$ and velocity $\mathbf{v}$.

Second-Order Differential Equations

Newton's Law yields an ordinary differential equation of second order:

$$\frac{d^2\mathbf{x}(t)}{dt^2} = \frac{\mathbf{f}(\mathbf{x},\mathbf{v},t)}{m}$$

A clever trick allows us to reuse the same numeric differentiation solvers for first-order differential equations. If we define a new phase space vector **y**, which consists of particle's position **x** and velocity **v**, then we can construct a new first-order differential equation whose solution will also solve the second-order differential equation.

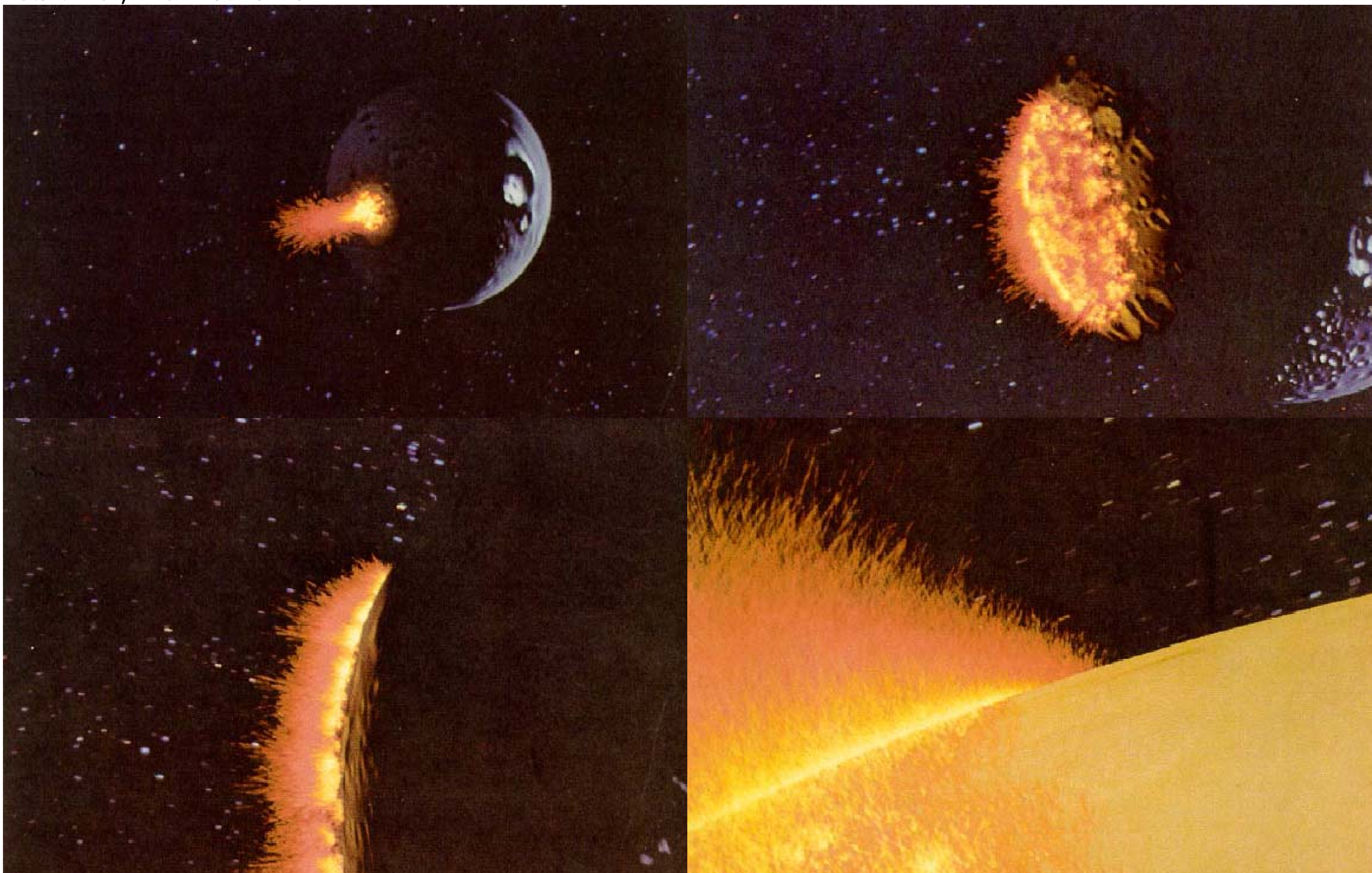
$$\mathbf{y} = \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix}, \quad \frac{d\mathbf{y}}{dt} = \begin{bmatrix} d\mathbf{x}/dt \\ d\mathbf{v}/dt \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{f}/m \end{bmatrix}$$

Particle Animation

AnimateParticles(n, $\mathbf{y}_0$, t_0, t_f)
{
    $\mathbf{y} = \mathbf{y}_0$
    $t = t_0$
    DrawParticles(n, $\mathbf{y}$)
    while(t != t_f) {
        $\mathbf{f}$ = ComputeForces($\mathbf{y}$, t)
        d$\mathbf{y}$dt = AssembleDerivative($\mathbf{y}$, $\mathbf{f}$)
        {$\mathbf{y}$, t} = ODESolverStep(6n, $\mathbf{y}$, d$\mathbf{y}$/dt)
        DrawParticles(n, $\mathbf{y}$)
    }
}

Particle Animation [Reeves et al. 1983]

