

NPlot is an open source charting library for the .NET framework. Its interface is both simple to use and flexible. The ability to quickly create charts makes NPlot an ideal tool for inspecting data in your software for debugging or analysis purposes. On the other hand, the library's flexibility makes it a great choice for creating carefully tuned charts for publications or as part of the interface to your Windows.Forms or Web application.

Tool: NPlot	Version: 0.9.9.3
Homepage: http://www.netcontrols.org/nplot	
Power Tools Page: -	
Summary: NPlot is a charting library for the .NET framework v2.0 with a simple to use yet flexible API.	
License Type: Custom. Effectively, a choice of GPL compatible or BSD with advertising clause.	
Resources:	
Related Tools: -	

Requirements

To develop applications that use NPlot, you will need to have a development environment set up that targets the .NET Framework version 2.0. Most people will be using an edition of Microsoft Visual Studio 2005 and this will be assumed in the remainder of this article.

You must also agree to the terms of the NPlot license. The license is very relaxed on the use of NPlot in other open source software or software written for personal use. If however you would like to distribute a closed source application that makes use of NPlot, there is one important condition: You must advertise the fact that your application makes use of NPlot in your application's "about box" or documentation distributed to all users of the application. For more information, please refer to the license included in the distribution. Note that the license has nothing to say and poses no restriction on the use of output of the library.

Setting Up

NPlot is distributed in a zip file which contains the library .dll together with complete C# source to the library and a C# demo showing it in action. You can download this from the NPlot web site.

If you would like to use NPlot with the Windows.Forms designer within Visual Studio, you should add the NPlot.Windows.PlotSurface2D control to the Toolbox. To do this, right click the toolbox panel and select "Choose Items". After the "Choose Toolbox items" dialog box appears, select the "Browse" button and navigate to the NPlot.dll location, double clicking to add the NPlot controls to the list available for selection (see figure 1). Make sure the NPlot.Windows.PlotSurface2D control is checked and press OK. You can now use the PlotSurface2D in the designer as you would any other control.

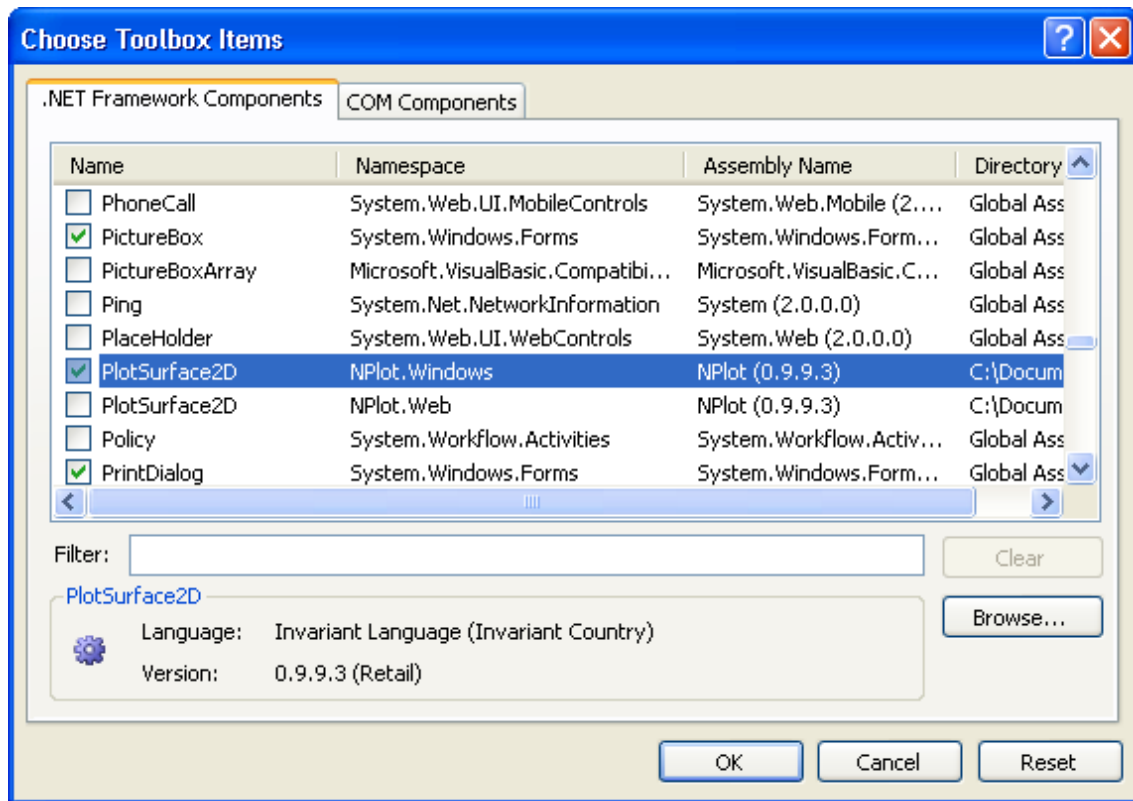


Figure 1 – The Toolbox item selection box.

