MiscGaugeCell

Inherits From: Cell

Declared In: <misckit/MiscGaugeCell.h>

Class Description

This class is a simple round analog Gauge that came about while extending the GaugeView class that comes with the BusyBox example (/NextDeveloper/Examples/AppKit/BusyBox). It should also be fairly easy to modify the look of the gauge by overriding **drawFace:** and/or **drawHand:flipped:**. Let me know if you try and run into any problems that I hadn't considered.

Instead of me explaining all the features, and what most of the methods do (unless you plan to subclass), load up the palette in Interface Builder and play with the options on the Inspector. If you have any comments and/or suggestions, make improvements, or <gasp> find a bug, please let me know.

Instance Variables

NXImage ***cache**;
Font ***titleFont**;
short **titlePosition**;
float **radius**;
NXPoint **center**;
float **startAngle**;
float **angleRange**;
float **degreesPerUnit**;
int **tickInterval**;
float **tickRatio**;
float **handRatio**;
NXColor **gaugeColor**;
NXColor **textColor**;
float **minValue**;
float **maxValue**;
float **value**;
char ***strValue**;
BOOL **needRedraw**;
NXRect **lastRect**;

cache	Cached NXImage of the gauge face.
titleFont	Font used to display the title (The numbers use the Cell's font).
titlePosition	Position of the title on the gauge.
radius	The radius of the gauge face.

center	The center of the gauge face.
startAngle	The beginning angle of the numbers on the gauge.
angleRange	The span of the gauge numbers, starting at startAngle.
degreesPerUnit	The number of degrees per unit mark.
tickInterval	The interval of the tick marks.
tickRatio	Tick radius in ratio to the gauge's radius.
handRatio	Hand length in ratio to the gauge's radius.
gaugeColor	Color of the gauge face.
textColor	Color of all the text and gauge hand.
minValue	Minimum value of the gauge.
maxValue	Maximum value of the gauge.
value	Current value of the gauge.

Method Types

Initializing a MiscGaugeCell	+ initialize - init - free
Overridden from Cell	- calcCellSize:inRect: - highlight:inView:lit:

Setting the Gauge's values	- doubleValue - setDoubleValue: - floatValue - setFloatValue: - intValue - setIntValue: - maxValue - setMaxValue: - minValue - setMinValue: - stringValue - setStringValue:
Manipulating the gauge face	- 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:
Display	- drawFace: ± drawHand:flipped: - drawInside:inView: - drawSelf:inView:
Archiving	- read: - awake - write:

Class Methods

initialize

+ **initialize**

Sets the class verison for archiving purposes.

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:

awake

- **awake**

Allocates any ivars that were not archived, which includes the NXImage for the gauge face.

See also: ± read:, ± write:

calcCellSize:inRect:

- **calcCellSize:**(NXSize *)*size*
  inRect:(NXRect *)*rect*

Overridden from Cell to calculate the minimum size that the cell will occupy.

doubleValue

- (double)**doubleValue**

Returns the current value of the gauge.

See also: ± **setDoubleValue:**, ± **floatValue**, ± **intValue**, ± **stringValue**

drawFace:

- **drawFace:**(const NXRect *)*rect*

Called by drawSelf:inView: to draw the gauge face in the image cache. This should be called only when one of the attributes that make up the gauge face changes. The passed *rect* is the same rect that was passed to drawSelf:inView:. If you wanted to alter the face of the gauge, this would be the method to override. Just composite whatever image into the cache and it will be composited when it is needed.

See also: ± **drawSelf:inView:**

drawHand:flipped:

- **drawHand:**(const NXRect *)*rect* **flipped:** (BOOL)*viewFlipped*

Called by drawInside:inView: to draw the gauge hand. If you want to change the look of the hand, this would be the method to override. Since the view that we are drawing into is already lockfocused, just draw to your heart's content. Make sure to compensate for flipped views so that your gauge can be used in a matrix too.

See also: ± **drawInside:inView:**

drawInside:inView:

- **drawInside:**(const NXRect *)*cellFrame*
  inView:*aView*

Does a minimal update, which composites the gauge face, then draws the gauge hand. If you are customizing the look of the gauge, you'll probably want to override **drawFace:** and/or **drawHand:flipped:** instead.

See also: **± drawSelf:inView:,** **± drawFace:,** **± drawHand:flipped:**

drawSelf:inView:

- **drawSelf:**(const NXRect *)*rect*
  inView: *controlViewtion:*

Calls **drawFace:** to redraw the cached gauge face, then calls **drawInside:inView:** to composite the new face and draw the hand

See also: **± drawFace:,** **± drawInside:inView:**

floatValue

- (float)**floatValue**

Returns the current value of the gauge.

See also: **± setFloatValue:,** **± doubleValue,** **± intValue,** **± stringValue**

free

- **free**

Frees the internal buffer and image cache.

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:

highlight:inView:lit:

- **highlight:**(const NXRect *)*cellFrame*
  inView:*aView*
  lit:(BOOL)*flag*

We don't want our cell to highlight, so it is overridden from Cell. This implementation does nothing and returns self.

init

- **init**

MiscGaugeCell's destinated initializer. Don't use either of Cell's other initializers (initTextCell: or initIconCell:).

intValue

- (int)**intValue**

Returns an integer representing the current value of the gauge.

See also: ± setIntValue:, ± doubleValue, ± floatValue, ± stringValue

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

read:

- **read:**(NXTypedStream *)*stream*

Reads an instance of MiscGaugeCell from *stream*.

See also: ± write:, ± awake

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

setDoubleValue:

- **setDoubleValue:**(double)*val*

Sets the value of the gauge to be *val*.

See also: ± doubleValue, ± floatValue, ± intValue, ± stringValue

setFloatValue:

- **setFloatValue:**(float)*val*

Sets the value of the gauge to be *val*. If *val* is less than the minimum or greater than the maximum value, it will be set to either the minimum or maximum value, respectively.

See also: ± floatValue, ± doubleValue, ± intValue, ± stringValue

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

setIntValue:

- **setIntValue:**(int)*val*

Sets the value of the gauge. Internally *val* will be converted to a float. If the value is not between the minimum and maximum, it will be adjusted so that it is.

See also: ± intValue, ± doubleValue, ± floatValue, ± stringValue

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:

setStringValue:

- **setStringValue:**(const char *)*val*

Trys to convert *val* into a floating point number using atof(). If no value can be found, the value of the gauge will be set to 0. If the converted value is not in between the minimum and maximum value, it will be set so that it is.

See also: ± stringValue, ± doubleValue, ± floatValue, ± intValue

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:

stringValue

- (const char *)**stringValue**

Returns a string representation of the gauge's current value. Do not free the value returned.

See also: ± setStringValue:, ± doubleValue, ± intValue, ± floatValue

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:

write:

- **write:**(NXTypedStream *)*stream*

Writes an instance of MiscGaugeCell to *stream*.

See also: ± read:, ± awake