Using Glade to create GTK+ Applications with FORTH.

euroFORTH 2010 - manfred.mahlow@forth-ev.de

September 8, 2010

Abstract

When talking about GUI development with FORTH, one of the most expressed desires is to have an IDE or a graphical editor. Some years ago, when I wrote an object-oriented GTK+ interface for cspForth, an IDE or graphical editor was not my concern. But, when I recently decided to port the GTK+ interface to MINFORTH, I remembered that often expressed desire and had a look at Glade and found that Glade and FORTH can be quite good companions.

What is Glade ?

Glade is a graphical user interface builder for GTK+. It's neither an IDE nor a code editor. It allows to design graphical user interfaces saved as XML files. The XML files describe the layout of the GUI, the properties of the widgets and the signal handling. Since version 3.5.0 two file formats are supported, an older one to be used with Libglade and a newer one for Libgtk.

Using a GUI created with Glade is not a tough task. The usual steps to write the required program code are

1. Decision what library to use, Libglade or Libgtk.
2. Initializing GTK+.
3. Creating a GladeXML or a GtkBuilder object per widget hierarchy.
4. Loading the widget hierarchies from the XMF file into this objects.
5. Reading widget identifiers from the GladeXML or GtkBuilder objects.
6. Writing signal handlers.
7. Assigning signal handlers to widgets.
8. Displaying the GUI.
9. Starting the GTK+ main loop.

A GUI created with Glade can be used with any program language, as long as the language gives access to the required Libgtk or Libglade functions.

Glade and FORTH

FORTH and Glade can be great companions. Let's have a quick look at a small example, written for MINFORTH 1.5(p).[1]

An Example ...

The Graphical User Interface

Our starting point is a GUI created with Glade. The newer GtkBuilder format is used. The GUI consists of two widget hierarchies, the main application window (window1 in Fig. 1) and a modal dialog (dialog1 in Fig. 2).

Fig. 1 and Fig. 2 show the same Glade instance. The only difference is the selected widget. In Fig. 1 it's window1, in Fig. 2 it's dialog1.

The main window is a GtkWindow with only one child, which is a GtkLabel (label1). The dialog is a GtkDialog widget and has some more children, packed into container widgets. A GtkImage (image1) and a GtkLabel (label2) are packed into a GtkHBox (hbox2) which is packed into the dialogs internal GtkVBox (dialog-vbox1) and two GtkButton widgets (button1, button2) are packed into the dialogs internal GtkHBox (dialog-action_area1).

All packing has been done and all widget properties have been set with Glade. The only thing that can not be done when using FORTH, is to assign signal handlers with Glade. This is only possible when using the program language C.

The GUI specification is stored in XML format (Fig. 3) to be used with the GtkBuilder object defined in Libgtk.

[1] MINFORTH 1.5(p) is MINFORTH 1.5 (for LINUX) with some additional hooks and minor extensions for the object-oriented GTK+ interface.

