Artisynth: Probe Editor

A technical project report by
Johnty Wang 16234031
APSC 411

Compiled for Human Communication Technologies Lab
UBC Electrical and Computer Engineering

in Partial Fulfillment of the Requirements for:
UBC Applied Science Co-op Program

April 30, 2007

Table of Contents

List of Figures

Introduction

Project Introduction

Artisynth is a bio-mechanical modeling software package developed by the Human Communications Technology's Speech group at UBC. The project is open source and written in Java. The development is done mostly in Linux, but is tested for and runs in Windows and Mac OS as well.

Artisynth is designed to be a highly flexible model based toolkit with specific emphasis on modeling of the vocal tract and upper airway (tongue, jaw, etc). Key features of this application include the ease of adding new models, and the simulation speeds (close to real-time). Another important aspect of the application is its graphical user interface that exposes many of the features. This allows non-programmers to test their models with relative ease, and reduce the need for researchers in other fields to rely on programmers.

It has been highly rewarding to work in the Artisynth team, not only for the software engineering skills learnt and applied during the course of the work term. The team consists of people from a wide range of faculties and spans many different disciplines from electrical engineering to computer science, dentistry, and linguistics. Working as a co-op student has exposed me to many aspects of cross-disciplinary research and was definitely a worthwhile experience.

Work Term Duties

Most of my work term duties involved the development of parts of the graphical user interface for Artisynth. The most significant section of work was designing and implementing an editing window for numeric probes, which are a fundamental concept behind the simulation system in Artisynth. The rest of this document provides extensive detail regarding the layout and features of the probe editor, and will not only provide a report on my duties of the work term, but also present a useful reference for potential users of the Artisynth modeling environment.

Scope

This document is appropriate for users of Artisynth with some basic understanding of the major components of the system (more specifically, the main viewing window and the timeline display). The reader is also expected to be familiar with the usage of basic graphical user interface elements such as dialog windows, buttons, and text fields.

Motivation

The probe editor is designed to allow interactive modification and addition of input and output numeric probes to the timeline. Before this point, probes were manually defined in code. While coding probes is a sometimes necessary and useful feature, the ability to change or add probes on the fly during runtime would greatly increase the flexibility of probes, and this was the primary motivation behind the making of the probe editor.

Numeric Probes

Probes

There are two types of numeric probes in Artisynth: input and output probes. An input probe controls properties during simulation based on some input data, and the output probe displays them. Before describing probes, it is useful to know what components and properties are.

Components and Properties

Artisynth models systems using basic building blocks, components. Each component contains certain inherent properties that describe its physical characteristics. For example, a single point particle will have properties such as mass, velocity, force, etc. A spring component would have a rest length, spring constant, as some more examples. There could also be non-real life properties used by Artisynth, such as rendering controls that define the color used to display the component, or the line width used for the drawing of that component.

Numeric Input Probes

An input probe consists of some input data in the form vectors connected to the property or properties of a component or components. The data is sampled at discrete time points, and applied to the connected property as the timeline progresses during simulation. The simplest example of an input property appears in the Spring Mesh model: In this case, a single input vector of size 3 is loaded from a data file. This data is connected to the "position" property of component 0. When the probe is applied, the data values from the input are applied to the X, Y, and Z components of the particle. During simulation we see the particle's position in space change according to the probe data. Please refer to figure 1 for a graphical display.

Numeric Output Probes

A numeric output probe contains some of the basic properties of the input probe. It too contains component properties, and their association with data. However the key difference is that while an input probe is designed to drive and control a component's property, the sole purpose of an output probe is to display the property information during simulation.