Start Trek, The Wrath of Kahn



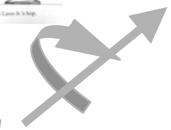
Particle Modeling [Reeves et al. 1983]

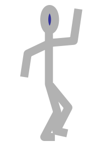
Overview

Keyframing and interpolation

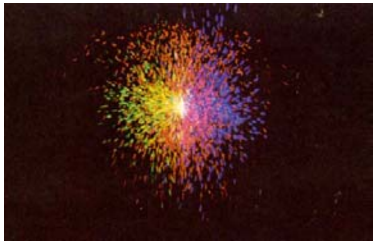
Interpolation of rotations, quaternions

Kimematrics, articulation

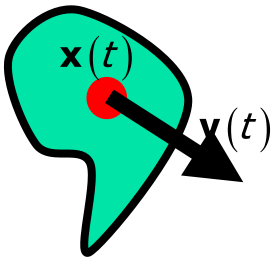




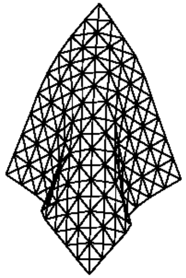
Particles



Rigid bodies

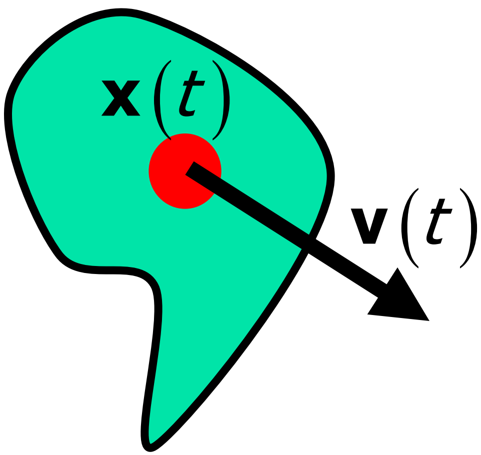


Deformable objects, clothes, fluids



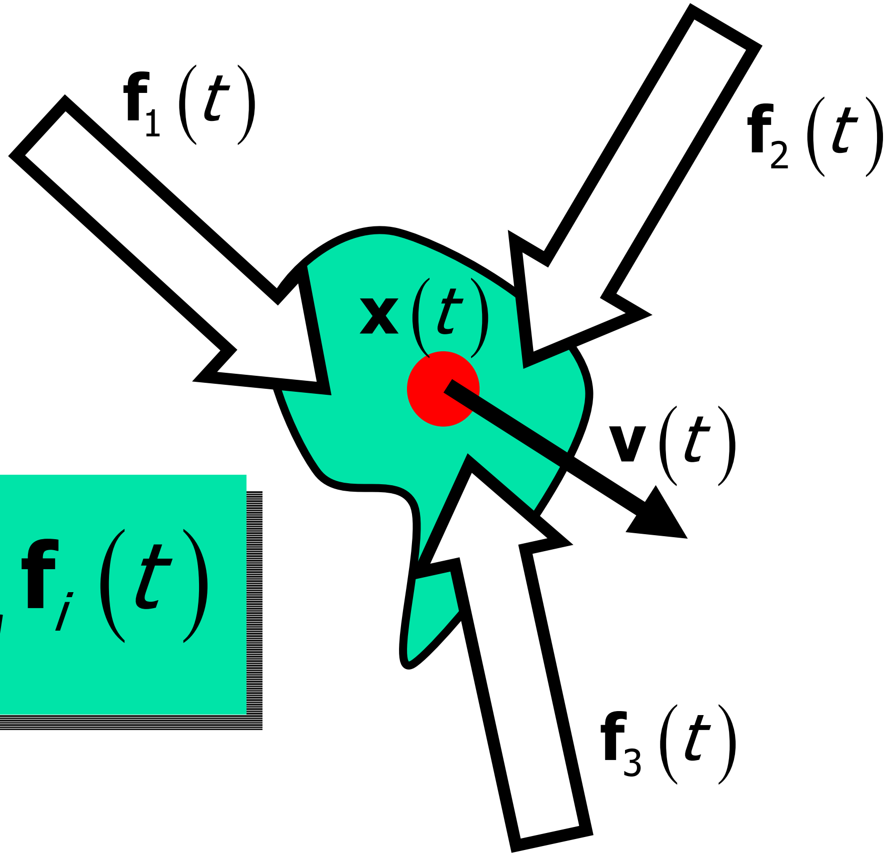
Rigid-Body Dynamics

We could compute the motion of a rigid-body by computing the motion of all constituent particles. However, a rigid body does not deform and position of few of its particles is sufficient to determine the state of the body in a phase space. We'll start with a special particle located at the body's center of mass.



