

GraphKit

Reference Guide

Architecture and Performance Group
Apple Computer, Inc.

September 2005

Table of Contents

Table of Contents	
Chapter 1 Introduction	
Overview	7
Architecture	7
Chapter 2 GRChartView	
GRChartView	10
defaultProperties	13
defaultPropertyForKey:	13
setDefaultProperty:ForKey:	13
setDefaultProperties:	14
propertyForKey:	14
setProperty:forKey:	14
setProperties:	14
setDataSource:	14
dataSource	14
reloadData	14
setDelegate:	15
delegate	15
setAxes:	15
axes	15
setTag:	15
tag	15
dataSets	15
addDataSet:loadData:	16
addDataSets:loadData:	16
moveDataSetAtIndex:toIndex:	16
removeDataSet:	16
removeDataSetAtIndex:	16
removeAllDataSets	16
numberOfDataSets	16
numberOfDataSetsOfKindOfClass:	16
indexOfDataSet:	17
classIndexOfDataSet:	17
dataSetOfKindOfClass:	17

autoscale	17
zoomInRect:	17
zoomOut	17
zoomIn:	17
zoomOut:	17
scrollView	17
centerSelection	18
leftJustifySelection	18
rightJustifySelection	18
canvasRect	18
plotRect	18
canvasRectForDataSetAtIndex:	18
computeLayout	18
setNeedsLayout:	18
needsLayout	18
chart:tileFractionForDataSet:atIndex:	18

Chapter 3 GRDataSet

GRDataSet	20
defaultProperties	22
defaultPropertyForKey:	22
setDefaultProperty:forKey:	22
setDefaultProperties:	22
defaultColors	22
setDefaultColors:	23
axesClass	23
propertyForKey:	23
setProperty:forKey:	23
setProperties:	23
initWithOwnerChart:	23
chart	23
setDataSource:	23
dataSource	24
reloadData	24
numberOfElements	24
setDelegate:	24
delegate	24
setIdentifier:	24

identifier	24
setAxes:	25
axes	25
drawLegendSampleInRect:	25
drawDataSetRect:	25
selectedRange	25
setSelectedRange:	25
clearSelectedRange	25
selectPrevious	25
selectNext	26
chart:dataSetSelectionDidChange:range:	26
GRDataSetDataSource	27
chart:numElementsForDataSet:	27
chart:colorForDataSet:element:	27
chart:calloutForDataSet:element:	27
GRXYDataSet	28
xIntervalAtIndex:	29
yValueAtIndex:	29
indexOfXvalue:yValue:exactMatch:	29
GRXYDataSetDataSource	30
chart:yValueForDataSet:element:	30
chart:xIntervalForDataSet:element:	30
GRAreaDataSet	31
GRLineDataSet	32
defaultMarkers	33
setDefaultMarkers:	33
GRColumnDataSet	34
GRPieDataSet	35
drawLegendSampleInRect:forWedgeIndex:	36
indexOfAngle:	36
GRPieDataSetDataSource	37
chart:yValueForDataSet:element:	37
Chapter 4 GRAxes	
 GRAxes	38
defaultProperties	41
defaultPropertyForKey:	41
setDefaultProperty:ForKey:	42

setDefaultProperties:	42
propertyForKey:	42
setProperty:forKey:	42
setProperties:	42
initWithOwner:	42
owner	42
chart	43
setDelegate:	43
delegate	43
setIdentifier:	43
identifier	43
deselectAllPoints	43
selectPoint:byExtendingSelection:	43
clickPoint:	44
xPixelValue	44
yPixelValue	44
xValueAtPoint:	44
yValueAtPoint:	44
locationForXValue:yValue:	44
canvasRect	44
setCanvasRect:	44
setPlotRect	45
legendRect	45
computeLayout	45
setNeedsLayout:	45
needsLayout	45
drawLegendRect:	45
drawGridRect:	45
drawAxesRect:	45
chart:categoryLabelForAxes:index:	45
GRXYAxes	47
computeXMajorMinorUnits	50
computeYMajorMinorUnits	50
drawXAxisRect:	50
drawXGridRect:	50
drawYAxisRect:	50
drawYGridRect:	51

chart.xLabelForAxes:value:defaultLabel:	51
chart.yLabelForAxes:value:defaultLabel:	51
GRPieAxes	52
Chapter 5 GRGradientColor	53
GRGradientColor	53
gradientOfType:withColors:...	54
init	55
initWithColors:...	55
initWithColorArray:	55
addColor:	55
colorCount	55
colors	55
setGradientType:	55
gradientType:	55
fillRect:	56
fillBezierPath:	56
fillBezierPath:withBounds:	56
set:	56
drawSwatchInRect:	56

Chapter 1

Introduction

Overview

GraphKit is a Cocoa framework for programmatically creating 2D line, area, column and pie charts using the Quartz graphics layer of MacOS X. GraphKit's basis in Quartz makes it extremely simple to export GraphKit charts as PDF, TIFF, or other popular data formats for use in the real world. The main design points of GraphKit are:

- Scalability to large datasets (10,000's of data points)
- Clean, colorful output
- Interactive Zooming and Selection
- Object-Oriented, Extensible Design

GraphKit is intended for use in Cocoa programs. This document assumes that the reader is at least somewhat familiar with the Objective-C programming language.

Architecture

Figure 1-1 diagrams the architecture of a GraphKit chart. At its highest level, a GraphKit chart consists of an instance of the `GRChartView` class. The `GRChartView` class manages an arbitrary number of datasets (instances of `GRDataSet` and its subclasses) and controls when and how they will be drawn. In GraphKit, a dataset is a single, related series of data points to be plotted in the same color, line style, etc. In the case of a line chart, for example, a single continuous line represents a dataset. `GRDataSet` and its subclasses (`GRLineDataSet`, `GRAreaDataSet`, `GRColumnDataSet`, and `GRPieDataSet`) are used to contain datasets in GraphKit. A `GRChartView` can display its datasets either overlayed on top of one another or tiled (vertically or horizontally) such that each dataset is displayed on its own set of axes. When in overlayed display mode, the datasets can be displayed independently or cumulatively such that each dataset element is drawn with its values stacked on the corresponding element of previously drawn datasets (total value or fraction of the total). Each set of axes is represented by an instance of `GRAxes` or its subclasses (`GRXYAxes`, `GRPieAxes`). The `GRAxes` classes are responsible for managing the scale and range of datasets as well as drawing any necessary ticks, grid lines, titles, or legends. There is an instance of the `GRAxes` class associated with the `GRChartView`

as well as with each GRDataSet contained in the chart view. In overlaid mode, the GRChartView's GRAxes is used. In tiled mode, the GRAxes of each GRDataSet is used.

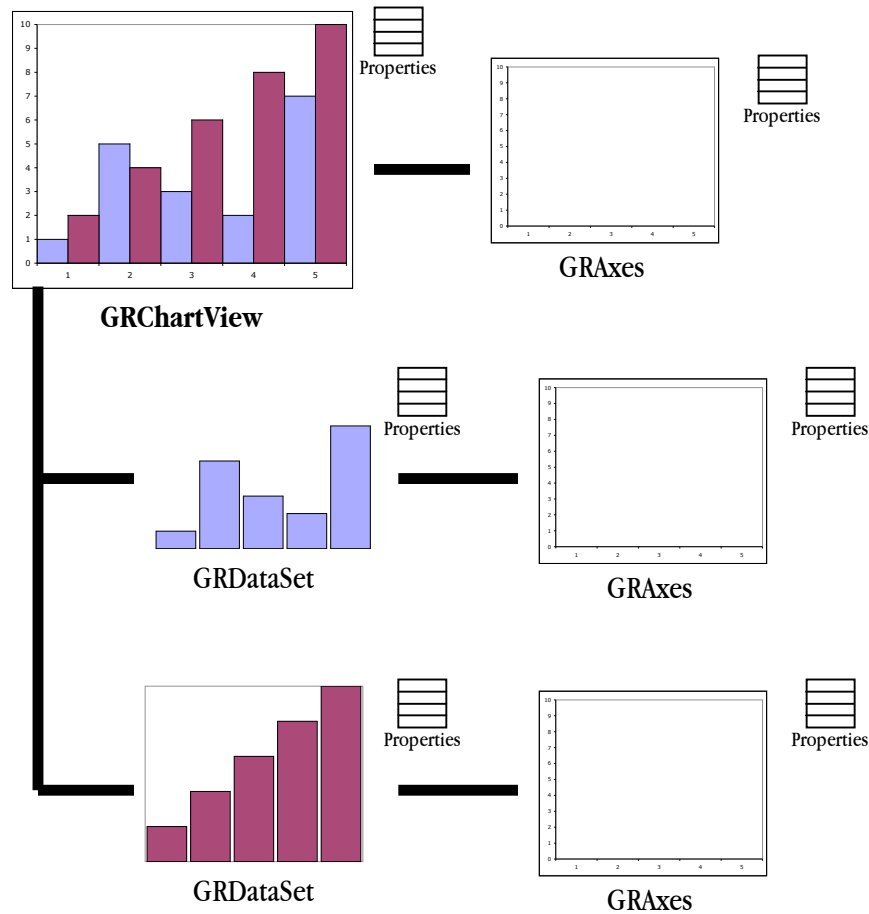


Figure 1-1. GraphKit Framework Architecture

Associated with every object in the GraphKit hierarchy is a properties dictionary. Each properties dictionary contains key-value pairs that are initialized to a default set of entries when a GraphKit object is created. Default properties for a class can be read or modified one at a time with the `+getDefaultPropertyForKey:` and `+setDefaultProperty:ForKey:` methods respectively. The entire default properties dictionary for a class can be retrieved with `+defaultProperties` or replaced with `+setDefaultProperties:`. Instance properties are read with the `-propertyForKey:` and modified with the `-setProperty:ForKey:` method. These methods are available in every GraphKit class, although the keys and values for each of class necessarily differ. Keys are verified by the set property methods to be appropriate for the class or instance being modified.

