Ruler

INHERITS FROM View

CLASS DESCRIPTION

The Ruler class is a general ruler designed to be used with the RuledScrollView class. Various instance variables (and the methods to set them and retrieve their values) are predefined in this class. These variables determine how the the ruler is drawn. By subclassing and overriding the **drawSelf::** method, custom ruler styles can easily be designed.

It is recommended that subclasses of ruler use the instance variables defined in the Ruler class wherever possible to represent the various parameters used by a particular ruler class's **drawSelf::** method. For example, if you design a ruler that draws a checkerboard pattern, use the *scaleSpacing* instanceVariable to define the size of the checkerboard squares rather than defining a new '*checkerSize*' variable. This will make it easy to change to another ruler style that you or someone else designs in the future. The standard

ruler attributes are described below.

scaleSpacing sets the distance between major marks, measured in points (1/72 of an inch). For example, an ruler displaying inches should set *scaleSpacing* to 72.0. (Note that the actual spacing on-screen will be dependent on the screen's dpi.). The default *scaleSpacing* is 72*0.39, which gives centimeters.

scale sets the measure (numbering) increment. The default is 1.0.

scaleOffset the value of the measure at the origin of the ruler. The default is 0.

border the distance between the origin of the ruler and the first mark or measure. This border is in addition to a margin that is automatically added for the width or height of an adjacent ruler. For example, a margin the width of the left ruler is automatically added to the beginning of the top ruler (in the absence of a 'stub' view [which aren't implemented]).

orientation sets the angle of the ruler, measured counter-clockwise from the positive X-axis. For horozontal or vertical rulers, the manifests "HORIZONTAL" or "VERTICAL" are defined to be 0.0 and 90.0, respectively. The default is HORIZONTAL.

inverted is a flag which is basically used to determine which edge of the main view the ruler is along, so that the ruler's graphics can be oriented to 'point' to the view that it is 'rulering'. For example, if we designed a ruler consisting of tick-marks with measures (numbers) over them, we would like the ticks to appear along the top edge of the ruler if the ruler was located at the bottom of a view, and along the bottom edge of the ruler if the ruler was located at the top of a view. Implement a Ruler's **drawSelf::** method such that it draws the ruler as it should appear on the top or right side of the main view when *inverted* == YES. When *inverted* == NO, draw the ruler as it should appear on the bottom or left side of the main view. The default

setting for *inverted* is NO.

mirrored a flag that is used to indicate the direction of numbering the ruler. By default (*mirrored* == NO) the 'origin' of a ruler is at the left end of a horizontal ruler and the bottom of a vertical ruler, and measures (numbers) increase in value towards the right or top or the ruler. Note that this flag is not used by the default **drawSelf::** method (due to programmer laziness). The default is NO.

alignment sets the alignment of text. The value should be one of the manifests NX_LEFTALIGNED, NX_CENTERED, or NX_RIGHTALIGNED (defined in appkit/Text.h). The default is NX_CENTERED.

font a font object which describes the font used to draw the text. The default is Helvetica size 12.

backgroundColor The color used to fill the ruler rectangle. The default is white.

backgroundGray The gray used to fill the ruler rectangle. The default is NX_WHITE.

image an image object for rulers that tile an image to the background. The default **drawSelf::** method does not do this, and therefore does not use this variable.

The default Ruler **drawSelf::** method displays measures (numbers), an optional string printed after each number (units), and a single black line adjacent to the mainView. The background is filled with the gray level specified by *backgroundGray* if *backgroundColor* is white (red, green, and blue components are all 1.0), otherwise the color given by *backgroundColor* is used. The default is NX_WHITE.

USAGE

Instructions explaining how Rulers are used with the RuledScrollView class are given in the

RuledScrollView class documentation.

LIMITATIONS/BUGS

Let me know if you discover any!

This is Version 1.0 of Ruler, released August, 1992.

Author:
Kevin Brain (ksbrain@zeus.uwaterloo.ca)
University of Waterloo / Department of Systems Design / Waterloo, Ontario/N2L 3G1

Based on classes and examples by Jayson Adams, NeXT Developer Support Team

INSTANCE VARIABLES

Inherited from Object	Class	isa;
Inherited from Responder	id	nextResponder;

