UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Controles básicos en *MetaCard* para el desarrollo de aplicaciones multimedia

Apellidos, nombre	Agustí Melchor, Manuel (magusti@disca.upv.es)
Departamento	Departamento de Informática de Sistemas y Computadores
Centro	Escola Tècnica Superior d'Enginyeria Informàtica Universitat Politècnica de València

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

1 Resumen

En este artículo y desde la perspectiva de una herramienta de autor multiplataforma como es MetaCard *[1]*, se va a explorar el uso de los elementos más básicos del interfaz de aplicaciones en un entorno de ventanas: los botones y los campos de texto.

En el presente documento se aborda la utilización de los elementos más básicos de esta herramienta para la realización de aplicaciones de características multimedia. Básicamente, se hará referencia a elementos que constituyen el interfaz de una aplicación.

2 Objetivos

La idea de construcción de una aplicación en *MetaCard* pasa por una primera parte dedicada al desarrollo del interfaz utilizando una aproximación visual: creación de ventanas y disposición de elementos en las mismas sobre los que el usuario interactúa. Aunque existen más controles en una aplicación típica de entorno gráfico de ventanas, en este caso nos centraremos en los dos más básicos:

- Exponer los tipos y operaciones a realizar sobre los controles de tipo botón.
- Exponer los tipos y operaciones a realizar sobre los controles de tipo campo de texto.

No espere el lector actividades complejas a realizar. En ambos casos se acompañará de ejemplos breves, para agilizar y facilitar la recreación de lo que aquí se expondrá por parte del lector. Se asume que el lector está dispuesto a comprobar, con la propia aplicación, que lo que aquí lee es directamente utilizable. En los siguientes apartados se recogen las propiedades y los usos que, en la experiencia del autor, se vienen a utilizar con mayor frecuencia.

3 Introducción

Es conveniente en este punto hacer un pequeño repaso de la nomenclatura que estas herramientas utilizan para hacer mención de un objeto o control:

- La forma general es:

  ```
  tipoDeObjeto nombreDelObjeto [deLaTarjeta] [deLaPila]
  ```

- Donde se indica "el tipo del objeto" mediante uno de los identificadores *stack*, *card*, *button*, *field*, *scrollBar*, *image* o *player*. Aunque también se pueden utilizar abreviaturas, para facilitar la lectura procuraremos evitarlas.

- `nombreDelObjeto` es el identificador que el programador asigna a un control, para permitir la inclusión de caracteres especiales como los espacios en blanco se intentará a lo largo de todos los ejemplos mantener la costumbre de utilizar las comillas dobles para "señalar" qué caracteres son.
- `deLaTarjeta` refleja en qué tarjeta está situado el control y puede obviarse, en cuyo caso se asume que es en la tarjeta activa:

  ```
  this card
  ```

- `deLaPila` refleja a qué pila se está haciendo referencia como contenedora del control y puede obviarse, en cuyo caso se asume que es en la pila por defecto:

  ```
  this stack
  ```

Es posible, pues hacer referencia a objetos en una tarjeta que no es la presente o a otras pilas que estén siendo utilizadas de manera "remota". Pero no dejemos que esto nos preocupe por ahora.

4 Botones (*button*)

Los botones son controles que tienen dos estados diferenciados entre los que cambia al ser pulsados (utilizando el ratón) o activados (mediante el teclado, haciendo uso de alguna combinación de teclas o llevando hasta el botón el foco del teclado y "pulsándolo" con la barra espaciadora).