If you have used ApplicationKit’s `NSTableView` class, then many of the concepts used by GraphKit charts will be familiar to you. Like an `NSTableView`, a GraphKit chart object uses a datasource to ask for the number of datasets, datapoints and values to plot. Also similar to many other AppKit classes, GraphKit utilizes the concept of a “delegate” to optionally customize the behavior of its charts.

Chapter 2

GRChartView

GRChartView

Inherits from:

[NSView](#) : [NSResponder](#) : [NSObject](#)

Conforms to:

NSCoding (NSResponder)

NSObject (NSObject)

Declared in:

GraphKit/GRChartView.h

Class Description

A GRChartView class manages an arbitrary number of datasets and controls when and how they will be drawn to the screen.

Properties

Key	Description
GRChartDrawBackground	If <code>true</code> (NSNumber, bool value) the background of the chart will be filled with the <code>GRChartBackgroundColor</code> .
GRChartBackgroundColor	The color of the chart background – either a single solid color (NSColor) or a gradient (GRGradientColor).
GRChartFrameColor	The color (NSColor) used to draw all grid lines, ticks, etc.
GRChartTitleColor	The color (NSColor) used to draw any title text.
GRChartMainTitle	The string (NSString) to be drawn at the top of the chart using the <code>GRChartMainTitleFont</code> in the <code>GRChartTitleColor</code> .
GRChartMainTitleFont	Font (NSFont) used to draw the <code>GRChartMainTitle</code> .

GRChartAutoscaleFonts	If true (NSNumber, bool value), font sizes will be scaled along with the chart view when it is resized.
GRChartVisualCompression	If 1 (NSNumber, int value), each GRDataSet will attempt to render only visually important datapoints. This can significantly speed up the rendering of large datasets. For larger datasets, 2 provides more aggressive compression to speed up rendering at the cost of some visual details. 0 disables visual compression.
GRChartAllowSelection	If true (NSNumber, bool value), single clicking on the chart will visually select the datapoint in each GRDataSet.
GRChartAllowMultipleSelection	If true (NSNumber, bool value), modifies selection so that multiple datapoints can be selected either by holding down the shift key (contiguous selection) or the option key (multiple selection) while selecting with the mouse.
GRChartAllowHorizontalZoom	If true (NSNumber, bool value), clicking and dragging in the chart will zoom in horizontally (magnify) the corresponding section of the chart. Holding down option while clicking zooms back out one level.
GRChartAllowVerticalZoom	If true (NSNumber, bool value), clicking and dragging in the chart will zoom in vertically (magnify) the corresponding section of the chart. Holding down option while clicking zooms back out one level.
GRChartAllowClick	If true (NSNumber, bool value), allows clicking on data points in the chart.
GRChartDrawValueCallouts	If true (NSNumber, bool value), the value of the datapoint under the mouse will be displayed after a delay.
GRChartIndependentlyZoomTiledDataSets	If true (NSNumber, bool value), and the chart is in vertical or horizontal tiled mode, each dataset will be zoomed independently. Otherwise, all datasets in the chart are zoomed simultaneously.
GRChartIndependentlySelectTiledDataSets	If true (NSNumber, bool value), and the chart is in vertical or horizontal tiled mode, datapoint selection in each dataset will occur independently. Otherwise, the corresponding datapoint in each dataset will be selected.
GRChartLayoutType	<p>The type of layout to use:</p> <p>GRChartOverlayLayout</p> <p>GRChartVerticalTileLayout</p> <p>GRChartHorizontalTileLayout</p> <p>In overlay mode, datasets are rendered on a single set of axes. In tiled mode each dataset is drawn on its own set of axes with axes arranged vertically or horizontally.</p>

GRChartOverlayLayout	<p>How datasets should be overlayed in overlay mode:</p> <p>GRChartIndependentOverlay</p> <p>GRChartStackedValueOverlay</p> <p>GRChartStackedFractionOverlay</p> <p>Overlayed datasets can be drawn independently or stacked to show how each dataset contributes to the total. The total can either be a value or normalized to 1.0 (fraction).</p>
GRChartBorderType	<p>The type of border to be drawn around the chart:</p> <p>GRChartNoBorder</p> <p>GRChartLineBorder</p> <p>GRChartBezelBorder</p> <p>GRChartGrooveBorder</p>

Method Types

Manipulating default properties

- [+ defaultProperties](#)
- [+ defaultPropertyForKey:](#)
- [+ setDefaultProperty:forKey:](#)
- [+ setDefaultProperties:](#)

Manipulating properties

- [- propertyForKey:](#)
- [- setProperty:forKey:](#)
- [- setProperties:](#)

Setting the data source

- [- setDataSource:](#)
- [- dataSource](#)
- [- reloadData](#)

Setting the delegate

- [- setDelegate:](#)
- [- delegate](#)

Retrieving the axes

- [- setAxes:](#)
- [- axes](#)

Setting the tag

- [- setTag:](#)
- [- tag](#)

Manipulating datasets

- [- dataSets](#)
- [- addDataSet:loadData:](#)
- [- addDataSets:loadData:](#)
- [- moveDataSetAtIndex:toIndex:](#)

- [removeDataSet:](#)
- [removeDataSetAtIndex:](#)
- [removeAllDataSets](#)
- [numberOfDataSets](#)
- [numberOfDataSetsOfKindOfClass:](#)
- [indexOfDataSet:](#)
- [classIndexOfDataSet:](#)
- [dataSetOfKindOfClass:atClassIndex:](#)

Zooming

- [autoscale](#)
- [zoomInRect:](#)
- [zoomOut](#)
- [zoomIn:](#)
- [zoomOut:](#)
- [scrollView](#)

Scrolling

- [centerSelection](#)
- [leftJustifySelection](#)
- [rightJustifySelection](#)

Layout

- [canvasRect](#)
- [plotRect](#)
- [canvasRectForDataSetAtIndex:](#)
- [computeLayout](#)
- [setNeedsLayout:](#)
- [needsLayout](#)

For delegates

- [canvasRectForDataSetAtIndex:](#)

Class Methods

defaultProperties

`+(NSDictionary *) defaultProperties`

Returns a copy of the default GRChartView class properties dictionary.

defaultPropertyForKey:

`+(id) defaultPropertyForKey:(NSString *)key`

Returns the default property value associated with *key* or nil if the GRChartView class default properties dictionary does not contain *key*.

setDefaultProperty:ForKey:

`+(BOOL) setDefaultProperty:(id)p forKey:(NSString *)key`

Sets the property value *p* for *key* in the GRChartView class default properties dictionary. Returns YES upon success or NO in the case of an illegal key.

setDefaultProperties:

+(void) setDefaultProperties:(NSDictionary *)pd

Replace the GRChartView class default properties dictionary with *pd*.

Instance Methods

propertyForKey:

-(id) propertyForKey:(NSString *)key

Returns the property value associated with *key* or nil if the properties dictionary does not contain *key*.

setPropertyForKey:

-(BOOL) setProperty:(id)p forKey:(NSString *)key

Sets the property value *p* for *key* in the properties dictionary. Returns YES upon success or NO in the case of an illegal key.

setProperties:

-(void) setProperties:(NSDictionary *)pd

Replace the GRChartView instance properties dictionary with *pd*.

setDataSource:

-(void) setDataSource:(id)anObject

Sets the receiver's data source to *anObject*. *anObject* should implement the appropriate methods of the [GRChartDataSource](#) informal protocol.

This method raises an `NSInternalInconsistencyException` if *anObject* doesn't respond to either `chart:numElementsInDataSet:` or `chart:dataset:yValueAtIndex:`.

See Also: [-dataSource](#)

dataSource

-(id) dataSource

Returns the object that provides the data displayed by the receiver. See the [GRChartDataSource](#) informal protocol specification for more information.

