

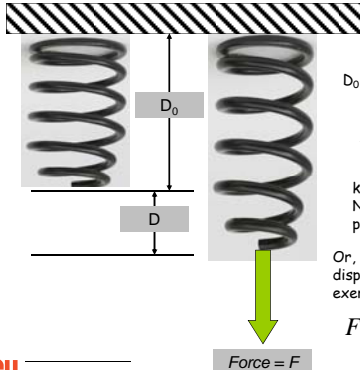
Modeling the World as a Mesh of Springs

Mike Bailey
Oregon State University



Oregon State University
Computer Graphics

How do Springs Work?



D_0 = unloaded spring length

$$D = D_0 + \frac{F}{k}$$

k = spring stiffness in Newtons/meter or pounds/inch

Or, if you know the displacement, the force exerted by the spring must be:

$$F = k(D - D_0)$$

Oregon State University
Computer Graphics

Solving Motion where there is a Spring

$$F_{spring} = k(D - D_0)$$

$$v(y, t) = \int \frac{\sum F}{m} dt$$

$$v(y, t) = \int \frac{W - ky}{m} dt$$

```

void
GetVelAcc( float t, float y, float vy, float *v, float *a )
{
    *v = vy;
    *a = ( W - K*y ) / MASS;
}
    
```

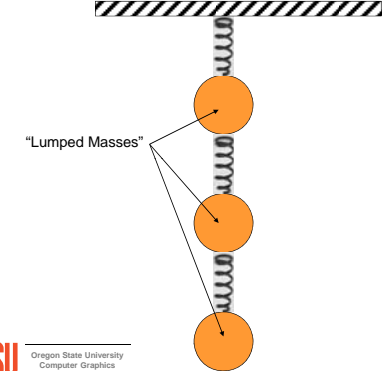
Note that, instead of using weight, we could write that last line with mass:

```

*a = ( MASS*GRAVITY - K*y ) / MASS;
    
```

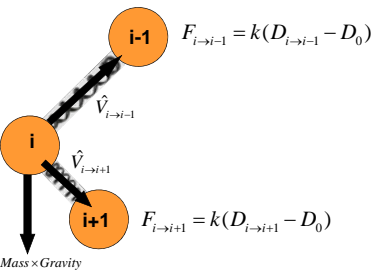
Oregon State University
Computer Graphics

Modeling a String as a Group of Masses Connected by Springs



Oregon State University
Computer Graphics

Computing Forces



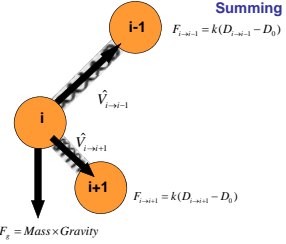
$$F_{i \rightarrow i-1} = k(D_{i \rightarrow i-1} - D_0)$$

$$F_{i \rightarrow i+1} = k(D_{i \rightarrow i+1} - D_0)$$

$$F_g = Mass \times Gravity$$

Oregon State University
Computer Graphics

Summing Forces



$$F_{i-1} = k(D_{i-1} - D_0)$$

$$F_{i+1} = k(D_{i+1} - D_0)$$

$$F_g = Mass \times Gravity$$

$$\sum F_{i,x} = \hat{v}_{i \rightarrow i-1,x} \cdot F_{i \rightarrow i-1} + \hat{v}_{i \rightarrow i+1,x} \cdot F_{i \rightarrow i+1} = Mass \cdot \ddot{x}$$

$$\sum F_{i,y} = \hat{v}_{i \rightarrow i-1,y} \cdot F_{i \rightarrow i-1} + \hat{v}_{i \rightarrow i+1,y} \cdot F_{i \rightarrow i+1} + F_g = Mass \cdot \ddot{y}$$

Oregon State University
Computer Graphics

Solve for Each State as a Whole, not as Individual Links: Do it this Way

Second Order solution:

```

for( int i = 0; i < NUMLINKS; i++)
{
    GetVelAccel( i, &vx1[i], &vy1[i], &ax1[i], &ay1[i] );
}
for( int i = 0; i < NUMLINKS; i++)
{
    vxtmp[i] = Links[i].vx + DT * ax1[i];
    vytmp[i] = Links[i].vy + DT * ay1[i];
    xtmp[i] = Links[i].x + DT * vx1[i];
    ytmp[i] = Links[i].y + DT * vy1[i];
}
for( int i = 0; i < NUMLINKS; i++)
{
    GetVelAccel( i, &vx2[i], &vy2[i], &ax2[i], &ay2[i] );
}
for( int i = 0; i < NUMLINKS; i++)
{
    Links[i].vx = Links[i].vx + DT * ( ax1[i] + ax2[i] ) / 2.;
    Links[i].vy = Links[i].vy + DT * ( ay1[i] + ay2[i] ) / 2.;
    Links[i].x = Links[i].x + DT * ( vx1[i] + vx2[i] ) / 2.;
    Links[i].y = Links[i].y + DT * ( vy1[i] + vy2[i] ) / 2.;
}
    
```

Get all the velocities and accelerations

Apply all the velocities and accelerations

Get all the velocities and accelerations

Apply all the velocities and accelerations

DSU Oregon State University Computer Graphics mp - November 5, 2000

Don't do it this Way!

