

# Omnis Graphs

Omnis Software

May 1997

The software this document describes is furnished under a license agreement. The software may be used or copied only in accordance with the terms of the agreement. Names of persons, corporations, or products used in the tutorials and examples of this manual are fictitious. No part of this publication may be reproduced, transmitted, stored in a retrieval system or translated into any language in any form by any means without the written permission of OMNIS Software.

© OMNIS Software, Inc., and its licensors 1997. All rights reserved.

Portions © Copyright Microsoft Corporation.

OMNIS® is a registered trademark and OMNIS 5™, OMNIS 7™, and OMNIS Studio are trademarks of OMNIS Software, Inc.

Microsoft, MS, MS-DOS, Visual Basic, Windows, Windows 95, Win32, Win32s are registered trademarks, and Windows NT, Visual C++ are trademarks of Microsoft Corporation in the US and other countries.

Apple, the Apple logo, AppleTalk, and Macintosh are registered trademarks and MacOS, Power Macintosh and PowerPC are trademarks of Apple Computer, Inc.

IBM and AIX is a registered trademark and OS/2 is a trademark of International Business Machines Corporation.

UNIX is a registered trademark in the US and other countries exclusively licensed by X/Open Company Ltd.

Sun, Sun Microsystems, the Sun Logo, Solaris, Java, and Catalyst are trademarks or registered trademarks of Sun Microsystems Inc.

HP-UX is a trademark of Hewlett Packard.

OSF/Motif is a trademark of the Open Software Foundation.

Acrobat is a trademark of Adobe Systems, Inc.

ORACLE is a registered trademark and SQL\*NET is a trademark of Oracle Corporation.

SYBASE, Net-Library, Open Client, DB-Library and CT-Library are registered trademarks of Sybase Inc.

INFORMIX is a registered trademark of Informix Software, Inc.

EDA/SQL is a registered trademark of Information Builders, Inc.

CodeWarrior is a trade mark of Metrowerks, Inc.

Other products mentioned are trademarks or registered trademarks of their corporations.

# Table of Contents

- CHAPTER 1—OVERVIEW .....5**
- CHAPTER 2—GRAPH BASICS .....9**
  - LIST STRUCTURE AND TERMINOLOGY .....9
  - CREATING A GRAPH .....12
  - MAJOR GRAPH TYPES.....17
  - MINOR GRAPH TYPES.....20
  - GRAPH OBJECTS.....24
  - GRAPH PROPERTIES.....26
  - DATA FORMAT.....31
- CHAPTER 3—MODIFYING YOUR GRAPH.....32**
  - SIZING AND POSITIONING OBJECTS .....32
  - COLORS AND PATTERNS .....33
  - TEXT OBJECTS AND FONTS .....34
  - GRAPH ORIENTATION.....35
  - LEGENDS .....35
  - PSEUDO 3D.....36
  - AXES .....37
  - SCALING.....40
  - DATA MARKERS.....45
  - PICTURES.....46
  - SWAPPING THE SERIES AND GROUPS.....46
  - GRID LINES.....48
  - 3D GRAPHS .....50
  - PIE CHARTS .....51
  - SCATTER GRAPHS .....55
  - HISTOGRAMS .....56
  - STOCK MARKET GRAPHS.....57
  - CHANGING A GRAPH AT RUNTIME.....60
  - SAVING YOUR GRAPH TO A FILE.....61
  - DRILLDOWN .....61
  - DRAW AND DROP.....64
- CHAPTER 4—GRAPH PROPERTY REFERENCE.....66**
  - GRAPH PROPERTIES.....66

# About This Manual

This manual describes OMNIS Graphs, the complete graphing system for your OMNIS applications. This manual covers the following topics

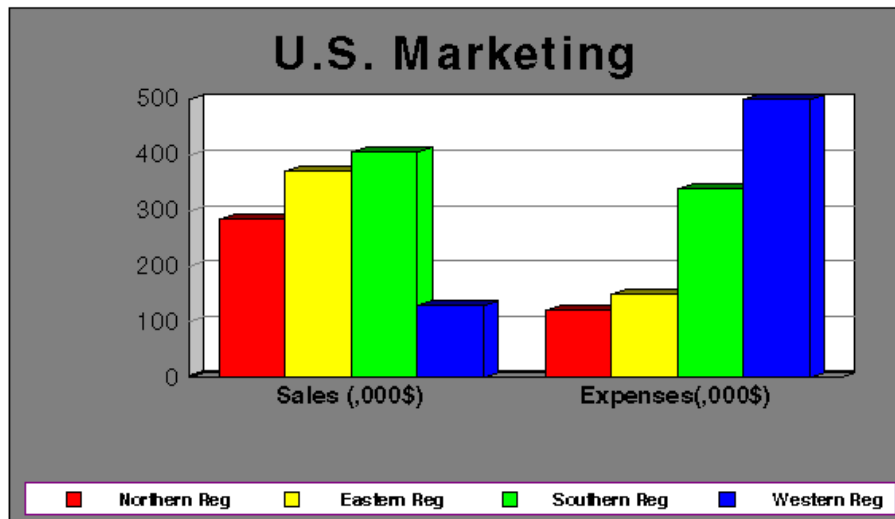
- ❑ *Graph Basics*  
an introduction to the underlying elements that make up a graph
- ❑ *Graph Types*  
the different types of graph supported in OMNIS
- ❑ *Modifying Your Graph*  
important features of 3D, Pie, Scatter graphs, Stock Market graphs, including techniques such as drag-and-drop and drilldown
- ❑ *Graph Property Reference*  
an alphabetical list of all the graph properties

Before using this manual you should have a thorough understanding of the following OMNIS features, described in the *Using OMNIS Studio* manual:

- ❑ Building and modifying OMNIS windows and fields on windows
- ❑ Creating and populating OMNIS lists
- ❑ OMNIS programming, data types, and OMNIS commands

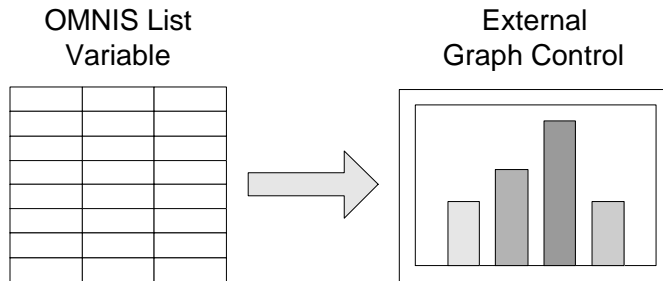
# Chapter 1—Overview

A graph is one of the most effective ways to represent numerical data. Your data or information takes on greater significance when the numbers appear as bars, columns, lines or markers on a grid, enhanced with descriptive titles and accentuated by colors. The following graph shows the Sales and Expenses for four different sales regions, allowing you to make an instant comparison of the different regions.



You can view your data in more than one style or type of graph, including Bar, Line, Area, Scatter, Pie and in 2D or 3D view as well. Graph objects are available in the Component Store as *external components*, and you can place them on a window or report class in exactly the same way as other external components and fields. The Graph Library containing the Graph Control is installed and loaded by default so it should be available at all times in the Component Store. If it is not, refer to the *Using OMNIS Studio* manual for details about loading it and for general information about external components.

When you create a graph object you select the type of graph most suited to representing your data. The data for your graph is taken directly from an OMNIS list variable.



When you first place a graph on your window or report in design mode, you can modify its properties using the Property Manager. A graph is given various design mode properties that let you set the type and basic appearance of your graph. To create a graph you

- ☐ Place an external Graph Control field on your window or report
- ☐ Set the dataname of the graph control to the name of your list variable
- ☐ Create a method to build your list of data, usually in the `$construct()` method of the window or report
- ☐ Open the window or report instance in runtime to draw the graph

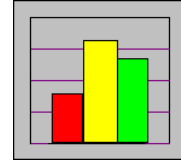
At runtime you can change the appearance and behavior of the graph using the notation. Also you can write methods behind the graph to respond to user clicks to enable such features as drag and drop, and drilldown.

# Types of Graph

The graph external component supports several standard or *major types*: Bar, Line, Area, Pie, 3D, and Special. Each of these major types has several subtypes or *minor types*. The following major graph types are available

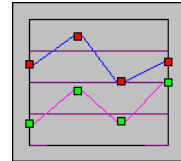
☐ **Bar chart**

represents individual amounts as risers, or compares figures over a period of time; you can change the graph orientation to show horizontal bars



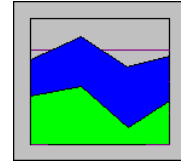
☐ **Line graph**

emphasizes the rate of change rather than individual amounts



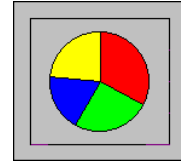
☐ **Area graph**

emphasizes the amount or magnitude of change rather than the rate of change



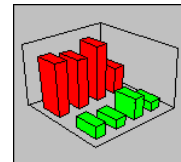
☐ **Pie chart**

shows the relationship of the various parts to the whole; you can highlight or explode individual segments, and show multiple pies at once



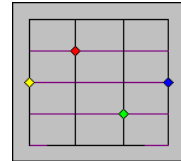
☐ **3D graph**

compares data points along two axis giving greater visual impact to your data; you can change the shape of bars to diamond, pyramid, octagon, and so on



☐ **Special graph**

includes Stock Market, Histogram, and Scatter; the latter plots the data as xy coordinates and shows the relationship of different groups of numerical data



# Software Requirements

Under Windows, the DGDSC.DLL file (DGDSC32.DLL for Win 95 and NT) must be present in the OMNIS directory, where it is placed by the OMNIS installer. Under MacOS, DGDSC must be present in the OMNIS folder.

## What's New in OMNIS Studio 2.0

The following properties and methods have been added to the Graph external component. In addition, the graph minor types kGRBoxPlot and kGRstemLeaf have been added to the kGRspecial major type.

### Graph Properties

\$objectname	returns the name of the current selected object, if \$selectobject is kTrue
\$columnheadings	a row or list variable name; the values stored in the first line are used to label the groups replacing the column names supplied in the data list

### Graph Methods

The following methods have been added to the Graph external component.

- \$getobject(cObjectname[,cSeries,cGroup])  
returns the current selected object and its series and group ID
- \$drawpixelline(iStartx,iStarty,iEndx,iEndy[,iColor,iLinewidth])  
draws a line of given color, between 2 virtual co-ordinates; returns an ID of the line
- \$drawdataline(iStartListEntry,iEndListEntry[bHidethese,iColor,iLinewidth])  
draws a line of given color, between 2 data points; returns an ID of the line
- \$removeline(iLineid)  
removes a line which was created by either \$drawpixelline or \$drawdataline
- \$pvpoint(iX,iY) or \$vppoint(iX,iY)  
converts a pixel co-ordinate (x,y) to a virtual co-ordinate, or vice versa
- \$pvscaleheight(iX) or \$vpscaleheight(iX)  
converts from a pixel height to a virtual height, or vice versa
- \$pvscalewidth(iX) or \$vpscalewidth(iX)  
converts from a pixel width to a virtual width, or vice versa
- \$snapshot([iPicWidth,iPicHeight])  
returns a picture field of the current graph



# Chapter 2—Graph Basics

This chapter describes the list structure you need to create a default graph, some basic list and graph terminology, and summarizes the various major and minor types of graph available in OMNIS.

This manual uses the following data to demonstrate the different types of graph. The table shows the sales and expenses data for four regions in an organization and includes the names of the sales executives and their region, the projected and actual sales, and expenses data (in \$1000s) for each region.

		\$1000s		
Sales executive	Region	Projection	Sales	Expenses
Fred	North	300	285	120
Sam	East	350	370	150
George	South	600	405	340
Niles	West	400	130	500

## List Structure and Terminology

The data in your list variable can be from a server or OMNIS database. You can define your list using the *Set current list* and *Define list* commands, or the `$define()` or `$definefromtable()` methods. You can build your list using the *Build list from select table* or *Build list from file* command, or for a list based on a table class you can use the `$fetch()` method. Alternatively, you can build a static list using the *Add line to list* or `$add()` method. You should refer to the *Using OMNIS Studio* and *OMNIS Programming* manuals for a description of OMNIS lists and client/server programming, and to the *OMNIS Help*, available under the Help menu in OMNIS, for a description of the commands to create, populate, and manipulate lists.

## Ordinals, Series, and Groups

The structure or format of your list determines the layout and contents of your graph. Consider the following list `SALES_LST`, built using a subset of the sample data.

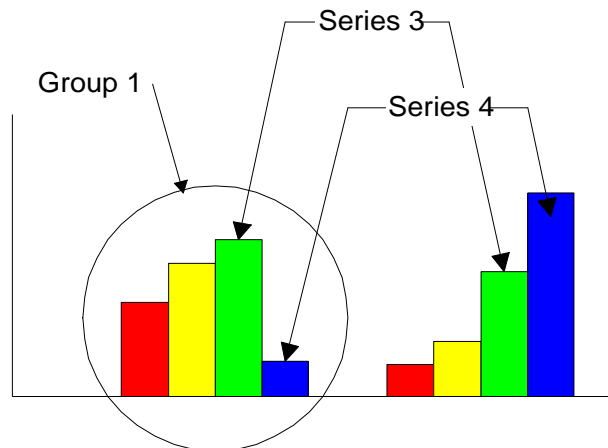
Variable `SALES_LST`

Options View

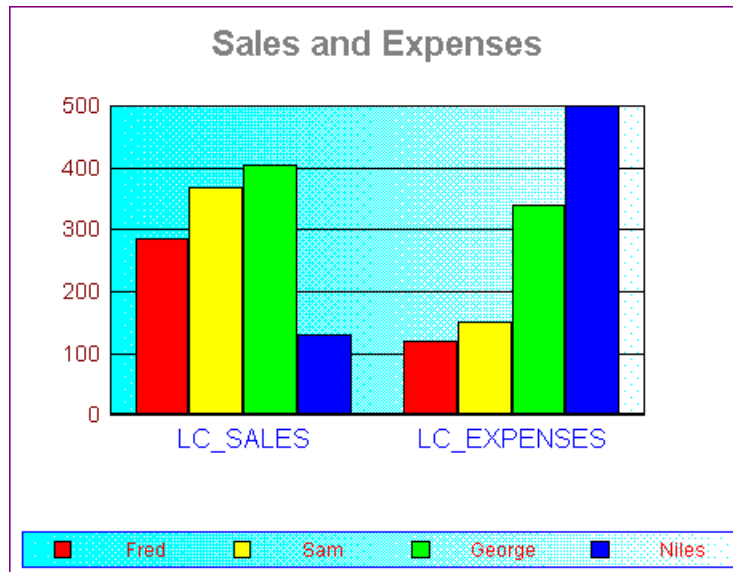
	LC_NAME	LC_SALES	LC_EXPENSES
1	Fred	285	120
2	Sam	370	150
3	George	405	340
4	Niles	130	500

SALES\_LST List

The rows and columns in your list and its column headings become specific objects on the graph. The data in the list is interpreted in the graph in the following way.



The contents of SALES\_LST displays the following default bar graph.



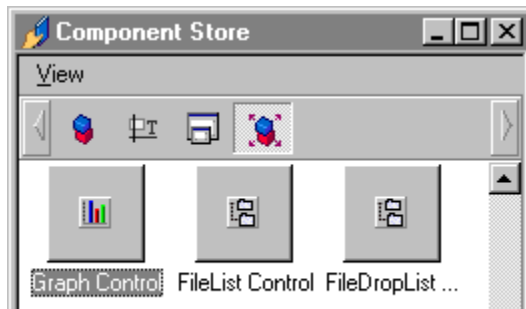
The names of the sales executives contained in the first column of the list LC\_NAME are used to label different parts of the graph, in this case the legend. The second list column LC\_SALES is the *first* group of data and is represented by the first group of risers on the graph, the third column LC\_EXPENSES is the *second* group and is represented by the second group of risers. The number of rows in the list translate into the number of series in each group in the graph, in this case there are 4 series for each group. The data for Fred is *series 1* in every group, the data for Sam is *series 2*, and so on. The variable names or column headings in the list are used to label the groups.

# Creating a Graph

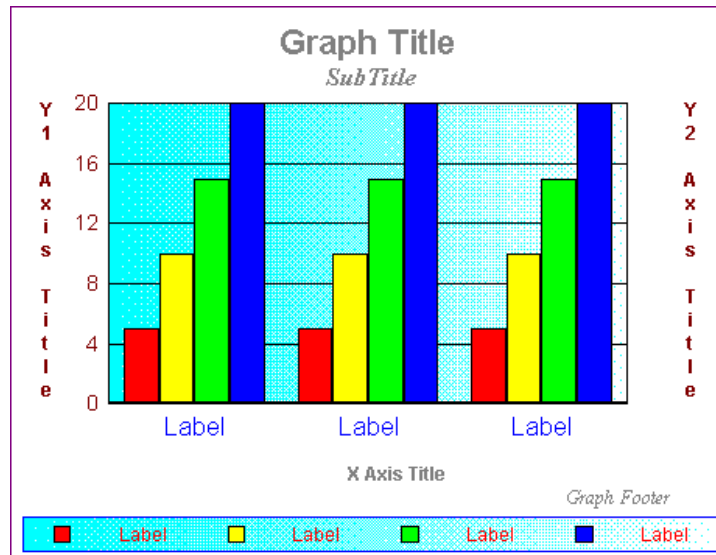
Graphs are available in the Component Store as *external components*, and you can place them on a window or report class in exactly the same way as other external components and fields. The graph external component or Graph Control is contained in the Graph Library which is installed and loaded in OMNIS by default. This section describes how you place a graph on a window class.

## To add a graph control to your window

- Open your window in design mode
- Click the External Components button on the Component Store toolbar



- Select the *Graph Control* icon and create a field on your window

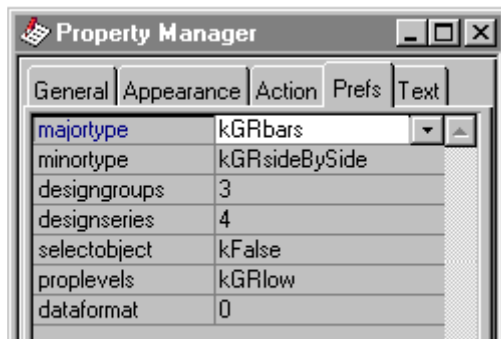


The graph control has certain default properties, including the Bar major type and Side by Side minor type. At present, in design mode, it does not contain or display your data.

The Property Manager lists the properties of the whole graph; at this stage you cannot click on individual objects in the graph. The **dataname** property under the General tab in the Property Manager specifies the name of the list variable supplying the data to your graph.

- Open the method editor for the class and create a list variable and the necessary columns for your list, otherwise use existing variables
- Click on your graph field and set its **dataname** to the name of your list variable

Under the Appearance tab, you can set the **edgefloat** and **effect** properties for the graph field. Under the Prefs tab you set the other properties for the graph, including the **majortype** and **minortype** properties which control the type of graph displayed.



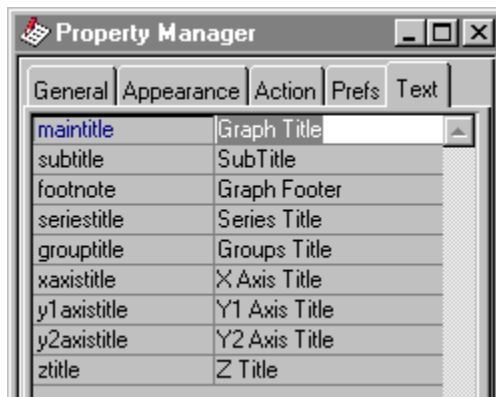
<b>majortype</b>	the major type of the graph, a constant: kGR3D, kGRarea, kGRbars (the default), kGRlines, kGRpie, kGRspecial
<b>minortype</b>	the minor type of the graph, a constant depending on the major type
<b>designgroups</b>	the number of groups in the graph in design mode; this relates to the number of data columns in your list ignoring the first column which is normally used for labels
<b>designseries</b>	the number of series in each group in design mode; this relates to the number of rows in your list
<b>selectobject</b>	if true lets you select individual objects within the graph, to view and change their properties
<b>proplevels</b>	the level or visibility of properties for the graph or current object, a constant: kGRlow (the default), kGRmedium, kGRhigh
<b>dataformat</b>	specifies the format of the data in your list required to define a single data point on the graph, it mainly affects Special graph

types; therefore the format or structure of the data in your list, the graph type, and the setting of this property determine the graph that is displayed; a value of 0 (the default) specifies the correct format for most basic types of graph

The **designgroups** and **designseries** properties let you control the overall appearance of your graph in design mode by setting the number of groups and series in your graph. At runtime your graph may well look different since the data in your list will change dynamically, in particular, the number of series in each group will probably differ since the number of rows in your list is likely to change. Using the sample data, you could set **designgroups** to 2 and **designseries** to 4 to show approximately how the graph would appear at runtime. The **selectobject**, **proplevels**, and **dataformat** properties are described later in this chapter.

- Set **designgroups** and **designseries** to the number of groups and series

You can set the main titles and labels for the graph under the Text tab in the Property Manager. Some of the labels are irrelevant for the default bar graph and apply only to other types of graph.



- Set the properties **maintitle**, **subtitle**, and **footnote** as appropriate by replacing the default text with your own; delete the text for any labels you want to hide at runtime

Having set up the graph on your design window, the next stage is to add a method to define and build your list of data.

- Right-button/Ctrl-click on your window and select the Class Methods from its context menu

Depending on where your data is stored you can use any of the following methods which use the sample data and variable names. You can create a class method called BuildList and call it from the \$construct() method in your window using the *Do method* command.

```

; $construct() method in your window
; declare variable SALES_LST of List type
Do method BuildList

; BuildList method for a list based on a table class
Do SALES_LST.$definefromtable(T_SALES)
Do SALES_LST.$select() Returns myFlag
Do SALES_LST.$fetch(4) Returns myFlag ;; fetches 4 rows

; BuildList method for OMNIS data using 4GL commands
; declare variables SALES_LST, LC_NAME, LC_SALES, LC_EXPENSES
Set current list SALES_LST
Define list {LC_NAME,LC_SALES,LC_EXPENSES}
Build list from file F_SALES

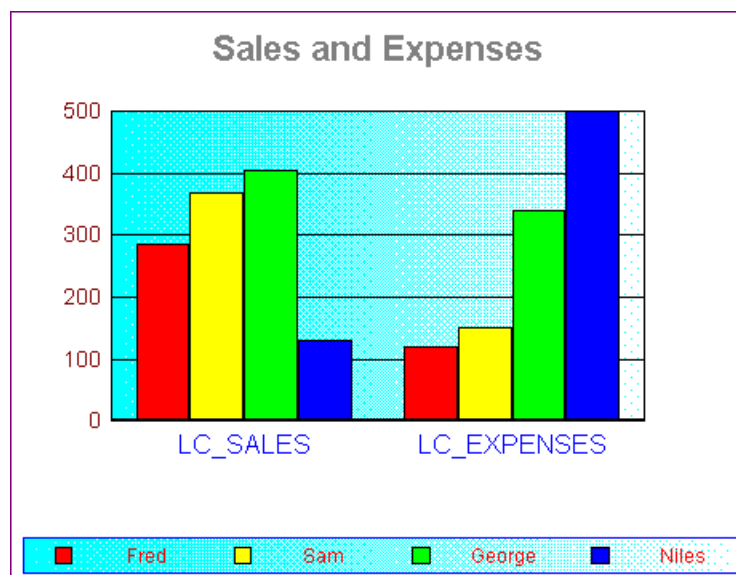
; BuildList method for a static list
Do SALES_LST.$define(LC_NAME,LC_SALES,LC_EXPENSES)
Do SALES_LST.$add('Fred',285,120)
Do SALES_LST.$add('Sam',370,150)
Do SALES_LST.$add('George',405,340)
Do SALES_LST.$add('Niles',130,500)

```

When you have added a method to define and build your list of data you can open your window and see how the graph looks.

- Right-button/Ctrl-click on your window and select Open Window, or press Ctrl/Cmnd-T to open your window

The sample data produces the following default bar graph.



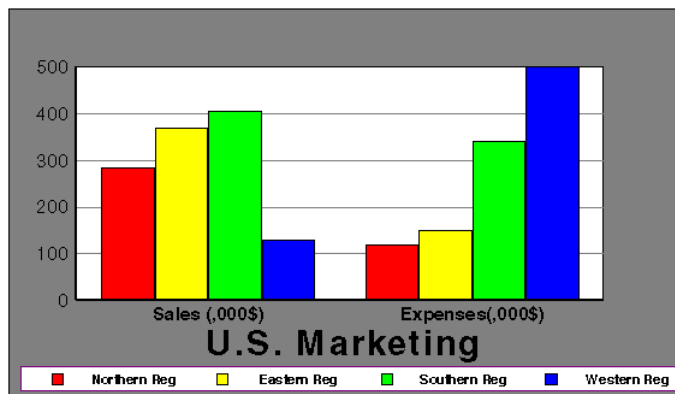


# Major Graph Types

The graph control displays a bar chart by default and sets the **major**type and **minor**type properties to kGRbars and kGRsideBySide, respectively. To create a different graph you need to change the major and minor type of the graph. For the more complex graph types such as 3D and Special graphs you may also need to change the structure or format of your list data as well, which is described later in this manual. The following section summarizes the range of major and minor types available.

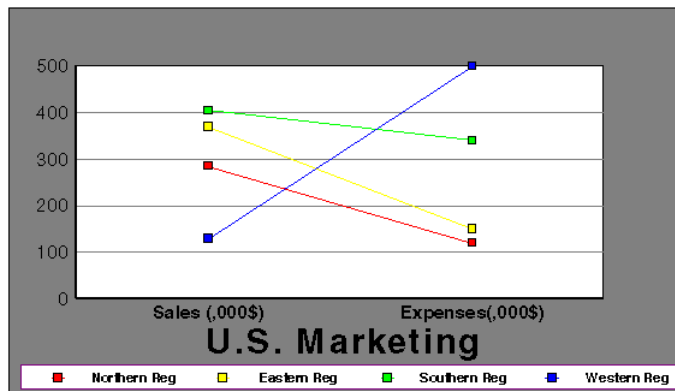
## Bar

A default bar graph represents each value as a separate rectangular area called a *riser*.



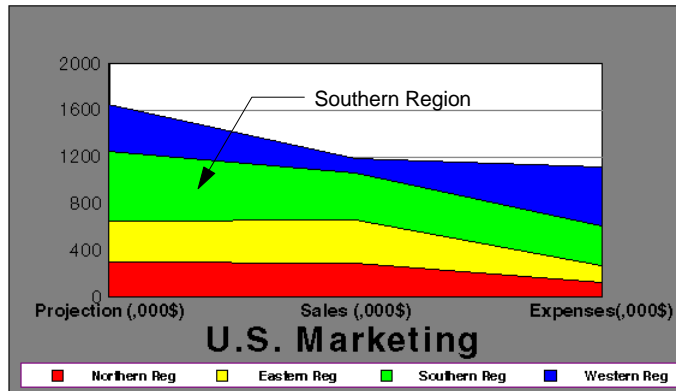
## Line

A line graph connects each data point in the same series with a line. The following graph uses the kGRlines **major**type and kGRabsolute **minor**type.



