Los Eventos

Los eventos son cosas que suceden. Ya sea que se presiona un botón del ratón, se redimensiona una ventana o nos tocan el timbre, Delphi procesa estos sucesos (bueno, tal vez el último no) y si hemos definido una respuesta para el mismo, la activa automáticamente.
No obstante, no todos los componentes pueden responder a todos los eventos. Por ejemplo, dado que las etiquetas no pueden ser editadas por el usuario final, estos componentes no responderán a las pulsaciones de teclas.

El código que escribimos en Delphi se ejecuta mayormente en respuesta a eventos.

Los pasos generales para programar un evento son:

- Seleccionar el componente en el que se producirá el evento.
- En el Inspector de Objetos, mostrar la página de eventos.
- Buscar el evento que deseamos.
- En la columna de la derecha (que inicialmente está en blanco) escribir un nombre para el procedimiento y presionar <Enter> o hacer "Doble Click": Delphi escribirá un nombre automáticamente. El cursor pasa a la ventana de edición, entre las palabras reservadas Begin y End, listo para escribir nuestro código.

Eventos del ratón

OnClick
Se produce al presionar y soltar el botón principal del ratón. Este evento existe en prácticamente todos los componentes visibles, dado que siempre es posible presionar el botón principal del ratón sobre ellos.

OnDblClick
Se produce al presionar y soltar el botón principal del ratón rápidamente dos veces sobre el mismo lugar (aproximadamente, con un margen pequeño de movimiento entre cada pulsación). La velocidad de pulsación necesaria para que el sistema considere los dos clicks como uno solo doble se configura en el Panel de Control de Windows.
Generalmente se asigna a este evento la acción por defecto de una caja de diálogo, de manera que el usuario experimentado pueda acelerar la ejecución.

OnMouseDown
Se produce al presionar uno cualquiera de los botones del ratón. En los parámetros del procedimiento se indica el botón presionado, la posición en que se encontraba el puntero cuando se produjo el evento, y si había alguna tecla especial (Shift, Control, Alt) presionada en ese momento.

OnMouseUp
Se produce al soltar uno cualquiera de los botones del ratón. El procedimiento ofrece los mismos parámetros que OnMouseDown.

Estos tres eventos se producen en sucesión cuando se presiona el botón principal del ratón. Primero se genera OnMouseDown, y al soltar el botón se producen OnClick y OnMouseUp.

OnMouseMove
Se produce cada vez que se mueve el puntero del ratón sobre la superficie considerada. Los parámetros incluyen las coordenadas del puntero al momento de producirse el evento y las teclas especiales que se encontraban presionadas.

Eventos de teclado

OnKeyPress
Se produce al presionar y soltar una tecla que corresponda a un carácter simple; no se producirá por ejemplo en respuesta a una tecla de función o alguna de las teclas modificadoras: Shift, Control, Alt.
Parámetros:

- Key: tipo Char, contiene el código ASCII de la tecla presionada.

OnKeyDown
Se produce al presionar una tecla. Ofrece los siguientes parámetros (además de Sender):

• Key: tipo Word. Contiene el *código virtual* de la tecla presionada. Para ver los códigos virtuales busque en la ayuda el tópico *virtual key code*s.

• Shift: es un conjunto de constantes que indican las teclas especiales modificadoras que estaban presionadas al producirse el evento, así como los botones del ratón presionados. Los valores posibles son:

- *ssShift* cualquiera de las teclas Shift está presionada
- *ssCtrl* tecla Control presionada
- *ssAlt* tecla Alt presionada
- *ssLeft* el botón izquierdo del ratón está presionado
- *ssMiddle* ídem para el botón del medio
- *ssRight* lo mismo para el botón derecho
- *ssDouble* los botones izquierdo y derecho del ratón están ambos presionados

Para determinar por ejemplo si estaba presionada la tecla <Ctrl> utilizamos el operador IN:

```
if ssCtrl in Shift then ...
```

OnKeyUp
Este evento se produce cuando se suelta una tecla. Tiene los mismos parámetros que el evento OnKeyDown.

Estos tres eventos se producen en sucesión: cuando presionamos una tecla se produce OnKeyDown; al soltarla se genera un evento KeyPress (si corresponde) y enseguida KeyUp.

El foco de atención

En Windows hay controles que aceptan eventos de teclado y otros que no; normalmente tendremos varios del primer tipo en una sola ventana. Por lo tanto, hay que definir de alguna manera a quién van a parar los eventos de teclado.
Sólo un control a la vez puede recibir estos eventos: se dice que este control tiene el *foco* de atención por parte del sistema.
Generalmente es muy fácil reconocer visualmente al control que tiene el foco: en los controles que aceptan texto aparecerá el cursor parpadeante, y los demás se muestran rodeados por una línea de puntos.
Cuando el foco de atención pasa de un control a otro, se producen dos eventos: OnEnter y OnExit.

OnEnter
Se genera en un control cuando éste recibe el foco. No tiene otros parámetros aparte de Sender.

OnExit
Se genera en un control cuando el mismo pierde el foco. No tiene otros parámetros aparte de Sender.

