12

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

''ARAGON''

INGENIERIA EN COMPUTACION

"FUNDAMENTOS DE PROGRAMACION PARA LA ELABORACION DE JUEGOS DE VIDEO"

TESIS PROFESIONAL

Que para obtener el Título de:

INGENIERO EN COMPUTACION

Presentan:

FRANCISCO XAVIER ESPINOSA MADRIGAL

ITSMAEL MANZO SALAZAR

Asesor: Victor R. Velasco Vega

San Juan de Aragón, Edo. de Méx. 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Dedico esta tesis

A mis padres Reveriano y Mariaelena, agradeciendo su apoyo y desvelos desde el inicio de esta gran aventura. Nadie sabe nunca con que cosas le saldrán los hijos.

A mis hermanos Ernesto y Anabel, esperando que esto sirva como estímulo a su carrera y pongan el mismo empeño.

A Francis, por su amor y comprensión.

A Olga, Tony, Raul, Paco, Gustavo, Martin, y a la familia Ortiz Pérez.

Y a todas aquellas personas que de una u otra manera me han apoyado en la vida.

Itsmael.

Paco:

- *Agradezco a mis papas todo el apoyo que me brindaron en mi carrera profesional, a ellos les dedico mi título profesional con todo mi corazón. Les agradezco también su preocupación por mi salud cuando me decían "deja ya esa computadora y vete a dormir", pero realmente si sirvieron esas horas extras que invertí.*

- *A todos mis hermanos por creer en mí. Especialmente a mi hermano Luis Felipe por crear e impulsar mi vocación profesional con su ejemplo; a mi hermana Yazmín por todo lo que me ha ayudado; y a mis hermanos Raúl y Víctor por dejarme estar a altas horas de la noche en la computadora sin acusarme con mis papas.*

- *A mis amigos, Ismael, Martín y Gustavo que lograron a picar mi amor propio, y así poder terminar regularmente mi ciclo de estudios, y por supuesto también por su amistad.*

Un agradecimiento especial al Ingeniero Victor Raul Velasco Vega por las facilidades y el apoyo brindado para la elaboración de esta tesis.

Itsmael

Paco

INDICE

OBJETIVOS

OBJETIVO GENERAL

⇒ Diseñar un juego de video del tipo comercial para mostrar sus herramientas de programación, complejidad y tendencias de su desarrollo.

OBJETIVOS ESPECIFICOS

⇒ Utilización de técnicas de programación especializadas.
⇒ Realización de un trabajo original e innovador.
⇒ Disipar algunas de las dudas e inquietudes de las personas interesadas en el tema.
⇒ Proporcionar información no disponible u oculta por programadores expertos.
⇒ Lograr la coordinación para un trabajo de tal magnitud.
⇒ Cimentar las bases para la realización de trabajos futuros de igual o mejor calidad.
⇒ Disminuir la dependencia de programas extranjeros al grado mínimo.
⇒ Sugerir la generación de un mercado inexistente en México como el de los juegos de video (a mediano plazo).

INTRODUCCION

Mucha gente piensa que hacer programas "serios" como hojas de cálculo, bases de datos y procesadores de texto es más complicado que realizar juegos para computadora. Esto no siempre es exacto, como en todo hay niveles de dificultad, y francamente, en nuestra experiencia como programadores hemos requerido de una mayor cantidad de conocimientos, análisis, y tiempo para desarrollar un juego que para cualquier otro tipo de programa. Es tal vez esta razón, la que nos haya persuadido para desarrollar el tema de esta tesis.

El programador profesional de juegos requiere un profundo conocimiento sobre gráficas, dispositivos periféricos, y programación a bajo nivel para poder imprimir mayor realismo a su creación. Aunque por regla general, casi nunca está solo, puede contar con un equipo formado por músicos, dibujantes y otros programadores que realizan el trabajo en equipo. Para darse una idea de los recursos necesarios para realizar un juego comercial para computadoras se tomará el ejemplo de la realización de *Leisure Suit Larry 5*, uno de los juegos de aventuras más populares desarrollado por Sierra On Line. Tomó alrededor de 300 personas/año para ser creado, 4,000 dibujos hechos a manos y luego digitalizados, 1/4 de millón de eventos MIDI (piezas o fragmentos de tonadas musicales) y 2,200,000 líneas de código fuente (probablemente escrito en lenguaje C y ensamblador). Estas impresionantes cifras se transforman en un juego que se vende por un precio promedio de N$200.00 en México o aproximadamente $40.00 dólares en Estados Unidos, y representa 3 días de duración para un fanático de los juegos de aventura (o probablemente varias semanas para otro tipo de jugadores menos apasionados).

¿Esto quiere decir que los programadores novatos o de nivel medio no puede diseñar juegos para computadora?. De ninguna manera, un juego no necesita tener una interface gráfica impresionante, ni efectos de sonido estéreo para resultar entretenido. Sin embargo, para desarrollar juegos de aventura interactivos, juegos de acción que impliquen combates, o juegos en modo protegido, sí es necesario un poco más de esfuerzo, dedicación y conocimientos.

En esta tesis nos proponemos demostrar que es posible realizar un juego para computadora de tipo comercial con los recursos técnicos y tecnológicos que se encuentran disponibles en México.

La presente obra se divide en 11 capítulos los cuales se explican a continuación:

CAPITULO 1. JUEGOS DE VIDEO: PASADO, PRESENTE Y FUTURO: Se relata la evolución de los juegos de computadora así como sus tendencias. También se explican los diferentes tipos de juegos para computadoras que existen.

CAPITULO 2. RECURSOS HUMANOS Y MATERIALES: Se hace una descripción de los recursos que en forma ideal se necesitan para realizar un juego para computadora profesional.

CAPITULO 3. LENGUAJES DE PROGRAMACION: Se explica la razón por la cual se utiliza pascal para el desarrollo del juego, y algunas rutinas y procedimientos especializados que pudieran causar confusión al lector.

CAPITULO 4. ORGANIZACION DE DATOS: Se muestran los formatos y tipos de registros utilizados en común por todos los programas desarrollados en la tesis.

CAPITULO 5. ENTORNO GRAFICO: Se da una explicación teórica de los modos gráficos utilizados por la computadoras personales y sus elementos.

CAPITULO 6. SOFTWARE DE APOYO: Se enumeran todos los programas que se utilizaron para el desarrollo del juego y la función que desempeñaron.

CAPITULO 7. EDITOR DE DUENDES Y ANIMACION: Se describe el entorno gráfico, utilización y formatos del programa editor de duendes y animación.

CAPITULO 8. EDITOR DE MUSICA: Se explica como incluir música y efectos especiales en un juego para computadora, y sus principios teóricos.

CAPITULO 9. LIBRETO Y PROGRAMA EJECUTOR: Se hace una descripción de los entornos gráficos, modo de uso y formatos del libreto y del programa ejecutor.

CAPITULO 10. DESCRIPCION DEL JUEGO DE VIDEO REALIZADO: Se hace una reseña de las herramientas, elementos y procedimientos que se utilizaron para el juego utilizado en esta tesis.

CAPITULO 11. VIDEO JUEGOS EN MEXICO: Se analiza la situación de los juegos para computadora en México y se explican los pasos para obtener los derechos de autor de un juego.

GLOSARIO Y ANEXOS. Se proporciona un glosario de términos y el código fuente de todos los programas realizados en el presente trabajo, lo cual representa una valiosa aportación para cualquier programador.

CAPITULO 1

JUEGOS DE VIDEO: PASADO, PRESENTE Y FUTURO

CAPITULO 1
JUEGOS DE VIDEO: PASADO, PRESENTE Y FUTURO

1.1. HISTORIA DE LOS JUEGOS PARA COMPUTADORA

Los juegos para computadora actuales son tan sofisticados que algunos prácticamente son películas animadas con imágenes y sonidos muy realistas. Por supuesto son resultado de una evolución que como todas las que implica la informática se ha desarrollado de manera vertiginosa.

Los primeros juegos fueron desarrollados en los años cuarenta, debe recordarse que en ese entonces las computadoras eran grandes estructuras que requerían de más de un operador y acababan de evolucionar de los tubos al vacío a los transistores, la información era suministrada a la máquina por medio de tarjetas perforadas. Estos programas no eran propiamente juegos, el reto era que la computadora imprimiera una imagen como un logotipo, una caricatura e incluso una foto, proporcionando un paquete de tarjetas perforadas al computador que contuviera una secuencia de letras que pudiera asemejarse, al imprimirse, con la imagen deseada, algo así como hacer un retrato con semillas donde cada letra es una semilla distinta. Debe suponerse el arduo trabajo requerido para realizar una imagen de este tipo.

Pudiera uno preguntarse ¿cuál es la diversión de este esfuerzo?, la respuesta es simple, para muchos programadores el solo hecho de realizar un programa para el esparcimiento en sus ratos de ocio que no implique largas nóminas, cálculos muy precisos o bases de datos complejas, es una diversión por si misma, también está el hecho de considerar que muchos videojuegos son difíciles y, sin embargo, se juegan por espacio de varios minutos de tensión continua.

AJEDREZ

Los juegos de ajedrez por computadora también empezaron a desarrollarse muy pronto, a finales de los años cuarenta y a principios de los cincuenta. Debido a lo complejo del análisis fue necesaria más dedicación para la elaboración del programa y no tan solo en los ratos libres.

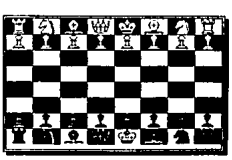


FIGURA 1.1. Tablero simple de ajedrez.

El primer juego de ajedrez por computadora fue desarrollado para ser jugado en un computador llamado Maniac 1, en los Alamos, Nuevo México, en 1956. Para hacer la tarea más sencilla, la máquina utilizaba un tablero con solo treinta y seis cuadrados en lugar de los sesenta y cuatro del tablero normal. El computador tardaba alrededor de doce minutos en hacer cada movimiento, y su fuerza de juego se comparaba a la de un humano que hubiera jugado alrededor de 20 partidas, de hecho un principiante.

Un año después en 1957 se programó a una computadora potente, una IBM 704 para jugar hasta el final el primer juego "real" de ajedrez por computadora, en un tablero de tamaño normal. El programa era diez veces más largo que el anterior pero el resultado siguió siendo el de un jugador amateur.

Todavia en la actualidad, los mejores jugadores pueden ganarle a los mejores programas de ajedrez (recuérdese el caso de Kasparov campeón mundial que le ganó en 10 jugadas a uno de los mejores programas de ajedrez, el cual fue hecho por una empresa profesional).

JUEGOS DE MESA

Los juegos familiares o de mesa son otro tipo de juegos más fáciles de jugar y por lo tanto de programar (por ejemplo, las damas chinas), éstos también se empezaron a desarrollar en los inicios del desarrollo de los videojuegos. Como en estos juegos es posible conocer de antemano todos los movimientos que pudiese realizar el adversario, una computadora bien programada puede encontrar siempre el mejor movimiento posible, y le ganará al ser humano en cualquier circunstancia, a menos que el programador tenga piedad del jugador, e instruya a la computadora para hacer errores ocasionales.

JUEGOS DE AVENTURA

Probablemente fue el desarrollo de juegos especiales para computadora, en lugar de las adaptaciones de los juegos conocidos lo que provocó el verdadero auge de las computadoras. A este tipo de juegos se les conocia como juegos de "aventura".

En esta clase de juegos el programador crea un mundo imaginario, ya sea realista o fantástico e incluso la combinación de ambos elementos, en el cual se le permite al usuario viajar de un lugar a otro para obtener y utilizar objetos en la búsqueda de un fin determinado. La forma de como llegar al objetivo no es explicada por el programa, el usuario debe ingeniárselas para conseguir su meta.

Entre los primeros juegos de aventuras se encuentra "Aventura", éste fue desarrollado por expertos en computación en Estados Unidos a finales de los años setenta.

Esta tesis está enfocada a este tipo de juegos, la dificultad de programar un juego de aventuras es comparable a la de realizar caricaturas. Deben hacerse muchas secuencias de dibujos para crear las animaciones, elaborar las pantallas de fondo, agregar música y efectos especiales, diseñar un libreto que indique cual es el objetivo y con que elementos se cuenta y finalmente crear el programa que integre todos los factores anteriores.

En las figuras 1.3 se observan dos escenas del juego de aventura *FULANO* desarrollado por nosotros (obviamente los autores de esta tesis), como diversión.

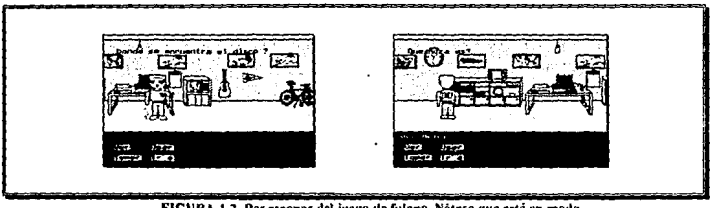


FIGURA 1.2. Dos escenas del juego de fulano. Nótese que está en modo CGA (320x200, 4 colores) y que está en español. ¿Cómo se sabe que el protagonista es un estudiante de la ENEP?. Véase la escena 2 donde se observa que en la mochila está escrita la palabra ENEP.

Este juego tiene 4 opciones principales: Ir a, Ver, Tomar y Usar. El personaje principal es un estudiante de la ENEP Aragón conocido simplemente como *Fulano* (nunca se nos ocurrió un nombre mejor), el cual tiene como objetivo salir de su cuarto e ir a la escuela, sólo que existe un pequeño inconveniente; la puerta de su cuarto está cerrada y no encuentra la llave. Nuestro descuidado héroe reconoce gran parte de los objetos del cuarto, puede tomar unos cuantos e incluso usarlos en algunas ocasiones. Definitivamente no compite con juegos comerciales, pero representó una gran experiencia y fue muy útil como herramienta de aprendizaje.

JUEGOS DE VIDEO

Muchos juegos caseros tienen como antecesores a las grandes computadoras. Por ejemplo, existían unos dispositivos mecánicos conocidos como máquinas del "millón". Estos juegos de salón consistían en un principio de muelles mecánicos y sensores eléctricos, e hicieron su aparición hace mucho tiempo. Al principio de los años setenta, surgieron los equivalentes electrónicos. Estos juegos tenían una pantalla como la del televisor, y el jugador utilizaba unas palancas de control para mover la raqueta, o disparar un rayo de luz.

El americano Nolan Bushnell se considera el padre de las máquinas de video. Desarrolló un prototipo de un juego de video llamado "Computer Space", una batalla entre un platillo flotante y una nave espacial, allá en el año de 1971. Bushnell hizo a continuación dos juegos clásicos más: el videotenis, donde dos jugadores golpeaban una pelota de lado a lado a través de la pantalla, y el Breakout, el juego en el que usted destruye un muro, de ladrillo a ladrillo. Bushnell siguió hasta fundar Atari, una de las empresas más grandes de juegos de video.

Al principio, estos sencillos juegos de video se encontraban en los bares y salones de juegos; más tarde se desarrollaron versiones para su uso en la casa. El primer juego para televisión fue producido por Magnavox, en 1972. Consistía en varias manchas de color que representaban el tablero de juego, en forma de recorte y una sola mancha blanca que le disparaba a los diferente blancos. El ping-pong aunque más sencillo se volvió más popular. En 1975, Atari produjo una nueva versión llamada "Pong". La computadora podía jugar únicamente Pong, pero sus sucesores consiguieron que pudiera jugar a otros juegos del tipo pelota y raqueta.

A finales de los años setenta, los juegos de televisión ya eran muy populares. Las computadoras se conocían como "dedicadas", ya que sólo podían realizar la tarea para la que fueron diseñadas, es decir, si era un juego de tenis exclusivamente podía jugar tenis, no era posible que el usuario escribiera un programa que, por ejemplo, realizara sumas o escribiera un mensaje. Después se crearon las computadoras caseras de propósito general en las que se podía llevar el presupuesto familiar y en los ratos de ocio, tal vez jugar un rato.

Los dueños de los juegos de salón han tratado de estar siempre a la vanguardia en cuanto a tecnología y atracción de los videojuegos, por lo que en cuanto la computadora casera tiene una versión parecida o igual a la de los juegos de salón (conocidos en el argot juvenil como "maquinitas"), lanzan una versión mejorada o nueva de cada tipo de juego. Así que en cuanto hicieron su aparición las primeras computadoras caseras con juegos de tenis, o Breakout, la recién creada empresa japonesa Taito, lanzó un juego conocido como invasores del espacio. Las nuevas características de este tipo de juegos (pantalla más llamativa con varios colores y acción continua) llamaron más la atención de las personas incluso que las máquinas del millón. En un lapso muy corto se difundieron por todo el mundo. Aunque los juegos para computadora se han ido perfeccionando, es muy difícil que alcancen a los juegos de salón que también están en evolución continua. Sin embargo, existen juegos que debido a sus características y al tiempo necesario para terminarlos muy difícilmente podrían ser prácticos para los juegos de salón, en los que se prefieren casi exclusivamente juegos de acción continua, nos referimos, por supuesto, a los juegos de aventura.

1.2. ORDENADORES DOMESTICOS

Las primeras computadoras baratas aparecieron alrededor del año de 1975. Marcas famosas fueron Altair, Sinclair, Commodore, Atari y Timex, con sus respectivos modelos. Aunque estas computadoras podían realizar programas correr aplicaciones de propósito general (agenda, inventario, gastos caseros, etc.), éstos no eran muy complejos debido a los limitado de las computadoras, por lo que se les consideraban de "juguete" u orientadas a los juegos.

Fue hasta el año de 1981 que IBM lanza al mercado la primer computadora personal, la cual contaba con un monitor CGA y 256 Kbytes en RAM que se inicia la era de las computadoras compatibles (con IBM por supuesto) o PC's clones. Otras computadoras eran más potentes y con

mejores disposición para los juegos, tal es el caso de la sorprendente Amiga de Commodore, la computadora Atari ST y tal vez la Macintosh de Apple, pero es la proliferación y el mejoramiento paulatino de los dispositivos periféricos de la computadora IBM, la que la convierte en la más popular mundialmente. Los juegos para las PC's estaban limitados a 4 colores cuando mucho y a diferencia de las computadoras dedicadas se utilizaba el teclado para mover al jugador en lugar de la palanca de control de los videojuegos.

Posteriormente fueron apareciendo dispositivos periféricos tales como el ratón, la palanca de juegos, las tarjetas de sonido, los tarjetas de video mejoradas EGA, VGA y SVGA, así como computadoras de mayor velocidad y capacidad. La programadores de juegos se fueron organizando y se crearon corporaciones dedicadas exclusivamente al desarrollo de los juegos para computadora. Nombres de compañías famosas elaboradoras de juegos son Sierra On-line, Accolade, Lucas Films, Virgin Games, Delphine Software, etc.

1.3. EL PRESENTE

Año de 1994...

Existen infinidad de juegos para computadora entre los que se cuentan simuladores, juegos de aventura, juegos de acción, juegos del tipo Arcade (o sea, el equivalente a las maquinitas). La mayoría de los juegos utilizan 256 colores lo que permite efectos casi fotográficos. Algunos juegos utilizan las resoluciones mejoradas de los monitores generalmente hasta 640x480 pixeles por 256 colores. La velocidad de los procesadores son explotadas al máximo por juegos que utilizan el modo protegido (el cual es una característica que permite direccionar toda la memoria sin necesidad de programas especiales y se utilizan casi sin desperdicio los recursos del microprocesador). Los juegos tridimensionales en tiempo real son muy buenos, algunos bastante realistas.

Los recursos multimedia (integración de voz, imágenes, sonidos y datos) ya dejaron desde hace algún tiempo de ser experimentales y se aplican en juegos donde los personajes hablan y se mueven con mucha fluidez y naturalidad, teniendo como fondo imágenes fotográficas con música y sonido estéreo.

Otros diseñadores de juegos empiezan a poner características de realidad virtual a sus programas (se dice que mientras más factores "engañen" a los sentidos, haciéndoles creer que realmente están viviendo lo que sucede en la computadora hay un grado mayor de realidad virtual), aunque por el momento, estos factores sólo afectan el sentido de la vista y del oído.

En medio de este alarde de tecnología dedicada al desarrollo de los juegos para computadora, se hace latente una realidad inmediata, prácticamente TODOS los programas de juegos profesionales son extranjeros (parece que existe una sola excepción, a esta regla. UNA única empresa mexicana que elaboró UN solo juego de fútbol). Aunque existen juegos traducidos al español (traducidos en España casi siempre), es desalentador que no haya más corporaciones mexicanas de computación que diseñen juegos aunque sea de prueba. Varias preguntas surgen y giran alrededor nuestro, ¿No es negocio realizar juegos para computadoras?, ¿no existe la tecnología necesaria en México para realizar un juego?, ¿no

existen entonces las bases educativas para realizar un proyecto de este tipo?, ¿Hay desinterés en los programadores mexicanos por realizar juegos para computadora?. Para la primera pregunta se deberia realizar una encuesta y/o una prueba de campo para saber si es costeable realizar un juego, para las demás nuestra experiencia como programadores y la recopilación de todas las expectativas de gente relacionada directa o indirectamente con la computación me permite decir que SI. Si hay tecnologia, educación, y ganas por realizar juegos para computadora, por lo que me sorprende que no exista la iniciativa en México por formar y ¿por qué no? obtener una parte del mercado que hasta ahora han acaparado las casas de computación extranjeras. Resolver de una vez por todas las interrogantes planteadas es uno de los objetivos primordiales de esta tesis (así que sujétense bien los cinturones por que esto se va a poner bueno).

1.4. EL FUTURO

El futuro de los juegos para computadora ya esta aquí, como siempre es cuestión de esperar a que bajen los precios del costosísimo equipo empleado en la realidad virtual, para que pueda ser adquirido por los usuario de computadora ávidos de nuevas sensaciones. Es, efectivamente, la Realidad Virtual el candidato más viable para que ocupe el palco de honor en el desarrollo de los juegos de video. Supongan una preparación preliminar antes de jugar, utilizar un guante sensible al movimiento, colocarse una especie de casco el cual cubre los oídos con un par de audífonos estereofónicos y los ojos con dos pequeños monitores planos de televisión, con un ligero defasamiento en la imagen para dar efecto tridimensional, ¿suena incomodo e innecesario?. Pues bueno, este equipo nos puede llevar a mundos tridimensionales increíbles, donde no sólo es ver y escuchar sino también "casi tocar" los objetos. Implica poder mover la cabeza libremente sin estar atados al área especifica del monitor de la computadora. Es tan real como estirar la mano físicamente y observar una gráfica de nuestro brazo atravesar paredes en el país fantástico elegido.

Las aplicaciones de la realidad virtual no se reducen de ninguna manera al esparcimiento, tiene una amplia de aplicaciones actuales y futuras en las áreas de medicina, arquitectura, diseño, proyectos espaciales, etc.

¿Cuál es la siguiente meta de los programadores de juegos?, bueno, pues la respuesta a esta pregunta está en función de los nuevos avances tecnológicos que se vayan desarrollando con el tiempo y que seguramente tratarán de llevar al aficionado de los juegos por computadora a nuevos universos de imaginación, donde atrapen, en lo posible, todos sus sentidos que no sólo vean y oigan sino también huelan e incluso sientan que están combatiendo a seres fantásticos, con los increíbles poderes con los que son dotados para rescatar a una princesa en peligro o salvar al universo de su destrucción. En otras palabras, probablemente la meta que siempre han intentado alcanzar los juegos de video ha sido la de convertir (aunque sea temporalmente) al mortal común y corriente en el héroe de sus sueños, sin necesidad de estar dormido ni drogado (recuerda di no a las drogas y juega con moderación).

1.5. TIPOS DE JUEGOS PARA COMPUTADORA

Existen muchos tipos distintos de juegos, por lo que es probable que haya uno para cada tipo de gusto. Hay, incluso, diferentes versiones del mismo juego, dependiendo de la edad o habilidad del usuario.

1.5.1. Juegos familiares

Se refieren a los populares juegos de mesa adaptados para poder ser jugados en la computadora. Ejemplos típicos son: Ajedrez, damas chinas, cartas, dominó, Backgammon, Turista, Monopolio, etc.

Existen algunas ventajas de jugar con la computadora en lugar de la forma tradicional:

- La computadora siempre está dispuesta a jugar. Estos juegos son generalmente para dos o más personas, pero que tal si no hay nadie que pueda jugar con usted. La computadora nunca se negará a jugar.

- La computadora le ahorrará esfuerzo. Es decir, reacomodará las fichas, barajará las cartas, llevará la puntuación, etc.

- Nunca hará trampas. No se tendrá que preocupar si la computadora lleva un as extra bajo la manga, ni que mueva las fichas mientras usted está volteado (a menos, por supuesto, que el programador del juego así lo hubiese querido). El programa incluso le advierte si es que está realizando algún movimiento ilegal o erróneo.

- Le puede servir de maestro. Algunos programas pueden sugerirle movimientos, estrategias, le permitirán reacomodar las fichas para analizar jugadas específicas y será posible cambiar el nivel de dificultad para que vaya mejorando paulatinamente su nivel de juego.

Por supuesto, hay muchas personas que seguirán prefiriendo el póker convencional, por que para ellos la emoción del apostar con dinero real no es lo mismo que apostar puntitos en la pantalla (y claro, es más difícil llevar una computadora en el bolsillo del pantalón).

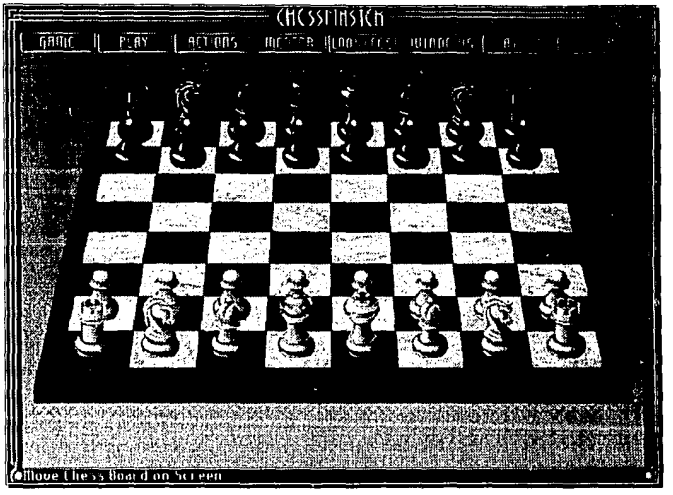


FIGURA 1.3. Chess Master 4000. Uno de los mejores y más completos juegos de ajedrez actualmente.

1.5.2. Juegos de acción

Son el tipo juego más buscado por los usuarios jóvenes. Estos juegos ponen en prueba la habilidad y la práctica psicomotriz. Ejemplo típicos son: Pacman, Mario Bros, Street Fighter, Double Dragon, Mortal Combat, Galaxian, etc. (¿no le suenan conocidos?, bueno, pregúntele a cualquier *niño* entre 5 y 30 años, él le sabrá explicar con lujo de detalles el objeto del juego).

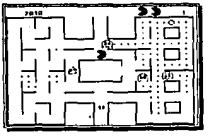


FIGURA 1.5. Pacman, probablemente es uno de los más representativos juegos para computadora.

En los juegos de manos y ojos se dispone generalmente de una palanca de control para mover al jugador y desde dos hasta seis botones para controlar las acciones que se pueden realizar (disparar, dar patada, saltar, dar golpes especiales, utilizar armas, etc.) Se pueden jugar entre uno, dos, e incluso tres jugadores simultáneamente, de forma cooperativa o todos contra todos.

Una muestra tipica de que un juego ingenioso puede resultar bastante popular es el *Tetris* (Ver figura 1.6). El objeto de este juego ruso es acomodar las diferentes figuras geométricas que van cayendo en un orden aleatorio, evitando dejar huecos en el proceso. Si se logran acomodar las figuras de modo que encajen los mejor posible se van eliminando las lineas en las que no haya huecos, dejando de esta manera, más espacio para maniobrar. Las figuras sólo pueden moverse a la izquierda, a la derecha o rotarse, y aumenta la velocidad con la que caen conforme se cambia de nivel. Actualmente existen muchas versiones de *Tetris* algunas son para dos personas, otras que corren bajo ambiente de *Windows*, y algunas más que son tridimensionales, eso sin contar a los juegos que se basan en sus principios básicos.

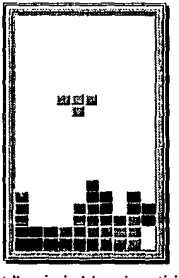


FIGURA 1.6. Pantalla principal de un juego típico de tetris

El objeto de los juegos de acción es derrotar oponentes cada vez más dificiles o terminar más niveles de juego. En los primeros, la llave para derrotar monstruos, naves u otro tipo de enemigos es encontrar el mayor número de combinaciones de botones con movimientos de la palanca de control, esto es porque con ciertas combinaciones se activan golpes o armas especiales, las cuales nos ayudan a acabar más fácilmente con la amenaza que nos preocupa. Esto no es tan sencillo como pudiera creerse, sobre todo cuando hay más de dos botones. En los juegos con varios niveles se tiene un objetivo (comerse todas las pastillas de un laberinto, acomodar piezas, destruir una pared con una pelotita, etc.) el cual se vuelve más dificil de alcanzar con cada nuevo nivel (laberintos más intrincados y con un mayor número de enemigos; un mayor número de piezas que se mueven más rápido y que son más dificiles de

acomodar; paredes más sólidas que requieren más golpes y una pelota que se mueve con mayor rapidez, etc.). En estos casos sólo hay una cosa que hacer: practicar y practicar.

1.5.3. Juegos de simulación

Los juegos de simulación toman una actividad real y la tratan de representar en la computadora de modo que se parezca los más posible a la realidad. Existen simuladores de casi cualquier tipo: de carros, de aviones, de golf, de billar, de la vida en la ciudad, de una casa de bolsa, e incluso de un transbordador espacial.

Los aficionados de este tipo de juegos pueden encontrar que también representan una forma de diversión didáctica. Efectivamente, la NASA, las escuelas de aviación, las escuelas de medicina, y los científicos utilizan programas de simulación más avanzados para adiestrar a los futuros pilotos, astronautas, cirujanos y químicos con la seguridad que no se perderán vidas ni costosos equipos por un error de novato.

1.5.4. Juegos de aventura

Los juegos de aventuras podrían considerarse como juegos de simulación, sólo que en lugar de tomar un tema de la realidad crean un mundo propio y fantástico. Se puede platicar con los objetos, los animales y las plantas, tener poderes increíbles, resolver enigmas, y hacer combinaciones de objetos para obtener algo más. En este tipo de juegos se tiene que alcanzar una meta y para ello pueden seguirse diferentes caminos o utilizarse diferentes personajes. A veces se lleva un marcador que nos indica cuantos puntos llevamos y cuantos nos faltan Cada punto representa una acción hecha correctamente y para llegar al final del juego casi nunca es necesario tener todos los puntos. Saber que acción realizar no siempre es fácil, conforme se avanza en el juego se recolectan más y más objetos de todos los lugares diferentes que se han recorrido. Algunos objetos se utilizan y otros pudieran no servir para nada Si tomamos en cuenta que el primer objeto puede ser útil en el primer escenario y el último objeto en el primer escenario, nos daremos cuenta que tenemos que recorrer varias veces todos los escenarios para utilizar todos los objetos útiles y así acercarnos cada vez más al final. En conclusión, hay que llegar como sea, pero llegar.

1.5.5. Juegos educativos

¿Su hijo se la pasa todo el día jugando Mario Bros y no quiere hacer la tarea? ¿Conoce perfectamente a todos los personajes de los caballeros del zodiaco pero no sabe leer ni escribir? ¿incluso usted, ya se cansó de intentar aprender inglés por los aburridos métodos tradicionalistas?. ¡Ajá!, entonces lo que usted necesita son juegos educativos. Deje que su hijo combata monstruos teniendo como única arma el resolver problemas o contestar preguntas antes de ser devorado. Deje que Seiya, Yoga y Chiru (los caballeros de pegaso, cisne y del dragón) le muestren a su hijo cuáles son las vocales, consonantes y palabras básicas, mientras disfruta (¡por fin!) del fútbol matutino. Aprenda inglés a su

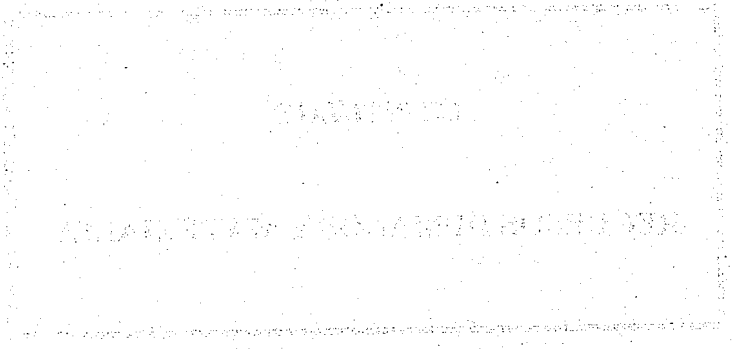
propio ritmo, escuchando las pronunciaciones y sintaxis correcta que le proporciona la profesora 80486, mientras se lleva un registro de su avance sin que usted tenga el temor de ser reprobado otra vez. ¿Parece comercial?. Bueno, es otra manera de explicar las aplicaciones de los juegos educativos.

Los juegos educativos enseñan de una forma amena, interactiva, incluyendo muchas veces gráficas bastante buenas y representativas, avanzan al ritmo del usuario, llevan un registro de los temas que ya se vieron (si es que no quiere seguir el orden estándar), puede hacerle pruebas para validar el nivel de sus conocimientos e incluso sugerirle que temas repasar, y se puede tener la seguridad que no nos aventarán el borrador si estamos platicando ni nos pegarán con una regla en la punta de los dedos.

Con las nuevas tendencias tecnológicas, la Enseñanza Asistida por Computadora (EAC) representará un complemento importante en la educación. Los juegos para computadora, que siempre han sido atacados por su influencia negativa en el rendimiento escolar infantil, han sido redimidos por los nuevos programas de juegos educativos. Ahora, en lugar de distraer, ayudan a enfocar la atención en el aprendizaje incluyendo procesos didácticos y entretenidos, tales como un perrito que les ayuda a contar, un payaso que les enseñe las formas geométricas, o un personaje más conocido (tal vez Cantinflas), que les platique la historia universal.

CAPITULO 2

RECURSOS HUMANOS Y MATERIALES



CAPITULO 2
RECURSOS HUMANOS Y MATERIALES.

En la actualidad, la cantidad de personas y equipo utilizado para elaborar un juego de video es impresionante. Se podria comparar con la producción de una pelicula de cine ya sea real o de dibujos animados, en la que intervienen todo tipo de personas y equipo material. Este capitulo muestra la importancia de cada uno de los elementos que intervienen. Se encuentra dividido en dos módulos principales :

- Personal de trabajo, que son los recursos humanos.
- Equipo necesario, que son los recursos materiales.

2.1. PERSONAL DE TRABAJO.

¿ Primero, qué se va a hacer ?

Lo que se va a hacer es una pelicula computarizada con interfase hacia el usuario, una historia en la que las acciones a tomar y la secuencia de éstas dependa sólo del que maneja el juego. Una aventura en la que se visiten escenarios, se conozcan personajes, se tomen objetos y se utilicen, y tenga un inicio, un clímax y un desenlace exactamente igual que una pelicula. Entonces ¿ que se necesita ?

Como se mencionaba, el hacer una historia gráfica no implica sólo a los programadores, sino a un equipo de trabajo más complejo que es descrito a continuación.

2.1.1. Escritores.

Lo primero y mas importante que se necesita es la historia.

Debe ser una historia escrita especialmente para un juego de video. Las historias escritas para películas, novelas u obras de teatro son para verlas, las escritas en libros son para leerlas. Las escritas para computadora deben causar el mismo impacto que una pelicula pero además estar hechas para interactuar con el usuario.

El género no importa, pueden ser de aventuras, suspenso, terror, ciencia ficción, etc. Los requerimientos de los personajes, caracteristicas, aspecto, nacionalidad, personalidad, no importan, todas éstas les serán proporcionadas por medio de la programación y a bajo costo.

Por lo tanto la primera persona que aparece en escena es el "escritor" o los "escritores". No hace falta mencionar que la originalidad de la historia repercute en el éxito de ésta dentro del mundo informático. Se trata hasta cierto punto de que se le complique un poco al usuario llegar al final. Se trata de que el usuario viva la emoción de la historia. Se trata de que se sienta el protagonista y sea el que

tenga que tomar las decisiones importantes. Se trata de divertir, de distraer, de relajar, de hacer pensar, de meter al usuario en un mundo fantasioso que no existe en la realidad pero que ahora está a su alcance. Y puede andar, correr, saltar, agarrar, ganar, perder e incluso hasta morir, y por supuesto revivir mil veces.

Se puede decir que el contenido de la historia es lo más importante en este tipo de juegos, puesto que pueden estar los mejores programadores del mundo metidos en una historia pobre y no sacarle el jugo que se espera de ésta.

Actualmente existen escritores especializados en este tipo de historias, pero también existen los que se dedican a hacer adaptaciones de películas exitosas al mundo de los video juegos, las cuáles por lo general son para juegos de acción (arcades).

Para la realización de esta tesis tuvimos que ser los escritores, - área en la que no estamos especializados - puesto que contar con un escritor de historias implica pagarle a una persona, y es algo que por lo pronto no podemos costear.

2.1.2 Músicos.

La incorporación de poderosas interfases de sonido a los ordenadores, ha hecho que éstos compitan con equipos especializados de sonido. Ahora el ordenador es capaz de reproducir una sinfonía completa y con excelente calidad.

Aquí es donde entra el segundo personaje, el "músico" o los "músicos".

Con música de fondo, las historias gráficas logran crear un ambiente que se escucha sincronizado con las acciones gráficas que se observan en pantalla.

No es necesario que los músicos sepan mucho de computación. Se limitarán a componer su música con los instrumentos electrónicos usuales de música y por medio de una interfase, se pasará a un código ejecutable por programas de música que maneja el ordenador. La calidad de la música será la misma que si se reprodujera por un instrumento de música especializado.

En la actualidad, todos los juegos, independientemente del su tipo, tienen fondo musical. Si un juego no lo tiene simplemente no llama la atención, y es por eso la actual preocupación de las casas productoras de software de entretenimiento de contratar músicos especializados.

En nuestro caso para los fondos musicales del juego tomamos archivos públicos de música que ya realizaron otras personas y los incorporamos.

2.1.3. Dibujantes.

La parte mas importante de un juego es la calidad con que cuentan los gráficos presentados. Se requiere de trabajos de dibujo profesionales.

En el caso de las historias gráficas se utilizan dibujos de cada movimiento de un personaje y de cada escena en la que se desarrolla la acción. A veces hay tantas escenas y tantos personajes, que el

número de dibujos necesarios para lograr el juego completo, se podria comparar al de dibujos necesarios para hacer una caricatura de televisión. Es decir podriamos hablar quizás de miles de bosquejos.

Al igual que en los dibujos animados, se requiere de "dibujantes" que serán los encargados de la parte mas atractiva del juego.

Los dibujantes se limitarán sólo a hacer los dibujos. Para ello contarán con una cantidad increible de herramientas de dibujo por ordenador que sustituye a su equipo de trabajo habitual. Se olvida por completo de hojas, pinceles, pinturas, lápices, y sobre todo de errores.

Se contara con programas que sustituyen la hoja por la pantalla, el pincel por el ratón, las pinturas por una paleta digital de colores inagotables y muy ricos, ya que se dispondrá de millones de colores que al ser digitales nunca se acaban.

Se puede pintar con un lápiz digital sobre la pantalla o sobre una tableta que introduce el dibujo a la computadora. Para los que tienen mas práctica, encontrarán en el ratón una valiosa ayuda.

2.1.4. Camarografos.

Otra forma de meter personajes y paisajes a la pantalla, es filmándolos. Es un método mucho mas rápido, pues se filmarian personajes y paisajes reales. Esta vez se trata de una historia gráfica digitalizada. Se sustituyen los dibujos por imágenes que son mas fáciles de hacer y mas rápidas. Las secuencias de los personajes deberán presentar los mismos movimientos que si se hubieran dibujado.

También se contará con un número enorme de programas que capturan en el ordenador, secuencias filmadas directamente de la cámara.

Se requerirá de actores vestidos de la manera que lo indique la historia del juego, y de escenarios reales o artificiales que también sean necesarios. Es decir igual que si se hiciera una pelicula para televisión.

Dado que se trata de imágenes digitalizadas, la cantidad de memoria en disco utilizada es enorme, por lo que ese tipo de juegos se manejan actualmente en CDROM.

2.1.5. Animadores.

Una vez que están hechos los dibujos, los animadores se encargan de armar las distintas secuencias que intervendran en una escena. Su función básica es darle "vida" a los personajes, eligiendo la secuencia de imágenes más representativa para un personaje. Aunque esto también puede ser hecho por los dibujantes, en muchas empresas desarrolladoras de juegos se prefiere hacer la división en estos dos grupos de trabajo. La razón para tener un grupo de dibujantes y otro animadores puede ser aumentar la velocidad de desarrollo del programa, otra razón tal vez sea que algunos dibujantes realizan sus gráficas directamente en una hoja de papel, la cuales digitalizan posteriormente, entonces los animadores se encargan de recortan las imágenes y ordenarlas en los grupos de archivos adecuados.

La función de los animadores es más relevante cuando no hay dibujantes. Este caso se puede dar cuando todo el juego se digitaliza, por medio de una cámara de video, un dispositivo de entrada como la

videoblaster y un software que genere una sucesión de imágenes como Video for windows (éste es el caso de esta tesis). El animador, entonces, elegiría las imágenes apropiadas, las recortaría (y si es necesario las retocaría para que se vieran mejor), y las guardaría en los archivos adecuados. Asimismo, puede participar en la creación del libreto, aunque esta tarea sería más propia para el director del juego.

2.1.6. Programadores.

Los programadores son la parte principal del equipo de trabajo, pues son los que generan el código que dará vida a los protagonistas de los video juegos.

Existen en este rango programadores especializados en música, en animación, en sonido, en graficación, en manejo de archivos, de memoria, de interfase con el usuario, etc. Cada uno trabajara por su cuenta en una parte del juego y después serán juntados por un programador en jefe.

Cuando se trabaja por primera vez en elaboración de juegos de video, los programadores se encuentran en una situación un poco difícil No existe software que sirva para que un usuario haga un juego de video completo, y menos tratándose de historias gráficas. Por lo tanto dentro de los programadores se formarán equipos para trabajar en cuatro módulos principales que son:

- Un editor de duendes y animaciones. Que sirve para crear a los protagonistas y objetos de la historia, así como darle la animación necesaria para cada escena. Este módulo se abordará con mayor claridad en el capítulo 7 (Editor de duendes y animación).

- Un editor de música. Utilizado para generar y reproducir música de fondo para las diferentes escenas, y sonido para cada diálogo hablado, sonidos de objetos y efectos especiales. Este tema se manejara en el capítulo 8.

- Un libreto. Que es el programa con el que se armara por completo cada escena, desde música hasta animaciones.

- Y por último un programa ejecutor Que es el que se encargara de ejecutar las escenas. Se puede decir que es el juego en si. Tanto el programa ejecutor como el libreto se explicarán con mayor claridad en el capítulo 9.

Cuando se tienen estos, módulos el trabajo de los programadores disminuye notablemente, ya que si se quiere hacer otro juego de video se utilizarán los programas anteriores. Esto sería equivalente a hacer varias cartas con en mismo formato, en las que lo único que cambiará serán los datos.

2.1.7. Director.

El papel quizás mas difícil es el de coordinar todas las acciones. El director al igual que en una película se encargará de supervisar que todo lo que se haga esté bien.

Partirá desde la aceptación de la historia a crear, la elaboración de los dibujos y animaciones, y la elaboración final del juego.

También se le conoce como programador en jefe.

2.2. EQUIPO BASICO.

Cuando se quiere hacer un juego de video se debe hacer una pregunta importante. ¿ Queremos hacer un juego de video o queremos hacer un intento de juego de video ?

La respuesta es obvia (esperamos) puesto que no se trata de perder el tiempo. Lamentablemente para hacer un juego que pueda competir en el mercado actual, se necesita un equipo muy especializado. No se puede hacer con una computadora que no cuente con los recursos necesarios. Afortunadamente el costo de dicho equipo se ha decrementado en los últimos años y es mas accesible tener acceso a ellos. A continuación se da una breve explicación del equipo mínimo que se necesita para elaborar un juego de video, y después una tabla actualizada de los costos de cada elemento.

2.2.1 Tarjeta de sonido Sound Blaster.

Las tarjetas de sonido fueron introducidas en el mercado hace aproximadamente cinco años, para utilizarse especialmente en juegos de video. La conocida tarjeta ADLIB fue la primera de este tipo. Después surgieron tarjetas de otras marcas y modelos como Sound Blaster, Sound Blaster Pro, Pro Audio Spectrum, Gold, MT 32/LAPC I, Sound Canvas, Tandy 3 PC, Roland, Sound Master, Thunder Board, etc. Actualmente el mercado de tarjetas sigue creciendo sin límites y la calidad de su sonido es cada vez mejor.

La tarjeta Sound Blaster cuenta con una ventaja muy importante, para empezar desde el momento que salió a la venta, contaba ya con muchisimos programas que la utilizaban, esto es por la compatibilidad que guarda con la tarjeta Adlib. Adicionalmente se produjeron nuevos programas para esta tarjeta. Dado que fue de las primeras tarjetas que salieron a la venta, casi todo mundo la maneja. En otras palabras, actualmente todos los juegos de video que se hacen cuentan con una interfase para manejar la Sound Blaster. Por si fuera poco, las nuevas tarjetas de sonido guardan compatibilidad con ella, es decir se puede configurar el programa que las utiliza como si tuviera la SB. De esta manera puede asegurar que los códigos presentados para el uso de la tarjeta de sonido, funcionarán casi en cualquier tarjeta.

La Sound Blaster es capaz de manejar sonidos y música.

Se entiende por música a una secuencia de notas musicales introducidas al ordenador por medio de un instrumento musical especializado y que son almacenadas en valores de notas y tiempos.

Se entiende por sonido al muestreo de ondas herzianas producidas por cualquier tipo de fuente (incluyendo instrumentos musicales) y que son digitalizadas y almacenadas en valores de amplitud y frecuencia.

Para la demostración del juego que elaboramos como tesis, se requiere del uso de una tarjeta Sound Blaster instalada. El software utilizado para el manejo de la Sound Blaster se manejará en el capitulo 8 (Editor de música).

La Tarjeta sound Blaster se conecta en un slot de 8 bits, por lo que puede funcionar incluso en una XT. Las nuevas tarjetas de sonido de mejor calidad y mas rápidas ocupan un slot de 16 bits.

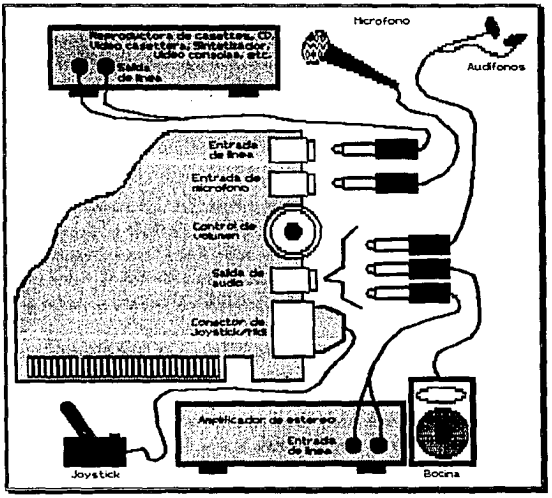


Figura 2.1. Conexión de la tarjeta de sonido Sound Blaster.

ESPECIFICACIONES GENERALES DE LA SOUND BLASTER.

- 11 voces de música FM. Esto significa que al mismo tiempo soporta el equivalente a 11 instrumentos musicales. Los sonidos son generados por frecuencia modulada. Los códigos manejados son de un formato muy pequeño y ocupan poco espacio en disco.

- 1 canal de 8 bits de voz digitalizada. Reproduce un sonido muestreado en un rango de 4KHz a 44.1KHz. Los sonidos pueden ser los generados por cualquier tipo de fuente que exista. Por ejemplo animales, voz humana, efectos sonoros, etc. El formato manejado para este tipo de

muestreos aumenta en proporción a la frecuencia utilizada y por supuesto ocupan mucho espacio en disco.

- Modos de transferencia directo y DMA. Es decir el modo de transferencia puede ser utilizando el CPU y no utilizando el CPU. Esto es la tarjeta puede estar trabajando en algún sonido mientras el CPU continua con un procedimiento muy aparte.

- Compresión de datos. La misma tarjeta comprime (por hardware) los sonidos grabados. Los datos originales son de 8 bits y se pueden comprimir a 4 bits, 2.6 bits y 2 bits. Una vez comprimidos no se pueden descomprimir. Es evidente que la calidad del sonido disminuye conforme se comprimen más los datos que lo forman.

- Amplificador interno con volumen. Cuenta con salida máxima de 4 watts por canal y 4 ohms en estéreo. Tiene una conector para bocinas y audífonos, o bien para conectarse a un amplificador externo. Este amplificador tiene su propio control de volumen que a diferencia de otras tarjetas, la provee de un sonido autosuficiente y perfectamente audible.

- Grabación de sonido. Tiene una conector para micrófono a manera de entrada para grabar sonidos. Muestrea con un rango de frecuencia de 4KHz a 15KHz. Es decir tiene un pequeño rango de grabación, pero un amplio rango de reproducción. También cuenta con una línea de entrada que provenga de otro aparato de sonido especializado.
  Su entrada de micrófono es de 600 ohms de impedancia y un rango de 0.004 a 0.7 Vrms.
  Su entrada de línea es de 39 KOhms y un rango de 0,2 a 5 Vrms.

- Interfase MIDI. Cuenta con una base para la instalación opcional de un microchip de interfase con instrumentos MIDI.

- Puerto de juegos. Por último tiene un puerto de 15 pins para conectar la palanca de juegos que es estandar entre los computadores compatibles.

2.2.2. Tarjeta de digitalizadora de video Video-Blaster.

La tarjeta de video ha sido creada para abrir al mismo tiempo una puerta hacia el mundo de la imagen y el sonido.

El objetivo de sus diseñadores ha sido orientar al PC al tratamiento de imágenes de video, para digitalizar imágenes, crear cabeceras, secuencias de animación, y mezclar imágenes con sonido y texto, o sea, el concepto de multimedia por entero.

La tarjeta de video se conecta en un slot de 16 bits de preferencia que se encuentre junto a la tarjeta VGA, ya que se debe conectar un cable desde la Video Blasters hasta la tarjeta VGA que no es precisamente muy largo. Se conecta un cable que va exteriormente de la Video Blaster a la salida de la VGA, donde antes se conectaba el monitor. Luego se conectan todas las entradas de video a los conectores que hay para tal efecto en el mismo cable. Por último se conecta el monitor a la salida se la Video Blaster y se conectan las salidas de audio a los altavoces o equipo de música, esto es porque también cuenta con una tarjeta de sonido incluida. Para la elaboración de la tesis no utilizamos esta ultima salida.

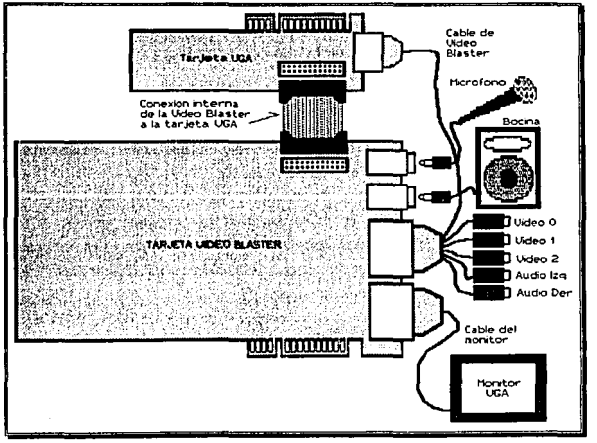


Figura 2.2. Conexión de la tarjeta Video Blaster.

EL SOFTWARE

Junto con la Video Blaster viene el software necesario para manejarla. Cabe hacer notar que es herramienta muy importante para la captura de imágenes puesto que por lo prontó esta fuera de nuestras posibilidades, hacer un software que maneje la tarjeta. Los programas que contiene para windows son :

- Video Kit. Para el manejo de imágenes.
- VB Sound. Para el manejo de sonido.
- VB Setup. Para la configuración de la tarjeta.

Este software es descrito en el capitulo de Software de Apoyo.

La Video Blaster no es precisamente un juguete, su costo actual es muy elevado, y es lógico debido a lo que puede hacer. Está orientada a profesionales que quieran utilizar su PC como herramienta de trabajo para unir el mundo del video con el PC. También es una buena forma de introducirse en el campo de la imagen para los aficionados que desean una opción económica (dentro de las posibilidades del mercado) y poner a su alcance posibilidades que, hasta ahora, sólo estaban implementadas en equipos y configuraciones de un costo substancialmente superior.

2.2.3. Monitor de alta resolución.

La incursión de los juegos de video en el mundo de las computadoras compatibles, ha estado en función de avance que han tenido éstas. Los primeros juegos en modo texto dejaban mucho que desear.

El avance fue relativamente rápido desde los juegos en CGA de 4 colores, hasta los juegos actuales de altaresolucion Super VGA en 16 millones de colores.

La mayoria de los programadores no se van tan lejos todavia. Los juegos de éxito actuales funcionan en un monitor VGA normal y con una resolución de 320 x 200 pixeles a 256 colores. La resolución es buena y los colores son suficientes, pero la razón principal de manejar este modo de video en especial es que se puede todavia optimizar memoria RAM y en disco. Las imágenes de alta resolución ocupan una cantidad enorme de memoria y su despliegue todavia es lento. Es decir mientras no se inventen nuevos métodos de compresión de imágenes y métodos de aceleración gráfica, se seguira usando el conocido modo MCGA de 256 colores.

2.2.4. Scanner (digitalizador)

Una manera opcional de introducir imágenes a la computadora es por medio de un scanner. Es óptimo cuando los dibujantes encuentra dificultad al realizar sus trazos directamente en pantalla. Se pueden hacer los dibujos primero en papel y después capturarlos con el scanner y meterlos en video.

La imágenes "scanneadas" son en tonos de gris. Pero si se cuenta con un scanner de color y una cámara fotográfica, bien se puede sustituir a la video blaster.

Un scanner es un dispositivo de lectura óptica que es capaz de reconocer diferentes tonos en una hoja por medio de un haz luminoso de luz y convertirlo a un formato binario reconocible por la computadora.

En este caso el scanner sólo es utilizado para hacer los juegos de video, no es necesario que este en la computadora que se jugará.

2.2.5 Ratón

Este conocido dispositivo de entrada, suele ser el principal medio de comunicación entre el usuario y la computadora. Suele ser mas práctico que el joystick cuando se trata de juegos de estrategia y de

aventuras. Incluso hay juegos que no funcionan sin este dispositivo, y no dan acceso a ningún otro tipo de control

2.2.6. Joystick (palanca de control)

Indudablemente cuando se trata de simuladores, juegos de acción, arcades, juegos de deportes y de pelea, no hay nada como la conocida palanca de juegos. Esta palanca puede ser sustituida casi en todos los casos por el teclado. A diferencia de consolas especializadas en juegos de video, el control que se tiene sobre este dispositivo en computadoras personales todavia es un poco deficiente, aunque suelen ser mas apantalladores.

2.2.7. Memoria extendida.

Cuando se trata de juegos que utilizan una amplia variedad de personajes, escenarios, sonidos y acciones, la memoria base de 640 suele ser muy poca e insuficiente. Por esto se incorporan rutinas para manejar la memoria extendida o bien la memoria expandida. De esta manera se puede disponer de un recurso muy importante ya que sin el la mayoria de los juegos no funcionarian. En especial los simuladores son los que actualmente le llevan la delantera a otro tipo de juegos en el uso de memoria extendida.

Lo recomendable es tener por lo menos 4 Mbytes en memoria RAM.

Actualmente la memoria se vende en paquetes de chips llamados SIMMS y suelen tener configuraciones de 256 Kbytes, 1 Mbyte, 2 Mbytes, 4 Mbytes y hasta 8 Mbytes.

2.2.8. Computadora PC compatible 386 o 486.

Y por supuesto lo mas importante, un buen equipo. Es recomendable ejecutar juegos en computadoras 386 y 486 principalmente por la velocidad que éstas incorporan. A pesar de que se han optimizado los códigos para manejo de imágenes en 256 colores, éstos siguen siendo muy lentos (comparándolos porsupuesto con consolas de juegos especializadas), esto se compensa al utilizar equipos mas rápidos. Muchos juegos todavia funcionan en equipos 286 pero son increiblemente lentos. A los computadores XT ya los abandonaron.

En especial cuando se trata de generar paisajes en tiempo de ejecución, se necesita la gran velocidad de estos equipos.

Ademas las 386 y 486 soportan el "modo protegido" que es muy utilizado en la actualidad , y que consiste en deshabilitar todas las funciones del DOS e incorporar los procedimientos necesarios para tomar un control completo de los recursos del CPU. Utilizando el modo protegido se hace lo que sea, los programadores hacen "magia" con la computadora.

2.3. EQUIPO COMPLEMENTARIO.

Por lo general para elaborar un juego de video, no basta con el equipo de computo. Al introducir los multimedios en la computadora, involucra conectar equipos externos y ajenos (quizás no tanto) a un ordenador. Si se va a manejar una tarjeta de video es obvio que de alguna manera se tiene que introducir el video. A continuación se presenta la descripción de este equipo extra utilizado.

2.3.1 Equipos de video.

Son conocidos la gran cantidad de aparatos que actualmente se conectan a una televisión. Tales como cámaras de filmar, video grabadoras, consolas de video juegos, señales de cable, etc.

Utilizando la video blaster, prácticamente se convierte el monitor en una televisión. Por lo que se puede conectar a la computadora todo lo que se le conecta a una televisión normal. Obviamente con las ventajas que acarrea el hecho de ser una computadora. Se podria decir que lo difícil es meter la imagen al monitor. Una vez estando la imagen en el monitor, hacemos con ella lo que nos plazca.

Es recomendable adquirir una cámara de filmar y una video grabadora cuando se quieren hacer juegos de video. En el caso de juegos con imágenes digitalizadas, nos servirán para capturar directamente las imágenes y simplemente añadirlas al juego. En el caso de juegos con dibujos, la cámara puede ser una opción para el dibujante, pues quizás le resulte mas fácil dibujar sobre papel y después capturar el dibujo con la cámara para introducirlo al monitor.

La tarjeta Video Blaster soporta los estandar PAL y NTS.

2.3.2. Equipos de sonido.

De igual manera, a una tarjeta de sonido le puede conectar todo lo que se le conecta a un aparato especializado de sonido. En este caso, no sólo se trata de salida de sonido, sino también de entrada. Lo que da acceso al manejo prácticamente de todo lo que produzca sonido. Esto puede ser un equipo completo de sonido, grabadora y reproductora de discos cassettes, reproductora de discos compactos, tocadiscos, micrófonos, bocinas, amplificadores, ecualizadores, etc.

Pero sin lugar a dudas, lo mas importante - tratándose de juegos de video - es que por medio de la interfase MIDI, se pueden también conectar instrumentos musicales. Estos instrumentos obviamente son electrónicos o generan una salida digital acústica. La computadora es capaz de capturar estos sonidos en un formato propio MIDI comprimido, y es capaz de reproducirlo a la perfección.

De la misma manera es recomendable adquirir un equipo de sonido completo, para conectarlo a la computadora.

2.4. COSTOS ACTUALES.

Los equipos descritos anteriormente son los principales para elaborar un juego de video y no podemos prescindir de alguno de ellos.

A continuación se presenta una tabla actualizada el dia 12 de octubre de 1994, en la que se muestran los costos de cada elemento que se utilizo para la elaboración del juego mostrado en la tesis.

Nombre del equipo.	Costo (N$)
Sound Blaster Pro	700.00
Video Blaster	1,500.00
Monitor Super VGA	1,000.00
Scanner H.P. (página completa, colores)	3,000.00
Ratón	70.00
Joystick	80.00
Memoria Extendida (Simm de 1 Mbyte)	200.00
Computadora 486 (Acer)	5,000.00

Tabla 2.1.

CAPITULO 3

LENGUAJES DE PROGRAMACION

CAPITULO 3
LENGUAJES DE PROGRAMACION

La elaboración de un juego para computadora puede ser realizado con una buena idea, ingenio y, por supuesto, conocimiento de un lenguaje de programación de propósito general (Pascal, C, ensamblador, etc.). Sin embargo, cuando se desea elaborar un juego profesional que tenga las caracteristicas para poder comercializarse es necesario, además, tener conocimiento de una gran diversidad de temas, de preferencia lo más profundo posible, ya que para hacer el programa más rápido, fácil de usar y profesional se requiere de programar directamente el hardware (es decir, la parte física de la computadora) y utilizar procesos que no están disponibles en ningún lenguaje de programación por completo que éste parezca ser. Por si fuera poco, mucha de la información necesaria no está disponible en los libros que se venden en México (sean extranjeros o no), por lo que se tiene que investigar, experimentar y desarrollar algún proceso antes de poder ser utilizado. Por ejemplo, supongamos que se desea ver una imagen con formato *BMP* (utilizado comúnmente por *Windows*) pero no se sabe como se guarda en disco. Primero se lee directamente del disco y se despliega en la pantalla en modo gráfico. Se observa basura al inicio y la parte de la imagen que se alcanza a ver está de cabeza, ¿qué se tiene que hacer?, pues se lee desde el final hasta el principio del archivo. Deducciones como esta parecen sencillas, pero los conocimientos "básicos" para su desarrollo son:

Saber ...

- ... como cambiar a modo gráfico.
- ... como leer directamente del disco a la memoria de video.
- ... que es y en que dirección se encuentra la memoria de video.
- ... que es una dirección de memoria y como se puede modificar.
- ... como realizar una lectura invertida de un archivo binario.
- ... como poner limites para que no se escriba en otras áreas de memoria lo que puede afectar al programa o peor ocasionar un paro total del sistema o "crash".

¿fácil verdad?.

En este capitulo se explicarán de forma general muchos conceptos sin los cuales los capitulos subsecuentes pudieran parecer confusos, además puede consultarse un glosario general con la terminologia más usada.

3.1. PASCAL COMO LENGUAJE DE PROGRAMACION

Antes de exponer nuestra justificación, se debe explicar la diferencia entre código fuente y código ejecutable. La computadora no entiende las palabras, letras y números como lo hacemos nosotros, para

poder comunicarnos con ella debemos hacerlo en un lenguaje especial que consiste en ceros y unos y se conoce como lenguaje de máquina. Por otra parte, sería muy difícil y fastidioso elaborar programas en base únicamente a ceros y unos. Por esta razón se diseñaron los lenguajes de programación, por medio de los cuales se puede dar una orden a la computadora escribiendo una frase que nos sea familiar, tal como <<Escribe un letrero que diga "hola" >> y el lenguaje se encarga de transformar esta instrucción en unos y ceros, de forma que sea entendible por la computadora (proceso al cual se le llama compilación). Así, se podría decir que existen dos tipos de programas: el primero llamado código fuente que nosotros podemos entender y el código ejecutable que entiende la computadora (este último mucho más pequeño).

Los lenguajes de programación pueden agruparse en 3 categorías: lenguajes de nivel alto, bajo y medio y ejemplos de ellos son, respectivamente, *Pascal*, *C* y *ensamblador*. Mientras mayor sea el nivel de un lenguaje se necesitarán menos instrucciones para hacer lo mismo, pero no sólo ocupará más código ejecutable (tamaño del programa) sino que puede ser más lento. Para muchas personas, el lenguaje ideal para programar es *C* (lenguaje de nivel medio), porque suponen que siempre será más rápido que uno de alto nivel y más fácil de programar que uno de bajo nivel. Sin embargo, al hacer una comparación de ejecución entre Turbo Pascal y Turbo C (los compiladores desarrollados por Borland) con los mismos procedimientos, las diferencias de tiempo fueron prácticamente despreciables. Además, como es posible integrar un programa en ensamblador (lenguaje de bajo nivel) directamente al programa de Pascal, la rutinas críticas, que requieren de muy alto desempeño y velocidad pueden programarse directamente en el microprocesador. De esta manera, se tiene un lenguaje de programación en el cual las rutinas generales son más fáciles de desarrollar, de entender y mejorar. Y se puede obtener la misma velocidad integrando lenguaje ensamblador. Los autores hemos desarrollado programas en Pascal aproximadamente 5 años, por lo que también se considera la experiencia en el manejo de este lenguaje para utilizarlo. Esta es, por tanto, la justificación para utilizar Turbo Pascal. Pero, esto no implica que debamos desechar el *C* e ignorarlo, por el contrario, se debe conocer este lenguaje lo más posible. Esto no es una contradicción, en párrafos anteriores se explicaba la necesidad de conocer de todos los temas lo más posible ya que muchas cosas deben investigarse. En el caso de *C* hay programas del dominio público y libros de computación, que tienen el código fuente de temas interesantes escritos en lenguaje *C*. Como puedes ver, si no se dispone de ninguna otra fuente de información, es el análisis de un programa de éstos lo que nos puede dar ese conocimiento extra necesario para desarrollar un programa mejor y más profesional.

El compilador de Pascal utilizado en el presente trabajo es Turbo Pascal 7.0 de Borland International.

3.2. LENGUAJE ENSAMBLADOR

Como ya se había mencionado, el lenguaje ensamblador es de nivel bajo y representa el último escalón entre el lenguaje de programación y el lenguaje de máquina. Las serie de unos y ceros que puede

interpretar la computadora se pueden agrupar en grupos de 8 bits o en 2 dígitos hexadecimales. Las instrucciones del lenguaje ensamblador son palabras contar y significativas que nos dan una idea de lo que hacen, se les conoce como *nmotécnicos* (también llamados como *nmónicos*). Estos nmotécnicos pueden tener 2, 1 o ningún parámetro adicional. Por ejemplo:

CODIGO DE MAQUINA	*ENSAMBLADOR*	*EXPLICACION*
10111000 00000101 ó B8 05	MOV AH, 5h	Asigna el valor de 5 al registro AH

3.2.1. Incluir lenguaje ensamblador en un código fuente de Pascal

Existen tres formas de incluir códigos de ensamblador en Turbo Pascal. La primera es por medio de la instrucción *InLine*, la segunda con la instrucción *ASM* (Disponible desde la versión 6.0) y la tercera es declarando una variable de tipo registro y utilizando la instrucción *Intr*. Ejemplo:
Supongamos que queremos cambiar al modo gráfico de 320x200 con 256 colores...

- ... utilizando *InLine*;

```
InLine($B8/$00/$B0/$13/$CD/$10)
```

- ... utilizando *ASM*;

```
ASM
  MOV AH,0
  MOV AL,$13
  INT $10
END
```

- ... utilizando *Intr*;

```
Var
  r : Registers;
.
.
.
r.AH := 0;
r.AL := $13;
Intr($10,r);
```

Obsérvese el signo de pesos ($) que precede a los números, en Turbo Pascal esto significa que se debe interpretar al número como hexadecimal.

La utilización de alguna de la técnicas anteriores depende de la necesidad y preferencia. La primera se utiliza si se quiere dificultad la interpretación del programa a terceros o para ocupar menos espacio.

La segunda es la más fácil de utilizar y es más conveniente cuando se requiere que todo un proceso se escriba en lenguaje ensamblador (tal vez, por la necesidad de una mayor velocidad), y es la más utilizada en el código fuente de esta tesis. La última se considera el punto intermedio entre las dos técnicas anteriores, se utiliza cuando hay que convertir direcciones de memoria (variables del tipo pointer) en valores numéricos, o se desea llamar a funciones para asignar valores a los registros.

3.3. MODO PROTEGIDO Y MODO REAL

El sistema operativo para disco desarrollado por *Microsoft* (o MS-DOS como se le conoce más comúnmente) para las primeras computadoras IBM y compatibles fue diseñado para que funcionara con el microprocesador 8088 (IBM desarrolló un poco después el PC-DOS, se desconoce por que encargó en primer lugar a Microsoft el desarrollo de un sistema operativo para su computadora) , esto implicaba que los recursos del sistema se administraban en base a los 8 bits de datos y control del procesador y una linea (*bus*) de direcciones de 20 bits, lo cual nos da una dirección máxima de un megabyte (equivalente a 2^20 bytes=1024 Kbytes=1048576 Bytes). Los diseñadores originales del MS-DOS consideraron un tope máximo de memoria 640 Kbytes y se reservaban los 360 Kbytes restantes para el área del sistema, además de utilizar el modo de direccionamiento segmentado (es decir, una dirección se forma con dos componentes, un desplazamiento y un ajuste. Por ejemplo, se especificaría la dirección $0080:$0010 en lugar de la dirección $8010), que se dividia en 1 conjunto de 16 bits y un conjunto de 4 bits, lo que podia direccionar un bloque continuo de cuando mucho 64 Kbytes. Esta solución funcionó mientras las aplicaciones ocuparon poca memoria, pero cuando los programas fueron tan extensos, que sus datos ya no podian acomodarse en memoria, fue necesario asignar más memoria a la computadora, o sea, más allá del limite de un megabyte. Para esto se tuvo que crear un nuevo microprocesador que pudiera manipular más lineas de direcciones y de datos: el 80286. El microprocesador 80286 tenía una linea de datos de 16 bits y una linea de direcciones de 24 bits (capaz de direccionar 16 Mbytes). Sin embargo, el limite que imponia el sistema operativo se quedó con bloques continuos de memoria de 64 Kbytes y un tope máximo de 640 Kbytes de memoria convencional. Ahora se le podía agregar más memoria a la computadora, pero el DOS no podia utilizarla por si mismo, necesitaba de un programa controlador de *memoria extendida* (memoria que está más allá del limite de 1 MegaByte). Otros sistemas operativos utilizaban la memoria extendida directamente tal como OS/2 y UNIX pero requerian de muchos recursos. El problema se atenuó cuando *Intel Corp.* lanzó al mercado la linea de procesadores 80386 y 80486. Estos procesadores de 32 lineas de dirección y de datos podian programarse para que funcionaran en *modo virtual* (simular varios procesadores 8086 trabajando al mismo tiempo) y en *modo protegido* (utilizar un método de manejo de memoria más eficiente que les permite direccionar directamente toda la memoria sin necesidad de programas especiales).

La tendencia actual de programación de juegos para computadora implica explotar todas las ventajas de los procesadores 80386 y 80486. Una de las características que más se está usando, es la

capacidad del procesador 80386 de utilizar todos los recursos del sistema sin las limitantes de tamaño del bloque de memoria y tomando el control del microprocesador para un desempeño más rápido.

Borland Pascal 7.0 puede compilar los programas en modo protegido por medio del compilador de línea BPC.EXE.

3.4. PROGRAMACION ESTRUCTURADA VERSUS PROGRAMACION ORIENTADA A OBJETOS

Los primeros lenguajes de programación eran lineales, es decir, ejecutaban todas las instrucciones desde la primera a la última, incluyendo , tal vez, algunos ciclos intermedios. Esto implicaba que si se utilizaba un mismo proceso (por ejemplo, dibujar un menú, un logotipo, o hacer una operación matemática compleja) en diferentes partes se tendrían que repetir todas las líneas de código las veces que fuera necesario.

Las variables podían ser declaradas en cualquier parte del programa tan solo con asignarles algún valor, por ejemplo: *palabra* = "hola" se asumía como cadena de caracteres; *distancia* = 10 se tomaba como un número, etc. Esto tenía un inconveniente a la hora de desarrollar programas muy largos, si se cometía algún error al escribir el nombre de una variable que ya se hubiera declarado anteriormente, podían obtenerse resultados incorrectos del programa. Por ejemplo supongamos que se le asigne el valor de 10 a la variable velocidad (*velocidad*=10) y en una parte más avanzada del programa se escriba (*aceleracion = velocidad/tiempo*), se observa que el programador omitió la última *d* de la variable *velocidad*, lo que provocaría que el programa se detuviera, e indicara que a la variable *velocida* no se le ha asignado ningún valor, o en el peor de los casos asigne un valor de cero a *aceleracion* y no reportará ningún error, en cuyo caso el programador tendría que rastrear en todas las partes del programa que tuvieran que ver con *velocidad*, *tiempo* y *aceleracion* para encontrar la fuente del error.

3.4.1. Programación estructurada

Pascal fue de los primeros lenguajes que nacieron con una nueva filosofía para el desarrollo de programas, *la programación estructurada*. La programación estructurada permite una forma de programar más elegante y ordenada. Entre otras cosas permite el manejo de procedimientos y funciones definidas por el usuario, las variables deben declararse al principio del programa, del procedimiento o de la función, y existen variables públicas y locales. Se recomienda nunca utilizar saltos incondicionales (aunque turbo pascal lo permite).

A partir de la versión 5.5 de Turbo Pascal y siguiendo el ejemplo de lenguajes como SmartTalk y Modula 2, se permite una nueva técnica de programación: *la programación orientada a objetos* u *OOP* por sus siglas en inglés.

3.4.2. Programación Orientada a Objetos (OOP)

A diferencia de la programación estructurada que se basa en módulos (procedimientos y funciones) y estructuras de datos la programación orientada a objetos encuentra primero las características comunes entre las entidades involucradas para poder crear estructuras de datos (objetos) que contengan estas características comunes.

La ecuación fundamental de la programación estructurada es:

algoritmos + estructuras de datos = programas.

Asimismo, para la OOP es :

código + datos = objeto.

Existen tres elementos básicos en la OOP :

Objetos; Es una descripción en forma de modelo de datos de un objeto perteneciente al mundo real, en la cual se toman en cuenta sus características y funcionalidad.

Clases; Los objetos pueden ser agrupados en clases y subclases de modo jerárquico. Son el equivalente a los tipos en pascal.

Métodos; Son las acciones que pueden realizarse con los objetos. Equivalen a los procedimientos y las funciones en pascal.

Las principales propiedades de las programación orientada a objetos son:

Herencia; Implica poder crear objetos nuevos tomando como base las características de algún objeto ya existente,

Paso de mensajes; Es posible enviar parámetros a los objetos para interactuar con ellos.

Encapsulamiento; Los datos de cada objeto sólo pueden ser modificados por medio de sus métodos.

Polimorfismo; Se relacionan con los métodos virtuales. Permiten el procesamiento de objetos cuyo tipo no se conoce en tiempo de compilación, son parecidos a los datos de tipo puntero.

En el presente trabajo se optó por la programación orientada a objetos, ya que sus características son las más adecuadas para representar un juego de aventuras, donde muchos de sus elementos pueden representarse por medio de objetos. Además, también es útil debido a que se puede dividir el trabajo de una forma más adecuada.

3.5. ACCESO DIRECTO DE LECTURA/ESCRITURA

El sistema operativo proporciona métodos de acceso de lectura/escritura por medio de las interrupciones, ya sea con llamadas al DOS o al BIOS[1] . Por ejemplo, para leer una tecla se puede usar

[1] La diferencia radica en que las llamadas al DOS utilizan únicamente la INT 21H (la cual está subordinada a las mejoras y cambios que se dan con cada nueva versión del sistema operativo) y las llamadas a la BIOS (que dependen del tipo de programación [firmware] con que cuenta la computadora al momento de su compra y que ya no cambia) utilizan diferentes interrupciones.

la función 00H de la INT 16 o la función 08H de la INT 21; para escribir un caracter en la pantalla se puede utilizar la función 09H de la INT 10H o la función 02H de la INT 21H; etc.

Los métodos anteriores proporcionan una forma fácil, conveniente y sobre todo portable (es decir, puede funcionar con diferentes configuraciones de hardware), pero tienen un inconveniente, son muy lentos. En el caso de lectura del teclado generalmente no importa la velocidad, sin embargo, para el caso de escritura de caracteres por pantalla y dibujo de gráficos la diferencia de velocidades está en proporción de 1 a 10 o mayor. Esta velocidad es crítica cuando se trata de programas gráficos que requieren que sean redibujadas continuamente las animaciones sobre las pantallas de fondo y en las pantallas de modo texto para que se disminuya al mínimo el parpadeo causado por la redefinición de los datos en la pantalla.

Para poder tener un acceso directo a la memoria convencional y a los puertos de entrada y salida se utilizan funciones especiales proporcionadas por el Turbo Pascal o por medio de directivas en lenguaje Ensamblador. Las dos más usadas son *Port* y *Mem*.

Con *Port* se puede accesar a un puerto físico para lectura/escritura. La sintaxis es la siguiente: **Port**[*Puerto*]; donde *puerto* es la dirección del puerto físico (por ejemplo, 60H es la dirección del puerto del teclado). Si se desea leer de ese puerto se toma el valor de la función, y si se quiere escribir se le asigna un valor a la función. Por ejemplo:

```
Teclado := Port[$60];    { Lectura del puerto del teclado }
Port[$60] := $27;        { Escritura del puerto del teclado }
```

Los equivalentes en ensamblador de *Port* son *IN* para lectura y *OUT* para escritura. Por ejemplo:

```
MOV DX, $60
IN AL, DX          ; Lectura del puerto del teclado
MOV DX, $60
MOV AL, $27
OUT DX, AL         ; Escritura del puerto del teclado
```

Con *Mem* se escribe/lee directamente en/de la memoria convencional de la computadora. La sintaxis es: **Mem**[Segmento:Ajuste]. *Mem* escribe/lee un byte de la memoria convencional en la dirección especificada por Segmento:Ajuste, otras funciones son *MemW* (lee/escribe una palabra, o sea, 2 bytes), y *MemL* (lee/escribe un entero largo, 4 bytes). Como sucede con *Port* si se requiere una lectura, *Mem* es asignado como constante y si se quiere escribir, se le asigna un valor como si fuera una variable. Por ejemplo:

```
1aLetra := Mem[$B800:0];    { Lee el código ASCII de la primera letra que      }
                            { aparezca en una pantalla de color.               }
Mem[$B800:1] := $17;        { Cambia el color del 1er caracter de la pantalla por}
                            { gris sobre azul oscuro.                          }
```

Mem tiene varios equivalentes en ensamblador, el más simple es la asignación directa. Por ejemplo:

```
MOV AX, $B800
MOV DS, AX
MOV BX, 0
MOV AX, [BX]            ; Lee el código ASCII de la primera letra que
MOV 1aLetra, AX         ; aparezca en una pantalla de color.
MOV AX, $B800
MOV DS, AX
MOV BX, 1               ; Cambia el color del 1er caracter de la pantalla por
MOV byte ptr [BX], $17  ; gris sobre azul marino.
```

3.5.1. Lectura de archivos binarios

Cuando se trabaja con archivos que no tienen un formato constante es necesario leer bloques discontinuos y/o aleatorios. Los más conveniente es utilizar *archivos sin formato*. Un archivo sin formato se declara de la siguiente manera después de la palabra reservada *VAR* ; Archivo : File. El siguiente paso es asignarle un nombre y abrirlo para lectura y/o escritura especificando el bloque de transferencia de un byte de longitud. Posteriormente si se desea se puede cambiar la posición con el comando *Seek* y se hacen las transferencias de la longitud que se necesiten . Por Ejemplo:

```
Var
  Arch : File;                  { archivo sin tipo }  .
  Almacen : Array[1..10] Of  Byte;      { Almacen temporal de los datos }
Begin
  Assign(Arch, 'Nombre.BIN');           { Se asigna un nombre de archivo }
  Reset(Arch, 1);                       { El tamaño del bloque de transferencia tiene }
                                { un byte de longitud.          }
  Seek(Arch, 32);                       { Salta a la posición 32 del archivo.   }
  BlockRead(Arch, almacen, 10); { Lee 10 bloques=10 bytes en Almacen. }
  .
  .                             { Se procesa la información. }
  .
```

```
Close(Arch);                    { Se cierra el archivo. }
End;
```

¿Qué tipo de archivos deberían manejarse como binarios?. Los archivos de imágenes (*.GIF, *.PCX, *.BMP, etc.), Los archivos de sonidos, los archivos de música y en general todos aquellos archivos que tuvieran varios elementos de diferente tamaño.

¿Qué utilidad tendría manejar archivos binarios en lugar de archivos convencionales con registros de tamaño fijo? Se podrían concentrar en un solo archivo todos los datos relativos a un mismo tema, objeto o programa y para evitar el problema de la búsqueda secuencial (que se da en este tipo archivos) se crearía una tabla índice con las posiciones de todos los objetos o un campo índice que nos dijera cual es la posición del siguiente objeto.

A pesar de todas las ventajas los archivos binarios deben tratarse con cuidado para no perder el rastro de un campo dado, o el significado de un cierto byte dentro de todo el archivo. Además, es preferible saber de antemano que bloques pueden leerse de forma continua y cuales no, o tal vez cargar todo el archivo en memoria y diseñar una función que desglose los datos que necesitamos. Como siempre la mejor solución está en función de las necesidades y preferencias del programador. Pongamos el ejemplo de la lectura de un archivo BMP. ¿Es más conveniente leerlo de forma inversa utilizando el comando *Seek*? ¿o sería mejor cargar todo el archivo en memoria e invertirlo ahí?. El factor que generalmente determina lo más adecuado es la memoria disponible. Si hay suficiente memoria es mucho más rápido hacer el proceso de inversión en memoria convencional que leerlo del disco byte por byte. Si se dispone de memoria extendida suficiente se puede utilizar ésta para ahorrar memoria convencional. Si el archivo es demasiado extenso se puede leer en bloques más pequeños, invertirlos en memoria e irlos colocando directamente en la memoria de vídeo.

3.5.2. Direcciones de la memoria de vídeo

Una cosa es saber *cómo* escribir en la memoria convencional y otra muy distinta es saber *dónde* se puede escribir. Los datos de la memoria de vídeo (lo que se ve en la pantalla) se almacena en una sección especial de la memoria convencional conocida como *área de vídeo de la BIOS*. Es aquí donde se guardan las letras y los colores en modo texto y las pantallas gráficas en modo gráfico. La razón de que se necesite un área de memoria especial para el vídeo es debido a la naturaleza del monitor. Las imágenes que se visualizan en un monitor, son formadas por un haz de luz que recorre toda la pantalla. Este haz es "registrado" por la pantalla fluorescente que reacciona a la intensidad del cañón de electrones lo que da como resultado una imagen que dura fija una fracción de segundo, suficiente tipo para que el haz de luz recorra la pantalla otra vez, dando la ilusión de una imagen continua. Si sólo se "dibujara" una sola vez la pantalla, la imagen se perdería casi instantáneamente. Por esta razón, los datos necesarios para redibujar la pantalla continuamente se almacenan en la memoria convencional.

Los rangos válidos de la memoria de video van de la dirección $A000:0000 a la $BFFF:FFFF, espacio suficiente para almacenar los datos de una pantalla gráfica de 640x480 pixeles y 16 colores, algo así como 150 KBytes. Este rango de direcciones se distribuye de la siguiente manera:

Modo de video	Memoria requerida	Rango de direcciones
Texto monocromático	4 Kbytes	$B000:0000-$B000:0FFF
Texto 16 colores	4 KBytes	$B800:0000-$B800:0FFF
Gráficos CGA	16 KBytes	$B800:0000-$B800:3FFF
Gráficos EGA, VGA	hasta 150 KBytes	$A000:0000-$BFFF:FFFF

La forma en que están organizados los datos en modo gráfico se discute en el capítulo V. **Entorno gráfico.**

3.6. INTERRUPCIONES

Una interrupción es una llamada de atención al procesador para indicarle que ha sucedido un cierto acontecimiento en el sistema. El procesador deja todo lo que está haciendo y salta a una sección de código que se encarga de atender la causa que originó la interrupción, y al terminar el procesador restaura las condiciones del sistema a como estaban inicialmente y sigue todo como si no hubiera pasado nada. El sistema operativo trabaja a base de interrupciones para administrar todos sus procesos, por ejemplo, si una tecla es oprimida se activa la interrupción de teclado (interrupción 09H), el sistema operativo lee la tecla y la almacena en una sección de memoria especial conocida como almacén del teclado (keyboard buffer) donde permanece hasta que sea leída por otra interrupción (por ejemplo la 16H). La razón por la que el sistema operativo utiliza interrupciones, es para no perder tiempo rastreando todos los dispositivos periféricos a ver si se movió el ratón, se pulsó una tecla, etc., proceso que haría intolerablemente lentos todos los procesos

Existen dos tipos de interrupciones: de hardware y de software. Las *interrupciones de hardware* se activan cuando un evento físico las provoca, como puede ser el pulsar una tecla. Por tanto, la interrupción 09H es una interrupción de hardware. *Las interrupciones de software*, en cambio, son activadas por el programa como si fueran llamadas a subrutinas. La interrupción 16H es un ejemplo de interrupción de software.

En un lenguaje de alto nivel es el compilador el que decide cuales interrupciones de software utilizar y cuando. En ocasiones, tal vez sería más conveniente que nosotros definiéramos qué tipo de interrupciones utilizar y cuándo, esto es particularmente útil cuando queremos adecuar el programa a nuestras necesidades, además existen interrupciones especializadas que no son soportadas por los lenguajes de alto nivel. En estos casos se utiliza el lenguaje de bajo nivel que utiliza directamente las interrupciones.

Las interrupciones más utilizadas son:

- *INT 9H*: (Hardware). Se activa cuando se pulsa cualquier tecla. Guarda el contenido en el almacén del teclado.
- *INT 10H*: (Software). Es la interrupción que maneja el vídeo y los modos gráficos. Se utiliza para elegir uno de los diferentes modos de vídeo, incluyendo los modos gráficos; para cambiar las paletas; para cambiar las páginas de vídeo; para escribir texto por medio de las rutinas de la BIOS (proceso que es bastante lento), y para cambiar el tipo de letra tanto en modo texto como en modo gráfico (solo tarjetas VGA o SuperVGA).
- *INT 16H*: (Software). Es la interrupción que controla los procesos de teclado tales como leer una tecla con eco y sin eco, puede reconocer cuando se trata de una tecla alfanumérica, del cursor, de función o de intercambio (Control, Alt, Shift, etc.).
- *INT 1CH*: (Software). Se activa con cada tic del reloj, aproximadamente 1,800 veces por segundo. No es activada físicamente, la interrupción 8H (Hardware) es la encargada de llamarla periódicamente.
- *INT 21H*: (Software). Es la interrupción del sistema operativo. Se encarga de la mayoría de los procesos de administración del sistema tales como: manipular archivos, leer y salvar datos en un archivo, cambiar atributos, manipular directorios, borrar archivos, administrar la memoria, etc.
- *INT 33H*: (Software). Maneja el ratón, lee y cambia su posición, los hace visible e invisible, cambia su forma, define el Punto caliente (Hot Spot), o sea, el lugar que se usa de referencia para sus coordenadas, etc.

3.6.1. Vectores de interrupción

Muchas de las rutinas que atienden a las interrupciones, ya se encuentran en la computadora en un circuito conocido como ROM BIOS, y las demás se cargan cuando se inicializa el sistema operativo. Las direcciones de cada rutina se encuentran en la zona de memoria comprendida entre las direcciones $0000:$0000 y $0000:$03FF. Cada dirección consta de 4 bytes que conforman una dirección completa y que al ser invertidos se encuentra que el segmento son los dos primeros bytes y el ajuste los dos siguientes. Cuando se genera una interrupción, el sistema operativo busca en la dirección de memoria $0000:Número_de_interrupción_x_4 (debido a que cada dirección consta de 4 bytes) y hace una llamada lejana a la rutina que se encuentre en esa dirección. Te preguntarás ¿por qué tomarse esa molestia?, ¿por qué no ir directamente a la dirección?. La razón de usar vectores de interrupción es la posibilidad de sustituir o agregar procesos a una interrupción. Como se comentó anteriormente algunas rutinas están escritas en la ROM BIOS acrónimo de (Read Only Memory - Basic Input Output System) y no pueden borrarse o modificarse, en ese caso, ¿qué pasaría si se tuviera una rutina que manejara el video (INT 10H) mejor que la proporcionada por el fabricante?, pues simplemente no se podría utilizar. Esta es la razón que los diseñadores originales del sistema operativo proporcionaron esta facilidad para las interrupciones.

Turbo Pascal incluye dos procedimientos para manejar la tabla de vectores de interrupción, éstos son : *GetIntVec*(N_Int:Byte, Var Direccion:Pointer) y *SetIntVec*(N_Int:Byte, Var Direccion:Pointer);

donde 'N_Int' es el número de interrupción que se desea usar y 'Direccion' es una variable de tipo apuntador que para *GetIntVec* sirve para almacenar la dirección de la rutinas de 'N_Int' y para *SetIntVec* se usa para obtener la dirección que sustituirá a 'N_Int' en la tabla de vectores de interrupción.

3.6.2. Programas residentes

El DOS no es multitarea, o sea, no puede realizar varias operaciones al mismo tiempo (digamos imprimir y seguir capturando o ejecutar dos programas simultáneamente), ésto es debido a que el primer procesador (el 8088) no disponia de capacidad suficiente para poder atender a dos tareas al mismo tiempo. Cuando se diseñaron procesadores más potentes y rápidos (80286,80386,etc), que simulaban multitarea con otros sistemas operativos tales como OS/2 el DOS se vio superado por el microprocesador[2]. Al mismo tiempo los programadores más avanzados modificaban las tablas de vectores de interrupción para incluir sus propias rutinas cada vez que se activará alguna interrupción y después devolver el control como si no hubiera pasado nada. Se conocían como *programas residentes*. Un programa residente típico permanece "dormido" hasta que se activa la interrupción que está utilizando y se cumple una condición evaluada por el programa, entonces se ejecuta los procedimientos requeridos. Tipicamente la forma de activación es por medio de una tecla o combinación de teclas. Un programa residente consta de dos partes, la parte de inicialización que prepara al sistema para alojar su código y que da valores a las variables globales, la cual sólo se ejecuta una vez y no se almacena en memoria. La segunda parte es el código en si, el cual es cargado en memoria en espera de ser activado. Desarrollar y explicar un programa residente completo no está dentro de las expectativas de esta tesis, sin embargo, si se utiliza una versión modificada de esta técnica para el control de procesos tales como poner música de fondo, se conoce como *rutinas residentes*.

3.6.3. Rutinas residentes

Hay rutinas dentro de un juego de computadora que deben ser llamadas constantemente y de forma periódica, o sea, a intervalos de tiempo fijo. Estas rutinas pueden controlar un reloj que siempre esté presente, checar, activar y desactivar banderas, controlar la música de fondo, etc. Si incluimos estas rutinas en un ciclo que involucre funciones o procedimientos donde se hagan cálculos o se pidan datos, entonces es muy difícil sincronizar los tiempos para que haya una constancia en las partes críticas. Por ejemplo, si se está tocando una música de fondo y se tarda el programa en algún procedimiento tal como dibujar la pantalla, el resultado es una melodía desincronizada e ininteligible. En estos casos es mejor declarar estas rutinas como residentes

El proceso para crear una rutina como residente es:

---

[2] Se supone que la última versión de DOS iba a ser la 4.0 y entonces IBM y Microsoft promocionarían OS/2 como el sistema operativo estándar, sin embargo, con el desarrollo de Windows, Microsoft decidió impulsar este en lugar de OS/2 y seguir con el mercado de DOS para los demás usuarios. Actualmente DOS está en su versión 6.2, Windows en su versión 3.1 y superan (por lo menos en México) con mucho en popularidad a OS/2 que ya está en su versión 2.1 y en el cual IBM todavía tiene mucha confianza, pero eso ya es otra historia.

1. **Elegir una interrupción adecuada.** Una interrupción que se activa constantemente y que representa menos problemas es la interrupción 1CH (del tic del reloj), que se activa más o menos 1,800 veces por segundo.

2. **Guardar la dirección de la interrupción que se va a sustituir.** Se puede guardar en una variable del tipo procedimiento, que debe ser global (Declarada antes de los procedimientos y funciones). Por ejemplo:

```
Var
  Direccion_1C_Proc : Procedure;
  Begin
    GetIntVec($1C, @Direccion_1C_Proc);
                :
                :
```

3. **Declarar el procedimiento como *interrupt* y lejano.** Esto se hace poniendo la directiva de compilación {$F+} antes del nombre del procedimiento o función y la palabra *interrupt* después de la declaración de la función. Debe recordarse llamar a la interrupción anterior al principio del procedimiento. Por ejemplo:

```
{$F+}
PROCEDURE Rutina_Residente; interrupt;
Begin
  Direccion_1C_Proc;
      :
      :
End;
```

4. **Sustituir la dirección de la interrupción por la de tu procedimiento.** Se utiliza el comando *SetIntVec*. Al sustituir la dirección de la interrupción tu procedimiento se activará cada vez que se active o llame a la interrupción 1C. Por ejemplo:

```
SetIntVec($1C, Addr(Rutina_Residente));
```

5. **Al final del programa restaurar la dirección de la interrupción, nuevamente con el comando *SetIntVec*.** Esto es porque al salir del programa, los datos del procedimiento se pierden, pero la tabla de vectores de interrupción sigue apuntando a la dirección donde antes

estaba tu rutina, por lo que al modificarse esta parte de memoria (al cargarse otro programa por ejemplo), pueden suceder cosas impredecibles e incluso atascarse el sistema. Por ejemplo:

```
                :
    SetIntVec($1C, Addr(Direccion_1C_Proc));
End.  { Fin del programa }
```

CAPITULO 4

ORGANIZACION DE DATOS

CAPITULO 4
ORGANIZACION DE DATOS

En este capitulo se hablara de la forma como están relacionadas las diferentes estructuras de datos utilizadas en los programas elaborados. Desde la implementación de rutinas para el manejo del ratón, hasta los programas importantes que surgen de éstos. Desde la organización de directorios hasta la explicación de cada archivo utilizado.

4.1. DESCRIPCION DE DIRECTORIOS Y ARCHIVOS.

4.1.1. Organización de directorios.

La cantidad de archivos utilizados para la elaboración de la tesis es enorme y son de todo tipo. Con el proposito de tener un perfecto control de estos archivos, generamos una estructura de directorios que debe existir para el buen funcionamiento de los tres programas principales.
Primero a partir del directorio raiz, se genera un subdirectorio que se lamara TESIS. y a dentro de este los siguientes directorios.

TESIS == EDITOR
LIBRETO
EJECUTOR
DND
ANM
LIB
SB
PCX
BIN
GIF
AVI
FONT

4.1.2. Descripción de directorios.

Directorio	Contenido
- TESIS	Programas del entorno de Turbo Pascal 7.0.
- EDITOR	Editor de duendes y animación.
- LIBRETO	Libreto, programa generador de escenas.

- EJECUTOR	Programa ejecutor de escenas.
- DND	Archivos de duendes.
- ANM	Archivos de animaciones de duendes.
- LIB	Unidades complementarias de pascal (TPU).
- SB	Archivos de música y sonido.
- PCX	Archivos gráficos en formato .PCX.
- BIN	Archivos gráficos en formato binario (.BIN).
- GIF	Archivos gráficos en formato .GIF.
- AVI	Archivos de secuencias gráficas en formato .AVI.
- FONT	Archivos con diferentes tipos de letras.

4.1.3. Descripción de archivos.

La manera actual de organizar archivos es sin duda alguna por su extensión, y a partir de ésta se hace la siguiente explicación

Extensión	Función.
*.DND	Archivos de duendes. Contienen las imágenes de todos los ángulos y vistas de un duende. Tienen un formato propio no comprimido. Se encuentran en el subdirectorio DND.
*.ANM	Archivos de animación de duendes. Contienen la información necesaria para darle animación por completo a un duende. Se encuentran en el directorio ANM.
*.LIB	Archivos de libretos, contienen la información necesaria para armar una escena completa que será interpretada por el programa ejecutor. Se localizan en el subdirectorio EJECUTOR.
*.PAL	Archivos de paletas de colores. Contienen la paleta para el modo MCGA de 256 colores. Se encuentran en el subdirectorio EJECUTOR.
*.FNT	Archivos de tipos de letra que sustituyen a los del BIOS. Se encuentran en el subdirectorio FONT.
*.CMF	Archivos de música en frecuencia modulada, son utilizados para los fondos musicales en cada escena. Se encuentran en el subdirectorio SB.
*.PCX	Archivos en formato gráfico comprimido PCX. Se encuentran en el directorio PCX.
*.GIF	Archivos en formato gráfico comprimido GIF. Se encuentran en el directorio GIF.
*.BIN	Archivos en formato gráfico no comprimido. Se encuentran en el directorio BIN.

*.AVI	Archivo de secuencias gráficas en formato AVI. Se encuentran en el subdirectorio AVI..

Archivos especiales.

FRASES.FRA	Contiene todos los mensajes escritos que aparecen durante el juego.
SBFMDRV.COM	Driver de música de fondo para la tarjeta Sound Blaster.
CT-VOICE.DRV	Driver de sonido para la tarjeta Sound Blaster.

LIBUTI.PAS	Libreria de rutinas básicas.
LIBPAL.PAS	Libreria de rutinas de manejo de paletas.
LIBGRA.PAS	Libreria de rutinas básicas de modo gráfico.
LIBPAN.PAS	Libreria de rutinas de manejo de formatos gráficos PCX, GIF, BIN y AVI.
LIBMUS.PAS	Libreria de rutinas de manejo de música de fondo.
LIBSBP.PAS	Libreria de rutinas para el control de la Sound Blaster.
LIBDND.PAS	Libreria de rutinas para el manejo completo de duendes.
LIBANI.PAS	Libreria de rutinas para el control de animaciones simultáneas.
LIBRAT.PAS	Libreria de rutinas para el control del dispositivo RATÓN.
LIBTXT.PAS	Libreria de rutinas para el control de mensajes durante el juego.
LIBDIR.PAS	Libreria de rutinas para el manejo de directorios.
LIBXMS.PAS	Libreria de rutinas para manejar memoria extendida.

4.1.4. Descripción de programas.

Nuestro trabajo está dividido en tres programas principales. Cada uno se explicará en su correspondiente capítulo. Pero ahora se definirán los archivos necesarios para cada uno, y que se deben encontrar en el mismo directorio.

Editor de duendes y animación

EDITOR.PAS	Fuente en pascal del programa.
DEFECTO.PAL	Paleta por default.
EDITOR.PCX	Entorno gráfico del programa.
8X8.FNF	Font de matriz de 8 x 8.
6X8.FNT	Font de matiz de 6 x 8.
EDITOR.DND	Archivo de botones del editor.

Libreto.

LIBRETO.PAS	Fuente en pascal del programa.

LIBRETO.DND	Archivo de botones de libreto.
LIBRETO.PCX	Entorno gráfico del programa.
8X8.FNF	Font de matriz de 8 x 8.
6X8.FNT	Font de matiz de 6 x 8.

Programa Ejecutor.

EJECUTOR.PAS	Fuente en pascal del programa.
FRASES.FRA	Archivo de mensajes.
Archivos.LIB	Archivos de librerias.

4.1.5. Requerimientos de librerias.

El archivo TURBO.TPL del pascal contiene los procedimientos pertenecientes a las unidades DOS y CRT que son las únicas que se utilizan de las que proporciona Turbo Pascal 7.0. El resto de las unidades utilizadas fueron programadas por nosotros. Ahora de cada programa utilizado se numeran los archivos utilizados por los mismos.

- LIBXMS.PAS	Ninguno.
- LIBUTI.PAS	Crt.
- LIBTXT.PAS	Ninguno.
- LIBSBP.PAS	Crt, Dos, LibXMS.
- LIBMUS.PAS	Ninguno.
- LIBRAT.PAS	Ninguno.
- LIBGRA.PAS	Dos, Crt, LibRat, LibUti, LibXMS.
- LIBPAN.PAS	Dos, Crt, LibGra.
- LIBDIR.PAS	Crt, Dos, LibGra, LibUti, LibRat, LibXMS.
- LIBDND.PAS	Dos, Crt, LibRat, LibUti.
- LIBPAL.PAS	Crt, Dos, LibGra, LibDnd.
- LIBANI.PAS	Crt, LibDnd, LibUti, LibGra, LibDir, LibXMS, LibRat, LibTxt, LibSB
- EDITOR.PAS	Crt, LibSB, LibDnd, LibRat, LibGra, LibUti, LibPan, LibXMS, LibDir, LibAni, LibPal.
- LIBRETO.PAS	Crt, Dos, LibUti, LibPan, LibPal, LibXMS, LibGra, LibRat, LibDnd, LibDir, LibAni.
- EJECUTOR.PAS	Crt, Dos, LibSB, LibAni, LibXMS, LibGra, LibPan, LibDnd, LibRat, LibUti, LibDir, LibTxt.

4.2. ORGANIZACION DE PROCEDIMIENTOS.

Dada la naturaleza de los datos requeridos para manejar animación, se implemento la Programación Orientada a Objetos (OOP). De manera que para dar animación a un duende, se heredan los datos de éste a las rutinas de animación.

Existen procedimientos generales como crear cuadros, mensajes, activar modos gráficos, leer datos, etc. que no requieren de las propiedades de la POO, por lo tanto se implementaron como procedimientos normales. Solo se utilizan los objetos en rutinas relativas a animación y manejo de duendes. Estos procedimientos se encuentran en las siguientes librerías :

LIBGRA.PAS
LIBMUS.PAS
LIBXMS.PAS
LIBTXT.PAS
LIBUTI.PAS
LIBSBP.PAS

Por otra parte en las otras librerías si se implemento la POO. La lista siguiente muestra los objetos que se encuentran en cada librería.

Libreria	Objeto
LIBRAT.PAS	RATON
LIBPAN.PAS	FORMATO
LIBDIR.PAS	DIR_MCGA
LIBDND.PAS	DUENDES
LIBPAL.PAS	PALETAS
LIBANI.PAS	ANIMAR
EDITOR.PAS	PRINCIPAL
EJECUTOR.PAS	PRINCIPAL

Acomodándolos de manera que se pueda apreciar la herencia tenemos.

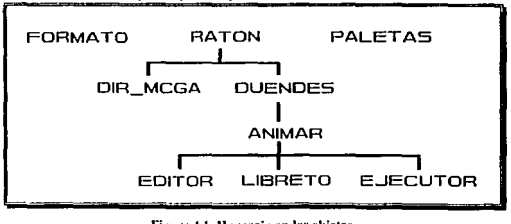


Figura 4.1. Herencia en los objetos.

4.3. DESCRIPCION DE LA ESTRUCTURA DE ARCHIVOS.

4.3.1. Estructura de Duendes.

Los duendes (sprites) no son mas que un grupo de imágenes rectangulares almacenadas en un mismo archivo. La forma en que se manejan estas imágenes es por medio de un arreglo de pointers. En el capitulo 3, se explicó como manejar estos apuntadores.

Cada una de las imágenes es un rectángulo que mide X pixeles de largo por Y pixeles de ancho. En el mismo apuntador se guardan en los cuatro primeros bytes, los valores de largo y ancho de esta imagen.

Si representa esta información en un apuntador (arreglo de bytes), se tiene lo siguiente :

Largo parte baja
Largo parte alta
Ancho parte baja
Ancho parte alta
Punto 1
Punto 2
Punto 3
Punto 4
.
.
.
.
.
Punto (largo x ancho)

Tabla 4.1.

Este apuntador es interpretado de la siguiente manera.

1	2	3	4			largo -1	Largo
largo + 1	largo + 2	largo + 3	largo + 4			2(largo)-1	2 (largo)
2(largo)+1	2(largo)+2	2(largo)+3	2(largo)+4			3(largo)-1	3 (largo)
						(ancho) (largo) - 1	(ancho) (largo)

Tabla 4.2.

Que visto en la pantalla, podria ser algo asi :

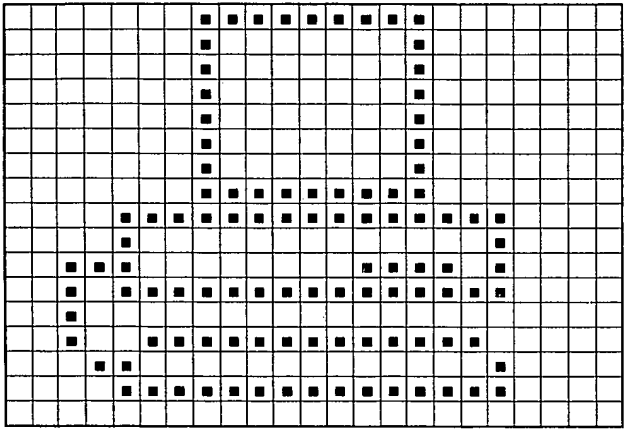


Tabla 4.3.

Por el momento bastara con visualizar una imagen de esta manera, "una matriz de pixeles o bien un arreglo rectangular de pixeles".

Cuando se manejan POINTER's en Pascal, se debe indicar el tamaño con el que se abrirá dicho apuntador. Para hacerlo de una manera mas rápida, se maneja una variable en la que se almacena el tamaño de este bloque, de manera que cuando se quiera hacer referencia a apuntador, no se tenga que estar calculando su tamaño.

Por ultimo se manejada una variable que es el contador del numero de estas imágenes dentro del duende.

La estructura de los duendes en pascal queda entonces declarada de la siguiente manera :

```
Const
   MaxImagen = 80;                    {Número máximo de imágenes en el duende    }
Duende = record
      Dnd  : array[1..MaximagenJ of pointer;   {Arreglo de imágenes               }
      Tam  : array[1..MazimagenJ of word;      {Tamaño de cada imagen según su indice }
```

```
    Total : byte;                    {Número total de imágenes          }
end;
```

La constante Maximagen, indica el número máximo de imágenes que puede estar almacenada en un duende. En este caso es de 80, esto depende del número de imágenes que se quieran manejar por duende.

Finalmente declaramos una variable de tipo duende, como por ejemplo :

```
Var
    Fulano : Duende.
```

Esta variable se utilizará en todos los procedimientos que pidan variables tipo duende que se encuentran en la libreria LIBDND.PAS (Ver anexo correspondiente).

La estructura de los duendes en disco, es decir de los archivos *.DND, está basada únicamente en el contenido de los pointers, de su tamaño y del número total de imágenes en el arreglo (almacenado en la variable total del registro duende). Dicha estructura es la siguiente :

Tamaño 1, 2 bytes	Bloque 1, bytes según tamaño 1
Tamaño 2, 2 bytes	Bloque 2, bytes según tamaño 2
:	:
:	:
Tamaño Total, 2 bytes	Bloque Total, bytes según tamaño total

Tabla 4.4.

Este archivo es generado por el Editor de Duendes (ver capitulo 7), y todos los procedimientos relativos al manejo de duendes se encuentran en la libreria LIBDND.PAS.

4.3.2. Estructura de animaciones.

Las animaciones, no son mas que una lista de acciones definidas bajo etiquetas (Ver capitulo 7). Estas acciones son guardadas en un pointer y el tamaño de sus registros es siempre el mismo.
Como se explicará mas adelante (capitulo 7), las animaciones tienen registros con la siguiente estructura:

```
Anima = record
  Estado   : byte;      {1 byte }   {Acción a realizar                }
  Etiqueta : string[8]; {9 bytes}   {Etiqueta de la acción            }
  Imagen   : byte;      {1 bytes}   {Numero de imagen actual de duende }
  Duracion : word;      {2 bytes}   {Duración de la acción            }
  IncreX   : integer;   {2 bytes}   {Incremento en X                  }
  IncreY   : integer;   {2 bytes}   {Incremento en Y                  }
end;                    {17total}
```

Esta estructura es similar para cada registro y es almacenada en un Pointer que es asignado a cada animación.

Su estructura en disco es exactamente igual a como se encuentran en el apuntador, por lo que la estructura general de los archivos *.ANM es la siguiente :

Estado 1 1 byte	Etiqueta 1 9 bytes	Imagen 1 1 byte	Duracion 1 2 bytes	IncreX 1 2 bytes	IncreY 1 2 bytes
Estado 2 1 byte	Etiqueta 2 9 bytes	Imagen 2 1 byte	Duracion 2 2 bytes	IncreX 2 2 bytes	IncreY 2 2 bytes
:	:	:	:	:	:
:	:	:	:	:	:
Estado n 1 bytes	Etiqueta n 9 bytes	Imagen n 1 byte	Duracion n 2 bytes	IncreX n 2 bytes	IncreY n 2 bytes

Tabla 4.5.

En el caso del campo de etiquetas que mide 9 bytes, se considera solo una cadena de 8 bytes siendo en primer byte el largo de la cadena.

Estos archivos son generados por el Editor de Animación, y todas las rutinas que son utilizadas para su manejo, se encuentran en la libreria LIBANI.PAS.

4.3.3. Estructura de libretos.

Los libretos son los que definen una escena completa. No son mas que una lista de las distintas situaciones que se encontrarán en dicha escena.

Los datos da cada situación son distintos y requieren de registros de diferente tamaño y numero de campos.

El manejar archivos con una estructura que no es constante, es muy utilizado actualmente, sobre todo en juegos de video. Se da el caso de que en un mismo archivo se guarda la información completa de todos los duendes, animaciones, paletas, escenas, etc. La ventaja principal de manejar este tipo de archivos es el ahorro en disco y lo rápido que resulta cargar la información.

En el caso de los libretos solo se maneja la información necesaria para armar una escena completa. Para el manejo de cada registro, se utilizara un campo llave que indica la longitud de la información que se va a leer. A continuación se la longitud de cada registro según su numero de acción.

Acción (campo llave)	Tamaño del registro
1	10
4	11
5	25
6	3
7	12
8	11
9	17

Tabla 4.6.

Una explicación mas detallada sobre el manejo de archivos *.LIB se encuentra en el capitulo 9. Los procedimientos relativos al manejo de estos archivos se encuentran en los programas LIBRETO.PAS y EJECUTOR.PAS.

4.3.4. Estructura de paletas.

Los archivos *.PAL son utilizados para almacenar la información completa de una o varias paletas de 256 colores.

Una paleta de colores ocupa 768 bytes (ver capitulo 5). Esta se guarda tal cual en disco en 768 bytes contiguos. Si el archivo mide 768 bytes significa que solo hay una paleta, si mide 1536 bytes, significa que se encuentra la información de 2 paletas, si mide 2304 bytes, tendrá la información de 3 paletas, etc. De tal manera que los primeros 768 bytes son de la primer paleta, los segundos 768 bytes son de la segunda paleta, etc.

La estructura de los archivos *.PAL queda definida de la siguiente manera :

Paleta 1 768 bytes
Paleta 2 768 bytes
:
:
Paleta n 768 bytes

Tabla 4.7.

Los procedimientos necesarios para el manejo de las paletas de colores se encuentran en la libreria LIBPAL.PAS (ver anexo correspondiente).

4.4. ESTRUCTURA GENERAL.

A manera de resumen, para la mejor visualización de lo descrito en este capitulo, obsérvese la siguiente figura :

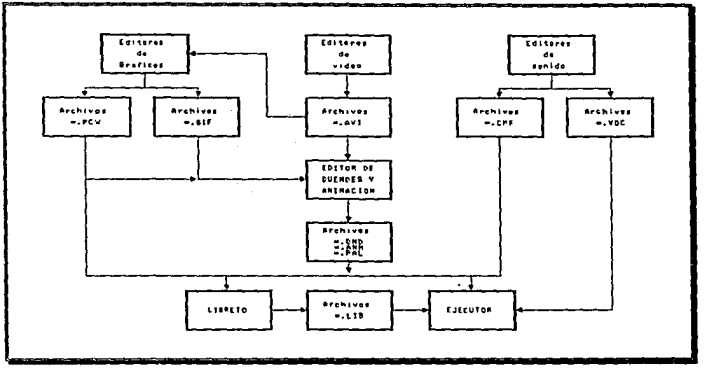


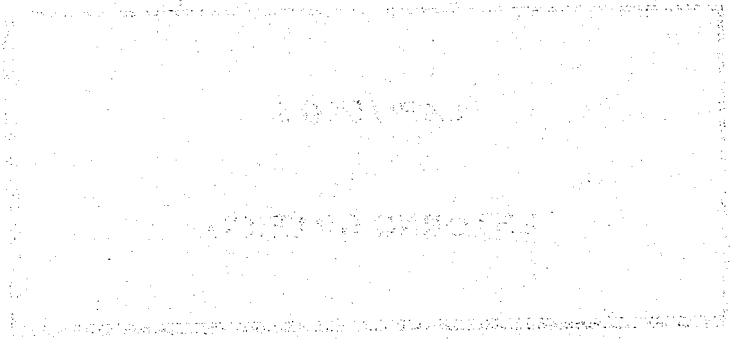
Figura 4.2. Relación entre archivos y programas.

Se puede observar que los tres programas principales dependen de un software auxiliar, pero se trato de minimizar lo mejor posible dicha dependencia. Para mayor información sobre el software de apoyo consultar el capitulo 6.

El programa ejecutor no modifica el contenido de los archivos que utiliza, solo se limita a utilizarlos. Es en el uso de este programa donde se apreciara el esfuerzo realizado para la generación de cada escena.

CAPITULO 5

ENTORNO GRAFICO



CAPITULO 5
ENTORNO GRAFICO

5.1. MODOS GRAFICOS ACTUALES.

El avance tecnológico actual también se aprecia en el aumento de la calidad gráfica de los monitores. La diferencia es enorme, desde los monitores en blanco y negro hasta los de 64 millones de colores. Desde resoluciones de 320x200 hasta resoluciones de 1024x 1280 pixeles. Estas resoluciones han mejorado a las de la televisión, y ganan en calidad. En un monitor de alta resolución, se puede ver un objeto como si estuviera detrás de una ventana.

Existen resoluciones que son estándar como la VGA, EGA, Hercules, CGA, etc. Pero actualmente no es raro encontrar un software que maneje resoluciones propias y extrañas, como decir por ejemplo, 700x400 en 256 colores.

Utilizando el sistema operativo DOS, se pueden utilizar las resoluciones estándar actuales, que son activadas por medio de la interrupción 10h.

La tabla siguiente muestra estas resoluciones.

Modo de video (Hex)	Resolución en pixeles	Número de colores	Modo texto o gráfico.
00h	40 x 25	16	Texto
01h	40 x 25	16	Texto
02h	80 x 25	16	Texto
03h	80 x 25	16	Texto
04h	320 x 200	4	Gráfico
05h	320 x 200	4	Gráfico
06h	640 x 200	2	Gráfico
07h	80 x 25	2	Texto
08h	160 x 200	16	Gráfico
09h	320 x 200	16	Gráfico
0Ah	640 x 200	4	Gráfico
0Bh	Reservado		
0Ch	Reservado		
0Dh	320 x 200	16	Gráfico

0Eh	640 x 200	16	Gráfico
0Fh	640 x 350	4	Gráfico
10h	640 x 350	16	Gráfico
11h	640 x 480	2	Gráfico
12h	640 x 480	16	Gráfico
13h	320 x 200	256	Gráfico

Tabla 5.1.

Cuando se hace referencia a modo texto y a modo gráfico se habla de la propiedad del video de manejar sólo caracteres o bien pixeles.

En el modo texto solo se podra desplegar en video cualquiera de los caracteres ASCII. Cualquier dibujo que se quisiera hacer sería con los mismos caracteres, y no se podría controlar pixel por pixel. A menos claro, que se utilizara la interrupción del generador de caracteres del DOS. Pero aun así sólo estaría limitado a los 255 caracteres (menos los que son de control).

En el modo gráfico, se puede hacer lo que uno quiera. Se puede controlar pixel por pixel y hacer cualquier tipo de figuras, desde un punto en la pantalla hasta figuras tridimensionales. Se puede desplegar desde un dibujo hasta una fotografía digitalizada. Y por supuesto también se puede escribir.

Evidentemente para hacer un juego de video de buena calidad, se debe pensar sin lugar a dudas en el modo gráfico.

5.1.1. El modo MCGA de 256 colores.

El lenguaje de programación Pascal y el lenguaje C, incorporan una unidad (librería) dedicada al manejo de gráficos. Proporcionan rutinas para activar un modo gráfico, para poner un pixel en pantalla, una línea, un círculo, un cuadrado, un rectángulo, rutinas para leer y poner bloques de una imagen en pantalla, para generar caracteres gráficos de diferente tamaño, para generar sólidos, patrones, cambiar colores, etc. Son de hecho las rutinas más que suficientes para hacer un pequeño juego de video, pero la desventaja principal es que los modos gráficos que soportan manejan cuando mucho 16 colores (ver tabla 5.1).

La ultima versión de Turbo Pascal (versión 7.0), incorpora incluso rutinas para manejar modos de super VGA para varios tipos de tarjetas gráficas, sin embargo como se mencionaba anteriormente solo soporta modos en 16 colores. El hecho principal de que se manejen pocos colores, es que entre mas colores soporte un modo gráfico, mayor es la cantidad de memoria que se necesita para almacenar una imagen.

Desafortunadamente Pascal no incorpora rutinas para comprimir imágenes, por lo que con unas cuantas de 256 colores se acabaría completamente la memoria.

Cuando empezamos a programar en el modo MCGA 320x200 en 256 colores, nos encontramos con un pequeño problema. El lenguaje Pascal con el que tanto nos familiarizamos, no era capaz de activar ese modo gráfico. La enorme libreria de manejo de gráficos no nos servia absolutamente para nada, ni para poner un punto en la pantalla, y nosotros queriamos hacer juegos de video. Tampoco podiamos utilizar las ventajas de las páginas de video con las que no cuenta esa resolución.

Afortunadamente, Pascal permite utilizar el manejo de interrupciones (ver Capitulo 3), de esta manera se puede por medio de la interrupción 10h función 00h subfuncion 13h, activar ese modo gráfico. Pero no bastaba con activar el modo grafico, habia que hacer librerias completas para sustituir por completo a la libreria GRAPH.TPU del pascal, desde poner un punto en pantalla hasta hacer rutinas de movimiento de duendes.

A partir de la versión 6.0 de Turbo Pascal, se incorpora en éste, la programación directa en ensamblador. Lo cual nos sirvió bastante para la elaboración de las rutinas mas importantes de manejo de gráficos. Al programar directamente en lenguaje ensamblador se gana principalmente en velocidad en el despliegue gráfico.

Existen dos tipos principales de modos en MCGA, el modo de 640x200 en 2 colores (que es equivalente al modo CGA de la misma resolución, ver subfuncion 06h en la tabla 5.1) y el modo de 320x200 en 256 colores (subfuncion 13h).

Este modo de video funciona perfectamente en un monitor que sea MCGA, VGA o Super VGA. Actualmente los monitores VGA son los que dominan el mercado y son más que suficientes para nuestros propósitos.

El decir 256 colores significa que se pueden manejar 256 colores al mismo tiempo y no más, pero... se puede manejar una paleta de nada menos que 256 mil colores diferentes (64 x 64 x 64). Es decir se toman de esos 256 mil colores sólo 256, los que sean, y son los se usan para poner las imágenes. Una paleta de 256 colores es perfecta para poner imágenes digitalizadas, y suficiente para poner dibujos en pantalla. Si bien la resolución 320x200 no es muy buena, se compensa con el manejo de 256 colores.

Cuando el video se encuentra en esa resolución y se quiere poner un punto en la pantalla, lo único que se debe hacer es poner el punto con el color número 5, o con el color número 20, o con el 67, etc. Previamente se eligió que el color 5 fuera el rojo, el 20 el azul, y el 67 el rosa. Pero también se pudo haber elegido que el 5 fuera el morado, el 20 el amarillo, y el 67 el negro. Se pueden manejar los colores a voluntad del programador y existe una amplia gama.

Para casos en los que se requieran procesamientos de imágenes de manera profesional, 256 colores no serán suficientes, entonces se utilizarán mejores resoluciones y mas colores. Una resolución muy buena es la de 1024x768 x 64 millones de colores. Pero su manejo va mas lejos del alcance de esta tesis.

En el resto de esta tesis y en los programas presentados nos limitaremos exclusivamente al manejo del modo MCGA 320x200 por 256 colores. Esto es porque son rutinas a las que no se tiene acceso y no

se cuenta con información. Además de que la mayoría de los juegos de video actuales que hay en el mercado, utilizan esa resolución para sus despliegues gráficos.

Utilizando ensamblador, definimos el procedimiento ModoDeVideo de la siguiente manera :

```
Procedure ModoDeVideo(Video, Modo : byte);
Begin
 Asm
   mov ah , video
   mov al , modo
   int 10h
 End
End;
```

De esta manera cuando queramos activar el modo gráfico MCGA utilizaremos la instrucción :

```
ModoDeVideo($10,$13);
```

Y para regresar al modo texto utilizaremos la instrucción :

```
ModoDeVideo($10,$04);
```

A partir de este momento se utilizaran pequeños programas de rutinas gráficas. Se dará por hecho que para entonces ya está activado el modo gráfico, en caso contrario los ejemplos no funcionaran. Por lo que quedan definidas la primera y la última línea del programa principal (Activar y desactivar modo gráfico).

5.2. MEMORIA DE VIDEO.

Dado que no existen rutinas ni para poner un punto, se debe que utilizar una programación desde lo mas básico. Para poner un punto en la pantalla la manera mas fácil de hacerlo es utilizando la memoria de video.

La memoria de video es un área que el sistema operativo reserva exclusivamente para el uso de lo que se despliega en el monitor. Si se escribe algo en esa posición de memoria, aparecerá automáticamente en video. Recordemos que las posiciones de memoria se manejan como un conjunto de 8 bits que forman un byte, para direccionar una parte en especial hacemos referencia a el segmento y al desplazamiento que ocupa ese byte en la memoria. Se puede adelantar que la dirección de memoria de el modo MCGA empieza en el segmento A000h y el desplazamiento 0000h.

Para entender mejor el formato de la memoria de video empezaremos desde el principio.

Si se tiene una resolución de 2 colores, cuántos bits se necesitan para representar estos 2 colores ?. La respuesta es sencilla 2 colores implica blanco y negro. Es decir hay color o no hay color. El "si" y el "no" se pueden representar como :

```
Si  = 1
No = 0
```

Por lo que bastara sólo un bit para representar 2 colores. Basandose en esto, en un byte se puede almacenar la información de 8 pixeles.

Y si son 4 colores posibles por pixel?

En este caso se hara la misma pregunta, con cuantos bits (unos y ceros) se arman 4 colores.

```
00 = color 1
01 = color 2
10 = color 3
11 = color 4
```

Es decir serán necesarios 2 bits. Por lo tanto en un byte se guardara la información de 4 pixeles.

De igual manera si se tratara de 16 colores se usarian 4 bits (ceros y unos) para armar 16 colores diferentes, y por lo tanto en un byte solo se podrían meter 2 pixeles.

Si se trata de un pixel que puede tener 256 posibles colores ?

En este caso y en los que le siguieran el procedimiento es exactamente el mismo.

Se empieza desde el color 0, el 1, el 2, etc. hasta el color 255, total 256 colores. Con cuantos unos y ceros se arman 256 números ?. La respuesta es con 8 bits, es decir :

```
00000000 = Color 0
00000001 = Color 1
00000010 = Color 2
00000011 = Color 3
    :          :
11111110 = Color 254
11111111 = Color 255
Total      256 Colores.
```

Por lo tanto cuando el monitor se encuentra en un modo gráfico que utilice 256 colores, se necesitara un byte completo para cada pixel.

Hasta aquí ya se sabe que un pixel es un byte, y el valor de este byte indica el número de color - de entre 256 - que tendrá ese pixel. Ahora que se hace con ese byte ?

Bien, el decir "resolución de 320 x 200" quiere decir 320 pixeles horizontalmente (columnas) y 200 pixeles verticalmente (renglones). Lo que indica que el total de pixeles que habrá en la pantalla es de 320 x 200 = 64000.

Si se sabe que cada pixel ocupa un byte, nos encontramos con la pequeña cantidad de 64000 bytes por pantalla. Este es el tamaño que tendrá la memoria de video para esta resolución. Es obvio que para una resolución de 1024 x 768 en 256 colores, se estaria hablando de una memoria de video que mide exactamente 786432 bytes, mucho mas grande que el tamaño de la memoria base de una PC (640kb). Es esta la razón principal por la que la mayoría de los juegos maneja resoluciones de 320 x 200. A menos que utilicen las ventajas del CDROM.

Si se representa la memoria de video como un arreglo continuo de 64000 bytes se tendra :

0	A000 : 0
1	A000 : 1
2	A000 : 2
.	.
63998	A000 : F9FE
63999	A000 : F9FF

Tabla 5.2.

El contenido de esta tabla será el valor que tiene la posición de memoria indicada. Esto sirve para ver que la memoria de video es continua como cualquier otra. De esta manera quizás resulte un poco difícil visualizarla al asociarla con el monitor, pero, es importante porque es como se maneja mas fácilmente.

Si se representa la memoria de video en una matriz de 320 x 200 se tiene.

	0	1	2	3	. . .	318	319
0	A000:0	A000:1	A000:2	A000:3	:	A000:013E	A000:013F
1	A000:0140	A000:0141	A000:0142	A000:0143	:	A000:027E	A000:027F
2	A000:0280	A000:0281	A000:0282	A000:0283	:	A000:03BE	A000:03BF
3	A000:03C0	A000:03C1	A000:03C2	A000:03C3	:	A000:04FE	A000:04FF
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
198	A000:F780	A000:F781	A000:F783	A000:F784	:	A000:F8BE	A000:F8BF
199	A000:F8C0	A000:F8C1	A000:F8C2	A000:F8C3	:	A000:F9FE	A000:F9FF

Tabla 5.3.

De esta manera se puede visualizar mejor. Se observa que la dirección inicial de la memoria de video empieza en el ángulo superior derecho y se va incrementando de izquierda a derecha y de arriba hacia abajo. La posición inicial se encuentra en el segmento y desplazamiento A000:0. La dirección siguiente será la A000:1, luego la A000:2, la A000:3, etc. hasta la A000:F9FF. F9FF es el número 63999 en hexadecimal. Esta última dirección será para el pixel que se encuentra en el ángulo inferior derecho.

La memoria de video es continua saltando de renglón en renglón. 64000 bytes en pixeles.

Para escribir en memoria de video utilizando Turbo Pascal, se utiliza la instrucción MEM, un ejemplo es :

```
MEM[$A000:$0005] := 34;
```

Los números en pascal van precedidos de el carácter $ para especificar que se trata de un número hexadecimal. El formato de la instrucción MEM es :

```
MEM[ segmento : desplazamiento] := valor;
```

En especial con la instrucción anterior se le dice a pascal que al pixel número 5 le ponga el color número 34. Si el fondo de la pantalla se encuentra negro por ejemplo y el color 34 es verde, cuando se escribe en memoria de video un 34 en la posición 5 de ese segmento, se pondra un pixel verde.

La forma de utilizar la memoria de video de la manera clásica en pascal es sustituir el procedimiento :

```
PUTPIXEL(X,Y,color).
```

esto se hace utilizando la siguiente función de mapeo :

```
posición = (Y-1) x 320 + X
```

El procedimiento que lo usa se puede definir de la siguiente manera :

```
Procedure PonPixel(X,Y : word; Color : byte).
Begin
  Mem[$A000:((Y-1)x320 + x)]:=color;
End;
```

De esta manera, para poner un punto en cualquier coordenada de la pantalla bastara con poner por ejemplo :

```
PonPixel(160,100,1);
```

Esta instrucción pondrá un punto en la mitad de la pantalla del color 1.

Cuando se escribe en memoria de video no existe riesgo de que se escriba en la dirección ocupada por alguna variable, pues como ya se mencionó es un segmento de memoria especial para manejo de video y es independiente de la memoria que pueda contener la tarjeta VGA a la que se encuentra conectado el monitor.

Si lo que interesa es saber que color se encuentra en una posición de memoria dada, podemos utilizar la misma función de mapeo. Ahora se define la función LeePixel.

```
Function LeePixel (X, Y : word) : byte;
Begin
  LeePixel:=MEM[$A000:((Y-1)x320+X);
End;
```

Por lo que si se quiere saber cual es el color del pixel que se encuentra a media pantalla se utilizara la siguiente instrucción :

```
Color := LeePixel(160,100);
```

Se concluye pues, que la memoria de video es un arreglo continuo de 64000 bytes, en el que cada byte representa un pixel, y el valor de el byte representa el número de color de la paleta de colores actual.

5.3. PALETA DE COLORES.

Una de las ventajas de este modo de video, es que, antes o durante el funcionamiento de un programa, se puede cambiar a voluntad cualquier color de los actuales. En el momento de activar el modo gráfico MCGA, pascal cuenta con una paleta estándar, que es la paleta que se utiliza en modo texto. Por lo general cuando se maneja una imagen, no se maneja la paleta estándar de manera que casi siempre se tiene que modificar.

Como se mencinonaba antes, se tendrán a disposición 256 colores, y se puede editar cada unos de estos colores. Cada color está formado por tres colores primarios; rojo, verde y azul. La intensidad que tengan éstos determinan el color resultante.

Cada uno de los tres colores primarios puede tener una intensidad que va desde 0 - que es lo más obscuro -, hasta 63 que es lo más luminoso. Para indicar un número entre 0 y 63 se necesita un byte. Dado que se trata de tres colores primarios, se requieren tres bytes para definir cada color de la paleta. El orden de los bytes de colores primarios es rojo, verde, y azul (RGB).

Se puede representar la paleta de colores como un arreglo continuo de 3 (colores primarios) x 256 (colores secundarios) de la manera siguiente.

Color secundario	Posición	Valor del primario
0	0	0
	1	0
	2	0
1	3	3
	4	3
	5	4
2	6	3
	7	4
	8	63
.	.	.
.	.	.
.	.	.
254	762	45
	763	32
	764	34
255	765	62
	766	42
	767	0

Tabla 5.4.

Se puede representar la paleta de colores como una matriz de 3 (colores primarios) x 256 (colores secundarios) de la manera siguiente.

Color secundario	Tono de rojo	Tono de verde	Tono de azul
0	0	0	0
1	3	3	4
2	3	4	63
.	.	.	.
.	.	.	.
254	45	32	34
255	62	42	0

Tabla 5.4.

Es decir una matriz de 768 bytes para manejar todos los colores. El valor de los colores primarios en la tabla anterior es sólo ilustrativo, y como se menciono pueden tener un valor entre 0 a 63.

Si se tienen los datos siguientes para el color 0 :

0	0	0	0

significaría que el nivel de rojo es el mas obscuro, el de verde y el de azul también. Esto generaría el color negro mas obscuro que se puede tener en el monitor (ausencia total de color).

Si por el contrario se pusiera en el color 0 los siguientes valores :

0	63	63	63

dado que los tres colores primarios tendrían la máxima luminosidad, se obtendría el color mas blanco posible.

Para obtener el rojo más intenso en el color 0 se usarían los valores :

0	63	0	0

Para obtener el verde más intenso en el color 34 se usarían los valores :

34	0	63	0

Para obtener el azul más intenso en el color 173 se pondrían los valores :

173	0	0	63

Para obtener un tono medio de rojo en el color 0 se pondrían los valores :

0	32	0	0

Para obtener un color que fuera combinación entre los tres primarios en el color 0 se pondrían los valores :

0	Valor	Valor	Valor

La forma mas fácil de obtener el valor de estos colores es experimentando entre diferentes valores de los colores primarios para ver el resultado. O bien con ayuda de algún editor de paletas (Ver Cápitulo de Software de Apoyo).

5.3.1. Modificación de la paleta.

De nuevo el DOS nos proporciona por medio de sus interrupciones una manera fácil de cambiar estos colores. Antes que nada se definira en el de pascal, un arreglo 768 bytes en donde se almacenaran los valores de la paleta de la siguiente manera.

```
VAR
 PALETA = ARRAY[0..767] OF BYTE;
```

Después, se define un procedimiento que servirá para modificar cualquiera de los valores del arreglo, según el número del color en la paleta y de la siguiente manera:

```
Procedure CambiaPaleta(pos,rojo,verde,azul : byte);
Begin
 Paleta[pos*3]:=rojo;
 Paleta[pos*3+1]:=verde;
 Paleta[pos*3+2]:=azul;
End;
```

De esta manera se modifica el color solo en el arreglo. Una vez que se han cambiado cambiado todos los colores deseados se procede a cambiarlos en el video utilizando de nuevo la interrupción 10h del DOS. Con el siguiente procedimiento :

```
Procedure PonPaleta;
Var
 seg1, ofs1 : word;
Begin
 seg1:=seg(paleta[0]);
 ofs1:=ofs(paleta[0]);
 asm
  mov ax, seg1
  mov es, ax
  mov ax, ofs1
  mov dx, ax
  mov ax, $1012
  mov bx, 0
  mov cx, 256
  int $10
 end;
End;
```

La subfuncion 1012h de la interrupción 10h sirve para poner una paleta como activa. En este caso se utiliza la paleta que se armo primero en el arreglo PALETA. Como se puede apreciar es necesario guardar en el registro ES la dirección del segmento en donde se encuentra el arreglo, y en DX el desplazamiento dentro del segmento. Es recomendable hacerse de un buen manual de interrupciones, para conocer perfectamente los parámetros utilizados por cada una de éstas.

Cuando se hace un cambio de video, ya sea gráfico a texto o de una resolución a otra, los valores de la paleta de colores se pierden, por lo que hará falta volver a utilizar el procedimiento anterior valiendose del arreglo en el que se encuentran los datos deseados.

En algunas ocasiones, sobre todo para hacer efectos especiales, será necesario conocer los valores de la paleta actual. Para esto se utiliza de nuevo la interrupción 10h subfuncion 1017h, que sirve para leer los datos de la paleta actual y ponerlos en una posición de memoria establecida, en este caso en el arreglo PALETA. A continuación se define el procedimiento completo que hace esto :

```
Procedure LeePaleta;
Var
 seg1,ofs1 : word;
Begin
  seg1:=seg(paleta[0]);
  ofs1:=ofs(paleta[0]);
  asm
    mov ax,seg1
    mov es,ax
    mov ax,ofs1
    mov dx,ax
    mov ax,$1017
    mov bx,0
    mov cx,256
    int $10
  end;
End;
```

De esta manera se puede aprovechar la paleta de una imagen leida de un formato gráfico conocido.

CAPITULO 6

SOFTWARE DE APOYO

CAPITULO 6
SOFTWARE DE APOYO

Cuando se empieza a desarrollar un sistema desde cero, se necesita utilizar herramientas de diseño generales ya hechas, en lo que se desarrollan las nuevas herramientas especializadas. Este también es el caso, para los que elaboran juegos para computadora. Por ejemplo, si se necesita un editor de animación que trabaje en un entorno gráfico con interface iconográfica, se deben diseñar las pantallas de fondo y los iconos para que personalicen el programa. Hacer un dibujo por medio de ceros y unos, o sea, utilizando una tabla de datos dentro del programa de diseño es un trabajo largo, tedioso e innecesario. Lo más fácil es utilizar un programa de dibujo ya hecho y diseñar todos los gráficos que se van a utilizar. Una vez terminado el programa, éste debe ser capaz de crear sus propios pantallas gráficas e iconos.

En este capítulo se mencionaran los paquetes de software que fue necesario utilizar, para el desarrollo de las herramientas de diseño para la elaboración del juego para computadora de esta tesis, así como la relación que tuvieron en las diferentes etapas que conformaron el programa completo. Todos los programas que se utilizaron eran originales, muchos de ellos incluidos junto con las tarjetas digitalizadoras y de sonido (Video Blaster, Video Spigot for Windows y Sound Blaster). Es conveniente hacer mención que no estamos promoviendo ningún paquete de software, existen otros programas similares con los que se obtienen los mismos resultados, ésta simplemente es una especie de guía para los programadores novatos, que les permite saber más o menos que tipo de programas pueden utilizar. Los programas que mencionaremos a continuuación nos fueron muy particularmente útiles.

6.1. STORY BOARD, LIVE! (IBM Corp.)

Este es un programa para la elaboración de gráficos. Se utilizó por su simplicidad en el manejo, su capacidad de edición en el modo gráfico de 320x200 pixeles y 256 colores, la facilidad de importar y exportar en formato PCX, GIF y BMP (que recientemente supimos como manipular), por la experiencia de uso, ya que fue de los primeros programas que se empezaron a utilizar y, por supuesto, porque no se desarrolló ningún programa que hiciera dibujos ya que esto está fuera de los objetivos del presente trabajo.