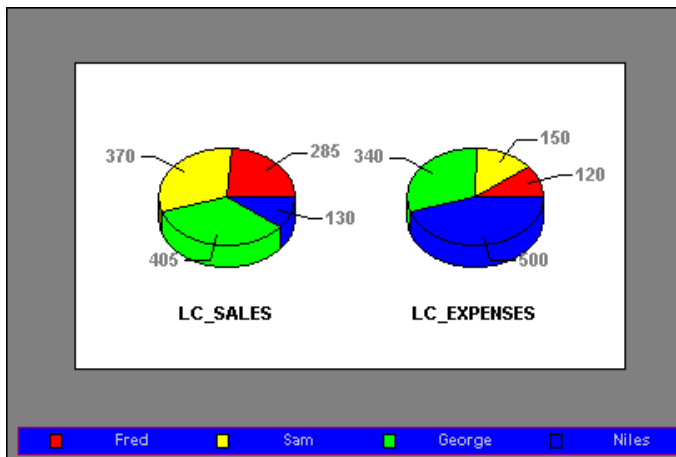
## Area

The following graph has kGRarea **majortype** and kGRstacked **minortype** and uses the Name, Projection, Sales, and Expenses columns of data, thus the graph has three groups. To plot area graphs, OMNIS places the first group on the left axis and spaces remaining groups along the horizontal axis. At each group along the x-axis, the area between two stacked lines represents an individual value. For example, the green area represents the Projection, Sales, and Expenses data for the Southern region or third series.



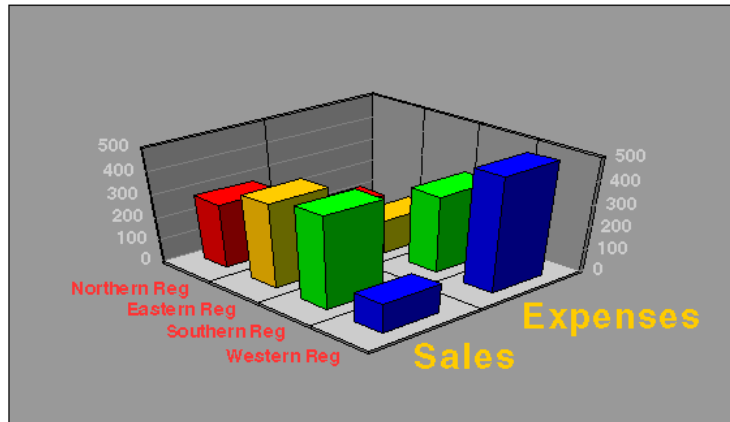
## Pie

A pie chart shows the relationship of the various parts to the whole and can be 2D, or 3D with thickness and tilt as shown. You can control the appearance and positioning of each *feeler* (text label and line pointer), the direction of the slices, and accentuate aspects of data by slicing off portions of the pie. The following graph uses the kGRpie **majortype** and kGRmultiplePie **minortype**; each pie represents a group, the slices represent series.



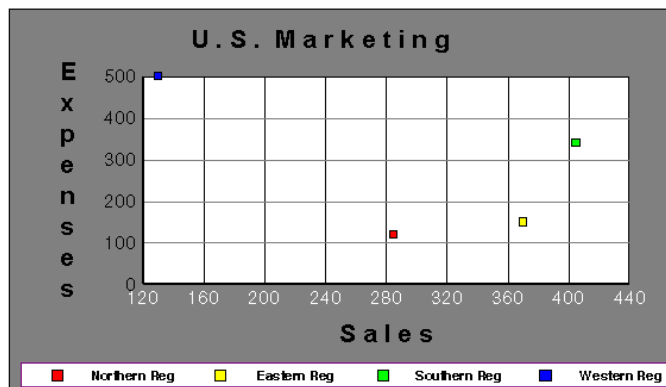
## 3D

A 3D graph represents each value as a 3D object or riser. You can control the appearance of the axes, floor, and walls, and the color and pattern for each object. The following graph uses the kGR3D **major**type and kGR3Dbars **minor**type.



## Special

There are a number of special graph types including Scatter, Stock Market, and Histogram. The following graph uses the kGRspecial **major**type and kGRscatter **minor**type. In a scatter graph each data point is plotted as a data marker against the x- and y-axis; in this case, the Sales and Expenses data for a region is plotted as a single data marker. You can control the size, shape, and color of the data markers, and you can change the style and frequency of the grid lines.

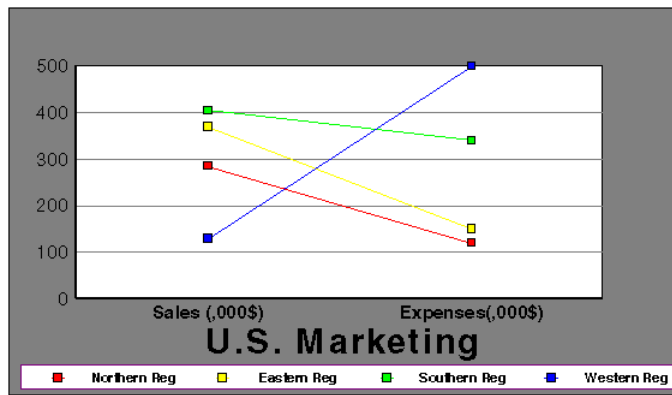


# Minor Graph Types

You can further modify the type of graph by setting its minor type.

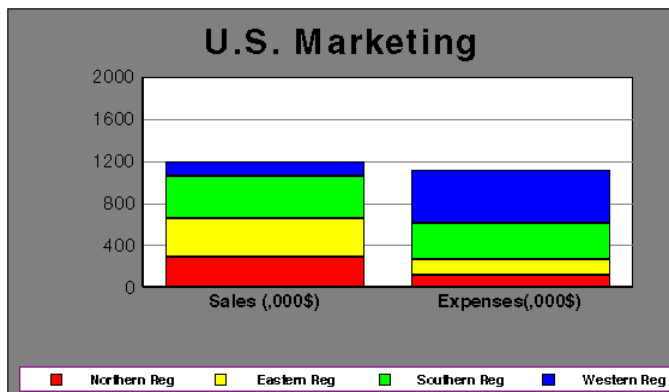
## Absolute

An *absolute* graph plots your data as absolute values, rather than relative values or percentages. The following graph uses the kGRline **major**type and kGRabsolute **minor**type.



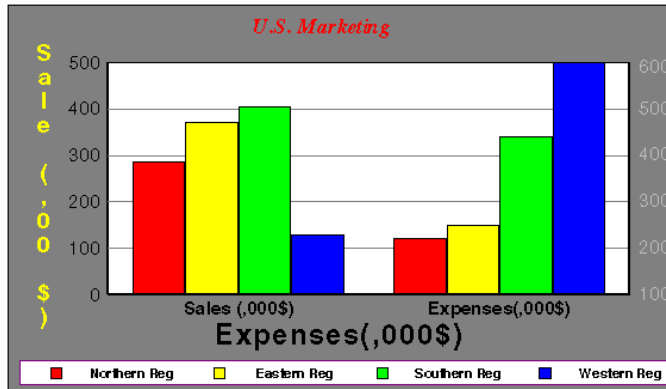
## Stacked

You can cumulate the values and plot the cumulative values on Bar, Line, and Area graphs using the *stacked* minor type. The following graph has kGRbars **major**type and kGRstacked **minor**type and shows the cumulative values for Sales and Expenses; note however the underlying list data is the same.



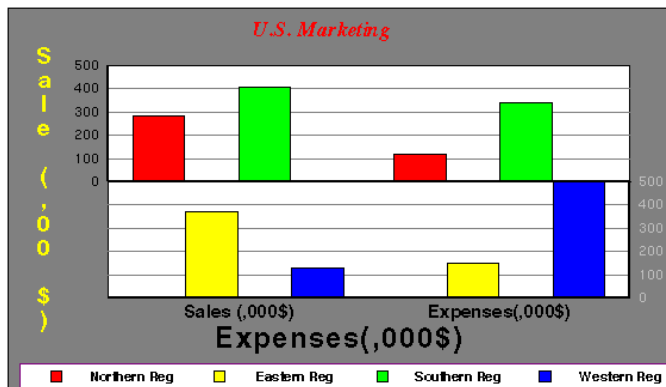
## Dual Y

A *Dual Y* graph has a y scale on the left and right side of the graph. The left scale relates to the first group of data, and the right scale the second group. The following graph uses the kGRbars **major**type and kGRsideBySideDualY **minor**type. Note that the range for the Sales figures runs from 0 to 500 on the left side, while that for the Expenses runs from 100 to 600 on the right side of the graph.



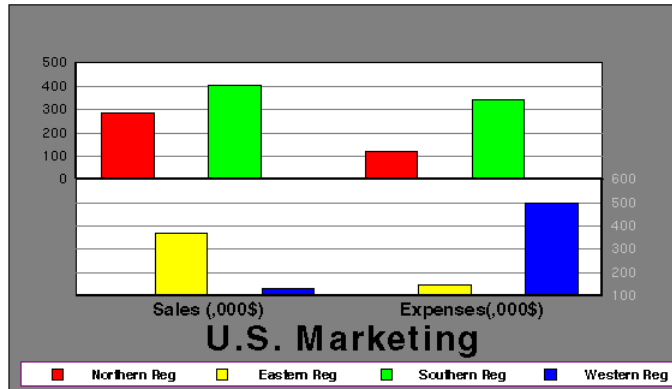
## Bipolar

Bipolar graphs have two horizontal axes and place alternate risers in each group in the top and bottom half of the graph. The scaling is the same for the vertical axis, but the labels are placed on alternate sides. The following graph uses the kGRbars **major**type and kGRsideBySideBiPolar **minor**type.



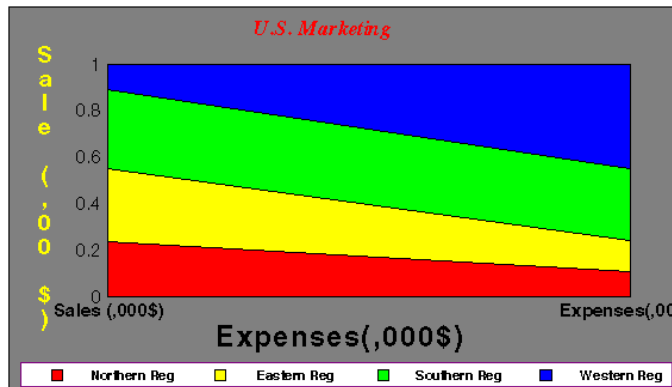
## Side by Side

The *side by side* minor type represents data as bars or risers side by side on the horizontal axis. The default graph type is a side by side bar graph, including several possible minor types. The following graph uses the kGRbars **major**type and kGRsideBySideDualYBiPolar **minor**type. Note that the y-axes have different scales.



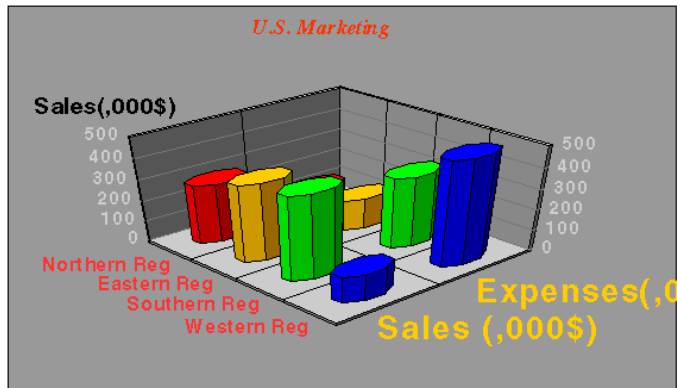
## Percent

The percent minor type displays each data point as a percentage of the whole group, on a scale from 0 to 1. The following graph uses the kGRarea **major**type and kGRpercent **minor**type.



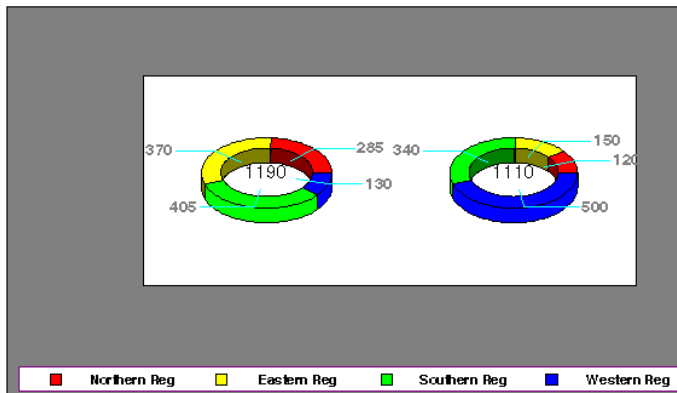
## 3D Graphs

You can specify several minor types for 3D graphs including Bars, Cubes, Row Area, Row Line, Column Line, Column Step, Honeycomb Surface, Model Surface, and so on. The following graph uses the kGR3D **major**type and kGR3Ddoctagon **minor**type.



## Pie

You can specify many different minor types for pie charts. For example, Ring pie, Multiple pie, and Multiple Ring pie. The following graph uses the kGRpie **major**type and kGRmultipleRingPie **minor**type. The numbers at the center of each ring indicate the cumulative totals for each group.

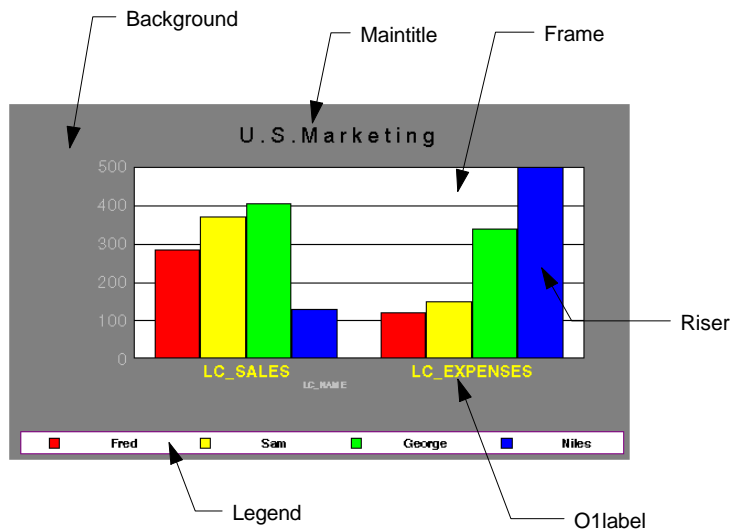


# Graph Objects

When you click on a graph in design mode the whole graph field is selected and you can view and change its properties in the Property Manager. However once you have set up the general properties of the graph field you may want to modify individual objects inside the graph. To do this you need to enable the **selectobject** property for the graph field under the Prefs tab in the Property Manager. When this property is enabled you can click on individual objects in the graph and view or change their properties, such as their color and text properties. When you click on an object inside your graph, a new tab may be added to the Property Manager which lets you view and change the properties of the selected object.

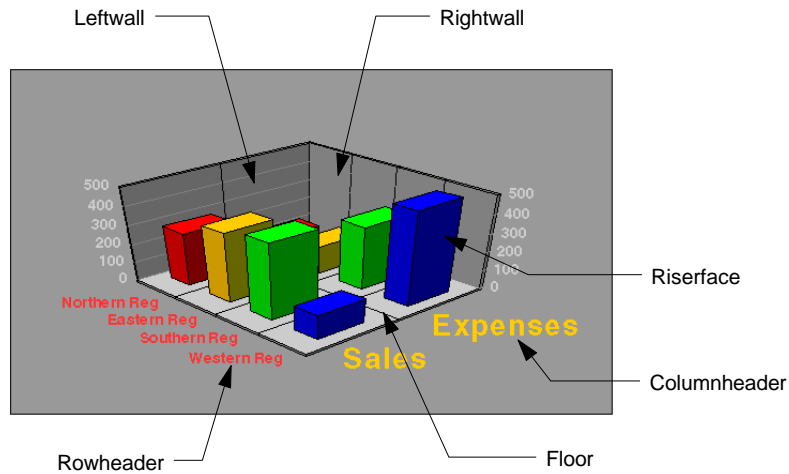
Some objects are common to all types of graph. For example, all graph types have a Title and Background object, and all except 3D graphs have a Frame. The following graphs show you some of the commonest objects for 2D, 3D, and Pie graphs.

## Common 2D Objects

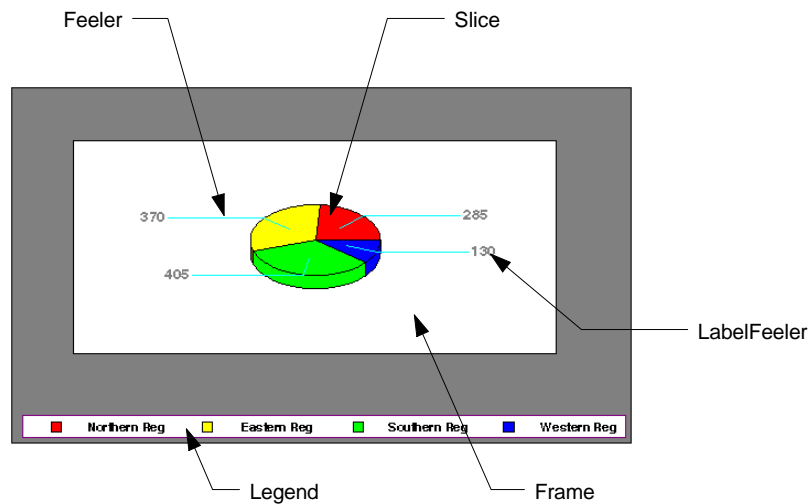




## Common 3D Objects



## Common Pie Objects



# Graph Properties

Graph properties define the appearance of each referenced object in a graph. The graph control has over 200 properties. Some are general and apply to most types of graph, while others apply to specific graph types or individual graph objects. Note that property names do not have a dollar sign when they appear in the Property Manager, but when you use them in the notation you need to include the dollar sign.

Most properties require only a single value. When you modify a graph in design mode you can type in a property value or select one from a droplist in the Property Manager. The values you set can be booleans, strings, numbers, and so on. Some properties require more than one value to define the appearance of the current object and these you enter as a comma-separated list.

## Graph Constants

Certain properties have enumerated values that OMNIS represents with constants. All the graph constants begin with the letters “kGR”. For example, to set the graph type to Ring pie you set the **major**type property to kGRpie and **minor**type to kGRringPie. In design mode, you can look up the Graph constants in the Catalog.

In addition, some properties take either a kTrue or kFalse value, usually to enable or disable functionality, or to show or hide an object.

## Showing Properties

When you first create a graph you can see only a few of its properties in the Property Manager. The **proplevels** property controls the level or visibility of the properties of a graph. By default this property is set to kGRlow, meaning you see only the lowest level or basic properties of a graph including its major and minor type. However, if you want to fully configure your graph or use some of the more advanced graph types, you will need to set **proplevels** to show more properties for the graph. In addition, when you write methods to change the graph at runtime, you can show the runtime only properties in the Property Manager using its context menu.

## High level Graph Prefs

When the **proplevels** property for the default bar graph is set to kGRhigh you get the following additional properties under the Prefs tab in the Property Manager, including the kGRmedium properties. These properties control the frequency, beginning and ending of the labels on the horizontal and vertical axes of the graph. Note that if you change the scales or axes of your graph all related objects including labels and risers are rescaled and redrawn.

<b>direction</b>	specifies the direction of the x-, y1-, or y2-axis; if enabled for an axis its default direction is reversed; for example if you enable direction for the y1-axis, values and labels will descend and the risers are displayed from the top of the frame
<b>excludezero</b>	excludes zero in the x-, y1-, or y2-axis; if enabled for an axis the base value of the scale is calculated from the data in your list, if disabled for an axis the scale begins at zero
<b>scalefreqo1</b>	the frequency of the group labels on the horizontal axis; for example, a value of 0 displays all labels, 1 displays every other label, 2 displays every third label, and so on
<b>scalefreqbego1</b>	the beginning label within a group of labels specified using scalefreqo1; for example, a value of 0 displays the first label, 1 displays the second label within each group of three when scalefreqo1 is set to 3
<b>scalefreqendo1</b>	specifies the last group label to be displayed; for example, a value of 0 displays all labels, 1 excludes the last label
<b>scaley1</b>	specifies the min and max values for the y1-axis or scales it automatically; takes the parameters ManualScale, ScaleMax, ScaleMin; ManualScale can be 0, 1, 2, or 3: zero means axis is scaled automatically (the default), 1 means the scale uses both ScaleMin and ScaleMax values, 2 means axis uses ScaleMin only, 3 means axis uses ScaleMax only; for example enter 1,600,0 to manually scale y1 axis from 0 to 600
<b>scaleendy</b>	specifies whether or not the min and max label on the vertical axis are displayed, a constant: kGRboth (the default), kGRminonly, kGRmaxonly, kGRnone
<b>scalefreqbegy1</b>	the beginning label within a group of labels on the y-axis specified using scalefreqy; for example, a value of 2 displays the second label within each group of three when scalefreqy is set to 3
<b>scalefreqendy1</b>	specifies the last y-axis label to be displayed; for example, a value of 0 displays all y-axis labels, 1 excludes the last label
<b>scalefreqy</b>	the frequency of the y-axis labels; for example, a value of 0 displays all labels, 1 displays every other label, 2 displays every third label, etc

<b>showzeroline</b>	shows or hides the zero line for the selected axes
<b>scalebase</b>	specifies whether risers are displayed from the zero line or the base of the frame, a constant: <code>kGRfromZero</code> (the default) or <code>kGRfromFrame</code> ; if set to frame, negative values are displayed from the base of the graph frame not above and below the zero line
<b>showoffscale</b>	specifies whether or not risers are shown if they are out of range on the specified axes; for example to hide risers with out of range values on the vertical axis, uncheck this property for the y1-axis

## High level Appearance Properties

When the **proplevels** property for the default bar graph is set to `kGRhigh` you get the following properties under the Appearance tab in the Property Manager, including the standard field properties. These properties control the overall appearance of the graph, its color, and grid settings.

<b>bargroupspacing</b>	spacing between the risers in each group, an integer between -100 and 100; for example, a value of 100 will put the maximum amount of space between risers available within the group
<b>barriserwidth</b>	affects the space between the groups and consequently the remaining space in the group in which the risers are drawn
<b>colorbyseries</b>	specifies whether the risers are colored by series ( <code>kTrue</code> , the default) or by group
<b>depthimsangle</b>	specifies the angle in degrees of pseudo 3D risers when <code>depthmode</code> is <code>kGRdepth3D</code>
<b>depthimsthick</b>	specifies the thickness or depth of pseudo 3D risers when <code>depthmode</code> is <code>kGRdepth3D</code>
<b>depthmode</b>	specifies whether a bar graph is drawn 2D ( <code>kGRdepthNone</code> , the default) or with psuedo 3D when <code>kGRdepth3D</code>
<b>gridlinesordo1</b>	specifies the major and minor grid lines on the o1-axis, takes the parameters <code>MajorHashStyle</code> , <code>MinorHashStyle</code> , <code>MinorHashCount</code> , <code>SuperHashStyle</code> , <code>SuperHashCount</code> ; the hash style parameters can be value 0 to 5 where 1 draws grid lines the height of frame, and 2 to 5 for various tick marks; hash count parameters specify the number of grid lines; for example, value 1,1,5,0,0 draws major and 5 minor grid lines on the o1-axis
<b>gridmodemajoro1</b>	controls the position of the major grid lines on the o1-axis, specify value 0, 1, or 2; value 1 draws major grid lines in center of groups
<b>gridmodeminoro1</b>	controls the position of the minor grid lines on the o1-axis, specify value 0, 1, or 2; value 1 draws minor grid lines at center of group
<b>gridlinesy1</b>	specifies the major and minor grid lines for the y1-axis, takes the parameters <code>LogScale</code> , <code>MajorHashManual</code> , <code>MinorHashManual</code> ,

	MajorHashStyle, MinorHashStyle, MajorHashCount, MinorHashCount; LogScale can be kFalse (the default) for linear scale or kTrue for logarithmic, the hash manual params can be kTrue to use manual settings otherwise kFalse for automatic settings, the hash style parameters can be value 0 to 5 where 1 draws grid lines the width of frame, and 2 to 5 for various tick marks; hash count parameters specify the number of grid lines; for example, value 0,1,1,1,1,5,5 draws major and 5 minor grid lines across the graph from the y1-axis
<b>orientation</b>	vertical (the default) or horizontal orientation of the graph
<b>shadowoffsets</b>	offset for riser shadows; positive values offset shadow to the right and up, or negative values for left and down; for example, 500,500 positions shadow to the top right of riser
<b>showshadows</b>	hides or shows riser shadows specified in shadowoffsets
<b>showlegend</b>	hides or shows the graph legend
<b>uniformqdrborders</b>	if true (the default) the same border style is used for all risers, otherwise if false you can set the border style for each riser
<b>squarelndicons</b>	if true (the default) legend markers are square, otherwise if false they are scaled to fit the available space in the legend
<b>uniformqdrshapes</b>	if false (the default) you can set the marker shape for each riser, otherwise if true the same shape is used for all risers
<b>legenditems</b>	the number of legend markers across the legend, or number of columns for vertical legendlayout
<b>legendlayout</b>	positioning of the legend markers and text in the graph legend, a constant: kGRauto (the default), kGRhorizontal, or kGRvertical
<b>seriesgroupswap</b>	changes the interpretation of your list data; if false (the default), list columns are interpreted in your graph as groups and rows become series, otherwise if true, list columns are represented as series and rows as groups; this does not alter the data in your list
<b>locatetitle</b>	positions the main graph title
<b>locatesubtitle</b>	sizes and positions the graph subtitle
<b>locatetitley1</b>	sizes and positions the y1-axis title
<b>locatefootnote</b>	sizes and positions the graph footnote
<b>locateframe</b>	sizes and positions the graph frame
<b>locatelegend</b>	sizes and positions the graph legend

## High level Text Properties

When the **proplevels** property for the default bar graph is set to **kGRhigh** you get the following properties under the Text tab in the Property Manager. These properties control the formatting and position of the o1- and y1-axis labels on your graph.

