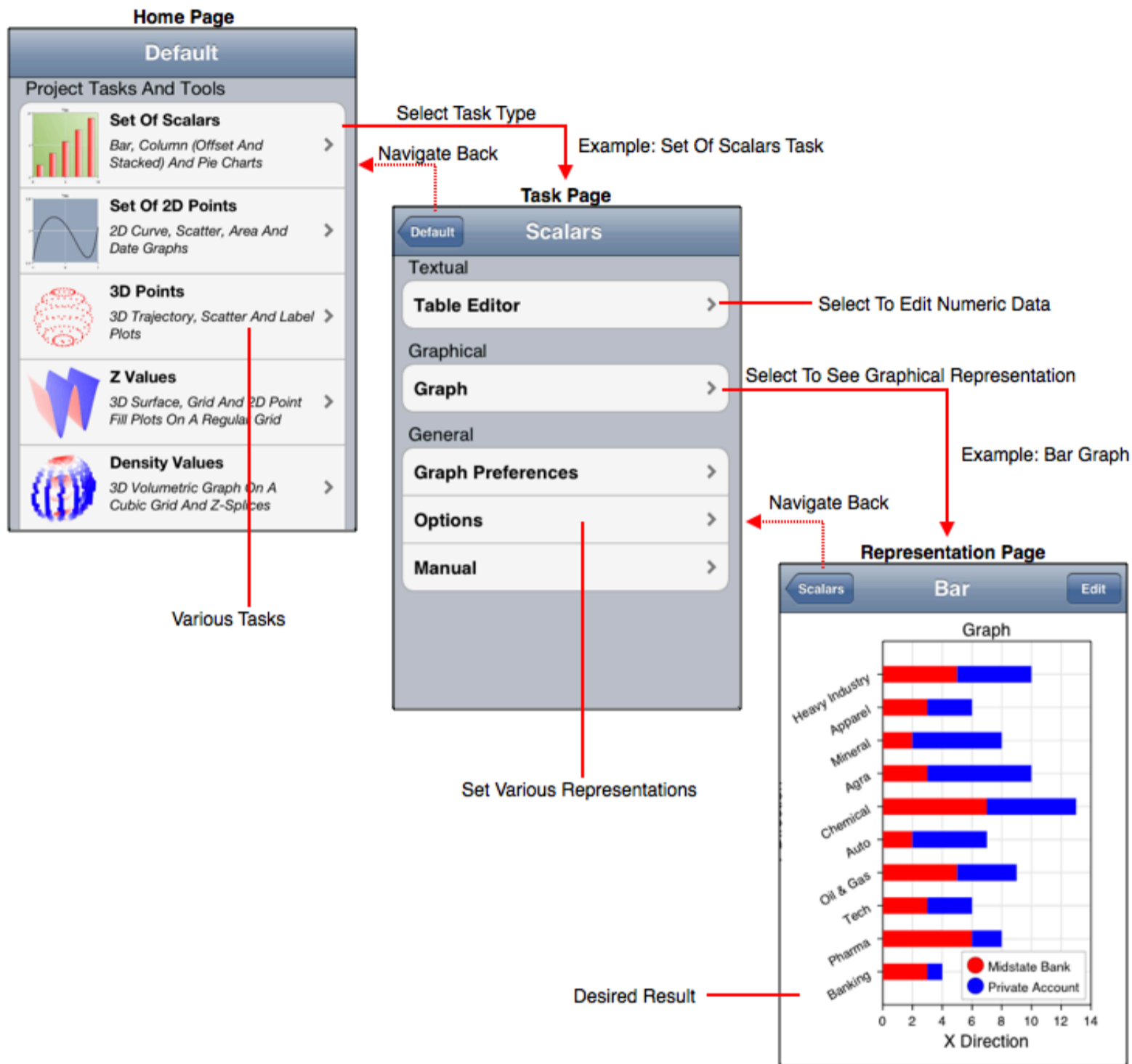


Graph And Vwidget User Manual

For iOS (iPhone, iPad and iPod touch)

Revision 1.3.1 (10.8.3)
 — August 26, 2012 —
 By VVI ®
 888-VVI-PLOT
www.vvi.com

Graph is an application with one design objective: "You give it data and it gives you a graph". The following diagram shows how it is done.



Probably the first thing you want to do is learn how to enter data. For that see the [Enter Data](#) section.

[Graph](#) > **Overview**

The following sections give an overview and also provide many links and references to other sections of this manual.

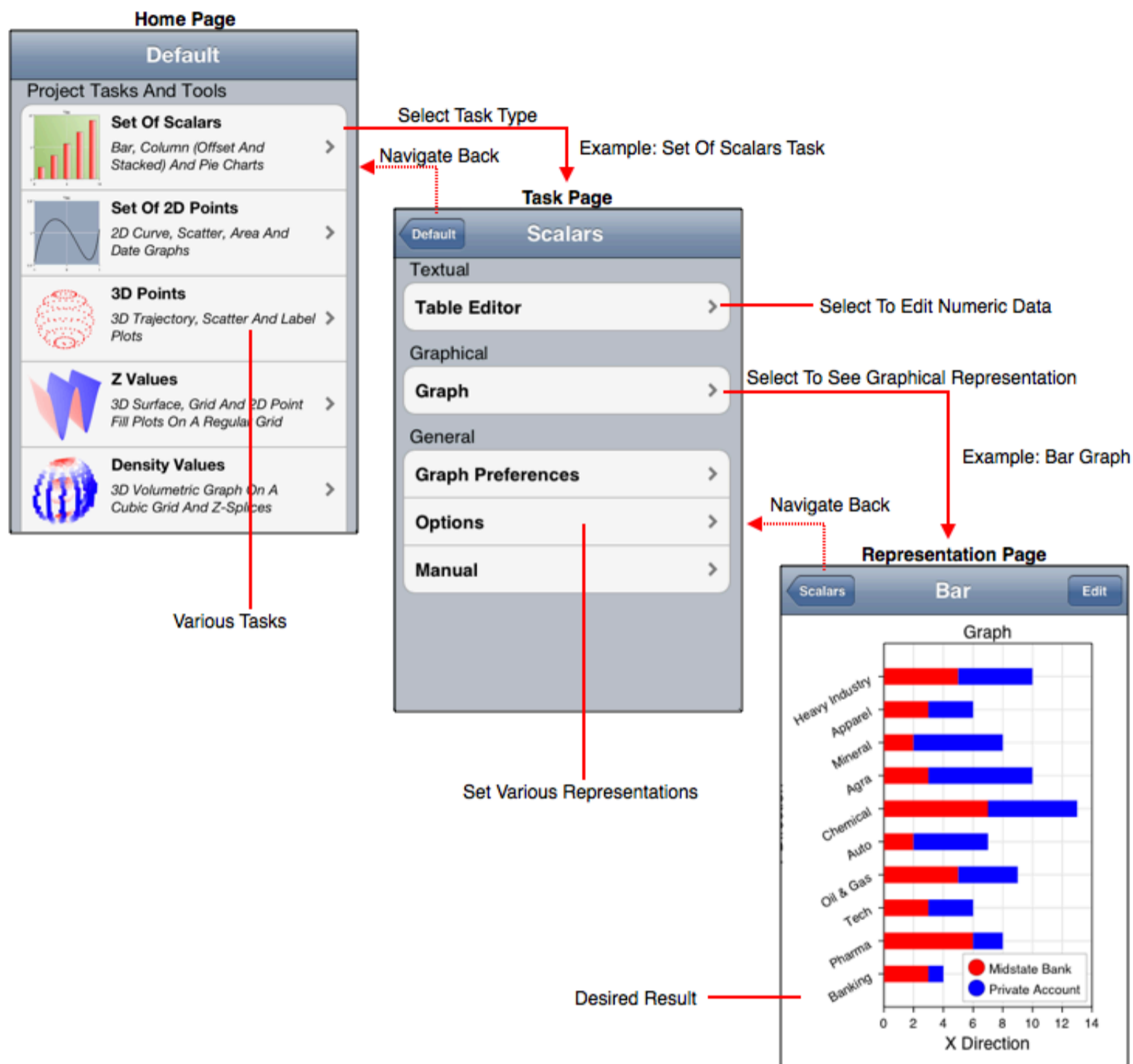
Overview	Description
Main Idea	Describes the main idea of this Graph application.
Highlights	Gives the highlights of this application.
Features	Runs down several features of this application.
App Preferences	Sets overall preferences for the application.
Popover	When you touch a data element a popover, or alert panel, comes forward. This sections describes that popover.
Representations	Names the representations, which are essentially the graphs, that are available.
Manual	Describes this manual.
Glossary	A list of terminology used in this manual.

© Copyright 1993-2012 by [Vimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

Graph > Overview > Main Idea

The figure below diagrams Graph's interface. It is a single window with access to various tasks and tools by navigating. The navigation pattern is Home > Task > Representation as shown in the figure below. Tools are described in the [Tools](#) section, graphs are described in the [Tasks](#) section and data input is described in various sections as referenced in the [Enter Data](#) section.

Probably the best starting point for learning how to use Graph is to actually use Graph. However, if you wish to read about using graph then perhaps a good starting point is the [Table Editor](#) section.



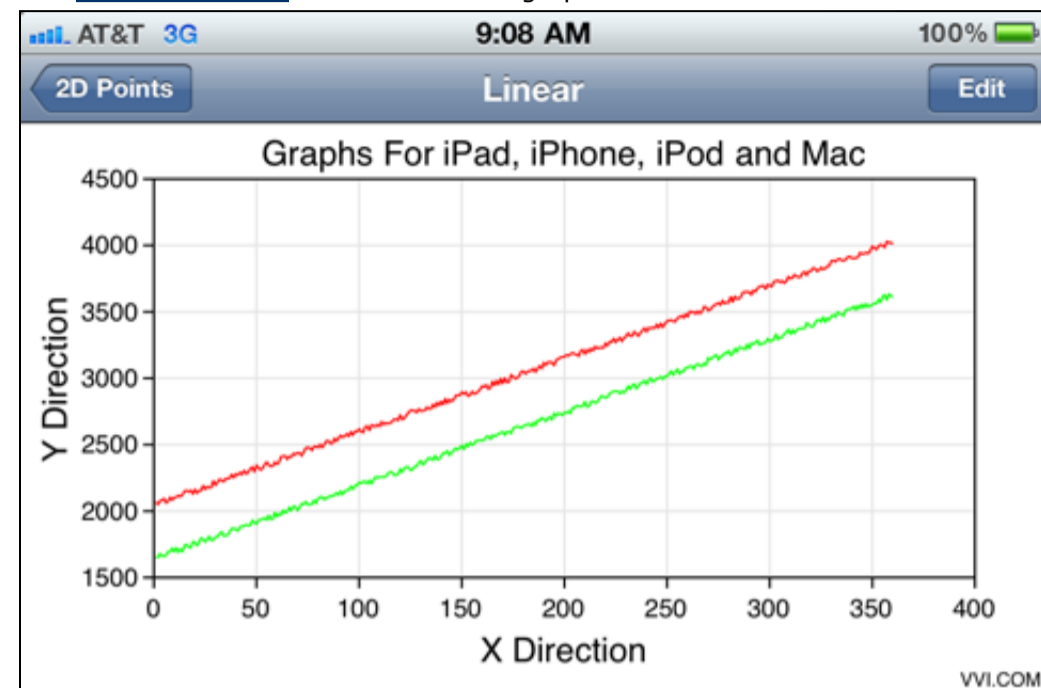
[Graph](#) > [Overview](#) > **Highlights****Highlights**

Mission Statement	The best and most powerful data visualization for iPhone, iPad and iPod touch.
Best Technologies	Developed from the ground up for the iPhone, iPad and iPod touch using Cocoa Touch and iOS standards.
Graph Types	Several 1D, 2D and 3D graph types including bar, column, pie, line, area, scatter, 3D perspective scatter, trajectory and surface, volumetric, z-sliced cell, user defined maps, linear, semi-log, x-log, log-log, polar, r-log, Gregorian date and many variations.
Maps	Maps are geographic, schematic, process maps, node graphs, floor plans or any diagram. A map of the United States is included and users can load an additional 30 maps they construct themselves using the optional Vwidget Builder application.
Skinning	Thousands of graphic, graph and data attributes, including customized artwork, autoscaling and many other attributes can be set by using Vwidget Builder, an optional Mac OS X desktop application, to make a skin document and applying that skin to this Graph application.
Data Sources	Paste from the pasteboard, fetch from web services, insert and edit using a table interface. Data is persistent so once you acquire data it is available the next time you need it.
Automated Sources	A few tasks include automated data generation. Least Squares computes linear regression, the Location task plots the device trajectory and the Accelerometer tasks plots the 3D vectors of the accelerometer sensor.
Touch	Touch graphs to scan data sets, rotate graphs, zoom, and scroll them. Touch a data point to see its value. Touch a curve segment to see the interpolated value, a bar or pie wedge to see its amplitude. Touch a map to see the description of the component of the map.
Organization	Data and preferences are organized by project and each project is persistent and automatically saved so you can retrieve and show your settings and data on subsequent use.
iTunes App Store	You can get the Graph application on the iTunes store by clicking this link: Graph .
Integration	The Graph application was written according to the Vwidget Code Reference Manual . With that API, a more exacting application can be developed that utilizes features shown in the Graph application.
Support	Should you have a question or comment then please contact VVI at 888-VVI-PLOT (888-884-7568) or info@vvi.com . To report a feature request please email support@vvi.com .

Graph > Overview > Features

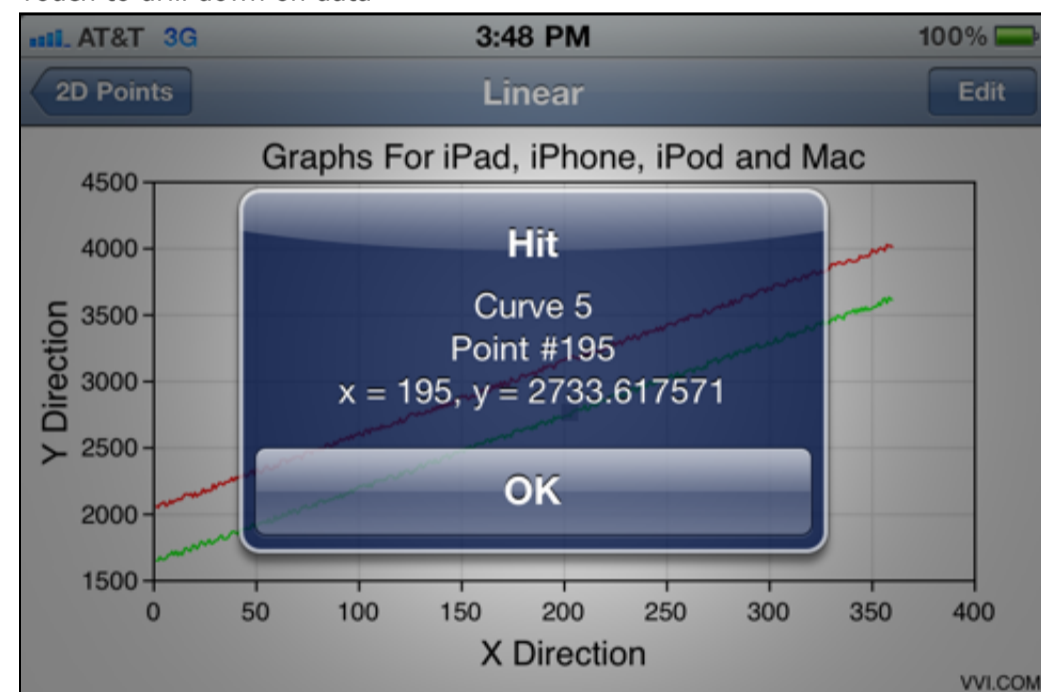
Features of Graph are described in the various sections. For example, the [Tasks](#) section describes each task and the [Tools](#) section describes each tool. However, since those sections are limited by their focus they lose the big picture even in their amalgamation. This section tries to remedy that by providing a sequence of features as a narrative which may be more consistent with the way a reader thinks.

The [Set Of 2D Points](#) task makes line graphs.



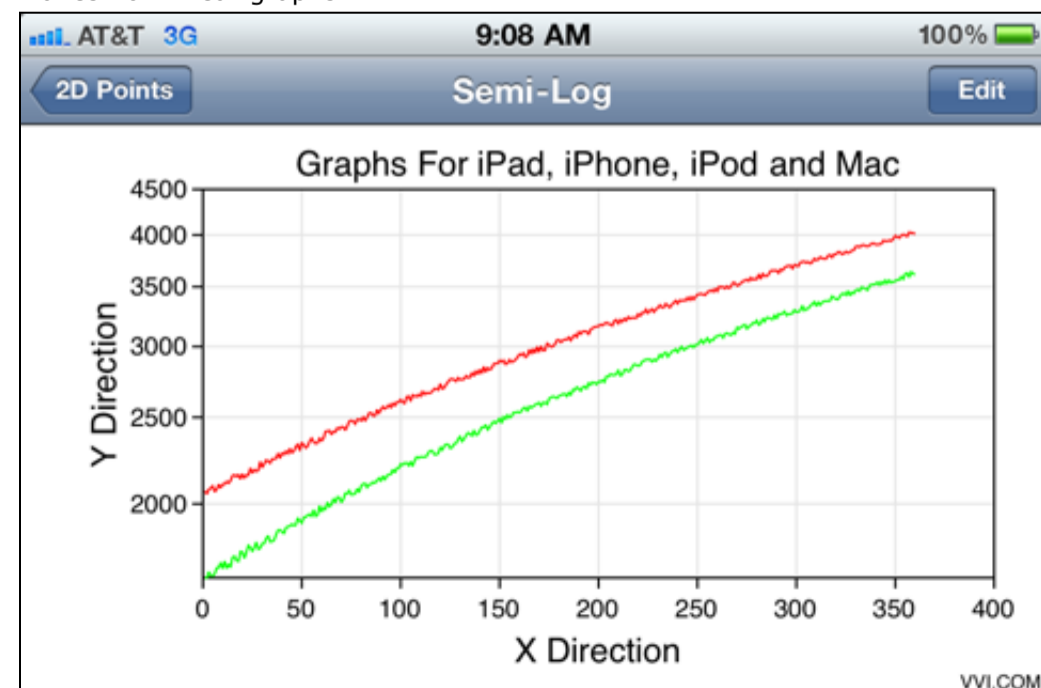
- Line graphs are a staple item. This picture shows two curves.
- Supports up to 20 curves on a single graph.
- Curve colors are defined by a color table skin, the hundreds of other graphical options are defined by the primary skin.
- Axes are set to autoscale, however they can also be set to fixed values or the autoscale parameters can be adjusted by altering the skin for this graph.

Touch to drill down on data



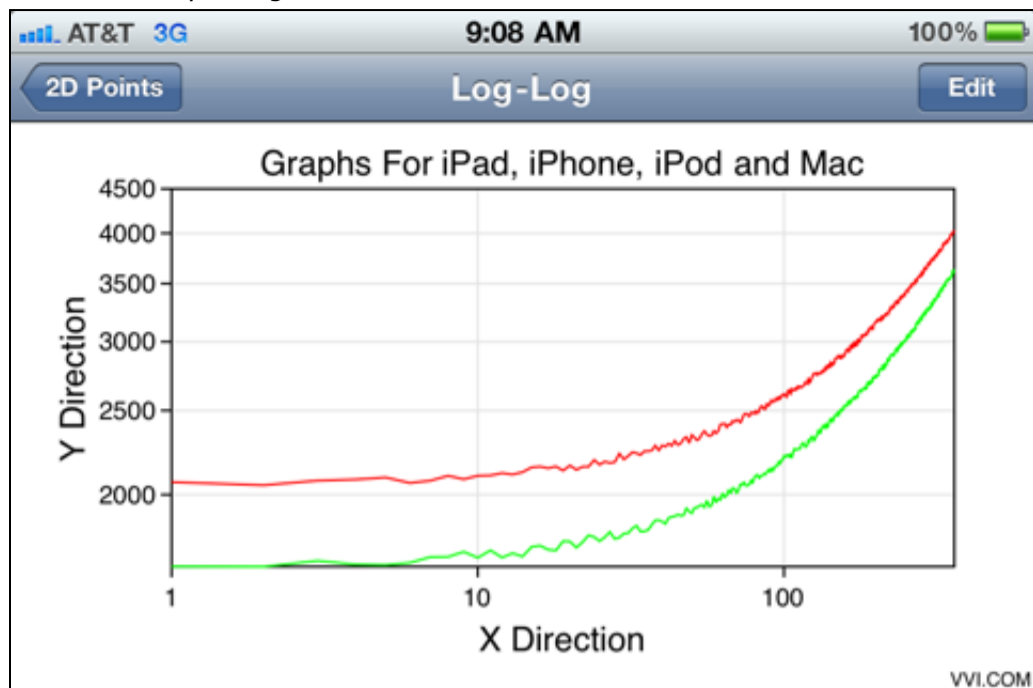
- Touch a curve to show its data.
- Touching a data point shows the values of that data while touching the curve (interpolated line segment) shows the interpolated value at the touch point.
- All the pictures shown here are in landscape format, but if you rotate the device then it shows portrait format just as you expect.

Makes non-linear graphs



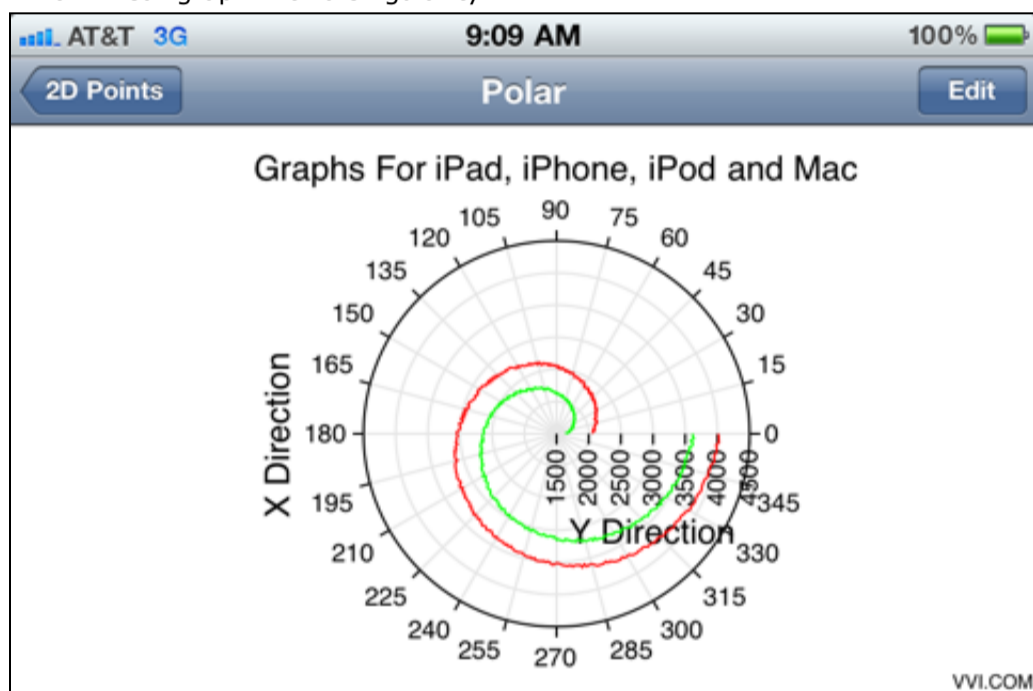
- Touch the Semi-Log entry to get a Semi-Log representation of the data.
- Log scales are automatic and there are many adjustments that can be made to the autoscale by using a skin or you can turn autoscale off.

Sub and full cycle log scales



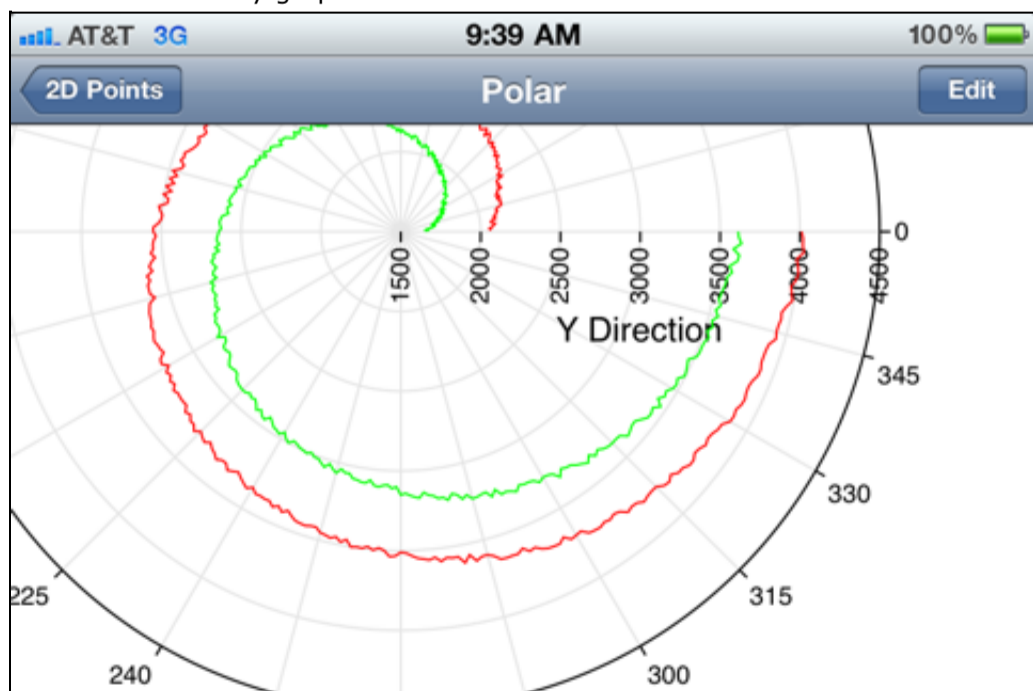
- Touch on the Log-Log 2D Points Task entry to get a Log-Log graph.
- No need to reenter the data, rather simply select the representation of the data you want.
- Touch works on non-linear graphs as well.
- Notice the sub-cycle log scale on the y-axis and the full-cycle scale on the x-axis as computed by the autoscaler.

A non-linear graph with a singularity



- Polar graphs can be shown too.
- Touch on the Linear, Semi-Log, X-Log, Log-Log, Polar, R-Log Polar entries for line, area or scatter plots to show the same data in different coordinates.
- The graph's titles can be set in the Graph Preference task entry or on the skin.
- Font size, type and other attributes can be set in the skin.

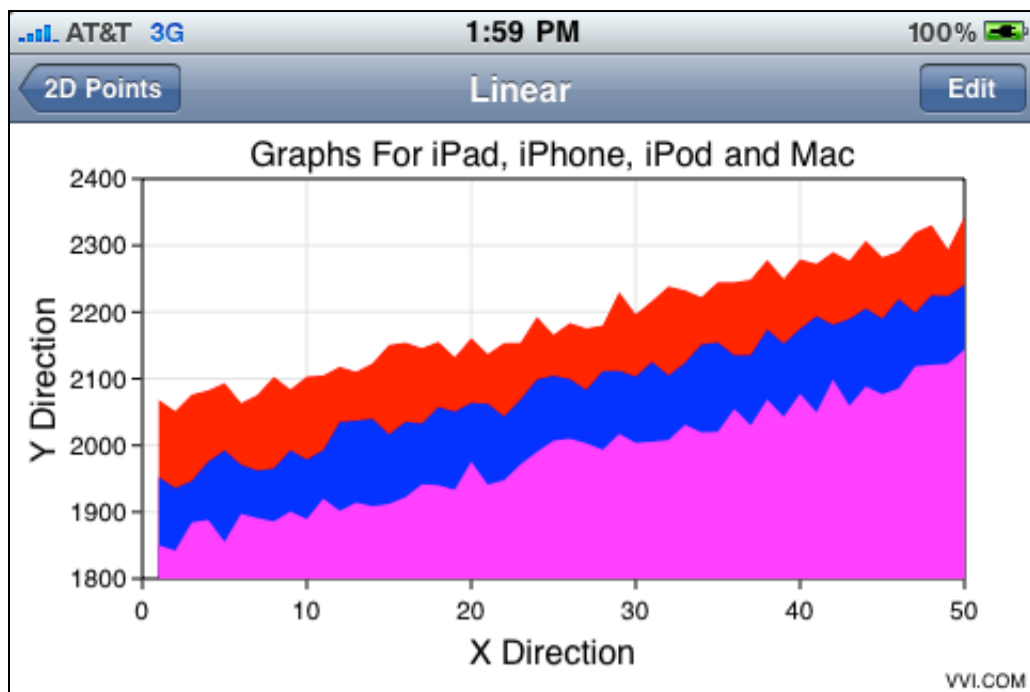
Can zoom in on any graph



- Any graph can be zoomed and scrolled. This picture shows the polar graph above zoomed into a section of the graph.
- Touch any part of the curve to see the curve's x and y values, or in the case of a polar plot, the curve's theta and radius values.
- The radius-axis labels are rotated 90 degrees. With a skin, you can set the rotation to any value simply by dragging a dial.

Makes area graphs

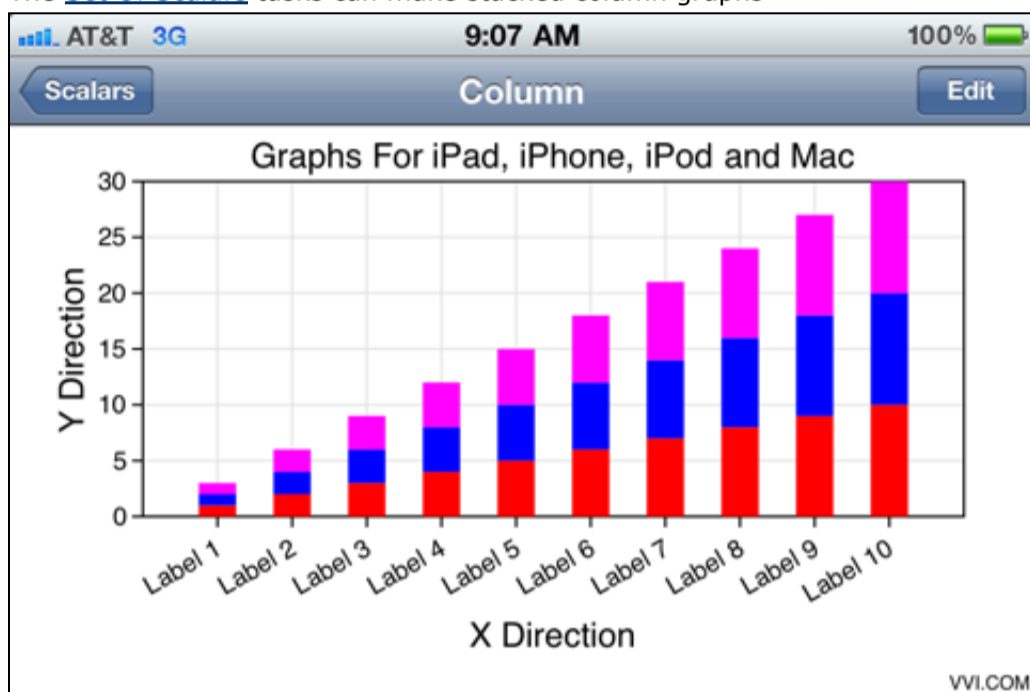
- Area graphs can be made too.
- Many of the graphs share the same data and simply present that data on a different coordinate system or with



different graphical effects. In this case and area graph is simply a different graphical representation of a line graph.

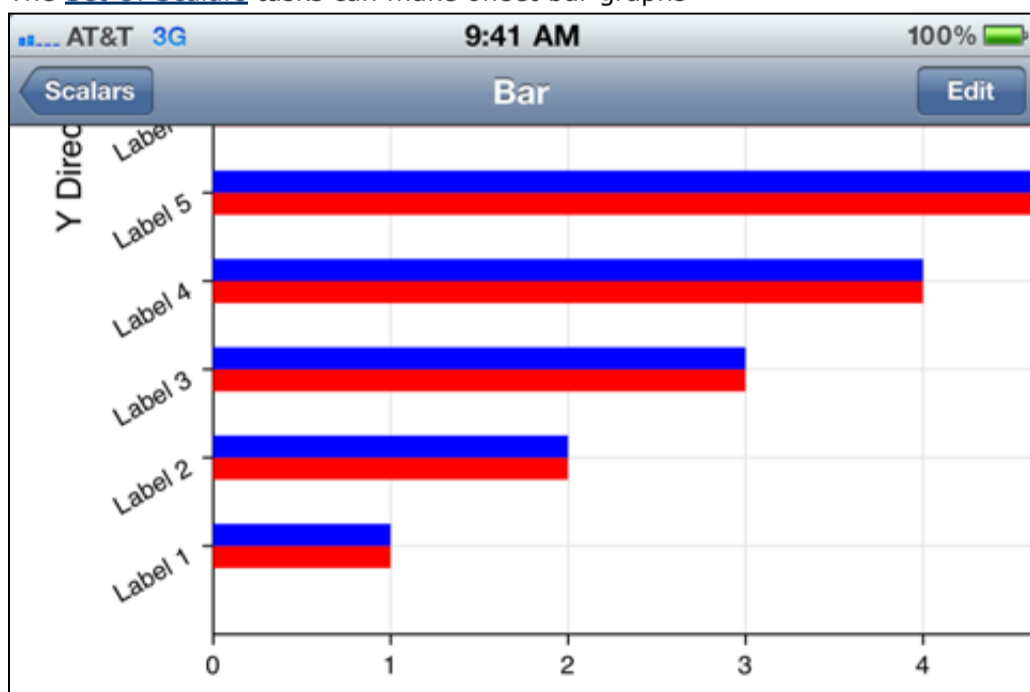
- The stacked effect gives a sense of accumulated data.
- If needed, the autoscale feature can be set so that data presents to the graph frame and there is no gap. However, usually ticks are defined on some regular interval such as shown here so that the data does not fill the entire graph.

The [Set Of Scalars](#) tasks can make stacked column graphs



- This picture shows a stacked column bar of a set of scalar data.
- Scalar data can be represented by column, bar and pie graphs.
- Grids are on by default, however you may want to turn them off with a skin.
- Gradients can be set via a skin as well. Gradients are particularly handsome for bar and column graphs.

The [Set Of Scalars](#) tasks can make offset bar graphs



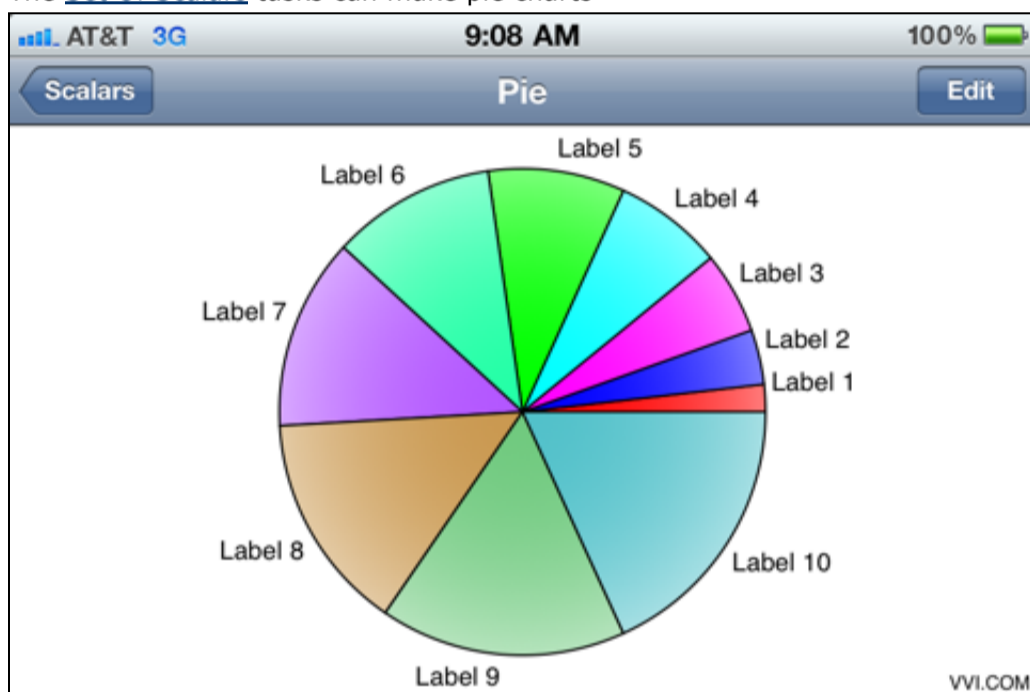
- Touching the offset bar task entry shows data in an offset bar format.
- The y-axis labels are defined by user input.
- Double touch to zoom in on the graph. The picture shows the graph zoomed into the lower left corner.

