FL: a Frame Layout Framework

by Aleksandras Gluchovas and others

January 2nd 2002

Contents

Copyright notice

Introduction

What is FL?

This manual describes FL (Frame Layout), a class library for managing sophisticated window layout, with panes that can be moved around the main window and customized. FL handles many decoration and dragging issues, giving applications the kind of docking facilities that Visual C++ and Netscape Navigator possess.

FL was written by Aleksandras Gluchovas, and is heavily used in wxWorkshop which he also wrote the bulk of.

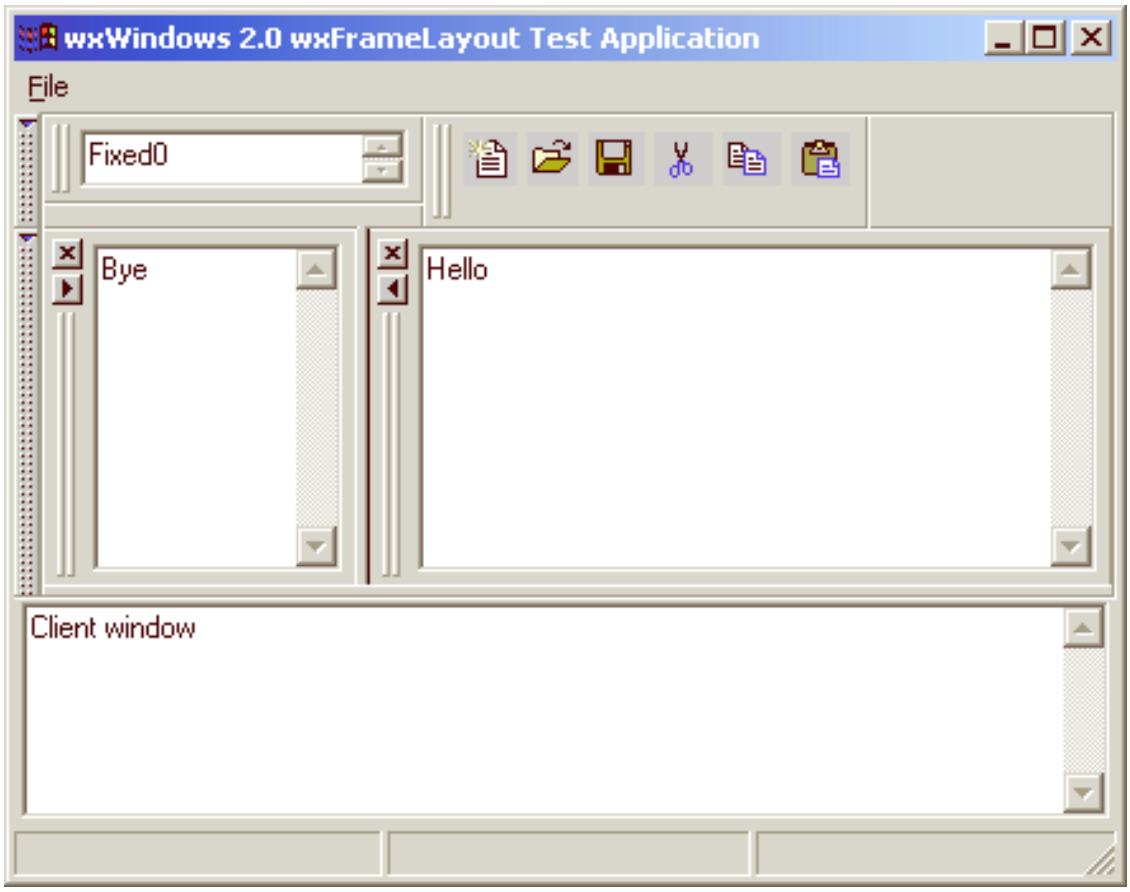
Please note that this guide is in its infancy, and contributions from FL users are very welcome.

The following screenshot (from fl_demo1) shows a frame with a number of bars that can be dragged around. The vertical grippers with two lines allow a bar to be dragged in that row, changing the ordering of the bar if necessary. The dotted grippers (as in Netscape Navigator) allow a whole row to be moved, again changing the position of the row if required. While moving a bar or row, immediate feedback is given as the moving bar displaces other bars.

Other features: the splitter bar shows a dotted thick line as it's dragged. Single-clicking on a row handle minimizes it to a horizontal tab which is given its own narrow row. This allows the user to temporarily hide a row while allowing quick access to it when required.

A close button (x) hides a bar completely. You can get it back again by right-clicking and selecting the appropriate menu item.

A left, right, up or down arrow button expands the pane in that direction.



Compiling and using FL

FL can be found under the 'contrib' hierarchy, in the following directories:

```
contrib/src/fl
contrib/include/wx/fl
contrib/samples/fl
contrib/docs/latex/fl
docs/html/fl
docs/htmlhelp/fl.chm
docs/pdf/fl.pdf
docs/winhelp/fl.hlp
```

To compile FL:

- Under Windows using VC++, open the flVC.dsw project and compile.
- Under Unix, FL should be configured when you configured wxWindows. Make FL by changing directory to contrib/src/fl and type 'make'. **Note:** there is currently a problem with the wxWindows build system that means that only the static version of library can be built at present.

To use FL:

- Under Windows using VC++, link with fl[d].lib.
- Under Unix, link with libfl[d].a.

FL concepts

These are typical steps when adding FL functionality to your application.

- include the appropriate header files;
- create a new *wxFrameLayout* (p. 75) passing the top-level frame and the window that is interpreted as the main 'client' window;
- set an updates manager for optimizing drag operations;
- add plugins for implementing various features;
- add bars;
- enable floating mode for the layout if required;
- delete the frame layout in the main frame's destructor.

The following is taken from fl_demo1 and shows the main code implementing the user interface as illustrated in *What is FL?* (p. 1).

```
// fl headers
#include "wx/fl/controlbar.h"     // core API

// extra plugins
#include "wx/fl/barhintspl.h"    // bevel for bars with "X"s and
grooves
#include "wx/fl/rowdragpl.h"     // NC-look with draggable rows
#include "wx/fl/cbcustom.h"      // customization plugin
#include "wx/fl/hintanimpl.h"

// beauty-care
#include "wx/fl/gcupdatesmgr.h"  // smooth d&d
#include "wx/fl/antiflickpl.h"   // double-buffered repaint of
decorations
#include "wx/fl/dyntbar.h"       // auto-layout toolbar
#include "wx/fl/dyntbarhnd.h"    // control-bar dimension handler
for it

MyFrame::MyFrame(wxFrame *frame)
    : wxFrame( frame, wxID_ANY, "wxWindows 2.0 wxFrameLayout Test
Application", wxDefaultPosition,
          wxSize( 700, 500 ),
          wxCLIP_CHILDREN | wxMINIMIZE_BOX | wxMAXIMIZE_BOX |
          wxRESIZE_BORDER | wxSYSTEM_MENU  | wxCAPTION,
          "freimas" )
{
    mpClientWnd = CreateTextCtrl( "Client window" );

    mpLayout = new wxFrameLayout( this, mpClientWnd );
```

```
    mpLayout->SetUpdatesManager( new cbGCUpdatesMgr() );

    // setup plugins for testing
    mpLayout->PushDefaultPlugins();

    mpLayout->AddPlugin( CLASSINFO( cbBarHintsPlugin ) ); //
fancy "X"es and bevel for bars
    mpLayout->AddPlugin( CLASSINFO( cbHintAnimationPlugin ) );
    mpLayout->AddPlugin( CLASSINFO( cbRowDragPlugin  ) );
    mpLayout->AddPlugin( CLASSINFO( cbAntiflickerPlugin ) );
    mpLayout->AddPlugin( CLASSINFO( cbSimpleCustomizationPlugin )
);

    // drop in some bars
    cbDimInfo sizes0( 200,45, // when docked horizontally
                      200,85, // when docked vertically
                      175,35, // when floated
                      FALSE,  // the bar is not fixed-size
                      4,      // vertical gap (bar border)
                      4       // horizontal gap (bar border)
                    );

    cbDimInfo sizes1( 150,35, // when docked horizontally
                      150,85, // when docked vertically
                      175,35, // when floated
                      TRUE,   // the bar is not fixed-size
                      4,      // vertical gap (bar border)
                      4       // horizontal gap (bar border)
                    );

    cbDimInfo sizes2( 175,45, // when docked horizontally
                      175,37, // when docked vertically
                      170,35, // when floated
                      TRUE,   // the bar is not fixed-size
                      4,      // vertical gap (bar border)
                      4,      // horizontal gap (bar border)
                      new cbDynToolBarDimHandler()
                    );

    mpLayout->AddBar( CreateTextCtrl("Hello"),  // bar window
                      sizes0, FL_ALIGN_TOP,     // alignment ( 0-
top,1-bottom, etc)
                      0,                        // insert into
0th row (vert. position)
                      0,                        // offset from
the start of row (in pixels)
                      "InfoViewer1",            // name for
reference in customization pop-ups
                      TRUE
                    );

    mpLayout->AddBar( CreateTextCtrl("Bye"),    // bar window
                      sizes0, FL_ALIGN_TOP,     // alignment ( 0-
top,1-bottom, etc)
                      1,                        // insert into
0th row (vert. position)
```

```
                        0,                          // offset from
the start of row (in pixels)
                        "InfoViewer2",              // name for
reference in customization pop-ups
                        TRUE
                    );

    mpLayout->AddBar( CreateTextCtrl("Fixed0"), // bar window
                        sizes1, FL_ALIGN_TOP,     // alignment ( 0-
top,1-bottom, etc)
                        0,                          // insert into
0th row (vert. position)
                        0,                          // offset from
the start of row (in pixels)
                        "ToolBar1",                 // name for
reference in customization pop-ups
                        TRUE
                    );

    wxDynamicToolBar* pToolBar = new wxDynamicToolBar();

    pToolBar->Create( this, -1 );

    // 1001-1006 ids of command events fired by added tool-
buttons

    pToolBar->AddTool( 1001, BMP_DIR "new.bmp" );
    pToolBar->AddTool( 1002, BMP_DIR "open.bmp" );
    pToolBar->AddTool( 1003, BMP_DIR "save.bmp" );

    pToolBar->AddTool( 1004, BMP_DIR "cut.bmp" );
    pToolBar->AddTool( 1005, BMP_DIR "copy.bmp" );
    pToolBar->AddTool( 1006, BMP_DIR "paste.bmp" );

    mpLayout->AddBar( pToolBar,             // bar window (can be
NULL)
                        sizes2, FL_ALIGN_TOP, // alignment ( 0-
top,1-bottom, etc)
                        0,                    // insert into 0th
row (vert. position)
                        0,                    // offset from the
start of row (in pixels)
                        "ToolBar2",           // name for reference
in customization pop-ups
                        FALSE
                    );

    mpLayout->EnableFloating( TRUE ); // off, thinking about
wxGtk...
}

MyFrame::~MyFrame()
{
    if ( mpLayout)
        delete mpLayout; // should be destroyed manually
}
```

Controlling dragging behaviour

Various pane-dragging behaviours are supported. FL can show an outline of where the window would be docked if you stopped dragging at that point.

This is a list of properties of interest in the cbCommonPaneProperties structure:

```
bool mRealTimeUpdatesOn;     // default: ON
bool mOutOfPaneDragOn;       // default: ON
bool mExactDockPredictionOn; // default: OFF
bool mNonDestructFrictionOn; // default: OFF
```

To get behaviour similar to Microsoft's DevStudio drag-ghost behaviour, mRealTimeUpdatesOn have to be set to FALSE, for example:

```
cbCommonPaneProperties props;
....
....
props.mRealTimeUpdatesOn = FALSE;
fl->SetPaneProperties( props, wxALL_PANES );
```

mOutOfPaneDragOn specifies whether bars can be dragged away from this pane. (Note: this may not currently be working.)

mExactDockPredictionOn is only relevant when *mRealTimeUpdatesOn* is FALSE, and then the hint rectangle behaves a little jumpily. It tries to show exatly how the bar would look and where it would be docked if the dragging finished right now, i.e. the final position, with all the 'friction-physics' calculated. Otherwise the hint flies smothly above the surface only hinting whether the bar will be docked vertically or horizontally if dropped now. This is a feature you won't find anywhere else!

mNonDestructFirctionOn causes the bars not being dragged to stay where they are, while the currently dragged one is 'diving' through the underlaying panes, docking itself in and out in real time. Otherwise the stationary bars would be pushed around messing up the composition permanently. This flag is irelevant when *mRealTimeUpdatesOn* is FALSE, as the ghost-rect does not do any docking until the drag finishes.

Alphabetical class reference

BagLayout

BagLayout lays out items in left-to-right order from top to bottom.

Derived from

LayoutManagerBase (p. 90)

Include files

<wx/fl/dyntbar.h>

Data structures

BagLayout::Layout

void Layout(**const wxSize&** *parentDim*, **wxSize&** *resultingDim*, **wxLayoutItemArrayT&** *items*, **int** *horizGap*, **int** *vertGap*)

Constructor.

wxBarIterator

Used for traversing through all bars of all rows in the pane.

Derived from

No base class

Include files

<wx/fl/controlbar.h>

Data structures

wxBarIterator::wxBarIterator

wxBarIterator(**RowArrayT&** *rows*)

Constructor, taking row array.

wxBarIterator::BarInfo

cbBarInfo& BarInfo()

Gets the current bar information.

wxBarIterator::Next

bool Next()

Advances the iterator and returns TRUE if a bar is available.

wxBarIterator::Reset

void Reset()

Resets the iterator to the start of the first row.

wxBarIterator::RowInfo

cbRowInfo& RowInfo()

Returns a reference to the currently traversed row.

cbAntiflickerPlugin

Implements double-buffering to reduce flicker. Bitmap and memory DC buffers are shared 'resources' among all instances of antiflicker plugins within the application.

Locking for multithreaded applications is not yet implemented.

Derived from

cbPluginBase (p. 51)

Include files

<wx/fl/antiflickpl.h>

Data structures

cbAntiflickerPlugin::cbAntiflickerPlugin

cbAntiflickerPlugin()

Default constructor.

cbAntiflickerPlugin(**wxFrameLayout*** *pPanel*, **int** *paneMask* = *wxALL_PANES*)

Constructor taking frame layout panel, and pane mask.

cbAntiflickerPlugin::~cbAntiflickerPlugin

~cbAntiflickerPlugin()

Destructor.

cbAntiflickerPlugin::AllocNewBuffer

wxDC* AllocNewBuffer(**const wxRect&** *forArea*)

Allocates a suitable buffer.

cbAntiflickerPlugin::FindSuitableBuffer

wxDC* FindSuitableBuffer(**const wxRect&** *forArea*)

Finds a suitable buffer. Returns NULL if a suitable buffer is not present.

cbAntiflickerPlugin::GetClientDC

wxDC& GetClientDC()

Gets the client device context.

cbAntiflickerPlugin::GetWindowDC

wxDC& GetWindowDC()

Gets the window device context.

cbAntiflickerPlugin::OnFinishDrawInArea

void OnFinishDrawInArea(**cbFinishDrawInAreaEvent&** *event*)

Handler for plugin event.

cbAntiflickerPlugin::OnStartDrawInArea

void OnStartDrawInArea(**cbStartDrawInAreaEvent&** *event*)

Handler for plugin event.

cbBarDimHandlerBase

Abstract interface for bar-size handler classes. These objects receive notifications whenever the docking state of the bar is changed, thus they provide the possibility to adjust the values in cbDimInfo::mSizes accordingly. Specific handlers can be hooked up to specific types of bar.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

cbBarDimHandlerBase::cbBarDimHandlerBase

cbBarDimHandlerBase()

to multiple bars, it's instance is reference-counted Default constructor. The initial reference count is 0, since the handler is not used until the first invocation of AddRef().

cbBarDimHandlerBase::AddRef

void AddRef()

Increments the reference count.

cbBarDimHandlerBase::OnChangeBarState

void OnChangeBarState(**cbBarInfo*** *pBar*, **int** *newState*)

Responds to "bar-state-changes" notifications.

cbBarDimHandlerBase::OnResizeBar

void OnResizeBar(**cbBarInfo*** *pBar*, **const wxSize&** *given*, **wxSize&** *preferred*)

Responds to bar resize notifications.

cbBarDimHandlerBase::RemoveRef

void RemoveRef()

Decrements the reference count, and if the count is at zero, delete 'this'.

cbBarDragPlugin

Plugin class implementing bar dragging.