Second Order solution:

```

for( int i = 0; i < NUMLINKS; i++)
{
    GetVelAccel( i, &vx1[i], &vy1[i], &ax1[i], &ay1[i] );
    vxtmp[i] = Links[i].vx + DT * ax1[i];
    vytmp[i] = Links[i].vy + DT * ay1[i];
    xtmp[i] = Links[i].x + DT * vx1[i];
    ytmp[i] = Links[i].y + DT * vy1[i];
}
for( int i = 0; i < NUMLINKS; i++)
{
    GetVelAccel( i, &vx2[i], &vy2[i], &ax2[i], &ay2[i] );
    Links[i].vx = Links[i].vx + DT * ( ax1[i] + ax2[i] ) / 2.;
    Links[i].vy = Links[i].vy + DT * ( ay1[i] + ay2[i] ) / 2.;
    Links[i].x = Links[i].x + DT * ( vx1[i] + vx2[i] ) / 2.;
    Links[i].y = Links[i].y + DT * ( vy1[i] + vy2[i] ) / 2.;
}
    
```

DSU Oregon State University Computer Graphics mp - November 5, 2000

Changing Variables on-the-fly

DSU Oregon State University Computer Graphics mp - November 5, 2000

Simulating a String

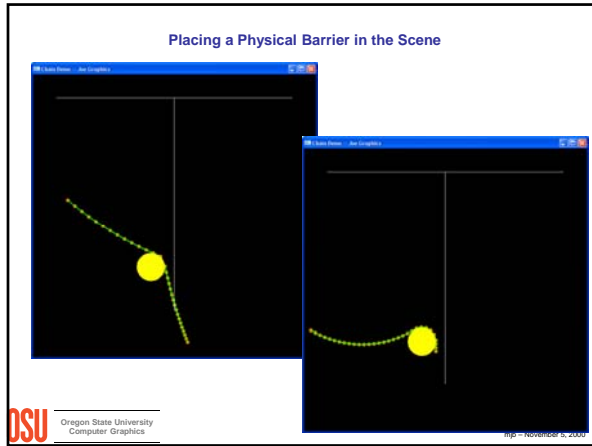
DSU Oregon State University Computer Graphics mp - November 5, 2000

