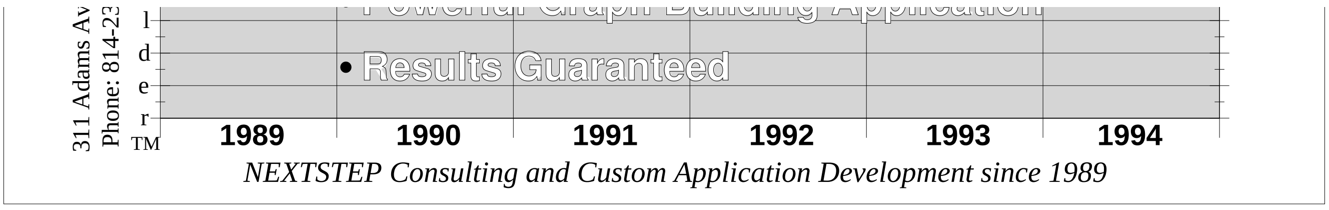

311 Adams Av
Phone: 814-23
l
d
e
r
TM
Powerful Graph Building Application
• Results Guaranteed
1989
1990
1991
1992
1993
1994
NEXTSTEP Consulting and Custom Application Development since 1989

VVI offers the following in regards to the Graph Object Library:

Support And Service

The Graph Object Library delivers a high level of data and graph capabilities. To help make the most of these capabilities VVI provides quality support such as:

- **On Site Instruction.** Introduces the architecture and features of the VVI-Class, Kit, and Graphic libraries. Shows how to write applications that control and display data in a graph format, and how the libraries fit into NEXTSTEP. Although experience with NEXTSTEP is recommended, it is not required.

  Syllabus
  - *Demonstration*: Demonstration and use of the application *GraphBuilder*.
  - *Conceptual Model*: Graphic-Plains, Coordinates, and Graphics.
  - *Class Overview*: Segmentation and explanation of the Class, Kit, and Graphic libraries and their respective classes.
  - *Custom Graphics*: How to write your own graphics, with examples provided.
  - *Postscript Optimization*: General discussion of user paths, caches, user objects, and selection of coordinate systems.

- **Technical Support.** Provides support for the VVI libraries in areas such as custom graphics, user interface integration, database integration, graphics optimization, and Objective-C, c++, and postscript language coding.

- **Contract.** Provides reliable and proven expertise which focus on and satisfy your project goals.

- **System Integration.** Provides comprehensive support and service for all aspects of your custom data application and computing needs. VVI is an authorized system integrator, developer, and reseller for many computer manufacturers as well as an authorized reseller of NEXTSTEP.

Licensing

Licensing is designed for both commercial and in-house developers and include the following formats:

- **Developer.** For use where the object libraries and resources are incorporated into applications for in-house use and distribution only. The libraries and technical support for the first year or on-site instruction must be purchased. Up to 5 in-house developers may use the libraries without additional fees.

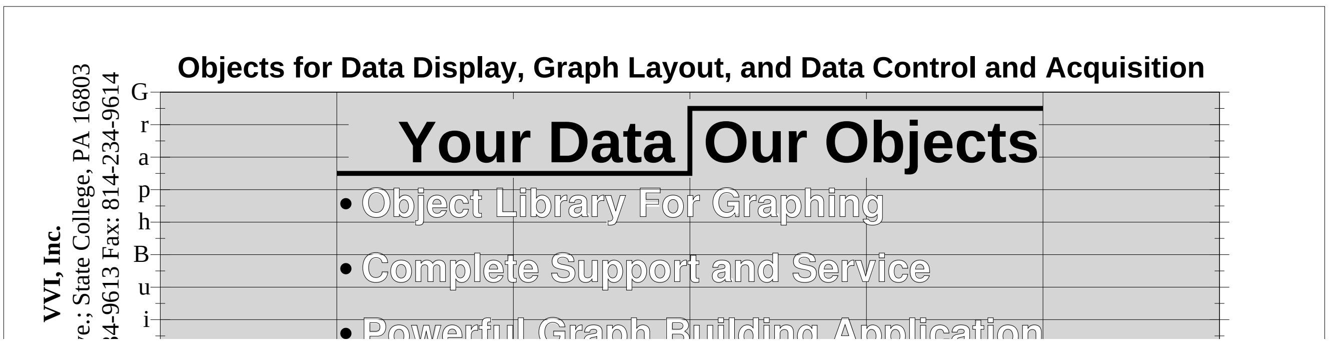
- **Commercial.** For general use and distribution of your application with the VVI object code incorporated. The libraries, technical support for the first year, and on-site instruction must be purchased. Additional fees are structured on a royalty basis or single up-front purchase. Contact VVI for additional information.

- **Source Code.** For distribution of your application with the VVI source code incorporated in object form. The libraries, technical support for the first year, and on-site instruction must be purchased. Contact VVI for additional information.

- **Extensive Documentation.** Online help using NEXTSTEP online help facilities. Online manual using the digital librarian. Several example documents regarding general drawing, graphing, target/action control loops, and program generated animated data.
- **Data Entry.** Data can be entered by drag/drop of file icons, palettes, pasteboard, program module hooks, or mouse editing. Program modules can be used for custom database and data acquisition server integration. Includes dragging from .eps, .tiff, .arrayData, and .matrixData type files.
- **Document System.** Documents are stored as multi-level UNIX directories. Internal document file and program module links maintain relations for large data sets and relocatable object code. Source code, notes, and other information can be put into the documents and edited using standard applications.
- **Controllers.** The graphics issue owner and target messages so your system can react to changes in graphic attributes and editing. For example graphics can be used to control auxiliary devices.
- **Palettes.** Any graphic, and hence data or database interface, may be stored and retrieved from palettes. This includes entire graphs.
- **Point and Click.** Every graphic implements a full complement of point and click mouse control.
- **Inspectors.** Graphic attribute can be modified directly from inspectors. Inspector-Editors maintain inspector user interface controls and relations between the associated graphic attributes. Inspector-Editors are linked to the graphics and between each other in a general network by a set of inspector-editor links.
- **Program Modules.** Control loops, animation, drawing and most other features can be controlled programmatically using relocatable object modules accessible through the program module editor which supports RTF formatted Objective-C, c++, postscript, and server source code as input. Almost any computation can be performed with program modules.
- **Optimized Drawing.** Drawing is optimized and takes full advantage of the most advanced postscript optimization techniques as well as bitmap caching and minimum bound clipping. Plot 16,000 point (x-y pairs) contour plot (10 levels) in 3 seconds. Plot 1024 points a few times per second.
- **Standard Conversions.** Conversion to eps, tiff, and ascii formats are included and supported in the pasteboard and file dragging mechanisms of NEXTSTEP.
- **NEXTSTEP Compliant.** Standard NEXTSTEP utilities such as the color, font, print, and fax panel, services, save and open panels, and pasteboard are seamlessly incorporated and used to their full extent.
- **Geometric Transformations.** All graphics, including data graphics, can be controlled through mouse and key commands. Rotation, translation, skewing, selecting, focusing, and point editing are but a few operations available.
- **Document Page Layout.** Multiple plots per page, labels, titles, and overlapping plots or graphing areas may be arranged using familiar click and drag methods.
- **Document Notes.** Notes regarding the document can be accessed by pressing Document/Notes... from the main menu or by selecting the appropriate inspector link.
- **Prototyping.** The graphics, document, and control loops can be developed and tested using GraphBuilder. This methodology is similar to the NEXTSTEP Interface Builder. For use with the Graph Object Library.
- **Scaled Drawing.** The cross hairs and inspectors give information which aid in lining up graphics and placing them at a particular coordinate. Coordinate information is retained for most operations.
- **Quick launch.** Secondary resources are deferred until needed resulting in a launch time of 5 seconds (Motorola ’040) or less.

