Aplicaciones GUI

Graphical User Interface

www.profmatiasgarcia.com.ar

Administrador de interfaz gráfica

Cuando se desarrolla en un sistema operativo, existe “algo” que se encarga de generar la forma que se visualiza en pantalla, de atrapar las interacciones del usuario en pantalla y que permite que nuestro código pueda interactuar con el usuario. Todo lo que vemos comúnmente en una interfaz gráfica, es generada y controlada por ese componente: el administrador de interfaz gráfica.

Para desarrollar en MS Windows se emplea de forma automática el gestor gráfico GDI (Graphics Device Interface). Este es el que pinta el formulario, escucha y atrapa los eventos de la interfaz (clics del mouse, manipulación de estados de una ventana, cierre de una aplicación, etc.).

El administrador GDI solo funciona bajo MS Windows, pero que pasa si deseamos construir una aplicación con una interfaz gráfica así de bonita para la plataforma Linux? La respuesta es que debemos buscar un equivalente que nos permita en la plataforma Linux, cumplir las tareas que hace el GDI de MS Windows.

Se han desarrollado muchos administradores de entorno gráfico (Gnome, KDE, XFCE…), todas en C++ pero sobre sale de todas ellas por su rápido crecimiento el Gtk (www.gtk.org). Este administrador permite desarrollar en la plataforma MS Windows o Linux, interfaces gráficas que lucen igual en los dos sistemas.

Qué es GTK?

El proyecto Gtk (www.gtk.org) es un grupo de desarrolladores cuyo objetivo es desarrollar un framework que permita a otros desarrolladores diseñar aplicaciones con interfaces gráficas en diferentes plataformas.

GTK (GIMP Toolkit) es una librería para crear interfaces de usuario gráficas. Esta licenciada usando la licencia LGPL, por lo que puedes desarrollar software abierto, software libre, o incluso software no libre usando GTK sin tener que pagar nada en licencias o derechos.

Se llama el toolkit de GIMP porque originariamente fue escrita para desarrollar el Programa de Manipulación de Imágenes GNU (GIMP), pero GTK se usa ahora en un amplio número de proyectos de software, incluyendo el proyecto de Entorno de Modelo de Objetos orientados a Red (GNOME). GTK está diseñada encima de GDK (Kit de Dibujo de GIMP) que básicamente es una abstracción de las funciones de bajo nivel para acceder a las funciones del sistema de ventanas (Xlib en el caso del sistema de ventanas X).

Qué es GTK#?

Básicamente GTK es un interfaz orientada a objetos para programadores de aplicaciones (API). Esta herramienta está desarrollada en C++ y requiere de una especie de puente entre el GTK+ y el lenguaje que estemos utilizando. Se han desarrollado una gran cantidad de wrappers para lenguajes como java, C#, Python y hasta Php.

El GTK Sharp (Gtk#) es el puente (wrapper) desarrollado para C#, el cual permite que desarrollemos interfaces gráficas en ambientes MS Windows y Linux.

La biblioteca facilita la creación de aplicaciones gráficas GNOME utilizando Mono o cualquier otro CLR compatible. Gtk# es un sistema orientado a eventos como cualquier otra librería de ventanas moderna, donde cada widget es una aplicación que cuenta con métodos que realizan alguna accion cuando ocurren determinados eventos. Las aplicaciones creadas usando Gtk# se podran ejecutar en varias plataformas incluyendo Linux, Microsoft Windows y Mac OS X.

Desarrollando en GTK#

Cuando se desarrolla para MS Windows se cuenta con los denominados WinForms, los cuales son administrados internamente por el GDI. Los desarrolladores de la plataforma mono han portado esta funcionalidad a través de conversiones hacia un administrador gráfico compatible con Windows y Linux, pero no han logrado ese estado de arte que caracteriza a una aplicación desarrollada para GDI.

En general, las aplicaciones GTK# se escriben usando MonoDevelop, que proporciona un diseñador visual para crear interfaces gráficas de usuario GTK #.

- Plataformas: Unix, Windows, OSX
- Un buen soporte para la accesibilidad.
- Motor de diseño para el manejo de entornos plurilingües, y se adapta al tamaño de fuente sin romper las aplicaciones.
- Las aplicaciones se integran con el escritorio Gnome.
- API familiar para los desarrolladores de GTK +.
- Gran comunidad Gtk+.
- Puerto Win32 disponible, con aspecto nativo en Windows XP.
- Soporte Unicode.

Creación de aplicaciones

oiu - MainWindow.cs* - MonoDevelop

Archivo Editar Ver Buscar Proyecto Construir Ejecutar Herramientas Ventana Ayuda

Barras de herramientas

Contenedores: Alignment, Event Box, Expander, Fixed, Frame, HBox, HButtonBox, HPaned, Notebook, Scrolled Window, Table, VBox, VButtonBox, VPaned

Componentes: Arrow, Button, Calendar, Check Box, Color Button, Color Selection, Combo Box, Combo Box Entry, Custom Widget, Drawing Area, Entry, File Chooser Button, File Chooser Widget, Font Button

Main.cs | MainWindow.cs*

btn_ignorar

Aplicación de prueba

Aceptar | Cancelar | Ignorar

Masculino

Femenino

Fuente | Diseñador

Contorno del documento: MainWindow, vpaned5, hbuttonbox3, button26, button27, btn_ignorar, fixed1

Propiedades | Señales

Name	btn_ignorar
Common Widget Properties	
MemberName	
WidthRequest	
HeightRequest	
Visible	☑
Sensitive	☑
Tooltip	
CanDefault	☐
HasDefault	☐
CanFocus	☑
Events	
ExtensionEvents	None
Show Scrollbars	☐
Button Properties	
Button Type	Text only
Stock Item	
Icon	
Label	_Ignorar
UseUnderline	☑
Response Id	
Extra Button Properties	

Construcción satisfactoria.

Stetic GUI Designer de Mono Develop 2.8.6.3

Contenedores en GTK#

Cuando diseñamos la interfaz gráfica de nuestra aplicación, decidimos los widgets que vamos a utilizar y cómo vamos a organizarlos en la aplicación. Para poder organizar nuestros widgets, utilizamos unos widgets especiales no visibles llamados contenedores.

- WINDOW es el contenedor principal, en el se dispondran el resto de los contenedores.
- El contenedor FIXED pone widgets hijos en posiciones fijas y tamaños fijos. Este contenedor no realiza ninguna gestión automático de la distribución.
- El contenedor ALIGNMENT controla la alineación y el tamaño de su widget hijo.
- El contenedor TABLE organiza widgets en filas y columnas.
- HBOX y VBOX
- HPANED y VPANED
- FRAME

Menues y Toolbars en GTK#

Una barra de menú es una de las piezas más comunes de las aplicaciones GUI. Se trata de un grupo de comandos ubicadas en distintos menús.

Podemos añadir imágenes y aceleradores a los comandos de nuestro menú. Los aceleradores son atajos de teclado para activar un elemento del menú.

Un CheckMenuItem es un elemento del menú con una casilla de verificación. Se puede utilizar para trabajar con las propiedades booleanas.

Las barras de herramientas proporcionan un acceso rápido a los comandos más utilizados. Una barra de herramientas es un panel horizontal o vertical con botones. Estos botones tienen las imágenes o imágenes y texto.

Eventos en GTK#

GTK+ es una librería orientada a eventos, lo que significa que se dormirá en el método Application.Run() hasta que un evento ocurra y el control pase a la función apropiada.

Esta delegación del control se realiza usando la idea de "señales". (Nótese que estas señales no son las mismas que las señales de los sistemas Unix, y no se implementan usando estas, aunque la terminología es casi idéntica). Cuando un evento ocurre, como cuando presionamos un botón del mouse, la señal apropiada será "emitida" por el el control que fue presionado. Así es cómo GTK+ hace la mayoría de su trabajo útil. En Gtk# las señales son representadas mediante el uso de eventos event del leguaje C#. Hay señales que todos los controles heredan, como Destroy, y hay señales que son específicas a cada control, como Toggled en un botón de activación.

Para hacer que un botón realice una acción, tenemos que configurar un manejador de señales que capture estas señales y llame a la función apropiada. Esto se hace añadiendo un método a uno de los eventos del objeto dado.

Eventos en GTK#

Por ejemplo:

```
object.Event += EventHandler(HandlerMetod);
```

donde object es la instancia de Widget (o una de sus derivadas) que estará emitiendo la señal, y el miembro Event es el evento que deseas capturar. EventHandler es el delegado que se usa para la mayoría de los eventos. El argumento HandlerMetod es el método que se quiere que sea llamado cuando ocurre el evento. El método recibe como segundo parámetro un objeto EventArgs, cuya propiedad RetVal puede usarse para desconectar o bloquear el manejador.

El método especificado en el constructor de EventHandler se llama "método de retrollamada", y generalmente es de la forma:

```
public static void HandlerMetod(object obj, EventArgs args)
{ ...
}
```

donde el primer argumento será una referencia al Widget (control) que emitió el evento, y el segundo (args) una referencia a varios datos útiles que podrían servir para el manejador del evento.

Eventos en GTK#

Todos las clases que derivan de Widget poseen un conjunto de eventos que reflejan el mecanismo del sistema de ventanas. Métodos de retrollamada se pueden asignar a cualquiera de estos eventos. Estos son:

ButtonPressEvent
ButtonReleaseEvent
ScrollEvent
MotionNotifyEvent
DeleteEvent
DestroyEvent
ExposeEvent
KeyPressEvent
KeyReleaseEvent
EnterNotifyEvent
LeaveNotifyEvent
ConfigureEvent
FocusInEvent
FocusOutEvent
MapEvent
UnmapEvent
PropertyNotifyEvent
SelectionClearEvent
SelectionRequestEvent
SelectionNotifyEvent
ProximityInEvent
ProximityOutEvent
VisibilityNotifyEvent
ClientEvent
NoExposeEvent
WindowStateEvent

Eventos en GTK#

objeto.señal += new EventHandler (función);

Se pueden tener tantas retrollamadas por señal como se necesite, y cada una se ejecutará por turnos, en el orden en el que fueron conectadas.

Puede eliminarse la retrollamada de la lista mediante:

object.disconnect(id)

También se puede deshabilitar temporalmente los manejadores de señal con los métodos

signal_handler_block() y signal_handler_unblock() .

object.signal_handler_block(handler_id)

object.signal_handler_unblock(handler_id

Desarrollando en Glade

Glade es una herramienta Open Source que permite visualmente diseñar nuestra interfase de usuario utilizando el toolkit de Gtk, luego guarda su representación visual como un archivo XML que puede ser dinámicamente cargado y utilizado por una aplicación que utilice la librería adecuada.

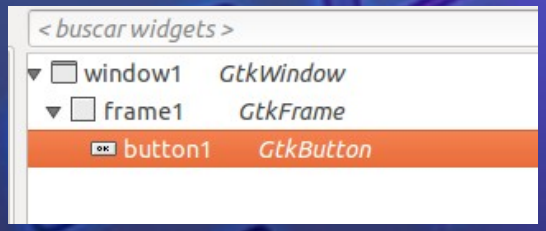
Una de las principales ventajas es la de poder definir mejor nuestra interfaz, lo cual es muy difícil de lograr a “puro código” o “a dedito” como lo llaman algunos. El contar con un lienzo (canvas) donde uno visualmente coloca y organiza objetos de forma visual, permiten que le demos nuestro toque (look-and-feel) similar en toda la aplicación.

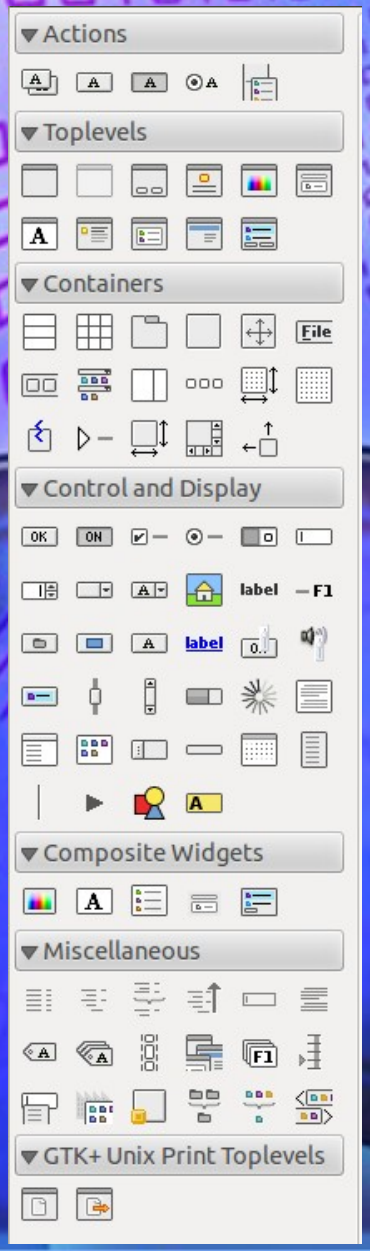
Para interactuar con Glade debemos hacerlo a través de su GUI, representada por sus tres ventanas principales (Proyecto o ventana principal, Paleta de Widgets y Propiedades). Estas tres ventanas interactúan unas con otras.

Desarrollando en Glade

La **paleta de widgets** muestra los controles (llamados widgets en el mundo Gtk) disponibles para diseñar su interfaz de usuario. Usted puede ver el grupo de controles disponibles de forma agrupada.

La ventana del árbol de widgets muestra una estructura jerárquica de los controles existentes en la ventana actualmente seleccionada.





Desarrollando en Glade

La **ventana de propiedades** permite ver y modificar una variedad de propiedades asociadas con el control u objeto que se tiene seleccionado. Incluye cinco categorías (tabs o pestañas) que permiten agrupar estas propiedades.

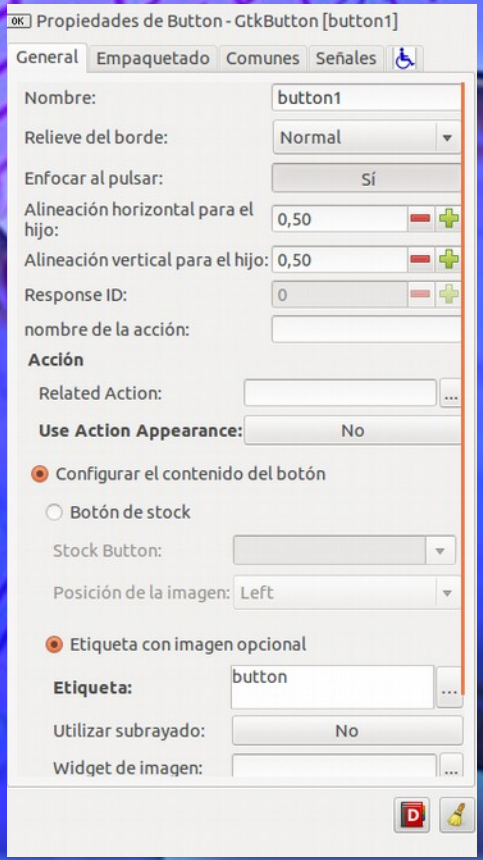
La categoría **Widget o General** contiene las propiedades asociadas al control para su representación visual y propias del tipo de objeto seleccionado.

La categoría **Packing o Empaquetado** es para aquellos objetos que sirven como contenedor de otros, que tienen un arreglo de otros objetos o que permiten definir un orden.

El grupo **Common o Comunes** incluye aquellas propiedades que son comunes a determinados controles.

El grupo **Signals o Señales** permiten asociar capturadores de eventos con los eventos que es capaz de generar un control. Una vez defina un evento, podrá asociarlo a un método dentro de sus aplicaciones, el cual tomará control una vez que el evento suceda

La categoría **Accessibility** permite modificar los atributos para que nuestra aplicación sea más accesible a personas con discapacidades visuales (por ejemplo hacer zoom en algunos controles, aumentando al fuente de letras).



Desarrollando en Glade

De forma automática Glade nos va mostrando como está quedando nuestra GUI. Esto es algo realmente ventajoso por que no tenemos que esperar a ejecutar nuestra aplicación para ver los efectos de diseño logrado.

Antes de colocar objetos en una ventana, debe señalar el widget de la paleta y hacer un clic sobre el punto exacto donde queremos colocarlo. Si no se puede alojar un objeto determinado sobre un contenedor, Glade lo anunciará con un mensaje en una caja de diálogo.

Antes de agregar elementos a una ventana se debe tener clara la distribución que haremos de la GUI. Ahí es cuando entran al escenario los widgets contenedores de otros. Estos nos permiten distribuir los demás objetos en la interfaz que estamos diseñando. Por eso es tan importante tener una imagen trazada de como queremos nuestra interfaz.

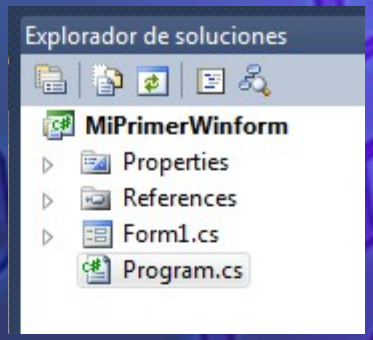
Para construir nuestro código XML, basta con hacer clic en el botón Build (Construir). Glade generará de forma automática el archivo que incluye nuestro diseño GUI. Cuando termina de escribir el código notifica en la barra de tareas que el código ha sido escrito.

Creación de aplicaciones

- Las aplicaciones tradicionales basadas en Microsoft Windows y las aplicaciones cliente-servidor se empiezan a crear mediante el diseño de los formularios utilizando el Diseñador de Windows Forms. Primero se asignan ciertas características a los formularios y se colocan controles en ellos y, a continuación, se escribe el código necesario para ampliar la funcionalidad de los controles y formularios. También es posible heredar de otros formularios.
- Utilizar el diseñador (Windows Forms Designer).
- Agregar todos los formularios necesarios para la aplicación.
- Asignar características a los formularios mediante sus propiedades.
- Agregar controles (drag and drop) a los formularios.
- Escribir código para capturar eventos y aumentar la funcionalidad de los controles y formularios.
- Puede heredar formularios y controles.
- Establecer la forma de ejecución de la aplicación windows.

Ejecución de aplicaciones

- Una aplicación Windows puede tener un número indefinido de formularios.
- La forma de ejecución se encuentra en el archivo Program.cs que es parte de la aplicación.
- Este archivo contiene el método Main() que define el punto de entrada para la aplicación.
- El formulario de inicio es controlado por la sentencia Application.Run(new Form1());
- Form1 puede ser cambiado por cualquier formulario que será el formulario inicial.



