User Manual for wxGrid Version 1.0

Julian Smart

May 13th 1997

Contents

Copyright notice

1. Introduction

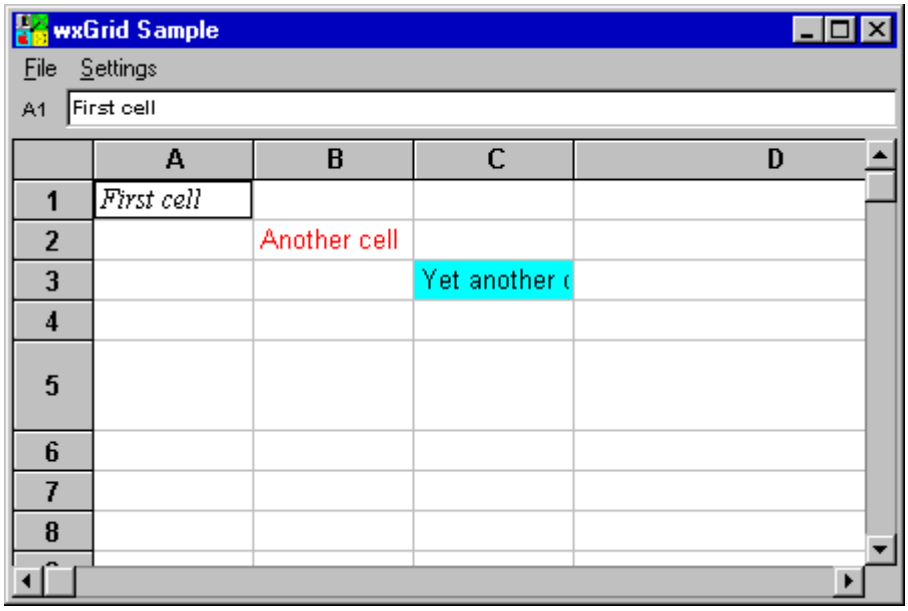
The wxGrid class is a window designed for displaying data in tabular format. Possible uses include:

- Displaying database tables;
- building spreadsheet applications;
- displaying files and their attributes;
- use as a more sophisticated listbox where different fonts and colours are required.

This manual currently describes the version of wxGrid that operates under Windows and Motif, implementing using generic wxWindows code. Under Motif, wxGrid can also be compiled using the public domain Xbae matrix widget, included in the wxGrid distribution. However, some work still needs to be done to provide full wxGrid functionality.

1.1. The appearance and behaviour of a wxGrid

The following screenshot shows the initial appearance of the sample wxGrid application.



The wxGrid class is a panel that provides a text editing area, and a grid with scrollbars. The grid has horizontal and vertical label areas whose colours may be changed independently from the cell area. The text editing area, and the label areas, may be switched off if desired.

The user navigates the grid using the mouse to click on cells and scroll around the virtual grid area (no keyboard navigation is possible as yet). If the edit control is enabled, it always has the focus for the currently selected cell and the user can type into it. The text in the edit control will be reflected in the currently selected cell.

If the row and column label areas are enabled, the user can drag on the label divisions to resize a row or column.

The sample application allows the user to change various aspects of the grid using the wxGrid

API. These include:

- Changing the background and foreground colour of labels and cells;
- toggling row and column label areas on and off independently;
- toggling the edit control on and off;
- toggling the light grey cell dividers on and off;
- changing cell alignment.

There are various other aspects that can be controlled via the API, including changing individual cell font and colour properties.

2. Files

The wxGrid class library comprises the following files.

UNIX files:

- caption.c
- clip.c
- matrix.c
- caption.h
- captionp.h
- clip.h
- clipp.h
- matrix.h
- matrixp.h
- wxgridm.cc
- wxgridm.h

Generic files:

- wxgridg.h (generic implementation)
- wxgridg.cc
- wxgrid.h (includes wxgridw.h or wxgridm.h as necessary)
- test.h (test application)
- test.cc

3. Implementation issues

This is an assortment of wxGrid implementation issues.

