CURSO IBM

PROGRAMAR es fácil

7

Borland Delphi

Programación Visual

Multimedia Ediciones, S.A.

Conceptos de programación (7)

Introducción a Pascal

Desde que en 1971 un profesor de la ciudad suiza de Zurich, Niklaus Wirth, creara el lenguaje de programación Pascal, éste se ha convertido, por lo intuitivo de su sintaxis y legibilidad, en el lenguaje más popular en escuelas y universidades de todo el mundo para la enseñanza de los conceptos de la programación estructurada.

El lenguaje de programación Pascal toma su nombre del célebre matemático francés Blaise Pascal, que se convirtió en el auténtico pionero en la historia del desarrollo de los ordenadores en 1641 cuando, con tan sólo 18 años de edad, diseñó y construyó la primera máquina aritmética, considerada hoy en día por muchos como la primera calculadora mecánica de la historia.

Para conocer el origen del lenguaje Pascal hay que remontarse a la época de los primeros ordenadores, que se programaban mediante código máquina y lenguaje ensamblador, casi siempre usando complicados sistemas de entrada y salida y con una enorme dificultad a la hora de depurar errores.

El hecho de que este método de programación se haga bastante tedioso y muy propenso a provocar fallos en los programas debido a su elevada complejidad, unido a que el código es extremadamente difícil de entender y modificar por parte de usuarios no especializados, hizo que aparecieran los denominados lenguajes de *alto nivel*.

Con ellos se resuelven gran parte de dichos problemas ya que hacen de la programación una especie de *conversación* con el ordenador usando un lenguaje más parecido al que emplean los humanos, en el caso de Pascal, el inglés.

Pascal es uno de esos lenguajes estructurados, entre los que también se encuentran C++, Java, Fortran, Basic o Ada, por nombrar unos pocos.

No obstante, y pese a su mayor facilidad de programación, muy pronto los desarrolladores se dieron cuenta de que los lenguajes tenían una importante dependencia de las nuevas estructuras tipo *goto* ("ir a").

Estas instrucciones mandan al ordenador "saltarse" algunas partes del código para pasar a otras, permitiendo al programa volver a pasos anteriores si fuera necesario.

Por ejemplo, en el lenguaje Basic original, cada línea del código incluía un número identificativo, al que se podía llamar mediante una instrucción *goto*:

```
10 CLS
20 GOTO 40
30 PRINT "ESTA LINEA NO SE
MUESTRA"
40 PRINT "ESTA LINEA SE
MUESTRA"
```

En este ejemplo el programa pasaría a la línea designada como 40 desde la 20, ignorando la número 30.

Este sistema puede parecer muy sencillo en un principio, pero la verdad es que incrementa enormemente la dificultad en el proceso de búsqueda de errores en el código, a la vez que hace más lenta su ejecución.

Además de estas desventajas, otros lenguajes de *alto nivel* como COBOL fueron diseñados con definiciones excesivamente elaboradas, débil soporte de estructuras de datos y falta de flexibilidad, haciendo que los programas fueran tediosos de codificar y muy difíciles de mejorar. En res-

FIG 1. Turbo Pascal ha sido durante años un referente en el mundo de la enseñanza.

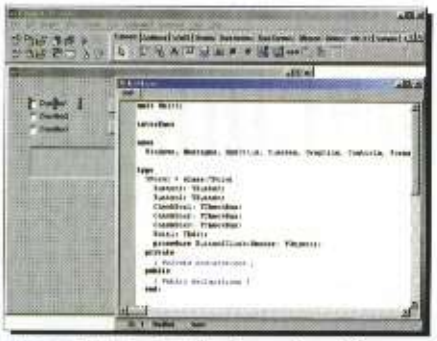

FIG 2. El lenguaje Pascal está muy estructurado, lo que facilita el análisis del código de los programas escritos en este lenguaje.

puesta a todos estos problemas al principio de la década de los 70 el profesor Niklaus Wirth completó el desarrollo del lenguaje de programación Pascal original, basándose en el estilo de bloques estructurados del lenguaje Algol.

Los objetivos principales del lenguaje Pascal original eran:

- Crear un lenguaje con el que se pudiera enseñar programación como una disciplina sistemática mediante conceptos fundamentales reflejados de manera clara y natural por el lenguaje.
- Definir un lenguaje con el que las implementaciones fueran a la vez eficaces y seguras para desarrollar aplicaciones en los ordenadores de la época. No obstante, y pese a su origen académico, el uso comercial del lenguaje Pascal ha excedido el mundo de la enseñanza para pasar al comercial, proporcionando toda clase de tipos de datos enumerados y de registro y definiendo un nuevo estilo en cuanto a la legibilidad de los programas se refiere. Desde el principio, Pascal ha proporcionado una aproximación directa y recursiva a las estructuras de datos, con arrays dentro de arrays, ficheros de registros, registros conteniendo arrays y tablas, etc.

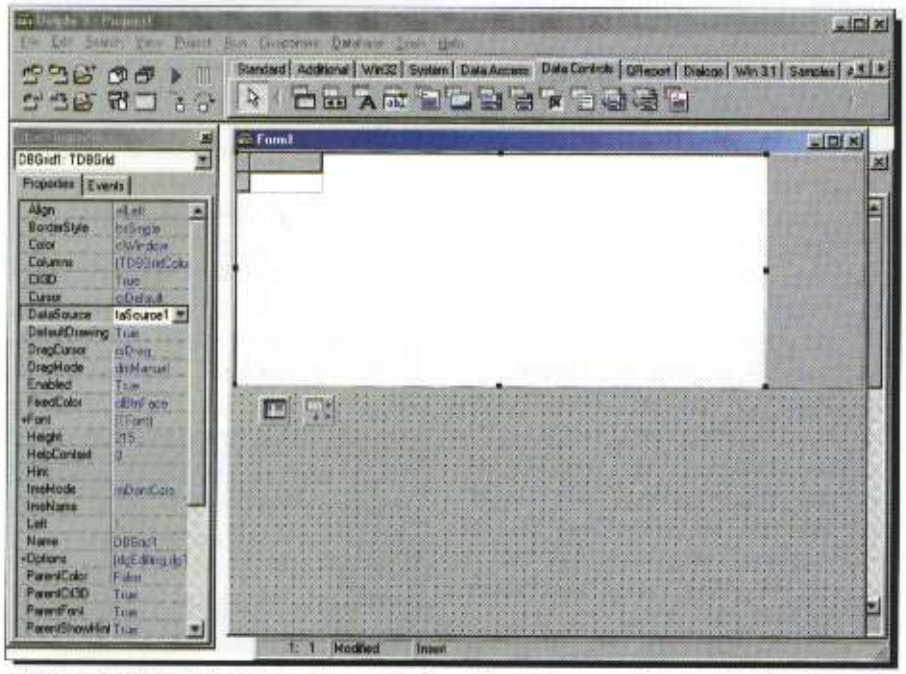

FIG 3. Con la ayuda del programa Borland Delphi se pueden crear visualmente aplicaciones para Windows basadas en Pascal.

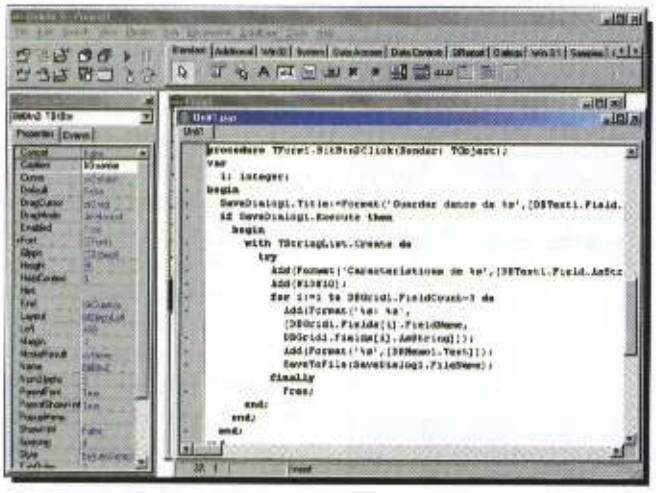

FIG 5. Es conveniente estructurar el código de forma que las estructuras jerárquicas se vean con claridad.

Debido a que la popularidad de Pascal se incrementó notablemente en muy poco tiempo, el comité de la Oficina Internacional de Estándares (ISO) decidió que era necesario crear una definición estándar del lenguaje para garantizar su estabilidad, es la conocida como ISO 7185 Pascal Standard y publicada originalmente en 1983.

Más tarde, en 1990, la necesidad de un desarrollo comercial del lenguaje hizo que evolucionara a la versión ISO 10206 Extended Pascal Standard, hasta que en 1993 el Comité de Estándares de Pascal publicó un informe técnico que introducía las extensiones del lenguaje para la programación orientada a objetos.

Sin embargo, en la actualidad hablar de Pascal es hablar de la compañía Borland International, verdadera difusora del lenguaje gracias a su compilador Turbo Pascal, uno de los más difundidos para la enseñanza a nivel mundial, y que fue la base para el nacimiento de Borland Delphi, una avanzada plataforma visual de desarrollo de aplicaciones basada en Pascal.

Borland Delphi proporciona herramientas de programación visual como las que se pueden encontrar en otros lenguajes similares, pero con las características de Pascal incorporadas.

Delphi está basado en las extensiones Object Pascal que se pueden encontrar en Borland Pascal desde su versión 5.5.

ESTRUCTURA DE PASCAL

El lenguaje Pascal está fuertemente tipificado y estructurado.

El *tipo* de una variable en Pascal consiste en su definición semántica y un rango de valores determinados, y puede ser expresado por su nombre de tipo, por un rango de valor explícito, o una combinación de ambos.

El rango de valores para un tipo viene definido por omisión por el propio lenguaje para los tipos ya predefinidos en Pascal, o por el programador si se

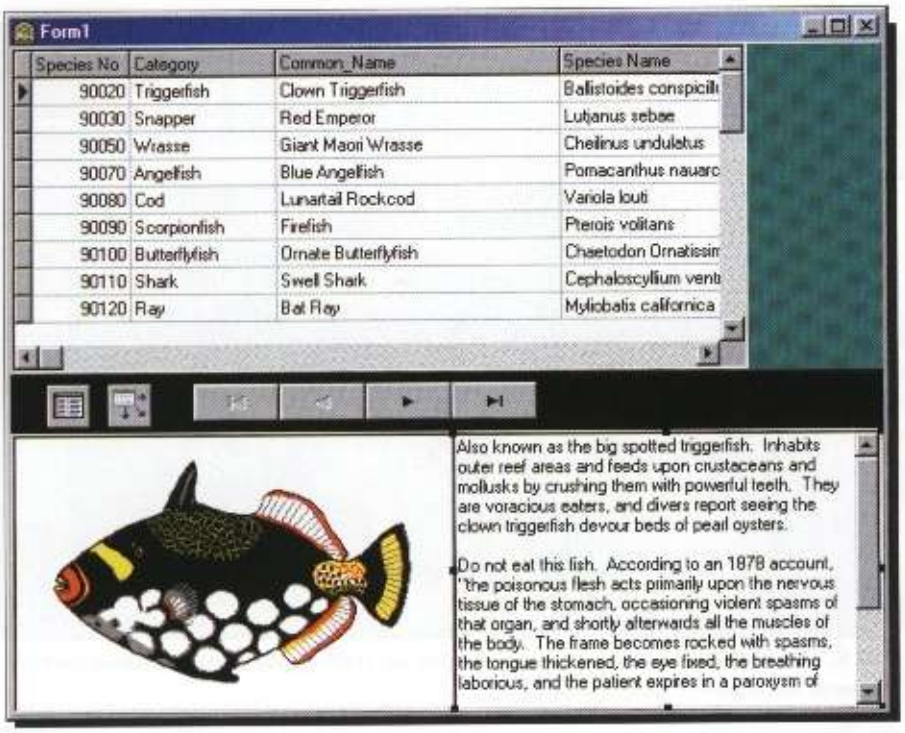

FIG 4. Las aplicaciones pueden incluir cualquier clase de objeto gráfico que imaginemos.

desean incluir nuevos tipos definidos por el usuario.

Estos últimos son tipos de datos únicos definidos mediante la sección de declaración TYPE de Pascal, y pueden consistir en tipos enumerados, arrays, registros, punteros, tablas y otros muchos más, así como combinaciones de diferentes tipos.

Cuando las variables se definen como de un tipo determinado, el compilador puede asumir que la variable será usada con ese tipo durante toda su vida (sin importar si se trata de variables globales al programa o locales a una función o procedimiento).

Así, este tratamiento consistente de las variables hace que el código sea mucho más sencillo de mantener.

El compilador detecta los errores de inconsistencia de tipos en tiempo de compilación, depurando muchos errores y reduciendo la necesidad de analizar el código mediante un *debugger* externo.

Las variables se declaran en Pascal mediante la sección de declaración VAR. En esta sección el programador tiene la capacidad de declarar cadenas, enteros, números reales y valores lógicos (por citar algunos), a la vez que se le permiten declarar variables como registros o *records* u otros tipos definidos por el programador. Una variable definida como RECORD permite a otra variable hacer un seguimiento de ciertos componentes de datos. Veamos un ejemplo de definición de tipos en Pascal:

```
Type Tipo_Empleado =
(Horario,Salario);
  Registro= RECORD
    nombre: packed array
[1..30] of char;
    salario: real;
    END;

Var
  indice: integer;
  sino:boolean;
  infor: FILE OF Registro;
```

Pascal también soporta la *recursividad*, una potente herramienta de programación que permite a una función o a un procedimiento dentro de un programa realizar llamadas a sí mismo.