See Also: [-setDataSource:](#)

reloadData

-(void) reloadData

Marks the receiver as needing redisplay, so it will instruct each dataset to reload the new data and then redraw the datasets.

setDelegate:

-(void) **setDelegate:**(id) *anObject*

Sets the receiver's delegate to *anObject*.

See Also: - [delegate](#)

delegate

-(id) **delegate**

Returns the receiver's delegate.

See Also: - [setDelegate:](#)

setAxes:

-(void) **setAxes:**(GRAxes *) *axes*

Sets the receiver's axes to *axes*. The GRChartView's axes are used in overlay mode. If the axes of the first GRDataSet to be added to a chart does not match those of the GRChartView, the GRChartView's axes are recreated based on the dataset's axes type. This is done in order to ensure compatibility between the chart's axes (used for overlay) and its datasets.

See Also: - [axes](#)

axes

-(GRAxes *) **axes**

Returns the receiver's axes. The GRChartView's axes are used in overlay mode. There is no corresponding setAxes: method because the GRChartView's axes are instantiated automatically when the first GRDataSet is added to the chart (the same class as the axes of this first dataset). This is done in order to ensure compatibility between the chart's axes (used for overlay) and its datasets.

setTag:

-(void) **setTag:**(int) *aTag*

Sets the receiver's tag to *aTag*, an integer that you can use to identify view objects in your application.

See Also: - [tag](#)

tag

-(int) **tag**

Returns the receiver's tag, an integer that you can use to identify view objects in your application.

See Also: - [setTag:](#)

dataSets

-(NSArray *) **dataSets**

Returns an array containing all of the GRDataSets in the receiver.

addDataSet:loadData:

```
-(void) addDataSet:(GRDataSet *)aDataSet
loadData:(BOOL)load
```

Adds *aDataSet* as the last dataset of the receiver. If *aDataSet* is the first dataset to be added, the correct GRAxes class will be automatically allocated. If *load* is YES, the data are reloaded, and the GRChartView is redrawn.

See Also: [-removeDataSet:](#)

addDataSets:loadData:

```
-(void) addDataSets:(NSArray *)arr loadData:(BOOL)loadData
```

Adds all of the datasets in *arr* as the last datasets of the receiver (same order as array). If *loadData* is YES, the data are reloaded, and the GRChartView is redrawn.

See Also: [-removeDataSet:](#)

moveDataSetAtIndex:toIndex:

```
-(void)moveDataSetAtIndex:(int)oldIndex
toIndex:(int)newIndex
```

Moves the dataset at *oldIndex* to *newIndex*.

removeDataSet:

```
-(void) removeDataSet:(GRDataSet *)aDataSet
```

Removes *aDataSet* from the receiver.

See Also: [-addDataSet:](#)

removeDataSetAtIndex:

```
-(void) removeDataSetAtIndex:(int)index
```

Removes dataset at index *index* from the receiver.

removeAllDataSets

```
-(void) removeAllDataSets
```

Removes all datasets from the receiver.

numberOfDataSets

```
-(int) numberOfDataSets
```

Returns the number of datasets in the receiver.

numberOfDataSetsOfClass:

```
-(int) numberOfDataSetsOfClass:(Class)aClass
```

Returns the number of GRDataSets in the receiver that are a member of *aClass* or its subclasses.

indexOfDataSet:

– (void) **indexOfDataSet:** (GRDataSet *) *aDataSet*

Returns index of *aDataSet*.

classIndexOfDataSet:

– (int) **classIndexOfDataSet:** (GRDataSet *) *aDataSet*

Returns index of *aDataSet* for all classes of the same class or subclasses of *aDataSet*.

dataSetOfKindOfClass:

– (GRDataSet *) **dataSetOfKindOfClass:** (Class) *aClass*
atClassIndex: (int) *index*

Returns the GRDataSet in the receiver that are a member of *aClass* or its subclasses at the specified *index*.

autoscale

– (BOOL) **autoscale**

Autoscales all the GRDataSets associated with the receiver.

zoomInRect:

– (BOOL) **zoomInRect:** (CGRect) *zoomRect*

Returns NO if all the associated GRDataSets could not zoom to the given rectangle

zoomOut

– (BOOL) **zoomOut**

Returns NO if the receiver could not return to its normal zoom.

zoomIn:

– (IBAction) **zoomIn:** (id) *sender*

Action that zooms in the receiver by 20%.

zoomOut:

– (IBAction) **zoomOut:** (id) *sender*

Action that calls **zoomOut**.

scrollView

– (NSScrollView *) **scrollView**

Returns the NSScrollView associated with the receiver.

centerSelection

–(void) **centerSelection**
 Causes the axes to center the selection.

leftJustifySelection

–(void) **leftJustifySelection**
 Causes the axes to left justify the selection.

rightJustifySelection

–(void) **rightJustifySelection**
 Causes the axes to right justify the selection.

canvasRect

–(NSRect) **canvasRect**
 Returns the NSRect upon which the chart is drawn.

plotRect

–(NSRect) **plotRect**
 Returns the NSRect upon which the plot itself is drawn.

canvasRectForDataSetAtIndex:

–(NSRect) **canvasRectForDataSetAtIndex:** (int) *dataSetIndex*
 Returns the canvas rectangle (bounds) of the dataset at index *dataSetIndex*.

computeLayout

–(BOOL) **computeLayout**
 Returns YES if the chart layout was successfully recomputed.

setNeedsLayout:

–(void) **setNeedsLayout:** (BOOL) *b*
 Mark the receiver as needing to recompute its layout

needsLayout

–(BOOL) **needsLayout**
 Returns YES if the receiver needs to recompute its layout.

Methods implemented by the Delegate

chart:tileFractionForDataSet:atIndex:

–(double) **chart:** (GRChartView *) *aChart*
tileFractionForDataSet: (GRDataSet *) *aDataSet*
atIndex: (int) *index*

Returns the fraction of the total chart height or width to be used for drawing the dataset *aDataSet* at index *index*. This delegate is intended to allow for customized tiling behavior (e.g. one of the tiled

datasets can be smaller or larger than the rest). The total of fractions for all datasets must add up to 1.0.

Chapter 3

GRDataSet

GRDataSet

Inherits from:

[NSObject](#)

Conforms to:

NSObject (NSObject)

Declared in:

GraphKit/GRDataSet.h

Class Description

A GRDataSet object manages and draws a set of related datapoints. The datapoints supplied by the data source are cached internally for efficiency. The GRDataSet class is an abstract class; it does not define how datapoints are to be cached or drawn. Subclasses of the GRDataSet class (GRLineDataSet, GRAreaDataSet, GRColumnDataSet, GRPieDataSet) implement specific caching and drawing methods.

Properties

Key	Description
GRDataSetAutoPlotColor	If true (NSNumber, bool value), the dataset will be assigned a plot color automatically when it is created.
GRDataSetAutoSelectionColor	If true (NSNumber, bool value), the dataset will be assigned a selection color automatically when it is created.
GRDataSetPlotColor	The color (NSColor) used to draw the dataset. This property is set automatically to a unique color (from the defaultColors array) when the GRDataSet is created).
GRDataSetSelectionColor	The color (NSColor) used to draw the selection line.
GRDataSetSelectionLineWidth	The width (NSNumber) of the line used to highlight a selection.
GRDataSetDrawPlotOutline	If true (NSNumber, bool value), the plot will be drawn with an outline.
GRDataSetPlotOutlineColor	The color (NSColor) used to draw the plot outline.

GRDataSetLegendLabel	If non-empty (NSString), the label text displayed in the legend.
GRDataSetHidden	If true (NSNumber, bool value), the dataset will not be drawn.
GRDataSetInheritChartDataSource	If true (NSNumber, bool value), the dataset will inherit its owner chart's datasource. If the owner chart's datasource is changed, the dataset's datasource will change as well.
GRDatasetInheritChartDelegate	If true (NSNumber, bool value), the dataset will inherit its owner chart's delegate. If the owner chart's delegate is changed, the dataset's delegate will change as well.
GRDataSetDrawShadow	If true (NSNumber, bool value), the dataset will be drawn with a shadow.
GRDataSetShadowOffset	The number of pixels (NSNumber) to offset the shadow from the dataset.
GRDataSetShadowAngle	The angle (NSNumber) at which to project the drop shadow.
GRDataSetShadowBlur	The amount of blur to apply to the drop shadow (NSNumber).
GRDataSetSumValue	(NSNumber)

Method Types

Manipulating default properties

[+ defaultProperties](#)
[+ defaultPropertyForKey:](#)
[+ setDefaultProperty:forKey:](#)
[+ setDefaultProperties:](#)
[+ defaultColors](#)
[+ setDefaultColors:](#)
[+ axesClass](#)

Manipulating properties

[- propertyForKey:](#)
[- setProperty:forKey:](#)
[- setProperties:](#)

Creating a dataset

[- initWithOwnerChart:](#)
[- chart](#)

Setting the data source

[- setDataSource:](#)
[- dataSource](#)

Loading data

[- reloadData](#)

[- numberOfElements](#)

