MiscSoundView

Inherits From: SoundView

Declared In: MiscSoundView.h

Class Description

The MiscSoundView adds a large number of features to NeXT's SoundView object, and cleans up some bugs left behind by NeXT in 2.x and 3.x versions of the SoundView.

Important Note: The MiscSoundView is designed to be used **solely within a ScrollView object**. Using it by itself, or within some other object (like an NXSplitView) could have unforseen consequences. Be certain to set the background gray of the ScrollView to NX_WHITE, and set the MiscSoundView inside to be vertically stretchable.

The Ruler. Perhaps the most useful element of the MiscSoundView is its ruler. Above its sound data, the MiscSoundView can show a ruler with major and minor tick marks, in units of seconds, samples, or a percentage of the entire sound (from 0 to 100). The major tick marks can be labeled (if there's not enough room, the labels will automatically disappear). The MiscSoundView automatically resizes itself to fit the ruler (or remove it) within the MiscScrollView boundaries. Note that the ruler will not appear until sound data actually exists. **Note:** To work its ruler magic, the MiscSoundView does a **Very Bad Thing™**. It draws the ruler *outside* of the bounds of the SoundView. This is because NeXT's SoundView provides no mechanism for drawing inside the SoundView bounds without drawing on top of SoundView sound data. This means that the MiscSoundView must override the **setSizeTo:** method to resize itself properly within a ScrollView. But Interface Builder doesn't always use **setSizeTo:** to modify a view's size. The result: the ruler is sometimes not drawn. If you get pinched by this, just remember to call **adjustBounds:** after putting the MiscSoundView into its parent view, and when you're ready to display the MiscSoundView (with or without sound data).

Amplitude Marks and The Zero Line. The MiscSoundView can overlay its sound with several different types of amplitude grid lines: three different linear overlays and a logarithmic, decibel-oriented overlay. In addition, the MiscSoundView can overlay a separate zero line through the middle of the sound.

Scaling. The MiscSoundView has several methods for zooming to the display scale you want, if you're not into setting it directly. **zoomAllIn:** and **zoomAllOut:** set the MiscSoundView for maximum or minimum detail. **zoomInOneReduction:** and **zoomOutOneReduction:** work as you'd expect. **zoomToSelection:** attempts to come as close as possible to filling the visible portion of the MiscSoundView (within a ScrollView) with the selected portion of the sound. It does this by scaling itself to fit the selection within the visible area, and then scrolling itself in its ScrollView so the selection is displayed. **getZoomValueFrom:** sends *sender* a **floatValue** message (which should return a value between 0 and 1) and sets the scaling of the MiscSoundView to that value (where 0 is maximum detail and 1 is minimum detail).

Scrolling. The MiscSoundView can scroll itself within its ScrollView to wherever you'd like. Use **scrollToSelection:** to scroll the MiscSoundView so that the left-hand side of its selection corresponds with the left-hand side of the visible area of the view. **scrollToSample:** will try to scroll to a specific sample in the MiscSoundView. **scrollSample** returns the leftmost visible sample.

The Play Mark. When displaying the ruler, the MiscSoundView can also display a *play mark*. The play mark is a small dot directly below the ruler which moves horizontally through the sound, reflecting what part of the sound is currently being played. It's also possible to set the MiscSoundView to automatically scroll itself while playing so that the play mark is always on the left-hand side of the MiscSoundView.

To use the play mark, you'll have to create a **MiscSoundTracker.** Set the MiscSoundTracker to track *samples*, and give it a fast refresh rate (say, 0.1). Set the MiscSoundView both as the MiscSoundTracker's *sound* and as its *target,* and run the MiscSoundTracker. The MiscSoundTracker waits for the MiscSoundView to start playing. When it does, the MiscSoundTracker rapidly sends **takeIntValueFrom:** messages to the MiscSoundView, telling it the current sample being played. MiscSoundView's **takeIntValueFrom:** method sends a **intValue** message to the MiscSoundTracker, and updates the play mark to the integer value received. **Note:** when the sound stops playing, the MiscSoundTracker won't say so to the MiscSoundView; it will simply stop sending **takeIntValueFrom:** to the MiscSoundView. This means that the play mark will be stuck at its final position (usually just a little ways away from the end of the played portion of the sound). You can get rid of the play yourself by calling **[***miscSoundView* **setPlayMark:-1]**. The easiest way to do this is to call it when the MiscSoundView's delegate method is sent the **soundDidPlay:** message.

Fixing SoundView's Bugs. There are a number of bugs in the SoundViewÐMiscSoundView tries to cover over only some of them. For one, bugs in the **selectAll:**, **setSelection:**, and **getSelection:** methods often return selection values much larger than the size of the actual selected sound. The MiscSoundView overrides these methods to attempt to fix this. Also, SoundView stores its reduction factors as floats, when actually it's only able to use them as ints (it just displays the floor of the current reduction factor). This makes zooming (scaling the view) a mess. The MiscSoundView's zoom methods deal with this so you don't have to.

Bugs: What's a SoundView without bugs? Even the MiscSoundView has introduced a few fun bugs of its own. In addition to the MiscSoundView's play mark and ruler bugs (described above), there may also be a

problem with the MiscSoundView's overriding of the **setSelection:** and **getSelection:** methods to return proper valuesÐthere is some small evidence the SoundView itself may internally rely on the incorrect values to do its work! It's possible that proper values conflict with the SoundView's **readSoundfile:** and **writeSoundfile:** methods. The evidence isn't strong enough to warrant any worry yet, though you should be warned.

Instance Variables

BOOL	**display_x_axis_marks;**
float	**minor_tick_spacing;**
int	**minor_tick_spacing_format;**
int	**major_tick_spacing;**
BOOL	**display_labels;**
float	**play_mark;**
float	**old_play_mark;**
BOOL	**display_y_axis_grid;**
BOOL	**display_zero_line;**
int	**y_display_format;**
BOOL	**scroll_to_reflect_playing;**
BOOL	**only_change_play_mark;**
BOOL	**wipe_clean;**

display_x_axis_marks	Display the ruler (the ªx axisº marks)
minor_tick_spacing	The number of units per minor tick
minor_tick_spacing_format	The minor tick unit type (samples, seconds, or percentage)
major_tick_spacing	The number of minor ticks per major tick
display_labels	Display labels on the ruler
play_mark	Position (in samples) of the play mark (-1 if none)
old_play_mark	Position (in samples) of the previous play mark (-1 if none)
display_y_axis_grid	Display an amplitude grid (the ªy axisº marks)
display_zero_line	Display the zero line
y_display_format	The type of amplitude grid (16, 20, or 24 lines, or a decibel grid)
scroll_to_reflect_playing	Try to scroll the MiscSoundView when playing so the play mark is always in the left-hand side of the screen.
only_change_play_mark	On next drawSelf:, only draw the play mark

wipe_clean	On next drawSelf:, erase and completely redraw the ruler

Method Types

Fixing SoundView bugs	- getSelection:size: - setSelection:size: - selectAll:
Setting Parameters	- set:::::::::: - toggleXAxisDisplayed: - toggleYAxisDisplayed: - toggleLabelsDisplayed: - toggleZeroLineDisplayed: - toggleScrollToReflectPlaying:
Querying Parameters	- xAxisDisplayed - yAxisDisplayed - zeroLineDisplayed - labelsDisplayed - scrollToReflectPlaying - majorTickSpacing - minorTickSpacing - minorTickSpacingFormat - yDisplayFormat
Zooming (Scaling)	- zoomAllIn: - zoomAllOut: - zoomInOneReduction: - zoomOutOneReduction: - zoomToSelection: - getZoomValueFrom:
Scrolling	- scrollToSelection: - scrollToSample: - scrollSample
Setting the Play Mark	- setPlayMark: - takeIntValueFrom:

Fixing the Ruler Display - adjustBounds:

Instance Methods

adjustBounds:
- **adjustBounds:***sender*

Modifies the MiscSoundView's bounds and frame to be able to display the ruler if necessary. If the MiscSoundView
isn't displaying a ruler, and it's supposed to, it may not have been given a chance to update itself by Interface Builder
or your code (it updates itself when given a **sizeTo:** method, or in **awakeFromNib**). You can call this method manually to give it this chance. Be *certain* to have placed the MiscSoundView within a visible ScrollView before calling the method.

getSelection:size:
- **getSelection:**(int *)*firstSample* **size:**(int *)*sampleCount*

Returns the first sample of the MiscSoundView's selection in *firstSample* and the total number of selected samples in *sampleCount*.

getZoomValueFrom:
- **getZoomValueFrom:***sender*

Calls **[sender floatValue]**, which should return a number between 0 and 1 inclusive. Based on this, **getZoomValue** scales the MiscSoundView to some level of detail between 0 (maximum) and 1 (minimum).

labelsDisplayed:
- (BOOL) **labelsDisplayed**

Returns YES if the ruler labels are displayed when the ruler is displayed. For more information, see **set::::::::::**.

majorTickSpacing:
- (int) **majorTickSpacing**

Returns the major tick spacing. For more information, see **set::::::::::**.

minorTickSpacing:
- (float) **majorTickSpacing**

Returns the minor tick spacing. For more information, see **set::::::::::**.

minorTickSpacingFormat:
- (int) **minorTickSpacingFormat**

Returns the minor tick spacing format. For more information, see **set::::::::::**.

scrollSample
- (int) **scrollSample**

Returns the leftmost visible sample in the MiscSoundView.

scrollToReflectPlaying:
- (BOOL) **scrollToReflectPlaying**

Returns YES if the MiscScrollView tries to scroll itself to keep up with the movement of the play mark. For more information, see **set::::::::::**.

scrollToSample:
- **scrollToSample:**(int)*samp*

Tries to scroll so that the sample samp corresponds with the left edge of the visible area in the MiscSoundView.

scrollToSelection:
- **scrollToSelection:***sender*

Tries to scroll so that the first sample of the MiscSoundView's selection corresponds with the left edge of the visible area in the MiscSoundView.

selectAll:
- **selectAll**:*sender*

Selects all of the MiscSoundView.

set::::::::::

- **set:**

(BOOL) *displayXAxis*:
(BOOL) *displayYAxis*:
(BOOL) *displayLabels*:
(BOOL) *displayZeroLine*:
(int) *majorTickSpacing*:
(float) *minorTickSpacing*:
(int) *minorTickSpacingFormat*:
(int) *thisYDisplayFormat*:
(BOOL) *scrollToReflectPlaying*

Sets the MiscSoundView's parameters. First the flags:

- *displayXAxis* displays the ruler and play mark strip.
- *displayYAxis* overlays a light gray amplitude grid on the displayed sound.
- *displayLabels* displays labels on the ruler if the ruler's displayed.
- *displayZeroLine* overlays a light gray zero line on the displayed sound.
- *scrollToReflectPlaying* makes the MiscScrollView scroll itself to keep up with the movement of the play mark.

Next the ruler parameters. A ruler is displayed using *displayXAxis* per above. Rulers are like inch-rulers: they have both large (major) and small (minor) tick marks. The major tick marks can be labeled using *displayLabels* per above. The minor tick marks are in specific unit measurements as described below. The major tick marks are measured in a certain number of minor tick marks; i.e., a major tick mark might come every four minor tick marks, or every 100 minor tick marks, and so on.

- *minorTickSpacingFormat* is the type of unit each minor tick is measured in. Units may be any of the folowing:

MISCSOUNDVIEW_XAXIS_SPACING_FORMAT_SAMPLES
MISCSOUNDVIEW_XAXIS_SPACING_FORMAT_SECONDS
MISCSOUNDVIEW_XAXIS_SPACING_FORMAT_PERCENT

MISCSOUNDVIEW_XAXIS_SPACING_FORMAT_PERCENT is the percentage of the sound prior to the tick, and runs from 0 to 100.

· *minorTickSpacing* is the amount of the unit each tick is counted in. For example, if the ticks were measured in samples, and *minorTickSpacing* is 1000, then a minor tick appears each 1000 samples. If ticks were measured in seconds, and *minorTickSpacing* is 0.1, then a minor tick appears each tenth of a second. If ticks were measured in percentage, and *minorTickSpacing* is 5, then a minor tick appears each 5 percent.