Esto facilita unas técnicas de codificación muy eficientes y elegantes, eliminando la necesidad de tediosos bucles. Un ejemplo típico de recursividad es la solución del juego denominado "Torres de Hanoi". Éste consiste en cambiar una pila de discos de tamaños diferentes, de mayor a menor, y moverlos de una estaca donde están insertados a otra adyacente, con las únicas reglas de que un disco siempre debe ser colocado sobre otro mayor, y que solamente se puede mover un disco cada vez. Observe el siguiente código:

```
Program TorresHanoi
(entrada,salida);

  Var
  discos: integer;

  Procedure Hanoi (origen,
temporal, destino: char; n:
integer);

  Begin
  If n > 0 then
    Begin
    Hanoi (origen, destino,
temporal, n - 1);
    writeln ('Mover disco ',n:1,'
desde estaca ', origen,' a
estaca ',destino);
    Hanoi (temporal, origen,
destino, n - 1);
    End;
  End;


Begin
  Write ('Introduzca número de
discos: ');
  Readln (discos);
  writeln ('Solución:');
  Hanoi ('A','B','C',discos);
End.
```

No se preocupe si no entiende el código, ya que lo único que se pretende es que vea la manera en que el procedimiento Hanoi se llama a sí mismo (observe que la llamada al procedimiento se encuentra dentro del propio procedimiento).

La solución a las "Torres de Hanoi" implica mover todos los discos menos uno de una estaca a otra, repetidamente, hasta que toda la pila haya sido movida.

La elegancia de la recursividad se ilustra en la solución al problema, en el que no hay ninguna clase de bucle que contenga instrucciones "ir a" o comparaciones lógicas de álgebra de Boole.

OTRAS CARACTERÍSTICAS DE PASCAL

Como ya hemos señalado anteriormente, Pascal elimina las órdenes *goto* para incluir estructuras más avanzadas. Entre éstas se encuentran los bucles REPEAT/UNTIL, WHILE/DO y FOR. Además, una característica importante en Pascal es la inclusión de la estructura CASE, en la que se pueden declarar diferentes destinos y acciones en casos diferentes (por ejemplo, para entradas desde teclado o desde una lista).

Otra característica importante de este lenguaje es el uso de las palabras *Begin* y *End* (Inicio y Fin) para delimitar bloques de código dentro de una cláusula, obligando al programador a crear bloques y estructuras claras y definidas, a la vez que otras útiles características lingüísticas. De esta forma, el lenguaje Pascal facilita la producción de un código correcto, seguro y de fácil mantenimiento. Cualquier lenguaje de programación puede ser comentado para mayor legibilidad, pero Pascal, gracias a la naturaleza de su sintaxis y arquitectura, hace que los programadores adquieran "buenas costumbres" a la hora de escribir código, lo que, frente a otros lenguajes, aumenta su legibilidad con un menor número de comentarios. Es importante señalar esto, ya que cuando se deben crear grandes proyectos con miles de líneas de código en archivos diferentes, una buena legibilidad y estructuración pueden ser decisivas, sobre todo si otras personas deben leer códigos fuente que no han escrito. También hemos comentado que Pascal se estructura en *bloques*. Un bloque es una secuencia de declaraciones, un *Begin*, una secuencia que establece las acciones que se desarrollarán sobre las estructuras de datos descritas en las declaraciones, y un *End*.

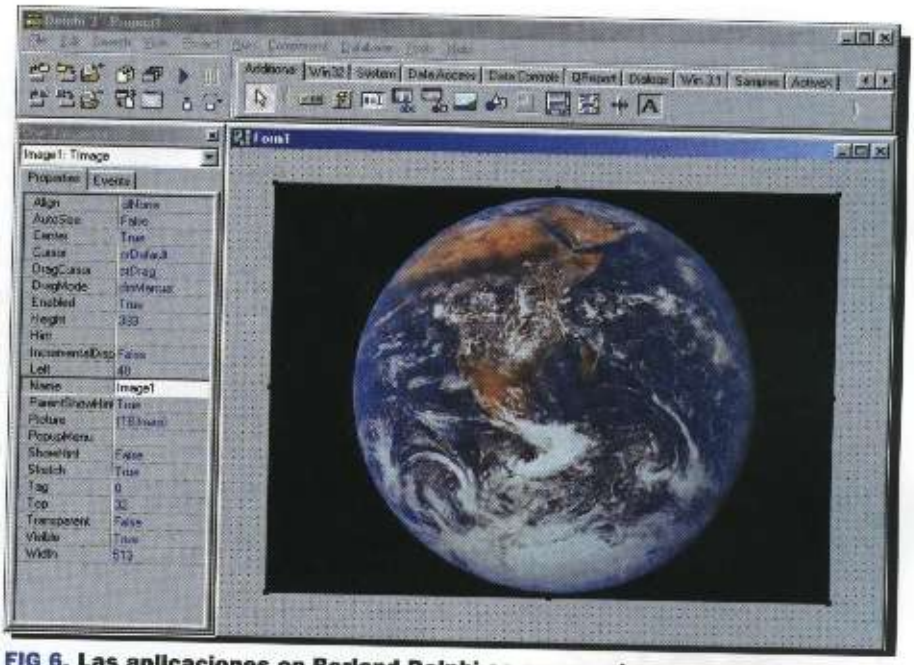

FIG 6. Las aplicaciones en Borland Delphi se crean sobre una ventana de formulario totalmente configurable.

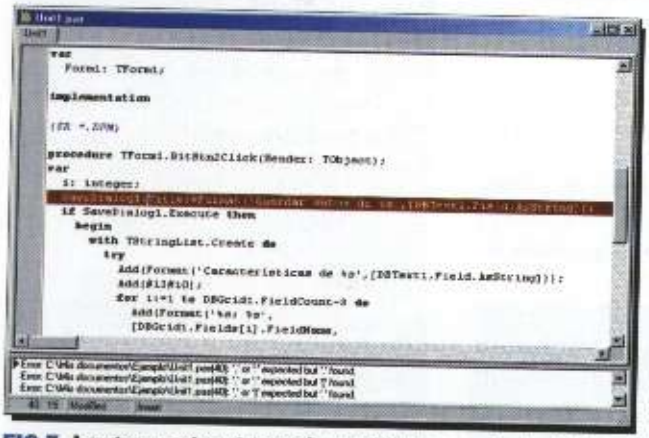

FIG 7. La gran estructuración del texto es vital a la hora de depurar los programas.

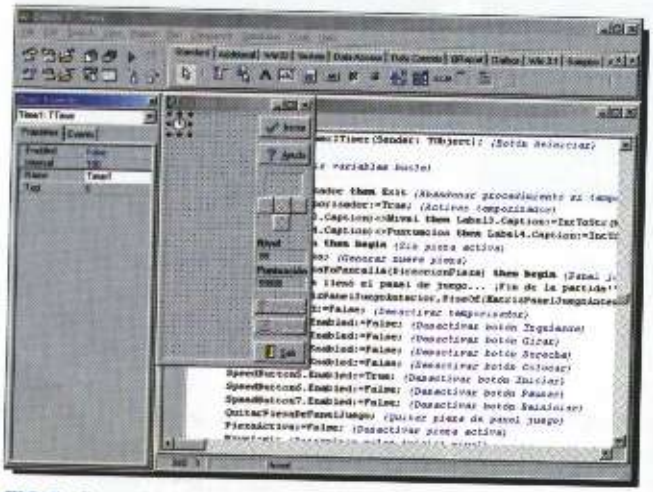

FIG 8. Con el lenguaje Pascal se pueden crear toda clase de aplicaciones, incluso juegos.

Un programa en lenguaje Pascal comienza siempre por el encabezado *Program* seguido por un bloque.

Dentro de ese bloque principal del programa pueden existir subprogramas, cada uno de los cuales está definido por su propio encabezado y declarado en un bloque. Dentro de cada bloque pueden existir otros bloques añadidos, *locales* al bloque padre. En esencia, se puede afirmar que Pascal es una construcción jerárquica de bloques, de ahí que se diga que Pascal es un lenguaje de programación estructurado por bloques.

Todos los valores de los datos declarados al principio de un bloque son accesibles al código escrito dentro del bloque, incluyendo otros bloques *hijos*, pero no a otros. Así, la utilidad de un lenguaje estructurado en bloques como Pascal no descansa solamente en su modularización, sino también en la protección de los datos que son exclusivos a unos bloques determinados dentro de un programa, por lo que se evita sean modificados por otros módulos, mejorando así enormemente la integridad del proyecto.

Soluciones prácticas (7)

TechSmith SnagIt: Los filtros

Hemos visto ya algunas de las opciones de configuración más importantes de SnagIt. En esta unidad profundizaremos aún más en el manejo de la aplicación comentando la elección y aplicación de filtros. Estas nuevas opciones le permitirán mejorar sus esquemas de ayuda.

Existen programas que parecen hacer todo lo posible para resultar tan complicados como extravagantes y en cambio otros, como TechSmith SnagIt, son tan sencillos que incluso un niño sabría sacarles provecho. Eso sí, no debemos perder de vista el hecho de que bajo su aparente sencillez, esta aplicación esconde muchas sorpresas. Una de ellas es su capacidad para aplicar filtros en tiempo real, es decir, en el momento de la captura. Como verá a continuación, la cantidad de tiempo que puede ahorrar el uso de esta particularidad de la aplicación hace que merezca la pena detenerse un momento para verla más de cerca.

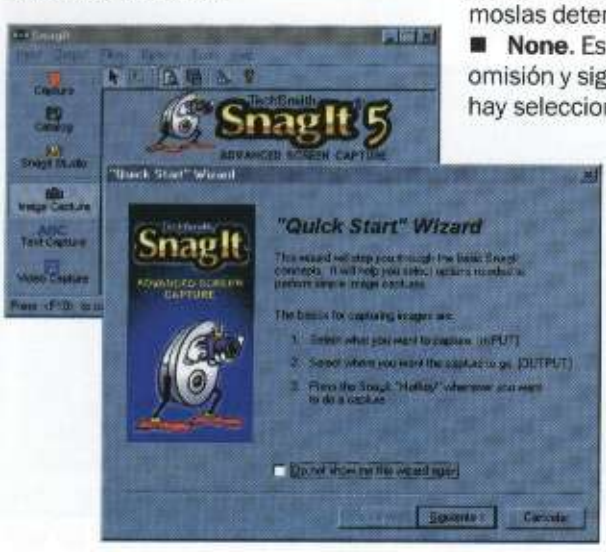

FIG. 1 Sencillo pero potente. Así es TechSmith SnagIt.

COLORES

Los filtros de TechSmith SnagIt se recogen en el menú **Filters** y son, en esencia, procesos especiales de la imagen tales como añadir una marca de agua o realizar una conversión de color. Estos filtros se pueden aplicar tanto en el momento de realizar la captura como a posteriori, por lo que su aplicación no está restringida en absoluto. La primera opción que encontrará al abrir el menú **Filters** es **Color Conversion**. Bajo este nombre se engloban todos los filtros que tienen que ver con la conversión de colores. Existen distintas posibilidades dependiendo de a qué se dediquen las capturas de pantalla; veámoslas detenidamente:

- **None**. Es la opción seleccionada por omisión y significa que actualmente no hay seleccionado ningún filtro.
- **Monocrome**. Convierte la imagen a blanco y negro. Al seleccionar esta opción, aparecerá un cuadro de diálogo en el que podrá configurar la relación entre la cantidad de blanco y de negro de la imagen final para poder determinar si será más clara o más oscura.
- **Halftone**: Es una opción similar a la anterior, pero la imagen monocroma estará suavizada.
- **Grayscale**: la imagen se convierte a escala de grises (256 tonalidades de gris). Este formato de imagen es ideal para imprimir un documento en una impresora de blanco y negro, e incluso para editar un manual minimizando el coste de producción.
- **Custom Color Resolution**: Escogiendo esta opción podrá configurar el número de colores que debe poseer la imagen. Existen opciones que van desde dos colores (1 bit) hasta Color verdadero (de 24 ó 32 bits).

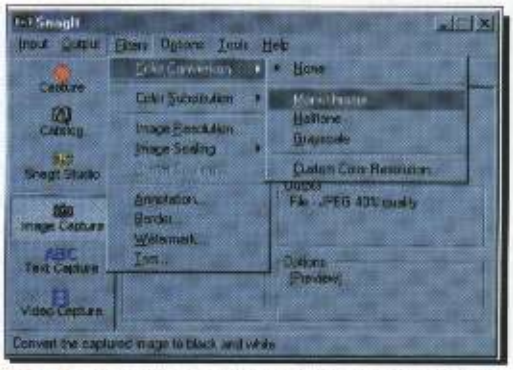

FIG. 2 En este menú se recogen los filtros de la aplicación.

Continuando con las opciones del menú **Filters**, **Color Subtitution** ofrece dos opciones, **Invert Colors** y **Custom Color Subtitution**, que permiten, respectivamente, invertir los colores y susti-

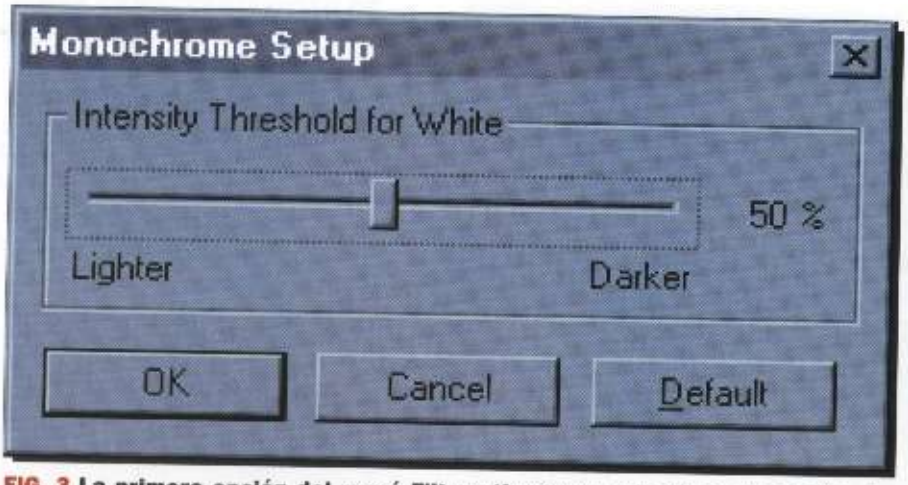

FIG. 3 La primera opción del menú Filters tiene que ver con la aplicación de filtros relacionados con el cambio del número de colores de la imagen.

tuir unas tonalidades por otras. El ámbito de uso de estas dos opciones es muy restringido, por lo que no nos extenderemos más.

