Home Contents

Menus and toolbars in wxWidgets

A menubar is one of the most visible parts of the GUI application. It is a group of commands located in various menus. While in console applications you had to remember all those arcane commands, here we have most of the commands grouped into logical parts. There are accepted standards that further reduce the amount of time spending to learn a new application. To implement a menubar in wxWidgets we need to have three classes: a `wxMenuBar`, a `wxMenu`, and a `wxMenuItem`.

Simple menu example

Creating a menubar in wxWidgets is very simple.

menu.h

```
#include <wx/wx.h>
#include <wx/menu.h>
```

```
class SimpleMenu : public wxFrame
{
public:
    SimpleMenu(const wxString& title);

    void OnQuit(wxCommandEvent& event);

    wxMenuBar *menubar;
    wxMenu *file;

};
```

menu.cpp

```
#include "menu.h"



SimpleMenu::SimpleMenu(const wxString& title)
       : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(280, 180))
{

  menubar = new wxMenuBar;
  file = new wxMenu;
  file->Append(wxID_EXIT, wxT("&Quit"));
  menubar->Append(file, wxT("&File"));
  SetMenuBar(menubar);

  Connect(wxID_EXIT, wxEVT_COMMAND_MENU_SELECTED,
      wxCommandEventHandler(SimpleMenu::OnQuit));
  Centre();

}

void SimpleMenu::OnQuit(wxCommandEvent& WXUNUSED(event))
{
  Close(true);
}
```

main.h

```
#include <wx/wx.h>

class MyApp : public wxApp
{
  public:
    virtual bool OnInit();
};
```

main.cpp

```
#include "main.h"
#include "menu.h"

IMPLEMENT_APP(MyApp)

bool MyApp::OnInit()
{

    SimpleMenu *menu = new SimpleMenu(wxT("Simple Menu"));
    menu->Show(true);

    return true;
}
```

```
menubar = new wxMenuBar;
```

First we create a menubar object.

```
file = new wxMenu;
```

Next we create a menu object.

```
file->Append(wxID_EXIT, wxT("&Quit"));
```

We append a menu item into the menu object. The first parameter is the id of the menu item. The second parameter is the name of the menu item. Here we did not create a `wxMenuItem` explicitly. It was created by the `Append()` method behind the scenes. Later on, we will create a `wxMenuItem` manually.

```
menubar->Append(file, wxT("&File"));
SetMenuBar(menubar);
```

After that, we append a menu into the menubar. The & character creates an accelerator key. The character that follows the & is underlined. This way the menu is accessible via the Alt+F shortcut. In the end, we call the `SetMenuBar()` method. This method belongs to the `wxFrame` widget. It sets up the menubar.

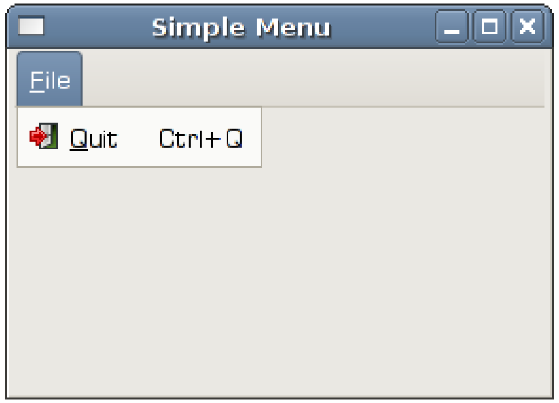

Figure: Simple menu example

Submenus

Each menu can also have a submenu. This way we can group similar commands into groups. For example we can place commands that hide or show various toolbars like personal bar, address bar, status bar, or navigation bar into a submenu called toolbars. Within a menu, we can separate commands with a separator. It is a simple line. It is common practice to separate commands like new, open, save from commands like print, print preview with a single separator. In our example we will see, how we can create submenus and menu separators.

menu.h

```
#include <wx/wx.h>
#include <wx/menu.h>

class SubMenu : public wxFrame
{
public:
  SubMenu(const wxString& title);

  void OnQuit(wxCommandEvent & event);

  wxMenuBar *menubar;
  wxMenu *file;
  wxMenu *imp;
  wxMenuItem *quit;

};
```

menu.cpp

```
#include "menu.h"


SubMenu::SubMenu(const wxString& title)
       : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(280, 180))
{

  menubar = new wxMenuBar;
  file = new wxMenu;

  file->Append(wxID_ANY, wxT("&New"));
  file->Append(wxID_ANY, wxT("&Open"));
  file->Append(wxID_ANY, wxT("&Save"));
  file->AppendSeparator();

  imp = new wxMenu;
  imp->Append(wxID_ANY, wxT("Import newsfeed list..."));
  imp->Append(wxID_ANY, wxT("Import bookmarks..."));
  imp->Append(wxID_ANY, wxT("Import mail..."));

  file->AppendSubMenu(imp, wxT("I&mport"));
  quit = new wxMenuItem(file, wxID_EXIT, wxT("&Quit\tCtrl+W"));
  file->Append(quit);

  menubar->Append(file, wxT("&File"));
  SetMenuBar(menubar);

  Connect(wxID_EXIT, wxEVT_COMMAND_MENU_SELECTED,
      wxCommandEventHandler(SubMenu::OnQuit));
  Centre();

}

void SubMenu::OnQuit(wxCommandEvent& WXUNUSED(event))
{
  Close(true);
}
```

main.h

```
#include <wx/wx.h>

class MyApp : public wxApp
{
  public:
    virtual bool OnInit();
};
```

main.cpp

```
#include "main.h"
#include "menu.h"

IMPLEMENT_APP(MyApp)

bool MyApp::OnInit()
{

    SubMenu *smenu = new SubMenu(wxT("Submenu"));
    smenu->Show(true);

    return true;
}
```

We created one submenu in a file menu. It is an import submenu, which can be seen in Opera web browser.

```
file->AppendSeparator();
```

A menu separator line is created calling an `AppendSeparator()` method.

```
imp = new wxMenu;
imp->Append(wxID_ANY, wxT("Import newsfeed list..."));
imp->Append(wxID_ANY, wxT("Import bookmarks..."));
imp->Append(wxID_ANY, wxT("Import mail..."));

file->AppendSubMenu(imp, wxT("I&mport"));
```

A submenu is created like a normal menu. It is appended with a `AppendSubMenu()` method.

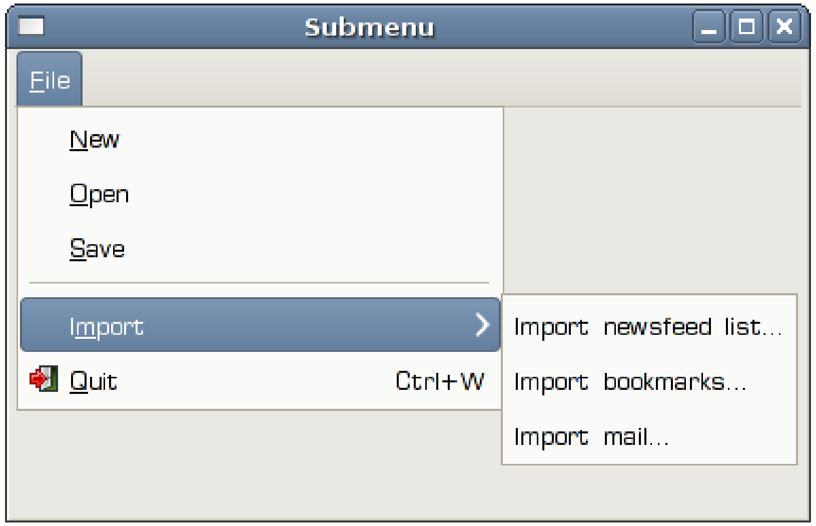

Figure: Submenu

Toolbars

Menus group all commands that we can use in an application. Toolbars provide a quick access to the most frequently used commands.

```
virtual wxToolBar* CreateToolBar(long style = wxNO_BORDER | wxTB_HORIZONTAL,
            wxWindowID id = -1, const wxString& name = "toolBar")
```

To create a toolbar, we call the `CreateToolBar()` method of the frame widget.

A simple toolbar

Our first example will create a simple toolbar.

toolbar.h

```
#include <wx/wx.h>

class Toolbar : public wxFrame
{
public:
    Toolbar(const wxString& title);

    void OnQuit(wxCommandEvent& event);


};
```

toolbar.cpp

```
#include "toolbar.h"


Toolbar::Toolbar(const wxString& title)
       : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(280, 180))
{

  wxImage::AddHandler( new wxPNGHandler );

  wxBitmap exit(wxT("exit.png"), wxBITMAP_TYPE_PNG);

  wxToolBar *toolbar = CreateToolBar();
  toolbar->AddTool(wxID_EXIT, exit, wxT("Exit application"));
  toolbar->Realize();

  Connect(wxID_EXIT, wxEVT_COMMAND_TOOL_CLICKED,
      wxCommandEventHandler(Toolbar::OnQuit));

  Centre();

}

void Toolbar::OnQuit(wxCommandEvent& WXUNUSED(event))
{
  Close(true);
}
```

main.h

```
#include <wx/wx.h>

class MyApp : public wxApp
{
  public:
    virtual bool OnInit();
};
```

main.cpp

```
#include "main.h"
#include "toolbar.h"

IMPLEMENT_APP(MyApp)

bool MyApp::OnInit()
{

    Toolbar *toolbar = new Toolbar(wxT("Toolbar"));
    toolbar->Show(true);
    return true;
}
```

In our example, we create a toolbar and one tool button. Clicking on the toolbar button will terminate the application.

```
wxToolBar *toolbar = CreateToolBar();
```

We create a toolbar.

```
toolbar->AddTool(wxID_EXIT, exit, wxT("Exit application"));
```

We add a tool to the toolbar.

```
toolbar->Realize();
```

After we have added the tools, we call the `Realize()` method.

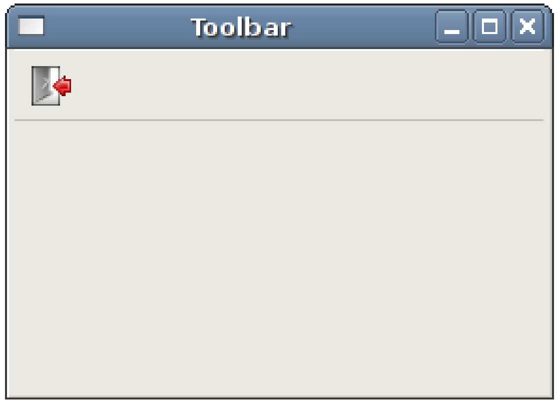

Figure: Toolbar

Toolbars

If we want to have more than one toolbar, we must create them in a different way, e.g. other than calling the `CreateToolbar()` method.

toolbars.h

```
#include <wx/wx.h>

class Toolbar : public wxFrame
{
public:
  Toolbar(const wxString& title);

  void OnQuit(wxCommandEvent& event);

  wxToolBar *toolbar1;
  wxToolBar *toolbar2;

};
```

toolbars.cpp

```
#include "toolbars.h"


Toolbar::Toolbar(const wxString& title)
       : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(280, 180))
{

  wxImage::AddHandler( new wxPNGHandler );

  wxBitmap exit(wxT("exit.png"), wxBITMAP_TYPE_PNG);
  wxBitmap newb(wxT("new.png"), wxBITMAP_TYPE_PNG);
  wxBitmap open(wxT("open.png"), wxBITMAP_TYPE_PNG);
  wxBitmap save(wxT("save.png"), wxBITMAP_TYPE_PNG);

  wxBoxSizer *vbox = new wxBoxSizer(wxVERTICAL);

  toolbar1 = new wxToolBar(this, wxID_ANY);
  toolbar1->AddTool(wxID_ANY, newb, wxT(""));
  toolbar1->AddTool(wxID_ANY, open, wxT(""));
  toolbar1->AddTool(wxID_ANY, save, wxT(""));
  toolbar1->Realize();

  toolbar2 = new wxToolBar(this, wxID_ANY);
  toolbar2->AddTool(wxID_EXIT, exit, wxT("Exit application"));
  toolbar2->Realize();

  vbox->Add(toolbar1, 0, wxEXPAND);
  vbox->Add(toolbar2, 0, wxEXPAND);

  SetSizer(vbox);

  Connect(wxID_EXIT, wxEVT_COMMAND_TOOL_CLICKED,
      wxCommandEventHandler(Toolbar::OnQuit));

  Centre();
}

void Toolbar::OnQuit(wxCommandEvent& WXUNUSED(event))
{
  Close(true);
}
```

main.h

```
#include <wx/wx.h>

class MyApp : public wxApp
{
  public:
    virtual bool OnInit();
};
```

main.cpp

```
#include "main.h"
#include "toolbars.h"

IMPLEMENT_APP(MyApp)

bool MyApp::OnInit()
{
    Toolbar *toolbar = new Toolbar(wxT("Toolbar"));
    toolbar->Show(true);
    return true;
}
```

In our example, we create two horizontal toolbars. We place them in a vertical box sizer.

```
toolbar1 = new wxToolBar(this, wxID_ANY);
...
toolbar2 = new wxToolBar(this, wxID_ANY);
```

Here we create two toolbars.

```
vbox->Add(toolbar1, 0, wxEXPAND);
vbox->Add(toolbar2, 0, wxEXPAND);
```

And here we add them to the vertical box sizer.

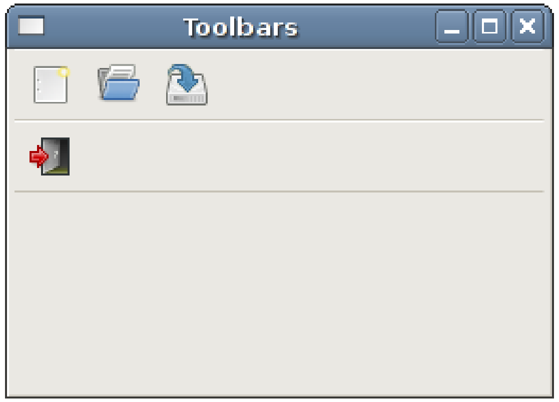

Figure: Toolbars

In this part of the wxWidgets tutorial, we have covered menus and toolbars.

Home ‡ Contents ‡ Top of Page