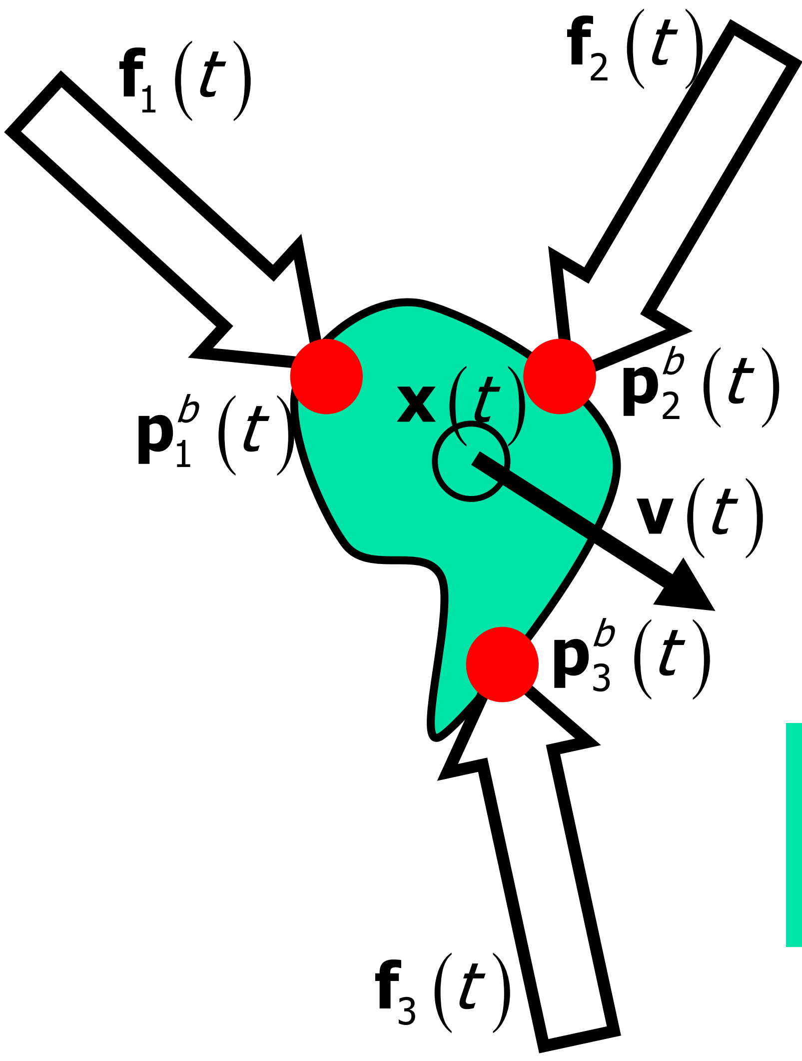
$$\mathbf{y}(t) = \begin{bmatrix} \mathbf{x}(t) \\ ? \\ \mathbf{v}(t) \\ ? \end{bmatrix}$$

Net Force



$$\mathbf{f}(t) = \sum_i \mathbf{f}_i(t)$$

Net Torque



$$\boldsymbol{\tau}(t) = \sum_i \left(\mathbf{p}_i^b - \mathbf{x}(t)\right) \times \mathbf{f}_i(t)$$

Rigid-Body Equation of Motion

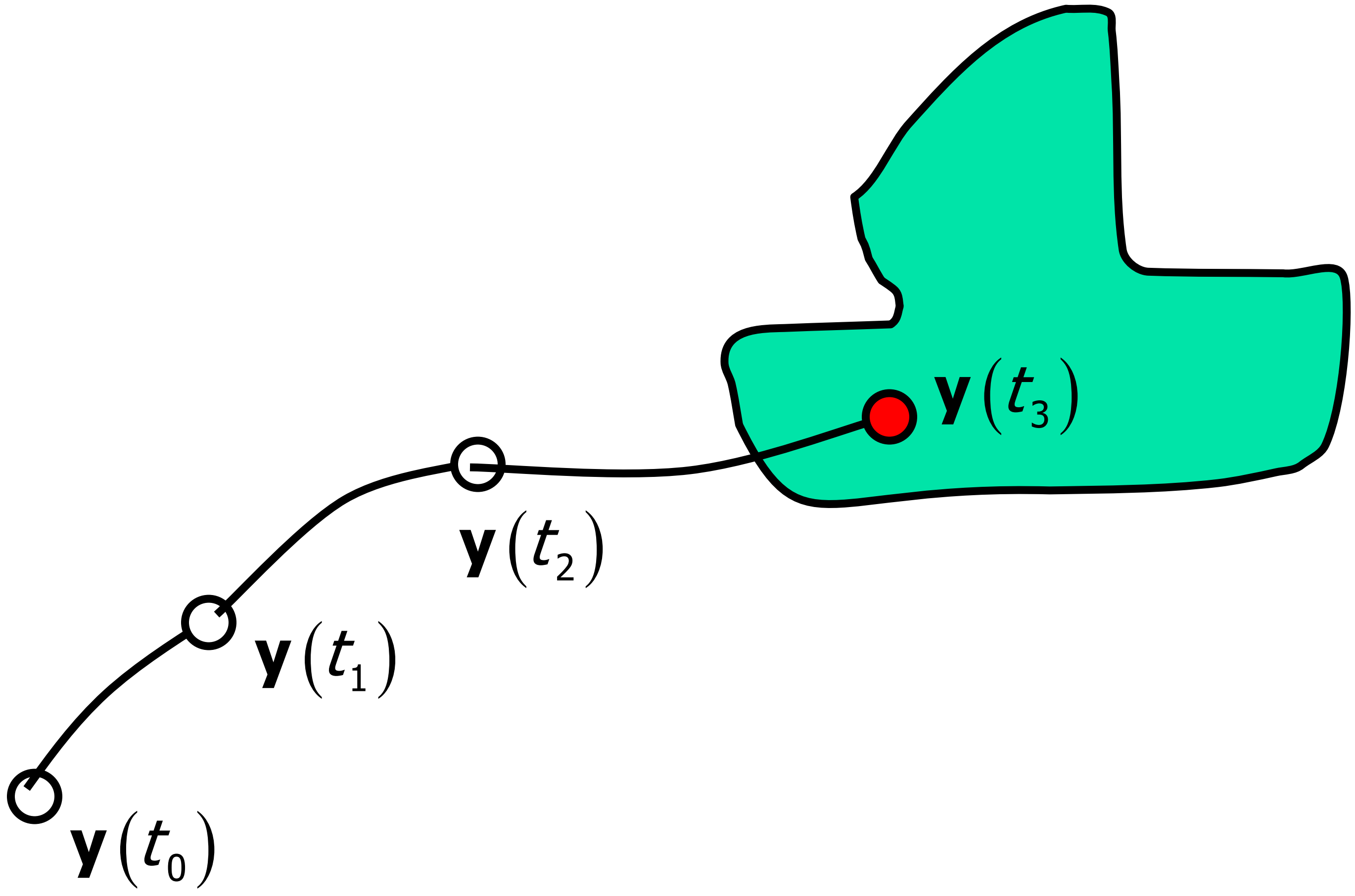
$$\frac{d}{dt}\mathbf{y}(t) = \frac{d}{dt}\begin{bmatrix} \mathbf{x}(t) \\ \mathbf{R}(t) \\ M\mathbf{v}(t) \\ \mathbf{I}(t)\boldsymbol{\omega}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{v}(t) \\ \omega(t) \times \mathbf{R}(t) \\ \mathbf{f}(t) \\ \boldsymbol{\tau}(t) \end{bmatrix}$$