<b>labelmodeo1</b>	specifies whether the o1-axis group labels are spaced automatically or manually; <b>scalefreqo1</b> controls which group labels get skipped, <b>scalefreqbego1</b> controls which group label is drawn first
<b>labelwraplineso1</b>	the number of lines allowed for the o1-axis group labels
<b>labelwrapmodeo1</b>	if false, o1-axis group labels are single line only, otherwise if true, multi-line labels are allowed, set via <b>labelwraplineso1</b>
<b>placeo1</b>	positions the value and/or label for risers to the center, left, or right, a constant: <b>kGRcenter</b> (the default), <b>kGRoutMin</b> , <b>kGRinMin</b> , <b>kGRinMax</b> , <b>kGRoutMax</b>
<b>sideo1</b>	positions the o1-axis group labels for vertical graphs, a constant: <b>kGRbottomOrleft</b> (the default), <b>kGRtopOrright</b> , <b>kGRboth</b>
<b>staggero1</b>	specifies whether or not to stagger the labels on the o1-axis
<b>formaty1</b>	specifies the format of labels on the y1-axis, value between 0 and 46; for example, value 31 formats the label as \$OK
<b>formatdtxty1</b>	specifies the format of data text or labels on risers or markers, value between 0 and 46; for example, value 7 formats value as \$0, that is, places dollar or current currency sign in front of numeric data
<b>placey1</b>	positions the value and/or label for risers to the center, bottom, or top, a constant: <b>kGRcenter</b> , <b>kGRoutMin</b> , <b>kGRinMin</b> , <b>kGRinMax</b> , <b>kGRoutMax</b> (the default)
<b>sidey1</b>	positions the o1-axis group labels for horizontal graphs, a constant: <b>kGRbottomOrleft</b> (the default), <b>kGRtopOrright</b> , <b>kGRboth</b>
<b>staggery1</b>	specifies whether or not to stagger the labels on the y1-axis
<b>showdatatext</b>	shows or hides values and/or labels for risers or data points, a constant: <b>kGRtextNone</b> (the default), <b>kGRtextValue</b> , <b>kGRtextText</b> , <b>kGRtextBoth</b> , <b>kGRtextABS</b> (absolute value of segment on stacked)

# Data Format

The graph control interprets your list data and draws a bar graph by default, using the list column names as the group labels and the data in the first column to label the series. If the format of the data in your list is incompatible with the current graph type you may not get a graph at all, or at best the graph is drawn but may not represent your data correctly. The **dataformat** property specifies the data format of 2D and 3D graphs, and determines how the data in your list is interpreted and drawn in the graph. For some of the more complex graph types you may need to supply more data to plot a single data point, therefore you need to match the graph type with the correct data format for the graph.

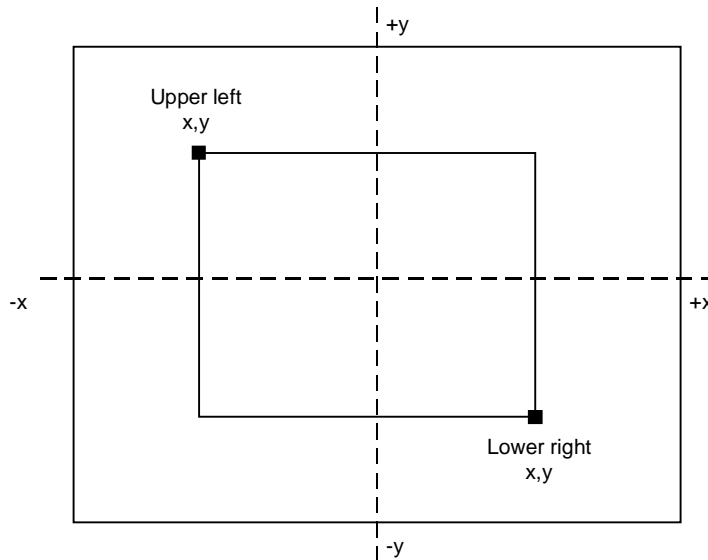
The default value for dataformat is 0 which is sufficient for simple or basic types of graph. In general, the graph control requires a single value to draw a riser or data point on your graph. For example, in the default bar graph each cell in your list is used to display a single riser in the graph. For scatter graphs (kGRspecial majortype and kGRscatter minortype), the graph control uses two values, an x and y value, to draw a single data point or marker. For stock market graphs (kGRspecial majortype and kGRhighLowOpenClose minortype), the graph control requires at least four values, a high, low, open, close value, to draw a single riser. The dataformat property is discussed in greater detail in the next chapter, and in the *Graph Property Reference*.

# Chapter 3—Modifying Your Graph

The *Graph Basics* chapter tells you how to create a simple graph and describes the general properties of a bar chart, the default graph type. This chapter discusses how you can further modify your graph, and in particular describes in more detail the advanced features of 3D, pie, and stock market graphs. Note that this chapter uses full property names that include the dollar sign, for example, the **dataformat** property is referred to as \$dataformat. To use many of the techniques described in this chapter you will need to enable the \$selectobject property in order to apply effects to specific objects, and switch \$proplevels to kGRhigh to show all the necessary properties.

## Sizing and Positioning Objects

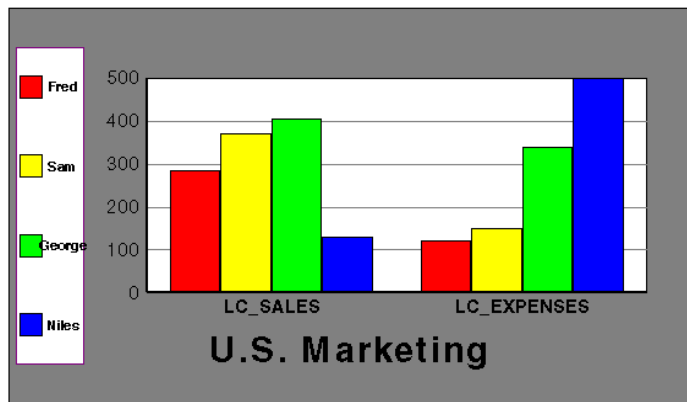
You can position and size the objects inside a graph, such as the main title, the legend, the graph footnote, and so on, using the \$locate... properties, found under the Appearance tab in the Property Manager. These properties take the parameters *upperleftx*, *upperlefty*, *lowerrightx*, *lowerrighty*, corresponding to the x,y position of the upper left and lower right corner of the object relative to the center of the graph background.





The parameters for the \$locate... properties are in a range from -16383 to 16383 for both the x and y value; these coordinates are virtual not absolute, which means all objects inside the graph will be resized when the whole graph is resized. Note that you cannot locate the graph Background object, but you can change its size and position either with the mouse or by changing the top, left, height, width properties of the graph field itself. Also note that all text objects are contained in an invisible box which you use to locate the object.

To position the graph legend to the left of the graph frame, for example, you can set \$legendlayout to kGRvertical and the \$locatelegend property to something like -15500,14000,-11000,-13000.

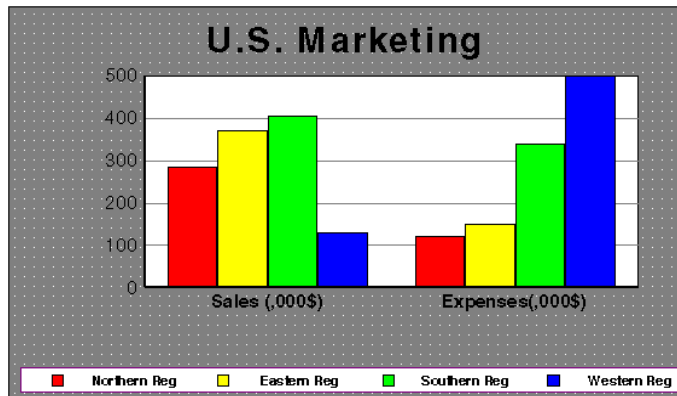


## Colors and Patterns

You can set the color and pattern of any object that has a definable area, including the graph frame, risers, the legend area, and so on. Such *area objects* have the following color, pattern and line properties.

<b>areabackcolor</b>	defines the backcolor of a patterned area object
<b>areacolor</b>	defines the forecolor of a patterned area object
<b>areapattern</b>	specifies the pattern to be used for the area object
<b>linecolor</b>	defines the color of a line object
<b>linestyle</b>	defines the style of a line object
<b>linewidth</b>	defines the width of a line object

To set the color of an area or line object, you use the color picker for the property. An object's color is stored as three values of range 0-255 for the red, green, and blue components. As well as colors, you can also use patterns to highlight different parts of a graph. By specifying \$areapattern for the Background, you could draw the following graph



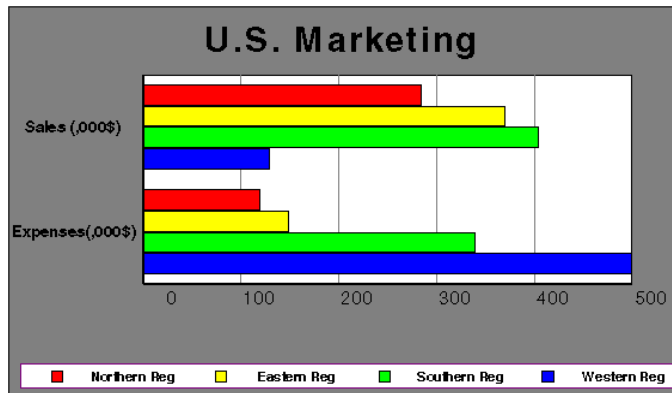
## Text Objects and Fonts

You can set the font properties of any text object under the Text tab in the Property Manager. Text objects have the following properties.

<b>fontalign</b>	aligns text in its bounding box, either left, center or right
<b>fontcolor</b>	defines the text color, an RGB value
<b>fontdropshadow</b>	adds a shadow to the text object; the first two parameters specify the offset of the shadow as an x,y virtual coordinate, the other three parameters specify the color of the shadow as an RGB value
<b>fontorient</b>	positions the text horizontally or vertically
<b>fontsize</b>	defines the font size in virtual coordinates
<b>fontstyle</b>	sets the font style of the text object

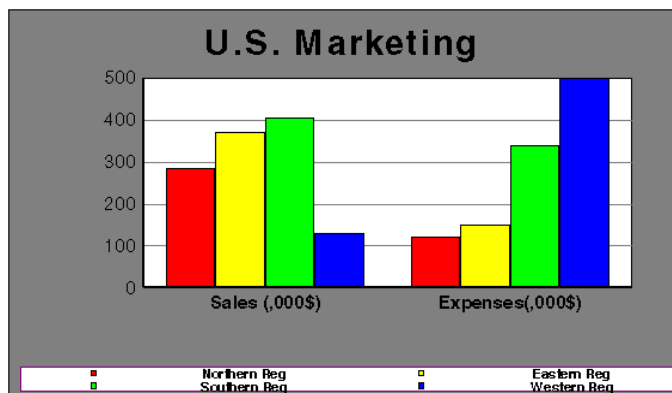
# Graph Orientation

By default all graphs are vertically aligned. You can change the orientation of bar, area, and spectral map graphs. You cannot change the orientation of line and 3D graphs. To change the orientation of a graph from vertical to horizontal, set \$orientation to kFalse.



## Legends

You can create a variety of different legends such as markers (the Legendmarker object) to the right of, left of, above, or below text or even text directly on top of markers with the properties \$insetlndtext, \$insetlndicon, and \$squarelndicons. By default the Legendarea contains the legends in a row at the bottom of the graph area. A graph displays the same number of legend markers as the number of series in your graph. \$legenditems lets you specify the number of legend entries per row or column if you display the legend set vertically. The legend text and icons or markers are fitted within the legend area automatically. A value of 2 gives the following graph.

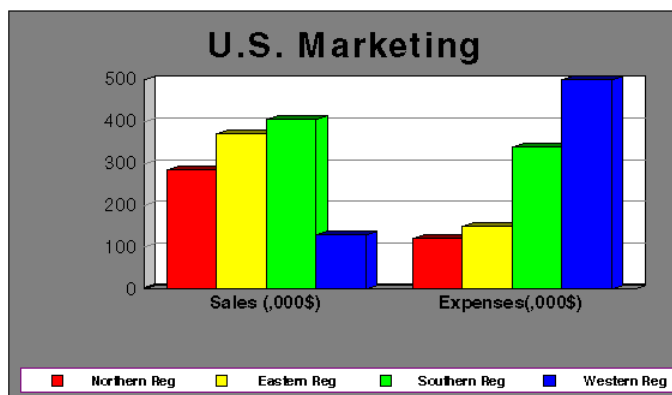


# Pseudo 3D

You can give a pseudo 3D effect to a 2D graph by giving the bars a depth or shadow effect using the following properties:

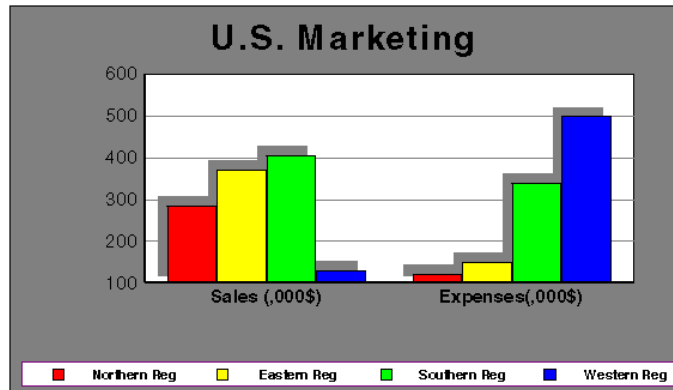
\$depthmode	sets pseudo 3D
\$depthimsangle	specifies the angle in degrees at which to apply the 3D effect: over 90 will give left perspective, 180+ gives underside view
\$depthimsthick	sets the thickness of the bars

Setting the angle to 30 degrees and thickness to 60 produces the following graph.



Alternatively, you can simulate a 3D effect on a 2D series object by dropping a shadow behind it at a given offset. You do this by showing the riser shadow object `Risersshadow` and applying `$shadowoffsets`, as follows

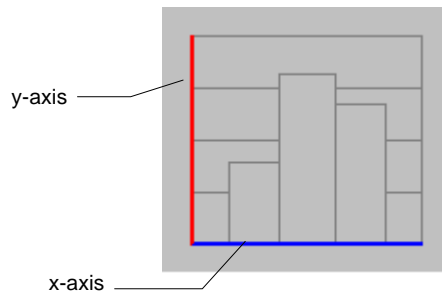
\$showshadows	kTrue
\$shadowoffsets	value1,value2, that is, the x- and y-coordinate for the offset. Positive values move the shadow up and to the right, negative values move the shadow down and to the left



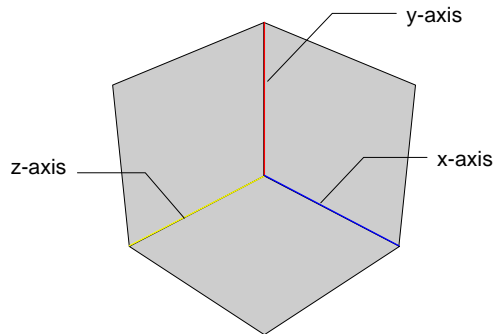
You can place a shadow behind any type of 2D bar, line, area, pie, scatter(XY), histogram, or hilo graph type. Only spectral map graphs cannot have a drop shadow.

# Axes

All 2D graphs have two axes: a horizontal x-axis and a vertical y-axis (bipolar graphs have a y1- and y2-axis). By default data values are plotted against the y-axis, and the graph categories are plotted against the x-axis.



3D graphs have three axes: an x-axis, a y-axis, and a z-axis.

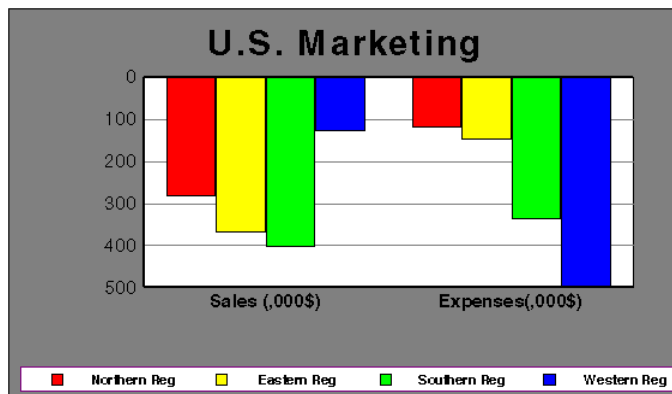


You can modify the direction of the axes on 2D graphs. Also you can stagger or alternate text labels on graph axes.

## Axis Direction

On 2D graphs the values ascend from left to right on the horizontal axis and from bottom to top on the vertical axis. However, you can use the property `$direction` to alter the default setting for each axis `x`, `y` and `y2`.

To turn your graph upside-down, set `$direction` to `kAxisY`. This gives the following graph:



## Axis Labels

You can create interesting effects by staggering or alternating the text on the axes of a graph. You can format the data labels on the axes in any one of more than thirty ways. The properties `$formatx`, `$formaty1`, and so on, control this feature for 2D and 3D graphs. Unless you specify the format, a graph uses the default setting type 1 whereby the number "one hundred thousand" appears as 100000.

## Data Values and Text

By default, a graph does not display data values and/or non-numeric text by a data point (a riser, a pie slice, marker, and so on). However, you can enable text and/or values using `$showdatatext`.

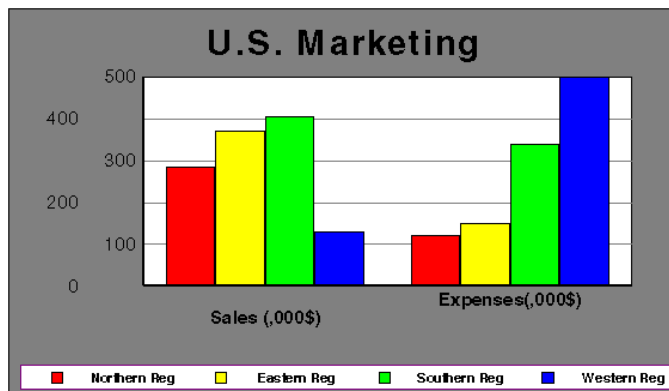
Once you enable such displays, you can control its appearance by operating on `$formatdtxtx`, `y1`, `y2` for the appropriate axis. You can also place the text with the `$placex`, `y1`, `y2` properties.

For example, setting `formatdtxtx [y1]` to 10 enables numeric data values for data points and formats the y-values as `$#,##0.00`.

## Staggered Labels

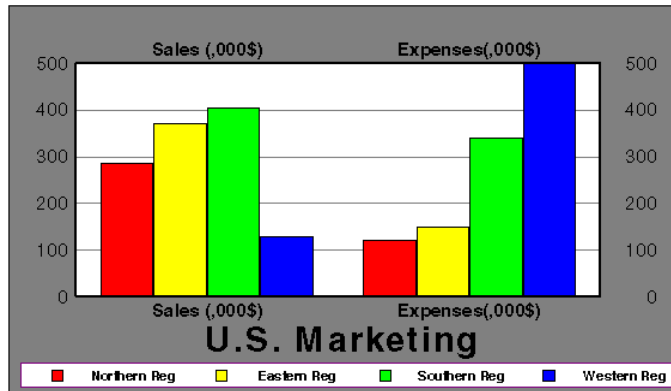
You can stagger the labels on any of the axes, that is, when enabled OMNIS draws the labels in a zig-zag fashion. The staggering for the x-, `y1`-, `y2`-, `o1`-axes is controlled by the `$stagger...` properties.

For example, setting `$staggero1` and `$stagger [y1]` to `kTrue` produces the following graph



## Adding Labels

You can show the axis text on a specific side of a graph (o1, o2, x, y1, and y2) using the \$side... properties. By default, axis text is drawn on the bottom or left (kLowSide), except for the y2 axis in which case it is not drawn (kNoImage). For example, to show the o1 and y1 labels on the top and right side of the graph (as well as the default bottom and left) you set \$sideo1 and \$sidey1 to kBothSides, which produces the following graph:



## Scaling

A graph uses the full range of values in your list to plot the scales. However, you can control the relative appearance of the risers on your graph by finely tuning the scales, through scaling the values and labeling the axes.

If you manually scale the graph, you can alter the range, display “out-of-range” values, and so on. You can use the \$scalex/y1/y2 property to turn off the autoscaling and scale the graphs as required. Each property, say \$scaley1, takes three values:

\$scaley1	ManualScale, ScaleMax, ScaleMin
-----------	---------------------------------

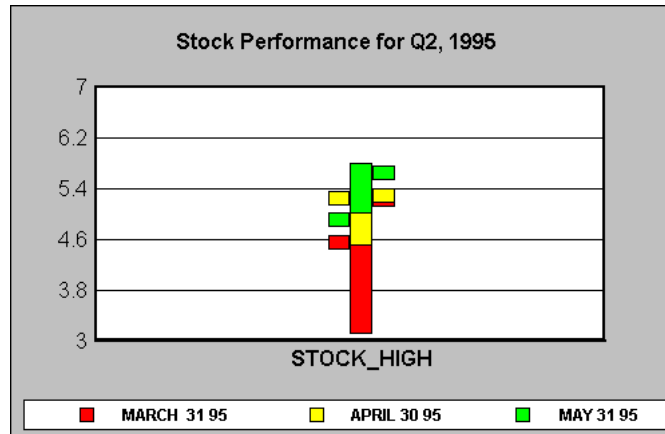
- ☐ **Manual Scale**  
whether to manually scale; if so, you supply the minimum and maximum values in the following two parameters
- ☐ **Maximum Manual Scale**  
the maximum value for the specified axis
- ☐ **Minimum Manual Scale**  
the minimum value for the specified axis

The following values set the maximum and minimum and scale to 7 and 3, respectively.



\$scaley1	1, 7, 3
-----------	---------

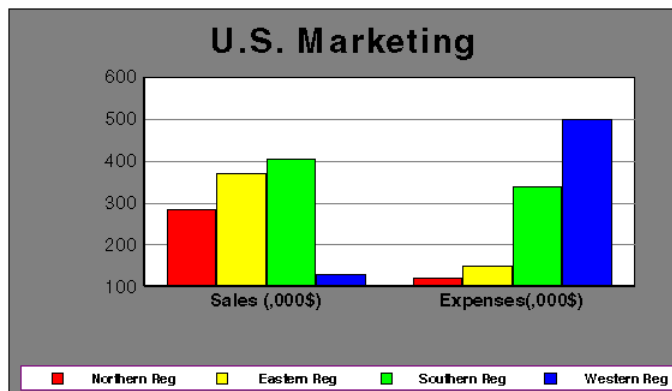
This produces the following graph; note the y1 scale.



## Scale Origin

Unless specified the scale origin is always zero. You enable the *\$excludezero* property by setting it to axes x, y and/or z to base the scale origin on the range of values you supply to the graph. (For a 2D graph, axes y and z denote y1 and y2.)

On the example graph, setting *\$excludezero* to kAxisY gives the following:



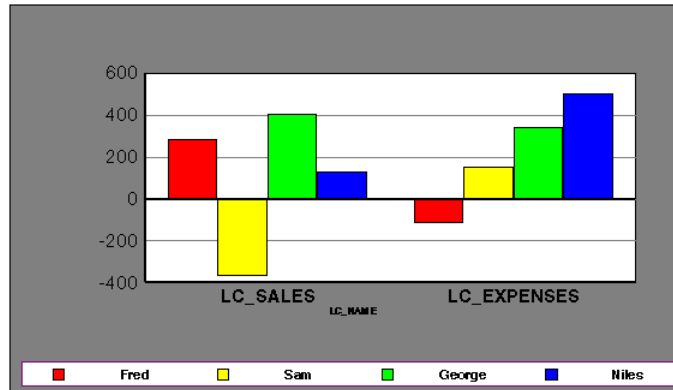
As you can see, the range for the vertical (y1) axis starts at 100. This reflects the fact that the lowest entry in the list is 120.

## Scale Base

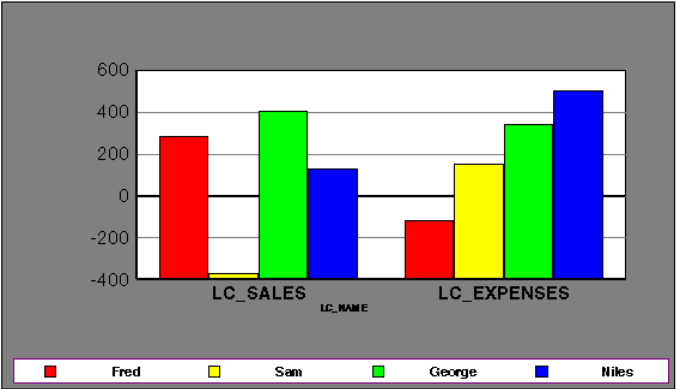
The risers of a graph are drawn above or below the y-axis baseline depending on whether the values are positive or negative. For example, given the following data

Field SALES_LST				F
	LC_NAME	LC_SALES	LC_EXPENSES	
1	Fred	285	-120	
2	Sam	-370	150	
3	George	405	340	
4	Niles	130	500	

the default graph would appear as



The `$scalebase` property controls the scale base in 2D and 3D graphs. If `kTrue`, bars are drawn from the zeroline as above; if `kFalse`, the default, they are drawn from the baseline like this:



# Labels

You can set the axis labels on your graph, that is, the start label, frequency of labels, and the end label. These labels affect the x, y1, and y2 axes only. The labeling does not affect the drawing of the grid lines.

By default, a graph displays all possible labels for the current type. To label the axes manually, you set the *\$scalefreqx/y1/y2* property. This property works in conjunction with the properties for the beginning and end labels, *\$scalefreqbegx/y1/y2* and *\$scalefreqendx/y1/y2*. For an entry *n* OMNIS divides the labels into *n+1* groups.

On 2D and 3D graphs you can specify whether Graphs should display the first and/or last values of a scale on the corresponding axis.

When shown as a scatter graph, the example graph has nine labels along the x axis and six labels along the y1 axis, but you can change this with the following property values for X1label and Y1label:

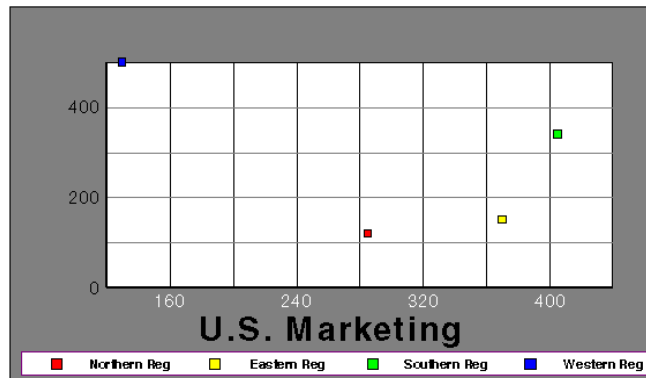
X1label

\$scalefreqx	1
\$scalefreqbegx	1
\$scalefreqendx	1

Y1label

\$scalefreqy1	1
\$scalefreqbegy1	0
\$scalefreqendy1	0

The following graph is drawn; note the scale frequency.



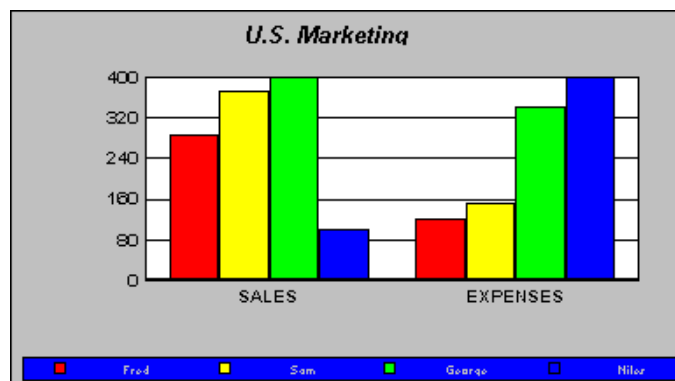
## Out of Range Values

Often you may need to consider the data that falls within a valid range, while alerting the viewer to “out-of-range” values. To set the required range values, you use the previously described `$scalex/y1/y2` property. You then enable the display of values that are out of range by clipping them to fit within the range using `$showoffscale [x, y, z]`. (For a 2D graph, `y` and `z` denote `y1` and `y2`.) When you apply the offscale property, Graphs draws values greater than a given range as either going off the scale or omits them completely.

