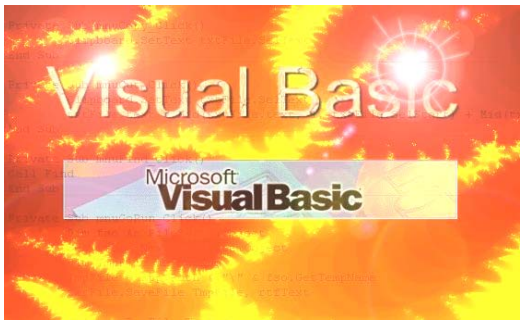
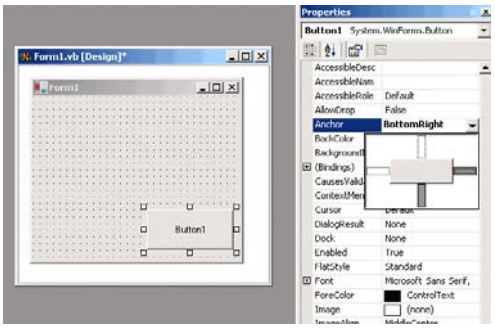
CURSO

Curso Completo de Visual Basic 6.0





Escuela Superior de Ingenieros Industriales
UNIVERSIDAD DE NAVARRA
Javier García de Jalón · José Ignacio Rodríguez
Alfonso Brazález · Patxi Funes · Eduardo Carrasco · Jesús Calleja

5.2 EL EDITOR DE MENÚS (MENU EDITOR)

En la Figura 5.4 se vuelve a recoger –a mayor tamaño y con algunos elementos ya definidos- el editor de menús mostrado en la **Figura 5.1**, que se abre con ***Tools/Menu Editor*** o clicando en el botón correspondiente de la barra de herramientas.

Se llama ***título*** a cada elemento que aparece en la barra de menús y ***línea*** o ***ítem*** a cada elemento que aparece al desplegarse un título. Para introducir un nuevo título en la barra de menús hay que definir el nombre con el que se quiere que aparezca en la caja de texto ***Caption*** de la Figura 5.4. Si se desea acceder a dicho título mediante teclado (***Alt+letra***), la letra que se desea utilizar deberá ir precedida por el carácter (&). Además, y al igual que todos los controles de ***Visual Basic 6.0***, conviene que el título tenga un *nombre* (caja de texto ***Name***) para que se pueda acceder a él desde programa. Los nombres de los títulos de los menús suelen comenzar por las letras ***mnu***, como por ejemplo ***mnuFile***, ***mnuEdit*** o ***mnuHelp***.

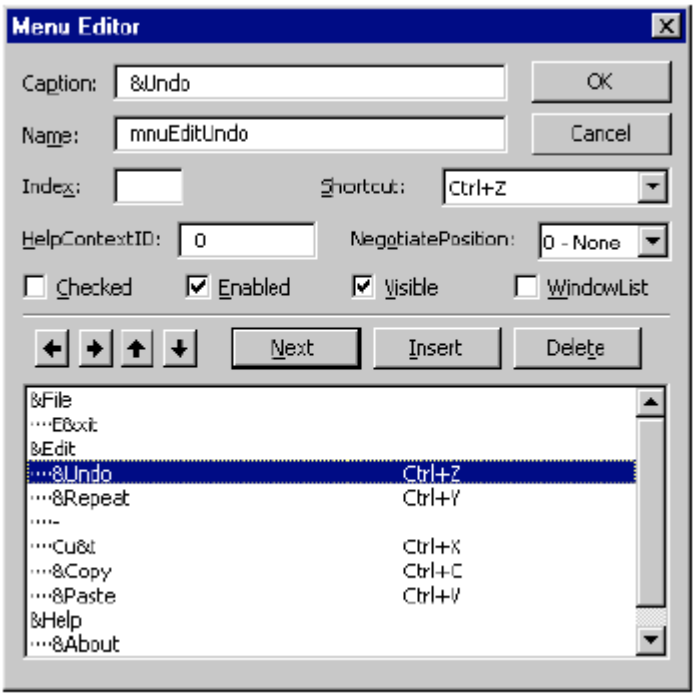


Figura 5.4. Definición de menús con *Menu Editor*.

