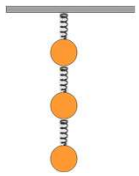
1

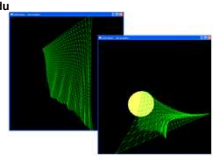
Modeling the World as a Mesh of Springs



Oregon State University

Mike Bailey

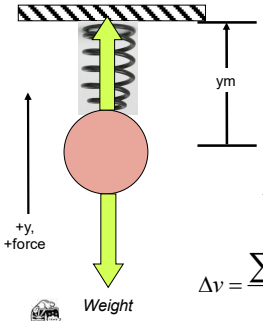
mjb@cs.oregonstate.edu



1

Solving for Motion where there is a Spring

2



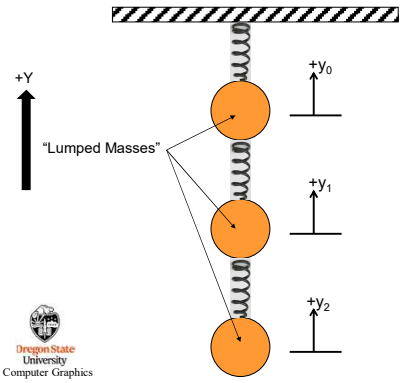
$$F_{spring} = -k(y - D_0)$$

$$\Delta v = \frac{\sum F}{m}\Delta t = \frac{-W - k(y - D_0)}{m}\Delta t$$

2

Modeling a String as a Group of Masses Connected by Springs

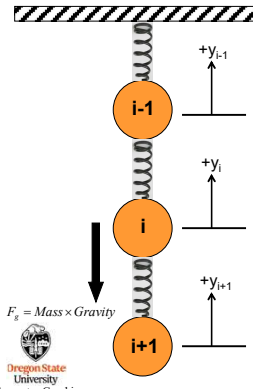
3



3

Computing Forces in 1D

4



Generalize by using indices:

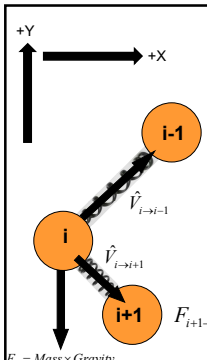
$$F_{i-1\to i} = k(Y_{i-1} - Y_i - D_0)$$

$$F_{i+1\to i} = k(Y_{i+1} - Y_i - D_0)$$

4

Computing Forces in 2D

5



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

X: $F_{i-1\to i,x} = F_{i-1\to i}\cos\theta_{i\to i-1} = F_{i-1\to i}\frac{x_{i-1} - x_i}{D_{i\to i-1}}$

Y: $F_{i-1\to i,y} = F_{i-1\to i}\sin\theta_{i\to i-1} = F_{i-1\to i}\frac{y_{i-1} - y_i}{D_{i\to i-1}}$

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

X: $F_{i+1\to i,x} = F_{i+1\to i}\cos\theta_{i\to i+1} = F_{i+1\to i}\frac{x_{i+1} - x_i}{D_{i\to i+1}}$

Y: $F_{i+1\to i,y} = F_{i+1\to i}\sin\theta_{i\to i+1} = F_{i+1\to i}\frac{y_{i+1} - y_i}{D_{i\to i+1}}$

5

From the *Physics* Notes: What does a Second Order solution look like in a Program?

6

```
void
AdvanceOneTimeStep( )
{
        GetDerivs( state, &derivs1);
        state2.t    = state.t   + Δt;
        state2.x    = state.x   + derivs1->vx * Δt;
        state2.vx = state.vx + derivs1->ax * Δt;

        GetDerivs( state2, &derivs2 );
        float aavg = ( derivs1->ax + derivs2->.ax) / 2.;
        float vavg = ( derivs1->vx + derivs2->vx) / 2.;

        state.x   = state.x   + vavg * Δt;
        sate.vx   = state.vx + aavg * Δt;
        state.t    = state.t   + Δt ;
}
```


