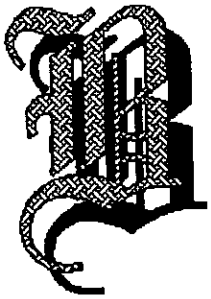
APENDICE



Manuales de usuario de CompFrac y CoDec Fractal

El contenido de este Apéndice consta de los Manuales de Usuario de las Aplicaciones *CompFrac* y *CoDec Fractal*, incluyendo una sección para la Descripción del Funcionamiento de CompFrac en base a su Código Fuente. Se tratará de ser lo más simple posible en las descripciones y análisis, sin embargo, se manejan conceptos y expresiones técnicas necesarias para describir propiamente las ideas y planteamientos que se manejan en el Código de cada programa. Por tanto se requiere un mínimo de conocimientos sobre Programación Estructurada, un tanto mas sobre Programación Orientada a Objetos en lenguaje C (es decir, C++) y sobre el manejo del Sistema de Desarrollo utilizado para implementar estas Aplicaciones : *Microsoft© Visual C++*.

Para el caso de *CoDec Fractal*, la descripción del código fuente con el cual se desarrolló no es demasiado importante desde el punto de vista de nuestro trabajo, pues en realidad las descripciones de las funciones de librería del Apéndice D, a partir de las cuales *CoDec Fractal* existe, proporcionan todo lo necesario para permitir desarrollar cualquier Aplicación (en cualquier compilador, y hasta compatibilizando con cualquier lenguaje de programación de uso general), de que utilice las facilidades de la Compresión Fractal de Imágenes.

Acorde a esta aplicación de Compresión Fractal de Imágenes, se define el formato **FIF**, del cual se habla en el Capítulo **[2]**.

MANUAL DE USUARIO DE *COMPFRAC* V. 1.0.0

Antes de proceder a describir el uso de CompFrac, se muestran los requerimientos del Sistema que CompFrac necesita como mínimo para tener un buen funcionamiento.

REQUERIMIENTOS

CompFrac es una Aplicación de Compresión/Descompresión de Archivos de Imágenes, basado en algoritmos Fractales. *CompFrac* se desarrolló en Microsoft© Visual C++ 4.0, bajo el ambiente Windows© 95. Cómo mínimo se requiere de una PC con un procesador 80386 (que soporte Windows© 95), o superior. Los restantes requerimientos de Hardware son :

- 8 MB en RAM. Se recomienda tener 16 MB.
- Tarjeta de video VGA o compatible. Se recomienda tener una super VGA.
- Espacio en Disco Duro de 5 Mb. Incluyendo las imágenes ejemplo que contiene CompFrac.
- Teclado, para usuarios con experiencia en el manejo de Aplicaciones sin ratón. Se recomienda utilizar ratón.

NOTA : No hay condiciones para las dimensiones disponibles para la Pantalla (puede ser desde 640x480 pixeles y mayores), con una paleta de colores mínima de 4 bits (16 colores); sin embargo, para mejores resultados la cantidad de colores a utilizar se aconseja sea de 16 bits, es decir una paleta de colores de alta densidad (65536 colores), o mayor.

INSTALACIÓN DE COMPFRAC

La instalación de CompFrac es sencilla. Para iniciar debe ejecutarse el programa de instalación contenido en el Disco 1, que acompaña a este trabajo, ya sea desde el Explorador de Windows o desde la opción "Ejecutar..." del menú Inicio de la Barra de Tareas. Y finalmente seguir la indicaciones del programa de instalación.

CÓMO TRABAJAR CON COMPFRAC

A continuación se describirá de una manera sencilla cómo se utiliza *CompFrac* para Codificar y Decodificar un archivo correspondiente a una imagen.

CompFrac se incluye dentro de los Discos que acompañan a este Trabajo. Para proceder a la

puesta en ejecución de este Programa refiérase a la sección anterior.

Una vez ubicado instalado *CompFrac* se procederá a su ejecución ya sea a través del Menú Programas ubicado dentro del Menú del botón INICIO de Windows© 95, y accesando al Grupo creado por la propia instalación de CompFrac, o mediante la opción "Ejecutar...", así mismo se puede poner en ejecución a *CompFrac*, desde el Explorador de Windows, accesando directamente al directorio donde reside *CompFrac*, en el Disco Duro.

La primer pantalla que muestra CompFrac es la ventana de la Figura B.1, la cual describe los Créditos sobre la Aplicación. Para continuar, sólo es necesario presionar el botón Aceptar.

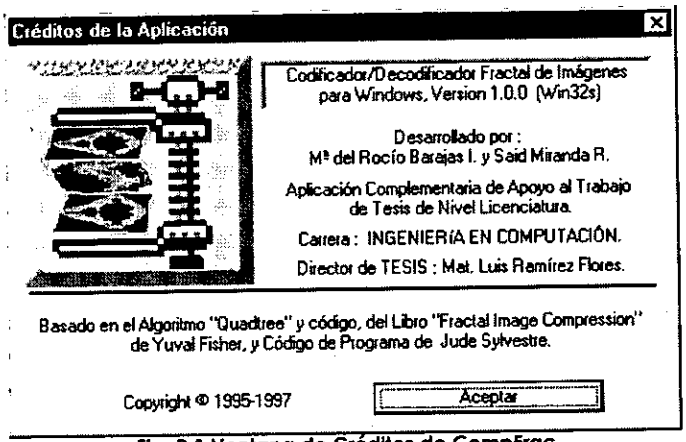


Fig. B.1 Ventana de Créditos de *CompFrac*.

A continuación se muestra la pantalla principal de *CompFrac*, desde la cual se podrán realizar las Codificaciones y Decodificaciones de Archivos. La ventana se divide en varias secciones, la sección superior sólo es la Descripción de la Aplicación. La siguiente sección delimitada por el rectángulo llamado Operación muestra mensajes sobre la acción necesaria para comenzar (o para proseguir, según sea el caso) con la ejecución de *CompFrac*, por ejemplo, el primer mensaje que muestra *CompFrac* al iniciar su ejecución es "INFO: Selecciona la Ruta y Nombre de Archivo mediante el Botón 'Abrir Archivo' ", además de que existe un seguimiento sobre el foco (señalización para responder al teclado) en los botones. Para este caso inicial de *CompFrac*, el foco está en el botón Abrir Archivo. Finalmente en la parte inferior se muestra un rectángulo denominado Estado, el cual presentará los resultados de la Codificación o Decodificación de un Archivo.

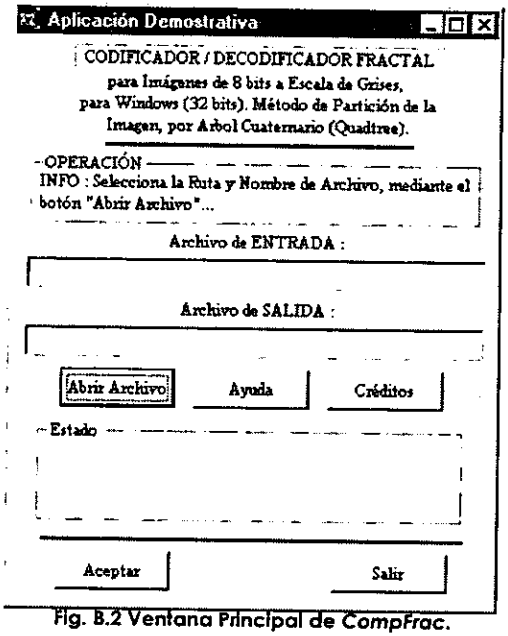


Fig. B.2 Ventana Principal de *CompFrac*.

En este momento se pueden elegir varias opciones, de acuerdo a los botones de la pantalla, estas opciones son Abrir un Archivo, Consultar la Ayuda[1], Abrir los Créditos de la Aplicación o Terminar el Programa (Salir).

Al presionar el botón Abrir Archivo, la pantalla siguiente que aparece es la ventana "Abrir", desde la cual se permitirá seleccionar un archivo de entre los siguientes tipos :

DAT / RAW, Archivos cuyo contenido es una imagen representada en forma de bytes consecutivos en bruto, en Escala de Grises, con profundidad de 8 bits.

FIC / TRN, Archivos de código Fractal. El contenido son varios datos necesarios y los coeficientes de la Transformada Iterada por Particiones, la cual codifica a la imagen.

CompFrac determina (de acuerdo a la extensión del archivo seleccionado) si lo que se desea es Codificar o Decodificar un archivo, para lo cual procede a mostrar la pantalla correspondiente a las opciones de Compresión (para el caso de archivos DAT / RAW) o Descompresión (para el caso de archivos FIC / TRN).

Una vez seleccionado el Archivo, presionar el botón Abrir para que *CompFrac* continúe con el trabajo.

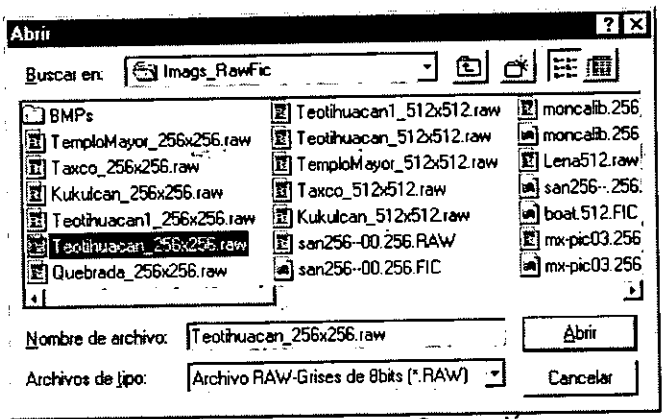


Fig. B.3 Ventana Abrir Archivo para Compresión.

Es importante aclarar que la extensión de Archivo es una de las varias formas para determinar o especificar el contenido de un archivo, sin embargo existen muchas otras Aplicaciones que manejan archivos con las mismas extensiones, y que sin embargo su contenido es totalmente diferente. Otro punto importante a establecer es que los archivos FIC y TRN, no son dos formatos diferentes, si no que son dos extensiones que se refieren al mismo formato de archivo. Lo mismo ocurre con las extensiones RAW y DAT. Esta situación se presenta por cuestiones de compatibilidad con las Aplicaciones que se hubieran desarrollado basándose exclusivamente en las extensiones que maneja el Código Base de *Yuval Fisher*, estas extensiones son TRN y DAT.

Codificación De Un Archivo

Ahora, de acuerdo a la Figura B.3 se ha seleccionado un Archivo RAW, y entonces *CompFrac* presentará la ventana "Opciones de Compresión" (ver Figura B.5), la cual contiene los parámetros que permiten modificar el proceso de Codificación de la Imagen, repercutiendo esto principalmente en el índice de Compresión y la Calidad de la Imagen Codificada, en formato compactado. Sin embargo, existen valores por *Default* de todos estos parámetros y que fueron elegidos en tal forma que permitan, en general, optimizar la codificación de la mayoría de imágenes.

Más adelante, en la sección de Descripción del Funcionamiento de CompFrac, se detallará sobre cada uno de estos parámetros. Por ahora sólo se especificarán los valores de estos parámetros que provocan variaciones importantes en el tamaño del archivo Compactado, y el tiempo de Codificación. Para el caso del proceso de Descompresión, se hablará de los valores de cada parámetro de Decodificación en conjunto con los de la Codificación lograda y que alteran la Calidad Visual del Archivo recuperado, con respecto a la Imagen Original.

Estos resultados se obtuvieron al ejecutar a CompFrac sobre la Imagen Original de Ejemplo (ver Figura B.4), en una PC 486DX4 a 100Mhz, con 12Mb en RAM, Windows © 95 OSR2, 100Mb de espacio

en Disco (para los posibles Archivos temporales de Windows) y los mínimos programas residentes que Windows requiere para mantener el Ambiente de operación funcionando.

#	Parámetro	Valor	Resultado		
			Proporción de Compresión	Error RMS	Tiempo Aprox.
1	Dimensiones de la Imagen Cuadrada	256	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	55 seg.
		512	(262144 / 11954)bytes ⇒ 21.93 : 1	29.9	125 seg.
2	Tolerancia RMS	1	(65536 / 12962) bytes ⇒ 5.06 : 1	15.4	59 seg.
		8 (Por Omisión)	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	55 seg.
		15	(65536 / 11204) bytes ⇒ 5.85 : 1	15.4	52 seg.
3	Bits de Escalamiento	4	(65536 / 11436) bytes ⇒ 5.73 : 1	15.4	54 seg.
		5 (Por Omisión)	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	55 seg.
		6	(65536 / 12401) bytes ⇒ 5.28 : 1	15.4	55 seg.
4	Bits de Desplazamiento	6	(65536 / 11451) bytes ⇒ 5.72 : 1	15.4	54 seg.
		7 (Por Omisión)	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	55 seg.
		8	(65536 / 12394) bytes ⇒ 5.29 : 1	15.4	54 seg.
5	Factor de Escala Máxima	1.0 (Por Omisión)	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	55 seg.
		2.0	(65536 / 11908) bytes ⇒ 5.50 : 1	15.4	55 seg.
6	Mínima Profundidad de Recursión	3	(65536 / 11930) bytes ⇒ 5.49 : 1	15.4	56 seg.
		4 (Por Omisión)	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	55 seg.
		5	(65536 / 12027) bytes ⇒ 5.45 : 1	15.4	54 seg.
7	Máxima Profundidad de Recursión	5	(65536 / 2822) bytes ⇒ 23.22 : 1	41.7	16 seg.
		6 (Por Omisión)	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	55 seg.
		7	(65536 / 47611) bytes ⇒ 1.38 : 1	2.0	312 seg.

Tabla B.1 Modificación de los Parámetros de Compresión y sus resultados.

8	Tipo de Contenedor de Dominios	0 (Por Omisión)	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	55 seg.
		1	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	55 seg.
		2	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	72 seg.
9	Tamaño del Intervalo del Contenedor de Dominios	1 (Por Omisión)	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	55 seg.
		2	(65536 / 12857) bytes ⇒ 5.10 : 1	15.4	72 seg.
		3	(65536 / 13781) bytes ⇒ 4.76 : 1	11.8	134 seg.
10	Intervalo de Dominio como Múltiplo	Si	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	54 seg.
		No (Por Omisión)	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	55 seg.
11	24 Clases de Dominio	Si	(65536 / 11916) bytes ⇒ 5.50 : 1	14.0	177 seg.
		No (Por Omisión)	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	55 seg.
12	3 Clases de Dominio	Si	(65536 / 11920) bytes ⇒ 5.50 : 1	15.4	64 seg.
		No (Por Omisión)	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	55 seg.
13	Sólo Escalamiento Positivo	Si	(65536 / 11916) bytes ⇒ 5.50 : 1	15.4	51 seg.
		No (Por Omisión)	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	55 seg.
14	Mostrar Proceso de Particionado	Si (Por Omisión)	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	55 seg.
		No	(65536 / 11922) bytes ⇒ 5.50 : 1	15.4	12 seg.

Tabla B.1 Modificación de los Parámetros de Compresión y sus resultados (continuación).

Fig. B.4 Imagen Original de 256x256 pixeles, utilizada como Ejemplo.

Los resultados mostrados se obtuvieron manteniendo todos los Parámetros en sus valores por omi-

sión, excepto el valor actual que sufre la modificación.

Regresando al ejemplo de Codificación, una vez modificados los parámetros de Compresión presionar el botón Aceptar para continuar. Si se desea restablecer los valores por Omisión de todos los parámetros, presionar el botón Restaurar Valores. Si se desea obtener ayuda sobre las opciones de Compresión mostradas en esta ventana, presionar el botón Ayuda o la tecla F1. Si lo que se requiere es seleccionar otro archivo diferente al actual será necesario presionar el botón Cancelar y posteriormente en la pantalla principal de *CompFrac*, nuevamente presionar el botón Abrir Archivo.

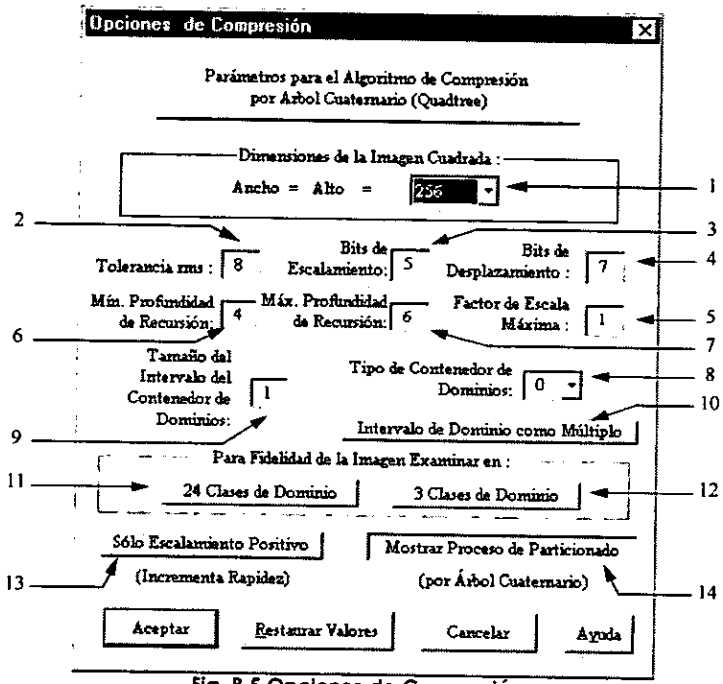


Fig. B.5 Opciones de Compresión.

Continuando con la Codificación del Archivo RAW y una vez que se haya presionado el botón Aceptar de la ventana "Opciones de Compresión", *CompFrac* regresa a su ventana principal y muestra la ruta completa del Archivo Abierto (Archivo de Entrada:), así como la ruta completa y el nombre propuesto para el Archivo Compactado resultante (Archivo de Salida:). Si se desea modificar el Nombre propuesto para el Archivo de Salida y/o la ruta donde se ha de guardar el Archivo, puede editarse directamente la ruta sobre el cuadro de texto etiquetado como Archivo de SALIDA: . Sin embargo la ruta que fuera establecida debe existir ya, pues en caso contrario en el cuadro inferior denominado Estado *CompFrac* notificará que no fue posible abrir el Archivo de Salida.

Hasta este momento ya se ha indicado el Archivo de Entrada a *CompFrac* para su Codificación y se han aceptado los valores de los parámetros de Compresión; si la ruta y nombre propuesto para el Archivo de Salida son apropiados, entonces sólo resta presionar el botón Aceptar para que *CompFrac* comience el proceso de Codificación del Archivo RAW.

Una vez comenzado el proceso de codificación no podrá ser cancelado sino hasta haber concluido la Compresión del Archivo. Inicialmente se cierra la ventana principal de CompFrac, dando paso a la ventana (ver Figura B.6) que muestra el Proceso de Particionado de la Imagen (esto ocurrirá sólo si el botón 14 de la Figura B.5 está activado), en este caso la barra de titulo de esta ventana se encarga de informar sobre las acciones principales que *CompFrac* realiza al iniciar y finalizar el proceso de Compresión, además de mostrar durante el desarrollo del proceso de Partición Visual de la Imagen, la Profundidad alcanzada en el Árbol Cuaternario de acuerdo al análisis de cada fracción de la Imagen.

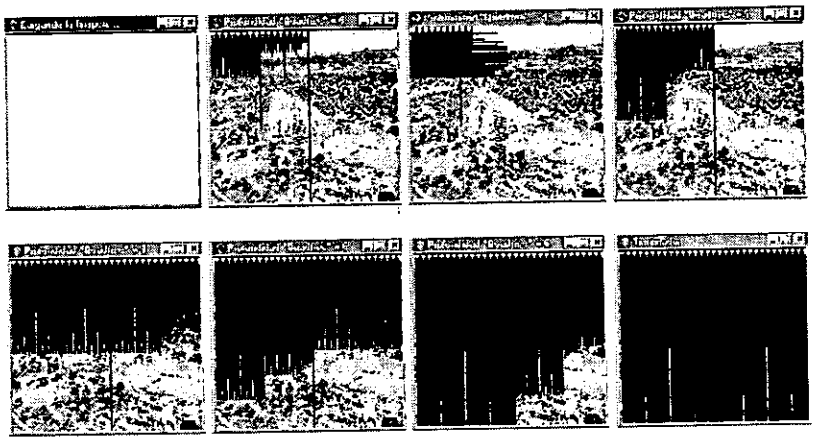


Fig. B.6 Desarrollo Visual de la Partición de la Imagen para Comprimirla.

Finalmente concluye el proceso con un mensaje que indica que la Operación de Compresión ha sido Finalizada en forma Correcta, al cual se responderá presionando el botón Aceptar. Ahora en el cuadro de Estado de la Pantalla principal de *CompFrac* aparecen los resultados de la Compresión realizada, cuya información es la siguiente :

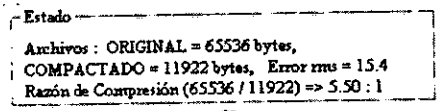


Fig. B.7 Resultados de la Compresión Fractal.

En esta pantalla *CompFrac* informa sobre el tamaño, en bytes, de los Archivos de la Imagen Original y la ahora Compactada, muestra además el error *RMS* que es un valor de estimación que permite

determinar cuán cerca se ha podido reproducir o aproximar la Imagen Compactada a la Imagen Original y finalmente se muestra el índice de Compresión logrado en este proceso de Codificación.

Decodificación

Ahora se explicará el proceso contrario de la Codificación, que consta de recuperar la Imagen en el formato basado en pixeles (en bruto), a partir de la Imagen Fractal contenida en el Archivo Compactado FIC / TRN.

Para realizar la descripción de la Decodificación, se iniciará a partir de la ventana Principal de *CompFrac*, ya sea que se haya realizado algún proceso de Compresión anterior o se haya iniciado CompFrac nuevamente.

Para comenzar, primero es necesario indicar a *CompFrac* cuál será el Archivo Compactado con el que se trabajará. Para ello es necesario presionar el Botón Abrir Archivo, el cual abrirá la ventana "Abrir" desde la cual se seleccionará el archivo FIC / TRN. Para ejemplificar el uso del Decodificador se seleccionará el Archivo FIC que *CompFrac* creó durante el proceso de Codificación (ver Sección anterior), el cual se muestra seleccionado en la ventana de la Figura B.8.

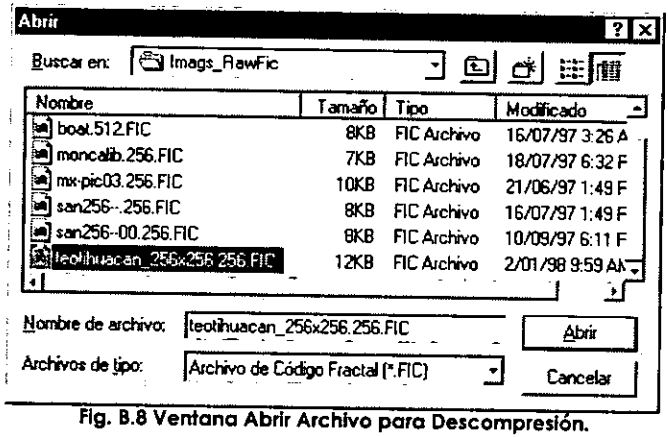


Fig. B.8 Ventana Abrir Archivo para Descompresión.

Una vez seleccionado el Archivo FIC, presionar el botón Abrir. *CompFrac* mostrará la ventana de parámetros de Descompresión (ver Figura B.9). Una descripción de cada parámetro se presenta en las secciones posteriores, por ahora se proporciona una tabla (dividida en dos), que contiene los resultados del tiempo de Proceso (en segundos), como consecuencia de la variación de los parámetros de Descompresión, en conjunto con las variaciones de los parámetros de Compresión. Estas tablas pueden ser muy útiles a la hora de realizar compresiones que deban cumplir, o se aproximen, con algunas condiciones de Tamaño y/o Calidad de la Imagen, necesarias para su manejo y manipulación.

Las diversas áreas de la tabla designan la Calidad de la Imagen resultante de acuerdo a la siguiente

escala:

Calidad buena / optima.

La imagen Decodificada se ve muy distorcionada.

Mala Calidad de la Imagen Recuperada.

Ocurre un error Inesperado en la ejecución de CompFrac.

Los valores de los Parámetros que cuentan con un asterisco (*) son los valores por Omisión que CompFrac asigna automáticamente o que son recuperados nuevamente al presionar el Botón Restaurar Valores, ya sea dentro del proceso de Compresión o Descompresión.

	RAW ⇒ FIC		1		2			3			4			5		6			7		
FIC ⇓ RAW	1. COD ⇒ 2. DECO ⇓		Dimensiones de la Imagen Cuadrada		Tolerancia RMS			Bits de Escalamiento			Bits de Desplazamiento			Factor de Escala Máxima		Máxima Profundidad de Recursión			Máxima Profundidad de Recursión		
			256*	512	1	8*	15	4	5*	6	6	7*	8	1.0*	2.0	3	4*	5	5	6*	7
1	Núm. de Iteraciones	5	16	55	15	14	16	15	14	15	15	13	15	14	14	14	15	15	12	15	16
		10*	25	95	25	25	26	31	26	29	29	25	27	24	24	24	24	24	25	24	25
2	Escala de la Imagen a Decodificar	1*	25	95	25	25	26	31	26	29	29	25	27	24	24	24	24	24	25	24	25
		2	84	345	89	77	88	96	89	109	98	97	100	96	85	97	90	90	92	94	101
3	Bits de Escalamiento	4	32	113	32	28	30	32	31	28	29	27	28	30	28	28	29	28	30	27	30
		5*	25	95	25	25	26	31	26	29	29	25	27	24	24	24	24	24	25	24	25
		6	25	91	25	26	25	25	26	28	27	26	25	28	26	29	26	25	30	25	31
4	Bits de Desplazamiento	6	26	115	26	29	25	24	26	28	23	28	27	25	26	28	29	27	25	27	25
		7*	25	95	25	25	26	31	26	29	29	25	27	24	24	24	24	24	25	24	25
		8	36	98	26	25	26	26	25	26	27	25	23	26	27	33	25	27	26	26	35
5	Factor de Escala Máxima	1.0*	25	95	25	25	26	31	26	29	29	25	27	24	24	24	24	24	25	24	25
		2.0	26	101	28	26	27	28	27	27	28	28	31	28	26	28	29	29	26	28	29
6	Post Procesamiento	Si*	25	95	25	25	26	31	26	29	29	25	27	24	24	24	24	24	25	24	25
		No	23	79	23	23	24	25	23	25	26	21	26	23	24	24	23	23	23	24	25
7	Mostrar Proceso de Partición	Si*	25	95	25	25	26	31	26	29	29	25	27	24	24	24	24	24	25	24	25
		No	4	12	5	4	4	3	5	4	3	4	4	4	4	4	4	4	3	3	6

Tabla B.2a Tiempos de Decodificación para varios valores de los Parámetros de Descompresión, en conjunto con los Parámetros de Compresión.

	RAW ⇒ FIC :		8			9			10		11		12		13		14	
FIC ⇓ RAW	1. COD. ⇒ 2. DECO ⇓		Tipo de Contenedor de Dominios			Tamaño del Intervalo del Contenedor de Dominios			Intervalo de Dominio como Múltiplo		24 Clases de Dominio		3 Clases de Dominio		Sólo Escalamiento Positivo		Mostrar Proceso de Particionado	
			0*	1	2	1*	2	3	SI	No*	SI	No*	SI	No*	SI	No*	SI*	No
1	Núm. de Iteraciones	5	15	15	14	14	15	17	15	14	15	14	14	15	14	15	14	15
		10*	24	25	24	25	25	25	26	25	24	25	25	26	26	25	24	25
2	Escala de la Imagen a Decodificar	1*	24	25	24	25	25	25	26	25	24	25	25	26	26	25	24	25
		2	83	80	85	83	82	82	85	83	82	83	83	83	82	83	83	86
3	Bits de Escalamiento	4	27	29	28	26	F	F	31	29	27	27	30	27	30	27	27	27
		5*	24	25	24	25	25	25	26	25	24	25	25	26	26	25	24	25
		6	27	29	28	27	F	F	29	28	27	27	28	27	28	27	27	27
4	Bits de Desplazamiento	6	27	28	27	27	F	F	29	27	29	27	27	27	29	27	27	28
		7*	24	25	24	25	25	25	26	25	24	25	25	26	26	25	24	25
		8	27	27	27	27	F	F	29	27	27	27	26	27	28	27	27	30
5	Factor de Escala Máxima	1.0*	24	25	24	25	25	25	26	25	24	25	25	26	26	25	24	25
		2.0	27	26	27	27	29	28	27	27	29	27	27	27	23	27	27	26
6	Post Procesamiento	Si*	24	25	24	25	25	25	26	25	24	25	25	26	26	25	24	25
		No	22	23	27	23	24	25	24	23	22	23	24	24	24	23	22	23
7	Mostrar Proceso de Partición	Si*	24	25	24	25	25	25	26	25	24	25	25	26	26	25	24	25
		No	4	4	5	4	4	4	5	4	5	4	4	4	4	4	4	4

Tabla B.2b Tiempos de Decodificación para varios valores de los Parámetros de Descompresión, en conjunto con los Parámetros de Compresión.

Siempre que sea posible se recomienda utilizar los valores por Omisión de los Parámetros o realizar alguna modificación pequeña en base a estos mismos valores.

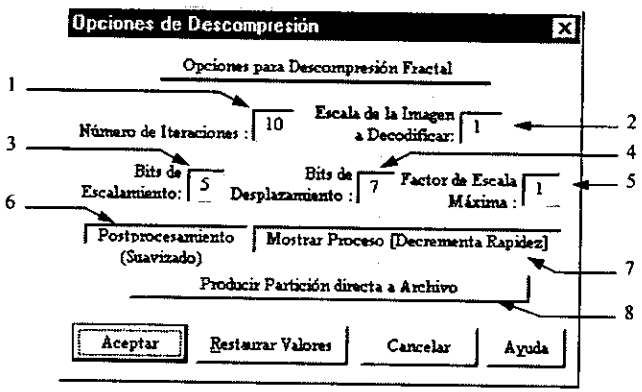
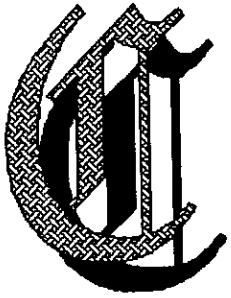
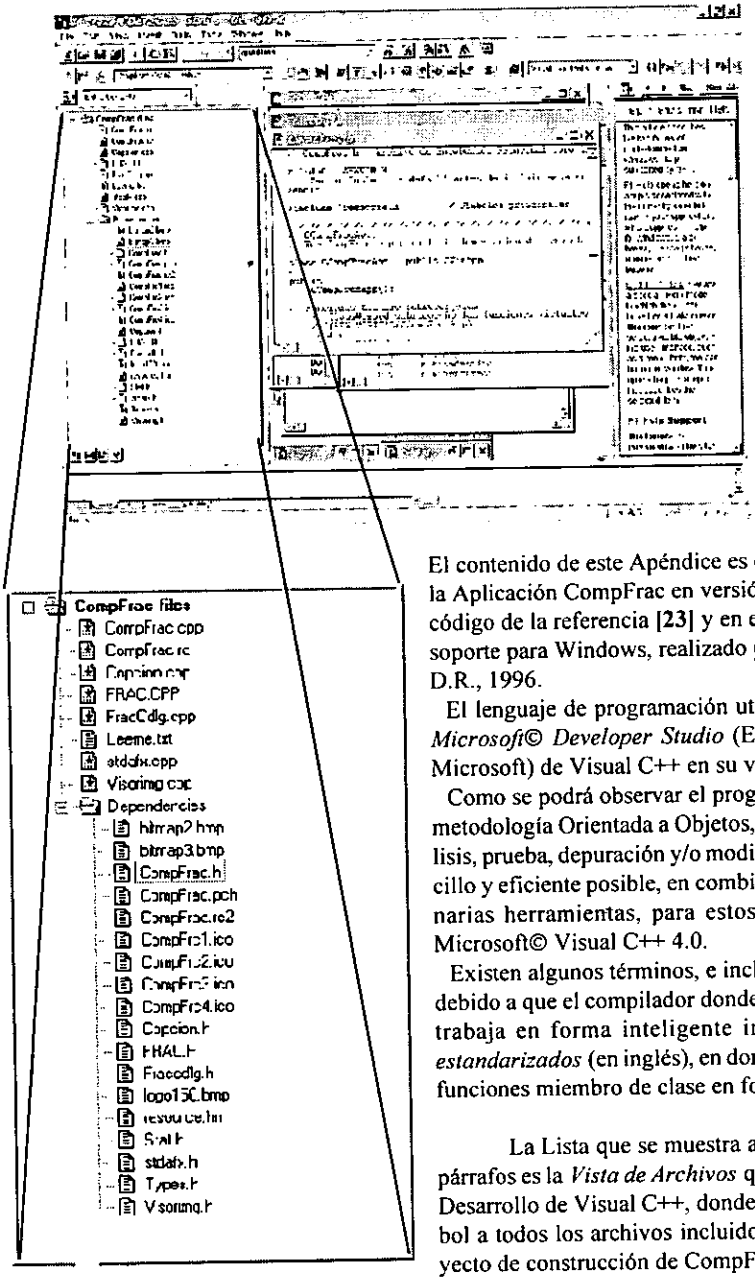


Fig. B.9 Opciones de Descompresión.

APENDICE



Código Fuente de CompFrac v1.0.0



El contenido de este Apéndice es el código completo de la Aplicación CompFrac en versión 1.0.0, basada en el código de la referencia [23] y en el código básico con soporte para Windows, realizado por Jude Sylvestre, © D.R., 1996.

El lenguaje de programación utilizado es C++, sobre el *Microsoft© Developer Studio* (Estudio de Desarrollo de Microsoft) de Visual C++ en su versión 4.0, en inglés.

Como se podrá observar el programa fue construido con metodología Orientada a Objetos, de tal forma que su análisis, prueba, depuración y/o modificación sean lo más sencillo y eficiente posible, en combinación con las extraordinarias herramientas, para estos fines, con que cuenta Microsoft© Visual C++ 4.0.

Existen algunos términos, e inclusive párrafos en inglés, debido a que el compilador donde se desarrolló CompFrac trabaja en forma inteligente incluyendo comentarios *estandarizados* (en inglés), en donde sea necesario, al crear funciones miembro de clase en forma automática.

La Lista que se muestra al lado izquierdo de estos párrafos es la *Vista de Archivos* que muestra el Sistema de Desarrollo de Visual C++, donde conjunta en forma de árbol a todos los archivos incluidos necesarios para el proyecto de construcción de CompFrac.

El texto del código fuente fue incorporado lo más cercanamente posible al código en texto plano, sin embargo pueden existir variaciones mínimas; como por ejemplo, las cadenas de caracteres o comentarios de código con los símbolos //, que sobrepasan la longitud de los renglones de este documento simplemente son recortados por el procesador de textos y pasados al siguiente renglón, sin embargo dentro del código fuente deben ir en forma continua para que el compilador reconozca dichas cadenas completas. Existen comentarios, sugerencias, notas y objetivos sobre las clases creadas, sus funciones (miembros de clase), variables y demás elementos del código.

Archivo LEEME.TXT

```
1.  ========================================================================
2.  MICROSOFT FOUNDATION CLASS LIBRARY :  COMPFRAC
3.  LIBRERÍA DE CLASES FUNDAMENTALES DE MICROSOFT : COMPFRAC
4.  ========================================================================
5.  ************************************************************************
6.  (Este archivo es generado por AppWizard de Visual C++ 2.0 ó 4.0, la traducción al Español y edición de la última
7.  sección, denominada DESCRIPCIÓN DE CLASES, fue realizada por Rocio Barajas Ibáñez)
8.  ************************************************************************
9.  AppWizard ha  creado esta Aplicación COMPFRAC por Ud.
10. Esta aplicación no sólo demuestra lo básico de la utilización de las Clases Fundamentales de Microsoft sino que es
11. también el punto inicial de la escritura de su propia aplicación.
12. Este archivo contiene un compendio de lo que Ud. encontrará en cada Archivo que compone a su Aplicación
13. COMPFRAC.
14.
15. COMPFRAC40.MAK
16. Este archivo de proyecto es compatible con el ambiente de desarrollo de Visual C++ ver 4.0.
17. Es también compatible con el programa NMAKE que está contenido en Visual C++.
18. Para compilar la versión DEBUG (prueba y corrección de errores) del programa desde
19. el prompt del DOS, escriba
20.
21. nmake /f COMPFRAC.MAK CFG="Win32 Debug"
22.
23. o para compilar la versión RELEASE (REVISION-VERSION FINAL) del programa, escriba
24.
25. nmake /f COMPFRAC.MAK CFG="Win32 Release"
26.
27. COMPFRAC.H
28. Este es el archivo de encabezado principal para la Aplicación  Incluye a otros encabezados específicos del proyecto
29. (incluyendo RESOURCE.H) y declara la clase de la aplicación CCompFracApp.
30.
31. COMPFRAC.CPP
32. Este es el archivo principal de código fuente de la aplicación y contiene la implementación de la clase CCompFracApp.
33.
34. COMPFRAC.RC
35. Este archivo contiene un listado de todos los recursos de Microsoft Windows que el programa utiliza. Incluye los iconos,
36. bitmaps, y cursores que son almacenados en el subdirectorio /RES.  Este archivo puede ser directamente editado
37. dentro del Estudio del ambiente de desarrollo de Visual C++.
38.
39. RES\COMPFRC[1-4].ICO
40. Estos archivos son archivos de iconos, los cuales son usados como iconos de la aplicación.  Estos iconos son incluidos
41. mediante el archivo de recursos principal COMPFRAC.RC.
42.
43. RES\COMPFRAC.RC2
44. Este archivo contiene recursos que no son editados por el ambiente de desarrollo de Visual C++.
45. Ud. deberá colocar todos los recursos no editables por el ambiente de desarrollo en este archivo.
46.
47. COMPFRAC.CLW
48. Este archivo contiene información utilizada por ClassWizard para editar
49. las clases existentes o adherir nuevas clases. ClassWizard también usa
50. este archivo para almacenar información necesaria para crear y editar mapas de mensajes y mapas de datos de los
51. Cuadros de Diálogo y para crear los prototipos de las funciones miembro.
52.
53. ////////////////////////////////////////////////////////////////////
54. AppWizard crea una clase para el Diálogo Principal :
55.
56. El Cuadro de Diálogo - FracCdlg.H, FracCdlg.CPP
57. Estos archivos contienen la clase CCompFracDlg. Esta clase define
58. el comportamiento del Diálogo Principal de la aplicación. La plantilla del
59. Diálogo esta en COMPFRAC.RC, la cual puede ser editada con el
60. Estudio de Desarrollo de Visual C++ de Microsoft.
61.
62. ////////////////////////////////////////////////////////////////////
63. Archivos de Contenido Específicos de la Aplicación :
64.
```

```
FRAC.H, FRAC.CPP
  Los Archivos contienen la librería en Clases, sobre Compresión Fractal, basado en el código de Yuval Fisher

VISORIMG.H, VISORIMG.CPP
  Estos archivos contienen el visualizador para el proceso de compresión y descompresión

CCOPCION.H,CCOPCION.CPP
  Estos archivos contienen las opciones para los procesos de compresión y descompresión fractal

////////////////////////////////////////////////////////
Otros archivo estándar:

STDAFX.H, STDAFX.CPP
  Estos archivos son utilizados para construir un archivo de encabezado precompilado (PCH)
  llamado COMPFRAC.PCH y un archivo precompilado de tipos llamado STDAFX.OBJ.

RESOURCE.H
  Este es el archivo de encabezado estándar, el cual define nuevos IDs de recursos.
  Visual C++ lee y actualiza este archivo.

////////////////////////////////////////////////////////
Otras notas:
AppWizard utiliza la frase en inglés " TO DO: " (" A Realizar: ") para indicar aquellas partes del código a las que se puede
agregar o modificar el código de la Aplicación en particular. Sin embargo, para este caso se han traducido también estas
líneas indicativas.


Si su aplicación utiliza MFC en librerías compartidas DLL, y su aplicación está en otro lenguaje diferente del que el
sistema operativo maneja, necesitará una copia de los recursos correspondientes localizados en MFC40XXX.DLL, del
directorio system o system32

 del CDROM de Visual C++ de Microsoft, y renombrarlo como MFCLOD.DLL.
XXX es la abreviación del idioma. Por ejemplo, MFC40DEU.DLL contiene los recursos trasladados al Idioma Alemán.
Si Ud. no realiza esto anterior, algunos elementos UI de su aplicación permanecerán en el lenguaje del sistema
operativo.

////////////////////////////////////////////////////////
DESCRIPCION DE CLASES :

La siguiente descripción es un agregado que AppWizard NO crea, sin embargo se determinó que es importante tener
esta definición de la constitución de clases para dimensionar la estructura, el contenido y como una rápida introducción
al funcionamiento del Programa ComprFrac.

Las clases se listarán en orden alfabético :

ARCHIVOS QUE CONTIENEN A LA CLASE : Frac.CPP y Frac.H

CLASE :     CCompFrac

        Descripción :     Contenido de las rutinas de Compresión y Descompresión.

    Miembros Dato : m_ArchEntr, m_ArchSal, m_bitmap, m_bits_desplz, m_bits_escalam, m_bits_necesarios,
        m_clase_completa_de_busq, m_dom_intervmult, m_dom_taminterv, m_dominio, m_factorescala,
        m_hwnd, m_ialfa_cero, m_id, m_imagen, m_imagen_dividida, m_imagenTemp, m_info_dominio,
        m_intrv_h_dom, m_intrv_v_dom, m_max_escala, m_max_exponente, m_max_part,
        m_num_bytes_expaq, m_numDomb, m_part_sal, m_ptr, m_ptransf, m_salida, m_solo_positivo,
        m_subclase_de_busq, m_sum, m_szNomArchEntr, m_szNomArchSal, m_Tablacolor, m_tam, m_tamh,
        m_tamv, m_tipo_dom, m_transformaciones, m_transformada_clase, m_transformada_rotacion,
        m_valrms


    Estructura :    datos_dominio
        { intrv_h_dom, intrv_v_dom, num_doms_h, num_doms_v, pixel, tam_h_dom, tam_v_dom }
    Estructura :    dominios_clasificados
        { infodom, sig }
    Estructura :    nodo_transformacion
        { desplz, domx, domy, escala, oper_sim, prof, mgx, mgy, mx, my, sig, tamx, tamy }
    Estructura :    pixeles_dominio
        { dom_x, dom_y, sum, sum2, sym, }

    Funciones Miembro :
```

Aplica_Tranformaciones(), Calcula_Sums(), CCompFrac(), ~CCompFrac(), Clasifica(), Compara(), Comprimir(), Descomprimir(), EscrAEncab(), EscribArchRaw(), HContxDsp(), HManejBitmap(), IncorporaNomArch(), Lee_Transformaciones(), LeeArchRAW(), LeeDeEncab(), Mensaje(), Mostrar(), Particiona_Imagen(int, int, int, int, double, HWND), Particiona_Imagen(int, int, int, int), Promedio(), Promedio1(), Quadtree(), Restablece(), Suaviza_Imagen()

ARCHIVOS QUE CONTIENEN A LA CLASE : CompFrac.CPP y CompFrac.H

CLASE : CCompFracApp

Descripción : Soporte general para la Aplicación.

Miembros Dato : ——

Funciones Miembro :
CCompFracApp(), InitInstace(), WinHelp()

ARCHIVOS QUE CONTIENEN A LA CLASE : FracCDlg.CPP y FracCDlg.H

CLASE : CCompFracDlg

Descripción : Soporte para el cuadro de Diálogo Principal de la Aplicación.

Miembros Dato : copcion, dopcion, m_abrearchivo, m_abreayuda, m_archEntrada, m_archSalida, m_btnAceptar, m_Compresion, m_creditos, m_hIcono, m_hwndEscritorio, m_lAltoDlg, m_lAnchoDlg, m_LpRegistro, m_rcDimsDialogo, m_rcDimsEscritorio, m_Realizado

Funciones Miembro :
CCompFracDlg(), DoDataExchange(), LimpiaRutas(), NomArchNuevo(), OnAbrearchivo(), OnAyudaWin(), OnInitDialog(), OnMuestracreditos(), OnOK(), OnPaint(), OnQueryDragIcon(), OnSysCommand()

ARCHIVOS QUE CONTIENEN A LA CLASE : COpcion.CPP y COpcion.H

CLASE : CCopcion

Descripción : Soporte para el cuadro de Diálogo para recoger los parámetros de Compresión.

Miembros Dato : m_24clasdom, m_3clasdom, m_altura, m_ancho, m_bitsdesplzc, m_bitsescalc, m_cambio, m_factescalmax,m_idm, m_maxprofrec, m_minprofrec, m_mostrar, m_pos, m_ticd, m_tipo_cont, m_tol

Funciones Miembro :
CCopcion(), DoDataExchange(), OnCommand(), OnDefaultc(), OnOK()

ARCHIVOS QUE CONTIENEN A LA CLASE : FracCDlg.CPP

CLASE : CCredDlg

Descripción : Soporte para el cuadro de Diálogo de Créditos de la Aplicación.

Miembros Dato : m_logoimg

Funciones Miembro :
CCredDlg(), DoDataExchange(), OnInitDialog()

ARCHIVOS QUE CONTIENEN A LA CLASE : COpcion.CPP y COpcion.H

CLASE : CDopcion

Descripción : Soporte para el cuadro de Diálogo para recoger los parámetros de Descompresión.

Miembros Dato : m_bitsdesplz, m_bitsescal, m_cambio, m_factescalmax, m_mostrar, m_numiter, m_postproc, m_salida, m_scalsal

Funciones Miembro :
CDopcion(), DoDataExchange(), OnDefaultd(), OnOK()

```
207.    ARCHIVOS QUE CONTIENEN A LA CLASE : VisorImg.CPP y VisorImg.H
208.
209.    CLASE :   CVisorImg
210.
211.    Descripción :      Soporte para la Visualización de la Imagen tanto en Compresión como Descompresión.
212.
213.  Miembros Dato : m_bitmap, m_GuardaDc, m_respaldo, m_Tamh, m_Tamv
214.
215.  Funciones Miembro :
216.       CVisorImg(), ~CVisorImg(), GuardaDC(), OnCreate(), OnPaint(), OnSysCommand()
217.
```

Fin de Archivo LEEME.TXT

Archivo CompFrac.MAK

```
1.  # Microsoft Developer Studio Generated NMAKE File, Format Version 40001
2.  # ** DO NOT EDIT **
3.
4.  # TARGTYPE «Win32 (x86) Application» 0x0101
5.
6.  !IF «$(CFG)» == «»
7.  CFG=CompFrac - Win32 Debug
8.  !MESSAGE No configuration specified.  Defaulting to CompFrac - Win32 Debug.
9.  !ENDIF

10. !IF «$(CFG)» != «CompFrac - Win32 Debug» && «$(CFG)» !=\
11.  «CompFrac - Win32 Release»
12. !MESSAGE Invalid configuration «$(CFG)» specified.
13. !MESSAGE You can specify a configuration when running NMAKE on this makefile !MESSAGE by defining
14. the macro CFG on the command line.  For example: !MESSAGE
15. !MESSAGE NMAKE /f «CompFrac.mak» CFG=»CompFrac - Win32 Debug»
16. !MESSAGE
17. !MESSAGE Possible choices for configuration are:
18. !MESSAGE
19. !MESSAGE «CompFrac - Win32 Debug» (based on «Win32 (x86) Application»)
20. !MESSAGE «CompFrac - Win32 Release» (based on «Win32 (x86) Application»)
21. !MESSAGE
22. !ERROR An invalid configuration is specified.
23. !ENDIF

24. !IF «$(OS)» == «Windows_NT»
25. NULL=
26. !ELSE
27. NULL=nul
28. !ENDIF
29. ################################################################################
30. # Begin Project
31. # PROP Target_Last_Scanned «CompFrac - Win32 Release»
32. CPP=cl.exe
33. RSC=rc.exe
34. MTL=mktyplib.exe

35. !IF  «$(CFG)» == «CompFrac - Win32 Debug»

36. # PROP BASE Use_MFC 6
37. # PROP BASE Use_Debug_Libraries 1
38. # PROP BASE Output_Dir «WinDebug»
39. # PROP BASE Intermediate_Dir «WinDebug»
40. # PROP Use_MFC 5
41. # PROP Use_Debug_Libraries 1
42. # PROP Output_Dir «WinDebug»
43. # PROP Intermediate_Dir «WinDebug\interm» OUTDIR=.\WinDebug
44. INTDIR=.\WinDebug\interm

45. ALL : «$(OUTDIR)\CompFrac.exe» «$(OUTDIR)\CompFrac.bsc»

46. CLEAN :
47. -@erase «.\WinDebug\interm\vc40.pdb»
48. -@erase «.\WinDebug\interm\CompFrac.pch»
49. -@erase «.\WinDebug\interm\vc40.idb»
50. -@erase «.\WinDebug\CompFrac.bsc»
51. -@erase «.\WinDebug\interm\FRAC.SBR»
52. -@erase «.\WinDebug\interm\stdafx.sbr»
53. -@erase «.\WinDebug\interm\FracCdlg.sbr»
54. -@erase «.\WinDebug\interm\CompFrac.sbr»
55. -@erase «.\WinDebug\interm\Copcion.sbr»
56. -@erase «.\WinDebug\interm\Visorimg.sbr»
```

```
57.             -@erase «.\WinDebug\CompFrac.exe»
58.             -@erase «.\WinDebug\interm\FRAC.OBJ»
59.             -@erase «.\WinDebug\interm\stdafx.obj»
60.             -@erase «.\WinDebug\interm\FracCdlg.obj»
61.             -@erase «.\WinDebug\interm\CompFrac.obj»
62.             -@erase «.\WinDebug\interm\Copcion.obj»
63.             -@erase «.\WinDebug\interm\Visorimg.obj»
64.             -@erase «.\WinDebug\CompFrac.res»
65.             -@erase «.\WinDebug\interm\CompFrac.map»

66.     «$(OUTDIR)» :
67.             if not exist «$(OUTDIR)/$(NULL)» mkdir «$(OUTDIR)»
68.     «$(INTDIR)» :
69.             if not exist «$(INTDIR)/$(NULL)» mkdir «$(INTDIR)»
70.     #       ADD BASE CPP /nologo /MD /W3 /GX /Zi /Od /D «_DEBUG» /D «WIN32» /D «_WINDOWS» /D «_MBCS» /D
71.     «_AFXDLL» /FR /Yu»stdafx.h» /c
72.     #       ADD CPP /nologo /MTd /W3 /Gm /GX /Zi /Od /D «_DEBUG» /D «WIN32» /D «_WINDOWS» /D «_MBCS» /Fr
73.     /Yu»stdafx.h» /c
74.     CPP_PROJ=/nologo /MTd /W3 /Gm /GX /Zi /Od /D «_DEBUG» /D «WIN32» /D «_WINDOWS»\
75.             /D «_MBCS» /Fr»$(INTDIR)/» /Fp»$(INTDIR)/CompFrac.pch» /Yu»stdafx.h»\
76.             /Fo»$(INTDIR)/» /Fd»$(INTDIR)/» /c
77.     CPP_OBJS=.\WinDebug\interm/
78.     CPP_SBRS=.\WinDebug\interm/
79.     #       ADD BASE RSC /l 0x409 /d «_DEBUG» /d «_AFXDLL»
80.     #       ADD RSC /l 0x80a /fo»WinDebug/CompFrac.res» /d «_DEBUG»
81.     RSC_PROJ=/l 0x80a /fo»WinDebug/CompFrac.res» /d «_DEBUG»
82.     BSC32=bscmake.exe
83.     #       ADD BASE BSC32 /nologo
84.     #       ADD BSC32 /nologo
85.     BSC32_FLAGS=/nologo /o»$(OUTDIR)/CompFrac.bsc»
86.     BSC32_SBRS= \
87.             «$(INTDIR)/FRAC.SBR» \
88.             «$(INTDIR)/stdafx.sbr» \
89.             «$(INTDIR)/FracCdlg.sbr» \
90.             «$(INTDIR)/CompFrac.sbr» \
91.             «$(INTDIR)/Copcion.sbr» \
92.             «$(INTDIR)/Visorimg.sbr»

93.     «$(OUTDIR)\CompFrac.bsc» : «$(OUTDIR)» $(BSC32_SBRS)
94.             $(BSC32) @<<
95.     $(BSC32_FLAGS) $(BSC32_SBRS)
96.     <<

97.     LINK32=link.exe
98.     #       ADD BASE LINK32 /nologo /subsystem:windows /debug /machine:I386
99.     #       SUBTRACT BASE LINK32 /pdb:none
100.    #       ADD LINK32 /nologo /subsystem:windows /pdb:none /map /debug /machine:I386
101.    #       SUBTRACT LINK32 /nodefaultlib
102.    LINK32_FLAGS=/nologo /subsystem:windows /pdb:none /map:»$(INTDIR)/CompFrac.map»\
103.    /debug /machine:I386 /out:»$(OUTDIR)/CompFrac.exe»
104.    LINK32_OBJS= \
105.                    «$(INTDIR)/FRAC.OBJ» \
106.                    «$(INTDIR)/stdafx.obj» \
107.                    «$(INTDIR)/FracCdlg.obj» \
108.                    «$(INTDIR)/CompFrac.obj» \
109.                    «$(INTDIR)/Copcion.obj» \
110.                    «$(INTDIR)/Visorimg.obj» \
111.                    «.\WinDebug\CompFrac.res»

112.    «$(OUTDIR)\CompFrac.exe» : «$(OUTDIR)» $(DEF_FILE) $(LINK32_OBJS)
113.            $(LINK32) @<<
114.    $(LINK32_FLAGS) $(LINK32_OBJS)
115.    <<

116.    !ELSEIF «$(CFG)» == «CompFrac - Win32 Release»

117.    #       PROP BASE Use_MFC 6
118.    #       PROP BASE Use_Debug_Libraries 0
119.    #       PROP BASE Output_Dir «WinRel»
```

Archivo *CompFrac.MAK*

```
120.  #       PROP BASE Intermediate_Dir «WinRel»
121.  #       PROP Use_MFC 5
122.  #       PROP Use_Debug_Libraries 0
123.  #       PROP Output_Dir «WinRel»
124.  #       PROP Intermediate_Dir «WinRel\interm»
125.  OUTDIR=.\WinRel
126.  INTDIR=.\WinRel\interm

127.  ALL : «$(OUTDIR)\CompFrac.exe» «$(OUTDIR)\CompFrac.bsc»

128.  CLEAN :
129.          -@erase «.\WinRel\CompFrac.bsc»
130.          -@erase «.\WinRel\interm\stdafx.sbr»
131.          -@erase «.\WinRel\interm\Visorimg.sbr»
132.          -@erase «.\WinRel\interm\CompFrac.pch»
133.          -@erase «.\WinRel\interm\CompFrac.sbr»
134.          -@erase «.\WinRel\interm\FRAC.SBR»
135.          -@erase «.\WinRel\interm\FracCdlg.sbr»
136.          -@erase «.\WinRel\interm\Copcion.sbr»
137.          -@erase «.\WinRel\CompFrac.exe»
138.          -@erase «.\WinRel\interm\Copcion.obj»
139.          -@erase «.\WinRel\interm\stdafx.obj»
140.          -@erase «.\WinRel\interm\Visorimg.obj»
141.          -@erase «.\WinRel\interm\CompFrac.obj»
142.          -@erase «.\WinRel\interm\FRAC.OBJ»
143.          -@erase «.\WinRel\interm\FracCdlg.obj»
144.          -@erase «.\WinRel\CompFrac.res»

145.  «$(OUTDIR)» :
146.      if not exist «$(OUTDIR)/$(NULL)» mkdir «$(OUTDIR)»

147.  «$(INTDIR)» :
148.      if not exist «$(INTDIR)/$(NULL)» mkdir «$(INTDIR)»

149.  #       ADD BASE CPP /nologo /MD /W3 /GX /O2 /D «NDEBUG» /D «WIN32» /D «_WINDOWS» /D «_MBCS» /D
150.  «_AFXDLL» /FR /Yu«stdafx.h» /c
151.  #       ADD CPP /nologo /MT /W3 /GX /O2 /D «NDEBUG» /D «WIN32» /D «_WINDOWS» /D «_MBCS» /Fr /Yu«stdafx.h»
152.          /c
153.  CPP_PROJ=/nologo /MT /W3 /GX /O2 /D «NDEBUG» /D «WIN32» /D «_WINDOWS» /D\
154.          «_MBCS» /Fr«$(INTDIR)/» /Fp«$(INTDIR)/CompFrac.pch» /Yu«stdafx.h»\
155.                  /Fo«$(INTDIR)/» /c
156.  CPP_OBJS=.\WinRel\interm/
157.  CPP_SBRS=.\WinRel\interm/
158.  #       ADD BASE RSC /l 0x409 /d «NDEBUG» /d «_AFXDLL»
159.  #       ADD RSC /l 0x80a /fo«WinRel/CompFrac.res» /d «NDEBUG»
160.  RSC_PROJ=/l 0x80a /fo«WinRel/CompFrac.res» /d «NDEBUG»
161.  BSC32=bscmake.exe
162.  #       ADD BASE BSC32 /nologo
163.  #       ADD BSC32 /nologo
164.  BSC32_FLAGS=/nologo /o«$(OUTDIR)/CompFrac.bsc» BSC32_SBRS= \
165.          «$(INTDIR)/stdafx.sbr» \
166.          «$(INTDIR)/Visorimg.sbr» \ «$(INTDIR)/CompFrac.sbr» \
167.          «$(INTDIR)/FRAC.SBR» \
168.          «$(INTDIR)/FracCdlg.sbr» \
169.          «$(INTDIR)/Copcion.sbr»

170.  «$(OUTDIR)\CompFrac.bsc» : «$(OUTDIR)» $(BSC32_SBRS) $(BSC32) @<<
171.  $(BSC32_FLAGS) $(BSC32_SBRS)
172.  <<

173.  LINK32=link.exe
174.  #       ADD BASE LINK32 /nologo /subsystem:windows /machine:I386
175.  #       SUBTRACT BASE LINK32 /pdb:none
176.  #       ADD LINK32 /nologo /subsystem:windows /machine:I386
177.  # SUBTRACT LINK32 /pdb:none
178.  LINK32_FLAGS=/nologo /subsystem:windows /incremental:no\
179.   /pdb:«$(OUTDIR)/CompFrac.pdb» /machine:I386 /out:«$(OUTDIR)/CompFrac.exe»
180.  LINK32_OBJS= \
181.          «$(INTDIR)/Copcion.obj» \
```

```
182.                    «$(INTDIR)/stdafx.obj» \
183.                    «$(INTDIR)/Visorimg.obj» \
184.                    «$(INTDIR)/CompFrac.obj» \
185.                    «$(INTDIR)/FRAC.OBJ» \
186.                    «$(INTDIR)/FracCdlg.obj» \
187.                    «.\WinRel\CompFrac.res»

188. «$(OUTDIR)\CompFrac.exe» : «$(OUTDIR)» $(DEF_FILE) $(LINK32_OBJS)
189.             $(LINK32) @<<
190.   $(LINK32_FLAGS) $(LINK32_OBJS)
191. <<

192. !ENDIF

193. .c{$(CPP_OBJS)}.obj:
194.             $(CPP) $(CPP_PROJ) $<

195. .cpp{$(CPP_OBJS)}.obj:
196.             $(CPP) $(CPP_PROJ) $<

197. .cxx{$(CPP_OBJS)}.obj:
198.             $(CPP) $(CPP_PROJ) $<

199. .c{$(CPP_SBRS)}.sbr:
200.             $(CPP) $(CPP_PROJ) $<

201. .cpp{$(CPP_SBRS)}.sbr:
202.             $(CPP) $(CPP_PROJ) $<

203. .cxx{$(CPP_SBRS)}.sbr:
204.             $(CPP) $(CPP_PROJ) $<

205. MTL_PROJ=

206. ##############################################################################
207. # Begin Target

208. # Name «CompFrac - Win32 Debug»
209. # Name «CompFrac - Win32 Release»

210. !IF  «$(CFG)» == «CompFrac - Win32 Debug»

211. !ELSEIF  «$(CFG)» == «CompFrac - Win32 Release»
212. !ENDIF
213. ##############################################################################
214. # Begin Source File

215. SOURCE=.\stdafx.cpp
216. DEP_CPP_STDAF=\
217.                     «.\stdafx.h»\


218. !IF  «$(CFG)» == «CompFrac - Win32 Debug»

219. # ADD BASE CPP /Yc«stdafx.h»
220. # ADD CPP /Yc«stdafx.h»

221. BuildCmds= \
222.                     $(CPP) /nologo /MTd /W3 /Gm /GX /Zi /Od /D «_DEBUG» /D «WIN32» /D «_WINDOWS»\
223.  /D «_MBCS» /Fr«$(INTDIR)/» /Fp«$(INTDIR)/CompFrac.pch» /Yc«stdafx.h»\
224. /Fo«$(INTDIR)/» /Fd«$(INTDIR)/» /c $(SOURCE) \
225. «$(INTDIR)\stdafx.obj» : $(SOURCE) $(DEP_CPP_STDAF) «$(INTDIR)»
226. $(BuildCmds)
```

Archivo *CompFrac.MAK*

```
227. «$(INTDIR)\stdafx.sbr» : $(SOURCE) $(DEP_CPP_STDAF) «$(INTDIR)»
228.         $(BuildCmds)

229. «$(INTDIR)\CompFrac.pch» : $(SOURCE) $(DEP_CPP_STDAF) «$(INTDIR)»
230.         $(BuildCmds)

231. !ELSEIF  «$(CFG)» == «CompFrac - Win32 Release»

232. # ADD BASE CPP /Yc«stdafx.h»
233. # ADD CPP /Yc«stdafx.h»

234. BuildCmds= \
235.         $(CPP) /nologo /MT /W3 /GX /O2 /D «NDEBUG» /D «WIN32» /D «_WINDOWS» /D «_MBCS»\
236.  /Fr»$(INTDIR)/» /Fp»$(INTDIR)/CompFrac.pch» /Yc«stdafx.h» /Fo»$(INTDIR)/» /c\
237.  $(SOURCE) \


238. «$(INTDIR)\stdafx.obj» : $(SOURCE) $(DEP_CPP_STDAF) «$(INTDIR)»
239.         $(BuildCmds)

240. «$(INTDIR)\stdafx.sbr» : $(SOURCE) $(DEP_CPP_STDAF) «$(INTDIR)»
241.         $(BuildCmds)

242. «$(INTDIR)\CompFrac.pch» : $(SOURCE) $(DEP_CPP_STDAF) «$(INTDIR)»
243.         $(BuildCmds)

244. !ENDIF

245. # End Source File
246. ################################################################################
247. # Begin Source File

248. SOURCE=.\CompFrac.cpp
249. DEP_CPP_COMPF=\
250.         «.\stdafx.h»\
251.         «.\CompFrac.h»\
252.         «.\Fraccdlg.h»\
253.         «.\Copcion.h»\
254.         «.\Visorimg.h»\


255. «$(INTDIR)\CompFrac.obj» : $(SOURCE) $(DEP_CPP_COMPF) «$(INTDIR)»\
256.  «$(INTDIR)\CompFrac.pch»

257. «$(INTDIR)\CompFrac.sbr» : $(SOURCE) $(DEP_CPP_COMPF) «$(INTDIR)»\
258.  «$(INTDIR)\CompFrac.pch»


259. # End Source File
260. ################################################################################
261. # Begin Source File

262. SOURCE=.\FracCdlg.cpp
263. DEP_CPP_FRACC=\
264.         «.\stdafx.h»\
265.         «.\CompFrac.h»\
266.         «.\Fraccdlg.h»\
267.         «.\FRAC.H»\
268.         «.\Copcion.h»\
269.         «.\Visorimg.h»\
270.

271. «$(INTDIR)\FracCdlg.obj» : $(SOURCE) $(DEP_CPP_FRACC) «$(INTDIR)»\
272.  «$(INTDIR)\CompFrac.pch»

273. «$(INTDIR)\FracCdlg.sbr» : $(SOURCE) $(DEP_CPP_FRACC) «$(INTDIR)»\
```

```
274. «$(INTDIR)\CompFrac.pch»


275. # End Source File
276. ################################################################################
277. # Begin Source File

278. SOURCE=.\CompFrac.rc
279. DEP_RSC_COMPFR=\
280.         «.\res\CompFrc1.ico»\
281.         «.\res\CompFrc2.ico»\
282.         «.\res\CompFrc3.ico»\
283.         «.\res\CompFrc4.ico»\
284.         «.\res\bitmap2.bmp»\
285.         «.\res\bitmap3.bmp»\
286.         «.\res\logo150.bmp»\
287.         «.\resource.hm»\
288.         «.\res\CompFrac.rc2»\


289. !IF  «$(CFG)» == «CompFrac - Win32 Debug»

290. # ADD BASE RSC /l 0x409
291. # ADD RSC /l 0x409
292. # SUBTRACT RSC /x
293. «$(INTDIR)\CompFrac.res» : $(SOURCE) $(DEP_RSC_COMPFR) «$(INTDIR)» $(RSC) /l 0x409 /fo»WinDebug/
294. CompFrac.res» /d «_DEBUG» $(SOURCE)
295. !ELSEIF  «$(CFG)» == «CompFrac - Win32 Release»
296. «$(INTDIR)\CompFrac.res» : $(SOURCE) $(DEP_RSC_COMPFR) «$(INTDIR)» $(RSC) /l 0x80a /fo»WinRel/
297. CompFrac.res» /d «NDEBUG» $(SOURCE)
298. !ENDIF
299. # End Source File
300. ################################################################################
301. # Begin Source File

302. SOURCE=.\FRAC.CPP
303. DEP_CPP_FRAC_=\
304.         «.\stdafx.h»\
305.         «.\FRAC.H»\
306.         {$(INCLUDE)}»\sys\Types.h»\
307.         {$(INCLUDE)}»\sys\Stat.h»\


308. !IF  «$(CFG)» == «CompFrac - Win32 Debug»
309. «$(INTDIR)\FRAC.OBJ» : $(SOURCE) $(DEP_CPP_FRAC_) «$(INTDIR)»\
310.  «$(INTDIR)\CompFrac.pch»

311. «$(INTDIR)\FRAC.SBR» : $(SOURCE) $(DEP_CPP_FRAC_) «$(INTDIR)»\
312.  «$(INTDIR)\CompFrac.pch»

313.         !ELSEIF  «$(CFG)» == «CompFrac - Win32 Release»


314.         «$(INTDIR)\FRAC.OBJ» : $(SOURCE) $(DEP_CPP_FRAC_) «$(INTDIR)»\
315.          «$(INTDIR)\CompFrac.pch»

316.         «$(INTDIR)\FRAC.SBR» : $(SOURCE) $(DEP_CPP_FRAC_) «$(INTDIR)»\
317.          «$(INTDIR)\CompFrac.pch»


318.         !ENDIF

319.         # End Source File
320.         ################################################################################
321.         # Begin Source File
```

```
322.  SOURCE=.\Visorimg.cpp
323.  DEP_CPP_VISOR=\
324.      «.\stdafx.h»\
325.      «.\Visorimg.h»\

326.  «$(INTDIR)\Visorimg.obj» : $(SOURCE) $(DEP_CPP_VISOR) «$(INTDIR)»\
327.   «$(INTDIR)\CompFrac.pch»

328.  «$(INTDIR)\Visorimg.sbr» : $(SOURCE) $(DEP_CPP_VISOR) «$(INTDIR)»\
329.   «$(INTDIR)\CompFrac.pch»

330.  # End Source File
331.  ##############################################################################
332.  # Begin Source File

333.  SOURCE=.\Leeme.txt

334.  !IF  «$(CFG)» == «CompFrac - Win32 Debug» !ELSEIF  «$(CFG)» == «CompFrac - Win32 Release»
335.  !ENDIF
336.
337.  # End Source File
338.  ##############################################################################
339.  # Begin Source File
340.  SOURCE=.\Copcion.cpp
341.  DEP_CPP_COPCI=\
342.      «.\stdafx.h»\
343.      «.\CompFrac.h»\
344.      «.\Copcion.h»\
345.  «$(INTDIR)\Copcion.obj» : $(SOURCE) $(DEP_CPP_COPCI) «$(INTDIR)»\
346.   «$(INTDIR)\CompFrac.pch»

347.  «$(INTDIR)\Copcion.sbr» : $(SOURCE) $(DEP_CPP_COPCI) «$(INTDIR)»\
348.   «$(INTDIR)\CompFrac.pch»

349.  # End Source File
350.  # End Target
351.  # End Project

352.  ##############################################################################
```

FIN de Archivo CompFrac.MAK.

Archivo Resource.H

```
1.  //{{NO_DEPENDENCIES}}
2.  // Microsoft Developer Studio generated include file. // Used by CompFrac.rc
3.  //
4.  #define IDC_DEFAULTC                    3
5.  #define IDC_DEFAULTD                    3
6.  #define IDM_CREDITOS                    0x0010
7.  #define IDM_PORTAPAPELES                0x0020
8.  #define IDD_CAJACREDITOS                100
9.  #define IDS_CREDITOS                    101
10. #define IDD_COMPFRAC_MAIN               102
11. #define IDD_OPCIONESC                   136
12. #define IDD_OPCIONESD                   138
13. #define IDR_APPCFRAC1                   151
14. #define IDI_APPCFRAC2                   152
15. #define IDI_APPCFRAC3                   153
16. #define IDI_APPCFRAC4                   154
17. #define IDC_COMPRESION                  1002
18. #define IDC_DESCOMPRESION               1003
19. #define IDC_OPCIONES                    1004
20. #define IDC_CADNOMBARCHSAL              1005
21. #define IDC_NOMBARCHSAL                 1006
22. #define IDC_CADNOMBARCHENTR             1007
23. #define IDC_NOMBARCHENTR                1008
24. #define IDC_ESTADO                      1009
25. #define IDC_TOLERANCIA                  1010
26. #define IDC_MINPREC                     1011
27. #define IDC_ANCHOIMG                    1012
28. #define IDC_ALTOIMG                     1013
29. #define IDC_TAMINTERVCDOM               1014
30. #define IDC_MAXPREC                     1015
31. #define IDC_TIPOCONTEN                  1018
32. #define IDC_BITSESCALC                  1019
33. #define IDC_BITSDESPLZC                 1020
34. #define IDC_FACTESCALMAXC               1021
35. #define IDC_NUMITER                     1022
36. #define IDC_ESCALASAL                   1023
37. #define IDC_FACTESCALMAXD               1024
38. #define IDC_BITSESCALD                  1025
39. #define IDC_BITSDESPLZD                 1026
40. #define IDC_POSTP                       1027
41. #define IDC_MOSTRARPROC                 1028
42. #define IDC_24CLASDOMIN                 1029
43. #define IDC_3CLASDOMIN                  1030
44. #define IDC_POSTIVO                     1031
45. #define IDC_IDM                         1032
46. #define IDC_SALIDA                      1033
47. #define IDC_BOTONIMG                    1034
48. #define IDC_MOSTRQUADTREE               1035
49. #define IDC_ABREARCHIVO                 1040
50. #define IDC_MUESTRACREDITOS             1041
51. #define IDC_AYUDAWIN                    1042
52. #define IDC_OPERACION                   1043

53. // Next default values for new objects
54. //
55. #ifdef APSTUDIO_INVOKED
56. #ifndef APSTUDIO_READONLY_SYMBOLS
57. #define _APS_NEXT_RESOURCE_VALUE        164
58. #define _APS_NEXT_COMMAND_VALUE         32771
59. #define _APS_NEXT_CONTROL_VALUE         1046
60. #define _APS_NEXT_SYMED_VALUE           101
61. #endif
62. #endif
```

Fin de Archivo Resource.H

Archivo CompFrac.RC

```
1.  //Microsoft Developer Studio generated resource script.
2.  //
3.  #include «resource.h»

4.  // Generated Help ID header file
5.  #define APSTUDIO_HIDDEN_SYMBOLS
6.  #include «resource.hm»
7.  #undef APSTUDIO_HIDDEN_SYMBOLS

8.  #define APSTUDIO_READONLY_SYMBOLS
9.  /////////////////////////////////////////////////////////////////////
10. //
11. // Generated from the TEXTINCLUDE 2 resource.
12. //
13. #include «afxres.h»

14. /////////////////////////////////////////////////////////////////////
15. #undef APSTUDIO_READONLY_SYMBOLS

16. /////////////////////////////////////////////////////////////////////
17. // English (U.S.) resources

18. #if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
19. #ifdef _WIN32
20. LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
21. #pragma code_page(1252)
22. #endif //_WIN32

23. #ifdef APSTUDIO_INVOKED
24. /////////////////////////////////////////////////////////////////////
25. //
26. // TEXTINCLUDE
27. //

28. 1 TEXTINCLUDE DISCARDABLE
29. BEGIN
30.     «resource.h\0»
31. END

32. 2 TEXTINCLUDE DISCARDABLE
33. BEGIN
34.     «#include «»afxres.h»»\r\n»
35.     «\0»
36. END

37. 3 TEXTINCLUDE DISCARDABLE
38. BEGIN
39.     «#include «»res\\CompFrac.rc2»»  // non-Microsoft Visual C++ edited resources\r\n»
40.     «\r\n»
41.     «#define _AFX_NO_SPLITTER_RESOURCES\r\n»
42.     «#define _AFX_NO_OLE_RESOURCES\r\n»
43.     «#define _AFX_NO_TRACKER_RESOURCES\r\n»
44.     «#define _AFX_NO_PROPERTY_RESOURCES\r\n»
45.     «#include «»afxres.rc»»  // Standard components\r\n»
46.     «\0»
47. END

48. #endif    // APSTUDIO_INVOKED

49. #endif    // English (U.S.) resources
50. /////////////////////////////////////////////////////////////////////
```

```
51.  //////////////////////////////////////////////////////
52.  // Spanish (Mexican) resources

53.  #if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ESM)
54.  #ifdef _WIN32
55.  LANGUAGE LANG_SPANISH, SUBLANG_SPANISH_MEXICAN
56.  #pragma code_page(1252)
57.  #endif //_WIN32

58.  //////////////////////////////////////////////////////// //
59.  // Icon
60.  //

61.  // Icon with lowest ID value placed first to ensure application icon
62.  // remains consistent on all systems.
63.  IDR_APPCFRAC1            ICON   DISCARDABLE   «res\\CompFrc1.ico»
64.  IDI_APPCFRAC2            ICON   DISCARDABLE   «res\\CompFrc2.ico»
65.  IDI_APPCFRAC3            ICON   DISCARDABLE   «res\\CompFrc3.ico»
66.  IDI_APPCFRAC4            ICON   DISCARDABLE   «res\\CompFrc4.ico»

67.  //////////////////////////////////////////////////////// //
68.  // Dialog
69.  //

70.  IDD_CAJACREDITOS DIALOGEX 34, 22, 289, 156
71.  STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
72.  CAPTION «Créditos de la Aplicación»
73.  FONT 8, «MS Sans Serif»
74.  BEGIN
75.                          PUSHBUTTON      «Aceptar»,IDOK,157,138,100,13,WS_GROUP
76.                          CTEXT           «Codificador/Decodificador Fractal de Imágenes\n para Windows, Version
77.                              1.0.0  (Win32s)»,
78.                          IDC_STATIC,109,7,171,20,0,WS_EX_CLIENTEDGE |
79.                              WS_EX_STATICEDGE
80.                          CTEXT           «Desarrollado por :\n Mª del Rocío Barajas I. y Said Miranda R.»,
81.                          IDC_STATIC,117,34,155,16
82.                          CTEXT           «Aplicación Complementaria de Apoyo al Trabajo\n de Tesis de Nivel
83.                              Licenciatura.»,
84.                          IDC_STATIC,117,55,155,17
85.                          CTEXT           « Carrera :  INGENIERÍA EN COMPUTACIÓN.\n ",
86.                          IDC_STATIC,117,75,154,9
87.                          CONTROL     «»,IDC_BOTONIMG,»Button»,BS_OWNERDRAW | WS_TABSTOP,5,5,
88.                          96,96
89.                          CTEXT           «Basado en el Algoritmo «»Quadtree»» y código, del Libro
90.                              «»Fractal Image Compression»» de Yuval Fisher, y Código de Programa de
91.                              Jude Sylvestre.». IDC_STATIC,7,109,271,17
92.                          CTEXT           «Copyright © 1995-1997»,IDC_STATIC,29,140,109,8
93.                          CONTROL     «»,IDC_STATIC,»Static»,SS_BLACKRECT,7,99,273,1
94.                          CTEXT           «Director de TESIS : Mat. Luis Ramírez Flores.»,
95.                          IDC_STATIC,124,87,144,8
96.  END

97.  IDD_COMPFRAC_MAIN DIALOGEX 0, 0, 210, 210
98.  STYLE DS_MODALFRAME | WS_MINIMIZEBOX | WS_MAXIMIZEBOX | WS_POPUP |
99.                          WS_VISIBLE | WS_CAPTION | WS_SYSMENU
100. EXSTYLE WS_EX_CONTEXTHELP
101. CAPTION «Aplicación Demostrativa»
102. FONT 8, «Times New Roman»
103. BEGIN
104.                         PUSHBUTTON      «Abrir Archivo»,IDC_ABREARCHIVO,21,122,50,14,0,0,
105.                             HIDC_ABREARCHIVO
106.                         DEFPUSHBUTTON «Aceptar»,IDOK,20,190,50,14
107.                         PUSHBUTTON      «Ayuda»,IDC_AYUDAWIN,79,122,50,14
108.                         PUSHBUTTON      «Salir»,IDCANCEL,140,190,50,14
109.                         PUSHBUTTON      «Créditos»,IDC_MUESTRACREDITOS,137,122,50,14
110.                         EDITTEXT    IDC_CADNOMBARCHENTR,6,81,200,12,ES_AUTOHSCROLL
111.                         EDITTEXT    IDC_CADNOMBARCHSAL,5,106,200,12,ES_AUTOHSCROLL
112.                         PUSHBUTTON      «Opciones»,IDC_OPCIONES,80,190,50,14,NOT WS_VISIBLE
113.                         CTEXT           «para Imágenes de 8 bits a Escala de Grises, \npara Windows (32 bits).
```

Archivo *CompFrac.RC*

```
                        Método de Partición de la Imagen, por Arbol Cuaternario (Quadtree).»,
                    IDC_STATIC,26,12,159,24
                    CONTROL   «»,IDC_STATIC,»Static»,SS_BLACKRECT,39,39,134,1
                    GROUPBOX «OPERACIÓN»,IDC_STATIC,7,42,196,27
                    CONTROL   «»,IDC_STATIC,»Static»,SS_BLACKRECT,13,185,183,1
                    LTEXT     «Archivo de SALIDA :»,IDC_STATIC,73,96,70,10
                    LTEXT     «Archivo de ENTRADA :»,IDC_STATIC,68,71,79,10
                    LTEXT     «»,IDC_ESTADO,15,153,179,24
                    GROUPBOX «Estado»,IDC_STATIC,11,141,185,38
                    LTEXT     «»,IDC_OPERACION,11,49,187,19
                    CTEXT     «CODIFICADOR / DECODIFICADOR FRACTAL»,IDC_STATIC,25,3,
                        162,9,SS_SUNKEN
END

IDD_OPCIONESC DIALOG DISCARDABLE  0, 0, 249, 237
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION «Opciones de Compresión»
FONT 8, «Times New Roman»
BEGIN
                    RTEXT     «Ancho = Alto =»,IDC_STATIC,44,51,81,9
                    COMBOBOX IDC_ANCHOIMG,141,49,40,36,CBS_DROPDOWNLIST | CBS_SORT |
                        WS_VSCROLL | WS_TABSTOP
                    CTEXT     «Alto :»,IDC_STATIC,166,50,27,10,NOT WS_VISIBLE
                    EDITTEXT  IDC_ALTOIMG,193,49,33,12,ES_AUTOHSCROLL | NOT WS_VISIBLE
                    DEFPUSHBUTTON «Aceptar»,IDOK,31,215,50,14
                    PUSHBUTTON    «Default»,IDC_DEFAULTC,103,215,50,14
                    PUSHBUTTON    «Cancelar»,IDCANCEL,175,215,50,14
                    RTEXT     «Tolerancia rms :»,IDC_STATIC,5,81,51,8
                    EDITTEXT  IDC_TOLERANCIA,60,78,18,12,ES_AUTOHSCROLL
                    RTEXT     «Bits de Escalamiento:»,IDC_STATIC,87,73,45,18
                    EDITTEXT  IDC_BITSESCALC,133,77,18,13,ES_AUTOHSCROLL
                    RTEXT     «Bits de Desplazamiento :»,IDC_STATIC,156,73,55,18
                    EDITTEXT  IDC_BITSDESPLZC,218,77,18,13,ES_AUTOHSCROLL
                    RTEXT     «Mín. Profundidad de Recursión:»,IDC_STATIC,4,94,53,15
                    EDITTEXT  IDC_MINPREC,60,97,18,12,ES_AUTOHSCROLL
                    RTEXT     «Máx. Profundidad de Recursión:»,IDC_STATIC,75,93,57,17
                    EDITTEXT  IDC_MAXPREC,133,97,18,12,ES_AUTOHSCROLL
                    RTEXT     «Factor de Escala Máxima :»,IDC_STATIC,158,93,53,16
                    EDITTEXT  IDC_FACTESCALMAXC,218,97,18,13,ES_AUTOHSCROLL
                    RTEXT     «Tamaño del Intervalo del Contenedor de Dominios:»,
                        IDC_STATIC,16,115,49,32
                    EDITTEXT  IDC_TAMINTERVCDOM,73,127,17,12,ES_AUTOHSCROLL
                    RTEXT     «Tipo de Contenedor de Dominios:»,IDC_STATIC,116,118,71,
                        16
                    COMBOBOX IDC_TIPOCONTEN,191,122,27,30,CBS_DROPDOWN | CBS_SORT |
                        WS_VSCROLL | WS_TABSTOP
                    CONTROL   «Intervalo de Dominio como Múltiplo»,IDC_IDM,»Button»,
                        BS_AUTOCHECKBOX | BS_PUSHLIKE | WS_TABSTOP,114,141,128,9
                    GROUPBOX «Para Fidelidad de la Imagen Examinar en : «,IDC_STATIC,
                        8,153,231,26,BS_CENTER
                    CONTROL   «24 Clases de Dominio»,IDC_24CLASDOMIN,»Button»,
                        BS_AUTOCHECKBOX | BS_PUSHLIKE | WS_TABSTOP,36,165,85,10
                    CONTROL   «3 Clases de Dominio»,IDC_3CLASDOMIN,»Button»,
                        BS_AUTOCHECKBOX | BS_PUSHLIKE | WS_TABSTOP,136,165,81,10
                    CONTROL   «Sólo Escalamiento Positivo»,IDC_POSTIVO,»Button»,
                        BS_AUTOCHECKBOX | BS_PUSHLIKE | WS_TABSTOP,10,185,96,10
                    CTEXT     «(Incrementa Rapidez)»,IDC_STATIC,19,198,79,8
                    CONTROL   «Mostrar Proceso de Particionado»,IDC_MOSTRQUADTREE,
                        «Button»,BS_AUTOCHECKBOX | BS_PUSHLIKE | WS_TABSTOP,125,
                        185,117,11
                    CTEXT     «(por Árbol Cuaternario)»,IDC_STATIC,130,198,109,8
                    CONTROL   «»,IDC_STATIC,»Static»,SS_BLACKRECT,31,29,183,1
                    CTEXT     «Parámetros para el Algoritmo de Compresión \npor Arbol Cuaternario
                        (Quadtree)», IDC_STATIC,51,9,144,19
                    GROUPBOX «Dimensiones de la Imagen Cuadrada :»,IDC_STATIC,16,39,
                        216,28,BS_CENTER | BS_FLAT
END

IDD_OPCIONESD DIALOG DISCARDABLE  0, 0, 222, 126
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
```

```
182.  CAPTION «Opciones de Descompresión»
183.  FONT 8, «Times New Roman»
184.  BEGIN
185.        DEFPUSHBUTTON  «Aceptar»,IDOK,20,106,50,14
186.        PUSHBUTTON     «Default»,IDC_DEFAULTD,86,106,50,14
187.        PUSHBUTTON     «Cancelar»,IDCANCEL,152,106,50,14
188.        EDITTEXT             IDC_NUMITER,80,25,19,13,ES_AUTOHSCROLL
189.        EDITTEXT             IDC_ESCALASAL,170,25,19,13,ES_AUTOHSCROLL
190.        EDITTEXT             IDC_BITSESCALD,52,48,17,13,ES_AUTOHSCROLL
191.        EDITTEXT             IDC_BITSDESPLZD,126,48,17,13,ES_AUTOHSCROLL
192.        EDITTEXT             IDC_FACTESCALMAXD,197,48,17,13,ES_AUTOHSCROLL
193.        CONTROL              «Postprocesamiento»,IDC_POSTP,»Button»,BS_AUTOCHECKBOX |
194.                             BS_PUSHLIKE | WS_TABSTOP,8,69,71,8
195.        CONTROL              «Mostrar Proceso [Decrementa Rapidez]»,IDC_MOSTRARPROC,
196.              «Button»,BS_AUTOCHECKBOX | BS_PUSHLIKE | WS_TABSTOP,81,
197.              69,134,10
198.        CONTROL              «Producir Partición directa a Archivo»,IDC_SALIDA,
199.              «Button»,BS_AUTOCHECKBOX | BS_PUSHLIKE | WS_TABSTOP,28,
200.              86,173,10
201.        LTEXT                «Número de Iteraciones :»,IDC_STATIC,5,30,73,8
202.        RTEXT          «Escala de la Imagen a Decodificar:»,IDC_STATIC,108,22,
203.                             60,16
204.        RTEXT          «Factor de Escala Máxima :»,IDC_STATIC,145,47,50,14
205.        RTEXT          «Bits de Escalamiento:»,IDC_STATIC,6,45,44,16
206.        RTEXT          «Bits de Desplazamiento :»,IDC_STATIC,71,45,53,16
207.        CTEXT          «Opciones para Descompresión Fractal»,IDC_STATIC,49,5,
208.                             118,8
209.        CONTROL              «»,IDC_STATIC,»Static»,SS_BLACKRECT,40,14,143,1
210.        CTEXT          «(Suavizado)»,IDC_STATIC,23,77,37,8
211.  END


212.  #ifndef _MAC
213.  /////////////////////////////////////////////////////////////
214.  //
215.  // Version
216.  //

217.  VS_VERSION_INFO VERSIONINFO
218.   FILEVERSION 1,0,0,0
219.   PRODUCTVERSION 1,0,0,0
220.   FILEFLAGSMASK 0x3fL
221.  #ifdef _DEBUG
222.   FILEFLAGS 0x1L
223.  #else
224.   FILEFLAGS 0x0L
225.  #endif
226.   FILEOS 0x4L
227.   FILETYPE 0x1L
228.   FILESUBTYPE 0x0L
229.  BEGIN
230.        BLOCK «StringFileInfo»
231.        BEGIN
232.              BLOCK «080a04b0»
233.              BEGIN
234.                        VALUE «Comments»,
235.                              «Proyecto de Tesis para Nivel Licenciatura de la Carrera: INGENIERÍA EN
236.                              COMPUTACIÓN.\0»
237.              VALUE «CompanyName»,
238.                        «Equipo de Desarrollo : Mª del Rocío Barajas Ibáñez y Said Miranda Rodríguez.
239.                        TESISTAS : UNAM - ENEP Aragón.\0»
240.              VALUE «FileDescription», «Codificador/Decodificador FRACTAL de Imágenes\0»
241.              VALUE «FileVersion», «1.0.0\0»
242.              VALUE «InternalName», «COMPFRAC - SAYRO\0»
243.              VALUE «LegalCopyright», «Copyright © 1997\0» VALUE «OriginalFilename»,    «COMPFRAC.EXE\0»
244.
245.
246.              VALUE «ProductName», «Compresor/Descompresor FRACTAL de Imágenes\0»
247.              VALUE «ProductVersion», «1.0.0\0»
248.        END
```

Archivo *CompFrac.RC*

```
END
BLOCK «VarFileInfo»
BEGIN
        VALUE «Translation», 0x80a, 1200
END
END

#endif    // !_MAC


/////////////////////////////////////////////////////////////////////// //
// Dialog Info
//

IDD_OPCIONESC DLGINIT
BEGIN
IDC_ANCHOIMG, 0x403, 4, 0
0x3532, 0x0036,
        IDC_ANCHOIMG, 0x403, 4, 0
0x3135, 0x0032,
        IDC_TIPOCONTEN, 0x403, 2, 0
0x0030,
        IDC_TIPOCONTEN, 0x403, 2, 0
0x0031,
        IDC_TIPOCONTEN, 0x403, 2, 0
0x0032,
        0
END
/////////////////////////////////////////////////////////////////////// //
// Bitmap
//

IDB_DOWNUP          BITMAP  DISCARDABLE    «res\\bitmap2.bmp»
IDB_DOWND           BITMAP  DISCARDABLE    «res\\bitmap3.bmp» IDB_LOGOCFRAC150      BITMAP
DISCARDABLE     «res\\logo150.bmp»

/////////////////////////////////////////////////////////////////////// //
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
        IDD_CAJACREDITOS, DIALOG
        BEGIN
                RIGHTMARGIN, 288
        END

        IDD_OPCIONESC, DIALOG
        BEGIN
                BOTTOMMARGIN, 236
        END

        IDD_OPCIONESD, DIALOG
        BEGIN
                RIGHTMARGIN, 221
        END
END
#endif    // APSTUDIO_INVOKED

/////////////////////////////////////////////////////////////////////// //
// String Table
//

STRINGTABLE DISCARDABLE
BEGIN
                                        ID_INDICATOR_EXT        «EXT»
```

```
306.                                    ID_INDICATOR_CAPS     «CAP»
307.                                    ID_INDICATOR_NUM      «NUM»
308.                                    ID_INDICATOR_SCRL     «SCRL»
309.                                    ID_INDICATOR_OVR      «OVR»
310.                                    ID_INDICATOR_REC      «REC»
311.  END

312.  STRINGTABLE DISCARDABLE
313.  BEGIN
314.                                    IDS_CREDITOS      «&Creditos de CompFrac...»
315.  END

316.  #endif    // Spanish (Mexican) resources
317.  /////////////////////////////////////////////////////////////////////////


318.  #ifndef APSTUDIO_INVOKED
319.  ////////////////////////////////////////////////////////////////////////// //
320.  // Generated from the TEXTINCLUDE 3 resource.
321.  //
322.  #include «res\CompFrac.rc2»  // non-Microsoft Visual C++ edited resources

323.  #define _AFX_NO_SPLITTER_RESOURCES
324.  #define _AFX_NO_OLE_RESOURCES
325.  #define _AFX_NO_TRACKER_RESOURCES
326.  #define _AFX_NO_PROPERTY_RESOURCES
327.  #include «afxres.rc»  // Standard components

328.  /////////////////////////////////////////////////////////////////////////
329.  #endif    // not APSTUDIO_INVOKED
```

Fin de Archivo CompFrac.RC

Archivo CompFrac.H

```
1.  // CompFrac.h : Archivo de Encabezado Principal para la Aplicacion CompFrac
2.  #ifndef __AFXWIN_H__
3.  #error Incluir 'stdafx.h' antes de incluir este archivo para el Archivo de Encabezado Pre-Compilado(PCH)
4.  #endif
5.  #include «resource.h»          // Símbolos principales
6.  ////////////////////////////////////////////////////////////
7.  // CCompFracApp:
8.  // Ver CompFrac.cpp para la implementacion de esta clase

9.  class CCompFracApp : public CWinApp
10. {
11. public:
12.             CCompFracApp();

13. // Funciones Miembro Sobrecargadas
14.             // ClassWizard sobrecargño las funciones virtuales
15.             //{{AFX_VIRTUAL(CCompFracApp)
16.             public:
17.             virtual BOOL InitInstance();
18.             virtual void WinHelp(DWORD dwDato, UINT nCmd = HELP_CONTEXT);
19. //}}AFX_VIRTUAL

20. // Implementación

21. //{{AFX_MSG(CCompFracApp)
22. // NOTE - the ClassWizard will add and remove member functions here.
23.                 //    DO NOT EDIT what you see in these blocks of generated code !
24.             //}}AFX_MSG
25. DECLARE_MESSAGE_MAP()
26. };


27. ////////////////////////////////////////////////////////////
```

Fin de Archivo CompFrac.H

Archivo CompFrac.CPP

```
1.  // CompFrac.cpp : Clase que define el comportamiento de la aplicación
2.  #include «stdafx.h»          // Acceso a las Clases Fundamentales de Microsoft (MFC)
3.  #include «CompFrac.h»
4.  #include «FracCdlg.h»

5.  #ifdef _DEBUG
6.  #undef THIS_FILE
7.  static char BASED_CODE THIS_FILE[] = __FILE__;
8.  #endif

9.  ////////////////////////////////////////////////////////////
10. // CCompFracApp

11. BEGIN_MESSAGE_MAP(CCompFracApp, CWinApp)
12. //{{AFX_MSG_MAP(CCompFracApp)
13. // NOTE - the ClassWizard will add and remove mapping macros here.
14. //    DO NOT EDIT what you see in these blocks of generated code!
15. //}}AFX_MSG
16. ON_COMMAND(ID_HELP, CWinApp::OnHelp)
17. END_MESSAGE_MAP()

18. ////////////////////////////////////////////////////////////
19. // Construcción de CCompFracApp

20. CCompFracApp::CCompFracApp()
21. {
22. // A REALIZAR : Colocar las inicializaciones pertinentes en InitInstance
23. }

24. ////////////////////////////////////////////////////////////
25. // El primero y el unico objeto CCompFracApp

26. CCompFracApp theApp;

27. ////////////////////////////////////////////////////////////
28. // Inicialización de CCompFracApp

29. BOOL CCompFracApp::InitInstance()
30. {
31. // Inicializacion estándar.
32. // Si no se esta utilizando esta accion y desea reducir el tamaño
33. //  del ejecutable final, deberia remover las lineas de inicializacion
34. // que no se requieran.

35. Enable3dControls();

36. // Establece el color de fondo de la aplicacion (omitido, pero opcional)                    //
37. SetDialogBkColor(RGB(162,224,225));
38.     //SetDialogBkColor();//gris po default

39.     CCompFracDlg dlg;
40.     m_pMainWnd = &dlg;
41.
42.     // nRespuesta pude recibir IDOK, IDCANCEL, IDREINICIAR
43.     // Nota : IDOK no puede ser retornado por DoModal() porque IDOK ya que esta implementado
44. // a través de una función miembro que responde al Click del Botón con ID=IDOK de CompFracDlg

45.     int nRespuesta = dlg.DoModal();

46.     // Debido a que ya esta implementada OnOK() para CCompFracDlg,
47.     // nunca DoModal() regresará dicho valor, después de presionar el Botón Aceptar
```

Archivo *CompFrac.CPP*

```
48.     if (nRespuesta == IDOK)
49.     {
50.             // A REALIZAR : Colocar aquir el código para manejar el mensaje cuando
51.             // el Cuadro de Diálogo recibe OK (esto ya no ocurrirá)
52.     }
53.     else if (nRespuesta == IDCANCEL)
54.     {
55.             // A REALIZAR : Colocar aquir el código para manejar el mensaje cuando
56.             // el Cuadro de Diálogo recibe CANCELAR
57.     }

58.             // Dado que el diálogo ha sido cerrado correctamente, el valor retornado es EXIT_SUCCESS
59.                 // para terminar la aplicación
60.                 return EXIT_SUCCESS;
61.     }

62.     void CCompFracApp::WinHelp(DWORD dwDato, UINT nCmd)
63.     {
64.     // A REALIZAR: Agregar el código especifico aqui y/o llamar a la clase base

65.     //CWinApp::WinHelp(dwDato, nCmd);
66.     CWinApp::WinHelp(0L, HELP_CONTENTS);
67.     }
```

Fin de Archivo CompFrac.CPP

Archivo FracCDlg.H

```
1.  // FracCdlg.h : Archivo de Encabezado
2.  //
3.  ///////////////////////////////////////////////////////////////////// // Diálogo CCompFracDlg
4.  #include «copcion.h»
5.  #include «visorimg.h»
6.  class CCompFracDlg : public CDialog
7.  {
8.  // Construcción
9.  public:
10. CCompFracDlg(CWnd* pPadre = NULL);     // Constructor Estándar
11.
12. // Datos (Estado) del Diálogo
13. //{{AFX_DATA(CCompFracDlg)
14. enum { IDD = IDD_COMPFRAC_MAIN };
15. CString m_LpRegistro;
16. CButton m_btnAceptar;
17. CString m_archEntrada;
18. CString m_archSalida;
19. CButton m_creditos;
20. CButton m_abrearchivo;
21. CButton m_abreayuda;
22. //}}AFX_DATA

23. BOOL m_Compresion;
24. BOOL m_Realizado;

25. // ClassWizard sobrecargó a las funciones virutales //{{AFX_VIRTUAL(CCompFracDlg)
26. protected:
27.         virtual void DoDataExchange(CDataExchange* pDX);     // soporte DDX/DDV

28. //}}AFX_VIRTUAL

29. // Implementación
30. protected:
31. HWND m_hwndEscritorio;
32. RECT m_rcDimsDialogo, m_rcDimsEscritorio;
33. long m_lAnchoDlg, m_lAltoDlg;
34. HICON m_hIcono;
35. COpcion copcion;
36. CDopcion dopcion;
37. // Funciones generadas de mapeo de mensajes
38. //{{AFX_MSG(CCompFracDlg)
39. virtual BOOL OnInitDialog();
40. afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
41. afx_msg void OnPaint();
42. afx_msg HCURSOR OnQueryDragIcon();
43. virtual void OnOK();
44. //afx_msg void OnOpciones();
45. afx_msg void OnMuestracreditos();
46. afx_msg void OnAbrearchivo();
47. afx_msg void OnAyudaWin();
48. //}}AFX_MSG
49.
50. BOOL NomArchNuevo(CString *szNomArch);
51. void LimpiaRutas();
52. DECLARE_MESSAGE_MAP()
53. };
```

Fin de Archivo FracCDlg.H

Archivo FracCDlg.CPP

```
1.  // FracCdlg.cpp : Archivo de Implementación
2.  #include «stdafx.h»          // Acceso a las Clases Fundamentales de Microsoft (MFC) #include «CompFrac.h»
3.  #include «FracCdlg.h»
4.  #include «frac.h»

5.  #ifdef _DEBUG
6.  #undef THIS_FILE
7.  static char BASED_CODE THIS_FILE[] = __FILE__;
8.  #endif

9.  /////////////////////////////////////////////////////////////////
10. // El Cuadro de Diálogo CCredDlg utilizado para mostrar los créditos de la Aplicación

11. class CCredDlg : public CDialog
12. {
13. public:
14. CCredDlg();
15.
16. // Datos (estado) del Diálogo
17. //{{AFX_DATA(CCredDlg)
18.                 enum { IDD = IDD_CAJACREDITOS };
19.                 CBitmapButton m_logoimg;
20. //}}AFX_DATA

21. // Implementación
22. protected:
23.                 virtual void DoDataExchange(CDataExchange* pDX);//soporte DDX/DDV
24.                 //{{AFX_MSG(CCredDlg)
25.                 virtual BOOL OnInitDialog();
26. //}}AFX_MSG
27. DECLARE_MESSAGE_MAP()
28. };

29. CCredDlg::CCredDlg() : CDialog(CCredDlg::IDD)
30. {
31. //{{AFX_DATA_INIT(CCredDlg)
32. //}}AFX_DATA_INIT
33. }

34. void CCredDlg::DoDataExchange(CDataExchange* pDX)
35. {
36. CDialog::DoDataExchange(pDX);
37. //{{AFX_DATA_MAP(CCredDlg)
38.                 DDX_Control(pDX, IDC_BOTONIMG, m_logoimg);
39. //}}AFX_DATA_MAP
40. }

41. BEGIN_MESSAGE_MAP(CCredDlg, CDialog)
42. //{{AFX_MSG_MAP(CCredDlg)
43.                 // No message handlers
44. //}}AFX_MSG_MAP
45. END_MESSAGE_MAP()

46. /////////////////////////////////////////////////////////////////
47. // Manejadores de mensajes de CCredDlg

48. BOOL CCredDlg::OnInitDialog()
49. {
50.                 // A REALIZAR : Agregar incialización del diálogo aqui
51.                 CDialog::OnInitDialog();
52. CenterWindow();
```

```
53.         m_logoimg.LoadBitmaps(«IDB_LOGOCFRAC150»);
54.         m_logoimg.SizeToContent();
55.
56.         return TRUE;
57.         }


58.         ////////////////////////////////////////////////////////
59.         // Diálogo CCompFracDlg

60.         CCompFracDlg::CCompFracDlg(CWnd* pPadre /*=NULL*/)
61.                     : CDialog(CCompFracDlg::IDD, pPadre)
62.         {
63.         //{{AFX_DATA_INIT(CCompFracDlg)
64.                     m_archEntrada = _T(«»);
65.                     m_archSalida = _T(«»);
66.         //}}AFX_DATA_INIT
67.                     // Observar que LoadIcon no requiere un destructor del recurso en Win32
68.                     m_hIcono = AfxGetApp()->LoadIcon(IDR_APPCFRAC1);
69.         m_Compresion = FALSE;
70.         m_Realizado = FALSE;
71.         }

72.         void CCompFracDlg::DoDataExchange(CDataExchange* pDX)
73.         {
74.                     CDialog::DoDataExchange(pDX);
75.                     //{{AFX_DATA_MAP(CCompFracDlg)
76.                     DDX_Control(pDX, IDOK, m_btnAceptar);
77.                     DDX_Text(pDX, IDC_CADNOMBARCHENTR, m_archEntrada);
78.                     DDX_Text(pDX, IDC_CADNOMBARCHSAL, m_archSalida);
79.                     DDX_Control(pDX, IDC_MUESTRACREDITOS, m_creditos);
80.                     DDX_Control(pDX, IDC_ABREARCHIVO, m_abrearchivo);
81.                     DDX_Control(pDX, IDC_AYUDAWIN, m_abreayuda);
82.                     //}}AFX_DATA_MAP
83.         }

84.         BEGIN_MESSAGE_MAP(CCompFracDlg, CDialog)
85.         //{{AFX_MSG_MAP(CCompFracDlg)
86.                     ON_WM_SYSCOMMAND()
87.                     ON_WM_PAINT()
88.                     ON_WM_QUERYDRAGICON()
89.                     //*+ON_BN_CLICKED(IDC_OPCIONES, OnOpciones)
90.                     // botón Opciones INVISIBLE desde el Dialogo Principal
91.                     ON_BN_CLICKED(IDC_MUESTRACREDITOS, OnMuestracreditos)
92.                     ON_BN_CLICKED(IDC_ABREARCHIVO, OnAbrearchivo)
93.                     ON_BN_CLICKED(IDC_AYUDAWIN, OnAyudaWin)
94.         //}}AFX_MSG_MAP
95.         END_MESSAGE_MAP()

96.         ////////////////////////////////////////////////////////
97.         // Manejadores de los mensajes de CCompFracDlg

98.         BOOL CCompFracDlg::OnInitDialog()
99.         {
100.        // A REALIZAR : Agregar inicialización extra aqui
101.        CDialog::OnInitDialog();
102.        CenterWindow();
103.        m_Compresion = TRUE;
104.        m_LpRegistro = AfxRegisterWndClass(CS_CLASSDC |
105.                    CS_DBLCLKS,0,CreateSolidBrush(PALETTERGB(255,255,255)),
106.                    AfxGetApp()->LoadIcon(IDI_APPCFRAC4));
107.        // Agrega el item «Creditos de CompFrac...» al Menu del Sistema
108.        // IDM_CREDITOS debe estar dentro del Rango del los comandos del Sistema.
109.        ASSERT((IDM_CREDITOS & 0xFFF0) == IDM_CREDITOS);
110.        ASSERT(IDM_CREDITOS < 0xF000);
111.                    CMenu* pMenuSist = GetSystemMenu(FALSE);
112.                    CString strMenuCreditos;
113.                    strMenuCreditos.LoadString(IDS_CREDITOS);
114.                    if (!strMenuCreditos.IsEmpty())
```

```
115.        {
116.            pMenuSist->AppendMenu(MF_SEPARATOR);
117.            pMenuSist->AppendMenu(MF_STRING, IDM_CREDITOS, strMenuCreditos);
118.        }

119.        // Incorpora Iconos al Diálogo
120.        SetIcon(m_hIcono, TRUE);    // Icono Grande 32x32
121.        SetIcon(m_hIcono, FALSE);   // Icono Pequeño 16x16

122.        // Elimina Opcion Mover
123.        pMenuSist->RemoveMenu((UINT)1, MF_BYPOSITION);// opcion Mover

124.        // Pone el mensaje en el cuadro superior del Diálogo Principal
125.        CString szMensaje(«INFO : Selecciona la Ruta y Nombre de Archivo, mediante el botón \»Abrir Archivo\»...»);
126.        SetDlgItemText(IDC_OPERACION, szMensaje);
127.        printf(«%s\n»,szMensaje);

128.        // Muestra Créditos Iniciales
129.        MessageBeep(MB_ICONASTERISK);
130.        CCredDlg dlgCred;
131.        dlgCred.DoModal();

132.        //Toma Dims. del Cuadro de Diálogo Principal para opciones de Max. Min y Restaurar
133.        GetWindowRect(&m_rcDimsDialogo);
134.        m_lAnchoDlg = m_rcDimsDialogo.right - m_rcDimsDialogo.left;
135.        m_lAltoDlg = m_rcDimsDialogo.bottom - m_rcDimsDialogo.top;

136.        // Obtiene coords. del escritorio con Funcs. Estándar (no MFCs)
137.        m_hwndEscritorio = ::GetDesktopWindow();
138.        ::GetWindowRect(m_hwndEscritorio, &m_rcDimsEscritorio);

139.        return TRUE;
140.        }

141.        void CCompFracDlg::OnSysCommand(UINT nID, LPARAM lParam)
142.        {
143.        if ((nID & 0xFFF0) == IDM_CREDITOS)
144.        {
145.            CCredDlg dlgCred;
146.            dlgCred.DoModal();
147.        }
148.        else
149.        {
150.            if ((nID & 0xFFF0) == SC_MAXIMIZE) // responde al botón maximizar
151.            {
152.                CDialog::OnSysCommand(nID, lParam);
153.                MoveWindow(0, 0, m_rcDimsDialogo.right - m_rcDimsDialogo.left,
154.                                m_rcDimsDialogo.bottom - m_rcDimsDialogo.top, TRUE);
155.            }
156.            else
157.        { if ((nID & 0xFFF0) == SC_RESTORE) // responde al botón-menu restaurar
158.            {
159.                CDialog::OnSysCommand(nID, lParam);
160.                // se usa esta func mejor que CenterWindow debido a que esta funcion
161.                // no depende del tamaño anterior de la ventana
162.                MoveWindow(( m_rcDimsEscritorio.right - m_lAnchoDlg ) / 2 +2, ( m_rcDimsEscritorio.bottom -
163.                m_lAltoDlg ) / 2 -10, m_lAnchoDlg, m_lAltoDlg, TRUE);
164.                /*      CenterWindow(CWnd::GetDesktopWindow()); */
165.            }
166.            else
167.            {   if ((nID & 0xFFF0) == SC_SIZE) // responde al botón-menu tamaño
168.                    CenterWindow();
169.                else
170.                    CDialog::OnSysCommand(nID, lParam);
171.            }
172.
173.        }
174.    }
175. }
```

```
176.        // Si se agrega un botón de minizar al diálogo se requerira del código
177.        // de abajo para dibujar el icono. Para las aplicaciones MFC que usan el modelo
178.        // documento - vista, esto se realiza automáticamente por la misma estructura.
179.        void CCompFracDlg::OnPaint()
180.        {
181.        if (IsIconic())
182.        {
183.            CPaintDC dc(this); // contexto de dispositivo para redibujar el objeto de clase
184.            SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);
185.
186.            // Centra el icono en el rectangulo cliente
187.            int cxIcono = GetSystemMetrics(SM_CXICON);
188.            int cyIcono = GetSystemMetrics(SM_CYICON);
189.            CRect rcCuadro;
190.            GetClientRect(&rcCuadro);
191.            int x = (rcCuadro.Width() - cxIcono + 1) / 2;
192.            int y = (rcCuadro.Height() - cyIcono + 1) / 2;
193.
194.            // Dibuja el icnono
195.            dc.DrawIcon(x, y, m_hIcono);
196.        }
197.        else
198.        {
199.            CDialog::OnPaint();
200.        }
201.        }

202.        // El sistema llama a este miembro para obtener el cursor a mostrar al momento
203.        // que el usuario arrastra la ventana minimizada.
204.        HCURSOR CCompFracDlg::OnQueryDragIcon()
205.        {
206.        return (HCURSOR) m_hIcono;
207.        }


208.        void CCompFracDlg::OnOK()
209.        {
210.        // A REALIZAR: Agregar validación extra aqui
211.        if (m_archEntrada.IsEmpty() || m_archSalida.IsEmpty())
212.        {   MessageBeep(MB_ICONEXCLAMATION);
213.            MessageBox(«Falta Nombre de Archivo de Entrada o Salida (o ambos).»,
214.                                        «Error Nombre de Archivo», MB_TASKMODAL |
215.                                        MB_ICONEXCLAMATION | MB_OK);
216.            return;
217.        }

218.        if(m_Realizado)
219.        {
220.            CDialog::OnOK();
221.            return;
222.        }

223.        UpdateData(TRUE);

224.        // Construye objeto de ventana de visualización de la imagen
225.        CVisorImg visorImg;
226.        visorImg.CreateEx(WM_PARENTNOTIFY,/*|WS_EX_CLIENTEDGE|WS_EX_CONTEXTHELP,*/
227.         m_LpRegistro,»Procesando...»,/*WS_EX_CONTEXHELP|*/
228.        WS_CAPTION|WS_POPUPWINDOW|WS_MINIMIZEBOX,0,0,256,256,GetSafeHwnd(),0);
229.        visorImg.CenterWindow();

230.        // Esconde ventana de Diálogo Principal, si la ventana de la Imagen es mostrada
231.        if((m_Compresion && copcion.m_mostrar) || (!m_Compresion && dopcion.m_mostrar))
232.                    ShowWindow(SW_HIDE);
233.        // Crea objeto de clase que contiene todo el código para Compresión/Descompresión
234.        CCompFrac *CompFrac = new CCompFrac(GetSafeHwnd(),IDC_ESTADO);
235.        CompFrac->IncorporaNomArch(m_archEntrada, m_archSalida);
236.        if(m_Compresion)
237.        {           if(copcion.m_cambio)
```

Archivo *FracCDlg.CPP*

```
238.                    {
239.                        if(copcion.m_mostrar)
240.                        {
241.                                CompFrac->Comprimir(visorImg.GetSafeHwnd(),copcion.m_tol,
242.                                atoi(copcion.m_altura),atoi(copcion.m_ancho),copcion.m_maxprofrec,
243.                                copcion.m_minprofrec,copcion.m_ticd,atoi(copcion.m_tipo_cont),
244.                                copcion.m_bitsescalc,copcion.m_bitsdesplzc,copcion.m_factescalmax,
245.                                copcion.m_idm,copcion.m_pos, copcion.m_24clasdom,copcion.m_3clasdom, m_lAnchoDlg,
246.                                m_lAltoDlg);
247.                                visorImg.GuardaDC(CompFrac->HContxDsp());
248.                        }
249.                        else
250.                                CompFrac->Comprimir(NULL,copcion.m_tol,atoi(copcion.m_altura),
251.                                atoi(copcion.m_ancho),copcion.m_maxprofrec,
252.                                copcion.m_minprofrec,copcion.m_ticd,atoi(copcion.m_tipo_cont),
253.                                copcion.m_bitsescalc,copcion.m_bitsdesplzc,copcion.m_factescalmax,
254.                                copcion.m_idm,copcion.m_pos, copcion.m_24clasdom, copcion.m_3clasdom,
255.                                m_lAnchoDlg, m_lAltoDlg);
256.                }
257.                    else
258.                    {
259.                        CompFrac->Comprimir(visorImg.GetSafeHwnd());
260.                        visorImg.GuardaDC(CompFrac->HContxDsp());
261.                    }
262.            }
263.            else            // m_Compresion = FALSE (es Descompresión)
264.            {
265.                    if(dopcion.m_cambio)
266.                    {
267.                        if(dopcion.m_mostrar)
268.                        {
269.                                CompFrac->Descomprimir(visorImg.GetSafeHwnd(),dopcion.m_scalsal,
270.                                dopcion.m_numiter, dopcion.m_postproc,dopcion.m_factescalmax,
271.                                dopcion.m_bitsescal,dopcion.m_bitsdesplz,dopcion.m_salida);
272.                                visorImg.GuardaDC(CompFrac->HContxDsp());
273.                        }
274.                        else
275.                                CompFrac->Descomprimir(NULL,dopcion.m_scalsal,
276.                                dopcion.m_numiter,dopcion.m_postproc, dopcion.m_factescalmax,
277.                                dopcion.m_bitsescal,dopcion.m_bitsdesplz,dopcion.m_salida);
278.                    }
279.                    else
280.                    {
281.                        CompFrac->Descomprimir(visorImg.GetSafeHwnd());
282.                        visorImg.GuardaDC(CompFrac->HContxDsp());
283.                    }
284.            }
285.            delete CompFrac;

286.            // Muestra ventana de Diálogo Principal, si la ventana de la Imagen fue mostrada
287.            if((m_Compresion && copcion.m_mostrar) || (!m_Compresion && dopcion.m_mostrar))
288.                    ShowWindow(SW_SHOW);

289.            MessageBeep(MB_OK);
290.            // Inhabilita botón Aceptar, para obligar a Abrir Archivo o Salir
291.            m_btnAceptar.EnableWindow(FALSE);
292.            MessageBox("Operación Ya Realizada.", "Información", MB_OK|MB_ICONINFORMATION);
293.            m_Compresion = FALSE;
294.            m_Realizado = FALSE;
295.    }

296.    /*
297.    void CCompFracDlg::OnOpciones()
298.    {
299.            A REALIZAR : Agregar el código del manejador del control (botón) aqui
300.            if(m_Realizado)
301.              return;
302.            if(m_Compresion)
303.              copcion.DoModal();
304.            else
```

```
305.            dopcion.DoModal();
306.    }       */

307.            void CCompFracDlg::OnMuestracreditos()
308.            {
309.            // A REALIZAR : Agregar el código del manejador del control (botón) aqui
310.            if(m_creditos)
311.            { CCredDlg dlgCred;
312.              dlgCred.DoModal();
313.            }
314.            }

315.            // Esta función se encarga de Abrir un Archivo el cual puede DAT o RAW (Compresión) o
316.            // FIC o TRN (Descompresión), y al determinar su extensión establece que Operación se ha
317.            // de realizar (Compresión o Descompresión) y llama al Cuadro de Opciones correspondiente
318.            // Y DETERMINA a partir del nombre del Archivo de Entrada, el nombre del Archivo de Salida
319.            // Agregando una Extensión Infijn, que indica sus dimensiones : .256  ó  .512
320.            // Asi como un número de dos dígitos postfijo en la base del nombre del Archivo
321.            void CCompFracDlg::OnAbrearchivo()
322.            {
323.            // A REALIZAR : Agregar el código del manejador del control (botón) aqui
324.            BOOL bSalir = FALSE, bArchSal = FALSE;
325.            CString szTemp1, szExtenArch, szMensaje;
326.            int n_pos = 0;
327.            CFileDialog *CajaDlgAbrirArch;
328.            CajaDlgAbrirArch = new CFileDialog(TRUE, NULL,«*.*»,OFN_HIDEREADONLY,
329.            «Archivo RAW-Grises de 8bits (*.RAW)|*.RAW|»//
330.            «Archivo RAW-Grises de 8bits (*.DAT)|*.DAT|»//
331.            «Archivo de Código Fractal (*.FIC)|*.FIC|»//
332.             «Archivo de Código Fractal (*.TRN)|*.TRN|»//
333.             «Todos los Archivos (*.*)|*.*||»);
334.            do {
335.              if(CajaDlgAbrirArch->DoModal() == IDOK)
336.              { //Identifica si el archivo de Entrada es para Comprimir o Descomprimir
337.                        szExtenArch = CajaDlgAbrirArch->GetFileExt();
338.                        if ( !(szExtenArch.IsEmpty()) )
339.                        {       szTemp1 = m_archEntrada = CajaDlgAbrirArch->GetPathName();
340.                                m_archEntrada.MakeLower();
341.                                szTemp1.MakeLower();
342.
343.                                // Verfica si el archivo es para Compresión
344.                                if ( !_stricmp(szExtenArch, «RAW») || !_stricmp(szExtenArch,
345.                                        «DAT»))
346.                          {     m_Compresion = TRUE;
347.                                n_pos = szTemp1.ReverseFind('.');//Ya se esta seguro que si
348.                                // existe el punto '.'
349.                                szTemp1.SetAt(++n_pos, 'F');
350.                                szTemp1.SetAt(++n_pos, 'I');
351.                                szTemp1.SetAt(++n_pos, 'C');
352.                                // agrega texto al Cuadro de Operación.
353.                                SetDlgItemText(IDC_OPERACION, szMensaje = «\t\t\tC O M P R E S I O N»);
354.                                printf(«%s\n»,szMensaje);
355.                                // carga caja de diálogo Opciones Compresión
356.                                if (copcion.DoModal() == IDCANCEL)
357.                                        bSalir = TRUE;
358.                          }
359.                          // Verfica si el archivo es para Descompresión
360.                          if ( !_stricmp(szExtenArch, «FIC») || !_stricmp(szExtenArch, «TRN») )
361.                          {     m_Compresion = FALSE;
362.                                n_pos = szTemp1.ReverseFind('.');//Ya se esta seguro que si
363.                          //existe el punto '.'
364.                                szTemp1.SetAt(++n_pos, 'R');
365.                                szTemp1.SetAt(++n_pos, 'A');
366.                                szTemp1.SetAt(++n_pos, 'W');
367.                                // Agrega texto al Cuadro de Operación.
368.                                SetDlgItemText(IDC_OPERACION, szMensaje = «\t\tD E S C O M P R E S I O
369.                                N»);
370.                                printf(«%s\n»,szMensaje);
371.                                // Carga caja de diálogo Opciones Descompresión
372.                                if (dopcion.DoModal() == IDCANCEL)
```

Archivo *FracCDlg.CPP*

```
373.                                    bSalir = TRUE;
374.                    }

375.                    // Ya sea compresión o Descompresión, si bSalir=FALSO,
376.                    // Copcion o Dopcion retorno IDOK, ecir, no se cancelo la Operación
377.                    if (!bSalir)
378.                    { // Verfica que si existe nombre de archivo numerado disponible
379.                      if (!NomArchNuevo(&szTemp1))
380.                      {           MessageBeep(MB_ICONEXCLAMATION);
381.                                  MessageBox("No es posible encontrar un nombre\n de archivo
382.                              disponible para Guardar."//
383.                                      "\nSerá necesario indicarlo manualmente.",
384.                                      "Información", MB_TASKMODAL | MB_ICONEXCLAMATION |
385.                                          MB_OK);
386.                              CFileDialog *CajaDlgGuardarArch;
387.                              CajaDlgGuardarArch = new CFileDialog(FALSE, (
388.                                  !_stricmp(szExtenArch, "FIC") || !_stricmp(szExtenArch,
389.                                  "TRN") ) ? "RAW"+"FIC"+"*.*",OFN_HIDEREADONLY,
390.                                  ( !_stricmp(szExtenArch, "FIC") ||
391.                                  !_stricmp(szExtenArch, "TRN") ) ?
392.                                  "Archivo RAW-Grises de 8bits (*.RAW)|*.RAW|"//
393.                                  "Archivo RAW-Grises de 8bits (*.DAT)|*.DAT||":
394.                                  "Archivo de Código Fractal (*.FIC)|*.FIC|"//
395.                                  "Archivo de Código Fractal (*.TRN)|*.TRN||");
396.                              if (CajaDlgGuardarArch->DoModal() == IDOK)
397.                              {       bArchSal = TRUE;
398.                                      szTemp1 = CajaDlgGuardarArch->
399.                                      GetPathName();
400.                              }
401.                              else
402.                              {       bArchSal = FALSE;
403.                                      bSalir = TRUE;
404.                              }

405.                                  delete CajaDlgGuardarArch;
406.                              }
407.                              else // Encontró nombre de archivo nuevo numerado
408.                              bArchSal = TRUE;

409.                              if (bArchSal)
410.                              {       m_archSalida = szTemp1;
411.                                      bSalir = TRUE;
412.                                      UpdateData(FALSE);
413.                                          // Asegura que este habilitado el botón Aceptar
414.                                      m_btnAceptar.EnableWindow(TRUE);
415.                                      // Coloca el foco al botón Aceptar
416.                                      m_btnAceptar.SetFocus();
417.                              }
418.                      }
419.                      else  // bSalir = TRUE, copcion o dopcion retornó IDCANCEL
420.                              LimpiaRutas();
421.                  }
422.                  else
423.                  {       MessageBeep(MB_ICONEXCLAMATION);
424.                          MessageBox("Se requiere un archivo con extensión\n como las indicadas"//
425.                           "en Tipo de Archivo.", "Error", MB_TASKMODAL | MB_ICONEXCLAMATION |
426.                          MB_OK);
427.                          bSalir = FALSE;
428.                  }
429.          }
430.          else // CajaDlgAbrirArch->DoModal() retorna IDCANCEL
431.          {
432.                  bSalir = TRUE;
433.                  LimpiaRutas();
434.          }

435.      }while(bSalir == FALSE);

436.      delete CajaDlgAbrirArch;
437.  }
```

```
438.  // Limpia las cuadros de edición de rutas de archivos de Entrada / Salida
439.  void CCompFracDlg::LimpiaRutas()
440.  {
441.          UpdateData(TRUE);
442.          m_archEntrada.Empty();
443.          m_archSalida.Empty();
444.          UpdateData(FALSE);
445.  }


446.  // Busca nombre de archivo numerado no existente
447.  BOOL CCompFracDlg::NomArchNuevo(CString *szNomArch)
448.  {

449.          WIN32_FIND_DATA InfoArchSal;
450.          int numArchivo = -1, nPos1 = 0, nPos2 = 0;
451.          CString szNomArchSalTmp, szTmp1, szDimImg;
452.          char cPenulBase, cUltBase;
453.          BOOL bNuevoArch = FALSE, bW95 = FALSE, bExt2 = FALSE;

454.          // Averigua si esta en Windows 95, para el soporte de nombres largos
455.          if (getenv(«winbootdir»))
456.                          bW95 = TRUE;

457.          szNomArchSalTmp = *szNomArch;

458.          // Averigua la existencia del archivo de Salida hasta encontrar
459.          // el nombre de archivo numerado no existente
460.          do {
461.                          nPos1 = szNomArchSalTmp.ReverseFind('.'); // Busca ubicación del punto de extensión de archivo
462.                          nPos2 = szNomArchSalTmp.ReverseFind('\\'); // Busca ubicación del dir actual '\\' donde esta el
463.                                                          //archivo

464.                          if (bW95)       // Corrobora si esta en Win 95 para agregar/verificar extensión 2 infija (numérica)
465.                  {               szDimImg = szNomArchSalTmp.Mid(nPos1-4, 4);// Probablemente tenga ya incluida la
466.                                                          //dimensión. Formato: \1.nnn.ext
467.                          if( (nPos1-nPos2) >= 6 && (szDimImg.Compare(«.256») == 0 || szDimImg.Compare(«.512») == 0) )
468.                                  nPos1 -= 4;
469.                                  else
470.                                  {
471.                                  if(m_Compresion)// En esta sección debe insertar la Dim. a partir de la inf. del cuadro
472.                                  //de diálogo de compresión
473.                                  {
474.                                  szTmp1 = szNomArchSalTmp.Right(4); // Guarda punto y extensión
475.                                  szNomArchSalTmp.SetAt(nPos1+1,(atoi(copcion.m_ancho)/100)+'0');
476.                                  szNomArchSalTmp.SetAt(nPos1+2,(atoi(copcion.m_ancho)%100)/10+'0');
477.                                  szNomArchSalTmp.SetAt(nPos1+3,(atoi(copcion.m_ancho)%10)+'0');
475.                                  szNomArchSalTmp += szTmp1; //Concatena punto y extensión
476.                                  }
477.                                  }
478.                                  // de esta sección la cadena ya debe contener extensión numérica
479.                                  bExt2 = TRUE;
480.                          }

481.                          if (FindFirstFile(szNomArchSalTmp, &InfoArchSal) != INVALID_HANDLE_VALUE)
482.                          // Si entra en esta sección entonces SI existe el archivo
483.                          {
484.                                  cPenulBase = szNomArchSalTmp.GetAt(nPos1-2);
485.                                  cUltBase = szNomArchSalTmp.GetAt(nPos1-1);
486.
487.                                  // Verifica que este archivo no sea numerado en la base
488.                                  if ( !( isdigit(cPenulBase) && isdigit(cUltBase) ) || (nPos1 == nPos2 +2) )
489.                                  {
490.                                  if ( (nPos1-nPos2) >= 2 && (nPos1-nPos2) <= (bW95 ? 65:9))
491.                                  {               szTmp1 = szNomArchSalTmp.Right((bW95 && bExt2) ? 8 : 4); // Guarda
492.                                                          //punto y extensión
493.                                  if (nPos1-nPos2 == (bW95 ? 64:8)) nPos1—; // base=7 caracteres, sobreescribe
494.                          //un dígito y el otro lo agrega
495.                                  if (nPos1-nPos2 == (bW95 ? 65:9)) nPos1 -= 2; // base=8 caracteres,
```

```
496.                                    //sobreescribe los dos dígitos
497.                            numArchivo++; // Comienza la cuenta de archivos desde 00
498.                            szNomArchSalTmp.SetAt(nPos1++,(numArchivo/10)+'0');
499.                            szNomArchSalTmp.SetAt(nPos1++,(numArchivo%10)+'0');
500.                            do{
501.                                    szNomArchSalTmp.SetAt(nPos1++, ' ');
502.                            }while(nPos1+1 <= szNomArchSalTmp.GetLength());

503.                            szNomArchSalTmp.TrimRight();// Elimina espacios vacios de más a la
504.                                    //derecha
505.                            szNomArchSalTmp += szTmp1; // Concatena punto y extensión
506.                            }
507.                            }
508.                            else  // El útilmo caracter no es alfabético y continua la cuenta
509.                            {       numArchivo++; //Inicia la cuenta desde 00. MEJORA : podria continuar la cuenta a
510.                                    //partir del dígito actual del archivo
511.                                    nPos1 -= 2;
512.                                    szNomArchSalTmp.SetAt(nPos1++,(numArchivo/10)+'0');
513.                                    szNomArchSalTmp.SetAt(nPos1,(numArchivo%10)+'0');
514.                            }
515.                    }
516.                    else
517.            {               bNuevoArch = TRUE;
518.                            *szNomArch = szNomArchSalTmp;
519.                    }

520.                    }while( !bNuevoArch );

521.                    return TRUE;
522.    }


523.    // AYUDA desde el control botón «Ayuda»
524.    void CCompFracDlg::OnAyudaWin()
525.    {
526.                    // A REALIZAR : Agregar el código del manejador del control (botón) aqui

527.                    //CWinApp::WinHelp(dwDato, nCmd);
528.                    WinHelp(0L, HELP_CONTENTS);
529.    }
```

Fin de Archivo FracCDlg.CPP

Archivo Frac.H

```
1.  // frac.h : Archivo de Encabezado de la clase que contiene todo el código para compresión / descompresión
2.  //      tomados del libro «Compresión Fractal de Imágenes» de Yuval Fisher.
3.  ////////////////////////////////////////////////////////////
4.  #define DIMS_IMG1    256
5.  #define DIMS_IMG2    512
6.  #define NIVELES_GRIS 255
7.  #define limita(a)   ((a) < 0.0 ? 0 : ((a)>255.0? 255 : a))
8.  #define intercambia(a,b,TYPE)        {TYPE _temp; _temp=b; b=a; a= _temp;}


9.  ////////////////////////////////////////////////////////////
10. class CCompFrac      :public CCmdTarget
11. {
12. // Constructor
13. public:
14.         CCompFrac(HWND hwnd = NULL,short id = -1,BOOL salida = TRUE);
15.         ~CCompFrac();
16. public:
17.         // recibe los nombnre de los archivos (de entrada y salida)
18.         void IncorporaNomArch(CString szarchEntrada, CString szarchSalida);
19.         void Comprimir(HWND hwndEntrada = NULL, double tolerancia = 8.0, int Tamh =256, int Tamv =256, int
20.         max_part = 6, int min_part = 4, int dom_taminterv = 1, int dom_intervmult = 0, int bits_escalam = 5, int
21.         bits_desplz = 7, double max_escala = 1.0, int divisor = 0, BOOL positivo = FALSE, BOOL dom24 = FALSE,
22.         BOOL dom3 = FALSE, long lAnchoDlg = 321, long lAltoDlg = 388);
23.         void Descomprimir(HWND hwndSal = NULL,int escala=1.0, int iteraciones=10, BOOL procesa = TRUE, double
24.                 maxima_escala = 1.0, int bits_escalam = 5, int bits_desplz = 7, BOOL part_sal = FALSE);
25.         HDC HContxDsp(void);
26.         HBITMAP HManejBitmap(void);

27. private:
28.         // Salida general de mesajes
29.         void Mensaje(CString mensaje);
30.         // Lee un archivo de imagen RAW y lo coloca en memoria, se asume que
31.         // ya se ha creado el espacio en memoria
32.         BOOL CCompFrac::LeeArchRAW(int tamh, int tamv, CString szNomArch);
33.         // Escribe la imagen desde memoria a Disco, un byte a la vez
34.         BOOL CCompFrac::EscribArchRaw(int tamh, int tamv, CString szNomArch);
35.         // retorna el valor de pixel promedio de una región de la imagen.
36.         void Promedio(int x, int y, int tamx, int tamy, double *sumpix, double *sumpix2); //Retorna el valor de pixel
37.                                                                 //promedio de una región de la imagen.
38.         // Esta rutina difiere de la anterior en un pequeña forma. Esta no promedia
39.         // sub-imágenes de 2x2 pixeles. Esto es necesario para clasificar rangos
40.         // mejor que dominios donde sea necesario en la partición.
41.         void Promedio1(int x, int y, int tamx, int tamy, double *sumpix, double *sumpix2); // Toma una región de la
42.                                                                 //imagen en x,y y la clasifica.
43.         // Los cuatro cuadrantes de la región son ordenados from de mayor a menor
44.         // valor promedio de brillo, entonces se rota en uno de las tres formas de
45.         // orietación canónicas posibles con el cuadro de mayor brillo
46.         // en la esquina superior izquierda.
47.         // La rutina retorna dos indices que son numeros de clases: primerclas
48.         // y segundaclas; también la operación de simetría que lleva al cuadro hacia
49.         // la posición canónica, y sumpix y sumpix^2 de los valores de pixel
50.         void Clasifica(int x, int y, int tamx, int tamy, int *primerclas, int *segundaclas,
51.                                 int *simoper, double *sumpix, double *sumpix2, int tipo);
52.         // Calcula sum and sum^2 de los valores de pixel en los dominios para posteriormente
53.         // usarlos en el calculo del rms. Debido a que un dominio es comparado
54.         // con muchos rangos, al hacer esto solo una vez se ahorran muchos calculos.
55.         // Esta rutina tambien llena una estructura de lista con los dominios
56.         // asi como estos fueron clasificados y asigna la memoria en una matriz para la informacion
57.         // de los dominios.
58.         BOOL Calcula_Sums(int tamh, int tamv);
59.         // Guarda valores usando el tamaño de bits y el archivo de salida
60.         long int EscrAEncab(int tam, long int valor);
61.         // Compara un rango con un dominio y regresa el valor rms y la cuantizacion
```

```
62.          //del escalamiento, asi como los valores de desplazamiento (pialfa, pibeta).
63.          double Compara(int atx, int aty, int tamx, int tamy, int prof, double rsum, double rsum2, int dom_x,  int dom_y,
64.                         int op_sim, int *pialfa, int *pibeta);
65.          // Recursivamente particiona una imagen, para calcular las mejores transformadas
66.          void Quadtree(int atx, int aty, int tamx, int tamy, double tol,int prof, HWND hdcSal);
67.          // Particiona recursivamente una imagen, al encontrar los cuadrados contenidos mas grandes
68.          // y llama a la rutina de quadtree que codifique ese cuadrado.
69.          // Esto permite codificar la imagen rectangular mas facilmente.
70.          void Particiona_Imagen(int atx, int aty,int tamh, int tamv, double tol,HWND hdcSal);
71.          //  Lee los valores del encabezado del archivo de entrada, utilizanco el tamaño de bits
72.          long LeeDeEncab(int tam);
73.          // Lee la información de la transformación desde el archivo de entrada.
74.          // Esta es una rutina recursiva a la cual el arbol de recursión prosigue a la
75.          // terminacion de la recursión por el método de codificación.
76.          void Lee_Transformaciones(int atx, int aty, int tamx, int tamy, int prof);
77.          // Aplica las transformaciones un vez a una imagen negra, inicialmente
78.          int Aplica_Transformaciones(HWND hdcSal, CString sTitulo);
79.          // Particiona recursivamente una imagen, al encontrar el cuadrado contenido mas grande
80.          // y llamar a Lee_Transformationes .
81.          void Particiona_Imagen(int atx, int aty,  int tamh, int tamv );
82.          // Revisa la imagen y promedia los limites de transofrmacion.
83.          void Suaviza_Imagen(HWND hdcSal);
84.          // Muestra la iteracion sobre el dispositivo de salida
85.          void CCompFrac::Mostrar(HWND pDest,int tamh,int tamv, CString sTitulo);
86.          // Reinicia los valores para Compresion y Descompresion
87.          void Restablece(void);

88.    private:
89.          int m_ptr; // cuantos bits son guardados o leidos en total hasta el momento
90.          int m_sum; // acumula el valor a escribir al archivos de salida, o leer desde el archivo de entrada
91.          long int m_num_bytes_empaq;
92.          double m_valrms;

93.          // Tabla de Color
94.          COLORREF  m_Tablacolor[NIVELES_GRIS];
95.          // Manejador para el contexto salida de la imagen
96.          HWND m_hwnd;
97.          HDC m_MemDc;
98.          HBITMAP m_bitmap;
99.          // Ubicacion de la cadena sobre el dispositivo de salida
100.         short m_id;
101.         // Bandera booleana para la cadena de salida hacia la terminal o ventana de salida
102.         BOOL m_salida;
103.         //Archivo de entrada (compresión)
104.         FILE *m_ArchEntr;
105.         CString m_szNomArchEntr;
106.         // Archivo de salida (descompresión)
107.         FILE *m_ArchSal;
108.         CString m_szNomArchSal;

109.         unsigned char **m_imagen;
110.         unsigned char **m_imagenTemp;
111.                              double m_max_escala;    // máximo factor de escala de nivel permitible
112.                              int m_bits_escalam;     // número de bits utilizados para almacenar el factor de escala
113.                              int m_bits_desplz; // número de bits utilizados para almacenar el desplazamiento—
114.                              int m_tamh;       // El tamaño horizontal de la imagen de entrada
115.         int m_tamv;       // El Tamaño vertical
116.         int m_factorescala;// Factor de escala —scalefactor
117.         int m_tam;                    // Imagen cuadrada mas grande que cabe en la imagen
118.         int m_min_part;               // Part. min y max determinan un rango de
119.         int m_max_part;               // Tamaños de Rango desde tamh>>min hasta tamh>>max
120.         int m_dom_taminterv;          // Densidad de los dominios relativa al tamaño
121.         int m_dom_intervmult;              // Bandera para m_dom_intervmult como multiplo o divisor
122.         int m_tipo_dom;           //m_dom_type;  // Método de generacion del contenedor de dominios (0,1,2)—
123.         int *m_numDomh;      //*m_no_h_domains;  // Número de dominios horizontal.
124.         int *m_intrv_h_dom;//*m_domain_hstep;// Tam. interv. de densidad de dominio.
125.         int *m_intrv_v_dom;//*m_domain_vstep;  // Tam. interv. de densidad de dominio.
126.         int *m_bits_necesarios;//*m_bits_needed;  // Número de bits para codificar la pos. del dom
127.         int m_ialfa_cero;   //m_zero_ialpha;// La Const. ialfa para cuando alfa=0
128.         BOOL m_part_sal;                              // Bandera para la salida del particionamiento
```

```
129.        int m_max_exponente;//m_max_exponent; // La potencia max de los 2 lados de la imagen cuadrada
130.                                              //que cabe en la imagen de entrada.
131.        double  **m_imagen_dividida[4];//**m_domimage[4];  // Imagen de entrada particionada usada para las
132.                                              //operaciones con dominios
133.         BOOL          m_solo_positivo;              // Bandera para especificar escalamiento positivo
134.        BOOL          m_subclase_de_busq;  //m_subclass_search; // Bandera para especificar las clases de
135.                                              //busqueda
136.        BOOL    m_clase_completa_de_busq;  //m_fullclass_search;// Bandera para especificar las clases de
137.                                              //busqueda
138.        int           m_transformada_clase[2][24];
139.        int           m_transformada_rotacion[2][8];//m_rot_transform[2][8];

140.        struct nodo_transformacion
141.        {
142.                      int rngx,rngy;                  // En una transformación la posicion del rango y tamaño
143.                      int tamx, tamy;
144.                      int rrx,rry;
145.                      int                             domx,domy;                  // Posición del dominio.
146.                      int oper_sim;                                   // Operación de simetria usada en la tranf.
147.                      int prof;                       // Posfundidad en una particion quadtree.
148.                      double escala, desplz;  // valores de escalamiento y desplazamiento
149.                      struct nodo_transformacion *sig;      // Puntero a la sig. tranf. de la lista
150.        }m_transformaciones,*m_ptransf;
151.         struct pixeles_dominio
152.         {
153.                                                  // Este es un arreglo de indices para el arbol
154.                      int dom_x, dom_y;       // el cual es asignado dinamincamente. El primer indice es
155.                      double sum,sum2;        // el tam de dominio, los otros dos son su
156.                      int sym;                // posición. Tambien contiene sum y sum al cuadadrado
          };

157.        struct datos_dominio
158.        {
159.         int *num_doms_h;                        // Num de doms horizontales por cada intervalo.
160.         int *num_doms_v;                  // Num de doms verticales por cada intervalo.
161.         int *tam_h_dom;                         // Tam del dominio.
162.         int *tam_v_dom;                   // Tam del dominio.
163.         int *intrv_h_dom;                 // Densidad de los dominios.
164.         int *intrv_v_dom;                       // Densidad de los dominios.
165.         struct pixeles_dominio ***pixel;    // Los valores de pixel en los dominios,
166.        }m_dominio;                              // se calculan solo una vez.

167.        struct dominios_clasificados
168.        {                                                       // Esta es una lista que contiene
169.          struct pixeles_dominio *infodom;                  // punteros a los datos de dominio
170.          struct dominios_clasificados *sig;          // de la estructura de arriba. Hay
171.        } **m_info_dominio[3][24];                    // tres clases con 24 subclases
172.                                                        // Al utilizar esta arreglo, solo
173.                                                        // los dominios y rangos de la misma
174.                                                        // clases son comparados..
175.                                                        // El primer puntero apunta al
176.                                                        // tam de dominio, el segundo lo hace a
177.                                                        //  la lista de dominios.

178.  };
```

Fin de Archivo Frac.H

Archivo Frac.CPP

```
1.  // Frac.cpp : Archivo de implementación de la clase que contiene todo el código para compresión / descompresión
2.  //tomados del libro «Compresión Fractal de Imágenes» de Yuval Fischer.
3.  ////////////////////////////////////////////////////////

4.  #include «stdafx.h»      // Acceso a las Clases Fundamentales de Microsoft (MFC)
5.  #include «io.h»
6.  #include «math.h»
7.  #include «stdlib.h»
8.  #include «frac.h»
9.  #include <fcntl.h>
10. #include <errno.h>
11. #include <sys/types.h>
12. #include <sys/stat.h>

13. ////////////////////////////////////////////////////////
14. // Implementacion de la Clase CCompFrac
15. CCompFrac::CCompFrac(HWND hwnd,short id,BOOL salida)
16. {
17.             //Establece dispositivo de salida de pantalla
18.             m_hwnd = hwnd;
19.             m_id = id;
20.             m_salida = salida;
21.             // inicializa contador de num bytes escritos en archivo FIC
22.             m_num_bytes_empaq = 0L;
23.             // Vars. llamadas por LeeEncab()/EscribEncab() para leer/escribir al archivo de entrada/salida
24.             m_ptr = 1;
25.             m_sum = 0;
26.             // valor rms final para la Ventana de Estado m_valrms = 0.0;
27.           }


28.           CCompFrac::~CCompFrac()
29.           {
30.             // Elimina el Contexto de Dispositivo
31.             DeleteDC(m_MemDc);
32.           }

33.           void CCompFrac::Restablece()
34.           {
35.             m_tamh = -1;
36.             m_tamv = -1;
37.             m_subclase_de_busq = 0,
38.            m_clase_completa_de_busq = 0,
39.             m_solo_positivo = 0;
40.             m_dom_intervmult = 0;
41.             m_tipo_dom = 0;
42.             m_max_escala = 1.0;
43.             m_numDomh = NULL;
44.             m_intrv_h_dom = NULL;
45.             m_intrv_v_dom = NULL;
46.             m_bits_necesarios  = NULL;
47.             m_part_sal=FALSE;
48.             m_imagen = NULL;
49.             m_imagenTemp = NULL;
50.             // m_transformada_clase proporciona las transformadas entre números de clasificación
51.             // para valores negativos de escalamiento, esto suede cuando el valor mayor de brillo se convierte en el
52.             //valor más obscuro, etc...
53.             m_transformada_clase[0][0] = 23;
54.             m_transformada_clase[0][1] = 17;
55.             m_transformada_clase[0][2] = 21;
56.             m_transformada_clase[0][3] = 11;
57.             m_transformada_clase[0][4] = 15;
58.             m_transformada_clase[0][5] =  9;
59.             m_transformada_clase[0][6] = 22;
60.             m_transformada_clase[0][7] = 16;
```

```
61.             m_transformada_clase[0][8] = 19;
62.             m_transformada_clase[0][9] =  5;
63.             m_transformada_clase[0][10] = 13;
64.             m_transformada_clase[0][11] =  3;
65.             m_transformada_clase[0][12] = 20;
66.             m_transformada_clase[0][13] = 10;
67.             m_transformada_clase[0][14] = 18;
68.             m_transformada_clase[0][15] =  4;
69.             m_transformada_clase[0][16] =  7;
70.             m_transformada_clase[0][17] =  1;
71.             m_transformada_clase[0][18] = 14;
72.             m_transformada_clase[0][19] =  8;
73.             m_transformada_clase[0][20] = 12;
74.             m_transformada_clase[0][21] =  2;
75.             m_transformada_clase[0][22] =  6;
76.             m_transformada_clase[0][23] =  0;
77.             m_transformada_clase[1][0] =  16;
78.             m_transformada_clase[1][1] =  22;
79.             m_transformada_clase[1][2] =  10;
80.             m_transformada_clase[1][3] =  20;
81.             m_transformada_clase[1][4] =  8;
82.             m_transformada_clase[1][5] =  14;
83.             m_transformada_clase[1][6] =  17;
84.             m_transformada_clase[1][7] =  23;
85.             m_transformada_clase[1][8] =  4;
86.             m_transformada_clase[1][9] =  18;
87.             m_transformada_clase[1][10] =  2;
88.             m_transformada_clase[1][11] = 12;
89.             m_transformada_clase[1][12] = 11;
90.             m_transformada_clase[1][13] = 21;
91.             m_transformada_clase[1][14] =  5;
92.             m_transformada_clase[1][15] = 19;
93.             m_transformada_clase[1][16] =  0;
94.             m_transformada_clase[1][17] =  6;
95.             m_transformada_clase[1][18] =  9;
96.             m_transformada_clase[1][19] = 15;
97.             m_transformada_clase[1][20] =  3;
98.             m_transformada_clase[1][21] = 13;
99.             m_transformada_clase[1][22] =  1;
100.            m_transformada_clase[1][23] =  7;
101.
102.            // m_transformada_rotacion proporciona las rotaciones para los dominios con escalamientos negativos.
103.            m_transformada_rotacion[0][0] = 7;
104.            m_transformada_rotacion[0][1] = 4;
105.            m_transformada_rotacion[0][2] = 5;
106.            m_transformada_rotacion[0][3] = 6;
107.            m_transformada_rotacion[0][4] = 1;
108.            m_transformada_rotacion[0][5] = 2;
109.            m_transformada_rotacion[0][6] = 3;
110.            m_transformada_rotacion[0][7] = 0;
111.            m_transformada_rotacion[1][0] = 2;
112.            m_transformada_rotacion[1][1] = 3;
113.            m_transformada_rotacion[1][2] = 0;
114.            m_transformada_rotacion[1][3] = 1;
115.            m_transformada_rotacion[1][4] = 6;
116.            m_transformada_rotacion[1][5] = 7;
117.            m_transformada_rotacion[1][6] = 4;
118.            m_transformada_rotacion[1][7] = 5;
119.            }

120.            // recibe los nombnre de los archivos (de entrada y salida)
121.            void CCompFrac::IncorporaNomArch(CString szarchEntrada, CString szarchSalida)
122.            {
123.                  //Nombre y ruta completa de archivo de entrada
124.                  m_szNomArchEntr = szarchEntrada;
125.                  // Nombre y ruta completa de archivo de Salida
126.                  m_szNomArchSal = szarchSalida;
127.    }
128.
129.    void CCompFrac::Mensaje(CString mensaje)
130.    {
```

Archivo *Frac.CPP*

```
131.        if(!m_salida)
132.            return;
133.        if(m_hwnd != NULL)
134.        {
135.                if(m_id < 0)
136.                {
137.                        HDC nDC = GetDC(m_hwnd); TEXTMETRIC lpMedidasTxt;
138.                        GetTextMetrics(nDC,&lpMedidasTxt);
139.                        TextOut(nDC,0,abs(m_id—)*lpMedidasTxt.tmHeight,mensaje,mensaje.GetLength());
140.            }
141.            else
142.                        SetDlgItemText(m_hwnd,m_id,mensaje);
143.        }
144.        printf(«%s\n»,mensaje);
145.  }

146.  BOOL CCompFrac::EscribArchRaw(int tamh, int tamv, CString szSal)
147.  {
148.        int i,j;
149.        CString msg;
150.        // Abre archivo de salida (archivo Descompactado)
151.        if ((m_ArchSal = fopen(szSal, «wb»)) == NULL)
152.        {
153.                Mensaje(«Error: No puede abrirse o crearse el archivo de Salida»);
154.                return FALSE;
155.        }
156.        // Escribe el archivo a disco, un byte a la vez
157.        // (Existe un error en la función _write de VC 2.0)
158.        for(i =0; i<tamh;i++)
159.            for(j=0;j<tamv;j++)
160.                        if((fputc(m_imagen[i][j],m_ArchSal)) != m_imagen[i][j])
161.                        {
162.                                msg = «Error:  No puede escribirse al disco (podría estar lleno)»; Mensaje(msg);
163.                                return FALSE;
164.                        }
165.
166.        msg.Format(« %d PIXELES ESCRITOS en %s.», tamh*tamv, szSal);
167.        Mensaje(msg);
168.        fclose(m_ArchSal);
169.        return TRUE;
170.  }

171.  BOOL CCompFrac::LeeArchRAW(int tamh, int tamv, CString szNomArch)
172.  {
173.        int i,manejArch;
174.        CString msg;

175.        if((manejArch = _open(szNomArch,_O_BINARY)) == -1)
176.        {
177.            if(manejArch == EACCES)
178.                    msg = «Error: Se intentó abrir un archivo de sólo lectura para escribir en él, o el modo"//
179.                            " de compartición del archivo no permite las operaciones especificadas, o la ruta"//
180.                            " dada no indica un archivo»;
181.            if(manejArch == EMFILE)
182.                    msg = «Error: No existen manejadores de archivo disponibles (hay demasiados archivos"//
183.                                    " abiertos)»;
184.            if(manejArch == ENOENT)
185.                            msg = «Error:  Archivo o ruta no encontrada»;
186.            if(msg.IsEmpty())
187.                            msg = «Error: Un Error inesperado ha ocurrido mientras se abre el archivo»;
188.            Mensaje(msg);
189.            return FALSE;
190.        }
191.        for(i=0;i<tamh; i++)
192.        {   if((_read(manejArch,m_imagen[i],tamv)) != tamv)
193.            {
194.                msg = «Error: El archivo está dañado, no puede leerse»;
195.                Mensaje(msg);
196.                return FALSE;
197.            }
```

```
198.                msg.Format("INFO : %d pixeles leidos de %s (%dx%d).", tamh*tamv,szNomArch,tamh,tamv);
199.                Mensaje(msg);

200.          }

201.          _close(manejArch);
202.          return TRUE;
203.    }

204.    HDC CCompFrac::HContxDsp()
205.    {
206.          // Maneja del contexto de Dispositivo actual. Si
207.          // el Contexto es proporcionado, este contendrá // la imagen final
208.          return(m_MemDc);
209.    }

210.    HBITMAP CCompFrac::HManejBitmap()
211.    {
212.          return(m_bitmap);
213.    }

214.    void CCompFrac::Mostrar(HWND pDest,int tamh,int tamv, CString sTitulo)
215.    {
216.          int color;
217.          SetWindowText(pDest,sTitulo);
218.          for (int domx = 0 ; domx < tamh; domx++)
219.                for (int cx = 0 ; cx < tamv ;cx++ )
220.                {
221.                      color = m_imagen[domx][cx]; SetPixel(m_MemDc,cx,domx,m_Tablacolor[color]);
222.                }

223.          BitBlt(GetDC(pDest),0,0,tamh,tamv,m_MemDc,0,0,SRCCOPY);
224.    }

225.    ////////////////////////////////////////////////////////////////////
226.    // Codificador y Decodificador de Dominio Publico
227.    ////////////////////////////////////////////////////////////////////
228.    void CCompFrac::Comprimir(HWND hwndEntrada, double tolerancia, int Tamh, int Tamv, int max_part, int min_part,
229.    int dom_taminterv, int dom_intervmult, int bits_escalam, int bits_desplz, double max_escala, int divisor, BOOL
230.    positivo, BOOL dom24, BOOL dom3, long lAnchoDlg, long lAltoDlg)
231.    {
232.          int i,j,k;

233.          Restablece(); // Reinicia todos los valores
234.          m_tamh = Tamh;                  //m_hsize = hsize;
235.          m_tamv = Tamv;                  //m_vsize = vsize;
236.          m_min_part = min_part;                //m_min_part = min_part;
237.          m_max_part = max_part;                //m_max_part = max_part;
238.          m_dom_taminterv = dom_taminterv;              //m_dom_step = dom_step;
239.          m_bits_escalam = bits_escalam;//m_s_bits = scaling_bit;
240.          m_bits_desplz = bits_desplz;  //m_o_bits = offset_bit;
241.          m_max_escala = max_escala; //m_max_scale = max_scale;
242.          m_dom_intervmult = dom_intervmult;      //m_dom_step_type = dom_step_type;
243.          m_dom_intervmult = divisor; //m_dom_step_type = divisor;
244.          m_solo_positivo = positivo;     //m_only_positive = positive;
245.          m_subclase_de_busq = dom24;//m_subclass_search = d24;
246.          m_clase_completa_de_busq = dom3;              //m_fullclass_search = d3;
247.
248.          CString msg;
249.
250.          // Revisa si las cadenas de nombre de archivo estan vacias
251.          if((m_szNomArchEntr.IsEmpty()) || (m_szNomArchSal.IsEmpty()))
252.          {
253.                Mensaje("Error: Falta nombre de archivo, ya sea de Entrada o Salida");
254.                return;
255.          }

256.          // Comienza Cursor de reloj de arena (omitido, pero opcional)
257.    /*    if(hwndEntrada != NULL)
```

```
258.            BeginWaitCursor(); */

259.        // Asigna el espacio de memoria para la imagen de entrada. /////————inicio
260.        unsigned char *reng_imagen;
261.        m_imagen = (unsigned char **)malloc((m_tamv)*sizeof(unsigned char *));
262.        reng_imagen = (unsigned char *)malloc((long)(m_tamh)*(long)(m_tamv)*sizeof(unsigned char));
263.        if (reng_imagen == NULL)
264.        {
265.            Mensaje(«Error: Memoria para la Imagen, agotada»);
266.            return;
267.        }
268.        for (i = 0; i<m_tamv; ++i, reng_imagen += m_tamh)
269.            m_imagen[i] = reng_imagen;          /////————fin /////————inicio

270.        double *reng_imagendom0;
271.        m_imagen_dividida[0] = (double **)malloc((m_tamv/2)*sizeof(double *));
272.        reng_imagendom0 = (double *)malloc((long)(m_tamh/2)*(long)(m_tamv/2)*sizeof(double));
273.        if (reng_imagendom0 == NULL)
274.        {
275.            Mensaje(«Error: Memoria para Dominio agotada»);
276.            return;
277.        }
278.        for (i = 0; i<m_tamv/2; ++i, reng_imagendom0 += m_tamh/2)
279.            m_imagen_dividida[0][i] = reng_imagendom0;      /////————fin /////————inicio

280.        double *reng_imagendom1;
281.        m_imagen_dividida[1] = (double **)malloc((m_tamv/2)*sizeof(double *));
282.        reng_imagendom1 = (double *)      malloc((long)(m_tamh/2)*(long)(m_tamv/2)*sizeof(double));
283.        if (reng_imagendom1 == NULL)
284.        {
285.            msg = «Error: Memoria para Dominio agotada»;
286.            Mensaje(msg);
287.            return;
288.        }
289.        for (i = 0; i<m_tamv/2; ++i, reng_imagendom1 += m_tamh/2)
290.            m_imagen_dividida[1][i] = reng_imagendom1;      /////————fin /////————inicio

291.        double *reng_imagendom2;
292.        m_imagen_dividida[2] = (double **)malloc((m_tamv/2)*sizeof(double *));
293.        reng_imagendom2 = (double *)      malloc((long)(m_tamh/2)*(long)(m_tamv/2)*sizeof(double));
294.        if (reng_imagendom2 == NULL)
295.        {
296.            msg = «Error: Memoria para Dominio agotada»;
297.            Mensaje(msg);
298.            return;
299.        }
300.        for (i = 0; i<m_tamv/2; ++i, reng_imagendom2 += m_tamh/2)
301.            m_imagen_dividida[2][i] = reng_imagendom2;      /////————fin /////————inicio

302.        double *reng_imagendom3;
303.        m_imagen_dividida[3] = (double **)malloc((m_tamv/2)*sizeof(double *)); reng_imagendom3 = (double *)
304.        malloc((long)(m_tamh/2)*(long)(m_tamv/2)*sizeof(double));
305.        if (reng_imagendom3 == NULL)
306.        {
307.                    msg = «Error: Memoria para Dominio agotada»;
308.                    Mensaje(msg);
309.                    return;
310.    }
311.
312.    for (i = 0; i<m_tamv/2; ++i, reng_imagendom3 += m_tamh/2)
313.        m_imagen_dividida[3][i] = reng_imagendom3;      /////————fin

314.    // Las particiones max_ y min_ son variables, asi que estan deben ser asignadas dinamicamente

315.    m_bits_necesarios = (int *)malloc(sizeof(int)*(1 + m_max_part - m_min_part));

316.    if(!LeeArchRAW(Tamh,Tamv,m_szNomArchEntr))
317.        return;
```

```
318.            // Asigna memoria para los datos de dominio y los inicializa
319.            if(!Calcula_Sums(m_tamh,m_tamv))
320.                    return;

321.            if ((m_ArchSal = fopen(m_szNomArchSal, «wb»)) == NULL)
322.            {
323.                    msg = «Error: No puede abrirse archivo de Salida»;
324.                    Mensaje(msg);
325.                    return;
326.            }

327.            // Pone algunos datos importantes en el archivo de salida
328.            EscrAEncab(4,(long)m_min_part);
329.            EscrAEncab(4,(long)m_max_part);
330.            EscrAEncab(4,(long)m_dom_taminterv);
331.            EscrAEncab(1,(long)m_dom_intervmult);
332.            EscrAEncab(2,(long)m_tipo_dom);
333.            EscrAEncab(12,(long)m_tamh);
334.            EscrAEncab(12,(long)m_tamv);

335.            // Este es un valor contabilizado en cero escalamiento.. necesario posteriormente //*conversion forzada
336.            m_jalfa_cero = (int)(0.5 + (m_max_escala)/(2.0*m_max_escala)*(1 << m_bits_escalam));

337.            // La siguiente rutina toma una imagen rectangular y llama a la
338.            // rutina «quadtree» para codificar la suma de imágenes cuadradas en ella.
339.            // la tolerancia es un parámetro utilizado debido a que para algunas aplicaciones
340.            // puede ser necesario comprimir diferentes regiones de la imagen, a diferentes tolerancias

341.            Mensaje(«Codificando la imagen.....»);
342.            if(hwndEntrada != NULL)
343.            {
344.                    // Elimina el dispositivo de contexto anterior
345.                    DeleteDC(m_MemDc);
346.                    // Crea uno compatible con el dispotivo de salida
347.                    m_MemDc =  CreateCompatibleDC(GetDC(hwndEntrada));
348.                    CRect rect;
349.                    GetWindowRect(hwndEntrada,rect);
350.                    // Establece el tamaño de la ventana, respecto al tamaño de la imagen
351.                    if (m_tamv ==  DIMS_IMG1) // posiciona la ventana de acuerdo al tamaño de la imagen
352.                                    SetWindowPos(hwndEntrada,HWND_TOP,(rect.left-(m_tamv+50)>0) ?
353.                            rect.left-(m_tamv+50):10, rect.top,m_tamv,m_tamh,SWP_SHOWWINDOW);
354.                    else        // tam de la imagen es de 512x512
355.            SetWindowPos(hwndEntrada,HWND_TOP,1,1,m_tamv,m_tamh,SWP_SHOWWINDOW);
356.            m_bitmap = CreateCompatibleBitmap(GetDC(hwndEntrada),m_tamv,m_tamh);
357.            SelectObject(m_MemDc,m_bitmap);
358.            for (i= 0; i < NIVELES_GRIS; i++)
359.            //m_Tablacolor[i] = GetNearestColor(m_MemDc,RGB(i,i,i));
360.            m_Tablacolor[i] = GetNearestColor(m_MemDc,PALETTERGB(i,i,i));  // colores grises
361.            ShowWindow(hwndEntrada,SW_SHOW);
362.            Mostrar(hwndEntrada,m_tamh,m_tamh,CString(«Cargando la Imagen...,»));
363.            MoveToEx(m_MemDc,(0+m_tamh/2),0,NULL);
364.            LineTo(m_MemDc,(0+m_tamh/2),m_tamv);
365.            MoveToEx(m_MemDc,0,(0+m_tamh/2),NULL);
366.            LineTo(m_MemDc,m_tamh,(0+m_tamv/2));
367.            SetWindowText(hwndEntrada,»Iniciando Árbol Cuaternario...»);
368.            BitBlt(GetDC(hwndEntrada),0,0,m_tamh,m_tamv,m_MemDc,0,0,SRCCOPY);
369.    }
370.

371.    Particiona_Imagen(0, 0, m_tamh,m_tamv, tolerancia, hwndEntrada);

372.    Mensaje(«Terminado.»);

373.    // Rellena el utimo byte por si fuera necesario
374.    EscrAEncab(-1,(long)0);

375.    fclose(m_ArchSal);
376.    long int num_bytes = EscrAEncab(-2,(long)0);
377.    msg.Format(« Archivos :  ORIGINAL = %d bytes,\n»//
```

Archivo *Frac.CPP*

```
                « COMPACTADO = %d bytes,  Error rms = %.1lf\n»//
                « Razón de Compresión (%d / %d) => %.2lf : 1»,
                m_tamh*m_tamv, num_bytes, m_valrms, m_tamh*m_tamv, num_bytes,
                (double)(m_tamh*m_tamv)/(double)num_bytes); Mensaje(msg);

    // Libera la memoria utilizada
    free(m_bits_necesarios);
    free(m_imagen_dividida[0]);
    free(m_imagen_dividida[1]);
    free(m_imagen_dividida[2]);
    free(m_imagen_dividida[3]);
    free(m_dominio.num_doms_h);
    free(m_dominio.num_doms_v);
    free(m_dominio.tam_h_dom);
    free(m_dominio.tam_v_dom);
    free(m_dominio.intrv_h_dom);
    free(m_dominio.intrv_v_dom);
    for (i=0; i <= m_max_part-m_min_part; ++i)
        free(m_dominio.pixel[i]);
    free(m_dominio.pixel);
    free(m_imagen[0]);
    // Libera la memoria utilizada para los doiminios, en la estructura de lista ligada
    struct dominios_clasificados *nodo;
    struct dominios_clasificados *nodotemp;
    for (i=0; i <= m_max_part-m_min_part; ++i)
        for (k=0; k<3; ++k)
          for (j=0; j<24; ++j)
          {
              nodo = m_info_dominio[k][j][i];
              while(nodo->sig != NULL)
              {
                  nodotemp = nodo;
                  nodo = nodo->sig;
                  free(nodotemp);
              }
              free(nodo);
          }

          SetWindowText(hwndEntrada,»Terminado.»);
          // Finaliza el cursor de reloj de arena, (omitido, pero opcional)
    /*    if(hwndEntrada != NULL)
                  EndWaitCursor();    */
}

void CCompFrac::Descomprimir(HWND hwndSal,int escala, int iteraciones, BOOL procesa, double maxima_escala, int
    bits_escalam, int bits_desplz, BOOL part_sal)
{
    // Variables
    int i;
    int x_exponen, y_exponen;
    int tam_dom,num_doms;
    int tam_v_escalado,tam_h_escalado;
    CString msg;
    Restablece();// Reinicia todos los valores
    m_factorescala = escala;//Factor de escalamiento
    m_min_part = 3;
    m_max_part = 4;
    m_dom_taminterv = 4;
    m_part_sal = part_sal;
    m_bits_escalam = bits_escalam;
    m_bits_desplz = bits_desplz;
    m_max_escala = maxima_escala;
    // Comprueba si estan las cadenas de archivo vacias
    if((m_szNomArchEntr.IsEmpty()) || (m_szNomArchSal.IsEmpty()))
    {
            Mensaje(«Error: Falta nombre de archivo de Entrada o Salida»); return;
    }
    // Abre archivo de compresion (FIF ó TRN)
    if ((m_ArchEntr = fopen(m_szNomArchEntr, «rb»)) == NULL)
    {
            Mensaje(«Error: No puede abrirse el archivo de entrada»); return;
```

```
448.        }
449.        // Inicia el cursor de reloj de arena, (omitido, pero opcional)
450.  /*     if(hwndSal != NULL)
451.                    BeginWaitCursor();      */
452.        LeeDeEncab(-2); // inicializa la rutina

453.  // Lee los datos del encabezado del archivo. Esta rutina debería // contener una estructura que leyera la //información

454.        m_min_part = (int)LeeDeEncab(4);
455.        m_max_part = (int)LeeDeEncab(4);
456.        m_dom_taminterv = (int)LeeDeEncab(4);
457.        m_dom_intervmult = (int)LeeDeEncab(1);
458.        m_tipo_dom = (int)LeeDeEncab(2);
459.        m_tamh = (int)LeeDeEncab(12);
460.        m_tamv = (int)LeeDeEncab(12);

461.        // Cálculo del tamaño
462.        x_exponen = (int)floor(log((double)m_tamh)/log(2.0));
463.        y_exponen = (int)floor(log((double)m_tamv)/log(2.0));
464.        // El exponente es el min de los exponentes x_ y y_
465.        m_max_exponente = (x_exponen > y_exponen ? y_exponen : x_exponen);
466.        // m_tam es el tamaño del cuadrado mayor que quepa en la imagen
467.        // Este es usado para calcular los tamaños de rango y dominio.
468.        m_tam =  1<<m_max_exponente;
469.
470.        // Este es un valor contabilizado de cero escalamiento
471.        //*conversion forzada
472.        m_ialfa_cero = (int)(0.5 + (m_max_escala)/(2.0*m_max_escala)*(1<<m_bits_escalam));

473.        // Asigna memoria para la imagen de salida
474.        tam_h_escalado = (int)(m_factorescala*m_tamh);
475.        tam_v_escalado = (int)(m_factorescala*m_tamv);
476.        if(hwndSal != NULL)
477.        {
478.                    // Elimina dispositivo de contexto anterior
479.                    DeleteDC(m_MemDc);
480.                    // Crea uno compatible con el dispositivo de salida
481.                    m_MemDc =  CreateCompatibleDC(GetDC(hwndSal));
482.                    CRect rect;
483.                    GetWindowRect(hwndSal,rect);
484.
485.                    // Establece el tamaño de la Ventana respecto del tamaño de la imagen.
486.
487.                    //SetWindowPos(hwndSal,HWND_TOP,rect.left,rect.top,tam_v_escalado,tam_h_escalado,
488.                    //              SWP_SHOWWINDOW);
489.                    if (tam_v_escalado ==  DIMS_IMG1)  //posiciona la ventana de acuerdo al tamaño de la imagen
490.                    SetWindowPos(hwndSal,HWND_TOP,(rect.left-(tam_v_escalado+50)>0) ?
491.                                        rect.left-(tam_v_escalado+50):10,
492.                                        rect.top,tam_v_escalado,tam_h_escalado,SWP_SHOWWINDOW);
493.              else          // tam de la imagen es de 257x257 en adelante (incluyendo 512x512)
494.                    SetWindowPos(hwndSal,HWND_TOP,1,1,tam_v_escalado,tam_h_escalado,
495.                                                                  SWP_SHOWWINDOW);
496.              m_bitmap = CreateCompatibleBitmap(GetDC(hwndSal),tam_v_escalado,tam_h_escalado);
497.              SelectObject(m_MemDc,m_bitmap);
498.              // Crea una tabla de color valida para el dispositivo
499.              for (i= 0; i < NIVELES_GRIS; i++)
500.                    //m_Tablacolor[i] = GetNearestColor(m_MemDc,RGB(i,i,i));
501.                    m_Tablacolor[i] = GetNearestColor(m_MemDc,PALETTERGB(i,i,i));
502.        }
503.        // Asigna memoria para las imágenes original y temporal
504.        unsigned char *reng_imagen;
505.        m_imagen = (unsigned char **)malloc(tam_v_escalado*sizeof(unsigned char *));
506.        reng_imagen = (unsigned char*)malloc((long)tam_h_escalado*(long)tam_v_escalado*sizeof(unsigned
507.                                                      char));
508.        if (reng_imagen == NULL)
509.        {
510.            Mensaje(«Error: Memoria para la Imagen, agotada»);
511.            return;
512.        }
513.        for (i = 0; i<tam_v_escalado; ++i, reng_imagen += tam_h_escalado)
```

```
514.            m_imagen[i] = reng_imagen;

515.        unsigned char *row_imageTemp;
516.        m_imagenTemp = (unsigned char **)malloc(tam_v_escalado*sizeof(unsigned char *));
517.        row_imageTemp = (unsigned char *)malloc((long)tam_h_escalado*(long)tam_v_escalado*sizeof(unsigned
518.                                                char));
519.        if (row_imageTemp == NULL)
520.        {
521.            Mensaje(«Error: Memoria Secundaria para la Imagen, agotada»);
522.            return;
523.        }
524.        for (i = 0; i<tam_v_escalado; ++i, row_imageTemp += tam_h_escalado)
525.            m_imagenTemp[i] = row_imageTemp;

526.        // Dado que las particiones max_ and min_ son variables, deben ser asignadas
527.        i = m_max_part - m_min_part + 1;
528.        m_bits_necesarios = (int *)malloc(sizeof(int)*i);
529.        m_numDomh = (int *)malloc(sizeof(int)*i);
530.        m_intrv_h_dom = (int *)malloc(sizeof(int)*i);
531.        m_intrv_v_dom = (int *)malloc(sizeof(int)*i);
532.
533.        // calcula los bits necesarios para leer cada tipo de dominio
534.        for (i=0; i <= m_max_part-m_min_part; ++i)
535.        {
536.            // primero calcula cuantos dominios hay horizontalmente
537.            tam_dom = m_tam >> (m_min_part+i-1);
538.            if (m_tipo_dom == 2)
539.                m_intrv_h_dom[i] = m_dom_taminterv;
540.            else if (m_tipo_dom == 1)
541.                                if (m_dom_intervmult ==1)
542.                                        m_intrv_h_dom[i] = (m_tam >> (m_max_part - i-1))*m_dom_taminterv;
543.                        else
544.                                        m_intrv_h_dom[i] = (m_tam >> (m_max_part - i-1))/m_dom_taminterv;
545.                else
546.                        if (m_dom_intervmult ==1)
547.                                        m_intrv_h_dom[i] = tam_dom*m_dom_taminterv;
548.                        else
549.                                        m_intrv_h_dom[i] = tam_dom/m_dom_taminterv;
550.                m_numDomh[i] = 1+(m_tamh-tam_dom)/m_intrv_h_dom[i];
551.
552.                // Ahora calcula cuantos dominios hay verticalmente
553.              if (m_tipo_dom == 2)
554.                m_intrv_v_dom[i] = m_dom_taminterv;
555.              else if (m_tipo_dom == 1)
556.                                if (m_dom_intervmult ==1)
557.                                        m_intrv_v_dom[i] = (m_tam >> (m_max_part - i-1))*m_dom_taminterv;
558.                                else
559.                                        m_intrv_v_dom[i] = (m_tam >> (m_max_part - i-1))/m_dom_taminterv;
560.                                 else
561.                                if (m_dom_intervmult ==1)
562.                                        m_intrv_v_dom[i] = tam_dom*m_dom_taminterv;
563.                                else
564.                                        m_intrv_v_dom[i] = tam_dom/m_dom_taminterv;

565.            num_doms = 1+(m_tamv-tam_dom)/m_intrv_v_dom[i];
566.            //*conversion forzada
567.            m_bits_necesarios[i] = (int)(ceil(log((double)num_doms*(double)m_numDomh[i])/log(2.0)));
568.    }
569.    // Abre archivo de salida (archivo descompactado)
570.    if ((m_ArchSal = fopen(m_szNomArchSal, «wb»)) == NULL)
571.    {
572.            Mensaje(«Error: No puede abrirse el archivo de Salida»);
573.            return;
574.    }

575.    // Lee los datos de las transformaciones
576.    m_ptransf = &m_transformaciones;

577.    Mensaje(«Leyendo Transformaciones...»);  //mensaje a la ventana de Estado del Diálogo Principal
```

```
578.  SetWindowText(hwndSal,»Leyendo Transformaciones...»);// Mensaje a la ventana de vision de la imagen

579.  Particiona_Imagen(0, 0, m_tamh,m_tamv );
580.  fclose(m_ArchEntr);

581.  Mensaje(«Terminado»); //mensaje a la ventana de Estado del Diálogo Principal

582.  SetWindowText(hwndSal,»Terminado.»);// Mensaje a la ventana de vision de la imagen

583.  // cuando se produce la particion, solo se leen las transformaciones
584.  // y estas con escritas hacia el archivo de salida
585.  if (m_part_sal)
586.  {
587.          // Abre arhivo de Salida (archivo descompactado)
588.          if ((m_ArchSal = fopen(m_szNomArchSal, «wb»)) == NULL)
589.          {
590.              Mensaje(«Error: No puede abrirse el archivo de Salida»);
591.              return;
592.          }
593.          fprintf(m_ArchSal,»\n%d %d\n %d %d\n%d %d\n\n»,
594.                                  0, 0, tam_h_escalado, 0, tam_h_escalado, tam_v_escalado);

595.          msg.Format(«Datos particionados a la Salida en %s.»,m_szNomArchSal);
596.          Mensaje(msg);

597.          fclose(m_ArchSal);
598.          return;
599.  }

600.  int iNumTransf = 0;
601.  if(hwndSal != NULL) ShowWindow(hwndSal,SW_SHOW);
602.          // Aplica las transformaciones
603.          for (i=0; i<iteraciones; ++i) if(hwndSal != NULL)
604.          {
605.              msg.Format(«Procesando ... [%d%%]»,(i*100)/iteraciones);
606.              iNumTransf = Aplica_Transformaciones(hwndSal,msg);
607.          }
608.          else
609.              iNumTransf = Aplica_Transformaciones(NULL,»»);
610.              // Ajusta los bordes de la imagen suavizandolos
611.
612.  if (procesa)
613.                if(hwndSal != NULL)
614.                  Suaviza_Imagen(hwndSal);
615.                else
616.                  Suaviza_Imagen(NULL);

617.        // Escribe la modificacion al archivo de salida
618.        if(!EscribArchRaw(tam_h_escalado,tam_v_escalado,m_szNomArchSal))
619.                return;

620.        SetWindowText(hwndSal,»Terminado.»);// Mensaje a la ventana de vision de la imagen

621.        Sleep((DWORD)1000);

622.        msg.Format(«%d Transf. Aplicadas.», iNumTransf);
623.        SetWindowText(hwndSal,msg);

624.        free(m_imagen);
625.        free(m_imagenTemp);
626.        // Finaliiza el cursor de reloj de arena, (omitido, pero opcional)
627.  /*          if(hwndSal != NULL)
628.                                  EndWaitCursor();  */
629.  }

630.  ////////////////////////////////////////////////////////////////////
631.  // Rutinas de Decodificación Especificas
632.  ////////////////////////////////////////////////////////////////////
```

Archivo *Frac.CPP*

```
633.  void CCompFrac::Suaviza_Imagen(HWND hdcSal)
634.  {
635.        unsigned char pixel1, pixel2;
636.        int i,j;
637.        int w1,w2;


638.        Mensaje(«PostProcesamiento (Suavizado).»);

639.        m_ptransf = &m_transformaciones;
640.        while (m_ptransf->sig != NULL)
641.        {     m_ptransf = m_ptransf->sig;
642.              if (m_ptransf->rngx == 0 || m_ptransf->rngy == 0)
643.                continue;

644.              if (m_ptransf->prof == m_max_part)
645.              {
646.                w1 = 5;
647.                w2 = 1;
648.              } else {
649.                w1 = 2;
650.                w2 = 1;
651.        }
652.        for (i=m_ptransf->rngx; i<m_ptransf->rrx; ++i)
653.        {     pixel1 = m_imagen[(int)m_ptransf->rngy][i];
654.              pixel2 = m_imagen[(int)m_ptransf->rngy-1][i];
655.              m_imagen[(int)m_ptransf->rngy][i] = (w1*pixel1 + w2*pixel2)/(w1+w2);
656.              m_imagen[(int)m_ptransf->rngy-1][i] = (w2*pixel1 + w1*pixel2)/(w1+w2);
657.        }
658.        for (j=m_ptransf->rngy; j<m_ptransf->rry; ++j)
659.        {
660.              pixel1 = m_imagen[j][(int)m_ptransf->rngx];
661.              pixel2 = m_imagen[j][(int)m_ptransf->rngx-1];
662.              m_imagen[j][(int)m_ptransf->rngx] = (w1*pixel1 + w2*pixel2)/(w1+w2);
663.              m_imagen[j][(int)m_ptransf->rngx-1] = (w2*pixel1 + w1*pixel2)/(w1+w2);
664.        }
665.  }
666.  if(hdcSal != NULL)
667.        Mostrar(hdcSal,m_tamh*m_factorescala,m_tamv*m_factorescala,CString(«PostProcesamiento"//
668.              "(Suavizado)...de la Imagen «));
669. }
670.
671. void CCompFrac::Particiona_Imagen(int atx, int aty,  int tamh, int tamv )
672. {
673.   int x_exponen,    // potencia mayor de 2 del tamaño de la imagen que ajuste
674.       y_exponen,    // horizontàl y verticalmentely el rectangulo.
675.       expon,        // El tamaño actual de la imagen que es codificada
676.       tam,
677.       prof;

678.   x_exponen = (int)floor(log((double)tamh)/log(2.0));
679.   y_exponen = (int)floor(log((double)tamv)/log(2.0));

680.   // el esponente toma el valor minimo de entre los exponenetes x y y
681.   expon = (x_exponen > y_exponen ? y_exponen : x_exponen);
682.   tam = 1<<expon;
683.   prof = m_max_exponente - expon;

684.   Lee_Transformaciones(atx,aty,tam,tam,prof);

685.   if (tam != tamh)
686.         Particiona_Imagen(atx+tam, aty, tamh-tam, tamv );

687.   if (tam != tamv)
688.         Particiona_Imagen(atx, aty+tam, tam,tamv-tam );
689. }
```

```
690. int CCompFrac::Aplica_Transformaciones(HWND hdcSal,CString sTitulo)
691. {
692.     unsigned char **imagen_temp;
693.     int i,j,i1,j1,cont=0;
694.     double pixel;
695.     CString msg;
696.     m_ptransf = &m_transformaciones;
697.     while (m_ptransf->sig != NULL)
698.     {
699.         m_ptransf = m_ptransf->sig;
700.         ++cont;
701.         // Dado que ciclo interno es el mismo en cada «case» de la estructura switch de abajo
702.             // solo se define una vez para facilitar posibles modificaciones
703.         switch(m_ptransf->oper_sim)
704.         {
705.             case 0:  for (i=m_ptransf->rngx, i1 = m_ptransf->domx;i<m_ptransf->rrx; ++i, i1 += 2)
706.                             for (j=m_ptransf->rngy, j1 = m_ptransf->domy;j<m_ptransf->rry; ++j, j1 += 2)
707.                         {
708.                             pixel =
709.             (m_imagen[j1][i1]+m_imagen[j1][i1+1]+m_imagen[j1+1][i1]+m_imagen[j1+1][i1+1])/4.0;
710.                                                                     //*conversion forzada
711.                             m_imagenTemp[j][i] = (unsigned char)
712.                                     (limita(0.5 + m_ptransf->escala*pixel+m_ptransf->desplz));
713.                         }
714.                         break;
715.             case 1:  for (j=m_ptransf->rngy, i1 = m_ptransf->domx;j<m_ptransf->rry; ++j, i1 += 2)
716.                         for (i=m_ptransf->rrx-1,j1 = m_ptransf->domy; i>=(int)m_ptransf->rngx; —i, j1 += 2)
717.                         {
718.                             pixel =
719.             (m_imagen[j1][i1]+m_imagen[j1][i1+1]+m_imagen[j1+1][i1]+m_imagen[j1+1][i1+1])/4.0;
720.                         //*conversion forzada
721.                             m_imagenTemp[j][i] = (unsigned char)
722.                                     (limita(0.5 + m_ptransf->escala*pixel+m_ptransf->desplz));
723.                         }
724.                         break;
725.             case 2:  for (i=m_ptransf->rrx-1,i1 = m_ptransf->domx; i>=(int)m_ptransf->rngx; —i, i1 += 2)
726.                         for (j=m_ptransf->rry-1,j1 = m_ptransf->domy; j>=(int)m_ptransf->rngy; —j, j1 += 2)
727.                         {
728.                             pixel =
729.         (m_imagen[j1][i1]+m_imagen[j1][i1+1]+m_imagen[j1+1][i1]+m_imagen[j1+1][i1+1])/4.0;
730.                             //*conversion forzada
731.                             m_imagenTemp[j][i] = (unsigned char)
732.                                     (limita(0.5 + m_ptransf->escala*pixel+m_ptransf->desplz));
733.                         }
734.                         break;
735.             case 3:  for (j=m_ptransf->rry-1,i1 = m_ptransf->domx; j>=(int)m_ptransf->rngy; —j, i1 += 2)
736.                         for (i=m_ptransf->rngx, j1 = m_ptransf->domy;i<m_ptransf->rrx; ++i, j1 += 2)
737.                         {
738.                             pixel =
739.             (m_imagen[j1][i1]+m_imagen[j1][i1+1]+m_imagen[j1+1][i1]+m_imagen[j1+1][i1+1])/4.0;
740.                             //*conversion forzada
741.                             m_imagenTemp[j][i] = (unsigned char)
742.                                     (limita(0.5 + m_ptransf->escala*pixel+m_ptransf->desplz));
743.                         }
744.                         break;
745.             case 4:  for (j=m_ptransf->rngy, i1 = m_ptransf->domx;j<m_ptransf->rry; ++j, i1 += 2)
746.                         for (i=m_ptransf->rngx, j1 = m_ptransf->domy;i<m_ptransf->rrx; ++i, j1 += 2)
747.                         {
748.                             pixel =
749.             (m_imagen[j1][i1]+m_imagen[j1][i1+1]+m_imagen[j1+1][i1]+m_imagen[j1+1][i1+1])/4.0;
750.                                                                     //*conversion forzada
751.                             m_imagenTemp[j][i] = (unsigned char)
752.                                     (limita(0.5 + m_ptransf->escala*pixel+m_ptransf->desplz));
753.                         }
754.                         break;
755.             case 5:  for (i=m_ptransf->rngx, i1 = m_ptransf->domx;i<m_ptransf->rrx; ++i, i1 += 2)
756.                         for (j=m_ptransf->rry-1,j1 = m_ptransf->domy; j>=(int)m_ptransf->rngy; —j, j1 += 2)
757.                         {
758.                             pixel =
759.             (m_imagen[j1][i1]+m_imagen[j1][i1+1]+m_imagen[j1+1][i1]+m_imagen[j1+1][i1+1])/4.0;
760.                                                                     //*conversion forzada
```

Archivo *Frac.CPP*

```
761.                            m_imagenTemp[j][i] = (unsigned char)
762.                                    (limita(0.5 + m_ptransf->escala*pixel+m_ptransf->desplz));
763.                    }
764.                    break;
765.            case 6:  for (j=m_ptransf->rry-1,i1 = m_ptransf->domx; j>=(int)m_ptransf->rngy; —j, i1 += 2)
766.                    for (i=m_ptransf->rrx-1,j1 = m_ptransf->domy; i>=(int)m_ptransf->rngx; —i, j1 += 2)
767.                    {
768.                            pixel =
769.            (m_imagen[j1][i1]+m_imagen[j1][i1+1]+m_imagen[j1+1][i1]+m_imagen[j1+1][i1+1])/4.0;
                                                                    //*conversion forzada
770.
771.                            m_imagenTemp[j][i] = (unsigned char)
772.                                    (limita(0.5 + m_ptransf->escala*pixel+m_ptransf->desplz));
773.                    }
774.                    break;
775.            case 7:  for (i=m_ptransf->rrx-1,i1 = m_ptransf->domx; i>=(int)m_ptransf->rngx; —i, i1 += 2)
776.                    for (j=m_ptransf->rngy, j1 = m_ptransf->domy; j<m_ptransf->rry; ++j, j1 += 2)
777.                    {
778.                            pixel =
779.            (m_imagen[j1][i1]+m_imagen[j1][i1+1]+m_imagen[j1+1][i1]+m_imagen[j1+1][i1+1])/4.0;
                                                                    //*conversion forzada
780.
781.                            m_imagenTemp[j][i] = (unsigned char)
782.                                    (limita(0.5 + m_ptransf->escala*pixel+m_ptransf->desplz));
783.                    }
784.                    break;
785.        }

786.     }
787.     imagen_temp = m_imagen;
788.     m_imagen = m_imagenTemp;
789.     m_imagenTemp = imagen_temp;

790.     msg.Format(«%d Transformaciones aplicadas.»,cont);
791.     Mensaje(msg);        // Mensaje al cuadro de Estado del Diálogo Principal

792.     if(hdcSal != NULL)
793.         Mostrar(hdcSal,m_tamh*m_factorescala,m_tamv*m_factorescala,sTitulo);

794.     return cont;
795. }

796. void CCompFrac::Lee_Transformaciones(int atx, int aty, int tamx, int tamy, int prof)
797. {
798.         // Vars. locales
799.         int ialfa,              // Factor de escalamiento en tipo entero
800.         ibeta;          // Desplazamiento en tipo entero
801.
802.         long domain_ref;
803.         double alfa, beta;
804.
805.         // se mantiene analizando hasta llegar a las partes lo suficientemente pequeñas
806.         if (prof < m_min_part)
807.         {
808.             Lee_Transformaciones(atx,aty, tamx/2, tamy/2, prof+1);
809.             Lee_Transformaciones(atx+tamx/2,aty, tamx/2, tamy/2, prof+1);
810.             Lee_Transformaciones(atx,aty+tamy/2,tamx/2, tamy/2,  prof+1);
811.             Lee_Transformaciones(atx+tamx/2,aty+tamy/2,tamx/2,tamy/2,prof+1);
812.             return;
813.         }

814.         if (prof < m_max_part && LeeDeEncab(1))
815.         {       // El 1 significa subdivision.. mediante quadtree
816.             Lee_Transformaciones(atx,aty, tamx/2, tamy/2, prof+1);
817.             Lee_Transformaciones(atx+tamx/2,aty, tamx/2, tamy/2, prof+1);
818.             Lee_Transformaciones(atx,aty+tamy/2, tamx/2, tamy/2, prof+1);
819.             Lee_Transformaciones(atx+tamx/2,aty+tamy/2,tamx/2,tamy/2,prof+1);
820.         } else {
821.             // tiene que leer una transformación
822.             m_ptransf->sig = (struct nodo_transformacion *)
823.                 malloc(sizeof(struct nodo_transformacion )); m_ptransf = m_ptransf->sig;
```

```
824.                m_ptransf->sig = NULL;
825.                ialfa = (int)LeeDeEncab(m_bits_escalam);
826.                ibeta = (int)LeeDeEncab(m_bits_desplz);
827.                alfa = (double)ialfa/(double)(1<<m_bits_escalam)*(2.0*m_max_escala)-m_max_escala;
828.                beta = (double)ibeta/(double)((1<<m_bits_desplz)-1)* ((1.0+fabs(alfa))*NIVELES_GRIS);
829.                if (alfa > 0.0)
830.                  beta -= alfa*NIVELES_GRIS;
831.                m_ptransf->escala = alfa;
832.                m_ptransf->desplz = beta;
833.                if (ialfa != m_ialfa_cero)
834.                {
835.                  m_ptransf-> oper_sim = (int)LeeDeEncab(3);
836.                  domain_ref = LeeDeEncab(m_bits_necesarios[prof-m_min_part]);
837.                              //*conversion forzada
838.    m_ptransf->domx = (int)(double)(domain_ref % m_numDomh[prof-m_min_part]) * m_intrv_h_dom[prof-m_min_part];
839.                              //*conversion forzada
840.    m_ptransf->domy = (int)(double)(domain_ref / m_numDomh[prof-m_min_part]) * m_intrv_v_dom[prof-m_min_part];
841.          } else {
842.                m_ptransf-> oper_sim = 0;
843.                 m_ptransf-> domx  = 0; m_ptransf-> domy = 0;
844.          }
845.          m_ptransf->rngx = atx;
846.          m_ptransf->rngy = aty;
847.          m_ptransf->prof = prof;
848.          m_ptransf->rrx = atx + tamx;
849.          m_ptransf->rry = aty + tamy;
850.          //Aplica el factor de escalamiento
851.          m_ptransf->rrx *= m_factorescala;
852.          m_ptransf->rry *= m_factorescala;
853.          m_ptransf->rngx *= m_factorescala;
854.          m_ptransf->rngy *= m_factorescala;
855.          m_ptransf->domx *= m_factorescala;
856.          m_ptransf->domy *= m_factorescala;
857.          if (m_part_sal)
858.                fprintf(m_ArchSal,»\n%d %d\n %d %d\n%d %d\n\n»,
859.                    atx,      m_tamv-aty-tamy, atx,      m_tamv-aty, atx+tamx, m_tamv-aty);
860.    }
861.  }
862.
863.
864.  long CCompFrac::LeeDeEncab(int tam)
865.  {
866.    int i;
867.    int valor = 0;
868.    //static int ptr = 1; // cuantos bits son guardados en total hasta aqui //static int sum;
869.    // tam == -2 significa inicialización
870.    if (tam == -2)
871.    {
872.          m_sum = fgetc(m_ArchEntr);
873.          m_sum <<= 1;
874.          return((long)0);
875.    }

876.  // tam == -1 significa procesar el siguiente bit, pero sin avanzar // el puntero de archivo

877.  if (tam == -1)
878.          return((long)((m_sum&256)>>8));

879.  for (i=0; i<tam; ++i, ++m_ptr,  m_sum <<= 1) {
880.          if (m_sum & 256)
881.                valor |= 1<<i;
882.          if (m_ptr == 8)
883.          {
884.                m_sum = getc(m_ArchEntr);
885.                m_ptr=0;
886.          }
887.    }
888.    return((long)valor);
889.  }
890.
```

Archivo *Frac.CPP*

```
891.  ////////////////////////////////////////////////////////////////////
892.  // Rutinas de Codificación Especificas
893.  ////////////////////////////////////////////////////////////////////
894.  void CCompFrac::Clasifica(int x, int y, int tamx, int tamy, int *primerclas, int *segundaclas, int *simoper, double *sumpix,
895.  double *sumpix2, int tipo)
896.  {
897.    int orden[4], i,j;
898.    double a[4],a2[4];

899.          // Obtiene los valores promedio de cada cuadrante
900.    if (tipo == 2)
901.    {
902.          Promedio(x,y,tamx/2,tamy/2,&a[0], &a2[0]);
903.          Promedio(x,y+tamy/2,tamx/2,tamy/2,&a[1], &a2[1]);
904.          Promedio(x+tamx/2,y+tamy/2, tamx/2,tamy/2, &a[2],&a2[2]);
905.          Promedio(x+tamx/2,y,tamx/2,tamy/2,&a[3],&a2[3]);
906.    }
907.    else
908.    {
909.          Promedio1(x,y,tamx/2,tamy/2,&a[0], &a2[0]);
910.          Promedio1(x,y+tamy/2,tamx/2,tamy/2,&a[1], &a2[1]);
911.          Promedio1(x+tamx/2,y+tamy/2, tamx/2,tamy/2, &a[2],&a2[2]);
912.          Promedio1(x+tamx/2,y,tamx/2,tamy/2,&a[3],&a2[3]);
913.    }
914.    *sumpix = a[0] + a[1] + a[2] + a[3];
915.    *sumpix2 =  a2[0] + a2[1] + a2[2] + a2[3];
916.
917.    for (i=0; i<4; ++i)
918.    {
919.    // después el orden bajo orden[i] es el i-ésimo cuadrante de mayor brillo
920.        orden[i] = i;
921.        // Convierte a2[] para guardar la varianza de cada cuadrante
922.        a2[i] -= (double)(1<<(2*tipo))*a[i]*a[i]/(double)(tamx*tamy);
923.    }

924.    // Ahora se ordena el valor promedio, así como en la var. orden[], cuyo
925.    // orden dara los indices (en a[]) de mayor brillo a mayor obscuro
926.    for (i=2; i>=0; --i)
927.        for (j=0; j<=i; ++j)
928.          if (a[j]<a[j+1])
929.          {
930.                intercambia(orden[j], orden[j+1],int)
931.                intercambia(a[j], a[j+1],double)
932.          }

933.    // Debido a la forma en que se ha ordenado a[], la rotación puede ser
934.    // leida a partir de la derecha de orden[]. Ello hará que el cuadrante de mayor brillo
935.    // esté en la esquina superior izquierda. Sin embargo, debe indicarse
936.    // que clase canónica ha de comienzar la porción de la imagen,
937.    // ya sea al realizar un reflejo (volteo) o solo una rotación. La siguiente tabla
938.    // reune las siguientes lineas
939.    // orden        clase        realiza rotación
940.    // 0,2,1,3        0            0
941.    // 0,2,3,1        0            1
942.    // 0,1,2,3        1            0
943.    // 0,3,2,1        1            1
944.    // 0,1,3,2        2            0
945.    // 0,3,1,2        2            1
946.    *simoper = orden[0];
947.    // rota los valores
948.    for (i=0; i<4; ++i)
949.        orden[i] = (orden[i] - (*simoper) + 4)%4;

950.    for (i=0; orden[i] != 2; ++i);
951.    *primerclas = i-1;
952.    if (orden[3] == 1 || (*primerclas == 2 && orden[2] == 1)) *simoper += 4;

953.    // Ahora se procede a clasficar la sub-imagen por variancion
954.    // de cuadrantes. Esto proporciona 24 subclases por cada una de las 3 clases
```

```
955.    for (i=0; i<4; ++i) orden[i] = i;

956.    for (i=2; i>=0; —i)
957.        for (j=0; j<=i; ++j)
958.          if (a2[j]<a2[j+1])
959.          {
960.                  intercambia(orden[j], orden[j+1],int)
961.                  intercambia(a2[j], a2[j+1],double)
962.          }

963.    // Ahora se realiza la operación de simetria
964.    for (i=0; i<4; ++i)
965.        orden[i] = (orden[i] - (*simoper%4) + 4)%4;
966.    if (*simoper > 3)
967.        for (i=0; i<4; ++i)
968.          if (orden[i]%2) orden[i] = (2 + orden[i])%4;

969.    // Se busca regresar un número de clase entre 0 y 23
970.    // que dependa del ordenamiento de los cuadrantes de acuerdo a su varianza
971.    *segundaclas = 0;
972.    for (i=2; i>=0; —i)
973.        for (j=0; j<=i; ++j)
974.          if (orden[j] > orden[j+1])
975.          {       intercambia(orden[j],orden[j+1], int);
976.                  if (orden[j] == 0 || orden [j+1] == 0)
977.                    *segundaclas += 6;
978.                  else if (orden[j] == 1 || orden [j+1] == 1)
979.                    *segundaclas += 2;
980.                  else if (orden[j] == 2 || orden [j+1] == 2)
981.                    *segundaclas += 1;
982.          }
983.  }

984.  BOOL CCompFrac::Calcula_Sums(int tamh, int tamv)
985.  {
986.        int i,j,k,l,
987.          dominio_x,
988.          dominio_y,
989.          prim_clase,
990.          segnd_clase,
991.          tam,
992.          x_exponen,
993.          y_exponen;

994.        struct dominios_clasificados *nodo; Mensaje(«Calculando la suma de Dominios... «);

995.        fflush(stdout);

996.        // Divide previamente la imagen dentro de la var. «m_imagen_dividida»
997.        // para evitar tener que realizar una repeticion de promedios de los grupos de // pixeles de 2x2. Hay 4 formas
998.        //de dividir la imagen, que dependen de
999.        // la ubicacion del dominio, ya sea par o impar
1000.       for (i=0; i<2; ++i)
1001.         for (j=0; j<2; ++j)
1002.               for (k=i; k<tamh-i; k += 2) for (l=j; l<tamv-j; l += 2)
1003.                     m_imagen_dividida[(i<<1)+j][l>>1][k>>1] =
1004.                       ((double)m_imagen[l][k] + (double)m_imagen[l+1][k+1] +
1005.                       (double)m_imagen[l][k+1] + (double)m_imagen[l+1][k])*0.25;
1006.       // Asigna memoria para «sum» and «sum^2» del dominio de pixeles
1007.       // Primero calcula «tam» (tamaño) del cuadrado mayor que quepa
1008.       // en la imagen.
1009.       x_exponen = (int)floor(log((double)tamh)/log(2.0));
1010.       y_exponen = (int)floor(log((double)tamv)/log(2.0));

1011.       // m_max_exponente toma el minimo de x_exponen y y_exponen
```

```
1012.   m_max_exponente = (x_exponen > y_exponen ? y_exponen : x_exponen);

1013.   // «tam» es el tamaño de cuadrado mayor que cabe en la imagen
1014.   // Este es usado para calcular los tamaños de dominios y rangos
1015.   tam =  1<<m_max_exponente;

1016.   if (m_max_exponente < m_max_part)
1017.   {
1018.     Mensaje(«Error: La Particion esta en un rango no válido»); return FALSE;
1019.   }
1020.   if (m_max_exponente-2 < m_max_part)
1021.     Mensaje(«Precuación: Demasiadas particiones \»Quadtree\» generan rangos no válidos.»);
1022.
1023.   i = m_max_part - m_min_part + 1;
1024.   m_dominio.num_doms_h = (int *)malloc(sizeof(int)*i);
1025.   m_dominio.num_doms_v = (int *)malloc(sizeof(int)*i);
1026.   m_dominio.tam_h_dom = (int *)malloc(sizeof(int)*i);
1027.   m_dominio.tam_v_dom = (int *)malloc(sizeof(int)*i);
1028.   m_dominio.intrv_h_dom = (int *)malloc(sizeof(int)*i);
1029.   m_dominio.intrv_v_dom = (int *)malloc(sizeof(int)*i);
1030.   m_dominio.pixel= (struct pixeles_dominio ***) malloc(i*sizeof(struct pixeles_dominio **));
1031.  if (m_dominio.pixel == NULL)
1032.  {
1033.     Mensaje(«Error: Input/Output name not given»);
1034.     return FALSE;
1035.  }

1036.  for (i=0; i <= m_max_part-m_min_part; ++i)
1037.  {    // Primero calcula el número de dominios que hay horizontalmente
1038.     m_dominio.tam_h_dom[i] = tam >> (m_min_part+i-1);
1039.     if (m_tipo_dom == 2)
1040.       m_dominio.intrv_h_dom[i] = m_dom_taminterv;
1041.     else if (m_tipo_dom == 1)
1042.          if (m_dom_intervmult == 1)
1043.               m_dominio.intrv_h_dom[i] = (tam >> (m_max_part - i-1))*m_dom_taminterv;
1044.          else
1045.               m_dominio.intrv_h_dom[i] = (tam >> (m_max_part - i-1))/m_dom_taminterv;
1046.       else
1047.          if (m_dom_intervmult == 1)
1048.               m_dominio.intrv_h_dom[i] = m_dominio.tam_h_dom[i]*m_dom_taminterv;
1049.          else
1050.               m_dominio.intrv_h_dom[i] = m_dominio.tam_h_dom[i]/m_dom_taminterv;
1051.                    m_dominio.num_doms_h[i] = 1+
1052.                    (tamh-m_dominio.tam_h_dom[i])/m_dominio.intrv_h_dom[i];

1053.     m_dominio.tam_v_dom[i] = tam >> (m_min_part+i-1);
1054.     if (m_tipo_dom == 2)
1055.       m_dominio.intrv_v_dom[i] = m_dom_taminterv;
1056.     else if (m_tipo_dom == 1)
1057.         if (m_dom_intervmult == 1)
1058.           m_dominio.intrv_v_dom[i] = (tam >> (m_max_part - i-1))*m_dom_taminterv;
1059.         else
1060.           m_dominio.intrv_v_dom[i] = (tam >> (m_max_part - i-1))/m_dom_taminterv;
1061.       else
1062.         if (m_dom_intervmult == 1)
1063.                    m_dominio.intrv_v_dom[i] = m_dominio.tam_v_dom[i]*m_dom_taminterv;
1064.         else
1065.                    m_dominio.intrv_v_dom[i] = m_dominio.tam_v_dom[i]/m_dom_taminterv;

1066.     m_dominio.num_doms_v[i] = 1+(tamv-m_dominio.tam_v_dom[i])/m_dominio.intrv_v_dom[i];

1067.     // Ahora se calcula el número de bits necesarios para almacenar la info. del dominio
1068.     //*conversion forzada
1069.     m_bits_necesarios[i] = (int)(ceil(log((double)m_dominio.num_doms_h[i]*
1070.         (double)m_dominio.num_doms_v[i])/log(2.0)) );
1071.     struct pixeles_dominio *row_domain_pixel;
1072.     m_dominio.pixel[i]=  (struct pixeles_dominio **)malloc((m_dominio.num_doms_v[i])*sizeof(struct
```

```
1073.                 pixeles_dominio *));
1074.               row_domain_pixel = (struct (m_dominio.num_doms_h[i])*(long)(m_dominio.num_doms_v[i])*sizeof(struct
1075.                 pixeles_dominio));
1076.             if(row_domain_pixel == NULL)
1077.             {
1078.                   Mensaje(«Error: Out of memory for image buffer»);
1079.                   return FALSE;
1080.             }
1081.             for (int _i = 0; _i<m_dominio.num_doms_v[i]; ++_i, row_domain_pixel += m_dominio.num_doms_h[i])
1082.                   m_dominio.pixel[i][_i] = row_domain_pixel;
1083.         }

1084.         // Asigna e inicializa en cero la lista que contendra la informacion de los dominios clasificados
1085.         i = m_max_part - m_min_part + 1;
1086.         for (prim_clase = 0; prim_clase < 3; ++prim_clase)
1087.             for (segnd_clase = 0; segnd_clase < 24; ++segnd_clase)
1088.             {
1089.               m_info_dominio[prim_clase][segnd_clase] =
1090.                  (struct dominios_clasificados **) malloc(i*sizeof(struct dominios_clasificados *));
1091.               for (j=0; j<i; ++j)
1092.                     m_info_dominio[prim_clase][segnd_clase][j] = NULL;
1093.             }

1094.     // Calcula  previamente las sumas de cuadrados para los dominios
1095.     // Esta parte puede agilizarse al superponer los dominios,
1096.     // si las acumulacion en «sum» se evita
1097.     for (i=0; i <= m_max_part-m_min_part; ++i) {
1098.       for (j=0,dominio_x=0; j<m_dominio.num_doms_h[i]; ++j, dominio_x+=m_dominio.intrv_h_dom[i])
1099.         for (k=0,dominio_y=0; k<m_dominio.num_doms_v[i]; ++k, dominio_y+=m_dominio.intrv_v_dom[i]) {
1100.                   Clasifica(dominio_x, dominio_y,
1101.                         m_dominio.tam_h_dom[i],
1102.                         m_dominio.tam_v_dom[i],
1103.                        &prim_clase, &segnd_clase, &m_dominio.pixel[i][k][j].sym,
1104.                        &m_dominio.pixel[i][k][j].sum, &m_dominio.pixel[i][k][j].sum2, 2);
1105.           // Cuando la info. del dominio es referenciada desde la lista,
1106.       // se necesita saber donde esta el dominio... para poder almacenar la posicion m_dominio.pixel[i][k][j].dom_x = j;
1107.     m_dominio.pixel[i][k][j].dom_y = k; nodo = (struct dominios_clasificados *) malloc(sizeof(struct dominios_clasificados));
1108.         // coloca este dominio en la estrutura de lista clasificada nodo->infodom = &m_dominio.pixel[i][k][j];
1109.       nodo->sig = m_info_dominio[prim_clase][segnd_clase][i]; m_info_dominio[prim_clase][segnd_clase][i] = nodo;
1110.         }
1111.     }

1112.         // Ahora debe asegurarse que ninguna clase de dominio este actualmente vacia.

1113.     for (i=0; i <= m_max_part-m_min_part; ++i)

1114.     for (prim_clase = 0; prim_clase < 3; ++prim_clase)
1115.       for (segnd_clase = 0; segnd_clase < 24; ++segnd_clase)
1116.         if (m_info_dominio[prim_clase][segnd_clase][i] == NULL) {
1117.                 nodo = (struct dominios_clasificados *)
1118.                       malloc(sizeof(struct dominios_clasificados)); nodo->infodom = &m_dominio.pixel[i][0][0];
1119.                 nodo->sig = NULL;
1120.                 m_info_dominio[prim_clase][segnd_clase][i] = nodo;
1121.         }

1122.           Mensaje(«Terminado «);
1123.           return TRUE;
1124.   }

1125.   long int CCompFrac::EscrAEncab(int tam, long int valor)
1126.   {
1127.       int i;
1128.       //static int ptr = 1, // indica cuantos bits seran empaquetados en «sum»
1129.           //                  sum = 0; // bits empaquetados
1130.       static long int num_bytes_empaq = 0L; // total de bytes a escribir

1131.   // tam == -1 signfica que se ha llegado al final, y se procede a escribir lo que ha quedado aun
```

```
1132.  if (tam == -1 && m_ptr != 1) {
1133.      fputc(m_sum<<(8-m_ptr), m_ArchSal); ++num_bytes_empaq;
1134.      m_num_bytes_empaq++;
1135.              return((long)0);
1136.    }
1137.
1138.  // tam == -2 significa que se esta solicitando el número de bytes escritos if (tam == -2)
1139.            //return(num_bytes_empaq);
1140.                  // en lugar de retornar la var. estatica (que mantiene la cuanta por sobre las llamadas de esta funcion)
1141.                  // se retorna la var. que puede reinciicarse por cada operación (Comp/Descomp)
1142.                return(m_num_bytes_empaq);

1143.    for (i=0; i<tam; ++i, ++m_ptr, valor = valor>>1, m_sum = m_sum<<1)
1144.          {
1145.            if (valor & 1) m_sum |= 1;

1146.                if (m_ptr == 8)
1147.                {
1148.                        fputc(m_sum,m_ArchSal);
1149.                        ++num_bytes_empaq;
1150.                                ++m_num_bytes_empaq;
1151.                        m_sum=0;
1152.                        m_ptr=0;
1153.            }
1154.          }
1155.            return((long)0);
1156.  }


1157.  double CCompFrac::Compara(int atx, int aty, int tamx, int tamy, int prof, double rsum, double rsum2, int dom_x, int dom_y,
1158.            int oper_sim, int *pialfa, int *pibeta)
1159.  {
1160.    int i, j, i1, j1, k,
1161.            dominio_x,
1162.            dominio_y;              // Posicion del dominio
1163.          double pixel,
1164.                  det,                      // el Determinante de la solucion
1165.                  dsum,                   // suma de los valores de dominio
1166.                  rdsum = 0,              // suma de los valores de los valores rango por dominio
1167.                  dsum2,                // suma de los valores de domain^2 (al cuadrado)
1168.                  w2 = 0,               // número total de valores probados
1169.                  rms = 0,                  // diferencia raiz media cuadrática
1170.                  alfa,                 // el factor de escala
1171.                  beta;                     // El desplazamiento

1172.    w2 = tamx * tamy;
1173.    dsum = m_dominio.pixel[prof-m_min_part][dom_y][dom_x].sum; dsum2 = m_dominio.pixel[prof-
1174.          m_min_part][dom_y][dom_x].sum2;
1175.    dominio_x = (dom_x * m_dominio.intrv_h_dom[prof-m_min_part]);
1176.    dominio_y = (dom_y * m_dominio.intrv_v_dom[prof-m_min_part]);
1177.    k = ((dominio_x%2)<<1) + dominio_y%2;
1178.    dominio_x >>= 1;
1179.    dominio_y >>= 1;

1180.            // Las Declaraciones principales son una declaracion «switch», la cual
1181.            // busca a traves de los dominios y rangos para compararlos
1182.
1183.    switch(oper_sim)
1184.    {
1185.            case 0  :
1186.                for (i=atx, i1 = dominio_x; i<atx+tamx; ++i, ++i1)
1187.                  for (j=aty, j1 = dominio_y; j<aty+tamy; ++j, ++j1)
1188.                  {
1189.                          pixel = m_imagen_dividida[k][j1][i1];
1190.                          rdsum += m_imagen[j][i]*pixel;
1191.                  }
1192.                  break;

1193.            case 1  :
```

```
1194.                for (j=aty, i1 = dominio_x; j<aty+tamy; ++j, ++i1)
1195.                  for (i=atx+tamx-1, j1 = dominio_y; i>=atx; —i, ++j1)
1196.                  {
1197.                          pixel = m_imagen_dividida[k][j1][i1];
1198.                                rdsum += m_imagen[j][i]*pixel;
1199.                  }
1200.                       break;

1201.           case 2  :
1202.                for (i=atx+tamx-1, i1 = dominio_x; i>=atx; —i, ++i1)
1203.                  for (j=aty+tamy-1, j1 = dominio_y; j>=aty; —j, ++j1)
1204.                  {
1205.                          pixel = m_imagen_dividida[k][j1][i1];
1206.                          rdsum += m_imagen[j][i]*pixel;
1207.                  }
1208.                       break;

1209.           case 3  :
1210.                for (j=aty+tamy-1, i1 = dominio_x; j>=aty; —j, ++i1)
1211.                  for (i=atx, j1 = dominio_y; i<atx+tamx; ++i, ++j1)
1212.                  {
1213.                                pixel = m_imagen_dividida[k][j1][i1];
1214.                         rdsum += m_imagen[j][i]*pixel;
1215.                       }
1216.                  break;

1217.           case 4  :
1218.                for (j=aty, i1 = dominio_x; j<aty+tamy; ++j, ++i1)
1219.                  for (i=atx, j1 = dominio_y; i<atx+tamx; ++i, ++j1)
1220.                  {
1221.                          pixel = m_imagen_dividida[k][j1][i1];
1222.                          rdsum += m_imagen[j][i]*pixel;
1223.                  }
1224.                  break;

1225.           case 5  :
1226.                for (i=atx, i1 = dominio_x; i<atx+tamx; ++i, ++i1)
1227.                  for (j=aty+tamy-1, j1 = dominio_y; j>=aty; —j, ++j1)
1228.                  {
1229.                          pixel = m_imagen_dividida[k][j1][i1];
1230.                           rdsum += m_imagen[j][i]*pixel;
1231.                  }
1232.                  break;

1233.           case 6  :
1234.                for (j=aty+tamy-1, i1 = dominio_x; j>=aty; —j, ++i1)
1235.                  for (i=atx+tamx-1, j1 = dominio_y; i>=atx; —i, ++j1)
1236.                  {
1237.                          pixel = m_imagen_dividida[k][j1][i1];
1238.                          rdsum += m_imagen[j][i]*pixel;
1239.                  }
1240.                  break;

1241.           case 7  :
1242.                for (i=atx+tamx-1, i1 = dominio_x; i>=atx; —i, ++i1)
1243.                  for (j=aty, j1 = dominio_y; j<aty+tamy; ++j, ++j1)
1244.                  {
1245.                          pixel = m_imagen_dividida[k][j1][i1];
1246.                           rdsum += m_imagen[j][i]*pixel;
1247.                  }
1248.                  break;
1249.                  }

1250.                  det = (tamx*tamy)*dsum2 - dsum*dsum;

1251.      if (det == 0.0)
1252.                          alfa =  0.0;
1253.                else
1254.                          alfa = (w2*rdsum - rsum*dsum)/det;
```

```
1255.  if  (m_solo_positivo && alfa < 0.0) alfa = 0.0;
1256.                          //*conversion forzada
1257.  *pialfa = (int)(0.5 + (alfa + m_max_escala)/(2.0*m_max_escala)*(1<<m_bits_escalam));
1258.  if (*pialfa < 0) *pialfa = 0;
1259.  if (*pialfa >= 1<<m_bits_escalam) *pialfa = (1<<m_bits_escalam)-1;

1260.  // Ahora calcula nuevamente alfa con lo anterior
1261.  alfa = (double)*pialfa/(double)(1<<m_bits_escalam)*(2.0*m_max_escala)-m_max_escala;

1262.  //calcula el desplazamiento
1263.  beta = (rsum - alfa*dsum)/w2;

1264.          // Convierte beta a entero
1265.  // Se usa la informacion señalada de alfa para empaquetar eficientemente
1266.  if (alfa > 0.0)  beta += alfa*NIVELES_GRIS;
1267.          //*conversion forzada
1268.  *pibeta = (int)(0.5 + beta/ ((1.0+fabs(alfa))*NIVELES_GRIS)*((1<<m_bits_desplz)-1));
1269.  if (*pibeta< 0) *pibeta = 0;
1270.  if (*pibeta>= 1<<m_bits_desplz) *pibeta = (1<<m_bits_desplz)-1;
1271.
1272.  // Se calcula nuevamente beta desde el valor como entero
1273.  beta = (double)*pibeta/(double)((1<<m_bits_desplz)-1)*((1.0+fabs(alfa))*NIVELES_GRIS);
1274.
1275.  if (alfa > 0.0) beta  -= alfa*NIVELES_GRIS;

1276.  // Calcula el valor rms basado en el alfa y beta contabilizado
1277.  rms= sqrt((rsum2 + alfa*(alfa*dsum2 - 2.0*rdsum + 2.0*beta*dsum) +
1278.                          beta*(beta*w2 - 2.0*rsum))/w2);
1279.  return(rms);
1280.  }
1281.
1282.
1283.  void CCompFrac::Quadtree(int atx, int aty, int tamx, int tamy,double tol,int prof, HWND hdcSal)
1284.  {
1285.    int i,
1286.          oper_sim,     // Operación de simetria del cuadrado
1287.          ialfa,        // Factor de escalamiento como entero ibeta,                    // Desplazamiento como entero
1288.          mejor_ialfa,  // el mejor ialfa encontrado
1289.          mejor_ibeta,
1290.          mejor_oper_sim,
1291.          mejor_dom_x,
1292.          mejor_dom_y,
1293.          prim_clase,
1294.          la_prim_clase,
1295.          ini_prim_clase,       // ciclo de valores de inicio y fin
1296.          fin_prim_clase,
1297.          segnd_clase[2],
1298.          la_segnd_clase,
1299.          ini_segnd_clase,      // ciclo de valores de inicio y fin  fin_segnd_clase,
1300.          rango_oper_sim[2],    // son las operaciones que llevan a los cuadrados  dom_oper_sim;
1301.          // hacia la posicion canonica
1302.          struct dominios_clasificados *nodo;  // variable desde donde buscar dominios
1303.          double rms, mejor_rms,      // valor rms y valor min encontrados hasta el momento
1304.          sum=0, sum2=0;      // sum and sum^2 del rango de pixeles
1305.          HDC tempDc = GetDC(hdcSal);
1306.          CString msg;
1307.          if (prof < m_min_part)
1308.          {
1309.            if(hdcSal != NULL)
1310.            {
1311.                    msg.Format(«Profundidad \»Quadtree\» = %d»,prof+1);
1312.                    SetWindowText(hdcSal,msg);
1313.                    MoveToEx(m_MemDc,(atx+tamx/4),aty,NULL);
1314.                    LineTo(m_MemDc,(atx+tamx/4),tamy/2);
1315.                    MoveToEx(m_MemDc,atx,(aty+tamy/4),NULL);
1316.                    LineTo(m_MemDc,tamx/2,(aty+tamy/4));
1317.                    BitBlt(tempDc,0,0,m_tamh,m_tamv,m_MemDc,0,0,SRCCOPY);
1318.            }
1319.              Quadtree(atx, aty, tamx/2, tamy/2, tol, prof+1,hdcSal);
```

```
1320.                   if(hdcSal != NULL)
1321.                   {
1322.                           msg.Format(«Profundidad \»Quadtree\» = %d»,prof+1);
1323.                           SetWindowText(hdcSal,msg);
1324.                           MoveToEx(m_MemDc,((atx+tamx/2)+tamx/4),aty,NULL);
1325.                           LineTo(m_MemDc,((atx+tamx/2)+tamx/4),tamy/2);
1326.                           MoveToEx(m_MemDc,(atx+tamx/2),(aty+tamy/4),NULL);
1327.                           LineTo(m_MemDc,tamx/2,(aty+tamy/4));
1328.                           BitBlt(tempDc,0,0,m_tamh,m_tamv,m_MemDc,0,0,SRCCOPY);
1329.                   }

1330.                   Quadtree(atx + tamx/2, aty, tamx/2, tamy/2,tol, prof+1,hdcSal);

1331.                   if(hdcSal != NULL)
1332.                   {
1333.                           msg.Format(«Profundidad \»Quadtree\» = %d»,prof+1);
1334.                           SetWindowText(hdcSal,msg);
1335.                           MoveToEx(m_MemDc,(atx+tamx/4),aty+tamy/2,NULL);
1336.                           LineTo(m_MemDc,(atx+tamx/4),tamy/2);
1337.                           MoveToEx(m_MemDc,atx,((aty+tamy/2)+tamy/4),NULL);
1338.                           LineTo(m_MemDc,tamx/2,((aty+tamy/2)+tamy/4));
1339.                           BitBlt(tempDc,0,0,m_tamh,m_tamv,m_MemDc,0,0,SRCCOPY);
1340.                   }

1341.                   Quadtree(atx, aty + tamy/2, tamx/2, tamy/2,tol,prof+1,hdcSal);

1342.                   if(hdcSal != NULL)
1343.                   {
1344.                           msg.Format(«Profundidad \»Quadtree\» = %d»,prof+1);
1345.                           SetWindowText(hdcSal,msg);
1346.                           MoveToEx(m_MemDc,((atx+tamx/2)+tamx/4),aty+tamy/2,NULL);
1347.                           LineTo(m_MemDc,((atx+tamx/2)+tamx/4),tamy/2);
1348.                           MoveToEx(m_MemDc,atx+tamx/2,((aty+tamy/2)+tamy/4),NULL);
1349.                           LineTo(m_MemDc,tamx/2,((aty+tamy/2)+tamy/4));
1350.                           BitBlt(tempDc,0,0,m_tamh,m_tamv,m_MemDc,0,0,SRCCOPY);
1351.                   }

1352.                   Quadtree(atx + tamx/2, aty + tamy/2, tamx/2, tamy/2,tol,prof+1,hdcSal); return;

1353.   }

1354.           // Se busca la mejor relacion dominio-rango y se procede a guardarla

1355.   mejor_rms = 100000000000.0;

1356.   Clasifica(atx, aty, tamx,tamy,
1357.                   &la_prim_clase, &la_segnd_clase,
1358.                   &rango_oper_sim[0], &sum, &sum2, 1);


1359.   // define cuanto se ha de buscar (como con los parametros -f y -F)
1360.   if (m_clase_completa_de_busq) {
1361.     ini_prim_clase = 0; fin_prim_clase = 3;
1362.   } else {
1363.     ini_prim_clase = la_prim_clase; fin_prim_clase = la_prim_clase+1;
1364.   }

1365.   if (m_subclase_de_busq) { ini_segnd_clase = 0; fin_segnd_clase = 24;
1366.   } else {
1367.     ini_segnd_clase = la_segnd_clase; fin_segnd_clase = la_segnd_clase+1;
1368.  }
1369.     // Estos ciclos for varian de acuerdo a las baderas de optimizacion
1370.   // establecidas para la subclase_de_busq y clase_completa_de_busq==1, se busca en todos
1371.   // los dominios (excepto todas las rotaciones).
1372.   for (prim_clase = ini_prim_clase;
1373.        prim_clase < fin_prim_clase; ++prim_clase)
1374.   for (segnd_clase[0] = ini_segnd_clase;
```

```
1375.     segnd_clase[0] < fin_segnd_clase; ++segnd_clase[0]) {

1376.     // Se debe revisar cada dominio dos veces. Una vez para el escalamiento positivo,
1377.     // y una vez para el negativo. Cada uno tiene su propia clase y operación de simetria (oper_sim)
1378.     if (!m_solo_positivo)
1379.       {
1380.         segnd_clase[1] =
1381.               m_transformada_clase[(prim_clase == 2 ? 1 : 0)][segnd_clase[0]];
1382.         rango_oper_sim[1] =
1383.               m_transformada_rotacion[(la_prim_clase == 2 ? 1 : 0)][rango_oper_sim[0]];
1384.   }

1385.         // solo_positivo es 0 o 1, de tal forma que se pueda o no hacer busqueda
1386.     for (i=0; i<(2-m_solo_positivo); ++i)
1387.     for (nodo = m_info_dominio[prim_clase][segnd_clase[i]][prof-m_min_part];
1388.         nodo != NULL;
1389.         nodo = nodo->sig) {
1390.         dom_oper_sim = nodo->infodom->sym;
1391.                               // La sig. declaracion if establece como tranformar
1392.         // el dominio a rango, que poporciona el saber como obtener
1393.         // cada posición canónica
1394.         if (((dom_oper_sim>3 ? 4: 0) + (rango_oper_sim[i]>3 ? 4: 0))%8 == 0)
1395.               oper_sim = (4 + dom_oper_sim%4 - rango_oper_sim[i]%4)%4;
1396.         else
1397.               oper_sim = (4 + (dom_oper_sim%4 + 3*(4-rango_oper_sim[i]%4))%4)%8;

1398.         rms = Compara(atx,aty, tamx, tamy, prof, sum,sum2,
1399.                               nodo->infodom->dom_x, nodo->infodom->dom_y, oper_sim, &ialfa,&ibeta);
1400.         if (rms < mejor_rms)
1401.         {
1402.               mejor_ialfa = ialfa;
1403.               mejor_ibeta = ibeta;
1404.               m_valrms = mejor_rms = rms;
1405.               mejor_oper_sim = oper_sim;
1406.               mejor_dom_x = nodo->infodom->dom_x;
1407.               mejor_dom_y = nodo->infodom->dom_y;
1408.         }
1409.     }
1410. }

1411. if (mejor_rms > tol && prof < m_max_part)
1412. {
1413.   // No se encontró un buen ajuste suficiente asi que se prosigue con el quadtree EscrAEncab(1,(long)1);
1414.   // Este bit signif. que quadtree continua
1415.     if(hdcSal != NULL)
1416.       {
1417.         msg.Format(«Profundidad \»Quadtree\» = %d»,prof+1);
1418.         SetWindowText(hdcSal,msg);
1419.         MoveToEx(m_MemDc,(atx+tamx/4),aty,NULL);
1420.         LineTo(m_MemDc,(atx+tamx/4),tamy/2);
1421.         MoveToEx(m_MemDc,atx,(aty+tamy/4),NULL);
1422.         LineTo(m_MemDc,tamx/2,(aty+tamy/4));
1423.         BitBlt(tempDc,0,0,m_tamh,m_tamv,m_MemDc,0,0,SRCCOPY);
1424.       }
1425.     Quadtree(atx,aty, tamx/2, tamy/2, tol,prof+1,hdcSal);
1426.                         if(hdcSal != NULL)
1427.                         {
1428.                           msg.Format(«Profundidad \»Quadtree\» = %d»,prof+1);
1429.                           SetWindowText(hdcSal,msg);
1430.                           MoveToEx(m_MemDc,((atx+tamx/2)+tamx/4),aty,NULL);
1431.                           LineTo(m_MemDc,((atx+tamx/2)+tamx/4),tamy/2);
1432.                           MoveToEx(m_MemDc,(atx+tamx/2),(aty+tamy/4),NULL);
1433.                           LineTo(m_MemDc,tamx/2,(aty+tamy/4));
1434.                           BitBlt(tempDc,0,0,m_tamh,m_tamv,m_MemDc,0,0,SRCCOPY);
1435.                         }
1436.           Quadtree(atx+tamx/2,aty, tamx/2, tamy/2,tol,prof+1,hdcSal);
1437.                         if(hdcSal != NULL)
1438.                         {
1439.                           msg.Format(«Profundidad \»Quadtree\» = %d»,prof+1);
1440.                           SetWindowText(hdcSal,msg);
```

```
1441.                                   msg.Format(«Profundidad \»Quadtree\» = %d»,prof+1);
1442.                                   SetWindowText(hdcSal,msg);
1443.                                   MoveToEx(m_MemDc,(atx+tamx/4),aty+tamy/2,NULL);
1444.                                   LineTo(m_MemDc,(atx+tamx/4),tamy/2);
1445.                                   MoveToEx(m_MemDc,atx,((aty+tamy/2)+tamy/4),NULL);
1446.                                   LineTo(m_MemDc,tamx/2,((aty+tamy/2)+tamy/4));
1447.                                   SetWindowText(hdcSal,msg);
1448.                                   BitBlt(tempDc,0,0,m_tamh,m_tamv,m_MemDc,0,0,SRCCOPY);
1449.                           }
1450.                         Quadtree(atx,aty+tamy/2, tamx/2, tamy/2,tol,prof+1,hdcSal);
1451.                           if(hdcSal != NULL)
1452.                           {
1453.                                   msg.Format(«Profundidad \»Quadtree\» = %d»,prof+1);
1454.                                   SetWindowText(hdcSal,msg);
1455.                                   MoveToEx(m_MemDc,((atx+tamx/2)+tamx/4),aty+tamy/2,NULL);
1456.                                   LineTo(m_MemDc,((atx+tamx/2)+tamx/4),tamy/2);
1457.                                   MoveToEx(m_MemDc,atx+tamx/2,((aty+tamy/2)+tamy/4),NULL);
1458.                                   LineTo(m_MemDc,tamx/2,((aty+tamy/2)+tamy/4));
1459.                                   BitBlt(tempDc,0,0,m_tamh,m_tamv,m_MemDc,0,0,SRCCOPY);
1460.                           }
1461.                         Quadtree(atx+tamx/2,aty+tamy/2, tamx/2, tamy/2,tol,prof+1,hdcSal);
1462.                   }
1463.                 else
1464.                 {
1465.                 // El ajuste fue los suficientemente bueno o no se pudo obtener rangos mas pequeños
1466.                 // Asi se se procede a guardar los datos
1467.                 if (prof < m_max_part)  // Si esta en el rango mas pequeño
1468.                   EscrAEncab(1,(long)0);  // entonces se le debe informar al decodificador que el proceso
1469.                   // de quadtree se ha detenido
1470.                 EscrAEncab(m_bits_escalam, (long)mejor_ialfa);
1471.                 EscrAEncab(m_bits_desplz, (long)mejor_ibeta);
1472.                   // Cuando el escalamiento es cero, no se necesita almacenar el dominio
1473.             if (mejor_ialfa != m_ialfa_cero)
1474.             {
1475.                 EscrAEncab(3, (long)mejor_oper_sim);
1476.                 EscrAEncab(m_bits_necesarios[prof-m_min_part], (long)(mejor_dom_y* m_dominio.num_doms_h
1477.                         [prof-m_min_part]+mejor_dom_x));
1478.             }
1479.         }

1480.     }

1481.   void CCompFrac::Particiona_Imagen(int atx, int aty,int tamh, int tamv,double tol,HWND hdcSal)
1482.   {
1483.       int x_exponen,    // tam. de imagen de potencia mayor de 2 que se ajuste
1484.       y_exponen,    // horizontalmente y verticalmente al rectangulo exponen,
1485.       // Tamaño actual de la imagen que esta codificada tam, prof;
1486.       x_exponen = (int)floor(log((double)tamh)/log(2.0));
1487.       y_exponen = (int)floor(log((double)tamv)/log(2.0));
1488.       // exponen es el min entre x_exponen y y_exponen
1489.       exponen = (x_exponen > y_exponen ? y_exponen : x_exponen);
1490.       tam = 1<<exponen;
1491.       prof = m_max_exponente - exponen;
1492.       Quadtree(atx,aty,tam,tam,tol,prof,hdcSal);
1493.       if (tam != tamh)
1494.         Particiona_Imagen(atx+tam, aty, tamh-tam, tamv, tol, hdcSal);
1495.       if (tam != tamv)
          Particiona_Imagen(atx, aty+tam, tam, tamv-tam, tol, hdcSal);

1496.   }

1497.   void CCompFrac::Promedio(int x, int y, int tamx, int tamy, double *sumpix, double *sumpix2)
1498.   {
1499.       register int i,j,k;
1500.       register double pixel;
1501.       *sumpix = *sumpix2 = 0.0;
1502.       k = ((x%2)<<1) + y%2;
1503.       x >>= 1; y >>= 1;
1504.       tamx >>= 1; tamy >>= 1;
```

```
1505.  for (i=x; i<x+tamx; ++i) for (j=y; j<y+tamy; ++j) {
1506.      pixel = m_imagen_dividida[k][j][i];
1507.      *sumpix += pixel;
1508.        *sumpix2 += pixel*pixel;
1509.      }
1510.  }

1511.  void CCompFrac::Promedio1(int x, int y, int tamx, int tamy, double *sumpix, double *sumpix2)
1512.  {
1513.        register int i,j;
1514.        register double pixel;
1515.        *sumpix = *sumpix2 = 0.0;
1516.        for (i=x; i<x+tamx; ++i) for (j=y; j<y+tamy; ++j)
1517.        {
1518.              pixel = (double)m_imagen[j][i];
1519.              *sumpix += pixel;
1520.              *sumpix2 += pixel*pixel;
1521.        }
1522.  }
```

Fin de Archivo Frac.CPP

Archivo COpcion.H

```
1.          // COpcion.h : Archivo de Encabezado
2.          /////////////////////////////////////////////////////////////////////////
3.          // Diálogo CCopcion
4.
5.          class CCopcion : public CDialog
6.          {
7.          // Construcción
8.      public:
9.          CCopcion(CWnd* pPadre = NULL);   // Constructor Estándar
10.
11.         // Datos del Diálogo
12.         // INFO : m_altura es definida por si en alguna actualización
13.         // las imágenes pueden ser rectangulares, mejor que cuadradas.
14.         // Esta variable miembro ya esta conectada a su Control de Edicion, solo
15.         // que este es invisible en el diálogo de opciones Compresión
16.         //{{AFX_DATA(CCopcion)
17.         enum { IDD = IDD_OPCIONESC };
18.         CString m_altura;
19.         int          m_ticd;
20.         int          m_bitsdesplzc;
21.         int          m_minprofrec;
22.         int          m_bitsescalc;
23.         int          m_tol;
24.         CString m_ancho;
25.         CString m_tipo_cont;
26.         double  m_factescalmax;
27.         int          m_maxprofrec;
28.         BOOL    m_24clasdom;
29.         BOOL    m_3clasdom;
30.         BOOL    m_pos;
31.         BOOL    m_idm;
32.         BOOL    m_mostrar;
33.         //}}AFX_DATA
34.         BOOL m_cambio;
35.
36.         // Funciones sobrecargadas
37.         // ClassWizard sobrecargó a las funciones virtuales
38.         //{{AFX_VIRTUAL(CCopcion)
39.         protected:
40.         virtual void DoDataExchange(CDataExchange* pDX);    // soporte DDX/DDV virtual BOOL
41.         OnCommand(WPARAM wParam, LPARAM lParam);
42.         //}}AFX_VIRTUAL

43.         // Implementación
44.         protected:
45.         // Funciones generadas de mapeo de mensajes
46.         //{{AFX_MSG(CCopcion)
47.                 virtual void OnOK();
48.                 afx_msg void OnDefaultc();
49.         //}}AFX_MSG
50.         DECLARE_MESSAGE_MAP()
51.         };

52.         /////////////////////////////////////////////////////////////////////////
53.         // Diálogo CDopcion

54.         class CDopcion : public CDialog
55.         {
56.         // Construcción
57.         public:
58.         CDopcion(CWnd* pPadre = NULL);   // Constructor Estándar
59.
60.         // Datos del Diálogo
61.         //{{AFX_DATA(CDopcion)
```

```
62.                 enum { IDD = IDD_OPCIONESD };
63.                 int             m_numiter;
64.                         BOOL            m_postproc;
65.                         int                     m_scalsal;
66.                         int                     m_bitsdesplz;
67.                         double  m_factescalmax;
68.                         BOOL            m_mostrar;
69.                         int                     m_bitsescal;
70.                         BOOL            m_salida;
71.         //}}AFX_DATA
72.         BOOL m_cambio;

73. // Funciones sobrecargadas
74.         // ClassWizard sobrecargó a las funciones virtuales
75.         //{{AFX_VIRTUAL(CDopcion)
76.                         protected:
77.                         virtual void DoDataExchange(CDataExchange* pDX);    // soporte DDX/DDV
78.         //}}AFX_VIRTUAL
79. // Implementación
80. protected:
81.         // Funciones generadas del mapa de mensajes
82.         //{{AFX_MSG(CDopcion)
83.         virtual void OnOK();
84.         afx_msg void OnDefaultd(); //}}AFX_MSG DECLARE_MESSAGE_MAP()
85. };
```

Fin de Archivo COpcion.H

Archivo COpcion.CPP

```
1.  // COpcion.cpp : Arhcivo de implementación de la clase COpcion
2.  #include «stdafx.h»          // Acceso a las Clases Fundamentales de Microsoft (MFC)
3.  #include «CompFrac.h»
4.  #include «copcion.h»

5.  #ifdef _DEBUG
6.  #undef THIS_FILE
7.  static char BASED_CODE THIS_FILE[] = __FILE__;
8.  #endif

9.  ////////////////////////////////////////////////////////////
10. // Diálogo COpcion

11. COpcion::COpcion(CWnd* pPadre /*=NULL*/)
12.         : CDialog(COpcion::IDD, pPadre)
13. {
14. //{{AFX_DATA_INIT(COpcion)
15.         m_altura = _T(«256»);
16.         m_ticd = 1;
17.         m_bitsdesplzc = 7;
18.         m_minprofrec = 4;
19.         m_bitsescalc = 5;
20.         m_tol = 8;
21.         m_ancho = _T(«256»);
22.         m_tipo_cont = _T(«0»);
23.         m_factescalmax = 1.0;
24.         m_maxprofrec = 6;
25.         m_24clasdom = FALSE;
26.         m_3clasdom = FALSE;
27.         m_pos = FALSE;
28.         m_idm = FALSE;
29.         m_mostrar = TRUE;
30. //}}AFX_DATA_INIT
31. m_cambio = FALSE;
32. }


33. void COpcion::DoDataExchange(CDataExchange* pDX)
34. {
35. CDialog::DoDataExchange(pDX);
36. //{{AFX_DATA_MAP(COpcion)
37. DDX_Text(pDX, IDC_ALTOIMG, m_altura);
38. DDX_Text(pDX, IDC_TAMINTERVCDOM, m_ticd);
39. DDX_Text(pDX, IDC_BITSDESPLZC, m_bitsdesplzc);
40. DDX_Text(pDX, IDC_MINPREC, m_minprofrec);
41. DDX_Text(pDX, IDC_BITSESCALC, m_bitsescalc);
42. DDX_Text(pDX, IDC_TOLERANCIA, m_tol);
43. DDV_MinMaxInt(pDX, m_tol, 1, 15);
44. DDX_CBString(pDX, IDC_ANCHOIMG, m_ancho);
45. DDX_CBString(pDX, IDC_TIPOCONTEN, m_tipo_cont);
46. DDX_Text(pDX, IDC_FACTESCALMAXC, m_factescalmax);
47. DDX_Text(pDX, IDC_MAXPREC, m_maxprofrec);
48. DDX_Check(pDX, IDC_24CLASDOMIN, m_24clasdom);
49. DDX_Check(pDX, IDC_3CLASDOMIN, m_3clasdom);
50. DDX_Check(pDX, IDC_POSTIVO, m_pos);
51. DDX_Check(pDX, IDC_IDM, m_idm);
52. DDX_Check(pDX, IDC_MOSTRQUADTREE, m_mostrar);
53. //}}AFX_DATA_MAP
54. }
```

```
55.  BEGIN_MESSAGE_MAP(CCopcion, CDialog)
56.      //{{AFX_MSG_MAP(CCopcion)
57.      ON_BN_CLICKED(IDC_DEFAULTC, OnDefaultc)
58.      //}}AFX_MSG_MAP
59.  END_MESSAGE_MAP()


60.  ////////////////////////////////////////////////////////////
61.  // Funciones miembro de CDopcion
62.  ////////////////////////////////////////////////////////////
63.  // Diálogo CDopion

64.  CDopcion::CDopcion(CWnd* pPadre /*=NULL*/)
65.      : CDialog(CDopcion::IDD, pPadre)
66.  {
67.  //{{AFX_DATA_INIT(CDopcion)
68.      m_numiter = 10;
69.      m_postproc = TRUE;
70.      m_scalsal = 1;
71.      m_bitsdesplz = 7;
72.      m_factescalmax = 1.0;
73.      m_mostrar = TRUE;
74.      m_bitsescal = 5;
75.      m_salida = FALSE;
76.  //}}AFX_DATA_INIT
77.      m_cambio = FALSE;
78.  }


79.  void CDopcion::DoDataExchange(CDataExchange* pDX)
80.  {
81.  CDialog::DoDataExchange(pDX);
82.  //{{AFX_DATA_MAP(CDopcion)
83.  DDX_Text(pDX, IDC_NUMITER, m_numiter);
84.  DDX_Check(pDX, IDC_POSTP, m_postproc);
85.  DDX_Text(pDX, IDC_ESCALASAL, m_scalsal);
86.  DDX_Text(pDX, IDC_BITSDESPLZD, m_bitsdesplz);
87.  DDX_Text(pDX, IDC_FACTESCALMAXD, m_factescalmax);
88.  DDX_Check(pDX, IDC_MOSTRARPROC, m_mostrar);
89.  DDX_Text(pDX, IDC_BITSESCALD, m_bitsescal);
90.  DDX_Check(pDX, IDC_SALIDA, m_salida);
91.  //}}AFX_DATA_MAP
92.  }


93.  BEGIN_MESSAGE_MAP(CDopcion, CDialog)
94.  //{{AFX_MSG_MAP(CDopcion)
95.      ON_BN_CLICKED(IDC_DEFAULTD, OnDefaultd)
96.  //}}AFX_MSG_MAP
97.  END_MESSAGE_MAP()
98.  ////////////////////////////////////////////////////////////
99.  // Funciones miembro de CDopcion y Ccopcion
100.
101. void CDopcion::OnOK()
102. {
103. // A REALIZAR: Agregar validación extra aqui
104. m_cambio = TRUE;
105. CDialog::OnOK();
106. }

107. void CCopcion::OnOK()
108. {
109. // A REALIZAR: Agregar validación extra aqui m_cambio = TRUE;
110. CDialog::OnOK();
111. }

112. void CCopcion::OnDefaultc()
```

```
113.        {
114.        // A REALIZAR : Agregar el código del manejador del control (botón) aqui
115.        m_altura = _T("256");
116.        m_ticd = 1;
117.        m_bitsdesplzc = 7;
118.        m_minprofrec = 4;
119.        m_bitsescalc = 5;
120.        m_tol = 8;
121.        m_ancho = _T("256");
122.        m_tipo_cont = _T("0");
123.        m_factescalmax = 1.0;
124.        m_maxprofrec = 6;
125.        m_24clasdom = FALSE;
126.        m_3clasdom = FALSE;
127.        m_pos = FALSE;
128.        m_idm = FALSE;
129.        UpdateData(FALSE);
130.        m_cambio = FALSE;
131.        }

132.        void CDopcion::OnDefaultd()
133.        {
134.        // A REALIZAR : Agregar el código del manejador del control (botón) aqui
135.        m_numiter = 10;
136.        m_postproc = TRUE;
137.        m_scalsal = 1;
138.        m_bitsdesplz = 7;
139.        m_factescalmax = 1.0;
140.        m_mostrar = TRUE;
141.        m_bitsescal = 5;
142.        UpdateData(FALSE);
143.        m_cambio = FALSE;
144.        }


145.        BOOL CCopcion::OnCommand(WPARAM wParam, LPARAM lParam)
146.        {
147.        // A REALIZAR: Agregar código especñifico aqui y/o llamar a la clase base

148.        // Verfica que exista cambio en el ComboBox para Tamaño de la Imagen
149.        if (HIWORD(wParam) == CBN_SELCHANGE && LOWORD(wParam) == IDC_ANCHOIMG)
150.        {           UpdateData(TRUE); // Abre actualizacion
151.                    m_altura = m_ancho;
152.                    UpdateData(FALSE);    // ya actualizado
153.        }
154.                    return CDialog::OnCommand(wParam, lParam);
155.        }
```

Fin de Archivo COpcion.CPP

Archivo VisorImg.H

```
1.  // visorimg.h : Archivo de Encabezado
2.  //

3.  /////////////////////////////////////////////////////////////////////////
4.  // Ventana del CVisorImg

5.  class CVisorImg : public CWnd
6.  {
7.  // Construcción
8.  public:
9.          CVisorImg();
10. // Atributos
11. public:
12.         CBitmap *m_bitmap;
13.         CDC *m_GuardaDc;
14.         int m_Tamh;
15.         int m_Tamv;
16.         BOOL m_respaldo;
17. // Operaciones
18. public:
19.         void GuardaDC(HDC pmemoria);
20. // Miembros Sobrecargados
21.         // ClassWizard sobrecargó a las funciones virtuales
22.         //{{AFX_VIRTUAL(CVisorImg)
23.         //}}AFX_VIRTUAL
24. // Implementación
25. public:
26.         virtual ~CVisorImg();

27.         // Mapa de mensajes de las funciones generadas
28. protected:
29.         //{{AFX_MSG(CVisorImg)
30.             afx_msg void OnPaint();
31.             afx_msg int OnCreate(LPCREATESTRUCT lpCreaEstructura);
32.             afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
33.         //}}AFX_MSG
34.         DECLARE_MESSAGE_MAP()
35. };

36. /////////////////////////////////////////////////////////////////////////
```

Fin de Archivo VisorImg.H

Archivo VisorImg.CPP

```
1.  // VisorImg.cpp : Archivo de implementación
2.  //

3.  #include «stdafx.h»          // Acceso a las Clases Fundamentales de Microsoft (MFC)
4.  #include «resource.h»  // declaración princ. de símbolos
5.  #include «visorimg.h»

6.  #ifdef _DEBUG
7.  #undef THIS_FILE
8.  static char BASED_CODE THIS_FILE[] = __FILE__;
9.  #endif

10. ////////////////////////////////////////////////////////////
11. // Clase CVisorImg

12. CVisorImg::CVisorImg()
13. {
14. m_bitmap = new CBitmap;
15. m_GuardaDc = new CDC;
16. m_respaldo = FALSE;
17. }

18. CVisorImg::~CVisorImg()
19. {
20. delete m_GuardaDc;
21. delete m_bitmap;
22. }
23.
24. BEGIN_MESSAGE_MAP(CVisorImg, CWnd)
25. //{{AFX_MSG_MAP(CVisorImg)
26.                 ON_WM_PAINT()
27.                 ON_WM_CREATE()
28.                 ON_WM_SYSCOMMAND()
29. //}}AFX_MSG_MAP
30. END_MESSAGE_MAP()
31. ////////////////////////////////////////////////////////////
32. // Manejadores de mensajes de CVisorImg
33.
34. void CVisorImg::OnPaint()
35. {
36. CPaintDC dc(this); // Contexto de dispositivo para dibujar
37.
38. // A REALIZAR : Agregar el código para el manejador, aqui
39. if(m_respaldo)
40.                 dc.BitBlt(0,0,m_Tamh,m_Tamv,m_GuardaDc,0,0,SRCCOPY);
41.                 // No llamar a CWnd::OnPaint() para el manejo de mesajes OnPaint
42. }

43. void CVisorImg::GuardaDC(HDC pmemoria)
44. {
45. m_respaldo = TRUE;
46. CRect rect;
47.
48. GetWindowRect(rect);
49. m_Tamh = rect.Height();
50. m_Tamv = rect.Width();
51. m_bitmap->CreateCompatibleBitmap(GetDC(),m_Tamh,m_Tamv);
52. m_GuardaDc->CreateCompatibleDC(GetDC());
53. m_GuardaDc->SelectObject(m_bitmap);
54. BitBlt(m_GuardaDc->GetSafeHdc(),0,0,m_Tamh,m_Tamv,pmemoria,0,0,SRCCOPY);
55. }
```

Archivo *VisorImg.CPP*

```
56.     int CVisorImg::OnCreate(LPCREATESTRUCT lpCreaEstructura)
57.     {
58.     if (CWnd::OnCreate(lpCreaEstructura) == -1)
59.             return -1;
60.     // A REALIZAR  Agregar código específico de inicialización
61.     CMenu* pMenuSist = GetSystemMenu(FALSE);
62.
63.     // IDM_PORTAPAPELES debe estar dentro del rango de comandos del sistema.
64.     ASSERT((IDM_PORTAPAPELES & 0xFFF0) == IDM_PORTAPAPELES);
65.     ASSERT(IDM_PORTAPAPELES < 0xF000);
66.     pMenuSist->AppendMenu(MF_SEPARATOR);
67.     pMenuSist->AppendMenu(MF_STRING, IDM_PORTAPAPELES, CString(«&Copiar al Portapapeles»));

68.     return 0;
69.     }

70.     void CVisorImg::OnSysCommand(UINT nID, LPARAM lParam)
71.     {
72.     // A REALIZAR : Agregar código del manejador de mansajes y/o llamar a la función OnSysCommand default
73.
74.        if ((nID & 0xFFF0) == IDM_PORTAPAPELES)
75.        {
76.             // Crea un bitmap y contexto de dispositivo temporales para hacer la copia al Portapapeles
77.             CBitmap bitmapClip;
78.             CDC TempDC;
79.             if(!bitmapClip.CreateCompatibleBitmap(m_GuardaDc,m_Tamh,m_Tamv))
80.                     return;
81.             if(!TempDC.CreateCompatibleDC(GetDC()))
82.                     return;
83.             TempDC.SelectObject(&bitmapClip);
84.             TempDC.BitBlt(0,0,m_Tamh,m_Tamv,m_GuardaDc,0,0,SRCCOPY);

85.             // Abre el Portapapeles
86.             if(!OpenClipboard())
87.                     return;

88.             // Elimina el contenido actual del Portapapeles
89.             EmptyClipboard();

90.             // Proporciona el manej. del bitmap al Portapapeles
91.             SetClipboardData (CF_BITMAP, bitmapClip.m_hObject);

92.             // Evita el destuir la imagen
93.             bitmapClip.Detach();

94.             // Cierra el Portapapeles
95.             CloseClipboard();
96.        }
97.        else
98.             CWnd::OnSysCommand(nID, lParam);
99.     }
```

Fin de Archivo VisorImg.CPP

APENDICE

Manual de Referencia de las Funciones de Librería para Compresión Fractal : FDK v1.2

Este apéndice se compone de los manuales de referencia de todas las funciones contenidas dentro de las librerías del **Paquete de Desarrollo de Aplicaciones para Compresión Fractal** o **FDK**, en su versión genérica 1.2. Tanto para la librería de Compresión como para la de Descompresión el contenido se divide en cuatro secciones, las cuales son :

INICIO. Esta sección explica asuntos relacionados con las compatibilidad de versiones, la utilización en conjunto con otros productos de Compresión Fractal, una descripción de los archivos que componen al paquete y las varias formas de incluir la librería dentro de un proyecto de programación para poder dotar a la Aplicación de las capacidades de Compresión Fractal.

UTILIZACIÓN DE LAS FUNCIONES DE COMPRESIÓN. Esta sección describe la secuencia de llamadas que han de realizarse a las funciones de librería, para asegurar tanto la utilización correcta de las librerías como el buen funcionamiento de la Aplicación. Además se proporcionan diversas explicaciones para entender el cómo y porqué debe realizarse así la secuencia de llamadas a las funciones de compresión y descompresión, y las diversas opciones de que se dispone para utilizar las librerías de acuerdo a las diversas opciones que están disponibles para construir el Formato de Imagen Fractal o FIF.

INDICE DE FUNCIONES. Esta sección es la más extensa y probablemente la más consultada al momento de desarrollar Aplicaciones con capacidad de Compresión Fractal, debido a que son las referencias completas del uso de cada función en especifico, clasificadas por orden alfabético y complementadas de tópicos tales como : comentarios, requisitos previos para su correcto uso y funcionamiento, referencias a funciones relacionadas, valores por omisión, análisis de cada parámetro de la función y de los valores que cada una de ellas retorna.

CÓDIGOS DE ERROR. Esta sección muestra las tablas de códigos de error generales, que las librerías de Compresión Fractal manejan. Es el conjunto de mensajes de la API para Compresión Fractal; es decir son los valores, que en caso de ocurrir alguna falla en la Aplicación son posibles de detectar a través de los valores de retorno de las funciones de las librerías de Compresión Fractal.

MANUAL DE REFERENCIA DE LAS FUNCIONES DE LIBRERÍA PARA COMPRESIÓN FRACTAL

FRACTAL DEVELOPER'S KIT V1.2

Iterated Systems, Inc.

La Dirección WEB de Iterated Systems es http://www.iterated.com

Soporte Técnico:
support@iterated.com

fax: (404) 264-8300
BBS: (404) 264-2770

Codificador FDK de Imágenes

****

Este es un Paquete Fractal para Desarrollares, sobre Windows a 32 bits, de Iterated Systems, Inc.

CONTENIDO

INICIO

¿ Qué es el llamado "Compresor" ? El Codificador *FDK* de Imágenes es también conocido como *Compresor*, debido a que éste convierte imágenes de gran tamaño basadas en pixeles a imágenes de tamaño muy inferior (imágenes en formato FIF).

Compatibilidad con otras versiones y productos

La Versión 1.1 de la interface para programadores del Codificador *FDK* de Imágenes está diseñada para mantener compatibilidad con la versión 5.0 de *Hi-Res*. Las funciones en Hi-Res 5.0 son las mismas que las funciones proporcionadas en este producto. Las funciones de Hi-Res 5.0 tienen el misma sintaxis y salida que las estas funciones.
Importante : El Codificador de Imágenes *FDK* realiza compresión sólo por software, en cambio Hi-Res Versión 5.0, realiza la compresión sólo por hardware. Las Aplicaciones que han sido escritas para utilizar el hardware FTC (tarjetas aceleradoras de compresión, para PCs; desarrolladas por Iterated Systems, Inc.) deben ser modificadas para acoplarse al proceso de compresión sólo por software. (Ver «Cambios con respecto a *Hi-Res* Versión 5.0".)

Esta versión del Compresor crea archivos de formato FIF, los cuales sólo pueden ser descompactados con Aplicaciones que utilicen la Versión 1.1 o posterior del Decodificador *FDK* de Imágenes.

Las Aplicaciones escritas con las funciones de la Versión 3.0 o más reciente del Compresor *Hi-Res* no funcionarán con las Librerías DLL de la versión 1.1 del FDK.

Cambios con respecto a *Hi-Res* Versión 5.0

El único cambio en la interface del programador consta del interruptor para procesamiento únicamente por hardware a únicamente por software.

- Las llamadas a OpenCompressor deben cambiarse para especificar que el tipo de Compresor es por Software.
- Las constantes y códigos de error que se refieren al hardware FTC han sido omitidas de los archivos de inclusión.

Librerías y Archivos del Compresor

Archivos de Inclusión

Incluir los siguientes archivos **.H** en el código de la Aplicación de Compresión.

- COMP_32.H contiene prototipos para las funciones de librería del Compresor.
- ISIERROR.H define los valores de retorno de las funciones del Compresor.

Librerías Windows

COMP_32.DLL es una DLL a 32 bits, la cual contiene al Compresor.
Se incluyen tres librerías. Para cargar en forma implícita al DLL Compresor, se requiere hacer un enlace con la librería que ya ha sido probada para cada compilador, de acuerdo con el que se esté desarrollando la Aplicación :

- COMP_32W.LIB para Watcom 10.0a,
- COMP_32M.LIB para Microsoft Visual C++ 2.1, y
- COMP_32B.LIB para Borland C++ 4.5.

Inclusión del Compresor en una Aplicación

1. Incluir COMP_32.H y ISIERROR.H en los módulos de su programa de Aplicación en C o C++, el cual llamará a las funciones del Compresor proporcionadas en las librerías.

   Si se está desarrollando la Aplicación en un lenguaje diferente a C o C++, probablemente se necesitará crear archivos de inclusión equivalentes a COMP_32.H y ISIERROR.H, con las definiciones de constantes, definiciones tipo, y funciones prototipo en la forma apropiada para el lenguaje que utilice.

2. COMP_32.DLL llama a DECO_32.DLL, el cual se proporciona también dentro del *FDK*. Asegurarse de que la librería DECO_32.DLL se encuentre en el mismo directorio que COMP_32.DLL, o sobre la misma ruta desde la variable de sistema "path".

3. Para cargar implícitamente a COMP_32.DLL, debe hacerse un enlace a la librería de importación apropiada COMP_32X.LIB dentro de la Aplicación, donde X indica el compilador correspondiente, como se muestra en la lista de la sección anterior.
   Si se está usando un compilador diferente, es necesario revisar la documentación del compilador para localizar las instrucciones de cómo crear una librería de importación a partir de una librería DLL.
   Para cargar explícitamente a COMP_32.DLL (con la función *LoadLibrary*), no necesita enlazarse a la librería de importación.

4. Utilizar las funciones del Compresor como se explica más adelante.

USO DE LAS FUNCIONES DE COMPRESIÓN

Estas secciones muestran, en secuencia, cómo pueden ser utilizadas las funciones del Codificador *FDK* de Imágenes. Se muestran con un ejemplo sencillo. Sin embargo, el uso de estas funciones pueden ser más complejo.

Dependiendo de las características de compresión del FDK, ya sea estándar o incremental que utilice, es necesario asegurarse de leer las secciones correspondientes, para obtener una descripción de cómo se utiliza cada función.

Inicialización del Compresor

Seguir estos pasos para inicializar cualquier Aplicación de Compresión.

1. Comprobar la Versión de la Librería

Llamar a *GetCompVersion* para asegurarse de que la Aplicación está utilizando las mismas DLLs del Compresor para las cuales fue escrita. Esta versión de las DLLs es la 6.0. Una Aplicación que haya sido escrita con esta versión del Codificador *FDK* de Imágenes será capaz de utilizar la Versión 6.0 o posterior de las DLLs.

2. Abrir una Instancia del Compresor

Utilizar a *OpenCompressor* para comenzar el proceso de Compresión. Una llamada a *OpenCompressor* debe preceder a las llamadas a todas las demás funciones del Compresor (excepto *GetCompVersion*). *OpenCompressor* inicializa una instancia del Compresor, la cual es una sesión de uso del mismo. *OpenCompressor* retorna un manejador (*handle*) de la instancia, el cual es un entero único que identifica a la instancia. Debe hacerse referencia a este manejador para utilizar cualquier otra función de compresión.
Por cada instancia, los parámetros que definen a las operaciones de compresión pueden ser configurados y leídos, y permanecerán en memoria hasta que sean reinicializados (con *ResetCompressor* o cualquiera de las funciones con Parámetros de Entrada), o hasta que la instancia sea cerrada (con *CloseCompressor*).

Más de un espacio de memoria (bufer[1]) de entrada puede ser compactado al utilizar una sola instancia, sin embargo sólo un espacio de memoria puede ser compactado a la vez. Para compactar múltiples espacios de memoria de imágenes simultáneamente, deben abrirse múltiples instancias del Compresor.

Selección del Método de manejo de Memoria

La Aplicación debe colocar y liberar espacios de memoria para el archivo de la imagen de entrada y el archivo con formato de salida FIF. El Compresor no tiene funciones para colocar y liberar espacios de memoria que puedan ser utilizados en la Aplicación, sin embargo este coloca memoria interna que dicha Aplicación utiliza para y durante el proceso.
El Compresor proporciona dos métodos de manejo de espacio de memoria :

- el método estándar, el cual lee completamente los datos de la imagen de entrada a partir de un bloque de memoria en una sola vez, o
- el método incremental, el cual lee los datos de la imagen de entrada en pequeños incrementos, compacta cada incremento en un espacio de memoria de salida separado, y entonces ensambla los datos dentro de un espacio de memoria de salida

El método incremental es más eficiente con la memoria cuando los datos de entrada comienzan a ser leídos a partir de un archivo. Esto reduce la cantidad de memoria que debe ser colocada en un cierto momento, sin embargo esto puede incrementar el tiempo de compresión. El archivo de formato FIF resultante es idéntico a aquel producido mediante el método estándar.

Instauración del manejo Estándar de Memoria

Seguir estos pasos para establecer la compresión estándar.

1. Colocar el espacio de memoria y establecer la Resolución

Colocar un espacio de memoria para la imagen de entrada completa, y cargar los datos de la imagen dentro de dicho espacio.
Llamar a *SetImageResolution* para especificar al Compresor el tamaño de la memoria de datos de entrada. La llamada es obligatoria, para permitir al Compresor estimar el tamaño del espacio de memoria de salida y asegurar que se leerán los datos correctamente. Opcionalmente, se pueden especificar la unidades y medidas que vayan a ser almacenadas dentro del archivo FIF para utilizarlos al momento de realizar la descompresión.

2. Establecer la Memoria de Datos de Entrada

Llamar a *SetImageBuffer* para especificar la memoria de datos de entrada para el Compresor.

3. Establecer el Formato de Color de Entrada

Utilizar *SetInputFormat* para indicarle al Compresor el orden en el cual se van a leer el color y las filas de pixeles de los datos de entrada.
Utilizar *GetInputFormat* en cualquier momento si se necesita leer los parámetros de configuración actual.

4. Obtener el Tamaño Máximo del Archivo de Salida

Llamar a *GetMaxFIFSize* para obtener el tamaño más grande posible del espacio de memoria de salida después de la compresión. Utilizar este valor para establecer la memoria de salida.

Instauración del manejo Incremental de Memoria

Siga estos pasos para establecer la Compresión de tipo incremental.

1. Establecer la Resolución de Entrada

Llamar a *SetImageResolution* para indicarle al Compresor el tamaño de la memoria de datos de entrada. La llamada es obligatoria, para permitir al Compresor calcular el tamaño de la memoria incremental de salida (ver la función *StartIncrementalCompression*). Opcionalmente, se pueden especificar las unidades y medidas que van a ser almacenadas en el archivo FIF para su uso al momento de la descompresión.

2. Establecer el Formato del Color de Entrada

Utilizar *SetInputFormat* para indicarle al Compresor el orden en el cual se van a leer el color y las filas de pixeles de los datos de entrada.
Utilizar *GetInputFormat* en cualquier momento si se requiere leer los parámetros de configuración actual.

Instauración Completa

Algunos de los parámetros descritos en esta sección pueden ser reinicializados a sus valores por omisión con la función *ResetCompressor*.

1. Establecer el Formato (de Salida) FIF

Para crear un archivo FIF de escala de grises, utilizar *SetFIFFormat* para establecer la bandera de escala de grises activa. Esta llamada es opcional. La configuración por omisión de esta bandera es producir a la salida los mismos colores que están en la entrada.
Usar *GetFIFFormat* para obtener los parámetros de configuración actual.

2. Establecer la Rapidez de la Compresión

Usar *SetCompressionSpeed* para especificar la rapidez contra el factor de tamaño del archivo que va desde 1 (compresión muy lenta, archivos muy pequeños) hasta 100 (compresión rápida, archivos grandes). El valor por omisión es 1. Usar *GetCompressionSpeed* para comprobar la configuración actual de esta parámetro.

3. Obtener el Tamaño Estimado

El tamaño del archivo de salida FIF puede variar dependiendo del tamaño de la imagen de entrada, el factor de calidad y la rapidez. Para derivar a una compresión anteriormente estimada, llamar a *GetFIFSizeEstimate*.
El tamaño del archivo de salida FIF es sólo una estimación, y puede variar dependiendo del contenido de la imagen. No utilice este valor para establecer el espacio de memoria para la salida; utilizar en cambio a *GetMaxFIFSize*.
Debido a que el Compresor debe utilizar el tamaño de la imagen de entrada para calcular la estimación, la utilización de la función *SetImageResolution* es un requisito previo.

4. Establecer el Tamaño del Cuadrado de Búsqueda

Un **cuadrado de búsqueda** define el área rectangular máxima en la cual el Compresor busca patrones Fractales. Generalmente, un cuadrado pequeño resulta en una compresión muy rápida, y en un archivo de salida FIF grande que requerirá menos memoria para su descompresión. Un cuadrado de búsqueda grande resulta en un proceso de compresión más lento y complejo.
El tamaño por omisión del cuadrado de búsqueda es de 256 x 256 pixeles, y es conveniente para la mayoría de imágenes de entrada. Podrá modificarse el tamaño del cuadrado de búsqueda, mediante la función *SetSearchBoxDimensions*, bajo cualquiera de las siguientes condiciones.

- Si la imagen de entrada es sólo ligeramente mayor a 320 x 200, la compresión será rápida y resultará en la misma o en una mejor calidad si existe sólo un cuadrado de búsqueda. Establecer las dimensiones del cuadrado de búsqueda al mismo tamaño de la imagen.

- Si ya se utilizó a *SetCompressionSpeed* para controlar el tamaño del archivo y la rapidez de compresión sin obtener los resultados deseados, intentar ajustar el tamaño del cuadrado de búsqueda. (La función *SetSearchBoxDimensions* no es redundante o reiterativa para la función *SetCompressionSpeed*.)

5. Establecer la Tabla de Colores para la Descompresión

El Compresor no realiza un mapeo del color, sin embargo el Descompresor si permite el mapeo del color mientras este descomprime los datos del archivo FIF. Con la función *SetFIFColorTable*, se puede almacenar en el archivo FIF, una tabla de colores para que el Descompresor la utilice.

6. Habilitar la Cancelación de la Compresión

Para permitir al usuario final cancelar la operación de Compresión, agregar una función propia, de retorno de llamada (*Callback*), a la Aplicación y especificarla al Compresor con *SetCompressCallback*. Esta función le permitirá a la Aplicación monitorear un diálogo de situación o estado, y responder al comando "cancelar" a partir del cual el usuario final interrumpirá la compresión.

Realización de la Compresión Estándar

1. Compactar los Datos

Llamar a *CompressToBuffer* para realizar el trabajo de Compresión. Con esta llamada, se especificará el factor de calidad para la compresión, sobre una escala de 1 a 100.

2. Escritura de los Datos a un archivo de formato FIF

La información en formato FIF ahora es almacenada en el espacio de memoria de salida. Escribir esta información a un archivo FIF.

3. Limpiar el Espacio de memoria o Bufer de Entrada

Llamar a *ClearImageBuffer* para inicializar el apuntador a la memoria de la imagen de entrada y liberar las variables internas.

Realización de la Compresión Incremental

1. Comenzar la Compresión Incremental

Llamar a *StartIncrementalCompression* para comenzar el proceso incremental. Esta función retorna el tamaño máximo de la memoria de salida de tipo incremental. Utilizar este valor para establecer el espacio de memoria. La función *StartIncrementalCompression* también retorna el número de paneles del proceso incremental en los cuales el Compresor dividirá a la imagen, este valor puede ser usado dentro de la estructura de ciclo, en el paso 4.

2. Compactar cada Incremento

Realizar los siguientes pasos iterativamente (dentro de una estructura de ciclo), para compactar cada incremento de la imagen de entrada.

A. Leer las Coordenadas de Pixel

Utilizar a *GetNextImageRect* para leer las coordenadas de pixel para el siguiente incremento rectangular de la imagen de entrada. Colocar un espacio de memoria de entrada de tipo incremental para contener a estos datos, y cargar la información dentro del espacio de memoria. *GetNextImageRect* retorna un código de error y coordenadas nulas si no existen más paneles que comprimir.

B. Establecer el espacio de Memoria Incremental de Entrada

Llamar a *SetImageBuffer* para especificar el espacio de memoria incremental de entrada al Compresor.

C. Comprimir la Información

Llamar a *CompressToBuffer* para compactar los datos dentro de la memoria incremental de entrada. *CompressToBuffer* carga la información compactada en la memoria de salida de tipo incremental (dicha memoria debe haber sido ya establecida después de llamar a *StartIncrementalCompression*).

D. Obtener el Tamaño de la Información Compactada

La Función *CompressToBuffer* retorna el tamaño actual de la información compactada. Usar este valor para establecer un nuevo bloque de memoria de ese tamaño (el cual será más pequeño que el bloque de memoria de salida de tipo incremental). Mover la información compactada desde el bloque de memoria de salida de tipo incremental hacia el nuevo bloque de memoria. Mantener un arreglo de apuntadores a estos bloques de memoria de salida (para que dicho arreglo sea usado por *EndIncrementalCompression*).

E. Limpiar (Inicializar) el Espacio de Memoria

Llamar a *ClearImageBuffer* para inicializar el apuntador a la memoria incremental de entrada y liberar las variables internas.

F. Salir del Ciclo de Iteraciones

Salir del ciclo ya sea mediante la contabilización del número de paneles compactados, retornados por *StartIncrementalCompression*, o mediante la prueba de las coordenadas nulas retornadas por *GetNextImageRect*.

3. Obtener el Tamaño Máximo del Archivo de Salida

Llamar a *GetMaxFIFSize* para obtener el tamaño máximo posible de la memoria de salida FIF. Usar este valor para establecer la memoria que mantendrá a la imagen completa compactada.
Nótese que la llamada a *GetMaxFIFSize* viene después de la compresión en la compresión de tipo incremental. Esta llamada debe ocurrir antes de la Compresión para el caso del manejo estándar del espacio de memoria.

4. Finalización de la Compresión Incremental

Llamar a *EndIncrementalCompression*, pasándole el arreglo de apuntadores a las memorias en donde la información compactada está almacenada. La función *EndIncrementalCompression* ensambla los incrementos, y los carga dentro de la memoria de salida FIF.

5. Escribir la Información en el Archivo FIF

Escribir la información de la memoria de salida FIF hacia un archivo de formato FIF.

Inicialización después de la Compresión

1. Compactar Otro Archivo

CompressToBuffer no reinicializa ningún parámetro del Compresor. Se mantendrá la configuración actual para la siguiente tarea de compresión, a menos de que se utilice la función *ResetCompressor*, para reinicializar los parámetros a sus valores por omisión. Para comprobar la configuración actual, usar cualquier función que comience con *Get—*.
En este punto, la Aplicación puede ramificarse para permitir al usuario final compactar otra imagen, comprimir la misma imagen a guardar o volver a introducir los parámetros de compresión, o salir del proceso de compresión.

2. Reinicialización de los Parámetros del Compresor

Si es apropiado, usar *ResetCompressor* para retornar todos los parámetros internos a sus valores por omisión. Esta llamada mantiene a la instancia del Compresor abierta.

3. Cerrar la Instancia del Compresor

Para eliminar esta instancia del Compresor, llamar a *CloseCompressor*. Esta función libera a los bloques internos de memoria del Compresor y libera al manejador de este. Si su Aplicación inicializa más de una instancia en un momento, ahorrará memoria al cerrar las instancias que no estén en uso.

INDICE DE FUNCIONES DE COMPRESIÓN

1. ClearImageBuffer

Compresor **Funciones de Compresión**

Inicializa el apuntador a la memoria interna de la imagen de entrada.

Sintaxis
```
extern long ClearImageBuffer(
 long handle
);
```

Parámetros **handle**
Entrada. Identifica a la instancia del Compresor, que es parámetro de salida en la función *OpenCompressor*.

Valor de Retorno
- ISI_OK si la ejecución fue exitosa.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO.
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia es no válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia no es inicializada.

Comentarios Llamar a esta función para inicializar la memoria de la imagen de entrada que previamente ha sido establecida con la función *SetImageBuffer*.

Debe llamarse a *ClearImageBuffer* antes de llamar a *SetImageBuffer* nuevamente, o antes de llamar a *CloseCompressor*.

Requisitos Previos
OpenCompressor, *SetImageBuffer*.

2. CloseCompressor

Compresor **Funciones de Limpieza-Reinicio**

Cierra una instancia del Compresor y libera cualquier espacio de memoria establecido internamente, asociado con esta instancia.

Sintaxis
```
extern long CloseCompressor(
 long handle
);
```

Parámetros **handle**
Entrada. Identifica a la instancia a eliminar, que es un parámetro de salida de la función *OpenCompressor*.

Valor de Retorno
- ISI_OK si la ejecución fue exitosa.

- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO.
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia es no válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia es no inicializada.

Comentarios Esta función debería ser llamada después de que la aplicación ha completado totalmente el proceso de Compresión.
Asegurarse de llamar primero a *ClearImageBuffer*, si la memoria de la imagen de entrada ha sido establecida con la llamada a *SetImageBuffer*.

Requisitos Previos
OpenCompressor.

---

3. CompressToBuffer

Compresor **Funciones de Compresión**

Comprime el contenido de la memoria de entrada al nivel de calidad especificado.

Sintaxis
```
extern long CompressToBuffer(
long handle,
long Quality,
unsigned char *pFIFBuffer,
unsigned long *pActualFIFSize
);
```

Parámetros
handle
Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función *OpenCompressor*.
Quality
Entrada. Puede ser cualquier valor entero desde 1 hasta 100, 100 es la mejor calidad.
pFIFBuffer
Salida. Es la memoria FIF. Esta memoria ya debe estar establecida.
pActualFIFSize
Salida. Es el tamaño real necesario para el archivo de salida FIF o para el archivo de tipo incremental.

Valor de Retorno
- ISI_OK si la ejecución fue exitosa.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO.
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia es inválido.
- COMP_NOT_INITIALIZED_ERROR si la instancia no esta inicializada.
- COMP_IMAGE_SIZE_ERROR si no se proporcionó el tamaño de la imagen.
- COMP_QUALITY_ERROR si el factor de calidad no es un entero entre 1 y 100.
- COMP_NO_IMAGE_BUFFER_ERROR si la memoria que contiene a la imagen no fue especificada con *SetImageBuffer*.
- Error de Memoria si ésta se saturó.

Comentarios Utilizar a *CompressToBuffer* para realizar una compresión de tipo incremental o no incremental.

La configuración de la calidad afectará al tamaño del archivo. La más alta calidad producirá una archivo grande para una imagen dada. (La rapidez, la cual también afecta al tamaño del archivo final, se establece con *SetCompressionSpeed*.)
Al realizar una compresión del tipo incremental, se puede hacer uso del parámetro *ActualFIFSize* para establecer la memoria más pequeña de salida, y copiar la memoria FIF retornada a la memoria pequeña

para conservar espacio de memoria.

Requisitos Previos

OpenCompressor, SetInputFormat, SetImageResolution, GetMaxFIFSize, SetImageBuffer.

Véase *SetFIFFormat, SetCompressionSpeed, CloseCompressor.*

---

4. EndIncrementalCompression

Compresor **Funciones para Operaciones de tipo Incremental**

Combina la memoria de salida de una compresión tipo incremental dentro de un espacio de memoria FIF, y termina el proceso de compresión incremental.

Sintaxis

```
extern long EndIncrementalCompression(
  long handle,
  unsigned char *pFinalFIFBuffer,
  unsigned char **pIncrementalFIFBuffers,
  unsigned long *pActualFIFSize
);
```

Parámetros

handle
Entrada. Identifica la instancia del Compresor, que es parámetro de salida de la función *OpenCompressor*.
pFinalFIFBuffer
Salida. Es la memoria de salida FIF. Esta memoria debe ya estar establecida.
pIncrementalFIFBuffers
Entrada. Es un arreglo de apuntadores a la memoria incremental de salida FIF, que contiene los resultados de la llamadas previas a *CompressToBuffer*.
pActualFIFSize
Salida. Es el tamaño real necesario para el nuevo archivo FIF.

Valor de Retorno

- ISI_OK si la ejecución fue exitosa.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO.
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia es No válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia no ha sido inicializada.

Comentarios Después de que la última sección de la imagen ha sido compactada mediante una llamada a *CompressToBuffer*, esta función es utilizada para construir la memoria FIF válida para representar a la imagen completa a partir de los espacios de memoria incremental FIF de salida.

Establecer la memoria FIF final antes de llamar a esta función. Utilizar a *GetMaxFIFSize* (después de comprimir todos los incrementos mediante *CompressToBuffer*) para determinar el tamaño final de la memoria de salida.
Después de llamar a esta función, escribir la imagen completa FIF en un archivo FIF.

Requisitos Previos

OpenCompressor, StartIncrementalCompression, y las funciones dentro de la estructura de ciclo para la compresión de tipo incremental.

Véase *GetMaxFIFSize.*

5. GetCompressionSpeed

Compresor **Funciones de Implantación**

Obtiene el factor de rapidez que ha de ser utilizado por el Compresor, parámetro establecido mediante *SetCompressionSpeed*.

Sintaxis

```
extern long GetCompressionSpeed(
  long handle,
  long *pCompressionSpeed
);
```

Parámetros

handle
Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función *OpenCompressor*.
pCompressionSpeed
Salida. Es el factor de rapidez de compresión, y está entre 1 y 100, donde 100 es la rapidez mayor.

Valor de Retorno

- ISI_OK si la ejecución fue exitosa.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO.
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia es No válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.

Comentarios Esta configuración varía desde 1 a 100, donde 100 proporciona la compresión más rápida y 1 proporciona la compresión más lenta. Al incrementar la rapidez (compresiones más rápidas), el tamaño del archivo compactado también se incrementa. La calidad permanece constante sin importar cuál sea la configuración de la rapidez

Valor por Default
1 (para archivos más pequeños, pero la compresión es la más lenta).

Requisitos Previos
OpenCompressor.

Véase *SetCompressionSpeed*.

6. GetCompVersion

Compresor **Funciones de Inicio**

Obtiene el número de versión actual de la librería del Compresor.

Sintaxis

```
extern long GetCompVersion(
  long *pPrimaryVersion,
  long *pSecondaryVersion
);
```

Parámetros **pPrimaryVersion**
Salida. Es el número primario de la versión. El número primario de la versión para esta DLL del Compresor es 6.
pSecondaryVersion
Salida. Es el número secundario de la versión (por ejemplo, el de una revisión). El número secundario de la versión de esta DLL es 0.

Valor de Retorno
ISI_OK si la ejecución fue exitosa.

Comentarios Utilizar esta función para verificar la versión actual de la librería COMP_32.DLL instalada. Cualquier Aplicación que utilice esta librería debería comprobar el número de versión de la DLL para asegurarse de que se está trabajando con la misma versión, o más reciente, que la versión de la DLL con la cual la Aplicación fue desarrollada. Esta DLL es la versión 6.0.

Requisitos Previos
Ninguno.

7. GetFIFFormat

Compresor **Funciones de Implantación**

Devuelve el valor de la bandera para el formato de salida, que fue establecido mediante la función *SetFIFFormat*.

Sintaxis
```
extern long GetFIFFormat(
  long handle,
  unsigned long *pGrayScale
);
```

Parámetros **handle**
Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función *OpenCompressor*.
pGrayScale
Salida. Los valores probables son *TRUE* o *FALSE*. *TRUE* convierte cualquier formato de entrada en color a escala de grises. *FALSE* no realiza la conversión del formato de color.

Valor de Retorno

- ISI_OK si la ejecución fue realizada con éxito.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO.
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.

Comentarios Cuando pGrayScale es *FALSE*, el formato de color del archivo FIF es el mismo que el formato de color del archivo de entrada.

Valor por Default

FALSE; el formato de salida es el mismo que el formato de entrada.

Requisitos Previos

OpenCompressor.

Véase *SetFIFFormat*.

8. GetFIFSizeEstimate

Compresor **Funciones de Implantación**

Obtiene una estimación del tamaño del archivo de salida FIF, al proporcionarle el tamaño de la imagen de entrada, el factor de calidad y de rapidez.

Sintaxis

```
extern long GetFIFSizeEstimate(
  long handle,
  long Quality,
  long Speed,
  unsigned long *pFIFSize
);
```

Parámetros **handle**
Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función OpenCompressor.
Quality
Entrada. Puede ser cualquier valor entero desde 1 hasta 100, donde 100 representa la mejor calidad.
Speed
Entrada. Puede ser cualquier valor entero desde 1 hasta 100, donde 100 representa a la mayor rapidez de compresión.

pFIFSize
Salida. Es el tamaño estimado del archivo de salida FIF.

Valor de Retorno

- ISI_OK si la ejecución fue exitosa.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO.
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.
- COMP_IMAGE_SIZE_ERROR si el tamaño de la imagen no es proporcionado.
- COMP_SPEED_ERROR si el factor de rapidez no es un entero entre 1 y 100.
- COMP_QUALITY_ERROR si el factor de calidad no es un entero entre 1 y 100.

Comentarios Esta función proporciona sólo una estimación del tamaño dela rchivo de salida. El tamaño real puede variar dependiendo del contenido de la imagen. Para determinar el tamaño máximo del archivo de salida llamar a *GetMaxFIFSize*. Para obtener mejores resultados, utilizar a *GetMaxFIFSize* para determinar el tamaño de la memoria de salida.

Requisitos Previos
OpenCompressor, SetImageResolution

Véase ***GetMaxFIFSize, SetCompressionSpeed, CompressToBuffer.***

---

9. GetImageResolution

Compresor **Funciones de Implantación**

Obtiene los valores actuales de los parámetros de la resolución de entrada para el Compresor.

Sintaxis

```
extern long GetImageResolution(
long handle,
unsigned long *pPixelWidth,
unsigned long *pPixelHeight,
unsigned long *pPhysicalWidth,
unsigned long *pPhysicalHeight,
unsigned long *pPhysicalUnits,
long *pPhysicalUnitScaleFactor
);
```

Parámetros **handle**
Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función OpenCompressor.

pPixelWidth, pPixelHeight
Salida. Es el ancho y altura de la imagen original en pixeles.
pPhysicalWidth, pPhysicalHeight
Salida. Es el ancho y altura de la imagen original en unidades físicas.
pPhysicalUnits
Salida. Son las unidades físicas mediante las cuale se expresan *pPhysicalWidth y pPhysicalHeight. Estos parámetros pueden tener cualquiera de estas unidades físicas:*

- **METER_UNITS**
- **INCH_UNITS**
- **UNKNOWN_UNITS**

pPhysicalUnitScaleFactor
Salida Potencia de 10 de la unidad, entre -128 y 127.

Valor de Retorno

- ISI_OK si la ejecución fue realizada con éxito.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO.
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.

Comentarios Esta función puede ser llamada en cualquier momento para comprobar los valores de la resolución de entrada, que son establecidos mediante la función *SetImageResolution.*

Requisitos Previos
OpenCompressor.

Véase *SetImageResolution.*

---

10. GetInputFormat

Compresor **Funciones de Compresión**

Obtiene los valores actuales de los parámetros del formato de entrada para el Compresor.

Sintaxis
```
extern long GetInputFormat(
  long handle,
  long *pColorPlane0,
  long *pColorPlane1,
  long *pColorPlane2,
```

```
    long *pColorPlane3,
    long *pRowOrder
);
```

Parámetros **handle**

Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función *OpenCompressor*.

pColorPlane0, pColorPlane1, pColorPlane2, pColorPlane3

Salida. Los valores para las superficies de color. Las constantes se definen como sigue:

- RED8
- GREEN8
- BLUE8
- NOT_USED (si se requieren menos de cuatro)
- BLANK8
- GRAY8

pRowOrder

Salida. Indica en qué orden los datos dentro de la memoria de entrada serán procesados. Tomar el valor *TOP_LEFT* o *BOTTOM_LEFT* para indicar dónde comienzan los datos de la imagen.

Valor de Retorno

- ISI_OK si la ejecución fue realizada con éxito.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO.
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.
- COMP_ILLEGAL_IMAGE_FORMAT_ERROR si no existe formato, o ha sido establecido un formato No soportado.

Comentarios Esta función puede ser llamada en cualquier momento para determinar el estado actual de los parámetros del formato de entrada. Llamar a *SetInputFormat* para realizar cambios a estos parámetros.

Requisitos Previos

OpenCompressor, *SetInputFormat*.

---

11. GetMaxFIFSize

Compresor **Funciones de Compresión**

Retorna el tamaño necesario de la memoria de salida más grande posible para contener los datos compactados.

Sintaxis
```
extern long GetMaxFIFSize(
  long handle,
  unsigned long *pMaxFIFSize
);
```

Parámetros **handle**
Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función *OpenCompressor*.
pMaxFIFSize
Salida. Es el tamaño más grande posible de la memoria de salida FIF.

Valor de Retorno
- ISI_OK si la ejecución fue realizada con éxito.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO.
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.
- COMP_IMAGE_SIZE_ERROR si el tamaño de la imagen No ha sido establecido.

Comentarios Esta función utiliza la resolución de la imagen de entrada, establecida mediante *SetImageResolution*, para determinar el tamaño máximo de la memoria FIF que podría ser retornado por la llamada a *CompressToBuffer*.

Utilizar este número para establecer una cantidad grande de memoria para recibir los datos FIF compactados, creados mediante *CompressToBuffer* o ensamblados a partir de bloques de memoria de tipo incremental mediante *EndIncrementalCompression*.

Para una compresión de tipo incremental, llamar a esta función después de que todos los incrementos hayan sido compactados para obtener la mejor estimación del tamaño final de la memoria FIF.

Requisitos Previos
OpenCompressor, *SetImageResolution*.

---

12. GetNextImageRect

Compresor **Funciones para Operaciones de tipo Incremental**

Retorna las coordenadas y dimensiones, en pixeles, de la siguiente región de la imagen a ser compactada con una Compresión de tipo incremental.

Sintaxis
```
extern long GetNextImageRect(
long handle,
```

```
unsigned long *pPixelX,
  unsigned long *pPixelY,
  unsigned long *pPixelWidth,
  unsigned long *pPixelHeight
);
```

Parámetros **handle**
Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función *OpenCompressor*.
pPixelX, pPixelY
Salida. Son las coordenadas de pixel x, y de la esquina superior izquierda de la siguiente región rectangular de tipo incremental.
pPixelWidth, pPixelHeight
Salida. El ancho y altura, en pixeles, de la siguiente región rectangular de tipo incremental.

Valor de Retorno

- ISI_OK si la ejecución fue realizada con éxito.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO.
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.
- COMP_NOT_INCREMENTAL_ERROR si la función StartIncrementalCompression no ha sido llamada.
- COMP_INCREMENTAL_COMPRESSION_COMPLETE cuando la compresión de tipo incremental se ha completado.

Comentarios *GetNextImageRect* debe ser llamada para determinar qué pieza de la imagen de entrada es la siguiente que el Compresor requerirá. Utilizar *GetNextImageRect* en una estructura de ciclo para comprimir a la imagen completa en forma incremental.
Las coordenadas del área rectangular se originan en la esquina superior izquierda de la imagen de entrada, y se leen de izquierda a derecha y de arriba hacia abajo.
Después de cada llamada a *GetNextImageRect*, cargar la pieza indicada de la imagen dentro de la memoria de entrada de tipo incremental, y entonces llamar a *SetImageBuffer*, *CompressToBuffer* y *ClearImageBuffer*. Cuando *GetNextImageRect* retorna el valor COMP_INCREMENTAL_COMPRESSION_COMPLETE y las coordenadas retornadas tienen valor NULO, el proceso de compresión ha finalizado.

Requisitos Previos
OpenCompressor, *SetImageResolution*, *StartIncrementalCompression*.

Véase *CompressToBuffer*.

13. GetSearchBoxDimensions

Compresor **Funciones de Implantación**

Recupera la configuración de las dimensiones del Cuadro de Búsqueda, que fueron establecidas mediante la función *SetSearchBoxDimensions*.

Sintaxis

```
extern long GetSearchBoxDimensions(
long handle,
unsigned long *pSearchBoxWidth,
unsigned long *pSearchBoxHeight
);
```

Parámetros **handle**
Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función *OpenCompressor*.
pSearchBoxWidth, pSearchBoxHeight
Salida. El ancho y altura, en pixeles, de la región de búsqueda.

Valor de Retorno

- ISI_OK si la ejecución fue realizada con éxito.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO.
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.

Comentarios La mayor de estas regiones de búsqueda, generará la mejor calidad de la imagen de salida. De cualquier forma, una región de búsqueda grande también incrementará el tamaño del archivo, el tiempo de compresión y la cantidad de memoria necesaria para descomprimir a la imagen. Generalmente, la configuración por omisión debería ser utilizada por ser la de mejor eficiencia.

Valor por Default
320x200

Requisitos Previos
OpenCompressor.

Véase *SetSearchBoxDimensions*.

14. OpenCompressor

Compresor **Funciones de Inicio**

Abre una instancia del Compresor, y reinicia todos los parámetros internos a sus valores por omisión.

Sintaxis

```
extern long OpenCompressor(
long *phandle,
unsigned long CompressorType,
unsigned long NumBoardAddresses,
long *pBoardAddresses
);
```

Parámetros

phandle
Salida. Proporciona una referencia a esta instancia del Compresor, para que sea utilizada en todas las otras llamadas a funciones del Compresor.

CompressorType
Se refiere al tipo de hardware compresor. Ningún hardware compresor es soportado en esta versión o revisión de las librerías; establecer este valor a ISI_SW (es decir, sólo software y no hardware compresor).

NumBoardAddresses
Número de los compresores de tipo hardware a ser utilizados por la instancia. Este debe tener un valor de cero.

pBoardAddresses
El arreglo de direcciones base de las tarjetas compresoras que serán utilizadas por la instancia. Este valor debe ser nulo.

Valor de Retorno

- ISI_OK si la ejecución fue realizada con éxito.
- COMP_TYPE_ERROR si *CompressorType* No es válido.
- COMP_BUSY_ERROR si no se dejaron manejadores de instancia libres.
- ISI_GLOBAL_ALLOC_ERROR si no existe suficiente memoria para iniciar otra instancia del Compresor.

Comentarios Llamar a esta función antes de llamar a cualquier otra función del Compresor (excepto *GetCompVersion*). Después de que la Aplicación ha realizado toda la actividad de compresión, llamar a *CloseCompressor* para liberar todas las variables internas y la memoria.

Si el manejador de salida es cero, entonces *OpenCompressor* no será capaz de establecer una nueva instancia para el Compresor. El número máximo de manejadores permitidos en un mismo momento es de 256.

Requisitos Previos
Ninguno.

Véase *CloseCompressor.*

15. ResetCompressor

Compresor **Funciones de Inicio**

Reinicializa todos los parámetros internos para esta instancia del Compresor a sus valores por omisión.

Sintaxis
```
extern long ResetCompressor(
  long handle
);
```

Parámetros **handle**
Entrada. Identifica a la instancia que va a ser reinicializada, que es parámetro de salida de la función *OpenCompressor.*

Valor de Retorno
- ISI_OK si la ejecución fue realizada con éxito.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO.
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.

Comentarios Esta función reinicializa a los parámetros siguientes a los siguientes valores.
- Las dimensiones de la Caja de Búsqueda a 320x200 (véase *SetSearchBoxDimensions*).
- La rapidez de la Compresión a 1 (véase *SetCompressionSpeed*).
- El parámetro *CallbackFunc* al valor NULO (véase *SetCompressCallback*).
- La bandera Gris/Color a *FALSE* (véase *SetFIFFormat*).
- La resolución de la imagen Original a No inicializada (véase *SetInputResolution*).
- La Tabla de Color a NULO (véase *SetFIFColorTable*).

Requisitos Previos
OpenCompressor.

Véase *CloseCompressor*, las funciones Set___ listadas arriba.

16. SetCompressCallback

Compresor **Funciones de Implantación**

Establece una función de retorno de llamada para el Compresor.

Sintaxis

```
extern long SetCompressCallback(
 long handle,
 ISI_Callback CallbackFunc
);
```

Parámetros **handle**

Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función *OpenCompressor*.

CallbackFunc

Entrada. Es un apuntador a una función de retorno de llamada definida por el usuario.

Valor de Retorno

- ISI_OK si la ejecución fue realizada con éxito.
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO.

Comentarios Establece la función interna de retorno de llamada para el Compresor. El Compresor «retorna la llamada» a la función apuntada mediante *CallbackFunc* durante el procedimiento de compresión.

Si la función de retorno de llamada regresa un número diferente de cero, el procedimiento de compresión será abortado y *CompressToBuffer* regresará el valor ISI_CANCEL. De lo contrario, la compresión continuará.

La frecuencia del retorno de llamada se mantiene internamente a través del Compresor.

La función de retorno de llamada debería tener el siguiente formato :

- long UserCallBack(long handle, long PercentComplete)

Valor por Default

La función de retorno de llamada es NULA. Por tanto, ninguna función de retorno de llamada será cargada.

Requisitos Previos

OpenCompressor.

17. SetCompressionSpeed

Compresor **Funciones de Implantación**

Establece el factor de rapidez que será utilizado por el Compresor.

Sintaxis

```
extern long SetCompressionSpeed(
  long handle,
  long CompressionSpeed
);
```

Parámetros **handle**
Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función *OpenCompressor*.
CompressionSpeed
Entrada. Es el factor de rapidez de la compresión, y su valor debe estar entre 1 y 100.

Valor de Retorno

- ISI_OK si la ejecución fue realizada con éxito.
- COMP_SPEED_ERROR si el factor de rapidez no es un valor entero entre 1 y 100.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO.
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.

Comentarios La configuración varía desde 1 hasta 100, donde 100 representa a la compresión más rápida y 1 representa a la compresión más lenta Al incrementar la rapidez (la compresión será más rápida), el tamaño del archivo compactado también se incrementa.
La calidad se mantiene constante sin importar la configuración de la rapidez de compresión. (La Calidad de la compresión se establece mediante *CompressToBuffer*.)

Valor por Default
1 (para el archivo más pequeño posible, pero la compresión es la más lenta).

Requisitos Previos
OpenCompressor.

Véase *GetCompressionSpeed*.

18. SetFIFColorTable

Compresor **Funciones de Implantación**

Establece una tabla de color para que sea incluida dentro de la memoria de salida FIF.

Sintaxis

```
extern long SetFIFColorTable(
  long handle,
  unsigned long NumColors,
  unsigned char *pColorTable
);
```

Parámetros **handle**
Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función *OpenCompressor.*
NumColors
Entrada. Es el número de entrada a la Tabla de Color. Este número debe ser menor o igual que 256.
pColorTable
Entrada. Es la tabla de color que se va a almacenar en el archivo de salida FIF.

Valor de Retorno

- ISI_OK si la ejecución fue realizada con éxito.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO.
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.
- COMP_MAX_COLORS_EXCEEDED_ERROR si el número de colores es mayor a 256.

Comentarios Esta función carga una tabla de color del archivo FIF para utilizarla en la descompresión. La tabla de colores no es utilizada durante el proceso de compresión.

NOTA. La tabla de color debe estar en el mismo formato que los datos de la imagen de entrada, como se han especificado en *SetInputFormat.*

Valor por Default

NumColors = 0 , *pColorTable* = *NULL*, lo cual significa que ninguna tabla de color será incrustada en la memoria FIF.

Requisitos Previos

OpenCompressor, *SetInputFormat.*

19. SetFIFFormat

Compresor Funciones de Implantación

Estableceel formato del color del archivo de salida FIF.

Sintaxis

```
extern longSetFIFFormat (
 long handle,
 unsigned long GrayScale
);
```

Parámetros **handle**
Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función *Open Compressor*.
GrayScale
Entrada. Los probables valores son *TRUE* y *FALSE*. *TRUE* convierrte cualquier formato de color de entrada a escala de gris. *FALSE* no realiza la conversión del formato del color.

Valor de Retorno

- ISI_OK si la ejecución fue realizada con éxito.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO (NULL).
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.

Comentarios Por omisión, el formato de color del archivo FIF es el mismo que el formato del archivo de entrada *SetFIFFormat* es útil si se desea guardar la imagen de entrada de color en escala de grises (establecer *GrayScale* al valor *TRUE*).

Valor por Dafault
FALSE; el formato de salida es el mismo que el formato de entrada.

Requisitos Previos
OpenCompressor.

Veáse *GetFIFFormat.*

20. SetImageBuffer

Compresor Funciones de Implanatción

Establece un apuntador interno a la memoria que contiene a la imagen de entrada.

Sintaxis

```
extern long SetImageBuffer(
  long handle,
  unsigned char *pImageBuffer,
  unsigned long RowBytes
);
```

Parámetros

handle
Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función *OpenCompressor*.

pImageBuffer
Entrada. Es la memoria para la imagen de entrada.

RowBytes
Entrada. El número de bytes en un renglón, de la imagen de entrada. Este número debe incluir algún relleno.

Valor de Retorno

- ISI_OK si la ejecución fue realizada con éxito.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO (NULL).
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.
- COMP_IMAGE_SIZE_ERROR si no ha sido proporcionado el tamaño de la imagen.
- COMP_UNKNOWN_INCREMENT_ERROR si *GetNextImageRect* no ha sido llamada en el modo de compresión incremental.
- Error de Memoria, si esta se ha agotado.

Comentarios Llamar a esta función para especificar la memoria para la imagen de entrada para el Compresor. Si una memoria para la imagen ya ha sido especificada, primero llamar a *ClearImageBuffer* para inicializarla. También llamar a *ClearImageBuffer* cuando la Aplicación realice el trabajo con la memoria actual de la imagen.
En una compresión de tipo incremental, *pImageBuffer* es una memoria para la imagen de tipo incremental que contiene el siguiente segmento (incremento) de la imagen de entrada. En este caso, primero debe llamarse a *GetNextImageRect* y cargar el incremento apropiado a la memoria de tipo incremental antes de llamar a *SetImageBuffer*.
NOTA . La memoria que es pasada a *SetImageBuffer* debe mantenerse válida hasta que sea llamada la función *ClearImageBuffer*. No liberar o modificar a la memoria hasta después de llamar a *ClearImageBuffer*. Sólo la memoria para una sola imagen puede ser establecida en un momento dado mediante una instancia de compresión.

Requisitos Previos

OpenCompressor, *SetImageResolution*. Para una compresión de tipo incremental, también se requiere a *StartIncrementalCompression*, *GetNextImageRect*.

Véase *CompressToBuffer*, *ClearImageBuffer*.

21. SetImageResolution

Compresor **Funciones de Implantación**

Le indica al Compresor el tamaño de la imagen de entrada en pixeles, y opcionalmente en unidades físicas que pueden ser almacenadas en el archivo FIF y utilizadas por el Descompresor.

Sintaxis

```
extern long SetImageResolution(
long handle,
unsigned long PixelWidth,
unsigned long PixelHeight,
unsigned long PhysicalWidth,
unsigned long PhysicalHeight,
unsigned long PhysicalUnits,
long PhysicalUnitScaleFactor
);
```

Parámetros

handle
Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función *OpenCompressor*.

PixelWidth, PixelHeight
Entrada. Es el ancho y la altura en pixeles de la imagen original.

PhysicalWidth, PhysicalHeight
Entrada. El ancho y la altura de la imagen original en unidades físicas. Las unidades por omisión para el ancho y la altura están dadas en pixeles, si el ancho y la altura físicas no son especificadas.

PhysicalUnits
Entrada. Son las unidades físicas en las cuales se expresan tanto *PhysicalWidth* como *PhysicalHeight*. Estas unidades pueden ser :

- METER_UNITS
- INCH_UNITS
- UNKNOWN_UNITS

El valor por omisión es UNKNOWN_UNITS si es que las dimensiones físicas no son especificadas.

PhysicalUnitScaleFactor
Entrada. Es la potencia de 10 de las unidades físicas entre -128 y 127.

Valor de Retorno

- ISI_OK si la ejecución fue realizada con éxito.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO (NULL).
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.

- COMP_IMAGE_SIZE_ERROR si no ha sido proporcionado el tamaño de la imagen.
- COMP_IMAGE_ASPECT_RATIO_ERROR si la proporción entre PixelWidth y PixelHeight es diferente de PhysicalWidth y PhysicalHeight.

Comentarios Esta función debe ser llamada para que el Compresor conozca el tamaño de los datos de entrada.
La función *GetMaxFIFSize* utiliza esta información para estimar el tamaño de la memoria de salida.. Esta función no establece el tamaño o la resolución de los datos FIF (cuya resolución es independiente debido a que es un formato fractal).
NOTA. El Compresor puede cambiar el factor de escala de *PhysicalWidth* y *PhysicalHeight*, y hacer una correspondiente modificación al parámetro *PhysicalUnitScaleFactor*, si se requiere almacenar estos valores más eficientemente dentro del archivo FIF.
El establecimiento de las dimensiones físicas es opcional. Si se especifican las dimensiones físicas, se puede especificar cualquier dimensión, sin embargo para obtener mejores resultados, especificar las dimensiones reales de la imagen de entrada, si estas son conocidas, o proporcionar las dimensiones físicas en pixeles y especificar el valor UNKNOWN_UNITS. La proporción de aspecto (véase el Glosario) de las dimensiones físicas debe ser la misma que la proporción de aspecto de las dimensiones en pixeles.
Utilizar las funciones del Descompresor *GetOriginalResolution* y *GetPhysicalDimensions* para recuperar esta información desde el archivo FIF al momento de la descompresión.

Requisitos Previos
OpenCompressor.

Véase *GetImageResolution.*

22. SetInputFormat

Compresor **Funciones de Implantación**

Define la ordenación del color y el orden por renglones de los datos de la imagen de entrada.

Sintaxis
```
extern long SetInputFormat(
  long handle,
  long ColorPlane0,
  long ColorPlane1,
  long ColorPlane2,
  long ColorPlane3,
  long RowOrder
);
```

Parámetros **handle**

Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función *OpenCompressor*.

ColorPlane0, ColorPlane1, ColorPlane2, ColorPlane3

Entrada. Los valores para las superficies de color.

Las Constantes se definen como sigue:

- RED8
- GREEN8
- BLUE8
- NOT_USED (valor usado si se requieren menos de cuatro)
- BLANK8
- GRAY8

RowOrder

Entrada. Especifica el orden en el cual los datos de la memoria de entrada serán procesados. Establecer el parámetro *RowOrder* a TOP_LEFT o BOTTOM_LEFT para indicar dónde comienzan los datos de la imagen.

Valor de Retorno

- ISI_OK si la ejecución fue realizada con éxito.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO (NULL).
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.
- COMP_ILLEGAL_IMAGE_FORMAT_ERROR si el formato no es soportado.

Comentarios Utilizar los cuatro valores de color para especificar el orden en el cual serán almacenados los valores RGB dentro de la memoria para la imagen de entrada. Si se requieren menos de cuatro valores, los valores restantes deberían de ser llenados con el valor NOT_USED.

Por ejemplo, el formato DIB de Windows almacena los datos de mapa de bits (bitmap) en pixeles de tres bytes como sigue : Los primeros 8 bits son para el color Azul, los segundos 8 bits son para el color Verde, los terceros 8 bits son para el Rojo. Los datos del mapa de bits deberían comenzar en la parte inferior izquierda de la imagen. Esto resultaría de una llamada a *SetInputFormat* como sigue:

```
SetInputFormat(manejador, BLUE8, GREEN8, RED8, NOT_USED, BOTTOM_LEFT);
```

Son soportados varios formatos conocidos de imágenes.

Requisitos Previos

OpenCompressor.

Véase *GetInputFormat*.

23. SetSearchBoxDimensions

Compresor Funciones de Implantación

Establece el tamaño de la región en la cual el Compresor hace la búsqueda de fractales.

Sintaxis

```
extern long SetSearchBoxDimensions(
  long handle,
  unsigned long SearchBoxWidth,
  unsigned long SearchBoxHeight
);
```

Parámetros **handle**
Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función *OpenCompressor*.
SearchBoxWidth, SearchBoxHeight
Entrada. Es el ancho y la altura de la caja de búsqueda en pixeles.

Valor de Retorno

- ISI_OK si la ejecución fue realizada con éxito.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO (NULL).
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.
- COMP_SEARCH_WIDTH_ERROR si el ancho de la caja de búsqueda < 0.
- COMP_SEARCH_HEIGHT_ERROR si la altura de la caja de búsqueda < 0.

Comentarios La mayor de estas regiones de búsqueda, generará la mejor calidad de la imagen de salida. De cualquier forma, una región de búsqueda mayor también incrementará el tamaño del archivo, el tiempo de compresión y la cantidad de memoria necesaria para descomprimir la imagen. Generalmente, la configuración por omisión debería ser utilizada para obtener la mejor eficiencia.

Valor por Default
256x256

Requisitos Previos
OpenCompressor.

Véase *SetCompressionSpeed, CompressToBuffer*.

24. StartIncrementalCompression

Compresor **Funciones para Operaciones de tipo Incremental**

Prepara al Compresor para una compresión de tipo incremental.

Sintaxis

```
extern long StartIncrementalCompression(
  long handle,
  long *pNumPanels,
  long *pIncrementalFIFBufferSize
);
```

Parámetros

handle
Entrada. Identifica a la instancia del Compresor, que es parámetro de salida de la función *OpenCompressor*.

pNumPanels
Salida. Es el número de incrementos en los cuales la imagen de entrada será subdividida.

pIncrementalFIFBufferSize
Salida. Es el tamaño máximo de un panel compactado de la imagen. Utilizar este tamaño para establecer la memoria de tipo incremental para la salida.

Valor de Retorno

- ISI_OK si la ejecución fue realizada con éxito.
- ISI_NULL_ERROR si el manejador de la instancia tiene valor NULO (NULL).
- COMP_ILLEGAL_HANDLE_ERROR si el manejador de la instancia No es válido.
- COMP_NOT_INITIALIZED_ERROR si la instancia No es inicializada.
- COMP_IMAGE_SIZE_ERROR si no ha sido proporcionado el tamaño de la imagen.

Comentarios La compresión de tipo incremental procesa la imagen de entrada en forma de trozos, de tal forma que no sea necesario cargar la imagen completa en memoria. Esto puede ser muy útil cuando se comprimen imagen muy grandes.
Para utilizar la compresión de tipo incremental, la Aplicación que utilice estas librerías de Compresión debe llamar a *StartIncrementalCompression*, y entonces utilizar una estructura de ciclo para leer cada segmento de la imagen de entrada dentro de una memoria, utilizando llamadas a *GetNextImageRect*, *SetImageBuffer*, *CompressToBuffer* y *ClearImageBuffer*.

Requisitos Previos
OpenCompressor.

Véase *GetNextImageRect*, *CompressToBuffer*, *EndIncrementalCompression*.

Códigos de Error del Compresor

Identificador	Valor	Significado
COMP_BUSY_ERROR	-100	El Compresor ya está en uso.
COMP_NOT_INITIALIZED_ERROR	-101	Compresor NO inicializado.
COMP_ILLEGAL_HANDLE_ERROR	-102	Manejador de la instancia del Compresor NO válido.
COMP_ILLEGAL_IMAGE_FORMAT_ERROR	-120	El Formato de la imagen proporcionado a *SetInputFormat* es NO válido o NO soportado.
COMP_IMAGE_SIZE_ERROR	-121	El ancho o altura de la imagen es 0, o la función *SetInputResolution* no fue llamada antes de proceder a compactar una imagen.
COMP_IMAGE_ASPECT_RATIO_ERROR	-122	La razón de aspecto o apariencia física es diferente de la razón de aspecto en pixeles.
COMP_MAX_COLORS_EXCEEDED_ERROR	-123	La tabla de Colores tiene más de 256 colores.
COMP_RESET_ERROR	-132	El Compresor podría no ser reincializado a su estado por omisión.
COMP_TYPE_ERROR	-140	El tipo de Compresor pasado a *OpenCompressor* es NO válido.
COMP_NOT_INCREMENTAL_ERROR	-150	La función de compresión de tipo Incremental fue llamada sin una llamada previa a *StartIncrementalCompression*.
COMP_INCREMENTAL_COMPRESSION_COMPLETE	-151	Una llamada a *GetNextImageRect* fue realizada cuando ya no existen más paneles que comprimir.
COMP_UNKNOWN_INCREMENT_ERROR	-152	Se requiere de una llamada a *GetNextImageRect*.
COMP_SPEED_ERROR	-160	El Factor de rapidez NO es válido. El factor de rapidez es un entero entre 1 y 100.
COMP_QUALITY_ERROR	-161	Factor de Calidad NO válido. El factor de Calidad es un entero entre 1 y 100.
COMP_SEARCH_WIDTH_ERROR	-170	El ancho del Cuadro de Búsqueda es demasiado pequeño (<80).
COMP_SEARCH_HEIGHT_ERROR	-171	La altura del Cuadro de Búsqueda es demasiado pequeño (<64).
COMP_NO_IMAGE_BUFFER_ERROR	-190	Se requiere de una llamada a *SetImageBuffer*.

Códigos de Error de Memoria

IDENTIFICADOR	VALOR	SIGNIFICADO
ISI_GLOBAL_ALLOC_ERROR	-20	No fue capaz de colocar la memoria global.
ISI_GLOBAL_FREE_ERROR	-21	No fue capaz de liberar la memoria global.
ISI_LOCAL_ALLOC_ERROR	-30	No fue capaz de colocar la memoria local.
ISI_LOCAL_FREE_ERROR	-31	No fue capaz de liberar la memoria local.
ISI_NULL_ERROR	-40	Inesperado apuntador o manejador NULO.

Códigos de Estado de Error

IDENTIFICADOR	VALOR	SIGNIFICADO
ISI_OK	0	Operación exitosa.
ISI_CANCEL	-2	Operación cancelada por el usuario.

Decodificador FDK de Imágenes

****

Este es un Kit Fractal, para Desarrollares sobre Windows a 32 bits, de Iterated Systems, Inc.

CONTENIDO

25. SetFTTBuffer
26. SetOutputColorTable
27. SetOutputDither
28. SetOutputFilter
29. SetOutputFormat
30. SetOutputResolution
31. SetProgressiveFIFBuffer
32. SetProgressiveStep
33. StartProgressiveDecompression
34. TestIfFIF

Códigos de Error

Códigos de Estado de Error
Códigos de Error de Memoria
Códigos de Error del Descompresor

INICIO

¿ Qué es el "Descompresor" ? El Decodificador *FDK* de Imágenes también es conocido como *Descompresor*, debido a que éste convierte imágenes de tamaño pequeño FIF a sus correspondientes de tamaño mayor, basadas en pixeles.

Compatibilidad con otras versiones y productos

La Versión 1.1 de la interface para programadores del Decodificador *FDK* de Imágenes está diseñada para mantener compatibilidad con la versión 5.0 de *Hi-Res*. Las funciones en Hi-Res 5.0 son un subconjunto de las funciones de este producto. Las funciones de Hi-Res 5.0 tienen la misma sintaxis y salida que las funciones actuales de esta versión de este producto.

Las Aplicaciones escritas con las funciones de la Versión 3.0 o más recientes del Descompresor de Hi-Res no funcionarán con la versión 1.1 de las librerías DLL del FDK.

Los archivos FIF generados con cualquier versión liberada previa del Compresor Hi-Res, que incluyan a las versiones previas de Hi-Res, POEM ColorBox, POEM Profesional, e Imágenes Incorporadas, pueden ser descompactadas con la Versión 1.1 del Decodificador de Imágenes FDK. La Versión 1.1 soporta el uso de Plantillas o Patrones de la Transformada Fractal (*Fractal Transform Templates*, FTT) creados con Compresores recientes.

Cambios con respecto a *Hi-Res* Versión 5.0

La interface del programador para el Decodificador *FDK* de Imágenes Versión 1.1 incluye los siguientes cambios con respecto a Hi-Res 5.0.

Prueba del Formato FIF : La función *TestIfFIF* permite a la Aplicación determinar si un archivo es un Archivo FIF antes de intentar decodificarlo.

Descompresión Progresiva : El Decodificador *FDK* de Imágenes Versión 1.1 ahora soporta la descompresión progresiva, diseñada para usarla en aplicaciones OnLine (con línea activa de recepción de datos, por ejemplo una Aplicación Internet), donde un flujo de datos es recibido y decodificado paulatinamente.

Librerías y Archivos del Descompresor

Archivos de Inclusión

Incluir los siguientes archivos **.H** en el código de la Aplicación de descompresión.

- DECO_32.H contiene prototipos para las funciones de las librerías del Descompresor.
- ISIERROR.H define los valores de retorno de las funciones del Descompresor.

Librerías Windows

DECO_32.DLL es una DLL a 32 bits, la cual contiene al Descompresor.
Se incluyen tres librerías. Para cargar implícitamente al DLL Descompresor, se requiere hacer un enlace con la librería que ya ha sido probada para cada compilador con el que se esta haciendo el desarrollo de Aplicaciones :

- DECO_32W.LIB para Watcom 10.5a,
- DECO_32M.LIB para Microsoft Visual C++ 4.0, y
- DECO_32B.LIB para Borland C++ 4.5.

Inclusión del Descompresor en una Aplicación

1. Incluir DECO_32.H y ISIERROR.H en los módulos de los programas de Aplicación ya sea en C o C++, los cuales llamarán a las funciones del Descompresor proporcionadas en las librerías.

   Si se está desarrollando una Aplicación en un lenguaje diferente a C o C++, probablemente se requerirá crear archivos de inclusión equivalentes a DECO_32.H y ISIERROR.H, con las definiciones de constantes, definiciones tipo, y funciones prototipo en la forma apropiada para el lenguaje de programación que utilice.

2. Para cargar implícitamente a DECO_32.DLL, debe hacerse un enlace a la librería de importación apropiada DECO_32X.LIB dentro de la Aplicación, donde X indica el compilador respectivo, como se muestra en la lista de la sección anterior.

   Si se está usando un compilador diferente, es necesario revisar la documentación del compilador para localizar las instrucciones sobre cómo crear una librería de importación a partir de una librería DLL.
   Para cargar explícitamente a DECO_32.DLL (con la función *LoadLibrary*), no se requiere enlazarse con la librería de importación.

3. Usar las funciones del Descompresor como se explican a continuación.

Uso de las Funciones de Descompresión

Estos tópicos muestran en secuencia la forma en cómo pueden ser utilizadas las funciones del Decodificador de Imágenes FDK. Se muestran con un ejemplo sencillo. Sin embargo., el uso de estas funciones puede ser más complejo. Dependiendo de la utilización de las características de progresión y de mapeo del color del FDK, asegúrese de leer las secciones correspondientes, para obtener una descripción de cómo se utiliza cada función.

Inicialización del Descompresor

1. Comprobar la Versión de la Librería

Llamar a *GetDecoVersion* para asegurarse de que la Aplicación está utilizando las mismas DLLs del Descompresor para las cuales fue escrita. Esta versión de las DLLs es la 6.0. Una Aplicación que haya sido desarrollada con esta versión del Decodificador de Imágenes FDK será capaz de utilizar la Versión 6.0 o posterior de las DLLs.

2. Cargar la información en formato FIF dentro del Espacio de memoria

Cargar la parte completa de la información en formato FIF de entrada dentro de la memoria (el bufer de entrada). Si se está recibiendo sólo una parte de los datos de entrada a la vez (como por ejemplo, desde un módem), entonces probablemente se necesitará realizar una descompresión tipo progresiva. Si los datos en formato FIF se reciben completos en un sólo momento, por ejemplo desde un archivo en disco local, entonces se debería realizar una descompresión de tipo no progresiva.

3 Prueba del Formato del Archivo

Llamar a *TestIfFIF* para comprobar el formato de un archivo antes de que la Aplicación proceda a decodificarlo tomándolo como un archivo FIF.
Esta función puede ser llamada antes o después de que la instancia del Descompresor sea abierta con *OpenDecompressor*.

4 Apertura de una instancia del Descompresor

Usar *OpenDecompressor* para comenzar el proceso de descompresión. Una llamada a *OpenDecompressor* debe preceder a las llamadas de todas las restantes funciones del Descompresor (excepto *GetDecoVersion* y *TestIfFIF*).
OpenDecompressor inicializa una instancia del Descompresor, la cual es una sesión de uso del mismo.
OpenDecompressor retorna una manejador (**handle**) de la instancia, el cual es un entero único que identifica a esta instancia. Debe hacerse referencia a este manejador para utilizar cualquier otra función de descompresión (excepto *CheckDecoVersion* y *TestIfFIF*).
Por cada instancia, los parámetros que definen a las operaciones de descompresión pueden ser establecidos y leídos, y permanecerán en memoria hasta que estos sean modificados (con las funciones que contienen Parámetros de Salida), o hasta que se cierre la instancia (con *CloseDecompressor*).

Más de un espacio de memoria de entrada puede ser descompactado al utilizar una sola instancia, sin embargo sólo un espacio de memoria podrá ser descompactado a la vez. Para descompactar múltiples espacios de memoria simultáneamente deben abrirse múltiples instancias del Descompresor.

Instauración de Avance No Progresivo

1. Establecer la Función de Retorno de llamada (*Callback*)

Para permitir al usuario final cancelar la operación de descompresión, agregar una función propia de retorno de llamada a la Aplicación y especificarla al Descompresor con *SetDecompressCallback*. Esta función le permitirá a la Aplicación establecer un diálogo de monitoreo de situación o estado, y responder al comando "cancelar" a partir del cual el usuario final interrumpirá la descompresión.

2. Establecer la Memoria de Entrada FIF

Colocar un bloque de memoria de entrada y leer la información deseada del formato FIF. Llamar a *SetFIFBuffer* para especificar este bloque de memoria de entrada al Descompresor, y establecer los parámetros internos del Descompresor. Esta llamada es obligatoria.
La memoria en formato FIF debe mantenerse activa hasta que la descompresión finalice y se haya llamado a *ClearFIFBuffer*.

Instauración de Avance Progresivo

1. Establecer la Memoria de Entrada en Formato FIF y Leer el Encabezado

La descompresión progresiva trabaja sobre un flujo de datos mientras este es recibido. Dentro de una estructura de ciclo, llamar iterativamente a *SetProgressiveFIFBuffer* para indicarle al Descompresor dónde se encuentran los datos de entrada, y retornar a la Aplicación el estado de los datos FIF de la entrada.
Los parámetros de estado de *SetProgressiveFIFBuffer* retornan un constante que indica qué porciones del archivo ya han sido recibidas. Este valor de estado es importante, dado que otras llamadas a funciones sólo son apropiadas en ciertos puntos durante la recepción de los datos. En este punto de la Aplicación, la salida del ciclo de iteración se realizará cuando la función *SetProgressiveFIFBuffer* indique que el encabezado ha sido completamente recibido.

La memoria FIF debe mantenerse activa hasta que la descompresión haya finalizado y se haya llamado a *ClearFIFBuffer*.

Instauración de Mapeo de Color

1. Especificación de la Salida con Mapeo en Color

Llamar a *SetOutputFormat* con los siguientes parámetros para especificar, al Descompresor, el formato de mapeo del color.

SetOutputFormat (handle, COLORMAP8, NOT_USED, NOT_USED, NOT_USED, RowOrder);

Esta llamada es obligatoria si se utiliza mapeo de color, y es un requisito previo para todas las restantes funciones de mapeo de color.

2. Formato de la Tabla de Colores

Llamar a *SetColorTableFormat* para establecer el orden en que serán usados los colores en la tabla de colores. Esta función proporciona más flexibilidad para conformar la salida descompactada para algún formato de tabla de colores estándar.
Si una tabla de colores será leída desde un archivo de formato FIF (ver Lectura de la Tabla de Colores FIF, de la sección siguiente), entonces esta llamada le indicará al Descompresor cómo ordenar los colores cuando éste cargue la Tabla de Colores FIF en un bloque de memoria.
GetColorTableFormat lee la configuración del formato actual de la tabla de colores.

3. Lectura de la Tabla de Colores FIF

Una tabla de colores debe estar almacenada en el archivo FIF, si la aplicación del compresor almacenó una al momento de la compresión. Para averiguar si existe, use *GetFIFNumColors*. Si la función retorna cero como el número de colores, entonces no existe la tabla de colores almacenada en el archivo.
Si *GetFIFNumColors* retorna un número de colores diferente de cero, entonces coloca un espacio de memoria que mantendrá una tabla de colores esa cantidad de colores. Utilice *GetFIFColorTable* para cargar la Tabla de Colores FIF dentro de la memoria de la Tabla de Colores.

Si no se hace uso de la tabla de colores del archivo de entrada FIF, entonces es posible crear o cargar una tabla de colores a partir de otra fuente u origen.

4. Establecer la Tabla de Colores de Salida

Llamar a *SetOutputColorTable* para especificar la tabla de colores de salida, cómo utilizar los colores en ella y cómo cuantificar el número de colores que posee.
Cada entrada en la tabla de colores tiene una bandera correspondiente de información, la cual le indica al Descompresor dónde obtener el color y si es que lo va a utilizar. La bandera puede ser establecida con los siguientes valores.

- **Dinámica** : los colores son generados mediante el Descompresor a partir de la información de la escala de colores de 24 bits o la escala de grises de 8 bits en el archivo FIF. Este proceso crea la tabla de colores óptima para esta imagen, sin embargo esto hace lenta la descompresión.

- **Estática** : proporcionada a partir del archivo FIF o alguna otra fuente. Por ejemplo, la Aplicación debe tener una tabla de colores estándar para utilizarla para todas las imágenes.

- **No Utilizada** : El color en la tabla de colores no está en uso.

GetOutputColorTable lee la tabla de colores actual.

5. Establecer el *Dithering*[2]

Llamar a *SetOutputDither* para indicarle al Descompresor si es que se le aplicará o no la técnica *dithering* a la imagen de salida de colores mapeados. La técnica *Dithering* promedia los errores entre los valores de colores reales y los valores de colores mapeados, de tal forma que los colores se aprecien más exactos cuando se muestren en pantalla.
El *Dithering* es activado por omisión, y debería permanecer así bajo circunstancias normales. *GetOutputDither* lee el estado actual del *dithering* en el Descompresor.

Instauración sin Mapeo de Color

1. Establecer el Formato del Mapeo de Color

Llamar a *SetOutputFormat* para especificar el formato a presentar en la salida. Esta función puede ser llamada en cualquier momento para modificar el tipo de pantalla, sin embargo sólo necesita ser llamada una vez.
En los sistemas Macintosh, los siguientes argumentos son utilizados comúnmente :

SetOutputFormat (handle, BLANK8, RED8, GREEN8, BLUE8, TOP_LEFT);

Instauración Completa

1. Establecimiento del Archivo FTT

Llamar a *GetFIFFTTFileName* para determinar si se necesita de un archivo FTT. Si este es necesario, se debe leer el archivo FTT en bloques de memoria y debe llamarse a *SetFTTBuffer*. La memoria FTT debe mantenerse activa hasta que la descompresión finalice y se haya llamado a *ClearFTTBuffer*.

2. Establecer el Filtrado de la Imagen

La función *SetOutputFilter* le indica al Descompresor si la imagen de salida será filtrada o no. El Filtrado elimina la ocurrencia ocasional de escalonamiento (pixelado) de bordes que puedan aparecer en los archivos FIF, y que fueron creados con altas proporciones de compresión.
El Filtrado está activado por omisión, y debería permanecer así bajo circunstancias normales. *GetOutputFilter* lee el estado actual del filtrado en el Descompresor.

3. Obtener la Información del Tamaño y Resolución

Utilizar algunas de las siguientes funciones para determinar el tamaño de la imagen ya descompactada, dependiendo de las necesidades de la Aplicación.

- Llamar a *GetOriginalResolution* para obtener las dimensiones originales en pixeles de la imagen antes de que fuera compactada, si se conoce.
- Llamar a *GetPhysicalDimensions* para obtener las dimensiones reales, en pulgadas o metros, de la

imagen original, si es que esta información fue especificada al momento de la compresión. Si *GetPhysicalDimensions* retorna las dimensiones en unidades desconocidas, y no se conocen otras dimensiones, puede utilizarse la proporción de aspecto de estas dimensiones para calcular el tamaño de la pantalla.

- Si la rapidez de la descompresión es más importante que el tamaño, utilice *GetFastResolution* para obtener las dimensiones de salida para la descompresión más rápida. Primero, llamar a *SetOutputResolution*, debido a que *GetFastResolution* calcula la resolución más rápida basada en la resolución que actualmente está especificada.

4. Establecimiento de la Resolución de la Imagen de Salida

Una vez que el tamaño de salida ha sido seleccionado, llamar a *SetOutputResolution* para especificar las dimensiones en pixeles de la imagen de salida. Esta llamada es obligatoria, debido a que no hay resolución de salida por omisión.
(Nótese que al configurar la resolución rápida requiere que se llame a *SetOutputResolution*, después a *GetFastResolution*, y entonces a *SetOutputResolution* nuevamente.)
Colocar un espacio de memoria para la imagen de salida de acuerdo a estas dimensiones y la profundidad del color especificado mediante *SetOutputFormat*.

Realización de la Descompresión No Progresiva

Llamar a *DecompressToBuffer* para descompactar la información en formato FIF a partir de la memoria de entrada (que previamente ha sido especificada con *SetFIFBuffer*) dentro de la memoria de salida (la cual fue ya establecida después de llamar a *SetOutputResolution*).
Puede especificarse un área rectangular para descomprimir sólo una porción de la información que está en formato FIF.

Realización de la Descompresión Progresiva

1. Leer el Número de Pasos del formato FIF

Se tiene la opción de llamar a *GetFIFNumSteps* para averiguar cuántos pasos contiene el archivo FIF. Cada paso representa un nivel de detalle que está contenido en el archivo FIF. Dependiendo de la forma en cómo actualice la Aplicación la pantalla, se podrán observar un número de vistas preliminares de la imagen FIF durante la descompresión de tipo progresiva, hasta alcanzar el número de pasos.

2 Establecer el Número de Pasos

Se debe llamar a *SetProgressiveStep* para indicarle al Descompresor cuántos pasos desea descompactar. No es necesario conocer el número total de pasos de la información FIF. Las siguientes constantes pueden ser utilizadas para especificar algunos números de pasos con precisión.

DECO_STEP_LAST y DECO_STEP_FIRST descomprime el último y el primer paso, respectivamente.
DECO_STEP_ALL descomprime todos los pasos del archivo FIF.
DECO_STEP_BEST descomprime el último paso que ha sido recientemente recibido en forma completa.

3. Iniciar la Descompresión de tipo Progresiva

Colocar un espacio de memoria para la imagen de salida. Entonces, llamar a *StartProgressiveDecompression* una vez, para comenzar el proceso de descompresión. Estos parámetros de funciones son similares a *DecompressToBuffer*, la cual es utilizada para descompresión de tipo no progresiva. Usar esta función para especificar la memoria de salida, y para indicarle al Descompresor si se desea una descompresión de pequeña expansión, o una descompresión enfocada en un punto particular (centralizada).

4. Descompactación de la Imagen

De la forma como sea apropiado para la Aplicación, continuar cargando el flujo de datos de entrada hacia un bloque de memoria, utilizando la función *SetProgressiveFIFBuffer* para especificar el espacio de memoria y comprobar el estado.

En cualquier momento durante o después de la recepción de los datos de la imagen FIF (los cuales siguen después del encabezado del formato FIF), llamar a *ProgressiveDecompress* sucesivamente hasta que la descompresión se haya completado.
Esta función le permite especificar con qué frecuencia será actualizada la memoria de salida y el retorno de la función. Mediante su actualización, tan frecuentemente como sea posible (con la constante FREQUENCY_HIGH), se pueden implementar retornos de llamadas mediante la comprobación de la entrada del teclado.

ProgressiveDecompress también la indica a la Aplicación si esta necesita más datos de entrada, o si se ha finalizado con la descompresión.

5. Fin de la Descompresión de tipo Progresiva

Cuando la descompresión se ha completado, llamar a *EndProgressiveDecompression* para liberar los parámetros internos que fueron reservados por el Descompresor especialmente para una descompresión progresiva.

Inicialización después de la Descompresión

1. Limpiar los bloques de Memoria

Llamar a *ClearFIFBuffer* (y a *ClearFTTBuffer* si es necesario) para terminar la descompresión de los datos en el (los) bloque(s) de memoria de entrada actualmente especificado(s). Después de llamar a estas funciones, la Aplicación podrá liberar los bloques de memoria de entrada sin causar algún error.
No liberar la memoria hasta después de llamar a la función correspondiente para limpiar los bloques de memoria (*ClearFIFBuffer* y *ClearFTTBuffer*).

2. Cerrar la Instancia del Descompresor

Para retirar esta instancia del Descompresor, llamar a *CloseDecompressor*. Esta función libera los bloques internos de memoria del Descompresor y libera al manejador. Si la Aplicación inicializa más de una instancia en un momento dado, se ahorrará memoria al cerrar las instancias que no estén en uso.

3. Descompactar otro Archivo FIF

DecompressToBuffer y *ProgressiveDecompress* no reinicializan a ningún parámetro del Descompresor. Puede mantenerse la configuración actual para la siguiente tarea de descompresión, o reinicializar los valores con las funciones apropiadas que comiencen con *Set*—. Para comprobar la configuración actual, utilice alguna de las funciones *Get*—.
Para realizar descompresiones adicionales sobre los mismos datos FIF de entrada (por ejemplo, al utilizar dimensiones diferentes o áreas de rectángulo diferentes para descompactarlas), simplemente reincialice los parámetros y llame a las funciones de descompresión nuevamente.

INDICE DE FUNCIONES DE DESCOMPRESIÓN

1. ClearFIFBuffer

Descompresor **Funciones de Limpieza-Reinicio**

Reinicia todas las variables FIF especificas.

Sintaxis extern long ClearFIFBuffer(
long handle
);

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor de la memoria FIF a la que ha de aplicarse una limpieza (reinicialización), y es parámetro de salida de la función *OpenDecompressor*.

Valor de Retorno
- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).

Comentarios Llamar a esta función para reinicializar a la memoria FIF que previamente haya sido establecida mediante la llamada a *SetFIFBuffer*.
Debe llamarse a *ClearFIFBuffer* antes de llamar a *SetFIFBuffer* nuevamente, o antes de llamar a *CloseDecompressor*.

Requisitos Previos
OpenDecompressor, *SetFIFBuffer*.

2. ClearFTTBuffer

Descompresor **Funciones de Limpieza-Reinicio**

Reinicia a todas las variables FTT especificas.

Sintaxis extern long ClearFTTBuffer(
long handle
);

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor de la memoria FTT a la que ha de aplicarse la limpieza (reinicialización), y es parámetro de salida de la función *OpenDecompressor*.

Valor de Retorno
- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.

- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).

Comentarios Llamar a esta función para reiniciar la memoria FTT que previamente haya sido establecida mediante la llamada a *SetFTTBuffer*.
Debe llamarse a *ClearFTTBuffer* antes de llamar a *SetFTTBuffer* nuevamente, o antes de llamar a *CloseDecompressor*.

Requisitos Previos
OpenDecompressor, *SetFTTBuffer*.

Véase *CloseDecompressor*.

---

3. CloseDecompressor

Descompresor **Funciones de Limpieza-Reinicio**

Cierra una instancia de la descompresión. Libera los parámetros internos utilizados por el Descompresor el cual fue establecido e inicializado mediante *OpenDecompressor*.

Sintaxis extern long CloseDecompressor(
long handle
);

Parámetros **handle**
Entrada. Identifica la instancia que ha de ser eliminada, que es parámetro de salida de la función *OpenDecompressor*.

Valor de Retorno
- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).

Comentarios Esta función debería ser llamada después de que la Aplicación haya completado todo el proceso de descompresión. Todos los bloques de memoria deben ser establecidos, tal como las memorias FIF y FTT, y deben ser inicializadas mediante el uso de *ClearFIFBuffer* y *ClearFTTBuffer* antes de llamar a esta función.

Requisitos Previos
OpenDecompressor, *ClearFIFBuffer* y *ClearFTTBuffer* si las memorias fueron ya establecidas.

---

4. DecompressToBuffer

Descompresor **Funciones No Progresivas**

La información FIF descompactada se mantiene en la memoria FIF (al utilizar los datos de la memoria FTT si fuera necesario) y almacena la imagen descompactada dentro de la memoria de salida.

Sintaxis extern long DecompressToBuffer(
long handle,
unsigned char *pImageBuffer,
long CropPixelX,
long CropPixelY,
long CropPixelWidth,
long CropPixelHeight,
long BytesPerRow
);

Parámetros

handle
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.

pImageBuffer
Entrada. Es un apuntador a la memoria de salida previamente establecida, y que contendrá los datos de la imagen descompactada.

CropPixelX, CropPixelY
Entrada. Para reproducir la imagen a partir de la imagen FIF de entrada, especificar las coordenadas x, y del primer pixel de la imagen que ha de ser la salida, o las coordenadas 0,0 si no se basará en la imagen de entrada. Estos valores deben ser proporcionados en parejas de números, pues si no la función retornará el valor DECO_CROP_RECT_ERROR (-56).

CropPixelWidth, CropPixelHeight
Entrada. Para reproducir la imagen FIF, especificar al ancho y la altura, en pixeles, de la imagen FIF que ha de ser la salida, o 0, 0 si no se basará en la imagen FIF.

BytesPerRow
Entrada. Es el número de bytes en un renglón, de la imagen de salida. Este valor debe tomar en cuenta cualquier pixel o renglón de relleno.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_CROP_RECT_ERROR (-56) si la reproducción del rectángulo es ilegal.
- ISI_CANCEL (-2) si el usuario ha cancelado la descompresión.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- DECO_FTT_NOT_PROVIDED_ERROR (-60) si se requiere de memoria FTT pero no ha sido proporcionada.
- DECO_FIF_NOT_PROVIDED_ERROR (-66) si una memoria FIF no ha sido proporcionada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).
- Error de Memoria, si esta se ha agotado.
- Error del Descompresor, si ocurre un error durante la descompresión.

Comentarios *OpenDecompressor*, *SetFIFBuffer* y *SetOutputFormat* deben ser llamadas para abrir el Descompresor e inicializar los parámetros internos antes de llamar a esta función. Pueden ser llamadas otras funciones si se desea modificar los valores por omisión de cualquiera de los parámetros internos.
Utilizar *DecompressToBuffer* para especificar un área de reproducción de la imagen, si se desea. Para el caso de una descompresión de tipo progresiva, no utilizar la función *DecompressToBuffer*; en su lugar véase *StartProgressiveDecompression*.

Requisitos Previos
OpenDecompressor, *SetFIFBuffer*, *SetOutputFormat*, *SetOutputResolution*.

Véase *SetOutputFilter*, *GetFIFFTTFileName*, *SetFTTBuffer*, *GetPhysicalDimensions*, *StartProgressiveDecompression*.

5. EndProgressiveDecompression

Descompresor **Funciones Progresivas**

Finaliza un proceso de descompresión de tipo progresivo. Libera los parámetros internos utilizados por el Descompresor los cuales fueron establecidos e inicializados mediante *StartProgressiveDecompression*.

Sintaxis extern long EndProgressiveDecompression(
long handle
);

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.

Valor de Retorno
- ISI_OK (0) si la ejecución fue realizada con éxito.

Comentarios Llamar a *EndProgressiveDecompression* inmediatamente después de la ejecución de la estructura de ciclo que esté precedida por *StartProgressiveDecompression*.

Requisitos Previos
OpenDecompressor, *StartProgressiveDecompression*.

Véase *CloseDecompressor*.

6. GetColorTableFormat

Descompresor **Funciones de mapeo de Color**

Obtiene los valores actuales de los parámetros internos de formato de la tabla de colores.

Sintaxis extern long GetColorTableFormat(
long handle,
long *pColorPlane0,
long *pColorPlane1,
long *pColorPlane2,
long *pColorPlane3
);

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
pColorPlane0, pColorPlane1, pColorPlane2, pColorPlane3
Salida. Son los valores para las superficies de color. Las Constantes se definen como sigue :
- RED8
- GREEN8
- BLUE8

- NOT_USED (si se requieren menos de cuatro)
- BLANK8
- GRAY8

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).

Comentarios El formato de la tabla de color previamente debe ser establecida mediante *SetColorTableFormat*, si no se regresará un error. El mapeo del Color es habilitado con una llamada a *SetOutputFormat*, que configura el formato de mapeo del color.

Requisitos Previos

OpenDecompressor, *SetOutputFormat* al especificar el formato del mapeo del color.

Véase *SetColorTableFormat*.

---

7. GetDecoVersion

Descompresor **Funciones de Inicio**

Obtiene el número de versión actual de la librería del Descompresor.

Sintaxis extern long GetDecoVersion(
long *pPrimaryVersion,
long *pSecondaryVersion
);

Parámetros **pPrimaryVersion**
Salida. Es el número de versión primario. El número de versión primario de esta DLL es el 6.
pSecondaryVersion
Salida. Es el número de versión secundario (por ej. Número de revisión). El número de versión secundario de esta DLL es 0.

Valor de Retorno

ISI_OK (0) si la ejecución fue realizada con éxito.

Comentarios Utilizar esta función para verificar la versión de la librería DECO_32.DLL, actualmente instalada. La Aplicación que utilice esta librería debería comprobar el número de versión de la DLL para asegurarse que va a trabajar con la misma versión o una versión más reciente de la DLL con la cual la Aplicación fue desarrollada. La librería DLL se este paquete de desarrollo es la versión 6.0.

Requisitos Previos

Ninguno.

Véase *TestIfFIF*.

8. GetFastResolution

Descompresor **Funciones de Implantación**

Retorna la resolución más grande posible, para la salida; que sea menor o igual a la resolución de salida actual, lo cual resultará en una descompresión más rápida.

Sintaxis extern long GetFastResolution(
long handle,
long *pPixelWidth,
long *pPixelHeight
);

Parámetros

handle
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.

pPixelWidth
Salida. Es el Ancho para la imagen de salida, en pixeles, para tener una rapidez óptima.

pPixelHeight
Salida. Es la altura de la imagen de salida, en pixeles, para tener una rapidez óptima.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).

Comentarios El Descompresor funciona más rápidamente a ciertas resoluciones de salida. Esta función retornará la resolución que producirá una mayor rapidez que sea lo más cercana, pero no mayor que la resolución actual (establecida con *SetOutputResolution*). Debe llamarse a *SetOutputResolution* para establecer la resolución actual antes de llamar a *GetFastResolution*. Entonces, si se desea se puede utilizar la resolución para mayor rapidez, al llamar a *SetOutputResolution* nuevamente para establecerla como la actual. NOTA. El Descompresor funcionará a cualquier resolución. Esta función es sólo para las Aplicaciones cuyo punto principal sea la rapidez de la descompresión, y que produzcan a la salida las imágenes a un tamaño muy pequeño (si así se necesita), para lograr una rápida descompresión.

Requisitos Previos
OpenDecompressor, *SetFIFBuffer*, *SetOutputResolution*.

Véase *GetOriginalResolution*.

9. GetFIFColorTable

Descompresor **Funciones de mapeo de Color**

Obtiene la tabla interna de color desde la memoria FIF actual.

Sintaxis extern long GetFIFColorTable(
long handle,

```
    unsigned char *pColorTable
);
```

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
pColorTable
Salida. La tabla de color dentro de la memoria FIF.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).
- DECO_NO_COLORMAP_ERROR (-63) si no está disponible la tabla de color.

Comentarios Primero, llamar a *GetFIFNumColors* para investigar si es que la tabla de colores está almacenada en el archivo FIF, donde dicha función ha de retornar el número de colores de la tabla. Debe establecerse una memoria para la tabla de color y definir el formato de salida de la misma (mediante *SetColorTableFormat*) antes de llamar a esta función. Después de esta llamada, debe llamarse a *SetOutputColorTable* si se desea utilizar la tabla de color.

Requisitos Previos
OpenDecompressor, *SetFIFBuffer*, *SetOutputFormat* para especificar el formato del mapeo del color, *SetColorTableFormat*, *GetFIFNumColors*.

Véase *SetOutputColorTable*.

---

10. GetFIFFTTFileName

Descompresor **Funciones de Implantación**

Obtiene el nombre del archivo FTT (si existe) asociado con los datos almacenados en la memoria FIF.

Sintaxis
```
extern long GetFIFFTTFileName(
    long handle,
    unsigned char *pFTTFileName
);
```

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
pFTTFileName
Salida. Es el nombre del archivo FTT. Retorna una cadena de longitud cero si no existe un archivo FTT asociado con los datos FIF.
NOTA. Esta cadena debe ser de al menos 256 caracteres de longitud.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).

Comentarios Si se retorna un nombre de archivo válido, debe procederse a leer el archivo FTT y llamar a *SetFTTBuffer* antes de intentar descompactar la imagen. Un nombre de archivo de longitud cero indica que no existe archivo FTT asociado con los datos FIF.

Requisitos Previos
OpenDecompressor, *SetFIFBuffer*.

Véase *SetFTTBuffer*.

---

11. GetFIFNumColors

Descompresor **Funciones de mapeo de Color**

Obtiene el número de colores almacenados en la tabla interna de colores dentro de la memoria FIF actual.

Sintaxis extern long GetFIFNumColors(
long handle,
long *pNumColors
);

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
pNumColors
Salida. Es el número de colores de la tabla de colores FIF, o es cero si no hay tabla de colores almacenada en la memoria FIF.

Valor de Retorno
- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).

Comentarios Si no hay tabla de colores almacenada en la memoria FIF, *pNumColors* será cero. Si *pNumColors* tiene valor diferente de cero, utilizar *GetFIFColorTable* para recuperar la tabla de colores de la memoria FIF.

Requisitos Previos
OpenDecompressor, *SetFIFBuffer*.

Véase *GetFIFColorTable*, *SetOutputColorTable*.

---

12. GetFIFNumSteps

Descompresor **Funciones únicamente Progresivas**

Retorna el número de pasos o intervalos en un archivo FIF.

Sintaxis extern long GetFIFNumSteps(
long handle,
long *pNumSteps
);

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
pNumSteps
Salida. Es el número de pasos dentro del archivo FIF.

Valor de Retorno
- ISI_OK (0) si la ejecución fue realizada con éxito.

Comentarios Los archivos FIF que han sido creados con la Versión 6.0 o posterior del Compresor están organizados en múltiples pasos que permiten la descompresión progresiva. Los archivos FIF más recientes siempre retornarán 1 como el número de pasos. Si *pNumSteps* = 1, entonces una descompresión de tipo progresiva generará los mismos resultados que una descompresión estándar.

Requisitos Previos
OpenDecompressor.

Véase *SetProgressiveStep*.

13. GetOriginalResolution

Descompresor **Funciones de Implantación**

Obtiene las dimensiones originales en pixeles (antes de la compresión) de la imagen que está almacenada en la memoria de datos FIF.

Sintaxis extern long GetOriginalResolution(
long handle,
long *pPixelWidth,
long *pPixelHeight,
long *pImageDepth
);

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
pPixelWidth, pPixelHeight
Salida. Son Apuntadores al ancho y altura, en pixeles, de la imagen original.
pImageDepth
Salida. Es un Apuntador a la profundidad, dada en bits, de la imagen original. Generalmente, una profundidad de 8 indica una imagen en escala de grises, y una profundidad de 24 indica una imagen a color.

Valor de Retorno
- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.

- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).

Comentarios Los valores de ancho y altura en pixeles pueden ser utilizados para establecer los tamaños de memoria interna, y para escalar la imagen mediante *SetOutputResolution*. Si no hay dimensiones en pixeles almacenadas en el archivo FIF, *GetOriginalResolution* retornará cero tanto para el ancho como para la altura. La profundidad de la imagen siempre será retornada. En este caso, obtener las dimensiones físicas mediante *GetPhysicalDimensions*.

Requisitos Previos
OpenDecompressor, *SetFIFBuffer*.

Véase *GetPhysicalDimensions*, *SetOutputResolution*.

---

14. GetOutputColorTable

Descompresor **Funciones de mapeo de Color**

Obtiene la tabla de colores actual para el Descompresor.

Sintaxis extern long GetOutputColorTable(
long handle,
unsigned char *pColorTable
);

Parámetros
handle
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
pColorTable
Salida. La tabla de colores actual para el Descompresor.

Valor de Retorno
- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).

Comentarios La tabla de colores previamente debe haber sido establecida mediante *SetOutputColorTable*, o generada dinámicamente mediante *DecompressToBuffer*, si el mapeo de color dinámico fuera especificado mediante *SetOutputColorTable*. Si se desea guardar una tabla de color generada dinámicamente, asegúrese de llamar a esta función antes de descompactar otra imagen. La tabla de color dinámica cambiará por cada imagen.

Requisitos Previos
OpenDecompressor, *SetOutputFormat* par especificar el formato del mapeo del color, *SetColorTableFormat*, *SetOutputColorTable*.

Véase *GetFIFColorTable*.

15. GetOutputDither

Descompresor **Funciones de mapeo de Color**

Obtiene el valor actual del parámetro interno para la técnica *dithering* del Descompresor.

Sintaxis extern long GetOutputDither(
long handle,
unsigned char *pDitherFlag
);

Parámetros **handle**
Salida. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
pDitherFlag
Salida. Es un apuntador a una variable que contendrá el parámetro para el *dithering*. Tendrá valor *TRUE* si el *dithering* ha sido habilitado, y *FALSE* si este ha sido deshabilitado.

Valor de Retorno
- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO.

Comentarios El *Dithering* está habilitado por omisión (*pDitherFlag* = *TRUE*), y debería ser aplicado bajo circunstancias de operación normales. La bandera para el *dithering* es ignorada si el mapeo del color no está habilitado mediante *SetOutputFormat*.

Valor por Default
TRUE

Requisitos Previos
OpenDecompressor.

Véase *SetOutputDither*.

16. GetOutputFilter

Descompresor **Funciones de Implantación**

Obtiene el valor actual del parámetro interno de filtro para el Descompresor.

Sintaxis extern long GetOutputFilter(
long handle,
unsigned char *pFilterFlag
);

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función

OpenDecompressor.

pFilterFlag

Salida. Es un apuntador a una variable que contendrá el parámetro para el filtro. Tendrá valor *TRUE* si el filtrado está habilitado, y *FALSE* si está deshabilitado.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).

Comentarios El filtrado elimina la ocurrencia ocasional de bordes "escalados" o "pixelados" que pueden aparecer en los archivos FIF que fueron generados a altas proporciones de compresión. El filtrado está habilitado por omisión (*pFilterFlag* = *TRUE*), y debería ser utilizado bajo circunstancias normales.

Valor por Default

TRUE

Requisitos Previos

OpenDecompressor.

Véase *SetOutputFilter*.

---

17. GetOutputFormat

Descompresor **Funciones de Implantación**

Obtiene los valores actuales de los parámetros de formato de salida para el Descompresor.

Sintaxis extern long GetOutputFormat(
long handle,
long *pColorPlane0,
long *pColorPlane1,
long *pColorPlane2,
long *pColorPlane3,
long *pRowOrder
);

Parámetros **handle**

Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.

pColorPlane0, pColorPlane1, pColorPlane2, pColorPlane3

Salida. Son los valores para las superficies de color. Las Constantes se definen como sigue:

- RED8
- GREEN8
- BLUE8
- NOT_USED (si se requieren menos de cuatro)
- BLANK8
- COLORMAP8
- GRAY8

pRowOrder

Salida. Indica en qué orden serán procesados los datos en la memoria de salida. Tomar el valor TOP_LEFT

o BOTTOM_LEFT pata indicar dónde comienza los datos de la imagen.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).

Comentarios Esta función puede ser llamada en cualquier momento para determinar el estado actual de los parámetros de formato de salida. Llamar a *SetOutputFormat* para realizar modificaciones a estos parámetros.

Requisitos Previos

OpenDecompressor.

Véase *SetOutputFormat*.

---

18. GetOutputResolution

Descompresor **Funciones de Implantación**

Obtiene los valores actuales de los parámetros para la resolución de salida del Descompresor.

Sintaxis extern long GetOutputResolution(
long handle,
long *pPixelWidth,
long *pPixelHeight
);

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
pPixelWidth, pPixelHeight
Salida. Es el ancho y la altura de la imagen de salida, en pixeles.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).

Comentarios Esta función puede ser llamada en cualquier momento para comprobar los valores actuales para la resolución de salida.

Requisitos Previos

OpenDecompressor.

Véase *SetOutputResolution*.

19. GetPhysicalDimensions

Descompresor **Funciones de Implantación**

Obtiene las dimensiones físicas originales (antes de la compresión) de la imagen en el archivo FIF.

Sintaxis extern long GetPhysicalDimensions(
long handle,
long *pPhysicalWidth,
long *pPhysicalHeight,
signed char *pPhysicalUnits,
long *pPhysicalUnitScaleFactor
);

Parámetros

handle
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.

pPhysicalWidth, pPhysicalHeight
Salida. Es el ancho y altura originales de la imagen, en unidades especificadas mediante *pPhysicalUnits*.

pPhysicalUnits
Salida. Son las unidades físicas en las cuales *pPhysicalWidth* y *pPhysicalHeight* se expresarán. Estas unidades pueden ser

- METER_UNITS
- INCH_UNITS
- UNKNOWN_UNITS

pPhysicalUnitScaleFactor
Salida. Potencia de 10 de las medidas de ancho y altura, entre -128 y 127.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).

Comentarios Tanto los valores físicos de ancho y altura como la proporción de aspecto podrán ser utilizados para establecer los tamaños de la memoria interna, y para escalar la imagen mediante *SetOutputResolution*. Las unidades de salida son una unidad de tamaño físico, ya sea METER_UNITS, INCH_UNITS o UNKNOWN_UNITS. El ancho y altura de salida son expresados en dichas unidades. Si las unidades son UNKNOWN_UNITS, entonces la proporción de aspecto podrá ser utilizada para calcular una resolución de salida para la imagen a descompactar.

Requisitos Previos
OpenDecompressor, SetFIFBuffer.

Véase *GetOriginalResolution, SetOutputResolution.*

20. OpenDecompressor

Descompresor **Funciones de Inicio**

Inicializa los parámetros internos para el Descompresor a sus valores por omisión y retorna un manejador de la instancia.

Sintaxis extern long OpenDecompressor(
long *phandle
);

Parámetros **phandle**
Salida. Proporciona una referencia a esta instancia del Descompresor, para que sea utilizada en todas las demás llamadas a las funciones del Descompresor.

Valor de Retorno
- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_BUSY_ERROR (-51) si no hay disponibles manejadores para la instancia.
- ISI_GLOBAL_ALLOC_ERROR (-20) si está saturada la memoria.

Comentarios Llamar a esta función antes de llamar a cualquier otra función del Descompresor (excepto *GetDecoVersion*). Después de que la Aplicación haya realizado todo el proceso de descompresión, llamar a *CloseDecompressor* para liberar todas las variables internas y la memoria. Si el manejador para la salida es cero, entonces *OpenDecompressor* no fue capaz de establecer una nueva instancia para el Descompresor. El número máximo de manejadores permitidos en un mismo momento es 256.

Requisitos Previos
Ninguno.

Véase *CloseDecompressor*.

21. ProgressiveDecompress

Descompresor **Funciones Progresivas**

Realiza una descompresión tipo progresiva.

Sintaxis extern long ProgressiveDecompress(
long handle,
long frequency,
long *pOutputBufferX,
long *pOutputBufferY,
long *pOutputBufferWidth,
long *pOutputBufferHeight,
long *pBytesNeeded,
long *pPercentDone
);

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función

OpenDecompressor.

frequency

Entrada. Controla la frecuencia en que la memoria de salida ha de ser actualizada, de acuerdo a la cantidad procesada en cada paso. Puede ser alguna de las siguientes constantes:

- FREQUENCY_NONE para generar, a la salida, la imagen completa cuando sea haya completado cada paso.
- FREQUENCY_LOW para actualizar la salida por cada renglón de paneles.
- FREQUENCY MEDIUM para actualizar la salida por cada panel.
- FREQUENCY_HIGH para actualizar la salida tan pronto como sea posible, siempre que no hayan datos adicionales que descompactar. Utilícese esta valor para implementar retornos de llamadas, o si la Aplicación debe realizar un mantenimiento total frecuentemente, al descompactar.

pOutputBufferX, pOutputBufferY

Salida. Son las coordenadas x, y, en pixeles, de la esquina superior izquierda de la región que ha sido actualizada en la memoria de salida.

pOutputBufferWidth, pOutputBufferHeight

Salida. Es el ancho y la altura, en pixeles, de la región que ha sido actualizada dentro de la memoria de salida. Se retornan ceros si la memoria de salida no ha sido actualizada.

pBytesNeeded

Salida. Reporta el número de bytes mas que son necesarios antes de que la siguiente llamada a *ProgressiveDecompress* realice más trabajo. Si tiene el valor 0, entonces la siguiente llamada puede realizar más trabajo sin datos adicionales desde la memoria de entrada. Si tiene el valor -1, entonces la descompresión ha sido completada. Si contiene un valor >0, entonces debe llamarse a *SetProgressiveFIFBuffer* antes de llamar a *ProgressiveDecompress* nuevamente.

pPercentDone

Salida. Es el porcentaje realizado de la tarea de descompresión completa.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.

Comentarios Esta función realiza una descompresión progresiva, al transferir los datos de la imagen descompactada hacia la memoria de salida que fue especificada con *StartProgressiveDecompression*. La función debe ser llamada en una estructura de ciclo que haga un monitoreo de la recepción de los datos de la memoria de entrada. El argumento *pBytesNeeded* reporta el número de bytes adicionales que son necesarios en la memoria de entrada antes de que la siguiente llamada a *ProgressiveDecompress* pueda realizarse. *ProgressiveDecompress* retorna el área de la memoria de salida que ha sido actualizada. En el caso de una descompresión de pequeña expansión (especificado con la función *StartProgressiveDecompression*), será retornada el área apropiada en el centro de la memoria de salida.

Requisitos Previos

OpenDecompressor, SetProgressiveFIFBuffer, StartProgressiveDecompression.

---

22. SetColorTableFormat

Descompresor **Funciones de mapeo de Color**

Define el orden de los colores de la tabla de colores para la descompresión.

Sintaxis extern long SetColorTableFormat(
long handle,
long ColorPlane0,
long ColorPlane1,

```
    long ColorPlane2,
    long ColorPlane3
);
```

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
ColorPlane0, ColorPlane1, ColorPlane2, ColorPlane3
Entrada. Son los valores para las cuatro superficies. Las constantes se definen como sigue:

- RED8
- GREEN8
- BLUE8
- NOT_USED (usado si se requieren menos de cuatro)
- BLANK8

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).

Comentarios Utilizar esta función para establecer el orden del rojo, verde, azul y otros componentes de la tabla de colores para la descompresión. Esta función debe ser llamada antes de proceder a la descompresión con salida de mapeo de colores. El mapeo de colores primero debe ser habilitado con una llamada a *SetOutputFormat*.

Requisitos Previos
OpenDecompressor, *SetOutputFormat* para especificar el formato del mapeo del color.

Véase *GetColorTableFormat*.

---

23. SetDecompressCallback

Descompresor **Funciones de Implantación**

Establece una función de retorno de llamada para el Descompresor.

Sintaxis extern long SetDecompressCallback(

```
    long handle,
    ISI_Callback CallbackFunc,
    long CallbackFreq
);
```

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
CallbackFunc
Entrada. Es un apuntador a una función de retorno de llamada especificada por el usuario.
CallbackFreq
Entrada. Especifica qué tan frecuentemente el Descompresor retornará la llamada. Las opciones válidas son:

- CALLBACK_FREQ_NONE
- CALLBACK_FREQ_LOW
- CALLBACK_FREQ_HIGH

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).
- DECO_CALLBACK_ERROR (-64) si la función de retorno de llamada falla o el valor especificado en *CallbackFreq* es inválido.

Comentarios Establece la función interna de retorno de llamada para el Descompresor. El Descompresor «retorna la llamada « a la función apuntada mediante *CallbackFunc* durante el procedimiento de descompresión. No utilizar *SetDecompressCallback* en un descompresión de tipo progresiva. En su lugar, simular los retornos de llamada con *StartProgressiveDecompression*. Si la función de retorno de llamada retorna cualquier valor diferente de cero, el procedimiento de descompresión será abortado y retornará el valor ISI_CANCEL (-2) mediante *DecompressToBuffer*. De otro modo, la descompresión continuará. La función de retorno de llamada debería tener el siguiente formato:

- long UserCallBack(long handle, long PercentComplete)

Bajo Windows NT, la frecuencia del retorno de llamada (*CallbackFreq*) debería establecerse a bajo nivel, debido a que otras aplicaciones pueden ejecutarse independientemente de la función de retorno de llamada. Bajo Windows y Macintosh, la frecuencia debería ser alta, dado que la función de retorno de llamada es utilizada tanto para permitir al usuario cancelar la operación, y para permitir a otras aplicaciones ejecutarse durante la descompresión.

Valor por Default

La función de retorno de llamada es NULA. Ninguna función de retorno de llamada es cargada.

Requisitos Previos

OpenDecompressor.

Véase *DecompressToBuffer*.

---

24. SetFIFBuffer

Descompresor **Funciones No Progresivas**

Establece un apuntador interno a una memoria que contiene el archivo de datos FIF completo.

Sintaxis extern long SetFIFBuffer(
long handle,
unsigned char *pFIFBuffer,
long FIFBufferSize
);

Parámetros

handle
Entrada. Identifica a la instancia, que es parámetro de salida de la función *OpenDecompressor*.
pFIFBuffer
Entrada. Es un apuntador a una memoria FIF.
FIFBufferSize
Entrada. El tamaño de la memoria FIF, proporcionada para asegurar que el Descompresor no lea más allá

del final de la memoria.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- DECO_FIF_UNSUPPORTED_ERROR (-62) si el tipo de archivo no es soportado.
- DECO_FIF_FUTURE_VERSION_ERROR (-69) si los archivos FIF requieren de un Descompresor.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).
- Error de Memoria, si esta se ha agotado.
- Error del Descompresor si ocurre un error en el archivo FIF.

Comentarios Llamar a esta función para especificar la memoria FIF de entrada para el Descompresor. Si una memoria FIF ya fue especificada, primero llamar a *ClearFIFBuffer* para inicializarla. También llamar a *ClearFIFBuffer* cuando la Aplicación esté realizando el trabajo con la memoria FIF actual. No utilizar a *SetFIFBuffer* en una descompresión de tipo progresiva. Utilizar en su lugar a *SetProgressiveFIFBuffer*. NOTA. La memoria que es pasada a *SetFIFBuffer* debe permanecer en estado válido hasta que sea llamada *ClearFIFBuffer*. No liberar o modificar la memoria hasta después de llamar a *ClearFIFBuffer*. Sólo una memoria FIF individual puede ser establecida en un momento dado para una instancia de descompresión dada.

Requisitos Previos
OpenDecompressor. La Aplicación debería tener ya reservada la memoria y la lectura del archivo FIF.

Véase *ClearFIFBuffer*, *SetProgressiveFIFBuffer*.

---

25. SetFTTBuffer

Descompresor **Funciones de Implantación**

Establece un apuntador interno a una memoria que contiene al archivo de datos FTT completo.

Sintaxis
```
extern long SetFTTBuffer(
    long handle,
    unsigned char *pFTTBuffer,
    long FTTBufferSize
);
```

Parámetros

handle
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.

pFTTBuffer
Entrada. Es un apuntador a la memoria FTT.

FTTBufferSize
Entrada. Es el tamaño de la memoria FTT, proporcionada para asegurar que el Descompresor no lea más allá del final de la memoria.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).
- Error de Memoria, si esta se ha agotado.

- Error del Descompresor si un error ocurre en el archivo FTT.

Comentarios Llamar a esta función para especificar la memoria FTT de entrada para el Descompresor. Si una memoria FTT ya ha sido especificada, primero llamar a *ClearFTTBuffer* para inicializarla. También llamar a *ClearFTTBuffer* cuando la Aplicación haya realizado el trabajo con la memoria FTT actual.
NOTA. La memoria que es pasada a *SetFTTBuffer* debe ser válida hasta que *ClearFTTBuffer* sea llamada. No liberar o modificar la memoria hasta después de llamar a *ClearFTTBuffer*. Sólo una memoria FTT individual puede ser establecida en un momento dado para una instancia de descompresión dada. De cualquier forma, la memoria FTT puede ser utilizada para más de un archivo FIF.

Requisitos Previos
OpenDecompressor. La Aplicación debería establecer la memoria y la lectura del archivo FTT.

Véase *ClearFTTBuffer*.

---

26. SetOutputColorTable

Descompresor **Funciones de mapeo de Color**

Le indica al Descompresor dónde localizar la tabla de colores para que sea utilizada durante la descompresión.

Sintaxis
```
extern long SetOutputColorTable(
    long handle,
    unsigned char *pColorTable,
    unsigned char *pEntryInfo,
    long NumColors
);
```

Parámetros

handle
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.

pColorTable
Entrada. Es un apuntador a una tabla de colores. Si el valor retornado es NULO, entonces se asume que la tabla de colores es completamente dinámica, y será creada mediante el Descompresor.

pEntryInfo
Entrada. Es una memoria que contiene información para cada color dentro de la tabla de colores. Los valores probables son:

- CM_NOT_USED: El color no va a ser utilizado.
- CM_STATIC: El color va a ser utilizado.
- CM_DYNAMIC: El color será creado y utilizado. Si *pEntryinfo* = NULL, entonces se asume que la tabla de colores será completamente estática. Utilizar a *GetOutputColorTable* para obtener una nueva tabla de colores después de llamar a *DecompressToBuffer*, especialmente si existen algunas entradas dinámicas.

NumColors
Entrada. El tamaño de la tabla de colores que va a ser utilizada.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO, o si tanto *pColorTable* como *pEntryInfo* son NULOS.

Comentarios Antes de llamar a esta función debe llamarse a *SetOutputFormat* para habilitar el mapeo del color, así como a *SetColorTableFormat* para especificar el orden de cada color en la tabla de colores. Utilizar el parámetro *pEntryInfo* para especificar cómo se genera cada color en la tabla. Una opción es permitir al Descompresor generar la tabla de colores, utilizando la información de color a 24 bits que está almacenada en el archivo FIF. Después de esta llamada, se está en la posibilidad de llamar a *DecompressToBuffer* para realizar la descompresión con la tabla de colores.

Requisitos Previos
OpenDecompressor, *SetFIFBuffer*, *SetOutputFormat* para especificar el formato del mapeo del color; y *SetColorTableFormat*.

Véase *GetOutputColorTable*.

---

27. SetOutputDither

Descompresor **Funciones de mapeo de Color**

Establece el parámetro para el método *dithering* interno para el Descompresor.

Sintaxis extern long SetOutputDither(
long handle,
unsigned char DitherFlag
);

Parámetros
handle
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
DitherFlag
Entrada. La bandera para el método *dithering*; debe contener el valor *TRUE* para habilitarlo o *FALSE* para deshabilitarlo.

Valor de Retorno
- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).

Comentarios El método *Dithering* está activado por omisión (*DitherFlag* = *TRUE*), y debería dejarse en dicho valor bajo circunstancias normales de operación. La bandera para el método *dithering* es ignorada si el mapeo del color no está habilitado mediante *SetOutputFormat*.

Valor por Default
TRUE

Requisitos Previos
OpenDecompressor, *SetOutputFormat* para especificar el formato del mapeo del color.

Véase *GetOutputDither*.

28. SetOutputFilter

Descompresor **Funciones de Implantación**

Establece el parámetro interno de filtrado para el Descompresor.

Sintaxis extern long SetOutputFilter(
long handle,
unsigned char FilterFlag
);

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
FilterFlag
Entrada. Es la bandera para habilitar el filtrado; debe contener el valor *TRUE* para habilitar al filtrado o *FALSE* para deshabilitarlo.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).

Comentarios El filtrado elimina la ocurrencia ocasional de bordes "escalados" o "pixelados" que pueden aparecer en los archivos FIF que fueron compactados a altas proporciones de compresión. El filtrado está habilitado por omisión (*FilterFlag* = *TRUE*), y debería ser utilizado bajo circunstancias normales.

Valor por Default
TRUE

Requisitos Previos
OpenDecompressor.

Véase *GetOutputFilter*, *DecompressToBuffer*.

29. SetOutputFormat

Descompresor **Funciones de Implantación**

Define el orden del color y de los renglones para los datos descompactados.

Sintaxis extern long SetOutputFormat(
long handle,
long ColorPlane0,
long ColorPlane1,
long ColorPlane2,
long ColorPlane3,
long RowOrder
);

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.

ColorPlane0, ColorPlane1, ColorPlane2, ColorPlane3
Entrada. Son los valores para las superficies del color. Las constantes se definen como sigue:

- NOT_USED (utilizar este valor si se requieren menos de cuatro)
- BLANK8
- COLORMAP8
- RED5
- GREEN5
- BLUE5
- GRAY8
- NOT_USED (utilizar este valor si se requieren menos de cuatro)
- BLANK8
- COLORMAP8
- RED5
- GREEN5
- BLUE5
- GRAY8

RowOrder
Entrada. Especifica el orden en el cual los datos en la memoria de salida serán procesados. Establecer en *RowOrder* el valor TOP_LEFT o BOTTOM_LEFT para indicar dónde comienzan los datos de la imagen.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).
- DECO_OUTPUT_FORMAT_ERROR (-57) si el formato especificado no es soportado.

Comentarios Esta función debe ser llamada antes de que *DecompressToBuffer* sea utilizada. De otra forma, el Descompresor no tendrá conocimiento del orden en el cual la información del color se genera a la salida. Utilizar los valores de los cuatro colores para especificar el orden en el cual los valores RGB sean almacenados en la memoria de la imagen de salida. Si se requieren menos de cuatro, los valores restantes deberían ser llenados con el valor NOT_USED. Por ejemplo, el formato DIB de Windows requiere que los datos de mapa de bits (bitmap) sean almacenados en pixeles de tres bytes como sigue: Los primeros 8 bits son Azul, los segundos 8 bits son Verdes, y los terceros 8 bits son Rojos. Los datos del mapa de bits deberían comenzar en la parte inferior izquierda de la imagen. Esto podría ocurrir con una llamada a *SetOutputFormat* como sigue:

```
SetOutputFormat(handle, BLUE8, GREEN8, RED8, NOT_USED, BOTTOM_LEFT);
```

Para preparar al Descompresor para el mapeo del color, llamar a *SetOutputFormat* como sigue:

```
SetOutputFormat(handle, COLORMAP8, NOT_USED, NOT_USED, NOT_USED,
RowOrder );
```

Otros formatos válidos son:

- RED8, GREEN8 y BLUE8 en cualquier orden más NOT_USED, y *RowOrder*
- RED5, GREEN5, BLUE5, NOT_USED, RowOrder
- GRAYSCALE8, NOT_USED, NOT_USED, NOT_USED, RowOrder

Requisitos Previos
OpenDecompressor.

Véase *GetOutputFormat*, *DecompressToBuffer*.

30. SetOutputResolution

Descompresor **Funciones de Implantación**

Establece los parámetros de la resolución de salida para el Descompresor.

Sintaxis extern long SetOutputResolution(
long handle,
long PixelWidth,
long PixelHeight
);

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
PixelWidth, PixelHeight
Entrada. Es el ancho y la altura de la imagen de salida, en pixeles.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).
- DECO_RESOLUTION_SIZE_ERROR (-58) si la resolución de salida es no válida.

Comentarios Esta función puede ser llamada en cualquier momento antes de que la descompresión sea realizada para establecer los valores de la resolución de salida.

Requisitos Previos
OpenDecompressor.

Véase *GetOutputResolution, GetOriginalResolution, GetPhysicalDimensions, GetFastResolution.*

31. SetProgressiveFIFBuffer

Descompresor **Funciones únicamente Progresivas**

Establecer un apuntador interno a la memoria que contiene los datos FIF de entrada durante un proceso de descompresión de tipo progresivo.

Sintaxis extern long SetProgressiveFIFBuffer(
long handle,
unsigned char *pFIFBuffer,
long FIFBufferSize,
long *pInputStatus,
long *pHeaderLength
);

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
pFIFBuffer
Entrada. Es un apuntador a la memoria FIF.

FIFBufferSize
Entrada. Es el tamaño de la memoria FIF, proporcionada para asegurar que el Descompresor no lea más allá del final de la memoria.
pInputStatus
Salida. Reporta la última sección de los datos FIF que han sido recibidos en *pFIFBuffer*.

- FIF_BEGIN: no han sido recibidos datos.
- FIF_HEADER: ha sido recibida una parte o todo el encabezado, pero no los datos de la imagen.
- FIF_DATA: Todo el encabezado y algo de los datos de la imagen han sido recibidos.
- FIF_COMPLETE: El archivo FIF completo ha sido recibido.

pHeaderLength
Si *pInputStatus* tiene el valor FIF_HEADER, entonces *pHeaderLength* indica el número de bytes adicionales necesarios para completar el encabezado. Si el valor es 0, entonces ninguna cantidad de datos del encabezado ha sido recibida para determinar el tamaño del encabezado.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- DECO_FIF_UNSUPPORTED_ERROR (-62) si el tipo de archivo no es soportado.
- DECO_FUTURE_FIF_VERSION_ERROR si los archivos FIF requieren un Descompresor actualizado.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO.
- Error de Memoria, si esta se ha agotado.
- Error del Descompresor si ha ocurrido un error en el archivo FIF.

Comentarios Llamar a esta función en una estructura de ciclo para especificar la memoria FIF de entrada, su tamaño, y el estado de los datos recibidos por el Descompresor. Esta memoria siempre debe contener los datos FIF completos recibidos. Dado que el tamaño y la ubicación de esta memoria puede cambiar durante la descompresión de tipo progresiva, la Aplicación puede hacer uso de *SetProgressiveFIFBuffer* para especificar una memoria diferente durante el proceso. Para una descompresión de tipo No progresiva, utilizar en su lugar a *SetFIFBuffer*. Las llamadas sucesivas a *SetProgressiveFIFBuffer* para modificar la memoria FIF de entrada, se realizan sin intervención de llamadas a *ClearFIFBuffer*. Llamar a *ClearFIFBuffer* para limpiar (reiniciar) la memoria de entrada actual sin reconstruir una nueva.
NOTA. La memoria que es pasada a *SetProgressiveFIFBuffer* debe permanecer en estado válido hasta que *ClearFIFBuffer* sea llamada, o sea especificada una memoria diferente con otra llamada a *SetProgressiveFIFBuffer*. De otra forma, no liberar o modificar la memoria establecida. Sólo la memo ria FIF individual puede ser establecida en un momento dado para una instancia de descompresión dada.

Requisitos Previos
OpenDecompressor. La Aplicación debería establecer la memoria y la lectura de los datos FIF.

Véase *ClearFIFBuffer*, *SetFIFBuffer*.

---

32. SetProgressiveStep

Descompresor **Funciones Progresivas**

Le indica al Descompresor qué pasos de un archivo FIF decodificado progresivamente ha de decodificar.
Sintaxis extern long SetProgressiveStep(
long handle,
long StepNumber);

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
StepNumber
El número de pasos que habrá que descompactar. Puede ser un entero positivo que denote al número de pasos, o una de las siguientes constantes.

- DECO_STEP_LAST decodifica sólo el último paso. Dado que la descompresión es acumulativa, esta opción produce la más alta resolución de salida sin mostrar intermitencia.
- DECO_STEP_FIRST decodifica sólo el primer paso. En un FIF multipasos, este valor resultará en la resolución más baja de salida.
- DECO_STEP_ALL decodifica todos los datos de la forma en que fueron recibidos hasta que la imagen entera esté completa.
- DECO_STEP_BEST decodifica el último paso de los datos hasta aquí recibidos. Esta opción produce la resolución más alta posible así como los datos sean recibidos.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.

Comentarios El parámetro *StepNumber* puede ser cualquier número desde cero hasta el número total de pasos en las datos FIF, menos uno. Obtener el número total al llamar a *GetFIFNumSteps*. *StepNumber* también puede tomar cualquiera de los valores constantes descritos bajo el parámetro *StepNumber*.

Requisitos Previos
OpenDecompressor, *SetProgressiveFIFBuffer*. Llamar a *GetFIFNumSteps* para determinar el número máximo de pasos progresivos.

---

33. StartProgressiveDecompression

Descompresor **Funciones Progresivas**

Comienza el proceso de descompresión de tipo progresivo.

Sintaxis
```
extern long StartProgressiveDecompression(
    long handle,
    unsigned char *pImageBuffer,
    long CropPixelX,
    long CropPixelY,
    long CropPixelWidth,
    long CropPixelHeight,
    long BytesPerRow,
    long Expand
);
```

Parámetros **handle**
Entrada. Identifica a la instancia del Descompresor, que es parámetro de salida de la función *OpenDecompressor*.
pImageBuffer
Entrada. Es un apuntador a una memoria de salida establecida previamente que contendrá los datos de la imagen descompactada.
CropPixelX, CropPixelY
Entrada. Para reproducir la imagen FIF de entrada, especificar las coordenadas x, y del primer pixel de la imagen FIF que ha de ser la salida, ó 0, 0 para no basarse en la imagen de entrada. Estos valores siempre deben ser números, o la función retornará el valor DECO_CROP_RECT_ERROR (-56).

CropPixelWidth, CropPixelHeight
Entrada. Para reproducir la imagen FIF, especificar el ancho y la altura, en pixeles, de la imagen FIF que ha de ser la salida, ó 0, 0 para no basarse en la imagen de entrada.
BytesPerRow
Entrada. Es el número de bytes en un renglón de la imagen de salida. Este valor debe tomar en cuenta cualquier pixel o renglón de relleno.
Expand
Entrada. Especifica el valor *TRUE* para una descompresión de pequeña expansión, o *FALSE* para una descompresión seleccionada.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_CROP_RECT_ERROR (-56) si el rectángulo de reproducción es ilegal.
- DECO_ILLEGAL_HANDLE_ERROR (-52) si el manejador de la instancia No es válido.
- DECO_NOT_INITIALIZED_ERROR (-50) si la instancia No es inicializada.
- DECO_FTT_NOT_PROVIDED_ERROR (-60) · si se requiere de memoria FTT pero no ha sido proporcionada.
- DECO_FIF_NOT_PROVIDED_ERROR (-66) si una memoria FIF no ha sido proporcionada.
- ISI_NULL_ERROR (-40) si el manejador de la instancia tiene valor NULO (NULL).
- Error de Memoria, si esta se ha agotado.
- Error del Descompresor, si ocurre un error durante la descompresión.

Comentarios Llamar a esta función sólo una vez, precediendo una estructura de ciclo que haga el monitoreo de los datos recibidos (con la función *SetProgressiveFIFBuffer*) y realizar una descompresión de tipo progresiva (mediante *ProgressiveDecompress*). Utilizar a la función *StartProgressiveDecompression* para especificar un área de reproducción, si se desea.

Requisitos Previos
OpenDecompressor, SetProgressiveFIFBuffer, SetProgressiveStep.

Véase *DecompressToBuffer, EndProgressiveDecompression, ProgressiveDecompress.*

---

34. TestIfFIF

Descompresor **Funciones de Inicio**

Comprueba los datos de un archivo, para ver si el archivo es un archivo FIF.

Sintaxis extern long TestIfFIF(
unsigned char *pFileBuffer,
long BufferLength,
unsigned char *pIsFIF
);

Parámetros
pFileBuffer
Entrada. Especifica la memoria donde los datos serán cargados. Esta memoria debe ser establecida y debe tener los datos del comienzo del archivo que haya sido cargado dentro de ella.
BufferLength
Entrada. Es la longitud de *pFileBuffer*. La longitud de la memoria debe ser de al menos 4 bytes para la prueba.
pIsFIF
Salida. Contendrá el valor *TRUE* si los datos son datos FIF; o *FALSE* en el caso contrario.

Valor de Retorno

- ISI_OK (0) si la ejecución fue realizada con éxito.
- DECO_FIF_FUTURE_VERSION_ERROR (-69) si los archivos FIF requieren un Descompresor actualizado.

Comentarios La memoria *pFileBuffer*, debe contener al menos 4 bytes a partir del principio del archivo a ser probado. La función retornará el valor *TRUE* si el archivo es un archivo FIF, aún cuando sea de una versión no soportada. Comprobar el error DECO_FUTURE_FIF_VERSION_ERROR para el soporte de la versión del archivo FIF.

Requisitos Previos

Ninguno.

Códigos de Error

Códigos de Estado de Error

IDENTIFICADOR	VALOR	SIGNIFICADO
ISI_OK	0	Operación exitosa.
ISI_CANCEL	-2	Operación cancelada por el usuario.

Códigos de Error de Memoria

IDENTIFICADOR	VALOR	SIGNIFICADO
ISI_GLOBAL_ALLOC_ERROR	-20	No fue capaz de colocar la memoria global.
ISI_GLOBAL_FREE_ERROR	-21	No fue capaz de liberar la memoria global.
ISI_LOCAL_ALLOC_ERROR	-30	No fue capaz de colocar la memoria local.
ISI_LOCAL_FREE_ERROR	-31	No fue capaz de liberar la memoria local.
ISI_NULL_ERROR	-40	Inesperado apuntador o manejador NULO.

Códigos de Error del Descompresor

IDENTIFICADOR	VALOR	SIGNIFICADO
DECO_NOT_INITIALIZED_ERROR	-50	Descompresor NO inicializado.
DECO_BUSY_ERROR	-51	El Descompresor está ocupado, No puede ser inicializado.
DECO_ILLEGAL_HANDLE_ERROR	-52	El manejador de la instancia del Descompresor No es válido.
DECO_FIF_BUFFER_INCOMPLETE	-53	El tamaño del bloque de memoria FIF es demasiado pequeño.
DECO_FTT_BUFFER_INCOMPLETE	-54	El tamaño del bloque de memoria FTT es demasiado pequeño.
DECO_FIF_FORMAT_ERROR	-55	La entrada está en un formato de archivo no conocido.
DECO_CROP_RECT_ERROR	-56	Error en el área rectangular señalada para la descompresión.
DECO_OUTPUT_FORMAT_ERROR	-57	El formato de Salida NO es válido.
DECO_RESOLUTION_SIZE_ERROR	-58	La resolución de Salido NO es válida.
DECO_FTT_FORMAT_ERROR	-59	El archivo FTT está en un formato NO conocido o NO soportado.
DECO_FTT_NOT_PROVIDED_ERROR	-60	El archivo FTT es necesario, pero no ha sido proporcionado.
DECO_FTT_INCORRECT_ERROR	-61	El Archivo FTT es erróneo para este archivo FIF.
DECO_FIF_UNSUPPORTED_ERROR	-62	La entrada está en un formato de archivo NO soportado.
DECO_NO_COLORMAP_ERROR	-63	La tabla de colores no fue proporcionada para el mapeo del color.
DECO_CALLBACK_ERROR	-64	Ha fallado la función de retorno de llamada
DECO_RESOLUTION_NOT_PROVIDED_ERROR	-65	No ha sido proporcionada la resolución de salida.
DECO_FIF_NOT_PROVIDED_ERROR	-66	Se requiere de una llamada a *SetFIFBuffer*.
DECO_INVALID_STEP_ERROR	-67	El número de pasos para el tipo de descompresión progresiva NO es válido.
DECO_NO_START_PROGRESSIVE_ERROR	-68	Debe hacerse una llamada primero a *StartProgressiveDecompression*.
DECO_FIF_FUTURE_VERSION_ERROR	-69	El archivo FIF de entrada requiere una versión más reciente del Descompresor.

[1] Bufer (del inglés : *Buffer*), es el espacio de memoria temporal para hacer las operaciones intermedias, que al final proporcionarán los resultados finales. Se utilizará indistintamente bufer o espacio de memoria. Para mayor referencia refiérase al Glosario.

[2] Ver Glosario.

GLOSARIO

imágenes digitalizados. Un simple codec puede incluir las funciones de conversión A/D (Analógico/ Digital) y D/A (Digital/Analógico), además de la compresión y descompresión. (2) (***COder/DECoder***, Codificador/Decodificador) Un circuito electrónico que convierte audio y video en código digital (y viceversa), utilizando técnicas tales como modulación por codificación de pulsos y modulación delta. Un codec es un convertidor tanto A/D como D/A. Este término es aplicable también al software que posee características funcionales similares.

10. Codificar. Es el proceso de crear un archivo de Formato de Imagen Fractal (FIF) a partir de una imagen basada en pixeles. La consecuencia de esta codificación es la Compresión.

11. Código Fractal. Un conjunto de datos que describen una ubicación de dominio, una ubicación de rango, y la transformación respectiva desde el rango hacia el dominio.

12. Código Reentrante. Es una rutina de programación que puede ser utilizada por múltiples programas simultáneamente. Esta es utilizada en sistemas operativos y otros sistemas de software incluyendo la ejecución multihebra, donde toman lugar los eventos concurrentes. Está escrita de tal forma que nada de su código sea modificable (ningún valor es modificado) y no mantiene datos de ningún tipo. Los programas que la llaman mantienen su propia información del progreso de la rutina (variables, banderas, etc.), de esta manera puede ser compartida una copia de la rutina reentrante a través de un número cualquiera de usuarios o procesos.

13. Colores Reales, Imagen de. Imágenes que requieren y manejan tres valores por pixel, es decir la profundidad de la Imagen : 24 bits/pixel por el método de definición de colores RGB: RED-GREEN-BLUE (8 bits ROJO + 8 bits VERDE + 8 bits AZUL = 24 bits).

14. Compilador. (1) Software que translada un lenguaje de programación de alto nivel (por ejemplo, COBOL, C, pascal, etc.) en lenguaje máquina. Un compilador por lo general produce primero lenguaje ensamblador y luego lo traslada en lenguaje máquina. El siguiente ejemplo compila las declaraciones de un programa, a lenguaje máquina :

Código Fuente	Lenguaje Ensamblador	Lenguaje Máquina
IF COUNT=10	Compara A con B	Compara 3477 con 2883
GOTO DONE	Si son iguales ir a C	Si = ir a 23883
ELSE	Ir a D	Ir a 23343
GOTO AGAIN		
ENDIF		

Código real de máquina
10010101001010001010100
10101010010101001001010
10100101010001010010010

(2) Software que convierte una representación de lenguaje de alto nivel a bajo nivel. Por ejemplo, un compilador de ayuda convierte un documento de texto, que incorpora comandos apropiados, en un sistema de ayuda en línea. Un compilador de diccionario convierte términos y deficiones en un sistema de diccionario de búsqueda.

Este Glosario consta de términos que han sido utilizados en este Trabajo de Tesis, incluyendo terminología con definiciones propias dentro del mismo contexto de la Teoría de Compresión Fractal.

1. **Aspect Ratio (Proporción o Razón de Apariencia).** Es la proporción establecida entre el ancho y la altura de un objeto. Para este caso se habla de la proporción de aspecto existente en las imágenes mostradas a través de la pantalla de un monitor.

2. **Atractor (Punto Fijo).** El punto al cual converge una función contractiva.

3. **API (*Application Program Interface*, Interface para Programas de Aplicación).** Es un formato de lenguaje y mensajes utilizado por un programa de Aplicación para comunicarse con otro programa que le proporciona servicios. Las APIs generalmente se implementan mediante la escritura de llamadas a funciones. Ejemplos de APIs son las llamadas realizadas por un programa de Aplicación hacia tales programas como son los Sistemas Operativos, los Sistemas de Mensajes o Sistemas de Administración de Bases de Datos (DBMS).

4. **Biblioteca de Clases.** Es un conjunto aislado de clases diseñadas para ser incorporadas en cualquier programa.

5. **Bit.** (***BI**nary Digi**T***, Dígito Binario*)* Es un simple dígito en el sistema binario (un 1 o un 0). En una computadora, un bit es físicamente un transistor o capacitor dentro de una celda de memoria, una marca magnética en disco o cinta, o un nivel de voltaje alto o bajo que circula a través de un circuito. Es una cantidad, la cual toma un valor de uno o cero. Un bit es como el estado de un foco eléctrico : o está apagado o está encendido.
   Los grupos de bits conforman unidades de almacenamiento llamadas caracteres, bytes, y palabras, las cuales son manipuladas como grupos. El grupo más común es el byte, el cual se compone de ocho bits y equivale a un símbolo o caracter alfanumérico.

6. **Brillo.** Respuesta del ojo a cada longitud de onda e intensidad dada de la imagen observada. Es el nivel de luz sobre la pantalla del monitor.

7. ***Buffer.*** Véase Memoria Temporal.

8. **Byte.** Conjunto de ocho bits.

9. **CoDec.** (1) (***COmpression/DECompression***, Compresión/Descompresión) Un circuito de hardware o una rutina de software (software codec) utilizado para comprimir o descomprimir audio, video o

15. **Compresión.** Una técnica que reduce la representación o tamaño de la información, pero no de la información en sí misma.

16. **Compresión Fractal.** Una clase de algoritmos que describen los datos en un modo compacto, al representarlos en términos de sus semejanzas afines en una imagen o entre imágenes.

17. **Compresión *Lossless*.** Método de compresión en el cual no hay pérdida de datos dentro del proceso de compresión , y los datos originales puedan ser recuperados idénticamente bajo un proceso de descompresión. Es decir, son métodos de Compresión para Uso General que reintegran la información Idéntica a la Original, por tanto es apto para Compresión de Datos (en su definición rigurosa) y Textos, principalmente.

18. **Compresión *Lossy*.** Método de Compresión en el cual algunos datos (usualmente aquellos redundantes), se pierden durante el proceso de compresión, datos que no podrán ya recuperarse.. Este método logra grandes índices de compresión (índices mayores que con los métodos Lossless), pero la información reintegrada NO es Idéntica a la Original, sin embargo, los Sentidos del Ser Humano no son capaces de percibir estas diferencias (generalmente), por tanto es apto para Compresión de Imágenes para Páginas WEB, Dibujos en general, Sonidos, etc.

19. **Contractividad.** La característica de una transformación afín (o cualquier otra función), para reducir el valor o tamaño en todas las dimensiones.

20. **Contraste.** Diferencia entre los valores de luminancia de dos objetos contiguos.

21. **Convergencia.** El proceso de aproximación de un función contractiva a su atractor. Una imagen fractal es una función contractiva la cual converge a su imagen fija (atractor).

22. **Decodificar.** El proceso de convertir un archivo de Formato de Imagen Fractal (FIF) a una imagen basada en pixeles para propósitos de impresión o visualización en pantalla.

23. **Descompresión.** El proceso inverso a la compresión, de forma de que los datos puedan ser visualizados o impresos.

24. **Digitalizar.** Es convertir una imagen en un medio impreso, a una imagen digital.

25. ***Dithering.*** (1) Técnica para simular colores realmente no existentes en un área de pixeles, a partir del ajuste de los pixeles adyacentes, que permite generar un tercer color no presente en dicha área de la imagen original. (2) Dentro del contexto de los gráficos por computadora, es la creación de colores adicionales y sombras a partir de una paleta de colores existente. En monitores monocromo, las sombras de gris son creadas mediante las variaciones y mezclas de puntos de los colores existentes. Esta técnica es utilizada para crear una amplia variedad de patrones para utilizarlos como fondo, relleno o sombreado, así como para la creación de tonos suaves para el trabajo de impresión.

26. **DLL, Librerías.** (*Dinamic Link Libraries*). Véase Librerías de Enlace Dinámico.

27. **Dominio.** Las áreas de una imagen que son fractalmente codificadas. Ver **Rango** .

28. Entropía. (1) La cantidad mínima necesaria para almacenar una imagen. Es decir, representa el número mínimo de bits necesario para almacenar, sin pérdida de información, el valor de cada *pixel* de la imagen. (2) En la compresión de datos, es una medida de la cantidad de información no redundante y no compactable de un objeto.

29. Formato de Imagen Fractal (FIF) , Archivo de. Un tipo de archivo, generado por Iterated Systems© para compactar mediante tecnología fractal, una imagen utilizando la Transformada Fractal.

30. Fractal. Es una estructura que es infinitamente modificable en tamaño, con estructura a cualquier escala, y puede ser representada típicamente por un pequeño y finito conjunto de datos. El término fractal proviene del latín "fractus", que significa fragmentado o dividido en partes. Este término fue acuñado por un investigador de IBM, el doctor en matemáticas Benoit Mandelbrot quien exploró las ideas de los primeros matemáticos de nuestra era y descubrió similitudes extraordinarias en figuras, formas y eventos caóticos y aleatorios.

31. Hardware. (1) Son las unidades físicas o componentes (eléctricos y electrónicos), que constituyen un sistema, que en este caso es una computadora. (2) Se le denomina así a todos y cada uno de los mecanismos y equipo (CPU, discos, cintas, módem, cables, tarjetas, etc.). En operación, una computadora es tanto software como hardware. Uno de estos dos elementos no es funcional sin la existencia del otro. En el diseño del hardware se especifican los comandos que este ha de cumplir, y las instrucciones le indican qué hacer.

32. Imagen. Es un conjunto de datos que representan un escenario bidimensional. Una imagen digital se compone de pixeles organizados en un arreglo rectangular de ciertas dimensiones. Cada pixel puede consistir de uno o más bits de información, que representan el brillo de la imagen y por lo general incluyen información sobre el color, codificado en tripletas RGB. Las imágenes por lo general son tomadas del mundo real vía una cámara digital, un grabador de escenas fijas o un digitalizador.

33. Independencia de Resolución. Una propiedad de las imágenes fractales que les permiten ser mostradas o impresas a resoluciones mayores o menores que la imagen original.

34. JPEG. (1) Un método de compresión con pérdida de datos redundantes (lossy) desarrollado por *Joint Photographic Experts Group* . (2) Es un método estándar para la compresión de imágenes usando la transformada discreta del coseno. Este método realiza una compresión del tipo *lossy* (pues existe pérdida de nitidez con respecto a la imagen original) y produciendo proporciones de compresión de 100:1 y mayores. Esto depende completamente del contenido de la imagen, sin embargo las proporciones de 10:1 y 20:1 generan una pérdida casi imperceptible. La mayoría de la pérdida de datos en una imagen puede ser aceptada, y por tanto esta puede ser compactada. La Compresión es almacenada al dividir la imagen en bloques pequeños de pixeles, los cuales son particionados una y otra vez hasta que la proporción de compresión sea lograda. El JPEG se ha implementado tanto en software como en hardware, *C-Cube Microsystems* introdujo al mercado el primer chip JPEG.
JPEG++ es una extensión de JPEG de *Storm Technology, Mountain View,* CA; el cual permite seleccionar diversas áreas de una imagen, compactadas a diferentes proporciones. Por ejemplo, el fondo podría ser compactado a una proporción mayor que la imagen de primer plano.
El JPEG utiliza el Formato para Intercambio de Archivos JPEG (o JFIF : ***JPEG File Interchange***

Format). Las extensiones utilizadas por este formato son : JPG o JFF. El formato MPEG es la contraparte del JPEG, pero aplicada al video digital.

35. Librerías de Enlace Dinámico. Rutinas utilizables disponibles para las aplicaciones en tiempo de ejecución. Estas son escritas generalmente en código reentrante, de tal forma que estas puedan servir a más de una aplicación en un mismo momento. Bajo DOS, los programas TSR (*Terminate and Stay Resident*, Terminar y Permanecer Residente), han sido utilizados como una forma de agregar funcionalidad en tiempo de ejecución. Estos permanecen en memoria, son interceptados bajo ciertas condiciones y entonces realizan su función. Los programas TSR nunca han sido formalmente prohibidos o condicionados y son fuente de conflictos. Windows, de cualquier forma, ha adoptado el método de las librerías de enlace dinámico, o DLL, como una forma estandarizada de crear una nueva funcionalidad que puede ser compartida dentro del sistema.

36. Luminancia (*Lightness*). Energía emitida por una unidad de área en una unidad de ángulo sólido, es la magnitud fotométrica que se corresponde con las sensación de claridad o *brillo*, cuando esa energía es percibida por el ojo. Describe el brillo del Color. Un color con luminancia del 100% es un Blanco puro, y un color con una luminancia de 0% es un Negro puro.

37. Luz. Radiación electromagnética que viene caracterizada por su frecuencia en hertzios (*Hz*), o por su longitud de onda, λ, en nanómetros (*nm*) o en micrómetros (μm).

38. Marco de Aplicación (de Software). Es una colección integrada de componentes de Software orientado a objetos, que ofrece todo aquello que sea necesario para tener, en primera instancia, una Aplicación de Software genérica en cuanto a características. El marco de Aplicación es un superconjunto de una biblioteca de clases.

39. Memoria Temporal o Bufer. Es una espacio o segmento de memoria utilizado para mantener datos mientras está en operación un proceso que los utiliza. En un programa, la memoria es utilizada para mantener los datos a partir de los archivos de dónde serán leidos o escritos. La memoria puede ser también hardware, como un pequeño banco de memoria, utilizado para propósitos específicos.

40. Muestreo. Para el contexto de las imágenes en medios impresos, representa la digitalización en el espacio de una imagen. Es decir, es digitalizar una imagen mediante el procedimiento de barrido fila por fila. el número de filas que se barran (m) y el número de pixeles que se barran por fila (n).

41. Pixel. La unidad mínima de una imagen de mapas de bits. También se le conoce como elemento básico de la imagen.

***42. PlugIn*, Software.** Es un programa que mantiene un vínculo de comunicación con otro programa o aplicación (programa principal), de manera que el primero permita a este último expandir sus capacidades funcionales en un cierto campo, aplicación o tema en específico.

43. Profundidad de Color. El número de bits requeridos para almacenar el color o intensidad para cada pixel.

44. Programa. Es un conjunto de instrucciones que le indican a la computadora lo que debe hacer. A un programa se le llama también software; por lo tanto, programa, software e instrucciones son

sinónimos. Un programa está escrito en un lenguaje de programación que ha sido trasladado al lenguaje máquina de la computadora mediante un software llamado ensamblador, compilador y/o intérprete.

Un programa se compone de

1. Instrucciones para la máquina.
2. Espacios temporales de memoria o buferes.
3. Constantes y contadores.

Las instrucciones son las indicaciones que la computadora ha de seguir (la lógica del programa). La memoria es espacio reservado, o son las áreas temporales de entrada/salida que aceptan y mantienen los datos mientras comienza su procesamiento. Estos espacios de memoria pueden recibir cualquier tipo de información que sea requerida por el programa.

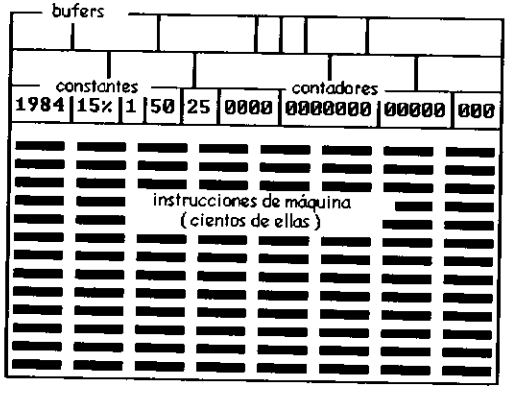
Las Constantes son valores fijos utilizados para comparar datos o valores calculados, tales como valores de rango máximo o mínimo. Los títulos de menú y los mensajes de error son otros ejemplos de constantes.

Los Contadores, también llamados variables, son espacio reservado para la suma o acumulación de cantidades, prácticamente cualquier cálculo, incluyendo aquellos necesarios para mantener las operaciones internas (a bajo nivel), tales como el número de veces que una misma función ha de ser ejecutada.

El programa maneja los datos en una secuencia de entrada, proceso y salida. Después de que los datos han sido incorporados en la memoria del programa a partir de un dispositivo de entrada de datos (teclado, disco, etc.), estos son procesados. Y los resultados entonces son arrojados a algún dispositivo de salida (pantalla, impresora, etc.).

El programa de aplicación, el cual se encarga de procesar los datos reales, no instruye a la computadora sobre todo lo que esta ha de realizar. Cuando ésta está lista para la introducción de datos o requiere de la salida de datos, la computadora manda una solicitud al sistema operativo, el cual activa estos servicios y retorna el control al programa de aplicación.

La siguiente es una ilustración conceptual de una programa residente en memoria. En realidad, en la memoria física existen sólo datos en forma de código de unos y ceros (1 y 0).



ANATOMIA DE UN PROGRAMA

Si bien las instrucciones de máquina son representadas como bloques pequeños, estos pueden tener longitud variable y estos contienen sus propias rutas de acceso lógicas.

45. Programa de Software. Es un programa de computadora (una aplicación para la computadora). Todos los programas de computadoras son software. El uso de dos palabras sinónimas es redundante, pero muy común. Véase **software**.

46. POO (Programación Orientada a Objetos). Abreviado como "POO", es una tecnología de programación que es más flexible que la programación estándar (modular). Es una forma evolutiva de la programación modular con más reglas formales que le permiten a los componentes de software ser reutilizados e intercambiados entre programas. Los conceptos principales de la POO son la Encapsulación, Herencia y Polimorfismo.

La Encapsulación es la creación de módulos autosuficientes que contienen datos y métodos de proceso (la estructura de datos y las funciones que manipulan estos datos). Estos tipos de datos definidos por el usuario, son llamados clases. Una instancia de una clase es llamada Objeto de la clase.

Las Clases son creadas jerárquicamente, y la herencia permite que el conocimiento de una clase sea transmitido o heredado a través de esta jerarquía. Los nuevos objetos pueden ser creados mediante las características que proporciona la herencia a partir de las clases ya existentes. Por ejemplo, el objeto MACINTOSH podría ser una instancia de la clase COMPUTADORAS PERSONALES, la cual hereda sus propiedades a partir de la clase SISTEMAS COMPUTACIONALES. Agregar un nuevo tipo de computadora sólo requiere de la incorporación de aquellas características que la hacen diferente de las computadoras ya existentes, mientras que las características generales de las computadoras personales pueden ser heredadas de la clase SISTEMAS COMPUTACIONALES.

La Programación Orientada a Objetos permite procedimientos sobre los objetos a ser creados cuyo tipo exacto no se conoce hasta el tiempo de ejecución. Por ejemplo, un cursos de pantalla puede cambiar su forma de una flecha a una línea, dependiendo del estado actual del programa. La rutina para mover al cursor sobre la pantalla en respuesta al movimiento del ratón, podría ser escrita como "cursor", y el polimorfismo podría permitir que el cursor sea cualquier figura o forma que se requieran en tiempo de ejecución. Esto podría permitir que una nueva figura sea integrada fácilmente al programa.

Smalltalk©, de Xerox©, fue el primer lenguaje de POO y fue utilizado para crear la interfaz gráfica de usuario, cuyas derivaciones son muy populares ahora en día. C++ ha llegado a ser el lenguaje de POO principal, debido a que este combina la tradicional programación en C con las características Orientadas a Objetos.

A continuación se presenta una comparación de terminología de los dos tipos de programación :

Programación Orientada a Objetos	Programación Tradicional
clase	tipo de datos + características
instancia	variables
instanciar	declarar una variable
método	procesamiento de código
mensaje	llamada
objeto	tipo de datos + proceso

47. Rango. Un conjunto de pixeles que se parecen a un dominio particular, mediante la aplicación de una transformada afín. Ver **dominio**.

48. Razón o Proporción de Aspecto o Apariencia. Véase ***Aspect Ratio***.

49. Razón o Proporción de Compresión. La razón existente entre el tamaño de un archivo no compactado y su contraparte compactado. Es una forma de expresar comparativamente los tamaños del archivo antes y después de la compresión.

50. Resolución. El número de pixeles en una imagen, lo que permite que esta se observe con más o menos nitidez.

51. RGB (Red, Green, Blue). Es un método de colorizado de las imágenes en base a la combinación de los colores Rojo, Verde y Azul. Basado en la Teoría de percepción visual de tri-estimulación de *Helmholtz*. Este método es el más común para definir la proyección de un color bajo la combinación de los colores aditivos Rojo, Verde y Azul. Por ejemplo, el Rojo puro se define mediante los valores Rojo=100%, Verde=0%, Azul=0%. El Negro puro tiene valores de 0% en Rojo, Verde y Azul, en cambio el Blanco puro tendrá valores de 100% en el Rojo, Verde y Azul.

52. Saturación. Es una característica del Color y describe la pureza del Tinte. Un color con una saturación del 100% muestra colores brillosos y *vivos*, y una saturación con 0% es una sombra de gris o colores opacos.

53. Simetría. En el contexto de la Compresión Fractal, es la orientación física de un bloque de rango con respecto a su dominio.

54. Software. Son el conjunto de instrucciones para una computadora. A la serie de instrucciones que realizan una tarea particular se le llama *programa*.
Las dos categorías principales son el software de sistema y el software de aplicación. El software de sistema se compone de programas de control, que incluyen a los sistemas operativos, software de comunicaciones y administradores de bases de datos.

Un concepto comúnmente erróneo es el que se piense que el software también es datos. No lo es. El software le indica al hardware cómo procesar los datos.

El Software se < ejecuta > Y Los Datos se < procesan >

55. Software de Aplicación. El software de aplicación es cualquier programa que procesa datos por y para el usuario (inventarios, nómina, hojas de cálculo, procesadores de texto, etc.). A continuación se presentan las principales categorías del software de aplicación.

Los paquetes integrados y *suites* de aplicación contienen las siguientes cinco aplicaciones :

ADMINISTRADORES DE DATOS. Creación y edición de registros maestros y de transferencia. Edición interactiva de la información. Consultas, compilaciones, ordenaciones, impresión de registros.

PROCESADORES DE TEXTO. Creación y edición de archivos de texto. Reemplaza a toda la funcionalidad de las máquinas de escribir. Algunos programas proporcionan un rudimentario escritorio para el desarrollo de publicaciones.

HOJAS DE CÁLCULO. Creación y edición de renglones y columnas de números para lograr reportes financieros y presupuestos. Los hojas de cálculo proporcionan cortes, o vistas, de los datos rápidamente. Los sistemas avanzados de planeación financiera proporcionan la búsqueda de objetivos así como los cálculos estadísticos.

GRÁFICOS DE PRESENTACIÓN. Creación de presentaciones, realización de dibujos sencillos y la implementación de valores numéricos en gráficos de negocios en 2 y 3 dimensiones.

COMUNICACIONES Y CORREO ELECTRÓNICO. Mandar y recibir datos y el acceso al correo vía módem o sobre alguna red.

Otras categorías de programas de Software de Aplicación son las siguientes :

HERRAMIENTAS DE PUBLICIDAD. Inclusión de texto y gráficos y control completo de los trabajos de impresión. Más precisión que los programas de procesamientos de palabras.

AIP, ADMINISTRADORES DE INFORMACIÓN PERSONAL. Organización de la información para su rápida consulta. Incluye características tales como listas telefónicas con marcador automático, calendario, planeador de actividades, etc.

ADMINISTRADOR DE PROYECTOS. Mantiene la información de proyectos y determina los impactos por posibles cambios. Se calcula la "ruta crítica", la cual monitorea todas las tareas que retardaran al proyecto entero si estas se retrasan.

CAD, DISEÑO ASISTIDO POR COMPUTADORA (gráficos vectoriales). Creación de dibujos para su ilustración y diseño industrial.

TRATAMIENTO DE IMÁGENES. Digitalización de documentos y manipulación de figuras y formas en imágenes, estilo TV.

SOFTWARE DE INFORMACIÓN. Diccionarios en línea, libros y otras referencias con enlaces hipertexto.

SOFTWARE MATEMÁTICO. Creación, ejecución e impresión de complejas ecuaciones matemáticas en variadas formas e interpretaciones.

SOFTWARE CIENTÍFICO. Análisis de eventos del mundo real mediante la simulación de estos en forma matemática. Las supercomputadoras se utilizan ampliamente en estos temas.

MULTIMEDIA. Juegos y educación. La multimedia agrega gráficos, sonido y video para la realización de juegos interactivos, enciclopedias y otras referencias, así como para software educativo para niños.

56. Tamaño Físico. El tamaño de una imagen en pulgadas o centímetros del monitor, impresión en papel o en el mundo real.

57. Tinte (*Hue*). Es una característica del Color que se encarga de describir el tono o sombra del mismo. Esta característica se mide al recorrer un espectro circular desde el color rojo hasta el verde y hasta llegar al azul, para nuevamente regresar al rojo.

58. Transformación Afín. El proceso de rotar, alargar y mover una figura. Una transformación lineal más un desplazamiento.

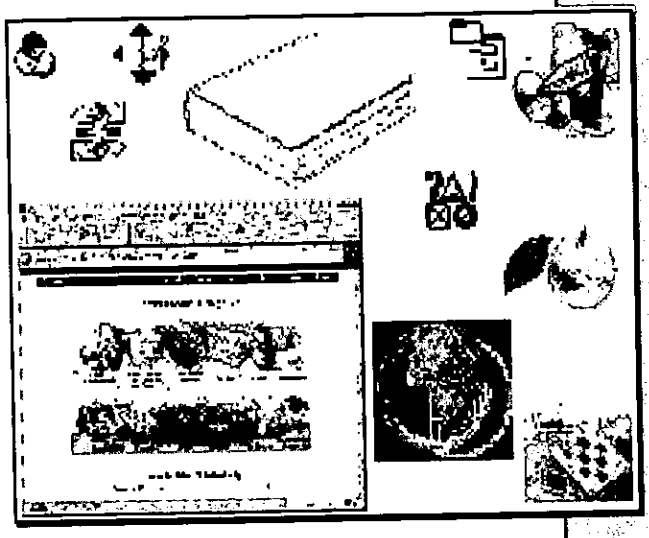
59. Transformada Afín. Expresión matemática de una transformación afín. Son los coeficientes que están dentro de una ecuación dada que convierte un pixel, en un plano x, y de una orientación y ubicación a otra.

60. Transformada Fractal. Se le llama así al proceso, desarrollado por Michael Barnsley, para expresar una imagen del Mundo Real en términos de sus propiedades fractales.

61. Umbral. Separar partes de la imagen que se encuentren entre dos niveles de gris determinados, mediante la umbralización.

62. Umbralización. También conocida como *thresholding*, consiste en respetar el contenido de la imagen salvo una porción, en la parte baja (o alta) de los niveles de intensidad, que se elimina de la imagen, dejándola a cero. Dicha región termina (o comienza), precisamente, en el valor conocido como **umbral**.

REFERENCIAS



BIBLIOGRAFÍA, SOFTWARE E INTERNET

BIBLIOGRAFÍA RECOMENDADA

1. **Michael F. Barnsley.** *Fractals Everywhere*, Ed. Academic Press, San Diego, 1988.

2. **Michael F. Barnsley, J. H. Elton** y **D. P. Hardin.** *Recurrent iterated function systems*, 5(1) : 3-48, 1989.

3. **Michael F. Barnsley** y **Lyman P. Hurd.** *Fractal Image Compression*, Ed. AK Peters, Wellesley. Mass., E.U., Diciembre 1992.

4. **Alberto Domingo Ajenjo.** *Tratamiento Digital de Imágenes*, Ed. Anaya-Multimedia, Madrid, España, 1994.

5. **K. Falconer.** *Fractal Geometry -Mathematical Foundations and Applications*, Ed. John Wiley and Sons, Chichester, 1990.

6. **Y. Fisher. E. W. Jacobs** y **R. D. Boss.** *Fractal image compression using iterated transforms.* Reporte Técnico *1408*, Centro de Sistemas Oceánicos y Navales, San Diego, Cal., 1991.

7. **Y. Fisher. E. W. Jacobs** y **R. D. Boss.** *Fractal image compression using iterated transforms.* Sección correspondiente a **James A. Storer.** *Image and Text Compression.* págs. 35-61, Kluwer Academic Publishers, Boston, 1992.

8. **J. E. Hutchinson**, *Fractals and self-similarity.* Revista Matemática de la Universidad de Indiana, 3(5) : 713-747, 1981.

9. **E. W. Jacobs, R. D. Boss y Yuval Fisher.** *Fractal-based image compression II.* Reporte Técnico *1362*, Centro de Sistemas Oceánicos y Navales, San Diego, Cal., Junio de 1990.

10. **E. W. Jacobs, R. D. Boss y Yuval Fisher.** *Image compression : A study of the iterated transform method*, Secc. Procesamiento de Señales, 29 :251-263, 1992.

11. **A. Jacquin.** *A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding*, Tesis Doctorado, Instituto Tecnológico de Georgia, Agosto de 1989.

12. **David C. Kay** y **John R. Levine.** *Graphics File Formats*, 2da. Edición, Windcrest / McGraw-Hill, E.U., 1995.

13. **Benoit B. Mandelbrot.** *Los Objetos Fractales (Forma, azar y dimensión)*, Tusquets Editores, 2a. De. de la Trad. a la 3ra. edición Francesa, Barcelona, España, 1987.

14. **Benoit B. Mandelbrot.** *The Fractal Geometry of Nature*, Ed. W.H. Freeman, Nueva York, 1983.

15. **James D. Murray** y **William vanRyper.** *Enciclopedia of Graphics File Formats*, 2da. Edición, Ed. O'Really & Associates, E.U., 1996, (Formatos : Libro o CD-ROM).

16. **Dick Oliver, Scott Anderson,** *et al. Tricks of the Graphics Gurus*, SAMS Publishing, *E.U.*, 1993.

17. **Dick Oliver** y **Daniel Houiss.** *Fractal Graphics for Windows*, SAMS Publishing, *E.U.*, 1994.

18. **H. O. Peitgen** y **D. Saupe, editores.** *The Science of Fractals Images*, Ed. Springer-Verlag, Nueva York, 1989.

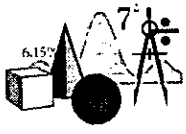
19. **H. O. Peitgen, D. Saupe** y **H. Jürgens.** *Fractals for the Clasroom*, Ed. Springer-Verlag, Nueva York, 1991.

20. **H. O. Peitgen, D. Saupe** y **H. Jürgens.** *Chaos and Fractals : New Frontiers of Science*, Ed. Springer-Verlag, Nueva York, 1992.

21. **Tim Wegner,** et al. *Fractals for Windows*, The Waite Group Press, E.U., 1992.

22. **Tim Wegner** y **Bert Tyler.** *El Mundo de los Fractales*, Trad. y Ed. en español por Anaya-Multimedia, Madrid, 1995.

BIBLIOGRAFÍA para Programación



23. Yuval Fisher. *Fractal Image Compression.* Ed. Springer-Verlag, Nueva York, 1995.

24. David J. Hruglinski. *Programación Avanzada con Visual C++,* Serie de Programación Microsoft, McGraw-Hill, España, 1996.

25. Craig A. Lindley. *Practical Image Processing in C,* De. John Wiley & Sons, E.U., 1991.

26. Mark Nelson. *The Data Compression Book,* 2da. Edición, M & T Books, E.U., 1994

27. Roger T. Stevens. *Fractal Programming in C,* M & T Publishing, E.U., 1989.

28. Roger T. Stevens. *Advanced Fractal Programming in C,* M & T Publishing, E.U.,1990.

29. Mickey Williams. *La esencia de Visual C++ 4,* Trad. Sergio Luis María Faudón, Ed. Prentice Hall Hispanoamericana, México, 1996.

SOFTWARE UTILIZADO

Software para la Edición e Impresión del Documento de Tesis :

30. **Microsoft® Word para Windows 95 v7.0,** © 1993-1995, Derechos Reservados por ***Microsoft Corporation.***

31. **Adobe® PageMaker® v6.5,** © 1993-1996, Derechos Reservados por ***Adobe Systems Incorporated.***

Software para Tratamiento Digital de Imágenes, con los cuales se generaron las Imágenes de este Trabajo de Tesis :

32. **Fractal Design Painter v4.0**, © 1995, Derechos Reservados por ***Fractal Design Corporation***. Acceso WEB : http://www.fractal.com

33. **Paint Shop Pro v4.2**, © 1996, Derechos Reservados por ***Jasc, Inc.*** : P.O. Box 44997, Eden Prairie, MN 55344, E.U.

34. **Corel Photo Paint v5.0**, © 1994, Derechos Reservados por ***Corel Corporation*** : 1600 Carling Avenue, Ottawa, Ontario, Canadá K1Z 8R7.

35. **MATLAB v4.0 y SIMULINK™ v1.2c para Windows,** © 1993, Derechos Reservados por ***The MathWorks, Inc.***

36. **Image Alchemy v1.10,** © 1990-1997, Derechos Reservados por ***Handmade Software, Inc.*** :

48860 Milmont Drive, Suite 106, Fremont, CA 94538. Acceso WEB : http://www.handmadesw.com

37. **WinFrac - FractInt para Windows v17.50,** © 1990-1992, Derechos Reservados por ***The Stone Soup Group and Waite Group Press***. Este software es freeware y acompaña al libro *Fractals for Windows*, Ed. Waite Group Press, 1994.

Software de Aplicación, Complementario para la Preparación y Desarrollo de este trabajo de Tesis, y para la Preparación de la Presentación para el Examen Profesional :

38. **Vault v1.12 para Windows 95/NT**, © 1995, Derechos Reservados por ***Pocket-Sized Software*** : 8547 E. Arapahoe Road, Suite J-147, Greenwood Village, CO 80112 USA.

39. **ACDSee 32 v2.1**, © 1997, Derechos Reservados por ***ACD Systems, Inc.*** : 177 Telegraph Rd #591, Bellingham, WA, USA 98226. Acceso WEB : http://www.acdsystems.com ó http://www.acdnet.com ó http://www.acdsys.com ó http://www.acdvictoria.com ;
Acceso FTP : ftp://ftp.acdsystems.com/acdsee95/ ó
ftp://ftp.servtech.com/pub/users/acdsys/acdsee95/ ó ftp://ftp.islandnet.com/acd/acdsee95/

40. **PicaView 32 v1.0 para Windows 95/NT**, © 1995-1996, Derechos Reservados por ***ACD Systems, Inc.*** : 177 Telegraph Rd #591, Bellingham, WA, USA 98226. Acceso WEB : http://www.acdsystems.com ó http://www.acdnet.com ó http://www.acdsys.com ó http://www.acdvictoria.com Acceso FTP : ftp://ftp.servtech.com/pub/users/acdsys/PicaView/ ó ftp://ftp.islandnet.com/acd/PicaView/ ó ftp://ftp.island.net/pub/acdsys/PicaView/ ó ftp://ftp.coast.net/pub/SimTel/win95/graphics/ ó ftp://ftp.winsite.com/pub/pc/win95/desktop/

41. **Loupe v2.2 para Windows 95/NT**, © 1996, Derechos Reservados por ***Gregory Braun*** : 5609 West Hadley Street, Milwaukee, WI 53210-1554 USA

42. **Jasc Media Center v2.02,** © 1990-1995, Derechos Reservados por ***JASC, Inc.*** : PO Box 44997, Eden Prairie, MN 55344 USA

43. **Thumb Plus 32 v3.0f-S,** © 1993-1997, Derechos Reservados por ***Cerious Software Inc.*** : 1515 Mockingbird Ln. Suite 910, Charlotte, NC 28209 USA.
Acceso WEB : http://www.cerious.com

44. **LabelPrint v2.5,** © 1995, Derechos Reservados por ***Kurt Van den Branden*** : Schoofland 49, 9120 Beveren, Belgium (Bélgica). Correo Electrónico : kvd2@bipsy.se.bel.alcatel.be

45. **Netscape Navigator™ y Editor, Version 3.0Gold-96204,** © 1994-1996, Derechos Reservados por ***Netscape Communications Corporation*** : 501 East Middlefield Road, Mountain View, California, 94043.

46. **Slim Show v1.0,** © 1996, Derechos Reservados por ***PC WholeWare*** : 33 Justin Street, Lexington.

MA 02173, USA. Acceso WEB : http://users.aol.com/WholeWare/index.html

47. **Microsoft® Power Point para Windows 95 v7.0,** © 1983-1995, Derechos Reservados por ***Microsoft Corporation.***

Software de Desarrollo, utilizado para realizar las implementaciones basadas en los códigos de programa de *Yuval Fisher* y *Jude Sylvestre*, y las librerías de Compresión Fractal v1.2, de *Iterated Systems Inc.*, así como para el desarrollo del Sistema de Ayuda :

48. **HotSpot Editor v3.50.784,** © 1991-1992, Derechos Reservados por ***Microsoft Corporation.***

49. **HWA/w - Help Writer's Assistant para Windows v1.0 beta IV,** © 1992-1994, Derechos Reservados por ***Olson Software*** : 4 Anaru Place, Palmerston North, New Zealand.

50. **Microsoft® Visual C++™ v4.0,** © 1994-1995, Derechos Reservados por ***Microsoft Corporation.***

51. **Microsoft® Help Workshop,** © 1994-1995, Derechos Reservados por ***Microsoft Corporation.***

52. **MFC Migration Tool,** © 1994, Derechos Reservados por ***Symantec Corporation*** y ***Microsoft Corporation.***

53. **Visual Help v2.1e,** © 1993-1994, Derechos Reservados por ***Firas Bushnaq*** de ***WinWare Inc.*** : P.O. BOX 2923, Mission Viejo CA 92690.

Referencias INTERNET

Las siguientes direcciones WEB en INTERNET sirvieron de fuentes de información complementaria para el desarrollo de este Trabajo, y fueron accesadas por última ocasión en el mes de Agosto de 1997. Se ha hecho un gran esfuerzo por mantener actualizadas las direcciones de acceso hasta el día de impresión final de este documento, sin embargo, dado que Internet es muy dinámico, *la vigencia de esta lista se deteriora con el transcurso de los días*; razón por la cual, lo más indicado es utilizar los "motores" o servidores de búsqueda en Internet para localizar las nuevas direcciones o nuevas páginas referentes al mismo tema.

Para mayor facilidad de uso de esta lista, se encuentra en el disco que acompaña a este Trabajo, una página HTML llamada **Fractals.htm**, con estas mismas referencias y las páginas iniciales de la mayoría de ellas. Para su utilización sólo es necesario contar con algún navegador WWW (Netscape Navigator o Internet Explorer, por ejemplo), y una conexión a INTERNET.
Existen dos formas, complementarias entre sí, para utilizar esta lista de direcciones : la primera llamada *OffLine*, que se refiere a navegar sobre las páginas, en base a la página base Fractals.htm, *sin línea activa a INTERNET*, lo cual permitirá analizar y clasificar en primera instancia toda la información contenida en las páginas iniciales (si está disponible una página inicial, esto se indicará en la página base Fractals.htm, mediante : "⇨ (Acceso Local)") y así planear, una vez que la conexión a INTERNET esté activa, las referencias a visitar; mientras que la segunda forma llamada *OnLine*, es la navegación sobre la información *con la conexión a INTERNET activa*, ya sea tanto para observar las páginas locales o accesar sobre la actualización de las mismas páginas, pero sobre INTERNET.

54. *Fractal Image Encoding*, **http://inls.ucsd.edu/y/Fractals/**, Esta página contiene un gran número de enlaces a una gran variedad de información y recursos sobre la Codificación Fractal de Imágenes y otros temas relacionados. El contenido general de esta página es : 1. Bibliografía y otros Referencias, 2. Libros, 3. Conferencias y Noticias, 4. Recursos Internet, 5. Documentación sobre la Codificación Fractal, 6. Software, y 7. Créditos sobre los estudiantes participantes en la página, reimpresiones, el Instituto de Ciencia No Lineal (INLS) y Yuval Fisher.

55. *Fractal Image Encoding Announcement and Questions*, **http://inls.ucsd.edu/y/Fractals/frac-an.html**, Esta página contiene varios anuncios y preguntas relacionadas a la Codificación Fractal de Imágenes, realizadas por investigadores del tema, principalmente.

56. *A Study of Fractal Image Compression*, **http://ipsun1.cs.nthu.edu.tw/~mr834342/fic.html**, Esta página es elaborada por un estudiante Taiwanés sobre la Compresión Fractal de Imágenes. Es desarrollador de aplicaciones de Compresión Fractal y permite el acceso a una aplicación llamada FICA, desarrollada por él mismo y que funciona bajo Windows 95.

57. *comp.compression Frequently Asked Questions (part 1/3)*,
http://www.landfield.com/faqs/compression-faq/part1/index.html, Documento compilado del *Grupo de Discusión* comp.compression, donde hay variada información sobre algoritmos de compresión en general, preguntas comunes sobre los programas comerciales de compresión, comparativas de algoritmos, clasificación, referencias a otras páginas y solución a dudas o problemas sobre compresión. El Contenido de esta página está clasificado en base a preguntas, las cuales son las siguientes (existen números de preguntas que no aparecen debido a que sólo se está proporcionanado aquellas cuestiones sobre la compresión que son más importantes) :

[1] ¿ Qué son los Grupos de Discusión ?
[2] ¿ Qué tipo de archivo compactado es de acuerdo a su extensión ?
¿ Dónde puedo encontrar los programas de Compresión correspondientes ?
[3] ¿ Cuál es la última versión de Pkzip ?
[4] ¿ Qué es un Archivador ?
[5] ¿ Cuál es el mejor programa de compresión de propósito general ?
[7] ¿ Qué libros me recomiendan leer ?
[8] ¿ Qué hay sobre los patentes de los algoritmos de compresión de datos ?
[9] Compresión de datos aleatorios (WEB, Gilbert y otros)
[10] Supuestos Programas de Compresión (OWS, WIC)
[11] ¿ Cuál es el estándar V.42bis ?
[12] Necesito las fuentes de código de los ganadores del concurso del Dr. Dobbs
[13] Necesito las fuentes de código para la codificación aritmética

Compresión de Imagen y audio :

[15] ¿ Dónde puedo obtener programas de Compresión de Imágenes ?
[16] ¿ Cuáles con los últimos avances en la compresión de imágenes tipo *lossless* ?
[17] ¿ Cuál es el estado actual de la compresión fractal ?
[18] Necesito las especificaciones y código fuente el estándar Fax 4 Grupo TIFF y CCITT.
[19] ¿ Qué es JPEG?
[20] Estoy buscado fuentes de código de un codec (codificador-decodificador) H.261/H.263 y MPEG
[25] Algoritmos de la Transformada Rápida del Coseno
[26] ¿ Existen algoritmos y estándares para la compresión de audio ?

Problemas Comunes :

[30] Mi Archivo está dañado!, ¿ qué puedo hacer ?
[31] PKunzip reporta un error CRC !
[32] El zip VMS no es compatible con pkzip!
[33] Tengo un problema con Stacker o DoubleSpace!

Cuestiones que no son parte de la información de comp.compression :

[50] ¿ Qué es el programa de compresión 'tar' ?
[51] Necesito un algoritmo de CRC
[53] ¿ Dónde se localizan los archivos de listas de Preguntas más solicitadas (FAQs) ?
[54] Necesito especificaciones sobre los formatos gráficos
[55] ¿ Dónde puedo encontrar la imagen original de Lenna y otras imágenes ?
[57] Perdí mi clave para descompactar un archivo .zip

58. *Waterloo Fractal Compression Project*, **http://links.uwaterloo.ca**, Servidor dedicado al estudio del análisis Fractal, los Sistemas de Funciones Iteradas y Transformadas Fractales, desde ambas perspectivas : teórica y práctica. El contenido general de la página principal es el siguiente : 1.Introducción, 2. El Grupo de Investigación Waterloo, 3. Noticias y Eventos próximos, 4. La «guía de Hitchhiker para la Compresión Fractal» para iniciadores, 5. Resumen actualizado de Investigación en Waterloo, 6. Material disponible a partir de este proyecto, 7. El proyecto de Colaboración Montreal-Waterloo-INRIA-Verona, 8. Calendario de Encuentros de personas relacionadas con los fractales en Waterloo y 9. Otros asuntos de Compresión Fractal y Sitios relacionados.

59. *Programas sobre Compresión Fractal*, **ftp://links.uwaterloo.ca/pub/Fractals/Programs/**, Contiene enlaces a códigos de programa desarrollados por estudiantes e investigadores en la materia, ya sea en forma individual o conjunta. Se encuentra el acceso a la Página llamada Hollatz (así se llama la persona que mantiene esta página), de la versión 2.0 del programa de Jude Sylvestre: **ftp://links.uwaterloo.ca/pub/Fractals/Programs/Hollatz/**, y otra página llamada Limbo (así se llama la aplicación de Compresión Fractal), además se encuentra el acceso a otros programa en código fuente llamados Pulcini.zip y Anson.zip, este último corresponde al código fuente del libro de la Referencia **[3]**.

60. *Wyne D. Young*, **ftp://inls.ucsd.edu/pub/young-fractal/**, Página con los enlaces para obtener código fuente de las 3 versiones del programa de compresión fractal de imágenes propiedad de Wyne D. Young. Dichas versiones Young las clasifica así : Archivo **unifs09b.tar.Z** para usuarios iniciales, archivo **unifs10.tar.Z** para usuarios Intermedios y archivo **yuvpak20.tar.Z** para usuarios avanzados.

61. *Waterloo BragZone*, **http://links.uwaterloo.ca/bragzone.base.html**, Página dedicada a la comparación de resultados de varios métodos de compresión de imágenes en base un conjunto de 32 elementos de prueba.

62. *BragZone*, **ftp://links.uwaterloo.ca/pub/BragZone/**, Página con enlaces a las imágenes originales, fuente de comparación de los métodos de compresión de imágenes de la referencia anterior, clasificadas por tipo de archivo (**ftp://links.uwaterloo.ca/pub/BragZone/Collected/**), por Escala de Grises (Conjunto 1 : 12 imágenes pequeñas **ftp://links.uwaterloo.ca/pub/BragZone/GreySet1/**, y Conjunto 2 : 12 imágenes medianas **ftp://links.uwaterloo.ca/pub/BragZone/GreySet2/**), y por Conjunto de Imágenes a Color (8 imágenes de colores reales de tamaño grande) **ftp://links.uwaterloo.ca/pub/BragZone/ColorSet/** .

63. *The SPANKY Fractal Database*, **http://spanky.triumf.ca/www/SPANKY.HTML**, Página de enlaces a otros sitios referentes al tema de los fractales en varias aplicaciones. Cuenta con enlaces a documentación, código de programa, aplicaciones software sobre Fractales, etc.

64. *Fractal Compression*,
http://www-iapr-ic.dimi.uniud.it/Udine/WebRes/ImageCoding/compress/fractal.htm, Página con información introductoria para la Compresión Fractal. Contiene referencias, preguntas, pros y contras de la compresión fractal argumentadas por investigadores y promotores de la Compresión Fractal.

65. *Fractal Image Compression*,
http://www-vs.informatik.uni-ulm.de/Mitarbeiter/Kassler/fractals.htm, Página mantenida por el propio autor, Andreas Kassler, con el fin de mostrar sus trabajos de Tesis sobre Compresión Fractal. El programa que proporciona en forma de aplicación se llama fracomp v1.0 Compresión Fractal de Imágenes bajo Windows.

66. *Ken's Fractal Image Compression Link Page*,
http://192.47.99.34/ken/Link4Compression.html, Página de enlaces a otros sitios referentes a la Compresión Fractal de Imágenes. Enlaces a páginas como : Iterated Systems, inc., Revolucionary Fractal Image Compression, Kumano.., Multimedia Imaging & Compression Services, etc.

67. *Iterated Systems*, **http://www.iterated.com/** y **http://www.iterated.com/contents.htm**, Página principal y de contenido de la empresa pionera en la compresión fractal de imágenes. Contiene enlaces a Productos y soluciones variadas de la misma empresa, fundamentadas en la Compresión Fractal, Soporte técnico, sección de preguntas más realizadas (FAQ), etc.

68. *Graphics*, **http://www.cs.cmu.edu/afs/cs.cmu.edu/user/spot/web/images.html**, Página de enlaces a imágenes, animaciones, código fuente, etc., sobre fractales.

69. *SONIC FRACTAL ambient music*, **http://www.galivant.com/~tjustman/sonic.htm**, Página dedicada a la experimentación de los Fractales en el mundo de la música, y presenta algunos cortos de sonido de interpretaciones logradas con fractales.

70. *Fractal Movie Archive*, **http://www.cnam.fr/fractals/anim.html**, Página con enlaces a diversas animaciones logradas proporcionadas por sus propios autores. Se encuentran en formatos Anim5, Fli, Flc, Mpeg y QuickTime.

71. *Genuine Fractals Demo Download*, **http://www.altamira-group.com./demo.html**, Página para la transferencia del demo de la aplicación Genuine Fractals la cual es un conjunto de filtros que permiten leer el formato FIF, desde cualquier aplicación de tratamiento de imágenes que soporte la incorporación de los PlugIn (aplicación o librería incorporada).

72. *The Fractory : An interactive Tool for Creating and Exploring Fractals*, **http://tqd.advanced.org/3288/fractals.html**, Esta página les permitirá aprender sobre los fractales de una forma más simple e intuitiva : ¿ qué son y cómo se diseñan ?. Presenta herramientas para utilizar los Fractales como predictores de eventos naturales.

Las siguientes son direcciones de páginas WEB cuyo contenido es principalmente de *imágenes* de todo tipo, incluyendo fotografías de pinturas de autores mexicanos, así como de lugares prehispánicos de México y de otros lugares antiguos de otros paises.

73. *Una Exhibición de Arte Mexicano*, **http://www.cibola.net/arte/**, Colección de imágenes de Pinturas de Diego Rivera, José Clemente Orozco, David Alfaro Siqueiros y Frida Kahlo.

74. *David Alfaro Siqueiros*, **http://www.udg.mx/recreacion/pinturas/david/**, Biografía y muestra de imágenes de pinturas de David Alfaro Siqueiros.

75. *Los Murales Mexicanos*, **http://www.spin.com.mx/ilustrado/murales/**, Muestras y referencias de murales de autores como Diego Rivera, José Clemente Orozco y David Alfaro Siqueiros.

76. *Museo de Arte Moderno - Escuela Mexicana*, **http://www.arts-history.mx/museos/mam/escmex.hmtl**, Muestrario de Obras de José Clemente Orozco, Diego Rivera, David Alfaro Siqueiros, además obras de Julio Castellanos, Olga Costa, Frida Kahlo, Guillermo Meza, Juan O'Gorman y Manuel Rodríguez Lozano.

77. *Museo de Arte Moderno - Ruptura*, **http://www.arts-history.mx/museos/mam/ruptura.hmtl**, Muestrario de Obras de Pedro Coronel, Manuel Felguéres, Gilberto Aceves Navarro, Lilia Carrillo, Fernando García Ponce, Rufino Tamayo y Cordelia Urueta.

78. *Muxeo Picasso Virtual*, **http://www.tamu.edu/mocl/picasso/**, Galeria Virtual de Pinturas de Picasso.

79. *Welcome to the Mexican Photo Album - Mexico Pictures*, **http://wwst10.wiwi.uni-wuppertal.de/~denache/mexico/mex.html**, Colección personal de fotografías de Daniel Enache sobre lugares de México, tales como : las ruinas de Palenque, Chichen Itza (Pirámide de Quetzalcoatl o Kukulcán), Teotihuacán (Pirámides del Sol y la Luna), La Selva de Palenque, La cascada de Misol-Ha en Chiapas, la Cd. de Taxco Guerrero, etc.

80. *Chiapas*, **http://www.colostate.edu/Orgs/LASO/Mexico/Documents/chiapas.html**, Colección de estupendas fotografías de Palenque, las Cascadas de Agua Azul, las Lagunas de Montebello y de la Cascada Misol-ha.

81. *Mex-Facts*, **http://sunbum.uwaterloo.ca/~leon/mex-fact.html**, Información general sobre México, su historia, geografía, población, recursos, mapas geográficos y fotografías de lugares turisticos como : La *quebrada* en Acapulco, Las playas de Cancún, Los Atlantes de Tula, El lago de Pátzcuaro, etc.

82. *THE GALERIES*, **http://www.metatron.se/galleryp.html**, Colección de fotografías de lugares antiguos de México, Egipto, Perú, la India e Inglaterra. Por ejemplo : las Pirámides del Sol y la Luna en México, la Pirámide de Keops en Egipto, el interior del sepulcro de los Faraones, Machupichu en Perú, El observatorio más antiguo del Mundo en Inglaterra, El palacio Taj Mahal en la India, etc.

83. *Planet Earth Home Page - Images Section*, **http://www.nosc.mil/planet_earth/images.hmtl**, Diversos

tipos de mapas del planeta Tierra, desde planisferios hasta mapas por regiones fotografiados por satélites.

84. *Infinite Fish*, **http://www.infinitefish.com/main.hmtl**, Dibujos, gráficos, texturas, tips de programación, tutoriales, diseño Web, etc.

85. *The Digital Photography Exhibit*, **http://www.bradley.edu/exhibit95**, Fotografías Digitales.

86. *Image Club Graphics*, **http://www.imageclub.com**, Noticias sobre novedades en el mundo de la imágenes, compra de productos, club, archivos y recursos en general.

87. *Agfa*, **http://www.agfahome.com**, Cámaras digitales, archivos de imágenes, muestras, noticias, etc.

88. *3D site*, **http://www.3dsite.com/3dsite**, Gráficos en 3D y todo tipo de recursos relacionados.

89. *Digital Stock*, **http://www.digitalstock.com**, Venta de imágenes digitales, muestras y catálogos.

90. *3D World*, **http://www.geocities.com/siliconvalley/4378**, Información relacionada con 3D Studio, Vistapro, galería de imágenes, enlaces a otros sitios, etc.

91. *Zoonet Image Archives*, **http://www.mindspring.com/~zoonet/gallery.hmtl**, Todo tipo de imágenes, dibujos y animaciones sobre animales.

94. *Free Art Web Site*, **http://www.mccannas.com**, Gráficos, libros, enlaces a otras páginas, etc.

93. *Graphics viewers, editors, utilities and info*, **http://www.hotlist/graphics.html** ó **http://www2.ncsu.edu/bae/people/faculty/walker/**, Listas de enlaces a páginas relacionadas al manejo de imágenes y gráficos.

94. *Graphics Archive*, **http://www.eff.org/pub/graphics**, Archivos gráficos en general y enlaces a otros sitios.

95. *The Plugin Head*, **http://pluginhead.eyecandy.com**, Aplicaciones, filtros, libros, noticias, revistas y enlaces a otras páginas.

96. *The electronic publishing authority*, **http://www.publish.com**, Publicación electrónica especializada sobre grafismo.