5 User Interface and Graphical Windows

The NSL graphical user interface provides interactive simulation control by means of the NSL Executive Window and different types of input and output displays customized for every model. Each model may include a number of *protocols* corresponding to different experiments involving different sets of input, parameters and graphics displays. In a well-written model, every model experiment should correspond to one of these protocols. Sometimes, however, models only come with scripts that must be read via the Script Window and sometimes they only come with "README" files that describe the proper script command sequences that should be issued to get the different results. (The script language is described in chapter 7.) The NSL graphical user interface is designed to provide an environment that protects as much as possible the novice model user from having to type too many commands. At the same time, the graphical interface provides flexibility for the advanced model builder to experiment with multiple simulation options in analyzing model results.

Ideally every model executes with just selecting one of the protocols from the Protocol menu, and then selecting the Simulation, TrainAndRunAll menu item.

5.1 NSL Executive User Interface

When NSL is first invoked, the NSL Executive Window is displayed as shown in figure 5.1. From the Executive window the user controls the simulation and brings up other display windows or frames.

Figure 5.1
The NSL Executive Window controls model simulation. It includes menu options as well as a script window for written commands.

The top portion of the NSL executive window corresponds to the "Window Control" section containing a title and three buttons for window control. Underneath the "Window Control" the "Executive Menu Bar" contains five menu buttons: **System, Edit, Protocol, Simulation, Train, Run, Display,** and **Help**, discussed in the following sections. The bottom portion of the window corresponds to the "Script Window" allowing the user to interactively type commands. Any command that can be selected from one of the menu pull-downs can also be typed in the Script Window as well as stored in files for later retrieval. Script commands are discussed more thoroughly in chapter 7.

System Menu

The **System** menu contains commands related to general system aspects. In particular, the menu contains the following items:

- **Source**—to read NSLS model scripts.
- **Set**—to read or modify system parameter values.
- **Exit**—to stop the simulation, close all windows, and exit.

Edit Menu

The **Edit** menu allows the user to select text from the script window, copy text from the script window into the clipboard or paste text into the script window from the clipboard. These commands are quite handy when importing commands to the script window or saving script commands into another file. The menu contains the following items:

- **Select**—select text from the script window.
- **Copy**—copy text from the script window into the clipboard.
- **Paste**—paste text from clipboard into the script window.

Protocol Menu

The **Protocol** menu specifies the *protocols* or experiments included with each model. Protocols are model driven, i.e. the modeler decides which protocols should be placed in the menu. The default "manual" protocol is always provided allowing the user to write script commands to set up stimuli and parameters for the simulation. Protocols are an easy way for the model builder to setup customized input and output windows to handle model input and output respectively. This customization takes place directly in NSLM. The protocol menu contains the following items:

- **Manual**—default user specified input by means of scripts.
- ***Additional Protocols***—Modeler define additional protocols (none by default).

Simulation Menu

The **Simulation** menu contains different options to control the general aspects of the simulation, such as setting up system variables and global initialization of the modules (see chapter 6, The NSLM Language). The following menu options exist:

- The **InitSys** menu item executes the *initSys* method for every module.
- The **InitModule** menu item executes the *initModule* method for every module.
- The **TrainAndRunAll** menu item executes the initialization of the training epochs, all of the epochs for the training phase of simulation as well as the initialization of the run epochs and all of the epochs for the run phase of simulation.
- The **EndModule** menu item executes the *EndModule* method of every module.
- The **EndSys** menu item executes the *EndSys* method of every module.

Train Menu

The **Train** menu item executes all the methods necessary to train the model. The menu contains the following options:

- The **InitTrainEpochs** menu item executes all of the "initTrainEpochs" methods for all modules.
- The **InitTrain** menu item executes all of the "initTrain" methods for all modules.
- The **SimTrain** menu item executes all of the "simTrain" methods over and over again until "system.trainEndTime" is reached. The system.trainEndTime is the system variable that specifies how long the training process should last. Based on the training delta used, the system.trainEndTime is used to calculate the number of cycles for each epoch.
- The **EndTrain** menu item executes all of the "endTrain" methods.
- The **EndTrainEpochs** menu item executes all of the "endTrainEpochs" methods.
- The **Train** menu item executes one epoch that includes the execution of initTrain, simTrain for the number of cycles or steps specified, and finally endTrain.
- The **DoTrainEpochTimes** menu item executes the initTrainEpochs method for all modules, then executes the initTrain, simTrain (repeated for n cycles), and endTrain methods for however many training epochs have been specified with system.numTrainEpochs. And finally, it executes the endTrainEpochs method for all modules.
- We can execute the **Break** command to stop the simulation between cycles. We can then use the "continue" menu option to continue the simulation.
- We can execute the **BreakModules** command to stop the simulation between modules.
- We can execute the **BreakCycles** command to stop the simulation between cycles.
- We can execute the **BreakEpochs** command to stop the simulation between epochs.

- The **Continue** menu item continues the simulation from the last break point. It then executes all of the "simTrain" methods over and over again until "system.trainEndTime" is reached.
- The **ContinueModule** menu item continues the simulation from the last break point. If the last break was with BreakModules, then it continues with the next module in the scheduler. It then executes all of the "simTrain" methods over and over again until the last module in the scheduler is executed.
- The **ContinueCycle** menu item continues the simulation from the last break point. If the last break was with BreakCycles, then it continues with the next cycle. It then executes all of the "simTrain" methods over and over again until "system.trainEndTime" is reached.
- The **ContinueEpoch** menu item continues the simulation from the last break point. If the last break was with BreakEpochs, then it continues with the next epoch. It then executes all of the epochs over and over again until **numTrainEpochs** is reached.
- The **StepModule** menu item executes the "simTrain" method of the next module in the scheduler.
- The **StepCycle** menu item executes all of the "simTrain" method once for each module in the scheduler.
- The **StepEpoch** menu item executes one epoch that includes the initTrain method, the simTrain methods for however many cycles are specified, and the endTrain method.

