

Lecture 6 – Animation (cont'd)

CS174A

Computer Generated Animations

Last Lecture

- Overall goals
- Traditional animation
- Computer-assisted animation
- **Computer-generated animation**
 - Key framing
 - Forward kinematics
 - Inverse kinematics
 - Procedural animation
- **Physical Simulations**
- **Motion capture**

Today

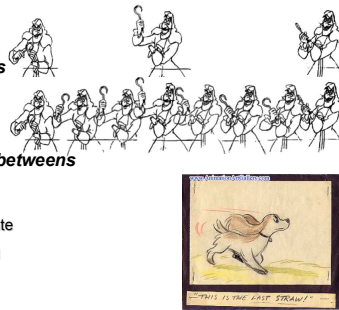
Traditional (Manual) Animation

Every frame is created individually by a human

- That's 24 frames/sec at traditional movie speeds
 - Roughly 130,000 frames for a 1.5 hr movie

A general pipeline evolved to support efficiency

- Start with a **storyboard**
 - A set of drawings outlining the animation
- Senior artists sketch important frames – **Keyframes**
 - Typically occur when motion changes
- Lower-paid artists draw the rest of the frames – **in-betweens**
- All line drawings are painted on **cels**
 - Generally composed in layers, hence the use of acetate
 - Background changes infrequently, so it can be reused
- Photograph finished **cel-stack** onto film

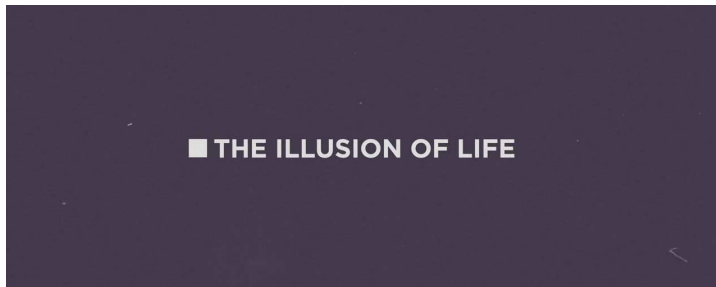


Illusion of Life - 12 Animation Principles

- 1. Squash and stretch
- 2. Anticipation
- 3. Staging
- 4. Straight ahead and pose-to-pose
- 5. Follow through and overlapping
- 6. Slow in and slow out
- 7. Arcs
- 8. Secondary action
- 9. Timing
- 10. Exaggeration
- 11. Solid drawings
- 12. Appeal

Illusion of Life

- 12 Principles of Animation
(slides courtesy of Mark Pauly)



- Cento Lodgiani, <http://vimeo.com/93206523>

The Cartoon Laws of Physics

(Originally "O'Donnell's Laws of Cartoon Motion", Esquire, 6/80)
[often quoted from "IEEE Institute", 10/94; V.18 #7 p.12]

The Cartoon Laws of Physics

- Law I:** Any body suspended in space will remain in space until made aware of its situation.
 - Daffy Duck steps off a cliff, expecting further pastureland. He loiters in midair, soliloquizing flippantly, until he chances to look down. At this point, the familiar principle of 32 feet per second per second takes over. (Exception: This does not apply to cool characters who've never studied law.)
 - (Appendum: Any species capable of flight, upon distraction of vertigo, will lose ability of flight. Conversely, any two feathers held in each hand and waved will (temporarily) give flight to any character that does so.)
- Law II:** Any body in motion will tend to remain in motion until solid matter intervenes suddenly.
 - Whether shot from a cannon or in hot pursuit on foot, cartoon characters are so absolute in their momentum that only a telephone pole or an outside boulder retards their forward motion absolutely. Sir Isaac Newton called this sudden termination of motion the stooge's surcease.
- Law III:** Any body passing through solid matter will leave a perforation conforming to its perimeter.
 - Also called the silhouette of passage, this phenomenon is the speciality of victims of directed-pressure explosions and of reckless cowards who are so eager to escape that they exit directly through the wall of a house, leaving a cookie-cutout-perfect hole. The threat of skunks or matrimony often catalyzes this reaction.
- Law IV:** The time required for an object to fall twenty stories is greater than or equal to the time it takes for whoever knocked it off the ledge to spiral down twenty flights to attempt to capture it unbroken.
 - Such an object is inevitably priceless, the attempt to capture it inevitably unsuccessful.
- Law V:** All principles of gravity are negated by fear.
 - Psychic forces are sufficient in most bodies for a shock to propel them directly away from the earth's surface. A spooky noise or an adversary's signature sound will induce motion upward, usually to the cradle of a chandelier, a treetop, or the crest of a flagpole. The feet of a character who is running or the wheels of a speeding auto need never touch the ground, especially when in flight.

The Cartoon Laws of Physics (contd.)

- Law VI:** As speed increases, objects can be in several places at once.
 - This is particularly true of tooth-and-claw fights, in which a character's head may be glimpsed emerging from the cloud of altercation at several places simultaneously. This effect is common as well among bodies that are spinning or being throttled. A 'wacky' character has the option of self-replication only at manic high speeds and may ricochet off walls to achieve the velocity required.
- Law VII:** Certain bodies can pass through solid walls painted to resemble tunnel entrances; others cannot.
 - This trompe l'oeil inconsistency has baffled generations, but at least it is known that whoever paints an entrance on a wall's surface to trick an opponent will be unable to pursue him into this theoretical space. The painter is flattened against the wall when he attempts to follow into the painting. This is ultimately a problem of art, not of science.
 - (Corollary: Portable holes work.)
- Law VIII:** Any violent rearrangement of feline matter is impermanent.
 - Cartoon cats possess even more deaths than the traditional nine lives might comfortably afford. They can be decimated, spliced, splayed, accordion-pleated, spindled, or disassembled, but they cannot be destroyed. After a few moments of blinking self pity, they reinflate, elongate, snap back, or solidify. Corollary: A cat will assume the shape of its container.
 - (Corollary 2: Cartoons cats have the uncanny ability to emit piano sounds when their teeth are transformed into piano keys after having a piano dropped on them.)
- Law IX:** Everything falls faster than an anvil.
 - Examples too numerous to mention from the Roadrunner cartoons.
- Law X:** For every vengeance there is an equal and opposite revengeance.
 - This is the one law of animated cartoon motion that also applies to the physical world at large. For that reason, we need the relief of watching it happen to a duck instead.

