MacApp® 2.0b5 UGridView Release Notes

Deb Orton

Overview

The purpose of the UGridView module is to provide the view that goes "on top" of a list or grid framework. UGridView knows nothing about the underlying cell contents (with the exception of the text methods which work for text grids or lists only); therefore, the user must (at least) provide a DrawCell method (or a GetText method in the TextView cases). The height and width of rows and columns are variable although optimized for a fixed value. The standard cell selection algorithms are provided (as described by the List Manager in Inside Macintosh, Volume IV), as well as a method for enabling single cell selection only. Selection through a method call is provided. View templates exist for all the view objects in this unit.

The following classes are defined in UGridView:

`TGridView`	For displaying a 1- or 2-dimensional grid of cells containing anything (text, picts, subviews, and so on).
`TTextGridView`	For displaying a 1- or 2-dimensional grid of cells containing only text.
`TTextListView`	For displaying a 1-dimensional grid of cells containing only text.
`TGridSelectCommand`	For building the other command objects.
`TCellSelectCommand`	For selecting a cell.
`TRowSelectCommand`	For selecting a row divider.
`TColSelectCommand`	For selecting a column divider.
`TVertexSelectCommand`	For selecting a row and column divider.

TGridView

Types

```
GridCell = Point;
```
A cell in the grid.

```
GridViewPart = (badChoice, inCell,
                    inRow, inColumn, inVertex);
```
Used to type mouse clicks.

```
RunArrayChunk = RECORD
                      count:    INTEGER;
                      size:     INTEGER;
                END;
```
RunArray contents (private).

```
RunArray = ARRAY [0..100000] OF RunArrayChunk;
```
For casting use (private).

```
PRunArray = ^RunArray;
```
Pointer to run array (private).

```
HRunArray = ^PRunArray;
```
A handle to a run array (private).

Template Types

```
GridViewTemplate = PACKED RECORD
     numOfRows:           INTEGER;
     numOfCols:           INTEGER;
     rowHeight:           INTEGER;
     colWidth:            INTEGER;
     rowInset:            INTEGER;
     colInset:            INTEGER;
     adornRows:           BOOLEAN;
     adornCols:           BOOLEAN;
     singleSelection:     BOOLEAN;
     filler:              0..8191;
END;

GridViewTemplatePtr = ^GridViewTemplate;

TextGridViewTemplate = PACKED RECORD
     itsFontFace:         Style;
     itsFontSize:         INTEGER;
     itsFontColor:        RGBColor;
     itsFontName:         Str255;
END;

TextGridViewTemplatePtr = ^TextGridViewTemplate;
```

Constants

Template identifiers for views defined in this unit:

```
kGridView =          'TGridView';
kTextGridView =      'TTextGridView';
kTextListView =      'TTextListView';
```

For setting column/row width/height in only one row/column:

```
kOneRow =            1;
kOneCol =            1;
```

Select Command Identifiers:

```
cCellSelect =        1;
cRowSelect =         2;
cColumnSelect =      3;
cVertexSelect =      4;
```

Booleans for SetSelection:

```
kExtend =            TRUE;
kDontExtend =        FALSE;
kHighlight =         TRUE;
kDontHighlight =     FALSE;
kSelect =            TRUE;
kDeSelect =          FALSE;
```

Booleans for CreateHighlightRgn:

```
kWholeRect =         TRUE;
kNotWholeRect =      FALSE;
```

