wxGlade user manual

Contents

List of Figures

Preface

This manual describes Alberto Griggio's wxGlade program, a Python, Perl, C++ and XRC Graphical User Interface ("GUI") Editor for UNIX and Microsoft Windows. Each of the chapters in this manual is designed as a tutorial for using wxGlade and a reference for widgets supported until now.

Abbreviations

THE FOLLOWING ABBREVIATIONS ARE USED IN THIS MANUAL:

X11 The X Window System version 11.

wx The wxWindows/wxWidgets open source C++ GUI framework.

WIN32 The Microsoft Windows 32-bit Application Programmer's Interface.

GUI Graphical User Interface

WYSIWYG What You See Is What You Get.

Contacts

Check the sourceforge project page http://sourceforge.net/projects/wxglade for the mailing list to discuss the project.

Use the lists for questions, proposals, bug reports and collaboration.

If you don't want to follow the list, you can reach the author of the program "Alberto Griggio" at albgrig@tiscalinet.it. Any kind of feedback is always welcome.

Information, support and bug reports can be addressed to the wxGlade mailing list.

The mantainer of this document is Marcello Semboli.

You can address any message regarding this document to "Marcello Semboli" dinogen@siena.linux.it. Please report grammatical errors to me, as English isn't my native language.

Copyrights and Trademarks

wxGlade is Copyright 2002-2003 by Alberto Griggio. Use and distribution of wxGlade is governed by the MIT license, located in Appendix A.

wxWindows/wxWidgets is Copyright (c) 1992-2002 Julian Smart, Robert Roebling, Vadim Zeitlin and other members of the wxWidgets team. See: http://www.wxwidgets.org for details.

UNIX is a registered trademark of the X Open Group, Inc.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Chapter 1

Introduction to wxGlade

1.1 What wxGlade is

wxGlade is a GUI designer written in Python with the popular GUI toolkit wxPython, that helps you create wxWidgets/wxPython user interfaces. At the moment it can generate Python, C++ and XRC (wxWidgets' XML resources) code.

As you can guess by the name, its model is Glade, the famous GTK+/GNOME GUI builder, with which wxGlade shares the philosophy and the look & feel (but not a line of code).

1.2 What wxGlade is NOT

It is not (and will never be) a full featured IDE, but simply a "designer". The generated code does nothing apart from displaying the created widgets. If you are looking for a complete IDE, maybe Boa Constructor http://boa-constructor.sourceforge.net or PythonCard http://www.pythoncard.org is the right tool.

1.3 Download

You can download source files for the stable version from the sourceforge project page http://sourceforge.net/projects/wxglade.

You can get the unstable version from Sourceforge anonymous CVS access. Refer to Sourceforge CVS documentation for details.

Also binaries for Microsoft Windows and Linux are provided for download.

1.4 Installation and requirements

Since Python is a pseudo-interpreted language, you don't need any "compile" or "make" steps.

wxGlade requires Python version 2.2 or later and wxPython version 2.3.2.1 or later.

The binary versions are stand-alone and don't have any requirement.

You can download wxWidgets from http://www.wxwidgets.org and wxPython from http://www.wxpython.org.

In Microsoft Windows, you only need to install wxPython. wxWidgets is not required.

In a Linux/Unix environment you need to install wxPythonGTK, which usually also requires the wxGTK package. Mandrake Linux 9.2 comes with a libwxPython rpm that works, without installing wxGTK.

If you use apt or yum, just install wxPythonGTK package, and let it resolve the dependencies.

1.5 Basics

You need to know the basics of wxWidgets or wxPython, as well as the basics of C++, Python or Perl. You can't use wxGlade if you do not have any basic understanding of programming. You can't learn wx programming from reading this manual either.

1.6 Contacts

Check the sourceforge project page http://sourceforge.net/projects/wxglade for the mailing list to discuss the project.

Use the list for questions, proposals, bug reports and collaboration.

If you don't want to follow the list, you can reach me at albgrig@tiscalinet.it. Any kind of feedback is always welcome.

Chapter 2

Exploring wxGlade

2.1 Quick start

We will design a simple form.

Start wxGlade by running the **wxglade.py** program.

You will see a Main Palette with several buttons, and a Tree Window with an icon marked “Application”. A Properties Window shows the properties of the Application.