6

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

7

Correct Second Order solution:

```
for( int i = 0; i < NUMLINKS; i++ )
{
        GetOneBodysDerivs( Links, i,  &vx1[i], &vy1[i], &ax1[i], &ay1[i] );
}
for( int i = 0; i < NUMLINKS; i++ )
{
        TmpLinks[i].vx = Links[i].vx  + DT * ax1[i];
        TmpLinks[i].vy = Links[i].vy  + DT * ay1[i];
        TmpLinks[i].x  = Links[i].x   + DT * vx1[i];
        TmpLinks[i].y  = Links[i].y   + DT * vy1[i];
}

for( int i = 0; i < NUMLINKS; i++ )
{
        GetOneBodysDerivs( TmpLinks, 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 first

Apply *all* the velocities and accelerations

Get *all* the velocities and accelerations first

Apply *all* the velocities and accelerations

Computer Graphics

mjb – July 31, 2021

7

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

8

C Extensions for Array Notion (CEAN) makes it look cleaner, and possibly more efficient:

```
for( int i = 0; i < NUMLINKS; i++ )
{
        GetOneBodysDerivs( Links, i,  &vx1[i], &vy1[i], &ax1[i], &ay1[i] );
}

TmpLinks[ 0 : NUMLINKS ].vx = Links[ 0 : NUMLINKS ].vx  + DT  * ax1[ 0 : NUMLINKS ];
TmpLinks[ 0 : NUMLINKS ].vy = Links[ 0 : NUMLINKS i].vy  + DT * ay1[ 0 : NUMLINKS ];
TmpLinks[ 0 : NUMLINKS ].x  = Links[ 0 : NUMLINKS ].x   + DT * vx1[ 0 : NUMLINKS ];
TmpLinks[ 0 : NUMLINKS ].y  = Links[ 0 : NUMLINKS ].y   + DT * vy1[ 0 : NUMLINKS ];

for( int i = 0; i < NUMLINKS; i++ )
{
        GetOneBodysDerivs( TmpLinks, i,  &vx2[i], &vy2[i], &ax2[i], &ay2[i] );
}

Links[ 0 : NUMLINKS ].vx = Links[ 0 : NUMLINKS ].vx  +  DT * ( ax1[ 0 : NUMLINKS ] + ax2[ 0 : NUMLINKS ] ) / 2.;
Links[ 0 : NUMLINKS ].vy = Links[ 0 : NUMLINKS ].vy  +  DT * ( ay1[ 0 : NUMLINKS ] + ay2[ 0 : NUMLINKS ] ) / 2.;
Links[ 0 : NUMLINKS ].x  = Links[ 0 : NUMLINKS ].x   +  DT * ( vx1[ 0 : NUMLINKS ] + vx2[ 0 : NUMLINKS ] ) / 2.;
Links[ 0 : NUMLINKS ].y  = Links[ 0 : NUMLINKS ].y   +  DT * ( vy1[ 0 : NUMLINKS ] + vy2[ 0 : NUMLINKS ] ) / 2.;
```

Oregon State University
Computer Graphics

mjb – July 31, 2021

8

GetLinkVelAcc( ), I

9

```
void
GetOneBodysDerivs(  array, node,  float *vxi, float *vyi, float *axi, float *ayi )
{
        float xm, ym;           // vector from node to previous node (up the chain)
        float xp,  yp;          // vector from node to next node (down the chain)

        float sumfx = 0.;
        float sumfy = -Weight;

        if( node == 0 )
        {
                xm = X0 - array[node].x;                // (X0,Y0) is the top of the chain
                ym = Y0 - array[node].y;
        }
        else
        {
                xm = array[node-1].x - array [node].x;
                ym = array[node-1].y - array [node].y;
        }

        float length = sqrt( xm*xm + ym*ym );           // length of the spring
        xm /= length;                                   // normalize the vector
        ym /= length;
        float stretch = length - D0;                    // amount the spring is stretched
        float force = K * stretch;
        sumfx += force * xm;
        sumfy += force * ym;
```

Computer Graphics

mjb – July 31, 2021

9

GetLinkVelAcc( ), II

10

```
        if( node < NUMLINKS-1 )
        {
                xp = array [node+1].x - array[node].x;
                yp = array [node+1].y - array [node].y;
                length = sqrt( xp*xp + yp*yp );
                xp /= length;
                yp /= length;
                stretch = length - D0;
                force = K * stretch;
                sumfx += force * xp;
                sumfy += force * yp;
        }

        sumfx -= Cd * array[node].vx;           // damping
        sumfy -= Cd * array[node].vy;