Touch a bar to get more information

- Touch a bar to see information about its value as well as grouping indices.
- Almost all graphs respond to touches.

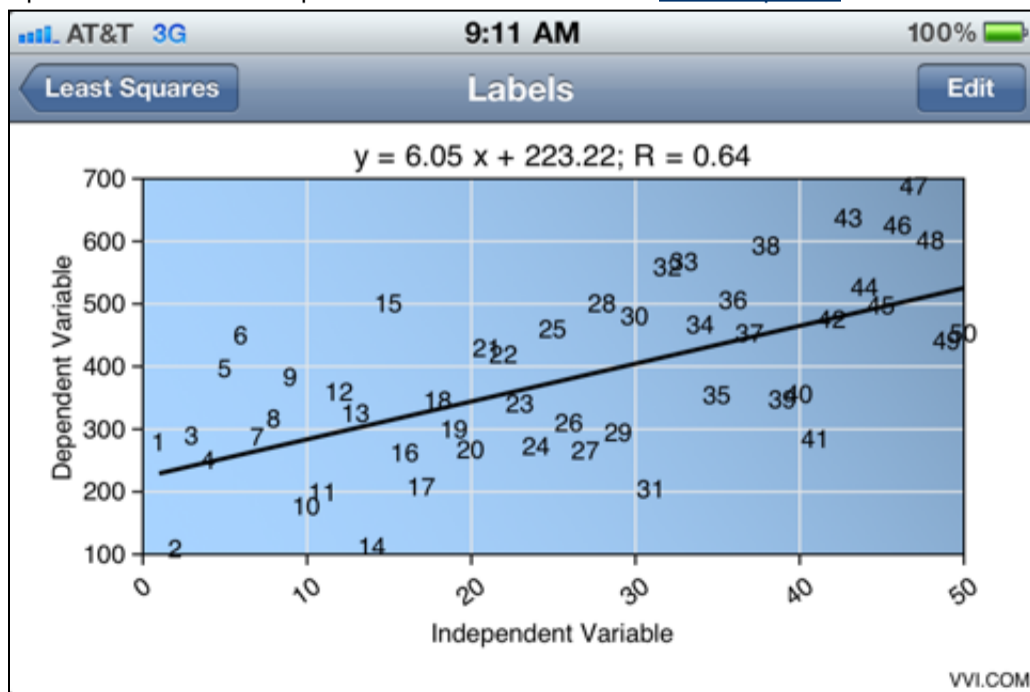


The [Set Of Scalars](#) tasks can make pie charts



- The default pie chart with labels has a gradient effect.
- Most other graphics can also have gradients by loading in the corresponding skin set to have gradients.
- The wedge border and other attributes are also defined by skins that you can make.

Specialties like least squares is done with the [Least Squares](#) task



- Point data can be entered into the Least Squares task and a linear regression will be computed automatically.
- Touch the line fit to see interpolated values, or a data point to see the values at that point.
- Choose different task entries to see dots or labels at the data point.
- The background of the graph has a blue gradient effect and you can change that yourself using the appropriate skin.

A map of the USA is one of many maps you can have, see the [Maps](#) task

- Maps can be loaded into the Map task, in this case a map of the United States of America.
- Maps are geographic, schematic, process maps, node graphs, floor plans or any diagram.
- Make your own map and load it into the Map task by



following the directions.

- When you touch on a map it shows the name that you defined for that part of the map, such as "Valve 1". If you choose to develop your own application then you can set the touch as a feedback command. For example: Touching Valve 1 opens and closes it.

Zooms in on a map



- Like any graph, maps can be zoomed in on and scrolled.
- The map of the United States is included with Vwidget Builder. You can use that as a starting point for your own map skin. For example, you can define state sets as sales territories and touch to see sales data.

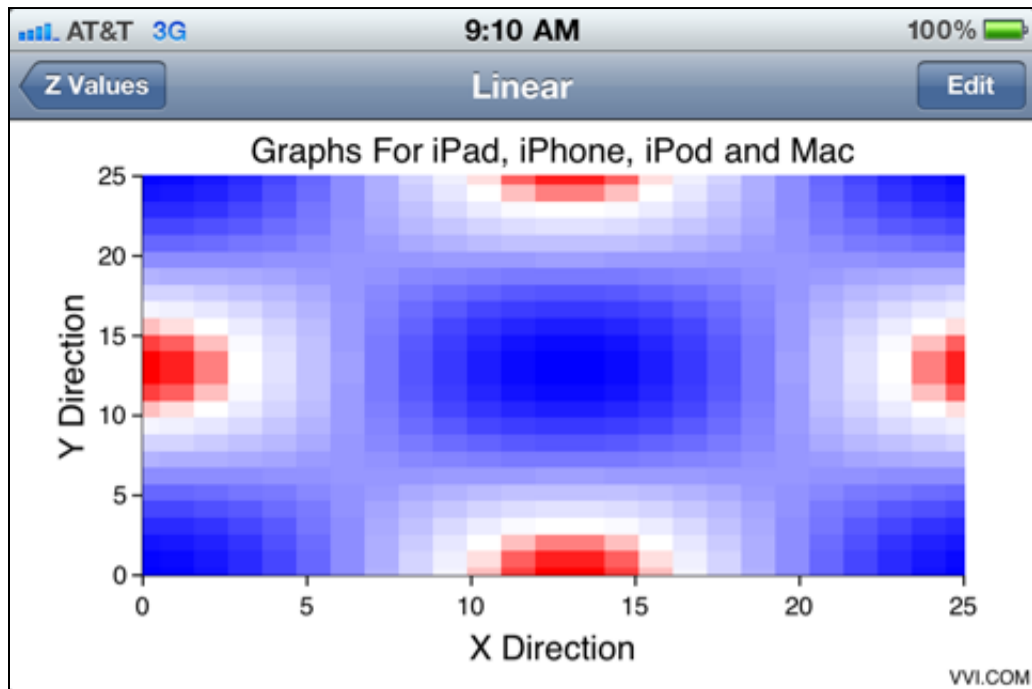
Drill down on a map component



- Touching a state on the map shows its name.
- With a little programming the touch can retrieve enterprise sales data based on geography and present it to the user. For example: Who is the sales rep. in Pennsylvania and what is his phone number?
- All data can be stored on the device so users have access to it at all times and without delay.

Point fill plot on a rectilinear grid

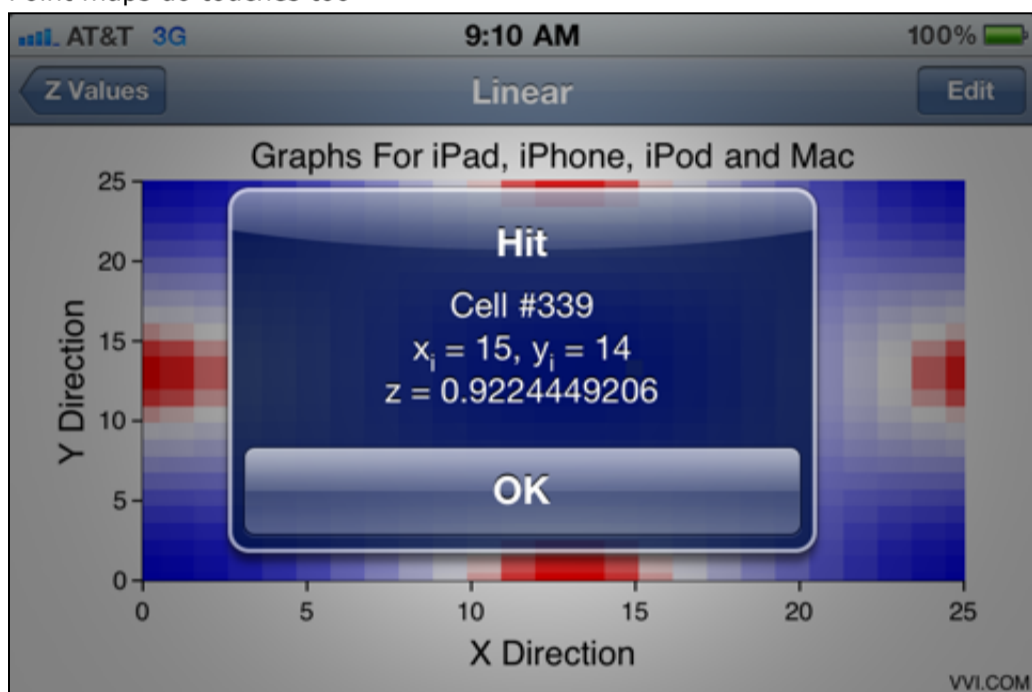
- Use the Z Values task to show a point fill of scalars on a regular grid.
- Although associated with Z-Values, a point fill plot can show any 2D density representation.
- Entries are available for all the graph coordinates, i.e.:



Linear, Semi-Log, X-Log, Log-Log, Polar and R-Log Polar.

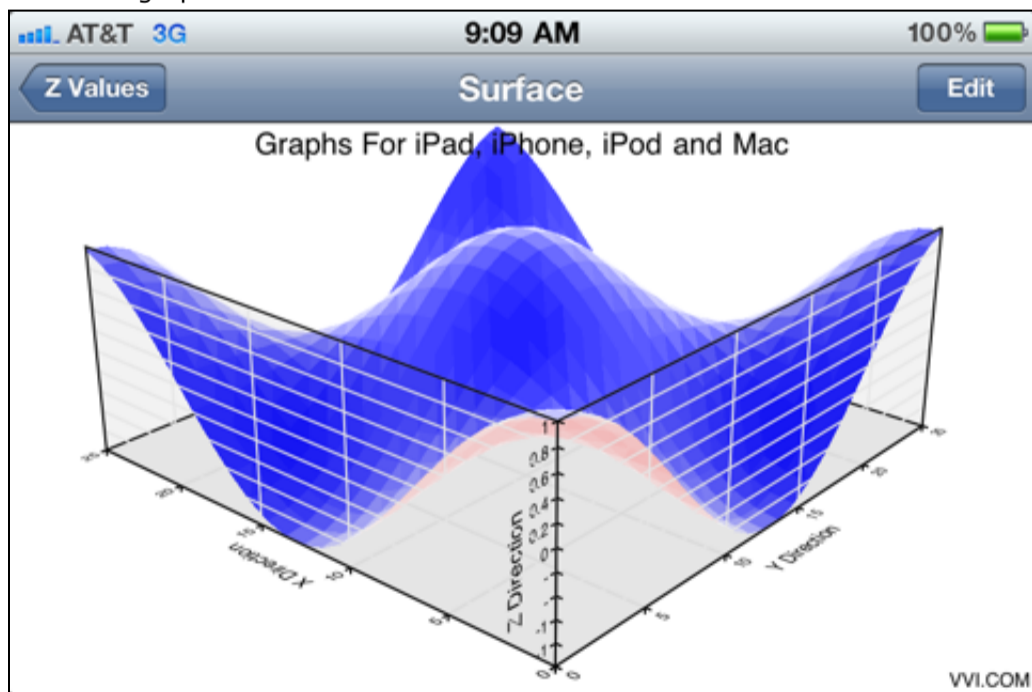
- Simply touch different Task Entries to see the point fill plot on different coordinate representations.

Point maps do touches too



- Touch the point fill graph to show the grid indices and amplitude of the data at the touch point.
- If you develop using the Vwidget library then touches can be intercepted to perform nearly any task upon a touch. See the Event Qualifier sections in the [Vwidget Code Reference Manual](#).

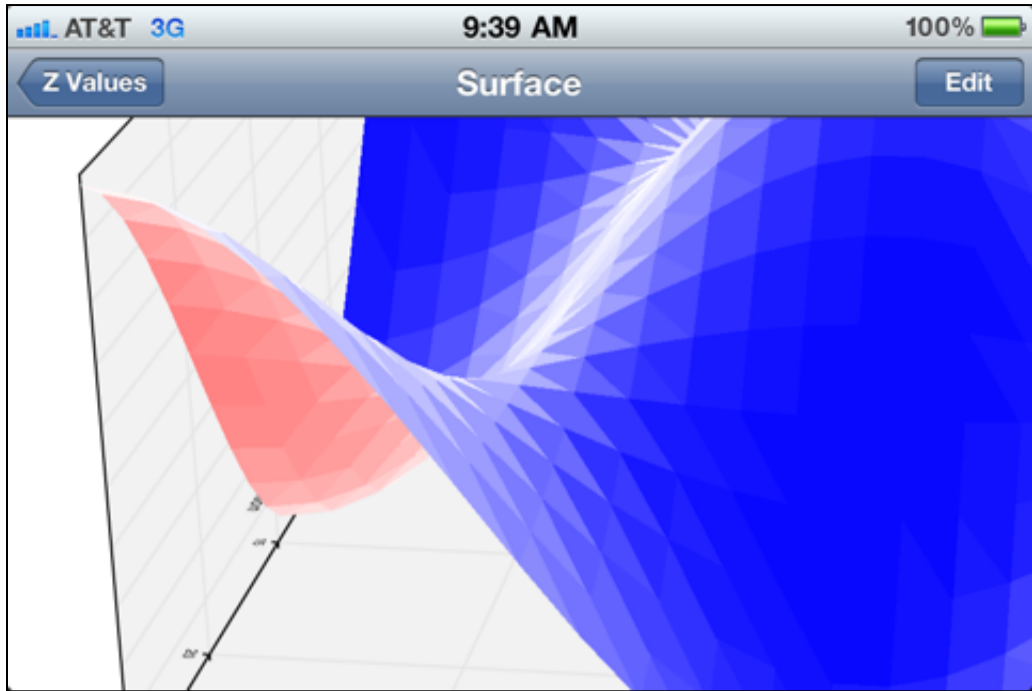
A surface graph



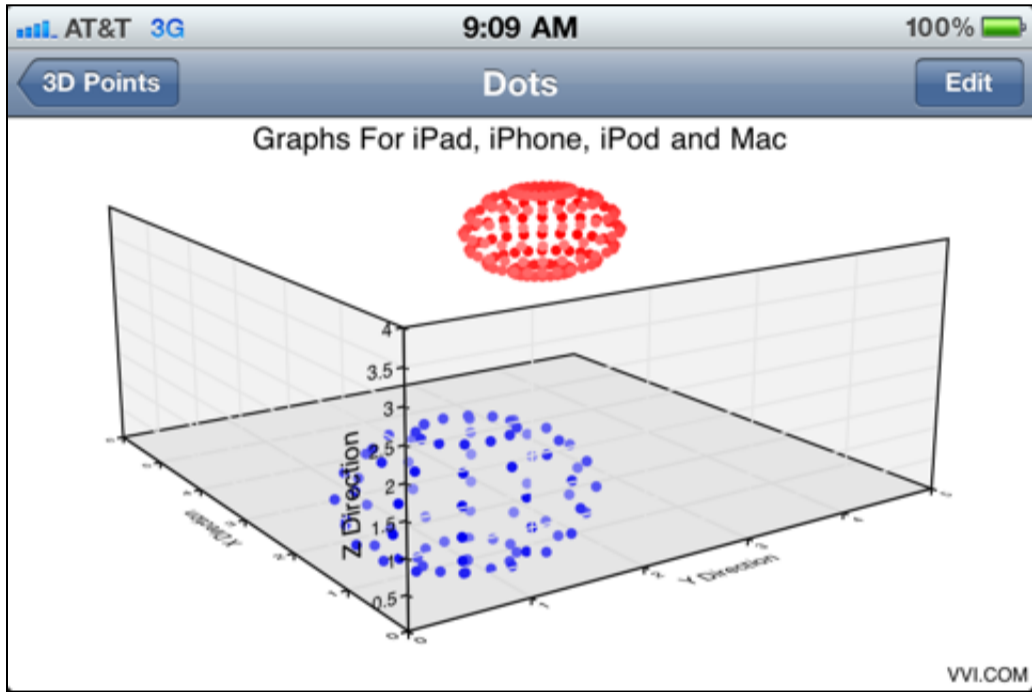
- This picture shows a 3D perspective surface chart.
- The surface chart is defined by scalars on a regular grid, same as for the point fill plot. Both are different representations of the same data and are hence associated with the same Task.
- Color mappings are defined in the skin.

Zooms in on 3D perspective graphs

- Double touching the surface chart zooms into it just as a double touch does for any graph or map.

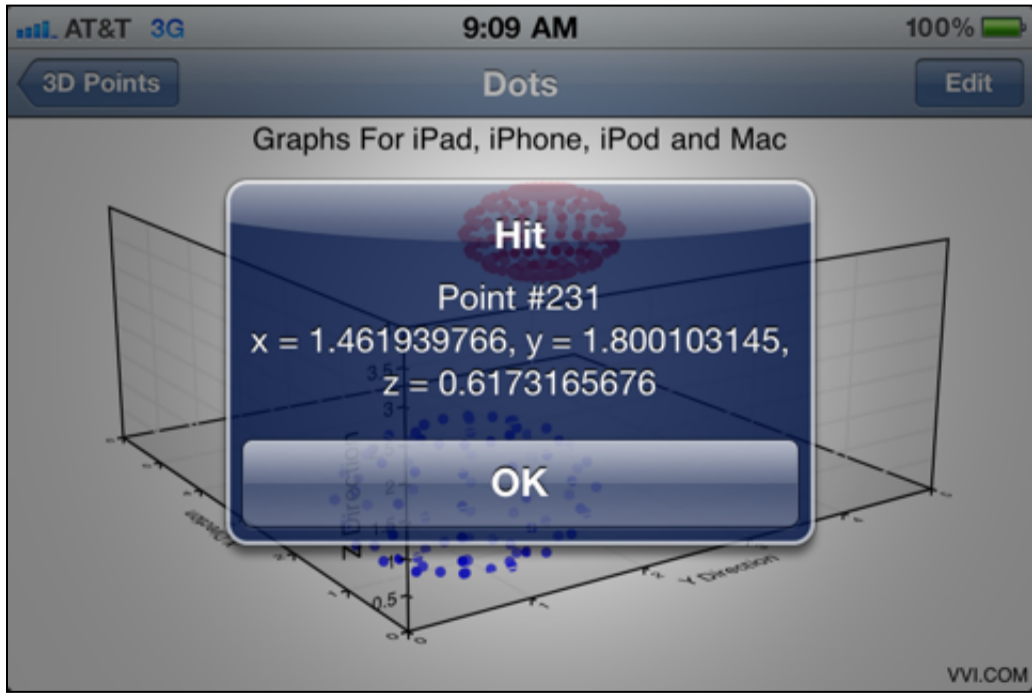


3D scatter plot



- This picture shows a 3D perspective scatter plot.
- Touch drag rotates the graph.

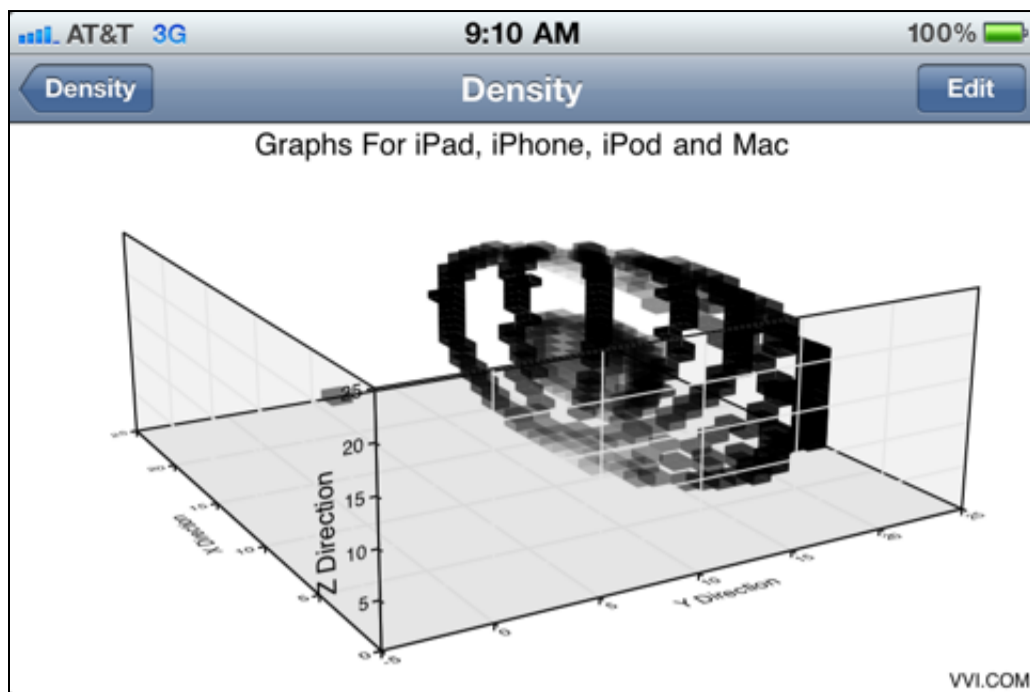
3D does touches too



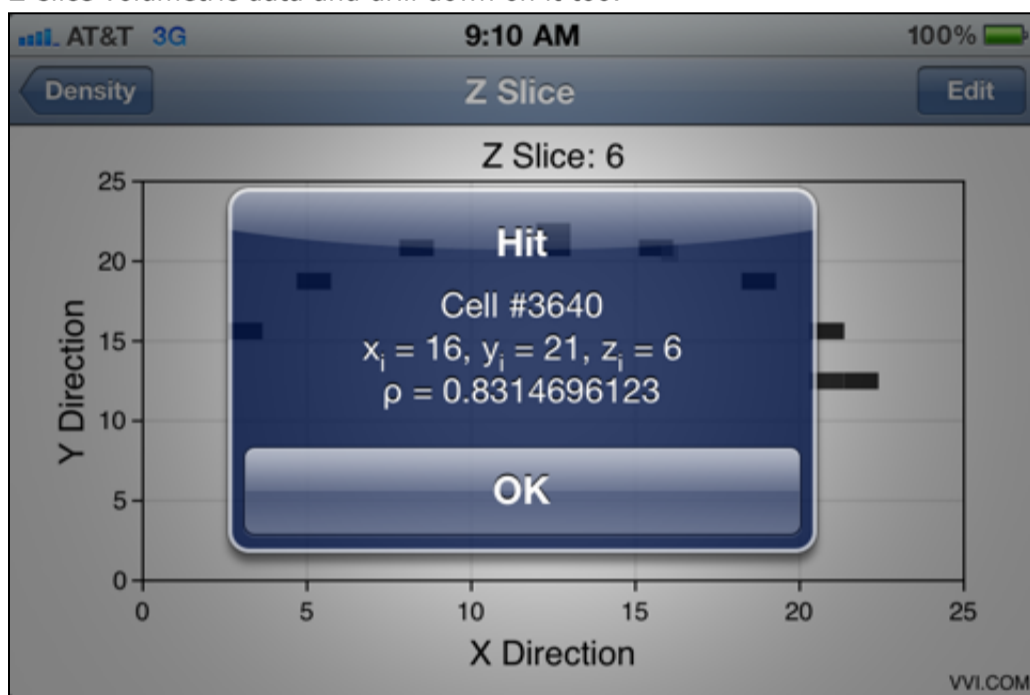
- Touching a 3D point on the graph gives its 3D point values.
- A combination of rotate and zoom helps to bring forward and separate the data point to touch.
- A combination of touch, graphical effects and programming can permit intricate exploration and querying of 3D data.

Volumetric density plot

- This picture shows a volume graph.
- A volume graph is a representation of density values on a 3D regular grid shown by color coded cubes.
- The color v.s. density mapping is defined through a skin.



Z-slice volumetric data and drill down on it too.



- This picture shows a z-slice of volume data.
- Touching a density values shows cell values and density values.
- The volume density values can be panned through z-slices by touch-drag horizontally or by animation.

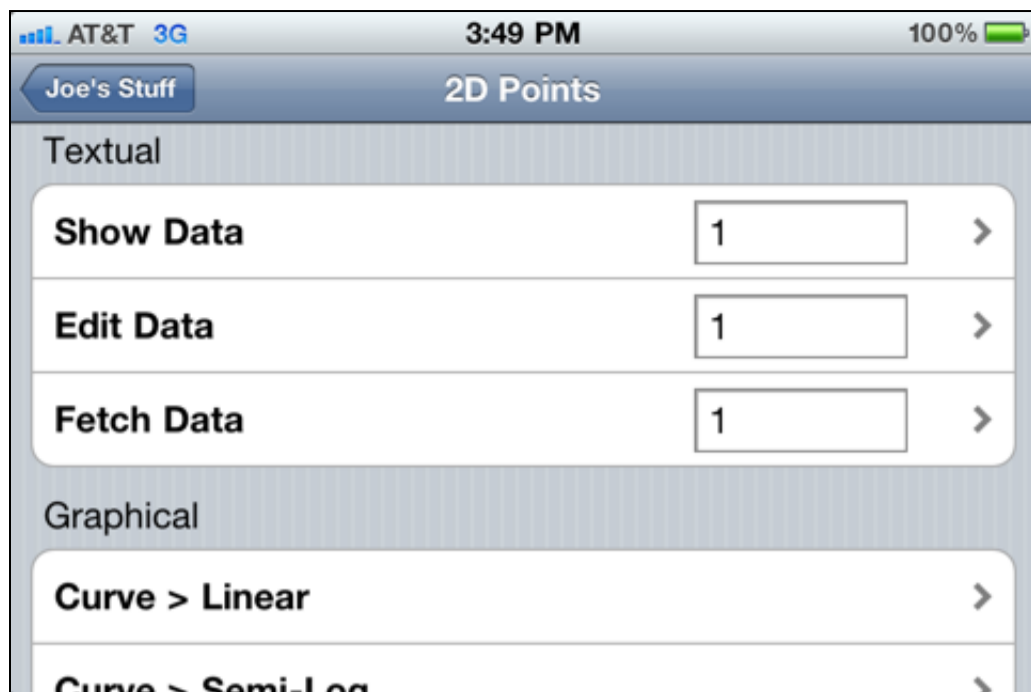
Organize by projects



- Data and preferences are organized by projects.
- You can make as many projects as you wish.
- Projects help organize tasks along project boundaries so you can work on one project without affecting another.

Insert data using textual methods

- This picture shows the task textual entries for the 2D Points task (line graph a.k.a. curves).
- There are 3 ways to acquire data, using a tabular entry (Show Data), using a normal text view to copy, paste and key in data with the keypad (Edit Data) and fetching data from a URL which can reference a web server or shared file



entry (Fetch Data).

- Touching a task textual entry navigates to that entry's input mechanism.

Fetch data from anywhere on the web



- This picture shows the Fetch Data task entry.

- Enter a URL and then touch the Fetch Data button.

• Data is retrieved from the URL. The URL can point to something as simple as a flat file, or can point to a web server CGI service which can, for example, access a database.

• Once the data is fetched then it resides on the device and does not have to be retrieved again. However, you can also setup fetch during first viewing or animation to reacquire data without clicking the Fetch Data button.

One more picture ...

- One more picture. In this case, showing the Xcode project for the Graph application.

• The Graph application does not require any programming and can be used right away. Skins are made with Vwidget Builder, an optional Mac OS X application, and although it is a complex application it also requires no programming to access all its features.

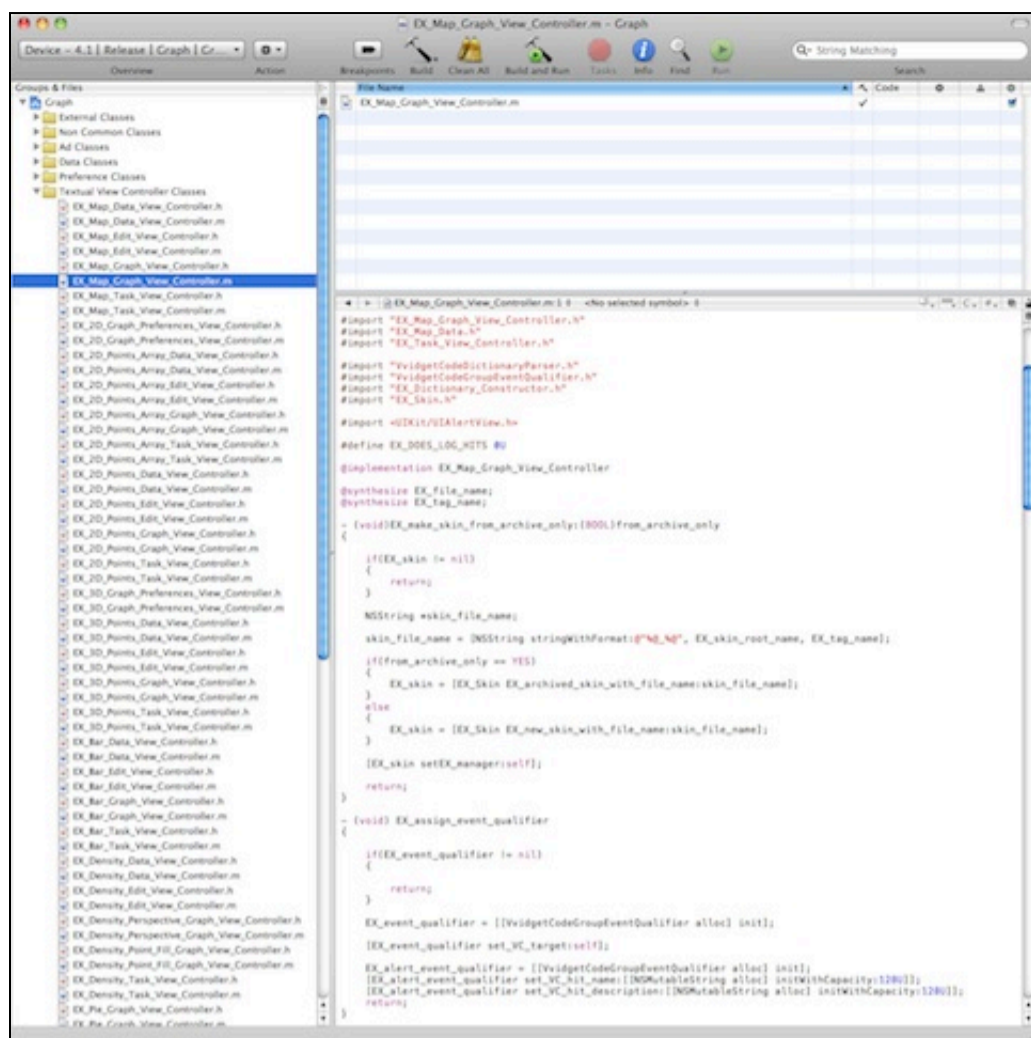
• The Vwidget library, documented at [Vwidget Code Reference Manual](#) is based on about a million of code, or so, but that isn't really of any concern since it comes as a prepackaged library that can be inserted into applications, such as this Graph application, without any programming at all.

• This Graph application is about 30,000 lines of code, a million characters and 100 classes in scope. That is a lot of typing by itself but is still small because it leverages against the much larger project of the libraries.

• If you do choose to program a data visualization application then your use of Vwidget may be as little as a handful of code, perhaps a few dozen lines.

• If you do not want to program at all then this Graph application is for you. No programming, just results based on data inserted.

• If you seek greater features then there are options available from VVI, the leading expert in the field of data visualization on iPhone, iPad and iPod touch as well as Mac

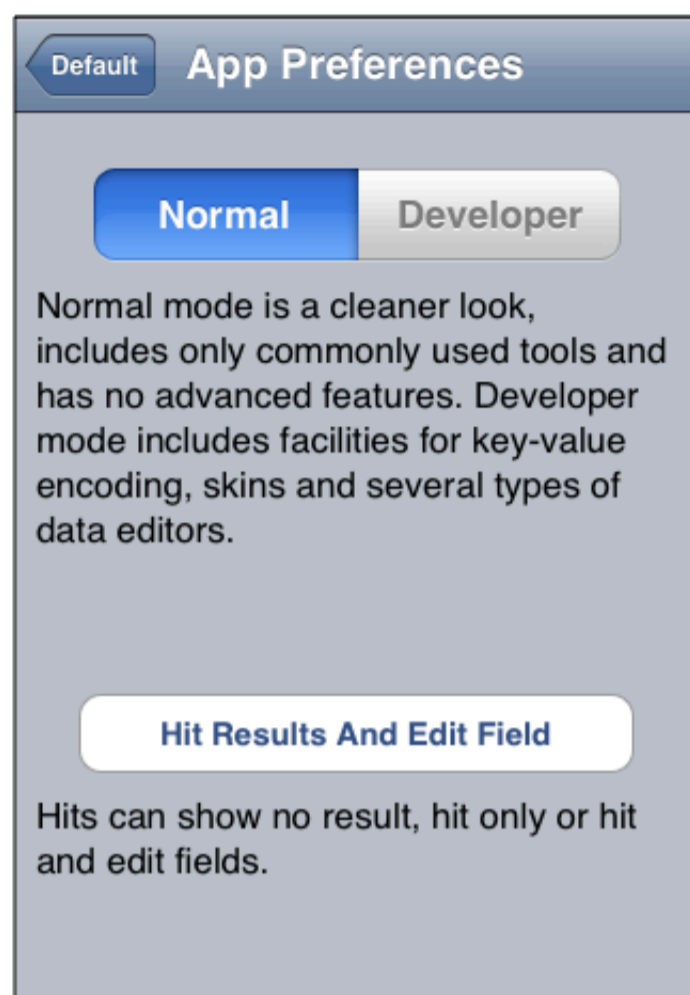


Most people desire results without programming. The Graph application is based on one principle: "You give it data and it gives you a graph". That is straightforward and the way it should be. It also employs skins which adds more features but with an increased level of indirection and complexity. If you don't mind the default visuals of the Graph application then you need not be concerned with skins.

There are also people and companies that prefer more automation and specific features for their own operations. But, they also do not desire a huge investment in data visualization programming. For that situation the Vwidget library may meet those needs. For additional information email info@vvi.com.

Graph > Overview > App Preferences

The App Preference tool helps set a few app-wide preferences. It is diagrammed in the following figure.



Sets overall app preferences

Settings are defined as follows:

Mode

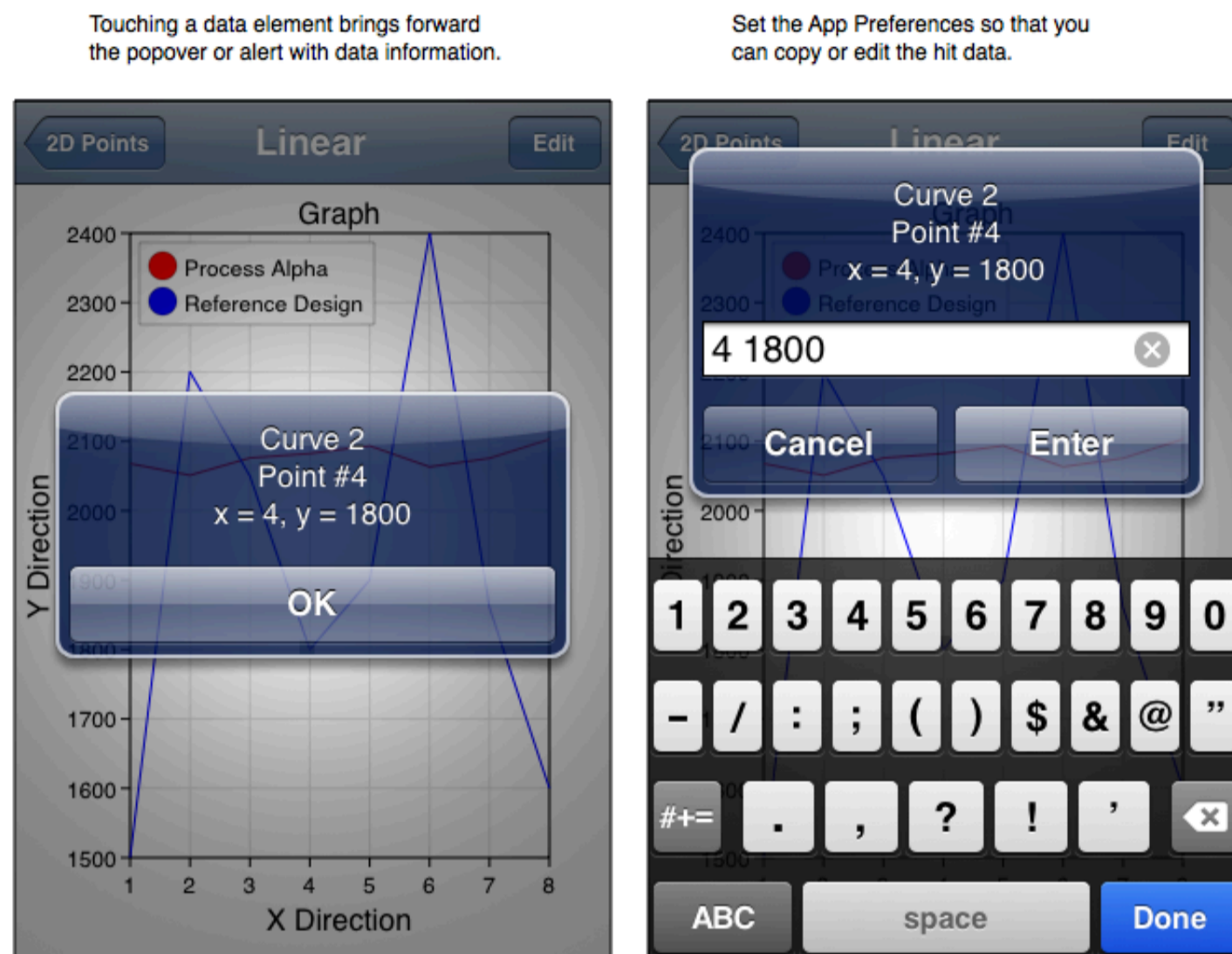
- Normal mode is a cleaner look, includes only commonly used tools and has no advanced features. Graphical representations are attribute based and set using the [Options](#) tool for each task.
- Developer mode includes the [Edit Data](#), [Fetch Data](#), [Show Data](#), [Info](#) and [Reset](#) tools and graphical representations are navigation based. Since graphical representations are navigation based the [Skins](#) apply to each representation separately.

