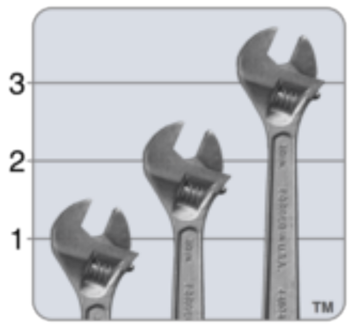
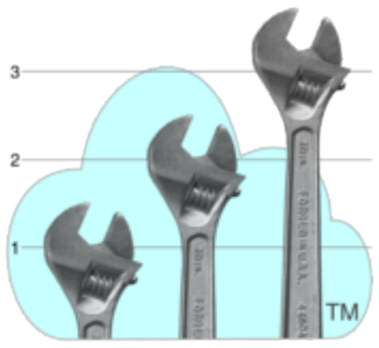
Graph IDE™ User Manual

Revision 5.5.3 (12.12.3)
— July 5, 2022 —
By VVI ®
www.vvi.com

Standalone



Cloud Enabled



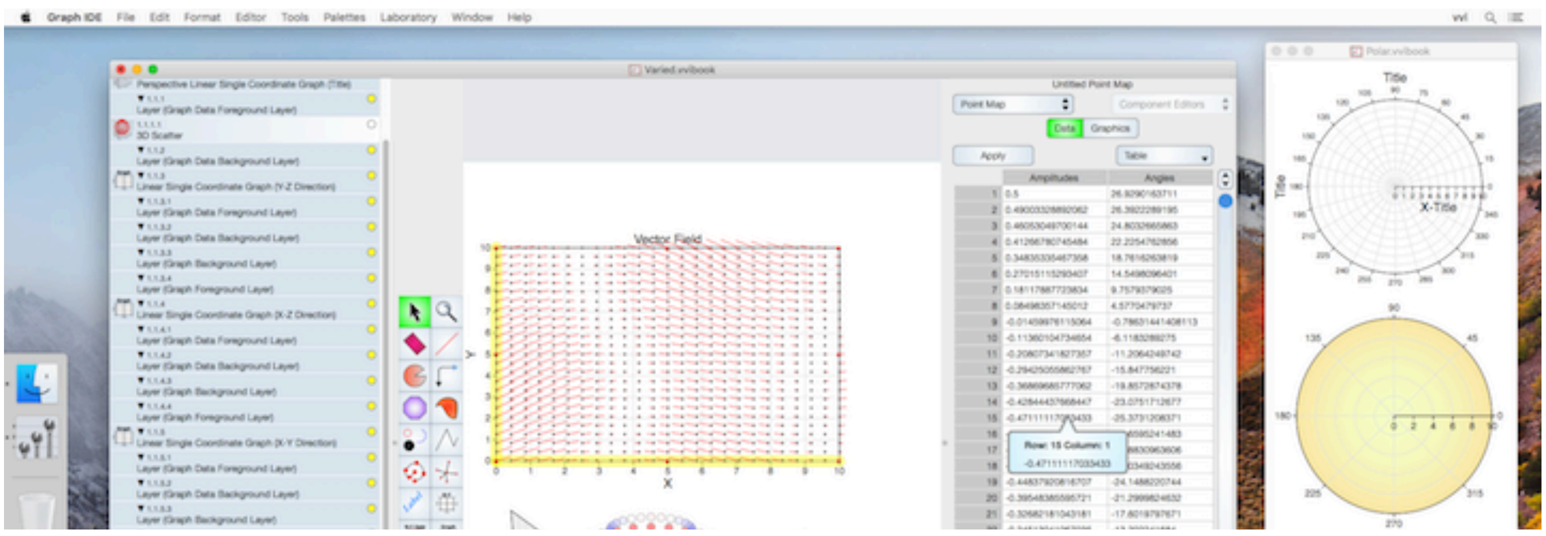
Graph IDE is the powerful data visualization application for Mac, iPad, iPhone and any web browser. It is rich in graphic creation, editing and programming to facilitate the visualization of information. To install Graph IDE on your device see: Install or login right now, without installing anything, using the following fields:

Username:	Username	Password:	Password	Login

To obtain an account see: Cloud Account.

Select a tab to see Graph IDE on the different platforms

Mac iPad iPhone Web



Graph IDE is very easy to use. That is because abstract concepts are recast into a uniform way of dealing with data. The Movies are an excellent way to see that abstraction in action thus instantiating the concepts of Graph IDE and making its features immediately accessible.

Graph IDE includes a large collection of Basic, Graph, 2D Data Graphic and 3D Data Graphic drawing objects that can be created via premade and custom Palettes, Chart Tasks, Direct Tool Creation Modes, several specialized interfaces in the Laboratory and many other ways. Graphs, data and graphics can be programmed, see the Programming section. To start understanding Graph IDE see Preamble.

If you wish to proceed to classical usage then first consult the Spreadsheet section to enter data and make a graph. Spreadsheets are powerful and are typically the main feature of a data analysis application but are only a fraction of possibilities incorporated within Graph IDE.

Graph IDE, and the underlying codebase, is the result of three decades of development and work in the area of data visualization. Even so, it is only the "tip of the iceberg". To broaden its applicability it also implements rapid development of other classes of data visualization applications. See Custom Application for additional information.

Graph IDE ► Primer

Sections throughout this manual describe how to solve various common tasks and the various components of Graph IDE, but they are not targeted to any one way of doing things. This Primer section can get you going fast.

The following is a brief list of Primer sections:

Section	Description
Importing Data	Describes how to import data into Graph IDE.

Graph IDE ► Primer ► Importing Data

There are just too many ways to import data and descriptions of those ways are scattered throughout this manual. Even though data importing is a common task and it is just "obvious" how to do it (once you know how), it can be daunting at first. So, lets describe succinctly one very common task here.

Lets start off with a CSV file whose contents are shown here:

```
Sector, Gross (M), Index
Oil & Gas,12.09633,1
Metals & Mining,5.65675,2
Financial,10.824,3
Utilities,8.547437,4
Real Estate,8.414734,5
Telecom,3.37747,6
Infrastructure,12.1683,7
Industrial,11.2435,8
Pulp & Paper,10.67935,9
Transportation,15.56745,10
```

and included within this manual here: Sector.csv. You can save that file to your local file system to work with it.

On a Mac, in the Finder, right-click on the file and select the menu Open With ► Graph IDE.app. If Graph IDE does not show up then use the Other... item and navigate to the Graph IDE app and use that. You may also want to set Graph IDE as the default app to use in which case you need only double-click the file to view it in Graph IDE.

Once you choose to open with Graph IDE then Graph IDE is launched with a document containing a Spreadsheet and the Import Selector will come forward. If the first row in the CSV file is a header row, as in this case, then select the First Sequence is Header switch and then select the Import To Table button.

Your data is now imported into a spreadsheet on a Document. When you save that document then your data is saved as well.

To see a pie chart of the data click on the second column header (the top grey cell named Gross (M)) of the spreadsheet (to set that column as the Amplitudes Column in the inspector) and within the Spreadsheet inspector, select the Representations ► Pie Chart menu.

You now have a basic spreadsheet and pie chart within the document. If you click on a pie section of the pie chart then you will be able to change that piece of data using the Data Selector. However, in this primer lets proceed a different way. In the spreadsheet inspector, make the Event Qualifier inactive. That places the document in a normal layout mode. Then click once on the pie chart to select it and move it around and resize it as desired. Then type command-2 which brings forward the Pie Chart Arranger. In the lower part of that arranger, choose column 1 for the Label column.

You are now well on your way to making the perfect pie chart; but recollect that there is no perfect pie chart or rather the perfect visual of your data is defined by you and not by this manual. That is the juncture where things get intricate. You may have specific guidelines on coloring your pie chart and need to use the Color Selector. Perhaps your pie chart will need a certain stroke width for each wedge section in which case you need to restrict editor to Wedges and then use the Group editor to set the stroke width, color and other attributes. There are simply too many ways to proceed to define all the options within this primer. However, here are a few guidelines:

Note these following guidelines:

- If you wish to place your data onto a graph then make the Event Qualifier inactive, choose a graph type from the Graphic Selector for example the Linear Graph cell. Then drag out a Linear Graph onto the Graphic View and near the spreadsheet. Then click on the spreadsheet and select a different representation such as a Line Graph. That will place a Function graphic on the graph's data layer.
- Once you have the visualization that you wish then consider making Palettes with your own specifications.
- Instead of first importing the data into Graph IDE, you may wish to use the many factory prototypes within Graph IDE first. For example, in the Graphic Selector choose the Spreadsheet cell and in the resulting factory inspector choose the Financial tab and drag out the Sector Allocation spreadsheet. Then in the spreadsheet inspector select the Table Controls Table ► Import From File... menu to import data.
- At the beginning of this Primer data was imported from a file which resulted in a document with a single spreadsheet. However, if you first make the spreadsheet then you can have multiple spreadsheets with their own representations all within one document. It is actually easier to make the spreadsheet first and then import data than vice versa. Also, because of security concerns, there is no way to import data into a web browser first and thus you will need to make the spreadsheet first and use the Table ► Import From File... menu.
- Notice how the predefined Sector Allocation spreadsheet referenced above has its formatters set for a particular purpose. All of those settings are not achievable by a simple CSV importing so it is best to start with the correctly formatted spreadsheet and then import data.
- When you have many columns in one spreadsheet and many spreadsheets within one document then you can use the Pie Chart arranger (or any arranger) to redirect data acquisition to different spreadsheets and columns within a single spreadsheets. That way you can observe different data sets simply by referencing them instead of making new representations. Notice how you are now operating in the reverse of the flow of logic from the start. Instead of starting from data you are now starting from a representation and redirecting to different data. It is that pattern of reversal which is difficult to comprehend at first because it is unusual.
- You can copy and paste the pie chart or spreadsheet as you wish. Doing so separately will duplicate the graphic, but not the data connection. You may also ungroup the pie chart and operate upon its elements individually and in that fashion lay it out exactly as you

wish, but the data connection is lost. There are many ways to work with spreadsheets and representations that are explained in other sections of this manual.

Importing some data and making a graph of that data can be very streamlined. When you build up your expertise and make palettes then that streamlined quality can be achieved while also making your own perfect graph. However, building up your expertise may take some time. Fortunately, Graph IDE features can be acquired incrementally as demonstrated in this primer.

See the Movies for examples on how to streamline very involved concepts and perform complex tasks with ease.

It should be noted that this primer is very user-oriented. In high-volume operations, it makes sense to Automate data importing instead of using laborious key and mouse events. Noticed how a programmed Graph IDE document can be placed on the web and its program is executed upon access thus making the web resource dynamic. A thorough description of all the programming techniques is beyond the scope of this manual.

The ultimate programmability of Graph IDE is, in fact, the Graph IDE Web App itself where all of Graph IDE is made into a dynamic web resource. That web resource is totally driven by Graph IDE documents.

Graph IDE ► Overview

Below is a brief list of overview sections. If you are new to Graph IDE then you may want to start off by reading the Preamble.

Section	Description
Account	Describes the Account, which is a way to save contents of a document to a cloud service (a server).
Document	Describes the Graph IDE document, which is the information Graph IDE stores and modifies on your hard disk.
Event Qualifier	Describes a programmable hit-detection system built into Graph IDE.
Glossary	Lists some basic words and definitions used in this manual.
Graphic Selector	The Graphic Selector is used to define graphic type and control modes.
Graphic View	The Graphic View is where you draw graphics, including graphs.
History	History is used to record user input.
Install	Explains how to buy and install Graph IDE.
Layer	Graphics that you make are stored and grouped in layers. This section describes how to use layers and where they appear in the Vvidget system.
Major Components	Shows the major components of Graph IDE's user interface.
Movies	Shows some movies of Graph IDE in action.
Navigator	The Navigator shows the tree structure of the graphics on a document.
Open Plugin	Describes how to load a plugin used for programming purposes.
Palettes	Palettes contain pre-made graphics. This section explains how to use palettes and also how to make your own.
Pages	This section describes how to use pages.
Preamble	Gives the preamble explanation of Graph IDE.
Quick Look & Spotlight	Describes the Quick Look and Spotlight plugin service.
Server	Describes the server built into Graph IDE.
Start Off	A checklist of things to watch out for when you first use Graph IDE.
Status Bar	Describes the status bar which is an area of a document window consisting of commonly used controls.

Graph IDE ▶ Overview ▶ Install

Graph IDE Web App:

The easiest installation procedure it to not install. Use Graph IDE now by logging into your VVI Cloud account here:

Username: Username Password: Password Login

To obtain an account see: Cloud Account.

Using Graph IDE through all major web browsers and operating systems is great when you simply want to use the best data visualization system available without being concerned about installing and updating.

Graph IDE MDM/DM (Sideloading):

The most controlled way to install Graph IDE is through your own device management service (sideloading). You can sideload the Mac version simply by downloading the install package referenced below. For iPhone, iPad and Web Server editions please email sales@vvi.com.

Graph IDE (Standalone):

If you have the latitude to install Graph IDE on your device then that is even better than the Web App edition. That is because Graph IDE can utilize all of your local processing power via its concurrent processing architecture. You may also have strict requirements to keep your data local to your own computer hence installing locally may be your only option.

The standalone edition does not require a login and all you data resides on your device, or optionally in the VVI Cloud.

- Mac Manufacturer Edition : Click this link to buy via the VVI PayPal Store and Download to install. The installation is from a signed install package.
- iPad/iPhone Edition : Click this link to install via the iTunes Store. The installation proceeds per the iTunes Store mechanism. If you are not on an iOS device then click this link: iPad/iPhone Edition.
- Mac App Store Edition : Click this link to install via the Mac App Store. The installation proceeds per the Mac App Store mechanism.

Graph IDE CE (Cloud Enabled):

Requires login to the VVI Cloud service (see Login). Once logged in, it is used just like the standalone edition.

- Mac Manufacturer Edition : Click this link to purchase a cloud account and then this link Download to download and install. The installation is from a signed install package.
- iPad/iPhone Edition : Click this link to install via the iTunes Store. The installation proceeds per the iTunes Store mechanism. If you are not on an iOS device then click this link: iPad/iPhone Edition.
- Mac App Store Edition : Click this link to install via the Mac App Store. The installation proceeds per the Mac App Store mechanism.

Cloud Service:

The Cloud Enabled (CE) versions of Graph IDE require login to the VVI Cloud service.

- Cloud Service : Click this link to purchase cloud service via the VVI PayPal Store.
- To setup your own cloud service so that you may use your local network email sales@vvi.com. The advantages are having a secure data repository, high-speed LAN connection, local administration of cloud accounts and an on-premises cloud service that is not shared with other companies.

Developer Editions:

The developer editions are platform agnostic.

- GitHub : Click this link to install from GitHub. The installation is the target of the Xcode project and is for the Mac, iPad or iPhone platforms.

No matter which way the software is downloaded or installed, Graph IDE is governed by the End User License Agreement which supersedes all other agreements. For enterprise and site licensing please email sales@vvi.com. If you intend an install from a custom application then please contact sales@vvi.com and consult the Redistribution Agreement. Redistribution requires a signed and valid purchase agreement.

Graph IDE ► Overview ► Start Off

This is a checklist of things to watch out for when you first use Graph IDE. You may want to used Graph IDE a bit, including reading this manual, and then come back to this section later to pickup on things that you missed.

- When you first launch Graph IDE it will present a default Document. That document is probably not configured the way you want it and you will have to reconfigure it each time you make a new document. That is unless you set the default document by following the instructions in the Document section.
- Graph IDE is multiplatform and runs on Mac, iPad, iPhone and Windows in various modes including headless. Because those platforms have different characteristics, not all features described in this manual are available on each platform. For the most part Graph IDE performs very similarly on each platform and when there is a difference then that difference is pointed out in the manual section, preferably at the top of the section.
- The first thing you should do is look at the Graph IDE ► Palettes menu. That menu is comprised of sub-menus and each item in the menu can be hovered over (do not select an item) to quickly pan the available palettes. Each of those palettes are a Document so by panning each one you can quickly see, within a couple of minutes, what Graph IDE can do. When you see a palette that you want to investigate further then click the menu item to select it. That places the palette on the screen permanently and then you can drag the graphics from that palette to another document, drop it and then work with it.
- Graph IDE has tools to make data visualization documents and when those tools are not good enough then it has bigger tools and then bigger tools until finally you are immersed in swiss-army-knife hand-to-hand combat directly with data representations. That is because Graph IDE is, for all practical purposes, a different way of doing things. It "inverts" the normal thought process in order to produce a highly scalable system. Given that as the case, you might want to start off with the Chart Tasks and enter some data and then export your representation to a Graph IDE document to mark it up.
- A lot of uses are repetitive. That is: Once a graph is made then that graph becomes a standard for your use and you will want to use that type of graph over and over again. Palettes greatly speed up the process of making standard graphs as you can make your own palettes that are accessible from the palettes menu.
- The Movies show and express things that simply can not be relayed well in this manual. That is because Graph IDE is more about an idea than a means of construction. It is the idea that is a means to an end and the movies bring out some important concepts.
- Graph IDE (aka: The face of the Vvidget project) represents just a fraction of the functionality of the underlying paradigm, less than ten percent. Embedded in Graph IDE are logic systems that haven't even been invented yet (outside of VVI) even though those logic systems were "invented" three decades ago. The Programming and Custom Application sections begins to unmask some of the nature of the underlying system. Once you become accustom to the facilities of Graph IDE then you may wish to start programming it to gain even more functionality.
- The Point Tags show some of the nature of the recursive design of Vvidget (Graph IDE) and are an interesting implementation. With the Point Tags implementation, the issue of markers are solved as an element of a graphic can be a graphic, ad infinitum.
- The Tables are a main way of entering numerical data in textual representation (base 10 numbers). The cells of a table accept atomic elements (such as two numbers for a 2D point) which is a bit unusual. You can right-click or click-hold a column header of a table to place that table in component mode which is the most usual presentation.

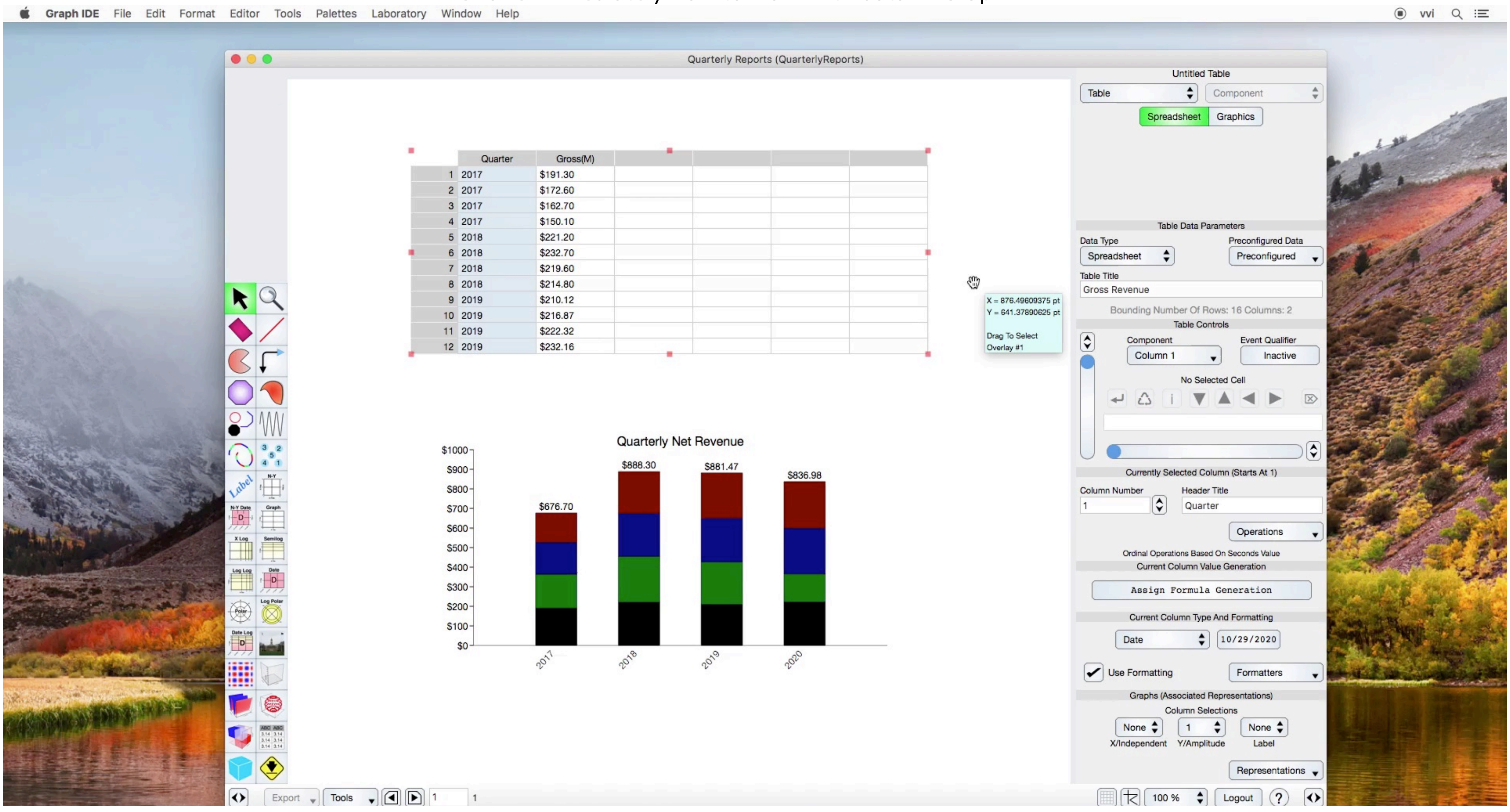
Graph IDE is a delivery system for you. It is intended to provide a kernel for your work from which you can expand upon and not be trapped into. Early versions of the Vvidget system (and Graph IDE) were so intensive (included a manual of a few thousand pages) that even the most robust team of programmers at the largest corporate sites could not digest it. This present incarnation attempts to deliver that functionality in a much more tractable way. Hopefully, as you become accustom to the features, you will find more and more value and utility.

Graph IDE ▶ Overview ▶ Movies

The following are movies showing Graph IDE in action. Click the usual controls to start a movie. To play the movies you need an Internet connection as the movies are hosted on the www.vvidget.com web site.

Some movies are on a desktop and some are within a web browser and each is applicable to platforms supported by Graph IDE.

Shows immediately how to work with data in Graph IDE

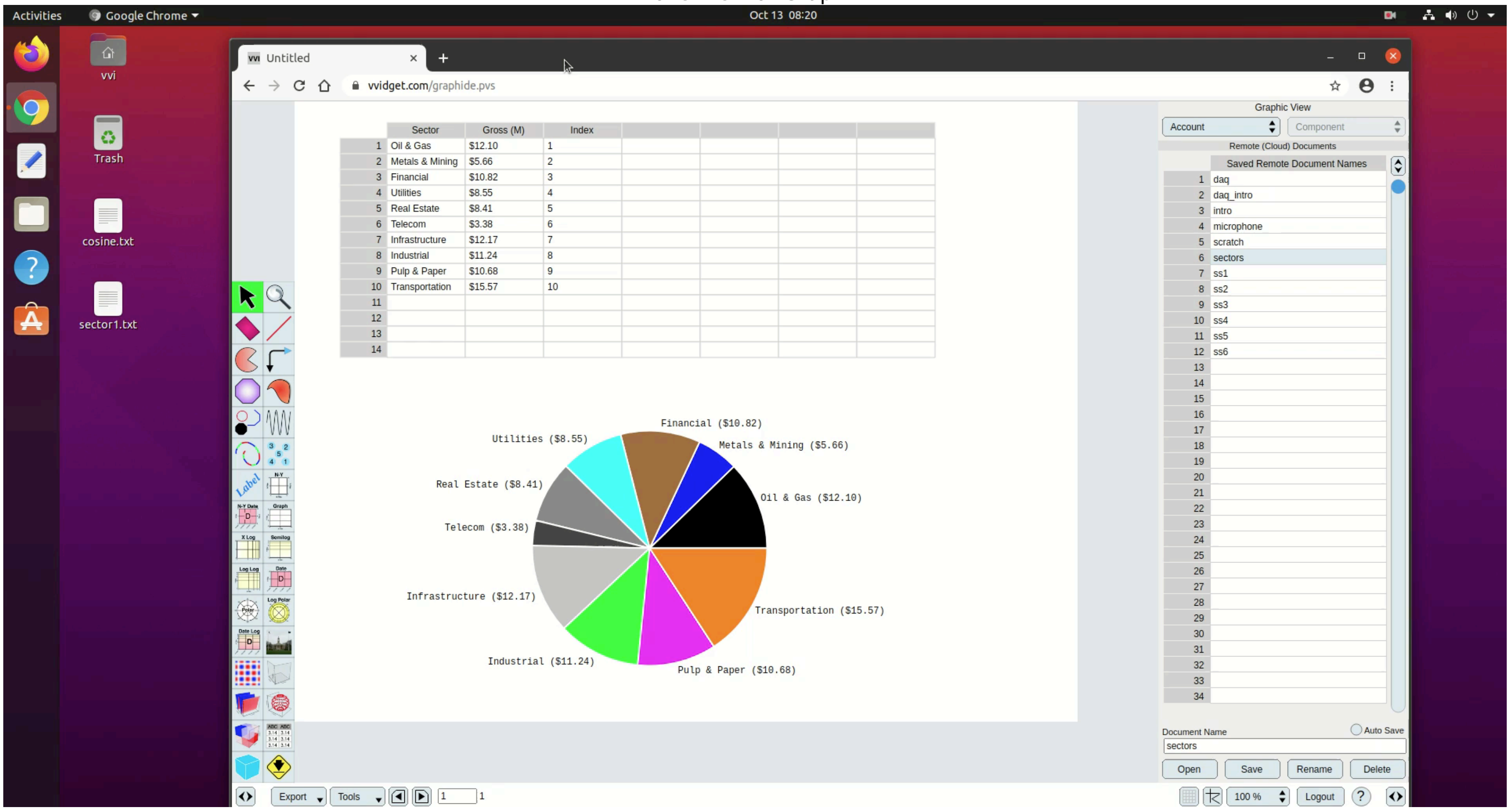


	Quarter	Gross(M)
1	2017	$191.30
2	2017	$172.60
3	2017	$162.70
4	2017	$150.10
5	2018	$221.20
6	2018	$232.70
7	2018	$219.60
8	2018	$214.80
9	2019	$210.12
10	2019	$216.87
11	2019	$222.32
12	2019	$232.16

X = 876.49609375 pt
Y = 641.37890625 pt
Drag To Select Overlay #1

Table Data Parameters
Data Type: Spreadsheet
Preconfigured Data: Preconfigured
Table Title: Gross Revenue
Bounding Number Of Rows: 16 Columns: 2
Table Controls
Component: Column 1
Event Qualifier: Inactive
No Selected Cell
Currently Selected Column (Starts At 1)
Column Number: 1
Header Title: Quarter
Operations
Ordinal Operations Based On Seconds Value
Current Column Value Generation
Assign Formula Generation
Current Column Type And Formatting
Date 10/29/2020
✓ Use Formatting
Formatters
Graphs (Associated Representations)
Column Selections
None X/Independent
1 Y/Amplitude
None Label
Representations
100 %

An overview of Graph IDE



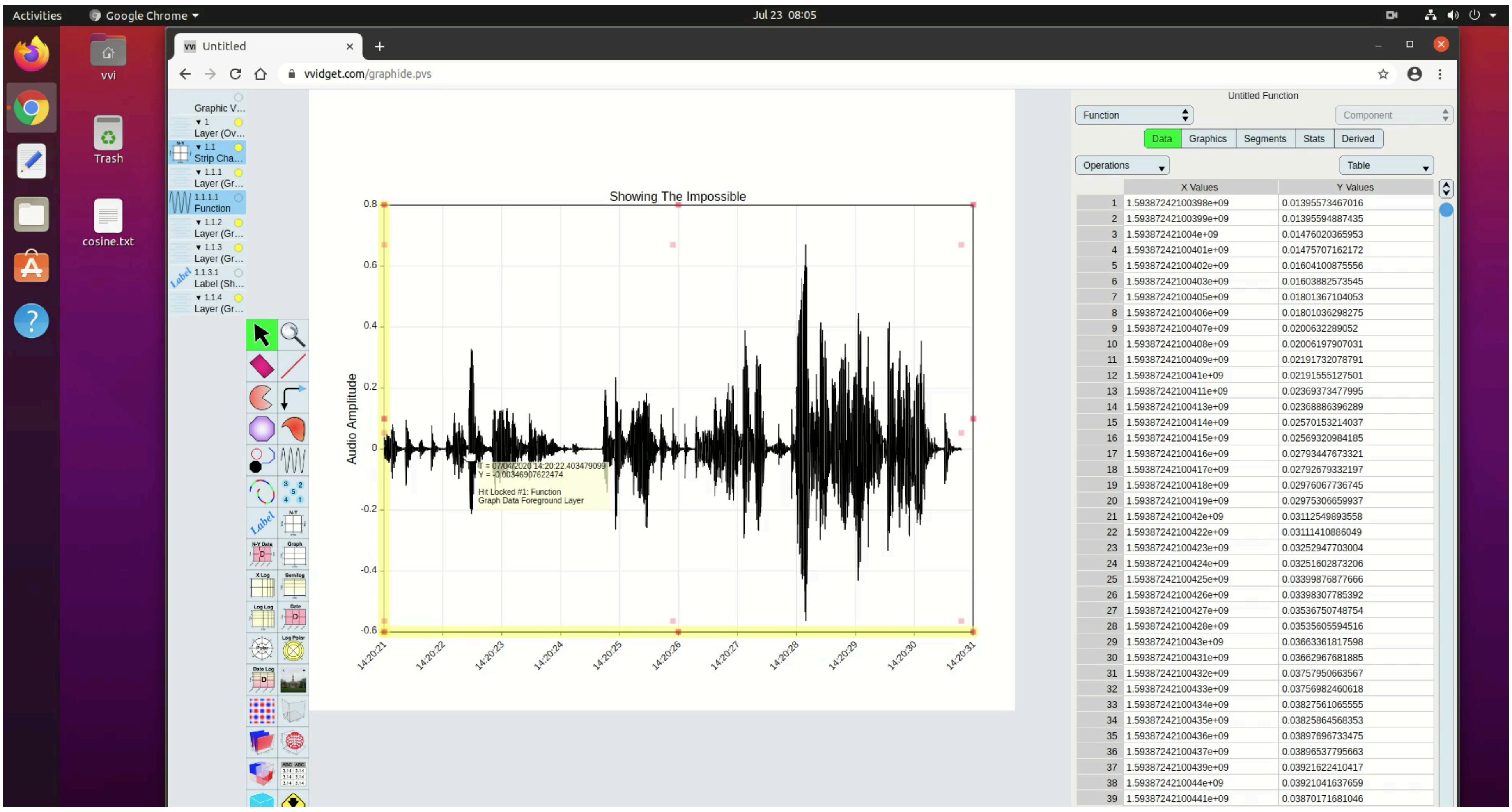
	Sector	Gross (M)	Index
1	Oil & Gas	$12.10	1
2	Metals & Mining	$5.66	2
3	Financial	$10.82	3
4	Utilities	$8.55	4
5	Real Estate	$8.41	5
6	Telecom	$3.38	6
7	Infrastructure	$12.17	7
8	Industrial	$11.24	8
9	Pulp & Paper	$10.68	9
10	Transportation	$15.57	10
11			
12			
13			
14			

Graphic View
Account
Component
Remote (Cloud) Documents

	Saved Remote Document Names
1	daq
2	daq_intro
3	intro
4	microphone
5	scratch
6	sectors
7	ss1
8	ss2
9	ss3
10	ss4
11	ss5
12	ss6
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	

Document Name: sectors
Auto Save
Open Save Rename Delete
100 %

Something that seems impossible

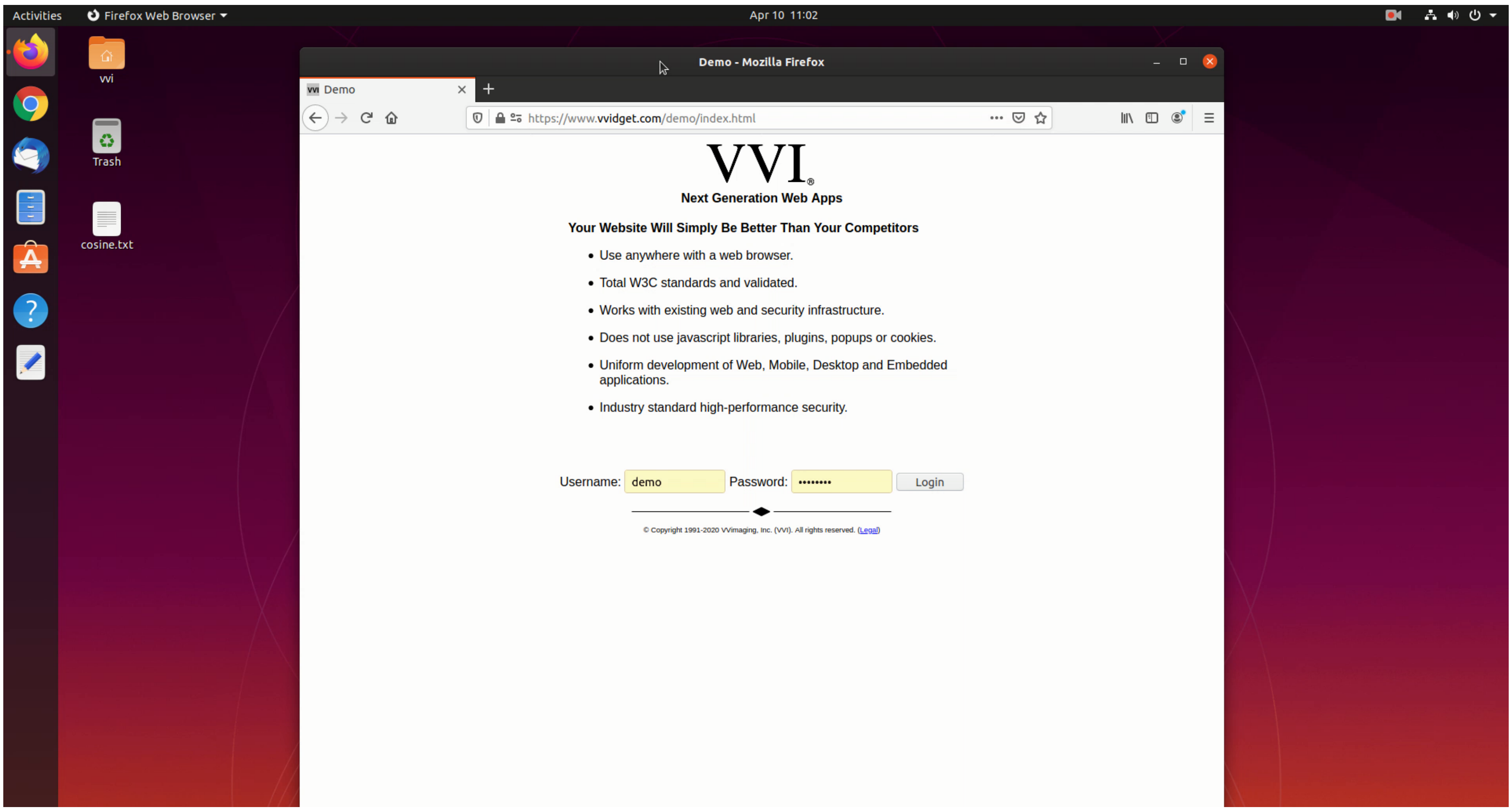


T = 07/04/2020 14:20:22.403479099
Y = -0.00346907622474

Hit Locked #1: Function
Graph Data Foreground Layer

	X Values	Y Values
1	1.59387242100398e+09	0.01395573467016
2	1.59387242100399e+09	0.01395594887435
3	1.593872421004e+09	0.01476020365953
4	1.59387242100401e+09	0.01475707162172
5	1.59387242100402e+09	0.01604100875556
6	1.59387242100403e+09	0.01603882573545
7	1.59387242100405e+09	0.01801367104053
8	1.59387242100406e+09	0.01801036298275
9	1.59387242100407e+09	0.0200632289052
10	1.59387242100408e+09	0.02006197907031
11	1.59387242100409e+09	0.02191732078791
12	1.5938724210041e+09	0.02191555127501
13	1.59387242100411e+09	0.02369373477995
14	1.59387242100413e+09	0.02368886396289
15	1.59387242100414e+09	0.02570153214037
16	1.59387242100415e+09	0.02569320984185
17	1.59387242100416e+09	0.02793447673321
18	1.59387242100417e+09	0.02792679332197
19	1.59387242100418e+09	0.02976067736745
20	1.59387242100419e+09	0.02975306659937
21	1.5938724210042e+09	0.03112549893558
22	1.59387242100422e+09	0.03111410886049
23	1.59387242100423e+09	0.03252947703004
24	1.59387242100424e+09	0.03251602873206
25	1.59387242100425e+09	0.03399876877666
26	1.59387242100426e+09	0.03398307785392
27	1.59387242100427e+09	0.03536750748754
28	1.59387242100428e+09	0.03535605594516
29	1.5938724210043e+09	0.03663361817598
30	1.59387242100431e+09	0.03662967681885
31	1.59387242100432e+09	0.03757950663567
32	1.59387242100433e+09	0.03756982460618
33	1.59387242100434e+09	0.03827561065555
34	1.59387242100435e+09	0.03825864568353
35	1.59387242100436e+09	0.03897696733475
36	1.59387242100437e+09	0.03896537795663
37	1.59387242100439e+09	0.03921622410417
38	1.5938724210044e+09	0.03921041637659
39	1.59387242100441e+09	0.03870171681046

The following is a more web apps focus on Graph IDE.



Next Generation Web Apps

Your Website Will Simply Be Better Than Your Competitors

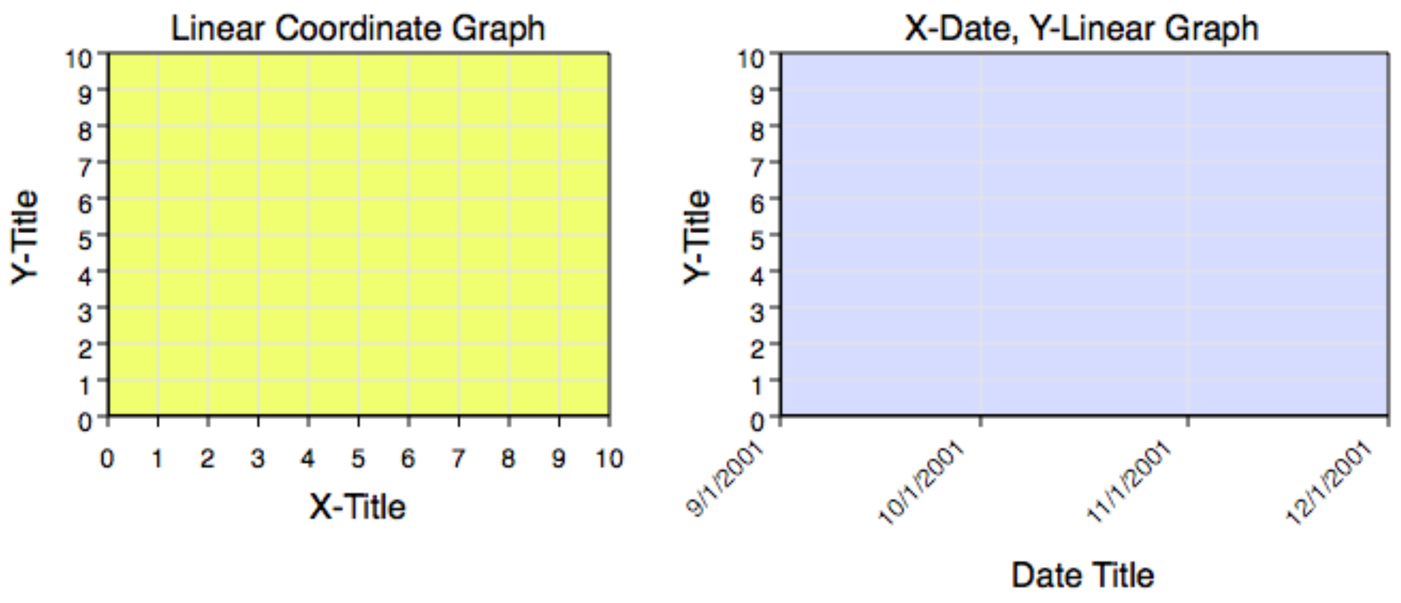
- Use anywhere with a web browser.
- Total W3C standards and validated.
- Works with existing web and security infrastructure.
- Does not use javascript libraries, plugins, popups or cookies.
- Uniform development of Web, Mobile, Desktop and Embedded applications.
- Industry standard high-performance security.

Username: demo Password: •••••••• Login

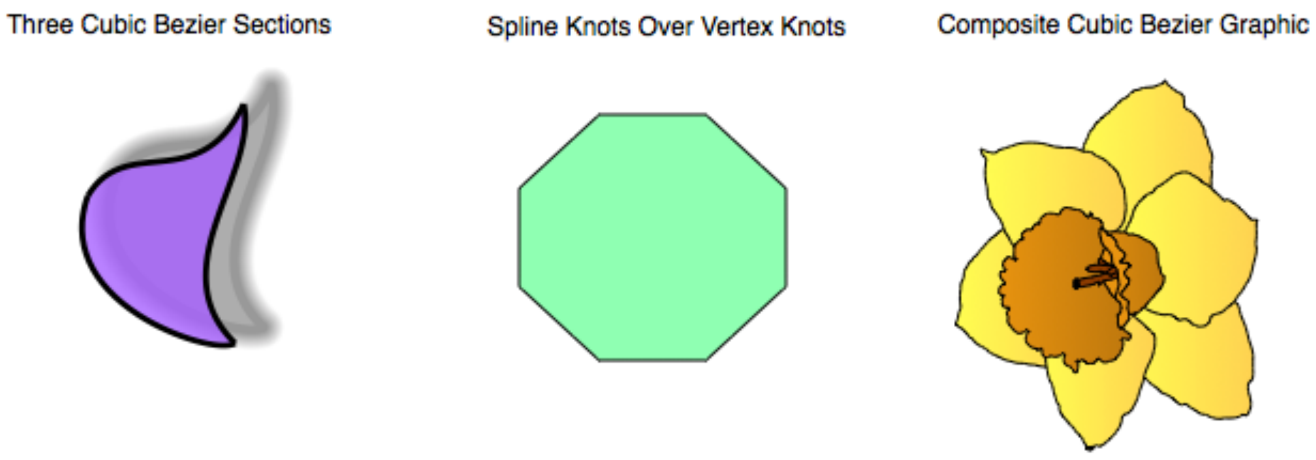
© Copyright 1991-2020 VVimaging, Inc. (VVI). All rights reserved. (Legal)

Graph IDE is an interactive drawing tool specialized to the construction of information visualization. It enables a range of visualizations from classical graphs to generic graphics. Those graphics can be combined to form unique presentations. Graph IDE also provides access to hundreds of attributes, some of which are simple and some of which are complex or recursive. Because of the large scope, Graph IDE also provides many different types of entry points.

In just a few clicks of the mouse and some typing you can make graphs like this:



and art like this:

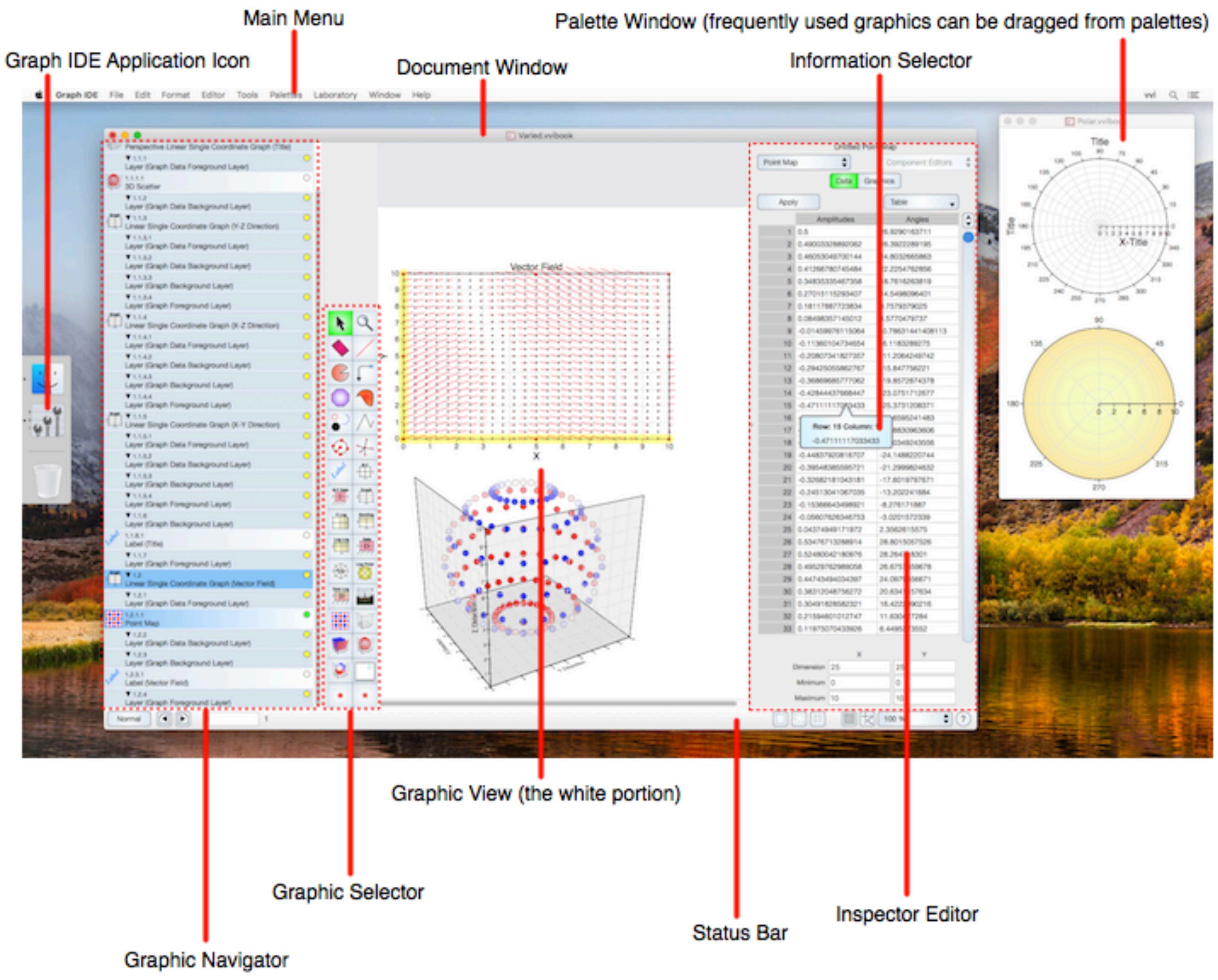

Three Cubic Bezier Sections
Spline Knots Over Vertex Knots
Composite Cubic Bezier Graphic

This manual is written as a combination of a reference guide and a user manual. As such, it defines the concepts of Graph IDE and also gives examples of real-world uses so that the reader can understand how all the features come together.

To learn about words used in this manual see Glossary. If you want to jump right in and learn about the central point of Graph IDE then see Introduction To Graphs or for a simple example graphic see Circle. The best bet would be to look at the Major Components of Graph IDE first.

Graph IDE ► Overview ► Major Components

Below is a screen shot of Graph IDE's major components:


Main Menu
Palette Window (frequently used graphics can be dragged from palettes)
Graph IDE Application Icon
Document Window
Information Selector
Graphic View (the white portion)
Graphic Selector
Status Bar
Inspector Editor
Graphic Navigator

The Graphic View is where graphs, diagrams and other visuals are drawn. You can select and focus graphics on that view. The Inspector Editor is where you can edit attributes of the focused graphic. The Graphic Selector is used to select a graphic type to create. The Document helps manage the environment you are using and has options for including edit controls directly on its window.

So, a very common operation is to click a cell in the Graphic Selector to select the graphic type, then drag the mouse over the Graphic View to create an instance of that graphic type, and finally use the Inspector Editor to modify attributes of the newly created graphic. If you need to modify the attributes of an existing graphic then you would click the "Arrow" cell in the Graphic Selector (put it into select mode), move the cursor over the graphic to select (in the Graphic View) and mouse click once to select.

Subsequent chapters in this manual detail all the various ways to create and edit graphics. Because of the complexity of some graphics (such as multi-coordinate graphs) the edit options can be overwhelming at first, but you should be able to settle in pretty quickly. The next chapter I would recommend is the Graphic View.

Graph IDE ► Overview ► History

History records user changes on a per Layer and leaf node (Graphics, etc.) basis. When attributes are altered in the inspector editor then each attribute change is recorded in a history. The history can then be viewed using the Edit menu and if desired a particular historical state can be retrieved to replace the focused graphic by clicking that history menu item or by using the undo and redo menu items.

Using History

Using the history is pretty simple but also very powerful. The following is a bullet list of some of the issues involved in the use of history.

- **Off By Default:** History is off by default and needs to be turned on on a per document basis.
- **Focus:** To scan the history of a graphic first focus on it by clicking that graphic with the mouse. Then see the History menu items in the Edit menu. To scan reordered and deleted graphics first click off all graphics (which focuses on the layer) and then use the History menu items.
- **Clear:** History is saved with the document and if the document is saved then consider clearing history before saving the document.
- **Auto Save:** If the auto save feature of a document is turned on along with history being turned on then the document is always being stored on the disk for each and every user action for which history is recorded.
- **Programming:** If the document is used for programming it is highly recommended that history be cleared before saving a document that is subsequently used for programming purposes (either that or keep history off). There are multiple reasons for doing that: (a) History probably does not need to transport with your custom application, (b) It reduces the size of an app, (c) it speeds processing and (d) if recursion is utilized along with previously reordered graphics then the algorithms that parse history state can be prone to malfunctioning. Every attempt has been made to optimize history parsing, however each new feature adds overhead and complexity. If a document is loaded into a Custom Application then part of the load process clears the history before the document is used.
- **Controls:** Some controls, such as a dial or text field, need focus and those controls have their own history or undo and redo mechanisms. In order to undo a graphic after an inspector editor change the graphic may first need to be clicked on to reestablish history focus.
- **Not Document Wide:** There is no paradigm for recording user events that work in all situations and multiple paradigms may be appropriate for a particular situation. Typically undo and redo operations can span many elements of a document and it is hard to determine what will happen if the undo (stack) is invoked because (unlike keyboard focus with a focus ring) there is no context for the undo operation save document-wide context. In Graph IDE the history context is on a per graphic and layer basis within a single graphic view. That may be somewhat atypical but is the desired implementation (having considered various implementations for years (decades)).

Graph IDE ▶ Overview ▶ Graphic View

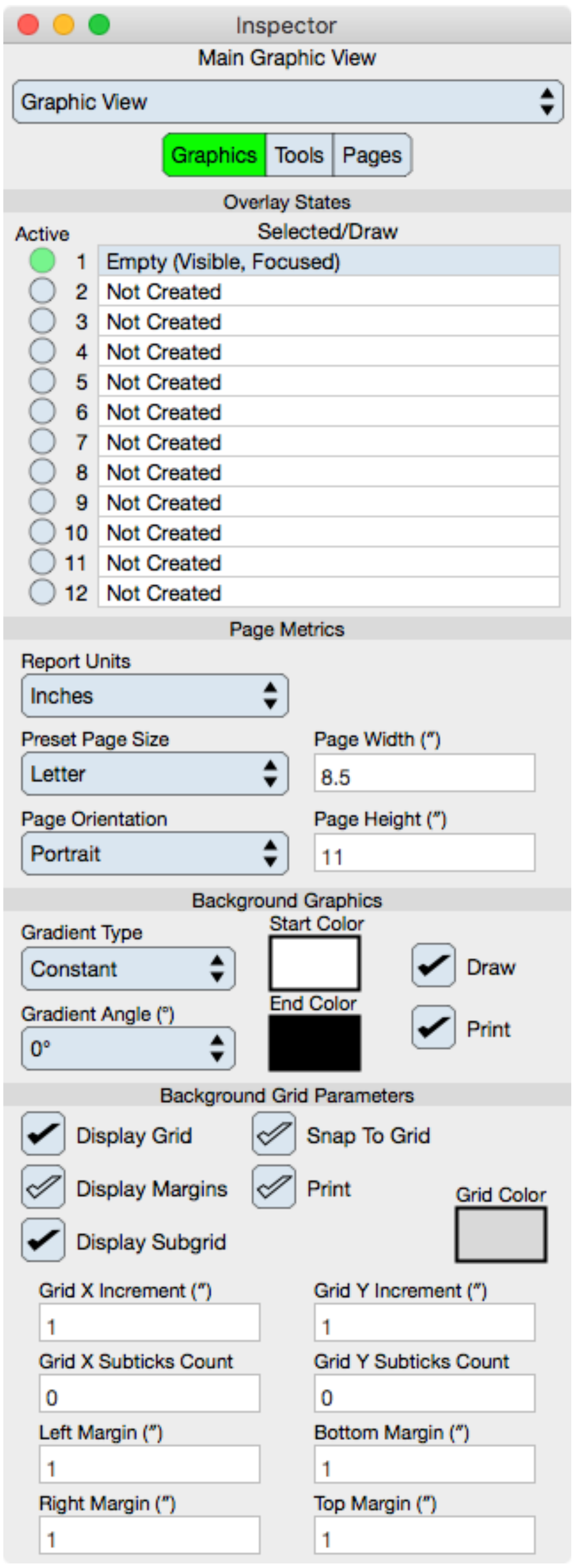
The Graphic View is the white portion of the Document Window (see Major Components for a diagram). Graphics are created and edited on the graphic view using the mouse, touch and keyboard as controls.

The graphic view delegates functionality to a sequence of Layers (Consult that section for further information) called Overlays. Overlays can be turned on or off producing the effect of "overlaying" sheets of transparencies for comparison purposes.

The following details the graphic view in terms of the inspector editor controls.

Graphics

The Graphics editor is used to adjust graphic-related attributes as detailed below. Notice that overlays are considered a graphical attribute even though they are a node-related concept. That is because all embedded graphics are contained in the overlays and those embedded graphics are graphical in nature.



Inspector

Main Graphic View

Graphic View

Graphics | Tools | Pages

Overlay States

Active		Selected/Draw
●	1	Empty (Visible, Focused)
○	2	Not Created
○	3	Not Created
○	4	Not Created
○	5	Not Created
○	6	Not Created
○	7	Not Created
○	8	Not Created
○	9	Not Created
○	10	Not Created
○	11	Not Created
○	12	Not Created

Page Metrics

Report Units: Inches

Preset Page Size: Letter

Page Width ("): 8.5

Page Orientation: Portrait

Page Height ("): 11

Background Graphics

Gradient Type: Constant

Start Color: □

✓ Draw

Gradient Angle (°): 0°

End Color: ■

✓ Print

Background Grid Parameters

✓ Display Grid

Snap To Grid

Display Margins

Print

Grid Color

✓ Display Subgrid

Grid X Increment ("): 1	Grid Y Increment ("): 1
Grid X Subticks Count: 0	Grid Y Subticks Count: 0
Left Margin ("): 1	Bottom Margin ("): 1
Right Margin ("): 1	Top Margin ("): 1

Overlay States

Active Overlays : The graphic view has twelve possible Layers which are called overlays. Those overlays can be made active (display turned on) or inactive (display turned off).

Overlay Table : Shows the status of each overlay. Select a cell to focus on that corresponding overlay. Only the focused overlay can be altered with Controls, namely the one that accepts mouse or touch events. Only one overlay can be focused at a time.

Page Metrics

The Graphic View's area is the page area by definition and is reported in the report units as specified below.

Report Units : Defines the units that are reported in the Cursor Information and also the Spatial Metrics section of the Graphics editor.

Preset Page Size : A list of commonly used sizes for the page.

Page Orientation : The dimension of the page is normally Portrait. However, it can also be set to Landscape which means that the x and y values are transposed. Notice that this distinction will not transpose the embedded graphics, rather it simply transposes the page coordinate length values.

Custom Width and Height : The Graphic View can be set to a custom width and height by entering those values into the corresponding text fields. The values are in report units.

Background Graphics

The background graphic is the Rectangle that provides the backdrop for the graphic view.

Draw : Normally the background is always drawn and should be left that way. If the graphic view is associated with a non-rectangular template then the graphic view background should be turned off.

Print : If on, the background prints otherwise it does not print.

Start Color : Specifies the Color of the background.

End Color : Specifies the gradient end Color. This only has an affect if the gradient type is non-constant.

Gradient : The background graphic is normally a solid color. The gradient attributes specify a function and orientation of a color change from the start color to end color.

Background Grid Parameters

The background grid is the Graph that provides an alignment grid for the graphic view.

Snap To Grid : When on certain movements of graphics such as dragging and cursor (arrow) keys increment according to the subgrid locations.

Display : Turns components of the grid on or off.

Print : When on, the grid prints along with the other graphics. Normally, the grid is not part of the print definition.

Margins And Grid : The Grid defines a discretization of the Graphic View's drawing area. When you snap to grid (to the subgrid actually), then that discretization is active in addition to simply displayed. The margins define the boundary of the grid, and do not define the margins of a page. Margins are a convention and not enforced.

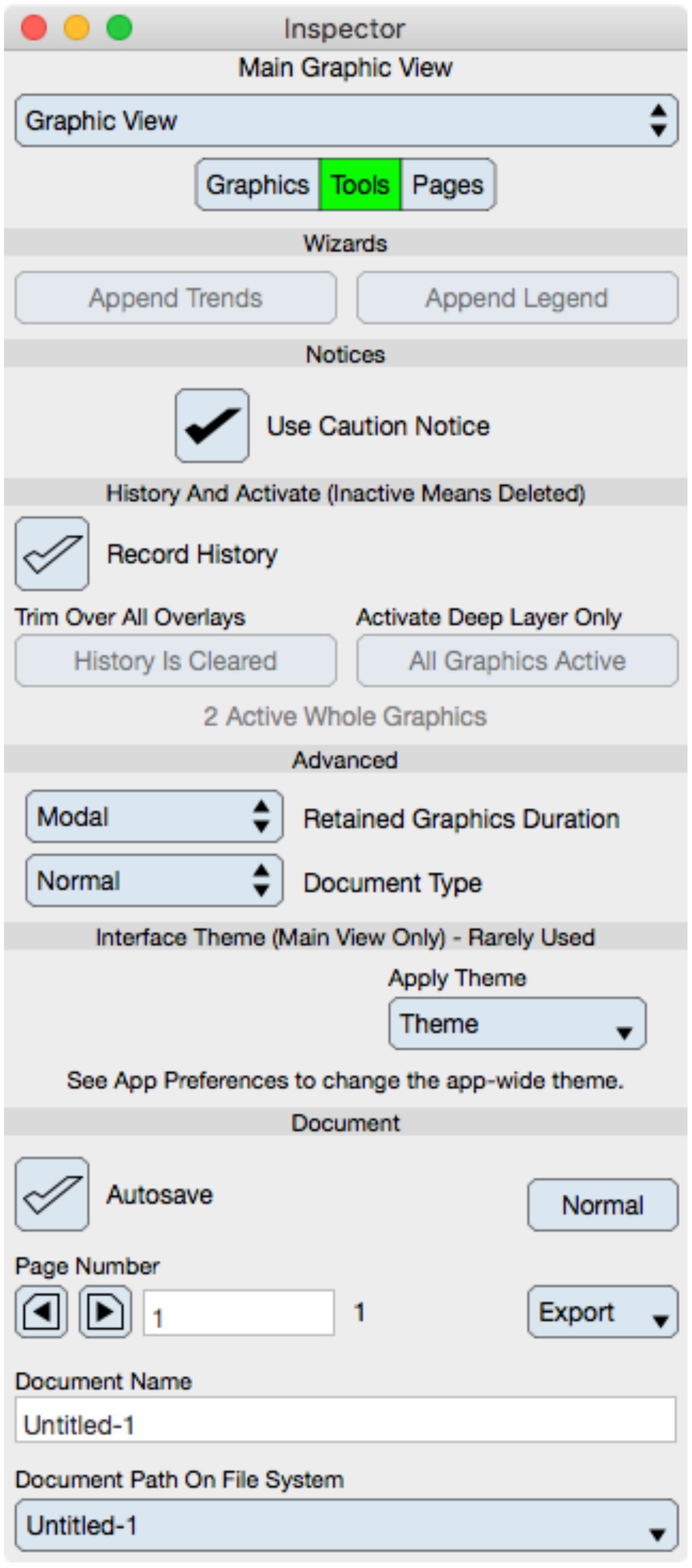
Grid Color : Specifies the grid Color.

Notice that the grid and background are graphics in page zero. They can be selected and altered directly, beyond the minimal attributes described above, by first choosing page zero in the Tools editor described below.

Tools

The Tools editor is used to access controls that affect the entire graphic view. They do not necessarily relate specifically to the graphic view but are dependent upon the graphic view.

Advanced Tip: Some tools such as wizards, clearing history, et. al. only make sense in a contextual focus (focusing on a particular graphic or layer) in which case the Graphic View editor is not visible. Use the Navigator to reestablish the Graphic View editor while not changing the focus on the Graphic View.


Inspector
Main Graphic View
Graphic View
Graphics | Tools | Pages
Wizards
Append Trends | Append Legend
Notices
✓ Use Caution Notice
History And Activate (Inactive Means Deleted)
✓ Record History
Trim Over All Overlays: History Is Cleared
Activate Deep Layer Only: All Graphics Active
2 Active Whole Graphics
Advanced
Retained Graphics Duration: Modal
Document Type: Normal
Interface Theme (Main View Only) - Rarely Used
Apply Theme: Theme
See App Preferences to change the app-wide theme.
Document
✓ Autosave
Normal
Page Number: 1 1
Export
Document Name: Untitled-1
Document Path On File System: Untitled-1

Wizards

Append Trends : Selecting this will append Trends to the graphic view. Use the tool menu in the Status Bar for a more contextual use.

Append Legend : Selecting this will append a Legend to the graphic view. Use the tool menu in the Status Bar for a more contextual use.

Notices

Use Caution Notices : A destructive action such as deleting graphics causes a query to ask if that is what you want. Turn this off if you do not want the query.

History

Record History : When on a History of user interaction is recorded. This is required for undo processing.

Auto Save : When on anything that dirties the document causes the document to save.

Clear All History : Select this button to clear the history of the currently focused graphic. If the focus is on a Layers then the history for all elements of the layer are cleared. This operation is irreversible (unless the document is reverted before it is saved).

Activate Graphics : Deleting graphics causes them to become deactivated and they are still retained (even if history is off). Those graphics can be recovered by selecting this button.

Advanced

Retained Graphics Duration : One of None, Modal or Always. This is an optimization setting and should probably be set to Modal. Set to None if you are using magnification. Set to Always to speed up the graphics display. Retained graphics uses concurrent processing so can be substantially faster for higher core computers.

Document Type : One of Normal, Template or Flat Template. Template is mainly for programming and for making a skin. The default and recommended setting for a user-oriented document is Normal. Normally a document stores each page in a separate directory so that each page can be retrieved efficiently on demand. A Flat document stores all pages into one archive file in the document directory. Flat is more efficient and good for simple template documents.

Interface Theme

Apply Theme : Select an option from the pull down button to change the theming of the graphic view. The theme is only applied to the current graphics and if additional graphics are created then the theme will need to be reapplied. Note that this is different then the app-wide control theme setting in the Application Preferences.

Document

Autosave : When selected then the document is saved upon each change of its content.

Palette Mode : One of Normal or Palette. While in Palette mode, graphics can be dragged from the Graphic View to a view in a different document or application.

Page Number : Pages are part of the Document. One page is shown in the Graphic View at a time, with the exception of page zero, which is a background page. The background page is always displayed and is where you can place footer or header information, for example, for multi-page documents. Use the page controls to select and display a specific page.

Export : Used to export the document contents to another format, mainly SVG.

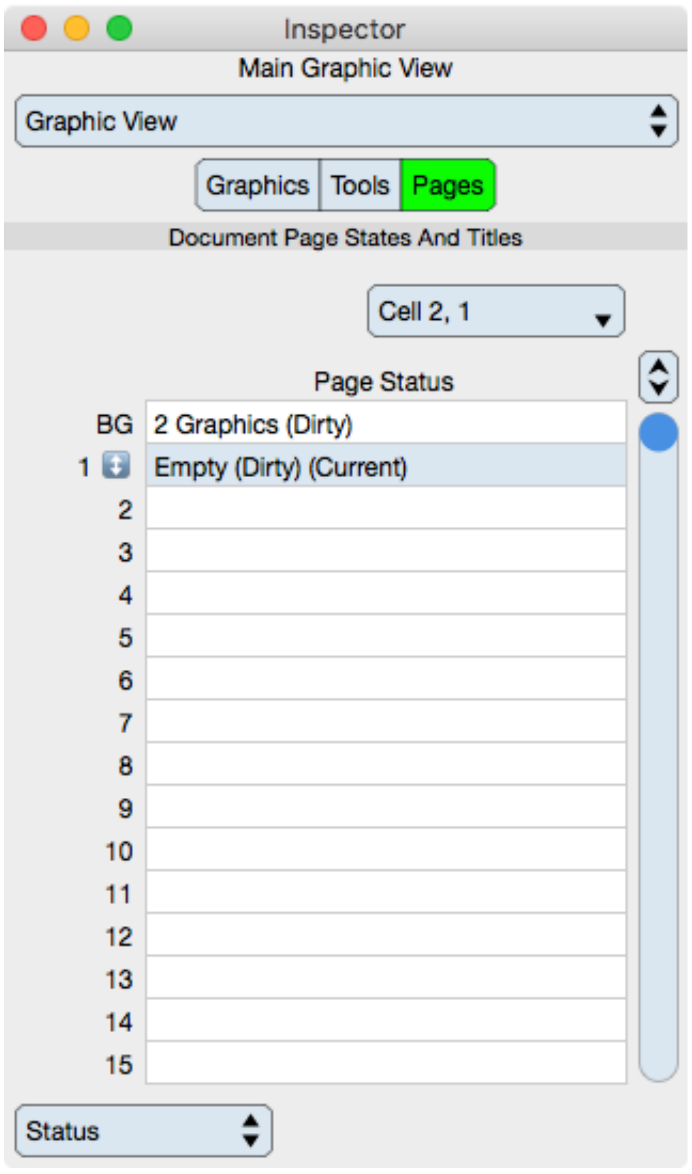
Document Name : This is the name of the document that appears at the top of the Window (for windowed systems such as the Mac and Windows). The name is separate from the document file name on the file system. The window title is comprised of the document name and file name. In a cloud situation the name makes more sense because the document does not have a (local) path in the cloud.

Document Path On File System : Shows the path of the document on the file system. Select this pop up button to see the path components. On the Mac, selecting a path component will present the Finder focused on that path element.

Pages

The Pages editor is used to navigate and make pages in the document.

Tip: Use the Status Bar to scrub all pages where scrub means to quickly move from one page to another using the number selector slider.



	Page Status
BG	2 Graphics (Dirty)
1	Empty (Dirty) (Current)
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

Table

Table : Shows the page information based upon the Table Mode. The first row is the Background Page. Click a cell to make the corresponding page if needed and to make that page current. The row shows the number of graphics, whether the page is dirty and how many inactive graphics are on the page.

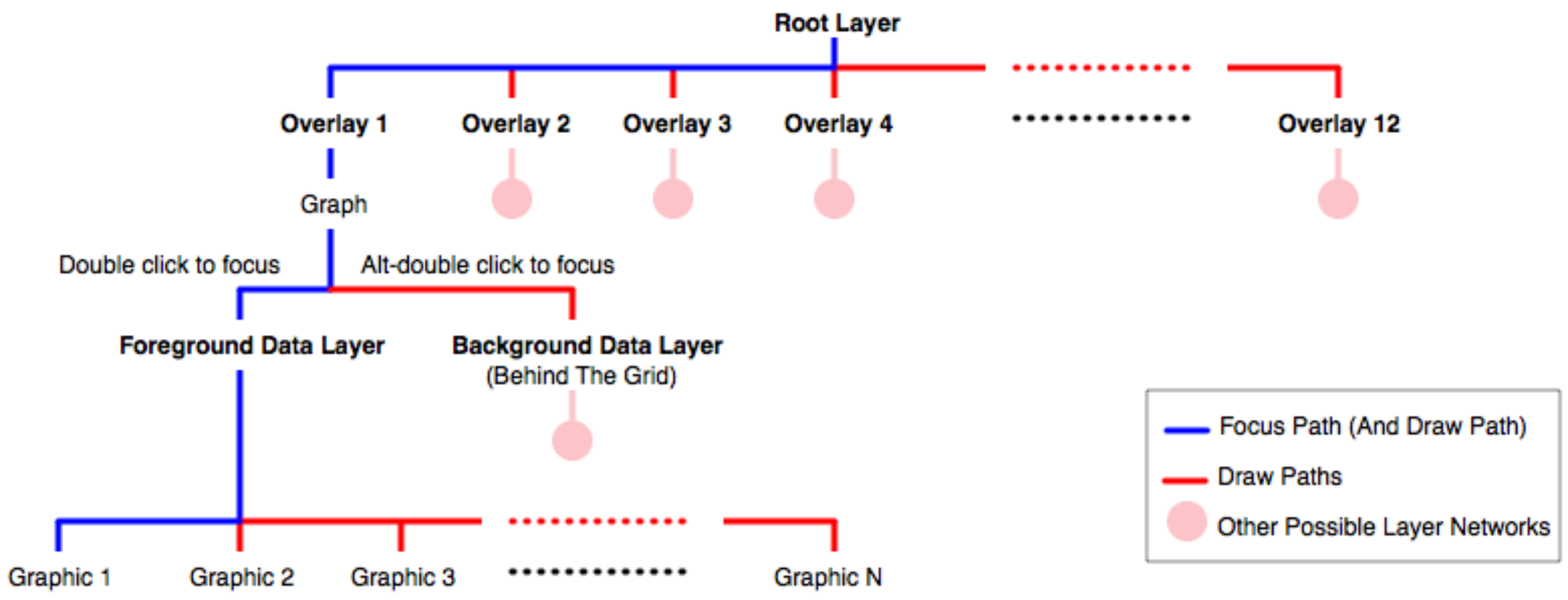
Table Mode : Either Status or Titles. While in Status mode the table can be used to show each pages status and also to select the current page. While in Titles mode, each cell of the table shows the title of the page and those titles can be edited using the operations as define in Tables.

Tip: Use the up and down arrows on the keyboard to swap pages.

Graph IDE ▶ Overview ▶ Layer

Graphics are organized by layers. The Graphic View has layers called Overlays. Graphs have four layers to hold data graphics and some graphical elements of a graph. Layers can also be encountered in other areas, which are left unspecified (implicit) at this time.

When you add a graphic to a Graphic View, in reality you are actually adding it to the focused layer which is contained in the Graphic View. The Graphic View maintains a tree structure network of such layers and Graph IDE's user interface helps navigate that network. The figure below shows a typical layer network where there is one graph on the focused layer path.



The section Getting Data On A Graph demonstrates how to navigate layers using mouse clicks. You can select and edit graphics in a single layer using Standard Editing controls. When graphics are selected in a layer using drag-select then the Layer inspector editor is loaded.

Layers can also be navigated using the Navigator which is a flattened and serialized representation of the tree structure of layers and embedded leaf nodes.

Graph IDE ► Overview ► Pages

A Document shows a Layer tree structure. More precisely, it shows the layers of page zero first and then the layers of the current page. Page zero is reserved as a background page and shows on all other pages. Only one page can be modified at a time.

The background page (page 0) contains the grid and background rectangle which are modifiable on the Graphic View inspector editor. In addition, if you select page 0 via the Status Bar then the graphics of page zero can be altered directly and added to. That way you can add any arbitrary graphic to the background of any other page. Typical uses may include header and footer labels.

Since the background page is directly modifiable then you can also change most of the background and grid parameters directly. A good use would be to change the fonts of the grid axes. Notice that the grid and background graphic are locked and can not be unlocked so that they may not be deleted or move. Use the Graphic View inspector editor to turn the grid (or background) off and to synchronize those graphic attributes with the document parameters. Typically, the grid and background graphic are not to be edited directly.

Notice that the Navigator only shows the current page layers even though the Graphic View shows both page 0 and current page layers. That is because the background page is not modifiable while another page is the current page.

Graph IDE ▶ Overview ▶ Account

An Account is a way to save the contents of a Document to a server. There are several aspects to using an account as described by the following:

- While using an account, document contents are stored on your device (Mac, iPad, iPhone or Windows). You do not necessarily have to use a server to store document contents. The document contents are stored in the application sandbox and inserted into the current document interface so working with accounts is analogous to a single window interface.
- While Logged In then document contents can be stored on, and retrieved from, a server. Saving to a server is implicit, but opening from a server is an explicit operation. Use of a server is optional.
- The server is either at cloud1.vvidget.org or is a computer running the Server within Graph IDE. The section below describes how to setup the server and establish accounts on the server computer.
- The Web Edition of Graph IDE operates through a browser and as such, due to security limitations, there is no local store on your device and all documents are directly from and to the cloud server.
- The local and cloud interaction is designed to synchronize a document and keep it synchronized to a shared server resource. That synchronization is dependent upon a network connection. When the network connection is disabled then the device if offline and only the local store is available for use. When the network connection is reestablished then the document can be synchronized to the shared server.
- The account tables shows document file names and not the Document Name as set in the Graphic View Tools tabloid.

Using an account, and the server, is a way to share documents among devices. By setting up the server (see below) on your local network documents and accounts can be very secure and private, accessed without an Internet connection, and retrieved and stored via a very fast network connection. In addition, documents on the server computer can be altered using the powerful Graph IDE application, browsed and searched using the powerful QuickLook and Spotlight plugin and archived and transported using familiar tools.

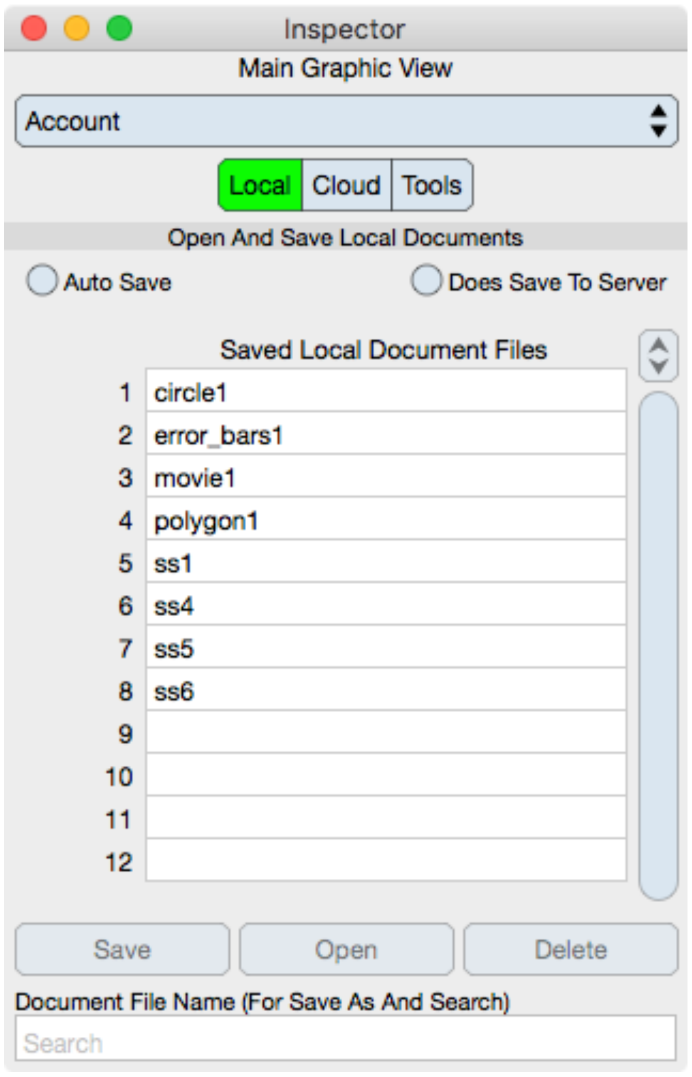
Reminder: When saving to a server please note these rules:

- You must Login to the account before using the server.
- You must select the Does Save To Server button as described below.

Using the Account consists of setting local parameters and cloud parameters as described below.

Local

Use the local settings to save and retrieve local documents (local documents are not available through a web browser).


Inspector
Main Graphic View
Account
Local
Cloud
Tools
Open And Save Local Documents
Auto Save
Does Save To Server
Saved Local Document Files
Save
Open
Delete
Document File Name (For Save As And Search)
Search

1	circle1
2	error_bars1
3	movie1
4	polygon1
5	ss1
6	ss4
7	ss5
8	ss6
9	
10	
11	
12	

Auto Save : When on, document contents are automatically saved. If Does Save To Server is also on then the document is also autosaved to the server. The autosave happens when the document is dirtied (edited) so saving is frequent and for this reason you may not want to autosave and save to the server at the same time because of network latency.

Does Save To Server : When on, a document save also saves its contents to the server. When off then a document save is only stored locally.

Saved Local Document Files : Shows the documents that are stored locally. Select a row entry to insert that document name into the Document File Name text field. Only the name is inserted and the document is not opened until you select the Open button.

Document File Name : Type a document name into the document name field, or select a document name from the table above it in order to define the name of the document. Once the name is defined then the document can be saved and opened. Typing a name will also limit document names in the table to those that match the textual pattern that is typed.

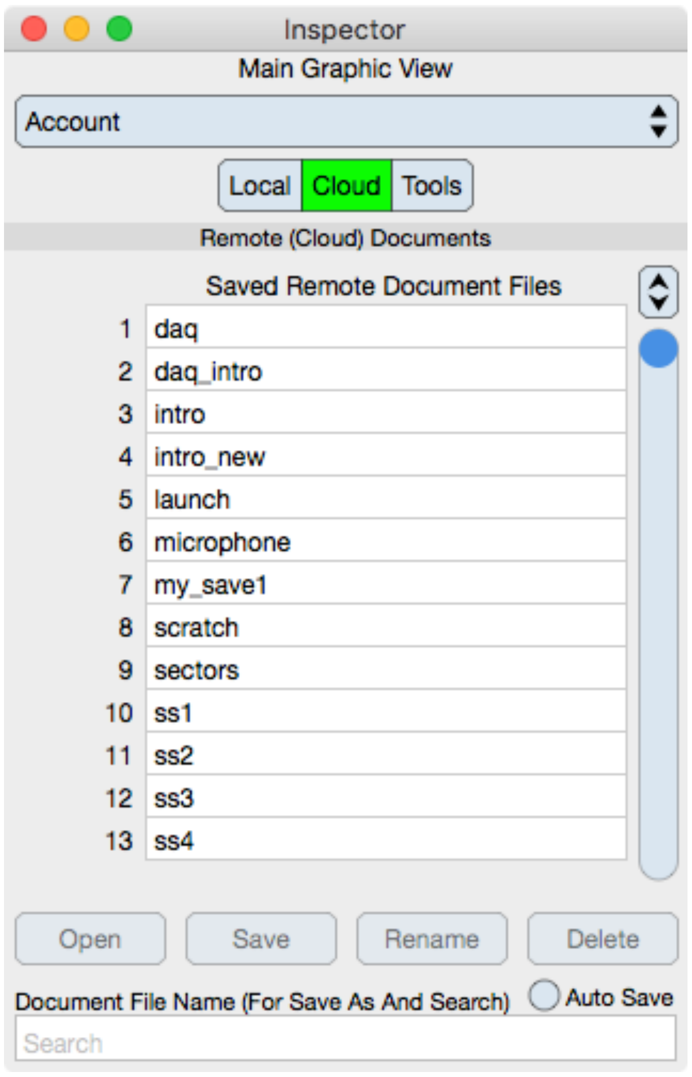
Save : Select the save button to save the document contents. If the Does Save To Server option is on then the document is also saved to the server (see the Cloud settings below to set the document's server values).

Open : Select the open button to open the document from the local store.

Delete : Select the delete button to delete the document from the local store. If the document is also on the server then that document is retained on the server.

Cloud

The Cloud editor is used to work with documents stored on a cloud account.


Inspector
Main Graphic View
Account

Remote (Cloud) Documents
Saved Remote Document Files

1	daq
2	daq_intro
3	intro
4	intro_new
5	launch
6	microphone
7	my_save1
8	scratch
9	sectors
10	ss1
11	ss2
12	ss3
13	ss4

Document File Name (For Save As And Search)
Auto Save
Search

Remote Documents

Saved Remote Document Files : Shows the document names on the cloud service. To see these names you must login using the Login editor.

Document File Name : The file name of the document on the server. Once the name is set then select the Open button. Typically you should select the name from the Saved Remote Document Files table because that will enter it into this field. Typing a name will also limit document names in the table to those that match the textual pattern that is typed.

Open : Select the Open button to open the document from the server. For native device editions, this will also save the document from the server to the local file system so that the document can then be used offline.

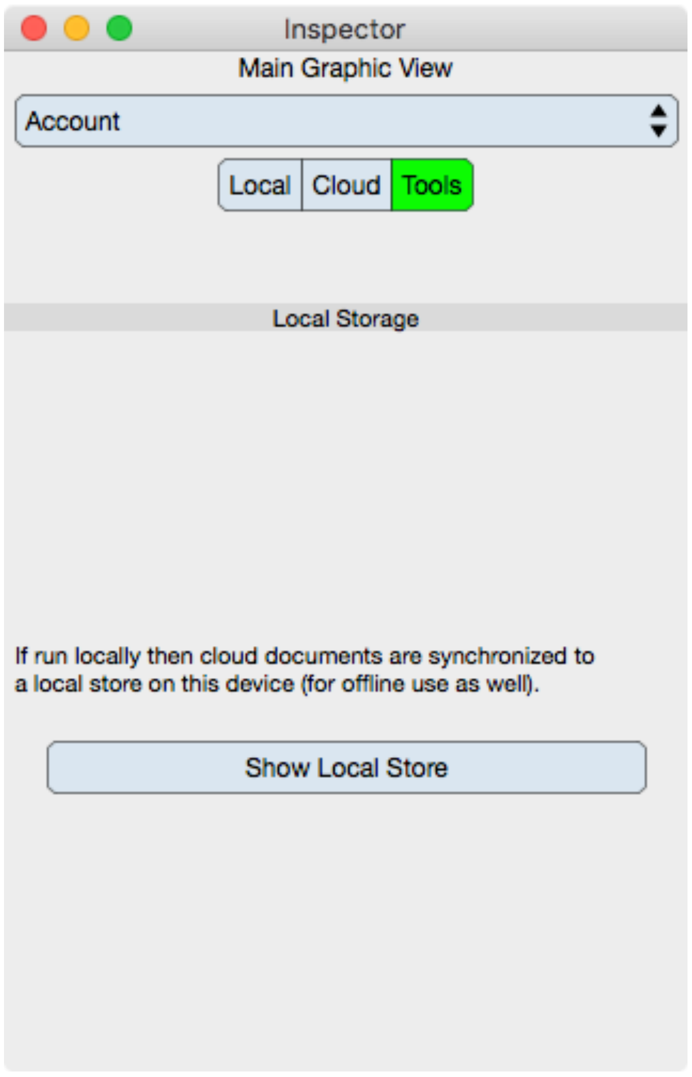
Save : Select the Save button to save the document (transmit its archive) to the server.

Rename : Select the Rename button to rename the document on the server. To do this, first select an entry in the table, select Rename and then enter the new file name in the resulting Data Selector.

Delete : Select the Delete button to delete the document on the server. This operation is irreversible, however if you previously opened the document then the document is the current document and can be saved again. Note that this pertains to the document selected in the table.

Tools

The Tools editor adjust features of the account backstore.


Inspector
Main Graphic View
Account

Local Storage

If run locally then cloud documents are synchronized to a local store on this device (for offline use as well).

Local Storage

Show Local Store : (Available on Mac) Shows the local store for the document by launching the Finder on a Mac. You can alter the document in the Finder, make new documents and then save them directly to the local store without using this Account feature. If you then go to the Local tab then the new documents will appear in the Saved Local Document Files. In this way, you can bypass the Account interface and then at a future time use the Account interface to synchronize with the cloud.

Server

The account interface can be used without a server. However, if you choose to use a server then this section defines how to setup and maintain a server (not available with Graph IDE CE). Note: You may typically use the cloud service at http://www.vvidget.org/cloud, however you may wish to setup your own cloud service for performance or security issues.

Choose a computer for the server host. To establish an account on the www.vvidget.org computer please email support@vvi.com. If you choose to use the www.vvidget.org host then the following is not needed.

To setup your own server host, first install Graph IDE on it, launch Graph IDE and then turn on the Server within Graph IDE by using the Preferences settings and select the Cloud Accounts option. The host is now an account server.

Cloud accounts can be created and maintained using the Administrator inspector editor. However, the first account which should be an administrator account needs to be created at the command line. That is because creating accounts via the Administrator inspector editor requires administrator privilege. Maintaining accounts is via a command line interface. The following is an example of that use and assumes the currently logged in user is the one that launches the Graph IDE application. The following commands primes the account called joe with password samantha and establishes that account as administrator.

For the Mac App Store Edition (sandboxed version):

```
cd ~/Library/Containers/com.vvi.Graph-IDE-Mac/Data/Library
mkdir -p Vvidget/Server
cd Vvidget/Server
mkdir joe
cd joe
echo -n "samantha" > password.txt
```

For the Manufacturer Edition (non-sandboxed version):

```
cd ~/Library
mkdir -p Vvidget/Server
cd Vvidget/Server
mkdir joe
cd joe
echo -n "samantha" > password.txt
```

To set that account as administrator put a file named `account_features_initial.vvidefinition` in the account path with the following contents:

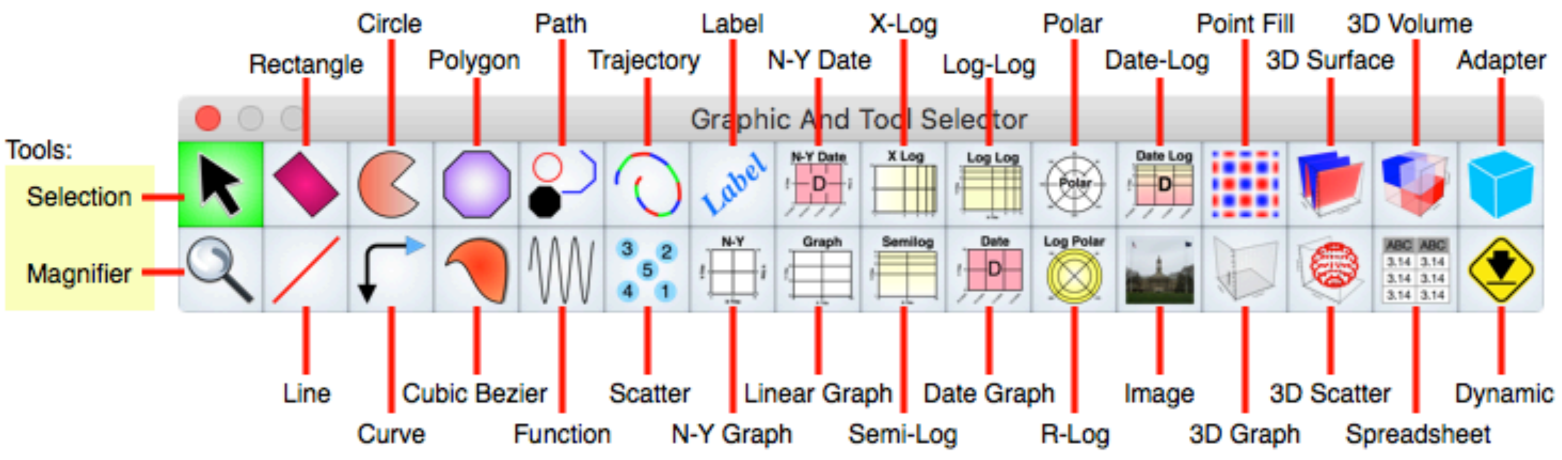
```
{
VVKEt = YES;
VVKEu = 2;
VVKEv = 0;
VVKF3 = 1;
VVKF4 = 200;
}
```

After doing so then that account as well as all other accounts can be maintained and created using the Administrator editor.

The account host should be statically routed, have a DNS entry and a valid IPv6 or/and IPv4 address. However it can also simply be a computer on your local network. If it is on a local network via a DHCP server (a usual configuration) then make sure the host is assigned a static IPv6 address (poke a hole in DHCP). In addition, pass TCP/IP port 9877 through any firewall if needed. If DNS is not running locally then the host name may possibly be looked up using a host.local syntax.

When clients save documents in their account then those documents can be altered and viewed in the usual ways on the server by browsing the account folders from the Finder. It is particularly handy to maintain documents on a Mac and then work with them on an iPad as needed.

The Graphic Selector determines which type of graphic you will make, or whether you want to select an existing graphic or not. The figure below show a typical Graphic Selector and annotates each cell of the Graphic Selector. This is just one form of Graphic Selector interface. The exact cell positioning and types can vary depending on what you have loaded into Graph IDE.



To select a mode of operation click on one of the cells in the Graphic Selector. Each cell is defined below.

Tool Cell Definitions

Click one of these cells to place the mouse operations in select or magnify mode.

Selection	Click to set in selection mode. See Standard Editing for information on control behavior during Selection Mode. Unless you are going to create a graphic, this Selection mode should always be selected.
Magnifier	Click to set in magnification mode. While in this mode a mouse drag on a Graphic View or Graph defines the area to magnify to. This also brings forward the Magnifier inspector editor.

Factory Cell Definitions

Click one of these cells to select the graphic type you will make next. After selecting move the cursor to a Graphic View and then mouse down and drag the mouse. Then release the mouse button. A graphic of the type selected will have been made. You can then proceed to alter its attributes. If you use a graphic with specific attribute values enough then you may want to put it on a Palettes for drag and drop creation. The graphic can also be instantiated from its Prototype inspector editor. That editor is loaded when a factory cell is selected.

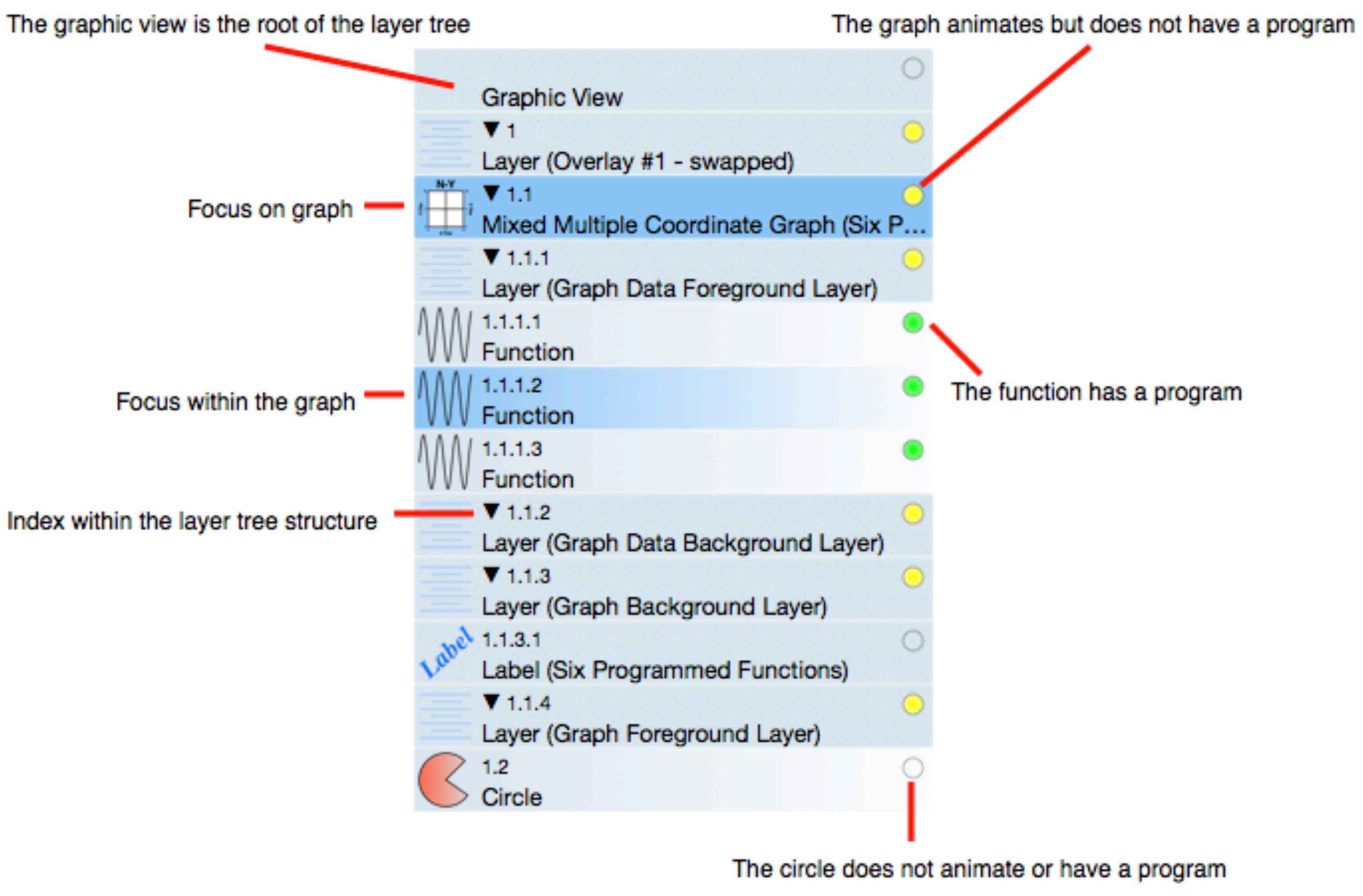
After you make a graphic the Graphic Selector will reset to the selection tool. If you alt-click a factory cell to then create a graphic then the Graphic Selector will be in sticky mode and does not reset to the selection tool after graphic creation. While in sticky mode the selected cell is blue, otherwise the selected cell is green.

Adapter	Specifies an adapter, which is a hook into the system views.
Circle	Specifies a circle, ellipse, or pie section.
Cubic Bezier	Specifies a cubic bezier graphic, which is a connected sequence of cubic bezier sections. After you create this graphic you will want to point-edit it to further create it.
Curve	Specifies a curved section, which is a line, elbow or curved elbow with caps. After you create this graphic you will want to add your own caps using the Caps sub-editor on the inspector.
Date N-Y Graph	Specifies a date x-axis and two linear y-axis, one on each side of the graph.
Date Graph	Specifies a date axis with one y-axis on the left side of the graph.
Date-Log Graph	Specifies a date axis with one log-y-axis on the left side of the graph.
Dynamic	Specifies a dynamic graphic which is actually a factory reference to load another type of, yet unknown, graphic.
Function	Specifies a function graphic. Use a function graphic to define a sequence of points running from the left to right. This is the prototypical curve on a graph used for plotting purposes.
Image	Specifies an image, which is any type of specification normally interpreted by the environment outside of Graph IDE. The image an be a PNG, TIFF or PDF representation for instance.
Label	Specifies a label, which is a single line of text encoded in UNICODE standard.
Line	Specifies a line segment.

N-Y Graph	Specifies a single x-axis and multiple y-axis graph. This is called a multiple coordinate rectangular graph and the coordinates can be of many mixed types.
Linear Graph	Specifies a graph. This is the graph one normally thinks about which is one linear x-axis and one linear y-axis. This is also called a rectilinear graph or Cartesian graph.
Log-Log	Specifies a graph with log axes in both directions.
Path	Specifies a path, which is a sequence of operations, some of which have operands.
Point Map	Specifies a point map graphic. Use a point map graphic to define a matrix of z-values where the z-values are represented by color.
Polar	Specifies a polar graph, also known as a radar graph.
Polygon	Specifies a polygon which is a sequence of points connected by line segments. When the line segments do not intersect midway then it is a polygon, otherwise it promotes to another form.
Rectangle	Specifies a rectangle, oval or parallelogram.
R-Log	Specifies a polar graph with a log scale in the r-direction. This is also known as a radar graph.
Scatter	Specifies a scatter graphic.
Semi-Log	Specifies a semi-log graph which is a graph with x-linear axis and y-log axis.
Spreadsheet	Specifies a spreadsheet graphic which is also a Table and can make a Pie Chart and Bar And Column Chart as well as other graphs.
Trajectory	Specifies a trajectory graphic.
X-Log	Specifies a X-oriented semi-log graph which is a graph with log x-axis and linear y-axis. Note that the word semi-log graph is usually reserved for a graph that has a log y-axis and linear x-axis.
3D Graph	The graph on which 3D data graphics are made. This is also called a 3D perspective graph or a 3D rectilinear graph which is projected onto a view plane.
3D Scatter	A 3D Scatter is a sequence of 3D points.
3D Point Map	A 3D Point Map is a regular grid of z-values. The z-values are represented as the z-coordinate value thus producing the three dimensional effect. The z-values are mapped (several times) to a projected representation and that typically involves a radiance function.
3D Volume	A Volume is a regular grid of densities (values ranging from 0 to 1). The densities are represented as cube sections on a regular grid where the cube has color specified by a color map.

Graph IDE ▶ Overview ▶ Navigator

The navigator shows and helps navigate the Layer structure of a document. An example is diagrammed in the following figure.


The graphic view is the root of the layer tree
The graph animates but does not have a program
Focus on graph
Focus within the graph
The function has a program
Index within the layer tree structure
The circle does not animate or have a program
Graphic View
▼ 1
Layer (Overlay #1 - swapped)
▼ 1.1
Mixed Multiple Coordinate Graph (Six P...
▼ 1.1.1
Layer (Graph Data Foreground Layer)
1.1.1.1
Function
1.1.1.2
Function
1.1.1.3
Function
▼ 1.1.2
Layer (Graph Data Background Layer)
▼ 1.1.3
Layer (Graph Background Layer)
1.1.3.1
Label (Six Programmed Functions)
▼ 1.1.4
Layer (Graph Foreground Layer)
1.2
Circle

Formally speaking, the navigator traverses the layer tree structure of a document contents and maps the tree nodes into a sequential (linear) format. The tree starts at the Root Layer, descends through branches and ends up at leaf nodes. It displays each node as a rectangular region with descriptions of several attributes of the nodes as listed in the following table.

The order of each node within a layer is considered a z-buffer and as such is user specified. That means that the navigator order for nodes in a specific layer may not have a specific order. If you apply a Layer sorting of Natural then the navigator order will correspond to left to right and then top to bottom page view ordering thus scrolling down the navigator is akin to scrolling down the main graphic view.

Attribute	Description
Tree Index	The tree index shows the depth of the node by a period delineator and the sequence index within a layer by the decimal. The number of periods corresponds to the depth within the tree structure.
Focus	When graphics are clicked upon and graphs double-clicked then they are focused and defocused. On the graphic view the focus is indicated by red knobs. On the navigator it is indicated by a blue rectangular region (a cell). Clicking on a cell in the navigator focuses the corresponding graphic in the graphic view. Shift-click the cell to maintain the current graphic selection and also add the corresponding graphic to the selection and focus upon it.
Node Type	A tree structure has two node types, a branch and a leaf. Leafs are terminal (have no subcomponents) while branches have lists of other nodes either other branches or leafs. For example, a Layer and Group are branches while a Circle is a leaf. Graphs are both leafs and branches. Graphs are leafs because they have content on their own right (draw axes and other things) and branches because they contain layers of data. Leafs are indicated by a gradient fill in a rectangle while branches are a solid fill color. In the Navigator, the rectangular region of a node reference is called a cell.
Program	Any node can be Programmed and the program is specified on a node basis. That is unlike other programming models (textual based) that are specified linearly (in a source code file). Because a program is node based it is difficult to know which nodes will execute when a document is animated. To solve this problem, each node in the navigator contains a circle. A green circle means the node contains a program that will be executed, a yellow circle means the node will be executed but does not contain a program and a hollow circle means the node does not animate. For example, a graph can animate its data layers and autoscale without containing source code.
Collapse/Expand	If a node is a branch node then it can be collapsed and expanded by clicking on its tree index. If collapsed, that node will not show its subnodes in the navigator.
Scroll	The navigator is located in a scroll view so it can be panned up and down to locate a node.

Exception Nodes	There is a one-to-one mapping between the Layer structure and the navigator cells, except for the following two exceptions: (1) the Root Layer is substituted by a Graphic View reference. That is because the Root Layer is inaccessible to the user and serves only to start the tree structure while the Graphic View is editable via its inspector and (2) some nodes, specifically the background grid and rectangle, are programmed as non-editable and those nodes do not appear in the navigator.

In Graph IDE nodes (a.k.a.: graphics) are constructed using the many techniques of mouse clicks, palettes, programming, wizards (chart tasks) and other methodologies. The graphics are focused, defocused, edited directly and indirectly. The navigator adds another tool to facilitate the visualization and traversal of the network (tree structure) of nodes displayed in the graphic view of the document. The navigator should be used in tandem with other methods of navigation and visualization. For example, when a graphic is clicked upon in the graphic view its node in the navigator is scrolled to. There is no need to manually (directly) scroll the navigator. When a Group graphic is clicked upon it is focused and its components can be edited in the inspector as a group but components can not be focused and edited individually. The navigator can be used to focus on a component of a group to edit that component directly. Without the navigator, the group is treated as a single entity and the only way to work with components individually is to ungroup the group, work with the component and then regroup the components.

The navigator also gives feedback regarding the focus and traversal through the network of graphics. In that way, the navigator augments the primary way of dealing with graphics and probably should not be used as a primary way to navigate the graphics on a graphic view.

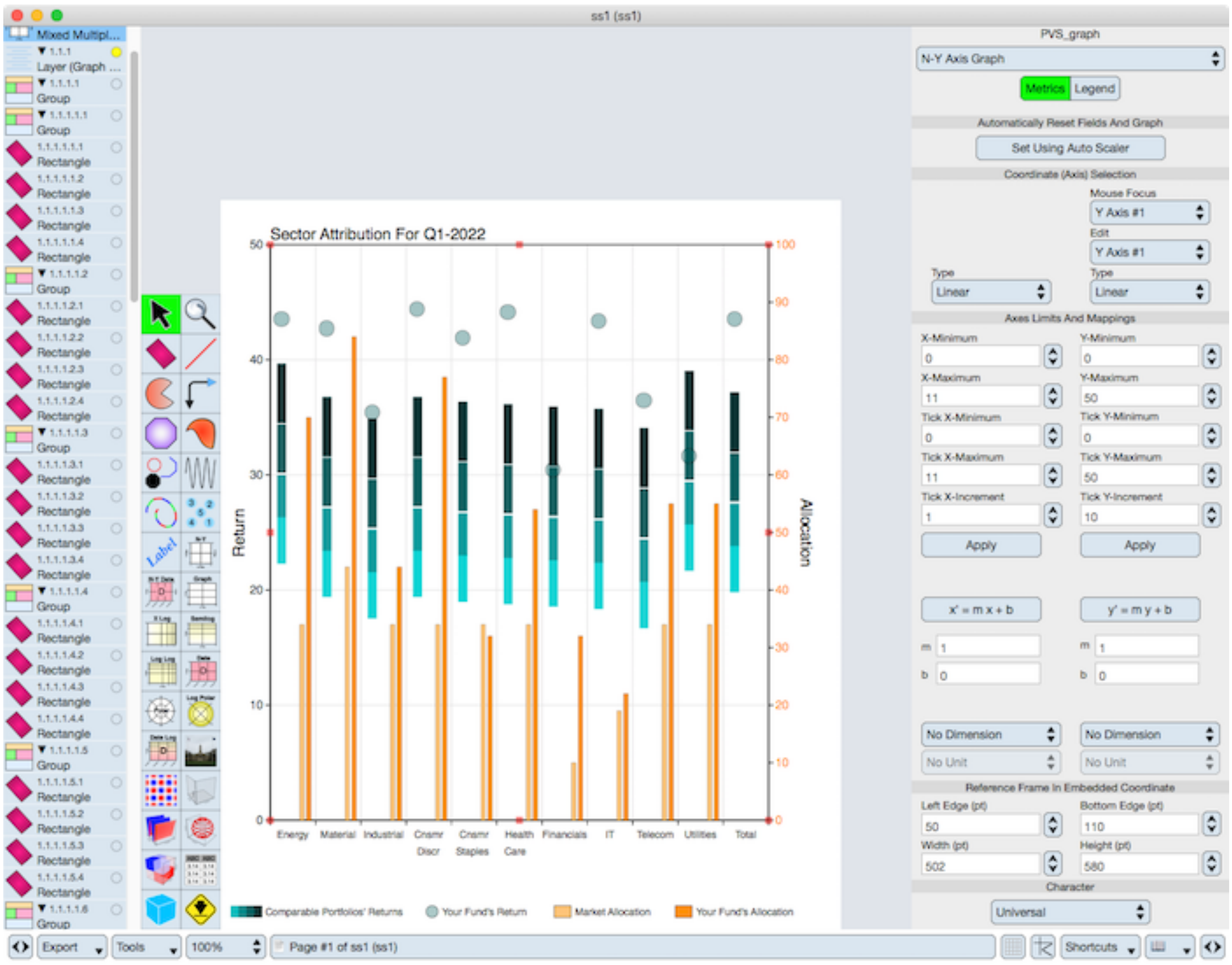
Table Of Contents

Graph IDE ▶ Overview ▶ Document

A document stores all the graphics you make on the Graphic View. It separates out overlays and pages to efficiently store and retrieve pieces of the document state.

The document window is the main user interface to the document. A document window always has a Graphic View and can have no controls or a comprehensive set of controls as shown in the figure below. The graphics you build show in the Graphic View whereas the attributes of those graphics, and settings for using those graphics, show in the controls of the Document Window.

Note: This section is mostly applicable to the Mac version as the iPad, iPhone and Windows version only has the "Scroll View With Everything" document window.



When you first launch Graph IDE, it presents the default document. You can change that default document and then save it to the file `~/Documents/Vvidget/Standard/default.vvibook`. Then select the Graph IDE ▶ Tools ▶ Authorize Folder ▶ Standard Documents... menu item and select the `~/Documents/Vvidget/Standard` (it should already be pre-selected) folder from the Open panel. Once that is done, Graph IDE will use your newly made default.vvibook document as the default document.

A document can have several different appearances which can be altered using the Graph IDE ▶ Tools ▶ Window Type menu as defined in the following table.

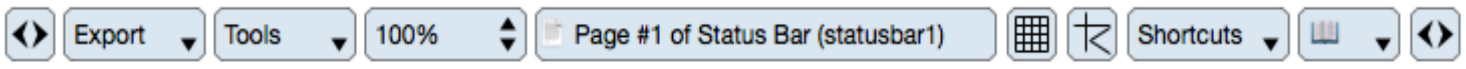
Window Type	Description
Scroll View With Everything	This is the most comprehensive document presentation. It includes a Graphic View in a scrollview and onboard Graphic Selector, Navigator, Inspector Editor, Status Bar and other controls. This is the document type used for single-window platforms such as the iPad, iPhone and Windows desktops.
Scroll View With Tools	Has a few simple controls and a Graphic Selector.
Scroll View	Has a Graphic View in a scroll view and some very simple controls.
Simple Controls	Has a resizable Graphic View and simple controls.
Simple Resize	Has a resizable Graphic View and no controls. This is the window type you want to set when making a palette.
Fixed Size	Has a fixed-size Graphic View and no controls. This is the window type used for palettes. If you do not set the document to this type then it will be set to this type when the document is used as a palette.

Graph IDE ▶ Overview ▶ Status Bar

The Status Bar is an area on a Document window that contains controls that appear in other areas but are so commonly used that they are placed right on the document itself.

When the document is dirtied (needs saved) then the status bar background turns red.

The following figure shows the Status Bar.



The controls shown above are itemized from left to right as follows.

Navigator Stepper : Determines how much of the Navigator is shown.

Export : The export menu gives access to export features such as Print, Export To and Paste Board.

Tools : The tools menu gives access to tools such as the Legend and Trends tool and other tools. Many of these tools are also available on the Graphic View tools editor.

Page : The page control is a Stepper that can be used to navigate and scrub pages in the document. The stepper title shows the page name and document name. The primary page navigation control is located on the Graphic View page editor.

Magnification : The magnification menu sets the magnification of the Graphic View. This setting is also available on the Magnifier editor.

Grid : The grid button sets the background grid on and off. This setting is also available on the Graphic View editor.

Snap : The snap button sets snap to grid on and off. This setting is also available on the Graphic View editor.

Shortcuts : Shortcuts to the Preferences, Login and other editors.

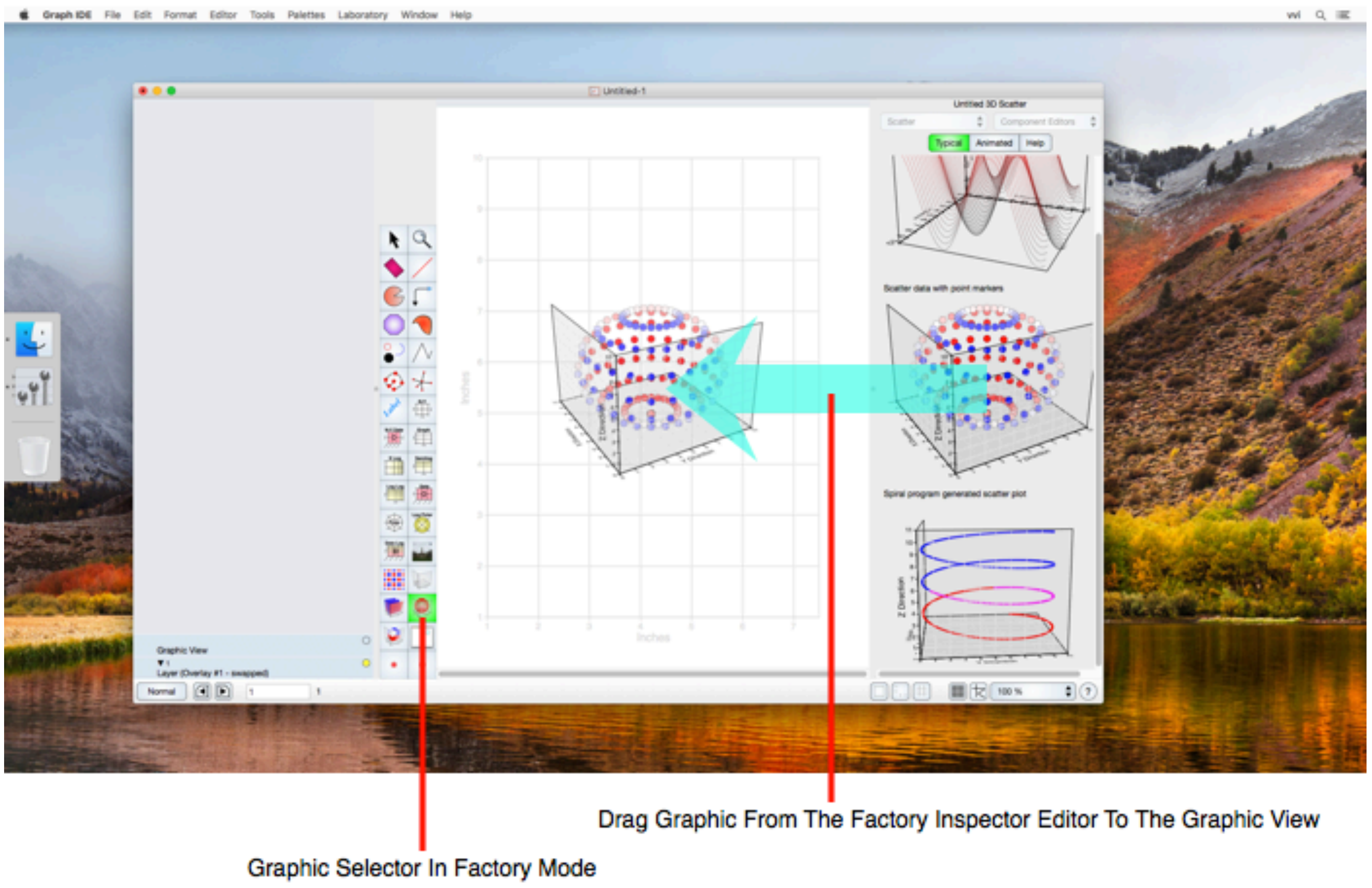
Documents : A menu of all the documents that have been opened.

Inspector Stepper : Determines how much of the Inspector Editor is shown.

When the Status Bar width decreases, the controls in the center will be removed. Those controls will reappear when the width increases to an amount where they can be accommodated.

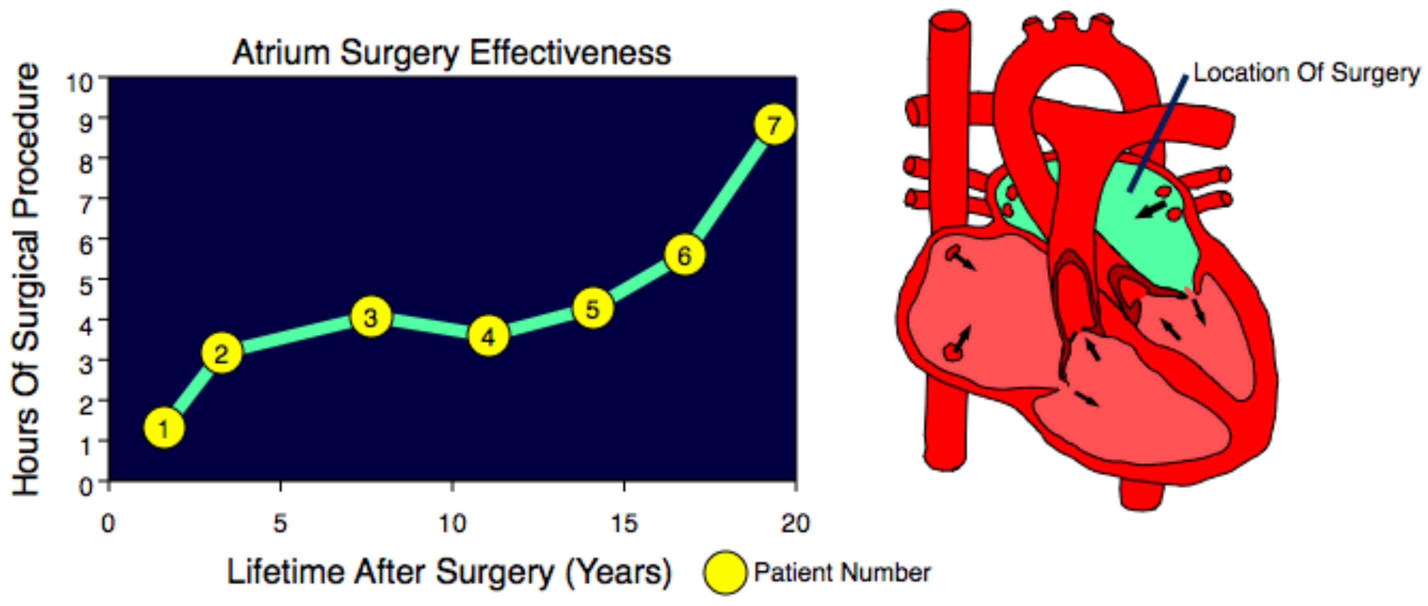
Graph IDE ► Overview ► Palettes

Graphics can be dragged from the factory inspector editor which contains palette documents embedded into Graph IDE. The Graphic Selector must be in factory mode to see the factory inspector editor. See the Prototype inspector editor description for more information. One such palette is diagrammed below.


Drag Graphic From The Factory Inspector Editor To The Graphic View
Graphic Selector In Factory Mode
Untitled 3D Scatter
Typical
Animated
Help
Scatter data with point markers
Spiral program generated scatter plot

Note: If you are making a custom version of Graph IDE then you can change any of the prototype palettes to contain prototypes specific to your use. The prototype palettes are simply Graph IDE documents that can be modified directly by Graph IDE.

Since the graphics on a palette are all live they can also be modified and incorporated into a larger piece of work, such as the graph in the following figure.

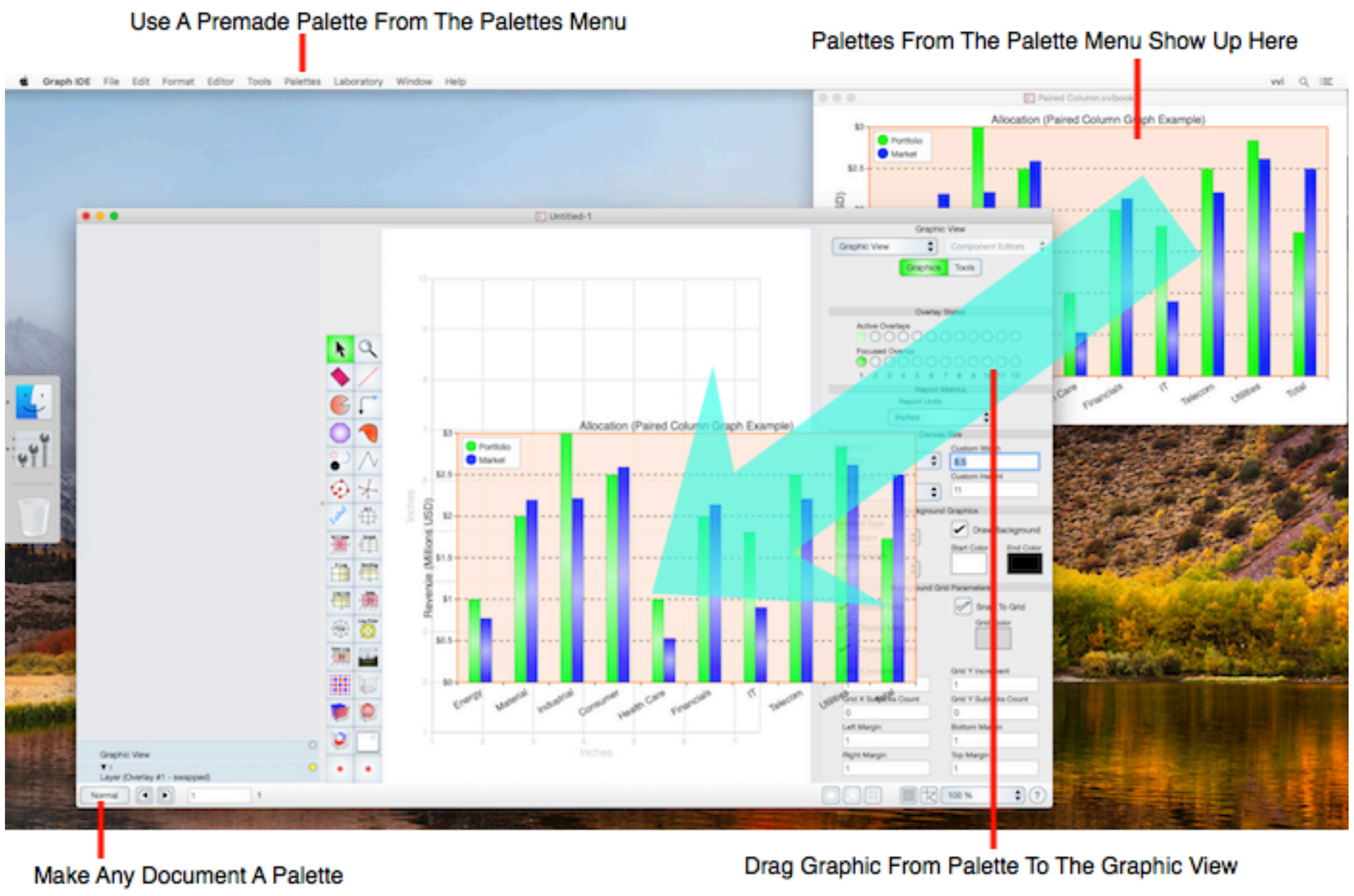


Palettes On The Mac

Palettes are Graph IDE Documents that contain pre-made graphics. They are accessible via Graph IDE's Palettes menu item. Any of the graphics on a palette can be dragged to your own document. You can make your own palette of graphics by adding a Graph IDE Document to the `~/Documents/Vvidget/Palettes` folder in your home folder. The next time Graph IDE is launched it will have a menu item in its Palettes menu which, when clicked, will bring forward the palette you made.

Note: If you are using the Mac App Store Edition of Graph IDE then you must first authorize the access of the `~/Documents/Vvidget/Palettes` folder before your own palettes will load into the Palettes menu. Do that by selecting the Graph IDE ► Tools ► Authorize Folder ► User Palettes... menu item and selecting the `~/Documents/Vvidget/Palettes` (it should already be pre-selected) folder from the Open panel. Once the access is authorized then your own palettes will load in the menu upon next launch of Graph IDE.

The figure below shows how a graphic (a completely built column graph) is dragged from a palette to a document.


Use A Premade Palette From The Palettes Menu
Palettes From The Palette Menu Show Up Here
Make Any Document A Palette
Drag Graphic From Palette To The Graphic View

To make a conforming Graph IDE palette follow these steps:

- Using Graph IDE, make a new document by clicking the File ► New menu item.
- Then click the Tools ► Window Type ► Simple Resize menu item.
- Resize the document to the desired size of the palette window.
- Turn off the background grid.
- Add the graphics you want.
- Save the document to your home folder's `Documents/Vvidget/Palettes` directory. (You may have to make that directory first).
- Quit Graph IDE and then relaunch it.

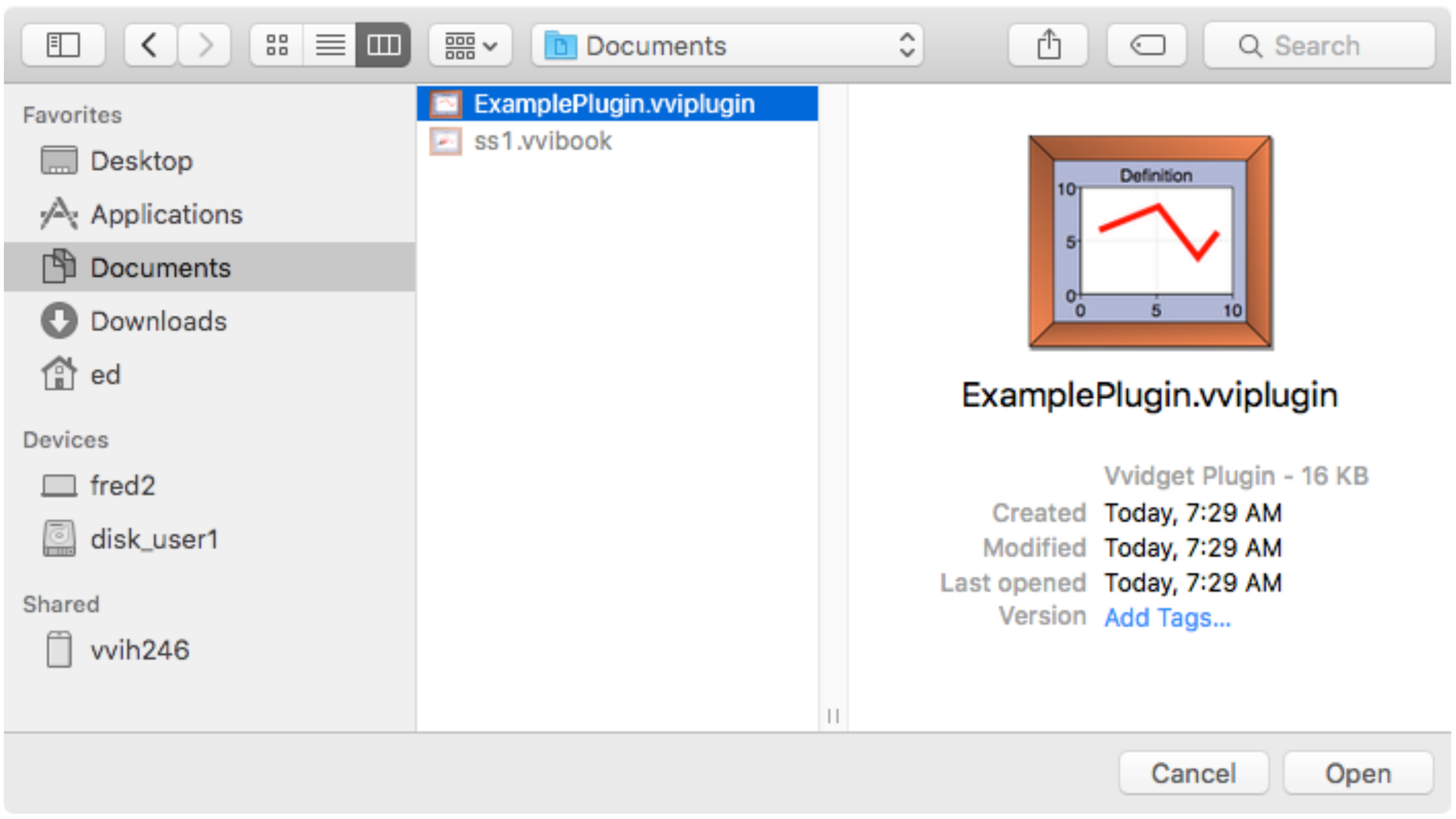
Hint: For complex palette graphics first make all the graphics, then select and group them. The group can be picked from the palette as a single entity. Once the group is on the target document then it can be ungrouped.

Palettes are loaded into the Palettes menu and organized by folder, so you may want to create subfolders in the `~/Documents/Vvidget/Palettes` folder in order to organize your palettes.

Graph IDE ► **Overview** ► **Open Plugin**
(Available on Mac)

Note: This section is only applicable to Graph IDE on the Mac. To make a custom Graph IDE for other platforms see GitHub/VVI.

Selecting the main menu item Tools ► Programming ► Load Plugin... brings forward the open panel as shown below.



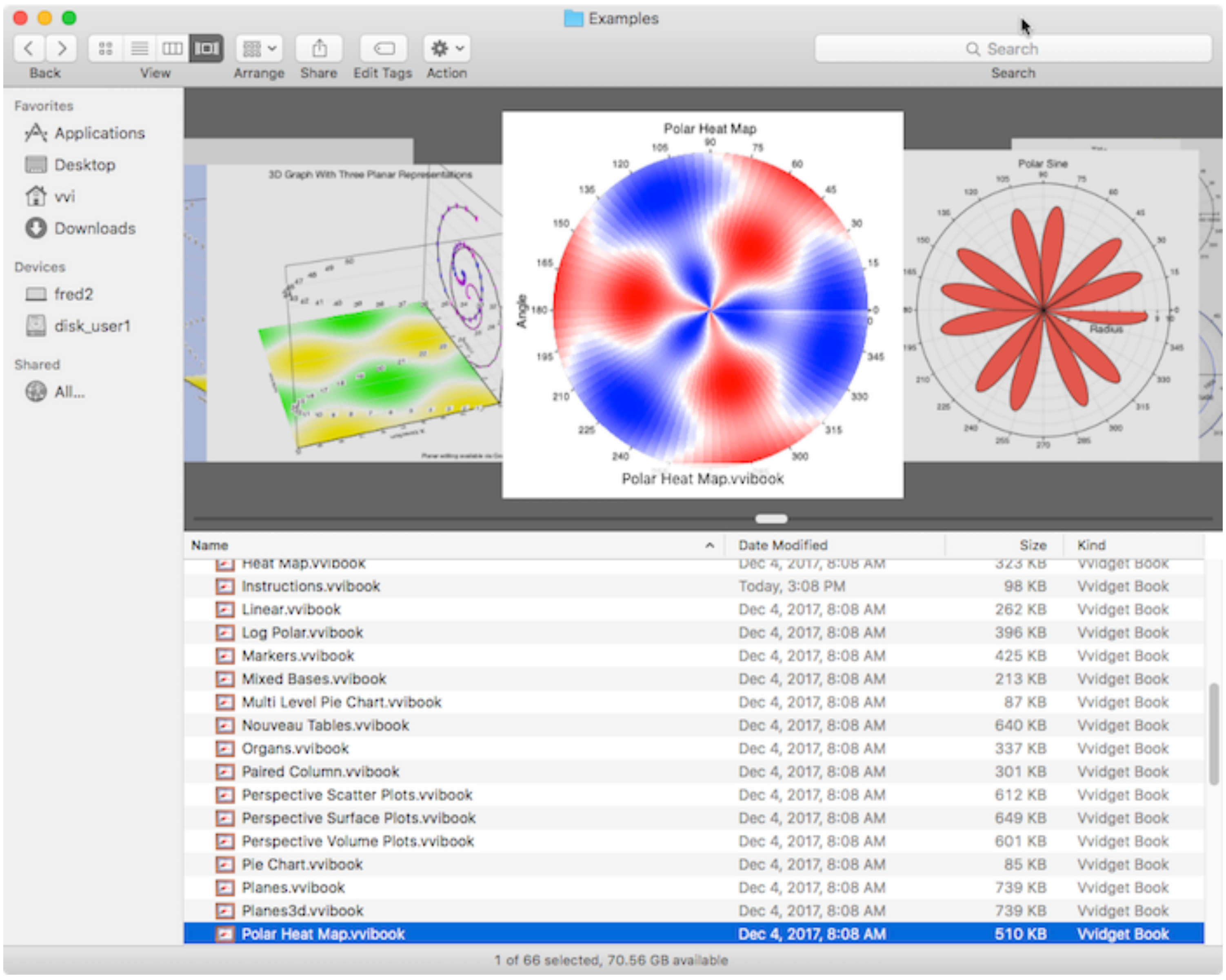
Navigate to a plugin and then click the Open button to load it for use with programming. The sections below describe the plugin functions within Graph IDE.

Component	Description
Plugin	This section describes how to write and compile a plugin bundle.
Programming	This section describes how to program a graphic. The plugin objects and methods are accessed via program method calls.
Program Inspector Editor	This section describes the user interface entrance for a program source code and hence plugin access point.

Graph IDE ▶ **Overview** ▶ **Quick Look & Spotlight**
(Available on Mac)

This is only available on the Mac. If documents are hosted on a Cloud Account which resides on a Mac then your other platform documents can be searched directly on that Mac.

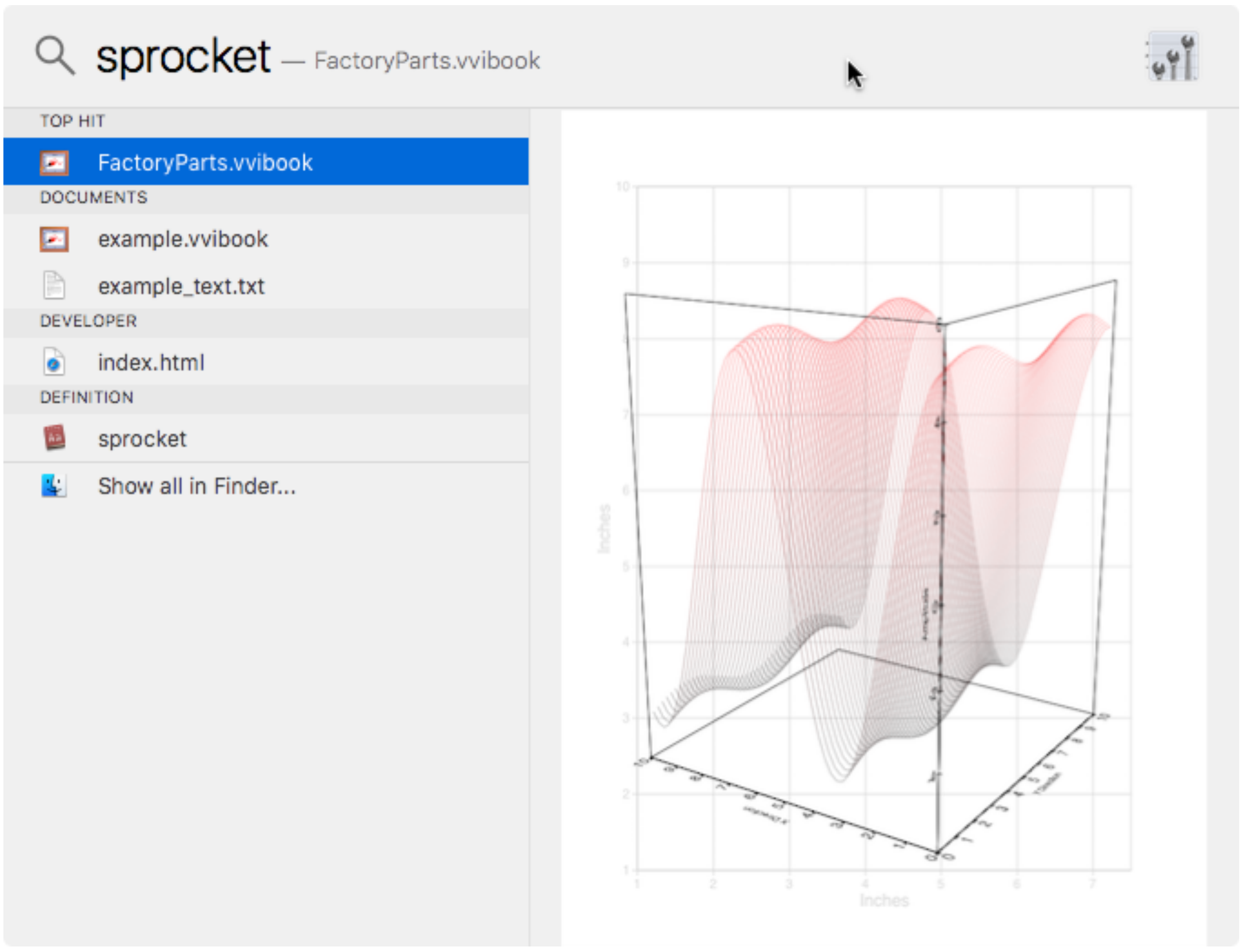
Graph IDE includes Quick Look and Spotlight plugins (service). The Quick Look plugin enables viewing of Graph IDE's documents in the Finder as shown below, in this instance in coverflow representation.



