Modeling the World as a Mesh of Springs

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

Mike Bailey

mjb@cs.oregonstate.edu

1

Computing Forces in 1D

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

void AdvanceOneTimeStep()
{
  GetDerivs(State, Derivatives1);
  State2.x = State.x + Δt;
  State2.y = State.y + Derivatives1.y * Δt;
  State2.vx = State.vx + Derivatives1.vx * Δt;
  State2.vy = State.vy + Derivatives1.vy * Δt;
  GetDerivs(State2, Derivatives2);
  float ravg = (Derivatives1.x + Derivatives2.x) / 2.;  
  float vavg = (Derivatives1.vx + Derivatives2.vx) / 2.;  
  State.x = State.x + ravg * Δt;
  State.vx = State.vx + vavg * Δt;
  State.t = State.t + Δt;
}

1

Solving for Motion where there is a Spring

2

F_{spring} = -k(y - D_b)

Δy = \sum_{m} \frac{F}{m} \Delta t = \frac{-W - k(y - D_b)}{m} \Delta t

3

Modeling a String as a Group of Masses Connected by Springs

"Lumped Masses"

4

Computing Forces in 2D

5

\begin{align*}
F_{i,i-1} &= k(D_{i,i-1} - D_b) \\
F_{i,i+1} &= k(D_{i,i+1} - D_b) \\
F_{\text{r}i} &= \text{Mass} \times \text{Gravity} \\
F_{\text{y}i} &= \text{Force} = F_{\text{r}i} - F_{\text{y}i}
\end{align*}

6

void AdvanceOneTimeStep()
{
  GetDerivs(State, Derivatives1);
  State2.x = State.x + Δt;
  State2.y = State.y + Derivatives1.y * Δt;
  State2.vx = State.vx + Derivatives1.vx * Δt;
  State2.vy = State.vy + Derivatives1.vy * Δt;
  GetDerivs(State2, Derivatives2);
  float ravg = (Derivatives1.x + Derivatives2.x) / 2.;  
  float vavg = (Derivatives1.vx + Derivatives2.vx) / 2.;  
  State.x = State.x + ravg * Δt;
  State.vx = State.vx + vavg * Δt;
  State.t = State.t + Δt;
}
Solve for Each State as a Whole, not as Individual Links: Do It This Way

Correct Second Order solution:

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

Get all the velocities and accelerations first.

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

Apply all the velocities and accelerations first.

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

GetLinkVelAcc(), II

Don’t do it this Way!

Incorrect Second Order solution:

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

Changes the state before we are done getting the derivatives!

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

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

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

GetLinkVelAcc(), I

GetLinkVelAcc(), I

Solve for Each State as a Whole, not as Individual Links: Do It This Way

GetLinkVelAcc(), I

Changing Variables on-the-fly in Project 8
Simulating a String

Less Damping

First Order Instability

Placing a Physical Barrier in the Scene

Placing a Physical Barrier in the Scene

Modeling Cloth

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;
    }
  }
}
Modeling Cloth

Examples

Examples

Examples

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

California Department of Transportation

A Bridge on Top of Loose Soil