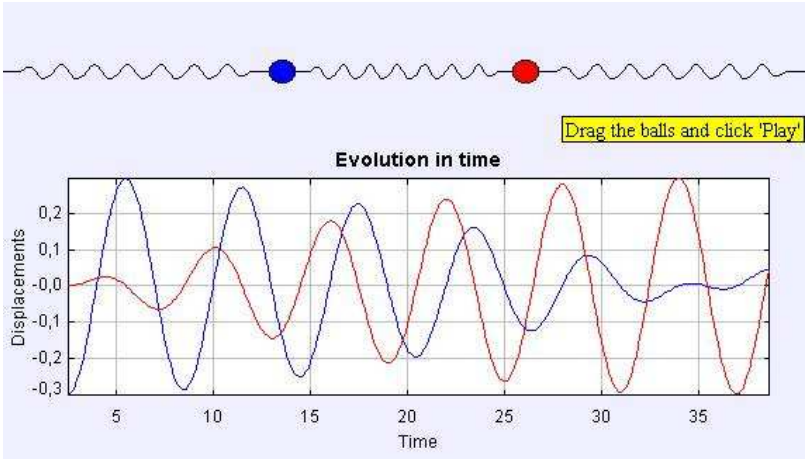
Coupled Oscillators



Description

We simulate the motion of two particle masses connected by three springs. One spring connects the two masses and two other springs connect them to the outer walls.

If k_1, k_2, and k_3 are the elastic constants of the springs (from left to right in the image), which both have a length at equilibrium l, and m_1 and m_2 the masses of the left and right particles, respectively, the equations of the positions (not displacement!) of the masses x1 and x2 are given by the coupled system of second-order differential equations:

$$m_1 \cdot \frac{d^2 x_1}{dt^2} = -k_1(\cdot x_1 - l) + k_2(x_2 - x_1 - l)$$
$$m_2 \cdot \frac{d^2 x_2}{dt^2} = -k_2(\cdot x_2 - x_1 - l) + k_3(2l - x_2)$$

Introducing additional variables for the horizontal velocities of the particles, vx_1 and vx_2, we can write this system as an equivalent system of four first-order differential equations:

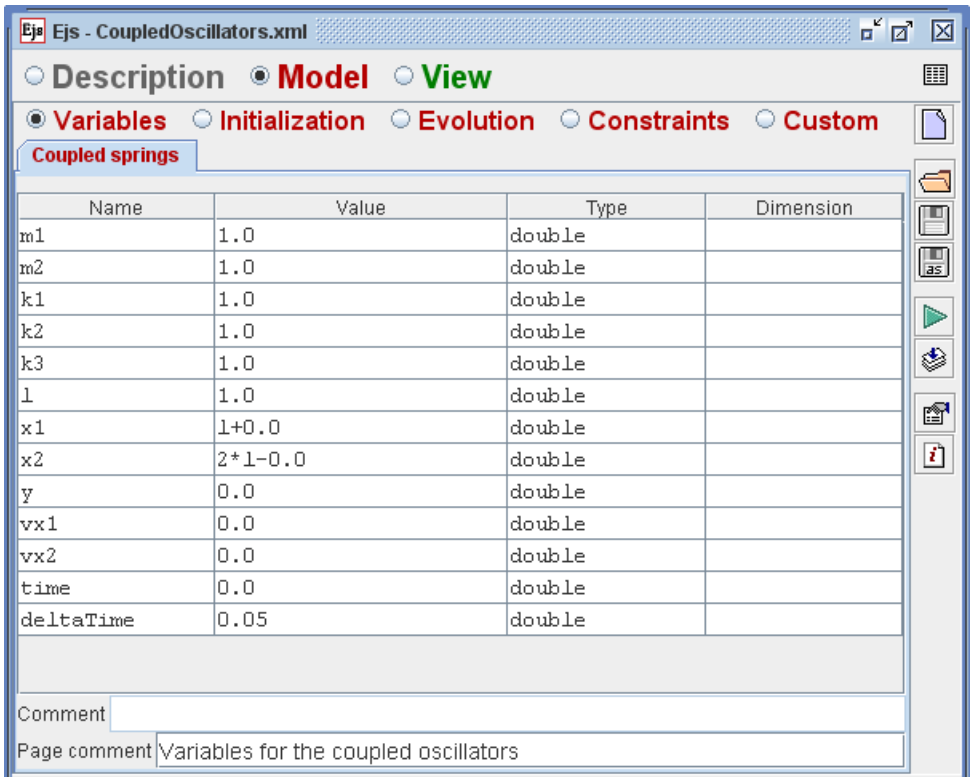
$$\frac{d \cdot x_1}{dt} = vx_1$$
$$\frac{d \cdot vx_1}{dt} = -\frac{k_1}{m_1}(x_1 - l) + \frac{k_2}{m_1}(x_2 - x_1 - l)$$
$$\frac{d \cdot x_2}{dt} = vx_2$$
$$\frac{d \cdot vx_2}{dt} = -\frac{k_2}{m_2}(\cdot x_2 - x_1 - l) + \frac{k_3}{m_2}(2l - x_2)$$

This formulation is ready for Ejs' editor of differential equations.

Model

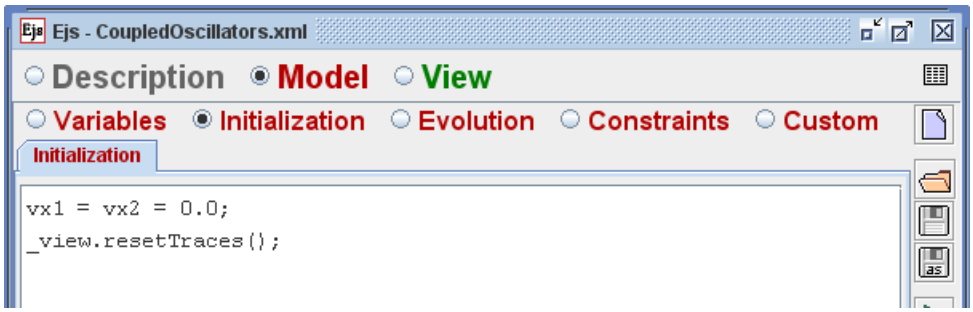
Variables

We need a rather long list of *double* variables whose meanings have been explained above.



Initialization

For the initialization we write the simple code:



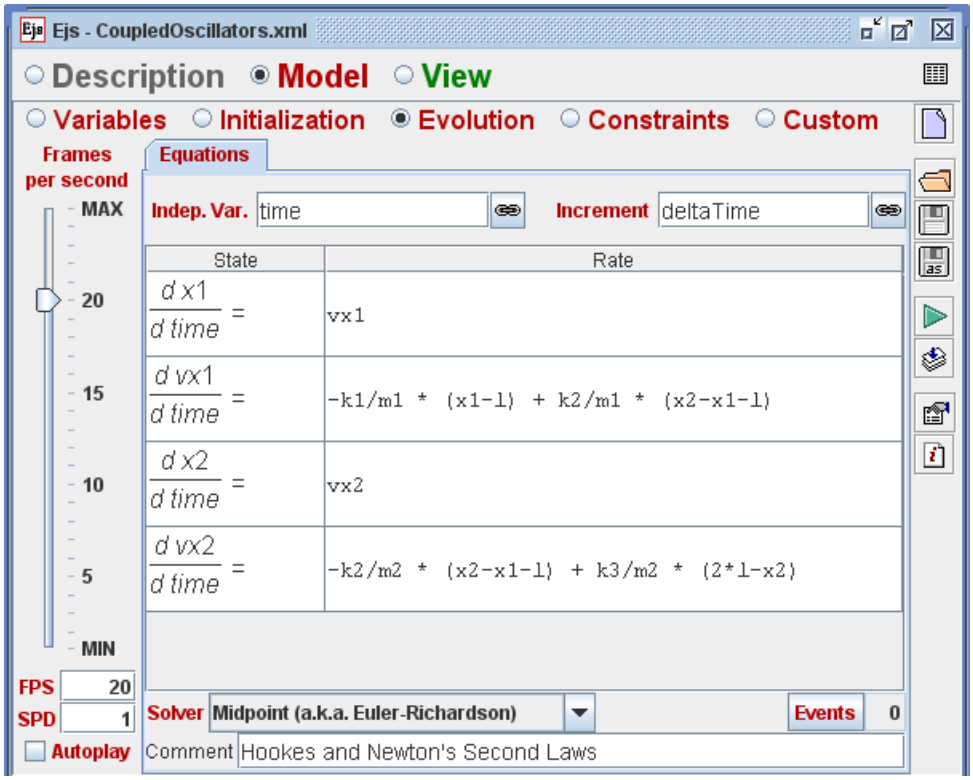
This code just freezes the particle at the given positions and the line:

```
_view.resetTraces();
```

clears the traces of the displacements.

Evolution

The evolution uses a page of ODEs:



We choose the Euler-Richardson second-order method which is powerful enough for these rather well-behaved equations. But there is no reason why a more powerful Runge-Kutta method can be selected instead.

Constraints

No constraints are required.

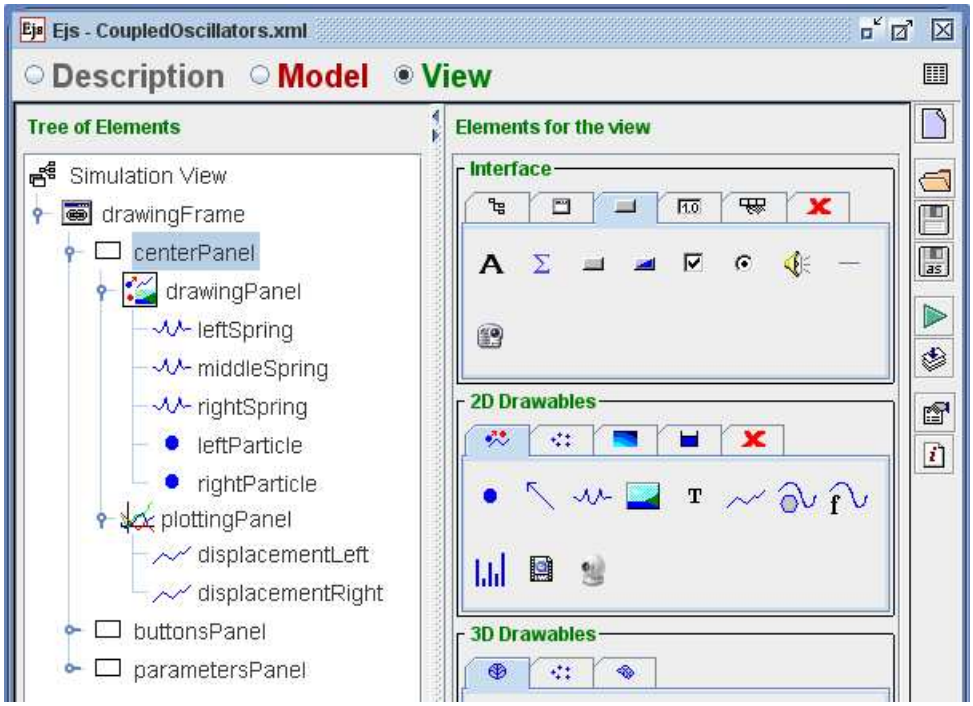
Custom code

No custom code required. See however the *Action* property of the "Special modes:" buttons in the view.

View

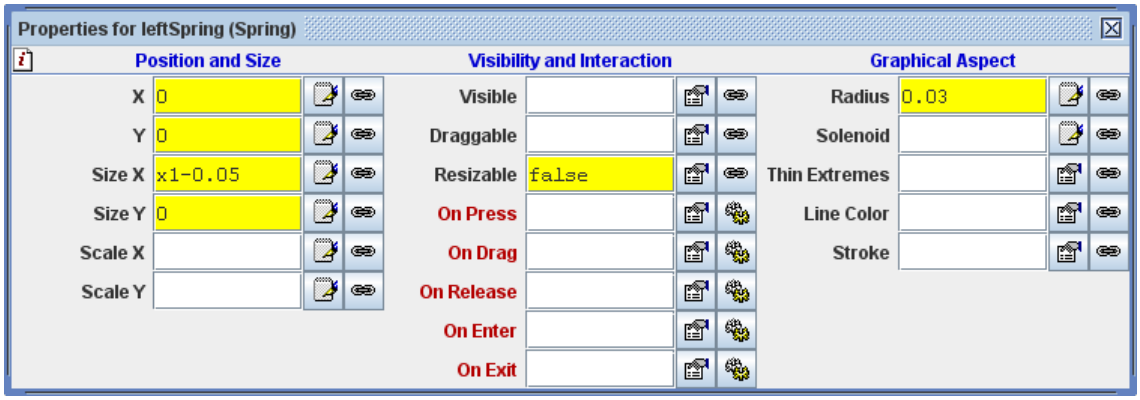
The view starts with the compound element based on a drawing panel with a default particle, but we will need to make several changes to it.