Derived from

cbPluginBase (p. 51)

Include files

<wx/fl/bardragpl.h>

Data structures

cbBarDragPlugin::cbBarDragPlugin

cbBarDragPlugin()

Default constructor.

cbBarDragPlugin(**wxFrameLayout*** *pPanel*, **int** *paneMask* = *wxALL_PANES*)

Constructor taking a parent frame, and flag. See cbPluginBase.

cbBarDragPlugin::~cbBarDragPlugin

~cbBarDragPlugin()

Destructor.

cbBarDragPlugin::AdjustHintRect

void AdjustHintRect(**wxPoint&** *mousePos*)

the thicker rectangle is drawn using hatched brush, the default border width for this rectangle is 8 pix. Internal implementation function.

cbBarDragPlugin::CalcOnScreenDims

void CalcOnScreenDims(**wxRect&** *rect*)

Internal implementation function.

cbBarDragPlugin::ClipPosInFrame

void ClipPosInFrame(**wxPoint&** *pos*)

Internal implementation function.

cbBarDragPlugin::ClipRectInFrame

void ClipRectInFrame(**wxRect&** *rect*)

Internal implementation function.

cbBarDragPlugin::DoDrawHintRect

void DoDrawHintRect(**wxRect&** *rect*, **bool** *isInClientRect*)

Internal implementation function. Draw the hint rectangle.

cbBarDragPlugin::DrawHintRect

void DrawHintRect(**wxRect&** *rect*, **bool** *isInClientRect*)

Internal implementation function. Draw the visual hint while dragging.

cbBarDragPlugin::EraseHintRect

void EraseHintRect(**wxRect&** *rect*, **bool** *isInClientRect*)

Internal implementation function. Erase the visual hint while dragging.

cbBarDragPlugin::FinishTracking

void FinishTracking()

Internal implementation function. Stop showing the visual hint while dragging.

cbBarDragPlugin::GetBarHeightInPane

int GetBarHeightInPane(**cbDockPane*** *pPane*)

Internal implementation function.

cbBarDragPlugin::GetBarWidthInPane

int GetBarWidthInPane(**cbDockPane*** *pPane*)

Internal implementation function.

cbBarDragPlugin::GetDistanceToPane

int GetDistanceToPane(**cbDockPane*** *pPane*, **wxPoint&** *mousePos*)

Internal implementation function.

cbBarDragPlugin::HitTestPanes

cbDockPane* HitTestPanes(**wxRect&** *rect*)

Internal implementation function. Finds the pane under the specified rectangle.

cbDockPane* HitTestPanes(**wxPoint&** *pos*)

Internal implementation function. Finds the pane under the specified point.

cbBarDragPlugin::HitsPane

bool HitsPane(**cbDockPane*** *pPane*, **wxRect&** *rect*)

Internal implementation function.

cbBarDragPlugin::IsInClientArea

bool IsInClientArea(**wxPoint&** *mousePos*)

Internal implementation function.

bool IsInClientArea(**wxRect&** *rect*)

Internal implementation function.

cbBarDragPlugin::IsInOtherPane

bool IsInOtherPane(**wxPoint&** *mousePos*)

Internal implementation function.

cbBarDragPlugin::OnDrawHintRect

void OnDrawHintRect(**cbDrawHintRectEvent&** *event*)

Handles event, which originates from itself.

cbBarDragPlugin::OnLButtonDown

void OnLButtonDown(**cbLeftDownEvent&** *event*)

Handler for plugin event.

cbBarDragPlugin::OnLButtonUp

void OnLButtonUp(**cbLeftUpEvent&** *event*)

Handler for plugin event.

cbBarDragPlugin::OnLDblClick

void OnLDblClick(**cbLeftDClickEvent&** *event*)

Handler for plugin event.

cbBarDragPlugin::OnMouseMove

void OnMouseMove(**cbMotionEvent&** *event*)

Handler for plugin event.

cbBarDragPlugin::OnStartBarDragging

void OnStartBarDragging(**cbStartBarDraggingEvent&** *event*)

Handler for plugin event.

cbBarDragPlugin::RectToScr

void RectToScr(**wxRect&** *frameRect*, **wxRect&** *scrRect*)

Internal implementation function. Converts the given rectangle from window to screen coordinates.

cbBarDragPlugin::ShowHint

void ShowHint(**bool** *prevWasInClient*)

Internal implementation function. Show the hint; called within OnMouseMove.

cbBarDragPlugin::StartTracking

void StartTracking()

on-screen hint-tracking related methods Internal implementation function. Start showing a visual hint while dragging.

cbBarDragPlugin::StickToPane

void StickToPane(**cbDockPane*** *pPane*, **wxPoint&** *mousePos*)

Internal implementation function.

cbBarDragPlugin::UnstickFromPane

void UnstickFromPane(**cbDockPane*** *pPane*, **wxPoint&** *mousePos*)

Internal implementation function.

cbBarHintsPlugin

This class intercepts bar-decoration and sizing events, and draws 3D hints around fixed and flexible bars, similar to those in Microsoft DevStudio 6.x

Derived from

cbPluginBase (p. 51)

Include files

<wx/fl/barhintspl.h>

Data structures

cbBarHintsPlugin::cbBarHintsPlugin

cbBarHintsPlugin()

Default constructor.

cbBarHintsPlugin(**wxFrameLayout*** *pLayout*, **int** *paneMask* = *wxALL_PANES*)

Constructor, taking parent frame and pane mask flag.

cbBarHintsPlugin::~cbBarHintsPlugin

~cbBarHintsPlugin()

Destructor.

cbBarHintsPlugin::CreateBoxes

void CreateBoxes()

Helper function: creates close and collapse boxes.

cbBarHintsPlugin::DoDrawHint

void DoDrawHint(**wxDC&** *dc*, **wxRect&** *rect*, **int** *pos*, **int** *boxOfs*, **int** *grooveOfs*, **bool** *isFixed*)

Helper function: draws a hint.

cbBarHintsPlugin::Draw3DBox

void Draw3DBox(**wxDC&** *dc*, **const wxPoint&** *pos*, **bool** *pressed*)

Helper function: draws a 3D box.

cbBarHintsPlugin::DrawCloseBox

void DrawCloseBox(**wxDC&** *dc*, **const wxPoint&** *pos*, **bool** *pressed*)

Helper function: draws a close box.

cbBarHintsPlugin::DrawCollapseBox

void DrawCollapseBox(**wxDC&** *dc*, **const wxPoint&** *pos*, **bool** *atLeft*, **bool** *disabled*, **bool** *pressed*)

Helper function: draws a collapse box.

cbBarHintsPlugin::DrawGrooves

void DrawGrooves(**wxDC&** *dc*, **const wxPoint&** *pos*, **int** *length*)

Helper function: draws grooves.

cbBarHintsPlugin::ExcludeHints

void ExcludeHints(**wxRect&** *rect*, **cbBarInfo&** *info*)

Helper function.

cbBarHintsPlugin::GetHintsLayout

void GetHintsLayout(**wxRect&** *rect*, **cbBarInfo&** *info*, **int&** *boxOfs*, **int&** *grooveOfs*, **int&** *pos*)

Helper function: gets the layout of a hint.

cbBarHintsPlugin::HitTestHints

int HitTestHints(**cbBarInfo&** *info*, **const wxPoint&** *pos*)

Helper function: returns information about the hint under the given position.

cbBarHintsPlugin::OnDrawBarDecorations

void OnDrawBarDecorations(**cbDrawBarDecorEvent&** *event*)

Handles a plugin event.

cbBarHintsPlugin::OnInitPlugin

void OnInitPlugin()

Called to initialize this plugin.

cbBarHintsPlugin::OnLeftDown

void OnLeftDown(**cbLeftDownEvent&** *event*)

Handles a plugin event.

cbBarHintsPlugin::OnLeftUp

void OnLeftUp(**cbLeftUpEvent&** *event*)

Handles a plugin event.

cbBarHintsPlugin::OnMotion

void OnMotion(**cbMotionEvent&** *event*)

Handles a plugin event.

cbBarHintsPlugin::OnSizeBarWindow

void OnSizeBarWindow(**cbSizeBarWndEvent&** *event*)

Handles a plugin event.

cbBarHintsPlugin::SetGrooveCount

void SetGrooveCount(**int** *nGrooves*)

Set the number of grooves to be shown in the pane.

cbBarInfo

Helper class used internally by the wxFrameLayout class. Holds and manages bar information.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

cbBarInfo::cbBarInfo

cbBarInfo()

Constructor.

cbBarInfo::~cbBarInfo

~cbBarInfo()

Destructor.

cbBarInfo::IsExpanded

bool IsExpanded() **const**

Returns TRUE if this bar is expanded.

cbBarInfo::IsFixed

bool IsFixed() **const**

Returns TRUE if this bar is fixed.

cbBarShapeData

Used for storing the original bar's positions in the row, when the 'non-destructive-friction' option is turned on.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

cbBarSpy

Helper class, used for spying for unhandled mouse events on control bars and forwarding them to the frame layout.

Derived from

wxEvtHandler

Include files

<wx/fl/controlbar.h>

Data structures

```
typedef cbBarInfo* BarInfoPtrT
```

forward declarations

```
typedef cbRowInfo* RowInfoPtrT
```

enumeration of hittest results, see cbDockPane::HitTestPaneItems(..)

```
enum CB_HITTEST_RESULT
{
    CB_NO_ITEMS_HITTED,

    CB_UPPER_ROW_HANDLE_HITTED,
    CB_LOWER_ROW_HANDLE_HITTED,
    CB_LEFT_BAR_HANDLE_HITTED,
    CB_RIGHT_BAR_HANDLE_HITTED,
    CB_BAR_CONTENT_HITTED
}
```

cbBarSpy::cbBarSpy

cbBarSpy(**wxFrameLayout*** *pPanel*)

Constructor, taking a parent pane.

cbBarSpy()

Default constructor.

cbBarSpy::ProcessEvent

bool ProcessEvent(**wxEvent&** *event*)

Performs special event processing.

cbBarSpy::SetBarWindow

void SetBarWindow(**wxWindow*** *pWnd*)

Sets the bar window.

cbCloseBox

cbCloseBox is a window close button, used in a wxToolWindow titlebar.

Derived from

cbMiniButton (p. 44)

Include files

<wx/fl/toolwnd.h>

Data structures

cbCloseBox::Draw

void Draw(**wxDC&** *dc*)

Draws the close button appearance.

cbCollapseBox

cbCollapseBox is a window collapse button, used in a wxToolWindow titlebar.

Derived from

cbMiniButton (p. 44)

Include files

<wx/fl/toolwnd.h>

Data structures

cbCollapseBox::Draw

void Draw(**wxDC&** *dc*)

Draws the collapse button appearance.

cbCommonPaneProperties

A structure holding configuration options, which are usually the same for all panes in a frame layout. For an explanation of the data members, please see *Controlling dragging behaviour* (p. 6).

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

```
class cbCommonPaneProperties : public wxObject
{
    DECLARE_DYNAMIC_CLASS( cbCommonPaneProperties )

    // Look-and-feel configuration

    bool mRealTimeUpdatesOn;     // default: ON
    bool mOutOfPaneDragOn;       // default: ON
    bool mExactDockPredictionOn; // default: OFF
    bool mNonDestructFrictionOn; // default: OFF

    bool mShow3DPaneBorderOn;    // default: ON

    // The following properties are reserved for the "future"

    bool mBarFloatingOn;         // default: OFF
    bool mRowProportionsOn;      // default: OFF
    bool mColProportionsOn;      // default: ON
    bool mBarCollapseIconsOn;    // default: OFF
    bool mBarDragHintsOn;        // default: OFF

    // Minimal dimensions for not-fixed bars in this pane (16x16
default)

    wxSize mMinCBarDim;

    // Width/height of resizing sash

    int    mResizeHandleSize;
```

```
        // Default constructor.

        cbCommonPaneProperties(void);
};
```

cbCustomizeBarEvent

Class for bar customization events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbCustomizeBarEvent::cbCustomizeBarEvent

cbCustomizeBarEvent(**cbBarInfo*** *pBar*, **const wxPoint&** *clickPos*, **cbDockPane*** *pPane*)

Constructor, taking bar information, mouse position, and pane.

cbCustomizeLayoutEvent

Class for layout customization events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbCustomizeLayoutEvent::cbCustomizeLayoutEvent

cbCustomizeLayoutEvent(**const wxPoint&** *clickPos*)

Constructor, taking mouse position.

cbDimInfo

Helper class used internally by the wxFrameLayout class. Holds and manages information about bar dimensions.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

cbDimInfo::cbDimInfo

cbDimInfo(**cbBarDimHandlerBase*** *pDimHandler*, **bool** *isFixed*)

Constructor. isFixed is TRUE if vertical/horizontal dimensions cannot be manually adjusted by the user using resizing handles. If FALSE, the frame-layout automatically places resizing handles among bars that do are not fixed.

cbDimInfo(**int** *dh_x*, **int** *dh_y*, **int** *dv_x*, **int** *dv_y*, **int** *f_x*, **int** *f_y*, **bool** *isFixed = TRUE*, **int** *horizGap = 6*, **int** *vertGap = 6*, **cbBarDimHandlerBase*** *pDimHandler = NULL*)

Constructor taking dimenstion information. dh_x, dh_y are the dimensions when docked horizontally. dv_x, dv_y are the dimensions when docked vertically. f_x, f_y are the dimensions when floating. For information on isFixed, see comments above. horizGap is the left/right gap, separating decorations from the bar's actual wndow, filled with the frame's background colour. The dimension is given in the frame's coordinates. vertGap is the top/bottom gap, separating decorations from the bar's actual wndow, filled with the frame's background colour. The dimension is given in the frame's coordinates.

cbDimInfo(**int** *x*, **int** *y*, **bool** *isFixed = TRUE*, **int** *gap = 6*, **cbBarDimHandlerBase*** *pDimHandler = NULL*)

Constructor.

cbDimInfo()

Default constructor.

cbDimInfo::~cbDimInfo

~cbDimInfo()

Destructor. Destroys handler automatically, if present.

cbDimInfo::GetDimHandler

cbBarDimHandlerBase* GetDimHandler()

Returns the handler, if any.

cbDimInfo::operator=

const cbDimInfo& operator operator=(**const cbDimInfo&** *other*)

Assignment operator.

cbDockBox

cbDockBox is a window dock button, used in a wxToolWindow titlebar.