En la **Figura 5.4** la caja de texto ***Index*** hace referencia a la posibilidad de crear *arrays de menús*. Se puede definir también un ***shortcut*** en la caja de texto correspondiente. En esta figura aparecen cuatro ***checkButtons*** (***Enabled***, ***Checked***, ***Visible*** y ***WindowList***) con los que se pueden especificar algunas propiedades iniciales del menú, como por ejemplo que esté activado o que sea visible.

Se pueden introducir ***items*** subordinados a un ***título*** por medio de la *flecha hacia la izquierda*. Para ello basta definirlos del modo habitual y luego clicar sobre dicha flecha. El resultado es que aparecen unos puntos a la izquierda del caption correspondiente.
Por ejemplo, en el menú definido en la Figura 5.4, ***Exit*** es una línea subordinada del menú ***File***, mientras que ***Undo***, ***Repeat***, ***Cut***, ***Copy*** y ***Paste*** son items subordinados del menú ***Edit***. En este último caso se ha introducido una línea de separación entre ***Repeat*** y ***Cut***; para ello basta introducir un *ítem* más cuyo ***caption*** sea el carácter *menos* (-).

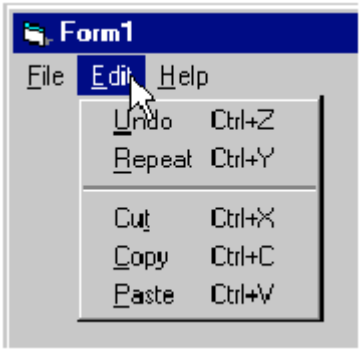


Figura 5.5. Menú ***Edit*** sencillo.

La **Figura 5.5** muestra el resultado de ejecutar la aplicación cuyos menús están definidos en el ***Menu Editor*** de la **Figura 5.4**. Obsérvese la línea horizontal de separación, los shortcuts y las letras subrayadas para poder abrir el menú desde teclado.

Respecto a los nombres de los items, lo habitual es seguir la nomenclatura que ya se muestra en la **Figura 5.4** para ***Undo***: primero se ponen las tres letras ***mnu***, y después los nombres del título y del ítem comenzando por mayúscula: ***mnuEditUndo***. Caso que haya menús en cascada, se ponen los distintos nombres siguiendo estas mismas normas. De esta forma siempre queda claro a partir del nombre a qué elemento del menú se está haciendo referencia.

La **Figura 5.4** es bastante auto-explicativa respecto a cómo se debe proceder para estructurar un menú, añadiendo, borrando o cambiando de posición los distintos elementos. De forma resumida, se pueden establecer las siguientes normas generales:

1. Para insertar un título o ítem basta seleccionar la línea por encima de la cual se quiere insertar y clicar en el botón ***Insert***. Para añadir un nuevo ítem al final de la lista se selecciona el último elemento introducido y se clica en el botón ***Next***. Para borrar un elemento, se selecciona y se clica en el botón ***Delete***.

2. Se puede cambiar de posición un título o ítem seleccionándolo y clicando en los botones que muestran las flechas hacia arriba y hacia abajo. Para convertir un título en ítem o para cambiar el nivel de un ítem se selecciona y se utilizan las flechas hacia la derecha y hacia la izquierda.

Conviene recordar que los nombres de los items (por ejemplo *mnuEditCopy*) deben estar siempre de acuerdo con su posición, según las normas explicadas anteriormente.

5.3 AÑADIR CÓDIGO A LOS MENÚS

Los items de los menús *admiten un único evento*: el evento ***click***, que consiste en ser seleccionados por medio del ratón o del teclado. Para añadir el código correspondiente basta elegir en el menú, estando en modo diseño, el ítem correspondiente para que se abra la ventana de código en el procedimiento ligado a ese evento. También puede buscarse directamente el objeto y el evento correspondiente en las listas desplegables de la ventana de código.

