initFrame:	initFrame:mode:cellClass:numRows:numCols: initFrame:mode:prototype:numRows:numCols: free
Setting the selection mode setMode:	mode
Configuring the Matrix setEnabled:	setEmptySelectionEnabled: isEmptySelectionEnabled setSelectionByRect: isSelectionByRect
Setting the Cell class setCellClass:	setPrototype: prototype
Laying out the Matrix addCol	addRow insertColAt: insertRowAt: removeColAt:andFree: removeRowAt:andFree: makeCellAt:: putCell:at:: renewRows:cols: setCellSize: getCellSize: getCellFrame:at:: setIntercell:

setTag:target:action:at::

Selecting Cells selectCell:
selectCellAt::
selectCellWithTag:
setSelectionFrom:to:anchor:lit:
selectAll:
selectedCell
getSelectedCells:
selectedCol
selectedRow
clearSelectedCell

Finding Cells findCellWithTag:
cellAt::
cellList

Modifying graphic attributes setBackgroundColor:
backgroundColor
setBackgroundGray:
backgroundGray
setCellBackgroundColor:
cellBackgroundColor
setCellBackgroundGray:
cellBackgroundGray
setBackgroundTransparent:
isBackgroundTransparent
setCellBackgroundTransparent:
isCellBackgroundTransparent
setFont:
font

Editing text in Cells selectText:
selectTextAt::

Setting Tab key behavior setNextText:
setPreviousText:

Assigning a Text delegate setTextDelegate:
textDelegate

Text object delegate methods textWillChange:
textDidChange:
textDidGetKeys:isEmpty:
textWillEnd:
textDidEnd:endChar:

Resizing the Matrix and Cells setAutosizeCells:
doesAutosizeCells
calcSize
sizeTo::
sizeToCells
sizeToFit
validateSize:

Scrolling setAutoscroll:
setScrollable:
scrollCellToVisible::

Displaying display
drawSelf::
drawCell:

setErrorAction:
errorAction
setTarget:at::
setAction:at::
sendAction
sendAction:to:
sendAction:to:forAllCells:
sendDoubleAction
setReaction:

Handling event and action messages

acceptsFirstMouse
mouseDown:
mouseDownFlags
performKeyEquivalent:

Managing the cursor resetCursorRects

Archiving read:
write:

initFrame:...

(BOOL)acceptsFirstMouse

Returns NO if the selection mode of the Matrix is NX_LISTMODE, YES if the Matrix is in any ot
The Matrix does not accept first mouse in NX_LISTMODE to prevent the loss of multiple selectio

mode

(SEL)action

Returns the default action of the Matrix. The returned method is used when a Cell with no action r
would ordinarily cause its action to be sentÐnormally a mouse-up in the Cell. In such cases, the M
to its own target.

only if new Cells are needed (since renewRows:cols: doesn't free Cells, it just rearranges them). T
and shrink a Matrix without repeatedly creating and freeing the Cells.

insertColAt:, makeCellAt::, renewRows:cols:, isAutodisplay (View)

addRow

Adds a new row of Cells to the bottom of the existing rows, creating new Cells if needed with mak
redraw even if autodisplay is on. Returns self.

If the number of rows or columns in the Matrix has been changed with renewRows:cols:, then mak
only if new Cells are needed (since renewRows:cols: doesn't free Cells, it just rearranges them). T
and shrink a Matrix without repeatedly creating and freeing the Cells.

insertRowAt:, makeCellAt::, renewRows:cols:, isAutodisplay (View)

(NXColor)backgroundColor

Returns the color used to draw the background (the space between the Cells).

setBackgroundColor:, backgroundGray, cellBackgroundColor

(float)backgroundGray

Returns the gray level used to draw the background (the space between the Cells). If the gray leve
background is transparent.

setBackgroundGray:, backgroundColor, cellBackgroundGray

calcSize

Your code should never invoke this method. It is invoked automatically by the system if it has to r
information about the Cells. It invokes calcDrawInfo: on each Cell in the Matrix. Can be overridd
necessary (Form overrides calcSize, for example). Returns self.

calcSize (Control, Form), validateSize:

cellAt:(int)row :(int)col

Returns the Cell at row row and column col, or nil if no such Cell exists.

getRow:andCol:ofCell:

(NXColor)cellBackgroundColor

Returns the color used to fill the background of a Cell.