Fields

`fnumOfRows:`	`integer;`	Number of rows.
`fnumOfCols:`	`integer;`	Number of columns.
`fAdornRows:`	`boolean;`	Adorn rows?
`fAdornCols:`	`boolean;`	Adorn columns?
`fRowInset:`	`integer;`	Space between cells in a row.
`fColInset:`	`integer;`	Space between cells in a column.
`fSingleSelection:`	`boolean;`	Allows only a single selection.
`fAllowDimSelection:`	`boolean;`	Allows dimming of the selection.
`fSelections:`	`RgnHandle;`	Cells currently selected (private).
`fRowHeights:`	`HRunArray;`	A handle to an integer array (private).
`fColWidths:`	`HRunArray;`	A handle to an integer array (private).
`fMaxHorizontal:`	`longint;`	Total width in pixels (private).
`fMaxVertical:`	`longint;`	Total height in pixels (private).
`fWholeRect:`	`boolean;`	Is the selection a rectangle (private).
`fLastWholeRect:`	`boolean;`	Was the old selection a rectangle (private).

```
fHLRegion:          RgnHandle;      Cells currently highlighted (private).
fnumOfRowChunks:    integer;        Keeps track of chunks in run array (private).
fnumOfColChunks:    integer;        Keeps track of chunks in run array (private).
fTmpRgn:            RgnHandle;      For temporary region use (private).
fTmpSelRgn:         RgnHandle;      For temporary selection region use (private).
fTmpHLRgn:          RgnHandle;      For temporary highlighting use (private).
fLastRow:           integer;        Used for run array caching (private).
fLastRowChunkNum:   integer;        Used for run array caching (private).
fLasRowtTotal:      longint;        Used for run array caching (private).
fLastRowIndex:      integer;        Used for run array caching (private).
fLastCol:           integer;        Used for run array caching (private).
fLastColChunkNum:   integer;        Used for run array caching (private).
fLastColTotal:      longint;        Used for run array caching (private).
fLasColIndex:       integer;        Used for run array caching (private).
```

Initialization and Free Methods:

```
PROCEDURE TGridView.IGridView ( itsDocument:      TDocument;
                                itsSuperView:     TView;
                                itsLocation:      VPoint;
                                numOfRows:        integer;
                                numOfCols:        integer;
                                rowHeight:        integer;
                                colWidth:         integer;
                                adornRows:        boolean;
                                adornCols:        boolean;
                                rowInset:         integer;
                                colInset:         integer;
                                singleSelection:  boolean );
```

Initializes the `TGridView` object.

```
PROCEDURE TGridView.IRes (itsDocument: TDocument; itsSuperView: TView;
    VAR  itsParams: Ptr); OVERRIDE;
```

This method initializes the `TGridView` object using a view template.

```
PROCEDURE TGridView.WRes (theResource: ViewRsrcHndl; VAR itsParams: Ptr);
    OVERRIDE;
```

This method writes this object out as a `'view'` resource.

```
PROCEDURE TGridView.WriteRes (theResource: ViewRsrcHndl; VAR itsParams: Ptr);
    OVERRIDE;
```

This method sets up the type and signature of this object and calls `WRes`.

```
PROCEDURE TGridView.Free; OVERRIDE;
```

This method frees a `TGridView` object.

Inherited Methods

```
PROCEDURE TGridView.CalcMinSize (VAR minSize: VPoint); OVERRIDE;
```

This method sets the extent of the view.

```
PROCEDURE TGridView.DoHighlightSelection (fromHL, toHL: HLState); OVERRIDE;
```

This method does highlighting. This method is called automatically when highlighting needs to be done.

```
FUNCTION  TGridView.DoMouseCommand (VAR theMouse: Point; VAR info: EventInfo;
    VAR hysteresis: Point) : TCommand; OVERRIDE;
```

This method processes a mouse command in the view. This method calls `DoCellClick`.

```
PROCEDURE TGridView.Draw (area: Rect); OVERRIDE;
```

This method calls `DrawRangeOfCells` and `AdornRow` or `AdornCol` as appropriate.

Methods to Override

```
PROCEDURE TGridView.AdornCol (aCol: INTEGER; area: Rect);
```

This method is called once for each cell that needs to be drawn or redrawn.

```
PROCEDURE TGridView.AdornRow (aRow: INTEGER; area: Rect);
```

This method is called once for each cell that needs to be drawn or redrawn.

```
FUNCTION  TGridView.CanSelectCell (aCell: GridCell) : BOOLEAN;
```

This method tests to see if the cell is selectable.

```
FUNCTION  TGridView.CellExists (aCell: GridCell) : BOOLEAN;
```

By default this method always returns TRUE. Cells that do not exists will be skipped when drawing is done.

```
PROCEDURE TGridView.DoHilite (startCell, stopCell: GridCell;
    fromHL, toHL: HLState);
```

This method actually does the highlighting (by inverting the rect). You can override it to get different highlighting behavior.

```
PROCEDURE TGridView.DrawRangeOfCells (startCell, stopCell: GridCell;
    aQDRect: Rect);
```

This method calls `DrawCell` for each cell in need of re-drawing.

```
PROCEDURE TGridView.DrawCell (aCell: GridCell; aQDRect: Rect);
```

This method draws a single cell.