The VVI Graph Object Library

The underpinnings of GraphBuilder is the VVI Graph Object Library. If you are involved with projects implementing custom data interfaces and analysis then the VVI Graph Object library will definitely be of interest to you. These objects represents a unique opportunity to enhance your projects with world class data processing and graph layout software.


Objects for Data Display, Graph Layout, and Data Control and Acquisition
VVI, Inc.
e.; State College, PA 16803
4-9613 Fax: 814-234-9614
G r a p h B u i
Your Data
Our Objects
• Object Library For Graphing
• Complete Support and Service
• Powerful Graph Building Application

```
float dt = [self VVdt];
int aLength = [self VVlength];
floatVVRArray *aPoints = [self VVpoints];
NXRect gRB = *[self VVgRBounds];
[self VVincrementTime];

// Get the data from myServer...

kern_return_t r;
port_name_t mydriver_port;
char *dbData, *dbP;
unsigned int dbData_len;

if(isFresh)
{
         r = vm_allocate((vm_task_t)task_self(), (vm_address_t *)&dbData
         , (vm_size_t) 8192, TRUE);
         r = netname_look_up(name_server_port, "", "mydriver0", &mydriver_port);
         isFresh = 0;
}
r = mydriver_export(mydriver_port, &dbData, &dbData_len);
dbP = dbData;

// Modify the internal data vector...

float *pP = aPoints->array();
float *pPE = pP + aLength*2+4;
*pP++ = gRB.origin.x; // Update the user path bounding box
*pP++ = gRB.origin.y;
*pP++ = gRB.origin.x + gRB.size.width;
*pP++ = gRB.origin.y + gRB.size.height;

for(;pP < pPE; pP+=2) *pP = *dbP++; // Update the data

return 2; // Graphic change, but no bounds change
}
```

Get the data. This example shows a kernel server hook. It could just as well open up a file and read in the data or simply compute the data.

Assign the internal data in a way which is conformal to the internal representation.

By returning 2 GraphBuilder knows to only update the draw and not recompute new or auxiliary bounds.

Another class of program module implements the target action control paradigm. In this paradigm the program module instantiates a target object which is controlled by the graphic. For example moving a point sends a message to the target object. This message informs the target that a point has been moved and gives other information like the actual point moved. This information can then be used by the target to update or control other data, servers, or its controller. Some action messages implemented are:

- (int)**VVgraphicSelected:***aGraphic*
- (int)**VVgraphic:***aGraphic* **changingPoint:**(float *)*aPoint* **atIndex:**(int)*index* **by:**(int)*instruction*
- (int)**VVgraphic:***aGraphic* **movingBy:**(NXSize *)*inc*
- (int)**VVgraphic:***aGraphic* **resizingAtCorner:**(int)*corner* **to:**(NXPoint *)*p*
- (int)**VVgraphic:***aGraphic* **rotatingFromPoint:**(NXPoint *)*origin* **toPoint:**(NXPoint *)*p*

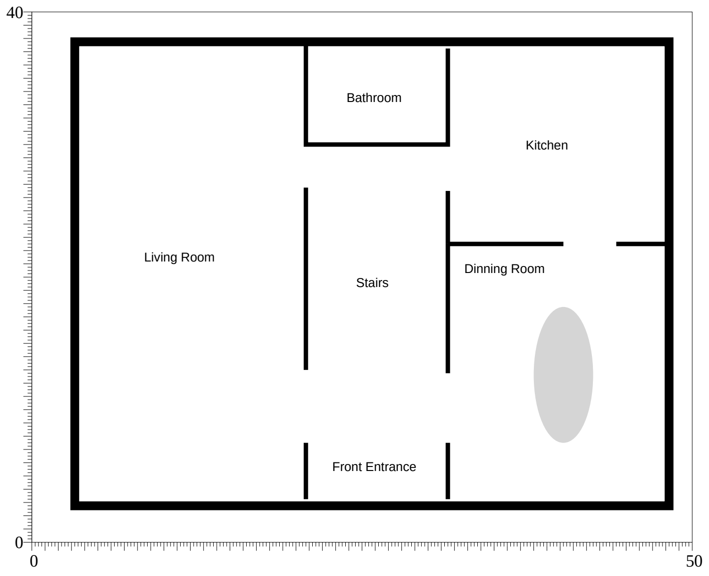
Features

GraphBuilder has extensive features including:

- **Easy to Use and Fast.** Point, click, drag, and paste. Inspector controls are efficiently refocused to deliver important information.
- **Data Graphics.** Any number and combination of histogram, line, smoothed curve, areas, scatter, contours, and signals on multiple coordinate systems. Since graphics, such as arbitrary connected spline graphics, maintain their data attributes they can be used to display data appropriate to that format.
- **Axes.** Rectangular, polar, and semi-log axes (Version 3.3). Just drag out axes like any other graphic, double click within the axes frame, and start dragging and pasting data, enter data using mouse point and click methods, or program calculations using the program module editor. All labels and titles can be altered using familiar mouse and inspector methods.
- **Error bars.** Symmetric and non-symmetric error bars, lines with caps, or straight line error bar types. Attributes include thickness, width, and color. Create almost any data point marker by drag and copy/paste methods.
- **General Graphics.** Bezier curves, lines, ellipses, ovals, parallelograms, polygons, rectangles, rtf, eps, images, wrapped text, groups are included. These graphics can undergo changes in attributes such as smoothness, shadows, interpolated fill, translation, scaling, skewing, rotation, line width, dash pattern, color, etc.

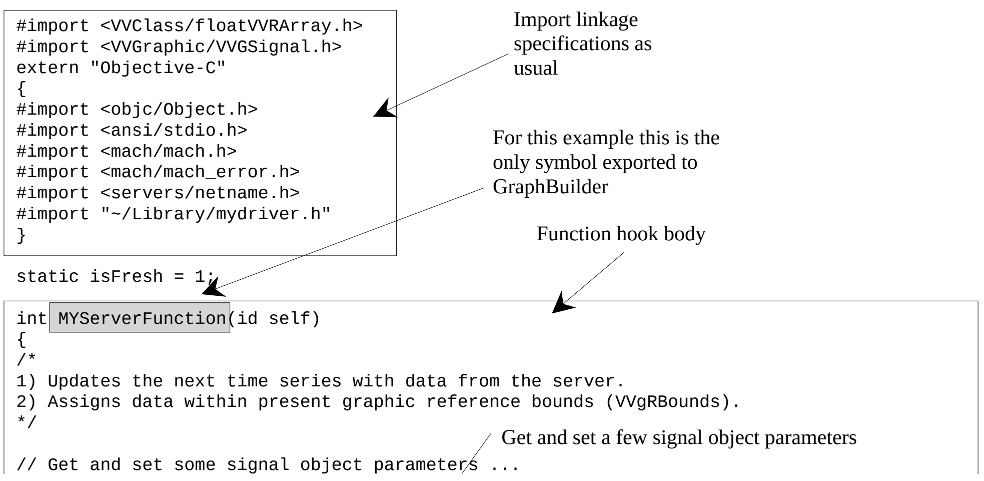
Scaled Coordinates

Since line width and graphic bounds are defined in absolute coordinates and retain their data values they can be used for scaled drawings. For example this simple floor plan:



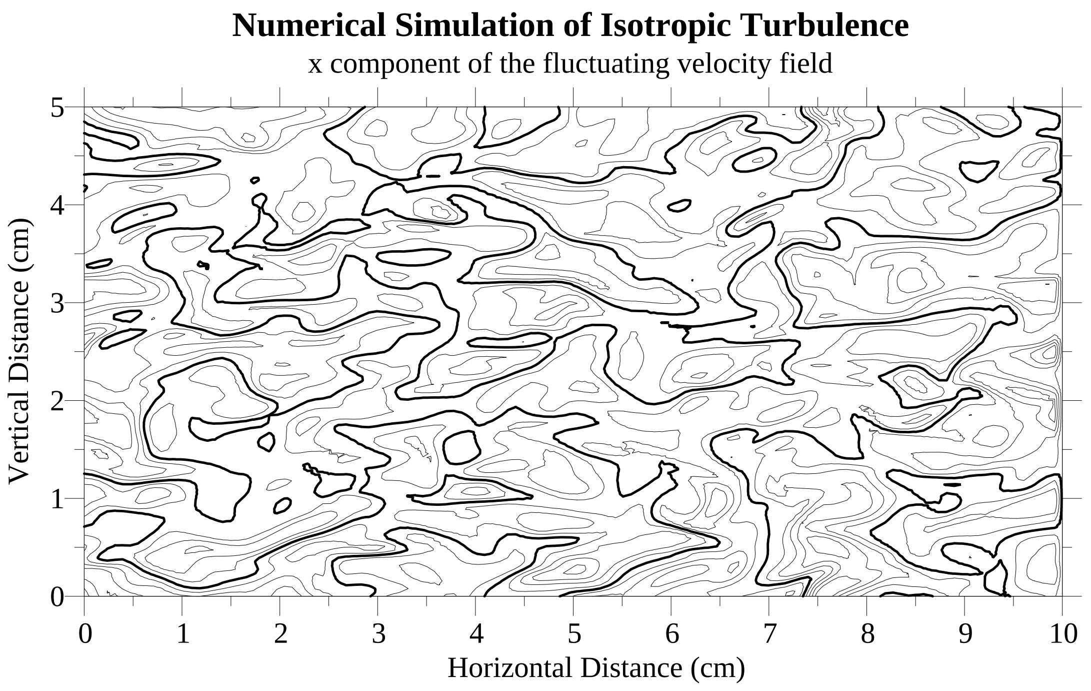
Programming Graphics

Every graphic includes a programmable interface called the program module editor. This editor retrieves files stored in the document and compiles, links, reformats, and loads the resulting code into GraphBuilder. This results in efficient and flexible programming capabilities. The programming languages are ANSI c, c++, Objective-C, postscript, and any server and kernel call. Templates are produced by invoking the program module editor. Examples are included which show how to implement computation based animation and target/action controllers. To get full programmability and access to the GraphBuilder API the library headers, resource files, and archive files contained in the VVLib folder are included so importing linkage specifications to your source code is efficient and systematic. The GraphBuilder library consists of about 50 Objects and 90,000 lines of code. The following diagrams a hypothetical program module for kernel server acquisition:


Import linkage specifications as usual
For this example this is the only symbol exported to GraphBuilder
Function hook body
Get and set a few signal object parameters

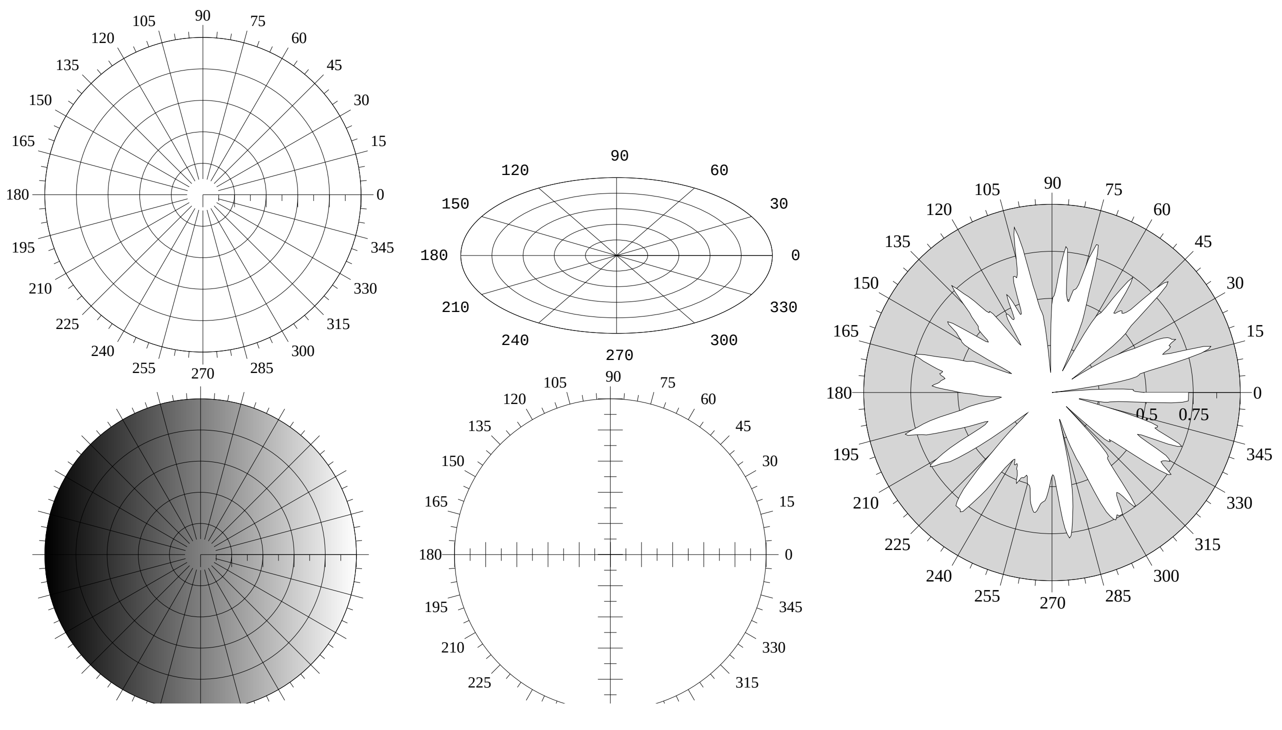
```
#import <VVClass/floatVVRArray.h>
#import <VVGraphic/VVGSignal.h>
extern "Objective-C"
{
#import <objc/Object.h>
#import <ansi/stdio.h>
#import <mach/mach.h>
#import <mach/mach_error.h>
#import <servers/netname.h>
#import "~/Library/mydriver.h"
}

static isFresh = 1;

int MYServerFunction(id self)
{
/*
1) Updates the next time series with data from the server.
2) Assigns data within present graphic reference bounds (VVgRBounds).
*/

// Get and set some signal object parameters ...
```

