La clase Tword

(Versión 2.0)

Como crear documentos Microsoft Word desde Fivewin

2003 Sebastián Almirón

Ante todo mi agradecimiento a José Giménez por su excelente aportación con la librería OLE2 sin la cual la clase Tword no hubiera sido posible. También a Pedro Arribas López que inicio esta clase.

Quiero aprovechar estas páginas para mostrar mi agradecimiento a todos aquellos que hacen que el foro de FiveWin sea un sitio único, en el que un montón de gente aporta su tiempo y conocimientos desinteresadamente para provecho de los demás. También a los que realizan preguntas aunque no las contesten, pues gracias a ellos se cumple una función indispensable que es mantener la atención de todos hacia el foro y que esté vivo, subiéndonos el ego a algunos de nosotros al poder contestar alguna que otra pregunta ☺.

ESTE MANUAL AUN NO ESTÁ TERMINADO

Índice

1. REQUISITOS

Para poder utilizar la clase Tword, se deberá compilar el fichero Tword.prg como cualquier otro de nuestra a aplicación y enlazarlo del mismo modo.

También es necesario enlazar la librería OLE2 de José Jiménez (HBOle en caso de que utilices Harbour), antes de las de Fivewin, ya que las funciones OLE originales de Fivewin no funcionan correctamente (al menos hasta la versión 2.4). Enlazarla antes significa que en la declaración LIB de nuestro fichero LNK debe ir antes de las de FW:

En caso de FW 16 bits:
LIB OLE2, FiveC, Five, etc.

En caso de Harbour:
Enlazar la librería HBOle antes de las de FW, dependerá del método que utilices para enlazar.

Ello nos garantiza que todas las llamadas realizadas a las funciones OLE (OleInvoke(), OleSetProperty(), etc.) por la clase Tword se harán a las de la librería OLE2 y no a las de FiveC, ya que los nombres de las funciones son los mismos.

También es necesario tener instalado Microsoft Word 97 o 2000 en el equipo donde se ejecute la aplicación, ya que esta clase está construida a base de llamadas OLE. No ha sido probada con otras versiones de Word, pero no podrá funcionar con versiones que no sean compatibles con OLE.

Hay que tener en cuenta que al crear un documento Word lo normal será que se lo enviemos a otro usuario o que lo abramos en otro ordenador, por ello hay que prestar mucha atención a los tipos de letra a utilizar, deberán estar instaladas tanto en el ordenador que crea el documento como en cualquier otro que lo abra, si no queremos que tome otro formato. Por ello lo mas recomendable es utilizar las fuentes que instala Windows (Arial, Times New Roman, etc.) jugando con sus tamaños, negritas, itálicas y colores para dar forma al documento.

2. USO DE LA CLASE

El uso de la clase Tword se asemeja en gran medida al uso de la clase Tprinter de FW. Básicamente hay que guardar la siguientes estructura:

- Crear el objeto Tword.
- Declarar las fuentes, los Pen, etc. a utilizar
- Crear el documento o abrir uno existente.
- Insertar una página.
- Escribir y dibujar lo necesario (Say, box, line, etc.).
- Terminar la página.
- Grabar el documento .
- Finalizar el objeto.
- Destruir las fuentes, los Pen, los Brush, etc.

O en sintaxis xBase:

```
oWord := TWord():New()
DEFINE FONT oFont1 NAME 'Arial' SIZE 0, -12 OF oWord
oWord:NewDoc('prueba.doc')
oWord:StartPage()
oWord:Say( 1, 1, 'Esto es una prueba', oFont1)
oWord:EndPage()
oWord:Save()
oWord:End()
oFont1:End()
```

Para crear el objeto Tword utilizaremos:

```
oWord := TWord():New()
```

Esto correspondería a la declaración PRINT oPrint de la clase TPrinter .

Luego declararemos las fuentes que se utilizarán, no olvidando poner para que objeto son:

```
DEFINE FONT oFont1 NAME 'Arial' SIZE 0, -12 OF oWord
```

Es muy importante poner OF oWord al final, ya que de lo contrario el tamaño de la fuente nos puede crear malas pasadas.

A continuación debemos crear el documento o abrir uno existente:

```
oWord:NewDoc(nombre del documento nuevo)
o
oWord:OpenDoc(nombre del documento existente)
```

En el caso de un documento nuevo, lo siguiente que debemos hacer antes de escribir nada es declarar una página en blanco:

oWord:StartPage()

Ello inserta una primera hoja en blanco, es el equivalente a PAGE de TPrinter. Al igual que con TPrinter, tu programa debe controlar cuando será necesario un salto de página.

A continuación escribiremos todo lo necesario en la página actual, mediante los métodos Say, box, line, etc.

Una diferencia importante respecto a la clase TPrinter es que Word diferencia entre el documento en si y la cabecera. Si no ponemos nada, todos los Say, lines, etc. se pondrán en el cuerpo principal del documento, pero si queremos que estos se impriman en la cabecera debemos poner oWord:SetHeader(). ¿ Para que es esto ?. En muchas ocasiones nos encontramos con la necesidad de crear documentos con muchas hojas, en las que varían solamente los datos, pero que contienen un "impreso" sobre el que los datos se colocan. Cuando esto ocurre será mucho mas rápido crear una cabecera a partir de la página actual y que se imprimirá en todas las hojas restantes, para a continuación rellenarla con los datos. Para volver al cuerpo principal del documento deberemos poner oWord:SetMainDoc().

