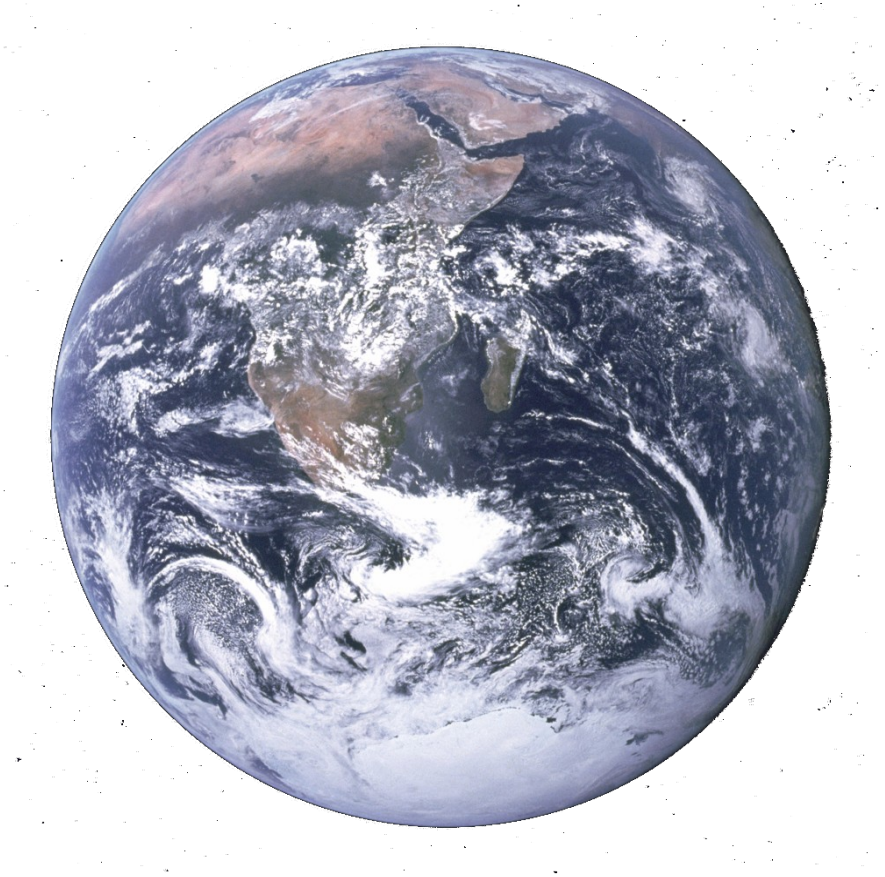
Lars Wirzenius <liw@iki.fi> wrote:

HELLO, WORLD



A brief tutorial on PyGTK programming

```
import pygtk
pygtk.require("2.0")
import gtk

greeting = gtk.Label("hello, world")

w = gtk.Window(gtk.WINDOW_TOPLEVEL)
w.add(greeting)
w.show_all()

gtk.main()
```

Basic concepts

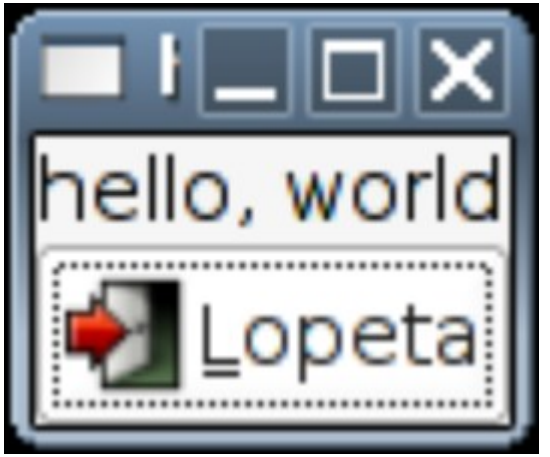
- Puzzle pieces
  - GTK+ = gtk, gdk, glib, atk, pango, ...
  - PyGTK = Python bindings for GTK+
  - GNOME = desktop environment built with GTK+
- Widget = visible object on screen (roughly)
- Signal = message sent by widget
  - "the OK button was clicked"
  - interested parties can register their interest in particular signals for particular widgets

PyGTK programming for the impatient

1. Create widgets
2. Connect signals to callback functions
3. Run

```
import pygtk
pygtk.require("2.0")
import gtk

def quit(*args):
    gtk.main_quit()

greeting = gtk.Label("hello, world")

w = gtk.Window(gtk.WINDOW_TOPLEVEL)
w.add(greeting)
w.connect("delete-event", quit)
w.show_all()

gtk.main()
```

```
import pygtk
pygtk.require("2.0")
import gtk

def quit(*args):
    gtk.main_quit()

greeting = gtk.Label("hello, world")

button = gtk.Button(stock=gtk.STOCK_QUIT)
button.connect("clicked", quit)

vbox = gtk.VBox()
vbox.pack_start(greeting)
vbox.pack_start(button)

w = gtk.Window(gtk.WINDOW_TOPLEVEL)
w.add(vbox)
w.connect("delete-event", quit)
w.show_all()

gtk.main()
```