Hit type

- Hit Results And Edit Field gives an edit field when a data point is hit. Thus the hit function also becomes a way to edit or copy data.
- Hit Results Only show the hit parameters and does not give a way to edit data.
- Do Not Query Hits disables hit detection and processing.

Graph > Overview > Popover (or Alert)

The popover (on the iPad) and alert panel (on the iPhone and iPod touch) is used to show the query of a task's graphical representation. For example, the figure below shows what happens when you touch a curve.



The popover shows the curve number (there can be up to 20 curves), the point sequence number and the point x and y values. Each task shows similar information in the popover related to that task's data. In the case of a map, the data is the map component's description as set in the Vwidget Builder map template (see [Making A Map](#)).

[Graph](#) > [Overview](#) > [Representations](#)

Each [Tasks](#) can be efficiently visualized in different representations, either textual or graphical. The graphical representation is usually a graph. This manual has task sections but not representation sections so finding the graph to make can be difficult. The following table solves that problem by associating representation with task.

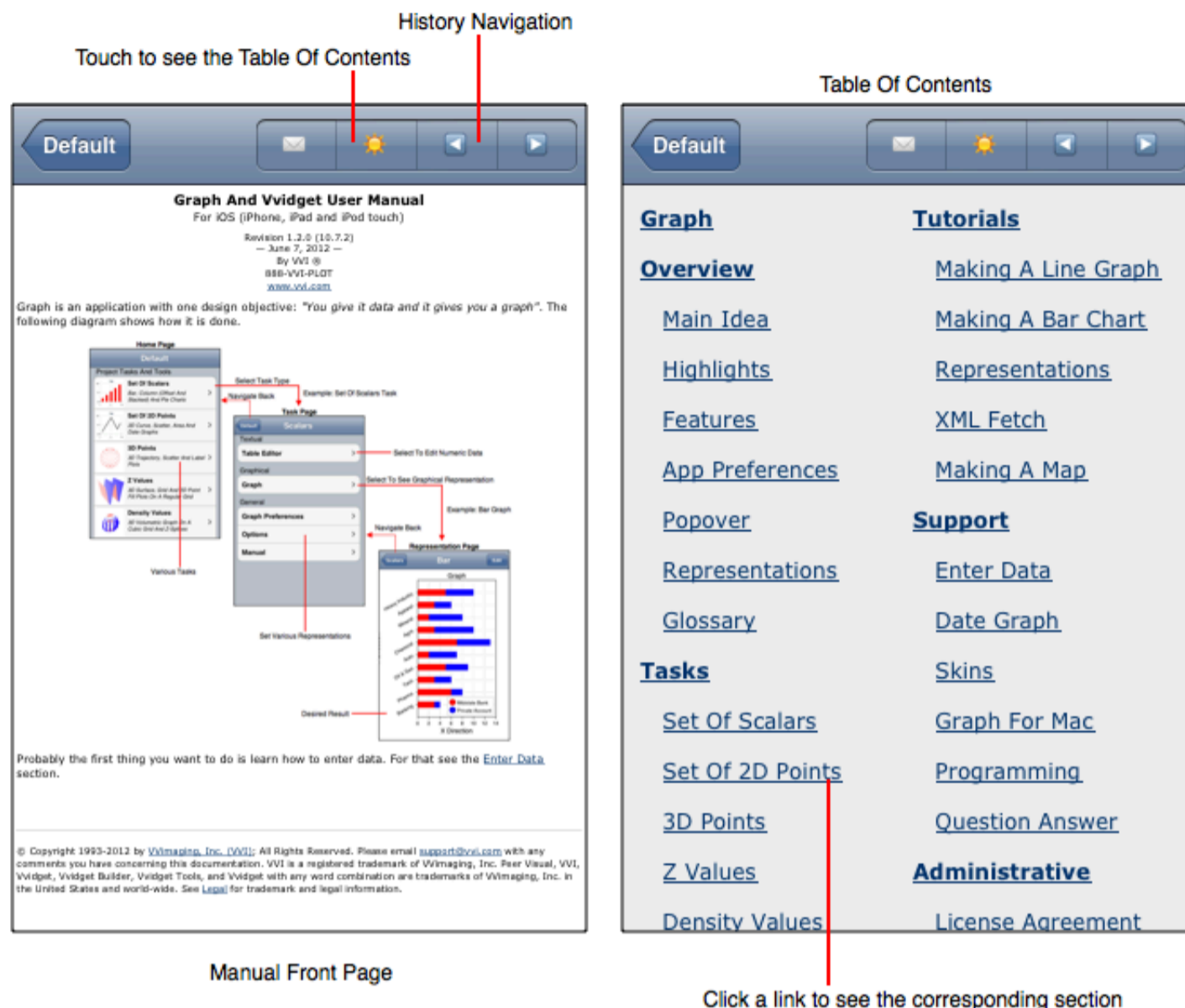
Representation	See Task
Bar chart, column chart, stacked and offset bar or column chart, pie chart	Set Of Scalars
Line, area, scatter chart in any of rectilinear, x-log, semilog, log-log, polar, r-log polar coordinates. Note that with a skin you can also transform a scatter plot into a trajectory plot.	Set 2D Points
3D scatter plot, 3D trajectory plot	3D Points
Point fill plot (aka: image plot), surface plot, 3D surface grid plot.	Z Values
Volumetric graph, z-slice image plot.	Density Values

Note that each task above is associated with several representations so that once a task is selected, operating on it by transformations to different representation is efficient. That is the essence of the task-oriented architecture.

© Copyright 1993-2012 by [Vimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

Graph > Overview > Manual

The manual tool shows the Graph embedded manual and is diagrammed below.



The manual is accessed from the home page or any of the tasks. If the manual is accessed from a task then that task's section is first shown by the manual tool.

The manual's figures are shown on the iPhone and for Normal Mode (set using the [App Preferences](#)) except when applicable (as when a feature is only available in Developer Mode or for the iPad).

[Graph](#) > [Overview](#) > [Glossary](#)

Below is a glossary of words used in this manual.

Additional definitions, and descriptions, are available in the online [Vwidget Builder](#) manual. Vwidget Builder is a more powerful tool that constructs tasks representations through a more comprehensive user interface.

Terminology	Definition
Column	A Column refers to a column of a table (in the vertical direction) or a bar of a bar chart oriented in the y-direction. This dual meaning leads to a bit of confusing use of the word column in the Set Of Scalars task.
Page	A Page (aka: View, Screen) is what is shown as you use the standard navigation buttons. Generally a Page is thought of as a screen of information. Graph's pages traverse from Home > Task > Representation pages, and back, as shown in the Main Idea section.
Representation	All Tasks are comprised of rows which when touched navigate to a representation. A representation is either numeric (a bunch of numbers) or graphical (a graph). The number representation is used to show, retrieve and edit numeric data and the graphical representations are used to show that data. The Main Idea section diagrams the navigation to a representation as a final result of the navigation.
Scalar	A scalar is a single number such as 3.1415. A scalar can also be called a number which is more common but in this manual the word scalar is preferred. (Note: Technically, the use of the word "number" leads to ambiguity so scalar is desired.)
Tasks	Tasks are specific class of problems that are associated with a class of data. For example: A hypothetical line graph task operates on 2D points and that is why a line graph task is actually called "Set Of 2D Points Task" because the data is what binds the different representations of that task together (line, scatter, area, etc.).
Tool	Tools operate upon tasks and are represented by their own Page.
2D Point	A 2D point is a pair of scalars, that is an ordered pair. The x-dimension is first and the y-dimension is second. The Set Of 2D Points task operates on 2D points.
3D Point	A 3D point is a triplet of scalars, that is an ordered triplet. The x-dimension is first, y-dimension second and z-dimension is third. The 3D Points task operates on 3D points.

© Copyright 1993-2012 by [Vimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

Graph > Tasks

When you operate Graph you first select an appropriate task using the [Home](#) tool. Once selected then you operate upon a task using all the features described in this manual including all the other [Tools](#). The following sections describe Tasks features.

Tasks	Description
Set Of Scalars	Describes the Set Of Scalars task, basically bar, column and pie charts.
Set Of 2D Points	Describes the Set Of 2D Points task, basically line graphs.
3D Points	Describes the 3D Points task. 3D points are represented on a 3D perspective graph in dot, label and line (trajectory) form.
Z Values	Describes the Z Values task. Z Values are a 2D grid of scalars interpreted as height in another orthogonal axis (the z axis). Such a representation can also be mapped onto a 2D point fill representation.
Density Values	Describes the Density task. Densities are interpreted as values between 0 and 1 that represent the density of a 3D object on a regular grid.
Maps	Shows how to work with a Map task.
Accelerometer Vectors	Describes the Accelerometer Vectors task.
Location Tracking	Describes the Location Tracking task.
LeastSquares	Describes the Least Squares task (linear regression).
Weight	Describes the Weight task.
Health	Describes the Health task.

© Copyright 1993-2012 by [Vimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

Graph > Tasks > Set Of Scalars

The Set Of Scalars task shows, among other things, bar graphs. Here are a few things to note about this task:

- Data is defined like this: $\{ \{s_{1,1}, s_{2,1}, \dots, s_{n1,1}\}, \{s_{1,2}, s_{2,2}, \dots, s_{n2,2}\} \dots \{s_{1,m}, s_{2,m}, \dots, s_{nm,m}\} \}$. Translation: Form a sequence of scalars (denoted by $s_{i,j}$ where i is the sequence index) and then form a sequence of sequence of scalars (where j is the set element index and an element of the set is a sequence of scalars). The "sequence of scalars" is called simply "scalars" or an element of the set of scalars, hence the name Set Of Scalars. An element is defined as a particular sequence in the set.
- Scalars are not entered in set or sequence notation. Rather each sequence of scalars are entered separately as a list of delimited numbers.
- Stacked bars are the summation, at each element index, of successive groups of data, while offset bars show absolute values.
- Pie charts show one group of data at a time. To pan groups touch and drag across the pie graph. Similarly with other representations.
- The [Edit Graph](#), and its skins, can be used to add distinction to the presentation such as bar gradients.

The figure below diagrams the Set Of Scalars task.

**Data Importing and Exporting Formats**

- The [Table Editor](#) atomic cell type is a textual value for column one and a scalar value otherwise. For column one, label elements must be separated by a return character delimiter while using column and block-select import. For all other columns, column, row and block-select import format is a list of scalars separated by a delimiter.
- The [Edit Data](#) (available in [Developer Mode](#) only) format is an ordered list of numbers separated by a blank. Each bar group (scalar sequence) is entered by first assigning a number from 1-20 in the textual row entry (corresponding to the element index in the set) and then navigating to that group's editing tool. 1 is the default and will do for a simple bar chart. To edit labels use 0 or 1 in the textual row entry.
- The [Fetch Data](#) (available in [Developer Mode](#) only) format is an ordered list of numbers separated by a blank or an XML structure.

Representations

The Set Of Scalars tasks implements the representations described in the following table. Remember that [Skins](#) can affect the representation and may even alter the representation to not conform to the following descriptions.

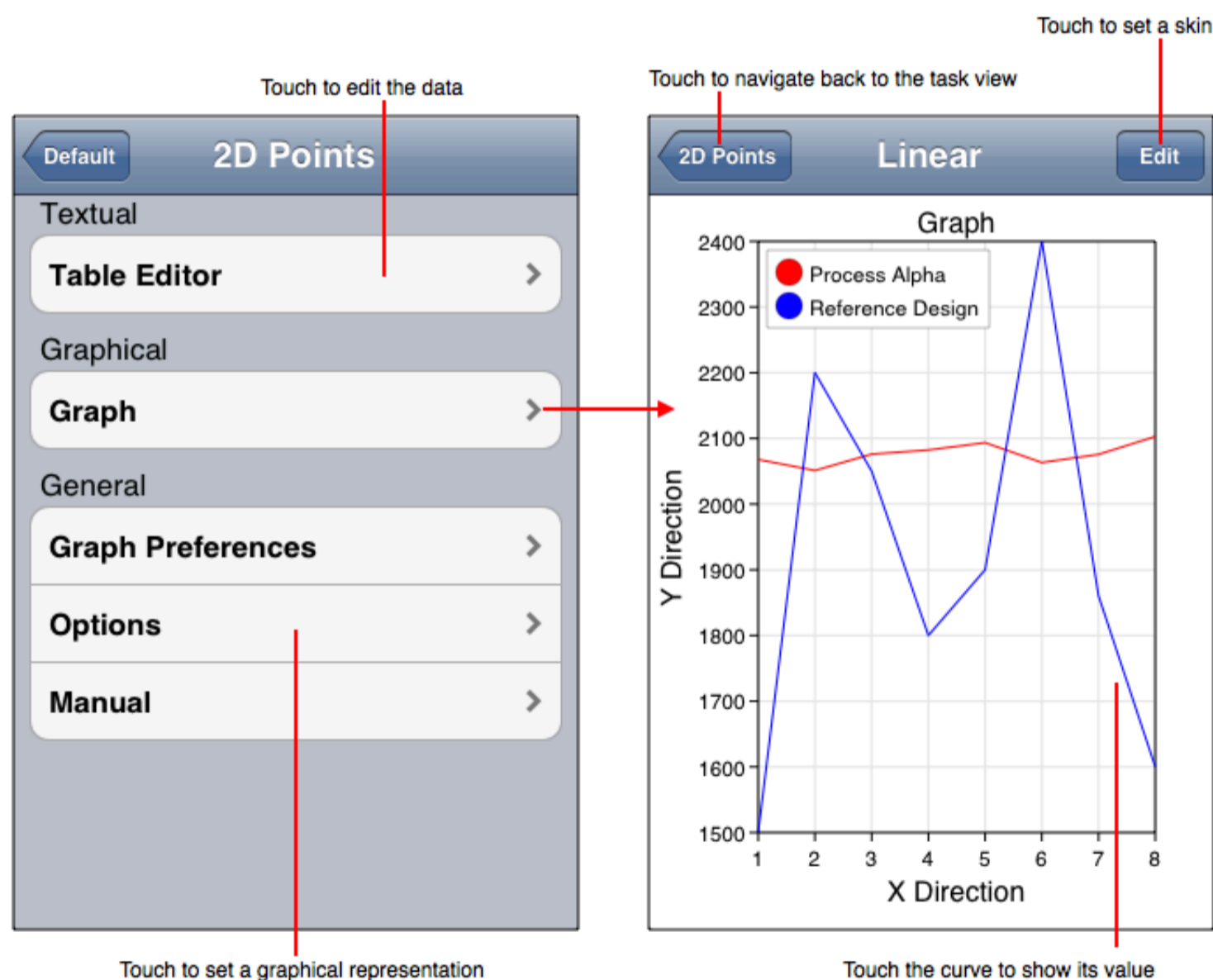
Representation	Description
Bar > Offset > Half	Shows a usual bar chart which has the following attributes: The bars are half the distance from each other, lengths oriented horizontally and sets of scalars are represented by successive offsets from the label for the particular scalar sequence index. The y-axis label is an integer or if the number of bars is too great then sparse integers unless (custom) labels are assigned.
Bar > Offset > Full	Same as Bar > Offset > Half except the bars touch each other.
Bar > Stacked > Half	Shows a stacked bar chart which has the following attributes: The bars are half the distance from each other, lengths oriented horizontally and sets of scalars are represented by successive summation of values for the particular scalar sequence index. The y-axis label is an integer or if the number of bars is too great then sparse integers unless (custom) labels are assigned.
Bar > Offset > Full	Same as Bar > Stacked > Half except the bars touch each other.
Column > Offset > Half	Shows a usual column chart which has the following attributes: The bars are half the distance from each other, lengths oriented vertically and sets of scalars are represented by successive offsets from the label for the particular scalar sequence index. The x-axis label is an integer or if the number of bars is too great then sparse integers unless (custom) labels are assigned.
Column > Offset > Full	Same as Column > Offset > Half except the bars touch each other.
Column > Stacked > Half	Shows a stacked bar chart which has the following attributes: The bars are half the distance from each other, lengths oriented vertically and sets of scalars are represented by successive summation of values for the particular scalar sequence index. The x-axis label is an integer or if the number of bars is too great then sparse integers unless (custom) labels are assigned.
Column > Offset > Full	Same as Column > Stacked > Half except the bars touch each other.
Pie Chart	Each set element of sequence of numbers is represented by a pie chart where each wedge has a arc-length proportional to the scalar it represents divide by the sum of scalars in the sequence. Note that only one pie chart can be shown at a time. In the case where there is more than one element of scalars then pan through the set by touch drag from left to right.

Graph > Tasks > Set Of 2D Points

The Set Of 2D Points task plots curves, well not really. The longer explanation is that it plots sets of sequences of pairs of scalars, but saying it plots curves is a lot easier. It also makes scatter plots and area graphs. By applying an appropriate skin using the [Edit Graph](#) tool it can also plot trajectories. Here are a few notes regarding this task:

- Data is defined like this: $\{ \{p_{1,1}, p_{2,1}, \dots, p_{n1,1}\}, \{p_{1,2}, p_{2,2}, \dots, p_{n2,2}\} \dots \{p_{1,m}, p_{2,m}, \dots, p_{nm,m}\} \}$. Translation: Form a sequence of 2D points (denoted by $p_{i,j}$ where i is the sequence index and $p_{i,j}$ is a pair of scalars $\{x_{i,j}, y_{i,j}\}$) and then form a sequence of sequence of points (where j is the set element index and an element of the set is a sequence of points). The "sequence of points" is called simply "2D Points" or an element of the set of points, hence the name Set Of 2D Points. An element is defined as a particular sequence in the set.
- Data are not entered in set or sequence notation. Rather each sequence of points are entered separately as a list of delimited pairs of numbers.
- Curves are formed from each sequence of points and multiple curves are implemented by elements of the set.
- The word "curve" is simply a convenience and does not indicate that a curve is defined. Curve is simply easier to understand.
- There is no enforcement of ascending x values in the numeric views, however when a curve is plotted then the points will be reordered to make x ascending. Scatter plots do not impose that restriction since they are not x or y ordered.
- Touching a curve or data point shows its values. Touch-drag over the chart scans one group of data (curve) at a time.

The figure below diagrams the Set Of 2D Points task.

**Data Importing and Exporting Formats**

- The [Table Editor](#) atomic cell type is a 2D Point, that is two scalars interpreted as the x-value and then y-value. The component cell type is a scalar representing x-values for odd numbered columns and y-values for even numbered columns. The column, row and block-select import format is a list of 2D Points separated by a delimiter for atomic cells and scalars for component cells.
- For the [Edit Data](#) (available in [Developer Mode](#) only) tool each curve (point sequence) is entered by first assigning a number from 1-20 in the Textual row entry (corresponding to the element index in the set) and then navigating to that curve's editing tool. 1 is the default and will do for a simple line graph. The format is a list of pair of numbers (2D points) separated by a blank.
- The [Fetch Data](#) format is a list of pair of numbers (2D points) separated by a blank or an XML structure.

Representations

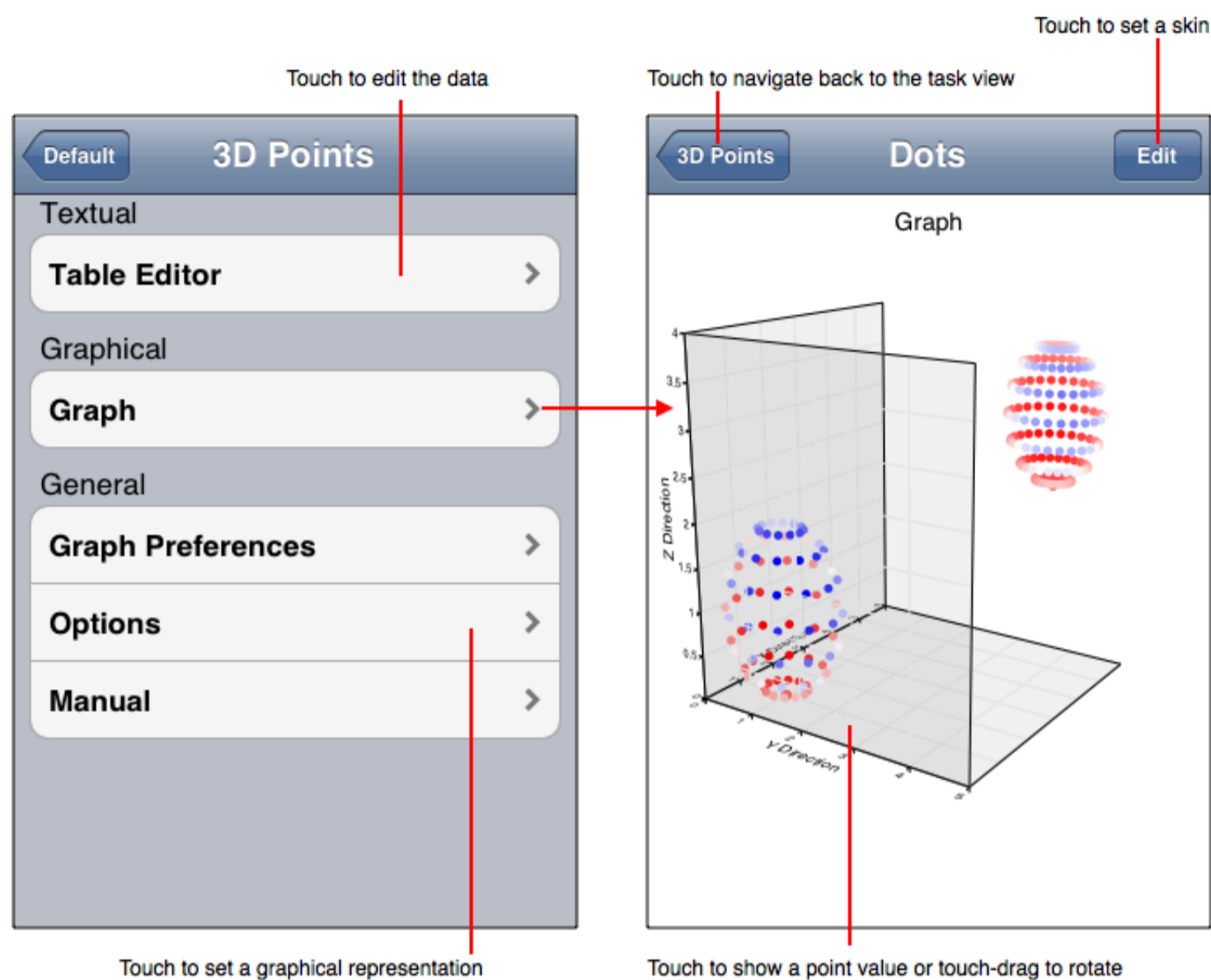
The Set Of 2D Points tasks implements the representations described in the following table. Remember that [Skins](#) can affect the representation and may even alter the representation to not conform to the following descriptions.

Representation	Description
Curve > Linear	Shows a usual line graph which is on a rectilinear (Cartesian, x-y) coordinate system. A line graph can have multiple curves on it. X values are reordered as needed to be ascending.
Curve > Semi-Log	Same as Curve > Linear except the coordinate system (and hence the graph) is semi log which means x-linear y-log. The log autoscaler is set to use full cycle or sub cycle log increments as needed to satisfy the data. Use Skins to change the plethora of autoscale parameters or to turn the autoscaler off. IMPORTANT: x and y are input in linear units and are presented on the log scale, do not enter log(y) values and expect things to work out.
Curve > X-Log	Same as Curve > Semi-Log except x-log y-linear coordinates.
Curve > Log-Log	Same as Curve > Semi-Log except x-log y-log coordinates.
Curve > Polar	Same as Curve > Linear except the data is shown on a polar coordinate system where x is interpreted as Θ and y as radius, i.e.: x and y are not transformed (via $r = \sqrt{x^2 + y^2}$, $\Theta = \arctan(y/x)$) but rather are interpreted as the independent variables of the coordinate system which means $x = \Theta$ and $y = \text{radius}$. Also note that this graph representation is not a true polar coordinate system as the origin is not zero but rather defined by the autoscaler and minimum r data value. To set the origin to zero use a Skins and turn the r-minimum autoscaler off and the limit value to zero. You can also fix the theta limits so that the coordinate system is shown by a partial polar grid, such as a semi-circular polar grid.
Curve > R-Log	Same as Curve > Polar except the r direction is a log scale not linear. Note that the origin is not zero as is required by log scales. Also note that the autoscaler will use sub or full cycle representations as needed.
Curve > Date	Same as Curve > Linear except the coordinate system (and hence the graph) is x-date y-linear. To enter dates see the Date Graph support section.
Area > Linear	Same as Curve > Linear except the area under the curve is filled in with a color, that is between y and y=0 so for $y < 0$ the area under the curve is actually above the curve. If $y = 0$ is not within the graph frame as when the y-axis minimum is greater than 0 then the area extends out of the graph frame. To accommodate that issue the clipping frame is turned on. Use a Skins to turn the clipping off if needed.
Area > Semi-Log	Same as Curve > Semi-Log except the area under the curve to the y-minimum frame value of the graph is filled with the expectation that it signifies the entire area under the curve.
Area > X-Log	Same as Curve > X-Log except the area under the curve to the y-minimum frame value of the graph is filled with the expectation that it signifies the entire area under the curve.
Area > Log-Log	Same as Curve > Log-Log except the area under the curve to the y-minimum frame value of the graph is filled with the expectation that it signifies the entire area under the curve.
Area > Polar	Same as Curve > Polar except the area from the r values of the curve to the origin of the coordinate system is filled in.
Area > R-Log	Same as Curve > R-Log except the area from the r values of the curve to the origin of the coordinate system is filled in.
Scatter > Linear	Same as Curve > Linear except the curve is now designated by circles at each point without interpolating line segments. Use Skins to change the circle to nearly anything and also to turn on the interpolating line segment and thus make a scatter plot into a trajectory plot.
Scatter > Semi-Log	Same as Scatter > Linear except the coordinate system is x-linear, y-log.
Scatter > X-Log	Same as Scatter > Linear except the coordinate system is x-log, y-linear.
Scatter > Log-Log	Same as Scatter > Linear except the coordinate system is x-log, y-log.
Scatter > Polar	Same as Scatter > Linear except the coordinate system is Θ , R. The information for Curve > Polar also applies
Scatter > R-Log	Same as Scatter > Linear except the coordinate system is Θ , R-log. The information for Curve > R-Log also applies.

Graph > Tasks > 3D Points

The 3D Points task makes 3D scatter plots of a sequence of triplet of numbers. It also makes 3D trajectory plots. Touching a point shows its value while touch-drag over the chart rotates it.

The figure below annotates the 3D Points task user interface.

**Data Importing and Exporting Formats**

- The [Table Editor](#) atomic cell type is a 3D Point and component cell type is a scalar. The column, row and block-select import format is a list of 3D points or scalars separated by a delimiter depending on atomic or component cell type representation.
- The [Edit Data](#) (available in [Developer Mode](#) only) tool format is a list of triplets of numbers (3D points) separated by a blank such as $x_1 y_1 z_1 x_2 y_2 z_2 \dots x_N y_N z_N$.
- The [Fetch Data](#) tool format is a list of triplets of numbers (3D points) separated by a blank such as $x_1 y_1 z_1 x_2 y_2 z_2 \dots x_N y_N z_N$ or an XML structure.

Representations

The 3D Points tasks implements the representations described in the following table. Remember that [Skins](#) can affect the representation and may even alter the representation to not conform to the following descriptions. If you need a combination of these representation types, such as a trajectory plot with each point marked by a dot (a combination trajectory and scatter plot) then set that in a skin.

Representation**Description**

Perspective With Dots	Shows a 3D perspective scatter plot on a 3D rectilinear coordinate system. Each 3D data point is denoted by a dot which is a small red circle. The dot (a marker) can be altered using Skins .
Perspective With Labels	Shows a 3D perspective scatter plot on a 3D rectilinear coordinate system. Each 3D data point is denoted by a number which is the sequence index of the point. The number font and other graphics can be altered using Skins .
Perspective With Lines	Shows a 3D perspective trajectory plot on a 3D rectilinear coordinate system. Each 3D data point connected by a line to adjacent points. The line width, dash and other graphical effects can be altered using Skins .

© Copyright 1993-2012 by [Vimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

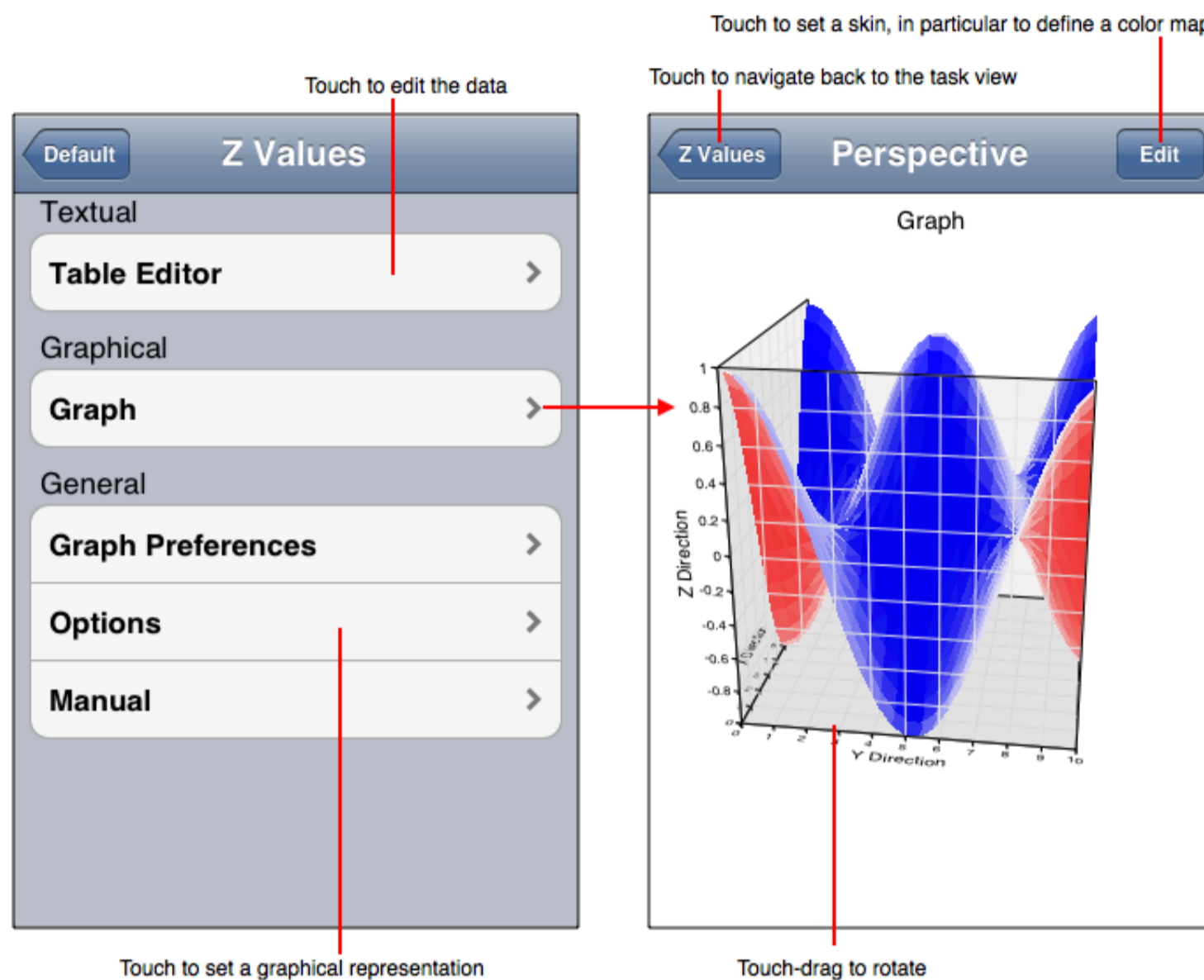
Graph > Tasks > Z Values

a.k.a: Scalars On a Regular 2D Grid

The Z Values task interprets a list of numbers as x-contiguous z-values on a x-y regular grid. The z-values correspond to height of cells on a 3D perspective plot or as cell amplitude on point fill plots. Some things to note about this task follow:

- This task makes surface graphs which means it interpolates z-values on a regular grid with a bi-linear function as a basis over a cell.
- The z-values are not actually plotted, that is a misnomer. What is actually plotted is a color gradation related to the z-values. For the flat (2D) point fill representation that gradation corresponds with the color map defined in the skin. For the 3D perspective plot, the color mapping is remapped using a normal-to-observer amplitude to give a sense of reflection and hence 3D quality. [Edit Graph](#) can be used to change some of the properties of the color mapping defined in the skin.

The Z Values task is diagrammed in the following figure.

**Data Importing and Exporting Formats**

- The [Table Editor](#) atomic cell type is a scalar value and the x and y grid lengths are interpreted as column and row lengths respectively. Column, row and block-select import format is a list of scalars separated by a delimiter.
- The [Edit Data](#) or [Fetch Data](#) (available in [Developer Mode](#) only) format is scalars separated by blanks. The x and y grid lengths are entered separately.

Representations

The Z Values tasks implements the representations described in the following table. Remember that [Skins](#) can affect the representation and may even alter the representation to not conform to the following descriptions.

Representation	Description
Perspective Surface Graph	Shows a 3D perspective surface graph on a 3D rectilinear coordinate system. Each Z Value is bi-linearly interpolated with adjacent values to form the surface. The color gradation can be altered using Skins .
Perspective Grid Graph	Shows a 3D perspective surface graph on a 3D rectilinear coordinate system. Each Z Value is connected to adjacent values by a line segment to form the grid.

Point Fill Graph > Linear	Shows each Z Value mapped to a color within a rectangle of the regular grid cell on a 2D rectilinear coordinate system. The color map can be altered using Skins .
Point Fill Graph > Semi-Log	Same as Point Fill Graph > Linear except for a semi-log coordinate system (x-linear, y-log).
Point Fill Graph > X-Log	Same as Point Fill Graph > Linear except for a x-log, y-linear coordinate system.
Point Fill Graph > Log-Log	Same as Point Fill Graph > Linear except for a x-log, y-log coordinate system.
Point Fill Graph > Polar	Same as Point Fill Graph > Linear except for a polar coordinate system.
Point Fill Graph > R-Log	Same as Point Fill Graph > Linear except for a polar r-log coordinate system.