RESOLUCIÓN Y TAMAÑO

Image Resolution muestra un cuadro de diálogo en el que se puede configurar la resolución en píxeles por pulgada que poseerá la imagen. Por omisión se encuentra activada la opción **Auto Configure** y la resolución elegida es de **96** píxeles por pulgada. **Image Scaling** es una opción realmente interesante, siempre que sea necesario reducir el tamaño de las capturas para adaptarlas al medio de difusión, como ocurriría, por ejemplo, en el caso de un curso HTML. Si despliega las subopciones de **Image Scaling** verá que existen muchos porcentajes distintos.

Los que son inferiores al 100% reducen el tamaño de la imagen en el porcentaje indicado, mientras que los que superan el 100% la amplían en el grado escogido.

Con **Custom Scale** podrá definir el porcentaje de tamaño. Si escoge valores distintos en las opciones **Width** (ancho) y **Height** (alto) del apartado **Scale by percentage of original** (*variar el tamaño por porcentaje del original*), distorsionará la imagen. Por otra parte, puede cambiar el tamaño de las capturas de pantalla para que se adapten a tamaños prefijados con las opciones del apartado **Scale by pixeles**. Por último, asegúrese de que está activada la opción **Keep aspect ratio** para mantener la proporción original entre la altura y la anchura de la captura de pantalla. Siempre que escoja una de estas subopciones, asegúrese de que está activada la opción **Smooth Scaling**. De esta forma se suavizará la imagen para que no se aprecie en demasía el detrimento en la calidad que conlleva cualquier cambio en su tamaño (sobre todo al ampliarla). **Printer Scaling** tiene que ver con la configuración de la impresora, y su objetivo es permitir que el usuario pueda elegir el tamaño de la captura y su posición en el momento de imprimir. Tenga en cuenta que sólo podrá acceder a esta opción si ha seleccionado la impresora como dispositivo de salida de las capturas de pantalla que vaya a realizar.

RESOLUCIÓN

La resolución que propone TechSmith SnagIt por omisión es más que suficiente para cualquier tarea que tenga pensada, desde crear una ayuda en formato HTML hasta un manual impreso, siempre y cuando no coloque las imágenes en el documento exageradamente grandes. La resolución de la pantalla es de 72 píxeles por pulgada (ppp) y cualquier cantidad superior hará que TechSmith SnagIt "invente" los puntos que faltan, incluyendo esos 96 ppp que ofrece por omisión. En definitiva, puede usar 96 ppp, pero no incremente nunca esta cifra excepto en circunstancias muy especiales.

INFORMACIÓN SOBRE LA IMAGEN

Annotation es una opción bastante útil, ya que permite realizar anotaciones en las capturas. Estas anotaciones son, como verá en el cuadro de diálogo de configuración, textos que pueden indicar un mensaje personalizado, una fecha, etc.

Si activa la opción **Enable Caption** podrá definir que aparezca un texto determinado en la imagen. Haciendo clic sobre **Options** accederá a un cuadro de diálogo en el que podrá configurar el aspecto del texto (con sombra, con línea exterior, el color del texto y de su filete, hacer que el fondo sea transparente, etc.); también podrá determinar su posición haciendo clic en cualquiera de los cuadrados que puede ver en el apartado **Position**. Además, si activa la opción **Prompt for a caption**, podrá determinar durante el proceso de captura cuál será el texto a escribir, algo muy útil, por ejemplo, para realizar comentarios de una interfaz para una futura corrección. En **System Caption** puede definir que se escriban anotaciones tales como una fecha o el nom-

FIG. 4 Con la opción Grayscale puede convertir las imágenes que capture a escala de grises.

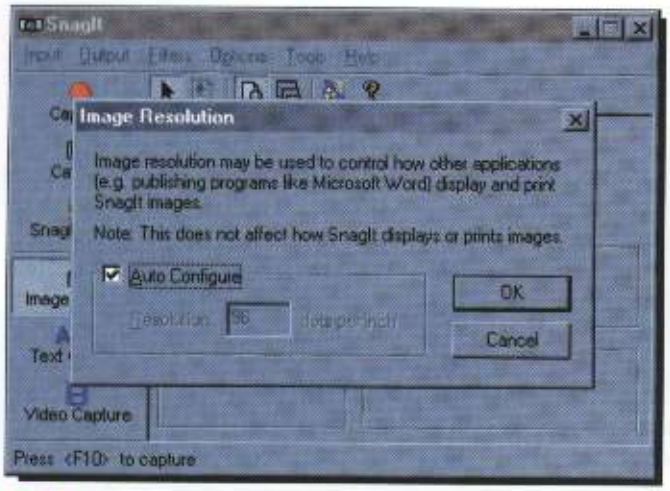

FIG. 5 La resolución por omisión que configura TechSmith SnagIt automáticamente es de 96 píxeles por pulgada.

FIG. 6 Con las opciones de Image Scaling podrá cambiar en tiempo real el tamaño de las imágenes según el porcentaje escogido.

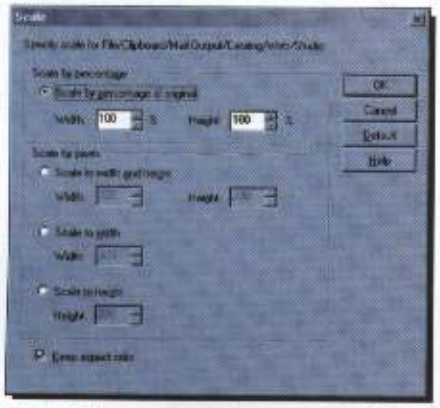

FIG. 7 Siempre conviene que está activada la opción Keep aspect ratio para que se guarde la relación entre la altura y la anchura de la captura.

bre del ordenador, por ejemplo. Igualmente, podrá configurar el aspecto del texto, así como del formato de la fecha. **Border** no es una opción tan útil como la anterior, pero tampoco conviene perderla de vista. En definitiva, permite dibujar bordes alrededor de la imagen capturada. Podrá configurar el marco en lo referente a su aspecto. **Watermark** significa, literalmente, *marca de agua*, y se refiere a estampar una marca en la imagen. Ésta es ideal, por ejemplo, para salvaguardar la propiedad intelectual de las capturas que realice. Para poder aprovechar esta función de TechSmith SnagIt, debe activar la opción **Enable Watermark**. En **Image Path** puede definir qué archivo de imagen se va a usar para la marca de agua. Para ello basta con hacer clic sobre el botón que muestra los tres puntos suspensivos (...) y seleccionar la imagen usando el procedimiento habitual en este tipo de cuadros de diálogo. Recuerde que puede realizar una previsualización de la marca de agua en cualquier momento haciendo clic sobre el botón **Preview**. Verá que ha aparecido una pequeña ventana en la que se muestra un degradado de tonos de grises; en un extremo, aparecerá la marca de agua a pequeño tamaño. De hecho, la marca de agua parece estar debajo del degradado y no encima. Esto se debe a que aparece seleccionada la opción **Underlay** del apartado **Display Effects** (*efectos de visualización*). Si hace clic sobre **Overlay**, la imagen se superpondrá. Puede activar si lo desea la opción **Use Transparent Color** para definir el color que será transparente en la imagen. Si desea que la marca de agua esté detrás de la imagen, puede activar la opción **Emboss Image**; activándola conseguirá realzar la marca de agua. El cuadro de lista desplegable **Direction** indica dos cosas: el efecto de sombra y la dirección donde se encuentra la luz que sirve para calcular el efecto. **Depth** (*profundidad*) determina el grado del efecto aplicado. Pruebe con distintas variaciones para ver cómo funciona este bonito efecto de marca de agua. En el último apartado de este cuadro de diálogo, **Image Positioning**, encontrará todas las opciones referentes

FIG. 8 Es importante activar la opción Smooth Scaling para conseguir que se suavicen las imágenes al variar su tamaño.

PREVISUALIZACIÓN

En el caso de las marcas de agua, conseguir una buena visualización puede llevar algún tiempo, y no porque TechSmith SnagIt lo ponga difícil ni mucho menos, sino porque existen varias opciones que pueden variar tanto el aspecto como la colocación de la marca, pasando por su tamaño, y debido a que es difícil calcular "a ojo" el aspecto final que poseerá.
Es por ello que existe el botón Preview, con el que podrá previsualizar la marca; también existe una opción oculta mediante la cual podrá ver en tiempo real, en esta misma ventana, los cambios que produce la elección de una u otra opción en la marca de agua. No cierre la ventana de previsualización mientras trabaja en la configuración; ahorrará mucho tiempo.

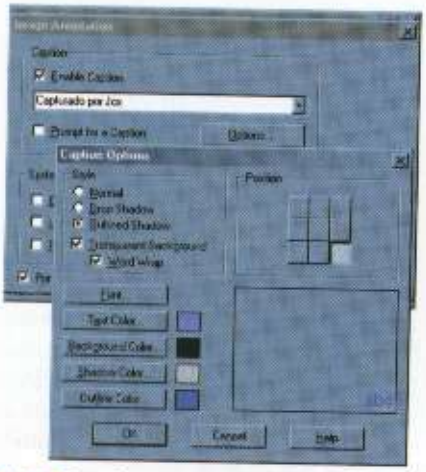

FIG. 9 Con las opciones de este cuadro de diálogo puede determinar el aspecto que tendrá el texto de la anotación.

FIG. 10 TechSmith SnagIt permite estampar una marca de agua en las capturas de pantalla.

al tamaño y la posición de la marca de agua. En la parte izquierda, por medio de ese conjunto de nueve cuadraditos, puede escoger en qué zona de la imagen se colocará la marca. Con **Image**

FIG. 11 El botón Preview le permite obtener una vista previa de la marca de agua elegida.

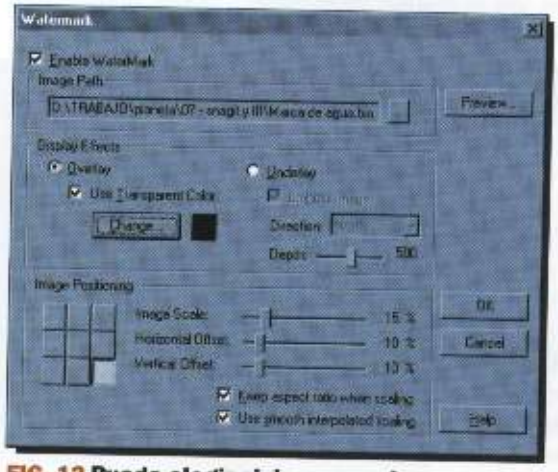

FIG. 12 Puede elegir si desea que la marca de agua se superponga a la imagen o que, en cambio, se encuentre detrás de ella.

Scale, **Horizontal Offset** y **Vertical Offset** podrá, respectivamente, determinar el grado de reducción del tamaño de la marca, y el desplazamiento horizontal y vertical de la marca en la imagen. Por último, existen dos opciones más encaminadas a hacerle la vida más sencilla. La primera es **Keep aspecto ratio when scaling**, con la que conseguirá que TechSmith SnagIt amplíe y reduzca el tamaño de la marca de agua de forma proporcional, y la segunda es **Use smooth interpolated scaling**, lo que hará que la aplicación suavice la imagen al variar su tamaño para que no pierda demasiada calidad. Sólo nos resta ya comentar un filtro de los incluidos en el menú **Filters**, y se trata de **Trim** (*recortar*). Activando la opción **Enable Trimming** podrá especificar en la parte inferior del cuadro de diálogo cuántos pixeles se deben recortar por arriba (**Top**), por la derecha (**Right**), por abajo (**Bottom**) o por la izquierda (**Left**) de la captura de pantalla. Activando la opción **Automatic trimming** indicará a TechSmith SnagIt que debe recortar cualquier área "en blanco", es decir, de color uniforme, que rodee el objeto a capturar.

EN DEFINITIVA...

Como ha podido ver a lo largo de las tres unidades que hemos dedicado a TechSmith SnagIt, de este programa se puede sacar más partido del que parece a simple vista. Con una sencilla reconfiguración de parámetros, se pueden realizar las más diversas capturas, con la gran comodidad que ello supone. Ahora tan sólo debe practicar un poco su manejo antes de comenzar a ahorrar horas de tedioso trabajo. Antes de terminar, queremos apuntar que en el CD-ROM que acompaña a esta unidad, concretamente en la carpeta **\Utilidades\SnagIt\Capturas**, encontrará capturas de ejemplo que le ayudarán a comprobar los efectos de la elección de alguno de los filtros que hemos tratado aquí.

FIG. 13 Puede colocar la marca de agua donde le plazca, al tamaño que quiera.

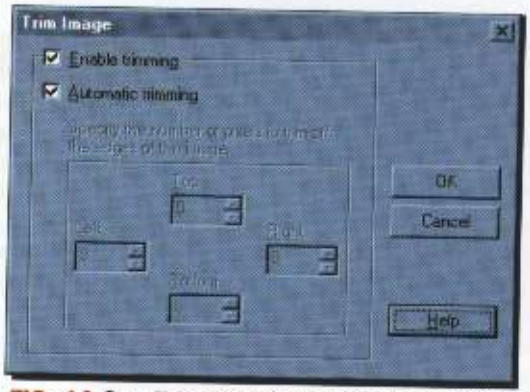

FIG. 14 Con Trim podrá recortar la captura de pantalla como desee. Incluso dispone de una opción de recorte automático.

Borland Delphi (1)

Conceptos básicos

Borland Delphi es un entorno visual de programación orientado a objetos creado para el desarrollo rápido de aplicaciones de propósito general. Adaptado a las distintas plataformas Windows, la versión que se entrega con esta obra permite crear aplicaciones eficientes reduciendo al mínimo la necesidad de codificación manual.

En las próximas unidades hablaremos extensamente sobre Borland Delphi, con la intención de ofrecer una visión general de esta potente plataforma de programación visual orientada a objetos, con ayuda de la cual se han escrito ya muchas aplicaciones destinadas a ámbitos muy diversos.

