MiscGaugeView

Inherits From: Control

Declared In: <misckit/MiscGaugeView.h>

Class Description

This class is basically a cover for a MiscGaugeCell which does pretty much all the work when we give it some View real estate to draw in. See the MiscGaugeCell class for more information on the inner workings of the gauge. Even the documentation below was basically cut and pasted from the MiscGaugeCell docs.

Instance Variables

None declared.

Method Types

Setting the cell to use	+ setCellClass:
Initializing a MiscGaugeView	+ initialize - initFrame: - initFrame:min:max:startAngle:range:tickInterval:
Setting the Gauge's values	- maxValue - setMaxValue: - minValue - setMinValue: - angleRange - setAngleRange: - handRatio - setHandRatio: - startAngle - setStartAngle: - tickInterval - setTickInterval: - tickRatio - setTickRatio:
Setting colors	- gaugeColor - gaugeGray - setGaugeColor: - setGaugeGray: - textColor

	- textGray - setTextColor: - setTextGray:
Manipulating the title	- title - setTitle: - titleFont - setTitleFont: - titlePosition - setTitlePosition:

Class Methods

initialize

+ **initialize**

Sets the default cell to MiscGaugeCell.

See also:

setCellClass:

+ **setCellClass:** *aClass*

Sets our factory cell class to *aClass*. This should probably be a subclass of MiscGaugeCell unless you enjoy run-time errors.

Instance Methods

angleRange

- (float)**angleRange**

Returns the sweep of the hand from minimum to maximum value, starting at *startAngle.* This will not be less than 0 or more than 360 degrees.

See also: ± setAngleRange:, ± startAngle, ± setStartAngle:

gaugeColor

- (NXColor)**gaugeColor**

Returns the current color of the gauge face.

See also: ± setGaugeColor, ± gaugeGray

gaugeGray

- (float)**gaugeGray**

Returns the current gray of the gauge face.

See also: ± setGaugeGray:, ± gaugeColor

handRatio

- (float)**handRatio**

Returns the ratio of the hand length to the radius of the face. For reference, a value of 0.5 would mean the

length of the hand was half of the radius.

See also: **± setHandRatio:**

initFrame:

- **initFrame:**(const NXRect *)*frameRect*

Just calls our designated initializer (right below) with some reasonable values.

See also: **± initFrame:min:max:startAngle:range:tickInterval:**

initFrame:min:max:startAngle:range:tickInterval:

- **initFrame:**(const NXRect *)*frameRect*
  min:(float)*min*
  max:(float)*max*
  startAngle:(float)*start*
  range:(float)*range*
  tickInterval:(int)*interval*

This our designated initializer. Sets all the ivars from parameters above. The default title font is also set to Helvetica 10pt. Returns self.

See also: **± initFrame:**

maxValue

- (float)**maxValue**

Returns the maximum value of the gauge.

See also: ± setMaxValue:, ± minValue

minValue

- (float)**minValue**

Returns the minimum value of the gauge.

See also: ± setMinValue:, ± maxValue

setAngleRange:

- **setAngleRange:**(float)*newValue*

Sets the span, in degrees, from the gauge's minimum to the maximum. The span starts at *startAngle* and continues clockwise. if *newValue* is less than 0 or greater then 360, it will be adjusted (set to either 0 or 360, respectively) so it is within a meaningful range.

See also: ± angleRange, ± startAngle

setGaugeColor:

- **setGaugeColor:**(NXColor)*color*

Sets the color of the gauge face. I've found that pale colors are not a good choice.

See also: ± gaugeColor, ±gaugeGray, ± setGaugeGray:

setGaugeGray:

- **setGaugeGray:**(float)*gray*

Sets the gray that the gauge face is drawn in.

See also: ± gaugeGray, ± gaugeColor, ± setGaugeColor:

setHandRatio:

- **setHandRatio:**(float)*newRatio*

Sets the hand length in ratio to the gauge's radius. A value of 0.5 would draw a hand with a length half of the gauge's radius. The value must be between 0.2 and 0.9. If it is either larger or smaller, the new ratio will be set to either 0.2 or 0.9 respectively.