3.1. wxMotifGrid

wxMotifGrid version needs to be implemented in terms the Xbae matrix widget. A start has been made: see wxgridm.h, wxgridm.cc. Probably not all of the API of wxGenericGrid can be emulated using Xbae.

3.2. wxGenericGrid

3.2.1. To do

- Keyboard functionality (see Problems).
- Selection of multiple cells, rows, columns by dragging or clicking on row/column labels.
- Interception of left, right, middle mouse clicks by application.
- Optimization of painting code: for instance, all columns and rows are painted from the cursor position, although some may be hidden. Under Windows, ::ScrollWindow could be used to help reduce flicker.

3.2.2. Problems

It will be tricky to make wxGenericGrid responsive to cursor key movement, because at present the wxText item always has the focus (in editable mode) and absorbs cursor key presses.

One possible solution is *not* to have the focus always set on the wxText item. When the user clicks on a cell, the focus is on the cell. When the user presses Return or double-clicks, the focus is set to the wxItem and text may be typed in. Immediately after the user has single-clicked on a cell, cursor keys can be used to navigate around the grid.

In non-editable mode, focus is always on the current cell, and therefore all cursor movements are immediately interpreted as navigation requests.

3.3. Other platforms

Other platforms, such as XView, should try to use wxGenericGrid unless free platform-specific grid widgets are available. However, the XView version of wxWindows has problems with user painting of panels so this needs investigating.

4. Alphabetical class reference

See also *wxGrid classes overview* (page 17)

4.1. wxGrid: wxPanel

See also *wxGrid classes overview* (page 17)

wxGrid::wxGrid

void wxGrid(**wxWindow ***_parent_, **int** _x_, **int** _y_, **int** _width_, **int** _height_, **long** _style=0_, **char ***_name="grid"_)

Constructor. Before using a wxGrid object, you must call CreateGrid to set up the required rows and columns.

wxGrid::AdjustScrollbars

void AdjustScrollbars(**void**)

Call this function whenever a change has been made via the API that might alter the scrollbar characteristics: particularly when adding or deleting rows, or changing row or column dimensions. For example, removing rows might make it unnecessary to show the vertical scrollbar.

wxGrid::AppendCols

Bool AppendCols(**int** _n=1_, **Bool** _updateLabels=TRUE_)

Appends *n* columns to the grid. If *updateLabels* is TRUE, the function OnChangeLabels is called to give the application the opportunity to relabel.

wxGrid::AppendRows

Bool AppendRows(**int** _n=1_, **Bool** _updateLabels=TRUE_)

Appends *n* rows to the grid. If *updateLabels* is TRUE, the function OnChangeLabels is called to give the application the opportunity to relabel.

wxGrid::BeginBatch

void BeginBatch(**void**)

Start a BeginBatch/EndBatch pair between which, calls to SetCellValue or SetCellBitmap will not cause a refresh. This allows you to speed up some operations (for example, setting several hundred cell values). You can nest, but not overlap, these two functions.

See also *EndBatch* (page 6), *GetBatchCount* (page 6).

wxGrid::CellHitTest

Bool CellHitTest(**int** *x*, **int** *y*, **int ***row*, **int ***col*)

Returns TRUE if the x, y panel position coincides with a cell. If so, *row* and *col* are returned.

wxGrid::CreateGrid

Bool CreateGrid(**int** *rows*, **int** *cols*, **wxString ****cellValues=NULL*, **short ***widths=NULL*, **short** *defaultWidth=wxGRID_DEFAULT_CELL_WIDTH*, **short** *defaultHeight=wxGRID_DEFAULT_CELL_HEIGHT*)

Creates a grid *rows* high and *cols* wide. You can optionally specify an array of initial values and widths, and/or default cell width and height.