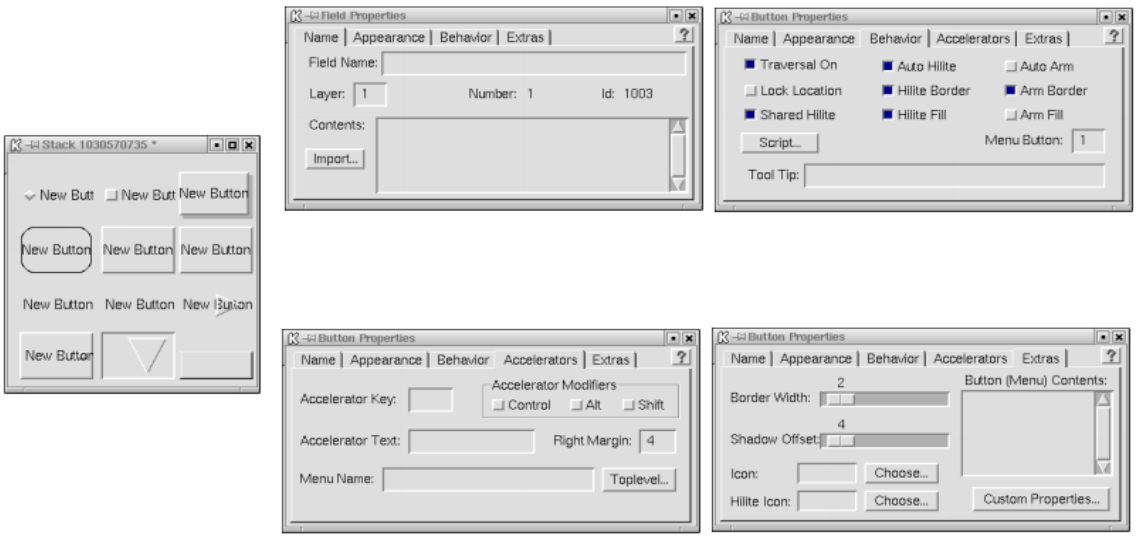

Figura 1: Tipos de botones disponibles en MetaCard y sus propiedades.

Es posible escoger la apariencia de un botón desde las propiedades del mismo. Así se pueden crear desde los más básicos como el "Standard" (y sus variantes "Rectangle", "RoundRect" y "Shadow"). El evento básico es el *mouseUp*.

También los más complejos. Distinguiremos entre los que que llamo desplegables (véase la columna de la derecha del grupo *Style* de las propiedades de un botón) y los de selección, los que guardan un número de opciones (textuales o gráficas).

Los desplegables permiten implementar elementos como un menú ("Menu buttons") o elementos de selección basados en pestañas ("Tabbed buttons"). Los de selección pueden guardar y representar el cambio de estado, hablaremos de botones de selección única (“Check Button”) o múltiple (“Radio button”) en función de si la selección de uno de ellos en un grupo, afecta a los demás. A modo de recopìlación de posibilidades la fig. 1 las muestras reunidas.

4.1 Botones desplegables

Mención aparte merece este tipo de botones que permite implementar menús y otros tipo de objetos que permiten al usuario escoger una opción entre una serie de ítems. Para estos casos se habrá de elegir una de las opciones disponibles a la derecha del menú de estilo de la apariencia de un botón, esto es: *Pulldown*, *Popup*, *Cascade*, *Option*, *ComboBox* o *Tabbed*.

Figura 2: Ejemplo de botones desplegables.

```
on menuPick opcioNova
 put "Ara: "&& opcioNova
end menuPick


on mouseUp
  put the label of button ("desplegable" & the short name of me)
end mouseUp
```

Listado 1. Ejemplo de código aplicable a un botón de tipo despegable.

Como ejemplo de estas características en la fig. 2 se ilustran los conceptos básicos. El lector puede reconstruir la figura, en la que se utiliza el evento **menuPick** y el modo de acceder al contenido de una opción escogida (**label**) en uno de estos botones con el código que muestra el listado 1: el primero de los manejadores (**menuPick**) atiende a la elección realizada sobre un botón (menú) de tipo *Option* y muestra la opción escogida; el segundo (**mouseUp**) permite recuperar la opción elegida, en cualquier momento. desde un elemento de la interfaz.

Para el caso de los botones de tipo *Tabbed*, hay recordar que para estos habrá que disponer grupos de objetos que muestren los contenidos adecuados en cada caso. *MetaCard* se ocupará de mostrar la pestaña activa, pero es labor nuestra actualizar los controles accesibles. Esta tarea es muy simple si se utilizan los mismos nombres para las diferentes opciones y para los grupos. El listado 2 ilustra el mecanismo de código básico para esta tarea.

```
on menuPick opcioNova opcioAnterior
  put "Antes:" && opcioAnterior && "i ara: "&& opcioNova
end menuPick
```

Listado 2. Ejemplo de código aplicable a un botón de tipo Tabbed.