Outline

- Physical Simulations
 - Particle Dynamics
 - Mass-Spring Systems:
 - Examples of Mass-Spring Systems
 - Fundamentals of Mass-Spring Systems
 - Applications
 - Heating and Melting Deformable Models – Mass-springs Model
 - Liquids – Particle Models
 - Cloth – Viscoelasticity – Mass-springs Model
 - Implicit Euler
 - Human Motion Modeling
 - Rigid Body Dynamics
 - Motion Capture – Statistical Analysis – Numerical Tensor Analysis

Computer Generated Animations

Physical Simulations

- We usually want realistic looking motion
 - People are extremely experienced at observing body language
 - They pick up on unnatural human motion instantly
- Some of the methods we've discussed can achieve realism
 - If our animator makes good enough key frames
 - Or we write good enough procedural scripts
 - Or we strap a bunch of sensors on an actor
- But there's another good alternative
 - Why not just **simulate the relevant physical laws** ?
 - Then we'll know that the motion is natural
 - And we'll still have decent control over it

Physical Simulation with Newton's Law of Motion

- Direct physical simulation (e.g., with Newton's laws of motion)
 - Specify positions, masses, forces,...
 - Apply relevant laws to compute accelerations, velocities, positions as a function through time
 - We can express the relevant laws as differential equations

$$\mathbf{f} = m\mathbf{a} \rightarrow \frac{d^2\mathbf{x}}{dt^2} = \frac{\mathbf{f}}{m} \text{ or } \ddot{\mathbf{x}} = \frac{\mathbf{f}}{m}$$

- And in general we must solve them numerically

Basic Particles

- Properties
 - mass
 - Position, velocity, acceleration
 - color
 - temperature
 - age
- Differential equations govern these properties
- Collisions and other constraints directly modify position and/or velocity

Dynamics

Basic governing equation

- Newton's Laws of Physics

$$\mathbf{f} = m\mathbf{a} \rightarrow \frac{d^2\mathbf{x}}{dt^2} = \frac{\mathbf{f}}{m} \text{ or } \ddot{\mathbf{x}} = \frac{\mathbf{f}}{m}$$

- And in general we must solve them numerically (discretize time)

\mathbf{f} is a sum of a number of forces due to

- Gravity: constant downward force proportional to mass
- Simple drag (damping force) : force proportional to negative velocity
- Particle interactions: particles mutually attract and/or repel
- Wind forces
- User interaction

Gravity

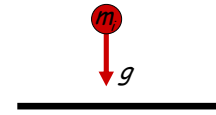
Select a "down" direction

- Here, we'll assume that the y-axis points up

Force due to gravity is simply

$$\mathbf{f}_i = \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix}$$

- g : gravitational constant
 - $\approx 9.78 \text{ m/sec}^2$ on Earth
- m : mass=1



Damping Force

- Behaves like viscous drag on all motion

$$\mathbf{f}_{\text{damping}} = -\gamma_i \dot{\mathbf{x}}_i$$



- γ_i is the damping coefficient

Discrete Fluid Model

The total force on a particle, i , due to all other particles

$$\mathbf{g}_i(t) = \sum_{j \neq i} \mathbf{g}_{ij}(t) \quad \begin{matrix} \text{attraction term} & \text{repulsion term} \end{matrix}$$

$$\mathbf{g}_{ij}(t) = m_i m_j (\mathbf{x}_i - \mathbf{x}_j) \left(-\frac{\alpha}{(d_{ij} + \zeta)^a} + \frac{\beta}{(d_{ij})^b} \right)$$

$a=2$ and $b=4$

α and β determine the strength of the attraction and repulsion forces

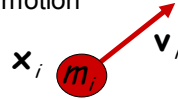
$$d_{ij} = \|\mathbf{x}_j - \mathbf{x}_i\|$$

ζ minimum required separation between particles

Particle Dynamics

Set of particles modeled as point masses in motion

- m_i : mass of particle i
- \mathbf{x}_i : position of particle i
- \mathbf{v}_i : velocity of particle i



Compute positions from Newton's second law

$$\mathbf{f}_i(t) = m_i \mathbf{a}_i(t)$$

\mathbf{f}_i : sum of all forces acting on particle

$$\mathbf{a}_i^t = \frac{\mathbf{f}_{i,total}^t}{m_i}$$

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \Delta t \mathbf{a}_i^t$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \Delta t \mathbf{v}_i^{t+1}$$

Translate by $\Delta t \mathbf{v}_i^{t+1}$

Outline

- Particle Dynamics
- **Mass-Spring Systems:**
 - Examples of Mass-Spring Systems
 - Fundamentals of Mass-Spring Systems
- Applications
 - Heating and Melting Deformable Models – Mass-springs Model
 - Liquids – Particle Models
 - Cloth – Viscoelasticity – Mass-springs Model
 - Implicit Euler
- Human Motion Modeling
 - Rigid Body Dynamics
 - Motion Capture – Statistical Analysis – Numerical Tensor Analysis

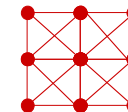
Deformable Solids: Mass-Spring-Damper Systems

Useful for building deformable models

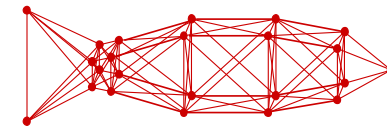
1-dimensional:



2-dimensional:



3-dimensional:

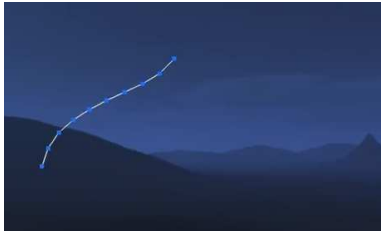


Deformable Models

Continuum mechanics

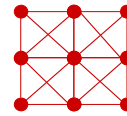
- Deformable solid models
 - Cloth
 - Rubber
 - Soft tissues (muscle, skin, hair, ...)
- Fluid models
 - Water (oceans, puddles, rain, ...)
- Gas-like models
 - Steam, smoke, fire, ...

Elastic Rope



<https://youtu.be/Co8enp8CH34>

Physics-Based Cloth Models

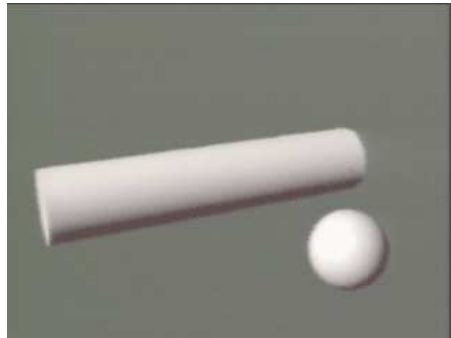


Earliest dynamic cloth (1986)



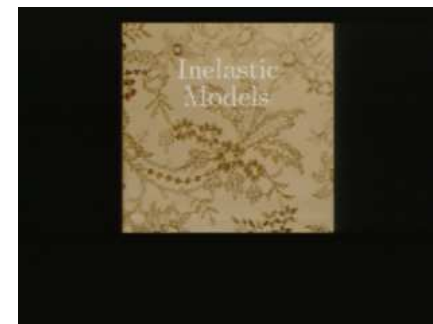
Flying Carpet

Gravity and collision forces (1987)

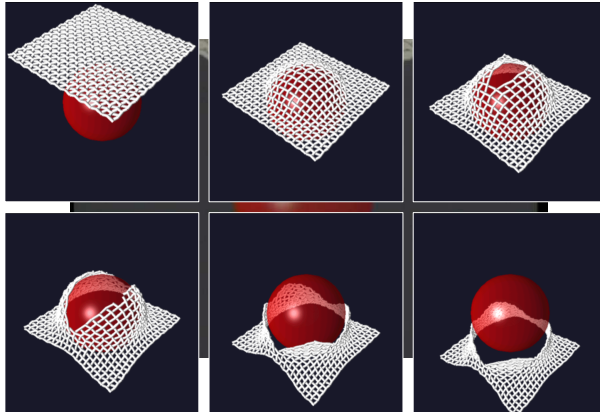


Curtain

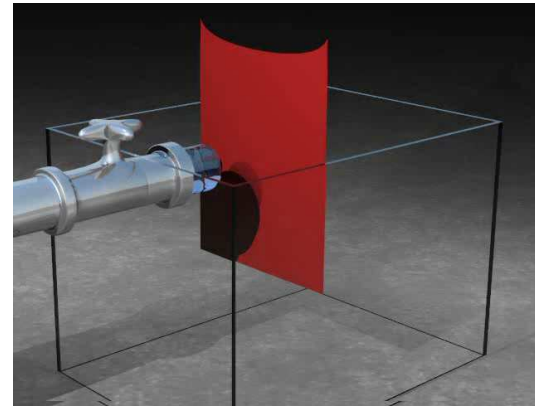
(1987)