En ocasiones habrá que cambiar las propiedades ***checked, active*** y ***visible*** desde los procedimientos. A estas propiedades se accede del modo habitual, con el nombre del ítem y el operador punto (.)

5.4 ARRAYS DE MENÚS

De la misma manera que pueden definirse arrays de controles, también pueden definirse arrays de items (y de títulos) en un menú. La ventaja de definir arrays de items es que *basta definir un único procedimiento* que se haga cargo del evento ***click*** de todos los items del array. Este procedimiento recibe como parámetro la variable entera ***Index***, que indica que ítem del array ha sido seleccionado por el usuario. Dentro de este procedimiento se podrá utilizar por ejemplo la sentencia ***Select Case*** para tratar de forma adecuada cada uno de los casos.

5.5 EJEMPLO: MENÚ PARA DETERMINAR LAS CARACTERÍSTICAS DE UN TEXTO

La Figura 5.6 muestra un formulario que contiene una caja de texto con una frase ("*Visual Basic es el lenguaje de programación que hace más fácil el desarrollar aplicaciones para Windows*") a la que se puede dar formato desde el menú ***Text***.
El menú ***Text*** tiene tres submenús: ***Font***, ***Size*** y ***Style***. El menú ***File*** sólo tiene la opción ***Exit***, que termina la ejecución.

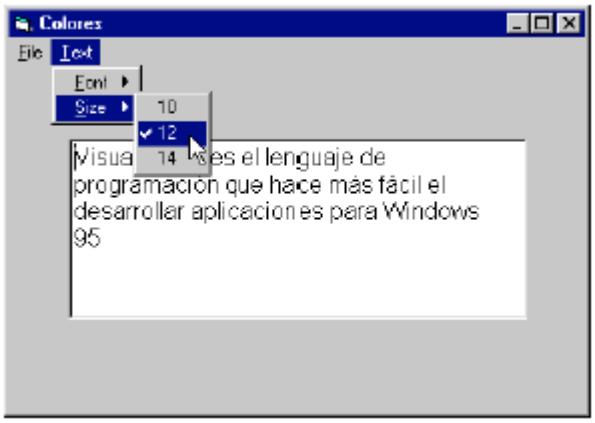


Figura 5.6. Caja de texto con formatos desde menú.

El sub-menú ***Font*** tiene tres opciones: ***Arial***, ***Courier New*** y ***Times New Roman***.
El sub-menú ***Size*** tiene 5 opciones: 10, 11, 12, 13, y 14. El sub-menú ***Style*** tiene 2 opciones: ***Bold*** e ***Italic***. Los tipos de letra y los tamaños deben actuar como los *Option Buttons*: sólo una opción puede estar seleccionada para el texto de la caja.
Sin embargo, los estilos ***Bold*** e ***Italic*** actúan como *Checked Boxes*: el texto puede ser a la vez ***Bold*** e ***Italic***, y puede no ser ninguna de las dos cosas.