Run Menu

The **Run** menu item executes all the methods necessary to run the model. The menu contains the following options:

- The **InitRunEpochs** menu item executes all of the "initRunEpochs" methods for all of the modules
- The **InitRun** menu item executes all of the "initRun" methods for all of the modules
- The **SimRun** menu item executes all of the "simRun" methods over and over again until "system.runEndTime" is reached. The system.runEndTime is the system variable that specifies how long the Running process should last. Based on the Run delta used, the system.runEndTime is used to calculate the number of cycles for each epoch.
- The **EndRun** menu item executes all of the "endRun" methods.
- The **EndRunEpochs** menu item executes all of the "endRunEpochs" methods.
- The **Run** menu item executes one epoch that includes the execution of initRun, simRun for the number of cycles or steps specified, and finally endRun.
- The **DoRunEpochTimes** menu item executes the initRunEpochs method for all modules, then executes the initRun, simRun (repeated for n cycles), and endRun methods for however many training epochs have been specified with system.numRunEpochs. And finally, it executes the endRunEpochs method for all modules.
- We can execute the **Break** command to stop the simulation between cycles. We can then use the "continue" menu option to continue the simulation.
- We can execute the **BreakModules** command to stop the simulation between modules.
- We can execute the **BreakCycles** command to stop the simulation between cycles.
- We can execute the **BreakEpochs** command to stop the simulation between epochs.
- The **Continue** menu item continues the simulation from the last break point. It then executes all of the "simRun" methods over and over again until "system.RunEndTime" is reached.
- The **ContinueModule** menu item continues the simulation from the last break point. If the last break was with BreakModules, then it continues with the next module in

the scheduler. It then executes all of the “simRun” methods over and over again until the last module in the scheduler is executed.

- The **ContinueCycle** menu item continues the simulation from the last break point. If the last break was with BreakCycles, then it continues with the next cycle. It then executes all of the “simRun” methods over and over again until “system.RunEndTime” is reached.
- The **ContinueEpoch** menu item continues the simulation from the last break point. If the last break was with BreakEpochs, then it continues with the next epoch. It then executes all of the epochs over and over again until **numRunEpochs** is reached.
- The **StepModule** menu item executes the “simRun” method of the next module in the scheduler.
- The **StepCycle** menu item executes all of the “simRun” method once for each module in the scheduler.
- The **StepEpoch** menu item executes one epoch that includes the initRun method, the simRun methods for however many cycles are specified, and the endRun method.

Display Menu

The **Display** menu contains commands to control output and input display window creation. The display options in the menu are,

- **NslOutFrame** used to create a frame to display results of model variables,
- **NslInFrame** used to create a frame to control input stimulus and model parameters.

The two frame types are described in the following sections.

Help menu

The **Help** menu retrieves help on any command. It contains three types of help: “How To”, “Command Help”, and “Setup”.

5.2 NslOutFrames

Upon selection of a new **NslOutFrame** from the executive window, a **NslOutFrame** will appear with a long name in the form of “.nsl.*frameNameX*” where the “.nsl” comes from the fact that all windows are actually subwindows of the **NslExecutiveWindow** associated with “.nsl” prefix. The *frameName* is set to “OutModule” when selecting an output frame (“InModule” when selecting an input frame). The *X* is assigned to an incremental integer number resulting in names such as “.nsl.OutModule2”. A popup window will appear first requesting a protocol name to be associated with the new **NslOutFrame**, as seen in figure 5.2. Note that if no protocols exist for the model then only the “manual” option on the right hand side will appear.

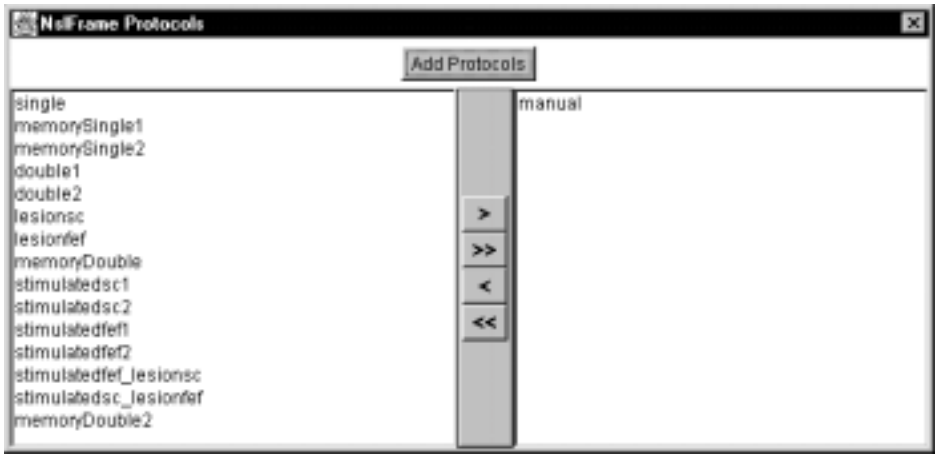

Figure 5.2
Popup Window for Adding a New NslOutFrame

Once the OutputFrame or more appropriately, the OutputModule has registered for certain protocols, a new window is instantiated as shown in figure 5.3. At the top is the **title** or frame name. Underneath the frame's title is the **frame's menu bar**. In this menu bar we have the **Frame**, **Canvas**, and **Help** menus, as follows,

- **Frame**—The **Frame** menu items are responsible for changing the items and attributes of the frame.
- **Canvas**—The **Canvas** menu items are responsible for changing the items and attributes of a selected canvas.
- **Help**—The **Help** menu displays information on any command.