(int)cellCount

Returns the number of Cell positions in the Matrix (that is, the number of rows times the number o

cellList

cellList

Returns a List object that contains the Cells of the Matrix. The Cells in the list are row-ordered tha
Cells appear first in the List, then the next row, and so on.

clearSelectedCell

Deselects the selected Cell or Cells, and returns the previously selected Cell (the last of the selecte
more than one). If the selection mode is NX_RADIOMODE and empty selection is not allowed, t
deselect the selected Cell. Doesn't redisplay the Matrix. It's often more convenient to use selectC
column of (1, 1), since this will clear the selected Cell and redisplay the Matrix.

selectCellAt::, mode, setEmptySelectionEnabled:

display

Draws the Matrix. This method invokes displayFromOpaqueAncestor::: if any part of the Matrix (
between Cells, or any Cell) is transparent, or display::: if the entire Matrix is opaque. Returns self.

display::: (View), displayFromOpaqueAncestor::: (View)

(BOOL)doesAutosizeCells

Returns YES if Cells are resized proportionally to the Matrix when its size changes the inter-Cell s
Returns NO if the inter-Cell spacing changes when the Matrix is resized the Cell size remains cons

setAutosizeCells:

(SEL)doubleAction

Returns the action sent by the Matrix to its target when the user double-clicks an entry. Unlike NX
returns NULL if there is no double-click action. The double-click action of a Matrix is sent after t
click action (for the Cell clicked or for the Matrix if the Cell doesn't have its own action). If there
action and the Matrix doesn't ignore multiple clicks, the single-click action is sent twice.

setDoubleAction:, action, target, sendDoubleAction, ignoreMultiClick: (Control)

drawCellAt:(int)row :(int)col

Displays the Cell at (row, col) if it's in the Matrix. Does nothing otherwise. Returns self.

drawCell:, drawCellInside:

drawCellInside:aCell

If aCell is in the Matrix, then its inside (usually all but a bezel or border) is drawn.

drawCell:, drawCellAt::, drawInside:inView: (Cell)

drawSelf:(const NXRect *)rects :(int)rectCount

Displays the Cells in the Matrix which intersect any of the rects.

(SEL)errorAction

Returns the action sent to the target of the Matrix when the user enters an illegal value for a Cell's setEntryType: method and checked by Cell's isEntryAcceptable: method).

setErrorAction:, setEntryType: (Cell), isEntryAccectable: (Cell)

findCellWithTag:(int)anInt

Returns the Cell which has a tag matching anInt, or nil if no such Cell exists in the Matrix.

setTag:at::, setTag: (ActionCell), setTag:target:action:at::, selectCellWithTag:

font

Returns the Font used to display text in the Cells of the Matrix, or nil if the Cells don't contain text

setFont:

free

Deallocates the storage for the Matrix and all its Cells, and returns nil.

getCellFrame:(NXRect *)theRect
at:(int)row
:(int)col

getCellFrame:at::, getIntercell:

getIntercell:(NXSize *)theSize

Returns self, and by reference in theSize the vertical and horizontal spacing between Cells.

getCellSize:

getNumRows:(int *)rowCount numCols:(int *)colCount

Returns self, and, by reference in rowCount and colCount, the number of rows and columns in the

getRow:(int *)row
andCol:(int *)col
forPoint:(const NXPoint *)aPoint

Returns the Cell at aPoint in the Matrix. aPoint must be in the coordinate system of the Matrix. If bounds of the Matrix or in an intercell spacing, getRow:andCol:forPoint: returns nil. Also returns col the row and column position of the Cell.

getRow:andCol:ofCell:

getRow:(int *)row
andCol:(int *)col
ofCell:aCell

Returns by reference in row and col the row and column indices for the position of aCell within the if it's in the Matrix, nil otherwise.

getRow:andCol:forPoint:

getSelectedCells:(List *)aList

Adds to aList the Cells of the Matrix that are selected. If aList is nil, a new List object is created a selected Cells. Your code may free the List object, but not the Cells in the List. Returns the List c

selectedCell

highlightCellAt:(int)row
:(int)col
lit:(BOOL)flag

Highlights or unhighlights the Cell at (row, col) in the Matrix by sending highlight:inView:lit: to th focus must be locked on the Matrix when this message is sent. Returns self.

new Matrix contains no rows or columns. The default mode is NX_RADIOMODE.

initFrame:mode:...

initFrame:(const NXRect *)frameRect
mode:(int)aMode
cellClass:classId
numRows:(int)numRows
numCols:(int)numCols

Initializes and returns the receiver, a new instance of Matrix, in frameRect with numRows rows an
aMode is set as the tracking mode for the Matrix, and can be one of four constants:

NX_TRACKMODEJust track the mouse inside the Cells
NX_HIGHLIGHTMODE Highlight the Cell, then track, then unhighlight
NX_RADIOMODEAllow no more than one selected Cell
NX_LISTMODEAllow multiple selected Cells

The behavior for these constants is more fully described in the class description. The new Matrix c
of class classId, which should be the return value of a class message sent to a subclass of Cell.

This method is the designated initializer for Matrices that add Cells by creating instances of a Cell

initFrame:, initFrame:mode:prototype:numRows:numCols:

initFrame:(const NXRect *)frameRect
mode:(int)aMode
prototype:aCell
numRows:(int)numRows
numCols:(int)numCols

Initializes and returns the receiver, a new instance of Matrix, in frameRect with numRows rows an
aMode is set as the tracking mode for the Matrix, and can be one of four constants:

NX_TRACKMODEJust track the mouse inside the Cells
NX_HIGHLIGHTMODE Highlight the Cell, then track, then unhighlight
NX_RADIOMODEAllow no more than one selected Cell
NX_LISTMODEAllow multiple selected Cells

The behavior for these constants is more fully described in the class description. The new Matrix c
copying aCell, which should be an instance of a subclass of Cell.

This method is the designated initializer for Matrices that add Cells by copying an instance of a Ce

initFrame:, initFrame:mode:cellClass:numRows:numCols:

insertColAt:(int)col

Inserts a new column of Cells before col, creating new Cells with makeCellAt::. If col is greater th
columns in the Matrix, enough columns are created to expand Matrix to be col columns wide. This
redraw even if autodisplay is on. Your code may need to use sizeToCells after sending this metho
fit the newly added Cells. Returns self.

Inserts a new row of Cells before row, creating new Cells with makeCellAt:... If row is greater tha
the Matrix, enough rows are created to expand Matrix to be row rows high. This method doesn't r
autodisplay is on. Your code may need to use sizeToCells after sending this method to resize the M
added Cells. Returns self.

If the number of rows or columns in the Matrix has been changed with renewRows:cols:, then mak
only if new Cells are needed (since renewRows:cols: doesn't free Cells, it just rearranges them). T
and shrink a Matrix without repeatedly creating and freeing the Cells.

addRow, insertColAt:, sizeToCells, makeCellAt::

(BOOL)isBackgroundTransparent

Returns YES if the Matrix background is transparent, NO otherwise.

setBackgroundTransparent:, backgroundGray

(BOOL)isCellBackgroundTransparent

Returns YES if Cells in the Matrix have transparent backgrounds, NO otherwise.

setCellBackgroundTransparent:, cellBackgroundGray

(BOOL)isEmptySelectionEnabled

Returns YES if it is possible to have no Cells selected in a radio-mode Matrix, NO otherwise.

setEmptySelectionEnabled:

(BOOL)isSelectionByRect

Returns YES if a rectangle of Cells in the Matrix can be selected by dragging the cursor, NO other

setSelectionFrom:to:anchor:lit:

makeCellAt:(int)row :(int)col

Creates a new Cell at the specified location in the Matrix. If the Matrix has a prototype Cell, it's c
Cell if the Matrix has a Cell class set, it allocates and initializes (with init) an instance of that class
had a Cell class set, the default class, ActionCell, is used. The new Cell's font is set to the font of
newly created Cell.

Your code should never invoke this method directly it's used by addRow and other methods when
It may be overridden to provide more specific initialization of Cells.

addCol, addRow, insertColAt:, insertRowAt:

Your code should never invoke this method, but you may override it to implement different mouse
does. The response of the Matrix depends on its selection mode, as explained in the class descripti

In any selection mode, a mouse-down in an editable text Cell immediately enters text editing mode
any other kind of Cell sends the double-click action of the Matrix (if there is one) in addition to the

sendAction, sendDoubleAction

(int)mouseDownFlags

Returns the flags (for example, NX_SHIFTMASK) that were in effect at the mouse-down event th
tracking session. Use this method if you want to access these flags, but don't want the overhead of
sendActionOn: to add NX_MOUSEDOWNMASK to every Cell to get them. This method is valid
it's not useful if the target of the Matrix initiates another tracking loop as part of its action method
a PopUpList does, for example).

sendActionOn: (Cell)

(BOOL)performKeyEquivalent:(NXEvent *)theEvent

If there is a Cell in the Matrix that has a key equivalent equal to the character in theEvent->data.ke
made to react as if the user had clicked it by highlighting, changing its state as appropriate, sending
and then unhighlighting. Returns YES if a Cell in the Matrix responds to the key equivalent in the
responds.

Your code should never send this message it is sent when the Matrix or one of its superviews is the
user presses a key. You may want to override this method to change the way key equivalents are p
or to disable them in your subclass.

prototype

Returns the prototype Cell that is copied whenever a new Cell needs to be made, or nil if there is n

setPrototype:, initFrame:mode:prototype:numRows:numCols:, makeCellAt::

putCell:newCell
at:(int)row
:(int)col

Replaces the Cell at (row, col) by newCell, and returns the old Cell at that position. Draws the new
on.

read:(NXTypedStream *)stream

Reads the Matrix from the typed stream stream. Returns self.

write:

removeRowAt:(int)row andFree:(BOOL)flag

Removes the row at position row. If flag is YES then the Cells from that row are freed. Doesn't re
autodisplay is on. Your code should normally send sizeToCells after invoking this method to resiz
the reduced Cell count. Returns self.

removeColAt:andFree:, addRow, insertRowAt:

renewRows:(int)newRows cols:(int)newCols

Changes the number of rows and columns in the Matrix. This method uses the same Cells as befor
only if the new size is larger it never frees Cells. Doesn't display the Matrix even if autodisplay is
normally send sizeToCells after invoking this method to resize the Matrix so it fits the changed Ce
method deselects all Cells in the Matrix. Returns self.

addRow, addCol

resetCursorRects

Sends resetCursorRect:inView: to each Cell in the Matrix. Any Cell that has a cursor rectangle to
message addCursorRect:cursor: back to the Matrix. Returns self.

resetCursorRect:inView: (Cell), addCursorRect:cursor: (View)

scrollCellToVisible:(int)row :(int)col

If the Matrix is in a scrolling View, then the Matrix will scroll to make the Cell at (row, col) visibl

scrollRectToVisible: (View)

selectAll:sender

If the mode of the Matrix is not NX_RADIOMODE, then all the Cells in the Matrix are selected ar
Matrix is redisplayed. The currently selected Cell is unaffected. Editable text Cells are not affecte

selectCell:, selectCellAt::, selectCellWithTag:, selectText:

selectCell:aCell

If aCell is in the Matrix, then the Cell is selected, the Matrix is redrawn, and the selected Cell is re
Cell's text is selected. Returns nil if the Cell is not in the Matrix.

selectCellAt::, selectCellWithTag:, selectAll:, selectText:

selectCellWithTag:(int)anInt

If the Matrix has a Cell whose tag is equal to anInt, that Cell is selected. An editable text Cell's te
self, or nil if there is no such Cell.

selectCell:, selectCellAt::, selectAll:, selectText:

selectedCell

Returns the currently selected Cell, or nil if no Cell is selected. If more than one Cell is selected, r
Cell that is, the Cell that is lowest and furthest to the right in the Matrix.

getSelectedCells:

(int)selectedCol

Returns the column number of the selected Cell, or 1 if no Cells are selected. If Cells in multiple c
this method returns the number of the last column containing a selected Cell.

selectedRow

(int)selectedRow

Returns the row number of the selected Cell, or 1 if no Cells are selected. If Cells in multiple rows
method returns the number of the last row containing a selected Cell.

selectedCol

selectText:sender

If sender is the next Text object of the Matrix (as set with setNextText:), the text in the last selecta
lowest and furthest to the right) is selected otherwise, the text of the first selectable text Cell is sele
whose text was selected, the Matrix if such a Cell wasn't found, and nil if the Cell was bound but w
wasn't selectable.

selectTextAt::, selectText: (TextField)

selectTextAt:(int)row :(int)col

Select the text of the Cell at (row, col) in the Matrix, if there is such a Cell and its text is selectable
whose text was selected, the Matrix if such a Cell wasn't found, and nil if the Cell was found but w
selectable.

selectText:, selectText: (TextField)

sendAction:(SEL)theAction to:theTarget

If both theAction and theTarget are non-null, sends theAction to theTarget. If theAction is null, se
Matrix to its target. If theAction is nil, sends theAction to the target of the Matrix. Returns nil if n
to theAction could be found otherwise returns self.

Your code shouldn't normally invoke this method. It is used by event handling methods such as C
inRect:ofView: to send an action to a target in response to an event within the Matrix.

sendAction, sendAction:to: (Control)

sendAction:(SEL)aSelector
to:anObject
forAllCells:(BOOL)flag

Iterates through the Cells in the Matrix, sending aSelector to anObject for each. aSelector must rep
takes a single argument: the id of the current Cell in the iteration. aSelector's return value must be
begins with the Cell in the upper-left corner of the Matrix, proceeding through all entries in the firs
next. Returns self.

If aSelector returns NO for any Cell, this method terminates immediately and return self, without s
other Cells. If it returns YES, this method keeps sending the message.

This method is not invoked to send action messages to target objects in response to mouse-down e
Instead, you can invoke it if you want to have multiple Cells in a Matrix interact with an object. F
use it to verify the titles in a list of items, or to enable a series of radio buttons based on their purpo
anObject.

sendAction:to:

sendDoubleAction

If the Matrix has a double-click action, sends that message to the target of the Matrix. If not, then
returned by selectedCell) has an action, that message is sent to the selected Cell's target. If the sel
action, then the action of the Matrix is sent to the target of the Matrix. This method only sends an
Cell is enabled. Returns self.

Your code shouldn't invoke this method it's sent in response to a double-click event in the Matrix.
override it to change the search order for an action to send.

sendAction, sendAction:to:, ignoreMultiClick: (Control)

setAction:(SEL)aSelector

Sets the default action of the Matrix, the message sent for a Cell which has no action of its own. T
is always sent to its target, never to the Cell's target. Returns self.

action, setDoubleAction, setTarget:, setAction:at::, setTarget:at::

setAction:(SEL)aSelector

If flag is YES and the Matrix is in a scrolling View, it will be automatically scrolled whenever a th
outside the Matrix after a mouse-down event within its bounds. Returns self.

setAutosizeCells:(BOOL)flag

If flag is YES, then whenever the Matrix is resized, the sizes of the Cells changes in proportion, ke
space constant further, this method verifies that the Cell sizes and inter-Cell spacing add up to the
Matrix, adjusting the size of the Cells and updating the Matrix if they don't. If flag is NO, then the
changes when the Matrix is resized, with the Cell size remaining constant. Returns self.

doesAutosizeCells, update (Control)

setBackgroundColor:(NXColor)aColor

Sets the background color for the Matrix to aColor. This color is used to fill the space between Ce
any non-opaque Cells. Doesn't redraw the Matrix even if autodisplay is on. Returns self.

backgroundColor, setBackgroundGray:, setCellBackgroundColor:, isAutodisplay (View)

setBackgroundGray:(float)value

Sets the background gray level for the Matrix to value. This gray level is used to draw the inter-Ce
behind any non-opaque Cells. If the gray level is 1, the background is transparent (that is, doesn't
the Matrix if the background gray level changes. Returns self.

backgroundGray, setBackgroundColor:, setCellBackgroundGray:, update (Control)

setBackgroundTransparent:(BOOL)flag

If flag is YES, sets the background gray level of the Matrix to 1 (transparent). If flag is NO, set th
to NX_WHITE.

isBackgroundTransparent, setBackgroundGray:

setCellBackgroundColor:(NXColor)aColor

Sets the background color for the Cells in the Matrix to aColor. This color is used to fill the space
Cells. Doesn't redraw the Matrix even if autodisplay is on. Returns self.

cellBackgroundColor, setCellBackgroundGray:, setBackgroundColor:, isAutodisplay (View)

setCellBackgroundGray:(float)value

Sets the background gray level for the Cells in the Matrix to value. This gray level is used to draw

gray level to NX_WHITE.

isCellBackgroundTransparent, setCellBackgroundGray:

setCellClass:classId

Configures a single Matrix to use instances of classId when creating new Cells. classId should be Cell, obtained by sending the class message to either the Cell subclass object or to an instance of th class is that set with the class method setCellClass: the default Cell class is ActionCell. Returns se

You only need to use this method with Matrices initialized with initFrame:, since the other initializ specify an instance-specific Cell class or Cell prototype.

setPrototype:, initFrame:

setCellSize:(const NXSize *)aSize

Sets the width and the height of each of the Cells in the Matrix to those in aSize. This may change Does not redraw the Matrix, even if autodisplay is on.

getCellSize:, calcSize, isAutodisplay (View)

setDoubleAction:(SEL)aSelector

Make aSelector the action sent to the target of the Matrix when the user double-clicks a Cell. A do always sent after the appropriate single-click action the Cell's if it has one, otherwise the single-cli Returns self.

If a Matrix has no double-click action set, then by default a double-click is treated as a single-click click action also sets allowMultiClick: to YES be sure to set the Matrix to ignore multiple-clicks if double-click action.

doubleAction, setAction:, setTarget:, ignoreMultiClick: (Control)

setEmptySelectionEnabled:(BOOL)flag

If flag is YES, then the Matrix will allow one or zero Cells to be selected. If flag is NO, then the M and only one Cell (not zero Cells) to be selected. This setting has effect only in NX_RADIOMOD

This method replaces the allowEmptySel: method of NEXTSTEP Release 2.

isEmptySelectionEnabled

setEnabled:(BOOL)flag

If flag is YES, enables all Cells in the Matrix if NO, disables all Cells. If autodisplay is on, this re Returns self.

isEnabled, setEnabled: (ActionCell), isAutodisplay (View)

setFont:fontObject

Sets the Font for the Matrix to fontObject. This will cause all current Cells to have their Font char
well as cause all future Cells to have that Font. If autodisplay is on, this redraws the entire Matrix.

font, isAutodisplay (View)

setIcon:(const char *)iconName
at:(int)row
:(int)col

Sets the icon of the Cell at (row, col) to the NXImage with the name iconName. If autodisplay is o
redrawn. Returns self.

setIcon: (ButtonCell, Cell), isAutodisplay (View)

setIntercell:(const NXSize *)aSize

Sets the width and the height of the space between Cells to those in aSize. Doesn't redraw the Ma
is on. Returns self.

getIntercell:, isAutodisplay (View)

setMode:(int)aMode

Sets the selection mode of the Matrix. aMode can be one of four constants:

NX_TRACKMODEJust track the mouse inside the Cells
NX_HIGHLIGHTMODE Highlight the Cell, then track, then unhighlight
NX_RADIOMODEAllow no more than one selected Cell
NX_LISTMODEAllow multiple selected Cells

The behaviors associated with these constants are explained in the class description.

mode

setNextText:anObject

Sets anObject as the object whose text is selected when the user presses Tab while editing the last
anObject should respond to the selectText: message. If anObject also responds to both selectText:
it's sent setPrevious: with the receiving Matrix as the argument this builds a two-way connection, s
the last text Cell selects anObject's text, and pressing Shift-Tab in anObject selects the last text Ce
Returns self.

setPreviousText:, selectText:

setPrototype:aCell

Sets the prototype Cell that is copied whenever a new Cell needs to be made. aCell should be an i
Cell. If a Matrix has a prototype Cell, it doesn't use its Cell class object to create new Cells if you
use its Cell class, invoke this method with nil as the argument. The Matrix is considered to own th
free it when the Matrix is itself freed be sure to make a copy of an instance that your code may use
the old prototype Cell, or nil if there wasn't one.

If you implement your own Cell subclass for use as a prototype with a Matrix, make sure your Cel
when it receives a copy message. For example, Object's copy copies only pointers, not what they
this is what it should do, sometimes not. The best way to implement copy when you subclass Cell
then copy instance variable values (for example, title strings) into your subclass instance individua
that freeing the prototype will not damage any of the copies that were made and put into the Matrix
pointers that are freed, for example).

prototype, initFrame:mode:prototype:numRows:numCols:

setReaction:(BOOL)flag

Sent to the Matrix by the target of an action message. If flag is NO, prevents the selected Cell fron
previous state if YES, allows it to revert to its previous state (to reflect unhighlighting, for example
action method if the action causes the Cell to change in such a way that trying to unhighlight it wo
example, if the Cell is deleted or its visual appearance completely changed by the action method.

setScrollable:(BOOL)flag

Sets all the Cells to be scrollable, so that the text they contain scrolls to remain in view if the user t
the Cell. Returns self.

setScrollable: (Cell)

setSelectionByRect:(BOOL)flag

If flag is YES, a rectangle of Cells in the Matrix can be selected by dragging the cursor if flag is N
possible.

isSelectionByRect, setSelectionFrom:to:anchor:lit:

setSelectionFrom:(int)startPos
to:(int)endPos
anchor:(int)anchorPos
lit:(BOOL)flag

Programmatically selects a range of Cells. startPos, endPos, and anchorPos are Cell positions, cou
upper left Cell of the Matrix, rows before columns. For example, the third Cell in the top row wou

startPos and endPos are used to mark where the user would have pressed the mouse button and rele

Sets the state of the Cell at row row and column col to value. For radio-mode Matrices, this is ide
except that the state can be set to any arbitrary value. If autodisplay is on, redraws the affected Ce
radio mode, the Cell is redrawn regardless of the setting of autodisplay. Returns self.

setState: (Cell), selectCellAt::, isAutodisplay (View)

setTag:(int)anInt
at:(int)row
:(int)col

If there's a Cell at (row, col), sets that Cell's tag to anInt and returns self.

setTag:target:action:at::, setTag: (ActionCell)

setTag:(int)anInt
target:anObject
action:(SEL)aSelector
at:(int)row
:(int)col

If there's a Cell at (row, col), sets that Cell's tag, target, and action to anInt, anObject, and aSelecto
self.

