# MiscGaugeView

**Inherits From:** NSControl
**Declared In:** MiscAppKit/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.

## Method Types

| | |
|---|---|
| Class initialization | + initialize<br>+ cellClass<br>+ setCellClass: |
| Initialization | - initWithFrame:<br>- initWithFrame:min:max:startAngle:range:tickInterval: |
| Our state | - maxValue<br>- setMaxValue:<br>- minValue<br>- setMinValue:<br>- startAngle<br>- setStartAngle:<br>- angleRange<br>- setAngleRange:<br>- tickInterval |

| | |
|---|---|
| | - setTickInterval:<br>- tickRatio<br>- setTickRatio:<br>- handRatio<br>- setHandRatio: |
| Our colors | - gaugeColor<br>- setGaugeColor:<br>- textColor<br>- setTextColor: |
| Title attributes | - titleFont<br>- setTitleFont:<br>- title<br>- setTitle:<br>- titlePosition<br>- setTitlePosition: |

## Class Methods

### cellClass

+ (Class)**cellClass**

Returns the class of the cell we will use. It should be a subclass of MiscGaugeCell or the class itself (the default).

### initialize

+ (void)**initialize**

Sets our cell class to be MiscGaugeCell.

### setCellClass:

+ (void)**setCellClass:**(id)*aClass*

Sets *aClass* to be the cell we'll use as our gauge cell. It should be a subclass of MiscGaugeCell.

## 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.

### 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 of the gauge face.

### initWithFrame:

- (id)**initWithFrame:**(NSRect)*frameRect*

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

### initWithFrame:min:max:startAngle:range:tickInterval:

- (id)**initWithFrame:**(NSRect)*frameRect*
  **min:**(double)*min*
  **max:**(double)*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**.

**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.

**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.

**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 between 0 and 360. If not, it will be adjusted so 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 the gauge's title. A value of **nil** will remove the existing 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 NX_IDENTITYMATRIX 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 the **setStartAngle:** method for more information.

**textColor**

- (NSColor*)**textColor**

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

**tickInterval**

- (int)**tickInterval**

Returns the current spacing of the tick marks on the gauge.

**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 (on the very edge of the gauge face. .

**title**

- (NSString*)**title**

Returns the gauge's current title.

**titleFont**

- (NSFont*)**titleFont**

Returns the font that we'll use to draw our title.

**titlePosition**

- (NSTitlePosition)**titlePosition**

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