$M\mathbf{v}(t) \rightarrow$ linear momentum

$\mathbf{I}(t)\boldsymbol{\omega}(t) \rightarrow$ angular momentum

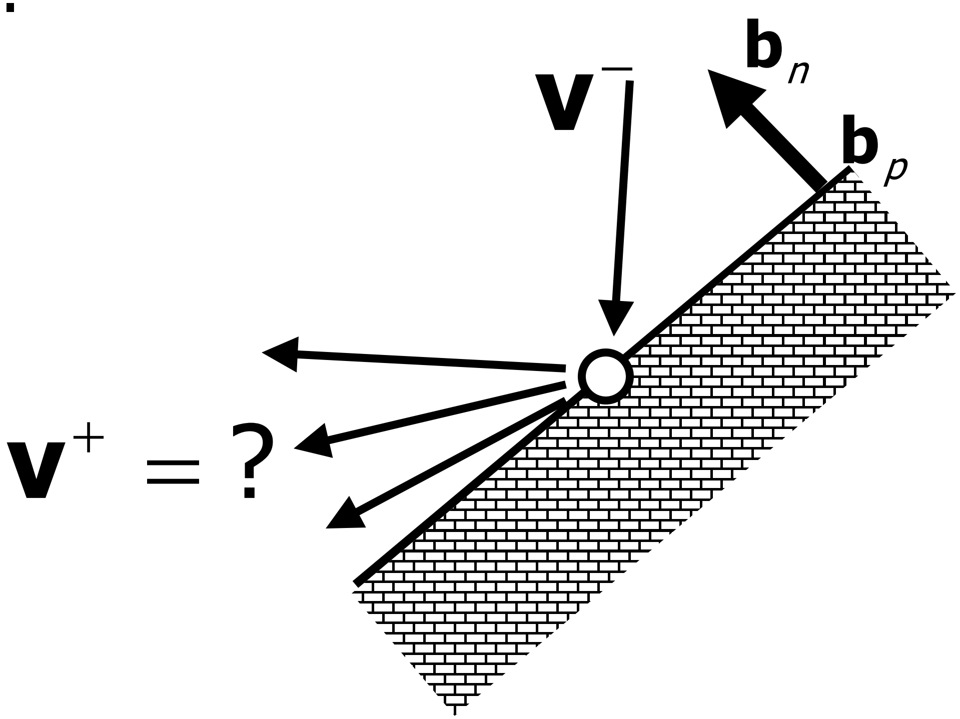
Simulations with Collisions

Simulating motions with collisions requires that we detect them (collision detection) and fix them (collision response).



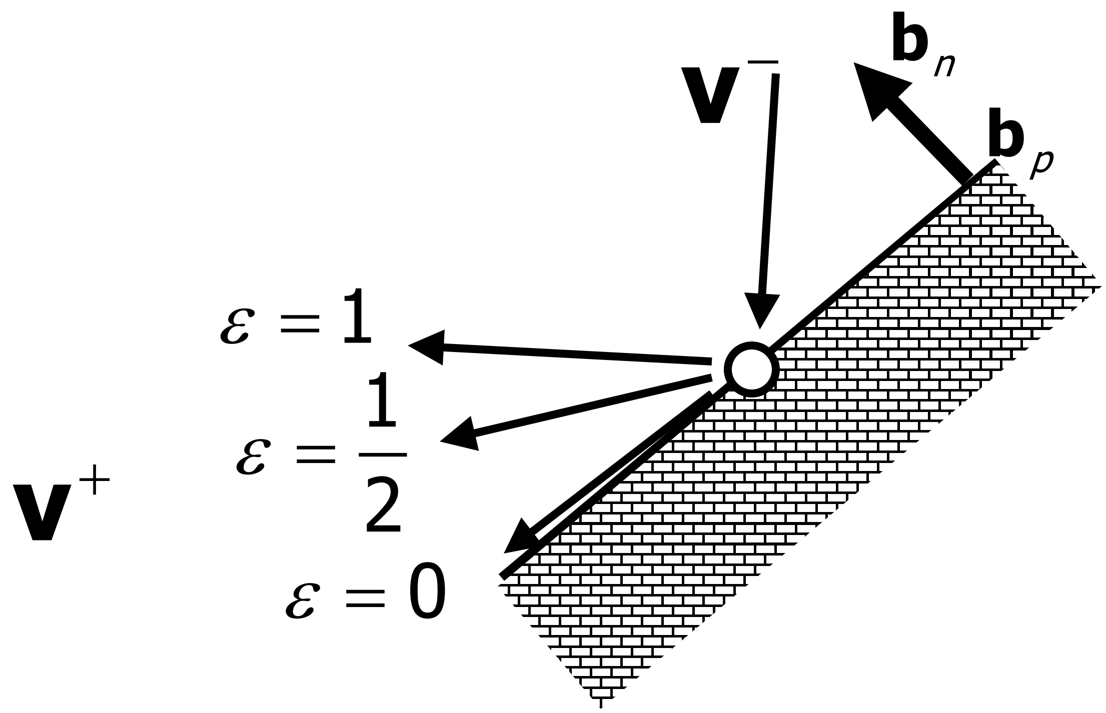
Collision Response

The mechanics of collisions are complicated and many mathematical models have been developed. We'll just look at a one simple model, which assumes that when the collision occurs the two bodes exchange collision impulse instantaneously.



Frictionless Collision Model

$$\mathbf{b}_n \cdot \mathbf{v}^+ = -\varepsilon\left(\mathbf{b}_n \cdot \mathbf{v}^-\right)$$

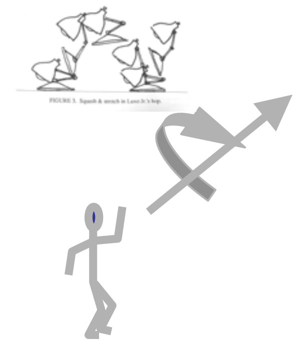


Overview

Keyframing and interpolation

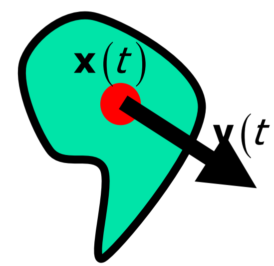
Interpolation of rotations, quaternions

Kimematrics, articulation

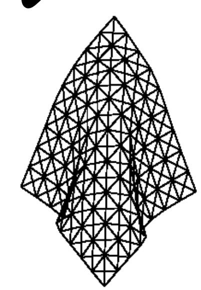


Particles

Rigid bodies



Deformable objects, clothes, fluids

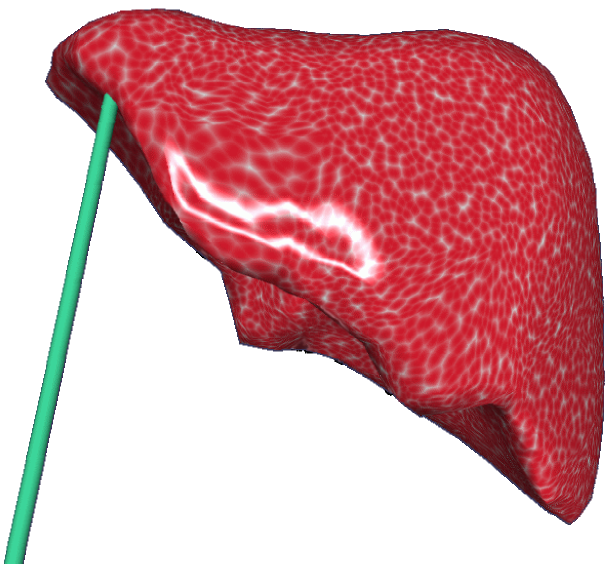


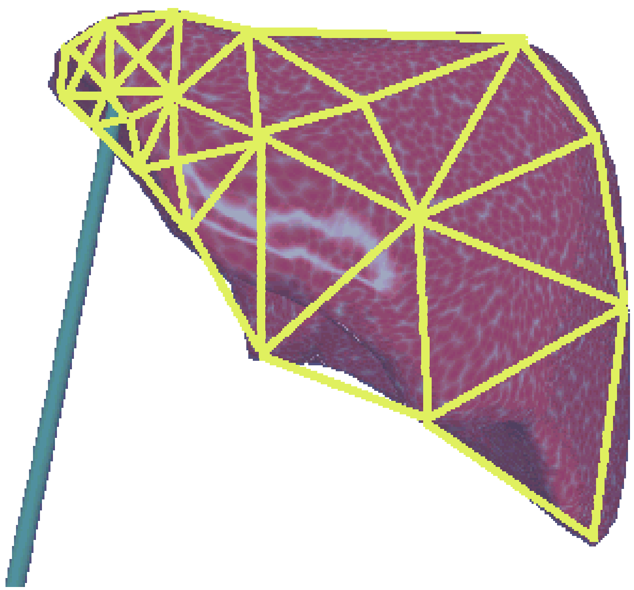
Deformable models

Shape deforms due to contact

Discretize the problem

Animation runs with smaller time steps than rendering (between 1/10,000s and 1/100s)