Call this function after creating the wxGrid object.

wxGrid::CurrentCellVisible

Bool CurrentCellVisible(**void**)

Returns TRUE if the currently selected cell is visible, FALSE otherwise.

wxGrid::DeleteCols

Bool DeleteCols(**int** *pos=0*, **int** *n=1*, **Bool** *updateLabels=TRUE*)

Deletes *n* columns from the grid at position *pos*. If *updateLabels* is TRUE, the function OnChangeLabels is called to give the application the opportunity to relabel.

wxGrid::DeleteRows

Bool DeleteRows(**int** *pos=0*, **int** *n=1*, **Bool** *updateLabels=TRUE*)

Deletes *n* rows from the grid at position *pos*. If *updateLabels* is TRUE, the function OnChangeLabels is called to give the application the opportunity to relabel.

wxGrid::EndBatch

void EndBatch(**void**)

End a BeginBatch/EndBatch pair between which, calls to SetCellValue or SetCellBitmap will not cause a refresh. This allows you to speed up some operations (for example, setting several hundred cell values). You can nest, but not overlap, these two functions.

See also *BeginBatch* (page 5), *GetBatchCount* (page 6).

wxGrid::GetBatchCount

void GetBatchCount(**void**)

Return the level of batch nesting. This is initially zero, and will be incremented every time BeginBatch is called, and decremented when EndBatch is called. When the batch count is more zero, some functions (such as SetCellValue and SetCellBitmap) will not refresh the cell.

See also *BeginBatch* (page 5), *EndBatch* (page 6).

wxGrid::GetCell

wxGridCell * GetCell(**int** *row*, **int** *col*)

Returns the grid cell object associated with this position.

wxGenericGrid implementation only.

wxGrid::GetCellAlignment

int GetCellAlignment(**int** *row*, **int** *col*)

int GetCellAlignment(**void**)

Sets the text alignment for the cell at the given position, or the global alignment value. The return value is wxLEFT, wxRIGHT or wxCENTRE.

wxGrid::GetCellBackgroundColour

wxColour& GetCellBackgroundColour(**int** *row*, **int** *col*)

wxColour& GetCellBackgroundColour(**void**)

Gets the background colour for the cell at the given position, or the global background colour.

wxGrid::GetCells

wxGridCell * GetCells**(**void**)

Returns the array of grid cell object associated with this wxGrid.

wxGenericGrid implementation only.

wxGrid::GetCellTextColour

wxColour& GetCellTextColour(**int** *row*, **int** *col*)

wxColour& GetCellTextColour(**void**)

Gets the text colour for the cell at the given position, or the global text colour.

wxGrid::GetCellTextFont

wxFont * GetCellTextFont(**int** *row*, **int** *col*)

wxFont * GetCellTextFont(**void**)

Gets the text font for the cell at the given position, or the global text font.

wxGrid::GetCellValue

wxString& GetCellValue(**int** *row*, **int** *col*)

Returns the cell value at the given position.

wxGrid::GetCols

int GetCols(**void**)

Returns the number of columns in the grid.

wxGrid::GetColumnWidth

int GetColumnWidth(**int** *col*)

Gets the width in pixels for column *col*.

wxGrid::GetCurrentRect

wxRectangle * GetCurrentRect(**void**)

Returns a pointer to the rectangle enclosing the currently selected cell. Do not delete this pointer.

wxGrid::GetCursorColumn

int GetCursorColumn(**void**)

Returns the column position of the currently selected cell.

wxGrid::GetCursorRow

int GetCursorRow(**void**)

Returns the row position of the currently selected cell.

wxGrid::GetEditable

Bool GetEditable(**void**)

Returns TRUE if the grid cells can be edited.

wxGrid::GetHorizScrollBar

wxScrollBar * GetHorizScrollBar(**void**)

Returns a pointer to the horizontal scrollbar.

wxGrid::GetLabelAlignment

int GetLabelAlignment(**int** *orientation*)