4.2 Botones de selección

La parte de selección de opciones, en general, se implementa con botones de selección múltiple con "Check buttons" o única (también denominada exclusiva) mediante "Radio buttons", véase fig. 3.. Los primeros cambian de estado a cada pulsación: véase la diferencia entre el código en la columna izquierda y la derecha del listado 3.

Figura 3: Botones de selección: mínima aplicación para las pruebas.

Allí podrá ver junto al ya conocido evento de **mouseUp** el uso de la propiedad hilite (que no es aplicable por igual a un subtipo y otro de botón) y la orden **hilite** y **unhilite** que nos permiten cambiar ese estado (e incluso el comportamiento habitual) por código. El listado 3 muestra dos conjuntos de código para los botones anteriores que muestran las posibilidades comentadas. Pruebe primero el código de la fila superior y compare su resultado con el de la inferior.

Primer conjunto:

 # Botón "Radio"

```
on mouseUp
  put the hilite of me
end mouseUp
```

 # Botón "Check"

```
on mouseUp
  put the hilite of me
end mouseUp
```

Segundo conjunto:

 # Botón "Radio"

```
on mouseUp
  put the hilite of me
  unhilite me
end mouseUp
```

 # Botón "Check"

```
on mouseUp
  put the hilite of me
  hilite me
end mouseUp
```

Listado 3. Botones de selección: propiedad y orden *hilite*.

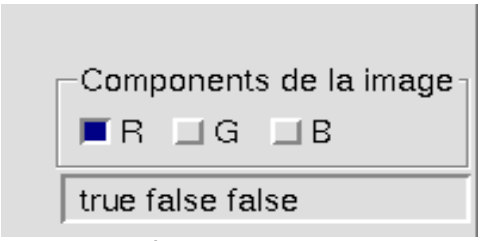

Figura 4: Ejemplo de selección mediante botones de selección múltiple.

```
# Cualquiera de los tres botones de selección
on mouseUp
  # Quin s'ha pulsat
  put the number of target  - the number of the first button of me + 1 into botoPulsat
  put not the word botoPulsat of field "llistaComponentsActives" into \
      word botoPulsat of field "llistaComponentsActives"
  --  put  field "llistaComponentsActives"
end mouseUp
```

Listado 4. Código para el ejemplo de selección mediante botones de selección múltiple.

Para que los de selección única funcionen en ese modo exclusivo, característica que les da nombre, sólo hay que constituirlos en un grupo. La fig. 4 muestra un

conjunto de tres botones de selección múltiple agrupados bajo un contenedor genérico de grupo de objetos. Este es quien recoge la pulsación del ratón y en función de sobre qué elemento de su ámbito se ha realizado mantiene actualizado en un campo de texto (para facilitar la visualización) el estado de pulsado o no de los botones que lo componen.

En determinados casos, cuando la selección no es singular, es necesario que se habilite un punto donde la selección del usuario se considere viable o no. Por ello, el listado 4 muestra un pequeño fragmento de código que permite llevar cuenta en un vector de las opciones escogidas. En ese mismo momento se pueden determinar incompatibilidades u opciones excluyentes que habrá que hacer llegar al usuario en forma de un aviso o sugerencia a gusto del desarrollador.

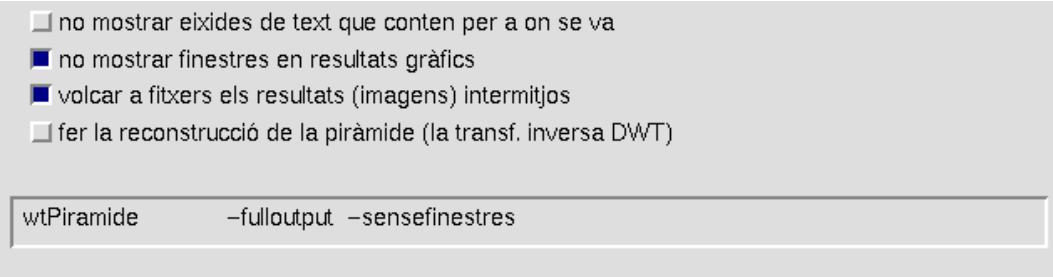

Figura 5: Composición de la línea de órdenes a ejecutar al seleccionar algunas opciones.