Images from Debunne et al. 2001

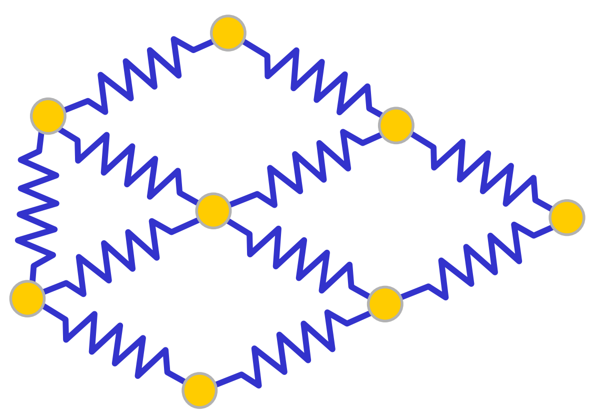
Mass-Spring system

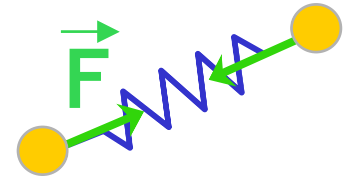
Network of masses and springs

Express forces

Integrate

Deformation of springs simulates deformation of objects





Images from Debunne et al. 2001

[Dorsey 1996]

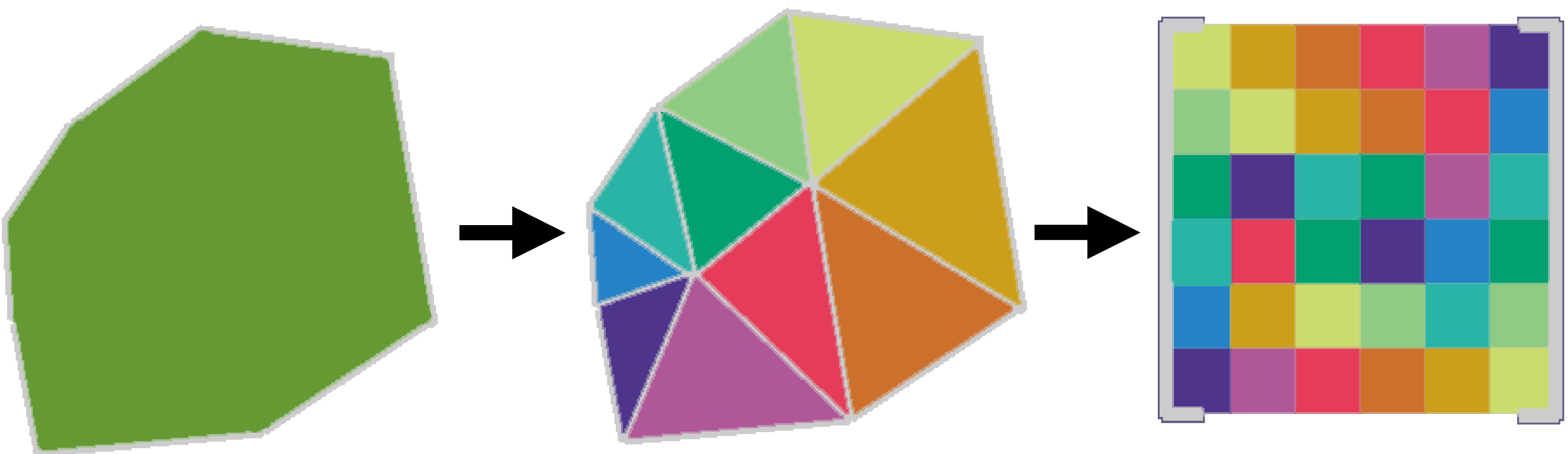
Implicit Finite Elements

Discretize the problem

Express the interrelationship

Solve a big system

More principled than mass-spring



Slide from Debunne et al. 2001

Explicit Finite Elements

Discretize the problem

Solve locally

Simpler but less stable than implicit

Object

Finite Elements

Independent matricial systems

Slide from Debunne et al. 2001

Formally: Finite Elements

We are trying to solve a continuous problem

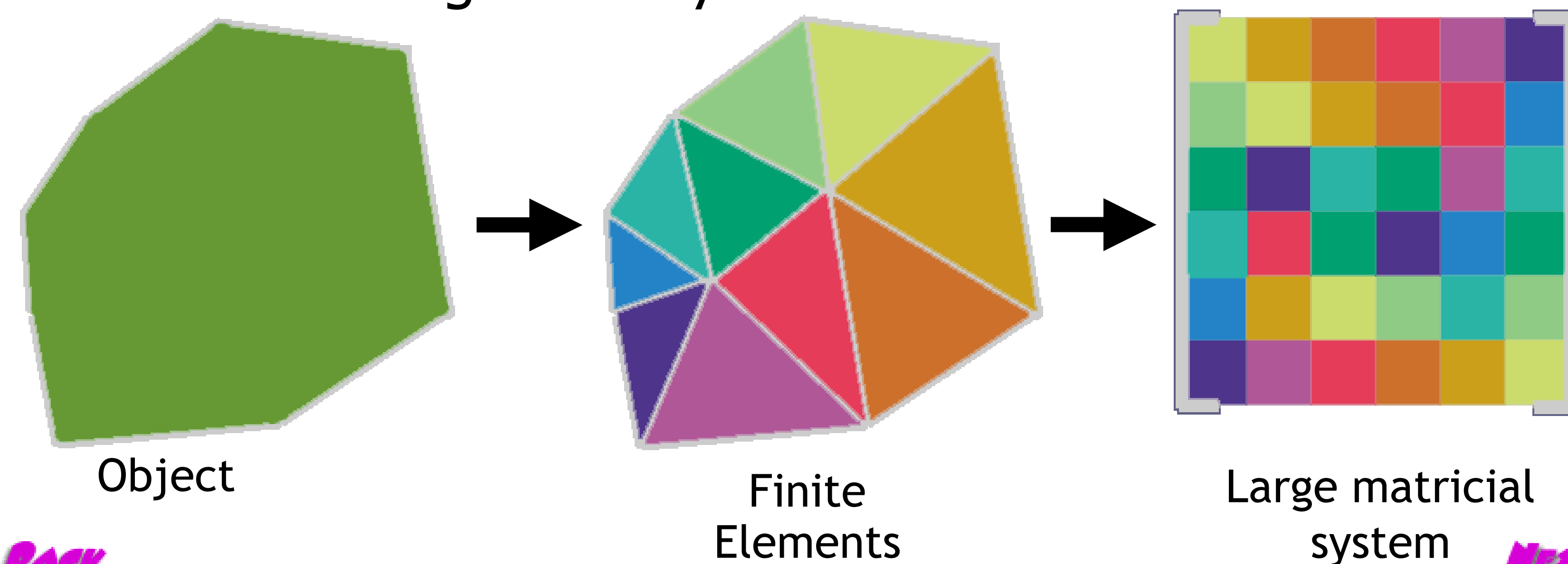
- Deformation of all points of the object
- Infinite space of functions

