

Physical simulation for animation

Case study: The jello cube

The Jello Cube
Mass-Spring System
Collision Detection
Integrators

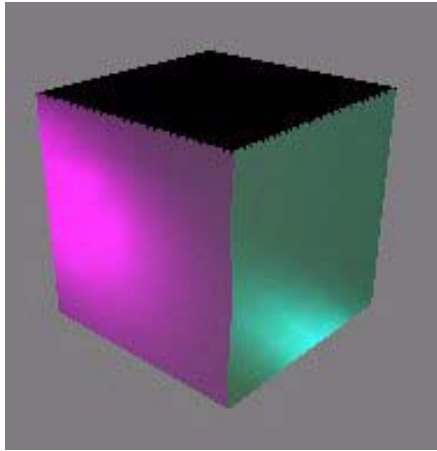


September 17 2002

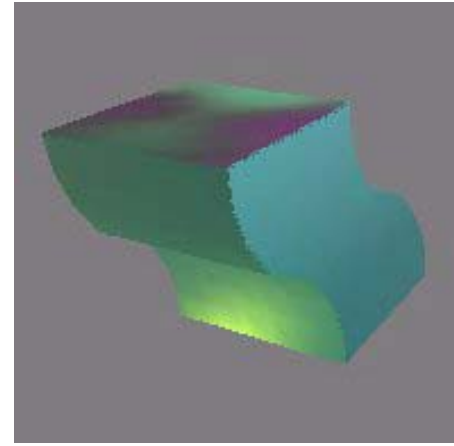
Announcements

- **Programming assignment 3 is out. It is due Tuesday, November 5th midnight.**
- **Midterm exam:**
 - Next week on Thursday in class

The jello cube



Undeformed cube



Deformed cube

- **The jello cube is elastic,**
- **Can be bent, stretched, squeezed, ...,**
- **Without external forces, it eventually restores to the original shape.**

Physical simulations

- **Model nature by using the laws of physics**
- **Often, the only way to achieve realism**
- **Alternative: try various non-scientific tricks to achieve realistic effects**
- **Math becomes too complicated very quickly**
Isn't it incredible that nature can compute everything (you, me, and the whole universe) on the fly, it is the fastest computer ever.
- **Important issues: simulation accuracy and stability**

Simulation or real?



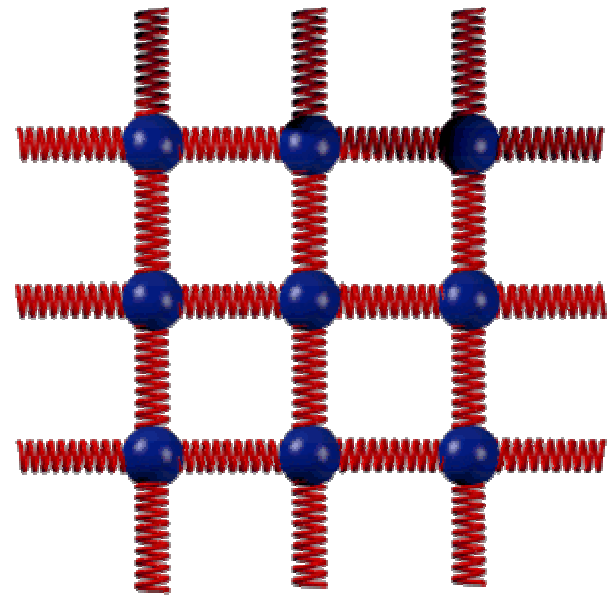
Mass-Spring System

- Several mass points
- Connected to each other by springs
- Springs expand and stretch, exerting force on the mass points
- Very often used to simulate cloth
- Examples:

[A 2-particle spring system](#)

[Another 2-particle example](#)

[Cloth animation example](#)

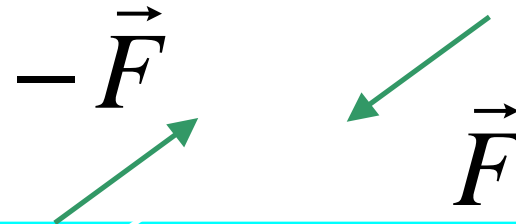


Newton's Laws

- Newton's 2nd law:

$$\vec{F} = m\vec{a}$$

- Tells you how to compute acceleration, given the force and mass
- Newton's 3rd law: If object A exerts a force F on object B, then object B is at the same time exerting force $-F$ on A.

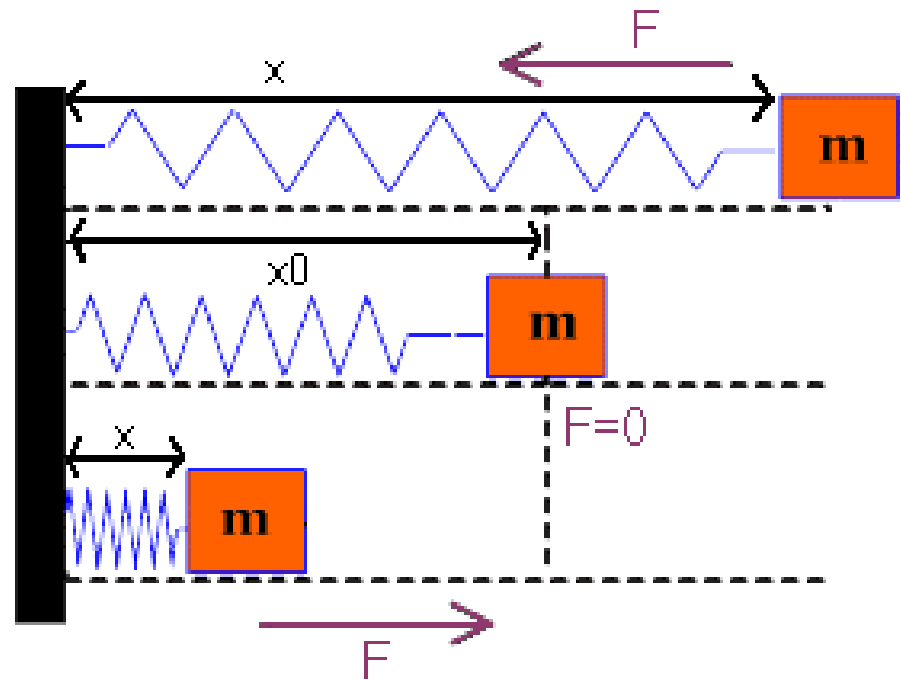


Single spring

- Obeys the *Hook's law*:

$$F = k (x - x_0)$$

- x_0 = rest length
- k = spring elasticity (aka stiffness)
- For $x < x_0$, spring wants to extend
- For $x > x_0$, spring wants to contract



Hook's law in 3D

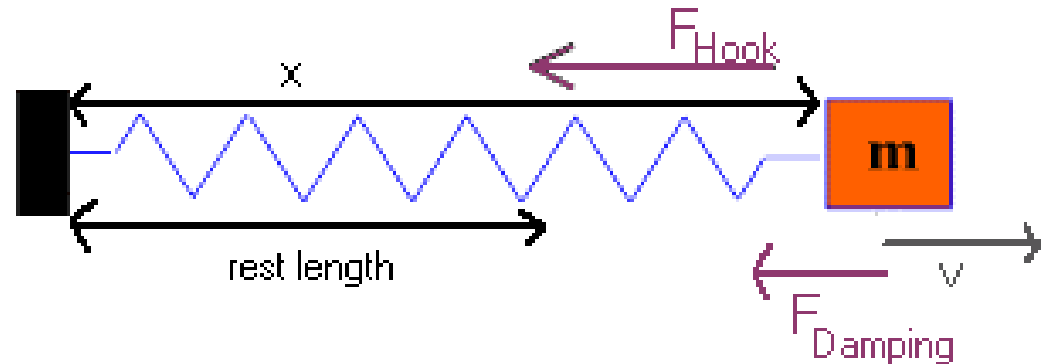
- Assume A and B two mass points connected with a spring.
- Let \vec{L} be the vector pointing from B to A
- Let R be the spring rest length
- Then, the elastic force exerted on A is:

$$\vec{F} = -k_{Hook} (|\vec{L}| - R) \frac{\vec{L}}{|\vec{L}|}$$

Damping

- Springs are not completely elastic
- They absorb some of the energy and tend to decrease the velocity of the mass points attached to them
- Damping force depends on the velocity:

$$\vec{F} = -k_d \vec{v}$$



- k_d = damping coefficient
- k_d different than k_{Hook} !!