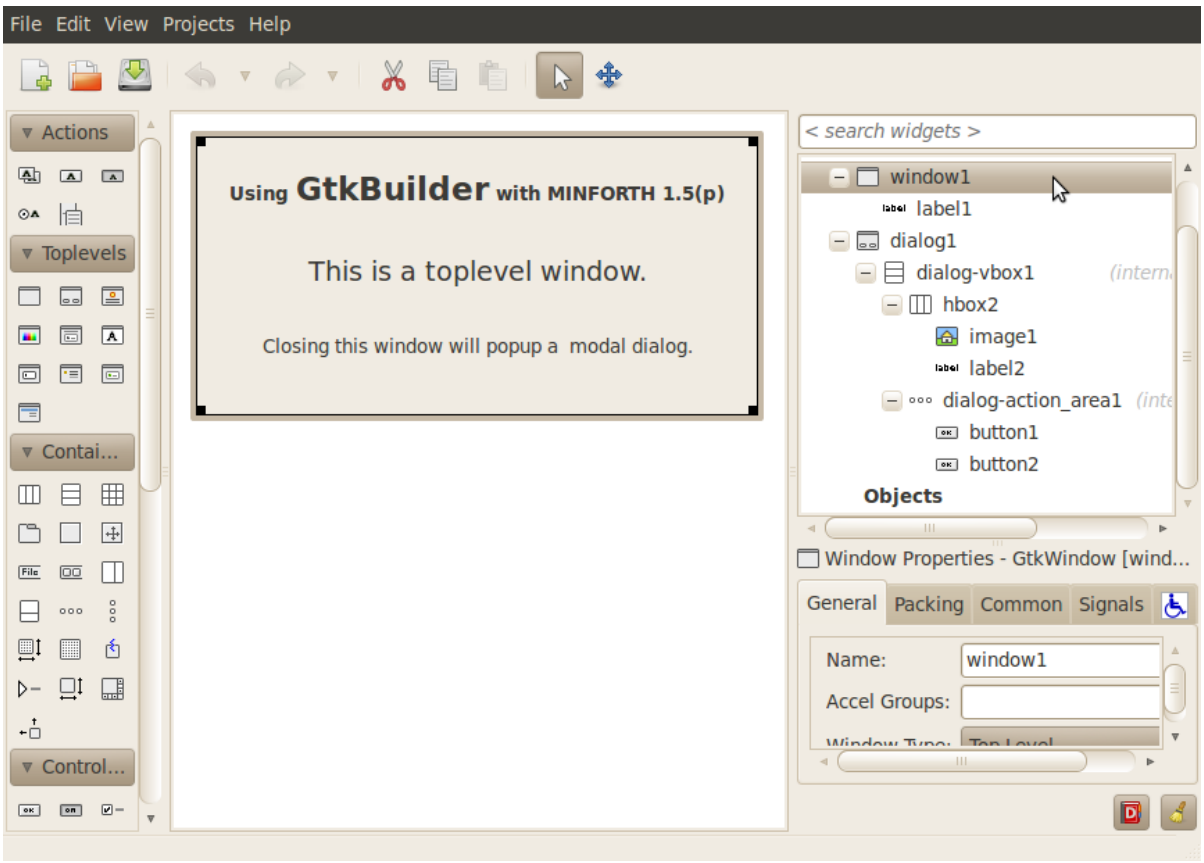

Figure 1: Glade GUI editor, window1 selected

The GTK+ Interface

To transform the GUI into a GUI application, some program code is required. We'll use an object-oriented GTK+ interface here.

The GTK+ classes are mapped to classes of the same name in FORTH. Classes are loaded on demand as required.

A widget is an instance of its widget class. The methods to create and initialize a widget and to modify its properties are defined in its class.

Widget properties are implemented as instance variables. Property and method names are choosen to be close to the corresponding GTK+ names.

The FORTH Code

Now lets have a look at the program code in Fig. 4.

The required classes are loaded in line 8 to 11.

A String object is needed for the name of the XML file, a GtkBuilder object to load the GUI specification from the file and a GtkDialog and a GtkWindow object to get access to the GUIs main window and to the dialog. The objects are created in line 15 to 18.

In line 18 and 19 two signal slots are created to be used to connect signals and signal handlers to the main window(window1).

A first signal handler is defined in line 23. It is called when a 'destroy' signal is received by the main window (window1). It's very simple here. Its only task is to terminate the GTK+ event processing by leaving the GTK+ main loop. It's called with two parameters. Both are not used here.

The second signal handler is defined in line 25 to 27. It is called, when the main window(window1) receives a 'delete-event' signal from the window manager. Its task is to open the dialog1, wait for a button press, destroy the dialog when a button is pressed and return a button-specific value. A 'destroy' signal will be send to the main window (window1) if FALSE is returned. This is the case if the 'QUIT' button (button2) was pressed. Otherwise no 'destroy' signal will be emitted and the 'delete-event' signal will have no further effect.