For example, to set the scale manually and show the values that go off the scale, set

<code>\$scaley1</code>	1,0,400
<code>\$showoffscale</code>	<code>kAxisY</code>

George’s sales, and Niles’ expenses are now shown offscale.

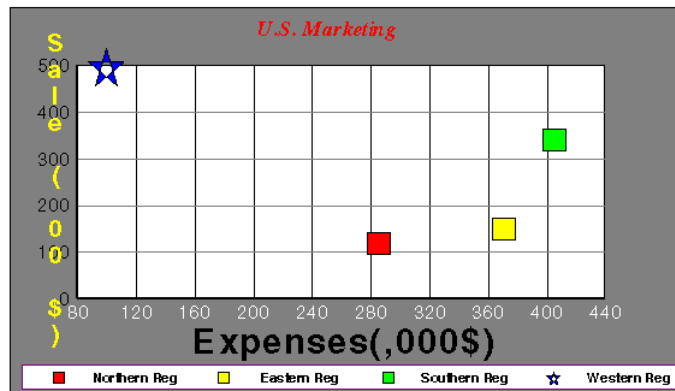


# Data Markers

You can specify the shape of the risers in bar graphs and the shape of markers in Line, Scatter, and Spectral Map graphs. Any change to the shape of the markers affects the legend icons that represent the markers. The properties that control the marker shapes are \$markershape, \$markersize, and \$uniformqdrshapes. They apply to Riser objects on bar graphs and Datamarker objects on line, scatter graphs and spectral map graphs.

The properties \$markershape and \$markersize control the shape and size of the markers on 3D Scatter graphs.

Setting markershape for the series 4 data point to kPentagram changes the marker shape for the Western region to a five-pointed star and enlarges the size of the markers on the graph. This produces the following graph; note the legend marker also changes.



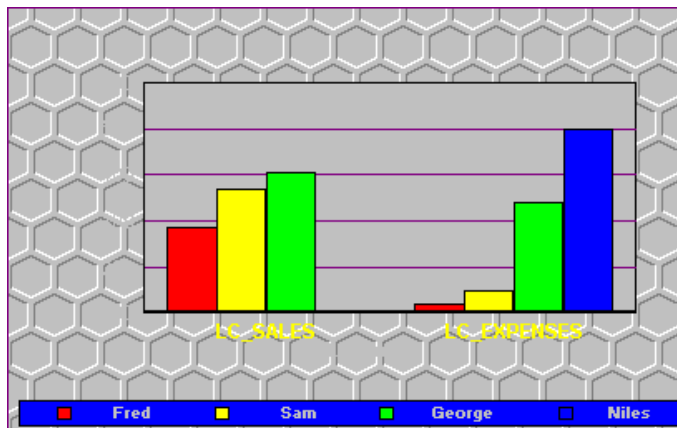
The *Graph Property Reference* section lists the possible marker shapes under \$markershape. You can view the marker shape constants in the Catalog.

# Pictures

You can add a bitmap picture to many area objects, including riser faces, the 3D graph wall, the graph frame, and background, using \$areaeffect and associated properties. You must give the full path and file name for the bitmap and specify whether the picture is scaled, tiled, or flipped.

For example, to place the “honey” windows bitmap on the graph background, select the background and set:

\$areaeffect	kGRpicture	set picture effect
\$pictname	c:\windows\ honey.bmp	the source bitmap
\$pictscale	kGRtiled	tiled
\$pictflip	kGRnofip	no flip



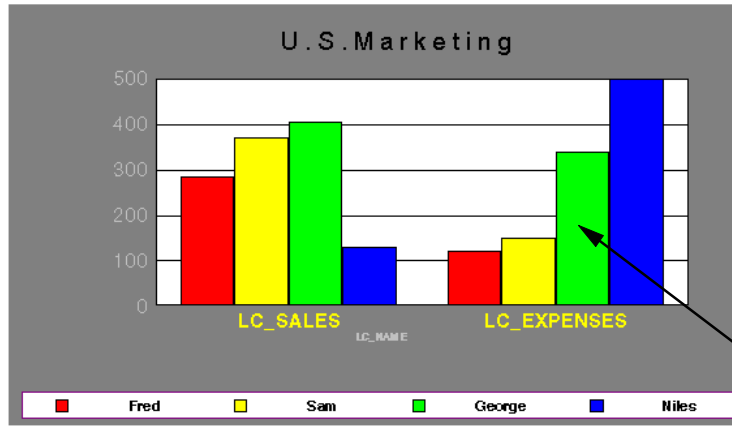
## Swapping the Series and Groups

The actual data point depends on the current grouping, whether you reference the list by rows or by columns or in OMNIS graph terms, by series or by groups.

By default, the column fields LC\_SALES and LC\_EXPENSES form two groups and the four rows form the series.

You can change the view of a list or change the manner of the graphic presentation organization without actually changing the structure of the list.

The following pictures show you how two graphs present the same data in different ways and highlight the position of a data point representing a single cell in your list.

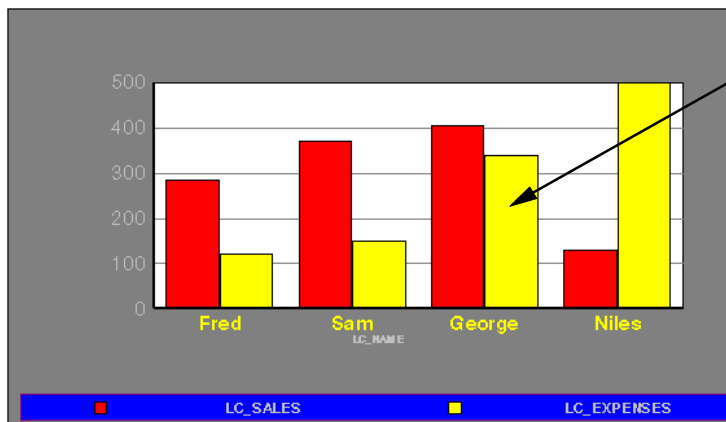


Series 3  
Group 2

Variable SALES_LST			
Options		View	
	LC_NAME	LC_SALES	LC_EXPENSES
1	Fred	285	120
2	Sam	370	150
3	George	405	340
4	Niles	130	500

SALES\_LST List

Series 2  
Group 3

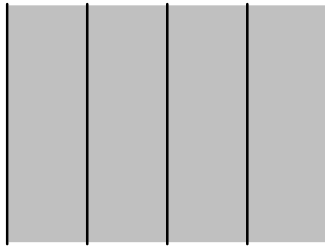


`$seriesgroupswap` controls whether to plot the graph by series (rows) or by groups (columns). It takes a Boolean value.

In the first graph, `$seriesgroupswap` was `kFalse` (the default). To obtain the orientation on the second graph, you set `$seriesgroupswap` to `kTrue`. As you can see in the second graph, the group names appear as the legend text and series names appear on the ordinal axis. The risers reflect the new groups.

## Grid Lines

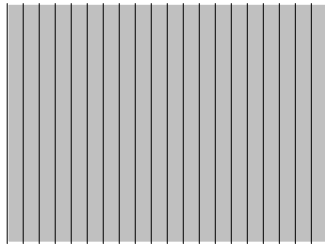
You can display grid lines for the x, y1, y2 and o1 axes. The vertical grid lines are x- and o1-axis grid lines. The horizontal grid lines are y1- and y2-axis grid lines. A graph has major and minor grid lines, and you can specify the number and style of the grid lines. The following table shows the orientation of the major and minor grid lines for 2D graphs.



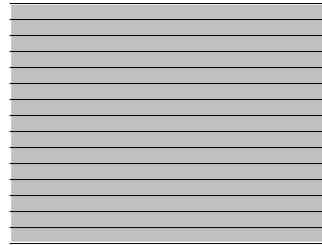
x-axis major grid lines



y-axis major grid lines



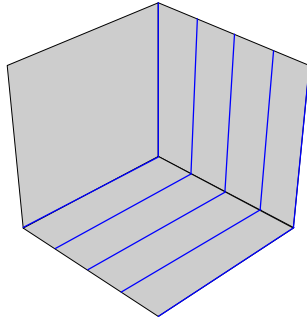
x-axis minor grid lines



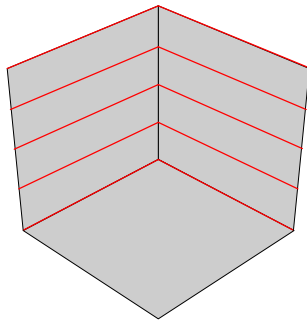
y-axis minor grid lines

By default, 2D graphs display the y-axis major grid lines only, scatter graphs display major grid lines for the x- and y-axis, and 3D graphs have x-, y-, and z-axis grid lines.

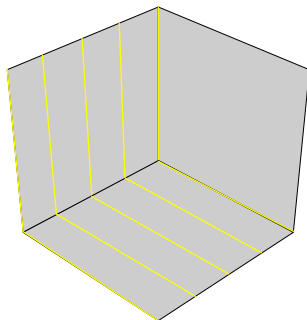




x-axis grid lines



y-axis grid lines

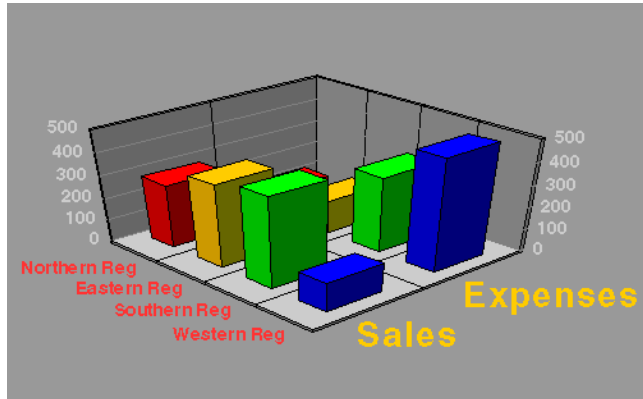


z-axis grid lines

You can control the x, y1, and y2 axis using the `$gridlinesx/y1/y2` properties.

# 3D Graphs

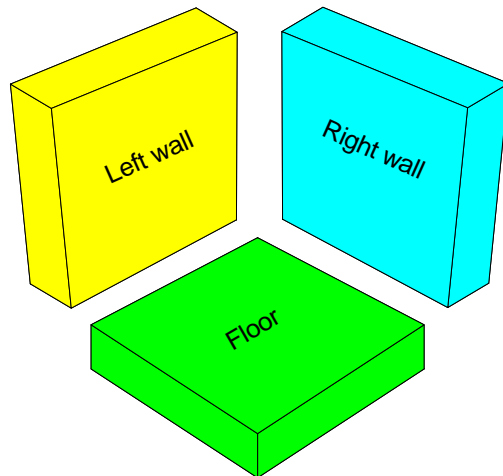
A three-dimensional graph provides a versatile way of displaying complex data.



The added features are wall and floor, riser faces, and properties that override the appearance of related 2D objects. A 3D graph has no Frame object. The grid lines on the walls and floor reflect the series and groups.

## Walls and Floor

The 3D objects Leftwall, Rightwall, and Floor represent the walls and floor of a 3D graph.

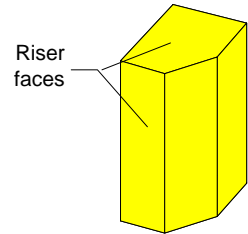


## 3D Riser Appearance

You can control whether individual series and groups can have their own appearance. The properties `$graphbywhat` and `$seriestype` control this feature.

## 3D Riser Faces

There are up to nine *riser faces*, `Risurface`, for each riser. You select these as you would any area object. For example, the `kCutCorner` riser object (shown right) has several visible faces.

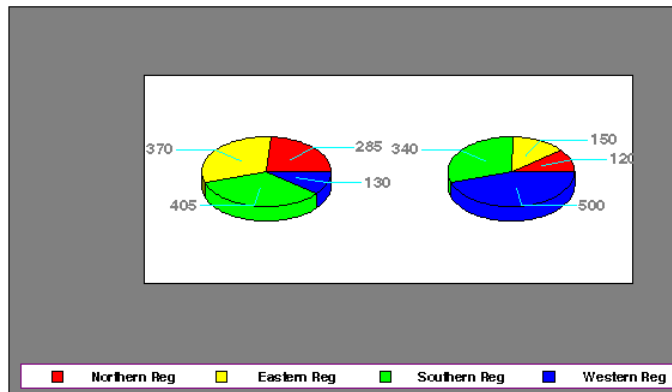


## Rotate and Zoom

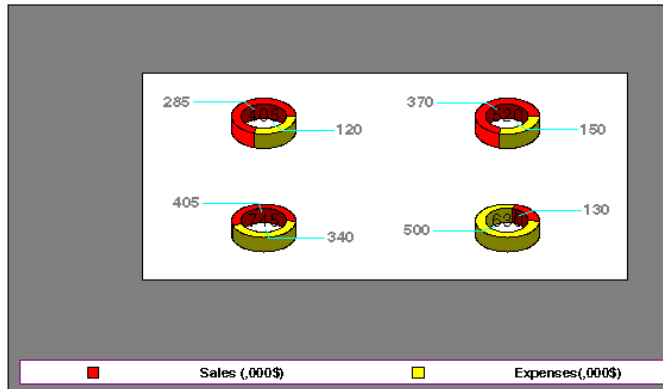
The runtime property `$customview` lets you zoom in and out, and change the viewing angle through different planes in graded steps.

# Pie Charts

To see the data represented as a Pie chart, you set `$graphmajor` to `kGraphPie`. The default minor type `$graphminor` is `kNormalPie`, showing a single pie, but choosing `kMultipleRingPie` displays two ring pies (assuming `$seriesgroupswap` is `kFalse`).

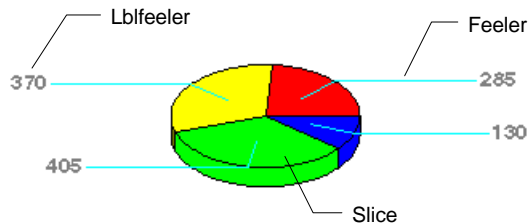


However, if you set `$seriesgroupswap` to `kTrue` you will get a ring pie for each data series.



## Pie Objects

The main objects in a Pie chart are



- ☐ **Slice**  
the values that together make up the Pie chart; each instance can be identified by a series and group ID (other related area objects are *slicering* when the pie is drawn as a ring, *slice crust*, which is the outer edge, and *slice face*, which is the inner edge of a pie slice, visible only if the you detach the slice)
- ☐ **Labels**  
on pie charts, the text that describes the graph, or slice
- ☐ **Feelers**  
the line that connects the slice to the corresponding label

## Orientation for Pie charts

Graphs by default draws the first slice of Pie charts at the “3 o’clock” position. It draws the pie with the additional slices moving in the counter-clockwise direction and the pie itself with a 40 degree tilt with respect to the horizontal axis. You can alter the way a Pie chart looks by starting the slice at a different position and varying the angle of tilt.

### Starting Position

`$rotate` determines from which position to draw the first pie slice. By default, Graphs draws the first slice at the 3 o’clock position and sets `$apirotate` to 0. Valid entries for this property range from 0 to 359.

### Drawing/Slicing Direction

`$drawclockwise` determines the direction in which to draw the pie slices. By default (`kFalse`) Graphs draws the slices in the counter-clockwise direction.

Under Windows you can change the direction of the slicing to be clockwise by setting `$drawclockwise` to `kTrue`.

Under MacOS you can only slice in *counter-clockwise* direction (the default case). `$drawclockwise` has no effect.

### Tilt

`$tilton` lets you tilt a Pie chart and `$tilt` sets the angle of tilt. By default, Graphs sets `$tilton` to `kTrue`, enabling you to tilt the graph. You can enter any integer from 0 to 89 to specify the amount of tilt. As you increase the tilt angle, the Pie chart appears to rotate backwards over the x-axis. A pie with 0 degrees tilt appears as a regular 2D pie. The default angle of tilt is 40 degrees.

### Depth

`$depth` controls the thickness of the pie crust. A minimum value of 0 (zero) yields a pie with no crust and the maximum entry of 300 yields a crust equal in thickness to the height of the frame for that graph.

## Pie Feelers

The callouts on a Pie chart consist of lines with text at the end. The line objects are Feelers, and the text object at the end of a feeler is a `Lblfeeler`. You can alter the visibility and type of the text objects. The property `$showfeeler` can only be set at runtime.

### Location

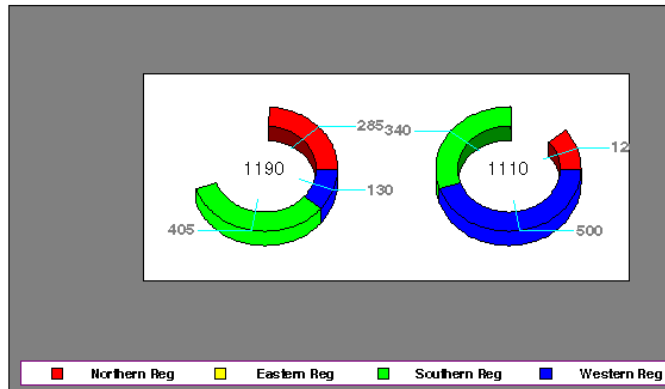
You can specify the starting position of a feeler, its position with reference to the center of the pie face, and the length of the feeler sections. The properties that do this are `$feelercenter`, `$feelerhorz` and `$feelerradial`, and can only be set at runtime.

## Special Pie charts

In order to look at the data from a different viewpoint by omitting a series, you can detach a slice from the others or remove it completely. The properties that do this are \$sliceddelete, \$slicemove and \$slicerestore and can only be set at runtime.

To remove the series 2 slice, for example, you set \$sliceddelete at runtime. This takes the parameters 'SeriesNumber, True/False'.

Do `$cwind.$objs.GraphField.$sliceddelete.$assign('2,1')`



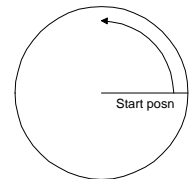
This graph has the following properties set.

\$graphmajor	kGraphPie
\$graphminor	kMultipleRingPie
\$tilton	kFalse

\$tilton is kFalse, that is, the default degree of tilt is turned off giving an aerial view.

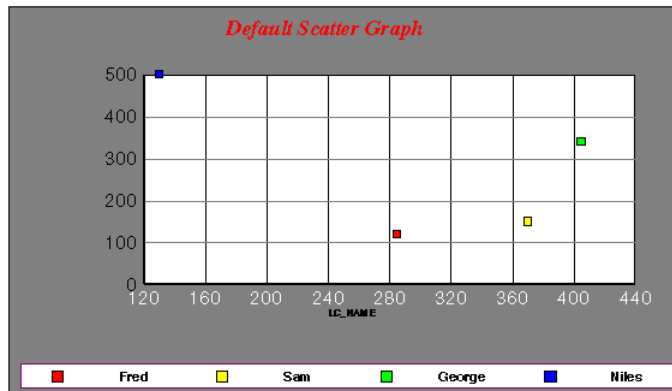
\$rotate defaults to zero, so the graph draws the first series (for the Northern region, in this case) from the 3 o'clock position in a counter-clockwise direction.

The pie slice for the second series (the Eastern region in this case) is deleted.



# Scatter Graphs

Graphs plots a scatter graph as a set of points, taking the first number in each row of the OMNIS list as the x-coordinate and the second number as the y-coordinate of a point. Thus Graphs plots the example data as follows.



Graphs automatically scales the axes and places the labels for the grid lines (note the x1 and y1 grid lines are drawn)

## Data Format for Special Graphs

If you are plotting a scatter graph, the values should be in the order (x1,y1), (x2,y2), and so on so you should build the OMNIS list accordingly. You need therefore a means whereby Graphs interprets the first value as the value for the horizontal axis and the second as that for the vertical axis.

Use the property `$dataformat` to pass a numeric value to Graphs. You can specify the number of values the graphs engine should expect from a data source to define a data point on a graph. When more than one value defines a data point, you specify the usage (or interpretation) of each value.

The number of values that you can use to define each data point and the interpretation of each value depends on the graph type. For example, if `$dataformat` is set to 1, the system expects two values to define each data point. If the graph type is a 2D Bar, Line, or Area graph, Graphs interprets the second value as a data label. If the graph type is a 2D X,Y scatter graph, Graphs interprets the first value as the x-axis value and the second as the y-axis value.

Once the graph is drawn correctly, you can enhance its appearance with the detailing features as described earlier.

If the selected data format is not compatible with the graph major and minor type, the resulting graph (if any) will not accurately reflect the data. Typically, only the graph background and frame are drawn when \$dataformat is not set properly.

# Histograms

Histograms show the number of occurrences of a value within the source data (the list). Each riser shows the cumulative frequency of a value or set of values. For example, in a class of students you could list the individual exam results and Graphs could plot the frequency of each score within a range of 0-100. Setting these properties produces the graph below.

\$graphmajor	kGraphSpecial
\$graphminor	kHistogram
\$scalex	1, 100, 0





# Stock Market Graphs

To plot stock market graphs, you need to supply details of the stock prices over a given time interval of your choice. You may plot the stock prices on a daily, weekly, monthly, quarterly, or annual basis. For the selected time interval you provide some or all of the following details on the stocks.

- ☐ **High value**  
a required value; the highest value for the stock during the time interval
- ☐ **Low value**  
a required value; the lowest value for the stock during the time interval
- ☐ **Open value**  
an optional value; the value for the stock at the start of the time interval
- ☐ **Close value**  
an optional value; the value for the stock at the end of the time interval
- ☐ **The time interval**  
this is optional; *if you include the time interval as part of the OMNIS list, it should be the first column in your list*

A stock market graph has the major graph type kGraphSpecial and either kHighLowOpenClose or kDualYHighLowOpenClose for the minor type.

Consider the following list definition and code.

```
; define variables GLIST, STOCK_DATE, STOCK_HIGH, STOCK_LOW,  
    STOCK_OPEN, STOCK_CLOSE
```

```
Set current list GLIST
```

```
Define list
```

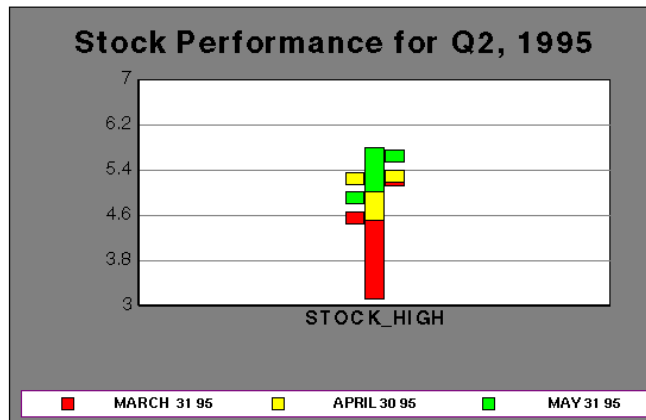
```
{ STOCK_DATE, STOCK_HIGH, STOCK_LOW, STOCK_OPEN, STOCK_CLOSE }
```

```
; then populate the list
```

Field GLIST						
	STOCK_DATE	STOCK_HIGH	STOCK_LOW	STOCK_OPEN	STOCK_CLOSE	F
1	MARCH 31 95	5.2	3.1	4.55	5.21	
2	APRIL 30 95	5.3	4.5	5.25	5.28	
3	MAY 31 95	5.8	5	4.9	5.65	

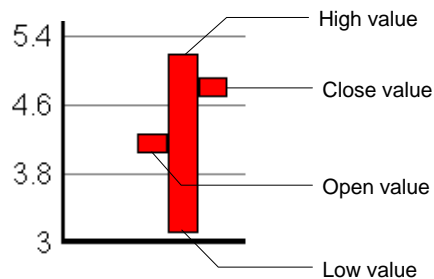
Setting these properties produces the graph below.

\$graphmajor	kGraphSpecial
\$graphminor	kHighLowOpenClose
\$dataformat	3
\$hlwidth	1000
\$socheight	1000
\$socwidth	1000
\$scaley1	1,7,3



To separate the risers for each month you should set \$seriesgroupswap to kTrue.

The risers for stock market graphs (Osmriser) indicate the high, low, open, and close values in the following way:



The small markers or “tick marks” on the left and right side of the riser indicate the open and close values, respectively. You can hide these tick marks by setting \$showopen and/or \$showclose to kFalse. If your list does not contain Open and Close data, Graphs will not draw the tick marks.

You can change the width of the risers using `$hlwidth` (default 700). Also, you can change the height and width of the open and close tick marks using `$ocheight` and `$ocwidth`, respectively (the default for both is 300).

## Data Format for Stock Market Graphs

To draw a stock market graph you need to inform Graphs of the format of the data in your list. You use `$dataformat` to set the data format for the different major and minor graph types; for `kSpecial`, `kHighLowOpenClose` graphs, that is, with High, Low, Open, and Close values in the list, `$dataformat` should be 3.

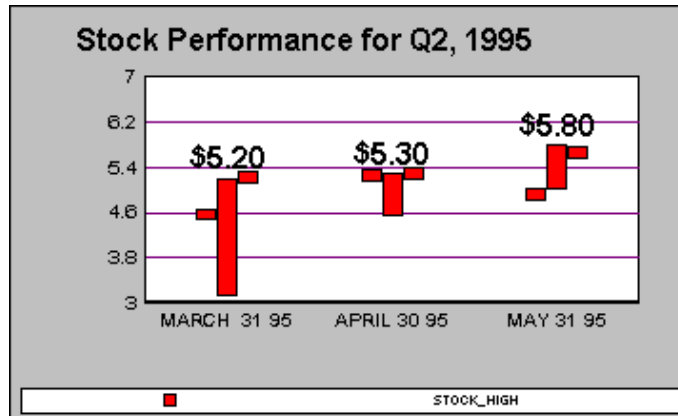
The following table shows the possible values for `$dataformat` that correspond to the structure of the data in your list.

<code>\$dataformat</code>	List Columns
1	High value, Low value
2	High value, Low value, Open value
3 or 4	High value, Low value, Open value, Close value
6	High value, Low value, Close value

## Text on Stock Market Graphs

The default stock market graph does not show labels against the hilo risers, but you can add them. For example, to display the high value make the following settings.

<code>\$seriesgroupswap</code>	<code>kTrue</code>
<code>\$showdatatext</code>	<code>kTextBoth</code>
<code>\$asmitemdtxt</code>	<code>kHigh</code>
<code>\$formatdtxty1</code>	10 (to display <code>\$##.##</code> )



Note that `$itemdtx` determines which value is shown on the hilo riser. It takes the following values

kGRhigh	Show High values
kGRlow	Show Low values
kGRopen	Show Open values
kGRclose	Show Close values

If you try to plot a stock market graph with a list that has one column and one row only, you will get a blank graph since there is insufficient data. You must set `$dataformat` to match your list.

## Changing a Graph at Runtime

You can set most properties of a graph at design time using the Property Manager, but you can also change a graph instance at runtime using the notation. Some properties apply only to the instance and can be set only at runtime. The runtime only properties are shown in the Property Manager. Properties affecting a particular series or group or changing a color can only be set at design time. `$canomit` and `$canassign` do not function with graph notation.