Damping in 3D

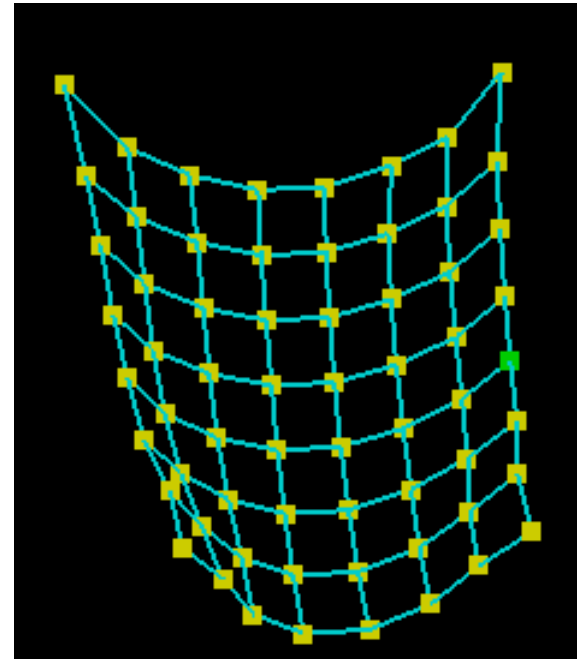
- Assume A and B two mass points connected with a spring.
- Let \vec{L} be the vector pointing from B to A
- Then, the damping force exerted on A is:

$$\vec{F} = -k_d \frac{(\vec{v}_A - \vec{v}_B) \cdot \vec{L}}{|\vec{L}|} \frac{\vec{L}}{|\vec{L}|}$$

- Here \vec{v}_A and \vec{v}_B are velocities of points A and B
- Damping force always **OPPOSES** the motion

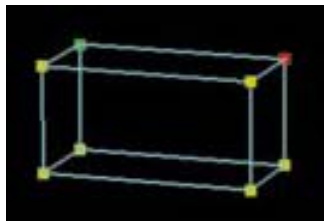
A network of springs

- **Every mass point connected to some other points by springs**
- **Springs exert forces on mass points**
 - Hook's force
 - Damping force
- **Other forces**
 - External force field
 - » Gravity
 - » Electrical or magnetic force field
 - Collision force

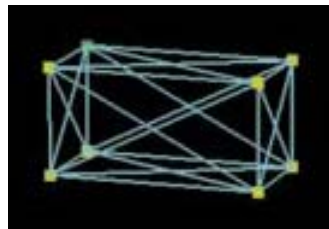


How to organize the network (for jello cube)

- To obtain stability, must organize the network of springs in some clever way
- Jello cube is a 8x8x8 mass point network
- 512 discrete points
- Must somehow connect them with springs



Basic network



Stable network

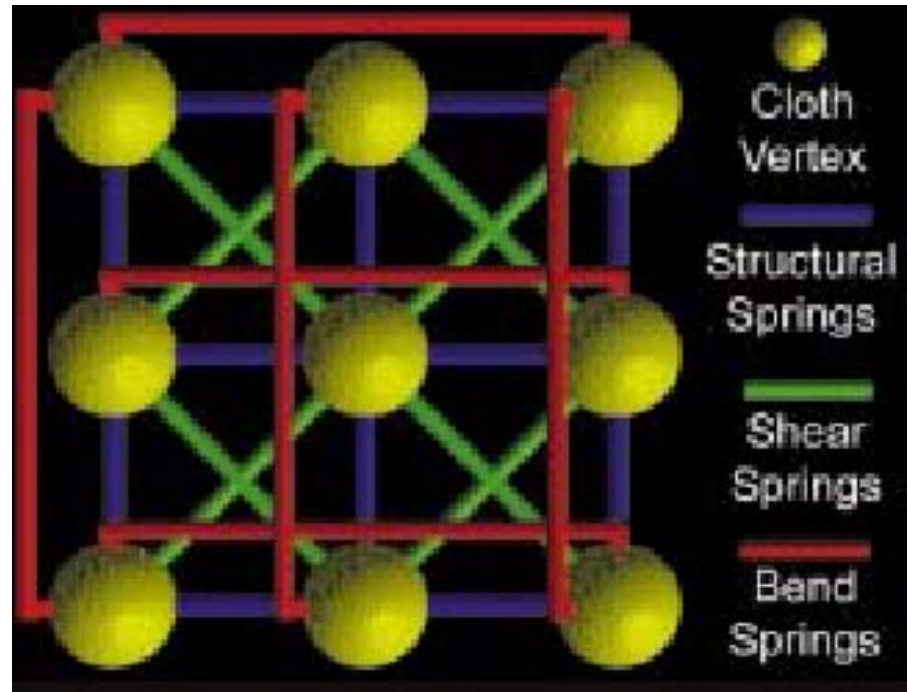


Network out
of control

Solution:

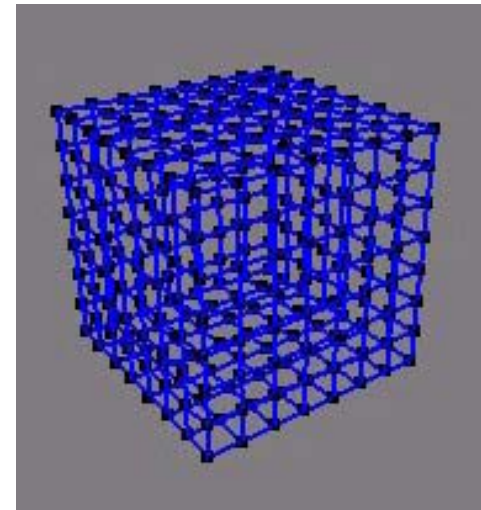
Structural, Shear and Bend Springs

- There will be three types of springs:
 - Structural
 - Shear
 - Bend
- Each has its own function



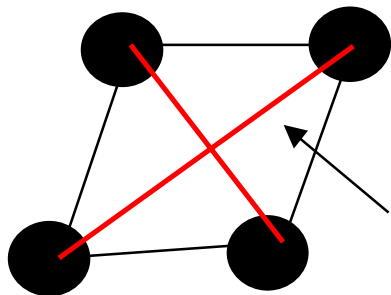
Structural springs

- **Connect every node to its 6 direct neighbours**
- **Node (i,j,k) connected to**
 - $(i+1,j,k), (i-1,j,k), (i,j-1,k), (i,j+1,k), (i,j,k-1), (i,j,k+1)$
(for surface nodes, some of these neighbors might not exist)
- **Structural springs establish the basic structure of the jello cube**
- **The picture shows structural springs for the jello cube. Only springs connecting two surface vertices are shown.**