In the middle of the frame is the **drawing area** or the place where canvases can be placed. And at the bottom of the frame is the **status bar**.

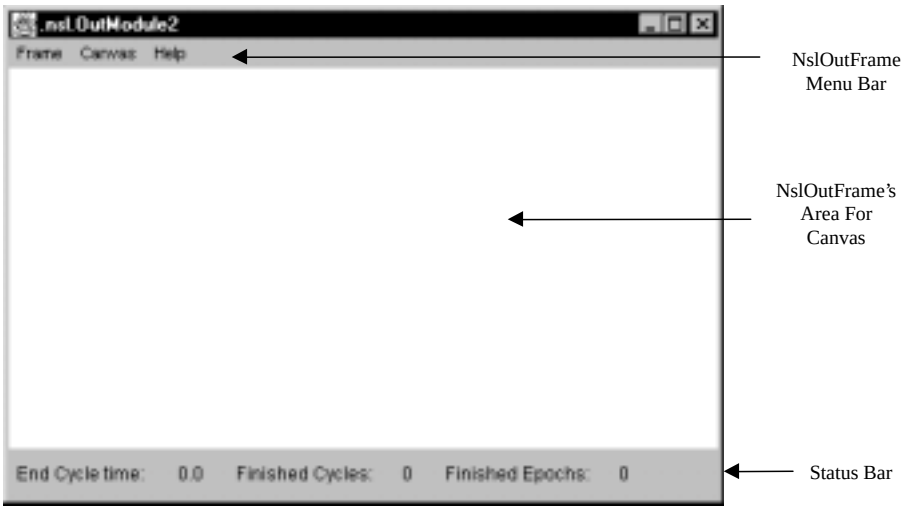

Figure 5.3
The NslOutFrame without any canvases

The NslOutFrame's Frame Menu

A **NslOutFrame** is basically a container for **NslOutCanvases** displays of NSL variables. All NSL variables that have been declared to have either "Read" or "Write" visibility can be displayed in a **NslOutFrame**. The **NslOutFrame** contains a menu bar for adding new canvases/variables and for manipulating the canvases, as shown in figure 5.4.

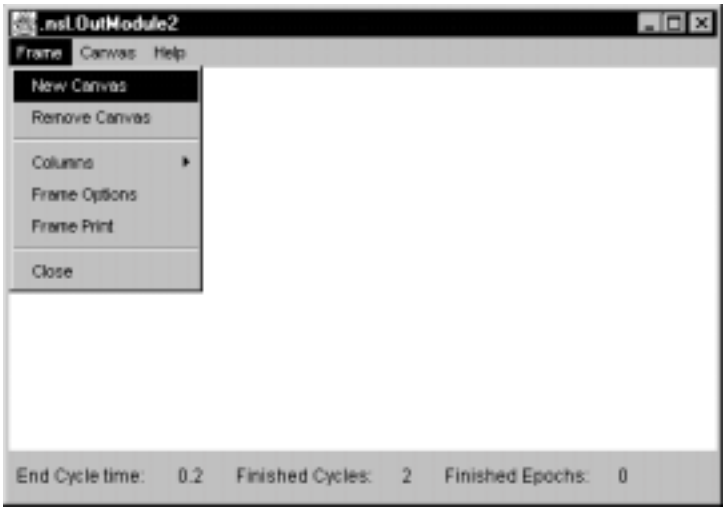

Figure 5.4
NslOutFrame's Menu

The **Frame** menu contains commands necessary to change the contents and attributes of the frame. These commands are **New Canvas, Remove Canvas, Columns, Frame Options, Frame Print,** and **Close**. The following describes these commands.

Create a New Canvas Containing a Plot of a NSL Variable

To add a variable display to an existing **NslOutFrame**, you go to the **NslOutFrame**'s menu bar and select "Frame→New Canvas" as shown in figure 5.4. A popup window as shown in figure 5.5 will appear.

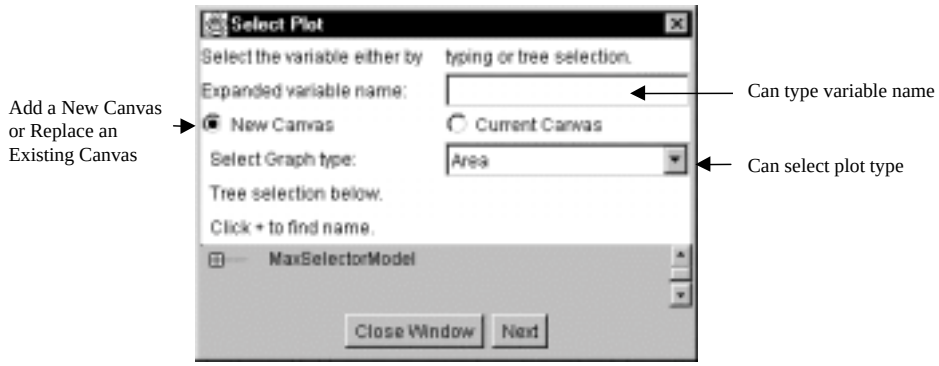

Figure 5.5
Add New or Change Current Canvas-Popup Window

At the top of the window, we have the choice of typing a full path variable name, or we can select the variable by tracing down the hierarchy tree in the lower gray area of the window. In this example, we will select the variable from the hierarchy tree. To do this, we first click on the little "plus sign" icon next to the word "**MaxSelectorModel**". Next we click on the "**plus sign**" icon next to the word "**stimulus**", and then the plus of **s_out**. At this point the tree should look like that in figure 5.6.