        *vx = array[node].vx;
        *vy = array[node].vy;
        *ax = sumfx / Mass;
        *ay = sumfy / Mass;
}
```

Oregon State University
Computer Graphics

mjb – July 31, 2021

10

Don't do it this Way!

11

Incorrect Second Order solution:

```
for( nt i = 0; i < NUMLINKS; i++ )
{
        GetOneBodysDerivs( Links, i,  &vx1[i], &vy1[i], &ax1[i], &ay1[i] );
        TmpLinks[i].vx = Links[i].vx  + DT * ax1[i];
        TmpLinks[i].vy = Links[i].vy  + DT * ay1[i];
        TmpLinks[i].x  = Links[i].x   + DT * vx1[i];
        TmpLinks[i].y  = Links[i].y   + DT * vy1[i];
}

for( nt i = 0; i < NUMLINKS; i++ )
{
        GetOneBodysDerivs( TmpLinks, 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.;
}
```

Changes the state before we are done getting the derivatives !

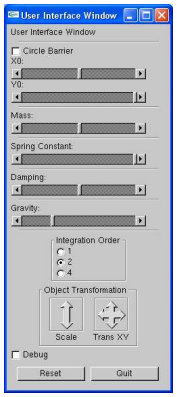
Oregon State University
Computer Graphics

mjb – July 31, 2021

11

Changing Variables on-the-fly in the String Project

12



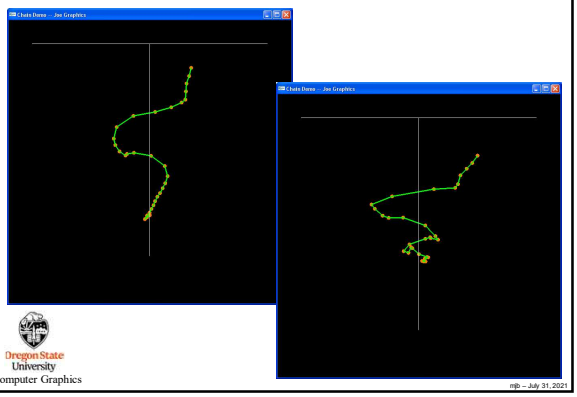
Oregon State University
Computer Graphics

mjb – July 31, 2021

12

Simulating a String

13



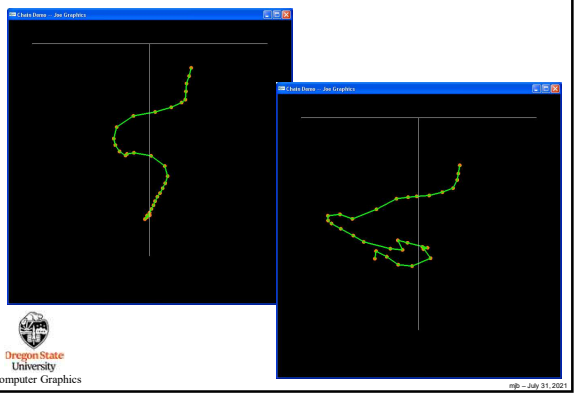
Oregon State University
Computer Graphics

mjb – July 31, 2021

13

Less Damping

14



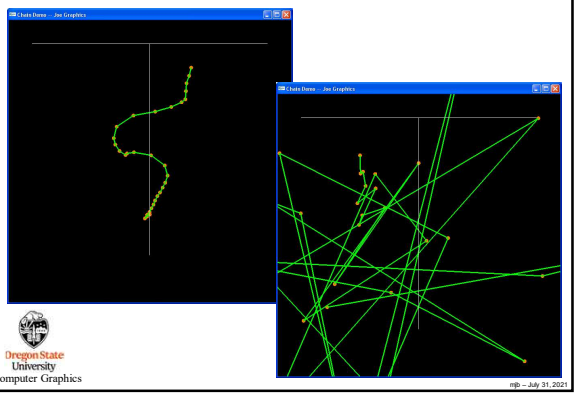
Oregon State University
Computer Graphics

mjb – July 31, 2021

14

First Order Instability

15



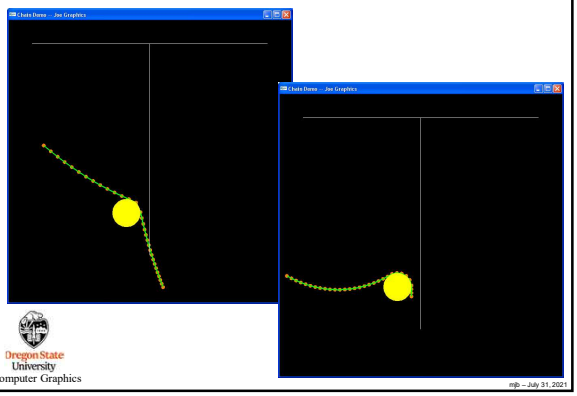
Oregon State University
Computer Graphics

mjb – July 31, 2021

15

Placing a Physical Barrier in the Scene

16



Oregon State University
Computer Graphics

mjb – July 31, 2021

16

Placing a Physical Barrier in the Scene

17