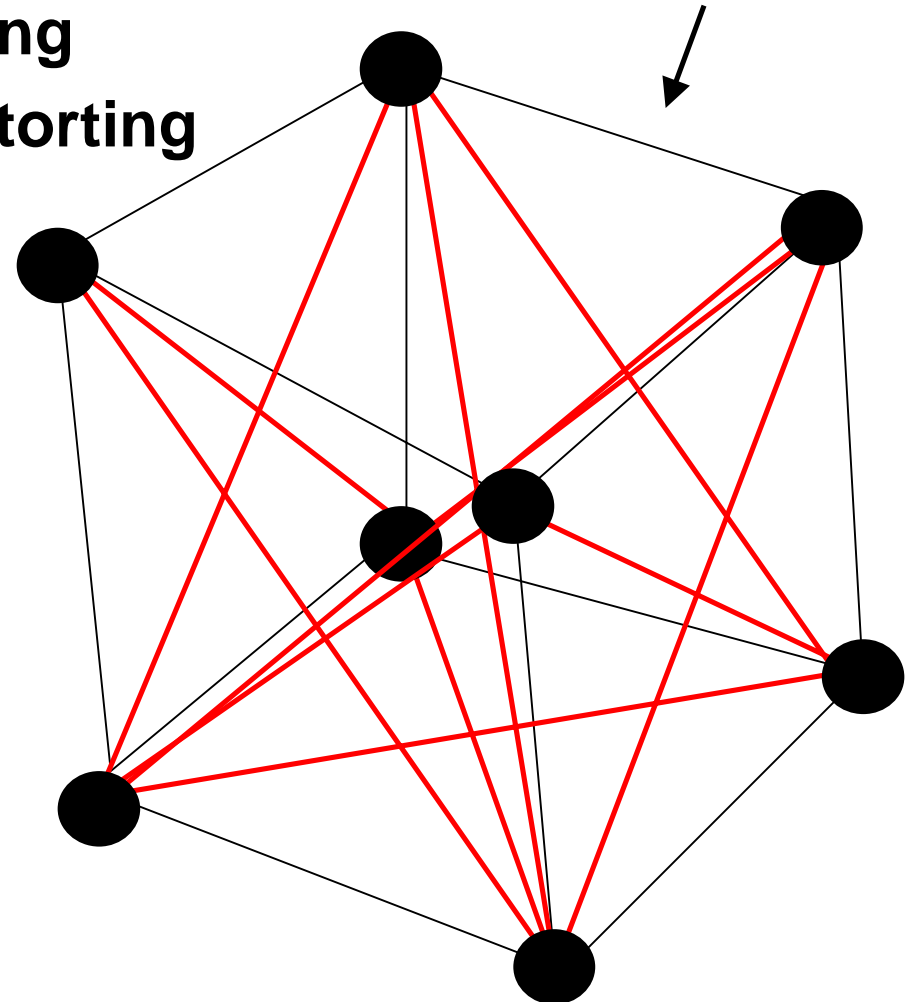
Shear springs

- Disallow excessive shearing
- Prevent the cube from distorting
- Every node (i,j,k) connected to its diagonal neighbors
- Structural springs = white
- Shear springs = red



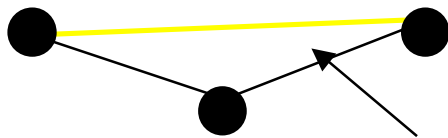
Shear spring (red) resists stretching and thus prevents shearing

A 3D cube
(if you can't see it immediately, keep trying)