Derived from

cbMiniButton (p. 44)

Include files

<wx/fl/toolwnd.h>

Data structures

cbDockBox::Draw

void Draw(**wxDC&** *dc*)

Draws the dock button appearance.

cbDockPane

This class manages containment and control of control bars along one of the four edges of the parent frame.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

cbDockPane::cbDockPane

cbDockPane(**int** *alignment*, **wxFrameLayout*** *pPanel*)

Constructor, taking alignment and layout panel.

cbDockPane()

public members Default constructor.

cbDockPane::~cbDockPane

~cbDockPane()

Destructor.

cbDockPane::BarPresent

bool BarPresent(**cbBarInfo*** *pBar*)

Returns TRUE if the given bar is present in this pane.

cbDockPane::CalcLengthRatios

void CalcLengthRatios(**cbRowInfo*** *pInRow*)

Calculate lengths. Internal function called by plugins.

cbDockPane::ContractBar

void ContractBar(**cbBarInfo*** *pBar*)

Contracts the bar. Internal function called by plugins.

cbDockPane::DoInsertBar

void DoInsertBar(**cbBarInfo*** *pBar*, **int** *rowNo*)

Inserts the bar at the given row number. Internal function called by plugins.

cbDockPane::DrawHorizHandle

void DrawHorizHandle(**wxDC&** *dc*, **int** *x*, **int** *y*, **int** *width*)

Row/bar resizing related helper-method.

cbDockPane::DrawVertHandle

void DrawVertHandle(**wxDC&** *dc*, **int** *x*, **int** *y*, **int** *height*)

protected really (accessed only by plugins) Row/bar resizing related helper-method.

cbDockPane::ExpandBar

void ExpandBar(**cbBarInfo*** *pBar*)

Expands the bar. Internal function called by plugins.

cbDockPane::FinishDrawInArea

void FinishDrawInArea(**const wxRect&** *area*)

Generates cbFinishDrawInAreaEvent and sends it to the layout. Internal function called by plugins.

cbDockPane::FrameToPane

void FrameToPane(**int*** *x*, **int*** *y*)

Coordinate translation between parent's frame and this pane. Internal function called by plugins.

void FrameToPane(**wxRect*** *pRect*)

Coordinate translation between parent's frame and this pane. Internal function called by plugins.

cbDockPane::GetAlignment

int GetAlignment()

Returns the alignment for this pane. The value is one of FL_ALIGN_TOP, FL_ALIGN_BOTTOM, FL_ALIGN_LEFT, FL_ALIGN_RIGHT.

cbDockPane::GetBarInfoByWindow

cbBarInfo* GetBarInfoByWindow(**wxWindow*** *pBarWnd*)

Finds the bar information by corresponding window.

cbDockPane::GetBarResizeRange

void GetBarResizeRange(**cbBarInfo*** *pBar*, **int*** *from*, **int*** *till*, **bool** *forLeftHandle*)

Returns the bar's resize range.

cbDockPane::GetDockingState

int GetDockingState()

Returns wxCBAR_DOCKED_HORIZONTALLY if the alignment is top or bottom, or wxCBAR_DOCKED_VERTICALLY otherwise.

cbDockPane::GetFirstRow

cbRowInfo* GetFirstRow()

Returns the first row.

cbDockPane::GetMinimalRowHeight

int GetMinimalRowHeight(**cbRowInfo*** *pRow*)

Returns the minimal row height for the given row. Internal function called by plugins.

cbDockPane::GetNotFixedBarsCount

int GetNotFixedBarsCount(**cbRowInfo*** *pRow*)

Returns the number of bars whose size is not fixed. Internal function called by plugins.

cbDockPane::GetPaneHeight

int GetPaneHeight()

Returns the height in the pane's coordinates.

cbDockPane::GetRealRect

wxRect& GetRealRect()

Returns the bounds of the pane, in parent coordinates.

cbDockPane::GetRow

cbRowInfo* GetRow(**int** *row*)

protected really (accessed only by plugins) Returns the row info for a row index. Internal function called by plugins.

cbDockPane::GetRowAt

int GetRowAt(**int** *paneY*)

Returns the row at the given vertical position. Returns -1 if the row is not present at given vertical position. Internal function called by plugins.

int GetRowAt(**int** *upperY*, **int** *lowerY*)

Returns the row between the given vertical positions. Returns -1 if the row is not present. Internal function called by plugins.

cbDockPane::GetRowIndex

int GetRowIndex(**cbRowInfo*** *pRow*)

Returns the row index for the given row info. Internal function called by plugins.

cbDockPane::GetRowList

RowArrayT& GetRowList()

Returns an array of rows. Used by updates-managers.

cbDockPane::GetRowResizeRange

void GetRowResizeRange(**cbRowInfo*** *pRow*, **int*** *from*, **int*** *till*, **bool** *forUpperHandle*)

Returns the row's resize range.

cbDockPane::GetRowShapeData

void GetRowShapeData(**cbRowInfo*** *pRow*, **wxList*** *pLst*)

Returns row shape data. cbBarShapeData objects will be added to the given pLst. cbBarShapeData is used for storing the original bar's positions in the row, when the 'non-destructive-friction' option is turned on.

cbDockPane::GetRowY

int GetRowY(**cbRowInfo*** *pRow*)

Gets the vertical position at the given row. Internal function called by plugins.

cbDockPane::HasNotFixedBarsLeft

bool HasNotFixedBarsLeft(**cbBarInfo*** *pBar*)

Returns TRUE if there are any variable-sized rows to the left of this one. Internal function called by plugins.

cbDockPane::HasNotFixedBarsRight

bool HasNotFixedBarsRight(**cbBarInfo*** *pBar*)

Returns TRUE if there are any variable-sized rows to the right of this one. Internal function called by plugins.

cbDockPane::HasNotFixedRowsAbove

bool HasNotFixedRowsAbove(**cbRowInfo*** *pRow*)

Returns TRUE if there are any variable-sized rows above this one. Internal function called by plugins.

cbDockPane::HasNotFixedRowsBelow

bool HasNotFixedRowsBelow(**cbRowInfo*** *pRow*)

Returns TRUE if there are any variable-sized rows below this one. Internal function called by plugins.

cbDockPane::HasPoint

bool HasPoint(**const wxPoint&** *pos*, **int** *x*, **int** *y*, **int** *width*, **int** *height*)

Returns TRUE if pos is within the given rectangle. Internal function called by plugins.

cbDockPane::HitTestPaneItems

int HitTestPaneItems(**const wxPoint&** *pos*, **cbRowInfo**** *ppRow*, **cbBarInfo**** *ppBar*)

Returns the result of hit-testing items in the pane. See CB_HITTEST_RESULT enumerated type. pos is the position in this pane's coordinates.

cbDockPane::InitLinksForRow

void InitLinksForRow(**cbRowInfo*** *pRow*)

Sets up links between bars. Internal function called by plugins.

cbDockPane::InitLinksForRows

void InitLinksForRows()

Sets up links between bars. Internal function called by plugins.

cbDockPane::InsertBar

void InsertBar(**cbBarInfo*** *pBarInfo*)

Inserts bar and sets its position according to the preferred settings given in pBarInfo.

void InsertBar(**cbBarInfo*** *pBar*, **const wxRect&** *rect*)

Inserts the bar into this pane. rect is given in the parent frame's coordinates.

void InsertBar(**cbBarInfo*** *pBar*, **cbRowInfo*** *pIntoRow*)

Inserts the bar into the given row, with dimensions and position stored in pBarInfo->mBounds. Returns the node of inserted bar.

cbDockPane::InsertRow

void InsertRow(**cbRowInfo*** *pRow*, **cbRowInfo*** *pBeforeRow*)

Inserts a row. Does not refresh the inserted row immediately. If pBeforeRowNode is NULL, the row is appended to the end of pane's row list.

cbDockPane::IsFixedSize

bool IsFixedSize(**cbBarInfo*** *pInfo*)

Returns TRUE if the bar's dimension information indicates a fixed size. Internal function called by plugins.

cbDockPane::IsHorizontal

bool IsHorizontal()

Returns TRUE if the pane is aligned to the top or bottom.

cbDockPane::MatchesMask

bool MatchesMask(**int** *paneMask*)

Returns TRUE if the given mask matches the pane's mask.

cbDockPane::PaintBar

void PaintBar(**cbBarInfo*** *pBar*, **wxDC&** *dc*)

Calls PaintBarDecorations and PaintBarHandles. Internal function called by plugins.

cbDockPane::PaintBarDecorations

void PaintBarDecorations(**cbBarInfo*** *pBar*, **wxDC&** *dc*)

protected really (accessed only by plugins) Generates a cbDrawBarDecorEvent and sends it to the layout to paint the bar decorations. Internal function called by plugins.

cbDockPane::PaintBarHandles

void PaintBarHandles(**cbBarInfo*** *pBar*, **wxDC&** *dc*)

Generates a cbDrawBarHandlesEvent and sends it to the layout to paint the bar handles. Internal function called by plugins.

cbDockPane::PaintPane

void PaintPane(**wxDC&** *dc*)

Paints the pane background, the row background and decorations, and finally the pane decorations. Internal function called by plugins.

cbDockPane::PaintPaneBackground

void PaintPaneBackground(**wxDC&** *dc*)

Generates cbDrawPaneBkGroundEvent and sends it to the layout. Internal function called by plugins.

cbDockPane::PaintPaneDecorations

void PaintPaneDecorations(**wxDC&** *dc*)

Generates cbDrawPaneDecorEvent and sends it to the layout. Internal function called by plugins.

cbDockPane::PaintRow

void PaintRow(**cbRowInfo*** *pRow*, **wxDC&** *dc*)

Calls PaintRowBackground, PaintRowDecorations, PaintRowHandles. Internal function called by plugins.

cbDockPane::PaintRowBackground

void PaintRowBackground(**cbRowInfo*** *pRow*, **wxDC&** *dc*)

Generates cbDrawRowBkGroundEvent and sends it to the layout. Internal function called by plugins.

cbDockPane::PaintRowDecorations

void PaintRowDecorations(**cbRowInfo*** *pRow*, **wxDC&** *dc*)

Calls PaintBarDecorations for each row. Internal function called by plugins.

cbDockPane::PaintRowHandles

void PaintRowHandles(**cbRowInfo*** *pRow*, **wxDC&** *dc*)

Generates cbDrawRowHandlesEvent and cbDrawRowDecorEvent and sends them to the layout. Internal function called by plugins.

cbDockPane::PaneToFrame

void PaneToFrame(**wxRect*** *pRect*)

Coordinate translation between parent's frame and this pane. Internal function called by plugins.

void PaneToFrame(**int*** *x*, **int*** *y*)

Coordinate translation between parent's frame and this pane. Internal function called by plugins.

cbDockPane::RecalcLayout

void RecalcLayout()

Generates events to perform layout calculations.

cbDockPane::RecalcRowLayout

void RecalcRowLayout(**cbRowInfo*** *pRow*)

Generates a cbLayoutRowEvent event to recalculate row layouts. Internal function called by plugins.

cbDockPane::RemoveBar

void RemoveBar(**cbBarInfo*** *pBar*)

Removes the bar from this pane. Does not destroy the bar.

cbDockPane::RemoveRow

void RemoveRow(**cbRowInfo*** *pRow*)

Removes the row from this pane. Does not destroy the row object.

cbDockPane::ResizeBar

void ResizeBar(**cbBarInfo*** *pBar*, **int** *ofs*, **bool** *forLeftHandle*)

Row/bar resizing related helper-method.

cbDockPane::ResizeRow

void ResizeRow(**cbRowInfo*** *pRow*, **int** *ofs*, **bool** *forUpperHandle*)

Row/bar resizing related helper-method.

cbDockPane::SetBoundsInParent

void SetBoundsInParent(**const wxRect&** *rect*)

Set the position and dimensions of the pane in the parent frame's coordinates.

cbDockPane::SetMargins

void SetMargins(**int** *top*, **int** *bottom*, **int** *left*, **int** *right*)

Sets pane's margins in frame's coordinate orientations.

cbDockPane::SetPaneWidth

void SetPaneWidth(**int** *width*)

Sets pane's width in the pane's coordinates (including margins).

cbDockPane::SetRowHeight

void SetRowHeight(**cbRowInfo*** *pRow*, **int** *newHeight*)

Sets the row height for the given height. newHeight includes the height of row handles, if present. Internal function called by plugins.

cbDockPane::SetRowShapeData

void SetRowShapeData(**cbRowInfo*** *pRowNode*, **wxList*** *pLst*)

Sets the shape data for the given row, using the data provided in pLst. cbBarShapeData is used for storing the original bar's positions in the row, when the 'non-destructive-friction' option is turned on.

cbDockPane::SizeBar

void SizeBar(**cbBarInfo*** *pBar*)

Generates a cbSizeBarWndEvent and sends it to the layout. Internal function called by plugins.

cbDockPane::SizePaneObjects

void SizePaneObjects()

Calls SizeRowObjects for each row. Internal function called by plugins.

cbDockPane::SizeRowObjects

void SizeRowObjects(**cbRowInfo*** *pRow*)

Calls SizeBar for each bar in the row. Internal function called by plugins.

cbDockPane::StartDrawInArea

wxDC* StartDrawInArea(**const wxRect&** *area*)

Generates cbStartDrawInAreaEvent and sends it to the layout. Internal function called by plugins.

cbDockPane::SyncRowFlags

void SyncRowFlags(**cbRowInfo*** *pRow*)

Sets up flags in the row information structure, so that they match the changed state of row items correctly. Internal function called by plugins.

cbDrawBarDecorEvent

Class for bar decoration drawing events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbDrawBarDecorEvent::cbDrawBarDecorEvent

cbDrawBarDecorEvent(**cbBarInfo*** *pBar*, **wxDC&** *dc*, **cbDockPane*** *pPane*)

Constructor, taking bar information, device context, and pane.

cbDrawBarHandlesEvent

Class for bar handles drawing events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbDrawBarHandlesEvent::cbDrawBarHandlesEvent

cbDrawBarHandlesEvent(**cbBarInfo*** *pBar*, **wxDC&** *dc*, **cbDockPane*** *pPane*)

Constructor, taking bar information, device context, and pane.

cbDrawHintRectEvent

Class for hint-rectangle drawing events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbDrawHintRectEvent::cbDrawHintRectEvent

cbDrawHintRectEvent(**const wxRect&** *rect*, **bool** *isInClient*, **bool** *eraseRect*, **bool** *lastTime*)

e.g. with fat/hatched border Constructor, taking hint rectangle and three flags.

cbDrawPaneBkGroundEvent

Class for pane background drawing events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbDrawPaneBkGroundEvent::cbDrawPaneBkGroundEvent

cbDrawPaneBkGroundEvent(**wxDC&** *dc*, **cbDockPane*** *pPane*)

Constructor, taking device context and pane.

cbDrawPaneDecorEvent

Class for pane decoration drawing events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbDrawPaneDecorEvent::cbDrawPaneDecorEvent

cbDrawPaneDecorEvent(**wxDC&** *dc*, **cbDockPane*** *pPane*)

Constructor, taking device context and pane.

cbDrawRowBkGroundEvent

Class for row background drawing events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbDrawRowBkGroundEvent::cbDrawRowBkGroundEvent

cbDrawRowBkGroundEvent(**cbRowInfo*** *pRow*, **wxDC&** *dc*, **cbDockPane*** *pPane*)

Constructor, taking row information, device context, and pane.

cbDrawRowDecorEvent

Class for row decoration drawing events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbDrawRowDecorEvent::cbDrawRowDecorEvent

cbDrawRowDecorEvent(**cbRowInfo*** *pRow*, **wxDC&** *dc*, **cbDockPane*** *pPane*)

Constructor, taking row information, device context, and pane.

cbDrawRowHandlesEvent

Class for row handles drawing events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbDrawRowHandlesEvent::cbDrawRowHandlesEvent

cbDrawRowHandlesEvent(**cbRowInfo*** *pRow*, **wxDC&** *dc*, **cbDockPane*** *pPane*)

Constructor, taking row information, device context, and pane.

cbDynToolBarDimHandler

Dynamic toolbar dimension handler.

Derived from

cbBarDimHandlerBase (p. 9)

Include files

<wx/fl/dyntbarhnd.h>

Data structures

cbDynToolBarDimHandler::OnChangeBarState

void OnChangeBarState(**cbBarInfo*** *pBar*, **int** *newState*)

Called when the bar changes state.

cbDynToolBarDimHandler::OnResizeBar

void OnResizeBar(**cbBarInfo*** *pBar*, **const wxSize&** *given*, **wxSize&** *preferred*)

Called when a bar is resized.

cbFinishDrawInAreaEvent

Class for finish drawing in area events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbFinishDrawInAreaEvent::cbFinishDrawInAreaEvent

cbFinishDrawInAreaEvent(**const wxRect&** *area*, **cbDockPane*** *pPane*)

Constructor, taking rectangular area and pane.

cbFloatedBarWindow

cbFloatedBarWindow is a kind of wxToolWindow, implementing floating toolbars.

Derived from

wxToolWindow (p. 94)

Include files

<wx/fl/toolwnd.h>

Data structures

cbFloatedBarWindow::cbFloatedBarWindow

cbFloatedBarWindow()

Default constructor.

cbFloatedBarWindow::GetBar

cbBarInfo* GetBar()

Returns the bar information for this window.

cbFloatedBarWindow::GetPreferredSize

wxSize GetPreferredSize(**const wxSize&** *given*)

Overridden function returning the preferred size.

cbFloatedBarWindow::HandleTitleClick

bool HandleTitleClick(**wxMouseEvent&** *event*)

Overridden function responding to mouse button clicks on the titlebar.

cbFloatedBarWindow::OnDblClick

void OnDblClick(**wxMouseEvent&** *event*)

Responds to double-click mouse events.

cbFloatedBarWindow::OnMiniButtonClicked

void OnMiniButtonClicked(**int** *btnIdx*)

Overridden function responding to mouse clicks on mini-buttons.

cbFloatedBarWindow::PositionFloatedWnd

void PositionFloatedWnd(**int** *scrX*, **int** *scrY*, **int** *width*, **int** *height*)

Position the floating window. The given coordinates are those of the bar itself; the floated container window's position and size are ajusted accordingly.

cbFloatedBarWindow::SetBar

void SetBar(**cbBarInfo*** *pBar*)

Sets the bar information for this window.

cbFloatedBarWindow::SetLayout

void SetLayout(**wxFrameLayout*** *pLayout*)

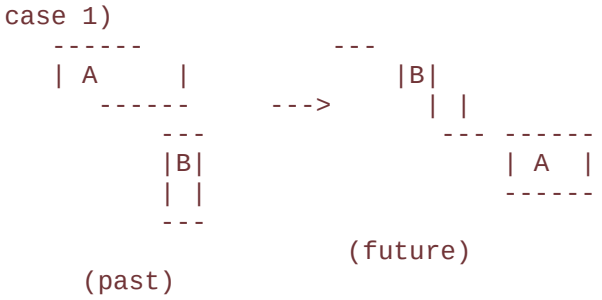
Sets the layout for this window.

cbGCUpdatesMgr

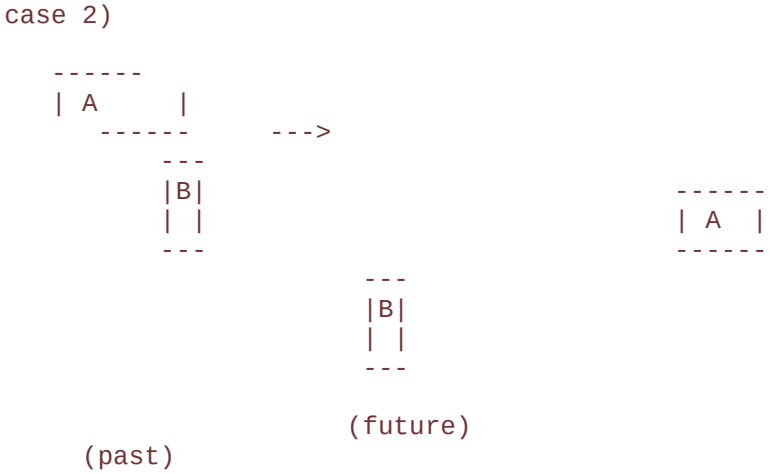
This class implements optimized logic for refreshing the areas of frame layout that actually need to be updated. It is used as the default updates manager by wxFrameLayout.