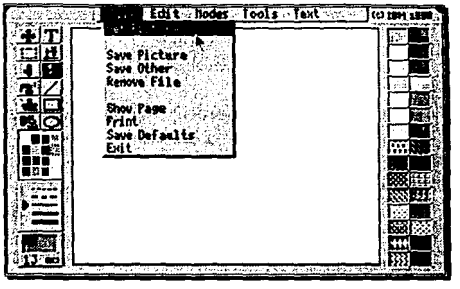


FIGURA 6.1. Pantalla de Picture Maker, la herramienta para dibujar de Story Board Live!.

Las principales aplicaciones fueron:

- Retocar imágenes digitalizadas. Cuando se captura una imagen por medio de una tarjeta digitalizadora se debe eliminar todo lo que no pertenezca al contorno de la figura, es decir, debe eliminarse el fondo. Además los rasgos de una persona u objeto se pierden al reducir la imagen, por lo que deben ser acentuados o modificados para que tengan la mejor apariencia posible.
- Crear iconos. Un icono es una pequeña gráfica que representa algún objeto, acción o tema. La facilidad edición en acercamiento (zoom), de Story Board Live!, permite diseñar de forma más conveniente los iconos que requerían los diferentes editores. Una vez dibujada una pantalla con varios tipos de iconos se guardaba en disco, para ser leída posteriormente por un programa diseñado especialmente para dividir una imagen en secciones y guardar una serie de bloques en disco (una libreria de bloques gráficos), que después se utilizaban los subprogramas que lo requirieran.

6.2. VIDEO FOR WINDOWS (Microsoft Corp.)

Este es un paquete que contiene diferentes programas para la edición de imágenes digitalizadas. Una imagen digitalizada puede provenir de una cámara de video conectado a una tarjeta especial que transforme una señal de video en pixeles. Este software viene incluido con las tarjetas Video Spigot y Video Blaster.

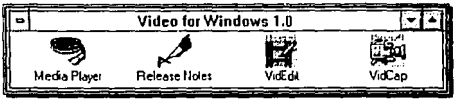


FIGURA 6.2. Grupo de trabajo para Video for windows.

6.2.1. VidCap Permite capturar una imagen o secuencia de imágenes. El tamaño de la ventana de captura puede ser definido por el usuario. Guarda los archivos con extensión *.AVI.

Se utilizó para capturar una secuencia de movimientos de las personas que se tomaron como modelos para los personajes del juego. También se usó para capturar las imágenes que sirven como fondo para cada diferente escena del juego.

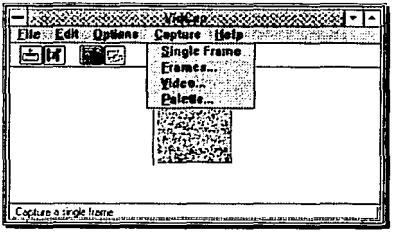


Figura 6.3. Ventana de trabajo para VidCap.

6.2.2. VidEdit. Lee, edita y guarda los archivos de video *.AVI. Permite seleccionar sólo aquellas imágenes que sean de utilidad, cambiar la paleta de colores y el formato de las gráficas, exportar a otros formatos (PCX, GIF, BMP, etc), cambiar de tamaño la secuencia de imágenes, etc.

Sirvió para elegir las imágenes que podían ser utilizadas en el juego, ya sea como paisaje de fondo o como personaje.

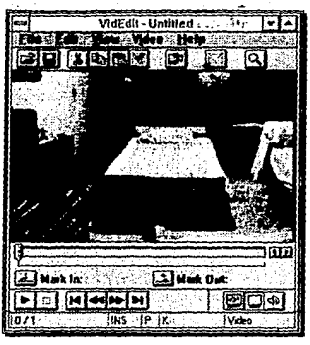


FIGURA 6.4. Ventana de trabajo para VidEdit.

6.2.3. PalEdit. Edita, y crea paletas (una paleta contiene todos los colores de una imagen).

Cada imagen digitalizada tiene, casi siempre, una paleta distinta de colores. Con PalEdit diseñamos una paleta que tuviera todos los tonos de colores más usados y luego igualamos los colores de cada imagen con nuestra selección de tonos para que no se presentara ningún problema al poner juntas dos imágenes distintas.

FIGURA 6.5. Grupo de trabajo para Multimedia Data Tools.

6.2.4. BitEdit. Edita y crea imágenes.

A pesar de que nuestro editor de gráficas base es *Story Board*, a veces era más conveniente utilizar este editor incluido en *Multimedia Data Tools* , en lugar de salirnos del entorno de windows y ejecutar el otro programa. Fue particularmente útil para cambiar de tamaño las imágenes y corregir pequeños detalles.

6.3. VOICE EDITOR 2.0 (Creative Labs)

Este programa que viene incluido en el conjunto de software que se proporciona al comprar la tarjeta de sonido Sound Blaster Pro y desarrollado por Creative Labs, es una herramienta muy útil para manejar diferentes sonidos e incluirlos dentro de un mismo archivo. Voice editor maneja archivos con el

formato *.VOC (usado por Creative Labs), y entre las modificaciones que permite hacer a cada archivo están:

- Amplificar sonidos (o reducir su volumen).
- Generar ecos.
- Modificar un conjunto de sonidos independientemente.
- Insertar marcas, etiquetas, silencios y repeticiones.

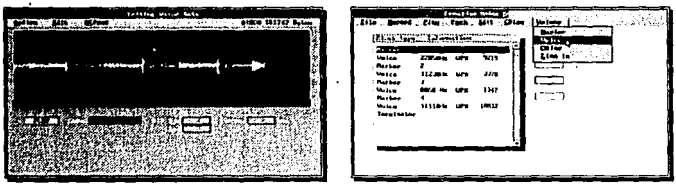


FIGURA 6.6. Las dos pantallas principales de Voice Editor 2.

6.3.1. JointVoc.exe

Desafortunadamente, no es posible insertar un archivo de sonidos independiente dentro de otro por medio del programa Voice editor, o grabar un sonido sin que se pierda el contenido de la memoria. Para poder juntar varios sonidos dentro de un mismo archivo se utiliza otro programa llamado JOINTVOC.EXE (otra utileria de creative labs), el cual nos permite especificar varios archivos de entrada y uno de salida. Ya una vez juntos todos los sonidos se usa Voice Editor para insertar marcas (una marca puede ser un número o una etiqueta), las cuales nos servirán para reconocer y diferenciar el inicio de cada sonido independiente. Por eso es importante saber de antemano, cuales y cuantos sonidos se utilizarán en el juego, de otro modo, cada sonido adicional deberá unirse al archivo principal y luego insertarle otra marca para diferenciarlo.

6.4. COMPOZ MUSIC UTILITY 1.0 (JOHN M. COON)

¿Alguna vez han escrito una canción?. Tal vez si, pero ¿le han puesto música a la letra?. Evidentemente la tarea del músico tiene sus méritos y requiere de conocimientos tales como incluir tiempos, notas, sostenidos, bemoles, instrumentos musicales, etc. y sobre todo tener buen oido musical. Por supuesto que no es lo mismo generar un sonido y grabarlo que llenar un pentagrama musical de forma coherente y que sea funcional. Es por eso que, para la música de fondo se utilizaron archivos ya hechos por músicos reales. Sin embargo, podemos realizar nuestras propias obras musicales con Compoz. Este programa genera archivos con su propio formato *.CZ, pero también puede exportar e importar archivos *.CMF (de Creative Music File), el cual es precisamente el formato que usamos para la música de fondo.

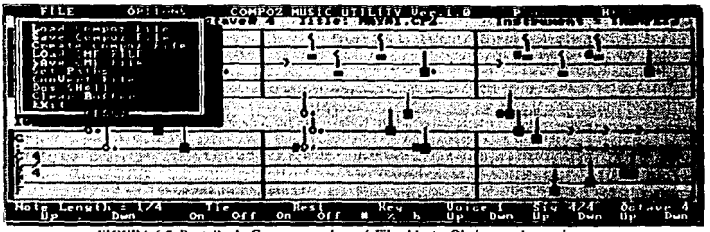


FIGURA 6.7. Pantalla de Compoz con el menú File abierto. Obsérvense las opciones para leer y salvar archivo CMF.

6.5. MANEJADORES DE CREATIVE LABS

Incluidos con todas las tarjetas Sound Blaster vienen los manejadores para la programación física de las mismas. En esta tesis se utilizaron los manejadores debido a la falta de bibliografía que explicara la programación a nivel físico. Se utilizaron dos manejadores: *SBFMDRV.COM* y *CT-VOICE.DRV*. En el capítulo ocho se explicará el código fuente, aquí nos limitaremos a describir las funciones de cada manejador.

6.5.1. Programar la tarjeta SoundBlaster

Aunque la programación de la tarjeta Sound Blaster no es particularmente difícil (desde una perspectiva de programación), se debe tener un nivel de programación aceptable. Programar toda la variedad de tarjetas Sound Blaster no es tarea para un programador novato.

Hay dos formas de programar la tarjeta Sound Blaster. La primera es programarla directamente accesando al DSP (Digital Signal Procesing : Procesamiento Digital de Señales). El otro método es utilizarlos manejadores de la Sound Blaster proporcionados en los programas que vienen con todas las tarjetas. Este método tiene sus ventajas, pero se sigue dependiendo de software de terceros y siempre es mejor saber programar a nivel hardware.

6.5.2. Programar usando los manejadores

Como se mencionó anteriormente, se puede ahorrar tiempo y esfuerzo si se utilizan los manejadores proporcionados con las diferentes tarjetas Sound Blaster. El único requisito es cargar el manejador en memoria antes del correr el programa para que este funcione. Existen varios tipos de manejadores para las tarjetas Sound Blaster, los más importantes son:

- El manejador de sonido FM
- El manejador de voz

EL MANEJADOR DE SONIDO FM

Si se decide utilizar el manejador de sonido FM ahorraremos una gran cantidad de problemas de programación. Este manejador se llama SBFMDRV.COM, y se tiene que cargar en memoria desde la línea de comandos del DOS, el cual es un programa TSR (Programa Residente en Memoria), accesible a los programadores a través de una interrupción. Cuando se carga, busca por un vector de interrupción libre entre 80h y BFh, utilizando el primero que encuentre. Encontrar el manejador puede ser engañoso. La primera cosa que se necesita es implementar una rutina de localización que lea los vectores entre 80h y BFh. Cuando se encuentra un vector que esté en uso, se asume que apunta al ajuste de dirección cero del manejador. En el ajuste 103h, se deberían encontrar las letras FMDRV. Esta cadena de cinco bytes identifica a SBFMDRV, y se puede utilizar esa interrupción para todos los demás accesos a las funciones del manejador. Si después de buscar en todos los vectores de interrupción no se puede encontrar la cadena de firma, el procedimiento de búsqueda debe regresar un error.

Las siguientes son las funciones que están disponibles para el manejador SBFMDRV:

Función 00h - Obtener la versión del manejador.
Regresa el número de versión del manejador SBFMDRV.
Registros de entrada
BX = 00h
Registros de salida
AH = Número mayor de la versión
AL = Número menor de la versión

Esta función debe ser usada al inicio de todos los programas para determinar la versión del manejador que se está usando. Algunas funciones no están disponibles en todas la versiones de los controladores, y puede usarse esta función para determinar cuales están disponibles.

Función 01h - Definir la dirección del byte de estado
Notifica al manejador la dirección del byte de estado
Registros de entrada
BX = 01h
DX:AX = Dirección del byte de estado de la música
Registros de salida
No regresa valores

El programa debe proporcionar la dirección de un byte de bandera , el cual será usado por el manejador para tenernos informados del progreso de la música que se está tocando. Este byte es puesto a cero bajo las siguientes circunstancias:

- Cuando se inicializa el manejador al usar la función 08h.
- Cuando se alcanza el final de un bloque de música.

- Cuando se detiene la música usando la función 07h.

El manejador cambia el byte de estado a FFh cuando se toca la salida de música por usar la función 06h y el byte de estado cambia su valor por el de cualquier dato de control hallado en el archivo de música.

Función 02h - Cambiar la tabla de instrumentos
Define el inicio de una tabla de instrumentos que se usará cuando se toque un archivo de música.
Registros de entrada
BX = 02h
CX = Número de instrumentos
DX:AX = Dirección de la tabla de instrumentos.
Registros de salida
No regresa valores.

Se utiliza esta función para redefinir la tabla de música interna usada por el manejador.

Función 06h - Toca la música
Define el inicio de un bloque de música y empieza a tocarlo.
Registros de entrada
BX = 06h
DX:AX = Dirección del bloque de música.
Registros de salida
AX = Estado de error

Esta función se usa para definir el inicio de un bloque de música y para empezar a tocarlo. La función regresa un 0 en AX si la música empezó a tocarse correctamente, o 1 si ya hubiese otro bloque de música tocándose.

Función 07h - Detiene la música
Para de tocar un bloque de música.
Registros de entrada
BX = 07h
Registros de salida
AX = Estado de error

Después de que un bloque de música se empezó a tocar con la función 06h, se puede detener con esta función. De regreso, AX será 0 si el bloque de música se detuvo, o 1 si no se había empezado a tocar ningún bloque de música.

Función 08h - Reinicia el manejador de FM
Reinicia el manejador de FM a las condiciones iniciales.
Registros de entrada
BX = 08h
Registros de salida
AX = Estado de error

Este comando se puede utilizar en la preparación para tocar un nuevo bloque de música o cuando se sale del programa. Reinicia los chips de síntesis de FM y pone la configuración del manejadores a sus valores por omisión. De regreso, AX es 0 si la reinicilización fue correcta, o1 si no se pudo efectuar. Un reinicio no puede llevarse a cabo si se está tocando un bloque de música. Se debe usar la función 07h para detener la música antes de que se pueda usar la función de reinicio.

Función 09h - Detiene la música
Temporalmente detiene la salida de música
Registros de entrada
BX = 09h
Registros de salida
AX = Estado de error.

Esta función se usa para suspender la salida del bloque de música. Si la pausa tuviera éxito, AX regresa 0. Si no estuviera tocándose un bloque de música, AX regresa un 1.

Función 0Ah - Continua la música
Continua tocando un bloque de música pausado anteriormente
Registros de entrada
BX = 0Ah
Registros de salida
AX = Estado de error

Si se ha usado anteriormente la función 09h para pausar un bloque de música, se puede usar esta función para empezar a tocar de nuevo. De regreso, AX será 0 si la música continua normalmente, o 1 si la música no hubiera estado en pausa.

USANDO EL MANEJADOR DE VOZ

El manejador cargable *CT-VOICE.DRV* puede ser usado para trabajar con los archivos VOC (voz). Aunque la estructura de los archivos VOC está fuera del contexto de esta obra, la información contenida aquí puede ayudar a comprender las funciones de programación disponibles por medio de este manejador.

Para usar el manejador, el programa debe leerlo del disco (esto no significa que deba ejecutarse desde la línea de comandos del DOS como sucede con algunos otros manejadores). Para cargar el manejador se debe proceder como sigue:

1. Localizar el archivo en el disco.
2. Determinar el tamaño del archivo.
3. Reservar suficiente memoria para cargar el archivo.
4. Cargar el archivo del disco a la memoria convencional.
5. Guardar la dirección donde está ubicado para después.

Se utiliza la dirección para acceder después al manejador por medio de una llamada lejana. El archivo debe cargarse en el ajuste 0 de este segmento de dirección. Cuando se llame al archivo, el valor de todos los registros se preserva excepto para AX y DX.

La siguiente sección detalla las diferentes llamadas a funciones disponibles con este manejador.

Función 00h - Obtiene la versión del manejador
Regresa el número de versión del manejador CT-VOICE.
Registros de entrada
BX = 00h
Registros de salida
AH = Número mayor de la versión
AL = Número menor de la versión

Esta función debe ser utilizada al principio de todos los programas de modo que se pueda determinar la versión del manejador que se está usando. Algunas funciones no están disponibles para todas las versiones, y puedes usar esta función para determinar cuales funciones son válidas.

Función 01h - Cambia la dirección de E/S
Cambia la dirección de E/S usa da por CT-VOICE.
Registros de entrada
BX = 01h
AX = Dirección base de E/S
Registros de salida
No regresa valores.

La dirección por omisión usada por el manejador es la misma que la usada en la Sound Blaster - 220h. Si la dirección base de E/S usada para la Sound Blaster es diferente de la de fábrica, se debe usar esta función para informarle al manejador del cambio. Debido a que virtualmente cada función del manejador es dependiente de su dirección, ésta debe ser una de las primeras funciones que se usen en el programa.

Función 02h - Cambia la dirección DMA
Cambia la interrupción DMA usada por CT-VOICE.DRV
Registros de entrada
BX = 02h
AX = Número de interrupción DMA
Registros de salida
No regresa valores

El número de interrupción DMA por omisión usado por el manejador es el mismo que el de la Sound Blaster - 7. Si la interrupción DMA de la Sound Blaster es diferente del valor por defecto, debe usarse esta función para cambiar el valor que usa el manejador. Esta debe ser una de las primeras funciones que use el programa.

Función 03h - Inicializa manejador
Reinicia el manejador CT-VOICE.DRV.
Registros de entrada
BX = 03h
Registros de salida
AX = Estado de error

Se usa esta función para preparar al manejador para programarse y que pueda recibir otros comandos. Debe usarse esta función una vez al inicio del programa. Las únicas funciones que se pueden usar antes son 00h, 01h, 02h y 13h.

De regreso, AX tendrá uno de los siguientes valores:

0 = No hay error.
1 = Versión incorrecta del manejador.
2 = Falla de E/S.
3 = Falla en la interrupción DMA.

Función 04h - Enciende o apaga el altavoz
Apaga o enciende el altavoz conectado al convertidor digital-analógico.
Registros de entrada
BX = 04h
AL = Ajuste del altavoz.
Registros de salida
No regresa valores.

Esta función es usada para apagar o encender el altavoz del DAC. Inicialmente es conectado cuando se usa la función 03h. Debe apagarse cuando se efectúa una grabación de voz o cuando termina el programa.

Función 05h - Cambia la dirección del Byte de estado.
Informa al manejador la dirección del byte de estado.
Registros de entrada
BX = 05h
ES:DI = Dirección de la palabra de estado.
Registros de salida
No regresa valores.

El programa debe proporcionar la dirección de una bandera de 16-bits, la cual es usada por el manejador para mantenernos informados del progreso de cualquier operación que se esté realizando. Esta bandera es puesta a cero cuando se inicializa el manejador por medio de la función 03h, o cuando se encuentra el final de un bloque de voz en memoria o en disco. Cuando se inicia una nueva función de E/S, la bandera adquiere el valor de FFFFh, y también toma los valores de cualquier tipo de marca encontrada en un archivo de voz.

Función 06h - Empezar salida de voz
Empieza a reproducir la información de un almacén temporal de voz.
Registros de entrada
BX = 06h
ES:DI = Dirección donde se almacena el bloque de datos de voz.
Registros de salida
AX = Estado de error.

Esta función se usa para empezar a procesar la porción llamada Bloque de Datos de Voz para un archivo VOC. La dirección ES:DI debe apuntar al inicio de este bloque. Esta función utiliza técnicas DMA para la salida de la información de voz. Así, el control del programa es regresado inmediatamente al programa después de la ejecución de esta función. Para monitorear el estado de la salida de voz, se checa el valor de la palabra de estado descrita en la función 05h.

De regreso, cualquier valor diferente de cero significa un error.

Función 08h - Detener la E/S de voz.
Detiene cualquier grabación o reproducción de voz en progreso.
Registros de entrada
BX = 08h.
Registros de salida
No regresa valores.

Esta función se utiliza para detener completamente cualquier operación de E/S de voz. Debe usarse después de que termine un bloque de datos de voz..

Función 09h - Terminar manejador
Prepara al manejador CT-VOICE.DRV para finalizar el programa.
Registros de entrada
BX = 09h
Registros de salida
No regresa valores.

Debe ser utilizado cuando se vaya a finalizar el programa para reiniciar la tarjeta Sound Blaster y ejecutar otras funciones de mantenimiento.

Función 0Eh - Empieza a reproducir de la memoria extendida.
Empieza a reproducir la información de un bloque de voz localizado en la memoria extendida.
Registros de entrada
BX = 0Eh
DS = Manipulador de memoria extendida.
DI:SI = Ajuste del bloque de voz en memoria extendida
Registros de salida
AX = Estado de error.

Esta función se usa para empezar a procesar la porción llamada Bloque de Datos de Voz en un archivo VOC. Difiere de la función 06h en que el bloque se localiza en memoria extendida en lugar de la memoria convencional. Por lo general esta función se utiliza para procesar grandes bloques de voz.

La dirección de ajuste en DI:SI, dentro del área de memoria especificada por DX, debe apuntar al inicio de la porción del Bloque de Datos de Voz en el formato para el archivo VOC. Esta función utiliza técnicas DMA para la salida de la información de voz. Así, el control es regresado inmediatamente después de ejecutar esta función. Para monitorear el estado de salida de voz, se debe checar la condición de la palabra de estado como se describe en la función 05h.

De regreso, cualquier valor diferente de cero significa un error.

Función 13h - Cambia el canal DMA.
Cambia el canal DMA usado por CT-VOICE.DRV.
Registros de entrada
BX = 13h.
AX = Número del canal DMA.
Registros de salida.
No regresa valores.

Esta función trabaja únicamente en las tarjetas Sound Blaster Pro, Pro2, o Pro Basic. El valor por omisión para el canal DMA es el mismo que se usa de fábrica para la tarjeta Sound Blaster. Si el canal DMA es diferente en la tarjeta que el valor de fábrica, debe usarse esta función para cambiar el que utiliza el manejador. Esta debe ser una de las primeras funciones que se utilicen al inicio del programa, justo después de las funciones 01h y 02h.

CAPITULO 7

EDITOR DE DUENDES Y ANIMACION

CAPITULO 7
EDITOR DE DUENDES Y ANIMACION.

Una de las herramientas creadas para la elaboración de juegos de video es el Editor de Duendes y Animaciones (AnimaGraf). Es la herramienta mas poderosa que proporcionamos puesto sirve para crear los diferentes actores, objetos y animaciones que intervendran en el juego, asi como la paleta de colores que se utilizará.

AnimaGraf no es exclusiva para el manejo de juegos de aventura, es capaz de crear personajes y objetos para cualquier tipo de juego, o bien escenas exclusivamente de animación. Cosa que no ocurre con el programa ejecutor, que sólo interpreta juegos de aventura.

Un duende es una secuencia de bloques gráficos rectangulares que son superpuestos uno después de otro para simular movimiento. Precisamente lo que hace AnimaGraf es recortar estos bloques y acomodarlos en la secuencia deseada.

El editor de animación permite definir una etiqueta como una secuencia de las distintas imágenes que conforman al duende y el desplazamiento que tendrá cada una de éstas con respecto a la anterior.

7.1. EDITOR DE DUENDES

7.1.1 Explicación del entorno

El programa AnimaGraf funciona por medio de un entorno iconográfico. A continuación se muestra su estructura.

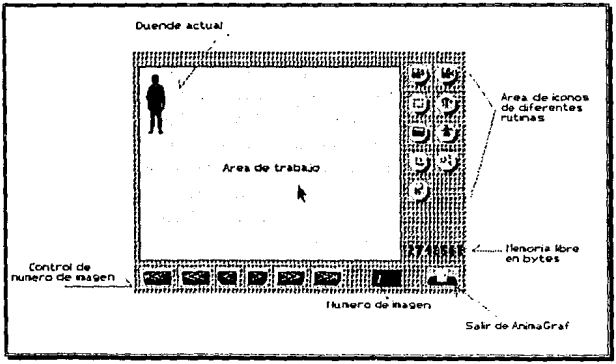


Figura 7.1. Explicación de las diferentes áreas del editor de duendes.

7.1.2. Manejador de archivos y directorios

En el programa AnimaGraf incorporamos para comodidad del usuario, un sistema de manejo de directorios y archivos, fácil y rápido de usar. Este aparece cada vez que se requiere de la lectura o escritura de información en disco. La figura 7.2 muestra la pantalla de manejo de archivos.

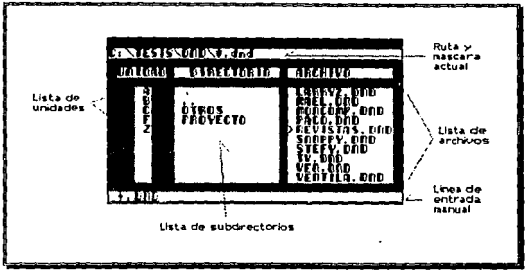


Figura 7.2. Manejador de directorios y archivos.

Este manejador se puede utilizar tanto con el ratón como con el teclado. Como se puede ver se selecciona la unidad en la que se encuentra o se encontrará el archivo, el subdirectorio y el nombre del archivo. Si se cuenta con ratón bastara con hacer un clic con el botón izquierdo sobre la unidad, el subdirectorio y el archivo. Si sólo se cuenta con el teclado, el uso es igual de sencillo, para moverse de izquierda a derecha y de arriba a abajo se hará con las teclas del cursor, y para seleccionar se hará con la barra espaciadora. En caso de que se trate de la escritura de un nuevo archivo bastará con teclear el nombre de éste y se escribirá en el subdirectorio y la unidad seleccionados.

Cuando se selecciona un subdirectorio automáticamente se despliegan los archivos que contiene según la mascara definida. El manejo de mascaras es similar al manejado por el dos, por ejemplo si se quieren todos los archivos de duendes bastará con teclear en la linea de entrada manual : *.dnd. O también se podria definir en la misma linea todo el directorio como : f:\tesis\dnd*.dnd.

Para cancelar el uso del manejador de archivos bastará con oprimir la tecla de escape o bien oprimiendo la tecla derecha del ratón.

7.1.3. Lectura y salvado de duendes

La lectura de duendes se hace oprimiendo el icono  , aquí aparece el manejador de archivos y se selecciona el archivo deseado. Cuando se lee un nuevo duende, el duende que se encontraba anteriormente en memoria es borrado y no existe manera de recuperarlo, por lo que hay que asegurarse de que se encuentre grabado en disco. Los duendes manejados utilizan la extensión .DND por lo que la mascara inicial en el manejador de archivos será f:\tesis\dnd*.dnd.

El salvado de duendes se hace oprimiendo el icono . Aquí aparece el manejador de archivos. Si se va a sobreescribir un archivo, el manejador de archivos pregunta si es lo que se quiere hacer. Si se trata de un archivo nuevo, bastará con teclear el nuevo nombre en la línea de entrada manual. Para escribir los datos de un duende en disco es necesario que exista un duende en memoria.

7.1.4. Area de trabajo y control de imagen

El área de trabajo muestra la imagen y el número de imagen del duende actual. Este número es importante pues es al que se hace referencia en el editor de animación. No importa el orden que tengan la imágenes en el duende, pero si interesa saber en donde se encuentra cada una de ellas.

Abajo del área de trabajo, aparecen unos botones que sirven para desplazarse a través de las imágenes del duende (Ver figura 7.1.). A continuación se explica cada una de ellos y las teclas equivalentes en el teclado, así como la tecla para salir del programa :

 Se posiciona en la primera imagen.	[INICIO]
Se posiciona en la última imagen.	[FINAL]
Incrementa la imagen actual en uno.	[Cursor derecha]
 Decrementa la imagen actual en uno.	[Cursor izquierda]
 Incrementa la imagen actual en cinco.	[Página adelante]
 Decrementa la imagen actual en cinco.	[Página actual]
 Salir del editor AnimaGraf	[ESCAPE]

7.1.5. Lectura y escritura de imágenes

AnimaGraf es capaz de leer archivos de imágenes que fueron creados por otros editores, estos archivos son muy comunes y son los que tienen los formatos GIF, PCX, AVI y archivos binarios (Para mayor información sobre estos archivos, referirse al anexo de la libreria LIBPAN.PAS). Para esto

seleccionamos el icono  que nos presenta dos opciones, leer  o escribir  . En cada una de las opciones seleccionadas aparecerá el manejador de archivos que ya conocemos.

En el caso de escritura , AnimaGraf sólo es capaz de escribir en formato binario utilizando el icono  .

Esto genera un archivo de 64768 bytes que no es más que una pantalla en MCGA. En algunas ocasiones es más conveniente manejar una imagen en binario que en otro formato, ya que en vez de comprimir la imagen y ocupar menos espacio, proporciona una imagen mucho mayor que ocupa más espacio. Este tipo de archivo es perfectamente leído por el programa ejecutor.

En el caso de lectura , aparece un submenú de iconos que permiten leer diferentes formatos de imágenes que son los siguientes :

- Lee archivos en formato GIF.
- Lee archivos en formato PCX.
- Lee archivos en formato AVI.
- Lee archivos en formato binario.

Cada formato gráfico cuenta con su propia paleta de colores. Sin embargo esta paleta puede ser desactivada con el icono de manejo de paletas para igualar la paleta en uso actual.

Los formatos GIF, PCX y BIN (binario), son sólo una imagen.

En el caso de los archivos AVI, se habla de un archivo que contiene varias imágenes, pueden ser decenas, cientos e incluso miles de imágenes (todo depende del disco duro), pero sólo se puede trabajar con una a la vez. Para seleccionar la imagen que adecuada de un archivo AVI se usarán las teclas del cursor para desplazarse y se acepta con la tecla de [INTRO].

7.1.6. Captura y borrado de duendes

La captura de las distintas imágenes que forman un duende se hace por medio del botón de bloques  , este muestra la imagen antes leída y un rectángulo de captura.

Este rectángulo se mueve a lo largo de la pantalla moviendo el ratón o utilizando las flechas del cursor en el teclado.

Si se está utilizando un ratón de tres botones, la tabla siguiente muestra la función de cada botón.

Botón del ratón	Acción que realiza.
Izquierda	Incrementa o decrementa, el ancho y largo del bloque.
Centro	Captura el bloque seleccionado.
Derecha	Sale del módulo de captura.

Tabla 7.1.

Si se cuenta con un ratón de 2 botones la captura se hará con la tecla [INTRO] del teclado.
Si se utiliza el teclado la captura de bloques se hará según la siguiente descripción:

Tecla.	Acción que realiza.
Flecha cursor izquierda.	Mueve el bloque de captura hacia la izquierda.
Flecha cursor derecha.	Mueve el bloque de captura hacia la derecha.
Flecha cursor arriba.	Mueve el bloque de captura hacia arriba.
Flecha cursor abajo.	Mueve el bloque de captura hacia abajo.
Inicio.	Decrementa el largo del bloque de captura.
Fin.	Decrementa el ancho del bloque de captura.
Pagina atrás.	Incrementa el largo del bloque de captura.
Pagina adelante.	Incrementa el ancho del bloque de captura.
Introducir.	Captura el bloque seleccionado.
Escape.	Sale del módulo de captura.
Mas (+)	Incrementa en pixeles los cambios en el bloque.
Menos (-)	Decrementa en pixeles los cambios en el bloque.

Tabla 7.2.

En caso de que se quiera borrar alguna de las imágenes del duende, se elige la imagen a borrar con las teclas de control de imagen y se oprime el icono que borrará sólo esa imagen.

La siguiente lista muestra los pasos necesarios para la captura de un duende.

- Se debe asegurar que no exista ningún duende en memoria, esto se ve cuando el cuadro de número de imagen no tiene un valor. En caso de que tenga un valor, en el área de trabajo aparecerá una imagen. Para borrar el duende actual se oprime el botón de bote de basura hasta que el área de trabajo esté despejada.
- Se habilita o deshabilita la igualación de la paleta actual. Esto es por si se quiere trabajar con una paleta fija o con la paleta de la imagen leída. Para hacer esto se entra al área de paleta con la opción .
- Se selecciona leer imagen con el icono . La imagen queda guardada en memoria.
- Se selecciona la opción de bloque con el botón que entra al área de captura y se procede a capturar todos los bloques requeridos.
- Por ultimo se graba el duende en disco oprimiendo el botón .

7.1.7. Copia de bloques de imágenes

El icono genera una subpantalla dentro del área de trabajo del editor como lo muestra la figura siguiente.

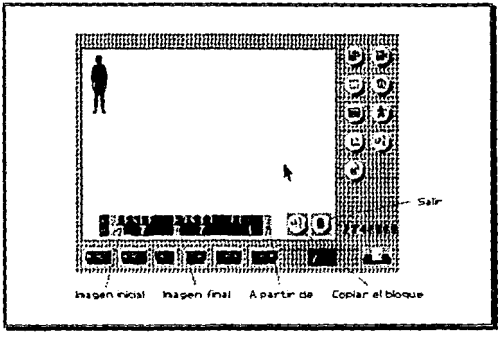


Figura 7.3. Copia de imágenes.

Este módulo sirve para hacer copias de bloques de una hasta el total de imágenes dentro del duende. A continuación se describen sus partes :

- **{DESDE}** Indica la imagen con la que inicia el bloque.
- **{HASTA}** Indica la imagen con la que finaliza el bloque.
- **{EN}** Indica la posición en la que se copiara el bloque.
- **{ + }** Incrementa en uno los valores anteriores. Se usa con el ratón.
- **{ - }** Decrementa en uno los valores anteriores. Se usa con el ratón.
- Ejecuta el proceso de copia.
- Sale del módulo de copia.

7.1.8 Modificación de imágenes.

En ocasiones son indispensables los espejeos de una imagen ya sea horizontal o verticalmente, asi como ciertas rotaciones. El editor de duendes puede hacer esas funciones. Para esto se selecciona primero la imagen que se va a modificar con los botones de control de imagen, se hace un clic del ratón directamente sobre la imagen que se encuentra en el área de trabajo. Cuando se hace esto aparece el siguiente submenú :

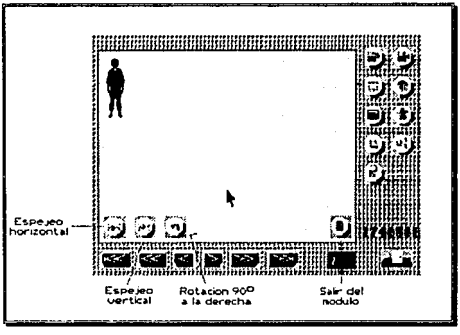


Figura 7.4. Rotación y espejeo de imágenes.