· *majorTickSpacing* is the number of minor ticks per major tick.

Lastly the overlay parameters. If *displayYAxis* is true, the MiscSoundView will display an amplitude grid over the visible sound. This grid can be in a logarithmic decimal format, or in a linear format divided into a certain number of lines (as given below).

· *thisYDisplayFormat* type of amplitude grid overlaid. It can be any of the following.

MISCSOUNDVIEW_YAXIS_DISPLAY_FORMAT_DECIBEL
MISCSOUNDVIEW_YAXIS_DISPLAY_FORMAT_SIXTEEN
MISCSOUNDVIEW_YAXIS_DISPLAY_FORMAT_TWENTY
MISCSOUNDVIEW_YAXIS_DISPLAY_FORMAT_TWENTYFOUR

setPlayMark:
- **setPlayMark:**(int)*sample*

Moves the MiscSoundView's play mark to reflect the sample *sample.* If *sample* is -1, **setPlayMark:** erases the play mark entirely.

setSelection:size:
- **setSelection**:(int)*firstSample* **size**:(int)*sampleCount*

Sets the first sample of the MiscSoundView's selection to *firstSample* and the total number of selected samples to *sampleCount*.

takeIntValueFrom:
- **takeIntValueFrom:***sender*

Calls **[sender intValue]**, which should return either -1 or a sample number between 0 and *sampleCount-1,* where *sampleCount* is the number of samples in one channel of the MiscSoundView's sound. **takeIntValueFrom:** then moves the play mark to reflect the sample using **setPlayMark:**. If the number is not -1 and the MiscSoundView is set to scroll to reflect playing, **takeIntValueFrom:** tries to scroll the view to display the sample, using **scrollToSample:**. This method is intended to allow the MiscSoundView to be used in conjunction with a MiscSoundTracker to automatically display where its playing sound is at. See the

introduction above for more information.

toggleLabelsDisplayed:
- **toggleLabelsDisplayed:***sender*

Toggles if the ruler labels are displayed or not. For more information, see **set::::::::::**.

toggleScrollToReflectPlaying:
- **toggleScrollToReflectPlaying:***sender*

Toggles whether the MiscScrollView tries to scroll itself to keep up with the movement of the play mark. For more information, see **set::::::::::**.

toggleXAxisDisplayed:
- **toggleXAxisDisplayed:***sender*

Toggles if the ruler is displayed or not. For more information, see **set::::::::::**.

toggleYAxisDisplayed:
- **toggleYAxisDisplayed:***sender*

Toggles if the amplitude grid is displayed or not. For more information, see **set::::::::::**.

toggleZeroLineDisplayed:
- **toggleZeroLineDisplayed:***sender*

Toggles if the zero line is displayed or not. For more information, see **set::::::::::**.

xAxisDisplayed:
- (BOOL) **xAxisDisplayed**

Returns YES if the ruler is displayed. For more information, see **set::::::::::**.

yAxisDisplayed:
- (BOOL) **yAxisDisplayed**

Returns YES if the amplitude grid is displayed. For more information, see **set::::::::::**.

yDisplayFormat:
- (int) **yDisplayFormat**

Returns the amplitude grid format. For more information, see **set::::::::::**.

zeroLineDisplayed:
- (BOOL) **zeroLineDisplayed**

Returns YES if the zero line is displayed. For more information, see **set::::::::::**.

zoomAllIn:
- **zoomAllIn:***sender*

Scales the MiscSoundView to maximum detail.

zoomAllOut:
- **zoomAllOut:***sender*

Scales the MiscSoundView to minimum detail.

zoomInOneReduction:
- **zoomInOneReduction:***sender*

Scales the MiscSoundView to slightly more detail (a difference of 1 in the reduction factor).

zoomOutOneReduction:
- **zoomOutOneReduction:***sender*

Scales the MiscSoundView to slightly less detail (a difference of 1 in the reduction factor).

zoomToSelection:
- **zoomToSelection:***sender*

Scales the MiscSoundView to be able to display all of the selection in its visible area, then calls

scrollToSelection: to move the selection into its visible area.