You can also handle the events for a graph to detect which part of the graph the user has clicked on, and take action accordingly.

You change a graph property at runtime using the `Do` command, just as for any other property. A single value can be a constant, but where there is more than one value to assign, they must be in a comma-separated string enclosed in quotes and you must use integers instead of constants.

For example \$sliceddelete takes the parameters SeriesNumber, True or False. To set series 2, kTrue:

```
Do $cwind.$objs.GraphField.$sliceddelete.$assign('2,1')
```

Some properties require you to set the parameters independently in the Property Manager. To set which axes are affected for example, kAxisNone denotes no axes, otherwise any or all of kAxisX, kAxisY and kAxisZ can be selected independently. To make these settings in notation, you enter a single value derived by treating each parameter as one bit of a three-bit binary number where each bit counting from the right is worth 1, 2, 4, and so on. For example, to set the X and Z axes use value 5.

## Saving your Graph to a File

To get your graph to look exactly how you want, you may need to set many of its properties. Having done this you might want to reproduce the same graph at other times or elsewhere in your application. The \$savetemplate(*pathname*) method lets you save all the properties of the current graph to a binary file, and the \$loadtemplate(*pathname*) method lets you load an existing template file. When you load a graph template all the properties saved in the file are applied to the current graph.

Both methods require a *pathname* to be passed to them, and both return a pathname of where the file was either loaded from or saved to. If an empty or incorrect *pathname* is passed, a prompt will appear, allowing the user to locate, or specify a file and path to be used. The new path is returned after the method has executed. For example

```
Do $cwind.$objs.GraphField.$savetemplate() Returns lpath  
; prompts the user for a file name and returns the path in lpath
```

## Drilldown

The event evGraphObjChange is sent when the user selects an object in the graph, and returns the name, series and group ID (if any) of the selected object.

pObjectname	string containing the name of the object
pSelectedSeries ...Group	series and/or group ID (if any)

evGraphObjChange returns an integer value for the series and group ID (if any). However, if the user clicks on a graph object, say the background, that does not have a series and/or group identifier, the event returns zero in the series and group ID parameters.

Your end users might need to examine a particular data point in greater detail. For example, you may want to see the *quarterly* figures for one particular data point from a graph containing the *yearly* figures: you can use the technique called drilldown to do this.

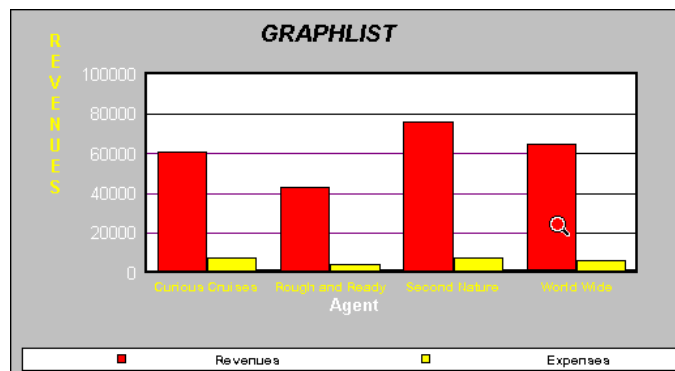
To implement drilldown you need to perform the following steps in your method(s):

- Get the series and/or group ID of the selected graph object with `evGraphObjChange`
- Locate the referenced line or cell in the data list using the series and/or group ID
- Build/load another list to use for the second graph
- Set the appropriate properties for the text objects for the second graph
- Position and/or redraw the second graph or open another window

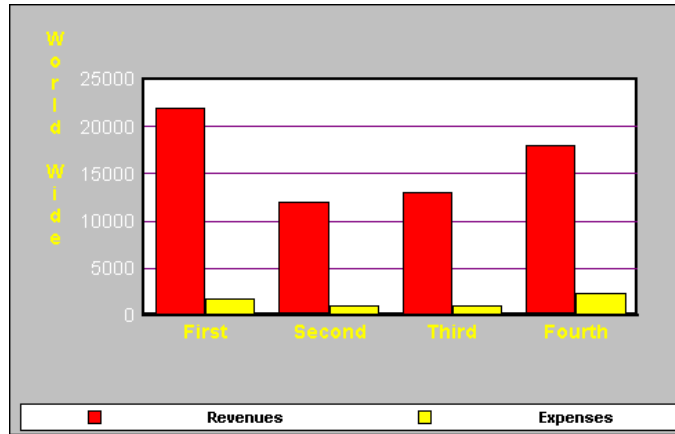
You can position the new graph in one of many ways:

- ☐ Draw the graph on a new window
- ☐ Hide the current graph and show the new one
- ☐ Show the new graph as an additional graph on the same window (using *Hide field* and *Show field* commands)
- ☐ Swap the lists, thus drawing the graph on the same graph control field in the old window

For example, the following code responds to a click on the graph and opens another window. The first graph window, `win_name1`, shows the yearly figures for different Travel Agents



The second window, `win_name2`, opens when you click on a riser in the first window, and shows the quarterly figures for the selected agent.



The method behind the graph field on the first window detects the click, gets the ID of the object clicked on, and calls a method in the second window

Method behind Graph

-----

On evGraphObjChange

```

    If pSelectedGroup > 0    ;; if riser selected
        Do method OpenWindow (pSelectedGroup)
            ; calls the second window, passes Group id
    End If

```

The method in the second window builds the list, opens the window, sets up the properties of the second graph, and redraws the graph field

```

; $construct wGraph
Set current list GRAPHLIST
Define list {Agent,cRevenues,cExpenses}
; Build list
Redraw lists

```

In addition, you can use the \$getobject() method to return the current selected object. The object returned using \$getobject() has the standard graph properties including.

- \$objectname  
the object name; same as returned in pObjectname
- \$series and \$group  
series or group id

On evGraphObjChange

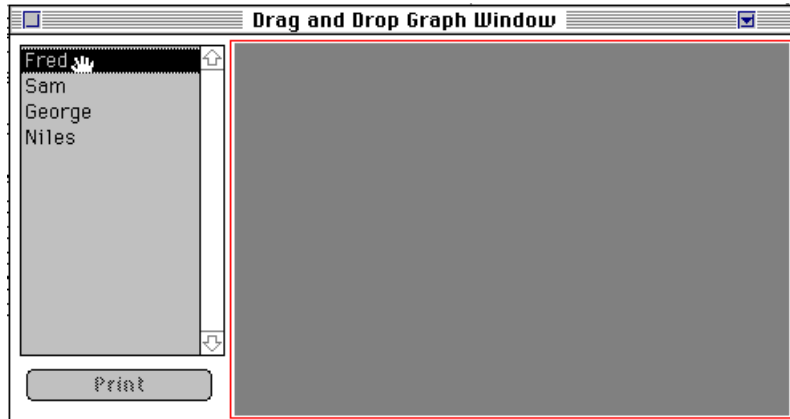
```

    Do $cinst.$objs.GRAPH.$getobject( pObjectname, pSelectedseries,
        pSelectedgroup ) Returns ivObjGraph ;; var of Object type
; now you could Do ivObjGraph.$rotate.$assign(180) ;; if pie

```

# Drag and Drop

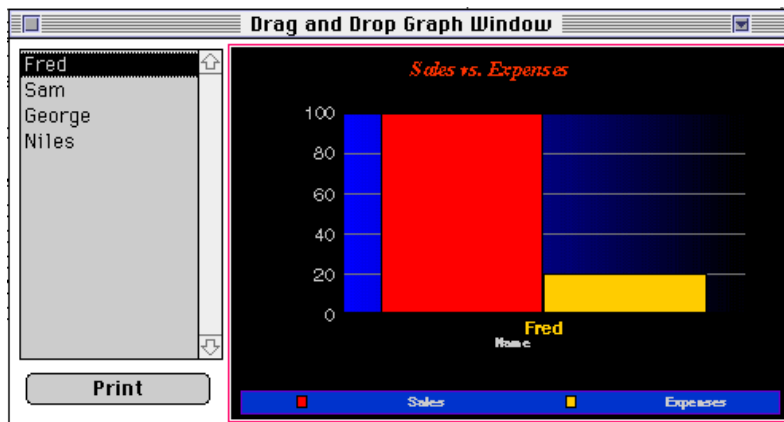
Drag and Drop lets the user select some data (usually from a list) and drag it onto a graph field in a window. The underlying field methods perform the necessary steps of creating a new list based on the selected data, populating this list and drawing the required graph.



In the window above, there is a list and graph field.

- ☐ **List**  
lets the user select the data
- ☐ **Graph field**  
the user drags the data onto this field

Using the example data, the following graph appears:



The graph field can have a method like:



```

On evDrop                                ;; a list line has been dropped
  Calculate iList as pDragValue ;; get the list line
  Calculate iGroup as List.$line
  Do method BuildList(iGroup)      ;; build the list for this group
  Do $cobj.$dataname.$assign(GraphList)
  Do $cobj.$xaxistitle.$assign([List.iGroup.SalesName])
  Redraw GraphField (Refresh now)

```

For information on drag and drop, see the *Window Programming* chapter in the *OMNIS Programming* manual.

## Redrawing 3d Graphs

3D graphs with special effects or ones that display the wire frame are sometimes slow to redraw; you can intercept the following events and report progress for the graph redraw.

- evGraphProgressBegin  
sent at the start of a long operation, such as redrawing a 3D graph with picture effects
- evGraphCalcProgress  
sent during any calculation loops to indicate the current status; pProgressStart, pProgressEnd, pProgress report the start, end, and current state of progress
- evGraphPaintStart  
sent at the end of any calculation process, but before the graph is re-painted
- evGraphProgressComplete  
sent at the end of the long operation; no parameters

The following method handles these events in a window containing a graph. The method is placed behind the graph field and the window has a status bar containing a single pane.

```

On evGraphProgressBegin
  Do $cwind.$statusbar.$panes.1.$text.$assign('Calculating
  Graph...')
On evGraphCalcProgress
  Do $cwind.$statusbar.$panes.1.$isprogress.$assign(kTrue)
  Do $cwind.$statusbar.$panes.1.$min.$assign(pProgressStart)
  Do $cwind.$statusbar.$panes.1.$max.$assign(pProgressEnd)
  Do $cwind.$statusbar.$panes.1.$value.$assign(pProgress)
On evGraphPaintStart
  Do $cwind.$statusbar.$panes.1.$text.$assign('Painting Graph...')
On evGraphProgressComplete
  Do $cwind.$statusbar.$panes.1.$text.$assign('Done')
  Do $cwind.$statusbar.$panes.1.$isprogress.$assign(kFalse)

```

# Chapter 4—Graph Property Reference

This chapter lists the graph properties in alphabetical order. Where appropriate it indicates if a property acts on a specific object, or selected series or group.

## Graph Properties

### **\$areabackcolor**

**Object:** Any area object.

**Description:** Defines the backcolor of a patterned area object such as a riser face. Each pattern, set using \$areapattern, has a backcolor and forecolor.

**Value:** RGB value.

### **\$areacolor**

**Object:** Any area object.

**Description:** This property defines the fore color of an area object. A pattern, set using \$areapattern, has a back color and fore color.

**Value:** RGB value.

## \$areaeffect

**Object:** Any area object.

**Description:** This property applies a bitmap or wash effect to an area object, such as the background, a riser face, or the floor of a 3D graph.

**Values:** kGRnoEffect (the default), kGRpicture, kGRwash.

**Notes:** Selecting kGRpicture shows the following properties:

\$pictscale	sets picture scaling
\$pictflip	sets the picture orientation
\$pictname	name and path of a BMP or PICT file

Selecting kGRwash shows the following properties:

\$washstart	the wash start color
\$washend	the wash end color
\$washtype	the wash form
\$washdir	the wash direction
\$washscale	wash scale (MacOS only)

## \$areapattern

**Object:** Any area object. Can affect a series only, where one is selected.

**Description:** This property specifies a pattern for an area object.

**Values:** Uses the default patterns.

## \$autoshaderisers

**Description:** Enables/disables autoshading of the 3D risers.

**Values:** kTrue to shade risers automatically, kFalse (the default) don't autoshade risers.

## \$bargroupspacing

**Description:** For bar graphs only, sets the spacing between all the groups of bars in a side-by-side bar graph. The larger the number, the greater the space between the bar groups; at 0 there is no space at all and all groups touch; at 100, the groups are as far apart as possible; all the bars are barely visible. At -100, the bars are stacked.

**Values:** -100 to 100

-100	Bars will overlap completely
0	No space between groups; risers will touch (the default)
100	Groups are as far apart as possible, in effect risers disappear

## \$barriserwidth

**Description:** For bar graphs only, sets the width of all the risers in a graph as a percentage of the space available for each group. This property interacts with \$bargroupspacing. Bar widths can only be adjusted as a percentage of what is left over after group spacing. A value of 0 percent gives no space between groups or maximum width of risers within group, a value of 99 percent gives minimum width of risers, and a value of 100 makes the risers disappear.

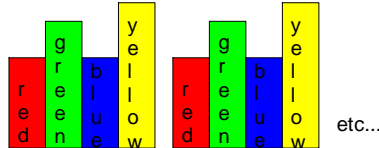
**Values:** 0 to 100 percent.

**Default Value:** 19 percent

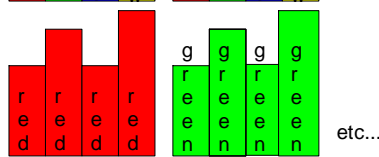
## \$colorbyseries

**Description:** This property determines whether the risers in a bar graph will be colored by series or by group. By default, risers are colored by series. This property affects all riser objects: it cannot be applied to a single series or group. You can use \$areacolor to specify the color of risers in the series or group.

\$colorbyseries = kTrue



\$colorbyseries = kFalse



**Values:** kTrue (the default) color by Series, or kFalse color by Group.

## \$colordivisions

**Description:** This property specifies the number of color divisions to be used by the \$colormodel property when the height color model is selected.

**Value:** Any positive integer (Short integer).

**Default Value:** 10

## \$colormodel

**Description:** This property specifies a coloring scheme for the risers in a 3D graph. For example, using the Height color model, the tallest risers have the palest color; the gradation of color depends on the setting of the \$colordivisions property.

**Values:**

kGRface	Color by riser face
kGRseries	Color by Series (the default); one color per series
kGRgroup	Color by Groups; one color per group
kGRangle	Color by Angle: color based on viewing angle (simulates directional light), most useful for surface graphs
kGRheight	Color by Height; highest risers have lightest color
kGRvalueX	Color depends on distance to floor (x-axis)
kGRvalueY	Color depends on distance to left wall (y-axis)
kGRvalueZ	Color depends on distance to right wall (z-axis)

## \$curvecolorasseries

**Usage:** This property can be set in runtime only.

**Description:** This property lets you assign a color to individual curve lines. You can set this property for a series and color the line independently, otherwise curve lines are the same color as their associated series.

**Values:** kTrue (the default) curve is the same color as the area portion of the series from which it is derived, or kFalse lets you color curve independently.

## \$curvemoving

**Usage:** This property can be set in runtime only.

**Description:** This property specifies the number of data points/groups to use when generating a moving average curve. It affects all Movavglines or moving average line objects in the graph. If you specify a number of points greater than the number of groups in the graph, the line is not drawn. Use the \$sdlinemova property to enable/draw this line. Use the \$scimovavg property to identify the moving averaging line type (scientific or financial).

**Values:** 0 to number of groups in the graph.

0	No points and no curve is drawn
1	Only one point will be used to generate the moving average curve; effectively no curve is drawn
2 to number of groups	Specifies the number of points/groups across which the moving average curve is drawn (scientific or financial type)

**Default Value:** Number of groups in the graph

## \$curvesmooth

**Usage:** This property can be set in runtime only.

**Description:** This property specifies the number of points used to generate a smooth curve. It affects the appearance of all series associated lines, including Linexp, Linrline, Curve. If you set this property to a value less than zero, the system will reset it to the default value of 100.

**Values:** Any positive integer value. Larger values draw a smoother curve.

## \$customview

**Usage:** This property can be set in runtime only.

**Description:** This property controls the viewing angle of a 3D graph. Making repeated assignments to this property rotates the graph dynamically.

**Parameters:** View mode, step size (Short integers).

**Value Ranges:** Step size 0 to 34:

0	rotate on X-axis counterclockwise
1	rotate on Y-axis counterclockwise
2	rotate on Z-axis counterclockwise
3	rotate on X-axis clockwise
4	rotate on Y-axis clockwise
5	rotate on Z-axis clockwise
6	move along X towards 3D origin point
7	move along Y towards 3D origin point
8	move along Z towards 3D origin point
9	move along X away from 3D origin point
10	move along Y away from 3D origin point
11	move along Z away from 3D origin point
12	decrease X cube dimension (cube width)
13	decrease Y cube dimension (cube width)
14	decrease Z cube dimension (cube width)
15	increase X cube dimension (cube width)
16	increase Y cube dimension (cube width)
17	increase Z cube dimension (cube width)
18	decrease left wall thickness
19	decrease floor thickness
20	decrease right wall thickness
21	increase left wall thickness
22	increase floor thickness
23	decrease right wall thickness
24	pan graph cube to left on screen (in 2D)
25	pan graph cube upwards on screen (in 2D)



26	turn relative rotation/movement ON (relative to graph axes)
27	pan graph cube to right on screen (in 2D)
28	pan graph cube downwards on screen (in 2D)
29	turn relative rotation/movement OFF (absolute: not relative to graph axes)
30	zoom into graph
31	increase perspective distortion on graph
32	unused
33	zoom away from graph
34	decrease perspective distortion on graph

**Example:** To rotate the graph on the Y-axis clockwise in steps of 10, set \$customview to 4,10.

## \$dataformat

**Description:** This property specifies the format of the data required to draw a graph of a certain type. The data format represents the number of values and/or labels a graph requires from your list to draw a single data point on 2D and 3D graphs; it is more important to set this property for Special graph types since their data points require multiple values. For example, to draw a simple riser you only require one data value, whereas to draw a label with the riser you need to supply the data value and label text. The number of values required to draw each data point and the interpretation of each value depends on the graph type and data format, as shown in the table below.

**WARNING:** If the data format is not compatible with the type of graph or the structure of your list, the resulting graph (if any) will not accurately reflect your data. Typically, the graph will be blank if \$dataformat is not set properly. When you change the \$majortype and \$minortype of a graph you may also need to change \$dataformat.

### Values:

Major / Minor graph type	Value and Description	
kGRarea, kGRbars, kGRlines, and all minor types	0	Value
	1	Value, Label
	3	Value, Label, Low error value, High error value
	5	Value, Low error value, High error value
kGRpie and all minor types	0	Value
kGRspecial / kGRhistogram	0	Value
kGRspecial / kGRspectralMap	0	Value
	1	Value, Label
kGRspecial / kGRscatter and kGRscatterDualY	1	x value, y value
	2	x value, y value, Label
	4	x value, y value, Label, Low error value, High error value
	6	x value, y value, Low error value, High error value

Major / Minor graph type	Value and Description	
kGRspecial / kGRhighLowOpenClose and kGRdualYHighLowOpenClose	1	High value, Low value
	2	High value, Low value, Open value
	3 or 4	High value, Low value, Open value, Close value
	6	High value, Low value, Close value
kGR3D and all minor types, except k3DScatter	0	Value
kGR3D / kGR3Dscatter	2	X value, Y value, Z value
	3	X value, Y value, Z value, Label

**Default Value:** 0 for all graph types.

**Notes:**

1. You can use \$showdatatext to show the values and labels associated with a data point on a 2D graph. Use \$formatdtxtx/y1/y2 to specify the format in which the values are displayed.
2. You can use \$showtextdatalabels to show the labels associated with a data point in a 3D scatter graph when \$dataformat value 3 is used. Data text values cannot be displayed in a 3D graph.

## \$dataname

**Description:** Specifies the name of the list variable for a graph.

## \$depth

**Description:** This property controls the depth of the pie, that is, thickness of the crust. A value of zero creates a pie with no crust. A value of 300 creates a crust thickness equal to the height of the frame.

**Values:** 0–300 (Long integer); a value between 10 and 50 is recommended.

**Default Value:** 30

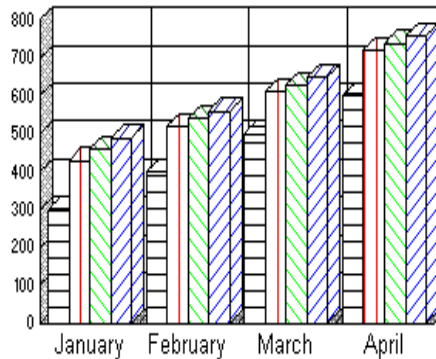
## \$depthimsangle

**Description:** Specifies the angle in degrees of the pseudo 3D effect on a 2D graph when the \$depthmode property is set to kGRdepth3D.

**Values:** 0–360 degrees

**Default Value:** 45 degrees

**Example:** The following bar graph has \$depthmode set to kGRdepth3D and the default values for \$depthimsangle and \$depthimsthick.



## \$depthimsthick

**Description:** Specifies the depth of a 3D effect applied to a 2D graph by \$depthmode. The \$depthmode property must be set to kGRdepth3D in order for this property to have any effect on the graph.

**Values:** 0 to 100 (the default).

## \$depthmode

**Description:** This property gives a 2D graph a pseudo 3D appearance by adding depth to the background and risers. \$depthimsthick specifies the depth.

**Values:** kGRdepthNone (the default) for normal 2D chart, kGRdepth3D to add depth to a normal 2D graph.

## \$designgroups

**Description:** The number of groups displayed in your graph in design mode. For example, you can set this property to the number of data columns in your list (ignoring label columns) to get an idea of how your graph will look at runtime.

**Value:** 1 to an appropriate number of groups (Long integer).

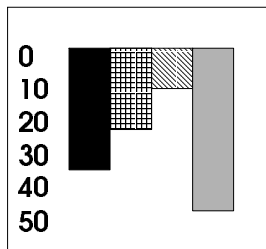
## \$designseries

**Description:** The number of series, that is risers or data markers, in each group displayed in your graph in design mode. For example, you can set this property to the approximate number of rows in your list to get an idea of how your graph will look at runtime.

**Value:** 1 to an appropriate number of series (Long integer).

## \$direction

**Description:** This property reverses the direction of an axis. For example, data is plotted from bottom to top on the y-axis, but when this property is enabled for the y-axis values are reversed and drawn bottom-to-top. Note that this property also reverses the order of the axis labels and changes all data-related objects, such as markers. \$direction for x only affects Scatter graphs. For example, \$direction is enabled for kGRaxisY1 in the following graph.



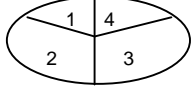
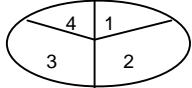
**Values:** kGRnoAxis (the default), kGRaxisX, kGRaxisY1, kGRaxisY2.

## \$drawclockwise

**Note:** Not implemented under MacOS.

**Description:** The slices of a pie can be drawn either clockwise or counter-clockwise. This property determines the direction pie slices are drawn beginning from the rotation angle defined by \$rotate. Slices are drawn counter-clockwise by default.

**Values:**

Value	Description	Example
kFalse	Slices are drawn counter clockwise from the starting angle (the default). The example assumes \$rotate is 90 degrees	
kTrue	Slices are drawn clockwise from the starting angle. The example assumes \$rotate is 90 degrees	

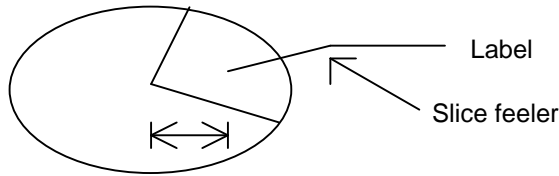
## \$excludezero

**Description:** This property excludes zero from a scale and draws it based on the range of data in the graph. If \$excludezero is disabled for an axis the scale will always include zero regardless of the range of your data. You can enable this property for a scale and use the \$scale... properties to set the scale manually for an axis.

**Values:** kGRnoAxis (the default), kGRaxisX, kGRaxisY1, kGRaxisY2.

## \$feelercenter

- Usage:** This property can be set in runtime only.
- Object:** Feeler. Select the series.
- Description:** This property sets the distance from the center of the pie to the start of the pie feeler. The distance is a percentage of the radius of the pie. 100% would cause the feeler to begin at the edge of the pie.
- Values:** 0–100 percent of pie radius (Short integer)
- Default Value:** 50 (percent, as shown)

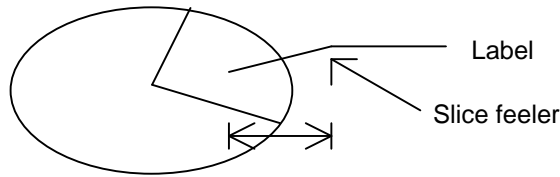


## \$feelerhorz

- Usage:** This property can be set in runtime only.
- Object:** Feeler. Select the series.
- Description:** Determines the label feeler horizontal length, from the end of the radial to the label text. The distance is measured as a percentage of the distance from the center of the pie to the edge of the frame. A value of zero hides the horizontal line on the label feeler.
- Values:** 0–100 percent (Short integer).

## \$feelerradial

- Usage:** This property can be set in runtime only.
- Object:** Feeler. Select the series.
- Description:** This property sets the radial distance of the feeler, that is, the distance from start of feeler set by \$feelercenter to the horizontal portion of the feeler. The distance is a percentage of the radius of the pie. Therefore, a value of 100% would cause the radial to be the same length as the radius of the pie.



- Values:** 0–100 percent of the pie radius.
- Default Value::** 50 percent (Short integer).

## \$font

- Object:** Any text object.
- Description:** Specifies the font for a text object.

## \$fontalign

- Description:** Aligns text in its bounding box. The text can be left, right, center, or fully-justified.
- Values:** kGRjusLeft, kGRjusCenter, kGRjusRight.

## \$fontcolor

- Description:** Specifies the color of the font for a text object.
- Values:** RGB value



## \$fontdropshadow

**Parameters:** x and y coords of shadow, and red, green, blue value of shadow color.

**Description:** This property adds a shadow to a text object; the first two parameters specify the offset of the shadow as an x,y virtual coordinate, the other three parameters specify the color of the shadow as an RGB value.

**Default Value:** 0,0,0,0,0 (no drop shadow).

## \$fontorient

**Description:** Defines the orientation of a text object; does not apply to some text objects.

**Values:** kGRhorizontal, or kGRvertical.

## \$fontsize

**Description:** Defines the size of the font for a text object in virtual coordinates.

**Values:** Any positive long integer.

## \$fontstyle

**Object:** Any text object. Can affect a series only, where one is selected.

**Description:** Defines the font style for a text object.

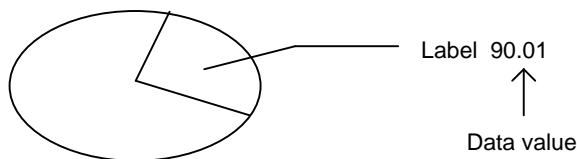
**Values:** kBold, kItalic, kUnderline (all off for Normal).

## \$footnote

**Description:** The graph footnote title, located in the bottom right-hand corner of the graph by default. You can change its position using the \$locatefootnote property.