Para los tamaños de letra se debe utilizar un array de menús con cinco elementos (propiedad ***Index*** de 0 a 4). Se deja al usuario que ponga los nombres que desee a los controles de la Figura 5.6, o que utilice los del código del programa que se muestra a continuación. Nótese que con los menús que se comportan como *Option Buttons* la propiedad ***Checked*** se pone a ***False*** en todas las opciones antes de poner a ***True*** la que el usuario ha elegido. Con el menú que se comporta como *Checked Box* simplemente se cambia la propiedad ***Checked*** de ***True*** a ***False*** o viceversa, cuando el usuario la elige. El código se muestra a continuación:

```
Option Explicit

Private Sub Form_Load()
      txtBox.Text = "Visual Basic es el lenguaje de programación " & _
        "que hace más fácil el desarrollar aplicaciones para Windows 95"
      txtBox.Font.Name = "Arial"
      mnuTextFontArial.Checked = True
      txtBox.Font.Size = 10
      mnuTextSizeA(0).Checked = True
      txtBox.Font.Bold = False
      txtBox.Font.Italic = False
End Sub

      Private Sub mnuFileExit_Click()
End
End Sub

      Private Sub mnuTextFontArial_Click()
      mnuTextFontCou.Checked = False
      mnuTextFontTimes.Checked = False
      txtBox.Font.Name = "Arial"
      mnuTextFontArial.Checked = True
End Sub









Private Sub mnuTextFontCou_Click()
      mnuTextFontArial.Checked = False
      mnuTextFontTimes.Checked = False
```

```
      txtBox.Font.Name = "Courier New"
      mnuTextFontCou.Checked = True
End Sub

Private Sub mnuTextFontTimes_Click()
      mnuTextFontArial.Checked = False
      mnuTextFontCou.Checked = False
      txtBox.Font.Name = "Times New Roman"
      mnuTextFontTimes.Checked = True
End Sub

Private Sub mnuTextSizeA_Click(Index As Integer)
      Dim i As Integer
      For i = 0 To 4
      mnuTextSizeA(i).Checked = False
      Next i
      Select Case Index
            Case 0
                  txtBox.Font.Size = 10
            Case 1
                  txtBox.Font.Size = 11
            Case 2
                  txtBox.Font.Size = 12
            Case 3
                  txtBox.Font.Size = 13
            Case 4
                  txtBox.Font.Size = 14
      End Select
            mnuTextSizeA(Index).Checked = True
      End Sub

Private Sub mnuTextStyleBold_Click()
            txtBox.Font.Bold = Not txtBox.Font.Bold
            mnuTextStyleBold.Checked = Not mnuTextStyleBold.Checked
      End Sub

Private Sub mnuTextStyleItalic_Click()
            txtBox.Font.Italic = Not txtBox.Font.Italic
            mnuTextStyleItalic.Checked = Not mnuTextStyleItalic.Checked
      End Sub
```

5.6 MENÚS CONTEXTUALES (POPUP MENU)

Los ***menús contextuales*** aparecen cuando el usuario clica con el botón derecho sobre un elemento de la aplicación. El programa debe reconocer el evento ***MouseUp*** o ***MouseDown***, ver si el usuario ha clicado con el botón derecho (argumento ***Button*** igual a 2) y llamar al método PopupMenu, que tiene la siguiente forma general:

```
            PopupMenu menuname [,flags[,x[,y]]]
```

donde ***menuname*** ee el nombre de un menú (con al menos un elemento), ***x*** e ***y*** son las coordenadas base para hacer aparecer el menú contextual, y ***flags*** son unas constantes que determinan más en concreto dónde y cómo se muestra el menú.
Las constantes que determinan dónde aparece el menú: *vbPopupMenuLeftAlign* (default), *vbPopupMenuCenterAlign* y *vbPopupMenuRightAlign*. Por otra parte *vbPopupMenuLeftButton* (default) y *vbPopupMenuRightButton* determinan si el comando

se activa con el botón izquierdo o con cualquiera de los dos botones. Las constantes se combina con el operador ***Or***. El nombre del menú que aparece en el método ***PopupMenu*** debe haber sido creado con el ***Menu Editor***, aunque puede tener la propiedad ***Visible*** a ***False***, si no se desea que se vea.

6. GRÁFICOS EN VISUAL BASIC 6.0

Visual Basic 6.0, además de hacer fácil la construcción de interfaces gráficas de usuario, tiene también grandes posibilidades gráficas en lo que se refiere a dibujo de líneas y formas geométricas, así como en lo referente a la introducción de gráficos y figuras realizados con otras aplicaciones. En este capítulo se presentarán brevemente las posibilidades gráficas más importantes de ***Visual Basic 6.0***.

6.1 TRATAMIENTO DEL COLOR

Antes de ver cómo se dibuja en ***Visual Basic 6.0*** se verá cómo se definen los colores. Al igual que en tantas aplicaciones informáticas, los colores de ***Visual Basic*** se definen por medio de las componentes fundamentales RGB (***Red***, ***Green*** and ***Blue***). La intensidad de cada color fundamental se define con un ***byte***, es decir con un número entero entre 0 y 255. Se utilizan pues tres bytes para definir los tres colores. ***Visual Basic 6.0*** utiliza un entero *long* (32 bits, 4 bytes) para guardar un color, lo cual quiere decir que existe un byte adicional donde se podrá guardar alguna otra información.