Setting the delegate

[- setDelegate:](#)

[- delegate](#)

Setting the identifier

[- setIdentifier:](#)

[- identifier](#)

Setting the axes

[- setAxes:](#)

[- axes](#)

Drawing

[- drawLegendSampleInRect:](#)

[- drawDataSetRect:](#)

Selection

[- selectedRange](#)

[- setSelectedRange:](#)

[- clearSelectedRange](#)

[- selectPrevious](#)

[- selectNext](#)

Class Methods

defaultProperties

`+(NSDictionary *) defaultProperties`

Returns a copy of the default GRDataSet class properties dictionary.

defaultPropertyForKey:

`+(id) defaultPropertyForKey:(NSString *)key`

Returns the default property value associated with *key* or nil if the GRDataSet class default properties dictionary does not contain *key*.

setDefaultProperty:forKey:

`+(BOOL) setDefaultProperty:(id)p forKey:(NSString *)key`

Sets the property value *p* for *key* in the GRDataSet class default properties dictionary. Returns YES upon success or NO in the case of an illegal key.

setDefaultProperties:

`+(void) setDefaultProperties:(NSDictionary *)aDictionary`

Replace the GRDataSet class default properties dictionary with *aDictionary*.

defaultColors

`+(NSArray *) defaultColors`

Returns a copy of the default GRDataSet class colors dictionary.

setDefaultColors:

```
+(void) setDefaultColors:(NSArray *)anArray
```

Replace the GRDataSet class default colors dictionary with *anArray*.

axesClass

```
+(Class) axesClass
```

Returns the class of the preferred GRAxes subclass. For example, if the dataset requires a GRXYAxes class, it should return [GRXYAxes class]. This method is used by initializer methods to allocate the correct type of axes as well as ensure that incompatible GRDataSets are not mixed.

Instance Methods**propertyForKey:**

```
-(id) propertyForKey:(NSString *)key
```

Returns the property value associated with *key* or nil if the properties dictionary does not contain *key*.

setProperty:forKey:

```
-(BOOL) setProperty:(id)p forKey:(NSString *)key
```

Sets the property value *p* for *key* in the properties dictionary. Returns YES upon success or NO in the case of an illegal key.

setProperties:

```
+(void) setProperties:(NSDictionary *)aDictionary
```

Replace the GRDataSet instance properties dictionary with *aDictionary*.

initWithOwnerChart:

```
-(id) initWithOwnerChart:(GRChartView *)aChartView
```

Initializes a newly created GRDataSet with *aChartView* as its owner. This method is the designated initializer for the GRDataSet class. Returns *self*.

chart

```
-(GRChartView *) chart
```

Returns the owner chart.

setDataSource:

```
-(void) setDataSource:(id)anObject
```

Sets the receiver's data source to *anObject*. *anObject* should implement the appropriate methods of the [GRChartDataSource](#) informal protocol.

This method raises an `NSInternalInconsistencyException` if *anObject* doesn't respond to either `chart:numElementsInDataSet:` or `chart:dataset:yValueAtIndex:`.

See Also: - [dataSource](#)

dataSource

-(id) **dataSource**

Returns the object that provides the data displayed by the receiver. See the [GRChartDataSource](#) informal protocol specification for more information.

See Also: - [setDataSource:](#)

reloadData

-(void) **reloadData**

Marks the receiver as needing redisplay, so it will instruct each dataset to reload the new data and then redraw the datasets.

numberOfElements

-(int) **numberOfElements**

Returns the number of datapoints (elements) in the dataset.

setDelegate:

-(void) **setDelegate:(id) anObject**

Sets the receiver's delegate to *anObject*.

See Also: - [delegate](#)

delegate

-(id) **delegate**

Returns the receiver's delegate.

See Also: - [setDelegate:](#)

setIdentifier:

-(void) **setIdentifier:(id) anObject**

Sets the receiver's identifier to *anObject*. This object is used by the data source to identify the attribute corresponding to the GRDataSet.

See Also: - [identifier](#)

identifier

-(id) **identifier**

Returns the object used by the data source to identify the attribute corresponding to the receiver.

See Also: - [setIdentifier:](#)

setAxes:

-(void) **setAxes:(GRAxes *)** *axes*

Sets the receiver's axes to *axes*. The GRDataSet's axes are used in tile mode. If the GRDataSet is the first one to be added to a chart, the GRChartView's axes are generated based on its axes type. This is done in order to ensure compatibility between the chart's axes (used for overlay) and its datasets.

See Also: - [axes](#)

axes

-(GRAxes *) **axes**

Returns the receiver's axes. The GRDataSet's axes are used in tile mode.

drawLegendSampleInRect:

-(void) **drawLegendSampleInRect:(NSRect)** *aRect*

Draws a sample of the dataset's plotting method (color, line width, etc.) inside of *aRect*. This is used when drawing the legend.

drawDataSetRect:

-(void) **drawDataSetRect:(NSRect)** *aRect*

Called to draw the dataset's datapoints inside of *aRect* as specified by the current GRAxes.

selectedRange

-(NSRange) **selectedRange**

Returns the range of selected datapoint indices.

setSelectedRange:

-(BOOL) **setSelectedRange:(NSRange)** *r*

Sets the dataset's range of selected datapoint indices to [*r.location*..*r.location* + *r.length*). Returns NO if the supplied range is not valid for the dataset.

clearSelectedRange

-(void) **clearSelectedRange**

Clears the range of selected datapoint indices.

selectPrevious

-(BOOL) **selectPrevious**

Slides the dataset's range of previously selected datapoint indices down by one place. Returns NO if the range is already at the start of the dataset, or if the selected range only includes one datapoint.

selectNext

-(BOOL) **selectNext**

Slides the dataset' range of previously selected datapoint indices up by one place. Returns NO if the range is already at the end of the dataset, or if the selected range only includes one datapoint

Methods implemented by the Delegate

chart:dataSetSelectionDidChange:range:

-(void) **chart:**(GRChartView *)*aChart*
dataSetSelectionDidChange:(GRDataSet *)*aDataSet*
range:(NSRange) *selectedRange*

Notifies a GRDataSetDelegate that the selection has been changed.

GRDataSetDataSource

Adopted By:

[NSObject](#)

Declared in:

GraphKit/GRDataSet.h

Protocol Description

The GRDataSetDataSource informal protocol declares the methods that a GRDataSet uses to access the contents of its data source object.

Method Types

GettingValues

- [chart:numElementsForDataSet:](#)

- [chart:colorForDataSet:element:](#)

- [chart:calloutForDataSet:element:](#)

Instance Methods

chart:numElementsForDataSet:

```
-(int) chart:(GRChartView *)aChart
numElementsForDataSet:(GRDataSet *)aDataSet;
```

Returns the number of datapoints managed for *aDataSet* in *aChart* by the data source object. A GRDataSet uses this method to determine how many datapoints it should retain and display.

chart:colorForDataSet:element:

```
-(NSColor *) chart:(GRChartView *)aChart
colorForDataSet:(GRDataSet *)aDataSet element:(int)index
```

Returns the color (NSColor) of the element at position *index* in *aDataSet*.

chart:calloutForDataSet:element:

```
-(NSString *) chart:(GRChartView *)aChart
calloutForDataSet:(GRDataSet *)aDataSet element:(int)index
```

Returns the callout (NSString) of the element at position *index* in *aDataSet*.

GRXYDataSet

Inherits from:[GRDataSet: NSObject](#)**Conforms to:**

NSObject (NSObject)

Declared in:

GraphKit/GRXYDataSet.h

Class Description

The GRXYDataSet class is a subclass of GRDataSet used to manage all of the common properties of datasets plotted on GRXYAxes. The GRXYDataSet class is an abstract class; it does not define how datapoints are to be cached or drawn. Subclasses of the GRXYDataSet class (GRLineDataSet, GRAreaDataSet, GRColumnDataSet) implement specific caching and drawing methods.

Properties

Key	Description
GRDataSetXMin	The minimum x value of the dataset. This is used by the GRXYAxes class when computing major and minor units.
GRDataSetXMax	The maximum x value of the dataset. This is used by the GRXYAxes class when computing major and minor units.
GRDataSetYMin	The minimum y value of the dataset. This is used by the GRXYAxes class when computing major and minor units.
GRDataSetYMax	The maximum y value of the dataset. This is used by the GRXYAxes class when computing major and minor units.

Method Types

Loading data

[- xIntervalAtIndex:](#)[- yValueAtIndex:](#)[- indexOfXvalue:yValue:exactMatch:](#)

Instance Methods

xIntervalAtIndex:

-(GRInterval) **xIntervalAtIndex:(int) index**

Returns the x interval of the datapoint at *index*. Raises an `NSInvalidArgumentException` and returns (0.0, 0.0) if the index is out of range.

yValueAtIndex:

-(double) **yValueAtIndex:(int) index**

Returns the y value of the datapoint at *index*. Raises an `NSInvalidArgumentException` and returns 0.0 if the index is out of range.

indexOfXvalue:yValue:exactMatch:

-(int) **indexOfXvalue:(double)x yValue:(double)y
exactMatch:(BOOL) exact**

Not implemented in the abstract class, returns -1.