In line 26 the GtkBuilder object (builder) is initialized from the XML file with the GUI specification for the dialog (dialog1). All widgets of the dialogs widget hierarchy are created here. Then, in line 27, the GtkDialog object for dialog1 is initialized by reading its widget identifier from the builder object and the dialog is shown

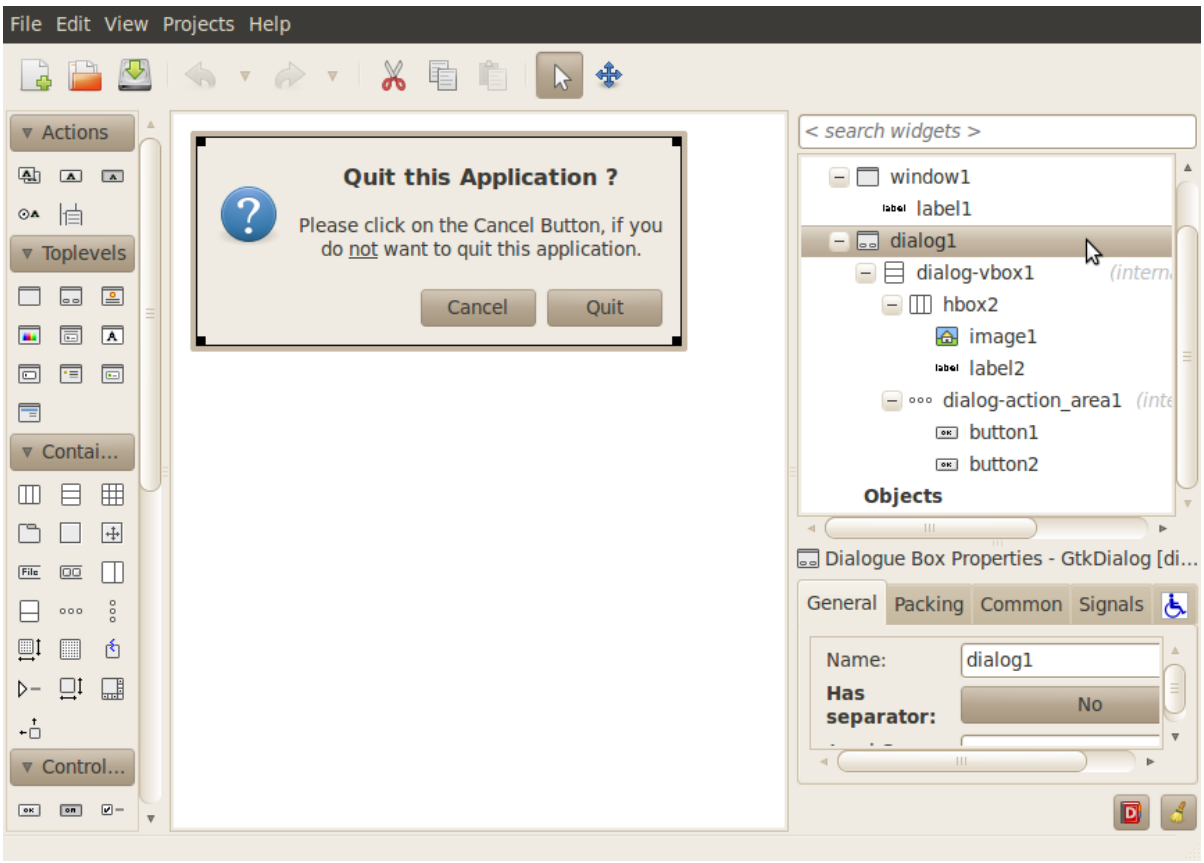

Figure 2: Glade GUI editor, dialog1 selected

to the user (dialog1 run), waiting for the user to press one of the dialog buttons.

In line 29 to 34 the word to start the application is defined.

In line 30 the GtkBuilder object (builder) is initialized from the XML file with the GUI specification for the main window (window1). All widgets of the main windows widget hierarchy are created here.

In line 31 the GtkWindow object for the main window (window1) is initialized by reading its widget identifier from the builder object.