General Methods

```
PROCEDURE TGridView.AllCellsDo (PROCEDURE DoToCell (aCell: GridCell));
```

This method executes `DoToCell` for every cell in the view.

```
PROCEDURE TGridView.CellToVRect (aCell: GridCell; VAR aRect: VRect);
```

This method calculates the bounding rectangle for a given cell (includes row and column insets).

```
PROCEDURE TGridView.ColToVRect (aCol: INTEGER; numOfCols: INTEGER;
    VAR aRect: VRect);
```

This method calculates the bounding rectangle for a given column (includes row and column insets).

```
PROCEDURE TGridView.RowToVRect (aRow: INTEGER; numOfRows: INTEGER;
    VAR aRect: VRect);
```

This method calculates the bounding rectangle for a given row (includes row and column insets).

```
PROCEDURE TGridView.CreateHighlightRgn (oldRgn, newRgn: RgnHandle; wholeRect:
    BOOLEAN);
```

This method creates a region containing all the cells in oldRgn. If wholeRect is TRUE, then a faster algorithm is used to compute the new region. Setting wholeRect to FALSE will work for all cases, but will be unnecessarily slow for the whole rectangle case.

```
PROCEDURE TGridView.DelColAt (aCol: INTEGER; numOfCols: INTEGER);
```

This method deletes `numOfCols` columns starting at `aCol`.

```
PROCEDURE TGridView.DelRowAt (aRow: INTEGER; numOfRows: INTEGER);
```

This method deletes `numOfRows` rows starting at `aRow`.

```
PROCEDURE TGridView.DelColFirst (numOfCols: INTEGER);
```

This method deletes `numOfCols` columns starting with the first column.

```
PROCEDURE TGridView.DelRowFirst (numOfRows: INTEGER);
```

This method deletes `numOfRows` rows starting with the first row.

```
PROCEDURE TGridView.DelColLast (numOfCols: INTEGER);
```

This method deletes the last `numOfCols` columns.

```
PROCEDURE TGridView.DelRowLast (numOfRows: INTEGER);
```

This method deletes the last `numOfRows` rows.

```
PROCEDURE TGridView.EachCellDo (startCell, stopCell: GridCell;
    PROCEDURE DoToCell (aCell: GridCell));
```

For each cell in the rectangle described by `startCell` and `stopCell`, this method executes the procedure `DoToCell`.

```
PROCEDURE TGridView.EachInRgn (aRgn: RgnHandle;
    PROCEDURE DoToCell (aCell: GridCell));
```

For each cell in the given region, this method executes the procedure `DoToCell`.

```
PROCEDURE TGridView.EachSelectedCellDo (PROCEDURE DoToCell (aCell: GridCell));
```

For each cell in the selected region, this method executes `DoToCell`.

```
FUNCTION  TGridView.FindRowChunk (aRow: INTEGER; VAR aChunkNum: INTEGER;
    VAR  theTotal: LONGINT; VAR indexInChunk: INTEGER) : BOOLEAN;
```

This method looks through the run array containing the row heights, determines if the specified row exists, and returns information about where it is (or should be).

```
FUNCTION  TGridView.FindColChunk (aCol: INTEGER; VAR aChunkNum: INTEGER;
    VAR  theTotal: LONGINT; VAR indexInChunk: INTEGER) : BOOLEAN;
```

This method looks through the run array containing column widths and determines if the specified column exists and returns information about where it is (or should be).

```
FUNCTION  TGridView.FirstSelectedCell : GridCell;
```

This method returns the top, left cell in the selection region (if any).

```
FUNCTION  TGridView.GetColWidth (aCol: INTEGER) : INTEGER;
```

This method returns the column width.

```
FUNCTION  TGridView.GetRowHeight (aRow: INTEGER) : INTEGER;
```

This method returns the row height.

```
FUNCTION TGridView.IdentifyPoint (theQDPoint: Point;
    VAR aRow, aCol: INTEGER): GridViewPart;
```