GRXYDataSetDataSource

Adopted By:

[NSObject](#) (informal protocol)

Declared in:

GraphKit/GRXYDataSet.h

Protocol Description

The GRXYDataSetDataSource informal protocol declares the methods that a GRXYDataSet uses to access the contents of its data source object.

Method Types

GettingValues

- [chart:yValueForDataSet:element:](#)

- [chart:xIntervalForDataSet:element:](#)

Instance Methods

chart:yValueForDataSet:element:

-(double) **chart:**(GRChartView *)*aChart*

yValueForDataSet:(GRDataSet *)*aDataSet* **element:**(int)*index*

Returns the y value at index *index* of the datapoint managed for *aDataSet* in *aChart* by the data source object.

chart:xIntervalForDataSet:element:

-(GRInterval) **chart:**(GRChartView *)*aChart*

xIntervalForDataSet:(GRDataSet *)*aDataSet*

element:(int)*index*

Returns the x interval (begin, end) at index *index* of the datapoint managed for *aDataSet* in *aChart* by the data source object. This function is optional.

GRAreaDataSet

Inherits from:

[GRXYDataSet](#); [GRDataSet](#); [NSObject](#)

Conforms to:

NSObject (NSObject)

Declared in:

GraphKit/GRAreaDataSet.h

Class Description

The GRAreaDataSet class is a subclass of GRXYDataSet used to manage and draw a dataset as an area chart.

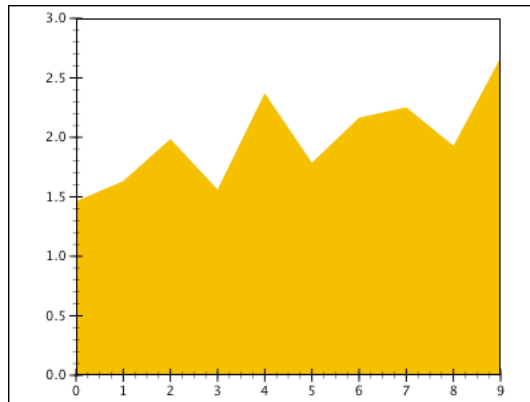


Figure 3-1. GRAreaDataSet

GRLineDataSet

Inherits from:[GRXYDataSet](#); [GRDataSet](#); [NSObject](#)**Conforms to:**

NSObject (NSObject)

Declared in:

GraphKit/GRLineDataSet.h

Class Description

The GRLineDataSet class is a subclass of GRXYDataSet used to manage and draw a dataset as line chart.

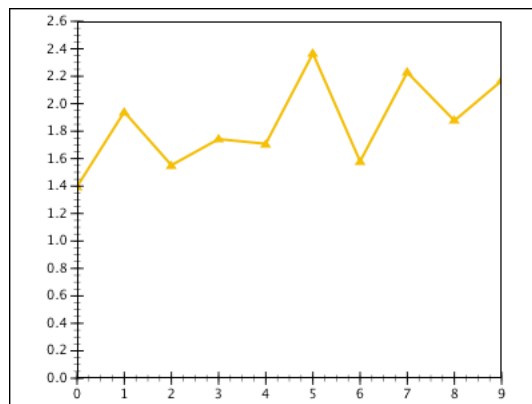


Figure 3-2. GRLineDataSet

Properties

Key	Description
<code>GRDataSetAutoMarkerColor</code>	If true (NSNumber, bool value), the dataset will be assigned a marker color automatically when it is created.
<code>GRDataSetAutoMarkerGlyph</code>	If true (NSNumber, bool value), the dataset will be assigned a glyph automatically when it is created.
<code>GRDataSetMarkerFont</code>	The font (NSFont) used to draw datapoint markers on the line chart.
<code>GRDataSetMarkerColor</code>	The color (NSColor) used to draw datapoint markers on the line chart.

GRDataSetMarkerGlyph	The glyph or symbol (NSString) drawn as a datapoint marker on the line chart.
GRDataSetDrawMarkers	If true (NSNumber, bool value) markers will be drawn for each datapoint in the dataset.
GRDataSetDrawPlotLine	If true (NSNumber, bool value) a line will be drawn to connect datapoints.
GRDataSetPlotLineWidth	The width (NSNumber) in pixels of the line used to connect datapoints.
GRDataSetPlotLineDashPattern	If the array (NSArray) contains at least one pair of numbers (NSNumbers), the plot line will be drawn with the pattern specified by on pixels/off pixels sequence. For Example, a pattern {2, 3, 5, 7} would create a pattern of 2 pixels on, 3 pixels off, 5 pixels on, 7 pixels off.
GRLineDataSetPlotStyle	One of the below types (NSString): GRLineDataSetPlotStroke GRLineDataSetPlotFill

Method Types

Loading data

[+ defaultMarkers](#)

[+ setDefaultMarkers:](#)

Class Methods

defaultMarkers

+(NSArray *) defaultMarkers

Returns a copy of the default GRLineDataSet class markers dictionary.

setDefaultMarkers:

+(void) setDefaultMarkers:(NSArray *) anArray

Replace the GRLineDataSet class default markers dictionary with *anArray*.

GRColumnDataSet

Inherits from:[GRXYDataSet](#); [GRDataSet](#); [NSObject](#)**Conforms to:**

NSObject (NSObject)

Declared in:

GraphKit/GRColumnDataSet.h

Class Description

The GRColumnDataSet class is a subclass of GRXYDataSet used to manage and draw a dataset as a column (vertical bar) chart.

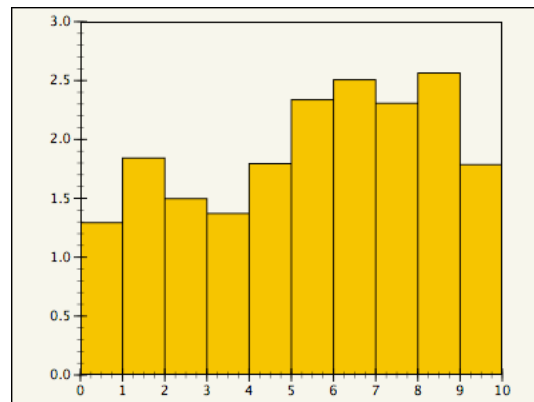


Figure 3-3. GRColumnDataSet

Properties

Key	Description
GRDataSetCategoryGapFraction	The amount of horizontal space (NSNumber) between columns specified as a fraction (0..1.0) of maximum column width.

GRPieDataSet

Inherits from:[GRDataSet: NSObject](#)**Conforms to:**

NSObject (NSObject)

Declared in:

GraphKit/GRPieDataSet.h

Class Description

The GRPieDataSet class is a subclass of GRDataSet used to manage and draw a dataset as a pie chart.

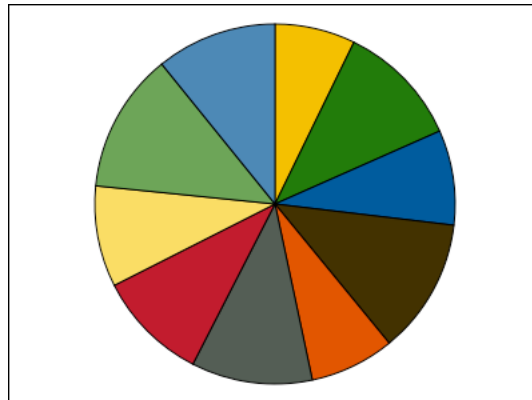


Figure 3-4. GRPieDataSet

Properties

Key	Description
GRDataSetPieStartAngle	The angle at which pie wedges will start being drawn clockwise.

Method Types

Drawing[-drawLegendSampleInRect:forWedgeIndex:](#)

Loading data
[- indexOfAngle:](#)

Instance Methods

drawLegendSampleInRect:forWedgeIndex:

**-(void) drawLegendSampleInRect:(NSRect) aRect
forWedgeIndex:(int) index**

Draws a sample of the dataset's plotting method (color, line width, ...) inside of *aRect* for the wedge at *index*. This is used when drawing the legend.

indexOfAngle:

-(int) indexOfAngle:(double) a

Returns the index of the datapoint (element) that contains the angle *a*. This is used internally for datapoint selection with the mouse.

GRPieDataSetDataSource

Adopted By:

[NSObject](#) (informal protocol)

Declared in:

GraphKit/GRPieDataSet.h

Protocol Description

The GRPieDataSetDataSource informal protocol declares the methods that a GRPieDataSet uses to access the contents of its data source object.

Method Types

GettingValues

- [chart:yValueForDataSet:element:](#)

Instance Methods

chart:yValueForDataSet:element:

```
-(double) chart:(GRChartView *)aChart  
yValueForDataSet:(GRDataSet *)aDataSet element:(int)index
```

Returns the y value at index *index* of the datapoint managed for *aDataSet* in *aChart* by the data source object.