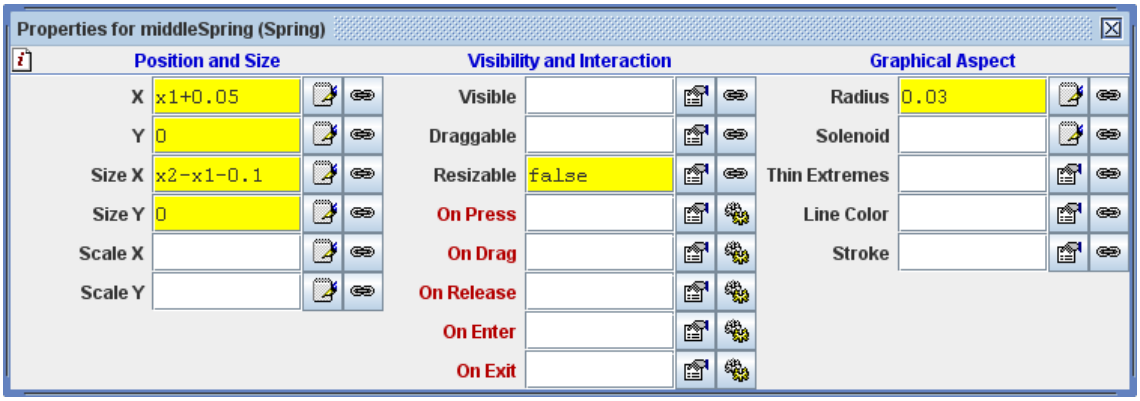
First, we replace the central drawing panel with a simpler *Panel* container with *Border* layout which will hold a drawing panel and a plotting panel, both with different sizes. See the detail of the *Tree of Elements* in the figure below:

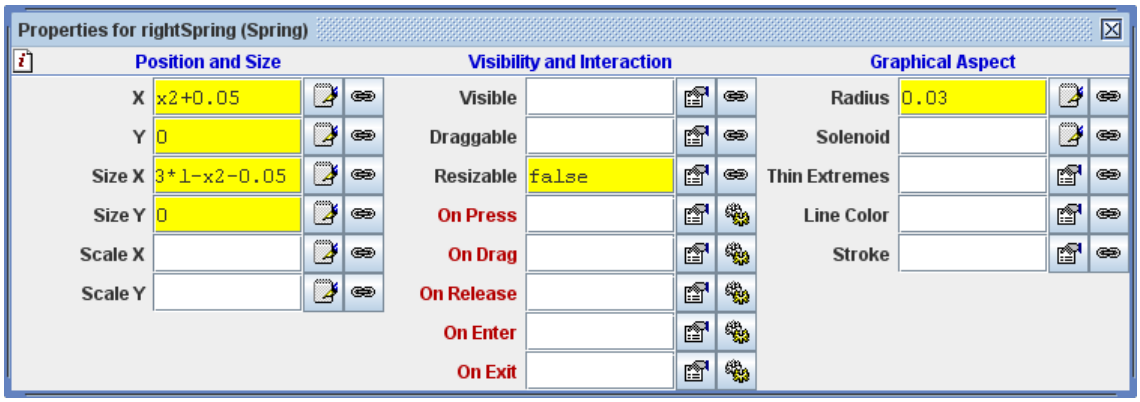


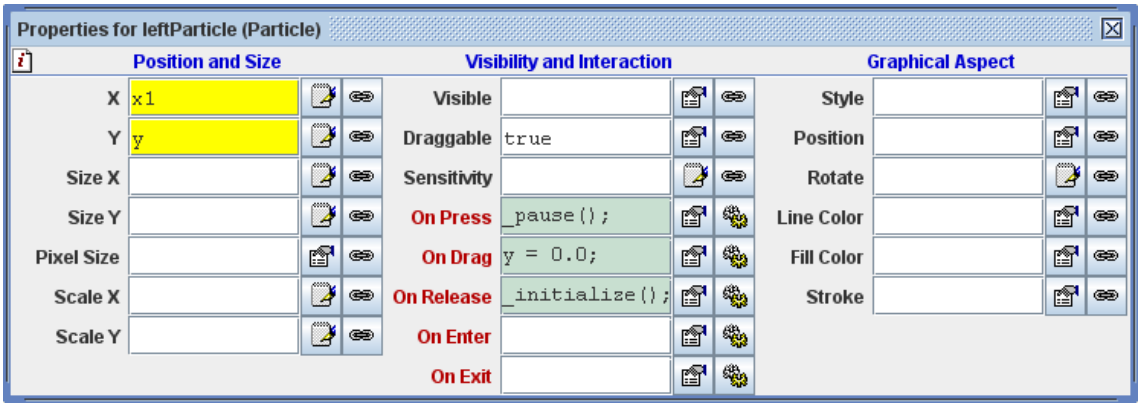
The *Up* position of *centerPanel* I soccupied by a drawing panel which will contain the particles and springs. We have adjusted its extremes so that the particles look like circles. The properties of the drawing panel and its children are the following:

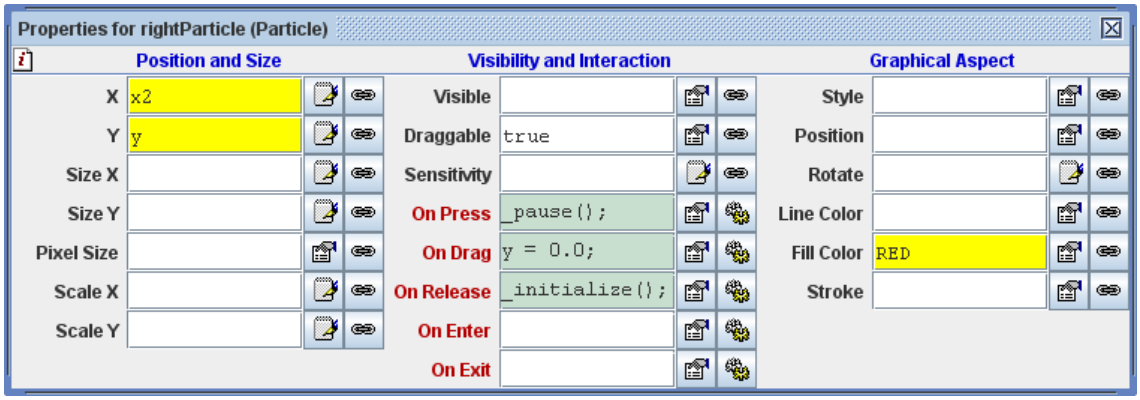
Note: The important figure in the *100,100* size is the second 100 which indicates the minimum height required by the panel. The parent *centerPanel* will stretch the drawing panel in the horizontal dimension as needed.