© Copyright 1993-2012 by [Wimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

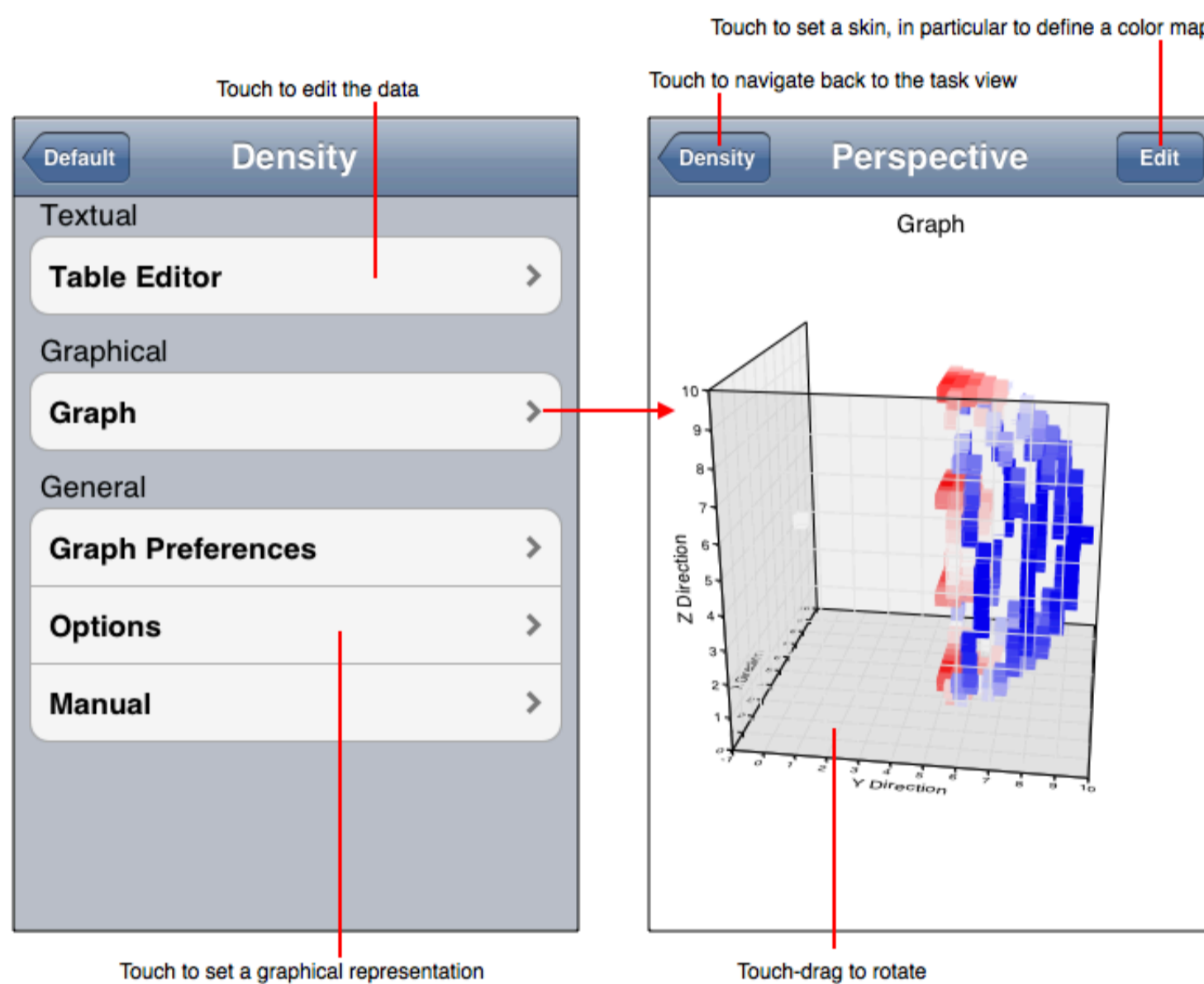
Graph > Tasks > Density Values

a.k.a: Scalars On a Regular 3D Grid

The Density Values Task takes a sequence of scalars and presents them as boxes on a uniform 3D grid. The scalars have values between 0 and 1 and any value outside that range (especially -1) means "no density". By selecting one of the point fill representations densities can be scanned in the z-direction one x-y plane at a time by touch drag over the point fill plot. Here are a few things to note about this task:

- The number of density values must equal the dimension values, entered on the [Edit Data](#) tool, multiplied together.
- The color map in the skin is key to making a good density plot. In particular, transparency is important to see within the density field. Many times a particular density is of interest and the color map can be adjusted so that that particular density value can show up as a unique color in the color map.
- Experimenting with the Volume 3D Data Graphic in Vwidget Builder might help with understanding the Density Values Task.
- The 3D perspective graph is intended to give an overall qualitative indication of the data. The Z Splice point fill representation gives a more quantitative representation. Touch-drag over the 3D graph rotates it. While in Z Splice representation touching a density (in the x-y plane) shows its value and touch-drag scans each x-y plane successively.

The figure below diagrams the Density task.

**Data Importing and Exporting Formats**

- The [Table Editor](#) cell type is a scalar. The column, row and block-select import format is a list of scalars separated by a delimiter. The grid lengths are set in the Table's Option tool.
- The [Edit Data](#) (available in [Developer Mode](#) only) format is a list of scalars separated by a blank.
- The [Fetch Data](#) (available in [Developer Mode](#) only) format is a list of numbers separated by a blank or an XML structure.

Representations

The Density Values tasks implements the representations described in the following table. Remember that [Skins](#) can affect the representation and may even alter the representation to not conform to the following descriptions.

Representation**Description**

Perspective Volume	Shows a 3D perspective volumetric plot on a 3D rectilinear coordinate system. Each density value is denoted by a cube on a 3D regular grid. The cube color, which corresponds to density, can be altered using Skins . By selecting color maps with appropriate transparency the density can be semi-translucent so that inner portions of the density can be viewed.
Z Slice	Shows a single plane of the density on a rectilinear coordinate system. In this representation, the z-plane can be panned by touch-drag from left to right.

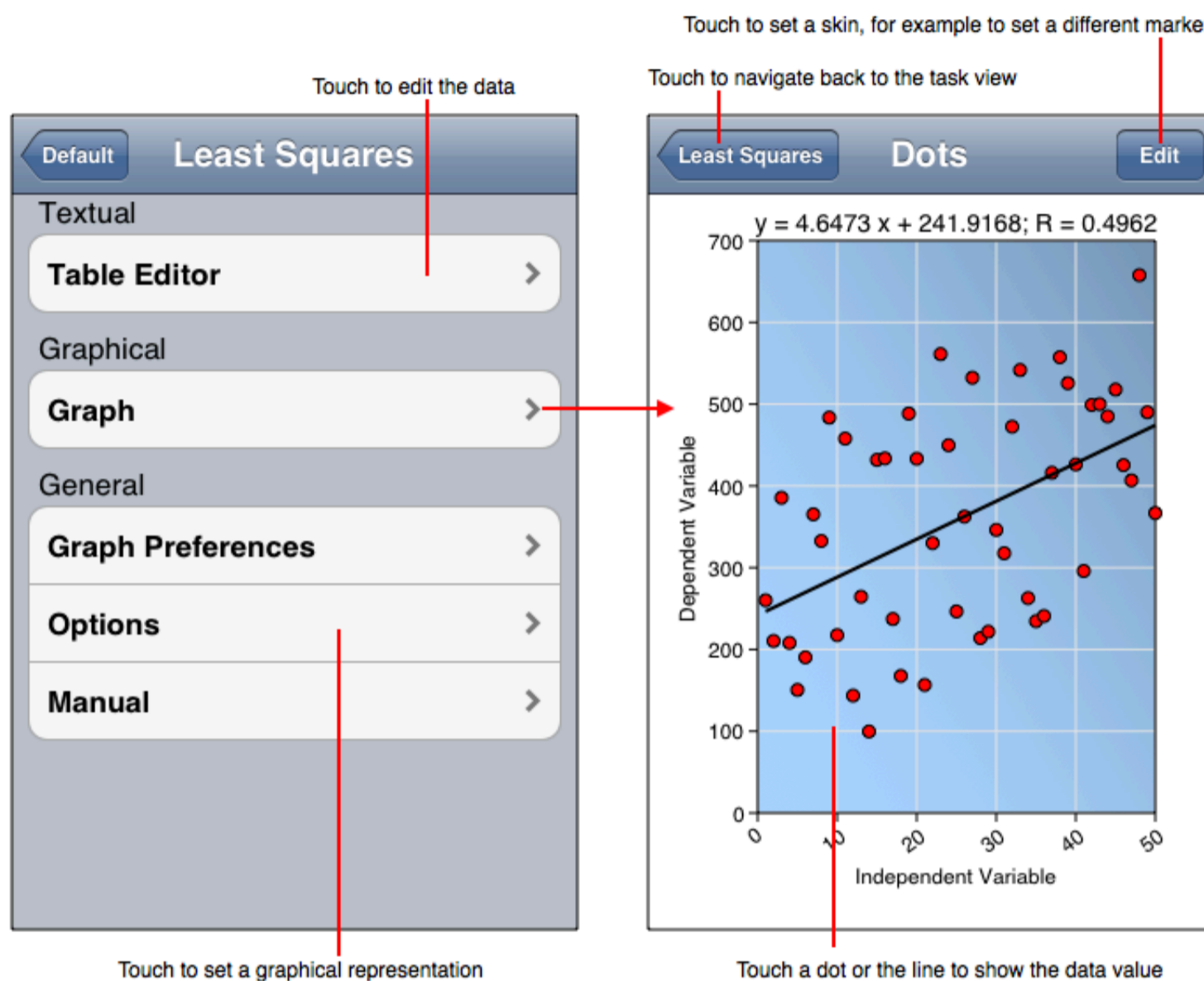
© Copyright 1993-2012 by [VVI](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

Graph > Tasks > Least Squares

Least Squares takes 2D Points and plots them as a 2D scatter plot and also plots a linear regression of those 2D points. Here are a few things to note about this task:

- Touching a data point shows its values and touching the line segment shows the interpolated value of the regression. If you select the Labels representation then you can see the sequence number of each data point.

The figure below annotates this task's user interface.

**Data Importing and Exporting Formats**

- The [Table Editor](#) atomic cell type is a 2D Point, that is two scalars interpreted as the x-value and then y-value. The component cell type is a scalar representing x-values for the first column and y-values for the second column. The column, row and block-select import format is a list of 2D Points separated by a delimiter for atomic cells and scalars for component cells.
- The [Edit Data](#) format is a list of pair of numbers (2D points) separated by a blank.
- The [Fetch Data](#) format is a list of pair of numbers (2D points) separated by a blank or an XML structure.

Graph > Tasks > Error Bars

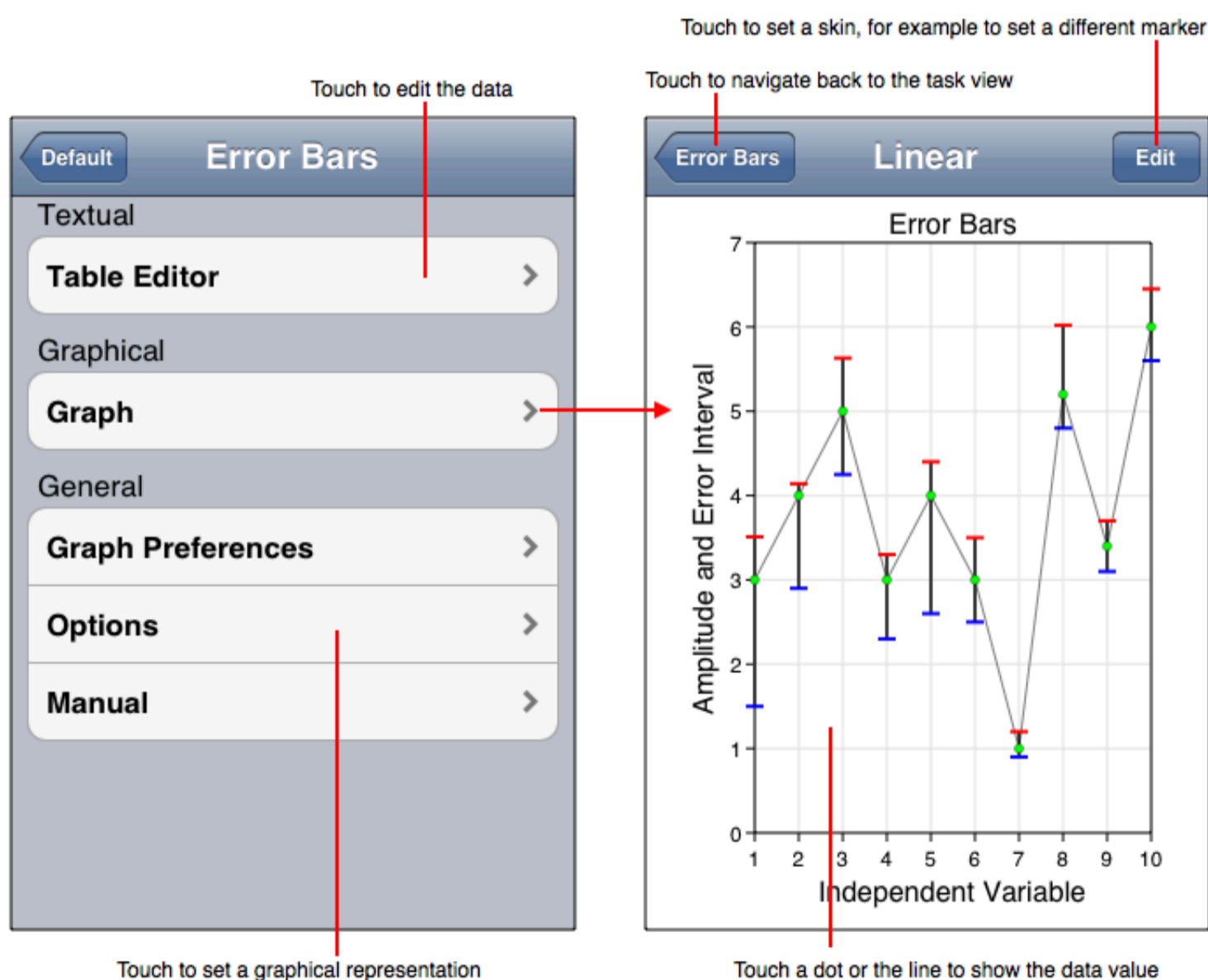
The Error Bars task helps make an error bar graph. Enter the data according to the following table.

Column Name	Explanation
X Values	The x-value for the data point and error bars. The data and error bar x-position are always identical.
Y Values	The y-value of the data (the y-location of the green dot)
Y-Delta Min	The length from the y-value of the data point to the lower part of the bar (the distance between the green dot and the blue bar).
Y-Delta Max	The length from the upper part of the bar to the y-value of the data point. (the distance between the red bar and the green dot).

Some things to note about this task are:

- Touch a data graphic component (green dot, red bar or blue bar) to edit its value.
- The data value (represented by the green dot) is entered in absolute value. However, the error bar data (y-delta min and y-delta max) are entered as the distance from the data point (relative values).
- To assign legend labels touch hold a column header and select Edit. See [Tables](#) for additional information.

The figure below annotates this task's user interface.

**Data Importing and Exporting Formats**

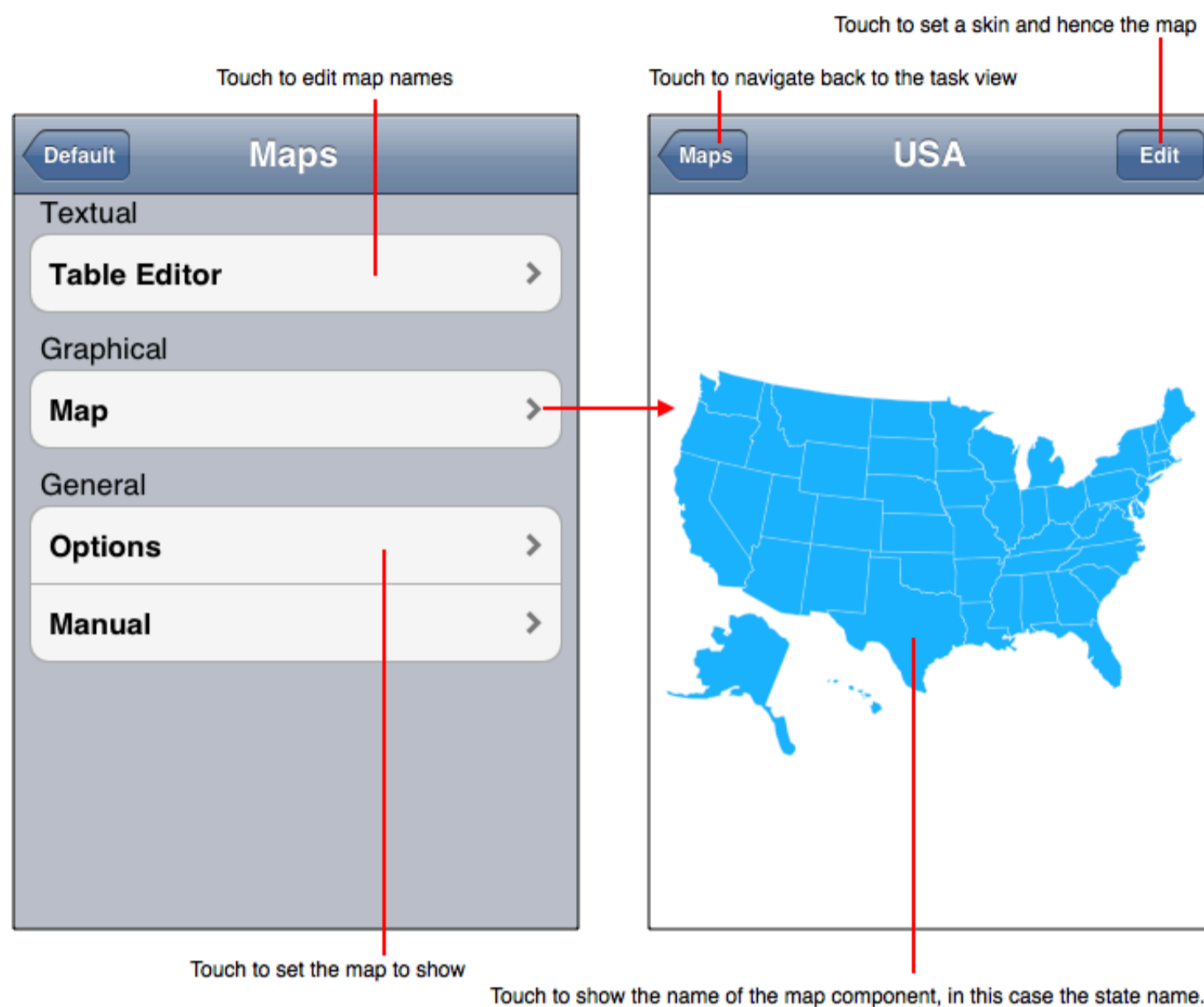
- The [Table Editor](#) cell type is a single number for all cases.
- The [Edit Data](#) format is a list of pair of numbers (2D points) separated by a blank.
- The [Fetch Data](#) format is a list of pair of numbers (2D points) separated by a blank or an XML structure.

© Copyright 1993-2012 by [VViimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

Graph > Tasks > Maps

The Map task shows a map. This is an unusual task because the "data" is really the Skin itself and the data entries, which are names of the map, are auxiliary. Hence, with the exception of the default USA map, a skin must be made to use the map task. Here are a few points regarding this task:

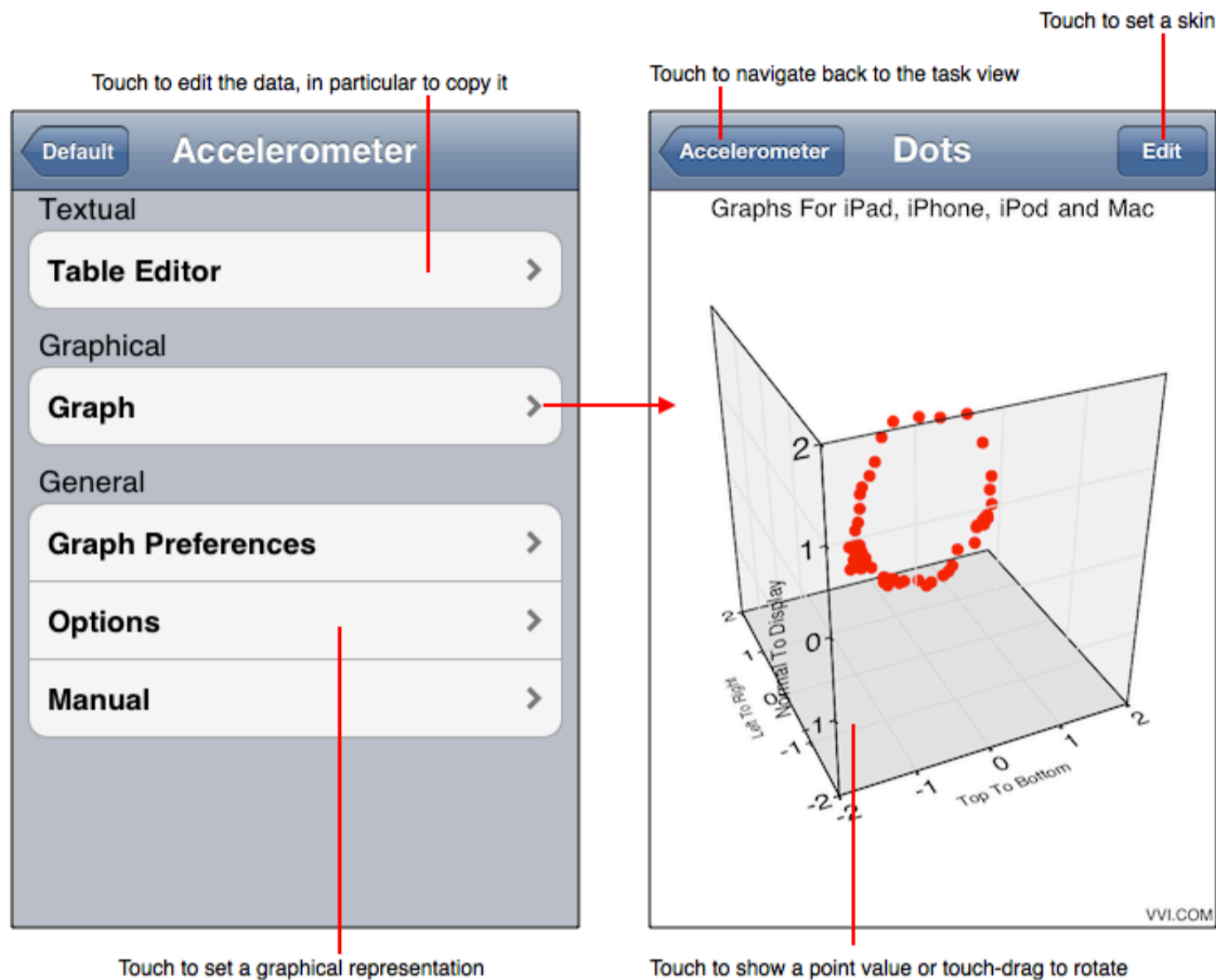
- The [Making A Map](#) Tutorial shows how to make a Vwidget Builder document and skin that is then imported using the [Edit Graph](#) tool.
- The data entries are somewhat superfluous to using a map.

**Data Importing and Exporting Formats**

- While importing data using [Edit Data](#) or [Fetch Data](#) keep in mind that the format is a list of names with one name per line.

Graph > Tasks > Accelerometer Vectors

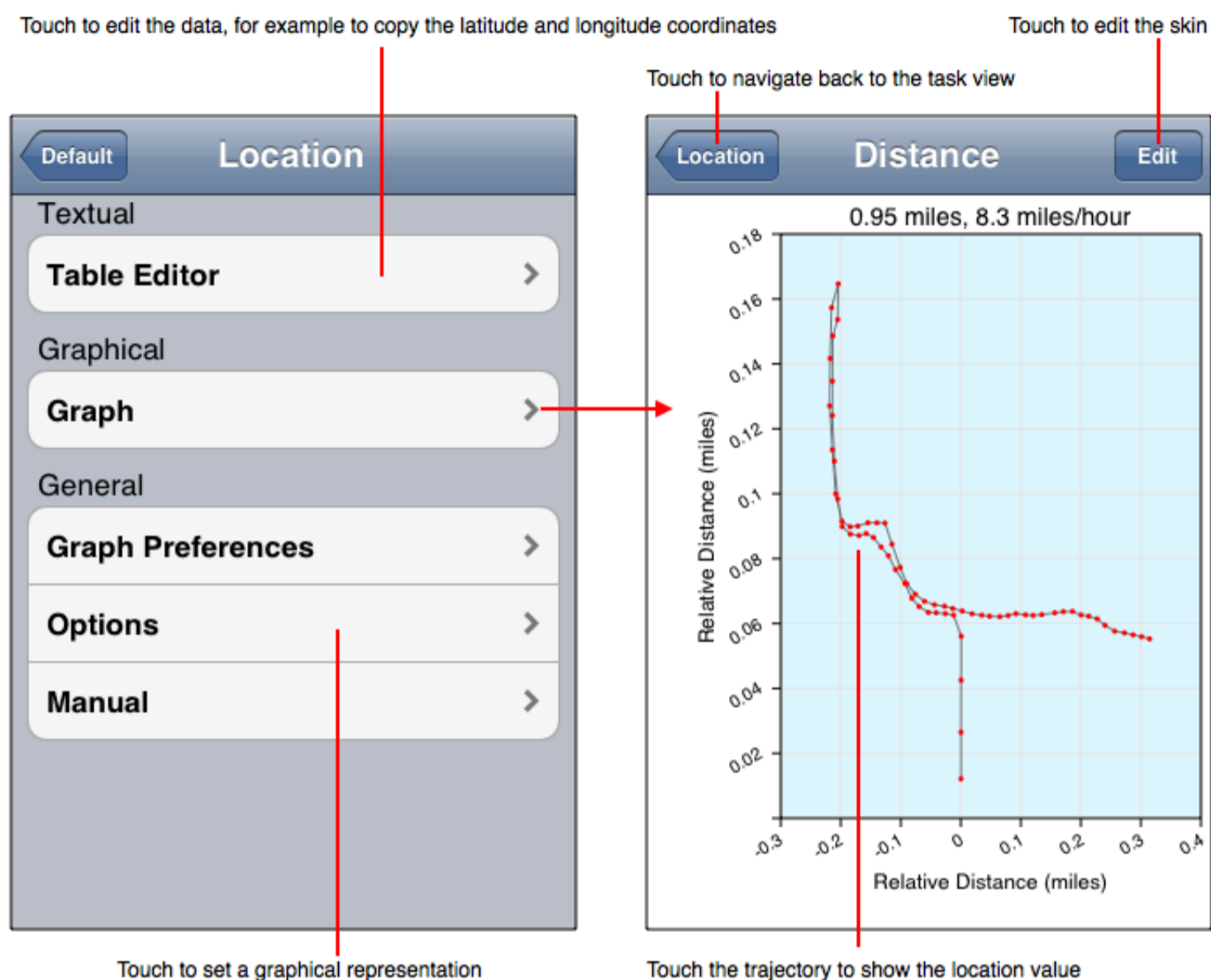
The Accelerometer Vectors task acquires acceleration data from the device and plots it as a 3D perspective plot as shown in the following figure.



© Copyright 1993-2012 by [VVI Imaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

Graph > Tasks > Location Tracking

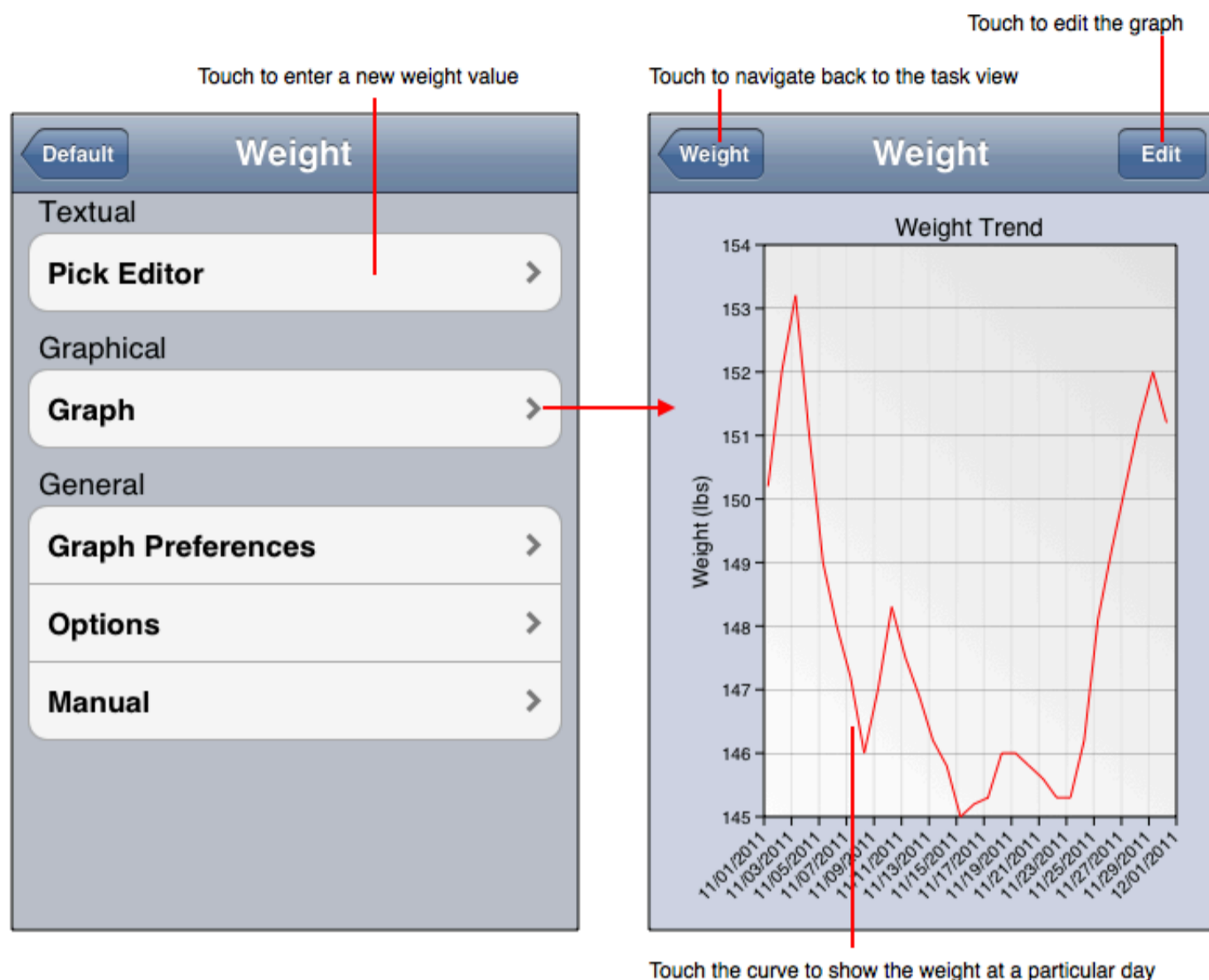
The location tracking task will plot your location on a latitude-longitude graph. To operate first touch the [Options](#) tool and touch the Start Recording button. Once recording is on your location is acquired and plotted. Since the acquisition may be over an extended period of time this task also disables your device from sleeping. To reenale sleeping turn the acquisition off or go to a different task.



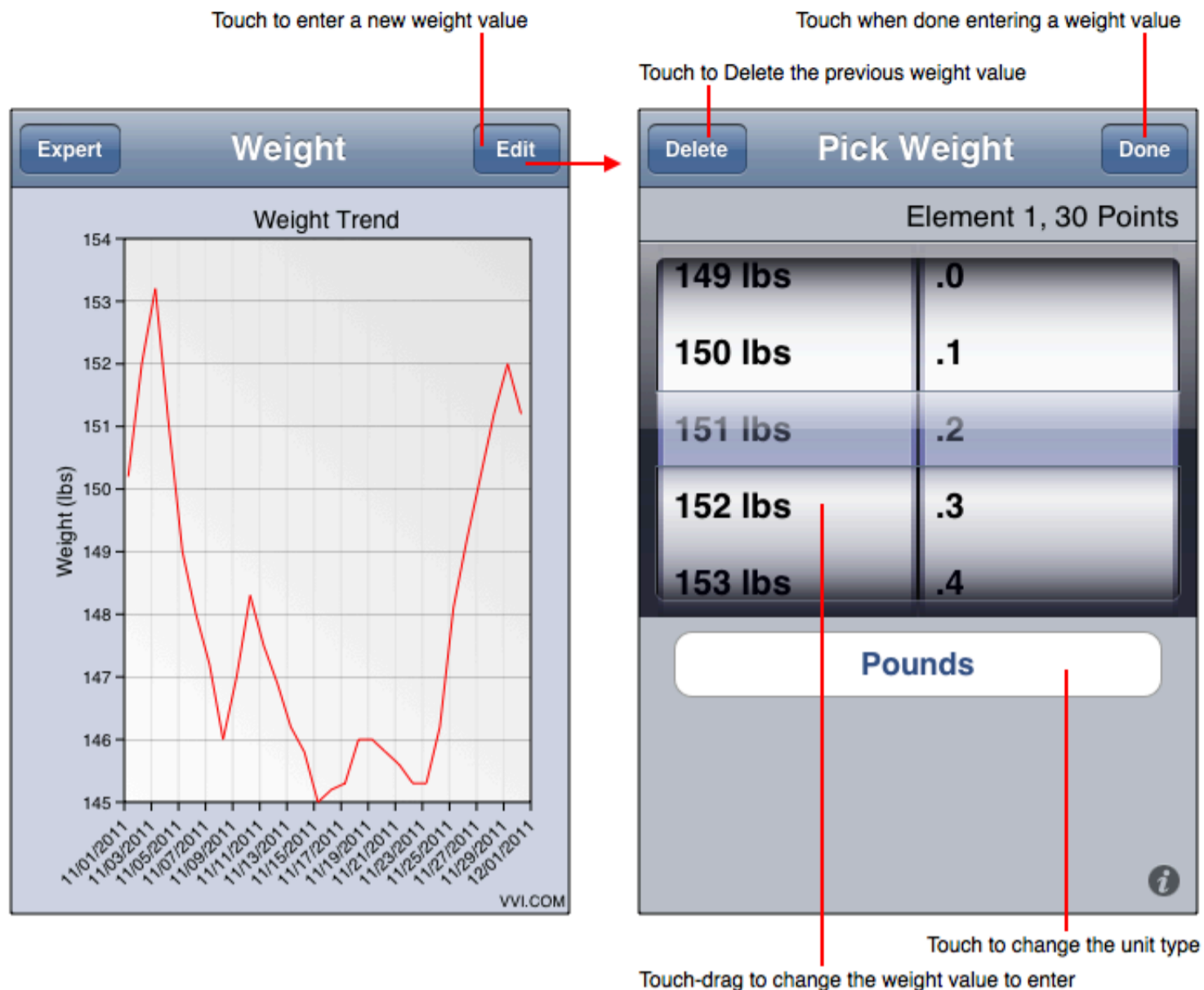
[Graph](#) > [Tasks](#) > **Weight**

The Weight task is used to conveniently graph your weight on a periodic basis. To operate set a weight value with the picker and then view a representation. Do that over the course of several days in order to see a weight trend.

The Weight task is diagrammed in the following figure.



The single purpose mode displays the Weight task as diagrammed below.

