Paradigmas de Programación

Práctica II - Curso 2015/16

Definición y requisitos

Esta práctica es una continuación de la práctica anterior en la que vamos a aprovechar las capacidades de las Interfaces Gráficas de Usuario (GUI) para crear una aplicación orientada a entorno de ventanas que permita:

- Iniciar o reiniciar una partida (aunque no esté terminada) permitiendo indicar el nivel deseado
- Mostrar una representación gráfica del estado del tablero
- Indicar las jugadas mediante la pulsación sobre las casillas afectada
- Deshacer la jugada anterior
- Mostrar el número de toques realizados
- Mostrar la lista de records en los niveles ya jugados.

Se utilizará la librería de Python **pygtk**, que permite emular el entorno de ventanas multiplataforma **GTK+** (Linux, Windows, Mac). El objetivo y las normas del juego son idénticas a las de la primera práctica, por lo que el único esfuerzo a realizar es el de definir la interfaz de usuario y tratar adecuadamente los eventos que genera la interacción con el usuario.

Nota: Se debe consultar al profesor encargado de la evaluación de la práctica la posibilidad de utilización de un entorno gráfico distinto de **pygtk**

Objetivos

Una parte importante de la evaluación corresponderá al diseño de la interfaz de usuario, ese diseño debe maximizar los siguientes principios:

- **Facilidad de uso**. Ejemplo: Para introducir una nueva jugada se podría usar un cuadro de texto (con la notación de la práctica anterior), o bien indicarla mediante la pulsación del ratón en las posiciones adecuadas del tablero. Es evidente que para el usuario es más sencilla e intuitiva esta segunda opción.
- **Retroalimentación**: Cada operación debe tener un efecto inmediato visible en la imagen del tablero y los otros elementos de la interfaz.
- **Robustez**: La interfaz debe evitar en lo posible que el usuario cometa errores. Por ejemplo, si no tiene sentido el usar de un control en un momento dado de la ejecución, ese control debería estar desactivado.
- **Información**: Deben existir controles que muestren la información relevante sobre el desarrollo de la partida
- **Diseño e Interactividad:** Se considerarán de forma positiva la inclusión de elementos gráficos, animaciones, etc., siempre que no superen los límites del buen gusto.

Realización de la práctica

La creación de una aplicación orientada a entorno de ventanas en GTK+ consiste en la creación de una serie de objetos que representan los **controles** (widgets) de la aplicación (botones, etiquetas, cuadros de texto, barras de deslizamiento, imágenes, tablas, etc.) que son los elementos que muestran información al usuario y permiten que el usuario interaccione con ellos para introducir información. También existen **objetos auxiliares** (rangos, cajas de eventos, almacén de tablas) que almacenan información auxiliar que pueden necesitar algunos controles, pero que no tienen una interfaz visual.

Los controles se agrupan en **ventanas** y **cuadros de diálogo**. Cada uno de ellos representa la interacción necesaria con el usuario en una determinada etapa de la ejecución de la aplicación. En esta práctica sólo se necesita crear una única ventana, la ventana principal de la aplicación.

La disposición de los controles en una ventana está definida por objetos denominados **contenedores**, los cuales almacenan la información de la posición de cada control con respecto a los otros y la variación de su tamaño vertical y horizontal cuando cambian las dimensiones de la ventana que los contiene. Cada ventana tiene un contenedor principal el cual puede contener a otros en una organización jerárquica. Es posible realizar la práctica usando únicamente contenedores del tipo *horizontal box*, *vertical box* y *table*, aunque no existe problema en que se utilicen contenedores más complejos.

Cuando el usuario interacciona con algún control (por ejemplo, al pulsar un botón), se necesita algún mecanismo para informar a la aplicación y que ésta ejecute el código asociado. En el paradigma de **orientación a eventos** (utilizado de manera casi universal en los entornos de ventanas) cada control puede generar una serie de **eventos** (típicamente producto de una interacción del usuario) y existe un mecanismo para indicar al sistema operativo (que es quien detecta los eventos) que ejecute un determinado código (encapsulado en una función) de nuestra aplicación cuando suceda un determinado evento en uno de nuestros controles. **Nota:** En GTK+ existe una distinción entre **eventos** y **señales**, que sólo afecta a los parámetros que debe definir la función que trate el evento.

El diseño de la interfaz y la **asociación** de los eventos con el código que debe tratarlos puede realizarse directamente mediante código en la propia aplicación, o bien puede utilizarse algún programa auxiliar que ayude en esa tarea. Para ésta práctica se recomienda la utilización del programa GLADE para el diseño de la interfaz y la asociación de eventos.

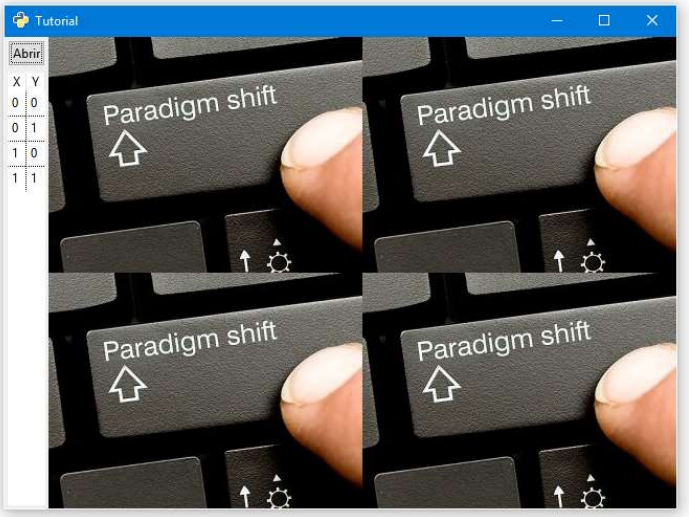
La estructura típica de una aplicación orientada a ventanas consiste en una clase que contiene los atributos y métodos que representan el estado y comportamiento de la aplicación, los métodos que van a procesar los eventos, y en el método de creación del objeto se realizan las siguientes tareas:

- Creación de la ventana principal, sus controles y objetos auxiliares y asignación a cada uno de ellos de los datos y parámetros de visibilidad adecuados.
- Creación del contenedor principal (y de los secundarios, si son necesarios) y agrupación de los controles (y contenedores secundarios) en el contenedor adecuado.
- Asociación, para cada control cuyos eventos queramos tratar, de cada evento con el método que se va a ejecutar cuando suceda el evento.

Mini-tutorial

El objetivo del tutorial es crear una aplicación sencilla que tenga un botón, una tabla de datos y un área central dividida en 4 cuadrantes en los que cada uno de ellos mostrará la misma imagen. Al pulsar el botón un diálogo de sistema nos permitirá elegir un fichero con la imagen que queremos mostrar, repetida 4 veces, en el área central. Al pulsar el ratón en alguno de las imágenes se añadirá en la tabla de datos la información referente al cuadrante en que se encuentra.

La apariencia de la aplicación será parecida a la siguiente:



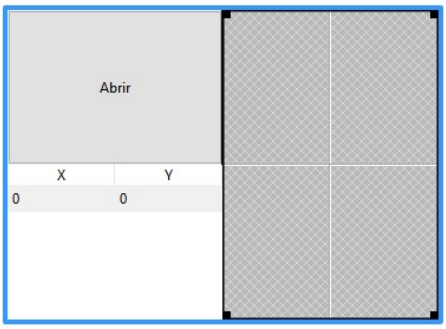
Diseño en Glade:

- Pulsamos en el widget Window (Toplevels) para crear la **ventana principal**. En la pestaña de propiedades [General] cambiamos su nombre (name) a ***ventana***, en título escribimos *Tutorial* y marcamos las casillas de altura y anchura predeterminada. En la pestaña [Common] marcamos *visible* y en la pestaña [Signals], rama [Gtkwidget], escribimos como handler del evento **delete-event** el nombre **on_ventana_delete_event**. Con ello indicamos que queremos responder a ese evento (pulsación del botón de cierre de la ventana) y que usaremos la cadena de texto "on_ventana_delete_event" para asociar un método de nuestra aplicación a ese evento.
- Pulsamos en el widget Horizontal Box (Containers) y en la ventana principal para crear el **contenedor principal**, indicamos que queremos 2 items.
- Pulsamos en el widget Vertical Box (Containers) y en la parte izquierda del contenedor principal, indicando que queremos 2 items. Con ello dividimos en dos mitades la parte izquierda del contenedor principal.
- Pulsamos en el widget Button (Control and Display) y pulsamos en la zona superior izquierda del contenedor principal. Lo renombramos ***btn_abrir***, le asignamos la etiqueta "Abrir" y en la pestaña [Signals] escribimos como handler del evento **clicked** la cadena **on_btn_abrir_clicked**.
- Pulsamos en el widget List Store (Tree Model) para crear un objeto auxiliar que almacenará el contenido de la tabla de datos. Lo renombramos ***aux_tabla***. Definimos dos columnas de contenido entero (gint) y las llamamos X e Y. Añadimos una fila de prueba con los valores 0, 0.
- Pulsamos en el widget Tree View (Control and Display) y pulsamos en la zona inferior izquierda del contenedor principal. Nos pide el modelo TreeView y le asociamos el objeto auxiliar *aux_tabla*. Lo renombramos como ***tabla*** y con el ratón situado sobre el objeto

pulsamos el botón derecho para que surja el menú contextual. Pulsamos en [Edit..], seleccionamos la pestaña [Hierarchy] y pulsamos el botón [Añadir] dos veces para crear las definiciones de las columnas. En cada una de ellas editamos la propiedad Titulo a las cadenas X e Y, respectivamente. Seleccionamos cada columna y pulsamos el botón derecho del ratón para obtener un menú contextual para definir el *traductor* (Cell Renderer) de cada columna. Elegimos [Add child text] y al editar cada nuevo nodo cambiamos la propiedad Texto a la columna adecuada ("X - gint" e "Y - gint" respectivamente)

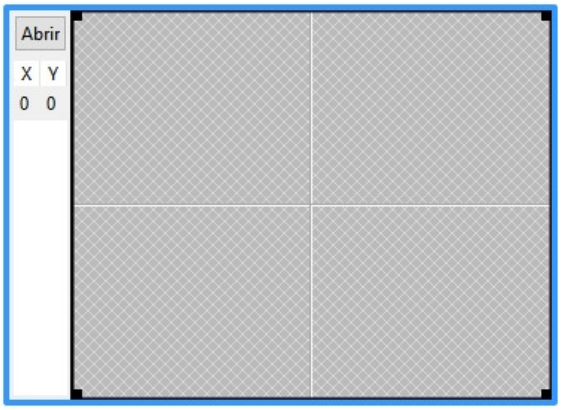
- Pulsamos en el widget Table (Containers) y en la parte derecha del contenedor principal. Elegimos unas dimensiones de 2x2. Lo renombramos como ***área.***

La apariencia de la aplicación en la zona de diseño debe ser similar a lo siguiente:



Nos queda definir los tamaños de los controles y como queremos que se modifique ese tamaño cuando modifiquemos el tamaño de la ventana principal.

Pulsamos en cada control y en la pestaña [Packing] marcamos o desmarcamos adecuadamente la casilla [Expand] para conseguir el aspecto de la imagen. También ajustamos el espaciado de cada elemento para mejorar su apariencia.



Nota: Quedan por añadir a la interfaz los 4 controles de tipo imagen que se incluirán en cada cuadrante del área, y asignarles un manejador al evento *button_clicked* para detectar cuando se ha pulsado el ratón sobre las imágenes. Aunque podríamos hacerlo en Glade, lo realizaremos directamente en el código de la aplicación ya que lo simplifica considerablemente.

Salvamos el fichero como "tutorial.glade" en el mismo directorio del código de la aplicación, eligiendo como formato del fichero "GtkBuilder".

Creamos un fichero Python, "tutorial.py", y escribimos el código de inicialización de la aplicación:

```
import pygtk
import gtk

class Aplicacion:
    def __init__(self):
        # Crear interfaz de usuario
        self.builder = gtk.Builder()
        self.builder.add_from_file("tutorial.glade")
        # Almacenamos referencias a los controles que
        # necesitamos acceder en el codigo
        self.aux_tabla = self.builder.get_object("aux_tabla")
        self.area = self.builder.get_object("area")
        # Enlazamos los eventos con sus manejadores
        eventos = {
            "on_ventana_delete_event": gtk.main_quit,
            "on_btn_abrir_clicked": self.abrir_fichero,
        }
        self.builder.connect_signals(eventos)
        # Creamos los controles imagenes y los almacenamos
        # en una tabla interna. Enlazamos el evento button_press
        # de cada imagen con su manejador, indicando el dato
        # extra (fila, columna) que nos permitira identificarla
        self.tab_img = []
        for fil in range(2):
            self.tab_img.append([])
            for col in range(2):
                # Creamos el control imagen
                img = gtk.Image()
                # Para que se muestre el control
                img.show()
                # Debemos usar una caja de eventos para
                # recibir pulsaciones de raton
                caja = gtk.EventBox()
                caja.add(img)
                caja.show();
                # Enlazamos el evento
                caja.connect("button-press-event", self.click, (fil,col))
                # Incluimos la caja con la imagen en el contenedor area
                self.area.attach(caja,col,col+1,fil,fil+1)
                self.tab_img[fil].append(img)

    def abrir_fichero(self, widget, data = None):
        # ..
    def click(self, widget, event, data = None):
        # ..
```