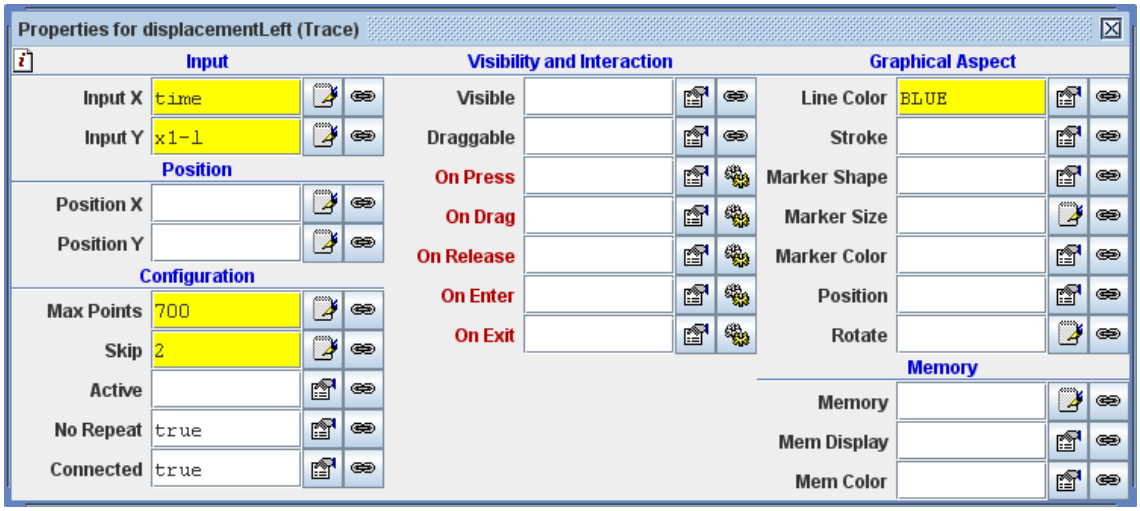


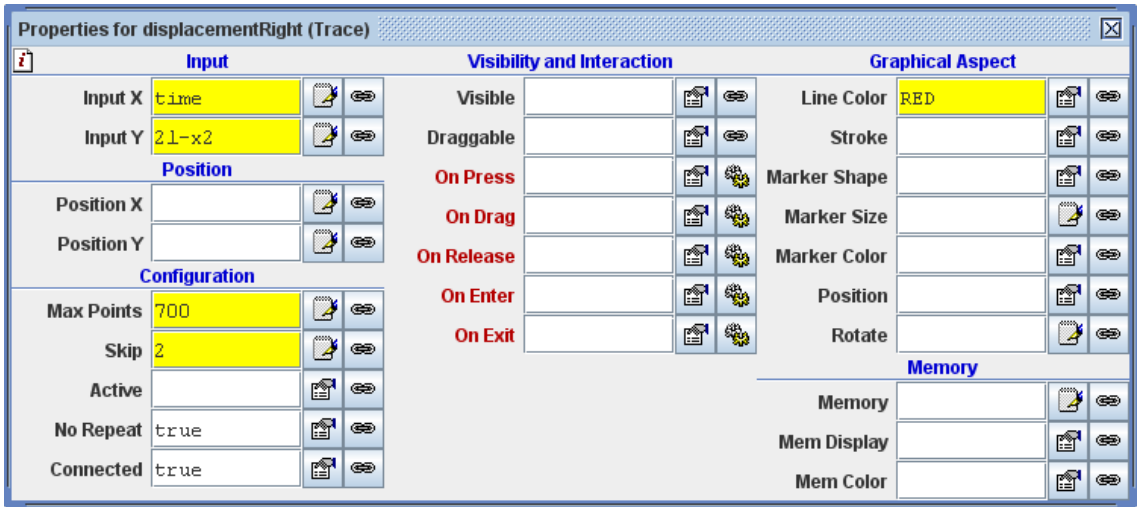




Notice the *Action* properties of both particles. The *On Drag* property is needed to make sure the particles do not move in the vertical direction.

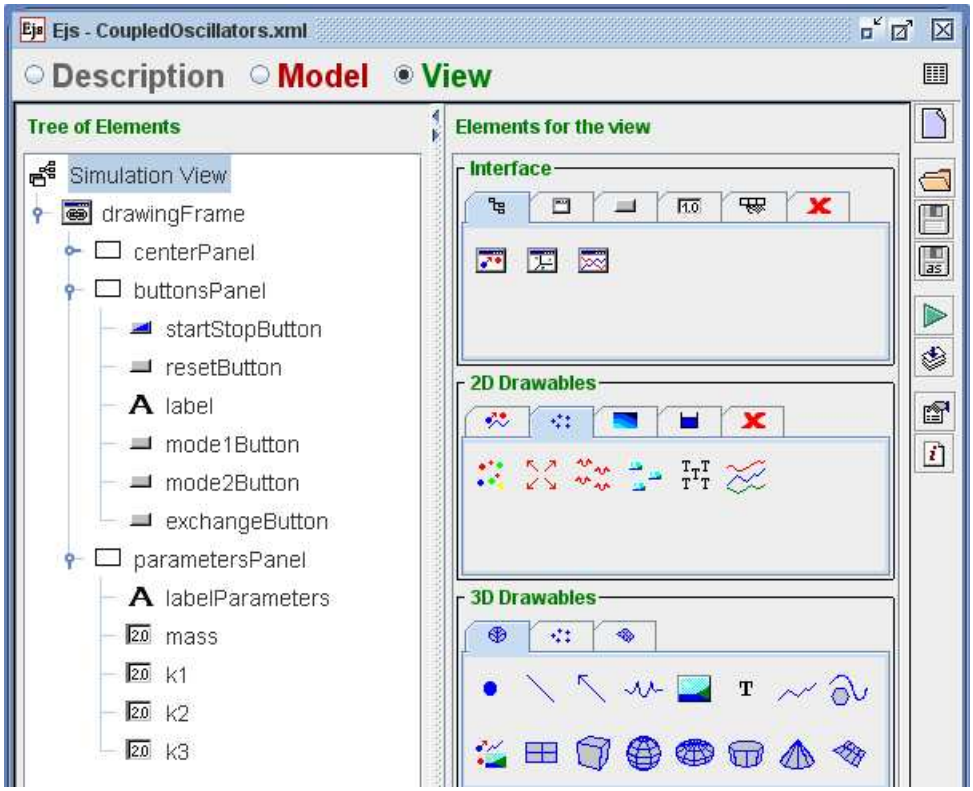
The plotting panel below the previous drawing panel occupies the center position of *centerPanel*. The plotting panel is rather standard, has autoscaling, and hosts two *Trace* elements. One for each of the displacements. The property panel of these are:





The *Skip* property is set to 2 so that the plots draw on point after two evolution steps.

We now show a detail of the *Tree of Elements* of the rest of the view



The elements used in these two panels are rather standard. It is interesting to see the *Action* properties of the *mode1Button*, *mode2Button*, and *exchangeButton* buttons. These contain the following code, respectively:

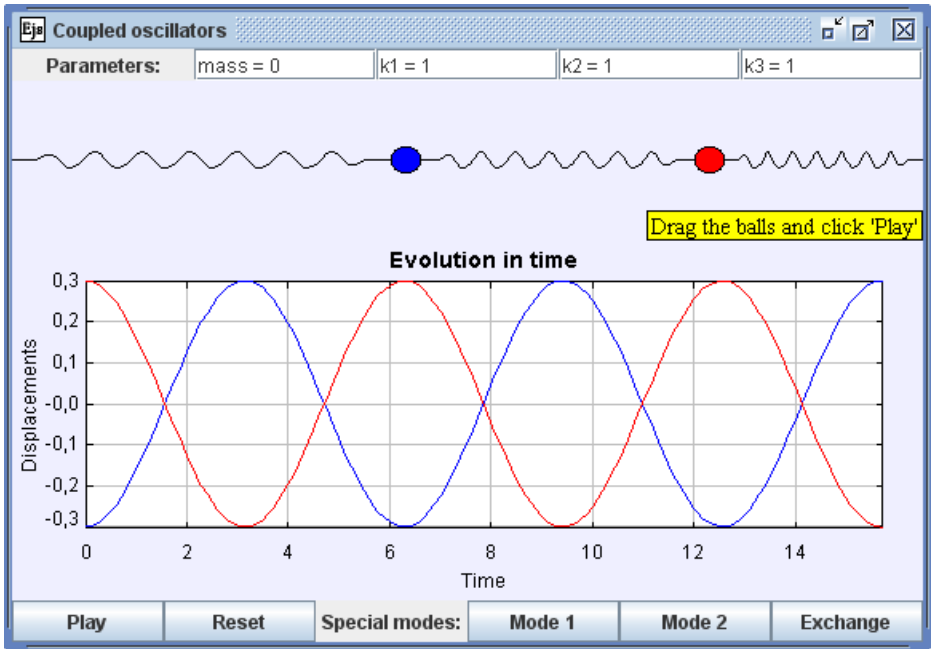
```
// Mode 1 button
x1 = 0.7*l;
x2 = 2.3*l;
k1 = k2 = k3 = 1.0;
initialize();

// Mode 2 button
x1 = 0.7*l;
x2 = 1.7*l;
k1 = k2 = k3 = 1.0;
initialize();

// Exchange of energies button
x1 = 0.7*l;
x2 = 2*l;
k1 = k3 = 1.0;
k2 = 0.1;
initialize();
```

Running the simulation

Here is a sample execution using the parameters and initial conditions of mode 2:



Author

Francisco Esquembre
Universidad de Murcia, Spain
July 2007