Tearing Mesh



Cloth-Fluid Interaction

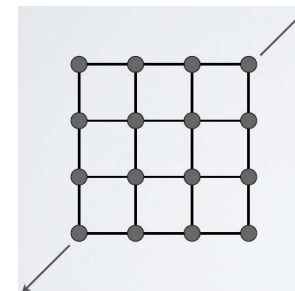


Other Uses of Mass-Spring Systems: Cloth Simulation



2D Mass-Spring Structures

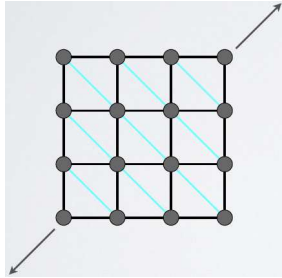
- This structure will not resist shearing
- This structure will not resist out-of-plane bending either...



- Slide courtesy of James O'Brian

2D Mass-Spring Structures

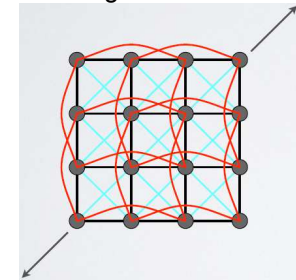
- This structure will resist shearing but has anisotropic bias
- This structure still will not resist out-of-plane bending



- Slide courtesy of James O'Brian

2D Mass-Spring Structures



- This structure will resist shearing
- Less bias
- Interference between spring sets
- This structure will resist out-of-plane bending
- Interference between spring sets



Outline

- Physical Simulations
- Particle Dynamics
 - **Mass-Spring Systems:**
 - Examples of Mass-Spring Systems
 - **Fundamentals of Mass-Spring Systems**
 - Applications
 - Heating and Melting Deformable Models – Mass-springs Model
 - Liquids – Particle Models
 - Cloth – Viscoelasticity – Mass-springs Model
 - Implicit Euler
 - Human Motion Modeling
 - Rigid Body Dynamics
 - Motion Capture – Statistical Analysis – Numerical Tensor Analysis

Data Primitives

- **Node** $i, i=1, \dots, N$ 
 - Lumped Mass: m_i
 - Position: $\mathbf{x}_i(t) = [x_i(t), y_i(t), z_i(t)]^T$
 - Velocity: $\mathbf{v}_i(t) = d\mathbf{x}_i(t)/dt$
 - Acceleration $\mathbf{a}_i(t) = d^2\mathbf{x}_i(t)/dt^2$
 - Net nodal force: $\mathbf{f}_i^{net}(t)$
 - **Spring** 
 - Connects node j to node i
 - Natural length l_{ij}
 - Stiffness k_{ij}
- ```

typedef struct node{
 float mass;
 vector position;
 vector velocity;
 vector force;
} node;

typedef struct spring{
 node *n1;
 node *n2;
 double rest_length;
 double spring_constant;
} spring;

```



## Total Nodal Forces

- External forces: gravity, interactive forces
- Internal forces: spring forces
- Damping Forces

## System Dynamics / Total Force Computation

1. For each nodal mass sum up all the forces:

$$\mathbf{F}_{i,\text{total}} = -\gamma_i \dot{\mathbf{x}}_i + \mathbf{g}_i + \mathbf{f}_i$$

- $\gamma_i$  is damping coefficient
- $\mathbf{g}_i$  total internal force on the node  $i$  due to neighboring nodes connected by springs
- $\mathbf{f}_i$  is the external force at node  $i$  (ie., gravity)

2. Compute the acceleration, velocity and position from Newton's 2<sup>nd</sup> Law of Dynamics

$$\mathbf{F}_{i,\text{total}} = m_i \ddot{\mathbf{x}}_i$$

## Damping Force

- Behaves like viscous drag on all motion

$$\mathbf{f}_{\text{damping}} = -\gamma_i \dot{\mathbf{x}}_i$$



- $\gamma_i$  is the damping coefficient

## External Forces

- Gravitational force

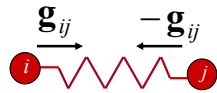


$$\mathbf{f}_{\text{gravity}} = m_i \mathbf{a}$$

$$\mathbf{a} = \begin{bmatrix} 0 \\ -9.8m/s^2 \\ 0 \end{bmatrix}$$

## Simple Spring

- Ideal zero length spring
- Force pulls points together

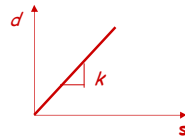


$$\mathbf{f}_i^t = \mathbf{f}_i^{t-1} + \mathbf{g}_{ij}^t$$

$$\mathbf{f}_j^t = \mathbf{f}_j^{t-1} - \mathbf{g}_{ij}^t$$

$$\mathbf{g}_{ij} = k_{ij} (\mathbf{x}_i - \mathbf{x}_j)$$

↑  
displacement  
spring constant



- Strength proportional to distance

## Non-zero length Spring Force

$$\mathbf{g}_{ij} = k_{ij} \underbrace{(\mathbf{x}_i - \mathbf{x}_j)}_{\text{displacement (magnitude and direction)}}$$

$$\mathbf{g}_{ij} = k_{ij} \left( \underbrace{\|\mathbf{x}_j - \mathbf{x}_j\| - l_{ij}}_{\text{elastic spring deformation}} \right) \underbrace{\frac{\mathbf{x}_j - \mathbf{x}_j}{\|\mathbf{x}_j - \mathbf{x}_j\|}}_{\text{direction of the force}}$$

rest length

$$\mathbf{g}_{ij} = k_{ij} e_{ij} \frac{\mathbf{d}_{ij}}{\|\mathbf{d}_{ij}\|}$$

## A Damped Spring

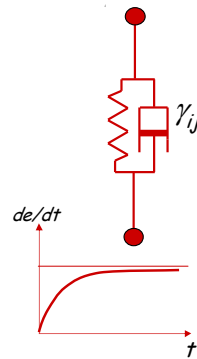
Parallel combination of spring and damper

- Known as Voigt model
- Spring damping coefficient  $\gamma_{ij}$

$$\mathbf{g}_{ij} = \left( k_{ij} e_{ij} - \gamma_{ij} \frac{de_{ij}}{dt} \right) \frac{\mathbf{d}_{ij}}{\|\mathbf{d}_{ij}\|}$$

Note:  $\frac{de_{ij}}{dt} = \mathbf{v}_{ij} \cdot \frac{\mathbf{d}_{ij}}{\|\mathbf{d}_{ij}\|}$

$$\mathbf{v}_{ij} = \mathbf{v}_j - \mathbf{v}_i$$



## Internal Spring Forces

Spring Forces:

$-\mathbf{g}_i(t)$  total force on the node  $i$  due to springs connecting it to neighboring nodes  $j \in N_i$

$$\mathbf{g}_i(t) = \sum_{j \in N_i} \mathbf{g}_{ij}$$

– the force spring  $ij$  exerts on node  $i$

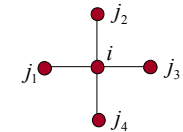
$$\mathbf{g}_{ij} = k_{ij} e_{ij} \frac{\mathbf{d}_{ij}}{\|\mathbf{d}_{ij}\|}$$

$\mathbf{d}_{ij} = \mathbf{x}_j - \mathbf{x}_i$  node distance/separation

$\|\mathbf{d}_{ij}\|$  actual spring length

$e_{ij} = \|\mathbf{g}_{ij}\| - l_{ij}$  spring deformation,  $l_{ij}$  natural spring length

$k_{ij}$  is the spring constant for the spring connecting node  $i$  and node  $j$



## System Dynamics / Total Force Computation

- Lagrange equations of motion:

$$m_i \ddot{\mathbf{x}}_i + \gamma_i \dot{\mathbf{x}}_i - \mathbf{g}_i - \mathbf{f}_i = 0,$$

- $\gamma_i$  is damping coefficient
- $\mathbf{g}_i$  total internal force on the node  $i$  due to neighboring nodes connected by springs
- $\mathbf{f}_i$  is the external force at node  $i$

Sum up all the forces:  $\mathbf{F}_{i,\text{total}} = -\gamma_i \dot{\mathbf{x}}_i + \mathbf{g}_i + \mathbf{f}_i$

Compute the acceleration, velocity and position  
From Newton's 2<sup>nd</sup> Law of Dynamics

$$\mathbf{F}_{i,\text{total}} = m_i \ddot{\mathbf{x}}_i$$

## Finite Differences

Discretization of time

- $t_i = i \Delta t = 0, \Delta t, 2\Delta t, \dots$

First finite differences of a function  $f$

- Let  $f^i = f(t_i)$ , for  $i = 0, 1, \dots$

- Forward difference:  $\frac{df(t)}{dt} \approx \frac{f^{i+1} - f^i}{\Delta t}$

- Backward difference:  $\frac{df(t)}{dt} \approx \frac{f^i - f^{i-1}}{\Delta t}$

- Central difference:  $\frac{df(t)}{dt} \approx \frac{f^{i+1} - f^{i-1}}{2\Delta t}$

## Discretization of Nodal Motion

Finite difference approximation of motion of node  $i$

- Velocity  $\mathbf{v}_i(t) = \frac{d\mathbf{x}_i(t)}{dt} \approx \frac{\mathbf{x}_i^{i+1} - \mathbf{x}_i^i}{\Delta t}$

- Acceleration  $\mathbf{a}_i(t) = \frac{d\mathbf{v}_i(t)}{dt} \approx \frac{\mathbf{v}_i^{i+1} - \mathbf{v}_i^i}{\Delta t}$

- Or,

$$\mathbf{a}_i(t) = \underbrace{\frac{\mathbf{v}_i^i - \mathbf{v}_i^{i-1}}{\Delta t}}_{\text{Backward Difference}} = \underbrace{\frac{\mathbf{x}_i^{i+1} - 2\mathbf{x}_i^i + \mathbf{x}_i^{i-1}}{(\Delta t)^2}}_{\text{Central 2<sup>nd</sup> Difference}}$$

## Integrating the Equations of Motion Through Time

The explicit Euler time-integration method

- For each node  $i$  do:

- Step 1:  $\mathbf{a}_i^t = \frac{\mathbf{F}_{i,\text{total}}^t}{m_i}$

- Step 2:  $\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \Delta t \mathbf{a}_i^t$

- Step 3:  $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \underbrace{\Delta t \mathbf{v}_i^{t+1}}_{\text{Translate by } \Delta t \mathbf{v}_i^{t+1}}$

## Equations of Motion & Code

### • Explicit Euler time integration:

$$\mathbf{F}_{i,total} = \mathbf{g}_i - \gamma_i \mathbf{v}_i + \mathbf{f}_{i,ext}$$

$$\mathbf{a}_i(t) = \frac{\mathbf{f}_{i,total}}{m_i}$$

$$\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \Delta t \mathbf{a}_i(t)$$

$$\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t + \Delta t)$$

```
void main_time_step(float dt, int num_nodes,
 node *nds, int num_springs, spring *sprs)
{vector accel, delta_vel, delta_pos;int i;
```

```
 compute_forces(num_nodes, nds, num_springs, sprs);
 for (i=0; i<num_nodes; i++){
 vscale((1 / nds[i].mass), nds[i].force, accel);
 vscale(dt, accel, delta_vel);
 vplus(delta_vel, nds[i].velocity, nds[i].velocity);
 vscale(dt, nds[i].velocity, delta_pos);
 vinc(delta_pos, nds[i].position);}}
```

### • Total force computation:

$$\mathbf{F}_{i,total} = \mathbf{g}_i - \gamma_i \mathbf{v}_i + \mathbf{f}_{i,ext}$$

- $\gamma_i$  is damping coeff
- $\mathbf{f}_i$  is the external force at node  $i$
- $\mathbf{s}_i$  total force on the node  $i$  due to springs connecting node  $i$  to neighboring nodes

```
void compute_forces(int num_nodes, node *nds,
 int num_springs, spring *sprs)
{ zeroize_forces(num_nodes, nds);
 spring_forces(num_springs, sprs);
 damping_forces(num_nodes, nds);
 external_forces(num_nodes, nds); }
```

## Force Computation & Code

### • Spring Forces:

- $\mathbf{g}_i$  total force on the node  $i$  due to springs connecting it to neighboring nodes  $j \in N_i$

$$\mathbf{g}_i(t) = \sum_{j \in N_i} \mathbf{g}_{ij}$$

- the force spring  $ij$  exerts on node  $i$

$$\mathbf{g}_{ij} = k_{ij} e_{ij} \frac{\mathbf{d}_{ij}}{\|\mathbf{d}_{ij}\|}$$

- where

$$\mathbf{d}_{ij} = \mathbf{x}_j - \mathbf{x}_i \text{ is separation of nodes}$$

$$\|\mathbf{g}_{ij}\| \text{ is actual length of spring}$$

$$e_{ij} = \|\mathbf{g}_{ij}\| - l_{ij} \text{ is deformation of spring}$$

```
void spring_forces (int num_springs, spring *sprs)
{ int i;
 for (i=0; i<num_springs; i++) spring_force(&sprs[i]);}
```

```
void spring_force (spring *s)
{ node *node1, *node2;
 double length, extension, scale_factor;
 vector direction, force;
 node1 = s->n1; node2 = s->n2;
 vminus(node2->position, node1->position, direction);
```

```
 length = vlength(direction);
```

```
 deformation = length - s->rest_length;
 scale_factor = ((deformation * s->spring_constant)/length);
 vscale(scale_factor, direction, force);
 vinc(force, node1->force);
 vdec(force, node2->force);}
```

