MiscGaugeCell

Inherits From: NSCell
Declared In: MiscAppKit/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 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

NSImage* **cache**;
NSCell* **titleCell**;
NSTitlePosition **titlePosition**;
float **radius**;
NSPoint **center**;
float **startAngle**;

float **angleRange**;
float **degreesPerUnit**;
int **tickInterval**;
float **tickRatio**;
float **handRatio**;
NSColor* **gaugeColor**;
NSColor* **textColor**;
double **minValue**;
double **maxValue**;

cache	Cached NXImage of the gauge face.
titleCell	Used to display the title.
titlePosition	Either NSAtTop or NSAtBottom.
radius	Radius of gauge face.
center	Center of gauge face.
startAngle	Beginning angle of the gauge numbers
angleRange	Span of the numbers, from 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 the text and gauge hand.

minValue	Minimum value of the gauge.
maxValue	Maximum value of the gauge.

Method Types

Initialization/deallocation	+ initialize - init - dealloc
Values	- maxValue - setMaxValue: - minValue - setMinValue: - setFloatValue: - setDoubleValue: - setIntValue: - setStringValue:
Angles and ratios	- startAngle - setStartAngle: - angleRange - setAngleRange: - tickInterval - setTickInterval: - tickRatio - setTickRatio: - handRatio - setHandRatio:
Colors	- gaugeColor

	- setGaugeColor: - textColor - setTextColor:
Titles	- titleFont - setTitleFont: - title - setTitle: - titlePosition - setTitlePosition:
Overridden	- cellSizeForBounds:
Drawing	- drawWithFrame:inView: - drawInteriorWithFrame:inView: - drawHand:flipped: - drawFace: - highlight:withFrame:inView:
Archiving	- initWithCoder: - encodeWithCoder:

Class Methods

initialize

+ (void)**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.

cellSizeForBounds:

- (NSSize)**cellSizeForBounds:**(NSRect)*rect*

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

dealloc

- (void)**dealloc**

Deallocates our cache, fonts and colors.

drawFace:

- (void)**drawFace:**(NSRect)*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.

drawHand:flipped:

- (void)**drawHand:**(NSRect)*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.

drawInteriorWithFrame:inView:

- (void)**drawInteriorWithFrame:**(NSRect)*cellFrame*
  inView:(NSView*)*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.

drawWithFrame:inView:

- (void)**drawWithFrame:**(NSRect)*rect*
  inView:(NSView*)*controlView*

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

encodeWithCoder:

- (void)**encodeWithCoder:**(NSCoder*)*aCoder*

Encodes an instance of MiscGaugeCell.

gaugeColor

- (NSColor*)**gaugeColor**

Returns the current color of the gauge face.

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.

highlight:withFrame:inView:

- (void)**highlight:**(BOOL)*flag*
  withFrame:(NSRect)*cellFrame*
  inView:*aView*

Overridden to do nothing since we never want highlighting to occur.

init

- init

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

initWithCoder:

- (id)**initWithCoder:**(NSCoder*)*aDecoder*

Returns a newly unarchived instance of MiscGaugeCell. It doesn't seem to be able to read pre-OpenStep created gauge cells because I haven't figured out how to unarchive an NXColor structure yet.

maxValue

- (double)**maxValue**

Returns the maximum value of the gauge.

minValue

- (double)**minValue**

Returns the minimum value of the gauge.

setAngleRange:

- (void)**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.

setDoubleValue:

- (void)**setDoubleValue:**(double)*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.

setFloatValue:

- (void)**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.

setGaugeColor:

- (void)**setGaugeColor:**(NSColor*)*color*

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

setHandRatio:

- (void)**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.

setIntValue:

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

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

setMaxValue:

- (void)**setMaxValue:**(double)*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*.

setMinValue:

- (void)**setMinValue:**(double)*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*.

setStartAngle:

- (void)**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 in between 0 and 360. If not, it will be adjusted so it is.

setStringValue:

- (void)**setStringValue:**(NSString*)*val*

Trys to convert *val* into a floating point number. 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.

setTextColor:

- (void)**setTextColor:**(NSColor*)*color*

Sets the *color* of the text and gauge hand.

setTickInterval:

- (void)**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.

setTickRatio:

- (void)**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.

setTitle:

- (void)**setTitle:**(NSString*)*newTitle*

Sets our title.

setTitleFont:

- (void)**setTitleFont:**(NSFont*)*newFont*

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

setTitlePosition:

- (void)**setTitlePosition:**(NSTitlePosition)*newPos*

Sets the position of the title. Currently the only valid positions are NSAtTop and NSAtBottom.

startAngle

- (float)**startAngle**

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

textColor

- (NSColor*)**textColor**

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

tickInterval

- (int)**tickInterval**

Returns the current tick interval.

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.

title

- (NSString*)**title**

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

titleFont

- (NSFont*)**titleFont**

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

titlePosition

- (NSTitlePosition)**titlePosition**

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