Su indudable interés para el programador se debe, de una parte, al empleo de un lenguaje de programación de tanto prestigio como Pascal, que siempre se cita como modelo de programación estructurada, y de otra a la facilidad de manejo de su entorno gráfico, mediante el cual es posible generar automáticamente la mayor parte del código necesario de forma automática, a partir de las especificaciones introducidas por medio de cuadros de diálogo y la asignación directa de valores a las distintas propiedades de los objetos.

Por otro lado y para no dejar de lado la eminente vocación práctica de este curso, desarrollaremos distintas aplicaciones que sirvan de apoyo a la exposición de conocimientos estrictamente teóricos sobre el entorno de programación. La que proponemos en esta primera entrega nos permitirá gestionar una base de datos, contando con botones de navegación para el avance y retroceso por registro, y acceso al comienzo y final de la base de datos, con posibilidad de generar archivos de texto con información extraída de un determinado registro.

CARACTERÍSTICAS

Borland Delphi cuenta con una serie de características, que deben tenerse en cuenta a la hora de decidir la herramienta idónea para la creación de aplicaciones, dentro de la oferta actual de productos. Reseñaremos brevemente las que pueden resultar más interesantes a los programadores y desarrolladores de aplicaciones:

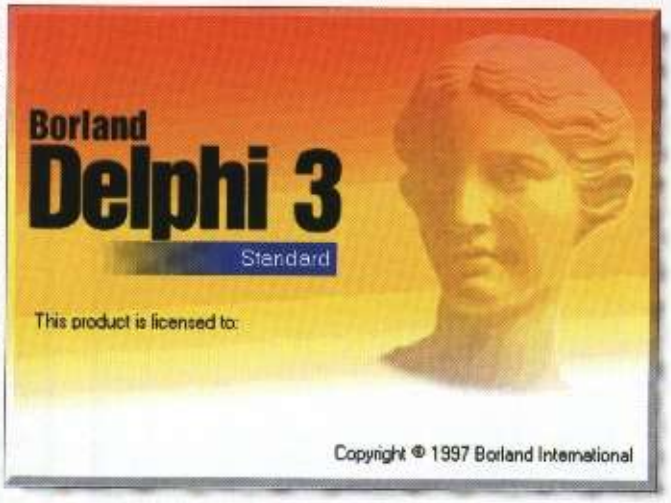

FIG. 1 Borland Delphi es un entorno visual de programación orientada a objetos concebido para el desarrollo rápido de aplicaciones de propósito general.

- El entorno de desarrollo integrado aglutina las aplicaciones auxiliares de depuración y mantenimiento necesarias para la puesta a punto de los programas, de modo que no es necesario abandonar el entorno para efectuar tarea alguna con la ayuda de programas externos.
- Su biblioteca de herramientas de diseño y componentes reutilizables facilita el desarrollo rápido mediante plantillas de aplicaciones, fichas y expertos de programación, con posibilidad de ampliación sobre la base de las aportaciones personales de cada usuario.
- La automatización de tareas repetitivas permite aumentar la productividad. Basándose en este principio, Borland Delphi permite arrastrar fácilmente con el ratón componentes sobre las fichas de trabajo desde la paleta **Component** (Componentes), permitiendo así crear aplicaciones con gran facilidad y rapidez, y reduciendo al mínimo la necesidad de codificación manual.
- Las modificaciones introducidas en cualquiera de los entornos, gráfico y de texto, tienen efecto inmediato en el otro. Así, por ejemplo, es posible tanto editar propiedades de objetos para que

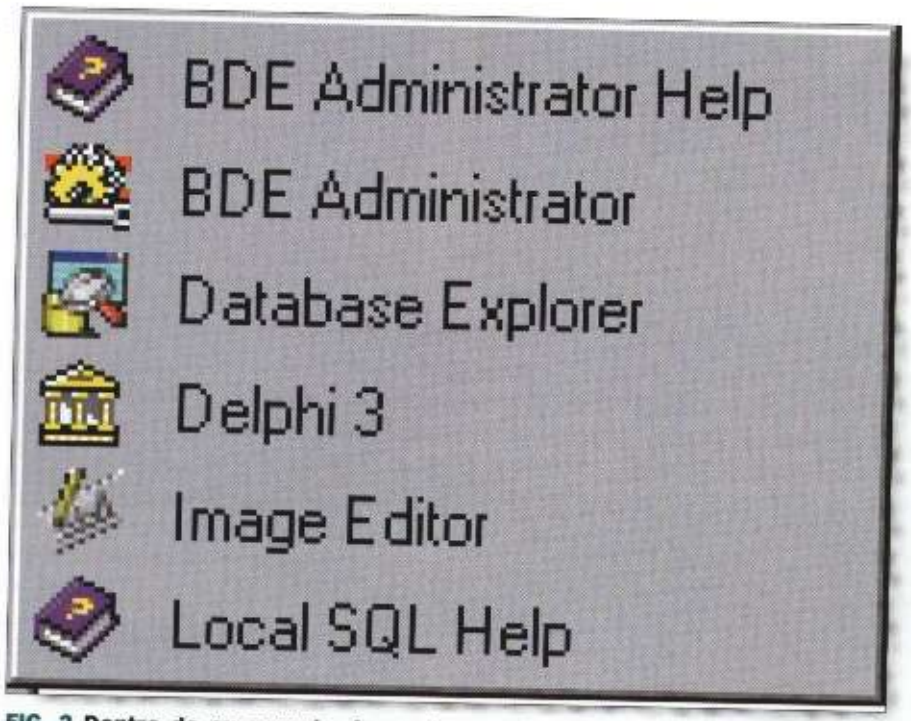

FIG. 2 Dentro de su carpeta de programas, la opción de menú Delphi 3 da acceso al entorno de programación.

se efectúen modificaciones automáticas en el código, como al revés, pudiendo emplearse en este último caso cualquier editor, por tratarse de archivos de texto convencionales.

■ El compilador genera archivos directamente ejecutables (EXE) compactos y autónomos, de modo que no es imprescindible depender de archivos de biblioteca dinámica (DLL - *Dynamic Linking Library*) en el momento de la ejecución de los programas, con la ventaja que ello supone tanto en cuanto a instalación de aplicaciones como en lo referente a posibles conflictos con otros programas instalados, cuestión esta que trae con frecuencia de cabeza a los programadores.

■ La herramienta de gestión de bases de datos de Borland Delphi permite crear aplicaciones profesionales cliente / servidor con un mínimo esfuerzo, con posibilidad de visualización directa de datos durante la fase diseño, lo cual facilita la puesta a punto y reduce el tiempo necesario para las pruebas de ejecución.

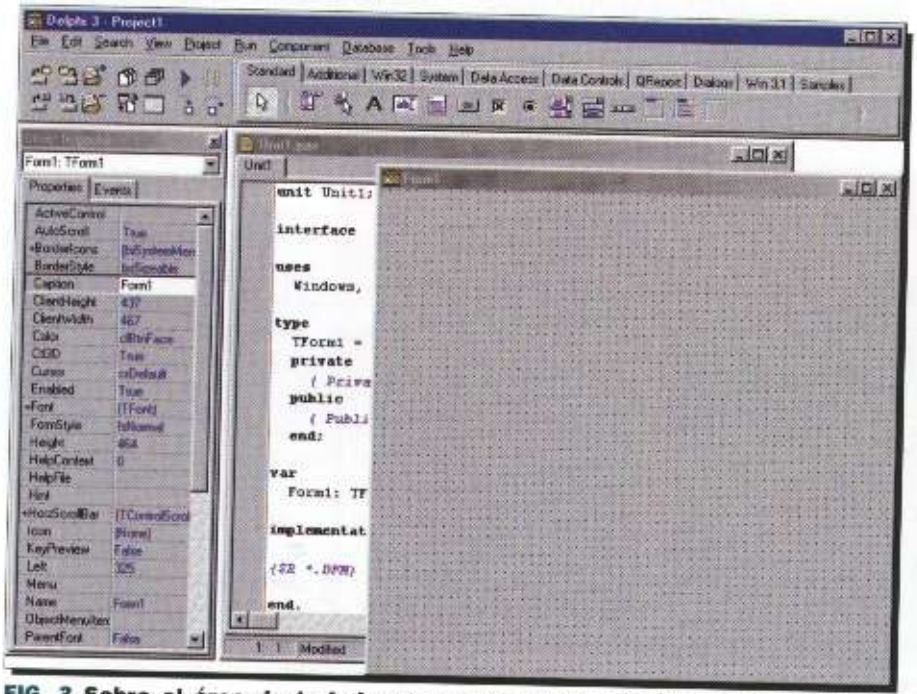

FIG. 3 Sobre el área de trabajo se muestran varias ventanas, ocupando la interfaz de usuario el margen superior.

NOVEDADES DE BORLAND DELPHI

Esta versión de Borland Delphi aporta interesantes novedades frente a versiones anteriores, sobre todo en lo referente al entorno cliente / servidor e implementación de aplicaciones concebidas para Intranets o Internet:

■ Las aplicaciones pueden utilizar bibliotecas de enlace dinámico denominadas paquetes, mediante las cuales es posible compartir código entre distintas aplicaciones y reducir el tamaño de los ejecutables.

■ El nuevo componente TClientDataSet permite crear objetos de conjuntos de datos, para su utilización tanto en aplicaciones locales, como para la parte cliente cuando se trabaja con servidores remotos mediante TRemoteServer.

■ Borland Delphi simplifica la creación de aplicaciones para Internet / Intranet, mediante componentes específicos que gestionan la comunicación con el servidor Web por protocolo HTTP, creando automáticamente el código HTML para la representación de tablas de datos.

■ Los asistentes permiten crear componentes estándar ActiveX a partir de controles VCL, utilizables con otras herramientas de desarrollo, como C++, Java, Visual Basic y PowerBuilder; para su despliegue Web posterior.

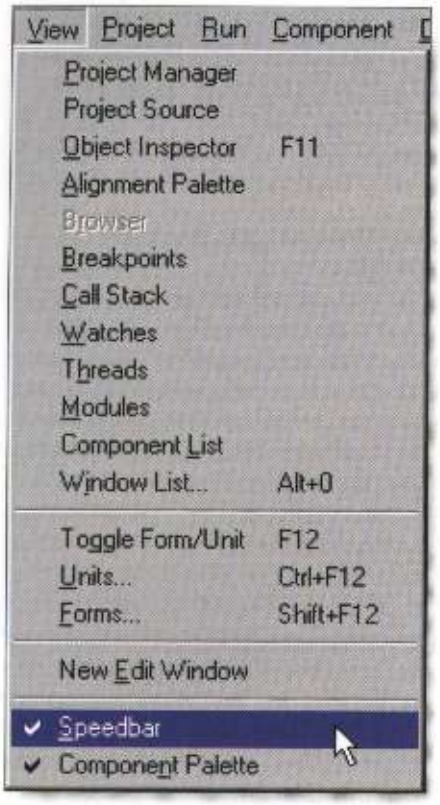

FIG. 4 Al mostrase la barra de botones Speedbar se reduce el espacio disponible para la paleta Component.

■ El motor de bases de datos de Borland (BDE - *Borland Database Engine*), que se instala como módulo independiente, permite acceder a bases de datos locales en los formatos más extendidos: Paradox, FoxPro, dBASE e InterBase Server. Permite acceder también a servidores SQL, como InterBase, Informix, Oracle, Sybase, Sybase System 10, DB2 y Microsoft SQL, mediante *SQL Links*.

ELEMENTOS BÁSICOS DEL ENTORNO

El proceso de instalación de Borland Delphi crea la correspondiente carpeta de programas, cuya opción de menú **Delphi 3** da acceso al entorno de programación. Junto con la primera entrega de este curso se instalan únicamente los componentes básicos, razón por la cual no se mostrarán inicialmente todos los componentes del grupo de programas, aunque la operatividad del entorno queda, sin embargo, completamente garantizada.

El margen superior del área de trabajo lo ocupa la interfaz de usuario, que está compuesta por la barra de título de la aplicación con los botones de control habituales, la barra de menús desplegables, la barra **Speedbar** (rápida) y la paleta **Component**. Estos dos últimos componentes son elementos, pudiéndose mostrar o no dependiendo de lo que indique el usuario por medio de las opciones del menú desplegable **View** (Ver). Así, por ejemplo, prescindir de la **Speedbar** permite mostrar completamente la paleta **Component** en pantallas con una resolución de 800 x 600 píxeles.

FIG. 5 Pueden elegirse también componentes mediante la lista alfabética del cuadro de diálogo Component.

La paleta **Component** está integrada por una serie de fichas en las cuales se encuadran los distintos objetos por tipos, seleccionables mediante pestañas, sirviendo los botones de dirección del margen derecho para recorrer las fichas no visibles, cuando la resolución de la pantalla no lo permite. Bajo las pestañas se muestra el repertorio de componentes de la ficha activa (en forma de botones), disponiéndose nuevamente de flechas de desplazamiento para mostrar posibles componentes ocultos, situadas en este caso a ambos márgenes.

Para colocar componentes sobre las fichas basta con hacer clic sobre los mismos y, seguidamente, sobre el lugar de la ficha adecuado, con lo cual adoptan el tamaño considerado por omisión; o bien arrastrar directamente con el ratón para definir un área específica sobre la ficha de trabajo. Alternativamente pueden elegirse también componentes mediante una lista alfabética, que se muestra con la secuencia de menús **View | Component List** (Ver | Lista de Componentes), al hacer clic sobre los mismos y pulsar el botón **Add to form**. (añadir formulario)

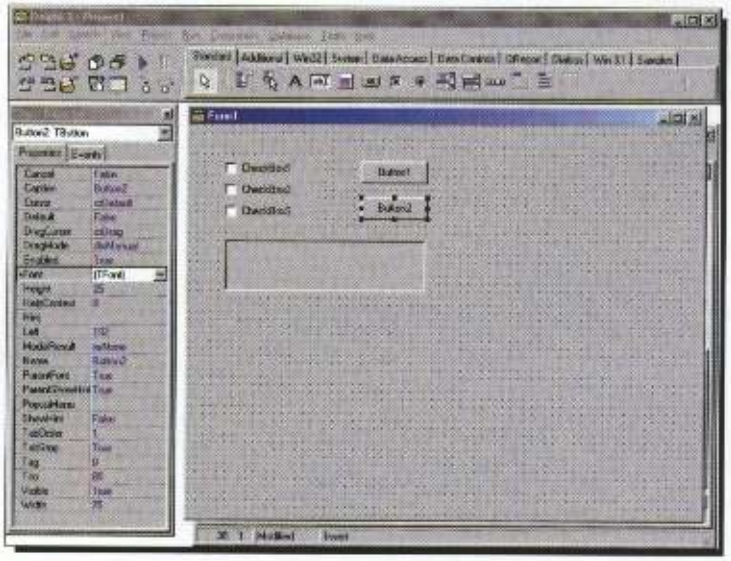

FIG. 6 Las propiedades de los componentes se establecen usualmente mediante el Object Inspector, generándose automáticamente el código necesario.