6.1.1 Representación hexadecimal de los colores

Para los números enteros entre 0 y 255 se utilizan dos dígitos hexadecimales. Con esta notación el cero es el “00” y el 255 el “FF”. El número que indica el color va precedido por el carácter “&” y la letra “H”. Así, el color verde se define en la forma: &H00FF00.

Visual Basic 6.0 dispone también de nombres para los colores fundamentales y los que son combinación de los colores fundamentales, según puede verse en la Tabla 6.1.

Nombre	Código HEX	Color
vbBlack	&H000000	Negro
vbRed	&H0000FF	Rojo.
vbGreen	&H00FF00	Verde.
vbYellow	&H00FFFF	Amarillo.
vbBlue	&HFF0000	Azul.
vbMagenta	&HFF00FF	Magenta.
vbCyan	&HFFFF00	Cyan.
vbWhite	&HFFFFFF	Blanco.

Tabla 6.1. Nombres de colores.

6.1.2 Acceso a los colores del sistema

El cuarto byte (en el entero *long* que contiene el color) puede utilizarse para hacer referencia a los ***colores del sistema***. Los colores del sistema son aquellos colores con los que ***Windows*** representa las ventanas y sus bordes, las barras de desplazamiento, etc. Dichos colores se eligen en el panel de control ***Display/Appearance***, y ***Visual Basic 6.0*** permite acceder a ellos a través de su nombre o de su valor hexadecimal, que empieza por "&H8" y utiliza el cuarto byte. **La Tabla 6.2** muestra algunos de estos valores. Para una descripción completa buscar ***Color Constants*** en el ***Help*** de ***Visual Basic 6.0***.

Nombre	Valor	Descripción
vbScrollBars	&H80000000	Scroll bar color.
vbDesktop	&H80000001	Desktop color.
vbActiveTitleBar	&H80000002	Color of the title bar for the active window.
vbInactiveTitleBar	&H80000003	Color of the title bar for the inactive window.
vbMenuBar	&H80000004	Menu background color.
vbWindowBackground	&H80000005	Window background color.
vbWindowFrame	&H80000006	Window frame color.
vbMenuText	&H80000007	Color of text on menus.
vbWindowText	&H80000008	Color of text in windows.
vbTitleBarText	&H80000009	Color of text in caption, size box, and scroll arrow.
...	...	...

Tabla 6.2. Colores del sistema.

6.1.3 Función RGB

Esta función devuelve un número que representa un color a partir de tres argumentos enteros entre 0 y 255, que son sus componentes RGB. Como ejemplo de uso:

```
form1.BackColor = RGB(127, 127, 64)
```

Si alguno de los argumentos tiene un valor mayor que 255, se toma como 255.

6.1.4 Paleta de colores

Elegir adecuadamente un color a partir de sus componentes RGB no es una tarea fácil. Por eso ***Visual Basic 6.0*** proporciona una paleta de 64 colores predefinidos, 16 de los cuales pueden ser definidos a medida por el usuario.

La Figura 6.1 muestra la paleta de colores, que aparece con ***View/Color Palette***.

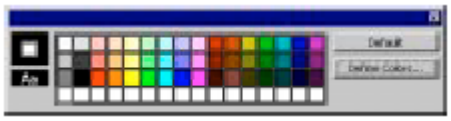


Figura 6.1. Paleta de colores.

La Figura 6.2 muestra el cuadro de diálogo que se abre al pulsar el botón ***Define Colors...*** en la parte inferior dcha. de la Figura 6.1. Para elegir un color se pueden introducir directamente los valores RGB, pero también se puede clicar en el mapa de colores de la parte superior izda. y luego mover el cursor de la parte superior dcha. Finalmente, clicando en el botón ***Add Color***. El color seleccionado se añade en la parte inferior de la paleta de colores (Figura 6.1).