Otra diferencia importante es que en Word tenemos el texto principal y los cuadros de texto independientes que incorporemos. El método Say() crea un cuadro de texto independiente, el cual podemos colocarlo en cualquier parte del documento. Sin embargo el método Say2() colocará el texto en la línea y columna indicadas, añadiendo tantos retornos de carro y tabuladores como sean necesarios para alcanzar dicha posición, incluyendo saltos de página tal y como lo hace Word cuando es necesario y el método write() escribirá en la posición actual del cursor. Personalmente, salvo que el documento contenga únicamente texto puro y duro, creo que es mejor el uso de Say() y desaconsejo la mezcla de ambos por la dificultad de evitar que algún cuadro de texto se coloque sobre el texto principal.

Llegados a este punto hay que señalar que Microsoft Word trabaja internamente con unidades lógicas para las coordenadas. Todos los métodos que tienen parámetros de coordenadas se refieren a unidades lógicas. Dado que manejar dichos métodos con este tipo de coordenadas puede resultar muy engorroso, se puede llamar al método SetCm(), a partir de ese momento todos los métodos trabajarán en centímetros. Para volver a trabajar con unidades lógicas, utilice SetUL(). En realidad existe una DATA llamada lSetCm que debe valer .t. para que se trabaje con centímetros y lo único que hacen dichos métodos es cambiar el valor de esta data.

Para pasar a la siguiente página utilizaremos la secuencia oWord:EndPage() y oWord:StartPage(). En realidad EndPage() no hace nada, pero se ha incluido para mantener la misma sintaxis de tPrinter.

Una vez terminemos de escribir lo necesario, podemos guardar el documento con el método Save([nombre del documento]). Nombre del documento es un parámetro opcional y por defecto tomará el nombre del documento declarado en el método NewDoc() o en el método OpenDoc().

Llegado a este punto podemos visualizar el documento con el método Visualizar(), el cual funcionará siempre que el documento esté activo.

Para finalizar destruiremos el objeto con oWord:End() y las fuentes que tengamos declaradas con oFontx:End() y los Brushs.

Lógicamente para poder implementar un Preview(), Visualizar() o VistaCompleta(), lo más lógico es construir dos funciones mas o menos de esta forma:

- En la primera declararemos oWord como private y crearemos un cuadro de dialogo con tres botones al menos, el segundo de ellos deshabilitado. Con el primero llamaremos a la segunda función que es la que realmente crea el documento, con el segundo hacemos la llamada a oWord:Preview() y con el tercero cerramos el cuadro de dialogo, tras lo cual destruiremos el objeto oWord.
- La segunda función será la encargada de crear el documento en si, pero sin llegar a destruirlo, para poder ejecutar desde la primera los métodos necesarios. En esta función pondremos Enable() al segundo botón de la primera función, el cual permitirá hacer el Preview() una vez creado el documento.

3. PROPIEDADES

lStartPag
Contiene un valor .f. si no se ha ejecutado al menos una vez ::StartPage. Esta data es necesaria ya que Word añade una página en blanco al crear un nuevo documento, y por lo tanto añadiría una segunda al hacer nuestro primer StartPage(). No manipules su valor.

oLastSay
Contiene el objeto Text del último objeto Say, CmSay o TextBox, ello es útil para poder manipularlo una vez creado, por ejemplo si queremos añadirle tabuladores, podemos llamar a AddTabulador(npos, ::oLastSay).

LSetCm
Esta propiedad indica en que tipo de medidas se tomas los parámetros de coordenadas. Valdrá .T. para indicar que las coordenadas que pasamos a los métodos son en centímetros. Por defecto vale .F. (en unidades lógicas de Word) por lo que si queremos indicar a nuestra aplicación que las coordenadas son en centímetros debemos poner oWord:lSetCm := .T. o lo que es lo mismo llamar a oWord:SetCm(). Se puede llamar a oWord.SetUL() para poner el valor de lSetCm a .F..

Loverflowing
Esta propiedad indica si en la última operación de crear un cuadro de texto (TextBox, Say) el texto entró completo (valor .T.) en el cuadro, o por el contrario no entró completamente (Valor .F.)

NlastRow
Esta propiedad contendrá la línea (en centímetros o unidades lógicas dependiendo del valor de ::lSetCM) donde acabó el último cuadro de texto, box, imagen, etc.

CtextOverflow
Si durante la última operación de crear cuadro de texto, el texto no entró completamente, esta propiedad contendrá la cadena de texto que no entró. Puede ser una cadena simple si se utilizó Say o textBox, o una cadena Fget si en la última operación se utilizó SayGTF.

4. METODOS

AddImagen(nTop, nLeft, nBottom, nRight, cImagen, alinea, ntipo, nrotacion) -> Nil

Añade y ajusta a las coordenadas indicadas la imagen contenida en el fichero cImagen.
-nTop, nLeft, nBottom y nRight son las coordenadas donde queremos insertar la imagen.
-cImagen es el fichero que queremos insertar, debe ser en alguno de los formatos admitidos por MS Word para insertar imágenes y por ello dependerá de la versión que tengamos instalada en nuestro equipo.
-alinea es un array con el formato de línea, por defecto una línea simple fina, consulta los valores que puede tomar en la explicación del método Box().
-nTipo es el tipo de autoforma en el que queremos insertar la imagen, por defecto un rectángulo, pero puede ser cualquiera de las autoformas de Word, igualmente consulta el método Box() para una explicación más detallada.
-nRotación es el ángulo de rotación sobre el eje Z que queremos aplicar, un valor positivo indica una rotación en el sentido de las agujas del reloj y un valor negativo al contrario.
Este método se ha mantenido por guardar la compatibilidad con Tprinter y con versiones anteriores, pero en realidad lo que hace es una llamada a ::Box().