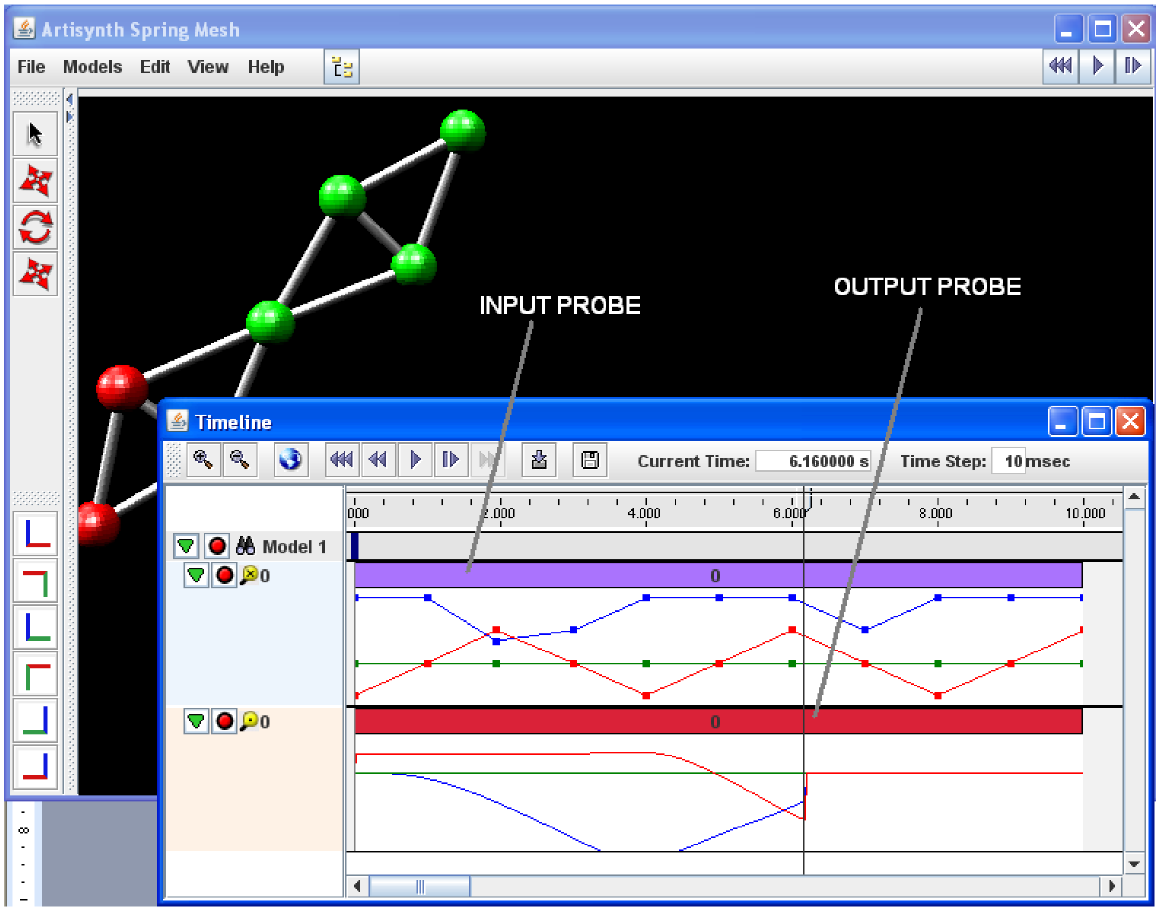


Figure 1:Input and output probes

As shown in the above figure, there is an input and output probe present in this model. The input probe in this case was driving the position of a particle in the model, while the output probe displayed the position of another. The components that were not driven by the input probe behaved according to their modeled properties (in this case, as masses and springs responding to gravity and spring forces).

The above example is a simple showcase of how probes may be used. In more complicated models, input probes can drive muscle activations, rendering properties, or any inherent property of a component, as long as it contains a numerical value.

In the above model, the probes present were coded in a file and loaded as the model is initiated. However, it was not possible for the user to add a new probe, or to edit the existing ones after they were created. This was the main motivation behind the probe editor: to allow the user to interactively add new probes, or to modify the existing ones.

Numeric Probe Editor

This section provides a detailed overview of the functionality of the probe editor.

A good editor must be able to access and modify all the useful contents of the object it is attempting to modify. A numeric probe contains the following information:

- A list of properties it is attached to
- A list of input/output vectors, and their dimensions
- A filename the probe data is associated with (For an input probe, this is the data that will be used to drive properties; For an output probe, it is a file where the plotted data could e exported to)
- The probe name
- Start and Stop times (duration which the probe is active)
- Scaling factor (for an input probe, this defines how much the data is scaled by before applied to a probe).

In addition, to allow for a level of flexibility that was absent from the previous model, a feature was added to allow a property to be driven by more than one set of data, and for the data to be conditioned before application. This introduced an extra field that allowed the user to input a “formula” that conditions input data before it is applied to the property. The resultant probe editor was a panel that allowed the user to access and modify all the above information, and to present the data in a logical, intuitive manner.