Line 32 connects the on.destroy signal handler from line 23 to the 'destroy' signal at window1, using the signal slot cb.destroy and line 33 connects the on.delete-event signal handler from line 25 to the 'delete-event' signal at window1, using the signal slot cb.delete.

The code in line 34 makes the main window (window1) visible on the computer display and, finally, the application is started in line 36.

Two modes of event processing are supported here. If MINFORTH runs in a terminal window, the GTK+ events are processed in the background while waiting for terminal input, to preserve FORTHs interactivity. Otherwise a GTK+ main loop is entered for event processing.

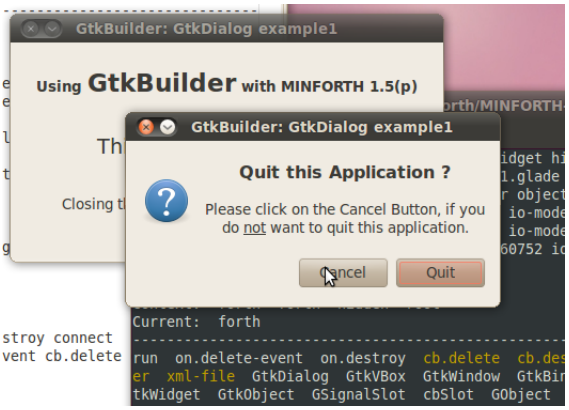

Figure 5: The running GUI example.

Up and Running

Fig. 5 shows the running application with the dialog waiting for user response after the close button of the main window was clicked.

The Benefit of using Glade

The advantage of creating a GUI with Glade instead of creating it from source code is, that no code needs to be written to create the widgets, to set the widget properties and to pack the widgets into container widgets to get the

```
<?xml version="1.0"?>
<interface>
  <requires lib="gtk+" version="2.16"/>
  <!-- interface-naming-policy project-wide -->
  <object class="GtkWindow" id="window1">
    <property name="border_width">24</property>
    <property name="title" translatable="yes">GtkBuilder: GtkDialog example1</
property>
    <property name="resizable">False</property>
    <property name="window_position">center</property>
    <child>
      <object class="GtkLabel" id="label1">
        <property name="visible">True</property>
        <property name="label">&lt;b&gt;Using &lt;span size='xx-
large'&gt;GtkBuilder&lt;/span&gt; with MINFORTH 1.5(p)&lt;/b&gt;


&lt;span size='x-large'&gt;This is a toplevel window.&lt;/span&gt;


Closing this window will popup a  modal dialog.
</property>
```

Figure 3: First twenty lines of Glades XML output.

```
\  euroFORTH-2010/GtkBuilder/example1.mf
\ ------------------------------------------------------------------------------
\  GtkBuilder              OOP Library for MINFORTH                  MM-100801
\ ------------------------------------------------------------------------------
\               Copyright (C) 2010 manfred.mahlow@forth-ev.de
\
\ ------------------------------------------------------------------------------
   requires String
   requires GtkBuilder
   requires GtkWindow
   requires GtkDialog

   forth definitions decimal
\ ------------------------------------------------------------------------------
   String     new xml-file
   GtkBuilder new builder
   GtkDialog  new dialog1
   GtkWindow  new window1            GSignalSlot new cb.destroy
                                     GSignalSlot new cb.delete

   s" euroFORTH-2010/GtkBuilder/example1.glade" xml-file !

   : on.destroy ( data oid -- ) 2drop gtk main_quit ;

   : on.delete-event ( event data oid -- f ) 2drop drop
     xml-file @ s" dialog1" builder init
     dialog1 init_from_builder  dialog1 run  dialog1 destroy ;

   : run ( -- )
     xml-file @ s" window1" builder init
     window1 init_from_builder
     ['] on.destroy 0 window1 signal destroy cb.destroy connect
     ['] on.delete-event 0 window1 signal delete-event cb.delete connect
     window1 show_all ;

   run term? [if] cr ?? [else] gtk main bye [then]
\ ------------------------------------------------------------------------------
\ Last revision: MM-100903
```

Figure 4: The FORTH code for the GUI created with Glade.

desired layout. Widgets that do not need to be manipulated by the program require no code at all.