# \$formatdatatext

**Description:** Sets the format of pie feeler labels. Changes you make here are also seen in labels for other graph types.



**Values:** 1 (the default) to 46. See \$formatdtxty1.

## \$formatdtxtx

## \$formatdtxty1

## \$formatdtxty2

**Object:** Datatext

**Parameters:** Short integer

**Description:** Specifies the format of the numeric data text for data points, displayed on risers or markers using \$showdatatext. The \$placeo1/o2 and \$placex/y1/y2 property control the placement of the values and/or labels on the risers.

**Values:** 1 (the default) to 46 (Short integer)

The following samples use one million (1,000,000) as the value to display.

Value	Format	Sample
1	General	1000000
2	0	1000000
3	0.0	1000000.0
4	0.00	1000000.00
5	#,##0	1,000,000
6	#,##0.00	1,000,000.00
7	\$0	\$1000000
8	\$0.00	\$1000000.00
9	\$#,##0	\$1,000,000
10	\$#,##0.00	\$1,000,000.00
11	0%	100000000%

Value	Format	Sample
12	0.0%	100000000.0%
13	0.00%	100000000.00%
14	%0	%100000000
15	%0.0	%100000000.0
16	%0.00	%100000000.00
17	0.0E+00	1.0E+6
18	0.00E+00	1.00E+6
19	0.000E+00	1.000E+6
20	0.0e+00	1.0e+6
21	0.00e+00	1.00e+6
22	0.000e+00	1.000e+6
23	0K	1000K
24	0.00K	1000.00K
25	#,##0K	1,000K
26	#,##0.00K	1,000.00K
27	K0	K1000
28	K0.00	K1000.00
29	K#,##0	K1,000
30	K#,##0.00	K1,000.00
31	\$0K	\$1000K
32	\$0.00K	\$1000.00K
33	\$#,##0K	\$1,000K
34	\$#,##0.00K	\$1,000.00K

The following samples use one billion (1,000,000,000) as the value to display.

Value	Format	Sample
35	0M	1000M
36	0.00M	1000.00M
37	#,##0M	1,000M
38	#,##0.00M	1,000.00M
39	M0	M1000
40	M0.00	M1000.00

41	M#,##0	M1,000
42	M#,##0.00	M1,000.00
43	\$0M	\$1000M
44	\$0.00M	\$1000.00M
45	\$\$,##0M	\$1,000M
46	\$\$,##0.00M	\$1,000.00M

## \$formatringtext

**Description:** This property sets the format of the text at the center of a ring pie.

**Values:** 1 (the default) to 46. See \$formatdtxty1.

## \$formatsdline

**Object:** Linrtext. Select the series.

**Usage:** This property can be set in runtime only.

**Description:** This property controls the format of values in the formula and correction coefficient strings that are displayed in associated with series-dependent lines. Note the formula/correction coefficient text must be enabled with \$sdlinelinrcorr or \$sdlinelinrformula.

**Values:** 1 to 46. See \$formatdtxty1.

**Default Value:** 19

## \$formatx

## \$formaty1

## \$formaty2

## \$formatz

**Object:** X1label, Y1label, Y2label, Zlabel.

**Description:** Specifies the format of data labels on the x, y1, y2, z axis.

**Values:** 1 to 46. See \$formatdtxty1.

## \$graphbywhat

**Description:** Determines whether or not individual series and groups of risers can have their own appearance. Once you have set this property you can use \$seriestype to specify a riser type of the group or series.

**Values:** kGRgraphByType (the default) all risers in a 3D graph have the same shape, kGRgraphBySeries, kGRgraphByGroup.

## \$graphconfig

**Description:** For internal use only.

## \$maintitle

**Description:** The main title in the graph; usually appears at the top of the graph background. You can use \$locatetitle to reposition the title.

## \$majortype

**Description:** Specifies the major type of a graph; used in conjunction with \$minortype. Only certain minor types are permitted with a given major type. See \$minortype.

**Values:**

kGR3D	3D graph type
kGRarea	area graph type
kGRbars	bar graph type (the default)
kGRlines	line graph type
kGRpie	pie graph type
kGRspecial	special graph type; minor types include scatter graphs, histograms, stock market graphs

## \$minortype

**Description:** Specifies the minor type of a graph; used in conjunction with \$major type. Only certain minor types are permitted with a given major type.

**Values:** Depends on the major type.

<b>kGR3D</b>	kGR3DBars	kGR3DRowStep
	kGR3DCube	kGR3DColArea
	kGR3DCutCorner	kGR3DColLine
	kGR3DDiamond	kGR3DColStep
	kGR3DOctagon	kGR3DhoneycombSurface
	kGR3DPyramid	kGR3DModelSurface
	kGR3DSquare	kGR3Dsurface
	kGR3DRowArea	kGR3Dscatter
	kGR3DRowLine	

<b>kGRarea</b>	kGRabsolute	kGRstacked
	kGRabsoluteDualY	kGRstackedDualY
	kGRabsoluteBiPolar	kGRstackedBiPolar
	kGRabsoluteDualYBiPolar	kGRpercent

<b>kGRbars</b>	kGRstacked	kGRsideBySide
	kGRstackedDualY	kGRsideBySideDualY
	kGRstackedBiPolar	kGRsideBySideBiPolar
	kGRstackedDualYBiPolar	kGRsideBySideBiPolarDualY
		kGRpercent

<b>kGRlines</b>	kGRabsolute	kGRstacked
	kGRabsoluteDualY	kGRstackedDualY
	kGRabsoluteBiPolar	kGRstackedBiPolar
	kGRabsoluteDualYBiPolar	kGRstackedDualYBiPolar
		kGRpercent

<b>kGRpie</b>	kGRnormalPie	kGRpropMultipleRingPie
	kGRringPie	kGRpropMultiplePie
	kGRmultiplePie	kGRpieBar
	kGRmultipleRingPie	kGRringPieBar

<b>kGRspecial</b>	kGRhistogram
	kGRspectralMap
	kGRpolar
	kGRscatter
	kGRscatterDualY
	kGRhighLowOpenClose
	kGRdualYHighLowOpenClose
	kGRcontour

# \$gridlinesord01

**Parameters:** MajorHashStyle, MinorHashStyle, MinorHashCount, SuperHashStyle, SuperHashCount (HashStyles are OMNIS constants, Counts are Short integers).

**Description:** Controls the appearance of the 01 grid lines, that is, lines attached to the horizontal axis. Since these grid lines are not based on any sort of numeric scale, they have fewer possible controls than numeric-based grid lines.

**Values:**

Parameter	Value	Description
MajorHashStyle and MinorHashStyle	0	No grid lines
	1	Normal grid lines, height of frame
	2	Normal grid lines extended beyond height of frame
	3	TickIn; small tick marks from frame edge inward
	4	TickOut; small tick marks from frame edge outward
	5	TickIn and TickOut tick marks span across frame edge
MinorHashCount	0–50	the number of minor grids
SuperHashStyle	Same as MajorHashStyle	
SuperHashCount	0–50	the number of major grid lines between super grid lines

# \$gridlinesx

# \$gridlinesy1

# \$gridlinesy2

**Parameters:** LogScale, MajorHashManual, MinorHashManual (Booleans), MajorHashStyle, MinorHashStyle, MajorHashCount, MinorHashCount (styles are constants, counts Short integers).

**Description:** These properties specify how the grid lines of an axis (x, y1, or y2) are drawn. They affect axis labels and major and minor grid lines.

**Values:**

Parameter	Value	Description
LogScale	1	Use logarithmic scale
	0	Use linear scale (the default)
MajorHashManual	1	Use MajorHashCount setting
	0	Use the automatic settings
MinorHashManual	1	Use MinorHashCount setting
	0	Use the automatic settings
MajorHashStyle & MinorHashStyle	0	No grids lines
	1	Grid lines, width frame
	2	Grid lines extend beyond width of frame
	3	TickIn; small tick marks from frame edge inward
	4	TickOut; small tick marks from frame edge outward
	5	TickIn and TickOut span across frame edge
MajorHashCount & MinorHashCount	Any integer value	Count for the number of Major/Minor grid lines to draw (Major/MinorHashManual must be kTrue)



# \$gridmodemajoro1

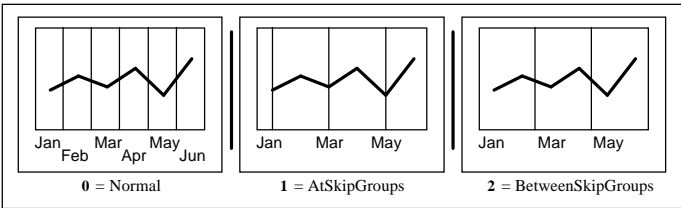
- Note:

Not implemented under MacOS.
- Object:

O1major
- Description:

This property controls which major grid lines show on the primary ordinal axis, and whether the grid lines are drawn between the groups or at the center of the groups (except pies).

When grids draw between groups, the grids display on either side of a group’s data points. When grids draw at the group, a grid aligns at the center of each group’s data points. If you want to hide some labels, this property lets you also hide the grid lines that would go with the missing labels as shown in the following illustrations:



If you skip the label for every other month, you can hide grid lines for the missing labels and “move” the drawn grid lines between the points or through the points.

**Values:** 0–2 (Short integer)

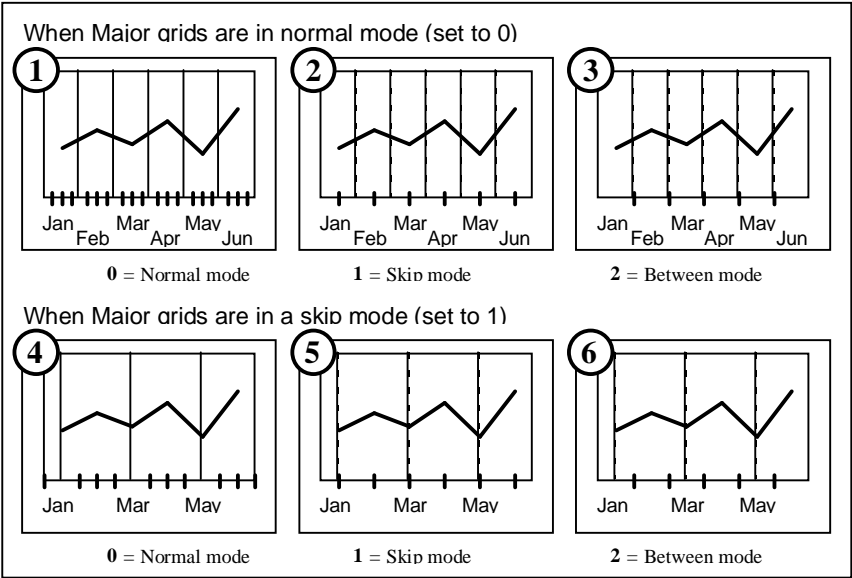
0	Grid lines are shown between each group (the default)
1	A grid line aligns at the center of each group that has a label displayed. When \$scalefreqo1 is >1, that value controls how many grid lines are drawn before one is skipped (hidden). For example, a label frequency value of “3” will draw 3 major grids, skip the 4th, draw 3 more major grids, skip the 8th, etc.
2	Same as 1, except grid lines align between groups with a label (according to value set for \$scalefreqo1)

# \$gridmodeminor01

- Note:**Not implemented under MacOS.
- Object:**O1minor
- Description:**This property controls whether grids are drawn between groups or at the center of the groups, in all graphs except pies. When labels are skipped (leaving certain data groups without labels), minor grid lines can be used to indicate the data groups (see \$labelmode01).

**Values:**

Value	Description
0	Minor grid lines are spaced evenly between each group. Values set within \$gridlinesord01 determine style and count (the default)
1	One minor grid line aligns at the center of each data group
2	One minor grid line aligns between each data group



Example #1 shows 3 minor grids, that is, 3 minor grids between each major grid. But if minor grid lines are set to a “skip” mode (examples #2 and #3), there is only one minor grid per data group and the minor grids align according to the data groups rather than the major grids. So a request for “3 minor grid lines” becomes irrelevant, since both skip modes ignore major grid lines.

Note that when major grids/labels are in a “skip” mode and minor grids are in the “normal” mode, as shown in example #4, the number minor grid lines is maintained despite the loss of major grid lines (compare examples #4 and #1).

Also note that whether major grids/labels are in “normal” or “skip” mode, they have no effect on minor grids that are set in “skip” mode (note how minor grids in examples #2 and #5, and #3 and #6 are the same). This is because minor grid lines in “skip” mode are placed according to data groups, not major grids.

## \$grouptitle

**Description:** The group title of a 3D graph.

## \$hlcolor

**Object:** Riser, Datamarker, Slice.

**Usage:** This property can be set in runtime only.

**Description:** This property highlights a particular riser, marker, or pie slice. It determines whether the data riser is associated with the series properties (same color, linestyle, marker shape, etc.) or is an independent, "highlighted" object. When set kTrue, this riser or marker will use a different color than the series color to which it belongs.

**Values:** kTrue the object is highlighted, kFalse (the default) the object has the properties of the entire series.

## \$hlwidth

**Description:** This property sets the width of the hilo risers in a Stock Market graph. It is similar to \$bargroupspacing, but it allows the stock market bars to be any width desired.

**Values:** Any positive integer value. (Long integer)

**Default Value:** 700

**Notes:** Although there is no limit to the width of risers/bars in a stock market graph, reasonable/readable values are limited to the output device and the number of groups/items being charted. Larger values may cause risers to overlap.

## \$ignoreseries

- Usage:** This property can be set in runtime only.
- Description:** This property lets you ignore or hide the specifies series. The riser or data marker for the specified series is hidden, and all associated objects including the legend markers and text objects are also hidden.
- Values:** kTrue to ignore or hide a series, kFalse (the default) show the series.

## \$insetlgndicon

## \$insetlgndtext

- Parameters:** LeftMargin, TopMargin, RightMargin, BottomMargin (Short integers).
- Usage:** This property can be set in runtime only.
- Description:** These properties specify the top, left, bottom, and right margins of the legend area. The legend icons or markers and text are fitted automatically within the remaining area. You specify the various margins as a percentage of the legend height or width.
- Values:** 0–100 percent for each parameter.

## \$itemdtxtx

- Description:** When data text values are enabled this property controls the type of data text items displayed in a Stock Market graph (either High, Low, Open, or Close). The \$showdatatext property must be enabled (kTrue) in order for this property to have any affect on the graph. You may also use the \$formatdtxty1 property to specify the format in which data text values are drawn, for example, \$9.99.
- Values:** kGRhigh, kGRlow (the default), kGRopen, kGRclose.

## \$labelmodeo1

**Object:** O1label

**Description:** This property sets the mode of labels on the o1 axis for all 2D graphs, except pies. For example, you can show every third label.

**Values:**

kGReven	all labels display and are evenly spaced along the o1-axis (the default)
kGRexact	all labels display, with each label aligned with its data group
kGRskipManual	labels are controlled manually via scale frequency properties; \$scalefreqo1 controls which group labels get skipped, \$scalefreqbego1 controls which group is first
kGRskipAuto	skip mode determined automatically

## \$labelwraplineso1

**Note:** Not implemented under MacOS.

**Object:** O1label

**Description:** This property sets the number of lines for labels on the o1-axis; the axis must be horizontal, it does not work on a vertical o1-axis. This property allows labels to have more than one line of text. You must set \$labelwrapmodeo1 to kTrue to enable multi-line labels. You can specify a large number of lines, but 3 to 4 is the practical limit. Multi-line labels do not support returns, soft returns, or tabs.

**Values:** 1 (the default) labels one line deep, to n allowing multiple lines for the o1-axis labels.

## \$labelwrapmodeo1

**Note:** Not implemented under MacOS.

**Object:** O1label

**Description:** This property controls whether or not the o1-axis labels can have more than one line.

**Values:** kFalse (the default) o1-axis labels are forced onto one line, or kTrue to allow multiple lines.

## \$legenditems

**Description:** This property sets the number of legend markers across the legend. The \$legendlayout property controls the horizontal or vertical layout of the legend. The following legend has \$legenditems set to 2.

---

 John	 George
 Paul	 Ringo

---

**Values:** 0 (the default) to the number of series in the chart (Short integer).

## \$legendlayout

**Description:** This property controls the layout or orientation of the graph legend. If you set this property to kGRauto the layout of the legend is set automatically according to the size and location of the legend box and the number of series in the graph. Alternatively, you can set this property to kGRvertical and use \$locatelegend to create a legend at the left or right side of your graph.

**Values:** kGRauto (the default), kGRhorizontal, or kGRvertical.

## \$linecolor

**Object:** Any line object, or border line of area object.

**Description:** Specifies the color of the line.

**Value:** RGB value.

## \$linestyle

**Object:** Any line object (implemented under Windows only)

**Description:** Specifies the style of the line object.

**Values:** Uses the default style

## \$linewidth

**Object:** Any line object.

**Description:** Specifies the width of a line object in pixels.

**Values:** Any positive integer.

## \$locatecoltitle2d

**Parameters:** *upperleftx, upperlefty, lowerrightx, lowerrighty*, corresponding to the x,y position of the upper left and lower right corner of the object relative to the center of the graph background.

**Description:** This property sizes and positions the column titles in a 3D graph.

**Values:** -16383 to 16383 for all x and y values.

## \$locatefootnote

**Parameters:** *upperleftx, upperlefty, lowerrightx, lowerrighty*, corresponding to the x,y position of the upper left and lower right corner of the object relative to the center of the graph background.

**Description:** This property sizes and positions the graph footnote. The footnote is positioned in the lower right corner of the graph by default.

**Values:** -16383 to 16383 for all x and y values.

## \$locateframe

**Parameters:** *upperleftx, upperlefty, lowerrightx, lowerrighty*, corresponding to the x,y position of the upper left and lower right corner of the object relative to the center of the graph background.

**Description:** This property sizes and positions the graph frame. Note that 3D graphs do not have a frame. When you resize and reposition the graph frame all risers, data points, labels, and text associated with the graph frame are resized and repositioned accordingly.

**Values:** -16383 to 16383 for all x and y values.

## \$locatelegend

**Parameters:** *upperleftx, upperlefty, lowerrightx, lowerrighty*, corresponding to the x,y position of the upper left and lower right corner of the object relative to the center of the graph background.

**Description:** This property sizes and positions the graph legend. When you resize and reposition the legend the data markers and legend text are resized and repositioned to fit the available area. You can use this property with \$legendlayout to position the legend precisely within the graph background.

**Values:** -16383 to 16383 for all x and y values.

## \$locatelytitle2d

**Parameters:** *upperleftx, upperlefty, lowerrightx, lowerrighty*, corresponding to the x,y position of the upper left and lower right corner of the object relative to the center of the graph background.

**Description:** This property sizes and positions the left y-axis title in a 3D graph.

**Values:** -16383 to 16383 for all x and y values.

## \$locaterowtitle2d

**Parameters:** *upperleftx, upperlefty, lowerrightx, lowerrighty*, corresponding to the x,y position of the upper left and lower right corner of the object relative to the center of the graph background.

**Description:** This property sizes and positions the row titles in a 3D graph.

**Values:** -16383 to 16383 for all x and y values.

## \$locaterytitle2d

**Parameters:** *upperleftx, upperlefty, lowerrightx, lowerrighty*, corresponding to the x,y position of the upper left and lower right corner of the object relative to the center of the graph background.

**Description:** This property sizes and positions the right y-axis title in a 3D graph.

**Values:** -16383 to 16383 for all x and y values.

## \$locatesubtitle

**Parameters:** *upperleftx, upperlefty, lowerrightx, lowerrighty*, corresponding to the x,y position of the upper left and lower right corner of the object relative to the center of the graph background.

**Description:** This property sizes and positions the graph subtitle. The subtitle appears at the top of the graph background under the Maintitle object by default.

**Values:** -16383 to 16383 for all x and y values.



## \$locatetitle

**Parameters:** *upperleftx, upperlefty, lowerrightx, lowerrighty*, corresponding to the x,y position of the upper left and lower right corner of the object relative to the center of the graph background.

**Description:** This property sizes and positions the main graph title. The Maintitle object appears at the top of the graph background by default.

**Values:** -16383 to 16383 for all x and y values.

## \$locatetitlex

**Parameters:** *upperleftx, upperlefty, lowerrightx, lowerrighty*, corresponding to the x,y position of the upper left and lower right corner of the object relative to the center of the graph background.

**Description:** This property sizes and positions the x-axis title in a 3D graph.

**Values:** -16383 to 16383 for all x and y values.

## \$locatetitley1

**Parameters:** *upperleftx, upperlefty, lowerrightx, lowerrighty*, corresponding to the x,y position of the upper left and lower right corner of the object relative to the center of the graph background.

**Description:** This property sizes and positions the y1-axis title.

**Values:** -16383 to 16383 for all x and y values.

## \$locatetitley2

**Parameters:** *upperleftx, upperlefty, lowerrightx, lowerrighty*, corresponding to the x,y position of the upper left and lower right corner of the object relative to the center of the graph background.

**Description:** This property sizes and positions the y2-axis title in a DualY graph.





























**Values:** -16383 to 16383 for all x and y values.

## \$markershape

**Object:** Datamarker. Select a data marker or series.

**Description:** This property specifies the shape of risers in Bar charts and the shape of markers in Line, Scatter, and Spectral Map graphs. It also affects the legend icons that represent the risers or markers. The \$uniformqdrshapes property determines whether all risers or markers on a graph use the same marker shape, or whether each series can have its own shape. The following shapes are available:

**Values:**

Shape	Constant	Shape	Constant
none	kGRnoMarker		kGRcastle
	kGRsRectangle (the default)		kGRisTri2
	kGRnStar45		kGRisTri3
	kGRplus		kGRisTri4
	kGRcircle		kGRrtTri1
	kGRdiamond		kGRrtTri2
	kGRspikedX		kGRrtTri3
	kGRplainX		kGRrtTri4
	kGRisTri1		kGRsEllipse
	kGRnStarSkewed		kGRsquare
	kGRfatPlus		kGRhexagon
	kGRnStar90		kGRpentagon
	kGRsoftX		kGRhouse
	kGRpiratePlus		kGRpentagram
	kGRfatX		

## \$markershapedefault

**Description:** This property sets the default marker for 3D Scatter graphs when the \$uniformqdrshapes property is enabled and the marker shape cannot be changed for individual series or groups.

**Values:** See \$markershape for details about setting the marker shape.

**Default Value:** kGRnoMarker

## \$markersize

**Object:** Datamarker. Select a data marker or series.

**Description:** This property affects the size of data markers in Line, Scatter, and Spectral Map type graphs.

**Values:** Any positive long integer value; a value of zero hides a marker.

**Default Value:** 750

**Notes:** See \$barriserwidth if you want to change the relative size of risers in a Bar chart.

## \$socheight

**Description:** This property specifies the height of the Open and Close tick marks that are displayed next to risers in a Stock Market graph. \$asmocwidth specifies the *width* of Open and Close tick marks.

**Values:** Any positive integer value (Long integer).

**Default Value:** 300

## \$socwidth

**Description:** This property specifies the width of the Open and Close tick marks that are displayed next to risers in a Stock Market graph. \$asmocheight specifies the *height* of Open and Close tick marks.

**Values:** Any positive integer value.

**Default Value:** 300

## \$orientation

**Description:** This property controls the vertical or horizontal orientation of a graph. Changing the graph major type resets a graph's orientation to vertical. This property only affects Bar, Area, and Spectral Map type graphs. Line graphs cannot be aligned horizontally.

**Values:** kTrue for vertical alignment (the default), kFalse for horizontal alignment.

## \$pictflip

**Object:** Any area object.

**Description:** This property specifies the orientation of a picture for an area object when \$areaeffect is set to kGRpicture.

**Values:**

kGRnoflip	Don't flip picture (the default)
kGRflipH	Flip picture horizontally
kGRflipV	Flip picture vertically
kGRflipHV	Flip picture horizontally/vertically

## \$pictname

**Object:** Any area object.

**Description:** This property specifies the name and path of a picture applied to an area object when \$areaeffect is set to kGRpicture.

**Value:** The name and path of a picture file: a BMP file under Windows, or PICT under MacOS.

## \$pictscale

**Object:** Any area object.

**Description:** This property sets the type of scaling for a picture when \$areaeffect is set to kGRpicture. By default the picture is not scaled and is drawn the exact size saved in the picture file.

**Values:**

kGRscaleNone	ScaleNone (the default)
kGRscale2Fit	Scale to fit
kGRscale2Frame	Scale to frame
kGRscale2Back	Scale to background
kGRtiled	Tiled
kGRtiled2Frame	Tiled to frame
kGRtiled2Back	Tiled to background

## \$piesperrow

**Description:** This property sets the number of pies displayed across the graph frame in a multiple pie graph, that is, the number of pies per row.

**Values:** 0 to number of groups in data range (Short integer).

0	Automatically calculate the best number of pies across the pie chart
n	n pies across the pie chart

**Default Value:** 2 pies per row

## \$placeo1

**Description:** Controls the placement of data text on or next to a graph riser or marker. If data text is showing on the graph (\$showdatatext is kTrue), changing this property will rearrange the placement of the data text. Placement controls are graph-wide; all instances of data text will follow the same placement scheme.

**Values:**

kGRcentre	Data text is centered on the riser (the default)
kGRoutMin	Data text is placed just outside the minimum of the riser (under bottom of bar or data marker)
kGRinMin	Data text is placed just inside the minimum of a riser (inside bottom of bar or data marker)
kGRinMax	Data text is placed inside the maximum of a riser (for a bar, text goes just inside the bar from the top of the riser; for a data marker, just inside the "top" side of the marker)
kGRoutMax	Data text is placed just outside the maximum of the riser (on top of bar or data marker)

# **\$placex**

## **\$placey1**

## **\$placey2**

**Object:** Datatext

**Description:** These properties control the placement of data text on or next to a graph riser or marker. If Data text is showing on the graph (\$showdatatext is kTrue), changing this property will rearrange the placement of the data text. Placement controls are graph-wide; all instances of data text will follow the same placement scheme. You can use \$formatdtxtx/y1/y2 to change the format of this data text.

**Values:** kGRcentre,kGRinMax, kGRinMin, kGRoutMax, kGRoutMin

kGRcentre	Data text is centered on the riser.
kGRoutMin	Data text is placed just outside the minimum of the riser (under bottom of bar or data marker).
kGRinMin	Data text is placed just inside the minimum of a riser (inside bottom of bar or data marker).
kGRinMax	Data text is placed inside the maximum of a riser (for a bar, text goes just inside the bar from the top of the riser; for a data marker, just inside the "top" side of the marker).
kGRoutMax	(the default) Data text is placed just outside the maximum of the riser (on top of bar or data marker).