El margen izquierdo del área de trabajo muestra el **Object Inspector** (Inspector de objetos), herramienta fundamental para editar las propiedades de los objetos y elegir manejadores de sucesos, a la cual se puede acceder rápidamente desde cualquier ventana al pulsar **F11**, o

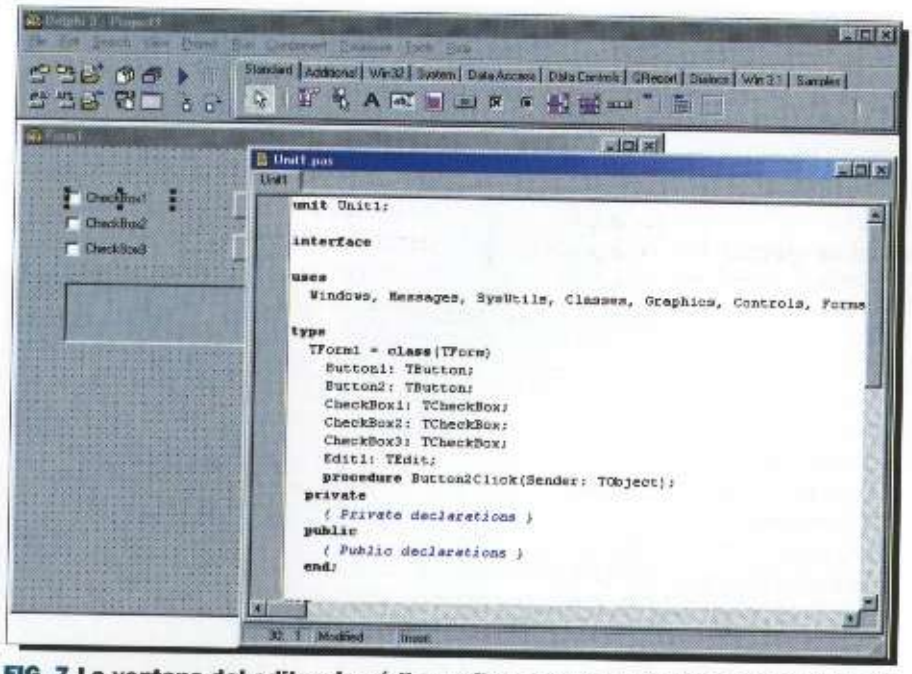

FIG. 7 La ventana del editor de código refleja las especificaciones del Object Inspector y puede personalizarse si se desea.

bien mediante la opción habilitada en el menú **View**. Finalmente, en el margen derecho, se sitúan las ventanas del editor de código de la ficha activa y una primera ficha de trabajo en blanco en disposición de alojar componentes, estando normalmente constituidas las aplicaciones por más de una de éstas.

Pueden agregarse componentes a la paleta **Component** por medio de la secuencia de menús **Component | New Component** (Componente | Nuevo Componente), así como editar su composición con ayuda del cuadro de diálogo que se muestra mediante la secuencia de menús **Component |**

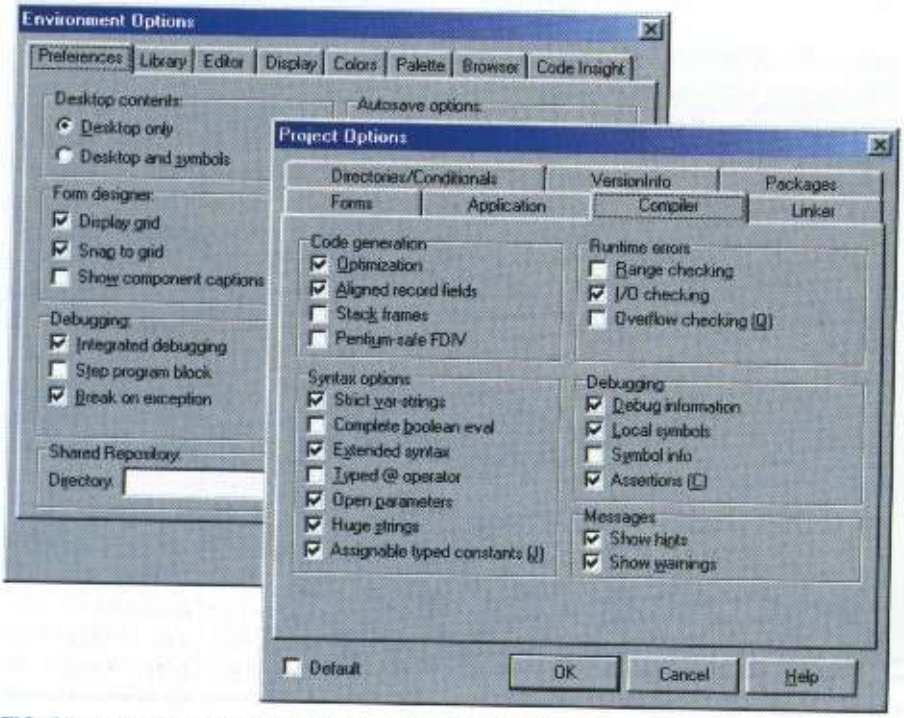

FIG. 8 La configuración de opciones de depuración de programas se distribuye en varios cuadros de diálogo.

Configure Palette (Componente | Configurar Paleta). Es posible también añadir o modificar el orden de las fichas por medio de la secuencia de menús **Tools | Environment Options** (Herramientas | Opciones de Entorno).

COMPONENTES, SUCESOS Y CÓDIGO

Tras insertar un componente de la paleta **Component** en una ficha se muestran automáticamente sus propiedades en el **Object Inspector**. Cuando se inserta más de un componente, lógicamente, cada objeto activo muestra sus propiedades al hacer clic sobre el mismo o, si se desea, también al elegir un componente dentro del cuadro de lista desplegable situado en el margen superior del **Object Inspector**. Éste muestra, por omisión, el contenido de su ficha **Properties** (Propiedades), que se organiza en dos columnas que corresponden a cada propiedad y valor del objeto activo. Así, para modificar una determinada propiedad, basta con hacer clic sobre la misma en el **Object Inspector** e introducir un nuevo valor en la columna situada a su derecha, que puede mostrarse como cuadro de texto o como cuadro de lista desplegable, en función de que permita introducir libremente un valor o deba elegirse obligatoriamente entre una serie de opciones determinada.

En ocasiones las propiedades van precedidas por un signo "+", lo cual indica que al hacer doble clic se mostrará el repertorio de subpropiedades completo, pasando el signo "+" a "-" para permitir el repliegue nuevamente con doble clic. Por otro lado, cuando la elección de valores depende de un cuadro de lista desplegable, se puede hacer doble clic sobre el valor para activar automáticamente el siguiente de la lista. Esto resulta especialmente útil cuando hay sólo dos valores posibles, como **True** (Verdadero) y **False** (Falso), conmutándose en este caso entre ambos con doble clic.

Para asociar un evento a un componente basta con pasar a la ficha **Events** (Sucesos) del **Object Inspector**, hacer clic sobre el elemento para seleccionarlo, elegir el evento a activar y seleccionar su valor mediante el cuadro de lista desplegable, o el cuadro de

HERRAMIENTAS DE DEPURACIÓN

El depurador integrado de programas es la mejor ayuda para la localización y solución de errores de programación. Permite controlar el flujo de ejecución del programa, verificar la sintaxis de las distintas estructuras y visualizar y modificar los valores de las variables durante la fase de depuración. El código se puede depurar en su totalidad o parcialmente, con posibilidad de ejecución paso a paso y definición de puntos de ruptura para efectuar comprobaciones de estado de variables, de la pila de llamadas a subrutinas y salida del programa a pantalla, disco o impresora. Las casillas de verificación del apartado Debugging (Depuración), de la ficha Preferences (Preferencias) del cuadro de diálogo Environment Options (Opciones del entorno), controlan las opciones de depuración. En el cuadro de diálogo Project Options (Opciones de Proyecto), ficha Compiler (Compilador) y apartado Debugging, existe una nueva batería de casillas de verificación con opciones. Durante la fase de depuración se controla el flujo de programa mediante el conjunto de botones situado en el margen derecho de la barra de botones Speedbar: Run (Ejecutar), Pause (Parar), Trace into (Depurar) y Step over (Saltar).

diálogo asociado para cambiar varias propiedades a la vez (se muestra un botón con puntos suspensivos) según corresponda. Se establece automáticamente el valor considerado por omisión al hacer doble clic sobre el valor, con lo cual se genera automáticamente el esquema del código Pascal necesario en el editor de código, listo para su puesta a punto, pudiendo seleccionarse también un valor distinto dentro del cuadro de lista desplegable.

La edición de propiedades de fichas y componentes, y asignación de eventos a componentes, genera automáticamente el código necesario (denominado subyacente) en la ventana del editor de código. La barra de título de la ventana muestra el nombre de archivo correspondiente a la ficha en edición; el área de trabajo, el código asociado; finalmente, en la barra de estado se muestra información de la posición del cursor en el documento y estado del conmutador de inserción / sobrescritura. Es posible, por ejemplo, definir combinaciones de teclas de abreviatura, fuentes y colores para resaltar los distintos tipos de componentes del lenguaje. El editor de código puede personalizarse, si se desea, mediante la secuencia de menús **Tools | Environment Options | Editor** (Herramientas | Opciones de Entorno | Editor).

Por otro lado, como ya se ha comentado anteriormente, es posible también editar directamente el código mediante un editor de textos independiente, actualizándose automáticamente los cambios en el entorno visual de edición del programa.

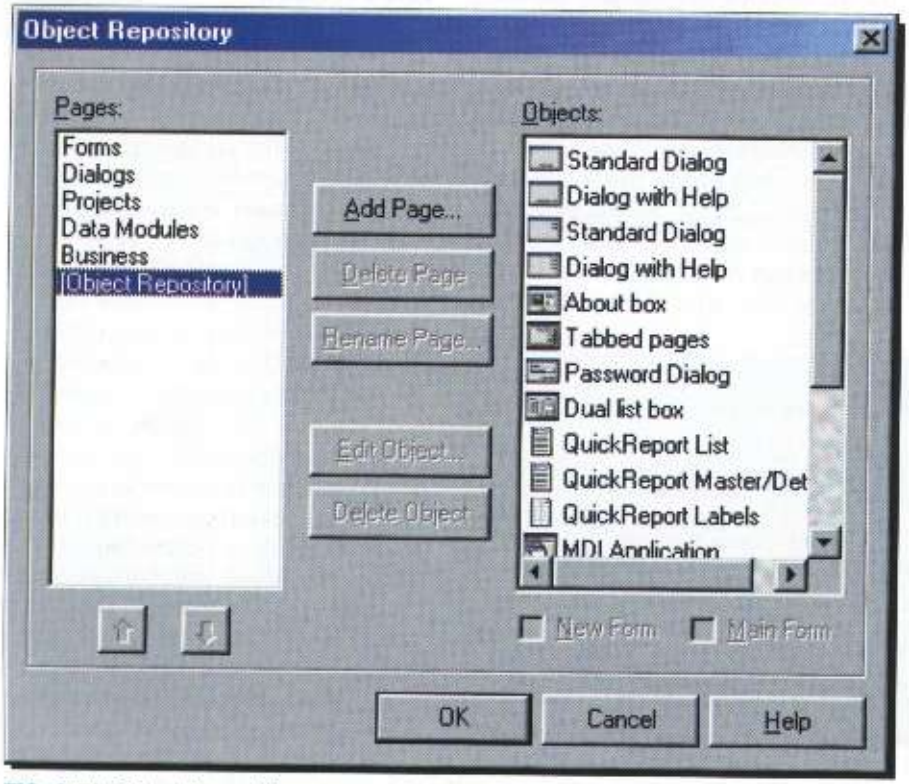

FIG. 9 El Object Repository pone al alcance del usuario una serie de componentes de ejemplo suministrados por Borland.

OTRAS HERRAMIENTAS Y FACILIDADES

La característica de exploración de bases de datos SQL permite crear y mantener alias de bases de datos, visualizar datos esquemáticos, mostrar campos de datos e índices, crear, visualizar y modificar datos contenidos en tablas, utilizar sentencias SQL para la consulta directa de bases de datos, y crear y mantener diccionarios para almacenar conjuntos de atributos.

El **Object Repository** (Almacén de Objetos), que cuenta por omisión con una serie de componentes de ejemplo suministrados por Borland, puede servir de almacén para los distintos objetos, fichas, y módulos de aplicaciones diseñados por el usuario para su inclusión posterior en otras aplicaciones, estando accesible mediante la secuencia de menús **Tools | Repository** (Herramientas | Almacén).

El **Project Manager** (Gestor de Proyectos) facilita la gestión de los archivos que componen el proyecto, accediéndose a la lista de archivos por medio de la secuencia de menús **View | Project Manager** (Ver | Gestor de Proyectos). La barra de botones de la ventana cuenta con opciones para añadir, visualizar y eliminar módulos y fi-

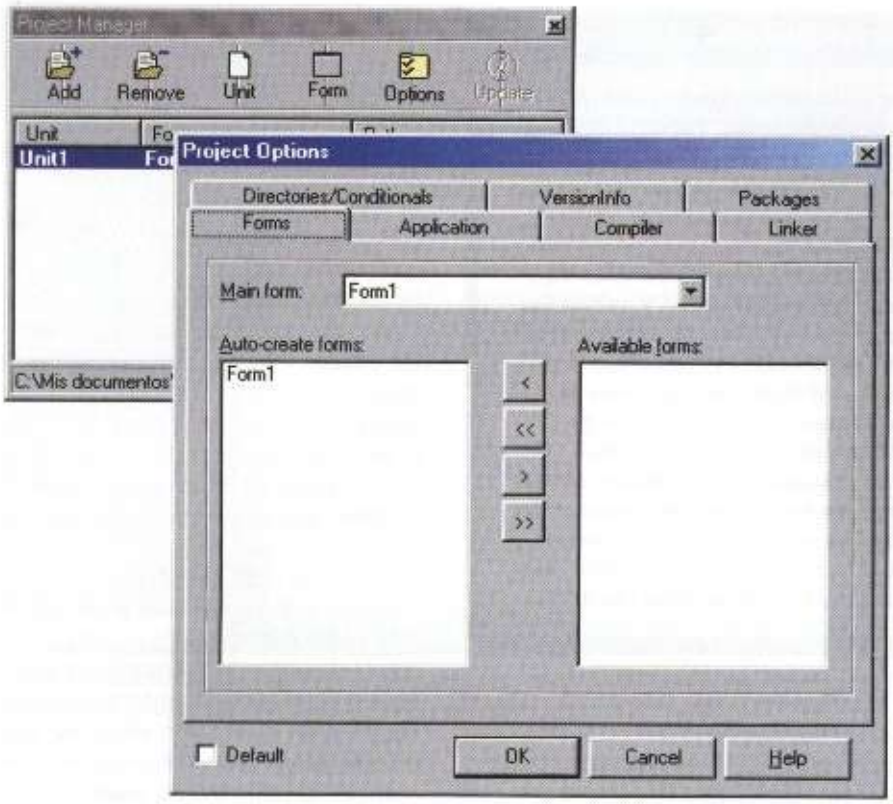

FIG. 10 El Project Manager permite organizar los archivos que componen un proyecto de la manera más eficaz.

chas, así como para modificar opciones de proyecto, indicándose igualmente el trayecto de disco completo de los archivos implicados, o como información general sobre el proyecto en la barra de estado de la ventana.

Como es habitual en aplicaciones desarrolladas para el entorno Windows 98, los menús contextuales dependientes del botón derecho del ratón, ponen al alcance del usuario las opciones de menú más importantes en función del elemento sobre el cual se pulsa. La barra de botones del margen superior, denominada **Speedbar** incluye, en el margen izquierdo de la paleta **Component**, los botones reservados a funciones estándar como: **Open project** (Abrir proyecto), **Save all** (Guardar todo), **Add file to project** (Añadir archivo al proyecto), **Select unit from list** (Seleccionar módulo de la lista), **Select form from list** (Seleccionar ficha de la lista), **Run**, **Pause**, **Open file** (Abrir archivo), **Save file** (Guardar archivo), **Remove file from project** (Eliminar archivo del proyecto), **Toggle Form / Unit** (Conmutar Ficha / Módulo), **New form** (Nueva ficha), **Trace into** (Depurar) y **Step over** (Saltar). Los nombres de los botones son visibles al mantener el puntero del ratón sobre los mismos, siempre que esté activada la opción **Show Hints** (Mostrar Sugerencias) del menú contextual de esta barra. Así mismo, el conjunto de botones al completo se puede ocultar al elegir la opción **Hide** (Ocultar) del menú contextual, para volver a mostrar por medio de la secuencia de menús **View | Speedbar** (Ver | Barra de botones de acción rápida). La posibilidad de obtener ayuda en línea es otra característica de interés, pues ésta recoge incluso más información que la que puede hallarse en el propio manual de la aplicación, accesible con sensibilidad al contexto al pulsar **F1** sobre un elemento determinado. Como es habitual en Windows, la ventana de ayuda se organiza en las fichas **Contents** (Contenido), **Index** (Índice) y **Find** (Búsqueda).

FIG. 11 La secuencia de menús File | Close All permite cerrar el proyecto activo y retirarlo del área de trabajo, asegurando previamente que se guardan en disco los posibles cambios pendientes.

CREACIÓN DE UNA APLICACIÓN PASO A PASO

Antes de iniciar un nuevo proyecto lo más recomendable es crear una carpeta independiente que aloje los archivos que lo compondrán, puesto que esta es la manera más eficaz de mantener perfectamente organizados los componentes en disco. Esto puede hacerse tanto desde el Explorador de Windows como desde el interior de la propia aplicación. Crearemos la subcarpeta *Ejemplos* alojada en la carpeta **Delphi3m** de instalación de Borland Delphi y accederemos al programa, con lo cual se mostrará automáticamente un nuevo proyecto en blanco sobre el área de trabajo. En el caso de que nos encontremos con otro proyecto en edición, puede cerrarse éste guardando los cambios, si procede, mediante la secuencia de menús **File | Close All** (Archivo | Cerrar Todo) para, a continuación, seguir la secuencia de menús **File | New Application** (Archivo | Nueva Aplicación) para crear un nuevo proyecto en blanco. Al hacerlo se crea por omisión el proyecto Project1 (de extensión DPR), como se

indicará en la barra de título de la aplicación, así como la ventana **Unit1.pas**, que contiene el código fuente Pascal del programa, y una ficha en blanco **Form1**. La secuencia de menús **File | Save All** (Archivo | Guardar Todo) nos permitirá almacenar en disco este proyecto, con sólo seleccionar la ruta adecuada (**C:\Delphi3m\Ejemplos**) dentro del cuadro de diálogo al efecto y hacer clic sobre el botón **Guardar** para cada uno de los nombres y extensiones propuestas, para el archivo de proyecto y el de código. Los botones del margen derecho del cuadro de diálogo permiten, ascender en la jerarquía de carpetas, crear una nueva carpeta dentro de la actual y elegir el modo de visualización, respectivamente. Si se inspecciona el contenido de la carpeta se han creado también dos más de proyecto con extensiones DOF y RES, así como otro de código con extensión DFM, para uso interno de la aplicación.

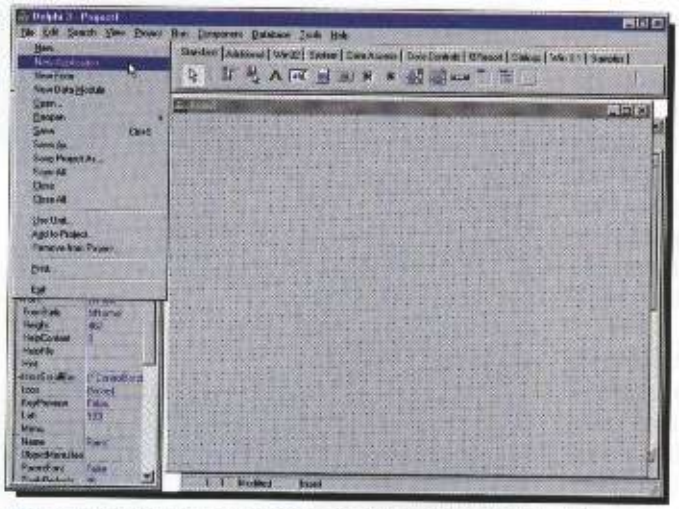

FIG. 12 La secuencia de menús File | New Application permite crear un nuevo proyecto en blanco.

Por otro lado, en cuanto a la ficha en blanco se refiere, puede comprobarse que se han creado automáticamente para ella los controles básicos, de manera que dispondrá de barra de título y los habituales botones de control del margen derecho de ésta, con su funcionalidad estándar, generándose automáticamente para ello el código necesario en la ventana de lenguaje Object Pascal. Estas fichas son las piezas fundamentales con las cuales se crean las aplicaciones. Pueden alojar menús, controles de todo tipo, e incluso asociarse para constituir complejos cuadros de diálogo formados por más de una ficha, con posibilidad de definir también anidamientos de ventanas. Lo realmente sorprendente es que, por ejemplo, la ficha que se acaba de crear ya tiene

FIG. 13 La paleta Component incluye todos los objetos que se pueden insertar en las fichas en forma de botones.

que se acaba de crear para alojar el nuevo proyecto, se descubrirá que, además de los dos archivos comentados,

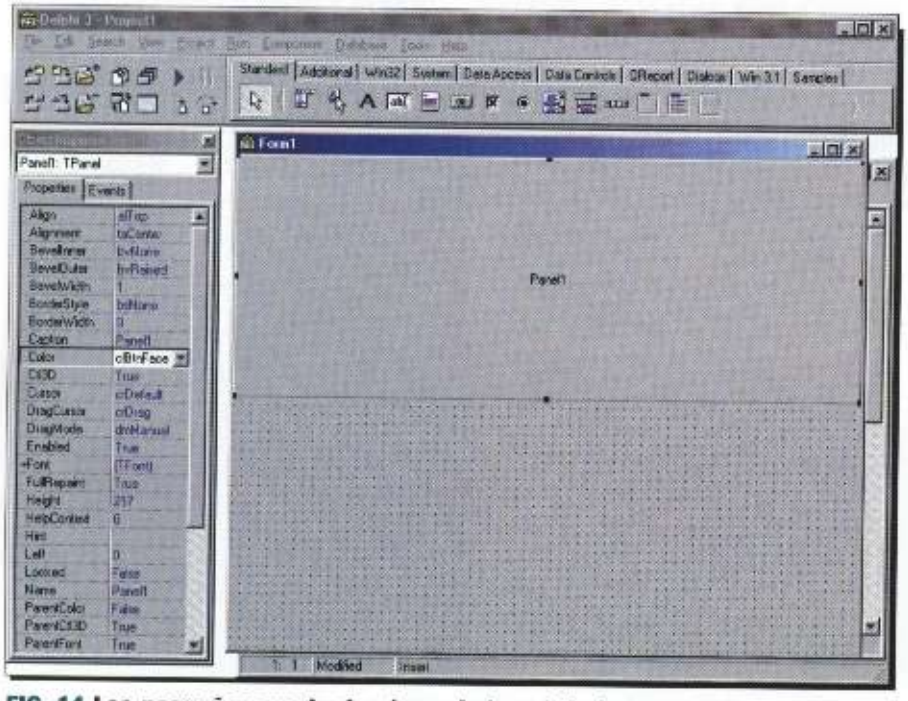

FIG. 14 Los pequeños cuadrados (manejadores) de los centros de los cuatro lados del componente permiten dimensionarlo en la ficha de trabajo.

esta funcionalidad básica operativa sin necesidad de codificación manual alguna. Para comprobarlo basta con pulsar **F9**, equivalente a la secuencia de menús **Run | Run** (Ejecutar | Ejecutar). Al pulsar con el botón izquierdo del ratón sobre los controles se puede minimizar, maximizar y cerrar la ventana, volviéndose al modo de edición normal al hacer esto último. La ejecución del programa de ejemplo, además, habrá creado un nuevo archivo de código de extensión DCU y otro de proyecto de extensión EXE, que es el ejecutable independiente de la aplicación.

El **Object Inspector** se hace eco también de la codificación básica. Así, por ejemplo, puede comprobarse por el cuadro de lista desplegable del margen superior que se ha creado un componente **TForm1** y que su nombre es **Form1**, como indica el valor de la propiedad **Caption** (Etiqueta).

ASIGNACIÓN DE VALORES A PROPIEDADES

Aunque, como ya se ha comentado anteriormente, es posible establecer las

propiedades de los objetos emplazados sobre las fichas directamente desde el código de programa, lo más deseable es dejar esta tarea en manos del **Object Inspector** y los editores de propiedades con que cuenta Borland Delphi. Además de comodidad, este comportamiento permite dejar en manos del programa la mayor parte del trabajo de codificación, con la protección frente a errores que ello supone.

Así, por ejemplo, para cambiar el nombre de la ficha basta con asignar a la propiedad **Caption** un nuevo valor, tecleando en el espacio situado a su derecha. Del mismo modo, cambiaremos el color de fondo de la ficha eligiendo uno alternativo dentro del cuadro de lista desplegable de tonos predefinidos asociado a la propiedad **Color**. Como podremos comprobar, el aspecto de la ficha habrá cambiado automáticamente con un mínimo de especificaciones.

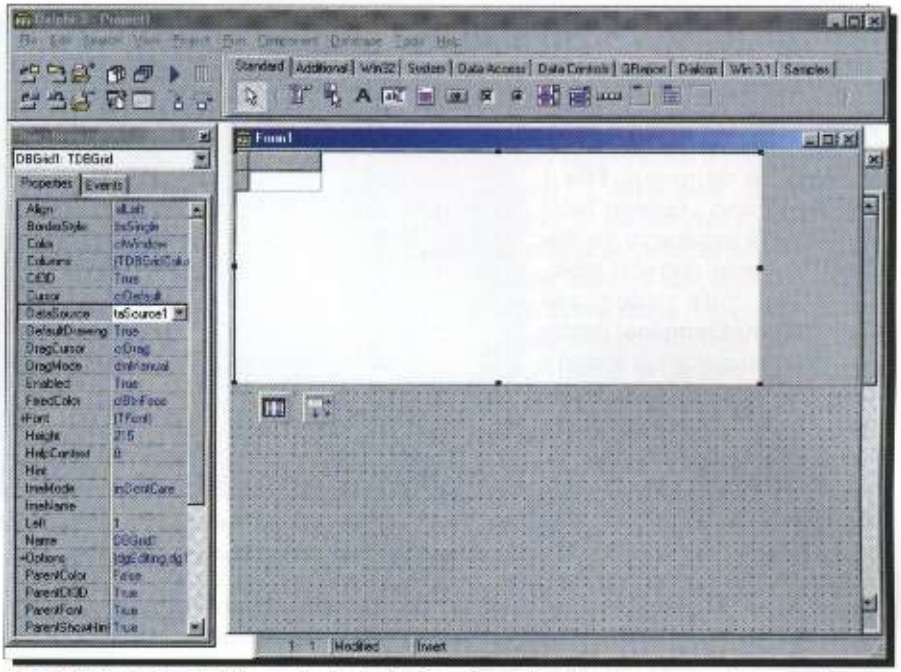

FIG. 15 Aspecto del área de trabajo tras insertar los componentes y antes de visualizar datos.

INCLUSIÓN DE OBJETOS EN LAS FICHAS

Comenzaremos por copiar los tres archivos de la base de datos de peces, almacenados en la carpeta **Ejemplos** del CD-ROM a la carpeta **C:\Delphi3m\Demos\DATA** donde vamos a ubicar las bases de datos gestionadas por Borland Delphi.

La paleta **Component** incluye todos los objetos que se pueden insertar en las fichas en forma de botones, convenientemente agrupados en fichas seleccionables mediante pestañas, en función de sus características. La inserción de componentes en las fichas es de lo más intuitivo, como mostraremos desarrollando un pequeño ejemplo de gestión de bases de datos.

Seleccionaremos la ficha **Standard** (Estándar) de la paleta **Component**, de no estarlo ya, al hacer clic sobre su pestaña correspondiente. Insertaremos ahora un objeto **Panel** que ocupe algo menos de la mitad superior de la ficha al hacer clic sobre el botón y arrastrar sobre la ficha con el botón izquierdo del ratón pulsado para definir arrastrando el área deseada, que se mostrará como **Panel1**, es decir, la primera instancia del objeto **Panel** que se coloca en la ficha de trabajo. Puede optarse también por hacer clic simplemente sobre cualquier posición de la ficha para que se coloque el objeto con las dimensiones por omisión para, seguidamente, modificar su propiedad **Align** (Alinear) para otorgarle el valor **alTop** (margen superior), que ajusta el componente al margen superior dándole todo el ancho de la ficha. En este caso, sólo restará arrastrar con el ratón el manejador del margen inferior hacia abajo hasta que ocupe algo menos de la mitad superior de la ficha de trabajo. La alineación puede efectuarse, si se desea, a los cuatro márgenes de la ficha, por medio de las distintas opciones del cuadro de lista desplegable de valores para esta propiedad. El valor **alClient** (cliente), por su parte, extiende las dimensiones del objeto al área completa de la ficha.

Pasaremos ahora a la ficha **Data Access** (Acceso a Datos) de la paleta **Component** e insertaremos un objeto

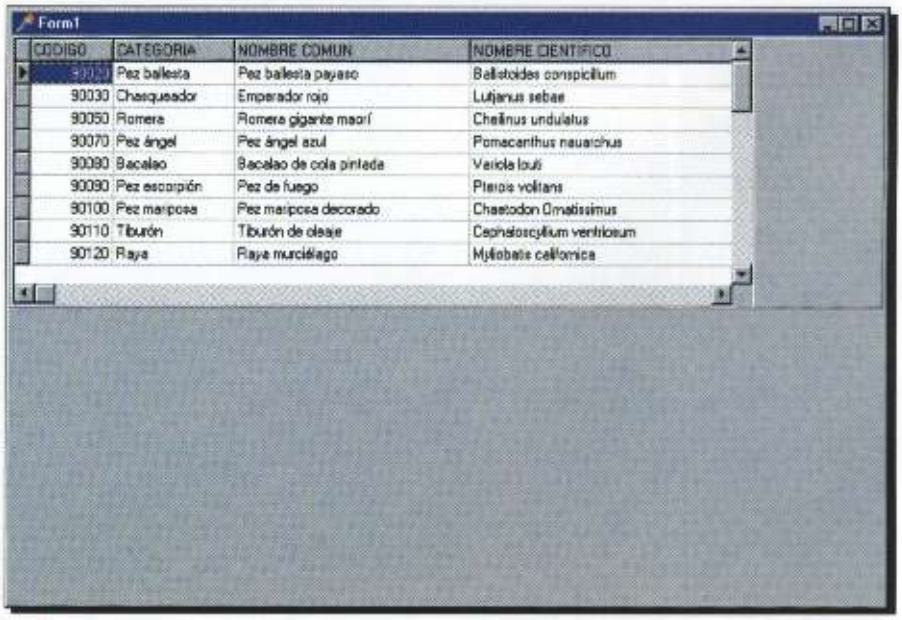

FIG. 16 Aspecto de la ficha en la ejecución del programa, donde no se muestran los objetos invisibles y los datos podrán ser opcionalmente editables.

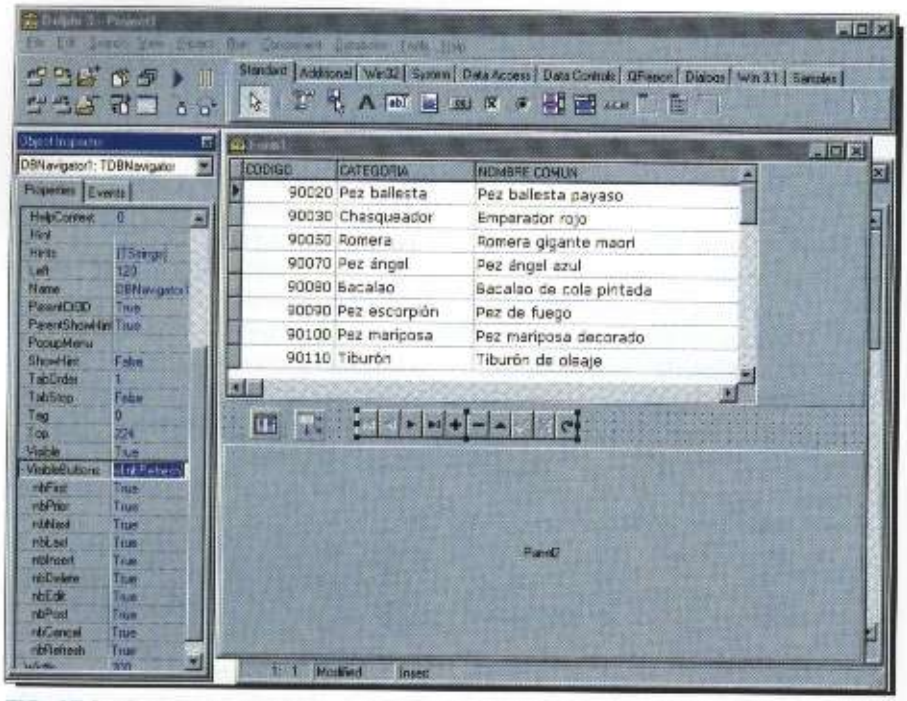

FIG. 17 La botonera muestra inicialmente todas las opciones, que se pueden activar / desactivar independientemente al mostrar las subpropiedades de la propiedad VisibleButtons

Table (Tabla) que colocaremos debajo y a la izquierda del panel. Estableceremos también la propiedad **ReadOnly** (sólo lectura) en **True** si deseamos proteger los datos frente a modificación del usuario y el valor **DBDEMOS** (cuadro de lista desplegable) para su propiedad **DatabaseName** (nombre de la base de datos). Es importante establecer de inicio la protección frente a escritura si se desea imponer puesto que, en caso contrario, una vez abierta la base de datos no será ya posible hacerlo y el valor por omisión es la libre edición de datos. Este componente, que no se muestra durante la ejecución del programa (podría colocarse en cualquier punto de la ficha, por lo tanto), habilita la funcionalidad de gestión de bases de datos. Debe indicarse ahora la fuente de la que provendrán los datos, insertando un objeto **DataSource** (origen de datos) del apartado **DataAccess** (acceso a datos), que resulta igualmente invisible en el momento de la ejecución del programa. Lo relacionaremos con el anterior objeto indicando para su propiedad **DataSet** (conjunto de datos) el valor **Table1**. Pasaremos al apartado **DataControls** (controles de datos) para insertar un objeto **DBGrid** (tabla de datos) que emplazaremos sobre el panel del margen superior de la ficha, con alineación a la izquierda al consignar **alLeft** (alineación izquierda) para la propiedad **Align**, como ya se ha comentado anteriormente. Debe asignarse también el valor **DataSource1** a la propiedad **DataSource** para vincular la tabla. Por último, reservaremos espacio en el margen derecho del panel para incluir posteriormente los dos botones de control de la aplicación.

Pasaremos de nuevo al objeto **Table1** al hacer clic sobre el mismo para indicar el nombre del archivo de bases de datos en su propiedad **TableName** (nombre de tabla) como **PECES.DB** (cuadro de lista desplegable) y activar la propiedad **Active** (activo) al asignarle el valor **True**, lo cual se puede hacer tanto mediante el cuadro de lista desplegable asociado, como al hacer doble clic sobre el valor, puesto que esta acción permite conmutar entre valores cuando el objeto sólo admite dos estados posibles, con lo cual se mostrarán automáticamente los nombres de campos y datos de la tabla de ejemplo. Esto resulta útil durante la fase de diseño para hacerse una idea del aspecto final que tendrá la ficha al ejecutarse la aplicación, aunque no será posible de momento modificar el contenido de la base de datos hasta no compilar el programa, siempre en función del valor indicado para la propiedad **ReadOnly** del componente **DBGrid**. Para ello pulsaremos **F9**, con lo que se mostrará el aspecto real de la ficha, volviéndose al modo de diseño al cerrar la ventana de la aplicación.

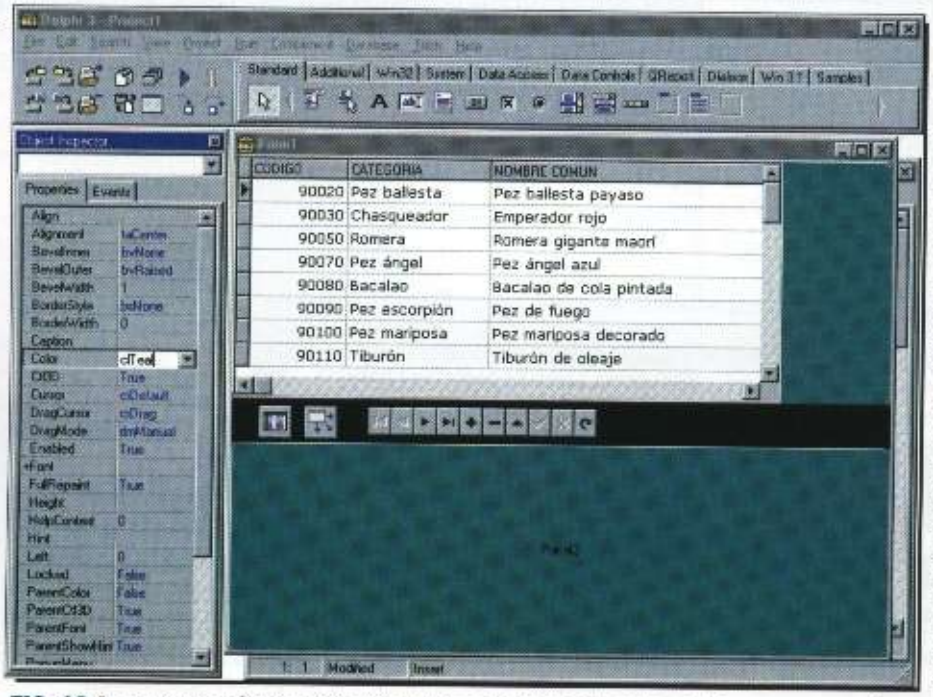

FIG. 18 Aspecto del área de trabajo tras el cambio de color de los componentes.

Incluiremos ahora una botonera de navegación por la base de datos para mejorar las prestaciones. Dentro del apartado **DataControls** elegiremos el componente **DBNavigator** (navegador de bases de datos), que posicionaremos a la derecha de los controles anteriores, justo debajo de la tabla de datos y aproximadamente centrado, asignado a su propiedad **DataSource** el valor **DataSource1** (cuadro de lista desplegable). En este caso sí es importante la ubicación del componente porque será visible. La barra de controles muestra, por omisión, todos los posibles.

Dado que la base de datos está protegida contra escritura y, por lo tanto, sólo sirve para su consulta, algunos de ellos se mostrarían pero sin estar nunca operativos.

Pueden eliminarse fácilmente. Para ello mostraremos la lista de subpropiedades de la propiedad **VisibleButtons** al hacer doble clic sobre la misma. Ahora sólo resta indicar el valor **False** para los botones sobrantes, a excepción de los de desplazamiento por registro hacia adelante y hacia atrás, y los de desplazamiento rápido al primer y último registro de la base de datos, respectivamente.

Deben quedar activos, por lo tanto, sólo los cuatro primeros: **nbFirst** (primero), **nbPrior** (anterior), **nbNext** (siguiente) y **nbLast** (último), tras lo cual puede volverse a replegar el detalle de subpropiedades.

Añadiremos un nuevo panel (apartado **Standard** de la paleta **Component**) en el margen inferior de tamaño similar al anterior, por el procedimiento descrito anteriormente. Su nombre automático será **Panel2**, asignándosele en este caso a su propiedad **Align** el valor **AlBottom** (margen inferior) y dimensionando el panel con ayuda del ratón y el manejador del margen superior de su contorno para que ocupe el resto del área visible de la ficha de trabajo.

Impondremos ahora un color adecuado al tipo de datos a mostrar. Para ello se debe hacer clic sobre la ficha para convertirla en elemento activo y establecer su propiedad **Color** en **clBlack** (negro), y seguidamente sobre los dos paneles para establecer la misma propiedad en **clTeal** (verde azulado).

En este segundo caso, y puesto que se va a asignar el mismo valor a una propiedad común, lo más cómodo es hacer clic sobre un panel y a continuación sobre el otro con la tecla **Mayús** pulsada, de manera que al asignar valor se aplique a todos los objetos resaltados al mismo tiempo. La técnica de selección múltiple de objetos es aplicable con carácter general para las propiedades en común.

El panel inferior va a estar destinado a mostrar dos tipos de datos especiales: una imagen y un campo de texto ampliado, comúnmente denominado *memo*. Volveremos al apartado **DataControls** para emplazar un componente **DBImage** (imagen de base de datos) que ocupe el margen izquierdo del panel inferior, asignando a su propiedad **Align** el valor **alLeft** como ya se ha comentado anteriormente, y ampliando sus dimensiones hasta ocupar la mitad del panel. Habrá que establecer también la relación con la base de datos, asignando a la propiedad **DataSource** el valor **DataSource1** y a **DataField** (campo de datos) **IMAGEN** (gráfico).

Insertaremos ahora el campo de texto extendido eligiendo el objeto **DBMemo** (campo memo) dentro del apartado **DataControls**, para ajustarlo por el procedimiento habitual de modo que ocupe el margen derecho del panel, en este caso otorgando el valor **alRight** a la propiedad **Align**. Determinaremos igualmente la relación con la base de datos asignando **DataSource1** a **DataSource** y **DESCRIPCION** a **DataField**. Incluiremos también una barra de desplazamiento vertical, puesto que por su tamaño no podrá mostrarse el texto completo en el espacio asignado para ello, otorgando a la propiedad **ScrollBars** (barras de desplazamiento) el valor **ssVertical** (vertical).

Reduciremos ahora ligeramente en vertical el espacio reservado dentro del panel al campo de texto ampliado, para dejar espacio al campo de datos **NOMBRE COMUN**, que contiene la descripción del pez, para el cual indicaremos un tipo de fuente distinto y de mayor tamaño, que sirva de título. Para ello basta con arrastrar con el ratón el manejador correspondiente hacia abajo, no sin antes restablecer su propiedad **Align** al valor **alNone** (ninguna) para activar el libre dimensionamiento. Puede incluirse ahora en el espacio ha-

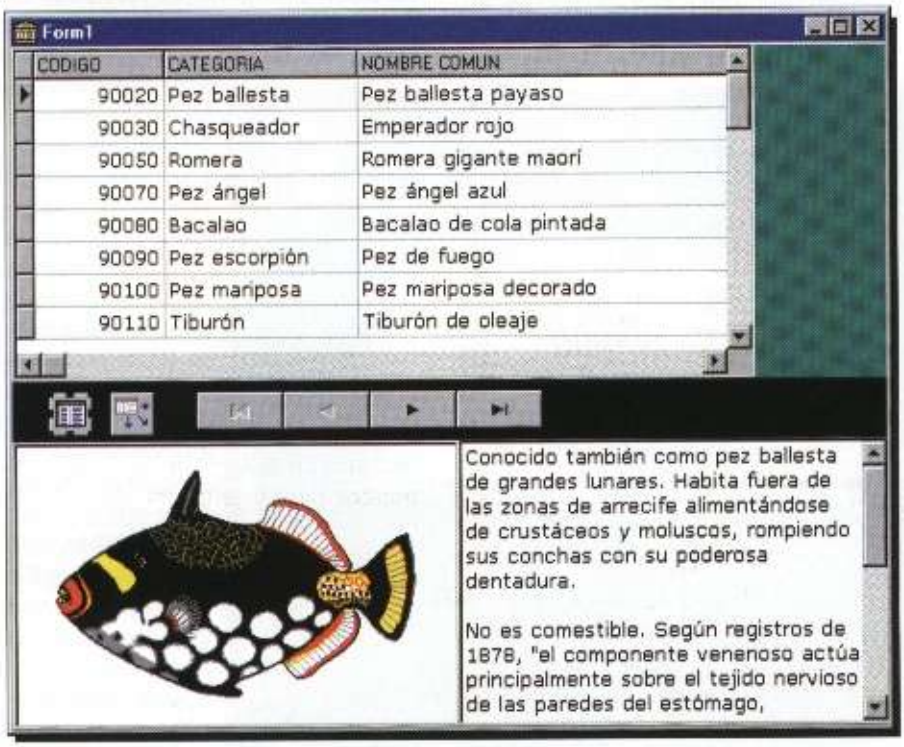

FIG. 19 Tras la inclusión del campo de texto ampliado con barra de desplazamiento la ficha debe tener un aspecto similar a éste.

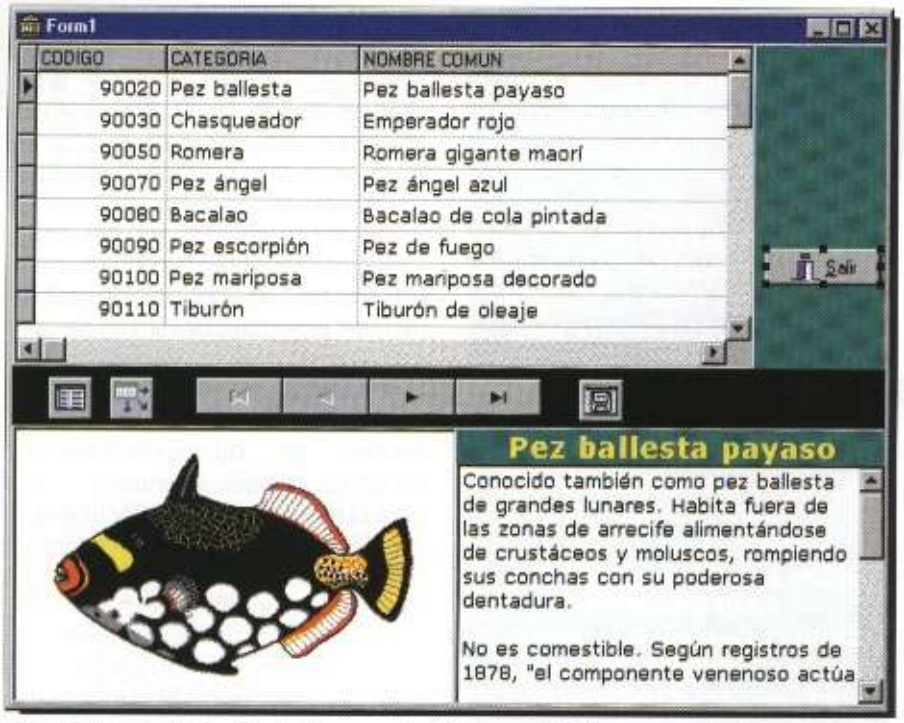

FIG. 20 Tras la inclusión del campo de título y el botón para salir el aspecto de la ficha debe ser similar a éste.

bilitado un componente **DBText**, vinculándolo a la base de datos y nombre de campo por el procedimiento descrito anteriormente, siendo el valor de **DataSource DataSource1** y el **DataField NOMBRE COMUN**. Centraremos también su contenido al asignar a la propiedad **Alignment** (Alineación) el valor **taCenter** (centrado). Para cambiar la fuente predeterminada de los campos de texto basta con hacer clic sobre el primero, y seguidamente sobre los otros dos con la tecla **Mayús** pulsada, y a continuación sobre el botón de puntos suspensivos para abrir el cuadro de diálogo asociado, eligiendo una fuente alternativa, como por ejemplo **Verdana** y validando al hacer clic sobre el botón **Aceptar**. Seleccionaremos a continuación sólo el campo de título, para ampliar su tamaño de fuente hasta **14** y añadir el atributo **Negrita**.

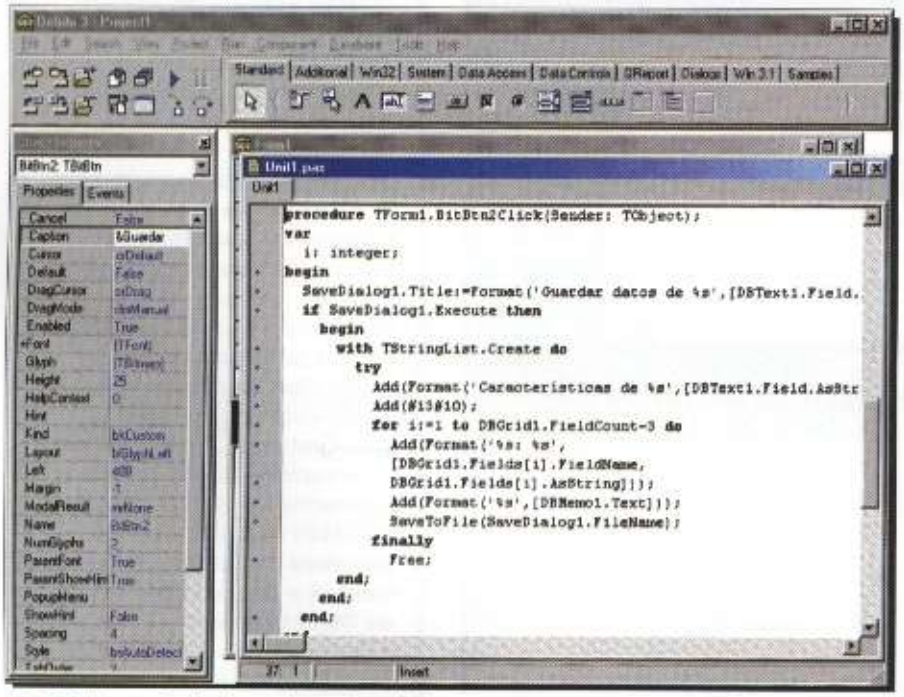

FIG. 21 Aspecto de la ventana de código Pascal tras la codificación del suceso asociado al botón para guardar en disco.

Para finalizar añadiremos un botón que permita abandonar la aplicación. Para ello elegiremos el objeto **BitBtn** (botón de acción) del apartado **Additional** (Adicional) de la paleta **Component**, que situaremos cerca del margen inferior del espacio reservado a la derecha del panel superior. A continuación asignaremos el valor **bkClose** (botón cerrar) a su propiedad **Kind** (clase), con lo cual se modificará el aspecto del mismo de acuerdo a la funcionalidad otorgada. Cambiaremos, además, el nombre y la tecla de acceso mediante teclado (en combinación con **Alt**), al teclear *&Salir* como valor de la propiedad **Caption**.

ASIGNACIÓN DE SUCESOS

Como podrá comprobarse al compilar (**F9**) se dispone ya de una versión completamente operativa, que permite inspeccionar la base de datos y salir de la aplicación con facilidad, sin haber necesitado para ello introducir manualmente ni una sola línea de código Object Pascal.

Añadiremos, finalmente, un segundo botón, que permita almacenar en disco los cambios introducidos en la base de datos antes de abandonar la aplicación.

En este caso sí será necesario codificar manualmente el manejador de sucesos asociado al botón, con lo cual escribiremos código Borland Delphi.

Pasaremos al apartado **Dialogs** (cuadros de diálogo) de la paleta **Component** para insertar un objeto invisible **SaveDialog**, que situaremos a la derecha de la barra de botones de control de desplazamiento en la base de datos. Dentro del apartado **Additional** seleccionaremos un objeto **BitBtn** para situarlo sobre el botón **Salir**. Para su propiedad **Caption** teclearemos *&Guardar* y para su propiedad **Glyph** (glifo) elegiremos dentro del cuadro de diálogo asociado el archivo FILESAVE.BMP,

al cual se accede al pulsar el botón **Load** (Cargar) del cuadro de diálogo **Picture Editor** (Editor de Imágenes) y examinar la carpeta ***\Images\Buttons*** situada dentro de la estructura de instalación del programa.

Crearemos ahora el manejador de sucesos por omisión del botón para almacenar en disco al hacer doble clic sobre el mismo, con o cual se genera el código básico y pasa a mostrarse la ventana de código Object Pascal.

Para ello basta con seguir las indicaciones reseñadas, puesto que por ahora no es relevante analizar a fondo el porqué de esta codificación.

No obstante, debe saberse que, en líneas generales, su misión es llamar al cuadro de diálogo estándar **Guardar como** (objeto **SaveDialog1** - guardar cuadro de diálogo) al pulsar el botón.

A continuación, se debe indicar un nombre de archivo y pulsar **Aceptar**, con lo cual se almacena en disco en un archivo de texto la información relativa al registro activo. Para ejecutar la versión definitiva del programa de ejemplo se debe pulsar **F9**, debiendo revisarse la redacción del código en el caso de que se produzcan errores de compilación. A continuación se detalla el código que debe introducirse.

```
procedure
TForm1.BitBtn2Click(Sender:
TObject);
var
 i: integer;
begin
 SaveDialog1.Title:=Format
('Guardar datos de
%s',[DBText1.Field.AsString]);
 If SaveDialog1.Execute then
   begin
     with TStringList.Create do
       try
      Add(Format('Caracteristicas
de %s',[DBText1.Field.AsString]));
         Add(#13#10);
         for i:=1 to
DBGrid1.FieldCount-3 do
           Add(Format('%s: %s',
[DBGrid1.Fields[i].FieldName,
DBGrid1.Fields[i].AsString]));
Add(Format('%s',[DBMemo1.
Text]));
SaveToFile(SaveDialog1.
FileName);
         finally
           Free;
     end;
   end;
end;
```

Nos dará una idea de la potencia del lenguaje la posibilidad de imponer el texto que se debe mostrar en la barra de título del cuadro de diálogo estándar de almacenamiento en disco de Windows, que en el caso del ejemplo es "Guardar datos de" más el valor de uno de los campos del registro activo. La expresión **Add (#13#10)**, por ejemplo, añade a la cadena de texto los caracteres especiales de retorno de carro y finalización de línea, generándose de este modo una línea en blanco de separación en el archivo en disco. El bucle **For ... To ... Do** obtiene el contenido de los campos estándar de datos para el registro en curso. Por último, la expresión **SaveToFile** es la encargada del volcado a disco de la información generada anteriormente. En definitiva, se trata de un sencillo programa de ejemplo que esperamos que haya servido de introducción al curso, y que pueda dar una idea de lo cómodo que puede llegar a resultar escribir aplicaciones de calidad con la ayuda de este potente entorno de programación visual.

FIG. 22 Aspecto final de la aplicación tras su compilación y puesta en marcha, con toda su funcionalidad.

RESUMEN

En esta primera unidad dedicada a Borland Delphi hemos expuesto sus características esenciales, comentando brevemente las diferencias más apreciables de esta nueva versión con respecto a las anteriores. Se ha hablado también sobre los elementos básicos del entorno como son la Speedbar, la paleta Component y el Object Inspector sin dejar de lado el conjunto de herramientas de depuración y otras facilidades para la programación interactiva de aplicaciones. Finalmente, y para fijar conocimientos, hemos iniciado el desarrollo de una pequeña aplicación de gestión de bases de datos de ejemplo, para practicar con la asignación de valores a propiedades a través de Object Inspector, la inclusión de objetos en las fichas y la asignación de sucesos a controles.