If you move the mouse over a button in the main window, a tooltip will display its function.

To add a frame in the design window, from the Main Palette choose the first button: “Add a frame”.

Then choose wxFrame as the base class.

Look at the tree window and see that two icons are generated under the application icon, a frame icon and a sizer icon.

If you double click with the mouse on the frame icon, the designer window appears. Notice that the sizer is displayed as a set of gray boxes: they are the “slots” of the grid sizer where you will place the widgets.

You put a widget on a sizer by selecting it on the Main Window, then click on an empty slot on the frame on the designer window. Try adding a static text, a text control and a button.

If you want to add something else, add empty slots on the sizer by right-clicking on the sizer on the tree window and selecting “Add slot”.

Play around, adding four or five widgets on the frame.

Now look at the properties form; there are three tabs. In the “Common” tab you can specify the name, size and color of the widget.

In the “Layout” tab you can adjust borders and alignments.

In the “Widget” tab you find the properties depending on the widget.

You can select the properties of a widget by clicking on the designer window or the corresponding icon on the tree window.

Try adjusting widgets with the properties form until you know you have played enough.

Now let’s generate the code.

Select the Application icon on the tree window and go to the properties window.

Check Name and Class, choose a Top window, check Single file and choose the language and set the Output path by pushing the button for selecting a path and a filename.

Finally press the Generate code button, and the code is generated.

Compile and enjoy.

2.2 Basics of wxGlade

The program wxGlade is a tool for designing Graphical User Interfaces (GUI).

It is intended to be used with the wxWidgets framework in all its flavors: C++, Perl, Python and XRC.

You use a visual editor for creating forms, menus and toolbars with the mouse.

Your design is saved in a .wxg file, which is the wxGlade file format.

Then you generate source code or XRC by using visual tools or invoking wxGlade at the command line.

You can also use wxGlade in your makefile by generating source code only when the .wxg changes.

A .wxg file can contain multiple forms, panels, menus and toolbars and generate either a single file containing all classes or multiple files containing one class each.

wxGlade does not manage events, file inclusion, function names, stubs or anything else but graphic interface code.

2.3 Command line invocation

You can run wxGlade without parameters to start the GUI on an empty application as follows:

wxglade.py

Run wxGlade GUI on an existing application specifying the .wxg file as follow:

wxglade.py FILENAME.wxg

If you only want to generate the code without starting the GUI, use the -g or --generate-code option with the language as argument as follows:

wxglade.py -g LANG FILENAME.wxg

wxglade.py --generate-cod=LANG FILENAME.wxg

Possible values for LANG are "python", "XRC", "perl" or "C++".

You can also specify the destination of the generated code with -o or --output option:

wxglade.py -g LANG -o DESTINATION FILENAME.wxg

The DESTINATION argument can be a file or a directory. It is a file when the FILENAME.wxg generates single-file source code. It is a directory when the FILENAME.wxg generates multiple-file source code.

This is the complete description of the command line:

```
wxglade.py [OPTIONS] [FILENAME.wxg]
OPTIONS:

-g LANG, --generate-code=LANG
Generate the code in the LANG language. Possible values for LANG are C++, python, perl, XR

-o DEST, --output=DEST
Only valid with the -g option. Specify the destination for the code generation.
The argument DEST can be a source code file or a directory.
```

2.4 Using the source code

How do you use source code generated by wxGlade? Basically in two ways.

You can safely edit the source code of the generated class. This is because wxGlade marks the untouchable code with the special comments “begin wxGlade” and “end wxGlade”.

So you can edit all you need outside these two tags. When you make changes in your forms, a new code generation will not modify the user code.

The other way is to generate the class with wxGlade in a file (e.g. myFormUI.xxx) and subclass it in another file (e.g. myForm.xxx). This gives you the psychological advantage that there is a file for you and one for wxGlade :-) . That way if you are stuck writing messy code you can erase your entire source file.

There is an annoying problem in wxWidgets: you can put widgets directly on a wxFrame or in a wxPanel that belongs to the wxFrame. If you choose the first option, in Microsoft Windows the frame looks quite odd.

Since wxGlade doesn't automatically generate a wxPanel within a wxFrame, a good way of working is to design a wxPanel with wxGlade (by choosing "dialog" in the main toolbar) and import it into the wxFrame. This gives you the double advantage of separating code and UI, as well as preserving the wxFrame-wxPanel-Widgets structure.