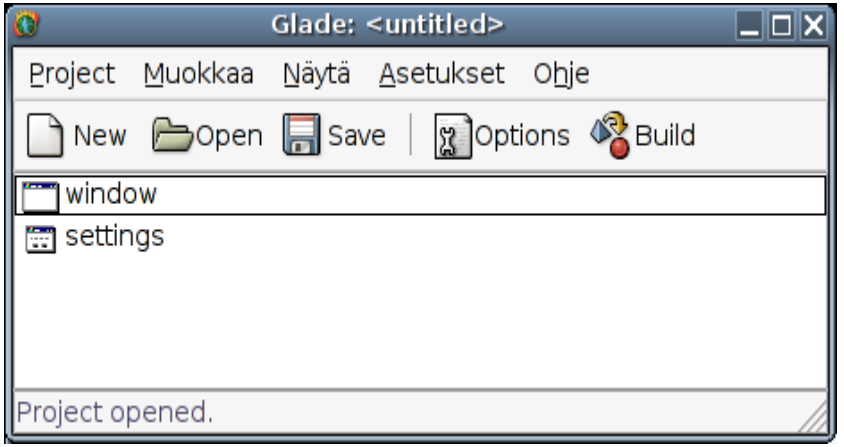

gtk.Window
gtk.Label
gtk.Button
gtk.VBox
hello, world
Lopeta

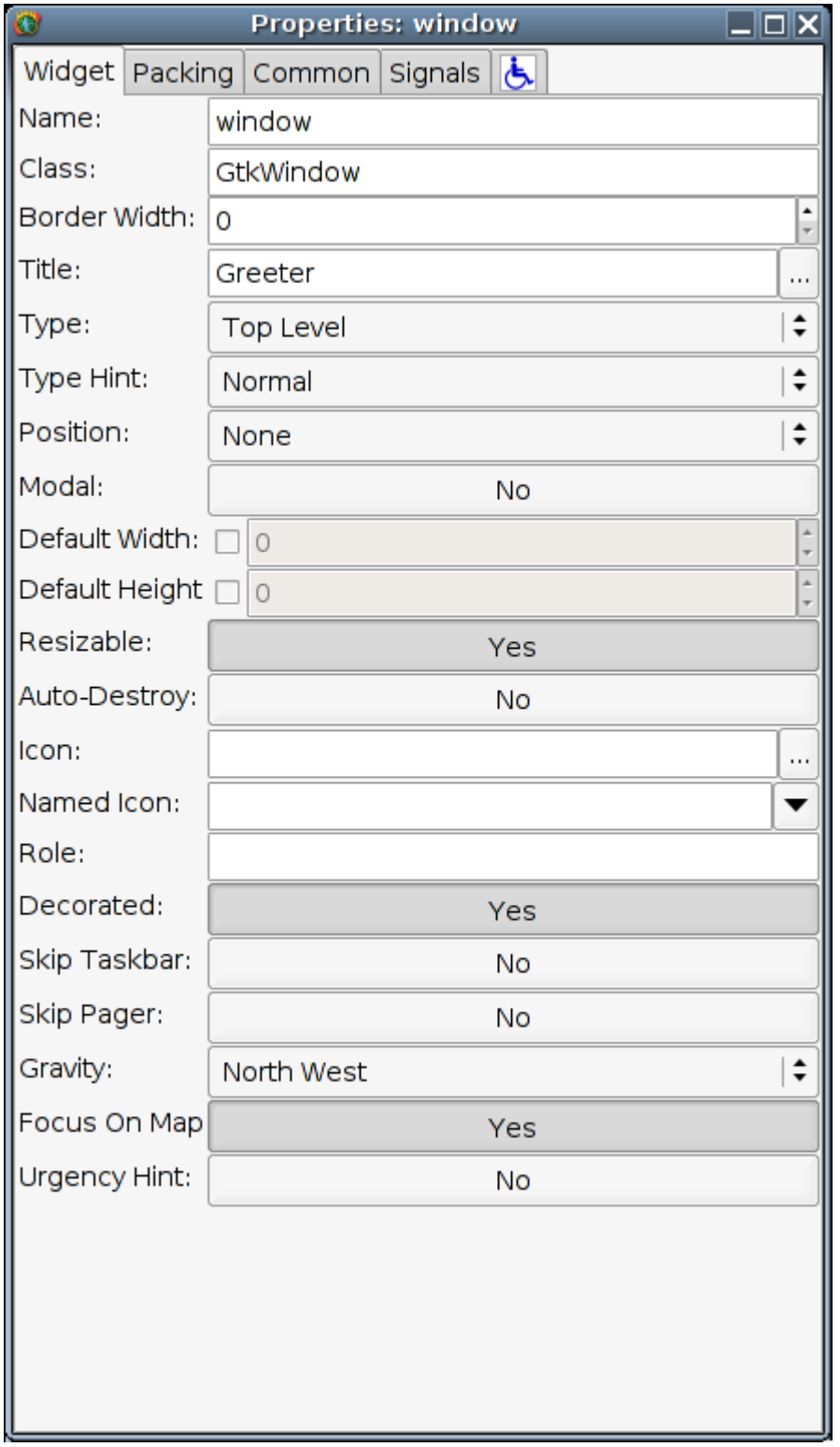
```
def quit(*args):
    gtk.main_quit()

quit_item = gtk.MenuItem("Quit")
quit_item.connect("activate", quit)

file_menu = gtk.Menu()
file_menu.append(quit_item)

file_item = gtk.MenuItem("File")
file_item.set_submenu(file_menu)

main_menu = gtk.MenuBar()
main_menu.append(file_item)

greeting = gtk.Label("hello, world")

vbox = gtk.VBox()
vbox.pack_start(main_menu)
vbox.pack_start(greeting)

w = gtk.Window(gtk.WINDOW_TOPLEVEL)
w.add(vbox)
w.connect("delete-event", quit)
w.show_all()

gtk.main()
```

