wxApp overview

Classes: wxApp

• A wxWindows application does not have a main procedure; the equivalent is the OnInit member defined for a class derived from wxApp.

• OnInit must create and return a main window frame as a bare minimum. If NULL is returned from OnInit, the application will exit.

• Note that the program's command line arguments, represented by argc and argv, are available from within wxApp member functions.

• An application closes by destroying all windows. Because all frames must be destroyed for the application to exit,
• it is advisable to use parent frames wherever possible when creating new frames, so that deleting the top level frame will automatically delete child frames. The alternative is to explicitly delete child frames in the top-level frame's wxFrame::OnClose member.

• In emergencies the wxExit function can be called to kill the application.
DrawingApp.h

#ifndef _DRAWINGAPP_H
#define _DRAWINGAPP_H

#include "wx.h"

// Define a new application
class DrawingApp: public wxApp
{
    public:
        DrawingApp(void) ;
        wxFrame *OnInit(void);
};

#endif
// application constructor

DrawingApp::DrawingApp() {}

// The `main program' equivalent, creating the windows and returning the

wxFrame *DrawingApp::OnInit(void)
{

}

// This statement initialises the whole application
DrawingApp drawingApp;
Device context overview

Classes: wxDC, wxPostScriptDC, wxMetaFileDC, wxMemoryDC, wxPrinterDC, wxScreenDC.

• A wxDC is a device context onto which graphics and text can be drawn.

• It is intended to represent a number of output devices in a generic way, so a canvas has a device context and a printer also has a device context.

• In this way, the same piece of code may write to a number of different devices, if the device context is used as a parameter.

• wxDC is abstract and cannot be used to create device context objects. Instead, use a derived class. wxCanvasDC is a context that cannot be created by the user but can be retrieved from a wxCanvas by using wxCanvas::GetDC.
• When writing code to draw into a device context, use wxDC as a parameter whenever possible, to allow the most general use of your drawing code.

• You can then pass a device context object of any derived type.

```cpp
wxDC *dc = canvas->GetDC();
dc->Clear();
Draw(*dc,TRUE);

wxPrinterDC dc(NULL, NULL, NULL);
Draw(dc, TRUE);

wxMemoryDC dc;
dc.Clear();
```
Writing a wxWindows application: a rough guide

• To set a wxWindows application going, you'll need to derive a wxApp class.

• An application must have a top-level wxFrame window (returned by wxApp::OnInit), each frame containing one or more instances of wxPanel, wxTextWindow or wxCanvas.

• A frame can have a wxMenuBar, a status line.

• A wxPanel is used to place items (classes derived from wxItem) which are used for user interaction. Examples of items are wxButton, wxCheckBox, wxChoice, wxListBox, wxSlider, wxRadioBox.
• If you want to draw arbitrary graphics, you'll need a wxCanvas. In fact, you never draw directly onto a canvas---you use a devicecontext (DC).

• wxDC is the base for wxCanvasDC, wxMemoryDC, wxPostScriptDC, wxMetaFileDC and wxPrinterDC.

• If your drawing functions have wxDC as a parameter, you can pass any of these DCs to the function, and thus use the same code to draw to several different devices.

• You can draw using the member functions of wxDC, such as wxDC::DrawLine and wxDC::DrawText. Control colour on a canvas (wxColour) with brushes (wxBrush) and pens (wxPen).

• Most modern applications will have an on-line, hypertext help system; for this,
you need wxHelp and the wxHelpInstance class to control wxHelp.

wxFrame: wxWindow

• A frame is a window which contains subwindows of various kinds. It has a title bar and, optionally, a menu bar, and a status line.

• Depending on the platform, the frame has further menus or buttons relating to window movement, sizing, closing, etc.

• Most of these events are handled by the host system without need for special handling by the application.

• However, the application should normally define an wxFrame::OnClose handler for the frame so that related data and subwindows can be cleaned up.

• A frame may contain the subwindows wxCanvas, wxPanel and wxTextWindow.
• Some of the MS Windows issues of Multiple Document Interface (MDI) versus Single Document Interface (SDI) frames are covered in the user manual.

• If you wish to have a toolbar on an MDI parent frame, create the toolbar as normal (as a child of the MDI frame), set the appropriate height for it, and call `wxFrame::SetToolBar`.

```cpp
drawingFrame :: DrawingFrame ( wxFrame *frame, char *title, int x, int y, int w, int h ) :
              wxFrame(frame, title, x, y, w, h)
{

}

OnMouseCommand()
OnMouseEvent()
```
OnClose()
OnSize()

wxCanvas: wxWindow

• A canvas is a subwindow onto which graphics and text can be drawn, and mouse and keyboard input can be intercepted.

• At present, panel items cannot be placed on a canvas.

• When you draw onto a canvas, you are really drawing onto a device context (see wxDC, wxCanvasDC).

• Although you can use the members of wxCanvas for drawing, it is much better to get the device context from the canvas (see GetDC) and draw into that.

• Code which can draw into one device context can be reused for others, such as
PostScript or memory device contexts (see \texttt{wxPostScriptDC} and \texttt{wxMemoryDC}).

\begin{verbatim}
DrawingCanvas::DrawingCanvas(wxFrame *frame, int x, int y, int w, int h, long style):
  wxCanvas(frame, x, y, w, h, style)
{

}

OnEvent()
OnPaint()
OnScroll()
\end{verbatim}
wxMenu: wxWindow

- A menu is a popup (or pull down) list of items, one of which may be selected before the menu goes away (clicking elsewhere dismisses the menu).

- Menus may be used to construct either menu bars or popup menus.

- A menu item has an integer ID associated with it which can be used to identify the selection, or to change the menu item in some way.
wxMenuBar: wxWindow

• A menu bar is a series of menus accessible from the top of a frame.

• Selecting a title pulls down a menu; selecting a menu item causes a MenuSelection message to be passed to the frame with the menu item integer id as the only argument.
popupMenu = new wxMenu(NULL, (wxFunction)PopupFunction);

popupMenu->Append(RectangleShape, "Rectangle");
popupMenu->Append(CircleShape, "Circle");
popupMenu->Append(TriangleShape, "Triangle");
popupMenu->Append(LineShape, "Line");
popupMenu->Append(PolygonShape, "Polygon");

menu_bar = new wxMenuBar;
menu_bar->Append(popupMenu, "&Shape");

DrawTool.cpp

// Declare global resources

DrawingFrame *frame = NULL;
wxMenuBar *menu_bar = NULL;
wxMenu *file_menu = NULL;
wxMenu *popupMenu = NULL;

// Must initialise these in OnInit, not statically
wxPen *draw_pen;
wxPen *select_pen;

// This statement initialises the whole application
// Define my frame constructor
DrawingFrame::DrawingFrame(wxFrame
*frame, char *title, int x, int y, int w, int h):
    wxFrame(frame, title, x, y, w, h)
{

}

// Intercept menu commands
void DrawingFrame :: OnMenuCommand
(int id)
{
    switch (id)
case HELLO.Quit:
{
    OnClose();
    delete this;
    break;
}
}

// Size the subwindows when the frame is resized
void DrawingFrame::OnSize(int w, int h)
{

}

// Define the behaviour for the frame closing
// - must delete all frames except for the main one.
Bool DrawingFrame::OnClose(void)
{


delete popupMenu;

}
// Define the repainting behaviour
void DrawingCanvas::OnPaint(void)
{
    frame->Draw(*(GetDC()),TRUE);
}

void DrawingCanvas::OnEvent (wxMouseEvent& event)
{
    if (event.RightDown())
    {
    }

    if (event.LeftDown())
    {
    }

    if (event.Dragging() &&
    event.LeftIsDown())
    {
}
void
DrawingCanvas::OnScroll(wxCommandEvent& event)
{
    wxCanvas::OnScroll(event);
}

DrawingApp.cpp

// application constructor

DrawingApp::DrawingApp() {}

// The `main program' equivalent, creating the windows and returning the

wxFrame *DrawingApp::OnInit(void)
{
    /* Create a pens for drawing (thin) and selecting (thicker)*/
draw_pen = new wxPen("RED", 1, wxSOLID);
select_pen = new wxPen("RED", 3, wxSOLID);

// Create the main frame window
frame = new DrawingFrame(NULL, "Hello wxWindows", 0, 0, 550, 500);

// Give it a status line
frame->CreateStatusLine(2);

// Make a file menu
wxMenu *file_menu = new wxMenu;

file_menu->Append(HELLO_QUIT, "&Quit", "Quit program");

// Make a cursor menu
wxMenu *cursor_menu = new wxMenu;
cursor_menu->Append (wxCURSOR_ARROW, "Arrow");
cursor_menu->Append (wxCURSOR_CROSS, "Cross");
cursor_menu-> Append (wxCURSOR_PENCIL, "Pencil");
cursor_menu->Append (wxCURSOR_HAND, "Hand");
cursor_menu->Append (wxCURSOR_NO_ENTRY, "No Entry");
cursor_menu->Append (wxCURSOR_SIZING, "Sizing");

// make a menu bar with file and cursor menus

menu_bar = new wxMenuBar;
menu_bar->Append(file_menu, "&File");
menu_bar->Append(cursor_menu, "&Cursor");

// Associate the menu bar with the frame
frame->SetMenuBar(menu_bar);
// Make a drawing area

int width, height;
frame->GetClientSize(&width, &height);

DrawingCanvas *canvas = new DrawingCanvas(frame, 0, 0, width, height, wxRETAINED);

wxCursor *cursor = new wxCursor(wxCURSOR_PENCIL);
canvas->SetBackground (wxWHITE_BRUSH);
canvas->SetCursor(cursor);

/* Give it scrollbars: the virtual canvas is 20 * 50 = 1000 pixels in each direction*/

canvas->SetScrollbars(20, 20, 50, 50, 4, 4);
canvas->SetPen(draw_pen);
frame->canvas = canvas;

frame->Show(TRUE);
// create a popup menu

popupMenu = new wxMenu(NULL,
(wxFuction)PopupFunction);

popupMenu-&gt;Append(RectangleShape,
"Rectangle");
popupMenu-&gt;Append(CircleShape,
"Circle");
popupMenu-&gt;Append(TriangleShape,
"Triangle");
popupMenu-&gt;Append(LineShape,
"Line");
popupMenu-&gt;Append(PolygonShape,
"Polygon");

// Return the main frame window
return frame;
}

void PopupFunction(wxMenu&amp; menu,
wxCommandEvent&amp; event)
{
// Take decision based on
event.commandInt