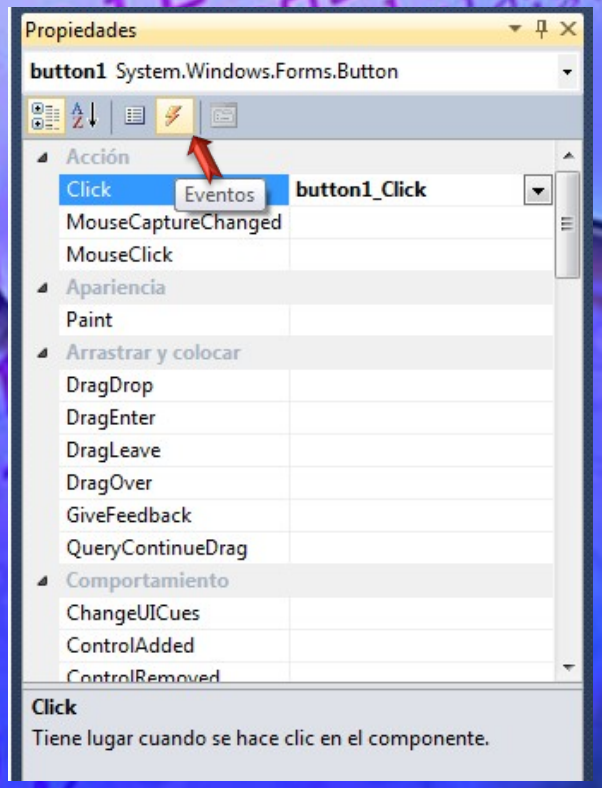

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace MiPrimerWinform
{
    static class Program
    {
        /// <summary>
        /// Punto de entrada principal para la aplicación.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new FormularioInicio());
        }
    }
}
```

Eventos

- Los eventos son parte fundamental de las aplicaciones Windows.
- Se pueden interceptar los eventos y programar la lógica necesaria.
- Los eventos de los controles se pueden establecer en la ventana de Propiedades/Eventos.
- Cuando se utiliza la ventana de propiedades/eventos para crear un manejador de evento, se crea un delegado que referencia al método y lo suscribe al evento. Ejemplo:

```
this.clear.Click += new
System.EventHandler(this.clearClick);
```



Winforms

- Windows.Forms es una unión desarrollada por Microsoft para el conjunto de herramientas Win32. Como una herramienta popular y utilizado por millones de desarrolladores de Windows (especialmente para las aplicaciones empresariales internas), el proyecto Mono decidió producir una aplicación compatible (Winforms) para permitir que estos desarrolladores a portar fácilmente sus aplicaciones se ejecuten en Linux y otras plataformas Mono.
- Considerando que la aplicación. Net es una unión a la caja de herramientas Win32, la ejecución Mono está escrito en C # para permitir que funcione en múltiples plataformas. La mayor parte de la API Windows.Forms funcionarán en Mono, sin embargo, algunas aplicaciones (y especialmente los controles de terceros) de vez en cuando pasar por alto la API y P / Invoke directamente a la API Win32. Estas llamadas probablemente tendrán que cambiar para trabajar en Mono.
-
- En general, las aplicaciones de Windows Forms se escriben utilizando Visual Studio o SharpDevelop de Microsoft, lo que le dará un diseñador visual para crear interfaces gráficas de usuario Winforms.
-
- Plataformas: Windows, Unix, OSX
-

Notación Húngara

- Ésta es una lista de prefijos de control y objeto de Visual Basic proporcionado como referencia para el uso coherente de notación húngara. Esta lista ayudará a estandarizar la estructura, codificación de estilo y la lógica de una aplicación.

Prefijo	Control	Ejemplo
btn	Button (Botones)	btnSalir
cbo	ComboBox (Lista desplegable)	cboCiudad
chk	CheckBox (Casillas de verificacion)	chkImpresora
dat	Data (Control de datos)	datBiblio
dbcbo	DataCombo (Lista despegable enlazada a datos)	dbcboFormato
dbgrd	DataGrid (Cuadrícula enlazada a datos)	dbgrdResultadoConsulta
dblst	DataList (Lista enlazada a datos)	dblstTipoTarea
dlg	Diálogo Común	dlgArchivoAbrir
frm	Form (Formulario)	frmEntrada
gbox	GroupBox(Marco)	gboxLenguaje
gra	Gráfico	graIngresos
grd	Grid (Cuadrícula)	grdPrecios
img	Image (Imagen)	imgBoton
lbl	Label (Etiqueta)	lblStatus
lst	ListBox (Lista)	lstNombres
mdi	MDIForm (Formulario MDI padre)	mdiNota
mnu	Menu	mnuArchivoAbrir
opt	RadioButton (Boton de opción)	optMasculino
pic	PictureBox (Cuadro para imagen)	picStatus
shp	Shape (Forma)	shpCirculo
txt	TextBox (Cajas de texto)	txtNombre

Webgrafía & Licencia:

- Textos tomados, corregidos y modificados de diferentes páginas de Internet, tutoriales y documentos.
- Este documento se encuentra bajo Licencia Creative Commons 2.5 Argentina (BY-NC-SA), por la cual se permite su exhibición, distribución, copia y posibilita hacer obras derivadas a partir de la misma, siempre y cuando se cite la autoría del **Prof. Matías E. García** y sólo podrá distribuir la obra derivada resultante bajo una licencia idéntica a ésta.
- Autor:

Matías E. García

Prof. & Tec. en Informática Aplicada
www.profmatiasgarcia.com.ar
info@profmatiasgarcia.com.ar