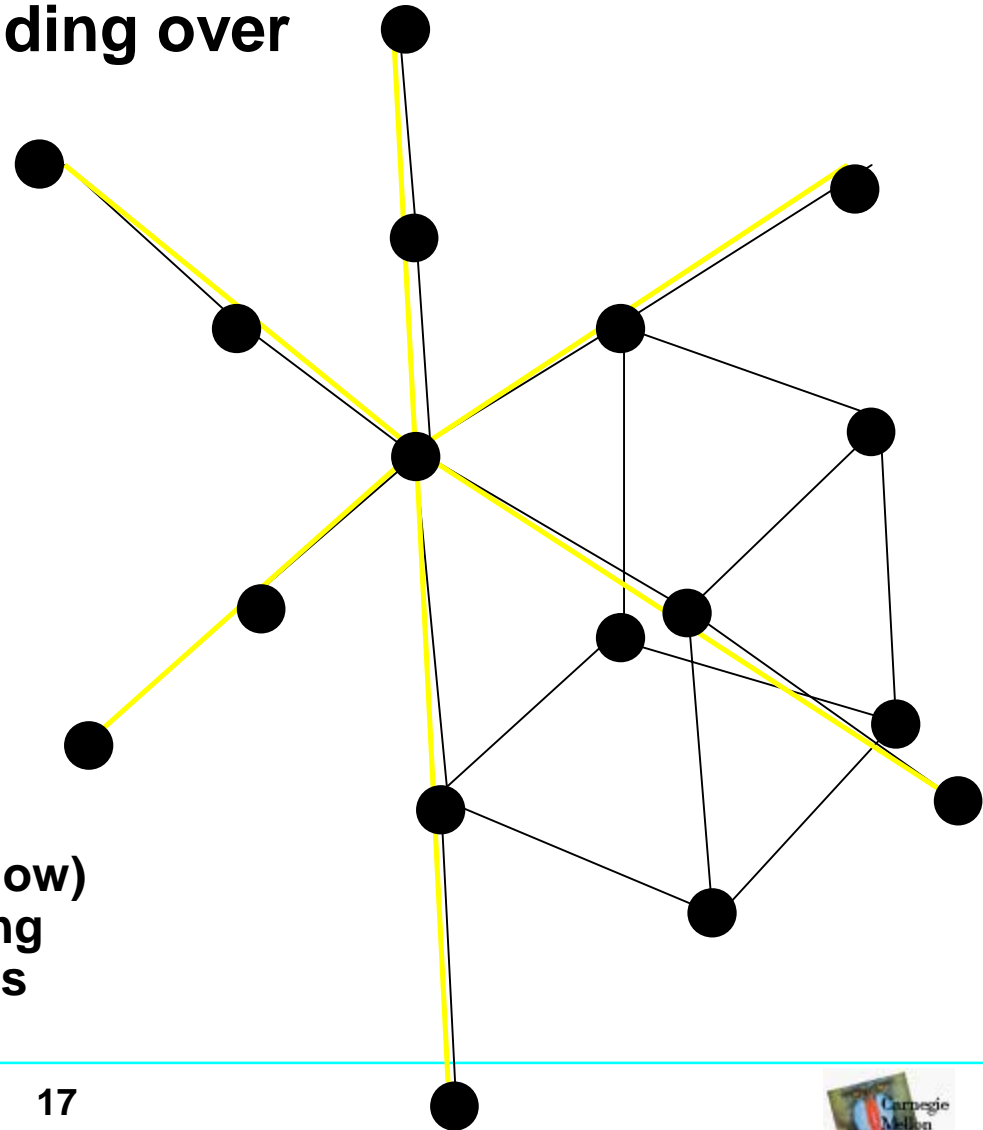


Bend springs

- Prevent the cube from folding over
- Every node connected to its second neighbor in every direction (6 connections per node, unless surface node)
- white=structural springs
- yellow=bend springs (shown for a single node only)



Bend spring (yellow) resists contracting and thus prevents bending

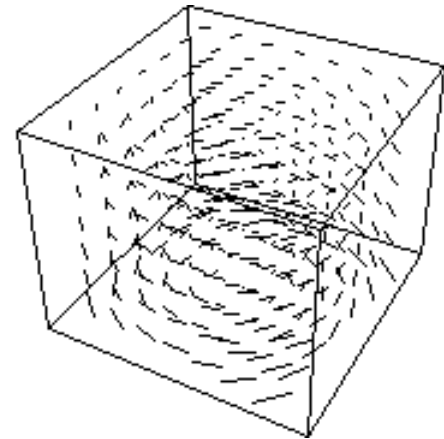


External force field

- If there is an external force field, add that force to the sum of all the forces on a mass point

$$\vec{F}_{total} = \vec{F}_{Hook} + \vec{F}_{damping} + F_{force\ field}$$

- There is one such equation for every mass point and for every moment in time



Collision detection

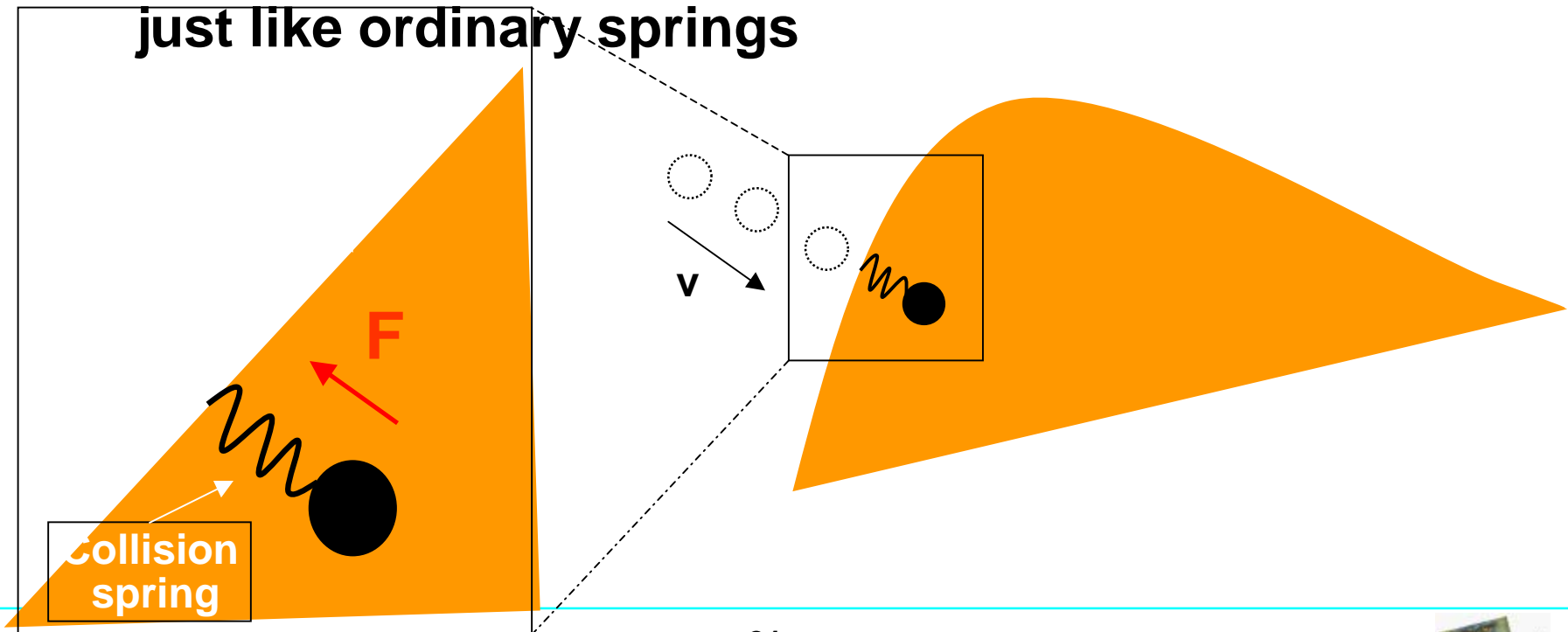
- The movement of the jello cube is limited to a bounding box
- Collision detection easy:
 - Check all the vertices if any of them is outside the box
- Inclined plane:
 - Equation:
$$F(x, y, z) = ax + by + cz + d = 0$$
 - Initially, all points on the same side of the plane
 - $F(x,y,z) > 0$ on one side of the plane and $F(x,y,z) < 0$ on the other
 - Can check all the vertices for this condition