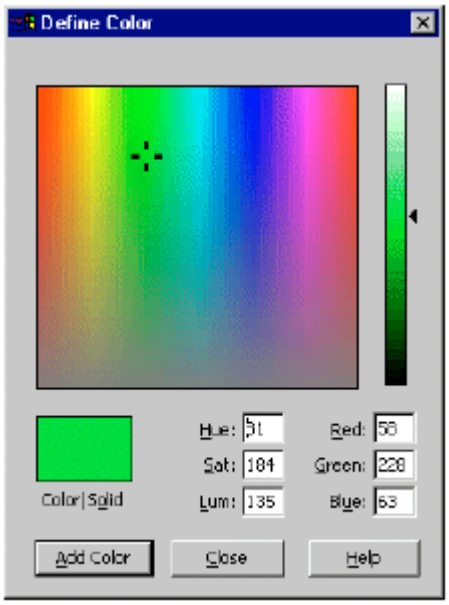


Figura 6.2. Creación de colores a medida.

Por supuesto es necesario tener en cuenta el número de colores soportado por la tarjeta gráfica del PC en el que se esté trabajando. Lo más frecuente es que los PCs estén configurados para soportar 256 colores (8 bits por pixel), 65.536 colores (16 bits por pixel) o 16.777.216 colores (24 bits por pixel). Si la tarjeta gráfica soporta 65.536 colores se elige el color más cercano al que el usuario ha querido representar con la función RGB, por ejemplo. Si la tarjeta gráfica soporta sólo 256 colores se utiliza el *dithering*, que consiste en mezclar pixels de distintos colores con objeto de obtener un efecto lo más parecido posible al color solicitado.

Una vez añadidos los colores a la paleta, al clicar en el pequeño triángulo que aparece en cualquier propiedad de color en la ventana de ***Properties*** aparecerán una ventana donde es posible elegir entre los *colores de la paleta* y los denominados *colores del sistema* (Figura 6.3).

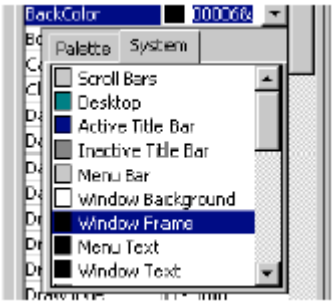


Figura 6.3. Colores de sistema

El ejemplo 1.5.4 (Colores RGB) mostrado en el Capítulo 1 de este manual es un buen ejemplo de la utilización de los colores en ***Visual Basic 6.0***.

6.2 FORMATOS GRÁFICOS

En un formulario de ***Visual Basic 6.0*** -y en los controles ***image*** y ***picture***- es posible insertar gráficos, tanto de tipo *bitmap* (los producidos por aplicaciones como ***Paint***, ***Paintbrush***, ***Paint Shop Pro***, etc.), como de tipo *vectorial* (los producidos por las herramientas gráficas de ***Word*** y ***PowerPoint***).

Visual Basic 6.0 admite varios formatos de ficheros gráficos: los ficheros ****.bmp*** y ****.ico*** para los gráficos de tipo *bitmap,* los ficheros ****.wmf*** (*Windows Meta File*) y ****.emf*** (***Enhanced Meta File***) para los gráficos de tipo vectorial y ****.jpg*** (*JPEG o Joint Photographic Experts Group*) y ****.gif*** (*Graphic Interchange Format*). Los ficheros ****.ico*** son ficheros *bitmap* de pequeño tamaño (32 por 32) destinados a contener iconos. Los ficheros JPEG y GIF son formatos gráficos comprimidos que soportan respectivamente color de 24 bit (~16 millones de colores) y 8 bit (256 colores).

Ambos formatos son los utilizados en Internet para mostrar imágenes.

Si se desea insertar ficheros gráficos que estén en otros formatos, habrá que convertirlos previamente a uno de estos formatos usando el programa adecuado.

Continuará.....

Nota de Radacción: El lector puede descargar este capítulo y capítulos anteriores del curso desde la sección ***"Artículos Técnicos"*** en el sitio web de **EduDevices (www.edudevices.com.ar)**