See also: ± handRatio

setMaxValue:

- **setMaxValue:**(float)*max*

Sets a new maximum value for the gauge. If the new maximum value happens to be lower than the current minimum, the minimum value is adjusted to be one less than *max*. If the value of the gauge (as returned by **floatValue**) is now larger than the gauge's new maximum it is also changed to equal *max*.

See also: ± maxValue, ± minValue, ± setMinValue:

setMinValue:

- **setMinValue:**(float)*min*

Sets a new minimum value for the gauge. If the new minimum value happens to be higher than the current

maximum, the maximum value is adjusted to be one greater than *min*. If the value of the gauge (as returned by **floatValue**) is now smaller than the gauge's new minimum it is also changed to equal *min*.

See also: ± minValue, ± maxValue, ± setMaxValue:

setStartAngle:

- **setStartAngle:**(float)*newValue*

Sets the angle, in degrees, where the minimum value of the gauge will appear. Zero degrees is due east, with increasing numbers moving the start counter-clockwise. The *newValue* should be inbetween 0 and 360. If not, it will be adjusted so it is.

See also: ± startAngle, ± angleRange, ± setAngleRange:

setTextColor:

- **setTextColor:**(NXColor)*color*

Sets the color of the text and gauge hand.

See also: ± textColor, ± textGray, ± setTextGray:

setTextGray:

- **setTextGray:**(float)*gray*

Sets the gray level of the text and gauge hand.

See also: ± textGray, ± textColor, ± setTextColor:

setTickInterval:

- **setTickInterval:**(int)*newValue*

Sets the interval that the gauge face's tick marks should be drawn (including the numbers). For example, if you had a minimum value of 10.0 and a maximum of 100.0, then a tick interval of 10 would probably be around right. A current limitation is that the tick interval cannot be less than 1. This will be fixed in a coming release.

See also: ± tickInterval

setTickRatio:

- **setTickRatio:**(float)*newRatio*

Sets the tick mark's radius in ratio to the gauge face's radius. Usually this is adjusted if either the numbers or the gauge's title overlap the tick marks.

See also: ± tickRatio

setTitle:

- **setTitle:**(const char *)*newTitle*

Sets the gauge's title. A value of NULL will remove the existing title.

See also: ± title, ± titleFont, ± setTitleFont:

setTitleFont:

- **setTitleFont:** *newFont*

Sets the font for the gauge's title. No matter if this view is flipped or not, *newFont* should have an NX_IDENTITYMATRIX matrix.

See also: ± titleFont, ± title, ± setTitle:

setTitlePosition:

- **setTitlePosition:**(int)*newPos*

Sets the position of the title. Currently the only valid positions are NX_ATTOP and NX_ATBOTTOM.

See also: ± titlePosition

startAngle

- (float)**startAngle**

Returns the gauge's start angle (where the minimum value is located). See **setStartAngle:** for more information.

See also: ± setStartAngle:, ± angleRange, ± setAngleRange:

textColor

- (NXColor)**textColor**

Returns the color that the text and gauge hand are drawn in.

See also: ± setTextColor:, ± textGray, ± setTextGray:

textGray

- (float)**textGray**

Returns the gray that the text and gauge hand are drawn in.

See also: ± setTextGray:, ± textColor, ± setTextColor:

tickInterval

- (int)**tickInterval**

Returns the current tick interval.

See also: ± setTickInterval:

tickRatio

- (float)**tickRatio**

Returns the radius of the tick marks in relation to the gauge's radius. This value will be between 0.0 (no tick marks) and 1.0.

See also: ± setTickRatio:

title

- (const char *)**title**

Returns the gauge's current title.

See also: ± setTitle:, ± titlePosition, ± setTitlePosition:

titleFont

- **titleFont**

Returns the current font used for displaying the title. By default it is the same as Cell's font.

See also: ± setTitleFont:, ± title, ± setTitle:

titlePosition

- (int)**titlePosition**

Returns the current position of the title even if there is no current title. Currently, either NX_ATTOP or NX_ATBOTTOM will be returned.

See also: ± setTitlePosition: