58

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

TUTORES PROGRAMADOS

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A :

SALVADOR DAVID HUERTA SINECIO

ASESOR DE TESIS: FISICO RAYMUNDO HUGO RANGEL G.

MEXICO, D. F. 1995

CONTENIDO

INTRODUCCION

El ser humano siempre se ha encontrado rodeado por objetos, los objetos son parte de nuestra vida diaria, por lo cual es más fácil entender un problema cuando en el intervienen objetos. Es por eso que las técnicas de análisis y diseño de sistemas (no sólo software) de hoy en día están orientadas hacia los objetos.

En las metodologías tradicionales de desarrollo de software, los modelos conceptuales para el análisis difieren de aquellos usados para el diseño. Los analistas usan diagramas de flujo de datos, diagramas de entidad relación, funciones de descomposición, y diagramas de proceso dependencia. Los diseñadores usan cartas estructuradas y diagramas de acción. Los programadores utilizan lenguajes como COBOL, BASIC, PASCAL, C, etcétera. Como se podrá observar, las personas que intervienen en la creación, en las distintas etapas, de un sistema (analistas, diseñadores, programadores y usuarios) poseen una perspectiva diferente y por lo cual el sistema resultante, en la mayoría de las veces, no satisface todas las necesidades del usuario.

En técnicas orientadas a objetos, analistas, diseñadores, programadores y (lo que es más importante) usuarios utilizan el mismo modelo conceptual. Todos ellos piensan en términos de objetos, tipos de objetos y cual es el comportamiento de dichos objetos. Ellos trazan jerarquías de tipos de objetos o clases en las cuales los subtipos heredan las propiedades de sus antecesores. Estas personas piensan en objetos compuestos de otros objetos que a su vez se

CONTENIDO

INTRODUCCION

El ser humano siempre se ha encontrado rodeado por objetos, los objetos son parte de nuestra vida diaria, por lo cual es más fácil entender un problema cuando en el intervienen objetos. Es por eso que las técnicas de análisis y diseño de sistemas (no sólo software) de hoy en día están orientadas hacia los objetos.

En las metodologías tradicionales de desarrollo de software, los modelos conceptuales para el análisis difieren de aquellos usados para el diseño. Los analistas usan diagramas de flujo de datos, diagramas de entidad relación, funciones de descomposición, y diagramas de proceso dependencia. Los diseñadores usan cartas estructuradas y diagramas de acción. Los programadores utilizan lenguajes como COBOL, BASIC, PASCAL, C, etcétera. Como se podrá observar, las personas que intervienen en la creación, en las distintas etapas, de un sistema (analistas, diseñadores, programadores y usuarios) poseen una perspectiva diferente y por lo cual el sistema resultante, en la mayoría de las veces, no satisface todas las necesidades del usuario.

En técnicas orientadas a objetos, analistas, diseñadores, programadores y (lo que es más importante) usuarios utilizan el mismo modelo conceptual. Todos ellos piensan en términos de objetos, tipos de objetos y cual es el comportamiento de dichos objetos. Ellos trazan jerarquías de tipos de objetos o clases en las cuales los subtipos heredan las propiedades de sus antecesores. Estas personas piensan en objetos compuestos de otros objetos que a su vez se

componen de otros objetos. Ellos piensan en eventos que cambian el estado de los objetos y de eventos que disparan ciertas operaciones. La perspectiva de estas personas es la misma. El usuario entiende el mundo de los analistas y diseñadores, y a su vez los analistas y diseñadores comprenden el mundo del usuario debido a que los creadores del sistema y el usuario hablan el mismo lenguaje, los objetos, y por ende el sistema resultante, de la labor de estas personas, es de más alta calidad.

Es difícil determinar el límite donde termina el análisis y el límite de donde inicia el diseño, es por eso que la transición del análisis al diseño es tan natural.

Al aplicar el enfoque orientado a objetos para la creación de sistemas informáticos, se consigue uniformizar las distintas etapas del ciclo de vida de un sistema. El enfoque orientado a objetos nos ofrece ciertas ventajas, las que a continuación se listan:

1. Uniformidad.
2. Entendible.
3. Código reusable.
4. A la larga resulta económico.
5. Los sistemas resultantes son de alta calidad.
6. Las habilidades del profesional en computación son mejores que las habilidades de los profesionales (tradicionales) que se niegan al cambio.
7. Sistemas más estables.
8. Tanto el análisis, el diseño y el desarrollo de sistemas es más rápido.

9. Integridad.
10. Fácil programación.
11. Fácil mantenimiento.
12. Ciclo de vida dinámico.
13. Refinamiento de módulos a lo largo del ciclo.
14. El modelado es más real y no tan abstracto como el modelo tradicional.
15. Bibliotecas de clases en venta por compañías de software.
16. Bibliotecas de clases propias.

La lista anterior seguiría creciendo debido a la gran variedad y cantidad de ventajas que nos ofrece el enfoque orientado a objetos.

Debido a estas ventajas que nos ofrece el enfoque orientado a objetos, en este trabajo se utilizó este enfoque para crear una aplicación. Los capítulos de este trabajo de tesis cubrirán a detalle las diferentes etapas de elaboración de una aplicación real. Hyptextw es una aplicación enfocada para crear libros electrónicos. Hyptextw más que nada será una herramienta para el escritor de libros. Para poder utilizar la aplicación es necesario seguir unas reglas que se explican en el apéndice D.

Hyptextw puede ser considerado en el ámbito, ya tan conocido por todos nosotros, de "multimedia". La multimedia abarca campos tales como redes de neuronas (simulación), procesamiento de imágenes y video, reconocimiento de voz, inteligencia artificial, traducción de un idioma a otros idiomas, robótica, etcétera. Por tal causa se hablará sobre la multimedia en el primer capítulo. En el capítulo dedicado al análisis se dará una breve

explicación de lo que son los tipos de objetos o clases, los objetos, las relaciones entre objetos, el esquema de generalización, el esquema de los objetos (object schema), el diagrama de transición de estado y el diagrama de eventos; todo ello enfocado hacia nuestro sistema (Hyptextw). En el capítulo dedicado al diseño, se tratarán temas como el polimorfismo, que clases se implementaran, operaciones que nos ofrecen las clases y cuales son sus métodos, que estructura de datos utilizaran las clases, etcétera, todo ello aplicado a nuestro sistema. En el capítulo dedicado a la implementación se indicará como se creo el sistema. La herramienta utilizada fue Turbo C++ 3.0 de Borland para Windows, hay que hacer hincapié que para crear la interfaz de usuario y el editor, se utilizo ObjectWindows. OjectWindows es una biblioteca de clases que nos ofrece ciertas ventajas para crear sistemas con una alta presentación. Renglones atrás hable del termino reusabilidad, la reusabilidad de código, métodos y clases hace posible un desarrollo más rápido y más eficiente, ObjectWindos nos ofrece ese código y esas clases. En el apéndice B se da un bosquejo de lo que es ObjectWindows.

CAPITULO 1

LA MULTIMEDIA EN NUESTRA VIDA DIARIA

La multimedia ya es una realidad hoy en día y es palpable en cualquier área: cine, educación, por sólo nombrar algunas. La multimedia, hace unas décadas atrás, era vista como de ciencia ficción, de hecho en la actualidad todavía no llega a un punto de madurez. La multimedia es proclamada como la siguiente revolución en computación y con justa razón debido a la gran calidad en la exhibición de imágenes, texto y sonido. Existen más tecnologías que por algunas personas no son consideradas dentro de la multimedia, tales como, los KBSs (Knowledge-Based Systems), los CBR (Case-Based Reasonig), las redes de neuronas, algoritmos genéticos y los sistemas híbridos (que están compuestos de dos o más tecnologías de las ya mencionadas). Estas tecnologías en conjunción con la multimedia (cuyo rango de definición es muy limitado: video, sonido, gráficas y texto) hacen de las nuevas aplicaciones, en verdad, sistemas casi humanos. Imaginese una aplicación con la calidad de la multimedia como interfaz y de un sistema KBS mezclado con redes de neuronas (más adelante hablare de las arquitecturas que existen entre KBSs y redes de neuronas, incluyendo ejemplos de aplicaciones existentes) como núcleo, el resultado es un sistema con inteligencia para resolver problemas de negocios o si usted prefiere un asistente para una biblioteca de video clips. En mi punto de vista la multimedia debe ser: video, sonido, gráficas, texto y tecnologías orientadas a la inteligencia. Es por eso que yo incluyo en este capítulo información sobre redes de neuronas, KBS (Knowledge-Based System), CBR (Case-Based Reasoning), algoritmos genéticos y sistemas híbridos, la cual podrá ser encontrada en la sección la técnica hace al software.

LA MULTIMEDIA SEGUN EL MUNDO DE LA INFORMATICA

El buscar mejores formas para presentar información a los demás ha ocupado a muchas mentes, desde que los humanos comenzaron a dibujar pinturas, en la búsqueda de una alternativa a la comunicación oral. Miles de años atrás antes de la aparición de la escritura, la comunicación entre las personas era en base a pinturas, gestos y sonidos que en una forma ayudaba a comunicarse entre ellos.

A partir de entonces la gente siempre se ha preocupado en buscar mejores representaciones de la información, el significado de representaciones de información ha variado a lo largo de los años.

La esperanza es ver si la gente puede entender nuevos conceptos en una forma rápida de tal forma que su capacidad en tareas intelectuales aumentará considerablemente. Para cualquiera, la mejor comprensión de las cosas (negocios) posee la promesa de una base solida de ideas.

Las habilidades de una computadora para manipular la información y presentarla en diferentes tipos de formas ha comenzado a destrozar la vieja forma de expresión estática. Muchas universidades y compañías se están enfocando en estas tendencias bajo el nombre de multimedia.

En un principio los esfuerzos fueron enfocados en los escasos requerimientos de mostrar un simple video en la pantalla del monitor de una computadora, la nueva visión es más amplia. El nuevo espectro de actividades

abarca: digitalizar, mezclar, manipular y comprimir múltiples señales de video y audio.

Desafortunadamente, la prensa no técnica ha notado y persigue con gran interés a la multimedia, ellos han cubierto la multimedia con tal fervor que han fallado en mostrar los verdaderos beneficios de este conjunto de tecnologías llamada multimedia. Tal entusiasmo ha llevado a un mar de confusión a los usuarios. Los departamentos en mercadotecnia también se han agregado al conjunto de confundidos al usar el término multimedia para describir cualquier cosa, desde ver TV en la pantalla de la computadora hasta juegos de video. Desafortunadamente, casi cualquier cosa con gráficas o sonido en un CD-ROM califica como multimedia para muchas gentes. Encaremoslo, ¿cuantas aplicaciones de multimedia realmente útiles ha visto usted en estos últimos años?, cierto existen muchas aplicaciones estilo Holywood que son realmente impresionantes, pero todo esto no deja de ser un estrecho campo de aplicación. Más allá de eso, usted sólo se encontrara con una pila descomunal de juegos de video para computadora, un poco de software educacional y una pizca de aplicaciones técnicas que realmente hacen muy poco por usted. Todo esto y además con la ayuda de la prensa no técnica, ayuda a que muchas personas sean desorientadas de lo que debe ser el objetivo real de la multimedia. Más adelante veremos como se pueden crear aplicaciones mucho más útiles al utilizar la multimedia en combinación con otras tecnologías orientadas al conocimiento. Veamos una analogía, existen muchas chicas bellísimas pero que tienen cero inteligencia y que por lo tanto no sirven para tareas intelectuales; pero también existen chicas bellísimas con un alto grado de sabiduría. Supongo que ya saben a donde deseo llegar, en efecto, la multimedia sola es la analogía

de la chica bella pero tonta, mientras que la multimedia en compañía de otras tecnologías orientadas al conocimiento (tecnologías inteligentes) son la analogía a la chica bella y además inteligente.

Más adelante me enfocaré en sólo explicar aquello que hace más útil a la multimedia, pero mientras continuemos viendo un enfoque muy somero de lo que es la multimedia en nuestros días.

Lo que es más halagador es que ya no es necesario poseer computadoras demasiado caras, tales como estaciones de trabajo o macrocomputadoras, para realizar este tipo de cómputos. Sistemas PC con alta potencia de proceso están apareciendo en el mercado, lo cual implica que millones de personas en el mundo podrán tener acceso a la multimedia.

Todavía se están teniendo muchos problemas con este conjunto de tecnologías, por ejemplo, para poder disfrutar de las magias del video es necesario procesar 30 imágenes diferentes por segundo. Cada imagen es construida a partir de una señal analógica que es capaz de representar un increíble rango de colores. Desafortunadamente, la cantidad de imágenes por segundo y los diferentes colores en cada imagen producen problemas al convertir una señal de video en datos digitales. Cada imagen de una típica señal de video ocupa más de 500 KB de disco duro. Cada segundo de una señal de video ocupa 15 MB de almacenamiento. Esto es realmente inaceptable, debido a que limita el tamaño de un segmento de video. Es peor cuando se almacena una señal de video en un CD-ROM, el cual provee de una alta densidad en almacenamiento pero es demasiado lento en mover información, más lento que

un disco duro. Pero todo problema lleva a una solución y la solución, para tal gasto en disco duro por parte de señales de audio y sobre todo de video, es la tecnología para comprimir imágenes (para más información, por favor ver la sección: El arte de comprimir imágenes).

Aunque algunos problemas de funcionalidad están siendo solucionados, existe todavía un largo trecho por recorrer, debido a que todos sabemos instintivamente que todas estas tecnologías de multimedia pueden ser utilizadas en un uso real y útil para todos.

El Arte de Comprimir Imágenes

Cuando se habla de que un video clip de 10 segundos de duración a 30 imágenes por segundo, con colores reales y con una resolución de 320 por 200 pixels, ocupa 57.6 MB de disco duro, claramente se necesita de la compresión de imágenes. JEPG es una técnica comúnmente usada en este tipo de problemas. Quizás usted nunca haya oído hablar de la técnica FIC (Fractal Image Compression) para comprimir imágenes. FIC combina un alto grado de compresión con una alta velocidad de descompresión.

Las tecnologías en compresión pueden ser divididas en dos categorías: lossless y lossy. Cuando se utiliza un método lossless, la imagen (ya descomprimida) que se obtiene es idéntica, pixel por pixel, a la original. Pero existe un problema, las aplicaciones que utilizan alguna metodología lossless sólo pueden ser comprimidas hasta un 50 %. Los métodos lossy, en cambio, poseen un altísimo porcentaje de compresión de imágenes. Tanto el DCT

(Discrete Cosine Transform) como el FIC son métodos del tipo lossy, pero es lo único en que son iguales porque en todo lo que resta son totalmente diferentes.

Compresión JPEG/DCT

En el mundo de la compresión, la actual respuesta es JPEG. Esta norma, definida por la Join Photographic Experts Group, describe las formas de tomar los mapas de bits, que representan imágenes en color o monocromático, y cuyo almacenamiento sólo ocupa una cantidad pequeña de Kilo Bytes. Esta norma vio la luz en el año de 1986. El principal objetivo, de la JPEG, fue el buscar el mejor método para la comprensión de imágenes y adoptarlo como una norma a nivel mundial. El corazón de esta tecnología es DCT.

DCT rompe una imagen en bloques de 8 por 8 pixels y después echa mano de trucos matemáticos para decidir que información puede ser desechada sin deteriorar mucho la imagen. DCT transforma la imagen matemáticamente desde un espacio (x, y) a un espacio de frecuencias. En lugar de ver los datos como un arreglo de 64 valores arreglados en un forma matricial de 8 por 8, DCT ve la información como una colección de 64 funciones coseno con apropiadas amplitudes.

La información más importante es concentrada en las funciones coseno con más baja frecuencia. Así, dándole menos peso a las funciones coseno que poseen más alta frecuencia y aproximando los coeficientes de menor valor a cero, la compresión puede ser realizada sin mucha degradación de la imagen.

Sin un DCT, la compresión producida por el JPEG sería de 2 a 1, mientras que si el DCT es incluido en la metodología JPEG, la compresión será de 25 a 1. Esta metodología es simétrica, la compresión y la descompresión toman el mismo tiempo. La cantidad de computaciones para producir el resultado final es algo lento.

FIC (Fractal Image Compression)

La palabra fractal fue creada por Benoit Mandelbrot para indicar que una estructura fracturada posee similares formas a diferentes tamaños. La metodología FIC fue creada especialmente para el video. Esta metodología es asimétrica, la descompresión es mucho más rápida que la compresión. La tasa de compresión puede aumentarse sin sacrificar la velocidad de descompresión y la calidad de la imagen. Una imagen en software JPEG, requiere 41 segundos tanto para comprimir, leer o descomprimir la imagen. Mientras que para alguna aplicación que utilice tecnología FIC, requerirá de 8 minutos para compresión, pero sólo requerirá de 7 segundos para leer o descomprimir una imagen, es decir, 6 veces más rápido que con la tecnología JPEG. Muchos fabricantes de software podrían encontrar la compresión en fractales preferible para aplicaciones de multimedia donde accesos rápidos e imágenes de alta calidad son esenciales. Microsoft, por ejemplo, actualmente usa FIC en su enciclopedia Encarta, la cual es otra aplicación más de multimedia.

He hablado de que la tecnología FIC es más eficiente en aplicaciones de multimedia, pero ahora veamos en que consiste esta tecnología. El primer paso de esta metodología es dividir la imagen en regiones de dominio que no se

traslapen. La suma de todas las regiones de dominio debe cubrir toda la imagen original, cada región de dominio puede ser de cualquier tamaño o forma. En seguida el algoritmo establece una posible colección de regiones de rango y no necesariamente la suma de las regiones de rango deberán cubrir la imagen en su totalidad.

Por cada región de dominio el programa deberá seleccionar alguna región de rango que más se aproxime a la región de dominio, esto después de aplicar una transformación de afinidad a la región de rango. Las transformaciones de afinidad no sólo comprimen y deforman la imagen dentro de las regiones de rango, ellas también disminuyen el contraste y la brillantes de la imagen. Cada transformación es descrita por medio de los coeficientes de afinidad. Después que todas las transformaciones fueron completadas, un archivo con formato FIF (Fractal Image Format) es creado. Este archivo consiste de un encabezado con información acerca de la selección de regiones de dominio, seguido de una lista de todos los coeficientes de afinidad que fueron seleccionados por cada región de dominio. Este proceso genera un archivo que es independiente de la resolución de la imagen original. Al final del proceso usted tendrá una ecuación que representa a una imagen.

EL CHIP EN AYUDA DE LA MULTIMEDIA

Video

Muchos procesadores están apareciendo en el mercado, estos procesadores no son de propósito general. El objetivo de estos procesadores es el poder librar al CPU de ciertas tareas y que este efectué aquellas que sabe

hacer mejor, mientras que los procesadores especializados son utilizados para fortalecer aquellas áreas que necesitan más potencia de proceso.

Los chips en la multimedia pueden ser clasificados en dos grupos: chips de propósito fijo y procesadores programables. Los chips de propósito fijo son fáciles de diseñar y de fabricar, donde cada chip se diseña para realizar cierta función o algoritmo. Para aplicaciones diseñadas para el entretenimiento, los chips de propósito fijo son mucho más baratos que aquellos que son programables, pero la gran desventaja de estos sobre los procesadores programables es que sólo implementan un número limitado de funciones, mientras que los procesadores programables son más flexibles, debido a que pueden ser reprogramados para realizar diferentes operaciones y además se pueden adaptar en una forma aceptable a nuevos cambios o mejoras en las normas.

Algunos chips de propósito fijo están diseñados para resolver viejos problemas como la compresión de video, algunos de estos chips son fabricados por: la LSI Logic, la SGS-Thomson y la C-Cube.

La SGS-Thomson está diseñando sus procesadores para la solución de problemas de comprimir imágenes y video. El STI3220 implementa un algoritmo de búsqueda de imágenes en bloques de 16 por 16 pixels. El STV3200 implementa un algoritmo de compresión de imágenes con la ayuda de la metodología DCT.

La C-Cube fue la primera empresa en crear un chip de compresión

basado en la metodología JPEG, el CL-550 posee una interfaz que acepta pixels directamente desde cualquier fuente de video, sin la necesidad de ocupar un área de memoria intermedia.

El Vision Processor fabricado por la Integrated Information Technology, posee una arquitectura optimizada para comprimir imágenes y video. De tal forma que incluya a varias normas de compresión, el VP es programable en microcódigo, el cual puede ser cargado desde la misma computadora. El VP puede ser controlado por medio del CPU, sólo si el VP incluye la norma JPEG, sin embargo si el VP incluye la norma MPEG o la norma Px64, será necesario utilizar a un controlador de visión (VC, por Vision Controller), el cual incluye a un microcontrolador, un VLC y un sistema de interfaz con un área de memoria intermedia.

Intel también ofrece procesadores de esta clase, el 750-nextgen es un procesador que puede ser programado al nivel de microcódigo. Este procesador, aparte de servir como compactador, también puede realizar otras funciones. Funciona en una forma excelente cuando utiliza la filosofía MPEG. El 750-nextgen puede trabajar en tiempo real cuando se utiliza la norma Px64, esto es, ofrece una funcionalidad de 30 imágenes por segundo.

Audio

Un buen número de fabricantes de ADCs y DACs están tratando de integrar tantas funciones como sean posibles en sus procesadores para señales de audio. El AD 1849, de Analog Divices, contiene filtros analógicos, ganancia

programable, facilidades de conversión micro-law/A-law para compactar y descompactar el audio.

El chip toma la entrada en estéreo desde el nivel de línea y desde el nivel de micrófono. Dos salidas en estéreo y una salida no en estéreo. Puesto que el AD 1849 posee lo esencial de una interfaz de audio, usted puede implementar una completa sección de audio en un sistema de multimedia con sólo un chip.

Aunque la compresión de señales de video, imágenes y audio es la clave en el éxito de la multimedia, los fabricantes de chips tendrán que digerir otras tecnologías para poder integrarlas en estos chips, de tal forma que la multimedia siga teniendo presencia en los mercados del futuro.

HARDWARE Y SOFTWARE HACEN PODEROSOS SISTEMAS

Antes que nada, para poder tener un software de alto nivel es necesario poseer un hardware, también de alto nivel, para soportar al software. Si no existieran computadoras adecuadas para editar películas o música, voz, muchos proyectos se quedarían en el escritorio.

En la segunda mitad del año 1993, aparecieron computadoras personales tanto de Apple y Silicon Graphics, que son dos compañías que se han destacado en desarrollar productos de alta calidad. La integración de el video en computadoras personales ha empezado y dos compañías que tienen el liderazgo en este campo son Apple con su serie de computadoras AV Macintosh y Silicon Graphics con su estación de trabajo Indy.

Nunca la entrada, salida y el proceso de video había sido integrado estrechamente entre hardware y software. Al crear el video como cualquier otro tipo de datos, tales como, texto, gráficas, o sonido, estos sistemas han abierto nuevos y efectivos canales de comunicación entre individuos y grupos de trabajo.

De hecho, el objetivo de Apple y SGI es el de considerar sus sistemas como auxiliares para mejorar la comunicación orientada a negocios.

Mac audio, Mac video

La nueva serie de Macs, la Quadra 840AV y la Centris 660AV, no solamente consolida nuevas características en una computadora, también nos ofrece nuevas formas para trabajar en una oficina o con otras personas, ya sea en el mismo edifico o al otro lado del país. Servicios telefónicos le permiten a la Mac llamar a otras personas, dar mensajes, mandar y recibir faxes, y actuar como un módem para servicios en línea. El sistema de video (hardware) le permite a usted ver vídeos y manejar video conferencias sobre una red de comunicaciones, ya sea una red de computadoras u otro servicio de comunicaciones. Finalmente un sistema de reconocimiento de voz, le permite a usted guiar a la Mac por medio de instrucciones orales, mientras que una máquina de reconocimiento de texto permite que los documentos puedan ser leídos en voz alta por medio de este sistema, lo cual le permite a usted realizar otras tareas mientras escucha la lectura del documento. Veamos algunos de los elementos que hacen posible esto (sólo para la Quadra 840AV):

Quadra 840AV:

1. 40-MHz 68040.
2. 8 MB RAM a 60 ns.
3. Ampliable a 128 MB de RAM.
4. Tres ranuras para NuBus 90.
5. 230 MB de disco duro.
6. 2 MB en ROMs.
7. Ethernet.
8. 66.67-MHz DSP
9. Escalador digital de video.
10. Video ADC.
11. Un controlador de video.
12. DRAM, VRAM, entre otros.

Se pude conectar una videograbadora a la Mac (Quadra 840AV o la Centris 660AV), para que usted pueda ver un video, o pueda usar estas AV Macs para tener una video conferencia con otro usuario en una Ethernet LAN. Para lograr esto, sin embargo, la video imagen tendrá que ser pequeña (tipicamente 160 por 120 pixels), las imágenes por segundo de la video conferencia serán de alrededor de 10 a 15.

La tecnología que usa la Apple para este par de AV Macs es lenguaje natural, esto es debido a que el usuario puede dictar ordenes oralmente. Plain Talk es una tecnología independiente del usuario y la cual le llevo a la Apple's Advanced Technical Group cinco años de investigación. Independiente del

usuario significa que Plain Talk no requiere de un entrenamiento por parte del operador, para que su software de reconocimiento de voz funcione correctamente.

La Indy de SGI

Silicon Graphics ha creado una nueva estación de trabajo llamada simplemente Indy, la cual está basada en el procesador Mips R4000PC. La Indy también es capaz de recibir, procesar y enviar salida tanto de sonido como de imágenes (video) ya sea en forma digital o analógica.

El micrófono que viene con la Indy no sólo sirve para integrar mensajes orales en documentos, también se puede utilizar para darle ordenes al Indy.

El procesador y reconocedor de voz es una implementación desarrollada por Scott Instruments. SGI ofrece una biblioteca para que los programadores implementen reconocimiento de voz en sus aplicaciones.

La Indy no viene con una unidad de discos flexibles normal, pero en cambio ofrece una unidad de discos flexibles ópticos, de 3.5" y de 20 MB, que puede leer y escribir tanto en discos flexibles para Mascintosh y PC, también puede leer y escribir formatos tar y cpio de UNIX.

Usted puede suministrar video en forma digital por medio de la cámara IndyCam (viene con la máquina) o por otro medio. También usted puede suministrar video en su forma analógica, ya sea por un aparato de video, un camcorder o una video cámara.

El sistema de audio es localizado en el Motorola DSP 56001, el cual posee entrada y salida en estéreo. La Indy posee puertos para: salida de audio en forma digital y analógica (para audífonos), entrada en forma analógica (para un micrófono). Al igual que la IndyCam, los audífonos y el micrófono también vienen incluidos con la computadora.

LA TECNICA HACE AL SOFTWARE

SISTEMAS HIBRIDOS

Al ligar sistemas orientados al conocimiento con otras tecnologías, como redes de neuronas, puede aumentar funcionalidad y confiabilidad. Un sistema híbrido puede realizar un trabajo cuando una simple tecnología no puede. El truco es crear un sistema con las características adecuadas para resolver el problema en el que usted está trabajando.

Los sistemas expertos se están convirtiendo en verdaderos dinosaurios. Ellos son la última generación de sistemas de alto nivel. Usando la tecnología orientada al conocimiento en combinación con la tecnología basada en el razonamiento (CBR, case-based reasoning), redes de neuronas, o algoritmos genéticos son la nueva tendencia para resolver problemas en ambientes empresariales. Los sistemas híbridos toman ventaja de las mejores características de cada tecnología. Por ejemplo, con sistemas basados en el razonamiento y una potente tecnología de búsqueda, se pueden crear sistemas de búsqueda de información que pueden ofrecer la mejor opción (u opciones).

La tecnología CBR provee de inteligencia a un sistema, mientras que la tecnología KBS (knowledge-based systems) se basa en el conocimiento de seres humanos y en experiencias pasadas de si mismo. Las redes de neuronas pueden agregar más precisión y flexibilidad a un sistema basado en el conocimiento. Y todo esto mezclado con sonido, animación, gráficas en tres dimensiones en sistemas inteligentes, el resultado será un producto de altísima calidad. Es obvio que cuando se unen fuerzas el producto será mucho mejor y más flexible que cuando se trabajaba con una sola tecnología (sistemas dinosaurios). Ligando diversas tecnologías se pueden crear sistemas que tomen decisiones en una forma que lo hacen las personas, pero en una forma más rápida, eficiente y confiable. Un sistema resultante de esta reciente tendencia para crear aplicaciones es Cyc, creado por Doug Lenat, un científico lider en Microelectronics and Computer Technology establecida en Austin Texas. Esta aplicación está basada en un KBS (Knowledge-Based System) con una interfaz en lenguaje natural.

Un KBS es una aplicación que puede almacenar, extraer y analizar vastos almacenes de conocimiento y datos. Los KBS son extensiones de los sistemas expertos, los KBS vieron la luz a mediados de los ochentas. Hoy en día son utilizados para diagnosticar fallas en maquinarias, en pronósticos de tendencias comerciales, calendarización de la producción y diseño de nuevos productos.

Una nueva generación de aplicaciones está ayudando a las compañías a trabajar en una forma más inteligente y de esta forma tomar ventaja de su vasto conocimiento y experiencia. Las aplicaciones son llamadas KBS (Knowledge-Based Systems).

Un KBS o varios, se encargan de capturar el conocimiento y experiencia colectiva de una empresa, también distribuyen esto a través de una red. Por ejemplo la Amarican Express (New York, NY) almacenó políticas y procedimientos de la empresa en su KBS, para asistir a sus autorizadores de crédito y ejecutar transacciones en una forma más rápida.

Nuevos sistemas inteligentes combinan KBS con otras tecnologías, tales como redes de neuronas, CBR (Case-Based Reasoning), algoritmos genéticos, realidad virtual y multimedia.

Una de las ventajas de los sistemas híbridos en sistemas empresariales es el de coordinar todas las funciones de todos los departamentos en una empresa. De esta forma se mejorará el servicio a clientes a un reducido costo y el desarrollo de software de alta calidad será más rápido.

Los KBS evolucionaron de una base muy solida la cual fueron los sistemas expertos, los cuales fueron diseñados para incluir el conocimiento de las personas en programas en la forma de IF ... THEN. Los sistemas expertos de principios de la década de los ochentas, probaron ser difíciles de construir debido a que era necesario lograr capturar todo el conocimiento de algún experto en cierta área. Los sistemas expertos también fueron difíciles de mantener debido a sus grandes bases de reglas, tenían poca organización. La mayoría de los sistemas expertos eran cargados en una estación de trabajo y funcionaban en una forma individual, es decir, no trabajaban en una forma integral con otras tecnologías.

A mediados de la década de los ochentas, los investigadores acoplaron a los sistemas basados en reglas con sistemas basados en estructuras y con la filosofía orientada a objetos. Esta combinación ayudo a representar complejas estructuras de datos y sus relaciones. Este acoplamiento produjo como resultado una nueva generación de herramientas y aplicaciones, llamadas KBS (Knowledge-Based Systems).

Con la habilidad de representar, manejar, analizar y procesar complejas bases de conocimiento. Sistemas basados en KBS son el producto de técnicas de alta calidad y las cuales ofrecen una nueva visión de la productividad. KBS es en si un superconjunto de tecnología orientada a objetos.

Investigadores y fabricantes de software están combinando herramientas KBS con otras tecnologías. Aunque los sistemas híbridos se encuentran en una etapa de experimentación, resultados preliminares indican que tal acoplamiento produce la solución de problemas más complejos.