setTag:at::, setTarget:at::, setAction:at::

setTarget:anObject

Sets the target object of the Matrix. This is the object to which actions will be sent for Cells that d
target. Returns self.

target, action

setTarget:anObject
at:(int)row
:(int)col

If there's a Cell at (row, col), sets that Cell's target to anObject and returns self.

setTag:target:action:at::, setTarget:, setTarget: (ActionCell)

setTextDelegate:anObject

Sets the object to which the Matrix will forward messages from the field editor. These messages i
textWillEnd:, textDidEnd:endChar:, textWillChange:, and textDidChange:. Returns self.

textDelegate, Text class delegate methods

sizeTo:(float)width :(float)height

Resizes the Matrix to width and height, but doesn't redraw it. If the Matrix has been set to autosiz
resized proportionally to the change in size of the Matrix, keeping the inter-Cell spacing constant i
autosize, then the inter-Cell spacing is adjusted, and the Cells remain the same size. If editing is g
it's aborted after the Matrix is redrawn, the text is reselected to allow editing to continue. Returns

sizeToCells, sizeToFit, setAutosizeCells:, selectText:

sizeToCells

Changes the width and the height of the Matrix frame so that it exactly contains the Cells. Does no
Returns self.

sizeTo::, sizeToFit

sizeToFit

Changes the Cell size to accommodate the Cell with the largest contents in the Matrix, then change
height of the Matrix frame so that it exactly contains the Cells. Doesn't redraw the Matrix. Return

sizeTo::, sizeToCells, calcCellSize: (Cell)

target

Returns the target of the Matrix. This object receives action messages for Cells that don't have the
and receives all double-click action messages.

setTarget:, setTarget:at::, action

textDelegate

Returns the object that receives messages passed on by the Matrix from the field editor. The field
the TextField class specification, is the Text object used to draw text in all Cells in a Window.

setTextDelegate:

textDidChange:textObject

Passes this message on, with the same argument, to the Text delegate of the Matrix. Override this
your subclass of Matrix to act as the field editor's delegate. Returns self.

textDidChange: (Text class delegate method)

You may want to override this method to interpret more characters (such as the Enter or Escape ke

sendAction, setNextText:, setPreviousText:, textDidEnd:endChar: (Text class delegate method)

textDidGetKeys:textObject isEmpty:(BOOL)flag

Passes this message on, with the same argument, to the Text delegate of the Matrix. Override this your subclass of Matrix to act as the field editor's delegate. Returns self.

textDidGetKeys:isEmpty: (Text class delegate method)

(BOOL)textWillChange:textObject

Invoked automatically during editing to determine if it is OK to edit the selected text. This method Cell is editable and sends textWillChange: to the TextField's Text delegate to allow it to respond. isn't editable NO if the text is editable but the Text delegate doesn't respond to textWillChange: th value for textWillChange: if the Text delegate responds to it.

setEditable: (Cell), setTextDelegate:, textWillChange: (Text class delegate method)

(BOOL)textWillEnd:textObject

Invoked automatically before text editing ends. Checks the text by sending isEntryAcceptable: to the entry isn't acceptable, sends the error action to the target. This method is then passed on to the same argument. The return value is based on whether the entry is acceptable and on the return valu delegate. If the delegate responds to textWillEnd:, then the return value is NO only if the entry is a delegate returns NO. Otherwise the return value is YES to indicate that editing shouldn't end, and a beep (to indicate an error in the entry).

isEntryAcceptable: (Cell), setTextDelegate:, textWillEnd: (Text class delegate method)

validateSize:(BOOL)flag

If flag is YES, then the size information in the Matrix is assumed correct. If flag is NO, then calcS before any further drawing is done. Returns self.

calcSize

write:(NXTypedStream *)stream

Writes the receiving Matrix to the typed stream stream. Returns self.

read: