

PyGTK

Creación de Aplicaciones gráficas sencillas

Pablo Barrera González

<barrera@gsync.escet.urjc.es>

Grupo de Usuarios de Linux (GUL)

22 de Marzo de 2004



Aplicaciones gráficas

- Cada día son más necesarias
- Es una parte muy importante del resultado final
- Motivos frikis
- ...
- Pero nunca tienes tiempo para aprender

Situación actual

- Hay muchas posibilidades
 - GTK+
 - Qt
 - TK
 - WxWindows
 - Motif
 - Xforms
 - Xlib

¿Por qué PyGTK?

- Es totalmente libre
- Es fácil de aprender
- Suficiente documentación
- Python es introspectivo
- Python es de alto nivel
- Puede usar libglade



Herramientas

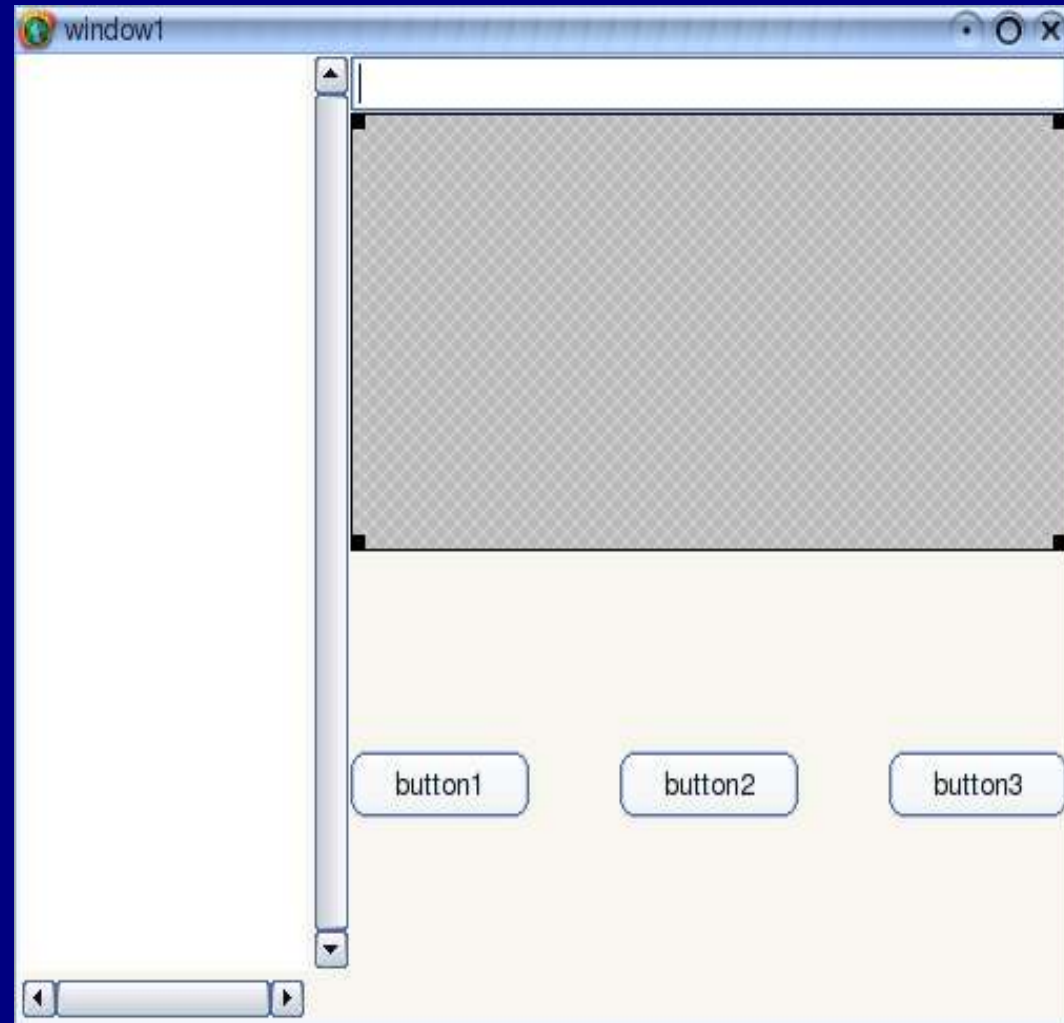
- Emacs (o lo que sea)
- Glade y libglade
- Python
- Devhelp (con libro de GTK+)

GTK+

- Gimp Toolkit
- Usada por Gnome
- Widget
- Pseudo Orientada a objetos
- Pensada para C
- Muchos recubrimientos

Widgets

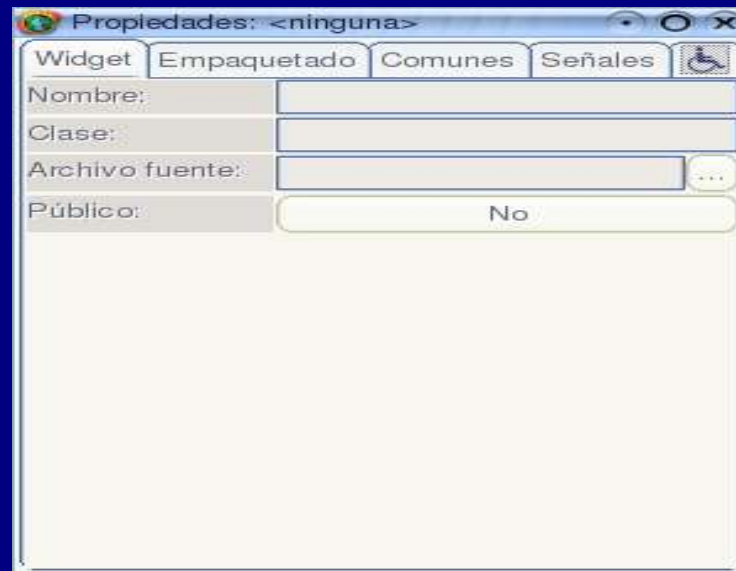
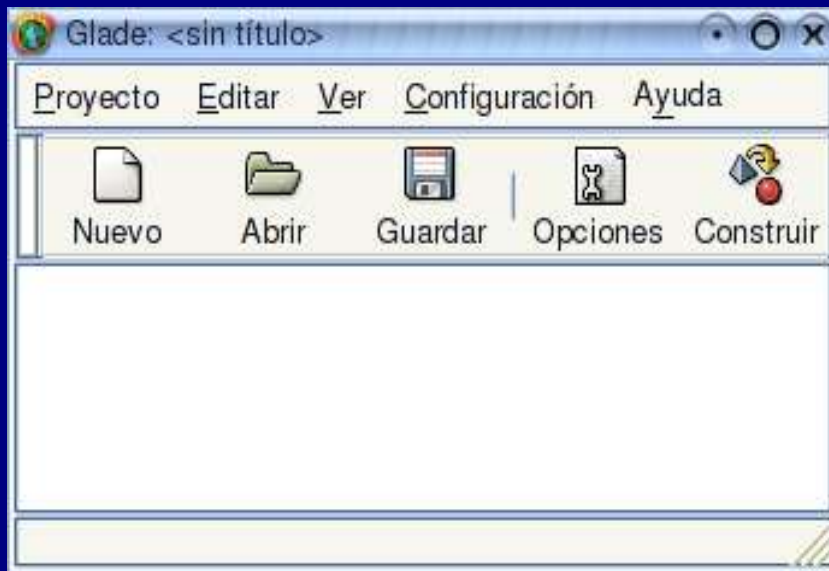
- Jerarquía
- Orientados a Objetos
- Ventana
 - Contenedores
 - Cualquier Widget





Glade

- Entorno gráfico
- Ventanas, Widgets y Señales





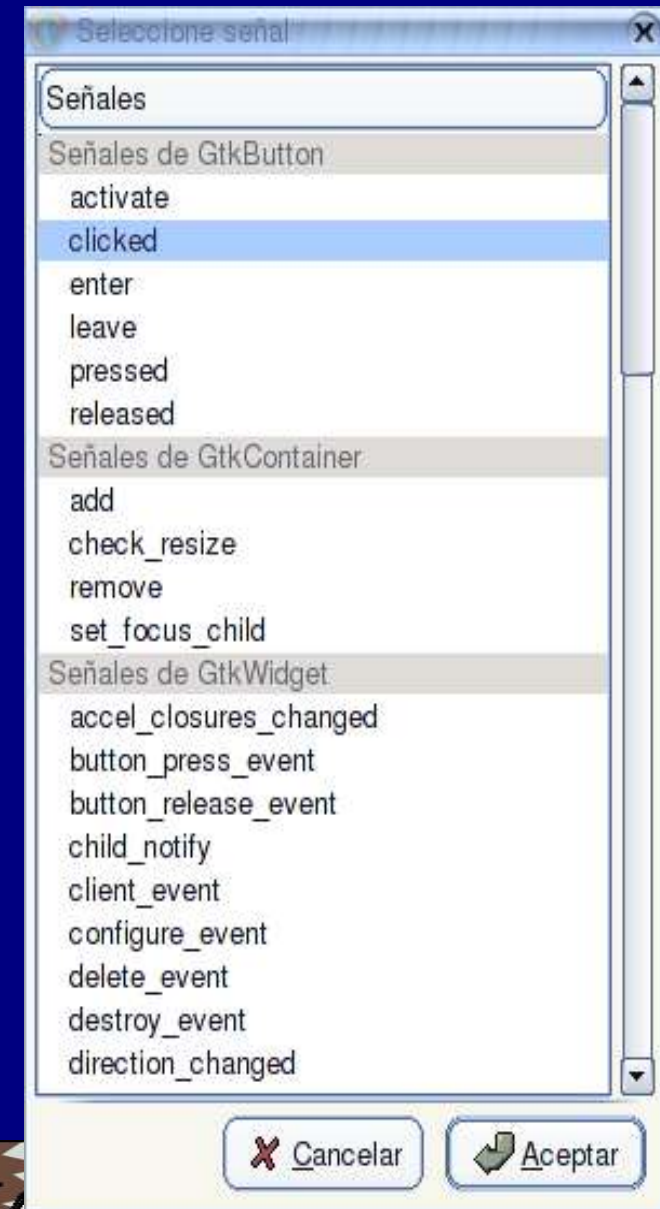
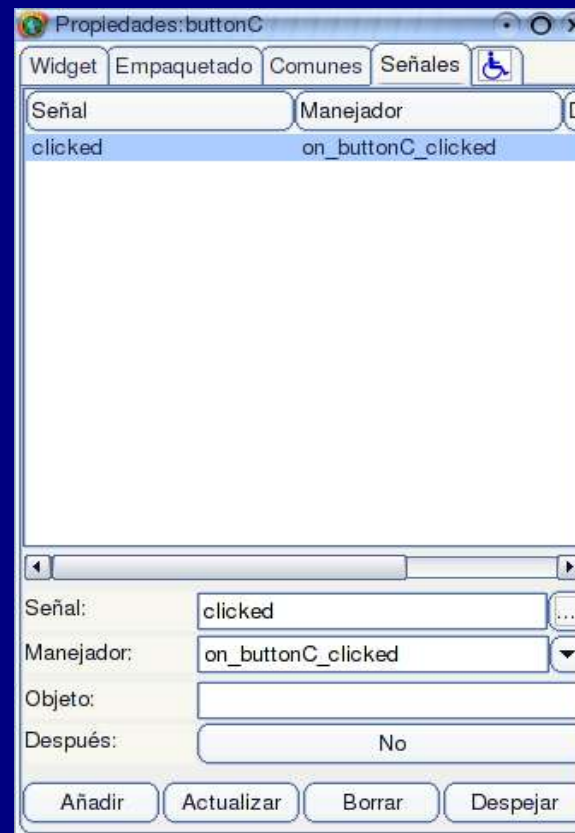
Comenzado una interfaz

- Colocar una o varias ventanas
- Colocar sus contenedores y elementos
- Cambiar los nombres
- Ajustar las propiedades (margen, tamaño,etc)
- Añadir las señales y los manejadores
- Añadir señal de destrucción



Señales y manejadores

- Funcionamiento por eventos
- Los manejadores tratan los eventos



Tirando código

- Usamos módulos: pygtk, gtk y gtk.glade

```
#!/usr/bin/python  
  
import pygtk # for testing GTK version number  
  
pygtk.require ('2.0')  
  
import gtk  
  
import gtk.glade
```



libGlade

- Permite cargar el XML creado por Glade
- Permite conectar las señales a sus manejadores
- Cambiando el archivo Glade, cambia la interfaz

```
self.xml = gtk.glade.XML('proyecto1.glade')
```

Una clase para la interfaz gráfica

Class GUI:

```
def __init__(self, oc):  
    # Load glade file  
  
    self.xml = gtk.glade.XML('proyecto1.glade')  
  
    # Connect handlers of gui to this class  
  
    self.xml.signal_autoconnect(self)  
  
    # To access a widget -> self.xml.get_widget("name")  
  
    # More information: help(gtk.glade.XML)
```

Manejadores

- Métodos para atender a las señales
- Dos tipos:
 - 1 parámetro de entrada (widget)
 - 2 parámetros de entrada (widget, evento)

```
def on_window1_destroy(self, widget):  
    gtk.main_quit()
```

Arrancando la aplicación

- Cargamos el archivo Glade
- Conectamos los manejadores
- Comenzamos el bucle de GTK
- Para salir: `gtk.main_quit()`

```
if __name__ == "__main__":  
    gui = GUI()  
    gtk.main()
```

Acceder a un widget

- Usando gtk.glade y el archivo xml

```
self.entryC = self.xml.get_widget("nombre")
```
- Tomándolo de los argumentos de entrada de los manejadores

Trucos

- Usar la ayuda de Python
- La introspección con los widgets
- Buscar ejemplos de GTK
- Usar Devhelp o el API de GTK+

Funcionamiento del bucle

- `gtk.main()` atiende a la lista de eventos
- `gtk_main_iteration()` ejecuta una única iteración
- Todo en un único hilo
- El programa solo funciona en modo reactivo

Funciones de tiempo ocioso

- Cuando el `gtk.main()` tiene tiempo ocioso
`gtk.idle_add(función)`
- Sigue siendo un único hilo
- Debe devolver el control suficientemente rápido
- Debe devolver `gtk.True`

Hilos de ejecución

- En general es mejor usar funciones idle
- Aún así es posible usar hilos

```
import thread

gtk.threads_init()

thread.start_new_thread(gui.run, ())

gtk.threads_enter()

gtk.main()

gtk.threads_leave()
```

Otras funcionalidades

- Archivos po
- Teclas rápidas
- Funciona en Microsoft Windows

¿Preguntas?