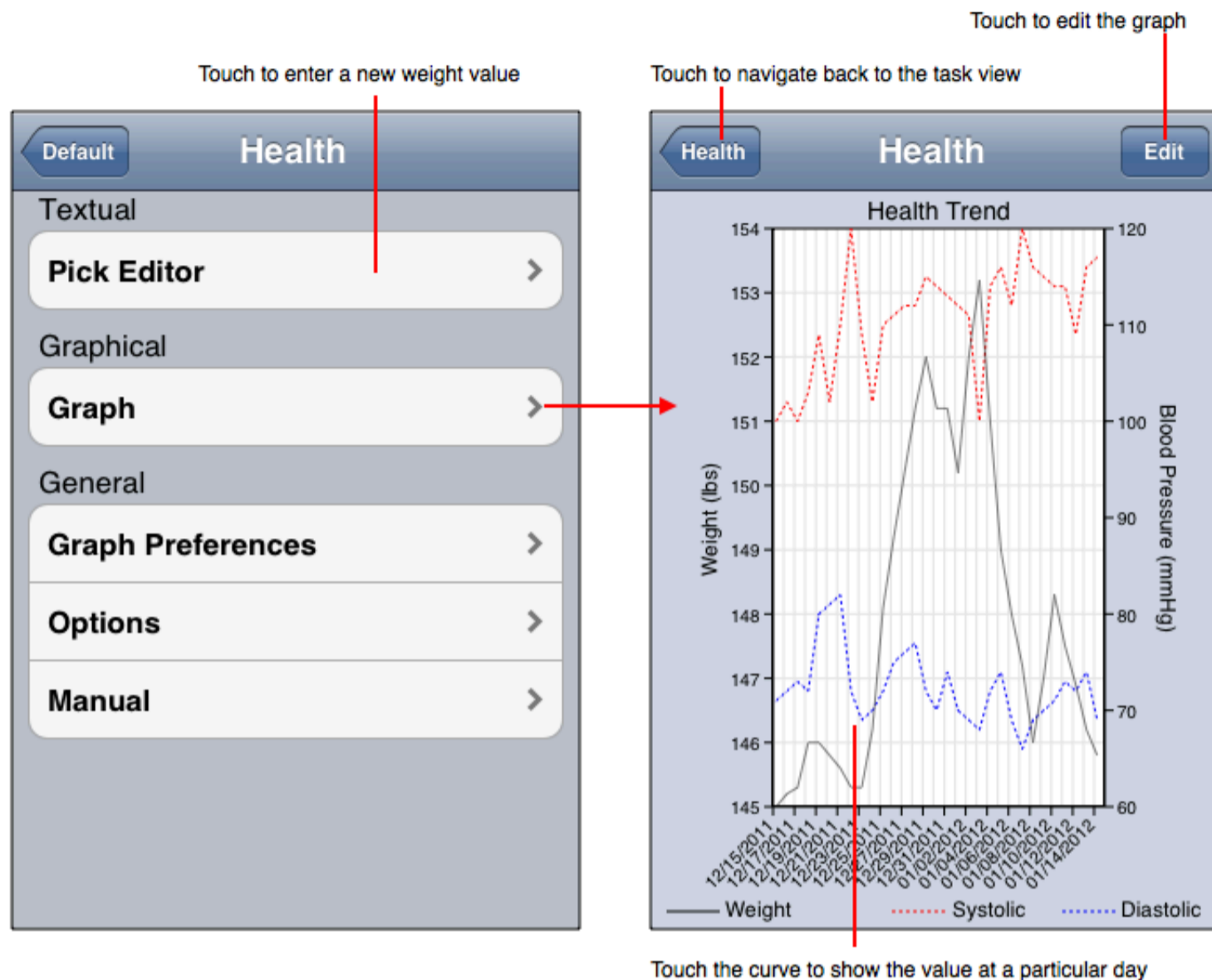


Data Importing and Exporting Formats

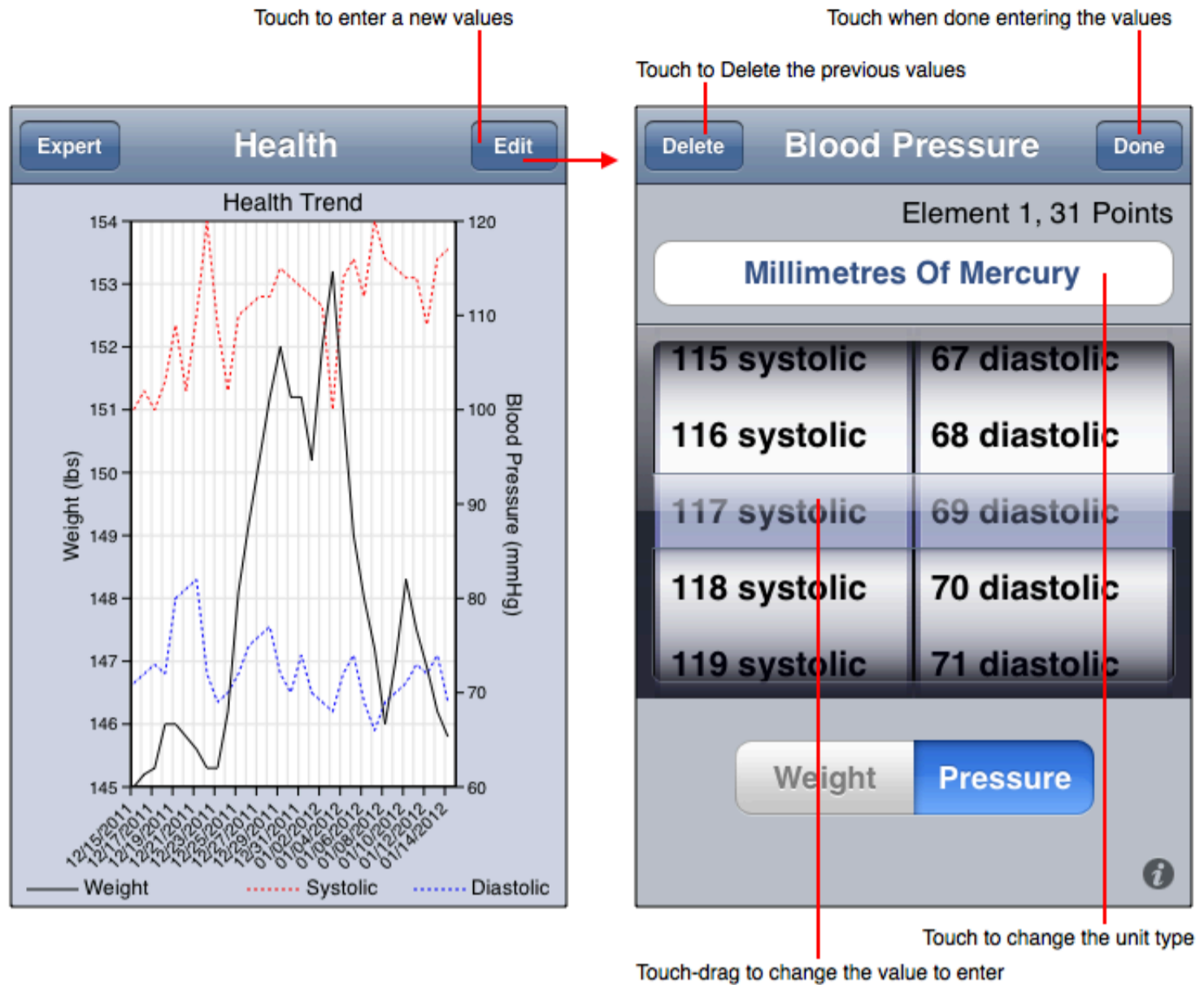
- The [Table Editor](#) atomic cell type is a x and y value (2D Point) where the x-value is formatted as a gregorian date (see: [Date Graph](#)). Column, row and block-select import format is a list of 2D Points separated by a delimiter.
- The [Edit Data](#) or [Fetch Data](#) (available in [Developer Mode](#) only) format is 2D Points (x and then y-values) separated by blanks.

Graph > Tasks > Health

The Health task is used to conveniently graph your weight and blood pressure on a regular basis. To operate set a weight or blood pressure value with the picker and then view a representation. Do that over the course of several days in order to see a health trend.



The Health Graph app default mode displays the Health task in a simple mode as diagrammed below.



Touching the Expert button places the application in the more standardized and generalized Graph mode.

Data Importing and Exporting Formats

Data formats are the same as [Set of 2D Points](#) with the following key:

Element Number	Description
1	Weight
2	Systolic Blood Pressure
3	Diastolic Blood Pressure

The x-value is a date format and the y-value is the value of the health parameter.

© Copyright 1993-2012 by [VVI](#) imaging, Inc. (VVI); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

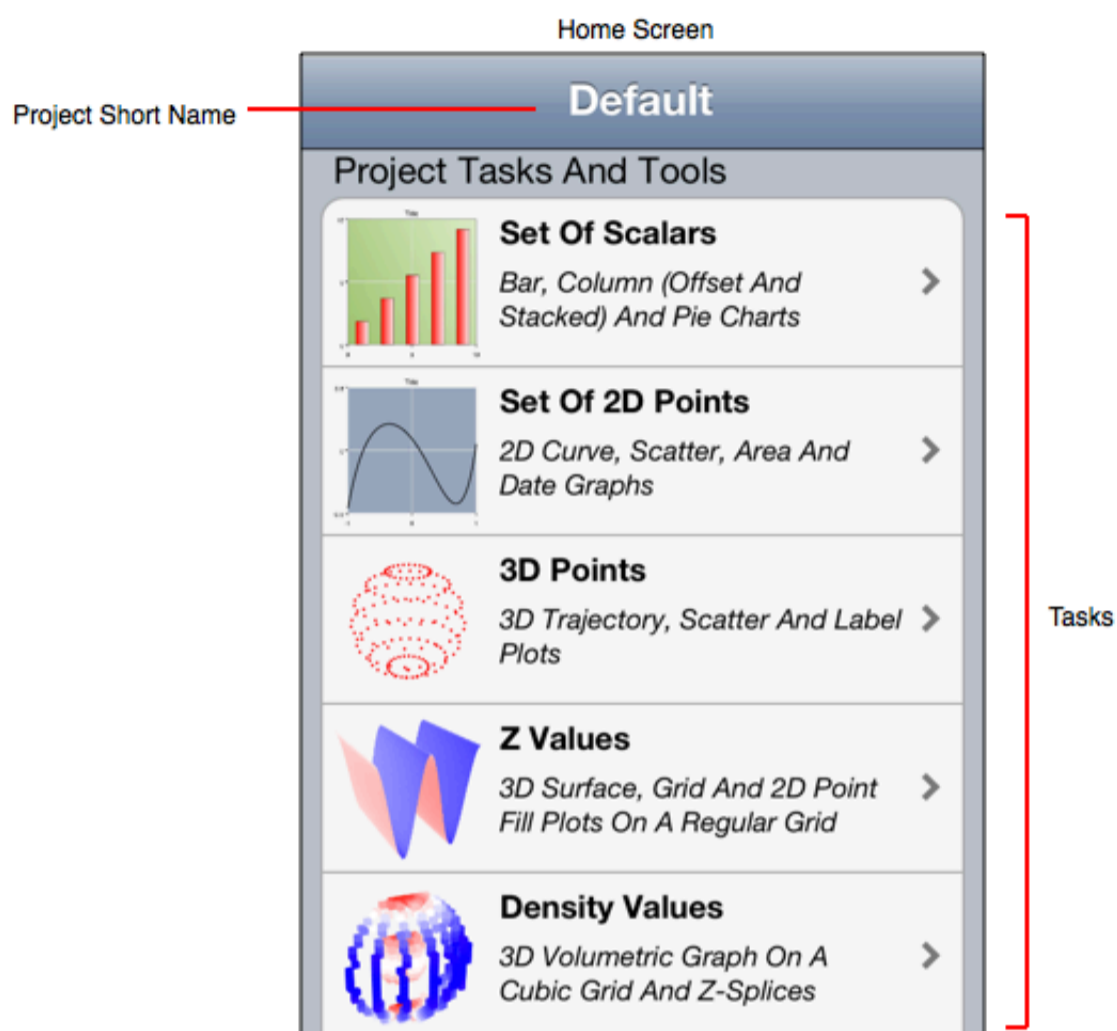
Graph > Tools

Tools operate on [Tasks](#) and are implemented as separate views. The following sections describe Tools.

Tool	Description
Home	The home tool is a task selector. Tasks are organized by data type and are one of Set Of Scalars , Set Of 2D Points , 3DPoints , Z Values , Density Values , Least Squares , Maps , Accelerometer Vectors or Location Tracking .
Projects	The Projects tool defines projects and sets a current project. Data and preferences are stored on a project basis and you can define new projects or remove projects as needed. You can also set a project to the current project. Only the current project's data, preferences and skins are used by the tasks.
Purpose	Use this tool to set the overall purpose of the application.
Fetch Data	Use this tool to set URLs for fetching data from the web.
Table Editor	Use this tool to edit data in a table format.
Edit Data	Use this tool to edit data.
Show Data	Use this tool to show data.
Edit Graph	This tool is used to define skins for the graph of the current project, task and data representation type. As a matter of convenience, it also has an animation button.
Preferences	This tool sets basic preferences that are common to all the representations of the task it operates on.
Reset	This tool resets the data of the task.
Info	This tool shows the key value pairs (dictionary) that generates the graph displayed in the task.
Export	This tool provides means to export a graph to other applications and data stores.
Help	This tool shows help for the current task. Help is built into the Tasks, however this manual provides much more extensive descriptions of Tasks.

Graph > Tools > Home

The home tool is where it all begins and is the tool that is presented when you first launched Graph. The figure below shows the home tool view. Essentially, it is a table of rows where each row can be touched to navigate to [Tasks](#).

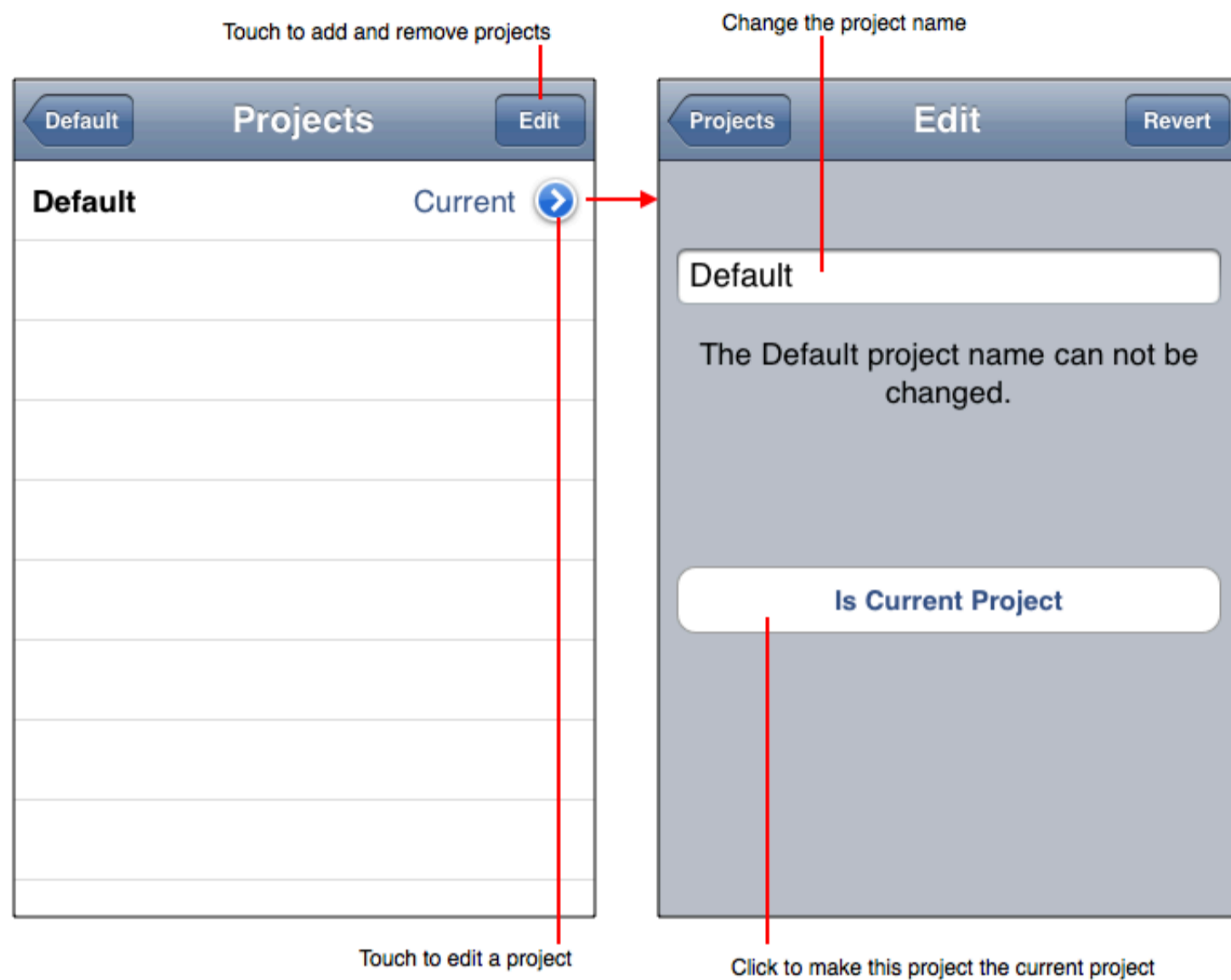


Each task is explained in the [Tasks](#) section.

Graph > Tools > Projects

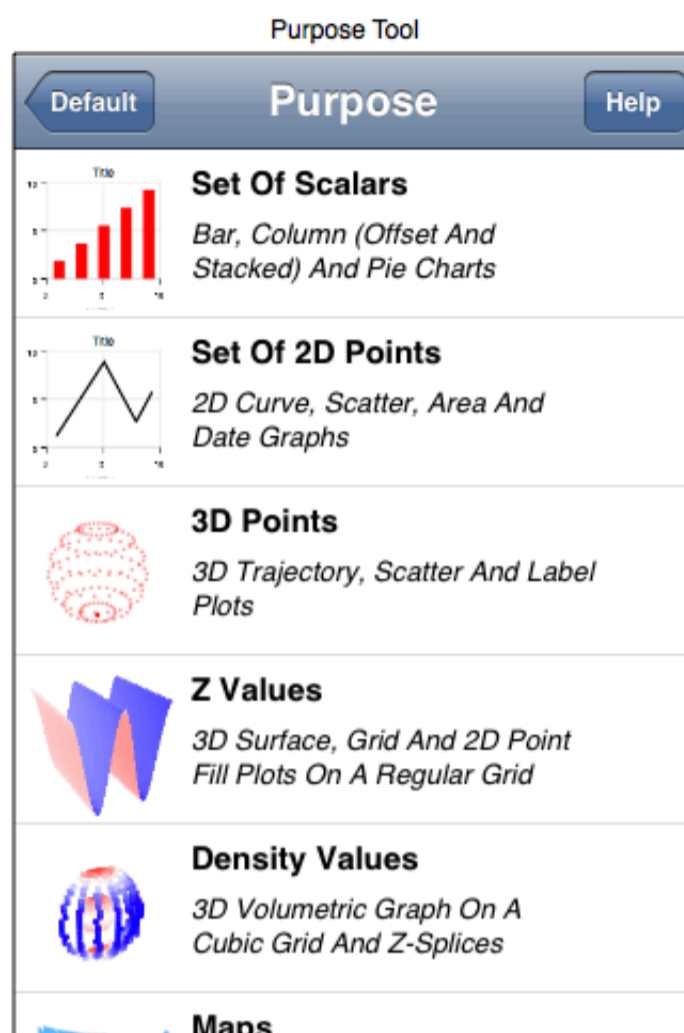
The Tasks' data, preferences, skins and other information are stored in a project. When you first use Graph the current project is the Default project. That project can not be renamed or removed. You can add new projects and delete and rename projects other than the Default project. Project preferences, data and other parameters are retrieved, saved and used without any explicit action on the part of the user.

Use the Projects tool to add and remove projects and set the current project. The following figure diagrams the Projects tool. As you can see you can add projects to a list, select projects and make a project the current project. Once a project is the current project its Tasks are available for use.



Graph > Tools > Purpose

The Purpose tool is used to refine the purpose of the Graph application. Touching a row in the table converts the user interface to the specific purpose.



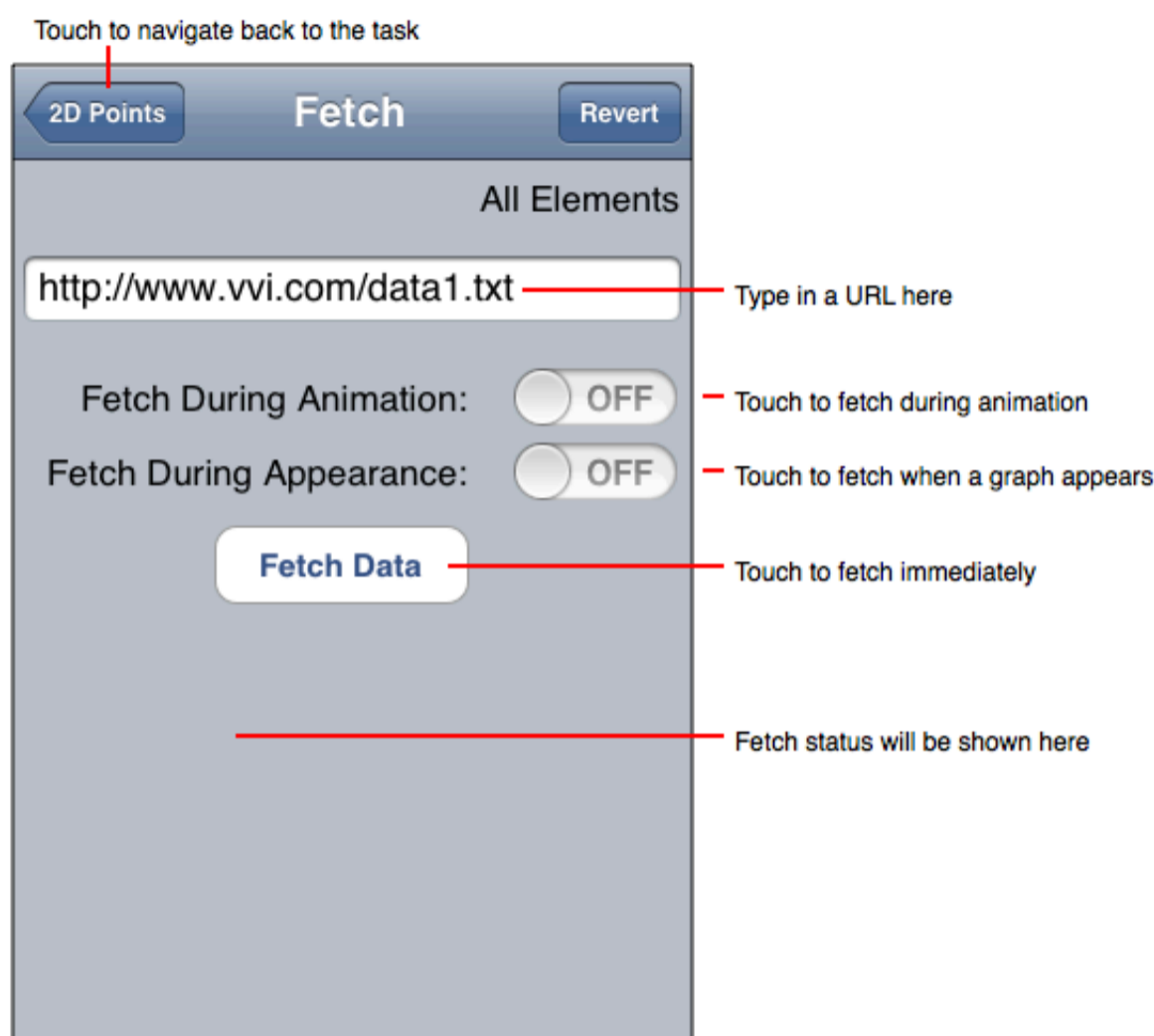
The two interface modes are defined as follows:

- **Expert**: The expert mode is based on nesting of features. To access those features you must navigate back and forth between the nesting levels.
- **Easy**: Selecting a purpose places the interface in an easy-to-use format where the interface is restricted only to what is required to satisfy the purpose. As such, the purpose interface is more efficient and easy to use, however it also sacrifices a powerful nested navigation metaphor and generality.

Either way, Easy or Expert, there are options to help satisfy your task at hand. If a purpose is grayed out then click it to buy Vvidget. The Vvidget application includes the grayed purpose as well as all the other features, does not have ads and is available at a modest price.

Graph > Tools > Fetch Data

Use the Fetch Data tool (available in [Developer Mode](#) only) to retrieve data from a web server. The Fetch Data tool is diagrammed here:

**URL Types**

The URL that can be used is defined here:

- Web: A web URL begins with "http://" and the rest of the URL is the path to the data file, such as: "http://www.vvi.com/data/mypoints.txt" (without the quotes). Notice that the path extension is "txt" in this case. Although the extension is arbitrary, txt is a safe extension as it informs a web server to use an ASCII MIME type. In practice, any extension, or no extension, also works. This is the same URL that you would type into a web browser to download the data. The data file must be a resource of the web server that the URL points to, either as a static file or a dynamic URL that retrieves the data file bytes from other places.

Notice that the Web URL gives access to a programmable system. For example, you can turn on your Web Sharing in the system preferences and then use PHP, perl or other scripting languages to write algorithms to retrieve computed data. The URL would be of this form: "http://localhost/cgi-bin/myalgorithm?data=lab1§ion=2¶m1=5" and you use the scripting engine's built-in form facilities to parse the URL parameters as input to an algorithm. Alternatively, the URL could point to an existing web service to retrieve SOA type information from queries.

Content Format Type

Generally the content is formatted as:

- A list of numbers for the [Set Of Scalars](#) task.
- A list of 2D points (two numbers) for the [Set Of 2D Points](#) task, such as: x1 y1 x2 y2 ... xN yN.
- A list of 3D points (3 numbers) for the [3D Points](#) Task.
- A list of numbers for the [Z Values](#) and [Density Values](#) tasks.

On the task page, if the Fetch Data row does not have a text entry or "All" is entered then the content retrieved from a URL (the response bytes) can be XML. For an explanation of this content see the [XML Fetch](#) tutorial.

Note that the fetch content format is simple, it is just a list of numbers with a blank delimiter. No fancy XML formatting and no dimension data. It is intended be as simple as possible in order to facilitate ease of use. Data formats are also explained in each Task's Help tool and

section in this manual.

Fetch Timing

Fetching data can occur at different time intervals as defined here:

- **Fetch Data:** Clicking the Fetch Data button fetches the data immediately.
- **Each Animation Step:** Selecting the Each Animation Step switch causes the data to be fetched upon each animation, which is at an interval of one second. If you use this button then go to the Data tool and select Animate.
- **At First Appearance:** Selecting the At First Appearance switch causes the data to be fetched when the task is made visible. Tasks are only made visible when switching tasks so this operation does not occur too frequently.

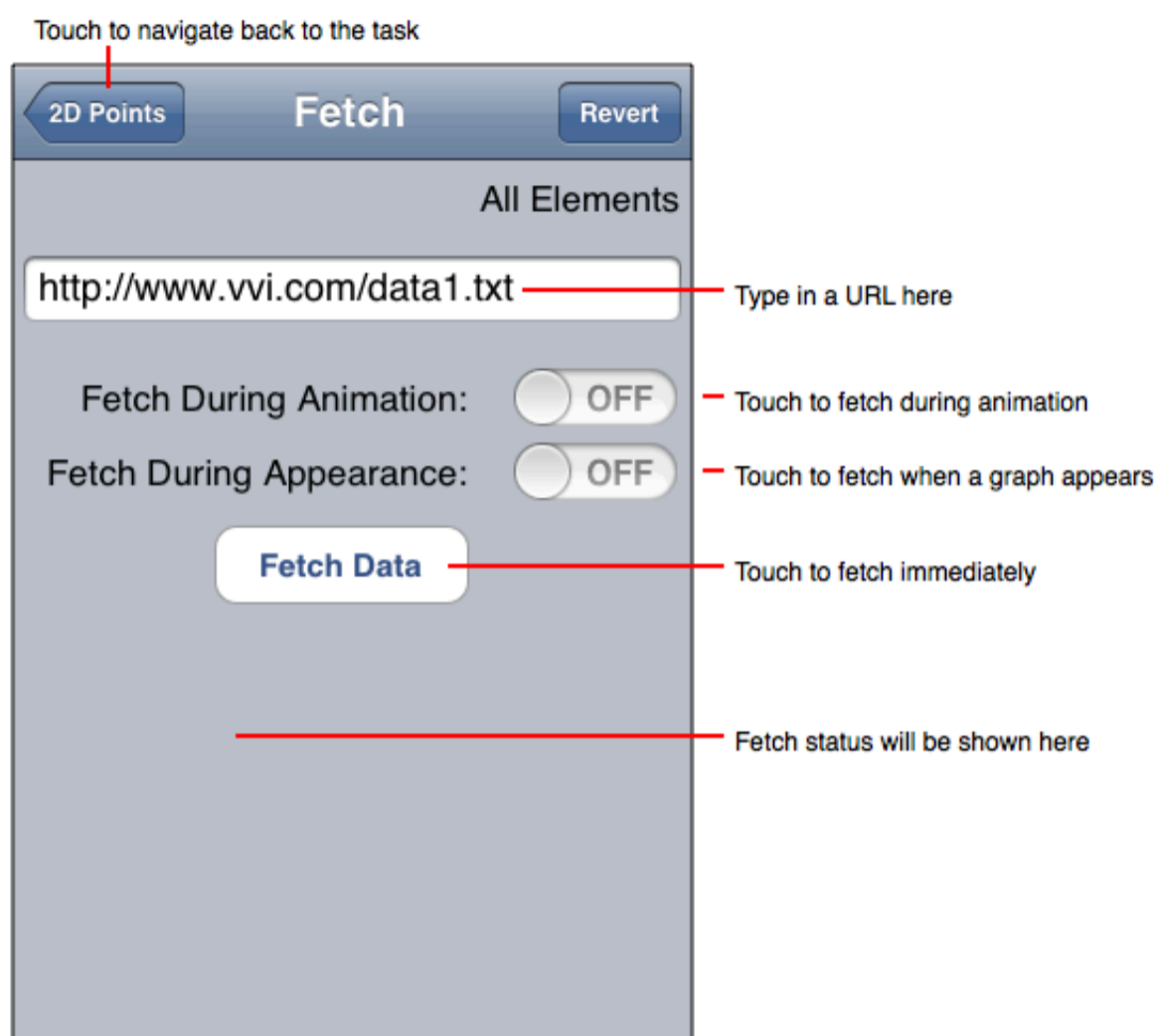
Fetch Operation

In a nutshell, when a fetch occurs the data is retrieved, entered into the task and then displayed. Usually the fetch appears to be atomic, that is: it seems to be a single operation. However, fetching is asynchronous and non-atomic. You may notice this fact if the fetch takes a long time. Also, for tasks that have multiple columns the fetch is on a per-column basis and the fetch retrieves each column one at a time until all columns are fetched and then updates the task with fetched data. Multiple column fetches are done in parallel and asynchronously and each column fetch rendezvous to amalgamate the fetch into a single result.

© Copyright 1993-2012 by [Vimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

Graph > Tools > Fetch Data

Use the Fetch Data tool (available in [Developer Mode](#) only) to retrieve data from a web server. The Fetch Data tool is diagrammed here:

**URL Types**

The URL that can be used is defined here:

- Web: A web URL begins with "http://" and the rest of the URL is the path to the data file, such as: "http://www.vvi.com/data/mypoints.txt" (without the quotes). Notice that the path extension is "txt" in this case. Although the extension is arbitrary, txt is a safe extension as it informs a web server to use an ASCII MIME type. In practice, any extension, or no extension, also works. This is the same URL that you would type into a web browser to download the data. The data file must be a resource of the web server that the URL points to, either as a static file or a dynamic URL that retrieves the data file bytes from other places.

Notice that the Web URL gives access to a programmable system. For example, you can turn on your Web Sharing in the system preferences and then use PHP, perl or other scripting languages to write algorithms to retrieve computed data. The URL would be of this form: "http://localhost/cgi-bin/myalgorithm?data=lab1§ion=2¶m1=5" and you use the scripting engine's built-in form facilities to parse the URL parameters as input to an algorithm. Alternatively, the URL could point to an existing web service to retrieve SOA type information from queries.

Content Format Type

Generally the content is formatted as:

- A list of numbers for the [Set Of Scalars](#) task.
- A list of 2D points (two numbers) for the [Set Of 2D Points](#) task, such as: x1 y1 x2 y2 ... xN yN.
- A list of 3D points (3 numbers) for the [3D Points](#) Task.
- A list of numbers for the [Z Values](#) and [Density Values](#) tasks.

On the task page, if the Fetch Data row does not have a text entry or "All" is entered then the content retrieved from a URL (the response bytes) can be XML. For an explanation of this content see the [XML Fetch](#) tutorial.

Note that the fetch content format is simple, it is just a list of numbers with a blank delimiter. No fancy XML formatting and no dimension data. It is intended be as simple as possible in order to facilitate ease of use. Data formats are also explained in each Task's Help tool and

section in this manual.

Fetch Timing

Fetching data can occur at different time intervals as defined here:

- **Fetch Data:** Clicking the Fetch Data button fetches the data immediately.
- **Each Animation Step:** Selecting the Each Animation Step switch causes the data to be fetched upon each animation, which is at an interval of one second. If you use this button then go to the Data tool and select Animate.
- **At First Appearance:** Selecting the At First Appearance switch causes the data to be fetched when the task is made visible. Tasks are only made visible when switching tasks so this operation does not occur too frequently.

Fetch Operation

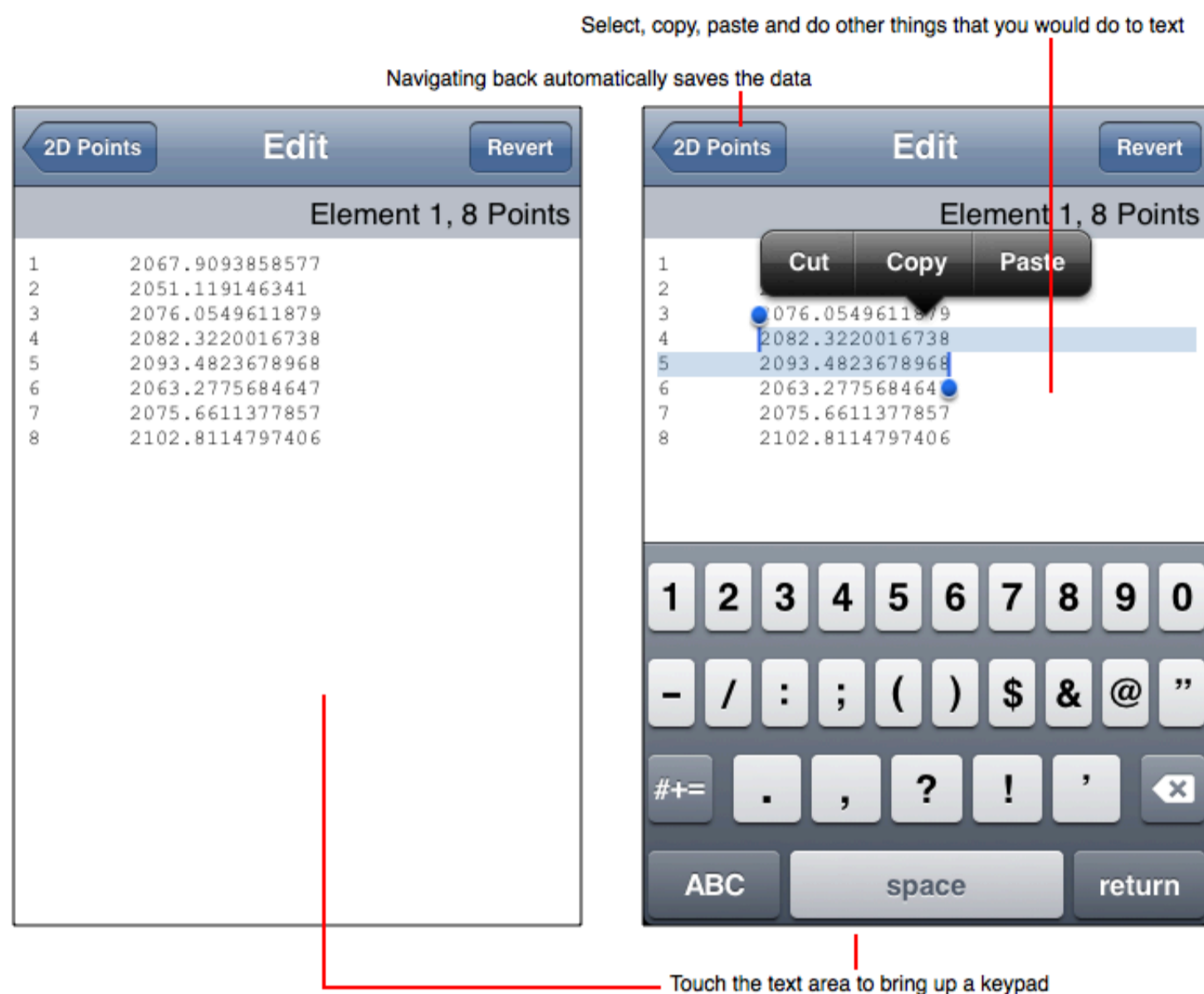
In a nutshell, when a fetch occurs the data is retrieved, entered into the task and then displayed. Usually the fetch appears to be atomic, that is: it seems to be a single operation. However, fetching is asynchronous and non-atomic. You may notice this fact if the fetch takes a long time. Also, for tasks that have multiple columns the fetch is on a per-column basis and the fetch retrieves each column one at a time until all columns are fetched and then updates the task with fetched data. Multiple column fetches are done in parallel and asynchronously and each column fetch rendezvous to amalgamate the fetch into a single result.

© Copyright 1993-2012 by [Vimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

Graph > Tools > Edit Data

Use the Edit Data tool (available in [Developer Mode](#) only) to literally type in data just like any text on the iPhone or iPad. See [Making A Line Graph](#) for a tutorial on entering data this way. The figure below diagrams the Edit Data tool.

To navigate to the Edit Data tool touch the Edit Data row on any of the tasks. If that row has a text field then first touch the text field to bring up the keypad and then enter the element number to edit. That number is an integer from 1 to 20 and in the case of a bar graph can also be 0 or the letter I for labels.



