Intro to PyGTK and Glade

P. TenHoopen - WMLUG

What is GTK+?

GTK+ is a toolkit for creating GUIs (graphical user interfaces). It originally was called the GIMP Tool Kit because it was developed for use by the GIMP application.

It is cross-platform and written in C. It has bindings to many other programming languages such as Python.

The current version is 2.16.0 and is licensed under the LPGL.

http://www.gtk.org/

What is PyGTK?

PyGTK is a binding of Python to GTK+.

It is used to create cross-platform GUIs for programs written in Python. It uses the GTK+ library.

The current version is 2.14.1 and is licensed under the LPGL.

http://www.pygtk.org/

What is Glade?

Glade is a graphical user interface designer for GTK+.

The interface files are saved in XML format.

The current version is 3.6.0 and is licensed under the GPL.

http://glade.gnome.org/

Glade GUI Designing

Glade uses a drag-and-drop interface for designing GUIs. It uses a system for placing widgets within containers on the window that allows for the widgets to rearrange themselves when the window is resized. A fixed layout system can also be used if desired.

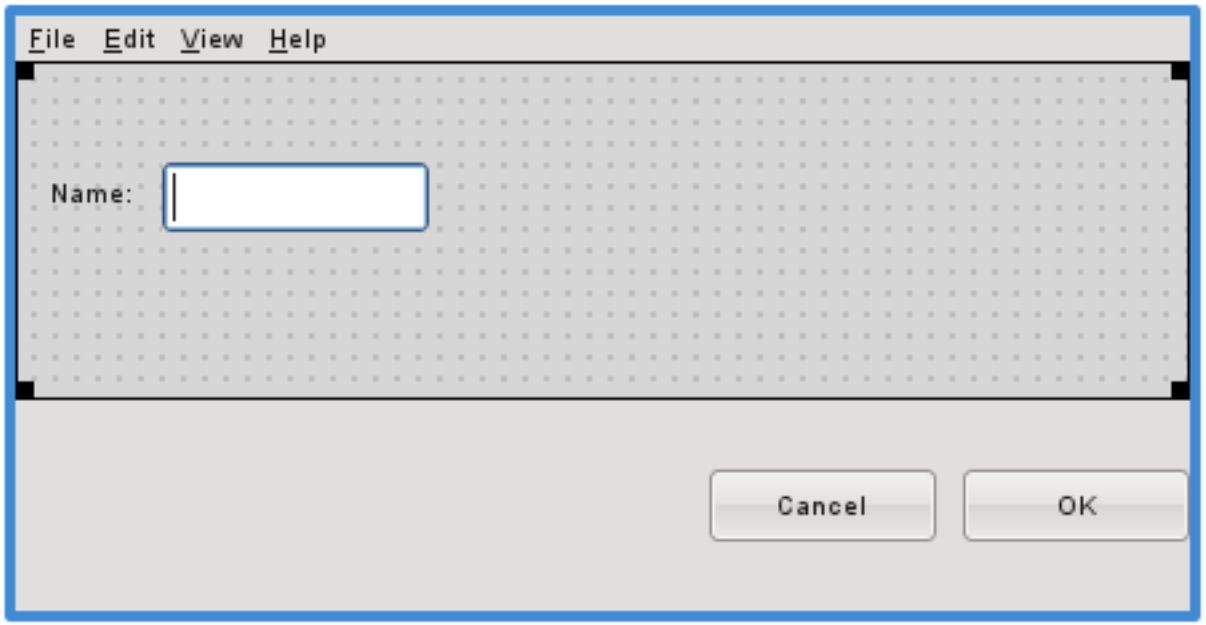
You start by adding a toplevel, then add containers (boxes), and finally adding control and display elements (widgets).

Toplevels: Window, Dialog Box, About Box, etc.

Containers: Horiz/Vert Box, Menu Bar, Tool Bar, Horiz/Vert Button Box, etc.

Control and Display: Buttons, Check Box, Label, Text Entry, etc.

