Contenido

Editorial

El Centro de Investigación Tecnológica en Computación es ampliamente reconocido en el ámbito académico por el desarrollo tecnológico que en materia de Ingeniería de Cómputo ha realizado, aún antes de su creación como tal.

Esto no significa que el desarrollo de sistemas de cómputo (hardware) sea el único tipo de investigación aplicada que se realiza en el Centro. El software ocupa también un lugar importante en nuestro quehacer, pero no como un fin terminal en si mismo, sino como la herramienta que nos permite trabajar más eficientemente los sistemas de cómputo desarrollados en el CINTEC. Es primordial entender que este tipo de trabajos están enfocados para el manejo de los sistemas de cómputo tal cuales, y que los manejadores de bases de información, sistemas expertos o aplicaciones de este tipo son el campo de trabajo del area de la Informática.

Desde uno de sus primeros proyectos de este tipo, el diseño del BIOS de la computadora IPN-e16, pasando por la implantación de algoritmos de análisis de maquinas rotatorias, programación de Procesadores Digitales de Señales (DSP's), algoritmos de control por lógica difusa, etc. hasta la modificación de un Sistema Operativo y sus directivas de direccionamiento para permitir procesamiento paralelo, el CINTEC a trabajado ampliamente en este campo, aún cuando no se haya presentado la suficiente difusión en este sentido.

El presente número de **polibits** refleja un poco el trabajo que en esta área se realiza en el Centro, dedicandose primordialmente a artículos relativos al tipo de programas que se analizan y desarrollan en el mismo. En esta ocasión se ha dado paso también a artículos escritos totalmente por nuestros alumnos, en los cuales se refleja en buena medida el trabajo académico que han realizado durante su estancia en la Maestría en Ingeniería de Cómputo del CINTEC.

Modelo Didáctico para Evaluar Sistemas Acústicos

M. en C. Miguel A. Partida Tapia
Subdirector Académico y de Investigación del CINTEC-IPN.
Ing. Francisco Flavio Cordova Quiroz
Estudiante de la Maestría CINTEC-IPN.
Ing. Rubén Peredo Valderrama
Profesor e Investigador del CINTEC-IPN.
Ing. Alberto Flores Rueda
Profesor e Investigador del CINTEC-IPN.

Para comprender adecuadamente la importancia de los sistemas acústicos debemos de considerarlos como un medio de comunicación entre transmisor y receptor, si este medio resulta ser de una mala calidad, el mensaje que envíe el transmisor nunca será legible para el receptor, por lo tanto no se tendrá la comunicación deseada entre estos dos. Por tanto, el propósito principal de este trabajo es evaluar tales sistemas y corregirlos para que sean de una buena calidad.

Introducción

Generalmente se reconoce que el rango de frecuencias audible para el ser humano es de 20 Hz a 20 Khz. Considerando este rango y dentro de un sistema acústico, se pueden presentar diferentes fenómenos que hacen más difícil el estudio y análisis del sistema. Para hacer una evaluación eficiente se debe dividir el espectro audible para aislar de la mejor forma posible las frecuencias indeseadas para cada estudio.

Cada una de estas zonas del espectro audible se encuentran plenamente definidas, y se han etiquetado como zona X, zona A, zona B y zona CS. En cada una de estas zonas se presentan distintos fenómenos que se explican a continuación

Zona A, región de baja frecuencia y es dominada por los modos normales de oscilación.

Zona B, región de frecuencias medias y es dominada por los fenómenos de difusión y difracción.

Zona CS, zona dominada por las reflexiones.

Zona X, zona con la característica de no ser afectada prácticamente por ningún fenómeno.

La división del espectro audible se puede hacer en cuatro zonas, como se muestra en la **figura 1**. Estas cuatro zonas pueden variar la posición de sus fronteras, ya que se encuentran en función de las dimensiones del recinto acústico, y conociendo estas dimensiones se puede calcular la frecuencia donde se encontrará dicha frontera, por lo tanto se tiene:

La frontera entre X y A está definida por la ecuación n=172/L, donde L es la longitud del recinto acústico, y se encuentra dada en metros.

La frontera entre A y B está definida por la ecuación n1 = 1904.9 (T / V), donde T es el tiempo que tarda en decaer una señal a -60 dB, y V es el volumen del recinto acústico dado en metros cúbicos.

La frontera entre B y C, esta definida por la relación n2 = 4 n1

Dado que la gráfica está en función de las dimensiones del recinto y la frecuencia a la cual se encuentra la frontera, se ve entonces que mientras mas pequeño sea un recinto las fronteras se encontrarán situadas en (X)Hz y por consiguiente se tendrá un fenómeno más acentuado, en cambio, si el recinto aumenta en sus dimensiones, estas fronteras se moverán a una nueva frecuencia y por consiguiente habrá un nuevo fenó-

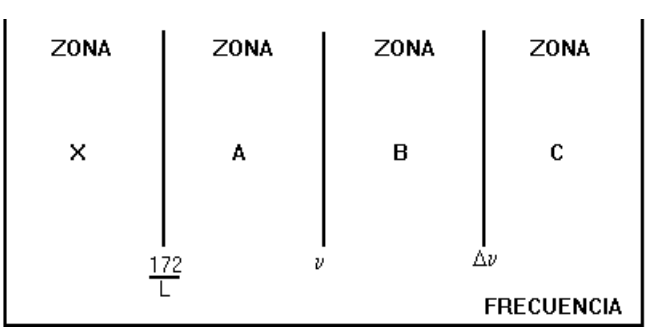


Figura 1. División del espectro audible.

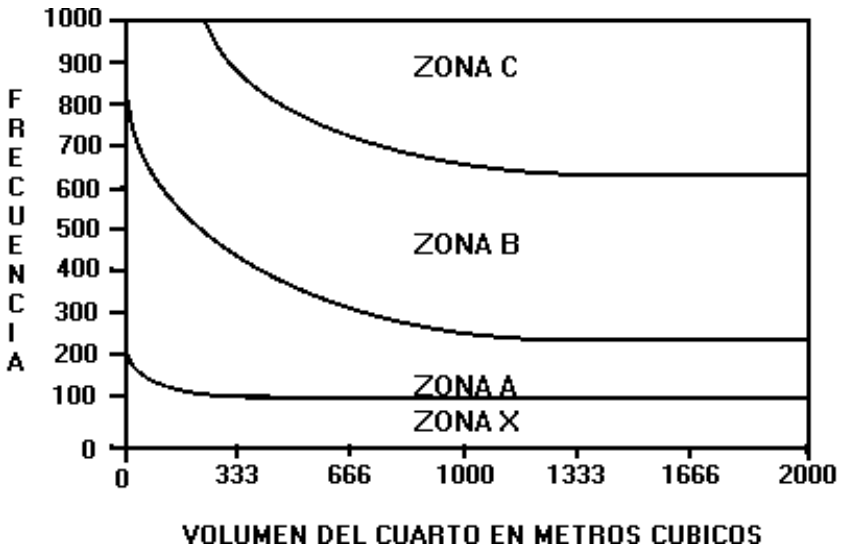


Figura 2. Variación de fronteras del espectro acústico en función de las dimensiones del recinto acústico.

meno predominante, ya que la influencia aumenta respecto a una zona determinada y por consecuencia disminuye a las otras zonas; esto se muestra en la **figura 2**. Como se ha visto, las características de cualquier recinto acústico varían dependiendo de sus dimensiones físicas, por lo tanto existe una cierta cantidad de nodos donde se observan uniones positivas y destructoras de las ondas, y que alterarán las configuraciones de reverberación, eco, aumento de la ganancia relativa de la frecuencia, tiempo de decaimiento y atenuación de la señal, tal como se muestra en la **figura 3**.

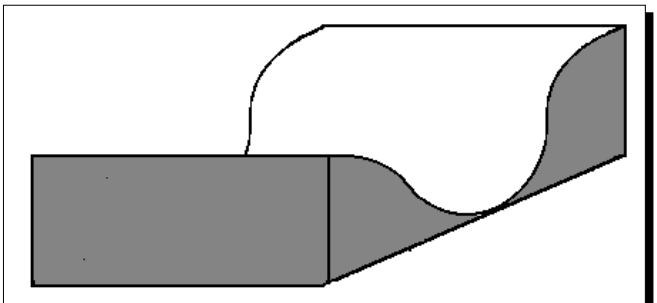


Figura 3. Nodos dentro de un recinto regular.

No está de más tener una aproximación de la cantidad de nodos que existen en un recinto a una frecuencia dada, esto se calcula con la siguiente ecuación.

N = (4 pvl3)(nic)3 + (psl4)(n/c)2 + (E/8)(n/c)

Sin embargo, sería más útil conocer la cantidad de nodos existentes para un ancho de banda, por lo que:

DN = (4 pv(n/c)3 + (pis)(nic)2 + (E18)(nic))(Dnin)

Con esto es posible tener una aproximación de donde ubicar los resonadores y atenuadores necesarios para uniformizar el recinto. Pero lo expuesto hasta el momento no es suficiente para la evaluación de un sistema acústico y consecuentemente su corrección, por consiguiente se tienen que hacer mediciones y caracterizaciones de los sistemas acústicos. Básicamente se pueden identificar 4 sistemas acústicos :

A) Recintos acústicos.

B) Elementos acústicos.

C) Instrumentos musicales y equipos de sonido.

D) Medios de comunicación.

Entonces se definirán todos los requerimientos para cada uno de estos sistemas acústicos, y así obtener un sistema computacional que pueda evaluar a estos sistemas de manera eficiente.

Recintos Acústicos

Se considera un recinto acústico a los siguientes lugares: teatros, cines, salas de conferencias, estudios de grabación, salas de música, estadios, plazas, palenques, etc., los cuales pueden presentar distintos problemas que impiden que se les considere como recintos acústicos de buena calidad. Generalmente son:

1.- Inducciones por la red eléctrica.

2.- Exceso de nivel de ruido del medio ambiente.

3.- Reverberación excesiva.

4.- Modos normales de oscilación destacados.

5.- Defectos de forma (distorción).

¿Cuales son las características de cada una de estos problemas y como se resuelven?

1.- Inducciones por la Red Eléctrica

Esto se identifica por un ligero zumbido constante que es característico de la alimentación eléctrica, este zumbido se detecta a través de mediciones constantes, con diferentes condiciones hasta detectar la fuente y aislar la inducción.

2.- Exceso de Nivel de Ruido del Medio Ambiente

Esto se percibe durante la medición, creciendo más rápidamente que otro ruido y con componentes espectrales de todas las frecuencias audibles (ruido rosa), lo que más deteriora la señal. Para aislar este tipo de ruido, se deben instalar canceles y reforzar las puertas y ventanas logrando con esto el objetivo

3.- Reverberación Excesiva

Se detecta cuando se presenta el fenómeno de prolongación de la energía después de haberse emitido un sonido, que es fácilmente identificable por la prolongación de cualquier fonema por más de 3 ms. Para corregir este defecto es necesario hacer un estudio acústico para encontrar las absorciones necesarias que permitan arreglar el recinto, de acuerdo con las técnicas tradicionales.

4.- Modos Normales de Oscilación Destacados

(MNO). Se descubre cuando el sonido de las palabras producidas por un individuo o sistema, se van convirtiendo a través de los diferentes ciclos de su emisión en sonidos metálicos hasta que finalmente solo se pueden escuchar sonidos de frecuencias definidas. Para la corrección de este defecto acústico se debe de hacer un barrido de frecuencias, para determinar la frecuencia de resonancia, con este dato se puede calcular la frecuencia del resonador absorbente con el cual se resuelve el problema, así como la ganancia relativa de la frecuencia para cuantificar la absorción que debe de tener el resonador, en función de la absorción promedio del recinto.

5.- Defectos de Forma (Distorción)

Este defecto es consecuencia de los (MNO), y provoca la distorción del sonido haciéndolo ininteligible, según sea el grado de distorción. Su solución esta dada de igual manera como se expresa en el punto anterior, con la única variante de que se define el lugar donde se ubican estos resonadores absorbentes.

Elementos Acústicos

Se consideran elementos acústicos a resonadores, filtros silenciadores, baffles, canceles, barreras acústicas, y por lo tanto para estudio se pueden dividir en dos grupos.

1.- Para el caso de los resonadores, filtros, baffles, se tiene por objetivo encontrar las frecuencias características de trabajo, para comprobar su diseño y funcionamiento. Además se obtienen las constantes de amortiguamiento Kn, factor de calidad Q y su ancho de banda Af.

2.- Para el caso de canceles y barreras acústicas, se hace un barrido en frecuencia para encontrar sus debilidades de aislamiento acústico, con el fin de reforzar y de tener el aislamiento deseado, también se obtiene la constante de amortiguamiento Kn, factor de calidad Q y su ancho de banda Af.

Instrumentos Musicales y Equipos de Sonido

Con este tipo de instrumentos y equipo de sonido, se hace la evaluación de la calidad del sonido, para comprobar su diseño o simplemente clasificar comparativamente su calidad entre varios para su selección. Esto puede ser evaluado con acoplamientos completos de micrófono preamplificador, amplificador, bocina aisladamente a amplificadores o parejas de micrófono-bocina. Así se puede hacer un barrido en frecuencia, para localizar las frecuencias de resonancia en cada uno de estos acoplamientos electro-mecáno-acústicos, y revelar si existen configuraciones de las distribución de estas frecuencias, resultado de una afortunada o desafortunada compensación en la respuesta de frecuencia total.

Para el caso especial de los instrumentos musicales, también se buscan las frecuencias de resonancia para observar el comportamiento total de los resonadores, membranas, cuerdas etc. En ambos casos se cuantifica la ganancia relativa de frecuencia, la constante de tiempo a -60dB, frecuencias de resonancia, factor de amortiguamiento Kn, factor de calidad Q y ancho de banda Af.

Medios de Comunicación

En forma parecida a los recintos acústicos, los medios de comunicación pueden ser evaluados en su calidad total con los siguientes puntos:

1.- Inducciones de la red eléctrica.

2.- Exceso de ruido en la transmisión o propagación.

3.- Distorción exagerada.

4.- Frecuencias de resonancia características del equipo.

¿Cuales son las características de cada uno de ellos, y como se resuelven?

1.- Inducciones de la Red Eléctrica

Este se identifica por un ligero zumbido constante, que es característico de la alimentación eléctrica, este zumbido se detecta y se elimina mediante constantes mediciones en diferentes condiciones, hasta detectar la fuente y aislar la inducción.

2.- Exceso de Ruido de Transmisión o Propagación

Este se detecta en la medición, ya que crece más rápidamente que otros defectos en la transmisión, es el que primero deteriora la señal; una vez identificado se procederá a su solución, dependiendo el tipo de equipo que se esté trabajando.

3.- Distorción Exagerada

Esa es posible detectarla en las mediciones, ya que deteriora la señal, provocando la pérdida del timbre de la voz. Para su corrección se debe de mejorar la respuesta en frecuencia del equipo.

4.- Frecuencias de Resonancia Características del Equipo

Se descubren cuando el sonido de las palabras producidas, por un individuo, se van convirtiendo a través de los diferentes ciclos de su emisión en sonidos metálicos, y que finalmente solo se pueden escuchar sonidos de frecuencias definidas.

Para la corrección de este defecto acústico, se debe de hacer un barrido de frecuencias; para determinar la frecuencia de resonancia, se calcula la ganancia relativa de frecuencias, el factor de amortiguamiento Kn, factor de calidad Q y ancho de banda Af, en cada etapa del sistema para hacer la corrección técnica necesaria.

Descripción del Modelo

De acuerdo con lo anteriormente expresado, se puede concluir que el modelo de evaluación acústico debe estar diseñado para evaluar las distintas características de los sistemas acústicos mencionados con anterioridad. Ya que el modelo de evaluación es completamente digital, este debe de ser sumamente amigable, y contener las siguientes funciones de evaluación:

1) Evaluación Acústica.
2) Modos normales de oscilación destacados.
3) Ganancia relativa de frecuencia.
4) Factor de amortiguamiento.
5) Reproducción de defectos.
6) Análisis de Fourier (FFT).
7) Generador de señales.

A continuación, se describen las operaciones que pueden realizarse con la ayuda de este modelo.

1.- Evaluación Acústica

Con esta prueba se evalúa la calidad total del sistema, ya que permite detectar el total de información que se pierde al propagarse en un medio (recinto acústico). Esto es debido a que el defecto tiene la característica de aumentar potencialmente con el número de veces que se repite una palabra o frase continuamente.

2.- Modos Normales de Oscilación Destacados

Bajo esta prueba se pueden obtener los modos normales de Oscilación de cualquier sistema acústico, electrónico o mecánico; así se pueden detectar los defectos en recintos acústicos y sistemas de comunicación; virtudes, como en instrumentos musicales y filtros pasa banda, e inclusive las debilidades de barreras acústicas y canceles.

Para su evaluación se puede ejemplificar la **figura 4**. Una vez que se han encontrado los modos normales de oscilación, este modelo tiene métodos de reproducción total o parcial, que permiten estudiar el comportamiento en detalle, para medir la ganancia, constante de amortiguamiento Kn, factor de calidad Q y

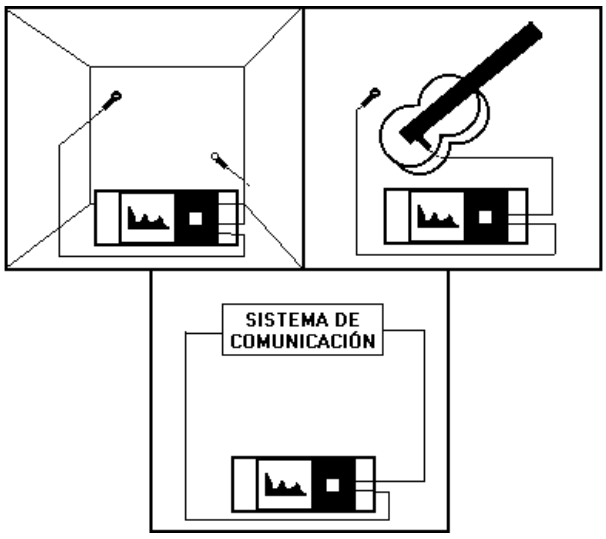


Figura 4. Distintos sistemas acústicos y su evaluación.

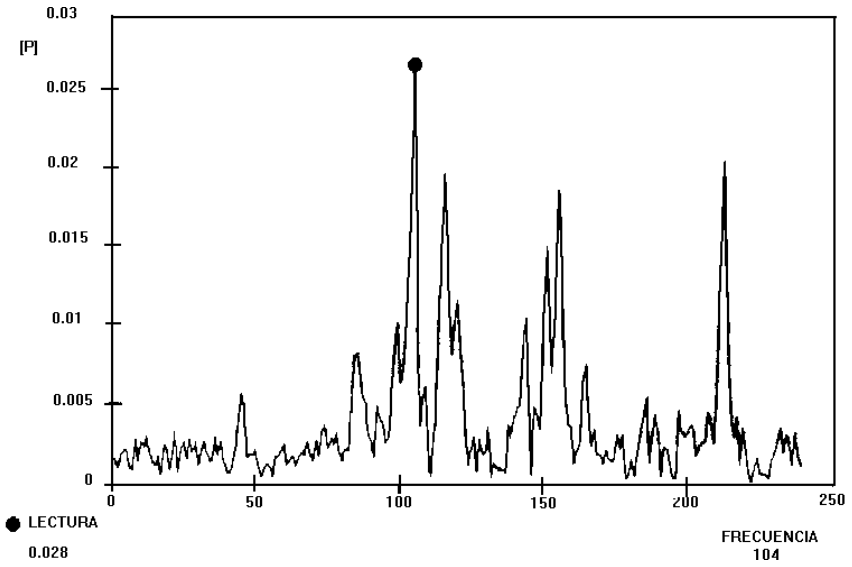


Figura 5. Modos normales de oscilación

ancho de banda presentando los resultados en una gráfica en pantalla o por impresora, como puede verse en la **figura 5**.

3.- Ganancia relativa de Frecuencia

La ganancia relativa de frecuencia de un sistema que tiene una respuesta característica en frecuencia, es una constante adimensional, que expresa la relación en los valores de respuesta de los módulos de una señal senoidal a una frecuencia f, a la cual se desea conocer su promedio unitario.

Cuando el promedio del modulo de esta señal de entrada se mantiene unitario, el valor del modulo de la respuesta de frecuencia fija representa el valor de la ganancia del sistema a esta frecuencia f, en caso contrario esta opción permitirá calcular el resonador y absorbedor adecuados para ajustar el sistema a nuestras necesidades.

4.- Factor de Amortiguamiento

Con esta opción del modelo acústico es posible calcular el tiempo que tarda en amortiguarse una señal cualquiera hasta el nivel de -60dB, y se puede representar a través de distintas gráficas que permitirían visualizar completamente los factores de amortiguamiento de cada una de las frecuencias involucradas en el análisis, así como la constante general del sistema, como se muestra en la **figura 6**.

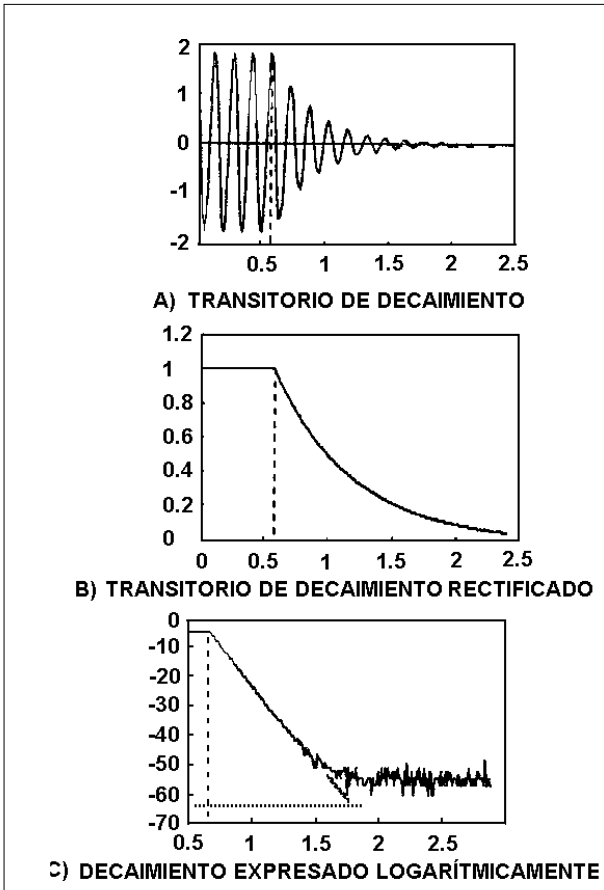


Figura 6. Representación gráfica del modelo acústico.

5.- Reproductor de Defectos

El sistema de evaluación tiene la capacidad de repetir indefinidamente la señal sonora retroexcitada, propia del sistema acústico a través del mismo medio que los produjo, para sostener el régimen característico, y poderlo estudiar cuidadosamente.

El programa que produce este efecto con la señal propia del sistema es fácil de usar, ya que después de que ha producido el ultimo ciclo de trabajo con la opción (2), el aparato queda preparado para producir este efecto especial.

6.- Analizador de Fourier

En caso especial de la operación (Modo de Oscilación) se puede utilizar un analizador de Fourier, alimentando por su entrada una señal cualquiera (voz, ruido, tonos etc.), obteniendo en su salida las componentes espectrales de la señal y observando en una gráfica la distorción producida por el sistema (convolución), para esto se usará el algoritmo de la transformada rápida de Fourier (FFT), dentro del rango audible de 20 a 20000 Hz con la capacidad de selección del número de muestras, que va de 512, 1024, 2048 a 4096 muestras para el análisis.

7.- Generador de Señales

Como se ha visto, en muchas de las distintas evaluaciones se debe hacer un barrido en frecuencia; por lo tanto, para ser congruente con las necesidades, se tiene integrado este generador, el que permite que el sistema tenga un mayor potencial de trabajo. El tipo de señal que puede generar es:

A) Pulso de ruido en banda ancha normalizado.

B.) Muestra de voz normalizada.

C) Senoidales en cualquier frecuencia, con una definición de 1 Hz en la banda de 1 a 2000 Hz, y de 2 Hz en la banda de 2001 a 20000 Hz.

D) Señal cuadrada.

De esta manera se cuenta con un equipo bastante eficaz y flexible para el trabajo de evaluación.

Hasta este punto se han mencionado las funciones que desarrolla el modelo, pero falta detallar cuales son las características del Hardware, cual es su velocidad de trabajo etc.

El modelo trabaja con un microprocesador 80C188EB de INTEL, opera a una velocidad de 16 MHz y dispone de dos canales de muestreo, que pueden trabajar con una frecuencia de muestreo de 0.25 Hz hasta los 20 Khz, sobre un rango de 5 Volts positivos y negativos; dichos canales de muestreo tienen una definición de 12 bits, lo que nos permite hacer divisiones de 1/4096 de señal.

Además se dispone de un banco de memoria de 256 Kb que permite respaldar la información obtenida durante alguna opción, y con una batería que puede mantenerla activa hasta 10 años, imprimir la información en una impresora de matriz de puntos y comunicarse vía puerto de comunicaciones con otras computadoras facilitándo aún más el manejo de información. Por otro lado, el sistema cuenta con una pantalla de cristal liquido (LCD), con una resolución de 620*200 puntos de imagen; para que el usuario observe las gráficas de los resultados en sus distintas modalidades, así como las señales de entrada.

Características del Sistema

El diseño está constituido por una tarjeta madre, un subsistema de alimentación y dos acoplamientos (para entrada y salida, respectivamente), en forma analógica.

En la **figura 7** se muestra el diagrama del 80C188EB, conjuntamente con su lógica de interfaces, así como el subsistema de memoria y control de despliegue de cristal líquido.

El procesador 80C188EB forma parte de la línea de microcontroladores de INTEL (Embedded Microcontroler and Processors), al cual le han sido adicionados elementos, que facilitan al diseñador el manejar subsistemas diversos con una cantidad mínima de componentes. Su operación, con un reloj a 16 MHZ, satisface las necesidades computacionales para esta aplicación, y el hecho de ser compatible en el conjunto de instrucciones con el resto de la línea de microprocesadores 80X86 de INTEL, facilita el diseño de software de aplicación, que se presenta en la **figura 7**.

En resumen el microprocesador integra:

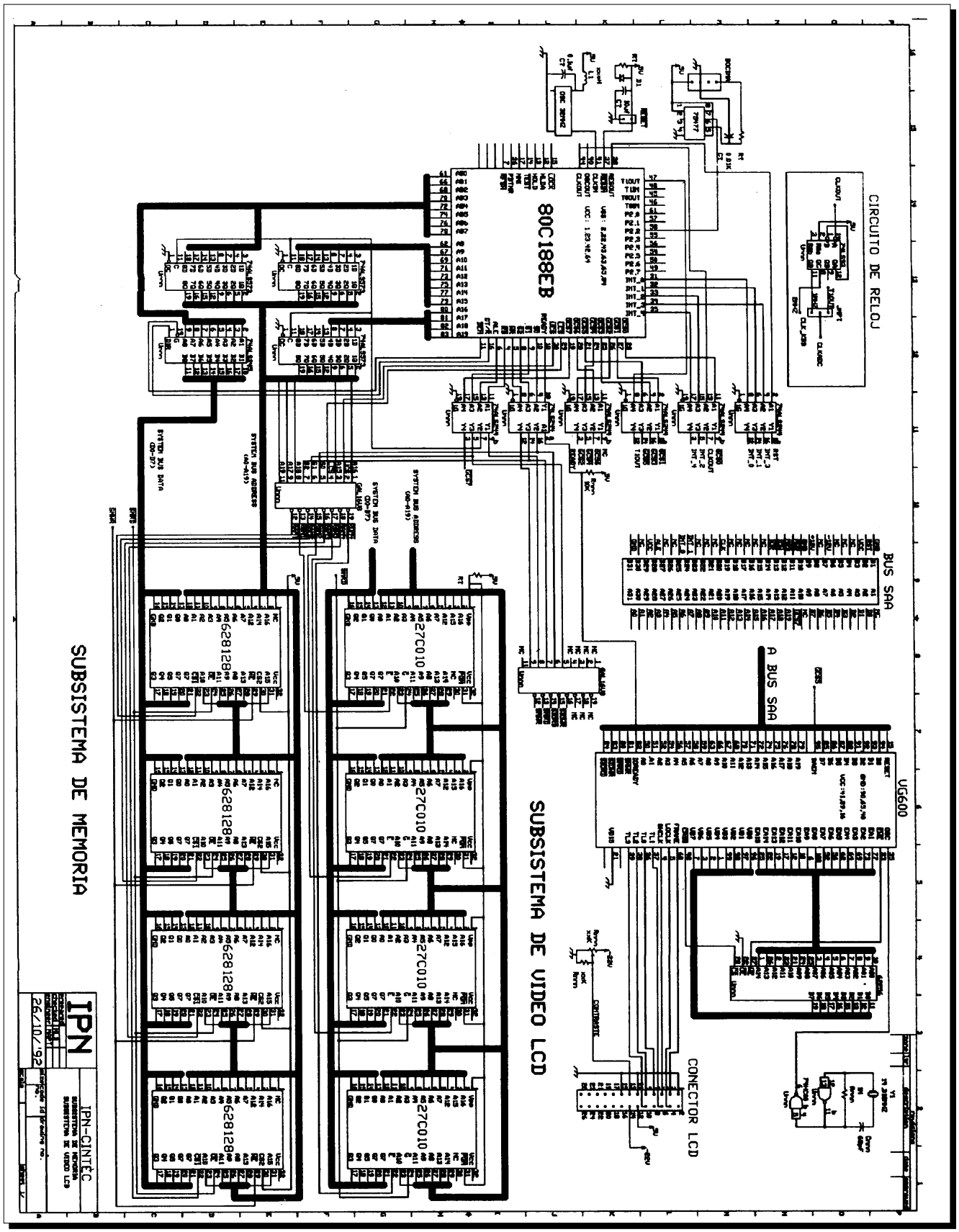
- 7 Líneas de Interrupciones
- 3 Canales de Timer/Count
- 2 Canales de comunicación serie
- 10 Líneas de salida para “Chip Select”
- 1 Unidad de puertos I/O
- 1 Unidad de control de refresco de memoria
- 1 Unidad de control de alimentación.

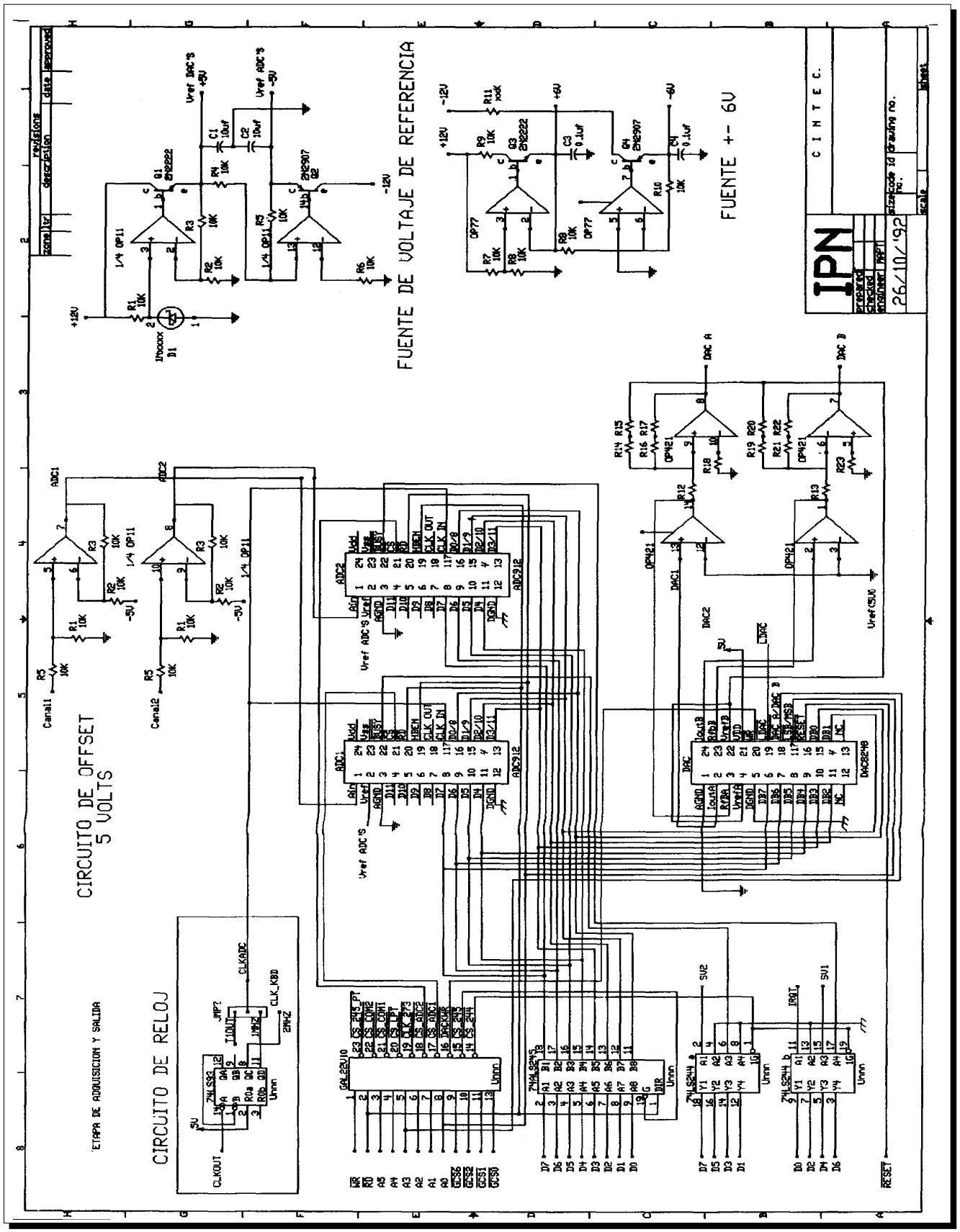
El subsistema de periféricos para esta aplicación se ajustó de tal modo que fuese 100% compatible. Esto permitirá el desarrollo de rutinas, tanto de graficación, como de comunicaciones que se enlacen con periféricos convencionales.

Su potencial permite utilizarlo como sistema de adquisición remoto, usando un ‘MODEM” para transmitir la información por telefonía digital.

En la figura 8 se describe el sistema de adquisición y salida de señales, así como las fuentes de voltajes de referencia para los ADC’s y DAC’s.

La conversión analogico-digital es efectuada por dos dispositivos ADC 912 de Precision Monolitics Inc.. Estos son convertidores de 12 bits, que operan con base en un proceso de aproximación sucesivas.


80C188EB
CIRCUITO DE RELOJ
BUS SAA
A BUS SAA
VG600
CONECTOR LCD
SUBSISTEMA DE VIDEO LCD
SUBSISTEMA DE MEMORIA
IPN
IPN-CINTEC
SUBSISTEMA DE MEMORIA
SUBSISTEMA DE VIDEO LCD
26/10/92
62512812
27C010


FUENTE DE VOLTAJE DE REFERENCIA
FUENTE +- 6V
CIRCUITO DE OFFSET
5 VOLTS
CIRCUITO DE RELOJ
ETAPA DE ADQUISICION Y SALIDA
C I N T E C.
IPN
26/10/'92

La conversión digital-analógica corre por cuenta de un convertidor dual, el dispositivo DAC8248, de Precision Monolitics, Inc. El dispositivo es del tipo multiplicador, puede ser utilizado para la multiplicación en cuatro cuadrantes con el circuito que se indica en la **figura 8**.

El DAC8248 incorpora un doble registro para cada uno de los convertidores, por lo que es posible su escritura secuencial y la habilitación simultánea de nuevos valores.

Cada proceso de conversión requiere de 13 ciclos de reloj; cuya frecuencia es de 1 MHZ como máximo. Esta frecuencia puede ser generada por dos vías, la primera derivada del reloj del sistema (16MHz) y la segunda, usando el TIMER 1 del 80C188EB. Ambas opciones se encuentran provistas en el circuito impreso, seleccionables a través de un "JUMPER«, y la tensión de entrada debe estar comprendida entre 0 y 10 volts (el resultado es proporcionado en forma binaria).

El procesador efectúa la conversión de complemento a dos. Los dispositivos requieren de una referencia de +-5 volts, obtenida a partir del circuito de voltaje de referencia (véase **figura 8**). Desde el punto de vista del procesador, cada convertidor representa un puerto de lectura.

En otro tipo de variantes para su aplicación, se ha experimentado con 64 canales de adquisición multiplexados, donde la adquisición en tiempo real no es tan crítica.

Adicionalmente a lo antes expuesto, se cuenta con un "bus" en formato IBM PC, que permite adicionar periféricos para aumentar la potencialidad del sistema en su conjunto. Para este mismo "bus"", se diseñó una tarjeta que reporta errores al momento de ejecución del sistema, con lo cual se facilita la detección de fallas.

Tal ha sido el caso, como incluir "BREAK POINTS" que indican el avance en la ejecución del software residente. Un ejemplo significativo sería que el microprocesador reporte una verificación exitosa o no, de la RAM, antes de continuar la inicialización del resto del sistema.

Software

El sistema básico de entrada salida contiene, además de les rutinas de control de despliegue y periféricos, las rutinas para calculo de la Transformada Rápida de Fourier, calculo de la Magnitud Cuadrada y la Magnitud del espectro.

En síntesis, el presente producto forma una plataforma para máquinas dedicadas a control y procesamiento digital de señales.

Bibliografía

[1] Lindig, M., Sánchez, E., Poujol, F. y Chavez, M. *"Un sistema de Procesamiento de Señales de Vibración"*. Polibits, CINTEC-I.P.N., Núm. 5, enero-marzo l990, p.p. 3-13.

[2] Lindig, M. y Partida, M., *"Una Tarjeta Controladora basada en el 80188"*, Polibits, CINTEC - I.P.N. Núm. 5, enero-marzo 1990, p.p. 56-69.

[3] Lindig, M., *"Una Microcomputadora para el Análisis de Señales"*, Boletín de Graduados e Investigación I.P.N., Núm. 8, julio-agosto 1982, p.p. 54-82

[4] Lindig, M., *"Diseño de un Filtro Digital Programable para Procesamiento de Señales Biológicas en Tiempo Real"*, Tesis de Maestría, División de Estudios Superiores, Facultad de Ingeniería, UNAM, enero de 1979.

[5] Linear Databook. National Semiconductor Corp., 1988, p.p. 1-41. Analog Databook Precision Monolithics Inc., 1988, p.p. 12-32, 11-47.

[6] Embedded Microcontrollers and Processors, Volume II. Intel, 1992, p.p. 24-75a, 24-819.

Las Posibilidades de la Educación Utilizando Métodos de Multimedia y de Realidad Virtual

M en C Héctor S. García Salas
Profesor e Investigador del CINTEC-IPN.
Ing. Héctor Garcia Rojas
Subdirector de Vinculación y Desarrollo Tecnológico del CINTEC-IPN.

La educación pública en México se encuentra en un punto clave, en el cual es necesario tomar decisiones a nivel nacional para integrar las nuevas tecnologías que en cuestión de educación se encuentran actualmente en uso en otros países y que han demostrado ser herramientas de aprendizaje muy importantes en el proceso educativo.

Problemática

Actualmente el proceso educativo hace uso de técnicas computarizadas para la transmisión de conocimientos. No es necesario demostrar que en nuestra época, el hábito de la lectura se ha perdido en gran parte y en su lugar se ha sustituido por la cultura "televisiva".

Dado el bajo nivel cultural y educativo, (programas que traen consigo una fuerte cantidad de violencia y de comportamientos antisociales), que la televisión presenta, es necesario utilizar otros medios que permitan llegar a las personas de una forma tan eficiente como lo hace el televisor. Esto desde luego implica orientar en otro sentido los modelos educativos tradicionales que se mantienen impermeables a los cambios tecnológicos.

Alternativas

En la actualidad, el uso de computadoras personales "PC" permite establecer nuevos horizontes desde el punto de vista educativo. Para esto, es necesario que las técnicas informáticas sean utilizadas para la generación de herramientas y medios educativos que lleguen a todos los niveles de la educación cuyo empleo sea integrado a los planes y programas oficiales de la misma, favoreciendo el uso doméstico de tales instrumentos.

Esto implica un gran esfuerzo en varias direcciones, ya que es necesario: cambiar mentalidades anquilosadas, capacitar a los Profesores e Instructores al uso de las herramientas informáticas, establecer centros de desarrollo de material de ayuda adecuado, integrar de manera dinámica en los planes de estudio a todos los niveles, el uso de las mismas, etc.

Es necesario hacer notar que el uso popular o doméstico que se está haciendo de la computadora, si bien es un medio visual parecido a la televisión, es por el contrario independiente a la filosofía comercial e ideológica de esta última, ya que se puede tener un control sobre el tipo de paquetes o programas que se adquieren, ésto, desde luego, es responsabilidad moral de las personas encargadas y/o de los padres de familia. La gran diferencia con los medios televisivos es que, en cuestión de computación, se pueden encontrar paquetes y programas de temas muy variados desde los dedicados al entretenimiento hasta educativos de muy alto nivel, como podrá constatarse en cualquier publicación periódica especializada en computación, (aparecen mensualmente alrededor de una decena de paquetes educativos); desafortunadamente la mayoría, por no decir la totalidad, se encuentran escritos en lengua extranjera.

La Multimedia

Hablando propiamente de las herramientas informáticas, las técnicas electrónicas y computacionales han puesto a punto los componentes necesarios con los cuales es posible integrar una técnica conocida como "**multimedia**". Un porcentaje muy grande de los productos **multimedia** están dedicados a los paquetes educativos. En general, en un paquete educativo **multimedia** se combinan al mismo tiempo medios textuales, visuales y auditivos, que permiten estructurar sesiones de enseñaza-aprendizaje interactivas, con el nivel deseado y del tema escogido. Desde luego, la participación de especialistas es requerida para la elaboración de este material.

Muchas preguntas se presentan en este momento; por ejemplo, ¿Estas técnicas dan resultado?, ¿Donde

conseguir este material?, ¿El material que se consigue es el adecuado?, etc. Para responder a estas y muchas otras preguntas sería necesario más tiempo y espacio, sin embargo existe una gran cantidad de literatura que permite ponerse al día y conocer los alcances de estos métodos. No obstante, podemos mencionar que países como: Japón, Francia, Alemania, E.U. y otros, que dedican presupuestos a la preparación, elaboración, producción y utilización de materiales especializados para la educación aplicados desde el nivel elemental, han obtenido como resultado una alta eficiencia en la transmisión de conocimientos en las áreas en las cuales se ha experimentado. Por ejemplo, en el aprendizaje de idiomas, las técnicas con las que se obtienen los mejores resultados son las de **multimedia** en las cuales los textos muestran la estructura gramatical, sintaxis, etc. Los videos y/o animaciones refuerzan la idea de las frases contextuales, creando la relación entre la situación y la utilización del lenguaje. El medio auditivo permite adecuar el oído así como el aprender la pronunciación. El promedio de aprendizaje con el uso de este material, ha permitido una mejora de aproximadamente el 50 %, según la literatura, disminuyendo también el tiempo de aprendizaje y, desde luego, al ser más amigable el ambiente, el esfuerzo de aprender es menor.

El método multimedia es una técnica que puede ser utilizada para la educación a cualquier nivel como herramienta o material didáctico. Actualmente el **multimedia** es utilizado no solamente como instrumento pedagógico sino también como medio de diversión, de propaganda, y otros aspectos.

Para poder aplicar o utilizar técnicas **multimedias**, en cualquier área educativa es necesario contar con el

Figura 1

apoyo adecuado, es decir, una estación **multimedia**; en la **figura 1** se esquematiza una de estas estaciones, compuesta por: una computadora, con velocidad de procesamiento aceptable (por ejemplo, la equivalente a una dotada de un microprocesador de la gama 486), con unidad lectora de discos compactos, tarjeta de video (tipo VGA 256 colores, para un despliegue correcto de imágenes), tarjeta de sonido (tipo SOUND "CUALQUIER MARCA" 16 bits, para obtener un reproducción bastante fiel de los sonidos), y un medio de interacción (tipo "MOUSE" o "JOYSTICK"), además de los otros aditamentos que conlleva una computadora (lector de discos flexibles, monitor color, disco duro de almacenamiento, etc). Con este equipo es posible y tener una sesión de **multimedia**, sin embargo, lo más importante es el paquete de aplicación que deberá ser el adecuado. El uso de un paquete **multimedia** no representa muchos problemas, ya que están diseñados para presentar un ambiente amigable al educando, presentando opciones de trabajo lo más visuales posible y permitiendo la interactividad.

Una estación de trabajo de esta índole permite ser utilizada por dos o tres usuarios al mismo tiempo, por lo cual es aconsejable disponer de un local de **multimedia** equipado con 6 ó 7 máquinas y estableciendo un máximo de 15 usuarios por sesión. El tiempo de cada sesión dependerá evidentemente de la aplicación.

El Laboratorio Multimedia

Establecer un **laboratorio de multimedia** es más costoso y requiere por otro lado de un buen conocimiento de la utilización de las técnicas informáticas y del **multimedia** así como de un buen sistema pedagógico. La elaboración de **material multimedia** pasa por una serie de etapas en las que se consideran los temas a tratar, la estrategia de aprendizaje, el escenario de educación, la integración del material, la experimentación y posteriormente si las estadísticas son buenas, la producción en serie.

El material indispensable para armar un laboratorio de **multimedia** es: una estación multimedia como la arríba citada pero además con una tarjeta de captura de video, una cámara de video, un aparato para grabar sonido en general, un escenario

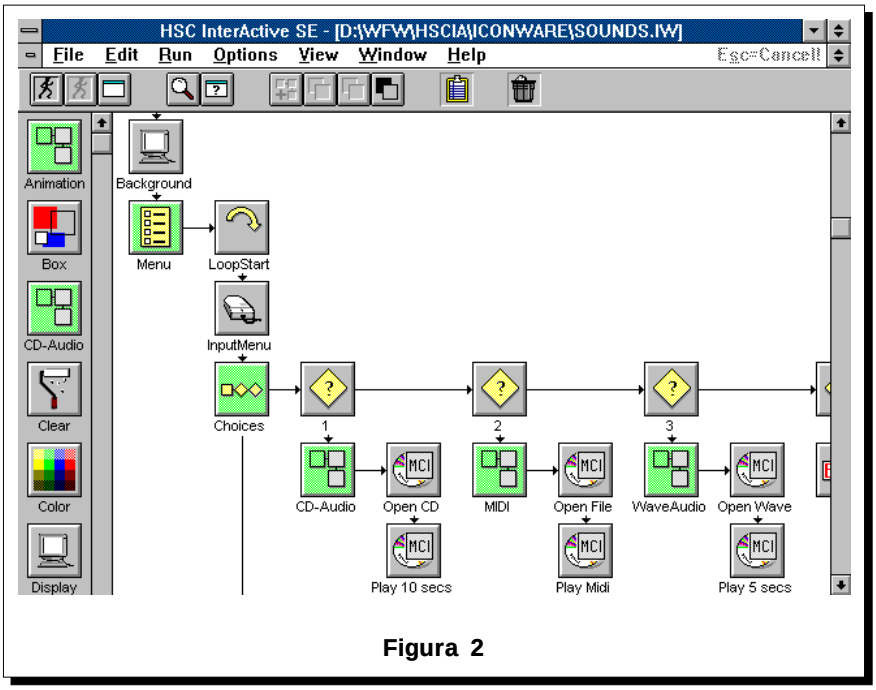


Figura 2

para montar actividades, un sistema de luz y sonido, etc. En cuestión de "software", es necesario un paquete de ensamblaje que permita estructurar paso a paso el desarrollo del tema [1], es decir, dar los tiempos de entrada tanto de video como de sonido o texto, los tiempos de duración de cada objeto, permitir y manejar la interacción (aunque sea limitada) del usuario. La **figura 2** muestra la pantalla de un programa que permite estructurar una presentación multimedia. Una vez instrumentado el material educativo, continuar con la experimentación de tal material elaborando los instrumentos de evaluación adecuados y proceder después a la producción si es que el material resulta exitoso en su función de ayuda educativa. Para la producción de discos compactos que soporten al material elaborado, no es necesario disponer de un dispositivo de lectura y escritura de discos CDROM, ya que existen empresas que se dedican a tal actividad y los costos de producción son aceptables.

Para elaborar material didáctico **multimedia** es necesario tener en cuenta que principalmente los medios naturales por los cuales aprende el ser humano son aprox. en un 90% visuales y auditivos. En base a esto, el tema que se toca deberá presentar una serie de conceptos visuales de tal manera que sean fácilmente asimilables, que se encuentren en un medio ambiente lo más natural posible si se trata de videos, o que presenten un escenario amigable si se trata de animaciones. la asociación de sonidos es muy importante y debe encontrarse sincronizada, estos deben estar relacionados con las imágenes. Los monólogos deberán ser claros con buena pronunciación y buena puntuación utilizando palabras de fácil comprensión.

Por otro lado, siempre deberá presentarse a cada paso en el desarrollo del tema, al menos tres opciones de desarrollo de la sesión que son: "Regresar", "Continuar" y "Salir". De las pantallas que se presenten al usuario, deberán marcarse los puntos clave (en la **figura 3** se muestra como ejemplo, una de las carátulas del paquete wilde animals de MICROSOFT tm) como son: los "Hipertextos", los "Botones de acción", las "Figuras" con animación o sonido específico, etc. También es necesario tomar en cuenta la fatiga visual a la que estará sometido el educando, por lo cual, los temas deberán ser presentados en forma concisa y lo más brevemente posible. Estos son algunos de los aspectos que hacen que un material didáctico o educativo **multimedia** de buenos resultados.

Figura 3

Visto lo anterior, se comprende que no todos los medios educativos puedan tener un laboratorio de **multimedia**, para elaborar su propio material educativo, lo que sería también un desperdicio de recursos. Sin embargo, lo que es fundamental es tener un local de **multimedia** en donde se puedan llevar a cabo las sesiones educativas. Para ésto es necesario que se dediquen los fondos necesarios para dotar a los planteles oficiales de los medios adecuados para este propósito. En lo que concierne a los laboratorios **multimedia**, es necesario la creación de los suficientes centros que permitan abastecer el mercado interno, y en los cuales participen especialistas de los diferentes dominios para obtener material **multimedia** de calidad.

La Realidad Virtual

Otra técnica computacional que en los dos últimos años a tomado mucha fuerza en lo que respecta el tratamiento de información es la "REALIDAD VIRTUAL". En ésta se manejan, producen y sintetizan imágenes que representan ya sea objetos o simples informaciones dentro de un ambiente "virtual", es decir, que existe solamente como concepto (datos numéricos) y que se puede representar gráfica-mente (por medio de imágenes). La **figura 4** muestra la generación de una molécula utilizando el paquete de diseño virtual On3DJunior **[5]**. Esta representación tridimensional permite desplazarse en cualquier dirección y sentido dentro del mundo virtual constituido por el volumen de la molécula.

Las aplicaciones de esta técnica se encuentran en una fase de expansión muy importante, ya que es prácticamente ilimitado el uso que se le puede dar. Los paquetes que utilizan técnicas de **realidad virtual** son muy costosos y son utilizados para la

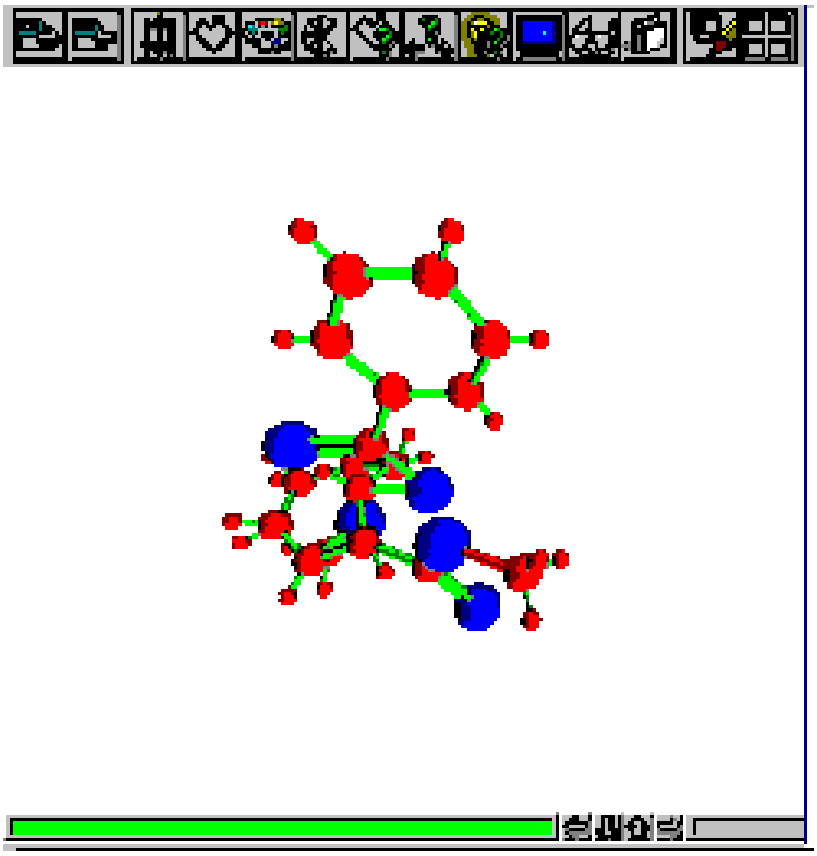


Figura 5

educación en tópicos muy sofisticados como son la ayuda en la formación de pilotos de aviones, en el manejo de artefactos peligrosos, para la instrucción y mantenimiento de reactores nucleares, para la frabicación de modelos moleculares, etc. Sin embargo, ya empiezan a circular paquetes más accecibles y que pueden ser utilizados como material didáctico, aunque esto todavía está en su fase de desarrollo. Un paquete educativo que utilize la **realidad virtal** deberá tomar en cuenta la participación activa del educando, es decir, una total interactividad o casi total, entre el observador y los objetos que se encuentren dentro del escenario gráfico. Los temas a tratar deberán ser de alguna manera integrados al mundo virutal. Como ejemplo, algunos cursos de aprender a conducir utilizan técnicas de realidad virtual para enseñar a los alumnos. A un futuro no muy lejano, se puede prever la utilización masiva de paquetes de **realidad virtual** educativos en diferentes áreas. Inclusive se habla actualmente de aplicaciones como son la de cirugía con **realidad virtual**, la industria con **realidad virtual**, etc.

Esta técnica, que hace uso de muchos y diferentes conceptos matemáticos y del tratamiento de imágenes, se ha aplicado con mucho éxito en lo que se refiere a la creación de escenarios tanto cinematográficos, como de entretenimiento y últimamente en cuestiones científicas, educativas y de capacitación manual.

Ya no es extraño escuchar en cualquier conversación las palabras de "realidad virtual" cuando se refie-

ren a temas en los cuales las soluciones a problemas o simplemente el desarrollo de una trama se llevan a cabo a través de una computadora y en un ambiente gráfico. La **figura 5** muestra una pantalla de un simulador vitual espacial de MicroSoft tm .

Básicamente esta técnica utiliza datos de diferentes tipos, por ejemplo, coordenadas tridimensionales de objetos, características fisicas de fenómenos naturales, objetos sintéticos calculados matemáticamente, valores obtenidos de aparatos de medición, etc. para ser convertidos en imágenes. Estas imágenes son utilizadas para crear secuencias animadas en tiempo real y son desplegadas al usuario, de tal manera que se sienta sumergido y/o integrado en lo que se llama un "**mundo virtual**". La interacción del observador (usuario), con los otros objetos virtuales se lleva a cabo a través de interfases especialmente diseñadas para tal efecto.

Conclusión

En cualquier caso, actualmente se pueden integrar técnicas modernas que utilizadas convenientemente pueden ser herramientas, sino indispensables, al menos con un gran impacto en el proceso de enseñanza-aprendizaje. Y volvemos hacer hincapié en el sentido de destinar un presupuesto oficial, para la elaboración de material didáctico **multimedia** o de **realidad virtual**, asi como para la adquisición del material de soporte adecuado.

Bibliografía

[1] *"HSC InterActive"*, Creative Labs Inc., Santa Mónica, Ca. 1991.

[2] R. Wodaski, *"La Réalité Virtuelle"*, Editorial Sybex, Paris,1994.

[3] J. Gradecki, *"Réalité Virtuelle"*, Editorial Sybex, Paris, 1994.

[4] R. A. Earnshaw, *"Virtual Reality Systems"*, Editorial Academic Press, San Diego, California, 1994.

[5] *"3DJUNIOR"*, Shareware de realidad virtual.

[6] *"3D LAB"*, software para la concepción de objetos en 3D.

Metodología Para el Diseño Orientado a Objetos y Programación Orientada a Objetos

Ing. Rubén Peredo Valderrama
Profesor e Investigador del CINTEC-IPN.
Francisco F. Cordova Quiroz
Alumno de la Maestría del CINTEC-IPN.
Ing. Alberto Flores Rueda
Profesor e Investigador del CINTEC-IPN.

Este artículo describe una metodología para el diseño y programación orientada a objetos, la cuál es muy apropiada cuando se desarrollan proyectos de gran tamaño y aquellos que involucran equipos de programadores, además de proporcionar un lenguaje gráfico al código escrito, para su mejor comprensión. El siguiente trabajo trata el diseño orientado a objetos, clase, organización, codificación, y así sucesivamente. Siguiendo las sugerencias del presente artículo se asegura que el código C++ será consistente y profesional en su organización y apariencia.

Diseño de Representaciones Orientadas a Objetos

Cualquier método que se seleccione para diseñar definiciones de clase debe cubrir los siguientes aspectos:

o Comunicar el diseño externo de la clase (interface)

o Comunicar el diseño interno de la clase (datos en particular)

o Establecer la relación de herencia entre las clases (la clase A es hija de B)

o Establecer la relación entre clases (la clase A es amiga de B)

Los cuatro puntos arriba mencionados se cubrirán a continuación, y son ampliamente recomendados para el diseño en C++, aunque toda esta metodología es aplicable a otros lenguajes de programación como Pascal, Ensamblador, entre otros.

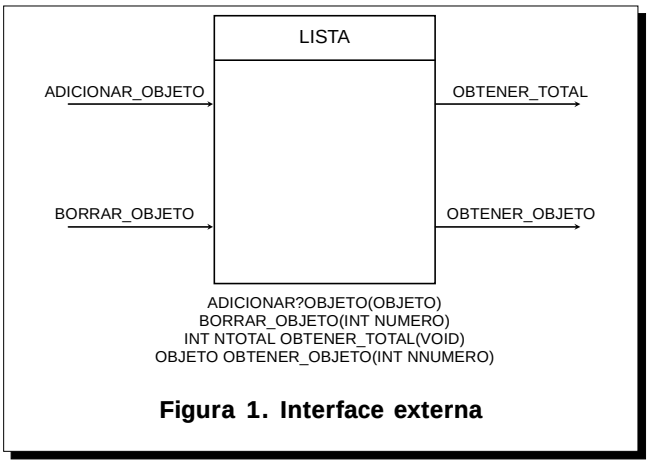


Figura 1. Interface externa

Diseño Externo

El primer paso en el diseño de una clase es determinar la interface hacia el exterior, y funciones miembros para su diseño. Como se muestra en la **figura 1**, la clase misma es representada como un rectángulo, en lo alto del rectángulo se pone el nombre de la clase. Si es necesario especificar él nombre de un objeto, este se coloca directamente bajo el nombre de la clase; esto es muy útil para el manejo de objetos en las aplicaciones, debido a que es posible determinar de manera rápida que tipo de objeto es, además de aumentar la legibilidad del código.

En el lado izquierdo se colocan flechas que entran al rectángulo y sus correspondientes etiquetas. Las etiquetas son los nombres de funciones miembros que ponen datos hacia el objeto o efectúan una acción sobre el objeto, e incluso en algunas ocasiones ambas tareas.

Debajo del rectángulo se ponen las definiciones de cada función miembro. Estas definiciones siguen los prototipos estandard con sus respectivos parámetros.

Diseño Interno

El diseño interno incluye datos incluidos por composición y funciones internas que no son accesibles por funciones externas. El diseño interno de la clase utiliza un formato similar al de la **figura 2**, la cuál muestra datos internos dentro del

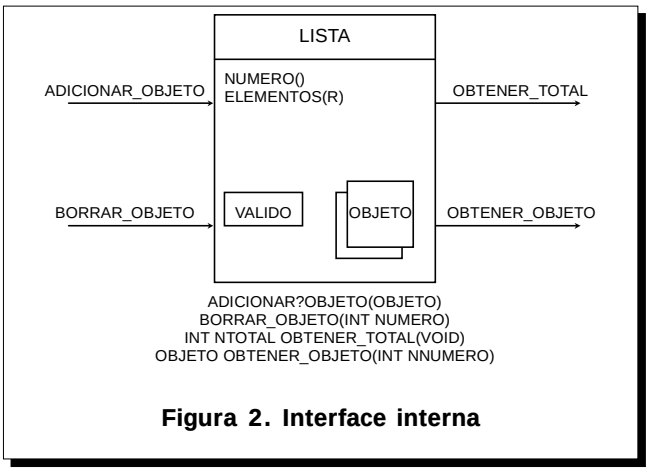


Figura 2. Interface interna

rectángulo. Los datos son listados por tipo y nombre en la parte alta del rectángulo. Es importante mostrar en seguida del dato un paréntesis, el cuál puede contener una L o una E, o en algunos casos nada. La L significa lectura y la E escritura. Dado que L y E aplican únicamente a datos, esta notación es encontrada únicamente dentro del rectángulo. Específicamente:

(L/E) las variables pueden ser leídas y escritas fuera de la clase. Las operaciones de lectura y escritura pueden ser efectuadas por funciones miembros (una que lea y otra que realice la escritura) para mantener la encapsulación interna de la clase. Sin embargo, es importante señalar que las funciones miembro que manipulan las variables no necesitan ser colocadas fuera del rectángulo con las otras funciones miembros. Las funciones miembro de lectura y escritura son implicadas por las letras L / E a continuación del nombre de la variable.

(L) las variables pueden ser leídas pero no escritas. Estas variables deben de ser inicializadas y mantenidas por la clase misma.

(E) las variables pueden ser escritas pero no leídas. Estas pueden ser banderas, o objetos tales como passwords que pueden ser establecidos pero no leídos.

Las variables sin L o E no pueden ser leídas o escritas fuera de la clase. Estas son variables son utilizadas internamente por la clase misma.

Otros objetos dentro de la clase son incluidos dentro del rectángulo debajo de los datos. Para cada uno de estos objetos, el nombre de la clase es mostrado en lo alto y un nombre descriptivo asignado a su instancia especifica de la clase (objeto), dentro del rectángulo. El nombre del objeto es frecuentemente escrito en itálicas. Si existen múltiples instancias de un objeto, tal como un arreglo de objetos, se deberá mostrar esto dibujando pilas de objetos como se muestran en la **figura 2**.

Las funciones internas que no son accesibles fuera de la clase deben mostrarse dentro del cuerpo del rectángulo del objeto. Los nombres de las funciones internas son incluidas en el cuerpo del rectángulo. Los parámetros específicos usados por funciones internas no son mostrados por lo regular en este nivel.

Relaciones de Herencia

Las relaciones de herencia entre los objetos, o alguna clase de jerarquía, son mostradas utilizando una organización jerárquica similar a la mostrada en la **figura 3**. Por ejemplo en dicha figura se muestra la relación entre la clase base (lista) y las clases derivadas (las demás) de esta.

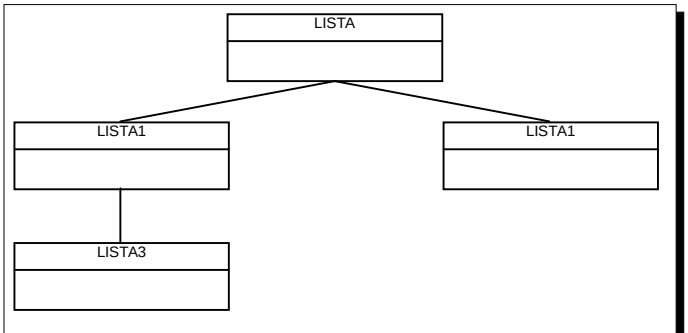


Figura 3. Relación de herencia entre clases.

Relaciones de Composición

Las relaciones de composición, forman parte de una jerarquía que se representa de manera similar a la **figura 4** (en forma de árbol). En este ejemplo, el titulo de la clase es LIBRO y esta compuesta por tres capítulos y dos secciones (dentro del capitulo 3).

Para mostrar más detalladamente la clase y sus funciones miembro, se realiza una subdivisión de clase y funciones miembro, es importante notar que no existen estandares para la representación de diseños en programación orientada a objetos, pero esta notación trabaja bastante bien para fines prácticos.

Organización de Filas Para el Desarrollo en C++

La **figura 5** muestra un ejemplo de organización para programas simples en C++ para Windows. Se contará con un conjunto de filas (encabezado y fuente) para cada clase que se

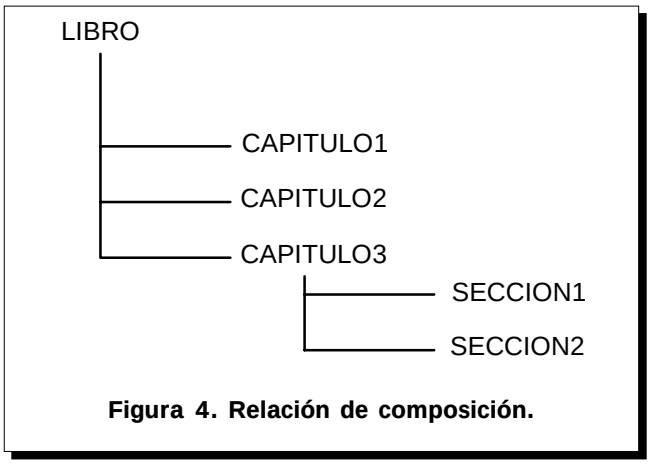


Figura 4. Relación de composición.

define. Es práctico y muy común entre los programadores poner en la fila de encabezado la definición de la clase, y definición de tipos que pertenecen a la clase; así como cualquier función relacionada para funciones no miembros. Para la clase, las definiciónes de función sirven como prototipos de las mismas. Para trabajar en Windows, es necesario incluir funciones miembro para exportarlas.

Es importante notar que la mayoría de los programadores no incluyen código, y aún código inline. La mayor parte de la literatura especializada en C++ recomienda de manera importante no poner código en las filas de encabezado para facilitar el mantenimiento y actualización del programa. El código es colocado en las filas fuente *.CPP. El único punto en contra de este manejo de definiciones y código, es que al definir la clase y colocar el código a continuación se obtiene una mayor eficiencia del mismo, dado que el código realmente se encuentra en inline, pero implica una mayor cantidad de trabajo al actualizar el programa. Sin embargo, es altamente recomendable el caso anterior cuando la eficiencia del código es el factor de mayor consideración a tomar en cuenta. Otra alternativa es colocar el código en la fila .CPP y utilizar la función inline para llegar a un resultado más o menos equivalente.

El código fuente es puesto en las filas *.CPP; por esto debe haber una correlación uno a uno entre las filas de código y las filas de encabezado. Dentro de la fila de código fuente, se definirá cada una de las funciones puestas en la fila de encabezado perteneciente a la clase.

A diferencia de los programas para DOS, los programas para Windows cuentan en la mayoría de los casos con filas de recursos .RC, las cuáles también cuentan con una fila de encabezado *.RH; en las filas RC se cuenta con los recursos propiamente dichos (esto es, equivalente al código en una fila *.CPP), y en las filas RH se encuentran con los identificadores de los recursos (similar a las filas de encabezado *.H).

Estructura de las Filas de Encabezado

Una fila de encabezado consiste principalmente de los siguientes elementos:

Un comentario: esta parte provee un nombre a la fila y una pequeña descripción de la misma.

Directivas para compilación: Esto previene al compilador de compilar dos veces la misma fila de encabezado, si es incluido más de una vez, dado que las filas de encabezado son incluidas múltiples veces en aplicaciones reales.

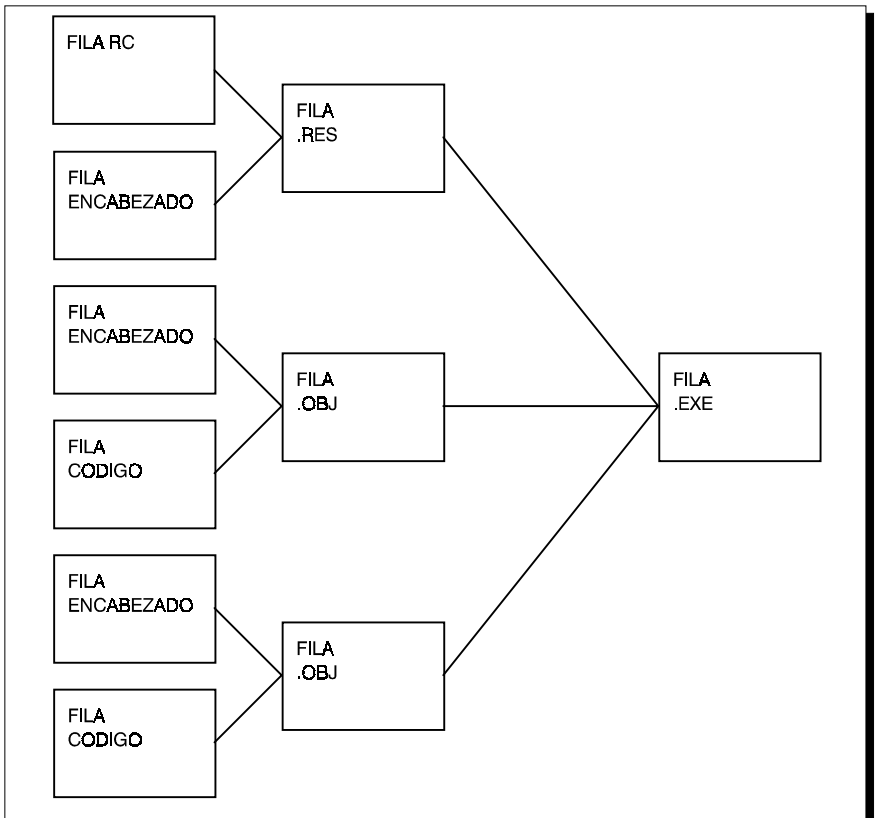


Figura 5. Organización de un proyecto en C++.

Definición de prototipos de funciones: esta parte incluye el establecimiento de funciones, para su posterior definición en las filas .CPP.

Directivas al compilador: Esta parte consiste de llamadas al compilador con palabras reservadas como **typedef** y inclusión de bibliotecas **include<>**.

Declaración y descripción de una clase: Esta parte incluye la definición de la clase y definición de los niveles de acceso de las funciones (**publicas, protegidas y privadas**), y declaración de datos miembros, además de establecer las relaciones entre otras clases (**friend**).

La directiva #endif: En caso de incluirla esta directiva al compilador establece una condicional, a la manera de **if** en C, pero al compilador.

Esto puede observarse en las siguientes secciones de código.

```
Parte 2
_____________________________________________________________________

#if !defined(__redesapp_h)
#define __redesapp_h
_____________________________________________________________________
Fin Parte 2

Parte 1
_____________________________________________________________________

/*  Proyecto Redes Neuronales
   INSTITUTO POLITECNICO NACIONAL


   SUBSISTEMA:   redes.exe Aplicación
   FILA:        redesapp.h
   AUTOR:       CINTEC-IPN

   DESCRIPCION
   ===========
   Definición de la clase para redesApp (TApplication).
*/
_____________________________________________________________________
Fin Parte 1

Parte 4
_____________________________________________________________________
#include <owl\owlpch.h>
#pragma hdrstop
#include <classlib\bags.h>
#include «rdsmdicl.h»
#include «redesapp.rh»   // Identificadores de los recursos.
_____________________________________________________________________
Fin Parte 4

Parte 5
_____________________________________________________________________

/* TFileDrop class Mantiene la información de la fila, su nombre,  donde fué
arrastrada, y si se encuentra en el área del cliente */

class TFileDrop {
public:
  operator == (const TFileDrop& other) const {return this == &other;}

   char*   FileName;
   TPoint  Point;
   bool    InClientArea;

Parte 3
_____________________________________________________________________

   TFileDrop (char*, TPoint&, bool, TModule*);
   ~TFileDrop ();
_____________________________________________________________________
Fin Parte 3

private:

Parte 3
_____________________________________________________________________

   TFileDrop (const TFileDrop&);
   TFileDrop & operator = (const TFileDrop&);
_____________________________________________________________________
Fin Parte 3
};
_____________________________________________________________________
Fin de la parte 5

Parte 4
_____________________________________________________________________
```

```
typedef TIBagAsVector<TFileDrop> TFileList;
typedef TIBagAsVectorIterator<TFileDrop> TFileListIter;
_____________________________________________________________________
Fin Parte 4

Parte 5
_____________________________________________________________________
class redesApp : public TApplication {
private:

   bool          HelpState;
   bool         ContextHelp;
   HCURSOR        HelpCursor;

Parte 3
_____________________________________________________________________

   void SetupSpeedBar (TDecoratedMDIFrame *frame);
    void AddFiles (TFileList* files);
_____________________________________________________________________
Fin Parte 3

public:

Parte 3
_____________________________________________________________________

   redesApp ();
   virtual ~redesApp ();
   void CreateGadgets (TControlBar *cb, bool server = false);
   redesMDIClient  *mdiClient;
_____________________________________________________________________
Fin Parte 3
   TPrinter       *Printer;
   int            Printing;
public:

Parte 3
_____________________________________________________________________

   virtual void InitMainWindow ();
   virtual void InitInstance ();
   virtual bool CanClose ();
   virtual bool ProcessAppMsg (MSG& msg);
_____________________________________________________________________
Fin Parte 3

protected:

Parte 3
_____________________________________________________________________

   void EvNewView (TView& view);
   void EvCloseView (TView& view);
   void CmHelpAbout ();
   void CmHelpContents ();
   void CmHelpUsing ();
   void EvDropFiles (TDropInfo drop);
   void EvWinIniChange (char far* section);
   void CmBAM1 ();
_____________________________________________________________________
Fin Parte 3

DECLARE_RESPONSE_TABLE(redesApp);
};
_____________________________________________________________________
Fin Parte 5

Parte 6
_____________________________________________________________________

#endif
_____________________________________________________________________
Fin Parte 6
```

Estructura de las Filas de Código Fuente

Una fila fuente consiste de los siguientes elementos:

Comentarios: sirven para indicar el nombre de la fila y una breve descripción de la misma.

```
Parte 1
_____________________________________________________________________

/*  Proyecto  Redes Neuronales
   INSTITUTO POLITECNICO NACIONAL


   SUBSISTEMA:    redes.exe Application
   FILA:         redesapp.cpp
   AUTOR:       CINTEC-IPN



   DESCRIPCION
   =============
   Fila fuente para la implementación de redesApp (TApplication).
*/
_____________________________________________________________________
Fin Parte 1

Parte 2
_____________________________________________________________________

#include <owl\owlpch.h>
#pragma hdrstop
#include <dir.h>
#include «redesapp.h»
#include «rdsmdicl.h»
#include «rdsmdich.h»
#include «rdsedtvw.h»
#include «rdsabtdl.h»                    // Definición acerca del dialogo.
_____________________________________________________________________
Fin Parte 2

Parte 3
_____________________________________________________________________

const char HelpFileName[] = «redes.hlp»;

TFileDrop::TFileDrop (char* fileName, TPoint& p, bool inClient, TModule*)
{
Código Fuente
}

TFileDrop::~TFileDrop ()
{
Código Fuente
}

const char *TFileDrop::WhoAmI ()
{
Código Fuente
}

// Etc....etc..
_____________________________________________________________________
Fin Parte 3
```

Incluir bibliotecas: consiste en incluir las bibliotecas necesarias para el programa.

Descripción de las funciones miembro: Declarar las funciones miembros incluidas en la fila de encabezado.

Bibliografía

[1] García-Pelayo y Gross, Ramón; *"Diccionario Pequeño Larousse en Color"*, 1981.

[2] *"Standard Dictionary of the English Language"*; Funk & Wagnalls, 1967.

[3] Wienderhold, Gio; *"File Organization for Data Base Design"*; McGraw-Hill, 1967.

[4] Shaft, Adam; *"Historia y Verdad. Teoría y Praxis"*.

[5] William Roetzheim; *"Programming Windows with Borland C++4.5"*; ZD PRESS 1994.

[6] Namir Shammas, Craig Arnush & Edward Mulroy; *"Teach Yourself Borland C++ 4.5 in 21 Days"*; SAMS Publishing 1995.

[7] Paul Perry; *"Using Borland C++ 4"*; QUE 1994.

Programación Orientada a Objetos: Simulación de una Red de Propagación Inversa

Rubén Peredo Valderrama
Profesor e Investigador del CINTEC-IPN.
Francisco F. Cordova Quiroz
Alumno de la Maestría del CINTEC-IPN.
Eduardo Rodríguez Escobar
Jefe del Departamento de Sistemas Digitales del CINTEC-IPN.

El presente artículo tiene la finalidad de mostrar el diseño de un simulador del modelo básico propuesto por Werbos [1974], de una red neuronal de propagación inversa (back-propagation) en lenguaje C++ orientado a objetos de Borland, versión 4.5 para Microsoft Windows.

Introducción

La red neuronal de propagación inversa es un modelo muy popular, dado que no tiene conexiones de retroalimentación y los errores son propagados inversamente durante el entrenamiento. Algunas aplicaciones pueden ser resueltas usando redes de propagación inversa y, dada su topología, es muy apropiada para multiestratos ("multilayer"). Los errores de la salida determinan la medida de los estratos ocultos, los cuales son usados como base para el ajuste de las conexiones de pesos entre la entrada y los estratos ocultos. El ajustar dos pesos entre parejas de estratos y recalcular la salida es un proceso interactivo que es acarreado hasta que el error cae por debajo del nivel de tolerancia. Los parámetros de la proporción de aprendizaje escalan el ajuste de los pesos. Un parámetro llamado momentum también puede ser usado en la escala de ajuste desde una interacción previa y adicionando el ajuste en la interacción presente.

Operación de la Red de Propagación Inversa

En este punto se explicara el funcionamiento de una red de propagación inversa. En principio la red aprende de una entrada-salida definida utilizando un ciclo de **propagación-adaptación**. Después de que el patrón ha sido aplicado como un estimulo al primer estrato de la red, este es propagado a través de cada estrato superior hasta que la salida es generada. La salida es entonces comparada con el valor deseado, y la señal de error es calculada para cada salida.

La señal de error se transmite desde el estrato de salida a cada nodo en el estrato intermedio, recibiendo únicamente una porción del error total y basado únicamente en la contribución relativa hecha a la salida original. Este proceso se repite, estrato por estrato, hasta que cada nodo en la red ha recibido una señal de error que describe las contribuciones relativas del error total. Basado en la señal de error que recibe, los pesos de las conexiones son entonces actualizados para cada unidad, ocasionando que la red converja hacia el estado que permite a todos los patrones de entrenamiento ser decodificados.

El significado de este proceso es que, durante el entrenamiento, los nodos en los estratos intermedios se organizan por si mismos tal como los diferentes nodos aprenden a reconocer diferentes características del espacio de entrada. Después del entrenamiento, cuando se presenta un patrón de entrada arbitrario (ruido o incompleto), las unidades en los estratos ocultos de la red responderán con una salida activa si la nueva entrada contiene un patrón semejante a las características de las unidades individuales aprendidas para reconocer durante el entrenamiento. Contrariamente, las unidades del estrato oculto tienen la tendencia a inhibir sus salidas si no tienen el patrón con él cual fueron entrenadas para reconocer. Como la señal se propaga a través de los diferentes estratos en la red, la actividad del patrón presente en cada estrato superior puede ser pensado como un patrón con características que pueden ser reconocidas por unidades en subsecuentes estratos. El patrón de salida generado puede verse como un mapa característico que proporciona la indicación de la presencia o ausencia de diversas combinaciones de características en la entrada. El efecto total de su funcionamiento es que la red de propagación inversa proporciona un

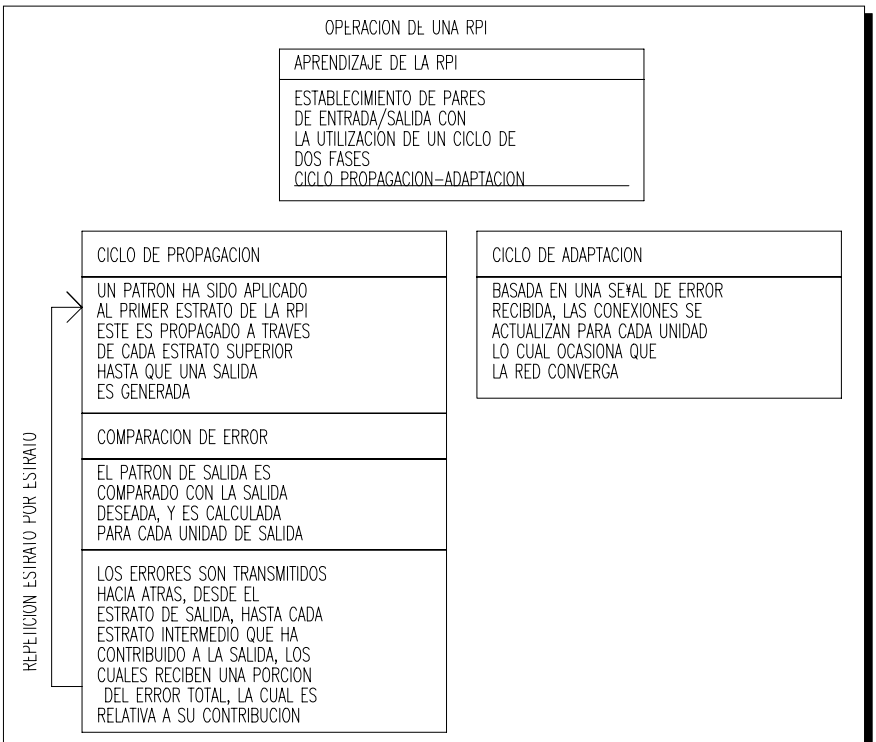


Figura 1. Esta figura muestra el funcionamiento de una red del tipo RPI.

efectivo significado al permitir a un sistema computarizado examinar patrones de datos que pueden ser incompletos o que pueden contener ruido, y reconocer patrones parciales de entrada. Esto puede apreciarse en la **figura 1**.

En muchas ocaciones se ha encontrado que durante el entrenamiento (**figura 3**), las redes de propagación inversa tienden a desarrollar relaciones internas entre nodos, así como a organizar el entrenamiento de datos dentro de ciertas clases de patrones. Esta tendencia puede ser extrapolada a la siguiente hipótesis: todas las unidades de los estratos ocultos en una red de propagación inversa están asociados con características especificas de los patrones de entrada como resultado de su entrenamiento. Esta asociación puede ser o no evidente a los observadores humanos. Lo importante es que la red ha encontrado una representación interna que posibilita las salidas deseadas cuando se dan las entradas de entrenamiento. Esta representación interna puede ser aplicada a las entradas que no fueron usadas durante el entrenamiento. La red de propagación inversa clasificará estas entradas de acuerdo a las características que comparten con las entradas de entrenamiento.

Construcción de una Red Neuronal de Propagación Inversa

La arquitectura de una red de propagación inversa se muestra en la **figura 2**. Pueden exisitir más estratos ocultos, pero en esta figura solo se representará uno. De igual forma, el número de neuronas en el estrato de entrada y en el estrato de salida están determinados por las dimensiones de los patrones de entrada y salida, respectivamente. Como en el

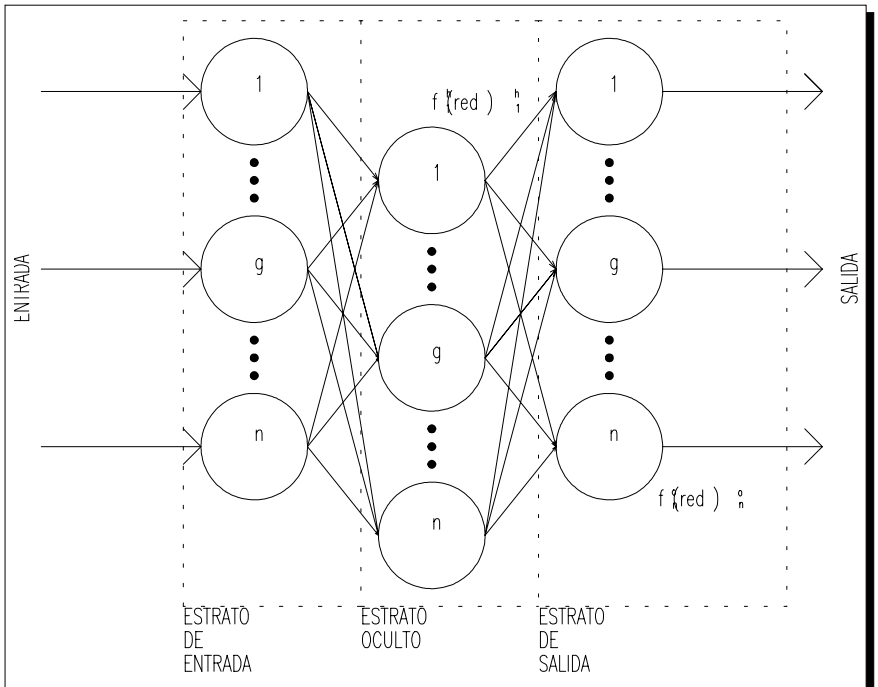


Figura 2. Esta figura muestra una arquitectura típica de una red del tipo RPI, y sus correspondientes estratos

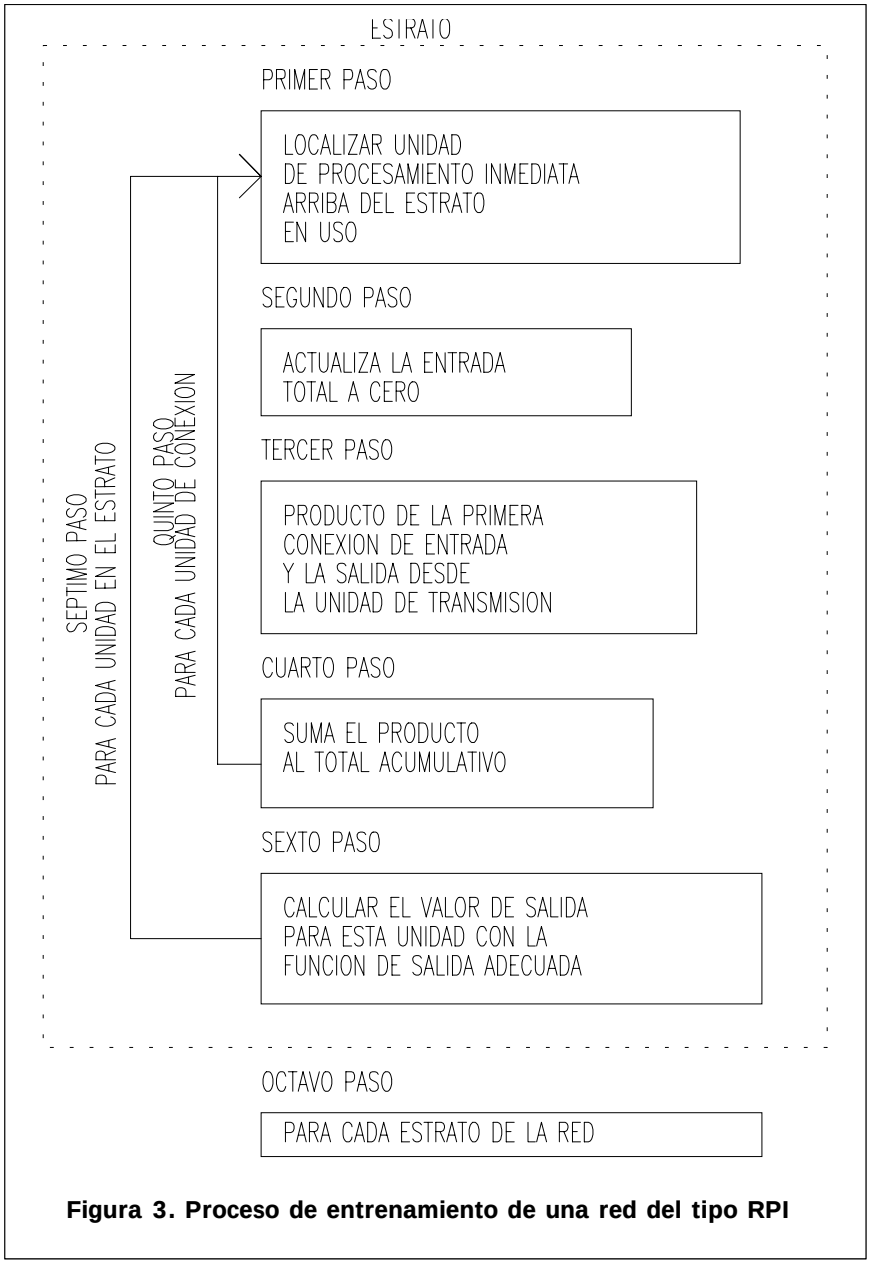


Figura 3. Proceso de entrenamiento de una red del tipo RPI

número de estratos ocultos, tampoco es fácil determinar cuantas neuronas son necesarias. Para evitar realizar una figura muy compleja se dibujará con cinco neuronas de entrada, tres neuronas en el estrato oculto, y cuatro neuronas de salida con algunas representaciones de conexiones.

Tamaño de la Red

¿Cuantos nodos son necesarios para resolver un problema particular? La respuesta involucra de manera considerable a los datos de entrenamiento, dado que no existe una respuesta estricta para resolver la pregunta. Por lo general 3 estratos son suficientes. Sin embargo algunas veces un problema podría ser más fácil de resolver con un solo estrato oculto. En este caso se entiende que más fácil significa que la red aprende más rápido.

El tamaño del estrato de entrada es usualmente dictado por la naturaleza de la aplicación. Frecuentemente se puede determinar el número de nodos de salida sabiendo si se buscan valores analógicos o valores binarios en las unidades de salida.

La determinación del número de unidades a usar en el estrato oculto comúnmente no es fácil como lo es en el caso de los estratos de entrada y salida. La idea principal es usar tan pocos estratos ocultos como sea posible, porque cada unidad adiciona una carga extra durante la simulación en el CPU de la computadora. De seguro, un sistema que es totalmente implementado en hardware (un procesador por elemento de procesamiento), adiciona cargas adicionales de trabajo al CPU.

Para una medida razonable, el tamaño del estrato oculto necesita ser relativamente una fracción más pequeña del estrato de entrada. Si la red falla al converger, se podría intentar adicionar más nodos ocultos para solucionar este problema. Si la red converge, se podría intentar reducir el número de nodos ocultos y asentar un tamaño básico para la ejecución de todo el sistema.

También es posible remover unidades ocultas que son superficiales. Si al examinar los valores de los pesos en los nodos ocultos periódicamente, cuando la red entrena, seguramente se observará que algunos pesos en ciertos nodos no cambian mucho de acuerdo a sus valores iniciales. Estos nodos puede que no participen del proceso de aprendizaje, y con ello es posible reducir el número de nodos ocultos, al mínimo necesario.

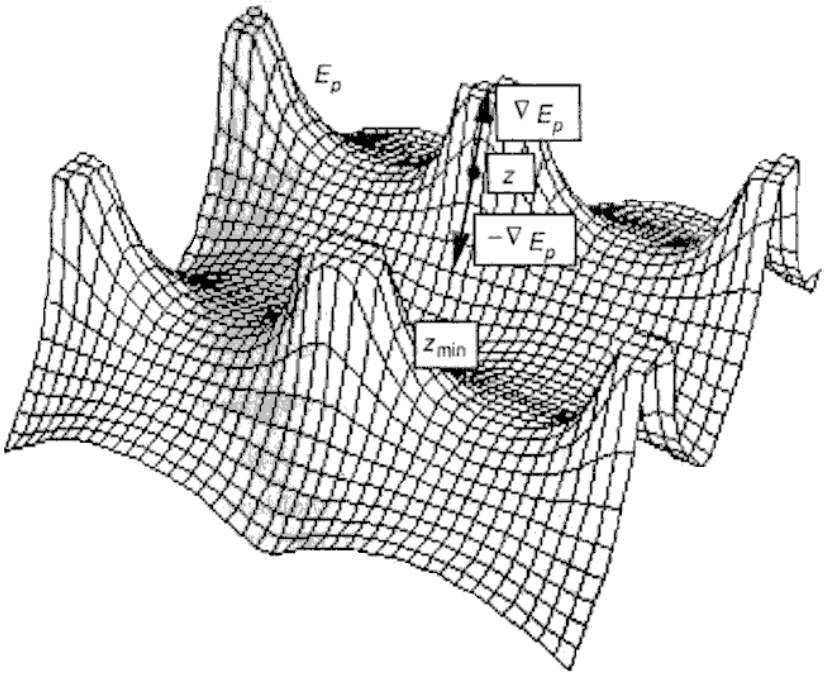


Figura 4. Una superficie hipotética de pesos en el espacio. El gradiente, ∇E_p , el punto z aparece a lo largo del gradiente negativo. Los cambios de pesos deberían ocurrir en la dirección del gradiente negativo, el cual es el punto más bajo de la superficie en el punto z. Además, los pesos cambiados deberían estar interactuando hasta que E_p alcance el punto mínimo que en este caso es z_{min}.

El factor de un ½ en la **ecuación 1** es por conveniencia para el calculo de la derivación. Con esto una constante arbitraria aparecerá en el resultado final, y la presencia de este factor no invalida la derivación.

Al determinar la dirección en la cuál cambian los pesos, debemos calcular el gradiente negativo de Ep, por lo tanto, ∇E_p con respecto a los pesos, w_{kj} . Por lo tanto, podemos ajustar los valores de los pesos tal que el error sea reducido. Esto sirve para pensar en Ep como una superficie de pesos en el espacio. La **figura 4** muestra un ejemplo en donde la red tiene únicamente pesos. La **figura 5** muestra un corte transversal de una superficie de pesos.

Para mantener de manera simple el desarrollo, se considerara cada componente de ∇E_p en forma separada. De la **ecuación (1)** y de la definición de δ_{pk}, obtenemos la siguiente ecuación

Actualización de Pesos en los Estratos

En la derivación de la regla de la delta, el error para el k ésimo elemento de entrada del vector es:

$$\delta_{pk} = (y_{pk} - d_{pk})$$

y donde la salida deseada es y_{pk} y la salida actual es o_{pk}. y los sufijos "p" se refiere al p-ésimo vector de entrenamiento, y "k" se refiere a la k-ésima unidad de salida. El error, que es minimizado por la regla generalizada de la delta, la cuál a su vez es el algoritmo de aprendizaje para la red, es la suma de los cuadrados de los errores de todas las unidades de salida:

$$E_p = \frac{1}{2}\sum_{k=1}^{M} \delta^2{}_{pk} \quad \text{.....(1)}$$

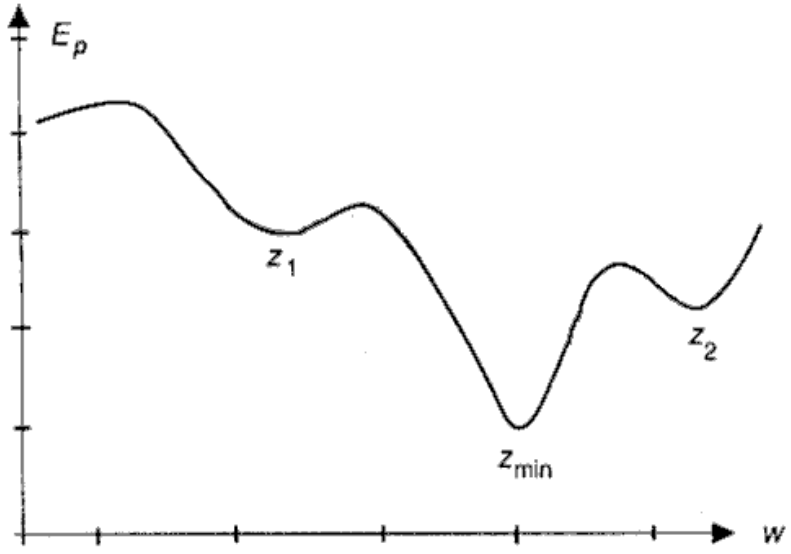


Figura 5 Esta gráfica muestra un corte de una superficie hipotética de error de pesos en el espacio. El punto, zmin, es llamado el mínimo global. Nota sin embargo, que hay otros puntos mínimos, z1 y z2. Un gradiente descendente busca el mínimo global lo que podría accidentalmente encontrar uno de estos mínimos locales en vez del mínimo global.

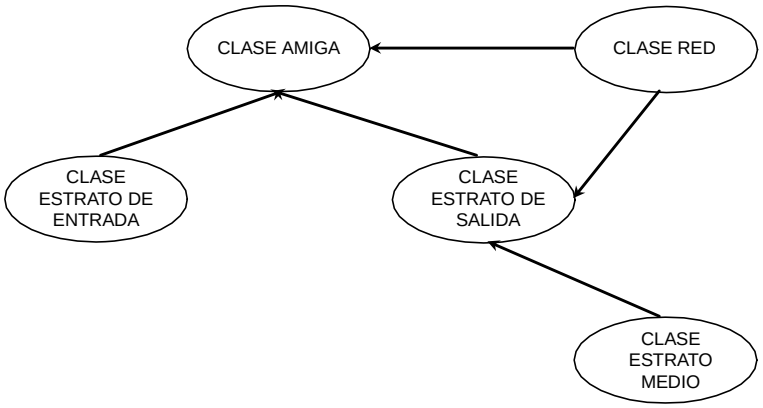


Figura 6. Muestra las clases de la red del tipo de RPI y su relación mutua en POO

$$E_p = \frac{1}{2}\sum_k (y_{pk} - o_{pk})^2 \quad \text{.....(2)}$$

y

$$o_{pk} = f_k^o(net_{pk}^o) \quad \text{.....(3)}$$

Sustituyendo (3) en (2) y derivando parcialmente

$$\frac{\partial E_p}{\partial w_{kj}^o} = -(y_{pk} - o_{pk})\frac{\partial f_k^o}{\partial (net_{pk}^o)}\frac{\partial (net_{pk}^o)}{\partial w_{kj}^o} \quad \text{.....(4)}$$

donde utilizamos la regla de la cadena para derivadas parciales. Por el momento, no intentaremos evaluar la derivada de f_k^o, en cambio escribiremos esto de manera más simple como:

$$f_k^{o'}(net_{pk}^o)$$

El último factor de la **ecuación (4)** es:

$$\frac{\partial (net_{pk}^o)}{\partial w_{kj}^o} = \left(\frac{\partial}{\partial w_{kj}^o}\sum_{j=1}^{L} w_{kj}^o i_{pj} + \theta_k^o\right) = i_{pj} \quad \text{.....(5)}$$

$$\frac{\partial E_p}{\partial w_{kj}^o} = -(y_{pk} - o_{pk}) f_k^{o'}(net_{pk}^o) i_{pj} \quad \text{.....(6)}$$

donde los pesos a la salida del estrato de salida se actualiza de acuerdo a la **ecuación (7)**.

$$w_{kj}^o(t+1) = w_{kj}^o(t) + \Delta_p w_{kj}^o(t) \quad \text{.....(7)}$$

$$\Delta w_{kj}^o = \eta (y_{pk} - o_{pk}) f_k^{o'}(net_{pk}^o) i_{pj} \quad \text{.....(8)}$$

donde h es el parámetro de proporción de aprendizaje en la ecuación de actualización de los pesos (8).

Implementación en Software de la Red Tipo RPI

En la **figura 6** se muestran las clases utilizadas en el programa y su relación entre ellas. La clase *estrato* se ramifica en dos. Una de estas ramificaciones es la clase *entrada_de_estrato* y la otra es la clase *salida_de_estrato*. La clase *estrato_medio* es muy parecida a la *salida_de_estrato* por lo cuál es derivada de esta clase.

La **figura 7** muestra a la red neuronal, con tres estratos para esta red. Un estrato contiene neuronas y pesos. El estrato es responsable de calcular su salida y almacenarla, y calcular y almacenar los errores para cada una de las neuronas. Notese que la clase de entrada no tiene pesos asociados a ella y por lo tanto es un caso especial, ya que el propósito del estrato de entrada es almacenar datos que han de ser propagados al siguiente estrato.

La **figura 8** es un esquema que trata de mostrar de manera clara una red de propagación inversa, con el cuál se pueden crear variaciones sobre la red y sus respectivas conexiones de retroalimentación.

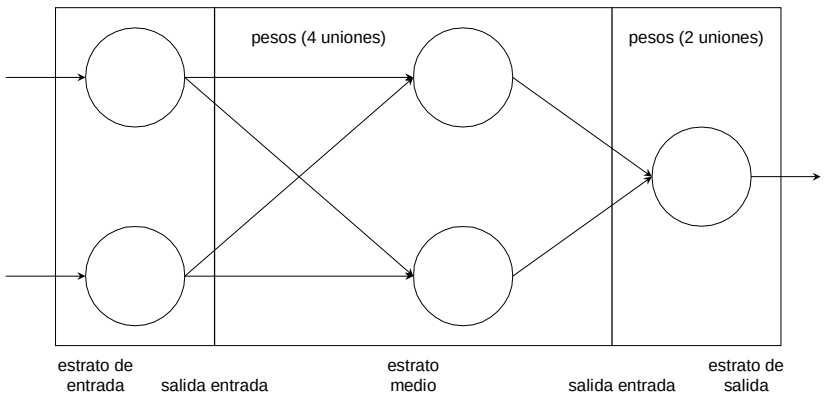


Figura 7. Organización de los estratos para la red de propagación inversa.

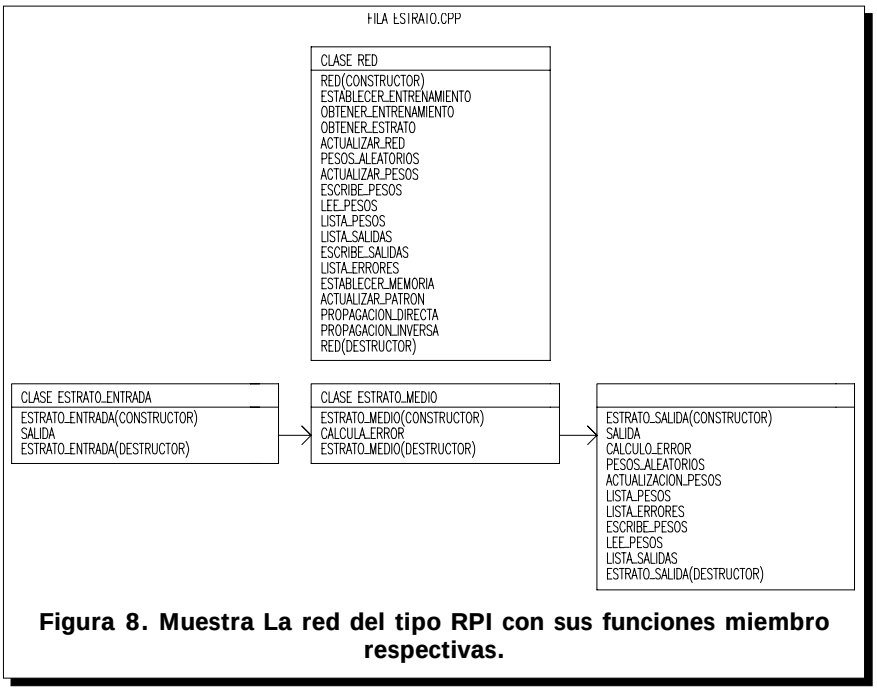


Figura 8. Muestra La red del tipo RPI con sus funciones miembro respectivas.

Las clase de una red de propagación inversa almacena arreglos de punteros a estratos y arreglos de tamaño de estratos para todos los estratos definidos. Estos objetos tipo estrato y arreglos son dinámicamente localizados en el heap con las palabras clave *new* y *delete* de C++. Como se puede ver, la red de propagación inversa puede rápidamente saturar la memoria y el CPU de una computadora, con redes largas y condiciones extensas de entrenamiento. El tamaño y arquitectura de la red por lo tanto estarán determinadas por la memoria y el CPU de la computadora.

La red del tipo RPI con sus clases y funciones miembros se muestra en la **figura 8**.

Conclusiones

Este modelo es uno de los más poderosos dentro de la teoría de redes neuronales, pero cuanta con un problema, y es que requiere de un gran poder de computo debido su sistema de entrenamiento, pero es excelente para el reconocimiento de patrones, por lo cuál se hace importante el encontrar un algoritmo adecuado para cada necesidad, y lograr disminuir su gran consumo de recursos computacionales.

Bibliografía

[1] García-Pelayo y Gross, Ramón; *"Diccionario Pequeño Larousse en Color"*, 1981.

[2] *"Standard Dictionary of the English Language"*; Funk & Wagnalls, 1967.

[3] Wienderhold, Gio; *"File Organization for Data Base Design"*; McGraw-Hill, 1967.

[4] Shaft, Adam; *"Historia y Verdad. Teoría y Praxis"*.

[5] Villee, Claude A.; *"Biología"*, 1986.

[6] van Gigch, Jonh P.; *"Teoría General de Sistemas"*, 1990.

[7] Hofstadter, Douglas R.; *"Godel, Escher, Bach: Una Eterna Trenza Dorada"*; CONACYT, 1982.

[8] Flores, Edmundo; *"La Creación de la Nueva Ciencia del Caos y el Ocaso de la Meteorología y de la Econometría"*; El BUHO, No. 253, EXCELSIOR, 1990.

[9] Freeman, James, *"Neural Networks, Algorithms, Applications, and Programming Techniques"*, Addison Wesley, 1991.

[10] Blum, Adam, *"Neural Networks in C++"*, Wiley, 1992.

[11] Kosko Bart, *"Neural Networks and Fuzzy Systems"*, Prentice Hall 1992.

[12] Freeman & Skapura, *"Neural Networks"*, Addison Wesley 1992.

La Filosofía de Windows: El Paradigma del Paso de Mensajes

Ing. Eduardo Vega Alvarado
Jefe del Departamento de Laboratorios Ligeros del CINTEC-IPN

Dado su gran éxito comercial, Windows se ha convertido en el estándar de facto para el desarrollo de aplicaciones dirigidas al usuario general. Varias han sido las razones para este suceso, destacando entre las más importantes, su interfaz de tipo gráfico con el usuario, el manejo multitarea, y la relativa independencia de los programas de aplicación en relación al hardware sobre el que se ejecutarán.

Sin duda alguna, el ambiente de trabajo amigable ("friendly") de Windows constituye la característica más apreciada por el usuario, aunque esto plantea al programador nuevos y más complejos problemas para el desarrollo de aplicaciones. En este artículo se presenta una descripción general del entorno Windows y de los modelos tanto de desarrollo de aplicaciones como de ejecución de las mismas; en artículos posteriores se ejemplificarán las técnicas específicas para el diseño de programas enfocados a este sistema.

Windows como un Sistema Operativo

Windows ofrece al programador más de 1500 funciones o llamadas al sistema (todas ellas diferentes), conocidas colectivamente como Interface para Programación de Aplicaciones (API, "Application Programming Interface"), y que se encuentran disponibles a través de los archivos de biblioteca KERNEL.EXE, USER.EXE, y GDI.EXE. Estas bibliotecas son del tipo de encadenamiento dinámico (DLL**,** "Dynamic Link Library), por lo que su acceso se efectúa al momento de ejecución del programa de aplicación, evitándose así el desperdicio de espacio de almacenamiento en disco ocasionado por la presencia de código objeto duplicado.

El archivo KERNEL proporciona los servicios del sistema, tales como el manejo de recursos, la ejecución de aplicaciones, la administración de memoria, y la multitarea; por su parte, la biblioteca GDI ("Graphics Device Interface", Interface de Dispositivo Gráfico) dirige las operaciones para crear gráficos tanto en el monitor como en la impresora, interactuando con el manejador de video del sistema. Finalmente, la función de USER es crear y mantener a las ventanas de aplicación, así como procesar los comandos del teclado y el ratón para mover, cambiar de tamaño o cerrar ventanas.

Por otra parte, Windows lleva a efecto la administración de los recursos del sistema, incluyendo no solo aquellos referentes al hardware (principalmente dispositivos de entrada/salida), sino también los suministrados por el Windows mismo, tales como el menú de sistema, las abreviaturas o aceleradores del teclado, los íconos y mapas de bits, entre otros, así como las ventanas mismas. Dada esta gestión, las aplicaciones no deben pretender el acceso directo a los dispositivos de entrada/salida ni a las unidades de memoria.

No obstante ser Windows un sistema de explotación multitarea, la interrupción de un programa para pasar el control a otro programa se rige por un esquema de tipo no prioritario. Esto significa que un proceso no es interrumpido, sino que se interrumpe así mismo para permitir que otros programas se puedan ejecutar; todo en base al llamado **paradigma de paso de mensajes**. Para un programa Windows, los mensajes constituyen las unidades de tratamiento, las cuales producen la verdadera repartición de tiempo en periodos en los cuales la aplicación es realmente ejecutada. En la versión Windows95, si bien el sistema presenta un método diferente para manejo de multitarea (ver recuadro), también se sigue permitiendo el esquema de paso de mensajes, con la finalidad de conservar compatibilidad con todos los programas de aplicación desarrollados para versiones de Windows anteriores a esta.

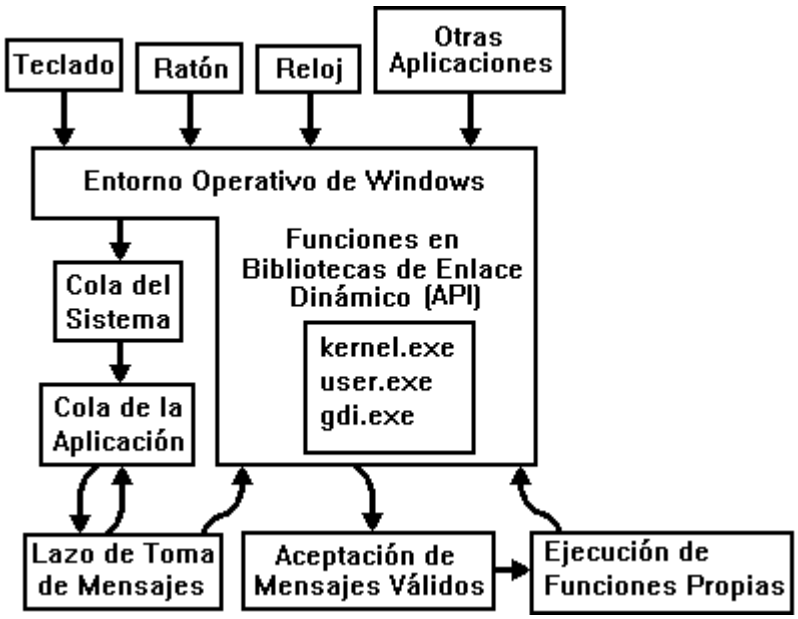


Figura 1. Paradigma del Paso de Parámetros

Paradigma del Paso de Mensajes

. Para el manejo y ejecución de los programas de aplicación, Windows se diseñó sobre el modelo conocido como **paradigma de paso de mensajes**. El esquema de este modelo se muestra en la **figura 1**.

En general, se dice que un mensaje (en el contexto Windows) es la notificación de la ocurrencia de un evento, el cual puede requerir que se produzca alguna acción específica. Como se observa en la figura, los mensajes son recibidos primeramente por Windows, quien los toma de la cola del sistema y los coloca en la cola de la aplicación requerida, informando a la misma de la presencia de un evento.

Cada vez que Windows envía un mensaje a una aplicación, esta se activa y recibe tiempo del procesador; de hecho, la aplicación solo dispone de atención del procesador al recibir un mensaje. Por ello, normalmente se dice que la tarea principal de cualquier aplicación para Windows es procesar mensajes. Sin embargo, es importante considerar que la aplicación debe compartir el tiempo del procesador y no intentar tomar el control exclusivo, ya que aunque Windows es un sistema multitarea, no puede obtener el control del procesador a menos que la aplicación lo libere después de ejecutar algún mensaje.

Básicamente, existen cuatro fuentes para la generación de mensajes, que son: el usuario (por medio de los dispositivos de entrada/salida), el sistema Windows, la aplicación misma, y otras aplicaciones en ejecución.

Las Ventanas y los Menús

Cada vez que se inicia una aplicación se crea una interface visual entre el usuario y el programa, denominada **ventana** (de ahí el nombre de Windows); toda aplicación tiene una ventana principal que sirve como su referencia. Cualquier ventana es tratada por Windows como un objeto, por lo que debe existir una clase **(class)** para modelarla (en programación orientada a objetos, un objeto es una particularización sobre un modelo constituido por una estructura de datos y funciones diversas que actúan sobre esa estructura, denominándose clase a dicho modelo).

¿Qué es Windows95?

Quizá la característica más importante de Windows95 **(W95)** es ser un sistema operativo completo en 32 bits, lo que elimina la necesidad de contar con una copia separada de MS-DOS; sin embargo, aún proporciona compatibilidad con aplicaciones escritas en 16 bits para versiones anteriores de Windows. W95 puede ejecutar cuatro tipos diferentes de programas: aquellos escritos para MS-DOS, los generados para Windows 3.1 **(W3.1)**, los escritos para Windows NT y los diseñados específicamente para W95, creándose automáticamente el medio ambiente adecuado para el tipo de programa presente.

W95 y W3.1

Desde el punto de vista del usuario, W95 proporciona una interface gráfica mejorada, basada en una organización centralizada para manejo de documentos **(folders)**. Así, el estilo de las ventanas ha sido modificado y se han agregado nuevos elementos de control, con la finalidad de permitir que cualquier usuario pueda ejecutar cualquier aplicación, aún sin un entrenamiento previo sobre el manejo del sistema.

Dado el enfoque visual de Windows, la ventana constituye el dispositivo primario para entrada/salida al medio ambiente de operación. Windows maneja las características estándar de las ventanas tales como el marco, las barras de desplazamiento y las barras de título, mientras que la aplicación se encarga del resto, destacando en ello la llamada área del cliente **(client area)**.

Un menú es una lista de comandos a desarrollar por la aplicación; aunque el programador determina los comandos específicos, el medio ambiente Windows se encarga de controlar la barra de menú (el seleccionador de las distintas opciones). Así mismo, existe un menú especial conocido como menú del sistema, el cual ya está determinado (con comandos fijos) como uno más de los recursos que ofrece Windows a sus usuarios.

Desarrollo de Aplicaciones

La filosofía de trabajo de Windows implica el que sus aplicaciones presenten características muy particulares en relación a los programas diseñados para MS-DOS. El desarrollo e implantación de aplicaciones para Windows debe contemplar el manejo de nuevas herramientas tales como los editores y los compiladores de recursos. Usualmente, una aplicación utiliza sus propios recursos, los cuales deben predefinirse en un archivo especial denominado **archivo de recursos**. Para la creación y complementación de estos archivos se emplean los editores de recursos (por ejemplo, WorkShop de Borland); posteriormente estos archivos se compilan utilizando un **compilador de recursos** (tal como RC, del mismo Borland), Los archivos de recursos no se usan directamente por el programa, sino que se encadenan al archivo .EXE de la aplicación.

Además del archivo de recursos (.RES), las aplicaciones también incluyen los siguientes archivos:

- Archivo de Definición del Módulo (.DEF),
- Archivo Fuente de la Aplicación (.C o .CPP) y,
- Archivo(s) de Encabezado (.H).

El archivo de definición del módulo es un archivo tipo texto (ASCII) que especifica, entre otras opciones, el nombre de la aplicación y los nombres de las funciones que servirán como punto de entrada a la misma, así como información al programa

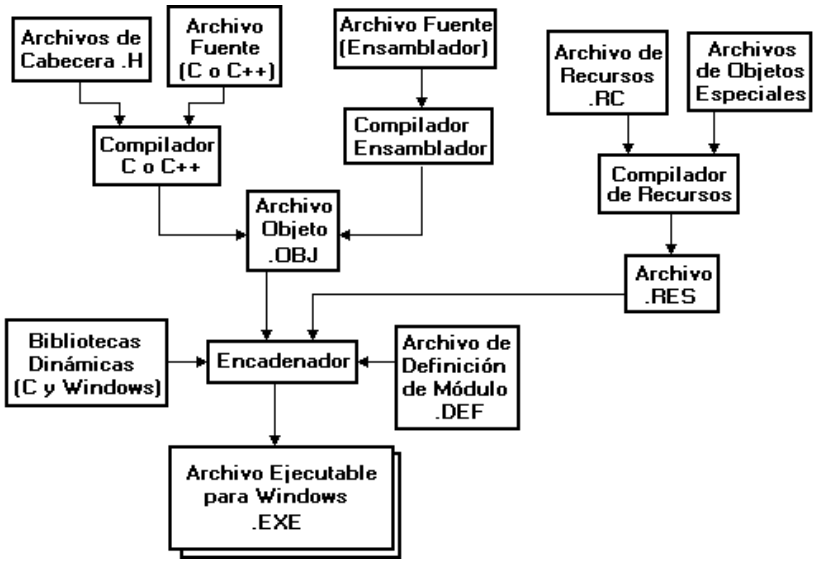


Figura 2. Procesamiento de archivos

Desde el punto de vista del programador, W95 presenta dos grandes mejoras. La primera de ellas la constituye el uso de memoria virtual y el direccionamiento completo a 32 bits, lo que permite a una aplicación disponer de hasta 4 GB de memoria virtual lineal. La segunda innovación corresponde a la multitarea. En W3.1 las aplicaciones corren concurrentemente por medio de un método conocido como multitarea cooperativa, en el que la aplicación verifica periódicamente la cola de mensajes para continuar o regresar control al sistema operativo. W95 también soporta esta multitarea en base a procesos, usando el paradigma de paso de mensajes, aunque normalmente se utiliza un segundo método, denominado multitarea por bloques **(thread-based multitasking)**.

Un bloque es una entidad ejecutable que pertenece a un proceso simple, que incluye además de su código un contador de programa, una pila para modo usuario, otra pila para modo kernel, y un conjunto de valores de los registros; todos los bloques en un proceso tienen igual acceso al espacio de direcciones del procesador, a los manejadores de objetos, y a los demás recursos del sistema. En W95 estos bloques se implementan como objetos, y se ejecutan e interrumpen en base a tiempo, correspondiendo al sistema operativo dicha temporización; de esta forma, es el sistema directamente quién determina el intercambio, ejecución e interrupción de tareas.

encadenador sobre los atributos de los segmentos de código y de datos, y la declaración del tamaño de las áreas de heap y de stack. El archivo fuente, por su parte, contiene el código para el manejo de la aplicación, mientras que en el archivo de encabezado se declaran las definiciones de variables, las listas de parámetros y los prototipos de las funciones empleadas en el archivo fuente. Como ya se mencionó anteriormente, en el siguiente artículo de esta serie se explicarán con mayor detalle estos archivos, principalmente en lo referente a la estructuración del archivo fuente, y la descripción de recursos propios en el archivo de definición de los mismos..

En la **figura 2** se ilustran los 4 archivos y el procesamiento necesario para obtener el programa .EXE final.

Windows maneja un mecanismo especial para encadenar a las aplicaciones con las funciones en los archivos de biblioteca, todo ello a tiempo de ejecución. Para conseguirlo, las aplicaciones están formuladas en un nuevo formato de archivo .EXE, conocido como **nuevo ejecutable**.

Bibliografía

]1] Adams, Lee. *"Programación Avanzada de Gráficos en C para Windows"*. Windcrest/McGraw-Hill. 1993.

[2] Ezzell, Ben. *"Windows 3.1 Graphics Programming"*. ZD Press. 1992.

[3] Farrell, Tim & Runnoe, Connally. *"Programming in Windows 3.1"*. QUE Corporation. 1992.

[4] Klein, Mike. *"Windows Programmer's Guide to DLLs and Memory Management"*. SAMS. 1992.

[5] Microsoft. *"Microsoft Windows95 Resource Kit"*. Microsoft Press. 1995.

[6] Murray, William H. & Pappas, Chris H.. *"Windows 3.1 Programming"*. Osborne/Mc Graw-Hill. 1992.

[7] Schildt, Herbert. *"Schildt's Windows 95 Programming in C and C++"*. Osborne/Mc Graw-Hill. 1995.

Manejo de Objetos en un Editor Gráfico en Tres Dimensiones

Ing. Amadeo Argüelles Cruz
Alumno de la Maestría del CINTEC-IPN.
Ing. Rubén García Duana
Alumno de la Maestría del CINTEC-IPN.

El presente artículo describe un programa para el manejo de imágenes de objetos en 3 dimensiones. Como el tema es bastante extenso, se dan solo los principios del manejo de éstos y se ejemplificará con algunas figuras de fácil comprensión. Se consideran en este trabajo, los siguientes puntos:

Generación de un numero restringido de objetos tridimensionales en memoria.

Representación gráfica de los objetos en la pantalla.

Implantación de operaciones básicas como son **Mover** y **Rotar** objetos en función de las coordenadas X, Y & Z.

Manejo de un ambiente amigable mediante la aplicación de pantallas gráficas de interacción con el usuario, como lo son menús, cajas de dialogo, etc.

Lectura y escritura de archivos de almacenamiento de los gráficos elaborados.

Impresión de las imágenes en una impresora de matriz de puntos.

Filosofía

Para la elaboración del programa, se optó por implementar un ambiente gráfico utilizando lenguaje C++ orientado a objetos. Es importante hacer notar que el programa fue elaborado en un ambiente DOS y no para Windows, debido a que la generación de un ambiente gráfico propio nos permitirá dominar la técnica de manejo de ventanas, menús, etc.; elementos que por estar definidos para Windows, restringen su utilización solo a hacer un llamado a sus funciones. Al ser independiente de la plataforma Windows, este paquete es fácilmente exportable ; además, al ser escrito para DOS resulta muy compacto, lo que permite ser ejecutado con características muy ventajosas.

La solución al problema está dividida en cuatro áreas:

1) Creación del ambiente gráfico.

2) Generación y representación de objetos tridimensionales.

3) Tratamiento de los objetos en tres dimensiones.

4) Manejo de E/S de la información generada.

A continuación se desglosarán las soluciones para cada una de las áreas:

Creación del Ambiente Gráfico

Dibujar la pantalla principal:

Se inicializa el modo gráfico y se dibuja el marco de la pantalla, se colocan los encabezados y la barra de menús.

Acceso a los menús:

Los menús se invocan presionando la tecla «Alt» y la tecla asociada para cada uno de ellos, dicha tecla es la letra mayúscula de las opciones de los menús; la alternativa <Alt>X para salir también esta habilitada.

Generación de una caja de dialogo general para todos los casos:

Para las opciones del menú se ideó una caja de dialogo general, mediante una función que lleva como parámetro un apuntador al titulo de la caja; a esta se añadió una función general que permitiera la selección de cualquier opción del menú por medio de las flechas, desde dos hasta nueve selecciones como máximo.

Generación y representación de objetos tridimensionales.

Generar objetos tridimensionales:

El cubo se genera dando la posición espacial de uno de los vértices, y la longitud de los lados del cubo en

cada una de las coordenadas; cuando proporcionamos estos datos, se ejecuta una función que genera los siete vértices restantes del cubo, y estos quedan grabados para cada objeto.

El cilindro lee las coordenadas de la base en las tres dimensiones, su longitud, su radio y la dirección del objeto sobre uno de los ejes. La función que dibuja al cilindro necesita generar dos circunferencias, una para la tapa y la otra para la base; se desarrollaron dos métodos para dibujar las circunferencias: uno es dibujándola punto a punto, y el otro generando un polígono de 20 vértices. La creación del objeto incluye los 40 vértices que generan al cilindro (20 para cada circunferencia).

La pirámide se crea con la misma filosofía que los objetos anteriores: la rutina lee las coordenadas de la punta, la altura y la dirección de la pirámide, el número de lados de la base (máximo 20) y el radio de la circunferencia en la que se encontrará inscrito el polígono de la base. Dicho programa calcula las posiciones de los vértices de la pirámide.

Dibujar objetos tridimensionales en pantalla:

Para el problema de dibujar los objetos tridimensionales en pantalla, se consideraron las coordenadas X & Y perpendiculares con el eje Z a 135 grados de ambos. La forma de conversión de coordenadas espaciales a planas se hace de la siguiente manera: Las coordenadas X & Y en 3 dimensiones, corresponden unitariamente a las coordenadas X & Y planas; la coordenada Z espacial se descompone en 2: una proyección en X, dada por su componente a 135 grados (usando el coseno), y una proyección en Y dada por su proyección en Y (usando el seno). Hay que recordar que en las coordenadas en pantalla, se tiene el origen (0,0) en la esquina superior izquierda y que Y crece hacia abajo, es decir, en sentido inverso al plano cartesiano propio; es por esto que en la conversión de coordenadas planas a pantalla, se debe mover el origen e invertir el sentido de las componentes Y.

Tratamiento de los objetos en tres dimensiones.

Girar objetos «base» en 3 dimensiones:

Para rotar un punto, ésto se efectúa en el espacio tridimensional y sobre los ejes X, Y & Z. Primero se calcula el vector del punto, es decir, su magnitud tomada desde el origen y su ángulo proyectado con respecto a los ejes; los ángulos obtenidos se incrementan con los ángulos de rotación del punto. Se calculan las nuevas proyecciones del vector en cada uno de los ejes, obteniendo con esto las nuevas coordenadas del punto rotado. Una vez girado el punto, se procesa como cualquier otro, para ser desplegado en pantalla.

Generación de círculos en 3 dimensiones:

Las circunferencias se calculan por medio de las ecuaciones trigonométricas del circulo, es decir, la función seno en cada coordenada (solo 2 de ellas) defasadas 90 grados entre si. La generación de los circunferencias utilizando este método proporciona una gran resolución, pero tiene como desventaja el ser demasiado lento el cálculo de todos los puntos. Este detalle es muy notorio en máquinas lentas; para corregir éste inconveniente, se ideó la forma de dibujar una circunferencia de baja resolución, ésto se logra produciendo un polígono regular con número alto de lados, lo cual a la vista da la sensación de una circunferencia; el número de lados utilizados es de 20, dado como valor fijo. La resolución utilizada para las circunferencias es conmutable por el usuario.

Manejo de Entrada/Salida (E/S) de la información generada.

Lectura y escritura de archivos:

Para la escritura del archivo, se graba un entero que contiene el número de cubos, e inmediatamente todos los objetos cubos; después se graba otro entero, pero ahora con el número de cilindros, seguido de todos los objetos cilindro; finalmente para las pirámides se realizo el mismo procedimiento. Para la lectura, se lee el primer carácter del archivo que es el número de cubos, y se realiza un ciclo de creación y lectura del todos los objetos cubos; se repite este procedimiento para los cilindros y las pirámides.

Impresión de archivos:

Para lograr habilitar esta función, se utiliza de la interrupción 05h del Bios, la cual realiza una impresión de la vista actual de la pantalla. Para ejecutar dicha interrupción, se debe limpiar la pantalla y solo dibujar los objetos, con el fin de no imprimir el marco y la barra de menús. Hay que recordar que para que la interrupción 05h pueda ejecutar la impresión en modo gráfico, es necesario cargar previamente el programa del sistema operativo llamado GRAPHICS.COM.

Algoritmo

Tomando en cuenta los elementos de este análisis , el paquete desarrollado efectúa las operaciones de acuerdo al siguiente algoritmo:

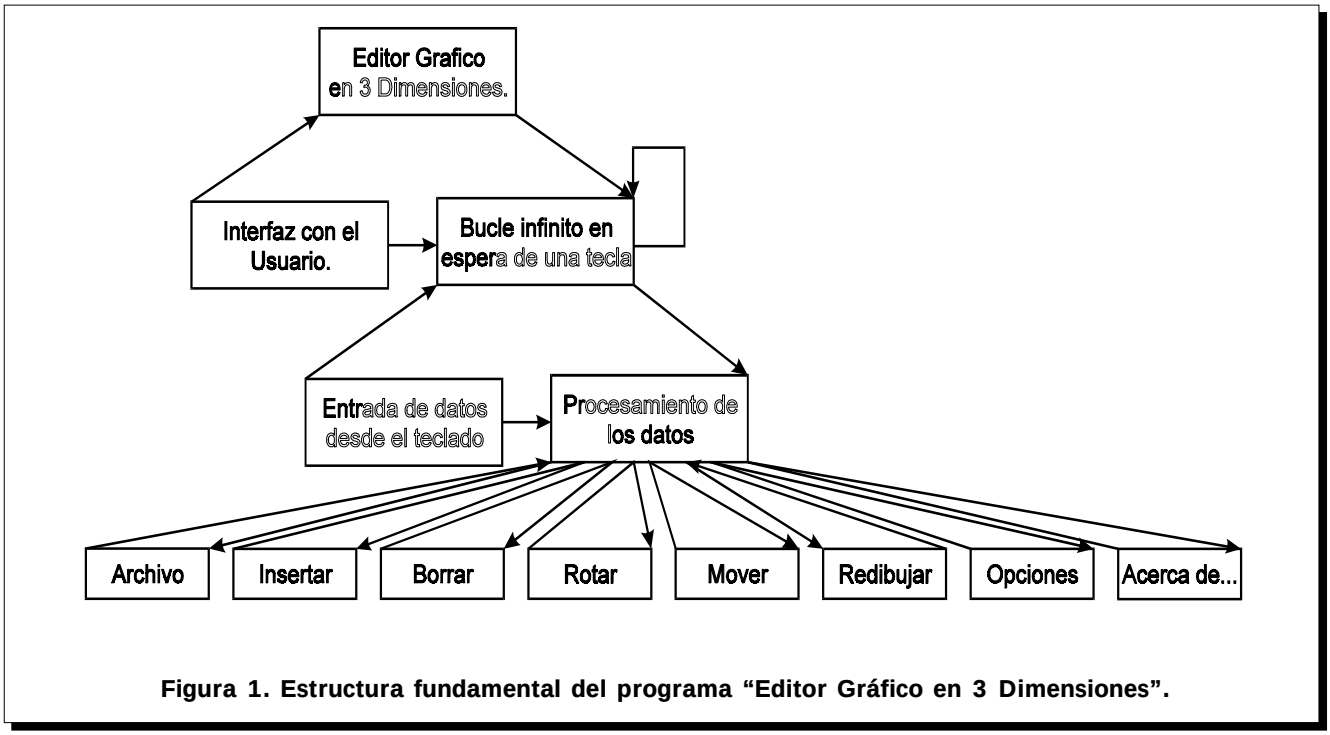


Figura 1. Estructura fundamental del programa "Editor Gráfico en 3 Dimensiones".

Inicialización del modo gráfico.
Limpieza de Pantalla.
Presentación de la interface gráfica.
Trabajo en ciclo infinito en la espera de acontecimientos (cualquier tecla o juego de teclas presionadas).

Una vez sucedido un evento, se efectúa su procesamiento ejecutando la función correspondiente.

De acuerdo a la opción escogida dentro de los menús de la interface gráfica, se lleva a cabo el proceso apropiado para cada una de las funciones a desarrollar.

Método Gráfico

A continuación se presenta el método gráfico empleado para llegar a la especialización de las tareas. En primer lugar se inicializa el ambiente gráfico, a continuación el proceso de ciclo infinito se desarrolla en la espera de mensajes. Esto se muestra en la **figura 1**.

Descripción Particular de los Elementos Fundamentales del Programa

Como siguiente paso se describen las partes principales del programa, siendo consideradas de acuerdo a la solución mostrada anteriormente en las siguientes clases:

Para el manejo de los objetos en tres dimensiones y su conversión a la pantalla gráfica, existen dos funciones muy importantes que son: *conv_coor* y *rotador*. El código de estas y otras funciones se muestra en los recuadros siguientes.

❶La función *conv_coor*, hace la conversión de las coordenadas en tres dimensiones de los objetos a dos dimensiones, que son las correspondientes a las de la pantalla.

❷La función «rotador», efectúa el cálculo de coordenadas de la nueva posición, utilizando para ello las antiguas coordenadas y los ángulos de rotación definidos para cada uno de los ejes X, Y & Z.

❸La lectura de cualquiera de los objetos solo consiste en leer cada uno de los elementos del objeto; una vez que se ha leído el objeto, este se genera y se dibuja. Véase el código de la generación y dibujo de un cubo.

❹Una rutina interesante es la que dibuja el cilindro, debido a que se deben generar las circunferencias a cualquier rotación, en alta y baja resolución.

❺Otra rutina digna de atención es la opción ARCHIVO; esta función permite leer y escribir archivos, así como imprimir e inclusive terminar el programa.

```
//    Convertidor de Coordenadas de 3 dimensiones a coordenadas
//    de pantalla.
void conv_coor(float X1,float Y1,float Z1,int *x,int *y)
{
     int aux;
     aux=xc+escala*(X1-(0.4*Z1));          // Obtiene coordenada X
     *x=aux;
     aux=yc+escala*(-Y1+(0.4*Z1));         // Obtiene coordenada Y
     *y=aux;
}
```

Función *conv_coor* ❶

```
// Genera los vértices del cubo.
void cubo::gen_cubo(void)
{
  float x,y,z;              // Variables intermedias              // Vértice

 conv_coor(xor,yor,zor,&vertice[0][0],&vertice[1][0]);       // 1ero
   rotador(xlong,0,0,rx,ry,rz,&x,&y,&z);
 conv_coor(xor+x,yor+y,zor+z,&vertice[0][1],&vertice[1][1]);   // 2o
   rotador(xlong,ylong,0,rx,ry,rz,&x,&y,&z);
 conv_coor(xor+x,yor+y,zor+z,&vertice[0][2],&vertice[1][2]);   // 3er
   rotador(0,ylong,0,rx,ry,rz,&x,&y,&z);
 conv_coor(xor+x,yor+y,zor+z,&vertice[0][3],&vertice[1][3]);   // 4o
   rotador(0,0,zlong,rx,ry,rz,&x,&y,&z);
 conv_coor(xor+x,yor+y,zor+z,&vertice[0][4],&vertice[1][4]);   // 5o
   rotador(xlong,0,zlong,rx,ry,rz,&x,&y,&z);
 conv_coor(xor+x,yor+y,zor+z,&vertice[0][5],&vertice[1][5]);   // 6o
   rotador(xlong,ylong,zlong,rx,ry,rz,&x,&y,&z);
 conv_coor(xor+x,yor+y,zor+z,&vertice[0][6],&vertice[1][6]);   // 7o
   rotador(0,ylong,zlong,rx,ry,rz,&x,&y,&z);
 conv_coor(xor+x,yor+y,zor+z,&vertice[0][7],&vertice[1][7]);   // 8o
}

// Dibuja el cubo en pantalla.
void cubo::dib_cubo(void)
{
   // Se dibujan las líneas entre los vértices para generar el cubo.

   line(vertice[0][0],vertice[1][0],vertice[0][1],vertice[1][1]);
   line(vertice[0][0],vertice[1][0],vertice[0][3],vertice[1][3]);
   line(vertice[0][0],vertice[1][0],vertice[0][4],vertice[1][4]);
   line(vertice[0][2],vertice[1][2],vertice[0][1],vertice[1][1]);
   line(vertice[0][2],vertice[1][2],vertice[0][3],vertice[1][3]);
   line(vertice[0][2],vertice[1][2],vertice[0][6],vertice[1][6]);
   line(vertice[0][7],vertice[1][7],vertice[0][3],vertice[1][3]);
   line(vertice[0][7],vertice[1][7],vertice[0][4],vertice[1][4]);
   line(vertice[0][7],vertice[1][7],vertice[0][6],vertice[1][6]);
   line(vertice[0][5],vertice[1][5],vertice[0][1],vertice[1][1]);
   line(vertice[0][5],vertice[1][5],vertice[0][4],vertice[1][4]);
   line(vertice[0][5],vertice[1][5],vertice[0][6],vertice[1][6]);
}
```

Generación y Dibujo de un Cubo ❸

```
// Obtiene las nuevas coordenadas de un punto rotado.
void rotador(float xe,float ye,float ze,float ax,float ay,float az,float *xs,float
*ys,float *zs)


{
  float d,xm,ym,zm,alfa;
  int kerror;

  if(xe<0)kerror=-1;
  else kerror=1;
  d=sqrt(xe*xe+ye*ye);          // Calcula la magnitud y el angulo
  if(xe!=0)alfa=atan(ye/xe);   // de la proyeccion en el plano XY
  else
  {
    if(ye<0)alfa=-PI/2;
    else alfa=PI/2;             // Obtiene el signo p/angulo de 90
                                // grados
  }

  if(kerror==-1)alfa=PI+alfa;
  alfa=alfa+az;
  xm=d*cos(alfa);               // Obtiene las nuevas
  ym=d*sin(alfa);               // coordenadas del punto

  if(ze<0)kerror=-1;
  else kerror=1;
  d=sqrt(ze*ze+xm*xm);          // Calcula la magnitud y el angulo
  if(ze!=0)alfa=atan(xm/ze);   // de la proyeccion en el plano ZX
  else
  {
    if(xm<0)alfa=-PI/2;         // Obtiene el signo p/angulo de 90 grados
    else alfa=PI/2;
  }
  if(kerror==-1)alfa=PI+alfa;
  alfa=alfa+ay;
  zm=d*cos(alfa);               // Obtiene las nuevas coordenadas
  xm=d*sin(alfa);               // del punto
  if(ym<0)kerror=-1;
  else kerror=1;
  d=sqrt(ym*ym+zm*zm);          // Calcula la magnitud y el ángulo
  if(ym!=0)alfa=atan(zm/ym);   // de la proyeccion en el plano YZ
  else
  {
    if(zm<0)alfa=-PI/2;        // Obtiene el signo p/angulo de 90 grados
    else alfa=PI/2;
  }
  if(kerror==-1)alfa=PI+alfa;
  alfa=alfa+ax;

  ym=d*cos(alfa);               // Obtiene las nuevas coordenadas
  zm=d*sin(alfa);               // del punto
  *xs=xm;                       // Regresa las nuevas coordenadas
  *ys=ym;
  *zs=zm;
}
```

Función «rotador» ❷

```
// Dibuja el cilindro en la pantalla
void cilindro::dib_cil(void)
{
   float i,xa=0,ya=0,za=0,cont=0;
   int x,y,xx,yy;
   float d,alfa,xxa,yya,zza;

   if(resol==1)          // Pregunta por la resolución
   {                              // Si es alta resolución:
      for(i=0;i<2*PI;i=i+.01)  // Dibuja punto por punto los círculos
      {
         xa=Kx*radio*sin(i+(Kx*Ky*PI/2));
         ya=Ky*radio*sin(i+(Ky*Kz*PI/2));
         za=Kz*radio*sin(i+(Kz*Kx*PI/2));
         rotador(xa,ya,za,rx,ry,rz,&xxa,&yya,&zza);
         xxa=basex+xxa;
         yya=basey+yya;
         zza=basez+zza;
         conv_coor(xxa,yya,zza,&x,&y);
         putpixel(x,y,WHITE);
         if(Kx==0)xa=clong;
         if(Ky==0)ya=clong;
         if(Kz==0)za=clong;
         rotador(xa,ya,za,rx,ry,rz,&xa,&ya,&za);
         xa=basex+xa;
         ya=basey+ya;
         za=basez+za;
         conv_coor(xa,ya,za,&xx,&yy);
         putpixel(xx,yy,WHITE);
         if(cont>=PI/(2*sqrt(radio)))
         {
            line(x,y,xx,yy);
            cont=0;
         }
         cont=cont+.01;
      }
   }
   else           // Si es baja resolucion:
   {                              // Dibuja poligonos de 20 lados
      for(i=0;i<19;i++)
      {
         line(verticeb[0][i],verticeb[1][i],verticeb[0][i+1],verticeb[1][i+1]);
         line(verticet[0][i],verticet[1][i],verticet[0][i+1],verticet[1][i+1]);
         line(verticeb[0][i],verticeb[1][i],verticet[0][i],verticet[1][i]);
      }
      line(verticeb[0][0],verticeb[1][0],verticeb[0][19],verticeb[1][19]);
      line(verticet[0][0],verticet[1][0],verticet[0][19],verticet[1][19]);
   }
}
```

Generación y dibujo de un cilindro ❹

```
// Opcion Archivo
void ARCHIVO(void)
{
   char op,inst='n',arc;
   int i,j,k,n,l=0,sel=0;
   FILE *arch;                // Apuntador al archivo

   n=0;
   for(i=0;i<2;i++)           // Pone los elementos de menú
   {
      for(j=0;j<3;j++)
      {
         outtextxy(200+(i*100),80+(j*20),&Archivo[n][0]);
         n++;
      }
   }
   while(sel==0)
   {
      op=getch();
      if(op=='\r')op=inst;
      switch(op)              // Selecciona
      {
         case 'n':            // Nuevo
         case 'N':
            res_vent=0;
            putimage(170,50,mem,COPY_PUT);
            n=n_cubos;                    // Borra todos los elementos
            for(i=0;i<n;i++)borra_elem('c',0);
            n=n_cil;
            for(i=0;i<n;i++)borra_elem('i',0);
            n=n_pir;
            for(i=0;i<n;i++)borra_elem('p',0);
            REDIBUJAR();
            sel=1;
            break;
         case 'a':            // Abrir
         case 'A':
            line(170,140,470,140);
            outtextxy(200,150,»Nombre del Archivo: «);
            for(k=0;k<20;k++)archivo[k]=NULL;
            for(k=0;k<20;k++)          // Lee el nombre del nuevo archivo
            {
               archivo[k]=getch();
               if(archivo[k]=='\r')
               {
                  archivo[k]=NULL;
                  k=20;
               }
               gotoxy(30,12);
               cout<<archivo;
            }
            res_vent=0;
            putimage(170,50,mem,COPY_PUT);
            n=n_cubos;                    // Borra todos los elementos
            for(i=0;i<n;i++)borra_elem('c',0);
            n=n_cil;
            for(i=0;i<n;i++)borra_elem('i',0);
            n=n_pir;
            for(i=0;i<n;i++)borra_elem('p',0);
            arch=fopen(&archivo[0],»rb»);          // Abre el archivo
            fread(&n_cubos,sizeof(int),1,arch);    // Lee el archivo
            for(i=0;i<n_cubos;i++)                 // Cubos
            {
               ap_cubo[i]=new cubo;
               if(ap_cubo[i]==NULL)LM();
```

❺...

```
            fread(ap_cubo[i],sizeof(cubo),1,arch);
          }
          fread(&n_cil,sizeof(int),1,arch);
          for(i=0;i<n_cil;i++) // Cilindros
          {
            ap_cil[i]=new cilindro;
            if(ap_cil[i]==NULL)LM();
            fread(ap_cil[i],sizeof(cilindro),1,arch);
          }
          fread(&n_pir,sizeof(int),1,arch);
          for(i=0;i<n_pir;i++)      // Piramides
          {
            ap_pir[i]=new piramide;
            if(ap_pir[i]==NULL)LM();
            fread(ap_pir[i],sizeof(piramide),1,arch);
          }
          fclose(arch);             // Cierra el archivo
          REDIBUJAR();
          sel=1;
          break;
        case 'c':                   // Salvar como...
        case 'C':
          line(170,140,470,140);
          outtextxy(200,150,»Nombre del Archivo: «);
          for(k=0;k<20;k++)archivo[k]=NULL;
          for(k=0;k<20;k++)    // Lee el nombre del archivo
          {
            archivo[k]=getch();
            if(archivo[k]=='\r')
            {
              archivo[k]=NULL;
              k=20;
            }
            gotoxy(30,12);
            cout<<archivo;
          }                         // No termina
        case 's':                   // Salvar
        case 'S':
          arch=fopen(&archivo[0],»wb+»);
          fwrite(&n_cubos,sizeof(int),1,arch);      // Salva cubos

for(i=0;i<n_cubos;i++)fwrite(ap_cubo[i],sizeof(cubo),1,arch);
          fwrite(&n_cil,sizeof(int),1,arch);        // Salva cilindros
          for(i=0;i<n_cil;i++)fwrite(ap_cil[i],sizeof(cilindro),1,arch);
          fwrite(&n_pir,sizeof(int),1,arch);        // Salva piramides
          for(i=0;i<n_pir;i++)fwrite(ap_pir[i],sizeof(piramide),1,arch);
          fclose(arch);
          sel=1;
                  break;
        case 'i':
        case 'I':
          RED_FIG();
          int86(0x5,NULL,NULL);     // Imprime
          sel=1;
          REDIBUJAR();
          break;
        case 'l':                   // Salir

        case 'L':
          exit(0);
          break;

        case '\x0':                 // Flechas
          l=flechas(l,6);
          switch(l)
```

... ❺ ...

```
          {
            case 0:                 // Nuevo
              inst='n';
              break;
            case 1:    // Abrir
              inst='a';
              break;
            case 2:    // Salvar
              inst='s';
              break;
            case 3:    // Salvar como...
              inst='c';
              break;
            case 4:    // Imprimir
              inst='i';
              break;
            case 5:    // Salir
              inst='l';
              break;
          }
          break;
      }
    }
}
```

Función Archivo ... ❺

En la **figura 2** se muestra el ambiente de trabajo, en el cual ya se han insertado tres figuras: un cubo, un cilindro y una pirámide de cinco lados.

Como ya se ha mencionado, el programa permite Mover, Rotar, Borrar e Insertar objetos; partiendo de la pantalla anterior se girará el cilindro 90 grados con respecto al eje X, desplazando además al cubo de su posición original, se borrará la pirámide y se insertará una nueva más alta y con ocho lados. La pantalla queda como se muestra en la **figura 3**.

Por su parte, se muestra en la **figura 4** una caja de dialogo referente al menú ARCHIVO.

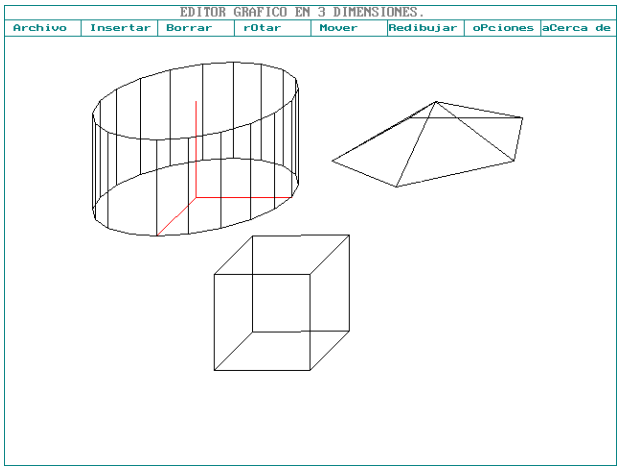


Figura 2

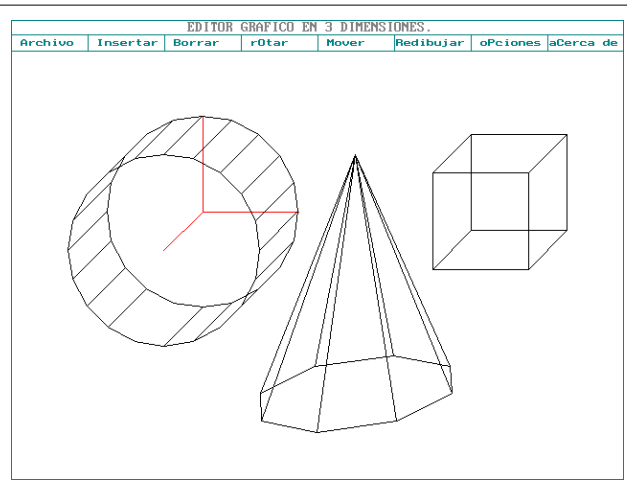


Figura 3

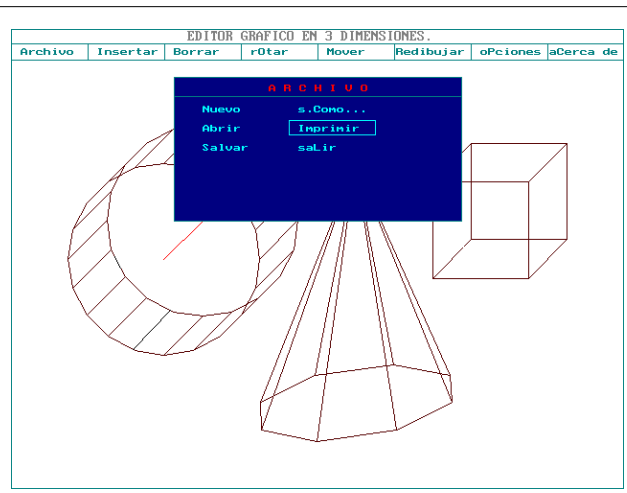


Figura 4

Conclusiones

El crear programas de ambiente gráfico, para el manejo elemental de figuras en tres dimensiones con una programación orientada a objetos, no solo facilita la realización de esta tarea, sino que además permite la generación y expansión de nuevas figuras tridimensionales a partir de las rutinas previamente diseñadas. Con esto se logra una reutilización del código y una reducción significativa en el consumo de tiempo de desarrollo de software, lo cual lo hace muy atractivo si tomamos en cuenta las necesidades de tener herramientas que nos permitan llevar a cabo nuestras tareas de una forma rápida y concisa.

Elaborar técnicas de desarrollo de software propias, para adecuarlas a nuestras necesidades de cómputo, nos hace ser menos dependientes de los cambios que sufren constantemente los actuales terrenos del software.

Bibliografía

[1] Herbert Schildt. *"Turbo C/C++: Manual de referencia"*. McGraw-Hill

[2] *"Funciones del DOS y BIOS"*. Addison-Wesley Iberoamericana.

Programación en Ambiente Windows Utilizando las Bibliotecas ObjectWindows de Borland

José Anibal Arias Aguilar
Alumno de la Maestría del CINTEC-IPN.
Osvaldo Espinosa Sosa
Alumno de la Maestría del CINTEC-IPN.

Es evidente que el uso del ambiente Windows se ha convertido en un estándar para el usuario de computadoras personales. Es por esto que se hace cada vez más necesario aprender a programar aplicaciones que funcionen bajo este ambiente. Existen dos tipos de programación para Windows: programación estándar y programación en base a objetos. Cuando se programa en base a objetos, la mejor alternativa consiste en utilizar las bibliotecas OWL ("ObjectWindows Library") que proporciona el compilador de Borland. En el presente artículo se pretende mostrar el mecanismo básico de una aplicación de este tipo implementando el popular juego Tetris ™.

La Programación Orientada a Objetos (POO)

La POO es una nueva forma de enfocar el trabajo de la programación. Toma las mejores ideas de la programación estructurada y las combina con nuevos conceptos que alientan una visión diferente de la tarea de la programación. La POO permite descomponer un problema en subgrupos de partes relacionadas. Así, se pueden traducir estos subgrupos en unidades autocontenidas llamadas objetos.

C++ es una versión expandida de C. Mantiene su eficiencia, flexibilidad y filosofía añadiendo al mismo tiempo soporte para la POO: objetos, polimorfismo y herencia.

Objetos. *Una clase es una entidad lógica que contiene datos y un código que manipula esos datos. Dentro de una clase, parte de ese código o datos pueden ser exclusivos de la clase e inaccesibles fuera de ella; a esto se le denomina encapsulación de datos. Los objetos se obtienen a partir de la definición de una variable del tipo clase, y esto se conoce como instanciación de clases.*

Polimorfismo. *El polimorfismo describe la capacidad del código para comportarse de diferentes maneras dependiendo de los tipos de datos que se estén tratando.*

Herencia. *La herencia es el proceso por el cual una clase puede adquirir las propiedades de otra. Esto es importante porque soporta el concepto de clasificación. Si se considera, la mayoría del conocimiento se hace manejable por medio de clasificaciones jerárquicas.*

Windows

Para el usuario corriente, Microsoft Windows es una extensión gráfica del sistema operativo MS-DOS. Windows extiende al DOS de diversas maneras: puede soportar múltiples programas concurrentes, maneja gráficas de alto nivel y proporciona una interface estándar de comunicación con los programas. Los programas que corren bajo esta plataforma son manejados por eventos, es decir, la estructura y operación de estos programas se centra alrededor de eventos generados por el usuario (como "clicks" del ratón o presión de teclas). Tradicionalmente, los programas dictan la secuencia que el usuario debe seguir para lograr los propósitos del programa, sin embargo en Windows (y en otras interfaces gráficas de usuario, como Apple Macintosh u OS/2 Presentation Manager) los programas le permiten al usuario decidir los pasos requeridos para completar una tarea, ver **figura 1**.

Windows es un sistema operativo multitarea en el que los programas no son interrumpidos para ejecutar otros, sino que ellos mismos se interrumpen para permitir que otros programas se puedan ejecutar. El «distribuidor de tareas» de Windows se integra por un sistema de transmisión de mensajes y es gracias a los mensajes que los programas pueden recibir información proveniente de los usuarios y de las funciones de la interface de usuario. Para el programador, un mensaje es la notificación de la ocurrencia de un evento que puede necesitar de alguna acción específica.

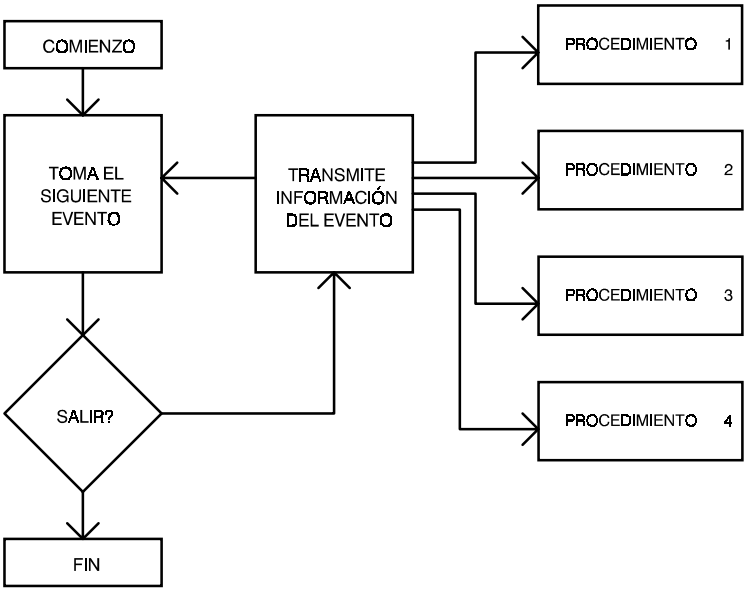


Figura 1. Programa manejado por eventos.

Manejo de los Recursos de Windows

Algunos de los recursos de Windows que pueden utilizar sus aplicaciones son:

- Contextos de dispositivo
- Pinceles
- Plumas
- Fuentes o tipos
- Cajas de diálogo
- Iconos

Windows maneja una interface muy compleja. La interacción entre las aplicaciones del usuario y Windows se efectúa mediante un grupo de más de 500 llamadas a función que en conjunto se denomina API ("Application Program Interface", interface de programas de aplicación). Al plantear el uso del lenguaje C, se producen programas Windows ilegibles, de difícil mantenimiento y depuración. Hay otras áreas que ofrecen dificultades especiales a los diseñadores de aplicaciones, como son el uso de la memoria, las bibliotecas de enlace dinámico (DLL), programas de interface con múltiples documentos (MDI) así como fuentes e impresoras.

Borland vino en ayuda del programador Windows C++ mediante un nuevo marco de aplicaciones denominado Biblioteca ObjectWindows (OWL). OWL sería a una aplicación Windows, lo que TurboVision es a una aplicación DOS. Con ayuda de la OWL se facilita considerablemente el proceso de escribir los programas Windows y se evita que aparezcan diversos detalles de bajo nivel en el código de aplicaciones.

Anatomía de una Aplicación OWL

Es necesario que todo programa escrito con ayuda de OWL soporte al menos dos clases de objetos: un objeto de aplicación y un objeto de ventana. La aplicación es un marco de trabajo complejo encargado de administrar todos los objetos del programa, entre otros: menús, ventanas descendientes y cajas de diálogo. La parte medular de toda aplicación OWL es su programa principal. El siguiente listado es el programa principal de la aplicación descrita en este artículo.

```
# include <owl.h>

int OwlMain(int,char **)
{
  TetrisApp  App;
  return  App.Run();
}
```

TetrisApp la principal clase de aplicación que tiene a su cargo el control de todas las ventanas, las cajas de diálogo y otros objetos de la aplicación. Todo programa OWL debe construirse con base en una clase de aplicación derivada de la clase *TApplication*, que es utilizada por todas las aplicaciones Windows para inicializar, crear la ventana principal y pedir al sistema que le envíe mensajes. El programa de aplicación interactúa con el proceso de inicialización a través de la función

TetrisApp::InitMainWindow()

que a su vez crea la instancia de una ventana de usuario asignada dinámicamente.

Con la ayuda de *TWindow*, el programa soporta las características básicas de ventana: registro, desplazamiento y modificaciones de tamaño. Los «procedimientos de ventana» son funciones contenidas en los objetos de la ventana de base TetrisWindow. Tales procedimientos son utilizados como respuesta a los mensajes de Windows.

El manejo de las cajas de diálogo es un procedimiento que consta de dos pasos: primero se usa el programa *Resource Workshop* para crear la ventana de diálogo y todos sus controles; después se escribe una clase derivada de *TDialog* para manejar los comandos de diálogo y los controles.

Descripción de la forma en que la Aplicación Opera Bajo Windows

Como ya se mencionó anteriormente, la apariencia de la ventana está determinada por las funciones miembro de la clase derivada de la clase *Tapplication*, y el manejo e interpretación de los recursos y el procesamiento de las funciones que entregan el resultado de la aplicación en sí están determinadas por las funciones miembro de la clase derivada de la clase *Twindow* (o cualquiera de la opciones disponibles).

En esta sección del artículo se describirá la forma de utilizar a la clase derivada de *TWindow* para implementar un juego, el ya conocido Tetris. Comenzaremos por mencionar que las funciones propias del juego se incluyen como funciones publicas de *TetrisWindow*, clase derivada de *TWindow*; esta clase contiene además funciones para el procesamiento de eventos como lo son el temporizador de Windows, el presionar una tecla, el utilizar opciones de los menús, etc. y que a su vez está relacionada con una tabla de definición de respuestas que ligará los comandos con las definiciones de función del archivo de recursos.

Cabe mencionar que este tipo de aplicación debe hacer uso del temporizador de windows, por lo que está declarado su uso tanto en la clase *TetrisWindow* como en la tabla de respuestas; la razón de hacerlo obedece a que el juego debe tener la misma velocidad aparente al ejecutar el programa en computadoras con diferente frecuencia de operación del microprocesador. Cuando se hace esto en ambiente DOS, simplemente se usan funciones de retardo en un bucle de procesamiento. En windows esto no es posible porque al generarse un bucle se interrumpe la búsqueda de los mensajes propios del ambiente Windows, y si el bucle es infinito, se pierde totalmente el control del sistema hasta que se destruye el bucle y se finaliza la función que se estaba procesado. Es por lo anterior que el proceso de mover las piezas en forma descendente se hace a través de una función que se llama constantemente a intervalos definidos de tiempo. Para lograr el propósito, se define una función *EvTimer* que es la encargada de controlar la velocidad de procesamiento del juego y que tiene acceso directo al temporizador.

También se ha definido en las clases una función *EvKeydown* que detecta el mensaje generado por la opresión de una tecla y que permitirá detectar la función que el usuario desea que se realice en el momento, a saber permitirá rotar, desplazar a la izquierda o derecha una figura o bien suspender momentáneamente el juego.

El juego Tetris se basa en ordenar figuras que van descendiendo en un área determinada y completar líneas para desaparecer y permitir seguir jugando, en caso de llenar el área al tope el juego finaliza. Es por esto que en la clase *TetrisWindow* se incluyen funciones para las siguientes actividades: dibujar la figura actual, dibujar el ambiente de juego, establecer la puntuación, desplazar la figura actual, comprobar el momento en que una pieza ya no pude descender, posicionamiento de la pieza, y detección de línea completa principalmente. El juego basa su operación en simular en pantalla lo que ocurre en una matriz m x n, en donde una localidad vacía contiene un cero y una llena contiene un valor de uno, lo que facilita el hecho de revisión del estado del juego, y permite realizar el proceso de detección de límites y topes para los desplazamientos de las figuras.

El proceso de dibujar una nueva figura consiste en copiar una imagen que se ha representado en forma de una matriz, y que se selecciona aleatoriamente en base a la función rand() del lenguaje C. El número de figuras es fácilmente ampliado con mínimas modificaciones al listado del programa.

Para simular el movimiento de una figura en el ambiente gráfico utilizamos las funciones getimage y putimage. En ambiente Windows se utiliza la función *BitBlt* para lograr lo mismo que las dos anteriores; para mostrar su uso en el programa, después del uso de *BitBlt* se pone como comentario la forma en que se utilizaría getimage y putimage ilustrando la similitud. El uso de esta función solo requiere entender lo que se denomina un «contexto» en la programación para ambiente Windows, podemos decir simplemente que es el medio en el cual y para el cual se realiza un proceso, en este caso tenemos los dos principales que son la memoria de trabajo y el despliegue.

Por último, mencionaremos que la compilación del programa se realiza tomando en cuenta al menos tres archivos: el archivo fuente con extensión CPP, el archivo de recursos con extensión RC y un archivo de definiciones con extensión DEF, los cuales se engloban en el archivo de proyecto con extensión IDE

La **figura 2** muestra el aspecto visual del programa.

Conclusiones

Uno de los aspectos más importantes que debe tener en cuenta el programador al utilizar programación orientada a objetos usando OWL, es que se requieren mayores recursos de cómputo para compilar los programas que con la programación tradicional, esto debido a que se busca

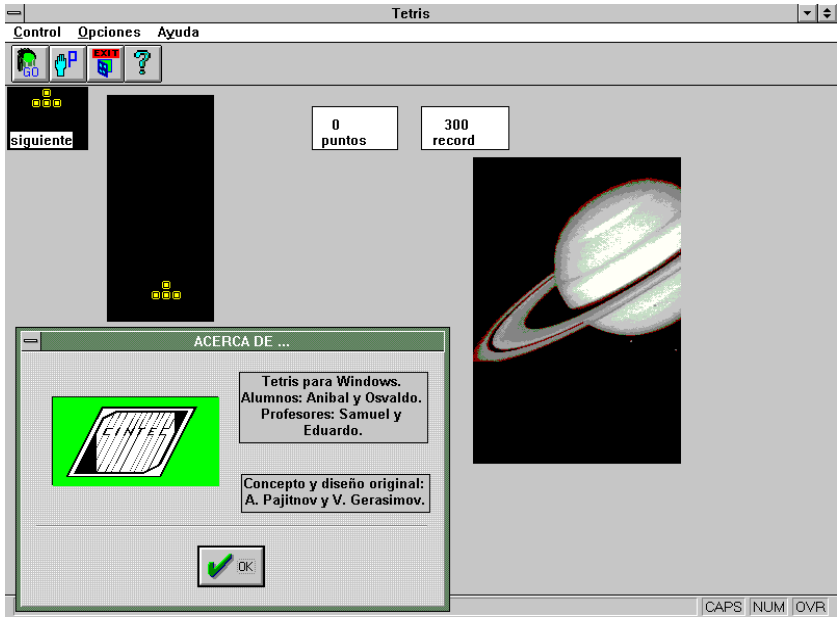


Figura 2. Aspecto visual del programa

información en muchas bibliotecas, y el trabajo de cómputo es intenso, aunque al programador le resulte más fácil escribir programas bajo este esquema, ya que con pocas líneas de código fuente es posible crear ventanas vistosas.

Un hecho indiscutible es el siguiente: el programa ejecutable producido por compilar usando OWL es bastante más grande en tamaño que el que provee la programación tradicional; es fácil deducir que se incluyen componentes muchas veces redundantes de código no optimizado. Es responsabilidad del programador elegir la ruta más conveniente para crear sus aplicaciones teniendo como base el panorama anterior.

Finalmente, el código correspondiente de este ejemplo se encuentra disponible en el CINTEC, para cualquier persona que lo requiera.

Bibliografía

[1] Nabajyoti Barkakati. *"C Programmer's guide with Borland C++ 4.0"*. Editorial SAMS

[2] Herbert Schildt. *"Turbo C ++: Manual del usuario"*. Editorial McGraw Hill.

[3] Borland. *"Object Windows Library Reference"*. Editorial Borland.

Generación de un Programa Para Realizar Diagramas de Flujo Mediante la Técnica de POO

Ing. Ignacio Raúl Rosas Román
Alumno de la Maestría del CINTEC-IPN.

Desde su aparición, la programación estructurada ha proporcionado a los programadores una metodología de diseño consistente que facilita la elaboración de un programa. Esta metodología se basa en la descomposición de un problema en piezas pequeñas que son más manejables que el problema original. A estas piezas que resultan de la primera división se les denomina "subtareas". El siguiente paso bajo esta metodología es ir descomponiendo cada una de estas subtareas, hasta llegar al punto donde se pueda expresar cada subtarea en comandos del lenguaje de programación que se está utilizando. De lo anterior, se puede observar que la metodología se centra en las acciones que el programa realizará sobre los datos.

Aun cuando los lenguajes de programación estructurada reconocen como objetivo final la reutilización de código, hasta ahora no ha sido posible obtener un programa fácil de actualizarse o modificarse con facilidad.

Por otra parte, la programación orientada a objetos fue uno de los primeros sistemas de programación en que se reconoció que el cambio y la evolución de un programa no sólo son inevitables, sino también deseables. La programación orientada a objetos busca minimizar el impacto del cambio a través de la técnica del encapsulado, la cual se basa en el agrupamiento de procedimientos y datos asociados de un objeto en un solo conjunto.

La idea general de la programación bajo esta metodología se basa en estructurar los programas alrededor de los datos, asociándoles íntimamente los tratamientos que les son específicos, en lugar de hacerlo sobre las funciones, las cuales son modificadas mas frecuentemente.

Desarrollo del programa.

El compilador que se utilizó para generar el programa, es el C++ versión 3.1 de Borland. Este compilador utiliza como plantilla para la construcción de objetos la estructura **class**, dentro de la cual es posible definir datos y funciones. Las funciones de una clase se llaman funciones miembro y son las encargadas de manipular los datos pertenecientes a la clase, así como de definir el acceso que se tiene a ellas.

Para definir las clases que se utilizarán, es necesario analizar primero el problema. Los diagramas de flujo son dibujos bidimensionales, por lo cual todos los dibujos pueden ser definidos dentro de un espacio rectangular. Este espacio rectangular dentro de un sistema de ejes cartesianos, puede ser caracterizado por cuatro variables que definen dos parejas de datos (x, y) distintas dentro del plano. Por esta razón, todos los objetos a dibujar dentro del programa se representarán mediante 5 variables, 4 de ellas están encaminadas a definir el área dentro de la cual se dibujará el objeto y una más para definir el tipo de objeto que se está dibujando. Con esto se logra que los datos que definen los objetos del dibujo sean solamente 5 enteros, en lugar de guardar el mapa de bits asociado a cada objeto. Esta estrategia resulta en un manejo de memoria más eficiente. La declaración de este objeto se encuentra en el archivo FIGURAS.H y se presenta a continuación.

```
class Base
{
 protected:
 int X1, X, Y1, Y, linea;
 public:
 void asigna(int, int, int, int, int);
};
```

Como se observa, la única función miembro dentro de este objeto, es la función *asigna*, la cual tiene como característica ser **pública**, lo cual nos permite utilizar dicha función en caso de que la clase se utilice como bloque de construcción de otro objeto. La tarea que realiza esta función miembro es actualizar las variables que definen al objeto. La importancia de esta clase es fundamental, ya que a partir de ella se definen las clases encargadas de dibujar los objetos en la pantalla y se genera una lista en la cual se almacenan los objetos.

Lo anterior se hace mediante el uso de **clases derivadas**, que tienen la característica de **heredar** las propiedades de la clase a partir de la cual fueron construidas.

En términos generales, la herencia tiene el objetivo de compartir y/o sacar puntos comunes entre los diferentes subconjuntos de problemas. El principio es describir un nuevo objeto sin que se tenga que partir de cero, sino por el contrario, por extensión o especialización de uno o varios objetos que ya existen. A partir de lo anterior, es evidente que las características de herencia de la programación orientada a objetos facilitan la reutilización del código o bien su transición hacia características más complicadas (evolución).

Una clase derivada que se utiliza dentro del programa es la que se describe a continuación:

```
class Elementos: public Base {
 int tipo;
 char *cadena;
 public:
 void entra_datos(int, int, int, int, int, int, char*);
 void sale_datos(int &, int &, int &, int &, int &, int
&, char *&);
 int busca_elemento(int, int);
};
```

La clase llamada "Elementos", tiene como clase básica para su construcción a la clase "Base". Dentro de "Elementos" se definen variables adicionales para describir características que dan la diferencia entre dos objetos similares, como son el ancho de línea o el tipo de carácter a utilizar (variable tipo) y, por otra parte, una dirección que indica donde está la cadena de caracteres asociada a un mensaje. Adicionalmente a los métodos de la clase base, se tienen métodos para actualizar los datos (función entra_datos), leer los datos (función sale_datos) y una función que busca si un punto definido por un par (X, Y) está dentro del área que define un objeto.

Las clases "figuras", también son clases derivadas a partir de la clase base. Esta clase solamente define una función miembro adicional que se encarga de dibujar una figura específica dentro de los límites definidos por los dos puntos de la clase "Base".

Por otra parte, es necesario definir un objeto que permita la interacción del usuario con el programa. Dicha interacción se realizará básicamente a partir de la clase MOUSE, que se encarga de recibir los eventos que provienen de este dispositivo. El uso del programa se hace casi enteramente a través del Ratón, a excepción de los mensajes para el texto. Es por esta razón que aparecerá un mensaje indicando la falta de este dispositivo en caso de no estar instalado en el equipo y el acceso al programa será negado.

La última clase que compone al programa, es la clase "Ambiente", la cual se encarga de realizar todas las tareas asociadas con la edición y manejo de los objetos gráficos definidos anteriormente. Los datos que maneja esta clase son principalmente instancias de la clase "Elementos" y variables que definen características diversas del área de trabajo que se utiliza (monitor). El manejo del monitor utiliza una resolución de 640 X 480 pixeles bajo el modo de video VGA en color o blanco y negro.

Estructura del programa.

El primer procedimiento que se ejecuta es la función "dibuja_pantalla" de la clase ambiente, que es la encargada de comprobar la existencia del Ratón, así como de inicializar el modo de video y dibujar el ambiente de trabajo para el usuario, el cual se puede observar en la **figura 1**.

El siguiente paso consiste en el uso de la función "maneja _icono", que es la encargada de detectar los mensajes provenientes del mouse para verificar si el usuario desea activar alguna de las funciones disponi-

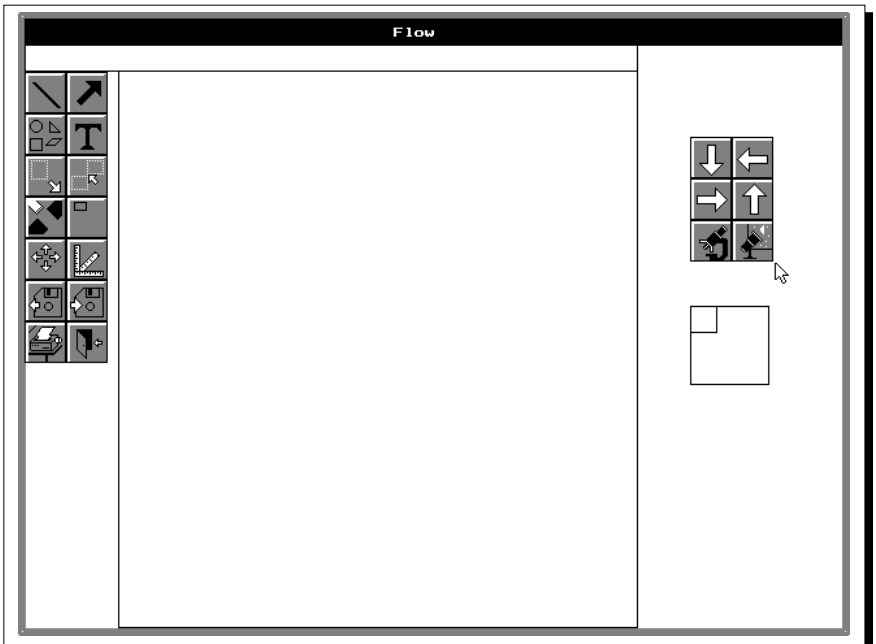


Figura 1. Ambiente de trabajo para el usuario

bles en el programa. Para hacer esto, simplemente se posiciona el apuntador del mouse sobre el ícono de la función deseada y se presiona el botón izquierdo.

Las funciones miembro de la clase ambiente pueden ser agrupadas en tres tipos:

1) Funciones que dibujan los objetos.
2) Funciones que editan los objetos en la pantalla.
3) Funciones que escriben o leen de archivos los diagramas generados en el paquete.

El primer tipo de funciones tiene la característica de presentar varias opciones de dibujo al usuario, las cuales aparecen en el lado derecho de la pantalla (ver **figura 2**). Estas opciones se pueden escoger presionando el botón izquierdo del ratón sobre ellas. Una vez hecho esto, la figura seleccionada aparecerá en video invertido para indicar cuál fue seleccionada. Para proceder con el dibujo, primero se marca un cuadro con el ratón, lo cual se realiza de la siguiente forma: primero se presiona el botón izquierdo en un punto y, sin soltar el botón, se desplaza el dispoditivo para formar un rectángulo; hecho esto se debe soltar el botón. Cuando se suelta el botón, la figura se dibuja automáticamente, y se puede volver a dibujar otra figura. En caso de que se desee seleccionar otra figura, se presiona el botón derecho del mouse y la figura seleccionada que aparecía en video inverso, vuelve a su estado original; llegado a este punto, se puede seguir el procedimiento descrito anteriormente para seleccionar otra figura y dibujarla, o bien presionar el botón derecho del mouse una vez más para salir de esa función.

Las funciones que operan de esta forma son:

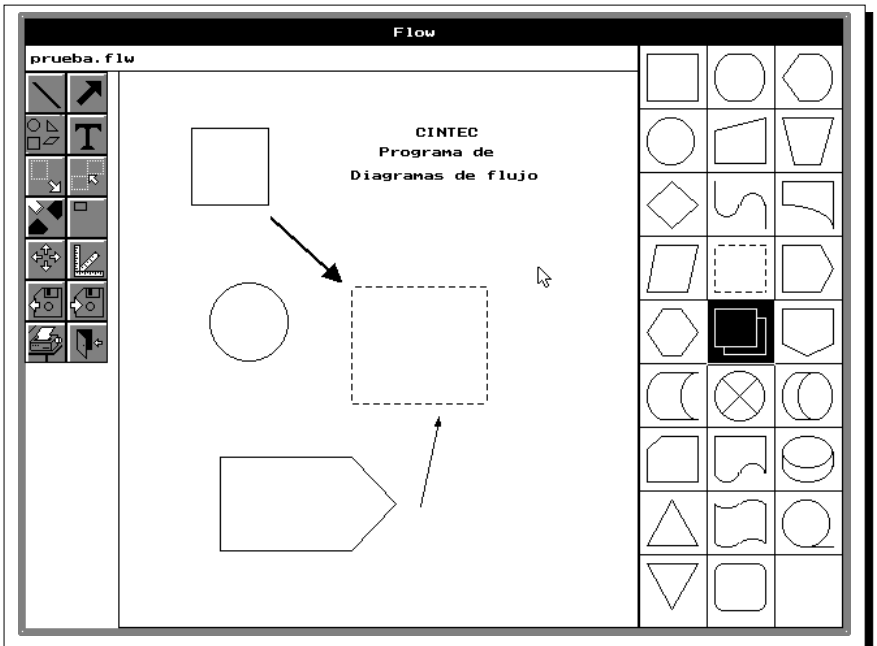


Figura 2. Opciones de dibujo del programa (lado derecho)

1) Dibuja_figuras.
2) Dibuja_lineas.
3) Dibuja_texto.
4) Dibuja_flechas.

El segundo grupo de funciones realizan la tarea de editar los objetos que fueron dibujados con el primer grupo de funciones. Las opciones de edición son: Copiar, Borrar, Mover y escalar objetos. Para su uso, primero se debe activar la opción de selección, una vez activada la opción simplemente se presiona el botón izquierdo del ratón para seleccionarlo, y dicho objeto aparecerá resaltado para indicar que fue seleccionado. Para salir de la opción de selección se presiona el botón derecho. Los objetos seleccionados por el método anterior pueden ser copiados, movidos o borrados, dependiendo de la función utilizada.

Las funciones que realizan las tareas anteriores son:

1) Selecciona_figuras.
2) Escalar_objetos.
3) Mover_objetos.
4) Copiar_objetos.
5) Borrar_objetos.

Por último se tienen las funciones del grupo 3, que son las encargadas de guardar en disco la información y de imprimir el dibujo. La rutina que maneja la impresión del dibujo está hecha para un dispositivo de impresión EPSON FX-80 o compatible.

Las funciones que realizan esta tarea son:

1) Salva_dibujo.
2) Carga_dibujo.
3) Imprime_dibujo.

Nota: El código fuente completo de esta aplicación se encuentra disponible en el CINTEC para cualquier persona que lo requiera.

Bibliografía

[1] Greg Voss. *"Programación orientada a objetos: Una introducción"*. Ed. McGraw Hill México D.F, 1994.

[2] Loren Heini. *"Power graphics using Turbo C"*. Ed. Adison Wesley.

[3] Clayton Walnum. *"Borland C++ power programming"*. Editorial QUE.

[4] Bjarne Stroustrup. *"The design and evolution of C++"*. Ed. Adison Wesley.

[5] Star Micronics 1000 User´s Guide.

OBSERVACIONES

Si te interesa contribuir con algún comentario o sugerencia, el Consejo Editorial de la Revista Polibits agradecerá utilices este espacio.

Y lo remitas a :

Centro de Investigación Tecnológica en Computación
Consejo Editorial Revista POLIBITS
Av. Té No. 950, Edificio de Graduados 2o. piso.
Colonia Granjas México. C.P. 08400
México D. F.
☎ 6-54-39-32, 6-49-50-36, 6-49-03-66 ext. 327. Fax : 6-57-74-53.
FAX 6-54-39-32 y 6-49-50-36 **e-mail** *cintec@redipn.ipn.mx*
Si tu contribución es en forma de un artículo, favor de anexar los siguientes datos:
Nombre del autor, Grado Académico e Institución de procedencia.