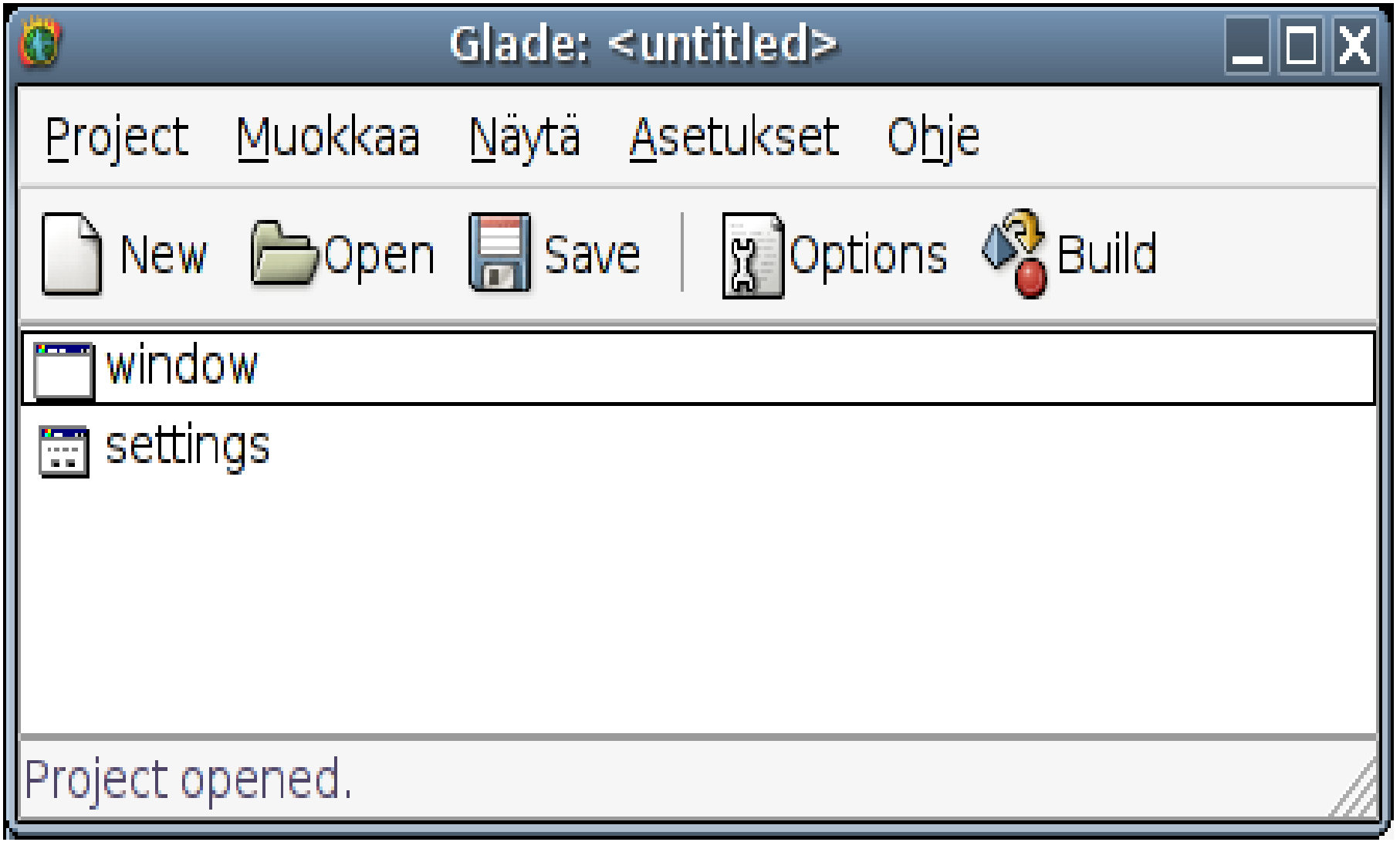
```
def settings(*args):
    dialog = gtk.Dialog(title="Settings",
                        parent=w,
                        flags=gtk.DIALOG_MODAL |
                                  gtk.DIALOG_DESTROY_WITH_PARENT,
                        buttons=(gtk.STOCK_OK, gtk.RESPONSE_ACCEPT))
    entry = gtk.Entry()
    entry.connect("activate",
                  lambda *args: dialog.response(gtk.RESPONSE_ACCEPT))
    dialog.vbox.pack_start(gtk.Label("New greeting:"))
    dialog.vbox.pack_start(entry)
    dialog.show_all()
    response = dialog.run()
    dialog.hide()
    if response == gtk.RESPONSE_ACCEPT:
        greeting.set_text(entry.get_text())

settings_item = gtk.MenuItem("Settings...")
settings_item.connect("activate", settings)
```

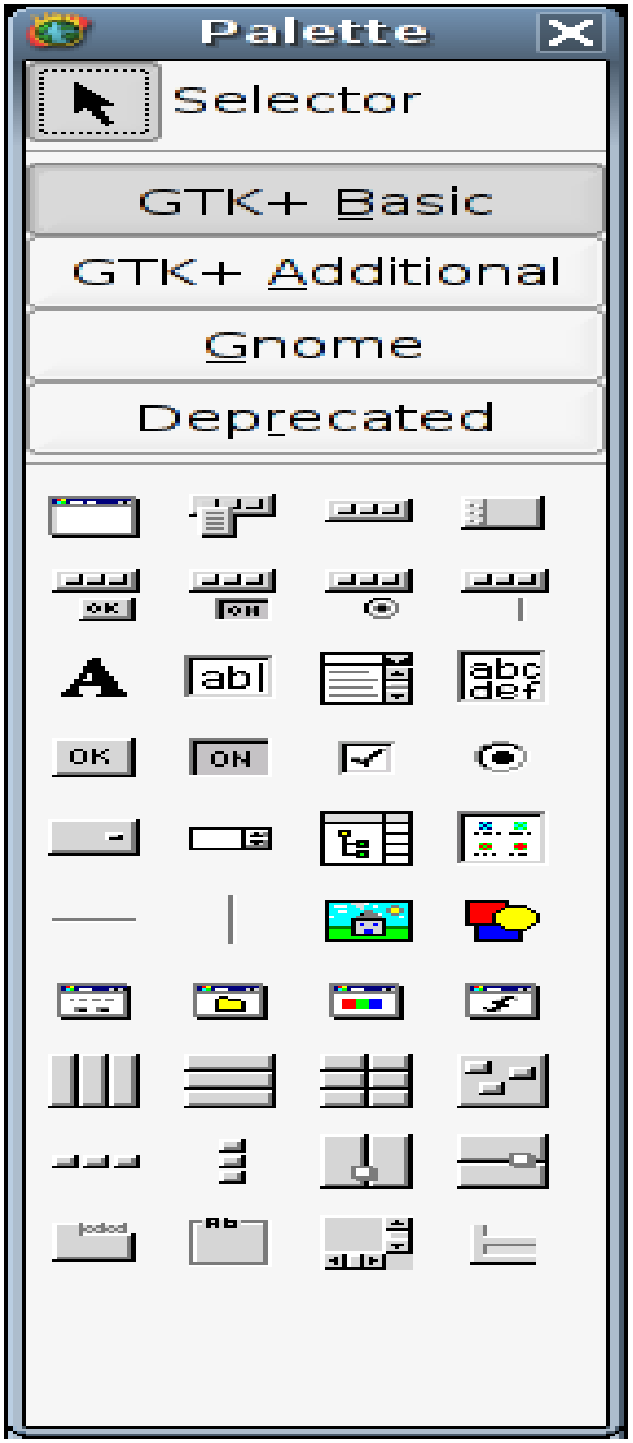
GLADE

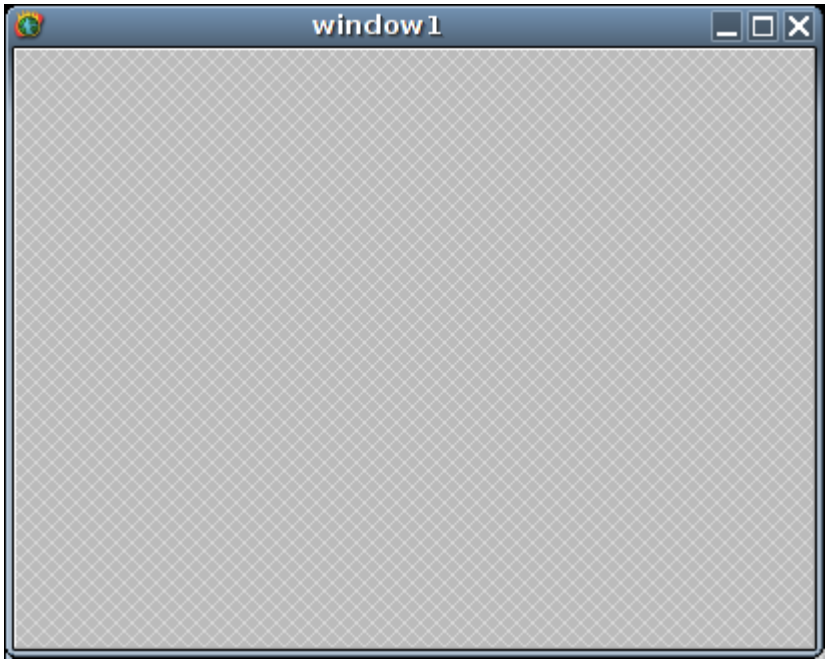
- GUI tool to combine widgets on-screen

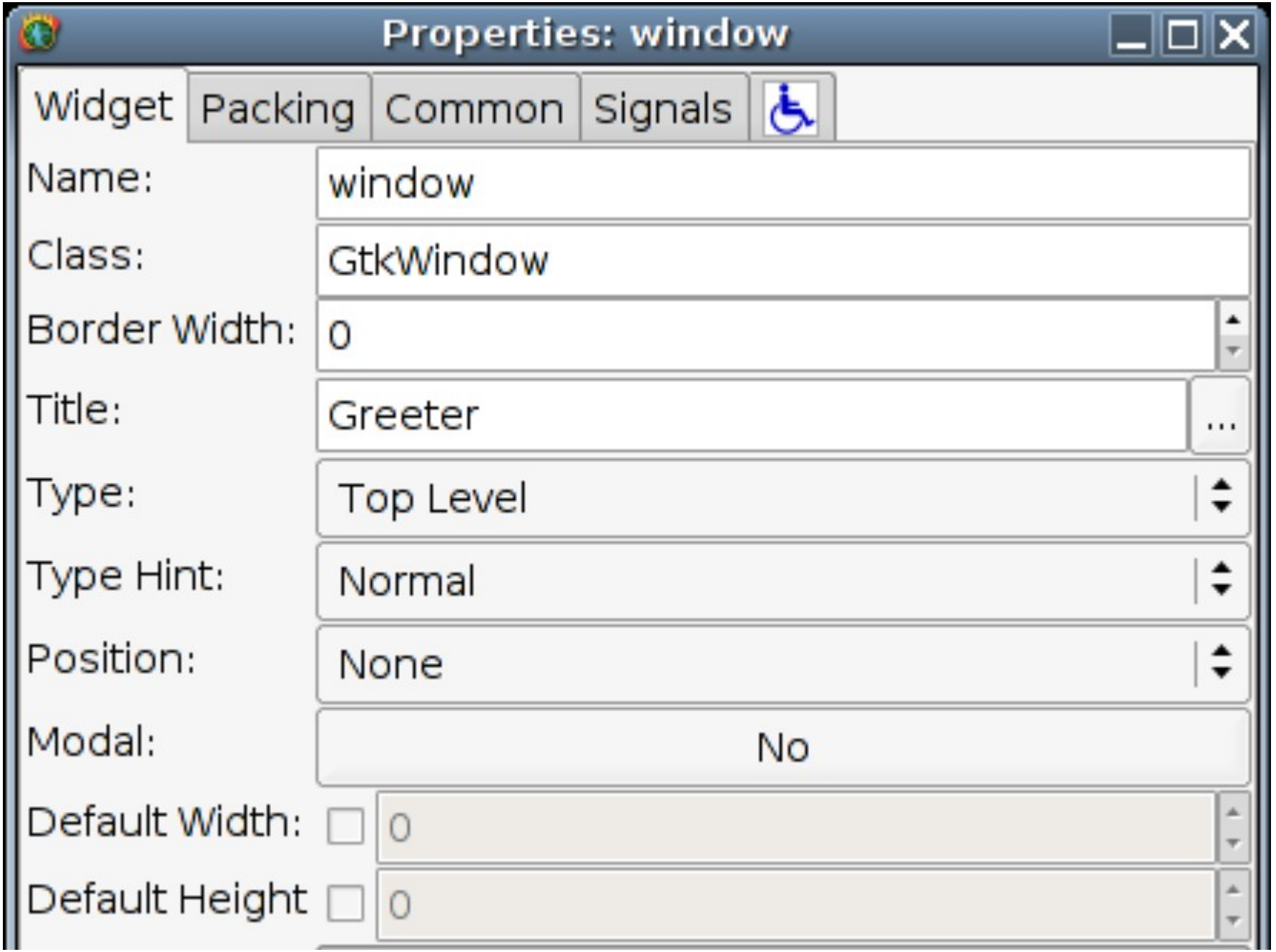


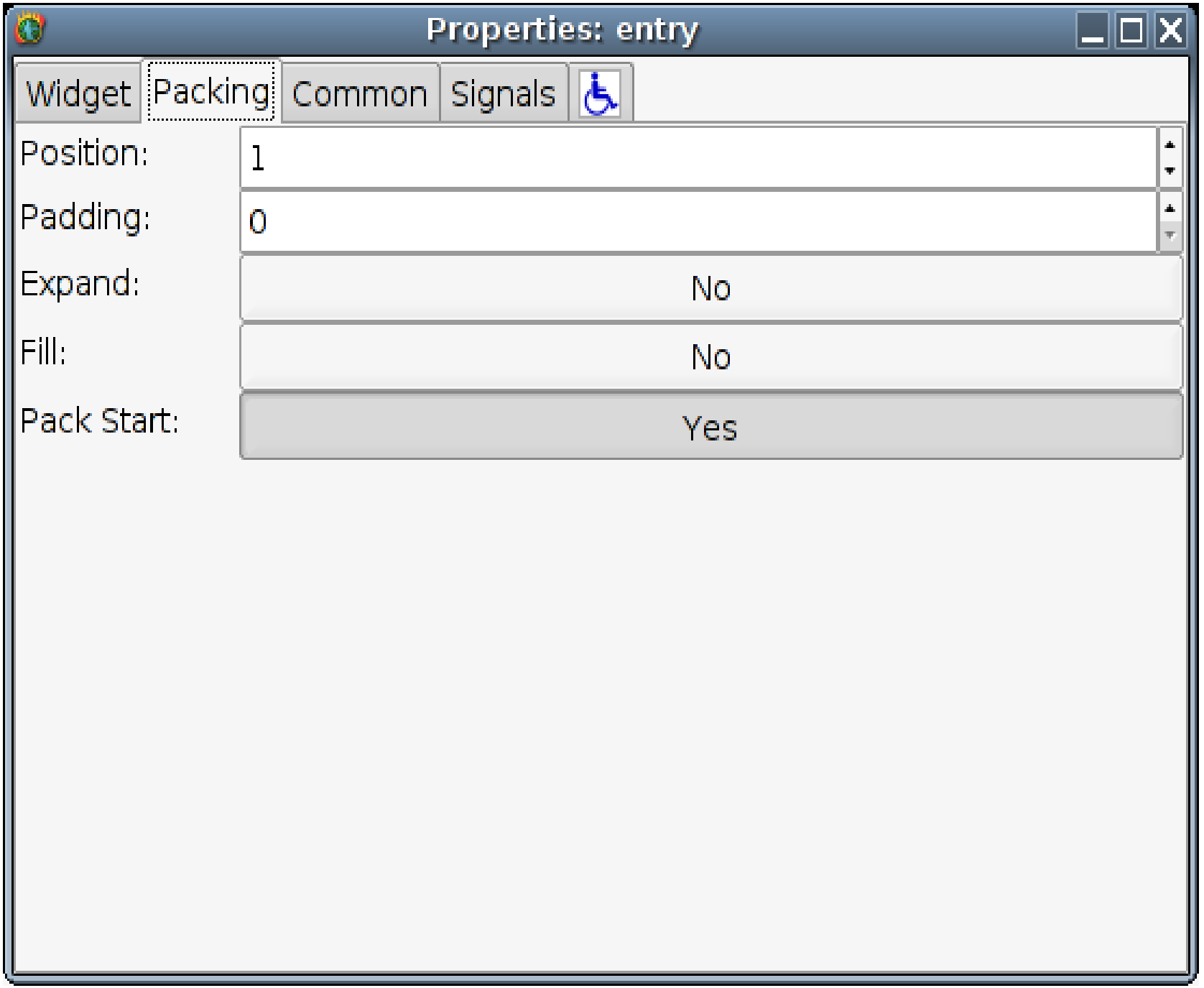


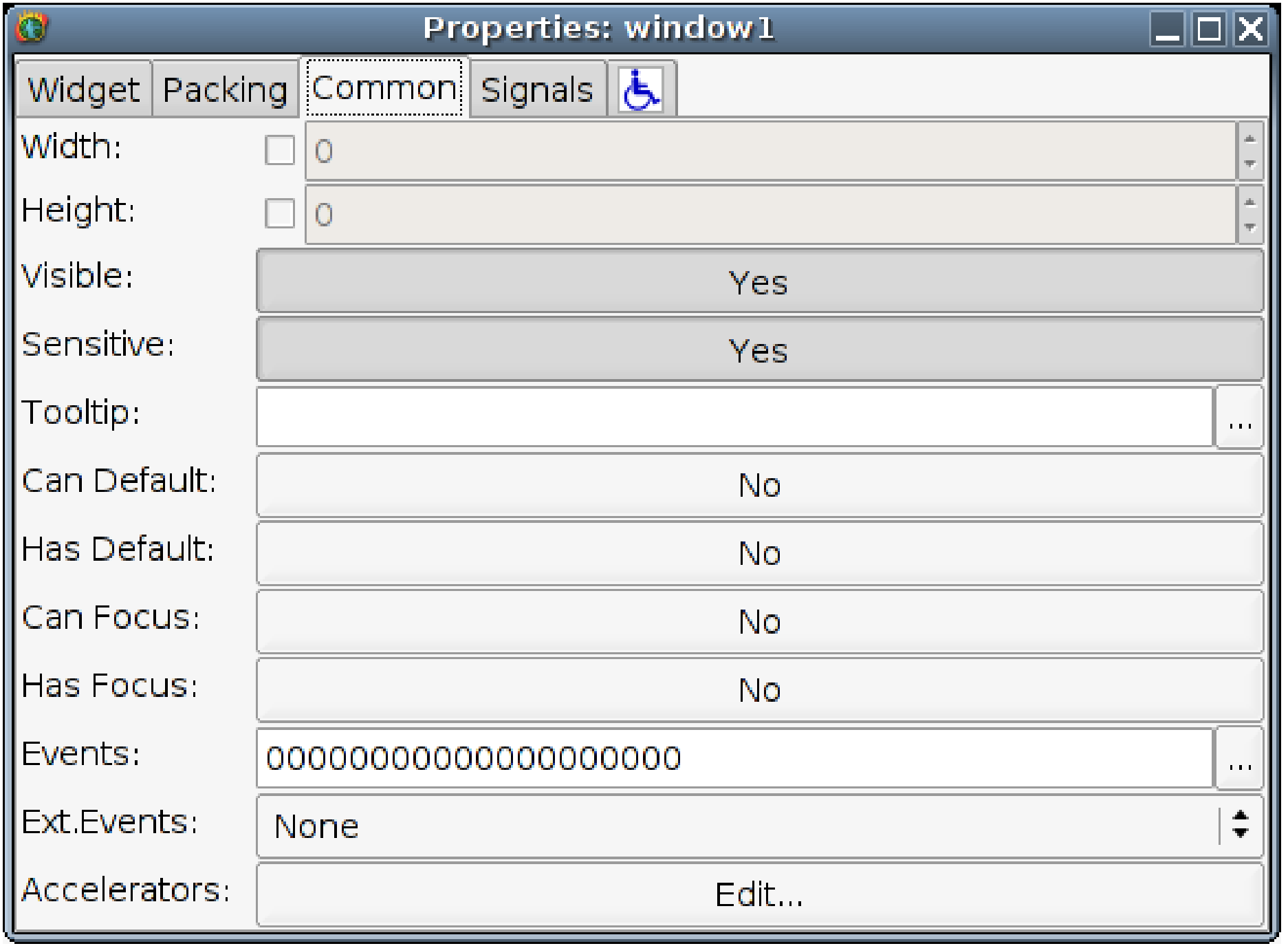

Glade: <untitled>
Project
Muokkaa
Näytä
Asetukset
Ohje
New
Open
Save
Options
Build
window
settings
Project opened.

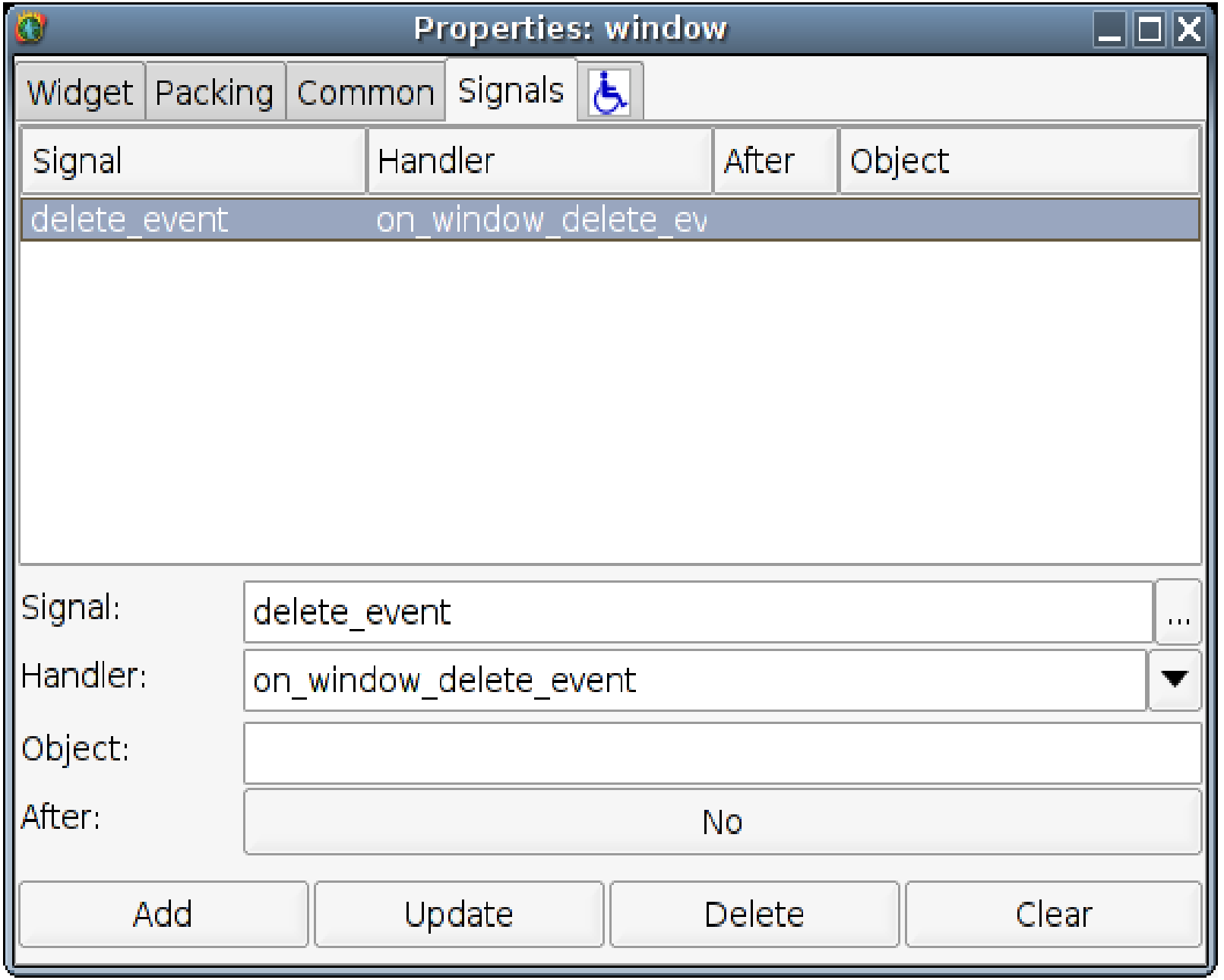

Palette
Selector
GTK+ Basic
GTK+ Additional
Gnome
Deprecated


window1


Properties: window
Widget
Packing
Common
Signals
Name:
window
Class:
GtkWindow
Border Width:
0
Title:
Greeter
...
Type:
Top Level
Type Hint:
Normal
Position:
None
Modal:
No
Default Width:
0
Default Height
0


Properties: entry
Widget
Packing
Common
Signals
Position:
1
Padding:
0
Expand:
No
Fill:
No
Pack Start:
Yes


Properties: window1
Widget
Packing
Common
Signals
Width:
0
Height:
0
Visible:
Yes
Sensitive:
Yes
Tooltip:
...
Can Default:
No
Has Default:
No
Can Focus:
No
Has Focus:
No
Events:
00000000000000000000
...
Ext.Events:
None
Accelerators:
Edit...


Properties: window
Widget
Packing
Common
Signals
Signal
Handler
After
Object
delete_event
on_window_delete_ev
Signal:
delete_event
...
Handler:
on_window_delete_event
Object:
After:
No
Add
Update
Delete
Clear

libglade = good

- Generated code will need to be re-generated
  - Are you **REALLY QUITE SURE** you did that?
  - And are you sure you don't need to edit the generated code?
- Load widget descriptions from file and create them
  - Glade creates a .glade file anyway, for itself
  - libglade (gtk.glade in PyGTK) uses the .glade file
  - Creates widgets at run time, no recompilation needed

```
import gtk.glade
import gnome

def settings(*args):
    dialog.show_all()
    response = dialog.run()
    dialog.hide()
    if response == gtk.RESPONSE_ACCEPT:
        greeting.set_text(entry.get_text())

gnome.init("hello6", "1.0")
xml = gtk.glade.XML("hello6.glade")
dict = {
    "on_window_delete_event": quit,
    "on_quit_activate": quit,
    "on_settings_activate": settings,
}
xml.signal_autoconnect(dict)
w = xml.get_widget("window")

greeting = xml.get_widget("greeting")
dialog = xml.get_widget("settings")
dialog.set_transient_for(w)
entry = xml.get_widget("entry")
entry.connect("activate",
              lambda *args: dialog.response(gtk.RESPONSE_ACCEPT))
gtk.main()
```

READTHEM

- http://developer.gnome.org/doc/API/
  - C API, but mostly easily translatable to Python
  - GtkWindow becomes gtk.Window, etc
- http://www.pygtk.org/
  - http://www.pygtk.org/pygtk2reference/index.html
  - Reference manual for the Python bindings, semi-automatically translated from the reference manual for the C API
- Matthis Warkus, "*The Official GNOME 2 Developer's Guide*"