Another advantage is that the GUI can be changed aesthetically and widget properties can be changed, without the need to change the code. The only restriction is not to change the widget names.

To see the benefit of using Glade take a look at Fig. 6 and 7. It's the listing of the code that is required to create the same small GUI example as in Fig. 4 but without using Glade.

Obviously the difference is significant and can be expected to be much more significant when writing real applications instead of small examples.

And - the same is true when using Libglade instead of Libgtk. The advantage of Libgtk is, that Libglade is not required at runtime.

References

[1] Andrew Krause. Foundations of GTK+ Development. Apress, 2007.

[2] Matthias Warkus. Das GTK+/GNOME Entwicklerhandbuch. dpunkt.verlag, 2008.

[3] http://www.gtk.org/documentation.html

```
\  euroFORTH-2010/GtkDialog/example1.mf
\ ------------------------------------------------------------------------------
\  GtkDialog example1      OOP Library for MINFORTH                  MM-100801
\ ------------------------------------------------------------------------------
\              Copyright (C) 2010 manfred.mahlow@forth-ev.de
\
\ ------------------------------------------------------------------------------
   requires GtkToplevel
   requires GtkDialog
   requires GtkHBox
   requires GtkImageFromStock
   requires GtkLabel

   forth definitions decimal
\ ------------------------------------------------------------------------------
   GtkDialog   new dialog1
   GtkHBox     new  hbox2
   GtkImage    new   image1
   GtkLabel    new    label2
   String      new    markup2
   GtkLabel    new  label1
   String      new  markup1
   GtkToplevel new window1          GSignalSlot new cb.delete-event
                                    GSignalSlot new cb.destroy

   176 chars markup1 init
   s" <b>Using <span size='xx-large'>GTKBuilder</span> " markup1 !
   s" with MINFORTH 1.5(p)</b>" markup1 +!cr
   s" " markup1 +!cr
   s" <span size='x-large'>This is a toplevel window.</span>" markup1 +!cr
   s" " markup1 +!cr
   s" Closing this window will popup a modal dialog" markup1 +!

   118 chars markup2 init
   s" <b>Quit this Application ?</b>" markup2 !cr
   s" " markup2 +!cr
   s" Please click on the Cancel Button, if you" markup2 +!cr
   s" do <u>not</u> want to quit this application." markup2 +!

   : destroy ( data oid -- ) 2drop  gtk main_quit ;

   : delete-event ( event data oid -- f )
     nip nip \ event and data are not used here
     s" GtkDialog example1" dialog1 init  GtkWindow @ dialog1 transient-for !
     6 dialog1 border-width !  dialog1 resizable no
```

Figure 6: FORTH code to create the GUI without using Glade, page 1.

```
     false 6 hbox2 init dialog1 vbox pack_start_defaults
     s" gtk-dialog-question" image1 from-stock dialog-size init
     hbox2 pack_start_defaults  6 image1 xpad !
     markup2 @ label2 init hbox2 pack_end_defaults
     label2 use-markup yes  label2 justify center  12 label2 ypad !

     s" gtk-cancel" -4  dialog1 add_button
     s" gtk-quit" false dialog1 add_button
     false dialog1 default-response !  dialog1 run  dialog1 destroy ;

   : run ( -- )
     s" GtkDialog example1" window1 init
     24 window1 border-width !  window1 position center

     markup1 @ label1 init window1 add
     label1 use-markup yes  label1 wrap yes  label1 justify center

     ['] delete-event 0 window1 signal delete-event cb.delete-event connect
     ['] destroy 0 window1 signal destroy cb.destroy connect

     window1 show_all ;

  run term? [if] cr ?? [else] gtk main bye [then]
\ ------------------------------------------------------------------------------
\ Last revision: MM-100903
```

Figure 7: FORTH code to create the GUI without using Glade, page 2.