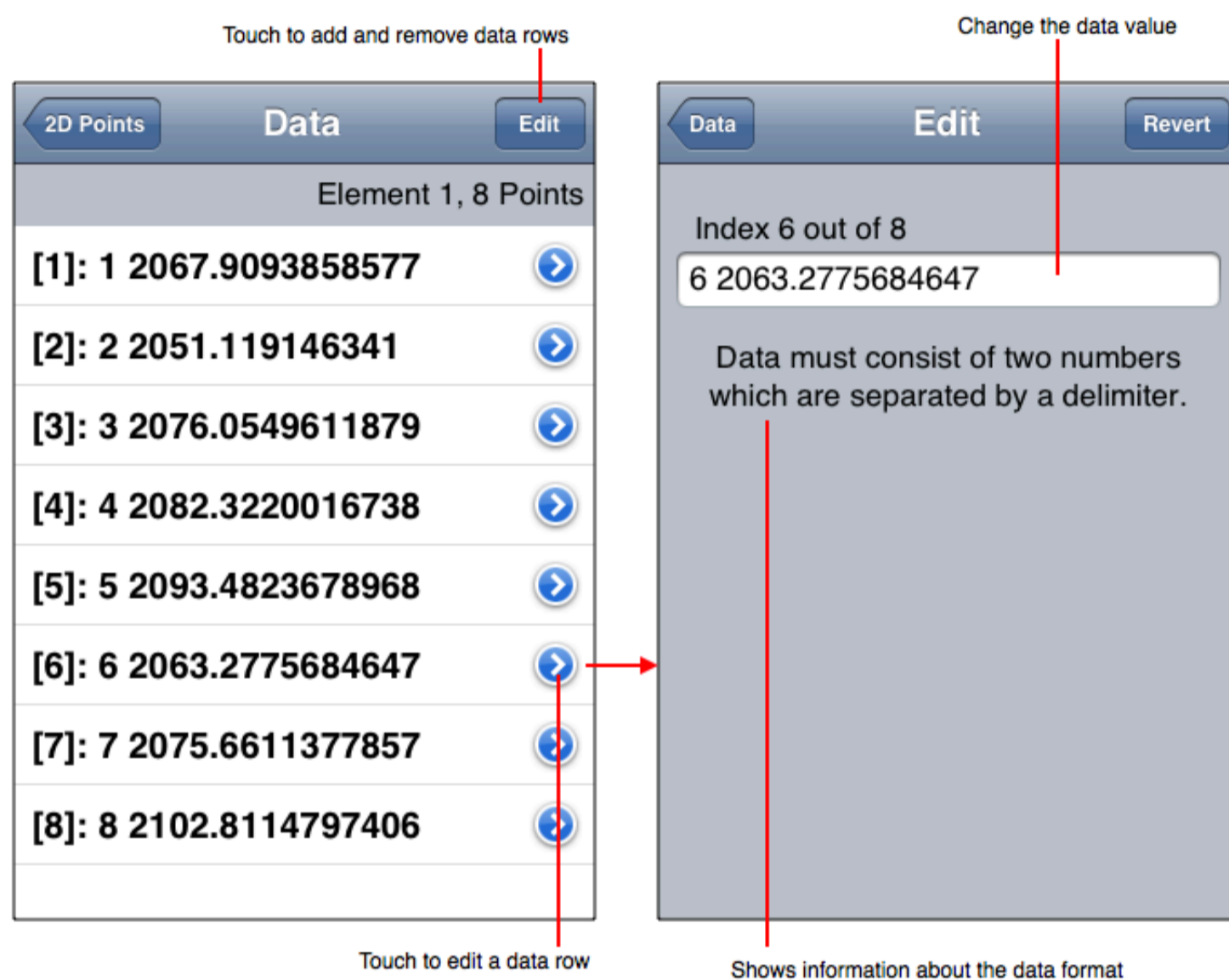
Notice that the [Fetch Data](#) and [Show Data](#) tools are also used to enter data.

Graph > Tools > Show Data

Use the Show Data tool (available in [Developer Mode](#) only) to:

- Add and Remove data elements (rows).
- Edit data rows.
- Show the data, including the number of data elements. This is particularly important for large data sets as this method of showing data is much more efficient than the [Edit Data](#) tool.

The figure below diagrams the Show Data tool.



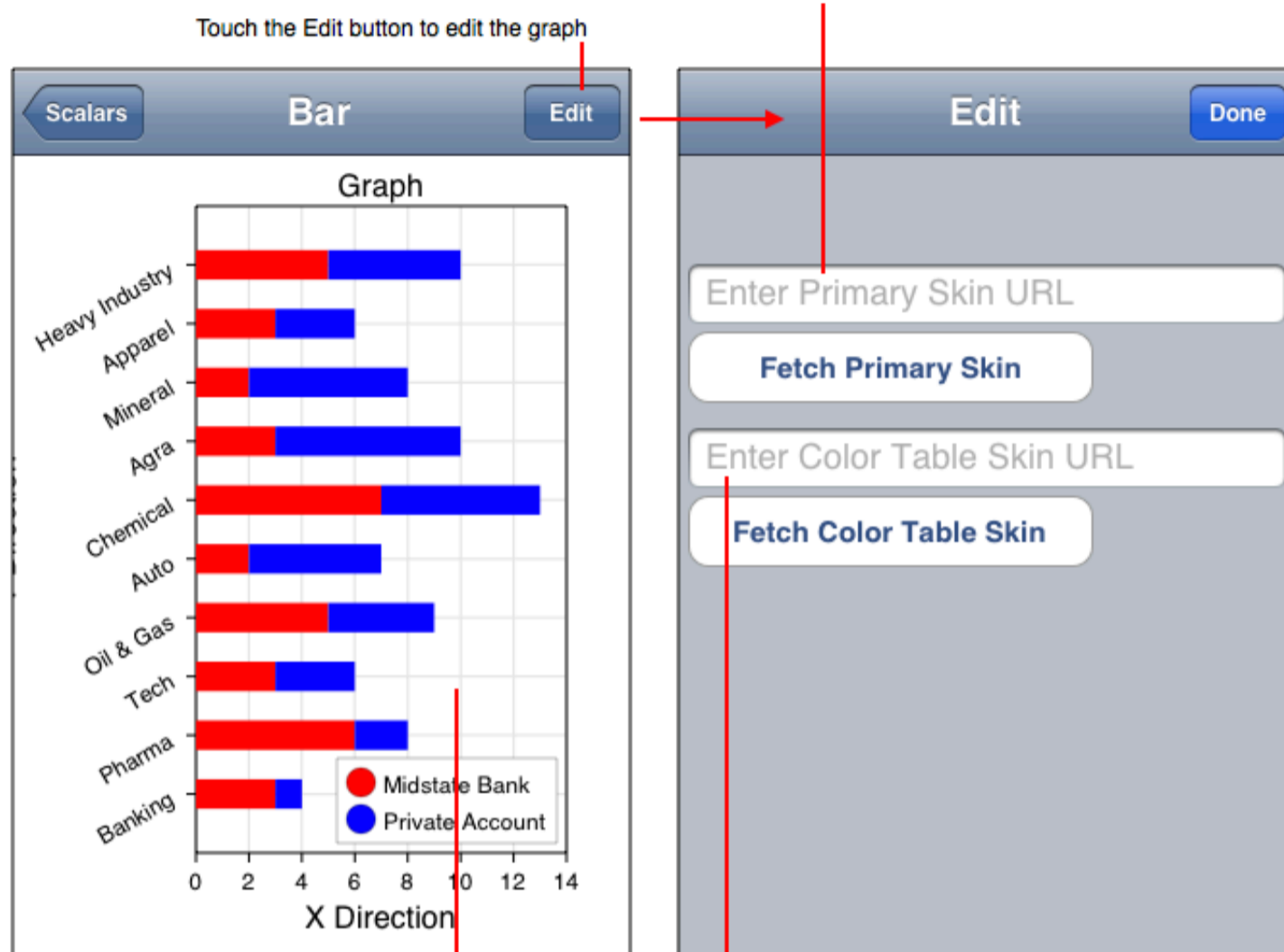
Notice that the [Fetch Data](#) and [Edit Data](#) tools can also be used to enter data.

Graph > Tools > Edit Graph

Each graph is defined by a specialized Vwidget Builder document called a skin. By having a graph use your own skin you can make the graph have distinctive qualities that are otherwise unavailable with the limited controls of the task. Making a skin is described in the Vwidget Builder and Vwidget Code manuals as well as the [Skins](#) section. This section explains how to get a skin into a task for use by one of its graphs.

Click the Edit button at the top right of each graph view. That navigates to a view tool like that shown below.

Editing a graph consists of primarily fetching a primary skin.
The primary skin defines most of the graph's graphical attributes (color, fonts, etc.).
Enter the URL to the location of the skin file and then touch Fetch Primary Skin



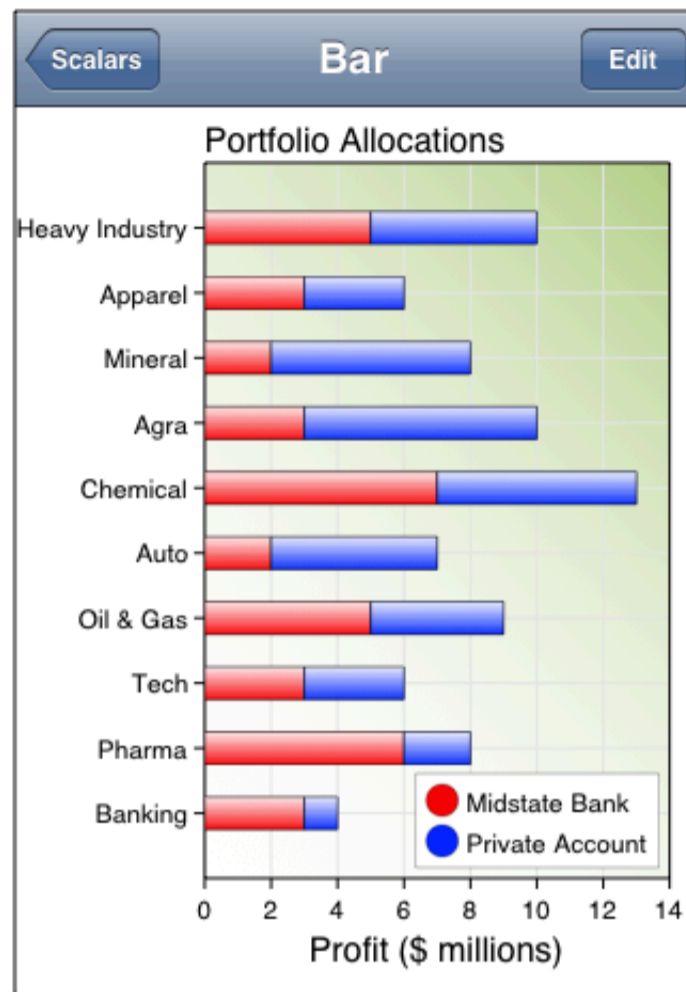
Example task representation (a bar graph)

Color tables, such as line color mapping, are defined by a skin too.

The Primary Skin defines the graph attributes while the Color Table Skin defines colors associated with distinct elements of the data graphics, such as curves. To import a skin type its location as a URL into the corresponding field and then click the respective Fetch button, then click Done. The graph of the current task will be updated to reflect the new skin.

By way of example, the bar chart shown above was altered using a skin and the result is shown in the following figure.

Result after applying a simple skin



As you can see, a skin can add many graphical attributes that make the graph more appealing.

URL Types

The URL that can be used is defined here:

- Web: A web URL begins with "http://" and the rest of the URL is the path to the skin document, such as: "http://www.vvi.com/skins/distinctive-line.book" (without the quotes). Notice that the path extension is "book". This is the same URL that you would type into a web browser to download the skin. You can use the skin directly in this manner, or first download it and then use the File type URL. Either way, the skin file must be a resource of the web server that the URL points to, either as a static file or a dynamic URL that retrieves the skin file bytes from other places.

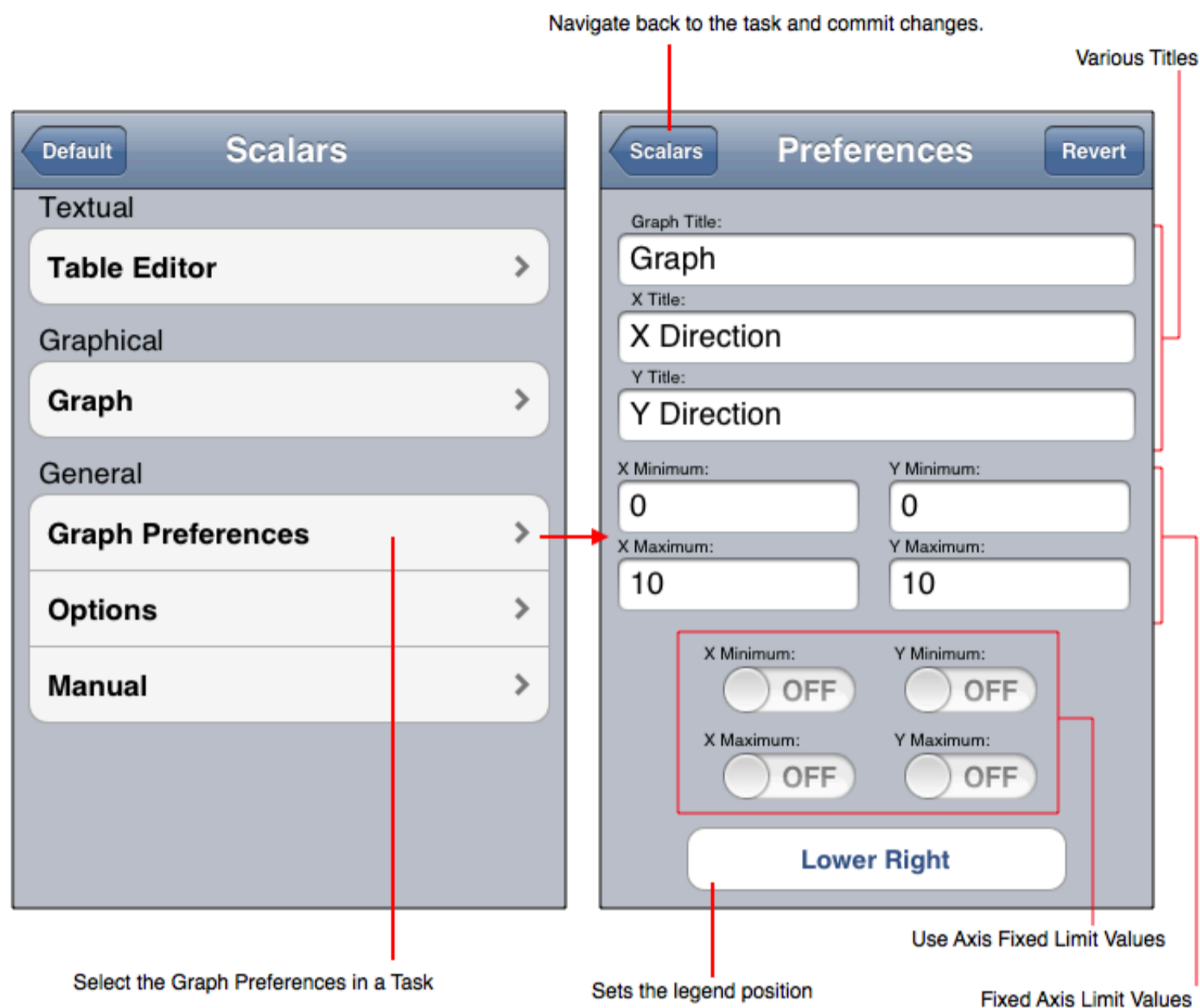
Constructing Skins

A Skin file is simply a Vwidget Builder document that has specialized graphical elements. To facilitate the retrieval of the document it should be exported as a Skin type using the menu item Vwidget Builder > File > Export To ... and choosing the Skin type on the resulting panel. That exported file must have a book extension. The export converts the Vwidget Builder document, which is a bundle of resources, into a compressed binary flat archive which can be efficiently transferred.

The hard part in making a skin is understanding the "specialized graphical elements" on the document. For that consult the [Skins](#) section.

Graph > Tools > Preferences

As diagrammed in the figure below, each [Task](#) (save the [Maps Task](#)) has a Preferences tool. That tool sets some basic attributes of the task's representations.



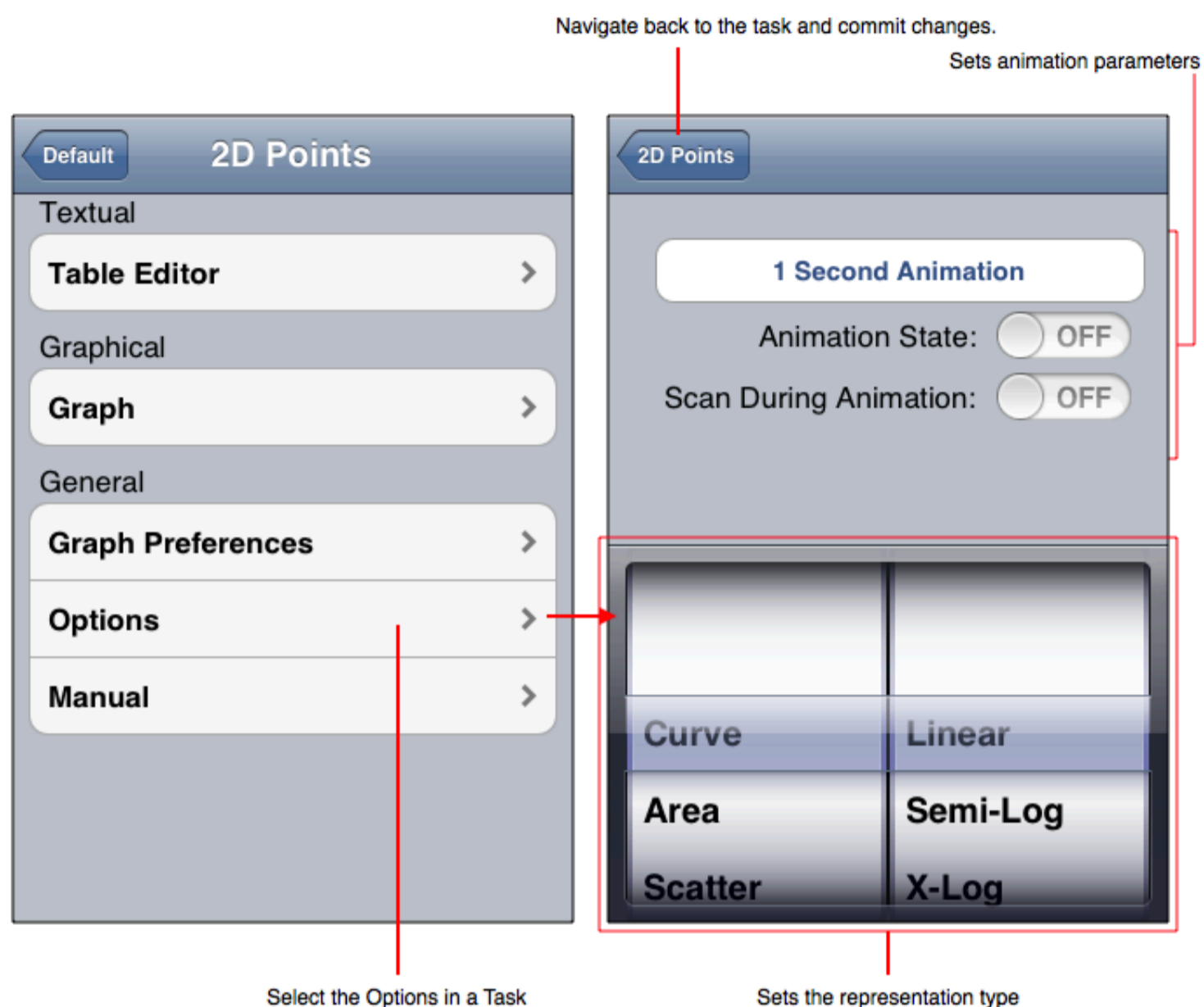
The preference tool for each task can be different but generally have the following components.

- **Titles:** Sets the graph and axes titles.
- **Limit Values:** Sets the respective fixed limit used on the axis.
- **Limit Values State:** Determines, for each limit value, if that value is used or if the autoscaler defines the respective limit. As a consequence if all limit values states are on then no autoscaler is used.
- **Legend:** Sets the legend position. To set legend text see the support [Legend](#) section.

It is important to note that the Preferences are intentionally minimalistic. They are intended only to provide essential attributes. For access to greater features see the [Skins](#) section.

Graph > Tools > Options

As diagrammed in the figure below, each [Task](#) has an Options tool. That tool sets the task's representation and animation attributes.



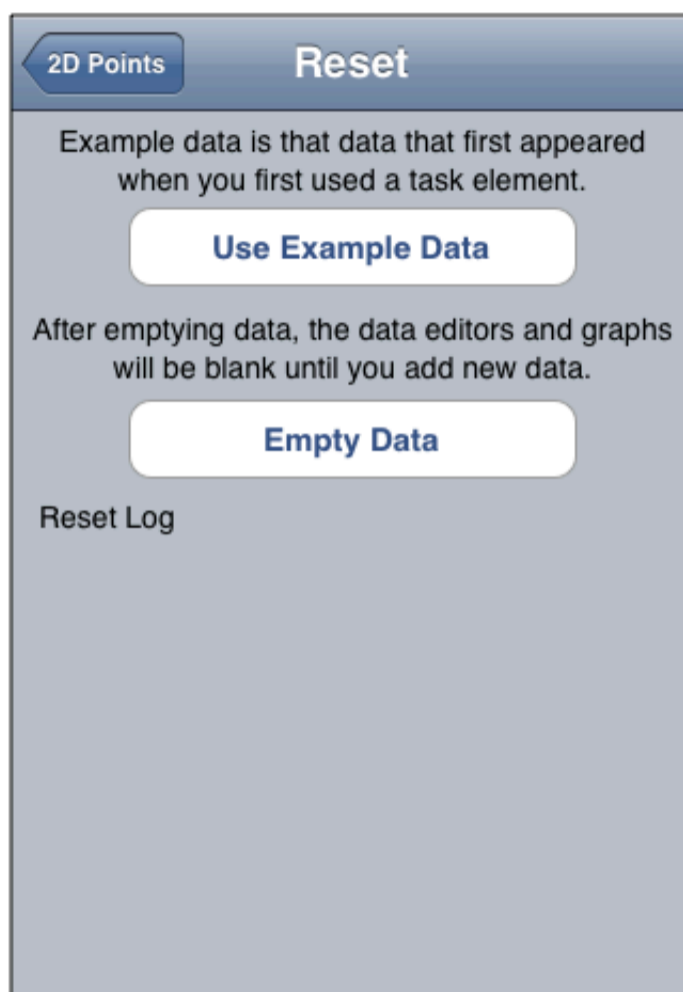
The Options tool for each task can be different but generally have the following components.

- **Animation:** Sets the animation time sample increment and states. The [Fetch Data](#) tool also sets attributes that involve animation.
- **Representation:** Sets the graphical representation of the data. This is called an attribute based setting of the representation.

In normal mode (see: [App Preferences](#)) the representation sets the graph type. In developer mode the representation sets the graph type for only the "Options Graph" as the other graphical representations are navigation based.

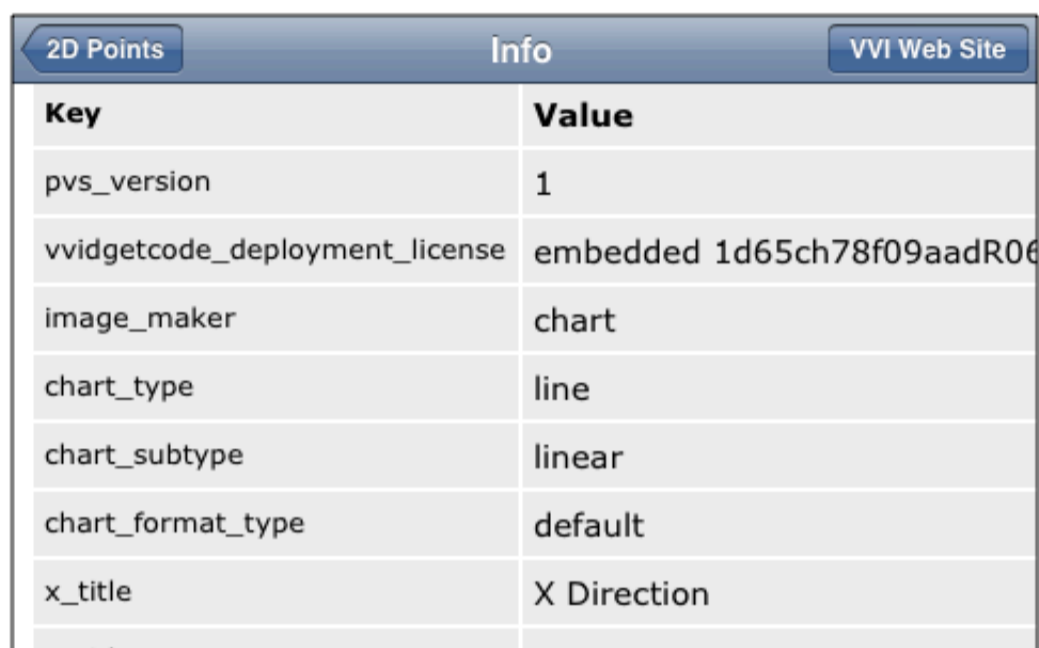
Graph > Tools > Reset

The Reset tool (available in [Developer Mode](#) only) clears data or sets it to example data. Its view is diagramed below.



Graph > Tools > Info

The Info tool (available in [Developer Mode](#) only) shows the key value pairs (dictionary) that is used to make the graph of the task. This information is very useful for constructing [XML Fetch](#) content and for developing applications with Vwidget Code. The figure below shows the info panel.

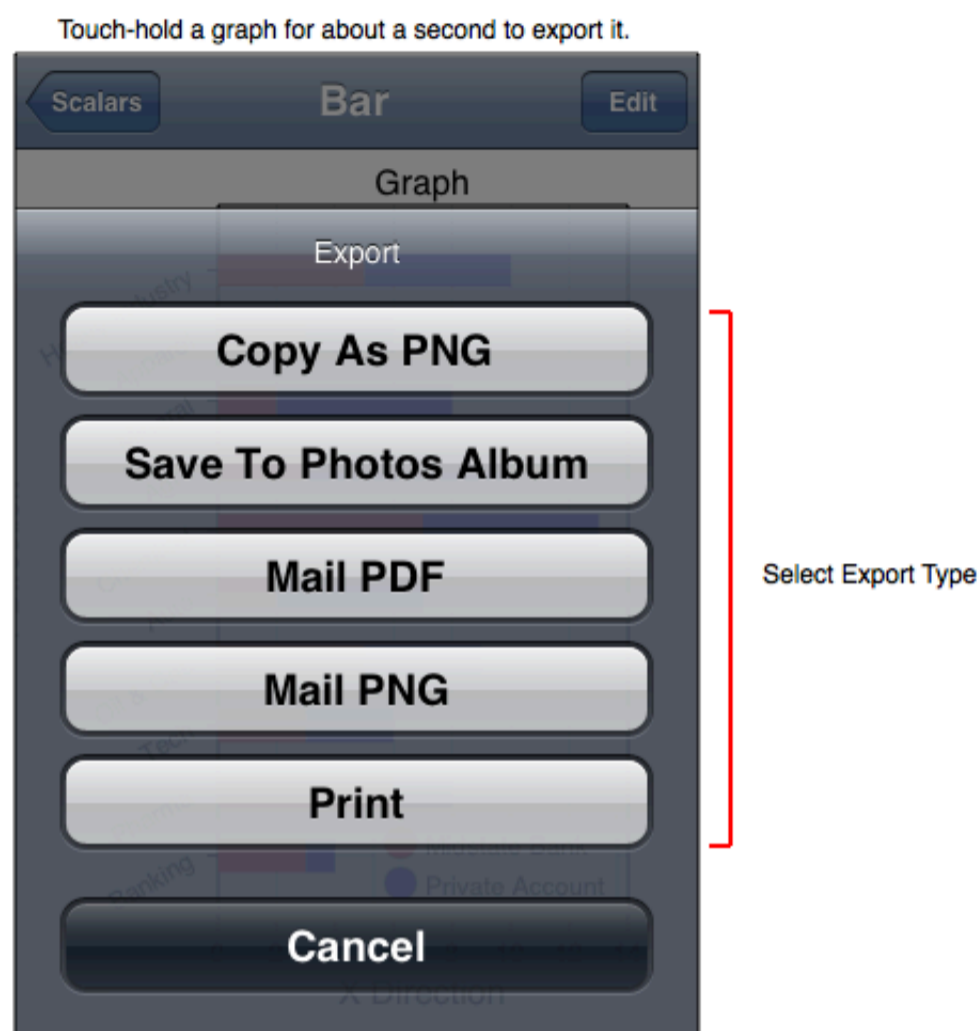


The screenshot shows the 'Info' panel for a '2D Points' graph. The panel has a header with '2D Points' on the left, 'Info' in the center, and 'VVI Web Site' on the right. Below the header is a table with two columns: 'Key' and 'Value'.

Key	Value
pvs_version	1
vwidgetcode_deployment_license	embedded 1d65ch78f09aadR06
image_maker	chart
chart_type	line
chart_subtype	linear
chart_format_type	default
x_title	X Direction
y_title	Y Direction

Graph > Tools > Export

When you find a graph you like then touch-hold it for over two seconds and then select an export method in the resulting sheet as shown below.

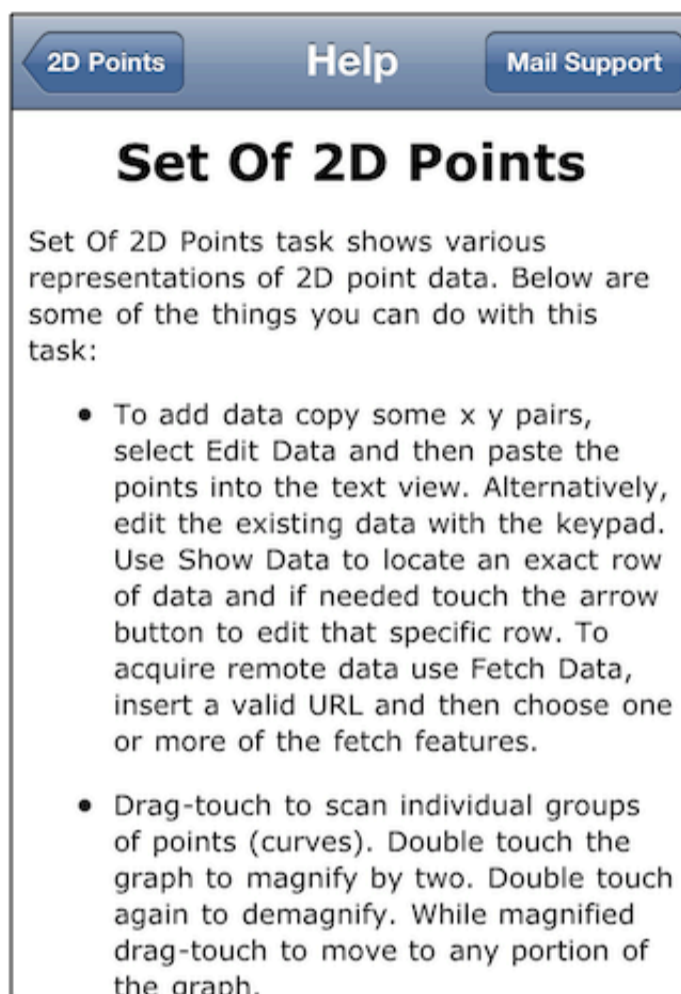


Each export type is described in the following table.

Export Type	Description
Copy As PNG	A PNG image of the graph is stored on the pasteboard. Once saved you can then go to another application that supports the pasteboard and paste the image to it.
Save To Photos Album	This will save a PNG representation to the Photos Album. From there you can apply many techniques to transfer the images to other systems. This is particularly good for bulk picture taking of graphs and then subsequent import into applications such as iPhoto.
Mail PDF	Mails a PDF representation as an attachment. After selecting this option a mail interface is brought forward so that you can fill in the email address to send to and then send the mail.
Mail PNG	Mails a PNG representation as an attachment. After selecting this option a mail interface is brought forward so that you can fill in the email address to send to and then send the mail.
Print	Prints the graph. The print parameters are mostly automatic. In general, the graph is resized to the printed page and then the standard print interface is presented so that you can select the printer to print to. Rotate your device to set the landscape v.s. portrait orientation.

[Graph](#) > [Tools](#) > [Help](#)

The help tool shows some basic help for each task. It is intended to be the primary means of help for Graph, however this manual is also needed to provide overviews, tutorials and reading material that is available without using Graph. The figure below shows the help tool view.



[Graph](#) > Tutorials

The following is a brief list of tutorial sections:

Tutorials	Description
Making A Line Graph	A step by step instruction for making a line graph.
Making A Bar Chart	A step by step instruction for making a bar chart.
Representations	Describes Representations once again.
XML Fetch	Describes XML Fetching and schema.
Making A Map	Describes How to make a map for the Maps task.

If you have a question that is not explained in this manual please contact support@vvi.com so that we may answer your question and update this manual as needed.

© Copyright 1993-2012 by [VViimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

Graph > Tutorials > Making A Line Graph

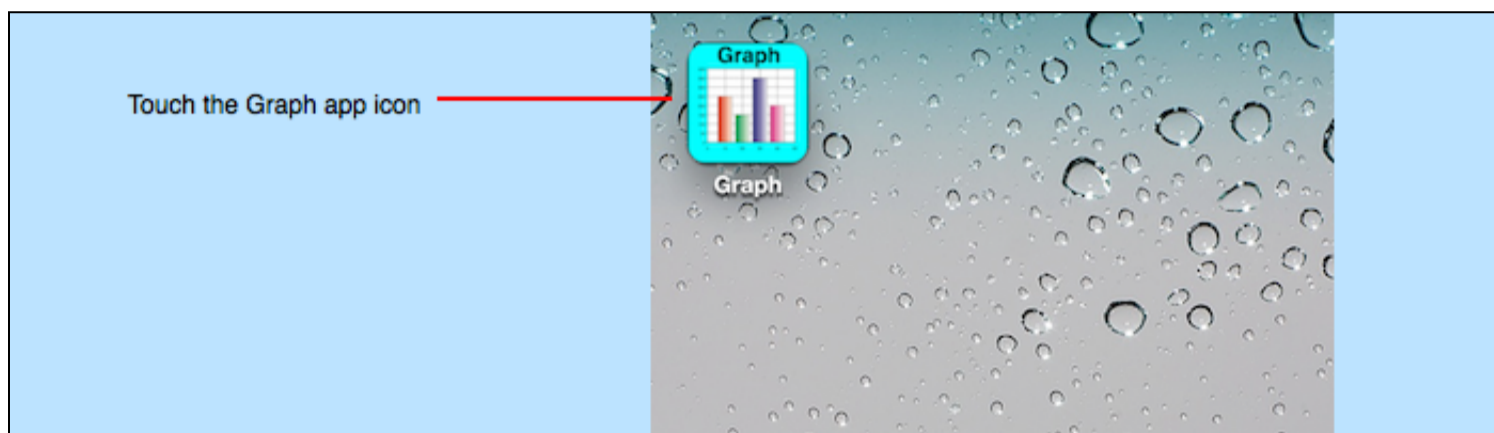
Note: this describes data entry with the Text Edit tool only available in [Developer Mode](#). For normal entry see the [Table Editor](#) section.