Podemos apreciar que la llamada **self.builder.add_from_file(..)** hace que se creen los objetos necesarios para representar los controles y resto de elementos de la aplicación. Podemos obtener una referencia a esos elementos usando el método **get_object** con el nombre usado en Glade para nombrar los controles. Para asociar los eventos definidos en Glade con el código de nuestra aplicación definimos un diccionario donde la clave es el nombre utilizado en Glade como "handler" del evento

y el valor es una referencia a un método propio que se ejecutará al producirse el evento (también es posible indicar la referencia a una función externa, como **gtk.main_quit** en el caso del evento de cierre de la aplicación).

Para ejecutar la aplicación se debe crear un objeto de tipo Tutorial y llamar a la función **gtk.main()**, que muestra la ventana principal e inicia el bucle de paso de mensajes (hasta que se produce una llamada a **gtk.main_quit()**).

```
>> import gtk
>> from tutorial import Aplicacion
>> Aplicacion()
>> gtk.main()
```

Si la ejecutamos con el código anterior veremos que se muestra la aplicación pero no responde a eventos (salvo el de cierre de la aplicación). Definimos el código de los eventos:

```
    # -- Manejadores de eventos tipo señal --

    def abrir_fichero(self, widget, data = None):
        # Mostrar dialogo para abrir un fichero
        dlg = gtk.FileChooserDialog(
            "Abrir fichero", None,
            gtk.FILE_CHOOSER_ACTION_OPEN,
            (gtk.STOCK_CANCEL, gtk.RESPONSE_CANCEL,
             gtk.STOCK_OPEN, gtk.RESPONSE_OK))
        if dlg.run() == gtk.RESPONSE_OK:
            nomfich = dlg.get_filename()
            # Cambiamos la imagen en todos los controles
            for fil in range(2):
                for col in range(2):
                    self.tab_img[fil][col].set_from_file(nomfich)
            # Borramos la tabla de pulsaciones
            self.aux_tabla.clear()
        dlg.destroy()

    # -- Manejadores de eventos tipo evento (raton) --

    def click(self, widget, event, data):
        # data contiene la tupla (fila, columna) de la imagen
        self.aux_tabla.append(data)
```

Para conocer el uso de los distintos widgets de GTK+ es recomendable consultar el siguiente tutorial (existe versión en español):

http://www.pygtk.org/pygtk2tutorial/

Para obtener la lista de propiedades y métodos de un elemento concreto se debe consultar la referencia de GTK+ (ordenada por nombre de objeto):

https://developer.gnome.org/pygtk/stable/

Para obtener información sobre el uso de Glade se puede consultar el siguiente tutorial:

https://live.gnome.org/Glade/Tutorials

Presentación y Evaluación de la práctica

La práctica se realizará **por parejas** (para otras alternativas consulte con su profesor) y su evaluación se divide en dos etapas:

1. Presentación electrónica de los ficheros que componen la práctica (los ficheros ***.py**, ***.glade** y otros ficheros necesarios). Para ello se habilitará en el Aula Virtual de la E.T.S. Informática (www.inf.uva.es -> menú Aula Virtual) una tarea de subida de ficheros cuya fecha límite será el **domingo 29 de mayo a las 23:59.** Al principio de todos los ficheros de código debe aparecer un comentario con el nombre de quienes la han desarrollado.
2. Evaluación **presencial**, en laboratorio, ante el profesor. Se realizará en el lugar, día y hora correspondiente al horario de prácticas del subgrupo al que pertenezca durante la semana del 25-29 de mayo.

En el caso de realización por parejas (la situación habitual), tan sólo es necesario que uno cualquiera de ellos realice la presentación electrónica. En la evaluación, sin embargo, si es necesaria la presencia de **ambos** y la evaluación puede ser distinta para cada uno de ellos.

En la evaluación de la práctica se tendrá en cuenta, entre otros, los siguientes factores:

- Autoría y participación en la misma.
- La correcta resolución del problema así como la modularidad, documentación y robustez de la solución presentada. El uso de orientación al objeto y otras técnicas contempladas en la asignatura tiene una influencia positiva en la evaluación.