Chapter 4

GRAxes

GRAxes

Inherits from:

[NSObject](#)

Conforms to:

NSObject (NSObject)

Declared in:

GraphKit/GRAxes.h

Class Description

The GRAxes class is responsible for determining the scale and range of values to be plotted as well as drawing any necessary ticks, grid lines, titles, or legends. There is an instance of the GRAxes class associated with the GRChartView as well as with each GRDataSet contained in the chart view. In overlayed mode, the GRChartView's GRAxes is used. In tiled mode, the GRAxes of each GRDataSet is used.

Properties

Key	Description
GRAxesBackgroundColor	The color of the chart background – either a single solid color (NSColor) or a gradient (GRGradientColor).
GRAxesDrawBackground	If true (NSNumber, bool value), the background of the axes will be filled with the GRAxesBackgroundColor
GRAxesLabelFont	Font (NSFont) used to draw axes labels.
GRAxesSubTitleFont	Font (NSFont) used to draw the GRAxesSubTitle.
GRAxesSubTitle	This string (NSString) will be drawn at the top of the chart using the GRAxesSubTitleFont in the GRAxesTitleColor.
GRAxesMajorLineWidth	The width (NSNumber) in pixels used to draw major lines.

GRAxesMinorLineWidth	The width (NSNumber) in pixels used to draw minor lines.
GRAxesMajorLineDashPattern	If this array (NSArray) contains at least one NSNumber pair, major lines will be drawn with the pattern specified by on pixels/off pixels sequence. For Example, a pattern {2, 3, 5, 7} would create a pattern of 2 pixels on, 3 pixels off, 5 pixels on, 7 pixels off.
GRAxesMinorLineDashPattern	If this array (NSArray) contains at least one NSNumber pair, minor lines will be drawn with the pattern specified by on pixels/off pixels sequence. For Example, a pattern {2, 3, 5, 7} would create a pattern of 2 pixels on, 3 pixels off, 5 pixels on, 7 pixels off.
GRAxesMajorLineColor	The color (NSColor) used to draw major lines.
GRAxesMinorLineColor	The color (NSColor) used to draw minor lines.
GRAxesMaxPrecision	The maximum allowable number (NSNumber) of digits to use for axis labels.
GRAxesMajorTickLength	Length of major ticks (NSNumber) in pixels.
GRAxesMinorTickLength	Length of minor ticks (NSNumber) in pixels.
GRAxesMinNonScientificValue	The minimum absolute value (NSNumber), below which scientific notation will be used.
GRAxesMaxNonScientificValue	The maximum absolute value (NSNumber) above which scientific notation will be used.
GRAxesLeftMargin	The minimum left margin (NSNumber) between the plot area and the border of the axes in pixels.
GRAxesRightMargin	The minimum right margin (NSNumber) between the plot area and the border of the axes in pixels.
GRAxesTopMargin	The minimum top margin (NSNumber) between the plot area and the border of the axes in pixels.
GRAxesBottomMargin	The minimum bottom margin (NSNumber) between the plot area and the border of the axes in pixels.
GRAxesXPlotMin	The minimum X value (NSNumber) to be plotted.
GRAxesXPlotMax	The maximum X value (NSNumber) to be plotted.
GRAxesYPlotMin	The minimum Y value (NSNumber) to be plotted.
GRAxesYPlotMax	The maximum Y value (NSNumber) to be plotted.
GRAxesFixedXPlotMin	
GRAxesFixedXPlotMax	
GRAxesFixedYPlotMin	

GRacesFixedYPlotMax	
GRAxesBorderType	<p>The type of border to be drawn around the axes:</p> <p>GRAxesNoBorder</p> <p>GRAxesLineBorder</p> <p>GRAxesBezelBorder</p> <p>GRAxesGrooveBorder</p>
GRAxesDrawLegend	<p>If true (NSNumber, bool value), a legend will be drawn.</p>
GRAxesDrawLegendBackground	<p>If true (NSNumber, bool value), the background of the legend will be filled with the GRAxesLegendBackgroundColor.</p>
GRAxesLegendBackgroundColor	<p>The color of the legend background – either a single solid color (NSColor) or a gradient (GRGradientColor).</p>
GRAxesLegendFont	<p>The font used for legend labels.</p>
GRAxesLegendPosition	<p>The placement of the legend relative to the axes:</p> <p>GRAxesLegendTopPosition</p> <p>GRAxesLegendBottomPosition</p> <p>GRAxesLegendLeftPosition</p> <p>GRAxesLegendRightPosition</p>
GRAxesLegendBorderType	<p>The type of border to be drawn around the legend:</p> <p>GRAxesLegendNoBorder</p> <p>GRAxesLegendLineBorder</p> <p>GRAxesLegendBezelBorder</p> <p>GRAxesLegendGrooveBorder</p>
GRAxesInheritOwnerDelegate	<p>If true (NSNumber, BOOL), GRAxes inherits delegate from parent.</p>

Method Types

Manipulating default properties

[+ defaultProperties](#)

[+ defaultPropertyForKey:](#)

[+ setDefaultProperty:forKey:](#)

[+ setDefaultProperties:](#)

Manipulating properties

- [propertyForKey:](#)
- [setProperty:forKey:](#)
- [setProperties:](#)

Creating a dataset

- [initWithOwner:](#)
- [owner](#)
- [chart](#)

Setting the delegate

- [setDelegate:](#)
- [delegate](#)

Setting the identifier

- [setIdentifier:](#)
- [identifier](#)

Selection

- [deselectAllPoints](#)
- [selectPoint:byExtendingSelection:](#)
- [clickPoint:](#)

Value/Location Conversions

- [xPixelValue](#)
- [yPixelValue](#)
- [xValueAtPoint:](#)
- [yValueAtPoint:](#)
- [locationForXValue:yValue:](#)

Layout

- [canvasRect](#)
- [setCanvasRect:](#)
- [plotRect](#)
- [setPlotRect:](#)
- [legendRect](#)
- [computeLayout](#)
- [setNeedsLayout:](#)
- [needsLayout](#)

Drawing

- [drawLegendRect:](#)
- [drawGridRect:](#)
- [drawAxesRect:](#)

Class Methods

defaultProperties

+(NSDictionary *) defaultProperties

Returns a copy of the default GRAxes class properties dictionary.

defaultPropertyForKey:

+(id) defaultPropertyForKey:(NSString *)key

Returns the default property value associated with *key* or nil if the GRAxes class default properties dictionary does not contain *key*.

setDefaultProperty:ForKey:

```
+(BOOL) setDefaultProperty:(id)p forKey:(NSString *)key
```

Sets the property value *p* for *key* in the GRAxes class default properties dictionary. Returns YES upon success or NO in the case of an illegal key.

setDefaultProperties:

```
+(void) setDefaultProperties:(NSDictionary *)aDictionary
```

Replace the GRAxes class default properties dictionary with *aDictionary*.

Instance Methods

propertyForKey:

```
-(id) propertyForKey:(NSString *)key
```

Returns the property value associated with *key* or nil if the properties dictionary does not contain *key*.

setProperty:forKey:

```
-(BOOL) setProperty:(id)p forKey:(NSString *)key
```

Sets the property value *p* for *key* in the properties dictionary. Returns YES upon success or NO in the case of an illegal key.

setProperties:

```
-(void) setProperties:(NSDiction *)aDict
```

Replace the GRAxes class properties dictionary with *aDict*.

initWithOwner:

```
-(id) initWithOwner:(id)anObject
```

Initializes a newly created GRAxes with *anObject* (must be GRChartView or GRDataSet) as its owner. This method is the designated initializer for the GRAxes class. Returns *self*.

owner

```
-(id) owner
```

Returns the owner GRChartView or GRDataSet.

chart

-(GRChartView *) **chart**

Returns the owner GRChartView.

setDelegate:

-(void) **setDelegate:**(id) *anObject*

Sets the receiver's delegate to *anObject*.

See Also: [- identifier](#)

delegate

-(id) **delegate**

Returns the receiver's delegate.

setIdentifier:

-(void) **setIdentifier:**(id) *anObject*

Sets the receiver's identifier to *anObject*. This object is used by the data source and delegate to identify the attribute corresponding to the GRAxes.