It is called 'Garbage Collecting' updates manager because its implementation tries to find out dependencies between bars, and to order them into a 'hierarchy'. This hierarchical sorting resembles the implementation of heap-garbage collectors, which resolve dependencies between references.

Example: there are situations where the order in which the user moves windows does matter.



Past/future positions of A and B windows completely overlap, i.e. depend on each other, and there is no solution for moving the windows without refreshing both of them -- we have a cyclic dependency here. The garbage collection algorithm will find this cyclic dependecy and will force refresh after movement.



In this case past/future positions do not overlap, so it is enough only to move windows without refreshing them. Garbage collection will 'notice' this.

There is also a third case, when overlapping is partial. In this case the refreshing can also be avoided by moving windows in the order of 'most-dependant' towards the 'least-dependent'. GC handles this automatically, by sorting windows by their dependency-level (or 'hierarchy').

See garbagec.h for more details of this method; garbagec.h/cpp implement sorting of generic dependencies and does not deal with graphical objects directly.

Summary: garbage collection improves performance when complex or large windows are moved around, by reducing the number of repaints. It also helps to avoid dirty non-client areas of moved windows in some special cases of 'overlapping anomalies'.

Derived from

cbSimpleUpdatesMgr (p. 65)

Include files

<wx/fl/gcupdatesmgr.h>

Data structures

cbGCUpdatesMgr::cbGCUpdatesMgr

cbGCUpdatesMgr()

Default constructor.

cbGCUpdatesMgr(**wxFrameLayout*** *pPanel*)

Constructor, taking a frame layout.

cbGCUpdatesMgr::AddItem

void AddItem(**wxList&** *itemList*, **cbBarInfo*** *pBar*, **cbDockPane*** *pPane*, **wxRect&** *curBounds*, **wxRect&** *prevBounds*)

Internal function for repositioning items.

cbGCUpdatesMgr::DoRepositionItems

void DoRepositionItems(**wxList&** *items*)

Internal function for repositioning items.

cbGCUpdatesMgr::OnStartChanges

void OnStartChanges()

Receives notifications from the frame layout.

cbGCUpdatesMgr::UpdateNow

void UpdateNow()

Refreshes the parts of the frame layout which need an update.

cbHintAnimationPlugin

A plugin to draw animated hints when the user drags a pane.

Derived from

cbPluginBase (p. 51)

Include files

<wx/fl/hintanimpl.h>

Data structures

cbHintAnimationPlugin::cbHintAnimationPlugin

cbHintAnimationPlugin()

Default constructor.

cbHintAnimationPlugin(**wxFrameLayout*** *pPanel*, **int** *paneMask* = *wxALL_PANES*)

Constructor, taking a layout panel and pane mask.

cbHintAnimationPlugin::~cbHintAnimationPlugin

~cbHintAnimationPlugin()

Destructor.

cbHintAnimationPlugin::DoDrawHintRect

void DoDrawHintRect(**wxRect&** *rect*, **bool** *isInClientRect*)

Internal function for drawing a hint rectangle.

cbHintAnimationPlugin::DrawHintRect

void DrawHintRect(**wxRect&** *rect*, **bool** *isInClientRect*)

Internal function for drawing a hint rectangle.

cbHintAnimationPlugin::EraseHintRect

void EraseHintRect(**wxRect&** *rect*, **bool** *isInClientRect*)

Internal function for erasing a hint rectangle.

cbHintAnimationPlugin::FinishTracking

void FinishTracking()

Internal function for finishing tracking.

cbHintAnimationPlugin::OnDrawHintRect

void OnDrawHintRect(**cbDrawHintRectEvent&** *event*)

Event handler respoding to hint draw events.

cbHintAnimationPlugin::RectToScr

void RectToScr(**wxRect&** *frameRect*, **wxRect&** *scrRect*)

Internal function for translating coordinates.

cbHintAnimationPlugin::StartTracking

void StartTracking()

speed is constant. Default: TRUE TBD:: get/set methods for above members Internal function for starting tracking.

cbInsertBarEvent

Class for bar insertion events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbInsertBarEvent::cbInsertBarEvent

cbInsertBarEvent(**cbBarInfo*** *pBar*, **cbRowInfo*** *pIntoRow*, **cbDockPane*** *pPane*)

Constructor, taking bar information, row information, and pane.

cbLayoutRowEvent

Class for single row layout events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbLayoutRowEvent::cbLayoutRowEvent

cbLayoutRowEvent(**cbRowInfo*** *pRow*, **cbDockPane*** *pPane*)

Constructor, taking row information and pane.

cbLayoutRowsEvent

Class for multiple rows layout events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbLayoutRowsEvent::cbLayoutRowsEvent

cbLayoutRowsEvent(**cbDockPane*** *pPane*)

Constructor, taking pane.

cbLeftDClickEvent

Class for mouse left double click events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbLeftDClickEvent::cbLeftDClickEvent

cbLeftDClickEvent(**const wxPoint&** *pos*, **cbDockPane*** *pPane*)

Constructor, taking mouse position and pane.

cbLeftDownEvent

Class for mouse left down events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbLeftDownEvent::cbLeftDownEvent

cbLeftDownEvent(**const wxPoint&** *pos*, **cbDockPane*** *pPane*)

Constructor, taking mouse position and pane.

cbLeftUpEvent

Class for mouse left up events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbLeftUpEvent::cbLeftUpEvent

cbLeftUpEvent(**const wxPoint&** *pos*, **cbDockPane*** *pPane*)

Constructor, taking mouse position and pane.

cbMiniButton

cbMiniButton is the base class for a small button that can be placed in a wxToolWindow titlebar.

Derived from

wxObject

Include files

<wx/fl/toolwnd.h>

Data structures

cbMiniButton::cbMiniButton

cbMiniButton()

Default constructor.

cbMiniButton::Draw

void Draw(**wxDC&** *dc*)

Draws the button. Override this to implement the desired appearance.

cbMiniButton::Enable

void Enable(**bool** *enable*)

Enable or disable the button.

cbMiniButton::HitTest

bool HitTest(**const wxPoint&** *pos*)

Returns TRUE if the given position was over the button.

cbMiniButton::IsPressed

bool IsPressed()

Returns TRUE if this button is pressed.

cbMiniButton::OnLeftDown

void OnLeftDown(**const wxPoint&** *pos*)

Responds to a left down event.

cbMiniButton::OnLeftUp

void OnLeftUp(**const wxPoint&** *pos*)

Responds to a left up event.

cbMiniButton::OnMotion

void OnMotion(**const wxPoint&** *pos*)

Responds to a mouse move event.

cbMiniButton::Refresh

void Refresh()

Refreshes the button.

cbMiniButton::Reset

void Reset()

Reset the button.

cbMiniButton::SetPos

void SetPos(**const wxPoint&** *pos*)

Set the position of the button.

cbMiniButton::WasClicked

bool WasClicked()

Returns TRUE if the button was clicked.

cbMotionEvent

Class for mouse motion events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbMotionEvent::cbMotionEvent

cbMotionEvent(**const wxPoint&** *pos*, **cbDockPane*** *pPane*)

Constructor, taking mouse position and pane.

cbPaneDrawPlugin

Simple but all-in-one plugin implementation. Resembles the look and feel of to MFC control-bars. The class handles painting of the pane and the items in it; it generates bar/layout customization events, when the user right-clicks the bar/pane. Hooking an instance of this and row-layout plugins for each pane would be enough for the frame layout to function properly (they are plugged in automatically by the wxFrameLayout class).

Derived from

cbPluginBase (p. 51)

Include files

<wx/fl/panedrawpl.h>

Data structures

cbPaneDrawPlugin::cbPaneDrawPlugin

cbPaneDrawPlugin(**wxFrameLayout*** *pPanel*, **int** *paneMask* = *wxALL_PANES*)

Constructor taking frame layout pane and a pane mask.

cbPaneDrawPlugin()

Default constructor.

cbPaneDrawPlugin::~cbPaneDrawPlugin

~cbPaneDrawPlugin()

Destructor.

cbPaneDrawPlugin::Clone

cbPluginBase* Clone()

Clone function, returning a new instance of this class.

cbPaneDrawPlugin::DrawBarInnerShadeRect

void DrawBarInnerShadeRect(**cbBarInfo*** *pBar*, **wxDC&** *dc*)

Internal helper: draws the inner bar shading.

cbPaneDrawPlugin::DrawDraggedHandle

void DrawDraggedHandle(**const wxPoint&** *pos*, **cbDockPane&** *pane*)

Internal helper: draws the dragged handle.

cbPaneDrawPlugin::DrawLowerRowHandle

void DrawLowerRowHandle(**cbRowInfo*** *pRow*, **wxDC&** *dc*)

Internal helper: draws the lower row handle.

cbPaneDrawPlugin::DrawLowerRowShades

void DrawLowerRowShades(**cbRowInfo*** *pRow*, **wxDC&** *dc*, **int** *level*)

Internal helper: draws the lower row shading.

cbPaneDrawPlugin::DrawPaneShade

void DrawPaneShade(**wxDC&** *dc*, **int** *alignment*)

Internal helper: draws the pane shading.

cbPaneDrawPlugin::DrawPaneShadeForRow

void DrawPaneShadeForRow(**cbRowInfo*** *pRow*, **wxDC&** *dc*)

Internal helper: draws the pane shading for a row.

cbPaneDrawPlugin::DrawShade

void DrawShade(**int** *level*, **wxRect&** *rect*, **int** *alignment*, **wxDC&** *dc*)

Internal helper: draws shading.

cbPaneDrawPlugin::DrawShade1

void DrawShade1(**int** *level*, **wxRect&** *rect*, **int** *alignment*, **wxDC&** *dc*)

Internal helper: draws shading.

cbPaneDrawPlugin::DrawUpperRowHandle

void DrawUpperRowHandle(**cbRowInfo*** *pRow*, **wxDC&** *dc*)

Internal helper: draws the upper row handle.

cbPaneDrawPlugin::DrawUpperRowShades

void DrawUpperRowShades(**cbRowInfo*** *pRow*, **wxDC&** *dc*, **int** *level*)

Internal helper: draws the upper row shading.

cbPaneDrawPlugin::OnDrawBarDecorations

void OnDrawBarDecorations(**cbDrawBarDecorEvent&** *event*)

Handler for draw bar decorations events.

cbPaneDrawPlugin::OnDrawBarHandles

void OnDrawBarHandles(**cbDrawBarHandlesEvent&** *event*)

Handler for draw bar handles events.

cbPaneDrawPlugin::OnDrawPaneBackground

void OnDrawPaneBackground(**cbDrawPaneBkGroundEvent&** *event*)

Handler for draw pane background events.

cbPaneDrawPlugin::OnDrawPaneDecorations

void OnDrawPaneDecorations(**cbDrawPaneDecorEvent&** *event*)

Handler for draw pane decoration events.

cbPaneDrawPlugin::OnDrawRowBackground

void OnDrawRowBackground(**cbDrawRowBkGroundEvent&** *event*)

Handler for draw row background events.

cbPaneDrawPlugin::OnDrawRowDecorations

void OnDrawRowDecorations(**cbDrawRowDecorEvent&** *event*)

Handler for draw row decoration events.

cbPaneDrawPlugin::OnDrawRowHandles

void OnDrawRowHandles(**cbDrawRowHandlesEvent&** *event*)

Handler for draw row handles events.

cbPaneDrawPlugin::OnFinishDrawInArea

void OnFinishDrawInArea(**cbFinishDrawInAreaEvent&** *event*)

Handler for finish draw in area events.

cbPaneDrawPlugin::OnLButtonDown

void OnLButtonDown(**cbLeftDownEvent&** *event*)

Handler for left mouse button down events.

cbPaneDrawPlugin::OnLButtonUp

void OnLButtonUp(**cbLeftUpEvent&** *event*)

Handler for left mouse button up events.

cbPaneDrawPlugin::OnLDblClick

void OnLDblClick(**cbLeftDClickEvent&** *event*)

Handler for left double-click mouse button down events.

cbPaneDrawPlugin::OnMouseMove

void OnMouseMove(**cbMotionEvent&** *event*)

Handler for mouse move events.

cbPaneDrawPlugin::OnRButtonUp

void OnRButtonUp(**cbRightUpEvent&** *event*)

Handler for right mouse button up events.

cbPaneDrawPlugin::OnSizeBarWindow

void OnSizeBarWindow(**cbSizeBarWndEvent&** *event*)

Handler for bar size events.

cbPaneDrawPlugin::OnStartDrawInArea

void OnStartDrawInArea(**cbStartDrawInAreaEvent&** *event*)

Handler for start draw in area events.

cbPaneDrawPlugin::SetDarkPixel

void SetDarkPixel(**int** *x*, **int** *y*, **wxDC&** *dc*)

Internal helper: sets a dark pixel at the given location.

cbPaneDrawPlugin::SetLightPixel

void SetLightPixel(**int** *x*, **int** *y*, **wxDC&** *dc*)

Internal helper: sets a light pixel at the given location.

cbPluginBase

Abstract base class for all control-bar related plugins. Note: pointer positions of mouse events sent to plugins are always in the pane's coordinates (the pane to which this plugin is hooked).

Derived from

wxEvtHandler

Include files

<wx/fl/controlbar.h>

Data structures

cbPluginBase::cbPluginBase

cbPluginBase(**wxFrameLayout*** *pPanel*, **int** *paneMask* = *wxALL_PANES*)

Constructor taking layout panel and a mask.

cbPluginBase()

Default constructor.

cbPluginBase::~cbPluginBase

~cbPluginBase()

Destructor. Destroys the whole plugin chain of connected plugins.

cbPluginBase::GetPaneMask

int GetPaneMask()

Returns the pane mask.

cbPluginBase::IsReady

bool IsReady()

Returns TRUE if the plugin is ready to receive events.

cbPluginBase::OnInitPlugin

void OnInitPlugin()

Override this method to do plugin-specific initialization. At this point plugin is already attached to the frame layout, and pane masks are set.

cbPluginBase::ProcessEvent

bool ProcessEvent(**wxEvent&** *event*)

Overridden to determine whether the target pane specified in the event matches the pane mask of this plugin (specific plugins do not override this method).

cbPluginEvent

Base class for all control-bar plugin events. This is not a dynamically-creatable class.