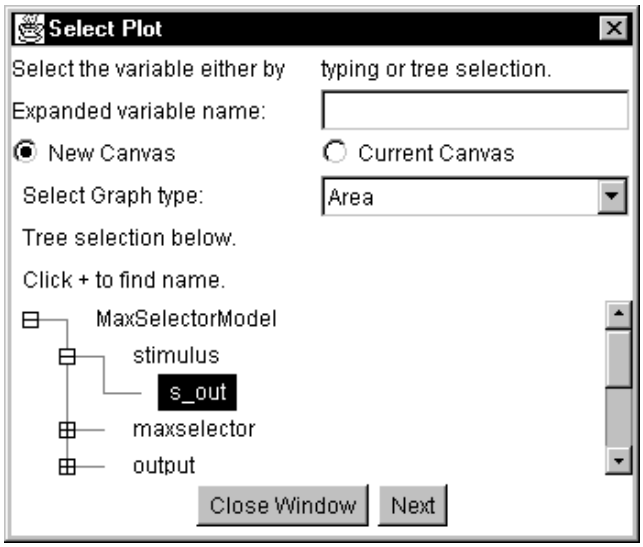

Figure 5.6
New Canvas or Change Current Canvas Popup Window with selection of s_out

To add the selection to the **NslOutFrame**, we select the **Next** button at the bottom of the window. We repeat the process again this time selecting a different type of plot. In the "Graph Type Selector" we change the graph or plot type to "**Temporal**". (The plot types are **Area**, **Bar**, **Dot**, **Spatial**, **String**, **Temporal**, **AreaColor**, **MultiTemporal** and **XY** being described in the section titled "Output Graph Types".) This time we expand the plus sign next to the **maxselector** module, then **u1**, then **up**. The window should look like that in figure 5.7.

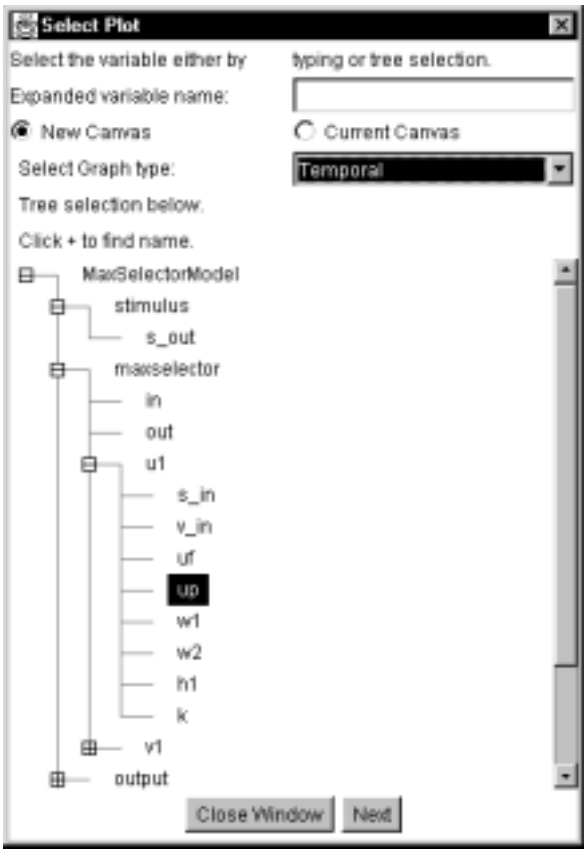

Figure 5.7
New Canvas or Change Current Canvas Popup Window with selection of up.

We select the **Next** button to add **up** to the NslOutFrame, and then we add **uf.** To add **uf,** we change the graph type back to **Area** and select the NSL variable "maxselector.u1.uf". After we have done this, we select the "**Next**" button at the bottom of the window. The resulting canvases are shown in figure 5.8.

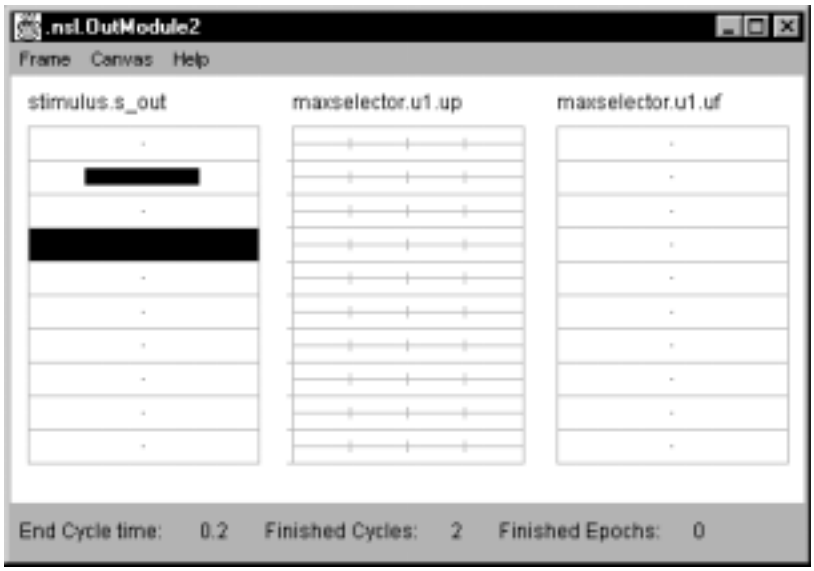

Figure 5.8
An example of a NslOutFrame that contains a menu bar, drawing area, and three NslOutCanvases corresponding to the variables stimulus.s_out, maxselector.u1.up, and maxselector.u1.uf.

