Paradigmas de Programación. Grado en Ingeniería Informática

Primeros pasos con Glade y PyGTK

Curso 2010–11

1. Introducción

El objetivo de este documento es servir de guía para la elaboración de una primera aplicación Python dotada de una interfaz PyGTK diseñada mediante Glade. La descripción detallada de las posibilidades de PyGTK o Glade, los criterios de diseño de una aplicación de estas características o el estudio de la elaboración de interfaces gráficas de usuario quedan fuera del ámbito de este documento.

Los ejemplos a que se refiere este documento han sido realizados utilizando PyGTK (v.2.22.0) Glade (v 3.6.7) y Python (v 2.6.6) Es posible que el código recogido en estos ejemplos no sea compatible con versiones diferentes de este software.

Este documento está planteado como una «guia de lectura» del código de los ejemplos que lo acompañan. Es recomendable jugar con los ejemplos, modificarlos y ver cómo funcionan antes de intentar elaborar nuestra propia aplicación con interfaz GTK. Tanto este documento como los ficheros Python y glade que lo acompañan pueden ser descargados de `http://www.infor.uva.es/~cvaca/asigs/paradigmas.html`.

2. Hola mundo (versión 0)

2.1. La interfaz

Lo primero que necesitamos es diseñar una pequeña ventana con un botón y un mensaje. Para ello utilizaremos Glade[1].

Al arrancar la aplicación, se nos pide que seleccionemos las preferencias para el proyecto que vamos a elaborar. Como formato del archivo de proyecto debemos seleccionar `GtkBuilder`, pues de otro modo el código que proporcionamos en estos ejemplos no será utilizable. En cuanto al resto de las opciones podemos explorarlas, pero en principio no es necesario modificar nada.

Una vez elegidas las preferencias para éste proyecto, Glade nos permite «dibujar» la interfaz gráfica de nuestra aplicación. Para este primer proyecto tan solo necesitaremos los siguientes elementos:

- *Una ventana principal*. Contendrá el resto de los elementos gráficos de la interfaz. Podemos explorar sus propiedades para fijar el tamaño, el nombre de la ventana y algunos otros elementos de su aspecto y comportamiento como, por ejemplo, si la ventana va a ser visible o no.
- *Una caja vertical*. En GTK como en muchos otros «frameworks» de interfaz de usuario, situamos los elementos de la interfaz (los «widgets») de forma relativa a otros elementos o al widget que los contiene. Así, en este caso, si vamos a dibujar una ventana con una etiqueta y un botón, necesitamos una caja vertical con dos espacios para albergar esos widgets. En sus diversos menús de propiedades fijaremos el aspecto que va a tener la caja, el número de widgets que puede albergar y la información sobre el tamaño que se reserva para ellos, entre otras cosas.

  Habitualmente diremos que la caja vertical es «hijo» de la ventana principal, mientras que los siguientes elementos son hijos de la caja vertical. Para colocar un widget como hijo de otro, basta seleccionarlo de la caja de herramientas y colocarlo sobre el widget que deseemos como padre.
- *Una etiqueta*. La etiqueta, «label», nos permite mostrar un mensaje al usuario. Podemos cambiar elementos como su tamaño, aspecto o color de la letra

[1] Para arrancar glade en jair debemos utilizar el comando `glade-3`

- *Un botón*. También es conveniente jugar con las propiedades de los botones para explorar sus posibilidades de representación

Es normal que el aspecto de la ventana no sea demasiado elegante al primer intento. Es bueno tomarse algún tiempo para revisar los diversos controles de que disponen los widgets hasta dar con un aspecto agradable.

En este sentido, aunque los widgets permiten ciertos elementos de configuración en términos absolutos, es preferible recurrir siempre a configuraciones relativas. Hay que recordar que las ventanas serán utilizadas en pantallas de diversas resoluciones y tamaños. Entre estos controles relativos resaltaremos dos:

- Expandir: Indica si el hijo debe recibir espacio extra cuando el padre crece.
- Rellenar: Indica si el espacio extra dado al hijo debe ser rellenado por éste o puede ser utilizado como separación.

2.2. El código asociado

Una vez que hemos dado con un aspecto aceptable para nuestra ventana, debemos escribir un código Python que nos muestre esos elementos gráficos. El ejemplo que se proporciona en la versión 0 de los ejemplos es muy fácil de leer, pero resaltaremos algunos elementos.

- Es necesario importar el paquete GTK para utilizar la interfaz Glade que hemos construido.
- Para acceder a esa interfaz, es necesario crear un objeto de tipo `Builder` y asociarle los elementos gráficos que hemos diseñado en Glade.
- Todos estos elementos gráficos se instancian como objetos GTK. Podemos acceder a esos elementos mediante `gtk_object()` y utilizando como referencia el nombre que les hemos dado en Glade. No es necesario acceder a todos esos objetos, pero es imprescindible disponer de una referencia a ellos si deseamos cambiar su aspecto dentro del programa
- GTK es un Framework. Esto quiere decir que lo que finalmente se ejecuta no es una de nuestras funciones, sino una función predefinida en GTK, en concreto `gtk_main()`. Esta función se encarga de la ejecución del bucle de control de eventos. Todo lo que hace una aplicación GTK es esperar a que se produzca un evento (pulsación de un botón, una tecla, destrucción de una ventana,...) y obrar en consecuencia.

De momento la aplicación que hemos construido es absolutamente inútil. De hecho, cuando cerramos la ventana, el programa Python sigue en ejecución (para detenerlo tenemos que utilizar control-C). En la próxima sección estudiaremos cómo podemos añadir un poco de funcionalidad a nuestro programa.

3. Añadiendo la gestión de eventos (versión 1)

3.1. Pulsación de un botón

Lo primero que vamos a hacer es añadir una reacción en el programa para la pulsación del botón. Para ello primero debemos volver a editar nuestra interfaz mediante Glade. Si seleccionamos el botón, entre sus propiedades encontraremos una pestaña dedicada a las señales. De entre todas estas señales elegimos `clicked` y le asociamos un manipulador, `on_button1_clicked`. En realidad el nombre del manipulador no es importante, sólo debemos recordarlo para utilizar el mismo nombre en nuestro código.

Ahora, una vez guardada nuestra nueva interfaz debemos realizar dos tareas en nuestro código Python:

- Conectar las señales de la interfaz a un objeto: En nuestro caso basta con utilizar una única llamada `self.glade.connect_signals(self)` para conectar todas las señales con las correspondientes funciones de la clase `holamundo`
- Escribir un método en esta clase encargado de manejar el «click» sobre el botón. En nuestro caso el método `on_button1_clicked`, si es que hemos utilizado ese nombre en Glade.

3.2. El cierre de la aplicación

El nuevo programa reacciona cuando pulsamos el botón (simplemente enviando un mensaje a la consola, pero reacciona), sin embargo cuando cerramos la ventana el programa sigue en ejecución. Para detenerlo necesitamos manejar el evento de borrado de la ventana, `delete_event`

En principio la mecánica es la misma que en el caso anterior, pero ahora la definición del manejador tiene tres argumentos. El motivo es que el borrado de la ventana no es una *señal*, sino un *evento*. Por este motivo el manejador recibe una información extra sobre lo que ha ocurrido en la interfaz, el propio evento.

La información extra proporcionada por el evento no es importante en este ejemplo, pero puede ser crucial cuando debemos reaccionar ante la pulsación de una tecla, por ejemplo.

Si lo que queremos es que el programa termine al cuando el usuario intenta borrar la ventana principal basta con utilizar `gtk.main.quit()` como hacemos en el ejemplo.

En determinadas situaciones, lo que deseamos es impedir la destrucción de la ventana, para pedir una confirmación al usuario, por ejemplo. Para ello es imprescindible que devolvamos un valor `True` como retorno de esta función. En caso contrario el evento de borrado de la ventana no será totalmente procesado y se transformará en una señal `destroy` que eliminará la ventana, pero no cerrará la aplicación.

Como se puede comprobar en esta segunda versión del ejemplo, tras la salida del bucle de comprobación de eventos (`gtk.main()`) se continúa con la ejecución del código Python, que en este caso se limita a escribir un mensaje por pantalla y terminar.

Con lo que hemos visto hasta el momento podemos ya elaborar algunas aplicaciones básicas. En la siguiente sección comprobaremos que podemos modificar el aspecto de los widgets que utilicemos y agruparlos en otros bloques, com las tablas.

4. Elementos gráficos más complejos (versión 2)

Para la elaboración de cualquier programa necesitaremos elementos más complejos que un simple botón y una etiqueta. En esta versión vamos a añadir a nuestro programa un conjunto de 9 botones dotados de una funcionalidad simple. Además veremos cómo podemos modificar el aspecto de nuestros botones desde el código Python.

4.1. El aspecto de los widgets

Ya hemos comentado antes que desde Glade podemos cambiar el aspecto de los widtgets que utilicemos. En ocasione, sin embargo, necesitamos que ese aspecto cambie en tiempo de ejecución. Para ello podemos utilizar mensajes a los objetos desde el código Python. En el ejemplo, cambiamos el aspecto de los botones cuando son pulsados, eliminando su etiqueta y añadiendo una imágen.

La información completa sobre el tipo de widgets utilizables y sus características está disponible en el manual de referencia de GTK[2]

Al trabajar con los widgets hay que tener cuidado en distinguir entre objetos y referencias. En nuestro código, por ejemplo, estamos fabricando una imagen diferente para cada botón a partir del mismo fichero. Si fabricamos una sola imágen para todos los botones el resultado será bastante diferente.

4.2. Una caja con botones

Además de las cajas horizontales y verticales, disponemos de un elemento que organiza los widgets en una cuadrícula, GtkTable. Las tabla se se utilizan igual que las cajas horizontales o verticales, pero la aparición de muchos botones nos provoca un nuevo problema: ¿Cómo vamos a gestionar los eventos de todos estos botones?

Por supuesto podemos aplicar la misma técnica que en las versiones anteriores, pero si lo hacemos en nuestro ejemplo, necesitaríamos definir 9 funciones diferentes para manejar esos botones.

[2] http://library.gnome.org/devel/pygtk/stable/

En su lugar, cuando el manejo de los botones sea similar, podemos definir una única función que maneje los nueve botones, en nuestro ejemplo `on_pulsador_clicked`.

Hay que tener en cuenta que el segundo argumento de esta función es precisamente el widget (en nuestro caso el botón) que ha disparado el evento, de modo que podemos determinar en el código Python cuál es el botón que se pulsó y actuar en consecuencia.

De todas formas, no siempre es recomendable dibujar todos los widgets de la interfaz desde el editor glade. En la siguiente sección comprobaremos un método alternativo para añadir widgets.

5. Añadir elementos gráficos desde Python (versión 3)

En algunos casos no resulta práctico definir todos los widgets de nuestra interfaz desde Glade. Imaginemos, por ejemplo que tenemos que añadir muchos botones, o incluso un número que varíe en tiempo de ejecución. En este caso lo único razonable es construir estos elementos gráficos desde el código Python.

Para elaborar la versión 3 de nuestro ejemplo hemos modificado la versión anterior de la interfaz Glade eliminando los nueve botones de la caja, pero manteniendo la tabla de botones y su tamaño 3x3[3].

Ahora en el código Python podemos acceder a la tabla de botones con la misma técnica que usamos en la versión anterior, y añadirle botones construídos a mano. En este código hay algunos elementos que merece la pena resaltar:

- No utilizamos un atributo para construir el botón. No es necesario, tras construirlo lo almacenamos en la tabla, podemos acceder a él desde la tabla.
- A diferencia de los elementos gráficos construidos desde Glade, estos botones no son visibles por defecto. Podemos optar por hacerlos visibles uno a uno mediante la función `show` o utilizar la función `show_all` sobre la ventana completa. La decisión depende de si deseamos que todos los widgets sean visibles o queremos ocultar parte de la interfaz.
- La función `connect_singals` de Glade sólo conecta las señales de los widgets definidos desde Glade. Para que los botones que hemos construído a mano reaccionen al ratón hay que conectar el evento utilizando la función `connect` con argumentos adecuados.
- En general disponemos de más control sobre el aspecto de los widgets desde el código Python, pero ese control es menos cómodo http://library.gnome.org/devel/pygtk/stable/

Con estos elementos podemos ya elaborar una aplicación muy básica, pero su aspecto dista aún del habitual en las aplicaciones GTK. A lo largo de la próxima sección comprobaremos cómo podemos añadir algunos otros widgets a nuestra interfaz.

6. Otros elementos gráficos (versión 4)

Una de las grandes ventajas de editores como Glade es que nos permite añadir a nuestros programas elementos gráficos habituales en las aplicaciones GTK de un modo muy simple. En esta nueva versión, partiendo de la versión 2, vamos a añadir a nuestro programa una barra de menús y una funcionalidad muy simple de ayuda, la ventana «Acerca de».

6.1. La barra de menú

Basta añadir un hueco a la caja vertical de nuestra aplicación y situar en él un elemento `GtkMenuBar` para que Glade nos muestre una barra de menú con los elementos habituales en las aplicaciones GTK. A partir de ahí sólo debemos movernos por el editor para añadir o eliminar elementos a nuestro gusto.

En este ejemplo sólo vamos a estudiar el botón de ayuda, más en concreto, una de sus entradas, «Acerca de» que suele contener información sobre el programa y sus autores.

[3] Desde el código Python también podemos cambiar el tamaño de esta tabla

La información a que hacemos referencia no aparece en la ventana principal del programa, sino en una «ventana emergente» que toma el foco de la aplicación hasta ser cerrada. Para construir esta venta utilizamos un nuevo elemento de tipo `GtkAboutDialog`. Se trata de un elemento predefinido en que sólo debemos rellenar los campos de información que aparecen en el menú «general» de Glade.

Una vez que la ventana tiene el aspecto deseado, nos falta conseguir que el comportamiento sea el habitual en estas ventanas:

- La ventana aparece sólo cuando pulsamos el botón adecuado de la barra de menú
- La ventana se cierra cuando pulsamos sobre el botón cerrar correspondiente, o cuando la destruimos.
- La ventana aparece tantas veces como lo deseemos

Para conseguir este comportamiento necesitamos manejar unas cuantas señales en nuestro código Python, en concreto:

- Manejamos el evento de pulsación del botón «Acerca de» desde la función `on_ayuda_activado` que se limita a mostrar la ventana de ayuda. La ventana de ayuda ha sido construida por Glade.
- También debemos manejar la pulsación sobre el botón de cierre de esta ventana de ayuda. Para ello utilizamos el manejador `on_dialogo_ayuda_response` que se limita a ocultar la ventana de ayuda mediante la función `hide`. Destruir la ventana no es buena idea, porque en ese caso deberíamos reconstruirla si el usuario vuelve a pulsar el botón
- Por el mismo motivo debemos interceptar la destrucción de la ventana. Para ello utilizamos el manejador `on_dialogo_ayuda_delete_event`. No hay que olvidar devolver `True` como resultado de éste manejador. En caso contrario el evento se transformará en `destroy` y la ventana será eliminada. Curiosamente, en este caso no hace falta ocultar la ventana mediante la función `hide`. La ventana se oculta automáticamente.

7. Y a partir de aquí

Evidentemente, conforme abordemos sistemas más complicados, los widgets que hemos descrito en este ejemplo resultarán claramente insuficientes. Para más información al respecto se puede acudir al manual de referencia de glade (`http://http://library.gnome.org/devel/gladeui/stable/`) y al manual de referencia de PyGTK (`http://www.pygtk.org/docs/pygtk/index.html`)