```
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 < CIRCR*CIRCR )
                {
                        float r = sqrt( rsqd );
                        dx /= r;
                        dy /= r;
                        Links[i].x = CIRCX + CIRCR * dx;
                        Links[i].y = CIRCY + CIRCR * 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.
dx = cosΘ
dy = sinΘ

Push the lumped mass from inside the circle to the circle's surface

Keep just the tangential velocity

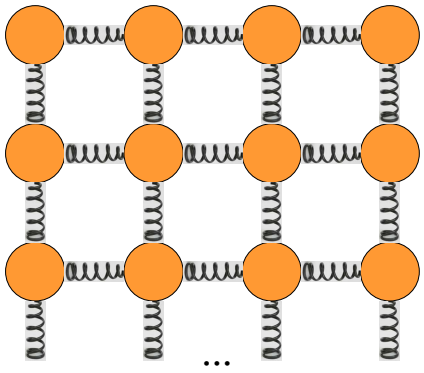
Oregon State University
Computer Graphics

mjb – July 31, 2021

17

Modeling Cloth

18



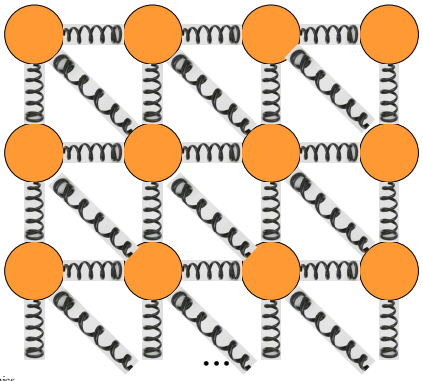
Oregon State University
Computer Graphics

mjb – July 31, 2021

18

Modeling Cloth

19



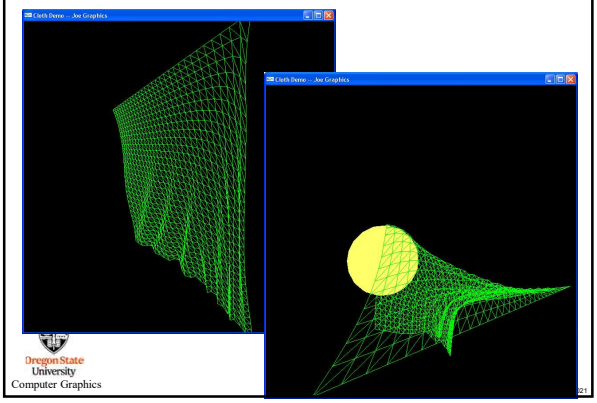
Oregon State University
Computer Graphics

mjb – July 31, 2021

19

Cloth Example

20



Oregon State University
Computer Graphics

mjb – July 31, 2021

20

Cloth Examples

21

David E. Breen Donald H. House, Michael J. Wozny: *Predicting the Drape of Woven Cloth Using Interacting Particles*

Oregon State University
Computer Graphics

mjb – July 31, 2021

21

Cloth Examples

22

MiraLab, University of Geneva

Oregon State University
Computer Graphics

mjb – July 31, 2021

22

Cloth Example

23

Pixar

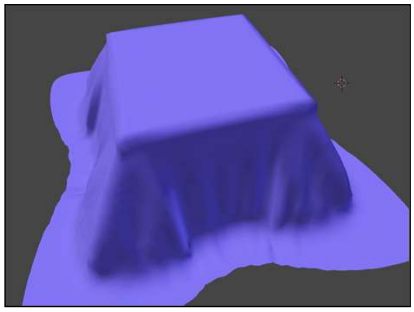
Oregon State University
Computer Graphics

mjb – July 31, 2021

23

Cloth Example with Blender

24



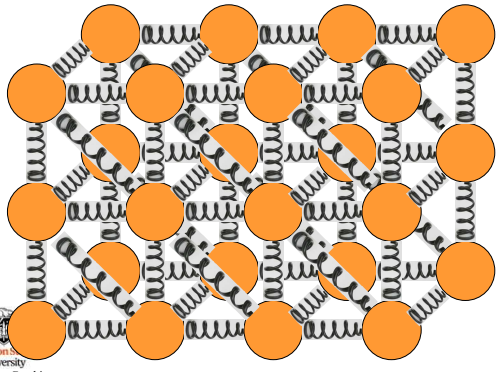
Oregon State University
Computer Graphics

mjb – July 31, 2021

24

Modeling Jello

25



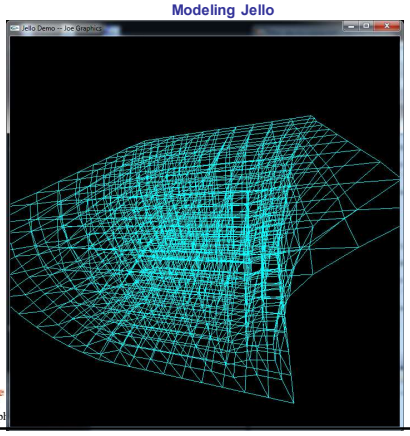
Oregon State University
Computer Graphics

mjb – July 31, 2021

25

Modeling Jello

26



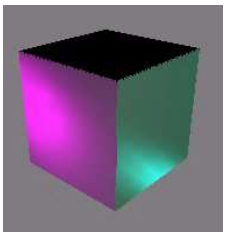
Oregon State University
Computer Graphics

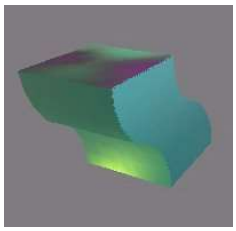
mjb – July 31, 2021

26

Modeling Jello

27





MIT

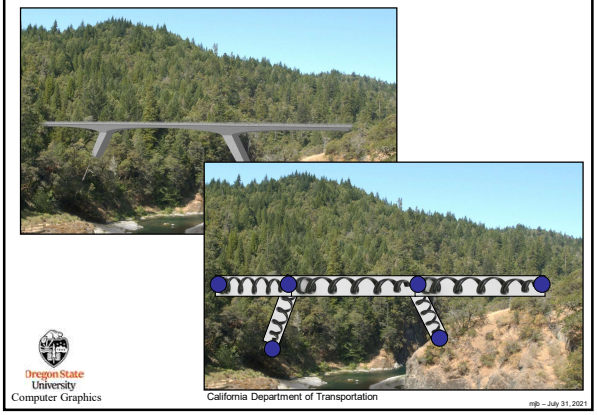
Oregon State University
Computer Graphics

mjb – July 31, 2021

27

We Can Also use this Same Method to Model and Analyze Rigid Objects

28



California Department of Transportation

Oregon State University
Computer Graphics

mjb – July 31, 2021

28

We Can Also use this Same Method to Model and Analyze Rigid Objects

29

OSU Johnson Hall

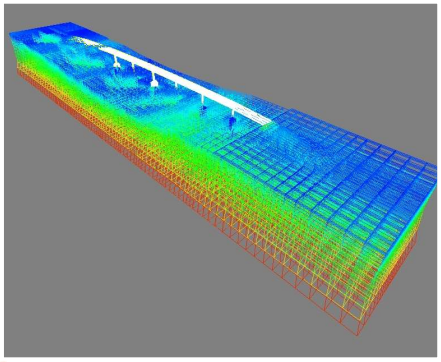
Oregon State University
Computer Graphics

mjb – July 31, 2021

29

A Bridge on Top of Loose Soil

30



Oregon State University
Computer Graphics

mjb – July 31, 2021

30