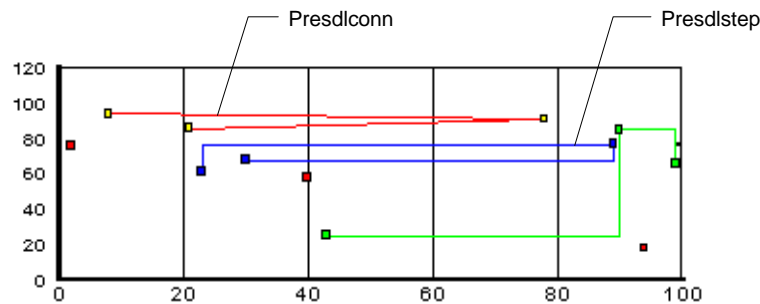
# \$presdlconn

# \$presdlconnbrk

# \$presdlstep

# \$presdlstepbrk

**Description:** For 2D Scatter graphs only, these properties enable/disable drawing of pre-series dependent data lines. Set the appropriate property to kTrue to show a data line.



The following table shows the object that is enabled or disabled by which property.

Property	Description
\$presdlconn	Enables Presdlconn, a solid line that connects data points
\$presdlconnbrk	Enables Presdlconnbrk, a line that connects data points with breaks for missing data points
\$presdlstep	Enables Presdlstep, a stepped line that connects data points
\$presdlstepbrk	Enables Presdlstepbrk, a stepped line that connects data points with breaks for missing data points

**Values:** kTrue to show the data line, or kFalse (the default) to hide the line.

## \$proplevels

**Description:** Determines the visibility of the properties of a graph shown in the Property Manager. When you first create a graph this property is set to kGRlow and the Property Manager shows only a few properties of a graph which lets you set up a basic graph. To fully configure your graph and its objects you will need to increase \$proplevels.

**Values:** kGRlow (the default), kGRmedium, kGRhigh.



## \$riserthick

**Parameters:** XThick, YThick, Zthick (Short integers)

**Description:** This property controls the thickness of the 3D risers, and it applies to floating cubes only. Since the y-axis value controls height, and height is usually determined by graph data (except for the floating cubes), y values do not serve well in controlling riser thickness. For the 3D connected graph type, the x-axis value affects series and the z-axis value affects groups. This property is not series dependent. It controls the thickness of all risers in a 3D graph.

**Value Values:** 0–200. Specified as ShortInts XThick, YThick, ZThick

5	Will produce “needles”
100	"Average" riser thickness (the default for x, y, and z values)
200	Risers touch each other

## \$rotate

**Description:** This property sets the rotation of a pie chart and determines where the first slice is drawn. A pie with 0 degrees rotation begins drawing slices at the 3 o'clock position in a counter-clockwise direction. Applying a 90 degree rotation causes the slices to begin drawing at the 12 o'clock position. The property \$drawclockwise determines the direction slices are drawn around the pie.

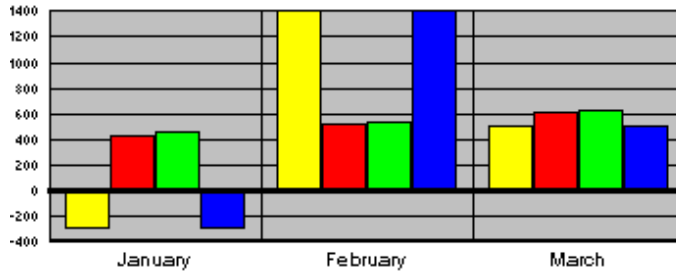
**Values:** 0–359 degrees (Long integer)

## \$scalebase

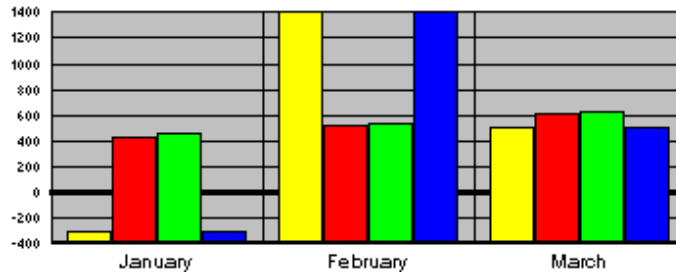
**Description:** Specifies whether risers are drawn from the zero line or the base of the graph frame. This is useful for comparing data that includes negative values. It affects axis labels, grid lines, and risers on all 2D graphs except pie charts.

**Values:** kGRfromZero (the default), or kGRfromFrame

For example, to draw risers from the zero line, set \$scalebase to kGRfromZero.



To draw risers from the base line of graph frame, set \$scalebase to kGRfromFrame.



**\$scaleendx**  
**\$scaleendy1**  
**\$scaleendy2**

**Object:** x1label, Y1label, Y2label

**Description:** These properties specify whether or not the scale labels for the first and last values of a scale will display on the axis (x, y1, or y2). They “mask off” the minimum and/or maximum values of a scale. For example, when the user does not want to see “0” as the origin, but merely wants the scale to start at the next increment, set this property to kMaxOnly to draw all numbers in the scale except the least (“0”).

**Values:**

kGRboth	A "normal" axis scale; all values, including min and max, are visible (the default)
kGRminOnly	The maximum value is masked off; only the minimum is visible
kGRmaxOnly	The minimum value is masked off; only the maximum is visible
kGRnone	Neither minimum nor maximum value is shown on the axis scale

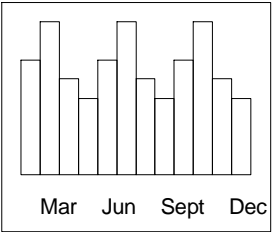
**\$scalefreqbego1**

**Object:** O1label

**Description:** This property works in conjunction with the \$scalefreqo1 property and let you skip group labels on the ordinal axis. The \$scalefreqo1 divides the o1 labels into sets and \$scalefreqbego1 lets you select the nth label in the set.

For example, assume you have group labels for each month of the year and have set \$scalefreqo1 to display every third label (Jan, Apr, Jul, Oct). The \$scalefreqbego1 property let you select a different label in each set (for example, the second label in each set Feb, May, Aug, Nov, the third in each set Mar, Jun, Sep, Dec, and so on). Using the months of the year example, the illustration shows the labels that are displayed when \$scalefreqo1 is set to 3 and \$scalefreqbego1 is set to 2.

Note that this may allow the last set to miss a label. For example, if there are 10 groups, \$scalefreqo1 is set to 4 (4



\$scalefreqo1 = 3 and  
\$scalefreqbego1 = 2

groups per set) and \$scalefreqbego1 is set to 3 (display third label in each set), then the third set will not have a label.

**Values:** 0 to n, where n is the number of labels in each set specified by \$scalefreqo1.

0	first group’s label from each set displays on the o1-axis (the default)
1	second group’s label from each set displays on the o1-axis
n	displays the nth label in each set; it must be less than or equal to the number of labels per set specified by \$scalefreqo1

**\$scalefreqbegx**  
**\$scalefreqbegy1**  
**\$scalefreqbegy2**

**Object:** X1label, Y1label, Y2label

**Description:** These properties control the number of labels that are drawn on the numeric axis and are used in conjunction with \$scalefreqx/y1/y2 and \$scalefreqendx/y1/y2. When a scale frequency is not specified or is set to zero by \$scalefreqx/y1/y2, this property identifies the first label to be drawn.

**Values:** 0 (the default) to n, where n is the number of labels in each set specified by \$scalefreqx/y1/y2.

**Example:** When \$scalefreqx/y1/y2 is non-zero, the labels are divided into groups (number of labels divided by \$scalefreqx/y1/y2+1) and \$scalefreqbeg... specifies the first label in each group to draw as follows

<b>\$scalefreqx/y1/y2 = 1</b> <b>\$scalefreqbeg... = 0</b>	<b>\$scalefreqx/y1/y2 = 1</b> <b>\$scalefreqbeg... = 1</b>
1000	
	900
800	
	700
600	
	500
400	

# \$scalefreqendo1

**Object:** O1label

**Description:** This property is used in conjunction with \$scalefreqo1 and \$scalefregobegob1 to control the number of labels that are drawn on the ordinal axis. When a label frequency has not been specified or a label frequency of zero is specified, this property defines the last label to be drawn:

**Values:** 0 (the default) to the number of ordinal axis labels or label groups specified by \$scalefreqo1.

**Example:**

\$scalefreqo1 = 0					
Jan	Feb	Mar	Apr	May	Jun
\$scalefreqo1 = 1					
Jan		Mar		May	
\$scalefreqendo1 = 0					
Jan	Feb	Mar	Apr	May	Jun
\$scalefreqendo1 = 1					
Jan	Feb	Mar	Apr	May	
\$scalefreqendo1 = 1					
Jan		Mar			

**\$scalefreqendx**  
**\$scalefreqendy1**  
**\$scalefreqendy2**

- Object:** X1label, Y1label, Y2label
- Description:** These properties are used in conjunction with \$scalefreqx/y1/y2 and \$scalefreqbegx/y1/y2 to control the number of labels that are drawn on the numeric axis. When a scale frequency is not specified or is set to zero by \$scalefreqx/y1/y2, this property identifies the last label to be drawn.
- Values:** 0 (the default) to the number of labels in the graph or label groups specified by \$scalefreqx/y1/y2.
- Example:** When \$scalefreqx/y1/y2 is non-zero, the labels are divided into groups (number of labels divided by \$scalefreq...+1) and \$scalefreqend... specifies the last group to draw as follows

<b>\$scalefreqx/y1/y2 = 1</b> <b>\$scalefreqend... = 0</b>	<b>\$scalefreqx/y1/y2 = 1</b> <b>\$scalefreqend... = 1</b>
1000	
800	800
600	600
400	400

# \$scalefreqo1

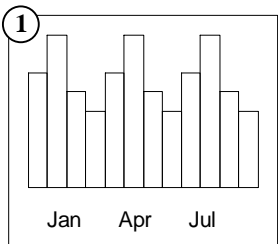
**Object:** O1label

**Description:** This property is used to specify the frequency of the labels on the primary and secondary ordinal axis. You can display every other label, every third label, and so on, using the \$scalefreqbego1 property.

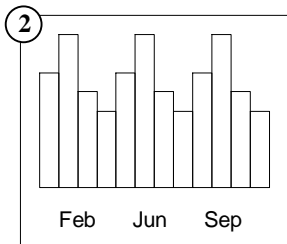
**Values:** Any positive integer, less than or equal to the total number of groups.

0	Every label is displayed (the default)
1	Every other label is displayed. \$scalefreqbego1 determines whether the first or second label of each set is displayed
n	o1 labels are divided into sets containing n number of labels; when the \$scalefreqbego1 is not used or set to the default value (0), the first label in each set is displayed

The \$scalefreq... properties are useful when you have many groups and want to create a less cluttered appearance by eliminating some of the group labels. The good example involves a graph containing a group for each month of the year. If you set the \$scalefreqo1 property to 3, the graph will display every third label (see example #1). In this case, the group labels, that is, the names of the months, are divided into sets of three and the first label from each set (January, April, July,...) is displayed.



\$scalefreqo1 = 3 and  
\$scalefreqbego1 = 0



\$scalefreqo1 = 4 and  
\$scalefreqbego1 = 1

In example #2, the group labels are divided into groups of four and the second label of each group is displayed.

## \$scalefreqx

## \$scalefreqy1

## \$scalefreqy2

**Object:** X1label, Y1label, Y2label

**Description:** These properties specify the frequency of labels displayed on the numeric axis (x, y1 and y2). They are used in conjunction with \$scalefreqbegx/y1/y2 and \$scalefreqendx/y1/y2 to limit the number of data labels displayed on the numeric axis. Note that these properties do not affect the axis grid lines which are drawn even if the associated label has been skipped by the \$scalefreq... properties.

**Values:** Any positive integer, less than or equal to the total number of groups.

0	Draw every label (the default)
1 to n	Divide the labels into groups of n+1. \$scalefreqbegx/y1/y2 determines which label in each group is drawn (the default is the first label in each group).

## \$scalerangex

## \$scalerangey

## \$scalerangez

**Parameters:** ManualScale (Boolean), ManualScaleMax, ManualScaleMin (Floats)

**Description:** These properties set the minimum and maximum values for the x, y, and z numeric axis.

Parameter	Value & Description	
ManualScale	0	Use automatic values to scale graph (the default)
	1	Use both the max and min scale values for graph
	2	Use manual scale for minimum, and autoscale for maximum
	3	Use autoscale for minimum, and manual scale for maximum
ScaleMax	Any real number. The value to use for the upper limit of the axis to draw the graph; 10000.0 is the default	
ScaleMin	Any real number. The value to use for the lower limit of the axis to draw the graph; 0.0 is the default	



## \$scaletype

**Description:** This property is used to specify a logarithmic scale (as opposed to linear) for the selected numeric axes (x, y or z). When the x-axis is numeric \$scaletype only applies to (XY) Scatter graphs. For the y1 and z axes it affects labels, major and minor grids, risers and markers on all graphs except Pies and Spectral Maps. Linear/logarithmic scaling can also be set by the \$gridlinesx/y1/y2 properties.

### Values:

kGRnoAxis	Use the linear scale for all axes (the default)
kGRaxisX	Use the logarithmic scale for the x-axis
kGRaxisY	Use the linear scale for the y1- or y2-axis
kGRaxisZ	Use the logarithmic scale for the z axis

## \$scalex

## \$scaley1

## \$scaley2

**Parameters:** ManualScale (Short Integer), ScaleMax, ScaleMin (Floats)

**Description:** These properties set scale values on a numeric axis (x, y1, or y2). They affect axis labels, grid lines, and data riser objects related to the x, y1, or y2-axis on all 2D graphs except Pies.

### Values:

Parameter	Value and description	
ManualScale	0	Use automatic values to scale graph (the default)
	1	Use both the max and min scale values for graph
	2	Use manual scale for minimum, and autoscale for maximum
	3	Use autoscale for minimum, and manual scale for maximum
ScaleMax	Manual minimum value to draw the graph and its axis	
ScaleMin	Manual maximum value to draw the graph and its axis	

## \$scatter

**Description:** Shows or hides the lines connecting the markers in a 3D scatter graph.

**Values:** kTrue to show connecting lines, or kFalse (the default) to hide them.

## \$scimovavg

**Object:** Movavglines.

**Usage:** This property can be set in runtime only.

**Description:** This property specifies the calculation method (scientific or financial) that will be used to draw a moving average curve (Movavglines) when it is enabled by the \$sdlinemova property. \$curvemoving controls the number of points used to generate the line.

**Values:** kTrue use scientific moving average, kFalse (the default) use financial moving average.

## \$sddatalinetype

**Object:** Datalines. Select the series.

**Usage:** This property can be set in runtime only.

**Description:** On Line and 2D Scatter charts, this series-dependent property determines if the marker, line, or both are drawn.

**Values:**

0	Display marker and line (the default)
1	Display marker only
2	Display line only

## \$sdemphasized

**Object:** Riser, Datamarker, Areariser. Select the series.

**Usage:** This property can be set in runtime only.

**Description:** This property emphasizes one or more series (of risers, data markers, or area risers) in a graph by making their “graph type” look different from the other series. For example, it lets you represent individual series of data with a line or area representation in a bar graph, or with a bar or area riser in a line graph, or with a line or bar in an area graph. The method of emphasis is different depending on the graph type as shown on the following table.

**Values:**

0	No emphasis (the default): bars, lines, and risers are unaffected
1	Series in a bar graph is represented by lines Series in a line graph is represented by a bar riser Series in an area graph is represented by lines
2	Series in a bar graph is represented by an area riser Series in a line graph is represented by an area riser Series in an area graph is represented by a bar riser

## \$sdlineconn

**Object:** Curve. Select the series.

**Usage:** This property can be set in runtime only.

**Description:** When enabled, this property draws a connected line along a specified series on any 2D graph (except Pies and Spectral Maps). This line, like all curve types, is series dependent, you must specify the ID of the series that the line connects.

**Values:** kTrue to draw a connecting line, kFalse (the default) don't draw a line.

## \$sdlinecurv

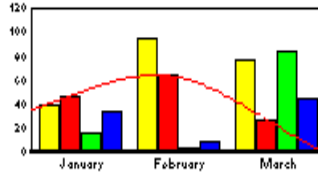
**Object:** Curve. Select the series.

**Usage:** This property can be set in runtime only.

**Description:** When enabled, this property draws a smooth curve line for a specified series. Like all curve types, this line is series dependent, that is, it specifies the series of data representations across which a curved line is drawn. The number of points to be used in calculating the curve is specified by \$curvesmooth.

**Values:** kTrue to draw a smooth curve line for this series, kFalse (the default) don't draw the line.

**Example:** To draw a smooth curve for the second series, set \$curvesmooth to 200, and \$sdlinecurv to kTrue for series 2.



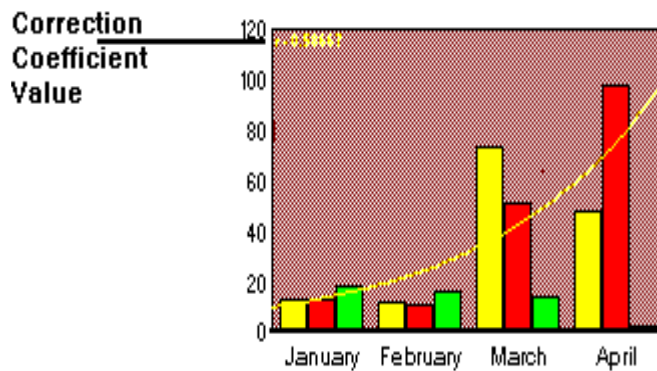
## \$sdlinelincorr

**Object:** Lincorr. Select the series.

**Usage:** This property can be set in runtime only.

**Description:** This property enables/disables the display of the correction coefficient used when linear regression, exponential regression, common log regression, natural log regression and polynomial fit lines are enabled and drawn.

**Values:** kTrue to show the correction coefficient, kFalse (the default) don't show the value.



## \$sdlinelinrexp

**Object:** Linrexp. Select the series.

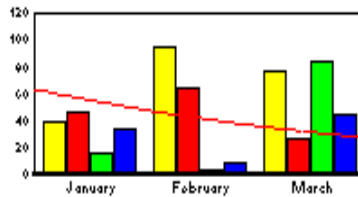
**Usage:** This property can be set in runtime only.

**Description:** This property enables/disables drawing of an exponential linear regression line for a specified series. It is calculated using the formula  $y = a * b^{**}x$ . This line, like all curve types, is series dependent, that is, the SeriesID identifies the series across which the line is drawn. Therefore, the value for the exponential regression is calculated from the values of all data elements that are in this series.

**Values:** kTrue to draw an exponential linear regression line for series, kFalse (the default) don't draw the line.

**Notes:** See \$sdlinelinrcorr or \$sdlinelinrformula if you want to draw the correction coefficient or formula text associated with this line.

**Example:** To enable an exponential linear regression line for the second series, set \$sdlinelinrexp to kTrue for series 2.



## \$sdlinelnrformula

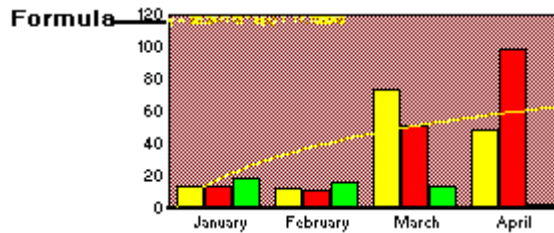
**Object:** Lnrtext. Select the series.

**Usage:** This property can be set in runtime only.

**Description:** This property enables/disables the display of the formula used when linear regression, exponential regression, common log regression, natural log regression and polynomial fit lines are enabled and drawn.

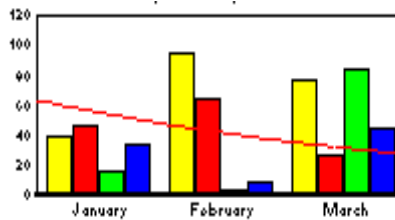
**Values:** kTrue to show the formula, kFalse (the default) to not show the formula.

**Example:** To show formula for the first series, set \$sdlinelnrformula to kTrue for Series 1.



## \$sdlinelinrline

- Object:** Linrline. Select the series.
- Usage:** This property can be set in runtime only.
- Description:** This property enables/disables drawing of a linear regression line for a specified series. This line, like all curve types, is series dependent, that is, the SeriesID specifies the series across which the line is drawn. Therefore, the value for the linear regression is calculated from the values of all data elements that are in this series.
- Values:** kTrue to draw a linear regression line for this series, kFalse (the default) to not draw the line.
- Notes:** See \$sdlinelinrcorr or \$sdlinelinrformula if you want to draw the correction coefficient or formula text associated with this line.
- Example:** To enable linear regression line for the second series, set \$sdlinelinrline to kTrue for Series 2.





## \$sdlinelinrlog

**Object:** Linrlog. Select the series.

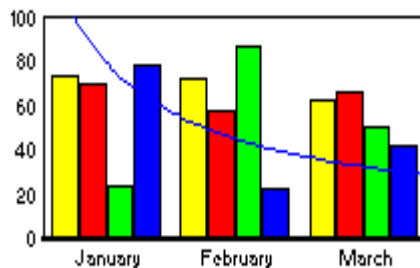
**Usage:** This property can be set in runtime only.

**Description:** This property enables/disables drawing of a common log regression line for a specified series. It is calculated using the formula  $y = a * x^{**}b$ . This line, like all curve types, is series dependent, that is, the line is drawn across a specified series. Therefore, the value for the common log regression is calculated from the values of all data elements that are in this series.

**Values:** kTrue to draw a common log regression line for a series, kFalse (the default) to not draw the line.

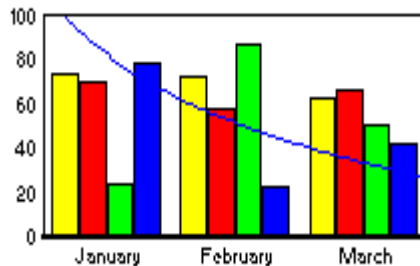
**Notes:** See \$sdlinelinrcorr or \$sdlinelinrformula if you want to draw the correction coefficient or formula text associated with this line.

**Example:** To enable common log regression line for the fourth series, set \$sdlinelinrlog to kTrue for series 4.



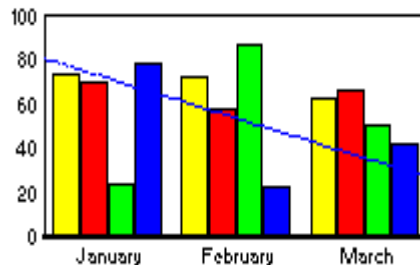
## \$sdlinlinrnatlog

- Object:** Linrnatlog. Select the series.
- Usage:** This property can be set in runtime only.
- Description:** This property enables/disables drawing of a natural log linear regression line for a specified series. The linear regression line is calculated as  $y = a \cdot \log(x) + b$ . This line, like all curve types, is series dependent, that is, the line is drawn across a specified series. Therefore, the value for the natural log regression is calculated from the values of all data elements that are in this series.
- Values:** kTrue to draw a natural log regression line for a series, kFalse (the default) to not draw the line.
- Notes:** See \$sdlinlinrcorr or \$sdlinlinrformula if you want to draw the correction coefficient or formula text associated with this line.
- Example:** To enable natural log regression line for the fourth series, set \$sdlinlinrnatlog to kTrue for series 4.



## \$sdlinelnrnpoly

- Object:** Linnrnpoly. Select the series.
- Usage:** This property can be set in runtime only.
- Description:** This property enables/disables drawing of a polynomial fit line for a specified series. It is calculated using the formula  $y = a + b \cdot x + b \cdot x^2$ . This line, like all curve types, is series dependent, that is, the line is drawn across a specified series. Therefore, the value for the polynomial fit is calculated from the values of all data elements that are in this series.
- Values:** kTrue to draw a polynomial fit line for this series, kFalse (the default) to not draw the line.
- Notes:** See \$sdlinelnrcorr or \$sdlinelnrformula if you want to draw the correction coefficient or formula text associated with this line. See \$sdlinelnrnpolyfac to set the degree of linear regression used in the calculation for this line.
- Example:** To enable a polynomial fit line for the fourth series, set \$sdlinelnrnpoly to kTrue for Series 4.



## \$sdlinelnrnpolyfac

- Object:** Linnrnpoly. Select the series.
- Usage:** This property can be set in runtime only.
- Description:** This property specifies the degree of linear regression for the polynomial fit line. The polynomial fit line is drawn by the \$sdlinelnrnpoly property. If \$sdlinelnrnpolyfac is set to less than one, it will be reset to 1. If it is set to greater than the number of groups, it will be reset to exactly the number of groups in the graph.
- Values:** 1 (the default) to the number of groups in the graph.

## \$sdlinemean

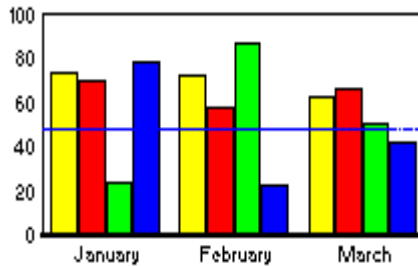
**Object:** Linrtext. Select the series.

**Usage:** This property can be set in runtime only.

**Description:** This property enables/disables drawing of a mean average line for a series. Like all line types, it is series dependent, that is, the line is drawn for a specific series. Therefore, the value for the mean is calculated from the values of all data elements in the specified series.

**Values:** kTrue to draw a mean average line for a series, kFalse (the default) to not draw the line.

**Example:** To enable a mean average line for the fourth series, set\$sdlinemean to kTrue for Series 4.



## \$sdlinemova

**Object:** Movavgline. Select the series.

**Usage:** This property can be set in runtime only.

**Description:** This property enables/disables drawing of a moving average line for this series. Like all line types, it is series dependent, hence the name. Therefore, the value for the moving average is calculated from the values of all data elements in the specified series.

**Values:** kTrue to draw a moving average line for a series, kFalse (the default) to not draw the line.

## \$sdlinestdd

<b>Object:</b>	Stddevline. Select the series.
<b>Usage:</b>	This property can be set in runtime only.
<b>Description:</b>	This property enables/disables drawing of a standard deviation line for a specified series. The value for the standard deviation is calculated from the values of all data elements that are in this series. This line, like all line types, is series dependent, hence the name "sdline..."
<b>Values:</b>	kTrue to draw a standard deviation line for a series, kFalse (the default) to not draw the line.

## \$selectobject

<b>Description:</b>	When enabled this property lets you select individual objects inside the graph control, otherwise when disabled you can select the whole graph. This lets you view and set the properties of specific objects within the graph.
---------------------	---

## \$seriesgroupswap

<b>Description:</b>	This property changes the interpretation of your list data; when kFalse (the default), list columns are interpreted in your graph as groups and rows become series, otherwise when kTrue, list columns are represented as series and rows as groups. When series and groups are swapped, all axis-related objects (labels, grids, risers, and so on) are repositioned. For example, on a bar graph the series names in the legend text will switch with the group names on the ordinal axis. In 3D graphs, data on the series x-axis swaps with the group z-axis data.
---------------------	--

## \$seriestitle

<b>Description:</b>	The series title of a 3D graph.
---------------------	---------------------------------

## \$seriestype

**Usage:** This property can be set in runtime only.

**Description:** This property specifies the shape of the risers in a series or group depending on *\$graphbywhat*. This property has no effect on the risers if *\$graphbywhat* is set to *kGRgraphByType*: in this case, all risers maintain the type defined by *\$minortype* regardless of *\$seriestype*. If *\$graphbywhat* is set to *kGRgraphBySeries* you can set the riser shape by series, or *kGRgraphByGroup* you can set the riser shape by group.