Finally, you will need to add a reference to the nplot.dll file to any Visual Studio project in which you want to use NPlot.

Library Overview

The PlotSurface2D Classes

To create a chart, the first thing you need to do is construct an instance of a PlotSurface2D class (a class that implements the IPlotSurface2D interface). The role of this object is to coordinate the display of axes, title and legend as well as all the data dependent elements of the chart. NPlot provides three such classes:

- **Windows.PlotSurface2D** - This is a Windows.Forms control that implements the IPlotSurface2D functionality. This class also encapsulates a flexible interaction framework which allows you specify how a user of your application can interact with the chart.
- **Bitmap.PlotSurface2D** - This class allows you to easily draw charts on a System.Drawing.Bitmap object. This class is often used in web applications to generate dynamic charts and is also useful whilst debugging.
- **Web.PlotSurface2D** - This is an ASP.NET control that implements the

IPlotSurface2D functionality. This control should be considered experimental. Due to implementation complexities, this will probably remain the case unless a future version of ASP.NET provides built in “image canvas” functionality. In all but the smallest applications, you should implement dynamic charting on the web via the Bitmap.PlotSurface2D class. An example of this is provided on the NPlot web site.

A PlotSurface2D class has also been written for an older version of NPlot which allows charts to be used in GTK# applications created with the free C# compiler / .NET implementation called Mono under Linux. This is currently not maintained as part of the library, but probably will be in the future.

Plot Elements

With a PlotSurface2D created, you are ready to start charting some data. To do this, you first need to create an instance of a class that implements the IDrawable or IPlot interface (NPlot provides many of these classes). You then point this object to your data, optionally set a few display properties and add it to your PlotSurface2D.

Classes that implement IPlot include LinePlot, PointPlot and ImagePlot – the purpose of a “plot” class is to wrap data and provide functionality for drawing it against a pair of axes. The different plot classes display data in different ways (e.g. as a series of points, or a line). Plot classes can also draw a representation of themselves in a legend if present and can also “suggest” to the PlotSurface2D the axes they would optimally be drawn against.

Classes that implement just the IDrawable interface (IPlot extends this interface) can also be added to a PlotSurface2D. These are lightweight plot elements which include TextItem and ArrowItem. These items can not draw a representation of themselves in the legend or influence the PlotSurface2D’s selection of axes.

A selection of the commonly used classes that implement IDrawable are:

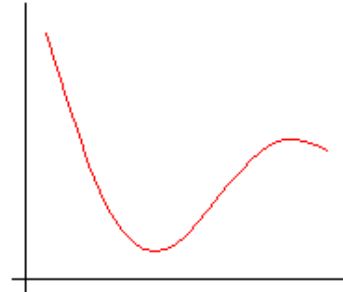
- ArrowItem – Draws an arrow pointing to a particular world coordinate.
- FilledRegion – Creates a filled area between two line plots.
- Grid - Adds grid lines to the plot surface which automatically align to axis tick positions.
- TextItem – Places text at a specific world coordinate.

A selection of the commonly used plot types are described below:

LinePlot

Use a line plot when it makes sense to interpolate between successive data points. For example you may have measurements of the ambient temperature of a room at various times throughout a day. If the reading was taken frequently, a line plot of temperature vs time would make sense.

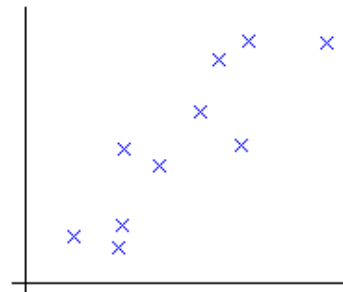
Tip: You can create lines of any color, thickness or dash pattern by specifying the Pen used for drawing.



PointPlot

Use a point plot (scatter chart) when it does not make sense to interpolate between successive data points. For example you might want to visualize the height and weight of a group of people on a chart. You could plot a point for each person with the x position determined by their weight and y position determined by their height.

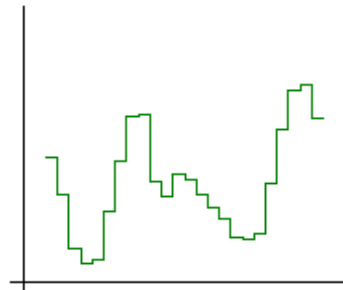
Tip: Around 15 different pre-defined marker styles are available.



StepPlot

Step plots are useful for displaying sample based data (such as PCM audio) where each value can be thought of as representing the value of the measured quantity over a specific time period.

Tip: You can choose whether the horizontal sections of the step plot are centered on the abscissa values or drawn between successive abscissa values.



CandlePlot

This type of plot is often used for displaying financial data. Each bar summarizes a price over a particular time period. The lower and upper positions of the thin sticks indicate the highest and lowest values of the price over the time period. If the filled region is red, the top of the filled region represents the price at the beginning of the time period and the bottom of the filled region the price at the end of the time period. If the filled region is green, the bottom of the filled region is the opening price and the top is the closing price.

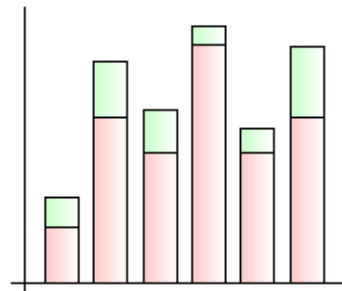
Tip: The candle fill colors are configurable. Also, this plot type can generate “stick” plots which are similar to candle plots.



BarPlot

A bar plot is usually used to chart the number of data values belonging to one or more categories. The height of the bar represents the number of values in the given category. For example, you might have a collection of dogs and data on the breed of each. You could create a chart of the number of each type of breed.

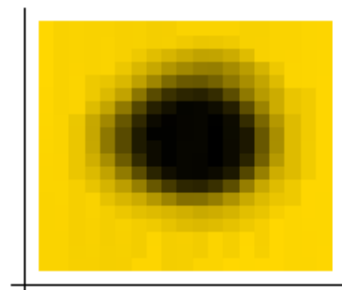
Tips: You will often want to make the lower x axis a LabelAxis (in the above example, the names of the dog breeds). Also, Bar plots can be stacked on top of each other.



ImagePlot

This type of chart is often used to display the variation of a value over a spatial area. Each value in the region is mapped to a color.

Tip: You can specify the color to value mapping using an object of any class that implements IGradient. For example LinearGradient.



Of course, if the built in IPlot or IDrawable classes don't provide the functionality you require, it is straight forward to create your own class that implements one of these interfaces. This is perhaps the most common way of extending NPlot.

You can add as many plots to a PlotSurface2D object as you like and can control the order in which they are drawn with the z order parameter of the Add method. Also, the PlotSurface2D classes define two independent x axes and two independent y axes. When you add an item, you can chose which x and y axis you would like it to be associated with.

Specifying Data

The IPlot interface does not enforce how data associated with the specific plot classes should be specified. However where it makes sense, these classes provide a consistent interface for this purpose.

Data can be provided in one of two forms:

1. In an object of type DataSet, DataTable or DataView from the System.Data namespace.
2. In any collection that implements the IEnumerable interface where it is valid to cast each of the elements to type double. Examples of such collections are
 - a. Double[]
 - b. System.Collections.ArrayList
 - c. System.Collections.Generic.List<System.Int16>

If you are working with very large data sets and efficiency is of concern, you should prefer to pass your data to NPlot via the built in array type double[], as this case has been highly optimized.

The following four properties provided by most plot classes are used to specify your data:

DataSource: If the data is to be taken from a DataSet, DataTable or DataView, this property should be set to that object.

DataMember: If the data is to be taken from a DataSet, this property should be set to a string containing the name of the DataTable in the DataSet to take the data from.

AbscissaData: The x coordinates of the data to plot. This property is optional – if not specified (or set to null), the abscissa data will be assumed to be 0, 1, 2, ... If data is being read from a DataTable or DataView, this should be a string containing the name of the column to take the data from. Otherwise, this can be set to any container that implements the IEnumerable interface.

OrdinateData: The y coordinates of the data to plot. If data is being read from a DataTable or DataView, this should be a string containing the name of the column to take the data from. Otherwise, this can be set to any container that implements the IEnumerable interface.

Where the above interface is not suitable for a particular plot type, the interface is as close to this as possible. For example CandlePlot provides OpenData, LowData, HighData and CloseData properties instead of OrdinateData.

Axes

A PlotSurface2D object automatically determines axes suitable for displaying the plot objects that you add to it. However, these are highly customizable. Some common things you might wish to add / adjust are:

- A label for the axis (using the Label property)
- Tick Text / Label fonts (using the LabelFont and TickTextFont properties)
- The angle of the text next to the ticks (using the TicksAngle property)
- The pen used to draw the axis (using the AxisPen property)
- World minimum and maximum values (using the WorldMin and WorldMax properties)

You can also replace the default axes with a completely different axis type. NPlot provides a number of different Axis types shown in table 1 whose characteristics are individually configurable.

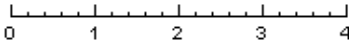
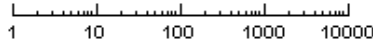
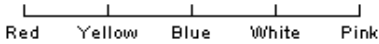

LinearAxis:	
LogAxis:	
LabelAxis:	
DateTimeAxis	

Table 1 – The main NPlot axis types.

Examples

Unfortunately, a detailed C# or VB.NET example of NPlot usage is out of the scope of this short article. Instead, refer to the NPlot website or the demo provided with the distribution. Figure 2 provides some examples of the types of output achievable using NPlot.

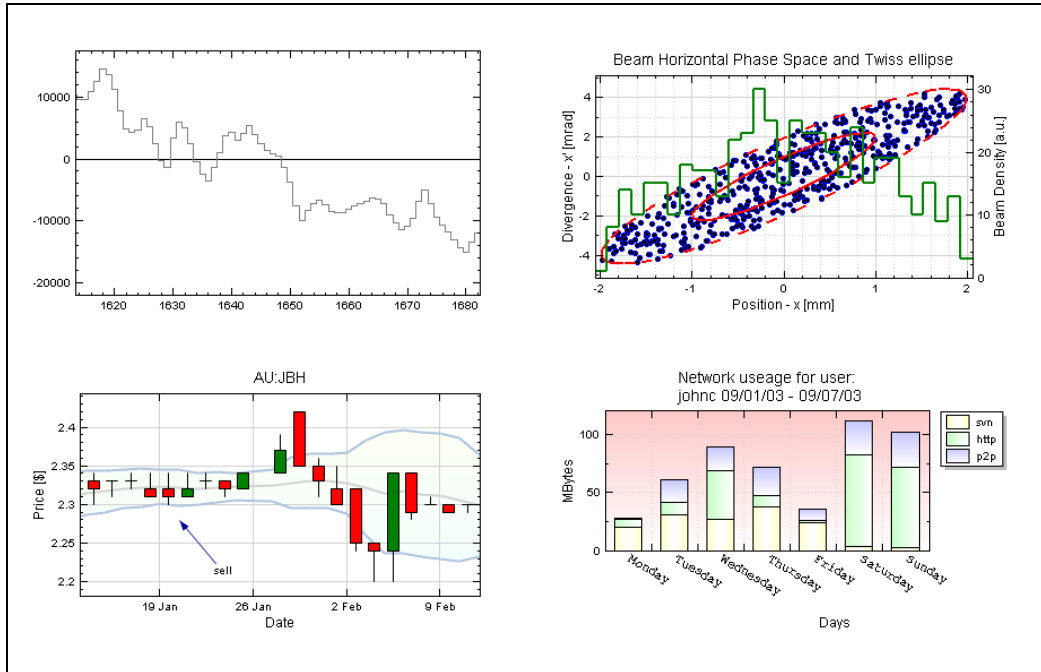


Figure 2 – Examples plots generated using NPlot.

Development Status

The current version of NPlot is 0.9.9.3 – it has not yet reached version 1.0. This reflects the fact that:

- There is still some functionality missing that many users would expect from a charting library. NPlot is not yet considered basic feature complete (though it is getting close).
- The API is still subject to change without notice or with regard to backwards compatibility. The focus remains on creating the best library design possible.
- There are no separate development / stable branches of the code. A given release of NPlot may include both bug fixes and significant enhancements. The latter have the potential to break functionality that was working in a previous release.

With the above said, NPlot is known to be used in several production systems. If you only use basic functionality of the library, you should find it very reliable.

Summary

NPlot is an easy to use, flexible charting library which has a wide range of applications. It is under active development with particular focus leading up to version 1.0 on polishing the interface and achieving a very stable release.