Gets the row or column label alignment. *orientation* should be wxHORIZONTAL to specify column label, wxVERTICAL to specify row label. *alignment* should be wxCENTRE, wxLEFT or wxRIGHT.

wxGrid::GetLabelBackgroundColour

wxColour& GetLabelBackgroundColour(**void**)

Gets a row and column label text colour.

wxGrid::GetLabelSize

int GetLabelSize(**int** *orientation*)

Gets the row label height, or column label width, in pixels. *orientation* should be wxHORIZONTAL to specify column label, wxVERTICAL to specify row label.

wxGrid::GetLabelTextColour

wxColour& GetLabelTextColour(**void**)

Gets a row and column label text colour.

wxGrid::GetLabelTextFont

wxFont * GetLabelTextFont(**void**)

Gets the font to be used for the row and column labels.

wxGrid::GetLabelValue

wxString& GetLabelValue(**int** *orientation*, **int** *pos*)

Gets a row or column label value. *orientation* should be wxHORIZONTAL to specify column label,

wxVERTICAL to specify row label. *pos* is the label position.

wxGrid::GetRowHeight

int GetRowHeight(**int** *row*)

Gets the height in pixels for row *row*.

wxGrid::GetRows

int GetRows(**void**)

Returns the number of rows in the grid.

wxGrid::GetScrollPosX

int GetScrollPosX(**void**)

Returns the column scroll position.

wxGrid::GetScrollPosX

int GetScrollPosY(**void**)

Returns the row scroll position.

wxGrid::GetTextItem

wxText * GetTextItem(**void**)

Returns a pointer to the text item used for entering text into a cell.

wxGrid::GetVertScrollBar

wxScrollBar * GetVertScrollBar(**void**)

Returns a pointer to the vertical scrollbar.

wxGrid::InsertCols

Bool InsertCols(**int** *pos=0*, **int** *n=1*, **Bool** *updateLabels=TRUE*)

Inserts *n* number of columns before position *pos*. If *updateLabels* is TRUE, the function OnChangeLabels is called to give the application the opportunity to relabel.

wxGrid::InsertRows

Bool InsertRows(**int** *pos=0*, **int** *n=1*, **Bool** *updateLabels=TRUE*)

Inserts *n* number of rows before position *pos*. If *updateLabels* is TRUE, the function OnChangeLabels is called to give the application the opportunity to relabel.

wxGrid::OnActivate

void OnActivate(**Bool** *active*)

Sets the text item to have the focus. Call this function when the wxGrid window should have the focus, for example from wxFrame::OnActivate.

wxGrid::OnChangeLabels

void OnChangeLabels(**void**)

Called when rows and columns are created or deleted, to allow the application an opportunity to update the labels. By default, columns are labelled alphabetically, and rows numerically.

wxGrid::OnChangeSelectionLabel

void OnChangeSelectionLabel(**void**)

Called when a cell is selected, to allow the application an opportunity to update the selection label (the label of the wxText item used for entering cell text). By default, the cell column letter and row number are concatenated to form the selection label.

wxGrid::OnCreateCell

wxGridCell * OnCreateCell(**void**)

Override this virtual function if you want to replace the normal wxGridCell with a derived class.

wxGrid::OnCellLeftClick

void OnLeftClick(**int** *row*, **int** *col*, **int** *x*, **int** *y*, **Bool** *control*, **Bool** *shift*)

Virtual function called when the left button is depressed within a cell, just after OnSelectCell is called.

wxGrid::OnCellRightClick

void OnRightClick(**int** *row*, **int** *col*, **int** *x*, **int** *y*, **Bool** *control*, **Bool** *shift*)

Virtual function called when the right button is depressed within a cell, just after OnSelectCell is called.

wxGrid::OnLabelLeftClick

void OnLeftClick(**int** *row*, **int** *col*, **int** *x*, **int** *y*, **Bool** *control*, **Bool** *shift*)

Virtual function called when the left button is depressed within a label.

row will be *-1* if the click is in the top labels.