Less Damping

DSU Oregon State University Computer Graphics mp - November 5, 2000

First Order Instability

DSU Oregon State University Computer Graphics mp - November 5, 2000



Placing a Physical Barrier in the Scene

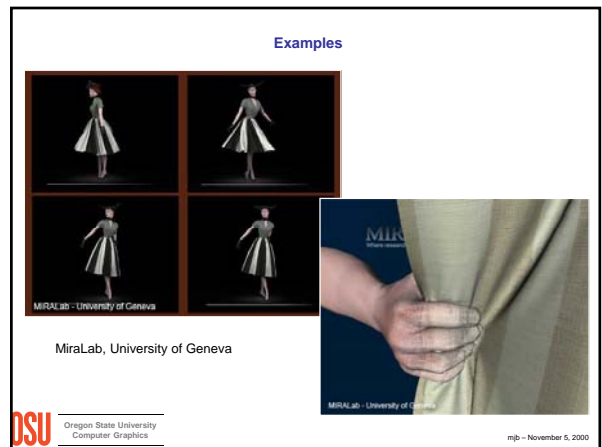
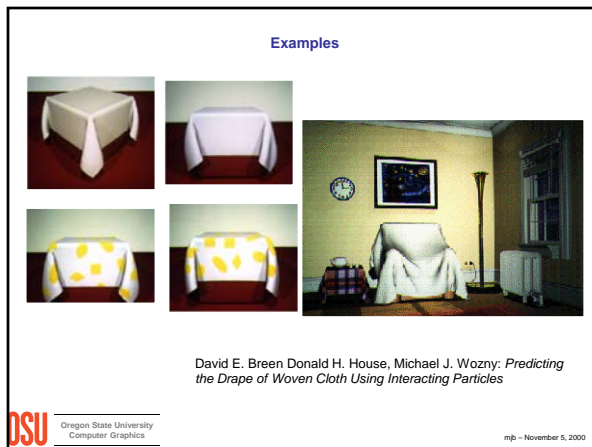
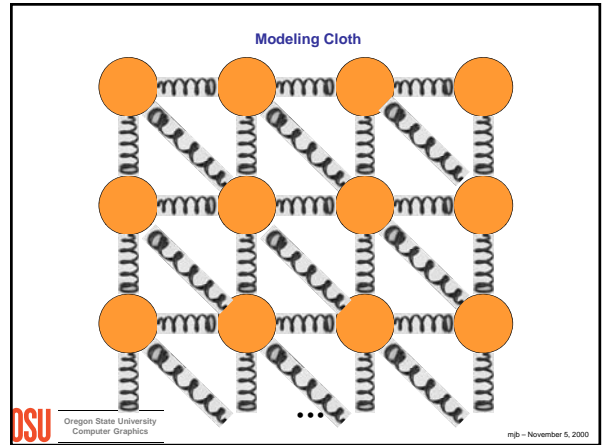
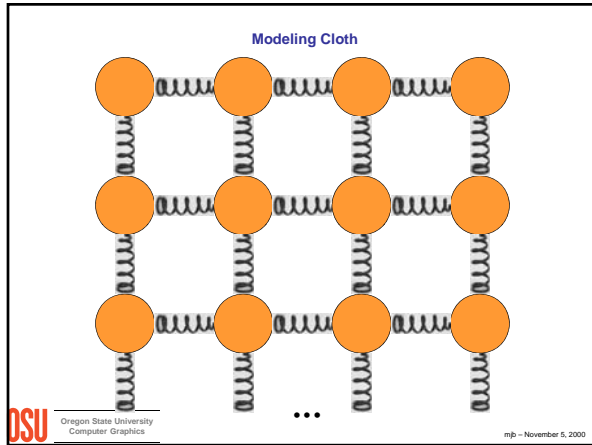
```

if( DoCircle )
{
  for( int i = 0; i < NUMLINKS; i++ )
  {
    float dx = Links[i].x - CIRCX;
    float dy = Links[i].y - CIRCY;
    float rsqd = dx*dx + dy*dy;
    if( rsqd < CIRCRC*CIRCRC )
    {
      float r = sqrt( rsqd );
      dx /= r;
      dy /= r;
      Links[i].x = CIRCX + CIRCRC * dx;
      Links[i].y = CIRCY + CIRCRC * dy;
      Links[i].vx *= dy;
      Links[i].vy *= -dx;
    }
  }
}
  
```


- Vector from circle center to the lumped mass
- If the lumped mass is inside the circle ...
- Unit vector from circle center to the lumped mass
- Push the lumped mass from inside the circle to the circle's surface
- Keep just the tangential velocity

Oregon State University
Computer Graphics

mp - November 5, 2000



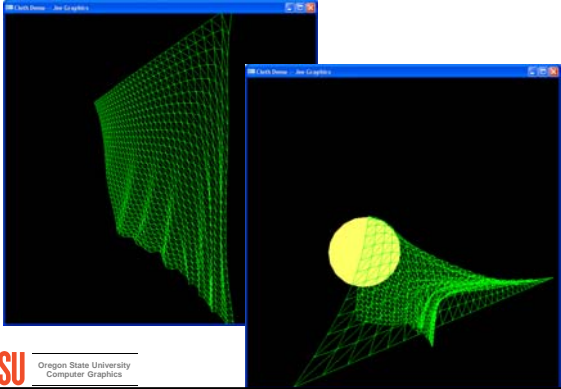
Examples



Pixar

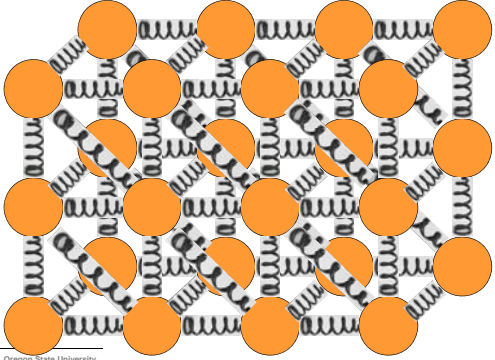
DSU Oregon State University Computer Graphics mp - November 5, 2000

Examples



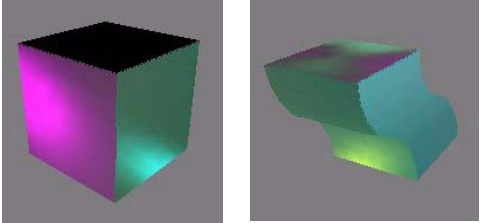
DSU Oregon State University Computer Graphics mp - November 5, 2000

Modeling Jello



DSU Oregon State University Computer Graphics mp - November 5, 2000

Examples



MIT

DSU Oregon State University Computer Graphics mp - November 5, 2000

We can also use this same Method to Model Rigid Objects



California Department of Transportation

DSU Oregon State University Computer Graphics mp - November 5, 2000

We can also use this same Method to Model Rigid Objects



DSU Oregon State University Computer Graphics mp - November 5, 2000