AddTabulador(npos, ocuadrotext) -> Nil

Añade un tabulador en la posición npos, en unidades lógicas o en centímetros dependiendo del estado de ::lSetCm. Si se omite el parámetro oCuadroText el tabulador se añadirá al documento principal. Puede pasarse como parámetro ::oLastSay, para indicar que queremos añadir un tabulador al último Say, CmSay o TextBox, o el valor almacenado en ::oLastSay inmediatamente después de llamar a estos métodos.

Box(nTop, nLeft, nBottom, nRight, afondo, alinea, ntipo, nrotacion, lPicTextured) -> Nil

Bibuja una autoforma en las coordenadas indicadas:
-nTop, nLeft, nBottom y nRight son las coordenadas en unidades lógicas o en centímetros dependiendo del valor de ::lsetCM
-afondo y alinea son los arrays para indicar el color de relleno, degradado, fichero de fondo, etc. Tienen las mismas aplicaciones que en el método TextBox(), mira la explicación de este.
-nTipo es el tipo de la autoforma, si no se indica este parámetro, dibujará un rectángulo, pero puede ser cualquier otra autoforma de las disponibles en Word (rectángulos, circulos, balones, etc.). Están disponibles unas 140 autoformas, por lo que explicarlas todas aquí no tiene sentido, si quieres ver los posibles valores, mira en el examinador de objetos de Word los posibles valores de msoAutoShapeType, las mas usuales son:

- 1 rectángulo o cuadrado
- 5 rectángulo o cuadrado con las esquinas redondas
- 9 circulo o ovalado

-nrotation es un valor numérico que indica el grado de rotación de la autoforma sobre el eje z, por defecto vale 0 (sin rotación), un valor positivo indica que se ha de rotar en el sentido de las agujas del reloj y un valor negativo al contrario.

-lPicTextured se utiliza junto con el octavo dato del array afondo (nombre de archivo a usar como fondo), para indicar como se usa dicho archivo de fondo. .T. para indicar como un dibujo único (valor por defecto), y .F. para indicar que se utilice como una textura

Este método actualiza la data ::nLastRow al valor pasado en nBottom.

Close(oDoc) -> Nil
Cierra el documento oDoc y despacha todo lo pendiente de enviar a la cola de impresión, que es el valor que contendrá ::OleActiveDoc, inmediatamente después de llamar a ::NewDoc() o a ::OpenDoc(). Si no se especifica el parámetro oDoc, se cerrará el documento. Ten en cuenta que si este método lo utilizamos antes de Save(), Word no lo guardará a no ser que el usuario lo guardase desde Word. Salvo que se quiera crear otro documento a continuación, bastará terminar con ::End().

CMSay(nLin, nCol, cTexto, oFuente, nSizeHorz, cClrText, nBkMode, nPad, naltura, nColorIndex , lVertAdjust) -> Nil
Es lo mismo que Say, salvo que las coordenadas son siempre en centímetros. Aunque no tiene mucho sentido utilizar este método una vez que está disponible SetCM(), se ha mantenido para guardar la compatibilidad con versiones anteriores y con Tprinter.
Devuelve un objeto TextRange para poder manipular posteriormente el texto.

CheckSpelling() -> Nil
Llama al corrector ortográfico. Si encuentra alguna falta mostrará el dialogo del corrector, en caso contrario no muestra nada.

End() -> Nil
Cierra todos los documentos abiertos, terminando todas las operaciones de entrada y salida (impresión, buffers, etc.) y destruye el objeto Tword.

EndPage() -> Nil
En realidad no hace nada, pero se ha mantenido por guardar la compatibilidad con Tprinter y para clarificar el código.

FillRect(aRect, oBrush) -> Nil
Rellena un rectángulo en la coordenadas indicadas en el array aRect, con el Brush oBrush.
ARect debe contener cuadro elementos: nTop, nLeft, nBottom y nRight. En unidades lógicas o en centímetros dependiendo del estado de ::lsetCM.
Este método se ha mantenido por guardar la compatibilidad con Tprinter y con versiones anteriores de TWord, se recomienda utilizar en su lugar ::Box() por ser mucho mas flexible.

GetTextHeight(oFont) ->nAltura
Devuelve la altura, en unidades lógicas o en centímetros, de la fuente oFont. La altura se corresponde con la distancia entre la parte superior de una E mayúscula y la inferior de una j minúscula.

GetTextWidth(cText, oFont) -> nAncho

En principio es de uso interno de Tword, devuelve el ancho aproximado que ocupa el texto cText con la fuente oFont. Es aproximado porque no se utiliza para su cálculo funciones de Word, sino de FW y puede haber diferencias.

GoBottom() -> Nil
Va al final del documento, es el equivalente a presionar Ctrl+End.

Gotop() -> Nil
Va al principio del documento, es el equivalente a presionar Ctrl.+Home

JustificaDoc(nJustify, oTexto) -> Nil
Justifica el texto indicado por el objeto oTexto, valor devuelto por ::Say, ::TextBox(), CMSay(), o por el objeto actual ::Oletexto (valor por defecto).
-Njustifi indica la justificación, puede tener uno de los siguientes valores:

0 Ninguna justificación, o lo que es lo mismo a la izquierda
1 Justificación a la derecha
2 Justificación centrada
3 Para justificación completa
4 Para justificación distribuida
5 Para indicar que el texto ha de escribirse a media altura del cursor
7 Para indicar que el texto ha de escribirse sobre el cursor
8 Para indicar que el texto ha de escribirse bajo el cursor

Line(nTop, nLeft, nBottom, nRight, oPen, nColor, nStyle) -> Nil
Dibuja una línea en las coordenadas especificadas.
-OPen es el lápiz a utilizar, si no se especifica se utilizará un lápiz estándar solido
-nColor es el color RGB de la línea, si no se especifica se utilizará el color declarado en el oPen y si este no fue pasado en negro
-nStyle es el estilo de la línea, se tomará por defecto el estilo de la línea del oPen, si este se pasó, en cuyo caso estará limitado a los estilos de línea de FW. Si se especifica, los estilos de línea pueden ser todos los disponibles en Word, mira los posibles valores para msoLineDashStyle en el examinador de objetos de Word. Por ello si queremos que tome el estilo de línea del oPen no debemos pasar este parámetro.