col will be *-1* if the click is in the left labels.

row and *col* will be *-1* if the click is in the upper left corner.

wxGrid::OnLabelRightClick

void OnRightClick(**int** *row*, **int** *col*, **int** *x*, **int** *y*, **Bool** *control*, **Bool** *shift*)

Virtual function called when the right button is depressed within a label.

row will be *-1* if the click is in the top labels.

col will be *-1* if the click is in the left labels.

row and *col* will be *-1* if the click is in the upper left corner.

wxGrid::OnSelectCell

void OnSelectCell(**int** *row*, **int** *col*)

Virtual function called when the user left-clicks on a cell.

wxGrid::OnSelectCellImplementation

void OnSelectCellImplementation(**wxDC ***dc*, **int** *row*, **int** *col*)

Virtual function called when the user left-clicks on a cell. If you override this function, call wxGrid::OnSelectCell to apply the default behaviour.

wxGrid::SetCellAlignment

void SetCellAlignment(**int** *alignment*, **int** *row*, **int** *col*)

void SetCellAlignment(**int** *alignment*)

Sets the text alignment for the cell at the given position, or for the whole grid. *alignment* may be wxLEFT, wxRIGHT or wxCENTRE.

wxGrid::SetCellBackgroundColour

void SetCellBackgroundColour(**const wxColour&** *colour*, **int** *row*, **int** *col*)

void SetCellBackgroundColour(**const wxColour&** *colour*)

Sets the background colour for the cell at the given position, or for the whole grid.

wxGrid::SetCellTextColour

void SetCellTextColour(**const wxColour&** *colour*, **int** *row*, **int** *col*)

void SetCellTextColour(**const wxColour&** *colour*)

Sets the text colour for the cell at the given position, or for the whole grid.

wxGrid::SetCellTextFont

void SetCellTextFont(**wxFont ***_font_, **int** *row*, **int** *col*)

void SetCellTextFont(**wxFont ***_font_)

Sets the text font for the cell at the given position, or for the whole grid.

wxGrid::SetCellValue

void SetCellValue(**const wxString&** *val*, **int** *row*, **int** *col*)

Sets the cell value at the given position.

wxGrid::SetColumnWidth

void SetColumnWidth(**int** *col*, **int** *width*)

Sets the width in pixels for column *col*.

wxGrid::SetDividerPen

void SetDividerPen(**wxPen ***_pen_)

Specifies the pen to be used for drawing the divisions between cells. The default is a light grey. If NULL is specified, the divisions will not be drawn.

wxGrid::SetEditable

void SetEditable(**Bool** *editable*)

If *editable* is TRUE (the default), the grid cells will be editable by means of the text edit control. If FALSE, the text edit control will be hidden and the user will not be able to edit the cell contents.

wxGrid::SetGridCursor

void SetGridCursor(**int** *row*, **int** *col*)

Sets the position of the selected cell.

wxGrid::SetLabelAlignment

void SetLabelAlignment(**int** *orientation*, **int** *alignment*)

Sets the row or column label alignment. *orientation* should be wxHORIZONTAL to specify column label, wxVERTICAL to specify row label. *alignment* should be wxCENTRE, wxLEFT or wxRIGHT.

wxGrid::SetLabelBackgroundColour

void SetLabelBackgroundColour(**const wxColour&** *value*)

Sets a row or column label background colour.

wxGrid::SetLabelSize

void SetLabelSize(**int** *orientation*, **int** *size*)

Sets the row label height, or column label width, in pixels. *orientation* should be wxHORIZONTAL to specify column label, wxVERTICAL to specify row label.

If a dimension of zero is specified, the row or column labels will not be shown.

wxGrid::SetLabelTextColour

void SetLabelTextColour(**const wxColour&** *value*)

Sets a row and column label text colour.

wxGrid::SetLabelTextFont

void SetLabelTextFont(**wxFont ***_font_)

Sets the font to be used for the row and column labels.

wxGrid::SetLabelValue

void SetLabelValue(**int** *orientation*, **const wxString&** *value*, **int** *pos*)

Sets a row or column label value. *orientation* should be wxHORIZONTAL to specify column label, wxVERTICAL to specify row label. *pos* is the label position.

wxGrid::SetRowHeight

void SetRowHeight(**int** *row*, **int** *height*)

Sets the height in pixels for row *row*.

wxGrid::UpdateDimensions

void UpdateDimensions(**void**)

Call this function whenever a change has been made via the API that might alter size characteristics. You may also need to follow it with a call to AdjustScrollbars.

5. Classes by category

A classification of grid classes by category.

5.1. Grid classes

- *wxGrid* (page 5)

6. Topic overviews

This chapter contains a selection of topic overviews.

6.1. wxGrid classes overview

wxGrid is a class for displaying and editing tabular information.

On each platform, the implementation has a different class name: on Motif, it is wxMotifGrid, and on Windows, the generic implementation wxGenericGrid is used. The wxGrid class is derived from a suitable platform-specific (or generic) implementation.

This separation of portable and platform-specific class names allows the programmer to use different grid implementations in the same program. For example, say the wxGenericGrid implementation is good at displaying lists without divisions, such as seen in the Windows 95 print manager window. But the wxMotifGrid implementation is better at displaying 1000 by 1000 grids, under Motif. In this case the programmer might choose to use both classes in the same program, thus 'optimizing' the Motif version of the application. (The example assumes that wxGenericGrid has been tested under Motif, which currently it has not).

To use wxGrid, include the wxgrid.h header file and link with the wxGrid library. Create a wxGrid object, or, if you need to override some default behaviour, create an object of a class derived from wxGrid. You need to call CreateGrid before there are any cells in the grid.

All row and column positions start from zero, and dimensions are in pixels.

If you make changes to row or column dimensions, call UpdateDimensions and then AdjustScrollbars. If you make changes to the grid appearance (such as a change of cell background colour or font), call Refresh for the changes to be shown.

6.1.1. Example

The following fragment is taken from the file test.cc. Note the call to UpdateDimensions, which is required if the application has changed any dimensions such as column width or row height. You may also need to call AdjustScrollbars. In this case, AdjustScrollbars isn't necessary because it will be called by wxGrid::OnSize which is invoked when the window is first displayed.

```
  // Make a grid
  frame->grid = new wxGrid(frame, 0, 0, 400, 400);

  frame->grid->CreateGrid(10, 8);
  frame->grid->SetColumnWidth(3, 200);
  frame->grid->SetRowHeight(4, 45);
  frame->grid->SetCellValue("First cell", 0, 0);
  frame->grid->SetCellValue("Another cell", 1, 1);
  frame->grid->SetCellValue("Yet another cell", 2, 2);
  frame->grid->SetCellTextFont(wxTheFontList->FindOrCreateFont(12,
wxROMAN, wxITALIC, wxNORMAL), 0, 0);
  frame->grid->SetCellTextColour(*wxRED, 1, 1);
  frame->grid->SetCellBackgroundColour(*wxCYAN, 2, 2);
  frame->grid->UpdateDimensions();
```

7. Change log

May 13th 1997, Version 1.0

- Added Sean Starkey's contribution for OnLabelLeftClick, OnLabelRightClick.

May 2nd 1996, Version 0.4

- Added BeginBatch, EndBatch, GetBatchCount functions for efficiency; SetCellValue and SetCellBitmap now update the cell display if not within BeginBatch/EndBatch functions.

April 20th 1996, Version 0.3

- Separated OnSelectCell from OnSelectCellImplementation.
- Made some optimisations for large grids.
- Can display bitmaps in cells.
- Compiled wxGenericGrid for Motif; discovered inconsistency in wxScrollBar::SetObjectLength, so changed its behaviour under Windows for wxWindos 1.66.

January 2nd 1996, Version 0.2

- First release.

Index