Input Probe Editor

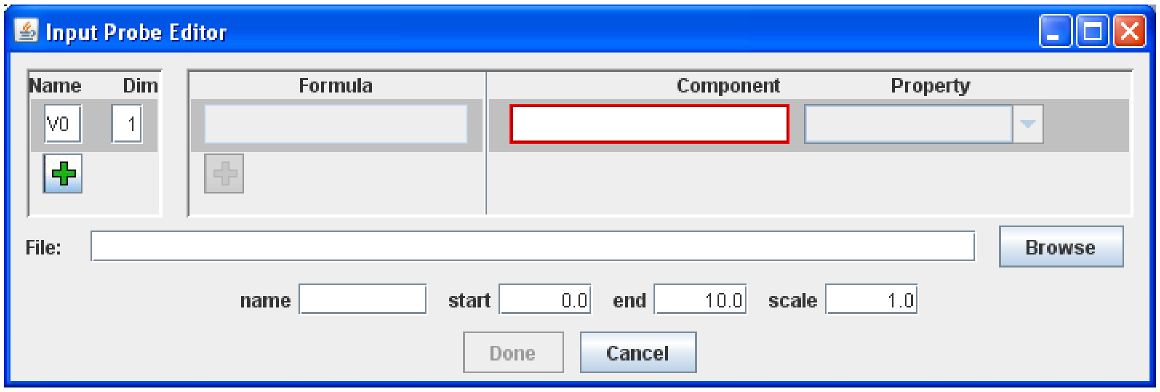


Figure 2: Input probe editor

The above figure shows an input probe editor. The overall layout of the editor is a dialog panel that contains text boxes, drop down menus and buttons. The panel contains enough information to fully specify the contents of a numeric probe (as defined at the time of writing of this article). The top section of the dialog contains (from left to right) the input vector(s) name and size, the formula used to condition the input, and then the component property to be driven by the input. The + shaped button allows the addition of more vectors and properties. The bottom half of the window contains various other information that defines a probe. The file and name fields are not mandatory, while the rest are filled out by default values.

Using the Input Probe Editor

Upon opening of a new probe editor, the following is displayed:

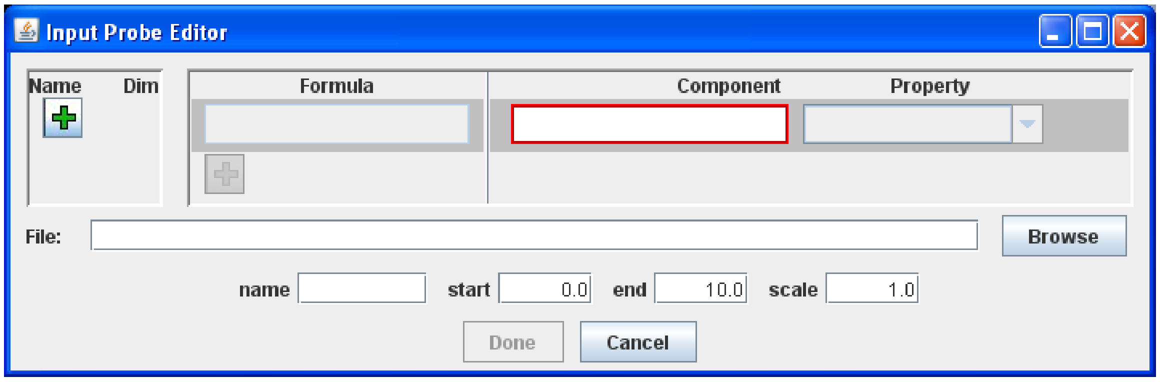


Figure 3: Blank input probe editor

Note that there are no vectors present when a new probe editor is opened. The "Done" button is grayed out until there is enough data entered to define a valid input probe. The first step in using the editor is usually to insert a property to be driven by the probe.

Component/Property Selector

To select a component and its property, we focus on the following section of the window:

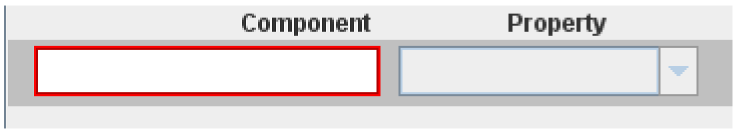
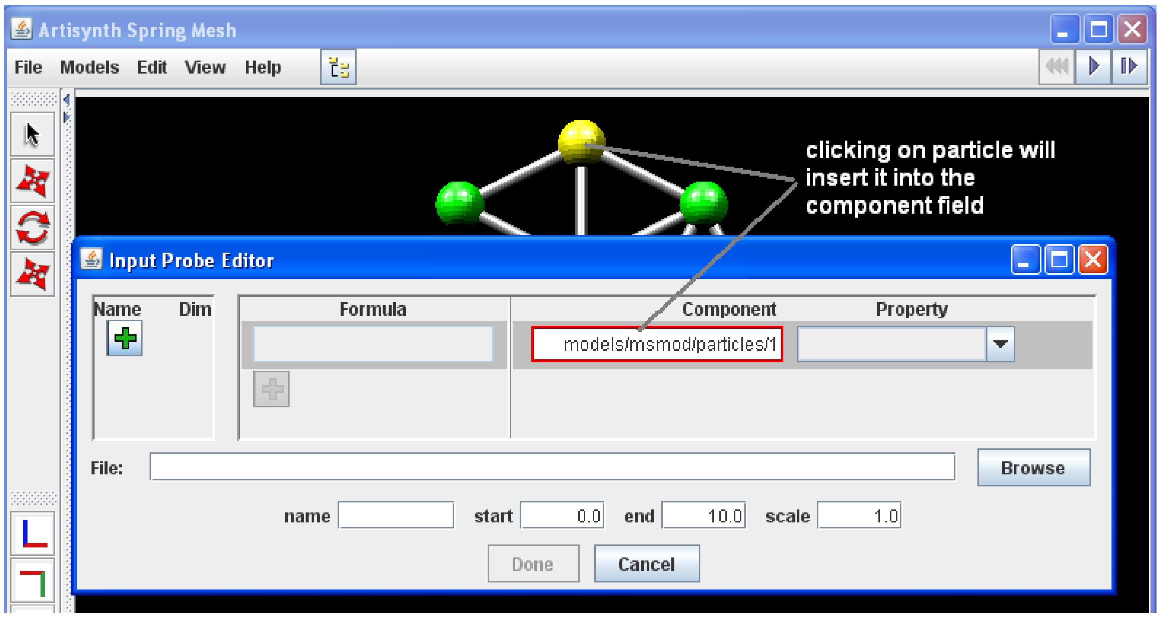


Figure 4: Component/property selector

The left component is a text field, while the right is a drop-down combo box. The red band around the component field signifies that it is currently active. By default a newly added component/property selector will be active while all others will be turned inactive. To change the active status, one can click anywhere within component field. An active component field is an indication that it will receive the path of the next selected component in the viewer.



Component: models/msmod/particles/1

name

start 0.0

end 10.0

scale 1.0

Figure 5: Selecting a component in the viewer

Another way to insert a component is to type its full path into the text field. Once a valid component has been inserted into the field, the Property combo box will be populated with a list of the properties of the component:

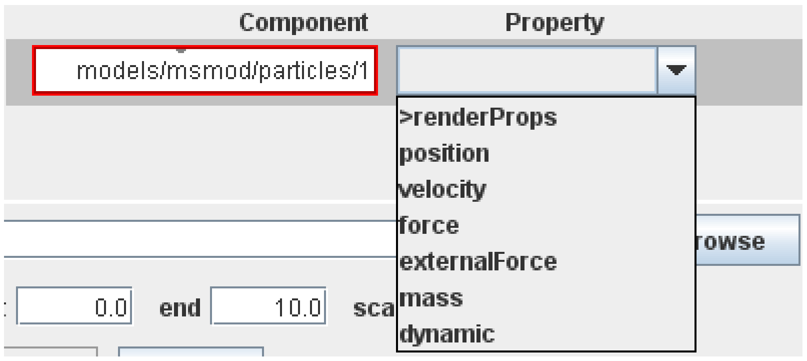


Figure 6: Properties of a component

Note the property name with a ">" in front. This signifies that the property does not contain numeric values itself, but instead owns child properties. Clicking on such a property would display all the child properties owned by the parent property.

Once a valid numeric property has been selected, an input vector of the corresponding size is automatically generated, and the dimensions of the property displayed on the right side of the combo box:

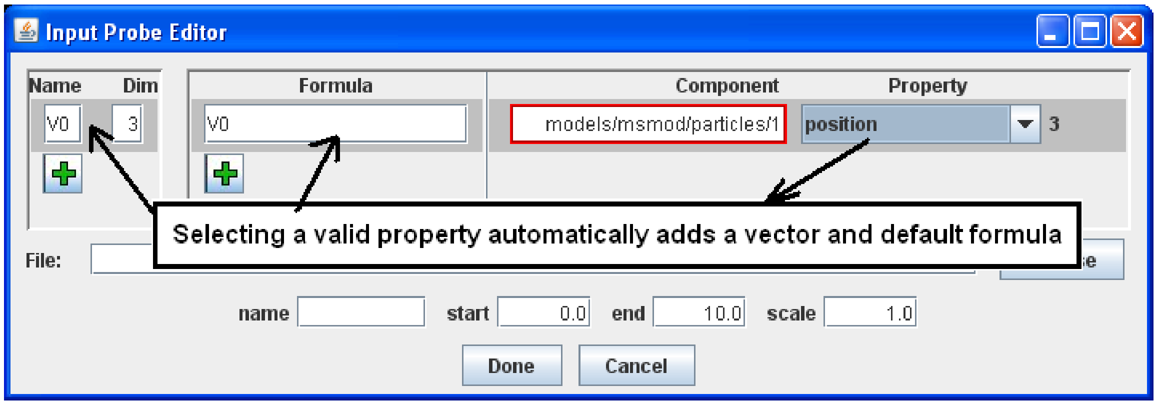
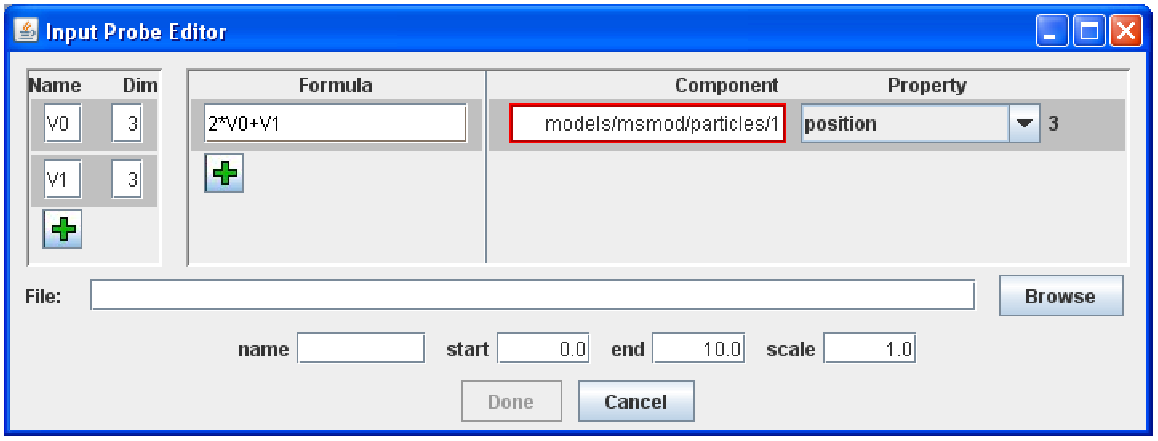


Figure 7: Property selection

At this point, a numeric input probe is fully defined. Note the default formula that is inserted. Note that the "Done" button is enabled. By clicking on this button, a new input probe will be added. This example shows the most basic input probe. There is a simple direct relationship between the single input vector and the property. One could modify this equation with mathematical expressions, and drive the property with more than one input. For example:


Input Probe Editor

Name	Dim
V0	3
V1	3

Formula	Component	Property	
2*V0+V1	models/msmod/particles/1	position	3

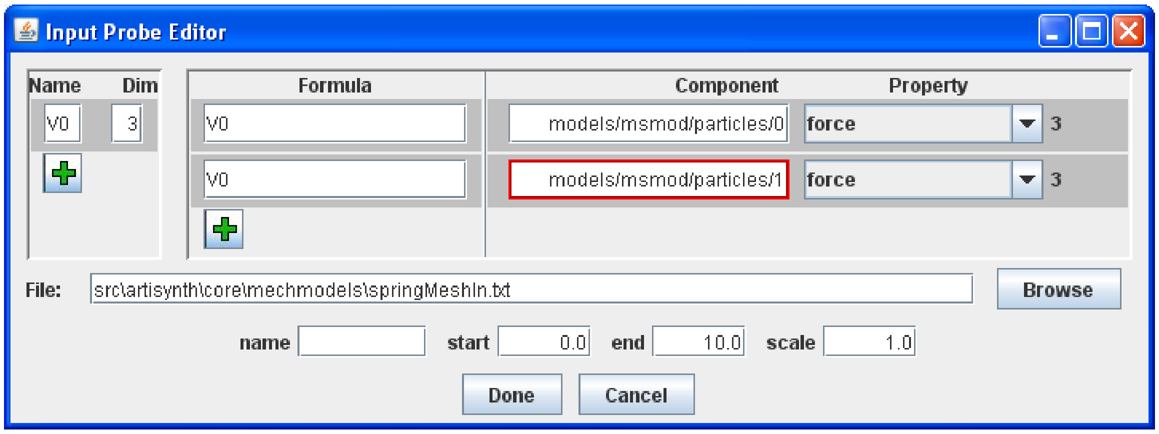
File:

name | start 0.0 | end 10.0 | scale 1.0

Figure 8: 2 Input vectors driving a single property

The above figure shows an input property driven by a linear combination of two inputs.

Another possible case would be:


Input Probe Editor

Name	Dim
V0	3

Formula	Component	Property	
V0	models/msmod/particles/0	force	3
V0	models/msmod/particles/1	force	3

File: src\artisynth\core\mechmodels\springMeshIn.txt

name | start 0.0 | end 10.0 | scale 1.0

Figure 9: 1 Input driving two properties

Here we see a single input vector driving the force property of two separate components. The key points to remember are the following:

- Each property has its own “Formula” field. This is what drives the property
- The formula field can contain any valid mathematical expression involving 0 or more input vectors, as long as the output size is consistent with the size of the property being driven. For example, a dot product of two input vectors of dimension 3 would have a dimension of 1, and therefore could not be used to drive a property of size 3.

The formula allows a high degree of flexibility over the controlling of properties that was impossible to achieve in the past. It would have been possible to manually code in such equations and relationships, but the amount of effort involved would be large and beyond the scope of most users, and tedious for testing since each change would require a recompilation and restart of the program. This created a strong dependence of the user on programmers who are familiar with the code.

Sources of Input

So far, no mentioning has been made regarding the source of the input files. The actual input data values can be acquired in two ways. First, by specifying a filename in the probe editor, one could specify the source of the data.

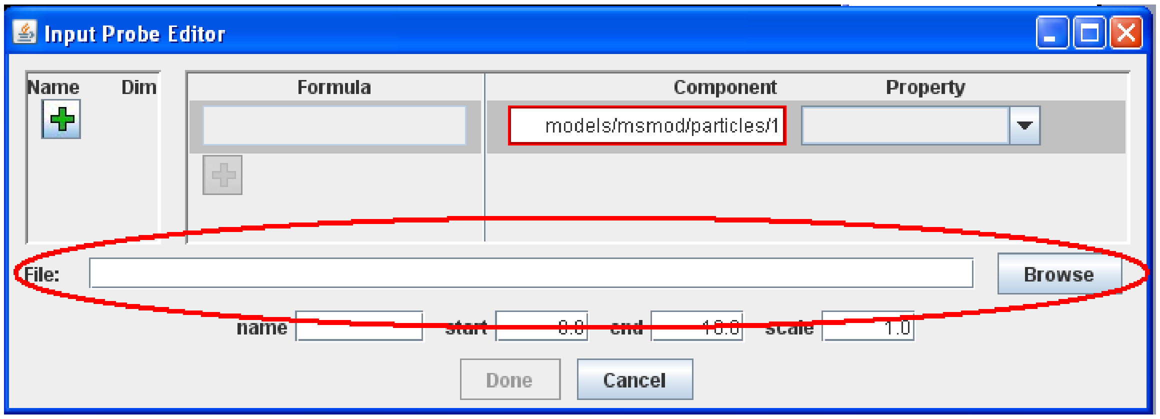


Figure 10: Input probe data file

If no file is specified, the probe will be created with 0 valued knot points. Below shows two input probes, one with a specified input file and the other without:

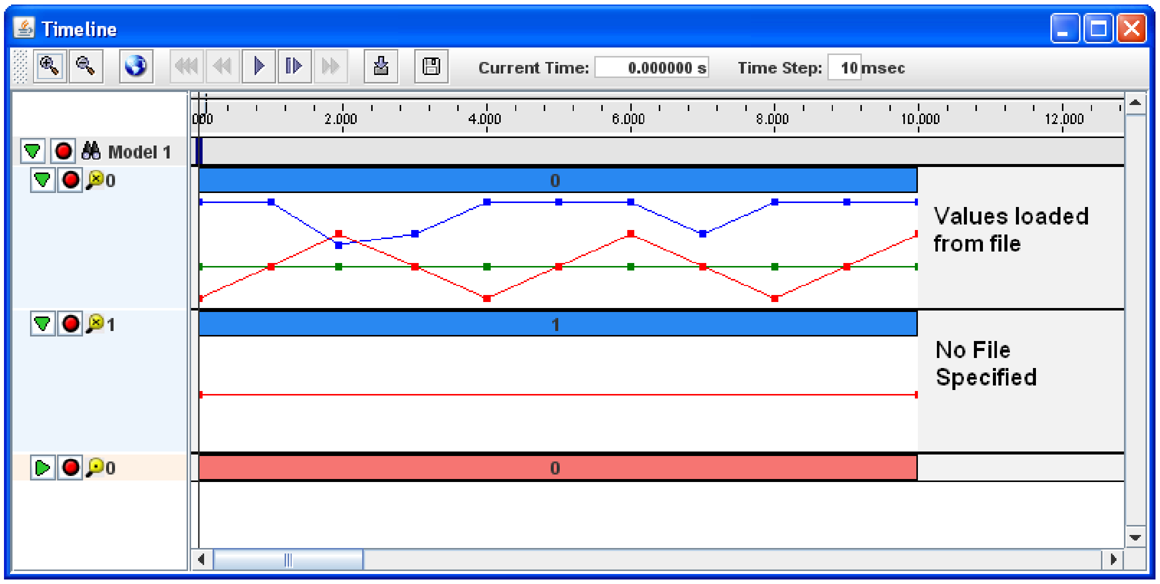
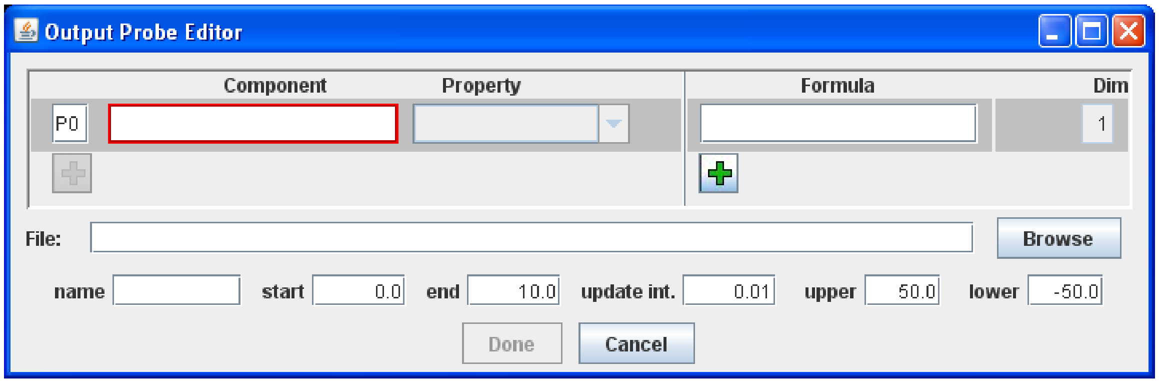


Figure 11: Input probe with no file

For the "empty" probe, one can add knot points by double clicking on the plot line. The values of a knot point can be edited by dragging the knot points.

Output Probe Editor

The output probe editor works in a very similar manner to the input probe editor. The key difference is that properties are the source of the data, and the vectors are the outputs. Here the formula conditions the output data with the values of properties as input. The usage of each graphical component is exactly identical compared to the input probe editor.



name

start 0.0

end 10.0

update int. 0.01

upper 50.0

lower -50.0

Figure 12: Output probe editor

Probe Editor Access

Opening

There are two ways to open up the probe editor. The first way is to add a new probe. This is done by clicking on the “edit” menu, and then “add input probe” or “add output probe”. The second way is to edit an existing probe in the timeline. This is done through the right click context menu.

