Plotter

by Matt Morse, NeXT Technical publications

Overview

Plotter is a simple graphing program. The user creates points on a graph either by entering values in a text field and clicking the Plot button or by clicking within the graph itself. As the user drags the mouse within the graphing area, a readout of the current coordinates is constantly updated. Through the Services mechanism, Plotter provides its plotting service to any application that can send it ASCII data.

As an example, Plotter is more comprehensive than most of the other mini-examples. It's not intended to reveal particular features of the Application Kit; rather it seeks to illustrate good object-oriented design. PlotView, the principal custom class in this application, was designed for reuse. A PlotView knows how to draw points on a graph, but it relies on a delegate for the actual data. You can also reuse PlotView by making it into a custom palette in Interface Builder. (See the **PlotterPalette** folder for more details about making PlotView into a palette object.)

Program Organization

The Plotter nib file

This nib file contains the specifications for Plotter's user interface. Plotter's standard window has a CustomView (which as been assigned the PlotView class), a scrolling text field, and two buttons. Plotter's main menu is unremarkable except for the Services command. At runtime, the Services submenu is managed entirely by the Application Kit, which enables and disables specific services according to your application's current state. (See the discussion of PlotController below and the Application Kit release notes for more information.)

The PlotView object is connected to the other objects in the application through four connections. The Plot and Clear buttons send **plot:** and **clear:** action messages to the PlotView when the user clicks them. The Print menu command sends the **printPSCode:** message to the PlotView to cause it to print it's graph. Finally, the PlotView is connected to the PlotController object (visible in Interface Builder's File window) through PlotView's **delegate** outlet.

The PlotController provides the glue to hold the application together. It has an outlet (**thePlotView**) that connects it to the PlotView object, and another outlet (**theScrollView**) that connects it to the scrolling text field. The PlotController is the Application object's delegate. (To see this connection, inspect the connections of the File's Owner object.)

Major Classes in the Application

The two custom classes in Plotter are:

PlotView	A subclass of View that knows how to draw axes (see **drawingFuncs.psw** for the drawing code), how to respond to mouse-down and mouse-dragged events within its bounds, and how to register and draw points. It communicates with a delegate object to get a stream of data to use for the points it will plot. PlotView uses a Storage object to hold the point structures it parses from the stream. This class is designed to be reusable. By providing it with an appropriate delegate and connecting Plot and Clear buttons to it, it could be used in another application.
PlotController	A subclass of Object that manages the PlotView in this application. It's primary function is to provide the PlotView with data when requested. To do this, the PlotController needs to

determine the **id** of the Text object within the scrolling text field (which it does by sending a query to **theScrollView** in its **appDidInit:** method). Whenever the PlotController receives from the PlotView a request for coordinates, it connects the PlotView to a stream of data coming from the Text object. The PlotController also updates the text field with new points whenever the PlotView notifies it that the user has created a new point by clicking the mouse within the graph. The PlotController makes itself the services delegate so that it can manage requests from other applications to plot data.

This class is not designed to be reusable. It's sole purpose is to connect the reusable PlotView class to the other objects in this particular application.

Other Peculiarities

Two other features are of interest. First, **drawingFuncs.psw** contains the PostScript language procedures PlotView uses to draw its axes and points. Take a look at how the an array is used to define the tick lengths for the x and y axes. This technique makes it easy to modify the lengths. Second, in **PlotView.m** note that a Cell is used to provide a readout of the cursor coordinates when the user drags the cursor within the graph. A Cell provides a very light-weight, responsive object for handling text within a host View.

Other Files

In addition to the files discussed above, the Plotter project includes these files:

Plotter.nib	The main nib file, the user-interface of the application.
PlotterIcon.tiff	The application icon.

drawingFuncs.psw	Wraps for PostScript procedures that draw PlotView's axes and points.
cross.tiff	A TIFF image that's used as the cursor when the user presses the mouse button within the PlotView.
services.txt	A description of the service protocol for Plotter.
Makefile.preamble	Adds a line to the LDFLAGS to load **services.txt** into a Mach object file segment of the Plotter executable.
Plotter_main.m, IB.proj, Makefile, Plotter.iconheader, drawingFuncs.h	Created by Interface Builder.

Not valid for 1.0
Valid for 2.0