Name	Date Modified	Size	Kind
Heat Map.vvibook	Dec 4, 2017, 8:08 AM	323 KB	Vvidget Book
Instructions.vvibook	Today, 3:08 PM	98 KB	Vvidget Book
Linear.vvibook	Dec 4, 2017, 8:08 AM	262 KB	Vvidget Book
Log Polar.vvibook	Dec 4, 2017, 8:08 AM	396 KB	Vvidget Book
Markers.vvibook	Dec 4, 2017, 8:08 AM	425 KB	Vvidget Book
Mixed Bases.vvibook	Dec 4, 2017, 8:08 AM	213 KB	Vvidget Book
Multi Level Pie Chart.vvibook	Dec 4, 2017, 8:08 AM	87 KB	Vvidget Book
Nouveau Tables.vvibook	Dec 4, 2017, 8:08 AM	640 KB	Vvidget Book
Organs.vvibook	Dec 4, 2017, 8:08 AM	337 KB	Vvidget Book
Paired Column.vvibook	Dec 4, 2017, 8:08 AM	301 KB	Vvidget Book
Perspective Scatter Plots.vvibook	Dec 4, 2017, 8:08 AM	612 KB	Vvidget Book
Perspective Surface Plots.vvibook	Dec 4, 2017, 8:08 AM	649 KB	Vvidget Book
Perspective Volume Plots.vvibook	Dec 4, 2017, 8:08 AM	601 KB	Vvidget Book
Pie Chart.vvibook	Dec 4, 2017, 8:08 AM	85 KB	Vvidget Book
Planes.vvibook	Dec 4, 2017, 8:08 AM	739 KB	Vvidget Book
Planes3d.vvibook	Dec 4, 2017, 8:08 AM	739 KB	Vvidget Book
Polar Heat Map.vvibook	Dec 4, 2017, 8:08 AM	510 KB	Vvidget Book

1 of 66 selected, 70.56 GB available

The Spotlight plugin enables quick searching of metadata in the Graph IDE document, as shown below.



To set the meta data see Metadata Inspector Editor.

Notice how the Spotlight facility also shows a Quick Look view of the document. Quick Look and Spotlight can work in unison and each service can be utilized by any conforming application.

Graph IDE ► Overview ► Server

Note: The server can start on an iPhone, iPad, Mac or Windows host. However, configuring it to vend in the larger network scheme is not covered in this manual. The following gives some information relevant to the Mac version.

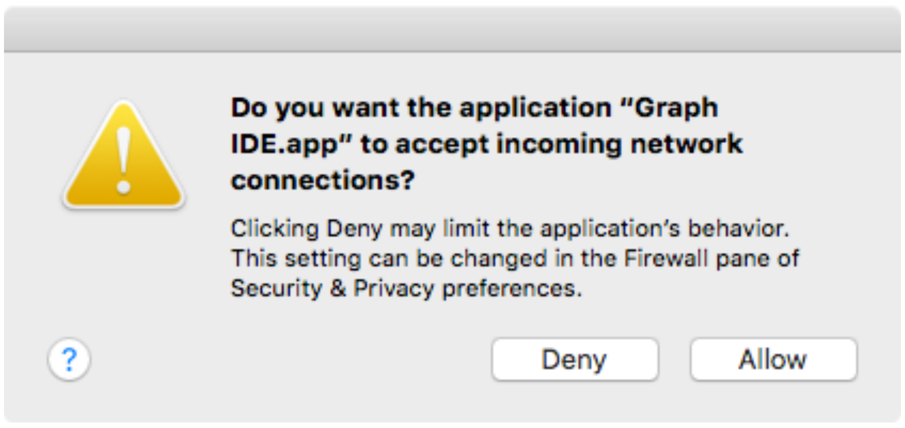
Graph IDE incorporates a server to vend results to client applications such as command line tools and network peers. The server is multithreaded, asynchronous, realtime, programmable and template based. That means you can, among other things, program a Graph IDE document (see the Programming section) and vend the results to anywhere in the world ... if you know how. This section is concerned with adjusting the server settings.

To adjust the server settings see the Preferences section. The server preferences has two options as described here:

- Server State: The server can be turned on or off. It is off by default. Only turn it on if you are going to use a client process. If the server state is altered then relaunch Graph IDE for the changes to take effect. You may want to set Graph IDE to automatically open and hide when you login so that it can vend results upon query. See the System Preferences for the auto open setting.
- Report Type: The server can log errors, summary statistics or no results to the system Console. The default is to not log results.

To learn about the other side of the equation (clients) see the Network Clients section.

Depending upon the security settings of your computer and the operating system version you might receive the following firewall dialog when turning the server to which you can either Deny or Allow.



Do you want the application "Graph IDE.app" to accept incoming network connections?

Clicking Deny may limit the application's behavior. This setting can be changed in the Firewall pane of Security & Privacy preferences.

A line such as the following is logged to the Console in appfirewall.log as well.

```
Sep 24 11:33:35 mac.local socketfilterfw[116] <Info>: Graph IDE is listening from 0.0.0.0:9877 proto=6
```

That is because Graph IDE will listen to TCP/IP port 9877 and other computers may communicate with it. There are various ways to account for this fact, such as firewall settings in a router, that are beyond the scope of this manual.

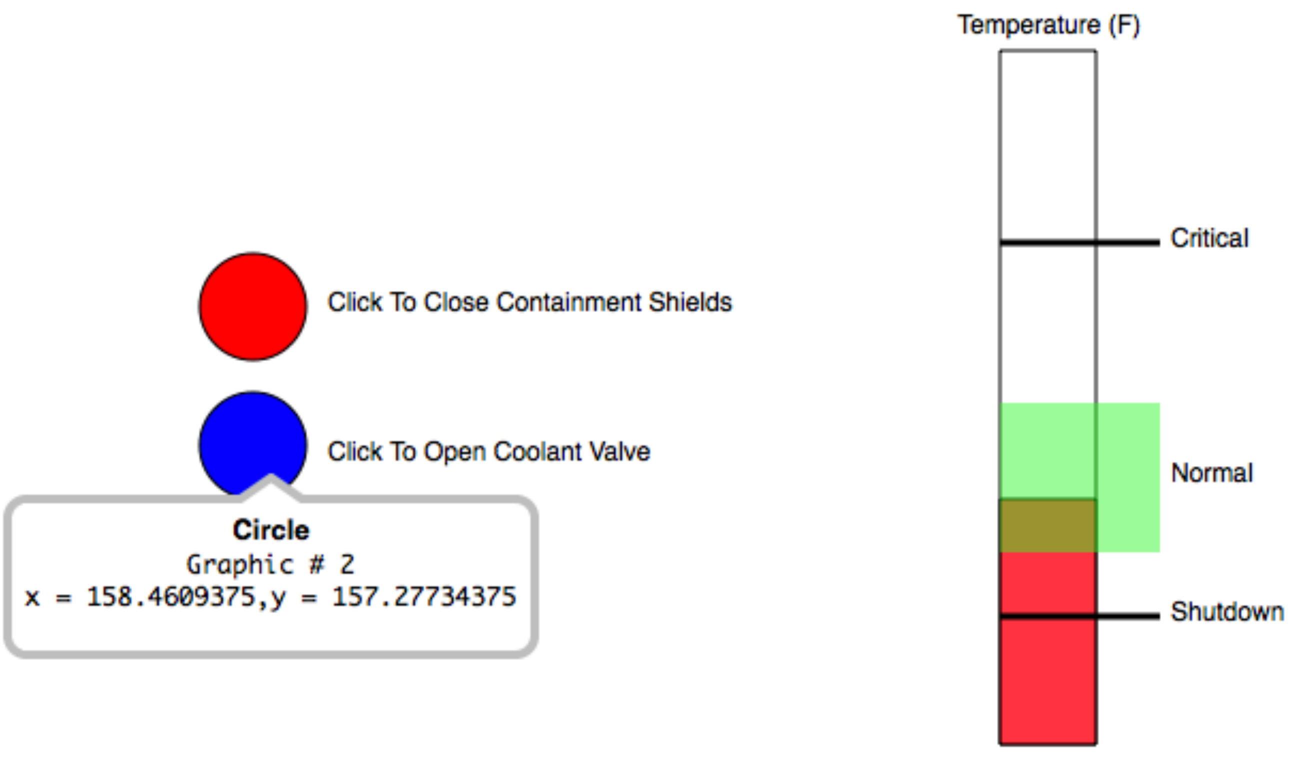
If you intend to vend Graph IDE documents via the server and Graph IDE is sandboxed (as in the Mac App Store Edition) then you must first authorize a folder for server access. You do that by selecting the Graph IDE ► Tools ► Authorize Folder ► Server Documents... menu item and selecting a folder from the Open panel. Once the access is authorized then the server can access the documents placed in the folder that you opened.

The server can also be launched from the command line (and hence a startup script with the following command:

```
/Applications/Graph\ IDE.app/Contents/MacOS/Graph\ IDE
```

The server is also available as a headless unix process that can distribute requests to a cluster. For more information on that configuration email sales@vvi.com.

Event Qualifiers are used to implement hit detection and subsequent processing by an algorithm you implement. For example, place two Circles on a Graphic View, implement the Event Qualifiers for those circles and then when the circle is clicked on an algorithm can be called to control a device such as a valve. Event Qualifiers work on data as well so you can hit a data point on a graph and interact with parameters associated with that data point. For example, a price and earnings scatter plot of symbols shows distributions and a user can click on a symbol to process results. The default behavior is to present a Information Selector such as in the following figure.


Temperature (F)
Critical
Click To Close Containment Shields
Click To Open Coolant Valve
Normal
Circle
Graphic # 2
x = 158.4609375,y = 157.27734375
Shutdown

In practice, the behavior you implement can be as simple as using the Information Selector or can be very specific and integrated. Once you have Event Qualifiers running on a document within Graph IDE then you can embed that document's Graphic View into a Custom Application to make a precise interface. The example above can be implemented on a Mac or iPhone so an operator can control a process at a station in a control center or can be driving to a remote location while touching a circle to control the process.

The capabilities of Event Qualifiers are too enormous to discuss in full detail so this section is merely concerned with an overview of the implementation, not the domain-specific details. To experiment with Event Qualifier processing select the menu item Graph IDE ► Palettes ► Event Qualifiers ► Conglomerate. That shows hit processing behavior for several objects including circles, 3D graphs and 2D graphs.

To implement the example shown in the figure above follow these steps:

- Add the circles, rectangles, text and lines to a document.
- Select each circle individually and enter the following into the circle's Program Inspector Editor and click the Apply button.

```
@@class() Circle:Object

@@method(public, class) (id)stored;

@@end

{
id myCircle;

myCircle = [Circle stored];
}
```

- Turn on Execute During Animation.
- Drag select some graphics to bring forward the layer inspector editor and in its Program Inspector Editor enter the following and click the Apply button.

```
@@class() Layer:Object

@@method(public, class) (id)stored;
@@method(public, instance) (void)enableEventQualifier;

@@end

{
id myLayer, myEventQualifier;
```

```
        myLayer = [Layer stored];

        [myLayer enableEventQualifier];
        }
```

- Execute During Animation is on by default for layers so you do not have to explicitly turn it on.

To engage the event qualifiers select the menu item Graph IDE ► Tools ► Programming ► Enable Event Qualifiers. To execute the programs select the menu item Graph IDE ► Tools ► Programming ► Execute Program. All normal event processing is now intercepted by the event qualifiers so when you mouse over a circle the Information Selector is shown. To turn off event qualifiers and restore normal event editing select the menu item Graph IDE ► Tools ► Programming ► Disable Event Qualifiers.

The following are a few things to consider when programming event qualifiers.

- Event Qualifier could have been implemented without programming. However, the idea is that a Plugin will be used to overload the Event Qualifier methods to implement your specific processing and that eventually the code for that plugin will be used by a Custom Application.
- The Information Selector behavior is optional and can be disengaged so that only your precise processing can be implemented. The Information Selector shows the parameters that are available for processing. Those parameters can include hit graphic index, hit coordinates, data point index and segment index.
- This event qualifier implementation may appear elaborate and unnecessarily complicated. However, the value that can be accessed by using it looms over the modest implementation details.

To read about the programmatic aspects of event qualifiers see the section Custom Application ► Event Qualifier.

Graph IDE ► Overview ► Glossary

Below is a glossary of words used in this manual.

Terminology	Definition
Axis	An axis is the part of a graph which shows coordinate values. For a two dimensional graph there are usually two axis. For additional information consult Linear Axis.
Coordinate	A coordinate is the geometric entity which is shown by a graph and for which data graphics are bound to. A coordinate is usually thought of as x and y paired quantities varying bi-linearly in space (Cartesian Coordinate System).
Cubic Bezier Spline	A Cubic Bezier Spline is a curve segment that is defined by four points. Two points define the ends of the curve section and are called "vertex points" and are shown as "Vertex Knobs" and the other two points define the end of (imaginary) tangent line segments which intersect the vertex points and define the slope of the curve at the end points of the curve. Those points are shown as "Knot Knobs". See Cubic Bezier for additional information.
Degenerate	Graphics are often referred to by their degenerate form. For example, a graphic which is an ellipse with pie section cut out and gradient and other graphic attributes applied has a degenerate form of a circle once parameters are simplified and is simply referred to as a circle. Graphics are many times referred to by their degenerate form within its class, and corresponding name, even though that form is not present at the time as a matter of compactness.
Dimension	A dimension is one component of a coordinate. For example, the {x,y} pairs of a coordinate have dimension x and dimension y. Compare with the definition of Unit below.
Function	A Function is a sequence of points whose x-values increase as the sequence progresses. For additional information consult Function.
Graph	A graph is a combination of graphics representing data and a single graphic which displays attributes of the coordinate system that the data is defined in. For additional information consult Introduction To Graphs.
Layer	A Layer is a grouping of graphics. You never see a layer as a layer merely forwards its implementation to other graphics. For additional information consult Layer.
Palette	A Graph IDE document that has pre-made graphics and can be accessed quickly using the Palette menu in Graph IDE. See Palette.
Point Tag	A point tag is an indicator for a point and is implemented as a marker and a label. See Point Tags for additional information.
Polygon	A Polygon is a sequence of points which are connected (in sequence) by lines. For additional information consult Polygon.
Representation	A representation is the graphical translation of data. Notably, the Spreadsheet maps data to various representations and also maintains that map (association) so that when the data is changed then the representation is updated to reflect the new data value. The representation can also be synonymous with the data, for example the use of a Function without using a spreadsheet.
RGBA	Four numbers from zero to one which correspond to the red green blue alpha channels of the color. Zero means less color component while one means most color component. Zero alpha means transparent while one alpha means opaque. For an example consult Point Tags Inspector Editor and Sequence Colors.
Selection Tool	The Selection Tool is a mode that is set so that you can mouse-select graphics (instead of creating them). It is not really a tool, but rather a mode. For additional information consult Graphic Selector. It is important to choose the selection tool after creating a graphic because you will probably want to do other things than create a graphic like reselect. Because that pattern of create and reselect is so common you can alt-click a factory tool to then create a graphic in which case the graphic selector reverts to the selection tool mode.
Segment	A segment is a graphic connecting two points and is usually a line or spline. Segments are usually arranged in a sequence because the points are arranged in a sequence. Consult Sequence Colors for an example of color indexed by a sequence of segments.
Spreadsheet	A Spreadsheet is a way of storing UNICODE textual entries within a document. For additional information consult Spreadsheet. Note that any data can reside within other graphics and that the use of a spreadsheet is optional.
Unit	A unit is defined as the type of measurement of a dimension. For a good example see Unit Selector. Note that the use of dimension in the unit definition is unrelated to that as defined above.
Vvidget	(Pronounced as "vijit") A Vvidget is anything really, as long as it implements its functionality in a predefined way. In the context of Graph IDE, a Vvidget is usually a graphical element. However, that is an unneeded, albeit convenient, restriction. The word Vvidget is a takeoff of the word Widget (with a W) which is a regular English word of a similar meaning.

Graph IDE ► Application

This section details application wide settings.

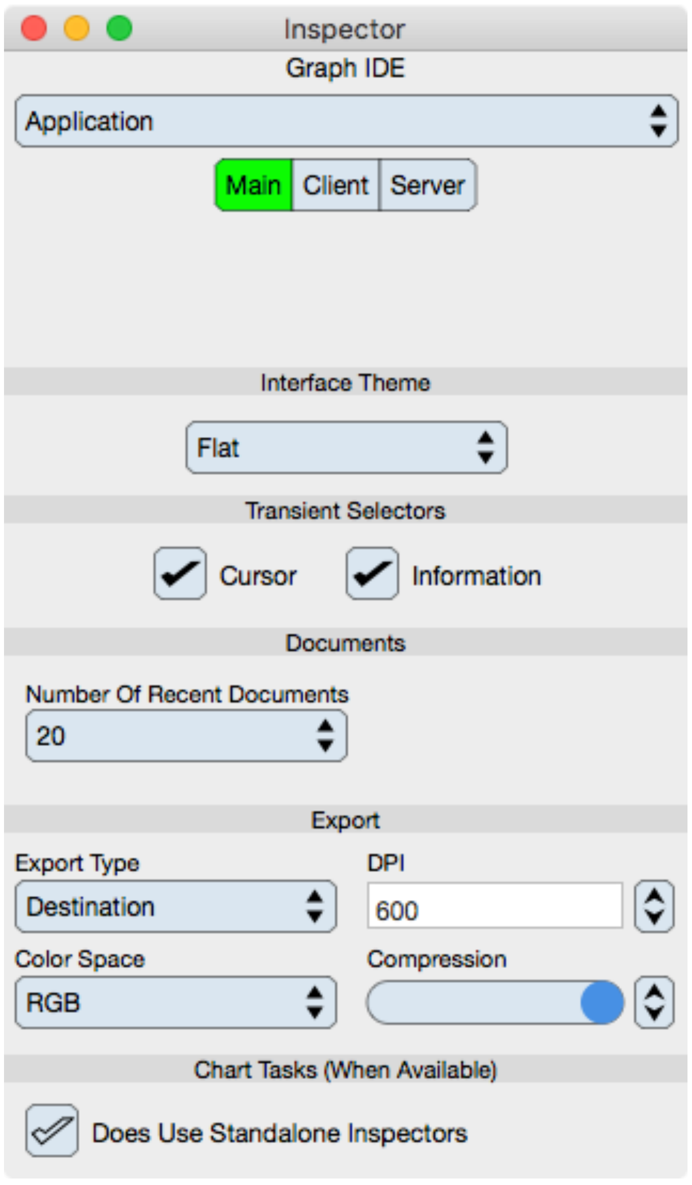
Section	Description
Administrator	The Administrator editor which is used to administer cloud accounts. This is only available to users with admin privilege.
Console	The Application Console.
Copy Service	The Application wide copy service.
Preferences	Application-wide Preferences.
Registration	Use this to register your use of the VVI Cloud Service and also to change your cloud login password.
Start/Login	This is the inspector that is shown when you first use Graph IDE. It consists of Login parameters as well as License Agreement, About and Welcome text.

Graph IDE ▶ Application ▶ Preferences

The Application-wide Preferences are accessed by the main menu Preferences or Status Bar Shortcut drop down button. The following is an explanation of each control on the Preferences editor.

Main

The Main settings define overall attributes of Graph IDE.



Interface Theme

Interface Theme : Sets the interface theme to one of original, flat or dark. The theme transforms immediately.

Transient Selectors

Cursor : When selected the Cursor Information appears on a Graphic View.

Information : When selected then various informational selectors appear over the Graphic Selector, Tables and at other locations.

Documents

Number Of Recent Documents : Sets the number of recent documents to show in the recent document menu item.

Export

Export Type : Specifies what happens when a representation is exported to another application. For the most part this should be set to the default of Destination which means the target application can query Graph IDE for the best representation.

Color Space : Determines the color space of the export. This should be left to RGB.

DPI : Determines the dots per inch of any resulting raster export.

Compression : Determines the compression amount of any resulting raster export.

Chart Tasks

This is applicable to the Chart Tasks interface when that interface is available. See Chart Tasks for additional information.

Does Use Standalone Inspectors : When checked, the inspector for a chart task is a standalone window, otherwise it is a sheet.

Client

The Client editor defines a custom cloud host to connect to.

Custom Cloud Account Server

Custom Cloud Host Name : A DNS host that provides cloud service. If an entry is provided then it will appear in the host entries for the Login editor. The entry must be a valid DNS name for example www.vvi.com, www.local or any entry that your domain name service can lookup.

Facility Settings

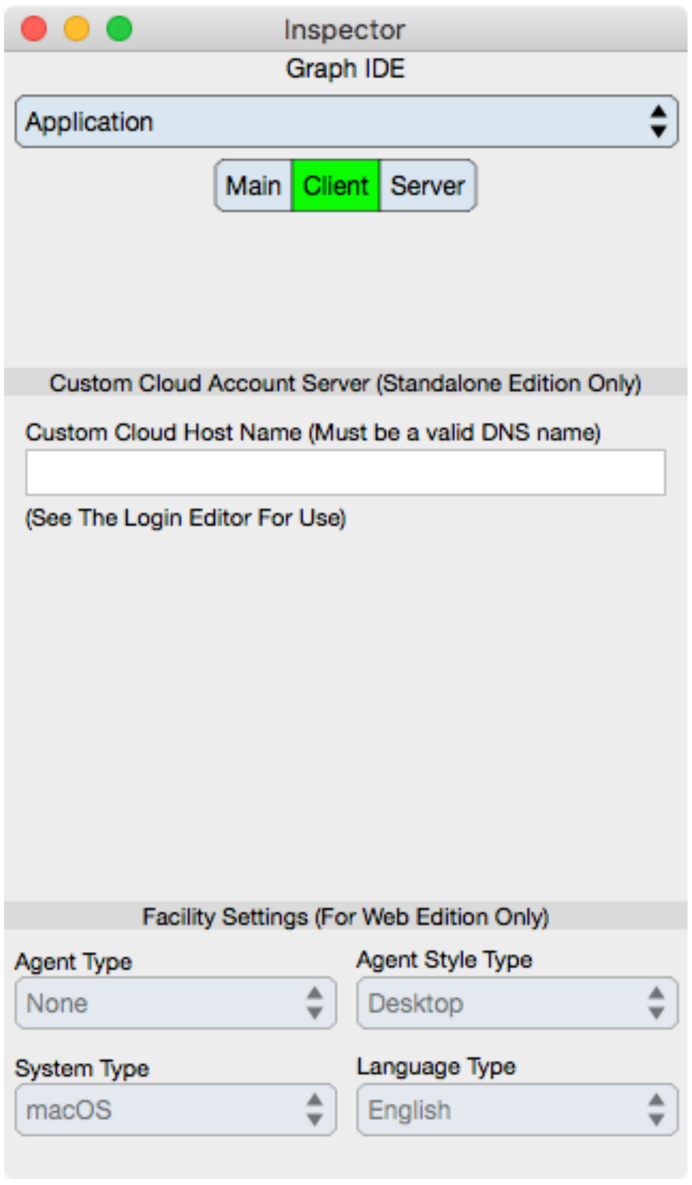
These settings are only enabled for the web edition. They are related to browser settings and can be overridden by the user so should not be relied upon.

Agent Type : The browser client agent type.

Agent Style Type : The browser client agent style type.

System Type : The browser client system type.

Language Type : The browser client language type.



Custom Cloud Account Server (Standalone Edition Only)

Custom Cloud Host Name (Must be a valid DNS name)

(See The Login Editor For Use)

Facility Settings (For Web Edition Only)

Agent Type: None

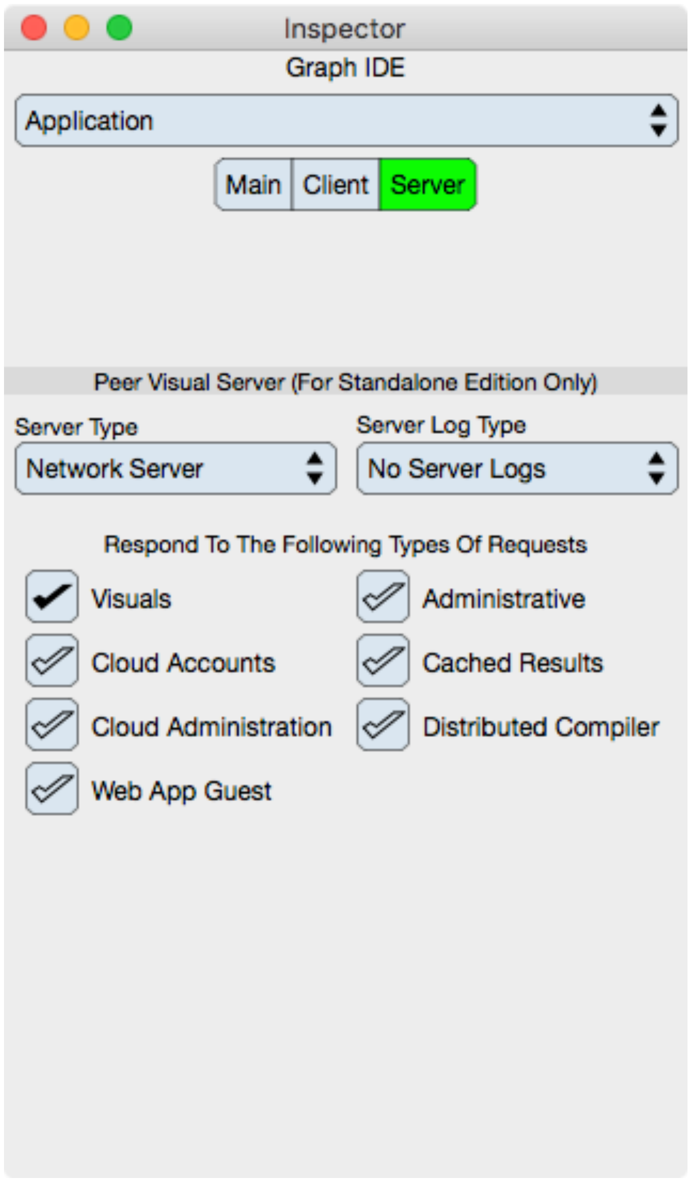
Agent Style Type: Desktop

System Type: macOS

Language Type: English

Server

The Server editor controls vending of results to network clients. Keep the server off unless you have a specific need.



Peer Visual Server (For Standalone Edition Only)

Server Type: Network Server

Server Log Type: No Server Logs

Respond To The Following Types Of Requests

- [✓] Visuals
- [ ] Administrative
- [ ] Cloud Accounts
- [ ] Cached Results
- [ ] Cloud Administration
- [ ] Distributed Compiler
- [ ] Web App Guest

Peer Visual Server

Server Type : Set to Network Server to start the internal network server or No Server to not start the server. See Server for additional information.

Server Log Type : Sets the level of logging for the server. Since the server logs always goes to stdout and stderr the log only make sense when those system logs can be accessed.

Visuals [1,2] : Select if you wish the server to vend visual results based upon SOA requests. In other words, make graphs for clients.

Cloud Accounts [1,2,3] : Select if you wish client Graph IDE applications to Login to the server (and save documents on the server).

Cloud Administration [1,2,3] : Select if you wish client Graph IDE applications to Administer cloud accounts on the server. It is recommended that you leave this unchecked unless you are actively adding or modifying cloud accounts.

Web App Guest [1,2,3] : Select if you wish web browsers to login remotely as guests without an account.

Administrative [1,2] : Select if you wish the server to respond to administrative queries. This should probably be left off unless there is an explicit need for it.

Cached Results [1,2] : Select if you wish the server to cache results for later retrieval. This should probably be left off unless there is an explicit need for it.

Distributed Compiler [1] : Select if you wish the server to act as a SOA distributed compiler and execution engine. This should probably be left off unless there is an explicit need for it.

1. Requires knowledge about network security and appropriate configurations.

2. Acts in conjunction with local network clients such as vvizard.

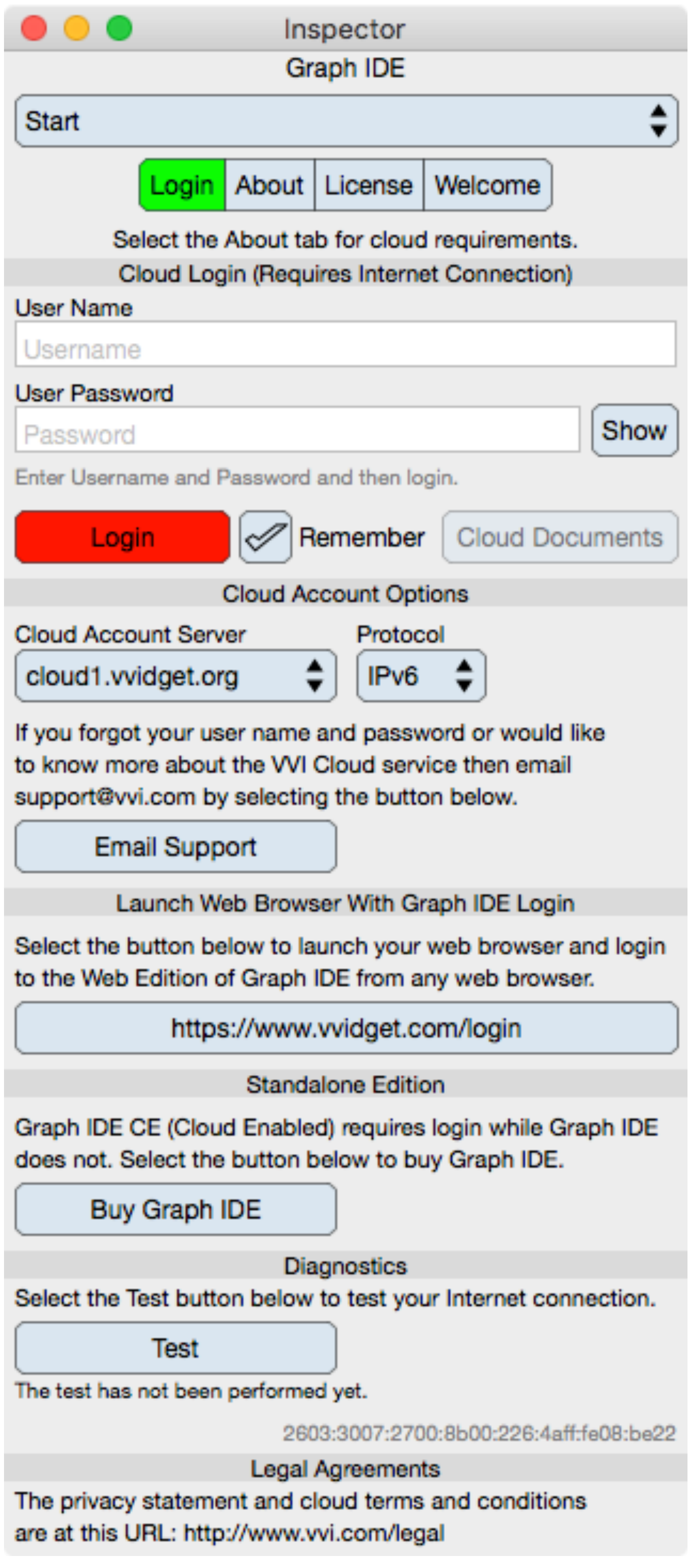
3. Only functions in production situations with a headless version of the server.

Graph IDE ► Application ► Start/Login

The Start inspector first shows when you begin to use Graph IDE. It consists principally of the Login editor. The login editor can also be accessed by selecting the Login button in the Status Bar.

Login Editor

The Login Editor is shown below. Login is optional for Graph IDE, but is required for the "Cloud Enabled" version of Graph IDE.



Select the About tab for cloud requirements.

Enter Username and Password and then login.

If you forgot your user name and password or would like to know more about the VVI Cloud service then email support@vvi.com by selecting the button below.

Select the button below to launch your web browser and login to the Web Edition of Graph IDE from any web browser.

Graph IDE CE (Cloud Enabled) requires login while Graph IDE does not. Select the button below to buy Graph IDE.

Select the Test button below to test your Internet connection.

The test has not been performed yet.

2603:3007:2700:8b00:226:4aff:fe08:be22

The privacy statement and cloud terms and conditions are at this URL: http://www.vvi.com/legal

Cloud Login

User Name : Enter the user name provided to you when you signed up to the VVI Cloud service.

User Password : Enter the user password provided to you when you signed up to the VVI Cloud service.

Login : Once the user name and password are entered then select the Login button to login. Once logged in then proceed to Registration in order to register your use of the Cloud Service including changing the initial password.

Remember Login : Select to remember the login for the next time Graph IDE is used. This is not available for the web edition of Graph IDE for security purposes and instead the login parameters are determined by the browser password management facilities.

Note: To forget the login first clear the user name and password fields and then uncheck the Remember Login button. The next time Graph IDE is launched it will not attempt login.

Cloud Documents : Select to proceed to the table of available cloud documents and controls for cloud documents. See Account. A shortcut to cloud documents is also provided on the Status Bar shortcut drop down.

Cloud Account Options

Host Name : The cloud host which is fixed to cloud1.vvidget.org unless a Custom Cloud Host Name entry is provided in the Preferences in which case that host can be selected and used as well.

IP Protocol : Either IPv6 or IPv4. The preferred protocol is IPv6 however if that is not available then IPv4 will be automatically selected. If either is available then the protocol type can be changed according to your preference.

Email Support : Select this button to launch the email application preconfigured with the support@vvi.com email address.

Launch Web Browser With Graph IDE Login

Web Login Button : Select this button to login to Graph IDE via your web browser at https://www.vvidget.com/login. With a web login you can access and modify your cloud-stored documents but there is no ability to use local processing or the file system on your device.

Standalone Edition

Buy Non-Cloud Edition : Select this button to be directed to the purchase instructions for the non-cloud version of Graph IDE.

Diagnostics

Test : Select this button to test the Internet connection to the VVI Cloud service. If the test fails then the most likely cause is that your Internet connection is down. The test checks your internet connection and also tests for a valid response from the VVI Cloud server.

IP Address : The grey text shows the DNS resolved IP Address of the current cloud server. If this is not resolvable then an appropriate error message is given instead.

Legal Agreements

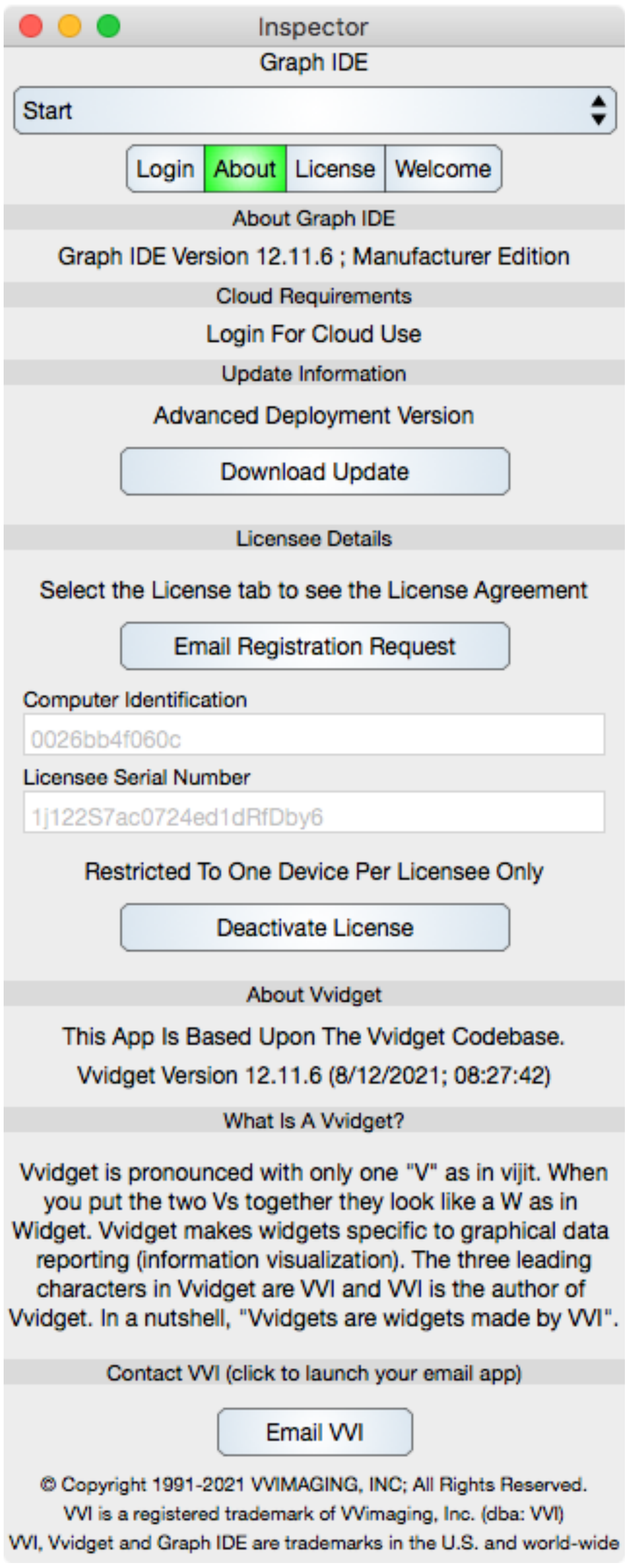
The cloud service is provided in accordance with the agreement at Legal. That Agreement is also within this manual at Cloud Agreement and is subject to change.

About Editor

The About Editor is shown below.

About Graph IDE

Version ; Edition : Shows the version number and edition type.


Inspector
Graph IDE
Start
Login | About | License | Welcome

About Graph IDE
Graph IDE Version 12.11.6 ; Manufacturer Edition

Cloud Requirements
Login For Cloud Use

Update Information
Advanced Deployment Version
Download Update

Licensee Details
Select the License tab to see the License Agreement
Email Registration Request
Computer Identification: 0026bb4f060c
Licensee Serial Number: 1j122S7ac0724ed1dRfDby6
Restricted To One Device Per Licensee Only
Deactivate License

About Vvidget
This App Is Based Upon The Vvidget Codebase.
Vvidget Version 12.11.6 (8/12/2021; 08:27:42)

What Is A Vvidget?
Vvidget is pronounced with only one "V" as in vijit. When you put the two Vs together they look like a W as in Widget. Vvidget makes widgets specific to graphical data reporting (information visualization). The three leading characters in Vvidget are VVI and VVI is the author of Vvidget. In a nutshell, "Vvidgets are widgets made by VVI".

Contact VVI (click to launch your email app)
Email VVI

© Copyright 1991-2021 VVIMAGING, INC; All Rights Reserved.
VVI is a registered trademark of Vvimaging, Inc. (dba: VVI)
VVI, Vvidget and Graph IDE are trademarks in the U.S. and world-wide

Cloud Requirements : Shows the cloud usage requirement. The standalone version of Graph IDE can be used without the cloud unless you wish to take advantage of cloud features; while the Cloud Enabled edition requires cloud login.

Update Information

Current Version : Indicates the version of your copy of Graph IDE as compared to the current version of Graph IDE. This is unavailable for the App Store Edition.

Download Update : Select to update from the App Store for the App Store Edition, or from https://www.vvidget.com/download for the Manufacturer Edition.

Licensee Details

Email Registration Request : Select to launch the default mail application with a pre-formatted email message to registrar@vvi.com. Once all the required information is filled out in that email then send the message. A reply will include a serial number.

Computer Identification : The identification used to generate the serial number.

Licensee Serial Number : The serial number for the application. Once received from a registration request then enter the serial number and select the Save Serial Number button.

Restriction Label : Shows the current restrictions based upon the serial number. For the Cloud Enabled edition the serial number does not need to be entered as the application is authorized by cloud account instead.

Save Serial Number : Select to save the serial number. When a valid serial number is saved then the application will operate in normal mode (non-demo). Once a valid serial number is entered then this save button becomes the Deactivate License button so that the computer can become deauthorized. Once deathorized then delete the application to uninstall.

About Vvidget

Gives the underlying Vvidget Frameworks version and build date.

What Is A Vvidget?

Explains the use of the Vvidget name.

Contact VVI

Email VVI : Select to email info@vvi.com.

License and Welcome Editor

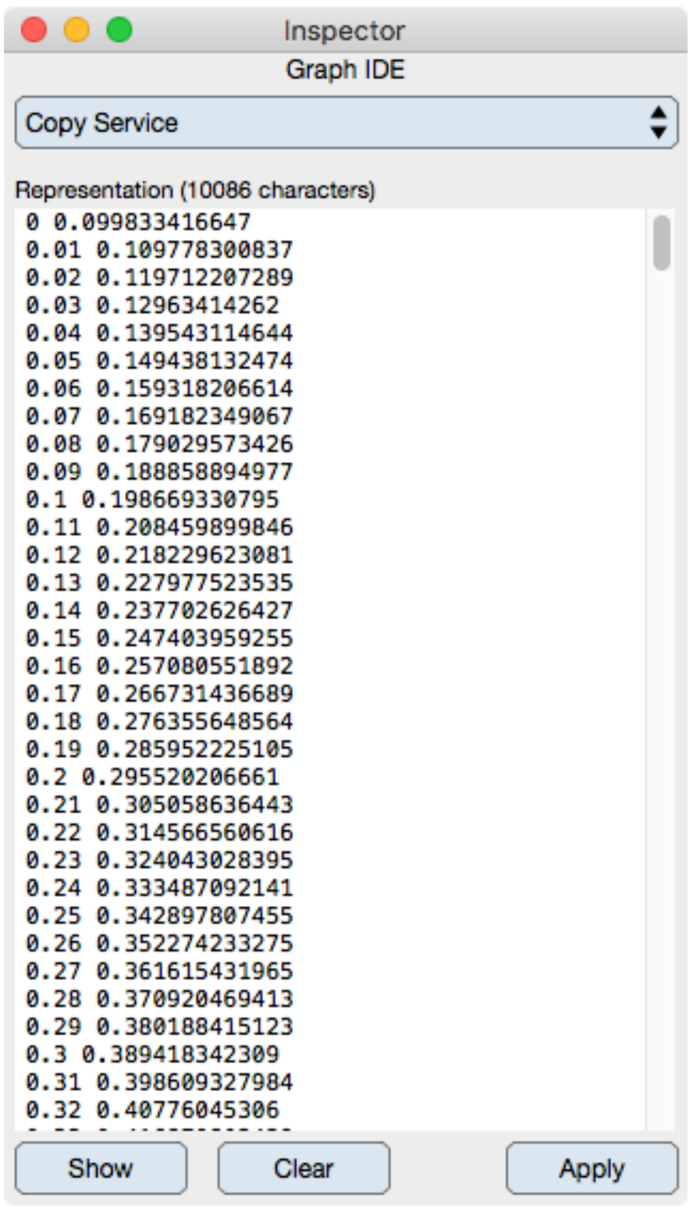
The License Editor shows the End User License Agreement (EULA). The Welcome Editor is shown when Graph IDE is first launched upon installing a newer version of Graph IDE.

Graph IDE ► Application ► Copy Service
a.k.a: Clipboard or Pasteboard

The Application-wide copy service is used while copy and pasting data and graphical information. It is usually never seen explicitly. The text store can be viewed using the Copy Service inspector editor as shown below.

A note about this copy service inspector: For a desktop or tablet native implementation this inspector is convenient to see what is on the pasteboard. Text can be copied and pasted into the textual data and by virtue of the paste, it is also pasted into the pasteboard itself. Not so for web browsers. In a web browser, the textual representation is represented in an expanding text box with appropriate div DOM nodes. When you copy or paste into that text box you are utilizing the clipboard (aka: pasteboard) of your browser and local device. To get the text to the server process space you need to click the Apply button which then transmits the data from the browser view to the server facilities (typically on a different computer).

You can load data directly from a file on a client to a cloud server process using the Export And Import facilities but if your data is not in a file and you do not wish to edit it into the tables directly then this Copy Service interface is an excellent way to transmit data for importing into a table with the normal paste operations.



```
0 0.099833416647
0.01 0.109778300837
0.02 0.119712207289
0.03 0.12963414262
0.04 0.139543114644
0.05 0.149438132474
0.06 0.159318206614
0.07 0.169182349067
0.08 0.179029573426
0.09 0.188858894977
0.1 0.198669330795
0.11 0.208459899846
0.12 0.218229623081
0.13 0.227977523535
0.14 0.237702626427
0.15 0.247403959255
0.16 0.257080551892
0.17 0.266731436689
0.18 0.276355648564
0.19 0.285952225105
0.2 0.295520206661
0.21 0.305058636443
0.22 0.314566560616
0.23 0.324043028395
0.24 0.333487092141
0.25 0.342897807455
0.26 0.352274233275
0.27 0.361615431965
0.28 0.370920469413
0.29 0.380188415123
0.3 0.389418342309
0.31 0.398609327984
0.32 0.40776045306
```

Representation : Type or paste text into the representation view. When initially loaded, the text is the current contents of the pasteboard. The title also shows the current number of characters in the copy service. As you type or paste into the representation view the text is not applied so will be a different number of characters in the copy service until the Apply button is selected.

Show : Shows the current text in the copy service. This may be different than what is in the representation view because various other operations can effect the copy service without updating the representation view and visa versa.

Clear : Clears the copy service. This clears what is in the copy service text buffer and also clears all other copy service buffers, in particular the copy service associated with binary archives that are used to copy and drag graphics.

Apply : Select Apply to transmit the text from the representation view to the pasteboard buffer within the Graph IDE process. For native implementations, the transmission occurs within Graph IDE itself, but for web clients the transmission is from the browser to the Graph IDE cloud service that you logged in at and is only within your own account process space.

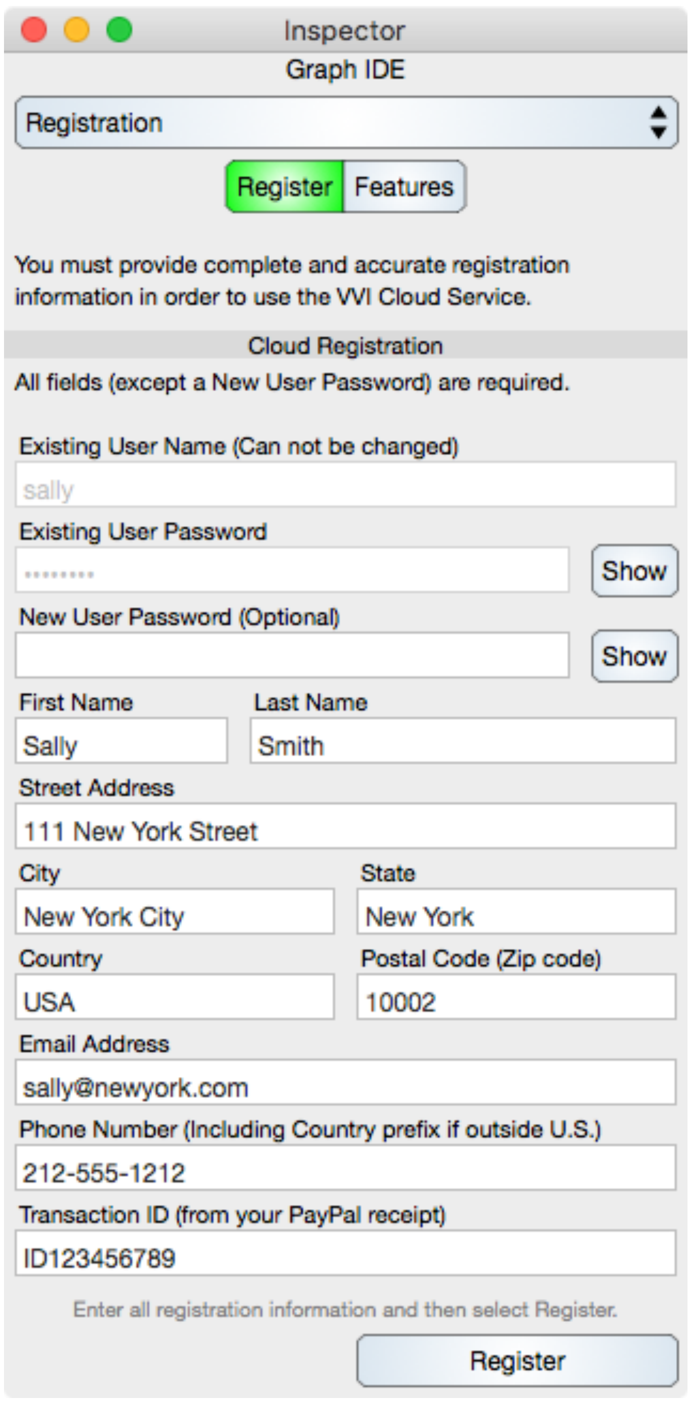
Whether using a native or client version of Graph IDE, the Copy Service inspector editor is a convenient way to use the pasteboard as a scratch area for data editing and subsequent use in other components of Graph IDE that can paste data, namely Tables.

Graph IDE ► Application ► Registration

The Registration inspector is used to register your use of the VVI Cloud Service and also change your account password. It is accessed via the Preferences sub-inspector which is the same way to access the Start/Login editor.

Please make sure to register your use of the cloud service and provide complete and accurate information.

Register Editor



You must provide complete and accurate registration information in order to use the VVI Cloud Service.

Cloud Registration

All fields (except a New User Password) are required.

Existing User Name (Can not be changed)	sally
Existing User Password	••••••••
New User Password (Optional)	
First Name	Sally
Last Name	Smith
Street Address	111 New York Street
City	New York City
State	New York
Country	USA
Postal Code (Zip code)	10002
Email Address	sally@newyork.com
Phone Number (Including Country prefix if outside U.S.)	212-555-1212
Transaction ID (from your PayPal receipt)	ID123456789

Enter all registration information and then select Register.

Cloud Register

Existing User Name : The name that was given to you when you signed up for the VVI Cloud Service and which you logged in with using the Start/Login editor.

Existing User Password : The existing password of your VVI Cloud Account.

New User Password : Enter a password if you wish to change the password of your account or leave it blank to keep the existing password.

First Name : Your first name.

Last Name : Your last name.

Street Address : Your physical location street address.

City : The name of the city that you live in.

State : The name of the state (or province) that you live in.

Country : The country that you live in.

Postal Code : The postal code of your physical address.

Email Address : Your permanent email address (must not be temporary and must be an active account).

Phone Number : Your phone number.

Transaction ID : The Transaction ID from your PayPal receipt or if you purchased it under an umbrella P.O then the purchase number of that P.O.

Register : Select the register button to register your use of the VVI Cloud Service. All fields must be valid and you must be logged into your account.

Features Editor

The Features Editor shows features that are enabled for the logged-in cloud account.

Features

Date Account Was Created : The date that the account was created.

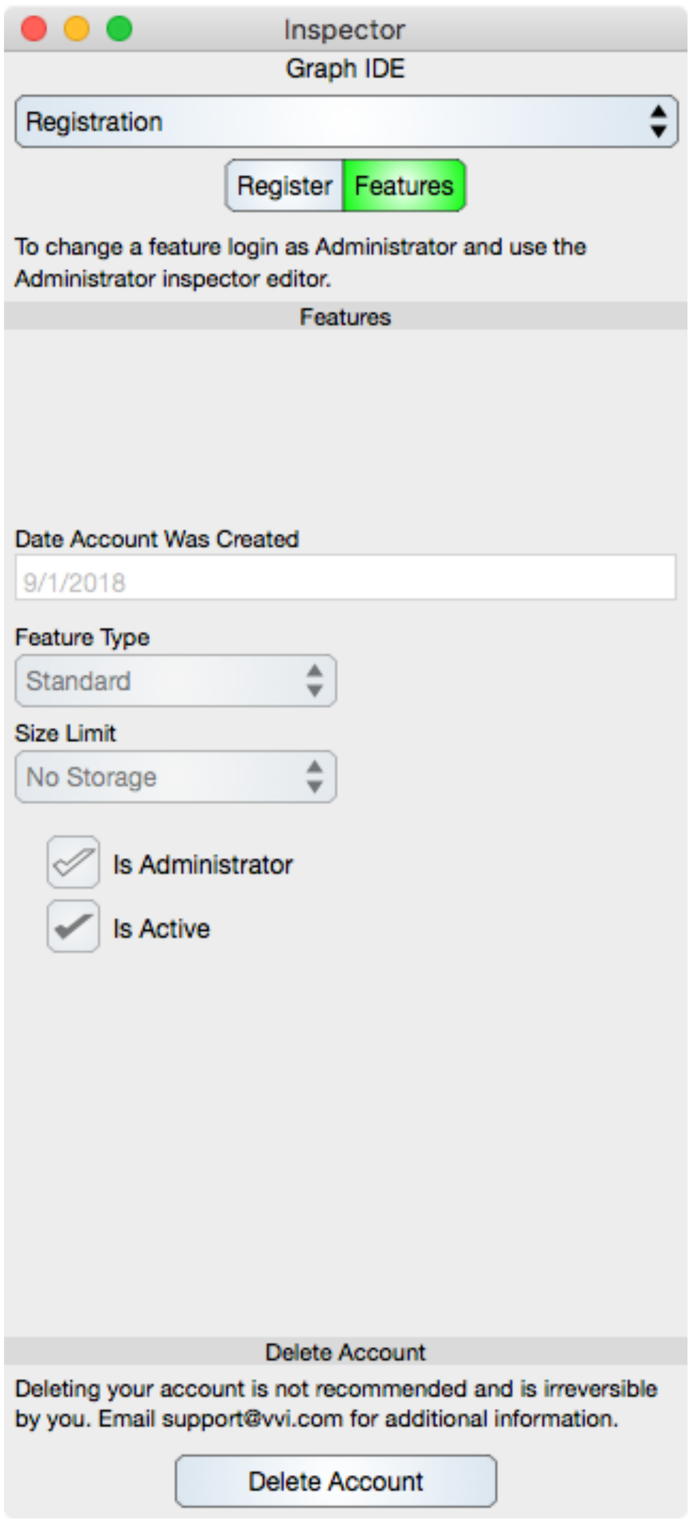
Feature Type : Standard or Pro.

Size Limit : No Storage or 100MB. This defines the amount of space authorized for use on the server in order to store Graph IDE documents.

Is Administrator : Checked if the account is an admin account.

Is Active : Checked if the account is active. A nonactive account can not be used so this button will always be checked.

Delete Account : Select this button to delete your account from the active accounts. If you delete your account then you will not be able to access it again. It may be possible to retrieve your account documents after deletion. If you deleted your account in error then email support@vvi.com as soon as possible to see if you account, and its documents, can be retrieved. A deleted account may not be able to be retrieved.



To change a feature login as Administrator and use the Administrator inspector editor.

Features

Date Account Was Created

9/1/2018

Feature Type

Standard

Size Limit

No Storage

☐ Is Administrator

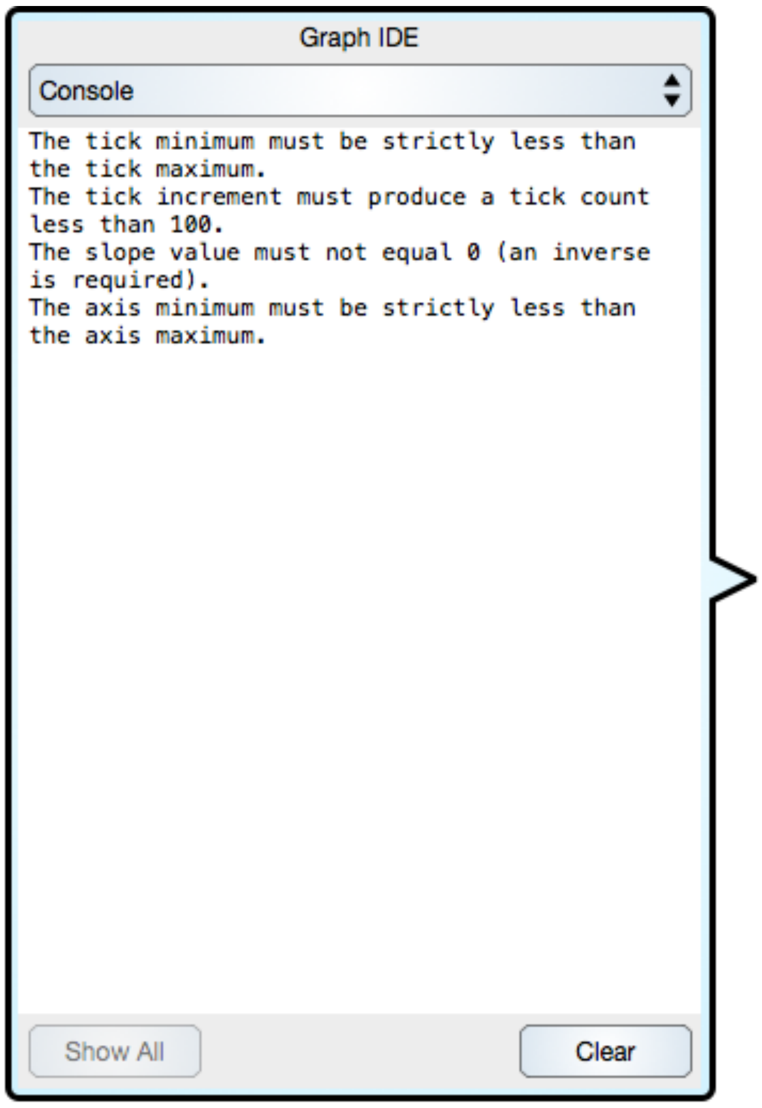
✓ Is Active

Delete Account

Deleting your account is not recommended and is irreversible by you. Email support@vvi.com for additional information.

Graph IDE ► Application ► Console

The Application-wide Console is accessed as a subeditor of the Application editor. It can be brought forward explicitly as a subeditor of the Application Preferences. It also comes forward during a warning message as shown below.



```
The tick minimum must be strictly less than
the tick maximum.
The tick increment must produce a tick count
less than 100.
The slope value must not equal 0 (an inverse
is required).
The axis minimum must be strictly less than
the axis maximum.
```

Console Text

Shows any text in the console.

Settings

Show All : Select to show all the console messages. If the console is brought forward during a warning message then only the last line in the console is shown. To show all lines select this button.

Clear : Select to clear all messages from the console. Clearing permanently deletes all messages.

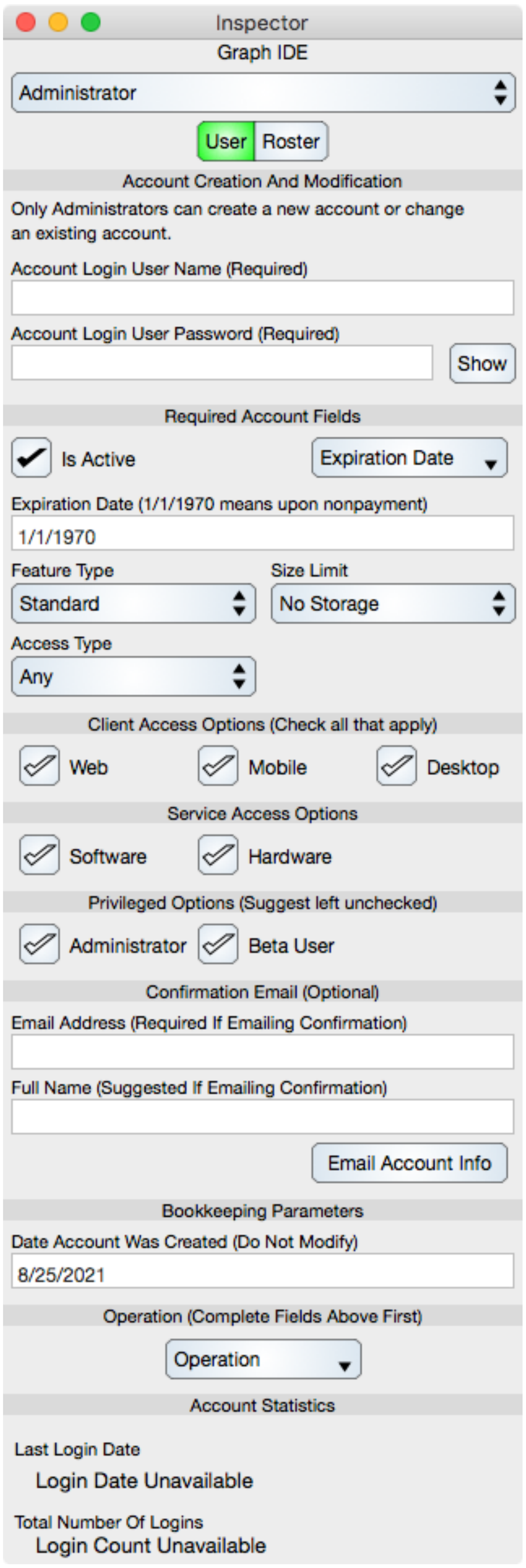
Graph IDE ▶ Application ▶ Administrator

For Standalone Edition only. Accessible only with Admin user privilege on the cloud service.

The Administrator inspector is used to create, modify and delete user accounts on the cloud server. It is accessed via the Preferences sub-inspector which is the same way to access the Start/Login editor.

If you have Admin privileges (see Account to prime an administrator account) then the Administrator Editors can be used to establish accounts on the Cloud service so that other client Graph IDE applications can login. The Administrator editors will not work until both Cloud Accounts and Cloud Administration options are selected on the cloud service (see Preferences).

User Editor


Inspector
Graph IDE
Administrator
User
Roster
Account Creation And Modification
Only Administrators can create a new account or change an existing account.
Account Login User Name (Required)
Account Login User Password (Required)
Show
Required Account Fields
Is Active
Expiration Date
Expiration Date (1/1/1970 means upon nonpayment)
1/1/1970
Feature Type
Standard
Size Limit
No Storage
Access Type
Any
Client Access Options (Check all that apply)
Web
Mobile
Desktop
Service Access Options
Software
Hardware
Privileged Options (Suggest left unchecked)
Administrator
Beta User
Confirmation Email (Optional)
Email Address (Required If Emailing Confirmation)
Full Name (Suggested If Emailing Confirmation)
Email Account Info
Bookkeeping Parameters
Date Account Was Created (Do Not Modify)
8/25/2021
Operation (Complete Fields Above First)
Operation
Account Statistics
Last Login Date
Login Date Unavailable
Total Number Of Logins
Login Count Unavailable

Account Creation And Modification

Account Login User Name : The user name of the account that will be either created or updated.

Account Login User Password : The user password of the account that will be either created or updated.

Required Account Fields

Is Active : Leave selected unless you wish to deny access to the account. The account otherwise remains unaltered (all documents are preserved). If the account is permanently inactive then consider deleting it using the Operation described below.

Expiration Date : The date that the account will expire. In the pop up button, choose Upon Nonpayment for an account that expires when the subscription is cancelled or One Year to expire one year from the date the option is chosen. Alternatively, enter a date in the text field. A date of 1/1/1970 means Upon Nonpayment option.

Feature Type : Standard or Pro.

Size Limit : No Storage or 100MB. This defines the amount of space authorized for use on the server in order to store Graph IDE documents.

Access Type : One of Any, Write Only or Read Only. Usually this is Any signifying that the account can be used to write and read documents. You may want this set to Read Only if the account is used only to view documents.

Client Access Options

Web : Login via a web browser is enabled.

Mobile : Login via the native tablet or mobile app is enabled.

Desktop : Login via the native desktop application is enabled.

Service Access Options

Software : Software services are available.

Hardware : Access to data acquisition hardware attached to the computer is available.

Privileged Options

Is Administrator : Select only if the account is used to administer other cloud accounts. Leave this to off. To establish a first administrative account, the admin account must be primed at the command line on the server host (see Account).

Is Beta : Select only if the account can access beta features.

Confirmation Email

Use these fields in order to generate a confirmation email for a new account. Select either the Email Account Info button or fill out the fields, create the account and then choose the Email Confirmation button at the end of creating the account.

Email Address : The address that will be used when mailing new account notification.

Full Name : The name used for salutations in the new account email.

Email Account Info : Select this button to email new account notification to the user specified in the email field.

Bookkeeping Parameters

Date Account Was Created : The date that the account was or is created. If a new account then this date is the current date and should probably remain at that date.

Operation

Operation : One of Create, Delete or Get Features. If Create then all fields must be set, if Delete or Get Features then the User Name must be specified. If an attempt to Create an existing account is made then the operation will be an update to that account (all documents will be preserved). Destructive operations like Delete or Update are prefaced with an accept modal dialog.

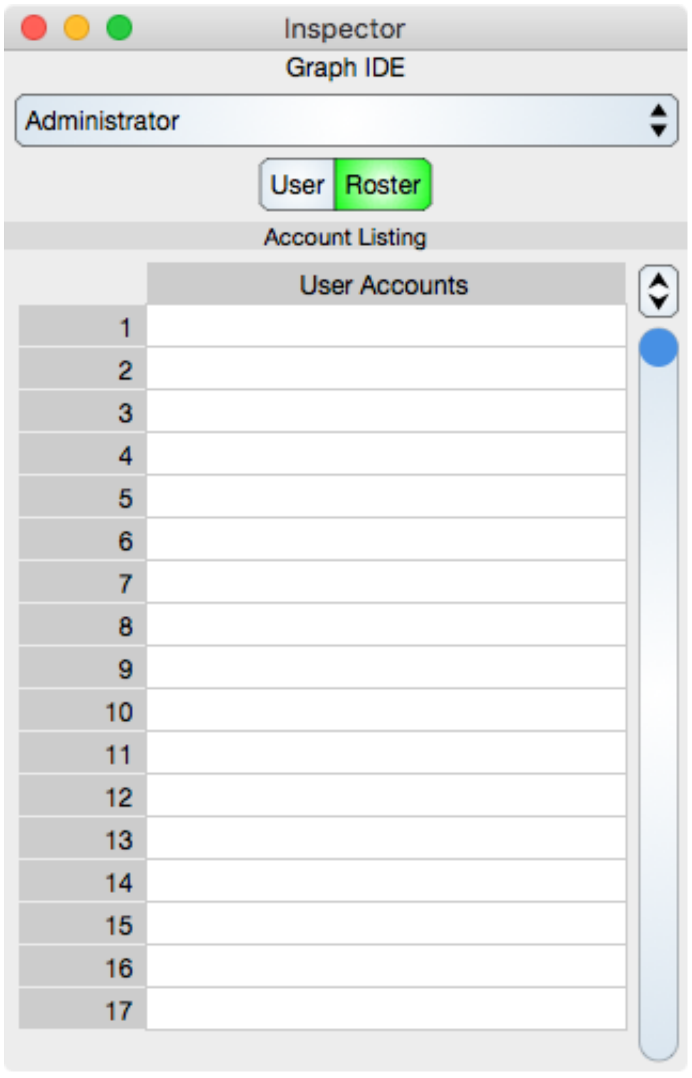
Account Statistics

These labels are populated when features are retrieved from the cloud server.

Last Login Date : Shows the date that the user last logged in.

Total Number Of Logins : Shows the total number of successful logins by the user.

Roster Editor



Account Listings

Table : A list of the user accounts on the cloud server. Select a row to populate the User fields described above for that user.

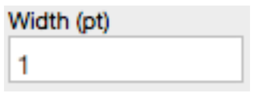
Graph IDE ► Controls

General controls specific to Graph IDE are detailed in the following sections.

Section	Description
Color Selector	Details the color selector which is a control for adjusting colors.
Data Selector	Details the data selector which shows and edits one piece of data.
Dial	Details the dial control which is a control for adjusting an angle.
Font Selector	Details the font selector which is a control for adjusting font name and size.
Format Selector	Details the format selector that is used to format cells in a spreadsheet on a per-column basis.
Formula Selector	Details the format selector that is used to define and compute cell values for a spreadsheet on a per-column basis.
Information Selector	Details the information selector which shows information in a transient selector.
Menu Selector	Details the menu selector control which is a control for adjusting a list of options.
Number Selector	Details the number selector control which is a control for adjusting a single number.
Slider	Details the slider control which is a control for adjusting a single number within a preset interval.
Standard Editing	A description of the standard mouse and keyboard editing actions performed on Vvidgets for editing purposes.
Text Field	Details the text field which is a control to enter a single line of text.
Unit Selector	A description of the unit selector for choosing coordinate dimension and units.

Other control descriptions are found throughout this manual, in particular the Inspector Editors section and each Basic Graphics, Data Graphics and 3D Data Graphics section.

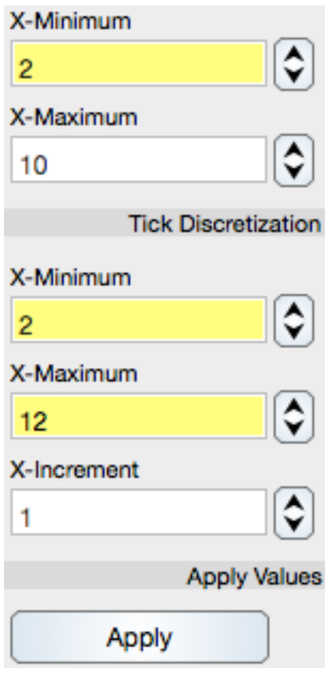
A text field is used to enter a single line of text. The figure below shows a text field on a window background.



Select the text field and start typing. As text is entered, there are three background changes as follows.

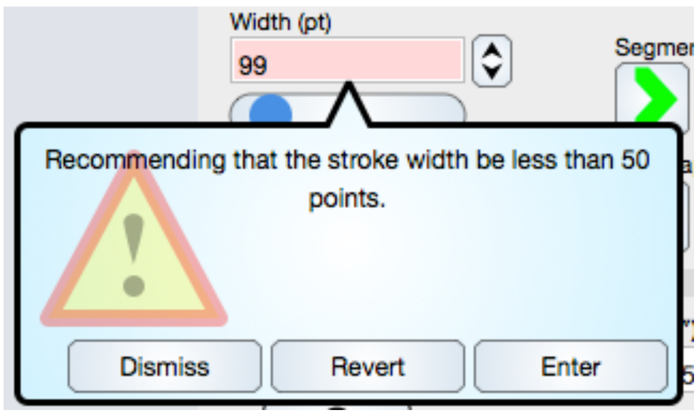
Background	Description
White	The text has not been edited (is fresh)
Yellow	The text has been edited (is dirty)
Red	The text has an error

The following figure shows three textual values that have been edited (are dirty) but not entered and two textual values that have not been edited:



All entries are entered only when the Apply button is selected, see Graphs for an explanation.

The following figure shows text that has an error:



Recommending that the stroke width be less than 50 points.

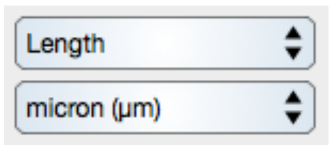
In this case, the error is associated with a numeric value that is outside a suggested range. The value can still be entered by selecting the Enter button or typing the return character. If the error was hard then the text field will refuse to propagate its value until the error is corrected.

Text fields work in cooperation with several types of validators so that a text field is actually much more than simple textual data entry. Each new character typed into a text field is validated as it is entered so that there is immediate feedback. Some text fields such as on the Color Selector are very specific, for example specifying a hex encoded byte of data.

Unlike some systems, text is not entered upon tabbing out of a text field and an explicit return key, or Apply button, et. al., is required to enter the text. Thus the yellow background to show that text is dirty but not entered.

Graph IDE ► Controls ► Unit Selector

A Unit Selector is a combination of two pop up buttons, as shown in the following diagram, along with their respective menus.

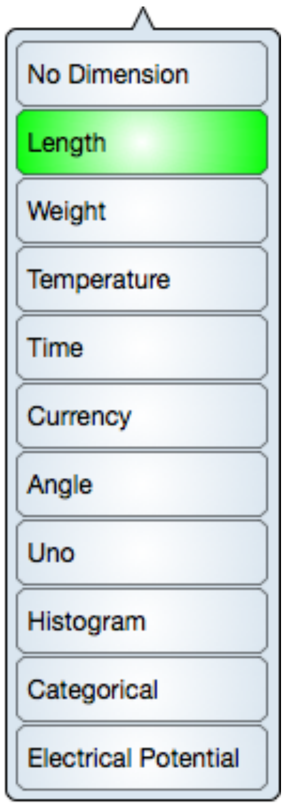


First, set a dimension using the top pop up button and then set a unit that is appropriate for that dimension using the lower pop up button.

The Unit Selector appears on many of the inspector editors, for example the Single Coordinate Graph and the spreadsheet. A Unit Selector without specifying dimension is incorporated into the Graphic View editor. In that case, the dimension menu is not required because the dimension is always length.

The unit type is not enough to specify a unit and the dimension is also required. For example, ounce is both a weight and volume. In Graph IDE (particularly its coding) the dimension and unit type are referred to as Quantity Type and the value with a particular unit is referred to as a Quantity Value. The type and value together are referred to as Quantity. There is another concept called Depiction which is the way the Quantity is represented. See Color Selector for an example of that.

Dimension Selector



Use the dimension selector (pop up button menu) to select a dimension. This diagram shows length as the selected dimension type. Once the dimension is selected then the unit selector is populated with values appropriate to that dimension.

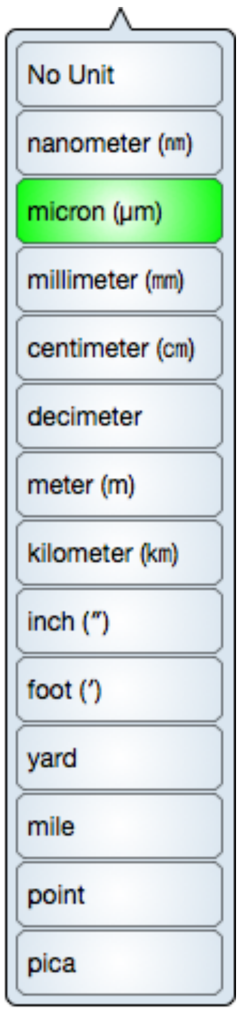
See the table below for a description of the types.

No Dimension	Specifies that the corresponding value is unitless.
Length	The length dimension includes metric, imperial and typographic units
Weight	The weight dimension includes metric, imperial, and atomic mass units.
Temperature	The temperature dimension includes metric, imperial and physics (kelvin) units.
Time	The time dimension includes imperial and calendar units. Note that that time is split between uniform and non-uniform units. For example, each second is the same span of time whereas a month can be a span of time from 28 to 31 days in the Gregorian calendar and a different amount of days in a different calendar system. The unit types do not imply the calendar type.
Currency	The currency dimension includes many international currencies. Currency is a dimension with units but the scale between units varies constantly (via exchange rates) making this dimension particularly interesting. In many systems Currency is not thought of as a fundamental dimension.
Angle	The angle dimension is either (imperial) degrees or radians. By convention, all angle units, such as rotation, are specified in degrees.
Uno	The uno dimension specifies units as decimal increments of the value one. For example, percent, parts per decimal base, etc. This is particularly needed when a value is expressed in percentages.
Byte	The byte dimension specifies units as bytes, both 2 (e.g. kilo) and 10 (e.g. kibi) base.
Histogram	The histogram dimension is reserved for histograms only. Typically, this is not considered a dimension but because histograms are available as a graph then it is convenient to give the graph dimension a unique unit.
Categorical	The Categorical dimension specifies some discrete dimension types. This is useful for bar charts, but is

	also useful for counting controls such as a table column or row specifier where there are not fractional amounts.
Electrical Potential	The electrical potential dimension specifies several metric volt units. Note that this dimension is usually synonymous with its unit because electrical potential is only specified in volts and there is no other unit type.

Unit Selector

Use the unit selector (pop up button menu) to select a unit. This diagram shows possible length units with the micron unit selected. If the unit has a symbol then that appears in parenthesis after the unit name.



Units are a standard concept and are for the most part self explanatory. However, the categorical dimension and corresponding units may require a bit of explanation as given in the table below.

The categorical dimension is not really a dimension per se and its units are really set-theoretic concepts. However, it is incorporated into the unit controls because many controls and graphs have values that are thought of in terms of a discrete variable all homomorphic to counting numbers. Note that this dimension is not named "discrete" because all other dimensions can have a discrete quality (for example, currency can be in basis points).

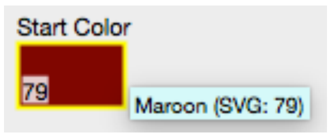
Nominal	This is the typical categorical type in statistics where the variables do not have an order. Note that in some statistical circles, Nominal is simply referred to as Categorical and Ordinal is a subtype of Categorical.
Ordinal	This is the typical categorical type in statistics where the variables have an order. Although this is simply conceptual, many dimensions are ordinal in the sense that they have a metric operation for ordering.
Dichotomous	Dichotomous is the binary unit of yes or no. It can also be referred to in many depictions such as the word "choice" or "on" and "off". Note the strong distinction between unit and depiction.
Integer	Integer is all non-fractional numbers. This is a set theoretic concept and not necessarily a unit.
Whole Number	All positive integers including zero are whole numbers. The worth of a unit is in its applicability and the whole number unit is utilized in the Color Map count and is particularly comforting to realize that the value can be zero ($\mathbb{N}_0$).
Natural Number	All positive integers (excluding zero) are defined as natural numbers in this implementation. The worth of a unit is in its applicability and the natural number unit is utilized in the Table controls to specifically indicate that the row and column indices begin at one (and not zero). A Natural Number may also be referred to as a counting number and sometimes a natural number includes zero, but not in this implementation. As such the set theoretic double-struck N is subscripted with a one to make it explicit that natural numbers begin at one ($\mathbb{N}_1$).

What makes a dimension and the set of units within that dimension clear is the fact that each unit [value] can be mapped into another unit [value] without a model specifying a different dimension (unit analysis). This is no clearer than with the Categorical dimension where the relationship between units within the categorical dimension is a simple matter of perception rather than transformation. For example, the ordinal unit is often mapped to integers as a matter of convenience to make the sequence index of the categorical value explicit. This is mere trickery as it is purely conceptual.

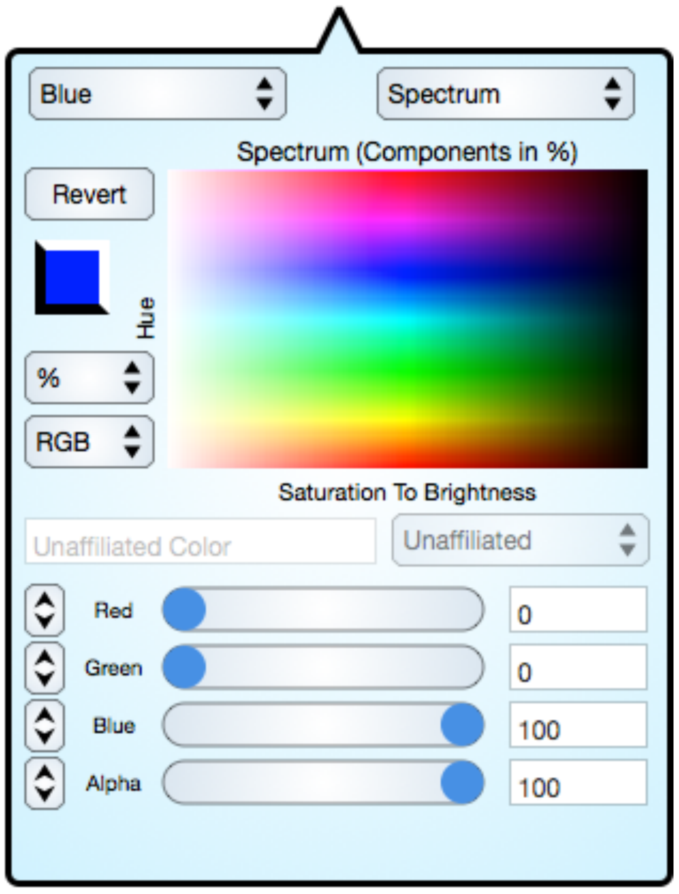
Graph IDE ► Controls ► Color Selector

The Color Selector controls a color. Bring the Color Selector forward by selecting a color well, for example from the Graphics inspector editor. The color well is shown as a solid-fill rectangle or in the case of a non-opaque color a rectangular region with a black and white background to contrast the transparent aspect of the color.

The following figure shows a typical color well with a maroon SVG indexed color. The index is 79 and is shown in the color well to indicate that the color is indexed. The cursor selector shows the color name, palette name and index name; or color space name and components if the color is not indexed.



The figures below shows the Color Selector panes. Choose one of the panes by using the pop up button in the upper right of the selector.



Spectrum Pane

To change the color select a color on the Spectrum color map or adjust the individual components using the component sliders, steppers or text fields. Alternatively, choose one of the common colors provided for in the Simple Name pop-up-button.

Simple Name : The pop up button in the upper left is called the Simple Names and shows and sets the color to one of the simple color values.

Color Map : Shows the color spectrum. Select a location on it to set the color.

Revert : Select to revert to the color that was loaded when the color selector came forward.

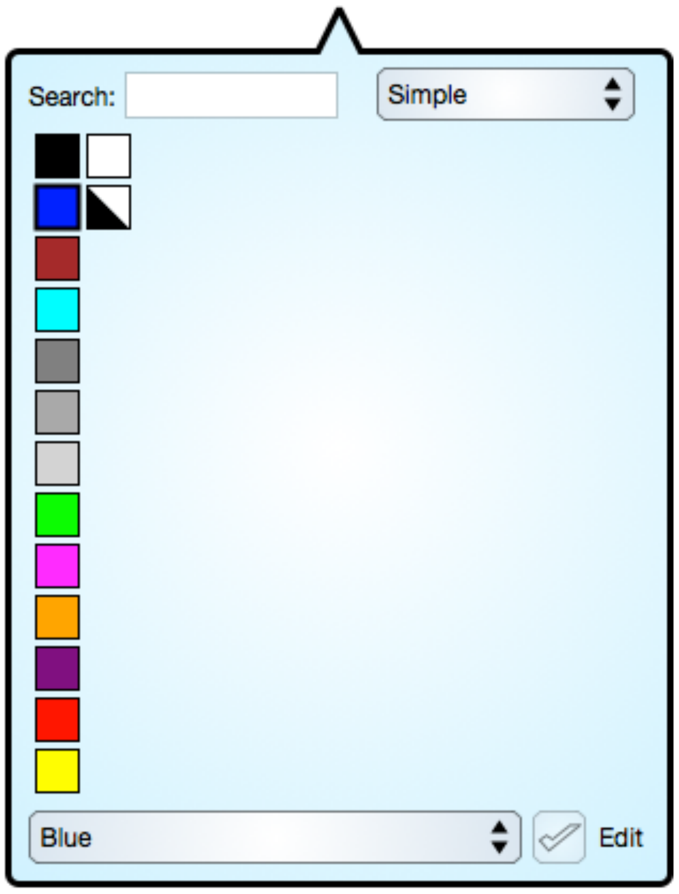
Color Well : A simple graphical representation of the current color on a black and white triangular background. The background is black and white so that the alpha effect of the color can be shown when alpha is less than 100 percent.

Depiction : Shows and sets the depiction of the color which is percent, unitized, hex or 255. Percent is a value between 0 to 100, Unitized a value between 0 to 1, hex a value between 0 and 255 in hexadecimal base, and 255 a value between 0 and 255 in decimal base.

Color Space : Shows and sets the color space to Gray, RGB or CMYK.

Named Color : Shows the name of the color as defined by the color palette. When the color palette is being edited then this name is editable, otherwise it simply shows the name if any and associated palette.

Component Controls : Modify the setting of the slider, stepper or text field to alter the color component values which are one of {Intensity}, {Red, Green, Blue, Alpha} or {Cyan, Magenta, Yellow, Black, Alpha} depending upon the color space set.



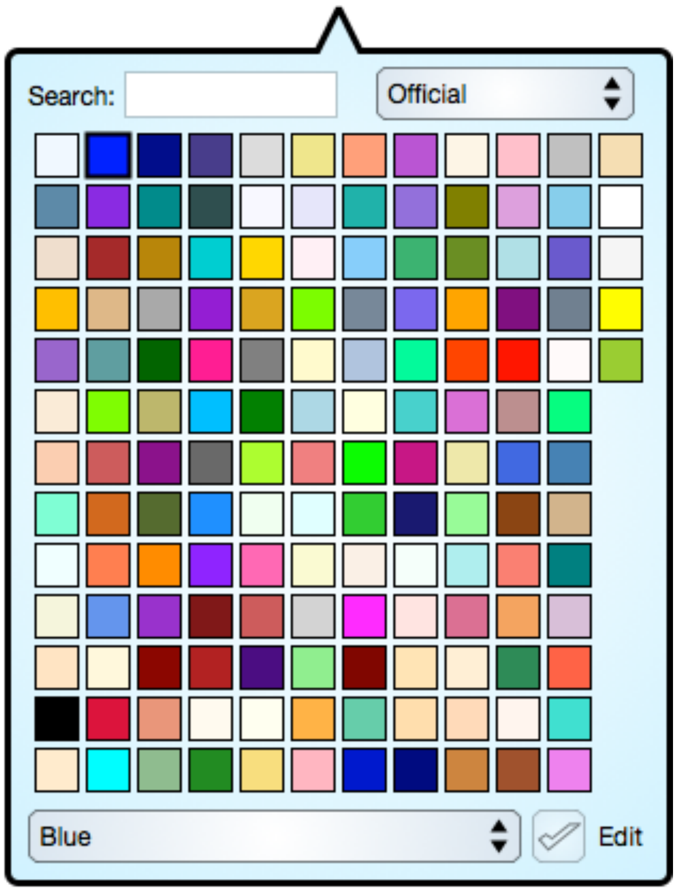
Simple Pane

Select a color well to choose and set the color. These are the same colors as in the Simple Name pop up button on the Spectrum Control panel except when a color is chosen on a palette then it is an indexed color.

Search : Type in a search fragment to limit the color wells shown. For example, type "red" (without quotes) to see all the red-named color wells. The search characters are associated with the color name and not the hue.

Name Pop Up : Choose the Name Pop Up (lower left) to select a color by name. If the search field is used then the list of names is limited by the search parameter. Clear the search field to see all the names in the resulting pop up.

Edit : Always disabled to indicate that only the user specified palette is editable.



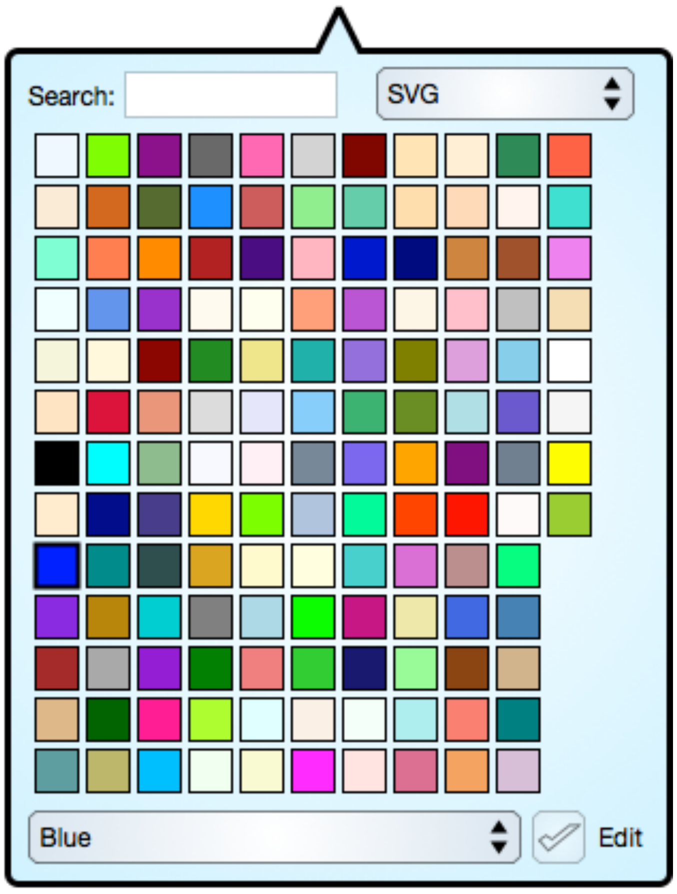
Official Pane

Select a color well to choose and set the color. These are official indexed colors and can not be edited.

Search : Type in a search fragment to limit the color wells shown. For example, type "red" (without quotes) to see all the red-named color wells such as "Pale Violet Red". The search characters are associated with the color name and not the hue.

Name Pop Up : Choose the Name Pop Up (lower left) to select a color by name. If the search field is used then the list of names is limited by the search parameter. Clear the search field to see all the names in the resulting pop up.

Edit : Always disabled to indicate that only the user specified palette is editable.



SVG Pane

Select a color well to choose and set the color. These are W3C SVG named colors and when the document is exported in SVG representation then the color is represented by the name only. The advantage to using named colors is that the color space, depiction and any discretization is not present and can be interpreted by the application that imports the SVG representation.

Search : Type in a search fragment to limit the color wells shown. For example, type "red" (without quotes) to see all the red-named color wells such as "Pale Violet Red". The search characters are associated with the color name and not the hue.

Name Pop Up : Choose the Name Pop Up (lower left) to select a color by name. If the search field is used then the list of names is limited by the search parameter. Clear the search field to see all the names in the resulting pop up.

Edit : Always disabled to indicate that only the user specified palette is editable.

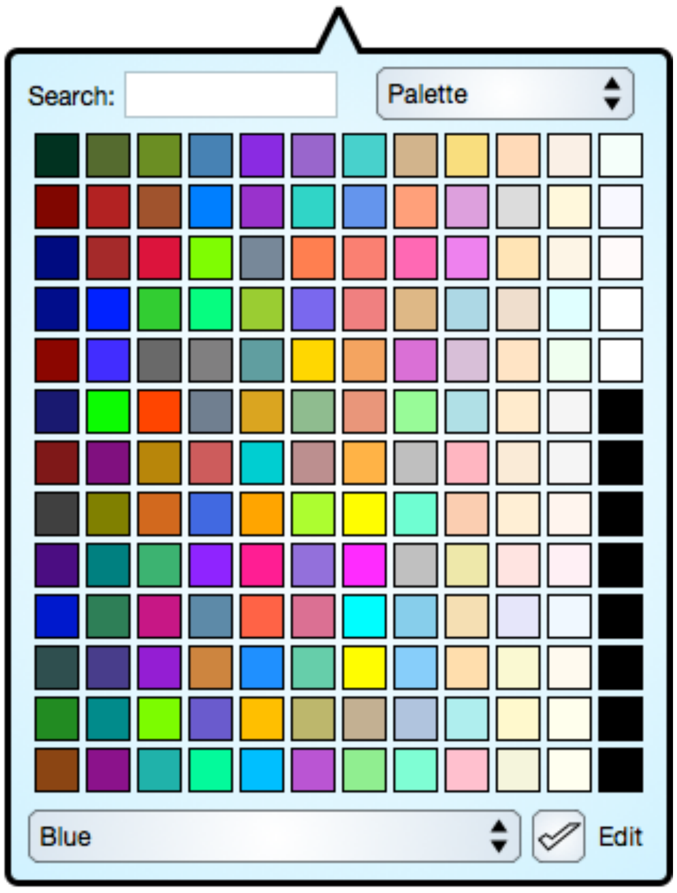
Palette Pane

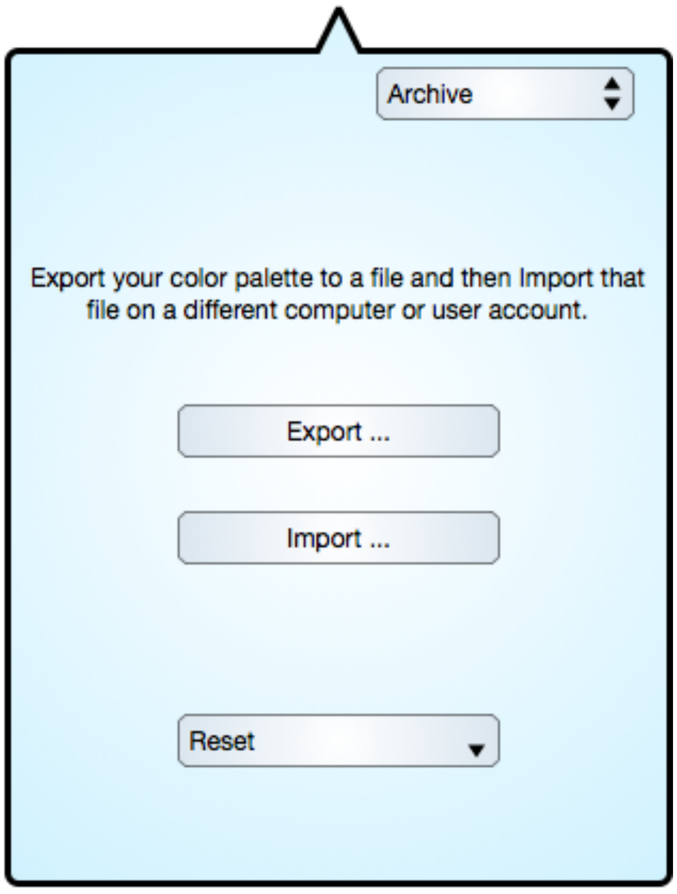
Select a color well to choose and set the color. The default palette is 144 named colors called "Intensity Ordered Official" and are ordered according to intensity of the color. Other initial colors are described below in the Archive section.

Search : Type in a search fragment to limit the color wells shown. For example, type "red" (without quotes) to see all the red-named color wells such as "Pale Violet Red". The search characters are associated with the color name and not the hue.

Name Pop Up : Choose the Name Pop Up (lower left) to select a color by name. If the search field is used then the list of names is limited by the search parameter. Clear the search field to see all the names in the resulting pop up.

Edit : When selected then each color well in the palette is editable instead of selected. The edit control is the Spectrum Pane as described above.





Export your color palette to a file and then Import that file on a different computer or user account.

Archive Pane

Making a palette can be time consuming. Once made, the values are permanent. Those values can be exported and imported to different computers as needed.

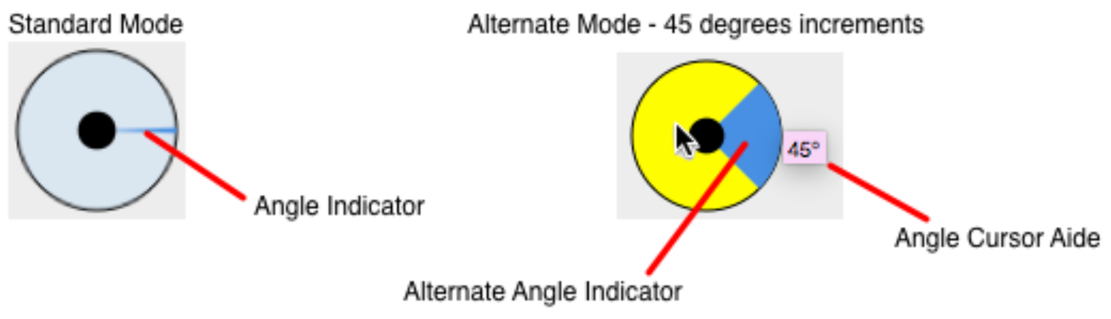
Export... : Select to export the palette to a file.

Import... : Select to import the palette from a file.

Reset : Select to reset the palette to one of Intensity Ordered Official, Name Ordered Official, Simple, or Black. The Black reset is useful to first clear all colors in the palette and then manually assign them to custom values.

Graph IDE ▶ Controls ▶ Dial

The dial controls an angle. The figure below shows a dial on a window background.


Standard Mode
Angle Indicator
Alternate Mode - 45 degrees increments
45°
Angle Cursor Aide
Alternate Angle Indicator

To operate a dial mouse down over the dial and drag the mouse. The angle indicator will rotate to follow the cursor and the angle value will show in the Cursor Aide. If you press the alternate key then the dial indicator moves in 45 degrees increments. If you press the control key then the dial indicator moves in 5 degree increments. By moving the cursor way from the dial center, while the mouse is down and the dial is active, you gain more angle resolution because the "lever" is longer so you must move the cursor more to affect a change in the angle. Thus, while the dial is being used, it is perfectly valid to move the cursor as far away from the dial as needed, even outside the control's display area.

Key Modifier	Mouse Drag Description
None	Rotates by one degree
Alt	Rotates by 45 degrees
Control	Rotates by five degrees
Shift-Control	Rotates by ten degrees

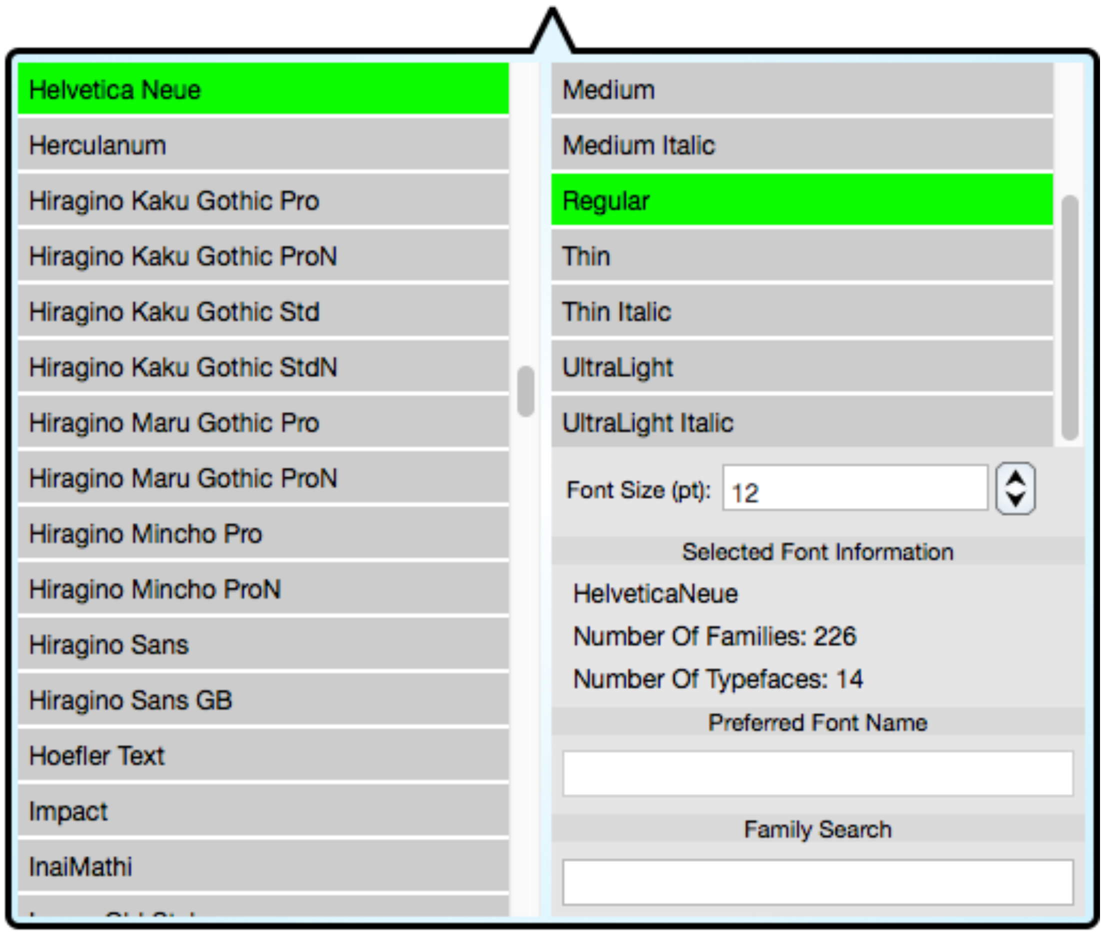
If you mouse click the dial then that dial is active and then you can use the arrow keys to affect a change in the angle. Left arrow rotates clockwise by one degree, right arrow rotates counter-clockwise by one degree, up arrow rotates counter-clockwise by five degrees, down arrow rotates clockwise by five degrees. If you depress the shift-key then that multiplies the resolution by five, while the alt-key multiplies the resolution by two.

For more control, a dial is often accompanied by a stepper control which brings forward a Number Selector. The number selector can be used to enter any angle in degrees.

The dial is used to specify angles for rotation in the Graphics and 3D Graph inspector editors, wedge angle in the Circle inspector editor and angles in other inspector editors. The angle unit is always in degrees.

Graph IDE ► Controls ► Font Selector

The Font Selector controls a font type and size. Bring the Font Selector forward by selecting a font well, for example from the Label inspector editor. The font well does not look like a well as it is simply a textual label indicating the font name and size and usually has grey text which makes it look like a non-control label. Nonetheless, selecting it will bring forward the Font Selector. The figure below shows a Font Selector.



Font Size (pt): 12

Selected Font Information

HelveticaNeue

Number Of Families: 226

Number Of Typefaces: 14

Font Family

The left scroll area shows all of the font families available on the system. In the Web Edition this shows the HTML available fonts and not the fonts of the cloud system.

Typeface

The right scroll area shows all typefaces for the selected family.

Controls

Font Size : Shows and sets the font size.

Font Information : Shows the selected font name, the number of available families and the number of typefaces for the currently selected font.

Preferred Font Name

Preferred Font Name : Enter text to define the preferred font name. If this is blank (which is usual) then the preferred font name is that of the selected font. This is required for various environments that do not have the same fonts available.

Family Search

Family Search : Enter text to search for a particular family name and limit the entries in the family scroll view. This is particularly important as there can be hundreds of families on a system.

To change the font select a family and typeface name and then font size.

Note that editors like the Graph Titles And Labels editor has multiple text wells, one below each title text field (for the main, x-axis title and y-axis title) and also one that specifies the label fonts.

Font Panel

This section applies to the Mac only.

To effect a change in a font first select the graphic and then bring forward the font panel (command-t) and change the font. If a graphic does not have a font then no action is performed. If the graphic is a Label then the font for the entire label is changed. If the graphic is a Data Graphic then the font change is usually associated with a Point Tag label font. If the graphic is part of a Network then the font change applies to the node label.

If the graphic is a Graph then the font changed depends on the component of the graph hit. For a graph there are three component hit types:

Title:	If the main, x or y title are hit then a font change is associated with that title only.
Labels:	If an axis label is hit then the font for all the labels of that axis are changed.
Frame:	If the hit is in the interior of the graph frame then the labels for all axes are changed.

This way of changing fonts depends implicitly upon the component hit. Sometimes it is easier to use the inspector editor font well because that is an explicit context.

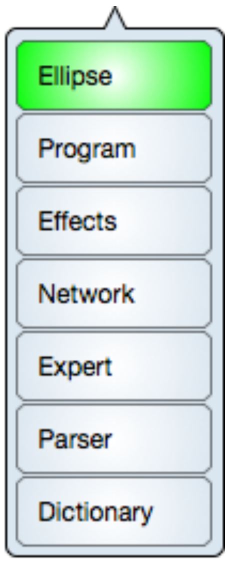
Graph IDE ► Controls ► Menu Selector

A Menu Selector is a control that is used to select one element of a list. Selecting a pop up button or pull down button brings forward the Menu Selector. Make a selection by selecting one of the elements in the enumerated list shown on the Menu Selector.

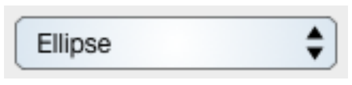
Pop up and pull down buttons appear on many of the inspector editors, for example the Graphics editor has several and they are identified by the right justified arrows on the button.

Pop Up Menu Selector

The Pop Up Menu Selector shows a list of options where one element in the list is the currently selected element.



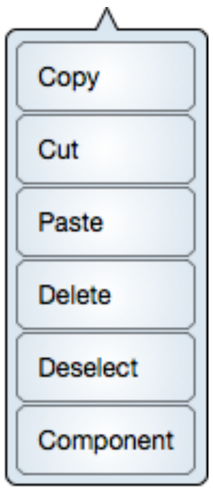
The Pop Up Menu Selector is brought forward by selecting a pop up button such as the one shown here:



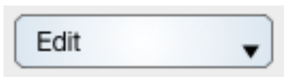
Select one of the options to make that selection current and dismiss the selector. The current option is highlighted green.

Pull Down Menu Selector

The Pull Down Menu Selector shows a list of options. Choosing one of those options causes the option to be executed but not selected.



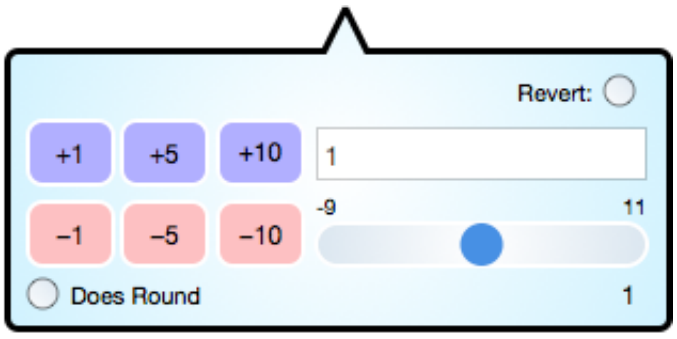
The Pull Down Menu Selector is brought forward by selecting a pull down button such as the one shown here:



Select one of the options causes that option to be executed. Since there is no selected option the title of the pull down button shows the category of possible options.

A menu selector may also appear in other locations without an apparent control such as for a context menu.

The Number Selector is a control that adjusts the value of a single number (scalar) and is shown below.

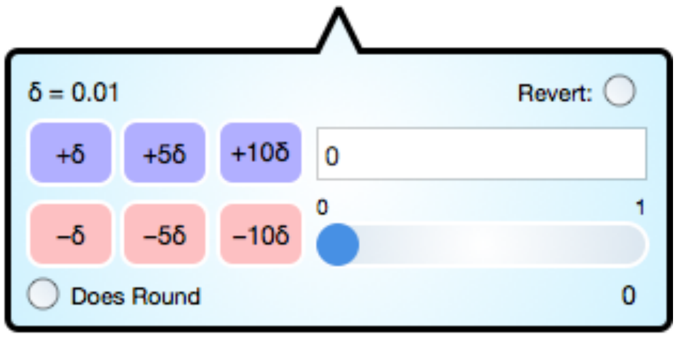


Selecting a Stepper brings forward the Number Selector. The buttons increment or decrement the number by values of 1, 5 or 10 respectively as shown on the button. The number can be typed into the text field or adjusted via the slider. Other buttons set options to round the number or revert to the number that was used when the selector first came forward.

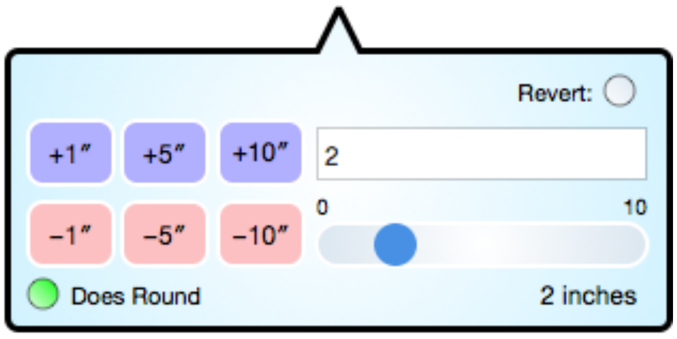
Steppers appear on many of the inspector editors, for example the Graphics editor has one near the stroke width text field entry as well as many other places.

On rare occasion, the Number Selector is placed in differential mode meaning that the scalar represents an increment instead of a absolute value. Currently, the only place that happens is for the dash pattern. In addition, the Number Selector can be set to implement heuristics such as keeping the number positive or within a certain interval. For example, the stroke width can never be less than zero. Those types of logic are implemented implicitly and there is no indication within the user interface unless the number is typed (meaning it has a unit).

If the increment is non-unit (not one) then that delta increment appears above the buttons and each button has a greek delta symbol to indicate as such, for example in the figure below.



If the number being adjusted has a unit then the unit symbol (if any) appears on each button and the unit name appears in the value label at the bottom right of the selector.



As much as possible, all controls also have units such as points, degrees, etc. and each graph may have a user-specified unit which will show up on the Number Selector.

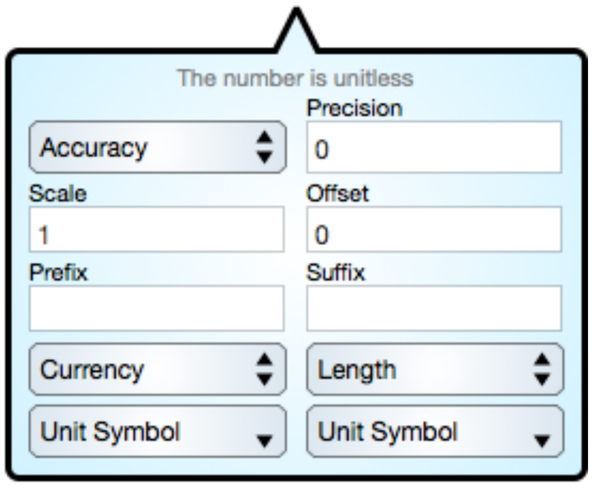
Graph IDE ► Controls ► Format Selectors

Format Selectors are used to reformat the display of text. It is important to note that the original text to be formatted is preserved by the formatter and that the formatter is used simply to display the original text. Because there are two different textual representations there is then a "domain" and "range" text where the formatter is the mapping function, the domain is the original text and the range is the displayed output of the formatter.

These formatters are used by the Spreadsheet.

Number Format Selector

The number format selector determines text formatting for decimal representation of a number.



Type : One of Accuracy, Fixed or Scientific.

Precision : The number of digits after the decimal point.

Scale, Offset : The linear mapping of the output (r = d *scale + offset). Often, the scale represents a unit transformation such as Euro to USD.

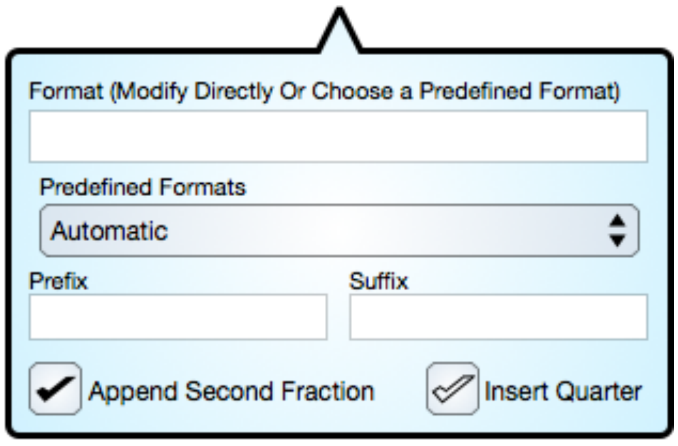
Prefix : Prefix to the domain.

Suffix : Suffix to the domain.

Currency : A pull down to select common currency character to be inserted into the Prefix.

Date Format Selector

The date format selector determines Gregorian date formatting for a Gregorian date. It is important to note that this formatter reformats its own type. For example, the Date Axis formats julian day fractions to Gregorian date while this formatter maps a Gregorian date to another Gregorian date representation.



Format : The Gregorian date format for the range string.

Usual Formats : A pull down that populates the other fields with common formatting parameters.

Prefix : Prefix to the domain.

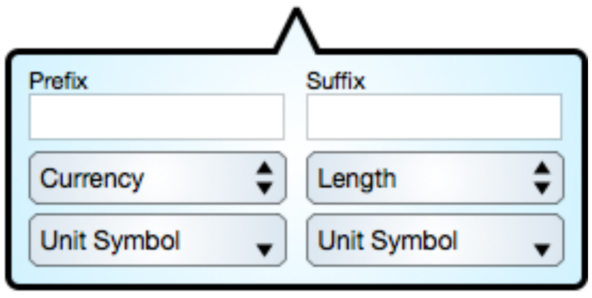
Suffix : Suffix to the domain.

Append Second Fraction : If selected then the fractional second is appended to the range string.

Insert Quarter : Inserts the quarter digit right after the prefix.

Text Format Selector

The text format selector determines textual formatting for text. Note that the domain and range strings are of the same type.



Prefix : Prefix to the domain.

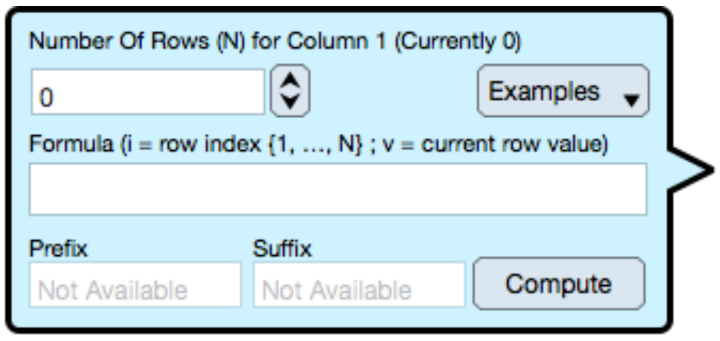
Suffix : Suffix to the domain.

The Formula Selector computes a sequence of values for a Spreadsheet and Tables column. It is important to note that a formula is a quick way to populate a column of values and that once populated then each table cell entry can be further modified.

For making more general algorithms see Programming.

Formula Selector

The Formula Selector is brought forward by selecting the "Formula" entry to the Component drop down. The Component drop down is brought forward by right-mouse-click or select-hold on a table column header or by using the Component drop down menu near the table. The Formula Selector is annotated below.


Number Of Rows (N) for Column 1 (Currently 0)
0
Examples
Formula (i = row index {1, ..., N} ; v = current row value)
Prefix
Suffix
Not Available
Not Available
Compute

Number Of Rows : The number of times that the formula is computed. Upon each execution the domain index ('i' which represents the row index of a column in a table) is incremented by one.

Examples : Gives a menu of possible formula entries. Select a menu item to load that entry. The last entry in the menu is "Revert to formatter settings" and if selected loads the formula last stored in a Spreadsheet column.

Formula : A formula that uses the symbol 'i' as its domain and utilizes the Programming facilities to compute new range values. The current range of the row is represented by the symbol 'v'. A formula is a single statement program.

Prefix : The prefix to the range of the formula. If the table column is numeric only type then the prefix is not available.

Suffix : The suffix to the range of the formula. If the table column is numeric only type then the suffix is not available.

Compute : Once all parameters have been entered then select the Compute button to perform the formula mapping and enter all values into the selected table column.

Formula Examples

Example formulas are shown in the table below. For language syntax and predefined function declarations see Language.

Formula	Description
i	Fill with row index 1 to 'N'. Notice that the domain variable begins at 1 and increments by 1 to the value of 'N'
-v	Reverse the sign of the current values. This is short for -1 * v.
2 * v	Double current values. Any algebraic formula can be used so that the formula can be quite complex. Use the normal parenthesis to define precedence of operation.
floor(v * 100 + 0.5)/100	Round current values to two decimal places. floor() means make its argument a whole number by truncation. The function ceil() means make its argument a whole number by increasing to the next whole number. To be explicit, making a whole number from a real number has two operators instead of the normal one.
i % 2	Alternate between 1 and 0. The percentage symbol is the modulus operator.
1	Fill with the constant 1. Any constant number can be used.
sin((i - 1.0) * 0.1)	Fill with a modulated signal. The Language section list all available functions.
NAN	Invalidates all rows. Note that NAN must be all capitals to be interpreted as the "Not a Number" entry. This is in opposition to entering NaN into a table cell as that is case insensitive.

Note that the symbol 'v' is the current value of the table row and must be a numeric value in order for the formula to be valid. If the table column is not of a numeric type then 'v' can not be used.

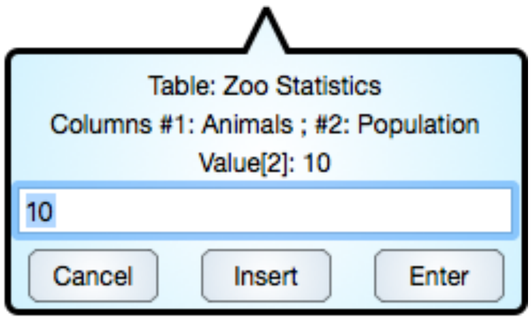
Here are a few ideas:

- If the Prefix is set to the word "Sample " (with a trailing space but with no quotes) and then 'i' (without the single quotes) is used as the formula then the column entries with be computed as "Sample 1", "Sample 2" ... "Sample N" which is appropriate for entries to a Legend for a graph.
- If 'i % 2' (without the single quotes) is used then the value alternates between 1 and 0 which may be appropriate for assigning values to the rgba components of segments of a Function. The constant '1' (without the single quotes) can be used to assign a solid alpha component to the color.
- Sometimes data is pasted into a Data Graphic and the coordinate is flipped, particularly the y-value coordinate. Applying '-v' (without the single quotes) flips the data values associated with that coordinate and table column. Sometimes data has two components (x and y) and both are sampled irregularly but it may be desirable to make one component regular (have regular intervals). Applying something like 'i', '100 * i' or some variation of that will make that coordinate values regular (uniform, of constant increment, et.al.).
- Formulas are very powerful, useful and specific. For more general programming see Programming. In particular, a program is a multi-statement algorithm (a sequence of functions) which can be animated (has an additional index symbol associated with animation step

a.k.a. time). When programs are combined with loading a Plugin (a shared object library) then data can be automated, fetched and animated from a variety of sources and not just an algorithm.

Graph IDE ► Controls ► Data Selector

The Data Selector is a control that shows and edits one piece of data, an example of which is shown below.



When a Spreadsheet is made active then all associated representations have a data selector. Selecting a component of a representation brings forward the data selector. The component is a wedge of a Pie Chart, bar in a Bar And Column Chart or point in a Function or Scatter representation.

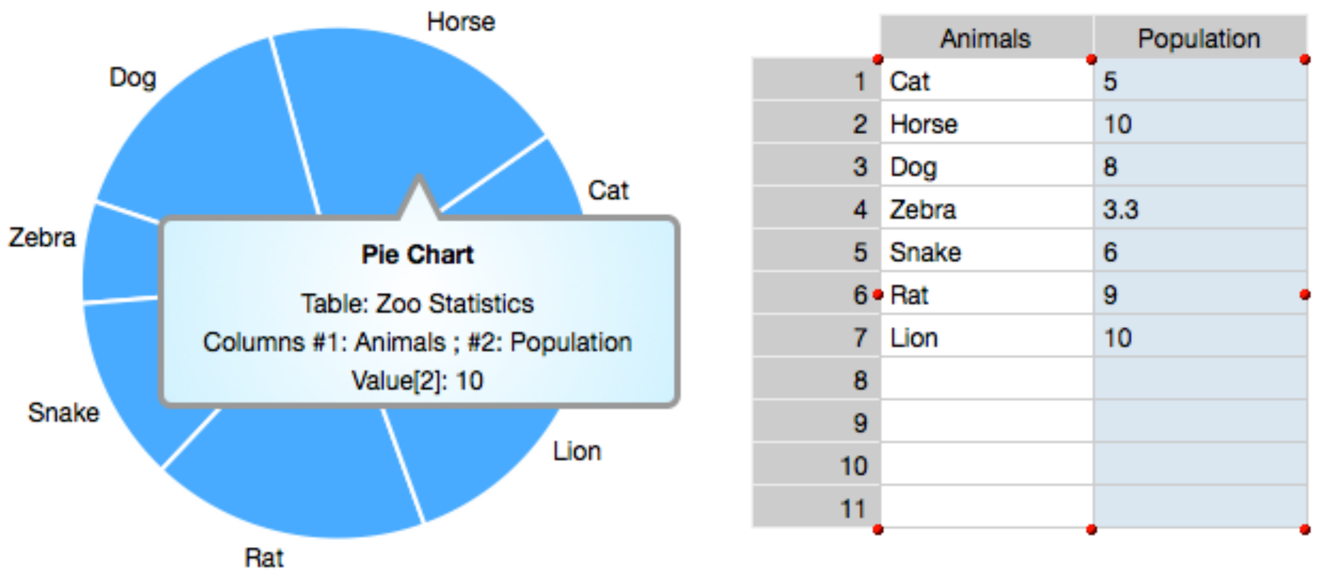
Entering a new data value changes the selected component of the representation and also the associated value in the spreadsheet. Selecting the Insert button will insert the value right after the currently selected component. The data selector can be cancelled by selecting the Cancel button or selecting outside the data selector region.

The Data Selector is also a feature of the Chart Tasks.

It should be noted that the data selector can be a useful feature, but that there are more robust and efficient ways to navigate to and edit data. For example, the Function Data inspector editor gives direct access to all the data and selecting a point or segment on the function (curve) will scroll the data table to the corresponding row so the point-and-click feature of the data selector is found other ways as well. Even so, the data selector does have advantages. For example, the data layer of a graph does not have to be focused on to directly edit the point of a curve.

The Information Selector is a control that shows information associated with a graphic, an example of which is shown below.

Pie chart on the left was made from columns 1 and 2 in the table on the right



Pie Chart

Table: Zoo Statistics

Columns #1: Animals ; #2: Population

Value[2]: 10

	Animals	Population
1	Cat	5
2	Horse	10
3	Dog	8
4	Zebra	3.3
5	Snake	6
6	Rat	9
7	Lion	10
8		
9		
10		
11		

Hovering over an enabled graphic brings forward the Information Selector.

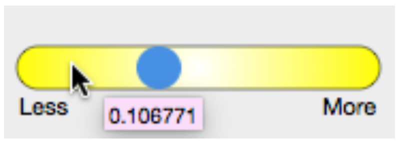
When a Spreadsheet is made active then all associated representations have an information selector. Tables implement an information selector that shows row and column number, data value or in the case of an empty cell a brief instruction. The Graphic Selector shows the factory type and the Navigator shows the instance type along with a brief instruction.

The Information Selector is also a feature of the Event Qualifier used while implementing a custom application. Also see Event Qualifier for additional information.

Since the Information Selector functionality is implemented using a hover state it only make sense on the Mac and Windows platform.

Graph IDE ► Controls ► Slider

A Slider is a control that is used to adjust a continuous scalar within a predefined interval and is shown below.



Adjust the associated scalar (a number) by selecting a point within the slider area. The scalar value will alter to a value proportional to the distance from the starting boundary of the slider.

The slider current value is indicated by the knob (the blue circle) on the slider. The active region of the slider is the entire slider because if any region of the slider is selected then the knob will jump to that selected location. Hence, the knob does not need to be selected to alter the scalar value.

Sliders appear on many of the inspector editors, for example the Graphics editor has several sliders. Sliders on Tables scroll the table. Sliders often work in conjunction with text fields and the Number Selector which all are different ways of defining a scalar value.

Graph IDE ► Controls ► Standard Editing

Overview

The main purpose of Graph IDE is to aid in the construction of new graphic-based documents. In order to do that it implements a few standard editing facilities. In addition, each graphic type may have editing facilities unique only to itself.

What follows is a description of each editing type.

Definitions

- **Non-Modal Mouse Editing**: A non-modal mouse editing control starts editing upon a mouse action, such as pushing down on the mouse button and then stops editing immediately after another action, such as when you release the mouse button.
- **Modal Mouse Editing**: A modal mouse editing control starts on a special mouse or keyboard event and only stops editing only upon a special sequence of actions, usually a mouse double-click.

Creation

- Creation is always non-modal. To create a new graphic bring forward the Graphic Selector, click the appropriate factory cell and then mouse down on a Graphic View and drag the cursor to another point and then release the mouse button. The initial and final cursor locations define a reference rectangle of the resulting graphic. Copies of existing graphics can also be made using the Palette menu by dragging a graphic from a palette to the Graphic View of your document and releasing the mouse button at the desired location.
- After making a new graphic it is rarely set to values that you want, so you next have to edit the graphic to the attribute values that you require.

Selecting Graphics

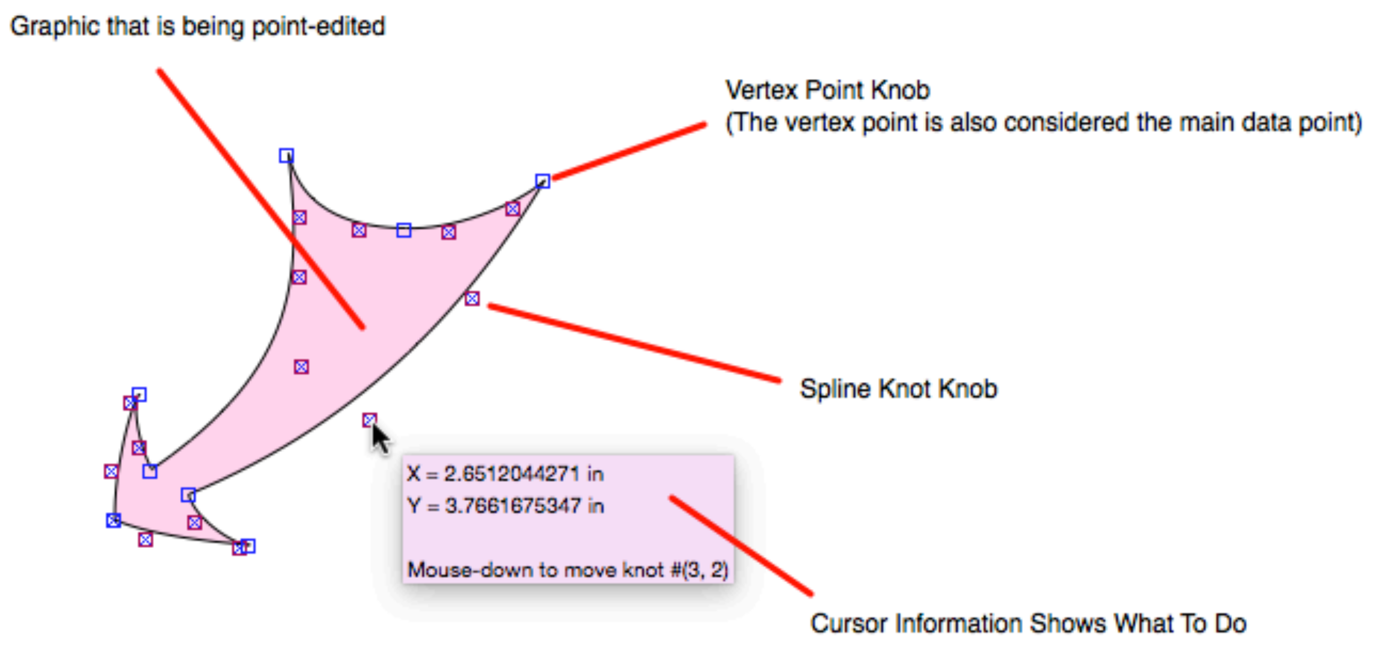
- To select a graphic first make sure the Graphic Selector is in selection mode and then move the cursor over the graphic and click once to select. To select a different graphic do the same thing over that graphic. To deselect click once over no graphics. To select multiple graphics depress the shift key and click once over each. To deselect one graphic from many selected graphics move the cursor over the selected graphic and with the shift key down click once.
- To select a rectangular region of graphics mouse down while not over any graphic and drag the mouse to the desire end point of the rectangular region and release. To add more graphics from another rectangular region repeat the drag, but this time keep the shift key depressed.

Moving And Resizing A Graphic

- To resize a graphic first select it and then mouse down on one of the corner or edge knobs, drag the mouse to the desired location and then release the mouse button. To constrain the resizing to constant aspect ratio move a corner knob while depressing the Alternate Key.
- To move a graphic or group of graphics first select the graphics, then press the mouse button while over any of the selected graphics, move the mouse and finally release the mouse button while at the desired location.

Point Editing

- Graphics that have points as a constitutive parameter can be mouse edited one point at a time. Those graphics include the Polygon, Cubic Bezier, Function, Trajectory and Scatter graphics.
- To point-wise edit a graphic using the mouse first depress the 'e' key ('e' for Edit), then move the cursor over the graphic to edit as if to select it and finally click the mouse button. You are now in the point-wise edit mode and can only exit it by double clicking the mouse button. While in point-wise edit mode the point vertex and spline knots, if any, are displayed by knobs, such as that shown below.


Graphic that is being point-edited
Vertex Point Knob
(The vertex point is also considered the main data point)
Spline Knot Knob
X = 2.6512044271 in
Y = 3.7661675347 in
Mouse-down to move knot #(3, 2)
Cursor Information Shows What To Do

- To move a vertex or spline knot move the cursor over it and drag it. To remove it depress the shift key and click once. To add a vertex (and spline knot pair if appropriate) move the cursor to the curve (boundary of the graphic and while not over an existing knob depress the shift key and click once.
- The cursor information panel displays the vertex or spline knot sequence number as well as the coordinate values of the cursor. If you need finer resolution while editing then first magnify the graphic view and then enter the point editing mouse mode.

Inspector Editor Editing

- Most main-inspector-editors have standard input fields, such as origin and size values. For additional information consult Inspector Editors or the particular graphic in question for controls specific to that graphic.

At this point, you should have a decent understanding of Graph IDE. I would look at one of the Basic Graphics and make one, such as a Circle or skip ahead and read how to add data to a graph in the section Getting Data On A Graph.

Graph IDE ► Tables

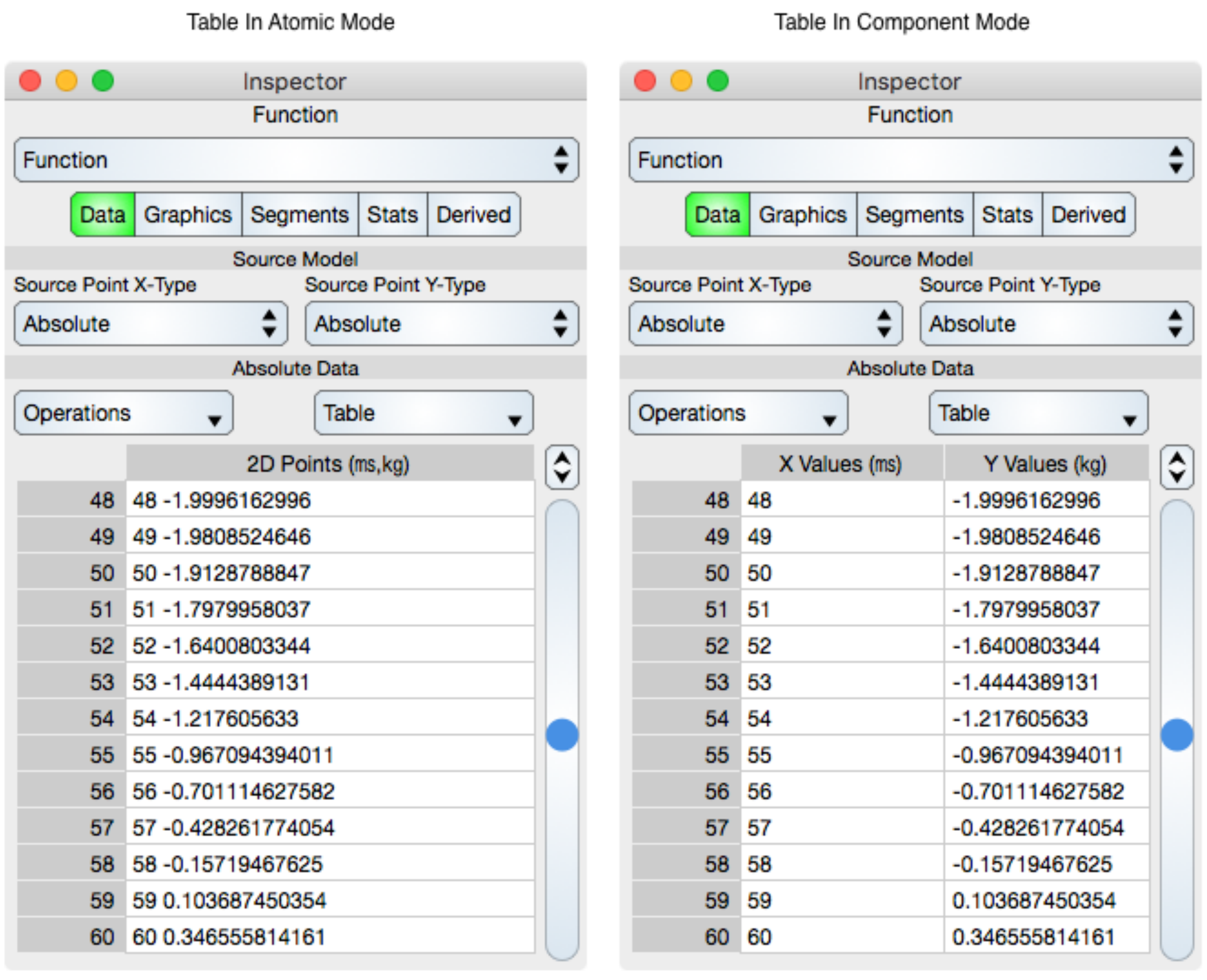
A table is used to display and alter sequences of data in textual representations, typically numbers or text but can also be two and three dimensional points.

Note: Spreadsheets are user-definable tables. For information on spreadsheets see: Spreadsheet.

Section	Description
Overview	A general explanation of a table.
Export And Import	Describes the mechanisms to export and import data from and to a table.

Graph IDE ► Tables ► Overview

Tables are used to show and edit numeric data in textual format and often show up in a data-oriented Inspector Editor and are utilized by a Spreadsheet. Table cells can represent many data types. For example, a table cell can represent a color in RGBA (red, green, blue and alpha) numeric representation, or a pair of scalars representing a 2D point. That representation is called atomic mode. A cell can also represent one component of a composite atomic. For example, a 2D point can be represented as the X Value in column one and the Y Value in column two. The figure below shows the two different modes for a Function graphic.



Function

Source Model

Source Point X-Type: Absolute

Source Point Y-Type: Absolute

Absolute Data

	2D Points (ms,kg)
48	48 -1.9996162996
49	49 -1.9808524646
50	50 -1.9128788847
51	51 -1.7979958037
52	52 -1.6400803344
53	53 -1.4444389131
54	54 -1.217605633
55	55 -0.967094394011
56	56 -0.701114627582
57	57 -0.428261774054
58	58 -0.15719467625
59	59 0.103687450354
60	60 0.346555814161

	X Values (ms)	Y Values (kg)
48	48	-1.9996162996
49	49	-1.9808524646
50	50	-1.9128788847
51	51	-1.7979958037
52	52	-1.6400803344
53	53	-1.4444389131
54	54	-1.217605633
55	55	-0.967094394011
56	56	-0.701114627582
57	57	-0.428261774054
58	58	-0.15719467625
59	59	0.103687450354
60	60	0.346555814161

Click-Hold Or Right-Click To Set Mode

To alternate between the modes, click-hold or right click on the table and choose the mode or use the table component drop-down menu (to the right above the table).

The following is an itemization of table features.

Component Selection

The table component is a column, row, single cell, column header, row header or entire table. The following lists operations on such components.

- Click on a column header to select a column. Shift-click twice to select an interval of columns. A single click on a column selects only that column for use and hence resulting operations are bounded to that column. A shift-click selects intervals of columns and hence resulting operations span columns.
- Click on a row header to select a row. Shift-click twice to select an interval of rows. A single click on a row selects only that row for use and hence resulting operations are bounded to that row. A shift-click selects intervals of rows and hence resulting operations span rows.
- Click away from the table to deselect the table. Make sure the click is near the table, but not upon it, so that no other element is selected.
- Click on a row or column header and then type command-a to select the entire table.
- Click-drag on the cells to select a group of cells and to scroll the table.
- Click-hold to select a single cell and also to bring up a menu of options. Click again to dismiss the options menu while keeping the cell selected.
- Many times, a data graphic can be used to focus on a table component. For example, clicking on a Function graphic on either that

graphic's vertex or line segment will select the corresponding row in the table.

Operations On Component Selection

Once a component is selected then choose one of the following:

- Delete (the delete key or del numeric keypad key) to delete the selected component.
- Copy (command-c, Copy main menu item, or select-hold menu) to copy the selected component.
- Paste (command-v, Paste main menu item, or select-hold menu) to paste the selected component.
- Cut (command-x, Cut main menu item, or select-hold menu) to cut the selected component.
- Select-hold on a selection to bring up the Component drop-down menu of operations.
- Use the table component drop-down menu (to the right above the table) as another means to show operations upon a selected component.
- If a column is selected then the Formula Selector is a good way to generate data based upon a formula for all rows of that column.

The Component drop-down menu is brought forward by right-click, select-hold on that component or by using the Component drop-down menu that is normally positioned at the top-right of the table. That drop-down menu has entries for Delete, Copy, Paste, Cut, Formula (brings forward the Formula Selector), etc.

If you select a row or column and that row or column header shows an arrow then the row or column can be swapped with the adjacent one using the arrow keys on the keyboard. For a Spreadsheet the swap swaps data but does not swap the associations with representations.

It should be noted that if you select only one column or row of data (with a click) then that column or row is extended with a subsequent paste. However, if you shift-click an interval of columns or rows then the subsequent paste wraps over the rectangular cell selection.

Cell Edit

When you click on a cell and then release, without dragging or holding, then the cell editor is brought forward. Once forward the following applies.

- Edit the cell text using the normal keyboard edit facilities.
- Click Return to enter the data, tab to enter and proceed to the next row in the column or shift-tab to enter and proceed to the previous row in the column.
- Use the arrow keys to enter the text and proceed to an adjacent cell in the direction of the arrow.
- Click the ESC or command-. to cancel cell editing and dismiss the cell editor.
- Use the on-board buttons to enter the data and dismiss, revert or proceed to adjacent cells.

Information Selector

- Move the cursor over a cell to see the information selector. The table information selector shows the cell indices and value. In the case of a blank cell it shows the cell data entry instruction.

Data Format

The data format for each import operation is define in the respective inspector editor section. Generally:

- For scalar data the column import (paste) format is a list of numbers, for point values it is a list of point components (for example, for 2D points: x1 y1 x2 y2 ... xN yN) where numbers are separated by a blank.
- When importing into numeric columns the format can be much more liberal. Any non-numeric ASCII delimiter can be used such as comma, semicolon, space, tab, Return, etc.
- When importing into a label row the delimiter is a Return character.
- When importing into a 2D point column or cell the format can be a x y numeric pair or a date number pair. The date is formatted as: MM/DD/YY HH:MM:SS.fraction and a numeric number representing the y-value follows.
- If you paste to the entire table then the table import sheet comes forward because pasted data is formatted without explicit delimiters and you need to supply additional information as follows.
- Choosing the double-return delimited format defines column ends as two consecutive return characters in the string serialization of the data.
- Under some circumstances, you can explicitly define the table dimensions and other data attributes.
- If the data dimensions is symmetric then you can transpose the data as needed. Non-symmetrical data can not transpose by inherent limitation.

Graph IDE ► Tables ► Export And Import

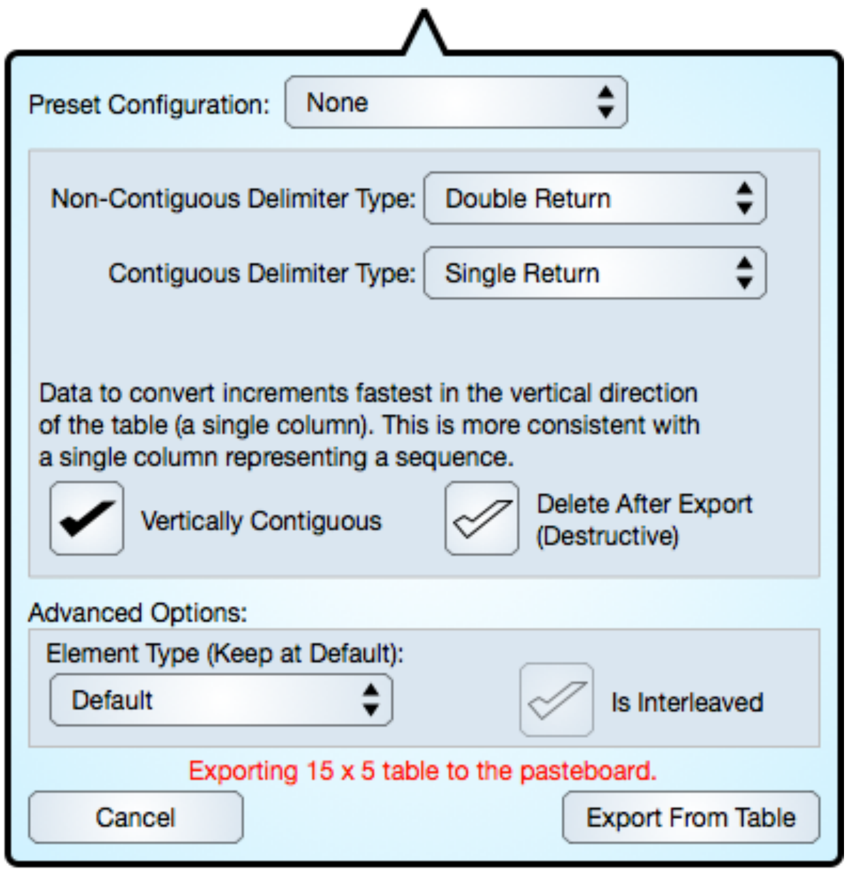
Importing and exporting data to and from tables is a matter of converting a serialized textual representation into discrete cell values and vice-versa. Those cells can be numbers, general text, pairs of numbers or sets of numbers. If importing to a cell with one value then importing and exporting is easy as it is a one-to-one relationship. However, if multiple cells are involved then the serialized textual representation must include inlined delimiters. If multiple rows and columns are specified then the serialization contiguous dimension must be specified. Historically, there are many heuristics involved during import and export operations. What follows is an explanation of this process from the perspective of the Import and Export Selectors.

Before proceeding it is important to clarify that vertical means with the row index running fastest, that is within one column. Saying row-contiguous does not make sense because that could mean the row-index varies fastest or the data is contiguous within one row. People that use row-contiguous nomenclature artificially agree upon one way of thinking about it and if everyone knows what the agreement is then it is OK. In this text things like row-contiguous is avoided and instead vertical-contiguous or single-column is used.

Historically speaking, exporting and importing is to and from ASCII files of data; especially for CSV and spreadsheet data. For this reason, the serialization is assumed to be UTF-8 encoded because that encompasses traditional character sets in data files. That means that when you work with UNICODE data then it must be in UTF-8 encoding.

Export Selector

The Export Selector is encountered while attempting to export data to a file as well as a pasteboard (or other service) string when the delimiter specifications are not implicit. The Export Selector is described below.


Preset Configuration: None
Non-Contiguous Delimiter Type: Double Return
Contiguous Delimiter Type: Single Return
Data to convert increments fastest in the vertical direction of the table (a single column). This is more consistent with a single column representing a sequence.
Vertically Contiguous: ✓
Delete After Export (Destructive)
Advanced Options:
Element Type (Keep at Default): Default
Is Interleaved
Exporting 15 x 5 table to the pasteboard.

Preset Configuration

One of None, Spreadsheet, CSV or Robust. If None then then the other parameters must be manually set. If Spreadsheet then parameters are set consistent to a spreadsheet and if CSV then consistent with comma separated values data. If Robust then the parameters are set to vertically contiguous which is more robust because each column has elements (cells) of the same type and the underlying data structure is contiguous in the vertical direction.

Main Parameters

Non-Contiguous Delimiter Type : For a spreadsheet or CSV this is a single return character and for robust it is a two return characters.

Contiguous Delimiter Type : One of White Space, Blank, Tab, Comma or Single Return. For a spreadsheet this is a tab character, for CSV a comma and for robust it is a single return character.

Vertically Contiguous : If on then the data runs fastest along a column (vertically) otherwise along a row. Since a column often represents a curve or other sequence of data the most effective and efficient ordering is vertically contiguous even though that is the opposite of a spreadsheet.

Delete After Export : If on then the data in the table is irretrievably deleted. This option is mostly encountered when the data is exported using a cut operation.

Advanced Options

Do not alter these parameters.

Perform Operation

Information Text : This specifies what is being exported.

Cancel : Select this to cancel the operation. Selecting a region away from the Export Selector also cancels the operation.

Export From Table : Select this to perform the export and also dismiss the selector.

Import Selector

The Import Selector is encountered while attempting to import from a file as well as a pasteboard (or other service) string when the delimiter specifications are not implicit. The Import Selector is described below.

Preset Configuration

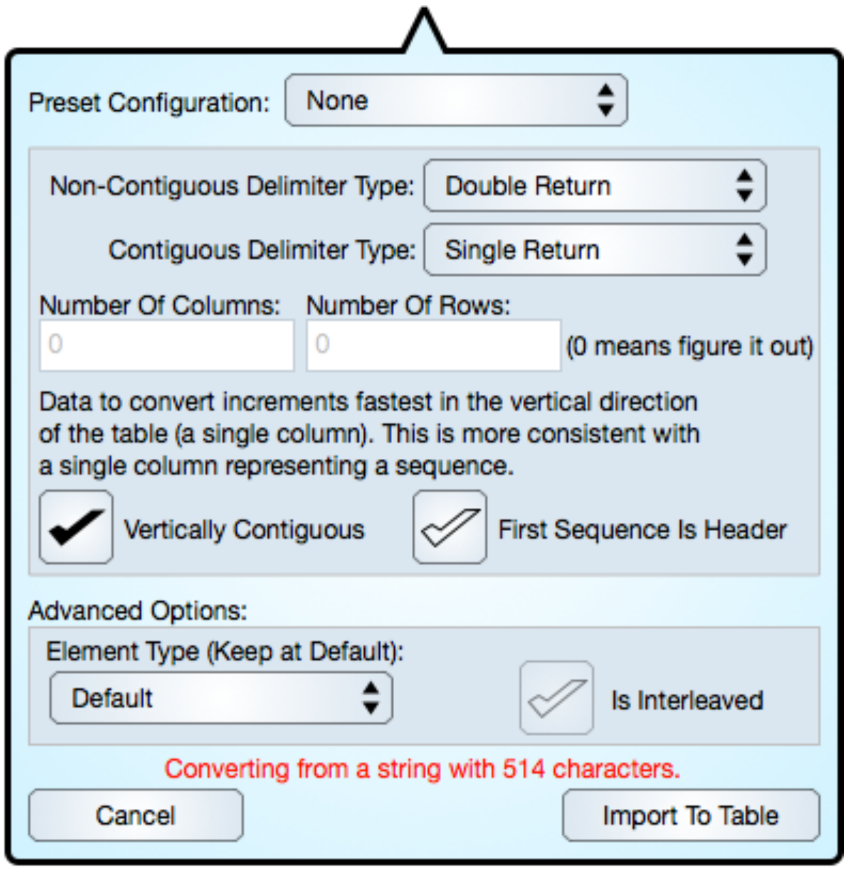
Same as Export.

Main Parameters

Non-Contiguous Delimiter Type : Same as Export.

Contiguous Delimiter Type : For a spreadsheet this is tab, for CSV a comma and for robust it is a single return character.

Number Of Columns : The number of columns to import to. This may be disabled if the table requires a specific value.


Preset Configuration: None

Non-Contiguous Delimiter Type: Double Return

Contiguous Delimiter Type: Single Return

Number Of Columns: 0 Number Of Rows: 0 (0 means figure it out)

Data to convert increments fastest in the vertical direction of the table (a single column). This is more consistent with a single column representing a sequence.

✓ Vertically Contiguous   First Sequence Is Header

Advanced Options:

Element Type (Keep at Default): Default   Is Interleaved

Converting from a string with 514 characters.

Number Of Rows : The number of rows to import to. Normally this is set to zero to let the import run its course.

Vertically Contiguous : Same as Export.

First Sequence Is Header : If the first sequence of the data to be imported is Header labels then that sequence is parsed and for a Spreadsheet those fields are placed in the table headers. For Spreadsheet and CSV data the first sequence is the first row of data. For Robust it is the number of lines up to the delimiter of two return characters.

Advanced Options

Do not alter these parameters.

Perform Operation

Information Text : This specifies how many characters is being imported. Characters for scalars are restricted to digits, -, + and 'e' which specifies exponential notation. For general strings the import is in terms of UTF-8 encoding.

Cancel : Select this to cancel the operation. Selecting a region away from the Import Selector also cancels the operation.

Import To Table : Select this to perform the import and also dismiss the selector.

Graph IDE ► Inspector Editors

An Inspector Editor is a special type of control that resides in its own location called the Inspector. The Inspector Editor provides an interface between the user (you) and a Vvidget (usually a graphic). That interface is a collection of various controls which alter parameters of the Vvidget one at a time.

The Inspector Editor editor is shown in a separate window called an Inspector Window. However, it can also show up in Graph IDE in many forms, including as control elements on the Graph IDE Document.

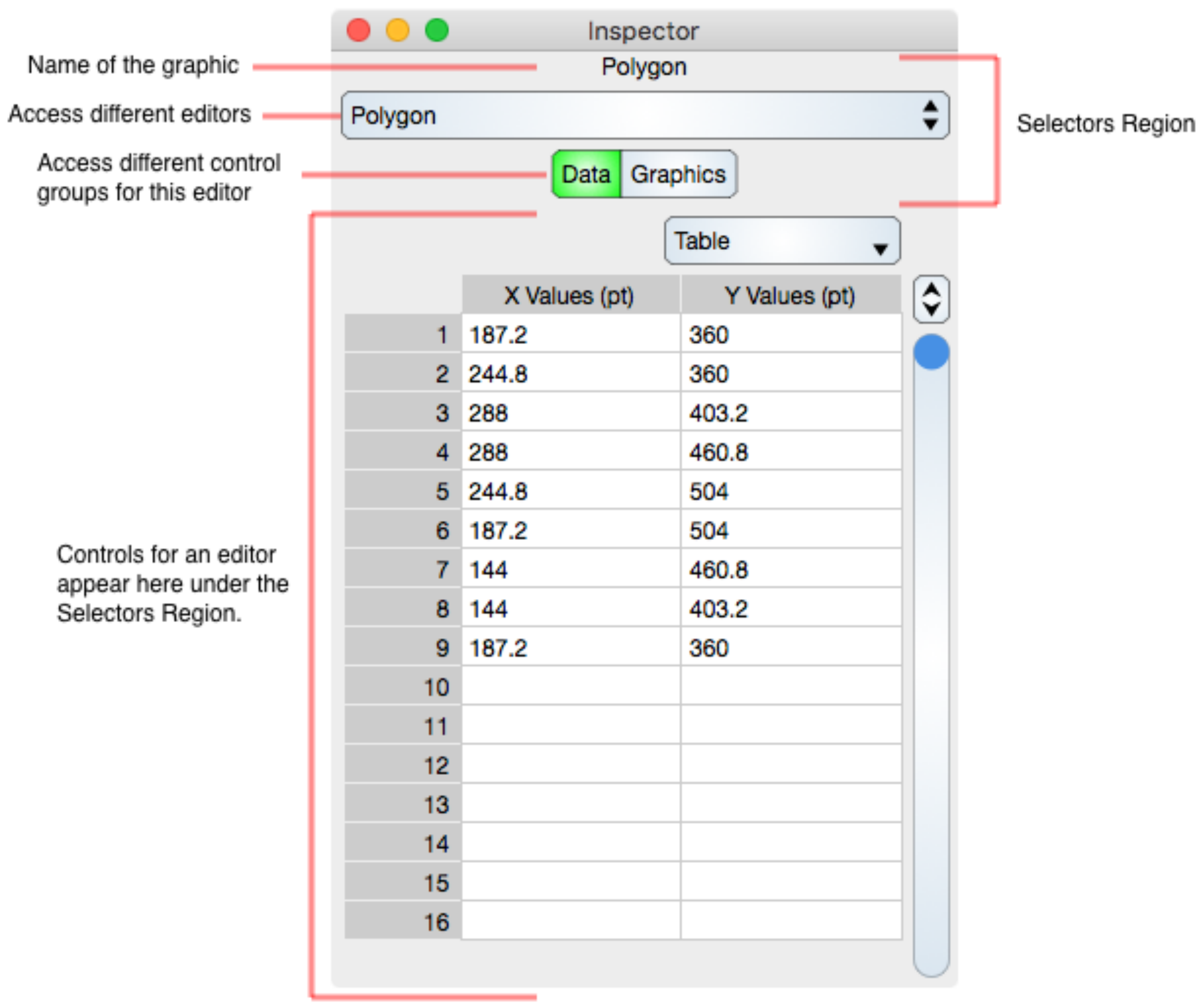
Each Basic Graphics, Graphs, Data Graphics and 3D Data Graphics has their own unique inspector editor. This inspector editor section deals with inspector editors that are common to those other sections. The following is a brief list of those common inspector editors:

Section	Description
Alignment	The alignment inspector editor is used to align a set of graphics, namely for the Group graphic.
Arranger	The arranger inspector editor is used to arrange a set of graphics, for example the Pie Chart graphic.
Animation	Used to set animation properties which includes the animation period and animation on or off state.
Caps	A caps inspector editor is used to add end caps to the Curve graphic.
Dictionary	A dictionary inspector editor gives access to the surrogate dictionaries.
Effects	Most graphics implement extended graphical attributes which are controlled by the effects editor.
Expert	An expert inspector editor is used to modify expert settings such as resize parameters and particular dictionary entries (names).
Graphics	All graphics, including Basic Graphics and Data Graphics, have a graphics inspector editor that control that graphic's graphical attributes.
Layer	An layer inspector editor is used to modify selections of graphics.
Magnifier	Describes the magnifier editor and functions.
Metadata	Describes the metadata editor. Metadata is utilized by Spotlight and other search facilities.
Network	A network inspector editor is used to define relationships between graphics.
Overview	Gives an overview of the inspector and inspector editors.
Parser	A parser shows different representations of graphics and is primarily for programming concerns.
Point Tags	A point tag inspector editor is used to denote points for Data Graphics and consists of a marker and a label at each point of the data graphic.
Program	The Program inspector editor is used to enter source code and execute or apply that source code.
Prototype	Gives an overview of the prototype inspector editors.

Graph IDE ▶ Inspector Editors ▶ Overview

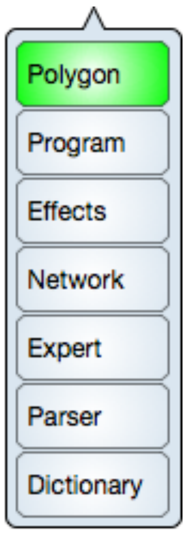
All graphics have a main inspector editor that control that graphic's main attributes. Those main inspector editors are described in other sections such as the Basic Graphics and Data Graphics sections.

Sections in this chapter detail sub-editors of the main editor. Those sub-editors are typically common to many main inspector editors. The following annotates the Polygon's Main Inspector Editor.


Inspector
Polygon
Name of the graphic
Access different editors
Polygon
Access different control groups for this editor
Data
Graphics
Selectors Region
Table
Controls for an editor appear here under the Selectors Region.

	X Values (pt)	Y Values (pt)
1	187.2	360
2	244.8	360
3	288	403.2
4	288	460.8
5	244.8	504
6	187.2	504
7	144	460.8
8	144	403.2
9	187.2	360
10		
11		
12		
13		
14		
15		
16		

The control to the subeditors is a Menu Selector. Shown below is the main menu selector for the Polygon.



Main Inspector Editor

Polygon : The menu item to the Polygon main inspector editor. The main inspector editor is always listed as the first element of the menu. The subsequent elements are subeditors.

Sub-Editors Of The Main Inspector Editor

Program : The menu item to the Program inspector subeditor.

Effects : The menu item to the Effects inspector subeditor.

Network : The menu item to the Network inspector subeditor.

Expert : The menu item to the Expert inspector subeditor.

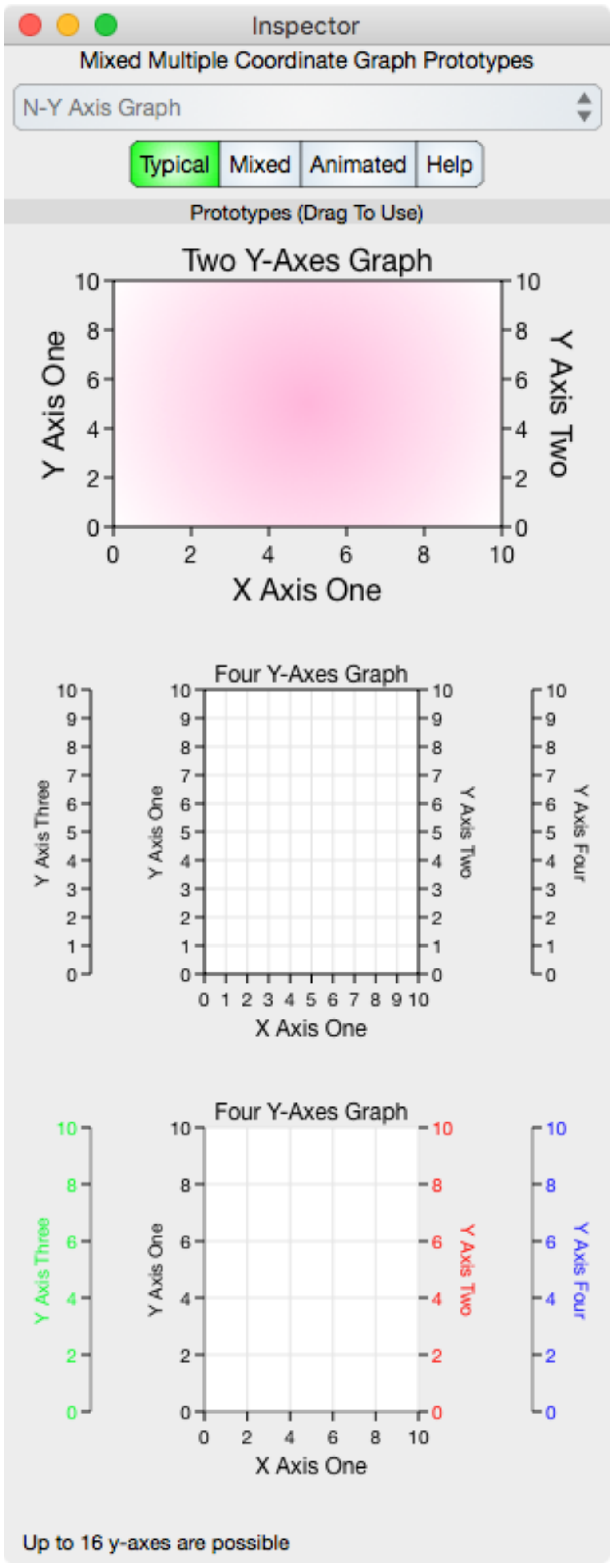
Parser : The menu item to the Parser inspector subeditor.

Dictionary : The menu item to the Dictionary inspector subeditor.

Graph IDE ▶ Inspector Editors ▶ Prototype

a.k.a.: Factory Inspector Editor

When a factory cell in the Graphic Selector is selected then the inspector is loaded with prototypes that correspond to that factory cell. One such prototype inspector editor for the Multiple Coordinate Graph is shown below.



Editor Selector

Choose one of the editor selector's cells to see various prototypes and help.

Typical : Typical prototypes

Mixed : Mixed coordinate prototypes, consisting of linear, log and date combinations.

Animated : Prototypes that are animated by a Program

Help : A small description of the Multiple Coordinate Graph. This description aides in the understanding of the class of graphic and the fuller description is within this manual.

Prototypes

Prototypes are draggable instances of the Multiple Coordinate Graph class. Drag them to the Graphic View to instantiate them.

All graphics in the Graphic Selector have their own prototypes and to see them click on the respective factory cell.

Note: A canonical graphic can be instantiated by selecting the graphic view and dragging out the canonical graphic. The prototype inspector editor is not the only way to instantiate graphics. Graphics can also be instantiated programmatically. See the Programming section for that.

Graph IDE ▶ Inspector Editors ▶ Layer

A Layer is a collection of independent graphics and is described in the Layer section.

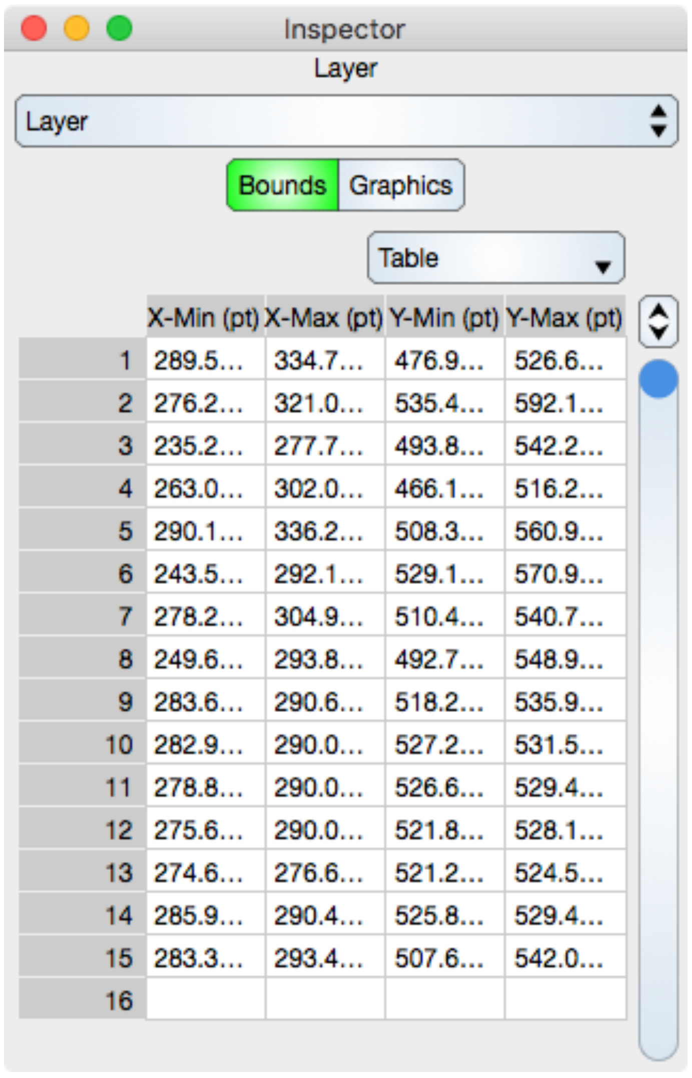
Some standard operations are itemized below.

- Layers are not created explicitly and are components of other graphics. For example, each overlay in the Graphic View is a layer. Graphs have four layers to hold graphics in their subcoordinates and also graphical elements in the coordinate of the graph.
- Graphics in a layer can be selected by first choosing the selection cell in the Graphic Selector and then dragging over the area of graphics to select.
- Any factory method as described in the Graphic Selector, Palettes and elsewhere are used to add graphics to a layer. Pasting within the Graphic View also adds graphics to the focused layer.
- To program a Layer see the Programming section.

Note that graphics in a layer are individually editable and selectable and operate independently. Contrast that to a Group graphic which is a container for graphics that are not independent.

Bounds Editor

The Bounds Editor for a Layer is shown below.



	X-Min (pt)	X-Max (pt)	Y-Min (pt)	Y-Max (pt)
1	289.5...	334.7...	476.9...	526.6...
2	276.2...	321.0...	535.4...	592.1...
3	235.2...	277.7...	493.8...	542.2...
4	263.0...	302.0...	466.1...	516.2...
5	290.1...	336.2...	508.3...	560.9...
6	243.5...	292.1...	529.1...	570.9...
7	278.2...	304.9...	510.4...	540.7...
8	249.6...	293.8...	492.7...	548.9...
9	283.6...	290.6...	518.2...	535.9...
10	282.9...	290.0...	527.2...	531.5...
11	278.8...	290.0...	526.6...	529.4...
12	275.6...	290.0...	521.8...	528.1...
13	274.6...	276.6...	521.2...	524.5...
14	285.9...	290.4...	525.8...	529.4...
15	283.3...	293.4...	507.6...	542.0...
16				

Table

Table : The table shows the bounds of the graphics in the layer. Each row shows the bound of an individual graphic in the layer in the sequence order of the layer. Those bounds can be edited directly or via any of the individual graphic's Graphics inspector editor. The table controls are described in the Tables section.

The graphic bounds shown in the table are determined by the Selection control described below.

Graphics Editor

The Graphics Editor for a Layer is shown below.

Menu Shortcuts

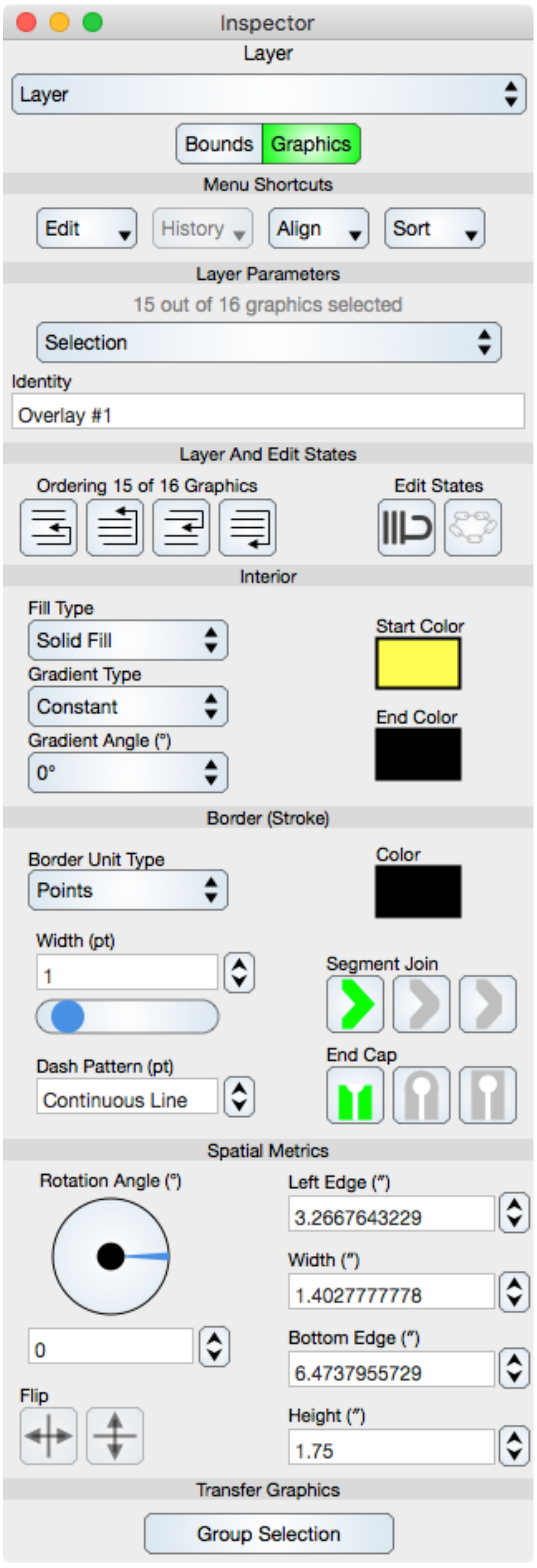
Edit : The Edit menu shortcut is one of Cut, Copy, Paste, Delete or Select All.

History : Shows the history of the layer. For this to be enabled, history recording must be selected in the Graphic View tool inspector. Selecting a menu item makes that history recording the current state. On platforms that accept hover, hovering over a menu item will pan the history. The last menu item clears the history only for the focused layer. All history can also be cleared in the Graphic View tool inspector.

Align : The align menu is used to align graphics in the layer.

Sort : Sorting the layer means to arrange the sequence index of the layer elements against another dimension. That dimension is either Natural or Y-Descending. Natural means the way text is read (from top to bottom) and in the case of intersecting elements then the element with greater area has a lesser sequence index. This is consistent with backdrops and also tabbed responder order. The ordering is most important when order refers to implicit functions such as tabbing in HTML. The Y-Descending sorting is appropriate for cumulative area graphs. Care must be taken to not sort when the sequence index order provides for a z-buffer effect.

Layer Parameters


Inspector
Layer
Layer
Bounds | Graphics
Menu Shortcuts
Edit | History | Align | Sort
Layer Parameters
15 out of 16 graphics selected
Selection
Identity
Overlay #1
Layer And Edit States
Ordering 15 of 16 Graphics
Edit States
Interior
Fill Type
Solid Fill
Gradient Type
Constant
Gradient Angle (°)
0°
Start Color
End Color
Border (Stroke)
Border Unit Type
Points
Color
Width (pt)
1
Segment Join
Dash Pattern (pt)
Continuous Line
End Cap
Spatial Metrics
Rotation Angle (°)
0
Left Edge (")
3.2667643229
Width (")
1.4027777778
Bottom Edge (")
6.4737955729
Height (")
1.75
Flip
Transfer Graphics
Group Selection

Selection : Determines which grouping in the layer is to be edited by the inspector. Normally this is kept to Selection indicating that only selected graphics are to be altered. However, it could also be any of the other options such as Draw indicating that any drawable graphic is to be altered.

Identity : Defines the identity string of the layer. The identity helps determine the current overlay as well as describes the intended use of the overlay or layer. It is displayed by the Navigator, Cursor Information and the data layers for Graphs.

Transfer Graphics

Group Selection : Removes the selected graphics from the layer and then places those selected graphics into a Group graphic which is then added to the layer. When this happens the focus is transferred to the group. From there, the group can be ungrouped if needed thus reversing this operation.

Common Controls

Other controls common to all graphics are described in the Graphics section.

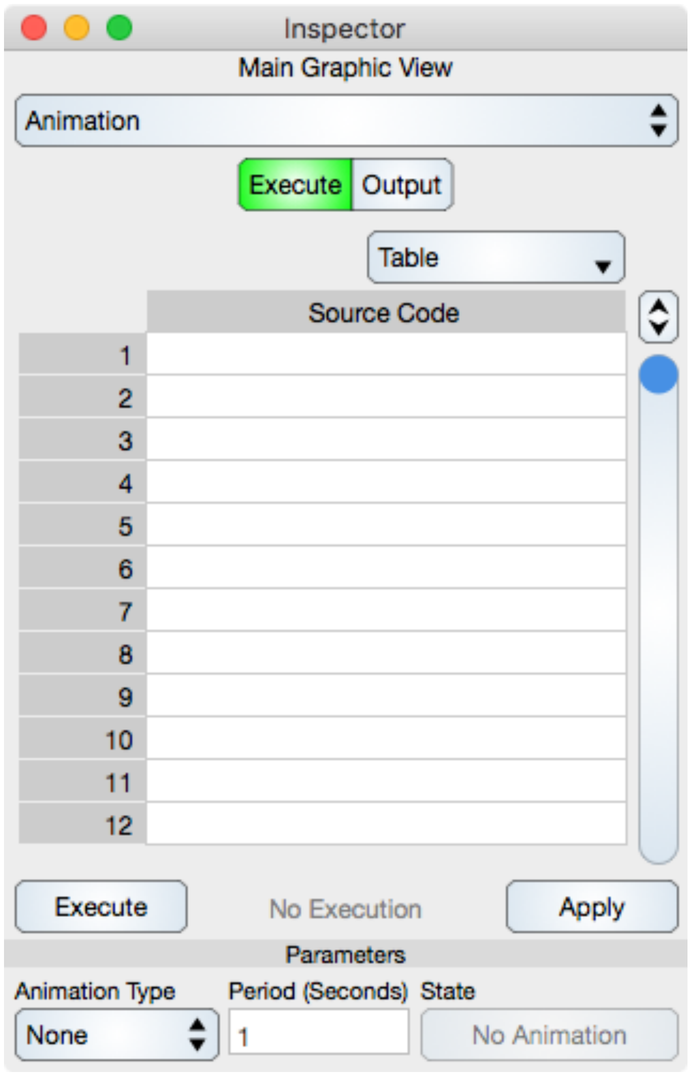
When altering a graphic attribute that attribute is applied to each element in the layer independently. Notice how if the graphics are rotated then they are rotated individually with the fulcrums at the center of each individual graphic. Contrast that to the way a Group rotates its graphics which rotates around a common fulcrum point which is the center of the group as a whole.

Animation calls each node in the Layer tree, i.e.: it calls upon all graphics, to execute that graphic's Program. It does that at a period defined in seconds.

Animation can be used to automate data retrieval, blinking of graphics, moving graphics and all sorts of things.

Since Animation starts at the root layer it is associated with the Graphic View and the animation inspector is a sub-editor of the graphic view inspector. The Animation inspector editor is shown below.

Execute Editor



Table

Source Code : A Table that shows and edits the program source code. The table cells are program lines, normally statements. The Programming section gives examples of source code. Once entered then you must at least Apply it or your entry is lost. For the most part, you should type source code in a separate text editor, copy it and paste it into the Source Code table as the table is not a full IDE.

Execute : Selecting the Execute button will parse and execute the source code. As a side effect, it will also apply (save) the source code.

Apply : Selecting the Apply button will save the source code but will not parse or execute it.

Information : Shows the number of times the program was executed.

Parameters

Animation Type : Defines how the animation should take place and is one of None, Once or Periodic.

Period : Defines the period that the animation is executed with. This value should be set at a reasonable value. If it is zero then that means the animation will happen as often as possible.

State : Select this to turn the animation on or off.

Output Editor

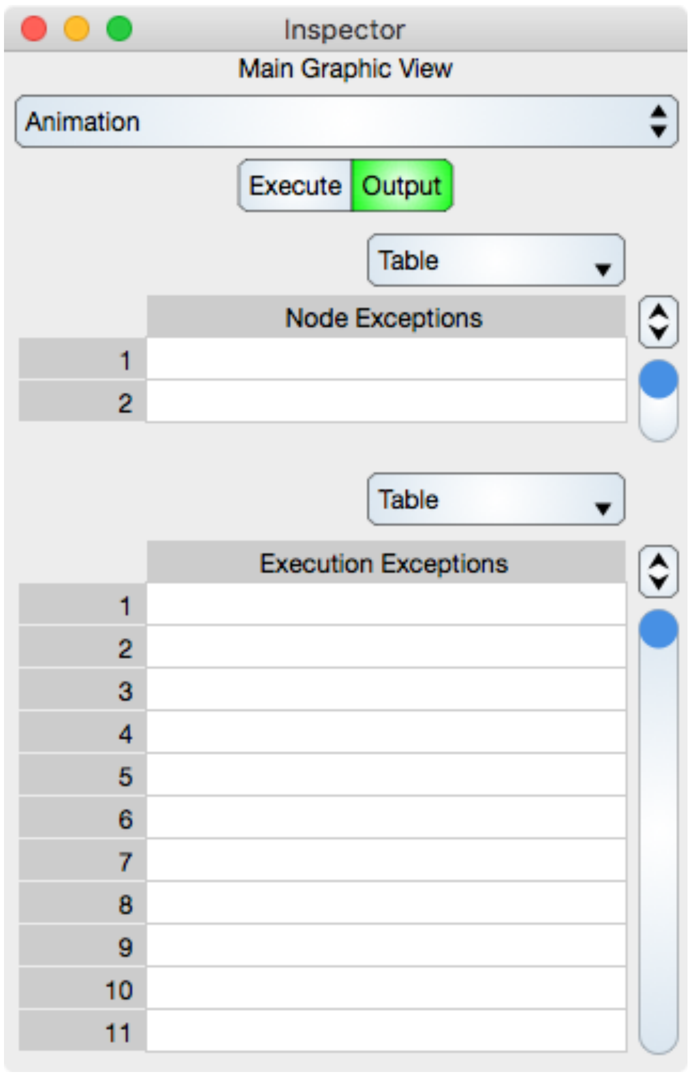
Output of the program is shown in the Output inspector editor.

Table

Standard Output : The result of fprintf(stdout,) function is shown here. In fact, any output to the stdout stream is shown in this table. The stdout stream is only redirected to this table when the Execute button is clicked. During animation the output goes to the system console.

Standard Error : The result of fprintf(stderr,) function is shown here. In fact, any output to the stderr stream is shown in this table. The stderr stream is only redirected to this table when the Execute button is clicked. During animation the output goes to the system console.

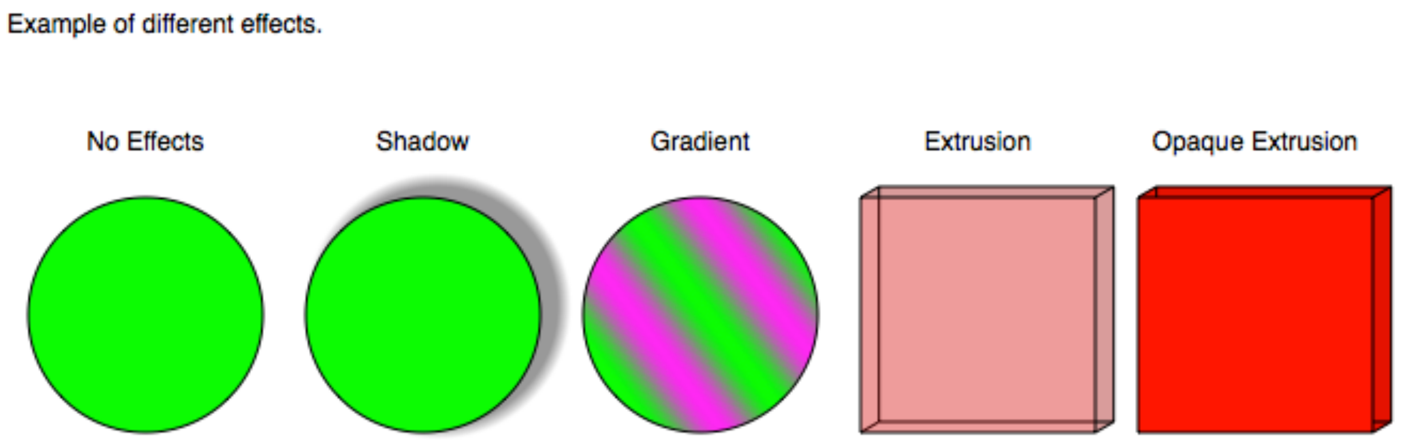
Other messages, not associated with unix streams, may appear in either the output or error tables. Windows does not support stream redirection. If your program does not generate the intended results upon selecting the Execute button then consult these tables for explanations.



	Node Exceptions
1	
2	

	Execution Exceptions
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	

All graphics, including Basic Graphics and Data Graphics have an effects inspector editor that control that graphic's extended effects attributes. Some of those effects are shown in the following figure.



Inspector Editor

The Inspector Editor for the effects controls is shown below.

Does Use Effects : An overriding flag to determine if any effects are used.

Shadow

Most graphics implement a shadow which gives the appearance of a light source at a particular angle far from the graphic and a backdrop on which the shadow casts itself.

Draw : If selected then the shadow will draw.

Offset : Sets the offset of the shadow.

Diffusion : Sets the diffusion of the shadow.

Color : Sets the Color of the shadow.

Orientation : The angle, from the y = 0, x > 0 line, of the shadow orientation. The dial control is described in the Dial section.

Gradient

Most graphics implements a gradient which is defined as a grading between two colors. Some gradients have parameters such as period and angle.

Gradient Type : Constant means a single-color fill, otherwise select one of the other options.

Start Color : Sets the start Color of the gradient. If the gradient is Constant then this is also the solid fill color of the interior of the graphic.

End Color : Sets the end Color of the gradient. If the gradient is Constant then this is unused.

Period : Sets the period of the gradient. Not all gradient functions have a period.

Orientation : The angle, from the y = 0, x > 0 line, of the gradient orientation. Only axial gradient types have an orientation. The dial control is described in the Dial section.

Extrusion

Some graphics implement an extrusion effect, which gives the graphic a pseudo 3D appearance. The effect varies from graphic to graphic and not all options are appropriate to the extrusion effect.

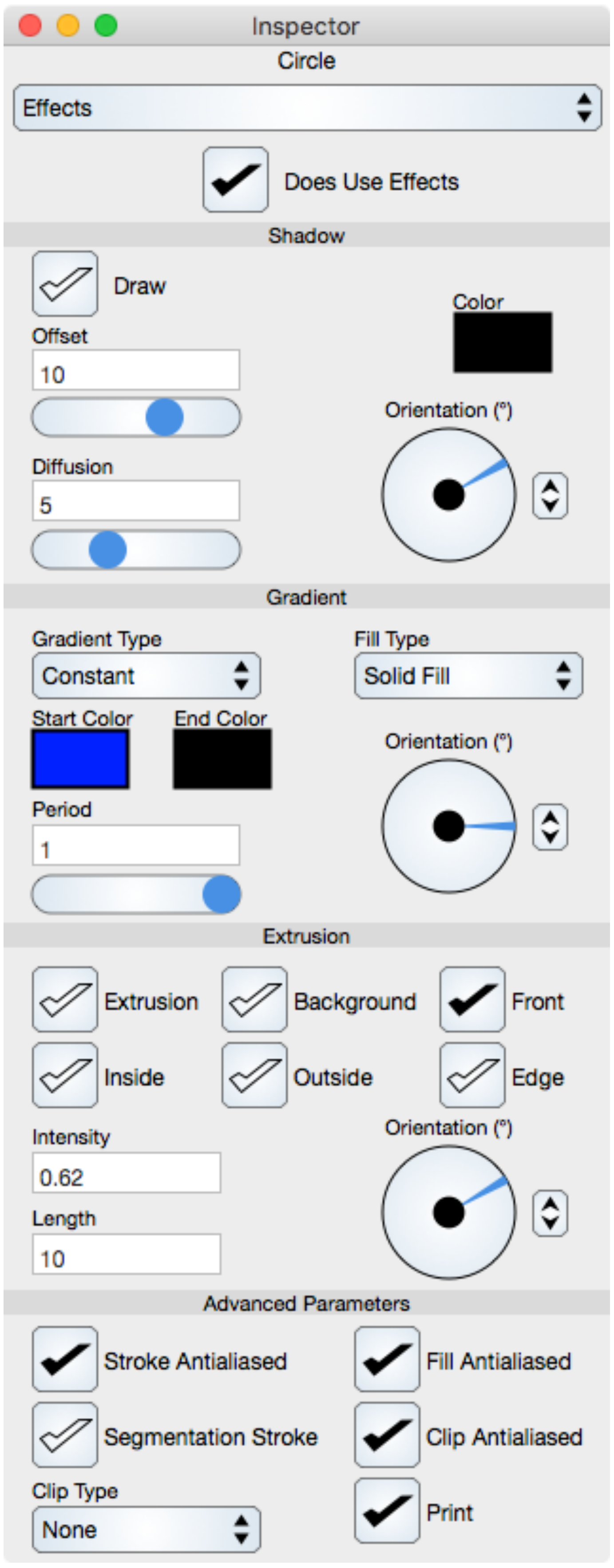
Advanced Parameters

Stroke Antialiased : When on the stroke is antialiased.

Fill Antialiased : When on the fill is antialiased.

Segmentation Stroke : Normally this is off indicating that an entire path is stroked at once if possible. Setting this to on makes each path segment stroke sequentially. This has the effect of all stroke attributes starting from their initial values for each segment, such as the dash pattern etc. Normally the segmentation stroke should be off.

Clip Antialiased : When on the clip is antialiased. Typically clipping should not be antialiased as the antialiasing appears as an unwanted artifact.



Inspector

Circle

Effects

✓ Does Use Effects

Shadow

☐ Draw

Offset: 10

Diffusion: 5

Color

Orientation (°)

Gradient

Gradient Type: Constant

Fill Type: Solid Fill

Start Color

End Color

Period: 1

Orientation (°)

Extrusion

☐ Extrusion ☐ Background ✓ Front

☐ Inside ☐ Outside ☐ Edge

Intensity: 0.62

Length: 10

Orientation (°)

Advanced Parameters

✓ Stroke Antialiased ✓ Fill Antialiased

☐ Segmentation Stroke ✓ Clip Antialiased

Clip Type: None

✓ Print

Clip Type : Determines when the graphic clips as a clipping mask. This should probably be left off because clipping masks are difficult to understand. The Label graphic is the only graphic that does not implement clipping as a mask and for that case clipping is limited to textual clipping.

Print : If on then when the Graphic View is printed then the graphic will be part of the print otherwise the graphic will be excluded from printing. Graphics are set to print by default, except for the grid on the graphic view.

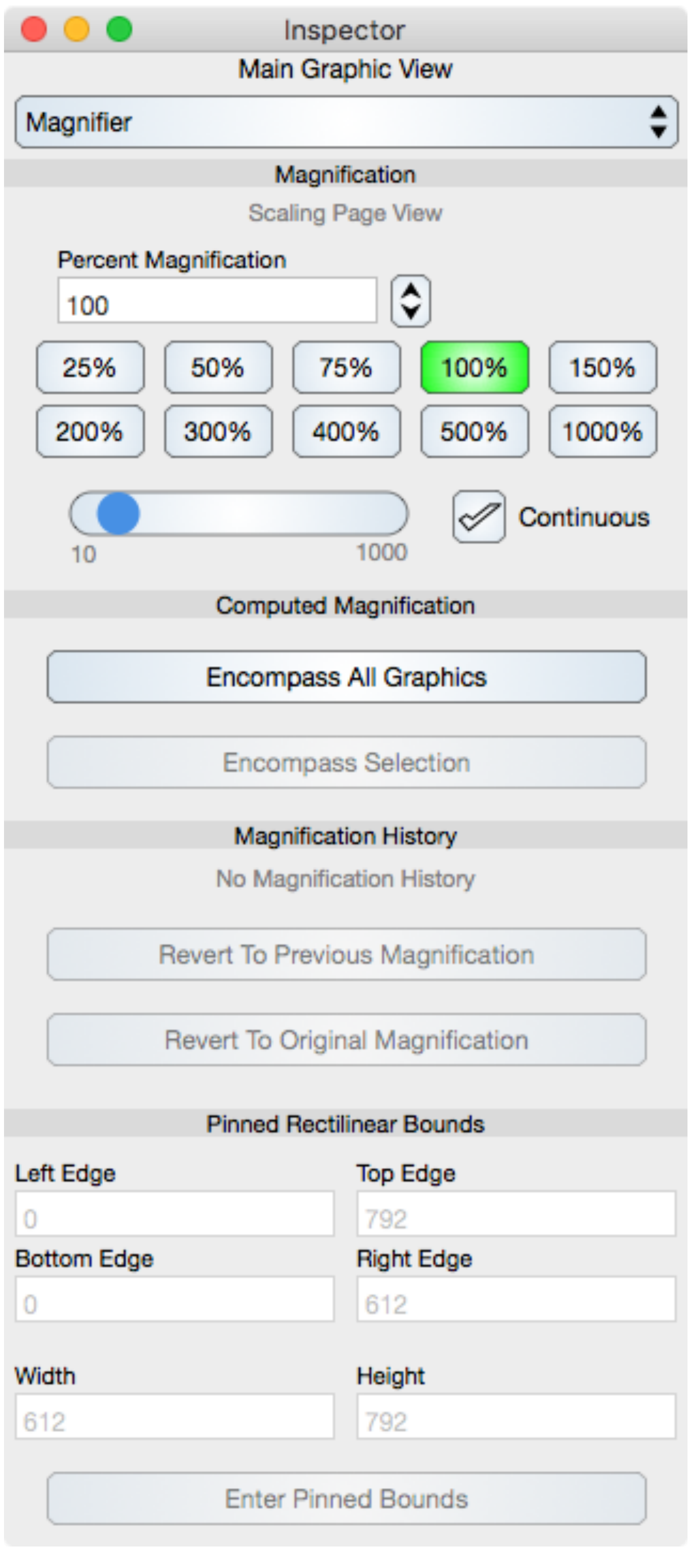
Graph IDE ▶ Inspector Editors ▶ Magnifier

All graphics with a coordinate system, including Graphs and the Graphic View are able to perform magnification.

Note the following distinction in two coordinate systems:

- For a graphic view, magnification increases the extent of the graphic view. For example, if the graphic view has letter size dimensions and is magnified by a factor of two then the new size, as shown on the screen, is twice letter size.
- For graphs and other representations that display metric information, magnification performs a scaling of the coordinate system and the resulting dimension on the screen remain the same. For example: The graph frame remains constant however the graph limits change.

This section defines that operation and the associated inspector editor as shown in the following figure.



Magnification

Percent Magnified : Enter a value in the text field to set the magnification. 100 means no magnification 50 means half magnification, 200 means twice magnification, etc.

Preset Magnification : Choose one of the predefined buttons to set the percent magnification.

Slider : Use the Slider to vary the magnification.

Continuous : If the continuous button is selected then the slider magnification is performed as the slider knob moves.

Computed Magnification

Encompass All Graphics : Adjusts the magnification so that all graphics are shown in the visible portion of the document's graphic view.

Encompass Selection : Adjusts the magnification so that only selected graphics are shown in the visible portion of the document's graphic view.

Magnification History

Revert To Previous Magnification : Select this button to revert magnification.

Revert To Original Magnification : Select this button to empty the magnification history and revert to the original magnification.

Pinned Rectilinear Bounds

Pinned bounds are the reference point for magnification. For a graphic view they are the page size and can not be changed except by changing the page size itself on the Graphic View inspector. For a graph they are the graph limits at the time the Magnification inspector was brought forward.

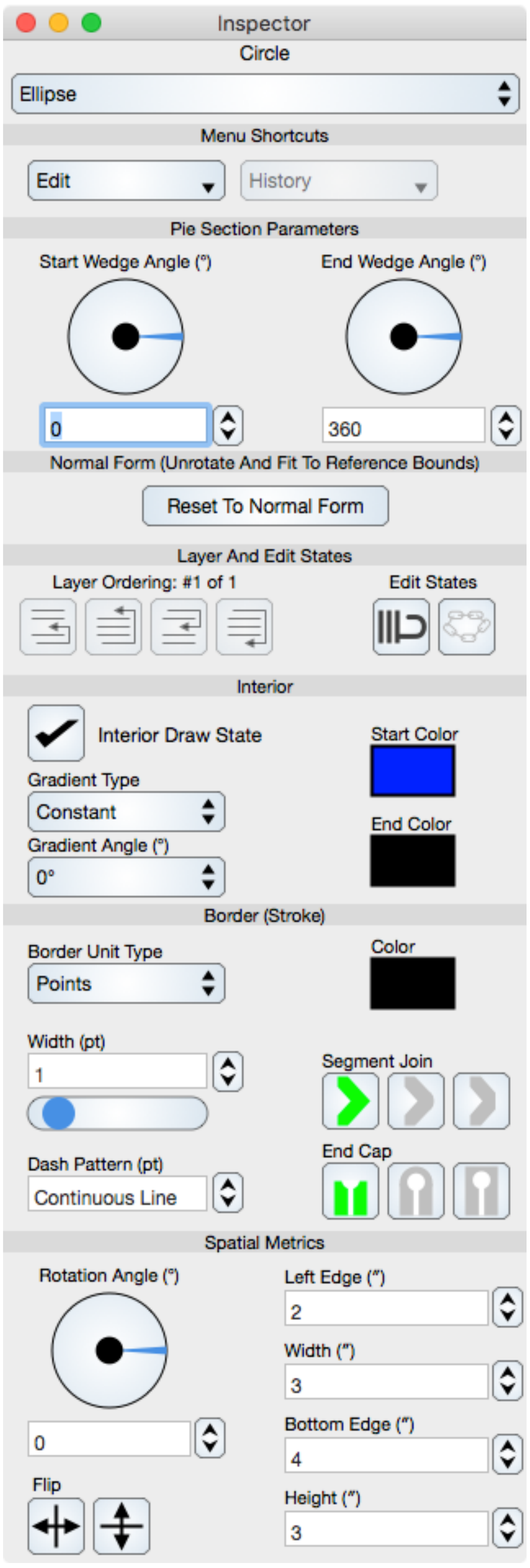
For graphs, you can set the pinned bounds. Pinned bounds and magnification references are always in rectilinear coordinates.

Enter Pinned Bounds : Select this button to enter new pinned bounds as shown in the text fields. Preference is given to the absolute values (not width or height). You can also type the return key in the text field to enter the respective bound change.

Graph IDE ▶ Inspector Editors ▶ Graphics

All graphics, including Basic Graphics and Data Graphics, have a graphics inspector editor that control that graphic's graphical attributes. This section defines one such inspector editor as an example, the Circle's Graphics Inspector Editor, which is shown in the following figure.

Although the figure below annotates some of the components the circle's graphics inspector editor, the best way to understand each control is by using Graph IDE, making a circle, and adjusting each control one at a time to see the affect it has on the circle.



Ellipse
Edit
History
Start Wedge Angle (°): 0
End Wedge Angle (°): 360
Layer Ordering: #1 of 1
Interior Draw State: ✓
Start Color: 🟦
End Color: ⬛
Gradient Type: Constant
Gradient Angle (°): 0°
Border Unit Type: Points
Color: ⬛
Width (pt): 1
Dash Pattern (pt): Continuous Line
Rotation Angle (°): 0
Left Edge ("): 2
Width ("): 3
Bottom Edge ("): 4
Height ("): 3

Menu Shortcuts

Edit : The Edit menu shortcut is one of Cut, Copy, Paste, Delete or Select All.

History : Shows the history of the graphic. For this to be enabled, history recording must be selected in the Graphic View tool inspector. Selecting a menu item makes that history recording the current state. On platforms that accept hover, hovering over a menu item will pan the history. The last menu item clears the history only for the focused graphic. All history can also be cleared in the Graphic View tool inspector.

Specific Controls

Controls specific to a graphic, in this instance the Pie Section and Normal Form controls, are defined in the respective sections; for example the Circle section. Those controls are typically at the top of the inspector editor controls and what follows are controls common to most graphics. This section only details common controls. For descriptions of specific controls consult the corresponding section in this manual.

Layer And Edit States

Layer Ordering : Controls the order of the focused graphic relative to other graphics in the same Layer. The buttons are: order up, forward, order down and bottom.

Edit States : The lock specifies whether the graphic can be altered by direct mouse interactions such as resize and move or deleted, and the link button defines whether the graphic's link status is shown. While locked, the graphic can not be deleted, moved or resized directly however it can be altered by explicit use of its inspector editor controls.

Interior

Interior Draw State : Select this to draw the interior of the graphic. This control can also be a button that can be one of No Fill, Solid Fill, E/O Fill. E/O means Even/Odd winding rule fill.

Gradient Type : Constant means a single-color fill, otherwise select one of the other options. See Effects for additional information.

Gradient Angle : Select one of the options for preset gradient angles. See Effects for additional information.

Start Color : Sets the start Color of the gradient. If the gradient is Constant then this is also the solid fill color of the interior of the graphic. See Effects for additional information.

End Color : Sets the end Color of the gradient. If the gradient is Constant then this is unused. See Effects for additional information.

Border (Stroke)

Border Unit Type : Select one of None, Points or Indigenous. None means do not draw the border.

Color : Sets the Color of the stroke.

Width : Sets the width of the border stroke. The width is from the outer to inter stroke boundary and the stroke is centered about the border of the graphic.

Dash Pattern : Sets the pattern of the dash. Leave this empty (Continuous Line) for a solid stroke, otherwise set this to a sequence of numbers that correspond to the solid and gap intervals of the dash (an even number of numbers). See Simple Draw Attributes for additional information.

Segment Join : Sets the join type. See Simple Draw Attributes for additional information.

End Cap : Sets the end cap type. See Simple Draw Attributes for additional information.

Spatial Metrics

Rotation Angle : Rotates the graphic. The rotation angle is relative to the x > 0, y = 0 line and positive angles are counterclockwise. The dial control is described in the Dial section.

Flip : Flips the graphic relative to the vertical or horizontal center line of the graphic.

Left Edge, Width, Bottom Edge, Height : Sets the left edge, width, bottom edge and height respectively of the reference bounds of the graphic. The bottom and y-minimum are the same (the y-coordinate is never flipped).

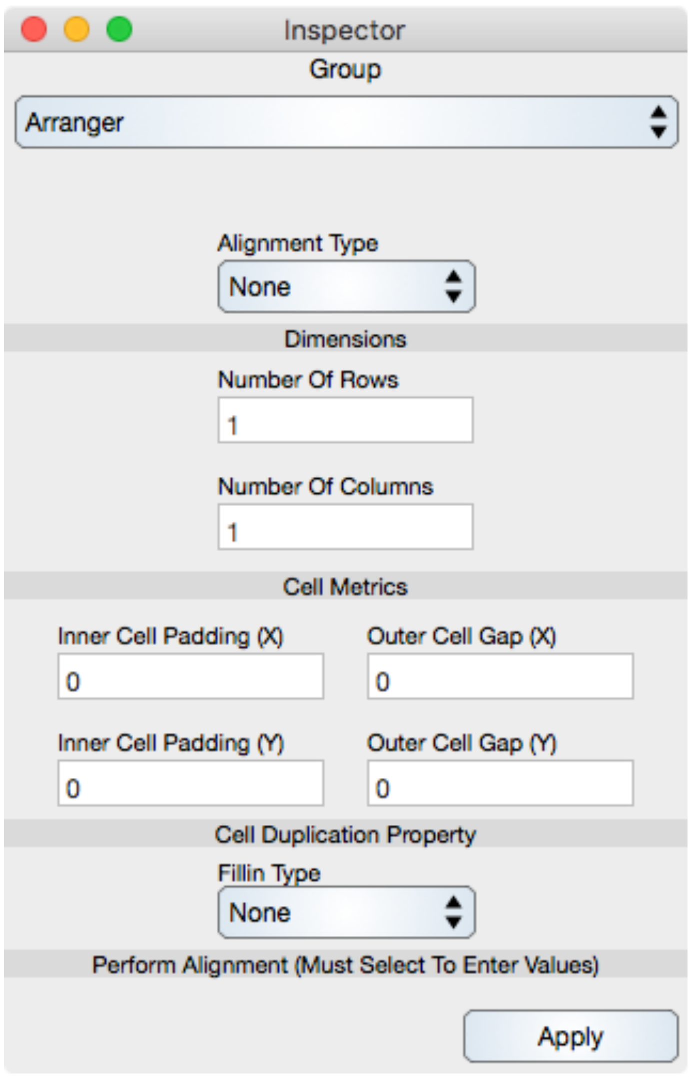
The units of the reference bounds are in the coordinate space of the graphic. If the graphic is not on a graph then the units are in the report units of the Graphic View. If the graphic is on a Graph then the reference units are that of the graph itself. The unit designations (symbols) appear in the label above the reference bound fields. If the graph is unitless (which is often the case) then there is no unit designation.

Graph IDE ► Inspector Editors ► Alignment

The alignment inspector editor is used to align elements of the Group graphic. Tables are a type of group and the alignment is used for tables as well. Alignment means to consider each graphic in a set as elements of a regular grid and to place those elements within the cells of the regular grid. When there are less graphics than cells then the graphics may be duplicated to fill in the table or matrix. If there are too many graphics then the graphics may be deleted to fulfill the requirements of the number of cells.

The alignment editor is in general a Arranger. When the group has special meaning then the alignment editor is substituted by a domain-specific alignment editor such as the Pie Chart and Bar Chart.

This section annotates the alignment inspector editor as follows.



Alignment

Alignment Type : One of None, Matrix or Table. Matrix aligns to uniform cells while Table aligns to cells that are uniform in the x-direction and then the y-direction independently. The Table type is the mechanisms by which Tables align their cells.

Dimensions

Number Of Rows : The number of rows in the matrix or table of cells.

Number Of Columns : The number of columns in the matrix or table of cells.

Cell Metrics

Inner Cell Padding : The space between the graphic of the cell and the cell borders.

Outer Cell Gap : The space between adjacent cells.

Cell Duplication Property

Fillin Type : Determines the way cells in the table or matrix are duplicated. None means to leave cells empty, X-Direction means to duplicate cells to fillin new columns, Y-Direction means to duplicate cells to fillin new rows, and Both means to fill in all cells. Cells are duplicated from the graphic of the last cell in the matrix or table.

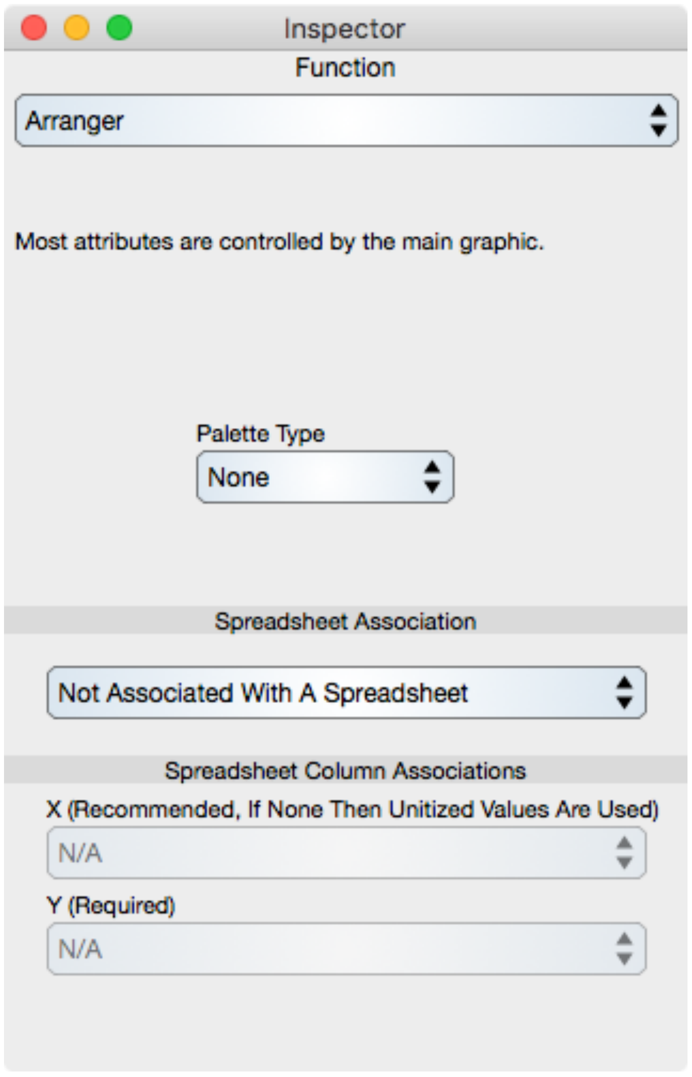
Perform Alignment

Apply : Attributes of the alignment specified above must be explicitly applied to take effect. Selecting the Apply button will perform the alignment and if needed duplicate cells.

Graph IDE ▶ Inspector Editors ▶ Arranger

The arranger is an abstraction to align elements of a collection. The Group and Tables graphics are arranged via the Alignment properties. The Pie Chart and Bar and Column Chart are arranged via domain specific algorithms. The Function and Scatter graphic consider the collection as their constitutive points.

Below is the arranger for the Function graphic. Note that the only properties are those that refer to other graphics, namely the Color Selector palette and Spreadsheet. That is because the Function is defined by its own constitutives.



Most attributes are controlled by the main graphic.

Palette Type: None

Spreadsheet Association

Not Associated With A Spreadsheet

Spreadsheet Column Associations

X (Recommended, If None Then Unitized Values Are Used): N/A

Y (Required): N/A

General

Palette Type : When selected then the colors assigned in a palette as define in the Color Selector section are used. Choose None in order to break the reference and then use a uniform color instead.

Spreadsheet Association

Defines and selects the spreadsheet used to generate the function points. Any spreadsheet can be chosen but that spreadsheet should conform to the intended use. When different spreadsheets are chosen then all representation parameter values are maintained so that this is a good way to flip through alternative data sets while maintaining the representation graphical attributes.

When the representation is made without a spreadsheet then it can be later associated with a spreadsheet in order to delegate the constitutive properties of the representation to an external source.

Column Associations

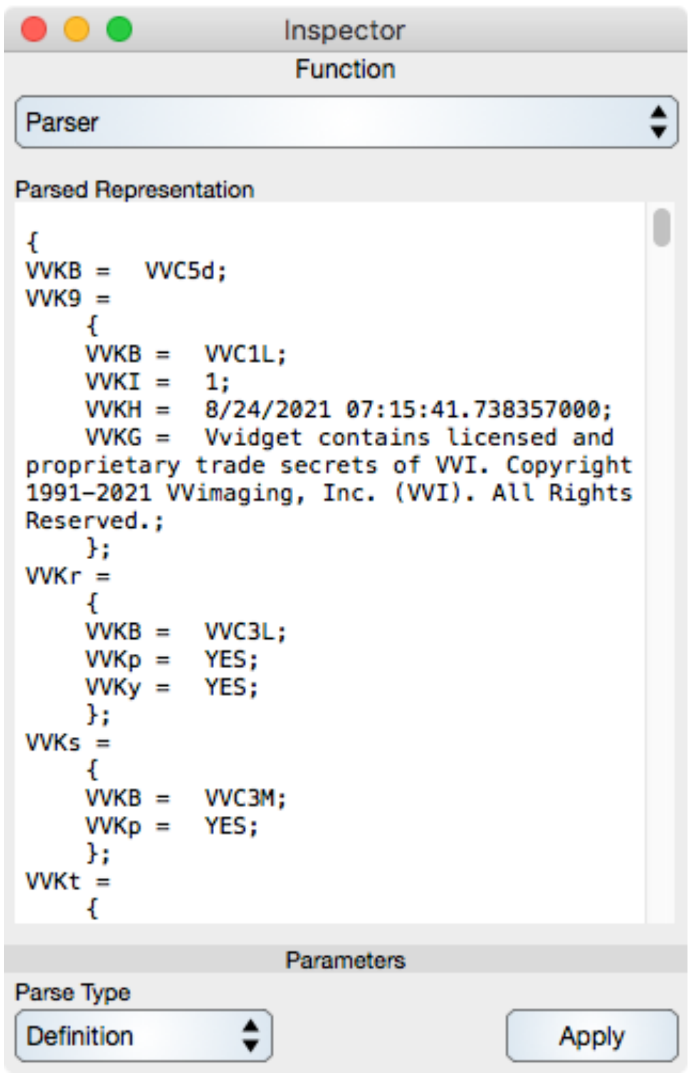
The column associations are set during representation creation but can be reset here. Choosing different columns are a good way to quickly view different data sets for one line graph.

X : Specifies the column to associate with the x component of the data points.

Y : Specifies the column to associate with the y component of the data points.

Graph IDE ► Inspector Editors ► Parser

The Inspector Editor for the parser is shown below.



```
{
VVKB =   VVC5d;
VVK9 =
    {
    VVKB =   VVC1L;
    VVKI =   1;
    VVKH =   8/24/2021 07:15:41.738357000;
    VVKG =   Vvidget contains licensed and
proprietary trade secrets of VVI. Copyright
1991-2021 VVimaging, Inc. (VVI). All Rights
Reserved.;
    };
VVKr =
    {
    VVKB =   VVC3L;
    VVKp =   YES;
    VVKy =   YES;
    };
VVKs =
    {
    VVKB =   VVC3M;
    VVKp =   YES;
    };
VVKt =
    {
```

Parsed Representation

The Parsed Representation area shows the representation. For Indigenous and Graphic type this area is a Graphic View from which the representation can be edited. This is a general purpose graphic edit view so all of the tools of Graph IDE are at your disposal while altering the representation. For other representation types it is a textual scrollview area which may be editable and applied back to the graphic.

Parsed Type

Use the pop up button on the lower left of the inspector to parse one of the following representations.

Indigenous : Shows an exact duplicate of the focused graphic. Because it is exact, the parsed representation may be outside the parser viewing region.

Graphic : Shows the graphical representation of the focused graphic. Because, as of yet, all focused elements are graphics the graphical representation is the same as the indigenous representation.

Data : Shows the textual encoding of the data of the graphic. The data is usually appropriate only for Data Graphics and is the point-wise data. This representation parses into a textual representation which can be edited and then applied. As such, it is an alternative to the table interface of the graphic's main data inspector.

PDF : Shows the textual PDF representation.

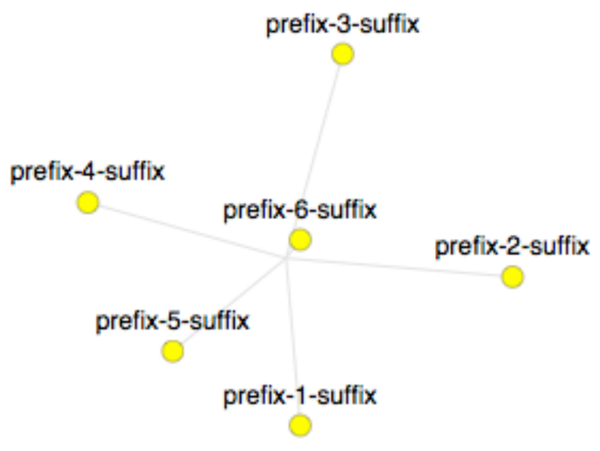
Definition : Shows the Dictionary representation. A dictionary is a key value encoding of the graphic. If you know how, it can be altered directly and then applied (not recommended).

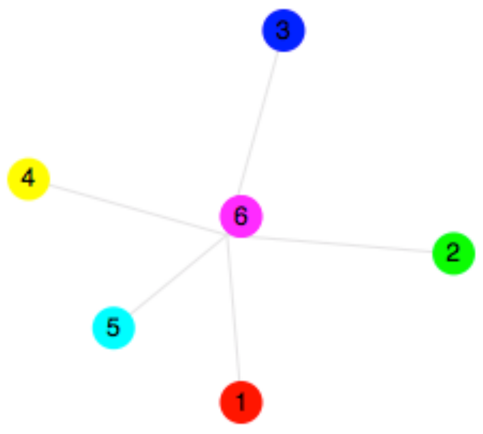
Apply

Some parsed representation operate symmetrically. That is, the graphic can be parsed into a representation and that representation can be parsed back into the graphic. When that is the case the Apply button is enabled, otherwise it is disabled.

Graph IDE ▶ Inspector Editors ▶ Point Tags

Most 2D Data Graphics and 3D Data Graphics have a point tags inspector editor that control that graphic's point-oriented graphical attributes called point tags. Point tags consist of markers and labels. Markers are an arbitrary graphic, such as a circle or square, placed at each data point. A marker can be made from scratch or selected from a pre-made palette of graphics as shown and described below. A Label is a single line of text centered about each point. The figures below show examples of point tags.





The inspector for point tags is described below.

Markers Editor

The Markers Editor is shown below.

Edit

Edit : The Graphic View from which a marker can be edited. This is a general purpose graphic edit view so all of the tools of Graph IDE are at your disposal while making a marker.

Inspect : Forwards an inspector editor that is applicable to the Edit view. To see controls in the resulting inspector editor for the marker first make sure that you have selected the marker in the Edit view.

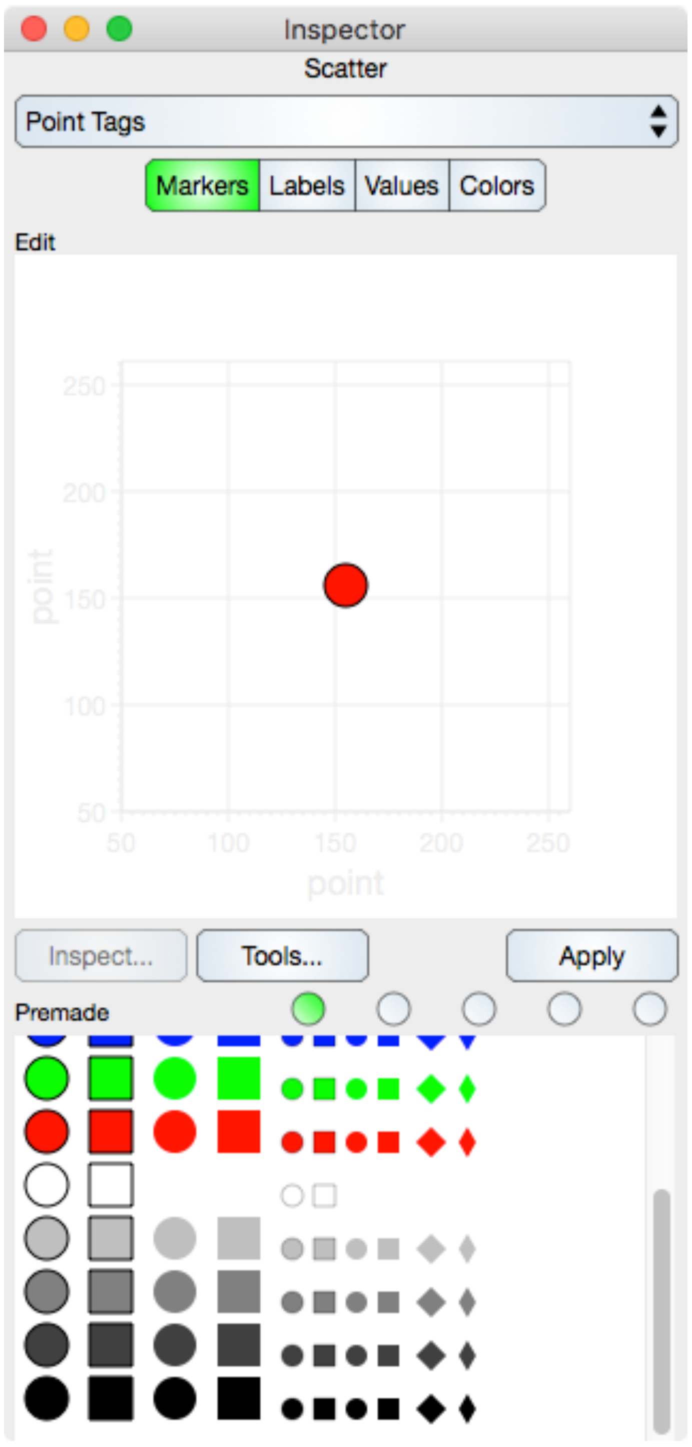
Tools : Forwards a Graphic Selector that is applicable to the Edit view.

Apply : Applies (copies) the graphic in the Edit view as the marker for the focused graphic.

Premade

A group of various premade graphics. Click on a graphic to make it the current marker, or click not-on-a-graphic to remove the marker.

Template Selection : Select one of the radio buttons to see up to five different premade templates.



Labels Editor

Labels are a single-line textual graphic, either arbitrary or predefined, placed at each data point. A label is formulated as described below.

Label Type

Label Type : The label is constructed as one of: {"None", "I", "X", "Y", "X, Y", "Fixes", "Values", "Custom Strings"}; where "None" means no label, "I" is the index of the point, X is the x-value, Y is the y-value, "X, Y" is the x-value and y-value pair with a comma in between, "Fixes" is the prefix and suffix text only, "Values" is the value as defined in Values (see below) and "Custom Strings" is the arbitrary text entered into the table. Each type (except None and Custom Strings) always shows the fixes (if any).

Label Placement and Other Parameters

Offset Angle : The offset angle of the label relative to each point.

Offset Distance : The offset distance of the label from each point.

Rotation Angle : The angle of the label relative to each label center point.

Font Name ; Size : Shows the label font name and size. Selecting this label brings up the Font Selector. This label is grey text and does not look like a control.

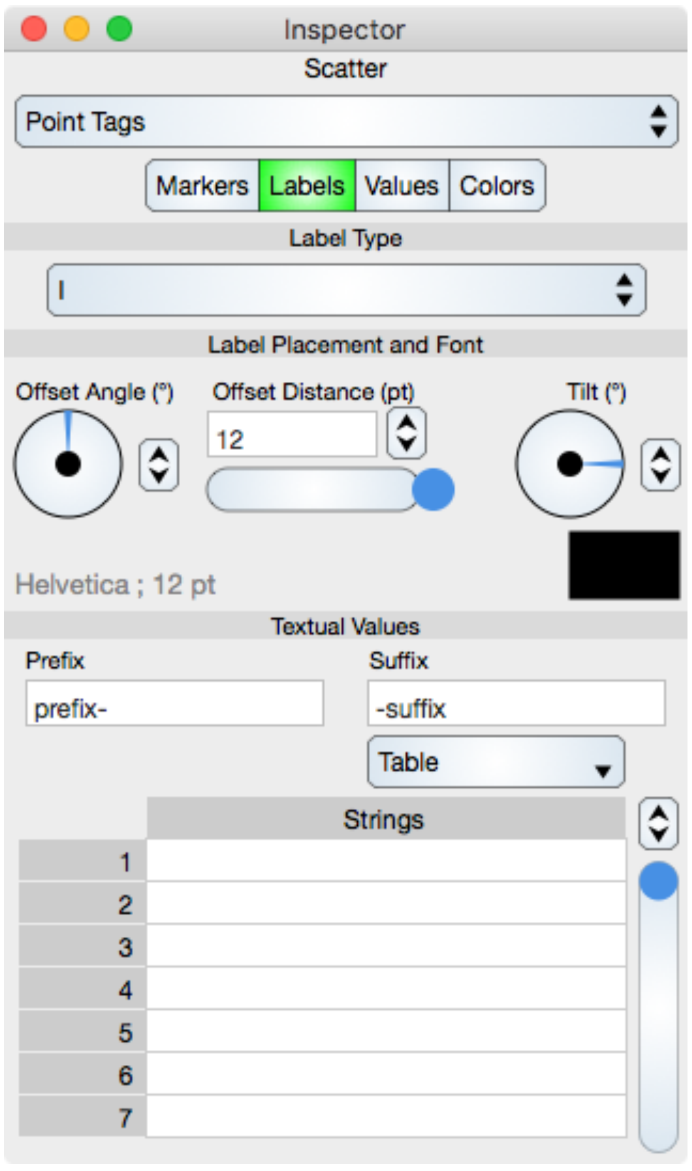
Color : The text Color of the label.

Textual Values

Prefix : The text that is prepended to the start of the label.

Suffix : The text that is appended to the end of the label.

Strings : Table controls are described in the Tables section. The rows represent custom label strings. The table cells are string (textual) elements. Note that if the Data sequence is resorted then the Custom Strings sequence will not be resorted.



Values Editor

Values are numbers that are applied to each point as either a label or a scaling of the marker (for bubble graphs). When scaling, the marker linear mapping factors are applicable. That mapping is defined as follows:

factor_i = scale * (value_i - min)/(max - min) + offset

Factor multiplies the marker's size (width and height). When min and max are the values minimum and maximum values and offset is zero then the factor is between zero and scale, meaning that markers are scaled by the value times scale. For example: If scale is 10 then the markers are made 10 times bigger after multiplying by value. When offset is one then the marker is no smaller than the marker as defined without values applied (with no values entered into the table).

When using the Set Of 2D Points task in the Chart tasks, min and max are the minimum and maximum of all data points, offset is one and scale is a value that is zero or greater.

Values controls are shown and defined below.

Table

Scalars : A Table that shows and edits scalar values. The table cells are scalar (number) elements. Scalars in the table are the values that are applied to each point as either a label or a scaling of the marker. Note that if the Data sequence is resorted then the Values sequence will not be resorted.

Marker Linear Mapping Factors

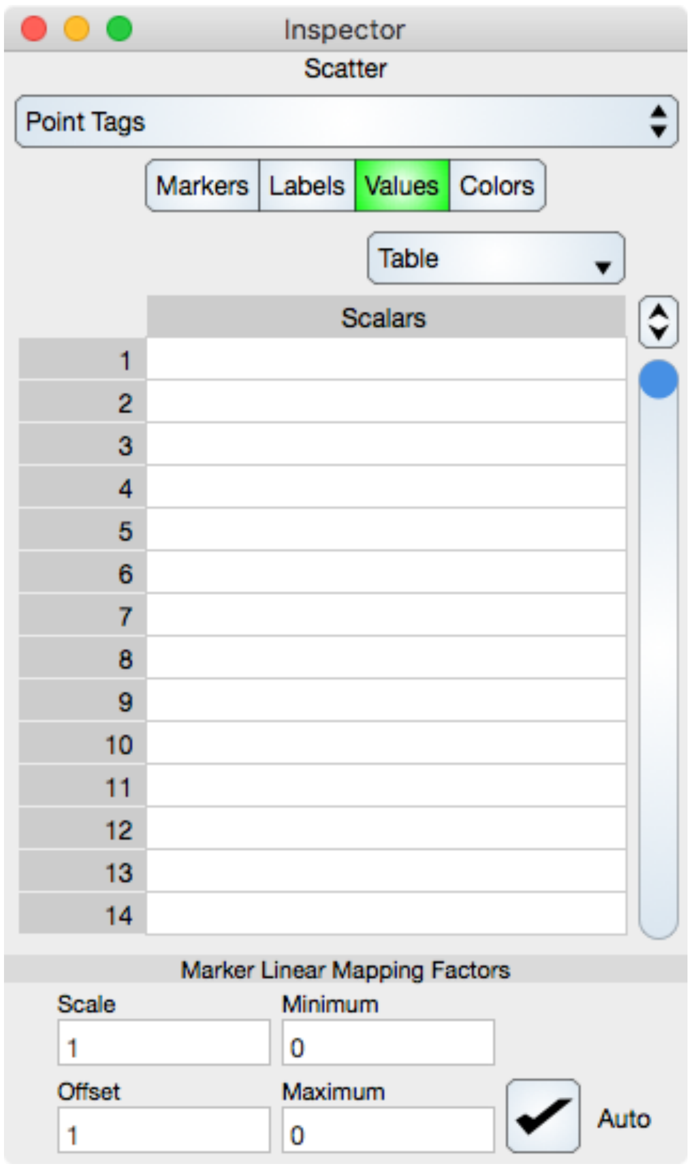
Scale : Scale factor

Offset : Offset factor

Minimum : The minimum value.

Maximum : The maximum value.

Auto : When on, the minimum and maximum are computed from the table entries.



Inspector

Scatter

Point Tags

Markers | Labels | Values | Colors

Table

	Scalars
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

Marker Linear Mapping Factors

Scale: 1

Minimum: 0

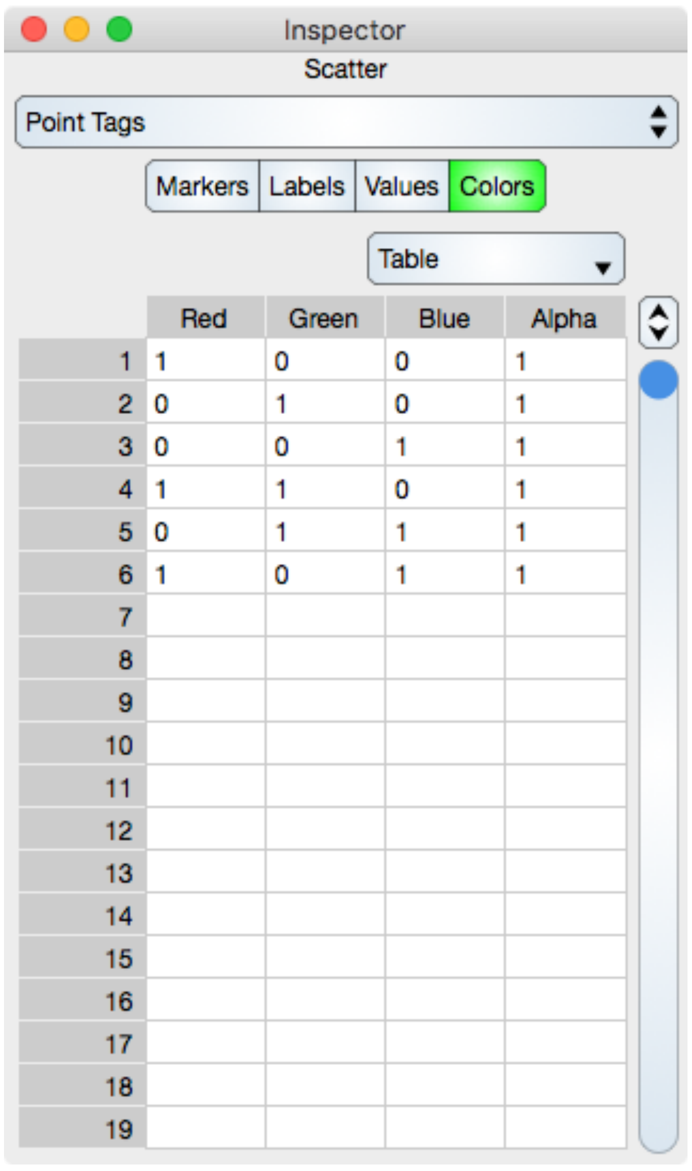
Offset: 1

Maximum: 0

✓ Auto

Colors Editor

Individual markers can have a separate fill color which can be assigned by the colors table shown below.



Inspector

Scatter

Point Tags

Markers | Labels | Values | Colors

Table

	Red	Green	Blue	Alpha
1	1	0	0	1
2	0	1	0	1
3	0	0	1	1
4	1	1	0	1
5	0	1	1	1
6	1	0	1	1
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				

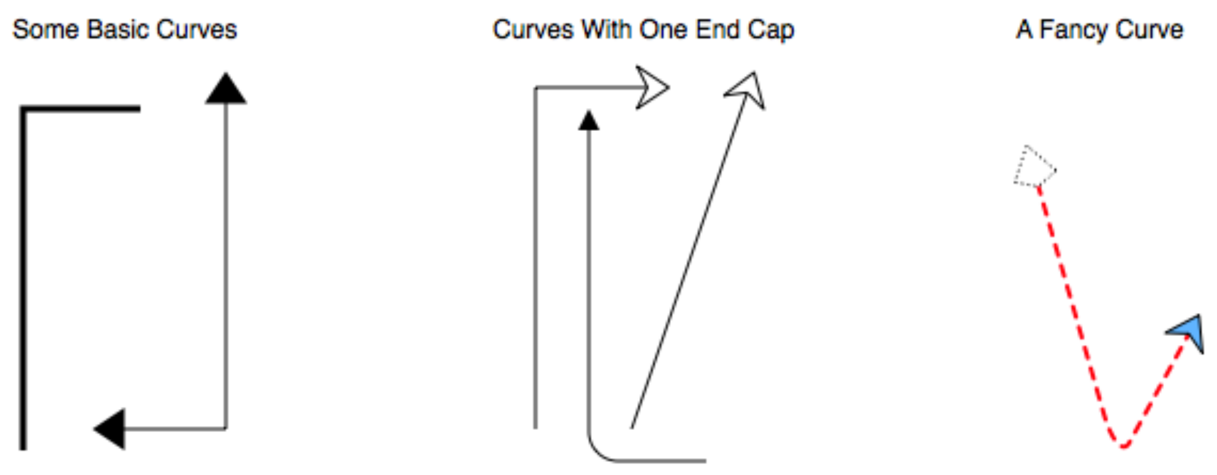
Table

Table controls are described in the Tables section.

The rows represent a marker color values. While in Atomic mode, the cell represents a rgba value and while in component mode the cell represents either single r, g, b and a. Hence, in atomic mode there is one column while in component mode there are four columns. When entering data into each cell make sure to provide four numbers from zero to one which correspond to the red green blue alpha channels of the color. Zero means less color component while one means most color component. Zero alpha means transparent while one alpha means opaque.

Note that if the Data sequence is resorted then the Color sequence will not be resorted.

A Curve graphic can have caps at the end as shown in the following figure.



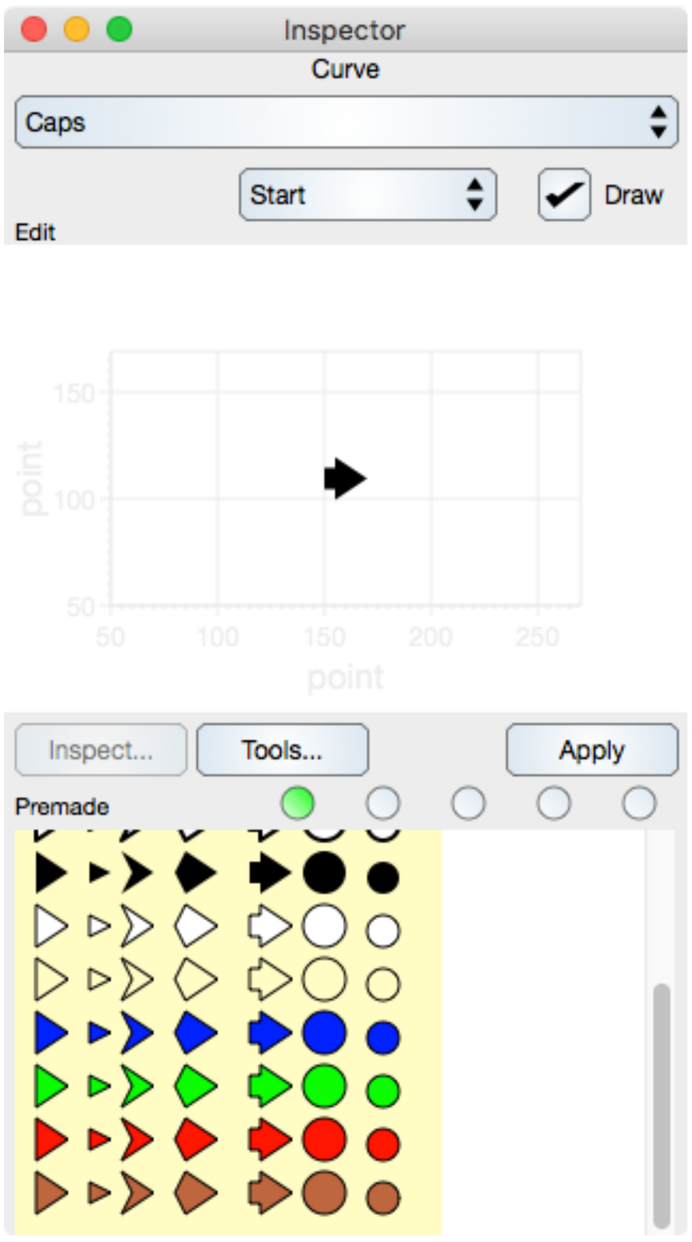
Caps are an arbitrary graphic, such as a circle or square, placed at the end of the curve. A cap can be made from scratch or selected from a pre-made palette of graphics as shown and described below.

Note that the cap maintains its orientation relative to the end point tangent of the curve. The orientation is coincidental with the x > 0 ; y= 0 ray of the cap as shown in the Edit graphic view. There is a bit of heuristics involved in maintaining this orientation.

Also note that while the caps inspector editor is a general editor you should nonetheless limit caps to simple graphics. You could conceivably have caps that are curves with caps (ad infinitum) and caps that are graphs but those are of little practical value and it can get very confusing.

Caps Editor

The Caps Editor is shown below.



Edit

Start Or End : There are two caps associated with a curve, the start cap and the end cap which are at opposite ends of the curve path. Only one cap can be edited at a time and the Start or End pop up button defines which one to show and edit.

Draw : Determines whether to draw the current cap (start or end cap). You can also simply delete the cap graphic if it is never to be drawn.

Edit : The Graphic View from which a cap can be edited. This is a general purpose graphic edit view so all of the tools of Graph IDE are at your disposal while making a cap.

Inspect : Forwards an inspector editor that is applicable to the Edit view. To see controls in the resulting inspector editor for the cap first make sure that you have selected the cap in the Edit view.

Tools : Forwards a Graphic Selector that is applicable to the Edit view.

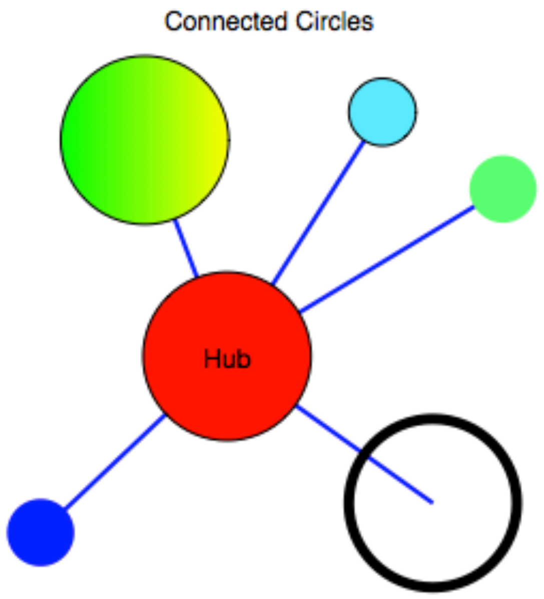
Apply : Applies (copies) the graphic in the Edit view as the cap for the focused graphic.

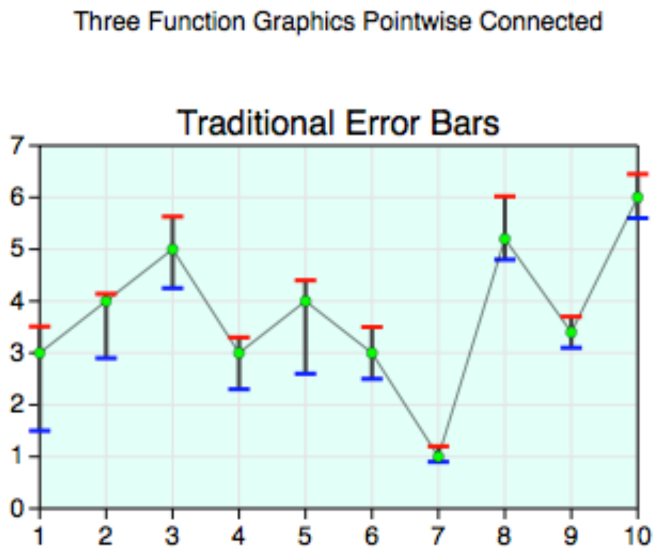
Premade

A group of various premade graphics. Click on a graphic to make it the current cap, or click not-on-a-graphic to remove the cap.

Template Selection : Select one of the radio buttons to see up to five different premade templates.

All Graphics have a network inspector editor that defines relationships (called a Network) between graphics in the same Layer or within the same Group graphic. That inspector is described below.





Connection

The connection is a arbitrary graphic, such as a line or square, placed between graphics that choose to participate in the network. A connection can be made from scratch or selected from a pre-made palette of graphics as shown and described below. Usually the connection (graphic) is a line segment, but any graphic including rectangles, circles and group graphics may be appropriate to use as a connection indicator. The connection terminating points are usually the (geometric) center of the connected graphics but can also be individual points of the graphic if the graphics are Data Graphics.

The connection network inspector editor is shown below. It is important to note that usually only one graphic in a network has a connection graphic (in the figures above that graphic is the connecting line segment), however each graphic in a network can also have its own connection graphic in which case the indicated network has more than one hub and has cycles, etc.

Connection Editor

The Connection Editor is shown below.

Edit

The Graphic View from which the connection can be edited. This is a general purpose graphic edit view so all of the tools of Graph IDE are at your disposal while making a connection.

Inspect : Forwards an inspector editor that is applicable to the Edit view. To see controls in the resulting inspector editor for the connection graphic first make sure that you have selected the connection graphic in the Edit view.

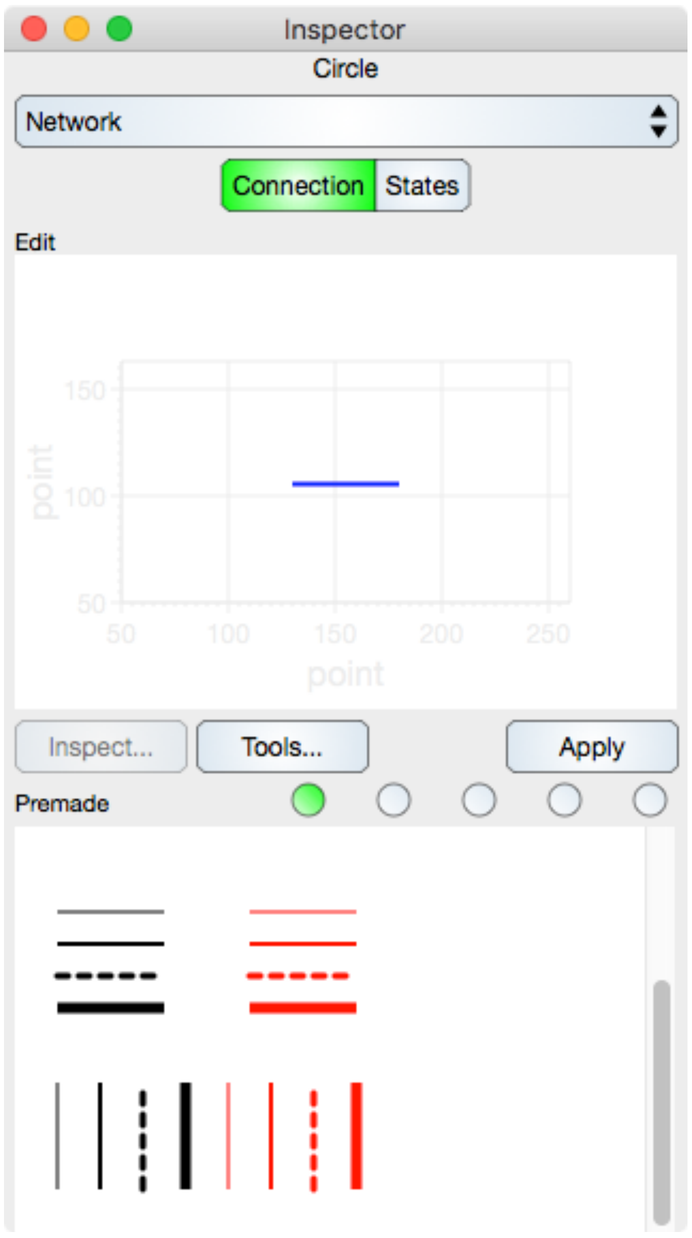
Tools : Forwards a Graphic Selector that is applicable to the Edit view.

Apply : Applies (copies) the graphic in the Edit view as the connection graphic for the focused graphic.

Premade

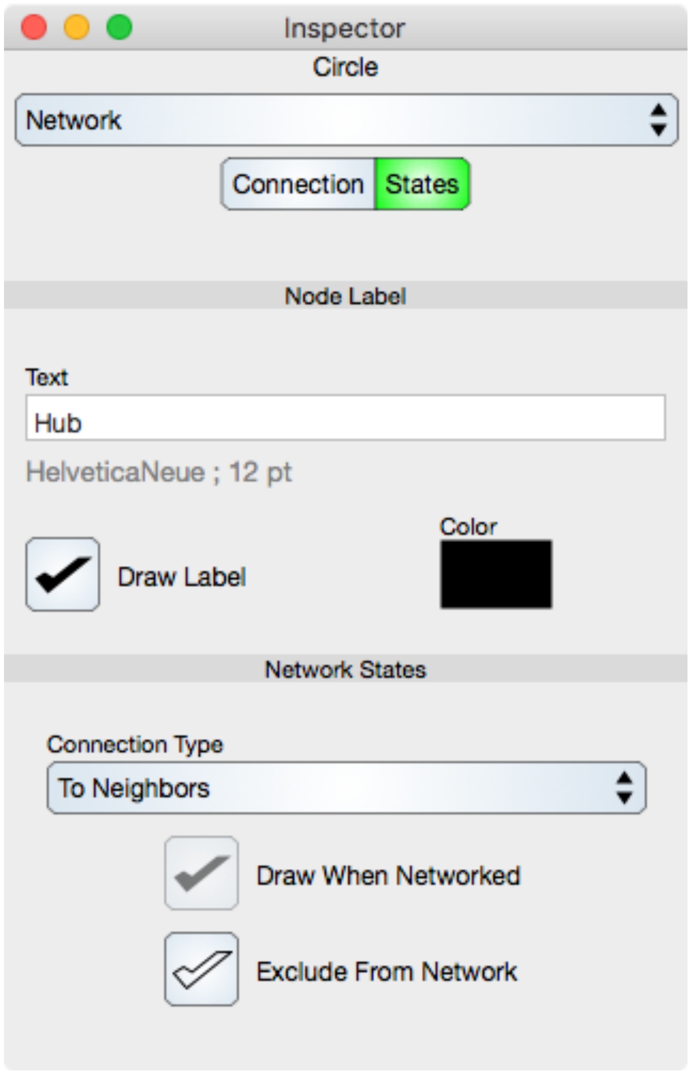
A group of various premade graphics. Click on a graphic to make it the current connection graphic, or click not-on-a-graphic to remove the connection graphic.

Template Selection : Select one of the radio buttons to see up to five different premade templates.



States Editor

The States Editor is shown below.



Text: Hub

HelveticaNeue ; 12 pt

Draw Label: ✓

Color: ■

Connection Type: To Neighbors

Draw When Networked: ✓

Exclude From Network

Node Label

A Label is a single-line textual graphic placed at the midpoint of the graphic. A label has attributes as described below. It is important to note that each graphic in the network can have a label (and not just the network hub graphic).

Text : This is the text that will appear at the center of the graphic. Click Return to enter the label string.

Font Name; Size : The font for the label text. Select this text to bring forward the Font Selector.

Draw Label : When selected, the label is drawn.

Color : The color for the label text. Select this color well to bring forward the Color Selector.

Network States

Connection Type : The type of connection as described in the table below and also the Network section.

Draw When Networked : For now, this is always on and the control is disabled.

Exclude From Network : When selected the graphic is excluded from any possible network connection from another graphic.

Connection Type	Description

No Connections	No connection graphic is drawn.
To Neighbors	Draw the connection graphic from the graphic to all neighbors in the layer. In this context a neighbor is any graphic in the layer.
Between Neighbors	Draw the connection graphic from the graphic between all neighbors in the layer. In this context a neighbor is any graphic in the layer. This is the same as To Neighbors.
To Neighbor Points	Draw the connection graphic from the graphic points to the nearest neighbors in the layer. In this context a neighbor is the next and previous graphics in the sequence of graphics in the layer. This only pertains to Data Graphics.
Between Neighbor Points	Draw the connection graphic between the nearest neighbors in the layer. In this context a neighbor is the next and previous graphics in the sequence of graphics in the layer. This only pertains to Data Graphics.
Vertical Neighbor Points	Draw the connection graphic from the graphic points to the nearest neighbors in the layer while keeping the connection graphic vertical. In this context a neighbor is the next and previous graphics in the sequence of graphics in the layer. This only pertains to Data Graphics, specifically the Function.
Horizontal Neighbor Points	Draw the connection graphic from the graphic points to the nearest neighbors in the layer while keeping the connection graphic horizontal. In this context a neighbor is the next and previous graphics in the sequence of graphics in the layer. This only pertains to Data Graphics, specifically the Trajectory.

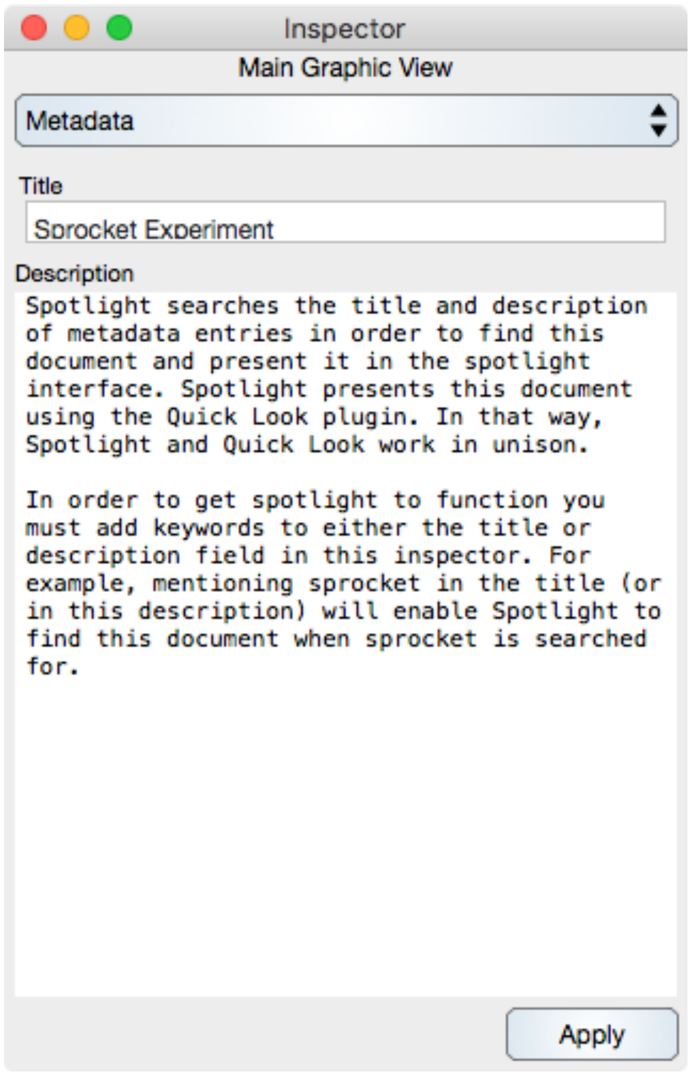
For different uses of the connection type and other network features see the Network section.

Graph IDE ▶ Inspector Editors ▶ Metadata

Metadata is used by Spotlight to search for documents (see Quick Look & Spotlight). Spotlight searches the title and description of metadata entries in order to find a document and present it in the spotlight interface. Spotlight presents the document using the Quick Look plugin. In that way, Spotlight and Quick Look work in unison.

In order to get spotlight to function you must add keywords to either the title or description field in the Metadata inspector. For example, mentioning sprocket in the title (or in the description) will enable Spotlight to find the document when sprocket is searched for.

The Metadata inspector editor is shown below.



Title: Sprocket Experiment

Description:

```
Spotlight searches the title and description
of metadata entries in order to find this
document and present it in the spotlight
interface. Spotlight presents this document
using the Quick Look plugin. In that way,
Spotlight and Quick Look work in unison.

In order to get spotlight to function you
must add keywords to either the title or
description field in this inspector. For
example, mentioning sprocket in the title (or
in this description) will enable Spotlight to
find this document when sprocket is searched
for.
```

Table

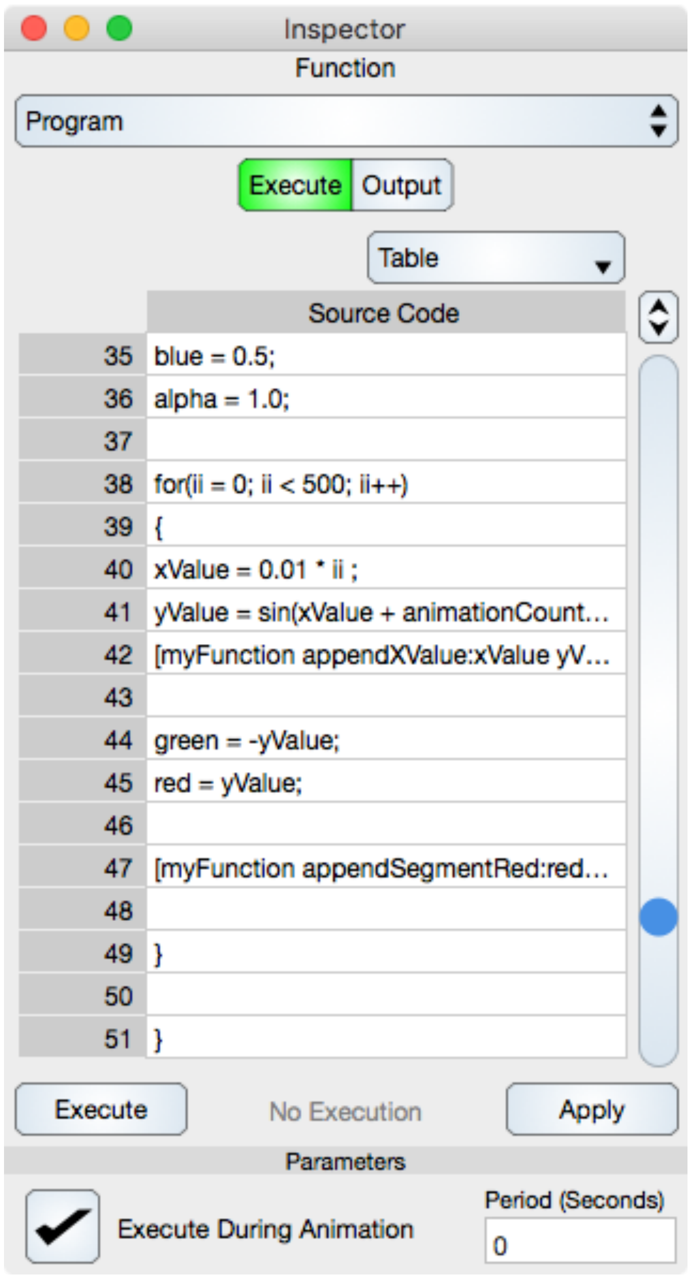
Title	The document title.
Description	The document description.
Apply	Apply the metadata and then when the document is saved the document becomes searchable by Spotlight (see Quick Look & Spotlight).

Graph IDE ► Inspector Editors ► Program

The Program inspector editor is where programs are entered and applied. Programming is described in the Programming section. This section simply describes how to apply the program.

Execute Editor

Program source code is entered and executed as follows.



```
blue = 0.5;
alpha = 1.0;

for(ii = 0; ii < 500; ii++)
{
xValue = 0.01 * ii ;
yValue = sin(xValue + animationCount…
[myFunction appendXValue:xValue yV…

green = -yValue;
red = yValue;

[myFunction appendSegmentRed:red…

}

}
```

Table

Source Code : A Table that shows and edits the program source code. The table cells are program lines, normally statements. The Programming section gives examples of source code. Once entered then you must at least Apply it or your entry is lost. For the most part, you should type source code in a separate text editor, copy it and paste it into the Source Code table as the table is not a full IDE.

Execute : Selecting the Execute button will parse and execute the source code. As a side effect, it will also apply (save) the source code.

Apply : Selecting the Apply button will save the source code but will not parse or execute it.

Information : Shows the number of times the program was executed. The information text appears between the Execute and Apply buttons.

Parameters

Execute During Animation : Turn this switch on if the source code needs to be executed during Graphic View animation. Most likely you should always turn this on.

Period : Defines the minimum period that the source code is executed during animation. A value of zero (default) means to execute the program on every animation step. Normally zero should be good enough since graphics will probably animate with a single period.

Output Editor

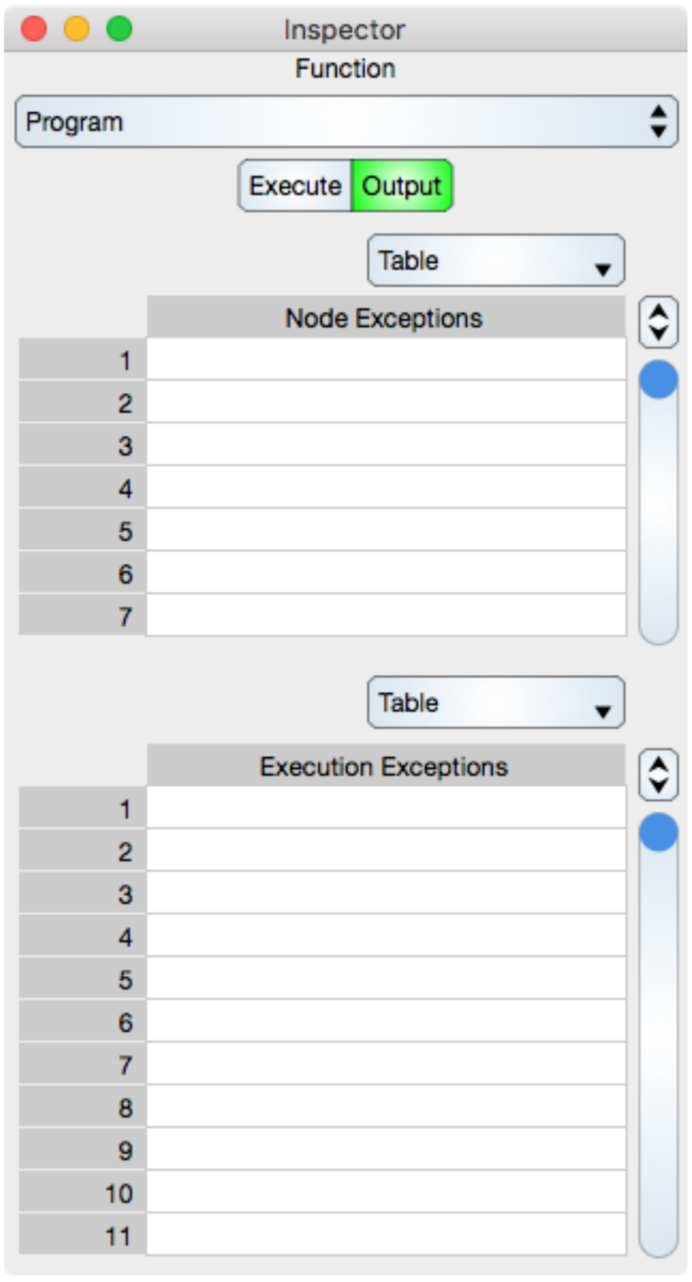
Output of the program is shown in the Output editor.

Table

Standard Output : The result of fprintf(stdout,) function is shown here. In fact, any output to the stdout stream is shown in this table. The stdout stream is only redirected to this table when the Execute button is clicked. During animation the output goes to the system console.

Standard Error : The result of fprintf(stderr,) function is shown here. In fact, any output to the stderr stream is shown in this table. The stderr stream is only redirected to this table when the Execute button is clicked. During animation the output goes to the system console.

Other messages, not associated with unix streams, may appear in either the output or error tables. Windows does not support stream redirection. If your program does not generate the intended results upon selecting the Execute button then consult these tables for explanations.



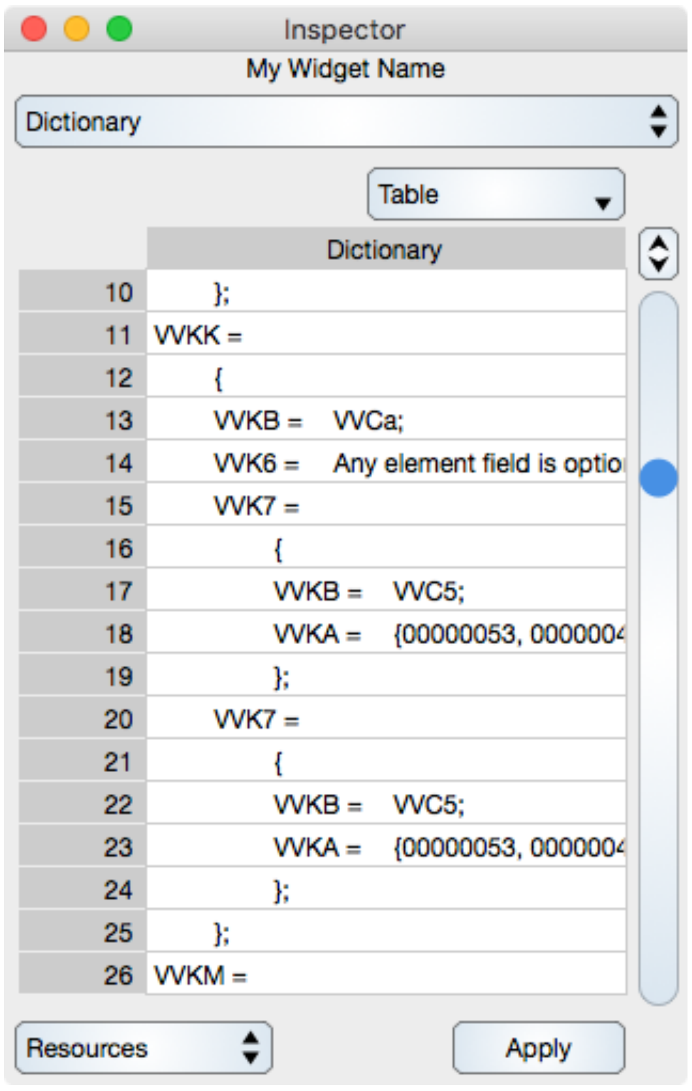
	Node Exceptions
1	
2	
3	
4	
5	
6	
7	

	Execution Exceptions
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	

Graph IDE ▶ Inspector Editors ▶ Dictionary

A Dictionary is something that is only used for internal purposes so this section can be safely ignored. There is a separate few thousand page reference guide that accompanies general programming API that is used in conjunction with dictionaries and that is completely separate from this manual.

The Inspector Editor for the dictionary is shown below.



	Dictionary
10	};
11	VVKK =
12	{
13	VVKB = VVCa;
14	VVK6 = Any element field is optio
15	VVK7 =
16	{
17	VVKB = VVC5;
18	VVKA = {00000053, 000000
19	};
20	VVK7 =
21	{
22	VVKB = VVC5;
23	VVKA = {00000053, 000000
24	};
25	};
26	VVKM =

Table

Dictionary : A Table that shows the dictionary content serialized as a UTF-8 based string.

Surrogate : A pop up button that selects the surrogate dictionary to show in the Table.

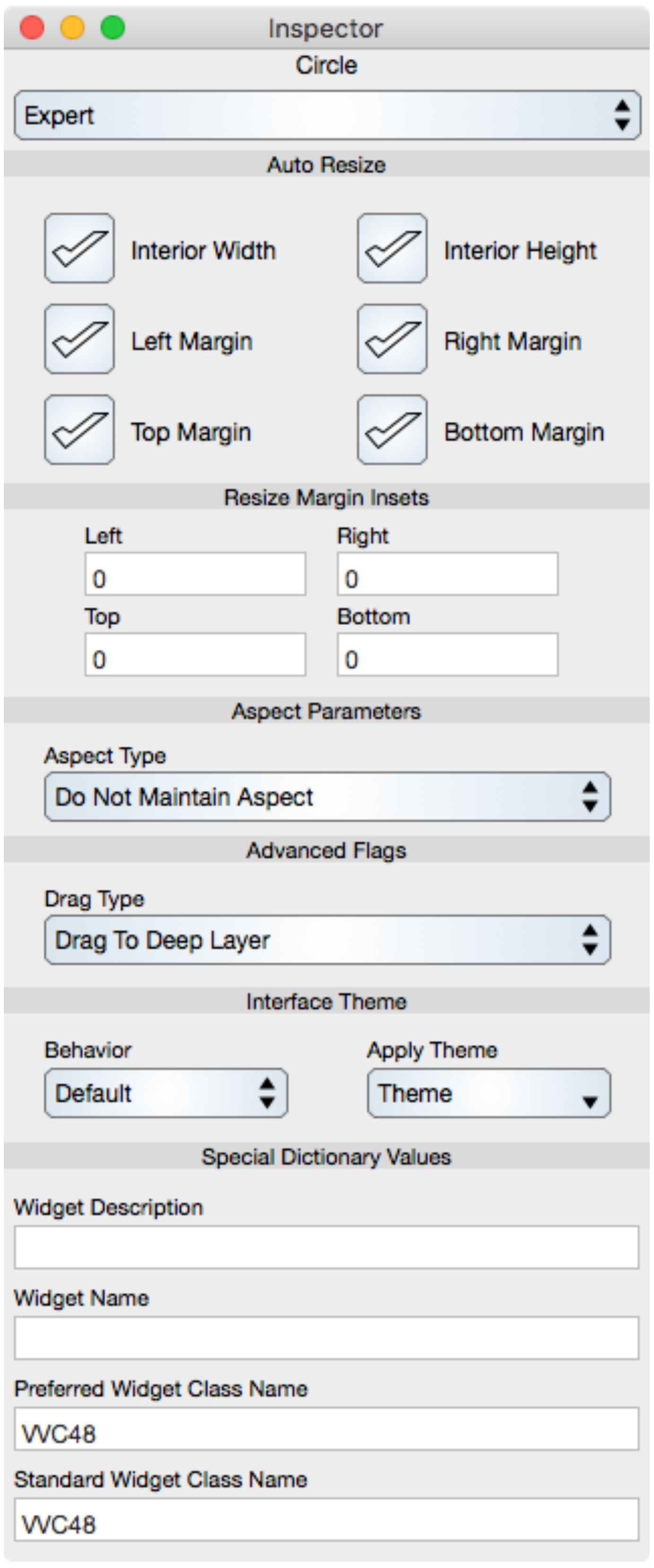
Apply : Selecting the Apply button will save the dictionary entries.

Note that the dictionary entries as well as all serialized archives are hashed in order to reduce size and make computation more efficient so that the ability to edit non-user entries is limited unless the hashing is inverted (as is only the case in special distributions).

The Expert Attributes are somewhat esoteric attributes that have to do with template building and programming.

Inspector Editor

The Inspector Editor for the expert attributes is shown below.



Inspector
Circle
Expert

Auto Resize
- [✓] Interior Width
- [✓] Interior Height
- [✓] Left Margin
- [✓] Right Margin
- [✓] Top Margin
- [✓] Bottom Margin

Resize Margin Insets
Left: 0
Right: 0
Top: 0
Bottom: 0

Aspect Parameters
Aspect Type: Do Not Maintain Aspect

Advanced Flags
Drag Type: Drag To Deep Layer

Interface Theme
Behavior: Default
Apply Theme: Theme

Special Dictionary Values
Widget Description:
Widget Name:
Preferred Widget Class Name: WC48
Standard Widget Class Name: WC48

Auto Resize

Determines the way that the graphic resizes when the view it is in resizes. To see this effect the document window needs to be a non-scrollview type.

Resize is proportional to the graphic view size change.

Interior Width : When selected, the width of the graphic resizes.

Interior Height : When selected, the height of the graphic resizes.

Left Margin : When selected, the left margin of the graphic resizes.

Right Margin : When selected, the right margin of the graphic resizes.

Top Margin : When selected, the top margin of the graphic resizes.

Bottom Margin : When selected, the bottom margin of the graphic resizes.

Resizing is relative to the reference bounds of the graphic and not its overall graphical bounds. For example, the reference bounds of a graph is its frame, the reference bounds of a 3d graph is its projection on the projection plane. The reference bounds of a circle is the graphical bounds of the circle when the stroke width is zero and if the circle is transformed or has a pie section then the reference bounds is the affine transformation of its normal form reference rectangle. In addition, an aligned group can have incremental resizing which means that the resizing is more complex and based upon a backstore frame, i.e.: resizing can be complex.

Resize Margin Insets

Normally, resize is determined from the graphic's reference bounds. The margin insets are used to offset that reference bounds for resizing purposes.

Left : The left side offset.

Right : The right side offset.

Top : The top side offset.

Bottom : The bottom side offset.

Aspect Parameters

Aspect Type : The default is to not maintain aspect. Setting this to another value will maintain the aspect of the graphic.

Advanced Flags

Drag Type : If the graphic is placed on a palette then this setting determines how that graphic responds to dragging. The default is to drag to the deep Layer which is the currently focused layer. However, it can also be set to drag to the overlay layer which is the topmost layer. Data graphics should drag to the deep layer (the data layer of a graph) while things like a graph should never drag to another graph's data layer and should rather drag to the topmost layer. If the graphic to drag is also a Group then ungrouping style can also be defined as ungroup with no select, select all or select first graphic.

Interface Theme

Theme settings is an advanced feature and probably not applicable to your use. However, if you are formatting a document for a particular interface theme then the theme settings can be used to transform document components to that interface theme.

Behavior : If Default then the graphic transforms according to the interface theme. If None then the graphic does not transform upon theme change. If Stroke Only then the graphic transforms the stroke attribute only.

Apply Theme : Select the theme type from the drop down to transform the graphic immediately regardless of the current theme. If None is chosen then the theme attributes are reset and a subsequent theme transformation will utilize the reset attributes when needed (for original and flat themes).

Special Dictionary Values

Description : This is a string that appears in the Cursor Information window and is also used for Legend labels.

Name : This is a string that is used for template processing in the Chart Tasks (skins) and Vvidget Code programming.

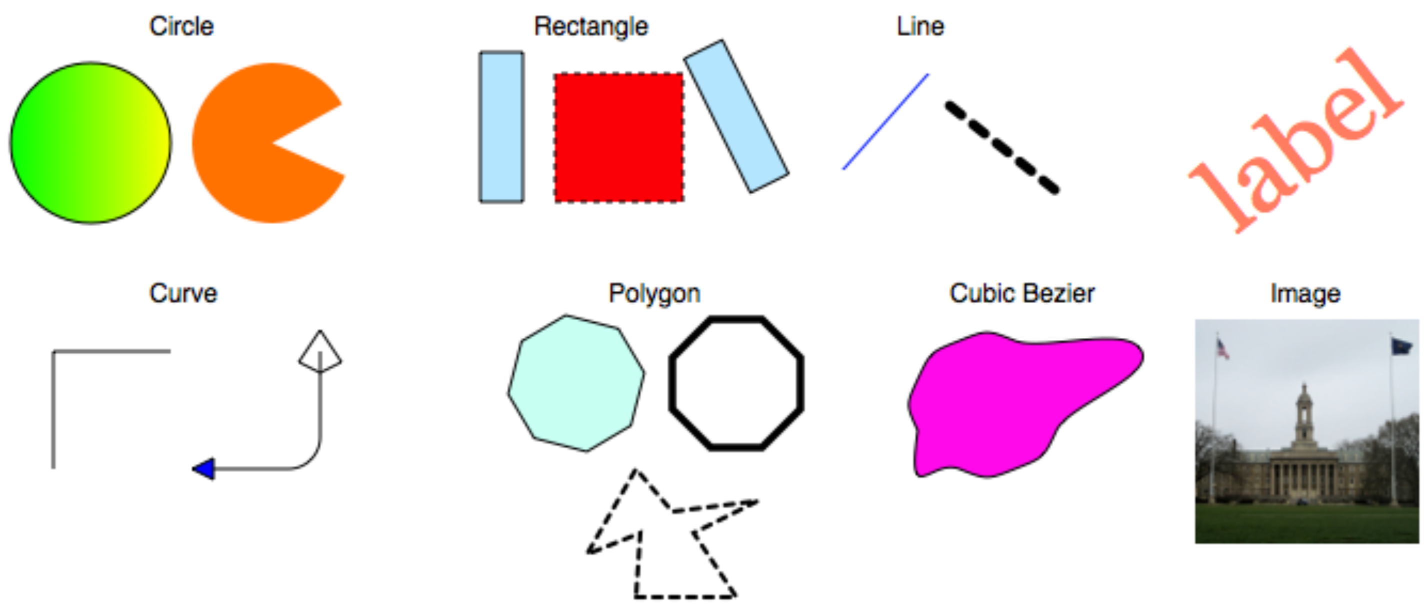
Preferred Class Name : This is used to intercept archive decoding and is an advanced feature. Do not edit this field.

