Issue No. 3, December 2012

G E D A I C A

*adj n 1.*of or relating to the culture, artifacts, customs of Gedae

Another year flys by! It's hard to believe that 2012 is almost at end. We hope everyone had a great year. I'm sure we can agree that the world has been a challenging place this year. If you're like us you see challenges as opportunities. Gedae has responded to the challenges of 2012 with focused growth – people, products and partners.

In 2012 we added proven advisor's and executives to the team, we announced two technology partners and we completed development of the Idea Language. 2013 will be a very exciting year at Gedae as we continue to increase staff, launch new products, and add more partners.

One of the real joys this holiday season is the opportunity to say thank you. Thank you for your business and for the numerous business relationships that have provided technical feedback to help guide the growth of Gedae's product family.

Wishing you the joy of family, the gift of friends, and the best of everything in 2013, *The Gedae Staff and their Families*

GEDAE IS GROWING

Website Update January 2013!
Visit the Gedae Website often to see huge upgrades we have in store. Check back often for news, information and tutorial updates.

DOLLARS AND SENSE

One of the principles that drives Gedae, Inc. is to create more value for the customer than we get from them. Not only should you get world class software and performance but world class value too! To that end Gedae's hardware simulation engine is now included with all licenses ***free!***

1247 N Church Street, STE 5, Moorestown, NJ 08057 USA www.gedae.com; gedae@gedae.com; (856) 231-4458

CODE, TOOLS & TIPS *Vibrating String Example*

Gedae 6.5 introduces iteration syntax to the !dea™ language. The syntax for iteration is derived from common procedural languages like C/C++ but has a few extra features to enable optimized parallelization and streaming.

The code shown to the right *(Fig. 1)* is the core loop of a vibrating string model. *(DISCLAIMER: the code may or may not be an accurate model of the physics of a string. No warranty expressed or implied. Use the code at your own risk.)*

```
do(p[n] = v_init[n], float v[n] = v_init[n], float t = 0) {
      // if pulse falls in this time period, apply it to the string
  int t1 = t/Pp;
  t2 = t - t1*Pp;
  Fp1 = t2 < dt ? Fp : 0.0;
  Fp2[n1] = (n1+1)==N0 ? Fp1 : 0.0;
      // curvature of string for small displacements
  d2p[n1]  = (p[n1]-2*p[n1+1]+p[n1+2]) / dx;
      // force on string segment
  f[n1]    = F * d2p[n1] - v[n1+1] * Kf + Fp2[n1];
  a[n1]    = f[n1]/M;  // F = M*a; - acceleration of segment
  v1[n1]   = v[n1+1] + a[n1] * dt; // v = v0 + v*t new velocity
  aveV[n1] = v[n1+1] + v1[n1]; // average velocity over dt
      // x = a*dt`2/2 + v*dt + x0; new position of string
  p1[n1]   = 0.5*a[n1]*dt`2 + aveV[n1]*dt/2 + p[n1+1];
  v[n]     = set(v,v1,1);
  p[n]     = set(p,p1,1);
  out[n]   = (n==0 || n==(N-1)) ? 0.0 : p1[n-1];
  t = t + dt;
  push out;
} while (1);
```

Figure 1: Idea Code

This example illustrates a do-while loop. In !dea, the do-while has an extra section after the "do" keyword for initialization of loop variables – variables whose memory is reused during the loop. Loop variables are unique in !dea in that they can appear on both the left and right hand side of expressions. The plot *(Fig. 2)* shows the position of the string at some point in the simulation. The vertical dimension is greatly exaggerated.

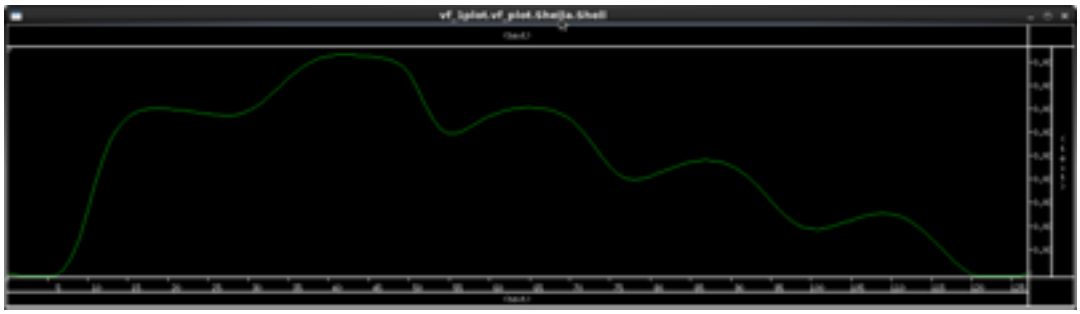

Figure 2: Simulation Output

The device we are modeling is a string that is anchored at both ends. The primary variables used in the simulation are listed in the table to the right. The dimensions (range variables) n and n1 are based on the number of segments used in the string model. The dimension n1 is 2 less than n since the endpoints are anchored and all variables are fixed at 0. The persistent time and velocity vectors are updated using the transient versions using the set() function. The key word push has been added to the language to support exporting data from a loop. It may seem a bit unusual. In this example, while the loop is infinite the variable out is exported on every iteration, producing a new token accessible outside the loop. The curious result is that there can be many loops executing within the application and all happily working concurrently. Hmmm! The Idea code can be easily tweaked to develop more interesting or accurate models. For example, in the simulation the pulse (tap) duration is one time increment of the models. It might make more sense to have a pulse duration variable as well as a pulse period. Download the example and experiment to your heart's content.

Name	Description
`p[n]/v[n]`	Persistent position / velocity of the string
`p1[n1]/v1[n1]`	Transient position / velocity of the string
`d2p[n1]`	2nd derivative of string position
`f[n1]`	Total force on string
`a[n1]`	Acceleration of string

Figure 3: Results Table

Is there an example you'd like to see in CODE, TOOLS & TIPS? – We're looking for industry parallel processing applications to implement in Idea Language and publish results. Submit yours today via cc@gedae.com.