A continuación se describen los botones que aparecen en el submenu :

- Realiza un espejeo horizontal.
- Realiza un espejeo vertical..
- Rota la imagen hacia la derecha 90°.
- Sale del módulo de modificación de imágenes.

Cuando se realizan estas modificaciones se hacen directamente sobre el duende, y si se salva quedaran como se modificaron. Cuando se requiere de las dos imágenes espejeadas, lo que conviene es copiar primero la imagen y luego espejear la copia.

7.1.9. Manejador de paletas.

Cuando se maneja una imagen, ésta se aprecia según la paleta de colores que se está utilizando. Si se modifica la paleta la imagen se cambia drásticamente. Los duendes no incluyen una paleta propia, sino que dependen de la paleta de la imagen de fondo. Por lo tanto para que se vea bien se requiere que la imagen de fondo tenga la misma paleta con la que se editaron los duendes.

Ahora, dado que el duende necesita estar en diferentes escenas, se requiere que todas las escenas tengan la misma paleta, o algo mas complicado seria que cada escena tuviera sus propios duendes.

De cualquiera de las dos maneras que se quiera hacer, el editor de duendes es capaz de manejarlo.

Si se trata de paletas diferentes para cada escena, se tienen que editar duendes con la paleta de la escena en la que se utilizarán. Si se va a utilizar la misma paleta en todas las escenas y duendes, existe una manera de indicarle al editor que no modifique la paleta aunque la imagen leída tenga una paleta diferente. A esto se le llama igualación de paletas.

En el área de iconos se encuentra el icono que entra al manejador de paletas pero al mismo tiempo indica si la paleta está cerrada (no modificable) o abierta (modificable). Este botón puede ser :

 Indica que la paleta está abierta.

 Indica que la paleta está cerrada.

De cualquier manera oprimiendo la tecla se entra al siguiente submenu :

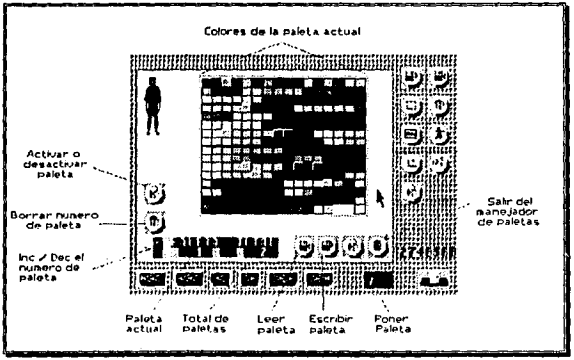


Figura 7.5. Manejador de paletas.

A continuación se describen sus botones :

⊙ **[PALETA]**	Indica el número de la paleta actual.
⊙ **[TOTAL]**	Indica el total de paletas en memoria.
⊙ [+]	Incrementa en uno el número de paleta.
⊙ [-]	Decrementa en uno el número de paleta.
⊙ 	Lee un archivo de paletas del disco.
⊙ 	Salva las paletas actuales en disco.
⊙	Activa la paleta actual.
⊙ 	Habilita y deshabilita el candado de la paleta.
⊙	Borra la paleta actual.
⊙ 	Sale del manejador de paletas.

Además se despliega una tabla con los 256 colores de la paleta actual. Los colores que aparecen con una línea blanca arriba y a la izquierda indican que están en uso. Es decir que por lo menos un pixel en toda la pantalla tiene ese color.

El formato de los archivos de paletas (*.pal) es descrito en el capitulo de Organización de Datos.

7.2. EDITOR DE ANIMACION.

Dado que las animaciones que se manejan en los juegos, serán animaciones de duendes, se decidió incorporar el Editor de Animación como una parte del Editor de Duendes.

El editor de animación se encarga simplemente de darle un orden a las distintas imágenes de un duende.

7.2.1. Explicación del entorno.

Para acceder al Editor de Animación se utiliza el botón . Para esto se debe tener un duende en memoria pues es el que se va a animar en caso contrario no se podrá entrar a este editor.

Una vez que se entra se verá la siguiente pantalla :

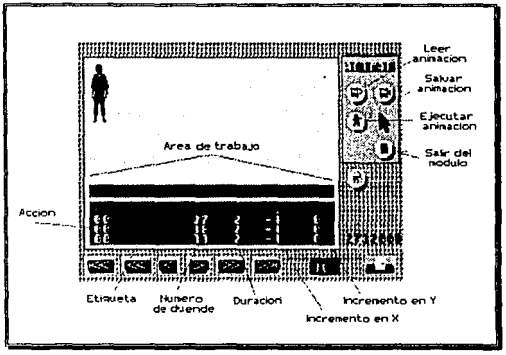


Figura 7.6. Editor de animación.

El editor de animación es el proceso mas importante de toda la tesis pues es el que le da vida a un duende. Su código completo se encuentra en el archivo LIBANI.PAS y el procedimiento principal que ejecuta las diferentes acciones de la animación se llama CORRE_ANIMA, que es utilizado tanto en el Programa Ejecutor como en el Editor de Animación. El Editor de Animación utiliza archivos (*.ANM). Su formato es descrito en el cápitulo de Organización de Datos. Sus partes se describen a continuación.

-  Lee un archivo de animación del disco.
-  Escribe un archivo de animación en disco.
-  Ejecuta una animación.
-  Sale del Editor de Animación.

En el área de trabajo aparecen los siguientes encabezados.

- **[E]** Indica la acción a realizar, es un número que va desde 0 a 99.
- **[ETIQUETA]** Es el nombre de la acción, es un texto de 8 caracteres.
- **[ND]** Número de duende, es un número de 0 a 99.
- **[DUR]** Duración, es un número de 0 a 999.
- **[INCX]** Incremento en x, es un número de -999 hasta 999.
- **[INCY]** Incremento en y, es un número de -999 hasta 999.

7.2.2. Acciones de la animación.

El funcionamiento de cada campo de este editor depende del número de acción que se realizará. A continuación se muestra una tabla de las distintas acciones disponibles.

Acción	Función a realizar
0	Despliega una imagen del duende.
1	Define el inicio de un procedimiento.
2	Define el final de un procedimiento.
3	Llama un procedimiento.
4	Redefine un procedimiento de otra animación.
5	Activa música de fondo.
6	Activa una frase de memoria.
7	Desactiva la frase desplegada.
8	Activa un sonido.
9	Lee un archivo PCX y lo pone en pantalla.
10	Genera un tiempo de espera.
11	Activa y desactiva el cursor en caso que exista.
12	Redefine el tipo de animación.
13	Define banderas.
14	Redefine la ventana de acción.
15	Salto absoluto de la animación a una posición.
16	Activa una nueva escena.

Tabla 7.3.

A continuación se describe el funcionamiento de cada campo dependiendo del número de acción.

- **Función** : Despliega una imagen del duende.
- **Accion** : 0
- **Etiqueta** : No utilizado.
- **Nd** : Número de imagen.
- **Dur** : Duración de la imagen.
  Si vale 900 la duración es infinita.
- **IncX** : Incremento posterior de x.
- **IncY** : Incremento posterior de y.

Función	: Define el inicio de una secuencia de instrucciones.
Acción	: 1
Etiqueta	: Indica el nombre del procedimiento. Absolutamente necesario.
Nd	: Número de imagen inicial.
Dur	: Duración de la imagen. Si vale 900 la duración es infinita. No es necesario definir el final.
IncX	: Incremento posterior de x.
IncY	: Incremento posterior de y.

Función	: Define el final de una secuencia de instrucciones.
Acción	: 2
Etiqueta	: Automática "FINAL"
Nd	: Número de imagen final.
Dur	: Duración de la imagen. Si vale 900 la duración es infinita.
IncX	: Incremento posterior de x.
IncY	: Incremento posterior de y.

Función	: Ejecuta una secuencia de instrucciones. Segun el nombre de la etiqueta.
Acción	: 3
Etiqueta	: Nombre del procedimiento a ejecutar. Definido con la acción 1.
Nd	: Depende de duración.
Dur	: Número de veces que se ejecutar el procedimiento. Si vale 900 la duración es infinita. Si vale 998 se ejecuta un número aleatorio de veces según ND.
IncX	: No utilizado.
IncY	: No utilizado.

Función	: Inicia una secuencia de una animacion ajena.
Acción	: 4
Etiqueta	: Procedimiento de la otra animación. Definido con la animación 1. Se encuentra en un archivo diferente.
Nd	: Número de la otra animación. Según el libreto.
Dur	: No utilizado.
IncX	: No utilizado.
IncY	: No utilizado.

- **Función** : Activa musica de fondo.
- **Acción** : 5
- **Etiqueta** : No utilizado.
- **Nd** : Número de la música en memoria.
- **Dur** : No utilizado.
- **IncX** : No utilizado.
- **IncY** : No utilizado.

- **Función** : Activa una frase de memoria.
- **Acción** : 6
- **Etiqueta** : No utilizado.
- **Nd** : Número de frase en memoria.
- **Dur** : Duración de la frase.
- **IncX** : Color del texto.
- **IncY** : Si vale 1 detiene la animación según la duración de la frase.

- **Función** : Desactiva la frase desplegada.
- **Acción** : 7
- **Etiqueta** : Automática "DES_LET"
- **Nd** : No utilizado.
- **Dur** : No utilizado.
- **IncX** : No utilizado.
- **IncY** : No utilizado.

- **Función** : Activa un sonido o voz.
- **Acción** : 8
- **Etiqueta** : No utilizado.
- **Nd** : Número de sonido en memoria.
- **Dur** : Número de veces que se repetirá.
- **IncX** : No utilizado.
- **IncY** : No utilizado.

- **Función** : Lee un archivo en formato PCX y lo despliega en pantalla.
- **Acción** : 9
- **Etiqueta** : Nombre del archivo en disco.
- **Nd** : No utilizado.
- **Dur** : No utilizado.
- **IncX** : No utilizado
- **IncY** : No utilizado.

- **Función** : Genera un tiempo de espera.
- **Acción** : 10
- **Etiqueta** : Automática "ESPERA"
- **Nd** : Se multiplica con los otros campos. El total es el tiempo de espera.
- **Dur** : Se multiplica con los otros campos. El total es el tiempo de espera.
- **IncX** : Se multiplica con los otros campos. El total es el tiempo de espera.
- **IncY** : Se multiplica con los otros campos. El total es el tiempo de espera.

- **Función** : Activa y desactiva el cursor.
- **Acción** : 11
- **Etiqueta** : Automática "CURSOR"
- **Nd** : 1 habilita el cursor.
  0 Deshabilita el cursor.
- **Dur** : No utilizado.
- **IncX** : No utilizado.
- **IncY** : No utilizado.

- **Función** : Redefine el el tipo de animacion.
- **Acción** : 12
- **Etiqueta** : Automática "DEFANIMA"
- **Nd** : Número de animación. Definido por el libreto.
- **Dur** : Nuevo tipo :
  0 - Animación normal.
  1 - Protagonista.
  2 - Objeto.
- **IncX** : No utilizado.
- **IncY** : No utilizado.

- **Función** : Modifica el valor de las banderas.
- **Acción** : 13
- **Etiqueta** : Automática "BANDERAS"
- **Nd** : Número de bandera.
- **Dur** : Valor de la bandera.
- **IncX** : No utilizado.
- **IncY** : No utilizado.

- **Función** : Redefine la ventana de acción.
- **Acción** : 14
- **Etiqueta** : Automática "VENTANA"
- **Nd** : Valor de X inicial.
- **Dur** : Valor de Y inicial.
- **IncX** : Valor de X final.
- **IncY** : Valor de Y final.

- **Función** : Salto absoluto de la animación a una posición.
- **Acción** : 15
- **Etiqueta** : Automática "X_Y_ABSO"
- **Nd** : Número de animación. Definida por el libreto.
- **Dur** : No utilizado.
- **IncX** : Posición en X.
- **IncY** : Posición en Y.

- **Función** : Activa una nueva escena.
- **Acción** : 16
- **Etiqueta** : Automática "ESCENA"
- **Nd** : Número de escena a habilitar.
- **Dur** : No utilizado.
- **IncX** : No utilizado.
- **IncY** : No utilizado.

CAPITULO 8

EDITOR DE MUSICA

CAPITULO 8
EDITOR DE MUSICA

Cuando se propuso el temario se tenía pensado utilizar la bocina de la computadora para los efectos sonoros, tanto de música como de efectos especiales, sin embargo, fue posible encontrar información para programar la tarjeta de sonido Sound Blaster y se optó por utilizar esta última para el juego. Desafortunadamente, tanto para el sonido como para la música es necesario cargar en memoria los manejadores (drivers) proporcionados por Creative Labs, lo que hace transparente para nosotros tanto el formato de los archivos como la forma de interactuar directamente con la tarjeta de sonido. Por esta razón no se diseñó un editor especial para música como se tenía planeado y en su lugar se explicará la forma de utilizar la bocina de la computadora y los manejadores especiales de la Sound Blaster.

8.1. LA BOCINA INTERNA DE LA COMPUTADORA

Bip. Bip. Tic. Tic. TiTu. Rrrr. ¿Adivinas qué es?. ¡Exacto!, son los sonidos típicos que se generan por medio de la bocina de la PC. Los sonidos se pueden usar para realzar una idea, llamar la atención sobre algún suceso o simplemente como música de fondo. Un juego que carezca de música y/o sonido, pierde una parte importante de la atención del usuario, los sonidos avisan si se acerca un enemigo, si hicimos algo correcto o incorrecto, si nos están hiriendo o golpeamos al enemigo, etc. Las melodías que vienen incluidas con los juegos le dan personalidad a las escenas y generalmente se toma una en particular como la representante oficial de ese juego y se utiliza en la siguientes secuelas de esa aventura (si las hay).

La computadora genera sonidos mandando pulsos eléctricos a la bocina[1], en intervalos diferentes para cada tono. Estos pulsos son de aproximadamente 5 Volts. Mientras mayor es la frecuencia de los pulsos más agudo es el tono, y mientras menor frecuencia es más grave.

Los sonidos que salen por la bocina de la computadora son generados por un contador de tiempo programable, el 8253. Este circuito es el que controla la frecuencia con que se mandan los pulsos eléctricos a la bocina. Cuando se le proporciona un número que representa el intervalo de tiempo entre dos pulsos, el 8253 cuenta los pulsos de reloj hasta que coinciden con el número dado, entonces manda un pulso al altavoz y empieza la cuenta de nuevo. Como la velocidad con la que se manda cada pulso de reloj del sistema es 1'193,180 se puede calcular el número que representa a la cuenta por medio de la siguiente expresión:

Cuenta = 1'193,180 / Frecuencia deseada

[1] Una prueba sencilla de como es que con un pulso eléctrico se puede producir sonido es, conectar una bocina cualquiera a una pila, en el momento de hacer contacto se oye un ruido como de chispa por un momento y después no se oye nada. Si se pudiera conectar y desconectar varias veces la pila, se escucharía un sonido constante, eso es precisamente lo que hace el 8253.

Las diferentes frecuencias pueden aproximarse a la notas musicales y entonces se usaría esta base para generar diferentes tonadas. No hay que esperar obtener sonidos de gran calidad por medio de la bocina de computadora, debido a que sólo se pueden generar dos variaciones en la amplitud del voltaje, a diferencia de un órgano electrónico que genera una señal analógica variable en su altavoz. Sin embargo, es posible generar sonidos de muy buena calidad digitalizando un sonido y adaptándolo a las frecuencias de la bocina.

8.1.1. Programar el 8253

Los sonidos en la bocina sólo requieren de los siguientes cuatro pasos: especificar una frecuencia, conectar la bocina, esperar un cierto tiempo y desconectar la bocina. La combinación de frecuencias y tiempos pueden producir tonadas musicales y efectos de sonido. Para programar en el 8253 se necesitan seguir los siguiente procedimiento:

- *Especificar una frecuencia.* Las frecuencias soportadas por el altavoz de las computadoras generalmente está entre los 100 y 5000 Hz. Los puertos en donde se especifican las frecuencias son el 66 y 67, los pasos a seguir son:
  1. Escribir en el puerto 67 el valor de 182 (prepara para recibir la cuenta).
  2. Escribir en el puerto 66 el byte más bajo del número entero que contiene la cuenta.
  3. Escribir en el puerto 66 el byte más alto del número entero que contiene la cuenta.
- *Encender la bocina.* El contador de tiempo 8253 siempre está activado, sin embargo sólo hasta que se le conecta se oyen sonidos por la bocina. Para conectarlo se debe:
  1. Leer el valor del puerto 97.
  2. Con el valor anterior realizar un OR con el número 3.
  3. Escribir el resultado de la operación anterior en el puerto 97.
- *Esperar un cierto tiempo.* Si se desconecta inmediatamente la bocina no se alcanzaría a escuchar ningún sonido o, tal vez, algunos chasquidos. Se puede utilizar un contador de tiempo o la instrucción *delay(Tiempo)*, de Turbo Pascal que se espera el número de milisegundos especificados por *Tiempo*.
- *Apagar la bocina.* Cuando ya no hay más notas o sonidos que enviar a la bocina debe desconectarse el altavoz de la computadora, ya que ésta no se apaga automáticamente al dejar de recibir datos. La secuencia para desconectar la bocina es:
  1. Leer el valor del puerto 97.
  2. Con el valor anterior realizar un AND con el número 253.
  3. Escribir el resultado de la operación anterior en el puerto 97.

  Las siguientes rutinas en Turbo Pascal engloban los pasos anteriores:

```
Procedure Frecuencia8253(Frecuencia : Word); { Especifica la frecuencia deseada}
 Var Cuenta : Word;
Begin
  Cuenta := 1193180 Div Frecuencia;  { obtiene la cuenta en base a la frecuencia }
                                     { deseada. }
  Port[67] :=  182;                  { Escribe el valor de 182 por el puerto 67 }
  Port[66] := Lo(Cuenta);            { Escribe el byte más bajo de la cuenta }
  Port[66] := Hi(Cuenta);            { Escribe el byte más alto de la cuenta }
End;
Procedure Bocina(Conectar : Boolean);  { Para conectar y desconectar la bocina }
Begin
  If  Conectar Then                  { Si Conectar es verdadero }
    Port[97] := Port[97] Or 3        { ... conectar la bocina.      }
  Else                               { Si Conectar es falso.      }
    Port[97] := Port[97] And 253;    { ... desconectar la bocina.  }
End;
```

Turbo Pascal proporciona dos instrucciones para el manejo del sonido *Sound*(frecuencia) y *NoSound*. Con *Sound* se inicializa el 8253 con la frecuencia puesta como parámetro y se conecta la bocina. *NoSound* simplemente apaga el sonido de la bocina. Es posible que te preguntes ¿qué objeto tiene entonces saber programar el 8253?, la respuesta tiene que ver con los sonidos digitales en la bocina de la computadora, para los cuales es más conveniente trabajar con cuentas en lugar de frecuencias (recuerda la fórmula Cuenta = 1'193,180/Frecuencia deseada). *Sound* utiliza exclusivamente frecuencias y utiliza el procedimiento de conectar continuamente la bocina, lo cual afecta la calidad de los sonidos digitales.

8.1.2. Rutinas residentes para incluir música de fondo

El procedimiento que controla los efectos sonoros debe ser llamado de forma constante, por lo que es recomendable utilizar rutinas residentes[2], lo cual permite que las acciones no se detengan y parezca que se está tocando la música de fondo de manera independiente. De otro modo, las acciones tendrían interrupciones de tiempos variables.

Un programa simple que explica como incluir música de fondo en un programa se muestra a continuación:

[2] La teoría de rutinas residentes e interrupciones se explica en el capítulo 3.

```
Programa Musica_de_Fondo;
Uses Dos, Crt;          { Unidades requeridas }

Type
 MusicaReg = Record          { Registro base para un archivo que maneje música.     }
  Frecuencia : Word;         { Solo se necesitan 2 datos, frecuencia y duración.    }
  Duracion   : Byte;         { Aunque en este programa no se usa, puede adaptarse }
 End;                        {  a un programa más formal.                          }
Const
 MaxNotas  = 107;            { Constante global que guarda el total de frecuencias y notas }
 EjemploMus : Array[1..MaxNotas*2] Of Word =          { Frecuencias y duraciones }
      (147,3,73,3,147,3,73,3,147,3,73,3,147,3,73,3,65,3,131,3,65,3,   { de un música de ejemplo.}
       131,3,65,3,131,3,65,3,131,3,147,3,73,3,147,3,73,3,147,3,73,3,
       147,3,73,3,65,3,131,3,65,3,131,3,65,3,131,3,65,3,131,3,147,3,
       73,3,147,3,73,3,147,3,73,3,147,3,73,3,65,3,131,3,65,3,131,3,
       65,3,131,3,65,3,131,3,147,3,147,3,147,3,147,3,147,3,0,1,294,3,
       147,3,294,3,147,3,294,3,147,3,294,3,147,3,131,3,262,3,131,3,262,3,
       131,3,262,3,131,3,262,3,294,3,147,3,294,3,147,3,294,3,147,3,294,3,
       147,3,131,3,262,3,131,3,262,3,131,3,262,3,131,3,262,3,294,3,147,3,
       294,3,147,3,294,3,147,3,294,3,147,3,131,3,262,3,131,3,262,3,131,3,
       262,3,131,3,262,3,294,3,294,3,294,3,294,3,0,1);
Var
 SalvaInt1C : Procedure;     { Aquí se guarda la dirección de la INT 1CH }
 NotaActual,                 { Indice del arreglo que contiene la tonada de ejemplo }
 Cuenta     : Word;          { Contador de duración de la nota actual. }

{$F+,S-}
PROCEDURE MusicaResidente; Interrupt;      { Procedimiento que sustituye a la INT 1CH }
Begin
  If Cuenta=EjemploMus[NotaActual*2] Then      { Si cuenta el duración de la nota actual... }
  begin
    If NotaActual<MaxNotas Then Inc(NotaActual)    { Pasa a la siguiente nota }
      Else NotaActual := 1;      { Si llegamos al final del arreglo, volvemos a empezar }
    Cuenta := 1;                 { Reiniciamos el contador de duración }
    Sound(EjemploMus[NotaActual*2-1]);    { Tocamos un sonido a la frecuencia dada }
  end Else Inc(Cuenta);                   { Incrementamos la cuenta de duración }
End;
```

```
{SF-,S+}

Var
  Tecla : Char;              { Aqui se almacena el valor de la tecla }
Begin
  GetIntVec($1C,@SalvaInt1C);                { Guardamos la dirección de la INT 1CH }
  SetIntVec($1C, Addr(MusicaResidente));     { Redireccionamos nuestro procedimiento }
                                             {  para que sustituyan a la INT 1CH.     }
  TextColor(0);                      { Realizamos un procedimiento interactivo cualquiera }
  TextBackGround(7);                 {  para ilustrar como se puede realizar otra actividad }
  ClrScr;                        {  mientras hay música de fondo. }
  Writeln('Pulsa ESC para Salir');
  Cuenta := 1;
  NotaActual := 1;
  GotoXY(28,7);
  Write('Prueba de Interactividad');
  GotoXY(15,8);
  Write('Pulsa la tecla alfanumérica que quieras que se repita');
  Window(20,10,60,15);
  Tecla := 'X';
  TextColor(7);
  TextBackGround(0);
  repeat
    Write(Tecla);
    If KeyPressed Then Tecla := Readkey;
    If Tecla=#0 Then
    begin
      Tecla := Readkey;
      Tecla := '.';
    end;
    If Not (Tecla In [#27,#32..#165]) Then Tecla := '.';
  until Tecla=#27;                   { Rompemos el ciclo si se oprimió la tecla [ESC] }
  NoSound;                       { Apagamos el sonido. }
  Window(1,1,80,25);                 { Restauramos la ventana normal. }
  SetIntVec($1C, Addr(SalvaInt1C));  { Restauramos la dirección de la INT 1CH }
  ClrScr;                        {  y borramos la pantalla. }
End.
```

Se observa la estructura *MusicaReg*, es la configuración típica de una nota que contiene frecuencia y duración. Aunque no se usa en este programa se incluyó para ayuda a comprender el arreglo *EjemploMus*, el cual contiene los datos de la tonada que se toca como música de fondo. Los valores nones son las frecuencias y los pares la duración. Notarás también que no se llama al procedimiento *SalvaIntIC* (el cual llama las rutinas de la INT 1CH) en nuestra rutina residente *MusicaResidente* como se recomienda en el capítulo 3. Esto es debido a que la INT 1CH no desempeña un papel crítico en el programa. Si este programa quedara residente, es muy probable que si tuviera que llamarse continuamente.

El funcionamiento del programa es simple:

- Se guarda el valor de la INT 1CH y se sustituye por el procedimiento *MusicaRes*. Casi inmediatamente se empieza a escuchar la música.
- Se ejecutan los procesos necesarios, la música seguirá tocando pase lo que pase (Excepto por supuesto que se apague la computadora).
- Una vez terminados todos los procesos que se hubieran requerido, se apaga la música y se restaura la dirección de la INT 1CH en la tabla de vectores de interrupción.

8.2. AUDIO DIGITAL

Antes de que las representaciones de audio fueran desarrolladas, la información de audio era guardada en formato analógico. El audio analógico tiene casi la forma del sonido que escuchamos -el sonido es representado por una fluctuación en un medio. Este medio es casi siempre el aire que nos rodea, y el sonido se crea comprimiendo el aire a diferentes frecuencias. Este medio no es sólo el aire, también puede ser un líquido o un sólido, sin embargo, no suenan igual debido a que las compresiones y descompresiones ocurren en un medio distinto.

Los sonidos pueden representarse eléctricamente por fluctuaciones en una onda senoidal regular. Existen diferentes forma de representar estas fluctuaciones, por ejemplo, la modulación en amplitud cambia la componente en amplitud de una señal para representar el sonido.

Por otro lado, el audio digital es una representación del sonido que es compatible con las computadoras. Esto significa que el sonido es representado por una serie de dígitos binarios que luego pueden ser guardados en archivos, cargados en la memoria de la computadora, y procesados como cualquier otro dato digital. El audio digital es menos susceptible a la degradación y distorsión, y por lo tanto es más parecido a los sonidos reales.

Hay varios conceptos del audio digital que es importante entender. Están relacionados con la calidad del sonido, el tamaño del archivo de audio y el tiempo requerido para procesar el archivo. Estos conceptos son: Muestreo, Frecuencia de muestreo, tamaño de la muestra y canales.

8.2.1. Muestreo

Los sonidos digitales se construyen a base de muestras que se toman de una fuente de sonido analógico. Estas muestras son guardadas en memoria y posiblemente escritas en un archivo de disco. El proceso de muestreo se conoce también como CAD, que significa Conversión Analógica a Digital (o ADC por sus siglas en inglés). La siguiente figura muestra como ocurre el proceso.

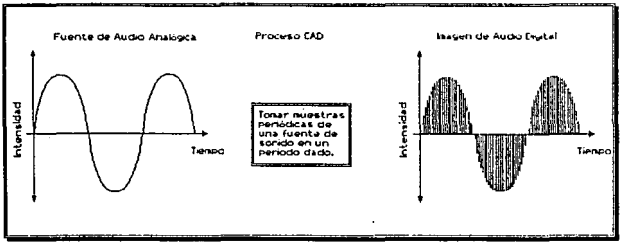


Figura 8.1. El proceso CAD

Una muestra de audio es tomada con cierta frecuencia llamada frecuencia de muestreo, y de un cierto tamaño conocido como tamaño de la muestra.

8.2.2. Frecuencia de muestreo

La frecuencia de muestreo se refiere a que tan seguido el programa de grabación toma una muestra de la fuente de sonido. Esta frecuencia se expresa en términos de *herts*, o *Hz*, que son ciclos por segundo. Una frecuencia de muestreo de 5,000 Hz significa que una muestra del sonido es tomada 5,000 veces por segundo. Cuando se quiere determinar que frecuencia de muestreo usar, debe recordarse que mientras mayor sea la frecuencia de muestreo, mejor es la calidad. Esto se debe a que el sonido es más continuo, con menos saltos notables entre cada muestra. El único problema es que cuando aumenta la frecuencia de muestreo también aumenta la cantidad de memoria necesaria (ya sea RAM o espacio en disco) para contener las muestras.

8.2.3. Tamaño de la muestra

El tamaño de la muestra se refiere al número de bits usados para cada muestra de audio. Mientras mayor sea el tamaño de la muestra mejor calidad tendrá el sonido digital. La mejor calidad de audio de entre todos los dispositivos de grabación la tienen los discos compactos (CD's) , los cuales tienen una frecuencia de muestreo de 44 KHz y un tamaño de la muestra de 16 Bits.

8.2.4. Canales

Cuando se graba un sonido digital, es posible grabar un solo canal (monoaural) o dos canales (estéreo). Estos son términos con los que casi siempres estamos estarás familiarizados. Existe una gran diferencia de calidad entre los sonidos mono y estéreo, y se puedes notar fácilmente la diferencia escuchando ambos tipos.

8.2.5. Determinar el tamaño del archivo

Un juego que tiene voces digitalizadas y efectos de sonido, adquiere un grado mayor de realismo. Sin embargo, el sonido digitalizado puede ocupar una cantidad de espacio en memoria demasiado grande lo que nos impedirá cargar otros datos que necesite el programa. La siguiente fórmula describe básicamente como determinar el espacio en memoria (o disco) requerido por cada segundo de grabación:

Espacio = Tamaño x Frecuencia x Canales

donde *Tamaño* es el tamaño de cada muestra en bytes, *Frecuencia* es la frecuencia de muestreo, y *Canales* el número de canales utilizados.

Si se desea tener un juego con efectos de sonido digital de alta calidad y en estéreo se necesitan muchos recursos de memoria. Por ejemplo, 1 minutos de sonidos de 16 bits (o sea 2 bytes), a una frecuencia de 44 KHz y en estéreo requerirán; 60x2x44000x2 = 10'560,000 bytes ¿Cuanta memoria RAM tiene la computadora?, ¿De cuánto espacio disponemos en el disco duro?. Debe recordarse que son 10 MBytes ¡por minuto!. Hoy en dia la computadoras tienen por lo regular 4 MegaBytes de memoria RAM, por lo que los sonidos digitales se graban en modo monoaural, a 8 bits y con una frecuencia de 11 MHz, para incluirlos en un juego para computadora (y aun así se oye bastante bien), lo que significa; 60x1x11000x1 = 660'000 bytes por minuto.

8.3. SINTESIS DE FM

La síntesis de FM permite crear sonidos que se acercan a los creados por instrumentos musicales. Está basada en una tecnología pionera que a principios de los setentas se desarrollara por John Chowning en la Universidad de Stanford. Esta tecnología involucra el uso de múltiples ondas de audio a baja frecuencia, para crear una onda resultante que se aproximara al sonido producido por los instrumentos musicales.

8.3.1. Operadores y celdas

La síntesis de FM es producida por la combinación de múltiples ondas de audio a baja frecuencia. Cada una de estas ondas es comúnmente conocida como *operador*. Los operadores son generados por circuitos electrónicos en el chip sintetizador de FM llamados *celdas de operador*. Hay diferentes parámetros para las celdas, los cuales se basan en el tipo de chip que contienen las celdas y pueden ser ajustados para crear diferentes sonidos.

Observemos el siguiente ejemplo donde se utilizan dos celdas para producir un sonido. La Figura 8.2 muestra un ejemplo simplificado de un arreglo de celdas. Obsérvese que hay dos celdas. La entrada a la primera es una onda senoidal pura. La celda modifica la onda de acuerdo a los parámetros mencionados en el párrafo anterior. La onda resultante es combinada con la segunda onda senoidal y realimentada a la primera celda. La salida de la primera celda es usada como entrada de la segunda, donde se termina de modificar la onda.

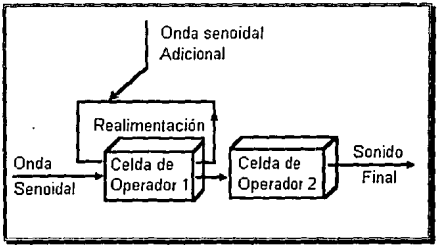


Figura 8.2. Un arreglo de celdas para síntesis de FM simplificados

La salida final de la segunda celda, debería ser un sonido musical. Hay tantas variables que afectan el sonido final, que un cambio en cualquiera de ellas puede producir un sonido marcadamente diferente. Los sonidos que correspondan exactamente a los de instrumentos musicales son generalmente el resultado de la prueba y error.

El número de celdas usadas en la síntesis de FM depende de como fue diseñado el chip y como se usa en la tarjeta final.

8.3.2. Diseñando la envolvente

Uno de los conceptos usados en la síntesis de FM es el de *envolventes*. La envolvente no es más que un patrón de amplitud usado para representar un sonido. La envolvente es definida por cuatro variables: ataque, caída, sostenido y liberación (ACSL o ADSR por sus siglas en inglés: Attack, Decay, Sustain y Release). Una envolvente típica definida por estas variables se muestra en la figura 8.3.

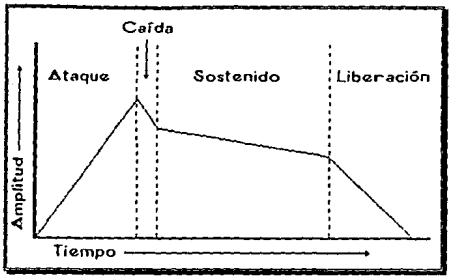


Figura 8.3. Una envolvente de amplitud definida por ACSL.

Las cuatro variables, ACSL, se definen típicamente en términos de rangos, pero dos de éstas (ataque y caída) incluyen una amplitud (una cantidad) y una duración.

El rango de ataque representa que tan rápido un sonido alcanza su amplitud máxima. Puede ser definida por la amplitud máxima y el periodo en que se alcanza.

El rango de caída es el declive inicial en amplitud después de que la máxima amplitud fue alcanzada. Está definida por la nueva amplitud y el periodo de tiempo en que es alcanzada.

El rango sostenido es, por definición, la duración en que la amplitud al final de la caída se mantiene.

Finalmente, el rango de liberación es el periodo de tiempo en que la amplitud llega a cero.

Esta envolvente de amplitud virtualmente existe para cualquier sonido musical imaginable. Cuando se presiona una tecla de piano, o se toca una trompeta, la nota inicial se incrementa a un volumen máximo (amplitud) muy rápido -el rango de ataque. Entonces disminuye un poco, de nuevo muy rápido -el rango de caída. El nivel volumen se mantiene un momento -el rango de sostenido. En el piano, este rango de sostenido puede ajustarse usando el pedal de sostenido. En la trompeta, se logra simplemente con mantener la nota un poco más. Finalmente, el volumen de la nota se desvanece y deja de escucharse.

Esta envolvente, y las cuatro variables que la definen, pueden ser adaptadas fácilmente a los sonidos producidos por cualquier instrumento musical. Este "moldeado de ondas" es la base de muchas de las modificaciones hechas por las celdas de los operadores en un chip de sintetizador de FM.

8.3.3. Armónicas

Mucha de la riqueza, frecuentemente conocida como timbre, de un instrumento musical es determinado por las armónicas producidas por el instrumento.

Consideremos, por ejemplo, un instrumento de cuerda. Cuando la cuerda es rasgada (o golpeada, en el caso del piano), la nota resultante no es un sonido puro a una cierta frecuencia. En su lugar, un amplio rango de armónicas se suma a la nota por la vibración de la cuerda.

Estas armónicas son múltiplos de la frecuencia base de la nota. Así, si la frecuencia base de una nota es 440 Hz, la primera armónica es 880 Hz. Armónicas adicionales ocurren a 1320 Hz, 1760 Hz, y así sucesivamente. Cada armónica disminuye en amplitud a partir de la frecuencia base. Esto simplemente significa que la frecuencia base es el sonido más fuerte que se escucha, la primera armónica tiene un volumen menor, la segunda todavía menos, etc. Puede haber más de una docena de armónicas diferentes para cada nota. Todas se suman para crear el sonido que se escucha cuando se toca la nota.

La creación de armónicas y el sonido resultante siempre ha sido el punto débil en la creación de música electrónica. ¿Qué armónicas deben sumarse? ¿Para qué instrumentos? ¿A qué volumen? ¿Por cuánto tiempo? Todas estas variables afectan el sonido final.

En el ejemplo de las celdas mostrado en el inciso de Operadores y Celdas, estas armónicas son introducidas por la realimentación de partes de la señal electrónica al procesador, o sea el ciclo que se efectúa en la primera celda. Este procedimiento multiplica las armónicas y agrega timbre al sonido.

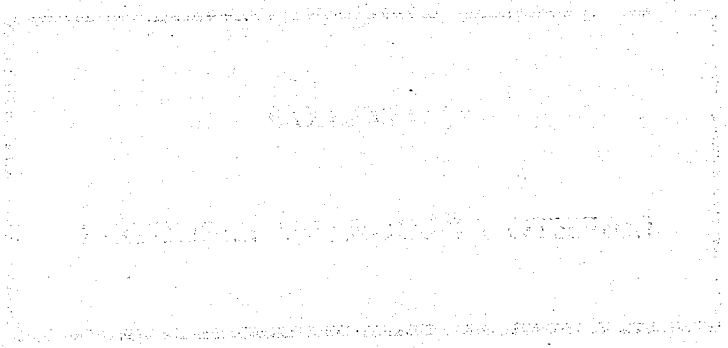
8.4. EL USO DE ARCHIVOS CMF

Todos los archivos de música que se utilizan en el juego para computadora que se incluye con esta tesis están en formato *CMF* (Creative Labs Music File). Para poder utilizar este formato se utiliza el manejador *TSR* que se incluye con la tarjeta Sound Blaster Pro; SBFMDRV.COM, el cual debe ser cargado antes de empezar el programa y quitado de la memoria al finalizar. Las funciones del manejador que son utilizadas se explicaron más ampliamente en el capítulo seis : software de apoyo.

Asimismo, todos los sonidos y efectos especiales se encuentran en un archivo de formato VOC, el cual utiliza el manejador *CT-VOICE.DRV*. La forma correcta de cargar el archivo se explica en el capítulo seis, así como también se proporciona una referencia de todas la funciones que son necesarias.

CAPITULO 9

LIBRETO Y PROGRAMA EJECUTOR



CAPITULO 9
LIBRETO Y PROGRAMA EJECUTOR

9.1. LIBRETO

Una vez creados los actores, las acciones que pueden realizar y los objetos que pueden utilizar, se procede a decidir el orden de aparición de cada elemento. El *libreto* se usa para crear cada una de las escenas de las que consta el juego de aventuras, se puede especificar cual será la pantalla de fondo, qué animaciones y qué objetos aparecerán, que actor encabezará el reparto, etc.

9.1.1. Definición de libreto

LIBRETO: Herramienta que nos permite definir e integrar todos los elementos que conforman una escena específica en una sola estructura de datos.

Aunque el libreto es solamente una base de datos, su contenido representa (junto con los cursos de acción definidos en las animaciones) el alma del juego. El libreto por regla general es usado por la persona que desarrolla la historia, ya que éste es una trabajo que combina la creatividad y una visión global de la trama del juego.

9.1.2. Explicación de la pantalla

El programa de libreto al igual que el programa editor también funciona por medio de un entorno iconográfico. Las diferentes áreas de la pantalla se explican en la figura de abajo.

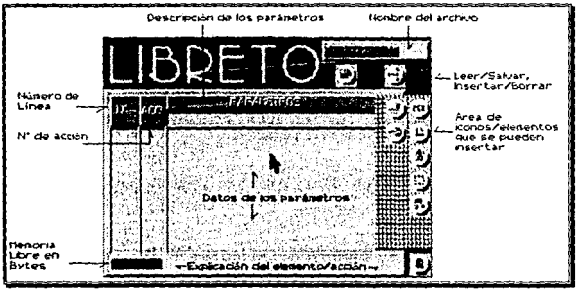


Figura 9.1. Explicación de las diferentes áreas del Libreto.

El objetivo del libreto es tomar los elementos que se necesiten y acomodarlos en el lugar que les corresponde, evidentemente es una especie de base de datos mas que un editor, asume que todos los

elementos o acciones han sido creados o modificados por los otros editores. El libreto es el penúltimo paso en la creación de un juego para computadora, el último paso es el programa ejecutor.

9.1.3. Acciones y elementos

Las acciones que se pueden realizar son:

Insertar Duende : Carga en memoria un duende. El primer duende siempre se considera como el protagonista.

Parámetros (2) :

- No. : Se incrementa/decrementa automáticamente con cada duende que se inserta/borra. No se puede modificar por el usuario.
- Nombre- : Cadena de ocho caracteres que especifica el nombre del archivo con extensión .DND donde se almacena el duende.

Subdirectorio de datos: ..\DND

Inserta Animación : Carga un listado de todas las acciones que puede realizar un duende y se declara una etiqueta con la acción inicial. Necesita que previamente se halla insertado el duende a que hace referencia.

Parámetros (6):

- No. : Se incrementa/decrementa automáticamente con cada animación que se inserta/borra. No se puede modificar por el usuario.
- -Nombre- : Cadena de ocho caracteres donde se guarda el nombre del archivo con extensión .ANM donde se almacena la animación.
- Etiqueta : Cadena de ocho caracteres con el nombre de la referencia que se ejecuta por defecto al leerse el archivo de animación.
- Dnd (Duende) : Número de duende al que hace referencia la animación.

Subdirectorio de datos: ..\ANM

- Coordenadas (X1, Y1) : Posición inicial de la animación.

Inserta PCX : Lee un archivo PCX de disco y lo copia en la pantalla virtual, la cual es usada como fondo para toda la escena.

Parámetros (1) :

- -Nombre- : Una cadena de ocho caracteres que especifica el nombre del archivo con extensión .PCX donde se guarda la gráfica.

Subdirectorio de datos : ..\PCX

Inserta Música FM: Lee un archivo de música en memoria y lo empieza a tocar inmediatamente un cierto número de repeticiones.

Parámetros (3):

- Se incrementa/decrementa automáticamente con cada tonada musical que se inserta/borra. No se puede modificar por el usuario.
- -Nombre- : Una cadena de ocho caracteres que especifica el nombre del archivo con extensión .CMF donde se almacena la tonada musical.
- Rep (repeticiones) : El número de repeticiones de la tonada musical, 99 = Infinito.

Subdirectorio de datos: ..\SB

 Insertar Objeto: Convierte una animación en objeto, o sea, en una entidad que puede tomarse y/o usarse. Necesita que se halla insertado previamente una animación y un duende a los cuales representar.

Parámetros (2)

- No. : Número del objeto
- (Anm) Animación : Reservado para aplicaciones futuras.

Subdirectorio de datos: Ninguno.

 Inserta Restricción : Especifica las zonas donde no puede pasar el duende protagonista o la zonas donde al señalarse con el cursor y presionarse el primer botón del ratón desplegarán un mensaje.

Parámetros (5).

- No. : Número de restricción. Se puede modificar por el usuario. Un valor de cero significa que es una zona donde al señalarse con el cursor y pulsarse el botón izquierdo del ratón se desplegará una frase referente a ese objeto. Un valor diferente de cero significa que es una restricción.
- Coordenadas (X1, Y1)-(X2, Y2) : Cuatro parámetros que especifican el ángulo superior izquierdo (X1, Y1) y el ángulo inferior derecho (X2, Y2) de un recuadro imaginario que servirá de restricción o de zona de mensajes.
- Anm (Animación o mensaje) : Si el valor de *Número* es cero representa el número del mensaje que se desplegará en pantalla. Si el valor de *Número* es diferente de cero representa el número de animación alrededor de la cual se pone la zona de restricción, para este caso un valor de cero es una restricción que hace referencia a la pantalla.

Subdirectorio de datos : Ninguno.

 Inserta Acción sobre Objeto : Especifica la acción que se realizará al utilizar un objeto con una animación.

Parámetros (8):

- Obj (Objeto) : Número de objeto que actuara sobre *Anm*.
- Anm (Animación) : Número de la animación sobre la que se hace referencia.

- Etiqueta: Cadena de ocho caracteres que representa la etiqueta de la animación *Anm* que se ejecutará al inicio.
- Ex1 (Acción 1): Bandera general que se modifica.
- E2 (Acción 2): Valor de la Bandera general *Ex1*.
- BN1 (Bandera 1): Reservado para futuras modificaciones.
- B2 (Bandera 2): Reservado para futuras modificaciones.
- BN3 (Bandera 3): Reservado para futuras modificaciones.

Subdirectorio de datos: Ninguno.

Otros iconos:

 : Leer libreto.
 : Salvar libreto.
 : Cancelar inserción.
 : Borrar elemento.
 : Insertar elemento.
 : Salir del programa Libreto.

9.1.4. Formato de archivos

El programa libreto proporciona un archivo de salida para cada escena que es interpretada por el programa ejecutor. El formato de este archivo (para variar) no tiene la forma de una base de datos relacional, sino que sus registros tienen un tamaño variable dependiendo del orden de los registros y el tipo y número de campos para cada linea del libreto. En la Tabla 9.1 se muestran los diferentes registros y sus campos.

Acción	Procedimiento	Parámetros	Bytes totales
1	Inserta_PCX	nombre de archivo	10
4	Inserta_Duende	número, archivo	11
5	Inserta_Anima	número, arch, Etiq, NoDnd, IniX, IniY	25
6	Inserta_Objeto	número, animación	3
7	Inserta_Musica	número, nombre de archivo, repeticiones	12
8	Inserta_Restricc	número, X1, Y1, X2, Y2, anm	11
9	Inserta_Uso	# Obj, # anm, Etiqueta, EX1, E2, BN1, B2, BN3	17

Tabla 9.1. Número de bytes necesarios para cada registro del archivo de libreto.

Todos los elementos necesarios para cada escena son cargados en memoria por el programa ejecutor y usados sólo cuando es necesario. Es por eso que aunque no se usen en un principio deben especificarse de cualquier manera que objetos y animaciones intervienen en cada escena.

9.1.5. Asistentes.

Los datos pueden capturarse directamente o se puede utilizar un *asistente*. Un asistente es un procedimiento que permite especificar datos y/o valores sin necesidad de capturarlos, por medio de una interface gráfica. Por ejemplo: En lugar de escribir el nombre de un archivo gráfico. Se elige el icono asistente gráfico, lo cual desplegará un listado de unidades/archivos/directorios para localizar un nombre existente. De esta manera, el usuario puede estar seguro que el nombre del archivo existe y se escribirá correctamente.

El icono de asistente (si está disponible), se ubica en la parte superior derecha de la pantalla, justo abajo del icono de eliminar registro. Este icono casi siempre tiene la misma figura que se elige cuando se inserta un nuevo registro.

 Asistente de imagen PCX.

Al elegir este icono aparece el listado de archivos con una máscara por defecto para archivos PCX. Una vez seleccionado el archivo, la imagen se despliega en la pantalla y para regresar al editor se presiona cualquier tecla o botón del ratón. La imagen se retendrá en memoria y se usará como pantalla de fondo para futuras operaciones con otros asistentes.

 Asistente de duende.

Cuando se usa este asistente aparece un listado de archivos por medio del cual se elige un archivo DND válido. Se despliega una copia del primer duende en forma de tapiz por toda la pantalla. Para regresar al editor se presiona cualquier tecla o botón del ratón.

 Asistente de animación.

Permite especificar el nombre y la posición de una animación/objeto. Cuando se utiliza, aparece un listado de archivos con una máscara *.ANM, se elige el nombre de la animación. Si se especificó un número de duende aparece otro listado de archivo, pero ahora con la extensión *.DND que se utiliza para confirmar el nombre del archivo de duendes. Después el 1er duende del conjunto se puede ubicar en la posición que le corresponde ayudado con un letrero que despliega sus coordenadas relativas en la pantalla (si no se eligió previamente leer una imagen, aparecerá un fondo en negro). Para elegir una posición y regresar al editor se presiona el botón 1 o la tecla [Enter]. Con esto las coordenadas son guardadas como la posición que tendrá la animación/objeto inicialmente. Para cancelar la operación y retener las coordenadas anteriores se presiona el botón 2 o la tecla [Esc].

Asistente de música.

Con este asistente se puede escuchar la tonada musical. Al usarse, aparece primero un listado de archivos con la máscara *.CMF, del cual se elige el nombre de la pieza que se tocará por defecto para esa escena. Después se manda a la tarjeta de sonido la música para que el usuario verifique que es el archivo correcto. Se detiene la reproducción con [Enter] o haciendo click con el ratón en "¿Detener música? [Si]".

Asistente de restricción

Permite especificar las coordenadas de un recuadro, el cual servirá de referencia para una restricción o para un mensaje. Cuando se elige este asistente aparece un pequeño recuadro el cual puede utilizarse de la siguiente manera

- Mover el recuadro. El recuadro sigue los movimientos del ratón. No debe apretarse ningún botón. Con el teclado se pueden utilizar las flechas del cursor [✱], [●], [✱] y [✱] para desplazar el recuadro.
- Cambiar de tamaño. Se coloca el recuadro de modo que coincida su ángulo superior derecho con el de la posición donde estará la restricción/mensaje. Se pulsa el botón izq. del ratón y se mueve éste hasta que coincidan las posiciones completamente. Se oprime el botón del centro para aceptar las coordenadas. Si se usa el teclado se oprimen la tecla [Home] = Reducir en X, [End] = Incrementar en X, [Pg Up] = Reducir en Y, [Pg Down] = Incrementar en Y. Se aceptan las coordenadas con [Enter].
- Cancelar. Para cancelar la operación se oprime el botón derecho o la tecla [Esc].

EJEMPLO:

Vamos a crear el libreto para la escena 1 (Cuarto de científico loco), del juego de esta tesis para ejemplificar el uso del editor de libretos[1].

- Primero debe insertarse una pantalla de fondo. Como no hay líneas estan disponibles todos los iconos de elementos. Se escoge insertar PCX (). Aparece el menú de archivos y escogemos MAYA1.PCX. La línea final es:

```
- Nombre -      { INSERTA PCX }
MAYA1
```

[1] El listado de todos los libretos y la explicación de la trama de la historia se encuentran en el capítulo 10.

- La música de fondo se incluye eligiendo Insertar elemento (), Insertar Música FM (), escribiendo el nombre del archivo y el número de repeticiones. Observese la linea siguiente:

```
No. -Nombre- Rep                    { Inserta música FM  }
  1,BH_MU8  ,99
```

- Ahora insertamos los duendes, y para cada uno se sigue la secuencia Insertar elemento () Insertar duende () y escoger el nombre del duende del menú de archivos. El primer duende siempre debe ser el protagonista, por lo que el nombre del primer duende es ITSMAEL.DND. Todos los duendes utilizados para la primera escenas son:

```
No. -Nombre-                        { Inserta duende     }
  1,ITSMAEL
  2,ENTORNO
  3,EJECUTOR
  4,PACO
  5,VENTILA
  6,MONCOMP
  7,TV
```

- Debemos incluir todas las restricciones de la pantalla, para esto se sigue la secuencia Insertar elemento (), Insertar restricción () y escribir los datos de la restricción. Como se mencionó en la descripción de los elementos, las restricciones que tienen un valor de cero en la columna de **No.** realmente son las coordenadas del un letrero. En el siguiente listado se encuentran todas las restricciones necesarias.

```
No.   X1    Y1    X2    Y2   Anm      { Inserta restricción }
  1,    9,    7, 310,   91,  0
  2,    0,    0,  11, 147,  0
  3, 306,     0, 319, 147,  0
  4, 126, 136, 200, 147,  0
  5,  68,  87, 109, 102,  0
  6, 104,  86, 186, 121,  0
  7, 236,  86, 258, 103,  0
  8, 259,  85, 296,  99,  0
  9,  12,  79,  57, 105,  0
 10, 175,  80, 241,  99,  0
  0, 181,  19, 206,  39,  1
  0, 261,  50, 293,  95,  2
```

- Como cada duende necesita una animación, procedemos a incluirlas con la secuencia Insertar elemento (), Insertar Animación (), y después capturar la información requerida para cada linea como se muestra a continuación.

```
No. -Nombre- Etiqueta  Dnd  X1   Y1 { Inserta animación }
```

```
 1,ITSMAEL ,QUIETO   ,  1, 231,  84
 2,ENTORNO ,F2       ,  2, 307, 169
 3,ENTORNO ,BOTON2   ,  2, 108, 169
 4,ENTORNO ,F1       ,  2, 307, 181
 5,MAYAOBJ ,REVISTA  ,  3, 122,  85
 6,MAYAOBJ ,BALON    ,  3, 183,  88
 7,MAYAOBJ ,CONTROL  ,  3, 152,  79
 8,MAYAOBJ ,SCANNER  ,  3, -30, -30
 9,MAYAOBJ ,LIBRO-1  ,  3, -10, -10
10,MAYAOBJ ,LIBROMAY,  3, -10, -10
11,MAYAOBJ ,CREDEBIB,  3, -10, -10
12,MAYAOBJ ,9        ,  3, -10, -10
13,MAYAOBJ ,10       ,  3, -10, -10
14,MAYAOBJ ,11       ,  3, -10, -10
15,MAYAOBJ ,12       ,  3, -10, -10
16,MAYAOBJ ,13       ,  3, -10, -10
17,MAYAOBJ ,14       ,  3, -10, -10
18,MAYAOBJ ,15       ,  3, -10, -10
19,MAYAOBJ ,16       ,  3, -10, -10
20,MAYAOBJ ,17       ,  3, -10, -10
21,PACO    ,PIENSA   ,  4,  18,  57
22,VENTILA ,OFF      ,  5, 267,  29
23,MONCOMP ,MON-COMP,  6, 206,  58
24,TV      ,OFF      ,  7,  80,  61
```

Debe recordarse que algunas animaciones se utilizan como objetos que puede utilizar el protagonista, asi que continuamos definiendo objetos por medio de la secuencia Insertar elemento (), Insertar Objeto (), y escribiendo el número de animación que será tratada como objeto en la columna **No.**. Los 2 objetos son activos inicialmente son:

```
No. Anm                                  { Inserta objeto     }
  6,   0
  7,   0
```

Finalmente deben definirse las lineas que muestran la interacción entre los objetos y las animaciones. Para esto se elige Insertar elemento (), Insertar Acción/Objeto () y se escriben los datos necesarios, como se muestran a continuación:

```
Obj Anm Etiqueta EX1 E2 BN1 B2 BN3  { Inserta Acción Obj.}
  4, 24,APAGADOR,  1, 1,  0, 0,  0
  2,  1,VERNOTI ,  4, 0,  0, 0,  0
  1, 22,APAGADOR,  0, 0,  0, 0,  0
  1, 21,PLATICA1,  2, 0,  0, 0,  0
  1, 21,PLATICA2,  2, 1,  0, 0,  0
  2, 21,PLATICA3,  3, 1,  0, 0,  0
  1, 21,PLATICA4,  3, 2,  0, 0,  0
  5, 21,PLATICA5,  3, 2,  0, 0,  0
```

9.2. PROGRAMA EJECUTOR

El programa ejecutor es el encargado de proporcionarle al usuario la interface final, para dar el efecto de un juego real. Para cada tipo de juego para computadora existe un diferente tipo de programa

ejecutor. Esto es debido a que son diferentes interfaces, tipos de movimientos, acciones, etc. dependiendo del efecto que se quiera obtener y la forma en que se interactua con el usuario. En un juego de aventura, es más importante poder reconocer objetos, almacenarlos y utilizarlos, también debe reconocer zonas y animaciones, esto permite dar la ilusión de estar recorriendo un mundo fantástico donde incluso se puede platicar con otros personajes. En un juego de acción, en cambio, el personaje debe poder realizar varios tipos de movimientos distintos y saber cuándo, en que condiciones y con qué otras animaciones "choca", por ejemplo cuando es herido por una bala o golpeado de una forma especifica.

Se podría decir, que para realizar diferentes tipos de juegos para computadora, casi siempre es suficiente con cambiar el programa ejecutor y, en algunos casos especificos, también el libreto

El programa ejecutor es el administrador de los recursos, por lo que el desempeño del juego depende de él. Como la mayor carga de programación (en cuanto a diseño), se centra en el programa ejecutor generalmente son los programadores más experimentados los encargados de su desarrollo.

9.2.1. Definición de programa ejecutor

EJECUTOR. Programa que interactua con el usuario, de modo que pueda utilizar todos los elementos definidos previamente por el libreto y el editor de duendes/animaciones.

9.2.2. Explicación del ejecutor

Especificamente el programa ejecutor debe poder realizar las siguiente tareas:

- Leer cada libreto en el momento adecuado.
- Armar cada escena del juego y mantener las animaciones funcionando.
- Controlar la interface usuario/protagonista.
- Dar mantenimiento al inventario de objetos.
- Actualizar y utilizar las banderas.
- Proporcionar un menú de opciones para el juego.

LEER CADA LIBRETO EN EL MOMENTO ADECUADO

En los juegos de aventuras se utiliza bastante el concepto de escena. Una *escena* es el lugar donde se desarrolla la acción. Puede ser, por ejemplo, un cuarto, un bosque, el espacio, etc. Para tener éxito en los juegos de aventura, debe alcanzarse un objetivo, y para ello deben explorarse todos los escenarios, obtener y usar objetos, platicar con diferentes personas, etc. Es tarea del ejecutor permitir hacer cambios de escena cuando se necesite y saber el nombre del libreto que será leido. Por otra parte es cuestión del diseñador de la historia hacer un cambio lógico de escenas, o sea, si se alcanza un primer objetivo (por ejemplo, encontrar una llave mágica) se puede avanzar a otro lugar (tal vez un cuarto escondido).

ARMAR CADA ESCENA DEL JUEGO Y MANTENER LAS ANIMACIONES FUNCIONANDO

Para dar la ilusión de que se están ejecutando varias acciones al mismo tiempo, deben dibujarse todas las animaciones en el lugar que les corresponde de forma alternada. Recordemos que cada animación está compuesta de una sucesión de imágenes acomodadas en cierto orden y que cambian en un lapso de tiempo determinado. Como cada animación debe tener sus propias pausas entre cada animación, el programa ejecutor coordina que vayan alternando de forma coherente, así como también que las animaciones no se traslapen unas con otras (lo que afectaría el efecto de tres dimensiones).

Cada pantalla se arma con un promedio de 30 imágenes por segundo (en una 486 SX a 25 Mhz), que es equivale al promedio que tiene una película de cine. Esto quiere decir que si se diseñaran animaciones más detalladas parecería que estamos viendo una película o caricatura, sin embargo, la cantidad de memoria requerida y el trabajo horas/diseño se incrementarían también. Existen juegos para computadora que tienen efectos de animación tipo televisor, pero debido a la gran cantidad de espacio ocupado necesitan estar grabados en discos compactos (CD ROM's) los cuales tiene una capacidad de hasta 640 Megabytes.

CONTROLAR LA INTERFACE USUARIO/PROTAGONISTA

En los juegos de aventura debe existir el protagonista. El *protagonista* es el personaje que representa al usuario y con el cual se intenta llegar al objetivo deseado. El ejecutor debe simular que el protagonista puede ver, platicar, tomar y usar objetos y explorar un área. Para esto debe tomar las ordenes del usuario de cualquier dispositivo periférico de entrada (teclado, ratón, palanca de control, etc.) e interpretarlas para saber que es lo que se desea que haga el personaje (caminar, agarrar, leer, usar, etc.). En las interfaces actuales de los juegos de aventura, es casi obligatorio la utilización del ratón, debido a su facilidad para moverse por toda la pantalla.

Existen varios tipos de interface:

- Armado de frases: Se escribe una frase que el ejecutor debe interpretar, por ejemplo: "Lee el letrero", "Toma la espada", "Pelea", etc. Este tipo de interfaces era el más popular, pero ya casi no se usa.
- Ejecución de acciones. En alguna parte de la pantalla existe un cierto número de acciones predefinidas en forma de botones o iconos, que pueden decir o representar "Leer", "Tomar", "Usar", etc. Actualmente es la forma más popular.
- Apunta y señala. También conocida por la frase en inglés "point and click". Esta interface es la más fácil de usar debido a que no se tiene que armar, escribir o usar frases/iconos, para realizar una acción. Basta simplemente con señalar una parte de la pantalla que nos llame la atención, ya sea con la flecha o con un objeto y esperar cual es el resultado. Por ejemplo, señalar a una persona con la flecha del ratón significa generalmente platicar con ella. Si señalamos un objeto, significa tomarlo, entonces la flecha del cursor cambia de forma y adopta la figura del objeto, con lo cual se usa el objeto con la pantalla. Este tipo de interface es la que se usará en el programa elaborado con esta tesis, debido a la facilidad con que el usuario la utiliza.

Dependiendo del diseñador del juego, se pueden agregar algunos detalles adicionales, por ejemplo, que cuando el cursor pase por un lugar se despliegue el nombre del objeto y se habilite una opción por defecto.

DAR MANTENIMIENTO AL INVENTARIO DE OBJETOS

La mecánica básica de los juegos de aventuras es obtener objetos y utilizarlos para llegar a una meta. Es por eso que el programa ejecutor debe mantener una lista de los objetos que el protagonista tiene en su inventario, y controlar altas y bajas. Aunque por regla general los objetos no tienen movimiento (casi siempre para ahorrar memoria), algunos objetos de ciertos juegos son animados.

El programa ejecutor debe proporcionar, también, una forma fácil de obtener los objetos del inventario, ya sea poniendo la lista siempre en la pantalla o haciendo que aparezcan al hacer click sobre un icono o al presionar una tecla.

ACTUALIZAR Y UTILIZAR LAS BANDERAS

Antes que nada tenemos que definir que entendemos por banderas. La bandera es un dato que nos indica un estado, dependiendo del cual se realiza una acción. Por ejemplo, si usamos un objeto en una animación se puede activar una bandera (cambio de estado), indicando que ya no se puede volver a utilizar ese objeto, pero se pueden realizar otras acciones nuevas.

El programa ejecutor en conjunción con el archivo proporcionado por el libreto ejecuta, permite o limita acciones o procesos. Podemos representar la función de las banderas de forma esquemática como se muestra a continuación en la Tabla 9.1:

OBJETO	*ANIMACION*	*BANDERA*	*ACCION*	*NUEVO ESTADO DE LAS BANDERAS*
Control	Televisión	B1 = 2	Se prende la TV	B1 = 3
Flecha	Cuidador	B2 = 1	Pide credencial	No hay cambios
Credencial	Cuidador	B2 = 1	Permite el paso	B2 = 2 B3 = 2
Credencial	Protagonista	B3 = 2	Decir "No salga sin ella"	B3 = 3

TABLA 9.1. Ejemplo para el manejo de banderas.

Como se observa en la tabla anterior existen 3 columnas que representan entradas (Objeto, animación, Bandera) y 2 que representan salidas (Acción, Nuevo estado de las banderas). La primera fila de la tabla significaría: "*Si* el Control se usa en la Televisión con la bandera B1=2 *entonces* se prende la TV y la bandera B1=3". Sólo puede haber una bandera de entrada pero se pueden modificar varias banderas a la salida.

Como el juego depende del valor de las banderas, es posible guardar el estado actual del juego guardando en un archivo las banderas, las posiciones actuales de las animaciones, los objetos obtenidos y tal vez algunos otros datos, de esta forma se necesitaría muy poco espacio en disco para guardar varios juegos. Sin embargo, no siempre es posible determinar las condiciones en las que está el juego tan fácilmente debido a que los procesos son independientes y las variables pueden se locales y no globales (una variable local no se puede modificar a menos que se pase el valor como parámetro, lo que significaria complicar demasiado el código). Por esta razón, hay programadores que determinan el inicio y final de sus variables dinámicas (apuntadores) y guardan *todo* el conjunto de memoria en disco. Aunque ésta es una forma más "fácil" y se conservan todas las posiciones y estados con exactitud, se requiere un mayor espacio en disco para salvar el juego. Por supuesto, se requiere una estructura de datos muy bien planeada para obtener todos los recursos de pocas fuentes (pocas variables dinámicas), y esa es la razón de las comillas en la palabra fácil escrita con anterioridad.

PROPORCIONAR UN MENU DE OPCIONES PARA EL JUEGO

Los juegos pueden tener efectos de sonido, música de fondo, frases de texto que desaparecen después de un cierto tiempo, etc. Pero a los usuarios en ocasiones les parece molesta o simplemente no les gusta que su computadora genere sonidos o les gustaria que el texto se tardará un poco más en desaparecer para alcanzar a leerlo (esto se aplica frecuentemente a los textos en otro idioma). Es por eso que los programadores proporcionan un menú especial de opciones para que el usuario ajuste el juego a su preferencia personal, que lo salve, que lea un nuevo juego o simplemente que se salga del programa.

Las opciones pueden ser las siguientes:

- Leer un juego salvado anteriormente.
- Salvar el juego actual.
- Salir del programa (Obviamente la única opción obligatoria).
- Volver a empezar el programa.
- Ajustar la música y/o efectos especiales (cambiar el volumen o desactivar la opción).
- Cambiar la velocidad de aparición del texto.
- Cambiar el nivel de detalle de la pantalla y las animaciones (esto es usado comúnmente para que el juego se vea más fluido en computadoras lentas).
- Cambiar el nivel de dificultad (para juegos de aventura que incluyen escenas de combate o de acción).

En el ejecutor que se incluye en esta tesis la única opción es salir del juego.

CAPITULO 10

DESCRIPCION DEL JUEGO DE VIDEO REALIZADO

CAPITULO 10
DESCRIPCION DEL JUEGO DE VIDEO REALIZADO

Una vez creadas todas la herramientas auxiliares y utilizando el software de apoyo procedemos a diseñar (¡por fin!) un juego para computadora. Este capitulo esta dedicado a describir los procedimientos que se requirieron para realizar el juego. Cabe aclarar que aunque el procedimiento general lleva un orden, algunos de los pasos se hicieron paralelamente (o sea, mientras un desarrollador del juego hacia la escena de la biblioteca el otro se dedicaba al centro de cómputo).

10.1. HISTORIA

Necesariamente se necesita alguna idea del tema que se desarrollará en el juego, podria ser del espacio, de caballeros y dragones, de fantasia, etc. Para este juego se desarrolló un tema que siempre es polémico: los ovnis, sólo que ahora mezclados con influencia de la historia prehispánica (especificamente los mayas) y nuestras vivencias personales.

Asi como sucede con los libros y el cine, la historia del juego debe adaptarse, de modo que no se pierda la trama y que el usuario pueda interactuar continuamente con el juego.

- Debe evitarse abusar de diálogos muy grandes.
- Ir avanzando por objetivos dependiendo en gran parte del hecho de obtener objeto para usarlos posteriormente en alguna parte.
- Tal como en las obras de teatro, se debe manejar la historia por escenas, haciendo una descripción del fondo, del ambiente, de los actores y objetos importantes.
- Y finalmente tener *mucha* imaginación, gran parte del éxito de cualquier historia (ya sea para cine, para televisión o para juegos de aventura) depende de su originalidad y la forma en como se desarrolla su trama. Además en un juego para computadora TODO es posible, no hay limitantes de presupuesto, actores temperamentales, ni siquiera de las leyes de la fisica del tiempo ni del espacio.

10.1.1. La trama

La historia de un juego descrita a grandes rasgos es la trama. Para este juego, la trama es la siguiente:

"Un estudiante de la E.N.E P. Aragón se entera del descubrimiento de un extraño objeto maya que tiene grabada la figura de lo que parece ser un ovni. Con ayuda de un amigo suyo se da a la tarea de intentar traducir los jeroglificos que hasta ese momento los científicos no habian podido descifrar. Gracias a sus conocimientos de computación van, poco a poco, resolviendo el misterio, sin imaginar las consecuencias sin precedentes que originará su inocente investigación. Existe información que es

resguardada a toda costa por muchos personajes, algunos de los cuales no son humanos. ¿Se podrá resolver este misterio por medio del **Proyecto Maya**?"

10.1.2. Descripción teatral

La descripción teatral, es la mejor forma de escribir una historia para un juego de aventuras, esto se debe a que debido a su forma de presentar la escena es mas fácil adaptarla al juego. Esta descripción no se le da al usuario, porque contiene la forma ideal de terminar el juego, y ese es un trabajo que debe realizar él mismo.