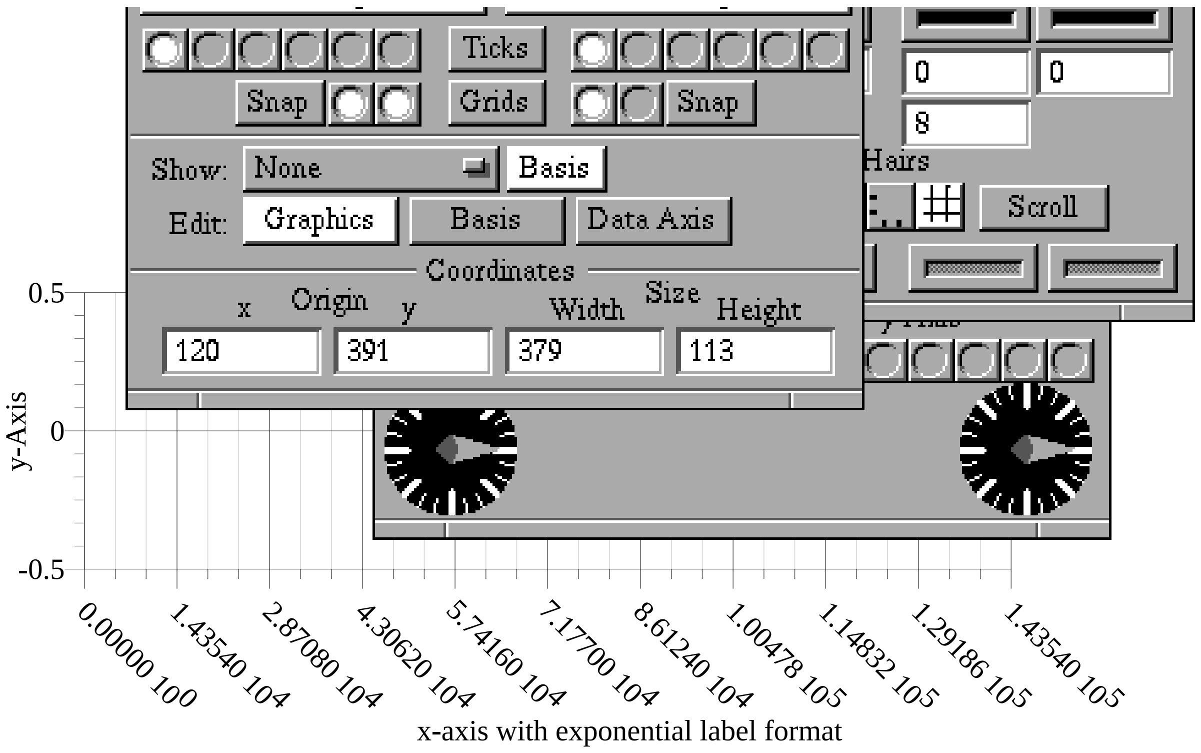
- An archaeologist asks, “What artifacts are found near *that* (pointing to it) level?”
- A seismologist asks, “What is the geology corresponding to *that* (pointing to it) time of arrival contour?”



Polar Plots - {θ, r}

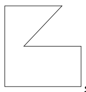
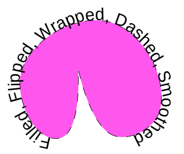
Polar axes are implemented for normal form elliptic coordinates. The ticks scale to retain their length as a function of theta to produce a true-elliptic coordinate axes. Array data defined in {θ, r} (x in units of degree, y unitless) can be placed on a rectangular or polar coordinate. A few examples are shown here:





Coordinates
Origin x: 120, y: 391
Size Width: 379, Height: 113

The axes main inspector editor is shown as the top most inspector in the preceding figure. The others are sub-inspectors which control other attributes of the axes. In addition, the inter-inspector link permits refocusing to closely related components of the graphic. It is quite conceivable to entirely edit a graph with only the inspectors.

Graphics are created by pasting, dragging file icons, mouse and keyboard editing, acquisitions, or computations. Once a graphic is created parameter adjustments may be performed. Parameters are adjusted point wise and collectively. The mouse controls point adjustments such as add, delete, and move. The inspector controls more global options like color, smoothness, and rotation. The following illustrates this procedure by changing this simple cubic bezier graphic, , to this final result: 

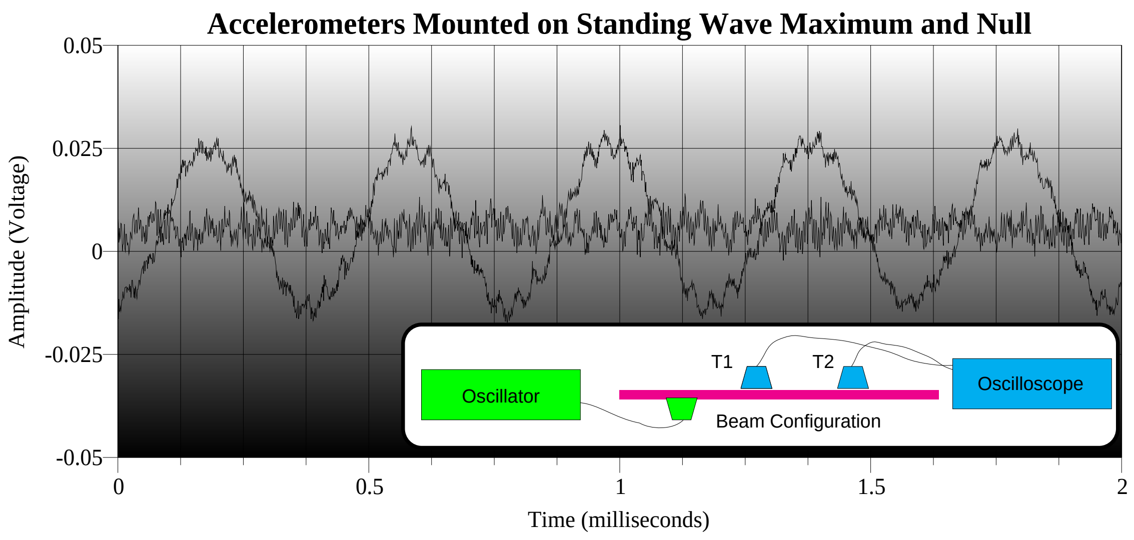
. First the simple cubic bezier graphic was made by mouse drag and click methods. The vertices were lined up with the snap to grid option to produce the “squared off” look. Then it was copied, pasted, smoothed, rotated, filled, dashed, and wrapped with text. Finally two spline knots were deleted via mouse shift-click on knot knobs to give a cusp at the bottom which gives the final result.