Determine if the given point is in a cell, row, column, or vertex.

```
PROCEDURE TGridView.InsColBefore (aCol: INTEGER; numOfCols: INTEGER;
    aWidth:  INTEGER);
```

This method inserts `numOfCols` columns before the given column.

```
PROCEDURE TGridView.InsRowBefore (aRow: INTEGER; numOfRows: INTEGER;
    aHeight: INTEGER);
```

This method inserts `numOfRows` rows before the given row.

```
PROCEDURE TGridView.InsColFirst (numOfCols: INTEGER; aWidth: INTEGER);
```

This method inserts `numofCols` columns before the first column.

```
PROCEDURE TGridView.InsRowFirst (numOfRows: INTEGER; aHeight: INTEGER);
```

This method inserts `numofRows` rows before the first row.

```
PROCEDURE TGridView.InsColLast (numOfCols: INTEGER; aWidth: INTEGER);
```

This method inserts `numOfCols` columns after the last column.

```
PROCEDURE TGridView.InsRowLast (numOfRows: INTEGER; aHeight: INTEGER);
```

This method inserts `numOfRows` rows after the last row.

```
PROCEDURE TGridView.InvalidateCell (aCell: GridCell);
```

This method causes a cell to be marked invalid (that is, in need of redrawing).

```
PROCEDURE TGridView.InvalidateSelection;
```

This method causes the rectangle bounding the selections to be marked invalid (that is, in need of redrawing).

```
FUNCTION  TGridView.IsCellSelected (aCell: GridCell) : BOOLEAN;
```

This method checks if the specified cell is currently selected.

```
FUNCTION  TGridView.LastCellSelected : GridCell;
```

This method returns the bottom, right cell in the selection region (if any).

```
FUNCTION TGridView.SameCell (aCell, bCell: GridCell): BOOLEAN;
```

This method checks if the cells are the same.

```
PROCEDURE TGridView.ScrollSelectionIntoView (isVisible: BOOLEAN);
```

This method scrolls the current selection into view.

```
PROCEDURE TGridView.SetColWidth (aCol: INTEGER; numOfCols: INTEGER;
    aWidth:  INTEGER);
```

This method sets the stated column width.

```
PROCEDURE TGridView.SetRowHeight (aRow: INTEGER; numOfRows: INTEGER;
    aHeight:  INTEGER);
```

This method sets the stated row height.

```
PROCEDURE TGridView.SelectCell (theCell: GridCell;
    extend, highlight, select: BOOLEAN);
```

This method sets the current selection to the specified cell, by setting up a region and then calling `SetSelection`. `select` contols whether the Region is selected or de-selected.

```
PROCEDURE TGridView.SetEmptySelection (highlight: BOOLEAN);
```

This method sets the current selection to empty (nothing). If `Highlight` is `TRUE`, then the old selection is de-highlighted as well.

```
PROCEDURE TGridView.SetSelection (theRegion: RgnHandle;
    extend, highlight, select: BOOLEAN);
```

This method sets the current selection to the specified region. If `extend` is `TRUE`, then the cells in `theRegion` are added to that already selected. If `highlight` is `TRUE`, then the selection is highlighted as well. `select` contols whether the cells in `theRegion` are selected or de-selected.

```
PROCEDURE TGridView.SetSelRect (theTop, theLeft, theBottom, theRight: INTEGER;
    extend, highlight, select: BOOLEAN);
```

This method sets the current selections to the specified rectangle. If `extend` is `TRUE`, then `theRegion` is added to that already selected. If `highlight` is `TRUE`, then `theRegion` is highlighted as well. Select contols whether theRegion is selected or de-selected.

```
PROCEDURE TGridView.SetSingleSelection (theSetting: BOOLEAN);
```

If `theSetting` is `TRUE`, then only one item/cell can be selected at a time.

```
PROCEDURE TGridView.VPointToCell (aPoint: VPoint; VAR aCell: GridCell);
```

This method determines which cell is located at a given point.

```
PROCEDURE TGridView.Fields ( PROCEDURE DoToField  (fieldName: Str255;
    fieldAddr: Ptr; fieldType: INTEGER)); OVERRIDE;
```

This method provides debugging support for the inspector.