See Also: [- identifier](#)

identifier

-(id) **identifier**

Returns the object used by the data source and delegate to identify the attribute corresponding to the receiver.

deselectAllPoints

-(BOOL) **deselectAllPoints**

Sets the selected range for each dataset drawn on the axes to be [0,0) or no selection. Not implemented in the abstract class.

selectPoint:byExtendingSelection:

-(BOOL) **selectPoint:**(NSPoint) *aPoint*
byExtendingSelection: (BOOL) *extend*

The datapoint index of the point *aPoint* is converted to the nearest datapoint in each GRDataSet drawn on the axes and set as the respective selected range. If multiple selection is enabled and extend is true, each dataset's current selected range will be extended to include the new selected point. Returns YES if *aPoint* was a valid point contained within the receiver's bounds. If the point *aPoint* is not within the receiver's bounds the selected range for each dataset will be set to [0,0) or no selection. Not Implemented in the abstract class.

clickPoint:

-(BOOL) **clickPoint:**(NSPoint) *aPoint*

Called when a point has been clicked. Not implemented in the abstract class.

xPixelValue

-(double) **xPixelValue**

Returns the numerical value associated with each pixel in the x direction.

yPixelValue

-(double) **yPixelValue**

Returns the numerical value associated with each pixel in the y direction.

xValueAtPoint:

-(double) **xValueAtPoint:**(NSPoint) *loc*

Returns the interpolated numerical value for the location *loc* on the x axis.

yValueAtPoint:

-(double) **yValueAtPoint:**(NSPoint) *loc*

Returns the interpolated numerical value for the location *loc* on the y axis.

locationForXValue:yValue:

-(NSPoint) **locationForXValue:**(double) *x* **yValue:**(double) *y*

Returns the location (NSPoint within chart) for the specified *x* and *y* values.

canvasRect

-(NSRect) **canvasRect**

Returns the size and position of the rectangle the axes (and it's corresponding datasets) will be drawn in.

setCanvasRect:

-(void) **setCanvasRect:**(NSRect) *aRect*

Resize the axes (and subsequently the datasets) to the new canvas rectangle size *aRect*.

plotRect

-(void) **plotRect**

Returns the size and position of the rectangle the axes and datasets will be drawn in.

setPlotRect

–(void) **setPlotRect:** (CGRect) *aRect*

Resizes the axes (and it's corresponding datasets) to the new plot rectangle size *aRect*.

legendRect

–(CGRect) **legendRect**

Returns the size and position of the rectangle the legend will be drawn in.

computeLayout

–(BOOL) **computeLayout**

Compute the layout of the graph if needed, taking the optional legend into account automatically. Always returns YES.

setNeedsLayout:

–(void) **setNeedsLayout:** (BOOL) *b*

Allows the graph's layout to be recomputed with **computeLayout** if set to YES.

needsLayout

–(BOOL) **needsLayout**

Returns YES if the graph's layout needs to be recomputed.

drawLegendRect:

–(void) **drawLegendRect:** (CGRect) *drawRect*

Draws the receiver's legend (if any) inside the current canvas rectangle.

drawGridRect:

–(void) **drawGridRect:** (CGRect) *drawRect*

Draws the receiver's gridlines (if any) inside the current canvas rectangle.

drawAxesRect:

–(void) **drawAxesRect:** (CGRect) *drawRect*

Draws the receiver's axes inside the current canvas rectangle.

Methods implemented by the Delegate

chart:categoryLabelForAxes:index:

–(NSString *) **chart:** (GRChartView *) *aChart*

categoryLabelForAxes: (GRAxes *) *anAxes* **index:** (int) *index*

Returns the label string to be used for the category at index *index*. This is used by the GRXYAxes subclass for labeling axes in category mode and by the GRPieAxes for legend labels.

GRXYAxes

Inherits from:

[GRAxes: NSObject](#)

Conforms to:

NSObject (NSObject)

Declared in:

GraphKit/GRXYAxes.h

Class Description

The GRXYAxes class is a subclass of GRAxes used to determine the scale and range of values to be plotted as well as drawing any necessary ticks, grid lines, titles, or legends. Only dataset subclasses of the GRXYDataSet class can be plotted on GRXYAxes.

Properties

Key	Description
GRAxesXAxisType	Selects the type of x-axis (category or value): GRAxesValueAxis GRAxesCategoryAxis
GRAxesXAxisScale	Selects the scale of the x-axis (linear or log10): GRAxesLinearScale GRAxesLog10Scale
GRAxesYAxisScale	Selects the scale of the y-axis (linear or log10): GRAxesLinearScale GRAxesLog10Scale
GRAxesXTitleFont	Font (NSFont) used to draw the GRAxesXTitle.
GRAxesYTitleFont	Font (NSFont) used to draw the GRAxesYTitle.
GRAxesXTitle	This string (NSString) will be drawn below the x-axis of the chart using the GRAxesXTitleFont.

<code>GRAxesYTitle</code>	This string (NSString) will be drawn below the y-axis of the chart using the <code>GRAxesYTitleFont</code> .
<code>GRAxesXMajorUnit</code>	Major unit used to draw the x-axis. Computed based on <code>GRAxesXPlotMin</code> , <code>GRAxesXPlotMax</code> , horizontal size of the chart area and labels or fixed if <code>GRAxesFixedXMajorUnit</code> is present.
<code>GRAxesXMinorUnit</code>	Minor unit used to draw the x-axis. Computed based on <code>GRAxesXPlotMin</code> , <code>GRAxesXPlotMax</code> , horizontal size of the chart area and labels or fixed if <code>GRAxesFixedXMinorUnit</code> is present.
<code>GRAxesYMajorUnit</code>	Major unit used to draw the y-axis. Computed based on <code>GRAxesYPlotMin</code> , <code>GRAxesYPlotMax</code> , vertical size of the chart area and labels or fixed if <code>GRAxesFixedYMajorUnit</code> is present.
<code>GRAxesYMinorUnit</code>	Minor unit used to draw the y-axis. Computed based on <code>GRAxesYPlotMin</code> , <code>GRAxesYPlotMax</code> , vertical size of the chart area and labels or fixed if <code>GRAxesFixedYMinorUnit</code> is present.
<code>GRAxesFixedXMajorUnit</code>	If true (NSNumber, bool value), the major x unit will not be recalculated based on <code>GRAxesXPlotMin</code> and <code>GRAxesXPlotMax</code> but will remain fixed.
<code>GRAxesFixedXMinorUnit</code>	If true (NSNumber, bool value), the minor x unit will not be recalculated based on <code>GRAxesXPlotMin</code> and <code>GRAxesXPlotMax</code> but will remain fixed.
<code>GRAxesFixedYMajorUnit</code>	If true (NSNumber, bool value), the major y unit will not be recalculated based on <code>GRAxesYPlotMin</code> and <code>GRAxesYPlotMax</code> but will remain fixed.
<code>GRAxesFixedYMinorUnit</code>	If true (NSNumber, bool value) the minor y unit will not be recalculated based on <code>GRAxesYPlotMin</code> and <code>GRAxesYPlotMax</code> but will remain fixed.
<code>GRAxesDrawXAxis</code>	If true (NSNumber, bool value) the x-axis will be drawn.
<code>GRAxesDrawXLabels</code>	If true (NSNumber, bool value) the x-axis major ticks will be labeled.
<code>GRAxesDrawXMajorTicks</code>	If true (NSNumber, bool value), the x-axis major ticks will be drawn.
<code>GRAxesDrawXMinorTicks</code>	If true (NSNumber, bool value), the x-axis minor ticks will be drawn.

<code>GRAxesDrawXMajorLines</code>	If true (NSNumber, bool value), the x-axis major grid lines will be drawn.
<code>GRAxesDrawXMinorLines</code>	If true (NSNumber, bool value), the x-axis minor grid lines will be drawn.
<code>GRAxesDrawYAxis</code>	If true (NSNumber, bool value), the y-axis will be drawn.
<code>GRAxesDrawYLabels</code>	If true (NSNumber, bool value), the y-axis major ticks will be labeled.
<code>GRAxesDrawYMajorTicks</code>	If true (NSNumber, bool value), the y-axis major ticks will be drawn.
<code>GRAxesDrawYMinorTicks</code>	If true (NSNumber, bool value), the y-axis minor ticks will be drawn.
<code>GRAxesDrawYMajorLines</code>	If true (NSNumber, bool value), the y-axis major grid lines will be drawn.
<code>GRAxesDrawYMinorLines</code>	If true (NSNumber, bool value), the y-axis minor grid lines will be drawn.
<code>GRAxesXLabelFormat</code>	The format string (NSString) used to display the x-axis labels which is either generated when <code>computeMajorMinorUnits()</code> is called or remains fixed if <code>GRAxesFixedXLabelFormat</code> is present.
<code>GRAxesYLabelFormat</code>	The format string (NSString) used to display the y-axis labels which is either generated when <code>computeMajorMinorUnits()</code> is called or remains fixed if <code>GRAxesFixedYLabelFormat</code> is present.
<code>GRAxesFixedXLabelFormat</code>	If true (NSNumber, bool value), the <code>GRAxesXLabelFormat</code> string will not be regenerated by <code>computeMajorMinorUnits()</code> .
<code>GRAxesFixedYLabelFormat</code>	If true (NSNumber, bool value), the <code>GRAxesYLabelFormat</code> string will not be regenerated by <code>computeMajorMinorUnits()</code> .
<code>GRAxesXLabelRotation</code>	The number of degrees (NSNumber) counter-clockwise to rotate x-axis labels.
<code>GRAxesYLabelRotation</code>	The number of degrees (NSNumber) counter-clockwise to rotate y-axis labels.
<code>GRAxesYMinTicks</code>	The minimum number of major ticks (NSNumber) allowed for the y-axis.
<code>GRAxesYMaxTicks</code>	The maximum number of major ticks (NSNumber) allowed for the y-axis.
<code>GRAxesXMinTicks</code>	The minimum number of major ticks (NSNumber) allowed for the x-axis.
<code>GRAxesXMaxTicks</code>	The maximum number of major ticks (NSNumber)

	allowed for the x-axis.
GRAxesXMinSpace	
GRAxesYMinSpace	

Method Types

Scaling

- [computeXMajorMinorUnits](#)