Derived from

wxEvent

Include files

<wx/fl/controlbar.h>

Data structures

cbPluginEvent::cbPluginEvent

cbPluginEvent(**wxEventType** *eventType*, **cbDockPane*** *pPane*)

Constructor, taking event type and pane.

cbPluginEvent::Clone

wxEvent* Clone() **const**

Not used, but required.

cbRemoveBarEvent

Class for bar removal events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbRemoveBarEvent::cbRemoveBarEvent

cbRemoveBarEvent(**cbBarInfo*** *pBar*, **cbDockPane*** *pPane*)

Constructor, taking bar information and pane.

cbResizeBarEvent

Class for bar resize events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbResizeBarEvent::cbResizeBarEvent

cbResizeBarEvent(**cbBarInfo*** *pBar*, **cbRowInfo*** *pRow*, **cbDockPane*** *pPane*)

Constructor, taking bar information, row information, and pane.

cbResizeRowEvent

Class for row resize events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbResizeRowEvent::cbResizeRowEvent

cbResizeRowEvent(**cbRowInfo*** *pRow*, **int** *handleOfs*, **bool** *forUpperHandle*, **cbDockPane*** *pPane*)

Constructor, taking row information, two parameters of currently unknown use, and pane.

cbRightDownEvent

Class for mouse right down events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbRightDownEvent::cbRightDownEvent

cbRightDownEvent(**const wxPoint&** *pos*, **cbDockPane*** *pPane*)

Constructor, taking mouse position and pane.

cbRightUpEvent

Class for mouse right up events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbRightUpEvent::cbRightUpEvent

cbRightUpEvent(**const wxPoint&** *pos*, **cbDockPane*** *pPane*)

Constructor, taking mouse position and pane.

cbRowDragPlugin

This plugin adds row-dragging functionality to the pane. It handles mouse movement and pane background-erasing plugin events. The behaviour and appearance resembles drag and drop positioning of the toolbar rows in Netscape Communicator 4.xx.

Derived from

cbPluginBase (p. 51)

Include files

<wx/fl/rowdragpl.h>

Data structures

cbRowDragPlugin::cbRowDragPlugin

cbRowDragPlugin(**wxFrameLayout*** *pLayout*, **int** *paneMask* = *wxALL_PANES*)

Constructor, taking paren layout frame and pane mask.

cbRowDragPlugin()

Default constructor.

cbRowDragPlugin::~cbRowDragPlugin

~cbRowDragPlugin()

Destructor.

cbRowDragPlugin::CaptureDCArea

wxBitmap* CaptureDCArea(**wxDC&** *dc*, **wxRect&** *area*)

Helper for drag and drop.

cbRowDragPlugin::CheckPrevItemInFocus

void CheckPrevItemInFocus(**cbRowInfo*** *pRow*, **int** *iconIdx*)

Helper for drag and drop.

cbRowDragPlugin::Clone

cbPluginBase* Clone()

Clone function, returning a new instance of this class.

cbRowDragPlugin::CollapseRow

void CollapseRow(**cbRowInfo*** *pRow*)

Helper for drag and drop.

cbRowDragPlugin::Draw3DPattern

void Draw3DPattern(**wxRect&** *inRect*, **wxDC&** *dc*)

Implements 'hard-coded metafile' for Netscape Navigator look.

cbRowDragPlugin::Draw3DRect

void Draw3DRect(**wxRect&** *inRect*, **wxDC&** *dc*, **wxBrush&** *bkBrush*)

Implements 'hard-coded metafile' for Netscape Navigator look.

cbRowDragPlugin::DrawCollapsedRowIcon

void DrawCollapsedRowIcon(**int** *index*, **wxDC&** *dc*, **bool** *isHighlighted*)

Draws collapsed row icon (appearance-dependent).

cbRowDragPlugin::DrawCollapsedRowsBorder

void DrawCollapsedRowsBorder(**wxDC&** *dc*)

Draws collapsed rows border (appearance-dependent).

cbRowDragPlugin::DrawEmptyRow

void DrawEmptyRow(**wxDC&** *dc*, **wxRect&** *rowBounds*)

Draws empty row (appearance-dependent).

cbRowDragPlugin::DrawOrtoRomb

void DrawOrtoRomb(**wxRect&** *inRect*, **wxDC&** *dc*, **wxBrush&** *bkBrush*)

Implements 'hard-coded metafile' for Netscape Navigator look.

cbRowDragPlugin::DrawRectShade

void DrawRectShade(**wxRect&** *inRect*, **wxDC&** *dc*, **int** *level*, **wxPen&** *upperPen*, **wxPen&** *lowerPen*)

Implements 'hard-coded metafile' for Netscape Navigator look.

cbRowDragPlugin::DrawRomb

void DrawRomb(**wxRect&** *inRect*, **wxDC&** *dc*, **wxBrush&** *bkBrush*)

Implements 'hard-coded metafile' for Netscape Navigator look.

cbRowDragPlugin::DrawRombShades

void DrawRombShades(**wxPoint&** *p1*, **wxPoint&** *p2*, **wxPoint&** *p3*, **wxPoint&** *p4*, **wxDC&** *dc*)

Implements 'hard-coded metafile' for Netscape Navigator look.

cbRowDragPlugin::DrawRowDragHint

void DrawRowDragHint(**cbRowInfo*** *pRow*, **wxDC&** *dc*, **bool** *isHighlighted*)

Draws row drag hint (appearance-dependent).

cbRowDragPlugin::DrawRowsDragHintsBorder

void DrawRowsDragHintsBorder(**wxDC&** *dc*)

Draws rows drag hints border (appearance-dependent).

cbRowDragPlugin::DrawTrianDown

void DrawTrianDown(**wxRect&** *inRect*, **wxDC&** *dc*)

Implements 'hard-coded metafile' for Netscape Navigator look.

cbRowDragPlugin::DrawTrianRight

void DrawTrianRight(**wxRect&** *inRect*, **wxDC&** *dc*)

Implements 'hard-coded metafile' for Netscape Navigator look.

cbRowDragPlugin::DrawTrianUp

void DrawTrianUp(**wxRect&** *inRect*, **wxDC&** *dc*)

Implements 'hard-coded metafile' for Netscape Navigator look.

cbRowDragPlugin::ExpandRow

void ExpandRow(**int** *collapsedIconIdx*)

Helper for drag and drop.

cbRowDragPlugin::FinishOnScreenDraw

void FinishOnScreenDraw()

Helper for drag and drop.

cbRowDragPlugin::GetCollapsedIconsPos

int GetCollapsedIconsPos()

Helper for drag and drop.

cbRowDragPlugin::GetCollapsedInconRect

void GetCollapsedInconRect(**int** *iconIdx*, **wxRect&** *rect*)

Helper for drag and drop.

cbRowDragPlugin::GetCollapsedRowIconHeight

int GetCollapsedRowIconHeight()

Gets the collapsed row icon height.

cbRowDragPlugin::GetFirstRow

cbRowInfo* GetFirstRow()

Helper for drag and drop.

cbRowDragPlugin::GetHRowsCountForPane

int GetHRowsCountForPane(**cbDockPane*** *pPane*)

Helper for drag and drop.

cbRowDragPlugin::GetRowDragHintWidth

int GetRowDragHintWidth()

Gets the row drag hint width.

cbRowDragPlugin::GetRowHintRect

void GetRowHintRect(**cbRowInfo*** *pRow*, **wxRect&** *rect*)

Helper for drag and drop.

cbRowDragPlugin::HitTestCollapsedRowIcon

bool HitTestCollapsedRowIcon(**int** *iconIdx*, **const wxPoint&** *pos*)

Test for the collapsed row icon position.

cbRowDragPlugin::HitTestRowDragHint

bool HitTestRowDragHint(**cbRowInfo*** *pRow*, **const wxPoint&** *pos*)

Test for the row drag hint position.

cbRowDragPlugin::InsertDraggedRowBefore

void InsertDraggedRowBefore(**cbRowInfo*** *pBeforeRow*)

Helper for drag and drop.

cbRowDragPlugin::ItemIsInFocus

bool ItemIsInFocus()

Helper for drag and drop.

cbRowDragPlugin::OnDrawPaneBackground

void OnDrawPaneBackground(**cbDrawPaneDecorEvent&** *event*)

Handles pane drawing plugin events (appearance-independent logic).

cbRowDragPlugin::OnInitPlugin

void OnInitPlugin()

Called to initialize this plugin.

cbRowDragPlugin::OnLButtonDown

void OnLButtonDown(**cbLeftDownEvent&** *event*)

Handles left button down plugin events (appearance-independent logic).

cbRowDragPlugin::OnLButtonUp

void OnLButtonUp(**cbLeftUpEvent&** *event*)

Handles left button up plugin events (appearance-independent logic).

cbRowDragPlugin::OnMouseMove

void OnMouseMove(**cbMotionEvent&** *event*)

Handles mouse move plugin events (appearance-independent logic).

cbRowDragPlugin::PrepareForRowDrag

void PrepareForRowDrag()

Helper for drag and drop.

cbRowDragPlugin::SetMouseCapture

void SetMouseCapture(**bool** *captureOn*)

Helper for drag and drop.

cbRowDragPlugin::SetPaneMargins

void SetPaneMargins()

Sets the pane margins.

cbRowDragPlugin::ShowDraggedRow

void ShowDraggedRow(**int** *offset*)

Helper for drag and drop.

cbRowDragPlugin::ShowPaneImage

void ShowPaneImage()

Helper for drag and drop.

cbRowDragPlugin::UnhighlightItemInFocus

void UnhighlightItemInFocus()

Helper for drag and drop.

cbRowInfo

Helper class used internally by the wxFrameLayout class. Holds and manages information about bar rows.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

cbRowInfo::cbRowInfo

cbRowInfo()

Constructor.

cbRowInfo::~cbRowInfo

~cbRowInfo()

Destructor.

cbRowInfo::GetFirstBar

cbBarInfo* GetFirstBar()

Returns the first bar.

cbRowLayoutPlugin

Simple implementation of a plugin which handles row layout requests sent from a frame layout.

Derived from

cbPluginBase (p. 51)

Include files

<wx/fl/rowlayoutpl.h>

Data structures

cbRowLayoutPlugin::cbRowLayoutPlugin

cbRowLayoutPlugin(**wxFrameLayout*** *pPanel*, **int** *paneMask* = *wxALL_PANES*)

Constructor taking frame layout pane and pane mask.

cbRowLayoutPlugin()

Default constructor.

cbRowLayoutPlugin::AdjustLengthOfInserted

void AdjustLengthOfInserted(**cbRowInfo*** *pRow*, **cbBarInfo*** *pTheBar*)

Internal helper relating to not-fixed-bars layout.

cbRowLayoutPlugin::ApplyLengthRatios

void ApplyLengthRatios(**cbRowInfo*** *pRow*)

Internal helper relating to not-fixed-bars layout.

cbRowLayoutPlugin::CalcRowHeight

int CalcRowHeight(cbRowInfo& *row*)

Row layout helper simulating bar 'friction'.

cbRowLayoutPlugin::CheckIfAtTheBoundary

void CheckIfAtTheBoundary(cbBarInfo* *pTheBar*, cbRowInfo& *rowInfo*)

Internal helper relating to not-fixed-bars layout.

cbRowLayoutPlugin::DetectBarHandles

void DetectBarHandles(cbRowInfo* *pRow*)

Internal helper relating to not-fixed-bars layout.

cbRowLayoutPlugin::DoInsertBar

void DoInsertBar(cbBarInfo* *pTheBar*, cbRowInfo& *row*)

Insert the bar before the given row.

cbRowLayoutPlugin::ExpandNotFixedBars

void ExpandNotFixedBars(cbRowInfo* *pRow*)

Internal helper relating to not-fixed-bars layout.

cbRowLayoutPlugin::FitBarsToRange

void FitBarsToRange(int *from*, int *till*, cbBarInfo* *pTheBar*, cbRowInfo* *pRow*)

Internal helper relating to not-fixed-bars layout.

cbRowLayoutPlugin::GetRowFreeSpace

int GetRowFreeSpace(cbRowInfo* *pRow*)

Internal helper relating to not-fixed-bars layout.

cbRowLayoutPlugin::InsertBefore

void InsertBefore(cbBarInfo* *pBeforeBar*, cbBarInfo* *pTheBar*, cbRowInfo& *row*)

Insert the bar before the given row.

cbRowLayoutPlugin::LayoutItemsVertically

void LayoutItemsVertically(cbRowInfo& *row*)

Row layout helper simulating bar 'friction'.

cbRowLayoutPlugin::MinimzeNotFixedBars

void MinimzeNotFixedBars(**cbRowInfo*** *pRow*, **cbBarInfo*** *pBarToPreserve*)

Internal helper relating to not-fixed-bars layout.

cbRowLayoutPlugin::OnInsertBar

void OnInsertBar(**cbInsertBarEvent&** *event*)

Responds to bar insertion event.

cbRowLayoutPlugin::OnLayoutRow

void OnLayoutRow(**cbLayoutRowEvent&** *event*)

Responds to row layout event.

cbRowLayoutPlugin::OnLayoutRows

void OnLayoutRows(**cbLayoutRowsEvent&** *event*)

Responds to rows layout event.

cbRowLayoutPlugin::OnRemoveBar

void OnRemoveBar(**cbRemoveBarEvent&** *event*)

Responds to bar removal event.

cbRowLayoutPlugin::OnResizeRow

void OnResizeRow(**cbResizeRowEvent&** *event*)

Responds to row resize event.

cbRowLayoutPlugin::RecalcLengthRatios

void RecalcLengthRatios(**cbRowInfo*** *pRow*)

Internal helper relating to not-fixed-bars layout.

cbRowLayoutPlugin::RelayoutNotFixedBarsAround

void RelayoutNotFixedBarsAround(**cbBarInfo*** *pTheBar*, **cbRowInfo*** *pRow*)

Internal helper relating to not-fixed-bars layout.

cbRowLayoutPlugin::ShiftLeftTrashold

void ShiftLeftTrashold(**cbBarInfo*** *pTheBar*, **cbRowInfo&** *row*)

Row layout helper simulating bar 'friction'.

cbRowLayoutPlugin::ShiftRightTrashold

void ShiftRightTrashold(**cbBarInfo*** *pTheBar*, **cbRowInfo&** *row*)

Row layout helper simulating bar 'friction'.

cbRowLayoutPlugin::SlideLeftSideBars

void SlideLeftSideBars(**cbBarInfo*** *pTheBar*)

Row layout helper simulating bar 'friction'.

cbRowLayoutPlugin::SlideRightSideBars

void SlideRightSideBars(**cbBarInfo*** *pTheBar*)

Row layout helper simulating bar 'friction'.

cbRowLayoutPlugin::StickRightSideBars

void StickRightSideBars(**cbBarInfo*** *pToBar*)

Row layout helper simulating bar 'friction'.

cbSimpleCustomizationPlugin

This class enables customization of a bar, popping up a menu and handling basic customization such as floating and horizontal/vertical alignment of the bar.

Derived from

cbPluginBase (p. 51)

Include files

<wx/fl/cbcustom.h>

Data structures

cbSimpleCustomizationPlugin::cbSimpleCustomizationPlugin

cbSimpleCustomizationPlugin(**wxFrameLayout*** *pPanel*, **int** *paneMask* = *wxALL_PANES*)

Constructor, taking parent pane and a pane mask flag.

cbSimpleCustomizationPlugin()

Default constructor.

cbSimpleCustomizationPlugin::OnCustomizeBar

void OnCustomizeBar(**cbCustomizeBarEvent&** *event*)

Plugin event handler for cbCustomizeBarEvent.

cbSimpleCustomizationPlugin::OnCustomizeLayout

void OnCustomizeLayout(**cbCustomizeLayoutEvent&** *event*)

Plugin event handler for cbCustomizeLayoutEvent.

cbSimpleCustomizationPlugin::OnMenuItemSelected

void OnMenuItemSelected(**wxCommandEvent&** *event*)

Menu event handler.

cbSimpleUpdatesMgr

This class implements slightly optimized logic for refreshing the areas of frame layout that actually need to be updated.

Derived from

cbUpdatesManagerBase (p. 69)

Include files

<wx/fl/updatesmgr.h>

Data structures

cbSimpleUpdatesMgr::cbSimpleUpdatesMgr

cbSimpleUpdatesMgr()

Default constructor.

cbSimpleUpdatesMgr(**wxFrameLayout*** *pPanel*)

Constructor taking frame layout panel.

cbSimpleUpdatesMgr::OnBarWillChange

void OnBarWillChange(**cbBarInfo*** *pBar*, **cbRowInfo*** *pInRow*, **cbDockPane*** *pInPane*)

Notification received from Frame Layout in the order in which they would usually be invoked.

cbSimpleUpdatesMgr::OnFinishChanges

void OnFinishChanges()

Notification received from Frame Layout in the order in which they would usually be invoked.

cbSimpleUpdatesMgr::OnPaneMarginsWillChange

void OnPaneMarginsWillChange(**cbDockPane*** *pPane*)

Notification received from Frame Layout in the order in which they would usually be invoked.

cbSimpleUpdatesMgr::OnPaneWillChange

void OnPaneWillChange(**cbDockPane*** *pPane*)

Notification received from Frame Layout in the order in which they would usually be invoked.

cbSimpleUpdatesMgr::OnRowWillChange

void OnRowWillChange(**cbRowInfo*** *pRow*, **cbDockPane*** *pInPane*)

Notification received from Frame Layout in the order in which they would usually be invoked.

cbSimpleUpdatesMgr::OnStartChanges

void OnStartChanges()

Notification received from Frame Layout in the order in which they would usually be invoked.

cbSimpleUpdatesMgr::UpdateNow

void UpdateNow()

Refreshes the parts of the frame layoutthat need an update.

cbSimpleUpdatesMgr::WasChanged

bool WasChanged(**cbUpdateMgrData&** *data*, **wxRect&** *currentBounds*)

Helper function.

cbSizeBarWndEvent

Class for bar window resize events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbSizeBarWndEvent::cbSizeBarWndEvent

cbSizeBarWndEvent(**cbBarInfo*** *pBar*, **cbDockPane*** *pPane*)

Constructor, taking bar information and pane.

cbStartBarDraggingEvent

Class for start-bar-dragging events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbStartBarDraggingEvent::cbStartBarDraggingEvent

cbStartBarDraggingEvent(**cbBarInfo*** *pBar*, **const wxPoint&** *pos*, **cbDockPane*** *pPane*)

Constructor, taking bar information, mouse position, and pane.

cbStartDrawInAreaEvent

Class for start drawing in area events.

Derived from

cbPluginEvent (p. 52)

Include files

<wx/fl/controlbar.h>

Data structures

cbStartDrawInAreaEvent::cbStartDrawInAreaEvent

cbStartDrawInAreaEvent(**const wxRect&** *area*, **wxDC**** *ppDCForArea*, **cbDockPane*** *pPane*)

to the obtained buffer-context should be placed Constructor, taking rectangular area, device context pointer to a pointer, and pane.

cbUpdateMgrData

A structure that is present in each item of layout, used by any particular updates-manager to store auxiliary information to be used by its updating algorithm.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

cbUpdateMgrData::cbUpdateMgrData

cbUpdateMgrData()

Default constructor. Is-dirty flag is set TRUE initially.

cbUpdateMgrData::IsDirty

bool IsDirty()

Returns the is-dirty flag.

cbUpdateMgrData::SetCustomData

void SetCustomData(**wxObject*** *pCustomData*)

Set custom data.

cbUpdateMgrData::SetDirty

void SetDirty(**bool** *isDirty* = *TRUE*)

Set the dirty flag.

cbUpdateMgrData::StoreItemState

void StoreItemState(**const wxRect&** *boundsInParent*)

Store the item state.

cbUpdatesManagerBase

This class declares an abstract interface for optimized logic that should refresh areas of frame layout that actually need to be updated. This should be extended in future to implement a custom updating strategy.

Derived from

wxObject

Include files

<wx/fl/controlbar.h>

Data structures

cbUpdatesManagerBase::cbUpdatesManagerBase

cbUpdatesManagerBase(**wxFrameLayout*** *pPanel*)

Constructor taking layout panel.

cbUpdatesManagerBase()

Default constructor

cbUpdatesManagerBase::~cbUpdatesManagerBase

~cbUpdatesManagerBase()

Destructor.

cbUpdatesManagerBase::OnBarWillChange

void OnBarWillChange(**cbBarInfo*** *pBar*, **cbRowInfo*** *pInRow*, **cbDockPane*** *pInPane*)

This function receives a notification from the frame layout (in the order in which they would usually be invoked). Custom updates-managers may utilize these notifications to implement a more fine-grained updating strategy.

cbUpdatesManagerBase::OnFinishChanges

void OnFinishChanges()

This function receives a notification from the frame layout (in the order in which they would usually be invoked). Custom updates-managers may utilize these notifications to implement a more fine-grained updating strategy.

cbUpdatesManagerBase::OnPaneMarginsWillChange

void OnPaneMarginsWillChange(**cbDockPane*** *pPane*)

This function receives a notification from the frame layout (in the order in which they would usually be invoked). Custom updates-managers may utilize these notifications to implement a more fine-grained updating strategy.

cbUpdatesManagerBase::OnPaneWillChange

void OnPaneWillChange(**cbDockPane*** *pPane*)

This function receives a notification from the frame layout (in the order in which they would usually be invoked). Custom updates-managers may utilize these notifications to implement a more fine-grained updating strategy.

cbUpdatesManagerBase::OnRowWillChange

void OnRowWillChange(**cbRowInfo*** *pRow*, **cbDockPane*** *pInPane*)

This function receives a notification from the frame layout (in the order in which they would usually be invoked). Custom updates-managers may utilize these notifications to implement a more fine-grained updating strategy.

cbUpdatesManagerBase::OnStartChanges

void OnStartChanges()

This function receives a notification from the frame layout (in the order in which they would usually be invoked). Custom updates-managers may utilize these notifications to implement a more fine-grained updating strategy.

cbUpdatesManagerBase::SetLayout

void SetLayout(**wxFrameLayout*** *pLayout*)

Sets the associated layout.

cbUpdatesManagerBase::UpdateNow

void UpdateNow()

Refreshes parts of the frame layout that need an update.

wxDynamicToolBar

wxDynamicToolBar manages containment and layout of tool windows.

Derived from

wxToolBarBase

Include files

<wx/fl/dyntbar.h>

Data structures

wxDynamicToolBar::wxDynamicToolBar

wxDynamicToolBar()

Default constructor.

wxDynamicToolBar(**wxWindow*** *parent*, **const wxWindowID** *id*, **const wxPoint&** *pos = wxDefaultPosition*, **const wxSize&** *size = wxDefaultSize*, **const long** *style = wxNO_BORDER*, **const int** *orientation = wxVERTICAL*, **const int** *RowsOrColumns = 1*, **const wxString&** *name = wxToolBarNameStr*)

Constructor: see the documentation for wxToolBar for details.

wxDynamicToolBar::~wxDynamicToolBar

~wxDynamicToolBar()

Destructor.

wxDynamicToolBar::AddSeparator

void AddSeparator(**wxWindow*** *pSepartorWnd = NULL*)

Adds a separator. See the documentation for wxToolBar for details.

wxDynamicToolBar::AddTool

void AddTool(**int** *toolIndex*, **wxWindow*** *pToolWindow*, **const wxSize&** *size = wxDefaultSize*)

Adds a tool. See the documentation for wxToolBar for details.

void AddTool(**int** *toolIndex*, **const wxString&** *imageFileName*, **wxBitmapType** *imageFileType = wxBITMAP_TYPE_BMP*, **const wxString&** *labelText = ""*, **bool** *alignTextRight = FALSE*, **bool** *isFlat = TRUE*)

Adds a tool. See the documentation for wxToolBar for details.

void AddTool(**int** *toolIndex*, **wxBitmap** *labelBmp*, **const wxString&** *labelText = ""*, **bool** *alignTextRight = FALSE*, **bool** *isFlat = TRUE*)

Adds a tool. See the documentation for wxToolBar for details.

wxToolBarToolBase* AddTool(**const int** *toolIndex*, **const wxBitmap&** *bitmap*, **const wxBitmap&** *pushedBitmap = wxNullBitmap*, **const bool** *toggle = FALSE*, **const long** *xPos = -1*, **const long** *yPos = -1*, **wxObject*** *clientData = NULL*, **const wxString&** *helpString1 = ""*, **const wxString&** *helpString2 = ""*)

Method from wxToolBarBase (for compatibility), only the first two arguments are valid. See the documentation for wxToolBar for details.

wxDynamicToolBar::Create

bool Create(**wxWindow*** *parent*, **const wxWindowID** *id*, **const wxPoint&** *pos = wxDefaultPosition*, **const wxSize&** *size = wxDefaultSize*, **const long** *style = wxNO_BORDER*, **const int** *orientation = wxVERTICAL*, **const int** *RowsOrColumns = 1*, **const wxString&** *name = wxToolBarNameStr*)

Creation function: see the documentation for wxToolBar for details.

wxDynamicToolBar::CreateDefaultLayout

LayoutManagerBase* CreateDefaultLayout()

Creates the default layout (BagLayout).

wxDynamicToolBar::CreateTool

wxToolBarToolBase* CreateTool(**wxControl*** *control*)

Creates a toolbar tool.

wxToolBarToolBase* CreateTool(**int** *id*, **const wxBitmap&** *bitmap1*, **const wxBitmap&** *bitmap2*, **bool** *toggle*, **wxObject*** *clientData*, **const wxString&** *shortHelpString*, **const wxString&** *longHelpString*)

Creates a toolbar tool.

wxDynamicToolBar::DoDeleteTool

bool DoDeleteTool(**size_t** *pos*, **wxToolBarToolBase*** *tool*)

Deletes a tool. The tool is still in m_tools list when this function is called, and it will only be deleted from it if it succeeds.

wxDynamicToolBar::DoEnableTool

void DoEnableTool(**wxToolBarToolBase*** *tool*, **bool** *enable*)

Called when the tools enabled flag changes.

wxDynamicToolBar::DoInsertTool

bool DoInsertTool(**size_t** *pos*, **wxToolBarToolBase*** *tool*)

Inserts a tool at the given position.

wxDynamicToolBar::DoSetToggle

void DoSetToggle(**wxToolBarToolBase*** *tool*, **bool** *toggle*)

Called when the tools 'can be toggled' flag changes.

wxDynamicToolBar::DoToggleTool

void DoToggleTool(**wxToolBarToolBase*** *tool*, **bool** *toggle*)

Called when the tool is toggled.

wxDynamicToolBar::DrawSeparator

void DrawSeparator(**wxDynToolInfo&** *info*, **wxDC&** *dc*)

Draws a separator. The default implementation draws a shaded line.

wxDynamicToolBar::EnableTool

void EnableTool(**const int** *toolIndex*, **const bool** *enable* = *TRUE*)

Enables or disables the given tool.

wxDynamicToolBar::FindToolForPosition

wxToolBarToolBase* FindToolForPosition(**wxCoord** *x*, **wxCoord** *y*) **const**

Finds a tool for the given position.

wxDynamicToolBar::GetPreferredDim

void GetPreferredDim(**const wxSize&** *givenDim*, **wxSize&** *prefDim*)

Returns the preferred dimension, taking the given dimension and a reference to the result.

wxDynamicToolBar::GetToolInfo

wxDynToolInfo* GetToolInfo(**int** *toolIndex*)

Returns tool information for the given tool index.

wxDynamicToolBar::Layout

bool Layout()

Performs layout. See definitions of orientation types.

wxDynamicToolBar::OnEraseBackground

void OnEraseBackground(**wxEraseEvent&** *event*)

Responds to background erase events. Currently does nothing.

wxDynamicToolBar::OnPaint

void OnPaint(**wxPaintEvent&** *event*)

Responds to paint events, drawing separators.

wxDynamicToolBar::OnSize

void OnSize(**wxSizeEvent&** *event*)

Responds to size events, calling Layout.

wxDynamicToolBar::Realize

bool Realize()

Overriden from wxToolBarBase; does nothing.

wxDynamicToolBar::RemveTool

void RemveTool(**int** *toolIndex*)

Removes the given tool. Misspelt in order not to clash with a similar function in the base class.

wxDynamicToolBar::SetLayout

void SetLayout(**LayoutManagerBase*** *pLayout*)

Sets the layout for this toolbar.

wxDynamicToolBar::SizeToolWindows

void SizeToolWindows()

Internal function for sizing tool windows.

wxDynToolInfo

This class holds dynamic toolbar item information.

Derived from

wxToolLayoutItem (p. 94)

Include files

<wx/fl/dyntbar.h>

Data structures

wxFrameLayout

wxFrameLayout manages containment and docking of control bars, which can be docked along the top, bottom, right, or left side of the parent frame.

Derived from

wxEvtHandler

Include files

<wx/fl/controlbar.h>

Data structures

wxFrameLayout::wxFrameLayout

wxFrameLayout(**wxWindow*** *pParentFrame*, **wxWindow*** *pFrameClient = NULL*, **bool** *activateNow = TRUE*)

Constructor, taking parent window, the (MDI) client of the parent if there is one, and flag specifying whether to activate the layout.

wxFrameLayout()

Default constructor, used only for serialization.

wxFrameLayout::~wxFrameLayout

~wxFrameLayout()

Destructor. It does not destroy the bar windows.

wxFrameLayout::Activate

void Activate()

Activate can be called after some other layout has been deactivated, and this one must take over the current contents of the frame window. Effectively hooks itself to the frame window, re-displays all non-hidden bar windows and repaints the decorations.

wxFrameLayout::AddBar

void AddBar(**wxWindow*** *pBarWnd*, **const cbDimInfo&** *dimInfo*, **int** *alignment = FL_ALIGN_TOP*, **int** *rowNo = 0*, **int** *columnPos = 0*, **const wxString&** *name = "bar"*, **bool** *spyEvents = FALSE*, **int** *state = wxCBAR_DOCKED_HORIZONTALLY*)

Adds bar information to the frame layout. The appearance of the layout is not refreshed immediately; RefreshNow() can be called if necessary. Notes: the argument pBarWnd can by NULL, resulting in bar decorations to be drawn around the empty rectangle (filled with default background colour). Argument dimInfo can be reused for adding any number of bars, since it is not used directly - instead its members are copied. If the dimensions handler is present, its instance is shared (reference counted). The dimension handler should always be allocated on the heap. pBarWnd is the window to be managed. dimInfo contains dimension information. alignment is a value such as FL_ALIGN_TOP. rowNo is the vertical position or row in the pane (if in docked state). columnPos is the horizontal position within the row in pixels (if in docked state). name is a name by which the bar can be referred in layout customization dialogs. If spyEvents is TRUE, input events for the bar should be "spyed" in order to forward unhandled mouse clicks to the frame layout, for example to enable easy draggablity of toolbars just by clicking on their interior regions. For widgets like text/tree control this value should be FALSE, since there's no certain way to detect whether the event was actually handled. state is the initial state, such as wxCBAR_DOCKED_HORIZONTALLY, wxCBAR_FLOATING, wxCBAR_HIDDEN.

wxFrameLayout::AddPlugin

void AddPlugin(**wxClassInfo*** *pPlInfo*, **int** *paneMask = wxALL_PANES*)

An advanced methods for plugin configuration using their dynamic class information, for example CLASSINFO(pluginClass). First checks if the plugin of the given class is already "hooked up". If not, adds it to the top of the plugins chain.

wxFrameLayout::AddPluginBefore

void AddPluginBefore(**wxClassInfo*** *pNextPlInfo*, **wxClassInfo*** *pPlInfo*, **int** *paneMask = wxALL_PANES*)

First checks if the plugin of the given class is already hooked. If so, removes it, and then inserts it into the chain before the plugin of the class given by pNextPlInfo. Note: this

method is handy in some cases where the order of the plugin-chain could be important, for example when one plugin overrides some functionality of another already-hooked plugin, so that the former plugin should be hooked before the one whose functionality is being overridden.

wxFrameLayout::ApplyBarProperties

void ApplyBarProperties(**cbBarInfo*** *pBar*)

Reflects changes in bar information structure visually. For example, moves the bar, changes its dimension information, or changes the pane to which it is docked.

wxFrameLayout::CanReparent

bool CanReparent()

Returns TRUE if the platform allows reparenting. This may not return TRUE for all platforms. Reparenting allows control bars to be floated.

wxFrameLayout::CaptureEventsForPane

void CaptureEventsForPane(**cbDockPane*** *toPane*)

Called by plugins; also captures the mouse in the parent frame.

wxFrameLayout::CaptureEventsForPlugin

void CaptureEventsForPlugin(**cbPluginBase*** *pPlugin*)

Captures user input events for the given plugin. Input events are: mouse movement, mouse clicks, keyboard input.

wxFrameLayout::CreateCursors

void CreateCursors()

Creates the cursors.

wxFrameLayout::CreateUpdatesManager

cbUpdatesManagerBase* CreateUpdatesManager()

Returns a new cbGCUpdatesMgr object.

wxFrameLayout::Deactivate

void Deactivate()

Deactivate unhooks itself from frame window, and hides all non-hidden windows. Note: two frame layouts should not be active at the same time in the same frame window,

since it would cause messy overlapping of bar windows from both layouts.

wxFrameLayout::DestroyBarWindows

void DestroyBarWindows()

Destroys the bar windows.

wxFrameLayout::DoSetBarState

void DoSetBarState(**cbBarInfo*** *pBar*)

Applies the state to the window objects.

wxFrameLayout::EnableFloating

void EnableFloating(**bool** *enable = TRUE*)

Enables floating behaviour. By default floating of control bars is on.

wxFrameLayout::FindBarByName

cbBarInfo* FindBarByName(**const wxString&** *name*)

Finds the bar in the framelayout, by name.

wxFrameLayout::FindBarByWindow

cbBarInfo* FindBarByWindow(**const wxWindow*** *pWnd*)

Finds the bar in the framelayout, by window.