TTextGridView

Fields

`fTextStyle:`	`TextStyle;`	The text style: color, size, and so on.
`fLineHeight:`	`INTEGER;`	Height of each item, including ascent
`fLineAscent:`	`INTEGER;`	Position of baseline relative to top of the line.

Initialization and Free Methods

```
PROCEDURE TTextGridView.ITextGridView (
```

`itsDocument:`	`TDocument;`	Its document.
`itsSuperView:`	`TView;`	Its parent view .
`itsLocation:`	`VPoint;`	Top, Left in parent's coordinates.
`numOfRows:`	`INTEGER;`	Number of rows.
`numOfCols:`	`INTEGER;`	Number of columns.
`colWidth:`	`INTEGER;`	Width of items in the columns.
`adornRows:`	`BOOLEAN;`	Adornment for Rows?
`adornCols:`	`BOOLEAN;`	Adornment for Columns?
`rowInset:`	`INTEGER;`	Horizontal space between cells.
`colInset:`	`INTEGER;`	Vertical space between cell .
`singleSelection:`	`BOOLEAN;`	Allow only a single cell to be selected.
`itsTextStyle:`	`TextStyle);`	The text style.

This method initializes the `TTextGridView` object.

```
PROCEDURE TTextGridView.IRes (itsDocument: TDocument; itsSuperView: TView;
    VAR itsParams: Ptr); OVERRIDE;
```

This method initializes the view template for `TTextGridView`.

```
PROCEDURE TTextGridView.WRes (theResource: ViewRsrcHndl; VAR itsParams: Ptr);
    OVERRIDE;
```

This method writes the object out as a `'view'` resource.

```
PROCEDURE TTextGridView.WriteRes (theResource: ViewRsrcHndl;
    VAR itsParams: Ptr); OVERRIDE;
```

This method sets up the type and signature of this object and calls `WRes`.

Methods to Override

```
PROCEDURE TTextGridView.GetText (aCell: GridCell; VAR aString: Str255);
```

Given a cell, this method returns the text to be displayed. *This method must be overridden.*

General Methods

```
PROCEDURE TTextGridView.DrawCell (aCell: GridCell; aQDRect: Rect); OVERRIDE;
```

This method calls `GetText` and then `INHERITED DrawCell`.

```
FUNCTION TTextGridView.Focus : BOOLEAN; OVERRIDE;
```

This method calls `INHERITED Focus` and then `SetUpFont`.

```
PROCEDURE TTextGridView.SetUpFont;
```

Sets the font characteristics for this view.

```
PROCEDURE TTextGridView.SetPen;
```

This method sets the font characteristics of the current port to `fTextStyle`. It is called at the beginning of the `Draw` method.

```
PROCEDURE TTextGridView.Fields ( PROCEDURE DoToField  (fieldName: Str255;
    fieldAddr: Ptr; fieldType: INTEGER)); OVERRIDE;
```

This method provides debugging support for the inspector.

TTextListView

Initialization and Free Methods

```
PROCEDURE TTextListView.ITextListView (
    itsDocument:        TDocument;          Its document.
    itsSuperView:       TView;              Its parent view.
    itsLocation:        VPoint;             Top, left in parent's coordinates.
    numOfItems:         INTEGER;            Number of items in the list.
    adornRows:          BOOLEAN;            Adornment for rows?
    adornCols:          BOOLEAN;            Adornment for columns?
    rowInset:           INTEGER;            Horizontal space between cells.
    colInset:           INTEGER;            Vertical  space between cells.
    singleSelection:    BOOLEAN;            Allow only a single item to be selected.
    itsTextStyle:       TextStyle);         The text style.
```

This method initializes the `TTextListView` object.

```
PROCEDURE TTextListView.WriteRes (theResource: ViewRsrcHndl; VAR itsParams:
    Ptr); OVERRIDE;
```

This method sets up the type and signature of this object and calls `WRes`.

Methods To OVERRIDE

```
PROCEDURE TTextListView.GetItemText (anItem: INTEGER; VAR aString: Str255);
```

Given an item number, this method fills in the text to be displayed. *This method must be overridden.*

```
FUNCTION  TTextListView.CanSelectItem  (anItem: INTEGER) : BOOLEAN;
```

This method checks if the specified item can be selected. The default always returns `TRUE`.