Collision response

- **When collision happens, must perform some action to prevent the object penetrating even deeper**
- **Object should bounce away from the colliding object**
- **Some energy is usually lost during the collision**
- **Several ways to handle collision response**
- **We will use the *penalty method***

The penalty method

- When collision happens, put an artificial *collision spring* at the point of collision, which will push the object backwards and away from the colliding object
- Collision springs have elasticity and damping, just like ordinary springs



Integrators

- Network of mass points and springs
- Hook's law, damping law and Newton's 2nd law give acceleration of every mass point at any given time
- $F=ma$
 - Hook's law and damping provide F
 - 'm' is point mass
 - The value for a follows from $F=ma$
- Now, we know acceleration at any given time for any point
- Want to compute the actual motion

Integrators (contd.)

- The equations of motion:

$$\frac{d\vec{x}}{dt} = \vec{v}$$

$$\frac{d^2\vec{x}}{dt^2} = \frac{d\vec{v}}{dt} = \vec{a}(t) = \frac{1}{m} (\vec{F}_{Hook} + \vec{F}_{damping} + \vec{F}_{force\ field})$$

- \mathbf{x} = point position, \mathbf{v} = point velocity, \mathbf{a} = point acceleration
- They describe the movement of any single mass point
- \mathbf{F}_{hook} = sum of all Hook forces on a mass point
- $\mathbf{F}_{damping}$ = sum of all damping forces on a mass point

Integrators (contd.)

- When we put these equations together for all the mass points, we obtain a system of ordinary differential equations.
- In general, impossible to solve analytically
- Must solve numerically
- Methods to solve such systems numerically are called *integrators*
- Most widely used:
 - Euler
 - Runge-Kutta 2nd order (aka the midpoint method) (RK2)
 - Runge-Kutta 4th order (RK4)

Integrator design issues

- **Numerical stability**
 - If time step too big, method “explodes”
 - $t = 0.001$ is a good starting choice for the assignment
 - Euler much more unstable than RK2 or RK4
 - » Requires smaller time-step, but is simple and hence fast
 - Euler rarely used in practice
- **Numerical accuracy**
 - Smaller time steps means more stability and accuracy
 - But also means more computation
- **Computational cost**
 - Tradeoff: accuracy vs computation time

Integrators (contd.)

- RK4 is often the method of choice
- RK4 very popular for engineering applications
- The time step should be inversely proportional to the square root of the elasticity k [*Courant condition*]
- For the assignment, we provide the integrator routines (Euler, RK4)
 - void Euler(struct world * jello);
 - void RK4(struct world * jello);
 - Calls to these routines make the simulation progress one time-step further.
 - State of the simulation stored in 'jello' and automatically updated

Tips

- **Use double precision for all calculations (double)**
- **Do not overstretch the z-buffer**
 - It has finite precision
 - Ok: `gluPerspective(90.0,1.0,0.01,1000.0);`
 - Bad: `gluPerspective(90.0,1.0,0.0001,100000.0);`
- **Choosing the right elasticity and damping parameters is an art**
 - Trial and error
 - For a start, can set the ordinary and collision parameters the same
- **Read the webpage for updates and check the bulletin board**