We project to a finite set of basis functions

- E.g. piecewise linear, piecewise constant

We project the equations governing the problem

This results in a big linear system



Cloth animation

Discretize cloth

Write physical equations

Integrate

Collision detection

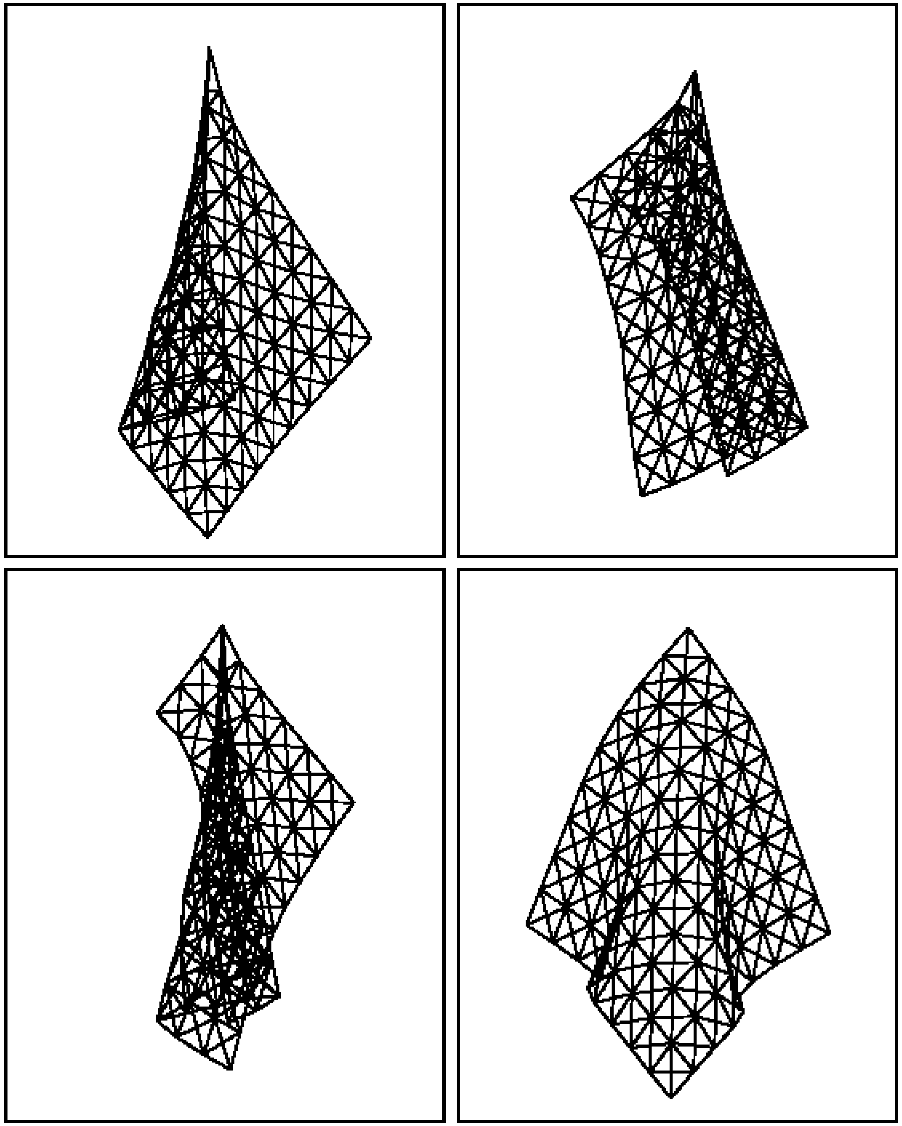


Image from Meyer et al. 2001

Fluid simulation

Discretize volume of fluid

- Exchanges and velocity at voxel boundary

Write Navier Stokes equations

- Incompressible, etc.