General Methods

```
PROCEDURE TTextListView.AllItemsDo (PROCEDURE DoToItem (anItem: INTEGER));
```

For all the items in the view, this method performs `DoToItem`.

```
FUNCTION  TTextListView.CanSelectCell  (aCell: GridCell) : BOOLEAN; OVERRIDE;
```

This method simply calls `CanSelectItem` with the vertical cell component.

```
PROCEDURE TTextListView.DelItemAt (anItem: INTEGER; numOfItems: INTEGER);
```

This method deletes `numOfItems` items starting at the given item.

```
PROCEDURE TTextListView.DelItemFirst (numOfItems: INTEGER);
```

This method deletes the first `numOfItems` items.

```
PROCEDURE TTextListView.DelItemLast (numOfItems: INTEGER);
```

This method deletes the last `numOfItems` items.

```
PROCEDURE TTextListView.EachItemDo (start, stop: INTEGER;
    PROCEDURE DoToItem (anItem: INTEGER));
```

For each item in the specified range, this method executes `DoToItem`.

```
PROCEDURE TTextListView.EachSelectedItemDo (PROCEDURE DoToItem (anItem:
    INTEGER));
```

For each item selected, this method executes `DoToItem`.

```
FUNCTION  TTextListView.GetItemHeight (anItem: INTEGER) : INTEGER;
```

This method returns the item's height.

```
FUNCTION  TTextListView.GetItemWidth: INTEGER;
```

This method returns the item's width.

```
PROCEDURE TTextListView.GetText (aCell: GridCell; VAR aString: Str255);
    OVERRIDE;
```

Given a cell number, this method returns the text, by calling `GetItemText`.

```
PROCEDURE TTextListView.InsItemBefore (anItem: INTEGER; numOfItems: INTEGER);
```

This method inserts `numOfItems` items before the given item.

```
PROCEDURE TTextListView.InsItemFirst (numOfItems: INTEGER);
```

This method inserts `numOfItems` items before the first item.

```
PROCEDURE TTextListView.InsItemLast (numOfItems: INTEGER);
```

This method inserts `numOfItems` items after the last item.

```
FUNCTION  TTextListView.IsItemSelected (anItem: INTEGER) : BOOLEAN;
```

This method checks if the specified item is currently selected.

```
PROCEDURE TTextListView.Resize (width, height: VCoordinate;
    invalidate: BOOLEAN); OVERRIDE;
```

This method resizes the view.

```
PROCEDURE TTextListView.SelectCell (theCell: GridCell;
    extend, highlight, select: BOOLEAN); OVERRIDE;
```

This method calls `SelectItem` with the appropriate item number.

```
PROCEDURE TTextListView.SelectItem (anItem: INTEGER;
    extend, highlight, select: BOOLEAN);
```

This method selects the given item. If `extend` is `true`, then the item is added to the current selection. Otherwise, the current selection is deselected. If `highlight` is `true` then the selected item is highlighted. `select` contols whether the item is selected or de-selected.

```
PROCEDURE TTextListView.SetItemHeight (anItem: INTEGER; numOfItems: INTEGER;
    aHeight: INTEGER);
```

This item sets the item height. It will change the height only of the `numOfItems` items starting at the

specified item.

```
PROCEDURE TTextListView.SetItemWidth (aWidth: INTEGER);
```

This method sets the item width to be different than that determined by `TGridView`. This routine will change the width of the list.

```
PROCEDURE TTextListView.Fields (PROCEDURE DoToField  (fieldName: Str255;
    fieldAddr: Ptr; fieldType: INTEGER)); OVERRIDE;
```

This method provides debugging support for the inspector.

TGridSelectCommand

Fields

`fGridView:`	`TGridView;`	The associated gridview.
`fShiftKey:`	`BOOLEAN;`	Shift Key is down.
`fCmdKey:`	`BOOLEAN;`	Command key is down.
fOldRgn:	RgnHandle;	Old highlighted region.
fNewRgn:	RgnHandle;	New highlighted region.