Inherited from View	NXRect	frame;
	NXRect	bounds;
	id superview;	
	id subviews;	
	id window;	
	struct __vFlags	vFlags;
	Declared in Ruler	NXCoord scaleSpacing;
	float scale;	
	float scaleOffset;	
	int orientation;	
	BOOL	inverted;
	BOOL	mirrored;
	int alignment;	
	(Font *)	font;
	NXColor	backgroundColor;
	float	backgroundGray;
	id image;	
	int unitsLength;	
	char *units;	
	BOOL	draggableRuler;
	NXCoord	ascender;

```
NXCoord         descender;
NXCoord         lineHeight;
NXCoord         leftOrBottomMargin;
NXCoord         rightOrTopMargin;
```

scaleSpacing	the distance between major marks, measured in points
scale	the measure (numbering) increment
scaleOffset	value of the measure at the origin of the ruler
orientation	angle of the ruler, measured counter-clockwise from the positive X-axis
inverted	sets the 'measuring' edge of the ruler
mirrored	indicates the direction of numbering for the ruler
alignment	sets the alignment of text
font	Font used for text
backgroundColor	The color used to fill the ruler rectangle
backgroundGray	The color used to fill the ruler rectangle;
image	an image object for rulers that tile an image to the background
unitsLength	length of units string
units	string printed after numbers (eg. "yrs")
draggableRuler	whether ruler is draggable
ascender	font info retrieved when font is set
descender	font info retrieved when font is set
lineHeight	font info retrieved when font is set

leftOrBottomMargin	Margin at edge of view as a result of other ruler
rightOrTopMargin;	Margin at edge of view as a result of other ruler

METHOD TYPES

Initializing and freeing an instance	- initFrame:(NXRect *)frameRect; - free; // - awake;
IB Custom Palette Support	//- (const char*)inspectorName; //- read:(NXTypedStream *) s; //- write:(NXTypedStream *) s;
Drawing the view	- drawSelf:(NXRect *)rects :(int)rectCount;
Event handling	- (BOOL)acceptsFirstMouse; - mouseDown:(NXEvent *)theEvent;
Sizing the rulers	- setSize;
Setting/returning Ruler attributes	- setScaleSpacing:(NXCoord)*points* - (NXCoord)scaleSpacing; - setScale:(float)*scaleIncrement*;

- (float)scale;
- setScaleOffset:(float)*origin*;
- (float)scaleOffset;
- setBorder:(NXCoord)*theBorder*;
- (NXCoord)border;
- setOrientation:(float)*angle*;
- (float)orientation;
- setInverted:(BOOL)*invertFlag*;
- (BOOL)inverted;
- setMirrored:(BOOL)*mirrorFlag*;
- (BOOL)mirrored;
- setAlignment:(int)*alignType*;
- (int)alignment;
- setUnits:(char *)*name*;
- (const char *)units;
- setFont:*aFont*;
- (Font *)font;
- setBackgroundColor:(NXColor)*bColor*;
- (NXColor)backgroundColor;
- setBackgroundGray:(float)*value*;
- (float)backgroundGray;
- setImage:*anImage*;
- image;

- setDraggableRuler:(BOOL)*yesOrNo*;
- (BOOL)draggableRuler;

INSTANCE METHODS

[NOTE: Methods that simply set or return the value of a ruler attribute instance variable are not included in this section (since these methods are trivial). Explanations of what the variables themselves refer to is included above.]

acceptsFirstMouse

- (BOOL)**acceptsFirstMouse**

Returns YES.

drawSelf::

- **drawSelf:**(NXRect *)**rects :**(int)**count**

Draws the Ruler according to the values of its attributes. Use the **display** method rather than sending this method directly.

free

- **free**

Frees all disposable storage used by the Ruler. Returns **nil**.

initFrame:

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

Initializes and returns the receiver, a new Ruler instance. See the variable descriptions above for default values.

mouseDown:

- **mouseDown:**(NXEvent *)theEvent

This method is invoked when the user presses the left mouse button within the bounds of a Ruler. If draggableRuler == YES, a window the size of the visible portion of the ruler is created, the ruler is drawn into the window, and a drag loop that allows the ruler to be dragged around the screen is implemented.

setSize

- **setSize**

Sets the size of the view. This method is invoked by the setSize method of RuledScrollView, which must be invoked whenever the size of the mainView is changed. The size of the Ruler is lengthened to the size of the mainView. In addition, this method invokes the **primaryRulers** method of the RuledScrollView (by sending to [superview superview], the Ruler's ClipView's superview) to find out which Rulers, top and bottom or left and right, extend to the edges of the RuledScrollView. If the top and bottom Rulers extend to the edges (they are 'primary'), then the width of the left vertical Ruler (if present) is obtained so that the origin of the horizontal rulers can be adjusted accordingly. (The width is obtained, essentially, by sending

[[[superview superview] leftRuler] getFrame:&rulerFrame].) Corresponding sizing and adjusting of margins is done for VERTICAL rulers. Returns self.
NOTE: Expect this method to disappear in the next version. (I plan to do all the resizing in the RuledScrollView's setSize method.