## Force Computation (Cont.) & Code

### • Damping Forces: $\gamma \dot{\mathbf{x}}_i(t)$

```
void damping_forces (int num_nodes, node *nds)
{ vector force; int i;
 for (i=0; i<num_nodes; i++){
 vscale(DAMPING, nds[i].velocity, force);
 vdec(force, nds[i].force);}}
```

## Other Time-Integration Methods

There are more stable and/or accurate explicit methods than the Euler method

- E.g., the Runge-Kutta method

Implicit methods are stable

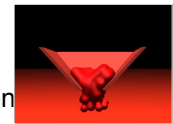
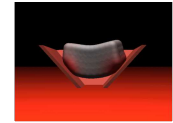
- The implicit Euler method is obtained using backward finite differences
- Implicit methods require the solution of systems of linear equations at each time step
- They are too complicated for us to cover in this introductory graphics course

## Outline

- Physical Simulations
- Particle Dynamics
  - Mass-Spring Systems:
    - Examples of Mass-Spring Systems
    - Fundamentals of Mass-Spring Systems
  - Applications**
    - Heating and Melting Deformable Models – Mass-springs Model
    - Liquids – Particle Models
    - Cloth – Viscoelasticity – Mass-springs Model
    - Implicit Euler
  - Human Motion Modeling
    - Rigid Body Dynamics
    - Motion Capture – Statistical Analysis – Numerical Tensor Analysis

## Mass-Spring Systems Applications

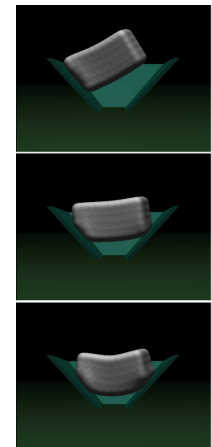
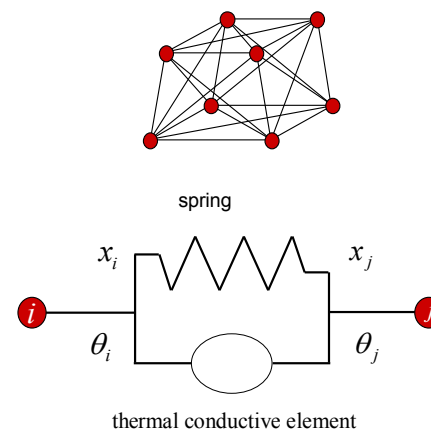
- Deformable objects capable of:
  - Heat conduction,
  - Thermoelasticity,
  - Melting and fluid-like behavior in the molten
- Cloth simulation
- Employ the Newton's Equations of Motions, but with **different types of internal forces**



## Outline

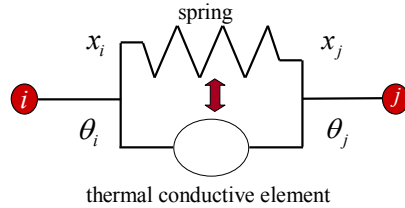
- Physical Simulations
- Particle Dynamics
  - Mass-Spring Systems:
    - Examples of Mass-Spring Systems
    - Fundamentals of Mass-Spring Systems
  - Applications**
    - Heating and Melting Deformable Models – Mass-springs Model**
    - Liquids – Particle Models
    - Cloth – Viscoelasticity – Mass-springs Model
    - Implicit Euler
  - Human Motion Modeling
    - Rigid Body Dynamics
    - Motion Capture – Statistical Analysis – Numerical Tensor Analysis

## Heating and Melting Deformable Models



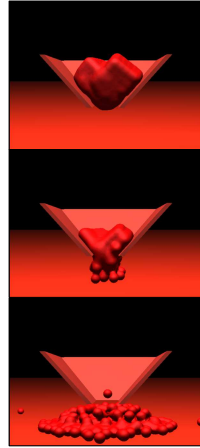
## Thermoelasticity

- Spring stiffness,  $k_{ij}$  is a function of  $\theta$



$$k_{ij} = \begin{cases} k_{ij}^0 & \text{if } \theta^a \leq \theta^s \\ k_{ij}^0 - \nu(\theta^a - \theta^s) & \text{if } \theta^s < \theta^a < \theta^m \\ 0 & \text{if } \theta^a \geq \theta^m \end{cases}$$

$$\nu = k_{ij} / (\theta^m - \theta^s)$$



## Heat/Diffusion Equation

- Nodal temperature changes are governed by the diffusion of heat in materials equation:

$$\frac{\partial}{\partial t}(\mu\sigma\theta) - \nabla \cdot (C\nabla\theta) = q$$

Heat is conducted from high temperature to low temperature .  
The rate of heat conduction per unit area is proportional to the gradient of the temperature.

The amount of heat required to raise the temperature of a material theta degrees is proportional to the mass of the sample per unit volume and the proportionality factor sigma, the specific heat of the material.

## Heat/Diffusion Equation

- Diffusion of heat in materials:

$$\frac{\partial}{\partial t}(\mu\sigma\theta) - \nabla \cdot (C\nabla\theta) = q$$

$q$  rate of heat generation/loss per unit volume

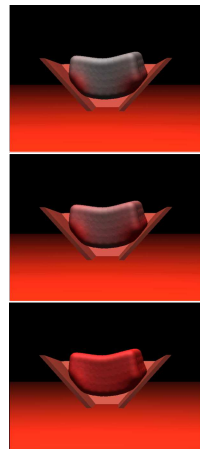
$\mu$  mass density, kg/m<sup>3</sup>

$\sigma$  specific heat, joule/(kg·Kelvin), material property

$\theta$  temperature, Kelvin

$C$  thermal conductivity matrix, material property

$$\nabla = \left[ \frac{\partial}{\partial u}, \frac{\partial}{\partial v}, \frac{\partial}{\partial w} \right]$$



## Discretize the Heat/Diffusion Equation

- Homogeneous, isotropic material:

$$\frac{\partial}{\partial t}(\mu\sigma\theta) - c\nabla^2\theta = q$$

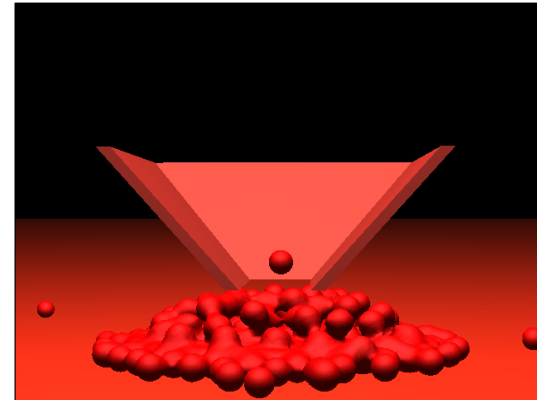
- Discretize the heat equation:

$$\mu\sigma \frac{(\theta^{t+\Delta t} - \theta^t)}{\Delta t} - c \left[ \frac{\theta^t_{u+\Delta u, v, w} - 2\theta^t_{u, v, w} + \theta^t_{u-\Delta u, v, w}}{\Delta u^2} + \frac{\theta^t_{u, v+\Delta v, w} - 2\theta^t_{u, v, w} + \theta^t_{u, v-\Delta v, w}}{\Delta v^2} + \frac{\theta^t_{u, v, w+\Delta w} - 2\theta^t_{u, v, w} + \theta^t_{u, v, w-\Delta w}}{\Delta w^2} \right] = q$$

## Update $\theta$

$$\theta_{u,v,w}^{t+\Delta t} = \theta_{u,v,w}^t + \frac{\Delta t}{\mu\sigma} c \left[ \frac{\theta_{u+\Delta u,v,w}^t - 2\theta_{u,v,w}^t + \theta_{u-\Delta u,v,w}^t}{\Delta u^2} + \frac{\theta_{u,v+\Delta v,w}^t - 2\theta_{u,v,w}^t + \theta_{u,v-\Delta v,w}^t}{\Delta v^2} + \frac{\theta_{u,v,w+\Delta w}^t - 2\theta_{u,v,w}^t + \theta_{u,v,w-\Delta w}^t}{\Delta w^2} \right] + q$$

## Thermoelasticity



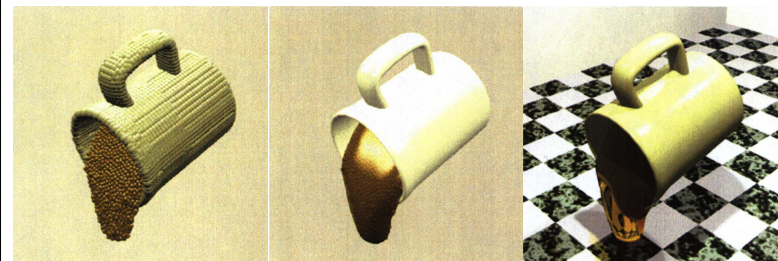
## Outline

- Physical Simulations
- Particle Dynamics
  - Mass-Spring Systems:
    - Examples of Mass-Spring Systems
    - Fundamentals of Mass-Spring Systems
  - Applications
    - Heating and Melting Deformable Models – Mass-springs Model
    - **Liquids – Particle Models**
    - Cloth – Viscoelasticity – Mass-springs Model
    - Implicit Euler
  - Human Motion Modeling
    - Rigid Body Dynamics
    - Motion Capture – Statistical Analysis – Numerical Tensor Analysis

## Liquids – Particle Models

Model long range attraction and short range repulsion forces between pairs of particles according to **Lennard-Jones potentials**.

Forces involving inverse powers of particle separation



## Discrete Fluid Model

The total force on a particle,  $i$ , due to all other particles

$$\mathbf{g}_i(t) = \sum_{j \neq i} \mathbf{g}_{ij}(t)$$

$$\mathbf{g}_{ij}(t) = m_i m_j (\mathbf{x}_i - \mathbf{x}_j) \left( \overbrace{-\frac{\alpha}{(d_{ij} + \zeta)^a}}^{\text{attraction term}} + \overbrace{\frac{\beta}{(d_{ij})^b}}^{\text{repulsion term}} \right)$$

$a=2$  and  $b=4$

$\alpha$  and  $\beta$  determine the strength of the attraction and repulsion forces

$$d_{ij} = \|\mathbf{x}_j - \mathbf{x}_i\|$$

$\zeta$  minimum required separation between particles

## Fluid Flow Simulation



P. Egbert, BYU

## Liquid Interacting with Scene Object

Liquid Interacting  
with  
Scene Object

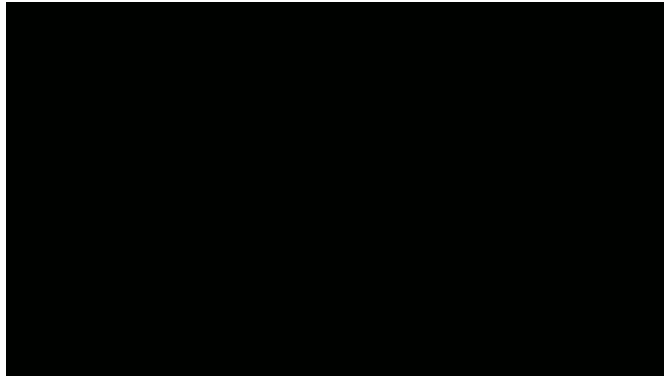
Liquid Interacting  
with  
Moving Object

## Outline

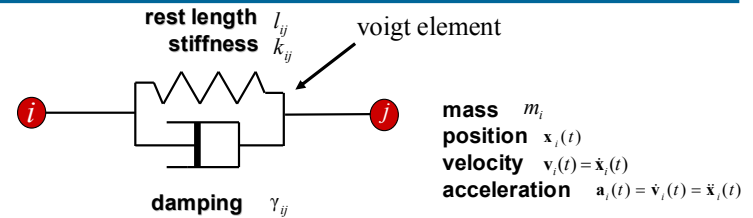
- Physical Simulations {
  - Particle Dynamics
  - Mass-Spring Systems:
    - Examples of Mass-Spring Systems
    - Fundamentals of Mass-Spring Systems
  - Applications
    - Heating and Melting Deformable Models – Mass-springs Model
    - Liquids – Particle Models
    - **Cloth – Viscoelasticity – Mass-springs Model**
    - Implicit Euler
  - Human Motion Modeling
    - Rigid Body Dynamics
    - Motion Capture – Statistical Analysis – Numerical Tensor Analysis



## Clothing -- Pixar's "Geri's Game"



## Viscoelastic Element



- Lagrange equations of motion

$$m_i \ddot{\mathbf{x}}_i + \gamma_i \dot{\mathbf{x}}_i + (k_{ij} e_{ij} + \gamma_{ij} \dot{e}_{ij}) \frac{\mathbf{d}_{ij}}{\|\mathbf{d}_{ij}\|} = \mathbf{f}_i \quad \mathbf{d}_{ij} = \mathbf{x}_j - \mathbf{x}_i$$

$$m_j \ddot{\mathbf{x}}_j + \gamma_j \dot{\mathbf{x}}_j - (k_{ij} e_{ij} + \gamma_{ij} \dot{e}_{ij}) \frac{\mathbf{d}_{ij}}{\|\mathbf{d}_{ij}\|} = \mathbf{f}_j \quad d_{ij} = \|\mathbf{d}_{ij}\|$$

$$e_{ij} = d_{ij} - l_{ij}$$

## Viscoelastic Element

$$m_i \ddot{\mathbf{x}}_i + \gamma_i \dot{\mathbf{x}}_i + c_{ij} \mathbf{r}_{ij} = \mathbf{f}_i, \quad \text{effective stiffness:}$$

$$m_j \ddot{\mathbf{x}}_j + \gamma_j \dot{\mathbf{x}}_j - c_{ij} \mathbf{r}_{ij} = \mathbf{f}_j \quad c_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \frac{k_{ij} e_{ij} + \gamma_{ij} \dot{e}_{ij}}{\|\mathbf{r}_{ij}\|}$$

In matrix form

$$\begin{bmatrix} m_i & 0 \\ 0 & m_j \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{x}}_i \\ \ddot{\mathbf{x}}_j \end{bmatrix} + \begin{bmatrix} \gamma_i & 0 \\ 0 & \gamma_j \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}}_i \\ \dot{\mathbf{x}}_j \end{bmatrix} + \begin{bmatrix} -c_{ij} & c_{ij} \\ c_{ij} & -c_{ij} \end{bmatrix} \begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_j \end{bmatrix} = \begin{bmatrix} \mathbf{f}_i \\ \mathbf{f}_j \end{bmatrix}$$

$$\mathbf{M} \ddot{\mathbf{x}} + \mathbf{G} \dot{\mathbf{x}} + \mathbf{K}(\mathbf{x}) \mathbf{x} = \mathbf{f}$$

## Other Time-Integration Methods

There are more stable and/or accurate explicit methods than the Euler method

- E.g., the Runge-Kutta method

Implicit methods are stable

- The implicit Euler method is obtained using backward finite differences
- Implicit methods require the solution of systems of linear equations at each time step

## Outline

Physical Simulations

- Particle Dynamics
- Mass-Spring Systems:
  - Examples of Mass-Spring Systems
  - Fundamentals of Mass-Spring Systems
- Applications
  - Heating and Melting Deformable Models – Mass-springs Model
  - Liquids – Particle Models
  - Cloth – Viscoelasticity – Mass-springs Model
  - **Implicit Euler**
- Human Motion Modeling
  - Rigid Body Dynamics
  - Motion Capture – Statistical Analysis – Numerical Tensor Analysis

## Implicit Numerical Solution

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{G}\dot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{f}_{total}$$

discretize time:  $t \Rightarrow 0, \Delta t, 2\Delta t, \dots, t, t + \Delta t, \dots$

Implicit Euler

$$\mathbf{x}^{t+\Delta t} = \mathbf{x}^t + \Delta t \dot{\mathbf{x}}^{t+\Delta t}$$

$$\dot{\mathbf{x}}^{t+\Delta t} = \dot{\mathbf{x}}^t + \Delta t \ddot{\mathbf{x}}^{t+\Delta t}$$

Using backward difference

$$\Rightarrow \ddot{\mathbf{x}}^{t+\Delta t} = (\dot{\mathbf{x}}^{t+\Delta t} - \dot{\mathbf{x}}^t) / \Delta t$$

Substitute in the top equation:

$$\mathbf{M} \left( \frac{\dot{\mathbf{x}}^{t+\Delta t}}{\Delta t} - \frac{\dot{\mathbf{x}}^t}{\Delta t} \right) + (\mathbf{G}^t \dot{\mathbf{x}}^{t+\Delta t}) + \mathbf{K}^t (\mathbf{x}^t + \Delta t \dot{\mathbf{x}}^{t+\Delta t}) = \mathbf{f}^t$$

Explicit Euler

$$\mathbf{x}^{t+\Delta t} = \mathbf{x}^t + \Delta t \dot{\mathbf{x}}^t$$

$$\dot{\mathbf{x}}^{t+\Delta t} = \dot{\mathbf{x}}^t + \Delta t \ddot{\mathbf{x}}^t$$

## Implicit Numerical Solution

We do not know actual values at  $t+\Delta t$ , must solve for them:

$$\left( \frac{\mathbf{M}}{\Delta t} \dot{\mathbf{x}}^{t+\Delta t} - \frac{\mathbf{M}}{\Delta t} \dot{\mathbf{x}}^t \right) + (\mathbf{G}^t \dot{\mathbf{x}}^{t+\Delta t}) + (\mathbf{K}^t \mathbf{x}^t + \Delta t \mathbf{K}^t \dot{\mathbf{x}}^{t+\Delta t}) = \mathbf{f}^t$$

$$\left( \frac{\mathbf{M}}{\Delta t} + \mathbf{G}^t + \Delta t \mathbf{K}^t \right) \dot{\mathbf{x}}^{t+\Delta t} = \mathbf{f}^t + \frac{\mathbf{M}}{\Delta t} \dot{\mathbf{x}}^t - \mathbf{K}^t \mathbf{x}^t$$

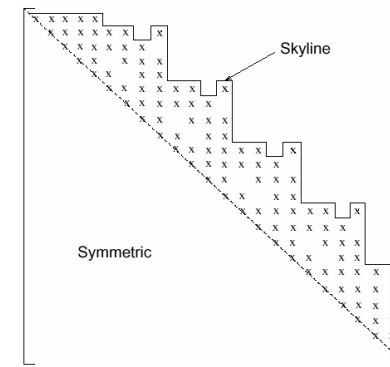
$$\underbrace{\left( \frac{\mathbf{M}}{\Delta t} + \mathbf{G}^t + \Delta t \mathbf{K}^t \right)}_{\mathbf{A}^t} \dot{\mathbf{x}}^{t+\Delta t} = \underbrace{\left( \mathbf{f}^t + \frac{\mathbf{M}}{\Delta t} \dot{\mathbf{x}}^t - \mathbf{K}^t \mathbf{x}^t \right)}_{\mathbf{b}^t}$$

$\mathbf{A}$ , effective system matrix

$\mathbf{b}$ , effective load matrix

## Skyline Storage Schemes for System Matrix A

$\mathbf{A}$   $n^2 \times n^2$



## Solving for Nodal Velocities

- Factorize:  $A^T = LDL^T$  where D = diagonal matrix  
L = lower triangular

$$\underbrace{LDL^T}_{\mathbf{L} \quad \mathbf{Q}} \dot{\mathbf{x}}^{t+\Delta t} = \mathbf{b}^t$$

$$\mathbf{L} \quad \mathbf{Q} = \mathbf{b}^t$$

- Solve lower triangular system for Q (forward-substitution)
- Solve upper triangular system (back-substitution)

$$DL^T \dot{\mathbf{x}}^{t+\Delta t} = \mathbf{Q}$$

$$\mathbf{L}^T \dot{\mathbf{x}}^{t+\Delta t} = \mathbf{D}^{-1} \mathbf{Q} \Rightarrow \mathbf{x}^{t+\Delta t} = \mathbf{x}^t + \Delta t \dot{\mathbf{x}}^{t+\Delta t}$$

## Untangling Cloth

SIGGRAPH 2002 Submission  
(Paper #018)

## Outline

- Physical Simulations
- Particle Dynamics
  - Mass-Spring Systems:
    - Examples of Mass-Spring Systems
    - Fundamentals of Mass-Spring Systems
  - Applications
    - Heating and Melting Deformable Models – Mass-springs Model
    - Liquids – Particle Models
    - Cloth – Viscoelasticity – Mass-springs Model
    - Implicit Euler
  - Human Motion Modeling**
    - Rigid Body Dynamics**
    - Motion Capture – Statistical Analysis – Numerical Tensor Analysis

## Rigid-Body Dynamics

To create a nearly rigid object using a mass-spring-damper system, make the springs really stiff

- This works in principle, but leads to numerical instability in practice

Better to use rigid-body dynamics

- There are no such things as perfectly rigid bodies in the real world, so this is an approximation

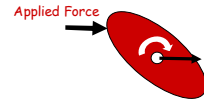
When a force is applied to extended bodies, the movement induced can consist of both translation and rotation

- Rotation is modeled explicitly in rigid-body dynamics
- A force applied other than at the **center of mass** (COM) of the extended body produces a **torque**

## Rigid Body Dynamics

Kinematics of 3D body in space

- Three translational degrees of freedom:  $\mathbf{x}$
- Three rotational degrees of freedom:  $\theta$



