# 7

# Plot

*Edward A. Lee*
*Christopher Hylands*

## 7.1 Overview

The plot package in Ptolemy II is one of several utility packages that provide support functionality for simulations and applets. It is available in a stand-alone distribution, or as part of the Ptolemy II system. The class diagram is shown in figure 7.1. The key classes are:

- PlotBox: A panel that draws a box with axes along the edges, tick marks along the axes, axis labels, a title, and a legend.
- Plot: An extension of PlotBox that supports a suite of two-dimensional plots of sets of data, including x-y plots, scatter plots, and bar graphs.
- PlotLive: An extension of Plot designed to run continuously in its own thread, continually updating an on-screen plot.
- PlotApplet: An applet that contains a single instance of Plot and that can read the data to be plotted from a URL.
- PlotLiveApplet: An extension of PlotApplet that contains an instance of PlotLive instead of Plot. This is used for applets with animated plots that are continually updated.
- PlotFrame: A window containing a single plot and a menu bar with commands for opening and displaying new data files.
- PlotApplication: An extension of PlotFrame that is an application (a standalone Java program).

## 7.2 User Interface

The user interface supported by these classes is very rudimentary. Zooming in and out is supported. To zoom in, drag the mouse downwards to draw a box. To zoom out, drag the mouse upward. In addition, several of these classes permit the placement of buttons that exercise certain simple con-