2.5 Specifying the path of bitmaps

In wxGlade some widgets need to specify a bitmap path. You can use any graphic format supported by wxWidgets.

The bitmap can be specified in several ways:

Usually you can type an absolute path in a textbox or browse for a bitmap with a file dialog. This will produce a wxBitmap object with the typed string as bitmap path (e.g. *wxBitmap("/usr/share/icons/application.png", wxBITMAP_TYPE_ANY)*)

You can enter a variable name using the *var:* tag in the textbox. This will produce a wxBitmap object with the variable name as bitmap path (e.g. *var:my_bitmap_path* produces *wxBitmap(my_bitmap_path, wxBITMAP_TYPE_ANY)*). In perl code generation a “$” sign is added if you omit it.

You can enter a code chunk returning a wxBitmap, by using the *code:* tag. This inserts verbatum the code you enter in brackets and nothing more (e.g.: if wxSomeWidget needs a wxBitmap as an argument, the string *code:if (x == 0) get_bitmap1() else get_bitmap2();* produces *wxSomeWidget((if (x == 0) get_bitmap1() else get_bitmap2();), option1, option2)*).

wxGlade never declares or assigns variable or function names, so after code generation, you have to provide extra code to declare your variables or functions.

If you use *var*: or *code:* tags the preview window shows an empty bitmap of fixed size.

Chapter 3

wxGlade User Interface

3.1 Main Palette

The main window is a palette that hosts the menu and the widget choice buttons.

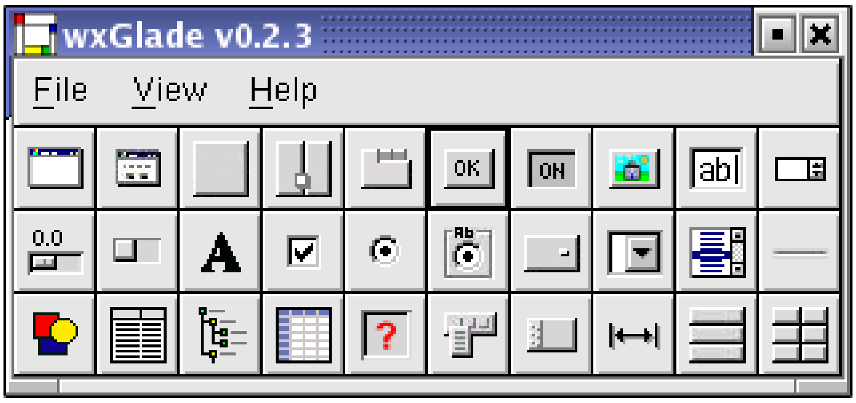


Figure 3.1: The Main Palette

If you pass the mouse pointer over a button a tooltip shows the button’s description.

The “Add a Frame” button and the “Add a Dialog/Panel” button bring up a dialog to add a frame, a dialog or a panel to your project.

The “Add a MenuBar” button asks you for the name of the class then adds a menu bar to your project.

The “Add a ToolBar” button asks you for the name of the class then adds a toolbar to your project.

The other buttons in the main window add widgets to a form. When you click on one, the mouse pointer changes to an arrow. Then you can click on a sizer’s empty cell to add the widget to it.

3.2 Tree Window

The tree window shows the logical hierarchy of widgets and its child-widgets. For example you can see a panel as a tree’s node and the widgets on it as child nodes.