Other Coordinates

Contours - {x,y} vs. z

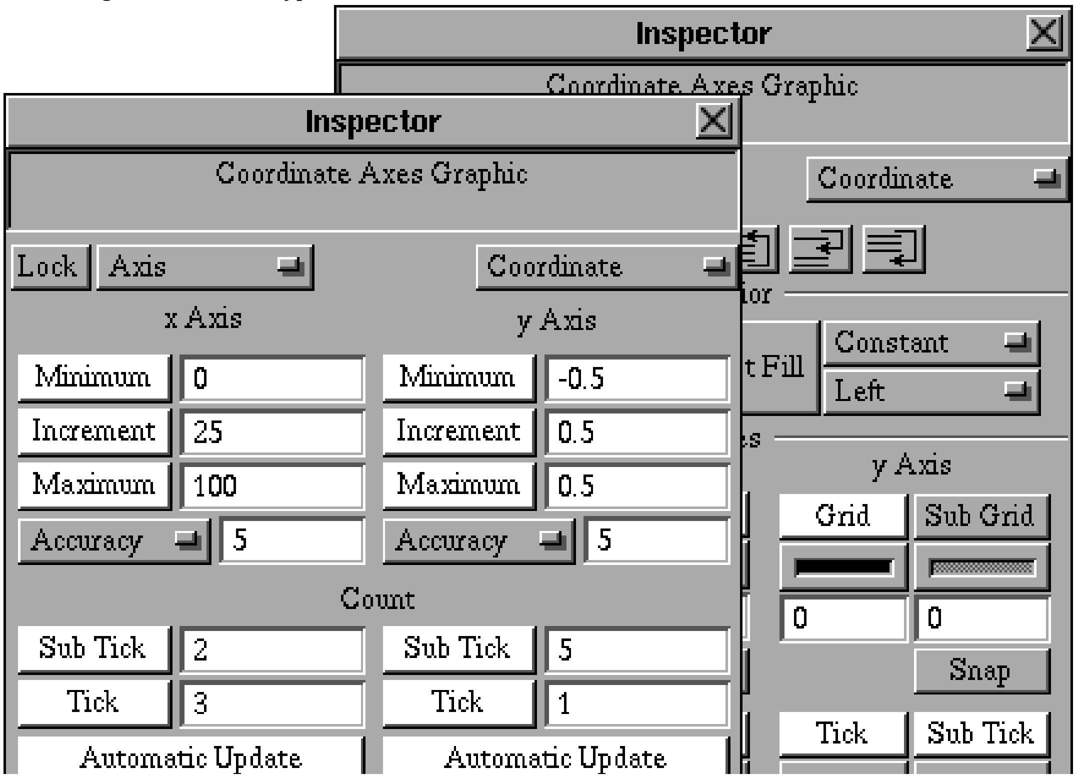
Contouring is an exceptional quantitative way to relay information as long as the correct features are properly highlighted and marked. In the figure below a single contour level is thickened and colored. The contour algorithm has been programmed to take advantage of Display Postscript, NEXTSTEP, and GraphBuilder’s interface. The following graph took three seconds (on a NeXT Turbo Color) to compute and display. It is the contour representation of over 16,000 points (128x128). The contour level to edit may be selected via the inspector or by simply clicking the mouse button while the cursor is over the desired level curve. The contour graphic issues action messages which permit targets to access databases indexed with level keys. Now questions such as the following may be answered in an intuitive, manageable, and immediate way.

Sometimes a story must be relayed. "*Accelerometer T1 is mounted at the center of the beam and T2 at the quarter beam length. This figure shows the frequency at a quarter multiple of the beam length. As you can see the null ...*". The following figure shows how GraphBuilder can be used to make a pictorial representation of that description.



The Mechanics of Altering Graphics

The inspector editors can be used to adjust attributes. The coordinate axes inspector editors are shown in the following figure. These are a few of dozens of inspectors which permit graphic attributes to be changed interactively. The axis editors control attributes such as x and y label values, scientific, fixed, and free label format, labels on/off, grids and subgrids on or off, data axes type, grid and tick widths and color, label rotation angle, axes size, fill type, and a whole host of other attributes.

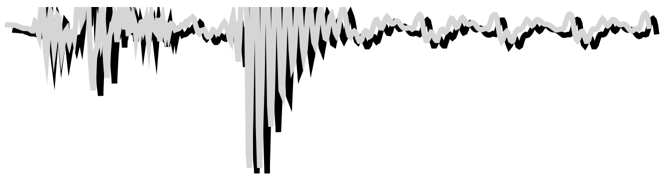


	x Axis		y Axis
Minimum	0	Minimum	-0.5
Increment	25	Increment	0.5
Maximum	100	Maximum	0.5
Accuracy	5	Accuracy	5

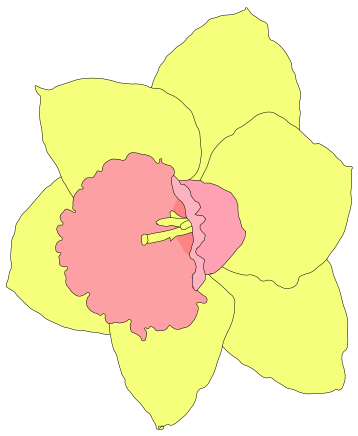
Count

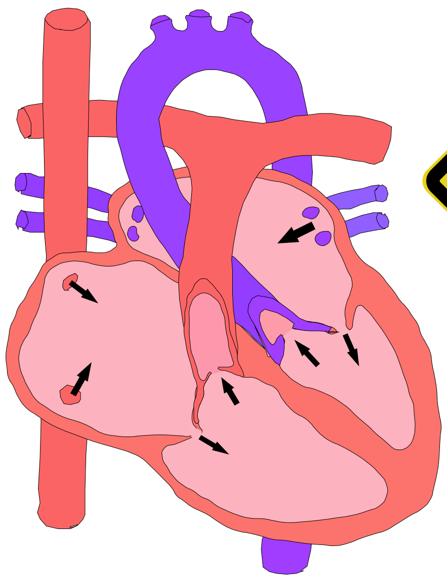
Sub Tick	2	Sub Tick	5
Tick	3	Tick	1

Automatic Update　Automatic Update

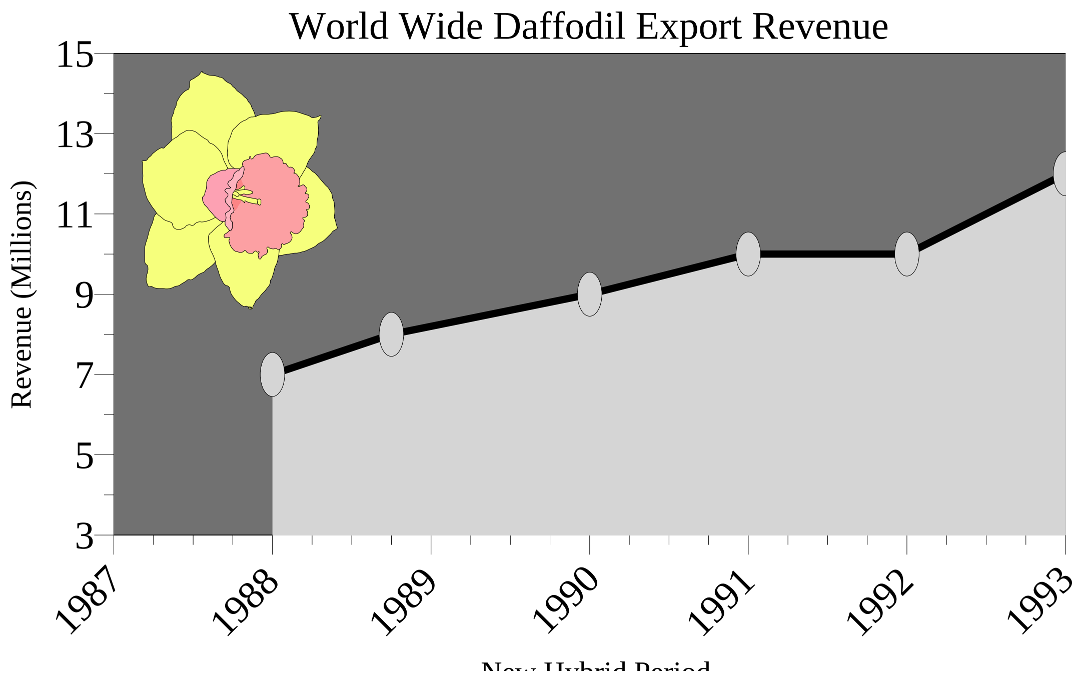


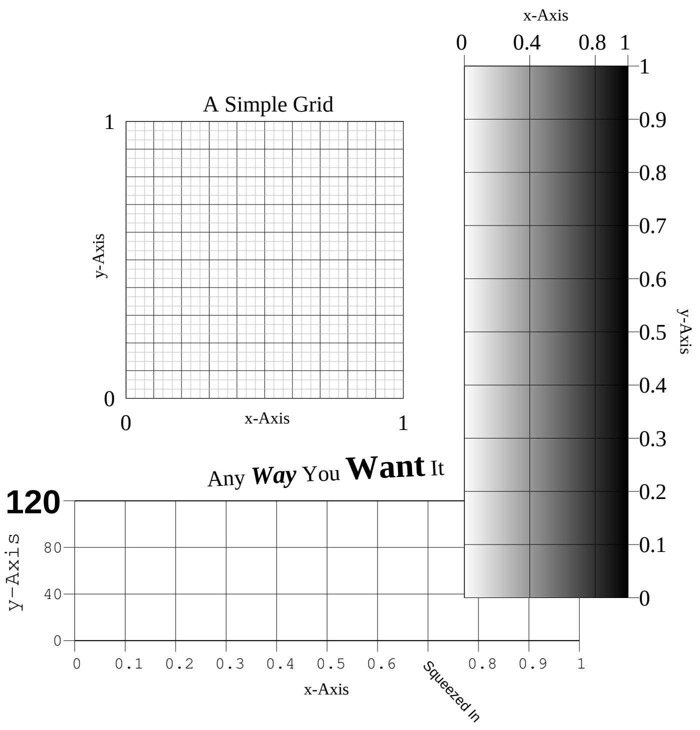
Composite graphics make interesting and useful art. This daffodil was made in just a few minutes whereas the heart took a little longer.





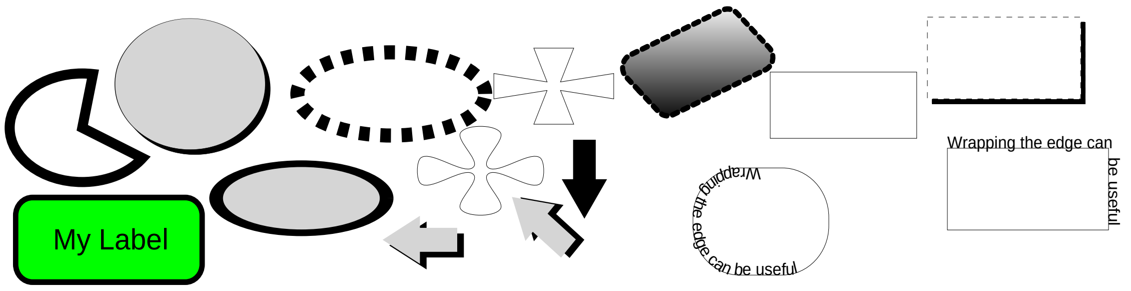
This art has been imported to the following chart. The x-axis labels are rotated by clicking a dial in the axes inspector. Labels can be aligned to any specification using common formatting operations. The markers are made by pasting an ellipse to the array editor marker matrix.