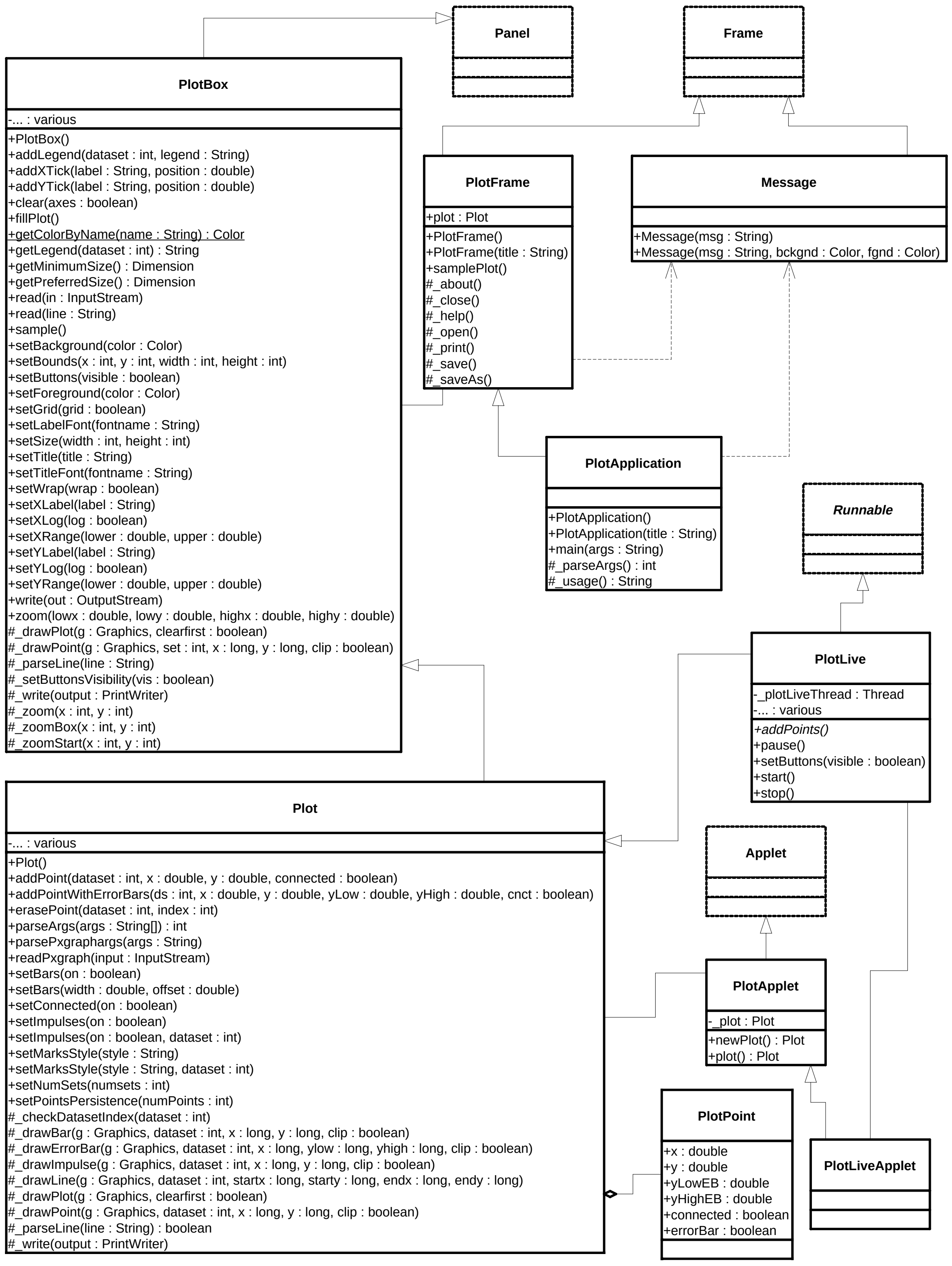
Panel
Frame
PlotBox
-... : various
+PlotBox()
+addLegend(dataset : int, legend : String)
+addXTick(label : String, position : double)
+addYTick(label : String, position : double)
+clear(axes : boolean)
+fillPlot()
+getColorByName(name : String) : Color
+getLegend(dataset : int) : String
+getMinimumSize() : Dimension
+getPreferredSize() : Dimension
+read(in : InputStream)
+read(line : String)
+sample()
+setBackground(color : Color)
+setBounds(x : int, y : int, width : int, height : int)
+setButtons(visible : boolean)
+setForeground(color : Color)
+setGrid(grid : boolean)
+setLabelFont(fontname : String)
+setSize(width : int, height : int)
+setTitle(title : String)
+setTitleFont(fontname : String)
+setWrap(wrap : boolean)
+setXLabel(label : String)
+setXLog(log : boolean)
+setXRange(lower : double, upper : double)
+setYLabel(label : String)
+setYLog(log : boolean)
+setYRange(lower : double, upper : double)
+write(out : OutputStream)
+zoom(lowx : double, lowy : double, highx : double, highy : double)
#_drawPlot(g : Graphics, clearfirst : boolean)
#_drawPoint(g : Graphics, set : int, x : long, y : long, clip : boolean)
#_parseLine(line : String)
#_setButtonsVisibility(vis : boolean)
#_write(output : PrintWriter)
#_zoom(x : int, y : int)
#_zoomBox(x : int, y : int)
#_zoomStart(x : int, y : int)
PlotFrame
+plot : Plot
+PlotFrame()
+PlotFrame(title : String)
+samplePlot()
#_about()
#_close()
#_help()
#_open()
#_print()
#_save()
#_saveAs()
Message
+Message(msg : String)
+Message(msg : String, bckgnd : Color, fgnd : Color)
PlotApplication
+PlotApplication()
+PlotApplication(title : String)
+main(args : String)
#_parseArgs() : int
#_usage() : String
Runnable
PlotLive
-_plotLiveThread : Thread
-... : various
+addPoints()
+pause()
+setButtons(visible : boolean)
+start()
+stop()
Plot
-... : various
+Plot()
+addPoint(dataset : int, x : double, y : double, connected : boolean)
+addPointWithErrorBars(ds : int, x : double, y : double, yLow : double, yHigh : double, cnct : boolean)
+erasePoint(dataset : int, index : int)
+parseArgs(args : String[]) : int
+parsePxgraphargs(args : String)
+readPxgraph(input : InputStream)
+setBars(on : boolean)
+setBars(width : double, offset : double)
+setConnected(on : boolean)
+setImpulses(on : boolean)
+setImpulses(on : boolean, dataset : int)
+setMarksStyle(style : String)
+setMarksStyle(style : String, dataset : int)
+setNumSets(numsets : int)
+setPointsPersistence(numPoints : int)
#_checkDatasetIndex(dataset : int)
#_drawBar(g : Graphics, dataset : int, x : long, y : long, clip : boolean)
#_drawErrorBar(g : Graphics, dataset : int, x : long, ylow : long, yhigh : long, clip : boolean)
#_drawImpulse(g : Graphics, dataset : int, x : long, y : long, clip : boolean)
#_drawLine(g : Graphics, dataset : int, startx : long, starty : long, endx : long, endy : long)
#_drawPlot(g : Graphics, clearfirst : boolean)
#_drawPoint(g : Graphics, dataset : int, x : long, y : long, clip : boolean)
#_parseLine(line : String) : boolean
#_write(output : PrintWriter)
Applet
PlotApplet
-_plot : Plot
+newPlot() : Plot
+plot() : Plot
PlotPoint
+x : double
+y : double
+yLowEB : double
+yHighEB : double
+connected : boolean
+errorBar : boolean
PlotLiveApplet

FIGURE 7.1. The classes of the plot package.

trols. The fill button fills the available space with the available data. The start and stop buttons, used by the PlotLive class, start and stop an animated plot.