Inertia tensor

- Specifies how mass is distributed about the COM

Equations of motion

$$m\mathbf{a} = \mathbf{f}$$

$$\frac{d}{dt} \mathbf{I} \boldsymbol{\omega} = \boldsymbol{\tau}$$

Torque

Angular Velocity  
 $d\theta/dt$

$$\mathbf{I} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix}$$

where

$$I_{xx} = \int (y^2 + z^2) dm \quad I_{xy} = \int xy dm$$

$$I_{yy} = \int (x^2 + z^2) dm \quad I_{xz} = \int xz dm$$

$$I_{zz} = \int (x^2 + y^2) dm \quad I_{yz} = \int yz dm$$

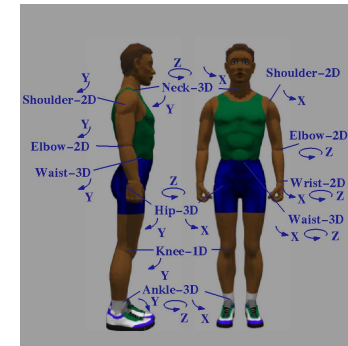
## Articulated Dynamics

Rigid bodies with joints

- A.k.a. constrained multibody systems

### Dynamic human model

- J. Hodgins, et al. GATech
- 15-17 rigid body parts
- 22-32 controlled dofs
- Body part densities from anthropometric data
- Masses & moments calculated from polygonal model



## "Atlanta in Motion"

J. Hodgins, et al.,  
Georgia Tech



## Falling Backward, Rolling Over, Rising, and Balancing in Gravity



Help, I've fallen! ...

*and I can get up!*

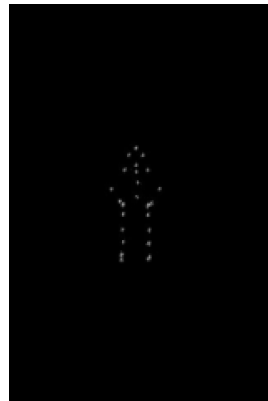
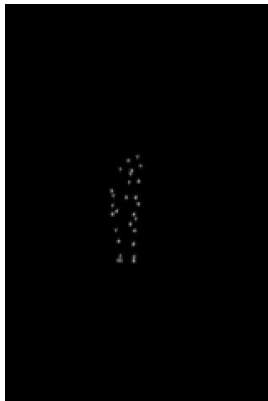
## The Virtual Stuntman Does a Kip Stunt



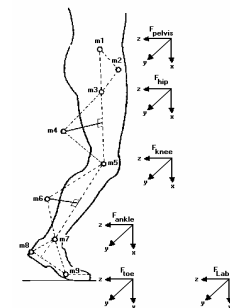
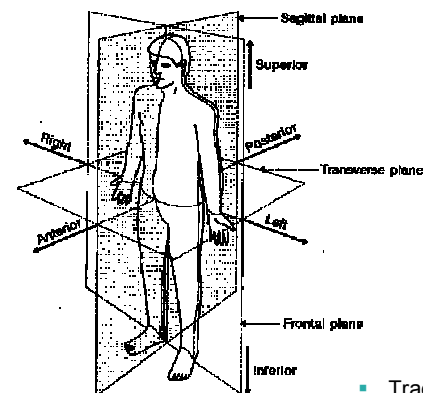
## Outline

- Physical Simulations
  - Particle Dynamics
  - Mass-Spring Systems:
    - Examples of Mass-Spring Systems
    - Fundamentals of Mass-Spring Systems
  - Applications
    - Heating and Melting Deformable Models – Mass-springs Model
    - Liquids – Particle Models
    - Cloth – Viscoelasticity – Mass-springs Model
    - Implicit Euler
  - Human Motion Modeling
    - Rigid Body Dynamics
    - **Motion Capture – Statistical Analysis – Numerical Tensor Analysis**

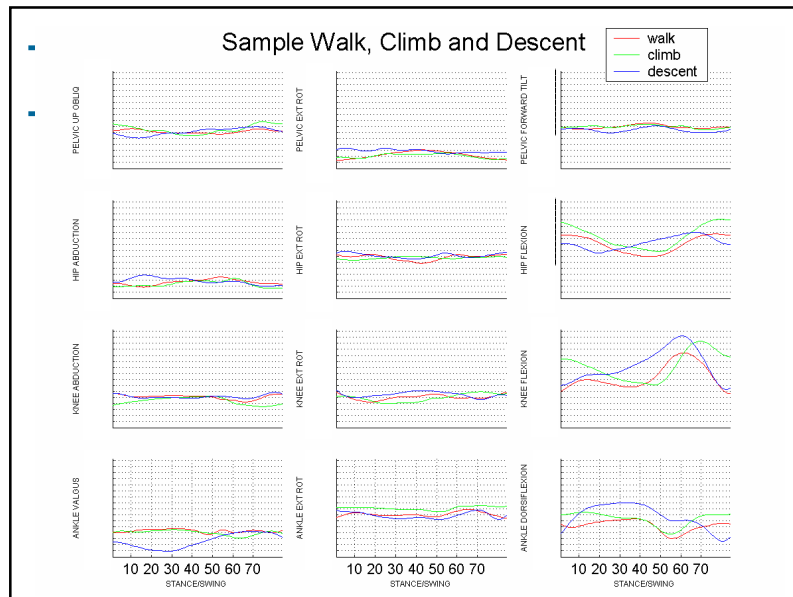
## Motion Capture Data



## Reference Planes



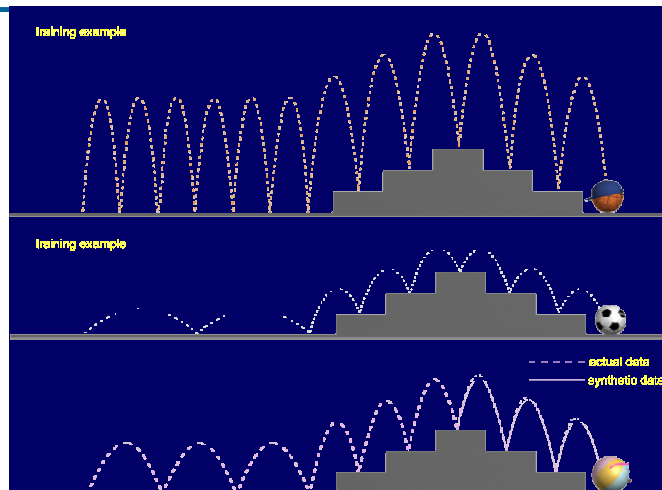
- Track 3 rotational angles for each body part by creating a plane for each body part using 3 reflective markers



## Human Motion Signatures -- Basic Idea

- Motion is modeled as the composite consequence of multiple elements
  - the action performed -- essence of an activity
  - an expressive cadence -- nuances of attitude or sentiment expressed temporally in motion
  - a motion signature -- distinctive pattern of movement of a particular individual
- Extract signature, action parameters from a corpus of data spanning multiple subjects performing different types of motions
- Synthesize novel motions
  - synthesize walk from descending stairs
  - synthesize ascent from walk and descent
  - ....

## Toy Example – Ball Demo



## Animated Ball – Toy Example