Numerical integration

- Finite elements, finite differences

Challenges:

- Robust integration, stability
- Speed
- Realistic surface

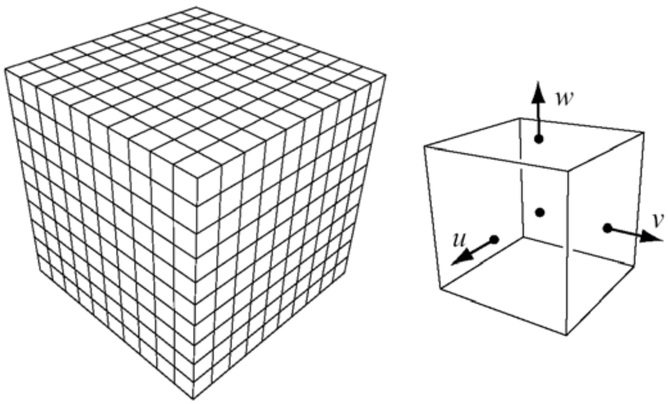


Figure from Fedkiw et al. 2001

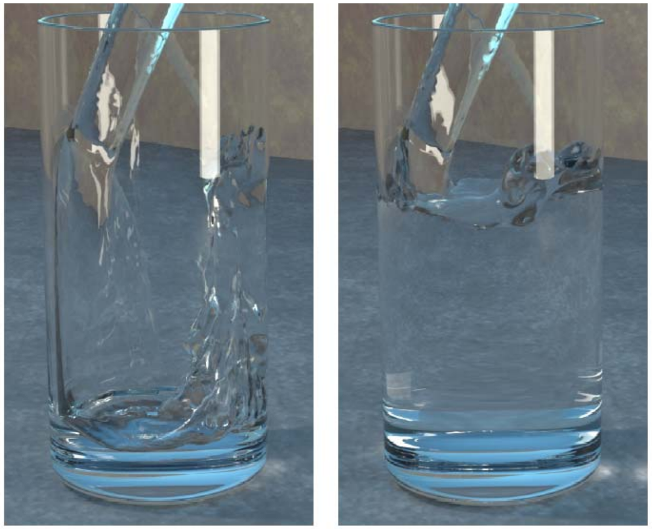


Figure 1: **Water being poured into a glass (55x120x55 grid cells).**

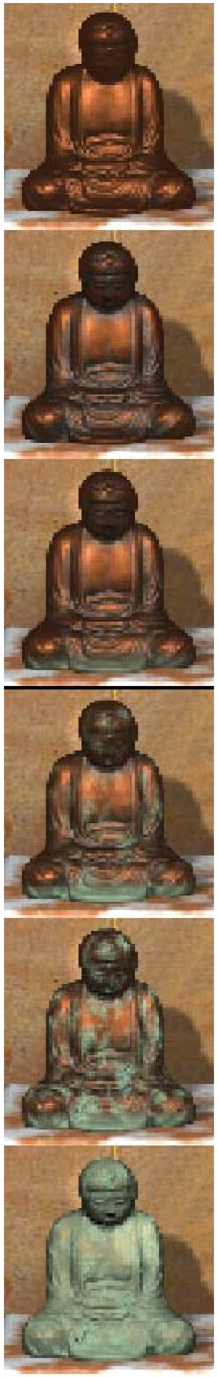
Figure from Enright et al. 2002

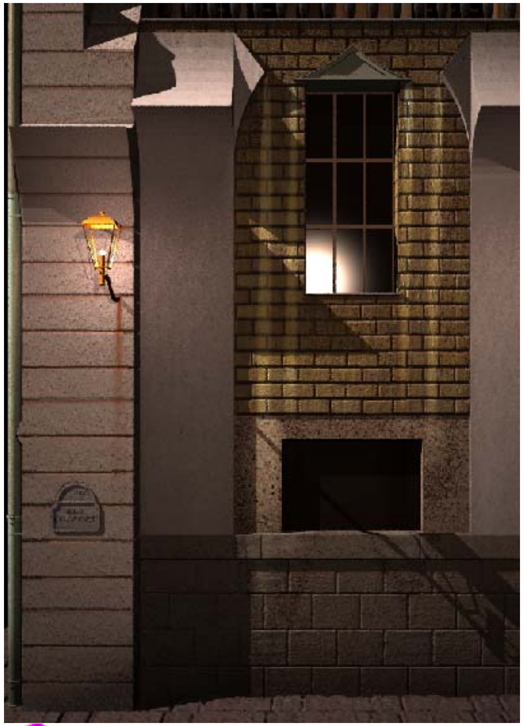
Other physical animation

Aging of materials

- Metallic patina, rust
- Water flow
- Stone aging

[Dorsey 1996-1999]





How do they animate movies?

Keyframing mostly

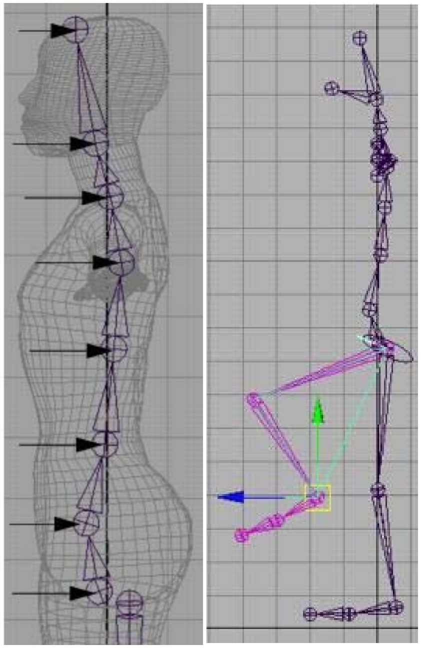
Articulated figures, inverse kinematics

Skinning

- Complex deformable skin
- Muscle, skin motion

Hierarchical controls

- Smile control, eye blinking, etc.
- Keyframes for these higher-level controls



Images from the Maya tutorial

A huge time is spent building the 3D models, its skeleton and its controls

Physical simulation for secondary motion

- Hair, cloths, water
- Particle systems for “fuzzy” objects

Next time: Texture mapping