The PlotFrame class adds a menu that contains a richer set of commands, including opening files, saving the plotted data to a file, printing, etc. Currently, the set of commands is far from complete. In the future, we hope that on-line formatting of the plots and exporting to popular graphics formats will be supported.

## 7.3 File Format

Instances of the PlotBox and Plot classes can read a simple file format that specifies the data to be plotted. These files can be accessed via URLs. Each file contains a set of commands, one per line, that essentially duplicate the methods of these classes. There are two sets of commands currently, those understood by the base class PlotBox, and those understood by the derived class Plot. Both classes ignore commands that they do not understand. In addition, both classes ignore lines that begin with "#", the comment character. The commands are not case sensitive. In addition, for backward compatibility, these classes can read the binary file format of a prior plotting program called pxgraph.

**NOTE:** We are likely to change the preferred file format to one based on XML. We hope to maintain backward compatibility with this format, but do not plan to extend this format.

### 7.3.1 Commands Configuring the Axes

The following commands are understood by the base class PlotBox. These commands can be placed in a file and then read via the read() method, or via a URL using the PlotApplet class. The recognized commands include:

- **TitleText**: *string*
- **XLabel**: *string*
- **YLabel**: *string*

These commands provide a title and labels for the X (horizontal) and Y (vertical) axes. A *string* is simply a sequence of characters, possibly including spaces. There is no need here to surround them with quotation marks, and in fact, if you do, the quotation marks will be included in the labels.

The ranges of the X and Y axes can be optionally given by commands like:

- **XRange**: *min*, *max*
- **YRange**: *min*, *max*

The arguments *min* and *max* are numbers, possibly including a sign and a decimal point. If they are not specified, then the ranges are computed automatically from the data and padded slightly so that datapoints are not plotted on the axes.

The tick marks for the axes are usually computed automatically from the ranges. Every attempt is made to choose reasonable positions for the tick marks regardless of the data ranges (powers of ten multiplied by 1, 2, or 5 are used). However, they can also be specified explicitly using commands like:

- **XTicks**: *label position*, *label position*, ...
- **YTicks**: *label position*, *label position*, ...

A *label* is a string that must be surrounded by quotation marks if it contains any spaces. A position is a number giving the location of the tick mark along the axis. For example, a horizontal axis for a frequency domain plot might have tick marks as follows:

```
XTicks: -PI -3.14159, -PI/2 -1.570795, 0 0, PI/2 1.570795, PI 3.14159
```

Tick marks could also denote years, months, days of the week, etc.

The X and Y axes can use a logarithmic scale with the following commands:

- **XLog**: on
- **YLog**: on

The tick labels, if computed automatically, represent powers of 10. Note that if a logarithmic scale is used, then the values must be positive. **Non-positive values will be silently dropped**.

By default, tick marks are connected by a light grey background grid. This grid can be turned off with the following command:

- **Grid**: off

It can be turned back on with

- **Grid**: on

Also, by default, the first ten data sets are shown each in a unique color. The use of color can be turned off with the command:

- **Color**: off

It can be turned back on with

- **Color**: on

Finally, the rather specialized command

- **Wrap**: on

enables wrapping of the X (horizontal) axis, which means that if a point is added with X out of range, its X value will be modified modulo the range so that it lies in range. This command only has an effect if the X range has been set explicitly. It is designed specifically to support oscilloscope-like behavior, where the X value of points is increasing, but the display wraps it around to left. A point that lands on the right edge of the X range is repeated on the left edge to give a better sense of continuity. The feature works best when points do land precisely on the edge, and are plotted from left to right, increasing in X.

All of the above commands can also be invoked directly by calling the corresponding public methods from some Java code.