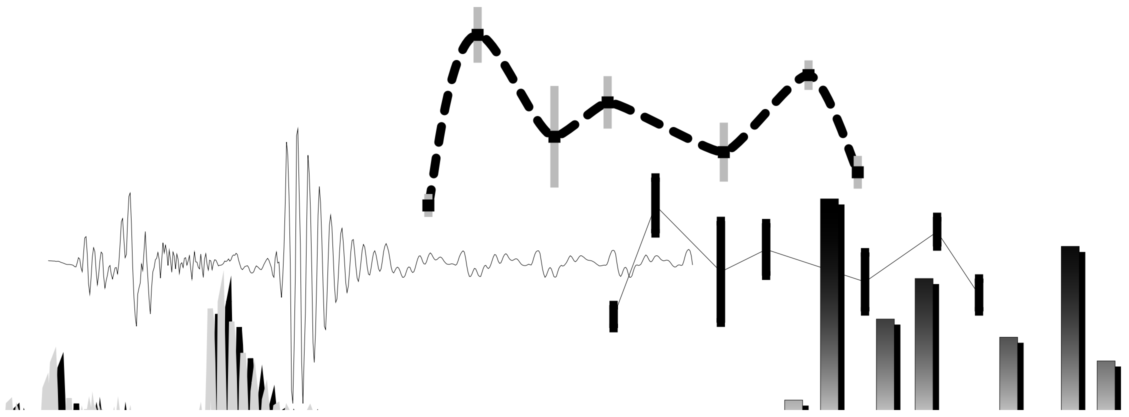


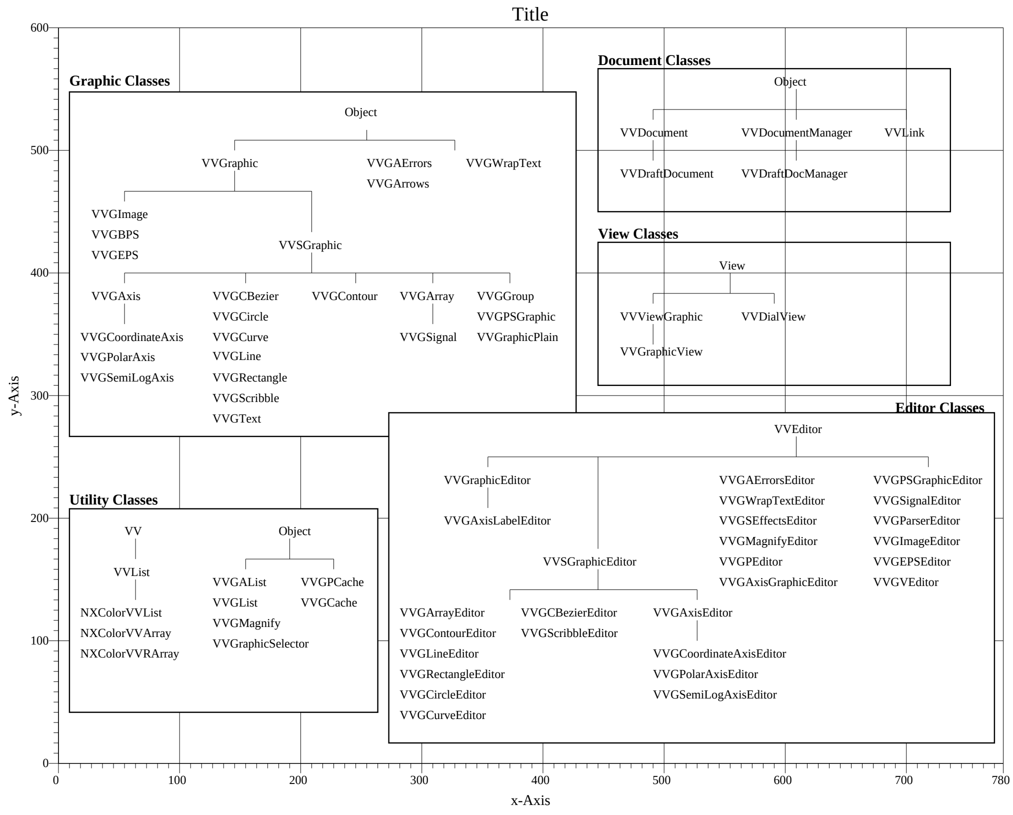
Drawing, Graphing, and Plotting

GraphBuilder combines quantitative graphing with the the drawing capabilities of Drafter. The ability to make graphics for labels, markers, highlighters, annotation, etc. is virtually unlimited. Here are a few examples:

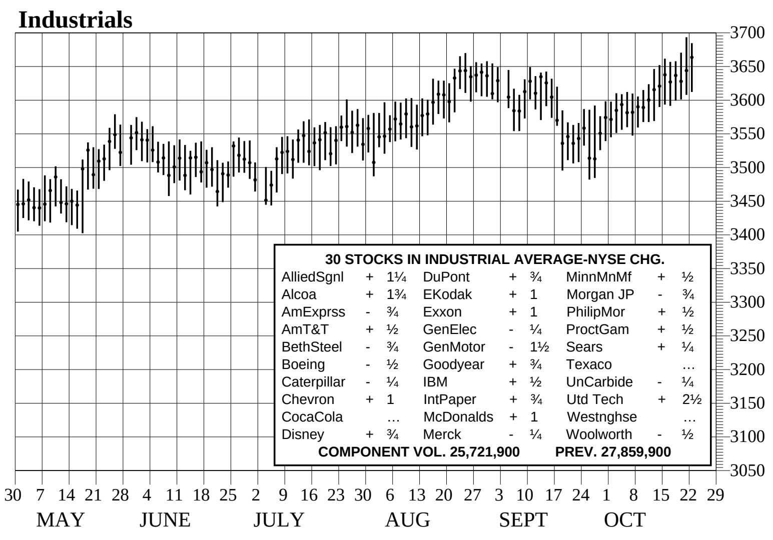


Data graphics have graphic features of non-data oriented graphics. These effects can make graphs more readable and attractive while retaining the quantitative nature of the representation. The figures below show a few possibilities.





The following figure shows different axis label options. Axis components such as labels, markers, and titles can be altered just like any other graphic. Your own axis labels (in this case MAY, JUNE, ...) can be added using point and click methods. The y-axis labels are on the right, the default is on the left. The close markers for the hi-low bar are made by copying a graphic, in this case a 2-point in diameter filled circle, and pasting it to the marker matrix in the array graphic inspector editor. Since markers are made from graphics the marker types are virtually unlimited.



30 STOCKS IN INDUSTRIAL AVERAGE-NYSE CHG.

AlliedSgnl	+	1¼	DuPont	+	¾	MinnMnMf	+	½
Alcoa	+	1¾	EKodak	+	1	Morgan JP	-	¾
AmExprss	-	¾	Exxon	+	1	PhilipMor	+	½
AmT&T	+	½	GenElec	-	¼	ProctGam	+	½
BethSteel	-	¾	GenMotor	-	1½	Sears	+	¼
Boeing	-	½	Goodyear	+	¾	Texaco		…
Caterpillar	-	¼	IBM	+	½	UnCarbide	-	¼
Chevron	+	1	IntPaper	+	¾	Utd Tech	+	2½
CocaCola		…	McDonalds	+	1	Westnghse		…
Disney	+	¾	Merck	-	¼	Woolworth	-	½

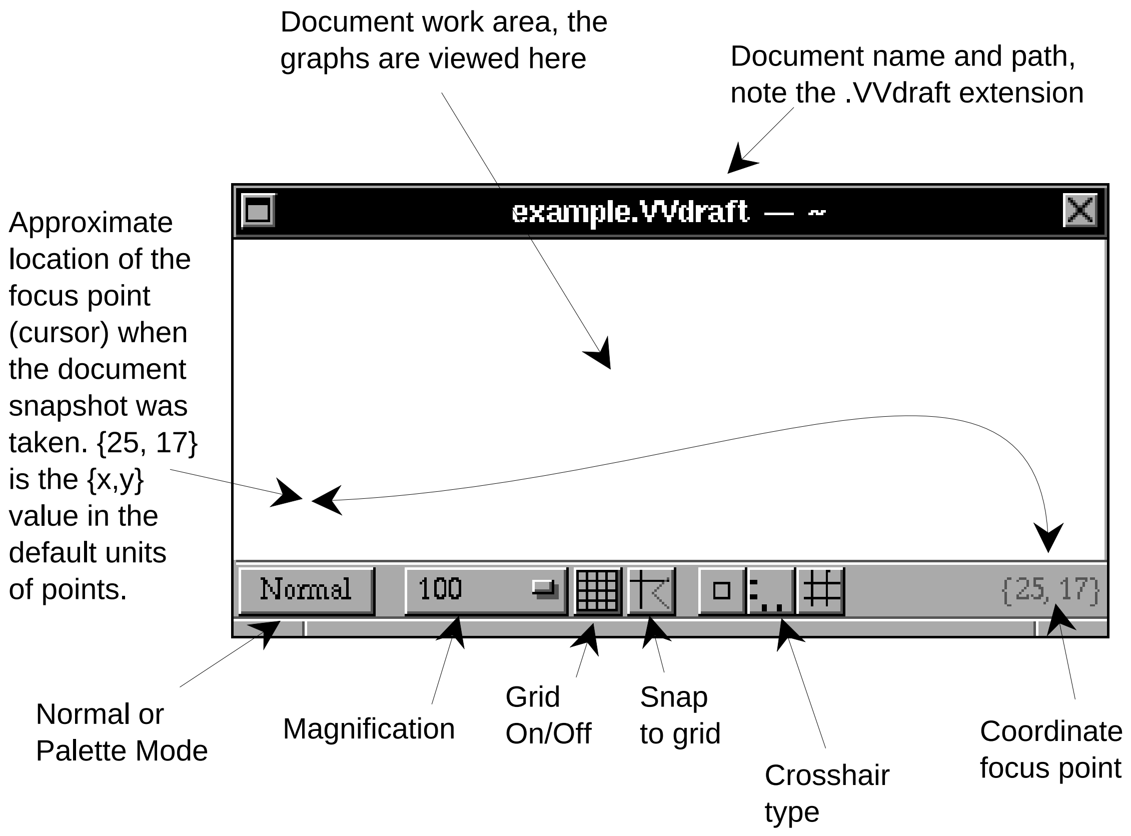
COMPONENT VOL. 25,721,900 PREV. 27,859,900

Graph Layout

Graph layout is essentially transparent. Just push the mouse button down and drag out axes where you want them. The following example exhibits a few possibilities.

