7 Programación de un diálogo.

En esta práctica se mostrará lo sencillo que puede resultar programar una ventana en windows que responda ante eventos (acciones del ratón). Se ilustrará la potencia que C++ nos da al trabajar con los objetos que Microsoft pone a disposición del programador en el sistema.

7.1 Introducción a Windows

El entorno de desarrollo de VC++ 6.0 ofrece muchas posibilidades al programador, desde crear aplicaciones de diversos formatos y características, pasando además por la creación de librerías DLL, iconos, bitmaps, cursores, etc.

En el fondo, Visual C++ es una aplicación desarrollada por Microsoft que nos facilita especialmente el uso de todos los recursos que Windows pone a disposición del programador para lograr hacer aplicaciones y librerías con el aspecto y modo de funcionamiento clásico de este sistema operativo. Por eso, todo el entorno de desarrollo es un gran conjunto de posibilidades e ingredientes que permiten desarrollar aplicaciones para windows, aunque incluye también la posibilidad de desarrollar aplicaciones para DOS como hemos ido haciendo hasta ahora.

Es importante aclarar que cuando se pretende crear una aplicación que corra bajo Windows con VC++ se debe tener en cuenta si la misma estará basada en MFC o si será una aplicación WIN 32 estándar.

Microsoft Foundation Class (MFC):

Visual C++ se suministra junto con la biblioteca MFC que es un conjunto de clases predefinidas de C++. Estas clases han pasado a ser un estándar de desarrollo para aplicaciones Windows y poseen clases que facilitan mucho la programación en C++, por ejemplo al incluir un tipo de dato CString para declarar cadenas, como se hizo en la primera práctica. Esta clase no existe en C++ aunque existen mecanismos parecidos en la librería estandar pero que explicarlos supondría un dolor de cabeza para más de uno.

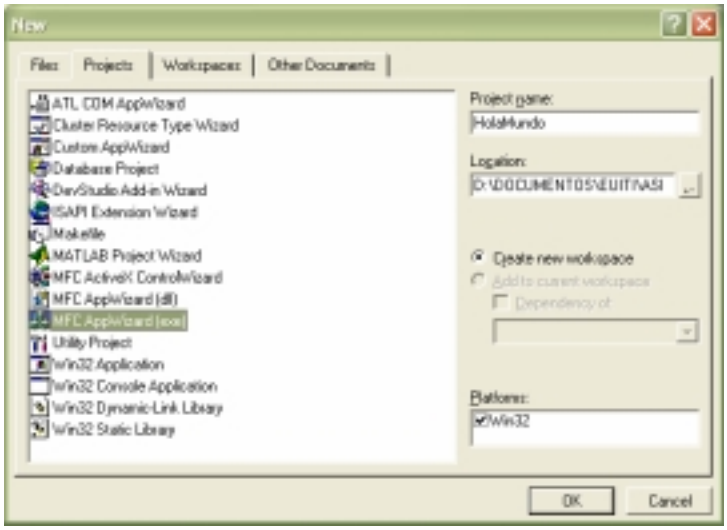
Las aplicaciones que utilizan las MFC necesitan distribuirse con estas librerías para asegurar que en aquellos sistemas en los que se ejecuten estas están presentes. Evidentemente, en muchos casos, al instalarse otros programas, se incluyen, por lo que en la mayoría de los casos, las MFC están instaladas ya en nuestro sistema por lo que no sería necesaria su instalación.

También se pueden crear programas Windows que no usen MFC, así se obtendría una aplicación con la cual no haría falta distribuir la librería MFC.

Veamos con el clásico ejemplo de programación como se puede realizar una aplicación que haciendo uso de las MFC nos permite mostrar una ventana con el mensaje “HOLA MUNDO”:

7.2 El diálogo Hola Mundo

Para comenzar a desarrollar esta pequeña aplicación abrimos Visual C++ y seleccionamos NEW en el menú FILE y apareciendo la siguiente pantalla:

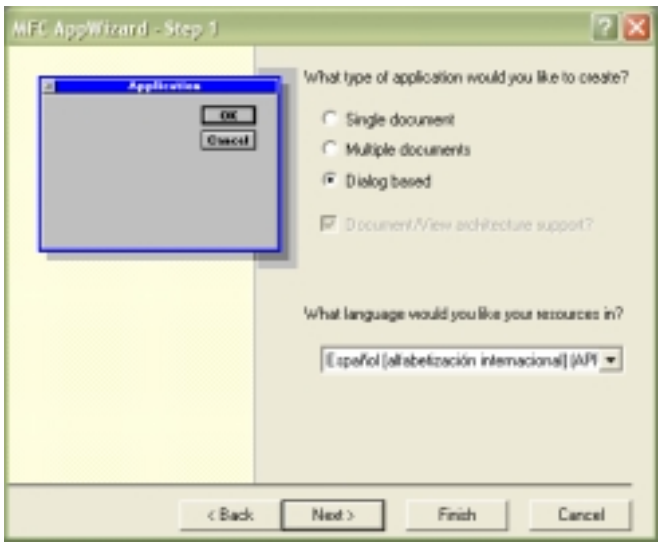


Esta pantalla muestra los distintos proyectos que uno puede desarrollar con VC++. Como queremos hacer una aplicación basada en MFC seleccionaremos **MFC AppWizard (exe)**. De esta forma un sensillo asistente nos permitirá crear una serie de ficheros que facilitan enormemente el desarrollo de la aplicación, además de seleccionarse automáticamente las opciones de compilación que permiten el desarrollo de aplicaciones basadas en MFC.

En **Project Name** escribimos el nombre de nuestro proyecto, y en **Location** indicamos el lugar en el que quedará almacenado el proyecto y todos sus ficheros por defecto. En el caso del laboratorio será necesario introducir "C:/ Usuarios/ Informática"

Una vez introducido el nombre del proyecto y la localización pulsamos el botón con lo que se da inicio al ASISTENTE DE APLICACIONES MFC el cual nos creará un armazón para nuestro programa.

Aparecerá por tanto la siguiente ventana:

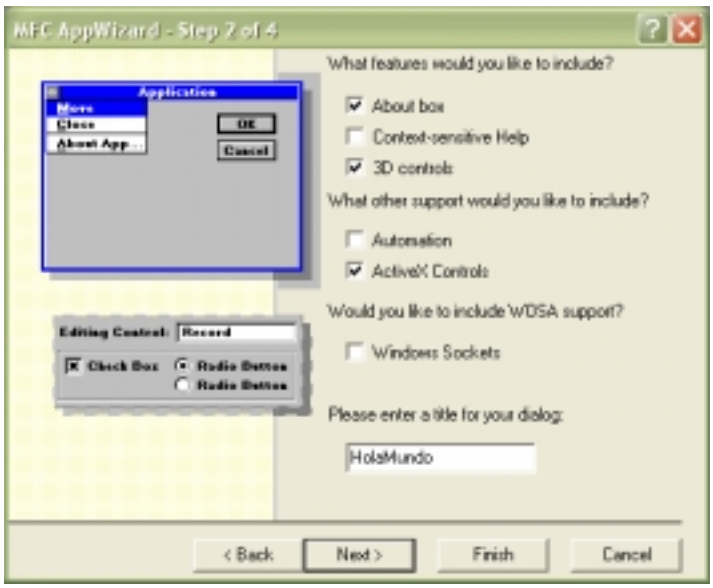


Una aplicación de Windows puede ser SDI (Single Document Interface), MDI (Multiple Document Interface) o Dialog Based (basada en dialogos), por lo que se solicita al programador que indique el tipo de aplicación que se quiere desarrollar.

Single Document es muy similar a Dialog Based, salvo que por defecto VC le agrega menús. El tipico programa SDI de Windows es el Bloc de Notas, el cual solo permite mostrar un documento, sobre el que se pueden realizar una serie de operaciones. Dialog Base, es una aplicación que no contiene un area de

viusta de documento, como por ejemplo puede ser el programa calculadora de Windows. MDI es una aplicación como por ejemplo Word, con una ventana principal que puede contener a muchas otras en su interior (como un contenedor). Estas ventanas internas son documentos que tienen una serie de vistas asociadas. Es fácil observar que la mayoría de las aplicaciones de Windows que manejamos responden a alguno de estos tres tipos. De estos tres el más sencillo de entender y desarrollar es el Dialog Based.

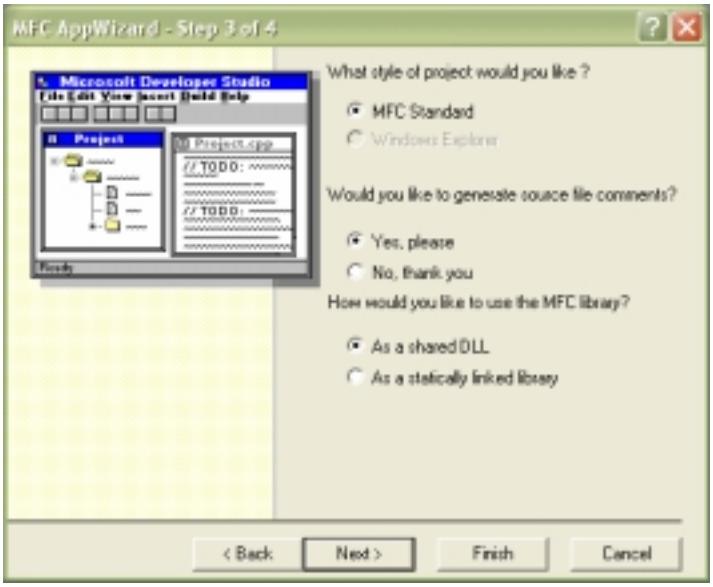
Seleccionamos por tanto **Dialog Based** y como lenguaje de recursos dejaremos por defecto Spanish, salvo que pretendamos que nuestra aplicación se use en Alemania entonces seleccionaríamos German. Presionamos NEXT para pasar a la siguiente pantalla:



Esta pantalla nos pregunta que características le queremos agregar a nuestra aplicación, About Box (un pequeño dialogo que muestra información sobre el programa), Context Sensitive Help (posibilidad de utilizar ficheros de ayuda), 3D Controls (botones y controles con aspecto tridimensional).

Nos aseguramos para este ejemplo, la posibilidad de incluir controles ActiveX (que es la selección por defecto).

Dejamos todo como está, y pasamos a la siguiente ventana pulsando nuevamente NEXT.



Finalmente se nos informa de que se creará un programa con aspecto MFC stándar y además se nos pregunta si deseamos o no que el código fuente generado automáticamente por éste asistente incluya comentarios explicativos y de guía. Obviamente dejamos YES, PLEASE porque esos comentarios son muy útiles para no perderse.

También debemos especificar como queremos usar la librería MFC con el ejecutable: como DLL compartida (dinámica) o como librería estática. Dejaremos la primera opción pues es común en Windows que las aplicaciones funcionen de esta forma, utilizando las librerías MFC que tenga el sistema. Si seleccionamos como librería estática, obtendremos un ejecutable más grande, que contendrá en su interior el código de las MFC por lo que ya no sería necesario distribuir la aplicación con estas librerías.

Si pulsamos NEXT accedemos a la última pantalla de este asistente que nos informa de los nombres de las clases que está a punto de crear. Pulsamos FINISH y se muestra una listado con las características del armazón de aplicación que el asistente ha creado. Pulsamos OK y accedemos por fin al entorno habitual de programación de Visual C++.

El asistente nos generó la base de una aplicación que consta de una ventana con dos botones (aceptar y cancelar) con un texto en el centro y además otra ventana (más pequeña), el Acerca de....

Este "armazón" es una aplicación completa que podemos ejecutar previa compilación y enlazado. Por defecto, la aplicación está en modo de Debug, esto es, se incluye en el código información que permite la ejecución paso a paso del programa, así como la visualización de los valores de las variables y atributos que se usen en el programa. Cuando se finaliza el desarrollo de la aplicación se debe pasar al modo Release, que además de excluir este código adicional, se realizan una sewrie de optimizaciones que reducen el tamaño y aumentan la velocidad del programa finalmente generado.

En el entorno de desarrollo se nos presenta a la izquierda, una ventana del proyecto que tiene tres solapas:

7.2.1 Class View, Resource View y File View.

Class View y File View mantienen el significado de siempre. Sin embargo, por vez primera se utilizará la pestaña de Resource View, la cual mostrará los recursos incluidos en el programa.

Seleccionando la solapa **Class View**, se nos muestran las clases de nuestra aplicación:

- **CAboutDlg**, la clase de la ventana AboutBox.
- **CHolaMundoApp**, la clase aplicación que existe en todas las aplicaciones de windows
- **CHolaMundoDlg**, la clase de la ventana principal (la que contiene los botones Aceptar y Cancelar, y donde luego agregaremos un botón más).

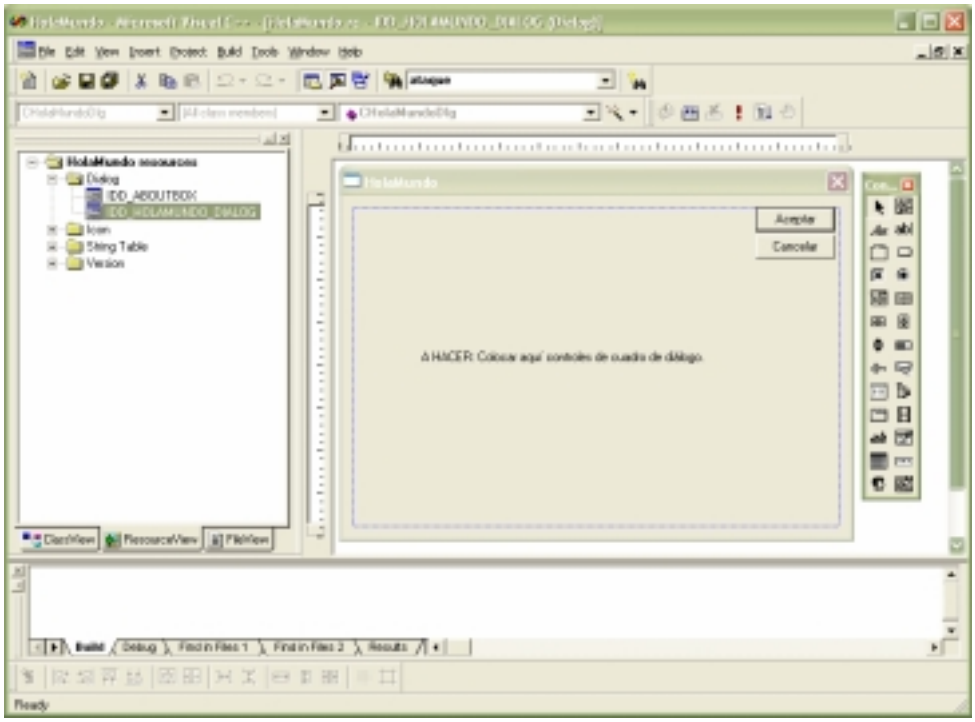
Además hay una carpeta llamada Globals donde se muestran las variables y métodos globales.

Las clases a su vez contienen más cosas: Funciones miembro y variables miembro.

Todo lenguaje de programación visual, sea VB o VC++, contiene una parte del entorno de sesarrollo en l que se puede diseñar el aspecto visual de la aplicación otro en donde se puede escribir el código necesario para los objeto diseñados de manera tal que las cosas respondan de acuerdo a lo que uno quiere.

En VC++ se accede al lugar donde se lleva a cabo el diseño gráfico seleccionando la pestaña RESOURCE (recursos). Al seleccionar RESOURCE se observan varias carpetas **Dialog**, **Icon**, **String Table** y **Version.** Expandiendo la carpeta Dialog se nos muestran los distintos diseños gráficos de las ventanas tipo diálogo de nuestra aplicación. En este ejemplo apaecerán los diseños IDD_ABOUTBOX y IDD_HOLAMUNDO_DIALOG que son las dos ventanas "diálogos" creadas automáticamente por el Asistente de Aplicaciones MFC. Modificaremos IDD_HOLAMUNDO_DIALOG, por lo que hacemos doble click en el mismo y se nos mostrará el diseño de esta ventana con dos botones por defecto "Aceptar" y "Cancelar" listos para ser modificados.

La pantalla que se nos muestra es la siguiente:



Haremos los siguientes pasos:

1. Primero marcamos con el mouse el texto "A HACER:..." y lo eliminamos presionando la tecla suprimir.
2. De la barra de herramientas seleccionamos el tercer icono  y situamos el cuadro de texto estático en algún lugar del cuadro de diálogo.
3. Escribimos el mensaje “Hola Mundo” accediendo con el botón derecho sobre el objeto a las propiedades del mismo. Además de modificar la propiedad *Caption* para poner el mensaje que queramos, se pueden indicar distintos estilos del cuadro de texto. La siguiente imagen muestra como queda el cuadro de texto si se indica que tiene aspecto modal y se centra el texto tanto vertical como horizontalmente. Además se han redimensionado la ventana los botones y el cuadro de texto, y se ha modificado su posición:

4. Compilamos y ejecutamos el programa para ver el resultado… Enhorabuena, has realizado tu primera aplicación sobre windows.

7.3 Ejercicio Práctico

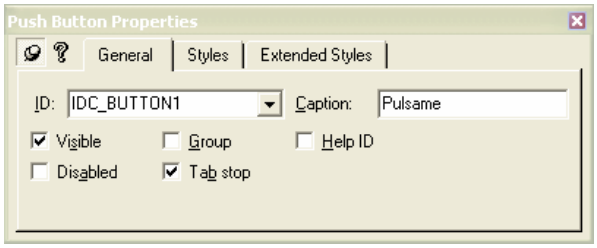
Realizaremos una serie de modificaciones sobre este programa básico que permitirán introducir algunos de los conceptos básicos de Visual C++:

7.3.1 Introducción de un botón adicional

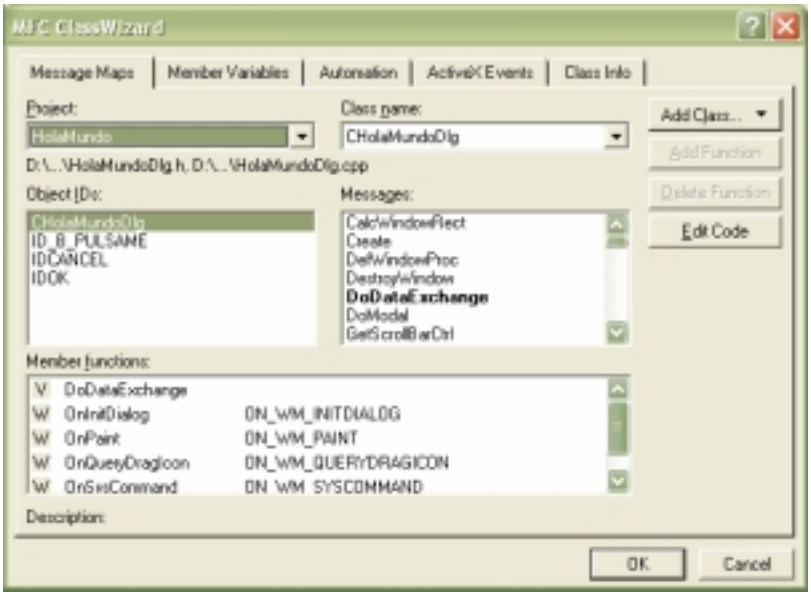
1. Volvemos a la ventana de recursos de nuestra aplicación, y accedemos al diálogo principal.

2. Editamos el aspecto visual de nuestro diálogo, e incluimos un nuevo botón por medio de la barra de herramientas, seleccionando el icono . Mediante las propiedades del botón se modifica su aspecto y el mensaje mostrado de forma que el aspecto final del diálogo debe quedar como sigue.

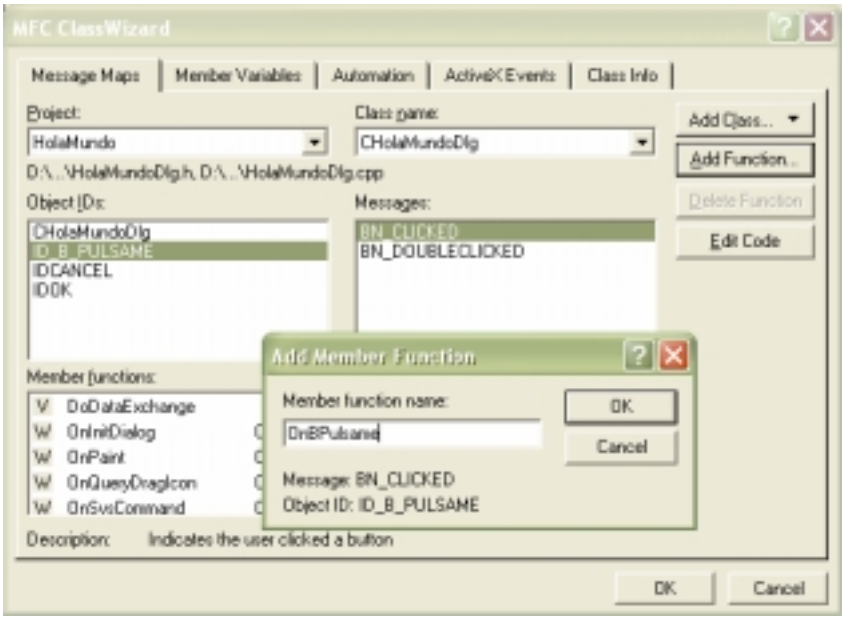
3. Accedemos de nuevo a las propiedades del botón mediante el botón derecho, y le cambiamos la cadena de identificación (ID). Este es el nombre que Visual utilizará para saber a que recurso nos referimos cada vez que queremos mediante código indicar alguna operación sobre el mismo. Siempre pondrá un identificador único por defecto, pero es conveniente cambiarlo para darle un nobre con cierto significado. Le pondremos ID_B_PULSAME. El aspecto entonces de las propiedades generales del botón será:



4. Nuestra idea es que, al pulsar el botón "Pulsame", se muestre el mensaje: "Hola Fulanito", por medio de un cuadro de mensaje. Para esto hay que tener en cuenta que se necesita una función que maneje la acción de "pulsar el botón". En Windows esta función así como muchas otras (mover el mouse, tocar una tecla del teclado, hacer doble click, etc), se denominan "Mensajes" que el sistema operativo transmite a la aplicación. Más o menos, lo que ocurre es que cuando el usuario pulsa el botón, windows nos dice mediante un mensaje que el botón ha sido pulsado, y nos poermite ejecutar una función cuando este *EVENTO* ocurre. Lo que se hará a continuación es capturar este mensaje e indicar que función se va a ejecutar cuando se produzca. Para ello vamos a el *ClassWizard* componente fundamental en la programación visual, que nos permite manejar todos los eventos y variables asociados a controles de Windows. Lo abrimos en *View->ClassWizard* o bien pulsando Ctrl-W, con lo que aparece la siguiente ventana:



Aparecen diversas pestañas, pero de momento seleccionaremos la que hace referencia al mapa de mensajes asociado a cada control. Por defecto, aparecerá seleccionado el ObjectID del diálogo de la aplicación. Obsérvese la cantidad de mensajes distintos que windows transmite a toda la aplicación (Aparecen mostrados en la lista Messages). A cada uno de estos mensajes le podemos indicar una función, o bien dejar que se ejecuten las funciones que por defecto windows tiene previstas. En los ID de objetos aparecen además los identificadores de los botones de Cancel y Ok y de el botón que se ha añadido. Seleccionamos el ID_B_PULSAME, por lo que la ventana de posibles mensajes varía:



Existen dos mensajes posibles para el botón: que se haga Clik o doble Clik. Seleccionamos la pulsación simple, e indicamos que se agregue una función a este mensaje mediante el botón AddFunction… Como consecuencia, un diálogo nos permite especificar el nombre de esta función. Pulsamos el botón de OK, por lo que en la lista de abajo se mostrará una nueva función asociada a un mensaje. Si hacemos doble click sobre la misma, nos llevará a la zona del código de nuestra aplicación en donde se puede especificar el código contenido en la función. Observemos por la cabecera de la función que esta es un método de la clase CHolaMundoDlg.

5. Escribimos el siguiente código:

```
void CHolaMundoDlg::OnBPulsame()
{
  // TODO: Add your control notification handler code here
  MessageBox("Hola Fulanito", "Mi Mensaje", MB_ICONINFORMATION);
}
```

Esto genera un dialogo de mensaje con el título “Mi Mensaje” y el mensaje “Hola Fulanito”, además se indica que aparezca en la ventana el icono de Información. Los parámetros de la función MessageBox son: Texto a mostrar, Título del mensaje e icono + botones a mostrar (en este caso como no se especifican los botones muestra por defecto OK). La siguiente tabla muestra otras posibles opciones para el MessageBox:

Botones posibles	*Iconos posibles*
MB_ABORTRETRYIGNORE	**MB_ICONEXCLAMATION**
MB_OK	**MB_ICONINFORMATION**
MB_OKCANCEL	**MB_ICONQUESTION**
MB_RETRYCANCEL	**MB_ICONSTOP**
MB_YESNO	
MB_YESNOCANCEL	

Si quisieramos una ventanita con tres botones (YES, NO, CACEL) y con el icono de STOP, pondríamos como tercer argumento lo siguiente: MB_YESNOCANCEL|MB_ICONSTOP.

6. Compilamos y vemos el resultado.

7.3.2 Cambiar la acción asociada a Cerrar la ventana del diálogo

Esta es una acción peligrosa, puesto que podemos deshabilitar la posibilidad de cerrar normalmente la aplicación. Lo que se va a hacer es pedir al usuario una confirmación de que realmente desea cerrar la aplicación:

1. Accedemos al ClassWizard, y seleccionamos el identificador correspondiente al diálogo. De entre los mensajes posibles, agregamos una función al correspondiente a WM_CLOSE, que es el mensaje de notificación de que la aplicación se va a cerrar.

2. En el código de la función escribimos lo siguiente:

```
void CHolaMundoDlg::OnClose()
{
    // TODO: Add your message handler code here and/or call default
    if(MessageBox("¿Desea Cerrar la aplicacion?",
          "Confirmación",MB_YESNO|MB_ICONEXCLAMATION)==IDYES)
          CDialog::OnClose();
}
```

De esta forma, estamos diciendo que si se pulsa el botón de YES, entonces se llame al método de cerrar el diálogo de la clase CDialog, de la cual nuestro diálogo ha sido heredado.

7.3.3 Abrir un fichero de video y mostrarlo.

Para finalizar, y por mostrar la gran potencia que nos da la programación sobre Windows con Visual C++, vamos a realizar en unos pocos pasos la posibilidad de abrir ficheros multimedia mediante la inclusión del visor de Windows Media Placer en nuestro diálogo. Para ello vamos a incluir dentro del entorno de programación este objeto como un elemento más agregable a la aplicación.

1. Activamos el comando de menú Proyect->Add to Proyect->Components and Controls. Nos sale un diálogo que nos permite buscar el tipo de componente que queremos añadir. Accedemos al directorio de Registered ActiveX Components, y dentro del mismo buscamos el Windows Media Placer. Le damos a Insertar, con lo que el sistema nos informa de todas las clases que va a ser necesario crear para poder utilizar este objeto. De nuevo aceptamos, y cerramos el diálogo. Ahora en la vista de las clases del proyecto vemos que se han agregado un montón.

2. Nos vamos a la edición gráfica de nuestro diálogo (Recursos), y observamos que en la barra de herramientas aparece el icono de Windows Media Placer como un objeto más que se puede añadir. Lo agregamos al diálogo, quedándonos un aspecto como el siguiente:

3. Modificamos sus propiedades –ya algo más complejas- y le cambiamos el identificador a un nombre que nos sea más accesible como por ejemplo ID_WINDOWSMEDIA. Podemos mostrar solo la ventana de vídeo si indicamos en las propiedades que no queremos que se muestre ninguno de los controles, con lo que el usuario no percibe que estamos utilizando el Windows media placer.

4. Vamos al *ClassWizard*, y le asociamos una variable de forma que podamos manejar el control desde nuestro programa. Esto lo haremos por medio de la pestaña de Member Variables. Seleccionamos el Identificador de el visualizador de Windows Media Placer y le decimos agregar una variable con el nombre m_WindowsMedia. Aceptamos, por lo que a partir de ahora, en el código de nuestro programa podemos manejar el visualizador como un objeto más con una serie de métodos y atributos.

5. Vamos al código de la función asociada al botón pulsame, y escribimos lo siguiente:

```
#include "wmpcontrols.h"
void CHolaMundoDlg::OnBPulsame()
{
	// TODO: Add your control notification handler code here
	MessageBox("Hola Fulanito: Mira un reloj", "Mi Mensaje",
		MB_ICONINFORMATION);
	m_WindowsMedia.SetUrl("C:\\Windows\\clock.avi");
	m_WindowsMedia.GetControls().play();
}
```

6. Compilamos y observamos el resultado… ¿Sorprendente verdad?.

7. Si aún queda tiempo, se propone el que al pulsar el botón “Pulsame”, se pida el usuario seleccionar el fichero que se desea abrir con Windows Media Placer por medio de un CFileDialog, tal y como se mostró en la tercera práctica.

7.4 Ejercicios Propuestos

Para aprender a manejar los distintos objetos de windows hay que pegarse un poco con ellos y ver sus distintas peculiaridades. Por ello, se recomienda intentar añadir y manejar cada uno de los elementos básicos que aparecen por defecto en la barra de herramientas de el diseño de diálogos.

Se recomienda vivamente la lectura o utilización del libro titulado “Aprenda Visual C++ en 21 días” como medio de introducirse a la programación en Visual. Este libro está en html en internet, tanto en ingles como en castellano.