Glade GUI Example



Using Glade

Glade Demo

Glade XML Excerpt

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE glade-interface SYSTEM "glade-2.0.dtd">
<!--Generated with glade3 3.4.5 on Thu Mar  5 06:26:42 2009 -->
<glade-interface>
  <widget class="GtkWindow" id="windowFileStats">
    <property name="visible">True</property>
    <signal name="destroy_event" handler="on_windowFileStats_destroy_event"/>
    <child>
      <widget class="GtkFixed" id="fixed1">
        <property name="visible">True</property>
        <child>
          <widget class="GtkButton" id="btnOK">
            <property name="width_request">100</property>
            <property name="height_request">40</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <property name="label" translatable="yes">OK</property>
            <property name="response_id">0</property>
            <signal name="clicked" handler="on_btnOK_clicked"/>
          </widget>
          <packing>
            <property name="x">211</property>
            <property name="y">313</property>
          </packing>
        </child>
```

Initiate Window

This is the start of the program. It creates a new instance of a class, in this case named SetupGTKInterface(). After that, the `gtk.main()` function is called to display the window and run the program. This runs until the `gtk.main_quit()` function is called.

```
# This initiates window setup and calls the 'main' GTK
function when this script is executed.
if __name__ == '__main__':

    # fswg = file stats window gtk handle
    fswg = SetupGTKInterface()
    gtk.main()
```

Loading the Glade File

The first thing the class does as part of its initialization is to read in the Glade file using the `gtk.glade.XML()` function and connect it to the XML object instance variable, wTree. wTree is short for widget tree.

```
class SetupGTKInterface:
    """This provides the GTK interface."""

    def __init__(self):

    # set and load the Glade file
    self.gladefile = "FileStats3.glade"
    self.wTree = gtk.glade.XML(self.gladefile)
```

PyGTK Events

Events are actions the user takes on the GUI such as clicking a button, selecting a menu choice, or exiting the program. The events generate signals which are used by the program via signal handlers.

You can use a dictionary with the `signal_autoconnect` function to connect the event signal handlers from the GUI to their corresponding functions.

For example, when the user clicks the Cancel button, the GUI sends the "clicked" signal, handled by the handler `on_btnCancel_clicked`, which is connected to the `btnCancel_clicked` function.

```
# create events dictionary and connect it
EventDict = { # signal handler : function
            "on_btnOK_clicked" : self.btnOK_clicked,
            "on_btnCancel_clicked" : self.btnCancel_clicked,
            "on_FileStatsWindow_destroy" : gtk.main_quit,
            "on_mnuQuit_activate" : self.mnuQuit_clicked,
            "on_mnuAbout_activate" : self.mnuAbout_clicked }
self.wTree.signal_autoconnect(eventDict)
```

Import Widgets

Widgets are the individual items on a GUI.

Here we are using the `get_widget()` function to set up access to (import) some of the widgets we want to access: the results window and the path the user entered. The function is given the name we used in the Glade file.

```
# set up the results window - import the widget
self.textviewResults = self.wTree.get_widget("textviewResults")
# set the text buffer
self.textbufferResults = self.textviewResults.get_buffer()

# import the text entry widget
self.entryPath = self.wTree.get_widget("entryPath")
```

Cancel Button Function

This is the function that is called when the user clicks the Cancel button on the GUI. The function is linked via the dictionary referenced in an earlier slide.

It calls the quit function which exits the program.

```
def btnCancel_clicked(self, widget):
    """
    This function is called when the user clicks the Cancel
button.
    It exits the program.
    """

    gtk.main_quit()
```

OK Button Function

This is the function that is called when the user clicks the OK button on the GUI. It retrieves the text that the user entered in the textbox by using the `get_text()` function and calls the analyze function.

```
def btnOK_clicked(self, widget):
    """
    This function is called when the user clicks the OK button.
    It gets the path the user entered in the Path box and calls
    the analyze function.
    """

    # use global variable defined above
    global userDir

    # get the text from the Path box
    userDir = self.entryPath.get_text()
    analyze()
```

Putting It All Together

Demo

PyGTK Resources

http://www.learningpython.com/2006/05/07/creating-a-gui-using-pygtk-and-glade/

http://www.pygtk.org/docs/pygtk/index.html

That's All Folks

Questions?