- [computeYMajorMinorUnits](#)

Drawing

- [drawXAxisRect:](#)

- [drawXGridRect:](#)

- [drawYAxisRect:](#)

- [drawYGridRect:](#)

Instance Methods

computeXMajorMinorUnits

-(void) **computeXMajorMinorUnits**

Compute the major and minor units of the x-axis according to the canvas rectangle (bounds).

computeYMajorMinorUnits

-(void) **computeYMajorMinorUnits**

Compute the major and minor units of the y-axis according to the canvas rectangle (bounds).

drawXAxisRect:

-(void) **drawXAxisRect:** (NSRect) *drawRect*

Draws the receiver's x-axis and inside the current canvas rectangle.

drawXGridRect:

-(void) **drawXGridRect:** (NSRect) *drawRect*

Draws the receiver's x-axis gridlines (if any) inside the current canvas rectangle.

drawYAxisRect:

-(void) **drawYAxisRect:** (NSRect) *drawRect*

Draws the receiver's y-axis and inside the current canvas rectangle.

drawYGridRect:

-(void) **drawYGridRect:** (NSRect) *drawRect*

Draws the receiver's y-axis gridlines (if any) inside the current canvas rectangle.

Methods implemented by the Delegate

chart:xLabelForAxes:value:defaultLabel:

-(NSString *) **chart:** (GRChartView *) *aChart*
xLabelForAxes: (GRXYAxes *) *aAxes* **value:** (double) *val*
defaultLabel: (NSString *) *aString*

Allows the delegate to modify or format x-axis labels. The label that would normally be used by default for the value *val* is passed in as *aString*.

chart:yLabelForAxes:value:defaultLabel:

-(NSString *) **chart:** (GRChartView *) *aChart*
yLabelForAxes: (GRXYAxes *) *aAxes* **value:** (double) *val*
defaultLabel: (NSString *) *aString*

Allows the delegate to modify or format y-axis labels. The label that would normally be used by default for the value *val* is passed in as *aString*.

GRPieAxes

Inherits from:

[GAXes](#); [NSObject](#)

Conforms to:

NSObject (NSObject)

Declared in:

GraphKit/GRPieAxes.h

Class Description

The GRPieAxes class is a subclass of GAXes used to determine the scale and range of values to be plotted as well as drawing any necessary ticks, grid lines, titles, or legends. Only dataset subclasses of the GRPieDataSet class can be plotted on GRPieAxes.

Chapter 5

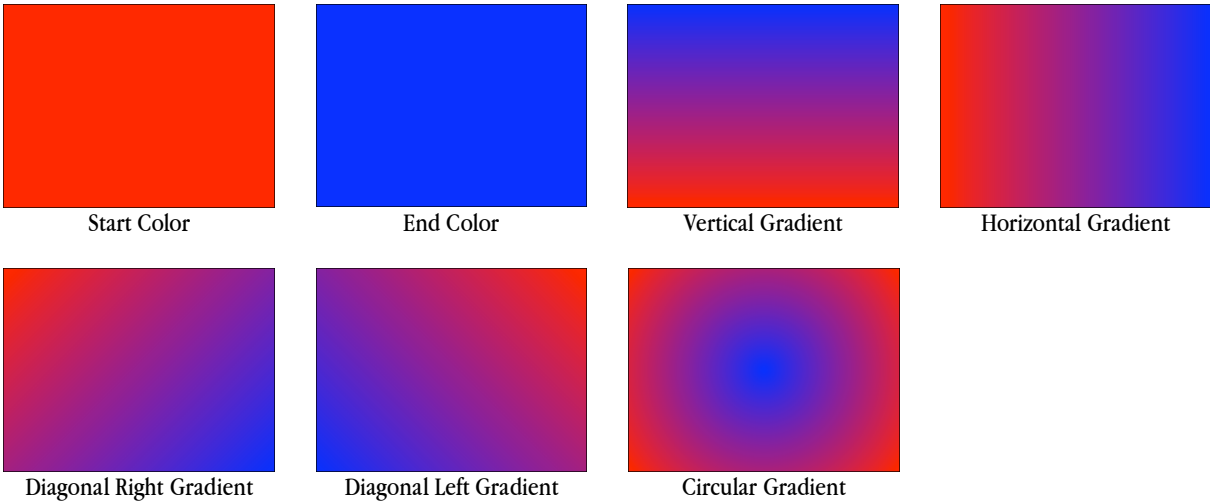
GRGradientColor

GRGradientColor

Inherits from:
[NSObject](#)
Conforms to:
NSObject (NSObject)
Declared in:
GraphKit/GRGradientColor.h

Class Description

An GRGradientColor object represents a range of color and sometimes opacity (alpha). By sending a `fillRect:` message to an GRGradientColor instance, you set the color gradient pattern for the current drawing context. This causes subsequently drawn graphics to have the specified color gradient pattern represented by the GRGradientColor instance.



Constants

Key	Description
<code>GRGradientFillVertical</code>	Fill the rectangle specified by <code>setInRect</code> , with a vertical color gradient drawn from the start color at the bottom to the end color at the

	top.
GRGradientFillHorizontal	Fill the rectangle specified by setInRect, with a horizontal color gradient drawn from the start color at the left to the end color at the right.
GRGradientFillDiagonalRight	Fill the rectangle specified by setInRect, with a diagonal color gradient drawn from the start color at the top-left to the end color at the bottom-right.
GRGradientFillDiagonalLeft	Fill the rectangle specified by setInRect, with a diagonal color gradient drawn from the start color at the top-right to the end color at the bottom-right.
GRGradientFillRadial	Fill the rectangle specified by setInRect, with a circular color gradient drawn from the start color at the perimeter to the end color at the center. (Not Implemented)

Method Types

Creating a gradient

[+ gradientOfType:withColors:...](#)

[- init](#)

[- initWithColors:...](#)

[- initWithColorArray:](#)

Modifying the colors

[- addColor:](#)

[- colorCount](#)

[- colors](#)

Modifying the gradient

[- setGradientType:](#)

[- gradientType](#)

Drawing

[- fillRect:](#)

[- fillBezierPath:](#)

[- fillBezierPath:withBounds:](#)

Compatibility with NSColor

[- set](#)

[- drawSwatchInRect:](#)

Class Methods

gradientOfType:withColors:...

```
+(id) gradientOfType:(gradient_t) aType withColors:(NSColor
*) firstColor, ...
```

Creates and returns a GRGradientColor representing the gradient of colors between the start color *firstColor* and the end color, if present. The gradient type is specified by *aType*.

Instance Methods

init

`-(id) init`

Initializes a GRGradientColor representing a default gradient.

initWithColors:...

`-(id) initWithColors:(NSColor *)aColor, ...`

Initializes a GRGradientColor representing the gradient of colors between the start color *aColor* and the end color, if present.

initWithColorArray:

`-(id) initWithColorArray:(NSArray *)colors`

Initializes a GRGradientColor representing the gradient of colors using the colors in *color* in their given order.

addColor:

`-(void) addColor:(NSColor *)aColor`

Adds *aColor* to the receiver's color array.

colorCount

`-(int) colorCount`

Returns the number of colors in the receivers color array.

colors

`-(NSArray *) colors`

Returns the receiver's color array.

setGradientType:

`-(void) setGradientType:(gradient_t) aType`

Sets the receiver's gradient fill type to be *aType*.

gradientType:

`-(gradient_t) gradientType`

Returns the receiver's gradient fill type.

fillRect:

```
-(void) fillRect:(CGRect) aRect
```

Creates a Bezier path from the given rectangle, and uses the Bezier path to fill the receiving gradient.

fillBezierPath:

```
-(void) fillBezierPath:(NSBezierPath *) path
```

Fills the receiving gradient with the given Bezier Path.

fillBezierPath:withBounds:

```
-(void) fillBezierPath:(NSBezierPath *) path  
withBounds:(CGRect) bounds
```

Fills the receiving gradient with the given Bezier path and clipping bounds.

set:

```
-(void) set
```

This function is provided for compatibility with NSColor objects. It calls **set** for the last color in the array, or for black if the array is empty.

drawSwatchInRect:

```
-(void) drawSwatchInRect:(CGRect) aRect
```

This function is provided for compatibility with NSColor objects. This functions the same as **fillRect**.
