

MiscSelectionMatrix

Inherits From:	Matrix : Control : View : Responder : Object
Conforms To:	none
Declared In:	misckit/MiscSelectionMatrix.h

Class Description

MiscSelectionMatrix is a subclass of Matrix which allows easily to select and deselect multiple cells directly from your source. In addition it keeps the selection order.

Note:

- ordering problem;
- dragging not ready;
- cell move/exchange methods will be introduced with version 1.0

Instance Variables

List *

selectionOrder;

selectionOrder

An ordered List of selected cells

Method Types

Initializing the class

+€initialize

Creating and freeing instances -

- initFrame:mode:prototype:numRows:numCols:
- initFrame:mode:cellClass:numRows:numCols:
- copyFromZone:
- free

Overridden methods

- getSelectedCells:
- mouseDown:

Selecting Cells

- addSelectionAt::
- addSelectionForCell:
- addSelectionForCellWithTag:
- addSelectionForColumn:
- addSelectionForRow:
- deselectCell:
- deselectCellAt::
- deselectCellWithTag:

- deselectColumn:
- deselectRow:

Archiving
- write:

- read:

Class Methods

initialize
+ **initialize**

Initializes the class instance doing things like setting the class version number.

Instance Methods

addSelectionAt::
- **addSelectionAt:(int)row:(int)col**

addSelectionForCell:
- **addSelectionForCell:aCell**

If *aCell* is in the MiscSelectionMatrix, then the Cell is selected and added to the selection list, the Matrix is redrawn, and the selected Cell is returned. Returns **nil** if the Cell is not in the Matrix.

addSelectionForCellWithTag:
- **addSelectionForCellWithTag:(int)aTag**

addSelectionForColumn:

- **addSelectionForColumn:**(int)*col*

addSelectionForRow:

- **addSelectionForRow:**(int)*row*

copyFromZone:

- **copyFromZone:**(NXZone *)*zone*

Makes a copy of the matrix. The cells in the selection list aren't copied; therefore, both selection lists contain pointers to the same set of cells. Memory for the new List is allocated from *zone*

deselectCell:

- **deselectCell:***aCell*

If *aCell* is in the MiscSelectionMatrix, then the Cell is deselected and deleted from the selection list, the Matrix is redrawn, and the selected Cell is returned. Returns **nil** if the Cell is not in the Matrix.

deselectCellAt::

- **deselectCellAt:**(int)*row* :(int)*col*

deselectCellWithTag:

- **deselectCellWithTag:**(int)*aTag*

deselectColumn:

- **deselectColumn:**(int)*col*

deselectRow:

- **deselectRow:**(int)*row*

deselectRow:

- **deselectRow:**(int)*row*

- **free**

- **free**

Frees the memory used for the ordered selection list.

- **getOrderedSelectedCells:**

- **getOrderedSelectedCells:**(List *)*aList*

Adds to *aList* the Cells of the Matrix that are selected, keeping the order of the selection (user and from insight the prham). If *aList* is **nil**, a new List object is created and filled with the selected Cells. Your code may free the List object, but not the Cells in the List. Returns the List containing the Cells.

See also: - **getSelectedCells:** (Matrix)

initFrame:mode:cellClass:numRows:numCols:

- **initFrame:**(const NXRect *)*frm* **mode:**(int)*aMode* **cellClass:***factory* **numRows:**(int)*rowsHigh* **numCols:**(int)*colsWide*

Initialize the ordered selection list

See also: ±**initFrame:mode:prototype:numRows:numCols:**, ±**copyFromZone:**, ± **free**

initFrame:mode:prototype:numRows:numCols:

- **initFrame:**(const NXRect *)*frm* **mode:**(int)*aMode* **prototype:***protoCell* **numRows:**(int)*rowsHigh* **numCols:**

(int)*colsWide*

Initialize the ordered selection list

See also: ±€**initFrame:mode:cellClass:numRows:numCols:**, ±€**copyFromZone:**, ± **free**

read

- **read:**(NXTypedStream *)*strm*

Reads the object from the typed stream. Overriden to read in the ordered selection list.

See also: ±€**write:**

write

- **write:**(NXTypedStream *)*strm*

Writes the object to the typed stream. Overriden to write the ordered selection list.

See also: ±€**read:**