Standard Class Name : This shows the standard class name of the selected graphic. This name can not be changed.

Graph IDE ► Basic Graphics

A basic graphic is a graphic which has a single major graphic property and also has draw-related attributes. The exception is a Group graphic, which can be a collection of any type of graphic, but for which does not have any major graphic property apart from those of its elements.

The figure below shows examples of a basic graphics.

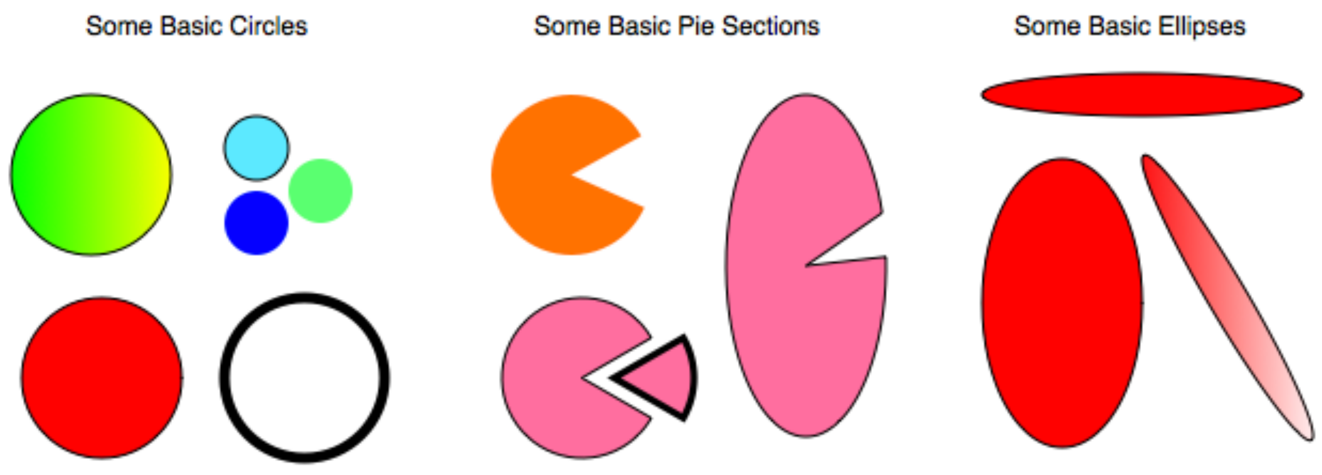


The following is a brief list and definition of basic graphics:

Section	Description
Circle	A circle which can be transformed into an ellipse and also has associated attributes, such as wedge angles and graphic attributes.
Cubic Bezier	A Cubic Bezier is a sequence of connected Cubic Bezier sections to form an open or closed curve.
Curve	A Curve is either an line, elbow, or curved elbow with end caps and can be used for pointing to something.
Group	A Group is a collection of graphics.
Image	An Image is a raster or vector representation of a graphic.
Label	A Label is a single line of text all of the same font and color.
Line	A Line is a line segment.
Path	A sequence of operations and operands which define a drawn figure.
Polygon	A Polygon is a sequence of connected line segments to form an open or closed polygon or other related shape.
Rectangle	A rectangle which can be transformed into a parallelogram and also has associated attributes, such as corner radius to produce an oval, and graphic attributes.

Graph IDE ► Basic Graphics ► Circle

A circle is defined as a closed curve of constant distance from a point. It can be transformed into an ellipse and also has associated attributes, such as wedge angles and graphic attributes. The figure below shows some examples of a circle.



Circles are useful in their own right and also used for Point Tags markers and Pie Chart wedge sections.

Some standard operations are itemized below.

- To create a circle bring forward the Graphic Selector, click the circle factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of a normal-form ellipse. If the width and height are the same then it is a circle.
- You can also create circles from the standard Palettes ► Basics ► Circles menu item or drag them out from the circle Factory Inspector.
- Resizing and rotating the circle transforms it to an ellipse. Defining different start or end wedge angles transforms the circle (or ellipse) into a pie-section. For more information see Standard Editing.
- To program a circle see the Programming section.

Inspector Editor

The Inspector Editor for the circle is shown below.

Menu Shortcuts

Menu shortcuts are common to all graphics and are described in the Graphics section.

Pie Section Parameters

Start Wedge Angle : The start angle, from the y = 0, x > 0 line, of the wedge section. Note: The wedge section is not the empty area but rather the remaining circle part. The dial control is described in the Dial section.

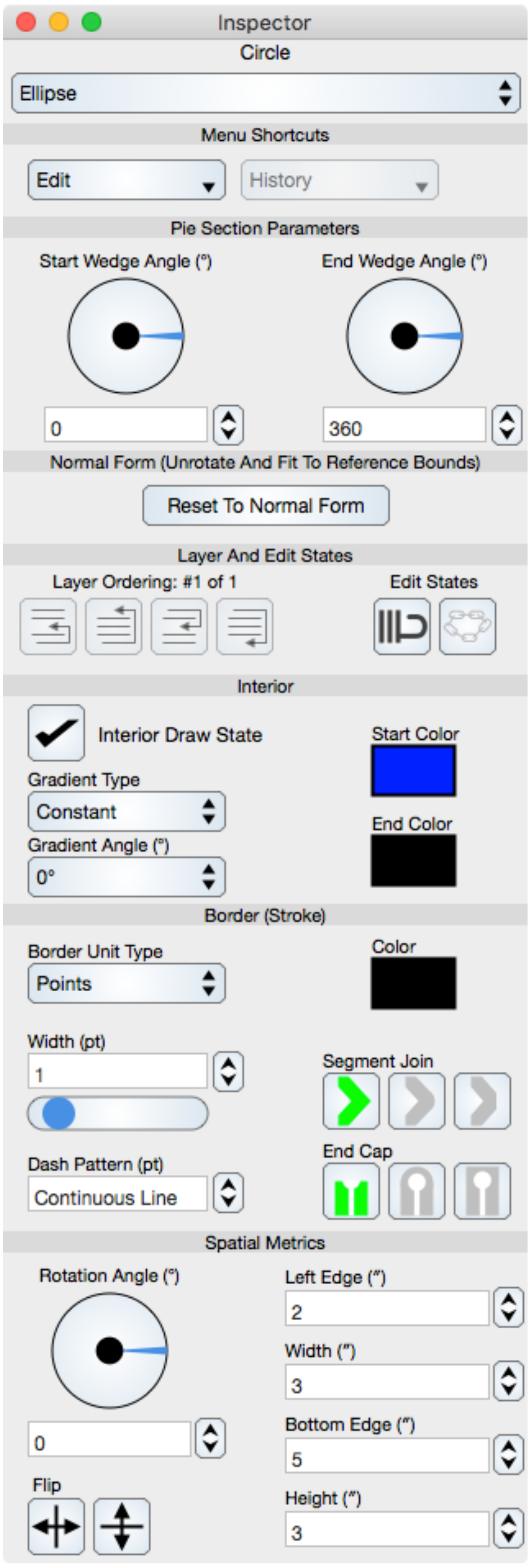
End Wedge Angle : The end angle, from the y = 0, x > 0 line, of the wedge section. Note: The wedge section is not the empty area but rather the remaining circle part. The dial control is described in the Dial section.

Normal Form

Reset To Normal Form : Select this to set the width and height of the ellipse to the reference frame of the graphic.

Common Controls

Other controls common to all graphics are described in the Graphics section.


Inspector

Circle

Ellipse

Menu Shortcuts

Edit | History

Pie Section Parameters

Start Wedge Angle (°): 0

End Wedge Angle (°): 360

Normal Form (Unrotate And Fit To Reference Bounds)

Reset To Normal Form

Layer And Edit States

Layer Ordering: #1 of 1

Edit States

Interior

✓ Interior Draw State

Start Color

Gradient Type: Constant

End Color

Gradient Angle (°): 0°

Border (Stroke)

Border Unit Type: Points

Color

Width (pt): 1

Segment Join

Dash Pattern (pt): Continuous Line

End Cap

Spatial Metrics

Rotation Angle (°): 0

Left Edge ("): 2

Width ("): 3

Bottom Edge ("): 5

Height ("): 3

Flip

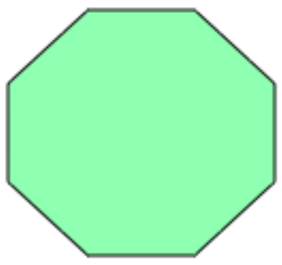
Graph IDE ► Basic Graphics ► Cubic Bezier

A Cubic Bezier graphic is a connected sequence of Cubic Bezier sections. Another way to put that is that given any two consecutive points in a sequence there is a curve that connects those two points and that curve is defined by a third-order parametric equation. To define the coefficients of that equation two artificial control points have to be introduced. Those points are predefined as the ends of tangent lines from the end points of the curve segment in question. In fact any control points can be used, but ones that define tangent lines are not only conventional but they also make sense. So, with that in mind and now that I told you what the definition of a Cubic Bezier graphic is I will now change the definition slightly to this:

A Cubic Bezier graphic is a connected sequence of points where each point in that sequence has two tangent lines drawn from it on either side to define the slope of a curve intersecting that point. One definition is segment-oriented the other point-oriented. The point-oriented definition is easier to deal with in many ways.

The figures below show examples of Cubic Bezier graphics.

Three Cubic Bezier Sections


Spline Knots Over Vertex Knots

Composite Cubic Bezier Graphic

Some standard operations are itemized below.

- To create a Cubic Bezier bring forward the Graphic Selector, click the Cubic Bezier factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of a predefined Cubic Bezier graphic.
- You can also create Cubic Bezier graphics from the standard Palettes ► Art ► or Palettes ► Maps ► menu items or drag them out from the Cubic Bezier Factory Inspector.
- Resizing and rotating the Cubic Bezier transforms all points simultaneously. For more information see Standard Editing.
- One thing you will probably want to do with a Cubic Bezier is modify its points. You can do that via the Data or parser Inspector Editor or by editing it directly with the mouse controls. Initially, the spline knots (tangent end points) are under the vertex points so you will not be able to get at them by clicking on them. You should first smooth out the graphic so that the control knots move away from the vertex points and then enter the mouse edit mode and move those knots as you wish. See Standard Editing for information on point editing.
- To program a Cubic Bezier see the Programming section.

Data Editor

The Data Editor for the Cubic Bezier is shown below.

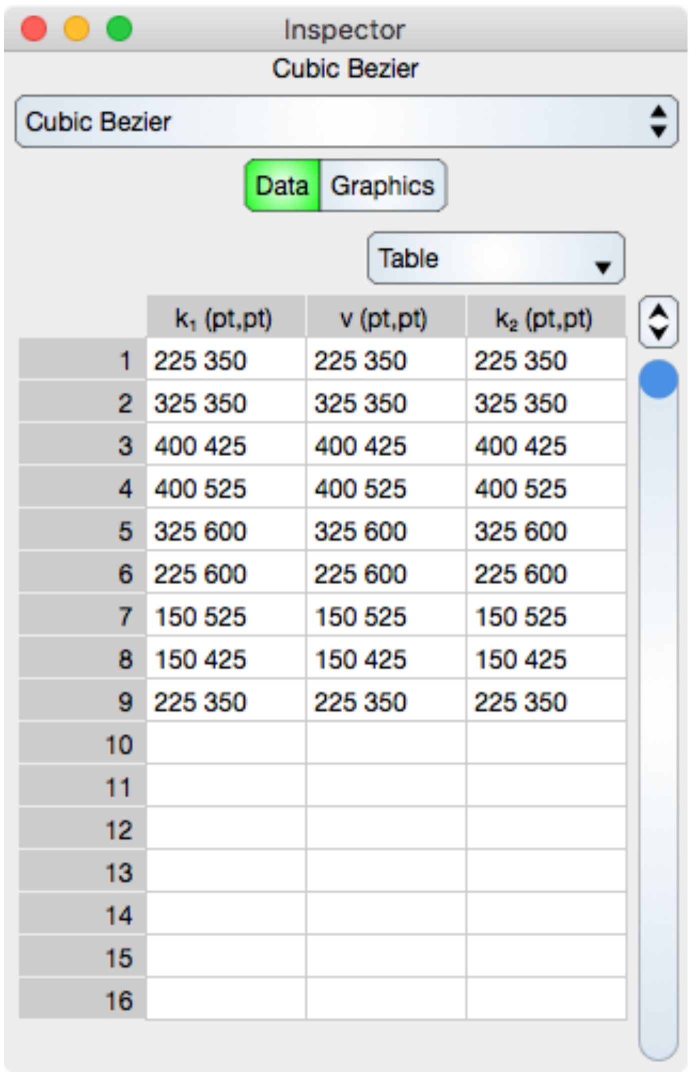
Table

Table controls are described in the Tables section.

The rows represent knot and vertex values at a particular vertex. While in Atomic mode, the cell represents a point (x and y value) and while in component mode the cell represents either a x or y value. Hence, in atomic mode there are 3 columns while in component mode there are six columns.

Notice that the table is point wise and not parametric-section wise. The first and last knots are non-functioning since they would be associated with a section outside the parametric-sections, however they are included in the dataset to make it uniform.

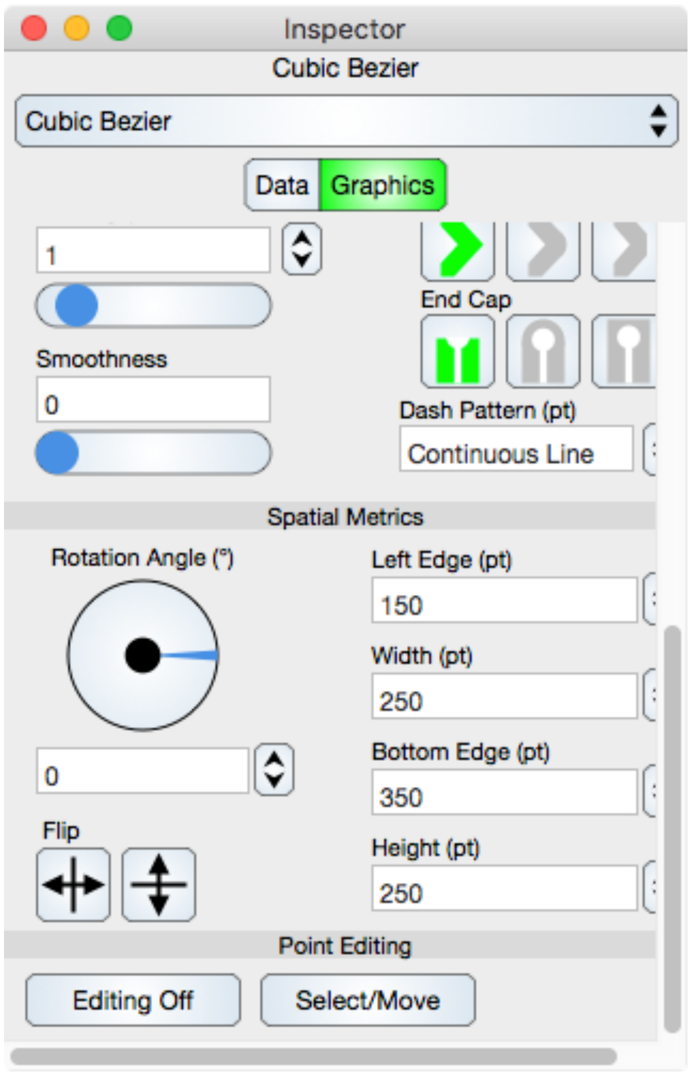
The units of the data values are in the unit of the coordinate system. For a Graphic View those units are always typographical points (and not the report units) while for Graphs the units are either user assigned or unitless.



	k_1 (pt,pt)	v (pt,pt)	k_2 (pt,pt)
1	225 350	225 350	225 350
2	325 350	325 350	325 350
3	400 425	400 425	400 425
4	400 525	400 525	400 525
5	325 600	325 600	325 600
6	225 600	225 600	225 600
7	150 525	150 525	150 525
8	150 425	150 425	150 425
9	225 350	225 350	225 350
10			
11			
12			
13			
14			
15			
16			

Graphics Editor

The Graphics Editor for the Cubic Bezier is shown below.



Common Controls

Other controls common to all graphics are described in the Graphics section.

Cubic Bezier Specific Controls

Smoothness : Adjust the spline knots of the Cubic Bezier so that the tangent lines at a vertex are more coincidental. Zero moves the knot locations under the vertex location while one is maximum smoothness. Choosing a smoothness greater than zero is a good way to expose the knots for point editing.

Point Editing

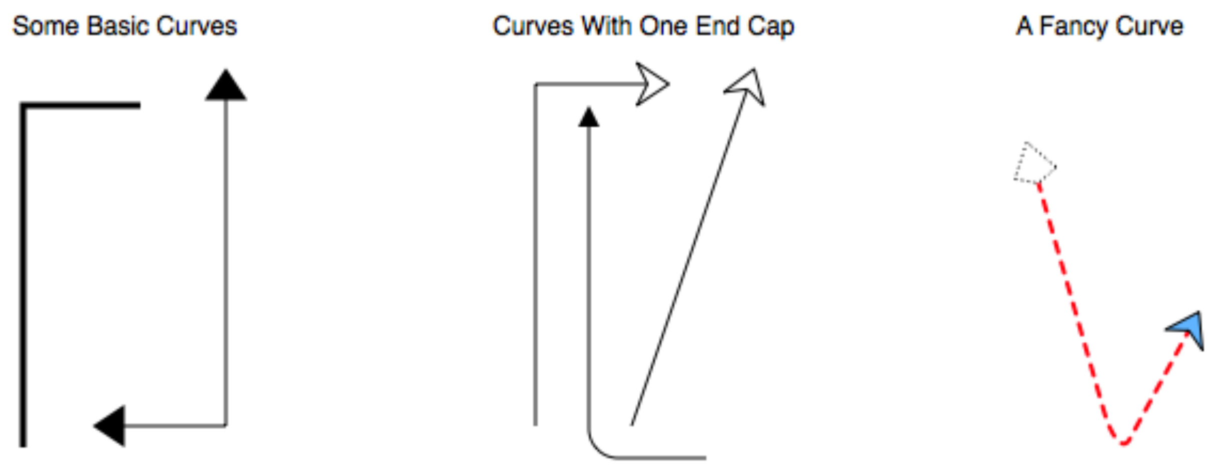
Editing Off/On : Places the graphic into or out of edit mode. While in edit mode the vertices and knots are shown by indicators and can be adjusted using mouse or touch events. Double-clicking the graphic also toggles this edit mode.

Select/Move or Add/Delete : Select/Move mode permits the knot and vertex editing to select and move those locations while Add/Delete mode will delete a knot or vertex if they are hit or add a knot and vertex (triplet) if a cubic bezier segment is hit. This can also be accomplished using the shift key if available.

Table Of Contents

Graph IDE ▶ Basic Graphics ▶ Curve

A curve is defined as a line, an elbow or a curved elbow with end Caps. It also has associated attributes, such as corner radius and other graphic attributes. The figure below shows some examples of a curve.

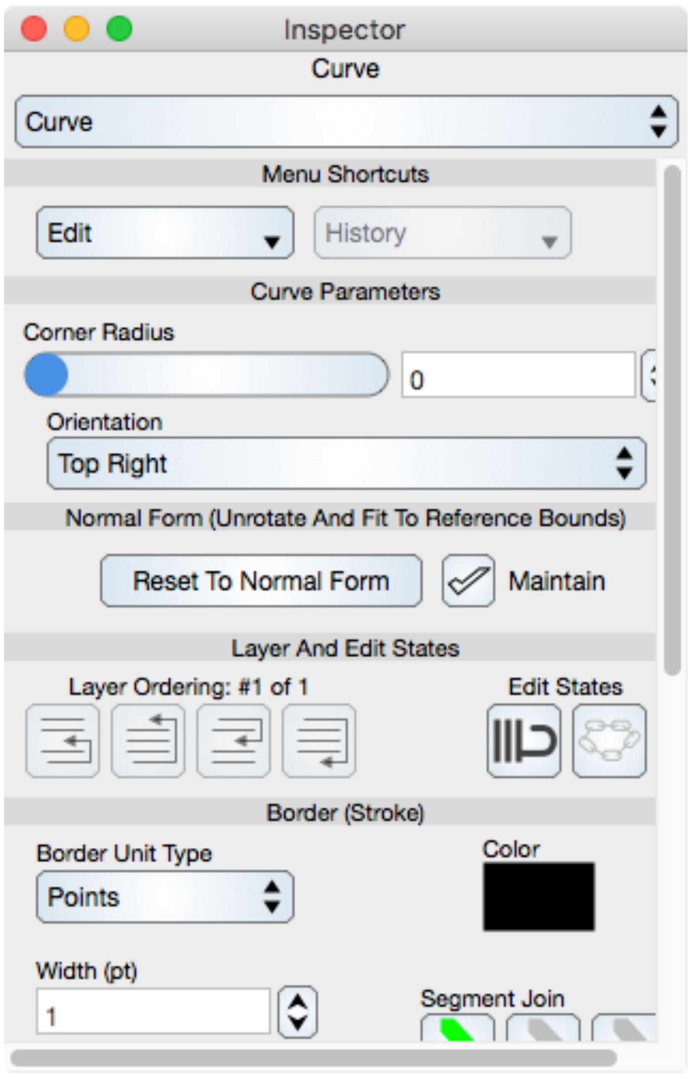


Some standard operations are itemized below.

- To create a curve bring forward the Graphic Selector, select the curve factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of a curve.
- You can also create curves from the standard Palettes ▶ Basics ▶ Curves menu item or drag them out from the curve Factory Inspector.
- For more information see Standard Editing.
- To add or modify the caps graphics see the Caps inspector editor.

Inspector Editor

The Inspector Editor for the curve is shown below.



Menu Shortcuts

Menu shortcuts are common to all graphics and are described in the Graphics section.

Curve Parameters

Corner Radius : Defines the radius of curvature of the corner of the L-shaped curve. When the radius is zero then the curve looks like a straight L. When the radius is non-zero then the L shape's corner becomes a section of a circle.

Orientation : Defines where the angle of the L shape is, or no angle for a straight line along the diagonal.

Normal Form

Reset To Normal Form : Select this to set the width and height of the curve to the reference frame of the graphic.

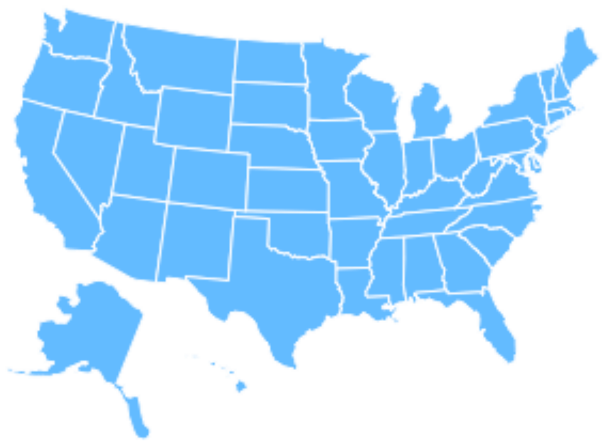
Common Controls

Other controls common to all graphics are described in the Graphics section.

A group is a collection of graphics which have been collected together to form a new indivisible graphic called a group. The following figure shows some examples of groups.

A Group Of One Cubic Bezier And Two Ovals

A Group of 52 Cubic Beziers Plus A Few Islands



Groups are convenient because they are indivisible which means you can work with all the graphics in a group at the same time. For example, when you grab a group off a palette instead of each graphic individually.

Some standard operations are itemized below.

- To create a group select the graphics you want to group and use the menu item Editor ► Group. To ungroup select the group and use the menu item Editor ► Ungroup.
- You can also create groups from the standard Palettes menu items. There are several groups there. Premade groups can also be dragged out from the group Factory Inspector.
- For more information on editing see Standard Editing.

Arranger

The secondary purpose of a group is to provide an algorithm to place the group elements. The default algorithm is a matrix or table alignment as specified in the Alignment section. When the group has a context then the alignment is called an arranger. The Pie Chart and Bar And Column Chart are examples of a different algorithm to align elements of a group.

Inspector Editor

The Inspector Editor for the group is shown below.

Menu Shortcuts

The Edit and History shortcuts are common to all graphics and are described in the Graphics section.

Sort : Sorting the group means to arrange the sequence index of the group elements against another dimension. That dimension is either Natural or Y-Descending. Natural means the way text is read (from top to bottom) and in the case of intersecting elements then the element with greater area has a lesser sequence index. This is consistent with backdrops and also tabbed responder order. The ordering is most important when order refers to implicit functions such as tabbing in HTML. The Y-Descending sorting is appropriate for cumulative area graphs. Care must be taken to not sort general groups for which the sequence index order provides for a z-buffer effect.

Group Parameters

Graphics In Group : Shows the number of graphics in the group.

Group Main Title : Any group can have a title which is placed over the group (at the y-maximum of the group). The title is a Label which can not be directly edited but can be selected and edited via the Navigator. The title is important for the Pie Chart and Spreadsheet. In general, a title is desired for any named group. The title is not an element of the group but is rather a separate element so if the group is ungrouped then the title is permanently lost.

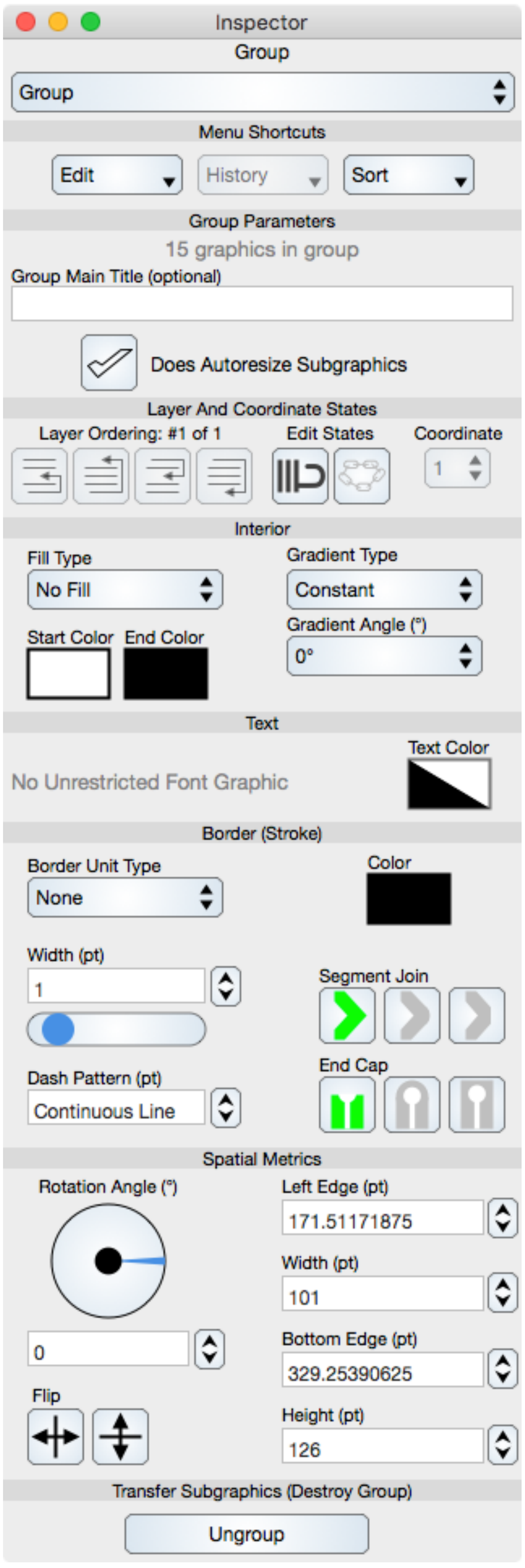
Does Autoresize Subgraphics : This should probably be left unselected. When selected, the resize algorithm will send a resize notice to each subgraphic, otherwise the group will simply apply a scaling factor to each component in the group.

Common Controls

Other controls common to all graphics are described in the Graphics section.

Transfer Subgraphics

Ungroup : Removes the group and places its subgraphics onto the layer of the group (makes each element operate individually) and selects those subgraphics. When this happens the focus is transferred to the selected graphics which are automatically the elements of the removed


Inspector

Group

Group

Menu Shortcuts

Edit | History | Sort

Group Parameters

15 graphics in group

Group Main Title (optional)

Does Autoresize Subgraphics

Layer And Coordinate States

Layer Ordering: #1 of 1

Edit States

Coordinate: 1

Interior

Fill Type: No Fill

Gradient Type: Constant

Start Color

End Color

Gradient Angle (°): 0°

Text

No Unrestricted Font Graphic

Text Color

Border (Stroke)

Border Unit Type: None

Color

Width (pt): 1

Segment Join

Dash Pattern (pt): Continuous Line

End Cap

Spatial Metrics

Rotation Angle (°): 0

Left Edge (pt): 171.51171875

Width (pt): 101

Bottom Edge (pt): 329.25390625

Height (pt): 126

Flip

Transfer Subgraphics (Destroy Group)

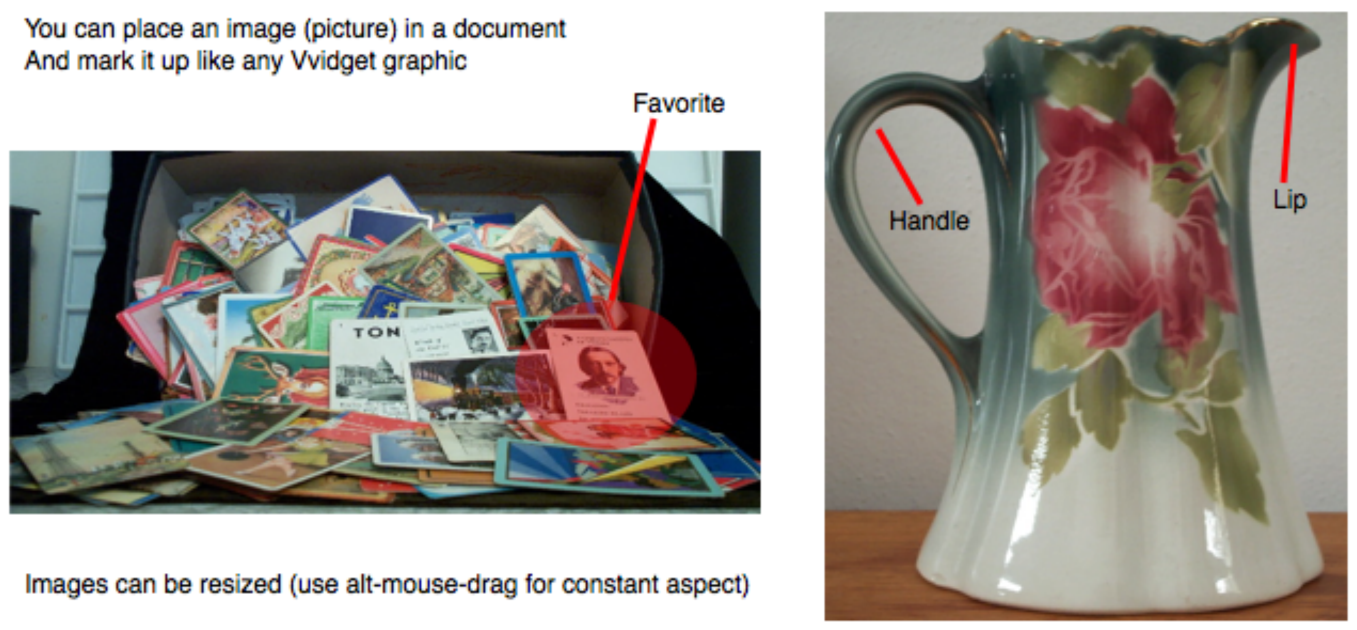
Ungroup

group. From there, those elements can be regrouped if needed.

Table Of Contents

Graph IDE ► Basic Graphics ► Image

An image is a graphic which has a standardized format such as JPEG, PNG, PDF, SVG or other standard. The figure below shows an example of an image.


You can place an image (picture) in a document
And mark it up like any Vvidget graphic
Favorite
Images can be resized (use alt-mouse-drag for constant aspect)
Handle
Lip

Some standard operations are itemized below.

- To create an image bring forward the Graphic Selector, select the image factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of a pre-made image.
- The first thing you will want to do is change the Image Data Location as described below.
- For more information see Standard Editing.
- To program an image see the Programming section.

Inspector Editor

The Inspector Editor for the image is shown below.

Menu Shortcuts

Menu shortcuts are common to all graphics and are described in the Graphics section.

Image Data Location

Path : Shows the path to the image file. On non-sandboxed editions this path can be changed directly by typing in the text field and clicking the return key. If the path is external then the path is a reference path. If the path is internal then the new path (name) is a renaming of the image data file.

Path Type : When an image file is opened then you should probably make that image internal to the document by choosing the Internal Document Path type. Making it internal copies the image file into the document. If you do not make it internal then the image graphic refers to an external file. If you move that external file then the image will no longer appear and you must reopen the image file at its new location. You may also need to use the Embedded (no path) option in which case the image is stored directly into the image archive and can be transmitted via drag and cloud operations. Notice that means that image files are not duplicated even if they are internal to the document.

External Import : Select the Open button to present the open selector. Use the open file sheet to navigate to the path of the image file and select the image file you desire. Then Open the image file. This way of constructing the image path is required in sandboxed situations.

Image Parameters

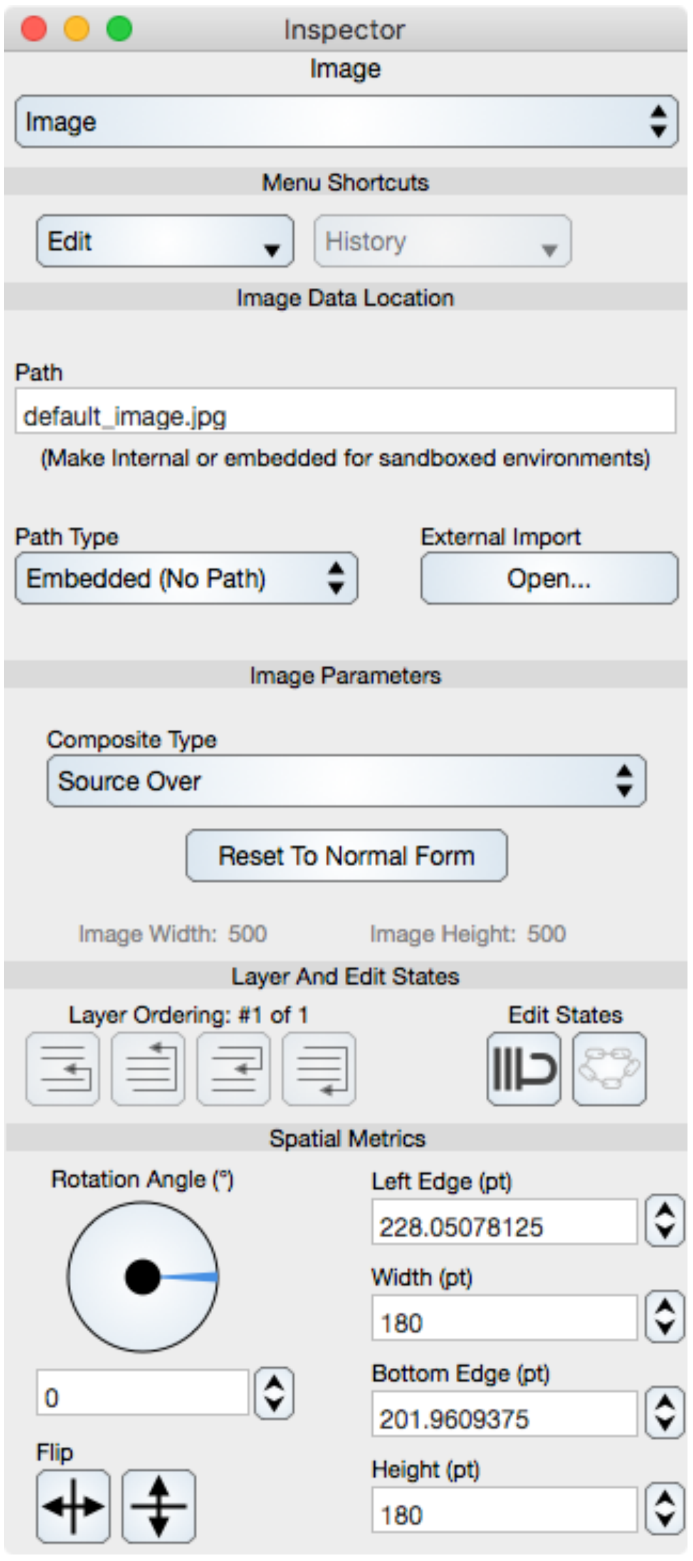
Composite Type : Defines the blending type of the image. Normally this is Source Over.

Reset To Normal Form : Select this to transform the image to the size of the image. This removes any rotation or skewing.

The width and height labels show the actual width and hight of the image in pixels.

Common Controls

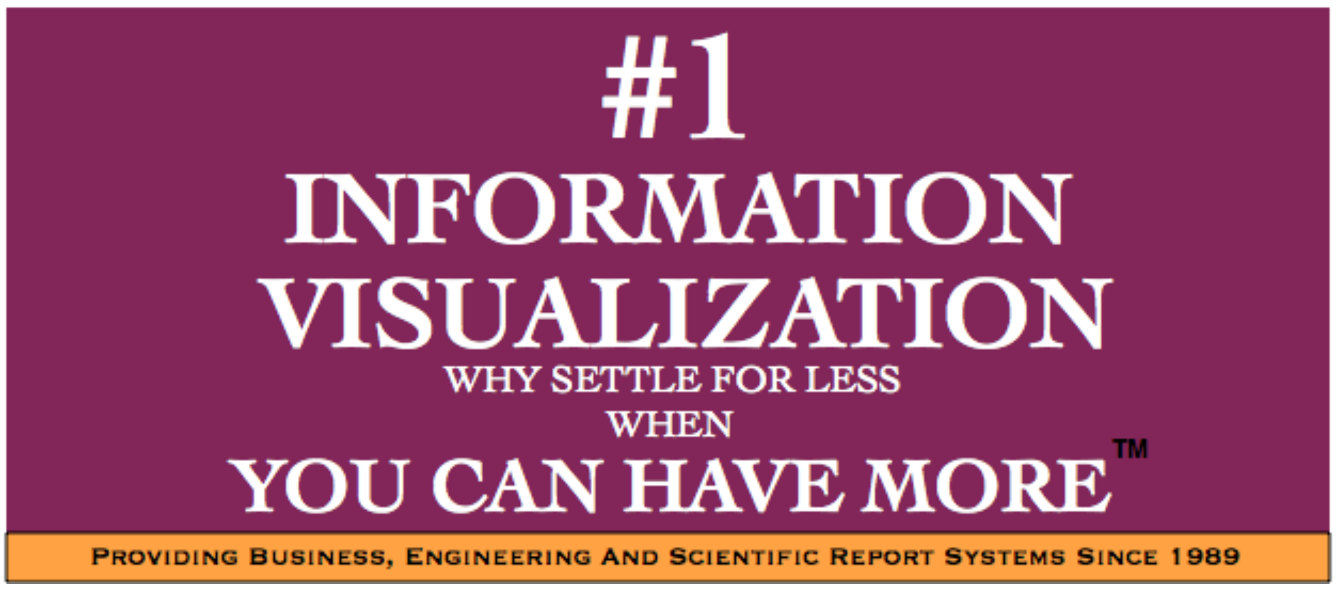
Other controls common to all graphics are described in the Graphics section.


Inspector
Image
Image
Menu Shortcuts
Edit
History
Image Data Location
Path
default_image.jpg
(Make Internal or embedded for sandboxed environments)
Path Type
Embedded (No Path)
External Import
Open...
Image Parameters
Composite Type
Source Over
Reset To Normal Form
Image Width: 500
Image Height: 500
Layer And Edit States
Layer Ordering: #1 of 1
Edit States
Spatial Metrics
Rotation Angle (°)
0
Flip
Left Edge (pt)
228.05078125
Width (pt)
180
Bottom Edge (pt)
201.9609375
Height (pt)
180

Graph IDE ► Basic Graphics ► Label

A label is a single line of text with uniform textual attributes. The figure below shows some example labels in front of background rectangles.

Some Labels On A Background


#1
INFORMATION
VISUALIZATION
WHY SETTLE FOR LESS
WHEN
YOU CAN HAVE MORE™
PROVIDING BUSINESS, ENGINEERING AND SCIENTIFIC REPORT SYSTEMS SINCE 1989

Some standard operations are itemized below.

- To create a label bring forward the Graphic Selector, select the label factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of a label. Labels can also be dragged out from the label Factory Inspector.
- For more information see Standard Editing.
- To program a label see the Programming section.

Inspector Editor

The Inspector Editor for the label is shown below.

Menu Shortcuts

Menu shortcuts are common to all graphics and are described in the Graphics section.

Label Parameters

Text : Shows and defines the label's text. Click Return to enter the text.

Unit Formatter : Select the unit formatter button to bring forward the unit selector. This is a quick way to add a unit symbol to the text as either a prefix or suffix.

Size To Fit : When selected, the text entered will cause the label graphic to size to fit and assume canonical (non-transformed) orientation.

Font Name ; Size : Shows the label font name and size. Selecting this label brings up the Font Selector. This label is right below the Text field entry and does not look like a control.

Alignment : Shows and defines the alignment of the text within the graphic bounds. If size to fit is used then the graphic bounds and the text bounds are identical and all alignments are the same. To see different alignments first resize the text graphic. Give the graphic a fill color to see how the alignment occurs within the reference bounds of the graphic.

Inset : Shows and defines the inset of the text from the alignment edge.

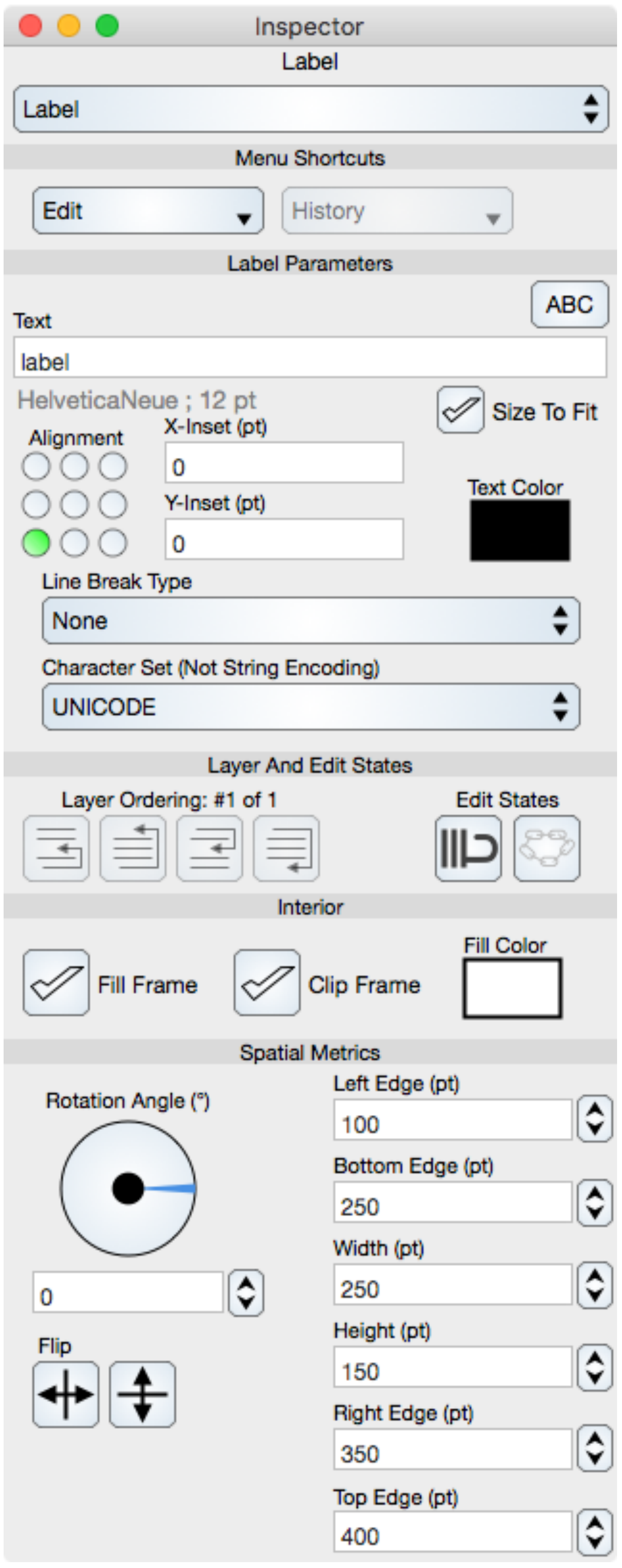
Line Break Type : Shows and defines how the text breaks within its reference bounds. The options are None, Truncate Ends, Truncate Middle, Ellipsis Ends, Ellipsis Middle, Character Wrap and Word Wrap. Character Wrap and Word Wrap are not implemented at the time of this writing. Line break modes are used for Spreadsheets because cells have limited width. They are used at other times as well. The line break setting interacts with the alignment, inset and reference frame settings. Line breaks only occur when the reference frame of the label inset by the inset values is less in width than the text width of the label. If Ellipsis Ends is used and the alignment is left aligned then ellipsis will never display on the left of the text. If the text is aligned horizontally centered then ellipsis display on both ends of the text. If the text is right aligned then the ellipsis will never appear on the right side of the text.

Character Set : The character set used for the text. This is normally UNICODE but can also be ASCII. Note that it is not UTF8, for example, because that is an encoding and not a character set. Leave this as UNICODE unless you need specific optimizations.

Color : Shows and defines the Color of the text.

Clip Frame : Clips the text to the frame of the graphic. Notice how other graphics define clipping as a mask while the label defines clipping only upon the text. If a line break is used then clipping is unneeded.

Common Controls


Inspector

Label

Label

Menu Shortcuts

Edit | History

Label Parameters

ABC

Text: label

HelveticaNeue ; 12 pt

Size To Fit: ✓

Alignment

X-Inset (pt): 0

Y-Inset (pt): 0

Text Color: ■

Line Break Type: None

Character Set (Not String Encoding): UNICODE

Layer And Edit States

Layer Ordering: #1 of 1

Edit States

Interior

Fill Frame: ✓

Clip Frame: ✓

Fill Color: □

Spatial Metrics

Rotation Angle (°): 0

Flip

Left Edge (pt): 100

Bottom Edge (pt): 250

Width (pt): 250

Height (pt): 150

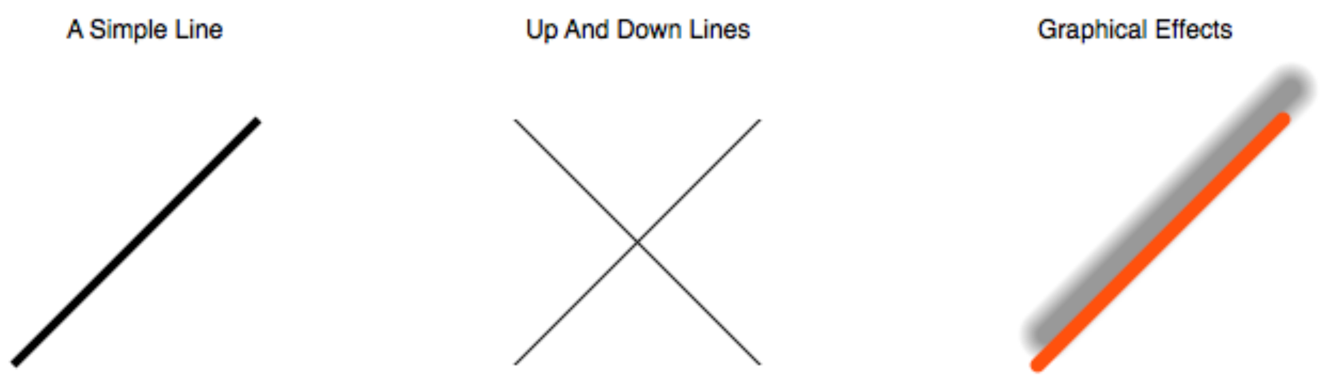
Right Edge (pt): 350

Top Edge (pt): 400

Other controls common to all graphics are described in the Graphics section.

Graph IDE ▶ Basic Graphics ▶ Line

A line graphic is a single line segment with other graphic attributes. The figure below shows examples of lines.



Standard Operations

To create a line bring forward the Graphic Selector, select the line factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the end points of the line segment. Premade lines can also be dragged out from the line Factory Inspector.

For more information see Standard Editing.

Inspector Editor

The Inspector Editor for the line is shown below.

Menu Shortcuts

Menu shortcuts are common to all graphics and are described in the Graphics section.

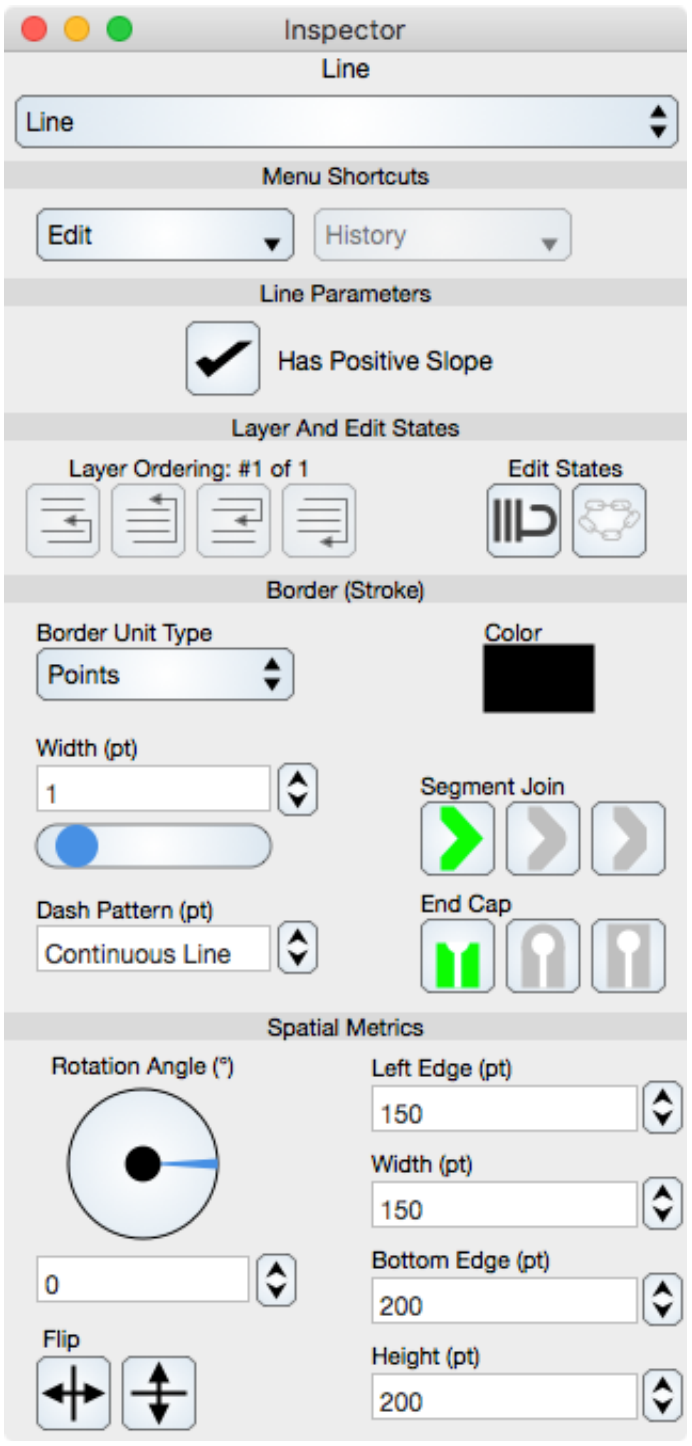
Line Parameters

Has Positive Slope : Defines the slope of the line as positive or negative. A line is always oriented along the diagonal of its reference bounds and as such there are only two diagonals.

Common Controls

Other controls common to all graphics are described in the Graphics section.

Spatial Metrics controls are in reference to the reference bounds of the line. A width of zero means a vertical line and a height of zero means a horizontal line. Contrast this to the stroke width which defines the width of the line segment.


Inspector

Line

Line

Menu Shortcuts

Edit | History

Line Parameters

[✓] Has Positive Slope

Layer And Edit States

Layer Ordering: #1 of 1

Edit States

Border (Stroke)

Border Unit Type: Points

Color

Width (pt): 1

Segment Join

Dash Pattern (pt): Continuous Line

End Cap

Spatial Metrics

Rotation Angle (°): 0

Left Edge (pt): 150

Width (pt): 150

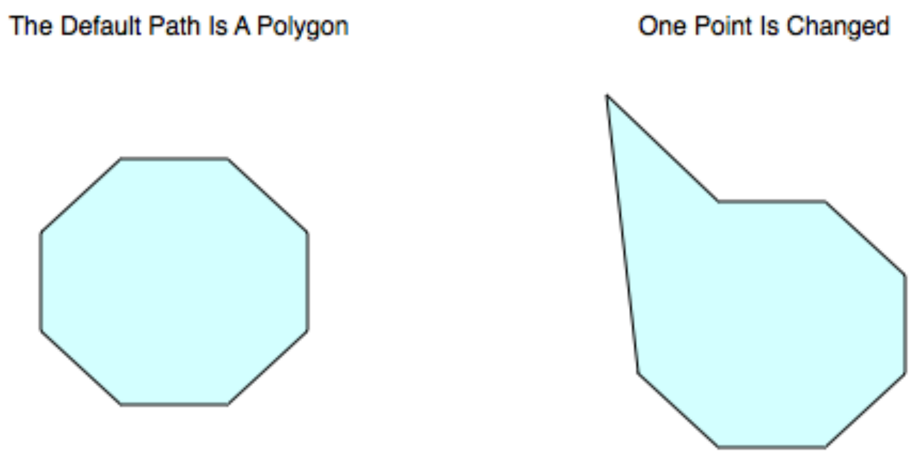
Bottom Edge (pt): 200

Height (pt): 200

Flip

Graph IDE ► Basic Graphics ► Path

A path is defined as a sequence of operation and operand segments. It is mainly used for programming and is provided at the Graph IDE level so that programmers can insert template paths into their documents. The figure below shows the default path.



The default path has these path operations and operands (the operands will have different values according to the size and origin of the path):

```
move to point: 159 203
add line to point: 219 203
add line to point: 264 246.5
add line to point: 264 304.5
add line to point: 219 348
add line to point: 159 348
add line to point: 114 304.5
add line to point: 114 246.5
add line to point: 159 203
```

Standard Operations

To create a path bring forward the Graphic Selector, select the path factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of the default path. You can use the inspector to change operation and operand sequences.

Resizing and rotating the path transforms all the operands. For more information see Standard Editing.

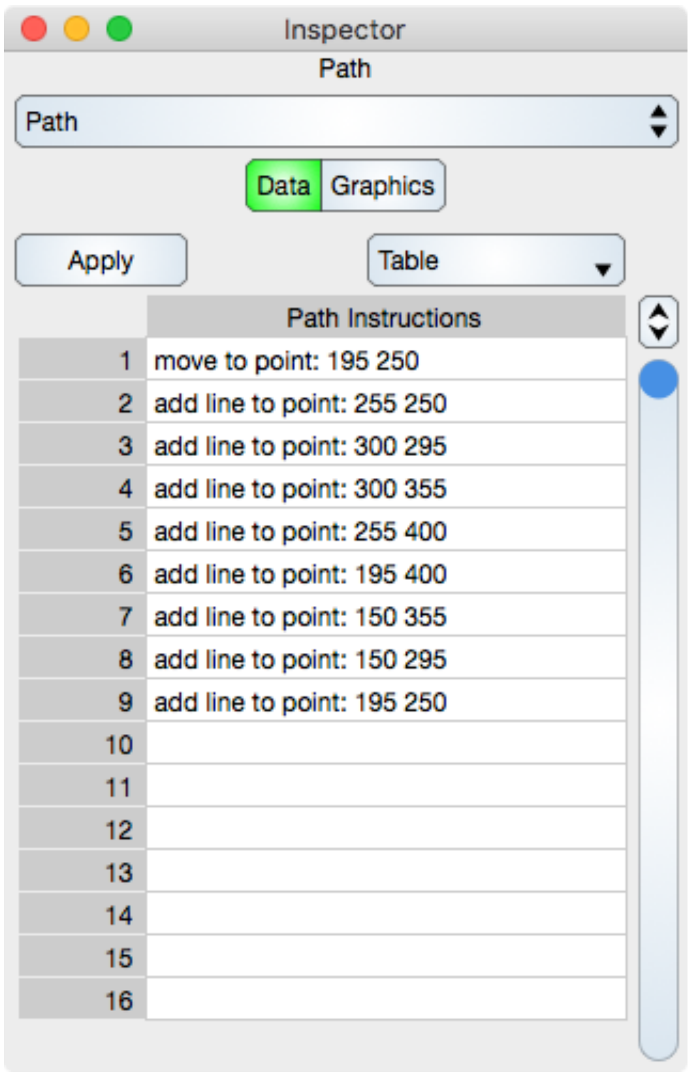
Data Editor

The Data Editor for the path is shown below.

Apply : Selecting this applies the path instructions shown in the table.

Table controls are described in the Tables section.

The rows represent path instructions.



	Path Instructions
1	move to point: 195 250
2	add line to point: 255 250
3	add line to point: 300 295
4	add line to point: 300 355
5	add line to point: 255 400
6	add line to point: 195 400
7	add line to point: 150 355
8	add line to point: 150 295
9	add line to point: 195 250
10	
11	
12	
13	
14	
15	
16	

Graphics Editor

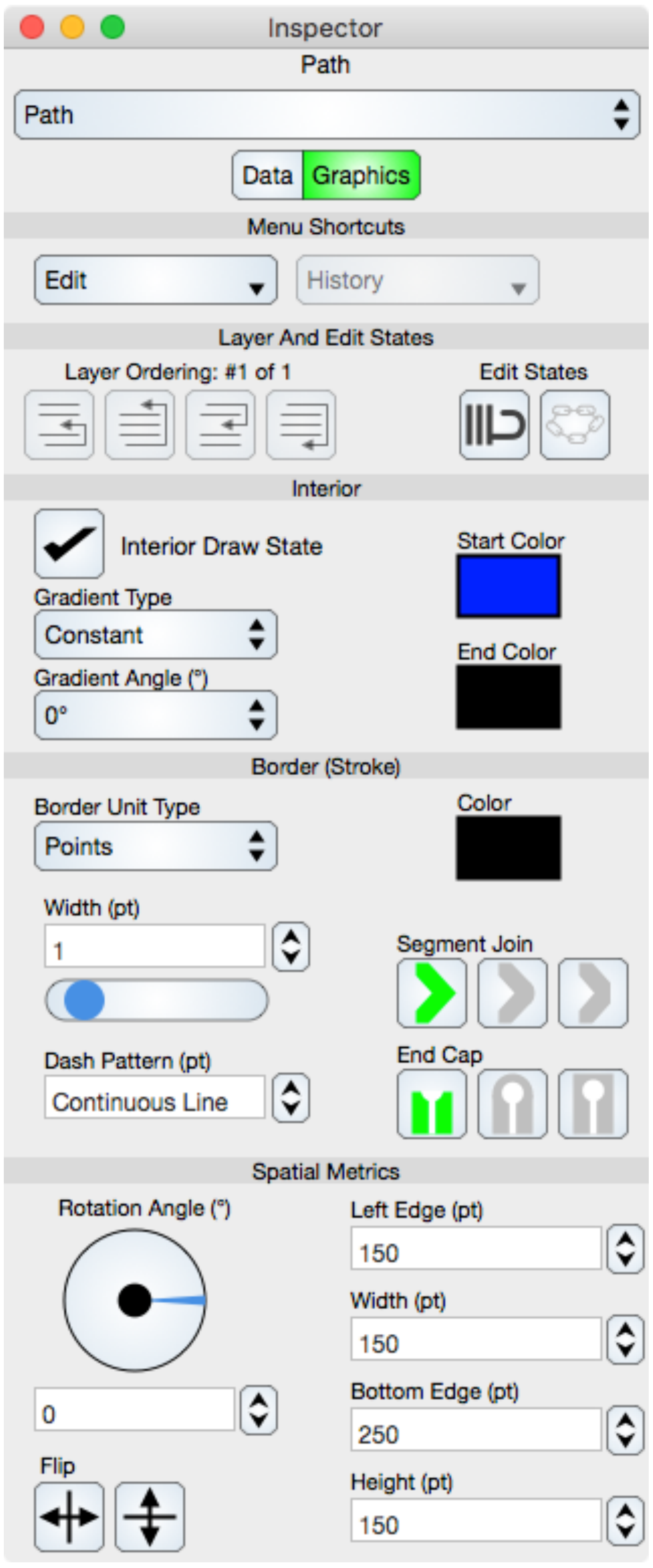
The Graphics Editor for the Path is shown below.

Common Controls

Controls common to all graphics are described in the Graphics section.

Path Specific Controls

There are no path specific graphic controls as all of the graphic operations are generic.



Inspector

Path

Path

Data Graphics

Menu Shortcuts

Edit History

Layer And Edit States

Layer Ordering: #1 of 1

Edit States

Interior

Interior Draw State ✓

Start Color

Gradient Type: Constant

End Color

Gradient Angle (°): 0°

Border (Stroke)

Border Unit Type: Points

Color

Width (pt): 1

Segment Join

Dash Pattern (pt): Continuous Line

End Cap

Spatial Metrics

Rotation Angle (°): 0

Left Edge (pt): 150

Width (pt): 150

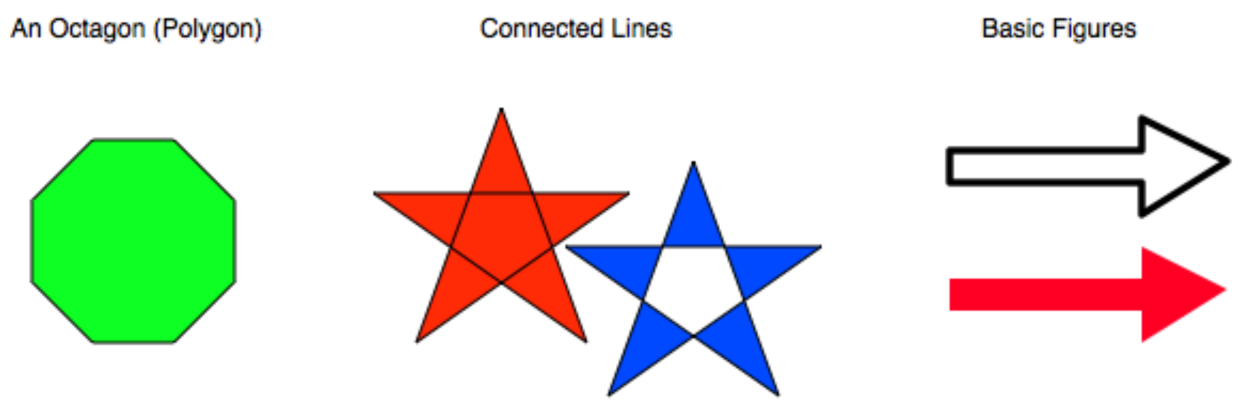
Bottom Edge (pt): 250

Height (pt): 150

Flip

Graph IDE ▶ Basic Graphics ▶ Polygon

A polygon graphic is a sequence of points connected by line segments. When no line segments defined by the points intersect then the graphic is a polygon. If they intersect then the graphic is more general geometrically speaking. The figure below shows some examples of a polygon.

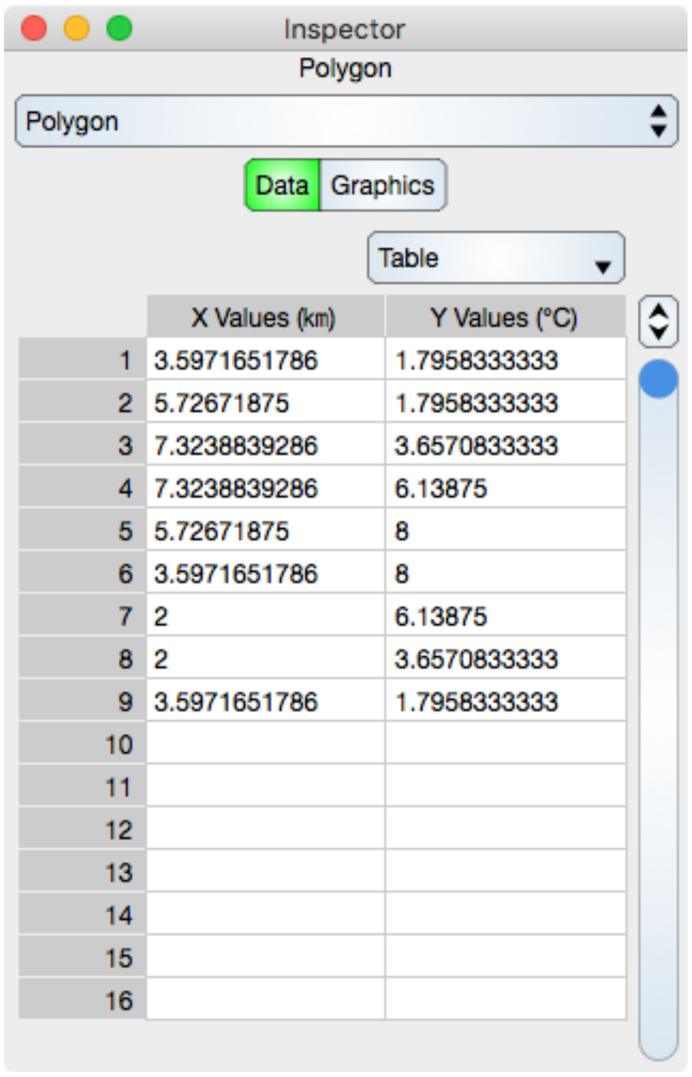


Some standard operations are itemized below.

- To create a polygon bring forward the Graphic Selector, select the polygon factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of a predefined polygon. Premade polygons can be dragged out from the polygon Factory Inspector.
- Resizing and rotating the polygon transforms all points simultaneously. For more information see Standard Editing.
- One thing you will probably want to do with a polygon is modify its points. You can do that via the Data or Parser Inspector Editor or by editing it directly with the mouse controls.
- To program a polygon see the Programming section.

Data Editor

The Data Editor for the Polygon is shown below.



	X Values (km)	Y Values (°C)
1	3.5971651786	1.7958333333
2	5.72671875	1.7958333333
3	7.3238839286	3.6570833333
4	7.3238839286	6.13875
5	5.72671875	8
6	3.5971651786	8
7	2	6.13875
8	2	3.6570833333
9	3.5971651786	1.7958333333
10		
11		
12		
13		
14		
15		
16		

Table

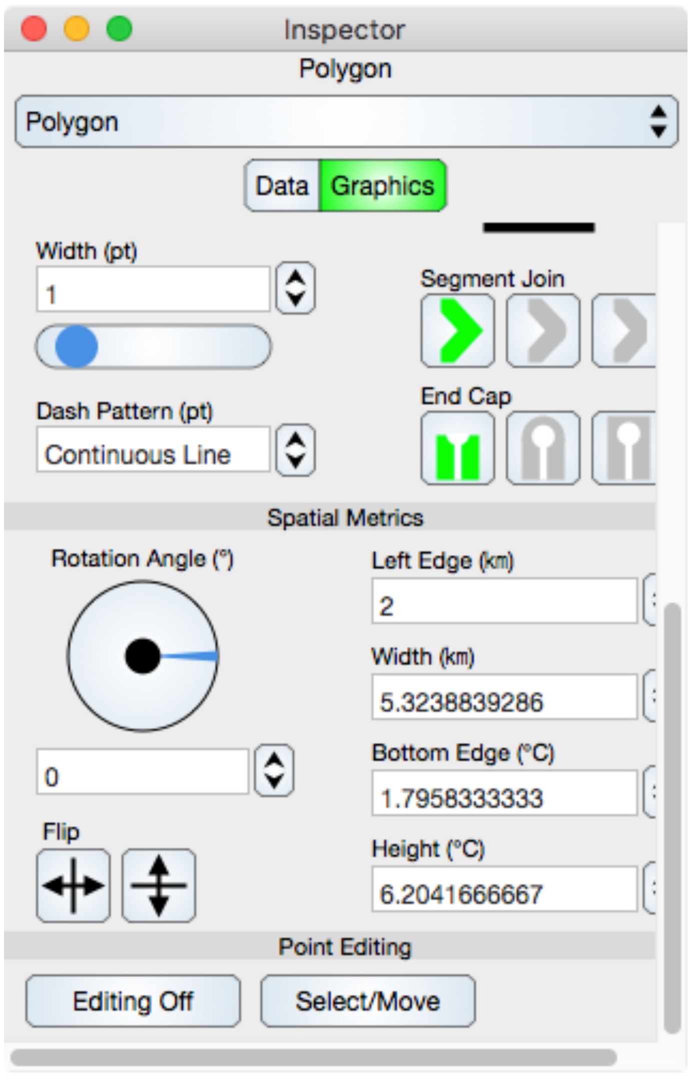
Table controls are described in the Tables section.

The rows represent vertex values. While in Atomic mode, the cell represents a point (x and y value) and while in component mode the cell represents either a x or y value. Hence, in atomic mode there is one column while in component mode there are two columns.

Notice that the units of the first column are kilometers while the units in the second column are degree Celsius. That is because the polygon is on a graph (data layer) and takes on the units of the graph coordinate which is set to kilometer for the X-axis and Celsius for the Y-axis.

Graphics Editor

The Graphics Editor for the Polygon is shown below.


Inspector
Polygon
Polygon
Data | Graphics
Width (pt): 1
Dash Pattern (pt): Continuous Line
Segment Join
End Cap
Spatial Metrics
Rotation Angle (°): 0
Left Edge (km): 2
Width (km): 5.3238839286
Bottom Edge (°C): 1.7958333333
Height (°C): 6.2041666667
Flip
Point Editing
Editing Off
Select/Move

Common Controls

Controls common to all graphics are described in the Graphics section.

Point Editing

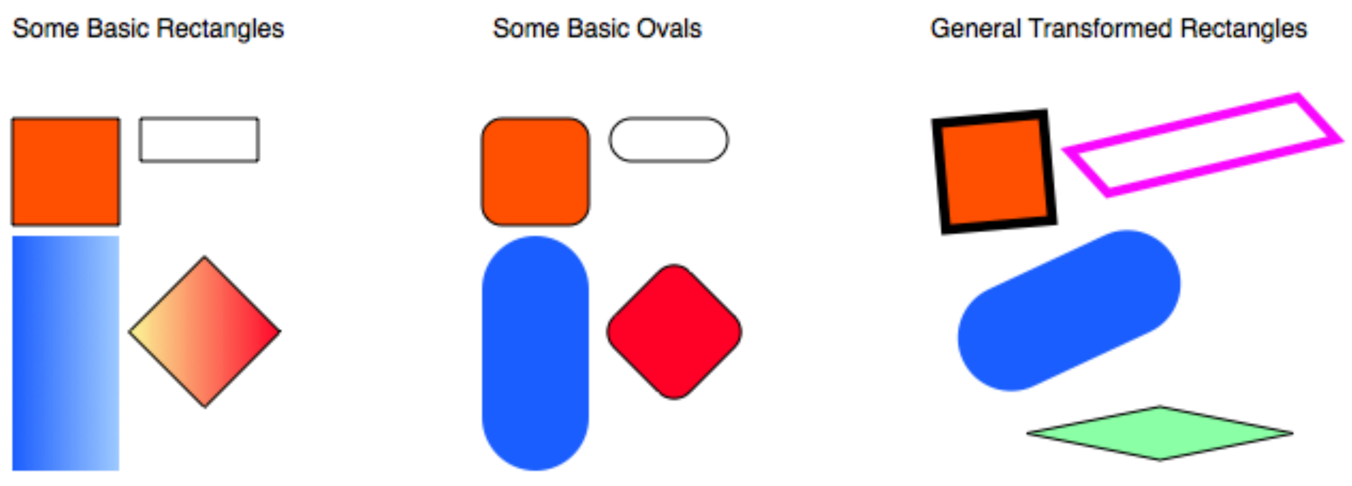
Editing Off/On : Places the graphic into or out of edit mode. While in edit mode the vertices are shown by indicators and can be adjusted using mouse or touch events. Double-clicking the graphic also toggles this edit mode.

Select/Move or Add/Delete : Select/Move mode permits the vertex editing to select and move the vertex location while Add/Delete mode will delete a vertex if it is hit or add a vertex if a polygon segment is hit. This can also be accomplished using the shift key if available.

Notice that while the units displayed in the reference frame values are those of the graph units as described in the Data Editor above. If the polygon where directly in a Graphic View (and not on a graph) then the units would be in report units of the Graphic View (not the units of the Graphic View coordinate which is always typographic points).

Graph IDE ► Basic Graphics ► Rectangle

A rectangle is defined as four points at the intersection of two horizontal and two vertical lines. It can be transformed into a parallelogram by a linear transformation, such as a rotation and then flattening. It also has associated attributes, such as corner radii, so it can be an oval, and other graphic attributes. The figure below shows some examples of a rectangle.



Some standard operations are itemized below.

- To create a rectangle bring forward the Graphic Selector, select the rectangle factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of a rectangle. If the width and height are the same then it is a square. Prototypes can also be dragged out from the rectangle Factory Inspector.
- You can also create rectangles from the standard Palettes ► Basics ► Squares menu item.
- Rotating and resizing the rectangle transforms it to a parallelogram. For more information see Standard Editing.
- To program a rectangle see the Programming section.

Inspector Editor

The Inspector Editor for the rectangle is shown below.

Menu Shortcuts

Menu shortcuts are common to all graphics and are described in the Graphics section.

Oval Parameters

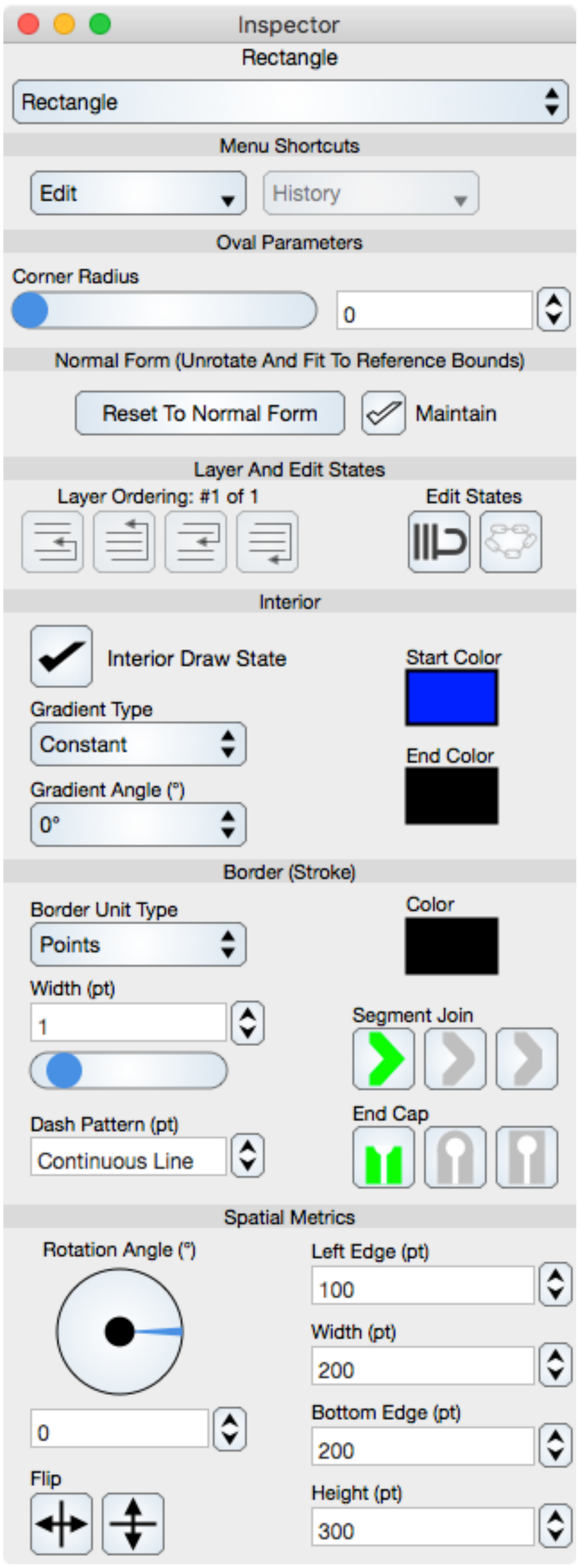
Corner Radius : Shows and defines the radius of each corner of the rectangle. When the corner radius is non-zero then the rectangle is an oval. If the rectangle is resized without maintaining aspect then the oval becomes a general affine transformed oval.

Normal Form

Reset To Normal Form : Select this to set the width and height of the rectangle to the reference frame of the graphic.

Common Controls

Other controls common to all graphics are described in the Graphics section.



Rectangle

Menu Shortcuts

Oval Parameters

Corner Radius: 0

Normal Form (Unrotate And Fit To Reference Bounds)

Maintain: ✓

Layer And Edit States

Layer Ordering: #1 of 1

Interior

Interior Draw State: ✓

Gradient Type: Constant

Gradient Angle (°): 0°

Start Color

End Color

Border (Stroke)

Border Unit Type: Points

Color

Width (pt): 1

Segment Join

Dash Pattern (pt): Continuous Line

End Cap

Spatial Metrics

Rotation Angle (°): 0

Left Edge (pt): 100

Width (pt): 200

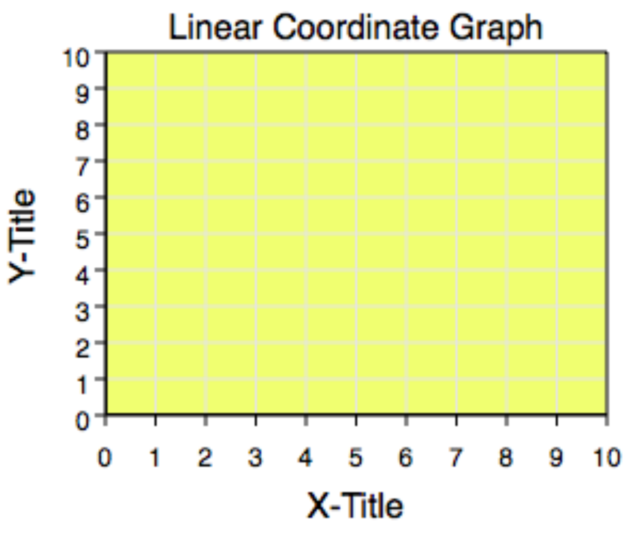
Bottom Edge (pt): 200

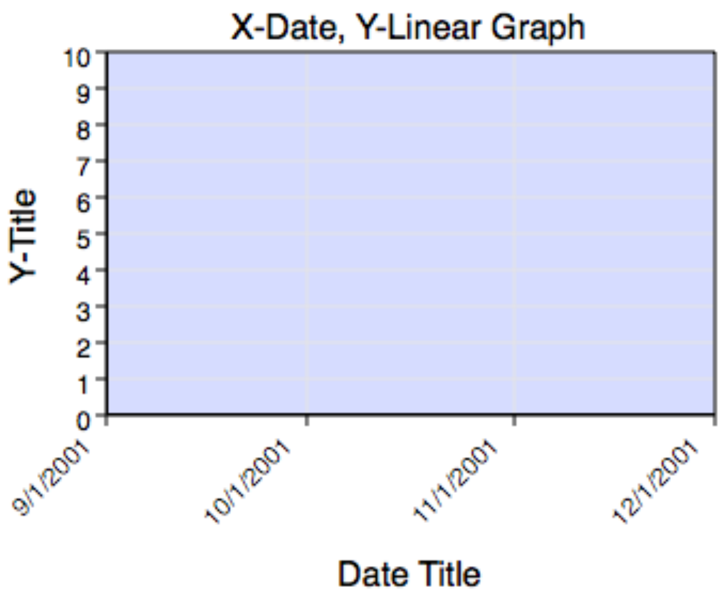
Height (pt): 300

Flip

Graph IDE ► Graphs

A graph is normally defined as combination of Data Graphics with a graphic that displays attributes of the coordinate system that the data is embedded in. The figure below shows examples of graphs without any data on them. Realize that most people think of a graph as both the graphic that represents the coordinate system and also the data graphics on the graph (such as a line graph). In Graph IDE, the graph definition excludes the data graphics.





You make a graph just like any other graphic, as explained in Standard Editing (by dragging it out or dragging it from Palettes or the Factory Inspector) or by using Chart Tasks. A subsequent section (Getting Data On A Graph) explains how to get data onto the graph.

Notice that each graph inspector editor has a Unit Selector for each dimension of the graph. When you create data on a graph is inherits the units of the graph dimensions so that there are no unit selectors for data graphics. Units show in the Cursor Information and other locations to aid in determining the currently focused coordinate system.

In the current implementation, units do not define a coordinate map and they merely label the coordinate dimensions. Coordinate mapping is a separate control explicitly set in each graph inspector editor. Setting units for a graph is straight forward, however setting coordinate maps can be challenging. Chances are, you will not need to set coordinate maps (except for a preset mapping of angle (from radians to degrees) on polar graphs, or perhaps non-decimal bases on a log graph). Typically, the data itself is remapped onto a default coordinate system. There are many ways to remap data, for example the Formula Selector. Notice that remapping data permanently alters the data while mapping coordinates maintains the data values and merely alters the mapping onto the graph coordinate (and by nesting the page coordinate).

The following is a brief list and definitions of graph related sections:

Section	Graphs
Autoscaler	The autoscaler adjust graph limits in predefined ways, normally associated with the base of the axis limits.
Date Axis	A single dimension of a coordinate of a graph is represented by an axis. The date axis represents a date dimension of a coordinate system. Note that date might imply a specific unit, but that is actually not the case. See the section on Date Axis for additional information.
Getting Data On A Graph	Explains how to get a Data Graphic on the graph.
Graphic Attributes	Almost all graphs have some common graphic-related attributes. This section describes them.
Label Mapping	Axis values are shown by their labels. Labels can be mapped into different representations in order to make the relationship between data and graph more understandable. It is the label mapping which defines this attribute.
Linear Axis	A single dimension of a coordinate of a graph is represented by an axis. The linear axis represents a linear dimension of a coordinate system.
Log Axis	A single dimension of a coordinate of a graph is represented by an axis. The log axis represents a log dimension of a coordinate system.
Multiple Coordinate Graph	A graph which defines several colocated coordinates. For each point on the graph the coordinate representing that point is defined by a related parameter, the focused coordinate index.
Non-Linear Graphs	Describes aspects of working with non-linear graphs.
Polar Axes	The polar axes represents a theta and r-oriented axis.
Single Coordinate Graph	A graph which defines only one coordinate. The coordinate can be of any type, linear, log, polar, nonuniform; however, there needs to be only one coordinate per point on the graph.

The following is a list of combinations of axis that can be made via the Graphic Selector in order to form 2 dimensional graphs.

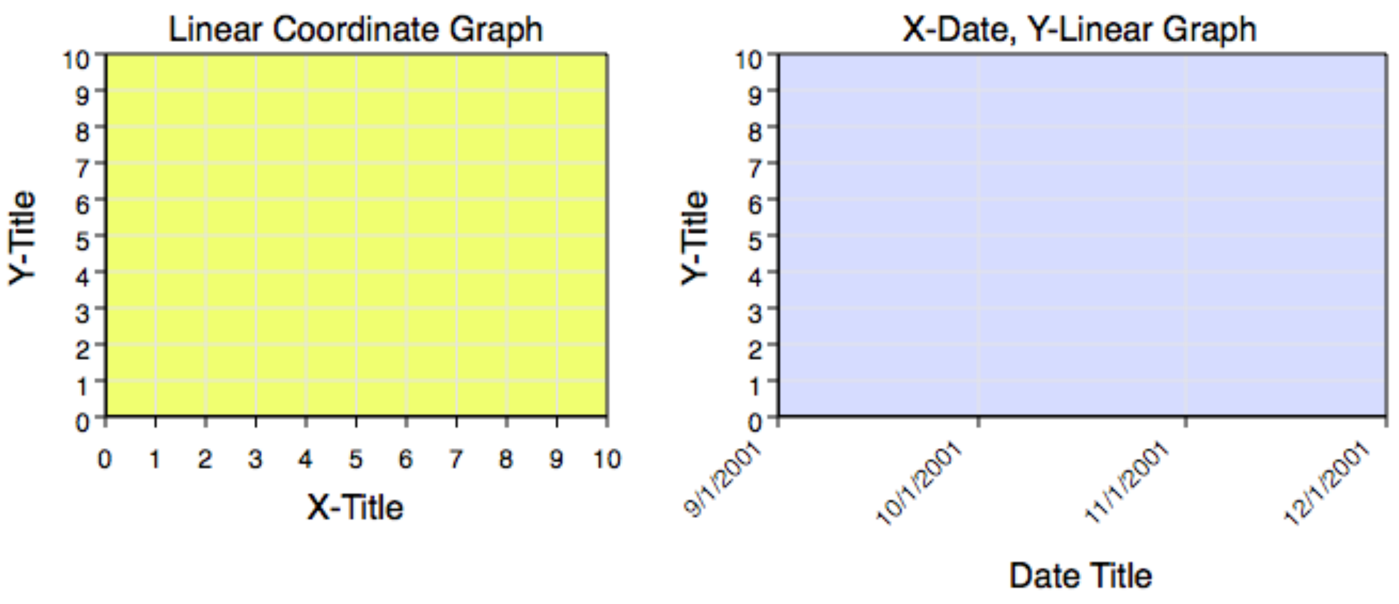
Section	Axis Combinations That Form Standard Graphs
X-Y Graph	This is the usual graph everyone knows about. It has an x-axis at the bottom and a y-axis on the left and both are linear axes.
X-Date, Y-Linear Graph	This graph has the usual y-axis on the left, but the x-axis is a date axis, which can have non-uniform tick placement if the date unit increments non-uniformly, such as for months (For example, February has 28 days and January has 31 days).
N-Y Graph	This is the graph you would use if your data had a common x-axis unit, such as distance in meters, but the y-axis units are different, for example one data curve of pressure and the other of temperature.
X-Date, N Y-Linear Graph	Use this type of graph if the data has x-values in units of seconds since 1970 and there are two or more data sets of different y-units.
X-Linear, Y-Log	The standard semi-log graph.
X-Log, Y-Linear	A semi-log graph where the log coordinate is the x-direction not the y-direction.
X-Log, Y-Log	Also known as the log-log plot.
X-Date, Y-Log	A date-graph where the amplitude (y-direction) is expressed in log units.
Polar	Also called a radar graph. This graph shows the x-axis in terms of angle about a point and the y-axis radially. The angle dimension is the independent variable dimension.
Log-R	Also called a log-polar graph. This graph shows the x-axis in terms of angle about a point and the y-axis radially as a log axis. The angle dimension is the independent variable dimension.

The Vvidget system generalizes combinations of axis to form graphs, so the combinations of axis to form graph can become very large. Graph IDE constrains the combinations to the usual types (and in some instance perhaps some very unusual ones).

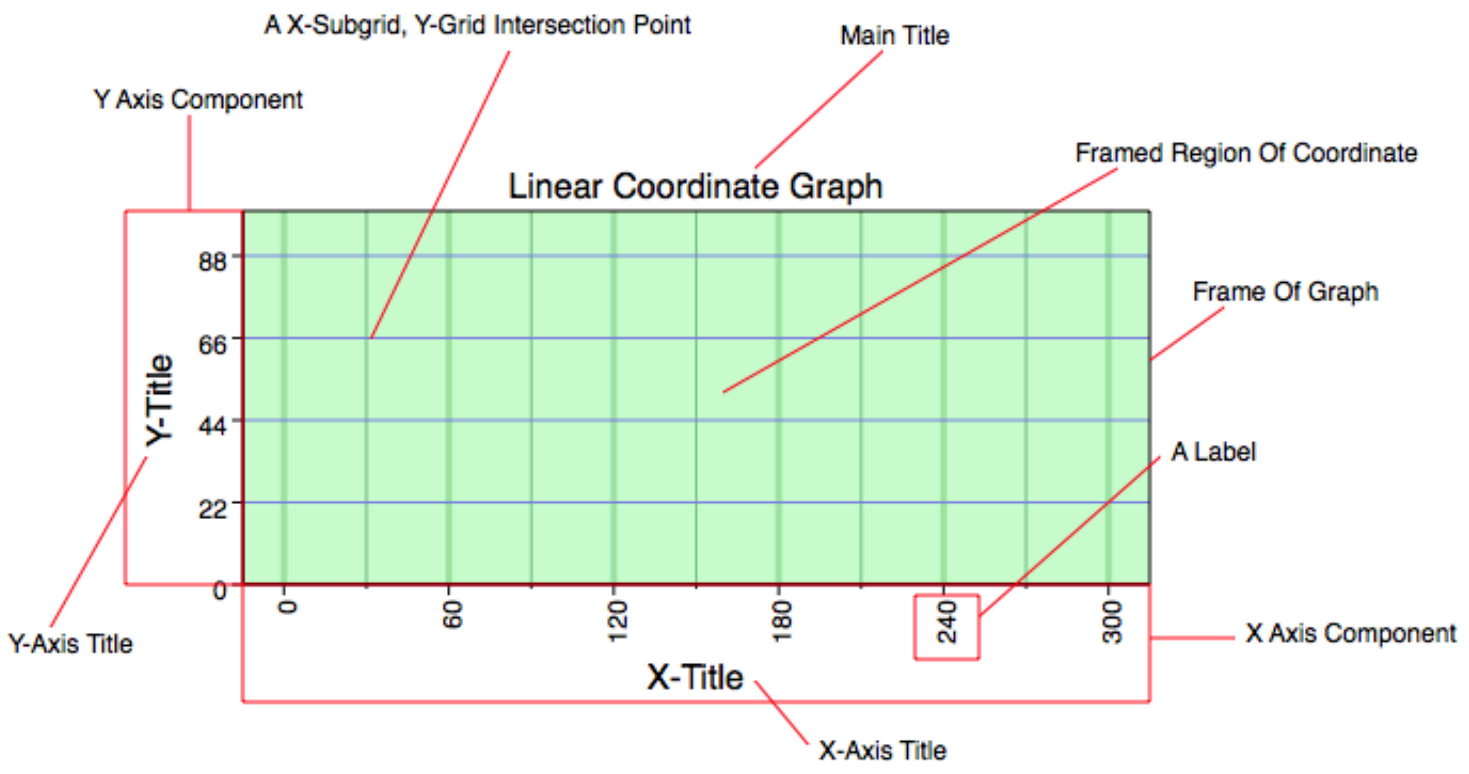
Graph IDE ► Graphs ► Single Coordinate Graph

A Single Coordinate Graph is the usual type of graph which displays only one coordinate. The coordinate can be of any type, linear, log, angular, radial, nonuniform and mixed or non-mixed amongst those types. The coordinate can also have non-orthogonal dimensions. However, there needs to be only one coordinate per point on the graph; contrast that to the Multiple Coordinate Graph.

The figure below shows two examples of a single coordinate graph without any data on them.



The figure below annotates the major components of the graph:



Note the x-axis and how the tick placement does not have to coincide with the left and right sides of the frame of the graph. The ticks can float along the graph by varying the tick x-minimum or the graph's x-minimum for example.

Inspector Editor

The Metrics Editor for the Single Coordinate Graph is described below.

Automatically Reset Fields And Graph

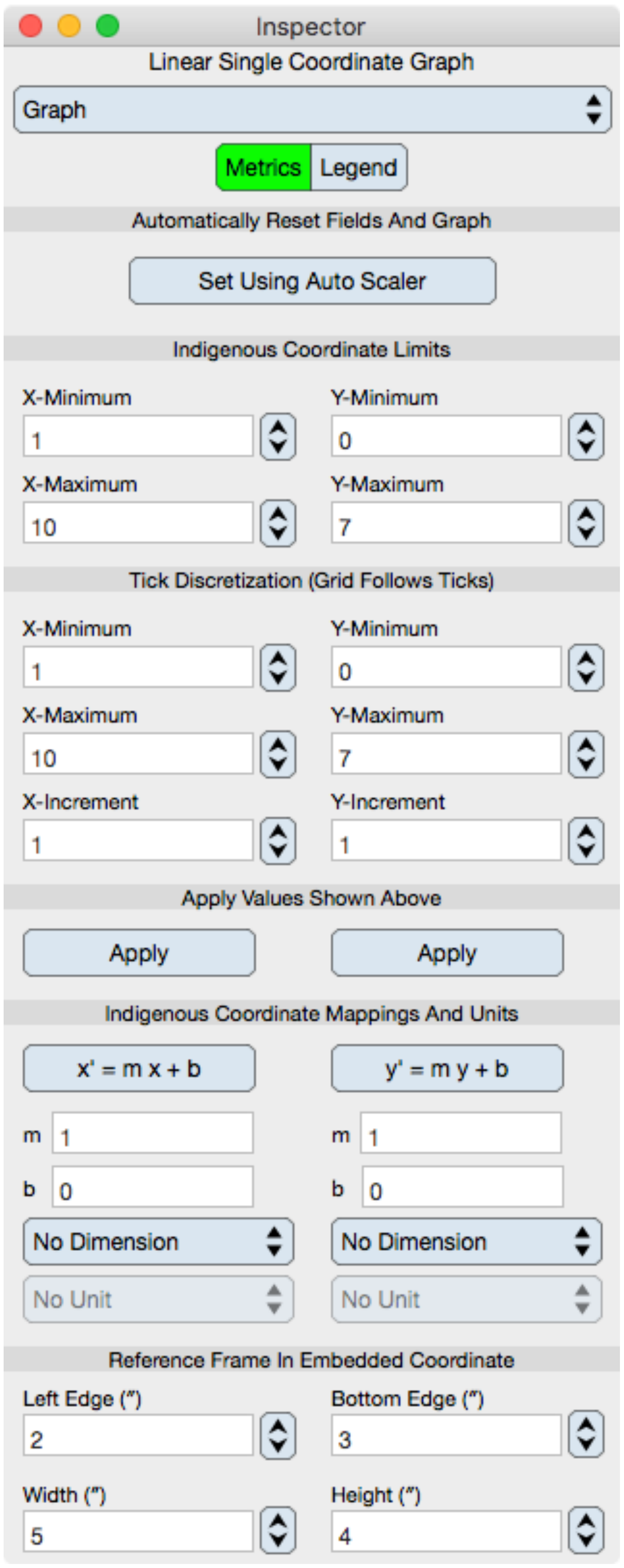
Set Using Auto Scaler : Selecting this button autoscales the graph according to the settings in the Auto Scaler inspector.

Indigenous Coordinate Limits

This specifies the actual limits of the coordinate system represented by the graph and those limits are defined at the graph frame. When you change these limits then you should also probably change the tick discretization limits to the same values.

X-Minimum : The x-minimum value of the graph.

X-Maximum : The x-maximum value of the graph.



Y-Minimum : The y-minimum value of the graph.

Y-Maximum : The y-maximum value of the graph.

Tick Discretization

This specifies the tick locations along the axes defined in the coordinate shown by the axes. Note that the ticks "float" which is to say that they don't follow the Indigenous Coordinate Limits.

X-Minimum : The x-minimum tick value of the graph. Usually this is the same as the x-minimum value of the graph.

X-Maximum : The x-maximum tick value of the graph. Usually this is the same as the x-maximum value of the graph.

Y-Minimum : The y-minimum tick value of the graph. Usually this is the same as the y-minimum value of the graph.

Y-Maximum : The y-maximum tick value of the graph. Usually this is the same as the y-maximum value of the graph.

Apply Values Shown above

Apply : Updates the graph to correspond to the values shown above the button. Note that entering a value in a text field will not change the graph even if the return key is clicked. That is because all of the numbers are related and updating the graph based upon only one of the values can produce an inconsistent state.

Coordinate Mappings

Graphics embedded on the graph are in reference to the Indigenous Coordinate Limits and will appear to translate and scale according to those limits relative to the page view coordinate system. However, those graphics do not actually map, only the mapping relative to the page view changes. However, there is also another mapping type which is the coordinate mapping and is determine by a function, normally a linear function. This mapping maps the actual data relative to the graph coordinate (not the page view coordinate). The data remains the same however and is not mapped.

x = mx + b : This is the function of the mapping. Click this button to change that function to include a absolute value of x.

m : The slope of the linear mapping.

b : The y-intercept of the linear mapping.

This mapping feature makes more sense for polar and logarithmic graphs where the x-values need to be mapped from degrees to radians for polar coordinates and in the case of logarithmic graphs the mapping function includes base, power and reference coefficients. In the case of logarithmic graphs the base coefficient also effects the base of the graph labels while in scientific notation. So, the mapping can be a simple function, can be a unit designation or can be a label formatting requirement.

Units

The Indigenous Coordinates and Tick Discretization are all defined in their own units separate from any typographical requirement. Set that unit using the Unit Selector available on the inspector. Units appear in the Cursor Information and coordinate with the Spreadsheet. Units are informational only and will not transform the data on the graph or set the coordinate mapping.

Reference Frame

This specifies the reference frame location of the graph. Note that the reference frame is the frame that coincides with the Indigenous Coordinate Limits location but in graphic view coordinate report units. For example, the y-axis of the graph can be in degrees Celsius while the reference frame is in units of inches. The unit of the reference frame is shown in the label next to the frame component to clarify that the reference frame and graph indigenous coordinates are in different units.

Left Edge : The left edge of the graph in units and coordinate of the Graphic View.

Bottom Edge : The bottom edge of the graph in units and coordinate of the Graphic View.

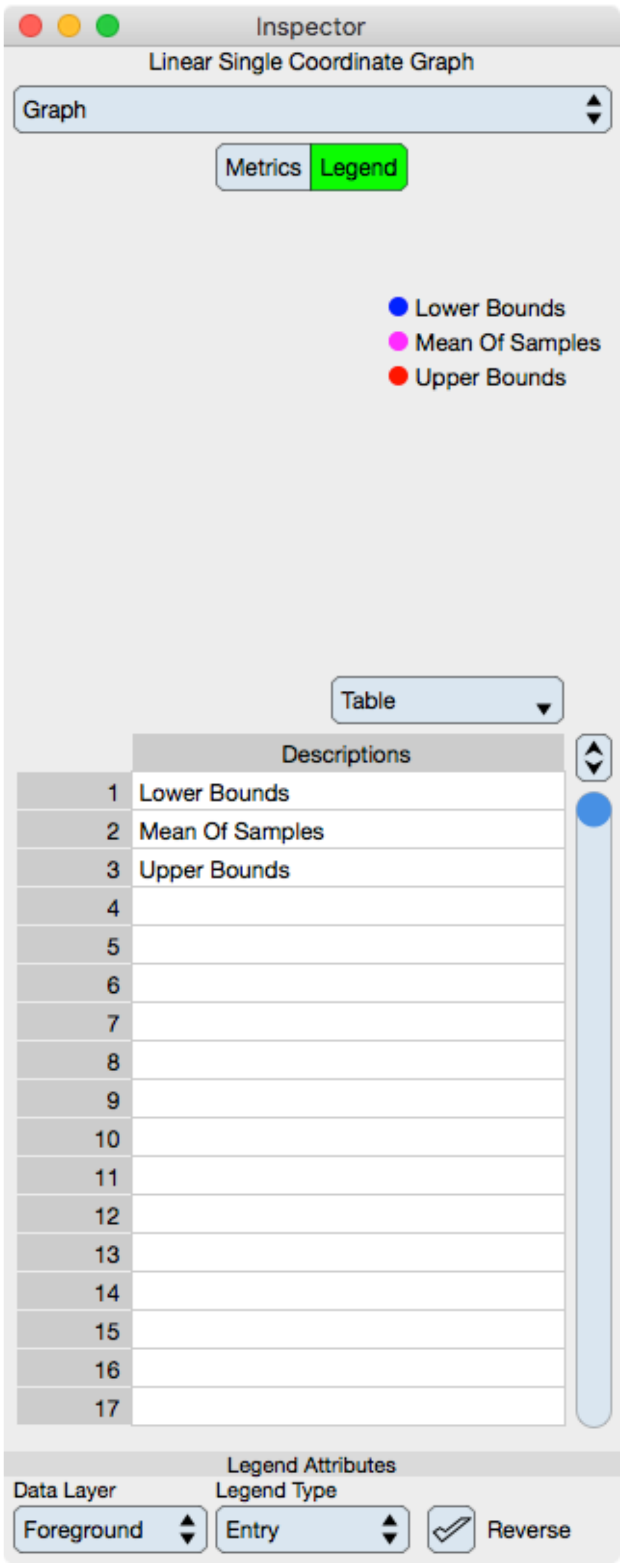
Width : The width of the graph in units and coordinate of the Graphic View.

Height : The height of the graph in units and coordinate of the Graphic View.

Legend

The Legend Editor for a Graph is described below. Legends are built from graphics on the graph's data layer so first place graphics on the graph, for example a Function (Curve) and then use this editor.

Legend



	Descriptions
1	Lower Bounds
2	Mean Of Samples
3	Upper Bounds
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	

Legend Attributes

Data Layer: Foreground
Legend Type: Entry
✓ Reverse

Legend Group : The legend is a Group Graphic and is built automatically using the attributes specified below. Once the attributes below are specified then drag the legend near your graph on the document's Graphic View as it is also an element of a Palette. The legend is set to drag and drop to the overlay layer of the graphic view and not the data layer of the graph.

Table

Descriptions : Each row of a legend has a description which can be edited into the text of the descriptions table. Those entries are stored in the description key of the graphic resource dictionary whose entry can be modified directly via the Expert inspector.

Legend Attributes

Data Layer : The legend is built from either the foreground or background data layer of the graph, but not both. See Layer for additional information.

Legend Type : Either Entry, Circle or Square. If circle or square then the legend marker is represented by a circle or rectangle respectively whose fill color is that of the represented graphic. If Entry then the marker queues of the marker, stroke or fill of the represented graphic.

Reverse : The row order of the legend is the draw order of the data layer. It makes sense that the draw order of the data goes from minimum to maximum value which is bottom to top but legends are constructed top to bottom. Reversing the legend order makes the legend appear be in the same order as the data magnitudes, which is a heuristic effect.

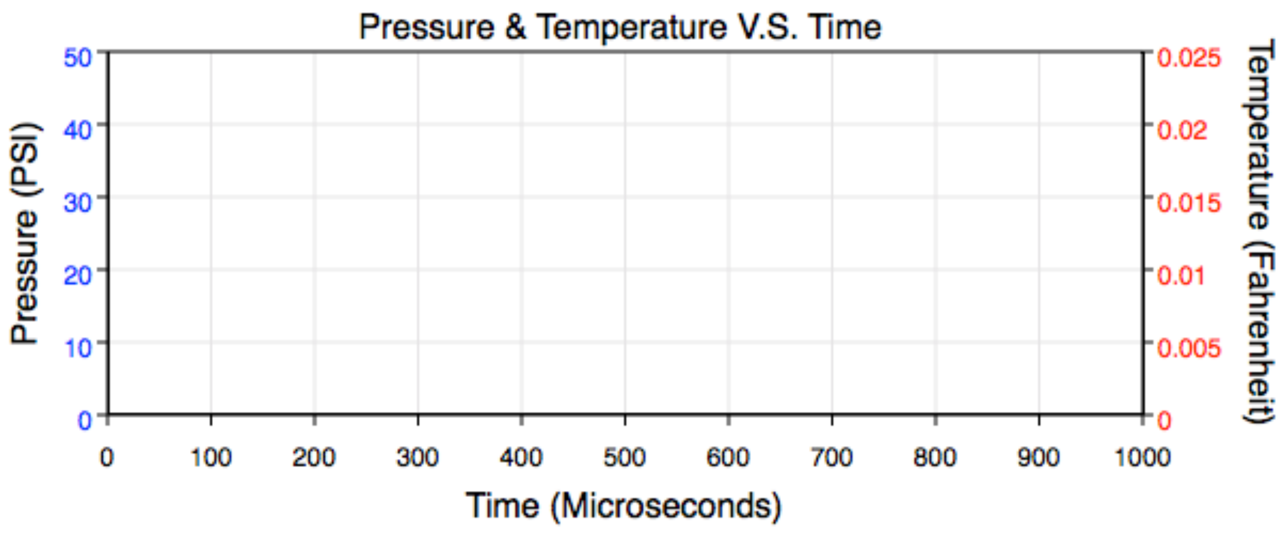
Regardless of the limitations of the above settings, once the legend is dragged onto the graphic view then it is independent and can be altered using Group Graphic controls. It can also be ungrouped, modified and regrouped for the most general alterations.

If the data graphics on the graph are altered then delete the previous legend and drag out the updated legend from this editor, i.e.: the legend is not attached to the data graphics once it is drag and dropped.

Graph IDE ▶ Graphs ▶ Multiple Coordinate Graph

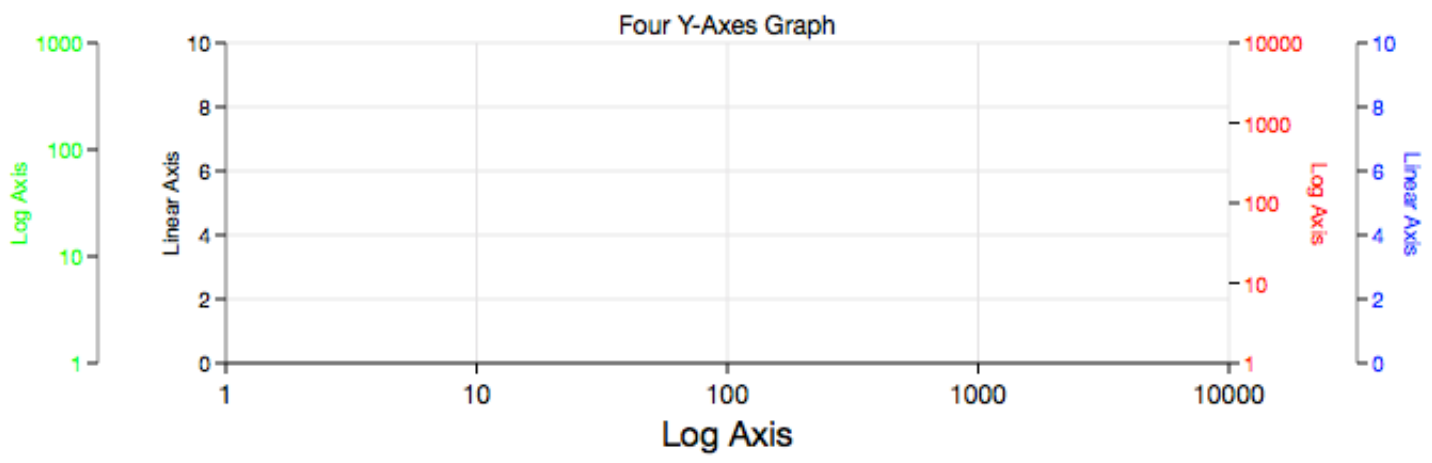
A Multiple Coordinate Graph is similar to the Single Coordinate Graph except the graph manages and shows more than one coordinate within its frame. The issue then becomes one of assigning data points to one coordinate v.s. another. For that purpose, the Multiple Coordinate Graph defines a focused coordinate index. That index must be set before creating data graphics on the graph.

The figure below shows a typical use of a multiple coordinate graph. The concept is to show how pressure changes in relation to temperature over time. The pressure would be shown by a blue curve, bar, etc. and the temperature by a red curve, bar or other graphical element(s), thus color distinguishes which coordinate the data goes with. Another way to view this data would be with a Single Coordinate Graph showing pressure v.s. temperature with a trajectory graphic where the parametric variable is time. There are reasons to show both formats.



Note that the scope of this implementation is limited by Graph IDE's User Interface to only rectangular coordinates and one x-axis. This may be relaxed to permit many more combinations of coordinates in the future.

The following figure shows 4 y-axes and one log x-axis. In all, there can be up to 16 y-axes of type None, Log, Linear and one x-axis of type Linear, Log or Date. That gives hundreds of permutations of coordinate types.



Inspector Editor

The Metrics Editor for the Multiple Coordinate Graph is described below. Notice how this description is the same as for the Single Coordinate Graph except for the coordinate selection controls.

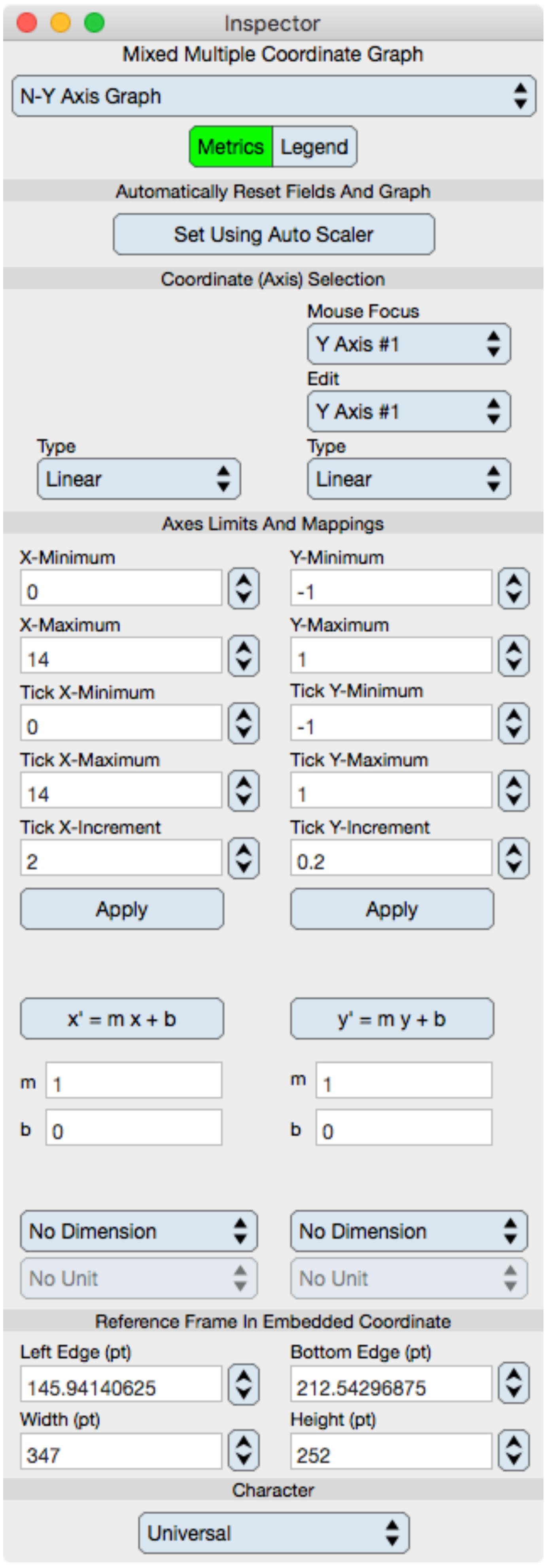
Automatically Reset Fields And Graph

Set Using Auto Scaler : Selecting this button autoscales the graph according to the settings in the Auto Scaler inspector.

Coordinate Axis Selection

Mouse Focus : You focus on a coordinate system to add data and graphics to that coordinate system which is represented and displayed by the graph's axes. In the case of multiple coordinates on one area you first need to define the coordinate to focus on. Use this setting for that purpose. Because there is only one x-axis the setting relates to one of 16 y-axis.

Edit : You can edit any one of sixteen separate y-axis limits. To do so, first select the y-axis to edit. Y-axis are ordered by closest to the graph frame to furthest away. So, left y-axis, right y-axis, 2nd left-y-axis, 2nd right-y-axis and so on for eight y-axes on the left and eight on the right. Once the Axis Number To Edit is set then the limits entries are for a single coordinate. For a definition of those limits entries see



Inspector

Mixed Multiple Coordinate Graph

N-Y Axis Graph

Metrics | Legend

Automatically Reset Fields And Graph

Set Using Auto Scaler

Coordinate (Axis) Selection

Mouse Focus: Y Axis #1

Edit: Y Axis #1

Type: Linear | Type: Linear

Axes Limits And Mappings

X-Minimum: 0	Y-Minimum: -1
X-Maximum: 14	Y-Maximum: 1
Tick X-Minimum: 0	Tick Y-Minimum: -1
Tick X-Maximum: 14	Tick Y-Maximum: 1
Tick X-Increment: 2	Tick Y-Increment: 0.2
Apply	Apply
x' = m x + b	y' = m y + b
m: 1	m: 1
b: 0	b: 0
No Dimension	No Dimension
No Unit	No Unit

Reference Frame In Embedded Coordinate

Left Edge (pt): 145.94140625	Bottom Edge (pt): 212.54296875
Width (pt): 347	Height (pt): 252

Character

Universal

the Single Coordinate Graph section.

Type : The type of either the x-axis or y-axis. The Type can by Linear, Log or in the case of the y-axis None. When the type is changed, the graphics for that coordinate are remapped and the Axes Limits And Mapping controls are changed to reflect the new type. If the type None is selected then graphics associated with that axis are deleted from the coordinate.

Axes Limits And Mappings

This specifies the actual limits of the coordinate system represented by the graph and those limits are defined at the graph frame. When you change these limits then you should also probably change the tick discretization limits to the same values. Note that the ticks "float" which is to say that they don't follow the Indigenous Coordinate Limits.

Apply : Updates the graph to correspond to the values shown above the button. Note that entering a value in a text field will not change the graph even if the return key is clicked. That is because all of the numbers are related and updating the graph based upon only one of the values can produce a inconsistent state.

Units

The Indigenous Coordinates and Tick Discretization are all defined in their own units separate from any typographical requirement. Set that unit using the Unit Selector available on the inspector. Units appear in the Cursor Information and coordinate with the Spreadsheet. Units are informational only and will not transform the data on the graph or set the coordinate mapping.

Reference Frame

This specifies the reference frame location of the graph. Note that the reference frame is the frame that coincides with the Indigenous Coordinate Limits location but in graphic view coordinate.

Left Edge : The left edge of the graph in units and coordinate of the Graphic View.

Bottom Edge : The bottom edge of the graph in units and coordinate of the Graphic View.

Width : The width of the graph in units and coordinate of the Graphic View.

Height : The height of the graph in units and coordinate of the Graphic View.

Character

One of the difficulties of a multiple coordinate graph is the sheer amount of attributes to set. Much like a Spreadsheet is used to control other representations, the multiple coordinate graph can also be controlled by a data source. One such data source is a Strip Chart where time series data (and transformed spectral data) can show up on a multiple coordinate graph. To facilitate that notion, the Character can be set.

Setting the Character to Strip Chart places the multiple coordinate graph under different input control than explained in this manual. For additional information see the Strip Chart User Manual.

Character : To set the character select the Character pop up button and then choose the Strip Chart character. Once set the Strip Chart inspector editor will be used. The Character can be reset from there by choosing the Universal character.

Legend

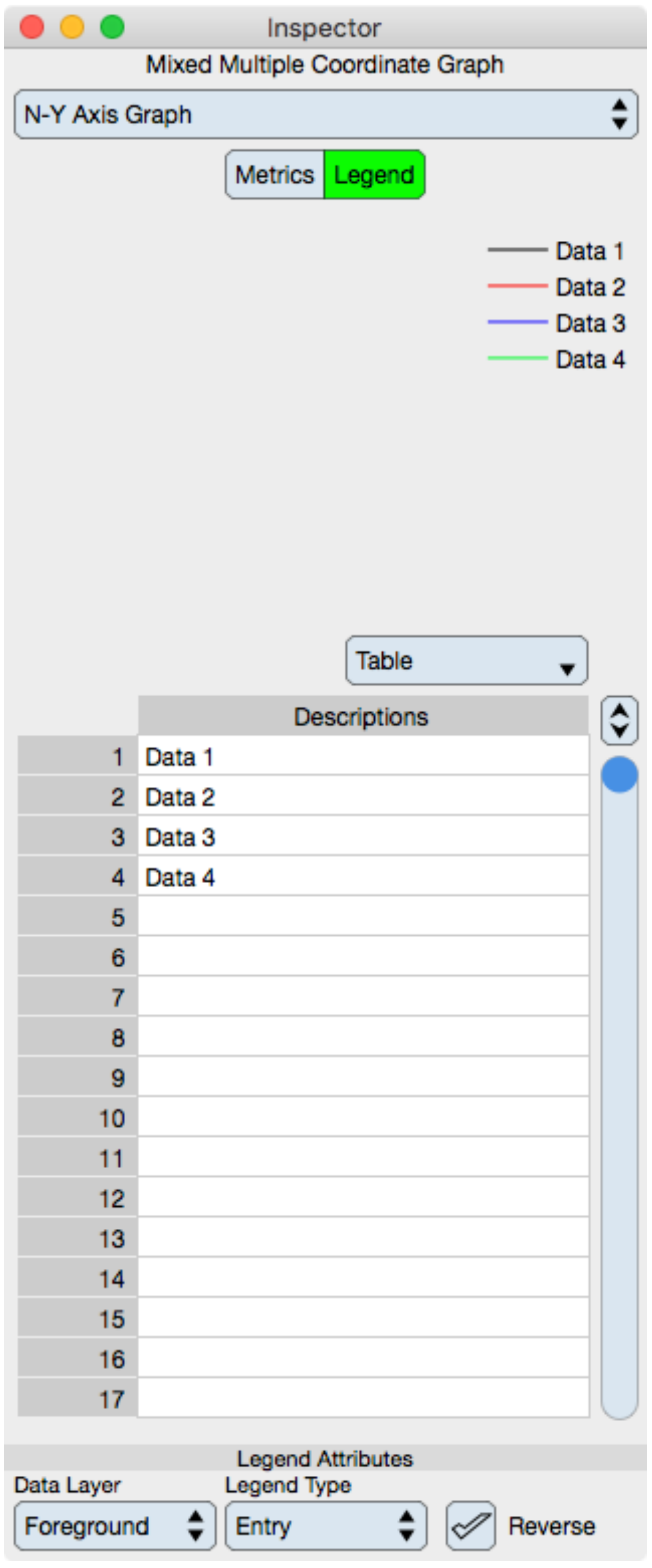
The Legend Editor for a Graph is described below. Legends are built from graphics on the graph's data layer so first place graphics on the graph, for example a Function (Curve) and then use this editor. Notice how this description is the same as for the Single Coordinate Graph. That is because although each graphic can be placed in a different coordinate system, those graphics are still only placed in the background or foreground data layer of the graph, just like for the single coordinate system situation.

Legend

Legend Group : The legend is a Group Graphic and is built automatically using the attributes specified below. Once the attributes below are specified then drag the legend near your graph on the document's Graphic View as it is also an element of a Palette. The legend is set to drag and drop to the overlay layer of the graphic view and not the data layer of the graph.

Table

Descriptions : Each row of a legend has a description which can be edited into the text of the descriptions table. Those entries are stored in



	Descriptions
1	Data 1
2	Data 2
3	Data 3
4	Data 4
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	

Legend Attributes
Data Layer: Foreground
Legend Type: Entry
✓ Reverse

the description key of the graphic resource dictionary whose entry can be modified directly via the Expert inspector.

Legend Attributes

Data Layer : The legend is built from either the foreground or background data layer of the graph, but not both. See Layer for additional information.

Legend Type : Either Entry, Circle or Square. If circle or square then the legend marker is represented by a circle or rectangle respectively whose fill color is that of the represented graphic. If Entry then the marker queues of the marker, stroke or fill of the represented graphic.

Reverse : The row order of the legend is the draw order of the data layer. It makes sense that the draw order of the data goes from minimum to maximum value which is bottom to top but legends are constructed top to bottom. Reversing the legend order makes the legend appear be in the same order as the data magnitudes, which is a heuristic effect.

Regardless of the limitations of the above settings, once the legend is dragged onto the graphic view then it is independent and can be altered using Group Graphic controls. It can also be ungrouped, modified and regrouped for the most general alterations.

If the data graphics on the graph are altered then delete the previous legend and drag out the updated legend from this editor, i.e.: the legend is not attached to the data graphics once it is drag and dropped.

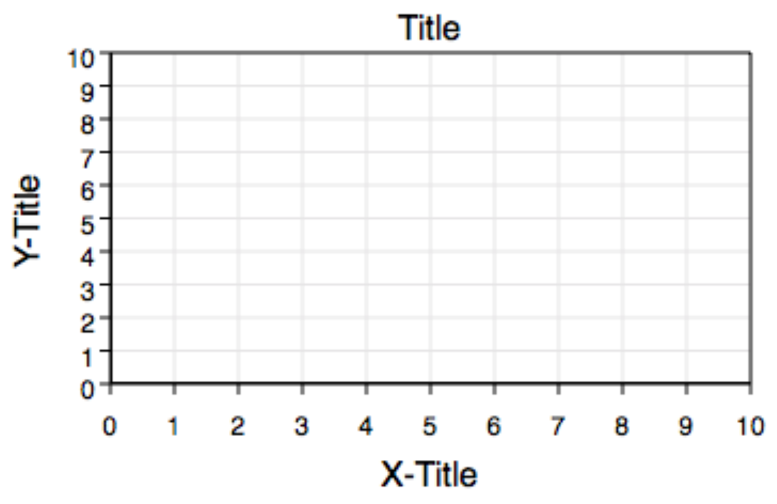
Graph IDE ► Graphs ► Getting Data On A Graph

You should remember that Graph IDE is not only a data-oriented application. You do not have to start with data and then push buttons to see representations. You can draw the representation first and then embed data-oriented or even graphic-oriented "Vvidgets" which represent what you want. If you wish to work the other way around then see the primer Importing Data.

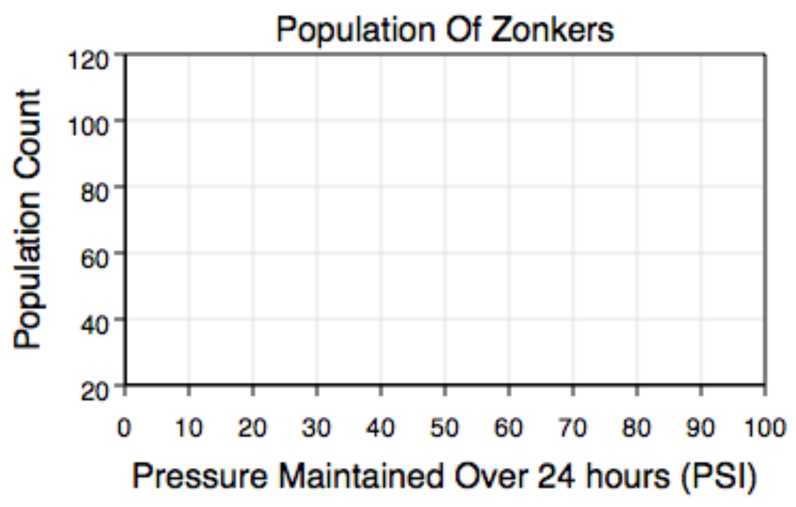
After drawing the representation then you can speed up the process considerably by putting your entire figure on a palette or using it as a template for a programmatic implementation.

What follows is a step-by-step example of making a typical graph.

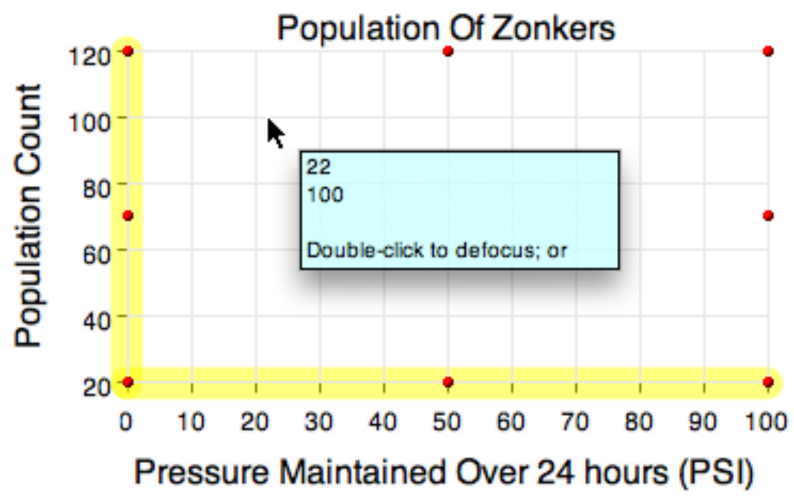
Step 1: Click the graph type you would like from the Graphic Selector and then drag it out on the Graphic View of your Graph IDE document. You will end up with a very basic graph as shown below.



Step 2: Use the graph's main Inspector Editor to change the axis limit and tick values and the graph's Titles & Labels sub-editor to change the graph title and axis titles to reflect the data you would like to import.



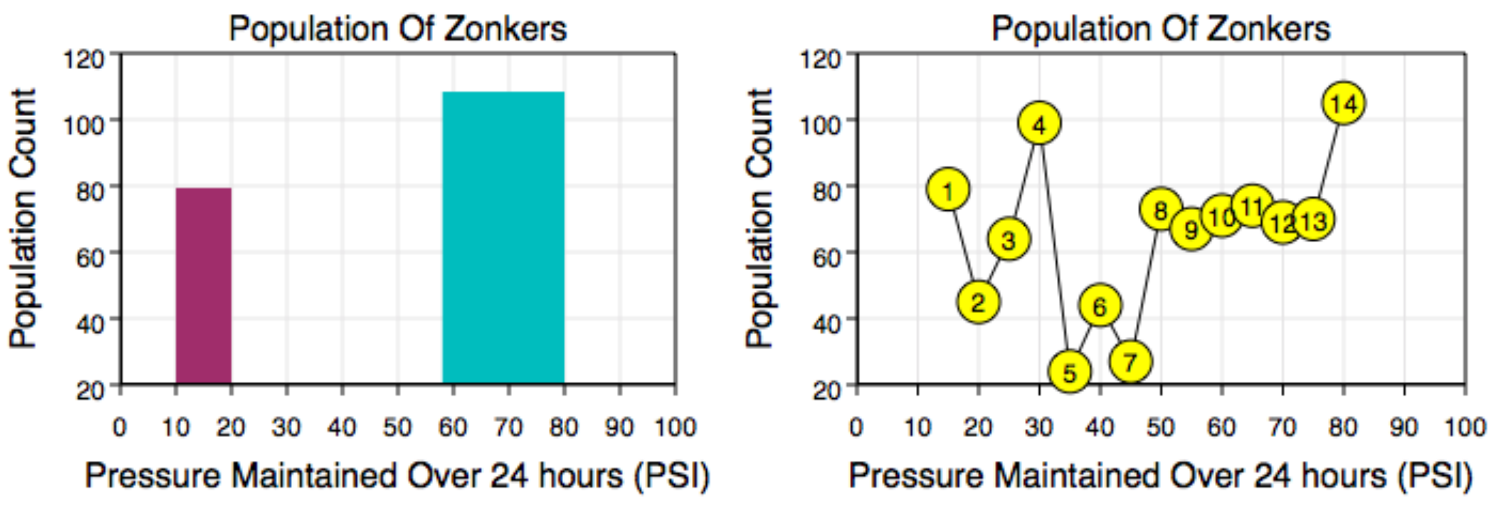
Step 3: Select the selection cell from the Graphic Selector and then double click within the graph frame to focus on it. The focus path you just established is diagrammed in the Layer section. You will end up with a graph that looks like the following:


Population Of Zonkers
Population Count
Pressure Maintained Over 24 hours (PSI)
22
100
Double-click to defocus; or

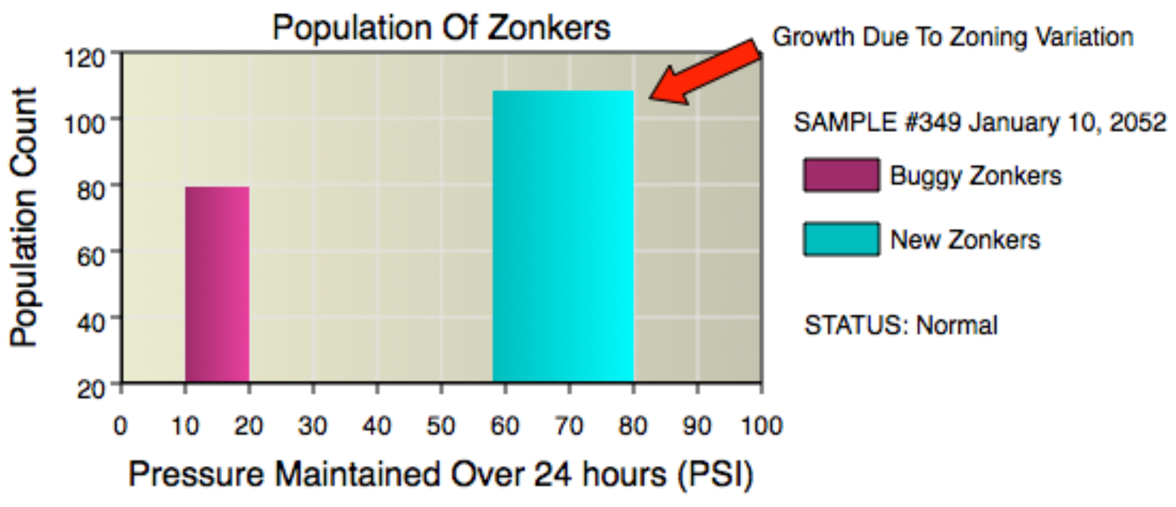
Notice the focus highlighters (yellow portion) on the baseline of both axes. This indicates the fact that any graphic you add will be associated with the coordinate represented by those axes. In a multiple coordinate graph which axis is highlighted is important and will tell you which coordinate you are focused on. Also note the Cursor Information Panel. The values it reports are in the dimension of the graph.

Step 4: Add the graphics you want. In the case on the left below, two Rectangles are added to show the fact that the data spanned 10 to 20 PSI and also 58 to 80 PSI and only one data point per pressure range is reported (such as an average of data over the interval of the respective domain). In the case on the right there is more data so it makes sense to use a Function graphic which has points as an attribute. The points can be entered as sequential white spaced x y values ordered from left to right (lowest to highest x-values).

After you add the graphics, then select the selection cell from the Graphic Selector again and double click to defocus from the graph's data layer. The focus highlighters will go away.



Step 5: After adding data and defocusing on the graph you can then add some supporting graphics to the same layer the graph is on (which is in the coordinate system of the page).

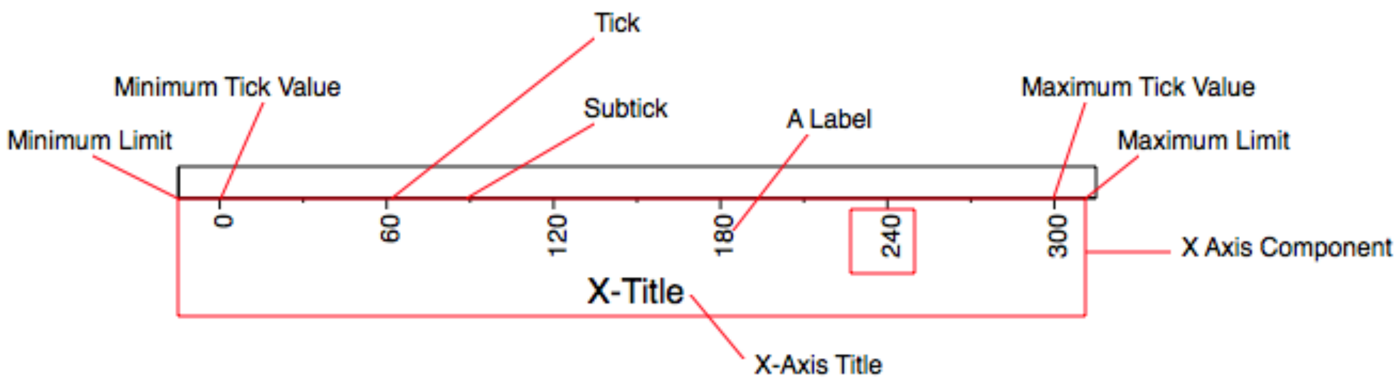


If you alter the graph such as moving it around or change its limits then the data representation (not the data values) will follow that change because you attached the graphical elements to the graph's data layers. Note that the data itself will remain intact, it is just its representation, i.e.: transformation from the graph coordinate to the page coordinate, that changes. You can refocus on the data graphics by double clicking as before, focus on them and change their graphical attributes. Navigating in and out of the graph's data layers is an important feature to understand.

Step 6: After making a few choice graphics you may want to put them on a Palette or even use the document they are defined on as a template for automation.

Graph IDE ► Graphs ► Linear Axis

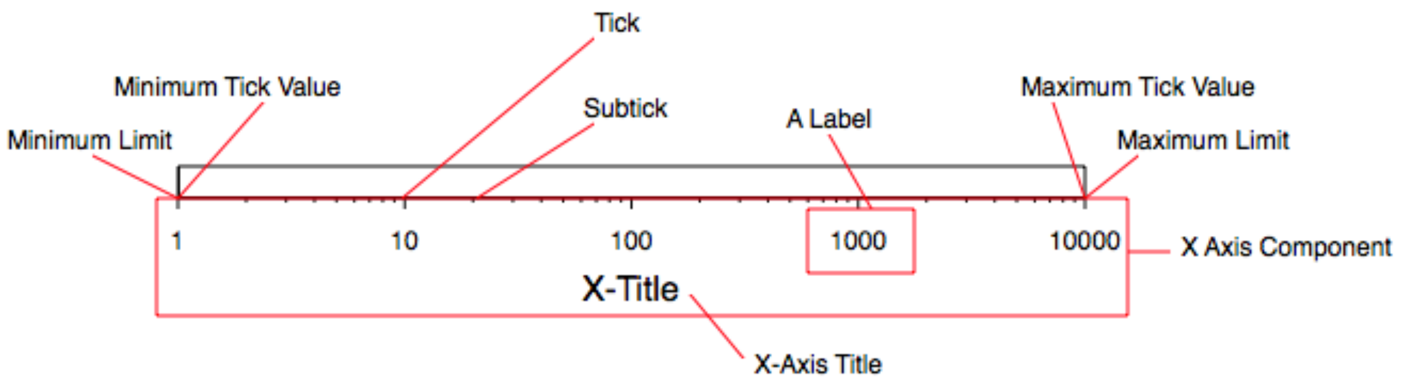
A typical graph has a linear axis. A linear axis is one that has uniform increments over the graph. It can be either x-oriented or y-oriented. The following figure shows an x-oriented linear axis.



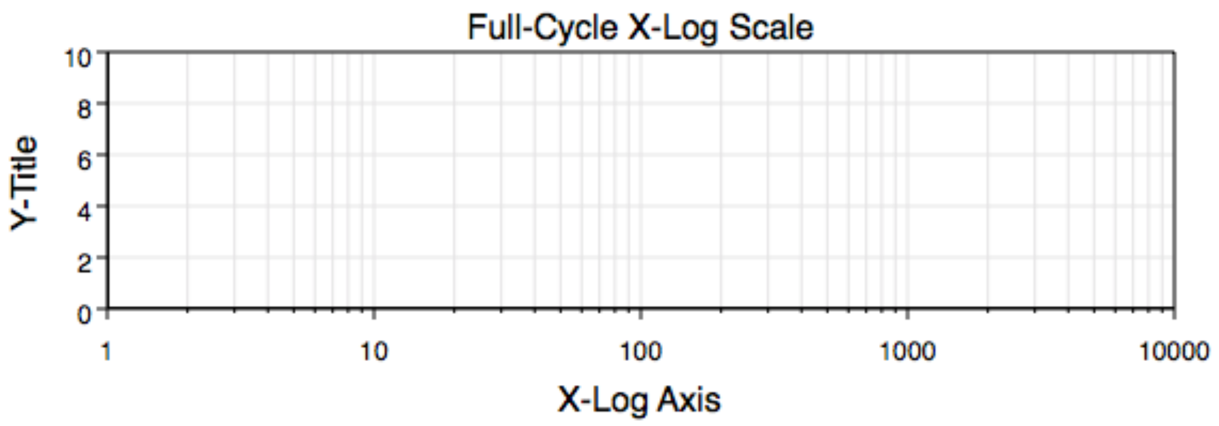
The axis has a lot of attributes, which are annotated in the figure above. At present, you can not actually make a axis, it can only be used as a component to a graph.

Graph IDE ► Graphs ► Log Axis

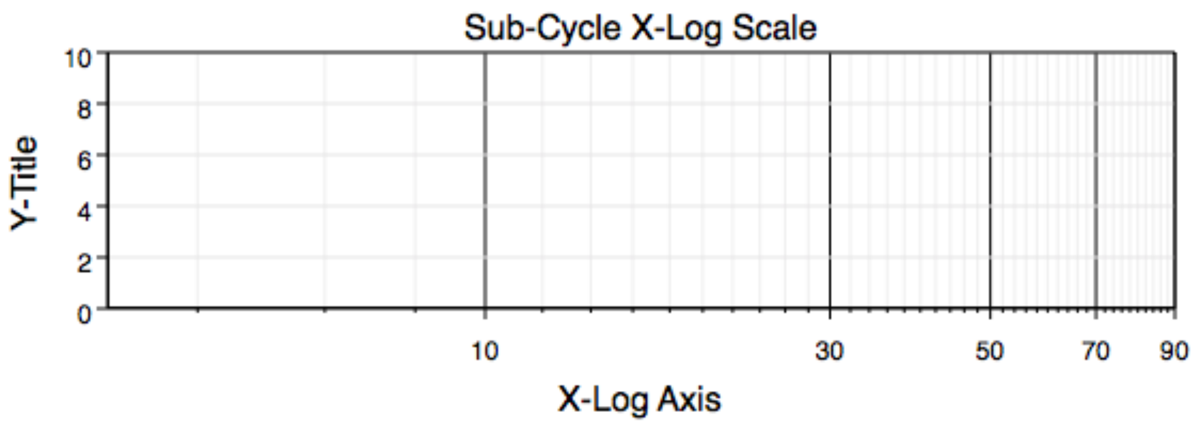
A log axis is one that has uniform log-dimension increments over the graph. It can be either x-oriented or y-oriented. The following figure shows a x-oriented log axis.



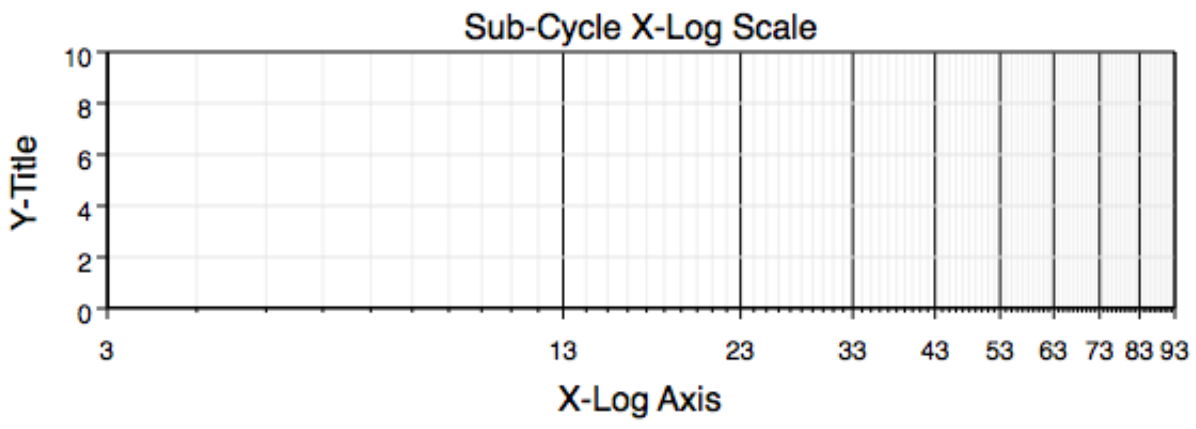
The most common type of log axis is the full-cycle format which has uniform increments in the log-range space and has ticks at integral values of that space.



If you specify a axis values less than 1.8 cycles in the log-range space then the axis will format as a sub-cycle graph where the major tick increments are uniform in the domain space of the log mapping. The figure below shows a graph whose x-axis is in sub-cycle format:



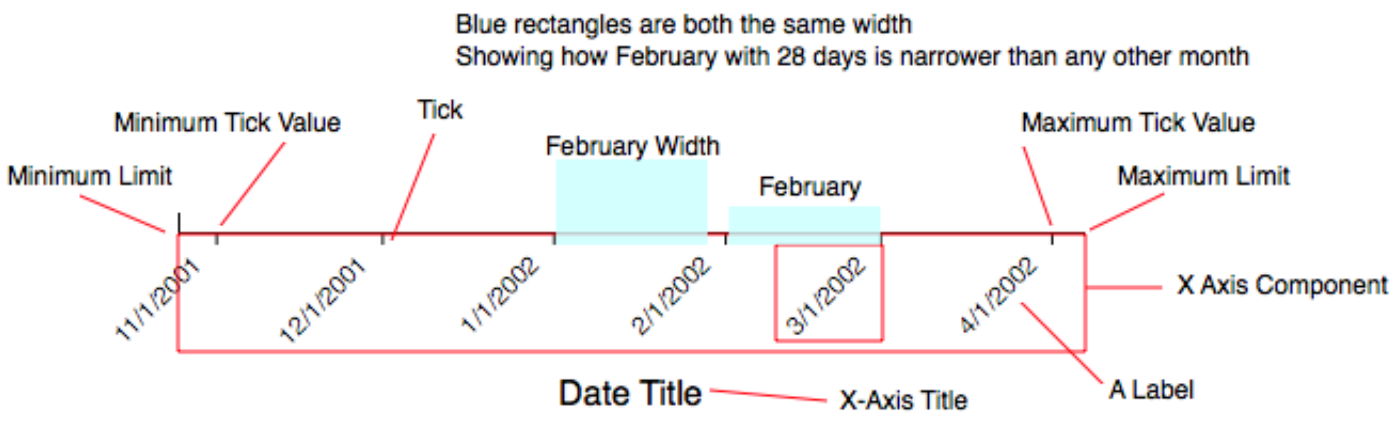
The x-axis of the following graph is also in sub-cycle format, but with major ticks starting at the axis limits.



When entering axis-tick values in the inspector editor make sure to enter the values in linear units, except for the tick increment in full-cycle format. That increment is specified in the range unit of the log mapping. That is the only way to specify a uniform increment for the full-cycle format.

Graph IDE ► Graphs ► Date Axis

A date axis is one that has date-unit increments which are not uniform if the unit is year or month, but is uniform if the unit is second, day, or the like. The date axis is constrained to be x-oriented only because we have seen no case of a y-oriented date axis. The following figure shows an x-oriented date axis.

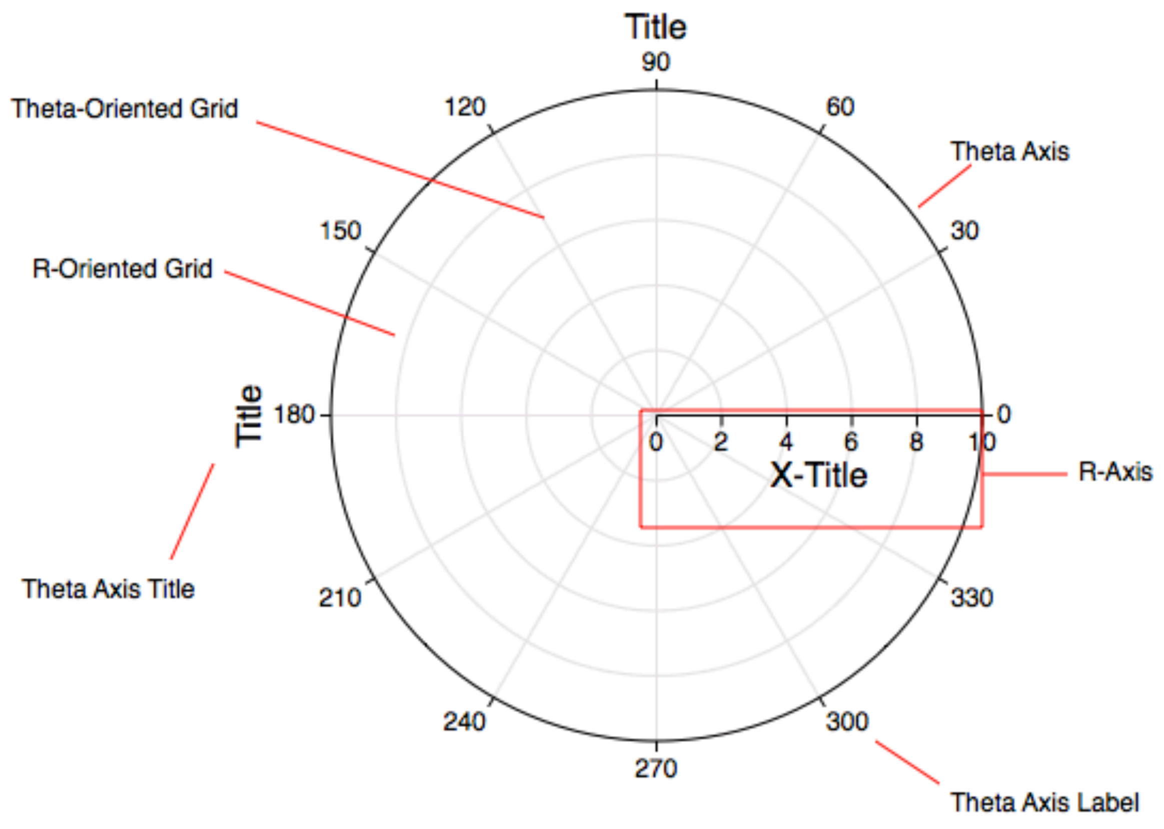


Note the blue rectangular areas showing how February is less in width than January.

The axis has a lot of attributes, which are annotated in the figure above. At present, you can not actually make a axis, it can only be used as a component to a graph.

Graph IDE ▶ Graphs ▶ Polar Axes

Polar Axes consists of radial and circular grid coordinate lines and is diagram in the following figure.



Polar axis are unlike rectilinear coordinates in that they show one representation, but you must be aware of three representations. For more information on non-linear representations see Non Linear Graphs.

Graph IDE ► Graphs ► Auto Scaler

The Auto Scaler changes the graph's limits, tick locations and some other attributes based on the values of the data on the graph. There are a few reasons to use the autoscaler:

- When you paste data into a data graphic, for example point data in a Function graphic inspector editor, then the data may be out of the current bounds of the graph. In order to correct this you can apply the autoscaler in the graph's inspector editor.
- If you program graphs using Vvidget Code or Vvidget Server then the graph autoscale parameters can be set in the template file.
- Animated graphs can produce data that goes out of bounds and the autoscaler can be used to automatically reset the graph limits to encompass the data during the animation.
- Multiple coordinate graphs can have many coordinate limits and using the Set Using Auto Scaler button on the main inspector editor will reset all limits at once to limits that are likely desired.

The Chart Tasks autoscale is based on autoscale parameters in a template so when you enter data you do not have to adjust the graph limits. If you create a graph in Graph IDE then you can explicitly set the graph metric parameters in the inspector editor and the autoscaler is a convenience algorithm that will probably get you very close to what you want. If you use Vvidget Code then the autoscaler becomes much more important because manually adjusting metric parameters during automation is not practical.

Inspector Editor

The Inspector Editor for the autoscaler is shown below. This inspector editor is for linear axes.

Update Graph : Selecting this button autoscales the graph according to the following settings. If there is no data on the graph then the update button is disabled.

Type

Autoscale Type : Keep at None or Decimal. None disables the autoscaler while Decimal will autoscale the axis according to decimal intervals. In many cases, it make sense to set the autoscaler type to None in one dimension and to a value other than None in the other dimension.

Sign Type : One of Natural, Positive Only, Negative Only, Symmetric. Natural is the usual value and will autoscale to include all data, Positive Only will set the axis limits only to positive values, Negative Only will set the axis limits only to negative values, Symmetric will set the axis limits to the same absolute value but with opposite signs and will autoscale to include all data.

Limits (both Minimum and Maximum)

Data : When the data limits row is checked then the actual data limits are used for autoscaler computation. There is a separate setting for minimum and maximum limits.

Fixed : When the fixed limits row is checked then value entered is used for autoscaler computation. There is a separate setting for minimum and maximum limits. Choosing the fixed limits deselects the data limits and visa versa.

Gap : The gap is an integer that corresponds to the number of tick intervals that are padded in the autoscaler computation. The gap is usually zero, but sometimes is one or more.

Align : Align is one of None, Tick, Subtick, Data. None is a technicality, Tick means align the axis limit to whole tick values, Subtick means align the axis limit to whole subtick values and Data means align the axis limits to data values.

Components To Update

The autoscaler will update all graph and axis components to make those components consistent, unless those updates are turned off. Those components are Ticks, Subticks, Grid and Label Format.

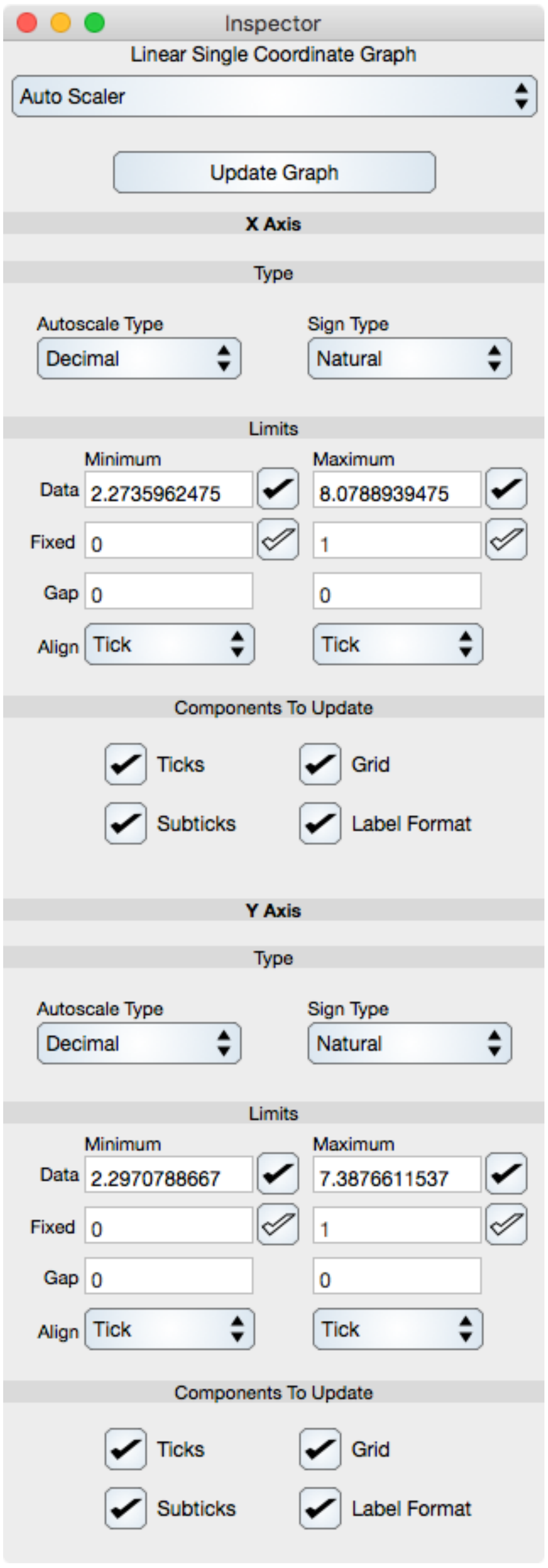
Note: The following discussion is based upon heuristics as desired results are dependent upon user preferences. As such, they may not be exactly what you are looking for and they also might not reflect the actual performance.

Ticks : Update the ticks. The autoscaler updates the graph limits and the ticks can float away from those limits. By letting the autoscaler update the ticks the limits and ticks will be coincidental, which is probably the way you are use to seeing them.

Subticks : The autoscaler can make the tick increment very different from its previous value. The subtick count typically is dependent upon the tick increment. For example: If the increment is every five units then the subtick count would most likely need to be 0, 1 or 4. Hence, in some cases the subtick count needs to be a function of the tick count.

Grid : With almost all cases, the grid discretization should follow the tick discretization so that the grid update should be left checked (on).

Label Format : Sometimes the autoscaler will remap the coordinate system to disparate ranges and in this case the Label Format may need to change, say from decimal notation to scientific notation. Leaving this selection checked (on) permits the autoscaler to account for that variation.


Inspector

Linear Single Coordinate Graph

Auto Scaler

Update Graph

X Axis

Type

Autoscale Type	Sign Type
Decimal	Natural

Limits

	Minimum		Maximum	
Data	2.2735962475	✓	8.0788939475	✓
Fixed	0	✓	1	✓
Gap	0		0	
Align	Tick		Tick	

Components To Update

✓ Ticks	✓ Grid
✓ Subticks	✓ Label Format

Y Axis

Type

Autoscale Type	Sign Type
Decimal	Natural

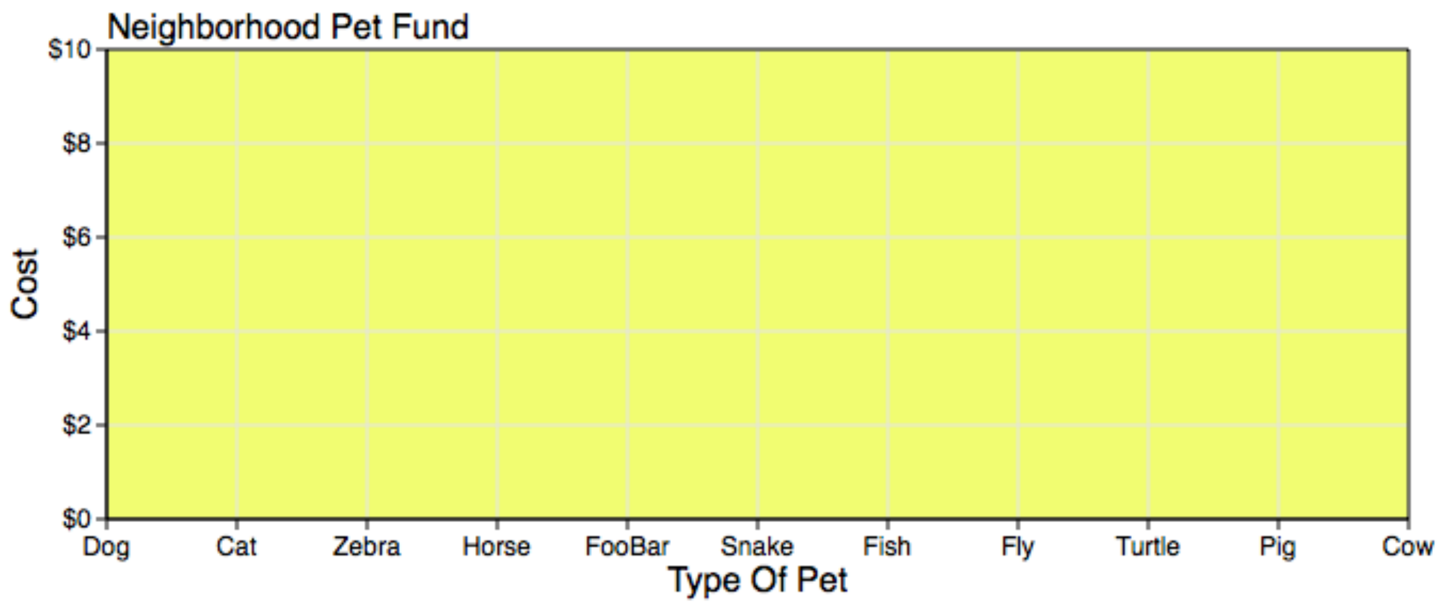
Limits

	Minimum		Maximum	
Data	2.2970788667	✓	7.3876611537	✓
Fixed	0	✓	1	✓
Gap	0		0	
Align	Tick		Tick	

Components To Update

✓ Ticks	✓ Grid
✓ Subticks	✓ Label Format

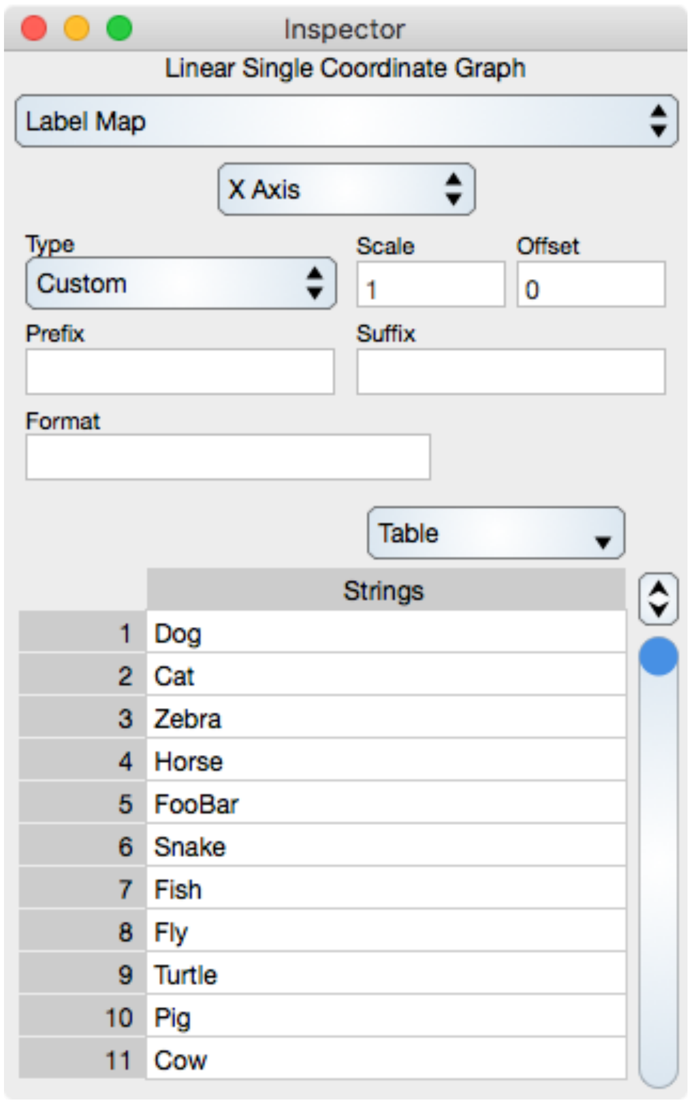
A Label Mapping changes the text shown in the labels of the axes of a graph, but does not change the coordinate system of the graph and does not change the data. The figure below shows a graph with labels mapped to text on the x-axis and to have a dollar sign as a prefix of the value on the y-axis.



The label mapping also accepts date formatting, scaling and offset values.

Inspector Editor

The Inspector Editor for the Label Mapping is shown below.



Type: Custom
Scale: 1
Offset: 0
Prefix:
Suffix:
Format:

	Strings
1	Dog
2	Cat
3	Zebra
4	Horse
5	FooBar
6	Snake
7	Fish
8	Fly
9	Turtle
10	Pig
11	Cow

Table

Focused Axis : Sets the axis to be updated by all the other controls.

Type : Defines the label mapping type. It is one of None, Normal Number Mapping, Fixes Only, Data or Custom.

Scale : Defines the numeric scale of the mapping. See Offset for additional information.

Offset : Defines the numeric offset of the mapping. When the mapping type is Normal Number Mapping then the numeric values of the axis are mapped according to: x' = scale • x + offset. That type of linear mapping is useful for a variety of graphs. For example, if data is specified in dollars but you want to show the axis in units of cents then offset is zero and scale is 100. Setting the prefix string to the dollar symbol completes the example.

Prefix : Defines a prefix string. The prefix appears before the mapped axis label except when custom labels are used.

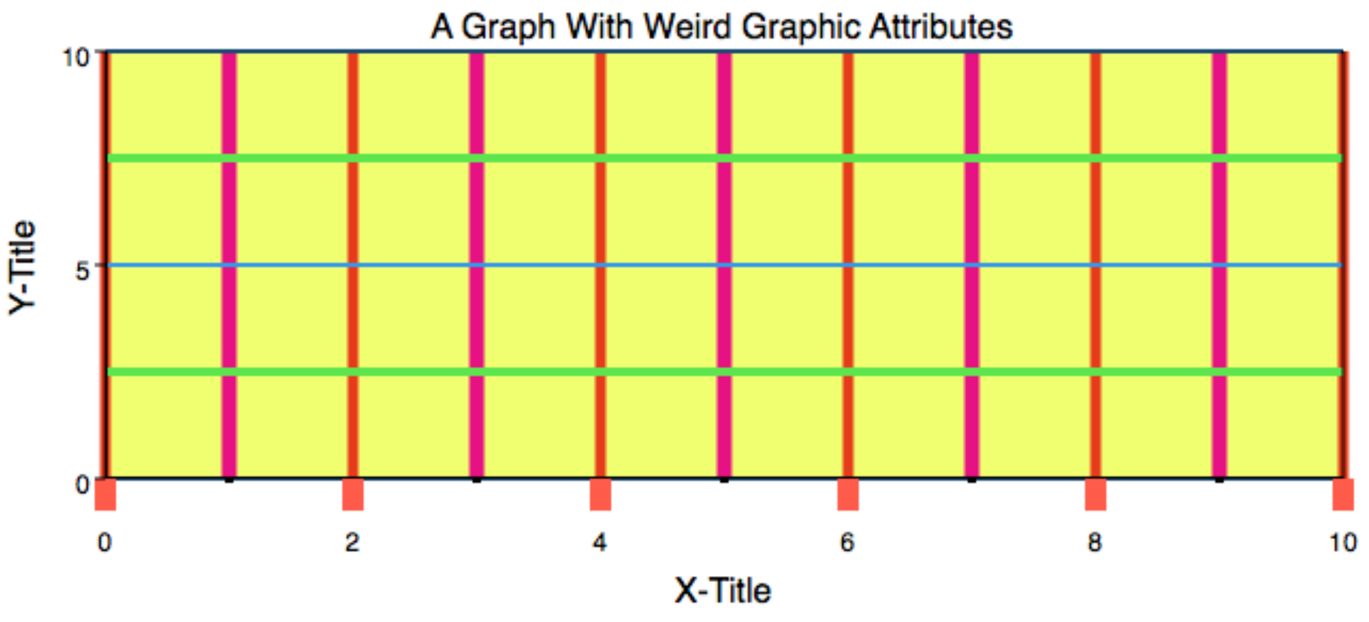
Suffix : Defines a suffix string. The suffix appears after the mapped axis label except when custom labels are used.

Format : Defines a format string for the mapping.

Strings : A Table that sets the label values according to arbitrary strings, one label per line. When custom labels are used then all other label mapping attributes are ignored.

Graph IDE ► Graphs ► Graphic Attributes

A Graph can have many Graphic Attributes. The figure below shows some more unusual settings.



Please play around with the Graph's Graphic Inspector Sub-Editor to figure out all the options.

Inspector Editor

The Inspector Editor for the Graph Graphic Attributes is shown below.

Common Controls

The first portion controls controls common to all graphics which are described in the Graphics section.

In this case, the interior area is within the graph frame, which is coincidental with the axes limits. The stroke parameters adjust the stroke of the frame itself.

Axis Ticks

Draw : Defines whether the ticks are drawn or not.

Length : The length of the ticks.

Width : The width of the ticks.

Color : Sets the Color of the ticks.

Axis Subticks

Draw : Defines whether the subticks are drawn or not.

Length : The length of the subticks.

Width : The width of the subticks.

Count : The number of subticks between ticks. Notice how the number of ticks is derived from a discretization of the graph coordinate while the number of subticks is defined directly.

Color : Sets the Color of the subticks.

Axis Tickbase

The tickbase is normally coincidental with the graph frame and shown as a line segment. It is a defining location as well as a graphical parameter. Normally the tickbase does not even matter and it is redundant. However, occasionally the axis needs to be offset (relocated) away from the graph frame. Although this can be done for a single coordinate graph, it is most applicable to a multiple coordinate rectilinear graph where the y-axes (or x-axes) are offset from the graph frame. See Multiple Coordinate Graph for an example.

Draw : Defines whether the tickbase is drawn or not.

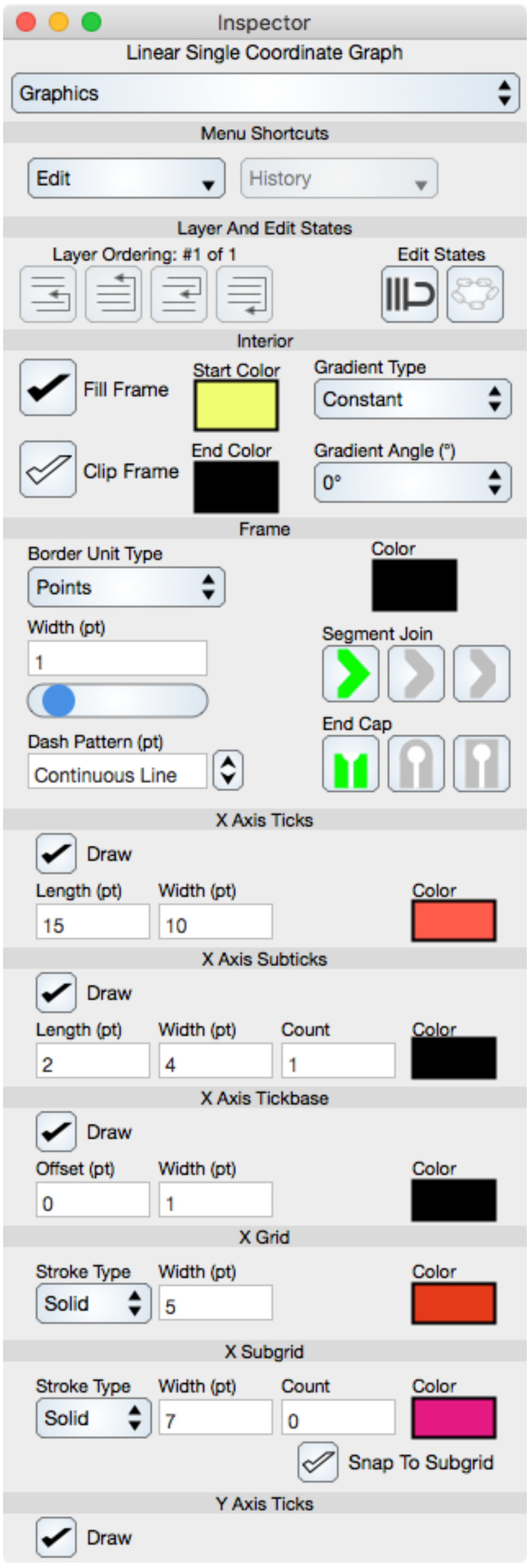
Offset : The offset of the tickbase relative to the graph frame.

Width : The width of the ticks.

Color : Sets the Color of the tickbase.

Grid

Stroke Type : Sets the dash pattern of the grid. Set this to Solid for a solid grid, None to turn the grid off or a pattern value for a particular dash.



Inspector
Linear Single Coordinate Graph
Graphics

Menu Shortcuts
Edit | History

Layer And Edit States
Layer Ordering: #1 of 1
Edit States

Interior
Fill Frame ✓ | Start Color | Gradient Type: Constant
Clip Frame | End Color | Gradient Angle (°): 0°

Frame
Border Unit Type: Points | Color
Width (pt): 1
Segment Join
Dash Pattern (pt): Continuous Line
End Cap

X Axis Ticks
Draw ✓
Length (pt): 15 | Width (pt): 10 | Color

X Axis Subticks
Draw ✓
Length (pt): 2 | Width (pt): 4 | Count: 1 | Color

X Axis Tickbase
Draw ✓
Offset (pt): 0 | Width (pt): 1 | Color

X Grid
Stroke Type: Solid | Width (pt): 5 | Color

X Subgrid
Stroke Type: Solid | Width (pt): 7 | Count: 0 | Color
Snap To Subgrid

Y Axis Ticks
Draw ✓

Width : The width of the grid strokes.

Color : Sets the Color of the grid.

Subgrid

Stroke Type : Sets the dash pattern of the subgrid. Set this to Solid for a solid subgrid, None to turn the subgrid off or a pattern value for a particular dash.

Width : The width of the subgrid.

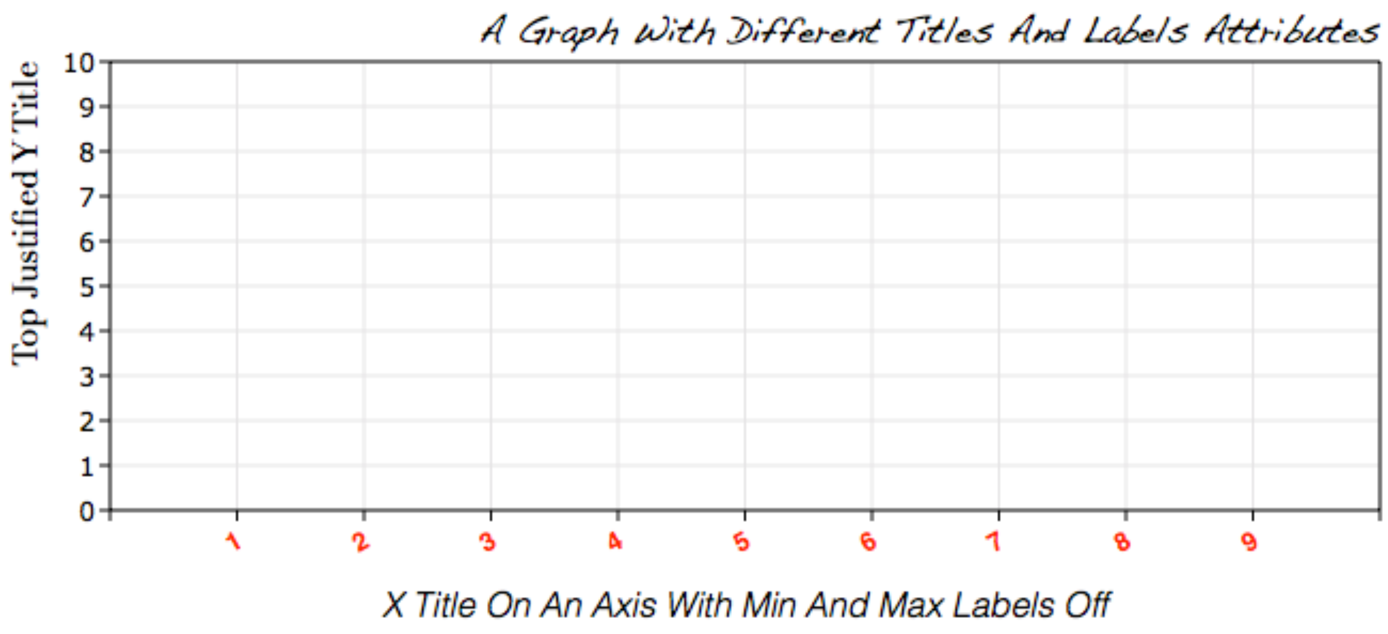
Count : The number of subgrids between grid lines.

Color : Sets the Color of the subgrid.

Snap To Subgrid : A flag used to determine if graphics aligned with the subgrid when they are moved. Normally this flag is set using the onboard document snap to grid button.

Graph IDE ► Graphs ► Titles & Labels

This section describes graph titles and labels and how to adjust them. The figure below shows some more unusual settings.



Please play around with the Graph's Title & Labels Inspector Sub-Editor to figure out all the options.

Fonts

To affect a change in a font first select the graph and then bring forward the font panel (command-t) and change the font. The font changed depends on the component of the graph hit. There are three component hit types:

Title:	If the main, x or y title is hit then a font change is associated with that hit title only.
Labels:	If an axis label is hit then the font for all the labels of that axis are changed.
Frame:	If the hit is in the interior of the graph frame then the labels for all axes are changed.

Inspector Editor

The Inspector Editor for the Titles And Labels is shown below.

Main Title

Draw : Determines if the main title is drawn or not.

Text : Shows and defines the title's text. Click Return to enter the text.

Font Name ; Size : Shows the title font name and size. Selecting this label brings up the Font selector. This label is right below the Text field entry and does not look like a control.

Alignment : Shows and defines the alignment of the title relative to the graph frame.

Text Color : Shows and defines the Color of the title text.

Background Color : Shows and defines the Color of the background of the title. The background is the graphical frame of the text and is coincidental with the bounds of the font metrics.

Draw : Determines if the main title background is drawn or not. This is next to the background color well.

Axis Title

Draw : Determines if the axis title is drawn or not.

Axis Title Formatter : Shows and sets a predefined axis suffix if the graph has a set unit. Select this button to bring forward the Axis Title Format Selector described below.

Text : Shows and defines the title's text. Click Return to enter the text.

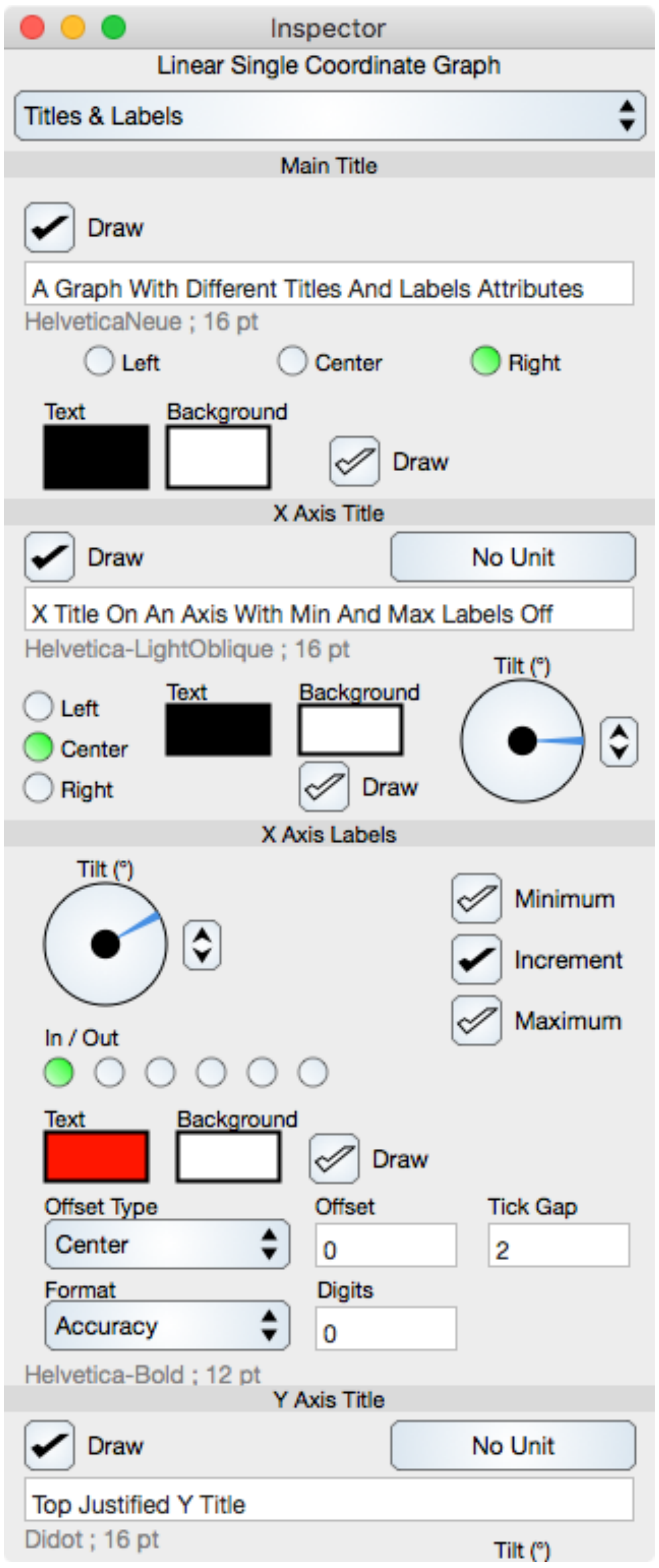
Font Name ; Size : Shows the title font name and size. Selecting this label brings up the Font selector. This label is right below the Text field entry and does not look like a control.

Alignment : Shows and defines the alignment of the title relative to the graph frame.

Text Color : Shows and defines the Color of the title text.

Background Color : Shows and defines the Color of the background of the title. The background is the graphical frame of the text and is coincidental with the bounds of the font metrics.

Draw : Determines if the main title background is drawn or not. This is next to the background color well.



A Graph With Different Titles And Labels Attributes
HelveticaNeue ; 16 pt

X Title On An Axis With Min And Max Labels Off
Helvetica-LightOblique ; 16 pt

Offset Type: Center
Offset: 0
Tick Gap: 2
Format: Accuracy
Digits: 0
Helvetica-Bold ; 12 pt

Top Justified Y Title
Didot ; 16 pt

Tilt Angle : Determines the tilt (rotation) of the labels. The dial control is described in the Dial section.

Axis Labels

Tilt Angle : Determines the tilt (rotation) of the labels. The dial control is described in the Dial section.

In / Out : Shows and defines the axis and hence label in /out positioning. Only one is selected at the time. For a y-axis rectilinear graph those are left-out, left-in, 0-value-left, 0-value-right, right-in and right-out. This is particularly applicable to a Multiple Coordinate Graph where axes are to the left and right of the graph frame.

Minimum, Increment, Maximum : Determines if the Minimum, Increment, Maximum labels are drawn or not.

Text : Shows and defines the title's text. Click Return to enter the text.

Text Color : Shows and defines the Color of the title text.

Background Color : Shows and defines the Color of the background of the title. The background is the graphical frame of the text and is coincidental with the bounds of the font metrics.

Draw : Determines if the label background is drawn or not. This is next to the background color well.

Offset Type : Determines the label position relative to the ticks.

Offset : Determines the label positional offset relative to the ticks.

Tick Gap : Determines the label gap between a tick and the label frame.

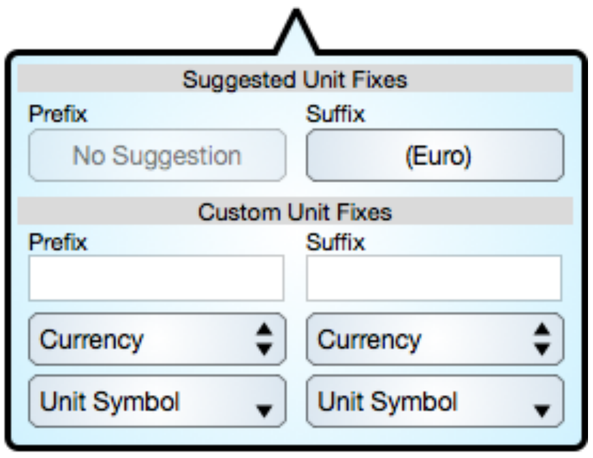
Format : Determines the label numeric format. This can be overridden by the LabelMapping.

Digits : Determines the number of significant digits to be used.

Font Name ; Size : Shows the title font name and size. Selecting this label brings up the Font selector. This label is right below the Text field entry and does not look like a control.

Axis Title Format Selector

The axis title format selector can be used to add suffixes to the axis title. If units are assigned to the graph then this formatter can make suggestions about the fixes to use.



Suggested Prefix : For an axis label this is always disabled as only the suffix is used.

Suggested Suffix : Selecting the suggested suffix will append the unit name in the format: ":Title (unit name)", i.e.: present axis title, then a space and then the parenthesized unit name.

Prefix : Text to be used as a custom prefix.

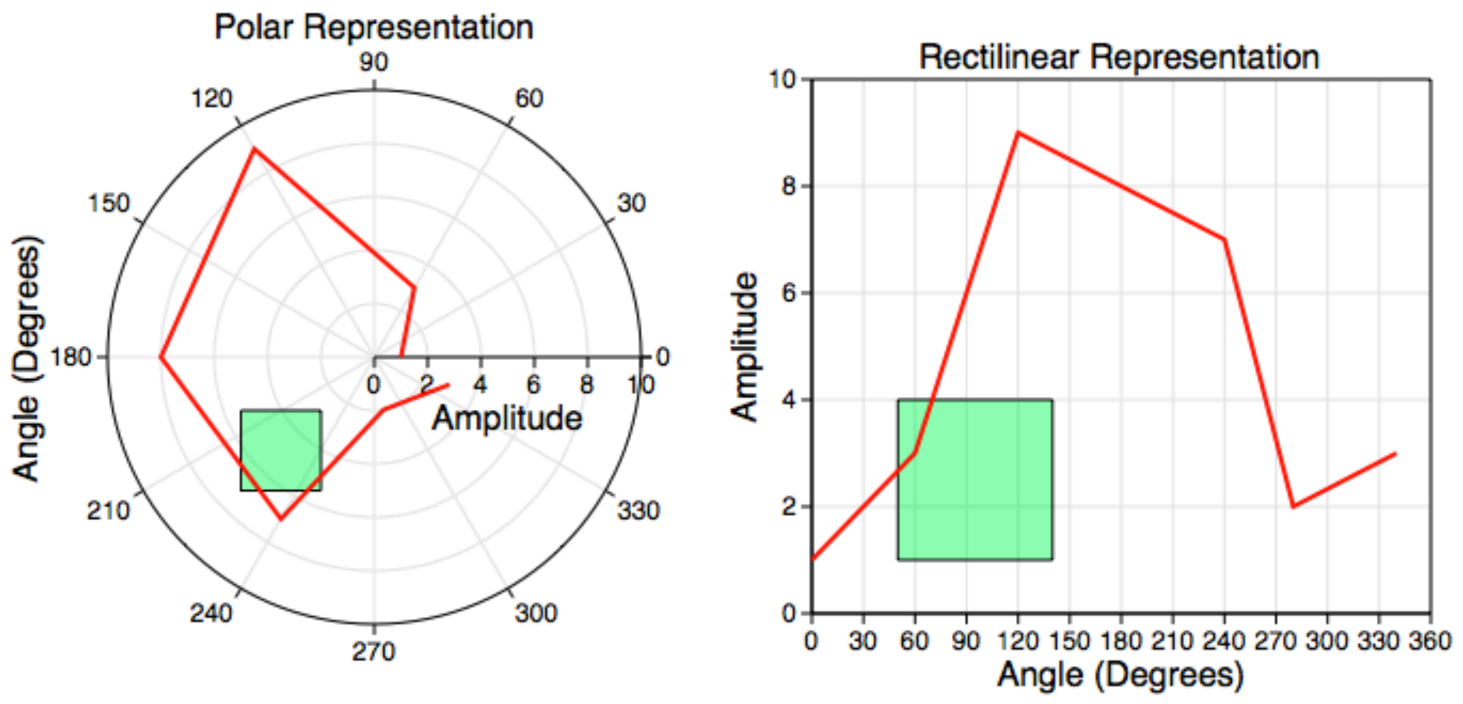
Suffix : Text to be used as a custom suffix.

Prefix Unit Selector : A Unit Selector used to prepend the unit symbol to the prefix text. This is a convienience method to unit symbols as the symbol can also be inserted directly using the prefix text field.

Suffix Unit Selector : A Unit Selector used to append the unit symbol to the suffix text. This is a convienience method to unit symbols as the symbol can also be inserted directly using the suffix text field.

Graph IDE ▶ Graphs ▶ Non-Linear Graphs

Non-Linear Graphs are graphs whose dimensions are not represented by uniformly spaced straight lines. For example, the polar graph shown here:



is considered non-linear while the rectilinear graph on the right is not. There are two graphics on each of those graphs, a green rectangle and a red sequence of line segments (a curve). The curve points are described in the circular and radial grid coordinate lines (namely {θ, r}) while the rectangle, which is a graphic-oriented graphic, is described in mapped rectilinear coordinates (namely {x, y}). The relationship is: r = sqrt(x x + y y) and θ = atan(y/x). The {θ, r} coordinate is called the domain space, while the {x, y} coordinate is called the range space.

So that the green rectangle has origin at point {-5, -5} and size {3, 3} for the left (polar) graph and origin at point {50, 1} and size {90, 3} for the right (rectilinear) graph. This is in contrast to the points that define the red curve. For both graphs the red curve has the same point values:

θ (Degrees)	Amplitude
0	1
60	3
120	9
180	8
240	7
280	2
340	3

That is because the curve is defined in the domain space of each graph, while the rectangle is defined in the range space of the graph. For general non-linear graphs there are three different representations (spaces) to be aware of:

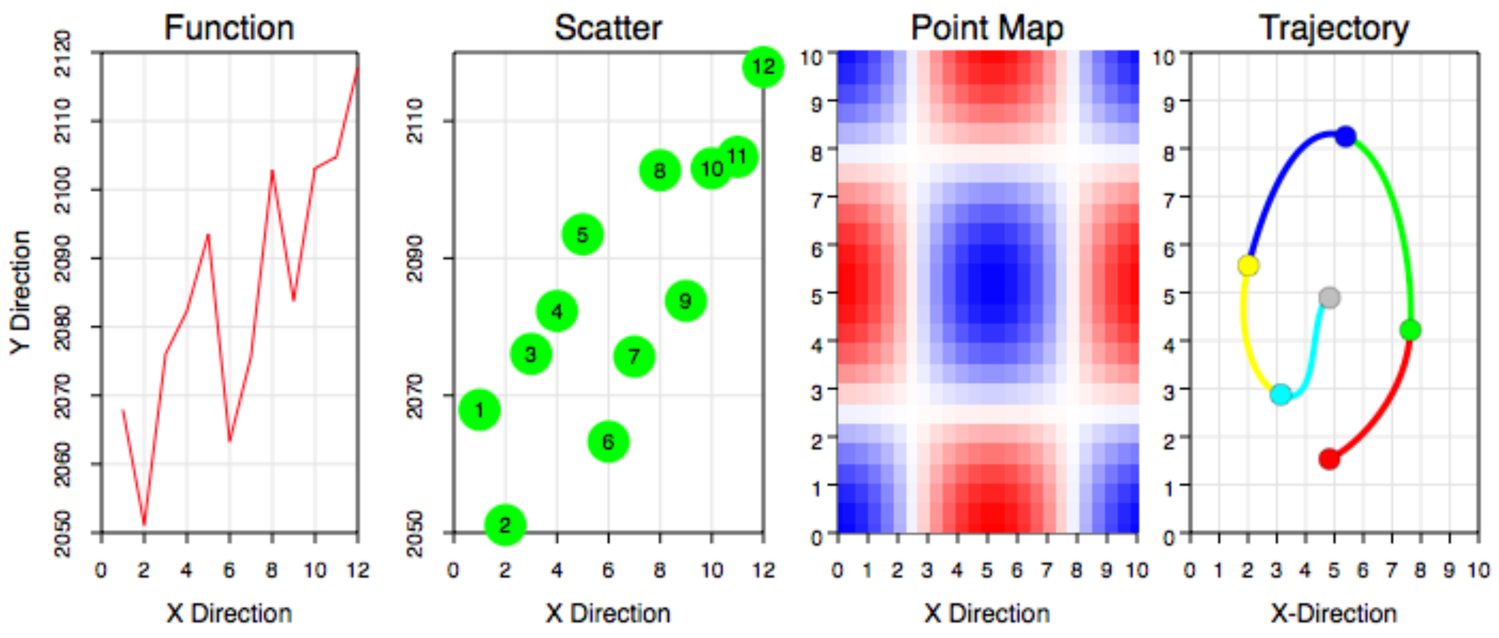
Representation Name	Description
Domain	This is the coordinate system as shown by the graph's grid lines and curves.
Range	This is the coordinate system after the non-linear mapping takes place and is a linear coordinate system. For example: xp = log(x).
Rectilinear	This is a secondary representation of either the domain or range representation. For polar coordinates this is another representation of the domain space, for log coordinates this is the range space.

The only coordinate system where the three representations described above are unique is the log-r polar graph. The only one where they are the same (and hence only the Domain Representation is considered) is the normal Linear Axis graph.

The important part of non-linear graphs is to keep the representations in your mind while working with the graph, and which representation corresponds to which graphic. When dealing with multiple y-axis or x-axis graphs with linear and nonlinear axis this issue needs to be well thought out because one area on the graph can have many coordinate systems and spaces.

Graph IDE ▶ Data Graphics

A data graphic is a graphic which is defined by points and has point-wise graphic representations. The following figures show examples of a data graphics in use.

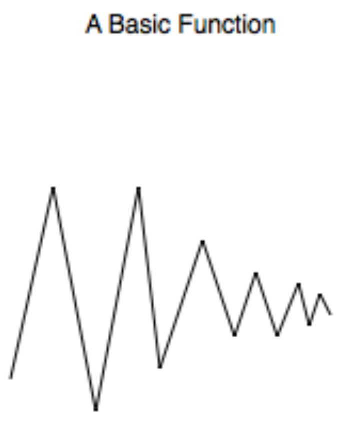


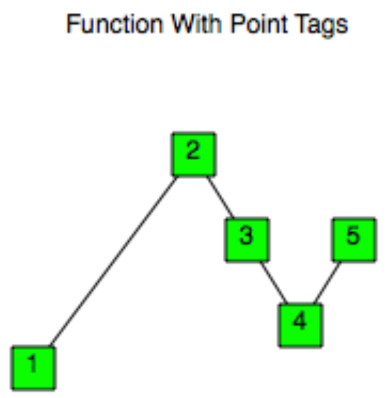
The following is a brief list and definition of data graphics:

Section	Description
Bar And Column Chart	A Bar And Column Chart is a sequence of rectangles whose length represent scalar values. The rectangle can also have an accompanying label.
Function	A function is a sequence of points whose x-values increase with sequence index.
Pie Chart	A Pie Chart is a sequence of circles whose wedge angles represent proportion of data. The wedge sections can also have an accompanying label.
Point Map	A Point Map is a regular grid of z-values. The z-values are represented by color. Each point can also have an angle in which case the Point Map represents a vector field on a regular grid.
Scatter	A scatter is a set of points either independent of each other, or relative to an origin which can be any point but can also be the mean of the data points.
Spreadsheet	A spreadsheet is a table with textual cell entries that are stored with the table.
Trajectory	A trajectory is a sequence of points connected by curves, each two consecutive points defines the end point of a Cubic Bezier section so that there are also two spline knot points between the points to form the spline, but are not considered part of the data proper.

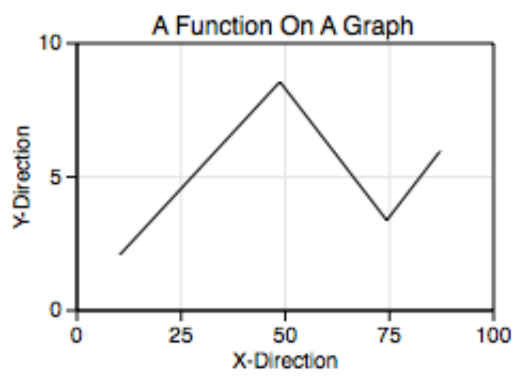
Graph IDE ▶ Data Graphics ▶ Function
a.k.a.: Curve or Line Graph

A function is a sequence of points connected by line segments. The x-values of the points increase as the sequence index increases. The figures below show examples of functions.



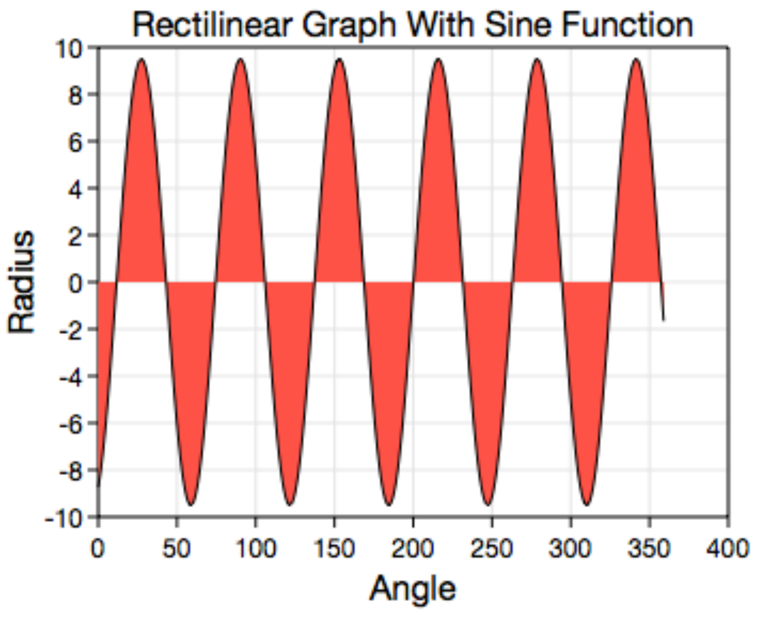


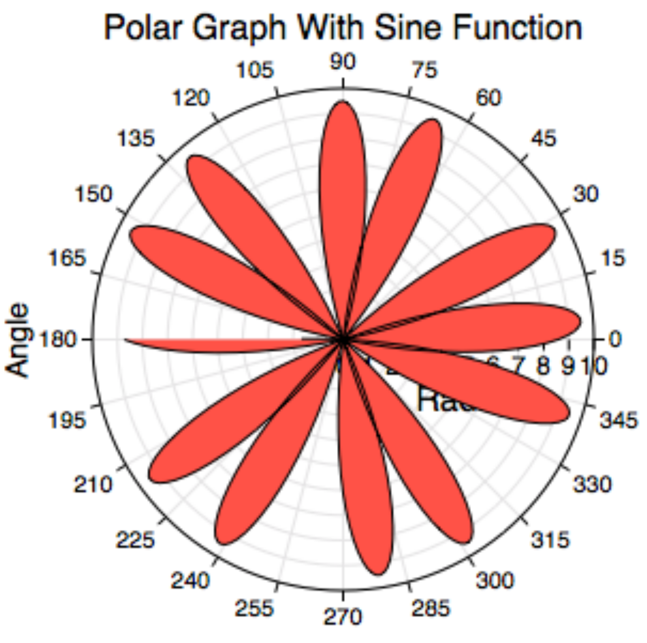
The function looks more familiar on a graph



Double-click on the graph to focus on it.
Create the function.
Then double click again to de-focus off the graph.
The function is attached to the graph.

All Data Graphics, including this function graphic, transform according to the coordinates of the graph it is on. So, for example, the following graphs show the same exact function graphic (and same x and y point data) but on two different coordinate types. In the polar graph, the radius is mapped with an absolute value function. For additional information consult the Non-Linear Graphs section.





Some standard operations are itemized below.

- To create a function bring forward the Graphic Selector, click the function factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of a predefined function graphic.
- You can also create functions from the standard Palettes ▶ 2D Graphs ▶ Six Functions menu item as well as other palettes or drag them out from the function Factory Inspector.
- Resizing the function transforms all points simultaneously. A function cannot be rotated because it has to maintain its function properties. For more information on editing a function, including point-wise editing, see Standard Editing.
- Once a function is created, the usual way to modify its data values is through the Table on its inspector editor.
- To program a function see the Programming section.

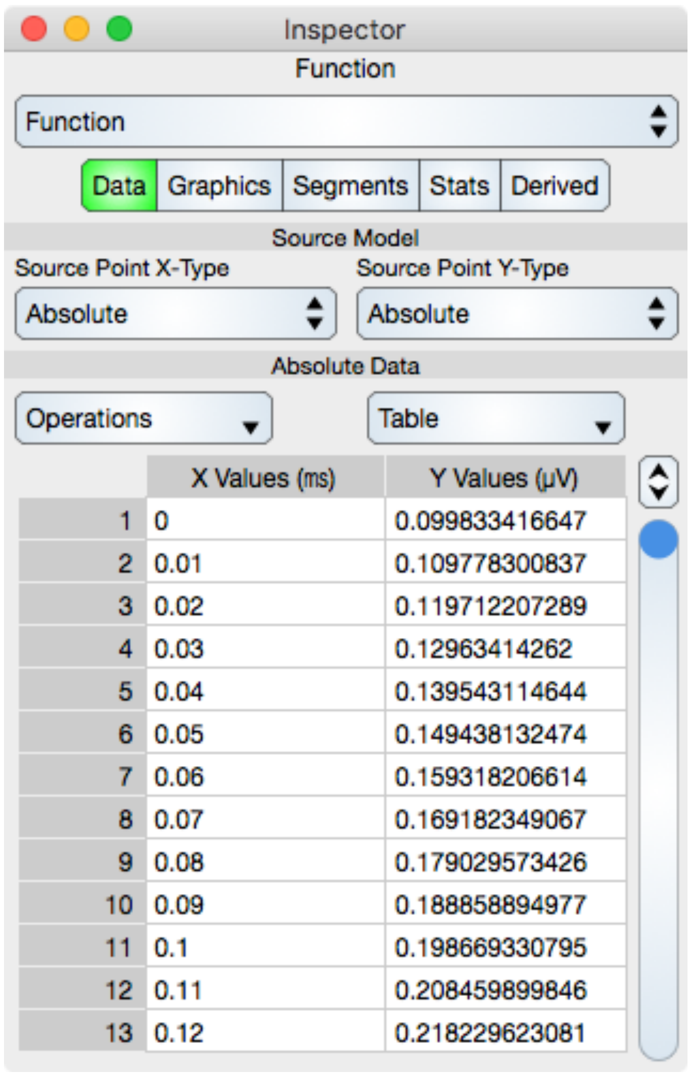
Data Editor

The Data Editor for the Function is shown below.

Source Model

Source Model transforms the original data to a new data set (called the range). Other models (for example, Programming Function) transform the original data directly, but Source Model takes the original data and transforms it. Once a source model is defined then user operations are on the range of the source model transformation.

The main reason to use a Source Model is when the data can not be computed directly. This happens for error bars and cumulative area



Source Point X-Type: Absolute

Source Point Y-Type: Absolute

	X Values (ms)	Y Values (µV)
1	0	0.099833416647
2	0.01	0.109778300837
3	0.02	0.119712207289
4	0.03	0.12963414262
5	0.04	0.139543114644
6	0.05	0.149438132474
7	0.06	0.159318206614
8	0.07	0.169182349067
9	0.08	0.179029573426
10	0.09	0.188858894977
11	0.1	0.198669330795
12	0.11	0.208459899846
13	0.12	0.218229623081

graphs where the data is relative to a previous or next function graphic.

Source Point X-Type : Defines the built in type which is Absolute, Relative To Previous, or Relative to Next. Absolute means no transformation and is the default, Relative To Previous means to interpret the data as values relative to the previous graphic, which must be a Function graphic. Relative To Next means to interpret the data as values relative to the next graphic, which must be a Function graphic. The next or previous graphic is that graphic in the Layer sequence of graphics. When changing the source model you are given the opportunity to convert the original data so that it represents the new Source Model type instead of the old type. For example, if data is in absolute coordinates then converting the data will alter that data to values relative to the related graphic.

Source Point Y-Type : Same as Source Point X-Type except in the Y direction.

Table

Operations : Select this to perform common sequence operations upon the data. The most important sequence operation is Sort X-Ascending because the table data should always be x-ascending.

Generic table controls are described in the Tables section.

The rows represent vertex values. While in Atomic mode, the cell represents a point (x and y value) and while in component mode the cell represents either a x or y value. Hence, in atomic mode there is one column while in component mode there are two columns.

If points are imported into the table or edited and that operation does not produce x-ascending data then the table should be resorted using the Operations control, i.e.: x-ascending is not enforced and needs to be done explicitly.

X-ascending data is a convention that helps with optimization. It is also required to meet the definition of the Function graphic.

Graphics Editor

The Graphics Editor for the Function is shown below.

Common Controls

Controls common to all graphics are described in the Graphics section.

Coordinate : When the function resides on a Multiple Coordinate Graph then the coordinate control is used to define which coordinate it resides on.

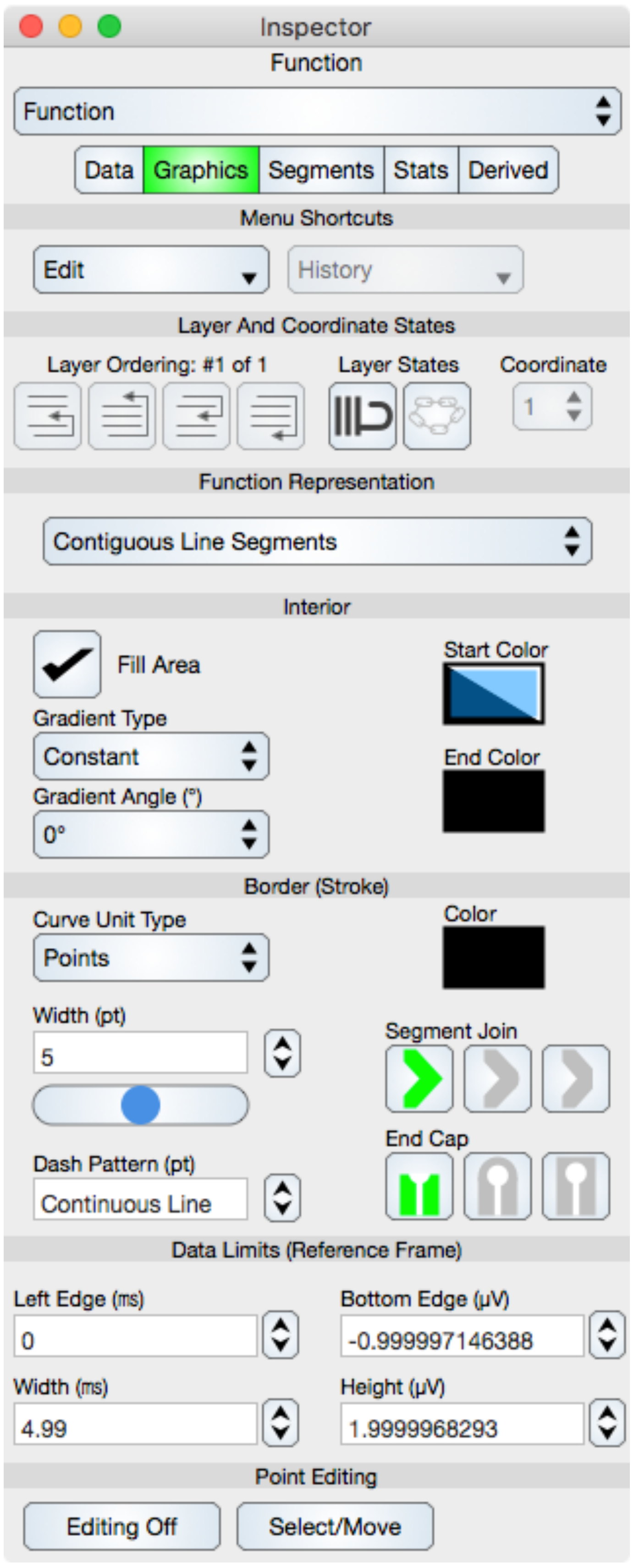
Function Representation

Representation Type : One of the following: (a) Contiguous Line Segments: The normal line graph representing a continuous function, (b) Disconnected Y-Constant Segments: Signifies that the data is discrete and non-interpolated or (c) Connected Y-Constant Segments: A box function representation also used for histograms. In the case of (b) and (c) the Y-Constant Segments are centered about the x-value of the data and the x-length are from the midpoint between adjacent points while the y-value is that of the data for that point.

Point Editing

Editing Off/On : Places the graphic into or out of edit mode. While in edit mode the vertices are shown by indicators and can be adjusted using mouse or touch events. Double-clicking the graphic also toggles this edit mode.

Select/Move or Add/Delete : Select/Move mode permits the vertex editing to select and move those locations while Add/Delete mode will delete a vertex if it is hit or add a vertex if a Function segment is hit. This can also be accomplished using the shift key if available.


Inspector
Function
Function
Data | Graphics | Segments | Stats | Derived
Menu Shortcuts
Edit
History
Layer And Coordinate States
Layer Ordering: #1 of 1
Layer States
Coordinate: 1
Function Representation
Contiguous Line Segments
Interior
Fill Area ✓
Start Color
Gradient Type: Constant
End Color
Gradient Angle (°): 0°
Border (Stroke)
Curve Unit Type: Points
Color
Width (pt): 5
Segment Join
End Cap
Dash Pattern (pt): Continuous Line
Data Limits (Reference Frame)
Left Edge (ms): 0
Bottom Edge (µV): -0.999997146388
Width (ms): 4.99
Height (µV): 1.9999968293
Point Editing
Editing Off
Select/Move

Segments Editor

The Segments Editor for the Function is shown below.

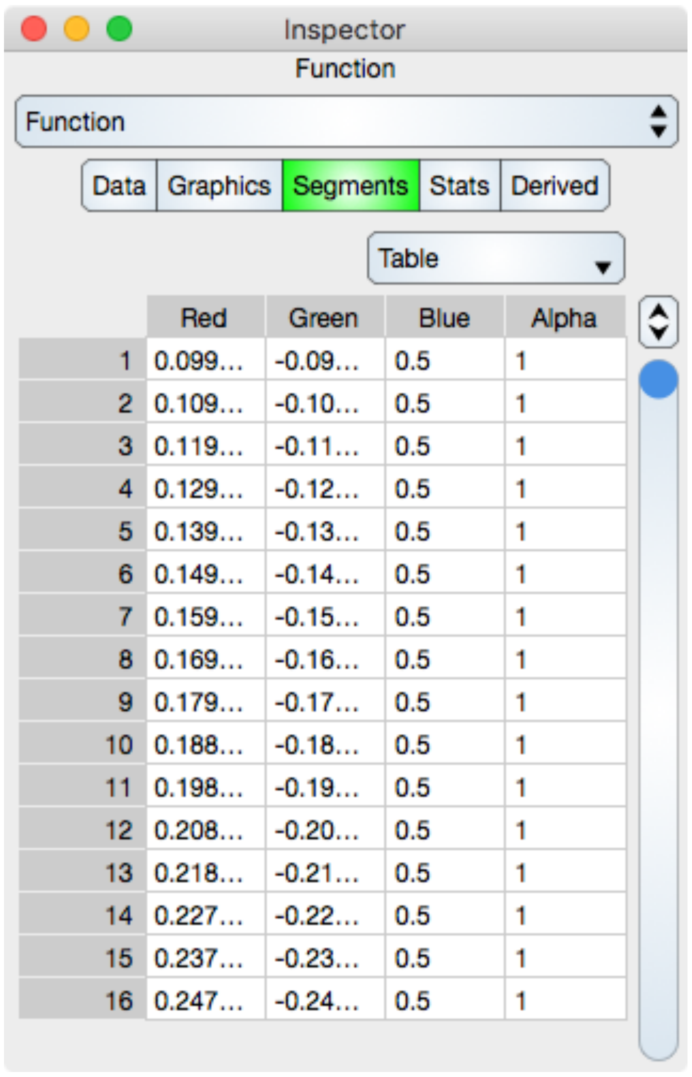
Segments are the line segments between adjacent points. See Sequence Colors for additional information.

Table

Table controls are described in the Tables section.

The rows represent segment color values. While in Atomic mode, the cell represents a rgba value and while in component mode the cell represents either single r, g, b and a. Hence, in atomic mode there is one column while in component mode there are four columns.

Note that if the Data sequence is resorted then the Segments sequence will not be resorted.



	Red	Green	Blue	Alpha
1	0.099...	-0.09...	0.5	1
2	0.109...	-0.10...	0.5	1
3	0.119...	-0.11...	0.5	1
4	0.129...	-0.12...	0.5	1
5	0.139...	-0.13...	0.5	1
6	0.149...	-0.14...	0.5	1
7	0.159...	-0.15...	0.5	1
8	0.169...	-0.16...	0.5	1
9	0.179...	-0.17...	0.5	1
10	0.188...	-0.18...	0.5	1
11	0.198...	-0.19...	0.5	1
12	0.208...	-0.20...	0.5	1
13	0.218...	-0.21...	0.5	1
14	0.227...	-0.22...	0.5	1
15	0.237...	-0.23...	0.5	1
16	0.247...	-0.24...	0.5	1

Stats Editor

The Stats Editor provides basic statistics for the Function and is shown below.

Basics

Number Of Points : Shows the number of points in the Function.

X-Minimum : x-minimum value of all the data points. Since the data points should be x-ascending, the x-minimum is the x-value of the first data point in the sequence.

Y-Minimum : y-minimum value of all the data points.

X-Maximum : x-maximum value of all the data points. Since the data points should be x-ascending, the x-maximum is the x-value of the last data point in the sequence.

Y-Maximum : y-maximum value of all the data points.

Basic Distribution

The basic distribution is shown separately for the x-dimension and the y-dimension. For a function, the y-distribution is the important distribution.

Mean : Shows the mean of the data.

Median : Shows the median of the data.

Standard Deviation : Shows the standard deviation of the data.

Range : Shows the range of the data (maximum - minimum).

Linear Regression Constants

Note that linear regression may also be referred to as line fit.

Slope : Shows the slope of the linear regression.

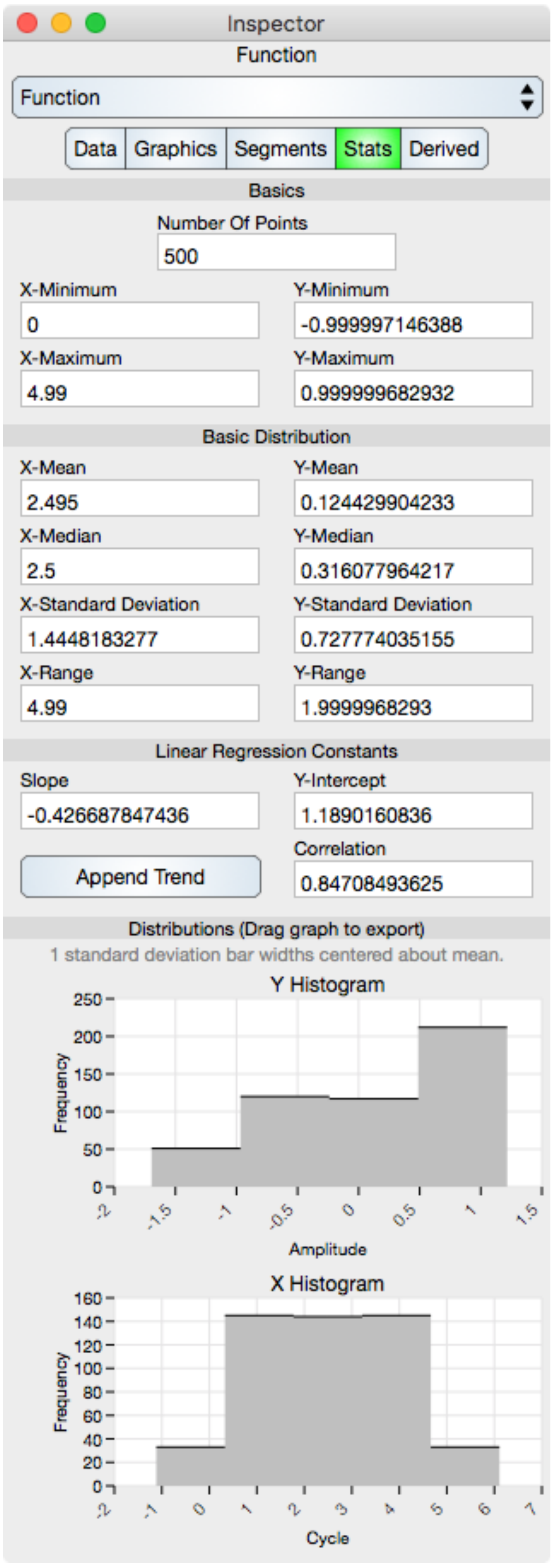
Y-Intercept : Shows the y-intercept of the linear regression.

Correlation : Shows the correlation constant of the linear regression.

Append Trend : Places a new trend line over the function. The trend is a least squares fit and each x-value of the function is also on the trend. Thus if the function is on a non-linear graph then the trend will map as well and not look like a linear fit on page view coordinates.

Distributions

Y Histogram : Shows the y-histogram of the data. This graph can be dragged onto the Graphic View.



Basics

Number Of Points: 500

X-Minimum: 0
Y-Minimum: -0.999997146388
X-Maximum: 4.99
Y-Maximum: 0.999999682932

Basic Distribution

X-Mean: 2.495
Y-Mean: 0.124429904233
X-Median: 2.5
Y-Median: 0.316077964217
X-Standard Deviation: 1.4448183277
Y-Standard Deviation: 0.727774035155
X-Range: 4.99
Y-Range: 1.9999968293

Linear Regression Constants

Slope: -0.426687847436
Y-Intercept: 1.1890160836
Correlation: 0.84708493625

Distributions (Drag graph to export)

1 standard deviation bar widths centered about mean.

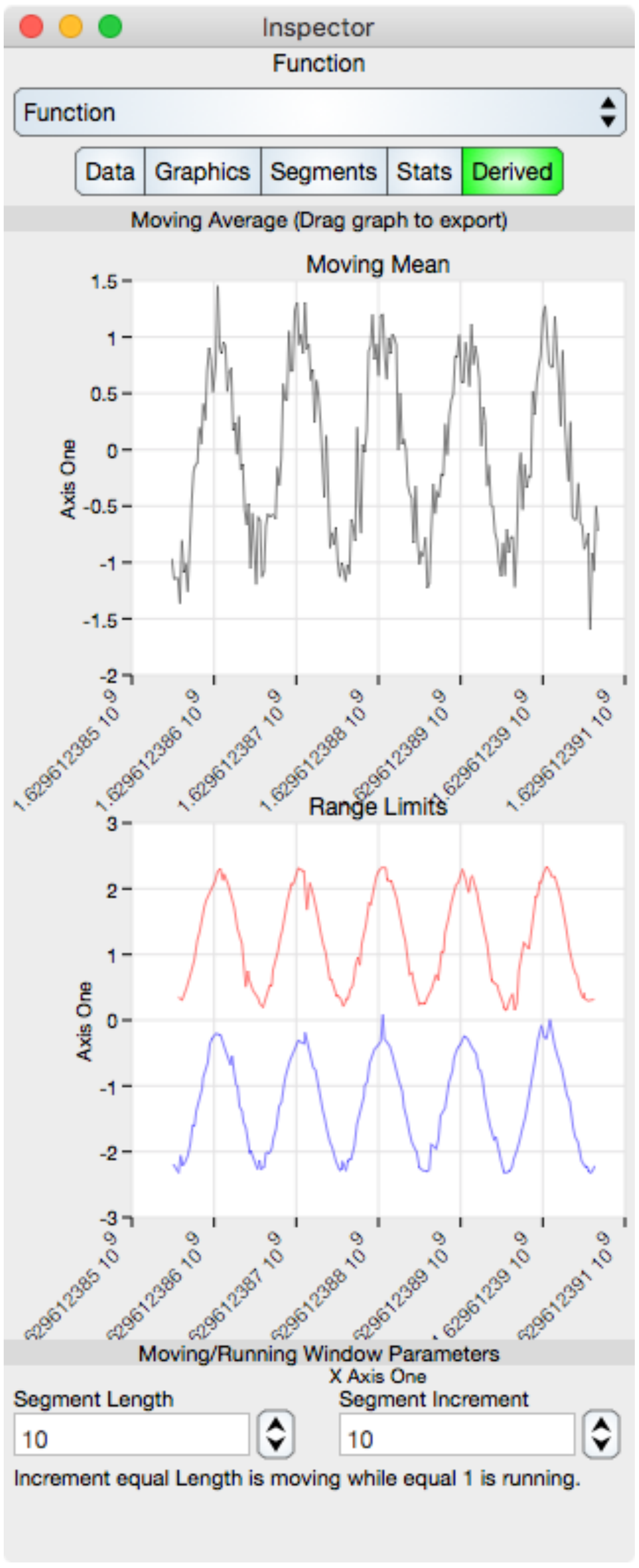
X Histogram : Shows the x-histogram of the data. This graph can be dragged onto the Graphic View.

Derived Editor

The Derived Editor for the Function is shown below.

Derived output can be anything, but in this current implementation it is limited to moving average and range limit graphs. The Moving Average smooths out the function while the Range Limits show the minimum and maximum y-value within a bin of the moving average.

Consider the following noisy function:

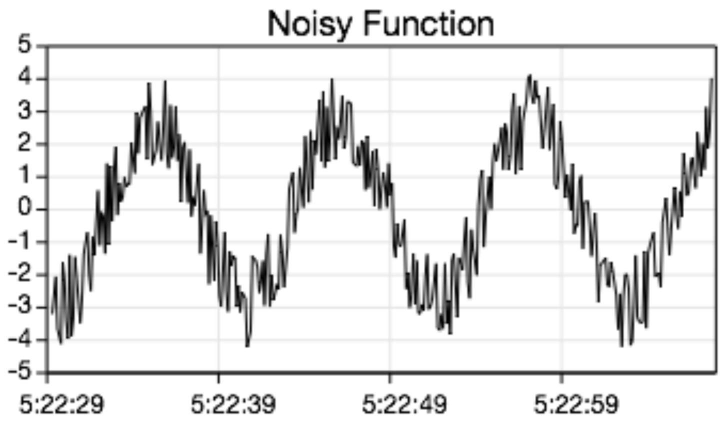


Moving/Running Window Parameters

Segment Length: 10

Segment Increment: 10

Increment equal Length is moving while equal 1 is running.



Running Mean Graph : Shows the running mean of the data. When the segment increment is not 1 then this shows the Moving Average of the data. If the segment length and increment is one then the Running Mean Graph is the Identity.

Range Limits : Shows the minimum and maximum values within a bin defined by the Segment Length. The minimum is the blue curve and the maximum is the red curve.

The graphs can be dragged to the main document graphic view and then the derived curves can be copied and pasted to the graph's data layer so that the derived curves can overlay the original data. While the Derived values on the inspector are recomputed when the data changes, any pasted result is not.

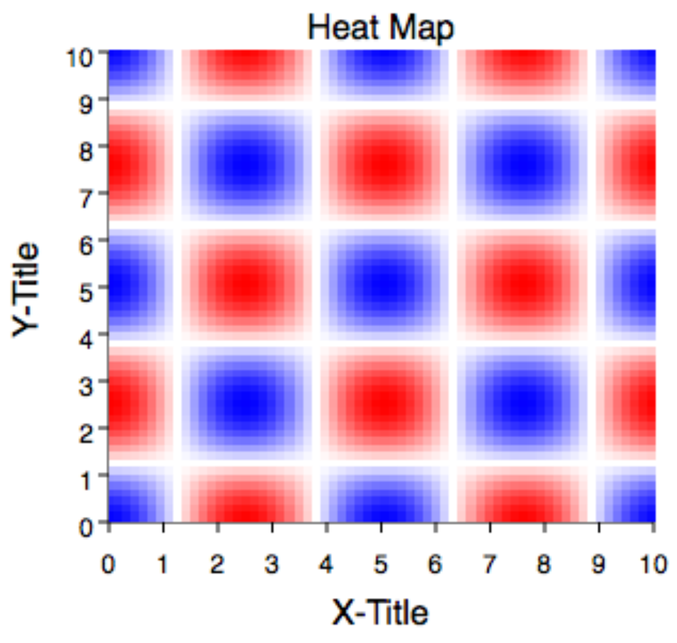
As a side note, in this example notice how the derived graphs are on a rectilinear graph while the Noisy Function curve is on a Gregorian graph. When the curves are pasted onto the target coordinate then its x-values (representing seconds from 1970) are remapped to the Gregorian x-unit.

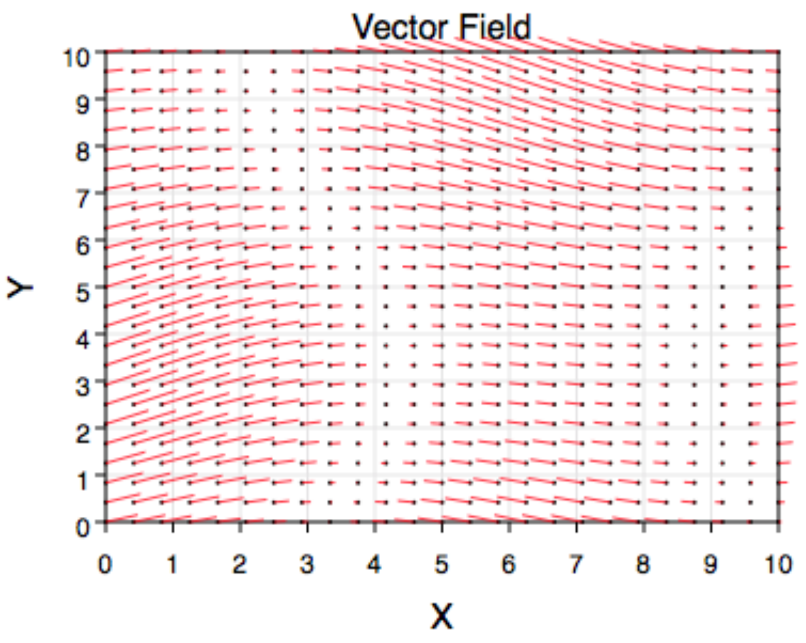
Segment Length : Defines the number of points that are used to compute the mean of the derived value. If 1 then the mean is the data itself. This value should be set great enough to span several values of noise while small enough to be within about 5 percent of the data variation.

Segment Increment : Defines the increment of the bin window. When 1 then the computation is a running mean and when equal to the segment length then the average is considered a moving mean. The main reason to set this greater than 1 is to reduce the number of points in the derived computation.

Graph IDE ▶ Data Graphics ▶ Point Map
a.k.a.: Heat Map, Image Map and Vector Field

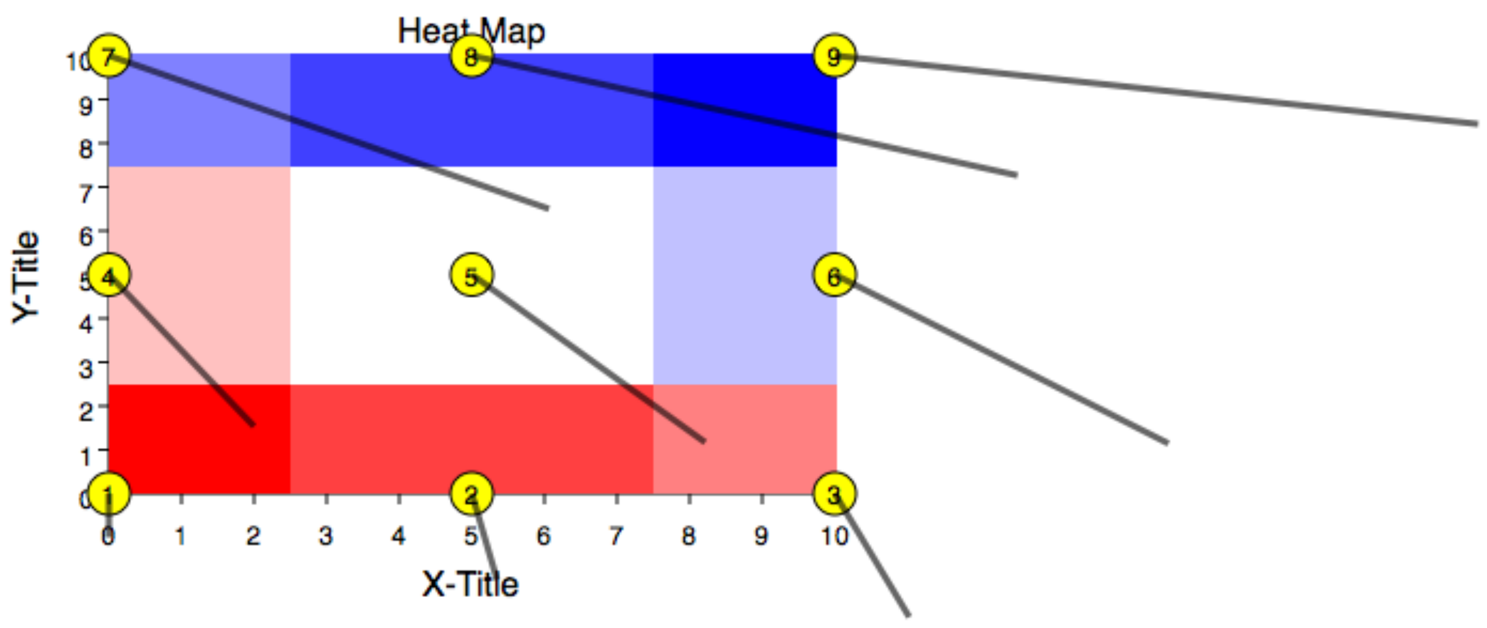
A Point Map is a sequence of x-contiguous values arranged as a matrix that is mapped to a regular grid of cells. The values can be either an amplitude or an amplitude and direction pair. The figures below shows two example point maps.





The vector field is shown with the point tag marker, a small dot, on. The direction of the vector is away from the dot. The dot is somewhat superfluous as the data is on a regular grid and the origin of the vector is easily determined visually. Sometimes a vector field is shown with an arrow at the end of the vector but in this implementation that is not the case as the dot provides a better visual of direction. For dense data it might be advisable to turn the dot off.

The figure below shows a low-sampled point map graph of nine values on a 3x3 grid. The values are amplitudes {1 2 3 4 5 6 7 8 9} and direction (angles) {-90 -80 -70 -60 -50 -40 -30 -20 -10}. It is low enough so you can see the edge cells are half cells, the corner cells are quarter cells, the interior cells (one in this case) are full cells, the dot is at the cell center and the vector origin is at the center of the cell. The dot sequence index shows the x-contiguous nature of the data representation. The edge and corner truncated cell areas are needed in order to make the cell pattern line up exactly with the x and y minimum and maximum entered into the inspector editor. Notice how the vector length corresponds to the color mapping.



Some standard operations are itemized below.

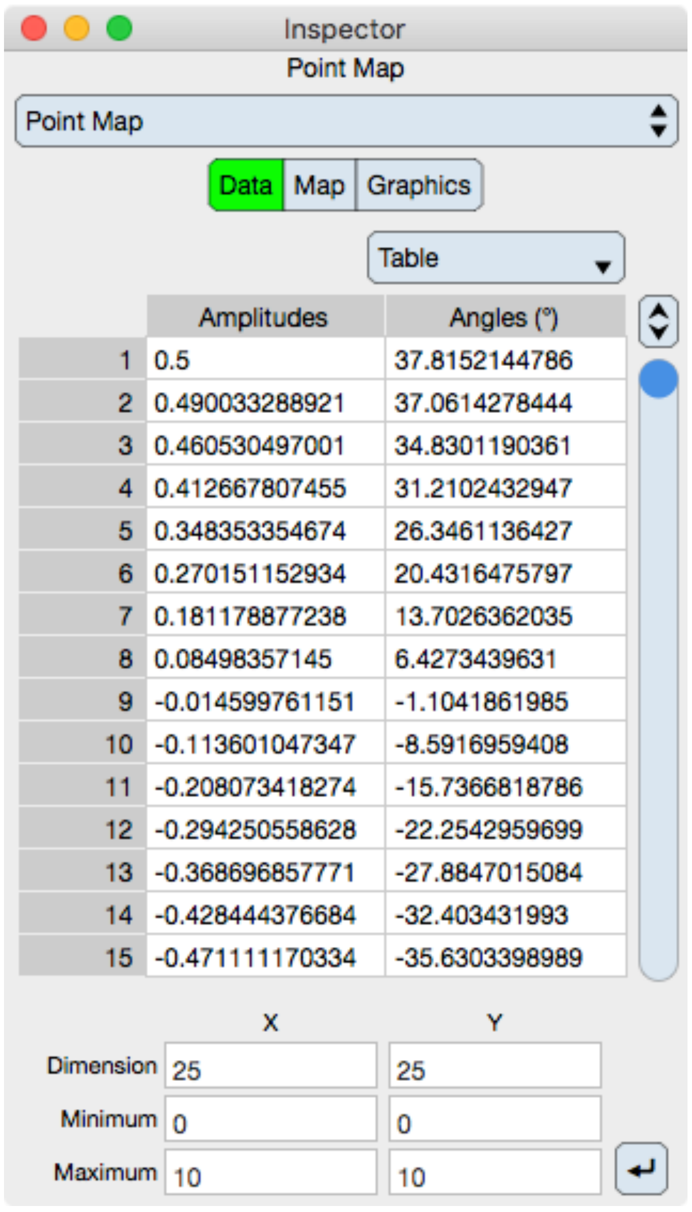
- To create a Point Map bring forward the Graphic Selector, click the Point Map factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of a predefined Point Map graphic.
- Resizing and rotating the Point Map transforms all points simultaneously. For more information see Standard Editing. The Point Map conforms to the Color Map standards where the color map gradient is usually considered a orthogonal z-dimension geometrically.
- Once a point map is created, the usual way to modify its data values is through the Table on the inspector editor.
- Some drag and drop examples are available through the menu item Palettes ▶ Programs ▶ 2D Graphs. Those examples are animated so the temporal dimension adds a fourth and fifth dimension to the point map graphic. Those dimensions are: {x, y, amplitude, angle, time}. Prototypes can also be drag from the Point Map Factory Inspector.
- The settings for a point map are for a heat map fill. To set for a vector field plot, in the Graphics Inspector Editor turn off the fill, turn on the stroke and enter angles into the table. Notice how this graphic can show the heat map and vector field representation at the same time if desired.

- To program a point map see the Programming section.

The point map conforms to non-linear graph features so it can be embedded onto any of the nonlinear graphs. For polar graphs the x-values are in degrees, usually from 0 to 360.

Data Editor

The Data Editor for the Point Map is shown below.



	Amplitudes	Angles (°)
1	0.5	37.8152144786
2	0.490033288921	37.0614278444
3	0.460530497001	34.8301190361
4	0.412667807455	31.2102432947
5	0.348353354674	26.3461136427
6	0.270151152934	20.4316475797
7	0.181178877238	13.7026362035
8	0.08498357145	6.4273439631
9	-0.014599761151	-1.1041861985
10	-0.113601047347	-8.5916959408
11	-0.208073418274	-15.7366818786
12	-0.294250558628	-22.2542959699
13	-0.368696857771	-27.8847015084
14	-0.428444376684	-32.403431993
15	-0.471111170334	-35.6303398989

	X	Y
Dimension	25	25
Minimum	0	0
Maximum	10	10

Table

Table controls are described in the Tables section.

The rows represent amplitude and angle values at a particular cell. When in atomic mode, each cell shows the amplitude followed by an optional angle value. When in component mode, the first column shows the amplitude values while the second column shows the angle values. In each case, angle values are optional and need not display if not entered. The angle is always input in units of degrees with origin y = 0, x > 0 and counterclockwise positive value orientation.

Angles are optional. If they are not present then the graphic represents a heat map. If they are present then the graphic represents a vector field map. The vector field ray origins are the center of the cells.

Grid Values

The data table does not specify grid parameters, it only specifies amplitude and angle values. That is because a point fill map is on a regular grid. A regular grid is defined by the six values described below.

Dimension : The number of cells in the x and y dimension, aka: The dimensions of the matrix of values. The product of x and y dimensions should equal the number of rows in the table.

Minimum : The x and y minimum of the grid.

Maximum : The x and y maximum of the grid.

Notice that cell center points, which is also considered where the data is on the grid, are coincidental with the grid minimum and maximum. As a result, the cells along the boundary are actually half cells and at the corners quarter cells. This is a very slight but important distinction.

Apply : Once the grid values have been entered then click the Apply button to redefine the grid.

Map Editor

The Map Editor for the Point Map is shown below.

Data Limits And Clipping

The color map spans the data unless it is clipped. Clipping is useful for two reasons: (1) clipping to an interval outside the data values extremum rounds the map to designated values and (2) clipping to an interval inside the data values extremum clips the map itself which is useful to see thresholds of the data.

Data Minimum : Shows the minimum amplitude of the data.

Data Maximum : Shows the maximum amplitude of the data.

Clipped Data Minimum : Defines the minimum value to clip the color map at.

Clipped Data Maximum : Defines the maximum value to clip the color map at.

Does Clip : Clips the color map to the clipping values.

Amplitude And Vector Representation

Amplitude Map : Select this to turn on the color map.

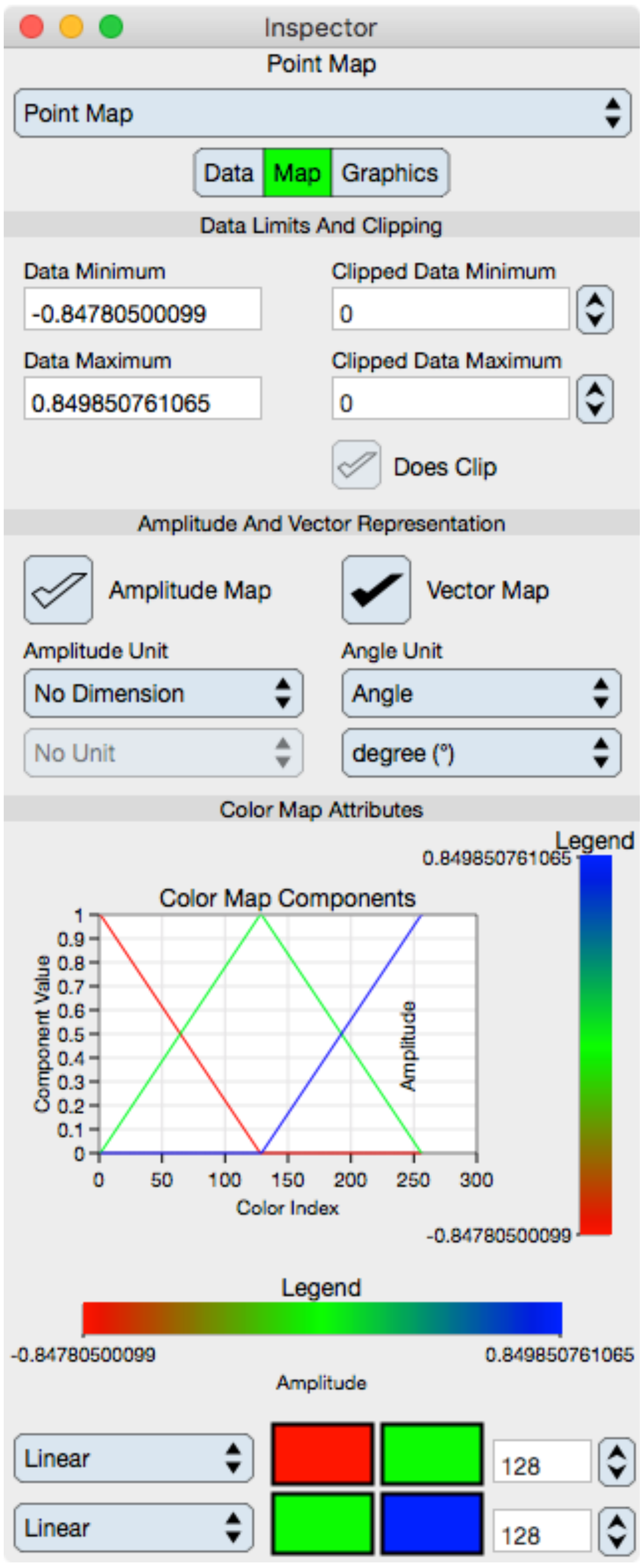
Vector Map : Select this to turn on the vector field.

Amplitude Units : Sets the units of the amplitude of the point map. Note that this unit is orthogonal to the graph units. See Unit Selector.

Angle Units : Sets the units of the angles of the vector field. This unit should typically be in units of degrees of an angle since that is the data input unit of the angles but can be any unit as it is reinterpreted by the user. See Unit Selector.

Color Map Attributes

The color map is further described in the Color Map section.



Graph : Shows a graph of the color map. This graph can be drag and dropped to the graphic view although the legend maps are probably more appropriate to describe the color map.

Legend : Shows a fill representation of the color map in both vertical and horizontal orientation. The independent variable of the graph (x-axis for horizontal and y-axis for vertical) is that of the point map color interpretation. These legends can be dragged to the graphic view as needed. Since the legends are also Point Map graphics they can be independently altered once dragged to the graphic view in accordance with this section of the manual.

Note that if the legend is dropped then its graph independent variable extremum should probably be reset to a rounded value.

Function : Select this to set the color map function. Typically it is set to linear.

Start Color : Defines the start Color of the color map.

End Color : Defines the end Color of the color map.

Number : Defines the number of samples in the color map.

Graphics Editor

The Graphics Editor for the Point Map is shown below.

Common Controls

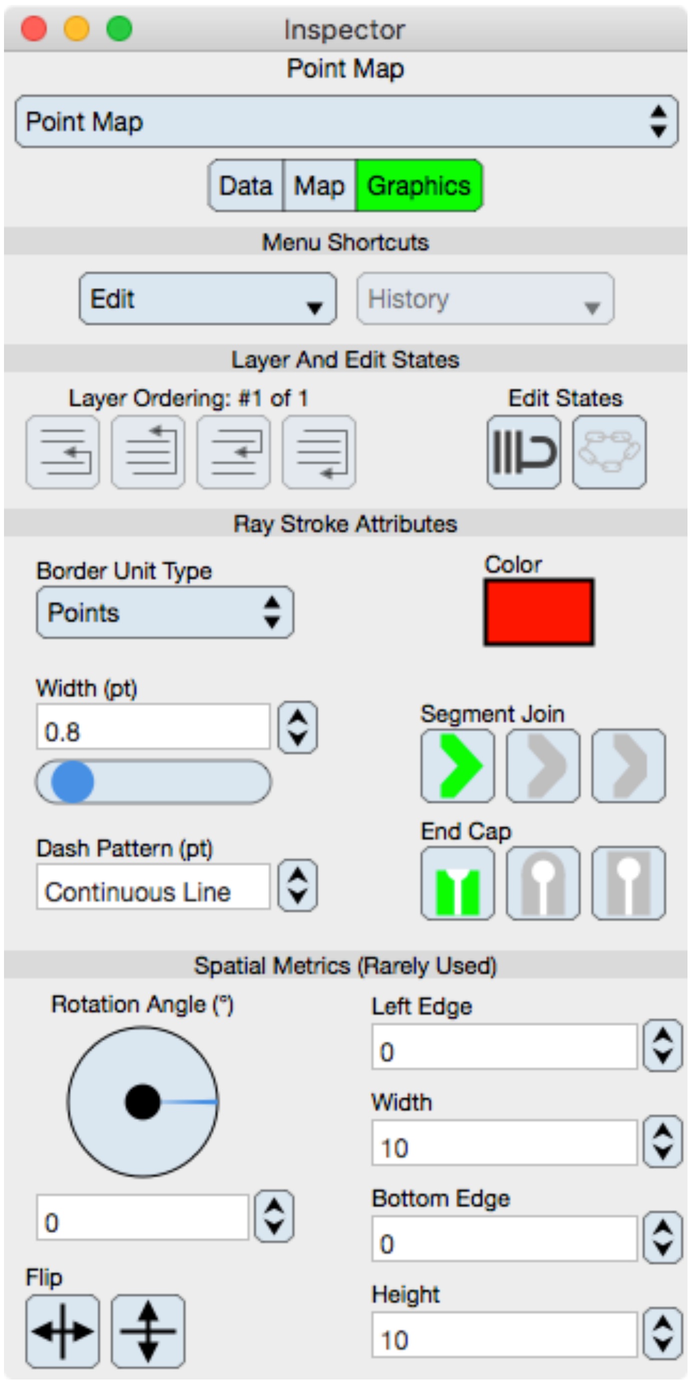
Controls common to all graphics are described in the Graphics section. Notice that the point fill can affine transform in the usual way. That transform is graphical and will not effect the data and as a result the data and graphic will be out of synch.

Point Map Specific

Stroke : The stroke relates to a vector graph rays.

Spatial Metrics

These spatial metric controls are described in the Graphics section. However, be aware that if they are used then the graphical-spatial coordinates will be out of synch with the data grid whose values are set in the Data editor. Setting grid spatial values (minimum and maximum) will update the graphic spatial values; however the converse is not true.


Inspector
Point Map
Point Map
Data | Map | Graphics

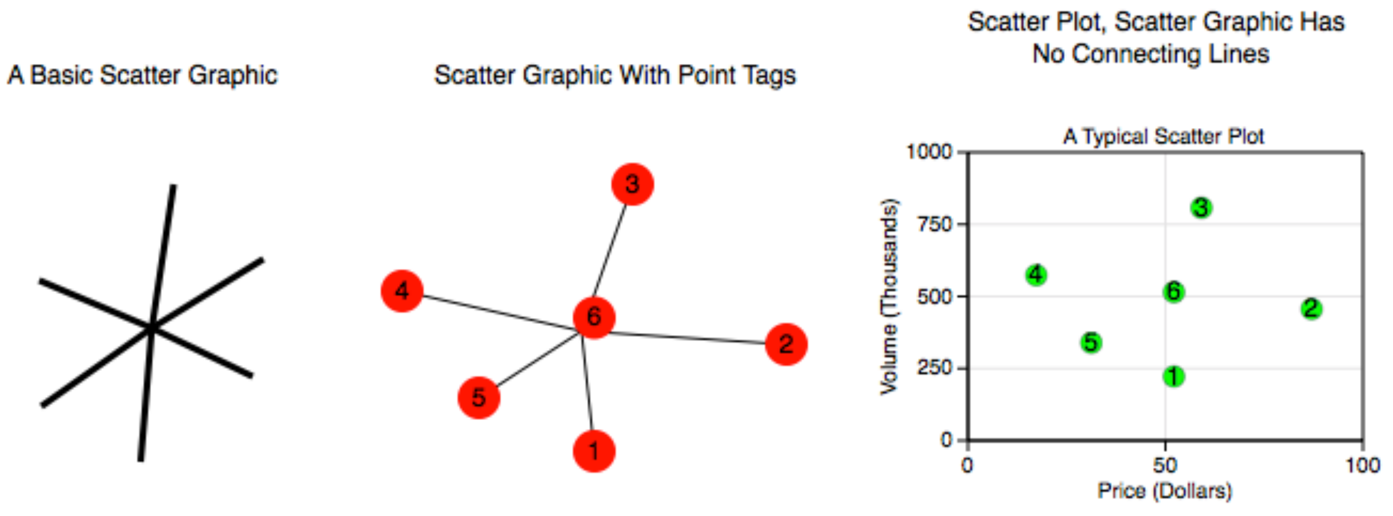
Menu Shortcuts
Edit
History

Layer And Edit States
Layer Ordering: #1 of 1
Edit States

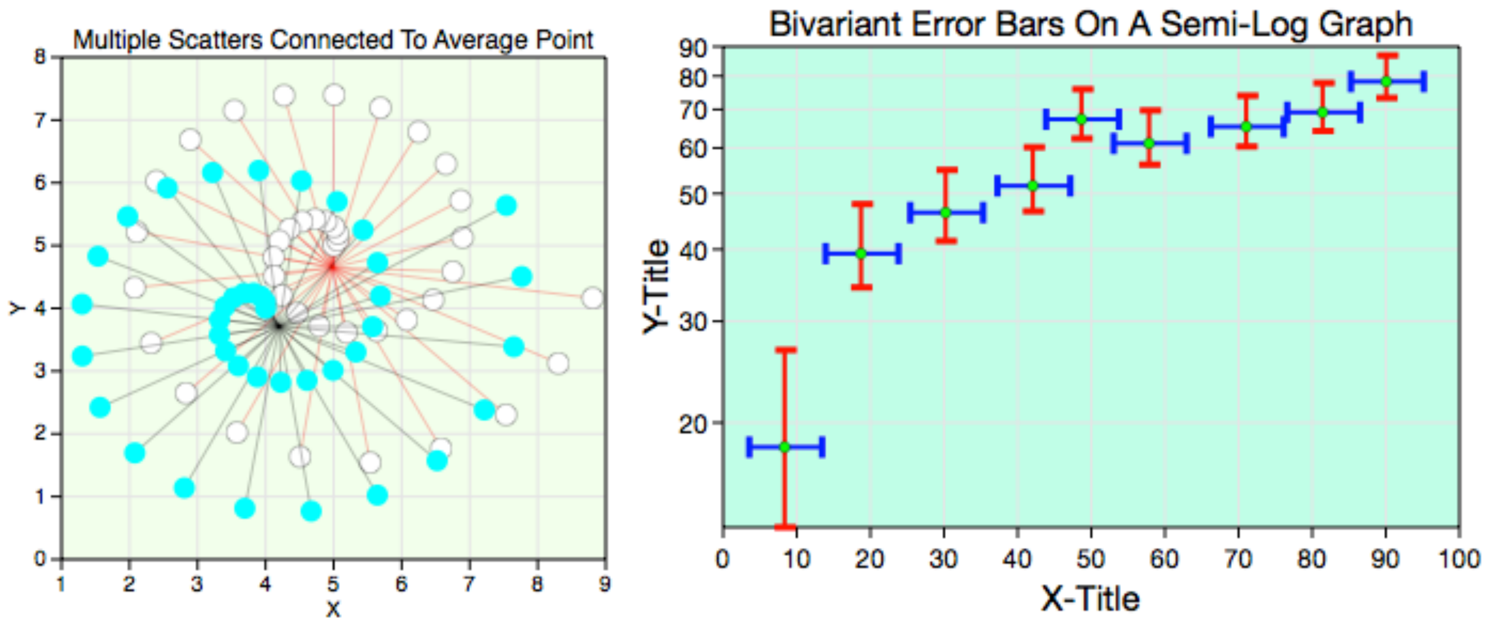
Ray Stroke Attributes
Border Unit Type: Points
Color
Width (pt): 0.8
Segment Join
Dash Pattern (pt): Continuous Line
End Cap

Spatial Metrics (Rarely Used)
Rotation Angle (°): 0
Left Edge: 0
Width: 10
Bottom Edge: 0
Height: 10
Flip

A scatter is a sequence of points. Those points may be connected to a common point, usually the mean of the points. A scatter is very similar to a Polygon (see Polygon) except for the way that the points are connected. The figures below show examples of the use of scatters.



Scatter graphics support the concept of a Network, are simply an element of a Layer in either an overlay or graph and transform according to the properties of a coordinate per the Non-Linear Graphs rules. As such, scatter graph and indeed any data graphic support attributes that can make them far superior to a simple scatter graphic canonical form. The figure below shows some examples of that.



Some standard operations are itemized below.

- To create a scatter bring forward the Graphic Selector, click the scatter factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of a predefined scatter graphic.
- You can also create scatters from the standard Palettes menu item or drag them out from the scatter Factory Inspector.
- Resizing the scatter transforms all points simultaneously. For more information on editing a scatter, including point-wise editing, see Standard Editing.
- The normal way to modify the points of a scatter is through the scatter's Inspector Editor's table.
- To program a scatter see the Programming section.

Data Editor

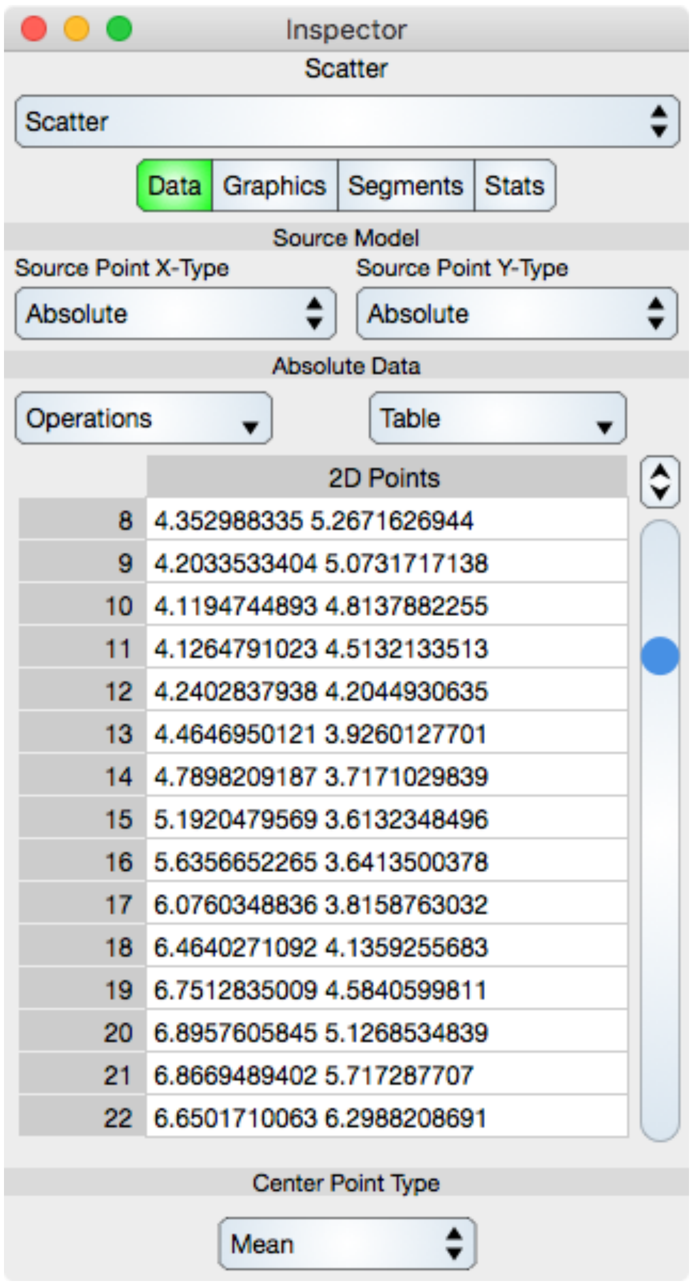
The Data Editor for the Scatter Graphic is shown below.

Source Model

A description of the Source Model is the same as for the Function graphic.

Note that for the Bivariant Error Bar graph shown above there are six scatter graphics. Two represent the data and the other four represent the error data. The four have their Source Model set to be relative to the two data scatter graphics. It is a bit tedious to setup and modify as the data is contained in the representation, however if the data resides in a Spreadsheet then data adjustments become much more direct.

Table



Scatter: Scatter

Source Model

Source Point X-Type: Absolute

Source Point Y-Type: Absolute

Absolute Data

	2D Points
8	4.352988335 5.2671626944
9	4.2033533404 5.0731717138
10	4.1194744893 4.8137882255
11	4.1264791023 4.5132133513
12	4.2402837938 4.2044930635
13	4.4646950121 3.9260127701
14	4.7898209187 3.7171029839
15	5.1920479569 3.6132348496
16	5.6356652265 3.6413500378
17	6.0760348836 3.8158763032
18	6.4640271092 4.1359255683
19	6.7512835009 4.5840599811
20	6.8957605845 5.1268534839
21	6.8669489402 5.717287707
22	6.6501710063 6.2988208691

Center Point Type: Mean

Operations : Select this to perform common sequence operations upon the data. The most important sequence operation is Switch X and Y because that transposes the data and hence the dimensions. Contrast this to the Function graphic which assumes that x is alway ascending and always the independent variable of the coordinate system (graph).

Generic table controls are described in the Tables section.

The rows represent vertex values. While in Atomic mode, the cell represents a point (x and y value) and while in component mode the cell represents either a x or y value. Hence, in atomic mode there is one column while in component mode there are two columns.

Center Point Type

Center Point Type : Scatter graphs are typically unconnected representations. However, data can be connected in sequence order, to the average point, median point or to the zero point (None, Average, Median and Zero). If connected in sequence order then the scatter graph is often referred to as a trajectory graph. Since trajectories are often smooth that designation is reserved for the Trajectory graphic which is points connected by Cubic Bezier spline sections.

If None then the data points are connected in sequence by line segments. Turning off the stroke in the graphic editor turns off the connected segments and results in a classic scatter representation.

Graphics Editor

The Graphics Editor for the Scatter graphic is shown below.

Common Controls

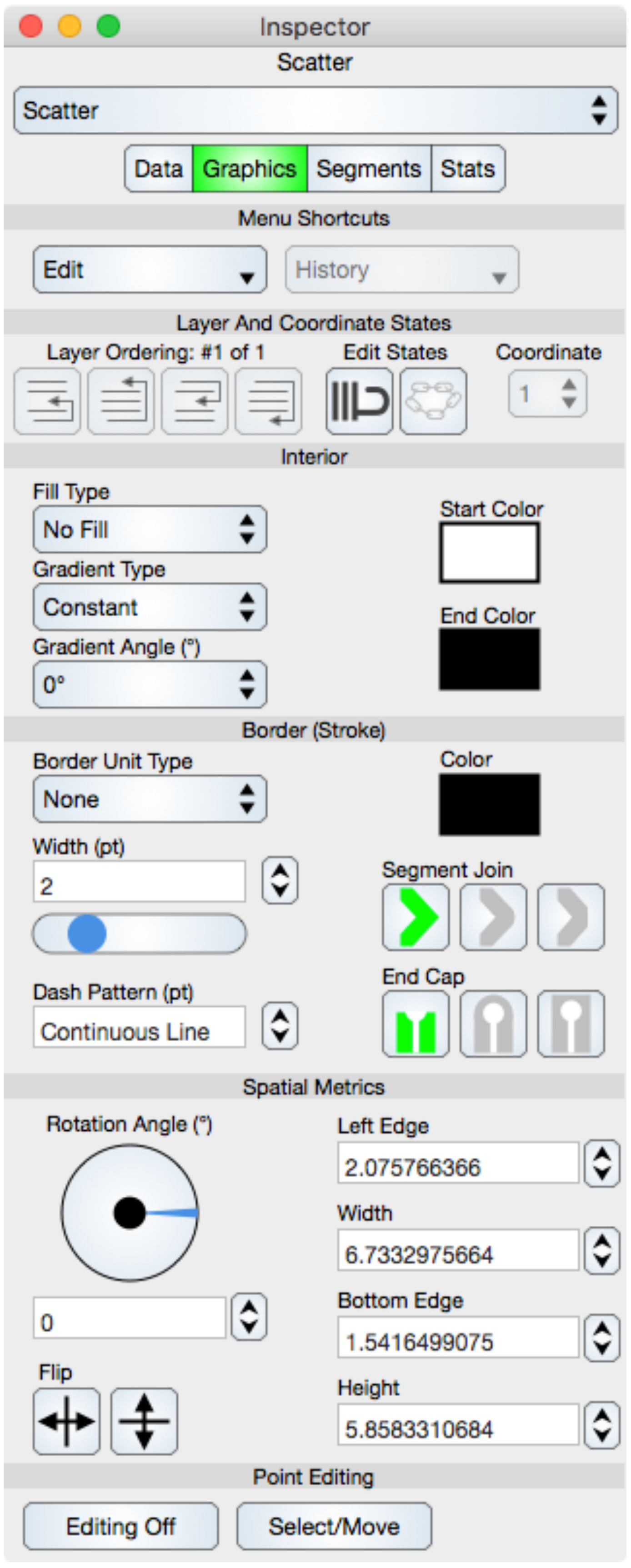
Controls common to all graphics are described in the Graphics section.

Coordinate : When the scatter resides on a Multiple Coordinate Graph then the coordinate control is used to define which coordinate it resides on.

Point Editing

Editing Off/On : Places the graphic into or out of edit mode. While in edit mode the vertices are shown by indicators and can be adjusted using mouse or touch events. Double-clicking the graphic also toggles this edit mode.

Select/Move or Add/Delete : Select/Move mode permits the vertex editing to select and move those locations while Add/Delete mode will delete a vertex if it is hit or add a vertex if a Scatter segment is hit. This can also be accomplished using the shift key if available.


Inspector

Scatter

Scatter

Data | Graphics | Segments | Stats

Menu Shortcuts

Edit | History

Layer And Coordinate States

Layer Ordering: #1 of 1 | Edit States | Coordinate: 1

Interior

Fill Type: No Fill

Gradient Type: Constant

Gradient Angle (°): 0°

Start Color

End Color

Border (Stroke)

Border Unit Type: None

Color

Width (pt): 2

Segment Join

End Cap

Dash Pattern (pt): Continuous Line

Spatial Metrics

Rotation Angle (°): 0

Left Edge: 2.075766366

Width: 6.7332975664

Bottom Edge: 1.5416499075

Height: 5.8583310684

Flip

Point Editing

Editing Off | Select/Move

Segments Editor

The Segments Editor for the Scatter graphic is shown below.

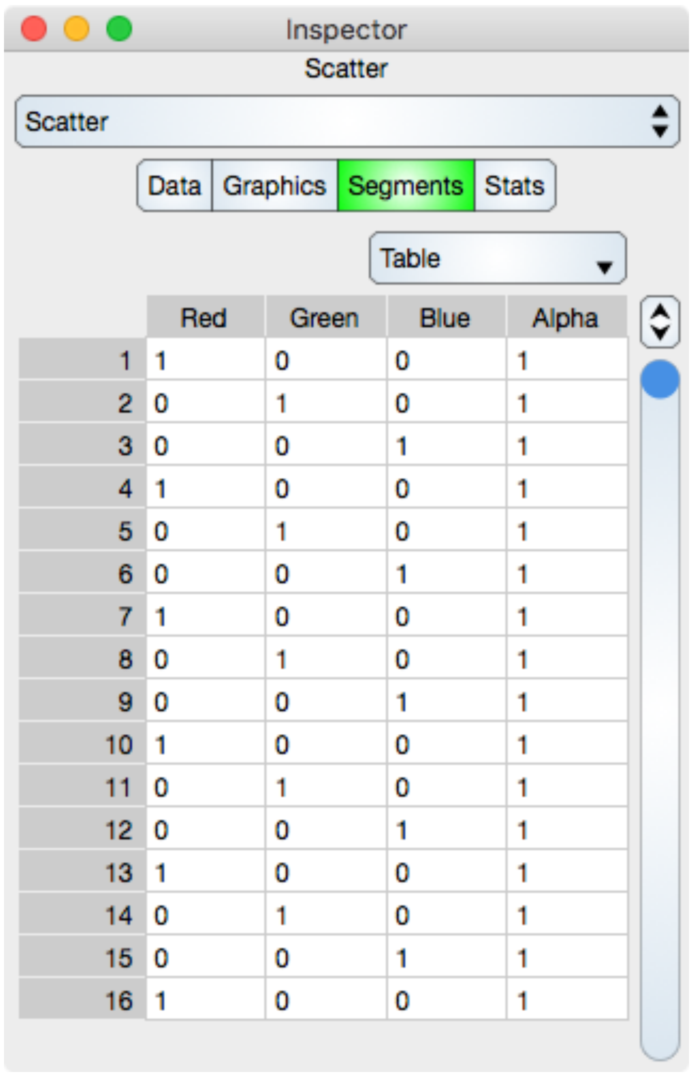
Segments are the line segments between adjacent points. See Sequence Colors for additional information.

Table

Table controls are described in the Tables section.

The rows represent a segment color values. While in Atomic mode, the cell represents a rgba value and while in component mode the cell represents either single r, g, b and a. Hence, in atomic mode there is one column while in component mode there are four columns.

Note that if the Data sequence is resorted then the Segments sequence will not be resorted.



	Red	Green	Blue	Alpha
1	1	0	0	1
2	0	1	0	1
3	0	0	1	1
4	1	0	0	1
5	0	1	0	1
6	0	0	1	1
7	1	0	0	1
8	0	1	0	1
9	0	0	1	1
10	1	0	0	1
11	0	1	0	1
12	0	0	1	1
13	1	0	0	1
14	0	1	0	1
15	0	0	1	1
16	1	0	0	1

Stats Editor

The Stats Editor shows basic statistics for the Scatter graphic and is shown below.

Basics

Number Of Points : Shows the number of points in the Scatter.

X-Minimum : x-minimum value of all the data points.

Y-Minimum : y-minimum value of all the data points.

X-Maximum : x-maximum value of all the data points.

Y-Maximum : y-maximum value of all the data points.

Basic Distribution

The basic distribution is shown separately for the x-dimension and the y-dimension.

Mean : Shows the mean of the data.

Median : Shows the median of the data.

Standard Deviation : Shows the standard deviation of the data.

Range : Shows the range of the data (maximum - minimum).

Linear Regression Constants

Note that linear regression may also be referred to as line fit.

Slope : Shows the slope of the linear regression.

Y-Intercept : Shows the y-intercept of the linear regression.

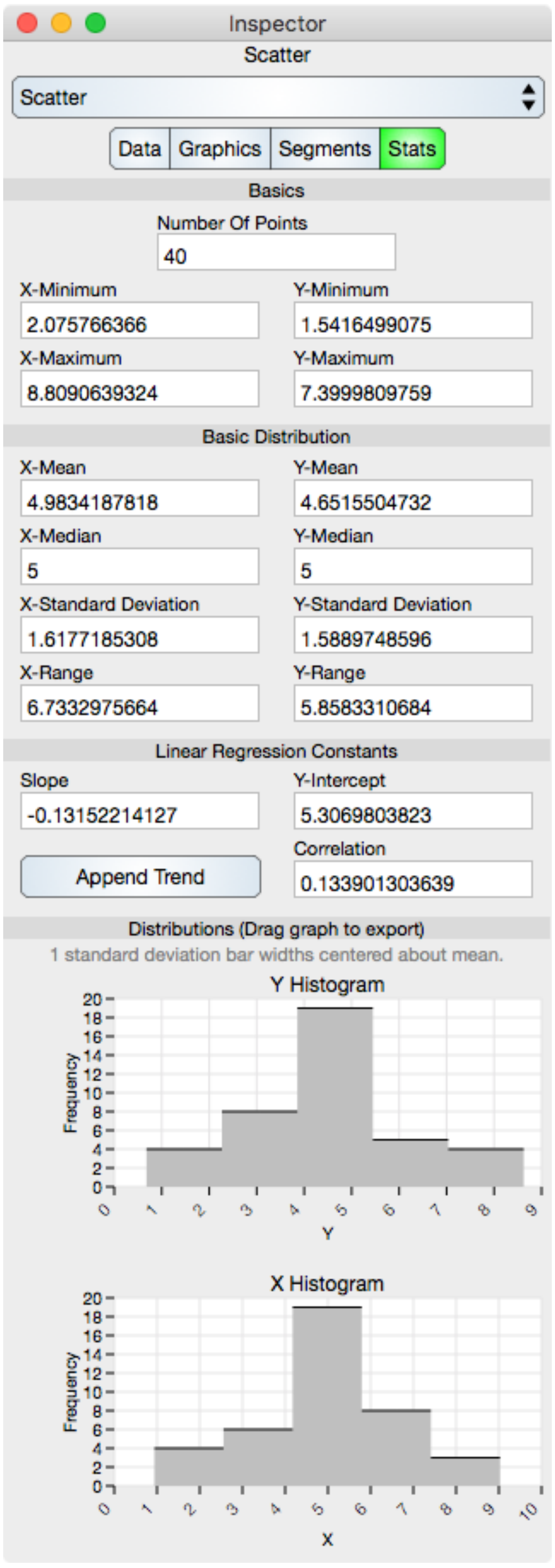
Correlation : Shows the correlation constant of the linear regression.

Append Trend : Places a new trend line over the scatter. The trend is a least squares fit and each data value of the scatter is also on the trend. Thus if the scatter is on a non-linear graph then the trend will map as well and not look like a linear fit on page view coordinates.

Distributions

Y Histogram : Shows the y-histogram of the data. This graph can be dragged onto the Graphic View.

X Histogram : Shows the x-histogram of the data. This graph can be dragged onto the Graphic View.



Basics

Number Of Points: 40

X-Minimum: 2.075766366
Y-Minimum: 1.5416499075
X-Maximum: 8.8090639324
Y-Maximum: 7.3999809759

Basic Distribution

X-Mean: 4.9834187818
Y-Mean: 4.6515504732
X-Median: 5
Y-Median: 5
X-Standard Deviation: 1.6177185308
Y-Standard Deviation: 1.5889748596
X-Range: 6.7332975664
Y-Range: 5.8583310684

Linear Regression Constants

Slope: -0.13152214127
Y-Intercept: 5.3069803823
Correlation: 0.133901303639

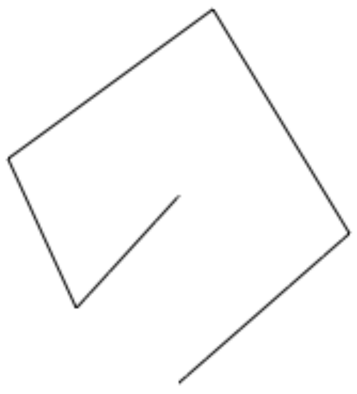
Distributions (Drag graph to export)

1 standard deviation bar widths centered about mean.

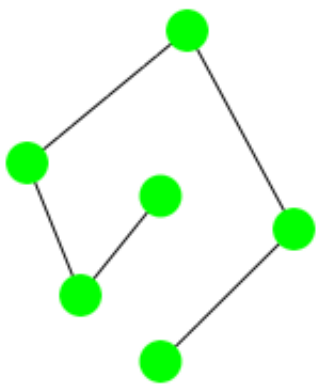
Graph IDE ► Data Graphics ► Trajectory

A trajectory is a sequence of points connected by curves. The curves are Cubic Bezier sections (see Cubic Bezier). The figures below show examples of trajectories.

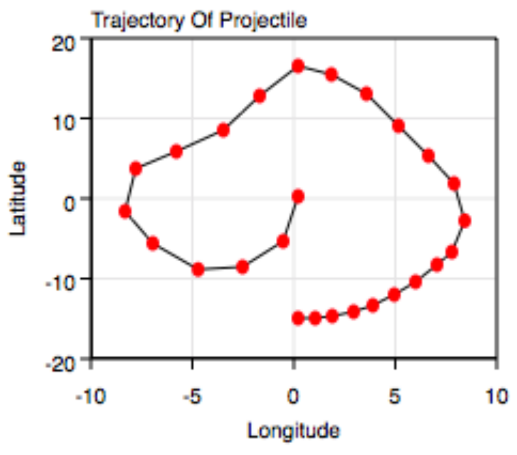
Simple Trajectory Graphic

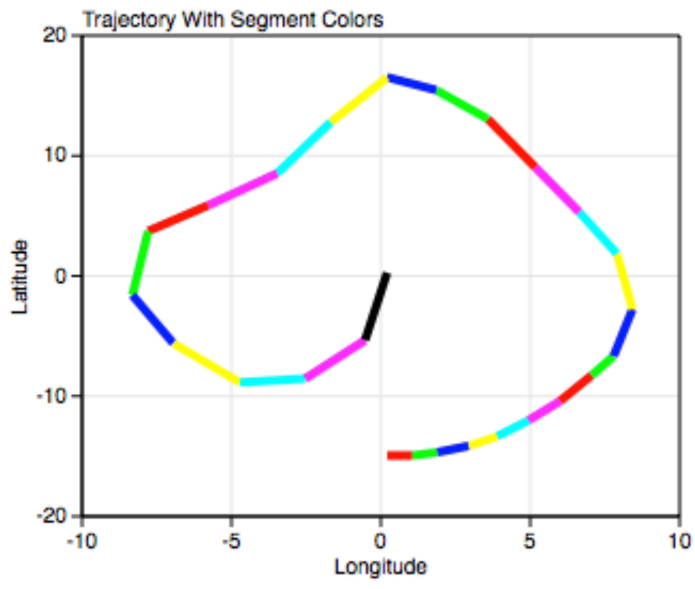


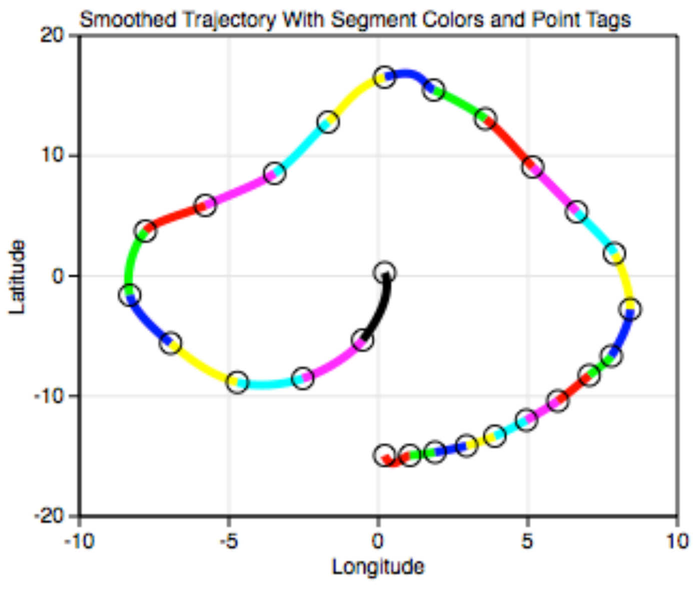
Trajectory Graphic With Point Tags



Trajectory On A Graph







Some standard operations are itemized below.

- To create a trajectory bring forward the Graphic Selector, click the trajectory factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of a predefined trajectory graphic.
- You can also create trajectories from the standard Palettes menu item or drag them out from the trajectory Factory Inspector.
- Resizing and rotating the trajectory transforms all points simultaneously. For more information see Standard Editing.
- One thing you will probably want to do with a trajectory is modify its points. You can do that via the main or parser Inspector Editor or by editing it directly with the mouse controls. Initially, the spline knots (tangent end points) are under the vertex points so you will not be able to get at them by clicking on them. You should first smooth out the graphic so that the control knots move away from the vertex points and then enter the mouse edit mode and move those knots as you wish.
- To program a trajectory see the Programming section.

Data Editor

The Data Editor for the Trajectory Graphic is shown below.

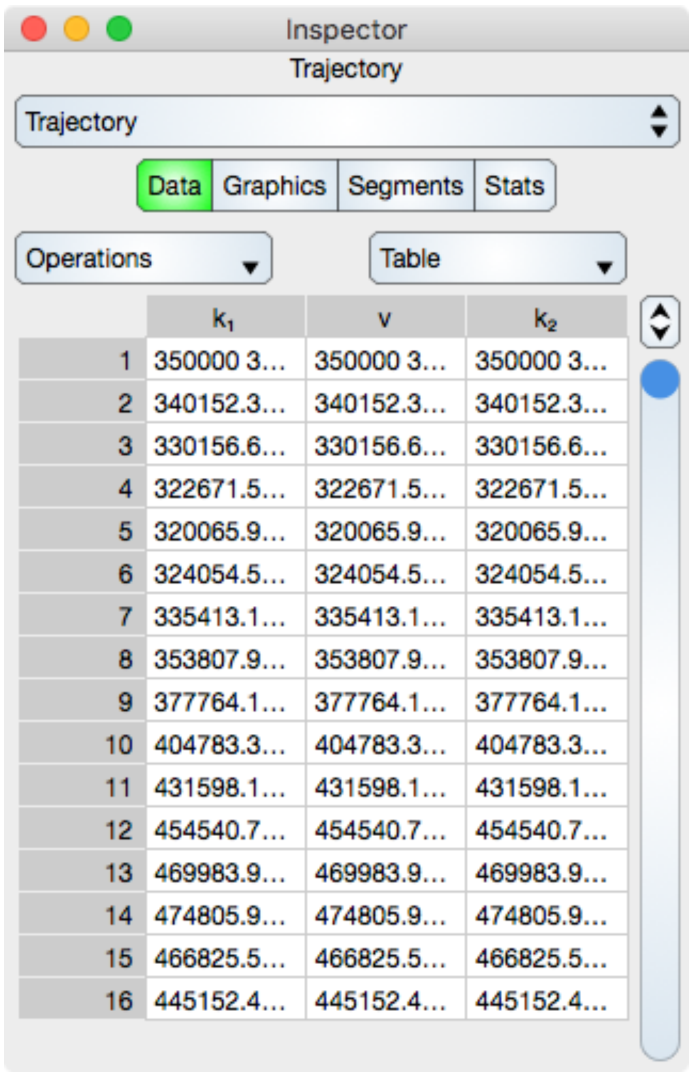
Trajectory Specific Controls

Operations : Select this to perform common sequence operations upon the data. The most important sequence operation is Switch X and Y because that transposes the data and hence the dimensions. Contrast this to the Function graphic which assumes that x is alway ascending and always the independent variable of the coordinate system (graph).

Table

Table controls are described in the Tables section.

The rows represent vertex and knot values. While in Atomic mode, the cell represents a point (x and y value) and while in component mode the cell represents either a x or y value. Hence, in atomic mode there are three column while in component mode there are six columns.



	k_1	v	k_2
1	350000 3...	350000 3...	350000 3...
2	340152.3...	340152.3...	340152.3...
3	330156.6...	330156.6...	330156.6...
4	322671.5...	322671.5...	322671.5...
5	320065.9...	320065.9...	320065.9...
6	324054.5...	324054.5...	324054.5...
7	335413.1...	335413.1...	335413.1...
8	353807.9...	353807.9...	353807.9...
9	377764.1...	377764.1...	377764.1...
10	404783.3...	404783.3...	404783.3...
11	431598.1...	431598.1...	431598.1...
12	454540.7...	454540.7...	454540.7...
13	469983.9...	469983.9...	469983.9...
14	474805.9...	474805.9...	474805.9...
15	466825.5...	466825.5...	466825.5...
16	445152.4...	445152.4...	445152.4...

Graphics Editor

The Graphics Editor for the Trajectory graphic is shown below.

Common Controls

Controls common to all graphics are described in the Graphics section.

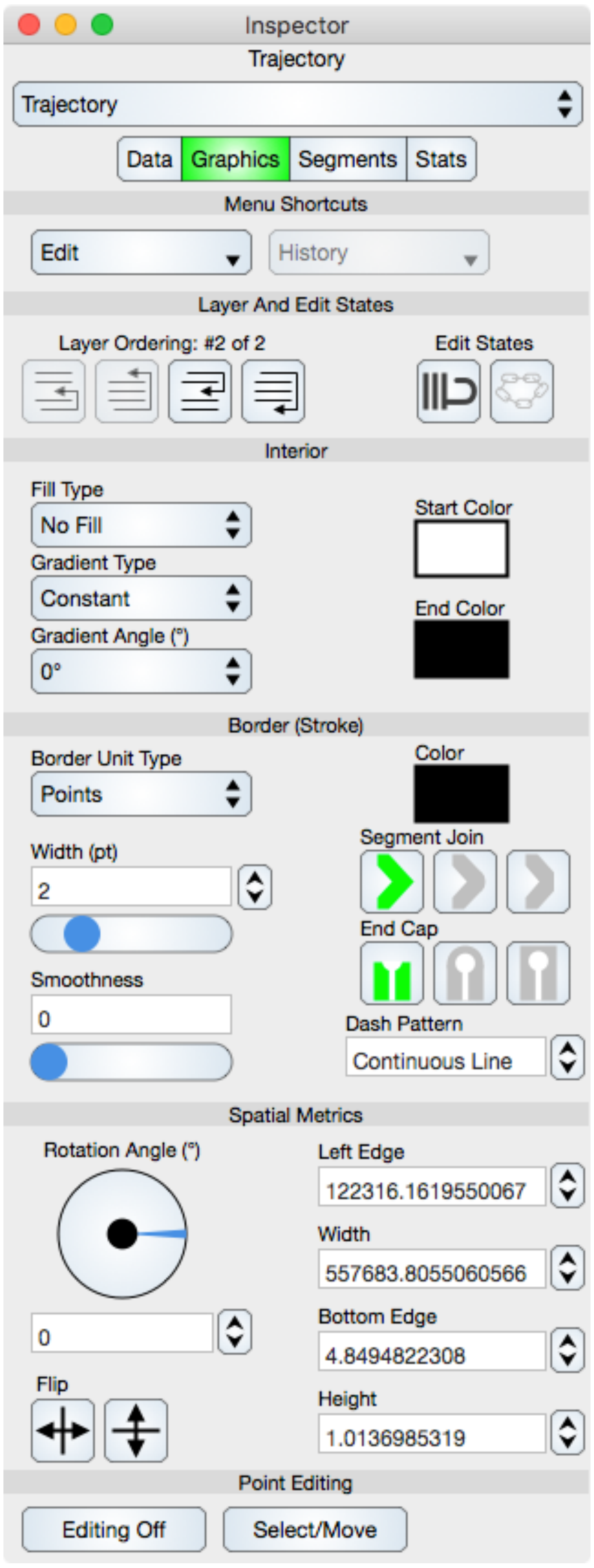
Trajectory Specific Controls

Smoothness : Adjust the spline knots of the Trajectory so that the tangent lines at a vertex are more coincidental. Zero moves the knot locations under the vertex location while one is maximum smoothness. Choosing a smoothness greater than zero is a good way to expose the knots for point editing.

Point Editing

Editing Off/On : Places the graphic into or out of edit mode. While in edit mode the vertices are shown by indicators and can be adjusted using mouse or touch events. Double-clicking the graphic also toggles this edit mode.

Select/Move or Add/Delete : Select/Move mode permits the vertex editing to select and move those locations while Add/Delete mode will delete a vertex if they are hit or add a vertex if a Trajectory segment is hit. This can also be accomplished using the shift key if available.



Segments Editor

The Segments Editor for the Trajectory graphic is shown below.

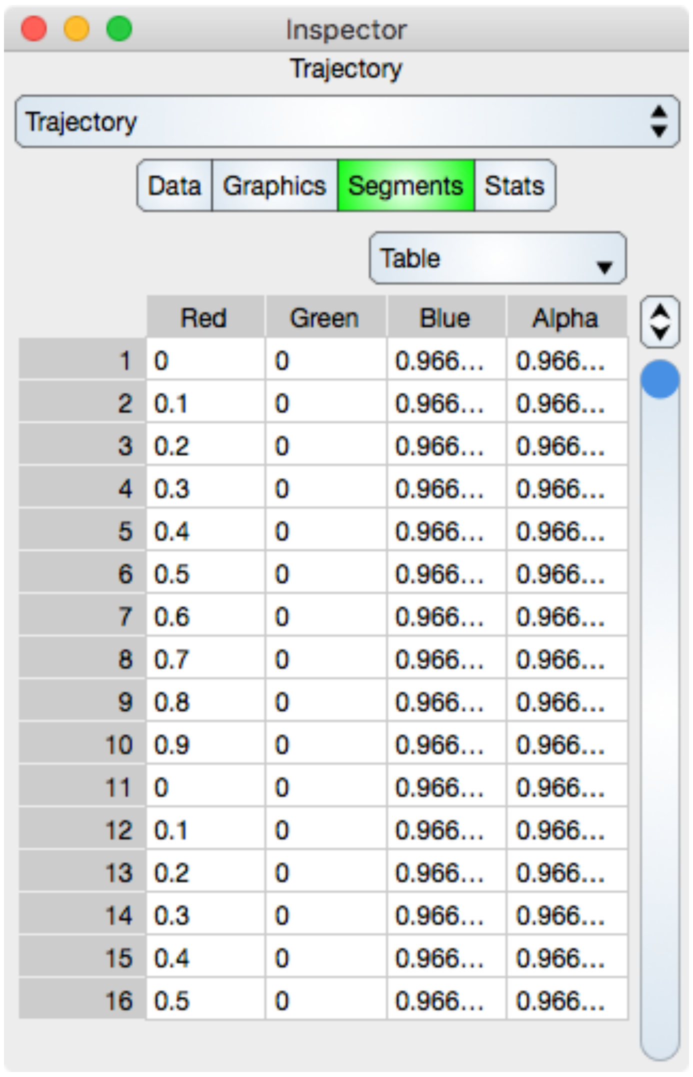
Segments are the parametric Cubic Bezier segments between adjacent vertices. See Sequence Colors for additional information.

Table

Table controls are described in the Tables section.

The rows represent a segment color values. While in Atomic mode, the cell represents a rgba value and while in component mode the cell represents either single r, g, b and a. Hence, in atomic mode there is one column while in component mode there are four columns.

Note that if the Data sequence is resorted then the Segments sequence will not be resorted.



	Red	Green	Blue	Alpha
1	0	0	0.966...	0.966...
2	0.1	0	0.966...	0.966...
3	0.2	0	0.966...	0.966...
4	0.3	0	0.966...	0.966...
5	0.4	0	0.966...	0.966...
6	0.5	0	0.966...	0.966...
7	0.6	0	0.966...	0.966...
8	0.7	0	0.966...	0.966...
9	0.8	0	0.966...	0.966...
10	0.9	0	0.966...	0.966...
11	0	0	0.966...	0.966...
12	0.1	0	0.966...	0.966...
13	0.2	0	0.966...	0.966...
14	0.3	0	0.966...	0.966...
15	0.4	0	0.966...	0.966...
16	0.5	0	0.966...	0.966...

Stats Editor

The Stats Editor for the Trajectory graphic is shown below. Note: The data is consider the vertex points while the knots are considered as graphical. Thus the statistics only reflect the vertex values.

Basics

Number Of Points : Shows the number of points in the Trajectory.

X-Minimum : x-minimum value of all the data points.

Y-Minimum : y-minimum value of all the data points.

X-Maximum : x-maximum value of all the data points.

Y-Maximum : y-maximum value of all the data points.

Basic Distribution

The basic distribution is shown separately for the x-dimension and the y-dimension.

Mean : Shows the mean of the data.

Median : Shows the median of the data.

Standard Deviation : Shows the standard deviation of the data.

Range : Shows the range of the data (maximum - minimum).

Linear Regression Constants

Note that linear regression may also be referred to as line fit.

Slope : Shows the slope of the linear regression.

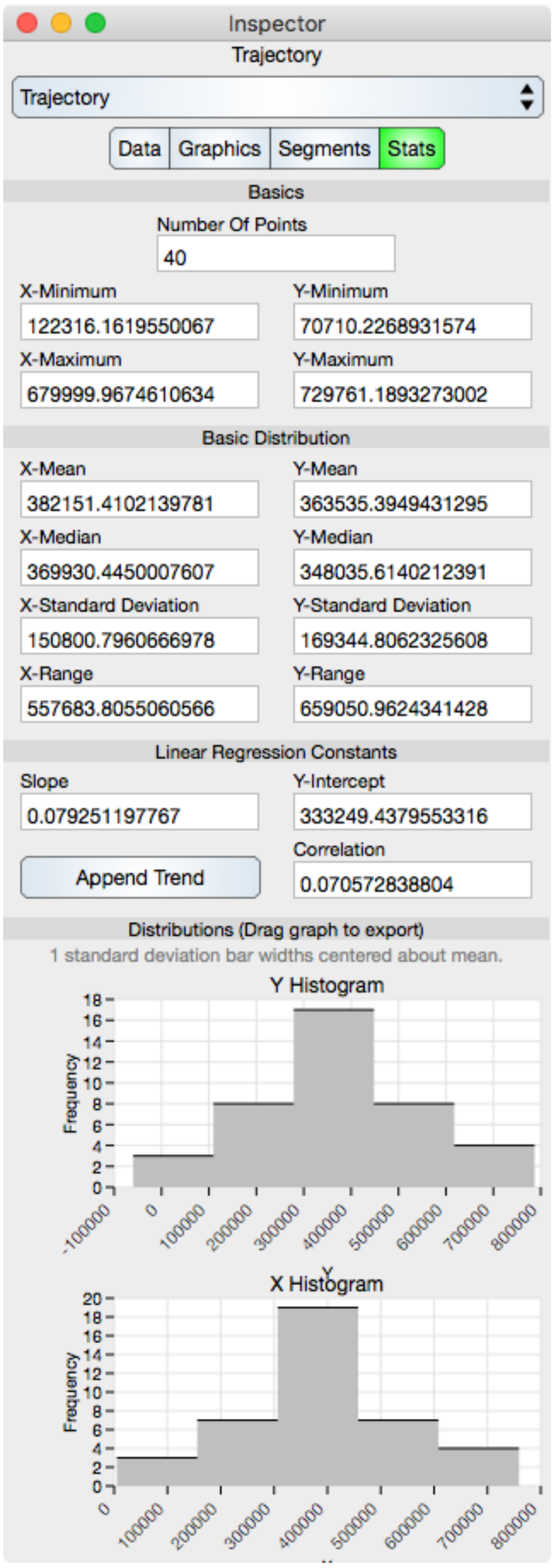
Y-Intercept : Shows the y-intercept of the linear regression.

Correlation : Shows the correlation constant of the linear regression.

Distributions

Y Histogram : Shows the y-histogram of the data. This graph can be dragged onto the Graphic View.

X Histogram : Shows the x-histogram of the data. This graph can be dragged onto the Graphic View.



Inspector

Trajectory

Trajectory

Data | Graphics | Segments | Stats

Basics

Number Of Points: 40

X-Minimum: 122316.1619550067
Y-Minimum: 70710.2268931574
X-Maximum: 679999.9674610634
Y-Maximum: 729761.1893273002

Basic Distribution

X-Mean: 382151.4102139781
Y-Mean: 363535.3949431295
X-Median: 369930.4450007607
Y-Median: 348035.6140212391
X-Standard Deviation: 150800.7960666978
Y-Standard Deviation: 169344.8062325608
X-Range: 557683.8055060566
Y-Range: 659050.9624341428

Linear Regression Constants

Slope: 0.079251197767
Y-Intercept: 333249.4379553316
Correlation: 0.070572838804

Append Trend

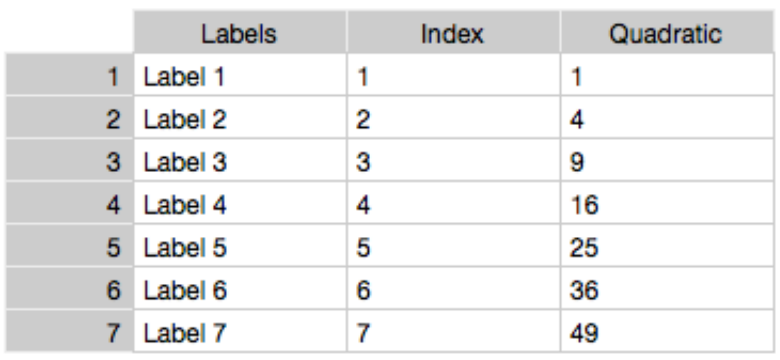
Distributions (Drag graph to export)

1 standard deviation bar widths centered about mean.

Graph IDE ► Data Graphics ► Spreadsheet

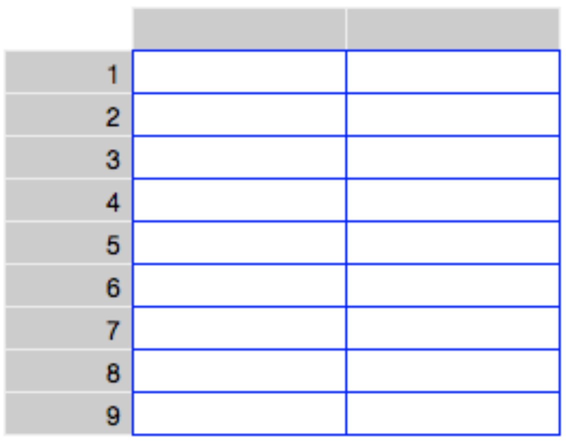
A spreadsheet is a Table with textual cell entries that are stored with the table. The following figure shows some examples of spreadsheets.

A spreadsheet with textual entries



	Labels	Index	Quadratic
1	Label 1	1	1
2	Label 2	2	4
3	Label 3	3	9
4	Label 4	4	16
5	Label 5	5	25
6	Label 6	6	36
7	Label 7	7	49

A blank spreadsheet with customized cell borders



1		
2		
3		
4		
5		
6		
7		
8		
9		

Spreadsheet are Tables, derived from the Group graphic, implement many type of cell editing and generation features and coordinate associated representations which are Pie Chart, Bar And Column Chart, Scatter Graph, Line Graph, Point Map, 3D Point Map and 3D Scatter.

Some standard operations are itemized below.

- To create a spreadsheet bring forward the Graphic Selector, click the spreadsheet factory cell and then drag one from the spreadsheet Factory Inspector. The Dynamic factory cell prototype contains galleries of various spreadsheets with pre-made associated representations.
- Tables are set to resize by adding rows and columns as needed per the Alignment settings.
- When a spreadsheet is used to make an associated representation then those representations are connected to the spreadsheet data. Altering the spreadsheet data alters the associated representation and visa versa. If a spreadsheet is deleted then the connection to the associated representations are broken but the associated representations still exist, and visa versa. If a spreadsheet is copied with the associated representations then the connections are maintained.
- If the associated representation requires a graph then the graph nearest to the spreadsheet will be used. To use multiple graphs first move a graph close to the spreadsheet and make the representation then move the graph away from the spreadsheet and move a different graph near the spreadsheet and make another representation.
- Once an associated representation is made then that representation will be reused when the data in the spreadsheet is updated. That means any alterations to the representation such as colors, stroke widths, markers and other graphical effects will be permanent even though the data is changing.
- For more information on editing see Standard Editing.

A spreadsheet does not translate its cell textual values, rather it simply stores them. However, spreadsheets can reformat their cell entries giving the appearance of translation. Spreadsheets may be a convenient way to store tabulated textual values in a document which can then be used for other purposes such as general viewing of textual data or copy and paste to other tables found in various inspector editors. Spreadsheets may simply be used to compute column values for placement into other tables.

Spreadsheets (and Tables) can be used right away by dragging one from the factory palette. If you wish to customize a spreadsheet then follow this general guideline: First, resize the spreadsheet so that it only has one column and one row. Then in the Navigator expand the table and select the row, column or data cell Label or Rectangle graphic and alter those graphics. Once the graphics are altered then resize the spreadsheet to the desired number of rows and columns. Cells are duplicated from the last cell (the cells you just altered) so new cells take on graphic attributes that were previously set.

Spreadsheet Editor

The Spreadsheet Editor for the spreadsheet is shown below.

Overall

Table Title : Shows and defines the title of the table. If this title is updated then the titles of all representations associated with the spreadsheet are also updated.

Table Information : Shows the bounding number of rows and columns. Each column or row can be of different length and the bounding information shows the union of row and column lengths (the maximum of all the minimums).

Table Edit Controls

Component : Shows the currently selected table component (a row, column, cell or table) and has a drop down menu of common commands such as copy, paste and delete and if the component is a column then Formula to bring forward the Formula Selector.

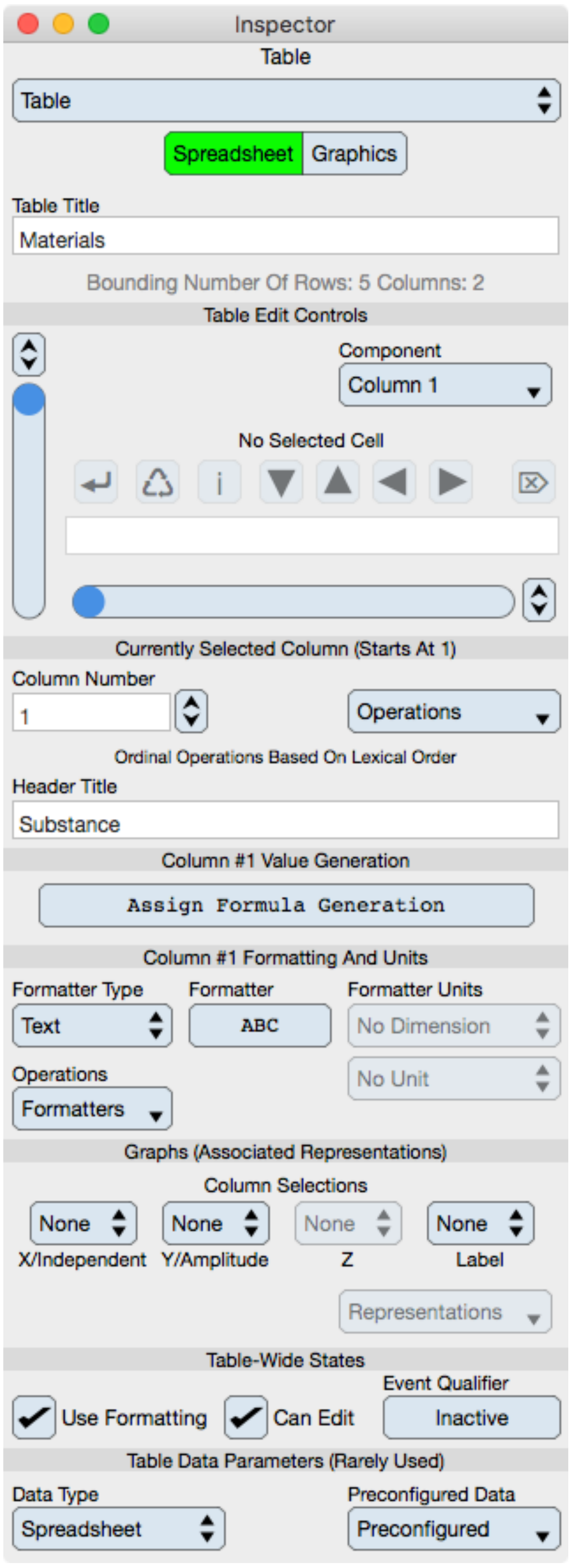

Inspector
Table
Spreadsheet
Graphics
Table Edit Controls
Currently Selected Column (Starts At 1)
Column #1 Value Generation
Assign Formula Generation
Column #1 Formatting And Units
Graphs (Associated Representations)
Column Selections
Representations
Table-Wide States
Table Data Parameters (Rarely Used)

Table: Table

Table Title: Materials

Bounding Number Of Rows: 5 Columns: 2

Component: Column 1

No Selected Cell

Column Number: 1

Operations

Ordinal Operations Based On Lexical Order

Header Title: Substance

Formatter Type: Text

Formatter: ABC

Formatter Units: No Dimension

No Unit

Operations: Formatters

X/Independent: None

Y/Amplitude: None

Z: None

Label: None

✓ Use Formatting

✓ Can Edit

Event Qualifier: Inactive

Data Type: Spreadsheet

Preconfigured Data: Preconfigured

Cell Controls : Used to select and edit a cell in the table. A cell can also be selected by selecting it directly in the spreadsheet itself.

Vertical Slider : Specifies the row offset of the table.

Horizontal Slider : Specifies the column offset of the table.

Currently Selected Column

Column Number : The column index associated with any column-wise operation such as sorting, formula generation, column header editing, column type and format assignments.

Operations : Operates on the data. The current operations are column-wise or full-spreadsheet sort ascending or descending. Note that sorting depends upon the data type so for numbers it is the usual ordinal operation, for strings it is lexical and for date values it is in terms of Julian day fractions (day and normalized-seconds within the day represented by the fractional amount).

Header Title : The title of the column.

Current Column Value Generation

Column entries can be generated using a formula as specified by the Formula Selector. Select the button to bring forward the formula selector.

Current Column Formatting And Units

The spreadsheet is a table with textual data. That textual data can be assigned a type and formatting on a per column basis. If a type is unassigned then the type is implied, either textual, scalar or date as the usage may indicate. For better control of data use, assign a type and then alter the formatter as needed.

Formatter Type : Shows the current type assigned. The default is None and can be changed to Text, Number or Date.

Formatter : Once a type is selected then select the Formatter Button to bring forward the Format Selector. Use that selector to enter formatting parameters. Formatting affects the translation of the data, but not the data itself.

Formatter Units : The Number Formatter accepts definable units Format Selector while other formatters implement implicit units. Use the unit selector to specify units of the data. This, in turn, is used to set units for representations (graphs) and also to limit graphs that can associate with the spreadsheet by the use of unit analysis (the units of the spreadsheet and graph must match).

Operations : Use this drop down to choose a formatter operation, typically to delete a formatter.

Graphs (Associated Representations)

This defines an association between a representation and a column. Once the representation is made then these associations can be changed using the representation's Arranger sub editor.

X/Independent : Specifies the column to associate with the independent variable (x axis). If None then the independent variable is unitized.

Y/Amplitude : Specifies the column to associate with the dependent variable (y axis) or amplitude values (in the case of a pie chart).

Z : Specifies the column to associate with the z axis. This is only enabled if a 3D perspective graph with no data is on the overlay with the spreadsheet.

Label : Specifies the column to associate with the label entries. If None then no labels are used.

Representations : Select an entry in the drop down menu to make a representation or to delete the associated representations. If the associated representation requires a graph then the operation will choose the graph nearest to the spreadsheet to place the representation in. If a graph does not exist in the Layer of the table then the operations that require a graph are disabled.

Table-Wide States

Use Formatting : When on then the formatters are used. When off then the table strictly represents cell elements without further modifications. You may wish to turn formatting of in order to copy and paste the raw data of the spreadsheet.

Can Edit : Momentarily turns the editing of the data on or off.

Event Qualifier : While Inactive spreadsheet acts like a general graphic, while Active the spreadsheet implements Table interaction so that mouse events operate on it like described in the Tables section, e.g.: The table values can be directly selected and edited. In addition, the associated graphics are also made active so that the Information Selector and Data Selector are enabled, see the Pie Chart section as an example of that.

Table Data Parameters

Data Type : Either Spreadsheet or None. When Spreadsheet then the table has a spreadsheet data type store. When None then the spreadsheet is a Tables and has no data backstore.

Preconfigured Data : One of Empty, Reference or Function. Empty clears all cells (deletes all the data), Reference places Cell descriptions in each cell and Function places various discrete values of a function into each column. Preconfigured data is mostly to give examples of the use of a spreadsheet.

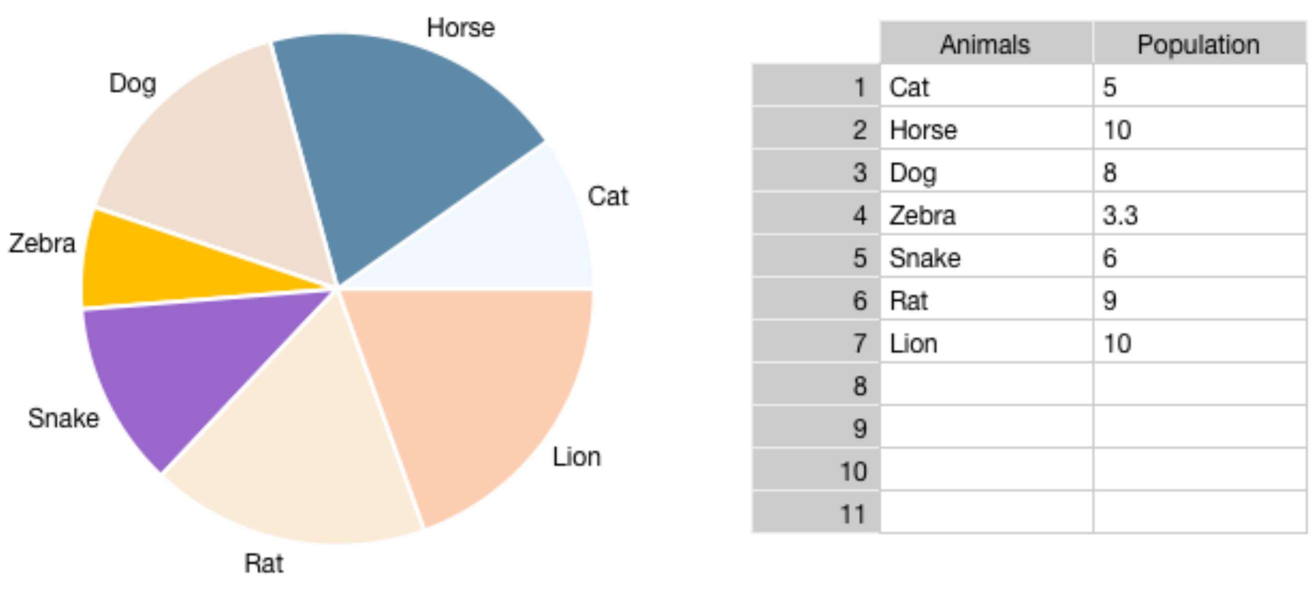
Graphics Editor

The Graphics Editor for the spreadsheet is the same as the Group editor. Only use this editor to alter the appearance of the table.

Graph IDE ▶ Data Graphics ▶ Pie Chart

A Pie Chart is a Group of Circles and Labels that is created using a Spreadsheet. The figure below shows a pie chart.

Pie chart on the left was made from columns 1 and 2 in the table on the right



	Animals	Population
1	Cat	5
2	Horse	10
3	Dog	8
4	Zebra	3.3
5	Snake	6
6	Rat	9
7	Lion	10
8		
9		
10		
11		

Each wedge section of the pie chart represents the portion of the data values (amplitudes) in the spreadsheet.

A pie chart can be constructed manually by creating circles and labels, aligning them appropriately and then grouping them. However, it is easier to use a Spreadsheet to create a pie chart.

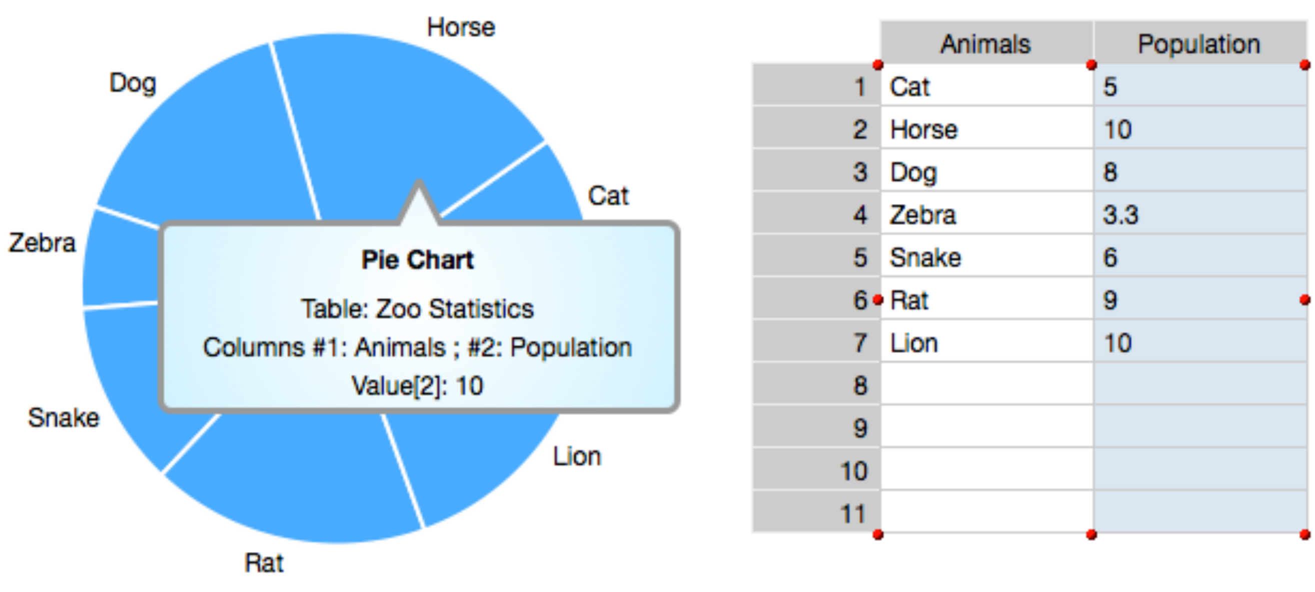
When data in the spreadsheet is changed then the spreadsheet will also change the pie chart. If the spreadsheet that made the pie chart is deleted then the pie chart will still operate as a pie chart, but data derived properties such as the wedge angles will need to be adjusted manually. For that reason, the spreadsheet should probably not be deleted unless the pie chart data is static. If the pie chart is ungrouped then the pie chart feature is deleted.

The pie chart group subgraphics (elements) can not be altered directly but the Graphic Navigator can be used to select an element of the group and then that element's attributes can be modified. When the data in the spreadsheet is modified then the pie chart will be updated and the existing element attributes will be maintained.

Information Selector

When the Spreadsheet is made Active then all the associated graphics are also made active which means they will show the Information Selector when hovered over and when a wedge element is selected then a Data Selector comes forward to change that wedge amplitude which will also change the spreadsheet value. An example is shown in the following figure.

Pie chart on the left was made from columns 1 and 2 in the table on the right



Pie Chart

Table: Zoo Statistics

Columns #1: Animals ; #2: Population

Value[2]: 10

	Animals	Population
1	Cat	5
2	Horse	10
3	Dog	8
4	Zebra	3.3
5	Snake	6
6	Rat	9
7	Lion	10
8		
9		
10		
11		

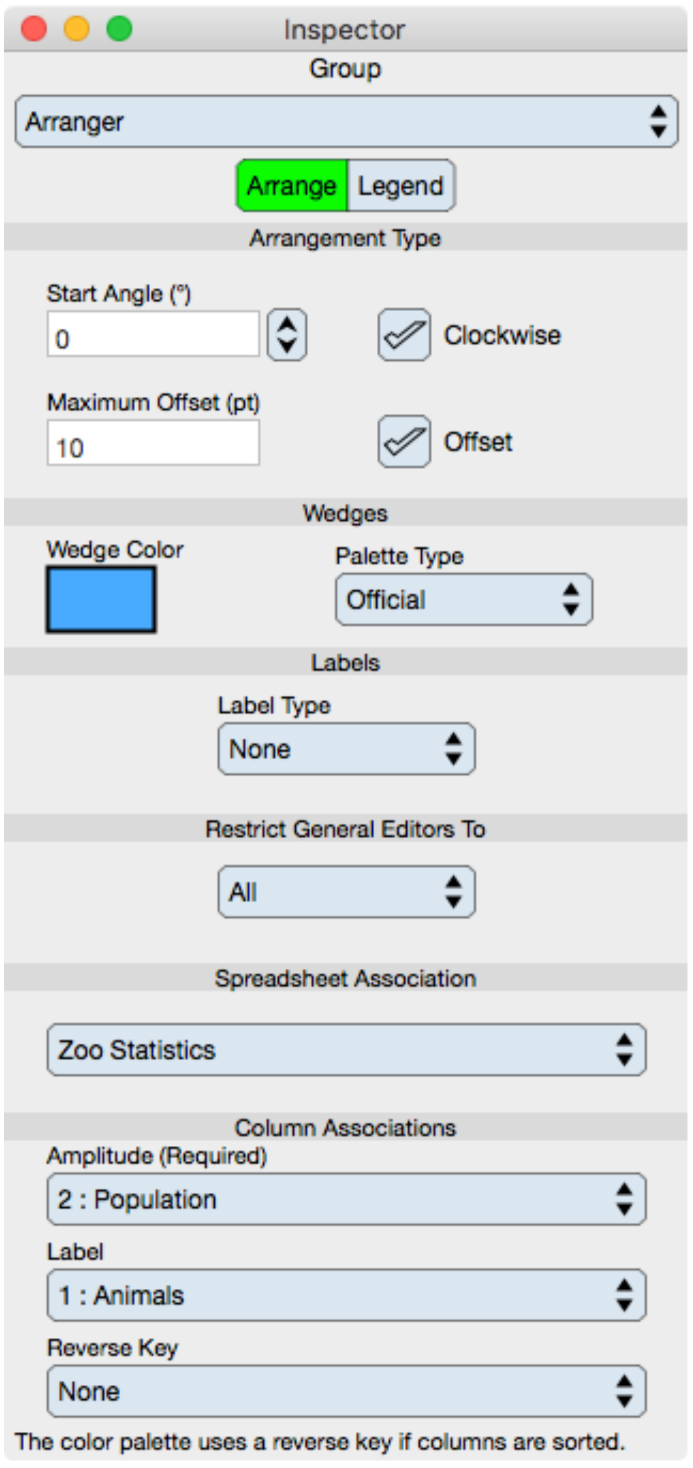
Arranger Editor

A pie chart is a group with a specific arranger so that the main inspector editor for a pie chart is the Group inspector editor. The Arranger sub-editor is where the pie chart specific controls are and is described below.

Note: If you ungroup the pie chart elements then its arranger is dereferenced and deleted. If you then group those same elements then the arranger is the Alignment editor for a general group.

Arrange

The Arrange controls modify pie chart specific arrangement of the wedge sections and labels.



Start Angle (°): 0
Clockwise ✓
Maximum Offset (pt): 10
Offset ✓
Wedge Color: 🟦
Palette Type: Official
Label Type: None
Restrict General Editors To: All
Spreadsheet Association: Zoo Statistics
Amplitude (Required): 2 : Population
Label: 1 : Animals
Reverse Key: None
The color palette uses a reverse key if columns are sorted.

Arrangement Type

Start Angle : By default, the pie sections start at angle zero from the y=0, x-positive ray and proceed counterclockwise. This default is consistent with the mathematical representation, but may not reflect classical usage. Modify the start angle as needed. A typical domain-specific start angle is 90 degrees which means wedges begin a the x=0, y-positive ray.

Clockwise : By default, the pie sections are sequenced in ascending angle counterclockwise. Select this button to order clockwise. A counterclockwise representation is more formal, however a clockwise representation is more consistent with traditional usage in many domain-specific applications.

Maximum Offset : Defines a radial offset of the maximum-value wedge.

Offset : When selected, the maximum-value wedge is offset from the origin (center) of the pie chart by the value in the Maximum Offset field.

Wedges

Wedge Color : Specifies the uniform color of each wedge.

Palette Type : When selected then the colors assigned in a palette as define in the Color Selector section are used. Choose None in order to break the reference and then use the uniform Wedge Color instead.

You may wish to deselect the Use Palette Colors and then modify each wedge color individually by choosing that wedge within the Navigator. Modifications to the graphics of each wedge will be maintained when the data is changed.

Labels

Label Type : The label can be automatically generated by using a specific type. If None is selected then the column associated with the Label (if any) is used. If Amplitude or Sequence is selected then the wedge value or sequence index is used. If the Label column is also chosen then the Amplitude value or Sequence index, prefixed by a space and enclosed in parenthesis, is appended to the Label.

Restrict General Editor To

The main editor which is the Group Editor will operate on all components of the pie chart unless restricted by this control. There are many uses for this restriction, for example to provide a fill background for the labels, but not the wedges; to provide a gradient to the wedges but not the labels; to turn the wedge boundary stroke on or off or change its color without affecting the labels; or a whole host of other uses.

Spreadsheet Association

Defines and selects the spreadsheet used to generate the pie chart. Any spreadsheet can be referenced but that spreadsheet should conform to the intended use. When different spreadsheets are chosen then all representation parameter values are maintained so that this is a good way to flip through alternative data sets while maintaining the pie chart graphical attributes.

Column Associations

The column associations are set during representation creation but can be reset here. Choosing different columns are a good way to quickly view different data sets for one pie chart.

Amplitude : Specifies the column to associate with the relative wedge arc.

Label : Specifies the column to associate with the label entries. If None then no labels are used.

Reverse Key : Specifies the column to associate with reverse keys to external elements such as the color palette. If the table is not sorted then this key is not needed. Sometimes, a sort is performed where external element order association need not be maintain in which case this reverse key is not needed.

Legend

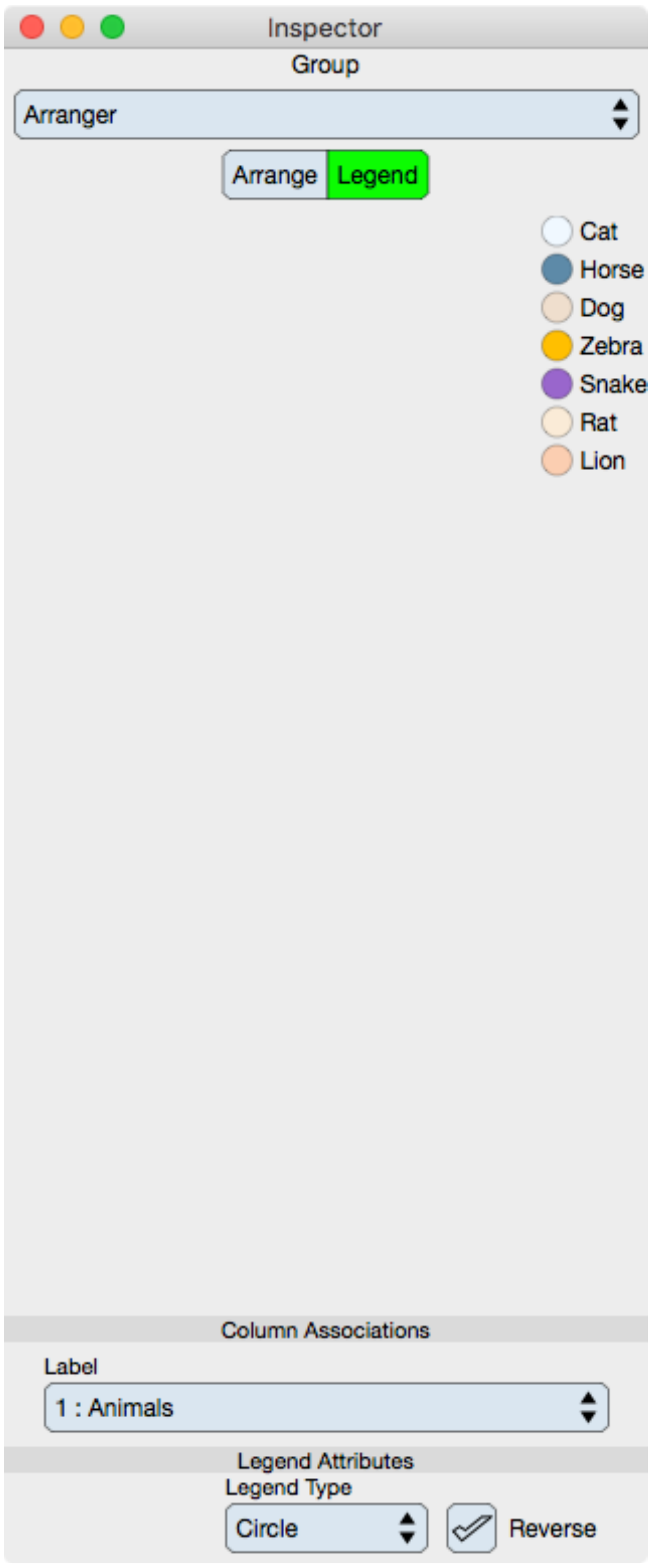
The legend tabloid shows a pre-built legend of the pie chart. It is based on the colors of the wedges and the column associated with the legend labels.

Legend

Legend Group : The legend is a Group Graphic and is built automatically using the attributes specified below. Once the attributes below are specified then drag the legend near your pie chart on the document's Graphic View as it is also an element of a Palette. The legend is set to drag and drop to the overlay layer of the graphic view.

When the legend is dropped onto the graphic view then it is also connected to the spreadsheet associated with the pie chart. Changing the pie chart colors or spreadsheet label column will also change the legend. The legend is a group graphic which means it has an arranger (just like the pie chart). That arranger is accessed via the group editor subeditor.

Legend Attributes



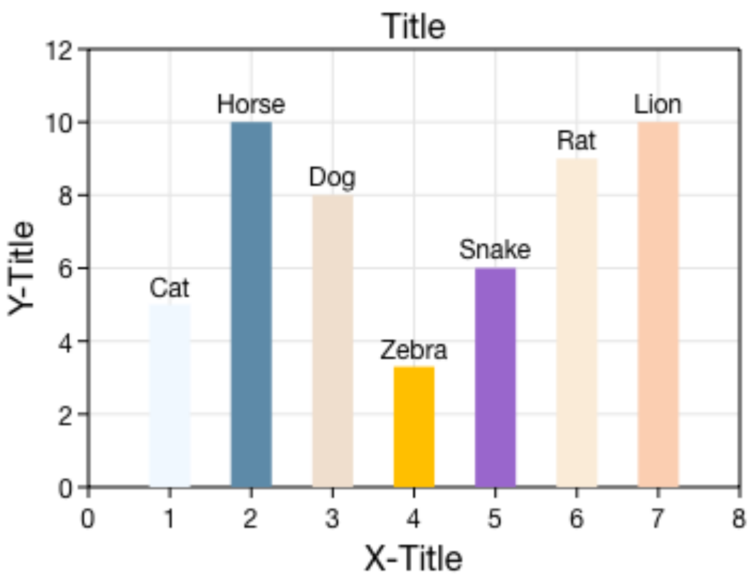
Legend Type : Either Circle or Square. If circle or square then the legend marker is represented by a circle or rectangle respectively whose fill color is that of each pie chart wedge section.

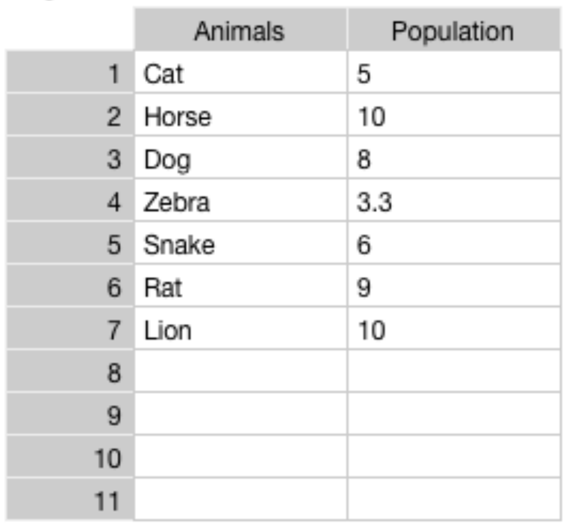
Reverse : The row order of the legend is same as the row order of the data. Select Reverse to reverse that order. This is helpful for many reasons for example, clockwise v.s counterclockwise sequencing of the pie chart data.

Graph IDE ► Data Graphics ► Bar And Column Chart

A Bar or Column Chart is a Group of Rectangles and Labels that is created using a Spreadsheet. The figure below shows a column chart.

Column chart on the left was made from columns 1 and 2 in the table on the right





	Animals	Population
1	Cat	5
2	Horse	10
3	Dog	8
4	Zebra	3.3
5	Snake	6
6	Rat	9
7	Lion	10
8		
9		
10		
11		

A column chart shows scalar amplitudes by adjusting the height of rectangles while a bar chart shows scalar amplitudes by adjusting the width of rectangles. When the distinction between the height and width is immaterial then the bar or column chart is simply referred to as a bar chart, thus making the bar nomenclature generic, and either width or height is referred to as length.

A bar or column chart can be constructed manually by creating rectangles and labels, aligning them appropriately and then grouping them. However, it is easier to use a Spreadsheet to create a bar or column chart.

When data in the spreadsheet is changed then the spreadsheet will also change the column or bar chart. If the spreadsheet that made the column or bar chart is deleted then the column or bar chart will still operate as a column or bar chart, but data derived properties such as rectangle lengths will need to be adjusted manually. For that reason, the spreadsheet should probably not be deleted unless the column or bar chart data is static. If the column or bar chart is ungrouped then the column or bar chart feature is deleted.

The column or bar chart group subgraphics (elements) can not be altered directly but the Graphic Navigator can be used to select an element of the group and then that element's attributes can be modified. When the data in the spreadsheet is modified then the column or bar chart will be updated and the existing element attributes will be maintained.

Note that it only makes sense to make a bar chart on a rectilinear coordinate system. There is no provision for other coordinates.

Arranger Editor

A bar chart is a group with a specific arranger so that the main inspector editor for a bar chart is the Group inspector editor. The Arranger sub-editor is where the bar chart specific controls are and is described below.

Note: If you ungroup the bar chart elements then its arranger is dereferenced and deleted. If you then group those same elements then the arranger is the Alignment editor for a general group.

Arrange

The Arrange controls modify bar chart specific arrangement of the bars and labels.

Arrangement Type

Bar Chart Type : One of Column or Bar. When Column then the Independent axis is the X-Axis. When Bar then the Independent axis is the Y-Axis.

Stacked Bars : When selected then the bars are stacked against the previous bar group, if any. Stacking means that the data values are relative to the previous bar group data values and that the stacked bar width (or height for column charts) is cumulative.

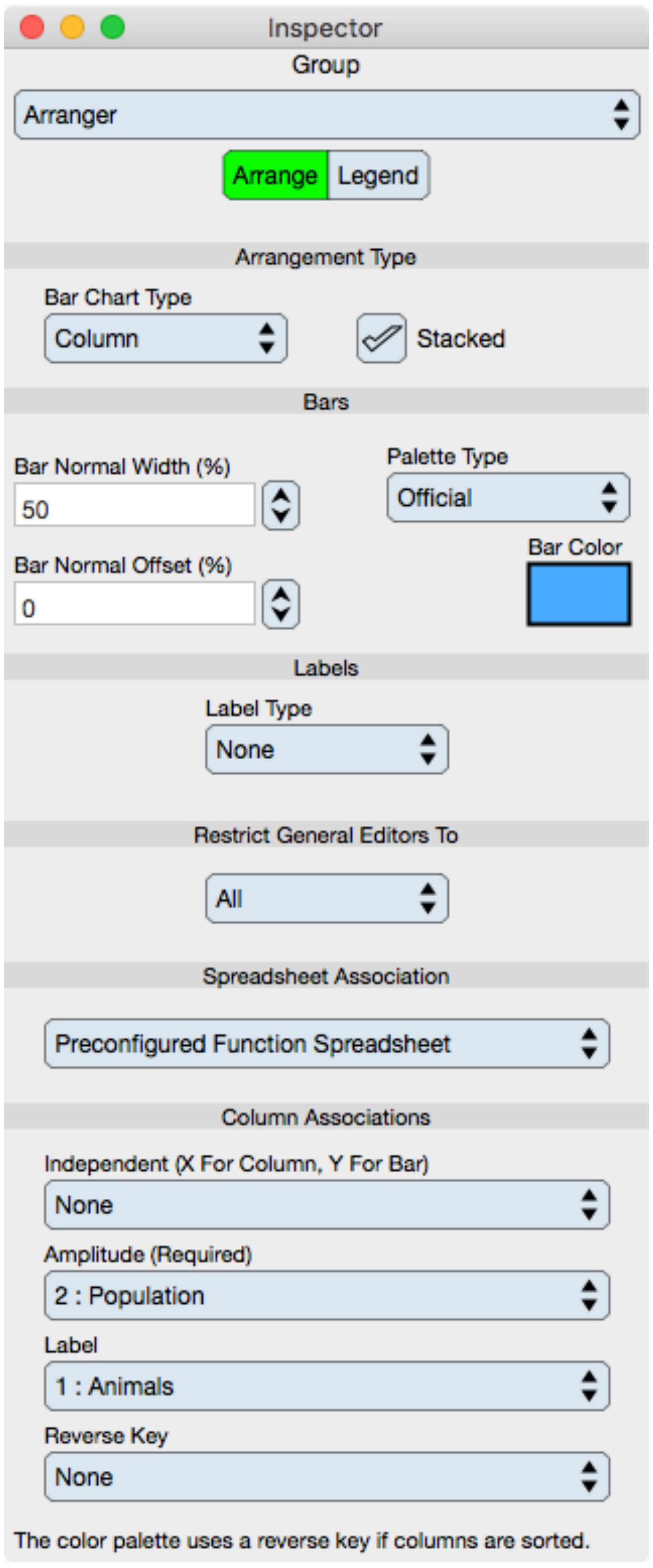
Note that if the independent variable is categorical (textual or date formatted) then if there are repeated category values then the spreadsheet algorithms will stack the repeated values within one data set (column). This is different than the Stacked Bars setting which stacks relative to the previous and separate data set (column).

Bars

Bar Color : Specifies the uniform color of each bar. Generally, a uniform color is suitable for stacked bars and non-uniform colors are suitable for categorical bars. Uniform colors may also be desired when the bar element is referenced to a label.

Palette Type : When selected then the colors assigned in a palette as define in the Color Selector section are used. You may wish to set this control to None and then modify each bar color individually by choosing that bar within the Navigator. Modifications to the graphics of each bar will be maintained when the data is changed.

Bar Normal Width : Specifies the width of the bar relative to adjacent bars. This works best when the bar placement is uniform but if the placement is not uniform then the minimum adjacent distance is used.



The color palette uses a reverse key if columns are sorted.

Bar Normal Offset : Specifies the offset of the bar relative to adjacent bars. This works best when the bar placement is uniform but if the placement is not uniform then the minimum adjacent distance is used. Note that offset is intended to be used when stacking is off. Stacking is an accumulative effect, while offset is an absolute effect.

Labels

Label Type : The label can be automatically generated by using a specific type. If None is selected then the column associated with the Label (if any) is used. If Amplitude or Sequence is selected then the bar value or sequence index is used. If the Label column is also chosen then the Amplitude value or Sequence index, prefixed by a space and enclosed in parenthesis, is appended to the Label.

Restrict General Editor To

The main editor which is the Group Editor will operate on all components of the bar chart unless restricted by this control. There are many uses for this restriction, for example to provide a fill background for the labels, but not the bars; to provide a gradient to the bars but not the labels; to turn the bar boundary stroke on or off or change its color without effecting the labels; or a whole host of other uses.

Spreadsheet Association

Defines and selects the spreadsheet used to generate the bar chart. Any spreadsheet can be choose but that spreadsheet should conform to the intended use. When different spreadsheets are chosen then all representation parameter values are maintained so that this is a good way to flip through alternative data sets while maintaining the bar chart graphical attributes.

Column Associations

The column associations are set during representation creation but can be reset here. Choosing different columns are a good way to quickly view different data sets for one bar chart.

Independent : Specifies the column to associate with the independent dimension (the x-axis for column charts and the y-axis for bar charts). If None then the independent variable is unitized. Note that the independent variable type can be a scalar, date or text. In the case of date or text then the variable is considered categorical; while scalar data is considered ordinal. This distinction allows for stacked bars within one data set when the type is categorical since one category can have duplicate entries.

Amplitude : Specifies the column to associate with the bar length (height for column charts and width for bar charts).

Label : Specifies the column to associate with the label entries. If None then no labels are used. The labels appear adjacent to the bar maximum length.

Reverse Key : Specifies the column to associate with reverse keys to external elements such as the color palette. If the table is not sorted then this key is not needed. Sometimes, a sort is performed where external element order association need not be maintain in which case this reverse key is not needed.

Legend

The legend tabloid shows a pre-built legend of the bar chart. It is based on the colors of the bars and the column associated with the legend labels.

Note that there are two possible legends of interest. If there are more than one bar grouping then the bar chart is either an offset, stacked or overlaid bar chart. In that case the Graph legend may be appropriate because that legend is specific to data elements of the graph and not elements of each bar grouping.

Legend

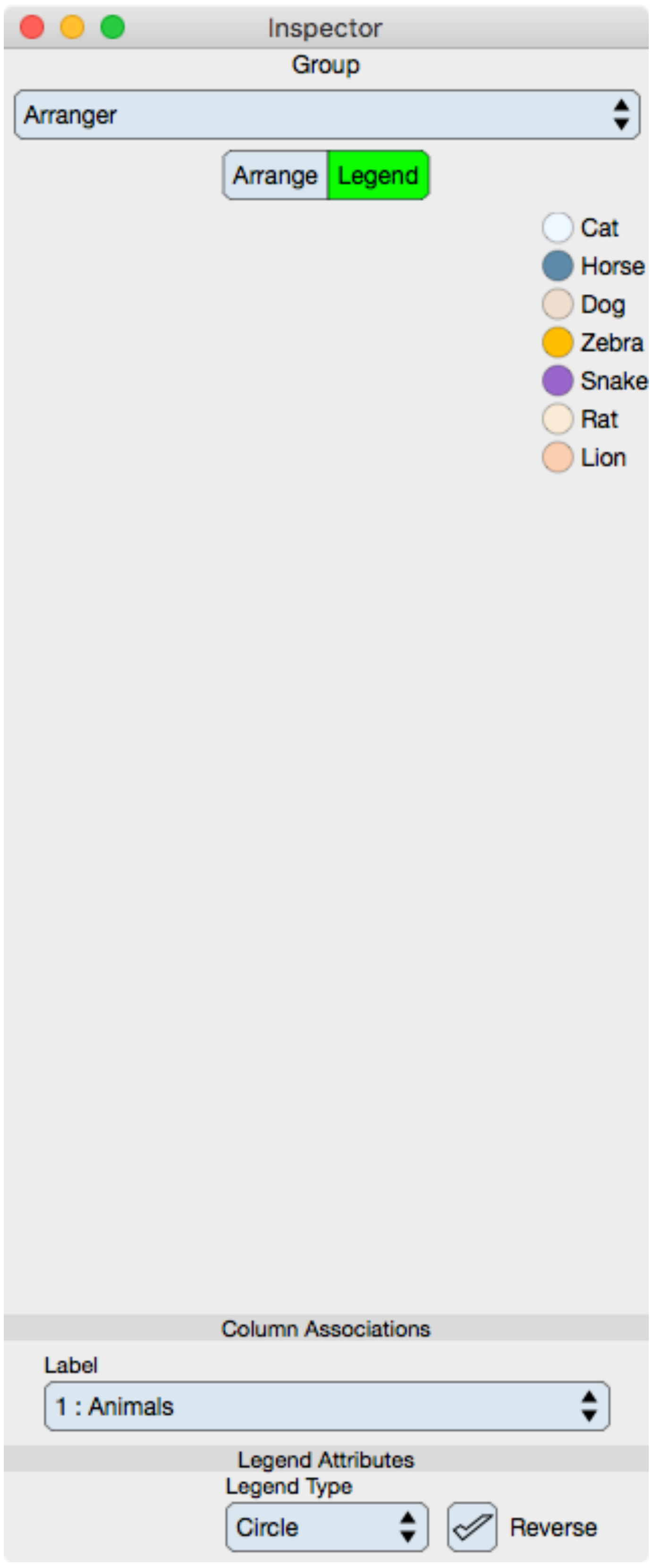
Legend Group : The legend is a Group Graphic and is built automatically using the attributes specified below. Once the attributes below are specified then drag the legend near your bar chart on the document's Graphic View as it is also an element of a Palette. The legend is set to drag and drop to the overlay layer of the graphic view.

When the legend is dropped onto the graphic view then it is also connected to the spreadsheet associated with the bar chart. Changing the bar chart colors or spreadsheet label column will also change the legend. The legend is a group graphic which means it has an arranger (just like the bar chart). That arranger is accessed via the group editor subeditor.

Legend Attributes

Legend Type : Either Circle or Square. If circle or square then the legend marker is represented by a circle or rectangle respectively whose fill color is that of each bar chart bar section.

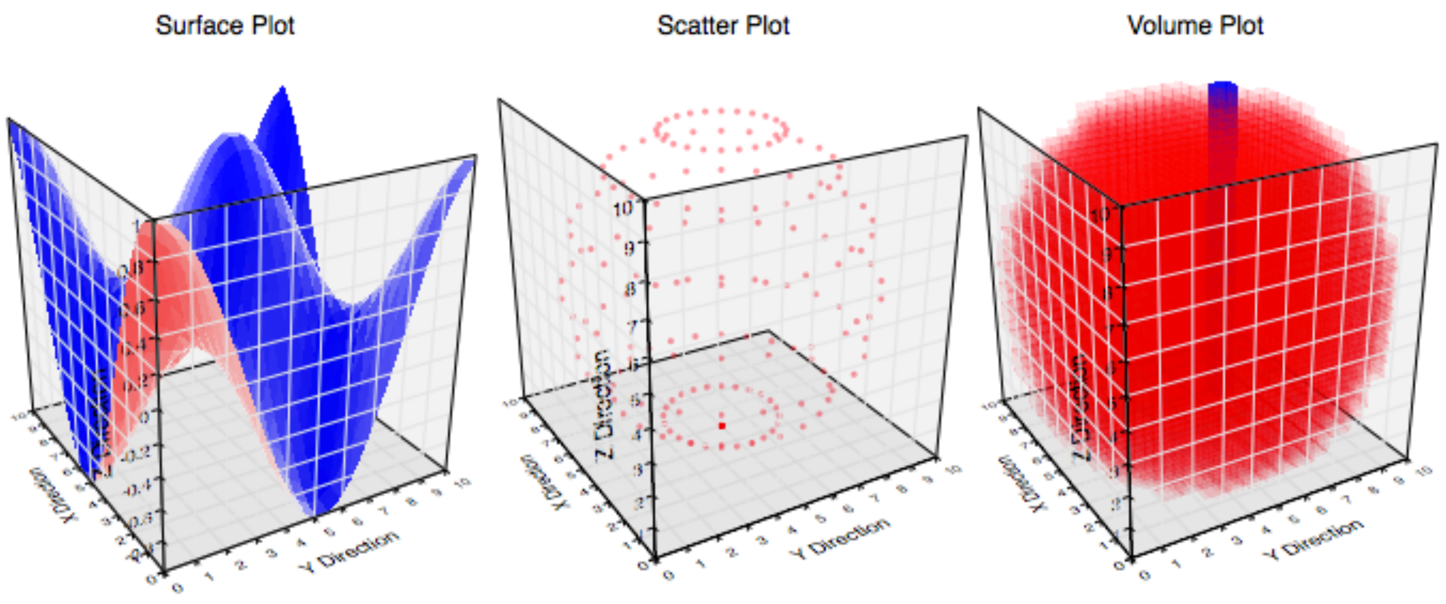
Reverse : The row order of the legend is same as the row order of the data. Select Reverse to reverse that order.


Inspector
Group
Arranger
Arrange
Legend
Cat
Horse
Dog
Zebra
Snake
Rat
Lion
Column Associations
Label
1 : Animals
Legend Attributes
Legend Type
Circle
Reverse

Graph IDE ► 3D Data Graphics

A three dimensional data graphic is a graphic which is defined by three dimensional points and has point-wise graphic representations. When a 3D graphic is created it is always accompanied by a graph. That is because the 3D graphic can not be represented on a page, it needs some sort of projection which is defined by the graph.

Examples of a three dimensional data graphics are shown below.

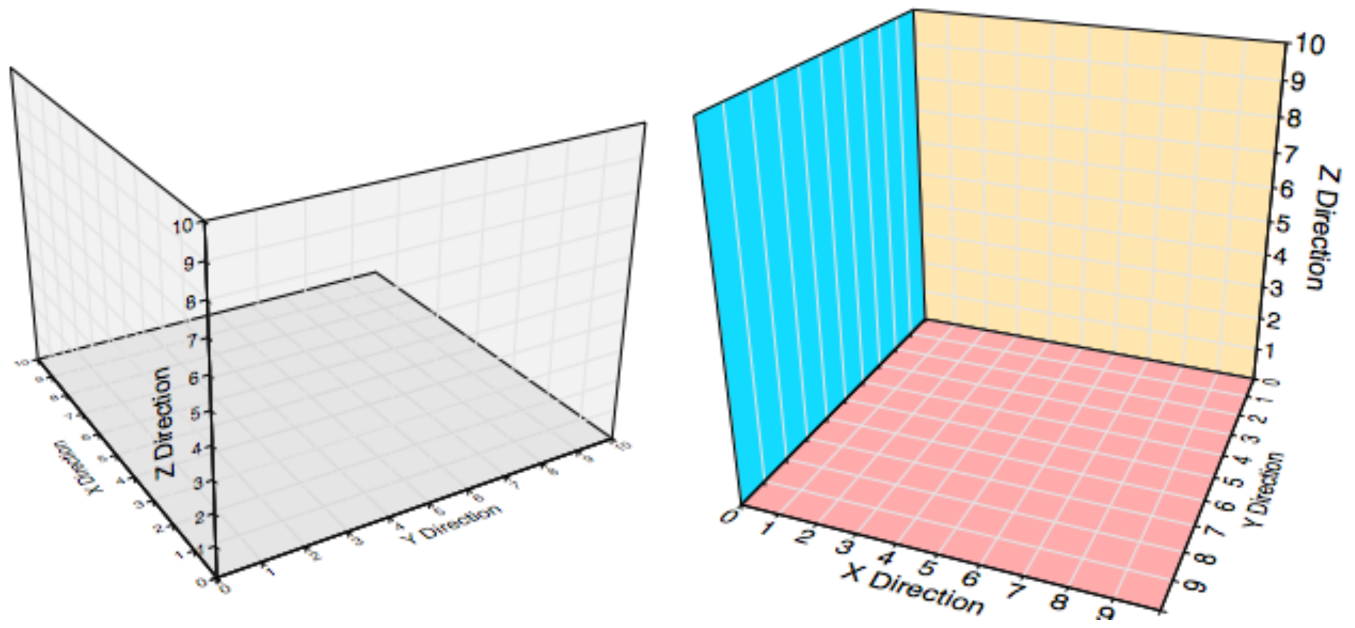


The following is a brief list and definition of three dimensional data graphics:

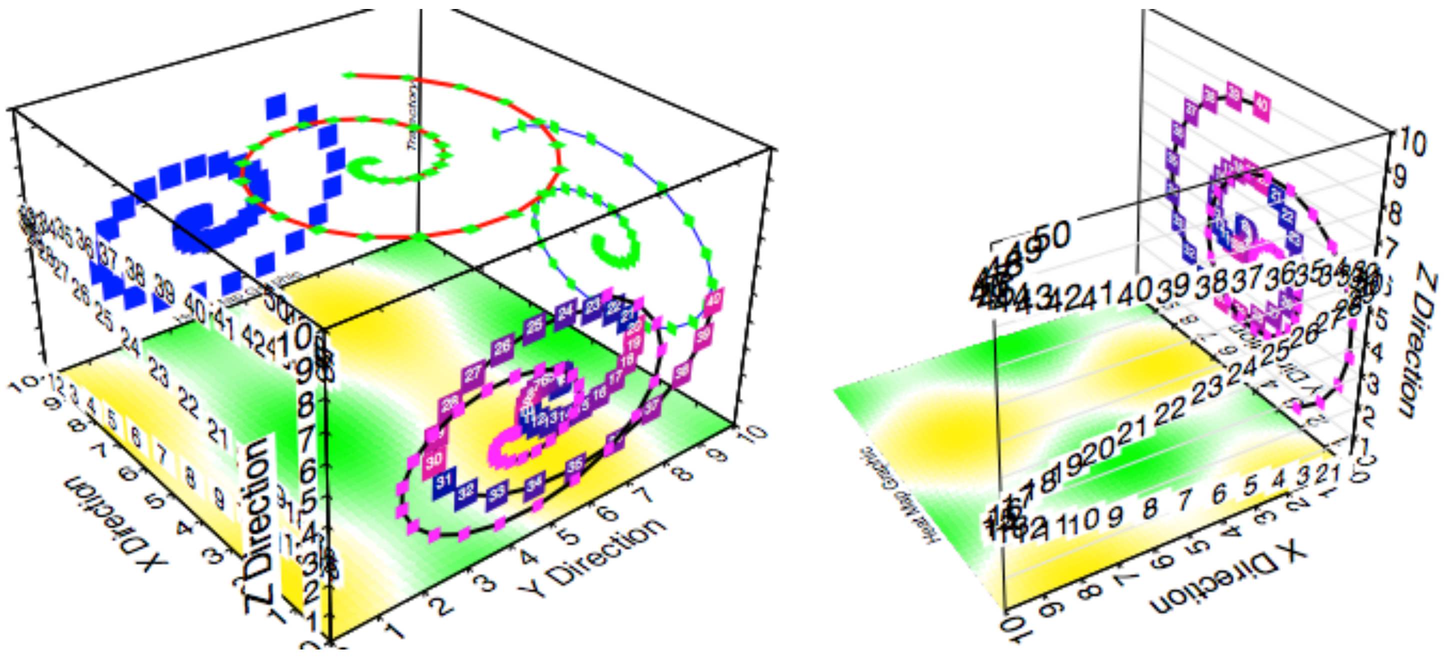
Section	Description
Graph	The graph on which 3D data graphics are made.
Point Map	A Point Map is a regular grid of z-values. The z-values are represented as the z-coordinate value thus producing the three dimensional effect.
Scatter	A scatter is a set of 3D points.
Volume	A volume is a 3D density plot made from a sequence of regularly placed density values and represented by cubes.

Graph IDE ► 3D Data Graphics ► 3D Graph

A 3D Graph is a projection of a 3D rectilinear coordinate system labeled with planar 2D graphs at the 3D rectangular frame. Example of which are shown below.



Each planar 2D graph can also have embedded graphics that can correspond to data in the respective dimensions, perhaps a projected representation of the 3D data. Examples of which are shown below.



Needless to say, the 3D graph can be powerful and it can also be overwhelmingly complex.

Standard Operations

To create a 3D graph bring forward the Graphic Selector, click the 3D graph factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of a predefined 3D graph graphic. You can also create 3D graphs from the standard Palettes menu item or drag them out from the 3D graph Factory Inspector

Inspector Editor

The Inspector Editor for the 3D graph is shown below. Notice how the subeditors give independent access to all the parameters of the projected 2D graph planes. For information on those settings see the Graphs section. In particular notice how you can drill down to each graph plane independently and alter tick, grid and all other graphical and limit attributes. If you do so then note that the 3D limit attributes, as defined below, will override 2D settings when you enter new 3D values. Also be aware that some of the 2D settings are purely graphical in nature so for example if you alter the 2D limits then that will not alter the 3D projection operator coefficients.

Metrics

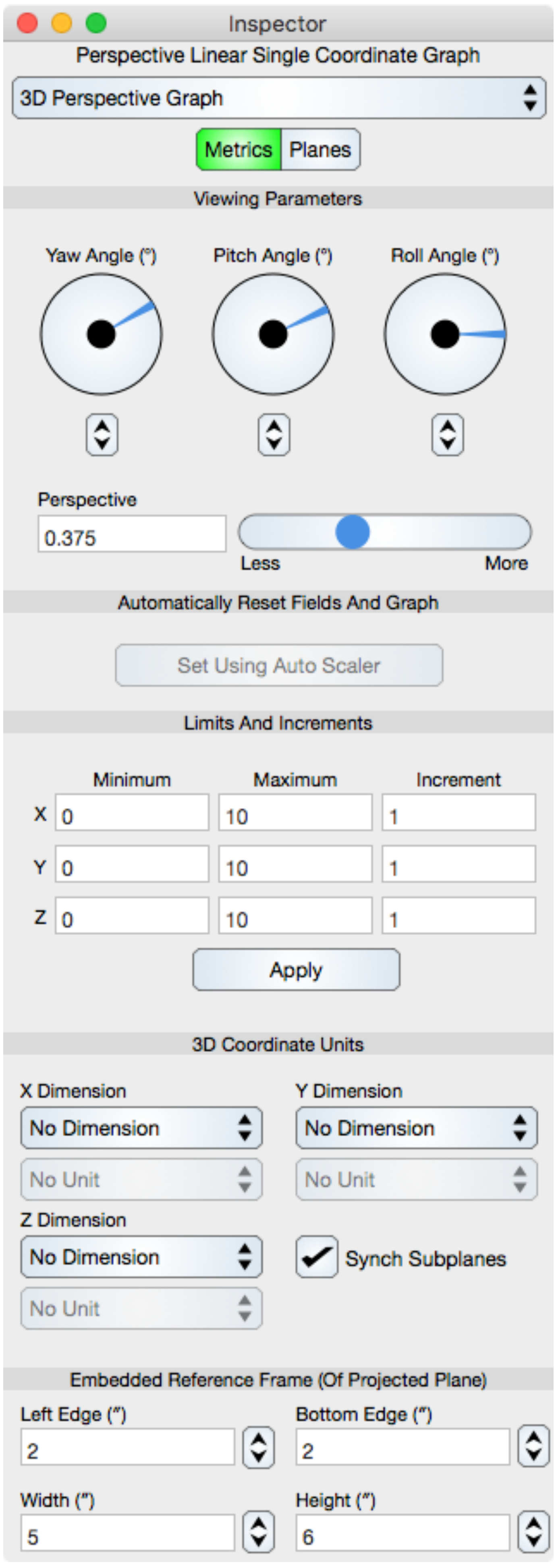
The Metrics editor for a perspective graph is shown below.

Viewing Parameters

Angles : Shows and defines the 3D yaw, pitch and roll angles of the view position. See Dial for a description on controlling the dials.

Perspective : Shows and defines the perspective projection from orthogonal to more skewed projection.

Automatically Reset Fields And Graph


Inspector
Perspective Linear Single Coordinate Graph
3D Perspective Graph
Metrics | Planes

Viewing Parameters
Yaw Angle (°) | Pitch Angle (°) | Roll Angle (°)
Perspective: 0.375 (Less — More)

Automatically Reset Fields And Graph
Set Using Auto Scaler

Limits And Increments

	Minimum	Maximum	Increment
X	0	10	1
Y	0	10	1
Z	0	10	1

Apply

3D Coordinate Units
X Dimension: No Dimension, No Unit
Y Dimension: No Dimension, No Unit
Z Dimension: No Dimension, No Unit
✓ Synch Subplanes

Embedded Reference Frame (Of Projected Plane)
Left Edge ("): 2
Bottom Edge ("): 2
Width ("): 5
Height ("): 6

Set Using Auto Scaler : Select this to set the graph limits to contain the 3D data.

Limits And Increments

When entering the limits and increments for a 3D graph first type in all values and then upon the last value typed, select the Apply button. Do that in order to enter all values at one time for a consistent set of values. The 3D limits must span the data in the 3D space. The entry of limits will not permit data to reside outside the 3D graph frame.

Minimum : Shows and defines the x, y, z minimum axis limits. The minimums must be less than the maximums and also contain the 3D graphics data.

Maximum : Shows and defines the x, y, z maximum axis limits. The minimums must be less than the maximums and also contain the 3D graphics data.

Increment : Shows and defines the x, y, z axis increment value.

Apply : Applies all the limits and increments at the same time.

3D Coordinate Units

X Dimension : Sets the X-Dimension unit. See Unit Selector.

y Dimension : Sets the Y-Dimension unit. See Unit Selector.

Z Dimension : Sets the Z-Dimension unit. See Unit Selector.

Synch Subplanes : When selected then setting the 3D coordinate units also sets the orthographic plane graph units. Those units can also be set by focusing on that graph plane and setting them directly in the graph Single Coordinate Graph editor.

Reference Frame

The reference frame is the frame of the orthographic projection of the 3D graph onto the projection plane. It is also coincidental to the frame of a hidden 2D linear graph. That frame can be altered in the way any 2D graphic can be altered.

Left Edge : Sets the left edge of the reference bounds of the graphic. The units are in the Graphic View units.

Width : Sets the width of the reference bounds of the graphic. The units are in the Graphic View units.

Bottom Edge : Sets the bottom edge of the reference bounds of the graphic. The bottom and y-minimum are the same (the y-coordinate is never flipped). The units are in the Graphic View units.

Height : Sets the height of the reference bounds of the graphic. The units are in the Graphic View units.

Planes

The Planes editor for a perspective graph is shown below. These options are only available for Graph IDE.

Graph Plane States

X-Y Minimum : The X-Y Minimum graph plane.

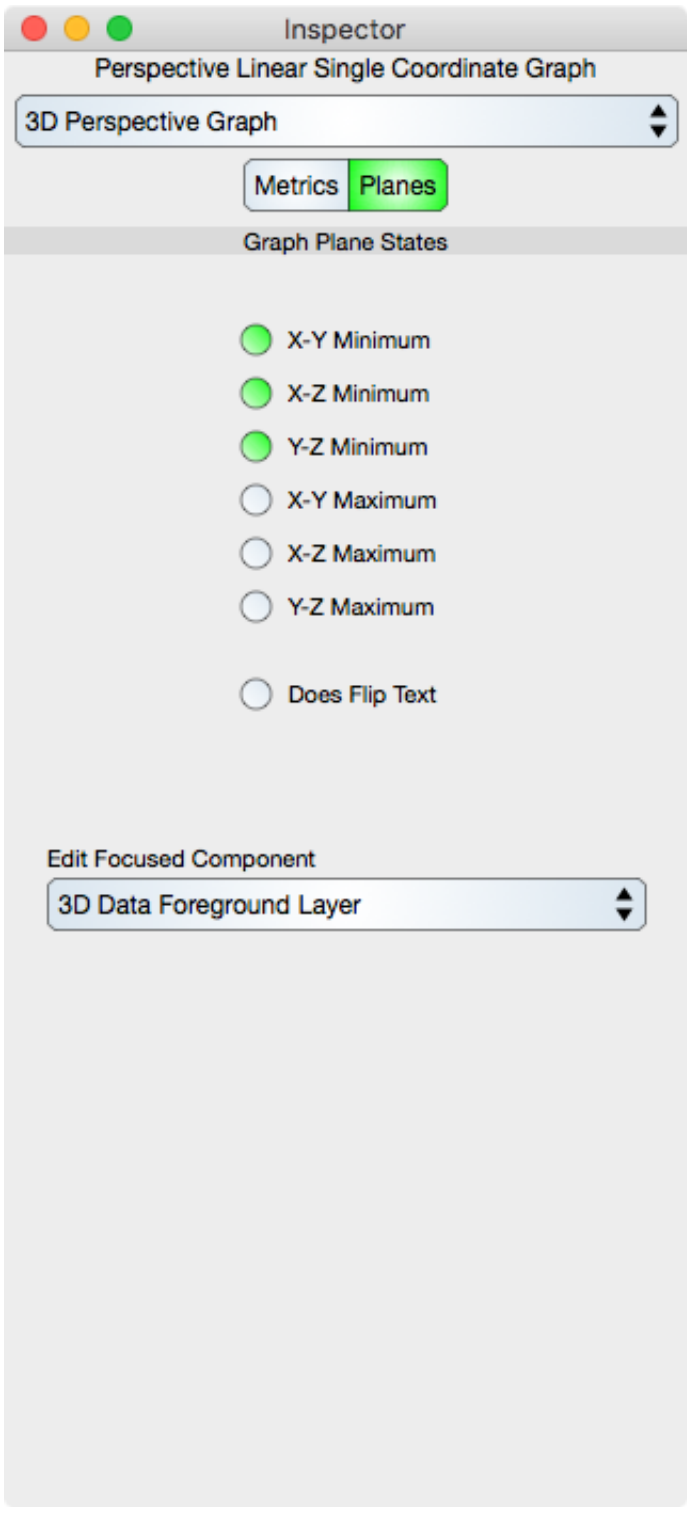
X-Z Minimum : The X-Z Minimum graph plane.

Y-Z Minimum : The Y-Z Minimum graph plane.

X-Y Maximum : The X-Y Maximum graph plane.

X-Z Maximum : The X-Z Maximum graph plane.

Y-Z Maximum : The Y-Z Maximum graph plane.


Inspector

Perspective Linear Single Coordinate Graph

3D Perspective Graph

Metrics | Planes

Graph Plane States

[✓ X-Y Minimum]
[✓ X-Z Minimum]
[✓ Y-Z Minimum]
X-Y Maximum
X-Z Maximum
Y-Z Maximum

Does Flip Text

Edit Focused Component: 3D Data Foreground Layer

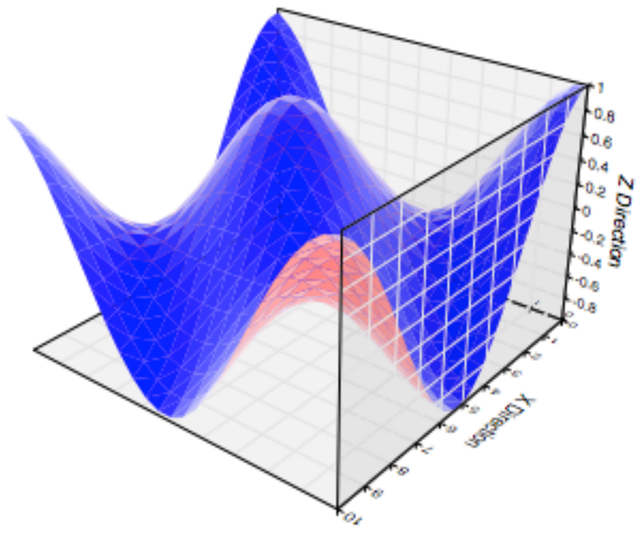
Does Flip Text : Normally text transforms according to the rotation and projection mapping which means that the text can appear from the reverse side, i.e.: it is backwards. Selecting the Does Flip Text option makes all text appear from the front, i.e.: its normal always points towards the observer.

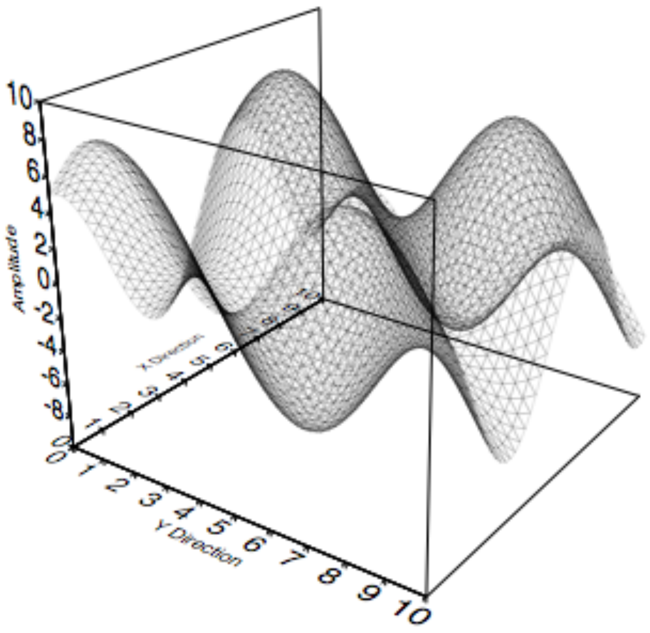
Edit Component

Edit Focused Component : Select this to set the 3D component to edit with the mouse. Normally this is set to focus on the 3D data. 2D data can also appear on each of the six graph planes and this setting is used so that edit events are processed to one of those planes.

Graph IDE ► 3D Data Graphics ► Point Map

A Point Map is a sequence of x-contiguous z-values arranged as a matrix that is mapped to a regular grid of cells. The z-value is mapped to the z-coordinate. The figures below show example point maps.





Some standard operations are itemized below.

- To create a Point Map bring forward the Graphic Selector, click the Point Map factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of a predefined Point Map graphic.
- You can also create 3D point maps from the standard Palettes menu item or drag them out from the 3D point map Factory Inspector.
- One thing you will probably want to do with a Point Map is modify its points. You can do that via the Data or Parser Inspector Editors.
- The Point Map conforms to the Color Map standards where the color map gradient is one of:

Color Mapping Type

No Color Map	The color map is off. This makes most sense while just drawing the stroke so that only the cell boundaries appear.
False Color	The cell color is mapped from arbitrary independent color values inserted into the color table. Notice that a cell is a triangular region that is interpolated from the nearest z-values in the data sequence.
Radiance	The color is mapped based on the angle between the surface normal and the viewing angle. Note that the colors start at negative curl and proceed to positive curl, i.e.: The underside of a surface is the start of the color map and the overside is the end.
Z-Value	The color is mapped from the z-value of the surface. This mapping is very common, albeit somewhat redundant because the surface z-values are already represented by 3d graph although a color mapping of this type is more quantitative because the projection mapping is not present in the color definition.
Distance	The color is mapped from a value related by the distance of the cell to the viewer. The distance is normalized so that the nearest and farthest points of surface define the distance extremum.

- To program a 3D point map see the Programming section.

Data Editor

The Data Editor for the Point Map is shown below.

Table

Apply : Once the grid values have been entered then click the Apply button to redefine the grid.

Table controls are described in the Tables section.

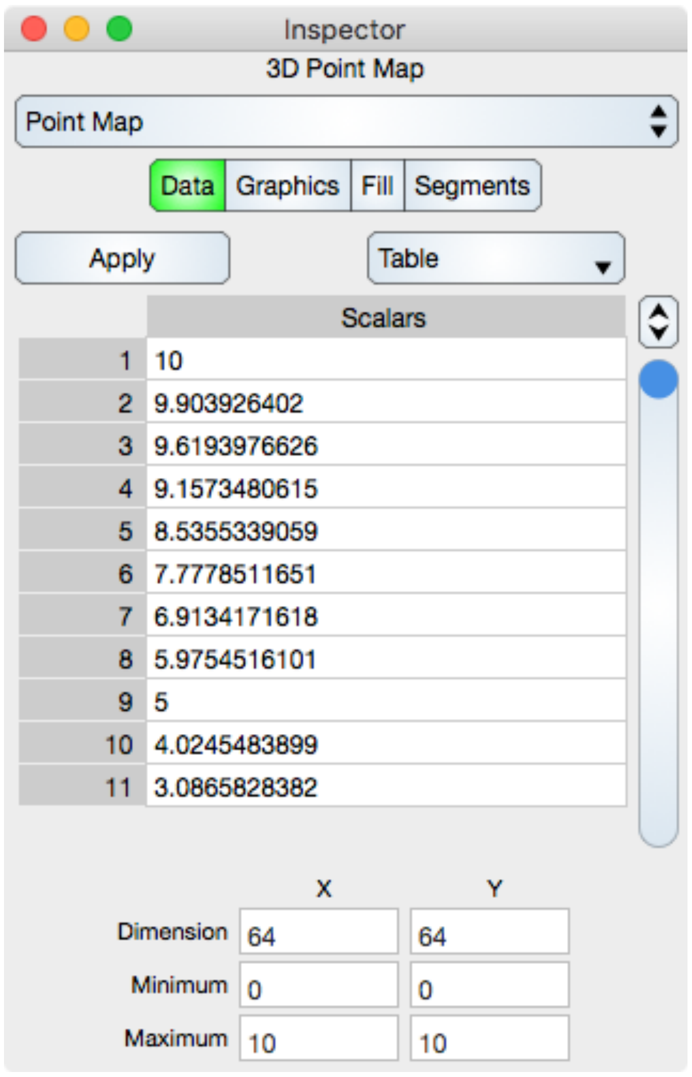
The rows represent the z-value of the regular grid.

Grid Values

The data table does not specify grid parameters, it only specifies amplitude values. That is because a point fill map is on a regular grid. A regular grid is defined by the six values described below.

Dimension : The number of cells in the x and y dimension, aka: The dimensions of the matrix of values. The product of x and y dimensions should equal the number of rows in the table.

Minimum : The x and y minimum of the grid.



	Scalars
1	10
2	9.903926402
3	9.6193976626
4	9.1573480615
5	8.5355339059
6	7.7778511651
7	6.9134171618
8	5.9754516101
9	5
10	4.0245483899
11	3.0865828382

	X	Y
Dimension	64	64
Minimum	0	0
Maximum	10	10

Maximum : The x and y maximum of the grid.

Notice that cell center points, which is also considered where the data is on the grid, are coincidental with the grid minimum and maximum. As a result, the cells along the boundary are actually half cells and at the corners quarter cells. This is a very slight but important distinction.

Graphics Editor

The Graphics Editor for the Point Map is shown below.

Common Controls

Controls common to all graphics are described in the Graphics section.

Color Map Attributes

The color map is further described in the Color Map section.

Graph : Shows a graph of the color map.

Function : Select this to set the color map function. Typically it is set to linear.

Start Color : Defines the start Color of the color map.

End Color : Defines the end Color of the color map.

Number : Defines the number of samples in the color map.

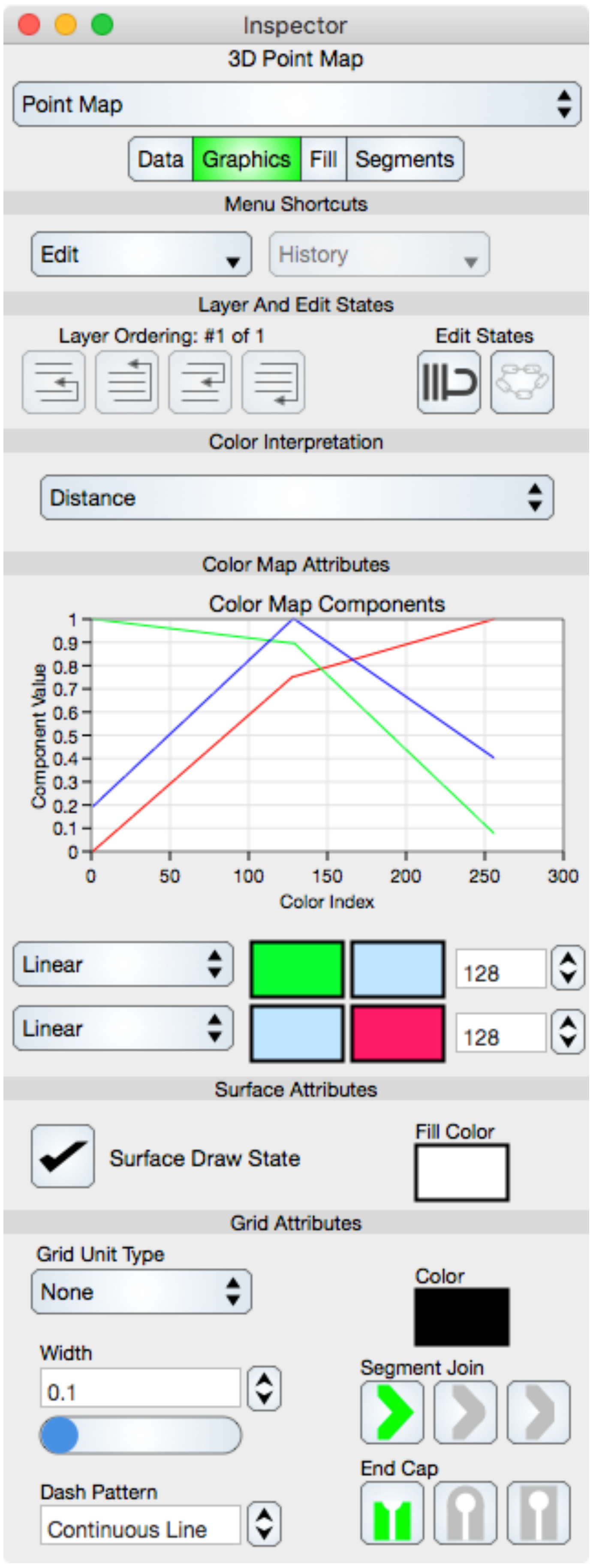
Surface Attributes

Surface Draw State : If checked then the surface is drawn.

Fill Color : Defines the fill Color for the surface. This is normally overridden by the Color Map.

Grid Attributes

Grid attributes are controlled by the normal stroke controls defined in the Graphics section.


Inspector

3D Point Map

Point Map

Data | Graphics | Fill | Segments

Menu Shortcuts

Edit | History

Layer And Edit States

Layer Ordering: #1 of 1

Edit States

Color Interpretation

Distance

Color Map Attributes

Linear 128

Linear 128

Surface Attributes

Surface Draw State ✓

Fill Color

Grid Attributes

Grid Unit Type: None

Color

Width: 0.1

Segment Join

End Cap

Dash Pattern: Continuous Line

Fill Editor

The Fill Editor for the 3D Point Fill graphic is shown below.

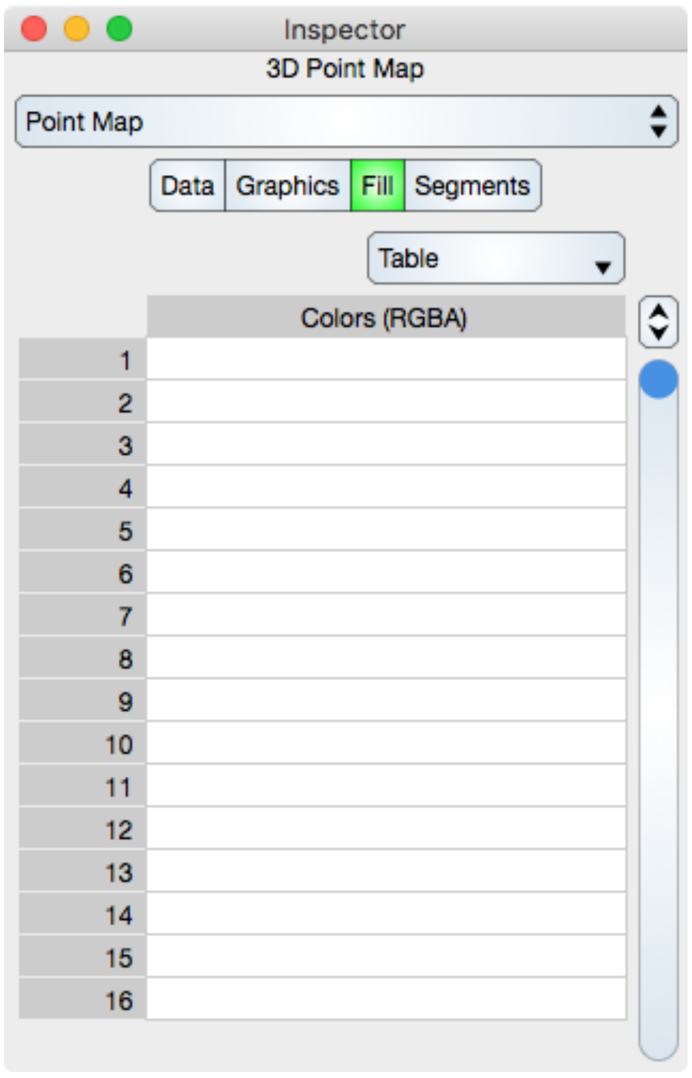
Fill attributes is a false color table of RGBA components, one for each cell. These can be overridden by the Color Interpretation setting and that setting must be False Color Map in order for the elements of this table to take effect.

Table

Table controls are described in the Tables section.

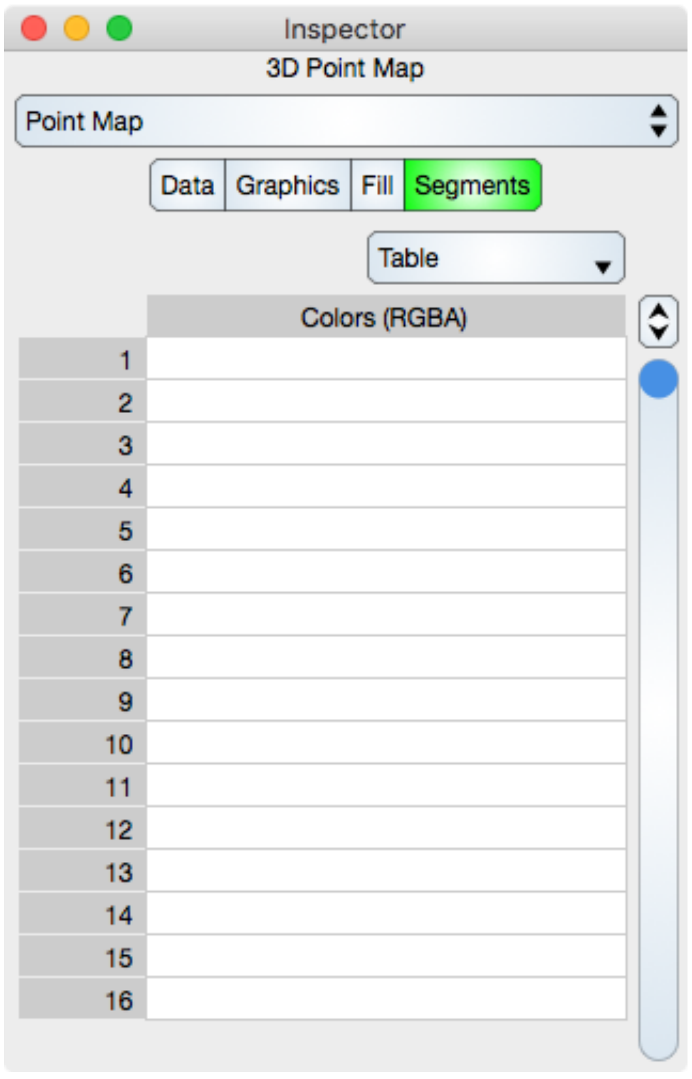
The rows represent a segment color values. While in Atomic mode, the cell represents a rgba value and while in component mode the cell represents either single r, g, b and a. Hence, in atomic mode there is one column while in component mode there are four columns.

Note that if the Data sequence is changed then the Fill sequence will not change.



Segments Editor

The Segments Editor for the 3D Point Fill graphic is shown below.



Segments are the line segments of the grid. See Sequence Colors for additional information. If the diagonal grid line segments are given a transparent color then the grid tessellation can appear to show x or y curves along the surface and those curves can have a defined color mapping and thus show another dimension of the data.

Table

Table controls are described in the Tables section.

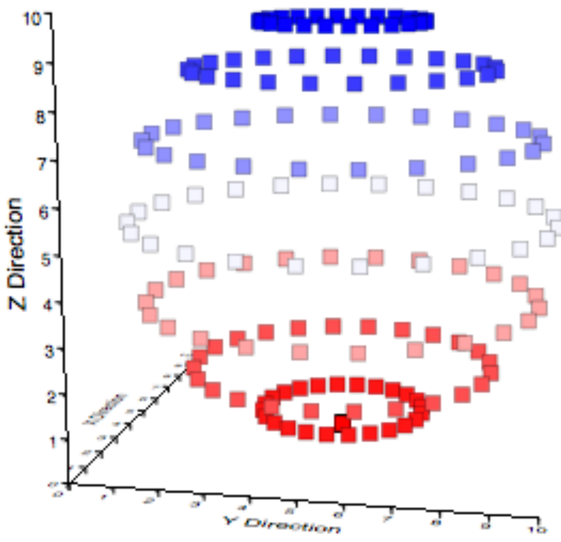
The rows represent a segment color values. While in Atomic mode, the cell represents a rgba value and while in component mode the cell represents either single r, g, b and a. Hence, in atomic mode there is one column while in component mode there are four columns.

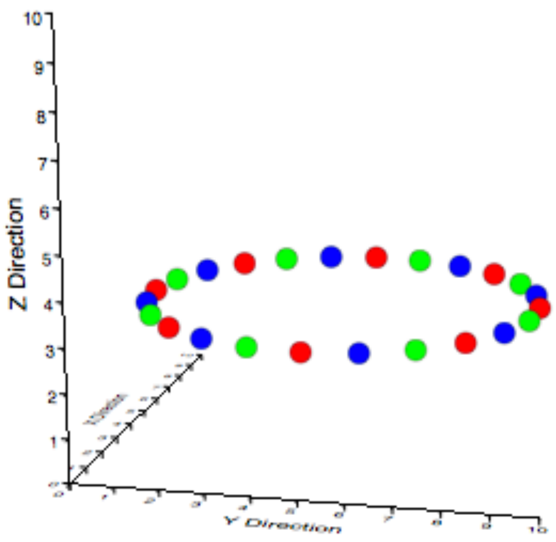
Note that if the Data sequence is changed then the Segments sequence will not be change.

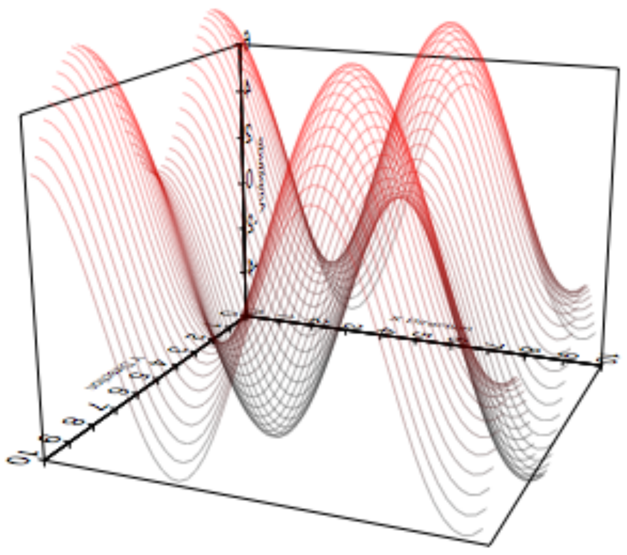
Graph IDE ▶ 3D Data Graphics ▶ Scatter

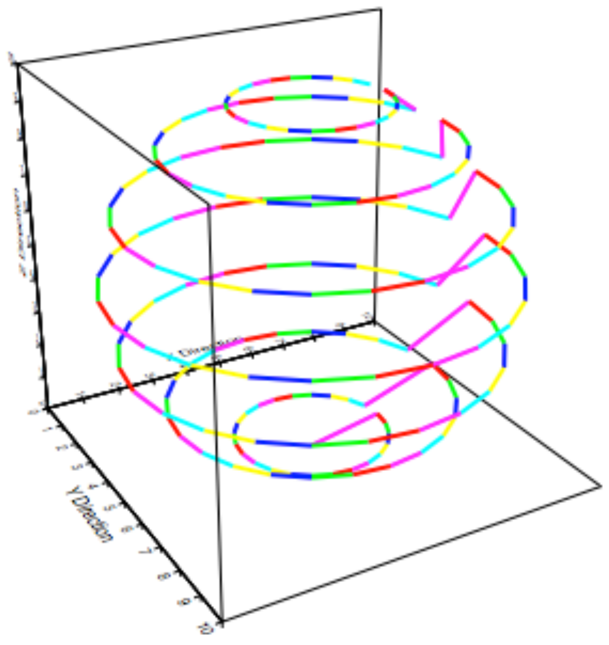
The 3D Scatter graphic represents a sequence of 3D points, examples of which are shown below. Notice that, with the appropriate graphical effects, the 3D points can represent a trajectory or a sequence of independent curves. In addition, with the appropriate colors, the 3D points can indicate separate data sets.

There should only be one 3D graphic per graph to maintain correct z-buffering and coloring is a way to show separate data sets while at the same time permit interleaving of the data sets within the projection.









Some standard operations are itemized below.

- To create a scatter bring forward the Graphic Selector, click the scatter factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of a predefined scatter graphic.
- You can also create scatters from the standard Palettes menu item or drag them out from the scatter Factory Inspector.
- The normal way to modify the points of a scatter is through the scatter's Inspector Editor's data Table. That table accepts a whitespace sequence of x y z triplets in its cells.
- The Scatter conforms to the Color Map standards where the color map gradient is one of:

Color Mapping Type (Applies to Markers Only)

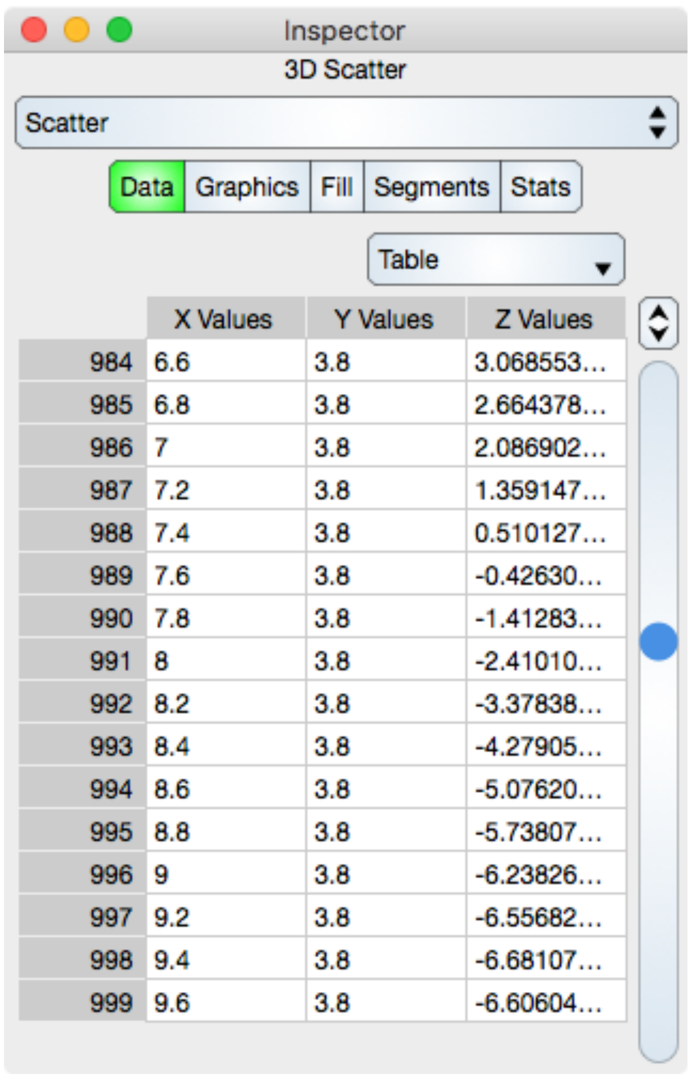
No Color Map	The color map is off. This makes most sense while just drawing the stroke so that only the trajectory appears.
False Color	The color is mapped from arbitrary independent values. Because there is no way to specify another independent value set this UI implementation maps from the z-values of the data. Programs can map from another basis.
Z-Value	The color is mapped from the z-value of the data points. This mapping is very common, albeit somewhat redundant because the data point z-values are already represented by 3d graph although a color mapping of this type is more quantitative because the projection mapping is not present in the color definition.
Distance	The color is mapped from a value related by the distance of the data point to the viewer. The distance

is normalized so that the nearest and farthest points of data define the distance extremum.

- To program a 3D scatter see the Programming section.

Data Editor

The Data Editor for the Point Map is shown below.



	X Values	Y Values	Z Values
984	6.6	3.8	3.068553...
985	6.8	3.8	2.664378...
986	7	3.8	2.086902...
987	7.2	3.8	1.359147...
988	7.4	3.8	0.510127...
989	7.6	3.8	-0.42630...
990	7.8	3.8	-1.41283...
991	8	3.8	-2.41010...
992	8.2	3.8	-3.37838...
993	8.4	3.8	-4.27905...
994	8.6	3.8	-5.07620...
995	8.8	3.8	-5.73807...
996	9	3.8	-6.23826...
997	9.2	3.8	-6.55682...
998	9.4	3.8	-6.68107...
999	9.6	3.8	-6.60604...

Table

Table controls are described in the Tables section.

The rows represent a 3D point while in atomic mode or x, y, z values in each respective cell while in component mode.

Graphics Editor

The Graphics Editor for the Point Map is shown below.

Common Controls

Controls common to all graphics are described in the Graphics section.

Color Map Attributes

The color map is further described in the Color Map section.

Graph : Shows a graph of the color map.

Legend : Shows a fill representation of the color map in both vertical and horizontal orientation. The independent variable of the graph (x-axis for horizontal and y-axis for vertical) is that of the scatter color interpretation. These legends can be dragged to the graphic view as needed. Since the legends are 2D Point Map graphics they can be independently altered once dragged to the graphic view.

Function : Select this to set the color map function. Typically it is set to linear.

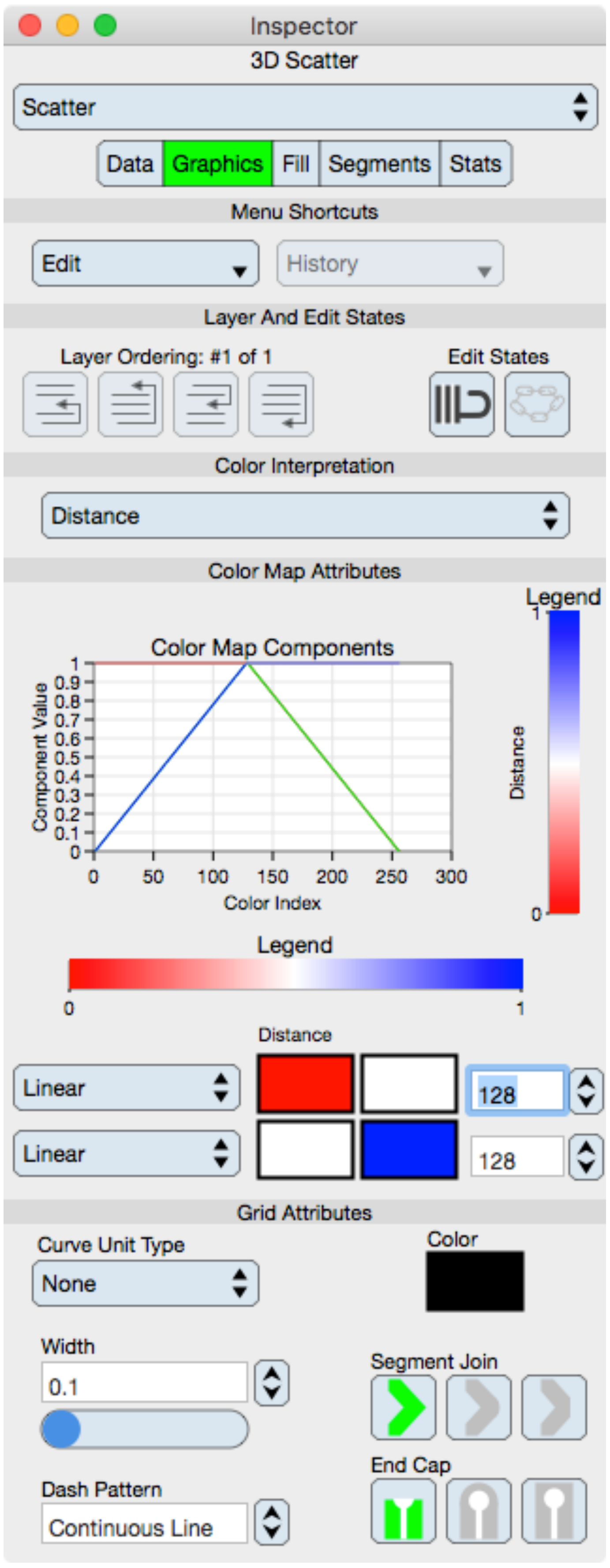
Start Color : Defines the start Color of the color map.

End Color : Defines the end Color of the color map.

Number : Defines the number of samples in the color map.

Grid Attributes

Grid attributes are controlled by the normal stroke controls defined in the Graphics section.



Fill Editor

The Fill Editor for the 3D Scatter graphic is shown below.

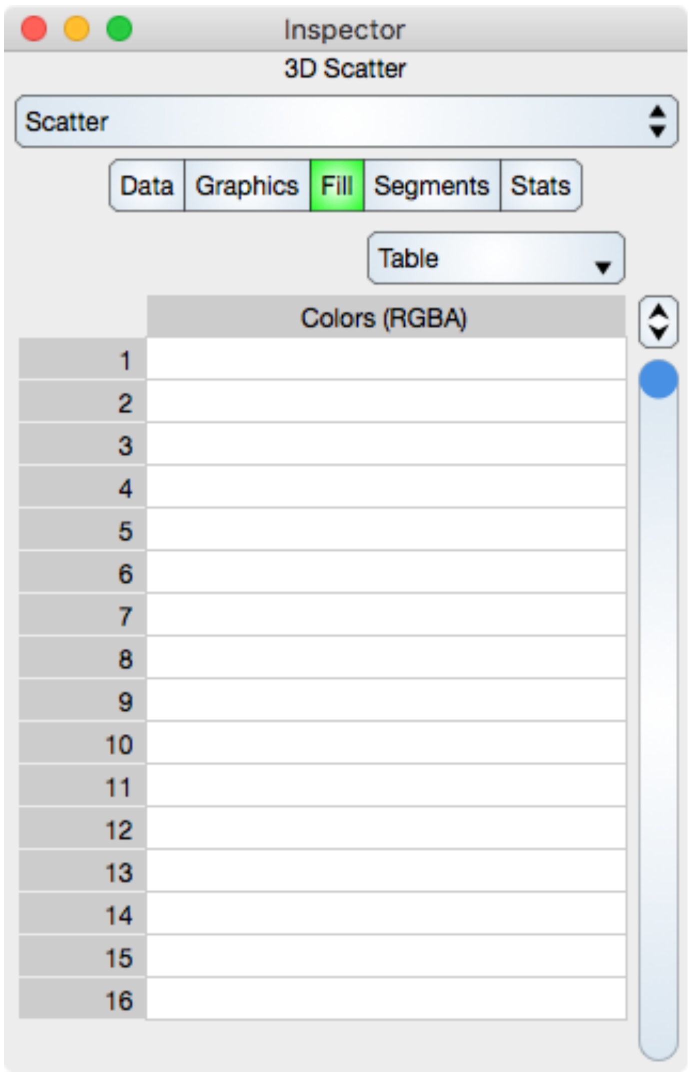
Fill attributes is a false color table of RGBA components, one for each point. These can be overridden by the Color Interpretation setting and that setting must be False Color Map in order for the elements of this table to take effect. Notice that this table is the same values presented in the Point Tags Colors editor.

Table

Table controls are described in the Tables section.

The rows represent a segment color values. While in Atomic mode, the cell represents a rgba value and while in component mode the cell represents either single r, g, b and a. Hence, in atomic mode there is one column while in component mode there are four columns.

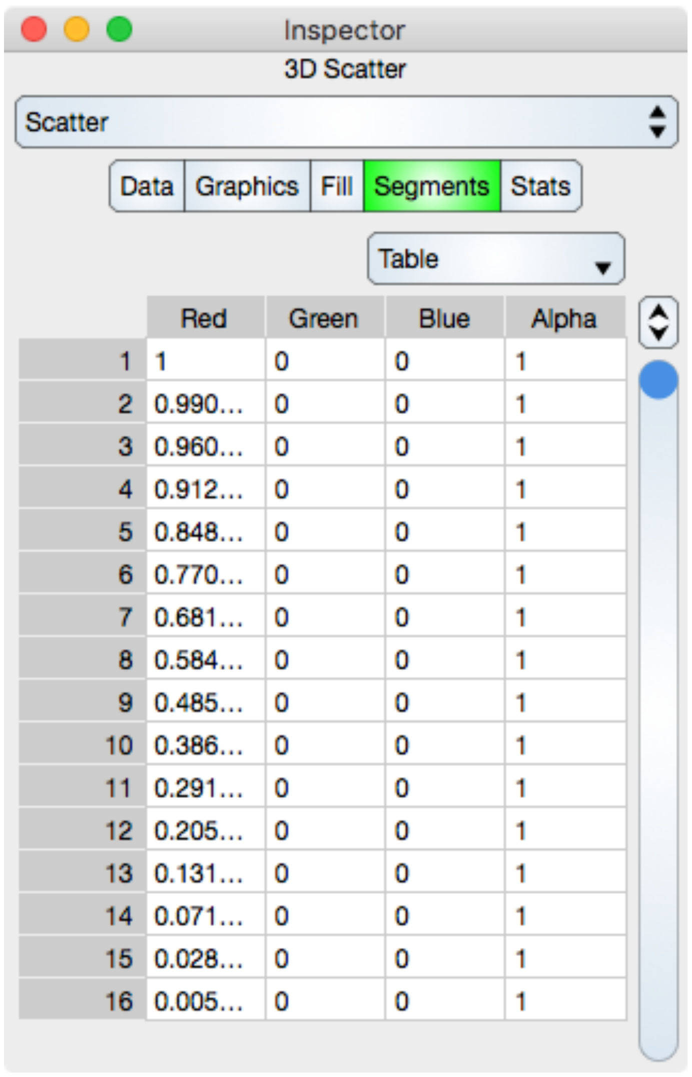
Note that if the Data sequence is changed then the Fill sequence will not change.



	Colors (RGBA)
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	

Segments Editor

The Segments Editor for the 3D Scatter graphic is shown below.



	Red	Green	Blue	Alpha
1	1	0	0	1
2	0.990…	0	0	1
3	0.960…	0	0	1
4	0.912…	0	0	1
5	0.848…	0	0	1
6	0.770…	0	0	1
7	0.681…	0	0	1
8	0.584…	0	0	1
9	0.485…	0	0	1
10	0.386…	0	0	1
11	0.291…	0	0	1
12	0.205…	0	0	1
13	0.131…	0	0	1
14	0.071…	0	0	1
15	0.028…	0	0	1
16	0.005…	0	0	1

Segments are the line segments interpolated between each 3D point. See Sequence Colors for additional information. A transparent color can be used to define an unconnected segment and then the data points can appear to be independent curves which represent separate data sets.

Table

Table controls are described in the Tables section.

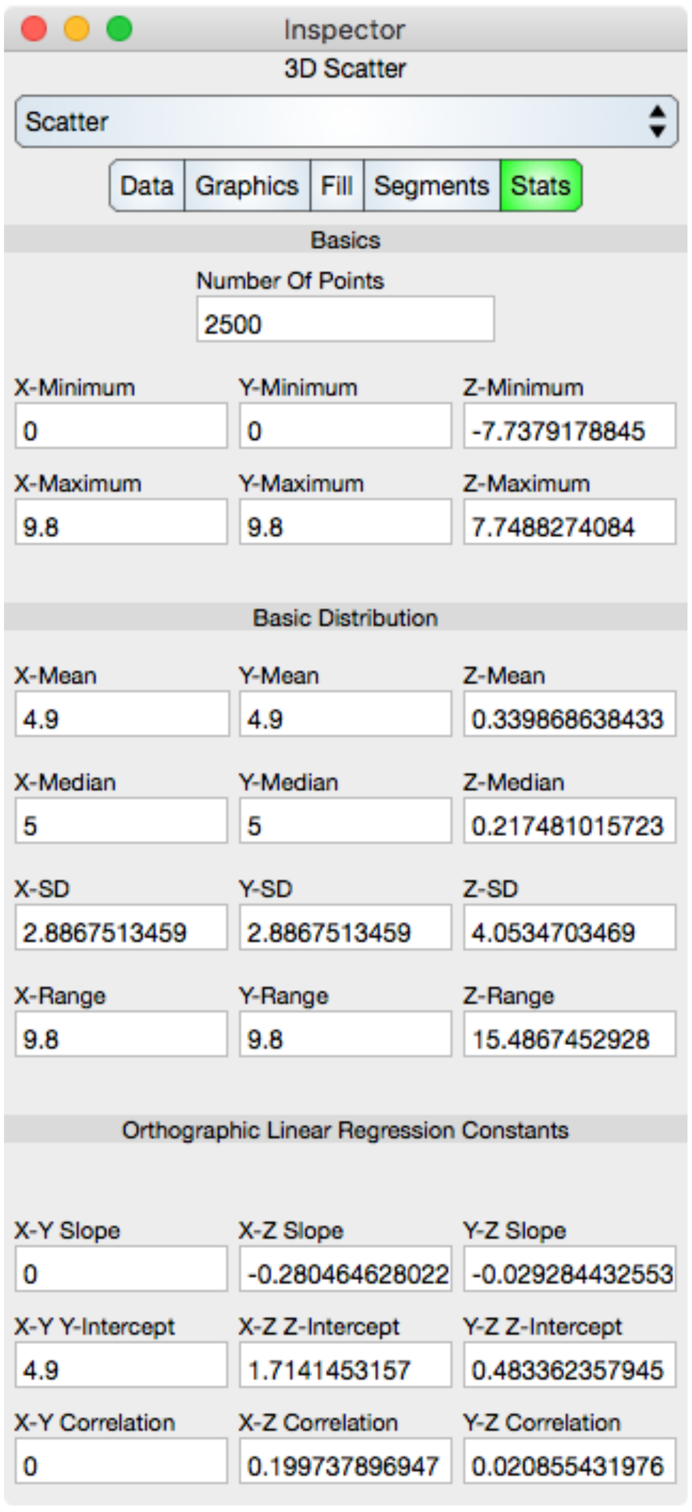
The rows represent a segment color values. While in Atomic mode, the cell represents a rgba value and while in component mode the cell represents either single r, g, b and a. Hence, in atomic mode there is one column while in component mode there are four columns.

Note that if the Data sequence is changed then the Segments sequence will not be change.

Stats Editor

The Stats Editor for the Scatter graphic is shown below. Note that each X, Y and Z statistic is an orthographic statistic, i.e.: That value that is the statistic on the respective graph planes.

Basics



Basics

Number Of Points		
2500		

X-Minimum	Y-Minimum	Z-Minimum
0	0	-7.7379178845
X-Maximum	**Y-Maximum**	**Z-Maximum**
9.8	9.8	7.7488274084

Basic Distribution

X-Mean	Y-Mean	Z-Mean
4.9	4.9	0.339868638433
X-Median	**Y-Median**	**Z-Median**
5	5	0.217481015723
X-SD	**Y-SD**	**Z-SD**
2.8867513459	2.8867513459	4.0534703469
X-Range	**Y-Range**	**Z-Range**
9.8	9.8	15.4867452928

Orthographic Linear Regression Constants

X-Y Slope	X-Z Slope	Y-Z Slope
0	-0.280464628022	-0.029284432553
X-Y Y-Intercept	**X-Z Z-Intercept**	**Y-Z Z-Intercept**
4.9	1.7141453157	0.483362357945
X-Y Correlation	**X-Z Correlation**	**Y-Z Correlation**
0	0.199737896947	0.020855431976

Number Of Points : Shows the number of points in the Scatter.

X-Minimum : x-minimum value of all the data points.

Y-Minimum : y-minimum value of all the data points.

Z-Minimum : z-minimum value of all the data points.

X-Maximum : x-maximum value of all the data points.

Y-Maximum : y-maximum value of all the data points.

Z-Maximum : z-maximum value of all the data points.

Basic Distribution

The basic distribution is shown separately for the x, y and z dimensions.

Mean : Shows the mean of the data.

Median : Shows the median of the data.

Standard Deviation : Shows the standard deviation of the data.

Range : Shows the range of the data (maximum - minimum).

Linear Regression Constants

Note that linear regression may also be referred to as line fit.

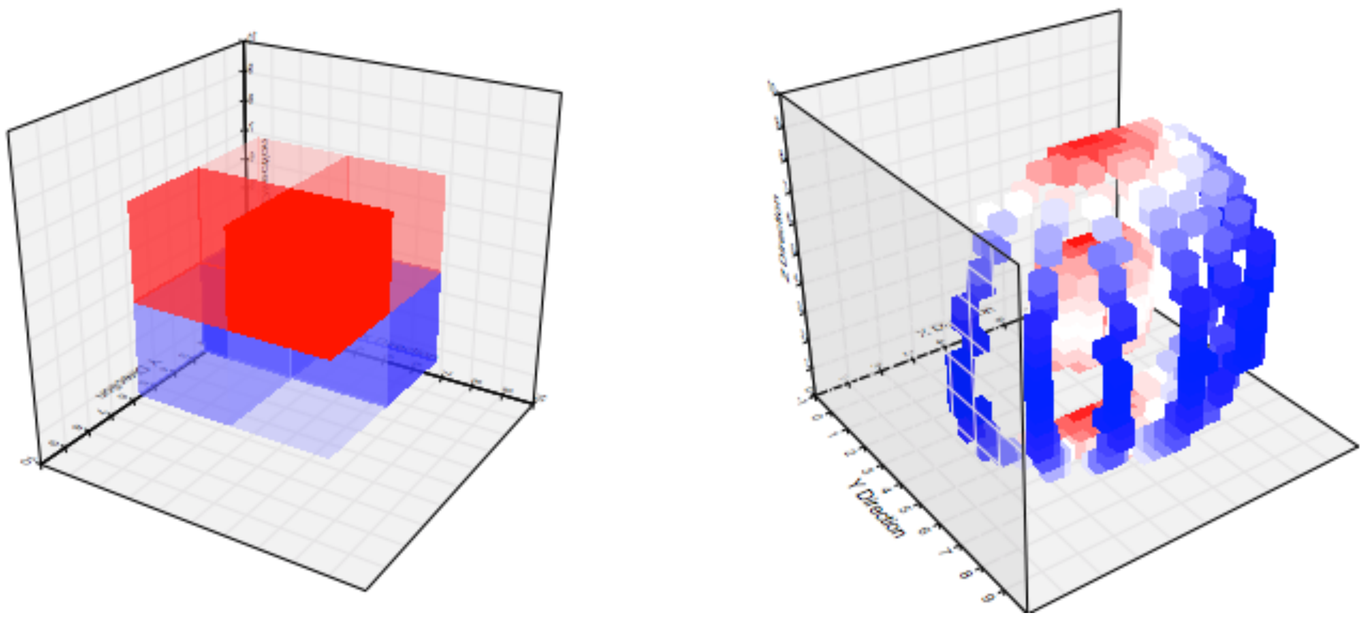
Slope : Shows the slope of the linear regression.

Y-Intercept : Shows the y-intercept of the linear regression.

Correlation : Shows the correlation constant of the linear regression.

Graph IDE ▶ 3D Data Graphics ▶ Volume

A Volume is a sequence of density values (from 0 to 1) that are mapped to colors via a color mapping with each density appearing as a cube on a regular 3D grid. The number of density values must be equal to the grid specifications (number of x, y, z cells). The figure below shows examples of 3D volume graphics.



Standard Operations

To create a volume bring forward the Graphic Selector, click the volume factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of a predefined volume graphic.

The normal way to modify the density of a volume is through the volume's Inspector Editor's data entry field. That field accepts a whitespace sequence of density values. The number of density values must be equal to the grid specifications (number of x, y, z cells).

For information on entering data points see Export And Import Data: Raw Data Points. The Volume conforms to the Color Map standards where the color map gradient is one of:

Color Mapping Type (Applies to Markers Only)

No Color Map	IGNORE: This has limited applicability. The color map is off. This makes most sense while just drawing the stroke so that only the trajectory appears.
False Color	IGNORE: This has limited applicability. The color is mapped from arbitrary independent values. Because there is no way to specify another independent value set this UI implementation maps from the z-values of the data. Programs can map from another basis.
Density-Value	The color is mapped from the density-value of the data points.
Distance	IGNORE: This has limited applicability. The color is mapped from a value related by the distance of the data point to the viewer. The distance is normalized so that the nearest and farthest points of data define the distance extremum.

Data Editor

The Data Editor for the Point Map is shown below.

Table

Apply : Once the grid values have been entered then click the Apply button to redefine the grid.

Table controls are described in the Tables section.

The rows represent density of a particular cell and must be a value between zero and one.

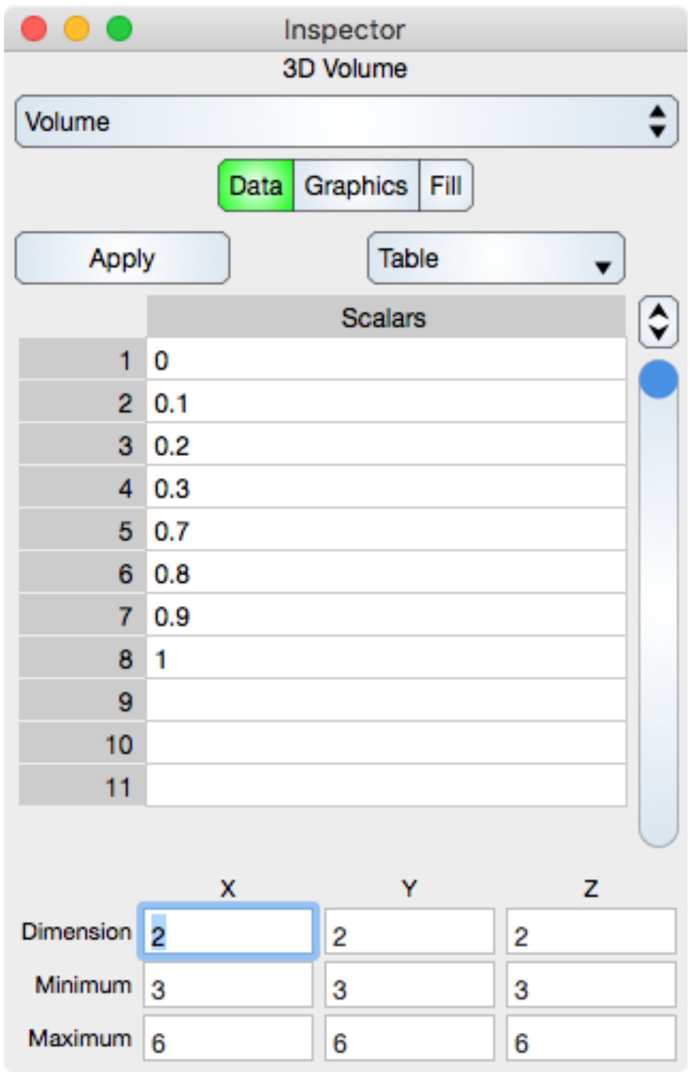
Grid Values

The data table does not specify grid parameters, it only specifies amplitude values. That is because a volume map is on a regular grid. A regular grid is defined by the nine values described below.

Dimension : The number of cells in the x, y and z dimension, aka: The dimensions of the matrix of values. The product of x, y and z dimensions should equal the number of rows in the table.

Minimum : The x, y and z minimum of the grid.

Maximum : The x, y and z maximum of the grid.



	Scalars
1	0
2	0.1
3	0.2
4	0.3
5	0.7
6	0.8
7	0.9
8	1
9	
10	
11	

	X	Y	Z
Dimension	2	2	2
Minimum	3	3	3
Maximum	6	6	6

Graphics Editor

The Graphics Editor for the Point Map is shown below.

Common Controls

Controls common to all graphics are described in the Graphics section.

Color Map Attributes

The color map is further described in the Color Map section.

Graph : Shows a graph of the color map.

Function : Select this to set the color map function. Typically it is set to linear.

Start Color : Defines the start Color of the color map.

End Color : Defines the end Color of the color map.

Number : Defines the number of samples in the color map.

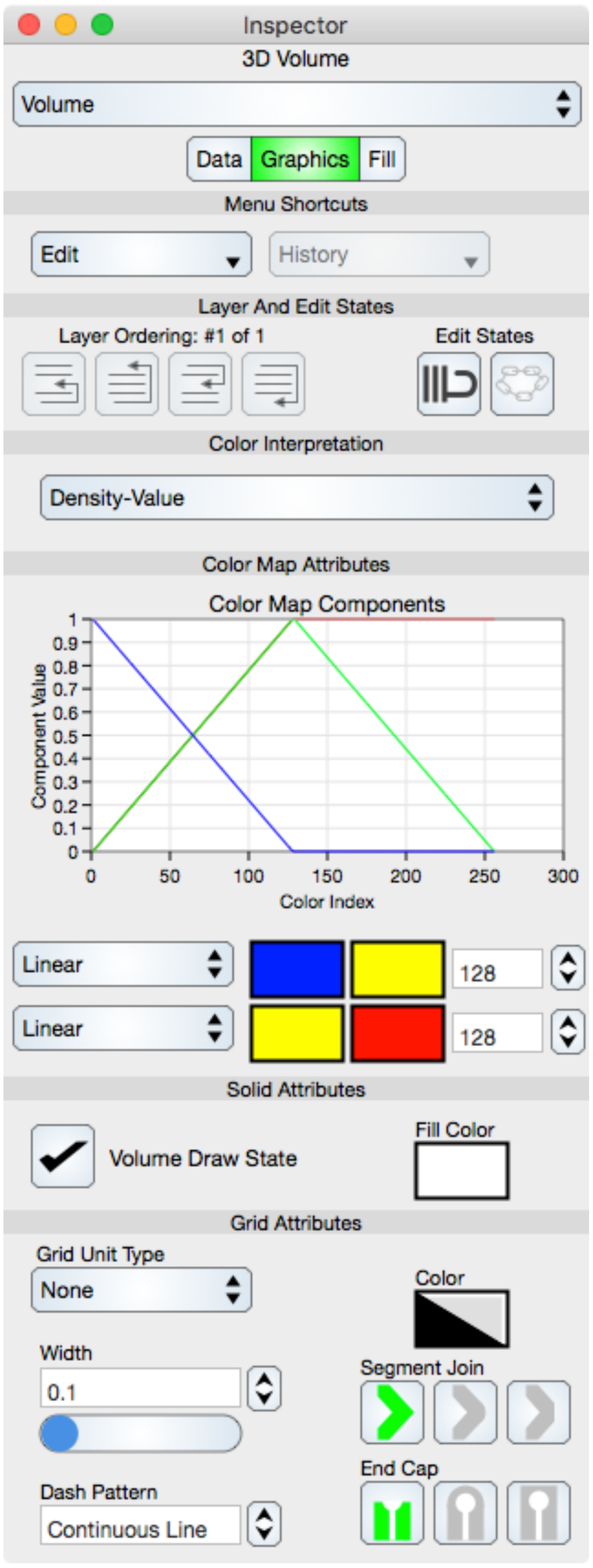
Solid Attributes

Volume Draw State : If checked then the volume is drawn.

Fill Color : Defines the fill Color for the volume. This is normally overridden by the Color Map.

Grid Attributes

Grid attributes are controlled by the normal stroke controls defined in the Graphics section.



Fill Editor

The Fill Editor for the 3D Volume graphic is shown below.

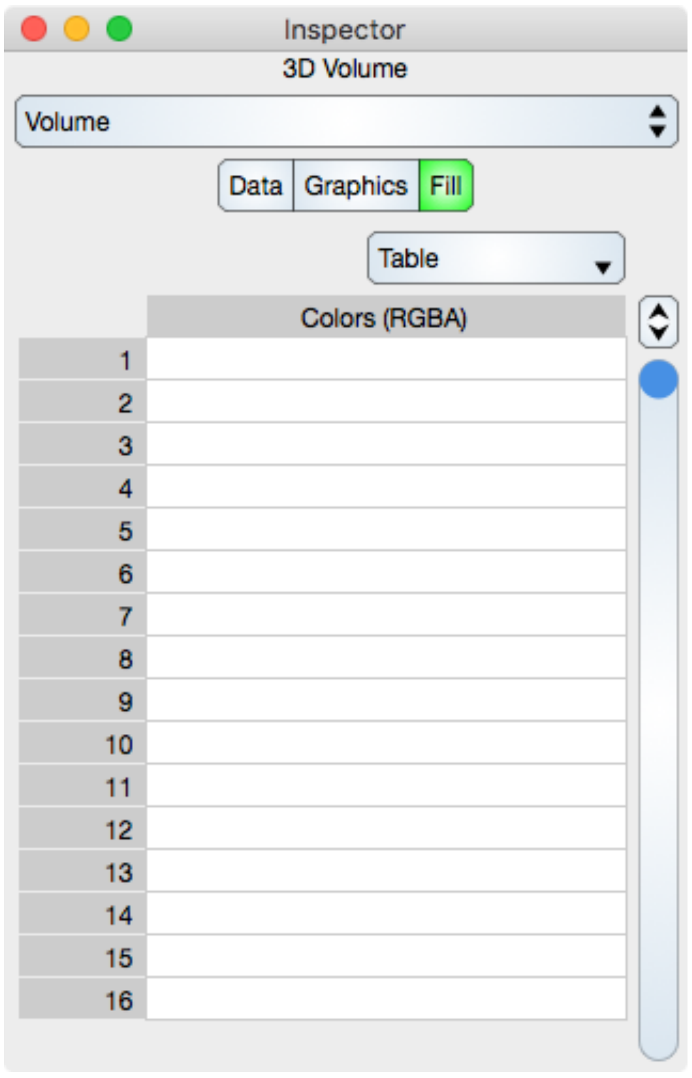
Fill attributes is a false color table of RGBA components, one for each cell. These can be overridden by the Color Interpretation setting and that setting must be False Color Map in order for the elements of this table to take effect.

Table

Table controls are described in the Tables section.

The rows represent a segment color values. While in Atomic mode, the cell represents a rgba value and while in component mode the cell represents either single r, g, b and a. Hence, in atomic mode there is one column while in component mode there are four columns.

Note that if the Data sequence is changed then the Fill sequence will not change.



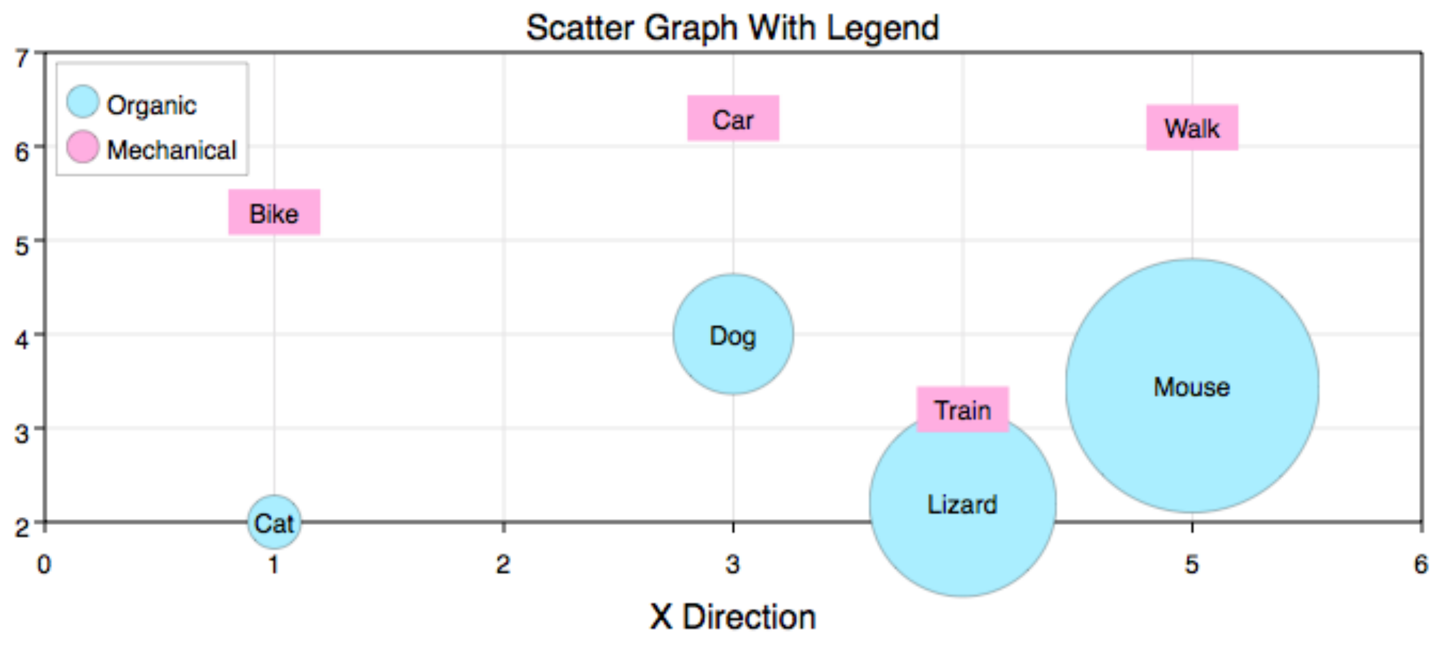
	Colors (RGBA)
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	

Graph IDE ► Composites

Composites are graphics that are combined in various ways to produce a particular added value. Composites are further explained in the following sections:

Section	Description
Legend	Describes how to make a legend for a graph.
Network	Describes how to make a network of otherwise disassociated graphics.
Trends	Describes how to place trend curves on a graph.

A legend associates a color with a description for each data graphic such as a curve. For a graph with curves on it, a legend shows consecutive rows of a circle filled with the curve color and a label description to the right of that circle. The following figure shows a graph with a legend in the upper left of the graph's frame.



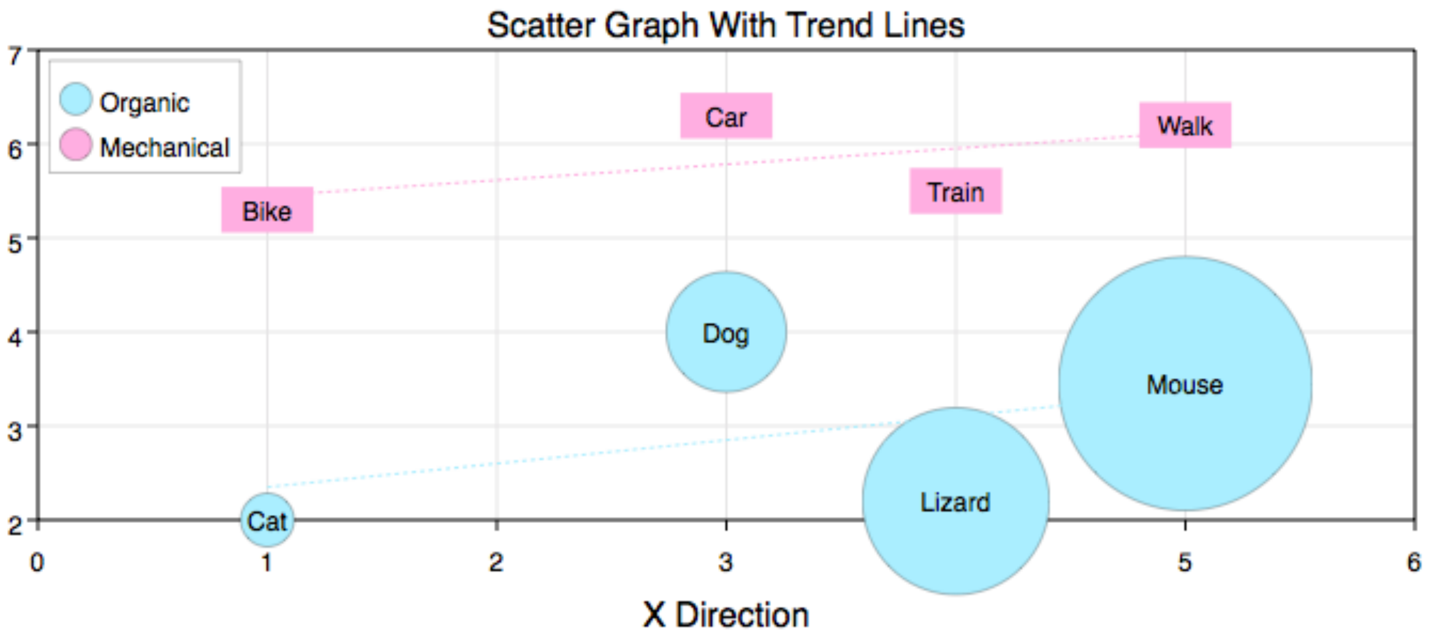
To make a legend follow these steps:

- Drag out a Graph. Double click within the graph frame to focus on the graph data layer.
- Add Data Graphics to the graph data layer. For each data graphic set the description in the Expert inspector. The description shows up in the legend label.
- Click the menu item Tools ► Make Legend. A legend will then be placed in the upper layer of the document (not in the graph's data layer). To get to the legend, double click to defocus the data layer and then select the legend graphic.
- The legend consists of a Rectangle background plus a Group graphic of circles and labels. You can work with those graphics or ungroup the legend group and work with the individual components of a legend. One thing you might want to do right away is to select the background and group and drag it to different location to un-obstruct any data that might be shown on the graph.

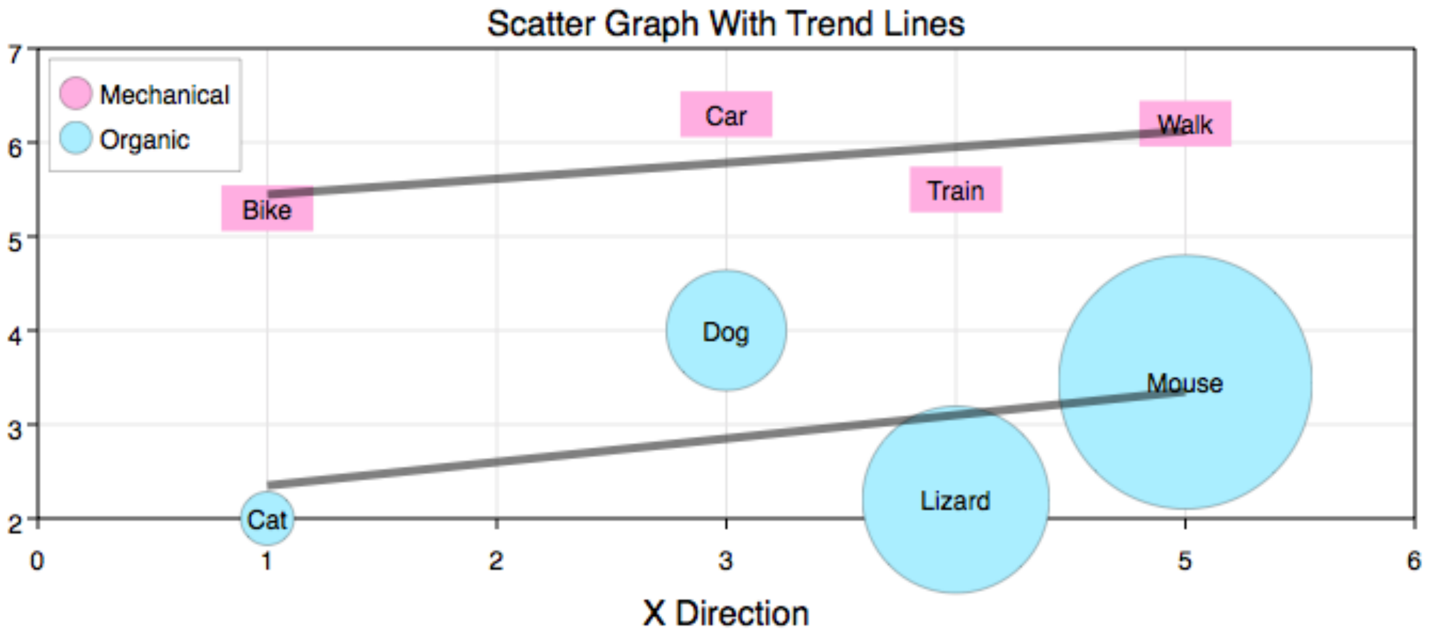
The Chart Tasks implements automatic legend generation and data is imported using a table. Consider that for more automated graph generation and then export the resulting graph to a Graph IDE document for fine tuning. See the Graph manual (accessible via the help menu) for further details.

Note that the same result can be attained by manually drawing Circle, Label and Rectangle graphics and that the Make Legend composite is simply a convenient way to make a legend. For graphs with one curve, it might be easier to simply make a legend yourself, but for a graph with many curves the automatic legend generation can save time.

A trend is a fit to 2D data points. The fit is a 2D linear regression which shows as a dashed curve by default. The following figure shows the Legend example with a trend curve for each scatter graphic.



Because the default trend graphics are not bold enough for this particular graph, they are altered in the following figure. Noticed how the legend entries are reversed ordered as well. All the alterations were made using Standard Editing techniques.



To make trends follow these steps:

- Drag out a Graph. Double click within the graph frame to focus on the graph's data layer.
- Add Data Graphics to the graph data layer.
- Click the menu item Tools ► Make Trends. Trend curves will be placed in the data layer of the graph.
- You can work with the trend curves as you would any graphic.

Trend curves instantiate as Function graphics. They are computed by least squares and are hence a line in a rectilinear coordinate system. However, that line maps to a curve in non-linear coordinates and hence the word curve instead of line when describing a trend.

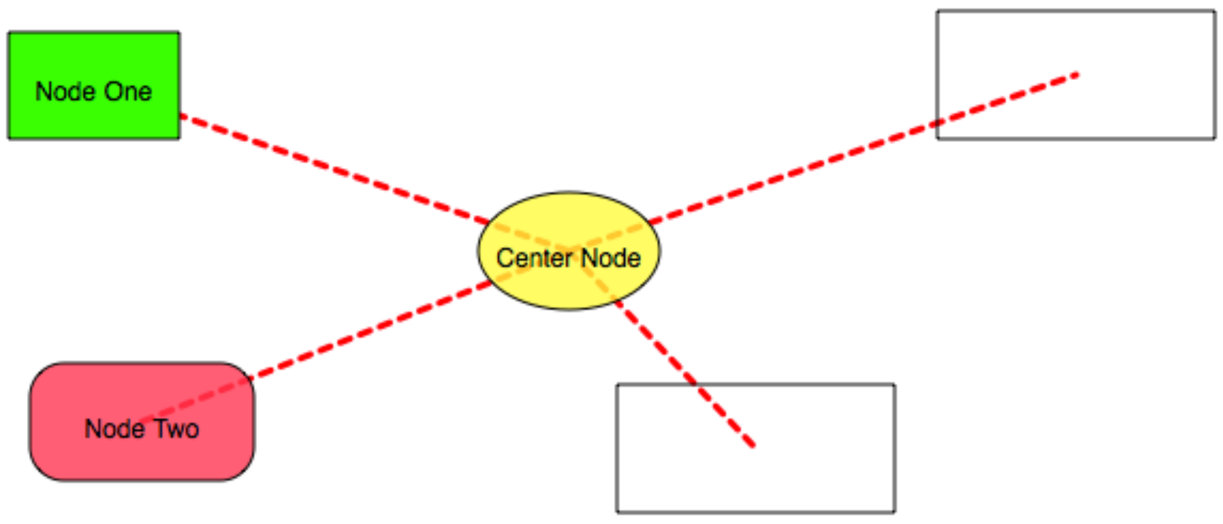
Graph IDE ► Composites ► Network

A network is a connected sequence of graphics. When graphics are added to a Layer while a network is active then a network connection graphic is drawn between all graphics of that layer. Conceptually the idea is easy but learning how to turn on the network for the first time is not obvious. Also not obvious is the fact that error bars are a network type. The generalization of a network to error bars is very powerful, but that power comes at the cost of being required to understand the more abstract model of a network as defined in Graph IDE. Because of that, it might be useful to watch the following movie:

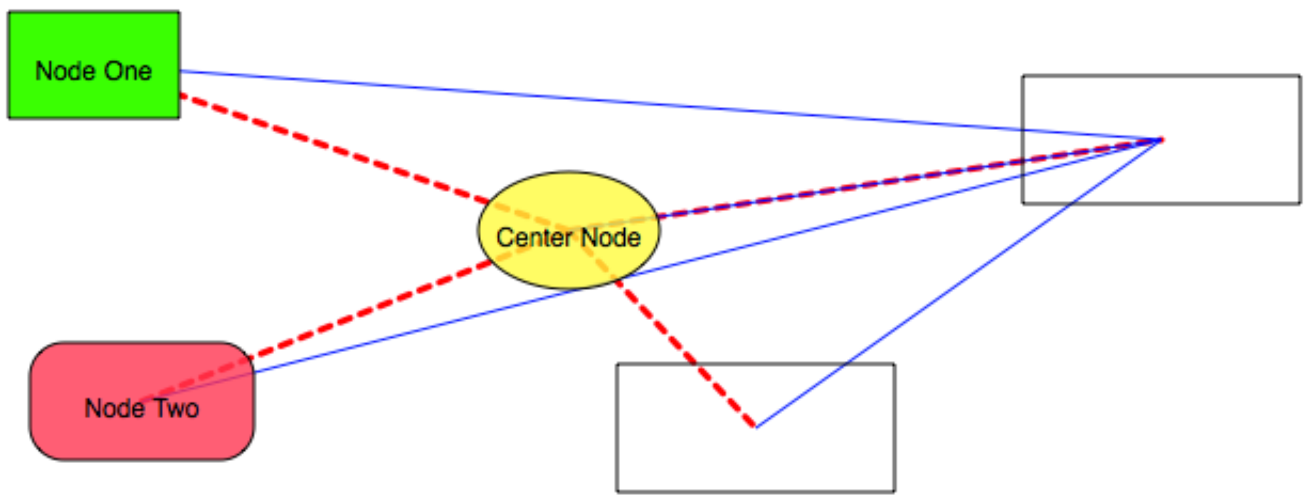
Loading

(Requires network connection www.vvidget.org)
For the full size movie click this link: Error Bars Movie.

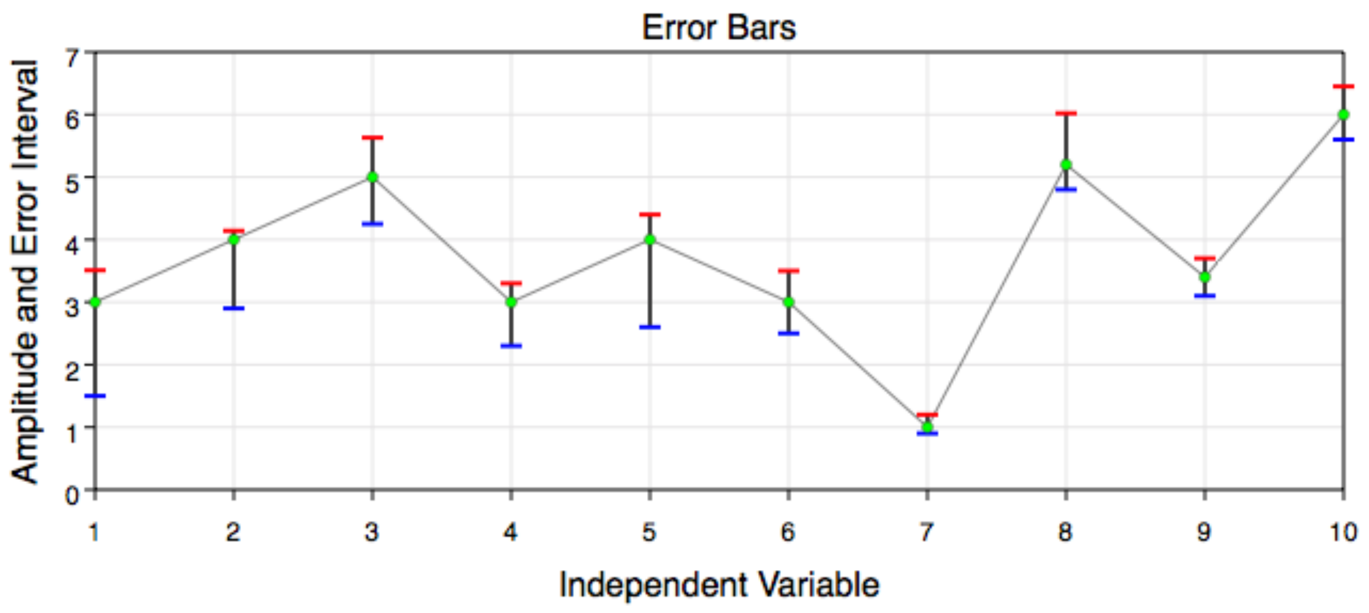
Configuring a network is explained in the Inspector Editors ► Network section. The figure below shows a graph network where the yellow oval is a network hub.



The following figure is the same as above except the upper-right rectangle is also a network hub with its own connection graphic which is a thin blue line. As you can see, introducing another hub produces a graph cycle.



The figure below shows an error bar network where the center function graphic (curve) is connected to its upper and lower neighboring graphic's points.



The easiest way to make an error bar network is to use the Error Bars assistant in the Chart Tasks and then to export to Graph IDE. Then you can dissect that output to discover how error bars work. Once you understand the nature of error bars then you can make simple or intricate error bar figures. Note that there can be multiple error bar hub nodes, the connection graphic can have caps, be vertical or horizontal or be an arbitrary graphic such as an image, oval or rectangle. There are many options and the Error Bars assistant instantiates only one.

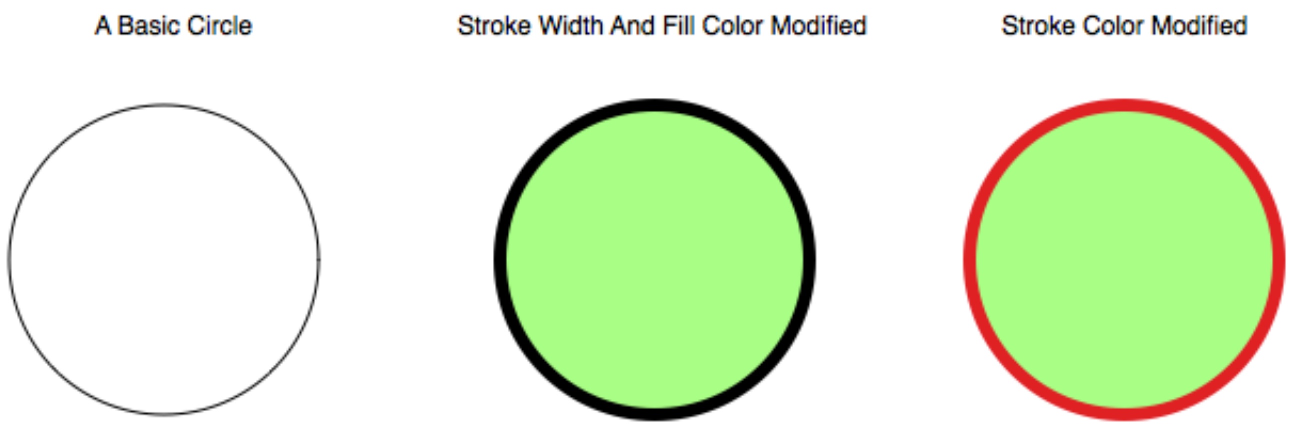
The following table recaps some of the terminology and ideas of a network composite graphic.

Term	Description
Connection	A connection is a relationship between two nodes. The connection is displayed by a connection graphic and is synonymous with that graphic.
Cycle	When there are two or more hubs then the connection graphics can form a polygon which is called a cycle.
Error Bars	Error bars relate one graphic to another and hence are one type of connection. The error bars connect two or three nodes by those node's point values. Because error bars are point-oriented they only apply to Data Graphics.
Graph	A graph, in the context of a network, is a network and should not be confused with chart-type Graphs which are completely different. Sometimes a graph is also called an "object graph".
Hub	A hub is that node for which the connection graphic is defined. Each node can have its own connection graphic and hence any node can also be a hub although usually only one node is a hub.
Neighbor	A neighbor is a node that is next to another node. Unfortunately there are two meanings to the phrase "next to". For a graph, "next to" means all nodes in a layer. For error bars, "next to" means the nodes immediately before and after the sequence index of the node being considered. If the layer has nodes {n_1 ... n_{i-1}, n_i, n_{i+1} ... n_N} then the neighbors of n_i are n_{i-1} and n_{i+1} for error bars but are the entire sequence of nodes for graphs.
Network	A network is a set of graphics on a layer that is related by a connection. In the context of a network the graphics are called nodes.
Node	A node is any graphic in the Layer.

Satellite	A satellite is any node that is not a hub for any given connection graphic.
Subnetwork	A subnetwork is a set of graphics that are connected where that set is a subset of a larger set of graphics. For example, a Group graphic connects only graphics within that group. A Layer does not connect to other layers so that subnetworks can be made to display by using different overlays. The Error Bars network type only connect to the two graphics before and after the hub graphic. In addition a node can be excluded from a network within a layer and the exclusion can then form a subnetwork.

Graph IDE ► Draw Attributes

Draw attributes augment the graphical representation of graphics such as all the Basic Graphics. The attributes include the border and interior colors and the border width in the following figure:

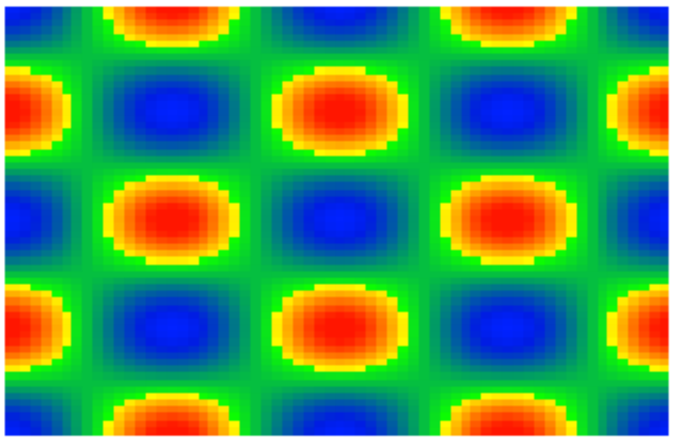


However, they do not include the orientation, aspect or other geometric properties of the graphic. The table below itemizes the draw attribute classes.

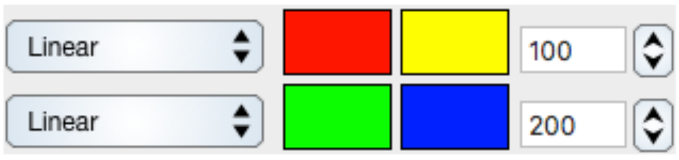
Section	Description
Color Map	A predefined two-domain mapping function which maps four color control points into a table of color values.
Extended	More complex draw attributes, including gradient and shadow effects.
Point Tags	Point-wise draw attributes. These are mainly for data-oriented graphics.
Sequence Colors	Inter-Point draw color. These are mainly for data-oriented graphics.
Simple	Simple draw attributes, including color and width parameters.

Color Map Draw Attributes define a gradient of colors related to the data values of a graphic. Color maps can add another dimension to the data representation or they can augment the representation of an existing dimension. The figure below is an example of a color map for the Point Map data graphic. The color gradient represents a hypothetical z-dimension in an otherwise x and y dimension coordinate representation.

Point Fill graphic with cell colors defined by a color map



Controls that define the color map



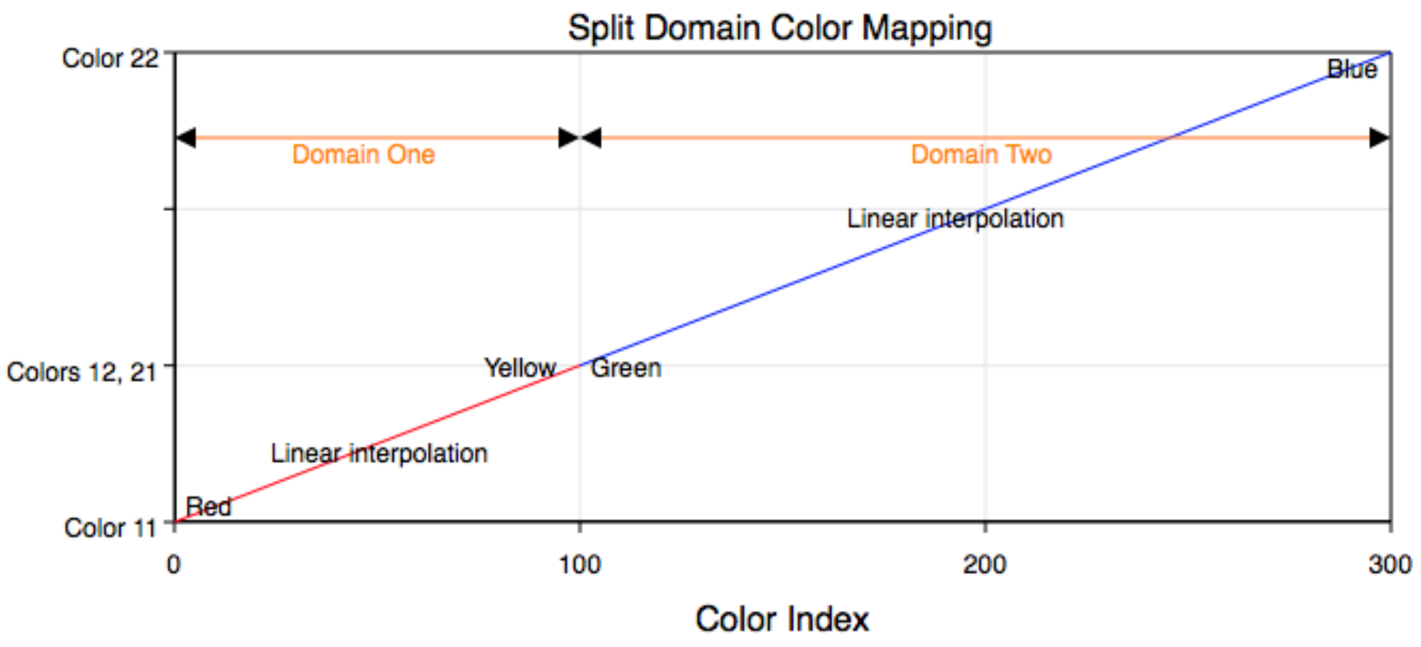
Map runs from red, yellow, green then to blue colors

All data graphics can potentially support at least one if not more color mappings. The data graphics that presently support color mappings are the Point Map, 3D Point Map, 3D Scatter graphics. The present color mapping is a gradient mapping in that the gradient does not necessarily have a one to one relationship with the data.

Attributes

The interpretation of a color map is specific to each data graphic. However, in the current implementation, a color map defining attributes (control parameters) are listed here:

The color mapping is a contiguous split-domain mapping. The following figure shows that mapping function for the parameters shown in the example point fill graphic above.



Domain one has 100 color sections that map the minimum data value to red and the data value 1/3 the way between minimum and maximum to yellow. It then maps the next data value to green and the maximum data value to blue. Between those control values it linearly interpolates the color values. The following defines the parameters for one of those color mapping domains:

Number Of Colors: Defines the number of colors in the domain.

First Color: The color of the domain's minimum data value.

Second Color: The color of the domain's maximum data value.

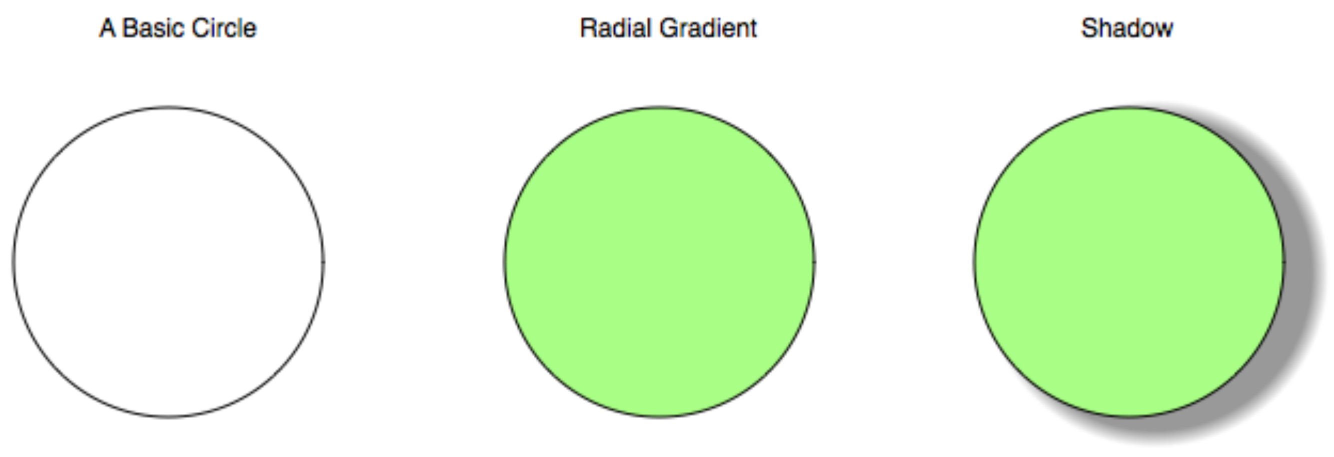
Interpolation: The curve profile of the color interpolation between the first and seconds colors. This is one of constant linear, quadratic or inverse quadratic.

By defining the first domain as a quadratic interpolation and the second domain as an inverse quadratic interpolation the resulting combined mapping profile is smooth. If either domain has interpolation of constant then there will only be a single color in the data range of that domain. Implicit in the definition is the fact that the number of colors in each domain are proportionally related to the data value interval of that domain. For example, if the first domain number of colors is zero then all colors are mapped by domain two. If the number of colors of each domain is the same then each domain is exactly half of the data value range, i.e.: from {minimum to (maximum + minimum)/2} and then from {(maximum + minimum)/2 to maximum}.

By defining domain-1-maximum value equal to domain-2-minimum value the mapping profile will be continuous. Otherwise it will be disjoint. A disjoint profile is desirable if the discontinuity is relevant to the associated data value.

Graph IDE ► Draw Attributes ► Extended Draw Attributes

Extended Draw Attributes are gradient and shadow related. The figure below is an example of some extended attributes.



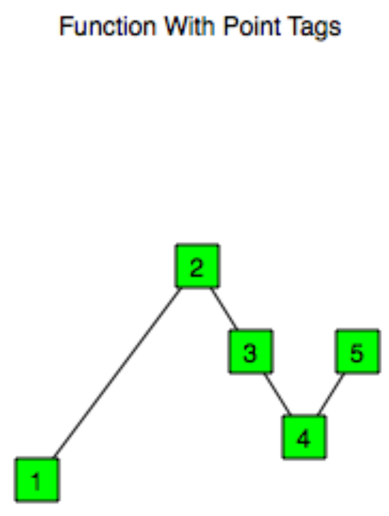
Attributes

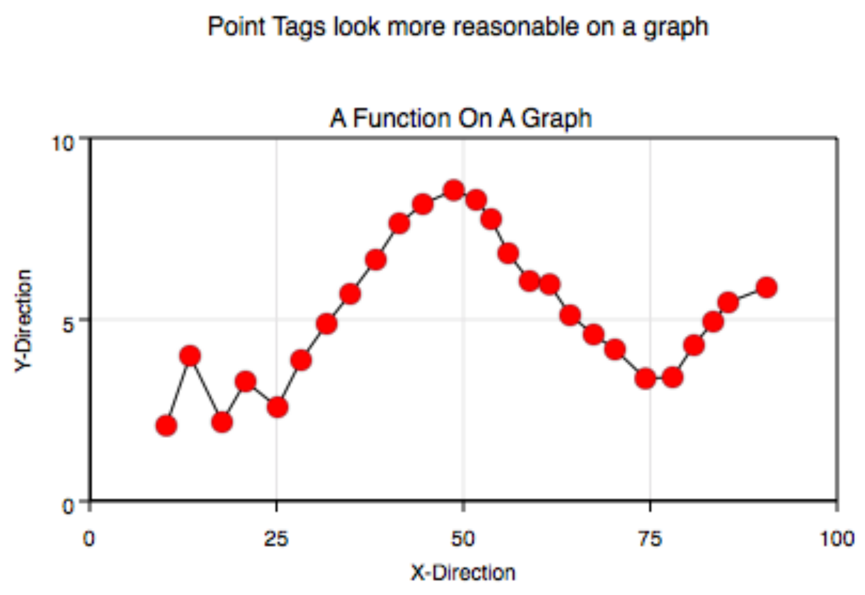
Gradient: The color grading of the fill. Gradients progress from one color to another.

Shadow: The diffuse and offset single color duplicate of a graphic. It is drawn behind a graphic to give the appearance of a shadow.

Graph IDE ► Draw Attributes ► Point Tags

Point Tags Draw Attributes are point related and normally drawn over each point in a graphic. The figure below is an example of some point tags.





Attributes

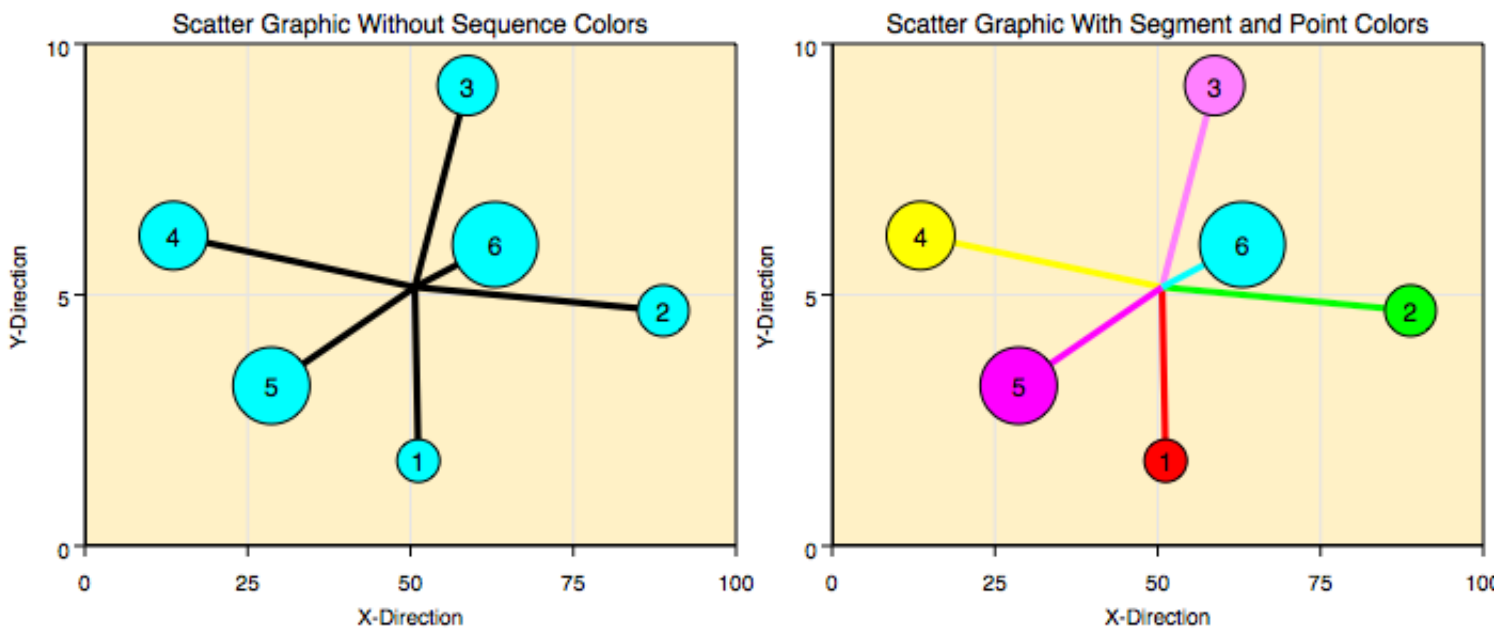
Marker: Another, usually smaller, graphic drawn at each point of the graphic which has the point tag as an attribute.

Label: A predefined piece of text drawn at each point.

For additional information see the Point Tags Inspector section.

Graph IDE ► Draw Attributes ► Sequence Colors

Colors that are an element of a sequence are called Sequence Colors. The figure below shows a scatter graphic with and without sequence colors.



Sequence Colors are implemented in many ways as described in the following table.

Component	Description
Segments	The Scatter, Function, Trajectory and 3D Scatter graphics are defined by a sequence of points that can be connected by a line or spline segment. Each of those segments can be given a color.
Markers	The Scatter, Function, Trajectory and 3D Scatter graphics are defined by a sequence of points and each of those points can be delineated by a marker (another graphic). The marker can be given a unique fill color for each point in the sequence. See the Point Tags inspector editor and its color tab for additional information.
False Color Maps	The 3D Point Map graphic is defined by a sequence of z values mapped onto a regular grid. That grid defines connected cells (surface) where the cells are ordered and form a sequence. Each cell can take on a different color as defined by a false color map.

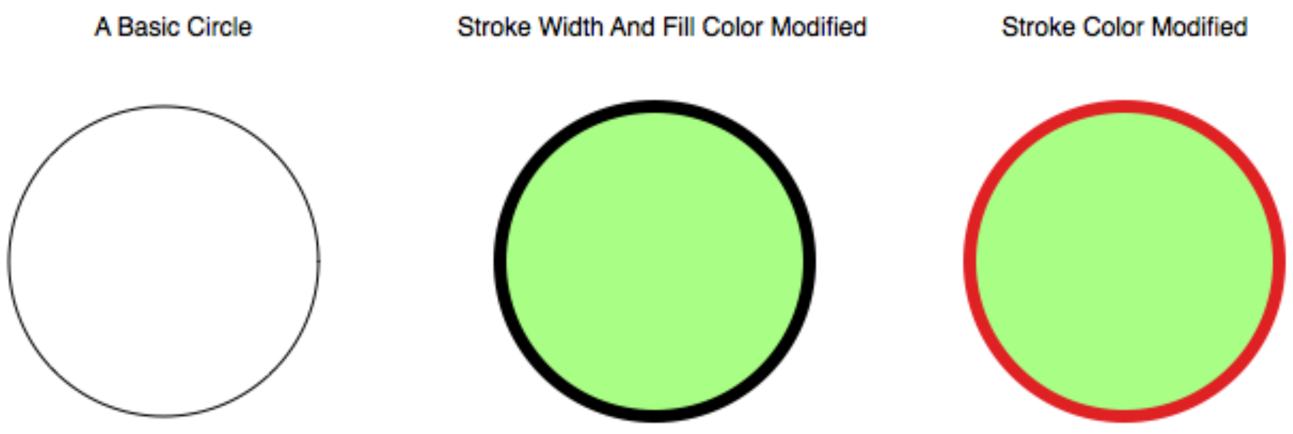
Sequence colors are entered using a Table where each cell in the table shows a color value in rgba numeric representation which are four numbers between zero and one representing the red green blue and alpha values of the color. The alpha is the level of opacity (1 being opaque and 0 being transparent).

Graph IDE ► Draw Attributes ► Simple Draw Attributes

Simple Draw Attributes are stroke and fill related, such as color and width. The figure below shows a very simple example of these attributes.

Fill And Stroke

Fill And stroke graphical attributes are the simplest draw attributes. Examples of which are shown in the figure below.



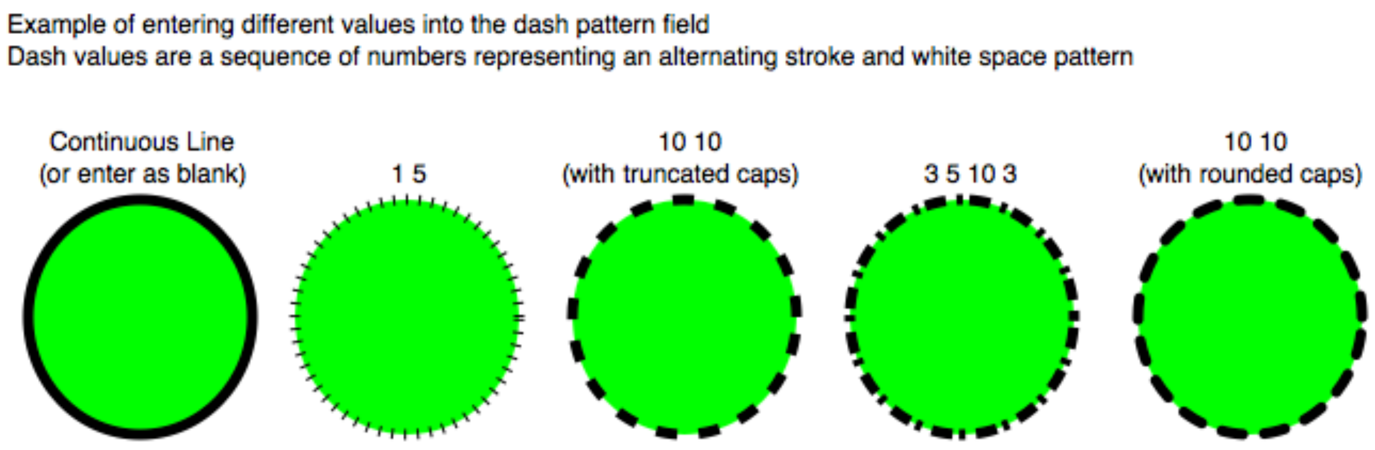
Note that for some graphics, such as a Scatter graphic, the stroke is not actually the boundary of the graphic. In that case it is the vectors drawn to each point from a predefined origin. Likewise, for a Function the stroke is the curve (which does not define a boundary of an area, but rather the top of an area).

Definitions

- Fill: The interior of a graphic.
- Fill Color: The color of the fill.
- Stroke: The boundary of a graphic. The stroke is drawn after the fill so the fill never obscures it.
- Stroke Color: The color of the boundary.

Dash Pattern

The dash pattern specifies the way a single contiguous stroke is drawn in regular pieces. Examples are shown below.


Example of entering different values into the dash pattern field
Dash values are a sequence of numbers representing an alternating stroke and white space pattern
Continuous Line (or enter as blank)
1 5
10 10 (with truncated caps)
3 5 10 3
10 10 (with rounded caps)

Definitions

- Continuous: The dash is one solid stroke, i.e.: there is no dash.
- s_i b_i: The dash is specified as a sequence of on and off segments define in stroke units such as: b_1 s_1 b_2 s_2 ... b_n s_n where b_i is the length of the blank segment and s_i is the length of the stroked (drawn) segment.
- Join: Defines the way segments are joined and is one of non-mitered, rounded or mitered.
- Cap: Defines the way one segment is terminated and is one of truncated, rounded or extended.

Graph IDE ► Advanced Graphics

The following is a brief list and definition of advanced graphics:

Section	Description
Adapter	An adapter is a graphic that supports native view drawing.
DB Curve	An example graphic that is separate from Graph IDE and is loaded as a Graphic Bundle.
Dynamic	The dynamic graphic is actually an entrance point the Graphic Bundle loader.

Graph IDE ► Advanced Graphics ► Adapter

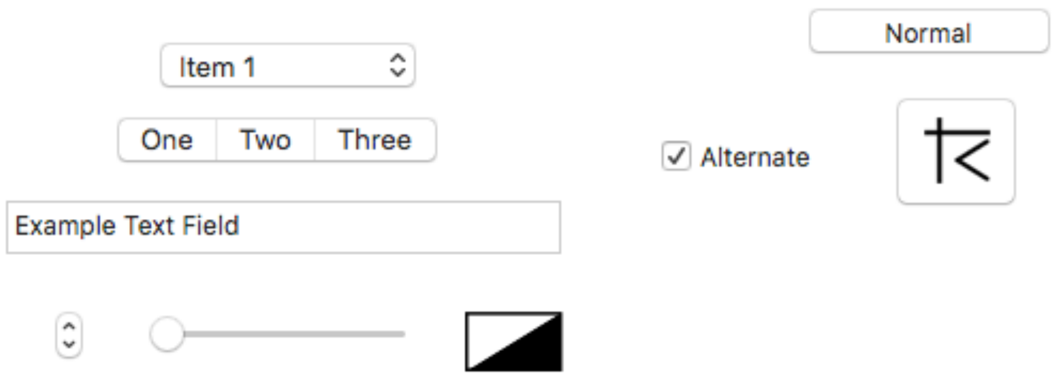
An adapter graphic supports adaptation of native system views into the Graph IDE document. You should only use them if you are programming in the Cocoa system. The entire Graph IDE user interface is programmed with a combination of Adapters and other graphics available in Graph IDE.

Please note the following fact: Only use adapters if you are an experienced developer and have made ample use of the system view software development kit (a.k.a.: Cocoa or Cocoa Touch).

To use an adapter, drag it out like any other graphic. Its base representation is a rectangle. Enter the new representation in the View Class Name field and click the Return key to enter the class. The runtime is sampled for that class and an instance of the view class is inserted over the adapter bounds. Then program the adapter according to the Programming section and other programming implementations details. If you are familiar with programming NSView (or UIView) then you can load in subclass implementations via a Plugin and utilize those views as well as all of the system views.

With the proper programming abilities, Graph IDE can be a simulator for cross platform GUI and IDE tool. The details are beyond the scope of this manual.

The following figure shows some pre-programmed adapters that are available on a Graph IDE palette.



Some standard operations are itemized below.

- To create an adapter bring forward the Graphic Selector, click the adapter factory cell and then mouse down on a Graphic View and drag the cursor to another point. The initial and final cursor locations define the boundary of a predefined adapter graphic.
- Once an adapter is created, the usual way to modify its values is through the Program inspector editor.
- To program an adapter see the Programming section.

Inspector Editor

The Inspector Editor for the adapter is shown below.

Menu Shortcuts

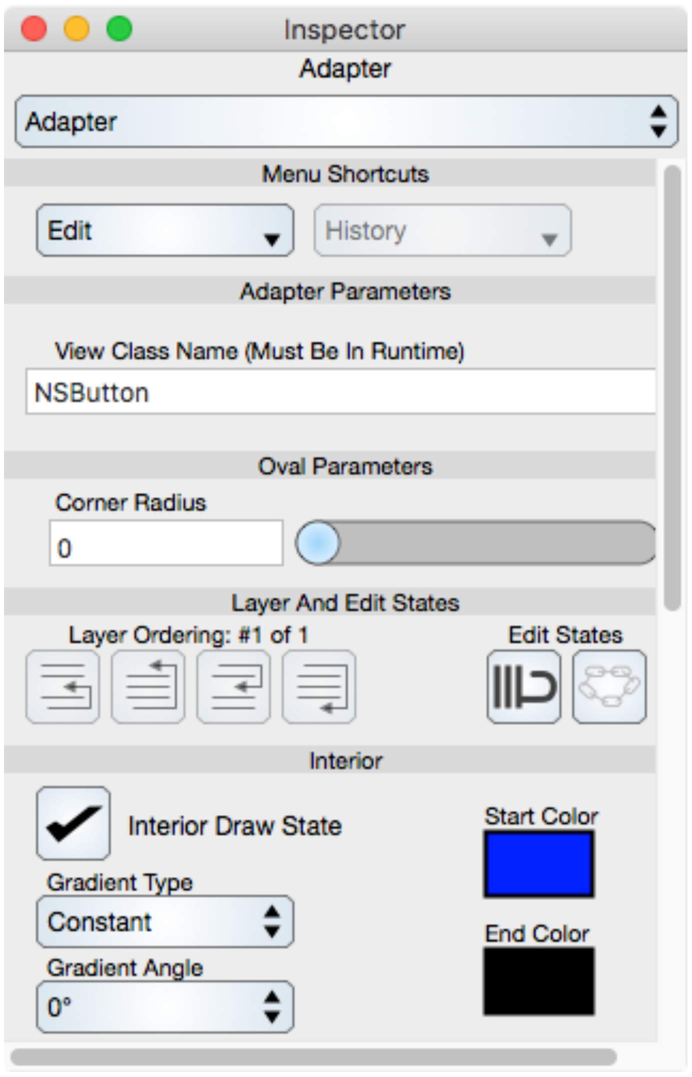
Menu shortcuts are common to all graphics and are described in the Graphics section.

Adapter Parameters

View Class Name : The name of the view class for the adapter. If you wish the control to be multiplatform then this class needs to be a cover class or some type of surrogate or alias for a platform specific class.

Other Controls

The backing for an adapter is a rectangle and the other controls that are on the Adapter's editor are those of a Rectangle.



Adapter Parameters

View Class Name (Must Be In Runtime): NSButton

Oval Parameters

Corner Radius: 0

Interior

Interior Draw State: ✓

Gradient Type: Constant

Gradient Angle: 0°

Start Color: 🟦

End Color: ⬛

The use of an adapter is described in the SineTable section.

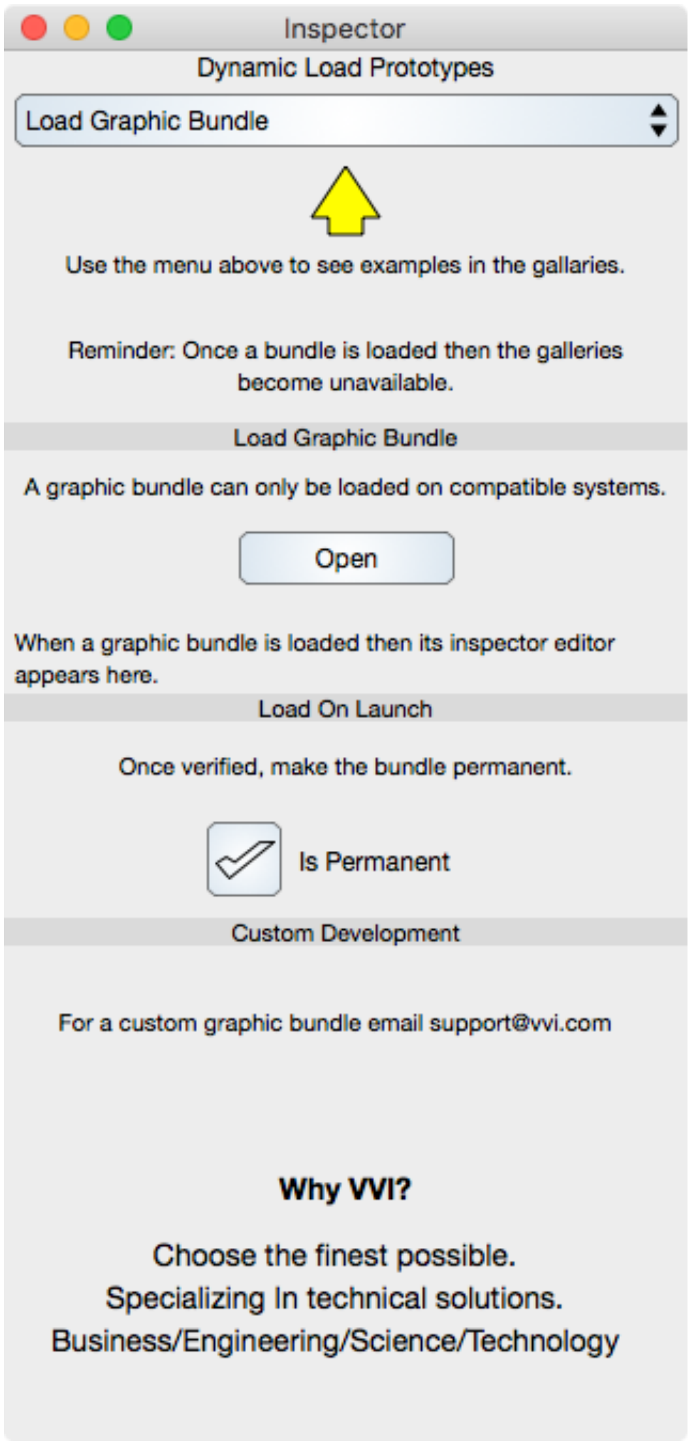
Graph IDE ► Advanced Graphics ► Dynamic

A dynamic graphic helps load a graphic bundle. To use the dynamic graphic bring forward the Graphic Selector, select the dynamic graphic factory cell and then use the resulting inspector editor to open up the graphic bundle. The dynamic graphic is a factory-only graphic so can not be instantiated.

Once a graphic bundle is loaded then it replaces the dynamic graphic in the Graphic Selector. An example is shown in the Database Curve section.

Load

The Load Inspector Editor is used to load a Graphic Bundle which defines resources for a, yet unknown, graphic. It is shown below.



Use the menu above to see examples in the gallaries.

Reminder: Once a bundle is loaded then the galleries become unavailable.

A graphic bundle can only be loaded on compatible systems.

When a graphic bundle is loaded then its inspector editor appears here.

Once verified, make the bundle permanent.

For a custom graphic bundle email support@vvi.com

Why VVI?

Choose the finest possible.
Specializing In technical solutions.
Business/Engineering/Science/Technology

Load Graphic Bundle

Open : Brings forward the open sheet to open a graphic bundle (Mac only).

Load On Launch

Is Permanent : When selected, the graphic bundle is loaded into the Graphic Selector when the application is launched. Otherwise, it is loaded only when the graphic bundle is opened.

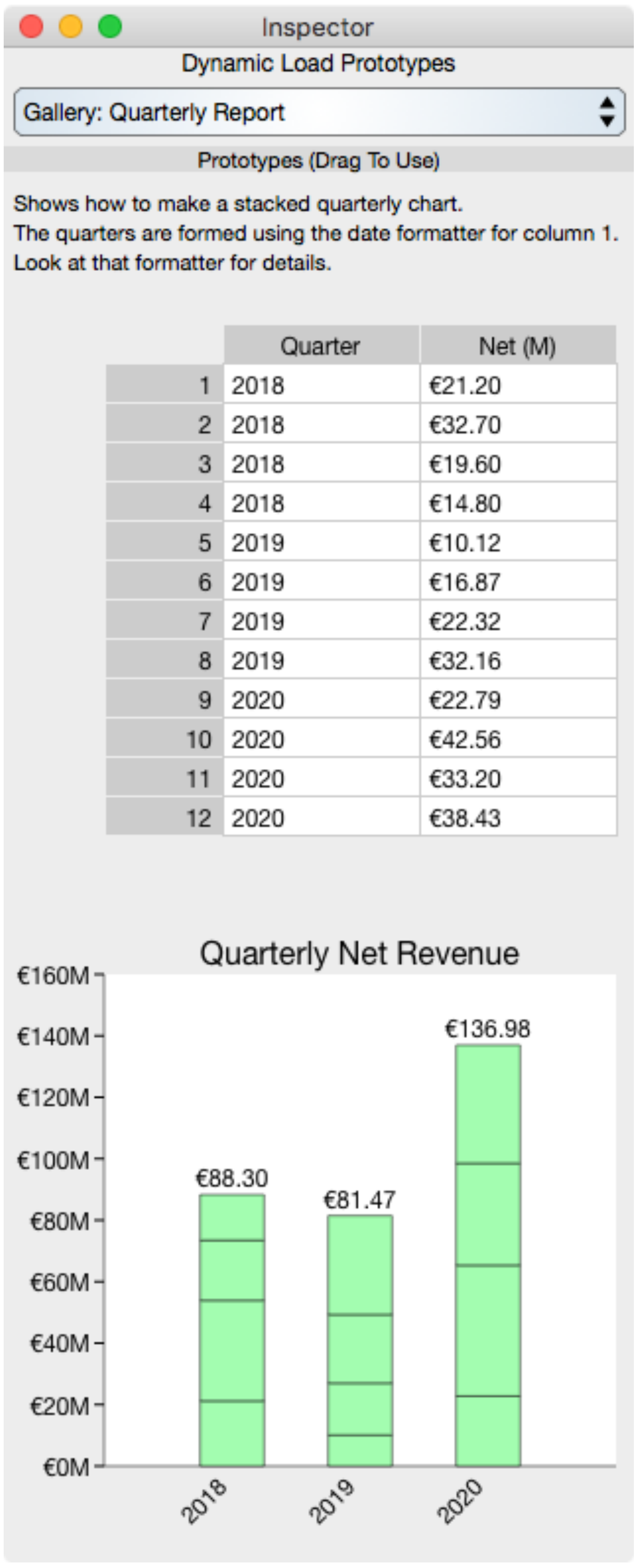
Custom Development

Developing a graphic bundle is fairly easy because of the way the code is architected. However, it is very intricate and requires specialized knowledge of the underlying programming concepts.

Contrast this to the Custom Application and Programming sections which do not require specialized knowledge and can be used in the field.

Gallery

The Gallery inspector editor shows some example graphics which can be utilized until a graphic bundle is loaded.



Inspector

Dynamic Load Prototypes

Gallery: Quarterly Report

Prototypes (Drag To Use)

Shows how to make a stacked quarterly chart.
The quarters are formed using the date formatter for column 1.
Look at that formatter for details.

	Quarter	Net (M)
1	2018	€21.20
2	2018	€32.70
3	2018	€19.60
4	2018	€14.80
5	2019	€10.12
6	2019	€16.87
7	2019	€22.32
8	2019	€32.16
9	2020	€22.79
10	2020	€42.56
11	2020	€33.20
12	2020	€38.43

The Gallery shows some interesting examples. In this case a Spreadsheet that has a Column Chart as an associated graphic. Both graphics are Grouped so can be dragged to a document as a single cohesive unit. Once placed on a document then the group is set to ungroup. The connection between the spreadsheet and pie chart is maintained so changing data in one will change the data in the other.

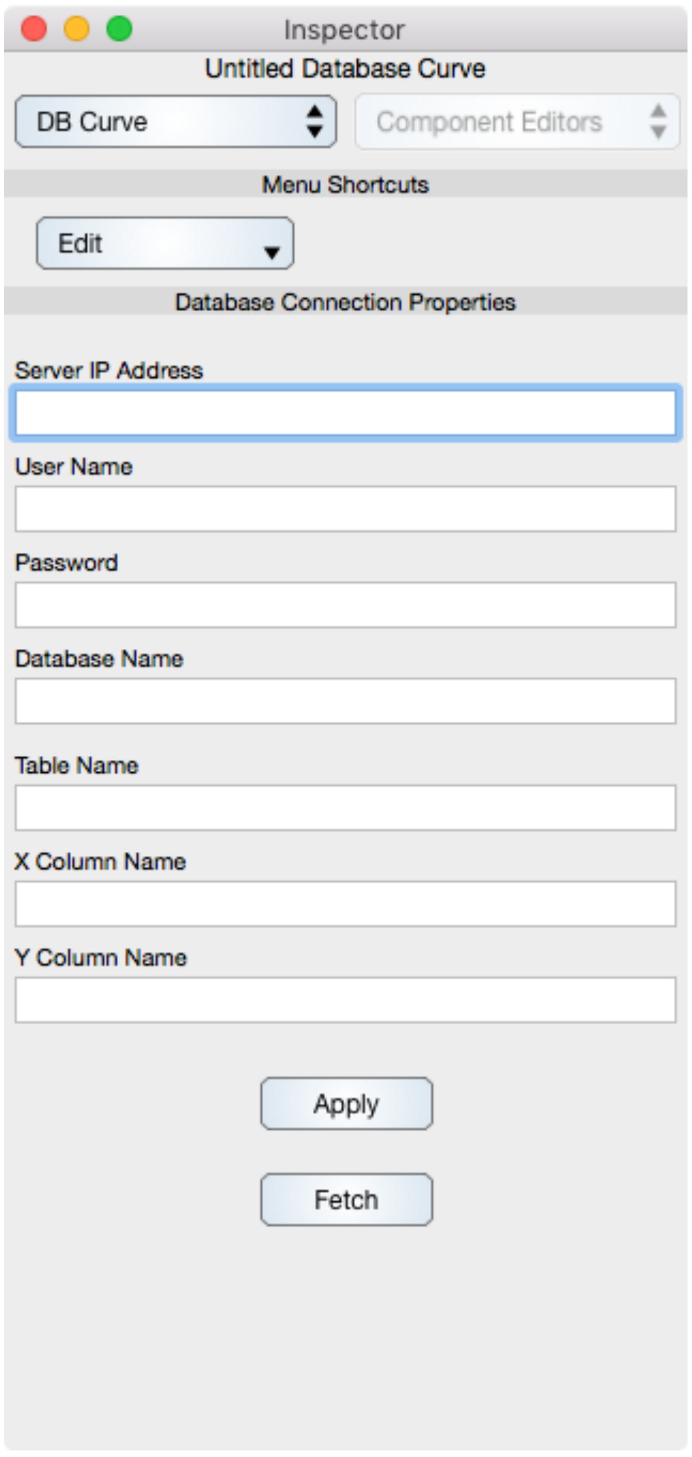
Once a Dynamic Graphic is loaded then it is replaced by an actual graphic that can be instantiated. That graphic is not part of the Graph IDE system and this section simply gives an example.

The example shows how a Scatter Graphic can be overloaded to present connection parameters for direct access to a database. For this to work, the connection library, such as an ODBC library, must be (statically) linked into the graphic bundle.

As you can see, the resulting graphic inspector editor shows little resemblance to the overloaded component within Graph IDE and that is the point. A graphic bundle can radically alter the Graph IDE functionality and focus only on the particular domain specific input parameters and business logic while still maintaining the generality of a comprehensive data visualization system.

Inspector Editor

The Inspector Editor for the database example is shown below.



Database Connection Properties

Server IP Address : The IP number of the server computer that hosts the database server.

User Name : The user name of an account within the database server.

Password : The password of an account within the database server.

Database Name : The database name within the database server.

Table Name : The table within the database.

X Column Name : The column name within the table that represents the independent variable.

Y Column Name : The column name within the table that represents the dependent variable.

Apply : Applies the Database Connection Properties.

Fetch : Fetches data from the database server and updates the associated graphical representation.

Graph IDE ► Programming

Graph IDE can be programmed using scripts and plugins. The script Language is similar to ANSI-C and Objective-C and making a Plugin utilizes Xcode and hence is a general-purpose facility that opens up a wide range of capabilities. Once scripts are inserted into the Program inspector editor then those scripts can be called periodically (animated) using the Animation inspector editor. In that way, a Graph IDE document can become dynamic and reach out to all resources available from your computer.

The following is a brief list of Programming sections:

Section	Description
Adapter	Describes how to program the adapter which is a hook to native view usage.
Advisory	Describes some programming issues. Some of the issues are usual and some are particular.
Circle	Describes how to program the Circle graphic.
Cubic Bezier	Describes how to program the Cubic Bezier graphic.
Function	Describes how to program the Function graphic.
Graphic	Describes how to program Graphics.
GraphicView	Describes how to program a GraphicView.
Image	Describes how to program the Image graphic.
Label	Describes how to program the Label graphic.
Language	Gives a basic overview of the programming language.
Layer	Describes how to program the Layer graphic.
Multiple Coordinate Graph	Describes how to program a Graph.
Overview	Gives an overview of the programming facilities.
Perspective Graph	Describes how to program the 3D Graph graphic.
Perspective Scatter	Describes how to program the 3D Scatter graphic.
Perspective Surface	Describes how to program the 3D Point Map graphic (a.k.a.: Surface Graph).
Point Map	Describes how to program the 2D Point Map graphic (a.k.a.: Heat Map).
Plugin	Describes how to write a plugin for optimized programming of a graphic.
Polygon	Describes how to program the Polygon graphic.
Rectangle	Describes how to program the Rectangle graphic.
Scatter	Describes how to program the Scatter graphic.
Single Coordinate Graph	Describes how to program a Graph.
Trajectory	Describes how to program the Trajectory graphic.

Graph IDE ► Programming ► Overview

Graphics are programmed using a general purpose scripting language. To see examples view the Palettes ► Programs menu which shows scripted and animated examples that are ready to drag and drop for use. The scripting language can be extended using a Plugin. The goal is to keep it simple yet very powerful. Before delving into serious programming issues, lets take a look at a script to animate setting colors of a circle.

```
/* Declarations */

@@class() Circle:Object

@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (void)setCurveRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)setInteriorRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)release;

@@end

/* Execution Block */

{
id myCircle;
int ii;
unsigned animationCount;
double red, green;

myCircle = [[Circle alloc] init];

animationCount = [myCircle animationCount];

red = (animationCount % 10) / 10.0;
green = (animationCount % 20) / 20.0;

[myCircle setInteriorRed:1.0 green:green blue:1.0 alpha:1.0];
[myCircle setCurveRed:red green:1.0 blue:1.0 alpha:1.0];

[myCircle release];

}
```

If you know the C language and the Objective-C language then you can immediately see what is happening in the script. The first part, Declarations, defines the API that will be used and the second part, Execution Block, is what is executed when the script runs. Lets focus on the execution block. The following are the main points:

- First you make a Circle in the normal way by dragging it out, select the Program inspector editor, paste the script above into the program inspector text field and then click the Execute button. That associates the script with an existing circle and executes the script.
- `myCircle = [[Circle alloc] init];` assigns a circle object to the existing Circle.
- `[myCircle setInteriorRed:1.0 green:green blue:1.0 alpha:1.0];` sets the interior (fill) color of that circle.
- `[myCircle setCurveRed:red green:1.0 blue:1.0 alpha:1.0];` sets the curve (a.k.a.: border or stroke) color of that circle.
- `[myCircle release];` frees the circle object that you made.
- When you set Animation on then the script is repeatedly executed at regular time intervals. Each animation step increments the integer returned by `[myCircle animationCount];` and in that way, the colors of the circle are made dynamic.
- Remember to set the Execute During Animation Program state if you animate a graphic.
- It is important to declare all the methods and functions in the script before using them. That is because the script binds to a multi-typed system and the only way to resolve binding is via declaration.

At this point, you have been exposed to an overview of the way to program graphics. The rest of the programming sections deal with programming specifications.

The script engine is a very powerful parsing facility that can parse, bind and execute source code similar to ANSI-C and Objective-C. It implements the usual ANSI-C functions and operators as well as syntax construction and binding to permit reference and execution of a function or method that can be looked up by the runtime system.

Because of the broad nature of the script engine only the most relevant features are described in this section. For additional information consult a reference on ANSI-C or Objective-C.

Functions

- The common functions are implemented such as: abs, acos, asin, atan, atan2, ceil, cos, cosh, div, exp, fabs, floor, fmod, frexp, labs, ldexp, ldiv, log, log10, modf, pow, rand, sin, sinh, sqrt, srand, tan, tanh. These functions are pre-declared for the Formula Selector principally because a formula is a single-statement program and as such can not accept a declaration statement.
- Theoretically, any function available to the binary can be accessed but you do need to specify the function declaration. For example the following declares the cosine function. Make sure to use the exact type needed, in this case a double type, when passing the argument. Without a specific declaration a function can not be accessed so it is important to declare functions in advance of the execution block.

```
double cos(double a);
```

- Scripting is not really a substitute for a large computation. Because of that you may want to consider implementing any large scale computation into a Plugin.

Operators

- The common operators are implemented such as: !, !=, %, %=, &&, &=, +, ++, +=, -, --, -=, /, <, >, <<, <<=, <=, =, ==, >, >=, >>, >>=. These operators are implicitly defined and do not have to be declared. They are all available within a single statement and as such can be used as a formula within the Formula Selector.

Expressions

The common expressions are implemented such as:

- if() else
- for()

Blocks

- The common blocks are implemented such as: () and {}. Note that the executable portion of a script (not the declaration) needs to be enclosed in a {} block. That is because the upper-most scope blocks are the entrance point to execution as well as scoping of variables.
- Both block types can be used in the Formula Selector formula statement, however the algebraic () block is the one that makes most sense.

Methods

- Methods are written like `@@method(public, instance)myMethod:(int)anArg` and `@@method(public, class)myMethod:(int)anArg` and wrapped in a class block like this: `@@class() Rectangle:Object ... @@end`.
- The analog in Objective-C is fairly straightforward.

Use

The following script fragment gives an example.

```
[myFunction emptyData];

for(ii = 0; ii < 500; ii++)
{
xValue = 0.01 * ii + animationCount * 0.5;
yValue = cos(xValue);
[myFunction appendXValue:xValue yValue:yValue];
}
```

That makes a cosine curve (function) in the usual way. For specific scripts see each API code section, for example the Function section.

The following is a complete script for programming a Single Coordinate Graph. It places a Function onto the graph and then autoscales the graph.

```
/* Declarations */

@@class() SingleCoordinateGraph:Object

@@method(public, class) (id)stored;
@@method(public, class) (id)alloc;
@@method(public, class) (id)append;
@@method(public, instance) (id)init;
@@method(public, instance) (void)autoscale;
@@method(public, instance) (void)focus;
@@method(public, instance) (void)unfocus;
@@method(public, instance) (void)release;

@@end

double cos(double a);

@@class() Function:Object

@@method(public, class) (id)alloc;
@@method(public, class) (id)stored;
@@method(public, class) (id)append;
@@method(public, instance) (id)init;
@@method(public, instance) (void)emptyData;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (void)appendXValue:(double)xValue yValue:(double)yValue;
@@method(public, instance) (void)setCurveRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)setInteriorRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)release;

@@end

/* Execution block */

{
id aGraphic, aGraph;
int ii;
unsigned animationCount;

aGraph = [SingleCoordinateGraph append];

[aGraph moveCenterToXValue:400.0 yValue:400.0];
[aGraph sizeToCenteredWidth:300.0 height:300.0];

[aGraph focus];

{
id myFunction;
int ii;
double xValue, yValue;
unsigned animationCount;
double red;

myFunction = [Function append];

animationCount = [myFunction animationCount];

/*
Empty the data and then append new data.
*/

[myFunction emptyData];

for(ii = 0; ii < 500; ii++)
{
xValue = 0.01 * ii + animationCount * 0.5;
yValue = cos(xValue);
[myFunction appendXValue:xValue yValue:yValue];
```

```
    }

    [myFunction finalize];

    }

    [aGraph unfocus];

    [aGraph autoscale];

    [aGraph finalize];

    }
```

The general API is define in the section Graphic. The following is API description specific to the Single Coordinate Graph graphic.

`@@method(public, instance) (void)autoscale;`

Call like this:

`[aGraph autoscale];`

Autoscales the receiver.

`@@method(public, instance) (void)focus;`

Call like this:

`[aGraph focus];`

Sets the focused layer to the receiver's foreground data layer. After this call any graphics that are inserted or appended are done so to the graph's foreground data layer.

`@@method(public, instance) (void)unfocus;`

Call like this:

`[aGraph unfocus];`

Sets the focused layer to the layer that the receiver is in. After this call any graphics that are inserted or appended are done so to the same layer as the graph.

`@@method(public, instance) (void)updateMainTitleToUFT8String:(const char *)aUTF8String;`

Call like this:

`[aGraph updateMainTitleToUFT8String:"My Title"];`

Sets the main title of the graph.

`@@method(public, instance) (void)updateXTitleToUFT8String:(const char *)aUTF8String;`

Call like this:

`[aGraph updateXTitleToUFT8String:"My X Title"];`

Sets the x-axis title of the graph.

`@@method(public, instance) (void)updateYTitleToUFT8String:(const char *)aUTF8String;`

Call like this:

`[aGraph updateYTitleToUFT8String:"My Y Title"];`

Sets the y-axis title of the graph.

The following shows how to script a Multiple Coordinate Graph.

```
/* Declarations */

@@class() MultipleCoordinateGraph:Object

@@method(public, class) (id)stored;
@@method(public, instance) (void)updateXTitleToUFT8String:(const char *)aUTF8String;
@@method(public, instance) (void)updateYTitleAtAxisIndex:(unsigned)axisIndex toUFT8String:(const char
*)aUTF8String;

@@end

/* Execution block */

{
id aGraph;

aGraph = [MultipleCoordinateGraph append];

[aGraph updateXTitleToUFT8String:"My X Title"];
[aGraph updateYTitleAtAxisIndex:1U toUFT8String:"My Y Title"];

}
```

The general API is define in the section Graphic. The following is API description specific to the Multiple Coordinate Graph graphic.

`@@method(public, instance) (void)autoscale;`

Call like this:

`[aGraph autoscale];`

Autoscales the receiver.

`@@method(public, instance) (void)focus;`

Call like this:

`[aGraph focus];`

Sets the focused layer to the receiver's foreground data layer. After this call any graphics that are inserted or appended are done so to the graph's foreground data layer.

`@@method(public, instance) (void)unfocus;`

Call like this:

`[aGraph unfocus];`

Sets the focused layer to the layer that the receiver is in. After this call any graphics that are inserted or appended are done so to the same layer as the graph.

`@@method(public, instance) (void)updateMainTitleToUFT8String:(const char *)aUTF8String;`

Call like this:

`[aGraph updateMainTitleToUFT8String:"My Title"];`

Sets the main title of the graph.

`@@method(public, instance) (void)updateXTitleToUFT8String:(const char *)aUTF8String;`

Call like this:

`[aGraph updateXTitleToUFT8String:"My X Title"];`

Sets the x-axis title of the graph.

```
@@method(public, instance) (void)updateYTitleAtAxisIndex:(unsigned)axisIndex toUFT8String:(const char
*)aUTF8String;
```

Call like this:

```
[aGraph updateYTitleAtAxisIndex:1U toUFT8String:"My Y Title"];
```

Sets the first y-axis title of the graph.

```
@@method(public, instance) (void)updateYTitleToUFT8String:(const char *)aUTF8String;
```

Call like this:

```
[aGraph updateYTitleToUFT8String:"My Y Title"];
```

Sets the first y-axis title of the graph.

Graph IDE ► Programming ► Graphic

This section describes the API specific to Graphic processing. To see the use of this API consult the Circle section.

```
@@method(public, class) (id)alloc;
```

Call like this:

```
[Circle alloc]
```

That allocates a Circle object. In practice the receiver class (Circle in this example) needs to be the one specific to the focused graphic such as Circle, Rectangle, Function, etc. or a subclass defined in your own Plugin.

```
@@method(public, class) (id)append;
```

Call like this:

```
myCircle = [Circle append];
```

That appends a new Circle graphic in the focused Layer, at the end of the graphic nodes, and then returns the circle object so that it can be programmed using Circle methods. The receiver class (Circle in this example) can be any of the Programming objects.

```
@@method(public, class) (id)insert;
```

Call like this:

```
myCircle = [Circle insert];
```

That inserts a new Circle graphic in the focused Layer, in front of the currently focused graphic, and then returns the circle object so that it can be programmed using Circle methods. The receiver class (Circle in this example) can be any of the Programming objects.

```
@@method(public, class) (id)stored;
```

Call like this:

```
myCircle = [Circle stored];
```

Returns an instance of the receiver class. That instance is stored by the associated graphic if the class implements the NSCoding protocol and hence the state is persistent and part of the document when saved and retrieved (see Plugin for additional details). If you call this method to make an instance then do not call release on the return. Reminder: if you use this then make sure to declare the method in a declaration section of the script. As none of the stock classes implement the NSCoding protocol this method is only applicable to custom classes defined in a plugin.

```
@@method(public, instance) (id)init;
```

Call like this:

```
myCircle = [[Circle alloc] init];
```

That allocates, initializes and associates the Circle object with the focused graphic. Make sure the variable assigned to is previously declared as an id type.

```
@@method(public, instance) (void)moveCenterToXValue:(double)xValue yValue:(double)yValue;
```

Call like this:

```
[myCircle moveCenterToXValue:200.0 yValue:350.0];
```

That sets the circle's center point to the coordinate {200.0, 350.0}.

```
@@method(public, instance) (void)display;
```

Call like this:

```
[myCircle display];
```

Displays the graphic of the receiver (a circle in this case). Normally the display occurs at the end of the animation loop of the graphic view. However, sometimes it is desired to directly display the graphic and this method does that. Great care is taken in the system to only display the region that the receiver's graphic occupies. Do not call this method from a script, rather call it from a Custom Application.

```
@@method(public, instance) (void)execute;
```

Call like this:

```
[myCircle execute];
```

Executes the script of the receiver (a circle in this case). Normally the script is executed in the animation loop of the graphic view by using the methods in the Container View class instance. However, sometimes it is desired to directly execute the script for only one particular graphic and this method does that. Executing does not display the graphic so in order to display the changes of the execution call the display method. Do not call this method from a script, rather call it from a Custom Application.

`@@method(public, instance) (void)remove;`

Call like this:

```
[myCircle remove];
```

That removes the receiver (a circle in this case) from the Layer and deallocates it. After this call the receiver is no longer valid.

`@@method(public, instance) (void)sizeToCenteredWidth:(double)width height:(double)height;`

Call like this:

```
[myCircle sizeToCenteredWidth:100.0 height:200.0];
```

That sets the circle's width to 100.0 and height to 200.0 while maintaining the circle's center point.

`@@method(public, instance) (void)setInteriorRed:(double)red green:(double)green blue:(double)blue alpha:(double)alpha;`

Call like this:

```
[myCircle setInteriorRed:0.5 green:0.4 blue:1.0 alpha:1.0];
```

That sets the interior color of the circle to the red, green, blue and alpha values of 0.5, 0.4, 1.0 and 1.0 respectively. Those values must be between 0.0 and 1.0. An alpha of 0.0 is transparent while 1.0 is completely opaque. Each argument must be a number literal or a variable (or expression) of type double.

`@@method(public, instance) (void)setCurveRed:(double)red green:(double)green blue:(double)blue alpha:(double)alpha;`

Call like this:

```
[myCircle setCurveRed:0.5 green:0.4 blue:1.0 alpha:1.0];
```

That sets the curve (a.k.a.: border, parameter) color of the circle to the red, green, blue and alpha values of 0.5, 0.4, 1.0 and 1.0 respectively. Those values must be between 0.0 and 1.0. An alpha of 0.0 is transparent while 1.0 is completely opaque. Each argument must be a number literal or a variable (or expression) of type double.

`@@method(public, instance) (void)release;`

Call like this:

```
[myCircle release];
```

That releases the previously made circle. It is very important that you release each object that has been previously allocated.

`@@method(public, instance) (unsigned)animationCount;`

Call like this:

```
animationCount = [myCircle animationCount];
```

That assigns the circle's animationCount to the variable on the left. The animation count can be used for all sorts of purposes. For example this: `(animationCount % 10) / 10.0` can be used to make a red color that modulates and hence makes a color appear to pulse.

`@@method(public, instance) (id)controller;`

Call like this:

```
controller = [myCircle controller];
```

That returns the controller. The controller is assigned utilizing either Document or Container View methods for loading views into a custom application. Once you are able to access the controller then you can send it any method you wish. In that way, you can synchronize the caller with other elements of an application.

`@@method(public, instance) (unsigned)executionCount;`

Call like this:

```
executionCount = [myCircle executionCount];
```

That assigns the circle's executionCount to the variable on the left. The execution count is mainly to determine when initialization code should be executed. If its value is one then the script is executed for the first time.

```
@@method(public, instance) (unsigned)recursionCount;
```

Call like this:

```
recursionCount = [myCircle recursionCount];
```

That assigns the circle's recursionCount to the variable on the left. Unless you call the recur method the recursionCount is always zero.

```
@@method(public, instance) (void)recur;
```

Call like this:

```
[myCircle recur];
```

Causes the script and associated graphic to be duplicated and executed. Since this can happen anywhere in the existing script it causes a recursion. To stop the recursion use recursionCount and a conditional. It is very important to bracket this call with a conditional so that the recursion is not infinite. There is no safety mechanism for infinite recursion so you must take precautions. This call is most important for data graphics such as curves where you might want to start with one curve and computer, say, ten curves for a line graph with many curves.

Recursion inserts new graphics into the layer of the graphic. Because the script runs in a general purpose graphic drawing application some care is taken to make sure the insertion is done correctly. If conditions are nominal then the insertion is as expected. However, there are some circumstances that break recursion. For example, if you execute the recursion and then drag out a new graphic, reorder that graphic into the recursion insertion then that graphic will break the existing recursion chain and execution of a new recursion will insert recursion graphical elements before the manually added graphic and will not reuse the recursive graphics after the manually inserted graphic.

```
@@method(public, instance) (void)registerWithName:(const char *)aName;
```

Call like this:

```
[myCircle registerWithName:"myName"];
```

That calls the controller's `register:withName:recursionIndex:` method. The controller is assigned utilizing either Document or Container View methods for loading views into a custom application. The method above is equivalent to `[[myCircle controller] register:myCircle WithName:"myName" recursionIndex:[myCircle recursionIndex]];`.

```
@@method(public, instance) (void)setSourceCode:(const char *)aString;
```

Call like this:

```
[myCircle setSourceCode:"<insert a script here>"];
```

That injects the source code into the receiver. The source code is then executed after the current script executes; thus this is not a serialized programming technique. Usually this method is only called in a Graphic View script.

```
@@method(public, instance) (void)finalize;
```

Call like this:

```
[myCircle finalize];
```

That finalize the graphic. Only call this method within a Graphic View script.

The following is a complete script for programming a Graphic View. There are a few important ideas demonstrated by the script which are listed as follows:

- Previous graphics on the Graphic View are first removed using empty. Thus this is not a script that leverages pre-made settings of the graphics. It is more like a script, or program, that is used to completely and programmatically make the contents of a view.
- Normally each graphic is programmed as needed. That has the advantage of being able to set attributes through a user interface and also define packets of programs within a nested hierarchy of objects thus you do not have to program the nesting. That nesting is not available to the Graphic View because the Graphic View is flat (one object). In order to inject code into a tree structure of graphics (a nested Layer) the setSourceCode method is used.
- Notice how the blue portion of text in the script is injected into an object in a manner similar to an anonymous function or block. That seems like a very complex way of programming, however the complexity is for a purpose. By injecting the code it is a reference to an object node. That node is then executed upon by the recur method. The code injection is done so that the object can be recursed. An alternative might be to setup a for() loop.

Programming a Graphic View is not nearly as simple as programming individual graphics and you may want to steer clear of this object. However, if you are a beefy programmer then this section may be for you.

```
/*
This is a complete example of how to program without any user interaction.
*/

/* Declarations */

@@class() GraphicView:Object

@@method(public, class) (id)stored;
@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (void)empty;
@@method(public, instance) (void)release;

@@end

@@class() SingleCoordinateGraph:Object

@@method(public, class) (id)stored;
@@method(public, class) (id)alloc;
@@method(public, class) (id)append;
@@method(public, instance) (id)init;
@@method(public, instance) (void)setDoesAutoscaleAfterExecution:(unsigned)doesAutoscaleAfterExecution;
@@method(public, instance) (void)autoscale;
@@method(public, instance) (void)focus;
@@method(public, instance) (void)unfocus;
@@method(public, instance) (void)release;

@@end

@@class() Circle:Object

@@method(public, class) (id)insert;
@@method(public, class) (id)append;
@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (void)setCurveRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)setInteriorRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)moveCenterToXValue:(double)xValue yValue:(double)yValue;
@@method(public, instance) (void)sizeToCenteredWidth:(double)width height:(double)height;

@@method(public, instance) (void)finalize;
@@method(public, instance) (void)release;

@@end

@@class() Rectangle:Object

@@method(public, class) (id)insert;
@@method(public, class) (id)append;
@@method(public, class) (id)alloc;
```

```
@@method(public, instance) (id)init;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (void)setCurveRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)setInteriorRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)moveCenterToXValue:(double)xValue yValue:(double)yValue;
@@method(public, instance) (void)sizeToCenteredWidth:(double)width height:(double)height;

@@method(public, instance) (void)finalize;
@@method(public, instance) (void)release;

@@end

@@class() Function:Object

@@method(public, class) (id)alloc;
@@method(public, class) (id)stored;
@@method(public, class) (id)append;
@@method(public, instance) (id)init;
@@method(public, instance) (void)setSourceCode:(const char *)aString;
@@method(public, instance) (void)emptyData;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (unsigned)recursionCount;
@@method(public, instance) (void)appendXValue:(double)xValue yValue:(double)yValue;
@@method(public, instance) (void)setCurveRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)setInteriorRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)release;

@@end

/* Execution block */

{
id aGraphicView;
id aGraphic, aGraph;
int ii;
unsigned animationCount;
double red, green, blue;

aGraphicView = [GraphicView stored];

animationCount = [aGraphicView animationCount];

[aGraphicView empty];

aGraphic = [Circle append];

red = (animationCount % 10) / 10.0;
green = (animationCount % 20) / 20.0;
blue = 1.0;

[aGraphic setInteriorRed:1.0 green:green blue:1.0 alpha:1.0];
[aGraphic setCurveRed:red green:1.0 blue:1.0 alpha:1.0];
[aGraphic moveCenterToXValue:300.0 yValue:400.0];
[aGraphic sizeToCenteredWidth:50.0 height:200.0];

[aGraphic finalize];

aGraphic = [Rectangle append];

red = (animationCount % 10) / 10.0;
green = (animationCount % 20) / 20.0;
blue = 0.0;

[aGraphic setInteriorRed:red green:green blue:blue alpha:1.0];
[aGraphic setCurveRed:0.0 green:0.0 blue:0.0 alpha:1.0];
[aGraphic moveCenterToXValue:300.0 yValue:400.0];
[aGraphic sizeToCenteredWidth:50.0 height:200.0];

[aGraphic finalize];

aGraph = [SingleCoordinateGraph append];

[aGraph moveCenterToXValue:400.0 yValue:400.0];
[aGraph sizeToCenteredWidth:300.0 height:300.0];
[aGraph setDoesAutoscaleAfterExecution:1];

[aGraph focus];

aGraphic = [Function append];

[aGraphic setSourceCode:"
double cos(double a);

@@class() Function:Object

@@method(public, class) (id)alloc;
@@method(public, class) (id)stored;
@@method(public, class) (id)append;
@@method(public, instance) (id)init;
@@method(public, instance) (void)setSourceCode:(const char *)aString;
@@method(public, instance) (void)emptyData;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (unsigned)recursionCount;
@@method(public, instance) (void)appendXValue:(double)xValue yValue:(double)yValue;
@@method(public, instance) (void)setCurveRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)setInteriorRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)release;

@@end

{
id myFunction;
int ii;
double xValue, yValue;
unsigned animationCount;
unsigned recursionCount;
double red;

myFunction = [MyFunction stored];

animationCount = [myFunction animationCount];
recursionCount = [myFunction recursionCount];

/*
Empty the data and then append new data.
*/

[myFunction emptyData];

for(ii = 0; ii < 500; ii++)
{
xValue = 0.01 * ii + animationCount * 0.5;
yValue = cos(xValue) + recursionCount;
[myFunction appendXValue:xValue yValue:yValue];
}

red = (animationCount % 10) / 10.0;

[myFunction setCurveRed:red green:0.0 blue:0.0 alpha:1.0];

if(recursionCount < 5)
{
[myFunction recur];
}

}"];

[aGraph unfocus];

[aGraph finalize];

[aGraphicView finalize];

}
```

The following is API description specific to the Graphic View.

```
@@method(public, class) (id)stored;
```

Call like this:

```
aGraphicView = [GraphicView stored];
```

Returns an instance of the receiver class. That instance is stored by the associated Graphic View if the class implements the NSCoding protocol and hence the state is persistent and part of the document when saved and retrieved (see Plugin for additional details). If you call this method to make an instance then do not call release on the return. Reminder: if you use this then make sure to declare the method in a declaration section of the script. As none of the stock classes implement the NSCoding protocol this method is only applicable to custom classes defined in a plugin.

```
@@method(public, class) (id)alloc;
```

Call like this:

```
[GraphicView alloc];
```

That allocates a Graphic View object.

```
@@method(public, instance) (id)init;
```

Call like this:

```
myGraphicView = [[GraphicView alloc] init];
```

That allocates, initializes and associates the GraphicView object with the focused Graphic View. Make sure the variable assigned to is previously declared as an id type.

```
@@method(public, instance) (unsigned)animationCount;
```

Call like this:

```
animationCount = [myGraphicView animationCount];
```

That assigns the Graphic View's animationCount to the variable on the left.

```
@@method(public, instance) (void)empty;
```

Call like this:

```
[myGraphicView empty];
```

Removes all graphics from the Graphic View.

```
@@method(public, instance) (void)release;
```

Call like this:

```
[myGraphicView release];
```

That releases the previously made Graphic View. It is very important that you release each object that has been previously allocated.

Graph IDE ▶ Programming ▶ Circle

The following is a complete script for programming a Circle graphic. When animated, it makes the circle blink and move around in a circle. Remember to set the Execute During Animation Program state if you animate a graphic.

```
/* Declarations */

double cos(double a);
double sin(double a);

@@class() Circle:Object

@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (void)setCurveRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)setInteriorRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)moveCenterToXValue:(double)xValue yValue:(double)yValue;
@@method(public, instance) (void)sizeToCenteredWidth:(double)width height:(double)height;
@@method(public, instance) (void)release;

@@end

/* Execution block */

{
id myCircle;
int ii;
unsigned animationCount;
double red, green;
double xValue, yValue;

myCircle = [[Circle alloc] init];

animationCount = [myCircle animationCount];

red = (animationCount % 10) / 10.0;
green = (animationCount % 20) / 20.0;

xValue = 250.0 + 100.0 * cos(animationCount/20.0);
yValue = 250.0 + 100.0 * sin(animationCount/20.0);

[myCircle setInteriorRed:1.0 green:green blue:1.0 alpha:1.0];
[myCircle setCurveRed:red green:1.0 blue:1.0 alpha:1.0];
[myCircle moveCenterToXValue:xValue yValue:yValue];

[myCircle release];

}
```

The general API is define in the section Graphic. The following is API description specific to the Circle graphic.

```
@@method(public, instance) (void)setWedgeStartAngle:(double)startAngleInRadians endAngle:
(double)endAngleInRadians;
```

Call like this:

```
[myCircle setWedgeStartAngle:0.0 endAngle:3.1415926];
```

Sets the start and end angle of the circle. The angles are specified in unit of radian.

Graph IDE ► Programming ► Rectangle

The following is a complete script for programming a Rectangle graphic. When animated, it makes the rectangle blink and move around in a circle. Remember to set the Execute During Animation Program state if you animate a graphic.

```
/* Declarations */

double cos(double a);
double sin(double a);

@@class() Rectangle:Object

@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (void)setCurveRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)setInteriorRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)moveCenterToXValue:(double)xValue yValue:(double)yValue;
@@method(public, instance) (void)sizeToCenteredWidth:(double)width height:(double)height;
@@method(public, instance) (void)release;

@@end

/* Execution block */

{
id myRectangle;
int ii;
unsigned animationCount;
double red, green;
double xValue, yValue;

myRectangle = [[Rectangle alloc] init];

animationCount = [myRectangle animationCount];

red = (animationCount % 10) / 10.0;
green = (animationCount % 20) / 20.0;

xValue = 250.0 + 100.0 * cos(animationCount/20.0);
yValue = 250.0 + 100.0 * sin(animationCount/20.0);

[myRectangle setInteriorRed:1.0 green:green blue:1.0 alpha:1.0];
[myRectangle setCurveRed:red green:1.0 blue:1.0 alpha:1.0];
[myRectangle moveCenterToXValue:xValue yValue:yValue];

[myRectangle release];

}
```

The general API is define in the section Graphic.

The following is a complete script for programming a Function. It computes a cosine curve and then duplicates that curve five times using the recur method. The cosine is shifted up upon each recursion to demonstrate how to alter the data based upon recursion. Since it is often the case that a graph has multiple curves that are computed based upon data, the recur method is very important to the function graphic.

```
/* Declarations */

double cos(double a);

@@class() Function:Object

@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (void)emptyData;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (unsigned)recursionCount;
@@method(public, instance) (void)appendXValue:(double)xValue yValue:(double)yValue;
@@method(public, instance) (void)setCurveRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)setInteriorRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)recur;
@@method(public, instance) (void)release;

@@end

/* Execution block */

{
id myFunction;
int ii;
double xValue, yValue;
unsigned animationCount;
unsigned recursionCount;
double red;

myFunction = [[Function alloc] init];

animationCount = [myFunction animationCount];
recursionCount = [myFunction recursionCount];

printf("animationCount: %d\n", animationCount);
printf("recursionCount: %d\n", recursionCount);

/*
Empty the data and then append new data.
*/

[myFunction emptyData];

for(ii = 0; ii < 500; ii++)
{
xValue = 0.01 * ii + animationCount * 0.5;
yValue = cos(xValue) + recursionCount;
[myFunction appendXValue:xValue yValue:yValue];
}

red = (animationCount % 10) / 10.0;

[myFunction setCurveRed:red green:0.0 blue:0.0 alpha:1.0];

if(recursionCount < 5)
{
[myFunction recur];
}

[myFunction release];

}
```

The general API is define in the section Graphic. The following is API description specific to the Function graphic.

```
@@method(public, instance) (void)appendXValue:(double)xValue yValue:(double)yValue;
```

Call like this:

```
[myFunction appendXValue:xValue yValue:yValue];
```

Appends the x and y values to the list of data points for the graphic. The x and y values forms a 2D point. Each value must be of type double.

```
@@method(public, instance) (void)appendMarkerRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
```

Call like this:

```
[myFunction appendMarkerRed:0.5 green:0.4 blue:1.0 alpha:1.0];
```

That appends the marker color to the red, green, blue and alpha values of 0.5, 0.4, 1.0 and 1.0 respectively. Those values must be between 0.0 and 1.0. An alpha of 0.0 is transparent while 1.0 is completely opaque. Each argument must be a number literal or a variable (or expression) of type double. Note that this call must accompany a appendXValue:xValue yValue: call in order to synchronize the parameters that depend upon sequence index.

```
@@method(public, instance) (void)appendSegmentRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
```

Call like this:

```
[myFunction appendSegmentRed:0.5 green:0.4 blue:1.0 alpha:1.0];
```

That appends the segment color to the red, green, blue and alpha values of 0.5, 0.4, 1.0 and 1.0 respectively. Those values must be between 0.0 and 1.0. An alpha of 0.0 is transparent while 1.0 is completely opaque. Each argument must be a number literal or a variable (or expression) of type double. Note that this call must accompany a appendXValue:xValue yValue: call in order to synchronize the parameters that depend upon sequence index.

```
@@method(public, instance) (void)appendBubbleValue:(double)aValue;
```

Call like this:

```
[myFunction appendBubbleValue:aValue];
```

Appends aValue to the list of bubble values for the graphic. aValue must be of type double.

```
@@method(public, instance) (void)sort;
```

Call like this:

```
[myFunction sort];
```

Sorts the data points in x-order. You should attempt to append the data points in x-ascending order but if that is not possible then call this sort method after appending all data points. The points must be x-ascending.

```
@@method(public, instance) (void)emptyData;
```

Call like this:

```
[myFunction emptyData];
```

Removes (empties) all data from the graphic. Call this right before adding new data points.

Graph IDE ► Programming ► Label

The following is a complete script for programming a Label graphic. It places a UTF8 string into the label. Not that assigning the string will not size to fit the label and will not change the label's bounding box.

Note that you can assign only string constants to the label. To construct strings using more general methods use a Plugin.

```
/* Declarations */

@@class() Label:Object

@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (void)setUFT8String:(const char *)aString;
@@method(public, instance) (void)release;

@@end

/* Execution block */

{
id myLabel;
unsigned animationCount;

myLabel = [[Label alloc] init];

animationCount = [myLabel animationCount];

[myLabel setUFT8String:"This is a new string"];

[myLabel release];

}
```

The general API is define in the section Graphic. The following is API description specific to the Label graphic.

```
@@method(public, instance) (void)setUFT8String:(const char *)aString;
```

Call like this:

```
[myLabel setUFT8String:"This is a new string"];
```

Assigns the string (aString) to the label without size to fit and without changing the label's bounding box.

The following is a complete script for programming a Layer graphic. It enables the layer's event qualifier. If an event qualifier is not enabled in this fashion then normal event processing occurs which is probably not the desired result so make sure to enable event qualifiers in all layers of the document.

```
@@class() Layer:Object

@@method(public, class) (id)stored;
@@method(public, instance) (void)enableEventQualifier;

@@end

{
id myLayer;
id myEventQualifier;

myLayer = [Layer stored];

[myLayer enableEventQualifier];
}
}
```

The general API is define in the section Graphic. The following is API description specific to the Layer graphic.

```
@@method(public, instance) (void)enableEventQualifier;
```

Call like this:

```
[myLayer enableEventQualifier];
```

Enables the layer's event qualifier.

```
@@method(public, instance) (void)disableEventQualifier;
```

Call like this:

```
[myLayer disableEventQualifier];
```

Disables the layer's event qualifier.

Graph IDE ► Programming ► Image

The following is a complete script for programming a Image graphic. When animated, it makes the image move around in a circle. Remember to set the Execute During Animation Program state if you animate a graphic.

```
/* Declarations */

double cos(double a);
double sin(double a);

@@class() Image:Object

@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (void)moveCenterToXValue:(double)xValue yValue:(double)yValue;
@@method(public, instance) (void)sizeToCenteredWidth:(double)width height:(double)height;
@@method(public, instance) (void)release;

@@end

/* Execution block */

{
id myImage;
int ii;
unsigned animationCount;
double xValue, yValue;

myImage = [[Image alloc] init];

animationCount = [myImage animationCount];

red = (animationCount % 10) / 10.0;
green = (animationCount % 20) / 20.0;

xValue = 250.0 + 100.0 * cos(animationCount/20.0);
yValue = 250.0 + 100.0 * sin(animationCount/20.0);

[myImage moveCenterToXValue:xValue yValue:yValue];

[myImage release];

}
```

The general API is define in the section Graphic.

The following is a complete script for programming a Polygon graphic. It computes a somewhat circular distribution of points.

```
/* Declarations */

double cos(double a);
double sin(double a);

@@class() Polygon:Object

@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (void)emptyData;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (void)appendXValue:(double)xValue yValue:(double)yValue;
@@method(public, instance) (void)release;

@@end

/* Execution block */

{
id myPolygon;
int ii;
double xValue, yValue;
unsigned animationCount;

myPolygon = [[Polygon alloc] init];

animationCount = [myPolygon animationCount];

printf("animationCount: %d\n", animationCount);

/*
Empty the data and then append new data.
*/

[myPolygon emptyData];

for(ii = 0; ii < 20; ii++)
{
red = (animationCount % 10) / 10.0;
green = (animationCount % 30) / 30.0;
blue = (ii % 20) / 20.0;
xValue = cos(ii * .02) + red * sin(ii * .01);
yValue = sin(ii * .02);

[myPolygon appendXValue:xValue yValue:yValue];
}

[myPolygon release];

}
```

The general API is define in the section Graphic. The following is API description specific to the Polygon graphic.

```
@@method(public, instance) (void)appendXValue:(double)xValue yValue:(double)yValue;
```

Call like this:

```
[myPolygon appendXValue:xValue yValue:yValue];
```

Appends the x and y values to the list of data points for the graphic. The x and y values forms a 2D point. Each value must be of type double.

```
@@method(public, instance) (void)emptyData;
```

Call like this:

```
[myPolygon emptyData];
```

Removes (empties) all data from the graphic. Call this right before adding new data points.

Graph IDE ▶ Programming ▶ Cubic Bezier

The following is a complete script for programming a Cubic Bezier graphic. It computes a somewhat circular distribution of points.

```
/* Declarations */

double cos(double a);
double sin(double a);

@@class() CubicBezier:Object

@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (void)emptyData;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (void)appendXValue:(double)xValue yValue:(double)yValue;
@@method(public, instance) (void)release;

@@end

/* Execution block */

{
id myCubicBezier;
int ii;
double xValue, yValue;
unsigned animationCount;

myCubicBezier = [[CubicBezier alloc] init];

animationCount = [myCubicBezier animationCount];

printf("animationCount: %d\n", animationCount);

/*
Empty the data and then append new data.
*/

[myCubicBezier emptyData];

for(ii = 0; ii < 20; ii++)
{
red = (animationCount % 10) / 10.0;
green = (animationCount % 30) / 30.0;
blue = (ii % 20) / 20.0;
xValue = cos(ii * .02) + red * sin(ii * .01);
yValue = sin(ii * .02);

[myCubicBezier appendXValue:xValue yValue:yValue];
}

[myCubicBezier release];

}
```

The general API is define in the section Graphic. The following is API description specific to the Cubic Bezier graphic.

```
@@method(public, instance) (void)appendXValue:(double)xValue yValue:(double)yValue;
```

Call like this:

```
[myCubicBezier appendXValue:xValue yValue:yValue];
```

Appends the x and y values to the list of data points for the graphic. The x and y values forms a 2D point. Each value must be of type double.

```
@@method(public, instance) (void)emptyData;
```

Call like this:

```
[myCubicBezier emptyData];
```

Removes (empties) all data from the graphic. Call this right before adding new data points.

The following is a complete script for programming a Scatter graphic. It computes a somewhat circular distribution of points and also assigns a bubble value. For this to work, the scatter graphic must have been made with a point tag marker.

```
/* Declarations */

double cos(double a);
double sin(double a);

@@class() Scatter:Object

@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (void)emptyData;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (void)appendXValue:(double)xValue yValue:(double)yValue;
@@method(public, instance) (void)setCurveRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)appendBubbleValue:(double)aValue;
@@method(public, instance) (void)appendMarkerRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)release;

@@end

/* Execution block */

{
id myScatter;
int ii;
double xValue, yValue;
unsigned animationCount;
double red, green;

myScatter = [[Scatter alloc] init];

animationCount = [myScatter animationCount];

printf("animationCount: %d\n", animationCount);

/*
Empty the data and then append new data.
*/

[myScatter emptyData];

for(ii = 0; ii < 20; ii++)
{
red = (animationCount % 10) / 10.0;
green = (animationCount % 30) / 30.0;
xValue = cos(ii * .02) + red * sin(ii * .01);
yValue = sin(ii * .02);

[myScatter appendXValue:xValue yValue:yValue];
[myScatter appendMarkerRed:red green:0.0 blue:1.0 alpha:1.0];
[myScatter appendBubbleValue:(ii * 1.0)];
}

[myScatter release];

}
```

The general API is define in the section Graphic. The following is API description specific to the Scatter graphic.

```
@@method(public, instance) (void)appendXValue:(double)xValue yValue:(double)yValue;
```

Call like this:

```
[myScatter appendXValue:xValue yValue:yValue];
```

Appends the x and y values to the list of data points for the graphic. The x and y values forms a 2D point. Each value must be of type double.

```
@@method(public, instance) (void)appendMarkerRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
```

Call like this:

```
[myScatter appendMarkerRed:0.5 green:0.4 blue:1.0 alpha:1.0];
```

That appends the marker color to the red, green, blue and alpha values of 0.5, 0.4, 1.0 and 1.0 respectively. Those values must be between 0.0 and 1.0. An alpha of 0.0 is transparent while 1.0 is completely opaque. Each argument must be a number literal or a variable (or expression) of type double. Note that this call must accompany a appendXValue:xValue yValue: call in order to synchronize the parameters that depend upon sequence index.

```
@@method(public, instance) (void)appendSegmentRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
```

Call like this:

```
[myScatter appendSegmentRed:0.5 green:0.4 blue:1.0 alpha:1.0];
```

That appends the segment color to the red, green, blue and alpha values of 0.5, 0.4, 1.0 and 1.0 respectively. Those values must be between 0.0 and 1.0. An alpha of 0.0 is transparent while 1.0 is completely opaque. Each argument must be a number literal or a variable (or expression) of type double. Note that this call must accompany a appendXValue:xValue yValue: call in order to synchronize the parameters that depend upon sequence index.

```
@@method(public, instance) (void)appendBubbleValue:(double)aValue;
```

Call like this:

```
[myScatter appendBubbleValue:aValue];
```

Appends aValue to the list of bubble values for the graphic. aValue must be of type double.

```
@@method(public, instance) (void)emptyData;
```

Call like this:

```
[myScatter emptyData];
```

Removes (empties) all data from the graphic. Call this right before adding new data points.

Graph IDE ► Programming ► Trajectory

The following is a complete script for programming a Trajectory graphic. It computes a somewhat circular distribution of points and also assigns a bubble value. For this to work, the trajectory graphic must have been made with a point tag marker.

```
/* Declarations */

double cos(double a);
double sin(double a);

@@class() Trajectory:Object

@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (void)emptyData;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (void)appendXValue:(double)xValue yValue:(double)yValue;
@@method(public, instance) (void)setCurveRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)appendBubbleValue:(double)aValue;
@@method(public, instance) (void)appendMarkerRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)release;

@@end

/* Execution block */

{
id myTrajectory;
int ii;
double xValue, yValue;
unsigned animationCount;
double red, green, blue;

myTrajectory = [[Trajectory alloc] init];

animationCount = [myTrajectory animationCount];

printf("animationCount: %d\n", animationCount);

/*
Empty the data and then append new data.
*/

[myTrajectory emptyData];

for(ii = 0; ii < 20; ii++)
{
red = (animationCount % 10) / 10.0;
green = (animationCount % 30) / 30.0;
blue = (ii % 20) / 20.0;
xValue = cos(ii * .02) + red * sin(ii * .01);
yValue = sin(ii * .02);

[myTrajectory appendXValue:xValue yValue:yValue];
[myTrajectory appendMarkerRed:red green:green blue:blue alpha:1.0];
[myTrajectory appendBubbleValue:(ii * 1.0)];
}

[myTrajectory release];

}
```

The general API is define in the section Graphic. The following is API description specific to the Trajectory graphic.

```
@@method(public, instance) (void)appendXValue:(double)xValue yValue:(double)yValue;
```

Call like this:

```
[myTrajectory appendXValue:xValue yValue:yValue];
```

Appends the x and y values to the list of data points for the graphic. The x and y values forms a 2D point. Each value must be of type double.

```
@@method(public, instance) (void)appendMarkerRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
```

Call like this:

```
[myTrajectory appendMarkerRed:0.5 green:0.4 blue:1.0 alpha:1.0];
```

That appends the marker color to the red, green, blue and alpha values of 0.5, 0.4, 1.0 and 1.0 respectively. Those values must be between 0.0 and 1.0. An alpha of 0.0 is transparent while 1.0 is completely opaque. Each argument must be a number literal or a variable (or expression) of type double. Note that this call must accompany a appendXValue:xValue yValue: call in order to synchronize the parameters that depend upon sequence index.

```
@@method(public, instance) (void)appendSegmentRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
```

Call like this:

```
[myTrajectory appendSegmentRed:0.5 green:0.4 blue:1.0 alpha:1.0];
```

That appends the segment color to the red, green, blue and alpha values of 0.5, 0.4, 1.0 and 1.0 respectively. Those values must be between 0.0 and 1.0. An alpha of 0.0 is transparent while 1.0 is completely opaque. Each argument must be a number literal or a variable (or expression) of type double. Note that this call must accompany a appendXValue:xValue yValue: call in order to synchronize the parameters that depend upon sequence index.

```
@@method(public, instance) (void)appendBubbleValue:(double)aValue;
```

Call like this:

```
[myTrajectory appendBubbleValue:aValue];
```

Appends aValue to the list of bubble values for the graphic. aValue must be of type double.

```
@@method(public, instance) (void)emptyData;
```

Call like this:

```
[myTrajectory emptyData];
```

Removes (empties) all data from the graphic. Call this right before adding new data points.

The following is a complete script for programming a 2D Point Map Plot (Heat Map Plot). It loads a wavy heat map made from 10,000 z-values on a square grid.

```
/* Declarations */

double cos(double a);
double sin(double a);

@@class() PointMap:Object

@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (void)emptyData;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (void) setGridXLength:(unsigned)xLength xMinimum:(double)xMinimum xMaximum:
(double)xMaximum yLength:(unsigned)yLength yMinimum:(double)yMinimum yMaximum:(double)yMaximum;
@@method(public, instance) (void) appendValue:(double)aValue;
@@method(public, instance) (void)release;

@@end

/* Execution block */

{
id myPointMap;
int ix, iy;
double xValue, yValue, aValue;
unsigned animationCount;

myPointMap = [[PointMap alloc] init];

animationCount = [myPointMap animationCount];

printf("animationCount: %d\n", animationCount);

/*
Empty the data and then append new data.
*/

[myPointMap emptyData];

[myPointMap setGridXLength:100 xMinimum:0.0 xMaximum:10.0 yLength:100 yMinimum:0.0 yMaximum:10.0];

for(iy = 0; iy < 100; iy++)
{
yValue = iy * 0.1;

for(ix = 0; ix < 100; ix++)
{
xValue = ix * 0.1;
aValue = 5.0 * cos(xValue) + 5.0 * sin(yValue);
[myPointMap appendValue:aValue];
}
}

[myPointMap release];

}
```

The general API is define in the section Graphic. The following is API description specific to the Point Map Plot.

```
@@method(public, instance) (void) setGridXLength:(unsigned)xLength xMinimum:(double)xMinimum xMaximum:
(double)xMaximum yLength:(unsigned)yLength yMinimum:(double)yMinimum yMaximum:(double)yMaximum;
```

Call like this:

```
[myPointMap setGridXLength:100 xMinimum:0.0 xMaximum:10.0 yLength:100 yMinimum:0.0 yMaximum:10.0];
```

Sets the grid parameters. Since the grid is rectangular and uniform these are the only parameters needed to specify the grid.

```
@@method(public, instance) (void) appendValue:(double)aValue;
```

Call like this:

```
[myPointMap appendValue:aValue];
```

Appends aValue to the list of values. Each value must be of type double. The number of values appended should equal the grid x-length times y-length.

```
@@method(public, instance) (void) appendAmplitude:(double)anAmplitude angle:(double)anAngle;
```

Call like this:

```
[myPointMap appendAmplitude:anAmplitude angle:anAngle];
```

Appends anAmplitude to the array of data values and anAngle to the array of angle values. Each value must be of type double and anAngle must be in units of radians. The number of amplitudes and angles appended should each independently equal the grid x-length times y-length. Notice how the unit of angle is in radians but when entering angles in the user interface the units are in degrees. In order to see the angle values (vectors) the stroke unit must be turned on in the point map graphic.

```
@@method(public, instance) (void)emptyData;
```

Call like this:

```
[myPointMap emptyData];
```

Removes (empties) all data from the graphic. Call this right before adding new data points.

Graph IDE ► Programming ► Perspective Graph

The following is API description specific to the Perspective Graph graphic.

```
@@method(public, instance) (void)autoscale;
```

Call like this:

```
[aGraph autoscale];
```

Autoscales the receiver.

```
@@method(public, instance) (void)focus;
```

Call like this:

```
[aGraph focus];
```

Sets the focused layer to the receiver's foreground data layer. After this call any graphics that are inserted or appended are done so to the graph's foreground data layer.

```
@@method(public, instance) (void) rotateToPhi:(double)phiAngle theta:(double)thetaAngle psi:(double)psiAngle;
```

Call like this:

```
[aGraph rotateToPhi:phi theta:theta psi:psi];
```

Rotates the receiver to the phi, theta and psi angles which are yaw, pitch and roll angles respectively in units of radians.

```
@@method(public, instance) (void)unfocus;
```

Call like this:

```
[aGraph unfocus];
```

Sets the focused layer to the layer that the receiver is in. After this call any graphics that are inserted or appended are done so to the same layer as the graph.

```
@@method(public, instance) (void)updateMainTitleToUFT8String:(const char *)aUTF8String;
```

Call like this:

```
[aGraph updateMainTitleToUFT8String:"My Title"];
```

Sets the main title of the graph.

```
@@method(public, instance) (void)updateXTitleToUFT8String:(const char *)aUTF8String;
```

Call like this:

```
[aGraph updateXTitleToUFT8String:"My X Title"];
```

Sets the x-axis title of the graph.

```
@@method(public, instance) (void)updateYTitleToUFT8String:(const char *)aUTF8String;
```

Call like this:

```
[aGraph updateYTitleToUFT8String:"My Y Title"];
```

Sets the y-axis title of the graph.

```
@@method(public, instance) (void)updateZTitleToUFT8String:(const char *)aUTF8String;
```

Call like this:

```
[aGraph updateZTitleToUFT8String:"My Z Title"];
```

Sets the z-axis title of the graph.

The following is a complete script for programming a 3D Scatter Plot. It loads in a spiral of data and rotates the graph.

```
/* Declarations */

double cos(double a);
double sin(double a);

@@class() PerspectiveScatter:Object

@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (void)emptyData;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (void)appendXValue:(double)xValue yValue:(double)yValue zValue:(double)zValue;
@@method(public, instance) (void)setCurveRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)setInteriorRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void) rotateToPhi:(double)phiAngle theta:(double)thetaAngle psi:(double)psiAngle;
@@method(public, instance) (void)release;

@@end

/* Execution block */

{
id myScatter;
int ii;
double tValue, xValue, yValue, zValue;
unsigned animationCount;
double red;
double phiAngle;

myScatter = [[PerspectiveScatter alloc] init];

animationCount = [myScatter animationCount];

printf("animationCount: %d\n", animationCount);

/*
Empty the data and then append new data.
*/

[myScatter emptyData];

for(ii = 0; ii < 500; ii++)
{
tValue = 0.04 * ii + animationCount * 0.5;
xValue = 5.0 * cos(tValue) + 5.0;
yValue = 5.0 * sin(tValue) + 5.0;
zValue = tValue / 2.0;
[myScatter appendXValue:xValue yValue:yValue zValue:zValue];
}

red = (animationCount % 10) / 10.0;
phiAngle = animationCount * 0.1;

[myScatter setCurveRed:red green:0.0 blue:0.0 alpha:1.0];

[myScatter rotateToPhi:phiAngle theta:0.0 psi:0.0];

[myScatter release];

}
```

The general API is define in the section Graphic. The following is API description specific to the 3D Scatter Plot.

```
@@method(public, instance) (void)appendXValue:(double)xValue yValue:(double)yValue zValue:(double)zValue;
```

Call like this:

```
[myScatter appendXValue:xValue yValue:yValue zValue:zValue];
```

Appends the x, y and z values to the list of data points for the graphic. The x, y and z values forms a 3D point. Each value must be of type double.

```
@@method(public, instance) (void)appendMarkerRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
```

Call like this:

```
[myScatter appendMarkerRed:0.5 green:0.4 blue:1.0 alpha:1.0];
```

That appends the marker color to the red, green, blue and alpha values of 0.5, 0.4, 1.0 and 1.0 respectively. Those values must be between 0.0 and 1.0. An alpha of 0.0 is transparent while 1.0 is completely opaque. Each argument must be a number literal or a variable (or expression) of type double. Note that this call must accompany a appendXValue:xValue yValue: call in order to synchronize the parameters that depend upon sequence index.

```
@@method(public, instance) (void)appendSegmentRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
```

Call like this:

```
[myScatter appendSegmentRed:0.5 green:0.4 blue:1.0 alpha:1.0];
```

That appends the segment color to the red, green, blue and alpha values of 0.5, 0.4, 1.0 and 1.0 respectively. Those values must be between 0.0 and 1.0. An alpha of 0.0 is transparent while 1.0 is completely opaque. Each argument must be a number literal or a variable (or expression) of type double. Note that this call must accompany a appendXValue:xValue yValue: call in order to synchronize the parameters that depend upon sequence index. In order for this to take effect, the scatter stroke type needs to be set to other than none. By setting some segment colors to have an alpha of 0.0, the stroke appears discontiguous. Using this method permits the 3D scatter plot to appear as a trajectory plot and when alpha is set to 0.0 for appropriate segments the trajectories appear as multiple 3D trajectories in space.

```
@@method(public, instance) (void)emptyData;
```

Call like this:

```
[myScatter emptyData];
```

Removes (empties) all data from the graphic. Call this right before adding new data points.

```
@@method(public, instance) (void) rotateToPhi:(double)phiAngle theta:(double)thetaAngle psi:(double)psiAngle;
```

Call like this:

```
[myScatter rotateToPhi:phi theta:theta psi:psi];
```

Rotates the scatter plot to the phi, theta and psi angles which are yaw, pitch and roll angles respectively in units of radians.

Graph IDE ▶ Programming ▶ Perspective Surface

The following is a complete script for programming a 3D Surface Plot. It loads a wavy surface made from 10,000 z-values on a square grid.

```
/* Declarations */

double cos(double a);
double sin(double a);

@@class() PerspectiveSurface:Object

@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (void)emptyData;
@@method(public, instance) (unsigned)animationCount;
@@method(public, instance) (void) setGridXLength:(unsigned)xLength xMinimum:(double)xMinimum xMaximum:
(double)xMaximum yLength:(unsigned)yLength yMinimum:(double)yMinimum yMaximum:(double)yMaximum;
@@method(public, instance) (void) appendValue:(double)aValue;
@@method(public, instance) (void)setCurveRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void)setInteriorRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
@@method(public, instance) (void) rotateToPhi:(double)phiAngle theta:(double)thetaAngle psi:(double)psiAngle;
@@method(public, instance) (void)release;

@@end

/* Execution block */

{
id mySurface;
int ix, iy;
double xValue, yValue, zValue;
unsigned animationCount;
double phiAngle;

mySurface = [[PerspectiveSurface alloc] init];

animationCount = [mySurface animationCount];

printf("animationCount: %d\n", animationCount);

/*
Empty the data and then append new data.
*/

[mySurface emptyData];

[mySurface setGridXLength:100 xMinimum:0.0 xMaximum:10.0 yLength:100 yMinimum:0.0 yMaximum:10.0];

for(iy = 0; iy < 100; iy++)
{
yValue = iy * 0.1;

for(ix = 0; ix < 100; ix++)
{
xValue = ix * 0.1;
zValue = 5.0 * cos(xValue) + 5.0 * sin(yValue);
[mySurface appendValue:zValue];
}
}

phiAngle = animationCount * 0.1;

[mySurface rotateToPhi:phiAngle theta:0.1 psi:0.0];

[mySurface release];

}
```

The general API is define in the section Graphic. The following is API description specific to the 3D Surface Plot.

```
@@method(public, instance) (void) setGridXLength:(unsigned)xLength xMinimum:(double)xMinimum xMaximum:
(double)xMaximum yLength:(unsigned)yLength yMinimum:(double)yMinimum yMaximum:(double)yMaximum;
```

Call like this:

```
[mySurface setGridXLength:100 xMinimum:0.0 xMaximum:10.0 yLength:100 yMinimum:0.0 yMaximum:10.0];
```

Sets the grid parameters. Since the grid is rectangular and uniform these are the only parameters needed to specify the grid.

```
@@method(public, instance) (void) appendValue:(double)aValue;
```

Call like this:

```
[mySurface appendValue:zValue];
```

Appends zValue to the list of surface values. Each value must be of type double. The number of zValues appended should equal the grid x-length time y-length.

```
@@method(public, instance) (void)emptyData;
```

Call like this:

```
[mySurface emptyData];
```

Removes (empties) all data from the graphic. Call this right before adding new data points.

```
@@method(public, instance) (void)appendFalseRed:(double)red green:(double)green blue:(double)blue alpha:
(double)alpha;
```

Call like this:

```
[mySurface appendFalseRed:0.5 green:0.4 blue:1.0 alpha:1.0];
```

That appends the half-cell color to the red, green, blue and alpha values of 0.5, 0.4, 1.0 and 1.0 respectively. Those values must be between 0.0 and 1.0. An alpha of 0.0 is transparent while 1.0 is completely opaque. Each argument must be a number literal or a variable (or expression) of type double. Note that this call must accompany a appendXValue:xValue yValue: call in order to synchronize the parameters that depend upon sequence index.

Note: A cell is defined by four adjacent points on the x-y grid. That cell is divided by a diagonal thus making two triangular regions. You should call this method twice with the same color to fill in the entire rectangular cell when making such false color maps.

```
@@method(public, instance) (void) rotateToPhi:(double)phiAngle theta:(double)thetaAngle psi:(double)psiAngle;
```

Call like this:

```
[mySurface rotateToPhi:phi theta:theta psi:psi];
```

Rotates the surface plot to the phi, theta and psi angles which are yaw, pitch and roll angles respectively in units of radians.

The following is a complete script for programming an Adapter graphic.

```
/*
Adapter script for a segmented control.
*/

@@class() Widestring:Object

@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (id)value;
@@method(public, instance) (void)copyUTF8String:(const char *)aUTF8String;
@@method(public, instance) (void)release;

@@end

@@class() GraphicAdapter:Object

@@method(public, class) (id)stored;
@@method(public, instance) (unsigned)executionCount;
@@method(public, instance) (id)value;

@@end

@@class() SAI_Segmented_Control:Object

@@method(public, instance) (void)set_SAI_number_of_segments:(unsigned)a_value;
@@method(public, instance) (unsigned)get_SAI_number_of_segments
@@method(public, instance) (void)set_SAI_segment_style_type:(unsigned)a_value;
@@method(public, instance) (unsigned)get_SAI_segment_style_type

@@method(public, instance) (void)SAI_update_segment_index:(unsigned)an_index to_widestring:(id)a_widestring

@@end

/* Execution block */

{
id myAdapter;
unsigned executionCount;

myAdapter = [GraphicAdapter stored];
executionCount = [myAdapter executionCount];

if(executionCount == 1U)
{
id myAdapterValue;
id myString;

myAdapterValue = [myAdapter value];
myString = [[Widestring alloc] init];

[myAdapterValue set_SAI_number_of_segments:3U];

[myString copyUTF8String:"One"];
[myAdapterValue SAI_update_segment_index:0U to_widestring:[myString value]];
[myString copyUTF8String:"Two"];
[myAdapterValue SAI_update_segment_index:1U to_widestring:[myString value]];
[myString copyUTF8String:"Three"];
[myAdapterValue SAI_update_segment_index:2U to_widestring:[myString value]];

[myAdapter registerWithName:"a_name"];
}

}
```

The general API is define in the section Graphic.

API specific to the adapter is defined below.

```
@@method(public, instance) (id)value;
```

Call like this:

```
myValue = [myAdapter value];
```

Returns the native view object. Once you have a handle to that view object you may call upon it as needed. Probably the best thing to do is to make remedial changes to it in the script and then register the adapter with the controller to set it up further.

Graph IDE ► Programming ► Plugin

Using plugins is a two step process. First you write and compile the plugin using Xcode and then you load it in. Loading it is simple, just select the Graph IDE menu item Tools ► Programming ► Load Plugin... and open the plugin you make. The rest of this section gives instructions on making the plugin.

Note: Only the Mac version supports Plugins. For other platforms your code must be linked directly into the Graph IDE project. See GitHub/VVI for additional information.

Premade Xcode Project

The fastest way to start with plugins is to use a premade one. Plugin resources are available from these links:

Link	Description Of Resource
ExamplePlugin.zip	The compressed ExamplePlugin project located on your disk within this manual.
ExamplePlugin.zip	The compressed ExamplePlugin project located at the vvidget.org web site.
Xcode	The Xcode application on the Mac App Store.

Download the ExamplePlugin zip file, uncompress, launch the Xcode project and click Run to make the plugin. The framework links and source code are already setup and ready to go. Eventually you may wish to make your own plugin project from scratch. For that purpose read the section below.

Xcode Instructions

The following gives the steps for making a Xcode plugin that can be utilized for programming graphics.

- Launch Xcode.
- In the Xcode menu select File ► Project...
- In the resulting panel select OS X ► Framework & Library ► Bundle. Then click the Next button.
- In the resulting panel fill in the required entries. Make sure to choose a Cocoa Framework and do not use Automatic Reference Counting. Then click the Next button.
- In the resulting save panel navigate to where you wish to save the project and click the Create button.
- In targetS ► Build Settings change the Wrapper Extension from bundle to vviplugin.
- In targetS ► Build Phases expand Link Binary With Libraries click the plus (add) button. In the resulting panel click the Add Other... button. In the resulting Open panel navigate to the Vvidget Frameworks (see the bullet below) and add all of them.
- Adding the Vvidget Frameworks employs a trick. Normally programming frameworks are installed on a development system via a SDK installer which is complex, overly burdensome and in the case of programming a plugin unnecessary. That is because the necessary Frameworks are already on your computer and are contained in the Graph IDE application wrapper. To expose those frameworks do the following. First (important!) move Graph IDE to a permanent location (for example: /Applications). By making it permanent you ensure that the path to the Frameworks is fixed. That path is encoded into the plugin project and can be changed but it is easier at first to make it fixed. Once you have placed Graph IDE where you like it then, using Finder, navigate to Graph IDE. Then control click on Graph IDE and choose Show Package Contents. Then navigate to the Contents/Frameworks folder. Select the Frameworks folder and drag it to the Xcode Open Panel. Within that Xcode panel, select all of the Frameworks and click the Open button.
- Doing the above places an entry into the Framework Search Paths. If you move the project or Graph IDE then modify the Framework Search Path entry appropriately. Specifically: You may wish to make the path an absolute path so you can move the plugin project as desired. Any which way, you will need to take appropriate measures if you move Graph IDE or the plugin project, however the measures are usual programming issues.
- Tidy up the project by dragging (moving) the framework references (within the Navigator view) to the Frameworks group.
- In the Xcode menu select File ► New ► File...
- In the resulting panel select OS X ► Cocoa ► Objective-C class. Then click the Next button. Give it the Class name MyFunction and Subclass of VVPUBLIC_Function. Then click the Next button. In the resulting panel make sure the target is selected and click the Create button.
- Add some source code to the MyFunction.h file, such as:

```
#import <Vvidget_GG/VVPUBLIC_Function.h>

@interface MyFunction : VVPUBLIC_Function

- (id)init;
- (void)doAllTheWork;

@end
```

- Add some source code to the MyFunction.m file, such as:

```
@implementation MyFunction

- (id)init
{
   self = [super init];

   NSLog(@"-[MyFunction init]");

   return self;
}

- (void)doAllTheWork
{

   NSLog(@"-[MyFunction doAllTheWork]");

   int ii;
   double xValue, yValue;
   unsigned animationCount;
   unsigned recursionCount;
   double red;

   animationCount = [self animationCount];
   recursionCount = [self recursionCount];

   [self emptyData];

    for(ii = 0; ii < 500; ii++)
   {
      xValue = 0.01 * ii + animationCount * 0.5;
      yValue = cos(xValue) + recursionCount;
      [self appendXValue:xValue yValue:yValue];
   }

   red = (animationCount % 10) / 10.0;

   [self setCurveRed:red green:0.0 blue:0.0 alpha:1.0];

   if(recursionCount < 5)
   {
      [self recur];
   }

   return;
}

@end
```

- Click the Run button to build the plugin.
- In the Navigator view expand the Products group, select the plugin, control-click it and choose Show in Finder. In the resulting Finder window drag the plugin to where you want it.
- The plugin is now available for use within Graph IDE.

Calling the Plugin

In the source code above the Function object class was subclassed. To instantiate an object of your own subclass allocate it in a call like `[MyFunction alloc]` instead of `[Function alloc]`. Notice how you subclassed `VVPUBLIC_Function`. As a matter of convenience to programming, the parser strips the prefix `VVPUBLIC_` from the class name when appropriate however in Xcode the prefix is not stripped and must be included when subclassing a known class.

The instructions above detail how to make the plugin. The plugin is applicable to programming the Function graphic. You can call upon the object of your own class and its methods using script code such as the following.

Script without saving state

```
@@class() MyFunction:Object

@@method(public, class) (id)alloc;
@@method(public, instance) (id)init;
@@method(public, instance) (void)doAllTheWork;
@@method(public, instance) (void)release;

@@end

{
id myFunction;
```

```
    myFunction = [[MyFunction alloc] init];

    [myFunction doAllTheWork];

    [myFunction release];

    }
```

If your plugin needs to save state within the Graph IDE document (see the plugin example project) then the script is written as follows.

Script that permits the instance of MyFunction to save state in the Graph IDE document.

```
@@class() MyFunction:Object

@@method(public, class) (id)stored;
@@method(public, instance) (void)doAllTheWork;

@@end

{
id myFunction;

myFunction = [MyFunction stored];

[myFunction doAllTheWork];

}
```

The method doAllTheWork can implement most anything such as DSP processing, FEM modeling, stock data feed retrievals, statistical algorithms, etc.

The ExamplePlugin given at the links above implements a few other features such as saving a MyFunction instance state. It also contains some programming notes and comments. Please email support@vvi.com if you have difficulty obtaining the plugin or have a plugin question.

Table Of Contents

Graph IDE ► Programming ► Advisory

Programming Graph IDE consists of running scripts that utilize a built-in script engine. That script engine can access methods in plugins that you write. In addition, the script engine can access many Cocoa and system classes, objects and associated methods as well as traditional functions. The script engine (called SAM for State AutoMation) is also a general purpose code parser. As a result of the complex nature of programming there are issues that you should be aware of. Those issues are listed as follows.

- Make sure to comprehensively specify the declaration section of the scripts. Without a declaration, the script engine may produce unintended results.
- The script engine implements comprehensive error checking to prevent known errors and cascading of errors. When you write a script then click the Execute button and check the Error tab for any errors. If the script does not execute then check it carefully for spelling and syntax errors. If you encounter an unexpected result then please email support@vvi.com with a bug report.
- The Program inspector editor is just a text view, it is not an IDE (Interactive Development Environment) editor. You may wish to first paste a pre-made script into a text editor (or Xcode or other IDE), modify it to your needs, paste it into the Program inspector editor source code text view and then click the Execute button to make sure it parses and alters the associated graphic. Once the code is confirmed to execute then save the document as desired.
- You can bind the script to any accessible object or class. If you do that then make sure to fully declare the method API in a class block. Because you are calling into code sections that are not known in advance you may encounter bugs and unpredictable behavior.
- The script engine parses and binds in a pre-stage separate from execution. That means it is fast to execute, but still not as fast as writing a Xcode plugin and compiling to assembly language with optimizations such as inlining, etc. At some point, you may wish to use a Plugin and call a single method defined in that plugin to execute the algorithm that would normally be in the script. There are several reasons to do that including optimization and familiarity with the Xcode IDE.
- If you have a preexisting code base and library that exists in Xcode and wish to use that then a plugin is the way to go. You can wrap your code into a single method which can then be called from a script.
- Recursion is implemented with the `recur` method. Of course recursion is a very powerful feature, but if you do not check for an end to the recursion (using `recursionCount`) then the recursion will be infinite and that will cause the usual problems (running out of memory, etc.) so make sure to terminate recursion appropriately by using a conditional.
- If you save the document while animating then when the document is subsequently opened it will start animating right away. If a script is not written correctly then the document may crash. That means the document will be inaccessible. As a precaution, opening a document that is set to animate will present a sheet with the option to animate or not. Choosing not to animate will give you an opportunity to fix the script.
- If you move a plugin (using the Finder for example) then it will not load and you will have to reload the plugin based upon its new location.
- The script parser is set to not warn about syntax errors nor is it set to format syntax. As a result, if you write code incorrectly then that code will silently fail to parse.
- Make sure to close blocks correctly (pair brackets such as curly brackets and parentheses).

A few issues are listed above. If you have a suggestion to make programming easier then please email support@vvi.com.

Graph IDE ► Custom Application

Graph IDE is a powerful application rich in graphic editing, creation and programming facilities. It is but one class of data visualization application. A different and very important class of data visualization application is one that presents data and contains only simple controls such as those that alter parameters for a process or model. For argument sake, lets call the later class a custom application. This section describes how to efficiently make a custom application by leveraging existing Graph IDE assets. The basic steps are as follows:

- Programming Graph IDE for the automatic importation of data and animation of graphics within the Graphic View.
- Write a separate custom application that requires Graph IDE type results. The custom application is written in Xcode or similar IDE.
- Load the Graphic View into a custom application. The Graphic View was pre-programmed to run seamlessly in the custom application so no further effort is needed.

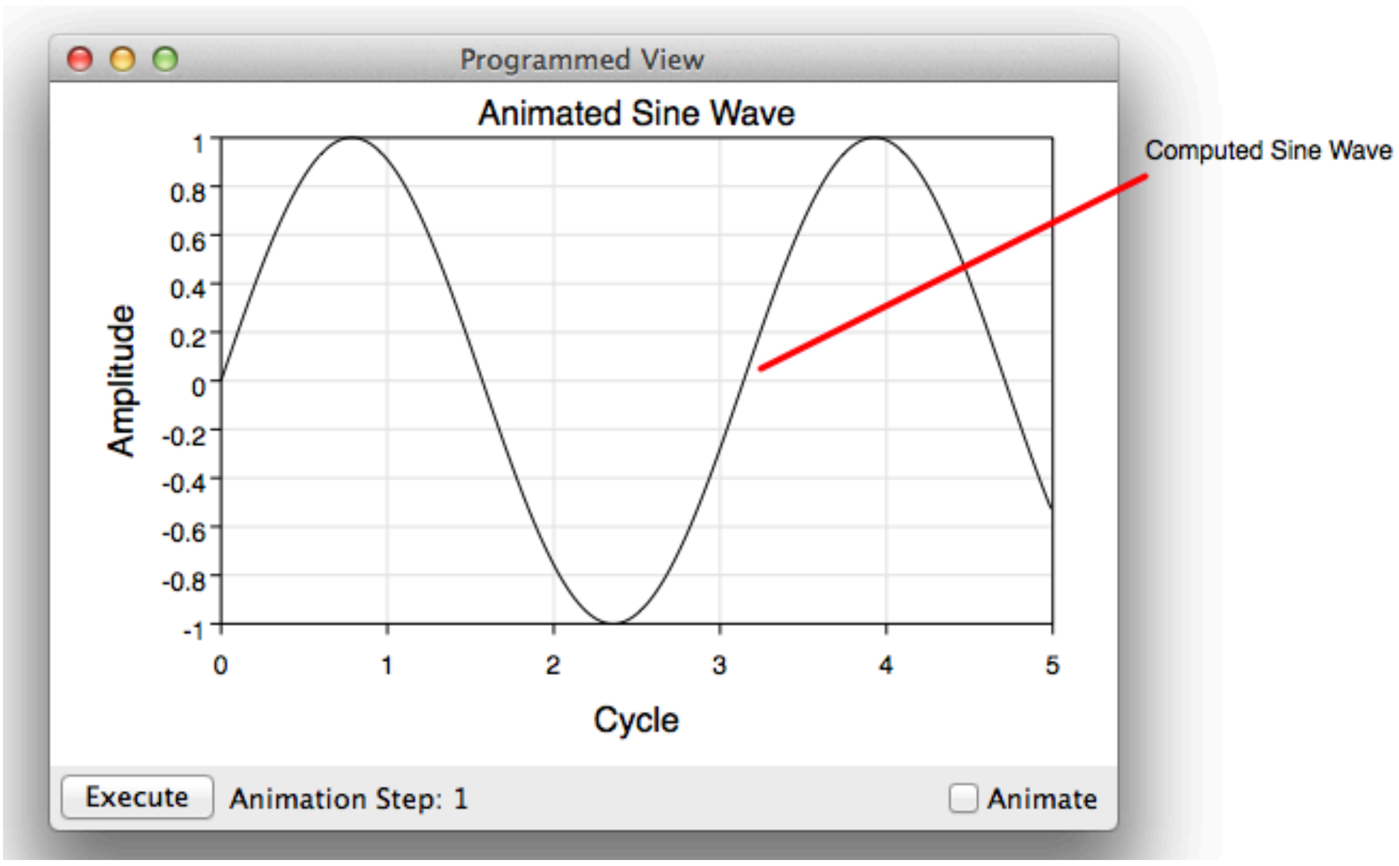
This represents the pinnacle of Graph IDE's evolution where Graph IDE becomes more of a visual IDE (Interactive Development Environment) for the creation of data visualization applications rather than a data visualization documentation creation application. Either way, the emphases is on empowering the author of either a data visualization document or a data visualization custom application.

The following is a brief list of Custom Application sections. Although the list is brief, the capabilities for custom application development are extensive.

Section	Description
Container View	Describes how to load a Graphic View into a custom application without programming.
Controller	Describes the controller method that is used to register graphic states.
Distribution	Describes what is needed to distribute applications to other people.
Document	Describes how to work with a Graph IDE document and how to load its Graphic View into a custom application.
Event Qualifier	Describes the event qualifier (hit detection) programming API.
GraphStrings	Describes the GraphStrings custom application, which shows how to implement callback methods to alter graph labels.
HandsOff	Describes the HandsOff custom application, which is a OS X (Mac) application.
HitButton	Describes the HitButton custom application, which is a OS X (Mac) application that exhibits hit-detection behaviors.
SineTable	Describes the SineTable custom application, which is a OS X (Mac) application.
SineWave	Describes the SineWave custom application, which is a OS X (Mac) application.
SwiftSineWave	Describes the SwiftSineWave custom application, which is virtually the same as the SineWave project except written with a Swift programming language component.
Surface	Describes the Surface custom application, which is a OS X (Mac) application.

Graph IDE ► Custom Application ► SineWave

This section describes how to write a custom application called SineWave. The user interface is diagrammed below.



The fastest way to write a custom application is to download the SineWave example project, compile it and modify it for your needs. SineWave resources are available from these links:

Link	Description Of Resource
SineWave.zip	The compressed SineWave project located on your disk within this manual.
SineWave.zip	The compressed SineWave project located at the vvidget.org web site.
Xcode	The Xcode application on the Mac App Store.

The following explains the essential steps of the SineWave project.

The graph and white portion in the figure above is a Graphic View that was pre-programmed using the Programming facilities of Graph IDE. The Execute button and Animate switch are programmed in the custom application. The steps are as follows.

The following code splice shows how to load a Graphic View into SineWave and how to execute and animate the graphic view program. This is a simple application and the loading is done in the Application Delegate. A more comprehensive custom application would probably do the loading in a document controller class.

```
@implementation AppDelegate

- (void)applicationDidFinishLaunching:(NSNotification *)aNotification
{
/* The graphic view is loaded from the document MyDocument.vvibook, which is a resource of the custom
application, when the application finishes launching. */

    NSString *aPath;

    aPath = [[NSBundle mainBundle] pathForResource:@"MyDocument" ofType:@"vvibook"];

    theDocument = [[VVPUBLIC_GraphicDocument alloc] init];
    programmedView = [theDocument viewForDocumentPath:aPath];

    [programmedView setFrame:[_switchView frame]];
    [programmedView setAutoresizingMask:[_switchView autoresizingMask]];

    [[_switchView superview] replaceSubview:_switchView with:programmedView];

    [_window makeKeyAndOrderFront:nil];

}

- (void)applicationWillTerminate:(NSNotification *)aNotification
```

```
{
/* When the application is terminated then stop the document animation and release the document */

    [theDocument stopAnimation];
    [theDocument release];
}

- (IBAction)performAnimation:(id)sender
{
/* This is the action of the switch and turns animation on or off */

    if([sender state])
    {
        [theDocument startAnimation];
    }
    else
    {
        [theDocument stopAnimation];
    }
}

- (IBAction)performExecute:(id)sender
{
/* This is the action of the Execute button and performs one step of the animation. That step performs one
acquisition of the data. */

    [theDocument performAnimationStep];
    [programmedView display];
}
```

The following is the program that the curve executes. It is applied to the Function in its Program inspector editor. It calls into the method called `calculateCurve` of an instance of the `MyFunction` class that you write (see the next code splice).

```
@@class() MyFunction:Object

@@method(public, class) (id)stored;
@@method(public, instance) (void)calculateCurve;

@@end

{
id myFunction;

myFunction = [MyFunction stored];

[myFunction calculateCurve];
}
```

The following is implementation the `MyFunction` class. The `calculateCurve` method computes the sine curve and assigns data points and attributes using the API in the Function programming section. Your implementation may do something more complex and reference a legacy codebase.

```
@implementation MyFunction

- (id)init
{

    if(self = [super init])
    {
        wavePeriod = 1.0;
        periodDelta = 1.0;
    }

    return self;
}

- (void)calculateCurve
{
    int ii;
    double xValue, yValue;
    unsigned animationCount;
    AppDelegate *appDelegate;
    NSTextField *infoTextField;
```

```
        animationCount = [self animationCount];

        appDelegate = [[NSApplication sharedApplication] delegate];
        infoTextField = [appDelegate infoTextField];

        [infoTextField setStringValue:[NSString stringWithFormat:@"Animation Step: %d", animationCount]];

        [self emptyData];

        if(wavePeriod > 20.0)
        {
            periodDelta = -1.0;
        }
        else if(wavePeriod < 2.0)
        {
            periodDelta = 1.0;
        }

        wavePeriod += periodDelta;

        for(ii = 0; ii < 500; ii++)
        {
            xValue = 0.01 * ii ;
            yValue = sin(xValue * wavePeriod);
            [self appendXValue:xValue yValue:yValue];
        }


    }
```

When making the Xcode project follow the steps in the Plugin section, but with these important distinctions:

- When making a new project choose Cocoa Application.
- Make a new Add Copy Files Build Phase with Destination Frameworks and drag the Vvidget framework references from the project navigator to the Copy Files build phase list.

The SineWave.zip project is already setup with the proper references and build phases so that might be a good starting point.

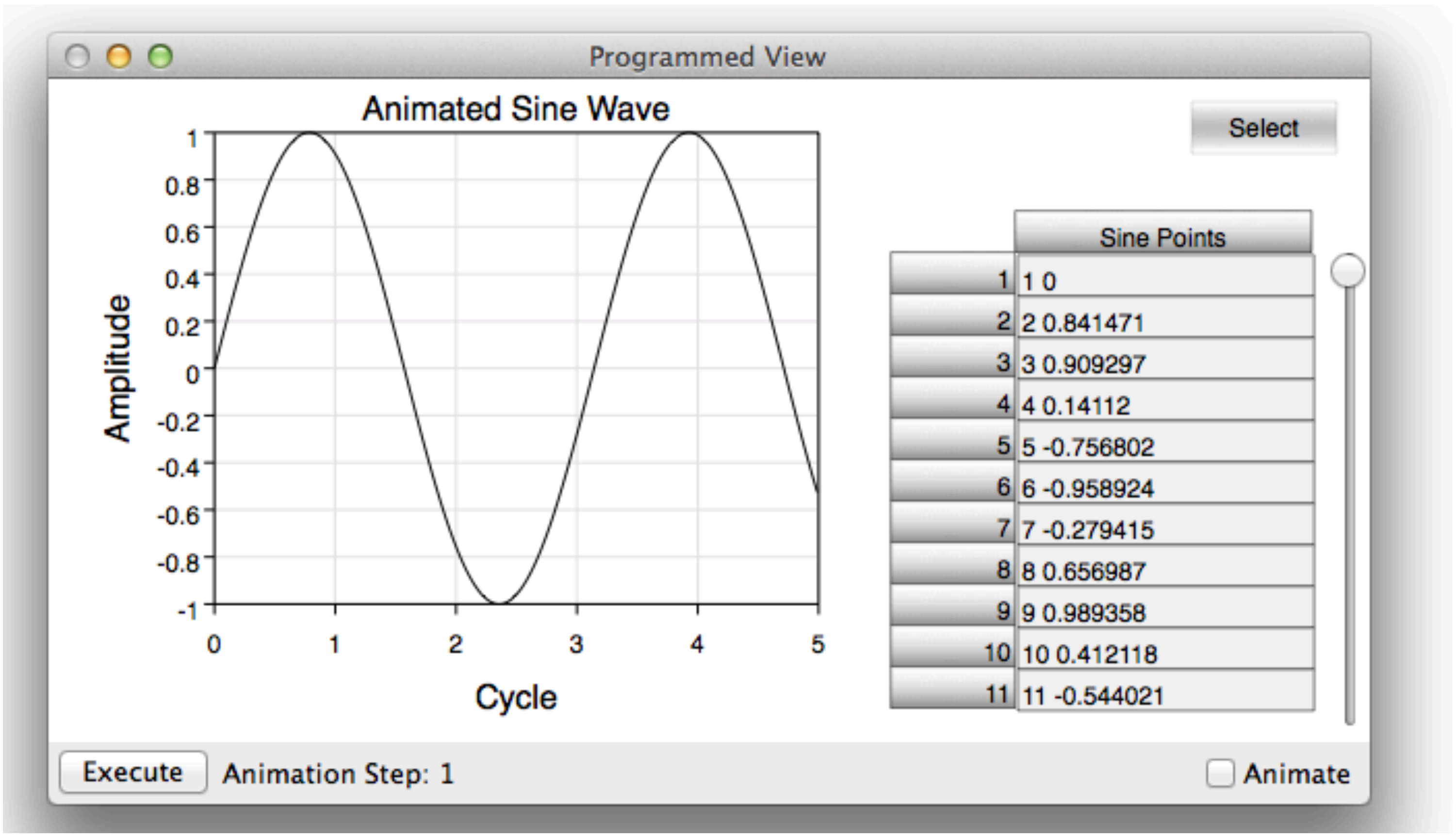
Graph IDE ► Custom Application ► SwiftSineWave

This section describes how to write a custom application called SwiftSineWave. The project is virtually the same as the SineWave project written in Objective-C except that the App Delegate, which is the main controller, is written in the Swift language and bridge files are added to fully communicate between Swift and Objective-C language files.

The fastest way to write a custom application is to download the SwiftSineWave example project, compile it and modify it for your needs. SwiftSineWave resources are available from these links:

Link	Description Of Resource
SwiftSineWave.zip	The compressed SwiftSineWave project located on your disk within this manual.
SwiftSineWave.zip	The compressed SwiftSineWave project located at the vvidget.org web site.
Xcode	The Xcode application on the Mac App Store.

This section describes how to write a custom application called SineTable. The user interface is diagrammed below.



	Sine Points
1	1 0
2	2 0.841471
3	3 0.909297
4	4 0.14112
5	5 -0.756802
6	6 -0.958924
7	7 -0.279415
8	8 0.656987
9	9 0.989358
10	10 0.412118
11	11 -0.544021

The fastest way to write a custom application is to download the SineTable example project, compile it and modify it for your needs. SineTable resources are available from these links:

Link	Description Of Resource
SineTable.zip/td>	The compressed SineTable project located on your disk within this manual.
SineTable.zip/td>	The compressed SineTable project located at the vvidget.org web site.
Xcode/td>	The Xcode application on the Mac App Store.

SineTable shows how to:

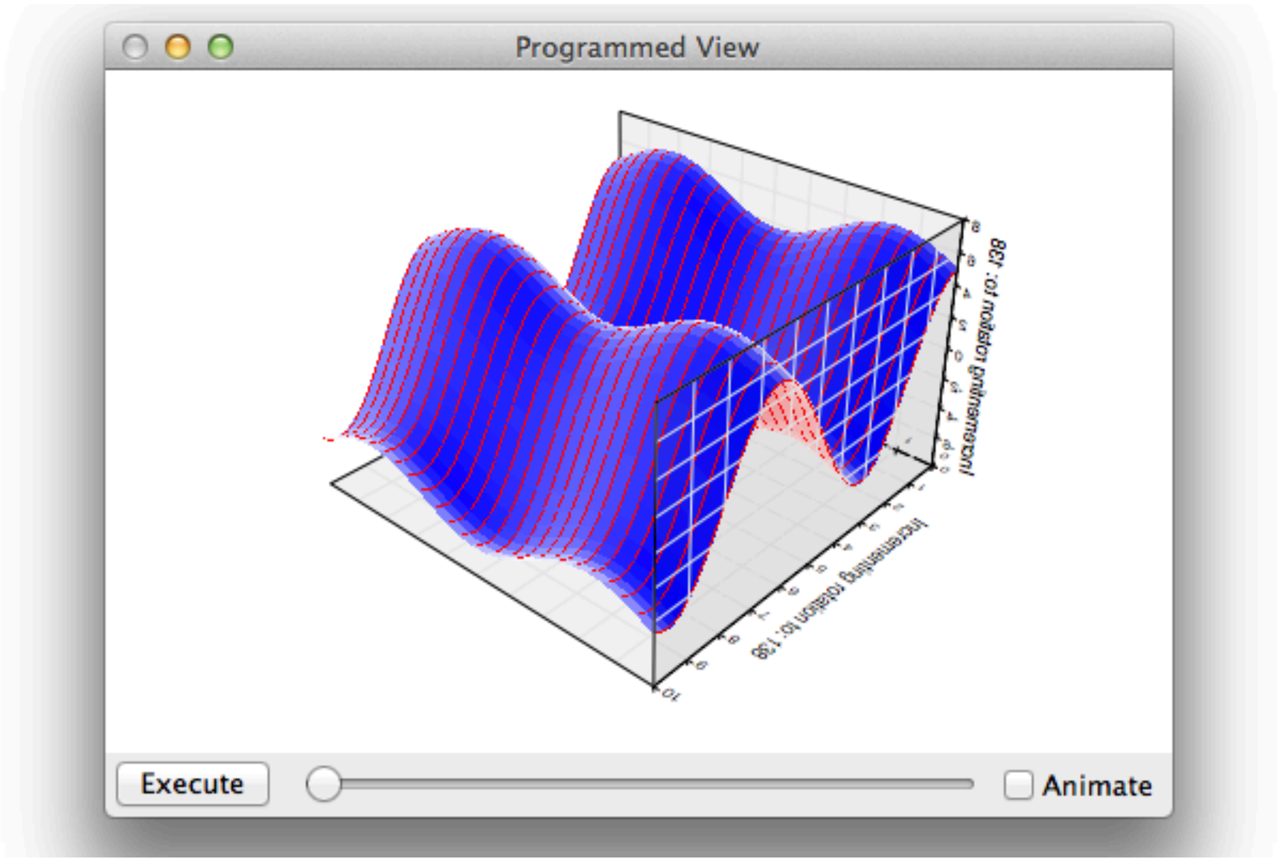
- Drag and drop a table and then hook it up to a data source.
- Use an adapter, in this case a vertical slider, to move the numerical textual values in the table.
- Use a controller to register all the states and objects in order to bind the system together.
- Animate a sine wave and synchronize that sine wave graph with the table.
- Do all of that with a minimal amount of code and load it into the custom application using a Container View, e.g.: with no programming.

Do not underestimate this SineTable project. The methodology and implementation is what is used to program the Graph IDE user interface. To see other examples, open up the Graph IDE application bundle (folder) and navigate to `Contents/Frameworks/Vvidget_GGE.framework/Versions/A/Resources/Base.lproj` where each inspector editor is programmed using techniques identified in the SineTable example project.

Graph IDE ► Custom Application ► Surface

The Surface custom application project produces output like that shown below. Its main features are:

- Makes a 3D surface graph with x-oriented colored grid lines. The graph can be rotated and animated.
- Programmed for the Mac, can be converted to iOS with minimal code changes.
- Shows how to load a Document using a ContainerView without any programming.
- Shows how to register a graphic (its PerspectiveGraph state) so that the program can directly work with it, in this case permit a slider to rotate the graph. This feature is applicable to any graphic state on the document.



The fastest way to write a custom application is to download the Surface example project, compile it and modify it for your needs. Surface resources are available from these links:

Link	Description Of Resource
Surface.zip	The compressed Surface project located on your disk within this manual.
Surface.zip	The compressed Surface project located at the vvidget.org web site.
Xcode	The Xcode application on the Mac App Store.

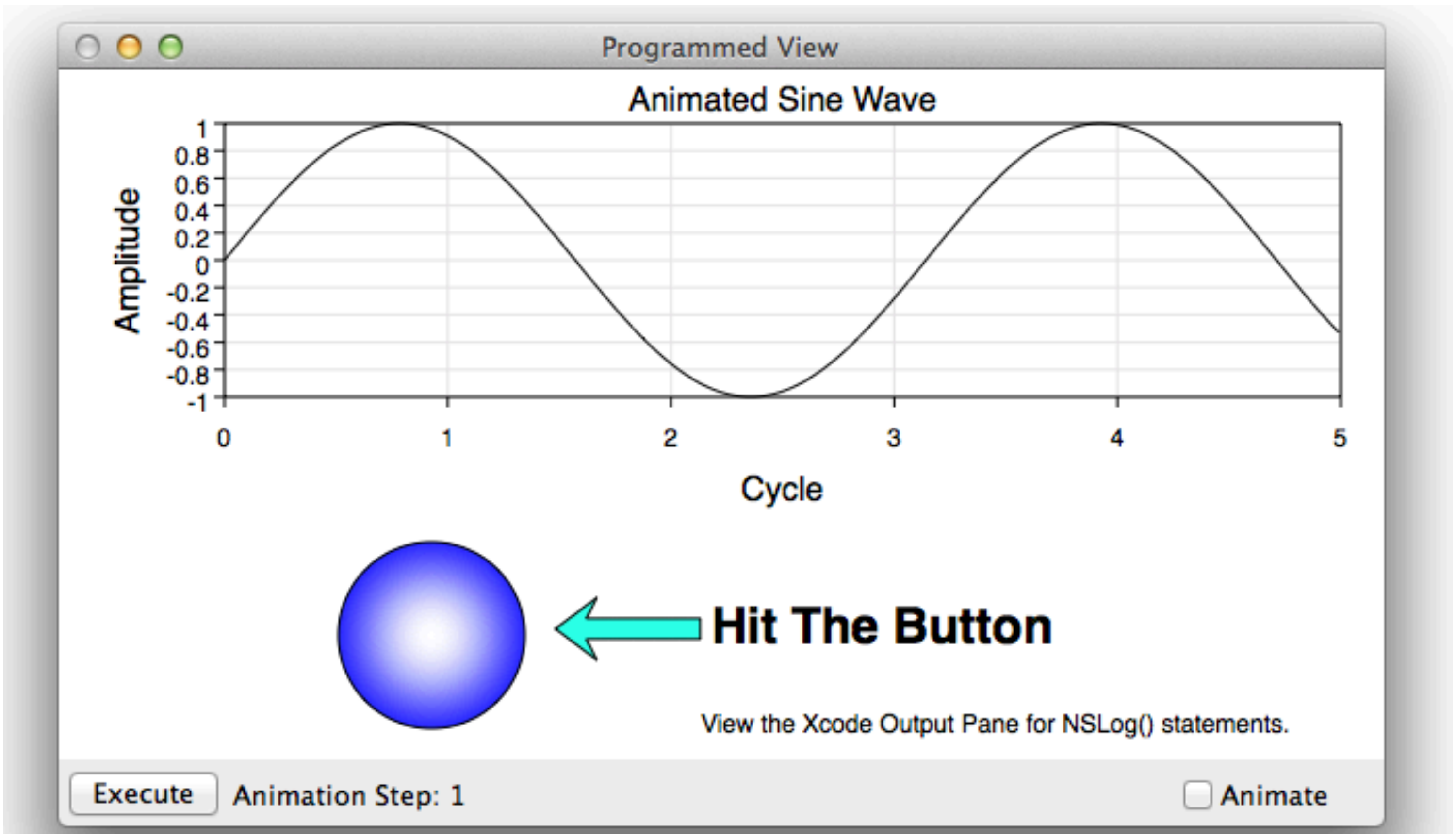
When making the Xcode project follow the steps in the Plugin section, but with these important distinctions:

- When making a new project choose Cocoa Application.
- Make a new Add Copy Files Build Phase with Destination Frameworks and drag the Vvidget framework references from the project navigator to the Copy Files build phase list.

The Surface.zip project is already setup with the proper references and build phases so that might be a good starting point.

Graph IDE ► Custom Application ► HitButton

This section describes how to write a custom application called HitButton. The user interface is shown below.



Placing the cursor over the cosine curve shows a popover. Placing the cursor over the circle send a log line to the Xcode output pane. Clicking the circle turns its color red.

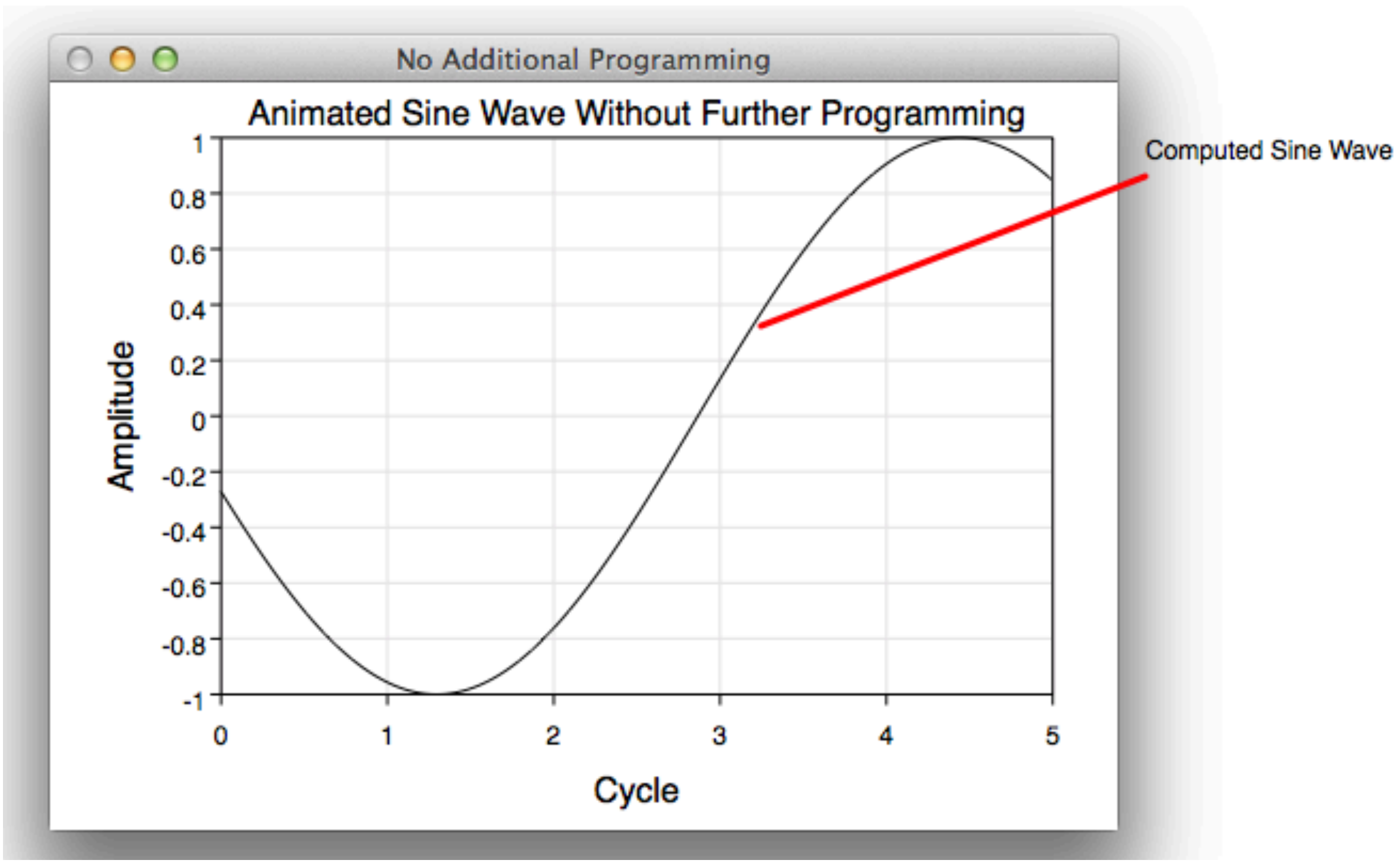
The fastest way to write a custom application is to download the HitButton example project, compile it and modify it for your needs. HitButton resources are available from these links:

Link	Description Of Resource
HitButton.zip	The compressed HitButton project located on your disk within this manual.
HitButton.zip	The compressed HitButton project located at the vvidget.org web site.
Xcode	The Xcode application on the Mac App Store.

The graph and white portion in the figure above is a Graphic View that was pre-programmed using the Programming facilities of Graph IDE. The Execute button and Animate switch are programmed in the custom application.

Notes are contained in the HitButton Xcode project. Instead of adding a lot of verbiage in this section which is pretty dry and hard to follow, notes have been incorporated into the HitButton Xcode project. See that project (see the references above) for further explanation.

This section describes how to write a custom application called HandsOff. The user interface is diagrammed below.


No Additional Programming
Animated Sine Wave Without Further Programming
Computed Sine Wave
Amplitude
Cycle
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5

The fastest way to write a custom application is to download the HandsOff example project, compile it and modify it for your needs. HandsOff resources are available from these links:

Link	Description Of Resource
HandsOff.zip	The compressed HandsOff project located on your disk within this manual.
HandsOff.zip	The compressed HandsOff project located at the vvidget.org web site.
Xcode	The Xcode application on the Mac App Store.

The following explains the essential steps of the HandsOff project.

The graph and white portion in the figure above is a Graphic View that is loaded from a Graph IDE Document. That document is set to animate and compute a sine curve on the graph. There is no other programming required. To get a document to load into an application follow these steps:

- Add a Custom View to the application window.
- In the Identity inspector of that custom view assign the Custom View to the VVPUBLIC_GraphicContainerView class. Then in the User Defined Runtime Attributes add a new row and assign:

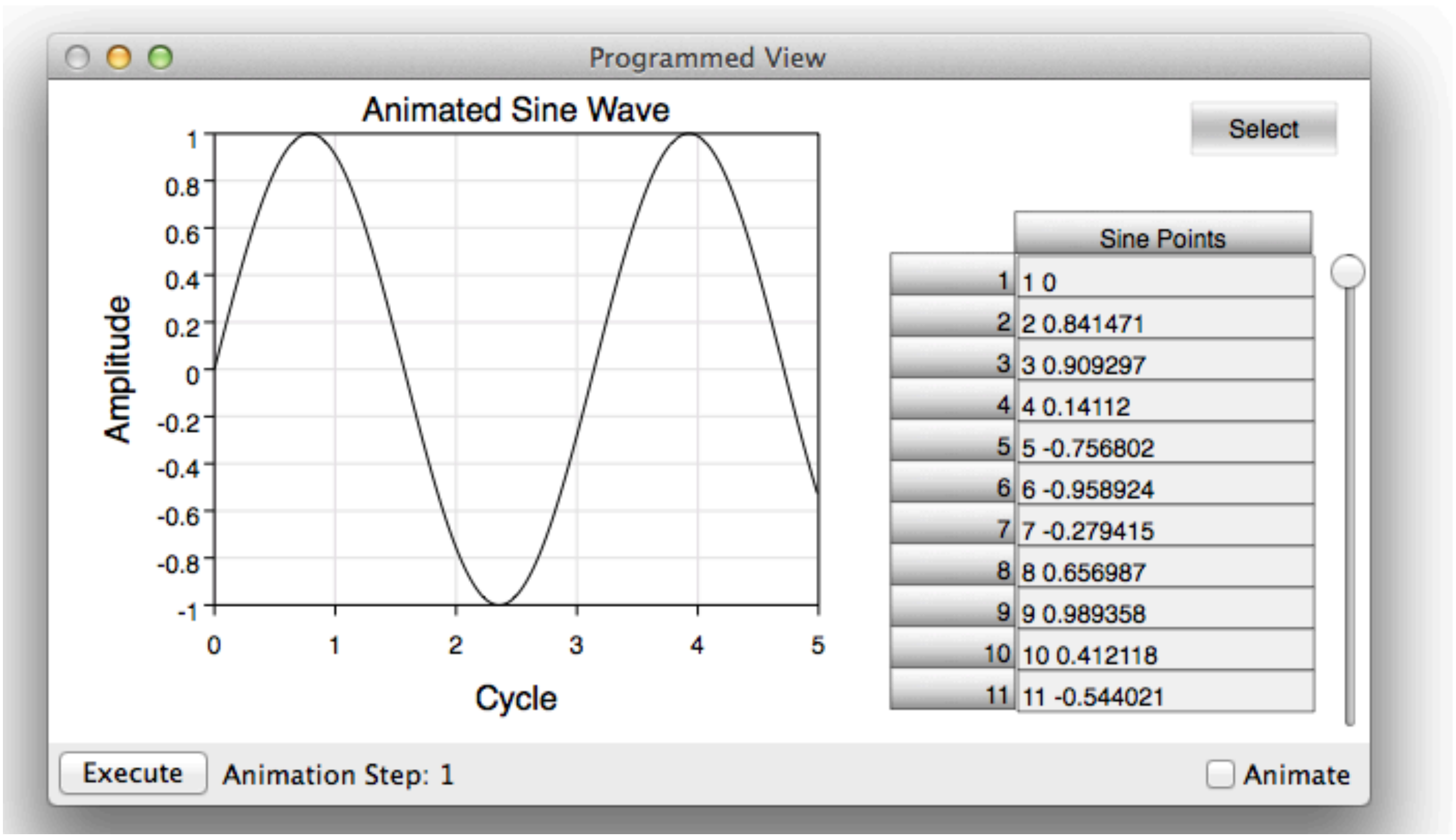
Key Path	:	documentName
Type	:	String
Value	:	HandsOff

- Link in the necessary frameworks.

The HandsOff.zip project is already setup with the proper references and build phases so that might be a good starting point.

Graph IDE ► Custom Application ► GraphStrings

This section describes how to write a custom application called GraphStrings. The user interface is diagrammed below.



	Sine Points
1	1 0
2	2 0.841471
3	3 0.909297
4	4 0.14112
5	5 -0.756802
6	6 -0.958924
7	7 -0.279415
8	8 0.656987
9	9 0.989358
10	10 0.412118
11	11 -0.544021

The fastest way to write a custom application is to download the GraphStrings example project, compile it and modify it for your needs. GraphStrings resources are available from these links:

Link	Description Of Resource
GraphStrings.zip	The compressed GraphStrings project located on your disk within this manual.
GraphStrings.zip	The compressed GraphStrings project located at the vvidget.org web site.
Xcode	The Xcode application on the Mac App Store.

GraphStrings shows how to:

- Implement callback methods in a state object in order to provide custom labels for a graph.

Graph IDE ► Custom Application ► Event Qualifier

The following shows how to instantiate an event qualifier for a Circle graphic. An event qualifier is enabled by default so all you need do is instantiate the programmatic state of the circle (or corresponding graphic's state).

```
@@class() Circle:Object

@@method(public, class) (id)stored;

@@end

{
id myCircle;

myCircle = [Circle stored];
}
```

The following API processes the events. If you wish to alter that processing then you must implement your own graphic state subclass and subclass these methods.

`@@method(public, instance) (void)disableEventQualifier;`

Call like this:

`[myLayer disableEventQualifier];`

This method disables event qualifier processing for the Layer and Single Coordinate Graph states. Most likely, you will not need to call upon this method.

`@@method(public, instance) (void)enableEventQualifier;`

Call like this:

`[myLayer enableEventQualifier];`

This method enables event qualifier processing for the Layer and Single Coordinate Graph states. Call upon this method in a program script. Calling upon this method for a graph enables that graph's processing for the data foreground and background layers so that graphics on the graph are detected and not the graph itself. If this method is not called upon for a graph then the graph itself is detected instead.

`@@method(public, instance) (void)retrieveHitParameters;`

Call like this:

`[self retrieveHitParameters];` That method is called in the `performHit` and `performHover` method to retrieve hit parameters. If you set the iVar `popoverType` to `NO_EVENT_QUALIFIER_POPOVER_TYPE` then the popover is not presented an only the hit parameters are retrieved in a call to `[super performHit]` or `[super performHover]`. Alternatively you can simply call `[self retrieveHitParameters];` without calling super in order to retrieve the hit parameters.

Appends the x and y values to the list of data points for the graphic. The x and y values forms a 2D point. Each value must be of type double.

`@@method(public, instance) (void)encodeHitMessage;`

Call like this:

`[self encodeHitMessage];`

This encodes the hit parameters into the strings hitTitle and hitMessage. This method can be overridden to present a custom message in the popover.

`@@method(public, instance) (void)performHit;`

Call like this:

`[super performHit];`

This method is called when the receiver is hit (clicked upon) by the mouse button. The default implementation calls `[self retrieveHitParameters];` and if the popover is on calls `[self encodeHitMessage];` and then presents the popover, if on. You can subclass this method for your own distinct processing.

`@@method(public, instance) (void)performHover;`

Call like this:

`[super performHover];`

This method is called when the receiver is hovered upon (when the mouse cursor is over the receiver). The default implementation calls `[self retrieveHitParameters];` and if the popover is on calls `[self encodeHitMessage];` and then updates the popover, if on. You can subclass this method for your own distinct processing.

`@@method(public, instance) (void)resetHitParameters;`

Call like this:

```
[self resetHitParameters];
```

This method resets the hit instance variables according to the following.

- Sets `hitIndex` to `0x7fffffff`, `hitXValue` to `0.0` and `hitYValue` to `0.0` for all Graphic objects.
- Sets `hitSegmentIndex` and `hitVertexIndex` to `0x7fffffff` in graphic objects that are point and segment related, such as Cubic Bezier, Function, Polygon, Scatter, Trajectory and Perspective Scatter.
- Sets `hitDataXValue` and `hitDataYValue` to `0.0` for the Cubic Bezier, Function, Polygon, Scatter, Trajectory objects.
- Sets `hit3DXValue`, `hit3DYValue` and `hit3DZValue` to `0.0` for the Perspective Scatter and Perspective Surface graphics.
- Sets `hitCellIndex` to `0x7fffffff` for the Perspective Surface graphic.

`@@method(public, instance) (void)retrieveHitParameters;`

Call like this:

```
[self retrieveHitParameters];
```

This method retrieves the hit instance variables according to the following.

- Sets `hitIndex` to the index of the receiver in its layer and `hitXValue` and `hitYValue` to the hit coordinates for all Graphic objects. Incase the graphic is on a graph, the hit coordinate values are in the units of the graph.
- Sets `hitSegmentIndex` and `hitVertexIndex` to the respective values in graphic objects that are point and segment related, such as Cubic Bezier, Function, Polygon, Scatter, Trajectory and Perspective Scatter. The index starts at zero.
- Sets `hitDataXValue` and `hitDataYValue` to the data value hit for the Cubic Bezier, Function, Polygon, Scatter and Trajectory objects. The data values can differ from the hit values in that hit coordinates are resolved to the hit aperture and event processing resolution while the data values are retrieved directly from the data of the object.
- Sets `hit3DXValue`, `hit3DYValue` and `hit3DZValue` to their respective values for the Perspective Scatter and Perspective Surface graphics. Those values are defined in terms of the 3D graph units.
- Sets `hitCellIndex` to the cell index hit for the Perspective Surface graphic. The index starts at zero and increments contiguously along the x-direction one cellular triangle at a time. To get the grid rectangle index hit divide by two.

`@@method(public, instance) (void)performUnHit;`

Call like this:

```
[super performUnHit];
```

This method is called when the receiver is unhit (the mouse button is released upon). The default implementation calls `[self resetHitParameters];`. You can subclass this method for your own distinct processing.

`@@method(public, instance) (void)performUnHover;`

Call like this:

```
[super performUnHover];
```

This method is called when the receiver is unhovered (the mouse cursor leaves the graphic). The default implementation calls `[self resetHitParameters];`. You can subclass this method for your own distinct processing.

A premade Xcode project demonstrating the use of the methods above is described in the HitButton section.

The following is a code splice for loading a Graphic View into a custom application's window. Note that this should only be used within Xcode (not in a Graph IDE script) and that the prefix `VVPUBLIC_` needs to be prepended to the class name.

```
NSString *aPath = [[NSBundle mainBundle] pathForResource:@"MyDocument" ofType:@"vvibook"];

theDocument = [[VVPUBLIC_GraphicDocument alloc] init];
programmedView = [theDocument viewForDocumentPath:aPath];

[programmedView setFrame:[_switchView frame]];
[programmedView setAutoresizingMask:[_switchView autoresizingMask]];

[[_switchView superview] replaceSubview:_switchView with:programmedView];
```

If you intend to register states then you need to assign a controller such as in this code splice:

```
NSString *aPath = [[NSBundle mainBundle] pathForResource:@"MyDocument" ofType:@"vvibook"];

theDocument = [[VVPUBLIC_GraphicDocument alloc] init];
[theDocument setController:self];
programmedView = [theDocument viewForDocumentPath:aPath];

[programmedView setFrame:[_switchView frame]];
[programmedView setAutoresizingMask:[_switchView autoresizingMask]];

[[_switchView superview] replaceSubview:_switchView with:programmedView];
```

The following is API description specific to the Document class.

```
@@method(public, class) (id)alloc;
```

Call like this:

```
[VVPUBLIC_GraphicDocument alloc]
```

That allocates a VVPUBLIC_GraphicDocument object.

```
@@method(public, instance) (void)enableEventQualifierProcessing;
```

Call like this:

```
[myDocument enableEventQualifierProcessing];
```

Enables event qualifier processing. It does that by setting some internal states and then executing all programs in the document. Note that, in addition to calling this method, you must enable the event qualifier for each Layer, otherwise normal Graph IDE event processing with occur.

```
@@method(public, instance) (id)init;
```

Call like this:

```
myDocument = [[VVPUBLIC_GraphicDocument alloc] init];
```

That allocates, initializes and assigns a document object to the myDocument variable. The document must not be released until its graphic view is no longer needed.

```
@@method(public, instance) (void)performAnimationStep;
```

Call like this:

```
[myDocument performAnimationStep];
```

Perform one step of the animation.

```
@@method(public, instance) (void)release;
```

Call like this:

```
[myDocument release];
```

Releases the document. Only release the document when you are completely done using the view returned by the method viewForDocumentPath:.

```
@@method(public, instance) (void)setController:(id)aController;
```

Call like this:

```
[myDocument setController:self];
```

The controller is accessible to all programming states for the purpose of registering those states, and subsequently synchronizing states, in a larger system.

```
@@method(public, instance) (void)startAnimation;
```

Call like this:

```
[myDocument startAnimation];
```

Starts the animation of the graphic view maintained by the document.

```
@@method(public, instance) (void)stopAnimation;
```

Call like this:

```
[myDocument stopAnimation];
```

Stops the animation of the graphic view maintained by the document.

```
@@method(public, instance) (id)viewForPath:(NSString *)aPath;
```

Call like this:

```
aView = [theDocument viewForPath:aPath];
```

Returns the Graphic View of a document. That view is a subclass of NSView and can be inserted into a view hierarchy as desired.

Graph IDE ► Custom Application ► Container View

The container view loads a Graphic View from a Graph IDE Document without programming. All that is required is to make a few settings in a Xcode interface file per the following instructions.

- Make a Graph IDE document with the requisite configuration, save it (for example sake say the name is *MyDocument*) and then add it as a resource to a Xcode project.
- Add a Custom View to the application window.
- In the Identity inspector of that custom view set its class to VVPUBLIC_GraphicContainerView. Then in the User Defined Runtime Attributes add a new row and assign:

Key Path	:	documentName
Type	:	String
Value	:	*MyDocument*

- If needed connect the controller outlet to the nib file owner or other controller object.

There is no programming API for the container view. The container view class (VVPUBLIC_GraphicContainerView) is subclassed from public classes so there is no further API that needs resolved. It does, however, require that the appropriate frameworks be linked into the project. For that information consult the HandsOff section.

The container view takes care of loading the graphic view into an interface at runtime. The burden of programming falls upon the Programming of that graphic view. That programming is done well in advance of using a container view to load the graphic view into a separate application.

Graph IDE ► Custom Application ► Controller

A controller is assigned while loading a view using the Container View or Document API. It is an instance that permits any Programming state to register with a custom application.

The controller class needs to implement the following method:

```
-(void)register:(id)aState withName:(const char *)aName recursionIndex:(unsigned)recursionIndex
```

Where the arguments are defined as follows:

- `aState`: The state that is being registered.
- `aName`: The name of the state.
- `recursionIndex`: The recursion index of the state. The recursion index is usually zero because most graphics will not be recursed (using the recur call).

The controller is a "glue" object that binds Graph IDE's object nodes to the objects in a custom application. See the SineTable project for its use.

Graph IDE ► Custom Application ► Distribution

Graph IDE makes writing a custom data visualization application comparatively easy (compared to other programming challenges). Writing and using such an application has merits of its own. Once you have a useful application then you may wish to deploy (distribute) it to other people for their own use. The following is a check list of things that need to be done before you can distribute such an application.

- You must accept the Redistribution Agreement and purchase a redistribution license.
- For iOS (iPhone, iPad and iPod touch) the libVvidget.a ARM archive library (device, not simulator) must be statically linked into your application.
- For OSX (Mac) the framework binaries must be copied into your application bundle in its Framework directory. Those binaries are in the folder "`/Applications/Graph Builder.app/Contents/Frameworks`" and are the Frameworks `Vvidget_GG.framework`, `Vvidget_GS.framework`, `Vvidget_SAF.framework`, `Vvidget_SAG.framework`, `Vvidget_SAM.framework`, `Vvidget_SAT.framework`, `Vvidget_SBM.framework`, `VvidgetCode.framework`. The Resources and Header folders must be removed before distribution. Redistributables does not include code signing assets. Do not copy or link against the Vvidget_GGE.framework, Vvidget_GSE.framework, Vvidget_PVS.framework, Vvidget_SCS.framework frameworks as they are not needed for a custom application.
- This is a non-comprehensive check list. Making a distributable application can be very easy or a major undertaking. There is no way of knowing in advance what your requirements are and hence how to go about solving them so such details are left to the reader.
- The example projects and applications in this section should be configured for distribution out of the box and provide a good starting point for configuring Xcode and related facilities.
- For additional help with deployment please email support@vvi.com.

Graph IDE ► Network Clients

Graph IDE includes a powerful Server that can vend results to clients. This section describes two such clients as referenced below.

Section	Description
Command Line Tool	Describes a command line tool that can be used to make graphs and script results from a unix shell, command line prompt and other facilities that use stdio.
Web Adapter	Describes a CGI adapter that permits Graph IDE results to be vended to the Internet dynamically and in real time.

It should be noted that the client server protocol is based upon industry standard TCP/IP communication and thus the server can vend to a variety of clients.

Graph IDE ▶ Network Clients ▶ Command Line Tool†

The following lists some resources available for the command line tool. You may wish to skip the following table on first pass.

Link	Description Of Resource
vvizard.zip	The compressed vvizard unix tool located on your disk within this manual.†
vvizard.zip	The compressed vvizard unix tool located at the vvidget.com web site.†
Vvidget™ Code Reference Manual	The Vvidget™ Code Reference Manual describes the state information used as input to the server. It also describes the vvizard tool.†

The following are brief examples of the use of the server built into Graph IDE. Download vvizard.zip (using the links shown above), uncompress the download by double clicking it and then drag the result to a permanent location to install it. For example sake the following assumes you installed vvizard to the ~/Downloads folder. Launch the Terminal application and type the following input.

Prints a menu of commands to the terminal window (stdout)

```
cd ~/Downloads
./vvizard
help
quit
```

Makes a line graph with two curves, displays it and then exports it to Graph IDE†

```
cd ~/Downloads
./vvizard
verbose on
server localhost
reset
add chart_type line
add chart_subtype linear
add chart_format_type default
add title Vvizard Help Example Graph
add x_title My X Title
add y_title My Y Title
add data_1 1 2 2 4 3 6 4 2
add data_2 1 3 2 1.4 3 8.7 4 1.1
display
export
```

Because vvizard accepts input from stdin it can be used in shell scripts and other scripting programming environments to make graphs. vvizard is a single unix binary that does not link against non-system frameworks which means you don't need any additional software (not even Graph IDE) to use it, as long as you have access to a graphing server.

Advisory: vvizard is a unix process which acts as a thin client to the server. It is not sandboxed. It injects byte streams into Graph IDE for processing. That means that the Mac App Store edition of Graph IDE can aide in the display of graphs but can not save or open the byte stream translations and for that functionality you need the non-sandboxed Manufacturer edition of Graph IDE.

† Sold and licensed separately. This section describes the use of the Server built into Graph IDE. The Command Line Tool is not part of Graph IDE.

Graph IDE ► Network Clients ► Web Adapter†

This section describes how Graph IDE results can be vended to the Internet. It might not dawn on you at first but because Graph IDE documents are Programmable that means that when they are opened a program is executed. The Web Adapter works with the built-in Server to respond to a web browser by opening up a document, executing it and then translating that result to an image that the web browser can use. Thus without any additional effort, beyond correctly setting up a web server, Graph IDE can be used to vend dynamic, programmable and real-time results to the Internet.

The following lists some resources available for the web adapter. You may wish to skip the following table on first pass.

Link	Description Of Resource
mod_pvs.so.zip	The compressed mod_pvs.so shared object library located on your disk within this manual.
mod_pvs.so.zip	The compressed mod_pvs.so shared object library located at the vvidget.com web site.
Vvidget™ Server Reference Manual	The Vvidget™ Server Reference Manual describes how to format query strings for input into the server.†
vvidget.com	The vvidget.com internet server is the same one incorporated into Graph IDE and you can use that site to vend results as well.†

The following assumes that you have the Apache web server running on the computer that is licensed for Graph IDE, the Graph IDE Server is set to on, Graph IDE is auto launched at login and your computer is set to login the user when it boots.

Download mod_pvs.so.zip (using the links shown above), uncompress the download by double clicking it and then drag the result to the module directory of your web server.

If you configured the web server correctly and are viewing this manual on the computer running that web server then the following image should show a line graph.

Which is the imaged results of the following URL:

```
<img src="http://localhost/graph.pvs?
1&EMAIL&chart&1&400,300&chart_type=1&chart_subtype=0&chart_format_type=1
&title=My%20Title&x_title=My%20X%20Title&y_title=My%20Y%20Title
&data_1=1%2020.0%202%2040.0%203%2035.4%204%2066.2%205%2077.3
&data_2=1%2030.0%202%2099.0%203%2010.0%204%2060.0
&data_3=1%20123.0%202%2034.0%203%2099.0%204%2077.3
&line_color=000000" width=400 height=300 border=0>
```

The above image was generated on the fly without any additional Graph IDE document. If you make your own Graph IDE document and save it in your web server's document path (for example: `/Library/WebServer/Documents`) then it too can be imaged. For example, if you make a Graph IDE document named GnomeStudy with Graphic View width 400 and height 300 and save that document in the web server's document path then its display is accessible to anyone on the Internet using a URL like this:

```
<img src="http://www.mydomain.com/graph.pvs?1&direct&document&2&400,300&GnomeStudy">
```

where "mydomain.com" is the domain of your own server.

Notice that the Mac App Store Edition of Graph IDE is sandboxed so can only open documents for which it has permission. The Manufacturer Edition is not sandbox and can open documents in the web server documents directory. Because of the very large matrix of deployment conditions it is impossible to itemize all of the deployment issues within the limited scope of this manual. If you are using a sandboxed version then it might be possible to create a link from the web server document folder and the Server Document's access point (see Server) and store documents at that access folder so that the web adapter can image them. On the other hand, depending on the version of OS X you are using and the security settings that might not be possible. As usual, there are many site-specific caveats and security issues.

The server (also part of Peer Visual Server† and Vvidget Server†) is based on an enterprise class SOA server which can be configured and scaled to server farm multi-homed, multi-process, distributed, multi-threaded, multi-client asynchronous, state and stateless use, fallback and autorestart configurations, i.e.: the full works. The configuration built into Graph IDE can power a substantial web site service as well as be used for scripting purposes. The extensive options, configurations, use and reliability are beyond the scope of this manual. This manual is simply concerned with turning the server on and demonstrating its use. Other referenced manuals are more comprehensive.

† Sold and licensed separately.

Graph IDE ► Export And Import

There are may ways to export and import data and representations. The Tables section shows how to do it for numeric and textual data. The Programming section shows how to generate data from an algorithm. Programming combined with a Plugin or directly linking with your code is the most efficient way to export and import data and representations.

This section itemizes some miscellaneous ways to export and import.

Section	Description
Dictionary	Describes the lossless encoding mechanism used by Vvidget.
Raw Data Points	Describes how to get raw data (x and y values) in and out of Graph IDE.
Standard	Standard methods for exporting and importing.

Graph IDE ► Export And Import ► Dictionary

The dictionary is the main way Graph IDE stores information. It is a lossless recursive key value coding of things that can be represented by Graph IDE (those things are called Vvidgets). It is useful to understand this encoding for programming considerations, but it probably has little other utility.

A user of Graph IDE would probably be most productive modifying the attributes of Vvidgets (such as the graphical attributes) via tools like Graph IDE and also using a data parser (see Raw Data Points) to enter bulk data points.

Example Of A Text Encoded Dictionary

The dictionary below describes a circle. As you can see, you will probably be more comfortable working with a graphical representation of the circle rather than the dictionary example.

```
{
VVKB = VVC48;
VVK9 =
      {
      VVKB = VVC1L;
      VVKI = 1;
      VVKH = 3/15/2004 6:20:22.903894000;
      };
VVKr =
      {
      VVKB = VVC3L;
      VVKp = YES;
      VVKy = YES;
      VVKz = NO;
      };
VVKs =
      {
      VVKB = VVC3M;
      VVKp = YES;
      };
VVKt =
      {
      VVKB = VVC46;
      VVKp = YES;
      VVK10 = YES;
      VVKz = NO;
      VVK34 = "1.35 0 0 1.28 112 337";
      VVK35 = 4;
      VVK36 = "0 0 100 100";
      VVK3A = 0;
      VVK3B = 0;
      VVK3C = 6.283185307179586232;
      };
VVKu =
      {
      VVKB = VVC47;
      VVKp = YES;
      VVK11 = YES;
      VVK12 = YES;
      VVK2t =
            {
            VVKB = VVC3T;
            VVK2B = 1;
            VVK2C = 0;
            VVK2J =
                  {
                  VVKB = VVC1O;
                  VVKN = 0;
                  VVKO = 0.78431373834609985352;
                  VVKP = 1;
                  VVKQ = 1;
                  VVKR = 0.8;
                  VVKS = 0;
                  VVKT = 4294967295;
                  };
            VVK2D = 2;
            VVK2E = 0;
            VVK2F = 0;
            VVK2G = 1;
            VVK2I =
                  {
                  VVKB = VVC1O;
                  VVKN = 0;
                  VVKO = 0;
                  VVKP = 0;
                  VVKQ = 0;
                  VVKR = 1;
                  VVKS = 0;
                  VVKT = 4294967295;
                  };
            VVK2K =
                  {
                  VVKB = VVCM;
                  VVKA = {};
                  };
            VVK2L = 0;
            VVK2M = YES;
            VVK2N = NO;
            VVK2O = NO;
            VVK2P = 0;
            VVK2Q = 0;
            VVK2R = 1;
            VVK2T = NO;
            VVK2U =
                  {
                  VVKB = VVC1O;
                  VVKN = 0;
                  VVKO = 0;
                  VVKP = 0;
                  VVKQ = 0;
                  VVKR = 1;
                  VVKS = 0;
                  VVKT = 4294967295;
                  };
            VVK2V = NO;
            VVK2W = NO;
            VVK2X = NO;
            VVK2Y = NO;
            VVK2Z = NO;
            VVK2a = YES;
            VVK2b = 0.52359877559829892668;
            VVK2c = 10;
            VVK2d = 0.62;
            VVK2e =
                  {
                  VVKB = VVC1O;
                  VVKN = 0;
                  VVKO = 0;
                  VVKP = 0;
                  VVKQ = 0;
                  VVKR = 1;
                  VVKS = 0;
                  VVKT = 4294967295;
                  };
            VVK2f =
                  {
                  VVKB = VVC1O;
                  VVKN = 0;
                  VVKO = 0;
                  VVKP = 0;
                  VVKQ = 0;
                  VVKR = 1;
                  VVKS = 0;
                  VVKT = 4294967295;
                  };
            VVK2g = 0;
            VVK2h = YES;
            VVK2i = NO;
            VVK2j = 0;
            VVK2k = 0.52359877559829892668;
            VVK2l = 10;
            VVK2m = 5;
            VVK2n =
                  {
                  VVKB = VVC1O;
                  VVKN = 0;
                  VVKO = 0;
                  VVKP = 0;
                  VVKQ = 0;
                  VVKR = 1;
                  VVKS = 0;
                  VVKT = 4294967295;
                  };
            VVK2o = YES;
            VVK2p = YES;
            VVK2q =
                  {
                  VVKB = VVC1O;
                  VVKN = 0;
                  VVKO = 0;
```

```
                  VVKP = 0;
                  VVKQ = 0;
                  VVKR = 1;
                  VVKS = 0;
                  VVKT = 4294967295;
                  };
            VVK2r =
                  {
                  VVKB = VVC1O;
                  VVKN = 0;
                  VVKO = 0;
                  VVKP = 0;
                  VVKQ = 0;
                  VVKR = 1;
                  VVKS = 0;
                  VVKT = 4294967295;
                  };
            };
      VVK2u = 0;
      };
VVKv =
      {
      VVKB = VVC3P;
      VVKp = YES;
      VVK13 = NO;
      VVK14 = NO;
      VVK15 = NO;
      VVK16 = NO;
      VVK17 = NO;
      VVK18 = NO;
      VVK19 = "0 0 0 0";
      VVK1A = 0;
      };
VVKx =
      {
      VVKB = VVC49;
      VVKp = YES;
      VVK1D = YES;
      VVK1E = NO;
      VVK1F = 0;
      };
}
```

Graph IDE ► Export And Import ► Raw Data Points

You can enter data points on a graphic-by-graphic (curve-by-curve) basis. To do that first make a prototype Data Graphic or other graphic that has points as a constitutive parameter. Then use that graphic's inspector editor Table interface or the Parser Inspector Editor to enter points.

The following shows how to enter data from a simple text representation using the data parser.

Point Data

Point data-oriented graphics include the Polygon, Function and Scatter graphics. Data points are entered as x and y whitespace separated pairs so, for example:

```
5 6
6 20
7 40
```

Defines a curve with x-values 5, 6, 7 and y-values of 6, 20, 40 respectively.

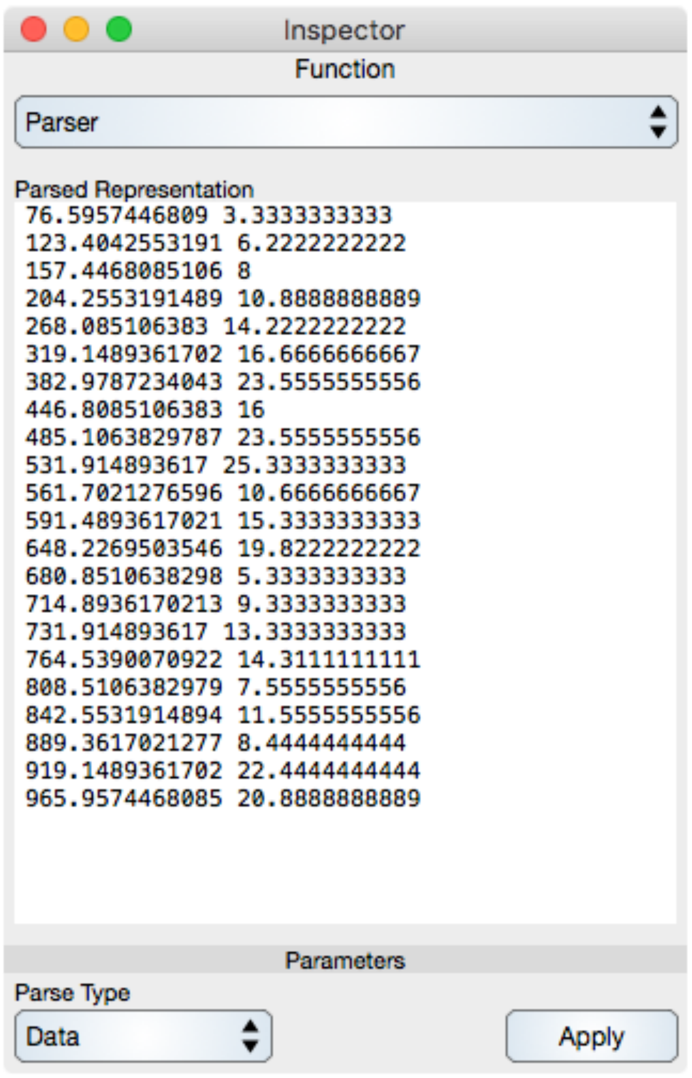
Spline Data

Spline data-oriented graphics include the Cubic Bezier and Trajectory graphics. In the case of spline-oriented graphics you enter the beginning tangent end point, the actual data point and then the ending tangent end point in that order and all white space delimited. So, something like this:

```
5 6 5 6 5 6
6 20 6 20 6 20
7 40 7 40 7 40
```

That case is graphically equivalent to the Point Data example of before because the tangent lines have zero length (coincide with the respective data point).

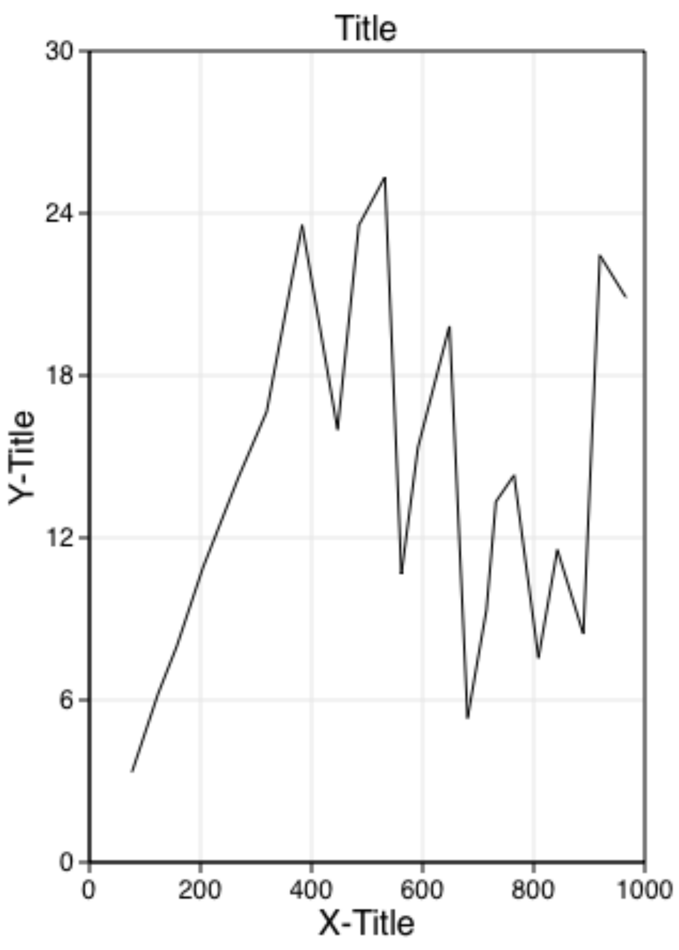
The Parser sub-inspector-editor has a Data representation interface where you can type in values of your data, or paste in all the values. An example is shown in the figure below. After typing or pasting the data click the Apply button to apply it to the respective data graphic, in this case a Function.



Parsed Representation

```
76.5957446809 3.3333333333
123.4042553191 6.2222222222
157.4468085106 8
204.2553191489 10.8888888889
268.085106383 14.2222222222
319.1489361702 16.6666666667
382.9787234043 23.5555555556
446.8085106383 16
485.1063829787 23.5555555556
531.914893617 25.3333333333
561.7021276596 10.6666666667
591.4893617021 15.3333333333
648.2269503546 19.8222222222
680.8510638298 5.3333333333
714.8936170213 9.3333333333
731.914893617 13.3333333333
764.5390070922 14.3111111111
808.5106382979 7.5555555556
842.5531914894 11.5555555556
889.3617021277 8.4444444444
919.1489361702 22.4444444444
965.9574468085 20.8888888889
```

Parse Type: Data



Graph IDE ► Export And Import ► Standard

There are many ways to export and import data and representations into and from Graph IDE. The Introduction to this section lists the many ways and what follows are some standard mechanisms.

Copy and Paste

All graphics respond to copy and paste. First select the graphics you wish and then type command-c and paste it (command-v) where you wish. Typically the paste will be a graphical image of the selection, however if the application pasted to only accepts textual entry then the paste will be a textual serialization of the selected graphics which is a Dictionary.

The copy operation defines an implicit frame of the selected graphics. If you so not like that frame then it can be artificially sized larger by first placing a rectangle under the selected graphics, turning that rectangle's drawing off (stroke and fill off) and then drag-selecting the previous selected graphics plus the hidden rectangle and copying the new selection. Once done with the paste you should probably reselect the hidden rectangle, using a drag select, and delete it.

Copying and Pasting in a Graphic View works with the graphical representation while copying and pasting in a Table works with the textual representation, which is usually a decimal representation.

Dragging

The Palettes section describes how graphics can be dragged within Graph IDE. On platforms that accept inter-app dragging graphics can also be dragged to applications running on those platforms.

Print

The Graphic View can be printed in the usual way, command-p. Before you print you may first wish to alter the page layout of the graphic view to correspond to the print pagination.

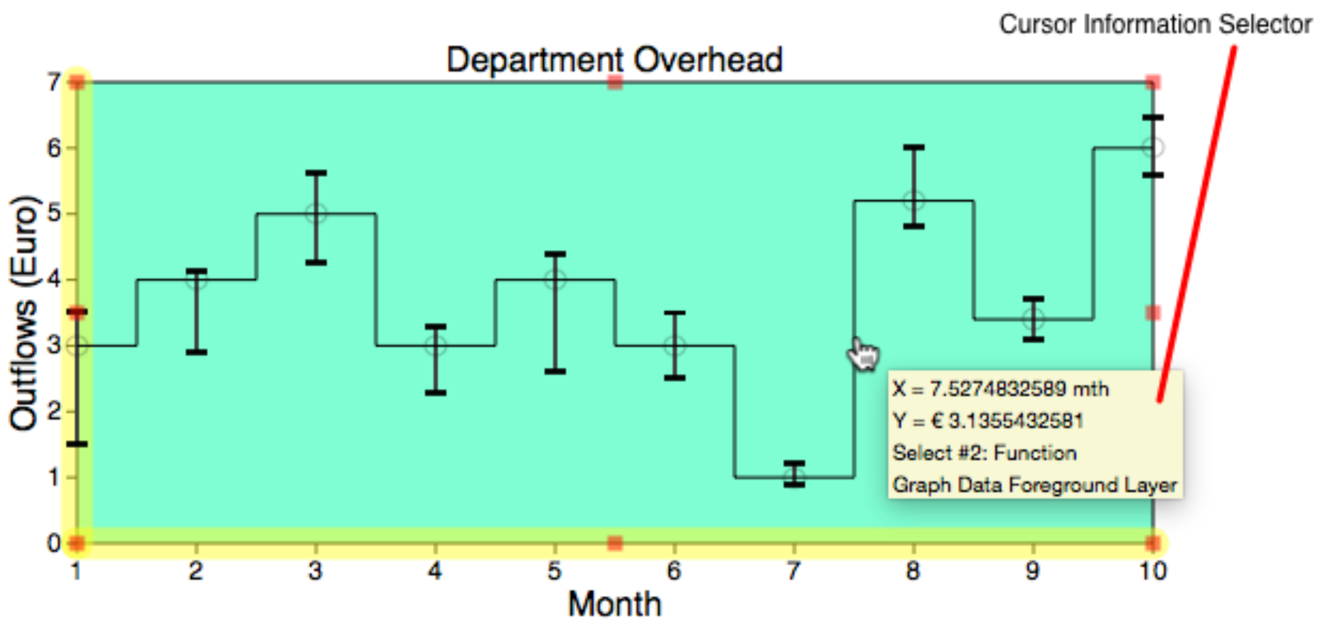
Graph IDE ► Miscellaneous

The following is a brief list of Miscellaneous sections:

Section	Description
Cursor Information	Describes the Cursor Information Panel.

Graph IDE ► Miscellaneous ► Cursor Information

The Cursor Information (short for Cursor Information Selector) is a small area that tracks the cursor as you move it and shows information pertinent to the current or proposed operation. The figure below shows the Cursor Information hovering over a Function graphic (curve) that resides on the graph's coordinate system.



X = 7.5274832589 mth
Y = € 3.1355432581
Select #2: Function
Graph Data Foreground Layer

Note these features of the cursor information shown above:

- The first two lines show the cursor's x and y values in the graph coordinate and units. The first line has units of mth (month) and the second line has units of Euros. Notice that month is an abbreviation suffix while Euro is a symbol prefix, in accordance with convention and standards.
- The third line shows the proposed operation (upon mouse click or touch), the sequence order of the graphic in its Layer and the type of graphic (or its description if one is assigned).
- The fourth line shows the identity of the layer that the graphic is in. The identity has a default or can be assigned using the Layer inspector editor.
- The coordinate variables (X and Y in this case) change according to the coordinate system of the Graph thus adding another element of information.

The Cursor Information changes according to focus and coordinate system. As you move it around you will see its background color change according to fixed criterion. The Cursor Information will also report the point or spline value and number being edited when a graphic with a point edit mode is being edited.

The Cursor Information Selector can be turned off in the Application Preferences inspector editor. The Cursor Information may appear distracting but it also provides valuable information that is gleaned from various layers, coordinates and contextual operations and presented in a immediate way so turning it on or off has different advantages.

The cursor (if available) itself also changes according to context and provide a rudimentary amount of information, but not nearly the amount of information as compared to the Cursor Information Selector.

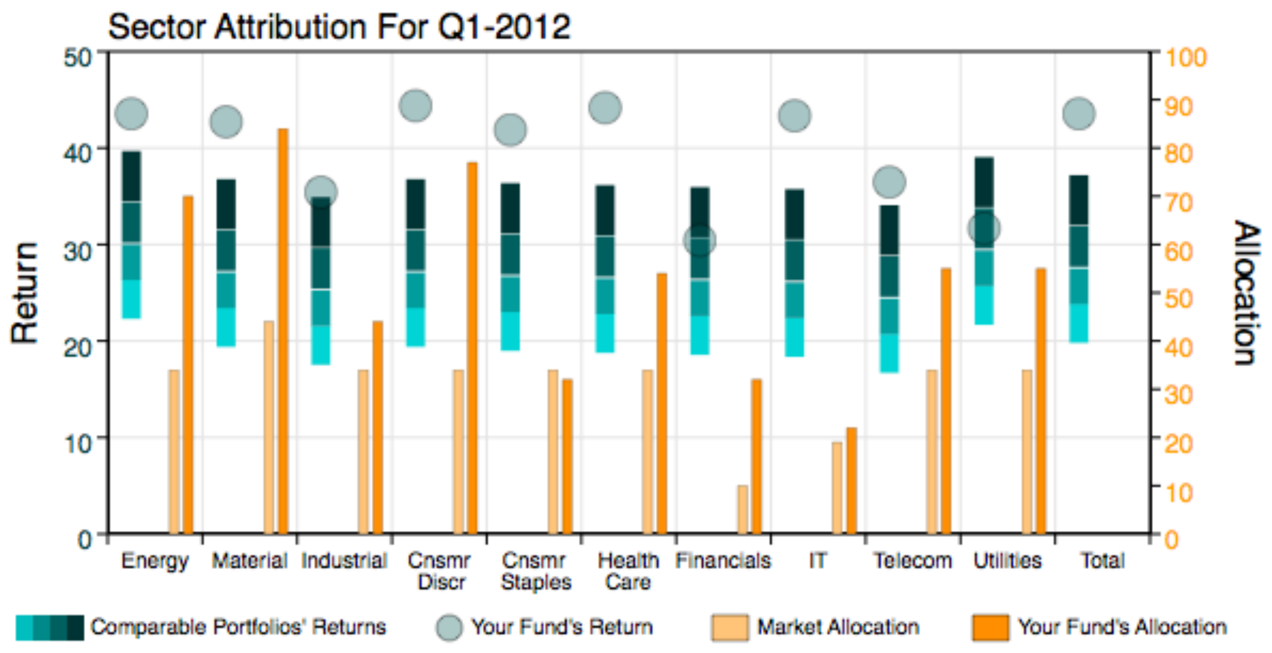
Table Of Contents

Graph IDE ► Examples

The following is a brief list of Examples sections:

Section	Description
Attribution Graph	An interesting graph used in financial management.
Button	Describes some of the things needed to make a button.

With some consideration complex and very exacting graphs can be constructed. The following is an Attribution Graph used in financial systems.



With a modicum of programming, the graph above can be made to show real-time portfolio data. When used with the Server, such a graph can vend real-time results over the Internet.

Graph IDE ▶ Examples ▶ Button

A button is a graphic that, when clicked, changes state and also sends a message to a target object so something can be done. Making a button and using a button is trivial and also an iconic computer science "homework assignment". In the 1980s I suggested using a button on an advanced test bed for a prototype military radar system being built by teams of PhD. researchers at a federal research center. The group leader told me that it was too risky and they preferred hardware toggle switches. That is not the case now and buttons are a good example of Graph IDE's Programming capabilities as well as a good example of how many ideas went from novel and risky to pedestrian.

The following figures shows some buttons.

The buttons shown above are stock buttons from a palette. They are made by making two graphics (in this case group graphics), placing them over each other and then grouping those graphics. The bottom graphic is the off state while the top graphic is the on state. Then the group is programming using the following script:

```
@@class() Button:Object

@@method(public, class) (id)stored;
@@method(public, instance) (id)registerWithName:(const char *)myName;

@@end

/* Execution block */

{
id myButton;

myButton = [Button stored];
[myButton registerWithName:"my name"]; }
```

The Button class is built into Graph IDE, however with a modest amount of use of the Event Qualifier specifications you can make your own button class and way of making buttons respond to mouse clicks and hovers. The `registerWithName:` line sends a message to the controller (see Container View) so that the button state (pointer aka: id) can be registered and the target and action set using the Button methods `setTarget:` and `setAction:`. When the target is called upon it can retrieve the button state by calling the method `state` upon it (which returns a BOOL type value).

The SineTable project shows the use of a button. This section is simply concerned with how to make a graphical element which meets the specifications of a button. With a little ingenuity it is not hard to figure out how to make a wide range of controls. In fact, a graph can be a control where it displays data and event qualifiers are used for feedback in order to control a process whose data is shown on the graph.

Graph IDE ► Legal

This documentation and the software it describes is governed by a End User License Agreement (EULA) between you and VVimaging, Inc. To read that license click License Agreement. The Intellectual Property contained in this product is further protected by trade-secret, trademark and tradedress properties as describe in Trademarks. This documentation and the software that it describes and all other material distributed with it is © Copyright 1991-2022 by VVimaging, Inc. (VVI) with All Rights Reserved.

The following is a brief list of Legal sections:

Section	Description
Cloud Agreement	The VVI Cloud Service Agreement.
License Agreement	The Graph IDE End User License Agreement.
Trademarks	A list of trademarks used in this manual.
Redistribution Agreement	The Vvidget Frameworks Redistribution License Agreement.
Credits	Credits for third-party software.

Derived products as described in the Custom Application section require a separate Redistribution Agreement when distributed to third parties. For that Redistribution Agreement contact VVI at support@vvi.com.

Below is a copy of the End User License Agreement that you received with your copy of Graph IDE and was presented to you before installing Graph IDE. Please note: Graph IDE, including any of its files together or separately, may only be installed on one computer at a time when the license fee has been remitted to VVIMAGING, INC. for that computer and only in accordance with the following license agreement.

Graph IDE™ v12.11.3

END-USER LICENSE AGREEMENT FOR VVI SOFTWARE

IMPORTANT-READ CAREFULLY: This End-User License Agreement ("Agreement") is a legal agreement between you (either an individual or a single entity) and VVimaging Corporation (VVI) for the VVI software product(s) identified above which may include associated software components, media, printed materials, and "online" or electronic documentation ("SOFTWARE PRODUCT"). By installing, copying, or otherwise using the SOFTWARE PRODUCT, you agree to be bound by the terms of this Agreement. If you do not agree to the terms of this Agreement, do not install or use the SOFTWARE PRODUCT. If the SOFTWARE PRODUCT was purchased by you, you may return it unopened to your place of purchase for a full refund.

The enclosed copy of the SOFTWARE PRODUCT is **never sold**. It is licensed by VVI to the original customer for his or her use only under the terms of this license agreement which follows:

- **1. SCOPE OF LICENSE**
  - This Agreement governs the use of the SOFTWARE PRODUCT and user documentation in printed and electronic forms (the "SOFTWARE PRODUCT"). The SOFTWARE PRODUCT also includes, and this Agreement also governs, later releases, IF ANY, of the SOFTWARE PRODUCT which VVI distributes without additional charge to licensees of your release of the SOFTWARE PRODUCT.
- **2. RESTRICTED USE**
  - STUDENT USE: If you licensed the SOFTWARE PRODUCT for student use you represent to VVI that you are a current member of an accredited educational institution's student body. ("Student User"). Student Users are licensed to use the SOFTWARE PRODUCT in accordance with this Agreement, and solely for the purposes directly related to satisfying the requirements of degree-granting programs ("Student Purposes"). Student Use is use of the SOFTWARE PRODUCT by Student Users and for Student Purposes only. Any use of the SOFTWARE PRODUCT for other than Student Use is expressly prohibited.
  - DEMONSTRATION USE: If you licensed the SOFTWARE PRODUCT for demonstration use then you may only use the SOFTWARE PRODUCT for Demonstration Purposes. Demonstration Purposes shall mean use of the SOFTWARE PRODUCT for the sole limited purpose of verify that the SOFTWARE PRODUCT performs in accordance with manufacture's representations as documented in the SOFTWARE PRODUCT online manuals. Demonstration Purposes shall not mean use for commercial, official university, government laboratory purposes, or any use other than Demonstration Purposes.

  THIS LICENSE WILL TERMINATE AUTOMATICALLY if you fail to comply with the terms and conditions set forth above if such terms are applicable to you.
- **3. LICENSEE's RIGHTS**

  YOU MAY:
  - A. Install and use the SOFTWARE PRODUCT on any computer, as long as it is used only on one computer by one user at a time and in a way which is consistent with this Agreement. If several persons use this SOFTWARE PRODUCT at the same time, or if one person uses it on more than one computer, you must pay one license fee for each copy being used as designated in the Purchase Agreement between you and VVI. You may only use this SOFTWARE PRODUCT on a computer network, if authorized under the Purchase Agreement and if you pay one license fee for each computer and terminal connected to the network.
  - B. Copy the SOFTWARE PRODUCT for back-up purposes only. You may make one (1) copy of the SOFTWARE PRODUCT for back-up purposes. All copies must contain the copyright notice contained in the original copy of the SOFTWARE PRODUCT.
  - C. Transfer of the SOFTWARE PRODUCT to another person or legal entity by any means, including but not limited to assumability, is expressly prohibited.
  - D. Terminate this license by destroying the original and all copies of the SOFTWARE PRODUCT in whatever form.
- **4. PROHIBITED ACTIVITIES**

  YOU MAY NOT:
  - A. Loan, rent, lease, give, sublicense or otherwise transfer the SOFTWARE PRODUCT (or any copy), in whole or in part, to any other person.
  - B. Copy or translate the User Manual included with the SOFTWARE PRODUCT.
  - C. Copy, alter, translate, decompile, or reverse engineer the SOFTWARE PRODUCT, including but not limited to, modify the SOFTWARE PRODUCT to make it operate on non-compatible hardware.
  - D. Remove, alter or cause not to be displayed, any copyright notices or startup messages contained in the SOFTWARE PRODUCT or documentation.

  THIS LICENSE WILL TERMINATE AUTOMATICALLY if you fail to comply with the terms and conditions set forth above.
- **5. OWNERSHIP**

  - SOFTWARE PRODUCT contains software proprietary to VVI. As a licensee, you own the media on which the SOFTWARE PRODUCT is originally or subsequently recorded, but VVI retains title and ownership to the SOFTWARE PRODUCT recorded on the media and all copyright and other intellectual property rights therein. This license is not a sale of the SOFTWARE PRODUCT or any copy.
  - Operations, features and methodologies of graph and data-oriented graphics by user interface oriented creation, manipulation and editing such as by or in relation with, but not exclusive to, a general purpose drawing system, application or software in any combination of the aforementioned is a trademark and tradedress of VVI with all rights reserved in the United States and all international locations.

- **6. WARRANTY**
  - VVI warrants, to you personally, for a period of ninety (90) days from the date of the Purchase Agreement for SOFTWARE PRODUCT subject to this License Agreement (the "Warranty Period"), that the media containing the SOFTWARE PRODUCT shall be free from defects in material and workmanship under normal use. VVI further warrants, for the Warranty Period, that the SOFTWARE PRODUCT shall operate substantially in accordance with the functional specifications in the accompanying documentation if properly used on a machine for which it was designed. If during the Warranty Period a defect in the SOFTWARE PRODUCT or media appears, you may return the SOFTWARE PRODUCT to your place of purchase for either, at the election of VVI, replacement or refund of the amounts paid by you for the license of the SOFTWARE PRODUCT. You agree that the foregoing constitutes your sole and exclusive remedy for breach by VVI of any warranties made under this Agreement.
- **7. DISCLAIMER**
  - Because it is impossible for VVI to know the purposes for which you acquired this SOFTWARE PRODUCT or the uses to which you will put this SOFTWARE PRODUCT, you assume full responsibility for the selection of the SOFTWARE PRODUCT, and for its installation and use and the results of that use.
  - EXCEPT FOR THE LIMITED WARRANTY SET FORTH IN SECTION 6 ABOVE, YOU AGREE THAT THE SOFTWARE PRODUCT IS FURNISHED ON AN "AS-IS" BASIS. NEITHER VVI NOR ANY THIRD PARTY SUPPLIER MAKES ANY WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED, AS TO ANY MATTER WHATSOEVER, INCLUDING WITHOUT LIMITATION THE CONDITION, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE OF THE SOFTWARE PRODUCT. SOME STATES DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. IN THE UNITED KINGDOM THE ABOVE DISCLAIMER OF WARRANTIES DOES NOT AFFECT YOUR STATUTORY RIGHTS AS A CONSUMER. NEITHER VVI NOR ANY THIRD PARTY SUPPLIER WARRANTS THAT THE SOFTWARE PRODUCT WILL MEET YOUR REQUIREMENTS, THAT IT WILL OPERATE IN THE COMBINATIONS WHICH YOU MAY SELECT, OR THAT ITS OPERATION WILL BE UNINTERRUPTED OR ERROR FREE. NEITHER VVI NOR ANY THIRD PARTY SUPPLIER ASSUMES ANY LIABILITY REGARDING USE OF, OR ANY DEFECT IN, THE SOFTWARE PRODUCT.
  - VVI is not responsible for problems caused by changes in the operating characteristics of the hardware or operating system software you are using which are made after the release date of this version of the SOFTWARE PRODUCT, nor for problems in the interaction of the SOFTWARE PRODUCT.
- **8. LIMITATION OF LIABILITY**
  - IN NO EVENT SHALL VVI OR THIRD PARTY SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, EVEN IF SUCH PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.
- **9. EXPORT**
  - You agree not to export or re-export the SOFTWARE PRODUCT, or any portion thereof, without the appropriate United States or foreign government licenses.
- **10. SEVERABILITY**
  - If any portion of this Agreement is held invalid or unenforceable for any reason, such invalidity shall not affect the validity of the remaining provisions of this Agreement, and the parties will substitute for the invalid provisions a valid provision which most closely approximates the intent and economic effect of the valid provision.
- **11. TERMINATION**
  - This Agreement shall be effective until terminated by Licensor. Any failure by you to comply with any of the provisions of this Agreement will be a material breach of this Agreement and entitle VVI to terminate this Agreement immediately. Upon termination, you are to immediately stop all use of the SOFTWARE PRODUCT and to return to VVI or erase all copies of the SOFTWARE PRODUCT in any form (including copies contained in any mass storage device).
- **12. ENTIRE AGREEMENT**
  - This Agreement shall constitute the entire agreement between the parties with respect to the subject matter hereof, and supersedes any prior agreements or understandings between the parties, whether written or oral, with respect hereto. No modification to this Agreement shall be of any force or effect unless made in writing signed by each party.
- **13. APPLICABLE LAW; JURISDICTION; VENUE**
  - This Agreement shall be governed and construed in accordance with the laws of the State of Pennsylvania. The parties agree that Centre County in the State of Pennsylvania shall be the proper venue for any action brought under the Agreement whether in state or federal court. You consent to the personal jurisdiction of such courts.
- **14. U. S. GOVERNMENT END USERS**
  - If the SOFTWARE PRODUCT is acquired by or on behalf of a unit or agency of the United States Government, the following

provisions apply. The documentation and Software licensed under this Agreement is provided with RESTRICTED RIGHTS and constitute restricted computer software, under the Federal Acquisition Regulations, as set forth below. The SOFTWARE PRODUCT: (a) is existing computer software and, was developed at private expense, (b) is a trade secret of VVI for all purposes of the Freedom of Information Act, (c) is "commercial computer software" subject to limited utilization as expressly stated in this Agreement or as provided in the contract between the vendor and the government entity, (d) in all respects is proprietary data belonging solely to VVI and (e) is unpublished and all rights are reserved under the copyright laws of the United States.

- The SOFTWARE PRODUCT is licensed only with "Restricted Rights" and use, reproduction or disclosure is subject to restrictions set forth in Alternate III(g)(3) of the Rights in Data - General Clause at 52.227-14 (June 1987) and subparagraphs (a) through (d) of the Commercial Computer Software - Restricted Rights clause at 52.227-19 (June 1987) of the Federal Acquisition Regulations and their respective successors. For units of the Department of Defense (DoD), this SOFTWARE PRODUCT is licensed only with "Restricted Rights" and use, duplication, or disclosure is subject to restrictions as set forth in subdivision (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at 252.227-7013 (June 1988) of the DoD Supplement to the Federal Acquisition Regulations and its successors.
- Contractor/manufacturer is VVimaging, Inc., 311 Adams Avenue, State College, Pennsylvania 16803.

EULA - 12.11.3 - 1/1/2021

Graph IDE ► Legal ► Cloud Agreement

Below is a copy of the Cloud Agreement that you received with your copy of Graph IDE and was presented to you before purchasing or using the Cloud Service.

VVI Cloud Terms and Conditions

THIS LEGAL AGREEMENT BETWEEN YOU AND VVIMAGING, INC. (dba: VVI) GOVERNS YOUR USE OF THE VVI CLOUD SERVICES. IT IS IMPORTANT THAT YOU READ AND UNDERSTAND THE FOLLOWING TERMS. YOU ARE AGREEING THAT THESE TERMS APPLY IF YOU CHOOSE TO ACCESS OR USE THE SERVICE.

VVI is the provider of the Service, which permits you to utilize certain Internet services, including logging in, storing your personal content and making it accessible on your compatible devices and computers, only under the terms and conditions set forth in this Agreement.

- **(1) DEFINITIONS**
  - (1.A) Account. Account shall mean the user name, hereinafter referred to as VVI ID, and user password that is assigned to you by VVI and enables the associated functionality described herein.
  - (1.B) Service. Service shall mean the VVI Cloud service described herein.
  - (1.C) "VVI" as used herein means VVIMAGING Inc., located at 311 Adams Ave., State College, PA.
- **(2) REQUIREMENTS FOR USE OF THE SERVICE**
  - (2.A) Age. The Service is only available to individuals age 18 years or older (or equivalent minimum age in the relevant jurisdiction) that have the legal authority to accept this contract.
  - (2.B) Capacity. To use the Service, you cannot be a person barred from receiving the Service under the laws of the United States or other applicable jurisdictions, including the country in which you reside or from where you use the Service. By accepting this Agreement, you represent that you understand and agree to the foregoing.
  - (2.C) Devices and Accounts. Use of the Service may require compatible devices, Internet access, and certain software (fees may apply); may require periodic updates; and may be affected by the performance of these factors. VVI reserves the right to limit the number of Accounts that may be created from a device and the number of devices associated with an Account. The latest version of required software may be required for certain transactions or features. You agree that meeting these requirements is your responsibility.
  - (2.D) Limitations on Use. You agree to use the Service only for purposes permitted by this Agreement, and only to the extent permitted by any applicable law, regulation, or generally accepted practice in the applicable jurisdiction. Exceeding any applicable or reasonable limitation of bandwidth, or storage capacity is prohibited and may prevent you from adding or receiving documents or other data. If your use of the Service or other behavior intentionally or unintentionally threatens VVI's ability to provide the Service or other systems, VVI shall be entitled to take all reasonable steps to protect the Service and VVI's systems, which may include suspension of your access to the Service. Repeated violations of the limitations may result in termination of your Account.
  - (2.E) No Associate. If you are a covered entity, business associate or representative of a covered entity or business associate (as those terms are defined at 45 C.F.R § 160.103), You agree that you will not use any component, function or other facility of Service to create, receive, maintain or transmit any "protected health information" (as such term is defined at 45 C.F.R § 160.103) or use Service in any manner that would make VVI Your or any third party's business associate.
  - (2.F) Availability of the Service. The Service, or any feature or part thereof, may not be available in all languages or in all countries and VVI makes no representation that the Service, or any feature or part thereof, is appropriate or available for use in any particular location. To the extent that you choose to access and use the Service, you do so at your own initiative and are responsible for compliance with any applicable laws.
  - (2.G) Changing the Service. VVI reserves the right at any time to modify this Agreement and to impose new or additional terms or conditions on your use of the Service, provided that VVI will give you 30 days' advance notice of any material adverse change to the Service or applicable terms of service, unless it would not be reasonable to do so due to circumstances arising from legal, regulatory, or governmental action; to address user security, user privacy, or technical integrity concerns; to avoid service disruptions to other users; or due to a natural disaster, catastrophic event, war, or other similar occurrence outside of VVI's reasonable control. With respect to paid cloud storage services, VVI will not make any material adverse change to the Service before the end of your current paid term, unless a change is reasonably necessary to address legal, regulatory, or governmental action; to address user security, user privacy, or technical integrity concerns; to avoid service disruptions to other users; or to avoid issues resulting from a natural disaster, a catastrophic event, war, or other similar occurrence outside of VVI's reasonable control. In the event that VVI does make material adverse changes to the Service or terms of use, you will have the right to terminate this Agreement and your account, in which case VVI will provide you with a pro rata refund of any pre-payment for your then-current paid term. VVI shall not be liable to you for any modifications to the Service or terms of service made in accordance with this Section (2.G).
- **(3) FEATURES AND SERVICES**
  - (3.A) Temporary Storage

    VVI shall use reasonable skill and due care in providing the Service, but, TO THE GREATEST EXTENT PERMISSIBLE BY APPLICABLE LAW, VVI DOES NOT GUARANTEE OR WARRANT THAT ANY CONTENT YOU MAY STORE OR ACCESS THROUGH THE SERVICE WILL NOT BE SUBJECT TO INADVERTENT DAMAGE, CORRUPTION, LOSS, OR REMOVAL IN ACCORDANCE WITH THE TERMS OF THIS AGREEMENT, AND VVI SHALL NOT BE RESPONSIBLE SHOULD SUCH DAMAGE, CORRUPTION, LOSS, OR REMOVAL OCCUR. It is your responsibility to maintain appropriate originals, copies and/or backups of your information and data.

- (3.B) File Sharing. The Service is not a file sharing service.
- (3.C) Login. The Service is used as a Login procedure for associated applications, namely Graph IDE CE. Graph IDE CE is only functional after Login.
- (3.D) Effects Of Termination. When the Account is terminated then the Features And Services terminate and are no longer available.

- **(4) SUBSCRIPTION FEES**
  - (4.A) Payment

    VVI, through its payment processor PayPal or other authorized processor, will automatically charge on a recurring basis the fee of Service. YOU ARE RESPONSIBLE FOR THE TIMELY PAYMENT OF ALL FEES AND FOR PROVIDING VVI (OR ITS AUTHORIZED PROCESSOR) WITH VALID CREDIT CARD OR PAYMENT ACCOUNT DETAILS FOR PAYMENT OF ALL FEES. If VVI is unable to successfully charge your credit card or payment account for fees due, VVI reserves the right to revoke or restrict access to your stored Content, delete your stored Content, or terminate your Account. If you want to designate a different credit card or payment account or if there is a change in your credit card or payment account status, you must change your information online; this may temporarily disrupt your access to the Services while VVI verifies your new payment information. VVI may contact you via email regarding your account, for reasons including, without limitation, reaching or exceeding your storage limit.

    VVI is only responsible for collecting sales tax in the Commonwealth Of Pennsylvania. For any other jurisdiction You are responsible for sales, use, value added (VAT) tax and any other tax or licensing fee associated with that other jurisdiction.
  - (4.B) Right of Withdrawal

    If you choose to cancel your subscription following its initial purchase or, if you are on an annual payment plan, following the commencement of any renewal term, you may do so by informing VVI with a clear statement within 14 days from when you received your e-mail confirmation by contacting Customer Support. You do not need to provide a reason for cancellation.

    To meet the cancellation deadline, you must send your communication of cancellation before the 14-day period has expired.

    Effects of cancellation. Your account will be terminated.
- **(5) Your Use of the Service**
  - (5.A) Your Account

    As a registered user of the Service, you must establish an Account. Do not reveal your Account information to anyone else. You are solely responsible for maintaining the confidentiality and security of your Account and for all activities that occur on or through your Account, and you agree to immediately notify VVI of any security breach of your Account. You further acknowledge and agree that the Service is designed and intended for personal use on an individual basis and you should not share your Account and/or password details with another individual. Provided VVI has exercised reasonable skill and due care, VVI shall not be responsible for any losses arising out of the unauthorized use of your Account resulting from you not following these rules.

    In order to use the Service, you must enter your VVI ID and password to authenticate your Account. You agree to provide accurate and complete information when you register with, and as you use, the Service ("Service Registration Data"), and you agree to update your Service Registration Data to keep it accurate and complete. Failure to provide accurate, current and complete Service Registration Data may result in the suspension and/or termination of your Account. You agree that VVI may store and use the Service Registration Data you provide for use in maintaining and billing fees to your Account.
  - (5.B) Use of Other VVI Products and Services

    Particular components or features of the Service provided by VVI and/or its licensors require separate software or other license agreements or terms of use. You must read, accept, and agree to be bound by any such separate agreement as a condition of using these particular components or features of the Service.
  - (5.C) No Conveyance

    Nothing in this Agreement shall be construed to convey to you any interest, title, or license in a VVI ID, domain name or similar resource used by you in connection with the Service.
  - (5.D) No Right of Survivorship

    Unless otherwise required by law, You agree that your Account is non-transferable and that any rights to your VVI ID or Content within your Account terminate upon your death. Upon receipt of a copy of a death certificate your Account may be terminated and all Content within your Account deleted. Contact VVI Support at support@vvi.com for further assistance.
  - (5.E) No Resale of Service

    You agree that you will not reproduce, copy, duplicate, sell, resell, rent or trade the Service (or any part thereof) for any purpose.
  - (5.F) No Encryption

    Data, including documents and user information, is not encrypted. As such, do not transmit or store confidential, personal or other information that necessitates encryption. If you have specialized security concerns then please contact support@vvi.com for information on on-premises cloud service which is not part of the Service of this Agreement.
- **(6) Content and Your Conduct**

- (6.A) Content

  "Content" means any information that may be generated or encountered through use of the Service, such as data files, device characteristics, written text, software, music, graphics, photographs, images, sounds, videos, messages and any other like materials. You understand that all Content, whether publicly posted or privately transmitted on the Service is the sole responsibility of the person from whom such Content originated. This means that you, and not VVI, are solely responsible for any Content you upload, download, post, email, transmit, store or otherwise make available through your use of the Service. You understand that by using the Service you may encounter Content that you may find offensive, indecent, or objectionable, and that you may expose others to Content that they may find objectionable. VVI does not control the Content posted via the Service, nor does it guarantee the accuracy, integrity or quality of such Content. You understand and agree that your use of the Service and any Content is solely at your own risk.

- (6.B) Your Conduct

  You agree that you will NOT use the Service to:

  (a) upload, download, post, email, transmit, store or otherwise make available any Content that is unlawful, harassing, threatening, harmful, tortious, defamatory, libelous, abusive, violent, obscene, vulgar, invasive of another's privacy, hateful, racially or ethnically offensive, or otherwise objectionable;

  (b) stalk, harass, threaten or harm another;

  (c) if you are an adult, request personal or other information from a minor (any person under the age of 18 or such other age as local law defines as a minor) who is not personally known to you, including but not limited to any of the following: full name or last name, home address, zip/postal code, telephone number, picture, or the names of the minor's school, church, athletic team or friends;

  (d) pretend to be anyone, or any entity, you are not - you may not impersonate or misrepresent yourself as another person (including celebrities), entity, another VVI Cloud user, a VVI employee, or a civic or government leader, or otherwise misrepresent your affiliation with a person or entity (VVI reserves the right to reject or block any VVI ID or email address which could be deemed to be an impersonation or misrepresentation of your identity, or a misappropriation of another person's name or identity);

  (e) engage in any copyright infringement or other intellectual property infringement (including uploading any content to which you do not have the right to upload), or disclose any trade secret or confidential information in violation of a confidentiality, employment, or nondisclosure agreement;

  (f) post, send, transmit or otherwise make available any unsolicited or unauthorized email messages, advertising, promotional materials, junk mail, spam, or chain letters, including, without limitation, bulk commercial advertising and informational announcements;

  (g) forge any TCP-IP packet header or any part of the header information in an email or a news group posting, or otherwise putting information in a header designed to mislead recipients as to the origin of any Content transmitted through the Service ("spoofing");

  (h) upload, post, email, transmit, store or otherwise make available any material that contains viruses or any other computer code, files or programs designed to harm, interfere or limit the normal operation of the Service (or any part thereof), or any other computer software or hardware;

  (i) interfere with or disrupt the Service (including accessing the Service through any automated means, like scripts or web crawlers), or any servers or networks connected to the Service, or any policies, requirements or regulations of networks connected to the Service (including any unauthorized access to, use or monitoring of data or traffic thereon);

  (j) plan or engage in any illegal activity; and/or

  (k) gather and store personal information on any other users of the Service to be used in connection with any of the foregoing prohibited activities.

- (6.C) Removal of Content

  You acknowledge that VVI is not responsible or liable in any way for any Content provided by others and has no duty to pre-screen such Content. However, VVI reserves the right at all times to determine whether Content is appropriate and in compliance with this Agreement, and may pre-screen, move, refuse, modify and/or remove Content at any time, without prior notice and in its sole discretion, if such Content is found to be in violation of this Agreement or is otherwise objectionable.

- (6.D) Back up Your Content

  You are responsible for backing up, to your own computer or other device, any important documents, images or other Content that you store or access via the Service. VVI shall use reasonable skill and due care in providing the Service, but VVI does not guarantee or warrant that any Content you may store or access through the Service will not be subject to inadvertent damage, corruption or loss.

- (6.E) Access to Your Account and Content

  VVI reserves the right to take steps VVI believes are reasonably necessary or appropriate to enforce and/or verify compliance with any part of this Agreement. You acknowledge and agree that VVI may, without liability to you, access, use, preserve and/or disclose your Account information and Content to law enforcement authorities, government officials, and/or a third party, as VVI believes is reasonably necessary or appropriate, if legally required to do so or if VVI has a good faith belief that such access, use, disclosure, or preservation is reasonably necessary to: (a) comply with legal process or request; (b) enforce this Agreement, including investigation of any potential violation thereof; (c) detect, prevent or otherwise address security, fraud or technical issues; or (d) protect the rights, property or safety of VVI, its users, a third party, or the public as required or permitted by law.

- (6.F) Copyright Notice - DMCA

  If you believe that any Content in which you claim copyright has been infringed by anyone using the Service, please contact VVI. VVI may, in its sole discretion, suspend and/or terminate Accounts of users that are found to be repeat infringers.

- (6.G) Violations of this Agreement

  If while using the Service, you encounter Content you find inappropriate, or otherwise believe to be a violation of this Agreement, you may report it by sending an email to support@vvi.com.

- (6.H) Content Submitted or Made Available by You on the Service

  (a) License from You. Except for material VVI may license to you, VVI does not claim ownership of the materials and/or Content you submit or make available on the Service. However, by submitting or posting such Content on areas of the Service that are accessible by the public or other users with whom you consent to share such Content, you grant VVI a worldwide, royalty-free, non-exclusive license to use, distribute, reproduce, modify, adapt, publish, translate, publicly perform and publicly display such Content on the Service solely for the purpose for which such Content was submitted or made available, without any compensation or obligation to you. You agree that any Content submitted or posted by you shall be your sole responsibility, shall not infringe or violate the rights of any other party or violate any laws, contribute to or encourage infringing or otherwise unlawful conduct, or otherwise be obscene, objectionable, or in poor taste. By submitting or posting such Content on areas of the Service that are accessible by the public or other users, you are representing that you are the owner of such material and/or have all necessary rights, licenses, and authorization to distribute it.

  (b) Changes to Content. You understand that in order to provide the Service and make your Content available thereon, VVI may transmit your Content across various public networks, in various media, and modify or change your Content to comply with technical requirements of connecting networks or devices or computers. You agree that the license herein permits VVI to take any such actions.

  (c) Trademark Information. VVI, the VVI logo and other VVI trademarks, service marks, graphics, and logos used in connection with the Service are trademarks or registered trademarks of VVIMAGING Inc. in the US and/or other countries. A list of VVI's trademarks can be found here - http://www.vvi.com/legal. Other trademarks, service marks, graphics, and logos used in connection with the Service may be the trademarks of their respective owners. You are granted no right or license in any of the aforesaid trademarks, and further agree that you shall not remove, obscure, or alter any proprietary notices (including trademark and copyright notices) that may be affixed to or contained within the Service.

- **(7) Software**

  - (7.A) VVI's Proprietary Rights. You acknowledge and agree that VVI and/or its licensors own all legal right, title and interest in and to the Service, including but not limited to graphics, user interface, the scripts and software used to implement the Service, and any software provided to you as a part of and/or in connection with the Service (the "Software"), including any and all intellectual property rights that exist therein, whether registered or not, and wherever in the world they may exist. You further agree that the Service (including the Software, or any other part thereof) contains proprietary and confidential information that is protected by applicable intellectual property and other laws, including but not limited to copyright. You agree that you will not use such proprietary information or materials in any way whatsoever except for use of the Service in compliance with this Agreement. No portion of the Service may be reproduced in any form or by any means, except as expressly permitted in these terms.

  - (7.B) License From VVI. THE USE OF THE SOFTWARE OR ANY PART OF THE SERVICE, EXCEPT FOR USE OF THE SERVICE AS PERMITTED IN THIS AGREEMENT, IS STRICTLY PROHIBITED AND INFRINGES ON THE INTELLECTUAL PROPERTY RIGHTS OF OTHERS AND MAY SUBJECT YOU TO CIVIL AND CRIMINAL PENALTIES, INCLUDING POSSIBLE MONETARY DAMAGES, FOR COPYRIGHT INFRINGEMENT.

  - (7.C) Export Control. Use of the Service and Software, including transferring, posting, or uploading data, software or other Content via the Service, may be subject to the export and import laws of the United States and other countries. You agree to comply with all applicable export and import laws and regulations. In particular, but without limitation, the Software may not be exported or re-exported (a) into any U.S. embargoed countries or (b) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Department of Commerce Denied Person's List or Entity List. By using the Software or Service, you represent and warrant that you are not located in any such country or on any such list. You also agree that you will not use the Software or Service for any purposes prohibited by United States law, including, without limitation, the development, design, manufacture or production of missiles, nuclear, chemical or biological weapons. You further agree not to upload to your Account any data or software that is: (a) subject to International Traffic in Arms Regulations; or (b) that cannot be exported without prior written government authorization, including, but not limited to, certain types of encryption software and source code, without first obtaining that authorization. This assurance and commitment shall survive termination of this Agreement.

  - (7.D) Updates. From time to time, VVI may update the Software used by the Service. In order to continue your use of the Service, such updates may be automatically downloaded and installed onto your device or computer. These updates may include bug fixes, feature enhancements or improvements, or entirely new versions of the Software.

- **(8) Termination**

  - (8.A) Voluntary Termination by You

    You may delete your VVI ID and/or stop using the Service at any time. To terminate your Account and delete your VVI ID, contact VVI Support at support@vvi.com. If you terminate your Account and delete your VVI ID, you will not have access to other VVI products and services with that VVI ID. This action may be non-reversible. Any fees paid by you prior to your termination are nonrefundable (except as expressly permitted otherwise by this Agreement), including any fees paid in advance for the billing year during which you terminate. Termination of your Account shall not relieve you of any obligation to pay any accrued fees or charges.

  - (8.B) Termination by VVI

VVI may at any time and at its sole discretion immediately terminate or suspend all or a portion of your Account and/or access to the Service. Cause for such termination shall include: (a) violations of this Agreement or any other policies or guidelines that are referenced herein and/or posted on the Service; (b) a request by you to cancel or terminate your Account; (c) a request and/or order from law enforcement, a judicial body, or other government agency; (d) where provision of the Service to you is or may become unlawful; (e) unexpected technical or security issues or problems; (f) your participation in fraudulent or illegal activities; (g) failure to pay any fees owed by you in relation to the Service, provided that in the case of non-material breach, VVI will be permitted to terminate only after giving you 30 days' notice and only if you have not cured the breach within such 30-day period; (h) insolvency; or (i) any other condition solely at the discretion of VVI. Any such termination or suspension shall be made by VVI in its sole discretion and VVI will not be responsible to you or any third party for any damages that may result or arise out of such termination or suspension of your Account and/or access to the Service. In addition, VVI may terminate your Account upon 30 days' prior notice via email to the address associated with your Account if (a) your Account has been inactive for one (1) year; or (b) there is a general discontinuance of the Service or any part thereof. Notice of general discontinuance of service will be provided as set forth herein, unless it would not be reasonable to do so due to circumstances arising from legal, regulatory, or governmental action; to address user security, user privacy, or technical integrity concerns; to avoid service disruptions to other users; or due to a natural disaster, a catastrophic event, war, or other similar occurrence outside of VVI's reasonable control. In the event of such termination no refund shall be provided. VVI shall not be liable to you for any modifications to the Service or terms of service in accordance with this Section (8.B).

- (8.C) Effects of Termination Upon termination of your Account you may lose all access to the Service and any portions thereof, including, but not limited to, your Account, VVI ID, Content and use of any related client such as the Graph IDE CE application. In addition, after a period of time, VVI will delete information and data stored in or as a part of your account(s). Any individual components of the Service that you may have used subject to separate software license agreements will also be terminated in accordance with those license agreements.

- **(9) Links and Other Third Party Materials**

  Certain Content, components or features of the Service may include materials from third parties and/or hyperlinks to other web sites, resources or Content. Because VVI may have no control over such third party sites and/or materials, you acknowledge and agree that VVI is not responsible for the availability of such sites or resources, and does not endorse or warrant the accuracy of any such sites or resources, and shall in no way be liable or responsible for any Content, advertising, products or materials on or available from such sites or resources. You further acknowledge and agree that VVI shall not be responsible or liable in any way for any damages you incur or allege to have incurred, either directly or indirectly, as a result of your use and/or reliance upon any such Content, advertising, products or materials on or available from such sites or resources.

- **(10) DISCLAIMER OF WARRANTIES; LIMITATION OF LIABILITY**

  SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF CERTAIN WARRANTIES, AS SUCH, TO THE EXTENT SUCH EXCLUSIONS ARE SPECIFICALLY PROHIBITED BY APPLICABLE LAW, SOME OF THE EXCLUSIONS SET FORTH BELOW MAY NOT APPLY TO YOU.

  VVI SHALL USE REASONABLE SKILL AND DUE CARE IN PROVIDING THE SERVICE. THE FOLLOWING DISCLAIMERS ARE SUBJECT TO THIS EXPRESS WARRANTY.

  VVI DOES NOT GUARANTEE, REPRESENT, OR WARRANT THAT YOUR USE OF THE SERVICE WILL BE UNINTERRUPTED OR ERROR-FREE, AND YOU AGREE THAT FROM TIME TO TIME VVI MAY REMOVE THE SERVICE FOR INDEFINITE PERIODS OF TIME, OR CANCEL THE SERVICE IN ACCORDANCE WITH THE TERMS OF THIS AGREEMENT.

  YOU EXPRESSLY UNDERSTAND AND AGREE THAT THE SERVICE IS PROVIDED ON AN "AS IS" AND "AS AVAILABLE" BASIS. VVI AND ITS AFFILIATES, SUBSIDIARIES, OFFICERS, DIRECTORS, EMPLOYEES, AGENTS, PARTNERS AND LICENSORS EXPRESSLY DISCLAIM ALL WARRANTIES OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. IN PARTICULAR, VVI AND ITS AFFILIATES, SUBSIDIARIES, OFFICERS, DIRECTORS, EMPLOYEES, AGENTS, PARTNERS AND LICENSORS MAKE NO WARRANTY THAT (I) THE SERVICE WILL MEET YOUR REQUIREMENTS; (II) YOUR USE OF THE SERVICE WILL BE TIMELY, UNINTERRUPTED, SECURE OR ERROR-FREE; (III) ANY INFORMATION OBTAINED BY YOU AS A RESULT OF THE SERVICE WILL BE ACCURATE OR RELIABLE; AND (IV) ANY DEFECTS OR ERRORS IN THE SOFTWARE PROVIDED TO YOU AS PART OF THE SERVICE WILL BE CORRECTED.

  VVI DOES NOT REPRESENT OR GUARANTEE THAT THE SERVICE WILL BE FREE FROM LOSS, CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, OR OTHER SECURITY INTRUSION, AND VVI DISCLAIMS ANY LIABILITY RELATING THERETO.

  ANY MATERIAL DOWNLOADED OR OTHERWISE OBTAINED THROUGH THE USE OF THE SERVICE IS ACCESSED AT YOUR OWN DISCRETION AND RISK, AND YOU WILL BE SOLELY RESPONSIBLE FOR ANY DAMAGE TO YOUR DEVICE, COMPUTER, OR LOSS OF DATA THAT RESULTS FROM THE DOWNLOAD OF ANY SUCH MATERIAL. YOU FURTHER ACKNOWLEDGE THAT THE SERVICE IS NOT INTENDED OR SUITABLE FOR USE IN SITUATIONS OR ENVIRONMENTS WHERE THE FAILURE OR TIME DELAYS OF, OR ERRORS OR INACCURACIES IN, THE CONTENT, DATA OR INFORMATION PROVIDED BY THE SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE.

  LIMITATION OF LIABILITY

  SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY BY SERVICE PROVIDERS. TO THE EXTENT SUCH EXCLUSIONS OR LIMITATIONS ARE SPECIFICALLY PROHIBITED BY APPLICABLE LAW, SOME OF THE EXCLUSIONS OR LIMITATIONS SET FORTH BELOW MAY NOT APPLY TO YOU.

  VVI SHALL USE REASONABLE SKILL AND DUE CARE IN PROVIDING THE SERVICE. THE FOLLOWING LIMITATIONS DO NOT APPLY IN RESPECT OF LOSS RESULTING FROM (A) VVI'S FAILURE TO USE REASONABLE SKILL AND DUE CARE; (B) VVI'S GROSS NEGLIGENCE, WILLFUL MISCONDUCT OR FRAUD; OR (C) DEATH OR PERSONAL INJURY.

  YOU EXPRESSLY UNDERSTAND AND AGREE THAT VVI AND ITS AFFILIATES, SUBSIDIARIES, OFFICERS, DIRECTORS, EMPLOYEES, AGENTS, PARTNERS AND LICENSORS SHALL NOT BE LIABLE TO YOU FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES, INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF PROFITS, GOODWILL, USE, DATA, COST OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR OTHER INTANGIBLE LOSSES (EVEN IF VVI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES), RESULTING FROM: (I) THE USE OR INABILITY TO USE THE SERVICE (II) ANY CHANGES MADE TO THE SERVICE OR ANY TEMPORARY OR PERMANENT CESSATION OF THE SERVICE OR ANY PART THEREOF; (III)

THE UNAUTHORIZED ACCESS TO OR ALTERATION OF YOUR TRANSMISSIONS OR DATA; (IV) THE DELETION OF, CORRUPTION OF, OR FAILURE TO STORE AND/OR SEND OR RECEIVE YOUR TRANSMISSIONS OR DATA ON OR THROUGH THE SERVICE; (V) STATEMENTS OR CONDUCT OF ANY THIRD PARTY ON THE SERVICE; AND (VI) ANY OTHER MATTER RELATING TO THE SERVICE.

INDEMNIFICATION

You agree to defend, indemnify and hold VVI, its affiliates, subsidiaries, directors, officers, employees, agents, partners, contractors, and licensors harmless from any claim or demand, including reasonable attorneys' fees, made by a third party, relating to or arising from: (a) any Content you submit, post, transmit, or otherwise make available through the Service; (b) your use of the Service; (c) any violation by you of this Agreement; (d) any action taken by VVI as part of its investigation of a suspected violation of this Agreement or as a result of its finding or decision that a violation of this Agreement has occurred; or (e) your violation of any rights of another. This means that you cannot sue VVI, its affiliates, subsidiaries, directors, officers, employees, agents, partners, contractors, and licensors as a result of its decision to remove or refuse to process any information or Content, to warn you, to suspend or terminate your access to the Service, or to take any other action during the investigation of a suspected violation or as a result of VVI's conclusion that a violation of this Agreement has occurred. This waiver and indemnity provision applies to all violations described in or contemplated by this Agreement. This obligation shall survive the termination or expiration of this Agreement and/or your use of the Service. You acknowledge that you are responsible for all use of the Service using your Account, and that this Agreement applies to any and all usage of your Account. You agree to comply with this Agreement and to defend, indemnify and hold harmless VVI from and against any and all claims and demands arising from usage of your Account, whether or not such usage is expressly authorized by you.

- **(11) GENERAL**
  - (11.A) Notices

    VVI may provide you with notices regarding the Service, including changes to this Agreement, by email to your email address associated with your Account, iMessage or SMS, by regular mail, or by postings on our website and/or the Service.
  - (11.B) Governing Law

    You agree that any dispute in the meaning, effect or validity of this Agreement shall be resolved in accordance with the laws of the Commonwealth of Pennsylvania without regard to the conflict of laws provisions thereof. You further agree that if one or more provisions of this Agreement are held to be illegal or unenforceable under applicable Pennsylvania law, such illegal or unenforceable portion(s) shall be limited or excluded from this Agreement to the minimum extent required so that this Agreement shall otherwise remain in full force and effect and enforceable in accordance with its terms. The parties agree that Centre County in the State of Pennsylvania shall be the proper venue for any action brought under the Agreement whether in state or federal court. You consent to the personal jurisdiction of such courts.
  - (11.C) Entire Agreement

    This Agreement constitutes the entire agreement between you and VVI, governs your use of the Service and completely replaces any prior agreements between you and VVI in relation to the Service. You may also be subject to additional terms and conditions that may apply when you use affiliate services, third-party content, or third-party software. If any part of this Agreement is held invalid or unenforceable, that portion shall be construed in a manner consistent with applicable law to reflect, as nearly as possible, the original intentions of the parties, and the remaining portions shall remain in full force and effect. The failure of VVI to exercise or enforce any right or provision of this Agreement shall not constitute a waiver of such right or provision. You agree that, except as otherwise expressly provided in this Agreement, there shall be no third-party beneficiaries to this agreement.
  - (11.D) ELECTRONIC CONTRACTING Your use of the Service includes the ability to enter into agreements and/or to make transactions electronically. YOU ACKNOWLEDGE THAT YOUR ELECTRONIC SUBMISSIONS CONSTITUTE YOUR AGREEMENT AND INTENT TO BE BOUND BY AND TO PAY FOR SUCH AGREEMENTS AND TRANSACTIONS. YOUR AGREEMENT AND INTENT TO BE BOUND BY ELECTRONIC SUBMISSIONS APPLIES TO ALL RECORDS RELATING TO ALL TRANSACTIONS YOU ENTER INTO ON THIS SERVICE, INCLUDING NOTICES OF CANCELLATION, POLICIES, CONTRACTS, AND APPLICATIONS. In order to access and retain your electronic records, you may be required to have certain hardware and software, which are your sole responsibility.

Last revised: March 29, 2018

CA - 12.9.2 - 3/29/2010

Graph IDE ► Legal ► Redistribution License Agreement

Below is a copy of the Vvidget Library Limited Redistribution License Agreement. If you incorporate Vvidget into your own applications and also distribute any of those applications to others then you are subject to the following license. Your action of distribution is an implicit acceptance of the following license agreement. You must also have a valid written and signed Purchase Agreement for the "Vvidget Library Redistribution" product executed by an authorized representative of VVI. Vvidget is commercial software and requires a Purchase Agreement, separate and in addition to the one for the EULA, to redistribute. Please email sales@vvi.com for purchase instructions before redistributing.

Vvidget™ Library Redistribution v12.10.3

VVIDGET LIBRARY LIMITED REDISTRIBUTION LICENSE AGREEMENT

This Redistribution License Agreement (the "Agreement") is made and entered into the Effective Date, by and between Developer and VVimaging, Inc., (hereinafter referred to as VVI), a Pennsylvania corporation, having principal place of business at 311 Adams Ave. State College, PA 16803.

VVI and Developer Agree as follows:

DEFINITIONS

For the purpose of this Agreement,

- "Purchase Agreement" means a separate agreement between Developer and VVI for purchase of the "Vvidget Library Redistribution" rights which is signed by an authorized representative of VVI.
- "Developer" means the individual that accepts this Agreement and has a valid Purchase Agreement with VVI.
- "Redistributables" means VVI software in binary form. For OSX (Mac computers) those binaries are in the folder "`/Applications/Graph Builder.app/Contents/Frameworks`" and are the Frameworks `Vvidget_GG.framework`, `Vvidget_GS.framework`, `Vvidget_SAF.framework`, `Vvidget_SAG.framework`, `Vvidget_SAM.framework`, `Vvidget_SAT.framework`, `Vvidget_SBM.framework`, `VvidgetCode.framework`. For iOS (iPhone, iPad, iPod) those binaries are `libVvidget.a` ARM assembly as contained in the example documentation. For the OSX frameworks, the Resources and Header folders must be removed before distribution. Redistributables does not include code signing assets.
- "EULA" means the VVI End User License Agreement which governs the use of Redistributables and is included with each copy of Redistributables.
- "Developer's Original Application" means software application developed by Developer which uses Redistributables and which adds significant and primary functionality to Redistributables.
- "Customer" means third-party which obtains Redistributables from Developer for use solely in a manner consistent with the EULA and as part of Developer's Original Application.
- "Party" means VVI or Developer.
- "Effective Date" is the date of the Purchase Agreement.

RECITALS

Whereas

- VVI has developed Redistributables and legally markets, distributes and sells Redistributables and represents and warrants that it has not been notified that any intellectual property related to Redistributables as contemplated in this Agreement violates any copyright, patent, trade secret, or trademarks of any person or organization;
- Developer wishes to redistribute Redistributables for lawful use, including commercial sale and educational use; and
- VVI wishes to grant Developer rights to redistribute Redistributables consistent with this Agreement.

TERMS AND CONDITIONS

Wherefore, subject to the terms and conditions of this Agreement:

- 1. GENERAL.
    - 1.1. Grant. VVI grants Developer and Developer hereby accepts a limited, non-exclusive, non-transferable, non-assignable, non-refundable, world-wide right to redistribute Redistributables to Customer.
    - 1.2. Proprietary Rights Notices. Developer shall reproduce all proprietary name, trademark, tradedress, patent and copyright notices present in Redistributables without modification or alteration.
    - 1.3. Customer License. Developer shall redistribute Redistributables only on an As-Is basis without warranty of any kind and pursuant to a written license agreement which protects VVI to the same or greater degree than does EULA. Developer shall display Developer own valid copyright notice which shall be sufficient to protect VVI's copyright in Redistributables.
    - 1.4. Marketing. Developer shall not use the name, logo, or trademarks of VVI to market Redistributables other than to indicate truthfully that Developer's Original Application is compatible with Redistributables.
    - 1.5. Re-redistribution. Developer agrees not to permit further distribution of the Redistributables by Customer. Developer may permit further redistribution of the Redistributables by Developer's distributors to Customers if Developer's distributors only distribute the Redistributables in conjunction with, and as part of, the Developer's Original Application and Developer and Developer's distributors comply with the terms of the EULA.
- 2. ENFORCEMENT.
    - 2.1. Indemnity. Developer agrees to indemnify, hold harmless and defend VVI from and against any claims or lawsuits, including attorney's fees, that arise or result from the use or distribution of Developer's Original Application.
    - 2.2. Limitation of Liability. EACH PARTY AGREES THAT ANY LIABILITY ON THE PART OF THE OTHER PARTY ARISING OUT OF OR RELATED TO THIS AGREEMENT FOR ANY CAUSE OF ACTION WHATSOEVER (REGARDLESS OF THE FORM OF ACTION INCLUDING BREACH OF CONTRACT, STRICT LIABILITY, TORT INCLUDING NEGLIGENCE OR ANY OTHER LEGAL OR EQUITABLE THEORY), SHALL BE LIMITED TO PARTY'S DIRECT DAMAGES IN AN AMOUNT NOT TO EXCEED THE TOTAL FEE AMOUNT RECEIVED BY

OTHER PARTY UNDER THIS AGREEMENT FOR THE ASSOCIATED PRODUCT DURING THE INITIAL TERM, IF NO RENEWAL TERM HAS COMMENCED, OR DURING THE MOST RECENT RENEWAL TERM. EACH PARTY AGREES THAT IN NO EVENT SHALL OTHER PARTY BE LIABLE FOR DAMAGES IN RESPECT OF INCIDENTAL, ORDINARY, PUNITIVE, EXEMPLARY, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES EVEN IF PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES INCLUDING, BUT NOT LIMITED TO, BUSINESS INTERRUPTION, LOST BUSINESS REVENUE, LOST PROFITS, FAILURE TO REALIZE EXPECTED SAVINGS, ECONOMIC LOSS, LOSS OF DATA, LOSS OF BUSINESS OPPORTUNITY OR ANY CLAIM AGAINST PARTY BY ANY OTHER PARTY. THIS SECTION STATES EACH PARTY'S ENTIRE LIABILITY AND THE SOLE REMEDIES WITH RESPECT TO ANY THIRD PARTY CLAIMS, INFRINGEMENT AND ALLEGATIONS OF INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OR OTHER PROPRIETARY RIGHTS OF ANY KIND.
    - 2.3. Injunctive Relief. Developer agrees that VVI shall be entitled, without waiving any additional rights or remedies otherwise available to VVI at law, or in equity or by statute, to injunctive and other equitable relief in the event of a breach or intended or threatened breach by Developer of any of the covenants pertaining to the Developer set forth herein.
    - 2.4. Severability. Developer agrees that any dispute in the meaning, effect or validity of this Agreement shall be resolved in accordance with the laws of the Commonwealth of Pennsylvania without regard to the conflict of laws provisions thereof. Developer further agrees that if one or more provisions of this Agreement are held to be illegal or unenforceable under applicable Pennsylvania law, such illegal or unenforceable portion(s) shall be limited or excluded from this Agreement to the minimum extent required so that this Agreement shall otherwise remain in full force and effect and enforceable in accordance with its terms. The parties agree that Centre County in the State of Pennsylvania shall be the proper venue for any action brought under the Agreement whether in state or federal court. You consent to the personal jurisdiction of such courts.

- 3. MISCELLANEOUS PROVISIONS.

    - 3.1. Ownership. Title and ownership in and to the Redistributables shall remain the sole and exclusive property of VVI and its licensors, as applicable. No ownership right, title or interest in Redistributables is transferred to Developer by this Agreement. All rights not expressly granted herein by VVI are reserved by and to VVI.
    - 3.2. Accept VVI EULA. Except as provided herein, Developer also accepts all End User License Agreements accompanying VVI products obtained by Developer.
    - 3.3 Noncontravention. Developer represents that Developer's performance of all the terms of this Agreement will not breach any agreement prior to this Agreement. Developer has not entered into, and Developer agrees Developer will not enter into, any agreement either written or oral in conflict with this Agreement.
    - 3.4 Independent Parties. VVI and Developer are independent contracting parties. Neither Party nor Party's employees, consultants, contractors or agents are agents, employees of other Party, nor do they have any authority to bind other Party by contract or otherwise to any obligation. Developer agrees not to make any statements which state or imply that Developer certifies or guarantees Redistributables or that Redistributables is warranted, tested or approved by Developer.
    - 3.5 Term and Termination. The term of this Agreement shall commence on the Effective Date and shall continue in perpetuity until terminated as set forth below:
        - a. This Agreement shall terminate immediately if Developer breaches any term of this Agreement.
        - b. Terms in Sections 2 and 3 survive termination of this Agreement.
        - c. Termination by either Party will not affect the rights of any Customer under the terms of the End User License Agreement (EULA).
    - 3.6 Binding Effect. This Agreement shall be effective as of the date Developer executes this Agreement and shall be binding upon Developer, Developer's heirs, executors, assigns, and administrators and shall inure to the benefit of VVI, successors and assigns.
    - 3.7 Modifications. This Agreement can only be modified by a subsequent written and signed agreement executed by Developer and an authorized representative of VVI.

LRLA - 10.9.14 - 1/24/2014

Graph IDE ► Legal ► Trademarks And Legal

Manipulating graphs and data graphics and other manipulations unique to Vvidget is a Trademark and Tradedress of VVimaging, Inc (VVI) in the United States and world-wide.

VVI is a registered trademark of VVimaging, Inc (VVI). Peer Visual, Vving, Vvidget, OpenGraph, Graph IDE, Graph Builder, VVI, VVimaging, SAM, "State Automation", "State AutoMation", and Vvidget with any word combination are trademarks of VVimaging, Inc (VVI) in the United States and world-wide.

The OpenGraph logo, Vvidget logo, Graph IDE logo, Graph Builder logo, VVI logo are registered trademarks or trademarks of VVimaging, Inc (VVI).

Apple, Power Macintosh, Macintosh, WebObjects are trademarks or registered trademarks of Apple Computer, Inc. Microsoft and Windows are registered trademarks of Microsoft, Inc. All other trademarks and service marks belong to their respective owners.

VVI has tried to make the information contained in this manual as accurate and reliable as possible. Nevertheless, VVI disclaims any warranty of any kind, whether express or implied, as to any matter whatsoever relating to this information, including without limitation the merchantability or fitness for any particular purpose. VVI will from time to time revise the software described in this manual and reserves the right to make such changes without obligation to notify the purchaser. In no event shall VVI be liable for any indirect, special, incidental, or consequential damages arising out of purchase or use of this manual or the information contained herein.

The following is a list of credits.

OpenSSL Credits

---

libffi Credits