The support section [Enter Data](#) gave some pointers on how to enter data but was not a step-by-step explicit instruction on that subject. For that type of explanation, lets accurately answer this question:

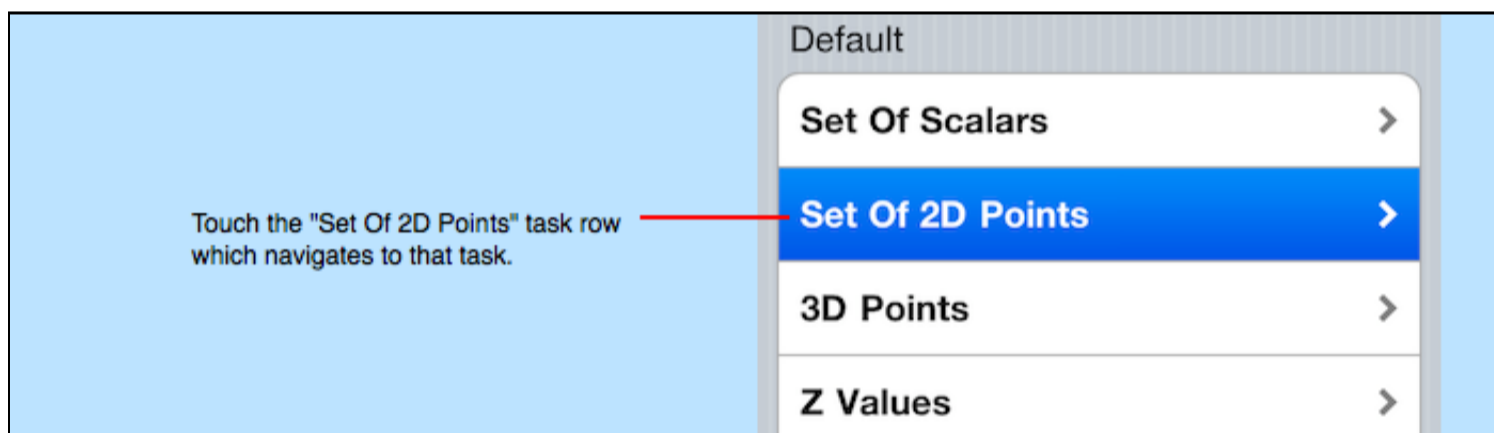
"Dear Tech Support, This program is entirely too complicated for a 78 year old man. All I want to do is entire my own x and y data and get a bar graph or curve. How can I enter my data? Where can I enter my data? My data consists of a range of years (10) and a Dollar amount. Please tell me in simple instructions."

The following are the instructions for a line graph. For a bar chart see the tutorial [Making A Bar Chart](#). The figures are on an iPhone, but are equally applicable to an iPad.

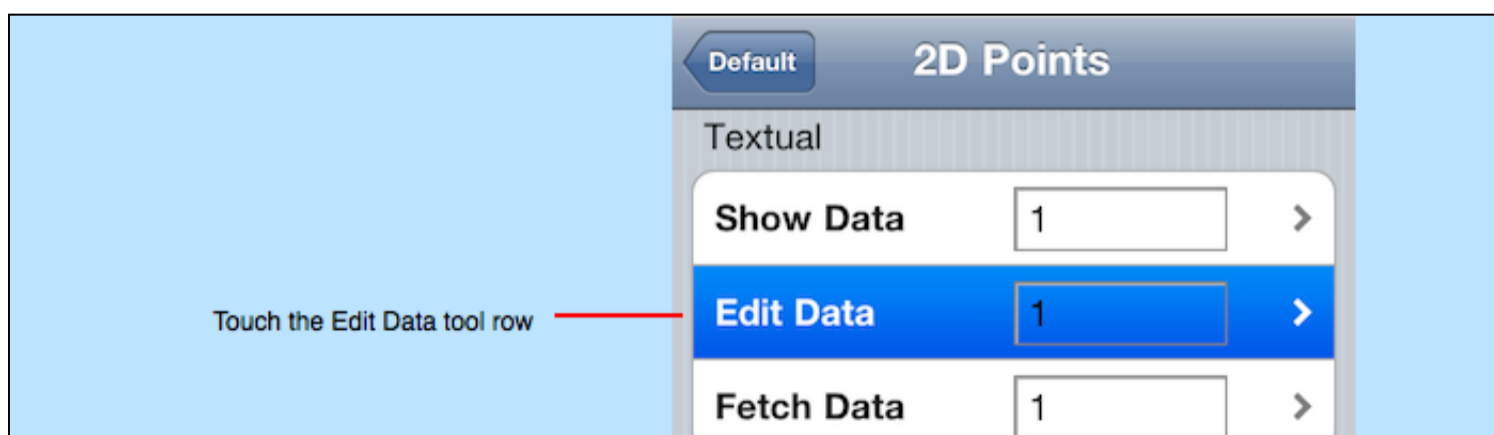
1. On your iPad, touch the Graph application icon to launch it.



2. Make sure the Graph application is on the [Home](#) page. To do this, touch the upper left navigation button until no buttons appear there.
3. On the home page touch the "Set Of 2D Points" task to navigate to the [Set Of 2D Points](#) task.

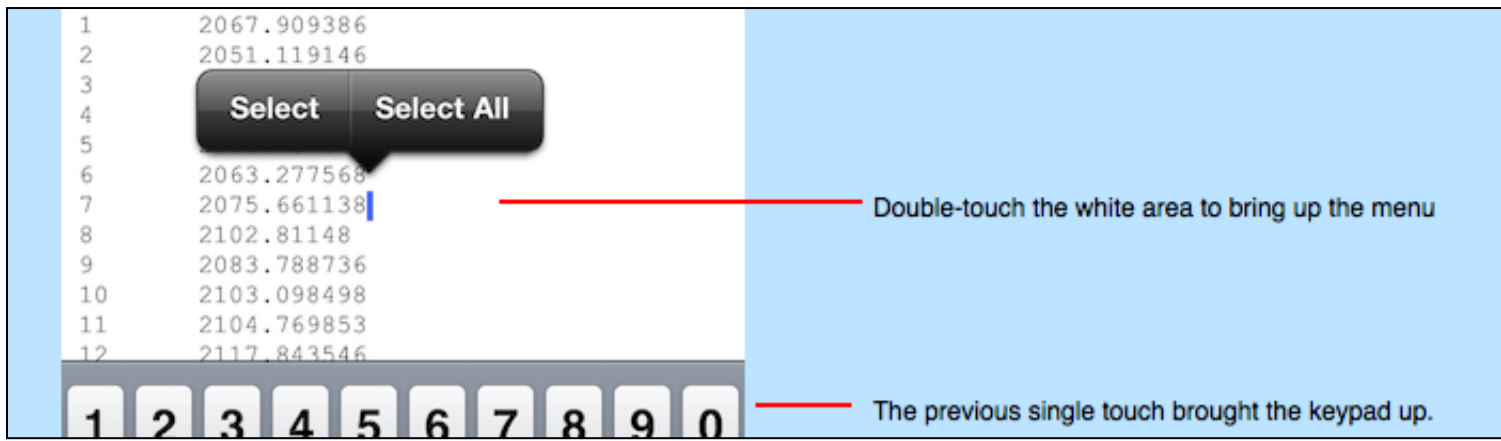


4. On the resulting page, touch "Edit Data" to navigate to the [Edit Data](#) tool.



5. You will see a bunch of numbers. Touch any number once. That brings up the keypad. Double-touch to the right of any number (in a blank area). That brings up a menu of options.

6. Touch "Select All" from the menu.

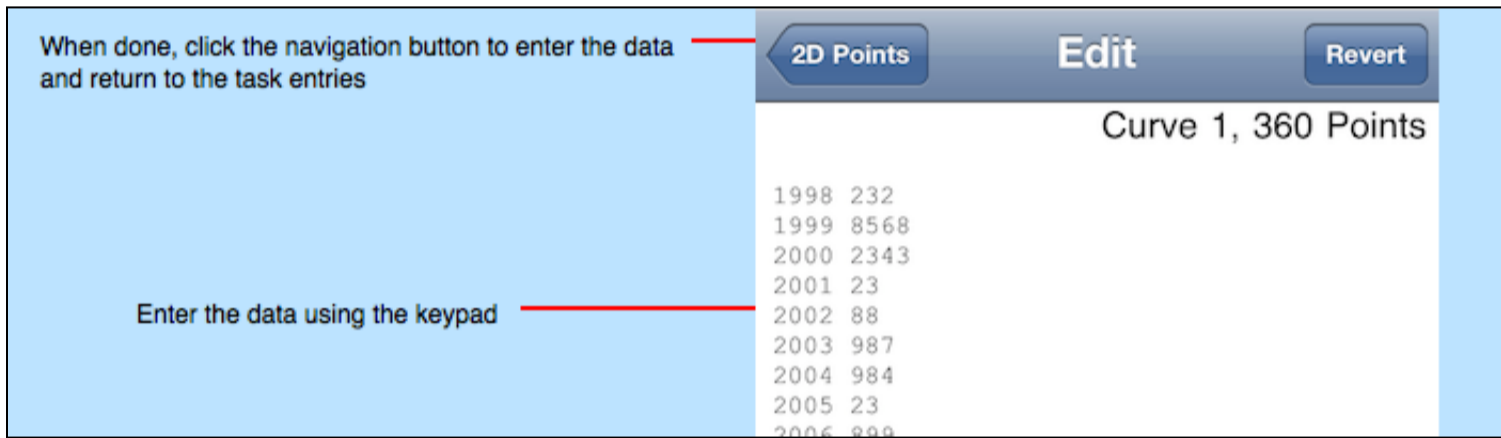


- 7. Touch the Delete key on the keypad (it looks like a box with a arrow on the left and an X in the middle).
- 8. Enter your numbers using the keypad. The numbers are consecutive x y pairs of points, one point per line. In your case, enter 10 lines of data where each line consists of two numbers where the first number is the year and the second number is the dollar value (without the dollar symbol and without commas) for example:

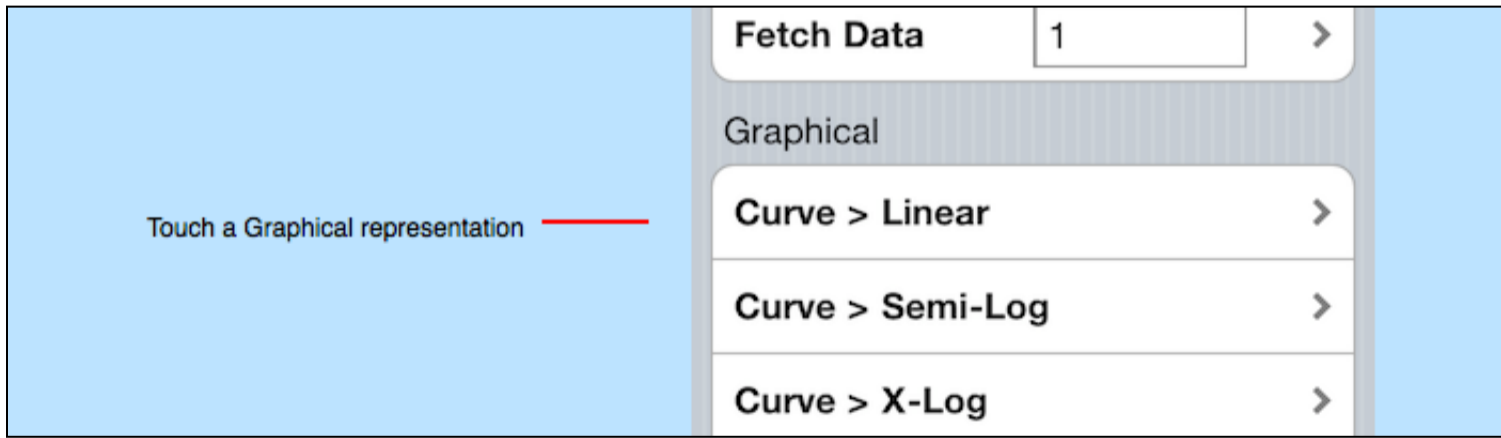
```

1998 232
1999 8568
2000 2343
2001 23
2002 88
2003 987
2004 984
2005 23
2006 899
2007 444
    
```

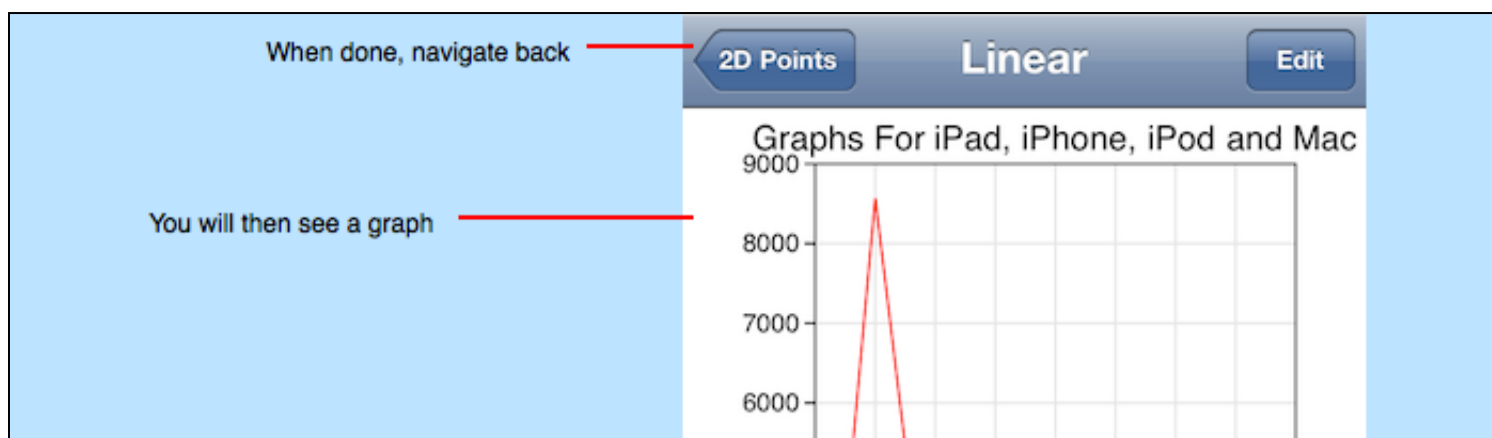
The result looks like the following:



- 9. Touch the "2D Points" button on the upper left of the page. That will enter your data and also return to the "Set Of 2D Points" task page.
- 10. Touch the "Curve > Linear" table row on that page.



- 11. That will navigate to a line graph representation of your data on a linear coordinate system.



12. If you want to see your data on a different coordinate system then touch the "2D Points" button on the upper left of the page and then touch another table row in the Graphical section. For example "Curve > Semi-Log" to see your data on a semi-log graph, or "Area > Linear" to see your data as an area graph.

13. If you want to place another curve on the graph then enter 2 in the text field in the "Edit Data" row of the task and then click the arrow to the right of it and enter the points for the 2nd curve. Then navigate back to the task page and view the graph. The graph will now have 2 curves on it. Repeat for lines 3 through 20.

After experimenting in the Set Of 2D Points task you can proceed to the [Making A Bar Chart](#) tutorial to see the rest of the answer.

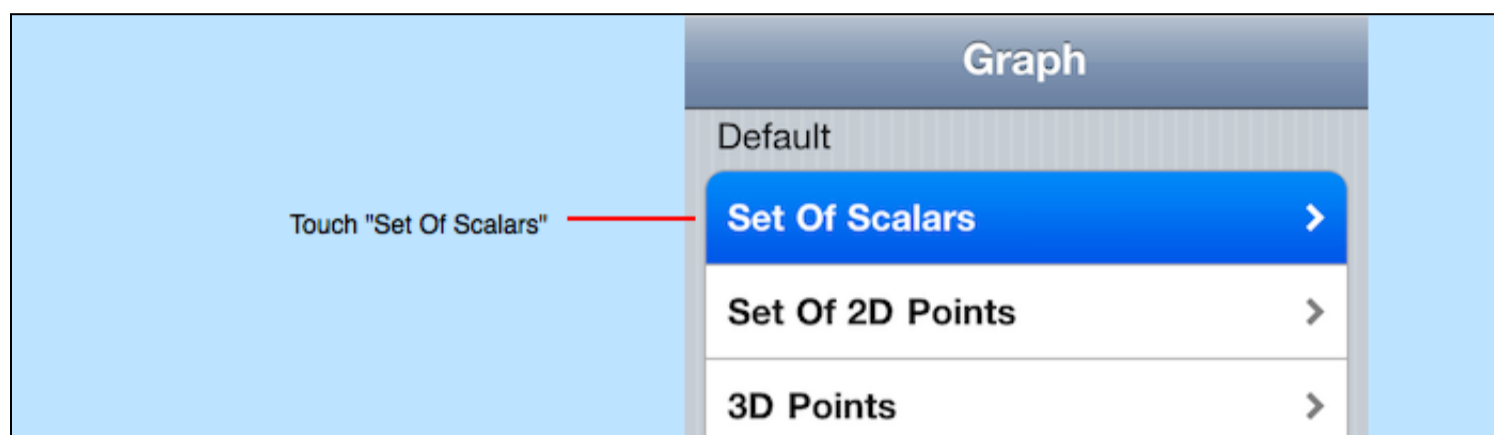
© Copyright 1993-2012 by [VVI](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

Graph > Tutorials > Making A Bar Chart

Note: this describes data entry with the Text Edit tool only available in [Developer Mode](#). For normal entry see the [Table Editor](#) section.

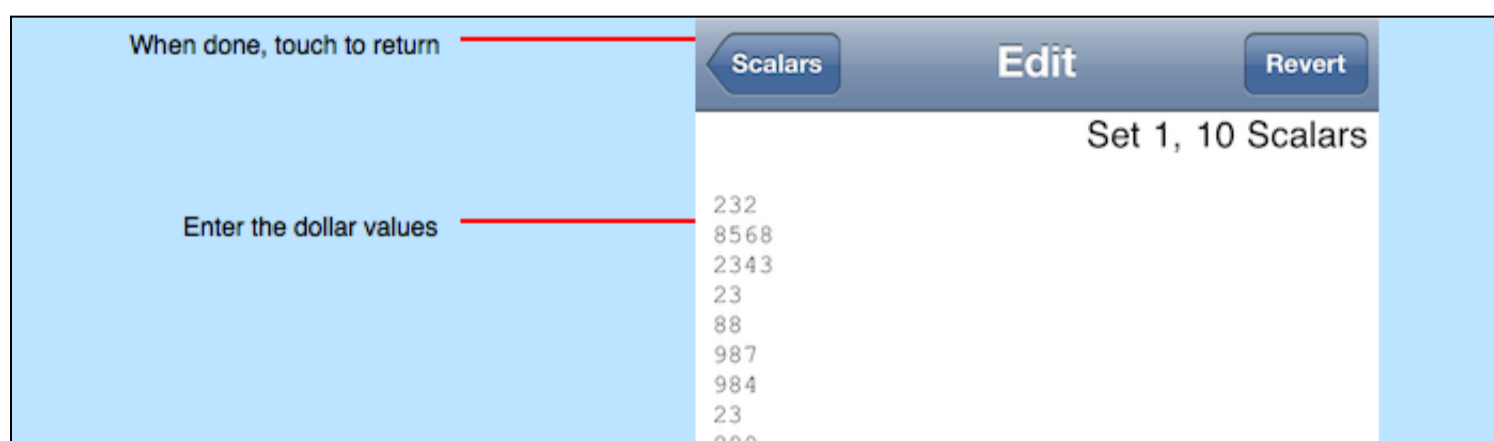
The following is a continuation of the question posed in [Making A Line Graph](#). This tutorial answers the second part of the question, how to make a bar graph from the data. The following are step-by-step instructions.

1. On your iPad, touch the Graph application icon to launch it.
2. Make sure the Graph application is on the [Home](#) page. To do this, touch the upper left navigation button until no buttons appear there.
3. On the home page touch the "Set Of Scalars" task to navigate to the [Set Of Scalars](#) task.

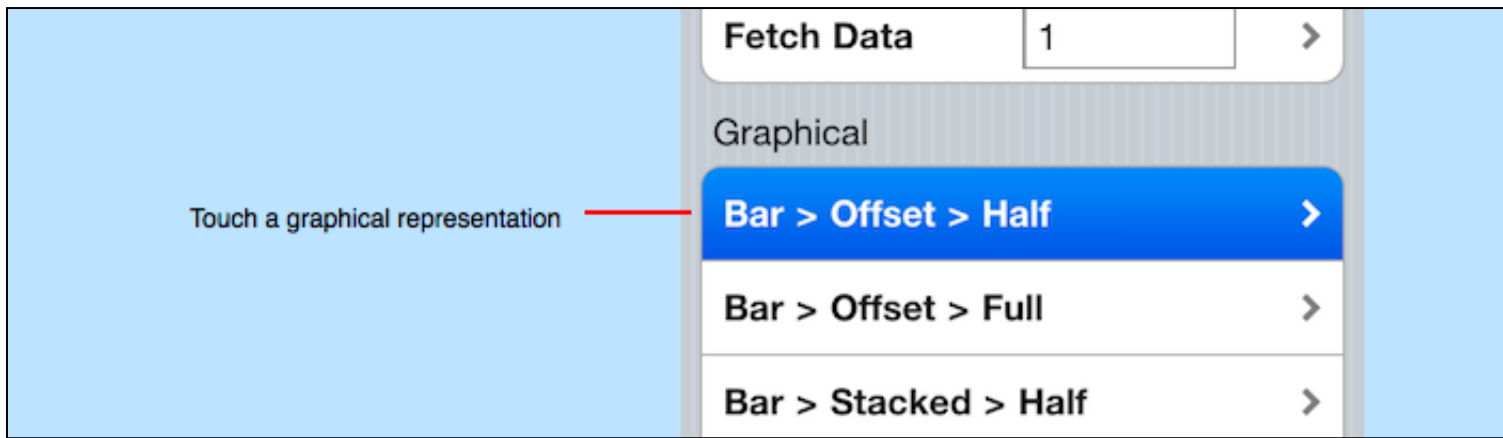


4. On the resulting page, touch "Edit Data" to navigate to the [Edit Data](#) tool.
5. You will see a bunch of numbers. Touch any number once. That brings up the keypad. Double-touch to the right of any number (in a blank area). That brings up a menu of options.
6. Touch "Select All" from the menu.
7. Touch the Delete key on the keypad (it looks like a box with a arrow on the left and an X in the middle).
8. Enter your numbers using the keypad. The numbers are consecutive numbers, one number per line, which represents the height of a bar. In your case, enter 10 lines of data where each line consists of one number which is the dollar value (without the dollar symbol and without commas) for example:

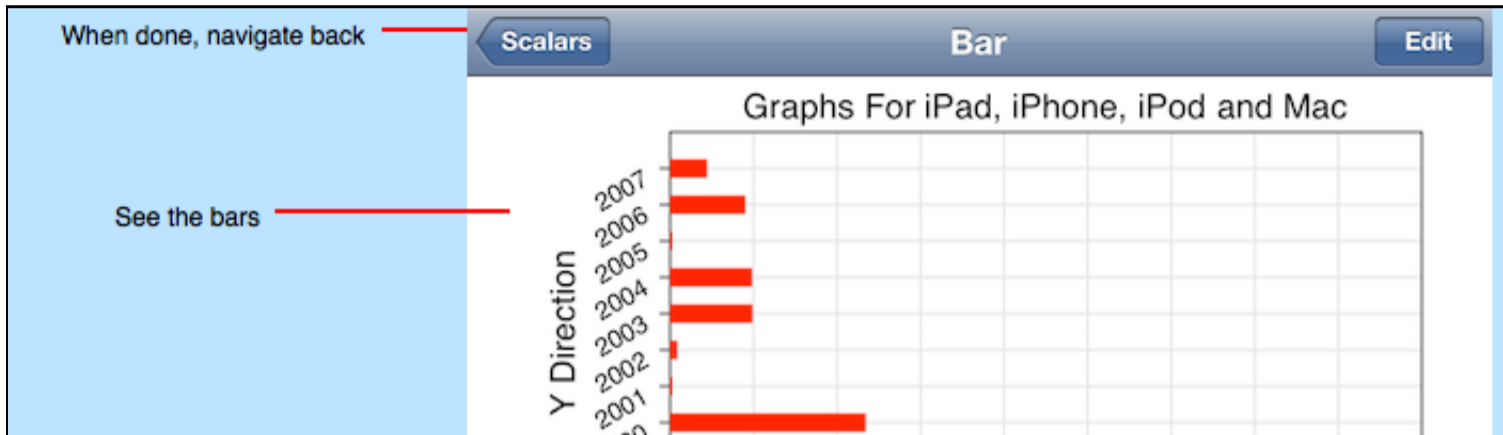
232
8568
2343
23
88
987
984
23
899
444



9. Touch the "Scalars" button on the upper left of the page. That will enter your data and also return to the "Set Of Scalars" task page.
10. Touch the "Bar > Offset > Half" table row on that page.

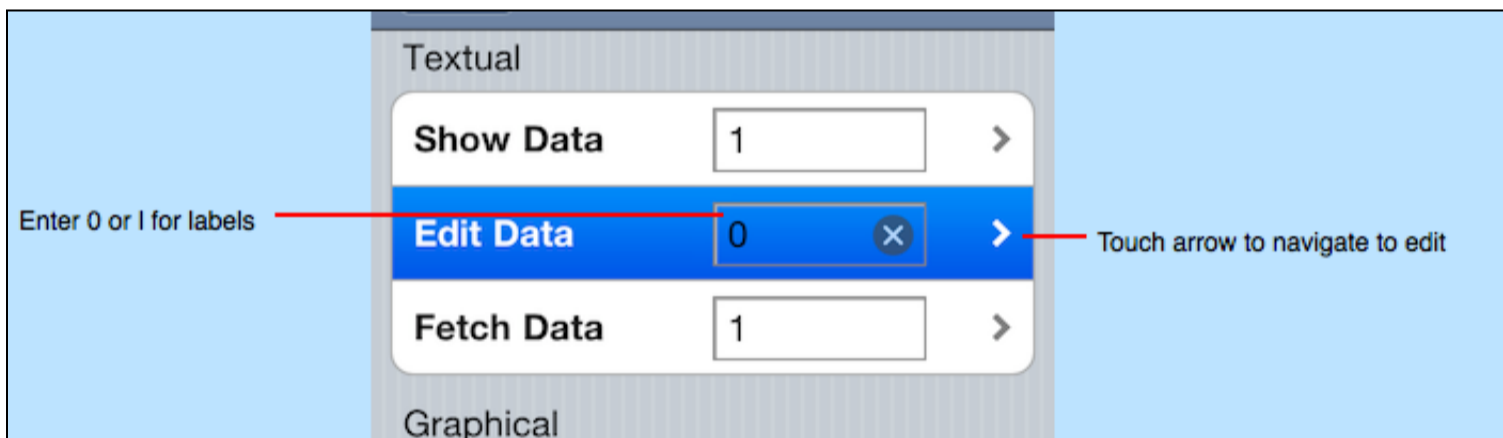


11. That will navigate to a bar chart representation of your data on a linear coordinate system.



12. If you want to see your data on a different coordinate system then touch the "Scalars" button on the upper left of the page and then touch another table row in the Graphical section. For example "Pie Chart" to see your data in pie chart form.

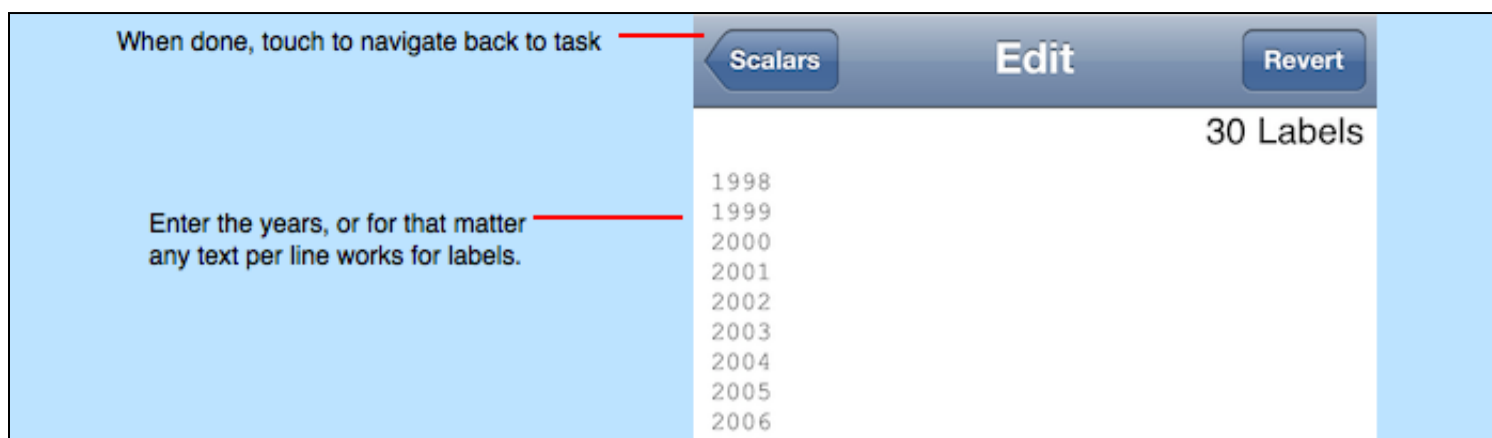
13. Notice how the bars align to integers on the axis. Now you need to figure out how to get the axis to display years. For that, navigate to the "Set Of Scalars" task page by clicking the "Scalars" button on the upper left of the page. Then in the "Edit Data" row touch the text field that currently has the numeral 1 in it. Use the keypad to change that to the letter I or the number 0 and then touch the "Edit Data" row or the arrow to the right in that row. You will now be able to edit the axis labels.



14. As before, touch any label once, double-touch to the right of any label, touch "Select All" from the menu, touch the Delete key on the keypad, and then enter the years like this:

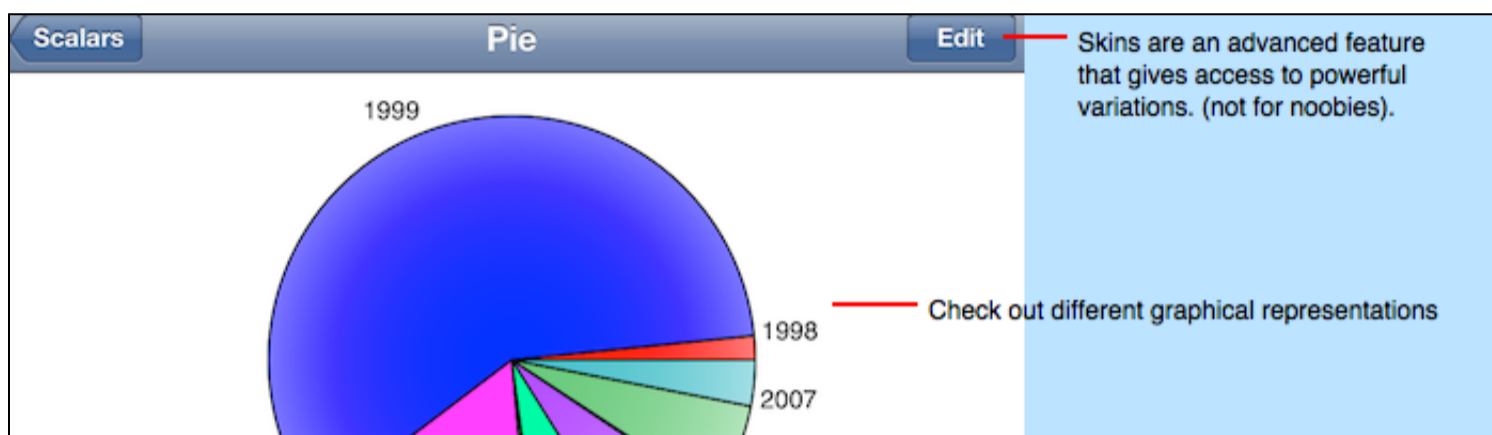
- 1998
- 1999
- 2000
- 2001
- 2002
- 2003
- 2004
- 2005
- 2006
- 2007

The result is shown below. Note that you can enter any label you wish, not just numbers. Each label is entered as one line of text.



15. Touch the "Scalars" button on the upper left of the page. That will enter your labels and also return to the "Set Of Scalars" task page.

16. Then touch one of the graphical representations and notice how with a bar chart the y-axis has the year labels, with a column chart the x-axis has the year labels and with a pie chart the labels are placed near each wedge. The pie chart only looks right in landscape orientation.



17. If you want to add another group of bars on the graph then enter 2 in the text field in the "Edit Data" row of the task and then click the arrow to the right of it and enter the scalars for the 2nd group of bars. Then navigate back to the task page and view the graph. The graph will now display two groups of bars either in offset or stacked format. Repeat for bar groups 3 through 20.

Conclusion

The key to using the navigation system effectively is to recognize these properties:

- The home page has a list of tasks and each task is defined by the data type it uses. Touch the task row appropriate to your task.
- Each task page has a list of textual edit rows that you can touch to navigate to view and edit text formatted data and a list of graphical rows that show the data entered. You have to navigate forward to the textual page, then back to the task and then forward again to a graphical page.

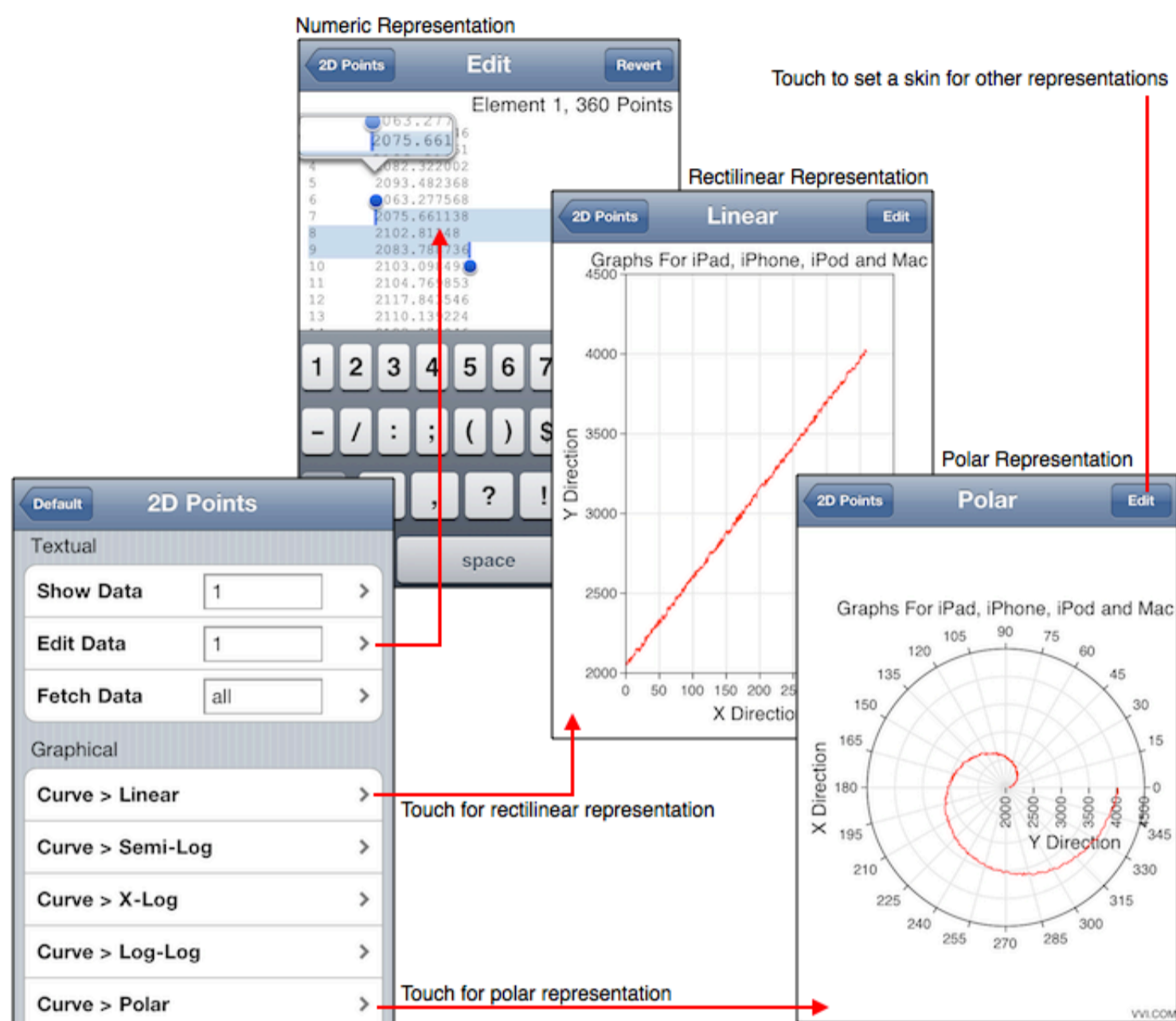
This tutorial explained how to enter data directly on your iPad and iPhone and how to use the default layouts. Using other tools and skins permit a larger variation and more effective data importing.

Graph > Tutorials > Representations

As you learned in the [Main Idea](#) section the user interface is totally dedicated to one core concept which is "task representations". It is so important that this section describes the process of navigating to representations for two tasks.