New() -> oWord
Crea un objeto Tword y devuelve dicho objeto.

NewDoc([nombre del documento]) -> Nil
Crea un documento MS Word nuevo. El parámetro "nombre del documento" debe contener una ruta válida y un nombre válido. Si se omite este parámetro se creará un documento con el nombre Documento1, en este caso no olvide indicar un nombre válido en la llamada a Save(). Aunque dado que como estamos ejecutando otra aplicación, podemos dejar a criterio del usuario el guardar el documento desde Word.

NLogPixelx() -> 55.38
Devuelve el valor 55.38. Este método es necesario tenerlo, ya que al declarar las fuentes, la clase Tfont llama a este método y es la única finalidad que tiene, evitar que al declarar las fuentes con OF oWord se emita un error.

NLogPixely() _> 55.38
El mismo caso que el anterior.

OpenDoc(nombre del documento) -> lOpened
Abre el documento "nombre del documento" el cual debe contener la ruta. Este método devolverá .F. en caso de que "nombre del documento" no exista o no sea un documento Word o no se pudiera crear el objeto y .T. si se abrió correctamente. Si solo se especifica el nombre del documento, sin la ruta, tomará por defecto el directorio actual.

Preview() -> Nil
Ejecuta la Vista Preliminar de en Word. Para poder tener la Vista Preliminar es necesario tener visible Word, cosa que ::Preview() hace.

PrintDoc(lbackground, lappend, nRgange, cOutputFile, nfrom, nto, nitem, ncopias, cpages) -> Nil
Imprime el documento activo o parte de el o lo manda a un fichero de salida.
-lbackground hace que continúe la ejecución mientras se imprime, si vale .T., o espera a colocar el documento en la cola de impresión antes de continuar, si vale .F., que es el valor por defecto.
-lappend se utiliza en combinación con cOutputfile, para indicar si al crearse un fichero de salida de impresora, se sobrescribirá (valor .T.) o se añadirá (valor .F.) al contenido de este en caso de que exista.
-nRange es el rango de impresión. Debe contener un valor del 0 al 4:

- 0, por defecto, indica que se imprima todo el documento
- 1 indica que se imprima solo el item nItem.
- 2 indica que se imprima la página actual.
- 3 indica que se imprima desde la página nFrom a la página nTo.
- 4 indica que se imprima el rango de páginas contenido en el parámetro cPages.

-cOutputfile es el fichero de salida de impresora, si se quiere especificar. Esto creará un fichero de salida, con todos los datos necesarios para imprimirlo desde un puesto que no tenga instalado Word, mediante un comando MS-DOS COPY fichero LPT1:. Como es lógico, al contener todas las secuencias de impresión, aunque no sea necesario el Word para mandar este fichero a la impresora, si será necesario que el modelo de impresora sea el mismo al que se mandó desde el ordenador donde se creó el fichero y que tenga las fuentes utilizadas.
-nFrom, nTo es el rango de páginas a imprimir, se deberá especificar nRange como 3, pero en cualquier caso, si alguno de estos parámetros contiene algún valor, el método automáticamente pone nRange como 3.
-nItem es lo que queremos que se imprima del documento, debe tener un valor de 0 a 5:

- 0 Para imprimir el contenido del documento, es el valor por defecto.
- 1 Para imprimir el resumen del documento, es el equivalente a seleccionar imprimir resumen en el dialogo de impresión de Word.
- 2 Para imprimir los comentarios, es el equivalente a imprimir comentarios.
- 3 Para imprimir los estilos
- 4 Para imprimir autotexto
- 5 Para imprimir Asignaciones de teclas

-nCopias es el número de copias que queremos, por defecto 1

-cPages es una cadena de caracteres que indica que se imprima el intervado de páginas cPages. Puedes indicar números de pagina o intervalos, separados por comas ('1,5, 9,12-16, 20').Para este valor consulta Word para ver como se declaran los rangos de páginas.

Hay que tener en cuenta, como ya se indica en otras partes de este documento, que es muy importante antes de mandar nada a oWord, hacer un StartPage(), para que la clase pueda tener un control sobre el número correcto de páginas. Consulta el método StartPage().

Protect(cpassword, nmodo) -> Nil
Permite proteger el documento activo con un password.