Make Great Graphs Quick

Informative and well devised graphs are easy to make with GraphBuilder. Quantitative graphs can be combined with text and graphics providing all the tools needed to create presentation quality documents. When GraphBuilder is launched from the WorkSpace Manager a blank document window, such as the one below, is brought forward.


Document work area, the graphs are viewed here
Document name and path, note the .VVdraft extension
example.VVdraft — ~
Approximate location of the focus point (cursor) when the document snapshot was taken. {25, 17} is the {x,y} value in the default units of points.
Normal
100
{25, 17}
Normal or Palette Mode
Magnification
Grid On/Off
Snap to grid
Crosshair type
Coordinate focus point

This document window is an interface to a UNIX file system directory and contained graphics. Files relating to the document are maintained in the directory whose name appears in the title bar of the window. GraphBuilder's main function is to provide an interface between the user and this document. To make graphics the user can select a factory cell from the Graphic Selector and then push, drag, and click the mouse as needed within the white portion of the document window. To make a graph select the axes cell in the Graphic Selector, , and drag out the axes. Double click within the axes frame to focus on the coordinate system defined by the axes and then start adding information. The following figure shows a diagram made by GraphBuilder. The graphics are added by click and type methods. The diagram resides on a coordinate system defined by the axes so the elements can be snapped to the axes discretization (the subtick intervals). To align the elements of the diagram "snap lower left corner to grid" was used while all the graphics were selected.

814-234-9613
814-234-9614 Fax

VVI
Data Control Specialist

311 Adams Ave.
State College, PA 16803

GraphBuilder

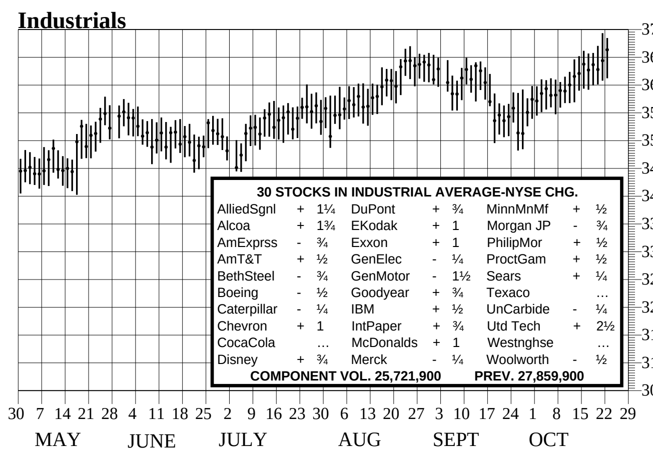
Overview and Product Sheet

The application GraphBuilder delivers easy to use and professional interactive, animated, and programmable graphing for end users and developers. It is used for trading systems, forecasting, engineering and scientific data display, and any process where quality data presentation and control software for production or interactive data needs are required.

GraphBuilder enhances your projects by providing:

- Reliable, easy to use, powerful, and optimized performance.
- Substantial savings in product development and graph composition.
- Access to your data in a quality and interactive format.
- Substantial user interface, data importing/exporting, and general graphic features and effects.

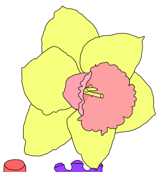
Graphs and figures, such as those below, are effortlessly constructed without extensions. Animation and user selection are built-in producing an implementation responsive to changing data and your changing requirements. All figures in this overview have been made using GraphBuilder.

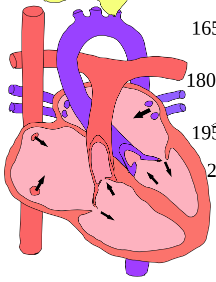


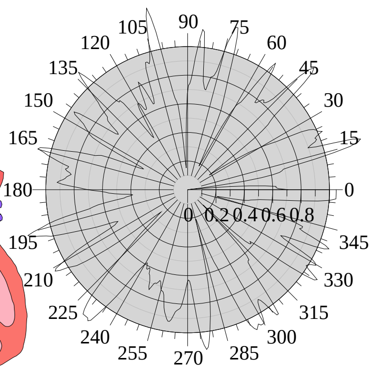
30 STOCKS IN INDUSTRIAL AVERAGE-NYSE CHG.

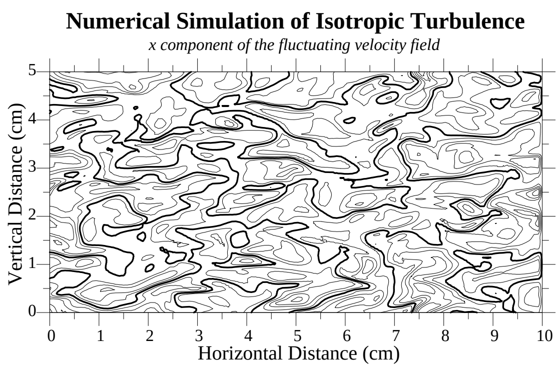
AlliedSgnl	+	1¼	DuPont	+	¾	MinnMnMf	+	½
Alcoa	+	1¾	EKodak	+	1	Morgan JP	-	¾
AmExprss	-	¾	Exxon	+	1	PhilipMor	+	½
AmT&T	+	½	GenElec	-	¼	ProctGam	+	½
BethSteel	-	¾	GenMotor	-	1½	Sears	+	¼
Boeing	-	½	Goodyear	+	¾	Texaco		…
Caterpillar	-	¼	IBM	+	½	UnCarbide	-	¼
Chevron	+	1	IntPaper	+	¾	Utd Tech	+	2½
CocaCola		…	McDonalds	+	1	Westnghse		…
Disney	+	¾	Merck	-	¼	Woolworth	-	½

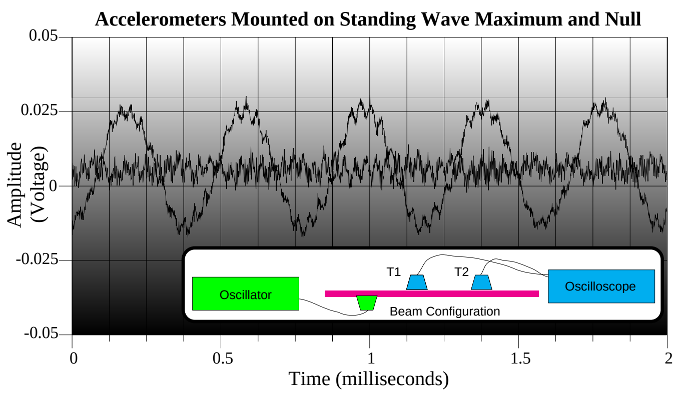
COMPONENT VOL. 25,721,900 PREV. 27,859,900






90
75
60
45
30
15
0
345
330
315
300
285
270
255
240
225
210
195
180
165
150
135
120
105
0 0.2 0.4 0.6 0.8


Numerical Simulation of Isotropic Turbulence
x component of the fluctuating velocity field
Vertical Distance (cm)
Horizontal Distance (cm)


Accelerometers Mounted on Standing Wave Maximum and Null
Amplitude (Voltage)
Time (milliseconds)
Oscillator
T1
T2
Oscilloscope
Beam Configuration