Otro uso habitual de los botones de selección es para incluir o no opciones en un caso que admita esta variabilidad. Al activar o desactivar uno de ellos en el campo situado en la zona inferior se ve reflejado la aparición de la cadena correspondiente al botón seleccionado. La fig. 5 muestra una pequeña captura de esta "sociedad" de controles en funcionamiento. De este manera, en todo momento hay una cadena de texto que refleja la situación de los botones y que, en el caso mostrado, se ha utilizado para crear una línea de órdenes cuya ejecución se le encargará al operativo de la máquina en un instante posterior. En este caso el interés estriba en la forma que se ha implementado la detección del estado de seleccionado (activado) de un botón, así como en la forma genérica del tratamiento.

La fig.6 muestra las propiedades características de uno de los botones que se emplean en el ejemplo, para los cuatro el planteamiento es similar. El listado 5 muestra el código asociado a cualquiera de ellos.

Figura 6: Botones de selección que tratan, de forma dinámica, la activación o desactivación de su estado.

```
on mouseUp
  get the properties of me
  if it["hilited"] is true
  then
    put (" " & the short name of me) after fld "ordreAExecutar"
  else
    put offset( the short name of me, fld "ordreAExecutar" ) into tOffset
    delete char tOffset to (tOffset + length(the short name of me)) of fld "ordreAExecutar"
  end if
end mouseUp
```

Listado 5. Código para cualquiera de los botones del ejemplo de la fig. 5.

5 Campos de texto (*field*)

Son los controles que se han de emplear en *MetaCard* para mostrar en pantalla contenidos textuales de una forma directa. En la mayor parte de los casos se utilizarán para recoger información del usuario y para permitir a este editarla. También es frecuente utilizarlos, a modo estático, para proporcionar información al usuario, para este caso en las propiedades de estos controles existe la opción "Show as Label" que elimina todas las propiedades de decoración y posible manipulación del texto que albergan. Junto a estas opciones la posibilidad de seleccionar su contenido en forma de lista con selecciones únicas o de múltiples líneas son las más utilizadas. Como ejemplo de esta variedad se puede observar la fig. 7 .

Centrémonos ahora en describir qué tipos de operaciones para manipulación del contenido admite el control de tipo campo de texto. Se puede controlar el punto de inserción del texto que resulta de ejecutar la orden **put** con los modificadores **into**, **after** y **before** como muestra el listado 6. Estos ejemplos pueden ser ejecutados directamente en la "*Menu Bar*" tecleando cada una de las líneas siguientes por separado:

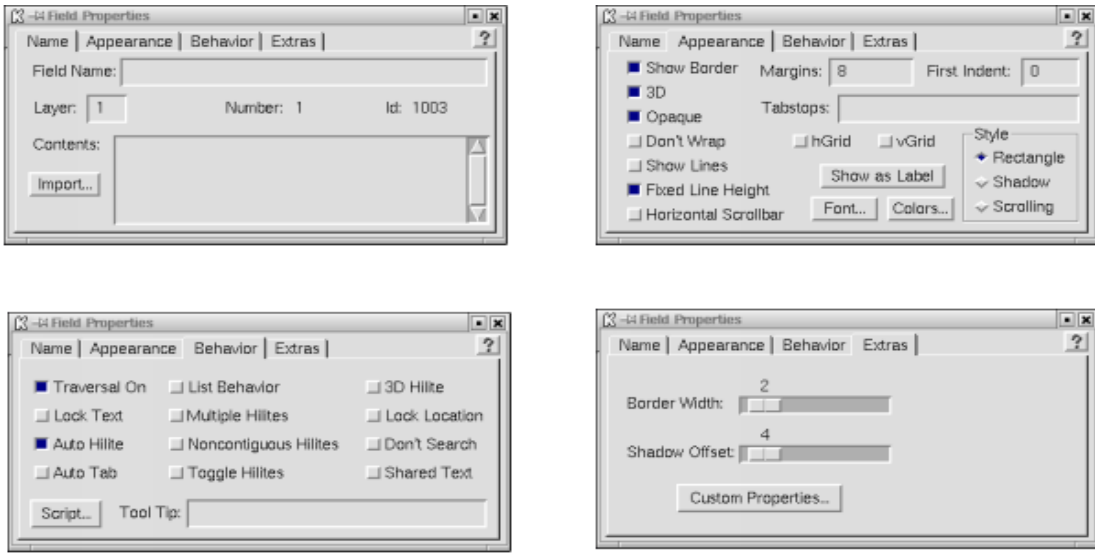

Figura 7: Aspecto y propiedades de un objeto tipo "field" (campo de texto).

```
put "Algo" into varAux; put empty into varAux; put varAux
put "Algo" into varAux; put " més" after varAux; put varAux
put "Algo" into varAux; put "Di: " after varAux; put varAux
```

Listado 6. Ejemplos de modificación del punto de inserción.

También se pueden restringir las operaciones a una selección del mismo: un trozo o parte (*chunks* en la nomenclatura de *MetaCard*). En concreto, es posible hablar de:

- **character** para indicar el más bajo de estos niveles y que involucra a, como parece obvio, uno sólo de los caracteres.
- **word** se utiliza para indicar el siguiente nivel y permite seleccionar las "palabras" de un texto, esto es, lo que está delimitado por espacios en blanco.
- **line** que permite la selección a más alto nivel de estas tres, por "lineas de texto". En este caso el delimitador es el separador de líneas o salto de línea (`return`).
- **item** se utiliza para indicar otro tipo de separador diferente del espacio en blanco o del salto de lineal y permite seleccionar las "entidades" de un texto que están separadas por el carácter deseado y que se ha de indicar utilizando **itemDelimiter** si no es el valor por defecto (el carácter ",").

Veamos ejemplos de aplicación de estos operadores sobre un contenido, para especificar la parte del texto a que se aplica y así:

- Obtener el número de cada uno de ellos, indicar el cardinal o el ordinal o indicar rangos de elementos que se quiere consultar o modificar a cualquiera de los niveles que es posible señalar, por ejemplo, con

```
put the number of characters of "1234567890"
put the word 1 of "Hola, Mon! Un, dos, tres, provant. Tot està com deu estar?"
put the second word of "Hola, Mon! Un, dos, tres, provant. Tot està com deu estar?"
put the item 1 to 4 of "Hola, Mon! Un, dos, tres, provant. Tot està com deu estar?"
```

- Componer texto en tiempo de ejecución. Por ejemplo, líneas con

```
put "Hola, Mon!" & return & "Un, dos, tres, provant." & return & "Tot està com deu estar?"
```

- Buscar subcadenas o recursos con la orden **is**. En sus diferentes variantes es posible preguntar por la existencia de objetos en la pila o recursos externos (como p. ej. ficheros) y para comparar números o cadenas de texto como tales.

```
put "3" is among the items of "22,33,44" # devuelve  false
```

- Operadores básicos. Como los lógicos, cuya evaluación devuelve un valor lógico de cierto (**true**) o falso (**false**). Como son los **()** el **not**, **and** y **or**; o los que permiten comparar números y cadenas de texto como el **<**, **<=**, **>**, **>=**, **=** y **><**. También los aritméticos, de uso con valores numéricos como el **+**, **-**, *, **/**, **div**, **mod**, **^**. Y, como no, los de manejo de bits, que operan sobre la representación en binario de valores numéricos y sólo se pueden aplicar a estos. Se dispone de **bitNot, bitAnd, bitOr, bitXor**.

- De comprobaciones y como validador de tipos (**date**, **number**, **integer**, **point**, **rect** y **boolean**), como en

```
is within     # para comprobar si unas coordenadas están dentro de un área
is not a(n)   # para comprobar el tipo de un objeto
if field "size" is a number then add 10 to field "size"
```

6 Conclusiones

A lo largo de este documento hemos visto el uso de los controles más básicos del interfaz típico de una aplicación desarrollada con *MetaCard*: los botones y los campos de texto.

La exposición de sus tipos y modos de operación se ha acompañado de ejemplos breves para que el lector haya podido llevarlos a cabo sin mucha demora en su elaboración.

7 Bibliografía

[1] MetaCard <http://www.metacard.com/>.