Methods

```
PROCEDURE TGridSelectCommand.IGridSelectCommand ( itsCmdNumber: CmdNumber;
    itsView: TGridView; theShiftKey, theCmdKey: BOOLEAN);
```

This method initializes the command object.

```
PROCEDURE TGridSelectCommand.Free; OVERRIDE;
```

This method frees the temporary regions.

```
PROCEDURE TGridSelectCommand.TrackFeedback (anchorPoint, nextPoint: VPoint;
    turnItOn, mouseDidMove: BOOLEAN); OVERRIDE;
```

This method provides feedback and tracks the mouse while the button is down.

```
PROCEDURE TGridSelectCommand.Fields (PROCEDURE DoToField  (fieldName: Str255;
    fieldAddr: Ptr; fieldType: INTEGER)); OVERRIDE;
```

This method provides debugging support for the inspector.

TCellSelectCommand

Fields

`fOldCell:`	`GridCell;`	The previous cell.
`fNewMouseDown:`	`BOOLEAN;`	Determine if this is a new mouse-down event.
`fKeepSelecting:`	`BOOLEAN;`	Keep selecting new cells (via Command Key)?

Methods

```
PROCEDURE TCellSelectCommand.ICellSelectCommand (itsView: TGridView;
    aCell: GridCell; theShiftKey, theCmdKey: BOOLEAN);
```

This method initializes the command object.

```
PROCEDURE TCellSelectCommand.HandleShiftKeyDown (aView: TGridView;
    aCell: GridCell);
```

This method provides support for `TrackFeedback` and `TrackMouse` when the shift key and the mouse button are down.

```
PROCEDURE TCellSelectCommand.TrackFeedback (anchorPoint, nextPoint: VPoint;
    turnItOn, mouseDidMove: BOOLEAN); OVERRIDE;
```

This method provides feedback and tracks the mouse while the button is down.

```
FUNCTION TCellSelectCommand.TrackMouse (aTrackPhase: TrackPhase;
    VAR anchorPoint, previousPoint, nextPoint: VPoint;
    mouseDidMove: BOOLEAN): TCommand; OVERRIDE;
```

This method handles the mouse-up event.

```
PROCEDURE TCellSelectCommand.Fields (PROCEDURE DoToField  (fieldName: Str255;
    fieldAddr: Ptr; fieldType: INTEGER)); OVERRIDE;
```

This method provides debugging support for the inspector.

TRowSelectCommand

Fields

```
fTheRow:        INTEGER;
```

The row.

Methods

```
PROCEDURE TRowSelectCommand.IRowSelectCommand (itsView: TGridView;
    aRow: INTEGER; theShiftKey, theCmdKey: BOOLEAN);
```

This method initializes the command object.

```
PROCEDURE TRowSelectCommand.Fields (PROCEDURE DoToField  (fieldName: Str255;
    fieldAddr: Ptr; fieldType: INTEGER)); OVERRIDE;
```

This method provides debugging support for the inspector.

TColSelectCommand

Fields

```
fTheColumn:     INTEGER;
```

The column.

Methods

```
PROCEDURE TColSelectCommand.IColSelectCommand (itsView: TGridView;
    aColumn: INTEGER; theShiftKey, theCmdKey: BOOLEAN);
```

This method initializes the command object.

```
PROCEDURE TColSelectCommand.Fields (PROCEDURE DoToField  (fieldName: Str255;
    fieldAddr: Ptr; fieldType: INTEGER)); OVERRIDE;
```

This method provides debugging support for the inspector.

TVertexSelectCommand

Fields

`fTheColumn:`	`INTEGER;`	The column associated with the vertex
`fTheRow:`	`INTEGER;`	The row associated with the vertex

Methods

```
PROCEDURE TVertexSelectCommand.IVertexSelectCommand (itsView: TGridView;
    aRow, aColumn: INTEGER; theShiftKey, theCmdKey: BOOLEAN);
```

This method initializes the command object.

```
PROCEDURE TVertexSelectCommand.Fields (PROCEDURE DoToField (fieldName: Str255;
    fieldAddr: Ptr; fieldType: INTEGER)); OVERRIDE;
```

This method provides debugging support for the inspector.