Control de cambios

Hay otro evento que se produce en varios componentes al momento de cambiar su valor, como por ejemplo los editores al cambiar el texto o las barras de desplazamiento al cambiar la posición del indicador. Es llamado, previsiblemente, **OnChange**. No tiene otros parámetros además de Sender.

Eventos del form

La clase Tform, de la que descienden todos las ventanas de Delphi, define algunos eventos especiales que no se encuentran en otros componentes.

OnActivate - OnDeactivate
Se generan cuando se activa y desactiva la ventana respectivamente, es decir cuando recibe o pierde el foco de atención.
Por ejemplo, si la ventana está cubierta por otra no tiene el foco de atención; si pulsamos el ratón sobre ella la traemos al frente, le damos el foco y se produce el evento OnActivate. Si luego la cerramos o traemos otra ventana al frente, se pierde el foco y se genera el evento OnDeactivate.

OnCloseQuery
Se genera cuando se va a cerrar un form, es una manera de "pedir permiso" al programa para cerrar la ventana. Es muy común utilizarla para preguntar al usuario si se deben grabar datos que se hayan modificado, y según su respuesta permitir o no el cierre de la ventana.
El permiso para cerrar la ventana se da cambiando el valor de un parámetro de tipo lógico llamado **CanClose**. Si le damos valor verdadero (TRUE), la ventana se cierra. Este es el valor por defecto que toma la variable si no le asignamos otro.

OnClose
Este evento es la continuación del anterior: cuando se cierra una ventana se genera primero OnCloseQuery, con el que le damos permiso a la ventana para cerrarse; inmediatamente se genera OnClose para darnos la oportunidad de especificar la acción a tomar con el *objeto* de la ventana, es decir, con la estructura interna que está en la memoria.
La acción deseada se especifica asignando un valor al parámetro **Action**, que puede ser una de las constantes siguientes:

- caNone: indicamos que no se permite cerrarse a la ventana, por lo tanto es equivalente a poner CanClose=FALSE en el evento OnCloseQuery.
- caHide: el form no se cierra, sólo se oculta a la vista. Todavía tenemos acceso a sus propiedades y métodos.
- caFree: el form se cierra y el objeto se destruye, es decir, se liberan todos los recursos que pueda haber ocupado (como la memoria).
- caMinimize: el form se minimiza, no se cierra. Es el comportamiento por defecto de las ventanas tipo MDIChild.

La acción por defecto de un form común es caHide, por lo que al cerrar una ventana -salvo que sea la principal, que cierra la aplicación completa- en realidad la estamos ocultando. Todavía podemos acceder a sus controles y propiedades.
La acción caFree es muy utilizada cuando el form es creado en tiempo de ejecución; de esta manera, nos aseguramos que al cerrar la ventana se libera la memoria y otros recursos ocupados.

OnCreate - OnDestroy
El evento OnCreate se genera al crearse una ventana. Es el lugar ideal para dar valores iniciales a otros controles, como por ejemplo llenar un ListBox o un ComboBox, poner valores iniciales en los editores, crear estructuras propias, etc.
El evento OnDestroy se produce al destruirse una ventana; aquí deben destruirse todas las estructuras y variables propias que fueron creadas en el evento OnCreate.

NOTA: el evento OnCreate se produce siempre al crearse la ventana, antes de aparecer en la pantalla. No obstante, hay que tener en cuenta que la acción por defecto al cerrar un form es su ocultación (vea arriba el evento OnClose) por lo que en ese caso no se producirá OnDestroy al cerrar una ventana. Si en el evento OnClose asignamos el valor caFree al parámetro Action, entonces sí se producirá OnDestroy.

OnShow - OnHide
Estos eventos se producen justo antes que se muestre o se oculte un form, respectivamente. En el momento de mostrar una ventana (llamando a los métodos Show o ShowModal) se produce OnShow. No obstante, *no* se produce cuando la ventana vuelve a mostrarse luego de quedar tapada por otra (en este caso, se produciría OnActivate).
Como regla práctica: OnShow se produce cuando se muestra por primera vez la ventana, o cuando sale de una ocultación previa. OnHide se produce cuando ocultamos la ventana (si recordamos que al cerrar una ventana en realidad se oculta, veremos que también al "cerrar" una ventana se produce OnHide).

OnPaint
Este evento se produce cada vez que se necesita redibujar una ventana. Se utiliza para dibujar algo especial en la superficie de la ventana.
Bajo Windows, cada ventana es responsable por su propio contenido. Esto implica también que cada ventana se debe redibujar cuando sea necesario. Por ejemplo, cuando movemos una ventana que estaba tapando una parte de otra, es necesario redibujar la parte que se reveló de la ventana inferior. Entonces Windows envía un mensaje a la ventana inferior indicando que es necesario redibujar la superficie; Delphi intercepta este mensaje y lo transforma en un evento OnPaint.
Notemos que al redibujar la superficie de una ventana no es necesario hacerlo con los controles que

están sobre la misma; esto se hace automáticamente.

OnResize
Este evento se produce cada vez que cambia el tamaño de la ventana. Se utiliza mayormente para redimensionar los controles de forma acorde con el nuevo tamaño, de manera que sigan manteniendo las proporciones.