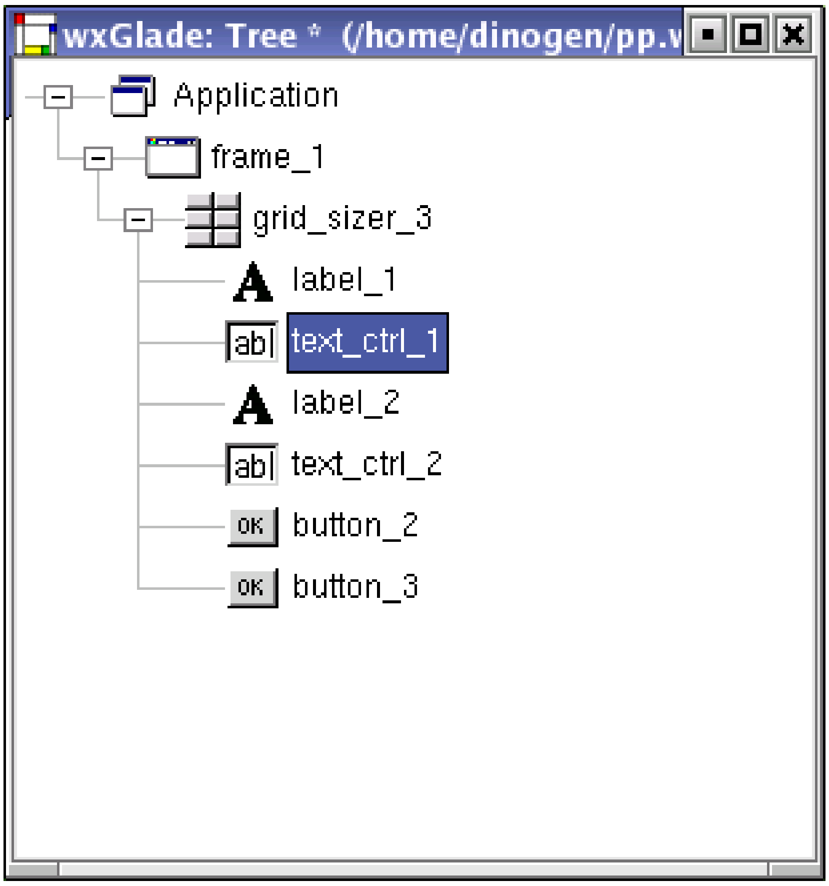


Figure 3.2: The Tree Window

You can show or hide the tree window by the menu item View/Show Tree or with the Ctrl-T shortcut.

Usually a frame or a panel contains a sizer, so you often see a sort of panel-sizer-widgets structure. The tree gets more complex when you nest sizers within sizers.

You can navigate the visual presentation of your widget tree by mouse, expand and collapse sizers, and copy, cut or remove widgets.

A click on an icon in the tree window displays the properties of the corresponding element in the properties window. A double click in a frame, dialog or panel icon, makes the designer window show it as it appears. Clicking with the right button of the mouse gives you a pop-up menu.

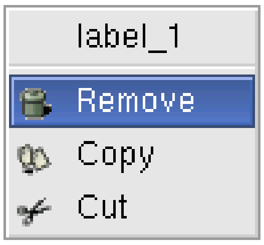


Figure 3.3: The menu for a widget

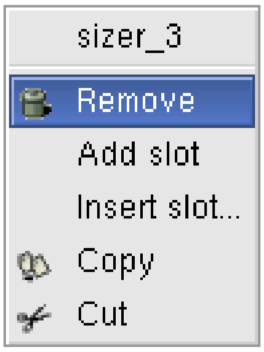


Figure 3.4: The menu for a sizer

The pop-up menu for a widget allows you to copy, cut or remove the element. The pop-up menu for a sizer allows you to copy, cut or remove the element, or add or insert an empty slot. *Note*: Often when you add an empty slot, you have to make the designer window larger, to show the new slot.

3.3 Design Window

The design window shows the frame or panel you are creating in WYSIWYG mode and allows you to select a widget from the main palette and to put it on an empty slot of a sizer. You can show the design window by double-clicking on the icon of a frame or dialog in the tree window.

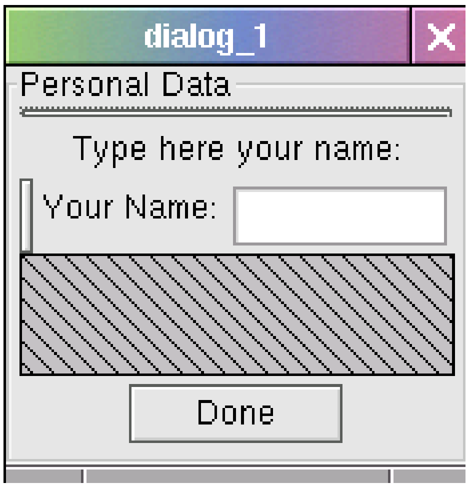


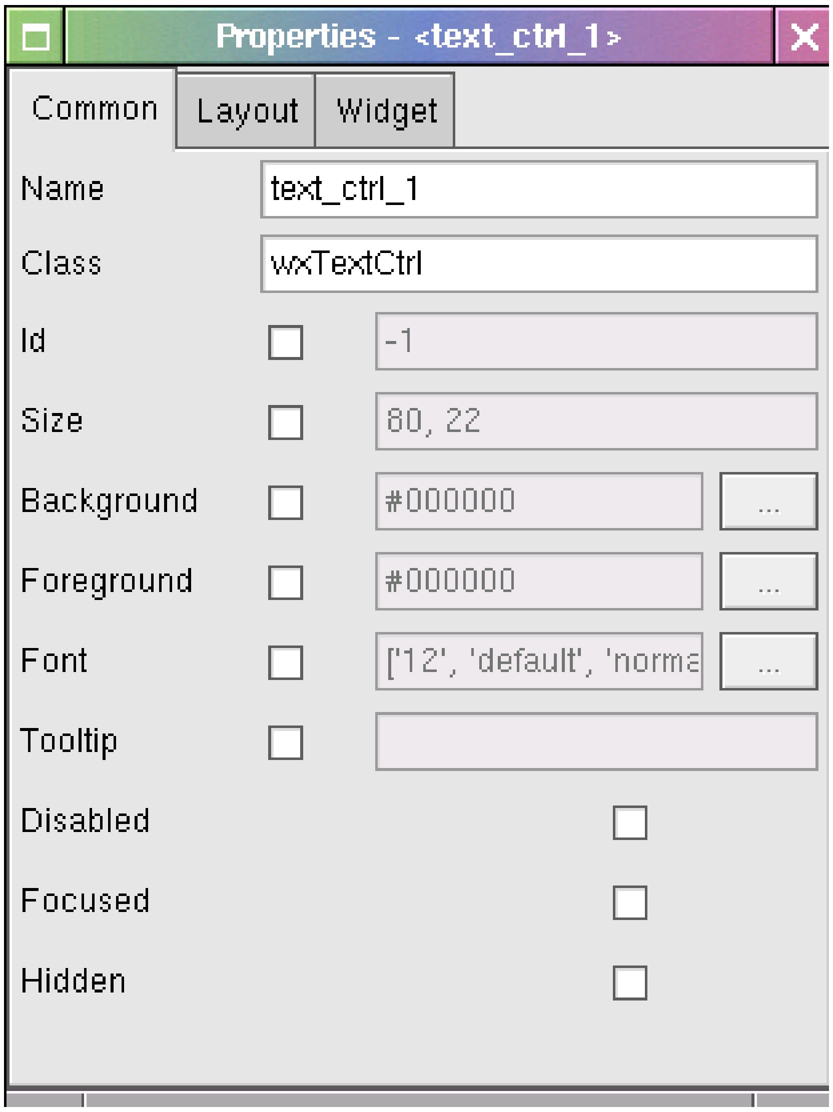
Figure 3.5: The Design Window

By clicking with the right mouse button on a widget you can access the context menu. Notice that the sizers, which are invisible elements, have a little gray “handle,” that you can click to select the sizer or let the pop-up menu appear.

The pop-up menu is the same as the one you get in the Tree Window, as shown in Figure 3.3 or in Figure 3.4.

3.4 Properties Window

The properties window lets you see and edit the properties that apply to the selected element. This window consists of three tabs. The first tab contains the common properties that apply to all widgets.

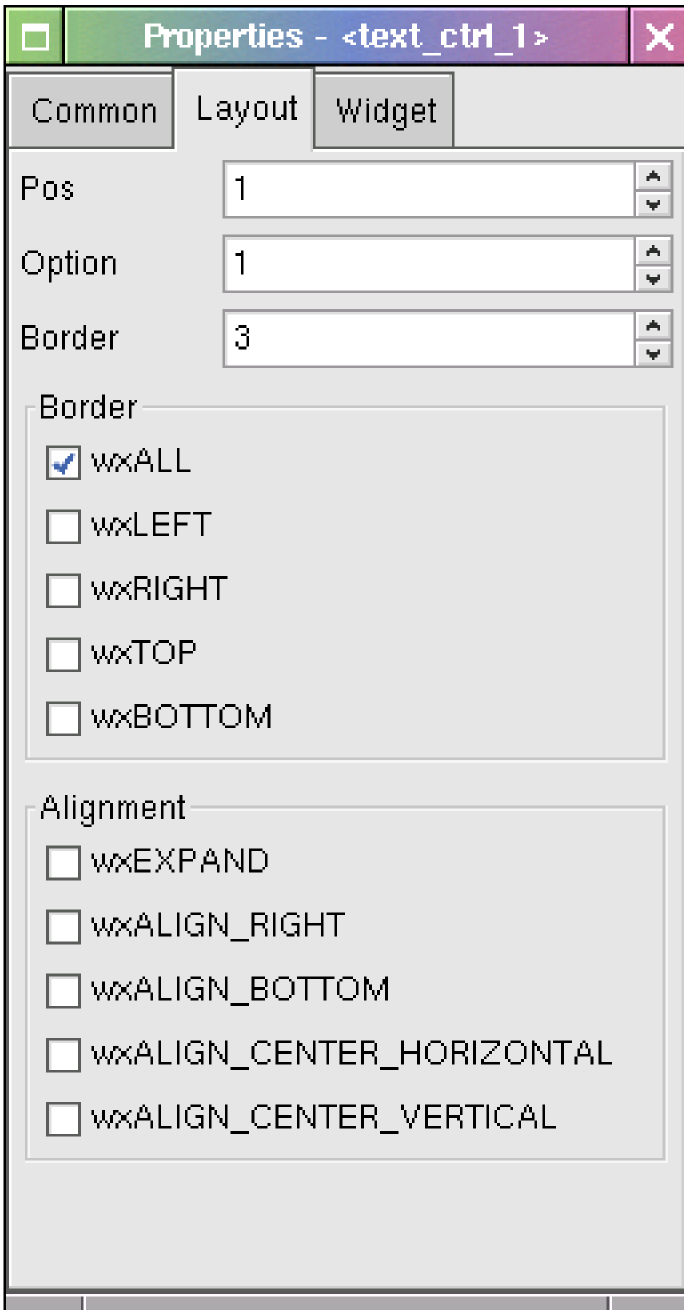


Name		text_ctrl_1
Class		wxTextCtrl
Id	☐	-1
Size	☐	80, 22
Background	☐	#000000
Foreground	☐	#000000
Font	☐	['12', 'default', 'norma
Tooltip	☐	
Disabled	☐	
Focused	☐	
Hidden	☐	

Figure 3.6: Common Properties

As shown in Figure 3.6 the common properties are related to name, size, colors and fonts.

The second tab is related to layout properties that control position and resizing within the sizer.



Pos 1

Option 1

Border 3

Border

[✓ wxALL]
wxLEFT
wxRIGHT
wxTOP
wxBOTTOM

Alignment

wxEXPAND
wxALIGN_RIGHT
wxALIGN_BOTTOM
wxALIGN_CENTER_HORIZONTAL
wxALIGN_CENTER_VERTICAL

Figure 3.7: Layout Properties

These properties apply to any widget. You can check or uncheck any option related to the placement in the sizer. Many widgets may have a default value of 3 in the "Border" property in the Preferences Dialog (see [?title]). If you let a widget have a default border, the `wxALL` option is also checked.

The third tab, named “Widget,” is different for each widget, and lets you edit properties for the specific element you have selected.