- cPassWord es una cadena de caracteres con el password
- nModo es el modo de protección y puede ser uno de los siguientes valores:
  - 0 para proteger todo excepto los cambios realizados.
  - 1 para proteger todo excepto los comentarios (valor por defecto).
  - 2 para proteger todo excepto los formularios

Replace(cOld, cNew) -> Nil
Este método nos permite reemplazar una cadena de caracteres por otra. Aunque en un principio pueda parecer que algo simple, con un uso adecuado nos brindará toda la potencia necesaria para crear mailings, o lo que es lo mismo combinar correspondencia en Word. Por ejemplo:
Supongamos que queremos mandar una carta personalizada a una serie de clientes en la que solo cambia el nombre, domicilio y población. Por supuesto será mas rápido crear dicha carta directamente con Word con el texto, dibujos y lo que queramos, únicamente tendremos la precaución de donde queramos que aparezca el nombre, domicilio y población poner lo siguiente:
([Nombre de cliente])
([Domicilio])
([Poblacion])
Bastará entonces con abrir dicha carta con oWord:Open y hacer las siguientes llamadas:
OWord:Replace('([Nombre de cliente])', clientes->nombre)
OWord :Replace('([Domicilio])', clientes->domicilio)
Oword :Replace('([Poblacion])', clientes->poblacion)
Por supuesto el uso de ([y de]) es solo una convención porque será difícil que en el texto aparezca esa secuencia de caracteres, pero se puede usar la secuencia que se desee.

Save(cnombredoc) -> Nil
Guarda en disco el documento, si no se especifica cnombredoc, se tomará por defecto el nombre del documento actual. Por ello este método es el equivalente a Save y a Saveas.

Say(nLin, nCol, cTexto, oFuente, nSizeHorz, cClrText, nBkMode, nPad, naltura, nColorIndex) -> Nil
Inserta un cuadro de texto en la posición indicada.
-NLin y nCol son las coordenadas (Recuerda que estos parámetros pueden ser en unidades lógicas o en centímetros dependiendo de si se llamó o no a SetCm().
-cTexto es el texto que se desea insertar
-oFuente es la fuente a utilizar.
-nSizeHorz es la anchura del cuadro de texto en unidades lógicas o centímetros.

-nClrText es el color de primer plano del texto en formato RGB
-nBkMode es para indicar si se desea que el fondo del texto sea transparente, 1 para transparente y 2 para opaco (valor por defecto)
-nPad es la alineación, 0 para ninguna, o lo que es lo mismo, a la izquierda, 1 para derecha y 2 para centrado.
-nAltura es la altura del cuadro de texto en unidades lógicas o centímetros.
-nColorIndex es un valor de 0 a 16 para el color de fondo, es el equivalente a resaltar texto en Word, para ver los colores disponibles consulte los posibles valores de WdColorIndex en el examinador de objetos del editor de VB de Word. Por ejemplo un valor 6 es para resaltar el texto en rojo. El valor por defecto es 0, sin resaltar.

.

Say2(nLinea, nColumna, cTexto, oFuente) -> Nil
Imprime el texto cTexto en la línea y columna especificada. Para ello parte de la posición actual del cursor en el texto principal y añade tantos retornos de carro como sean necesarios hasta alcanzar la línea nLinea, a continuación añade tabuladores hasta alcanzar la columna nColumna. En este caso nLinea se refiere a línea de texto y no a coordenadas y nColumna a saltos del tabulador.

SayGTF(nTop, nLeft, ctextFormat, nBottom, nRight) -> aresult

Este método nos permite crear un cuadro de texto con el contenido de una variable con formato FGET, conservando los tipos de letra, tamaño y colores del FGET original.
Todos los parámetros son obligatorios y pueden ser unidades lógicas o centímetros dependiendo del estado de ::lSetCM.
Con ello podemos tener cuadros de texto con distintos tipos de letra mezclados en una misma cadena, según como lo pusiera el usuario (Para mas información ver la magnifica clase FGET de Ramón Avedaño).
Este método devuelve un array de dos posiciones. La primera valdrá .T. si el texto ctextformat a entrado dentro de los límites del cuadro indicado y .F. si no cabe o no es un texto con formato FGET. La segunda posición contiene, en caso de que la primera valga .T., una nueva cadena con formato FGET con el resto del texto que no ha cabido en el cuadro. De este modo podemos decidir que hacer con el resto del texto, Pj: ponerlo en la siguiente página con un nuevo SayGTF() y así sucesivamente.

SetCM() -> Nil
Selecciona que las coordenadas pasadas a los métodos de Tword se harán en centímetros. Pone a .T. la data ::lSetCM.

SetHeader() -> Nil
Provoca que todos los métodos que imprimen en el documento lo hagan en la cabecera del mismo. De este modo podemos crear un "impreso" que se repetirá en las sucesivas páginas.

SetLandScape() -> Nil
Selecciona el papel en modo apaisado

SetMainDoc() -> Nil

Provoca que todos los métodos que imprimen en el documento lo hagan en el cuerpo principal de este. Debe utilizarse después de SetHeader() si queremos que el resto de lo que enviemos no lo haga a la cabecera.

SetPortrait() -> Nil
Selecciona el papel en modo vertical.

SetUL() -> Nil
Selecciona que las coordenadas paradas a los métodos de Tword se harán en unidades lógicas. Pone ::lSetCM a .F.

StartPage() -> Nil
Añade una página al documento actual. Aunque dadas las características de Word, que añade sin indicárselo una primera página en blanco, pudiera parecer que no es necesario indicar la primera página, ya que de todos modos los Says, Lines, etc. se imprimirán, Tword necesita conocer el número de páginas declarados, por lo que es muy importante hacer un StartPage() al comienzo, antes de mandar nada, y cada vez que se necesite, ya que de lo contrario podríamos encontrarnos con sorpresas a la hora de declarar rangos de páginas al imprimir, etc. En realidad y dada esta característica StartPage() no hace nada la primera vez que se llama (dado que la primera página ya existirá), pero pone a .f. la data lStartPage, para mantener el control de las páginas declaradas.

TabClearAll(ocuadrotext) -> Nil
Borra todos los tabuladores declarados. Si se omite el parámetro oCuadroText borrará todos los tabuladores del documento principal. Puede pasarse como parámetro ::oLastSay, para indicar que queremos borrar los tabuladores del ultimo Say, CmSay o TextBox, o el valor devuelto por alguno de estos tres métodos.

TabPrederteminado(ncada) -> Nil
Establece el intervalo de las tabulaciones predeterminadas del documento activo. Ncada es un valor en unidades lógicas o en centímetros dependiendo del valor de ::lsetcm.

TextBox(nTop, nLeft, nBottom, nRight, ctexto, ofuente, nclrtext, nJustify, afondo, alinea, lVertAdjust, norientacion) -> aDatos
Este método es alternativo al uso de Say() y puede ser usado cuando necesitemos crear cuadros de texto mas complejos o que necesiten ser rellenados, hacer transparencias, degradados, etc.

-Los parámetros nTop, nLeft, nBottom y nRight se refieren a las coordenadas (en unidades lógicas o en centímetros dependiendo de ::lSetCM).

-Ctexto es el texto a imprimir.

-Ofuente el la fuente a utilizar.

-NclrText es el color del texto en formato RGB

-Njustifi indica la justificación, puede tener uno de los siguientes valores:

0 Ninguna justificación, o lo que es lo mismo a la izquierda
1 Justificación a la derecha

2 Justificación centrada
3 Para justificación completa
4 Para justificación distribuida
5 Para indicar que el texto ha de escribirse a media altura del cursor
7 Para indicar que el texto ha de escribirse sobre el cursor
8 Para indicar que el texto ha de escribirse bajo el cursor

-Afondo es un array de hasta 8 posiciones para especificar las características del fondo del cuadro de texto, las posiciones son las siguientes y se han de mantener este orden, aunque no se especifiquen todas, en cuyo caso se ha de poner la coma. Por ejemplo si queremos especificar solamente la posición 3, el array se ha de declarar {,,0.5}:

1. Color del primer plano del fondo en formato RGB
2. Color del segundo plano del fondo en formato RGB
3. Nivel de transparencia, con un valor de 0 a 1. Un valor 0 es ninguna transparencia, un valor 0.5 se correspondería con la semitransparencia del checkbox de Word y un valor 1 sería transparencia total, o lo que es lo mismo, no se vería el cuadro.
4. La cuarta posición es la forma del degradado, si se quiere especificar e irá desde del color de primer plano al color del segundo plano, puede tener uno de los siguientes valores:

   1 Horizontal
   2 Vertical
   3 Diagonal hacia arriba
   4 Diagonal hacia abajo
   5 desde la esquina
   7 desde el centro

5. La quinta posición se corresponde con las cuatro casillas de selección del tipo de degradado en Word y debe tener un valor de 1 a 4 si se especifica. Para esta opción se ha de especificar también la 4.
6. Es el tipo de trama de relleno. Es un valor comprendido entre 1 y 49. Dado que enumerar aquí todas las posibilidades sería muy largo, puede consultar en el editor de VB de Word los valores que puede tomar MsoPatternType. Por ejemplo para mostrar ladrillos en horizontal el valor sería 35.
7. Es el tipo de textura. Es un valor comprendido entre 1 y 21. Al igual que en el apartado anterior, consulte los posibles valores de MsoPresetTexture.
8. Es el nombre de un archivo a utilizar como textura de fondo. Los tipos de archivo admitidos son los mismo que admite Word para especificar una textura de fondo.

Ten en cuenta que algunas de estas opciones anulan otras, por ejemplo si se especifica una trama y además un archivo de fondo, la trama no se pintará, ya que al llegar a la opción 8 se sobrescribe. De igual modo no se puede aplicar una transparencia a un degradado, Word no lo permite.

-Alinea es un array de hasta siete posiciones para indicar las características de la linea de contorno, al igual que en el caso anterior, puedes no especificar todas, pero han de mantener el siguiente orden:

1. El ancho de la línea en unidades lógicas
2. El color del primer plano de la línea en formato RGB
3. El color de fondo de la línea en formato RGB
4. El nivel de transparencia
5. El tipo de línea y puede tener un valor de 1 a 8. Se corresponde con las opciones Tipo del folder Colores y Líneas del formato de cuadro de texto en Word. Un valor 1 hace la línea sólida.
6. Estilo de la línea, debe tener un valor de 1 a 5, se corresponde con las opciones Estilo de la pestaña de Colores y Líneas de formato de cuadro de texto en Word, aunque solo en parte, ya que en Word esta opción es una mezcla de estilos y grosores, estos son los valores que puede tomar:

   1 Línea simple
   2 Línea doble
   3 Una línea fina y en el interior otra mas gruesa
   4 Una línea gruesa y en el interior otra mas fina
   5 Una línea fina, una gruesa y otra fina

   Cualquier otro valor tomará el valor por defecto 1

-LvertAdjust, es un valor lógico que indica si queremos que en caso de que sobre espacio en el cuadro, queremos que el cuadro se encoja de largo para adaptarlo al tamaño que realmente ocupa el texto. Un valor .T. indica que si queremos adaptarlo y un valor .F., que es el valor por defecto, indica que no queremos. Si el texto no entra en el cuadro, este parámetro no tiene efecto, ya que no alarga el cuadro. De esta manera, si desconocemos la longitud del texto, podemos indicar un valor .T. para lVertAdjust y un valor para nBottom que coincida con el final de la página y dependiendo de lo que nos devuelva decidir que hacer.

-nOrientacion es un valor numérico que puede contener uno de los siguientes valores:

- 1 para orientación horizontal, por defecto
- 2 para orientación vertical izquierda
- 3 para orientación vertical derecha

Si especificas lVertAdjust = .T. este valor tiene que ser 1 o no indicarlo.

Este método pondrá a .T. la data ::lOverflowing si el texto entró en el cuadro indicado y a .F. en caso contrario. En este último caso la data ::cTextOverflow contendrá una cadena de caracteres con la parte del texto que no entró.

También se actualizarán las datas ::oLastSay, para que contenga el objeto TextRange usado, de manera que posteriormente podamos modificarlo, y ::nLastRow contendrá la línea donde acabó el cuadro, tanto si se especificó lVertAdjust como si no.

Supongamos que estamos creando un cuadro de dialogo en el que el texto es un campo memo cuya longitud desconocemos, al guardar en ::cTextOverflow la cadena que no ha entrado, podemos crear otro cuadro, probablemente en otra página, con la parte de no cabe y así sucesivamente.
También podemos seguir colocando otros objetos a continuación, ya que actualiza la data ::nLastRow a la posición donde acabó el cuadro.

UnProtect(cpassword) -> Nil
Desprotege el documento activo. CPassWord es la cadena de contraseña utilizada para proteger el documento. Las contraseñas hacen distinción de mayúsculas y minúsculas.

VistaCompleta() -> Nil
Activa la Vista Completa de Word.

Visualizar() -> Nil
Hace visible Word con el documento activo.

Write(cTexto, cFuente, nSize, lBold, lShadow, nColor) -> Nil
Escribe en la posición actual del cursor del texto principal del documento. Utiliza Write cuando quieras crear textos simples (contratos, cartas, etc.). En este caso deberás tener en cuenta que no tienes ningún control sobre la posición del cursor, simplemente el texto se mandará a donde está el cursor, por lo que los espacios deberás crearlos con CHR(13).

5. POSIBLES CAUSAS DE ERROR

Si durante la ejecución del programa se produce algún GPF (General Protectión Faillure o Fallo de Protección General), puede ser por una de las siguientes causas:

- La librería OLE2 debe incluirse antes de las de FW y nunca después, como ya se indicó en el apartado requisitos.

- El staksize debe contener un valor suficientemente alto ya que las llamadas OLE consumen mucha pila. El valor mínimo será 17.000, aunque se puede requerir hasta 27.000 dependiendo del tamaño de la aplicación.

- Otra posible causa es, que si usas Office 2000, en algunas versiones tiene un error en OLE. En la web de Microsoft hay un parche para corregir este problema en las versiones de lo tienen (hay que instalar el Pack2). Este es un problema de Office 2000 y no de FW ni de TWord.

-Si utilizas Tword con Harbour, hay que señalar que versiones antiguas de OLE2 (HBOle.LIB) dan un error en muchos métodos al hacer la llamada a OleGetProperty() cuando se le pasa como parámetro un objeto obtenido con OleInvoke(). Por ejemplo al hacer un Say(), puede dar un error en la línea que pone:
oFill := OleGetProperty(oCuadro ,"Fill") dentro del método Say()
En este caso deberás actualizar la librería HBOle, se ha comprobado que funciona con la versión que tiene fecha 23-10-2002. En cualquier caso, José Giménez recomienda volver a crear la librería con la versión de Harbour que utilicemos (ver mas detalles en las instrucciones de OLE2).

- TWord se ha probado con exito con varias versiones de FW, de bLinker y de Clipper. También se ha probado satisfactoriamente con Harbour Alpha Build 4.0 (Flex).

6. ¿ DONDE OBTENER MAS INFORMACIÓN ?

Para entender mejor el funcionamiento de Tword y sobre todo el valor que pueden tomar algunos parámetros de los métodos, es muy recomendable echarle un vistazo al editor de Visual Basic que incorpora MS Word (Herramientas->Macro->Editor de Visual Basic), una vez en el, podemos ejecutar el examinador de objetos (Ver->Examinador de Objetos). Esta maravilla nos ofrece una ayuda completísima de todo lo que podemos hacer con macros y de los valores que pueden tomar los parámetros. Por ejemplo, en este manual se habla en algún punto de los valores que puede tomar MsoPatternType, para ver dichos posibles valores, escribe en el Get que aparece junto a los prismáticos del examinador de objetos la palabra MsoPatternType, en el apartado Clases quedará seleccionada MsoPatternType, a la derecha aparecerán los miembros de MsoPatternType y al seleccionar alguno de ello podremos ver su valor abajo. Además con el botón de la derecha podremos obtener una ayuda bastante completa cuando tengamos seleccionado algo que no sea una simple variable.
Si examinamos el código de la clase Tword y buscamos su correspondencia en el examinador de objetos, enseguida trazaremos la relación de uno con el otro.
Algunos parámetros de Tword se modifican antes de la llamada OLE para adecuarlos a la sintaxis de Fivewin, por ejemplo en el método Line(), los estilos del Pen cuando se

declara en FW no se corresponden con los estilos DashStyle en Word, por lo que en dicho método hay un Do Case para según que estilo declaremos en el Pen, adecuarlo a los parámetros que acepta DashStyle.

Si solamente vas a crear un documento Word sencillo, probablemente con los métodos ya declarados en Tword te sea suficiente, pero si necesitas algo que Tword no tenga, puedes crear una macro en Word y luego examinar si código, verás como es relativamente sencillo pasarlo a formato xBase e incorporarlo a TWord.

Tword está construida con llamadas OLE, pero no hace uso de ToleAuto, por lo que en la mayoría de los casos será necesario tomar un objeto con OleGetProperty() y luego asignar un valor con OleSetProperty() o ejecutar algo con OleInvoke(), no pudiéndose encadenar en una sola llamada como ocurre con ToleAuto.

Por ultimo indicar que es posible crear documentos Word directamente con la clase ToleAuto.

7. EJEMPLOS

Todos los ejemplos se han metido en el fichero sample1.prg, ya que lo que interesa es lo que realmente hace tWord y no el crear muchos documentos.

El ejemplo muestra una manera de hacer un menú, en el que se diferencian las funciones de creación del documento, visualización e impresión. En los casos reales, este sistema se deberá pasar la mayor parte de las veces a un dialogo con botones para dichas opciones.

Al seleccionar crear documento word, se crea un documento, pero no se cierra, los siguientes apartados se corresponden con lo que hace el código situado inmediatamente después de su número en sample1.prg:

- //1 :Se crea un cuadro de texto con el método TextBox(), dicho cuadro se rellena de un degradado blanco-rojo-blanco, se le aplica un borde y se pone en el un texto.
- //2 :Se crea un cuadro de texto con una variable cuyo contenido no cabe en el cuadro, por lo que se añaden a continuación los cuadros necesarios hasta que entre todo (en realidad uno mas si no cambias el contenido de la variable) y además para los cuadros sucesivos se le indica que ajuste la longitud del cuadro al tamaño del texto.
- //3 :Se utiliza AddImagen para crear una autoforma (flecha), la cual se rellena con un BMP y se le añade una rotación. Ahora AddImagen, al contrario de lo que sucedía en versiones anterior, admite además de colocarla en un rectángulo, valor por defecto, colocarla en cualquier autoforma, y permite rotación.
-
- Se habilita las opciones del menú Ver documento e Imprimir documento.

Al seleccionar Ver Documento Creado se llama al método ::Visualizar() el cual muestra en pantalla Word con el documento creado

Al seleccionar imprimir se manda a la impresora

Al seleccionar Salir, si no se guardó el documento desde Word se mostrará el cuadro de dialogo de Word para guardar el documento. Si deseas que Tword guarde el documento y no muestre este dialogo, deberás utilizar ::Save(). Finalmente se destruye el objeto tWord.

Espero que os sea de utilidad

Sebastián Almirón
sebas@moralzarzal.com

```
#include "Fivewin.ch"

function Main()

 private oMenu, oMnu1, oMnu2, oMnu3, oMnu4, oMnu5, oWord

 DEFINE WINDOW oWnd FROM 0, 0 TO 24, 70 TITLE 'Ejemplo 1 de TWord';
  MENU BuildMenu()

 ACTIVATE WINDOW oWnd

return Nil
//----------------------------------------------
function BuildMenu()

MENU oMenu
    MENUITEM 'Documentos Word'
      MENU
        MENUITEM oMnu1 PROMPT 'Crear documento Word';
          MESSAGE 'Crea un documento Word';
          ACTION CreaWord()
        MENUITEM oMnu2 PROMPT 'Ver documento creado';
          MESSAGE 'Muestra Word con el documento creado';
          ACTION Visualiza() DISABLED
        MENUITEM oMnu4 PROMPT 'Imprimir documento';
          MESSAGE 'Imprime el documento';
          ACTION PrintDoc() DISABLED
        MENUITEM oMnu5 PROMPT 'Cierra el documento';
          MESSAGE 'Cierra el documento';
          ACTION Cierra() DISABLED
        SEPARATOR
        MENUITEM oMnu3 PROMPT 'Salir';
          MESSAGE 'Sale del ejemplo';
          ACTION salir()

      ENDMENU
ENDMENU

return

//----------------------------------------
function CreaWord()
local afondo := {nRGB(255,0,0), nRGB(255,255,255), ,2,4}
local alinea := {5}, oFont1
local ctexto2 := 'En este caso la prueba no cabe dentro del cuadro indicado, se demuestra como el r
esto del texto puede ser enviado a otro cuadro'
local atext

if valtype(oWord) = 'O'
  oWord:End()
endif

oWord := TWord():New()
oWord:NewDoc('Prueba.doc')
```

```
DEFINE FONT oFont1 NAME 'Arial' SIZE 0, -12 OF oWord

oWord:SetCm()
oWord:StartPage()

// 1
oWord:TextBox( 1, 2, 3, 10,  'Esto es una prueba con TWord', oFont1,,,,afondo, alinea)

// 2
oWord:TextBox( 4, 2, 5, 10, ctexto2, oFont1)
do whil oWord:lOverflowing = .t.
   oWord:textBox( oWord:nLastRow, 2, oWord:nLastRow + 2, 10, oWord:cTextOVerflow, oFont1,,,,,,.
t.)
enddo

//3
oWord:AddImagen( oWord:nLastRow, 10, oWord:nLastRow + 3, 13, 'Sample1.bmp',,35,36)

//4
oWord:Say( oWord:nLastRow, 1, 'Esto es un Say', oFont1)

//5
oWord:SetUL()
oWord:CMSay( 15,3,'Esto es un CMSay')

//6
oWord:Say2( 10, 3,'Esto es un Say2', oFont1)

//7
oWord:Say2( 1,1,'Esto es otro Say2 forzando el salto de pagina', oFont1)

oMnu2:Enable()
oMnu4:Enable()
oMnu5:Enable()

oFont1:End()

return

//---------------------
function Visualiza()
oWord:Preview()
return

//---------------------
function cierra()
 oMnu2:Disable()
 oMnu4:Disable()
 oMnu5:Disable()
 oWord:Close()
return
//---------------------
function salir()
if valtype(oWord) = 'O'
```

```
   oWord:End()
endif
quit

//----------------------
function PrintDoc()
oWord:PrintDoc()
return
```