wxFrameLayout::FindPlugin

cbPluginBase* FindPlugin(**wxClassInfo*** *pPlInfo*)

Finds a plugin with the given class, or returns NULL if a plugin of the given class is not hooked.

wxFrameLayout::FirePluginEvent

void FirePluginEvent(**cbPluginEvent&** *event*)

This function should be used instead of passing the event to the ProcessEvent method of the top-level plugin directly. This method checks if events are currently captured and ensures that plugin-event is routed correctly.

wxFrameLayout::ForwardMouseEvent

void ForwardMouseEvent(**wxMouseEvent&** *event*, **cbDockPane*** *pToPane*, **int**

eventType)

Delegated from "bar-spy".

wxFrameLayout::GetBarPane

cbDockPane* GetBarPane(**cbBarInfo*** *pBar*)

Returns the pane to which the given bar belongs.

wxFrameLayout::GetBars

BarArrayT& GetBars()

Gets an array of bars.

wxFrameLayout::GetClientHeight

int GetClientHeight()

Returns the client height.

wxFrameLayout::GetClientRect

wxRect& GetClientRect()

Returns the client's rectangle.

wxFrameLayout::GetClientWidth

int GetClientWidth()

Returns the client width.

wxFrameLayout::GetFrameClient

wxWindow* GetFrameClient()

Returns the frame client, or NULL if not present.

wxFrameLayout::GetPane

cbDockPane* GetPane(**int** *alignment*)

Returns a pane for the given alignment. See pane alignment types.

wxFrameLayout::GetPaneProperties

void GetPaneProperties(**cbCommonPaneProperties&** *props*, **int** *alignment* = *FL_ALIGN_TOP*)

Gets the pane properties for the given alignment.

wxFrameLayout::GetPanesArray

cbDockPane GetPanesArray**()

Returns an array of panes. Used by update managers.

wxFrameLayout::GetParentFrame

wxWindow& GetParentFrame()

Returns the parent frame.

wxFrameLayout::GetPrevClientRect

wxRect& GetPrevClientRect()

Returns the previous client window rectangle.

wxFrameLayout::GetTopPlugin

cbPluginBase& GetTopPlugin()

Returns the current top-level plugin (the one that receives events first, except if input events are currently captured by some other plugin).

wxFrameLayout::GetUpdatesManager

cbUpdatesManagerBase& GetUpdatesManager()

Returns a reference to the updates manager. Note: in future, the updates manager will become a normal plugin.

wxFrameLayout::HasTopPlugin

bool HasTopPlugin()

Returns true if there is a top plugin.

wxFrameLayout::HideBarWindows

void HideBarWindows()

Hides the bar windows, and also the client window if present.

wxFrameLayout::HitTestPane

bool HitTestPane(**cbDockPane*** *pPane*, **int** *x*, **int** *y*)

Returns TRUE if the position is within the given pane.

wxFrameLayout::HitTestPanes

cbDockPane* HitTestPanes(**const wxRect&** *rect*, **cbDockPane*** *pCurPane*)

Returns the pane for which the rectangle hit test succeeds, giving preference to the given pane if supplied.

wxFrameLayout::HookUpToFrame

void HookUpToFrame()

Hooks the layout up to the frame (pushes the layout onto the frame's event handler stack).

wxFrameLayout::InverseVisibility

void InverseVisibility(**cbBarInfo*** *pBar*)

Toggles the bar between visible and hidden.

wxFrameLayout::LocateBar

bool LocateBar(**cbBarInfo*** *pBarInfo*, **cbRowInfo**** *ppRow*, **cbDockPane**** *ppPane*)

The purpose of this function is unknown.

wxFrameLayout::OnActivate

void OnActivate(**wxActivateEvent&** *event*)

Handles activation events. Currently does nothing.

wxFrameLayout::OnEraseBackground

void OnEraseBackground(**wxEraseEvent&** *event*)

Handles background erase events. Currently does nothing.

wxFrameLayout::OnIdle

void OnIdle(**wxIdleEvent&** *event*)

Handles idle events.

wxFrameLayout::OnKillFocus

void OnKillFocus(**wxFocusEvent&** *event*)

Handles focus kill events. Currently does nothing.

wxFrameLayout::OnLButtonDown

void OnLButtonDown(**wxMouseEvent&** *event*)

Event handler for a left down button event.

wxFrameLayout::OnLButtonUp

void OnLButtonUp(**wxMouseEvent&** *event*)

Event handler for a left button up event.

wxFrameLayout::OnLDblClick

void OnLDblClick(**wxMouseEvent&** *event*)

Event handler for a left doubleclick button event.

wxFrameLayout::OnMouseMove

void OnMouseMove(**wxMouseEvent&** *event*)

Event handler for a mouse move event.

wxFrameLayout::OnPaint

void OnPaint(**wxPaintEvent&** *event*)

Handles paint events, calling PaintPane for each pane.

wxFrameLayout::OnRButtonDown

void OnRButtonDown(**wxMouseEvent&** *event*)

Event handler for a right button down event.

wxFrameLayout::OnRButtonUp

void OnRButtonUp(**wxMouseEvent&** *event*)

Event handler for a right button up event.

wxFrameLayout::OnSetFocus

void OnSetFocus(**wxFocusEvent&** *event*)

Handles focus set events. Currently does nothing.

wxFrameLayout::OnSize

void OnSize(**wxSizeEvent&** *event*)

Event handler for a size event.

wxFrameLayout::PopAllPlugins

void PopAllPlugins()

Pop all plugins.

wxFrameLayout::PopPlugin

void PopPlugin()

Similar to wxWindow's "push/pop-event-handler" methods, execept that the plugin is deleted upon "popping".

wxFrameLayout::PositionClientWindow

void PositionClientWindow()

Called to apply the calculated layout to window objects.

wxFrameLayout::PositionPanes

void PositionPanes()

Called to apply the calculated layout to window objects.

wxFrameLayout::PushDefaultPlugins

void PushDefaultPlugins()

Adds the default plugins. These are cbPaneDrawPlugin, cbRowLayoutPlugin, cbBarDragPlugin, cbAntiflickerPlugin, cbSimpleCustomizePlugin. This method is automatically invoked if no plugins were found upon firing of the first plugin-event, i.e. when wxFrameLayout configures itself.

wxFrameLayout::PushPlugin

void PushPlugin(**cbPluginBase*** *pPugin*)

Similar to wxWindow's "push/pop-event-handler" methods, execept that the plugin is deleted upon "popping".

wxFrameLayout::RecalcLayout

void RecalcLayout(**bool** *repositionBarsNow = FALSE*)

Recalculates the layout of panes, and all bars/rows in each pane.

wxFrameLayout::RedockBar

bool RedockBar(**cbBarInfo*** *pBar*, **const wxRect&** *shapeInParent*, **cbDockPane*** *pToPane = NULL*, **bool** *updateNow = TRUE*)

ReddockBar can be used for repositioning existing bars. The given bar is first removed from the pane it currently belongs to, and inserted into the pane, which "matches" the given rectangular area. If pToPane is not NULL, the bar is docked to this given pane. To dock a bar which is floating, use the wxFrameLayout::DockBar method.

wxFrameLayout::RefreshNow

void RefreshNow(**bool** *recalcLayout = TRUE*)

Recalculates layout and performs on-screen update of all panes.

wxFrameLayout::ReleaseEventsFromPane

void ReleaseEventsFromPane(**cbDockPane*** *fromPane*)

Called by plugins; also releases mouse in the parent frame.

wxFrameLayout::ReleaseEventsFromPlugin

void ReleaseEventsFromPlugin(**cbPluginBase*** *pPlugin*)

Releases user input events for the given plugin. Input events are: mouse movement, mouse clicks, keyboard input

wxFrameLayout::RemoveBar

void RemoveBar(**cbBarInfo*** *pBar*)

Removes the bar from the layout permanently, and hides its corresponding window if present.

wxFrameLayout::RemovePlugin

void RemovePlugin(**wxClassInfo*** *pPlInfo*)

Checks if the plugin of the given class is hooked, and removes it if found.

wxFrameLayout::ReparentWindow

void ReparentWindow(**wxWindow*** *pChild*, **wxWindow*** *pNewParent*)

Reparents pChild to have parent pNewParent.

wxFrameLayout::RepositionFloatedBar

void RepositionFloatedBar(**cbBarInfo*** *pBar*)

Applies the calculated layout to a floating bar.

wxFrameLayout::RouteMouseEvent

void RouteMouseEvent(**wxMouseEvent&** *event*, **int** *pluginEvtType*)

Routes the mouse event to the appropriate pane.

wxFrameLayout::SetBarState

void SetBarState(**cbBarInfo*** *pBar*, **int** *newStatem*, **bool** *updateNow*)

Changes the bar's docking state (see possible control bar states).

wxFrameLayout::SetFrameClient

void SetFrameClient(**wxWindow*** *pFrameClient*)

Passes the client window (e.g. MDI client window) to be controlled by frame layout, the size and position of which should be adjusted to be surrounded by controlbar panes, whenever the frame is resized or the dimensions of control panes change.

wxFrameLayout::SetMargins

void SetMargins(**int** *top*, **int** *bottom*, **int** *left*, **int** *right*, **int** *paneMask* = *wxALL_PANES*)

Sets the margins for the given panes. The margins should go into cbCommonPaneProperties in the future. Note: this method should be called before any custom plugins are attached.

wxFrameLayout::SetPaneBackground

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

Sets the pane background colour.

wxFrameLayout::SetPaneProperties

void SetPaneProperties(**const cbCommonPaneProperties&** *props*, **int** *paneMask* = *wxALL_PANES*)

Sets the pane properties for the given alignment. Note: changing properties of panes does not result immediate on-screen update.

wxFrameLayout::SetTopPlugin

void SetTopPlugin(**cbPluginBase*** *pPlugin*)

Hooking custom plugins to frame layout. Note: when hooking one plugin on top of the other, use SetNextHandler or similar methods of wxEvtHandler class to compose the chain of plugins, than pass the left-most handler in this chain to the above methods (assuming that events are delegated from left-most towards right-most handler). This secenario is very inconvenient and "low-level", so use the Add/Push/PopPlugin methods instead.

wxFrameLayout::SetUpdatesManager

void SetUpdatesManager(**cbUpdatesManagerBase*** *pUMgr*)

Destroys the previous manager if any, and sets the new one.

wxFrameLayout::ShowFloatedWindows

void ShowFloatedWindows(**bool** *show*)

Shows all floated windows.

wxFrameLayout::UnhookFromFrame

void UnhookFromFrame()

Unhooks the layout from the frame.

wxFrameManager

Derived from

wxObject

Data structures

wxFrameManager::wxFrameManager

wxFrameManager()

wxFrameManager::~wxFrameManager

~wxFrameManager()

wxFrameManager::ActivateView

void ActivateView(**wxFrameView*** *pFrmView*)

void ActivateView(**int** *viewNo*)

wxFrameManager::AddView

void AddView(**wxFrameView*** *pFrmView*)

wxFrameManager::DeactivateCurrentView

void DeactivateCurrentView()

wxFrameManager::DestroyViews

void DestroyViews()

wxFrameManager::DoSerialize

void DoSerialize(**wxObjectStorage&** *store*)

wxFrameManager::EnableMenusForView

void EnableMenusForView(**wxFrameView*** *pView*, **bool** *enable*)

wxFrameManager::GetActiveView

wxFrameView* GetActiveView()

wxFrameManager::GetActiveViewNo

int GetActiveViewNo()

wxFrameManager::GetActiveViewNode

wxNode* GetActiveViewNode()

wxFrameManager::GetClientWindow

wxWindow* GetClientWindow()

wxFrameManager::GetObjectStore

wxObjectStorage& GetObjectStore()

wxFrameManager::GetParentFrame

wxFrame* GetParentFrame()

synonyms

wxFrameManager::GetParentWindow

wxWindow* GetParentWindow()

wxFrameManager::GetView

wxFrameView* GetView(**int** *viewNo*)

wxFrameManager::GetViewNo

int GetViewNo(**wxFrameView*** *pView*)

wxFrameManager::Init

void Init(**wxWindow*** *pMainFrame*, **const wxString&** *settingsFile = ""*)

if file name is empty, views are are not saved/loaded

wxFrameManager::ReloadViews

bool ReloadViews()

wxFrameManager::RemoveView

void RemoveView(**wxFrameView*** *pFrmView*)

wxFrameManager::SaveViewsNow

void SaveViewsNow()

wxFrameManager::SetClinetWindow

void SetClinetWindow(**wxWindow*** *pFrameClient*)

wxFrameManager::SyncAllMenus

void SyncAllMenus()

wxFrameManager::ViewsAreLoaded

bool ViewsAreLoaded()

GarbageCollector

This class implements an extremely slow but simple garbage collection algorithm.

Derived from

No base class

Include files

<wx/fl/garbagec.h>

Data structures

GarbageCollector::GarbageCollector

GarbageCollector()

Default constructor.

GarbageCollector::~GarbageCollector

~GarbageCollector()

Destructor.

GarbageCollector::AddDependency

void AddDependency(**void*** *pObj*, **void*** *pDependsOnObj*)

Prepare data for garbage collection.

GarbageCollector::AddObject

void AddObject(**void*** *pObj*, **int** *refCnt = 1*)

Prepare data for garbage collection.

GarbageCollector::ArrangeCollection

void ArrangeCollection()

Executes garbage collection algorithm.

GarbageCollector::DestroyItemList

void DestroyItemList(**wxList&** *lst*)

Destroys a list of items.

GarbageCollector::FindItemNode

wxNode* FindItemNode(**void*** *pForObj*)

Internal method for finding a node.

GarbageCollector::FindReferenceFreeItemNode

wxNode* FindReferenceFreeItemNode()

Internal method for findind and freeing a node.

GarbageCollector::GetCycledObjects

wxList& GetCycledObjects()

Get cycled objects.

GarbageCollector::GetRegularObjects

wxList& GetRegularObjects()

Accesses the results of the algorithm.

GarbageCollector::RemoveReferencesToNode

void RemoveReferencesToNode(**wxNode*** *pItemNode*)

Remove references to this node.

GarbageCollector::Reset

void Reset()

Removes all data from the garbage collector.

GarbageCollector::ResolveReferences

void ResolveReferences()

Internal method for resolving references.

LayoutManagerBase

This is a base class for layout algorithm implementations.

Derived from

No base class

Include files

<wx/fl/dyntbar.h>

Data structures

LayoutManagerBase::~LayoutManagerBase

~LayoutManagerBase()

Destructor.

LayoutManagerBase::Layout

void Layout(**const wxSize&** *parentDim*, **wxSize&** *resultingDim*, **wxLayoutItemArrayT&** *items*, **int** *horizGap*, **int** *vertGap*)

Constructor.

wxNewBitmapButton

This is an alternative class to wxBitmapButton. It is used in the implementation of dynamic toolbars.