Selecting and Deselecting a Canvas from a Frame

We can select a specific canvas in a frame by moving the mouse to the desired canvas and then clicking the left mouse button on it. The canvas will become highlighted in the

default highlighting color. Once selected there are a number of operations that can be applied to the canvas as will be seen later on such as changing the type of graph or variable being displayed. To deselect the canvas either click the mouse's right button or select another canvas.

Deleting a Canvas from a Frame

To delete a canvas from a frame, first select the canvas and then choose "Frame→Delete Canvas" from the Frame's menu.

Modifying the Number of Rows and Columns in a NslOutFrame

To modify the number of rows or columns **NslOutFrame** displays, select "Frame→Columns" and then the number of columns desired. This will also affect the number of rows displayed since if we display 8 canvases, we can display them in 1 row with 8 columns, or 2 rows with 4 columns, or 4 rows and 2 columns, etc.

Positioning and Resizing a Canvas

To change the position of a canvas, you need to delete it first and then re-add it in the desired location. To resize a canvas we can only make the frame larger or smaller. Currently all canvas sizes are the same in every frame as seen in figure 5.8.

Changing Options that Effect All Canvases

To change the update time, the starting graph time, ending graph time, the vertical minimum, vertical maximum, the default colors (background, grid color, drawing color) in every canvas currently displayed or later instantiated in this frame, select the "Frame→Frame Options" menu item. This will cause the popup window shown in figure 5.9. to appear. The "Apply to Future" button causes the canvases that are created in the future to have these default properties. The "Apply to All" button causes all of the current and future canvas to have these properties. And the "Cancel" button takes no action.

Figure 5.9
The NslOutFrame's Options for Properties Popup Menu

Printing a Frame

To print a **NslOutFrame** or to save an image of a **NslOutFrame** in any of the supported formats[1] select the **NslOutFrame**'s "Frame→Frame Print" menu item. To print one of the **NslOutCanvases**, first select that canvas and then select the "Frame→Print option". A print popup window will appear. The look of this window varies depending on the environment.

Closing a NslOutFrame

To close a **NslOutFrame**, simply select the "Frame→Close" option. This will close the frame, but will leave the simulation still executing. To exit the NSL System, select "System→Exit" from the Executive window.

The NslOutFrame's Canvas Menu

All of the Frame's Canvas Menu items pertain to changing the properties of a particular canvas. A canvas must first be selected with the mouse. Once selected the canvas will become highlighted in the default highlighting color, as shown in figure 5.10 where the "**maxselector.u1.up**" graph has been selected.

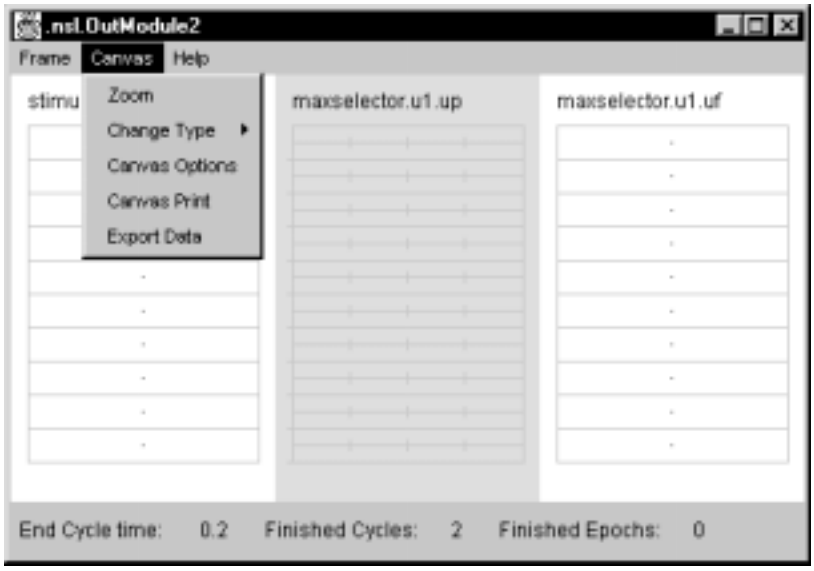

Figure 5.10
The NslOutFrame's Canvas Menu only appears when one of the canvases is selected. In this case, the canvas selected is "maxselector.u1.up" as indicated by the shaded area or highlighted area.

Change Type of Graph in Canvas

To change the type of graph displayed within the canvas, select "Canvas→Change Type". A submenu will appear with the following graph type options: Area, Bar, Dot, Spatial, String, **Temporal**, **AreaColor**, **ImageColor**, **MultiTemporal** and **XY**. All of these graph types are described in the section titled "**Output Graph Types**".

Zoom Canvas

To see the labels on the axis and tick marks, first select the canvas, and then select "Canvas→Zoom". A separate Zoom window will appear, as shown in figure 5.11. Once the window appears, drag the mouse over the area of interest starting in one corner and holding down the mouse button, drag mouse to the opposite corner. Then select "ZoomIn", and the window should now magnify the area selected. (In figure 5.11, we have actually executed the model before selecting the graph, and then we selected zoom.)

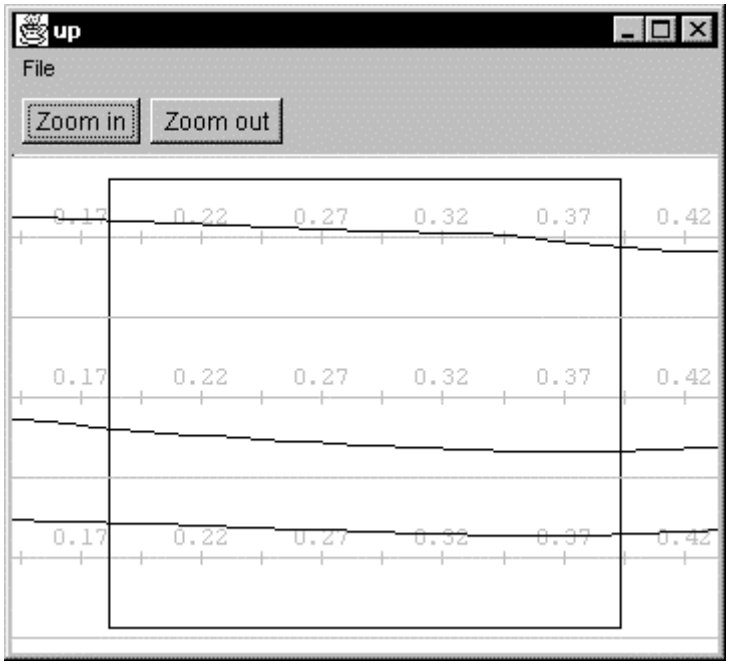

Figure 5.11
Zoom Window Pop-up Displaying the Zoomed Temporal Graph

Options for a Canvas

To change the property options of a selected canvas, choose "Canvas→Canvas Options" from the NslOutFrame menu bar. A Properties popup window based on the graph type displayed will appear. figure 5.12 displays the options for the "Area Level Graph".

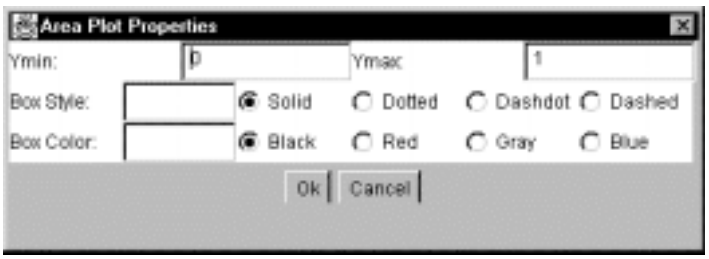

Figure 5.12
Canvas Options—Change Area Level Graph Plot Properties

The Canvas or Plot Properties that can be changed are the *y* minimum, *y* maximum values, the style of the box, and the color of the box.

Print a Canvas

To print a canvas in any of the predetermined formats, first select the canvas and then the "Canvas→Canvas Print" menu item. A popup window should appear that looks exactly like that of the "Frame→Frame Print" menu.

Exporting the Data from a Canvas Window to a File

To export the data from a Canvas Window in one of the specified binary formats first select the canvas and then select "Canvas→Export Data". A pop-up window will appear as explained in more detail in Appendix II.

NSL Output Graph Types

NSL canvases can display different graph types, as either a basic intensity plot (shows variable's values at the current time) or a temporal plot (shows a variable's values over a certain time period). This list of graph types will grow as more and more modelers add their custom output widgets or graph types to the standard set of Nsl Output Widgets or Graphs. Thus it is recommended that you consult the NSL web site for new widgets and graphs that have been added (see Appendix II). The graph types are the following:

- **Area**—The window is divided into small rectangular boxes each representing the activity of an element of the variable during one cycle. Negative values are drawn with an open box while positive values are shaded. The shaded boxes are centered in the middle of the element and the stronger the element value, the larger the box. The graph is updated every Display Delta increment. figure 5.13 shows two such canvases, the one on the left (s_out) and the one on the right (uf).

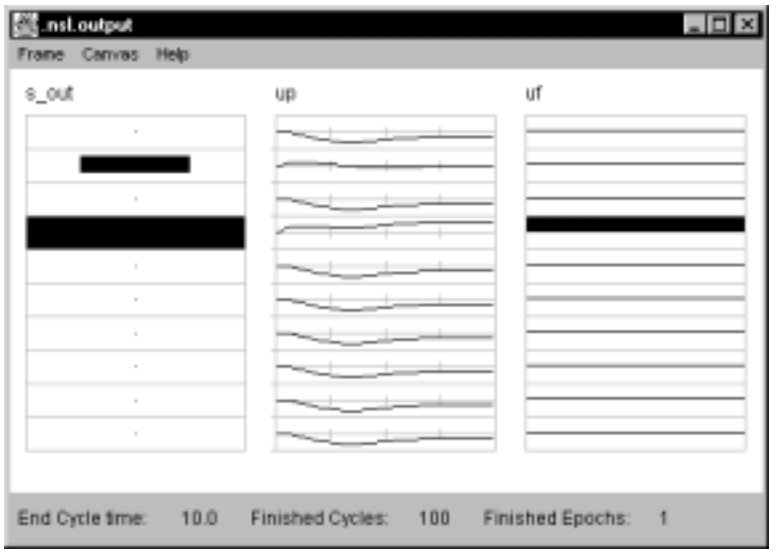

Figure 5.13
Examples of Area Level and Temporal Graph Types

- **Bar**—The bar graph is similar to the area plot. Instead of drawing a box representing the value of the element, a bar is drawn instead. The bottom of the box represents the *y* minimum value and the top of the box the *y* maximum value. Positive values display a filled in bar while negative values display an open bar. The graph is updated every Display Delta increment.
- **Dot**—The dot plot is most similar to the area plot, however, instead of plotting each element as a box, a small "dot" is drawn instead and no grid is displayed. Typically, the dot plot is only applied to two-dimensional matrices and the location of the dot represents x and y coordinates. If a value is zero or negative, it is not drawn. The graph is updated every Display Delta increment. The graph on the right-hand side of figure 5.14 contains a dot plot.

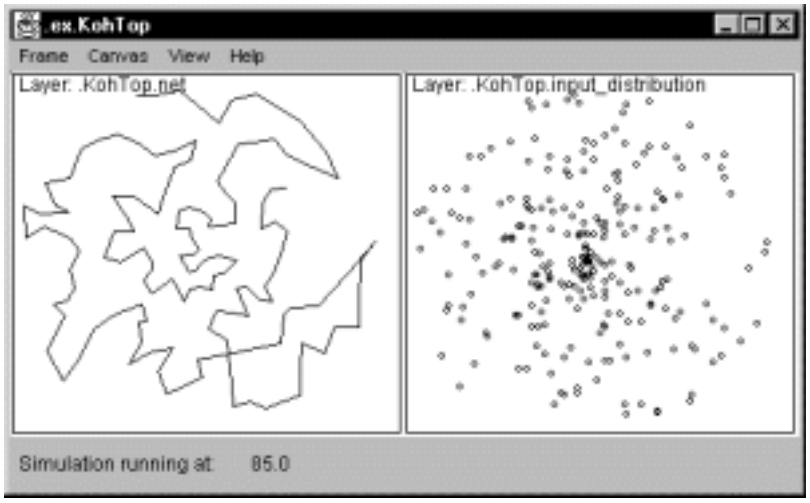

Figure 5.14
Example of String and Dot Plots

- **Spatial**—The Spatial plot is similar to the Area plot, however, instead of shading the boxes, a point is drawn on the *y*-axis representing the activity of the variable. Negative values are drawn below the zero line. Positive values are drawn above the zero line. Once all elements are plotted, a line is drawn connecting the points. For two-dimensional data, the plot is draw in three dimensions. The graph is updated every Display Delta increment. figure 5.15 shows a canvas containing a three dimension spatial plot.

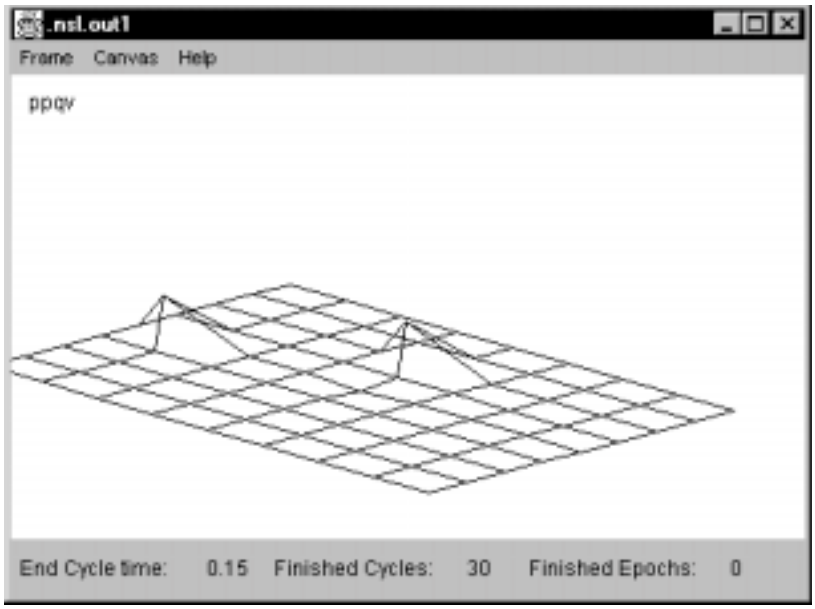

Figure 5.15
Example of Spatial Graph or Three Dimensional graph.

- **String**—The String plot is similar to the Dot plot, however, the dots are much smaller and a line is draw between consecutive dots (time wise). The graph is updated every Display Delta increment. The graph on the left-hand side of figure 5.14 contains a String plot.
- **Temporal**—The Temporal graph represents Time along the horizontal axis and the value of the element of the variable along the vertical axis. In the current version, only 1000 cycles can be viewed at any given time. If a variable has several elements, as in one and two-dimensional arrays, each element will be displayed in its own temporal plot. The middle graph shown in figure 5.13 contains a temporal plot
- **AreaColor**—For the area-level graph in color is just like the area level graph except that it uses both size denote the value of the element represented by the box and color to represent the data type of the element within the box. Each different color can represent a different type of data, such as a special type of neuron. The graph is updated every Display Delta increment.
- **ImageColor**—The color scale map represents each element of a color array as a pixel. The greater the value of the element the warmer the color.
- **MultiTemporal**—The Multi-variable Temporal graph is just like the regular temporal plot, only instead of plotting one variable it can plot up to ten variables each in its own color and line style.
- **XY**—The x axis represents one variable and the y axis represents another variable.

5.3 NslInFrames

NslInFrame is a container for one or more **NslInCanvases** that contain widgets that control the input to NSL variables. Just like the **NslOutFrames**, **NslInFrames** have a standard menu system.

The NslInFrame‘s Menu

The Frame menu contains all of the commands necessary to change the contents and attributes of the frame. These commands are **New Canvas, Remove Canvas, Columns, Frame Options, Frame Print,** and **Close**. These commands have already been described in section 5.2 except for the types of input widgets that can be placed on the frame when a “New Canvas” command is selected, as shown in figure 5.16.

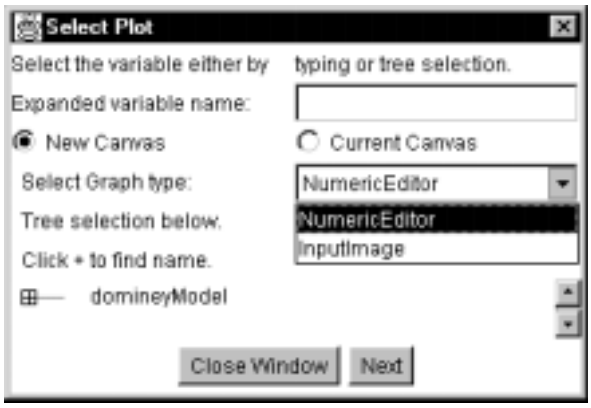

Figure 5.16
Input Widget Types that can be selected at Run-time and placed on a NslInFrame.

The NslInFrame's Canvas Menu

All of the Frame's Canvas Menu items pertain to changing the properties of a particular canvas. The canvas menu options are the same as they are in section 5.3 for the NslOutFrame.

NSL Input Graph Types

The input graph type options currently supported are:

- **NumericEditor**—The NumericEditor graph displays a one-dimensional or a two-dimensional grid containing the values of the elements. It is unique in that the values shown can also be modified for input to the simulation. figure 5.17 shows three canvases containing a numeric editor each.

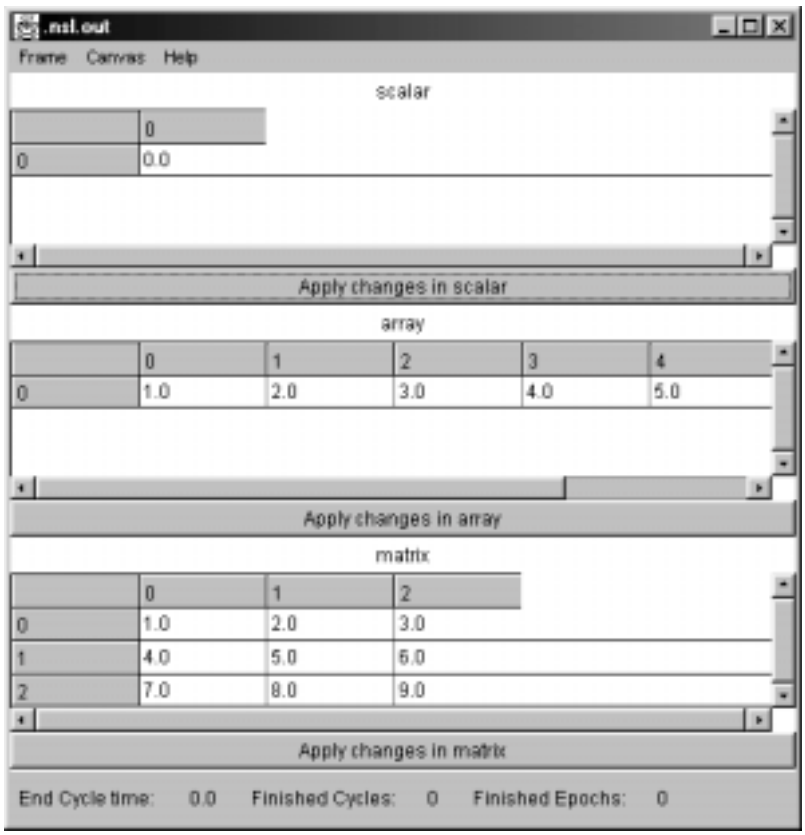

Figure 5.17
Example of the Three NumericEditor Canvases. The first canvas shows the variable scalar, and the single value it contains as well as an Apply button. The second canvas shows the variable array, and the four values it contains as well as an Apply button. The third canvas shows the variable matrix, and the nine values it contains as well as an Apply button. The values are updated every Display Delta increment.

- **InputImage**—InputImage graphs divides a canvas into small boxes; each box represents the absolute value of an element of the variable during one cycle. If the box is not selected the variable will take *wymin* value, otherwise it will be *wymax*. figure 5.18 shows two canvases containing the second one an image editor.

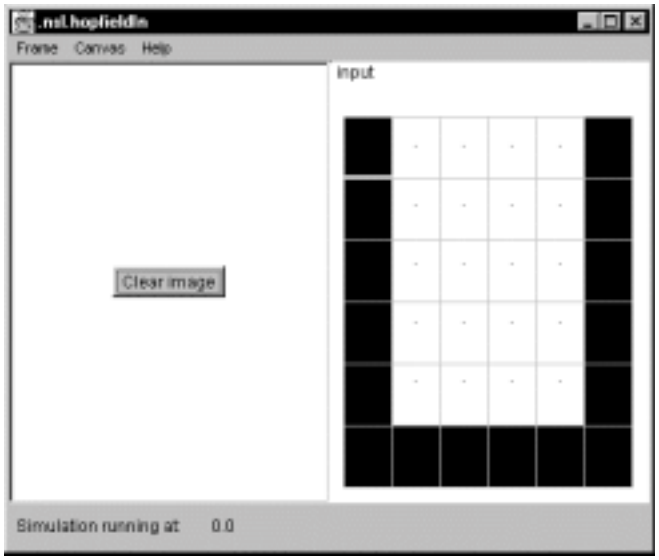

Figure 5.18
The Hopfield Example (chapter 3) of an input image editor.

5.4 Summary

In this chapter we have demonstrated the NSL user interface. We showed how the NSL Executive window is used to control the complete simulation as well as to display new windows via pull-down menus. (Users more comfortable with scripts can use the scripting language within the NSL Script window to accomplish the same tasks as explained in chapter 7.) We have also demonstrated how users can use the built-in graph types to display the results of their simulation, or as input to the simulation.

Notes

1. Different formats depending on the particular environment are PostScript (PS), Graphics Interchange Format (GIF), or Joint Picture Extraction Group (JPEG).