Set Of 2D Points

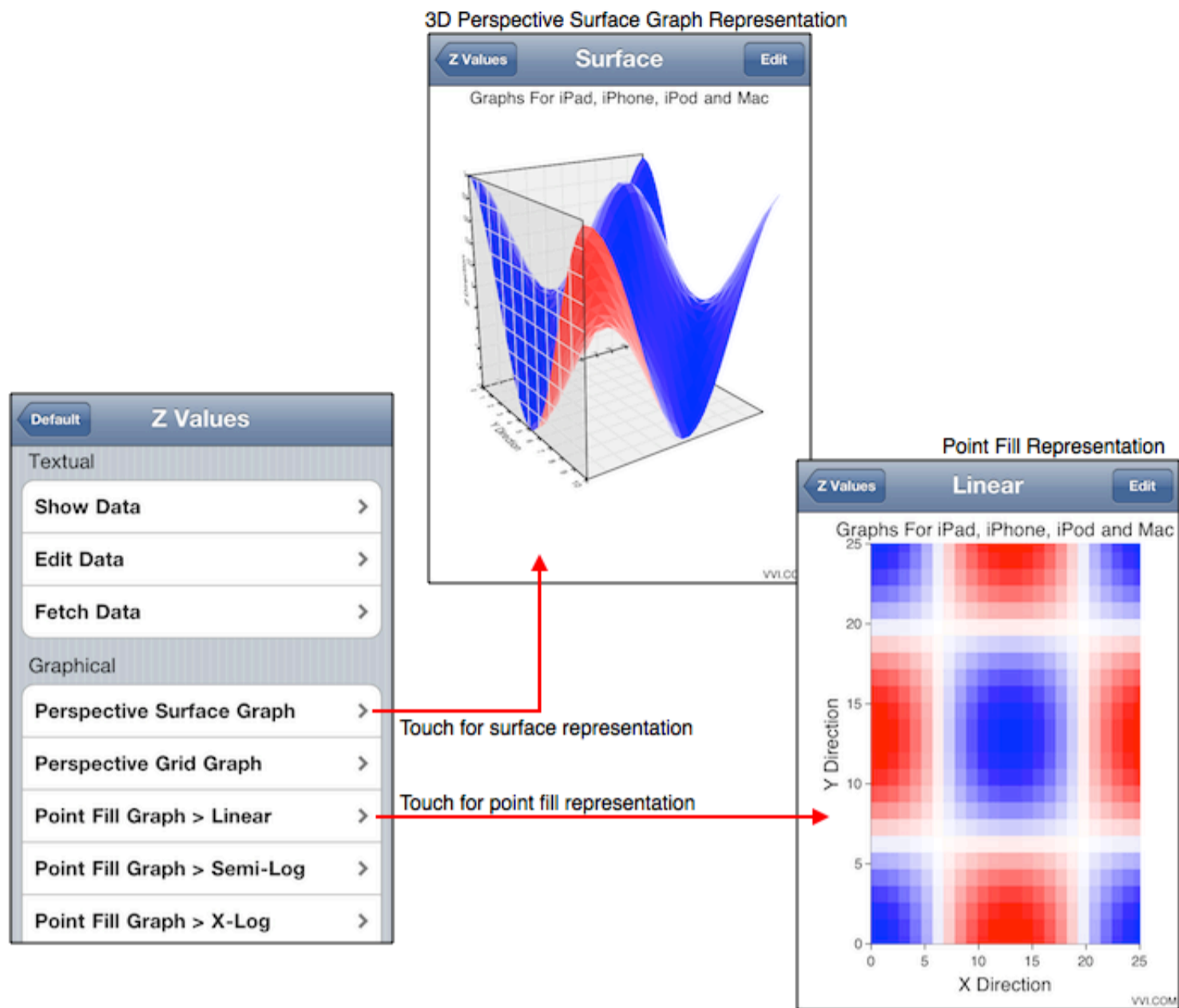
The following diagram shows some of the representations for the Set Of 2D Points task.



Essentially, once on the [Home](#) tool you navigate to a task, in this case the [Set Of 2D Points](#) task by selecting the "Set Of 2D Points" row on the home tool. Once on the Set Of 2D Points task page then, as you see in the diagram above, you then select a row for a given representation. Selecting the [Edit Data](#) row brings you to the textual representation of the data which in this case are 2D points. The textual representation are also used to edit the numeric data directly, as you can see by the diagram. If you, instead, select the "Curve > Linear" row then you navigate to a rectilinear line graph. If you select the "Curve > Polar" row then you navigate to a polar line graph. And so on and so on with all the rows on the task page, save the bottom grouping.

Z Values

The following diagram shows some of the representations for the Z Values task.



Navigate to this task from the [Home](#) tool by selecting the "Z Values" row on the home tool. Once on the Z Values task page then, as you see in the diagram above, you then select a row for a given representation.

Conclusion

With a little experimentation and common sense one should be able to quickly enter data for a task and then navigate to the various representations of that task to see a desired result. The idea of a task is that it transforms the data to the representations. As such, the data needs to be entered only once and the various representations can be had by selecting a row of the task.

© Copyright 1993-2012 by [VVI](#) imaging, Inc. (VVI); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

[Graph](#) > [Tutorials](#) > XML Fetch

The [Fetch Data](#) section shows how to fetch content from a URL. The [Info](#) section shows how to view the key value dictionary description of a graph. This Tutorial amalgamates those sections and describes how to fetch XML content which defines the task state.

Note: If you are viewing this on an iPhone, iPad or iPod touch then see the section below to find out how to copy the plist URLs to do the fetch.

Line Graph

The following is the XML needed to make a line graph (curves). Line graphs are made with the [Set Of 2D Points](#) task so first select that task from the [Home](#) page, enter "All" in the Fetch Data row and then touch the Fetch Data arrow to navigate to the [Fetch Data](#) tool. Then enter the URL to the XML content shown below. Then touch the "Fetch Data" button.

Available at this URL: [linegraph.plist](#)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>title</key>
<string>Set Of 2D Points XML Test</string>
<key>x_title</key>
<string>XML-X-Title</string>
<key>y_title</key>
<string>XML-Y-Title</string>
<key>data_3</key>
<string>1 15 2 16 3 17 4 18 5 5</string>
<key>data_2</key>
<string>1 2 2 26 3 2 4 28 5 2</string>
<key>data_1</key>
<string>1 31 2 36 3 32 4 33 5 41</string>
</dict>
</plist>
```

Once the fetch is confirmed to work then you may wish to turn on "Fetch During Animation" and "Fetch During Appearance", click "Done", touch the [Edit Graph](#) tool and touch the start acquisition button. The XML will be fetched and its results will be displayed every second. As such, when you change the content of the XML then the graph will change to reflect the changes to the XML. If the URL points to a SOA service then the XML it serves will be polled and displayed every second.

Now you have some XML and know how to fetch it, but perhaps you don't know what XML exactly is so lets hit some bullets on that issue:

- The type of XML shown above is called a "Property List". It is a pretty well documented format on the Apple system so I won't describe it too much. Note that if you install the developer toolset then there is a Property List Editor application that can help you view and edit property list type XML. But, this XML is so simple that it is probably best to use a text editor. It is pretty obvious to tell that the content has a header, footer and key value pairs that are delimited by HTML type tags where a key is delimited by the `<key></key>` pair and a value is delimited by one of `<string></string>`, `<integer></integer>`, `<float></float>` pair.
- The keys and values in the XML are gathered in the [Info](#) tool and explained in the [Vwidget Code Reference Manual](#). For the most part, you can simply see a graph you want and then view the Info tool table to determine the keys and values to use in the XML.
- Titles values are `<string>` type, data arrays are also `<string>` type as the numeric values are not atomic, which saves a lot on using repetitive tags. Lengths are generally of type `<integer>` and numbers are of type `<float>`. However, since XML is simply a delimited string you can also use `<string>` type instead of `<integer>` and `<float>` type.
- Of course, this is not a tutorial on HTML or XML so I didn't explain the DOCTYPE, enclosing plist and dict types and other common knowledge items. Suffice it to say that if you don't know of such things then simply do what the pros do: Copy the XML above and only change the keys and values you need.

Column Chart

The following is the XML needed to make a column chart. Column charts are made with the [Set Of Scalars](#) task so first select that task from the [Home](#) page, enter "All" in the Fetch Data row and then touch the Fetch Data arrow to navigate to the [Fetch Data](#) tool. Then enter the URL to the XML content shown below. Then touch the "Fetch Data" button.

The next time you select a representation on the [Set Of Scalars](#) page the fetched data will show.

Available at this URL: [columnchart.plist](#)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>title</key>
<string>Set Of Scalars XML Test</string>
<key>x_title</key>
<string>XML-X-Title</string>
<key>y_title</key>
<string>XML-Y-Title</string>
<key>data_3</key>
```

```

<string>1 2 3 4</string>
<key>data_2</key>
<string>1 2 3 4</string>
<key>data_1</key>
<string>2 2 3 6</string>
<key>label_1</key>
<string>a kitty cat</string>
<key>label_2</key>
<string>dog eat dog</string>
<key>label_3</key>
<string>buddy</string>
<key>label_4</key>
<string>more</string>
<key>label_5</key>
<string>the sky</string>
</dict>
</plist>

```

Surface Graph

The following is the XML needed to make a surface graph. Surface graphs are made with the [Z Values](#) task so first select that task from the [Home](#) page and then touch the Fetch Data arrow to navigate to the [Fetch Data](#) tool. Then enter the URL to the XML content shown below. Then touch the "Fetch Data" button.

Available at this URL: [surfacegraph.plist](#)

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>title</key>
<string>Z2 Values XML Test</string>
<key>x_title</key>
<string>XML-X-Title</string>
<key>data_values</key>
<string>1 1 1 2 3 2 0 1 0 5 5 3</string>
<key>grid_x_length</key>
<integer>4</integer>
<key>grid_y_length</key>
<integer>3</integer>
</dict>
</plist>

```

3D Density Graph

The following is the XML needed to make a 3D density graph. 3D density graphs are made with the [Density](#) task so first select that task from the [Home](#) page and then touch the Fetch Data arrow to navigate to the [Fetch Data](#) tool. Then enter the URL to the XML content shown below. Then touch the "Fetch Data" button.

Available at this URL: [densitygraph.plist](#)

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>title</key>
<string>Density XML Test</string>
<key>x_title</key>
<string>XML-X-Title</string>
<key>data_values</key>
<string>0.2 0.2 0.1 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.3 0.3 0.3 0.3 0.3 0.3 0.3 0.3 0.3 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6</string>
<key>grid_x_length</key>
<integer>3</integer>
<key>grid_y_length</key>
<integer>3</integer>
<key>grid_z_length</key>
<integer>3</integer>
</dict>
</plist>

```

Least Squares

The following is the XML needed for least squares. Least Squares graphs are made with the [Least Squares](#) task so first select that task from the [Home](#) page and then touch the Fetch Data arrow to navigate to the [Fetch Data](#) tool. Then enter the URL to the XML content shown below. Then touch the "Fetch Data" button.

Available at this URL: [leastsquares.plist](#)

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">

```

```
<dict>
<key>title</key>
<string></string>
<key>x_title</key>
<string>XML-X-Title</string>
<key>y_title</key>
<string>XML-Y-Title</string>
<key>data_values</key>
<string>1 31 2 36 3 32 4 33 5 25</string>
</dict>
</plist>
```

Least Squares is unusual because the Info tool shows `<key>data_1</key>` and `<key>data_2</key>` as data keys, but the XML uses the `<key>data_values</key>` key. That is an intricate implementation detail of this task that I won't explain, but mention so that you know not to totally rely upon the Info tool for the Least Squares task.

Note

If you are viewing this on an iPhone, iPad or iPod touch then you can copy the plist URLs in this tutorial by touch-hold on a URL and then select the "Copy" button. Then launch the Graph (or Vwidget) application, go to the respective Task's [Fetch Data](#) page and touch-hold in the URL text field and then Paste the URL. Otherwise, simply touch the Fetch Data URL text field to bring up the keypad and type in a URL.

Conclusion

There are many graph types that can be updated with XML and those graphs can be accessed using the various tasks and their representations. With a little experimentation XML fetching may solve a lot of problems. Note that in a more vertical-market SOA client the Fetch Data URL is embedded in the application and not entered by the user. However, in the context of this manual and the application it describes, which is a horizontal-market application, the URL is entered by the user.

© Copyright 1993-2012 by [VViimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

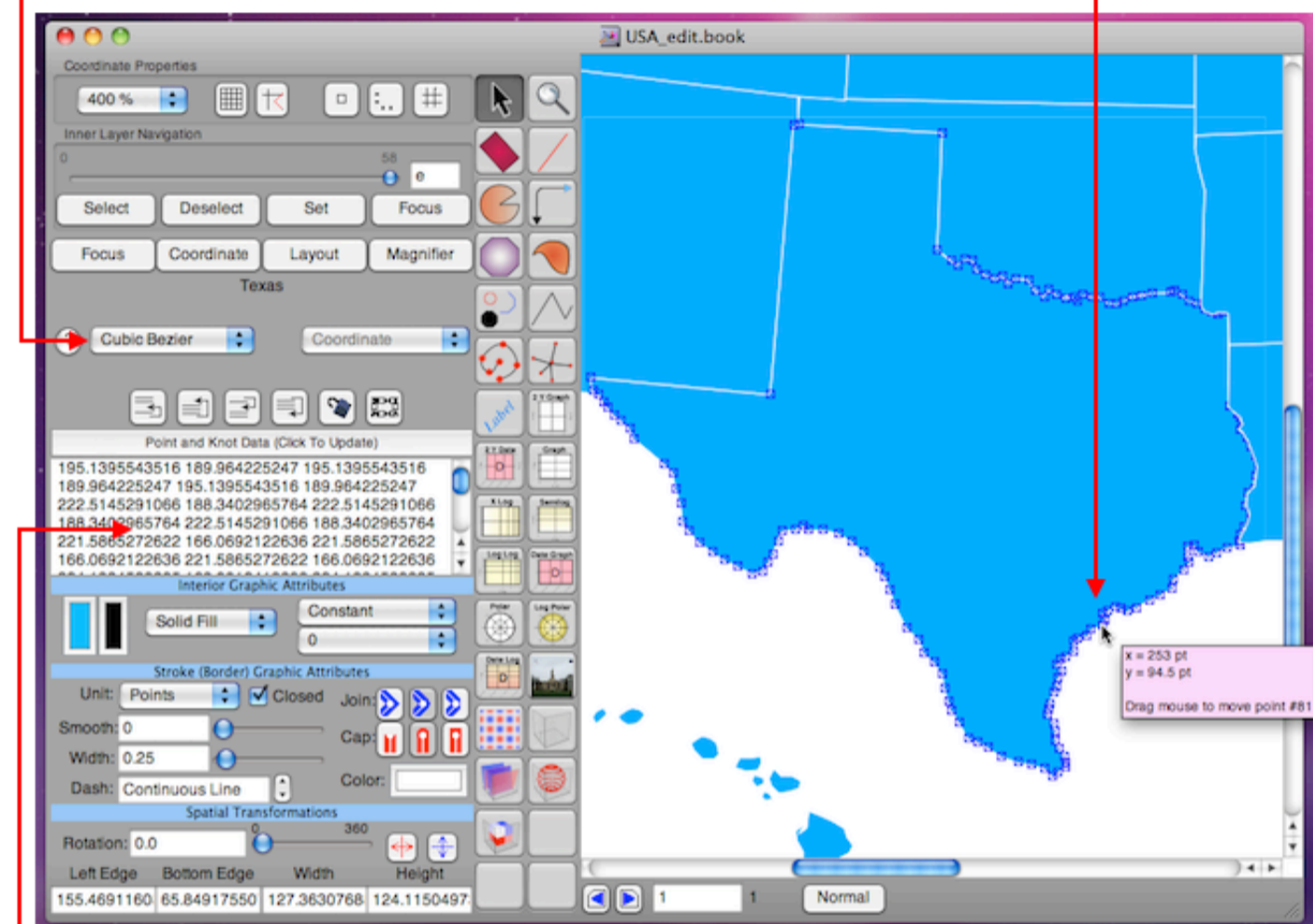
Graph > Tutorials > Making A Map

This section explains how make a document from a user perspective. The type of document is called a "Map" because it can be loaded into other applications and its components can be queried upon. Making a map is described first and loading the map into an existing application is described next.

First make a map using point and click methods of Vwidget Builder as demonstrated in the following diagram. Vwidget Builder has its own extensive manual, consult that manual for additional information.

Once done constructing the graphic then proceed to the Expert Inspector Editor to make settings.

You can edit graphics directly using usual mouse actions or enter the numeric data.



Enter numeric data here. This cubic bezier graphic accepts knot and vertex control points.

Other graphics such as Polygon accepts point data, which is much easier to deal with.

Custom programs exist which can batch process point and knot numeric values to automate document production.

The diagram above shows how the map of the United States of America (USA) was constructed. That map, called USA, is available in the Graph application. It was made by point and click methods building up each state individually and then abutting the states to form the entire USA. Here is a hint: scan an image, drag out the Image graphic, load your scanned image into it, trace over that image with other graphics and then if desired delete the original scanned image from the document leaving only the live graphical components. Geographic maps are challenging whereas process maps, control consoles and flow charts can be very easy, being rectangles and circles connected by lines or simply connected by each other. The creation of your own map is a matter of ingenuity since there are many ways to make maps. If you need to bulk process node data into maps then please contact support@vvi.com so we can support your efforts. We have specialized tools to process point information into documents and hence maps.

Once the map is constructed then go back to each element and assign a Description and other attributes as shown in the following diagram.

The Inspector window for the 'Texas' widget is shown. It has a title bar with three colored buttons (red, yellow, green) and the text 'Inspector' and 'Texas'. Below the title bar are two dropdown menus: one with a question mark icon and 'Expert' selected, and another with 'Coordinate' selected. A blue header bar contains the text 'Auto Resize Flags'. Below this are six checkboxes, all of which are checked: 'Left Margin', 'Horizontal', 'Right Margin', 'Bottom Margin', 'Vertical', and 'Top Margin'. Below the checkboxes is the text 'Resize Margin Insets:' followed by four input fields labeled 'Left:', 'Right:', 'Bottom:', and 'Top:', each containing the number '0'. Below the input fields is a dropdown menu with 'Do Not Maintain Aspect' selected. A blue header bar contains the text 'Special Dictionary Values'. Below this are four rows of text labels and input fields: 'Widget Description:' with 'Texas', 'Widget Name:' with 'Texas', 'Preferred Widget Class Name:' with 'VVC50', and 'Standard Widget Class Name:' with 'VVC50'.

Set the Auto Resize Flags accordingly.

Leave all the Margins zero unless you intend otherwise.

Only maintain aspect if you need that feature.

Enter the "Description", which shows up when you click or touch the component (in this case Texas).

Enter the "Name". The name can be anything for a component. When you group the components then the group name must be PVS_group.

When all the attributes described above are set then select all the elements (in this case the states) and group them (using the menu item Format > Group and choosing a normal group) and give the resulting group the name PVS_group (very important). You should probably set the group to autoresize in width and height and to maintain aspect.

To load your newly created document into the Graph application you will need to make it "Internet ready". That means you will have to take the normal Vwidget Builder document and export it as a skin document using the menu item File > Export To ... and choosing the Skin export type. It is that skin file, which is a compressed flat file that gets imported into the Graph application. Place that skin file on a web server and import it into Graph this way: touch the Maps task, touch a map entry other than USA for example the Map 1 row or a row that you have renamed, touch Edit, enter the URL of the skin file on the web server, touch the Fetch Primary Skin button and then finally touch the Done button. Your skin shows in the resulting view. *Hint:* On your Mac computer you can turn on Web Sharing and place the skin file in the folder /Library/WebServer/Documents and then turn on Wi-Fi on your iOS device. The URL will then be something like: `http://192.168.1.2/USA.book` where the IP address would be the one of your Mac and the USA.book part would be your own skin file name.

In summary, the steps to making a map are:

- Make a Map document.
- Export that document to a Skin file.
- Place that Skin file on a web server.
- Import that Skin file into the Graph application or similar application that has Vwidget Builder Skin importing features.

Making a simple map can take a few minutes, making a hard map can take days, batch processing thousands of maps from existing map data can take a few CPU hours. Once you have a map and wish to do more with it than simply query component names then consult the Document section to program your own application. For example, a process map can be used to touch a process element, for example schematic element representing a valve, to then issue commands to that valve to turn it on and off. Another example: Touch a state or other jurisdiction such as sales territory to then call the sales representative for that territory. The possibilities with maps are very interesting.

[Graph](#) > **Support**

The following is a brief list of support sections:

Support	Description
Legend	Describes how to make a legend for a graph.
Enter Data	Describes various ways to enter data.
Skins	Describes the basics for making a skin.
Graph For Mac	Gives a reference to the Graph User Manual for the Mac.
Programming	Gives a reference on how to make your own graphing application.
Question Answer	A brief list of question answers on subjects that are not covered elsewhere in this manual, or not obvious to find.

If you have a question that is not explained in this manual please contact support@vvi.com so that we may answer your question and update this manual as needed.

© Copyright 1993-2012 by [Vimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

[Graph](#) > [Support](#) > **Legend**

A legend is a graph key that associates a color with a label describing each data graphic (curve, bar, etc.) The [Preferences](#) tool is used to set the position of a legend and [Table Editor](#) editing is used to set the labels for legend.

Follow these steps to make a legend:

- Touch-hold each table column header (the grey portion at the top) and then from the resulting menu touch Edit. The text you enter in the Edit text field will also be entered into the label of the legend.
- Set the position of the legend using the Preference tool.

© Copyright 1993-2012 by [Vimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

[Graph](#) > [Support](#) > **Enter Data**

This section explains the ways to enter data. First off, there is no "one size fits all" way to enter data so don't expect a quick answer. The following is a bullet list of issues regarding entering data.

- The [Accelerometer Vectors](#) task show the correct way to enter data, the sensor does it for you. The [Location Tracking](#) task also enters data automatically although you do have to move for meaningful results.
- The [Health](#) and [Weight](#) tasks show how to enter data in a limited fashion. Data is entered as needed at the time and accumulated to show results. That way you are not subjected to entering data all at once. This is especially applicable when you use the [Purpose](#) tool.
- The next easiest way to enter data is with the [Fetch Data](#) tool. With that you can specify a URL on the web and fetch its contents. Of course, with that option your web engineers (if you have them!) must enter the data so that may be considered as cheating.
- The most flexible way to enter data is to write your own graph application, for a reference see the [Programming](#) section.
- If you are not fortunate enough to have data imported automatically then you must use manual entry. For that see the [Table Editor](#) tool. You can also use the [Edit Data](#) tool to enter data using conventional text editing methods and the [Show Data](#) tool to enter data using a conventional single column list format.
- Data formats are explain in each respective [Tasks](#) section.
- A tutorial on entering data is available at [Making A Line Graph](#) section. That explains textual entry using the conventional text editor, but the [Table Editor](#) tool may be more appropriate.

As was explained in advance, there is no silver bullet when entering data because there are simply too many perspectives on the subject. Hopefully one of the ways referenced in this section will be satisfactory. Finally, we are always looking for general classes of data generation so if you have a good idea please email support@vvi.com.

© Copyright 1993-2012 by [VViimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

Graph > Support > Date Graph

The [Set Of 2D Points](#) task supports date entry for the x-value. If you enter x-values in this format: MM/DD/YYYY hh:mm:ss.f where:

Date Fields

MM	Months, a number from 1 to 12
DD	Days, a number from 1 to 31
YYYY	Years, a four digit number
hh	Hours, a number from 0 to 24
mm	Minutes, a number from 0 to 60
ss	Seconds, a number from 0 to 60
f	Fraction of second, an integer

then the x-value is interpreted as a Gregorian date. The MM/DD/YYYY part is mandatory while the hh:mm:ss.f is optional. Here are some date entry examples:

Date	Explanation
12/1/2010	A valid calendar date
12/1/2010 8:0:0	A valid calendar date and time
12/1/2010 8	An invalid time, the time part must contain two colons and three numbers.
12/1/99	An invalid date, the date part must contain four decimals indicating the century.

The following are a few issues when working with dates:

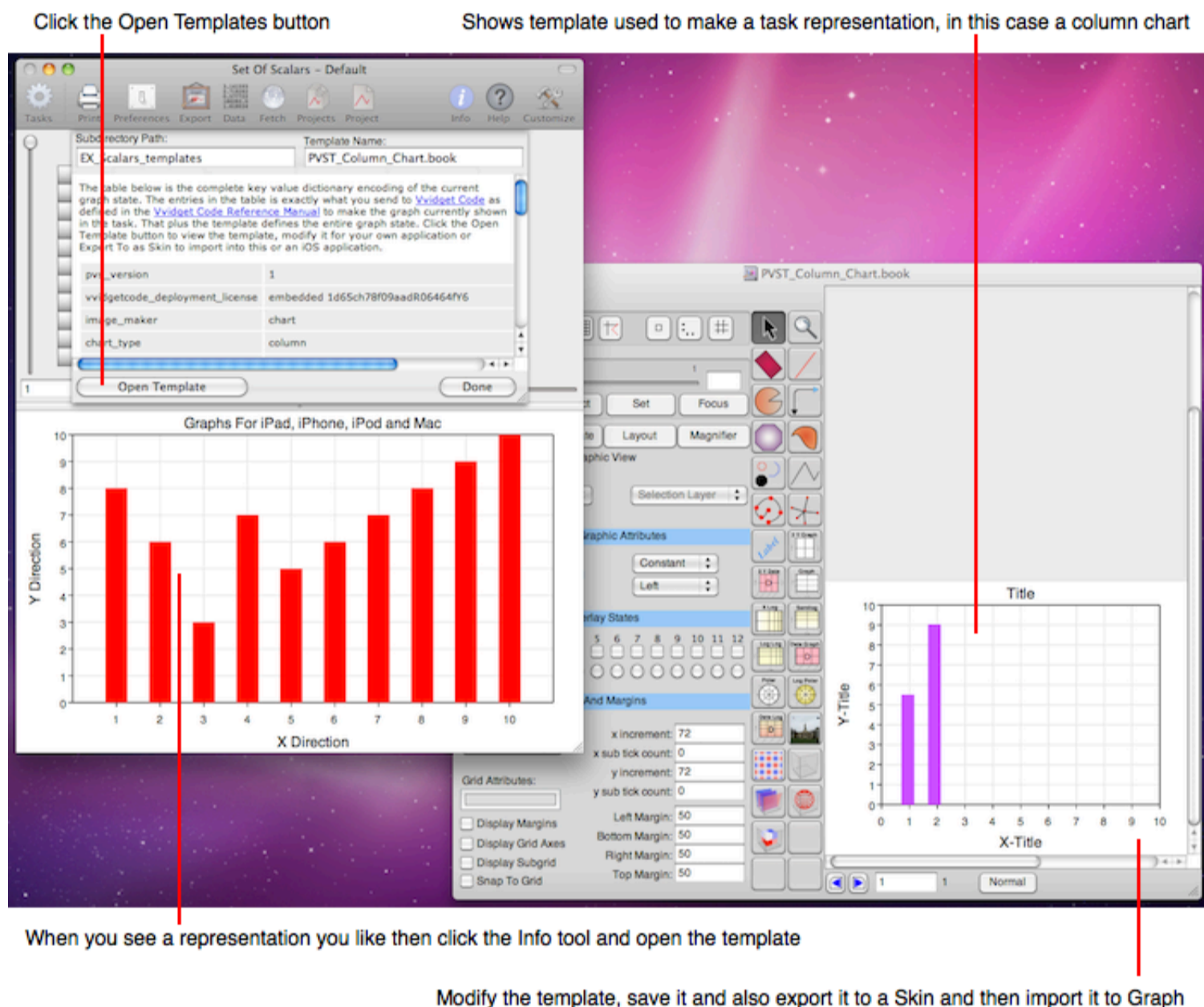
- The date is simply the x-value and must be followed by a y-value scalar in order to complete the 2D point entry.
- Only the Gregorian calendar is supported and the month must be the first number in the date.
- Data entry supports date or scalar representation for the x-value but not both at the same time. For table cell entry, if you enter a date for a x-value then all x-values are then interpreted as dates. If you enter a scalar for a x-value then all x-values are then interpreted as scalars in which case any previously entered dates are shown as seconds since 1/1/1970.
- If you paste in a series of dates then each date y-value entry must be on one line only, i.e.: a delimiter of a return character must separate each 2D point. In this case, date entry is determined by looking for a slash (/) in the first 20 characters of the pasteboard string. If a slash is found then all subsequent x-values are assumed to be in date format.
- In order to appear on all the coordinate systems, the date is remapped to a second since 1/1/1970 as an intrinsic basis, hence dates can also be entered as seconds since January 1, 1970.
- Dates are most important for the Curve > Date graph where the x-axis is shown in Gregorian calendar format and scaled to any of the fields specified in the table above. For example, if your x-values span months then the x-axis will show increments in months, if your x-values span years then the x-axis will span years, if your x-values span seconds then the x-axis will span seconds, and so on and so on. This autoscale feature and the resulting x-axis label format is rather complex and can be altered with the use of [Skins](#). Also note that the x-axis tick increments are uniform in date field, but not uniform in the second basis hence the ticks may appear non-uniformly spaced. A good date graph is particularly complex.

Graph > Support > Skins

The Graph application is designed to be very straightforward, easy to use and also unencumbered by indirection or any lack of focus. It is simply *"You give it data and it gives you a graph"* as stated in the front of this manual. It does that in a stylistic way of presenting [Tasks](#) and [Representations](#). The use of skins are an essential component which facilitates that goal but making skins is a completely different process. The goal of a skin is to leverage the juggernaut of graph tools in order to fine tune graphical attributes of a task's representation. Thus if you decide to use skins to their fullest then you will need to employ the following:

- **Vwidget Builder:** A general graph layout program, see its manual: [Vwidget Builder User Manual](#)
- **Vwidget Code:** A general key-value based graph generator, see its manual: [Vwidget Code Reference Manual](#). You will only need to know about the template and graphic prototype names.

You might think that the machinery to use skins is vast, complex and overwhelming. Well, you are half right. But, perhaps the following will provide an entrance point for you. The diagram below shows how you can use the Chart Tasks in Vwidget Builder to get a template for a task representation. Simply use the Chart Tasks (or the Graph program on the Mac) to view a representation and when you like the representation then click to Info Tool and then the Open Templates button on the Info Tool, as shown below.

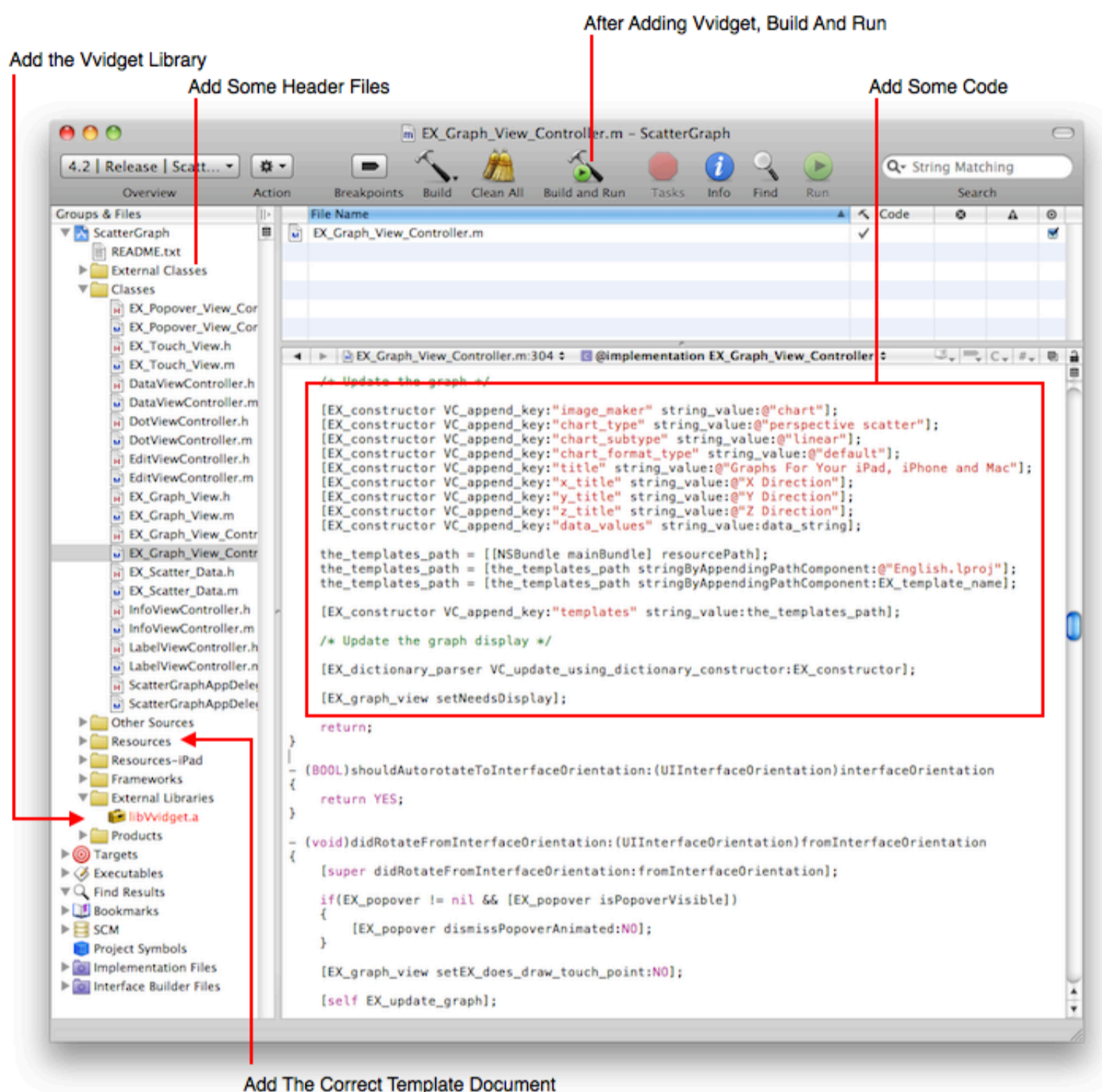


That will open a Vwidget Builder document with the template that correspond to the task and representation. Then Save As the template to your disk and then modify the appearance of the template as you wish. Once finished with the modifications choose the File > Export To ... menu item and save the template as a skin. That skin can then be imported into the Graph application using the [Edit Graph](#) tool.

Graph > Support > Programming

The [Fetch Data](#) tool provides some capability to program data generation. Combined with animation, the data can be updated (programmed) periodically.

However, for true programming you may be interested in the [Vwidget Code Reference Manual](#). The figure below shows the essence of what you have to do for that. Once you get the hang of it it is actually very easy and effective.



[Graph](#) > [Support](#) > **Question Answer**

Below are answers to commonly asked questions about Graph. If you have a question please mail support@vvi.com.

Question: How do I export a graph?

- **Answer:** To export a graph touch and hold it for about two seconds and then select one of the export options.

Question: How do I program my own graph application like Graph?

- **Answer:** See the [Programming](#) section.

Question: What is the difference between the Vwidget application and the Graph application?

- **Answer:** The Graph application is currently free and has ads. The Vwidget application is the same as the Graph application but without ads and currently is not free.

© Copyright 1993-2012 by [VViimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

[Graph](#) > Administrative

The following is a brief list of Administrative sections:

Administrative	Description
License Agreement	The Graph End User License Agreement.
Trademarks	A list of trademarks used in this manual.

© Copyright 1993-2012 by [VViimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

Graph > Administrative > End User License Agreement

Below is a copy of the End User License Agreement that you received with your copy of Graph and was presented to you before installing Graph.

Graph v10.7.2

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 VVi Imaging 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 the SOFTWARE PRODUCT permanently to another person ONLY IF:
 - (a) that person agrees in writing to accept all of the terms and conditions of this Agreement; and
 - (b) you notify VVI in writing that you have transferred your copy of the SOFTWARE PRODUCT; and
 - (c) you cease all use of the SOFTWARE PRODUCT.
- 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 traddress 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 - 10.5.8 - 7/19/2009

© Copyright 1993-2012 by [VVimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.

[Graph](#) > [Administrative](#) > Trademarks And Legal

Manipulating graphs and data graphics and other manipulations unique to Vwidget 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, Vwidget, OpenGraph, VVI, VVimaging, SAM, "State Automation", "State AutoMation", and Vwidget with any word combination are trademarks of VVimaging, Inc (VVI) in the United States and world-wide.

The OpenGraph logo, Vwidget logo, Vwidget 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.

© Copyright 1993-2012 by [VVimaging, Inc. \(VVI\)](#); All Rights Reserved. Please email support@vvi.com with any comments you have concerning this documentation. See [Legal](#) for trademark and legal information.