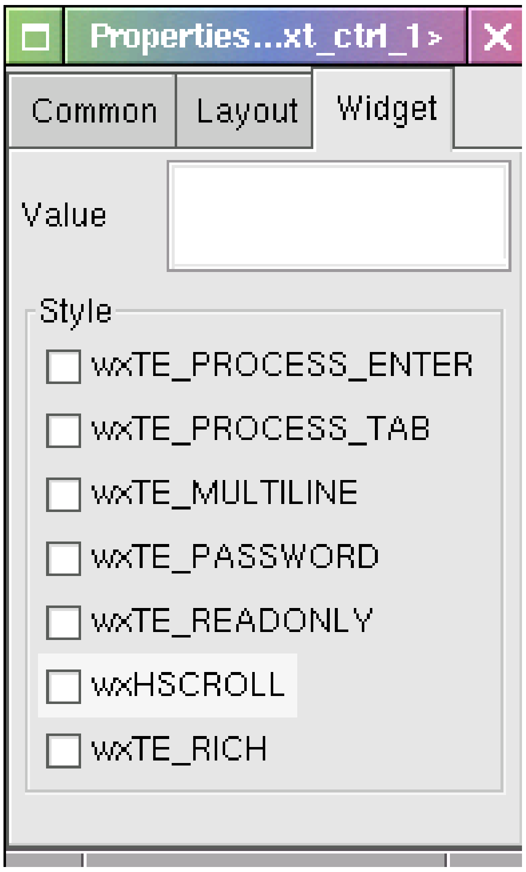


Figure 3.8: Widget Properties

The set of options may also be quite complex in the case of widgets that have a great deal of methods and properties (such as grids and treeviews). In this case, wxGlade greatly simplifies the process of designing forms.

You can show or hide the properties window by the menu item View → Show Properties (**Ctrl-P**).

3.4.1 Editing properties

Editing properties is quite simple; Properties are represented by buttons, text boxes and checks. Usually they are referenced by the same name or symbol that you find writing C++ code.

Usually you get the changes in the design window in real time. In some cases you have to push the "Apply" button. For example, the wxNotebook widget shows in its properties window a list of child wxPanels. You have to press the apply button to show changes you make when you add or remove panels.

3.5 Preferences Dialog

You can access the Preferences Dialog with the menu item View → Preferences. You can choose some decoration options, like whether to show icons in menus or not, but also something more effective. For example, you can modify the number of buttons in the Main Palette. If you type a value of 15 or 30, you get a long toolbar-like Main Palette. You can also choose the default path where you save wxGlade files or generate source code.

Another useful option is to enable a default border of 3 around some widgets. In many cases this can be useful to have set.

You need to restart wxGlade for changes to take effect.

3.6 The wxGlade Menu

wxGlade has only a few very small menus.

3.6.1 The FILE menu

In the FILE menu there are the classic File → New, File → Open... and File → Save items. When opening or saving a new file, the file dialog defaults to the directory that you put in the “Initial path” textbox in the Preferences dialog, usually the user home directory.

The File → Generate code item produces the code from the current design.

3.6.2 The VIEW menu

In the VIEW menu, you can show or hide the tree window and the properties window.

In this menu you access the Preferences Dialog as well.

Chapter 4

Supported widgets

4.1 Introduction

wxGlade supports a number of widgets and helps you to edit the properties and visual look of each one.

4.2 Widget list

Follow the widget list as it appears in the wxGlade main window.

4.2.1 Frame

This prompts for a wxFrame or a wxMDIChildFrame. A vertical wxBoxSizer is appended. In the properties window you can choose the styles and you can add an icon.

4.2.2 Dialog or Panel

This prompts for a wxDialog or a wxPanel in top level. In the properties window you can choose the styles and, for the dialog, you can add an icon.

4.2.3 Panel

This allows you to add a panel to a sizer.

In the properties window you can choose the styles.

4.2.4 Splitter window

This produces a wxSplitterWindow and two associated panels as well. You can choose vertical or horizontal splitting.

In the properties window you can choose the styles and the sash position.

Be careful not to put too large a widget in a splitter panel, because while it might appear normal in the design window, when you run your program one of two panels will take all the available space and the other will shrink to the minimum size possible.

4.2.5 Notebook

This produces a wxNotebook and one panel for each tab.

In the properties window you can add and remove tabs, which appear in a list.

Don’t forget to click on the “Apply” button to transfer changes that you have made in the list to the design window.

4.2.6 Button

This produces a wxButton. You can enter a caption and the “default” flag. If you want to add an image you need a bitmap button (see [?title]).

4.2.7 Toggle button

This produces a wxToggleButton. You can enter a caption and the status (clicked or not) of the button.

4.2.8 Bitmap button

This produces a wxBitmapButton. You can set the “default” flag on or off. You also can choose the bitmap for the button and, optionally, the bitmap for the disabled status. Refer to [?title] for bitmap path specifications.

4.2.9 Text control

This produces a wxTextCtrl. In the properties window you can enter the text and also set the style.

4.2.10 Spin control

This produces a wxSpinCtrl. In the properties window you can enter the value, the range and also set the style.

4.2.11 Slider

This produces a wxSlider. In the properties window you can enter the value, the range and also set the style.

4.2.12 Gauge

This produces a wxGauge. In the properties window you can enter the range and set the style.

4.2.13 Static text

This produces a wxStaticText. In the properties window you can enter the text, set the style and tell wxGlade whether to store the control as an attribute.

4.2.14 Check box

This produces a wxCheckBox. In the properties window you can enter the text, and the status, checked or not, of the button.

4.2.15 Radio button

This produces a wxRadioButton. In the properties window you can enter the text, and the status, clicked or not, and the style.

4.2.16 Radio box

This produces a wxRadioBox. In the properties window you can enter the dimension. The style determines whether the dimension is the number of rows or columns.

You also can set which button is selected with the “Selection” spin starting from 0. You can edit the list of choices, but remember to click on the “Apply” button to consolidate changes.

4.2.17 Choice

This produces a wxChoice. In the properties window you can enter the position of the selected item starting from 0. You can edit the list of choices, but remember to click on the “Apply” button to consolidate changes.

4.2.18 Combo Box

This produces a wxComboBox. In the properties window you can enter the position of the selected item starting from 0. You can edit the list of choices, but remember to click on the “Apply” button to consolidate changes.

4.2.19 List Box

This produces a wxListBox. In the properties window you can enter the position of the selected item starting from 0. You can edit the list of choices, but remember to click on the “Apply” button to consolidate changes.

4.2.20 StaticLine

This produces a vertical or horizontal wxStaticLine. In the properties window you can tell wxGlade whether to store the object as an attribute of the frame class.

4.2.21 Static bitmap

This produces a wxStaticBitmap. You will be prompted for the bitmap path. Refer to [?title] for bitmap path specifications. In the properties window you can set the style and you can tell wxGlade whether to store the object as an attribute of the frame class.

4.2.22 List Control

This produces a wxListCtrl. In the properties window you can set the style.

4.2.23 Tree Control

This produces a wxTreeCtrl. In the properties window you can set the style.

4.2.24 Grid

This produces a wxGrid. In the properties window you can set the style, the row number, the label size, the line and background color and the selection mode. You can edit the list of columns, but remember to click on the “Apply” button to consolidate changes. Also you can choose to let wxGlade to create the grid or leave it to the user code.

4.2.25 Custom Widget

When you put a custom widget in the design window you will be prompted for a class name. In the properties window you can set a number of custom attributes that will appear in the costructor call. These attributes have different effects in C++, perl, python or XRC code generation. Four special attributes, *$id, $parent, $width and $height* each return the value you specify in the “common” tab of the custom widget.

4.2.26 Spacer

When you put a spacer into a sizer slot in the design window you will be prompted for the size; wxGlade will generate the code to set an empty space in that slot of the sizer.

Chapter 5

Menu, Toolbar and Statusbar

5.1 Introduction

wxGlade helps you to design the menu and the toolbar for your application.

You can create the menu and toolbar as stand alone classes by clicking the corresponding button in the main window.

Alternatively you can make the menu, toolbar and statusbar associated with a wxFrame, by selecting the related checkboxes in the wxFrame properties window.

5.2 Menu

In the menu properties window click on the “Edit menus...” button. A dialog will let you edit your menu. Use the “Add” button to add items to the menu; enter the label, an optional name and help string. You can use numbers or variable names as the item id. If you use a variable name, you have to provide extra code in the generated source code.

Choose the type of the item: Normal, Checkable or Radio.

You can move menu items with “Up” and “Down” buttons, and you can modify the hierarchy of the menu with “<” and “>” buttons.

5.3 Toolbar

You can edit the Toolbar’s style and bitmap size in the properties window.

Click on the “Edit tools...” button to edit the toolbar buttons. Use the “Add” button to add buttons to the toolbar; enter the label, an optional name and help string. You can use numbers or variable names as the button id. If you use a variable name, you have to provide extra code in the generated source code.

Choose the type of the button: Normal, Checkable or Radio.

You can move toolbar buttons with “Up” and “Down” buttons.

You have to enter two bitmaps, one for normal status and the other for the pushed status.

Refer to [?title] for bitmap path specifications.

5.4 Statusbar

In the properties window you can edit the list of fields and their size, but remember to click on the “Apply” button to consolidate changes.

Appendix A

Appendix

Copyright (c) 2002-2003 Alberto Griggio albgrig@tiscalinet.it.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.

IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.