**Values:**

0	3DBars (the default)	9	3DRowStep
1	3DCube	10	3DColArea
2	3DCutCorner	11	3DColLine
3	3DDiamond	12	3DColStep
4	3DOctagon	13	3DHoneycomb
5	3DPyramid	14	3DModel
6	3DSquare	15	3DSurface
7	3DRowArea	16	3DScatter
8	3DRowLine		

## \$shadowoffsets

**Description:** This property controls the offset for the shadows on risers; you must enable *\$showshadows* to see the effect of this property. You specify an x- and y-coordinate for the offset. Positive values move the shadow up and to the right, negative values move the shadow down and to the left. This property affects all the risers on a graph.

**Values:** -1000 to 1000 for each x and y offset value.

**Default Value:** 0,0 (no offset); ignores further values

## \$showbaseline

- Object:** X1body, Y1body, Y2body
- Description:** Shows or hides the numeric axis baseline for the selected axes, also affecting the grid lines and label objects.
- Values:** kAxisX, kAxisY, kAxisZ. For a 2D graph, kAxisZ denotes Y2.
- Default Value:** kAxisX, kAxisY, kAxisZ

## \$showclose

- Object:** Osmriser. Select the series or group.
- Description:** Hides or shows the Close tick marks for stock market risers. You can hide/show the Open tick marks using \$showopen. You have to supply the data for the Open and Close values for stock market graphs, otherwise the tick marks will not show.
- Values:** kTrue (the default) to show Close tick marks, or kFalse to hide them.

## \$showdatatext

- Object:** Datatext
- Description:** This property shows or hides values and/or labels for a riser or data point. It affects the data text and slice values for pie charts.
- Values:**

kGRtextNone	no values or non-numeric text is shown (the default)
kGRtextValue	show numeric values only
kGRtextText	show non-numeric text only
kGRtextBoth	show both values and non-numeric text
kGRtextABS	use absolute segment value on stacked
kGRtextABSboth	show both absolute segment value on stacked and non-numeric text
kGRtextABSvalue	show numeric and absolute segment values on stacked

## \$showdivbipolar

**Object:** Divbipolar

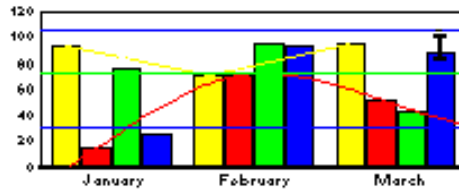
**Description:** This property shows or hides the line that separates the two halves of a bipolar graph.

**Values:** kTrue (the default) to show the line, or kFalse to hide it.

## \$showerrorbar

**Usage:** This property can be set in runtime only.

**Description:** This property shows or hides error bars on a graph. To show error bars on a graph you need to supply the graph with enough data and set the \$dataformat property to the appropriate data format. The size of the error bars is controlled by \$sizeerrorbars.



**Values:** kTrue to show error bars, kFalse (the default) to hide them.

## \$showfeeler

**Object:** Pie slice, specify series id.

**Parameters:** SeriesID (Short integer), Show/Hide (Boolean).

**Usage:** This property can be set in runtime only.

**Description:** This property shows or hides a feeler for the specified series.

**Values:** kTrue (the default) to show feeler, or kFalse to hide feeler.

## \$showlblfeeler

**Object:** Pie slice, specify series id.

**Parameters:** SeriesID (Short integer), Show/Hide (Boolean).

**Usage:** This property can be set in runtime only.

**Description:** This property shows or hides the label for a pie chart feeler.

**Values:** kTrue (the default) to show feeler label, or kFalse to hide it.



## \$showblpie

**Description:** This property hides or shows the group labels for a pie chart.

**Values:** kTrue (the default) to show pie label, or kFalse to hide the labels.

## \$showblring

**Description:** This property hides or shows the labels at the center of ring pies.

**Values:** kTrue (the default) to show ring pie labels, or kFalse to hide them.

## \$showlegend

**Description:** This property hides or shows the graph legend.

**Values:** kTrue (the default) to show the legend, or kFalse to hide it.

## \$showoffscale

**Description:** This property specifies whether or not risers or data points are shown if they are out of range on the x, y, or z axes. When set to kFalse for an axis, risers that fall out of range are not drawn, otherwise if kTrue risers are shown truncated.

**Values:** kTrue (the default) to display risers even if they are out of range, or kFalse to hide risers if they are out of range.

**Example:** To omit y1 values over 500, set \$scaley1 to 1, 0, 500, and \$showoffscale to kFalse.

## \$showopen

**Object:** Osmriser. Select the series or group.

**Description:** This property hides or shows the Open tick marks for stock market risers. You can hide/show the Close tick marks using \$showclose. You have to supply the data for the Open and Close values for stock market graphs, otherwise the tick marks will not show.

**Values:** kTrue (the default) to show Open tick marks, or kFalse to hide them.

## \$showshadows

**Description:** Shows a shadow on risers, markers, and lines. The amount of shadow offset is controlled by \$shadowoffsets.

**Values:** kFalse (the default) not to show shadows, kTrue to show them.

## **\$showsliders**

**Description:** Shows a slider control window enabling graph rotation and zooming, for 3D Graphs only. On opening, the slider control is set relative to the current position of the graph.

**Values:** `kFalse` (the default) not to show slider control, `kTrue` to show sliders.

## **\$showtextdatalabels**

**Description:** This property hides or shows the data and text labels for the risers in a 3D graph.

**Values:** `kFalse` (the default) to hide the data text, `kTrue` to show the data text.

## **\$showtextdetails**

**Description:** This property hides or shows text labels for the column and row headers, and the right and left numeric axis in a 3D graph.

**Values:** `kGRnoHeader` (the default), `kGRcolHeader`, `kGRrowHeader`, `kGRleftNumber`, `kGRrightNumber`.

## **\$showwall**

**Description:** This property hides or shows the floor, left wall, and right wall of a 3D graph.

**Values:** `kGRnoWall` (the default), `kGRleft`, `kGRfloor`, `kGRright`.

## **\$showzeroline**

**Object:** `X1zero`, `Y1zero`, `Y2zero`

**Description:** Shows or hides the zero line for the selected axes.

**Values:** `kGRaxisX`, `kGRaxisY`, `kGRaxisZ`; for a 2D graph, `kAxisZ` denotes `Y2`.

## \$sideo1

**Description:** Controls which side of the graph an o1 axis will be drawn. Although the vast majority of all graphs will draw their ordinal text on the bottom or left of the graph, or low side, you can position it to the top or right, or high side, or both sides.

**Values:**

kGRbottomOrleft	Draw text at the bottom or left (the default)
KGRtopOrright	Draw text at the top or right
KGRboth	Draw text on both sides
KGRnoImage	Don't draw text

## \$sidex

## \$sidey1

## \$sidey2

**Description:** These properties control which side of the graph the x, y1, or y2 axis is drawn. The usual positioning for the numeric axis is y1 on the left and y2 on the right, but using these properties you can, for example, place both y1 and y2 labels on the same side of the graph.

**Values:**

kGRbottomOrleft	Draw text at the bottom or left
kGRtopOrright	Draw text at the top or right
kGRboth	Draw text on both sides.
kGRnoImage	Don't draw text

## \$sizeerrorbars

**Object:** Errorbar. Select the series.

**Usage:** This property can be set in runtime only.

**Description:** Sets the width of the error bar "tails". It affects error bar objects on Bar and Scatter type graphs.

**Values:** 0 (the default) to any number in pixels (Long integer).

## \$sliceddelete

<b>Object:</b>	Pie slice.
<b>Usage:</b>	This property can be set in runtime only.
<b>Description:</b>	This property hides or shows a pie slice. \$slicerestore displays all previously deleted or hidden slices.
<b>Values:</b>	kTrue deletes or hides the slice, kFalse (the default) displays or restores a pie slice.

## \$slicemove

<b>Object:</b>	Pie slice.
<b>Usage:</b>	This property can be set in runtime only.
<b>Description:</b>	This property moves or detaches a pie slice from the pie. The amount of detachment is a percentage of the radius of the pie. Therefore, a value of 100% would cause the center point of the slice to draw at the edge of the pie. Detaching a slice too much may draw it outside the chart frame. \$slicerestore restores all slices to their default positions.
<b>Values:</b>	0 (the default) to 100 percent of pie radius.

## \$slicerestore

<b>Object:</b>	All splices in the pie chart.
<b>Usage:</b>	This property can be set in runtime only.
<b>Description:</b>	This property displays all slices deleted or hidden with \$sliceddelete, or restores all slices moved with \$slicemove.

## \$splity

<b>Object:</b>	Riser. Select the series.
<b>Usage:</b>	This property can be set in runtime only.
<b>Description:</b>	This is a series dependent property which determines whether or not a series belongs to the y1 or y2 axis.
<b>Values:</b>	kFalse (the default) series belongs to the primary y1 axis, or kTrue series belongs to the secondary y2 axis.

## **\$squarelndicons**

**Object:** Legendmarker

**Description:** Normally, the rectangle for a marker is locked to the current setting of \$insetlndicon, and takes the same shape as the margin settings for that property. When this property is kTrue, the legend marker is a square. The square is positioned in the center of the rectangle defined by \$insetlndicon. This property affects all legend marker objects (colored icons within the legend) and all 2D graph types.

**Values:** kTrue legend marker is square (the default), or kFalse uses other marker shape.

## **\$staggero1**

**Description:** Specifies whether or not the labels for the o1-axis are staggered. The default is to draw them in line.

**Values:** kTrue stagger labels, or kFalse (the default) draw labels in line.

## **\$staggerx**

## **\$staggery1**

## **\$staggery2**

**Description:** Specifies whether or not the labels for the x-, y1-, or y2-axis are staggered. The default is to draw them in line.

**Values:** kTrue stagger labels, or kFalse (the default) draw labels in line.

## **\$subtitle**

**Description:** The subtitle for the graph; appears under the main graph title by default.

## \$tilt

- Description:** This property defines the amount of tilt applied to a pie chart. A pie with 0 degrees tilt is drawn as a regular 2D pie. As tilt is applied, the pie is rotated backward on the x-axis. The tilting creates a 3D effect, especially when used in conjunction with \$depth. Tilting must be enabled using \$tilton which is on by default.
- Values:** 0–89 degrees (Short integer)
- Default Value:** 40 (degrees)

## \$tilton

- Description:** This property lets you tilt the pies in a pie chart. You can set the amount of tilt using \$tilt. Set \$tilton to kFalse for a 2D or flat pie chart.
- Values:** kTrue (the default) to enable tilting, kFalse for no tilt.

## \$uniformqdrborders

- Description:** This property determines whether all risers or data markers on a graph use the same border style and color, or whether each series can have its own style. It affects all data risers on the graph (bars and markers) and all graph types except pies.
- Values:** kTrue (the default) the same style on all risers or markers, or kFalse means you can set the border style for each series.

## \$viewmatrix

- Description:** This property lets you redefine the view of a 3D graph, and specifically identifies the transformation matrix for a graph.
- Values:** x, y, z values (Real numbers).

## \$viewposition

- Description:** This property lets you redefine the view of a 3D graph, and specifically changes the viewing angle of a graph.
- Values:** x, y, z values (Long integers) identifying the graph cube.

## **\$viewsize**

- Description:** This property lets you redefine the view of a 3D graph, and specifically identifies the graph cube size.
- Values:** 0 to 16383 for all x, y, z values (Long integers) as a proportion of the graph background. Larger values explode the graph cube.

## **\$viewwallthick**

- Parameters:** Leftwall, Floor, Rightwall (Long integers)
- Description:** This property specifies the thickness of the left wall, floor, and right wall of a 3D graph.
- Values:** 0 or over; 1000 applies a reasonable thickness.

## **\$viewpanx**

- Description:** This property lets you redefine the view of a 3D graph, and specifically controls the horizontal displacement of the graph cube. A zero value places the cube at the center of the graph.
- Values:** -16383 to 16383, larger values place the graph cube outside the graph.

## **\$viewpany**

- Parameters:**
- Description:** This property lets you redefine the view of a 3D graph, and specifically controls the vertical displacement of the graph cube. A zero value places the cube at the center of the graph.
- Values:** -16383 to 16383, larger values place the graph cube outside the graph.

## \$washdir

**Description:** This property controls the direction of a wash effect for an area object when \$areaeffect is set to kGRwash.

**Values:**

kGRnoWashDir	No Wash (the default)
kGRwashUpLeft	Up Left
kGRwashUp	Up
kGRwashUpRight	Up Right
kGRwashLeft	Left
kGRwashRight	Right
kGRwashDownLeft	Down Left
kGRwashDown	Down
kGRwashDownRight	DownRight
kGRwashOval	Zoom oval
kGRwashZoomRect	Zoom rectanlge

## \$washend

**Description:** This property defines the ending color of a wash for an area object when \$areaeffect is set to kGRwash. The \$washstart property defines the beginning color of a wash. You can use \$washdir and \$washtype to set the direction and type of wash between \$washstart and \$washend. The wash is applied according to the current coloring scheme specified by \$colormodel.

**Values:** RGB value.

## \$washscale

**Object:** Any area object.

**Description:** This property controls whether or not a wash effect is scaled to fit the area object, and is implemented under MacOS only.

**Values:** kTrue to scale wash, kFalse not to scale.



## \$washstart

**Description:** This property defines the beginning color of a wash for an area object when \$areaeffect is set to kGRwash. The \$washend property defines the ending color of the wash. You can use \$washdir and \$washtype to set the direction and type of wash between \$washstart and \$washend. The wash is applied according to the current coloring scheme specified by \$colormodel.

**Value:** RGB value.

## \$washtype

**Object:** Any area object.

**Description:** This property specifies the type of wash for an area object when \$areaeffect is set to kGRwash.

**Values:**

kGRnoWashType	No wash effect (the default)
kGRabsingle	a to b, or start to end
kGRabloop	a to b, or start to end looped
kGRabAsingle	a to b to a, or start to end to start
kGRabAloop	a to b to a, or start to end to start looped

## \$xaxistitle

**Description:** The main x-axis title for a graph, normally shown below the graph frame.

## \$y1axistitle

**Description:** The y1-axis title for a graph, normally shown to the left of the graph.

## \$y2axistitle

**Description:** The y2-axis title for a graph, normally shown to the right of the graph.

## \$ztitle

**Description:** The z-axis title for 3D graphs only.

# Index

\$areabackcolor, 66  
\$areacolor, 66  
\$areaeffect, 67  
\$areapattern, 67  
\$autoshaiderisers, 67  
\$bargroupspacing, 68  
\$barriserwidth, 68  
\$colorbyseries, 69  
\$colordivisions, 69  
\$colormodel, 70  
\$curvecolorasseries, 70  
\$curvemoving, 71  
\$curvesmooth, 71  
\$customview, 72  
\$dataformat, 74  
\$dataname, 75  
\$depth, 75  
\$depthimsangle, 76  
\$depthimsthick, 76  
\$depthmode, 76  
\$designgroups, 77  
\$designseries, 77  
\$direction, 77  
\$drawclockwise, 78  
\$excludezero, 78  
\$feelercenter, 79  
\$feelerhorz, 79  
\$feelerradial, 80  
\$font, 80  
\$fontalign, 80  
\$fontcolor, 80  
\$fontdropshadow, 81  
\$fontorient, 81  
\$fontsize, 81  
\$fontstyle, 81  
\$footnote, 81  
\$formatdatatext, 82  
\$formatdtxtx, 82  
\$formatdtxty1, 82  
\$formatdtxty2, 82  
\$formatringtext, 84  
\$formatsdline, 84  
\$formatx, 84  
\$formaty1, 84  
\$formaty2, 84  
\$formatz, 84  
\$graphbywhat, 85  
\$graphconfig, 85  
\$gridlinesordo1, 87  
\$gridlinesx, 88  
\$gridlinesy1, 88  
\$gridlinesy2, 88  
\$gridmodemajoro1, 89  
\$gridmodeminoro1, 90  
\$grouptitle, 91  
\$hlcolor, 91  
\$hlwidth, 91  
\$ignoreseries, 92  
\$insetlgndicon, 92  
\$insetlgndtext, 92  
\$itemdtx, 92  
\$labelmodeo1, 93  
\$labelwraplineso1, 93  
\$labelwrapmodeo1, 93  
\$legenditems, 94  
\$legendlayout, 94  
\$linecolor, 94  
\$linestyle, 94  
\$linewidth, 94  
\$loadtemplate(), 61  
\$locatecoltitle2d, 95  
\$locatefootnote, 95  
\$locateframe, 95  
\$locatelegend, 95  
\$locatelytitle2d, 96  
\$locaterowtitle2d, 96  
\$locaterytitle2d, 96  
\$locatesubtitle, 96  
\$locatetitle, 97  
\$locatetitlex, 97  
\$locatetitley1, 97  
\$locatetitley2, 97  
\$maintitle, 85  
\$majortype, 85  
\$markershape, 98  
\$markershapedefault, 99

\$markersize, 99  
 \$minortype, 86  
 \$socheight, 99  
 \$socwidth, 99  
 \$orientation, 100  
 \$pictflip, 100  
 \$pictname, 100  
 \$pictscale, 101  
 \$piesperrow, 101  
 \$placeo1, 102  
 \$placex, 103  
 \$placey1, 103  
 \$placey2, 103  
 \$presdlconn, 104  
 \$presdlconnbrk, 104  
 \$presdlstep, 104  
 \$presdlstepbrk, 104  
 \$proplevels, 104  
 \$riserthick, 105  
 \$rotate, 105  
 \$savetemplate(), 61  
 \$scalebase, 106  
 \$scaleendx, 107  
 \$scaleendy1, 107  
 \$scaleendy2, 107  
 \$scalefreqbego1, 107  
 \$scalefreqbegx, 108  
 \$scalefreqbegy1, 108  
 \$scalefreqbegy2, 108  
 \$scalefreqendo1, 109  
 \$scalefreqendx, 110  
 \$scalefreqendy1, 110  
 \$scalefreqendy2, 110  
 \$scalefreqo1, 111  
 \$scalefreqx, 112  
 \$scalefreqy1, 112  
 \$scalefreqy2, 112  
 \$scalerangex, 112  
 \$scalerangey, 112  
 \$scalerangez, 112  
 \$scaletype, 113  
 \$scalex, 113  
 \$scaley1, 113  
 \$scaley2, 113  
 \$scatter, 114  
 \$scimovavg, 114  
 \$sddatalinetype, 114  
 \$sdemphasized, 115  
 \$sdlineconn, 115  
 \$sdlinecurv, 116  
 \$sdlinelinrcorr, 117  
 \$sdlinelinrexp, 118  
 \$sdlinelinrformula, 119  
 \$sdlinelinrline, 120  
 \$sdlinelinrlog, 121  
 \$sdlinelinrnatlog, 122  
 \$sdlinelinrnpoly, 123  
 \$sdlinelinrnpolyfac, 123  
 \$sdlinemean, 124  
 \$sdlinemova, 124  
 \$sdlinestdd, 125  
 \$selectobject, 125  
 \$seriesgroupswap, 125  
 \$seriestitle, 125  
 \$seriestype, 126  
 \$shadowoffsets, 126  
 \$showbaseline, 127  
 \$showclose, 127  
 \$showdatatext, 127  
 \$showdivbipolar, 128  
 \$showerrorbar, 128  
 \$showfeeler, 128  
 \$showlblfeeler, 128  
 \$showlblpie, 129  
 \$showlblring, 129  
 \$showlegend, 129  
 \$showoffscale, 129  
 \$showopen, 129  
 \$showshadows, 129  
 \$showsliders, 130  
 \$showtextdatalabels, 130  
 \$showtextdetails, 130  
 \$showwall, 130  
 \$showzeroline, 130  
 \$sideo1, 131  
 \$sidex, 131  
 \$sidey1, 131  
 \$sidey2, 131  
 \$sizeerrorbars, 131  
 \$sliceddelete, 132  
 \$slicemove, 132  
 \$slicerestore, 132  
 \$splity, 132  
 \$squarelgndicons, 133  
 \$staggero1, 133  
 \$staggerx, 133

- \$staggy1, 133
- \$staggy2, 133
- \$subtitle, 133
- \$tilt, 134
- \$tilton, 134
- \$uniformqdrborders, 134
- \$viewmatrix, 134
- \$viewpanx, 135
- \$viewpany, 135
- \$viewposition, 134
- \$viewsz, 135
- \$viewwallthick, 135
- \$washdir, 136
- \$washend, 136
- \$washscale, 136
- \$washstart, 137
- \$washtype, 137
- \$xaxistitle, 137
- \$ylaxistitle, 137
- \$y2axistitle, 137
- \$ztitle, 137

- 2D graph objects, 24
- 3D graph objects, 25
- 3D graphs, 19, 50
- 3D minor type, 23
- 3D risers, 51

- Absolute minor type, 20
- Area graphs, 18
- areabackcolor property, 33
- areacolorrgb property, 33
- areapattern property, 33
- Axes, 38
  - staggered labels, 39
- Axis direction, 38
- Axis labels, 39

- Bar graphs, 17
- bargroupspacing property, 28
- barriserwidth property, 28
- Bipolar minor type, 21

- Changing a graph at runtime, 60
- colorbyseries property, 28
- Colors, 33
- Constants, 26
- Data format, 31
  - for stock market graphs, 59
- Data markers, 45
- Data values and Text, 39
- dataformat property, 13, 31
- dataname property, 13
- depthimsangle property, 28
- depthimsthick property, 28
- depthmode property, 28
- designgroups property, 13
- designseries property, 13
- direction property, 27
- Drag and drop, 64
- Drilldown, 61
- Dual Y minor type, 21

- excludezero property, 27

- fontalign property, 34
- fontcolorrgb property, 34
- fontdropshadow property, 34
- fontorient property, 34
- Fonts, 34
- fontsize property, 34
- fontstyle property, 34
- formatdtxty1 property, 30
- formaty1 property, 30

- Graph Control, 12
- Graph objects, 24
- Graph orientation, 35
- Graph templates, 61
- Graphs
  - Creating a graph, 12
  - Data format, 31
  - Major types, 17
  - Minor types, 20
  - Overview, 5
  - Properties, 26
  - Types of graph, 7, 9
- Grid lines, 48
- gridlinesordol property, 28
- gridlinesy1 property, 28
- gridmodemajorol property, 28
- gridmodeminorol property, 28
- Groups, 10

- labelmodeol property, 30

- Labels, 39, 40
  - labelwraplineso1 property, 30
  - labelwrapmodeo1 property, 30
  - legenditems property, 29
  - legendlayout property, 29
- Legends, 35
- Line graphs, 17
  - linecolorrgb property, 33
  - linepattern property, 33
  - linewidth property, 33
- Locate properties, 33
  - locatefootnote property, 29
  - locateframe property, 29
  - locatelegend property, 29
  - locatesubtitle property, 29
  - locatetitle property, 29
  - locatetitle1 property, 29
- Major graph types, 17
- Major types, 7, 9
  - majorproperty property, 13
- Minor graph types, 20
- minortype property, 13
- Ordinals, 10
- Orientation, 35
  - orientation property, 29
- Overview, 5
- Patterns, 33
- Percent minor type, 22
- Pictures, 46
- Pie charts, 18, 51
- Pie feelers, 53
- Pie graph objects, 25
- Pie minor types, 23
- Pie objects, 52
  - placeo1 property, 30
  - placey1 property, 30
- Positioning objects, 32
- Properties, 26
  - Showing more levels, 26
- proplevels property, 13, 26
- Pseudo 3D, 36
- Riser faces, 51
- Runtime
  - Changing a graph at runtime, 60
- Saving your graph to a file, 61
  - scalebase property, 28
  - scaleendy property, 27
  - scalefreqbego1 property, 27
  - scalefreqbegy1 property, 27
  - scalefreqendo1 property, 27
  - scalefreqendy1 property, 27
  - scalefreqo1 property, 27
  - scalefreqy property, 27
  - scaley1 property, 27
- Scaling, 40
  - scale origin, 41
- Scatter graphs, 55
- selectobject property, 13, 24
- Series, 10
  - seriesgroupswap property, 29
- shadowoffsets property, 29
- showdatatext property, 30
- showlegend property, 29
- showoffscale property, 28
- showshadows property, 29
- showzeroline property, 28
- Side by Side minor type, 22
- sideo1 property, 30
- sidey1 property, 30
- Sizing objects, 32
- Software requirements, 8
- Special graphs, 19, 55
- squarelgnicons property, 29
- Stacked minor type, 20
- staggero1 property, 30
- staggery1 property, 30
- Stock market graphs, 57
- Templates, 61
- Text, 34
- Types of graph, 7, 9
- uniformqdrborders property, 29
- uniformqdrshapes property, 29
- Walls and floor
  - 3D graphs, 50

# How to use this manual


The on-line documentation is designed to make the task of identifying and accessing information about Omnis Studio as easy and intuitive as possible.

You can navigate this PDF document, or find topics, in a number of different ways.

## Bookmarks



Bookmarks mark each topic in a document. To view the bookmarks in this document, click on the Bookmark icon on the Acrobat toolbar or select the **View>>Bookmarks and Page** menu item. In Acrobat Reader 4, you can click on the Bookmarks tab.

Click on an arrow icon  to open or close a topic, and click on a topic name or double-click a page icon  to move directly to a topic.

## Thumbnails



Thumbnails are small images of each page in the document. To view the Thumbnails in this document click on the Thumbnails button on the Acrobat toolbar, or select the **View>>Thumbnails and Page** menu item. In Acrobat Reader 4, you can click on the Thumbnails tab.

You can click on a thumbnail to jump to that page. Also you can adjust the view of the current page by moving and/or sizing the gray page-view box shown on the current thumbnail.

## Browsing



You can use the Browse buttons on the Acrobat toolbar to move back and forth through the document on a page by page basis. You can also click on the **Go Back** to return to your last view or location.

## Find

You can find a text string using the **Tools>>Find** menu item. To find the next occurrence of the text you can use the **Tools>>Find Again** option. If you reach the end of the document, you can use the Ctrl-Home key to go to the beginning and continue your find. See also Search (on the next page of this guide).

## Search

If you have the Acrobat Search plug-in (available under the **Tools>>Search** menu in some versions of Acrobat Exchange and Reader), you can use the Studio Index to perform full-text searches of the entire Omnis Studio on-line documentation set. Searching the Studio Index is much faster than using the **Find** command, which reads every word on every page in the current document only.

To Search the Studio Index, select **Tools>>Search>>Indexes** to locate the Studio index (Index.pdx) on the Omnis CD. Next, select

**Tools>>Search>>Query** to define your search text: you can use Word Stemming, Match Case, Sounds Like, wildcards, and so on (refer to the Acrobat Search.pdf file for details about specifying a query). In the Search Results window, double-click on a document name (the first one probably contains the most references). Acrobat opens the document and highlights the text. To go to the next or previous occurrence of the text, use the Search Next or Search Previous button on the Acrobat toolbar.



## Grabbing Text from the Screen



You can cut and paste text from this document into the clipboard using the Text tool. For example, you could copy a method or code snippet and paste it into the Omnis method editor.

## Getting Help

For more information about using Acrobat Reader see the PDF documents installed with the Reader files, or select the **Help** menu on the main Reader menu bar.