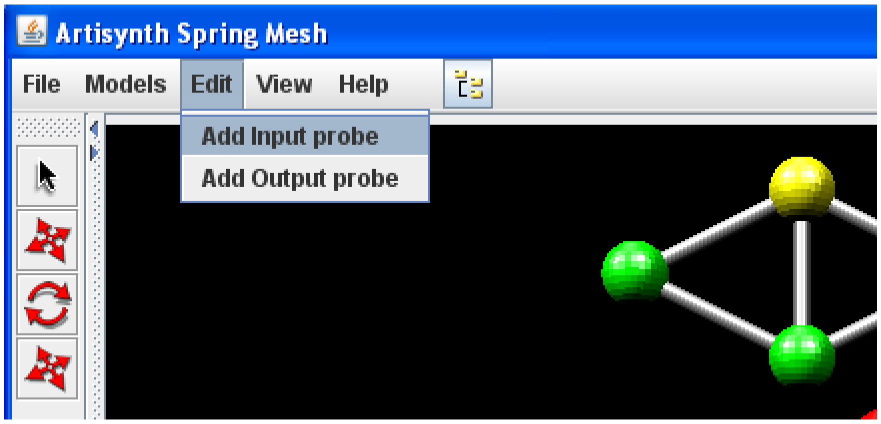


Figure 13: Opening the probe editor

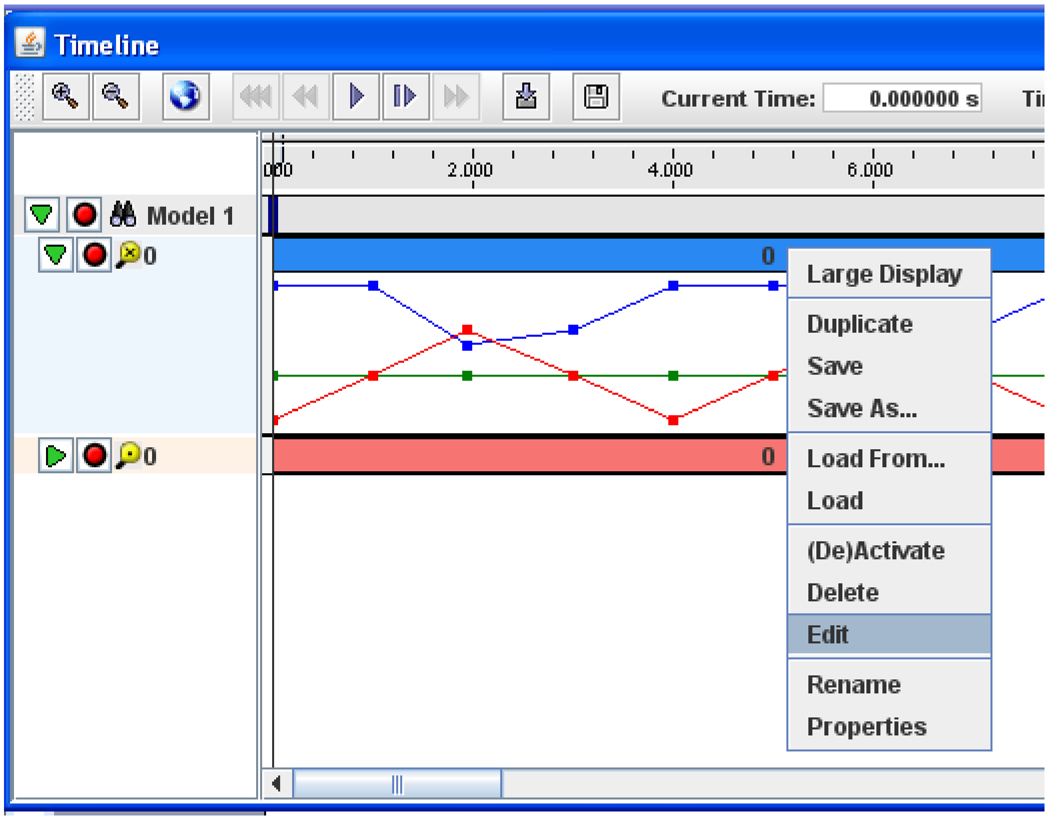


Figure 14: Editing an existing probe

Closing

The panel can be closed at any time by pressing the “Cancel” button, or the “X” at the top right of the window. Additionally, the panel will be automatically closed when a new model is loaded into Artisynth.

Conclusion

The probe editor is a very important addition to the features of Artisynth. It allows users to interactively create and modify input and output probes, which greatly simplifies the process and allows non-programming researchers to quickly and easily perform modifications to their tests without having to rely on a programmer.

The challenge of creating such a user interface is to provide all the required features to access the underlying data structure and exposing them in a logical and intuitive manner. Additionally, since a graphical interface can be highly non-linear during use, a lot of error checking must be done in the background to ensure the consistency of user-entered data, and provide the necessary corrections and output hints to ensure predictable behavior of the system.

Appendix

Quick-start Guide

This is a brief tutorial that demonstrates the usage of the numeric probe editor in Artisynth.

- Load the Artisynth Spring-mesh model from the “Models” drop-menu.
- Click on the “Edit” menu, select “Add Output Probe”. The output probe editor will appear. Note that a blank component and property field, with a default label is created automatically upon initialization of the editor.
- Select a component that contains a numeric property (e.g. any of the particles in the model). At this point the path of the component will appear in the component path. If required, the component path can be manually changed at this point by clicking inside the text field and editing the value.
- Once the desired component has been selected, click on the “Property” drop-down combo box to select the property to be connected to the output probe. (e.g. the “position” of a particle)
- Click on “Done” to add the output probe into the timeline.