18

UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
CAMPUS ARAGON

FUNCIONES BASICAS PROGRAMADAS EN VISUAL BASIC PARA DISEÑAR PROGRAMAS EDUCATIVOS Y JUEGOS DE VIDEO.

T E S I S

QUE PARA OBTENER EL TITULO DE:
INGENIERO EN COMPUTACIÓN
P R E S E N T A :

ELIAS GARCIA SANTILLAN

DIRECTOR DE TESIS :
LIC. ISRAEL JUÁREZ ORTEGA.

San Juan de Aragón, Edo.de Méx. 2002

DEDICATORIA

A mis padres:

Que gracias a su apoyo logro culminar una etapa muy importante en mi vida.

A mi esposa e hija:

Por motivarme a continuar siempre adelante y lograr todas mis metas.

Por su cariño, su amor y comprensión.

A todos mis compañeros y amigos:

Por dedicarme su tiempo en asesorías, por prestarme sus instalaciones, equipo y por todo el apoyo recibido.

ÍNDICE

INTRODUCCIÓN

Con el gran auge tecnológico, el desarrollo de nuevos lenguajes de programación y la utilización de multimedia, se han desarrollado en el mundo simuladores de todo tipo y programas de realidad virtual que interactúan con el sujeto y lo acercan cada vez más a la realidad.

Un recurso muy importante en el proceso de Enseñanza-Aprendizaje en forma significativa del niño o del adolescente son los juegos y programas interactivos, ya que les llama mucho la atención, disfrutan de aprender jugando, aumenta su capacidad de recepción, su razonamiento, ejercitan la memoria y la adquisición del conocimiento es mucho mayor, comparado con cualquier otra fuente de información.

Existen empresas especiales para diseñar programas y juegos interactivos con todos lo nuevo en sistemas de audio y vídeo, como: *MICROSOFT, SEGA, NINTENDO, GENESIS, NEOGEO,* entre otras.

En el diseño de imágenes y la programación en lenguajes sofisticados, se requiere de muchas personas especializadas: *programadores, técnicos en sonido, diseñadores, investigadores, científicos, etc.* así como de gran equipo de cómputo, de fotografía y de instalaciones adecuadas. El presente trabajo tiene la finalidad de aportar algunas funciones básicas con las que se desarroilan juegos de video y programas educativos en 2D por medio del lenguaje de programación Visual Basic y utilizando algunos paquetes de edición de sonido, audio y video bajo ambiente Windows.

El uso de estas funciones y herramientas no esta limitado: se pueden utilizar para realizar retoques fotográficos, presentaciones publicitarias, hacer grabaciones de audio, efectos de sonido, presentación de videos, crear enciclopedias, manejar bases de datos y en general para todo tipo de programas que se utilizan en las empresas o instituciones.

REQUERIMIENTOS MÍNIMOS

Para poder ejecutar los programas del CD necesita lo siguiente:

HARDWARE

- Computadora con microprocesador Pentium I o posterior
- Tarjeta de sonido de 16 bits o de mayor calidad de audio
- Unidad de CD
- Memoria RAM 12 MegaBytes
- 200 MegaBytes de espacio disponibles en disco duro
- Monitor VGA o Superior (Configurado al modo 800x600)

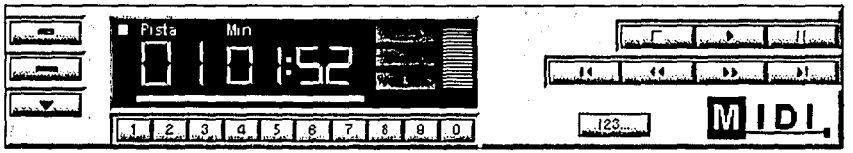
SOFTWARE

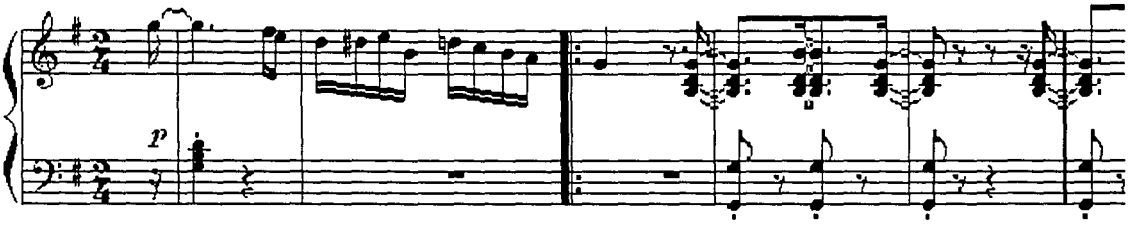
- Windows 98 o posterior
- Visual Basic versión 5.0 o posterior (opcional)
- CoolEdit profesional o posterior (opcional)
- Adobe Photoshop 5.5. o posterior (opcional)

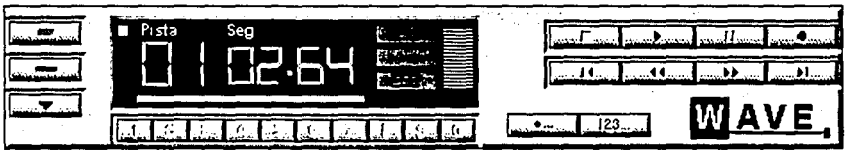
CAPÍTULO

1

Edición, grabación y efectos de sonido con "CoolEdit Pro"







El sonido es parte fundamental en el desarrollo de cualquier programa o juegos, se utiliza principalmente para llamar la atención y crear un ambiente agradable.

En este capítulo abordaremos algunos aspectos y características importantes sobre la calidad de audio, la codificación y decodificación del sonido y los nuevos sistemas de grabación digital.

La edición de archivos de audio y grabación desde fuentes externas como el micrófono, la radio, casetes, CD de audio o la televisión, lo llevaremos a cabo por medio de un programa muy fácil de utilizar llamado "***Cool Edit***"; además con este programa también se pueden crear efectos de sonido, mezclas, arreglar la calidad del audio, unir segmentos de archivos, aumentar o disminuir volumen, entre otras cosas.

DIGITALIZACIÓN Y REPRODUCCIÓN DEL SONIDO

El sonido es una vibración que se propaga a través del aire y cuando llegan a nuestros oídos, este las trasforma en señales eléctricas que pueden ser entendidas por nuestro cerebro. Un micrófono actúa de manera similar trasformando las señales acústicas en eléctricas de manera que puedan guardarse.

Cuando un sonido se graba en un formato digital, no se almacena de forma continua, como en los medios analógicos sino que se almacenan una serie de muestras a pequeños intervalos de tiempo por medio de un convertidor analógico-digital (ADC), que transforma la tensión recibida por cada canal de la línea de entrada en un valor numérico.

El número de muestras tomadas en un periodo de tiempo prefijado es conocido como "*frecuencia de muestreo*" y varía entre 4 y 48 KHz.

La cantidad de datos grabados en cada muestra se llama "*amplitud*". Cuanto mayor sea la amplitud, mayor detalle se almacena, y por tanto se consigue mayor fidelidad al sonido original en la reproducción. Por ejemplo, 32 bits definen un sonido con más precisión de lo que hacen 16 bits pero los archivos ocupan un mayor espacio en disco duro.

La reproducción se lleva a cabo por medio de un convertidor digital-analógico (DAC) que viene integrado en la tarjeta de sonido y produce una determinada tensión audible para cada valor de la muestra y la mantiene hasta encontrar la siguiente muestra como se muestra en la figura 1.1.0.

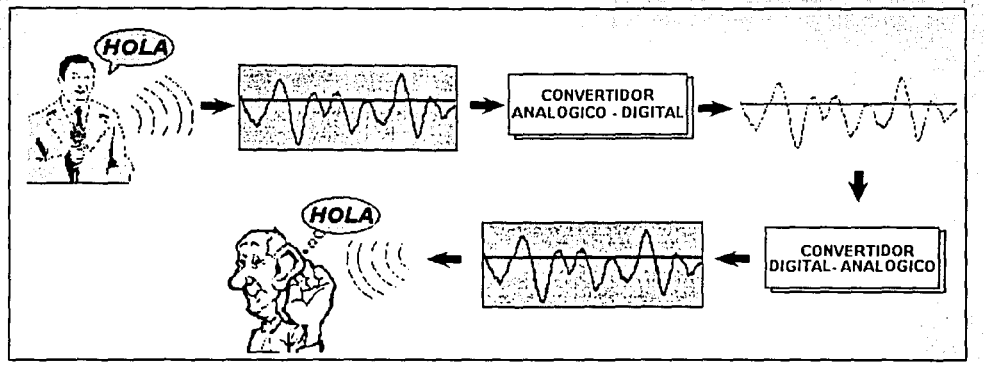


Figura 1.1.0. Codificación y decodificación del sonido.

CALIDAD DEL SONIDO

La calidad del sonido que percibe el ser humano depende de la señal emitida por la fuente, además de factores físicos: unos directos, como la posición y la orientación relativa de la fuente y el oyente, otros indirectos, como la geometría del espacio circundante a la fuente, la señal se percibe de distinta manera en una iglesia y en un estadio de fútbol.

El sonido estéreo es un sistema de *grabación/reproducción* que utiliza una fuente codificada en dos canales (*figura 1.1.1.*). A pesar de estar disponible desde hace más de 38 años aún tiene gran aceptación.

De acuerdo con las posibilidades técnicas actuales y los recientes desarrollos en el campo acústico, el sistema de sonido estéreo se muestra insuficiente y obsoleto comparado con las nuevas técnicas de generación de sonido tridimensional que producen una mayor fidelidad.

El oído humano percibe el sonido en tres dimensiones. Posee dos canales de recepción independientes, y la recomposición del espacio acústico tridimensional se realiza en distintas partes de la corteza cerebral.

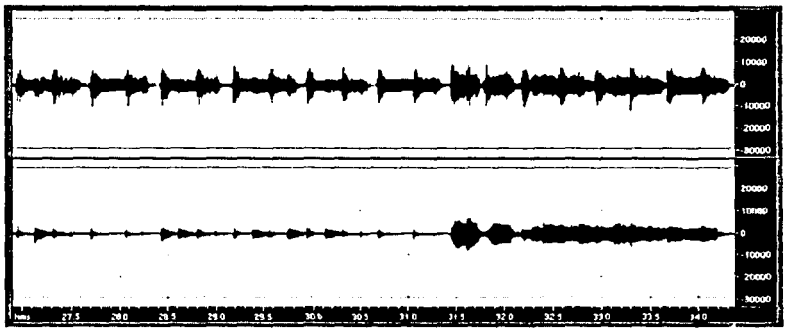


Figura 1.1.1. El sonido estéreo.

TÉCNICAS DE GENERACIÓN DE SONIDO TRIDIMENSIONAL

Existen técnicas que realzan el sonido cambiando la fuente de sonido original, introduciendo retardos, desplazamientos de fase o distorsiones armónicas (*codificación previa*) y que no necesitan un método específico de decodificación, se reproducen mediante los dos canales estéreo tradicionales. El exponente más claro de este sistema es el *Qsound.*

El sistema QSound permite posicionar distintos sonidos con cierta precisión dentro de un arco de 180° por delante del oyente.

Para conseguir este efecto de localización múltiple se codifica el sonido utilizando bocinas estéreo estándar.

Las tarjetas de sonido Sound *Blaster 32 AWE/Value* están especialmente preparadas para reproducir sonido Qsound, y los resultados de la reproducción de estos sonidos preprocesados QSound 3D son de muy buena calidad.

Utilizando una codificación previa y decodificadores especiales para reproducir el sonido por dos o más canales de salida se logra una excelente calidad en sonido. Se trata de los sofisticados sistemas *Dolby Stereo* que se utilizan en las más modernas salas comerciales de cine.

Desde finales de los años 70 se usan técnicas de grabación y reproducción del sonido en el cine. La mayor parte de los sonidos de la banda sonora de una película no se registran durante el rodaje de la misma, sino durante el proceso de postproducción. En esta etapa, los diálogos, los efectos, el ambiente y la música se registran por separado y se codifican usando técnicas especiales en dos únicos canales de sonido. Este conjunto de técnicas de codificación reciben el nombre de codificación Dolby Stereo.

Para reproducir el sonido original se debe usar uno de los tres tipos de decodificadores disponibles: *Dolby Pro Logic*
Los decodificadores Dolby Pro Logic ofrecen una mayor separación percibida entre los canales de audio y una mayor localización de sonido. En este caso, a partir de la fuente estéreo original se extraen cuatro señales, un canal para cada bocina frontal (*derecha e izquierda)*, otro para las bocinas traseras (*envolventes*), y el cuarto canal para la bocina central (*Ver figura 1.1.2.*). El canal central y el envolvente se derivan de ciertas partes de los canales derecho e Izquierdo.

Figura 1.1.2.- Reproducción del sistema Dolby Stereo mediante Dolby Pro Logic

FORMATO DE SONIDO DIGITAL WAV

Este tipo de formato es de los más utilizados en las PCs y en Internet. Dentro de la RED se utilizan 8 bits a 11,024 Hz. Se pueden hacer grabaciones tipo CD a 44,000 Hz y es de más alta calidad ya sea monoaural o estéreo (*Ver tabla 1*). Este tipo de formato fue desarrollado por Microsoft para utilizarse en ambiente Windows pero existen reproductores de archivos *Wav* para casi todas las plataformas existentes.

Un archivo de sonido sin comprimir puede ocupar mucho espacio en su disco duro. Por ejemplo, un minuto de sonido estéreo grabado con una resolución de 44 KHz ocupará aproximadamente 10.5 *MegaBytes*.

La grabación de archivos de sonido en formato ADPCM* reduce considerablemente el tamaño de los archivos.

La grabación en monoaural requiere menos espacio para su almacenamiento, pero solo utiliza un canal de grabación. Si requiere de mejor calidad de audio para alguna presentación es necesario grabar el archivo en estéreo.

Con 8 bits de resolución, nuestro archivo tendrá 256 niveles de volumen, lo cual es muy superior a lo que una bocina de computadora pueda reproducir.

Se pueden grabar archivos de audio en formato en WAV con una resolución de 16 bits (*65,536 niveles de volumen*) y después grabarlos en un CD por medio de una grabadora de CDs y reproducirlos en un estéreo o reproductor de CDs para música.

8,000 Hz	*Teléfono*
11,000 Hz	*Radio AM*
22,050 Hz	*Radio FM*
32,075 Hz	*Mucho mejor que la calidad de FM*
44,100 Hz	*CD de sonido*
48,000 Hz	*DAT (Digital Audio Tape)*

Tabla 1. Diferentes Calidades de Grabación

* Modulación Diferencial Adaptiva por Código de Pulsos, una técnica de compresión de datos de Microsoft.

SOFTWARE DE EDICIÓN DE SONIDO

Existen editores y reproductores de sonido disponibles en el mercado como el SoundEdit de Macromedia, Wave Studio, CoolEdit, Pro Audio, Q3D, SoftEncode Dolby Digital, Sound Forge, Mp3toCD, CakeWalk, entre otros.

Además de poder grabar y reproducir archivos de audio en distintos formatos de audio, estos programas tienen opciones muy importantes para diseño como efectos de sonido, mezcladora de sonidos y muchas otras herramientas de gran importancia.

Por medio de la línea de entrada de la tarjeta de sonido se pueden realizar mezclas desde diferentes medios (*televisión, la radio, cassettes, micrófono etc.*); o directamente desde archivos MIDI* o WAV.

Algunos de estos programas se encuentran disponibles en la red y se pueden bajar de sitios como:

http://www.syntrillium.com/cooledit/
http://www.adobe.com/products/photoshop/
http://www.qsound.com
http://www.microsoft.com/direcx/download.asp
http://www.winamp.com

Para la edición, reproducción y grabación de archivos con formato *WAV* únicamente nos enfocaremos en el ***CoolEdit Pro***.

* Interfase Digital para Instrumentos Musicales. Un estándar internacional de Hardware y Software que permite que diversos dispositivos, instrumentos y computadoras intercambien códigos y sucesos musicales.

CoolEdit Pro

Es una herramienta de edición de sonidos. Con ella puede grabar, reproducir y editar archivos de ondas de 8 bits (*calidad de cinta*) y de 16 bits (*calidad de CD*) y hasta 32 bits. También puede reproducir y editar otros formatos, como archivos de 4 bits, 2 bits y 1 bit.

Puedes bajar o actualizar tu versión por medio de la red en la siguiente dirección: http://www.syntrillium.com/cooledit/

CoolEdit Pro, puede grabar sonidos de diversas fuentes para luego manipularlos a su gusto, cortar y pegar segmentos de archivos de ondas (*WAV*) para combinar sonidos y añadir efectos especiales como reverberación, eco y desvanecimiento, entre otros.

También reproducir un fragmento musical de un CD, un mensaje hablado por el micrófono y juntarlos en un archivo para utilizarlo en una presentación audiovisual.

Cool Edit Pro nos proporciona varias opciones que podemos seleccionar desde la barra de menú.

El menú "*File*" contienen los comandos básicos para el manejo de archivos como: *New*(*Nos crea un archivo nuevo*), *Open*(*Abre archivos ya grabados en el disco*), *Close* (*Para cerrar nuestros archivos*), y también nos muestra una lista de los archivos utilizados anteriormente, para su rápida localización.

En el menú "Edit" encontramos la opción *Undo*(*para deshacer la última modificación que se realizo*), *Cut*(*Corta el fragmento de la señal de audio*), *Copy*(*Copia en memoria una porción de señal*), *Paste*(*Pega la porción de señal que se almacenó en la memoria mediante copy*).

Debajo de la barra de menú encontramos una serie de botones con las funciones más frecuentes que selecciona en forma directa los comandos y facilitan el trabajo a realizar. Bajo estos botones está representada la onda sonora que representa el sonido o canción que se haya abierto.

En la parte inferior de la pantalla nos encontramos con otros botones que son similares a los que tenemos en cualquier reproductor (play, adelante, atrás, record, etc.). Ver figura 1.2.0.

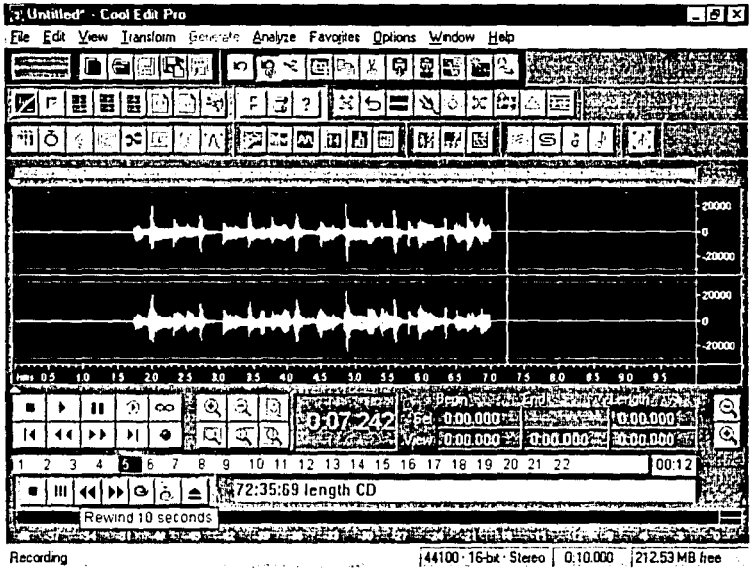


Figura 1.2.0. Pantalla principal de Cool Edit Pro.

GRABACIÓN DESDE UN CD

Para grabar canciones de un CD a un archivo ***"WAV",*** lo primero que necesita es crear un nuevo archivo seleccionando *<New>* del menú *<File>*. Aparece una ventana en donde debe seleccionar la frecuencia, el número de canales y los bits a los que se grabará el nuevo archivo.

El segundo paso es activar las herramientas del "CD" seleccionando la opción **<Show CD Player>** dentro del menú ***<View>*** *y* presionar el botón ***<Record>*** *(Grabar),* en este momento aparece otra ventana donde indicamos el tiempo de grabación. Finalmente debe seleccionar la pista que desea escuchar y presionar el botón de **<Play>.** Todo lo que escucha en las bocinas se esta almacenando en el buffer de la memoria de la computadora hasta que presione el botón de **<Stop>.**

Para almacenar nuestro archivo en el disco duro será necesario seleccionar *<Save>* dentro del menú *<File>* .

Los menús gráficos son parecidos a los símbolos de las grabadoras de video o de los estéreos caseros; además, para evitar confusiones, si desplaza el cursor por encima de un botón del menú aparecerá en un recuadro el nombre de la acción que realiza.

Como nos damos cuenta, CoolEdit tiene la opción de grabar a 48,000 Hz, una calidad superior a la de un CD de música; pero el inconveniente es que entre mayor sea la frecuencia de muestreo, mayor será el tamaño de nuestro archivo (Ver *Figura 2.2.1*).

Además de grabar en formato con extensión "WAV" podemos grabar otros tipos de formatos como: *.VOC, .PCM, .AU, .SND,* entre otras.

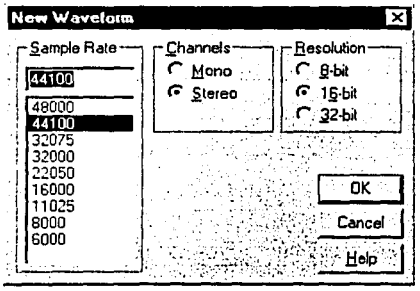


Figura 1.2.1. Calidad de grabación.

CAMBIOS DE VOLUMEN

Si queremos una reducción continua de volumen en una parte del archivo utilizaremos la herramienta <Fade Out> que se encuentra dentro del menú <Amplify>, seleccionando la zona en que deseamos reducir el volumen y aplicar esta herramienta.

En las figuras 1.2.2. y 1.2.3 se aprecia claramente la selección de pistas y como queda el nuevo archivo final.

Es importante que la zona seleccionada sea de al menos de 8 segundos, porque de lo contrario se obtiene un final muy precipitado.

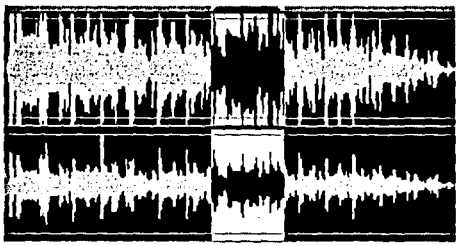


Figura 1.2.2. Selección para utilizar como fin de pista.

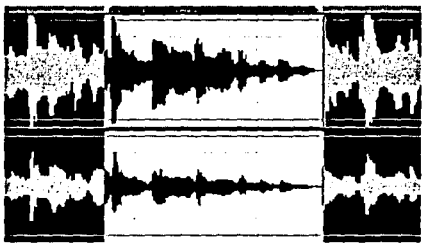


Figura 1.2.3. Después de cortar el final, con Fade Out se obtiene la clásica bajada de volumen que identifica el nuevo final.

MEZCLA DE CANALES

La calidad del sonido puede mejorar con el mezclador de canales <Channel Mixer>, además de permitir una labor creativa empleando los dos canales del sonido estéreo.

Básicamente su mejor utilidad es mejorar la respuesta de una señal monofónica. Las canciones antiguas o todas las ondas procedentes de grabaciones monoaurales pueden pasar por alto a un estéreo simulado.

El primer paso es duplicar el canal monoaural para obtener dos canales, proceso que se realiza copiando y pegando sobre una nueva onda en estéreo o utilizando la conversión del tipo de "*sampleado*" <Convert Sample Type> que se encuentra dentro del menú <Edit> (*Ver figuras 1.2.4, 1.2.5 y 1.2.6*).

Cuando ya se tiene la onda estéreo artificial hay que rebajar muy levemente el volumen y la amplificación de uno de los canales, porque el oído es muy sensible a estos cambios. Se apreciará la diferencia y será más clara la percepción de los dos canales. Ambos canales siempre contendrán la misma información sonora, pero por lo menos se habrá logrado superar las limitantes de una señal monofónica *(figura 1.2.7.)*.

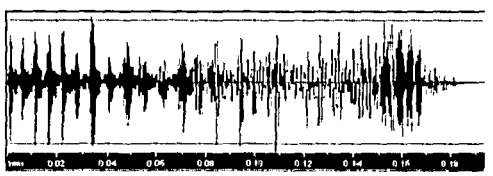


Figura 1.2.4. Señal monoaural grabada a 11,000 Hz. con 8 bits.

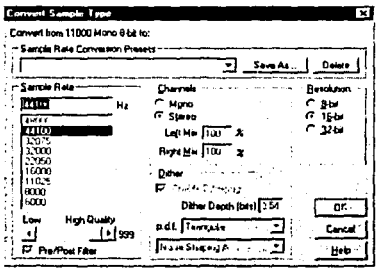


Figura 1.2.5. Utilización del Conver Sample Type.

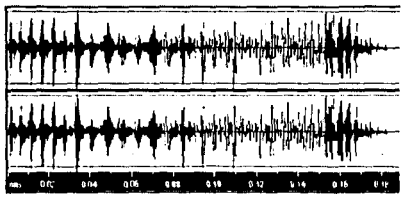


Figura 1.2.6. Señal estéreo resultante sin variación de volumen en ninguno de sus canales.

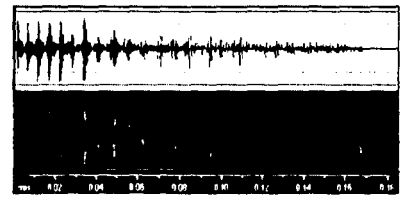


Figura 1.2.7. Señal estéreo resultante con disminución de volumen en el canal izquierdo.

El mezclador de canales permite invertir las pistas, unirlas para obtener un efecto psicodélico de música de los 60 o tomar la señal de ambos canales para centralizar la imagen sonora remarcando un canal central que mejora notablemente la percepción de las voces *(figura 1.2.8.)*.

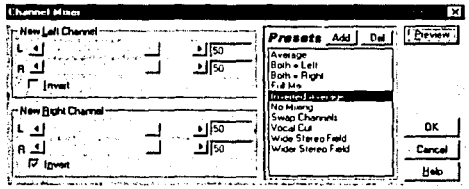


Figura 1.2.8. Mezclador de canales.

Con la opción *"panoramizar"* el sonido de un canal aumenta de izquierda a derecha; mientras el otro canal disminuye para construir un movimiento del sonido. Por ejemplo, se puede simular el ruido del motor de un automóvil desde la derecha hacia la izquierda, lo que dará una impresión de movimiento y espacio dentro del campo sonoro como se muestra en las figuras 1.2.9 y 1.2.10.

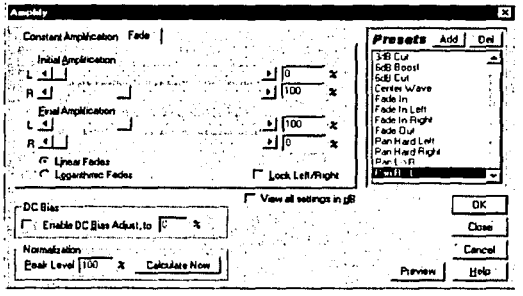


Figura 1.2.9. Opción de Panoramizar L->R.

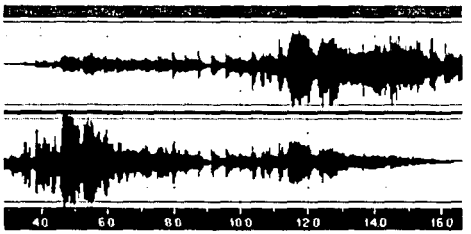


Figura 1.2.10. Disminución y aumento de Volumen de los canales izquierdo y derecho mediante Panoramizar del canal Izquierdo al Derecho.

EFECTOS 3D

Para obtener un efecto envolvente existen diversos sistemas de codificación multicanal que disponen de hasta seis canales para rodear al oyente en un entorno acústico de 360º, pero el precio de las herramientas de software capaces de codificar en Dolby Surround o en Dolby Digital es muy elevado, además del equipo de descodificación. Otra opción es recurrir a sistemas de expansión como el Qsound.

Los efectos 3D se aplican por lo general a **películas**, **juegos**, etc. y algunos músicos como Sting apoyan sistemas como el Qsound.

La enorme calidad del Dolby Digital ya está empezando a entrar en la **industria discográfica**, que planea codificar la música en formato multicanal para ir más allá del sonido estéreo.

Con los efectos de retardo <Delay>, tales como los coros, ecos y retardos simples es posible conseguir un aceptable aumento de la dimensión sonora hacia un canal trasero fantasma.

Los coros <Chorus> aumentan la riqueza de la señal estéreo aclarando el sonido de las voces o de los instrumentos mediante la reflexión del sonido como si la música se estuviera interpretando en una sala con paredes que rebotasen las ondas sonoras.

También sirve para crear estéreo simulado a partir de una onda monoaural. Es recomendable utilizarlo con valores mínimos en todas las canciones, por que parecerá que todos los instrumentos y voces están más separados y que ha aumentado su número y claridad. En la figura 1.3.0 se encuentran todos los valores ajustables y los modos predefinidos disponibles.

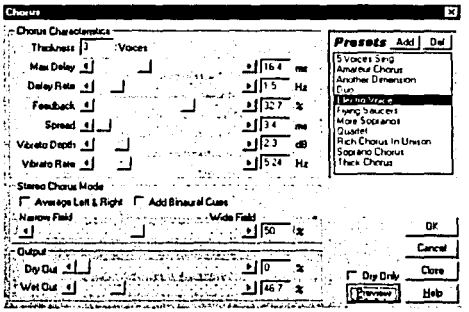


Figura 1.3.0. La configuración de los coros debe efectuarse teniendo en cuenta el efecto deseado, con la opción preview se escucha el resultado antes de ser aplicado.

El eco <Echo> es el efecto más recurrido y mediante los valores predefinidos o alternando el retardo del volumen y la ecualización del eco se consigue transformar una pieza musical en una interpretación de una sala de conciertos al aire libre, de tal forma que parezca una audición en vivo.

En <Cool Edit Pro>, la función 3d <Echo Chamber> da más posibilidades a este efecto y permite ajustarnos a las dimensiones de nuestra sala de escucha. Además, Echo Chamber es útil en la conversión de una muestra monoaural estéreo, porque eliminando la intensidad y el número de ecos, y duplicando o triplicando las distancias entre los micrófonos izquierdo y derecho se consigue separar la voz entre los dos canales dirigiéndola levemente a uno de ellos, lo que constituye una imagen estéreo más creíble.

ELIMINACIÓN DE RUIDOS

En algunas grabaciones es necesario eliminar picos de señal que distorsionan el sonido. Por medio de la función de normalización (Normalize), eliminamos los excesos de señal conforme a un nivel estándar que puede ser alterado para ajustar el sonido a las características de la fuente de reproducción.

Antes de la eliminación de ruidos, conviene realizar un ajuste de las frecuencias si la onda original muestra exceso o falta de agudos y graves. Estos efectos son casi imposibles en las grabaciones digitales realizadas durante los últimos 10 años, pero son bastante frecuentes en grabaciones obtenidas desde casette, de canciones antiguas o en formato monoaural.

Con las funciones de ecualización gráfica y paramétrica se efectúa un ajuste similar al que el usuario ha efectuado en su equipo de alta fidelidad.

Como se aprecia en la figura 1.3.1., la ecualización gráfica sobre 30 bandas otorga una enorme precisión en todas las frecuencias y los ajustes predeterminados realizan el efecto deseado con sólo seleccionar uno de ellos ya que cubren todas las necesidades; como la recuperación de agudos, enfatización de graves, entre otras.

Para comprobar el resultado antes del cambio la opción <Preview> es indispensable. Si se requiere profundizar en la corrección de las frecuencias, el ecualizador paramétrico es más eficaz porque permite dirigir el efecto en exclusiva hacia una determinada banda estableciendo con mayor exactitud la frecuencia mediante el uso de una representación gráfica con dos barras de nivel de graves y agudos. En ambos sistemas de ecualización es conveniente aumentar al máximo el valor de precisión <Acuracy> aunque el procesamiento requiera de más tiempo.

El filtro FFT es útil para limpiar el sonido eliminando frecuencias que no son audibles por el oído humano y recortando los agudos innecesarios.

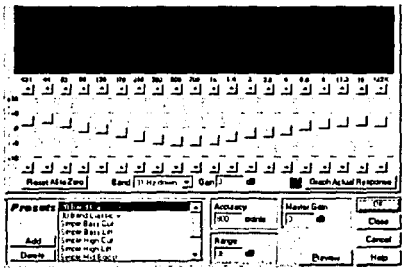


Figura 1.3.1. Ecualizador gráfico

FILTROS REDUCTORES DE RUIDO

Son algoritmos que reducen todos los ruidos, pero la reducción excesiva también recortará frecuencias de la calidad musical, sobre todo en los agudos.

<Click/Pop Eliminator> es un filtro que siempre se utiliza para reducir los picos altos, defectos de sonido y en general ruidos típicos en casettes y sobre todo en los discos de acetato, esos chasquidos continuos si el disco está en mal estado o la aguja es de mala calidad. Estos chasquidos son claramente visibles en la onda, son esas finas líneas verticales que muestran una repentina subida en la amplitud durante un mínimo espacio de tiempo.

<Hiss Reduction> elimina el molesto ruido continuo que producen los mecanismos de arrastre de las cintas de un casette. Como tamaño FFT el valor mínimo aconsejable es de 12,000 puntos, siendo 30 un buen factor de precisión, 10 db un valor correcto para <Transition Width>, un 50% en "Spectral Decay" y 12 db como reductor de ruido. La función <Get Noise Floor> es muy útil para identificar el nivel de ruido si logramos aislar una zona de muestra en la que sólo exista el ruido.

Para eliminar el ruido de fondo es necesario seleccionar una sección de la onda que sólo contenga ruido de fondo, como se muestra en la figura 1.3.2.

Estas zonas normalmente se encuentran el principio o al final de la canción. Entonces se abre el menú de <Noise Reduction> y se emplea la función <Get Profile From Selection> con un valor de <Snapshots> mínimo de 84. Esta captura de nivel de ruido determinará los valores de FFT y del factor de precisión (ver figura 1.3.3). Este procesamiento llevará probablemente unos cinco minutos, pero al final obtendremos una canción libre de ruidos con una musicalidad casi intacta como se muestra en la figura 1.3.4. Los tramos iniciales o finales que siempre contienen ruido de fondo pueden ser limpiados con la opción de cortar. Posteriormente, para recuperar esos segundos que debe haber entre pista y pista, se debe insertara dos segundos de silencio digital mediante la función <generate silence>, aunque por lo general los programas de grabación de CD Audio ya insertan silencio digital entre pistas.

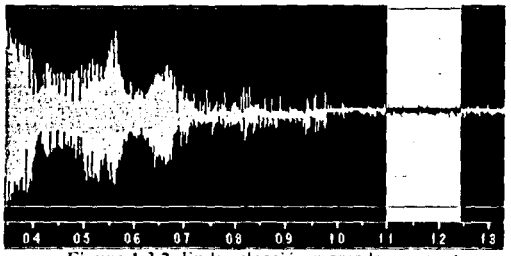


Figura 1.3.2. En la selección marcada se muestra ruido de fondo en nuestra señal de onda.

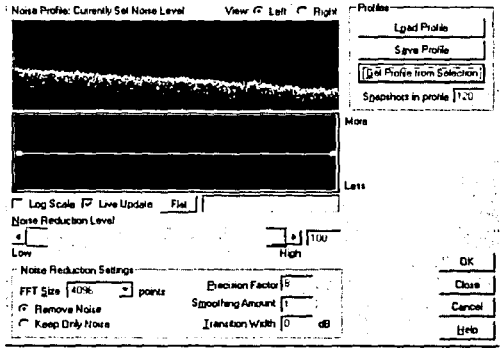


Figura 1.3.3.. Mediante *"Noise Reduction"* utilizamos como patrón una parte de ruido para limpiar todo el archivo de onda.

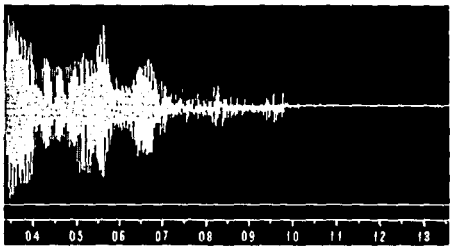
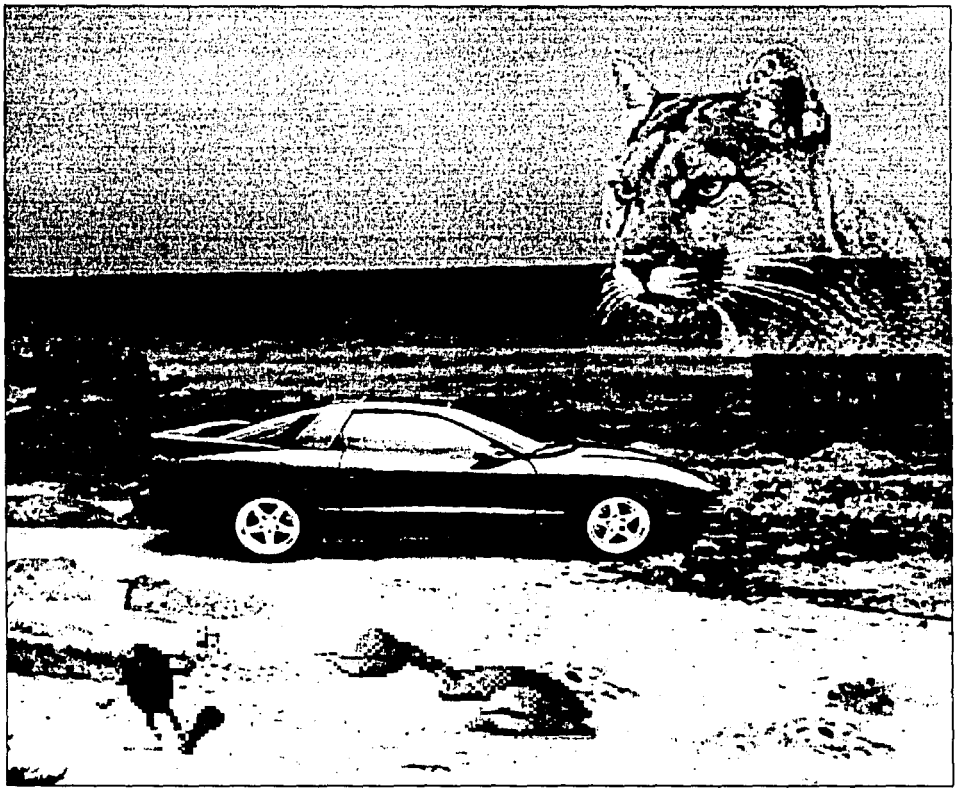


Figura 1.3.4. Archivo de ondas resultante, casi libre de ruido después de aplicar *"Noise Reduction"*.

CAPÍTULO 2

Diseño y modificación de imágenes digitales por medio de Adobe Photoshop 5.5.

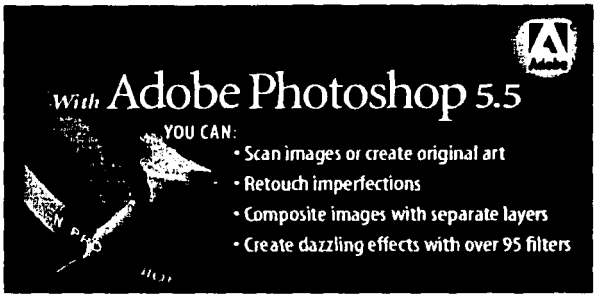


Una imagen dice más que mil palabras, la imagen es otra herramienta fundamental para el desarrollo de video juegos y programas educativos. La mayoría de pantallas de programas se pueden capturar en el portapapeles de la computadora, únicamente presionando la tecla de Impresión de Pantalla, abriendo el *Paint* de Windows y seleccionando la opción *Pegar* dentro del menú de Edición.

Algunos programas de edición de Imágenes tienen su propio capturador de pantallas pero no todas las aplicaciones permiten la captura, otros contienen una aplicación que permite "*escanear*" imágenes y paralelamente pueden aplicarse sobre ellas diversidad de filtros y efectos, cambios de resolución o tamaños de imágenes, rotar, cambiar los colores de alguna parte de la imagen, sobreponer otras imágenes, aumentar el brillo, etc. Como ejemplo de estos programas pueden nombrarse aplicaciones como **Adobe PhotoShop** versión 5.5. para Windows 98, el cual trataremos de abordar las opciones más importantes. Da un toque de especial y una mejor presentación a nuestros programas. Son un conjunto imágenes diferentes presentadas en forma consecutiva. Existen programas para el tratamiento de imágenes en 2 y 3 dimensiones y generar videos, entre estos tenemos:

- *AutoCAD*
- *Autodesk Animator*
- *3D Studio Max*

Estos programas ofrecen enormes posibilidades en la creación de secuencias de animaciones proporcionando cierto realismo a la aplicación final. Los archivos de video, gracias a la tarjeta digitalizadora de video y a las imágenes que le proporciona una televisión, cámara o reproductor de video, permiten dar el toque de realismo a cualquier aplicación a desarrollar.



Adobe Photoshop es un programa para modificar y diseñar imágenes digitales. Partiendo de fotografías o diseños importados de otros programas se puede dar un realismo casi perfecto utilizando variaciones de tonos, texturas, color, etc.; solo es necesario un poco de creatividad de parte del usuario.

Este programa se encuentra disponible en la red en diferentes versiones y plataformas en la siguiente dirección: http://www.adobe.com/products/photoshop/

La versión 5.5 cuenta con 20 herramientas básicas para la creación y modificación de imágenes como se muestra en la figura 2.1.0.

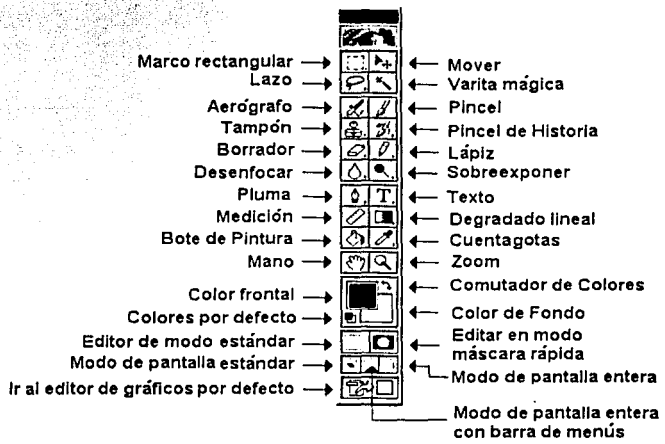


Figura 2.1.0 Herramientas básicas de PhotoShop

Todas las herramientas permiten opciones. Al hacer clic sobre ellas, se abre la paleta de opciones con las propiedades específicas para la herramienta seleccionada.

Lazo magnético

Se utiliza para seleccionar una imagen o parte de la imagen en forma automática en función del porcentaje del contraste establecido en la paleta de opciones de lazo magnético en una escala de 1 a 100% . Por ejemplo, si se requiere extraer parte de una imagen de la figura 2b, se debe seleccionar el contorno de la imagen deseada manteniendo presionado el botón izquierdo del ratón y la imagen se cierra automáticamente al dejar de presionar el botón del ratón.

Figura 2.2.0. Selección de una imagen

Una vez seleccionada la imagen se puede copiar en otro archivo diferente, abriendo uno nuevo y copiando el contenido del portapapeles* en este nuevo archivo.

* Espacio de memoria donde se almacenan datos temporalmente usando la opción copiar.

Varita mágica 

Esta es una herramienta de gran utilidad que selecciona con sólo pulsar sobre un área de la imagen todos los píxeles que se encuentran en el área de tolerancia definida en la paleta de opciones de herramientas. Mediante el uso de esta herramienta, es posible perfilar imágenes bien definidas, o seleccionar campos de color siempre que exista un mínimo de contraste en los perfiles. Por ejemplo en la figura 2.2.1 se selecciona únicamente el color negro de la zona marcada.

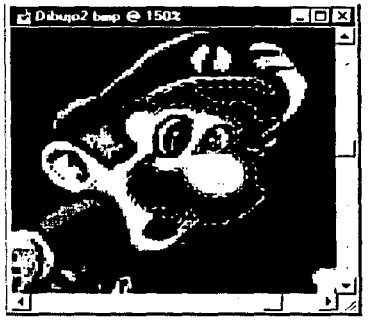


Figura 2.2.1. Selección de color en una imagen

Bote de pintura 

Esta herramienta se utiliza para rellenar parte de una imagen que tenga bien definida su interior, de lo contrario se cubre el total de la imagen. Como ejemplo tenemos la figura 2.2.2 que originalmente tenía el color blanco de fondo y utilizando el bote de pintura se le cambió el color de fondo. Se debe de utilizar esta herramienta otra vez para rellenar la parte que falta para completar el fondo verde. Si el color de fondo es una combinación de varios colores, no es posible utilizar esta herramienta.

Figura 2.2.2. Rellenado de color.

Zoom

Con el uso de esta herramienta se puede visualizar la imagen en diferentes escalas con sólo pulsar sobre la zona deseada. Como ejemplo tenemos una imagen y visualizada en un 500%. Como se muestra en las *figuras 2.2.3. y 2.2.4.*

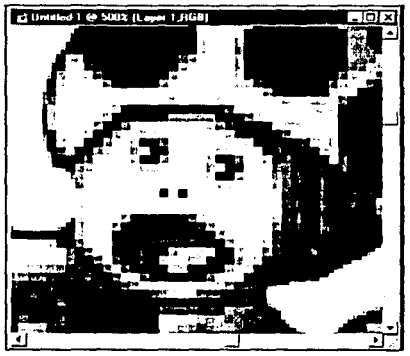


Figura 2.2.3. Imagen amplificada en un 500%

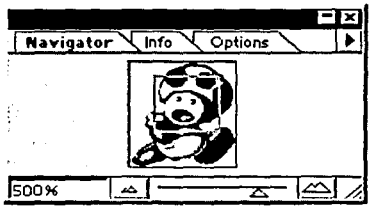


Figura 2.2.4. Selección de la parte a visualizar mediante el navegador.

Borrador

Esta herramienta permite corregir los errores y eliminar partes que no se deseen. En la *figura 2.2.5.* utilizamos el borrador para eliminar una parte de la boca pintada de rojo utilizando los diferentes tipos de gomas.

Figura 2.2.5. Imagen con parte de la boca borrada.

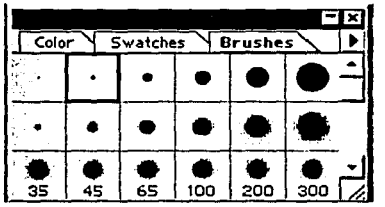


Figura 2.2.6. Uso de los diferentes tipos de pinceles o Brochas, que en este caso se utilizan como una goma.

Mano

Cuando se realizamos un Zoom y la imagen o no se puede visualizar en forma completa se utiliza esta herramienta, se presionando el botón Izquierdo del ratón y se arrastra la imagen hacia la parte que en que se desea trabajar.

Figura 2.2.7. Desplazamiento de una imagen para su visualización

Pincel

Es la forma usual de dibujar utilizando la paleta de pinceles con diferentes formas y tamaños. Ejemplo:

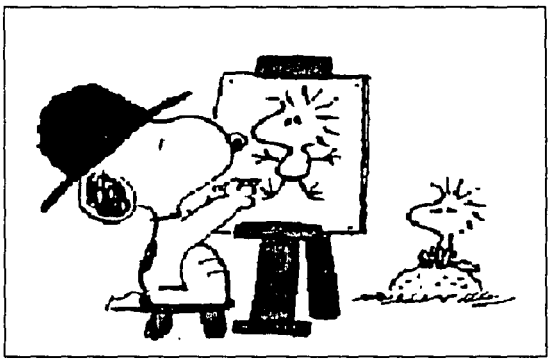


Figura 2.2.8. Utilización de diferentes tipos de pinceles.

Aerógrafo

Esta herramienta es una versión electrónica que emula el aerógrafo mecánico tradicional y se utiliza para crear luces, sombras suaves y aplicar colores difuminados.

Como ejemplo tenemos una imagen y un letrero sobre un muro; para lograr este tipo de imágenes es necesario un poco de dedicación, acercar un poco la imagen para ver mejor los detalles, usar los colores y tonos adecuados. También es necesario seleccionar el tamaño y el porcentaje de pintado del aerógrafo.

Figura 2.2.9. Uso de aerografo para pintar un graffiti en una pared virtual.

Tampón

Esta herramienta es de gran utilidad para restaurar áreas deterioradas de una imagen, cubriéndola con otros fragmentos similares de la misma imagen utilizando la tecla *Alt* y el *botón izquierdo* del ratón. Ejemplo:

- Selecciona la herramienta tampón y presiona la tecla *Alt* y el botón Izquierdo del ratón en la parte de la figura que deseas copiar.
- Desplaza la parte que tiene el círculo al lugar donde quieres que se copie la parte que tiene la cruz.

Mediante esta herramienta se puede copiar un fragmento de una imagen y aplicarla posteriormente pintando como si fuera un pincel que utiliza la imagen copiada como pintura.

Si mueves el ratón se moverá también la parte que tiene la cruz y copiará automáticamente esa parte en donde se encuentra el círculo, como se muestra en las figuras 2.2.10 y 2.2.11.

Figura 2.2.10. Imagen normal

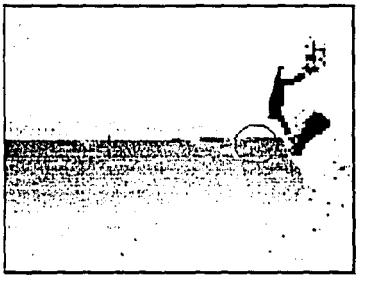


Figura 2.2.11. Imagen con copiado de secciones por medio del tampón

Pincel y paleta de historia

Una aplicación del pincel de historia se produce cuando después de abrir una imagen y aplicarle cualquier efecto o herramienta de dibujo se desea rescatar la imagen original y seleccionando en la casilla que indica el estado a recuperar.

Como ejemplo abrimos el archivo león1.bmp, reemplazamos el color de la imagen, dibujamos unas líneas en el brazo y finalmente cambiamos el color de fondo. Si seleccionamos la acción de pintar (*Airbrush*) dentro de la paleta de historia, nos regresa a la acción realizada anteriormente.

Ejemplo:

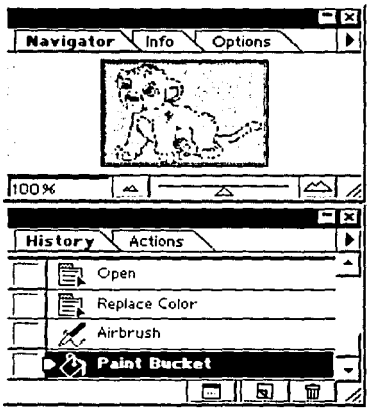


Figura 2.2.12. Imagen normal con la última acción de pintado de fondo

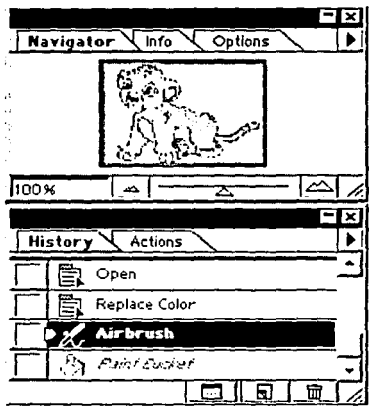


Figura 2.2.13. Imagen con la penúltima acción de pintado de manchas sobre el brazo.

Borrador mágico

Esta herramienta es de gran utilidad para crear imágenes semitransparentes y constituye un medio rápido de eliminar (con sólo pulsar en un punto de la imagen) todas las áreas contenidas en un espacio de tolerancia a partir del punto donde se pulsa. Ejemplo: Para obtener imágenes o parte de una imagen en forma semitransparente *(Figura 2.2.15)*, se debe seleccionar el tamaño de la goma y el grado de transparencia *(figuras 2.2.16 y 2.2.17)* y borrar la parte que se desee.

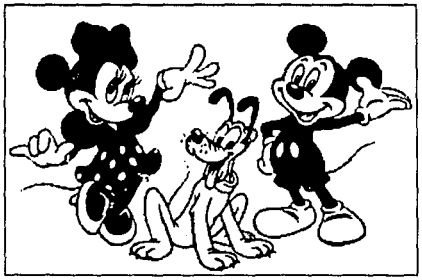


Figura 2.2.14. Imagen normal.

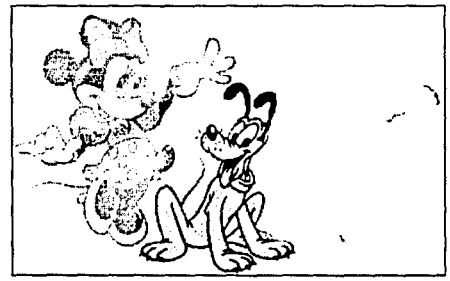


Figura 2.2.15. Imagen utilizando el borrado semitransparente.

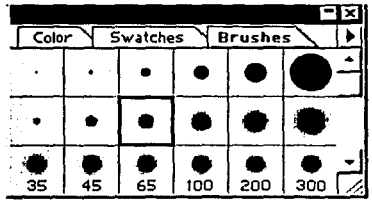


Figura 2.2.16. Tipos y tamaños de pincel.

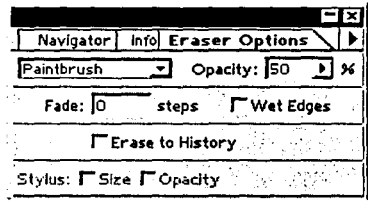


Figura 2.2.17. Porcentaje de Transparencia para el borrador.

Lápiz

Esta herramienta emula con bastante fidelidad los trazos que se realizan con un lápiz convencional. Ajustando el tamaño del lápiz y la visualización en un 200% para visualizar mejor los detalles se pueden crear dibujos de historietas o caricaturas; para ello se debe tener un muy buen pulso y un poco de paciencia. Ejemplo:

Figura 2.2.18. Imagen dibujada utilizando la herramienta Lápiz.

Dedo

Esta herramienta permite modificar una imagen diluyendo los colores mediante arrastre, con un efecto similar al que produciría frotar con un dedo sobre una pintura fresca. Esta herramienta es muy útil para retoques. Al Realizar un fotomontaje se puede utilizar para unir el color del contorno de la segunda imagen con el de la primera o con el fondo de la imagen; pareciendo así que se trata de una imagen real. En el ejemplo siguiente se utiliza esta herramienta para unir el contorno de la imagen con el color de fondo pero únicamente del lado izquierdo para notar la diferencia con la del lado derecho.

Figura 2.2.19. Imagen suavizada de los bordes del lado izquierdo mediante la herramienta dedo.

Degradado

Mediante esta herramienta es posible realizar mezclas de color que se funden en forma gradual, mezclas de color y espacios trasparentes. Se puede utilizar para rellenar letras, crear efectos de sombra, luces y fondos de pantalla.

A través de la paleta de opciones de degradado se puede especificar el modo de fusión, tipos de colores y la Transparencia. Existen varios tipos de degradado entre los que destacan los de las siguientes figuras:

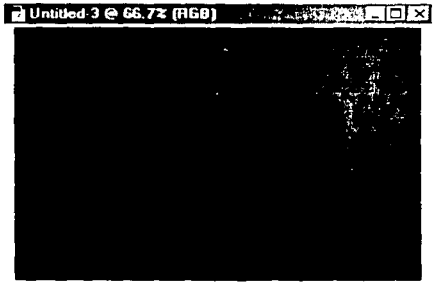


Figura 2.2.20. Degradado Reflejado

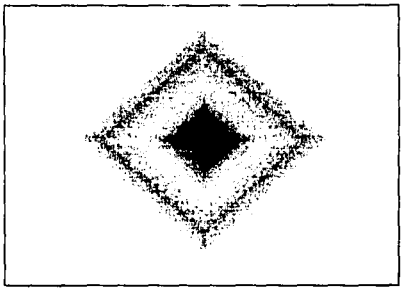


Figura 2.2.21. Degradado en forma de Diamante

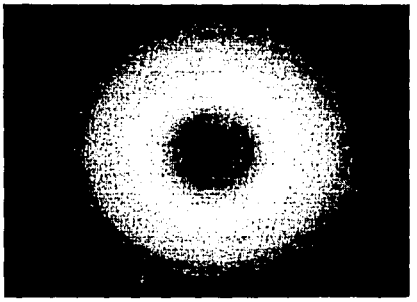


Figura 2.2.22. Degradado Radial

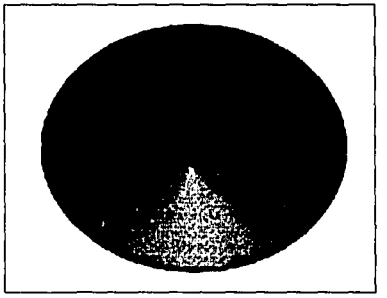


Figura 2.2.23. Degradado Angular

CAMBIO DE TAMAÑO O REMUESTREO DE UNA IMAGEN

Para aumentar o reducir el tamaño de una imagen, necesitamos cambiar el número de píxeles* de la imagen dentro del diálogo que aparece al ejecutar la orden del *menú> tamaño de imagen...*

Al remuestrear (*cambiar de tamaño*) una imagen se pueden producir pequeñas variaciones de la misma que después de múltiples cambios se puede llegar a producir variaciones que serán irrecuperables, a menos que se haya guardado o que se pueda hacer uso de la paleta Historia para retornar al original. Si se aumenta su tamaño también aumenta el espacio que ocupa en disco y si el número de píxeles no es proporcional, la imagen queda distorsionada (mas pequeña o más grande de un lado). Ejemplo:

Figura 2.3.0. Imagen en tamaño Normal
(283 x 207) píxeles

Figura 2.3.1. Imagen aumentada a
(500 x 366) píxeles

* Serie de puntos en que se divide la pantalla y depende de la resolución del monitor.

BRILLO Y CONTRASTE

A través de la orden *Menú Imagen> Ajustar> Brillo/Contraste*, se mejora la calidad de definición de las imágenes y es de las opciones más utilizadas, pues la mayor parte de las fotografías digitalizadas necesitan un pequeño retoque que es muy fácil y rápido de realizar a través de esta ventana. Como ejemplo tomamos una imagen digitalizada de una revista y aumentamos un poco de brillo y contraste para mejorar un poco la visión de la imagen. Ver figuras 2.3.2 y 2.3.3.

Figura 2.3.2. Imagen normal

Figura 2.3.3. Imagen con más brillo y contraste

EQUILIBRIO DE COLOR

Mediante la orden menú *Imagen> Ajustar> Equilibrio de color*, es posible efectuar modificaciones sobre la mezcla de colores, referidas a las sombras, semitonos y luces. Como ejemplo tenemos una imagen capturada de un juego y queremos que cambie su tono de color, para esto aumentamos el color rojo, el Magenta y el azul para crear una imagen semejante, pero con tonos de color diferentes, como se muestra en las figuras 2.3.4, 2.3.5 y 2.3.6.

Figura 2.3.4. Imagen normal

Figura 2.3.5. Imagen con ajuste de color

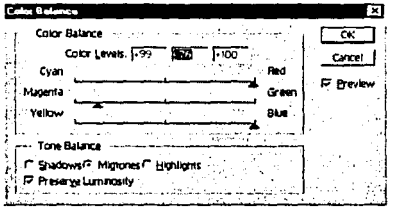


Figura 2.3.6. Balance de color.

REEMPLAZAR COLOR

Esta opción permite modificar áreas del mismo color y a la que se accede con la orden menú *Imagen> Ajustar> Reemplazar color...*

En el siguiente ejemplo, se selecciona de la imagen 2.3.7 el color verde y se le cambian los tonos hasta quedar de color amarillo o cualquier otro color que desee. Ver figura 2.3.9.

Figura 2.3.7. Imagen Normal.

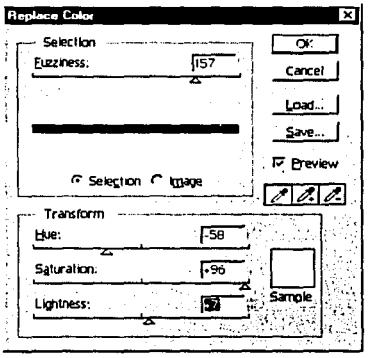


Figura 2.3.8. Reemplazo de color.

Figura 2.3.9. Imagen con reemplazo de color.

INVERTIR UNA IMAGEN

Mediante esta orden del menú *Imagen> Ajustar> Invertir*, se transforman todos los píxeles de la imagen por sus complementarios produciendo el efecto de la imagen en negativo.

De la figura 2.4.0 se crea una sombra, dando el efecto de tener un nuevo contrincante y así agregar a nuestros juegos.

Figura 2.4.0. Imagen Normal

Figura 2.4.1. Imagen con inversión de colores y rotación en forma horizontal.

EXTRACCIÓN DE IMÁGENES

Al pulsar la orden menú *Imagen> Extraer...*, se muestra una ventana mediante la cual es posible definir el perfil de la imagen y el contenido que se desea conservar en la extracción. Para extraer un objeto es necesario definir el área que se desea extraer utilizando la herramienta de resaltar bordes, arrastrar y seguir el contorno de la imagen.

Si tenemos una fotografía *escaneada*[*] (Figura 2.4.2) y deseamos extraer solo el automóvil, seleccionamos el contorno de la imagen mediante esta opción, utilizamos el bote de pintura y hacemos *click* sobre el interior del área formada, a continuación presionamos el botón *preview* para ver si nuestra extracción a sido favorable, finalmente presionamos *ok* y se crea automáticamente una capa con la imagen extraída.

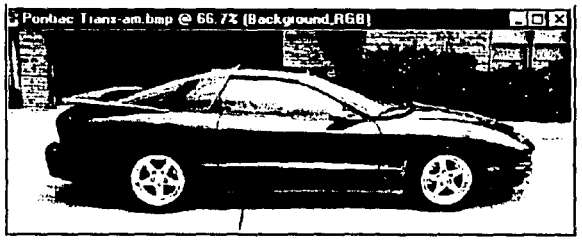


Figura 2.4.2. Imagen Escaneada de una Foto.

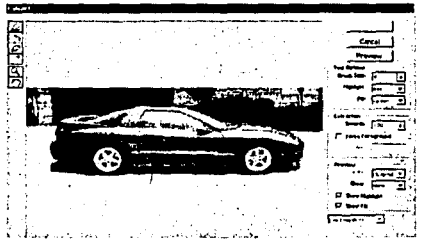


Figura 2.4.3. Área seleccionada para la extracción del automóvil.

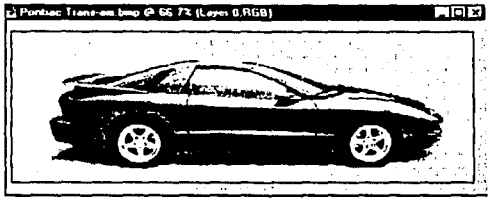


Figura 2.4.4. Extracción Final del automóvil en una Capa.

[*] Imagen digitalizada por medio de un escáner.

Ya que tenemos nuestra capa de la imagen extraída la podemos copiar a otra imagen y diseñar nuestras propias pantallas para juegos.

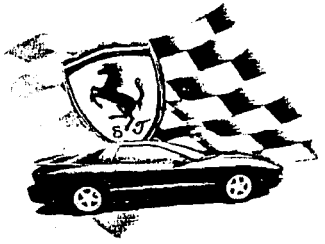


Figura 2.4.5. Unión de imagen de fondo y capa de automóvil.

Si queremos incluir nuestro automóvil en una pista de carreras tenemos que hacer una rotación, ajustar el tamaño y darle un poco de inclinación como se muestra en la figura. También podemos aplicarle un filtro de desenfoque para dar una ilusión de una fotografía con la imagen en movimiento (figura 2.4.7).

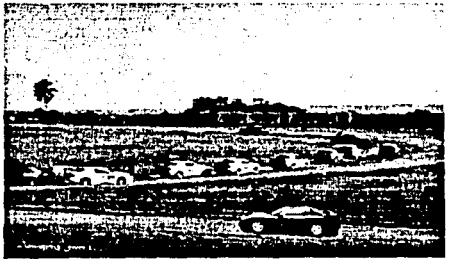


Figura 2.4.6. Automóvil en una pista de carreras.

Figura 2.4.7. Efecto de desenfoque de movimiento.

INSERCIÓN DE TEXTOS

El texto es sin duda un complemento muy importante a cualquier imagen ya muestra mensajes o datos que resultan de mucha importancia.

Al crear texto con la herramienta de texto y con la herramienta de texto vertical, se crea automáticamente una nueva capa, la cuál es posible modificar haciendo doble clic y aparecerá una ventana con opciones de tipos de fuente, estilos, tamaño, color, posición del texto y algunos efectos. Como ejemplo podemos crear un archivo de 600 x 400 píxeles y agregar el texto "UNAM" con tipo de letra **Arial**, estilo **Bold**, tamaño **100** puntos y alineación **izquierda,** como se muestra en la siguiente figura:

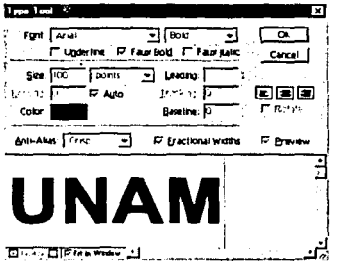


Figura 2.4.8. Ventana de Opciones de la herramienta de texto.

Efectos en capas

Además a nuestro texto le podemos agregar un efecto de sombra para que se vea un poco mejor por medio del menú *Capas>Efectos>Sombreado de Gota*.

También se pueden modificar algunas opciones como el ángulo la distancia, el tamaño de la sombra y la intensidad como se muestra en la siguiente figura:

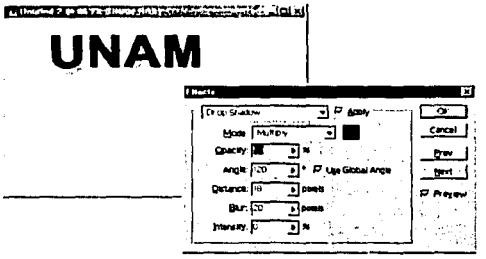


Figura 2.4.9. Ventana de opciones con efecto de sombra.

Uso de Capas en imágenes

Por medio de los menús de la paleta de capas, es posible crear, mostrar, copiar, combinar y eliminar capas. A través de las opciones de la paleta, se puede también especificar el modo de fusión, la opacidad y la transparencia *(Figura 2.5.1)*.

Al crear una nueva capa y pegar nuestra imagen visualizamos las capas que deseemos o borrarlas con el bote de basura *(Figura 2.5.3.)*.

Una vez visualizadas todas nuestras capas no podemos modificar parte de una capa si esta capa no se encuentra activada de color azul en la paleta de capas.

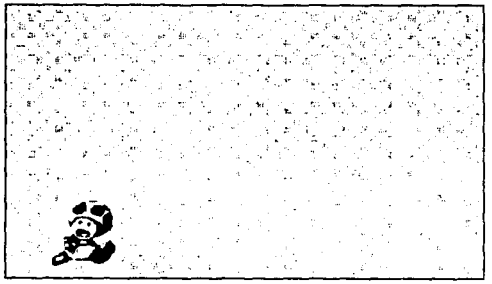


Figura 2.5.0. Capa número 2 con la imagen de un hongo.

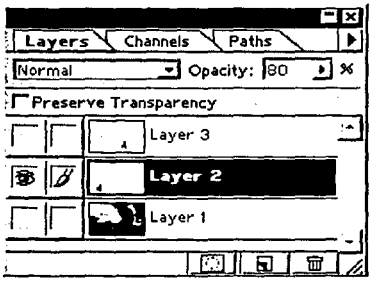


Figura 2.5.1. Visualización de la capa número 2.

Figura 2.5.2. Unión de tres capas.

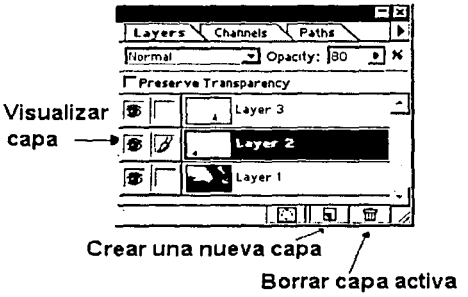


Figura 2.5.3. Visualización de 3 capas y con la segunda capa activa.

Uso de capas trasparentes

Para dar una mejor presentación a nuestras pantallas, hacemos uso de las capas trasparentes, como ejemplo utilizamos nuestro texto de *figura 2.4.9.* y se le añade una nueva capa de color azul trasparente cambiando la opacidad a un 20% en las opciones de la capa y se incluyó una imagen de un puma. Ejemplo:

Figura 2.5.4. Capa color azul al 20%.

Cambio de color y posición en textos

Para cambiar el color es necesario utilizar la herramienta bote de pintura y para rotar las letras utilizamos la opción del menú *imagen>rotar*. Esto es muy importante ya que da una presentación diferente y colorida a nuestros textos y es muy común verlo en cualquier tipo de juegos.

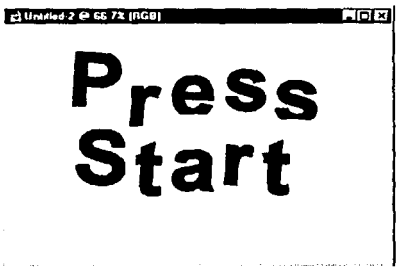


Figura 2.5.5. Balance de color.

Creación de textos en perspectiva

Con la opción Edición>Transformar>Distorsionar se provoca una distorsión geométrica de la imagen, creando cambios en su forma. Como ejemplo crearemos un texto en perspectiva utilizando esta opción, para ello realizaremos los siguientes pasos:

- Utilice la herramienta mascara de texto y escribir X-MEN.
- Seleccione la opción ***Edición>Transformar>Distorsionar*** y jale las esquinas como se muestra en la siguiente figura:

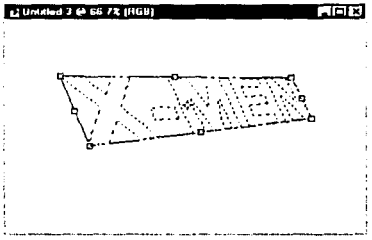


Figura 2.5.6. Letras en perspectiva.

- Pinte las letras con el bote de pintura de color amarillo.
- Cree una copia de las letras en otra capa y pinte las letras de la nueva capa en color azul.
- Mueva la capa del texto de color azul un poco hacia arriba y a la izquierda.
- Cambie el color de fondo a color negro. *(Figura 2.5.7).*

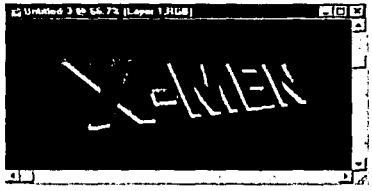


Figura 2.5.7. Capa número uno en color amarillo y Capa número dos en color azul.

- A añada el letrero "1995 ALL RIGHTS RESERVED" en la parte de abajo
- Finalmente agregue un efecto de luz en el fondo de la imagen por medio del menú *Filtro>Modelado>Luz* *(Figura 2.5.8).*

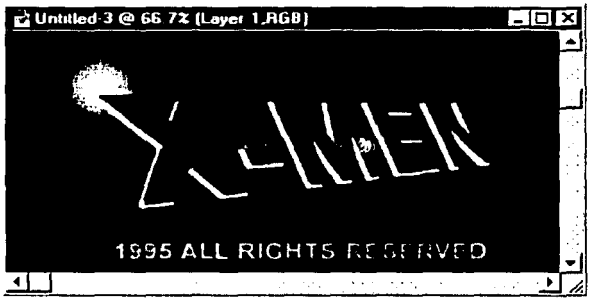


Figura 2.5.8. Balance de color.

Efectos de Iluminación

Filtro>Modelado>Efectos de Iluminación.

Este efecto permite aplicar diversos tipos de puntos de luz, y cada uno puede aplicarse con diferentes propiedades. Esta opción permite resaltar áreas específicas de la imagen y también modificar el control de luz.

En la siguiente figura se aplican varios efectos, entre ellos el degradado a las letras y efectos de iluminación para así utilizar en una pantalla principal de juego de video.

Figura 2.5.9. Efectos de luces y degradado de letras.

Transformación 3D

Filtro>Interpretar>Transformación 3D

Este filtro permite manipular una imagen normal como si se hubiera creado en un programa 3D, marcando únicamente el área correspondiente y seleccionando la forma que se desee otorgar, ya sea un volumen en forma de cilindro, cubo o esfera. como se muestra en las siguientes figuras:

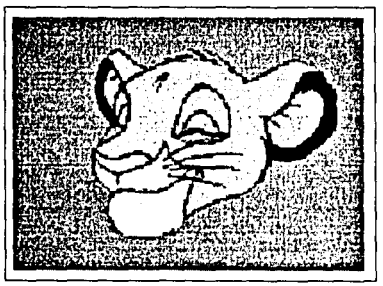


Figura 2.6.0. Imagen Normal

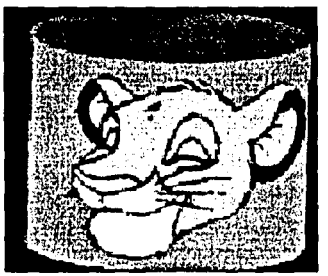


Figura 2.6.1. Imagen en forma de Cilindro

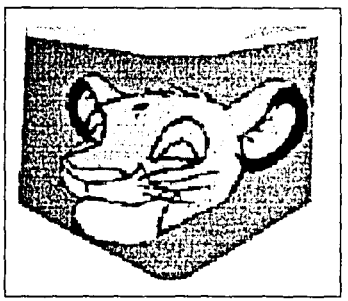


Figura 2.6.2. Imagen en forma de Cubo

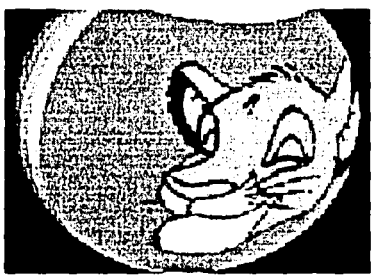


Figura 2.6.3 Imagen en forma de Esfera

FILTRO MOLINETE

Al aplicar el *Filtro>Distorsionar>Molinete* sobre la imagen o parte de la imagen, produce un giro concéntrico que presenta una mayor distorsión cuando mayor es el ángulo y se obtendrá un efecto como el de la *figura 2.6.7*. Este es un buen efecto para la creación de efectos para juegos; en este caso da un efecto de que el auto va a entrar a una dimensión desconocida.

Figura 2.6.4. Selección de imagen para aplicación de un filtro.

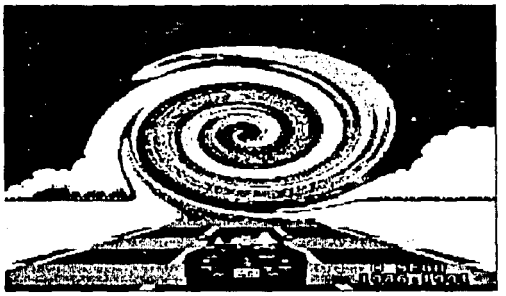
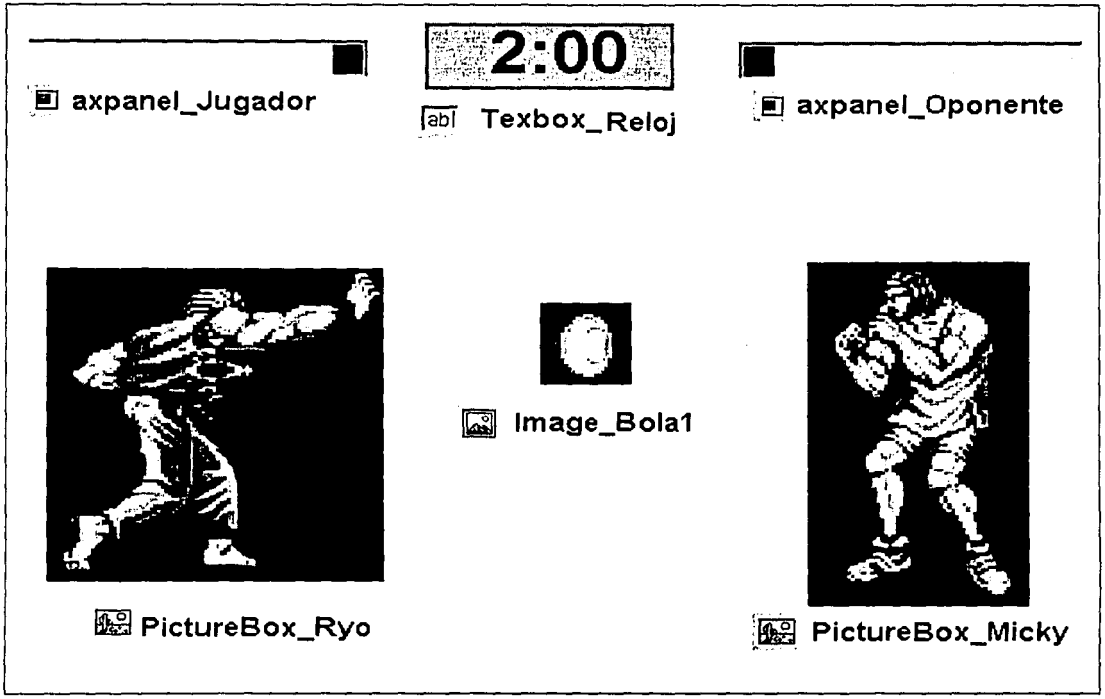


Figura 2.6.5. Selección con Filtro de distorsión.

CAPÍTULO 3

Elementos de programación para el diseño de funciones

En Visual Basic



El uso de un programa Orientado a objetos y relacionado a eventos como lo es ***Visual Basic***, facilita enormemente el diseño y elaboración de programas: únicamente se agregan los controles, se le asignan las propiedades adecuadas a cada control y se programan las funciones lógico-matemáticas relacionadas a cada evento.

En este capítulo encontraras lo indispensable para el manejo básico de *Visual Basic,* como: crear formularios, aprender a utilizar todas las herramientas que ofrece el lenguaje y crear tus propios menús; también encontrarás como utilizar variables, los operadores aritméticos y lógicos, el manejo de sentencias de control, ejemplos de funciones y procedimientos, entre otras cosas.

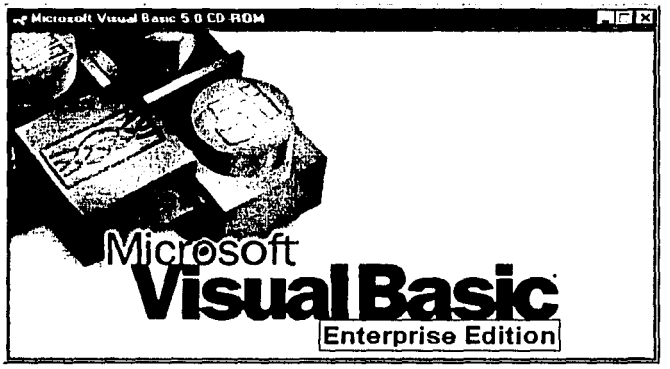
En la creación de nuestros juegos utilizaremos tres sucesos muy importantes para el manejo de las teclas: KeyDown, KeyPress y KeyUp, los cuales nos dan un control total en la utilización del teclado y así poder manejar nuestros juegos.

Un control muy importante y que casi siempre utilizaremos en nuestros juegos es el temporizador o reloj que sirve para realizar cualquier proceso en segundo plano cada vez que transcurra un intervalo de tiempo.

En este capítulo se describe como utilizar el manejo de la paleta de colores utilizando la función RGB o la función QBColor.

También se describe como crear gráficos en nuestro formulario y hacer combinaciones de líneas, cuadros y círculos. Cambiar sus propiedades: moverlos de lugar, cambiar colores, hacerlos desaparecer y aparecer, etc.

Por último, se da un ejemplo de cómo manejar una base de datos realizada en *Dbase III* mediante un *DataControl* y visualizarla información mediante cajas de texto.



Visual Basic es un lenguaje de programación diseñado para crear aplicaciones con interfaz gráfica de una forma rápida y sencilla. Utiliza dos tipos de objetos: **ventanas** y **controles**, que permiten diseñar aplicaciones casi sin programar.

Al iniciar Visual Basic crea automáticamente un **formulario** que sirve para crear nuevas aplicaciones. Si se desea crear otros formularios utilice la opción formulario dentro del menú Insertar.

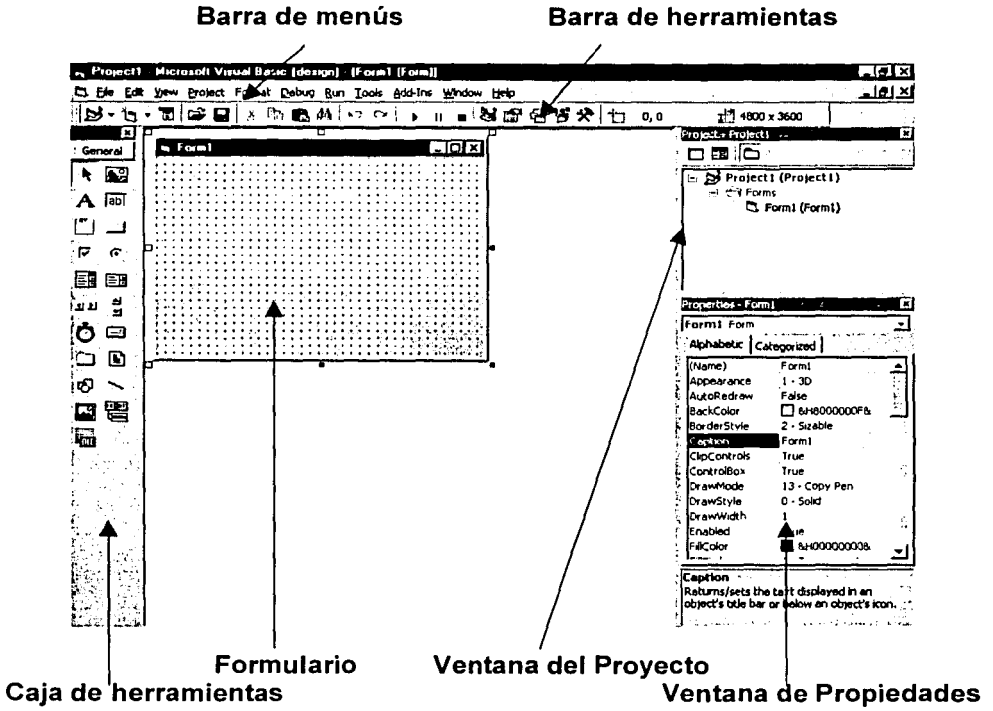
Un **formulario** es una ventana que sirve de fondo para la inserción de **controles** y gráficos. Se pueden utilizar varios formularios según se necesiten, ya sea para que contengan gráficos, para visualizar información o para aceptar datos. Los formularios tienen propiedades como color tamaño, etc. que se pueden seleccionar desde el menú...

Los **controles** son objetos que dibujamos sobre el formulario, tales como etiquetas, cajas de texto, marcos, botones de opción, etc.

Visual Basic tiene varias herramientas para facilitar el diseño de cualquier aplicación gráfica.

ELEMENTOS DE VISUAL BASIC

Cuando se inicia Visual Basic se ve una ventana como la siguiente:



En esta ventana se distinguen los siguientes elementos:

Barra de menús .- Visualiza las órdenes que se utilizan para desarrollar una aplicación (*Archivo, Edición, Ver, etc.*).
Barra de título.- Muestra el nombre del proyecto.
Barra de herramientas.- Facilita un acceso rápido a las órdenes más comúnmente utilizadas. Como:

- Abrir proyecto
- Guardar proyecto
- Ejecutar la aplicación

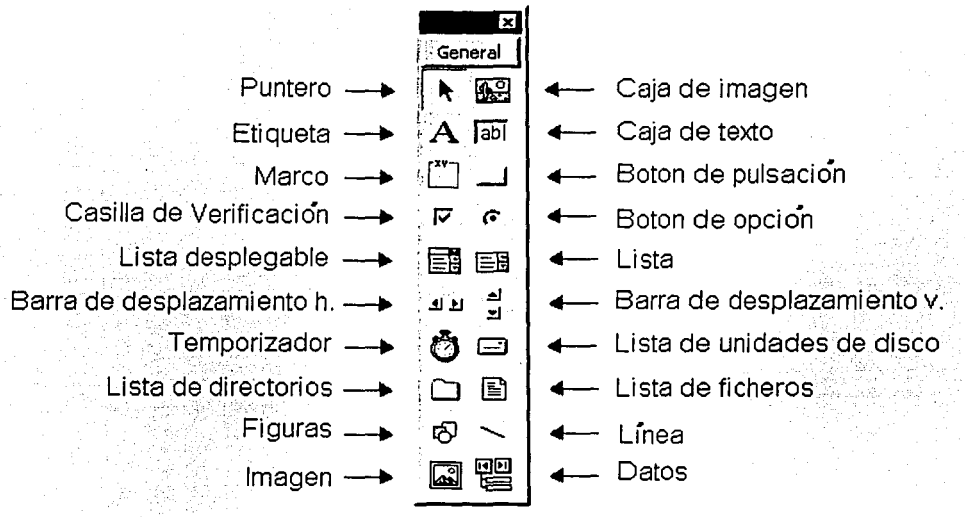
Ventana del Proyecto.- Es el conjunto de todos los ficheros (formularios, módulos, clases y recursos) que forman la aplicación.

Ventana de Propiedades.- conjunto de propiedades asociadas de cada objeto (*nombre, posición, tamaño, color, etc.)*

Para abrir la ventana de propiedades, ejecute la orden **Propiedades** del menú **Ver** o desde la ventana de herramientas presione el icono de Visualizar ventana de propiedades.

Formulario.- Es la ventana sobre la que colocamos los controles que utilizamos para nuestro programa.

Caja de herramientas.- Provee un conjunto de herramientas que permiten colocar los controles en el formulario durante el diseño de nuestro programa. Los controles mas comunes se exponen a continuación:

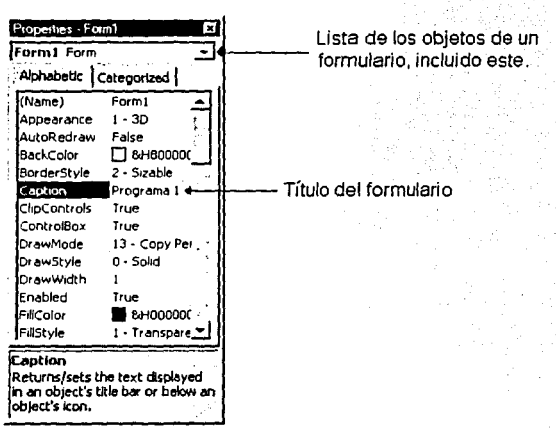


Si se requiere agregar mas controles a la caja de herramientas se requiere activar el control necesario utilizando la opción ***componentes*** del *menú proyecto.*

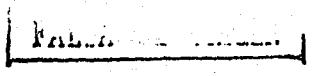
PROPIEDADES DE LOS OBJETOS

Cada objeto tienen definidas un conjunto de propiedades, como*: título, nombre, color,* etc.

Algunas propiedades son comunes a varios objetos y otras son únicas para un objeto determinado. Cada propiedad de un objeto tiene un valor por defecto que puede ser modificado si se desea, cambiando el valor de la propiedad dentro de la lista de propiedades. Por ejemplo: si se desea cambiar el título a nuestro formulario es necesario cambiar la propiedad *Caption*.



En la parte inferior de la ventana de propiedades se muestra una pequeña descripción de la propiedad seleccionada.



COMO CREAR UNA APLICACIÓN

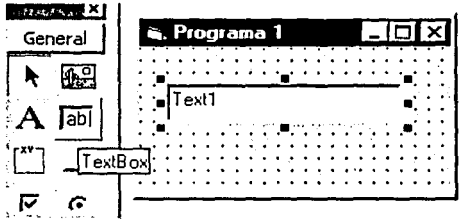
Para realizar una nueva aplicación se ejecuta la orden *Archivor>Nuevo proyecto*; también, al iniciar el programa automáticamente se crea una nueva aplicación. Una aplicación en Visual Basic puede estar formada por cuatro clases de ficheros: módulos de formularios (*.Frm*), módulos estándar (*.bas*), módulos de clases (*.cls*) y ficheros de recursos (*.res*).

Dentro del formulario se pueden dibujar controles, tales como cajas de texto, botones, etiquetas, etc.

Para borrar un control, primero se tiene que seleccionar el control y luego presionar la tecla *Supr o Del.*

Como ejemplo podemos crear una aplicación que muestre un mensaje en nuestra ventana; Para esto siga las siguientes instrucciones:

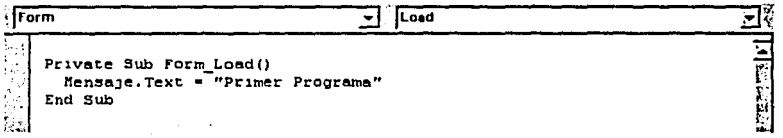
1. **Agregue una caja de texto al formulario,** seleccionando la caja de texto y dentro del formulario, presione el botón izquierdo de ratón y arrastrando el ratón sin soltar el botón hasta dar el tamaño adecuado a la caja detecto. Ejemplo:



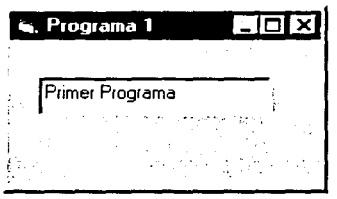
2. **Cambie el nombre de la caja de texto** utilizando la propiedad (Name) a valor Mensaje (para referirnos a ella en el código).

3. **Presione Doble Clic** sobre el formulario.

4. **Agregue dentro de la función:** Mensaje.Text = "Primer Programa". Ejemplo:



5. Finalmente **ejecute la aplicación** con el botón  y obtendrá el siguiente resultado:



Una aplicación en Windows es conducida por sucesos y orientada a objetos. Esto significa que podemos ligar unidades de código escritas para un determinado objeto a sucesos que pueden ocurrir sobre dicho objeto, de tal forma que cuando ocurra un suceso se ejecute la unidad de código correspondiente.

Un objeto es una entidad que tiene atributos particulares (*datos o propiedades*), y unas formas de operar sobre ellos (*métodos o procedimientos*). Cada objeto esta definida por una clase (*class*), entendiendo por clase un tipo de objetos.

Cuando se crea una aplicación se crean formularios y sobre ellos se dibujan controles; los formularios y los controles son objetos que cada uno de ellos tiene asociados un conjunto de propiedades.

Para referirse a una propiedad de un formulario desde otro formulario o módulo, debe especificar el nombre del formulario. Por ejemplo:

```
Form1.Caption = "Formulario 1"
```

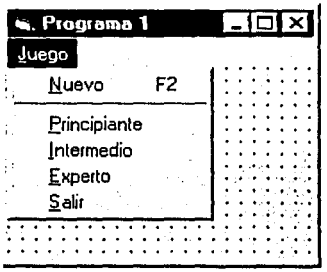
Para referirse a un control de un formulario desde otro formulario o módulo, debe especificar el nombre del formulario separado del nombre del control por el operador !. Por ejemplo:

```
Form1!Text1.ForeColor = &H0
```

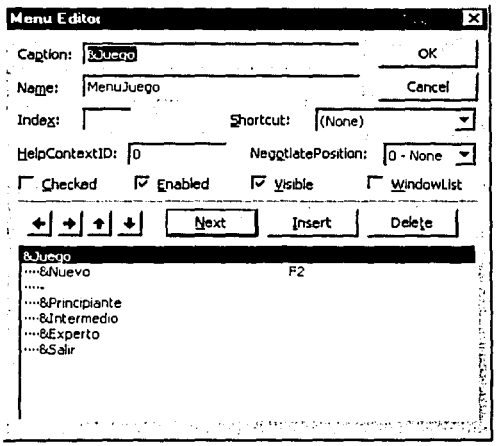
CREACIÓN DE MENÚS

Cuando en un programa se utilizan muchas órdenes, generalmente se agrupan en uno o más menús. Un menú se diseña utilizando el **editor de menús** que se encuentra dentro del menú **Herramientas**, o haciendo clic en el botón correspondiente de la barra de herramientas.

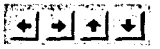
El menú **Juego** que vemos editado en la siguiente figura, contiene varias opciones : Nuevo, Principiante, Intermedio, Experto y Salir.



Para crear este menú utilizaremos el editor de menús como se muestra en la siguiente figura:



Como se observa, **Caption** es el título del menú y **Name** es el nombre del menú al que hacemos referencia mediante código. Para subrayar la letra que da acceso a un menú se utiliza un *ampersan* (&).

Para crear los submenús se utilizan las flechas  según el nivel que se requiera.

Si deseamos agregar un separador se escribe un guión (-) en la caja **Caption** y utilizar un nombre cualquiera para (**Name**).

La propiedad **Shortcut** permite activar un elemento del menú presionando una tecla o alguna combinación de teclas ya definidas.

CARACTERÍSTICAS DEL LENGUAJE VISUAL BASIC

Comentarios.- Un comentario es una frase que va precedida de una comilla simple (') y Visual Basic no ejecuta acción ninguna sobre ella. Ejemplo:

Suma= 0 **' Se inicia la variable suma**

Constantes.- Es un valor constante que no cambia durante la ejecución de un programa, por lo general se colocan al inicio de un programa. Ejemplo:

Const PI = 3.14159265358979

Variables.- Una Variable contiene un valor que puede modificarse a lo largo de la ejecución de la aplicación. Cada variable tiene atributos propios, como:
Nombre.- Es el nombre que utilizamos para referirnos a la variable en la aplicación.
Tipo.- Determina qué clase de valores puede almacenar la variable.
Ámbito.- Especifica en que parte de la aplicación la variable es conocida y por lo tanto puede utilizarse.

Restricciones para el nombre de variables

- El nombre de una variable tiene que comenzar por una letra, puede contener hasta 255 caracteres de longitud y debe ser el único dentro su ámbito.
- Los caracteres pueden ser letras, dígitos y los caracteres siguientes: *%,&,#,_,@,y $.*
- No se puede utilizar el punto ni otros caracteres o palabras reservadas ya que tienen un significado especial para Visual Basic.
- Una palabra reservada son sentencias predefinidas por Visual Basic como:
  For, Val, Hide, Caption, Long, And, etc.

Tipos de Variables

Tipo	Descripción	Rango
Integer	Entero (2 bytes)	-32768 a 32767
Long	Entero Largo (4 bytes)	-2147483648 a 2147483647
Single	Real simple (4 bytes)	-3.40E+38 a 340E+38
Double	Real doble (8 bytes)	-1.79D+308 a 1.79D+308
Currency	Número con punto Decimal fijo (8 bytes)	-922337203685477.58 a 922337203685477.58
String	Cadena de caracteres (1 byte por carácter)	64K cars. En 16 bits 2^{31} cars. En 32 bits
Byte	Carácter (1 byte)	0 a 255
Boolean	Boolean (2bytes)	True o False
Date	Fecha/Hora (8 bytes)	1/Enero/100 a 31/Diciembre/9999
Objet	Referencia a un objeto (4 bytes)	Cualquier referencia a Un objeto
Variant	16 (con Núm) o 22 (con Cars.) bytes + 1 byte por carácter.	Almacena datos de Cualquier tipo de los anteriores

Antes de utilizar una variable es necesario declarar su tipo utilizando la sentencia **Dim, Puplic**, **private** o **Static**. Cualquiera de estas declaraciones inicializa las variables numéricas con el valor de cero y las variables alfanuméricas con el carácter nulo. Ejemplos:

Dim Direccion As Integer

Static Img As Integer

Las sentencias anteriores declaran I como una variable entera , R como una variable real de precisión doble. Si una variable se utiliza y no se declara, se asume que es de tipo **Variant** que puede cambiar su tipo en el momento que se requiera. Si se sabe que una variable numérica nunca va a contener valores fraccionarios, es mejor declararla como entera, ya que con las variables enteras es más rápido el proceso.

Cuando una variable numérica de un tipo se asigna a otra variable numérica de un tipo diferente, Visual Basic realiza la conversión correspondiente.

Variables locales

Una *variable local* se reconoce solamente en el procedimiento en el que está definida. Fuera de este procedimiento, la variable no es conocida.

Para declarar una variable local a un procedimiento, coloque la sentencia **Dim** correspondiente dentro de este procedimiento. Ejemplo:

```
Sub Command1_Click()
   Dim x As Integer, y As Integer
   .
   .
   .
end Sub
```

Las variables locales "*x*" y "*y*" son reconocidas únicamente en el procedimiento del evento Command1_Clic() que se realiza cuando se hace un *click* en el botón1.

Variables globales

Una *variable global* es una variable declarada a nivel del módulo pero que puede ser accedida desde cualquier otro módulo.

Para hacer que una variable sea global o pública, hay que declararla como **Public** en la sección de declaraciones del módulo. Ejemplo:

```
Public Multimedia As New Mmedia
Public Archivo As String

Private Sub Play_Click()
  Archivo = App.Path & "\title.mid"
  Multimedia.mmOpen Archivo
  Multimedia.mmPlay
End Sub
```

En el ejemplo anterior podemos notar que las variables *Multimedia* y *Archivo* se utilizan en el procedimiento Play_Clic() para reproducir el archivo especificado.

Operadores

Dentro de una sentencia, los operadores realizan una operación específica de acuerdo al tipo que se refiera. En la tabla siguiente los operadores se encuentran ordenados de mayor a menor prioridad y los que aparecen en la misma línea tienen igual prioridad, solo que se llevan a cabo de izquierda a derecha. Las operaciones con paréntesis se evalúan primero, ejecutandose primero los paréntesis más internos.

OPERACIÓN	OPERADOR
Exponenciación	^
Cambio de signo	-
Multiplicación y división	*, /
División entera	\
Resto de una división entera	Mod
Suma y resta	+, -
Concatenar o enlazar (unir)	&
Igual, distinto, menor, mayor, ...	=, <>, <, >, <=, >=
Negación	Not
And	And
Or inclusiva	Or
Or exclusiva	Xor

Variables tipo String

Una variable tipo String permite almacenar una cadena de caracteres* a una variable. Ejemplo:

```
Dim cadena As String
```

Sentencias

Una sentencia es una línea de texto que indica una o más operaciones a realizar.

Una línea puede tener varias sentencias, separadas unas de otras por dos puntos. Ejemplo:

```
Im(0).BorderStyle = 1: Imagen.Picture = LoadPicture(App.Path & "\Imagen 0.bmp")
```

Una sentencia en Visual Basic se puede escribir en varias líneas utilizando el carácter de subrayado (_). Ejemplo:

```
archivo = App.Path & "\Imagen " _
 + Trim(Str(actual)) + ".bmp"
```

* Conjunto de letras símbolos o números.

SENTENCIAS DE CONTROL

Este tipo de estructuras , permiten tomar decisiones y realizar un proceso repetidas veces. Visual Basic dispone de las siguientes estructuras o sentencias de control:

- ***If ... Then... Else***
- ***Select Case***
- ***Do...Loop***
- ***For...Next***
- ***For Each... Next***

If ... Then... Else

Esta estructura permite ejecutar condicionalmente una o más sentencias y puede escribirse de la siguiente forma:

```
If Pelota.Left >= (Form1.ScaleWidth - Pelota.Width) Then
   Direccion = 1
ElseIf Pelota.Top <= 0 Then Direccion = 3
End If
```

Si la condición es verdadera, se ejecutan las sentencias que están a continuación de **Then** y si la condición es *False*, se ejecutan las sentencias que están a continuación de **Elseif**.

Select Case

Esta sentencia permite ejecutar una de varias acciones en función del valor de una expresión. Se utiliza cuando se compara una expresión con diferentes valores. Ejemplo:

```
Select Case Index
  Case 0
    Im(0).BorderStyle = 1
    imagen.Picture = LoadPicture(App.Path & "\imagen 0.bmp")
  Case 1
    Im(1).BorderStyle = 1
    imagen.Picture = LoadPicture(App.Path & "\imagen 1.bmp")
End Select
```

Cuando se ejecuta la sentencia Select Case, Visual Basic evalúa la expresión Index y busca el primer Case que incluya el valor evaluado, ejecutando a continuación el correspondiente bloque de sentencias. El control pasa a la siguiente sentencia hasta llegar a **End Select** que indica el fin de **Select Case**.

Do ... Loop

Un Loop (bucle) repite un conjunto de sentencias mientras se cumpla una condición. En el siguiente ejemplo, el valor de la variable x se incrementa en una unidad hasta que el valor de la variable h se mayor que Timer:

```
Do
  x = x + 1
  If x > 15 Then Exit Do
Loop While h > Timer
```

La sentencia Exit Do permite salir del bucle Do ... Loop antes de que éste finalice si el valor de x es mayor a 15.

For ... Next

La sentencia for permite ejecutar un conjunto de sentencias cierto número de veces, de acuerdo al valor de una variable. Ejemplo:

```
For i = 1 To 120
   Load Image1(i)
   If j > 20 then Exit For
Next
```

La sentencia **Exit For** permite salir del *bucle* **For .. Next** antes de que éste finalice cuando j es mayor a 20.

SENTENCIA With

With permite ejecutar una serie de sentencias sobre un mismo objeto de un tipo predefinido o de un tipo definido por el usuario. Su sintaxis es la siguiente:

```
With objeto
  [sentencias]
End With
```

Por ejemplo, si se tiene que cambiar las propiedades de una caja de texto, como el color de fondo, el color de letra y el tipo de fuente se debe hacer haciendo referencia una sola vez que hacerlo refiriéndose al objeto en cada propiedad que desea cambiar.

```
With Text1
   .BackColor = &HC0FFFF
   .ForeColor = &HF80FF
   .Font = "Arial"
End With
```

Si no utilizáramos la sentencia With, el código anterior se escribiría de la siguiente forma:

```
Text1.BackColor = &HC0FFFF
Text1.ForeColor = &HF80FF
Text1.Font = "Arial"
```

PROCEDIMIENTOS Y FUNCIONES

Cuando varios procedimientos producidos por sucesos ejecutan un mismo proceso se debe colocar el código en un **procedimiento estándar**, perteneciente a un módulo estándar, que será invocado desde cada procedimiento conducido por un suceso que necesite ejecutar dicho código. De esta forma se elimina la necesidad de duplicar código.

Un procedimiento estándar puede escribirse como procedimiento **Sub** o como función **Function**. Un procedimiento producido por un suceso siempre es un procedimiento **Sub**.

Un procedimiento **Property** permite crear propiedades para una clase y manipularlas. Los módulos de clases son la base de la ***programación orientada a objetos*** en Visual Basic. Los objetos de estas clases pueden incluir sus propias propiedades y métodos, pero no sus propios sucesos.

Cuando un procedimiento es llamado para su ejecución , Visual Basic busca ese procedimiento en el módulo donde nos encontramos. Si no lo encuentra, entonces continúa la búsqueda en el resto de los módulos de la aplicación.

Para crear un procedimiento estándar se abre la ventana de código correspondiente al módulo donde quiere definir el procedimiento , y después se ejecuta la orden *Insertar>Procedimiento.*

Para editar un procedimiento estándar existente, seleccione *"(General)"* en la lista de objetos de la ventana de código, y finalmente seleccione el procedimiento en la lista de procedimientos.

FUNCIONES (Function)

La sintaxis correspondiente a una función es la siguiente:

[**Private**|**Public**] [**Static**] **Function** nombre [(parámetros)] [**As** tipo]
  [sentencias]
  [nombre=expresión]
  [**Exit Function**]
  [sentencias]
  [nombre= expresión]
End Function

Donde:

nombre	Es el nombre de la función y su tipo determina el tipo de datos que devuelve la función.
parámetros	Son una lista de variables, que se corresponden con los argumentos que son pasados cuando es invocada la función.
expresión	Define el valor devuelto por la función. Este valor es almacenado en el propio nombre de la función. Si no se efectúa esta asignación, el resultado devuelto será cero si ésta es numérica o nulo ("") si la función es de caracteres.
Exit Function	Permite salir de la función y retornar a la sentencia inmediatamente a continuación de la que efectuó la llamada.
End Function	Indica el fin de la función y devuelve el control a la sentencia inmediatamente a continuación de la que efectuó la llamada.

Para llamar a una función se utiliza la siguiente sentencia.

[*variable*=] nombre [(*argumentos*)]

Donde los argumentos son una lista de constantes, variables o expresiones separadas por comas. El número de argumentos debe ser igual al número de parámetros de la función. Los tipos de los argumentos deben coincidir con los tipos de sus correspondientes parámetros.

PROCEDIMIENTOS (Sub)

La sintaxis que define un procedimiento es la siguiente:

[**Private**|**Public**] [**Static**] **Sub** nombre [(parámetros)]
  [sentencias]
  [**Exit Sub**]
  [sentencias]
End Sub

La llamada a un procedimiento se puede hacer de las dos formas siguientes:

a) **Call** *nombre* [(*argumentos*)]

b) *nombre* [*argumentos*]

Un procedimiento no puede ser utilizado en una expresión, ya que no retorna a través de su nombre un valor.

Para declarar procedimientos externos como los que se encuentran contenidos en una Biblioteca Dinámica (*DLL*), se utiliza la sentencia **Declare** dentro de la sección de declaraciones del módulo.

Para llamar a un procedimiento (*Sub o Funcion*) público de un formulario o de una clase desde cualquier otro módulo hay que utilizar la siguiente sintaxis:

Objeto.procedimiento(args)

donde objeto representa el formulario o el objeto de la clase al cual pertenece el procedimiento.

Para llamar a un procedimiento público de un módulo estándar desde cualquier otro módulo se puede hacer de las dos maneras siguientes:

a) módulo.procedimiento(args)

b) procedimiento(args)

donde *módulo* se refiere al nombre del módulo al que pertenece el procedimiento.

SUCESOS RELACIONADOS CON EL TECLADO

Cada vez que presionamos una tecla ocurren tres sucesos sobre el objeto actualmente seleccionado: **KeyDown, KeyPress** y **KeyUp.** El Suceso KeyPress se genera solamente cuando se introduce un carácter ASCII.

Los caracteres ASCII incluyen todos los caracteres imprimibles, las combinaciones CTRL+(A-Z) y otros caracteres estándar como el *Enter* (ASCII 13) y *BackSpace* (ASCII 8 – Retroceso). Para interceptar cualquier otra tecla o combinación de teclas que no produzcan un código ASCII, se utilizan los sucesos **KeyDown** y **KeyUp**.

Los sucesos KeyDown y KeyUp utilizan los argumentos *KeyCode* y *Shift*. El primer argumento contiene el código ANSI de la decla pulsada y el segundo argumento contiene un valor de 1,2 o 4, dependiendo de que se haya pulsado la tecla Shift, Ctrl o Alt, respectivamente (si no se ha pulsado una de estas teclas el valor de Shift es 0), o un valor suma de los anteriores si se han pulsado dos o más teclas simultáneamente; por ejemplo, al pulsar Shift+Alt el argumento toma el valor de 5.

Los sucesos KeyDown y KeyUp no hacen distinción entre minúsculas y mayúsculas. Para hacer esta distinción tenemos que ver si el argumento Shift es 0 (minúscula) o 1 (mayúscula). Por ejemplo, si pulsamos las teclas "Shift"+"a", el valor del argumento *KeyCode* es 65 y el valor del argumento *Shift* es 1.

Conversión de texto a formato numérico

El contenido de una caja de texto es una cadena de caracteres y, por tanto no tiene un valor numérico. Para realizar operaciones aritméticas es necesario que este contenido sea convertido a un formato numérico.

Para convertir una cadena de caracteres en un número, se utiliza la función **Val.** Por ejemplo:

```
GradosFahr = Val(GradosC.Text) * 9 / 5 + 32
```

En el ejemplo anterior, *GradosC.Text* representa el texto de la caja nombrada GradosC. La función Val convierte este texto a un número y después de realizar las operaciones indicadas almacena el resultado en la variable GradosFahr de tipo **Double.**

Conversión números a formato de texto

Si queremos presentar en una caja de texto el valor numérico obtenido como resultado de una serie de operaciones aritméticas, la forma más fácil de hacerlo es utilizar la función **Str** para convertirlo en una cadena de caracteres y asignar esta cadena a la propiedad Text de la caja de texto. Ejemplo,

```
GradosF.text = Str(GradosFahr)
```

La función **Format** realiza la misma operación que Str, y además permite dar formato a la salida de acuerdo con un patrón especificado. Por ejemplo:

```
GradosF.text = Format(GradosFahr, "#,##0.00")
```

El símbolo # representa un dígito cualquiera, excepto los ceros no significativos. El símbolo 0 representa un dígito cualquiera, incluyendo los ceros no significativos y además redondea el resultado. El punto indica la posición del punto decimal y la coma es el separador de los miles.

Visualización de la Fecha y la Hora

La función **Now** da como resultado la fecha y la hora actuales. Para visualizar este fecha y hora de acuerdo con diversos patrones se tiene que utilizar la función **Format** con los símbolos d(día, m(mes), y(año), h(horas, m(minutos) y s(segundos). Ejemplo:

Patrón	Format(Now,"Patron")
d-m-yy	19-8-99
dd/mm/yy	19/08/99
dd-mmmm-yyyy	19-dic-1999
d-mmm m-yyyy	19-dic 12 1999
dddd dd – mmm m – yyyy	Domingo 19 – dic 12 –1999
h:m:s, d-mmm-yy	22:24:50, 19-dic-1999
hh:mm:ss AM/PM, dd-mmm-yyyy	10:24:50 PM, 19-dic-1999

En el siguiente ejemplo, la función **Now** Proporciona la fecha y **Format** especifica el formato con el que se ha de visualizar. El resultado hay que almacenarlo en la propiedad Text del objeto Fecha; esto es, en *Fecha.Text*.

Fecha.Text = **Format**(*Now*, "dddd dd-mmm-yyyy")

Temporizador

Un temporizador es un control de Visual Basic que responde a intervalos regulares de tiempo. Es útil para realizar procesos en segundo plano. Esto quiere decir que en el procedimiento asociado con el mismo, especificaremos las acciones que deseemos que se ejecuten cada vez que transcurra un intervalo de tiempo.

Cada temporizador tiene una propiedad **Interval** que especifica el intervalo de tiempo en milisegundos que tiene que transcurrir para que el procedimiento asociado se ejecute, independientemente del usuario. El valor de la propiedad **Interval** puede oscilar entre 0 y 64767 (0 a 64.8 segundos).

El sistema genera 18 tics de relog por segundo. Aunque el valor de la propiedad Interval se exprese en milisegundos, la precisión no puede ser mayor de 1000/18 milésimas de segundo.

Si nuestra aplicación está realizando una tarea que mantiene ocupados los recursos del ordenador por un espacio largo de tiempo, tal como un bucle largo, cálculos intensivos, acceso a los puertos, etc., puede ser que la aplicación no responda de acuerdo con los intervalos de tiempo programados.

Botón de opción

Un botón de opción es un control que indica si una determinada opción está activada o desactivada. Cada botón de opción es independiente de los demás, ya que cada uno de ellos tiene su propio nombre (Propiedad **Name**). El número de opciones representadas de esta forma puede ser cualquiera, y de ellas el usuario sólo puede seleccionar una cada vez.

Si durante la ejecución se hace clic sobre un botón de opción, la opción queda seleccionada. La selección de una opción de este tipo provoca que si hay otra opción del mismo grupo actualmente seleccionada pase a no estarlo.

Para saber si una determinada opción está seleccionada, hay que verificar el valor de su propiedad **Value**. Este valor puede ser falso (**False**), el botón aparece vacío, o verdadero (**True**), el botón aparece un ◉ .

Cuando una opción se pone a valor **True**, se da el suceso **Click**.

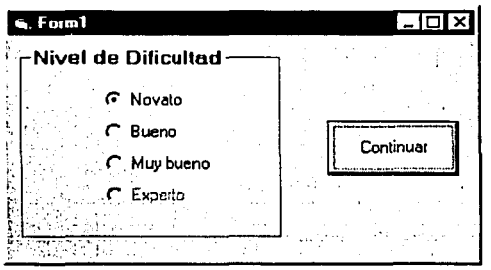
Cuando una de estas opciones está inhabilitada, la etiqueta asociada aparece en gris. Esto se consigue poniendo su propiedad **Enabled** al valor **False**.

Ejemplo:

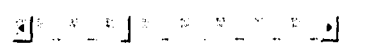
Programa:

```
Private Sub Form_Load()
  Opción(3).Enabled = False
End Sub

Private Sub Command1_Click()
  If Opción(0).Value = True Then
    MsgBox "Seleccionaste Novato"
  ElseIf Opción(1).Value = True Then
    MsgBox "Seleccionaste Bueno"
  ElseIf Opción(2).Value = True Then
    MsgBox "Seleccionaste Muy Bueno"
  ElseIf Opción(3).Value = True Then
    MsgBox "Seleccionaste Experto"
  End If
End Sub
```

Ejecución del programa:



Barras de desplazamiento



Las barras de desplazamiento son utilizadas en cajas de texto y ventanas para desplazar la información hacia abajo o hacia arriba de la ventana, o hacia la izquierda o hacia la derecha de la ventana. También pueden utilizarse como controles de entrada.

Una barra de desplazamiento representa un valor entero. Cada barra de desplazamiento tiene un cuadrado que se desplaza a lo largo de la misma para fijar un valor. La posición más a la izquierda o más arriba se corresponde con el valor mínimo, la posición más a la derecha o más abajo se corresponde con el valor máximo, y cualquier otra posición es un valor entre estos dos.

El valor mínimo se especifica mediante la propiedad **Min**, y el valor máximo mediante la propiedad **Max**. Cualquier valor comprendida entre el mínimo y el máximo especificados depende de la posición del cuadrado de desplazamiento de la barra y viene dado por la propiedad **value**.

Cuando se hace clic encima o debajo del cuadrado de desplazamiento, éste se desplaza una cantidad fija, negativa o positiva, con respecto a la posición actual, que está fijada por la propiedad *Large Change*. Igualmente, cuando se hace clic en alguna de las flechas de los extremos de la barra, el cuadrado de desplazamiento se mueve una cantidad fija, negativa o positiva, con respecto a la posición actual, que está fijada por la propiedad *SmallChange*.

Cada vez que se mueve el cuadro de desplazamiento a lo largo de la barra se produce el suceso *Change*. El suceso *Change* ocurre después de que el cuadrado de desplazamiento ha sido movido; mientras es movido ocurre el suceso *Scroll* (este suceso sólo ocurre cuando el cuadrado se arrastra, no ocurre si se mueve haciendo clic sobre la barra o sobre las flechas de los extremos).

Colores

Durante el diseño de una aplicación, se le puede dar color fácilmente a los formularios, a los controles y a los gráficos utilizando la paleta de colores de Visual Basic. La paleta de colores contiene 48 colores fijos y 16 colores más que el usuario puede definir y añadir a la paleta.

Para poner color a un objeto durante el diseño seleccione la propiedad que define el color de fondo del objeto (**BackColor**) o la propiedad que define el color del primer plano (**ForeColor**).

También se puede modificar el color de cualquier objeto durante la ejecución de una aplicación, utilizando la función **RGB** o la función **QBColor.**

La función **RGB** tiene tres parámetros enteros en el rango de 0 a 255 que especifican el nivel de color rojo, verde y azul, respectivamente. Por ejemplo, si queremos cambiar el color de una caja de texto que contiene un valor numérico para diferenciar los valores positivos (color azul) y negativos (color rojo), utilizaremos el siguiente procedimiento:

```
Private Sub Resultado_Change( )
  If Val(Resultado.Text) >= 0 then
     Resultado.ForeColor = RGB(   0,   0,255)  'color azul
  Else
     Resultado.ForeColor = RGB(255,   0,   0)  'color rojo
  End If
End sub
```

La sentencia **If** analiza el valor de la caja de texto Resultado; si es positivo, asigna el color azul al primer plano (al texto), y si es negativo, le asigna el color rojo.

Visual Basic dispone también de la función **QBColor** y del método **Point**, que están directamente relacionados con la función **RGB**.

La función **QBColor** tiene un único argumento entero en el rango de 0 a 15 que se corresponde con un color estándar de acuerdo a la siguiente tabla:

0.-	Negro
1.-	Azul
2.-	Verde
3.-	Aguamarina
4.-	Rojo
5.-	Fucsia
6.-	Amarillo
7.-	Blanco
8.-	Gris
9.-	Azul Brillante
10.-	Verde Brillante
11.-	Aguamarina Brillante
12.-	Rojo Brillante
13.-	Fucsia Brillante
14.-	Amarillo Brillante
15.-	Blanco Brillante

Devuelve como resultado un entero largo correspondiente al color **RGB** equivalente. Ejemplo:

Form.BackColor = **QBColor** (1) 'color azul

El método **Point** tiene dos argumentos de tipo real de precisión simple que especifican las coordenadas de un punto de un formulario o de una imagen y devuelve como resultado un entero largo correspondiente al color RGB que tiene el punto especificado. Ejemplo:

```
Private Sub Form_Click ( )
   Dim ColorForm As Long, ColorPict As Long
   ColorForm = Point (1440, 1440)
   ColorPict = Picture1.Point
(1440,1440)
End Sub
```

El valor de 1440 es expresado en *twips*[*]. Si las coordenadas especificadas quedan fuera del objeto, el valor retornado es –1.

[*] Unidad de medida para desplazar y dimensionar un objeto.

Propiedades de las Fuentes

Las propiedades de una fuente determinan las características visuales del texto, tales como su aspecto (*Times New Roman, Curier New, etc.*), tamaño (*10,11,12, etc.)* y estilo (**negrita**, *cursiva*, subrayada, etc.).

Propiedad	Tipo	Descripción
FontName	String	Especifica el nombre de la fuente.
FontSize	Integer	Especifica el tamaño en puntos de la fuente.
FontBold	Boolean	Si es True, el texto se presenta en **negrita**.
FondItalic	Boolean	Si es True, el texto se presenta en *cursiva*.
FondStrikethru	Boolean	Si es True, el texto se presenta en ~~tachado~~.
FontUnderline	Boolean	Si es True, el texto se presenta en subrayado.

El tamaño de una fuente es la distancia que hay entre la parte superior de una letra minúscula ascendente, y la inferior de una descendente. El tamaño se mide en **puntos**; una pulgada tiene 72 puntos y un punto tiene 20 **twips**.

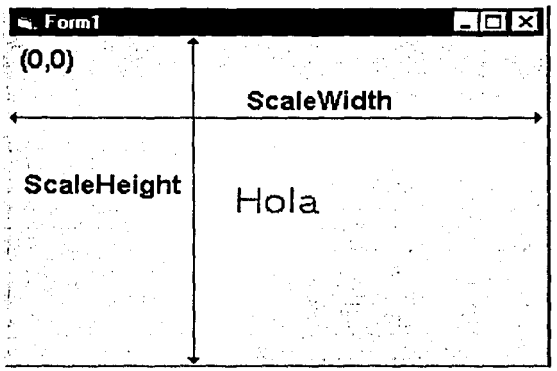
Para escribir texto en un formulario, en una caja de imagen o en la impresora , utilizaremos el método **Print**.

Si no se especifica el objeto sobre el cual queremos escribir, por defecto se escribira sobre el formulario. Ejemplo:

```
Print Mensaje            'escribe Mensaje en el formulario actual
Imagen1.Print Mensaje    'escribe Mensaje en Imagen1
```

Es necesario especificar también su posición mediante las propiedades **CurrentX** y **CurrentY**, que por defecto tienen valor 0. Ejemplo:

```
Font.Size = 18
CX = ScaleWidth / 2 + ScaleLeft
Cy = ScaleHeight / 2 + ScaleTop
Cls
Mensaje = "Hola"
CurrentX = CX – TextWidth(Mensaje) / 2
CurrentY = Cy – TextHeight(Mensaje) / 2
ForeColor = RGB( 0,255, 0)
Print Mensaje
```



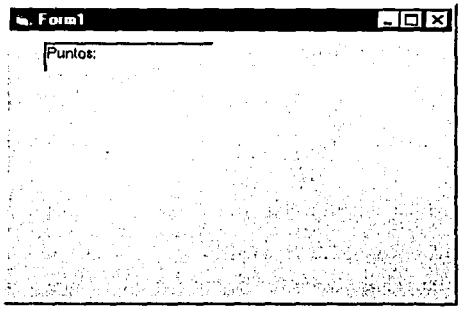
Si se utiliza el procedimiento *Form_Load* para dibujar gráficos (*líneas o texto*) en un formulario o en una caja de imagen, es necesario poner la propiedad **AutoRedraw** de ese objeto a valor **True** para que se visualice el gráfico.

Para visualizar un texto en una caja de imagen siga los siguientes pasos:

- Inicie una nueva aplicación
- Asigne al formulario el título Texto
- Agregue una caja de imagen (**Picture Box**)
- Asigne el nombre de Caja de texto a la caja de imagen
- Asigne la propiedad **AutoRedraw** de ese objeto a valor **True**
- Escriba el siguiente procedimiento:

```
Private Texto As String

Private Sub Form_Load( )
   Texto = "Score"
   CajaTexto.Print Texto
End Sub
```

- Guarde la aplicación
- Ejecute la aplicación y obtendrá un formulario con el mensaje en la caja de imagen como el siguiente:

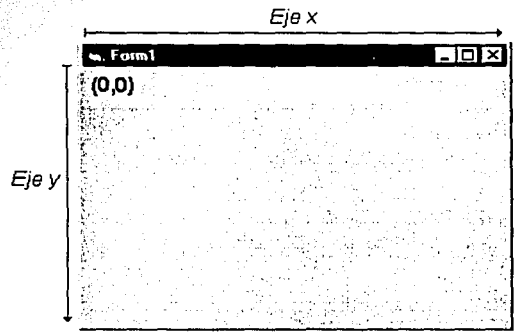


GRÁFICOS EN VISUAL BASIC

Visual Basic proporciona varias maneras de producir efectos gráficos de una forma sencilla. Se pueden mover objetos por la pantalla, hacerlos aparecer y desaparecer, modificar su tamaño y realizar todo tipo de gráficos que se nos puedan ocurrir combinando figuras básicas, como líneas, cuadros, círculos, etc.

Es importante saber utilizar el sistema de coordenadas para definir la posición de los formularios y controles de la aplicación.

Un punto **P** queda determinado por las coordenadas **P**(*x,y*). Donde el valor de *x* es la posición del punto a lo largo del *eje x*. El valor de *y* es la posición del punto a lo largo del *eje y*. Si se omiten los valores de *x o y* su valor es cero y se localizan en la parte superior Izquierda del área del dibujo. Ejemplo:



Por defecto Visual Basic utiliza los **twips** como medida para desplazar y dimensionar un objeto, así como para los métodos gráficos (*Una pulgada = 1440 twips* y *un centímetro = 567 twips*).

Cuando se mueve o se modifica el tamaño de un control, se utiliza el sistema de coordenadas del formulario, de la caja de imagen, del marco o del objeto sobre el que se este trabajando.

Controles Gráficos

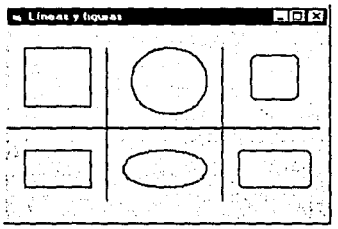
Existen controles diseñados para crear efectos gráficos en una aplicación. Como son los siguientes:

Líneas

Este control permite dibujar líneas durante el diseño. Un objeto de esta clase tiene propiedades que permiten controlar su posición, longitud, color, estilo, etc.

Figuras

El control figuras permite dibujar rectángulos, cuadrados (con sus esquinas redondeadas o sin redondear), círculos y elipses. Un objeto de esta clase tiene propiedades que permiten controlar su posición, tamaño, color de borde, color del interior, recubrimiento, tipo de figura, estilo de borde, etc. La sigiente figura muestra un formulario con diferentes tipos de figuras y líneas.



Imagen

Un control imagen es un área rectangular dentro de la cual se pueden cargar ficheros de imágenes de tipo .BMP (*mapa de bits*), .ICO (*iconos*) y .WMF(*metaarchivos de Windows*). La imagen puede ser puesta directamente sobre el formulario, dentro de la caja de imagen o utilizando el control gráfico de imagen.

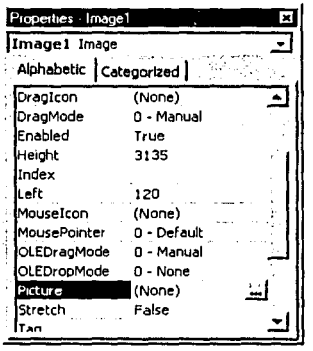
Agregar una Imagen a un formulario mediante el diseño

Para añadir una imagen durante el diseño, siga los siguientes pasos:

◈ Añada, si es necesario, un control imagen o caja de imagen al formulario.

◈ Seleccione el formulario, el control de imagen o el control imagen dando clic sobre el mismo.

◈ Seleccione la opción **Picture** en la lista de propiedades y haga clic en el botón [...].



◈ En la caja de dialogo visualizada, elija el fichero **.BMP**, **.ICO** o **.WMF** que desee o en su defecto que muestre todos los archivos gráficos.

◈ Seleccione el directorio donde se encuentran los archivos gráficos y haga un DobleClic sobre el archivo que desea abrir.

La imagen queda dentro del formulario o en el control imagen detrás de cualquier control que se hubiera colocado allí, o dentro de la caja de imagen.

Para eliminar una imagen, ponga de nuevo la propiedad **Picture** a valor *"(None)"*. Para ello, seleccione el formulario o el control, elija la propiedad Picture en la ventana de propiedades, haga un doble clic en la palabra que se visualiza y pulse la tecla ***Supr*** (*Del*).

Las imágenes cargadas durante el diseño son guardadas y cargadas con la aplicación y al crear el fichero .EXE ya no son necesarias las imágenes porque quedaron integradas dentro de este archivo.

Añadir una imagen mediante código

Para añadir una imagen mediante la programación de un proyecto, utilice la función **LoadPicture**. Esta función permite asignar un fichero .BMP, .ICO o .WMF a la propiedad Picture de un formulario, de una caja de imagen o de un control imagen. Ejemplo:

```
Private Sub Form_Load ( )
  Form1.Picture = LoadPicture ("c:\juego3\fondo.bmp")
  Imagen1.Picture = LoadPicture ("c:\juego3\Ryu.bmp")
End Sub
```

Al ejecutar este programa es necesario contar con los archivos gráficos dentro del directorio *c:\juego3*.

La primera instrucción asigna la imagen FONDO.BMP al formulario Form1. La segunda instrucción asigna la imagen RYU.BMP a la caja de imagen llamada Imagen1.

Para la caja de imagen adapte automáticamente el tamaño de la imagen es necesario que la propiedad **AutoSize** sea igual a **True**.

Un control imagen no tiene propiedad AutoSize, pero automáticamente sus dimensiones se ajustan al tamaño de la imagen cargada. Para que la imagen cargada se adapte al tamaño del control, la propiedad **Stretch** debe tener el valor **True**.

Las imágenes que se cargan mediante código utilizando la función **LoadPicture** no se guardan al crear el fichero **.EXE** y será necesario acompañar este archivo de los ficheros de imágenes utilizados por nuestra aplicación.

Es posible asignar una imagen a un control desde otro control. Si en el destino ya hay una imagen, esta es sustituida por la nueva imagen. Ejemplo:

```
Imagen2.picture = Imagen1.Picture
```

Para borrar una imagen se utiliza la función **LoadPicture( )**. Ejemplo:

```
Imagen2.picture = LoadPicture( )
```

Movimiento de Controles

Durante la ejecución de una aplicación se puede mover cualquier control a otra posición, modificar su tamaño, ocultarlo y añadir nuevos controles.

Para mover un control es necesario utilizar el método **Move** indicando la nueva posición: izquierda (**Left**) y superior (**Top**). Ejemplo:

```
Nave.Move Nave.Left + sentido, Nave.Top
```

En el ejemplo anterior la caja de imagen llamada nave se mueve ciertas coordenadas a la izquierda o derecha dependiendo de el valor de la variable sentido y la coordenada vertical se mantiene constante.

Si queremos modificar su tamaño tenemos que especificar los nuevos valores para el ancho (**Width**) y alto (**Height**). (Ver programa # 2 del capítulo 4).

Manejo de Bases de Datos con Data Control

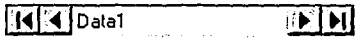
La mayoría de los programas utilizan información que se almacena y se lee desde una base de datos. Para acceder a esta información Visual Basic utiliza el control de datos *(Data Control);* este control permite visualizar, editar y actualizar información de varios tipos de bases de datos, como *Microsoft Access, Microsoft FoxPro, Dbase*, entre otros.

Los datos de una base de datos están divididos en registros y estos a su vez en campos, cada campo puede contener información de tipo numérico, cadena, tipo fecha, entre otros. Los registros serían los equivalentes a las filas en una hoja de Excell y los campos serían semejantes a las columnas.

El acceso a bases de datos permite seleccionar información en forma rápida y precisa, realizar filtros únicamente de la información que se requiere, ordenar los datos de acuerdo a uno o varios campos, realizar reportes y enviarlos a impresión.

Para acceder a una base de datos es necesario añadir el control a nuestro formulario y especificar la base de datos de la que deseamos obtener información.

La información obtenida será visualizada en otros controles, como cajas de texto, entrelazados al control de datos. En la siguiente figura se muestra los siguientes botones de control:



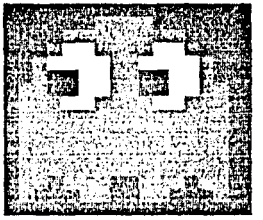
Botón	Función	Método
	Se posiciona en el primer registro	*Data1.Recordset.**MoveFirst***
	Se posiciona en el registro anterior	*Data1.Recordset.**MovePrevious***
	Se posiciona en el siguiente registro	*Data1.Recordset.**MoveNext***
	Se posiciona en el último registro	*Data1.Recordset.**MoveLast***

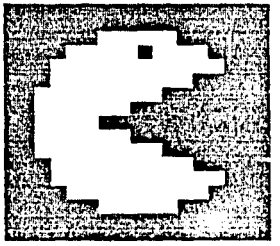
El control de datos puede utilizarse para crear aplicaciones de bases de datos sencillas sin escribir nada de código, para aplicaciones más complejas será necesario el soporte de código de Visual Basic. Si desea hacer un ejemplo sobre este control siga los pasos del programa 20 del siguiente capítulo.

CAPÍTULO 4

Códigos de Programas

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)

                                        If KeyCode = vbKeyDown Then

End Sub

Private Sub Timer1_Timer()

End Sub
```





CÓDIGOS DE PROGRAMAS

En este capítulo se muestran los códigos fuente de cada una de las 20 funciones básicas para diseñar juegos y en general cualquier tipo de programa. Se pretende que el usuario obtenga un panorama general de cada función y sea capaz de crear sus propias aplicaciones utilizando algunas de estas funciones.

Cabe mencionar que algunos programas no son tan fáciles de realizar y se requiere tener conocimientos previos sobre el manejo de Windows y algunos antecedentes básicos de programación; sin embargo, la mayoría de las funciones descritas en este capítulo son muy sencillas y tienen poco código de programación; pero se requiere que el usuario ponga todo su empeño y dedicación en el estudio de estas funciones.

Anexo a esta tesis se encuentra un CD que contiene una demostración de cada una de estas funciones; pero antes será necesario configurar la pantalla del monitor a una resolución de 800x600 píxeles. Si se desea ver el código de alguna de las funciones será necesario que se tenga instalada una versión de Visual Basic 5.0 o posterior y copiar el directorio D:\PROGRAMAS a la unidad "C:\" de su disco duro para leer y ejecutar los programas de cada subdirectorio.

Espero que te sean útiles estas funciones y puedes enviar tus dudas, comentarios y sugerencias a mi correo electrónico: eliasgs@servidor.unam.mx

PROGRAMA # 1
INCLUIR UNA IMAGEN EN UN FORMULARIO

Figura 4.1.0. Imagen dentro de un formulario.

Para realizar este primer programa, presione "**Doble Click**" dentro de un nuevo formulario y escriba el siguiente listado:

Código del Programa

```
Private Sub Form_Load ( )
  Directorio = "c:\programas\programa01\"
  Form1.WindowState = 2
  Form1.Picture = LoadPicture(Directorio + "King of Figter.bmp")
End Sub
```

DESCRIPCIÓN:
Directorio.- Es el lugar en donde se encuentra almacenada nuestra imagen.
Form1.WindowState = 2.- Maximiza nuestro formulario llamado Form1.
Form1.Picture = LoadPicture(Directorio + "King of Figter.bmp") .- Muestra la Imagen dentro de nuestro formulario. (*Ver figura 4.1.9*).

Otra forma sencilla de asignar una imagen a un formulario es asignando la ruta y el nombre de la imagen directamente desde la ventana de propiedades del formulario en su propiedad **Picture**.

PROGRAMA # 2
MOVIMIENTO DE UNA IMAGEN

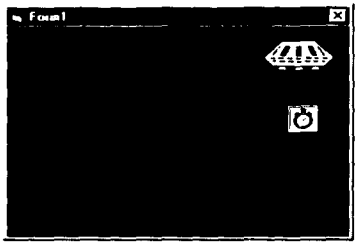


Figura 4.1.1. Movimiento de una Imagen.

Para dar movimiento a la imagen de la nave espacial de la figura anterior utilizamos la función **Move**, esta función desplaza la caja de texto que contiene la imagen de izquierda a derecha 100 *twips* de acuerdo al sentido. Al detectar la nave las coordenadas horizontal igual a100 o igual a 8000, se cambia el sentido del movimiento.

El programa utiliza los siguientes objetos:

OBJETO	PROPIEDAD	VALOR
Form1	**Name** **Backcolor** **WindowState**	Form1 &H00000000& Maximized
PictureBox	**Name** **Picture**	Nave **Nave.Bmp**
Timer	**Name** **Interval**	Timer1 100

Código del Programa

```
Dim sentido As Integer
Private Sub Form_Load( )
  Nave.Left = 7000
  Nave.Top = 500
  sentido = -100
End Sub
Private Sub Timer1_Timer( )
  Nave.Move Nave.Left + sentido, Nave.Top
  If Nave.Left < 100 Or Nave.Left > 8000 Then
    sentido = -sentido
  End If
End Sub
```

PROGRAMA # 3
ANIMACIÓN DE IMÁGENES

Para diseñar este programa es necesario crear un formulario con las siguientes propiedades:

Name = Form1 *(Nombre del formulario = Form1)*
Backcolor = &H00FFFFFF& (*Color de fondo = Blanco*)
WindowState= Maximized (*Estado de la ventana = Maximizado*)

Dentro de este Formulario se deben agregar los siguientes objetos con sus respectivas propiedades:

Objeto	Propiedad	Valor
Caja de Imagen1	Name	Ryo
	Index	0
	Picture	Ryo sin presionar 1
	Visible	False
Caja de Imagen2	Name	Ryo
	Index	1
	Picture	Ryo sin presionar 2
	Visible	False
Caja de Imagen3	Name	Ryo
	Index	2
	Picture	Ryo sin presionar 3
	Visible	False
Caja de Imagen4	Name	Ryo_Pic
	Index	(Vacío)
	Picture	(Vacío)
	Visible	***True***
Timer	Name	Timer1
	Interval	100

Al terminar de cargar los objetos dentro del formulario quedara como el de la figura 4.1.3.

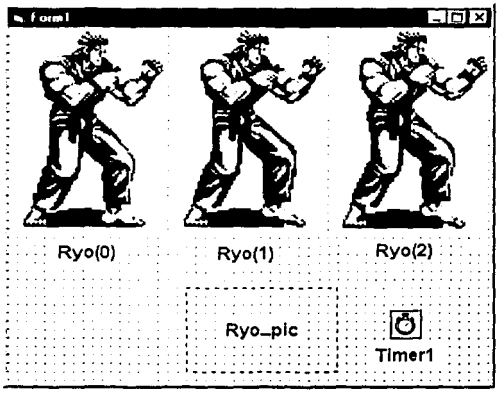


Figura 4.1.3. Movimiento de una Imagen.

Código del Programa

```
Private Sub Form_Load()
   Ryo_pic.Left = 1000
   Ryo_pic.Top = 3600
End Sub
```



```
Private Sub Timer1_Timer()
Static Img As Integer
   If Img = 2 Then Img = -1
   Img = Img + 1
   Ryo_pic.Picture = Ryo(Img).Picture
End Sub
```

Al iniciar el programa, se sitúa la caja de imagen *Ryo.pic* dentro del formulario en las coordenadas (1000,3600). En la Variable *Img* se almacena el valor del arreglo de las Imágenes *Ryo* (0, 1 o 2) que en un principio tiene valor = 0. Si el valor de la variable *Img* = 2, la variable toma el valor de –1 y después se incrementa una unidad y finalmente el programa asigna a la caja de imagen *Ryo_pic* la imagen del arreglo *Ryo* correspondiente.

Al presentarse estas imágenes dentro del *Timer1* a intervalos de 100 milisegundos (*una a la vez*), da la impresión de que las imágenes *Ryo(0), Ryo(1)* o *Ryo(2)* se mueven por si solas, dando el realismo de un peleador esperando a su oponente.

PROGRAMA # 4
REBOTE DE UNA PELOTA

Este pequeño programa simula el rebote de una pelota dentro de nuestra ventana (*Formulario*) y utiliza una caja de imagen (*PictureBox*) con el nombre de pelota y un reloj (*Timer1*) que realiza la función de movimiento a intervalos de 10 milisegundos (Ver figura 4.1.4).

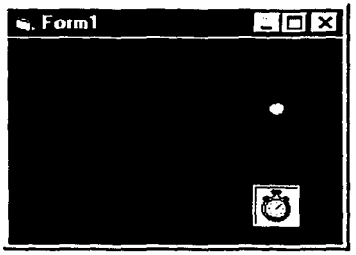


Figura 4.1.4. Simulación de rebote de una pelota.

Al iniciar el programa, la *variable K* se le asigna el valor de 50, que es el número de twips que se desplaza la pelota (dependiendo del valor de la dirección), esta se determina inicialmente por un número entero en forma aleatoria (*de 1 a 4*) por la función "*RND*".

Si la dirección =1, la pelota se mueve hacia la izquierda y hacia arriba y si "choca" con el lado izquierdo de la pantalla, cambiara la dirección = 2 (*Movimiento de la pelota a la derecha y hacia arriba*) y si choca en la parte de arriba de la ventana, cambiará la dirección = 4 (*Movimiento hacia abajo y a la izquierda*).

En forma similar se moverá la pelota para los demás casos y cambiando de dirección de acuerdo a los límites de la ventana.

Propiedades de los objetos

OBJETO	PROPIEDAD	VALOR
Form1	**Name** **Backcolor** **WindowState**	Form1 &H00000000& Maximized
PictureBox	**Name** **Picture**	Pelota **Pelota.Bmp**
Timer	**Name** **Interval**	Timer1 10

Código del Programa

```
Dim k As Integer
Dim Direccion As Integer

Private Sub Form_Load()
  k = 50
  Direccion = 4 * Rnd + 1
  Timer1.Interval = 10
End Sub

Private Sub Timer1_Timer()
 Select Case Direccion
   Case 1 'Mover pelota izquierda y arriba
     Pelota.Move Pelota.Left - k, Pelota.Top - k
     If Pelota.Left <= 0 Then Direccion = 2
     If Pelota.Top <= 0 Then Direccion = 4
   Case 2 'Mover pelota derecha y arriba
     Pelota.Move Pelota.Left + k, Pelota.Top - k
     If Pelota.Left >= (Form1.ScaleWidth - Pelota.Width) Then
       Direccion = 1
     ElseIf Pelota.Top <= 0 Then Direccion = 3
     End If
   Case 3 'Mover pelota izquierda y arriba
     Pelota.Move Pelota.Left + k, Pelota.Top + k
     If Pelota.Left >= (Form1.ScaleWidth - Pelota.Width) Then
       Direccion = 4
     ElseIf Pelota.Top >= (Form1.ScaleHeight - Pelota.Height) Then
       Direccion = 2
     End If
   Case 4 'Mover pelota izquierda y abajo
     Pelota.Move Pelota.Left - k, Pelota.Top + k
     If Pelota.Left <= 0 Then Direccion = 3
     If Pelota.Top >= (Form1.ScaleHeight - Pelota.Height) Then Direccion = 1
 End Select
End Sub
```

PROGRAMA # 5
MOVIMIENTO DE UNA BARRA

En este programa utilizamos un formulario con el fondo negro (*Form1*.Backcolor = &H00000000&), ya que la imagen de la barra no se encuentra bien delineada y contiene un poco de negro en sus bordes.

Otra propiedad importante del formulario es *KeyPreview* = *True* para que detecte la última tecla presionada y puede ser la tecla izquierda *(vbKeyLeft)* o la derecha *(vbKeyRight)*. Al iniciar el programa la posición actual de la barra cambia mas o menos al centro de la pantalla en su coordenada "*X*" y hacia abajo en su coordenada "*Y*". Al presionar una de estas dos teclas, el programa mueve la barra 150 *twips* hacia el lado correspondiente, siempre y cuando no rebase los límites establecidos (20-8300). Ver figura 4.1.5.

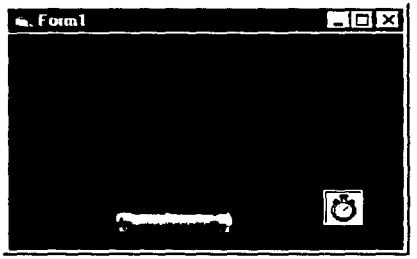


Figura 4.1.5. Uso de las teclas de navegación.

Propiedades de los objetos

OBJETO	PROPIEDAD	VALOR
Form1	Name Backcolor WindowState KeyPreview	Form1 &H00000000& Maximized True
PictureBox	Name Picture	Barra Barra.Bmp
Timer	Name Interval	Timer1 100

Código del Programa

```
Dim izquierda As Boolean, derecha As Boolean
Const Salto = 150

Private Sub Form_Load()
   Form1.KeyPreview = True
   ChDir App.Path
   Barra.Left = 2500
   Barra.Top = 7500
End Sub


Private Sub Timer1_Timer()
      If derecha Then
        If Barra.Left < 8300 Then Barra.Move Barra.Left + Salto
      End If
      If izquierda Then
        If Barra.Left > 20 Then Barra.Move Barra.Left - Salto
      End If
End Sub

Private Sub Form_Keydown(KeyCode As Integer, Shift As Integer)
  If KeyCode = vbKeyLeft Then izquierda = True
  If KeyCode = vbKeyRight Then derecha = True
End Sub

Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)
   If KeyCode = vbKeyLeft Then izquierda = False
  If KeyCode = vbKeyRight Then derecha = False
End Sub
```

PROGRAMA # 6
SELECCIÓN DE JUGADORES

Para realizar este programa se utilizan 8 cuadros de imagen en un arreglo de imágenes (*Im(0) a Im7*), cada uno de ellos contiene una imagen de mapa de bits (*BMP*) de un jugador en pequeño; también necesita un cuadro de imagen llamado *imagen* que contiene la imagen en grande del jugador seleccionado, por último utilizamos un reloj (Timer1) para detectar si se presiona una tecla en un periodo de 100 milisegundos.

Al iniciar el programa se cambia la propiedad del estilo en el borde de la imagen actual (***Im(actual).BorderStyle = 1***), en este caso *Im(0)* y también llama la imagen en grande del jugador actual para la selección del jugador. Ver figura 4.1.6.

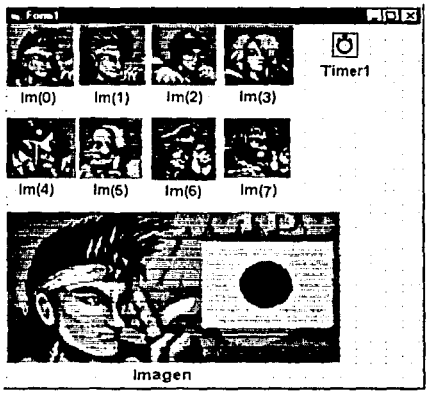


Figura 4.1.6. Uso de las teclas de navegación y la función MouseMove.

El programa utiliza también las funciones para la detección del teclado (*Keydown y KeyUp*). Al presionar la flecha derecha, aumenta en una unidad el valor de la variable "*actual*" de la imagen actual, siempre y cuando no tenga el valor de 7, ya que en este caso el valor de la variable actual será igual a cero. Algo similar pasa cuando se presiona la flecha Izquierda, solo que en lugar de aumentar la variable "*actual*", disminuye y cuando tiene el valor de cero cambia su valor a siete.

Para el manejo del ratón, utilizamos la función *Im_MouseMove* que detecta cuando el ***evento*** del movimiento del ratón sobre alguna imágenes (*Im(0)-Im(7)),* desactiva la propiedad del borde de la imagen actual y cambia el borde de la imagen seleccionada y la imagen en grande.

Código del Programa

```
Dim izquierda As Boolean, derecha As Boolean
Dim actual As Single, i As Single
Dim archivo As String
Private Sub Form_Load()
   imagen.Left = 500
   imagen.Top = 3500
   actual = 0
   Im(actual).BorderStyle = 1
   archivo = App.Path & "\imagen " & Trim(Str(actual)) & ".bmp"
   imagen.Picture = LoadPicture(archivo)
   KeyPreview = True
End Sub
Private Sub Im_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
  For i = 0 To 7: Im(i).BorderStyle = 0: Next i
  Select Case Index
    Case 0
      Im(0).BorderStyle = 1: imagen.Picture = LoadPicture(App.Path & "\imagen 0.bmp")
    Case 1
      Im(1).BorderStyle = 1: imagen.Picture = LoadPicture(App.Path & "\imagen 1.bmp")
    Case 2
      Im(2).BorderStyle = 1: imagen.Picture = LoadPicture(App.Path & "\imagen 2.bmp")
    Case 3
      Im(3).BorderStyle = 1: imagen.Picture = LoadPicture(App.Path & "\imagen 3.bmp")
    Case 4
      Im(4).BorderStyle = 1: imagen.Picture = LoadPicture(App.Path & "\imagen 4.bmp")
    Case 5
      Im(5).BorderStyle = 1: imagen.Picture = LoadPicture(App.Path & "\imagen 5.bmp")
    Case 6
      Im(6).BorderStyle = 1: imagen.Picture = LoadPicture(App.Path & "\imagen 6.bmp")
    Case 7
      Im(7).BorderStyle = 1: imagen.Picture = LoadPicture(App.Path & "\imagen 7.bmp")
  End Select
End Sub
Private Sub Timer1_Timer()
      If derecha Then
        'imagen.Picture = LoadPicture()
        Im(actual).BorderStyle = 0
        actual = actual + 1
        If actual > 7 Then actual = 0
        archivo = App.Path & "\imagen " + Trim(Str(actual)) + ".bmp"
        imagen.Picture = LoadPicture(archivo)
        Im(actual).BorderStyle = 1
      End If
      If izquierda Then
        Im(actual).BorderStyle = 0
        actual = actual - 1
        If actual < 0 Then actual = 7
        archivo = App.Path & "\imagen " + Trim(Str(actual)) + ".bmp"
        imagen.Picture = LoadPicture(archivo)
        Im(actual).BorderStyle = 1
      End If
End Sub
Private Sub Form_Keydown(KeyCode As Integer, Shift As Integer)
  If KeyCode = vbKeyLeft Then izquierda = True
  If KeyCode = vbKeyRight Then derecha = True
End Sub
Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)
   If KeyCode = vbKeyLeft Then izquierda = False
  If KeyCode = vbKeyRight Then derecha = False
End Sub
```

PROGRAMA # 7
REPETICIÓN DE IMÁGENES

Para realizar nuestro programa necesitamos únicamente una caja de imagen (*PictureBox*) con el archivo de la imagen que deseamos repetir, cambiar la propiedad de esta imagen a "*visible=false*" y "*AutoSize=True*".

Al ejecutar este programa calcula el ancho y el alto de la imagen, dibujando una imagen después de la anterior en forma consecutiva hasta llenar la pantalla; para hacer esto utilizamos dos ciclos *While* donde inicialmente las coordenadas "X" o "Y" se inicializan con cero y se incrementan el ancho o el largo de la imagen; después de dibujar la imagen este ciclo se repite hasta que las coordenadas "X" o "Y" sobrepasen el tamaño de la ventana. Ver figura 4.1.7.

Figura 4.1.7. Uso de función PaintPicture.

Código del Programa

```
Private Sub Form_Paint()
  Dim ancho As Single
  Dim largo As Single
  Dim x As Single
  Dim y As Single
  ancho = Picture1.ScaleWidth
  alto = Picture1.ScaleHeight
  y = 0
  Do While y < ScaleHeight
     x = 0
     Do While x < ScaleWidth
       PaintPicture Picture1.Picture, x, y, ancho, alto
       x = x + ancho
     Loop
     y = y + alto
  Loop
End Sub
```

PROGRAMA # 8
DESPLAZAMIENTO DE DOS PANTALLAS

Esta rutina simula el movimiento de dos escenarios, desplazando dos cajas de imagen (*PictureBox*) en forma simultánea, una después de otra. Inicialmente se sitúa la primera imagen en la esquina superior izquierda (coordenadas *(0,0)*) y la segunda imagen inmediatamente después de la primera imagen (coordenadas *(0,9600)*). Se utiliza también las funciones para la detección del teclado (*Keydown y KeyUp*) para que al presionar la flecha izquierda, desplace al mismo tiempo las dos pantallas 100 twips a la izquierda y si se presiona la tecla de la derecha realiza la misma operación, solo que en sentido contrario. Ver figura 4.1.8.

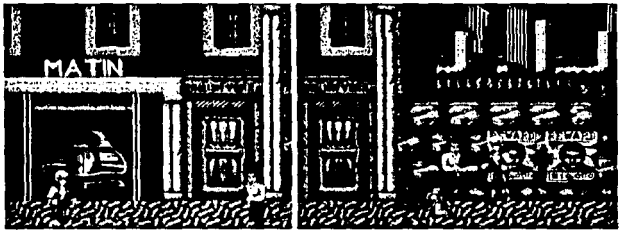


Figura 4.1.8. Desplazamiento de dos imágenes por medio del teclado.

Código del Programa

```
Dim x As Integer, y As Integer
Dim img1 As Integer, img2 As Integer
Dim izquierda As Boolean, derecha As Boolean
Private Sub Form_Load()
  y = 0: x = 0
  img1 = 9600
  Picture1.Move x, y
  Picture2.Move x + img1, y
End Sub
Private Sub Timer1_Timer()
 If derecha Then
   Picture1.Move x, y
   Picture2.Move x + img1, y
   x = x - 100
 End If
 If izquierda Then
   Picture1.Move x, y
   Picture2.Move x + img1, y
   x = x + 100
 End If
End Sub
Private Sub Form_Keydown(KeyCode As Integer, Shift As Integer)
  If KeyCode = vbKeyLeft Then izquierda = True
  If KeyCode = vbKeyRight Then derecha = True
End Sub
Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)
   If KeyCode = vbKeyLeft Then izquierda = False
  If KeyCode = vbKeyRight Then derecha = False
End Sub
```

PROGRAMA # 9
SEGUIR A OTRA IMAGEN

Esta rutina es clásica en los juegos de video. El programa Utiliza dos imágenes: una de un pacman y otra de un fantasma que lo sigue desde su lugar de origen hasta la posición en que se encuentra el pacman.

El pacman es controlado por las teclas de navegación, y de acuerdo a la tecla presionada se mueve la imagen del pacman en 100 twips hacia arriba, abajo, izquierda o derecha; pero antes se tiene que verificar si no se encuentra fuera de las coordenadas establecidas y si es así lo mueve 100 unidades en dirección contraria.

El movimiento del fantasma es controlado por medio de un Timer que verifica la posición del pacman cada 100 milisegundos y mueve el fantasma de acuerdo a las coordenadas del pacman y la distancia Horizontal y Vertical entre el fantasma y el pacman.

Por medio de sentencias ***IF*** se comparan las distancias y la ubicación de cada imagen, desplazando el fantasma hacia el *Pacman*. Para terminar este juego es necesario que la distancia entre el pacman y el fantasma sea inferior a 50 twips. Ver figura 4.1.9.

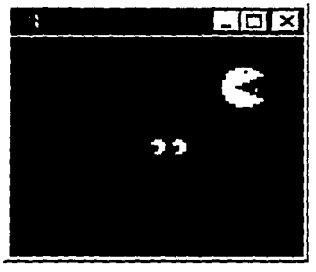


Figura 4.1.9. Simulación de seguimiento de dos imágenes.

Código del Programa

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
  If Pacman.Left <= 0 Then Pacman.Left = Pacman.Left + 100
  If Pacman.Left >= 9480 Then Pacman.Left = Pacman.Left - 100
  If Pacman.Top >= 8040 Then Pacman.Top = Pacman.Top - 100
  If Pacman.Top <= 100 Then Pacman.Top = Pacman.Top + 100
  If KeyCode = vbKeyUp Then
     Pacman.Top = Pacman.Top - 100
  End If
  If KeyCode = vbKeyDown Then
     Pacman.Top = Pacman.Top + 100
  End If
  If KeyCode = vbKeyLeft Then
     Pacman.Left = Pacman.Left - 100
  End If
  If KeyCode = vbKeyRight Then
     Pacman.Left = Pacman.Left + 100
  End If
End Sub
Private Sub Timer1_Timer()
  Dim DistVert As Integer, DistHori As Integer
  DistVert = Fantasma.Top - Pacman.Top
  DistHori = Fantasma.Left - Pacman.Left
  If Fantasma.Left <= 0 Then Fantasma.Left = Fantasma.Left + 100
  If Fantasma.Left >= 9480 Then Fantasma.Left = Fantasma.Left - 100
  If Fantasma.Top >= 8040 Then Fantasma.Top = Fantasma.Top - 100
  If Fantasma.Top <= 100 Then Fantasma.Top = Fantasma.Top + 100
  If Fantasma.Top <= Pacman.Top Then Fantasma.Top = Fantasma.Top + 50
  If DistVert >= DistHori Then Fantasma.Left = Fantasma.Left + 50
  If Fantasma.Left <= Pacman.Left Then Fantasma.Left = Fantasma.Left + 50
  If DistVert <= DistHori Then Fantasma.Left = Fantasma.Left - 50
  If DistHori <= DistVert Then Fantasma.Top = Fantasma.Top - 50
  If Abs(DistHori) <= 50 And Abs(DistVert) <= 50 Then
    MsgBox ("Pacman Capturado")
    Unload Form1
  End If
End Sub
```

PROGRAMA # 10
ABRIR ARCHIVOS DE TEXTO

El programa abre un archivo de texto llamado Jugadores.txt línea por línea utilizando un **ciclo Do** hasta que sea fin de archivo (*EOF*). El contenido del archivo se coloca en un cuadro de texto con las barras horizontal y vertical activadas (*ScrollBars= 3 Both*). También se puede bloquear la edición de texto por medio de la propiedad *Text1.Locked = True*.

En nuestro formulario agregamos un botón de comando (*Commad1*) para que al presionar el botón abra nuestro archivo de texto. Ver figura 4.2.0.

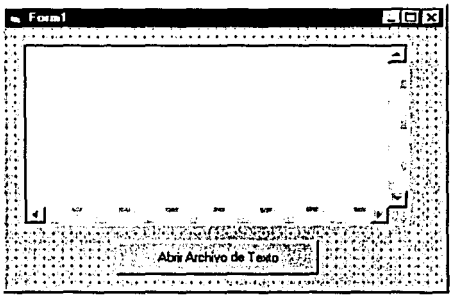


Figura 4.2.0. Apertura de archivos de texto.

Código del Programa

```
Private Sub Command1_Click()
  Dim Archivo As String
  Dim linea As String
  Archivo = App.Path & "\Jugadores.txt"
  Open Archivo For Input As #1
  Do
    Line Input #1, linea
    Form1.Text1.Text = Form1.Text1.Text + linea + Chr(13) + Chr(10)
  Loop Until EOF(1)
   Text1.Locked = True
  Close #1
End Sub
```

PROGRAMA # 11
ABRIR ARCHIVOS DE AYUDA

Este programa utiliza una etiqueta con la propiedad ***caption = Nombre del Archivo (.hlp)***, una caja de texto que al iniciar el programa su propiedad *Text="helpsql.hlp"*, un botón para abrir el archivo de ayuda y otro para finalizar el programa. También incorpora un módulo para manejo de archivos de ayuda que incorpora la librería *"eser32.dll"* de windows.

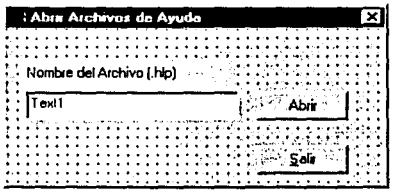


Figura 4.2.1. Apertura de archivos de ayuda.

Código del Programa

```
Public Sub Form_Load()
  Caja_de_Texto1.Text = "helpsql.hlp"
End Sub

Private Sub Botón_Abrir_Click()
  ChDir App.Path
  Función_Ayuda Me.hwnd, 3, ""
End Sub

Private Sub Botón_Salir_Click()
  Unload Me
End Sub
```

Modulo.bas

```
Declare Function WinHelp Lib "user32" Alias "WinHelpA" (ByVal hwnd As Long,
ByVal lpHelpFile_ As String, ByVal wCommand As Long, ByVal dwData As Any)
As Long

Sub Función_Ayuda(Handle As Long, Action As Integer, Chaine As String)
Dim IRtn As Long
  IRtn = WinHelp(Handle, Form1.Caja_de_Texto1.Text, Action, 0&)
End Sub
```

PROGRAMA # 12
CANTIDAD DE FUERZA DE UN JUGADOR

Con el uso de dos Paneles de Avance, simulamos la cantidad de fuerza de un jugador; en este caso, al presionar alguna de las flechas de la barra de desplazamiento Horizontal, disminuye o aumenta el valor del panel. El Panel1 que es de color rojo se desplaza de Izquierda a derecha y el Panel2 que es de color azul se desplaza de derecha a izquierda.

Al iniciar el programa, a la barra de desplazamiento horizontal le asignamos el valor de 50 (*HScroll1.Value = 50)*, los valores mínimo de 0 (*HScroll1.Min = 0)* y máximo de 100 (*HScroll1.Max = 100)*. A el panel1 y Panel2 le cambiamos la propiedad de color (*FloodColor)*, la propiedad de desplazamiento (*FloodType)* y la de porcentaje (*FloodShowPct)*. Ver figura 4.2.2.

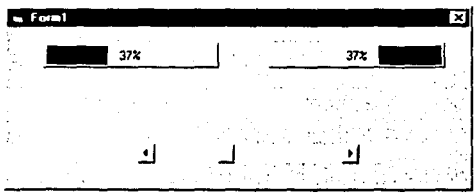


Figura 4.2.2. Uso de Paneles de Avance y barra de desplazamiento.

Código del Programa

```
Private Sub Form_Load()
  HScroll1.Value = 50
  HScroll1.Min = 0
  HScroll1.Max = 100
  axPanel1.FloodColor = RGB(255, 0, 0)
  axPanel1.FloodType = 1
  axPanel1.FloodShowPct = True
  axPanel2.FloodColor = RGB(0, 0, 255)
  axPanel2.FloodType = 2
  axPanel2.FloodShowPct = True
End Sub

Private Sub HScroll1_Change()
  axPanel1.FloodPercent = HScroll1.Value
  axPanel2.FloodPercent = HScroll1.Value
End Sub
```

PROGRAMA # 13
CONTADOR REGRESIVO

El Programa necesita una etiqueta de color amarillo y con la propiedad *Caption=00:00*, tres Botones: uno Para activar el reloj, otro para hacer una pausa y otro para salir del formulario.

Al iniciar nuestro programa muestra nuestra ventana en el centro de la pantalla y con el reloj en 2 minutos. Al activarlo el reloj (*Timer1*) se disminuye el valor de la variable segundos en una unidad hasta que el valor de los segundos y los minutos sea igual a cero; es importante que este proceso se haga cada 1000 milisegundos para que nuestro contador regresivo se realice cada segundo. Ver figura 4.2.3.

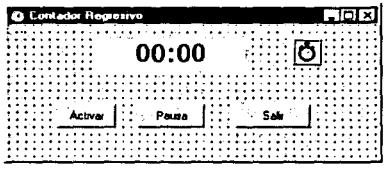


Figura 4.2.3. Uso del Timer para crear un reloj.

Código del Programa

```
Dim Minutos As Integer
Dim Segundos As Integer
Dim Tiempo As Date
Private Sub Form_Load()
Form1.Top = (Screen.Height - Form1.Height) / 2
Form1.Left = (Screen.Width - Form1.Width) / 2
Timer1.Interval = 1000
Minutos = 2
Segundos = 0
Tiempo = TimeSerial(0, Minutos, Segundos)
Label1.Caption = Format$(Tiempo, "nn") & ":" & Format$(Tiempo, "ss")
End Sub
Private Sub Activar_Click()
   Timer1.Enabled = True
End Sub
Private Sub Pausa_Click()
   Timer1.Enabled = False
End Sub
Private Sub Salir_Click()
   End
End Sub
Private Sub Timer1_Timer()
      Tiempo = DateAdd("s", -1, Tiempo)
      Label1.Caption = Format$(Tiempo, "nn") & ":" & Format$(Tiempo, "ss")
      If Label1.Caption = "00:00" Then End
End Sub
```

PROGRAMA # 14
LETRAS EN MOVIMIENTO

El programa simula el movimiento de un mensaje contenido en una etiqueta. Para realizarlo es necesario asignar a la etiqueta *Label1* el título de " *1995 ALL RIGHTS RESERVED...* " y cambiar el color de la letra a color Verde: *Label1.ForeColor = RGB(0, 255, 0)*. También contiene una rutina que reemplaza el título actual por una copia de todos los caracteres contenidos en el título previo, exceptuando el último carácter y agregándole al final el primer carácter; este proceso se realiza cada 400 milisegundos, pero se le puede cambiar a 100,200 o 300 para que el avance se vea más rápido y las propiedades de la fuente para tener una mejor presentación. Ver figura 4.2.4.

Figura 4.2.4. Manejo de cadenas de caracteres (copiado y borrado).

Código del Programa

```
Private Sub Form_Load()
  Label1.AutoSize = True
  Label1.FontSize = 14
  Label1.Caption = "        1995 ALL RIGHTS RESERVED...    "
  Label1.ForeColor = RGB(0, 255, 0)
  Timer1.Interval = 300
End Sub

Private Sub Timer1_Timer()
  Label1.Caption = Right$(Label1.Caption, Len(Label1.Caption) - 1) & _
  Left$(Label1.Caption, 1)
End Sub
```

PROGRAMA # 15
CREACIÓN DE MENÚS

Para crear este Programa utilizamos el **editor de menús** que se encuentra dentro del menú **Herramientas**; colocando como título (*Caption:*) del menú **&Juego**, el nombre del menú (*Name:*) como **MenuJuego.** Ver figuras 4.2.5 y 4.2.6.

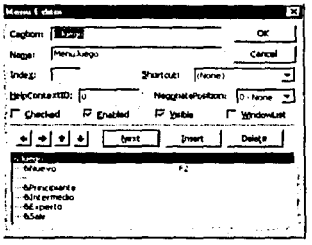


Figura 4.2.5. Editor de menús.

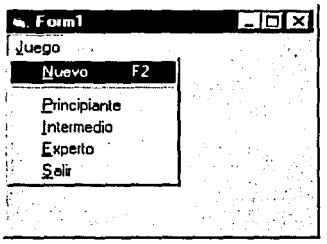


Figura 4.2.6. Menú Juego.

A continuación se presiona la flecha hacia abajo y a la derecha para marcar el nivel que se requiera. También utilizaremos el botón de *Siguiente*, el de *Insertar* y el de B*orrar* si es que nos llegamos a equivocar. Se tiene que colocar el nombre del título y el nombre del menú con el mismo nombre del título sin el *ampersand* a cada uno de los submenús (*&Nuevo, &Principiante, &Intermedio, &Experto y &Salir*)., Para agregar el separador después del submenú *Nuevo*, se utiliza un guión (-). Si queremos que al seleccionar un menú, realice alguna operación como el de desplegar un mensaje o salir del programa, tendremos que teclear el siguiente código de Programa:

```
Private Sub Nuevo_Click()
  MsgBox "Seleccionaste Nuevo"
End Sub
Private Sub Principiante_Click()
  MsgBox "Seleccionaste Principiante"
End Sub
Private Sub Intermedio_Click()
  MsgBox "Seleccionaste Intermedio"
End Sub
Private Sub Experto_Click()
  MsgBox "Seleccionaste Experto"
End Sub
Private Sub Salir_Click()
  Unload Form1
End Sub
```

PROGRAMA # 16
BOTONES DE OPCIÓN

Tenemos que crear un cuadro de opciones y dentro de este cuadro insertamos un arreglo de botones de opción (*Opción(0), Opción(1), Opción(2) y Opción(3)*). También necesitamos un botón de continuar para que nos muestre el programa en una caja de diálogo la opción que seleccionamos previamente. Ver figura 4.2.7.

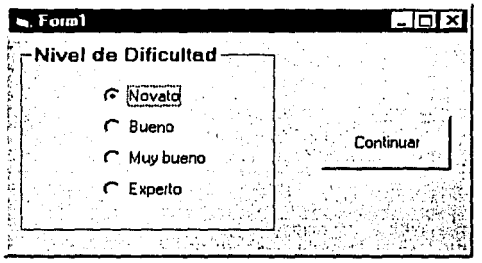


Figura 4.2.7. Uso de botones de opción.

Código del Programa

```
Private Sub Command1_Click()
  If Opción(0) = True Then
    MsgBox "Seleccionaste Novato"
  ElseIf Opción(1) = True Then
    MsgBox "Seleccionaste Bueno"
  ElseIf Opción(2) = True Then
    MsgBox "Seleccionaste Muy Bueno"
  ElseIf Opción(3) = True Then
    MsgBox "Seleccionaste Experto"
  End If
End Sub
```

PROGRAMA # 17
REPRODUCTOR DE ARCHIVOS AVI

La reproducción de videos es otra función más que se incorpora en esta lista de programas. Un archivo *AVI* no es más que varias imágenes en secuencia (*Frames*) y algunos archivos también incorporan el audio; esta función es de gran importancia para dar una mejor presentación a nuestros proyectos.

El programa incorpora una clase que es necesaria para la ejecución de nuestro programa llamada *Mmedia* y utiliza los siguientes métodos:

mm.Open.- Para abrir el archivo indicado en nuestro programa.
mm.Pause.- Interrumpe la reproducción del archivo que estamos escuchando.
mm.Stop.- Finaliza la reproducción de nuestro archivo y lo cierra automáticamente.
mm.Play.- Reproduce el archivo indicado.

También utiliza algunas variables de uso interno y la librería "*Winmm*" de windows y selecciona automáticamente el tipo de archivo "*WAV*", "*AVI*" o "MID" y lo reproduce según sea el caso. Para realizar nuestro programa, el primer punto es abrir un nuevo formulario y agregar a la propiedad *Caption* con el nombre "*Reproductor de AVI*", con esto colocamos un título a nuestro formulario. Como segundo paso agregamos un control llamado *Microsoft Common Dialog Control 6.0* que se encuentra dentro los componentes, al presionar el botón derecho del ratón dentro de la caja de herramientas. También tenemos que agregar dos botones, uno con el nombre de *Play* y otro con el nombre Salir.

Al presionar el botón *Play* el programa abre el archivo *"Confuso.avi"* que se encuentra en el directorio actual y esta hecho en **3D Studio Max**. El botón Salir cierra el dispositivo MCI de *Audio-Video* si estuviera abierto y también cierra nuestro formulario. Ver figuras 4.2.8 y 4.2.9.

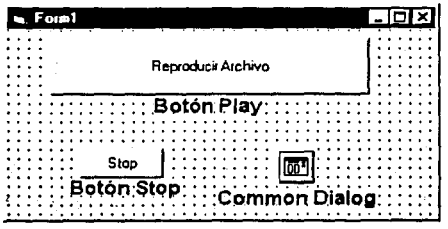


Figura 4.2.8. Uso de la función Mmedia.

Figura 4.2.9. Reproduccion del archivo .avi

Código del Programa

```
Dim Multimedia As New Mmedia
Dim Archivo As String
Private Sub Play_Click()
  Archivo = App.Path & "\confuso.avi"
  Multimedia.mmOpen Archivo
  Multimedia.mmPlay
End Sub
Private Sub Stop_Click()
   Multimedia.mmStop
   Multimedia.mmClose
   Unload Me
End Sub
```

Mmedia.cls

```
Private sAlias As String, sFilename As String,  bWait As Boolean
Private Declare Function mciSendString Lib "winmm.dll" _
  Alias "mciSendStringA" (ByVal lpstrCommand As String, _
  ByVal lpstrReturnString As String, ByVal uReturnLength As Long, _
  ByVal hwndCallback As Long) As Long
Public Sub mmOpen(ByVal sTheFile As String)
   Dim nReturn As Long,  sType As String
   If sAlias <> "" Then
      mmClose
   End If
   Select Case UCase$(Right$(sTheFile, 3))
     Case "WAV"
       sType = "Waveaudio"
     Case "AVI"
       sType = "AviVideo"
     Case "MID"
       sType = "Sequencer"
       Case Else
       Exit Sub
   End Select
   sAlias = Right$(sTheFile, 3) & Minute(Now)
   If InStr(sTheFile, " ") Then sTheFile = Chr(34) & sTheFile & Chr(34)
   nReturn = mciSendString("Open " & sTheFile & " ALIAS " & sAlias & " TYPE " & sType & " wait", "", 0, 0)
End Sub
Public Sub mmClose()
   Dim nReturn As Long
   If sAlias = "" Then Exit Sub
   nReturn = mciSendString("Close " & sAlias, "", 0, 0)
   sAlias = "":    sFilename = ""
End Sub
Public Sub mmPause()
   Dim nReturn As Long
   If sAlias = "" Then Exit Sub
   nReturn = mciSendString("Pause " & sAlias, "", 0, 0)
End Sub
Public Sub mmPlay()
   Dim nReturn As Long
   If sAlias = "" Then Exit Sub
   If bWait Then
      nReturn = mciSendString("Play " & sAlias & " wait", "", 0, 0)
   Else
      nReturn = mciSendString("Play " & sAlias, "", 0, 0)
   End If
End Sub
Public Sub mmStop()
   Dim nReturn As Long
   If sAlias = "" Then Exit Sub
   nReturn = mciSendString("Stop " & sAlias, "", 0, 0)
End Sub
```

PROGRAMA # 18 y # 19
REPRODUCCIÓN DE ARCHIVOS WAV Y ARCHIVOS MIDI

Estos programas son parecidos al reproductor de archivos AVI , ya que utilizan la misma función.

Para reproducir archivos WAV y MIDI únicamente se cambia el nombre del archivo y su respectiva extensión:
Archivo = App.Path & "**\normal.mid**" o
Archivo = App.Path & "**\tchaikovsky2.wav**" .

Cabe mencionar que si se encuentra mal escrito el nombre del archivo o no se encuentra el archivo en el directorio actual de trabajo, no se efectuara esta reproducción.

Estos programas incorporan dos botones más: uno para la *Pausa* y otro para *Continuar.*

Al iniciar el programa desactiva el botón Pausa y el botón Continuar, al presionar el botón de Reproducción activa el botón de pausa y al presionar el botón pausa se desactiva y activa el botón Continuar; análogamente al presionar el botón continuar este se desactiva y activa el botón de pausa. Ver figura 4.3.0.

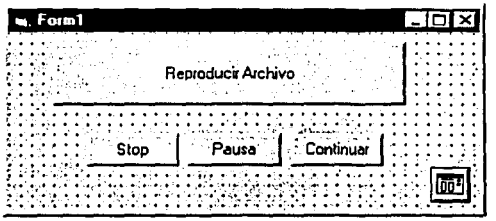


Figura 4.3.0. Uso de la función Mmedia para la reproducción de archivos WAV y MIDI.

Código del Programa

```
Dim Multimedia As New Mmedia
Dim Archivo As String
Private Sub Form_Load()
  Continuar.Enabled = False
  Pausa.Enabled = False
End Sub
Private Sub Play_Click()
  Archivo = App.Path & "\normal.mid"
  Multimedia.mmOpen Archivo
  Multimedia.mmPlay
  Continuar.Enabled = False
  Pausa.Enabled = True
End Sub
Private Sub Continuar_Click()
  Multimedia.mmPlay
  Continuar.Enabled = False
  Pausa.Enabled = True
End Sub
Private Sub Pausa_Click()
  Multimedia.mmPause
  Pausa.Enabled = False
  Continuar.Enabled = True
End Sub
Private Sub Stop_Click()
   Multimedia.mmStop
   Multimedia.mmClose
   Pausa.Enabled = True
   Continuar.Enabled = False
   Unload Me
End Sub
```

PROGRAMA # 20
ABRIR UNA BASE DE DATOS

Este Programa no utiliza código, solo hay que crear un *data control* para manejar la base de datos, dos etiquetas y dos cuadros de texto como se muestra en la figura 4.3.1.

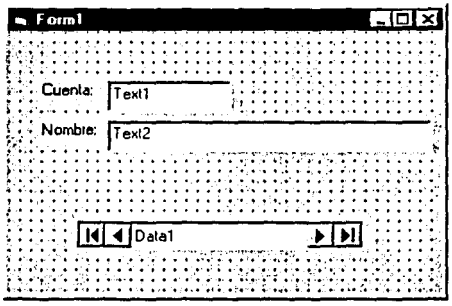


Figura 4.3.1. Uso del Data Control para el manejo de bases de datos.

A cada objeto correspondiente se les asigna las siguientes propiedades:

Objeto	Propiedad	Valor
Data1	*Caption*	Data1
	Connect	dBaseIII;
	DataBaseName	d:\Programas\Programa20 (Base de Datos)
	RecordSource	primeros
Label1	*Caption*	Cuenta:
Label2	*Caption*	Nombre:
Text1	*DataField*	Cuenta
Text2	*DataField*	Nombre

Al ejecutar el programa abre aromáticamente la base de datos "primeros" de DBASEIII en el directorio especificado y muestra los campos **cuenta** y **nombre** en los cuadros de texto. Al Presionar los botones de nuestro data control cambia la posición del registro con los datos correspondientes según sea el caso:

en el primer registro
en el registro anterior
en el siguiente registro
en el último registro

CONCLUSIONES

Se pueden programar juegos en cualquier lenguaje de programación y en cualquier plataforma. Los juegos y programas que se describen en el presente trabajo se realizaron en Visual Basic, ya que es un lenguaje muy fácil de aprender y utilizando los controles adecuados rápidamente podemos crear juegos sin tener que escribir mucho código ni algoritmos complicados; solo tenemos que cambiar las propiedades de los objetos y tener un poco de creatividad.

Existen otras herramientas que permiten la creación de juegos y aplicaciones multimedia para Windows, como lo es DirectX, que es una API (*Interfaz de Programación de Aplicaciones*) y proporciona a los programadores las estrategias y herramientas necesarias y no tener que preocuparse por la marca de la tarjeta de sonido, del fabricante de la tarjeta gráfica, o si va a tener aceleración 3D. Entre las ventajas principales se encuentra la rapidez de acceso directo al hardware, en especial para las tarjetas de video y la facilidad para la configuración del tipo de video, la profundidad de colores y la reproducción de sonido. El Kid de desarrollo de Software de DirectX se compone de múltiples componentes que mejora el rendimiento de la programación, entre los cuales se encuentran:

DirectDraw.- Permite al programador tener acceso directo a la memoria de vídeo, obteniendo mucho mayor rapidez a la hora de implementar las técnicas de animación, además, permite en los juegos cambiar la resolución de la pantalla sin necesidad de salir de los mismos.

DirectSound.- Facilita a los programadores la reproducción y las mezclas de sonido tanto por Software como por Hardware. También puede reproducir sonido en 3D.

DirectMusic.- Es el componente musical de DirectX, trabaja con datos musicales basados en mensajes como los ficheros MIDI, que se convierten en ficheros *.WAV para su reproducción.

DirectInput.- Se utiliza para habilitar los controladores de juegos como el joystick, el ratón, el teclado, los pads y cualquier dispositivo de entrada de datos.

DirectPlay.- Habilita la conexión de juegos a través de un módem o de una red de ordenadores. Proporciona una manera eficaz para que las aplicaciones o los juegos se pueden comunicar entre sí y poder realizar juegos con multijugador.

Direct3D.- Es el componente de DirectX que proporciona a los programadores de juegos una interfaz de alto nivel que permite programar aplicaciones 3D fácilmente.

DirectSetup.- Proporciona a los programadores un sencillo procedimiento instalación de programas bajo ambiente Windows.

AutoPlay.- Ejecuta las aplicaciones de forma automática al insertar un CD-ROM. Si el disco introducido en la unidad de CD-ROM fuera en compact disc de música, se ejecutaría inmediatamente el programa "Reproductor de CD" de Windows.

Los juegos de video estimulan el proceso de aprendizaje de los niños y adolescentes, además de que ejercitan su mente, la memoria, la capacidad de razonamiento y la coordinación entre sus sentidos.

Los programas educativos son una fuente muy importante de conocimiento en los cuales aprendes divirtiéndote, desafortunadamente no en todos los hogares se encuentra una computadora o las que se encuentran en las escuelas son insuficientes.

La filosofía del constructivismo apoya la enseñanza mediante el juego, ya que el juego hace reflexionar y genera conocimiento.

Al igual que en todos los medios de información como los libros, revistas, radio, televisión, videos, o Internet, el uso de los juegos por periodos prolongados de tiempo (más de cuatro horas diarias en promedio), afectan su vida personal y pueden llegar a dañar su sentido de la vista.

Es importante que los padres de familia y profesores concienciemos a nuestros hijos y alumnos en el uso adecuado de estas nuevas tecnologías y fuentes de información.

REFERENCIAS BIBLIOGRÁFICAS

1. Quirós Peñalva, Enrique y Quirós Domínguez, Sergio, *Photoshop Práctico*, España, 2000, Edit. Osborne-McGraw-Hill, p.p. 100-200.

2. "Técnicas Avanzadas de Photoshop", pp. 176-179, en *PC MANIA*, año *VIII*, #79, mensual, España, 1999, Editorial Hobby Press, S.A.

3. "Dígalo con imágenes", p.p. 78-81, en *Pc Magazine en español*, año II, #5, mensual, México, 2000, Editorial Televisa, S.A. de C.V.

4. "El Sonido, el Audio, la Imagen y el Video", pp. 1-20, *en Curso práctico de Audio y Video Digital*, #1, mensual, España, 2000, Editorial DAT House, S.L.

5. "Watts, Volts, Hertzs y algo mas", pp. 6-8, en *Audio Car*, #12, *mensual*, México, 2000, Editorial Mina, S.A. de C.V.

6. "Dolby Digital", pp. 24-26, en *Audio y Video*, #3, mensual, México, 2000, Editorial Mina, S.A. de C.V.

7. "Decibeles y Watts", pp. 44-45, en *Audio Car*, #15, *mensual*, México, 2000, Editorial Mina, S.A. de C.V.

8. "Búsqueda en el Word Wide Web", pp. 50-51, en *Pc Computing*, año V, *mensual*, #6, México, 1998, Editorial Televisa, S.A. de C.V.

9. "Juegos en Internet", pp. 18-23, en *Las mejores Webs Actualizadas*, año 1, #3, *mensual*, España, 1997, Editores TU.

10. "Sonido 3D", pp. 10-14 en *Pc Media*, Año II, #18, México, *1995*, Editorial Ness, S.A. de C.V.

11. "Técnicas orientadas a Objetos", pp. 18-21, en *Pc Programación*, #3, *mensual*, España, 1994, Editorial Magim Ediciones S.L.

PÁGINAS CONSULTADAS EN INTERNET:

1. http://www.yahoo.com/
2. http://www.altavista.com/
3. http://www.tlmsn.com/
4. http://www.ciudadfutura.com/
5. http://www.lawebdelprogramador.com/
6. http://www.qsound.com/
7. http://www.winamp.com/
8. http://www.microsoft.com/downloads/
9. http://www.microsoft.com/msagent/
10. http://www.syntrillium.com/cooledit/
11. http://www.adobe.com/products/photoshop/
12. http://www.kidwaresoftware.com/
13. http://www.zarr.net/vb/download/
14. http://www.alvbcode.com/
15. http://www.hormiga.org/
16. http://software.starmedia.com/Programacion/
17. http://www.codearchive.com/
18. http://members.ozemail.com.au/~kjsmith/vbstuff.htm