Derived from

wxPanel

Include files

<wx/fl/newbmpbtn.h>

Data structures

wxNewBitmapButton::wxNewBitmapButton

wxNewBitmapButton(**const wxBitmap&** *labelBitmap = wxNullBitmap*, **const wxString&** *labelText = ""*, **int** *alignText = NB_ALIGN_TEXT_BOTTOM*, **bool** *isFlat = TRUE*, **int** *firedEventType = wxEVT_COMMAND_MENU_SELECTED*, **int** *marginX = 2*, **int** *marginY = 2*, **int** *textToLabelGap = 2*, **bool** *isSticky = FALSE*)

Constructor.

wxNewBitmapButton(**const wxString&** *bitmapFileName*, **const wxBitmapType** *bitmapFileType = wxBITMAP_TYPE_BMP*, **const wxString&** *labelText = ""*, **int** *alignText = NB_ALIGN_TEXT_BOTTOM*, **bool** *isFlat = TRUE*, **int** *firedEventType = wxEVT_COMMAND_MENU_SELECTED*, **int** *marginX = 2*, **int** *marginY = 2*, **int** *textToLabelGap = 2*, **bool** *isSticky = FALSE*)

Use this constructor if buttons have to be persistant

wxNewBitmapButton::~wxNewBitmapButton

~wxNewBitmapButton()

Destructor.

wxNewBitmapButton::DestroyLabels

void DestroyLabels()

Internal function for destroying labels.

wxNewBitmapButton::DrawDecorations

void DrawDecorations(**wxDC&** *dc*)

Draws the decorations.

wxNewBitmapButton::DrawLabel

void DrawLabel(**wxDC&** *dc*)

Draws the label.

wxNewBitmapButton::DrawShade

void DrawShade(**int** *outerLevel*, **wxDC&** *dc*, **wxPen&** *upperLeftSidePen*, **wxPen&** *lowerRightSidePen*)

Draws shading on the button.

wxNewBitmapButton::GetStateLabel

wxBitmap* GetStateLabel()

Returns the label that matches the current button state.

wxNewBitmapButton::IsInWindow

bool IsInWindow(**int** *x*, **int** *y*)

Returns TRUE if the given point is in the window.

wxNewBitmapButton::OnEraseBackground

void OnEraseBackground(**wxEraseEvent&** *event*)

Responds to an erase background event.

wxNewBitmapButton::OnKillFocus

void OnKillFocus(**wxFocusEvent&** *event*)

Responds to a kill focus event.

wxNewBitmapButton::OnLButtonDown

void OnLButtonDown(**wxMouseEvent&** *event*)

Responds to a left mouse button down event.

wxNewBitmapButton::OnLButtonUp

void OnLButtonUp(**wxMouseEvent&** *event*)

Responds to a left mouse button up event.

wxNewBitmapButton::OnMouseMove

void OnMouseMove(**wxMouseEvent&** *event*)

Responds to a mouse move event.

wxNewBitmapButton::OnPaint

void OnPaint(**wxPaintEvent&** *event*)

Responds to a paint event.

wxNewBitmapButton::OnSize

void OnSize(**wxSizeEvent&** *event*)

Responds to a size event.

wxNewBitmapButton::RenderAllLabelImages

void RenderAllLabelImages()

Renders label images.

wxNewBitmapButton::RenderLabelImage

void RenderLabelImage(**wxBitmap*&** *destBmp*, **wxBitmap*** *srcBmp*, **bool** *isEnabled = TRUE*, **bool** *isPressed = FALSE*)

Renders the label image.

wxNewBitmapButton::RenderLabelImages

void RenderLabelImages()

Renders label images.

wxNewBitmapButton::Reshape

void Reshape()

This function should be called after Create. It renders the labels, having reloaded the button image if necessary.

wxNewBitmapButton::SetAlignments

void SetAlignments(**int** *alignText = NB_ALIGN_TEXT_BOTTOM*, **int** *marginX = 2*, **int** *marginY = 2*, **int** *textToLabelGap = 2*)

Sets the text alignment and margins.

wxNewBitmapButton::SetLabel

void SetLabel(**const wxBitmap&** *labelBitmap*, **const wxString&** *labelText = ""*)

Sets the label and optionally label text.

wxToolLayoutItem

Tool layout item.

Derived from

wxObject

Include files

<wx/fl/dyntbar.h>

Data structures

```
typedef wxToolLayoutItem* wxToolLayoutItemPtrT


typedef wxDynToolInfo* wxDynToolInfoPtrT
```

wxToolWindow

A tool window is a special kind of frame that paints its own title, and can be used to implement small floating windows.

Derived from

wxFrame

Include files

<wx/fl/toolwnd.h>

Data structures

wxToolWindow::wxToolWindow

wxToolWindow()

Default constructor.

wxToolWindow::~wxToolWindow

~wxToolWindow()

Destructor.

wxToolWindow::AddMiniButton

void AddMiniButton(**cbMiniButton*** *pBtn*)

Adds a button. Buttons are added in right-to-left order.

wxToolWindow::AdjustRectPos

void AdjustRectPos(**const wxRect&** *original*, **const wxSize&** *newDim*, **wxRect&** *newRect*)

Helper function.

wxToolWindow::CalcResizedRect

void CalcResizedRect(**wxRect&** *rect*, **wxPoint&** *delta*, **const wxSize&** *minDim*)

Calculate resized rectangle.

wxToolWindow::DrawHintRect

void DrawHintRect(**const wxRect&** *r*)

Draws the hint rectangle.

wxToolWindow::GetClient

wxWindow* GetClient()

Returns the client window.

wxToolWindow::GetMinimalWndDim

wxSize GetMinimalWndDim()

Helper function.

wxToolWindow::GetPreferredSize

wxSize GetPreferredSize(**const wxSize&** *given*)

Returns the preferred size for the window.

wxToolWindow::GetScrMousePos

void GetScrMousePos(**wxMouseEvent&** *event*, **wxPoint&** *pos*)

Gets the mouse position in screen coordinates.

wxToolWindow::GetScrWindowRect

void GetScrWindowRect(**wxRect&** *r*)

Maps client coordinates to screen coordinates.

wxToolWindow::HandleTitleClick

bool HandleTitleClick(**wxMouseEvent&** *event*)

Handles clicking on the title. By default, does nothing.

wxToolWindow::HitTestWindow

int HitTestWindow(**wxMouseEvent&** *event*)

Tests if the mouse position is in this window.

wxToolWindow::LayoutMiniButtons

void LayoutMiniButtons()

Lays out the buttons.

wxToolWindow::OnEraseBackground

void OnEraseBackground(**wxEraseEvent&** *event*)

Responds to an erase background event.

wxToolWindow::OnLeftDown

void OnLeftDown(**wxMouseEvent&** *event*)

Responds to a mouse left down event.

wxToolWindow::OnLeftUp

void OnLeftUp(**wxMouseEvent&** *event*)

Responds to a mouse left up event.

wxToolWindow::OnMiniButtonClicked

void OnMiniButtonClicked(**int** *btnIdx*)

Called when a mini button is clicked. By default, does nothing.

wxToolWindow::OnMotion

void OnMotion(**wxMouseEvent&** *event*)

Responds to a mouse move event.

wxToolWindow::OnPaint

void OnPaint(**wxPaintEvent&** *event*)

Responds to a paint event.

wxToolWindow::OnSize

void OnSize(**wxSizeEvent&** *event*)

Responds to a size event.

wxToolWindow::SetClient

void SetClient(**wxWindow*** *pWnd*)

Sets the client for this tool window.

wxToolWindow::SetHintCursor

void SetHintCursor(**int** *type*)

Sets the hint cursor.

wxToolWindow::SetTitleFont

void SetTitleFont(**wxFont&** *font*)

Sets the title font.

Classes by category

A classification of FL classes by category.

Plugin classes

Plugins can be added to frame layouts to extend behaviour.

cbAntiflickerPlugin (p. 8)	Double-buffering class
cbBarDragPlugin (p. 10)	Implements drag behaviour.
cbBarHintsPlugin (p. 14)	Implements bar decoration and sizing behaviour.
cbHintAnimationPlugin (p. 40)	Draws animated hints when the user drags a pane.
cbPaneDrawPlugin (p. 46)	Implements most of MFC-style control bar implementation.
cbPluginBase (p. 51)	Abstract base class for all control-bar related plugins.
cbRowDragPlugin (p. 54)	Implements row-dragging functionality.
cbRowLayoutPlugin (p. 61)	Implements row layout functionality.
cbSimpleCustomizationPlugin (p. 64)	Enables customization of a bar.
cbBarSpy (p. 18)	Helper class used for spying for unhandled mouse events on control bars and forwarding them to the frame layout.

Window classes

Windows classes (note that the mini-button implementations are not true windows in that they do not derive from wxWindow).

wxToolWindow (p. 94)	A small frame that paints its own titlebar.
cbFloatedBarWindow (p. 37)	A kind of wxToolWindow implementing floating windows.
cbMiniButton (p. 44)	Base class for wxToolWindow titlebar buttons.
cbCloseBox (p. 19)	Close button for wxToolWindow titlebar.
cbCollapseBox (p. 19)	Collapse button for wxToolWindow titlebar.
cbDockBox (p. 23)	Dock button for wxToolWindow titlebar.
cbCloseBox (p. 19)	Close button for wxToolWindow titlebar.
wxNewBitmapButton (p. 91)	Alternative bitmap button class.

Layout management classes

These classes relate to the layout management framework.

cbDockPane (p. 23)	Manages containment and control of bars in a prent frame.
BagLayout (p. 7)	BagLayout lays out items in left-to-right order from top to bottom.
cbUpdatesManagerBase (p. 69)	An abstract interface for display update optimization logic.
cbSimpleUpdatesMgr (p. 65)	Implements optimized logic for refreshing areas of frame layout that need to be updated.
cbGCUpdatesMgr (p. 38)	Implements optimized logic for refresh, based on a garbage collection algorithm.
GarbageCollector (p. 88)	A garbage collection algorithm for use in display refresh optimization.
wxFrameLayout (p. 75)	Manages containment and docking of control bars, which can be docked along the top, bottom, right, or left side of the parent frame.

Event classes

Events are used to decouple parts of the layout framework. For event macros and identifiers, please see the topic *Event macros and identifiers* (p. 101).

cbCustomizeBarEvent (p. 21)	Class for bar customization events.
cbCustomizeLayoutEvent (p. 21)	Class for layout customization events.
cbDrawBarDecorEvent (p. 33)	Class for bar decoration drawing events.
cbDrawBarHandlesEvent (p. 33)	Class for bar handles drawing events.
cbDrawHintRectEvent (p. 33)	Class for hint-rectangle drawing events.
cbDrawPaneBkGroundEvent (p. 34)	Class for pane background drawing events.
cbDrawPaneDecorEvent (p. 34)	Class for pane decoration drawing events.
cbDrawRowBkGroundEvent (p. 35)	Class for row background drawing events.
cbDrawRowDecorEvent (p. 35)	Class for row decoration drawing events.
cbDrawRowHandlesEvent (p. 35)	Class for row handles drawing events.
cbFinishDrawInAreaEvent (p. 36)	Class for finish drawing in area events.
cbInsertBarEvent (p. 42)	Class for bar insertion events.
cbLayoutRowEvent (p. 42)	Class for single row layout events.

cbLayoutRowsEvent (p. 43)	Class for multiple rows layout events.
cbLeftDClickEvent (p. 43)	Class for mouse left double click events.
cbLeftDownEvent (p. 43)	Class for mouse left down events.
cbLeftUpEvent (p. 44)	Class for mouse left up events.
cbMotionEvent (p. 46)	Class for mouse motion events.
cbPluginEvent (p. 52)	Base class for all control-bar plugin events.
cbRemoveBarEvent (p. 52)	Class for bar removal events.
cbResizeBarEvent (p. 53)	Class for bar resize events.
cbResizeRowEvent (p. 53)	Class for row resize events.
cbRightDownEvent (p. 53)	Class for mouse right down events.
cbRightUpEvent (p. 54)	Class for mouse right up events.
cbSizeBarWndEvent (p. 67)	Class for bar window resize events.
cbStartBarDraggingEvent (p. 67)	Class for start-bar-dragging events.
cbStartDrawInAreaEvent (p. 67)	Class for start drawing in area events.

Topic overviews

This chapter contains a selection of topic overviews, first things first:

Notes on using the reference

In the descriptions of the wxWindows classes and their member functions, note that descriptions of inherited member functions are not duplicated in derived classes unless their behaviour is different. So in using a class such as wxScrolledWindow, be aware that wxWindow functions may be relevant.

Note also that arguments with default values may be omitted from a function call, for brevity. Size and position arguments may usually be given a value of -1 (the default), in which case wxWindows will choose a suitable value.

Most strings are returned as wxString objects. However, for remaining char * return values, the strings are allocated and deallocated by wxWindows. Therefore, return values should always be copied for long-term use, especially since the same buffer is often used by wxWindows.

The member functions are given in alphabetical order except for constructors and destructors which appear first.

Event macros and identifiers

These are the event macros and event identifiers defined by FL.

Event macro	Event identifier
EVT_PL_LEFT_DOWN(func)	cbEVT_PL_LEFT_DOWN
EVT_PL_LEFT_UP(func)	cbEVT_PL_LEFT_UP
EVT_PL_RIGHT_DOWN(func)	cbEVT_PL_RIGHT_DOWN
EVT_PL_RIGHT_UP(func)	cbEVT_PL_RIGHT_UP
EVT_PL_MOTION(func)	cbEVT_PL_MOTION
EVT_PL_LEFT_DCLICK(func)	cbEVT_PL_LEFT_DCLICK
EVT_PL_LAYOUT_ROW(func)	cbEVT_PL_LAYOUT_ROW
EVT_PL_RESIZE_ROW(func)	cbEVT_PL_RESIZE_ROW
EVT_PL_LAYOUT_ROWS(func)	cbEVT_PL_LAYOUT_ROWS
EVT_PL_INSERT_BAR(func)	cbEVT_PL_INSERT_BAR
EVT_PL_RESIZE_BAR(func)	cbEVT_PL_RESIZE_BAR

EVT_PL_REMOVE_BAR(func)	cbEVT_PL_REMOVE_BAR
EVT_PL_SIZE_BAR_WND(func)	cbEVT_PL_SIZE_BAR_WND
EVT_PL_DRAW_BAR_DECOR(func)	cbEVT_PL_DRAW_BAR_DECOR
EVT_PL_DRAW_ROW_DECOR(func)	cbEVT_PL_DRAW_ROW_DECOR
EVT_PL_DRAW_PANE_DECOR(func)	cbEVT_PL_DRAW_PANE_DECOR
EVT_PL_DRAW_BAR_HANDLES(func)	cbEVT_PL_DRAW_BAR_HANDLES
EVT_PL_DRAW_ROW_HANDLES(func)	cbEVT_PL_DRAW_ROW_HANDLES
EVT_PL_DRAW_ROW_BKGROUND(func)	cbEVT_PL_DRAW_ROW_BKGROUND
EVT_PL_DRAW_PANE_BKGROUND(func)	cbEVT_PL_DRAW_PANE_BKGROUND
EVT_PL_START_BAR_DRAGGING(func)	cbEVT_PL_START_BAR_DRAGGING
EVT_PL_DRAW_HINT_RECT(func)	cbEVT_PL_DRAW_HINT_RECT
EVT_PL_START_DRAW_IN_AREA(func)	cbEVT_PL_START_DRAW_IN_AREA
EVT_PL_FINISH_DRAW_IN_AREA(func)	cbEVT_PL_FINISH_DRAW_IN_AREA
EVT_PL_CUSTOMIZE_BAR(func)	cbEVT_PL_CUSTOMIZE_BAR
EVT_PL_CUSTOMIZE_LAYOUT(func)	cbEVT_PL_CUSTOMIZE_LAYOUT

See also the classes:

cbCustomizeBarEvent (p. 21)	Class for bar customization events.
cbCustomizeLayoutEvent (p. 21)	Class for layout customization events.
cbDrawBarDecorEvent (p. 33)	Class for bar decoration drawing events.
cbDrawBarHandlesEvent (p. 33)	Class for bar handles drawing events.
cbDrawHintRectEvent (p. 33)	Class for hint-rectangle drawing events.
cbDrawPaneBkGroundEvent (p. 34)	Class for pane background drawing events.
cbDrawPaneDecorEvent (p. 34)	Class for pane decoration drawing events.
cbDrawRowBkGroundEvent (p. 35)	Class for row background drawing events.
cbDrawRowDecorEvent (p. 35)	Class for row decoration drawing events.
cbDrawRowHandlesEvent (p. 35)	Class for row handles drawing events.
cbFinishDrawInAreaEvent (p. 36)	Class for finish drawing in area events.
cbInsertBarEvent (p. 42)	Class for bar insertion events.
cbLayoutRowEvent (p. 42)	Class for single row layout events.

cbLayoutRowsEvent (p. 43)	Class for multiple rows layout events.
cbLeftDClickEvent (p. 43)	Class for mouse left double click events.
cbLeftDownEvent (p. 43)	Class for mouse left down events.
cbLeftUpEvent (p. 44)	Class for mouse left up events.
cbMotionEvent (p. 46)	Class for mouse motion events.
cbPluginEvent (p. 52)	Base class for all control-bar plugin events.
cbRemoveBarEvent (p. 52)	Class for bar removal events.
cbResizeBarEvent (p. 53)	Class for bar resize events.
cbResizeRowEvent (p. 53)	Class for row resize events.
cbRightDownEvent (p. 53)	Class for mouse right down events.
cbRightUpEvent (p. 54)	Class for mouse right up events.
cbSizeBarWndEvent (p. 67)	Class for bar window resize events.
cbStartBarDraggingEvent (p. 67)	Class for start-bar-dragging events.
cbStartDrawInAreaEvent (p. 67)	Class for start drawing in area events.

FAQ

A row of all non-fixed bars don't position properly

By Julian Smart.

I found that if I added all non-fixed bars, bars would overlap. This seems to be because the proportional resizing doesn't work before the window is laid out. I worked around this by setting pane sizes *before* the bars are added:

```
wxSize sz = GetClientSize();

// Set width for panes to help it do the calculations
int i;
for (i = 0; i < 2; i++)
{
    cbDockPane& pane = * (m_frameLayout->GetPane(i));
    pane.SetPaneWidth(sz.x);
}
```