### 7.3.2 Commands for Plotting Data

The set of commands understood by the Plot class support specification of data to be plotted and control over how the data is shown.

The style of marks used to denote a data point is defined by one of the following commands:

- **Marks**: none
- **Marks**: points
- **Marks**: dots
- **Marks**: various

Here, “points” are small dots, while “dots” are larger. If “various” is specified, then unique marks are used for the first ten data sets, and then recycled. Using no marks is useful when lines connect the points in a plot, which is done by default. If the above directive appears before any DataSet directive, then it specifies the default for all data sets. If it appears after a DataSet directive, then it applies only to that data set.

To disable connecting lines, use:

- **Lines**: off

To re-enable them, use

- **Lines**: on

You can also specify “impulses”, which are lines drawn from a plotted point down to the x axis. Plots with impulses are often called “stem plots.” These are off by default, but can be turned on with the command:

- **Impulses**: on

or back off with the command

- **Impulses**: off

If that command appears before any DataSet directive, then the command applies to all data sets. Otherwise, it applies only to the current data set.

To create a bar graph, turn off lines and use any of the following commands:

- **Bars**: on
- **Bars**: *width*
- **Bars**: *width*, *offset*

The *width* is a real number specifying the width of the bars in the units of the x axis. The *offset* is a real number specifying how much the bar of the i-th data set is offset from the previous one. This allows bars to “peek out” from behind the ones in front. Note that the frontmost data set will be the first one. To turn off bars, use

- **Bars**: off

To specify data to be plotted, start a data set with the following command:

- **DataSet**: *string*

Here, *string* is a label that will appear in the legend. It is not necessary to enclose the string in quotation marks.

To start a new dataset without giving it a name, use:

- **DataSet**:

In this case, no item will appear in the legend.

If the following directive occurs:

- **ReuseDataSets**: on

then datasets with the same name will be merged. This makes it easier to combine multiple data files that contain the same datasets into one file. By default, this capability is turned off, so datasets with the same name are not merged.

The data itself is given by a sequence of commands with one of the following forms:

- *x*, *y*
- **draw**: *x*, *y*
- **move**: *x*, *y*
- *x*, *y*, *yLowErrorBar*, *yHighErrorBar*

- **draw**: *x*, *y*, *yLowErrorBar*, *yHighErrorBar*
- **move**: *x*, *y*, *yLowErrorBar*, *yHighErrorBar*

The “draw” command is optional, so the first two forms are equivalent. The “move” command causes a break in connected points, if lines are being drawn between points. The numbers *x* and *y* are arbitrary numbers as supported by the Double parser in Java (e.g. “1.2”, “6.39e-15”, etc.). If there are four numbers, then the last two numbers are assumed to be the lower and upper values for error bars. The numbers can be separated by commas, spaces or tabs.

The number of data sets to be plotted does not need to be specified.

## 7.4 Exporting

Currently, the ability to export a plot to other formats is rather limited. A small set of key bindings are provided:

- Cntr-c: Export the plot to the clipboard.
- D: Dump the plot to standard output.
- E: Export the plot to standard output in EPS format.
- F: Fill the plot.
- H or ?: Display a simple help message.

The encapsulated postscript (EPS) that is produced is tuned for black-and-white printers. In the future, more formats may supported. Also at this time (jdk 1.1.4), Java's interface the clipboard does not work, so Cntr-c might not accomplish anything.

Exporting to the clipboard and to standard output, in theory, is allowed for applets, unlike writing to a file. Thus, these key bindings provide a simple mechanism to obtain a high-resolution image of the plot from an applet, suitable for incorporation in a document. However, in some browsers, exporting to standard out triggers a security violation. You can use Sun's appletviewer instead.

## 7.5 Limitations

The plot package is a starting point, with a number of significant limitations.

- The PlotFrame and PlotApplication classes should be greatly extended to allow on-line changes in the format of the plots.
- A binary file format that includes plot format information is needed.
- If you zoom in far enough, the plot becomes unreliable. In particular, if the total extent of the plot is more than $2^{32}$ times extent of the visible area, quantization errors can result in displaying points or lines. Note that $2^{32}$ is over 4 billion.
- The log axis facility has a number of limitations listed in the documentation of the _gridInit() method in the PlotBox class.
- Graphs cannot be currently copied via the clipboard.
- There is no EPS export for graphs.
- There is no mechanism for customizing the colors used in a plot.