Por ejemplo si usted combina la multimedia y realidad virtual con tecnología KBS, usted creará una potentísima interfaz de usuario para el KBS. Los programadores podrán extraer recursos de multimedia como si fueran cualquier tipo de datos (texto, enteros, etcétera) y crear agentes inteligentes en una realidad virtual. Como resultado, usted conseguirá una aplicación que parecerá tener vida.

Cuando un sistema combina tecnologías KBS y CBR (Case-Based Reasoning), este sistema puede manejar eficientemente un amplio rango de

información. El CBS le permite al sistema híbrido manejar experiencias (casos) de seres humanos. Esto le permite al sistema realizar un amplio razonamiento a través de estos casos.

Sistemas híbridos compuestos de tecnología KBS y CBR están probando ser realmente productivos en manejar ciclos de vida de producción (coordinando el diseño, mercadeo y el inventario de productos). Con su rica capacidad de representación, un KBS le permite a una compañía almacenar y obtener decisiones de producción que podrán servir de mucho para el futuro. DEC, Boeing y General Motors son unas pocas de las empresas que usan tecnología KBS para coordinar tales tareas.

La nueva generación de KBS y otras tecnologías están dando a las computadoras la inteligencia para adaptarse en ambientes laborales, acceso a vastos bancos de información y ayudar en la realización de tareas en una forma más inteligente. El reto es el de aplicar correctamente estas nuevas tecnologías en problemas de alta complejidad.

Los KBS en si son potentes, pero no dejan de tener ciertas limitaciones. Ellos no pueden manejar eficientemente grandes aplicaciones, su razonamiento no es adaptivo y su funcionalidad no mejora mucho con la experiencia. En adición ellos requieren demasiada entrada por parte del ser humano. Estas desventajas pueden ser eliminadas al acoplar un KBS con redes de neuronas y algoritmos genéticos.

Las redes de neuronas consisten de redes paralelas, o grupos, altamente

interconectadas unidades de proceso. Las redes de neuronas son bien apropiadas para el reconocimiento de patrones, traslación a lenguas extranjeras, procesos de control, e implementaciones de ciertas tareas industriales en paralelo.

Pero las redes de neuronas están evolucionando en estos momentos, y por lo tanto, también tienen defectos. Por ejemplo, las redes de neuronas necesitan la entrada de gran cantidad de casos probados para poder obtener resultados más precisos, y aún los resultados son difíciles de explicar. Una red de neuronas necesita un extensivo entrenamiento. El número de neuronas limita la capacidad de almacenamiento en una red de computadoras, absorbe muchos recursos de almacenamiento.

A pesar de todo lo anterior, la lógica naturaleza de un KBS complementa bien la naturaleza numérica, asociativa, de autoaprendizaje y una biológica naturaleza de las redes de neuronas. Larry Medsker, un profesor en ciencias de la computación en la Amarican University (Washington, DC), afirma que la combinación de redes de neuronas con KBS ofrece una mejora en muchas áreas, incluyendo la degradación de sistemas, razonamiento explícito e implícito, incremento en el aprendizaje, confiabilidad y flexibilidad.

Existen diferentes arquitecturas para integrar estas técnicas y de esta forma obtener ciertas ventajas, una es el modelo debilmente acoplado, el cual usa a un KBS y a una red de neuronas, pero por separado, y la única forma de comunicación es por medio de archivos de datos.

Científicos que trabajan para la Computer Sciences, que provee de programas para computadora para la NASA, han usado esta arquitectura en el Goddard Space Flight Center. El proceso usa redes de neuronas para filtrar datos de pobre calidad, los cuales provienen de satélites espaciales. Los datos resultantes son enviados a un KBS para ser clasificados.

Otra arquitectura es el modelo fuertemente acoplado. En este modelo, un KBS y una red de neuronas trabajan, también, por separado. Pero esta vez la comunicación es por medio de parámetros y no por archivos de datos.

La Loral Aerospace, en Houston Texas, hace uso de esta arquitectura. Matthew Hanson, gerente de ingeniería de software y desarrollo, y Robert Brekke, gerente de integración, verificación y pruebas, desarrollaron WMES (Workload Managemente Expert System). El sistema usa una red de neuronas para determinar las necesidades en grupos de trabajo, basado en la disponibilidad de grupos de trabajo, habilidades de los trabajadores y fechas de inicio de los proyectos. Un KBS interactua con la red de neuronas para hacer estimaciones de los recursos (humanos y materiales) que serán requeridos.

Una tercera arquitectura es el modelo totalmente acoplado, el cual usa una estructura de datos compartida. Sistemas con procesamiento distribuido en paralelo pueden utilizar esta arquitectura.

El acoplamiento de KBS con redes de neuronas tiene también ciertas limitantes. La complejidad para desarrollar y mantener este tipo de aplicaciones, requiere de grupos de trabajo con muchísima experiencia y

conocimiento de esta área de la computación. Aunque el mercado ofrece excelentes aplicaciones como Nuex Shell de Charles River Analytics, en Cambridge Masachusets, aún no existen suficientes aplicaciones de esta naturaleza. Finalmente se necesita de un sistema de multiproceso y con una aceptable funcionalidad.

CBR (Case-based reasoning) le permite a un sistema almacenar pasadas experiencias o situaciones como casos, analiza y procesa los datos, y sugiere soluciones (si hay más de una) para un problema. Un sistema CBR posee dos principales componentes: una base de casos y un solucionador de problemas.

Una base de casos contiene descripciones de problemas resueltos y de problemas no resueltos. Un solucionador de problemas consiste de un recuperador de casos y de un razonador de casos. El recuperador de casos identifica la información (ya sea por el método de búsqueda del vecino más cercano, o usando índices u otras técnicas) en la base de casos, los datos más apropiados son presentados al razonador de casos. El razonador de casos examina los casos, y con la ayuda de el dominio de conocimiento, realiza adaptación, síntesis, o predicción.

Una razón por el creciente interés en desarrollar aplicaciones CBR es la mayor disponibilidad de herramientas CBR. Un ejemplo es Archie, una herramienta CBR desarrollada por Janet Kolodner, Ashok Goel, Erik Domeshek y sus colegas en el Georgia Institute of Technology (Atlanta), para ayudar a los arquitectos a solucionar problemas de diseño. Remind de Cognitive Systems, CBR Express de Inference (un par de compañías

californianas) y Esteem de Esteem Software (Indianapolis, IN) son unos paquetes que hacen uso de esta tecnología.

Otra técnica disponible son los algoritmos genéticos. Un algoritmo genético mantiene una lista de posibles soluciones para un problema. Basado en si o no las previas soluciones fueron exitosas, las más apropiadas soluciones no sólo sobreviven sino también intercambian información con otras candidatas para así formar otra solución. The Navy Center for Applied Research in Artificial Intelligence en el Naval Research Laboratory (Washington, DC) ha estado usando algoritmos genéticos para ayudar a algunos robots a desarrollar su capacidad de aprendizaje y adaptación.

Usted puede integrar algoritmos genéticos con métodos basados en reglas. Los algoritmos genéticos son muy útiles para introducir inteligencia, resolución de conflictos y clasificación.

Aplicación de los Sistemas Híbridos en el Hogar

Imagina el día en el cual puedas tener acceso a vastas bibliotecas de video clips relacionados con casi cualquier tópico de interés y con la posibilidad de que asistentes informáticos con inteligencia puedan atenderte, obtener la información que buscas y explorar áreas de interés. Esto es exactamente el trabajo de experimentación que está realizando la Northwestern University's Institute of the Learning Sciences (Evanston, IL), dirigida por Roger Schank.

El grupo de trabajo de Schank ha desarrollado una inteligente herramienta que mezcla tecnologías tales como orientada a objetos, CBR (Case-Based Reasoning) y multimedia (video, sonido, aunque las técnicas orientadas a la inteligencia son también parte de este enorme conjunto conocido por todos nosotros como multimedia. Este conjunto es tan grande que algunas personas no saben que es lo que pertenece y que es lo que no pertenece a la multimedia. Pero para este propósito, designemos aquello perteneciente al audio y al video , como multimedia), esta herramienta es llamada Ask, la cual puede ordenar y obtener video clips (de una duración de uno a dos minutos) y texto en relación al tema deseado. Si usted está interesado, digamos, en la localización de algún incidente en la mitología, usted podría obtener un mapa, digamos de Grecia, seguido por un video clip de algún lugar de Grecia, y si usted es atraído por algún museo, alguna plaza o simplemente es usted atraído por una playa, usted podría pedir información de como viajar a ese lugar. De hecho Ask funciona como una enciclopedia de distintos tópicos, inclusive la guerra, si usted está interesado en obtener información acerca de la Guerra: Tormenta del Desierto, usted la obtendrá en forma de video clips y texto.

El Institute of the Learnig Sciences (ILS) es pionero en estos conceptos y los cuales pueden ser empleados en otros campos, como por ejemplo el mundo de los negocios. El grupo de Roger Shank ha creado aplicaciones usando a Ask como parte central, estas aplicaciones incluyen servicios financieros y administración de empresas entre otros.

Otra aplicación de ILS, es Creanimate, el cual le enseña a los estudiantes biología. Creanimate usa inteligentemente vídeos para enseñar acerca de la vida salvaje y de características de los animales que viven en estos ecosistemas.

Sistemas experimentales alrededor de una base de conocimientos están surgiendo por todas partes. Por ejemplo un fabricante esta investigando en sistemas KBS (Knowledged-Based Systems) con multimedia como una forma de obtener consejos orientados a los negocios. Usando estos tipos de sistemas, un ingeniero de diseño puede obtener un análisis de algún producto de un competidor y los requerimientos para el diseño. Esta compañía ha desarrollado prototipos en esta área usando normas industriales (por ejemplo, distintas plataformas de computadoras y dispositivos de video). Tales sistemas podrían ser usados vía una red de computadoras, para resolver problemas de tipo empresarial.

La Andersen Consulting (Chicago, IL) ha sido una innovadora en aplicar tecnologías de alto nivel como lo son KBS, CBR, multimedia y síntesis de voz. La Andersen Consulting está actualmente trabajando en un sistema de soporte, el cual podrá ayudar a oficiales de préstamos a realizar llamadas, estructurar tratos (negocios) y mantener una relación de clientes. La aplicación es llamada TIM (Total Information Management). "Nosotros podemos hacer a una computadora parecer desde una revista, un salón de clases, un supermercado o una cabina de avión. Esto vuelve a una computadora en un camaleón", esto lo afirma Joe Carter de Andersen Consulting.

Andersen Consulting está trabajando en un sistema de realidad virtual, usando multimedia para crear un supermercado electrónico. Usted puede "caminar" a través de los almacenes en un espacio de tres dimensiones usando un ratón. Agentes (informáticos) inteligentes podrán aconsejarle, digamos, en comprar un elegantísimo traje para los negocios. Un agente podría decirle

acerca de las tendencias en la moda del vestir. Otro agente podría funcionar como un sastre. Otro podría decirle, francamente, que ropa le sienta bien.

El equipo necesario para simular al supermercado sería instalado en su propio hogar, el cual incluiría a una computadora personal, un convertidor conectado a su televisor, un aparato CD-ROM y un puerto conectado a su teléfono. El sistema está en la etapa de pruebas (1993) y el cual será puesto en marcha en algunos hogares de Estados Unidos en el año de 1994.

El problema más grande es el ancho de banda de las líneas telefónicas. La aplicación requiere un sistema de líneas, hechas con fibras ópticas, dichos sistemas de comunicaciones están ciendo instalados a lo largo y ancho de Los, Estados Unidos.

En Fakespace (Menlo Park, CA), se están usando técnicas tales como KBS y realidad virtual con robots. Unas cámaras son montadas en los robots. Un KBS (Knowledge-Based System) captura los datos enviados por el robot y checa estos contra su modelo. Si existen discrepancias, el KBS detecta la causa. Por ejemplo, si material radioactivo ha sido lanzado al aire, una computadóra puede diagnosticar la causa y recomendar o iniciar las acciones para corregir la falla.

Aunque la integración de sistemas orientados al conocimiento con tecnología de realidad virtual y multimedia se encuentra en su etapa experimental, la implementación de sistemas con inteligencia, análisis y solución de problemas es realmente tremendo. Usted se podría sentir como si estuviera usando algo que perteneciera a la ciencia ficción.

LA VOZ Y LA PLUMA COMO ALTERNATIVAS PARA EL RATON Y EL TECLADO

Los noventas serán recordados como la década en la cual los usuarios de computadoras empezaron a usar el diálogo y la escritura como medios de comunicación entre computadoras y personas. El diálogo y la escritura se encargarán de derribar las barreras impuestas por el teclado y el ratón, todo por una comunicación más natural entre los usuarios y las computadoras.

Una de las más obvias aplicaciones para la voz y la pluma son las notas, si las notas diarias que se intercambían entre grupos de trabajo. Por ejemplo suponga que usted se encuentra leyendo un borrador de otra persona en la pantalla del monitor de su computadora. Si su computadora y su software aceptan entrada tanto de una pluma y de un micrófono, usted podría escribir algunas correcciones, trazar alguna figura a mano o un pequeño croquis o grabar sus comentarios acerca del trabajo de su compañero o su secretaria en el mismo documento. Más tarde, cuando el autor escuche los comentarios (que usted dejó grabados en el documento) el o ella podrá ver las correcciones que usted hizo.

La computadora simplemente captura y almacena en forma digital la señal de la pluma (escritura) y la señal del micrófono (voz), asociando cada una a alguna parte del documento. Los beneficios son demasiados, desde que ambos usted y el autor de el documento (regresando al ejemplo) pueden ahorrarse mucho tiempo con el diálogo para la comunicación entre ustedes e indicar algo como "este párrafo necesita más detalle y la gráfica no me gusta en

azul, he dibujado otra con diferentes características, espero que ...." y al trazar una flecha entre dos palabras o párrafos, se podría indicar que se cambiaran, de hecho, usted podría trazar a mano algunas figuras para resaltar cualquier cosa. Algunas aplicaciones con estas características están comenzando a aparecer.

Considere ahora a una hoja de cálculo. Es fácil imaginar el introducir datos ya sea hablado o escrito en las celdas. Por ejemplo, para introducir algún número realmente grande, usted podría dictarle el número a su aplicación en lugar de voltear repetidas veces hacia su cuaderno y copiar el número dígito a dígito. Para introducir algún nombre que le pueda ser extraño al reconocedor de voz, usted podría escribirlo en lugar de deletrearlo en voz alta (esto sólo en casos que así lo ameriten), esto es debido a que algunas pronunciaciones de ciertas palabras poseen un sonido similar y esto puede confundir al reconocedor de voz.

Sin embargo, es difícil señalar alguna zona de una hoja de cálculo tanto con el uso de la voz como con el teclado. Una buena solución sería señalar la celda deseada con la pluma, una vez seleccionada la celda deseada usted podría comenzar a dictar la información correcta en tal celda. Algunas veces sería más fácil escribir, en otras lo sería el señalar, mientras que en otras sería más útil algo hablado. Lo que estoy intentando de explicar, es que cada filosofía de entrada se complementa con las demás.

Usted no sólo podría utilizar la pluma para señalar el lugar de entrada, también la podría utilizar para activar procesos, por ejemplo, activar el micrófono.

Un importante objetivo y el cual ha llevado mucho tiempo de investigación es la de generar texto por medio de la voz. El construir un sistema con estas características, implica que los mecanismos de edición y control sean realmente eficientes y flexibles.

La tarea de reconocer el idioma es en verdad tremenda, los sistemas que desarrollan tal acción, llevan a cabo un detallado proceso. Los primeros sistemas, que están apareciendo, fallan frecuentemente en la traducción del lenguaje hablado al texto, lo cual implica que el usuario tendrá que realizar ciertas correcciones a sus documentos después del dictado. El problema no es sólo en los algoritmos, también es debido a palabras con un sonido parecido y el vacilar un poco al fabricar los sonidos (uno muchas veces al hablar lo hacemos a un volumen bajo, luego subimos nuestro volumen al hablar, es decir, no mantenemos un volumen constante. También hay que tomar en cuenta el tartamudeo, el atropellamiento de palabras al hablar [hablar muy rápido]). Los problemas son demasiados, lo cual ayuda al reconocedor de voz a confundirse. Por lo que la tarea para los fabricantes de software no es nada fácil, la primera generación ya está aquí pero todavía dista de los ideales.

El usar una pluma para corregir equivocaciones es de gran ayuda. Por ejemplo, conforme usted está hablando y detecta algún error, usted podría tachar la palabra errónea, una lista de posibilidades podría aparecer y de allí usted se encargaría de escoger la palabra más apropiada, o simplemente, borrar el error y escribir la palabra correcta y seguir de esta forma con el proceso de dictado.

El uso de la voz y de la pluma es alternado conforme usted trabaja con su sistema. Por ejemplo, usted podría utilizar la pluma para marcar un párrafo o alguna palabra; después usted señalaría con la pluma el lugar en el documento en el cual a usted le gustaría que apareciera el texto marcado, usted le ordenaría al sistema, por ejemplo: "mueve el texto marcado a la nueva posición". Tal mescolanza del lenguaje hablado y de la escritura implica una intimidante integración de ambas tecnologías (el habla y la escritura).

El combinar el lenguaje hablado con la escritura podrá realzar la comunicación del ser humano con las computadoras. Los resultados son obvios en aplicaciones más fáciles de usar en una forma natural.

El Reconocimiento de la Voz con la Ayuda de las Computadoras

Una computadora, en estos días, puede almacenar recados hablados o actuar como un teléfono. Esta también puede analizar la voz de algún usuario con fines de seguridad. De todas las funciones posibles, el reconocimiento de voz posee un gran potencial para cambiar la forma con la cual usted se comunica con su computadora. La tecnología está produciendo una nueva interfaz humana.

Hasta la fecha, las aplicaciones en esta área han sido confinadas a un pequeño circulo. Una nueva generación de software está apareciendo en forma comercial, dos de las principales características que poseen estas aplicaciones es que poseen un bajo precio de venta y que además pueden correr en una PC normal. Ahora el gran reto para fabricantes de software es moverse desde esta básica tecnología a un nivel de aceptables aplicaciones.

El hardware básico para alguna computadora y así poder utilizar este tipo de aplicaciones es un micrófono y un chip ADC (un convertidor analógico-digital). Un DSP (Digital Signal Processor) es no estrictamente requerido, pero si sería deseable su presencia en la computadora, éste al menos ayudará a doblar la potencia de proceso para el reconocimiento de voz. DSPs son mucho mejores que los microprocesadores de propósito general para desarrollar este tipo de tareas.

Los fabricantes de sistemas de voz seleccionan diferentes tipos de DSPs, dependiendo en necesidades específicas. Algunas DSPs sirven para ciertos propósitos, tales como el actuar como un fax o un módem, comprimir señales de video, o el procesado de la voz, mientras que el CPU es librado de realizar tales tareas y podrá seguir haciendo sus actividades cotidianas.

El software para el reconocimiento de voz puede operar en señales digitales que representan el habla del usuario para realizar dos tipos de tareas: convertir palabras habladas en texto o en comandos para la computadora. Este tipo de tecnologías son realmente útiles, especialmente para personas que poseen algún impedimento físico, lo cual les permite realizar sus labores en una forma más eficiente.

Interactuando con la computadora vía voz es mucho más fácil que usando el teclado. Una aplicación obvia es el poder utilizar el teléfono para darle instrucciones a la computadora. Supongamos que usted se encuentra en su coche y se encuentra perdido en medio de un inmenso tráfico y usted necesita realizar cierta actividad importantísima, como el elaborar un reporte de la junta

del día de ayer. Sin más preocupaciones usted podría tomar su celular y comenzar a darle ordenes a la computadora. El primer paso sería abrir un documento y empezar a dictarle a la máquina las ideas que usted lleva en la cabeza. Inmediatamente usted podría ordenarle a la computadora que le lea el reporte para verificar que no tenga errores, una vez satisfecho con su reporte usted podría indicarle a su máquina que envié por correo electrónico el reporte a las personas adecuadas. Inclusive podría dejar recados hablados para su secretaria, lo cual implica que el tráfico de vehículos no lo ha detenido en su trabajo y así usted llegará a su oficina sin tantas preocupaciones y más tranquilo y aún le sobrara tiempo para tomarse su clásico café. Esto provocará que la productividad en todos los ámbitos aumente considerablemente.

El reconocimiento de voz se encuentra en estos momentos en una etapa embrionaria, con muchos defectos, y los errores que los reconocedores de voz cometen son numerosos, ahora el diálogo sobre una vía telefónico provoca más errores.

La calidad de la línea varia significativamente con los celulares, teléfonos inalámbricos, teléfonos normales y lo que es aún peor, los nefastos teléfonos públicos. Las llamadas telefónicas, muchas veces son hechas desde ruidosos lugares públicos, lo cual agrega un ruido de fondo cuando una persona habla por un teléfono, lo cual le complica la vida al reconocedor de voz.

La mayoría de las aplicaciones que intervienen en los diálogos vía telefónica, deben ser independientes del usuario y los reconocedores de voz deben ser lo suficientemente flexibles con la voz de las personas (voz de

hombre, mujer o niño), con los dialectos regionales y los diferentes acentos regionales y extranjeros. Un reconocedor debe ser capaz de detectar cuando una persona desea interrumpir la comunicación con la computadora.

Conforme la precisión en el reconocimiento de voz mejore, nuevas y mejores aplicaciones aparecerán. Acceso telefónico a servicios tales como: información de la sección amarilla y a asistentes de directorios telefónicos serán los primeros pasos en la creación de este tipo de servicios telefónicos. Todos estos problemas encontrados en las aplicaciones serán solucionados y la gente podrá comunicarse, vía telefónica, con las computadoras en una forma fluente y fácil, tal como si estuvieran hablando con otra persona.

La Pluma en la Computación

La pluma ha experimentado un duro comienzo y la credibilidad de su eficacia en el medio está en duda, esto como resultado del escaso campo de aplicación y del malísimo software que existía para implementar el reconocimiento de la escritura. Todo esto sucedía en 1992, en este año la gente se preguntaba si esto iba a resultar o si las aplicaciones en base a la pluma sobrevivirían su infancia. En 1993 comenzaron a aparecer PCs, con la opción de utilizar a la pluma como un periférico más, tal como la NCR 3125. La característica de estas PCs era de una regular capacidad de almacenamiento en disco y de una pésima calidad de exhibición.

La pluma es ahora reconocida, por algunos sistemas operativos, como un dispositivo más de entrada, por ejemplo veamos algunos de estos sistemas operativos:

1. En computadoras de escritorio: OS/2 2.1 de IBM.
2. En laptops: Power-Book de Apple.
3. Sistemas híbridos: Convertible de Grid.
4. En PDAs (Personal Digital Assistants): GEOS de Geo Works, Newton de Apple y PenPoint de Go Corporation (son los sistemas operativos más importantes en tecnología para PDAs).

El reconocimiento de la escritura a mano con la ayuda de la pluma en sistemas PDA es muy mala, en algunas PDAs el reconocedor de escritura necesita que el usuario lo entrene para poder reconocer su forma de escribir. Un verdadero problema surge cuando otra persona desea utilizar la misma PDA, esto es debido a que el sistema no reconocerá el estilo de escritura de este nuevo usuario y por lo cual no entenderá muchas palabras de éste. Por ejemplo las PDAs (la serie MessagePad), basadas en el sistema operativo Newton, necesitan ser entrenadas y con el tiempo se adaptan automáticamente a su forma de escribir.

En algunas computadoras la principal característica de una pluma es de señalar y no de escribir (en las PDAs la pluma sustituye totalmente al teclado y se utiliza para escribir, señalar, seleccionar y arrancar aplicaciones que también tienen como entrada principal a la pluma), para manipular por completo la interfaz de usuario. Cualquier empleado que su trabajo le implique estar caminando o estar parado la mayor parte del tiempo es necesario que posean un sistema basado en la pluma. Por ejemplo, la recolección de datos, digamos, durante inspecciones a pie o con un la ayuda de algún vehículo, necesita de una

computadora ligera, basada en alrededor de una pluma (para llenar posibles formas de inspección y escribir posibles comentarios) y con baterías de larga duración.

Ahora para computadoras más grandes, de escritorio, la pluma, el ratón, el teclado y la voz podrán coexistir en las computadoras que están por llegar, de hecho ya existen algunas pero su funcionamiento no es el ideal todavía.

CAPITULO 2

REQUERIMIENTOS

En este capítulo se hablará de el porque de un sistema de hipertexto para la creación de información electrónica, de el porque de la selección de la tecnología orientada a objetos y el porque de la utilización de Turbo C++ 3.0 para Windows.

Hyptextw Como un Sistema de Hipertexto

El objetivo inicial para la implementación de un sistema de hipertexto era la de poder ofrecerle al escritor una herramienta para crear sistemas de información de la clase de hipertexto y de esta forma poder presentar sus escritos en una forma diferente a la acostumbrada. Los usuarios de un sistema de hipertexto tienen la posibilidad de presentar su información creada en la pantalla de la computadora y en una forma dinámica, en vez de en el papel y en forma estática. Hyptextw ofrece todo lo anterior además de la posibilidad de imprimir en papel la información requerida por un usuario.

Hyptextw es una aplicación para Windows y posee características ya comunes de las aplicaciones que funcionan sobre el sistema operativo Windows, tales como: el manejo de ventanas, menús de cortina, el uso del ratón, botones de presión, cajas de verificación, botones de radio o cajas de diálogo. Hyptextw es una aplicación fácil de usar y con la cuál el usuario podrá crear aplicaciones en una forma rápida y eficaz.

Hyptextw está diseñado para que lo usen diferentes tipos de personas, desde algún estudiante hasta un escritor. Con Hyptextw se pueden crear toda clase de trabajos, por ejemplo una aplicación muy útil sería la creación de tutores programados. Un tutor programado podría tratar de cualquier área de la cultura, por ejemplo un tutor enfocado al álgebra. El tutor llevaría de la mano a un estudiante, en dado caso que al estudiante tuviera alguna duda en algún tópico, éste podría regresar a algún tópico anterior para reafirmar algunos conocimientos o seleccionar un tópico de alguna lista de opciones, la cual es mostrada por el tutor. Los saltos entre tópicos, vendrían siendo enlaces y desde una perspectiva general, el recorrido del tutor por todos los tópicos tendría la forma de un árbol. Un tutor debe estar construido de una forma didáctica y programado de una forma tal que le sirva al estudiante como una herramienta más de su auto aprendizaje. Además de tutores se podrían crear manuales de toda clase, por ejemplo, manuales de algún sistema de software, de alguna máquina o un recetario, de tal forma que la información esté disponible en forma electrónica. Lo cual nos permite tener información de todo tipo en discos flexibles y no en papel, lo cual hace que la información disponible sea más barata.

Empleo de el Enfoque Orientado a Objetos

A lo largo de estos últimos años, el uso de un enfoque orientado a objetos ha demostrado ser una herramienta poderosa para el desarrollo de aplicaciones de todo tipo. De hecho, este enfoque se ha llevado hasta la raíz de un sistema, el sistema operativo y un ejemplo muy claro es el sistema operativo Newton Intelligence de la Apple y el sistema operativo Penpoint de Go

Corporation (ambos sistemas operativos para PDAs). Esto es un indicativo de que todos los sistemas operativos del futuro y todas las aplicaciones que corran sobre estos, van a ser desarrollados utilizando el enfoque orientado a objetos. Las aplicaciones desarrolladas bajo este enfoque poseen una alta integración en funcionalidad de todas sus partes, un código compacto (únicamente el código necesario), partes remplazables, son fáciles de actualizar, fácil de mantener y más fáciles de crear, todo esto en comparación con el enfoque tradicional, (porque en la creación de sistemas nada es fácil) lo que hace el enfoque orientado a objetos es allanarnos más el sinuoso camino de el desarrollo de software.

Por naturaleza, cuando se crean aplicaciones complicadas siempre se trabaja en grupos. Cuando todos los elementos de algún grupo de trabajo poseen un mismo enfoque, la comunicación entre ellos se vuelve más fácil debido a la uniformidad y claridad de esta técnica. Esta uniformidad es encontrada en todas las etapas del desarrollo de aplicaciones. Con la tecnología tradicional es casi imposible mantener una uniformidad en todas las etapas, todo empieza desde la selección de herramientas de análisis, diseño y desarrollo (esto incluye al lenguaje seleccionado para la implementación, por ejemplo un programador conoce muy bien COBOL, otro conoce más o menos Basic, mientras que otros conocen Pascal y otros C).

En la actualidad muchas personas se niegan al cambio, debido que todo cambio implica un esfuerzo y mucha gente no está dispuesta a realizar ese esfuerzo. Tarde o temprano se tiene que dar este cambio pero en una forma total, de tal forma que todos dialoguemos en forma natural (objetos) y no en

una forma artificial. Todos necesitamos adaptarnos a los cambios de nuestra época.

La filosofía orientada a objetos por si sola no va a resolver nuestros problemas, nosotros debemos darle un uso adecuado para que de esta forma funcione y nuestras aplicaciones desarrolladas bajo esta filosofía realmente trabajen y resuelvan nuestros problemas, al decir nuestros problemas, se está incluyendo a todas las áreas, debido a que las aplicaciones computacionales han invadido ya todas las áreas: la salud, la educación, los negocios, la guerra, en las investigaciones, en los vuelos terrestres y espaciales, en los deportes, en las comunicaciones, en el entretenimiento, etcétera.

Todos nosotros estamos acostumbrados a los objetos desde que nacemos, todos los días estamos rodeados por objetos. Todo mundo sabe que algún objeto tiene características especiales y que puede tener ciertas funciones y que otros objetos están integrados de tal forma que forman parte de otro objeto. Así como un auto que se encuentra formado de otros objetos que a su vez están formados de otros objetos que a su vez están formados de otros objetos y así sucesivamente. Una sistema de software es también un objeto, quizás no en forma física (puesto que no es tangible), y por lo cual debería de estar formado de otros objetos que poseen ciertas características y alguna función. Muchas de las aplicaciones creadas bajo el enfoque tradicional (estructurado), no cumplen con estas especificaciones, inclusive muchas de las aplicaciones actuales son una masa gigantesca de código que crecio en forma anárquica y por lo cual estas aplicaciones son difíciles de mantener y actualizar, por lo cual muchas veces se decide desecharlas cuando fallan y crear una nueva

versión, pero desde el principio, lo cual crea una perdida de tiempo y de dinero. Muchas de estas aplicaciones, que se encuentran formadas por una inmensa masa de código, presentan redundancias en diversos módulos (funciones que se repiten a lo largo de el código). Las redundancias son lo de menos, estas enormes moles de código presentan severos errores por lo que fallan con frecuencia y por ende necesitan mantenimiento. Una persona que les da mantenimiento a estas moles, y que no intervino en el desarrollo de éstas, siempre se encuentra en estado de shock debido a que no sabe donde comenzar y lo peor sucede cuando estos verdaderos dinosaurios no poseen una adecuada documentación o de plano no poseen documentación alguna.

Las ventajas de usar el enfoque orientado a objetos, es que al menos existe un conocimiento uniforme entre todas las personas que intervienen en algún proyecto. Y las personas que poseen este enfoque al menos hicieron el esfuerzo para adoptar una nueva tecnología (para ellos) y lo cual implica que pueden (nadie lo garantiza) tener menos vicios que las personas que se niegan al cambio y que se aferran a lo tradicional.

En este momento existen muchas tecnologías esperando ser utilizadas en una forma correcta. En estos últimos años yo he escuchado a personas decir: "yo creía que la computación era muy buena, no podemos resolver muchos de nuestros problemas con las computadoras y por lo cual todavía tenemos que resolverlos en la forma que solíamos hacerlo, a mano y con la ayuda de una calculadora (esto fue dicho por una contadora)". Sin importar que tecnologías estén usando las empresas, si éstas están utilizando en una forma correcta todos sus recursos computacionales, es probable que no tengan muchos dolores de

cabeza y que en verdad si puedan resolver sus problemas en una forma eficiente. Es en verdad vergonzoso que algunas empresas (por lo regular pequeñas) sigan trabajando en una forma prehistórica y que sigan resolviendo a mano todos sus problemas. Muchas personas siguen usando tecnologías tradicionales, como el enfoque estructurado, pero lo peor viene cuando algunas personas de plano no utilizan ninguna tecnología.

Para cerrar este tema relacionado al enfoque orientado a objetos, podemos decir que no es un nuevo enfoque y que no es una nueva forma de pensar, pero que si es una tecnología muy completa y que nos ofrece otra opción para crear aplicaciones y por lo cual es la filosofía que yo he decidido usar para desarrollar Hyptextw (si usted desea conocer el funcionamiento de Hyptextw, por favor vea el apéndice D).

Turbo C++ en el Desarrollo de Hyptextw

He seleccionado Turbo C++, especificamente Turbo C++ para Windows de Borland, debido a que es un lenguaje muy poderoso. En el pasado el lenguaje C fue muy utilizado para desarrollar aplicaciones comerciales tales como Clipper, Dbase III y IV, Lotus 123, Informix-4GL, inclusive para desarrollar sistemas operativos tales como UNIX. Ahora C++ es utilizado para desarrollar ese tipo de software, aunque C es todavía usado fuertemente debido a su gran flexibilidad y potencia para desarrollar aplicaciones. Por ejemplo el sistema operativo Newton de la Apple, consiste de cuatro partes principales: la arquitectura de reconocimiento, la arquitectura de comunicaciones, la arquitectura de información y un asistente inteligente, la mayor parte de esto

fue desarrollado en C y C++, con una pequeña porción del kernel escrito en lenguaje ensamblador. La interfaz de usuario fue escrita en C, C++ y NewtonScript (que es un nuevo lenguaje y el cual posee rasgos similares de C y Pascal, NewtonScript hace un manejo automático de la memoria y hace una recolección de posible basura en memoria, todas las referencias son a objetos y no identificadores o apuntadores). NewtonScript es un lenguaje en el cual Apple espera que todas las aplicaciones que corran sobre el sistema operativo Newton sean escritas en este lenguaje. Aunque siguen apareciendo nuevos lenguajes, C++, como una extensión al lenguaje C, sigue siendo una fuerte herramienta para desarrollar aplicaciones. Un ejemplo muy claro es Newton, la Apple no utilizo a NewtonScript en la mayor parte del código de su sistema operativo, la Apple utilizo para la dura faena a C y a C++ en lugar de NewtonScript. Va ser muy difícil sustituir de la noche a la mañana a C y C++ por otros lenguajes, debido a que mucho software ha sido desarrollado bajo C o C++ o ambos (como es el caso de el sistema operativo Newton).

Ya hablando especificamente de Turbo C++ de Borland (debido a que existen varios compiladores y de diferentes compañías), se puede decir que éste es una extensión de Turbo C. De hecho se puede combinar mucho de Turbo C y mucho de Turbo C++ en un mismo programa, pero hacer esto no es muy sano (de hecho esto ya no sigue un enfoque estrictamente orientado a objetos). Digamos que turbo C++ es un C más potente y más flexible, puesto que conserva lo de un turbo C más un algo, ese algo más Turbo C hacen a Turbo C++. Ahora agregándole al poder de Turbo C++ para Windows a ObjectWindows (que es una biblioteca de clases y que promueve la filosofía de reusabilidad), el resultado es que se pueden crear aplicaciones más rápidamente

y de una manera más amigable. Aunque la utilización de una estructura de trabajo, como lo es ObjectWindows, implica que tenemos que seguir ciertas reglas aunque también ofrece muchas ventajas para crear interfaces de usuario en una forma fácil.

En conclusión a todo de lo que se ha hablado en este capítulo, podemos decir que he seleccionado una aplicación de hipertexto debido a la importancia de la información electrónica y que además puede ser de utilidad para aquellos que se dedican a escribir información programada y para aquellos que necesitan de esta información programada (estudiantes). En tanto a un enfoque orientado a objetos, podemos decir que es una tecnología de alto nivel y cuya filosofía de trabajo no es extraña para nosotros puesto que se basa en cosas familiares para nosotros, los objetos. Todo esto en conjunción con Turbo C++ y con una biblioteca de clases (listas para desarrollar interfaces) son las causas principales de que haya yo seleccionado a:

1. Hyptextw como aplicación a desarrollar.
2. El enfoque orientado a objetos como metodología a seguir.
3. Y a Turbo C++ para Windows como herramienta de desarrollo.

CAPITULO 3

ANALISIS

Para un mejor análisis de sistemas es necesario poseer herramientas que nos faciliten el trabajo. Sin la existencia de herramientas para el análisis, los sistemas finales serían más que nada un montón de código (monolítico), los cuales tendrían infinidad de errores. Los primeros sistemas que los analistas y programadores crearon fueron más que nada monolíticos.

Hoy en día existen muchas técnicas para desarrollar sistemas, a pesar de ello, muchas compañías siguen creando sus sistemas como si no existieran dichas técnicas. En las compañías en las que yo he trabajado no utilizan técnicas para desarrollar sus sistemas. Inmediatamente que ellos conocen el problema empiezan a la programación del mismo en la computadora, ya sea en Clipper o C, sin si quiera pensar un poquito el curso que llevará el desarrollo del mismo. Yo pensaba que esto solo ocurría en empresas de gobierno, pero no es así, esto también ocurre en empresas particulares.

Para el desarrollo de sistemas de software con objetos es necesario utilizar herramientas adecuadas, el enfoque estructurado no entra aquí, para ello es necesario utilizar el enfoque orientado a objetos, en este caso el análisis orientado a objetos.

El objetivo de este capítulo es el de explicar los diagramas utilizados para el sistema Hyptextw.exe, en la etapa de análisis, y no el de explicar el

análisis orientado a objetos, para ello el lector tendría que leer un libro sobre este tema.

Los diagramas que intervienen en la etapa de análisis son los siguientes:

1. Diagramas de jerarquía.
2. Diagramas de objeto-relación.
3. Diagramas de composición.
4. Esquema de objetos.
5. Diagrama de transición de estados.
6. Esquema de eventos.
7. Diagrama de flujo de objetos.
8. Diagrama de mensajes.

Antes de hablar sobre la clase de diagramas, mencionados renglones atrás, es necesario aclarar que los diagramas (dependiendo del enfoque que se esté utilizando) son un lenguaje con el cual se pueden comunicar las personas implicadas en el desarrollo de sistemas, sin un lenguaje adecuado estas personas no podrían comunicar sus ideas a otros y viceversa. O en caso de que si pudieran expresar sus ideas a los demás, quizás dichas ideas no fueran lo suficientemente claras para expresar el pensamiento de tal persona.

Los diagramas tienen varios propósitos:

Uno: El poder modelar el mundo en papel y de esta forma entender en una forma más clara el problema.

Dos: Los documentos sirven para futuras modificaciones a un sistema. De esta forma las personas, que vayan a modificar su sistema en un futuro y en caso que usted ya no se encuentre laborando en la empresa, tendrán suficiente información para revisar el sistema y de esta forma poder realizar modificaciones o mantenimiento, según sea el caso. Estas personas deberán echar mano de los diagramas para dejar por escrito la nueva versión del sistema.

Tres: Los documentos de un sistema también sirven para las auditorías.

Cuatro: El lenguaje que deberían utilizar las personas implicadas en un proyecto es el de diagramas, de esta forma la comunicación es (no sería) más transparente y más clara, de esta forma los errores se reducirían considerablemente y por consiguiente se reduciría el mantenimiento al sistema.

Muchas compañías que en verdad si echan mano de diagramas para desarrollar sus sistemas hacen mal uso de ellos. Todo mundo sabe que existen normas casi para cualquier cosa, incluyendo las de diagramas. Muchos departamentos de sistemas usan diagramas pero los suyos propios, es conveniente aclarar que es necesario utilizar los diagramas que ya hayan sido aprobados como normas y no los inventados por uno mismo. Esto puede provocar que cuando una persona revise tus diagramas no entienda nada de ellos debido a que no se están utilizando los diagramas correctos.

Diagramas de Jerarquía

Una vez habiendo explicado sobre la gran utilidad de los diagramas en el desarrollo de sistemas, es necesario explicar aquellos diagramas que intervienen en la etapa de análisis para sistemas que incluyen objetos.

El análisis orientado a objetos incluye dos mitades, una la estructura y la otra el funcionamiento de los objetos.

Empecemos por el análisis de la estructura de objetos. En esta etapa la siguiente información es identificada:

1. Los tipos de objetos y sus asociaciones.
2. Como están organizados los tipos de objetos, en súper tipos y en subtipos.
3. La composición de objetos.

Los diagramas de jerarquía entran en el punto dos, es decir, un diagrama de jerarquía nos indica que objetos se derivan de cuales objetos y cuales objetos se derivan de ellos mismos.

Para el desarrollo de Hyptextw, se hecho mano de una estructura de trabajo ampliamente conocida como ObjectWindows. Esta estructura de trabajo nos ofrece facilidades como ventanas, botones, barras de recorrido, cajas de diálogo, editores, un sistema básico de archivos, en si una interfaz de usuario compatible con cualquier interfaz de usuario para Windows.

ObjectWindows en si es una jerarquía de objetos que en realidad nos ahorraron el trabajo de diseñar una interfaz de usuario. En lo único que me tuve que preocupar fue en el análisis de los objetos que se pegarían en la jerarquía de objetos pertenecientes a ObjectWindows. En realidad en la jerarquía resultante, aparecen tanto objetos de ObjectWindows como objetos de Hyptextw (la aplicación).

Diagramas de Objeto-relación

En los diagramas de objeto relación podemos identificar a los objetos y sus relaciones.

Diagramas de Composición

Los diagramas de composición nos muestran cuantos objetos y de que tipo están incluidos dentro de otro objeto. Por ejemplo un carro posee cuatro ruedas y no sólo una, posee también varias puertas y ventanillas, también posee un numero determinado de asientos. A su vez la máquina de un carro posee varias bujías, varios pistones. En si el objetivo de un diagrama de composición es el de mostrar los objetos contenidos dentro de otros.

Un diagrama de composición también puede mostrar la jerarquía de los objetos y también las relaciones que existen entre ellos. De esta forma se pueden incluir varios tipos de diagramas en uno solo. De esta forma usted podrá observar jerarquías, relaciones y composición en un solo diagrama. Esto es realmente difícil de observar por una persona no acostumbrada a este tipo de

diagramas, pero para una persona capaz de leer esta clase de diagramas es demasiado útil tener toda esta información a la mano en un solo documento.

Lo aconsejable es crear un documento separado para cada tipo de diagrama.

En los diagramas de composición incluidos en este presente trabajo se pueden observar la relación entre objetos y la composición de objetos en un solo diagrama.

Esquema de Objetos

Un esquema de objetos es aquel en el que podemos encontrar jerarquías de objetos, objetos y sus relaciones con otros objetos y composición de objetos en un mismo diagrama.

Diagrama de Transición de Estados

En esta parte cubriremos el análisis del funcionamiento de objetos.

En el análisis del funcionamiento de objetos, la siguiente información es identificada:

1. Los estados en los que puede estar un objeto.
2. Las transiciones de estados.
3. Los eventos que ocurren.

4. Las operaciones que toman lugar.
5. Las interacciones entre objetos.
6. Los disparadores que reaccionan cuando un evento ocurre.

El diagrama de transición de estados nos muestra en que posibles estados puede encontrarse un objeto y en cuales no y por ende poder determinar el ciclo de vida de un objeto, desde el momento en que se crea hasta el momento que desaparece.

Un diagrama de transición de estados nos muestra el funcionamiento pero de un solo objeto, para determinar el funcionamiento de todos los objetos que existen en un sistema es necesario crear un diagrama de transición de estados para cada objeto.

Esquema de Eventos

Un esquema de eventos nos muestra operaciones, eventos y disparadores. Una operación es resultado de un evento. Al llegar a un evento determinado, éste se encargara de disparar la siguiente operación. Un objeto se encuentra en cierto estado debido a la ocurrencia de un evento. Esto es, cada vez que un evento ocurre una operación es disparada, esta operación es la que modifica el estado del objeto.

Para no confundirlo, en un esquema de eventos usted encontrará, como ya mencione lineas atrás, operaciones, eventos, disparadores, condiciones y relojes (un evento especial que dispara operaciones en determinados períodos

de tiempo). Las operaciones y eventos pueden ser externos o internos al sistema.

Un esquema de eventos tiene mucha relación con el diagrama de transición de estados.

Diagrama de Flujo de Objetos

En un diagrama de flujo de objetos, se muestran las distintas operaciones que intervienen y los objetos que son pasados a través de estas operaciones. Las operaciones y objetos pueden ser internos o externos a un sistema.

Diagramas de Mensajes

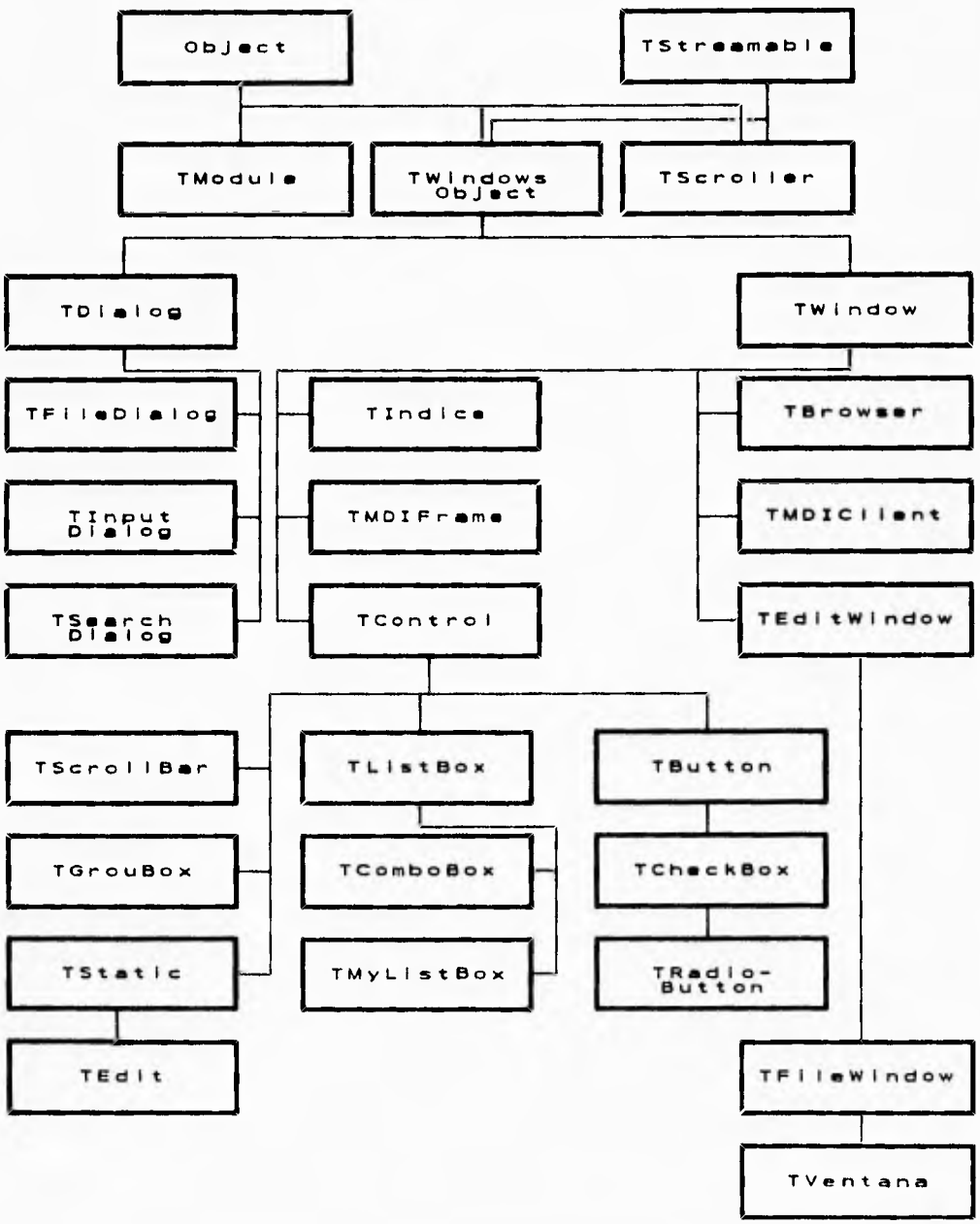
Los diagramas de mensajes muestran a los objetos y en la forma que interactuan en base a mensajes. Por supuesto, los mensajes son también mostrados.

Si usted lee los distintos libros dedicados al análisis y diseño orientados a objetos, usted se encontrará que en muchos de esos libros se utilizan distintas clases de diagramas y que son, por lo regular, los cuales les han funcionado a los autores de esos libros en la vida diaria. Por ejemplo Sally Shlaer y Stephen J. Mellor, en su libro OBJECT LIFECYCLES (Modeling the World in States), utilizan un conjunto de símbolos muy confusos y complicados, mientras que James Martin y James J. Odell, en su libro Object Oriented Analysis and Design, utilizan símbolos muy simples y que no dan lugar a la confusión, y es la

metodología y la simbología de Martin y Odell la que escogí para modelar Hyptextw.

Existen bastantes metodologías y simbologías para el análisis y diseño de sistemas, pero existen en plan de sugerencia. Esto dependerá de las mayorías el cual utilizar, pero al menos la metodología y simbología de Martin y Odell parecen los más populares y por ende los más utilizados.

DIAGRAMA DE GENERALIZACION DE TIPO FERN MOSTRANDO LAS CLASES DE EL SISTEMA HYPTEXTW



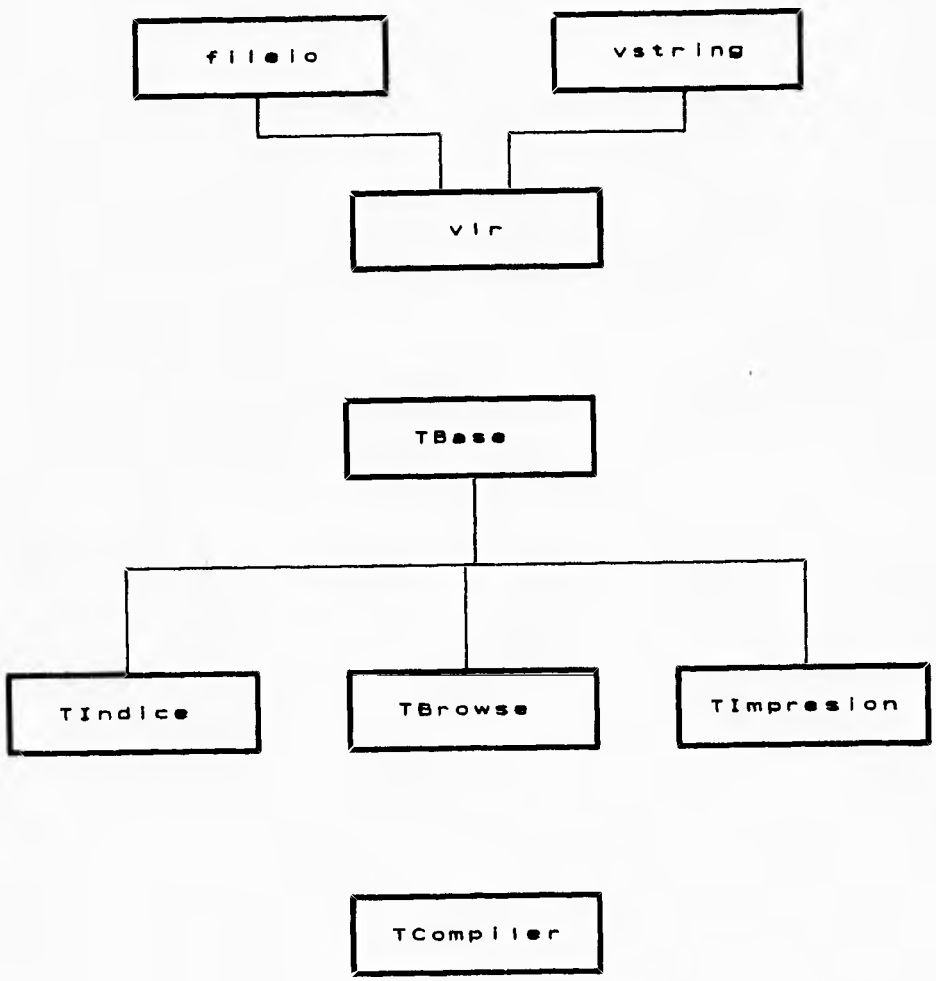

filelo
vstring
vlr
TBase
TIndice
TBrowse
TImpresion
TCompiler

Diagrama de composición para Hyptextw

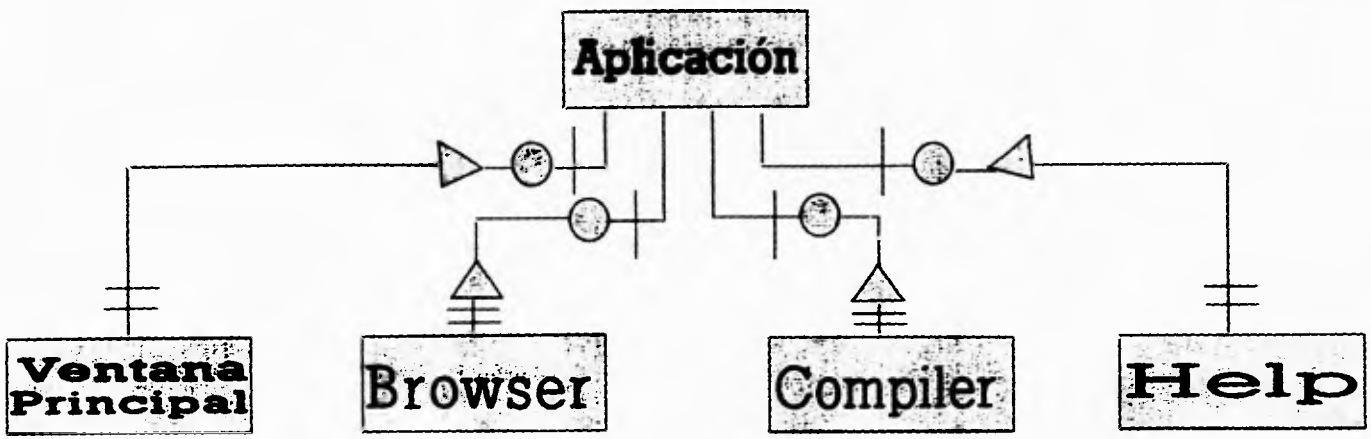


Diagrama de Composición para La Ventana Principal

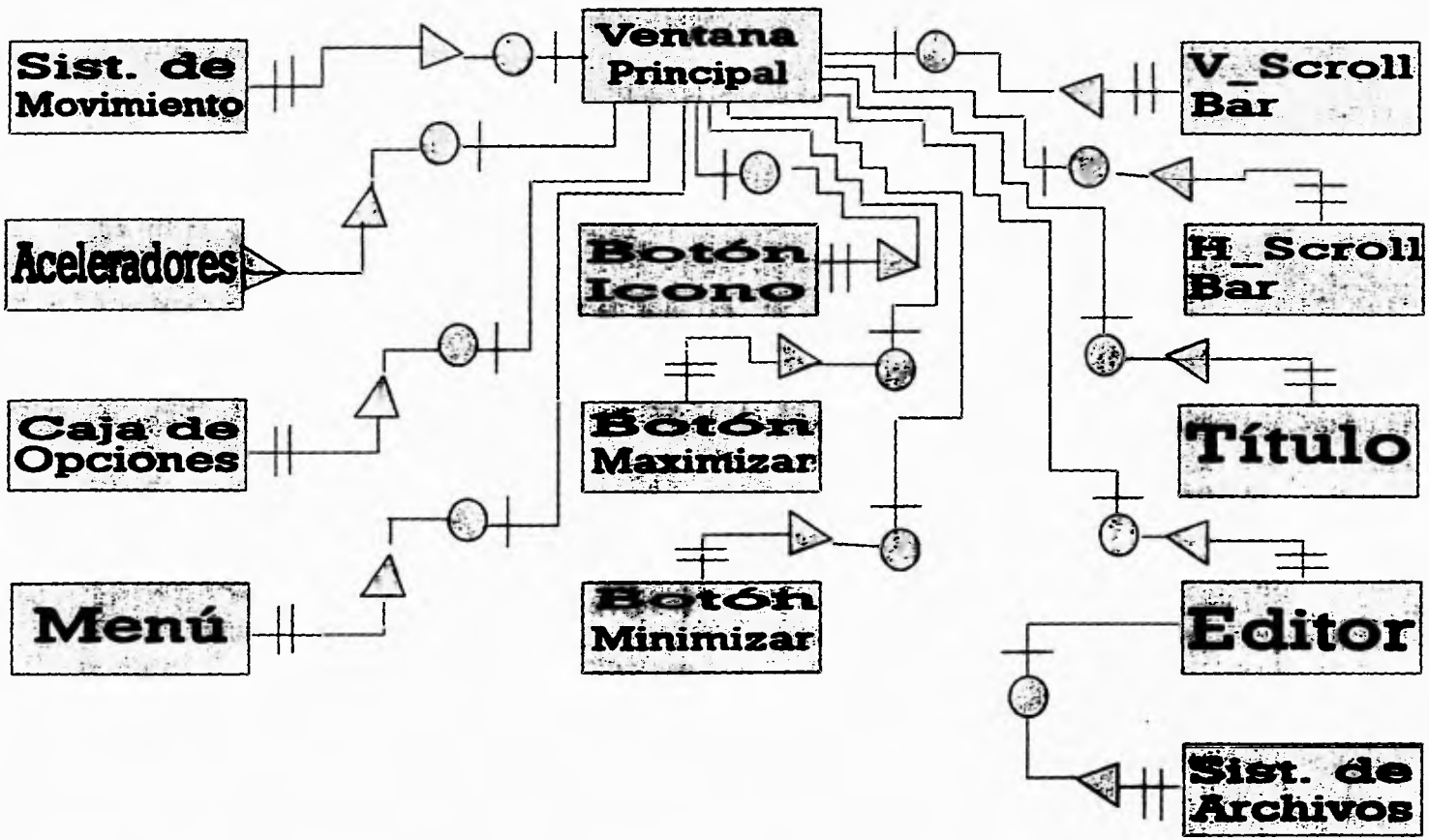


Diagrama de Composición
para
El Browser

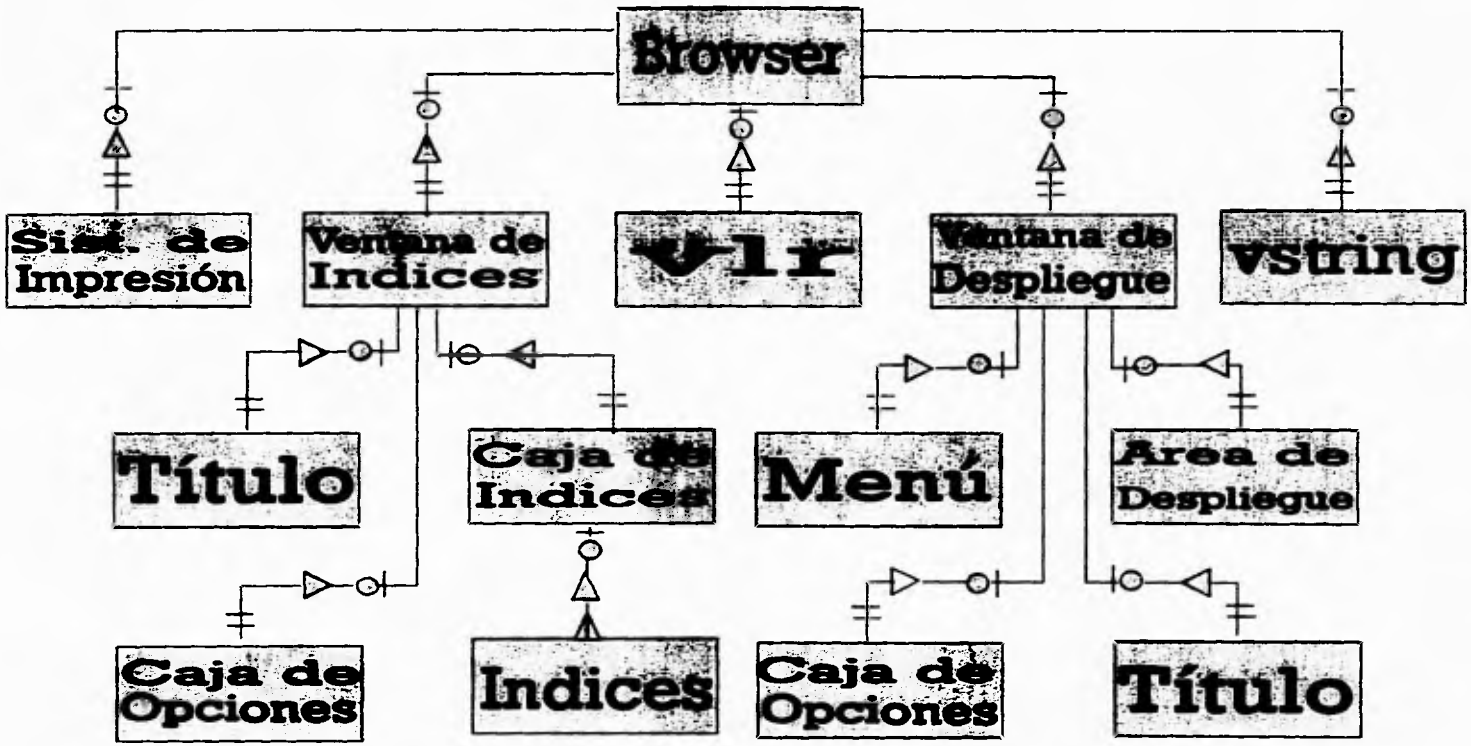
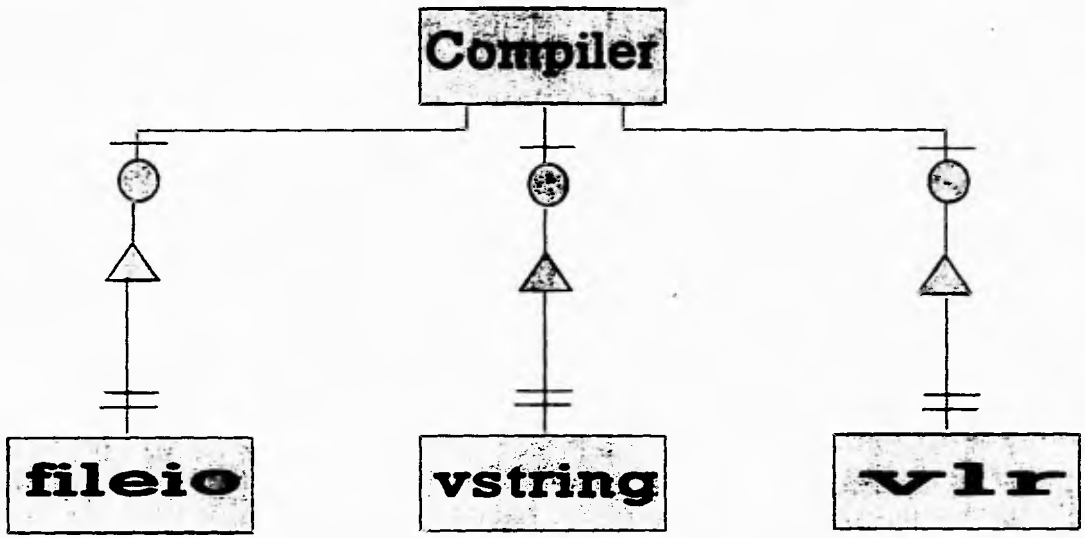
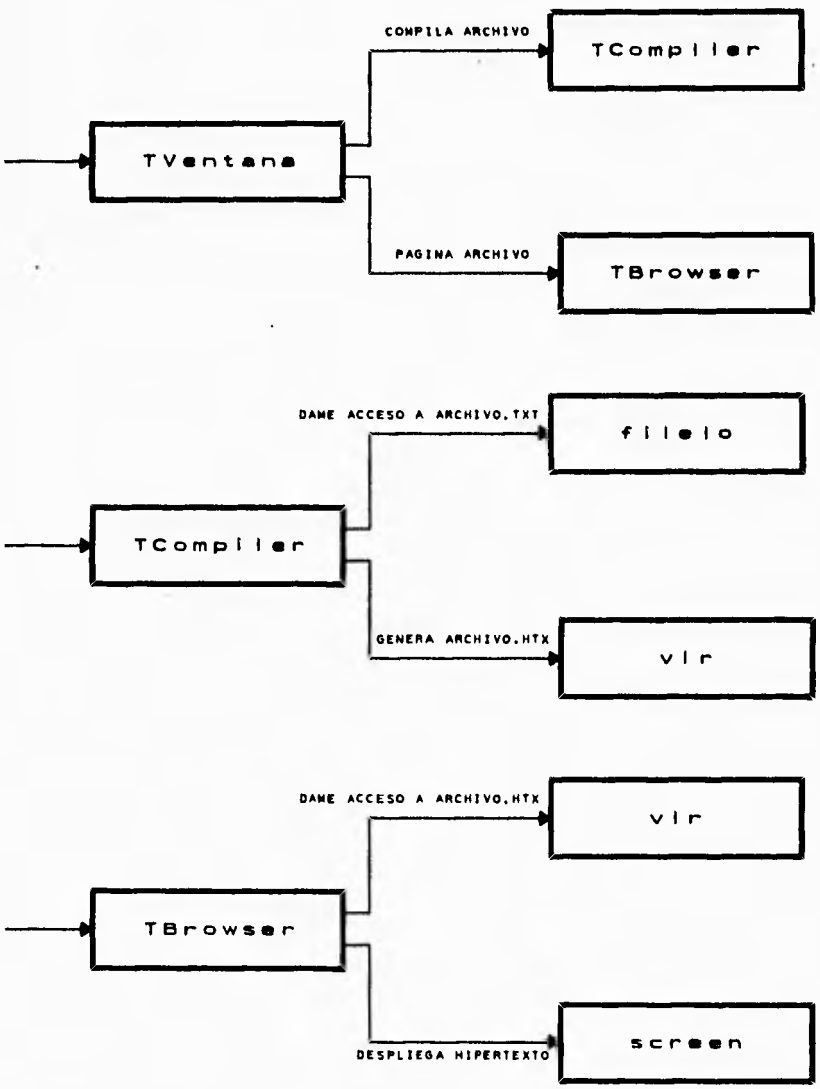


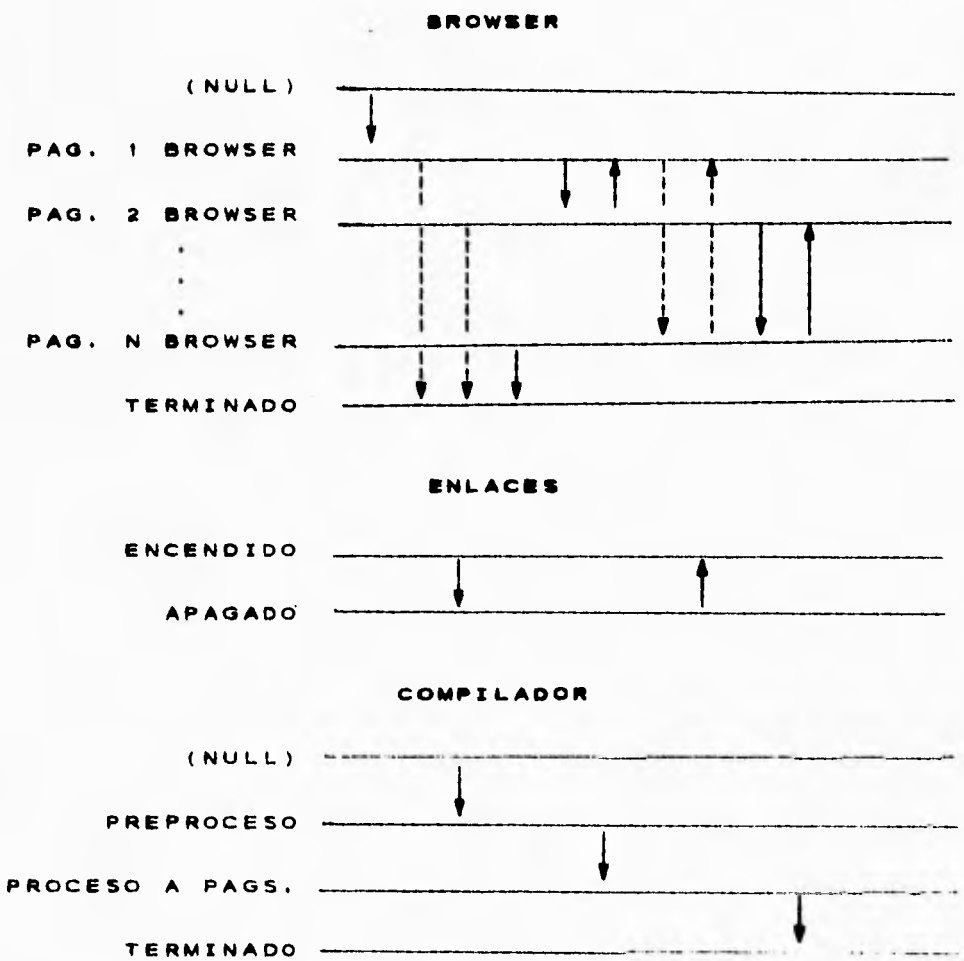
Diagrama de Composición
para
El Compilador



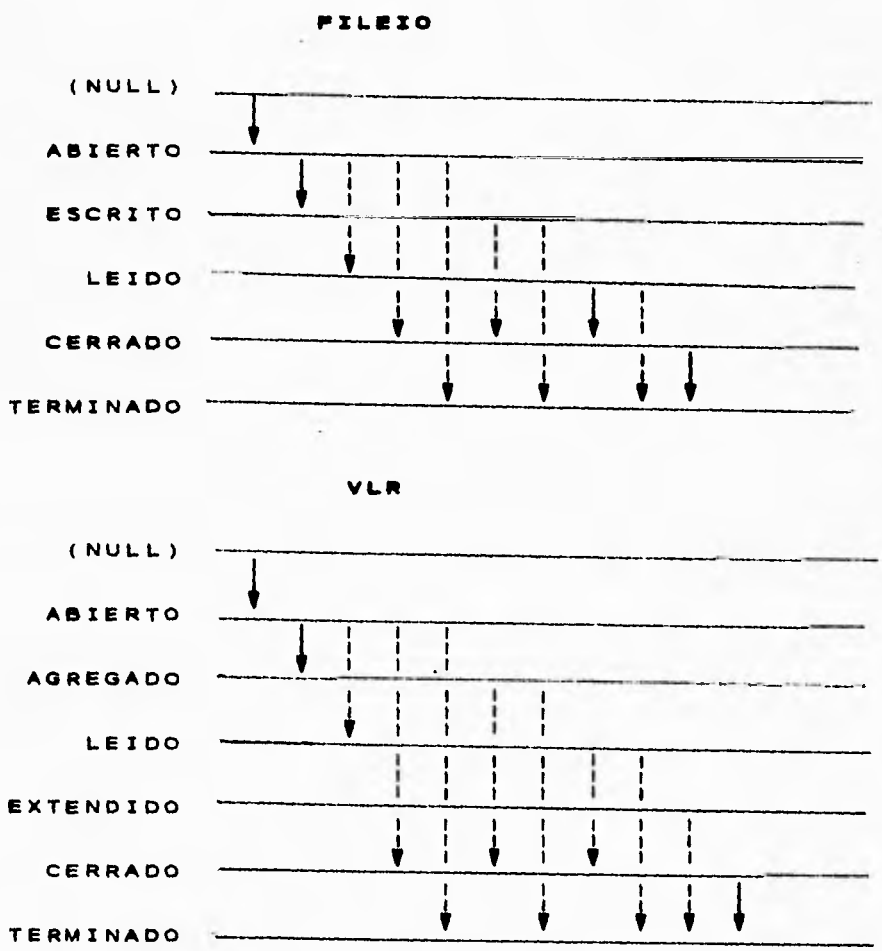
DIAGRAMAS DE MENSAJES



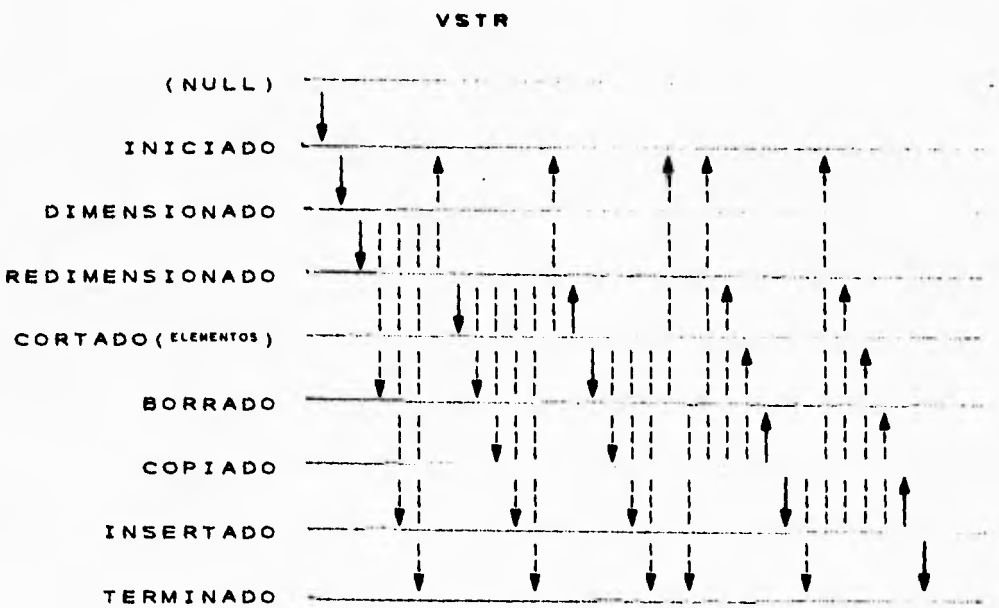
DIAGRAMAS DE TRANSICION DE ESTADOS



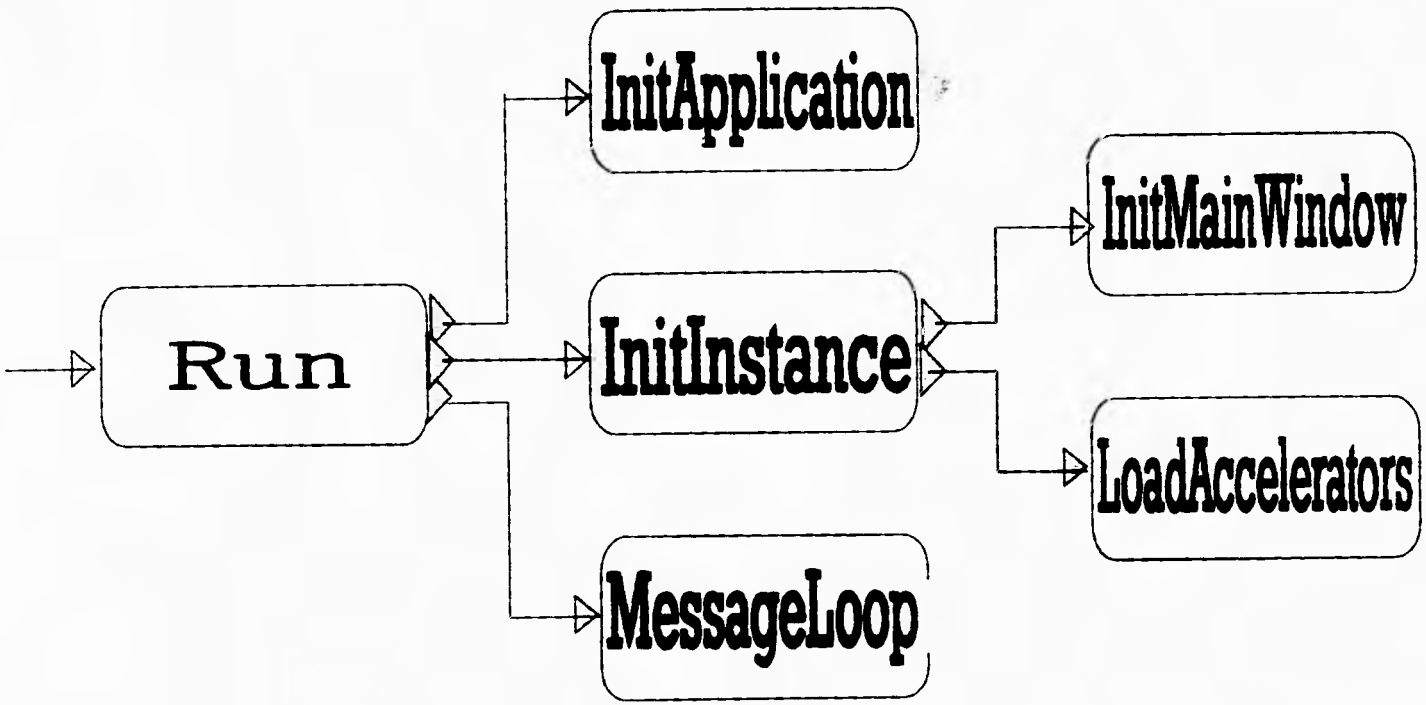
DIAGRAMAS DE TRANSICION DE ESTADOS



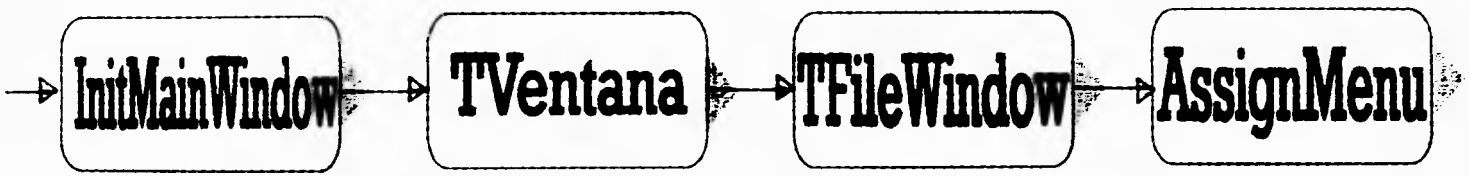
DIAGRAMAS DE TRANSICION DE ESTADOS



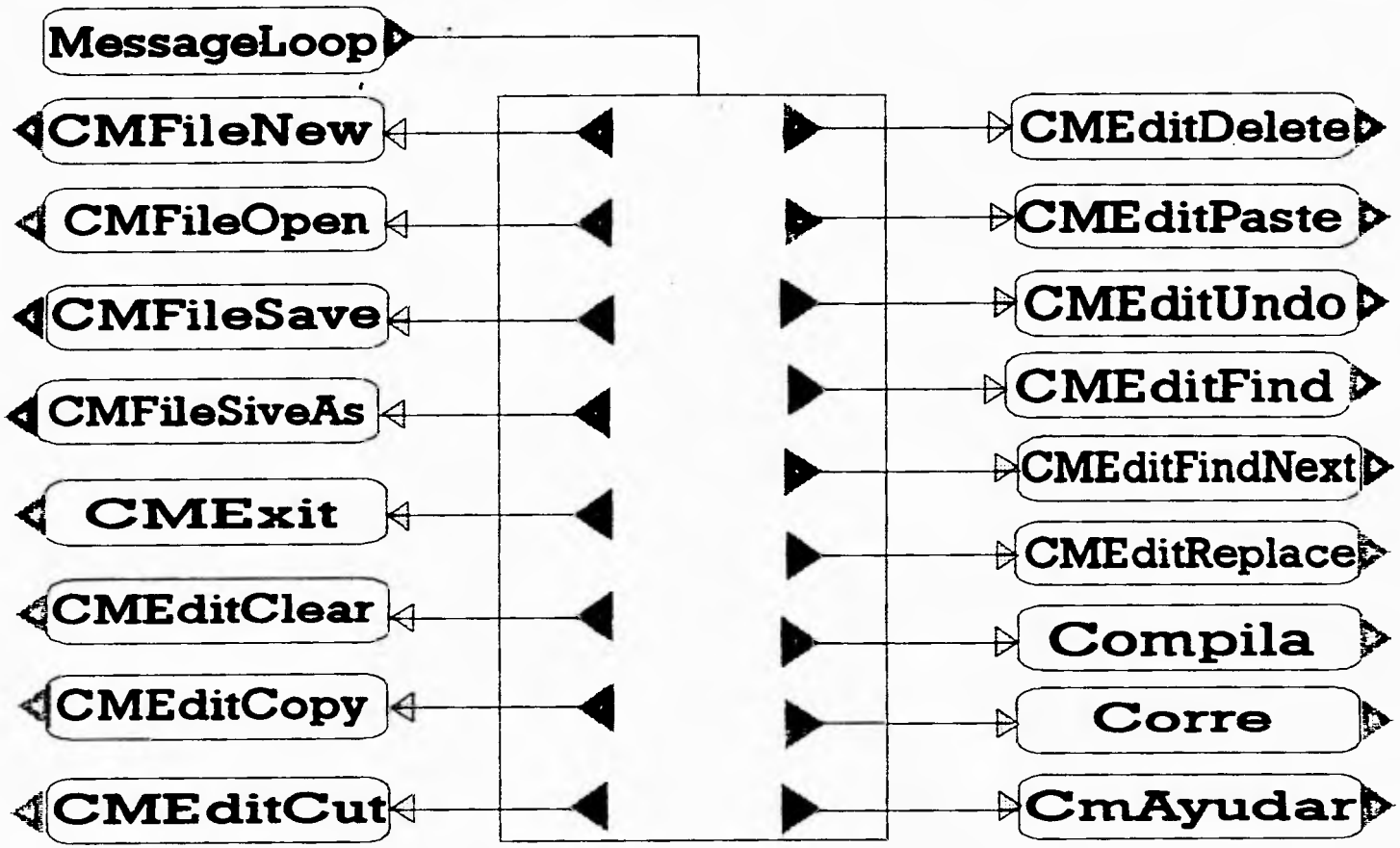
Esquema de Eventos para Hyptextw



Esquema de Eventos para la Ventana Principal

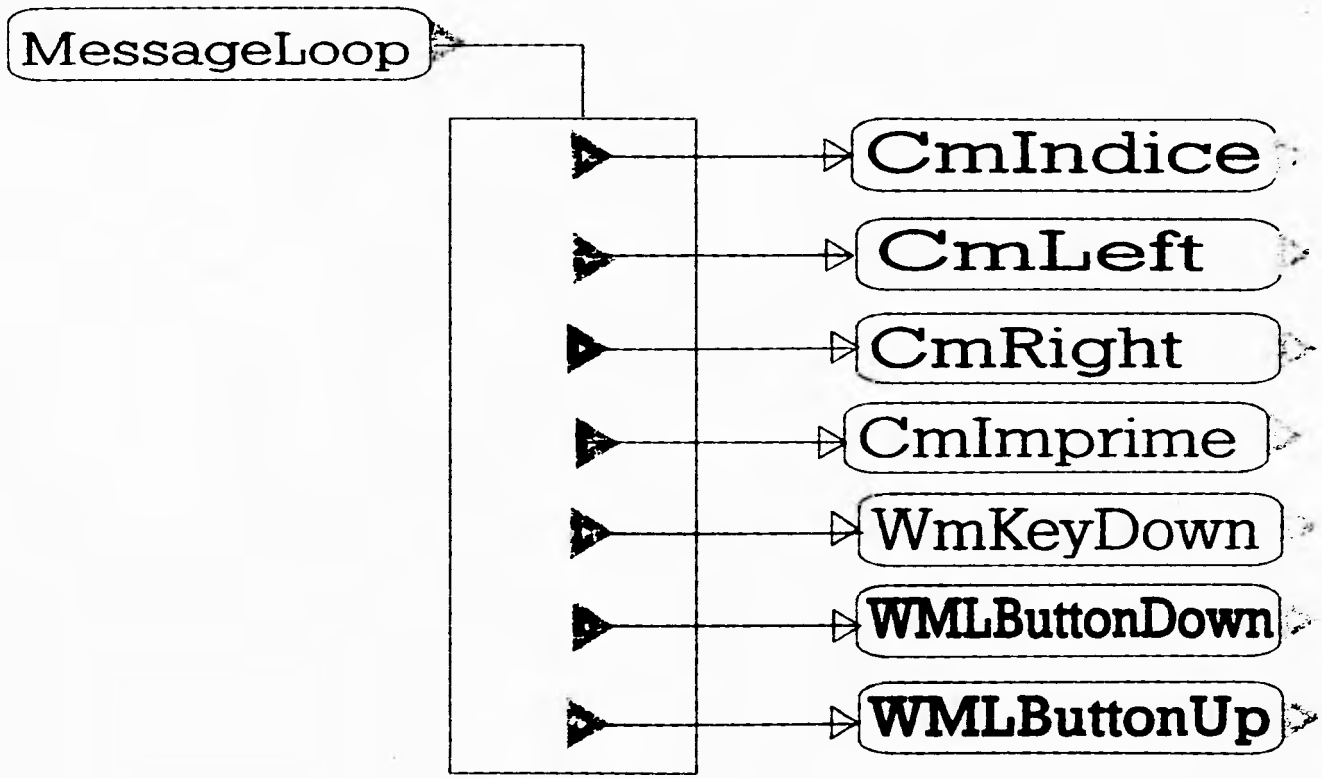


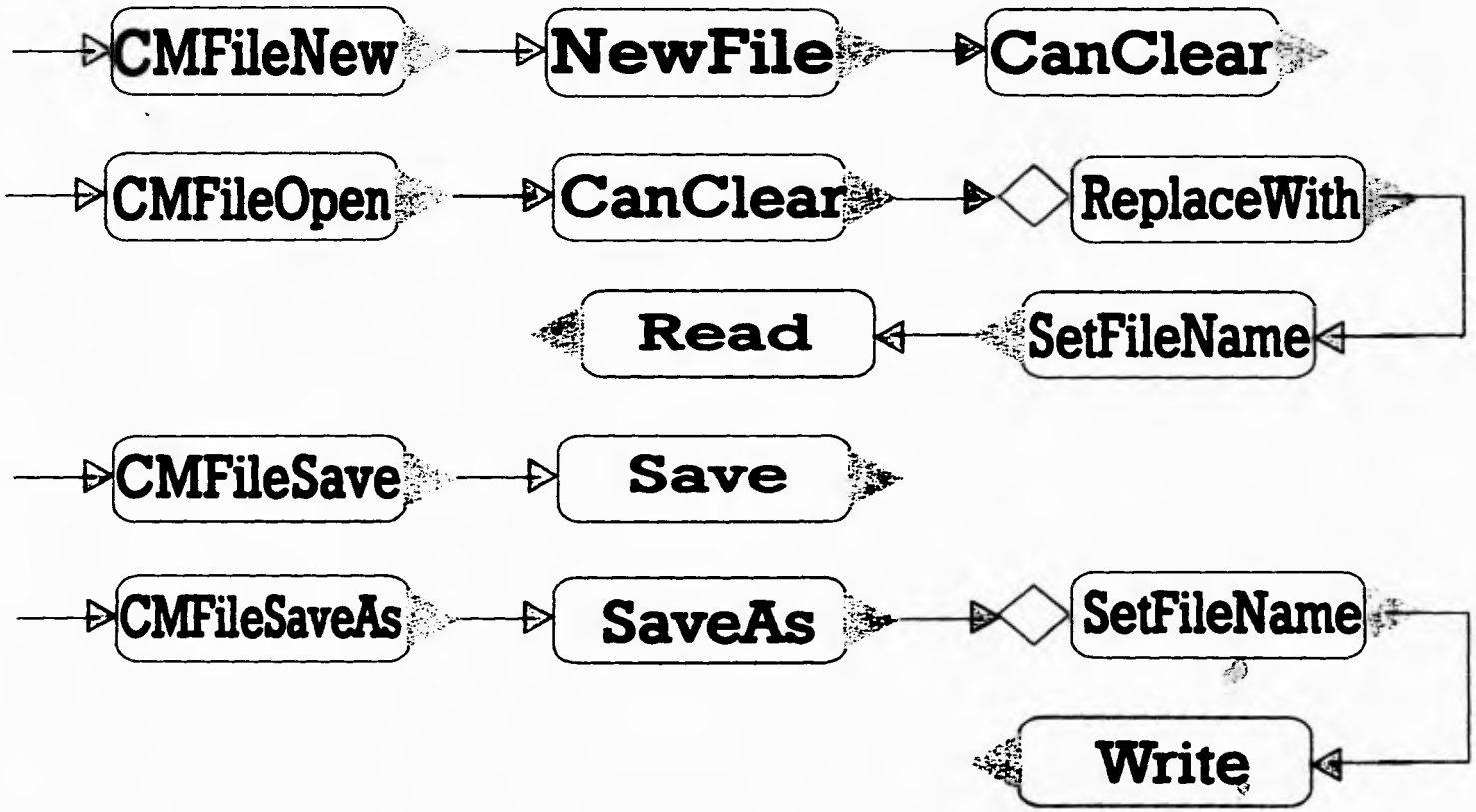
Esquema de Eventos Para el Proceso de Mensajes

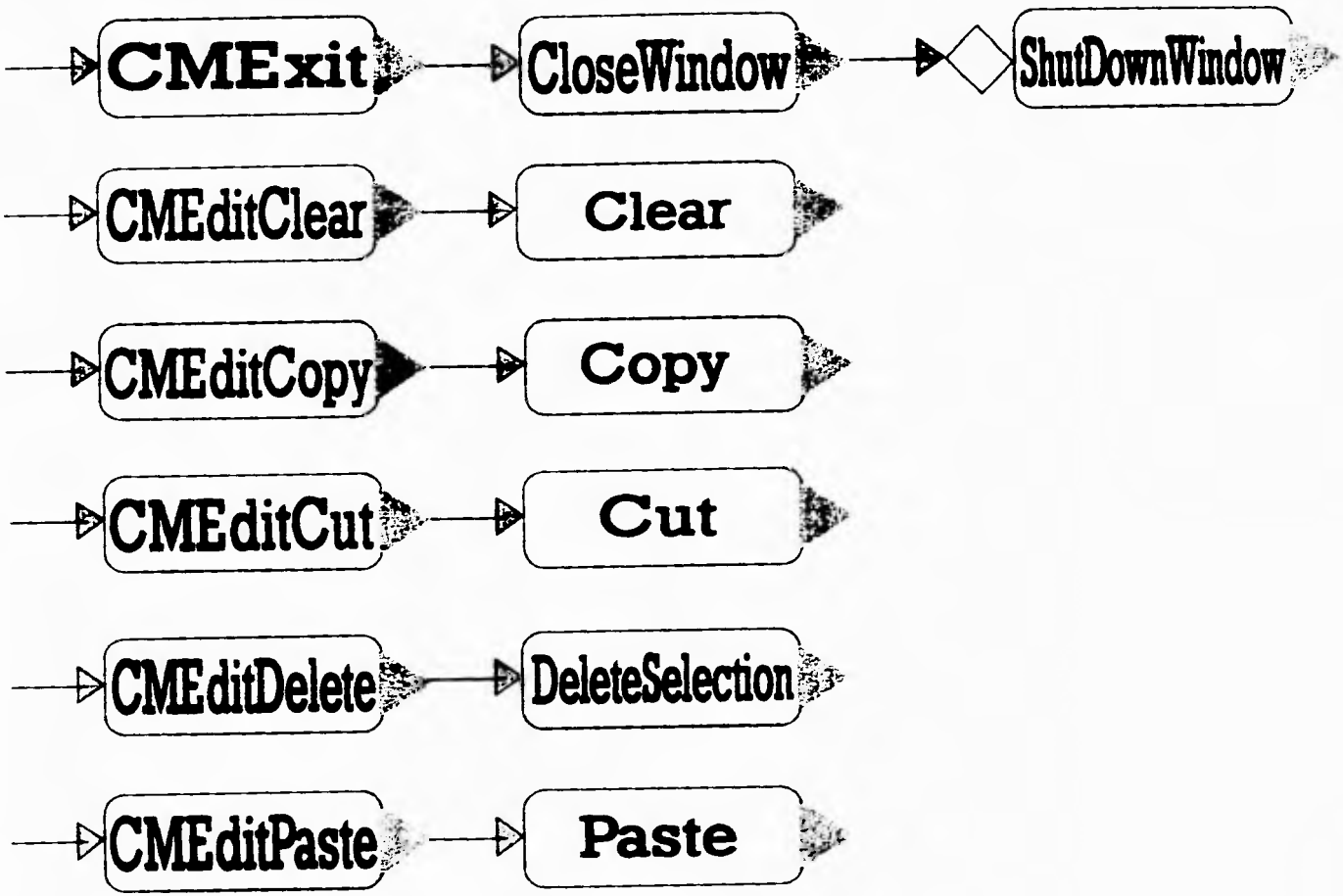


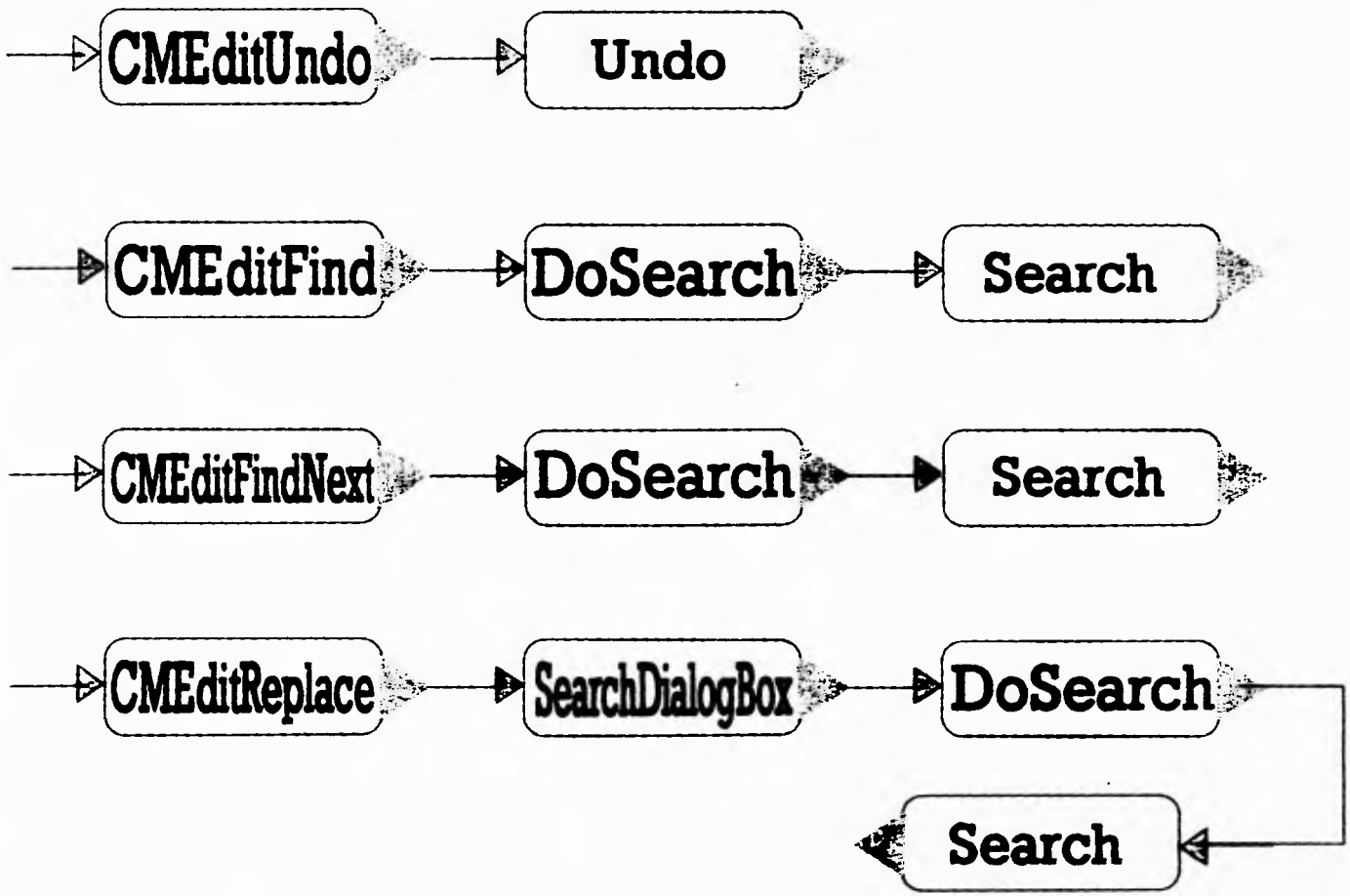
Esquema de Eventos Para el Proceso de mensajes

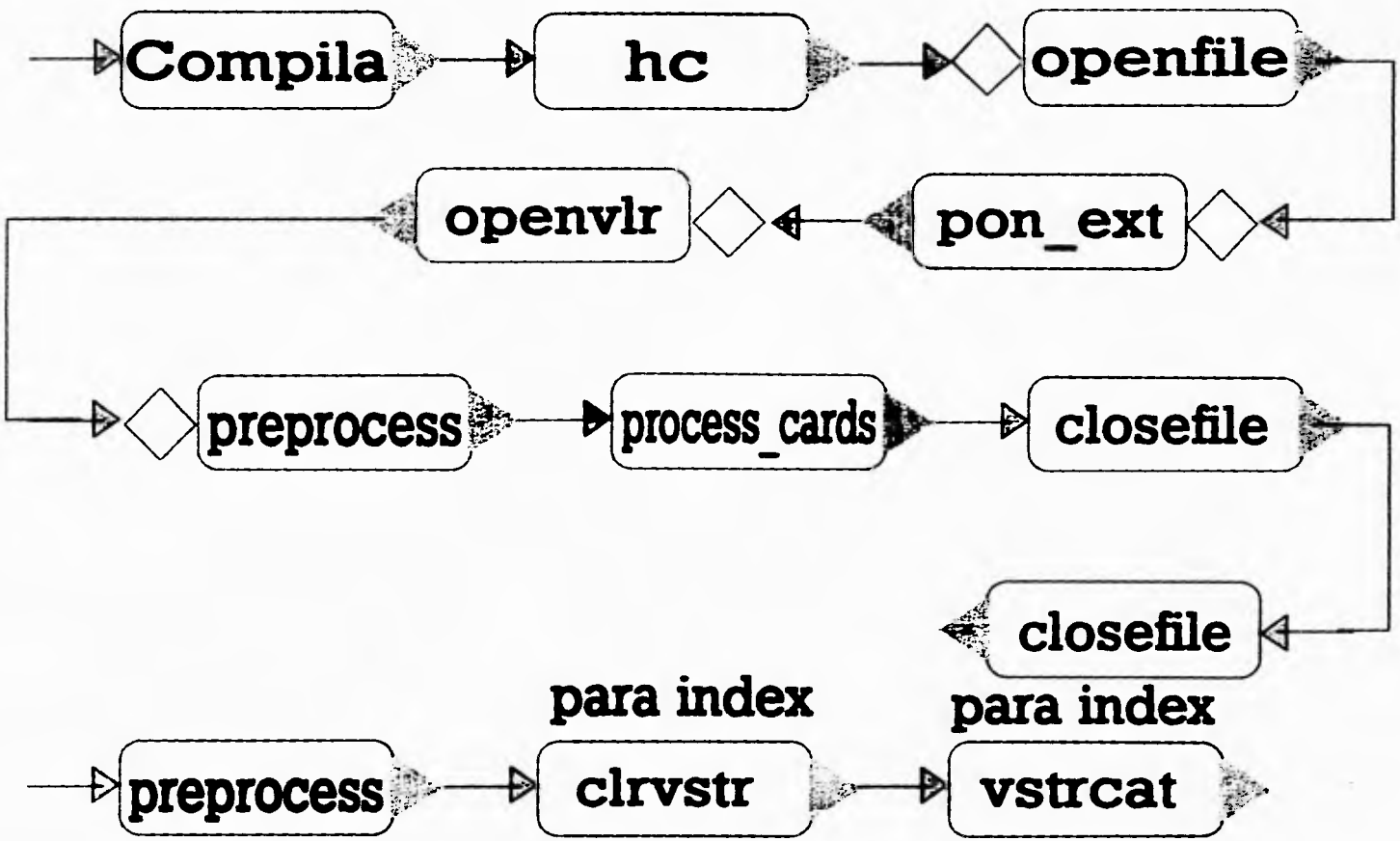
Para la Ventana del Browse

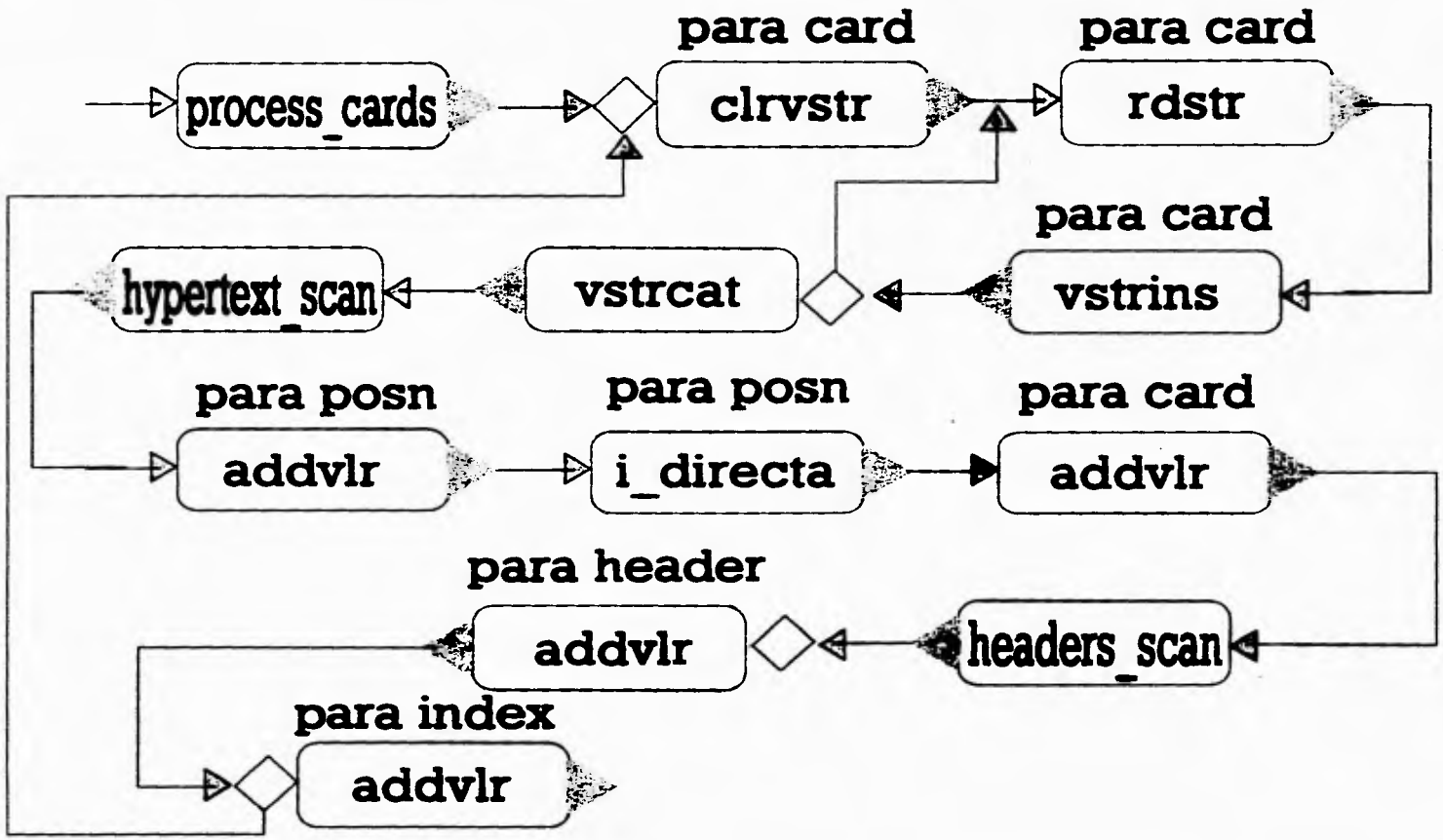


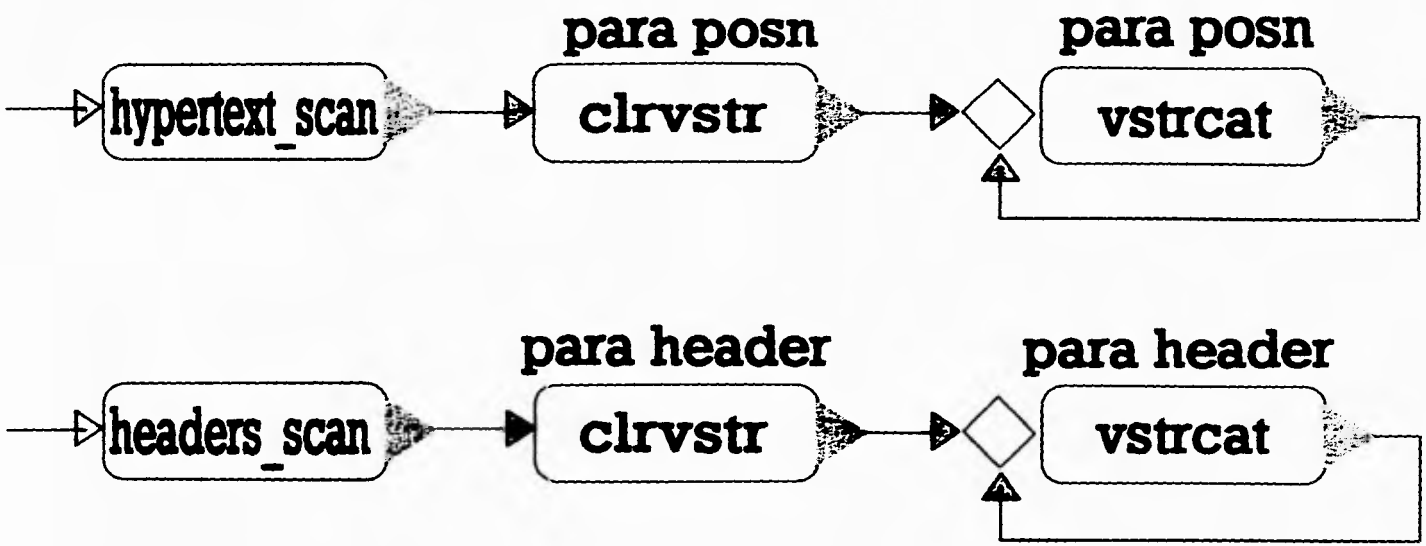

CMFileNew
NewFile
CanClear
CMFileOpen
CanClear
ReplaceWith
Read
SetFileName
CMFileSave
Save
CMFileSaveAs
SaveAs
SetFileName
Write

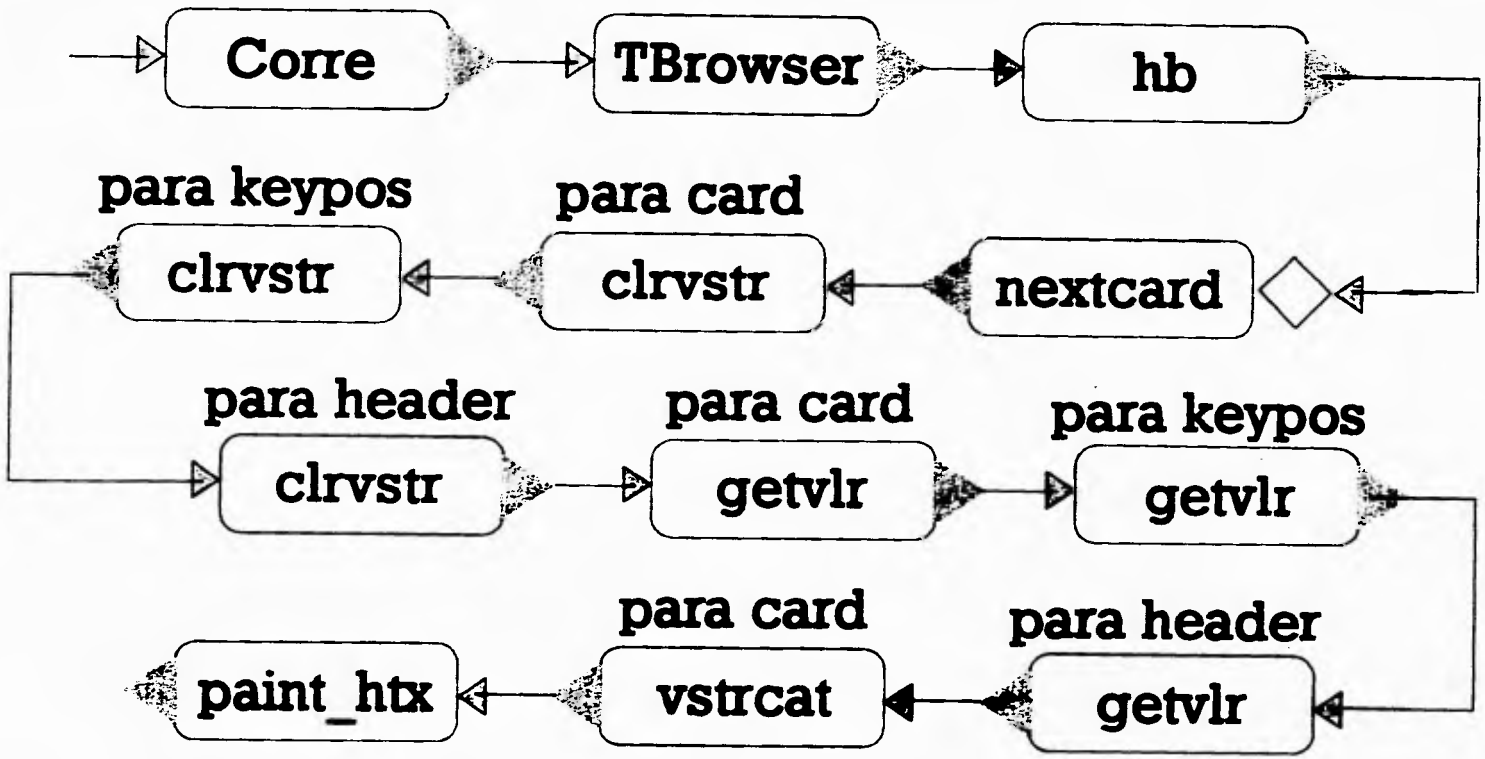

CMExit
CloseWindow
ShutDownWindow
CMEditClear
Clear
CMEditCopy
Copy
CMEditCut
Cut
CMEditDelete
DeleteSelection
CMEditPaste
Paste

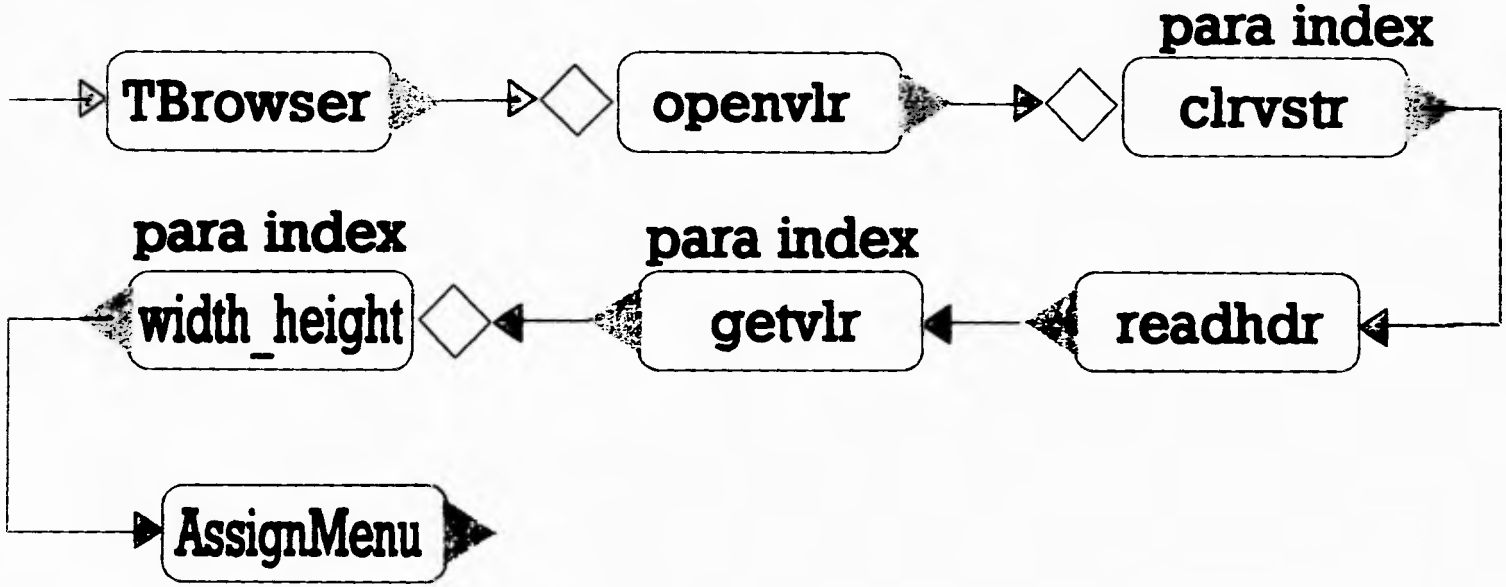

CMEditUndo
Undo
CMEditFind
DoSearch
Search
CMEditFindNext
DoSearch
Search
CMEditReplace
SearchDialogBox
DoSearch
Search

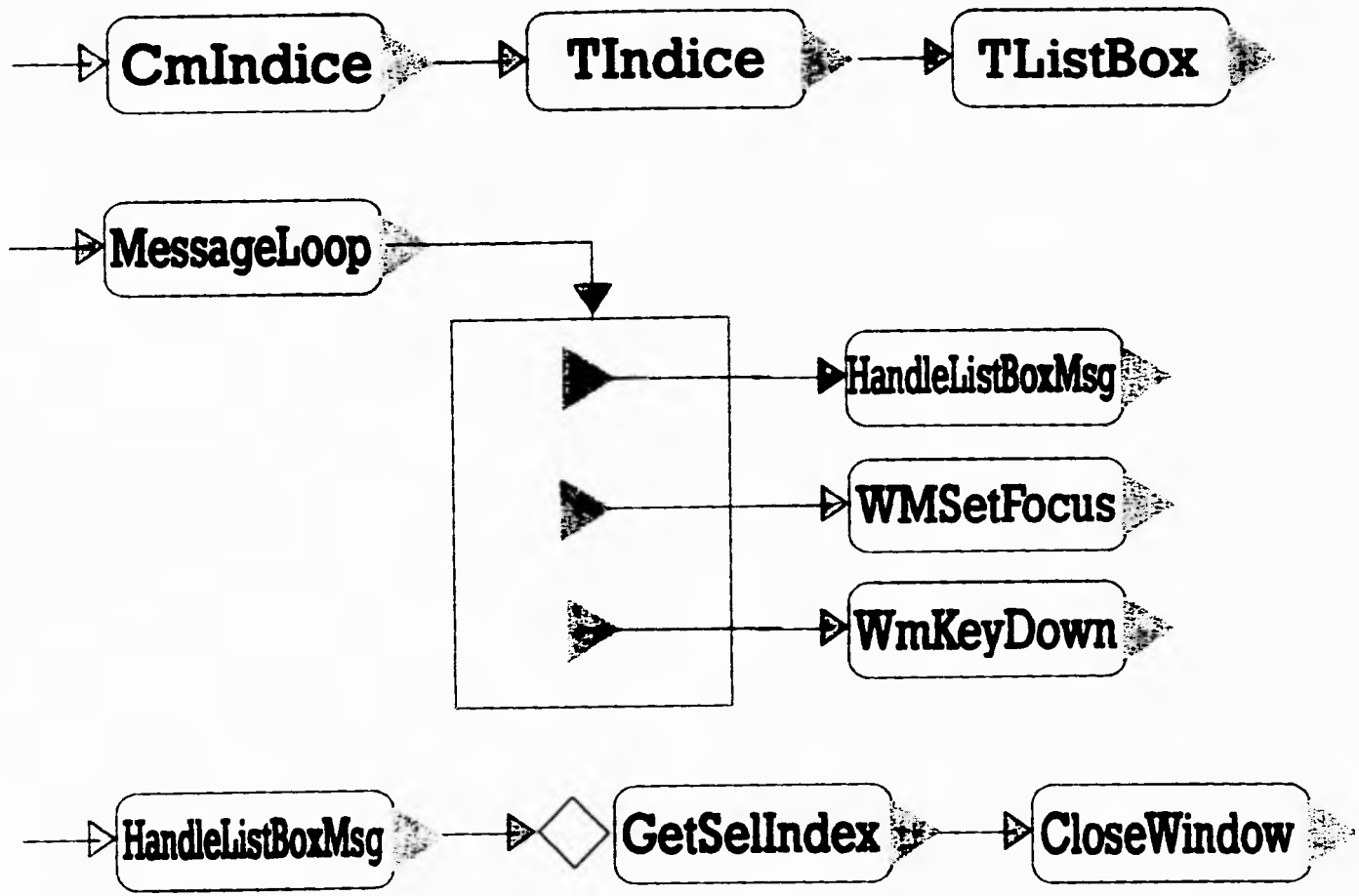

Compila
hc
openfile
openvlr
pon_ext
preprocess
process_cards
closefile
closefile
para index
para index
preprocess
clrvstr
vstrcat

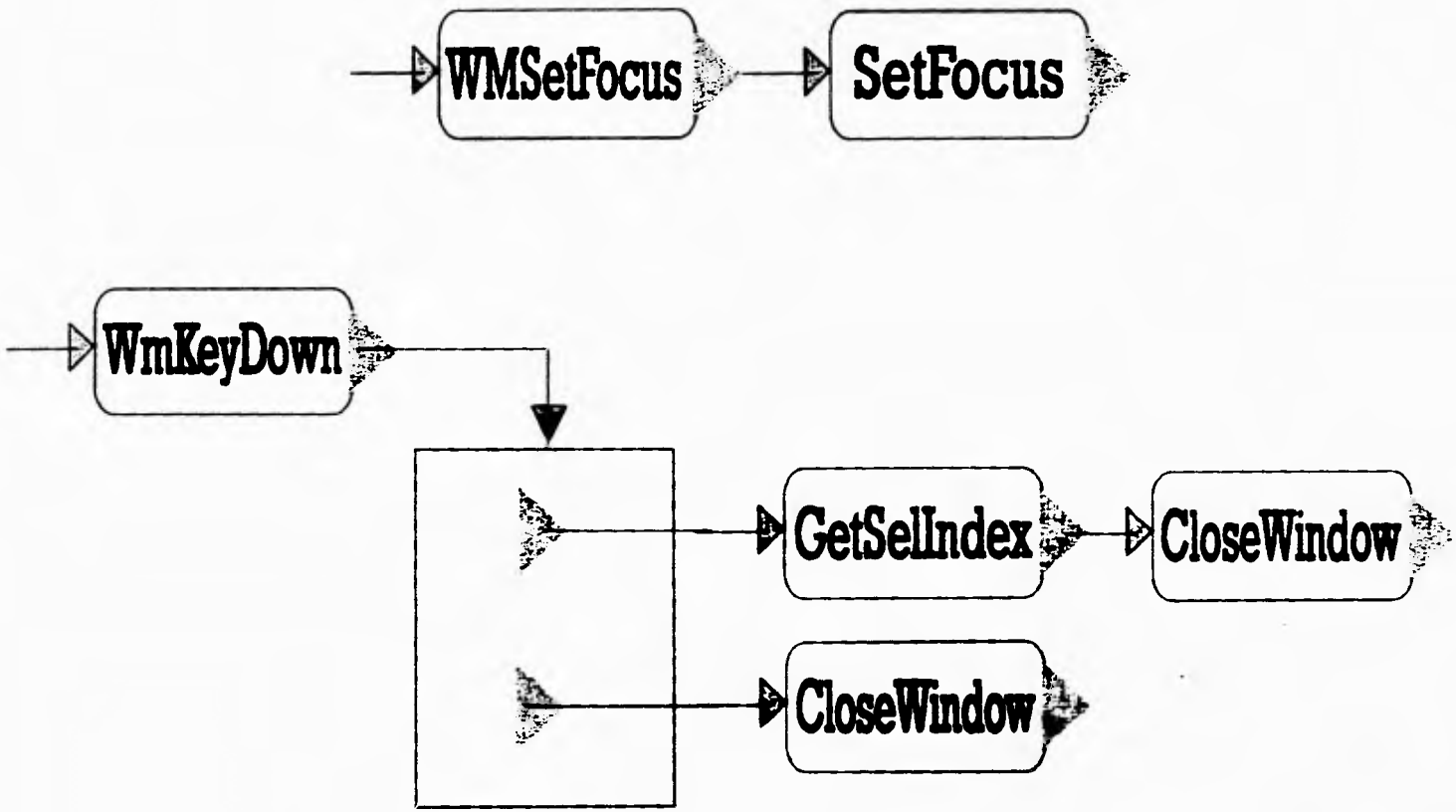

process_cards
para card
clrvstr
para card
rdstr
hypertext_scan
vstrcat
para card
vstrins
para posn
addvlr
para posn
i_directa
para card
addvlr
para header
addvlr
headers_scan
para index
addvlr

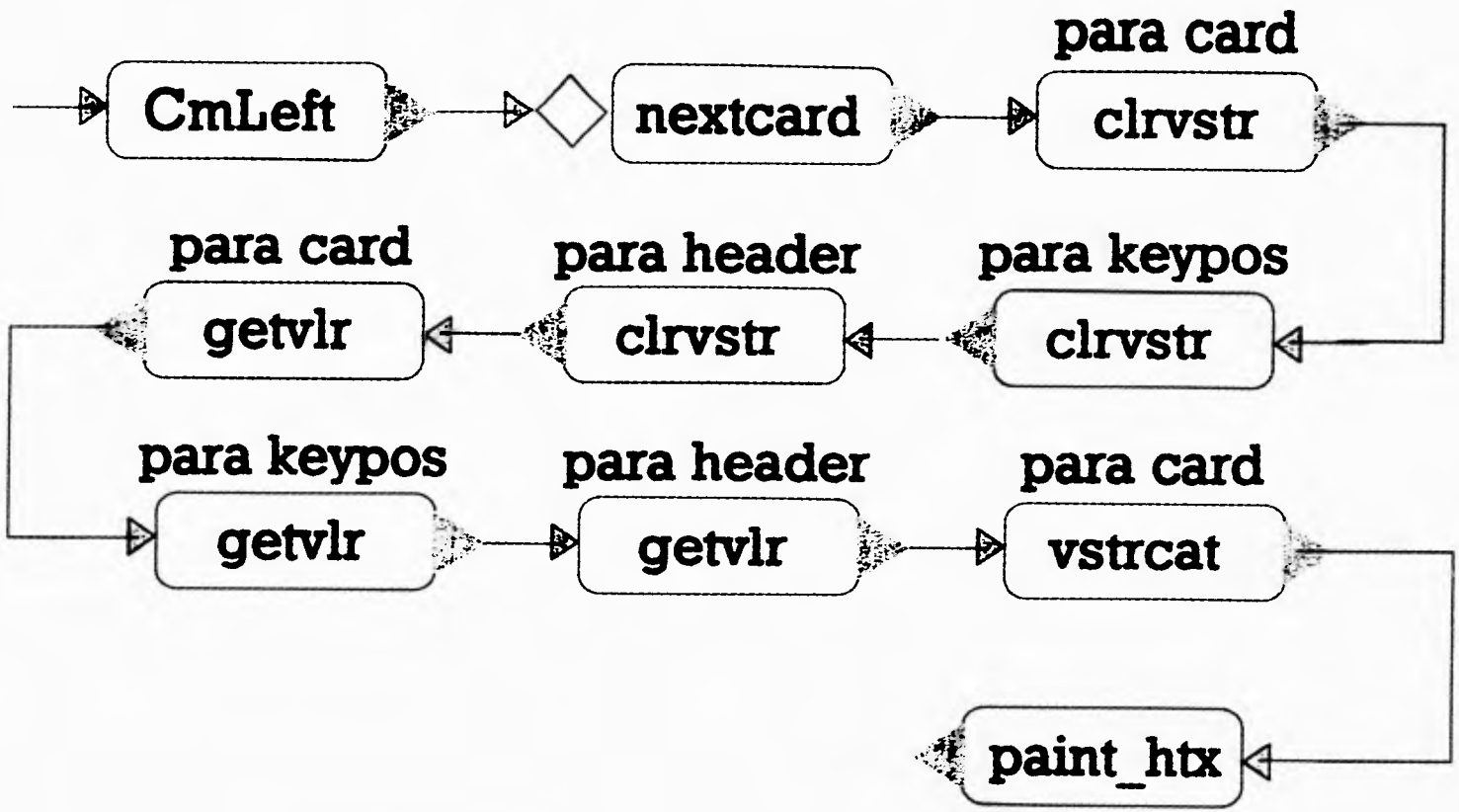


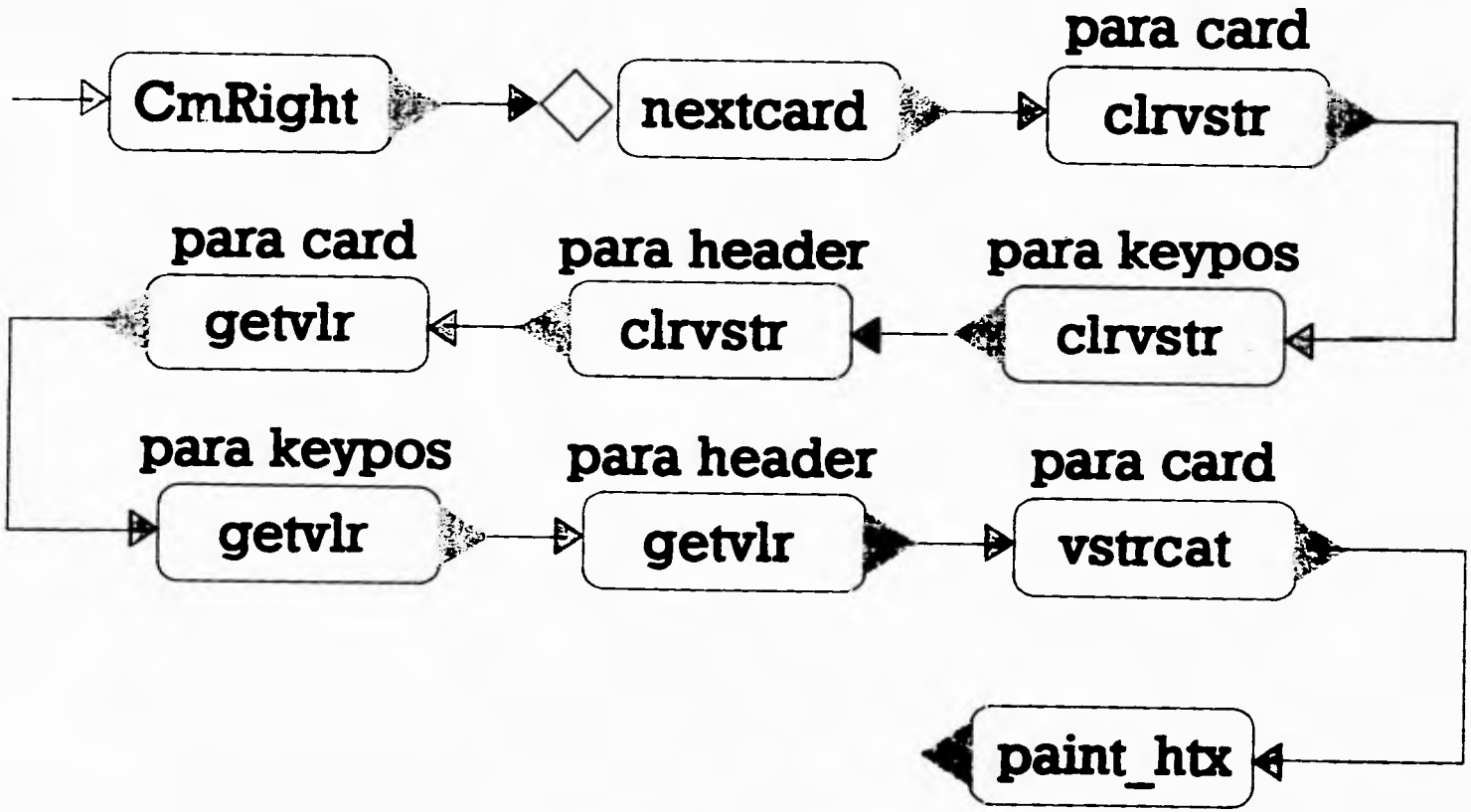

Corre
TBrowser
hb
para keypos
clrvstr
para card
clrvstr
nextcard
para header
clrvstr
para card
getvlr
para keypos
getvlr
para card
vstrcat
para header
getvlr
paint_htx

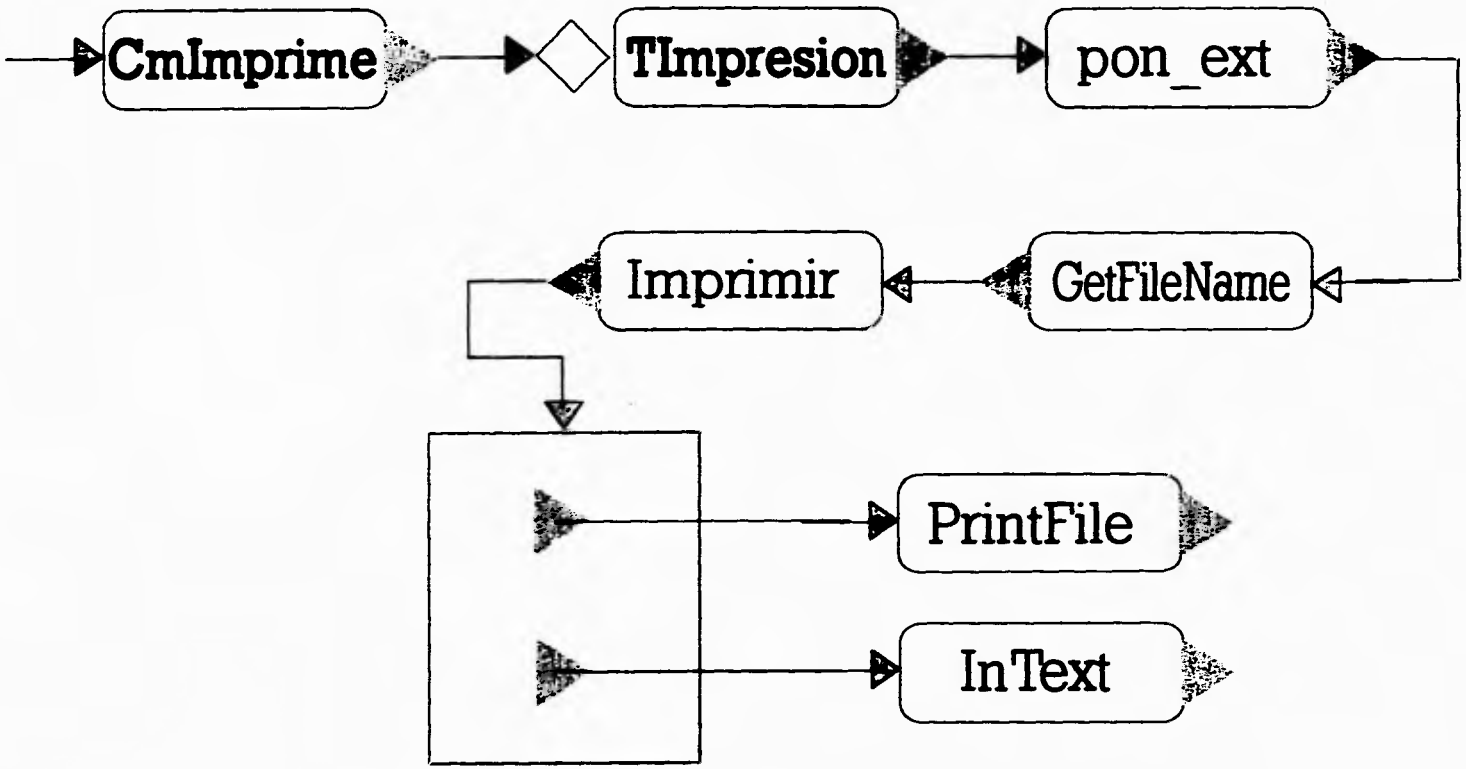

TBrowser
openvlr
para index
clrvstr
readhdr
para index
getvlr
para index
width_height
AssignMenu

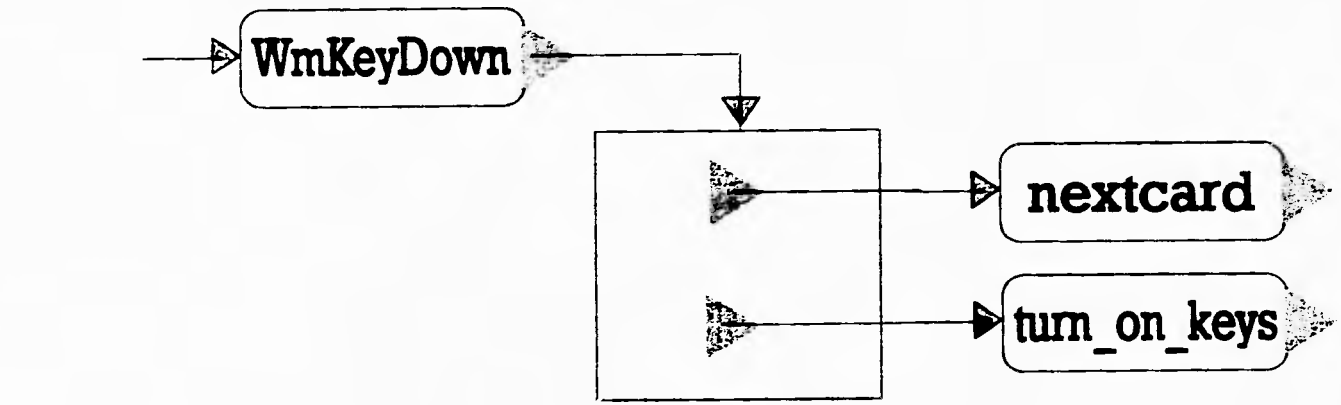

CmIndice
TIndice
TListBox
MessageLoop
HandleListBoxMsg
WMSetFocus
WmKeyDown
HandleListBoxMsg
GetSelIndex
CloseWindow


WMSetFocus
SetFocus
WmKeyDown
GetSelIndex
CloseWindow
CloseWindow

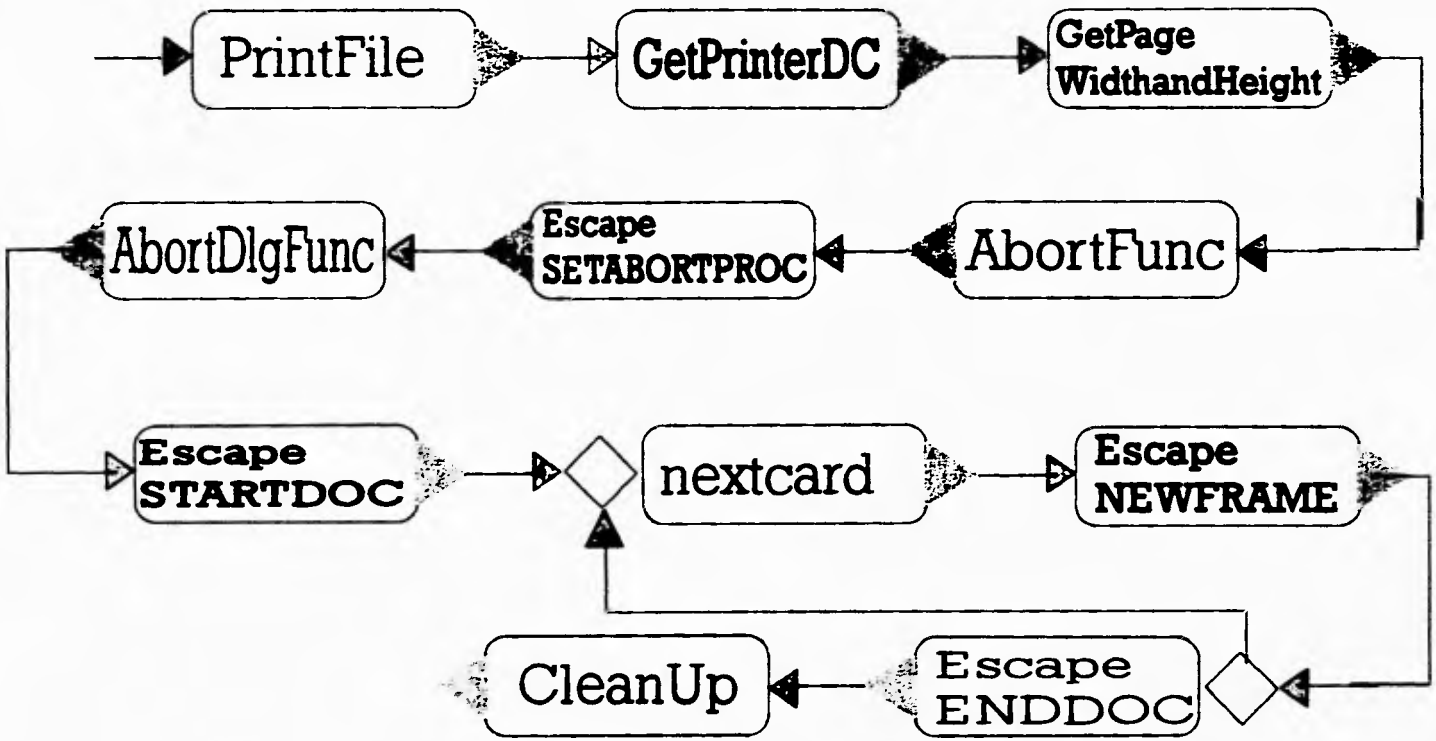

CmLeft
nextcard
para card
clrvstr
para card
getvlr
para header
clrvstr
para keypos
clrvstr
para keypos
getvlr
para header
getvlr
para card
vstrcat
paint_htx

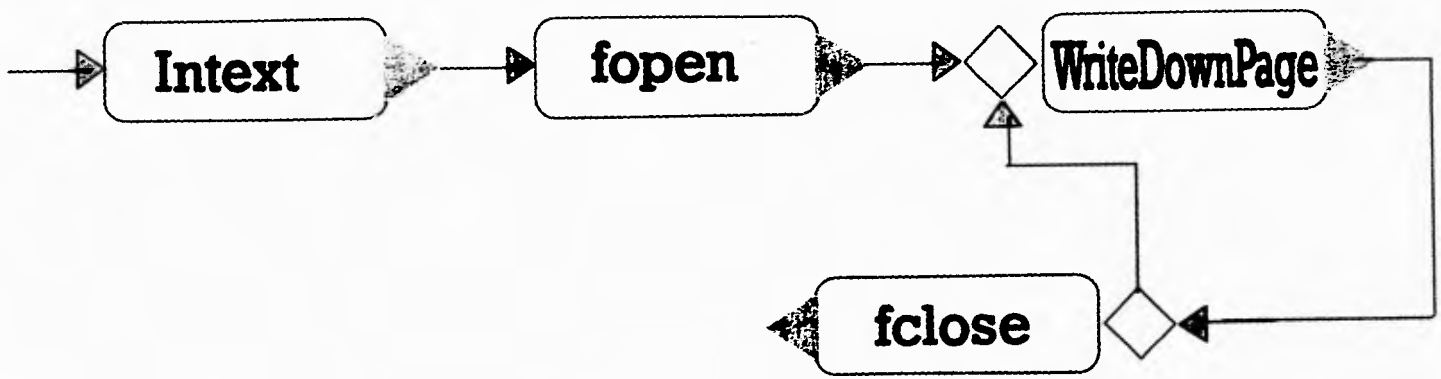

CmRight
nextcard
para card
clrvstr
para keypos
clrvstr
para header
clrvstr
para card
getvlr
para card
vstrcat
para header
getvlr
para keypos
getvlr
paint_htx


CmImprime
TImpresion
pon_ext
Imprimir
GetFileName
PrintFile
InText


WmKeyDown
nextcard
turn_on_keys


WMLButtonDown
on_key
nextcard


PrintFile
GetPrinterDC
GetPage
WidthandHeight
AbortDlgFunc
Escape
SETABORTPROC
AbortFunc
Escape
STARTDOC
nextcard
Escape
NEWFRAME
CleanUp
Escape
ENDDOC


Intext
fopen
WriteDownPage
fclose

CAPITULO 4

DISEÑO

El análisis orientado a objetos posee dos aspectos: El análisis de la estructura de objetos y el análisis del funcionamiento de objetos. En el diseño orientado a objetos sucede lo mismo: El diseño de la estructura de objetos y el diseño del funcionamiento de los objetos. En los lenguajes orientados a objetos, como Turbo C++, se pueden crear estructuras de datos y operaciones, ambos sujetos a la herencia y combinados en unidades llamadas clases.

En el diseño de la estructura y funcionamiento de los objetos, los siguiente componentes son identificados:

1. Las clases que serán implementadas. Los tipos de objetos encontrados en la etapa de análisis ayudarán para tomar estas decisiones.
2. La estructura de datos para cada clase.
3. Las operaciones que cada clase ofrece.
4. ¿Cómo la herencia se implementará y cómo esto afectará las especificaciones de los datos y operaciones?.
5. Identificación de los elementos (clases) reusables.

Mapeando Esquemas en Estructuras de un OOPL

El producir diseños a partir de esquemas es directo, debido que los OOPL (Objet Oriented Programming Languages) están estructurados en una

forma similar a la que conceptualizamos, es decir, un OOPL, como lo es Turbo C++, nos ofrece medios para poder representar objetos dentro de un sistema mediante el uso de clases.

La conversión de el análisis orientado a objetos a estructuras de Turbo C++ es la siguiente:

Elemento de Esquema	Turbo C++
Tipo de objetos	clases
Atributos	campos
Jerarquía (generalización)	clase/subclase herencia
Operaciones	funciones miembro
disparador	tipo de solicitud (no equivalente)
Tipo de eventos	(no equivalente)

El uso de "no equivalente" significa que no es soportado por Turbo C++. Por ejemplo, para poder implementar un esquema de eventos es realmente necesario crear un calendario de eventos. Un calendario de eventos es más que nada un sistema que se basa principalmente en una cola de eventos. Es decir, se introducen eventos en una cola y los eventos extraídos de la cola provocan la activación de los disparadores y estos a su vez disparan la ejecución y pasarán parámetros a sus operaciones respectivas. Un evento puede tener desde uno o más disparadores. En un calendario de eventos, cada evento es un objeto. Cuando un evento es introducido en la cola, es también necesario introducir los súper eventos o súper tipos del evento citado. Es decir, si un evento es derivado

de otros, es necesario incluir en la cola sus súper tipos (toda su ascendencia), y obviamente, es necesario incluir los disparadores de los súper tipos en la cola. Los súper tipos y el evento mismo, incluyendo sus disparadores, son empacados en una sola unidad o paquete. Una vez que un paquete de este tipo es extraído de la cola, el proceso siguiente es desempacar al evento, sus súper tipos y sus disparadores. Una vez terminado el proceso de desempacamiento, los disparadores procederán a activar y ejecutar sus operaciones correspondientes. Recuerde que una operación es el resultado de un evento y que un evento es el resultado de una operación, esto es, para que un evento ocurra es necesario que una operación se ejecute y a su vez para que una operación se ejecute es necesario que un evento ocurra. Un evento dispara uno o más disparadores, un disparador invoca y pasa los parámetros correspondientes a una operación, si la operación fue exitosa un evento ocurrirá y así sucesivamente. Como un evento puede incluir varios disparadores, es posible que en un determinado momento del proceso se ejecuten varias operaciones en paralelo, y para ello se tendrá que poseer un sistema operativo que soporte un medio ambiente de multitarea. Una vez más aclaro que un evento es un objeto y para manejar este tipo de objetos es absolutamente necesario poseer un calendario de eventos. Turbo C++ no posee un mecanismo igual o semejante a un calendario de eventos, por lo cual es necesario implementar uno en un ambiente de objetos. Otra forma simple y modular para implementar un esquema de eventos es mediante la invocación de operaciones dentro de operaciones, es decir, si una operación B necesita ser invocada después de una operación A será absolutamente necesario incluir la llamada a la operación B dentro del código de la operación A. Lo anterior implica que cada vez que se necesite hacer mantenimiento al esquema de condiciones para

la invocación de una operación, será absolutamente necesario modificar el código de la otra operación. Es decir, en un esquema de eventos, algunas veces es necesario cumplir con ciertas condiciones para que una operación se ejecute. Veámoslo de otra forma, bien, muchas veces cuando un evento dispara a un disparador, éste último (el disparador) verifica que ciertas condiciones se cumplan para de esta forma ejecutar dicha operación, si la condición o condiciones fallan, dicha operación no se ejecutará.

. Una vez explicado lo de las condiciones de ejecución de operaciones en un esquema de eventos, usted podrá ver que sin un mecanismo como un calendario de eventos, es necesario implementar estas condiciones dentro de las mismas operaciones. Es decir, el resultado será más que nada un sistema con características estructuradas y modulares. En una forma más clara, el sistema resultante será por ende una mezcla de el enfoque estructurado y de el enfoque orientado a objetos. El sistema final será un híbrido, con lo mejor del mundo estructurado y del mundo de los objetos.

Para que una aplicación sea considerada totalmente dentro del enfoque orientado a objetos, es necesario que se diseñe e implemente un calendario de eventos y a su vez será necesario implementar los eventos como objetos. De esta forma si usted desea modificar el código correspondiente al mecanismo de condiciones que cada disparador posee, no será necesario tocar el código de las operaciones, es decir, serán entes totalmente independientes.

James Martin y James J. Odell afirman que para que un sistema tenga partes (no módulos, porque un objeto es considerado una parte de algo y por

consiguiente si hablamos de módulos estaríamos hablando entonces de un enfoque estructurado) reusables es necesario tener un mecanismo aparte para el manejo de eventos. Si ese fuera el caso, entonces ObjectWindows de Borland no sería considerado como un paquete de clases reusables, ni tampoco serían consideradas las clases fileio, vlr y vstring como clases reusables.

Las bibliotecas de clases de ObjectWindows, fileio, vlr y vstring, son clases totalmente reusables y se pueden usar casi para cualquier sistema. Claro que para poder usar estas clases es necesario, como en todo, cumplir con ciertas reglas, nada es por concesión.

Estas clases son reusables sin siquiera tener un calendario de eventos, yo no estoy diciendo que Martin y Odell están equivocados, simplemente es mi punto de vista. En lo que si estoy de acuerdo es que para que un sistema sea considerado como dentro de un enfoque orientado a objetos es que cada elemento del sistema sea un objeto. Pero ningún sistema que corra bajo un sistema operativo con un sistema de archivos como el que tiene UNIX, MS-DOS o Windows puede ser considerado totalmente con objetos, debido a que un archivo no es derivado de otro u otros archivos (en caso de ser más de uno).

Volviendo al tema del calendario de eventos, usted podrá implementar un esquema de eventos en una forma modular y para ello no será necesario crear un calendario de eventos dentro de una aplicación.

Las Clases de Hyptextw

Las clases identificadas son las siguientes:

Clase: TAplicacion.
Derivada de: TApplication.
Funciones miembro (públicas):
TAplicacon:
LPSTR AName.
HANDLE hInstance y hPrevInstance.
LPSTR lpCmdLine.
int nCmdSwow.
virtual void InitMainWindow.
virtual void InitInstance.

Clase: TVentana.
Derivada de: TFileWindow.
Funciones miembro (públicas):
TVentana:
PTWindowsObject AParent.
LPSTR ATitle y AFileName.
virtual void GetWindowClass:
WNDCLASS& AWndClass.
virtual void CmAyudar() = [CM_FIRST + CM_AYUDAR]
virtual void Compila() = [CM_FIRST + CM_COMPILA]
virtual void Corre() = [CM_FIRST + CM_CORRE]

Clase: TBase.

Estructura de datos (Protegido):

RECT Rectangle.

int cardno, key_pos, headline_pos, Width y Height.

HDC DISPLAY.

HWND Window.

HANDLE OldFont.

BOOL ButtonDown y (público) error.

char file[MAXPATH].

Funciones miembro (públicas):

TBase.

~TBase.

void nextcard.

virtual void paint_htx:

vstr *header, *keypos y *text.

void find_m:

char *string.

void scan_get:

char *table y *page.

int *n.

int on_key:

vstr *keypos.

int x, y.

int weight_heidht:

vstr *index.

Clase: TBrowser.

Derivada de: TWindow y TBase.

Funciones miembro (públicas):

```
TBrower.
~TBrowser.
int hb:
    HDC DC.
  virtual BOOL  CanClose();
  virtual void  CmIndice() = [CM_FIRST + CM_INDICE];
  virtual void  CmLeft()  = [CM_FIRST + CM_LEFT];
  virtual void  CmRight() = [CM_FIRST + CM_RIGHT];
  virtual void  CmImprime() = [CM_FIRST + CM_IMPRIME];
  virtual void  WmKeyDown(RTMessage Msg) = [WM_FIRST +
                                            WM_KEYDOWN];
  virtual void  WMLButtonDown(RTMessage Msg) = [WM_FIRST +
                                            WM_LBUTTONDOWN];
  virtual void  WMLButtonUp(RTMessage Msg) = [WM_FIRST +
                                            WM_LBUTTONUP];
  virtual void  Paint(HDC DC, PAINTSTRUCT & PS);
```

Clase: TIndice.

Derivada de: TWindow y TBase.

Estructura de datos (protegida):

```
PTListBox ListBox.
BOOL    ListBoxState.
```

Funciones miembro (públicas):

TIndice:

PTWindowsObject AParent.

LPSTR ATitle.

HWND Win.

HDC DC.

~TIndice.

virtual void SetupWindow.

virtual void HandleListBoxMsg(RTMessage Msg) = [ID_FIRST + ID_LISTBOX]

virtual void WMSetFocus(RTMessage) = [WM_FIRST + WM_SETFOCUS]

virtual void WmKeyDown(RTMessage Msg) = [WM_FIRST + WM_KEYDOWN]

virtual BOOL CanClose.

Clase: TImpresion.

Derivada de: TBase.

Estructura de datos:

int inicio, fin, CharHeight, CharWidth, milimetersW

int milimetersH y FileLen.

char rango[79] y Text[MAXPATH].

BOOL Selection y estado.

FILE *pFText.

Funciones miembro:

TImpresion:

char *FileName y *ran.

HWND win.

```
~TImpresion.
void GetPageWidthandHeight.
void GetPage.
void WriteDownPage:
        char *page.
        int PageNumber.
void WriteDownPageNumber.
void FillWithBlanksBeforeOf:
        char *string.
void next_card.
virtual void painthtx:
        vstr *header y *text.
int imprimir.
int obten_rango:
        char *rango.
pon_ext:
        char      *arch.
        const char *ext.
int InText.
PBYTE GetFileName:
        char *string.
BOOL GetCharWidthandHeight.
BOOL PrintFile.
BOOL IsHeader:
        char *string.
static BOOL FAR PASCAL _export AbortFunc:
```

HDC hPR.

short nCode.

static BOOL FAR PASCAL _export AbortDlgFunc:

HWND hDlg.

unsigned message.

WORD wParam.

LONG lParam.

HDC GetPrinterDC.

Clase: TCompiler.

Estructura de datos:

HWND Show.

Funciones miembro:

TCompiler.

~TCompiler.

void preprocess:

vstr *index.

void hypertext_scan:

vstr *card, *index y *posn.

void i_directa:

vstr *keypos.

int process_cards:

vstr *index, *card, *posn y *header.

int hc:

HWND Window.

char *FileName.

int numnewlines:

char *s.

int *n y *w.

int headers_scan:

vstr *card y *header.

```
typedef struct fh_struct {
 FILE *fp;
 char *name;
 int recsize;
} file_entry;
```

Clase: fileio

Estructura de datos:

file_entry file_attributes.

Funciones miembro:

int openfile:

char *fname y *access_type.

int recsize.

int closefile.

int iobytes:

logn recno.

unsigned char *d.

int nr e iodir.

int rdstr:

char *str.

int n.

int wrtstr:

char *str.

int rdfstr:

char *format.

int wrtfstr:

char *format.

long movepos:

long recno.

int smode.

Clase: vlr.

Derivada de: fileio y vstring.

Estructura de datos:

int padsize

Funciones miembro:

vlr.

~vlr.

int openvlr:

char *fname y *access_type.

long rs.

int readhdr:

long *fs, *fe, *ita y *nl.

int writehdr:

long fs, fe, ita y nl.

```
int getvlr:
        vstr *s.
        long locn.
int reuse_vlr:
        long locn.
        vstr *w.
        int offset y use_fs.
int delvlr:
        long locn.
int addvlr:
        vstr *w.
        long *locn.
int getvlrelem:
        long pos y *nxtlocn.
        unsigned char *datablk.
int extend_file:
        vstr *s.
        long *locn.
        int  offset.
int pon_ext:
        char      *arch.
        const char *ext.

typedef struct vstr_struct {
  char marker[5];
  unsigned int dimlen;
```

```
  unsigned int currlen;
  int esize;
  int inc;
  void far *data;
} vstr;
```

Clase: vstring.

Estructura de datos:

HANDLE hMemory.

Funciones miembro:

vstring.

~vstring.

int vstr_inited:

    vstr *v.

int dimvstr:

    vstr *v.

    unsigned int dimlen.

    int esize e inc.

int redimvstr:

    vstr *v.

    unsigned int newdimlen.

int clrvstr:

    vstr *v.

int delvstr:

    vstr *v.

int copyvstr:

```
        vstr *t y *s.
int vstrdel:
        vstr *v.
        int  p y n.
vstrins:
        vstr *v.
        int  p.
        void *s.
        int  n.
```

Elementos Reusables

Los elementos reusables, identificados durante el proceso de análisis son las clases:

1. Fileio: Es un sistema básico de archivos. Podría ser considerado como un archivo objeto, el cual posee operaciones permisibles de entrada/salida, y obviamente de construcción/destrucción. Un objeto de este tipo es un archivo sin formato y de texto.

2. Vlr: Es también considerado como un archivo objeto. Vlr se deriva tanto de fileio y de vstring. Un objeto del tipo "vlr" posee las cualidades de un objeto del tipo "fileio" y del tipo "vstring". Un objeto del tipo "vlr" es un archivo con formato y binario, el cual permite operaciones de entrada/salida.

3. Vstring: Un objeto del tipo "vstring" es un bloque de memoria. Las operaciones permisibles sobre un objeto de este tipo, son la de insertar, concatenar, borrar, copiar, crear y destruir (entre otras).

CAPITULO 5

IMPLEMENTACION

El problema fue propuesto por el Fisico Raymundo Hugo Rangel G., quien me facilito cierta bibliografia para el desarrollo de Hyptextw y la cual es la siguiente:

1. Netzahualcoyótl: Manual de el usuario.
   Por Megatron de México.
   Primera edición.

2. Advanced Turbo C (Programmer's Guide).
   Donna Morich, Namir Shamas y Brian Flaming.
   Ed. Wiley, 1988.

Netzahualcoyótl es un creador de asistentes, en la forma de hipertexto. El manual viene acompañado con un disco flexible, el cual contiene la aplicación. Netzahualcoyótl corre sobre MS-DOS. Los asistentes resultantes son más que nada programas residentes en memoria, usted pude activarlos en el momento que usted desee.

El poder tener una aplicación de hipertexto en mis manos me aclaro muchas dudas acerca de este tipo de sistemas.

Mientras que el libro de programación avanzada en C, me proporcionó

conocimientos de cómo usar el ratón, el manejo de ventanas y la ventaja de usar archivos con formato (entre otras cosas). Este libro hace uso del manejo del ratón en base a interrupciones (debido a que el autor dirige sus conocimientos para aplicaciones que corren sobre MS-DOS), cosa que ya no es necesario en Windows, debido a que Windows nos ofrece mensajes en lugar de interrupciones, es decir, para el manejo de el ratón en una aplicación sobre Windows es necesario interceptar y manejar mensajes y no interrupciones, como es en el caso de MS-DOS. En el libro de Advanced Turbo C (Programmer's Guide), la creación de ventanas es con el uso de la función window. En Windows, dicha función ya no es válida debido a que la filosofía es otra.

Lo que verdaderamente me sirvió fueron los tópicos acerca de archivos con formato y el de cadenas de longitud variable.

Una vez teniendo estas bases fue necesario conocer la herramienta a utilizar para el desarrollo de Hyptextw, C++. Más que nada elegí C++ debido a que mi objetivo original fue utilizar el enfoque orientado a objetos para el desarrollo de alguna aplicación. Pero las preguntas que me agobiaban a cada momento fueron, ¿Qué clase de aplicación debería de desarrollar? o ¿Esto irá a ser realmente útil?. Yo ya sabía que la tecnología a usar sería el enfoque orientado a objetos y como herramienta Turbo C++. Mis dudas acerca de la aplicación a desarrollar fueron aclaradas cuando conocí el tema de tesis a desarrollar. Una vez ya conociendo el problema a enfrentar, el siguiente paso sería estudiar el enfoque orientado a objetos, con la ayuda del libro de James Martin y James J. Odell: Object-Oriented Analysis and Design, y estudiar un

libro de Turbo C++, más exactamente el libro de Herbert Schildt: Aplique Turbo C++, el cual es un libro para crear programas que corran sobre MS-DOS. Los programas ejemplo los desarrollé en Turbo C++ 2.0 de Borland.

Más tarde yo compraría el paquete de Turbo C++ 3.0 para Windows de Borland, el cual nos permite crear aplicaciones que corran sobre Windows. Pero aquí surgió otro problema, yo no sabía absolutamente nada de como programar en Windows, la ventaja de este paquete es que viene acompañado con una estructura de trabajo (ObjectWindows) para crear interfaces de usuario y con cinco manuales, los cuales son los siguientes:

1. Turbo C++ 3.0 for Windows (programmer's guide):
   a. Language structure.
   b. Streams library.
   c. Error messages.

2. Turbo C++ 3.0 for Windows (user's guide):
   a. Integrated environment.
   b. Installation.
   c. ObjectBrowser.

3. ObjectWindows for C++ (user's guide):
   a. Tutorials.
   b. Class reference.

4. Turbo Debugger 3.0 for Windows (user's guide).

5. Resource WorkShop (user's guide).

Con la ayuda de los manuales ya mencionados y con la disponibililidad de una estructura de trabajo como lo es ObjectWindows, y además con la fortuna de que este paquete viene con muchos ejemplos ya listos para ser compilados y ejecutados, mi trabajo de aprendizaje fue más fácil. Claro todo esto requirió mucho tiempo de inversión para entender de como debería yo crear una aplicación que pudiera correr propiamente sobre Windows.

ObjectWindows posee muchas cualidades, pero carece (al menos la versión 3.0) de un apropiado conjunto de clases para el manejo de la impresión. Lo cual significó que tuviera que realizar una investigación exhaustiva para crear una clase que representará al objeto de impresión (TImpresion). La programación tuvo que ser hecha en base a primitivas de Windows para de esta forma crear e incluir en el sistema (Hyptextw) al objeto de impresión. Pero nuevamente aquí surgieron nuevos problemas, debido a que yo nunca había programado utilizando primitivas de Windows, pero afortunadamente en la biblioteca de DGSCA tienen una amplia gama de libros y revistas de cómputo. Ahí encontré un libro de Brent Rector: Developing Windows 3.1 Applications with Microsoft C, el cual posee una amplia cobertura sobre como imprimir en un ambiente Windows.

El hecho de que Turbo C++ 3.0 para Windows venga con una estructura de trabajo, como lo es ObjectWindows, no significa que ésta resuelva todo y de que su uso sea sencillo, de hecho esto implica varias cosas:

Disponibilidad de clases: El tener un árbol o jerarquía de clases para el desarrollo de interfaces de usuario, significa que nosotros ya no nos tenemos que preocupar en crear clases cuyas ocurrencias puedan representar elementos de Windows, tales como diálogos, cajas tipo combo, cajas de tipo lista, botones y barras de recorrido (entre otros más elementos que puedan ser representados en la pantalla de un monitor).

Seguimiento de un formato de trabajo: Pare hacer un efectivo uso de ObjectWindows es necesario seguir ciertas reglas, esto es común cuando nosotros tenemos acceso a código hecho por otras personas. Lo cual significa que nosotros debemos conocer lo que nos ofrece esta biblioteca de clases y en que forma podemos utilizar dichas clases. Es absolutamente necesario leer primero el manual de ObjectWindows o libros relacionados al manejo de esta estructura de trabajo. ObjectWindows oculta en gran manera las dificultades de programar en Windows. El programar utilizando las primitivas de Windows es realmente pesado y engorroso. Si usted desea programar en Windows, no basta con conocer las funciones de su API (Application-Programming Interface) y que es lo que hacen, es absolutamente necesario saber en que forma organizar dichas funciones (más de 600) dentro de una aplicación para de tal forma hacer un uso efectivo del API de Windows. Vuelvo a repetir, aunque ObjectWindows oculta gran parte de la complejidad de Windows, éste no deja de ser complejo para un programador que se inicia en Windows, inclusive también para un programador veterano.

La creación de Hyptextw implicó un arduo trabajo de aprendizaje y una etapa de prueba y error, yo diría bastantes etapas de prueba y error. Por

ejemplo, para la reservación de memoria yo utilice las clásicas funciones de Turbo C++, pero para MS-DOS, malloc, calloc y realloc. Estas funciones, aunque son perfectamente aceptadas en un ambiente de Windows, sin embargo, provocan errores en el momento que Windows empieza a hacer movimientos de bloques de memoria. Estos errores solamente surgen cuando se compila en modo Huge, Large o Compact, es decir, cuando tenemos apuntadores a memoria del tipo far, y Windows mueve el bloque al cual nuestro apuntador estaba apuntando. En este momento surgirán errores debido a que dicho apuntador ya no posee una dirección válida. Siempre que nosotros intentamos hacer uso de los recursos del sistema, es absolutamente necesario obtener una identificación de ese recurso, para que de esta forma si en cualquier momento nosotros hacemos referencia al mismo, lo sea mediante ese identificador o handle. La memoria, así como los tipos de letras (fonts), las plumas, iconos, cadenas o los menús, es también un recurso y para evitar los errores ya descritos, es necesario obtener un identificador (handle) para la cantidad de memoria que deseemos. En cualquier momento que usted quiera hacer uso de algún recurso primero obtenga el identificador de este recurso, sin importar si se está trabajando en modo Small o Medium. Es más sano echar mano de lo que nos ofrece ObjectWindows y en caso de que éste no tenga lo que usted busca, eche mano de lo que ofrece Windows.

Los elementos descritos hasta este momento, son los siguientes:

1. Una tecnología: El enfoque orientado a objetos.
2. Una herramienta: Turbo C++ para Windows.
3. Un problema: Una aplicación de hipertexto.

Cada elemento fue una etapa de aprendizaje, y como mencione lineas atrás, y de arduo trabajo.

En muchos proyectos, las personas que trabajan en ellos ya conocen la herramienta y la tecnología a usar, lo único que desconocen es el problema y en muchos casos, para esa clase de personas, el problema a resolver es similar a problemas anteriores, lo cual implica casi cero contratiempos, casi no análisis, casi no diseño y casi no pruebas.

En mi caso es ampliamente diferente, yo no conocía ni la tecnología, ni la herramienta, ni el problema. Esto no quiere decir que ya tenga un amplio dominio de lo ya mencionado. Es decir, para tener un dominio de cierta metodología o de cierta herramienta, es necesario batallar con ellas por un largo tiempo para de esta forma darle filo a nuestras habilidades.

Para dominar ciertas metodologías, no basta con nada más leer. Vuelvo a repetir, para aprender algo realmente bien es necesario sufrir y trabajar con ellas y por ende, hacerlas propias.

En estos momentos yo no tengo un amplio dominio de el enfoque orientado a objetos y de la programación usando primitivas de Windows en base a un lenguaje de alto nivel como lo es Turbo C++, pero espero que en base a futuras batallas con futuros proyectos obtenga un dominio de éstas.

CAPITULO 6

CONCLUSIONES

Ambientes Gráficos

En base al desarrollo del sistema Hyptextw, podemos concluir que todo sistema debe poseer una interfaz de usuario amigable e intuitiva. Es decir, una interfaz debe poseer elementos gráficos que sean reconocibles o familiares al usuario. Por ejemplo, una aplicación podría mostrarle un diálogo que contenga información correspondiente a un empleado o indicándole que debe introducir información en los elementos pertenecientes al diálogo. Estos elementos son ya familiares a un usuario, por ejemplo, cuando usted llena una forma de solicitud de empleo, lo primero que observa son las opciones a marcar con una cruz o una flecha; esto podría ser implementado en un diálogo mediante cajas de verificación (cuando existe la posibilidad de varias opciones simultaneas) o con radio botones (cuando es sólo posible elegir una sola opción). El ejemplo más claro es la interfaz de usuario de Windows, la cual le permite al usuario realizar diferentes tareas sin necesidad de aprenderse comandos, así como en sistemas operativos orientados a comandos como UNIX o el mismo MS-DOS. Para una persona es mucho más fácil utilizar menús, botones, diálogos, ventanas e iconos, que utilizar comandos.

Es más fácil utilizar los elementos gráficos, ya mencionados, con la ayuda de un ratón y el teclado. El usuario, en muchos casos, sólo tiene que revisar lo que tiene en pantalla para realizar las acciones pertinentes, claro,

todo esto en base a la aplicación. Muchas veces, también, es necesario utilizar un manual para entender más que nada la filosofía de un sistema; en cambio un sistema orientado a comandos obliga al usuario a estudiarlos y por ende dedicarle más tiempo al aprendizaje que al uso del sistema. En un ambiente gráfico, un usuario puede iniciar el trabajo sin conocer a la perfección el sistema, veamos un ejemplo:

Un día estando en la casa de un amigo estaba yo jugando con Word para Windows, éste es un procesador de texto el cual yo no conocía en absoluto, pero el ambiente gráfico que posee es muy amigable. En caso de una duda, en lugar de recurrir a mi amigo, sólo tuve que navegar a través de la ayuda, la cual es un sistema de hipertexto, y así en tan solo unos minutos pude crear un texto con imágenes incluidas. Si dicha aplicación no hubiera tenido un ambiente ya familiar para mí, como lo es un medio en Windows, quizás hubiera sido necesario tomar un curso o leer los manuales o en su caso libros para de esta manera crear un texto con imágenes.

En nuestros días muchas personas hacen uso de ciertas aplicaciones gráficas sin tener una licenciatura o maestría en ciencias de la computación. Lo que quiero decir es que los ambientes gráficos son tan amigables que le facilitan la vida a muchas gentes.

Una de mis conclusiones es que todo sistema de software debe y deberá incluir una interfaz de usuario de tal forma que la persona que haga uso de dicho sistema no tenga dolores de cabeza como en antaño.

Objetos

Los objetos han sido, son y serán parte de nuestras vidas. Nosotros siempre hemos pensado en objetos, esto no es nuevo para cada uno de nosotros. Debido a que las personas entienden mejor los problemas cuando estos se abordan por medio de cosas ya conocidas, como lo son los objetos, no sería extraño ver que todos los sistemas de software del futuro sean desarrollados con un enfoque a objetos, y al decir todos, estoy incluyendo desde los poderosos sistemas operativos hasta un simple sistema de nómina.

Quizás los sistemas operativos de hoy en día, como los son UNIX y MS-DOS (entre otros), no desaparescan pero si sufran una metamorfosis hacia los objetos.

El desarrollo de Hyptextw me permitió conocer lo que es el enfoque orientado a objetos, yo ya había oído acerca de ello, pero eso era todo. No sabía en que forma era usado por la industria fabricadora de software y mucho menos conocía las ventajas que pudiera ofrecer. Ahora puedo decir que realmente conozco un poco más acerca de esta filosofía, no estoy diciendo que la domine pero al menos se de lo que se trata.

El enfoque orientado a objetos puede ocupar el lugar del enfoque estructurado (o por bloques lógicos o cajas negras) en una forma total, yo no estoy diciendo que el enfoque orientado a objetos sea mejor que el enfoque tradicional (el cual ha sido utilizado en gran manera por las empresas de software. De hecho las aplicaciones que hasta la fecha existen son de gran

calidad [como lo son UNIX o INFORMIX, los cuales son sistemas que fueron desarrollados en C]), solamente quiero decir que el enfoque orientado a objetos le lleva una gran ventaja al enfoque estructurado, debido que el enfoque orientado a objetos es realmente en la forma que los seres humanos pensamos y por ende es comprensible por todos. Lo cual significa que los sistemas creados bajo esta filosofía serán más fáciles de desarrollar debido a que la información, que fluye entre las personas encargadas de estos proyectos, es más clara.

APENDICE A

GLOSARIO DE TERMINOS

En el presente apéndice se hablará un poco de los elementos comunes en una interfaz de usuario, especialmente una interfaz de usuario compatible con Windows. Al decir compatible con Windows, se tiene que cualquier elemento visual de una interfaz de usuario de una aplicación tiene que tener las mismas características que su contra parte en la interfaz de usuario de Windows, es decir, tienen que ser iguales en funcionamiento y en apariencia. Por ejemplo un menú o una caja de edición en una aplicación tienen que parecerse en funcionamiento y en apariencia a sus contra partes en la interfaz de Windows. De esta forma, un usuario que hace uso normalmente de Windows, se encuentra que cuando tiene acceso a otra aplicación que corre sobre Windows, ya conoce todos los elementos de dicha aplicación. Esta es una gran ventaja de las aplicaciones que corren sobre Windows contra las aplicaciones que corren, por ejemplo, sobre MS-DOS.

Aceleradores: Una forma de ejecutar comandos es por medio de los menús, otra forma lo son los aceleradores. Los aceleradores son formados por combinaciones de teclas que al ser presionadas simultáneamente producen como resultado una acción, por ejemplo: la combinación de teclas, en la mayoría de los editores que corren sobre Windows, <shift><del> eliminan texto marcado para guardarlo en un área de almacenamiento temporal (conocida como el portapapeles).

Barras de recorrido: Las barras de recorrido también son parte de una interfaz de usuario compatible con Windows. Por lo regular, las barras de recorrido son asociadas (no necesariamente) con ventanas. Usted podrá encontrar que algunas cajas de tipo lista poseen una barra de recorrido vertical, esto para recorrer los elementos de la lista. Pero lo más común es verlas con ventanas. La función principal de las barras de recorrido es el de recorrer cierto espacio de exhibición, para así de esta forma poder ver cierta parte del área de exhibición (del elemento mostrado en pantalla) que anteriormente estaba fuera de visión.

Botones de presión: La función principal de este tipo de botones es el de producir cierta acción (la que usted quiera) después que han sido presionados, por lo regular con la tecla entrar (enter) o con el botón izquierdo del ratón. Estos elementos usualmente son asociados con ventanas o cajas de diálogo.

Cajas de verificación: Las cajas de verificación ofrecen otra forma (la otra lo son los radio botones) de seleccionar información, la única diferencia es que le permiten marcar o seleccionar más de una caja de verificación.

Cajas de grupo: Los elementos de grupo se utilizan para agrupar cajas de verificación o radio botones. Las cajas de grupo usualmente tienen una etiqueta que indica el significado del conjunto de radio botones o de cajas de verificación.

Cajas de tipo combo: Las cajas de tipo combo están formadas por otros dos elementos, una caja de edición y una caja de tipo lista. Las cajas de tipo

combo son usadas, por lo regular, para seleccionar algún elemento de la lista de cadenas de caracteres o para introducir una nueva cadena desde la caja de edición. La cadena seleccionada, en la caja de tipo lista, por lo regular es mostrada en la caja de edición. Usted le puede dar a este tipo de elementos el uso que usted quiera, pero un ejemplo muy claro es el uso que se les da para seleccionar archivos para abrir o para salvar archivos con un nuevo nombre, etcétera. El uso ya depende de usted.

Cajas de tipo lista: Es un elemento de la interfaz de usuario de Windows. Su función principal es el de mostrar una lista de cadenas de caracteres. Una caja de tipo lista permite seleccionar una o varias cadenas de caracteres con el ratón o con el teclado. Un cadena de caracteres puede representar cualquier cosa, desde el nombre de un archivo hasta el nombre de alguna fruta.

Diálogos: Los diálogos son ventanas que ofrecen una comunicación, más amigable, entre el sistema y el usuario. Los diálogos pueden contener varios de los elementos pertenecientes a una interfaz de usuario, tales como: cajas de edición, cajas de tipo combo, cajas de verificación, grupos, radio botones, etcétera. Es decir, un diálogo contiene elementos visuales, como los ya citados, para facilitar el entendimiento del usuario.

Elementos estáticos: La mayoría de los elementos visuales de una interfaz de usuario son dinámicos, es decir cambian con el tiempo, debido a ciertas acciones del usuario, de una característica visual a otra, mostrando así una realidad virtual por medio del software, un ejemplo muy claro son los botones y las barras de recorrido. Por el contrario, los elementos estáticos no

poseen una dinámica similar a la de los demás elementos de Windows. La función principal de los elementos estáticos es el de proveer un medio para describir las acciones de los demás elementos. Es decir, los elementos estáticos ofrecen una forma, por ejemplo, de ponerle un título a un diálogo o un letrero a una caja de tipo combo. Los elementos estáticos sirven para poner etiquetas.

Interfaz de usuario: Una interfaz de usuario es un conjunto de elementos visuales que permiten la comunicación entre la aplicación y el usuario.

Mapa de bits: Los mapas de bits son arreglos de bits que contienen información gráfica. Los mapas de bits son almacenados en memoria o en disco para posteriormente ser mostrados, por un programa, en el monitor de una computadora.

Menús: Los menús son parte de una interfaz de usuario. Un menú principal, por lo regular, contiene submenús que a su vez pueden contener otros submenús o comandos. Un comando ofrece a un usuario una forma de ejecutar acciones. Los comandos siempre poseen una etiqueta descriptiva, la cual sugiere la acción que ocurrirá en dado caso de ser seleccionado el comando. Es normal ver un submenú con una lista de comandos. Un menú principal, por lo regular, no necesariamente, es relacionado con una lista de submenús.

Radio botones: Los radio botones son elementos que son parte de una interfaz de usuario. Su función principal es el de proveer al usuario con un mecanismo de selección. Por lo regular usted encontrará un conjunto de radio botones (por lo menos dos), de los cuales usted sólo podrá seleccionar uno

solo. No se preocupe, un radio botón siempre tendrá información descriptiva, para que de esta forma usted pueda seleccionar el radio botón adecuado.

Tipos de letra (fonts): Los tipos de letra son conjuntos de caracteres, cuyos elementos tiene ciertas características en común. Los elementos de un conjunto poseen un mismo estilo, una altura, un ancho promedio (en caso de variar el ancho, en caso contrario todos los símbolos tendrán un ancho fijo o igual) y un ángulo de inclinación de las letras, entre otras características.

APENDICE B

OBJECTWINDOWS

ObjectWindows es una biblioteca de clases que encapsula el comportamiento (a nivel aplicación y a nivel ventana) de las aplicaciones para que funcionen en un ambiente Windows. ObjectWindows es incluido , por Borland, en la versión 3.0 de Turbo C++ para Windows. ObjectWindows nos facilita el desarrollo de aplicaciones debido a que nos ofrece:

1. Una consistente, intuitiva, y simplificada interfaz a Windows.
2. El comportamiento para un manejo de ventanas y el proceso de mensajes.
3. Una básica estructura de trabajo para crear aplicaciones que funcionen en un ambiente Windows.

Nosotros heredamos esta funcionalidad, la cual nos libera para concentrarnos en nuestros esfuerzos, dedicados ha cubrir los requerimientos de nuestras aplicaciones. Al reusar las clases que ObjectWindows nos ofrece, nosotros significativamente reduciremos el tiempo para desarrollar aplicaciones, y por supuesto, tener menos código que mantener.

ObjectWindows es una biblioteca que consiste de una jerarquía de clases que podemos usar, modificar, o agregar, para utilizar las características heredadas de las clases existentes. A continuación se dará un breve bosquejo de la biblioteca de clases de ObjectWindows.

Object

Object es la clase base para todas las clases derivadas y también es definida en el contenedor de la biblioteca de clases.

TModule

Las aplicaciones, bajo ObjectWindows, construyen una ocurrencia de TAplication, derivada de TModule. TModule define el comportamiento de nuestras aplicaciones. Algunas de las características de las funciones miembro de TModule, son el manejo de memoria y el proceso de errores.

TAplication

Esta clase define el comportamiento requerido por nuestras aplicaciones. El primer requisito para crear aplicaciones es el de derivar una clase de TAplication. Nuestra clase derivada deberá tener un constructor, el constructor creará un objeto, en realidad este objeto es nuestra aplicación. La aplicación heredará:

1. Creación y exhibición de la ventana principal.
2. Inicialización de la primera ocurrencia de una aplicación.
3. Inicialización de cada ocurrencia de una aplicación; por ejemplo, cargar una tabla de aceleradores.

4. Procesamiento de los mensajes que Windows pasa a la aplicación.
5. Cerrar la aplicación.

Además de definir una clase derivada de TAplication, usted debe de agregarle la habilidad de construir la ventana principal, la ventana principal es otro objeto. Usted tiene la opción de redefinir el comportamiento de la ventana principal.

El programa principal de su aplicación de ObjectWindows, normalmente consistirá de sólo tres instrucciones.

La primera instrucción llama al constructor de la aplicación, para crear el objeto que representará a nuestra aplicación. El constructor también se encarga de inicializar a la estructura de datos de la aplicación.

La segunda instrucción llama a la función miembro Run, la cual llama a InitAplication y a InitIntance para realizar a la primera ocurrencia y a las ocurrencias restantes de la aplicación, respectivamente. InitMainWindow es entonces invocada por InitInstance para crear la ventana principal. En la mayoría de los casos usted necesitará redefinir el comportamiento de la ventana principal.

Run pone a la aplicación en movimiento al llamar a la función miembro MessageLoop que procesa todos los mensajes que recibe la aplicación. MessageLoop invoca a las funciones correctas, según el tipo de mensajes. El tiempo de vida de MessageLoop es la duración de la aplicación. Una vez que cerramos la aplicación, la función MessageLoop deja de existir.

La tercera instrucción retorna el estado de la aplicación. El estado final queda almacenado en la variable miembro Status. El estado de la aplicación es muy importante ya que WinMain debe retornar un entero al sistema operativo Windows.

Funciones y Datos Heredados por la Aplicación

+ Run es una función miembro de TAplication heredada a nuestra clase derivada. Inicializa la primera ocurrencia de nuestra aplicación al llamar a InitAplication y si se encuentran otras ocurrencias en ejecución (por supuesto de nuestra aplicación), entonces llamará a la función InitInstance. Y si la inicialización tuvo éxito, pone a la aplicación en movimiento al llamar a la función MessageLoop.

+ InitAplication es una función miembro de TAplication heredada a nuestra clase derivada. Realiza cualquier inicialización necesaria solamente para la primera ocurrencia de la aplicación. Por omisión, InitAplication no hace nada, usted tiene que definir el comportamiento de esta función.

+ InitInstance es una función miembro de TAplication heredada a nuestra clase derivada. Realiza cualquier inicialización necesaria para cualquier ocurrencia de la aplicación. InitInstance llama a InitMainWindow, y crea y muestra a la ventana principal al llamar a TModule::MakeWindow y TWindowsObject::Show. Si la ventana principal no puede ser creada, la variable Status será puesta a EM_INVALIDMAINWINDOW. Si usted redefine a esta función, asegurese de llamar a TAplication::InitInstance.

+ InitMainWindow es una función miembro de TAplication heredada a nuestra clase derivada. Por omisión, InitMainWindow construye a una ventana principal con características básicas, la ventana es construida por medio del constructor de TWindow. La mayoría de las veces usted tendrá que redefinir a InitMainWindow para construir una ventana principal más útil.

+ MessageLoop es una función miembro de TAplication heredada a nuestra clase derivada. Realiza una iteración de rastreo de mensajes, la duración de la iteración es igual a la duración de nuestra aplicación. Inspecciona a Windows por mensajes que pudieran ser mandados a la aplicación. Si un mensaje es recibido, llamará a ProcessAppMsg de lo contrario llamará a IdleAction para realizar algún proceso en tiempo de inactividad.

+ Status es una variable miembro de tipo entero de la clase TModule heredada tanto a TAplication como a nuestra clase derivada. Status contiene el estado de algún módulo. El estado puede ser puesto por ObjectWindows para indicar algún error en la creación de la ventana principal de alguna aplicación. Este puede ser puesto para indicar algún error en el arranque de un módulo de la biblioteca de DLL (Dinamic-link Libraries).

TWindowsObject

TWindowsObject es una clase base y es común a todas las clases que encapsulan el comportamiento de los objetos que sirven para crear interfaces de usuario, estas clases son: TDialog, TWindow, y su clase derivada, TControl. TWindowsObject define el comportamiento básico y común a las ventanas,

diálogos y controles. Las funciones miembro de TWindowsObject realizan lo siguiente:

+ Mantienen la estructura dual de objeto/Windows, donde objeto pertenece a ObjectWindows, mientras tanto que el elemento visual es de Windows. Estas funciones se encargan de destruir y crear tanto al objeto (1) como al elemento (2).

+ Automáticamente soportan relaciones padre e hijo entre los objetos que sirven de interfaz (esto no quiere decir herencia de clases).

+ Registran nuevas clases de Windows.

El trabajo de TWindowsObject es tras bambalinas, usted no lo notará. Usted raramente derivará nuevas clases de TWindowsObject. Esta clase define mucha de la funcionalidad que un objeto hereda cuando usted deriva nuevas clases de TWindow y TDialog.

---

(1) En este caso muy especial, los objetos se encuentran atrás de los elementos.

(2) Un elemento es aquel que podemos observar en el monitor de una computadora, tales como: cajas de diálogo y controles.

TWindow

TWindow es un clase de propósito general para ventanas, cuyas ocurrencias pueden ser ventanas principales, ventanas presto, o ventanas descendientes (o hijos) de una aplicación. Ocurrencias de TWindow pueden mostrar gráficas, pero usted puede especializar el comportamiento de sus ventanas al derivar nuevas clases de TWindow.

TEditWindow

TEditWindow es derivada de TWindow y además permite definir una clase que permite editar texto en una ventana. Ocurrencias de TEditWindow son ventanas que permiten edición de texto pero no permiten leer y escribir en un archivo.

TFileWindow

TFileWindow es derivada de TEditWindow. TFileWindow define a una clase que permite la edición de texto en una ventana, pero también permite salvar el texto creado en un archivo y cargar un archivo del disco.

TDialog

Esta clase sirve como base para clases derivadas que manejan cajas de

diálogos propias de Windows. Los objetos de diálogo (1) son asociados con recursos de diálogo, y pueden ser ejecutados como cajas de diálogo con modelo o como cajas de diálogo sin modelo. Las funciones miembro son capaces de manejar la comunicación entre diálogos y sus controles (2).

TFileDialog

TFileDialog es un clase de diálogo que es muy útil en muchas aplicaciones. Esta clase define un diálogo que le permite al usuario seleccionar un archivo para cualquier propósito, tales como abrir, editar y salvar a un archivo. Los diálogos del tipo TFileDialog aceptan tres parámetros:

+ Un apuntador a la ventana padre.
+ Un identificador de recurso (SD_FILEOPEN o SD_FILESAVE).
+ El nombre de un archivo o una mascara (*.*, *.cpp, *.pas, programa.*, etcétera).

---

(1) En ObjectWindows, muchas veces las imágenes que podemos ver en el monitor de una computadora son objetos puros (sin utilizar archivos de recursos); pero en otras ocasiones las imágenes que se observan (tales como controles, cajas de diálogo, etcétera) no son objetos puros, dichas imágenes son una mezcla de objetos y recursos (los recursos se encuentran en otro archivo y que son ajenos al código de C++).

(2) Los diálogos son objetos y como ya sabemos muchas veces los objetos contienen a otros objetos; como por ejemplo un motor contiene pistones, bujías, cables, etcétera, así los diálogos pueden contener controles como botones, los botones entre otros controles forman parte de los diálogos.

TInputDialog

Los "diálogos de entrada" son diálogos ya definidos y que vienen incluidos en OjectWindows. Estos diálogos simplemente leen una cadena de texto del teclado. Esto es una tarea común en una aplicación bajo Windows. Los "diálogos de entrada" aceptan cinco parámetros:

+ Un apuntador a la ventana padre.
+ Un apuntador al texto que es el titulo de la ventana.
+ Un apuntador al texto que funciona como indicador.
+ Un apuntador a una área de memoria que será llenada por el texto que el usuario introducirá.
+ La longitud del texto de entrada.

TControl

TControl es una clase abstracta que sirve como una clase base y común para todos los objetos de control, incluyendo cajas de tipo lista y botones. Esta clase define funciones miembro que pueden crear controles y pueden procesar mensajes para sus clases derivadas.

TButton

Ocurrencias de TButton (1), representan botones de presión de Windows. Este tipo de objetos se seleccionan con el ratón, dando la sensación de que se está presionando el botón que se encuentra mostrado en la pantalla de

(1) Clase derivada de TControl.

la computadora. Los botones de presión se pueden utilizar para realizar cualquier cosa: salir de alguna ventana o actualizar una base de datos, usted les dará una aplicación adecuada a este tipo de objetos en sus programas.

TCheckBox

TCheckBox es derivada de TButton, ocurrencias de TCheckBox representan cajas de verificación de Windows y provee de funciones miembro para manejar el estado de las cajas de verificación.

TRadioButton

TRadioButton es derivada de TCheckBox, ocurrencias de TRadioButton manejan la creación y el manejo del estado para los radio botones. Los objetos de este tipo se utilizan en un ambiente de Windows para realizar selecciones. La selección se realiza al activar un botón del tipo radio (circular).

TListBox

Ocurrencias de TListBox(1) representan a una caja de tipo lista perteneciente a Windows. A estos objetos se les conoce como cajas de tipo lista, debido a que exhiben una lista de cadenas de caracteres dentro de un marco; las cadenas de caracteres pueden representar cualquier cosa (nombres de archivos, nombres de personas, nombres de figuras geométricas, etcétera). Esta clase define la creación de este tipo de objetos y a su vez define algunas

funciones miembro para manipular a los objetos de su lista. Este tipo de objetos se utilizan en un ambiente de Windows para realizar selecciones de objetos que se encuentran en una lista (se puede seleccionar de uno a más objetos de la lista).

TComboBox

TComboBox es derivada de TListBox, TComboBox define el comportamiento para objetos de este tipo. Una caja tipo combo está formada por dos controles, uno de ellos es una caja de tipo lista y el otro control es una caja de edición. En la caja de tipo lista, al seleccionar un objeto de ella, éste es mostrado en la caja de edición. Las cajas tipo combo, pertenecen a un ambiente del tipo Windows.

TScrollBar

TScrollBar (1) define funciones miembro que manejan el rango y la posición del thumb de una barra de recorrido.

TStatic

TStatic (2) define funciones miembro que ponen, solicitan y borran el texto de un control de tipo estático (sólo salida).

---

(1) Clase derivada de TControl.

(2) Clase derivada de TControl.

TEdit

TEdit es derivada de TStatic, TEdit provee de unas capacidades de proceso de texto para el control del tipo de edición.

TGroupBox

Ocurrencias de TGroupBox (1) representan cajas de tipo grupo. Los objetos de este tipo se utilizan para agrupar controles tales como: radio botones o cajas de verificación.

TMDIFrame

TMDIFrame provee de un comportamiento adecuado a la ventana principal de una aplicación que sigue la especificación MDI(2) de Windows.

TMDIClient

TMDIClient provee de un adicional soporte para ventanas que manejan la norma MDI. El cliente es el objeto que maneja el área principal de una ventana, la cual es la que contiene los documentos.

---

(1) Windows maneja un estándard para el control múltiple de ventanas que funciona dentro de una estructura de trabajo de una sola ventana. Este estándard es conocido como The Multiple Document Interface (MDI).

TScroller

TScroller es la clase que da vida a las barras de recorrido de una ventana, dando una forma automática para recorrer el texto y gráficas que usted pone en una ventana. TScroller también recorre a una ventana cuando el usuario arrastra el ratón hacia fuera de la ventana, por consiguiente, TScroller funciona para cualquier ventana, aún sin barras de recorrido.

APENDICE C

PRINCIPIOS BASICOS DE LA PROGRAMACION EN UN AMBIENTE DE WINDOWS

Funciones de Windows API

Existen más de 600 funciones en el Windows Aplication Programming Interface. Una aplicación en Windows, manipula su medio ambiente, modifica su apariencia, o actúa en respuesta a entradas de parte del usuario. Estas entradas podrían ser, por dar un par de ejemplos, en base a un ratón o a un menú. Al presionar cualquier botón del ratón, varios mensajes son mandados a la ventana activa (muchas veces la ventana principal); el objeto que representa a la ventana posee funciones que responderán a los mensajes propiciados por la manipulación del ratón, estos mensajes pueden ser e indicar, por ejemplo, la presión de uno de los botones del ratón o el arrastre del mismo ratón (el movimiento del ratón en el escritorio); otra información sería la posición (coordenadas x e y) del cursor en pantalla, esto es debido a que al mover el ratón el cursor también se mueve. Más que un mensaje, las coordenadas se envían junto con el mensaje en cuestión, indicando que hubo un movimiento. Usando ObjectWindows, usted puede crear ventanas, mostrar cajas de diálogo y manipular controles sin llamar directamente a las funciones del Windows Aplication Programming Interface.

ObjectWindows llama a funciones de Windows

Muchas funciones miembro de ObjectWindows llaman a funciones de Windows. Esto no quiere decir que ObjectWindows esté duplicando funcionalidad, más bien, está empaquetando a algunas funciones de Windows en una estructura de trabajo orientada a objetos. ObjectWindows, grandemente, simplifica la tarea de especificar los numerosos parámetros requeridos por las funciones de Windows. Muchas veces ObjectWindows pasa automáticamente dichos parámetros, liberándolo a usted de tal tarea.

Acceso a funciones de Windows

Todas las funciones del Windows Aplication Programming Interface, están disponibles a un programa en ObjectWindows. Por ejemplo el siguiente código llama a una función de Windows, conocida como MessageBox:

```
respuesta   =   Messagebox(HWindow, "Desea salvar?", "El archivo
                a cambiado", MB_YESNO|MB_ICONQUESTION);
```

Muchas funciones de Windows tienen importantes valores de retorno. En este caso la función MessageBox, regresa un valor del tipo entero que es almacenado en la variable "respuesta", el valor de este entero indica la acción que el usuario tomo al cerrar la caja de mensaje. Si el usuario presiono el botón "Yes", el resultado será igual a IDYES (que es una constante con un valor ya definido por Windows). Si el usuario presionó el botón "No", el resultado es IDNO.

Constantes de Estilo

Funciones de Windows que producen elementos de interfaz, usualmente requieren de algunos parámetros de tipo WORD o del tipo DWORD, estos parámetros son para el estilo de exhibición. Windows define cientos de constantes de estilo. Los identificadores, para estas constantes de estilo, consisten de un mnemónico (prefijo de dos letras) y de un subrayado seguido de un nombre descriptivo. Por ejemplo, WS_POPUP es una constante de estilo; donde WS indica estilo de ventana para una ventana presto.

Estos estilos pueden ser combinados usando el operador de bits | (or) para producir otro estilo. Por ejemplo en la función MessageBox, combinamos dos constantes de estilo para producir un estilo diferente: MB_YESNO|MB_ICONQUESTION. Este estilo produce a una caja de mensaje con dos botones, Yes y No, y un signo de interrogación (icono).

Tipos de Funciones en Windows

Esta sección discute brevemente las clases de funciones que Windows nos ofrece y que pueden ser utilizadas en sus programas hechos en ObjectWindows (1).

---

(1) En ObjectWindows se utiliza un formato poco distinto al de Turbo C++. Por ejemplo en Turbo C++ se utiliza main (...) {...} para el módulo principal, mientras que en ObjectWindows se utiliza WinMain (...) {...} para el módulo principal. Una aplicación en ObjectWindows siempre recibe cinco parámetros de parte del sistema operativo Windows.

Las funciones que manejan mensajes, manipulan ventanas y cajas de diálogos, y generan salida del sistema son conocidas como funciones de interfaz. Esta categoría incluye funciones para menús, cursores, y el manejo de el portapapeles.

Las funciones que muestran texto, gráficas y mapas de bits en una variedad de dispositivos, incluyendo la pantalla y la impresora son conocidas como funciones de interfaz direccionadas a dispositivos gráficos. Estas funciones no están ligadas a un dispositivo en particular, por que son independientes del dispositivo que se maneje en un preciso instante.

Las funciones que manejan un amplio rango de los servicios del sistema, incluyendo manejo de memoria, comunicación con el sistema operativo, manejo de recursos, y comunicaciones son conocidas como funciones de interfaz para servicios del sistema.

Funciones de Tipo Callback

Las funciones callback, son creadas por el usuario. Estas funciones son invocadas por rutinas que son externas al programa o aplicación. Tipicamente Windows invocará a tales funciones.

Cuando una función callback es invocada, tal llamada es de un contexto diferente al del programa. Para que una función del tipo callback pueda tener acceso a las variables "globales" de tú programa (las variables que residen en el

segmento de datos de el programa), debes de pasar la dirección del segmento de datos(1) de la aplicación. Un Thunk (ver pie de página) es utilizado para que el registro de segmento de datos apunte al segmento de datos de tú aplicación y entonces transfiera el control a la función callback. Los thunks son creados por medio de un función de Windows y que es MakeProcInstance, veamos un ejemplo:

```
FARPROC ThunkPtr = MakeProcInstance((FARPROC)
                   Mi_prog_callback,GetAplication()->hInstance)
```

El primer argumento de MakeProcInstance es un apuntador a la función callback, el segundo parámetro es un identificador de la aplicación.

Todas las funciones de callback deben ser exportadas usando la palabra clave _export. Para todas las funciones del tipo callback, se asume que siguen el formato de llamada de Pascal, para eso es necesario declararlas de tipo PASCAL.

Mensajes de Windows

Cuando Windows determina que un evento a ocurrido y que éste afecta a la aplicación, Windows manda un mensaje a la aplicación. Existe una gran variedad de mensajes, los cuales provienen de una variedad de fuentes:

+ El usuario, al mover o presionar un botón del ratón o al teclear en el teclado, genera un mensaje del tipo evento de usuario.

+ Tú programa puede mandarse mensajes así mismo.

+ Tú programa puede llamar a funciones pertenecientes a Windows que resulten en funciones que reciben mensajes de Windows.

+ Tú aplicación puede mandar mensajes a otras aplicaciones a través del DDE (Dynamic Data Exchance).

+ El sistema operativo Windows puede mandar una variedad de mensajes a tú aplicación.

Una aplicación en un ambiente Windows es responsable en recibir y responder a los mensajes que le son mandados. Existe una forma de responder a los mensajes recibidos, ésta se logra al definir funciones que se encarguen de responder de una forma adecuada a dichos mensajes.

ObjectWindows se encarga de manejar este engorroso problema por usted. ObjectWindows ofrece un proceso por omisión para el adecuado manejo de mensajes recibidos por nuestra aplicación. Usted sólo necesita definir las respuestas que su aplicación le dará a los mensajes recibidos, cada respuesta es única para cada tipo de mensaje. Renglones atrás se menciono que es necesario definir ciertas funciones para poder responder a los mensajes que recibe nuestra aplicación, estas funciones tienen que ser definidas en las clases que nosotros derivamos de las clases de ObjectWindows. Estas funciones son conocidas como funciones de respuesta. En dado caso que no se definan funciones de respuesta (por parte del usuario), la aplicación se comportará en una forma ya definida, debido a que ObjectWindows ya tiene funciones de respuesta por omisión que serán heredadas a las clases que derivemos (esto en dado caso que no definamos nuestras propias funciones).

Tipos de Mensajes en Windows

Esta sección da una breve explicación de los diferentes tipos de mensajes en un ambiente de Windows.

Mensajes de administración: Este tipo de mensajes indican que el estado de una ventana a cambiado. Por ejemplo, WM_CLOSE es mandado cuando una ventana es cerrada y WM_PAINT es mandado cuando parte de una ventana (o toda la ventana) necesita ser redibujada.

Mensajes de inicialización: Estos mensajes, incluyen (entre otros) a WM_CREATE y WM_INITDIALOG, son mandados cuando un programa crea a una ventana o una caja de diálogo.

Mensajes de entrada: Estos mensajes son recibidos como resultado de una entrada a través del ratón, del teclado, barras de recorrido o del reloj del sistema. Uno de los más comunes mensajes de entrada es WM_COMMAND, el cual es el resultado de la selección de un menú o de un acelerador, o de un evento realizado por un control, semejante al de presionar un botón.

Mensajes del sistema: Estos mensajes son enviados en respuesta a las acciones realizadas, por el usuario, sobre una ventana, tales como: minimizar o maximizar una ventana. El mensaje enviado, por ejemplo, es WM_SYSCOMMAND. La mayoría de las aplicaciones no interceptan mensajes del sistema.

Mensajes de el portapapeles: Estos mensajes son enviados cuando otra aplicación intenta tener acceso a los datos que tú aplicación posee o que está leyendo en ese instante.

Mensajes de información del sistema: Estos mensajes son enviados cuando un atributo (colores o los tipos de letras) del sistema ha cambiado.

Mensajes de manipulación del control: Estos mensajes son enviados por una aplicación a sus controles. Cada control (hay que recordar que los controles son objetos) entiende su propio conjunto de mensajes.

Mensajes de notificación del control: Estos mensajes son enviados por los controles a su ventana padre (no madre, debido a que en el ambiente de objetos se maneja de esta forma), para indicarle que un evento a sucedido. Por ejemplo una caja de tipo lista le indicará a su padre que un objeto de su lista a sido seleccionado.

Mensajes de notificación de las barras de recorrido: Estos son mensajes especializados para indicar ciertos eventos, tales como: WM HSCROLL y WM_VSCROLL.

Mensajes no pertenecientes a una área cliente: Estos mensajes, incluyen a WM_NCMOUSEMOVE y WM_NCPAINT, son similares a los mensajes de entrada, estos mensajes son enviados cuando se actúa fuera del área cliente de una ventana, esto incluye a la barra del menú (no el menú) de la ventana, a la barra donde se encuentra el titulo de la ventana y los bordes de la misma

ventana. Usualmente, usted no necesitará redefinir a las funciones que se encargan de responder a este tipo de mensajes.

Mensajes de la interfaz de los documentos múltiples: Estos mensajes son enviados por una aplicación que sigue la convención MDI (Multiple Document Interface). Entre los mensajes que siguen esta norma, son: WM_MDIACTIVATE y WM_MDIDESTROY.

Procesamiento de Mensajes por Omisión

Windows nos ofrece una manera de responder a los mensajes que son enviados a una aplicación. Cuando no definimos funciones de respuesta a mensajes, ObjectWindows se encarga de invocar a funciones de Windows. ObjectWindows siembre se encargará de proveer ese comportamiento (para las respuestas en este caso) para sus objetos. El apropiado proceso, por omisión, de mensajes para un control, una ventana, o una caja de diálogo es especificado por su función miembro DefWndProc (propia a cada objeto, recuerde que cada uno de estos objetos son ocurrencias de alguna clase y que cada clase tiene sus propias funciones. Algunas funciones pueden tener el mismo nombre, esto se debe a una maravilla conocida con el nombre de polimorfismo).

Sin embargo, cuando usted define un método de respuesta para algún mensaje de entrada, usted podría querer ejecutar el comportamiento normal, aparte del que usted definió. Para realizar eso, usted necesitará llamar a la función de respuesta de ObjectWindows antes de su código.

Rangos para los Mensajes

Constantes	Valor	Rango	Significado
WM_FIRST	0x0000	0x0000-0x03FF	Mensajes de Windows
WM_USER	0x0400	0x0400-0x7F00	Definidos por el programador
WM_INTERNAL	0x7F00	0x7F00-0x7FFF	Reservado para uso interno
ID_FIRST	0x8000	0x8000-0x8EFF	Definidos por el programador
ID_INTERNAL	0x8F00	0x8F00-0x8FFF	Reservado para usointerno
NF_FIRST	0x9000	0x9000-0x9EFF	Definidos por el programador
NF_INTERNAL	0x9F00	0x9F00-0x9FFF	Reservado para uso interno
CM_FIRST	0xA000	0xA000-0xFEFF	Definidos por el programador
CM_INTERNAL	0xFF00	0xFF00-0xFFFF	Reservado para uso interno

Windows le permite a usted definir sus propios mensajes para uso de sus aplicaciones. Estos mensajes son muy útiles cuando se define y se responde a un nuevo evento.

Dentro del permisible rango para constantes de mensajes de Windows, existe un espacio o rango para que el usuario cree sus propias constantes de mensajes. El rango reservado para mensajes predefinidos de Windows, es de 0 a WM_USER-1. El rango permitido para que el usuario defina sus propias constantes de mensajes es de WM_USER a WM_USER + 31,744. Es mejor definir constantes, iniciando con los valores WM_USER, WM_USER + 1, WM_USER + 2, etcétera. Por ejemplo:

```
#define WM_MIPRIMERMENSAJE    WM_USER
#define WM_MISEGUNDOMENSAJE   WM_USER + 1
#define WM_MITERCERMENSAJE    WM_USER + 2
```

Recuerde que Turbo C++ nos permite hasta 31 caracteres para utilizarlos como identificadores para variables, constantes y funciones.

Para recibir y responder a los mensajes relacionados con las constantes anteriores, veamos el siguiente ejemplo:

```
class MiVentana
{
.
.
.
  virtual void WMMiPrimerMensaje(RTMessage Msg) =
    [WM_FIRST + WM_MIPRIMERMENSAJE];
  virtual void WMMiSegundoMensaje(RTMessage Msg) =
    [WM_FIRST + WM_MISEGUNDOMENSAJE];
  virtual void WMMiTercerMensaje(RTMessage Msg) =
    [WM_FIRST + WMMITERCERMENSAJE];
.
.
.
};
```

APENDICE D

INTRODUCCION AL USO DE UN PAQUETE DE HIPERTEXTO Y UNA INTRODUCCION BASICA DE LO QUE ES HIPERTEXTO

Pasos Para Utilizar el Hypertext

El paquete de Hyptextw posee ciertas características que le pueden ser ya comunes a un usuario de Windows. Si usted ya conoce la filosofía Windows, usted podría saltarse esta sección sin ningún problema.

Aquí no se intentará cubrir la filosofía Windows, si usted no conoce el sistema operativo Windows, usted debería de consultar antes que nada al manual de usuario de Windows. Aquí solamente se describirá de como usar el paquete Hyptextw.

Para tener acceso a cualquier opción del menú, usted deberá pulsar las teclas <ALT> y la tecla que hace referencia a la letra subrayada de la opción, por ejemplo, si usted desea compilar su archivo, usted tendrá que presionar simultáneamente las teclas <ALT> <C> o si usted desea pasar a la sección de archivos, usted tendrá que presionar simultáneamente las teclas <ALT> <A>, donde, tanto C como A son las letras subrayadas en el nombre de la opción elegida por usted. Si usted deseara abandonar el sistema, usted tendría que

pulsar las teclas <ALT><A> para seleccionar la opción archivos (la opción de archivos es un submenú que tiene opciones tales como: Nuevo, abrir y salir (entre otras más)), y una vez situado en este submenú, seleccione la opción salir, ya sea por medio de las flechas (y presionando return) o presionando la letra subrayada de esta opción (en este caso la letra l). Existe una forma más rápida para abandonar el sistema y es precisamente la combinación de teclas <ALT><F4>.

Las opciones completas en el menú principal son: Archivos (submenú), Edición (submenú), Compila (submenú), cOrre (submenú) y ayUda. En tanto que las opciones completas de los submenús son:

Archivos: Nuevo, Abre ..., Salva ..., salva Como ... y saLir

Edición: Cortar, cOpiar, Pegar, Borrar, borrar Todo, Deshacer, bUscar ..., Siguiente y Remplazar ...

Compila: Interno y Externo ...

Corre: Interno y Externo ...

Las opciones que poseen puntos suspensivos (...) indican que al seleccionarlas aparecerá una caja de diálogo, la cual solicitará más información de entrada.

Se podrán utilizar ciertas combinaciones de teclas para ejecutar

procedimientos, parecidos a las opciones del submenú de edición. Enseguida se mostrarán todas las combinaciones posibles de teclas para realizar dichas acciones:

Combinación de Teclas

<Shift><Del>	equivale a la opción de Cortar.
<Ctrl><Ins>	equivale a la opción de cOpiar.
<Shift><Ins>	equivale a la opción de Pegar.
<Del>	equivale a la opción de Borrar.
<Ctrl><Del>	equivale a la opción de borrar Todo.
<Alt><Bksp>	equivale a la opción de Deshacer.
<F3>	equivale a la opción Siguiente ...
<Alt><F4>	equivale a la opción de saLir del submenú de Archivos y de hecho con esta combinación de teclas se pude salir de cualquier nivel, ya sea: del sistema, ventanas y cajas de tipo lista.

Las ventajas que se tienen al utilizar estas combinaciones de teclas, es que nos permiten realizar acciones con rapidez sin tener que pasar por el menú.

Para poder copiar, cortar o pegar texto, es absolutamente necesario poseer un ratón. Usted se preguntará de el por qué de un ratón, bueno, dejeme explicarle que el ratón se utiliza para marcar (resaltar) el texto deseado, y una vez marcado el texto se procederá con las acciones pertinentes, tales como cortar el texto resaltado hacia el portapapeles de Windows o copiar el texto

hacia el mismo portapapeles, ya una vez situado el texto en el portapapeles usted podrá pegarlo en cualquier parte de el documento. El editor del Hyptextw no nos permite crear marcas (tales como: <Ctrl><B> y <Ctrl><K>) por medio del teclado.

Pasos Para Generar un Archivo de Hipertexto

1. Crear un archivo de texto (con marcas de formato).
2. Compilar el archivo texto (se produce archivo.htx).
3. Ejecutar archivo resultante (el archivo.htx sirve de entrada para el Browser).

Archivo Tipo Texto

Para crear un archivo de tipo texto o ASCII, usted deberá utilizar un procesador de texto que le permita generar esta clase de archivos. Una opción es utilizar el editor de el paquete de Hyptextw. El archivo deberá de tener la extensión "txt" y además de tener algunos símbolos de formato, tales como:

"#card nombre-de-tarjeta#" y "#h título-del-texto#"

donde #card le indica al compilador que el texto que continua es el nombre de el núcleo (o tarjeta, pero de aquí en adelante seguiremos utilizando la palabra núcleo). El nombre del núcleo deberá estar entre las marcas de formato #card y #, solamente se utilizará la marca #card sin un nombre cuando un núcleo no tenga alguna etiqueta que lo identifique, veamos algunos ejemplos:

1. #card Planeta Tierra#
2. #card Plano A#
3. #card día caluroso...#
4. #card
5. #card

los núcleos de los ejemplos del uno al tres poseen un nombre, en cambio los dos últimos ejemplos no poseen algún nombre que los identifique.

Después de la parte correspondiente al nombre del núcleo, se podrá incluir (esto a partir de otro renglón) el texto correspondiente al núcleo. El número máximo de líneas que podrá contener un núcleo es de 28; mientras que el número máximo de caracteres será de 80 por línea. Si usted está creando un documento y usted necesita más espacio, usted podría crear un nuevo núcleo pero sin nombre, veamos:

#card Nuestro Mundo#

Nuestro planeta es un lugar maravilloso, en el cual abundan un sin número de especies....

.....

Algunos ríos se están secando otros los está contaminando

(Hasta este punto usted ya no tiene más espacio para más líneas, lo que queda es crear otro núcleo para así seguir escribiendo más texto)

#card

el hombre y esto no parece tener fin. La destrucción de nuestro entorno no se ve como se pueda parar, debido a que el ser humano es el mismo quien lo

destruye y a su vez es el mismo que se está destruyendo con su medio ambiente ...

El comienzo de un núcleo comienza con su nombre y termina cuando la siguiente marca de #card es encontrada, por ejemplo:

#card noches de Abril#

Aquí le corresponde ir al texto de este núcleo.

.....

.....

#card Tlaxcala un estado único#

Sección del texto.

.....

.....

#card

más texto de la sección anterior.

.....

.....

#card

más texto relacionado con Tlaxcala un es...

....

....

.....

#card RIOS#

más texto, pero ya no relacionado con Tlaxcala ....

.....

De esta forma se podrá crear un sistema de información electrónica realmente útil. El único límite que usted podrá encontrar será su propia imaginación.

Otras marcas de formato y de única utilidad para el compilador son: #h y #, donde se podrán incluir encabezados dentro de un núcleo (no confundir los títulos con el nombre del núcleo), veamos algunos ejemplos:

#card México#
#h Contaminación#
La contaminación es .....
.....
#h Historia de México#
La historia nos ha indicado a lo largo de los años que ...
.....
#h Presidentes de México#
Los presi ...
.....

Usted podrá incluir tantos títulos y texto como se lo permita el compilador, recuerde que el número total de líneas es de 28 por núcleo y de 80 caracteres por línea.

Los símbolos de formato sirven para indicarle al compilador cuales son las ligas (hacia un núcleo) y cuales son los títulos.

Hasta el momento sólo hemos visto como crear núcleos (con sus títulos), ahora veamos un simple ejemplo de como crear nuestra información, para que ésta cumpla con la filosofía de hipertexto:

#card
1. México.
2. Estados Unidos de América.
3. Canadá.

#card México#
#h Economía#

La economía de nuestro país se ha ido estabilizando a lo largo de los años anteriores. El futuro parece alentador, debido a la unión económica entre México, Estados Unidos de América y Canadá.
#card Estados Unidos de América#

Estados Unidos de América es un país vecino de nuestro país y de Canadá.
#card Canadá#
#h Deportes#
Canadá posee a grandes equipos de béisbol.

En este ejemplo existen cuatro núcleos, un núcleo sin ningún nombre y tres que si poseen nombre (México, Estados Unidos de América y Canadá).

El primer núcleo posee tres ligas hacia las demás núcleos, mientras que el núcleo de México también posee tres ligas, una así mismo y las demás hacia los núcleos de Estados Unidos de América y Canadá; el núcleo que hace referencia a Estados Unidos de América posee dos ligas, una así mismo y la

otra hacia el núcleo de Canadá; finalmente el núcleo que hace referencia a Canadá, no posee ligas, es decir, es un núcleo terminal.

Compilación

El archivo a compilar deberá tener:

1. Una extensión "txt".
2. Un máximo de 28 líneas por página.
3. Un máximo de 80 caracteres por línea.
4. Símbolos especiales de formato.

Al compilar un archivo de texto con formato, lo que sucede es que el compilador procesa algunas marcas (o Símbolos) para generar las ligas y los títulos correspondientes a cada página (también conocido como núcleo o tarjeta)(para más información acerca de núcleos, ver el tema de hipertexto).

Una vez compilado el archivo ASCII (texto), el compilador habrá producido un archivo que es aceptado por el Browser, este archivo posee una extensión "htx".

Corrida de un Archivo de Hipertexto

Si usted intentara ejecutar un archivo que no existe o que no fue producido por el compilador, el Browser provocará un abandono total del sistema (del paquete). El archivo de hipertexto, producido por el compilador, tendrá una extensión "htx".

Al correr un archivo de tipo hipertexto se produce el efecto de texto debajo de cada liga, es decir, al seleccionar una liga (ya sea con el ratón o con el teclado) aparecerá el texto correspondiente a la liga seleccionada. En un núcleo habrá cualquier cantidad de ligas, y por su puesto usted tendrá la posibilidad de seleccionar cualquiera de ellas; al seleccionar de nueva cuenta a otra liga aparecerá más texto con o sin ligas. Usted podrá seguir paginando a través del texto en la forma que usted quiera, ya sea con el ratón o con la ayuda del teclado. El menú del Browser posee dos opciones que le permitirán ir y venir a través de todo el texto. Otra posibilidad que existe para ir a cualquier parte del texto, es la opción "lista" del menú. La opción lista posee una lista de todos los núcleos (que poseen nombre) y de todos los núcleos que no poseen nombre. Usted puede tener acceso a los núcleos que no tienen nombre por medio de un nombre especial, por ejemplo:

1. page 20
2. page 33
3. page 55
4. México Independiente.
5. Entregas a Clientes:
6. Lista de alumnos.
7. Rosa Hernández.

el primer núcleo sin nombre tiene una etiqueta general page 20, el segundo tiene una etiqueta general page 33 y el tercero una etiqueta page 55, mientras que los demás si tienen un nombre descriptivo.

Ligas o Llaves

Una liga es la dirección a más texto relacionado el cual puede poseer un significado relacionado a la liga, por decir si la liga es la palabra gol, el texto relacionado podría tratar de una definición de gol o hablar de fútbol. Aunque el texto relacionado podría hablar de cualquier cosa, por decir la selva, pero es muy raro hablar de selva cuando la liga nos dice la palabra gol. Lo que quiero explicarle es que usted puede poner cualquier texto debajo de la liga, pero si sería conveniente crear un texto que diga algo con significado para la liga y que a su vez la liga tenga un significado que tenga que ver con el texto que esta liga encierra.

El texto que encierra una liga dada puede contener más ligas y éstas a su vez contener mas texto relacionado.

Para obtener más información acerca de este tópico, por favor ver el tema acerca de hipertexto.

Hipertexto

Un archivo de hipertexto es una opción más que tiene una persona para tener acceso a información almacenada y que puede representar cualquier cosa ya sea un manual o una novela.

El hipertexto en si es texto, así como lo son estas líneas o cualquier

Ligas o Llaves

Una liga es la dirección a más texto relacionado el cual puede poseer un significado relacionado a la liga, por decir si la liga es la palabra gol, el texto relacionado podría tratar de una definición de gol o hablar de fútbol. Aunque el texto relacionado podría hablar de cualquier cosa, por decir la selva, pero es muy raro hablar de selva cuando la liga nos dice la palabra gol. Lo que quiero explicarle es que usted puede poner cualquier texto debajo de la liga, pero si sería conveniente crear un texto que diga algo con significado para la liga y que a su vez la liga tenga un significado que tenga que ver con el texto que esta liga encierra.

El texto que encierra una liga dada puede contener más ligas y éstas a su vez contener mas texto relacionado.

Para obtener más información acerca de este tópico, por favor ver el tema acerca de hipertexto.

Hipertexto

Un archivo de hipertexto es una opción más que tiene una persona para tener acceso a información almacenada y que puede representar cualquier cosa ya sea un manual o una novela.

El hipertexto en si es texto, así como lo son estas líneas o cualquier

otro libro, pero este texto tiene ciertas características que le difieren del texto común y corriente. La principal característica es la que una o más palabras tienen encerrado en ellas más texto.

Cada una de las páginas del hipertexto son llamados núcleos (o tarjetas) y cuando una palabra es capaz de transportarnos a otro núcleo, se dice que la palabra es una liga. El acto de pasar de un núcleo a otro se llama un salto, y la acción de regresar a un núcleo anterior se conoce como un regreso.

Los núcleos que no contienen ligas a otros núcleos son núcleos terminales. El núcleo actual es conocido como núcleo activo.

Cuando se selecciona una liga a otro núcleo, se dice que ésa es la liga activa y cuando no se selecciona se dice que es una liga no activa o simplemente una liga.

BIBLIOGRAFIA

1. Object-Oriented Analysis and Design.
   James Martin y James J. Odell.
   Prentice-Hall, 1992.

2. Object Lifecycles (Modeling the World in States).
   Sally Shlaer y Stephen J. Mellor.
   Prentice-Hall, 1992.

3. Borland C++ 3.0 Programming.
   Ben Ezzell.
   Addison Wesley, 1992.
   Second Edition.

4. Borland C++ 3.1 Programming for Windows.
   Paul Yao.
   Bantam Books, 1992.

5. Aplique Turbo C++.
   HerBert Schildt.
   McGraw-Hill, 1991.

6. Advanced Turbo C (Programmer's Guide).
   Donna Morich, Namir Shamas y Brian Flaming.
   Addison Wesley, 1988.

7. Netzahualcoyótl: Manual de el usuario.
   Por Megatron de México.
   Primera edición.

8. Turbo C++ 3.0 for Windows (programmer's guide).
Borland, 1991.

9. Turbo C++ 3.0 for Windows (user's guide).
Borland, 1991.

10. ObjectWindows for C++ (user's guide).
Borland, 1991.

11. Turbo Debugger 3.0 for Windows (user's guide).
Borland, 1991.

12. Resource WorkShop (user's guide).
Borland, 1991.

13. Multimedia: Solutions Anticipating a Market.
John W. Donovan.
Byte de Diciembre, 1991, pp. 150-151.

14. Information's Human Dimension.
Tom Yager.
Byte de Diciembre, 1991, pp. 153-162.

15. Chips Deliver Multimedia.
Yongmin Kim.
Byte de Diciembre, 1991, pp. 163-176

16. New Knowledge Tools.
Sara Hedberg.
Byte de Julio, 1993, pp. 106-112.

17. See, Hear, Learn.
Sara Hedberg.
Byte de Julio, 1993, pp. 119-128.

18. Roll Your Own Hybrids.
Jay Liebowitz.
Byte de Julio, 1993, pp. 113-118.

19. Apple, SGI Blaze Video Trail.
Tom Thompson y Ben Smith.
Byte de Septiembre, 1993, pp. 81-90.

20. Pen and Voice Unite.
Hewitt D. Crane y Dimitry Rtischev.
Byte de Octubre, 1993, 98-104.

21. Pen Computing Catches on.
Dan Mezic.
Byte de Octubre, 1993, pp. 105-112.

22. Talk to Your Computer.
William S. Meisel.
Byte de Octubre, 1993, pp. 113-120.