Para Proyecto Maya, se tiene la siguiente descripción teatral.

PROYECTO MAYA

ESCENA 1 (Cuarto científico loco).

Itsmael (El protagonista) se encuentra en una habitación típica de universitario, (del tipo de un científico loco), con 2 computadoras una de las cuales está funcionando sola, la otra es manejada por Paco (amigo de Itsmael). Se observa en la pared un cuadro pintado a mano que representa un atardecer (aunque se ve tan ambiguo que también pudiera representar un amanecer), varios posters y una ventana por la cual entra el sol. Como muebles hay un estante cerrado con llave, una cama (ya tendida), una mesa larga donde está la computadora sola, algunos libros y otros objetos sin importancia. Una mesa donde se encuentra una televisión de 20" a color apagada y un mueble para computadora donde está trabajando Paco. Hay algunos objetos regados por el cuarto como unas pesas, un periódico y un control remoto sobre la cama y un balón en el suelo.

>> Usar cursor/objetos.
1 - Flecha/cuadro: "Este cuadro parece una puesta de sol... ¿o es un amanecer? Por otro lado... Bueno, no importa."
2 - Flecha/gabinete: "Esto parece un refrigerador en lugar de un gabinete."

Itsmael aburrido se pone a platicar con Paco.

>> Platica: Itsmael / Paco.
3 - Its : "¿Qué es lo que estás haciendo?"
4 - Paco : "Quitándole la protección a un programa."
5 - Its : "Pero si ese programa tiene una clave codificada, ¿cómo adivinas cuál de todas las millones de combinaciones es la clave real?"
6 - Paco: "Acabo de hacer un programa capaz de descifrar cualquier tipo de clave secreta, incluyendo aquéllas que usen gráficas en vez de letras."

7 - Its: "¡Cualquier clave secreta! ¿Qué no tienes otra cosa que hacer? ¿Cómo le hiciste?."
8 - Paco: "Después de poner en práctica todos mis conocimientos de probabilidad, estadística, y análisis lógico, ... pues de suerte le atine. Sin embargo, podría tardarse mucho tiempo en encontrar la clave ya que se basa en la interpretación de imágenes y símbolos raros."
9 - Its: "Lo que hace uno por jugar..."

Itsmael intenta proseguir la platica

>> Plática: Itsmael/Paco
11 - Paco: "No te estoy corriendo, pero, no podrías ir a jugar con el supernintendo o a ver la tele... bueno, Si te estoy corriendo, déjame trabajar."

Itsmael en vista de que no tiene otra cosa mejor que hacer prende la televisión. En la televisión pasan la noticia del momento, en la cual se habla de un gran descubrimiento Maya, pero no dicen detalles ya que empieza un partido de fútbol. Recordando que acaba de comprar el periódico Itsmael se pone a leerlo.

>> Usar cursor/objetos
10 - Periódico/Its: "'INSOLITO DESCUBRIMIENTO' Un grupo de reconocidos arqueólogos encontraron una figura de origen Maya, que parece estar hecha de un material desconocido. La figura muestra de un lado unos extraños jeroglíficos cuyo contenido no se ha podido descifrar. Y del otro lado tiene el dibujo de lo que a simple vista parece una nave espacial."

Intrigado Itsmael le muestra el periódico a Paco.
>> Usar cursor/objetos
* Periódico/Paco:

>> Platica Its/Paco.
12 - Paco: "No estés molestando, no ves que si no le quito las claves no puedo seguir jugando."
13 - Its : "Pero... esto parece interesante. Además, no te estoy preguntando."
14 - Paco: "¿Qué es esto?"
15 - Its : "Parece que encontraron algo extraño. ¿Ya viste eso que parece una nave espacial?"
16 - Paco: "¿Y qué es lo que dice el jeroglífico?"
17 - Its : "Parece que todavía no lo han podido descifrar... pero a lo mejor tu si puedes con tu programa."
18 - Paco: "Pero no está tan avanzado. Aunque... Esta bien, necesito una manera de introducir la imagen en la computadora."
19 - Its : "Eso déjamelo a mi, tu sigue trabajando con tu programa para que pueda interpretar los jeroglíficos."
Itsmael intenta platica con Paco.

>> Platica: Itsmael/Paco.

20 - Paco: "¿Ya está lo que te encargue?... ¿No verdad? Apúrate, que es para hoy."

ESCENA 2 (Laboratorio de computación).

Departamento de analistas. Entra el jefe y se pone a trabajar en una computadora. Hay páneles de separación, una credencial, un scanner, un extinguidor y una computadora.

>> Plática: Itsmael/Jefe.

37 - Its: "Me gustaria ver la posibilidad de que me prestara el scanner, por que hay una imagen que quiero digitalizar.

38 - Jefe: "Claro que si, con la condición que me consigas en estos momentos un libro que necesito. Se llama 'Multimedia'"

Itsmael le pregunta al jefe.

>> Plática: Itsmael/Jefe.

39 - Its: "¿Me podria repetir el nombre del libro?"

40 - Jefe: "El libro se llama 'Multimedia'"

>> Usar cursor/objetos.

45 - Flecha/extinguidor: "En caso de incendio, o por si alguien piensa demasiado."

ESCENA 3 (Biblioteca)

Se alcanza a ver una parte de la biblioteca. Hay tres estantes llenos de libros y la barra para préstamo que por el momento está vacia. En la entrada se encuentra la encargada de seguridad (Estefy), pidiendo credenciales a todos los que entran.

Itsmael intenta entrar a buscar el libro que le encargaron.

>> Platica: Itsmael/Estefy

21 - Estefy: "Credencial por favor."

22 - Its: "¿Del trabajo? ... ¿Para votar con fotografía? ...

¿Del gimnasio? ..."

23 - Estefy: "Por el momento me conformo sólo con la de la biblioteca, con resello actual o algo que la haga válida."

24 - Its: "No podríamos llegar a un arreglo $$$ ..."

25 - Estefy: "¿Quieres morir violentamente?"

26 - Its: "No hay problema. Ahorita le traigo la credencial."

Itsmael regresa sin credencial.

>> Platica: Itsmael/Estefy
27 - Estefy: "¿Otra vez tú? ¿ya traes la credencial?"
28 - Its: "Err.. Creo que la deje en el otro traje de superheroe, la voy a buscar."

ESCENA 2 (Laboratorio de computación).

Itsmael le pide la credencial a su jefe.

>> Plática Its/Jefe.
41 - Its: "No puedo entrar a la biblioteca porque me hace falta una credencial. ¿Me podría prestar la suya?"
42 - Jefe: "Está sobre la mesa. Puedes tomarla."

ESCENA 3 (Biblioteca) Con la credencial.

Itsmael consigue la credencial de su jefe y se la muestra a Estefy.

>> Platica: Itsmael/Estefy
29 - Estefy: "Veamos, Umm. No es la firma del director, pero es suficiente, puedes pasar."

>> Usar cursor/objetos
31 - Flecha/Sección de libros 1: "'Derecho romano 25' ... 'Constitución política de los Estados Unidos Mexicanos' ... '1001 formas de probar la inocencia de alguien (sea culpable o no)' ... Toda esta sección es para la carrera de Derecho. ¿Por qué siento escalofrios?"
32 - Flecha/Sección de libros 2: "'Técnicas de enseñanza avanzadas" ... "Pedagogía aplicada 1' ... 'Enseñanza infantil (¡Peligro!, sólo para personas con los nervios de acero)'... Esta parte corresponde a la carrera de Pedagogía. Que curioso. Los libros de enseñanza infantil están casi nuevos."
33 - Flecha/Sección de libros 3: "'Mecánica de suelos' ... 'Puentes, presas y carreteras' ... 'Construya edificios económicos (incluye sección especial de amparos)' ... Definitivamente estos son libros para la carrera de Ingeniería civil. Aquí hay una nota que dice : 'Remember Tlatelolco'.
34 - Flecha/Sección de libros 4: "'Redes eléctricas y electromecánica' ... 'Ingeniería térmica' ... ¿'Playboy'? ... Por supuesto, estos anaqueles son para la carrera de Ingeniería Mecánica Eléctrica."
35 - Flecha/Sección de libros 5: "Por fin esta es la sección de la gloriosa carrera de Ingeniería en Computación ... Ejem,..., creo que me emocione un poquito ... Veamos ... Por aquí deben estar los libros que necesito."
Itsmael regresa a la biblioteca y le muestra la credencial de nuevo a Estefy:

>> Platica Itsmael/Estefy:

36 - Estefy: "Hola de nuevo. Pásale."

ESCENA 2 (Laboratorio de computación).

Itsmael le da el libro de historia antigua Maya a su jefe.

>> Usar cursor/objetos

* Libro de hist. ant. Maya/Jefe:
43 - Jefe: "Ese libro no es el que te pedí."

* Libro de multimedia/Jefe
44 - Jefe: "Perfecto. Ese es exactamente el libro que necesitaba. Puedes llevarte el scanner."

Itsmael toma el scanner de la mesa y regresa al cuarto tipo científico loco.

ESCENA 1. (Cuarto) De regreso con el scanner.

Itsmael le da el scanner a Paco.

>> Usar cursor/objetos
* Scanner/Paco.

>> Plática : Its - Paco.
46 - Its: "Aquí está el scanner, con esto puedes meter la imagen a la computadora."
47 - Paco: "Ok. Dejame conectarlo..."
48 - Paco: "err... no funciona... tal vez si lo golpeo aquí...muy bien ya está procesando."
49 - Paco: "esto es increible... no vas a creer lo que dice ..."

10.2. PANTALLAS DE FONDO

Las pantallas de fondo junto con los duendes son la base para armar una escena. Sobre la gráfica de fondo se insertarán las animaciones las cuales cambiarán un poco para dar la ilusión de movimiento. Estas pantallas no se modifican y son guardadas en alguna parte de la memoria (referenciadas por alguna variable global) para ser utilizadas constantemente.

No podiamos grabar un lugar (digamos una oficina) e incluirla simplemente como un escenario en el juego, notamos que se perdian las proporciones cuando el protagonista subia y bajaba. Esto se debia a que en las tomas inevitablemente habia una perspectiva (los objetos de enfrente eran ligeramente más grandes que los de atrás), esto junto con el hecho que el protagonista siempre era del mismo tamaño, daba un efecto de desproporción. Para solucionar este problema se tuvieron que diseñar las pantallas de fondo de forma manual, y se siguieron los siguientes pasos:

- Escoger una textura para el piso y para la pared, y dibujarlos de modo que ocupen el 40% y el 60% del espacio total de la pantalla respectivamente.
- Incluir ventanas, cortinas y puertas si es necesario.

- Escalar todos los objetos que deberían estar en el cuarto para que fueran proporcionales al tamaño del protagonista (esos objetos no se pueden tomar, son sólo de adorno).
- Recortar cada objeto de adorno e irlos colocando en la pantalla.

En la figura 10.1 se observa el proceso simplificado para crear los escenarios. Se agregan objetos a una pantalla de fondo para obtener la escena deseada. Los objetos que se pueden agarrar (por ejemplo el periódico y el balón) no están incluidos en la pantalla de fondo, sino que se convierten en duendes y reciben la categoría de objetos. Estos objetos aparecerán cuando se arme la escena final.

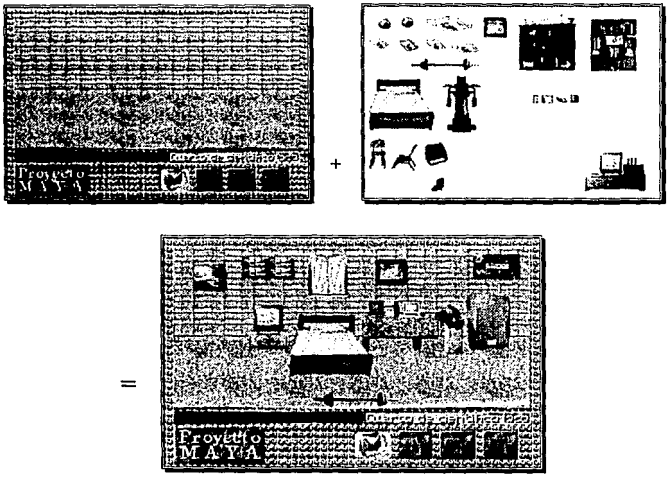


FIGURA 10.1. Secuencia para crear una pantalla de fondo.

Para esta tesis se crearon tres pantallas de fondo, una pantalla de menú, y tres adicionales (Figura 10.2):

- Cuarto de científico loco
- Laboratorio de cómputación
- Biblioteca

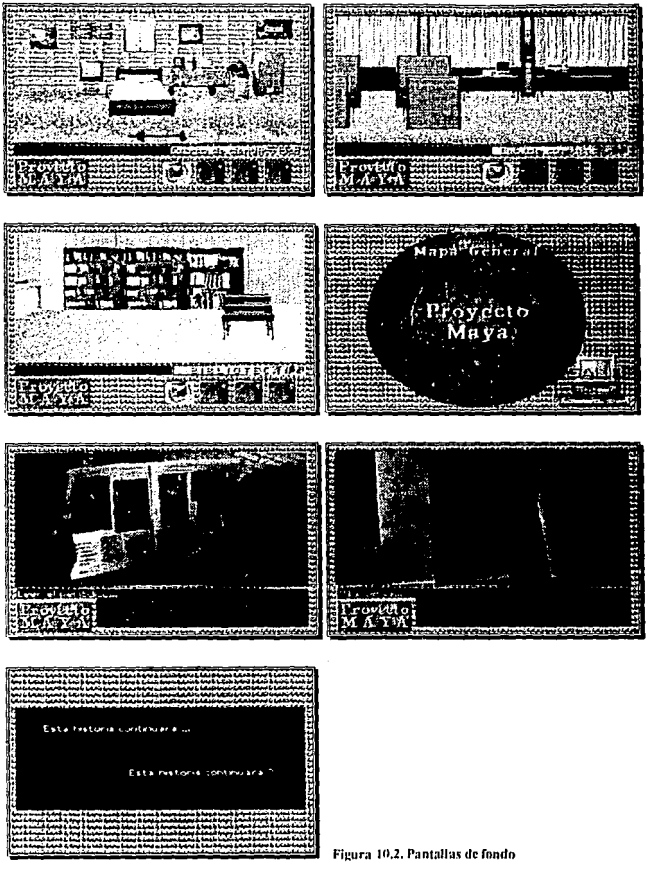


Figura 10.2. Pantallas de fondo

10.3. DUENDES

Un duende es una sucesión de imágenes del mismo personaje (digamos el protagonista), en las cuales cambian las posiciones o posturas para obtener todos los movimientos requeridos para hacer las animaciones. Había dos opciones para crear a los duendes: Dibujarlos a mano (que requiere que se digitalicen, se retoquen y se les de color), o filmar a modelos humanos en diferentes posiciones (filmarlos, grabar la sucesión deseada, y recortarlos), y se optó por la segunda opción, ya que ninguno de los dos programadores tenía facilidad (y velocidad) para el dibujo. En la figura 10.3 se muestra un ejemplo de la sucesión de imágenes que se requirió para hacer las posturas de frente de Itsmael y de Paco.

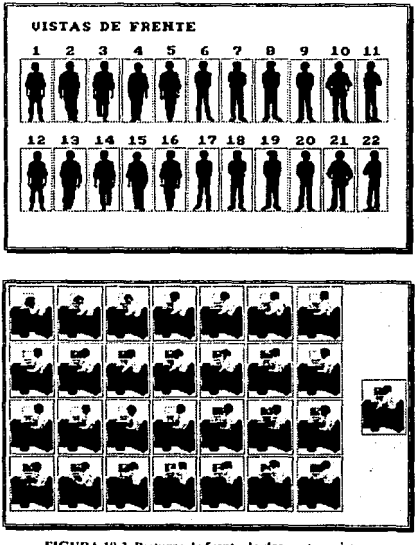


FIGURA 10.3. Posturas de frente de dos protagonistas.

A cada "cuadro" se le tenía que quitar el fondo normal que se grabó en la filmación y sustituirlo con el color de la máscara (para que se transparente la pantalla del fondo y el duende no parezca un cuadro).

Además de las posturas de la figura 10.2 se tenía que realizar el mismo proceso para las vistas laterales y posteriores.

El software de apoyo que se requirió fue:

- VidCap (Video for windows): Para capturar todas las secuencias de movimiento. El programa se ajustó con la paleta de fondo estándar para el juego, a un tamaño de 80x60 pixeles y con el formato para paleta de 8 bits. Los datos se guardaban en formato AVI.
- VidEdit (Video for windows): Para eliminar los cuadros innecesarios.
- Picture Taker (Story Board Live!): Para capturar las imágenes una por una.
- Picture Maker (Story Board Live!): Para quitar el fondo natural de la grabación y sustituirlo por el color de la mascara. La imagen se guardaba en formato PCX.

Las herramientas especializadas que se usaron fueron:

- Editor de duendes y animación: Para leer cada imagen de los archivos AVI y pudieran ser capturadas por Picture Maker. Para crear los duendes, una vez que las gráficas con las sucesiones de imágenes habían sido depuradas en Picture Maker.

10.4. ANIMACIONES

Aunque en el archivo de duendes están todos los movimientos necesarios de un personaje, es necesario escoger las imágenes adecuadas del duende para cada acción. Además, es necesario determinar que tanto se mueve cada animación en los ejes coordenados, o si va a activar alguna otra función. A continuación se describen las animaciones usadas y el efecto que causan.

ESTEFY.ANM

```
01 ATRAS     01 005    0   -3   Un paso hacia atras.
00           02 005    0   -3
00           03 005    0   -3
00           04 005    0   -3
00           05 005    0   -3
00           06 005    0   -3
02 FIN       07 005    0   -3
00 --------  00 000    0    0
01 FRENTE    08 005    0    3   Un paso hacia adelante.
00           09 005    0    3
00           10 005    0    3
00           11 005    0    3
00           12 005    0    3
00           13 005    0    3
00           14 005    0    3
02 FIN       15 005    0    3
00 --------  00 000    0    0
01 DERECHA   16 005    3    0   Un paso hacia la derecha.
00           17 005    3    0
00           18 005    3    0
00           19 005    3    0
00           20 005    3    0
```

```
00              21 005     3     0
00              22 005     3     0
02 FIN          23 005     3     0
00 --------     00 000     0     0
01 IZQUIERD     24 005    -3     0    Un paso hacia la izquierda.
00              25 005    -3     0
00              26 005    -3     0
00              27 005    -3     0
00              28 005    -3     0
00              29 005    -3     0
00              30 005    -3     0
02 FIN          31 005    -3     0
00 --------     00 000     0     0
01 VFRE         36 100     0     0    Secuencia de espera de frente.
00              38 100     0     0
00              37 100     0     0
00              35 100     0     0
00              37 150     0     0
00              38 025     0     0
00              39 025     0     0
00              40 025     0     0
00              41 025     0     0
00              40 100     0     0
00              39 025     0     0
00              38 025     0     0
00              37 025     0     0
00              36 150     0     0
00              35 025     0     0
00              34 025     0     0
00              33 025     0     0
00              32 025     0     0
00              33 100     0     0
00              34 025     0     0
02 FIN          35 150     0     0
00 --------     00 000     0     0
01 VIZQ         49 100     0     0    Secuencia de espera a la izquierda.
00              50 050     0     0
00              51 050     0     0
00              52 100     0     0
00              53 100     0     0
00              54 100     0     0
02 FIN          55 100     0     0
00 --------     00 000     0     0
01 VDER         42 100     0     0    Secuencia de espera a la derecha.
00              43 050     0     0
00              44 050     0     0
00              45 100     0     0
00              46 100     0     0
00              47 100     0     0
02 FIN          48 100     0     0
00 --------     00 000     0     0
01 VATRAS       01 000     0     0    Secuencia de espera hacia atras.
02 FIN          01 000     0     0
00 --------     00 000     0     0
01 ABAJO        08 000     0     0    Camina hacia el frente.
03 FRENTE       00 999     0     0
```

```
03 VFRE       00 900     0     0
02 FIN        00 000     0     0
00 --------   00 000     0     0
01 IZQ        24 000     0     0    Camina hacia la izquierda.
03 IZQUIERD   00 999     0     0
03 VIZQ       00 900     0     0
02 FIN        00 000     0     0
00 --------   00 000     0     0
01 DER        16 000     0     0    Camina hacia la derecha.
03 DERECHA    00 999     0     0
03 VDER       00 900     0     0
02 FIN        00 000     0     0
00 --------   00 000     0     0
01 ARRIBA     01 000     0     0    Camina hacia atras.
03 ATRAS      00 999     0     0
03 VATRAS     00 900     0     0
02 FIN        01 000     0     0
00 --------   00 000     0     0
01 QUIETO     08 000     0     0    Estado inicial.
03 VDER       00 900     0     0
02 FINAL      08 000     0     0
00 --------   00 000     0     0
01 PASO       42 000     0     0    *Its. le da la credencial a Estefy.
04 VIZQ       01 000     0     0    Its. voltea a la izq.
11 CURSOR     00 000     0     0
07 DES_LET    00 000     0     0
00            42 001     0     0
14 VENTANA    00 000   312   199    (recortamos la ventana de duendes)
12 DEFANIMA   22 000     0     0
12 DEFANIMA   09 002     0     0
12 DEFANIMA   10 002     0     0
08            29 000     0     0
06            29 100    12     1
03 DERECHA    00 004     0     0    Estefy camina a la derecha hasta
14 VENTANA    00 000   319   199     salir de la pantalla.
13 BANDERAS   05 002     0     0
13 BANDERAS   06 002     0     0
11 CURSOR     01 000     0     0
03 QUIETO     00 001     0     0
15 X_Y_ABSO   22 000   330     0
02 FINAL      00 000     0     0
00 --------   00 000     0     0
01 CREDE      42 000     0     0    Cursor/Estefy. Platica 1.
04 VIZQ       01 000     0     0    Itsmael platica con Estefy y
11 CURSOR     00 000     0     0       esta le pide una credencial.
07 DES_LET    00 000     0     0
00            42 001     0     0
08            21 000     0     0
06            21 045    12     1
08            22 000     0     0
06            22 100    10     1
08            23 000     0     0
06            23 150    12     1
08            24 000     0     0
06            24 065    10     1
08            25 000     0     0
```

```
06            25 055    12     1
08            26 000     0     0
06            26 100    10     1
11 CURSOR     01 000     0     0
13 BANDERAS   05 001     0     0
13 BANDERAS   06 001     0     0
13 BANDERAS   07 002     0     0
03 QUIETO     00 000     0     0
02 FINAL      00 000     0     0
00 --------   00 000     0     0
01 CREDE2     42 000     0     0   Cursor/Estefy. Platica 2.
04 VIZQ       01 000     0     0   Itsmael platica de nuevo con
11 CURSOR     00 000     0     0    estefy, y esta le vuelve a pedir
07 DES_LET    00 000     0     0    la credencial.
00            42 001     0     0
08            27 000     0     0
06            27 080    12     1
08            28 000     0     0
06            28 100    10     1
11 CURSOR     01 000     0     0
03 QUIETO     00 900     0     0
02 FINAL      00 000     0     0
00 --------   00 000     0     0
01 PASO2      42 000     0     0   Credencial/Estefy.
04 VIZQ       01 000     0     0   Itsmael le da la credencial a
11 CURSOR     00 000     0     0    Estefy, esta lo reconoce y lo deja
07 DES_LET    00 000     0     0    pasar.
00            42 000     0     0
08            36 000     0     0
06            36 060    12     1
12 DEFANIMA   22 000     0     0
12 DEFANIMA   09 002     0     0
12 DEFANIMA   10 002     0     0
14 VENTANA    00 000   312   199
03 DERECHA    00 004     0     0   Estefy sale de la escena.
14 VENTANA    00 000   319   199
11 CURSOR     01 000     0     0
03 QUIETO     00 001     0     0
02 FINAL      00 000     0     0
00 --------   00 000     0     0
01 SALTO      16 001     0     0   Manda la animacion fuera de pantalla.
15 X,Y  ABS   22 000   330     0
02 FINAL      16 001     0     0
```

ITSMAEL.ANM

```
01 IZQUIERD   36 002    -4     0   Un paso hacia la izquierda.
00            37 002    -4     0
00            38 002    -4     0
00            39 002    -4     0
00            40 002    -4     0
00            35 002    -4     0
```

```
00            41  002    -4     0
00            42  002    -4     0
00            43  002    -4     0
00            44  002    -4     0
02 FINAL      35  002    -4     0
00 --------   00  000     0     0
01 DERECHA    26  002     4     0    Un paso hacia la derecha.
00            27  002     4     0
00            28  002     4     0
00            29  002     4     0
00            30  002     4     0
00            25  002     4     0
00            31  002     4     0
00            32  002     4     0
00            33  002     4     0
00            34  002     4     0
02 FINAL      25  002     4     0
00 --------   00  000     0     0    Un paso hacia el frente.
01 FRENTE     10  004     0     2
00            09  004     0     2
00            08  004     0     2
02 FINAL      11  004     0     2
00 --------   00  000     0     0
01 ATRAS      13  004     0    -2    Un paso hacia atras.
00            14  004     0    -2
00            15  004     0    -2
02 FINAL      16  004     0    -2
00 --------   00  000     0     0
01 VIZQ       03  080     0     0    Secuencia de espera a la izquierda.
00            01  160     0     0
00            03  160     0     0
00            02  160     0     0
02 FINAL      03  080     0     0
00 --------   00  000     0     0
01 VDER       06  080     0     0    Secuencia de espera a la derecha.
00            04  160     0     0
00            06  160     0     0
00            05  160     0     0
02 FINAL      06  080     0     0
00 --------   00  000     0     0
01 VFRE       17  080     0     0    Secuencia de espera al frente.
00            18  160     0     0
00            17  160     0     0
00            19  160     0     0
00            20  200     0     0
02 FINAL      17  080     0     0
00 --------   00  000     0     0
01 VATRAS     12  080     0     0    Secuencia de espera hacia atras.
00            21  160     0     0
00            12  160     0     0
00            22  160     0     0
00            24  160     0     0
02 FINAL      12  080     0     0
00 --------   00  000     0     0
01 IZQ        35  000     0     0    Camina a la izquierda.
03 IZQUIERD   00  999     0     0
```

```
03 VIZQ      00 900     0    0
02 FINAL     35 000     0    0
00 --------  00 000     0    0
01 DER       25 000     0    0    Camina a la derecha.
03 DERECHA   00 999     0    0
03 VDER      00 900     0    0
02 FINAL     25 000     0    0
00 --------  00 000     0    0
01 ABAJO     07 000     0    0    Camina hacia abajo.
03 FRENTE    00 999     0    0
03 VFRE      00 900     0    0
02 FINAL     07 000     0    0
00 --------  00 000     0    0
01 ARRIBA    12 000     0    0    Camina hacia arriba.
03 ATRAS     00 999     0    0
03 VATRAS    00 900     0    0
02 FINAL     12 000     0    0
00 --------  00 000     0    0
01 QUIETO    17 001     0    0    Estado inicial.
03 VFRE      00 900     0    0
02 FINAL     17 001     0    0
00 --------  00 000     0    0
01 VERNOTI   17 001     0    0    Periodico/Itsmael.
11 CURSOR    00 000     0    0    Itsmael lee en el periodico la
00           17 001     0    0     noticia completa sobre el
09 NOTICIA1  00 000     0    0     descrubrimiento maya.
10 ESPERA    50 999     2    0
09 NOTICIA2  00 000     0    0
10 ESPERA    50 999     2    0
13 BANDERAS  03 001     0    0
13 BANDERAS  04 001     0    0
11 CURSOR    01 000     0    0
03 QUIETO    00 900     0    0
02 FINAL     17 000     0    0
```

JEFE.ANM

```
01 ATRAS     08 003     0   -3    Un paso hacia atras.
02 FINAL     09 003     0   -3
00 --------  00 000     0    0
01 FRENTE    01 003     0    3    Un paso hacia el frente.
00           02 003     0    3
00           03 003     0    3
00           04 003     0    3
00           05 003     0    3
02 FINAL     06 003     0    3
00 --------  00 000     0    0
01 DERECHA   20 002     3    0    Un paso hacia la derecha.
00           21 002     3    0
00           22 002     3    0
00           23 002     3    0
00           24 002     3    0
```

```
00            25  002     3     0
00            26  002     3     0
00            27  002     3     0
00            28  002     3     0
02 FINAL      29  002     3     0
00 --------   00  000     0     0
01 IZQUIERD   10  002    -3     0    Un paso hacia la izquierda.
00            11  002    -3     0
00            12  002    -3     0
00            13  002    -3     0
00            14  002    -3     0
00            15  002    -3     0
00            16  002    -3     0
00            17  002    -3     0
00            18  002    -3     0
02 FINAL      19  002    -3     0
00 --------   00  000     0     0
01 VFRE       01  100     0     0    Secuencia de espera hacia el frente.
02 FINAL      01  150     0     0
00 --------   00  000     0     0
01 VIZQ       10  100     0     0    Secuencia de espera a la izquierda.
02 FINAL      10  100     0     0
00 --------   00  000     0     0
01 VDER       20  100     0     0    Secuencia de espera a la derecha.
02 FINAL      20  100     0     0
00 --------   00  000     0     0
01 VATRAS     07  000     0     0    Secuencia de espera hacia atras.
02 FINAL      07  000     0     0
00 --------   00  000     0     0
01 ABAJO      08  000     0     0    Camina hacia abajo.
03 FRENTE     00  999     0     0
03 VFRE       00  900     0     0
02 FIN        00  000     0     0
00 --------   00  000     0     0
01 IZQ        24  000     0     0    Camina hacia la izquierda.
03 IZQUIERD   00  999     0     0
03 VIZQ       00  900     0     0
02 FIN        00  000     0     0
00 --------   00  000     0     0
01 DER        16  000     0     0    Camina hacia la derecha.
03 DERECHA    00  999     0     0
03 VDER       00  900     0     0
02 FIN        00  000     0     0
00 --------   00  000     0     0
01 ARRIBA     01  000     0     0    Camina hacia arriba.
03 ATRAS      00  999     0     0
03 VATRAS     00  900     0     0
02 FIN        01  000     0     0
00 --------   00  000     0     0
01 QUIETO     01  000     0     0    Estado inicial.
03 VDER       00  900     0     0
02 FINAL      01  000     0     0
00 --------   00  000     0     0
01 ENTRADA    29  001     0     0    El jefe entra al laboratorio de
11 CURSOR     00  000     0     0     computación y empieza a trabajar
14 VENTANA    08  000   319   199     en una computadora.
```

```
03 DERECHA    00 007    0    0
03 ATRAS      00 006    0    0
11 CURSOR     01 000    0    0
14 VENTANA    00 000  319  199
03 ESCRIBE    00 900    0    0
02 FINAL      09 001    0    0
00 --------   00 000    0    0
01 ESCRIBE    30 007    0    0   Escribe en la computadora.
02 FINAL      31 007    0    0
00 --------   00 000    0    0
01 PLATICA1   04 001    0    0   Itsmael le pide el scanner al jefe
07 DES_LET    00 000    0    0    y este le pide un libro de
11 CURSOR     00 000    0    0    de "multimedia"
04 VATRAS     01 900    0    0
00            04 001    0    0
08            37 000    0    0
06            37 150   10    1
08            38 000    0    0
06            38 165   14    1
16 ESCENA     03 000    0    0
11 CURSOR     01 000    0    0
13 BANDERAS   07 001    0    0
03 ESCRIBE    00 900    0    0
02 FINAL      00 000    0    0
00 --------   00 000    0    0
01 PLATICA2   04 001    0    0   Itsmael le pide de nuevo el nombre
07 DES_LET    00 000    0    0    del libro al jefe.
11 CURSOR     00 000    0    0
04 VATRAS     01 900    0    0
00            04 001    0    0
08            39 000    0    0
06            39 075   10    1
08            40 000    0    0
06            40 075   14    1
11 CURSOR     01 000    0    0
03 ESCRIBE    00 900    0    0
02 FINAL      00 000    0    0
00 --------   00 000    0    0
01 PLATICA3   04 001    0    0   Itsmael le pide una credencial de
07 DES_LET    00 000    0    0    la biblioteca al jefe.
11 CURSOR     00 000    0    0
04 VATRAS     01 900    0    0
00            04 001    0    0
08            41 000    0    0
06            41 145   10    1
08            42 000    0    0
06            42 075   14    1
13 BANDERAS   07 003    0    0
12 DEFANIMA   11 002    0    0
11 CURSOR     01 000    0    0
03 ESCRIBE    00 900    0    0
02 FINAL      04 001    0    0
00 --------   00 000    0    0
01 PLATICA4   04 001    0    0   Itsmael le entrega el libro
07 DES_LET    00 000    0    0    equivocado al jefe.
11 CURSOR     00 000    0    0
```

```
04 VATRAS    01 900    0    0
00           04 001    0    0
08           43 000    0    0
06           43 070   14    1
11 CURSOR    01 000    0    0
03 ESCRIBE   00 900    0    0
02 FINAL     04 001    0    0
00 --------  00 000    0    0
01 PLATICA5  04 001    0    0    Itsmael le da el libro correcto
07 DES_LET   00 000    0    0     al jefe y este le permite tomar
11 CURSOR    00 000    0    0     el scanner.
04 VATRAS    01 900    0    0
00           04 001    0    0
08           44 000    0    0
06           44 150   14    1
13 BANDERAS  07 004    0    0
12 DEFANIMA  08 002    0    0
17 ELIMINAO  09 000    0    0    Elimina el libro de multimedia de
11 CURSOR    01 000    0    0     la lista de objetos.
03 ESCRIBE   00 900    0    0
02 FINAL     04 001    0    0
```

MONCOMP.ANM

```
01 PANT-COM  01 005    0    0    Secuencia de animación de la
00           02 005    0    0     computadora que se encuentra
00           03 005    0    0     trabajando sola en el cuarto
00           04 005    0    0     tipo científico loco.
00           05 005    0    0
00           06 005    0    0
00           07 005    0    0
00           08 005    0    0
00           09 005    0    0
00           10 005    0    0
00           11 005    0    0
00           12 005    0    0
00           13 005    0    0
00           14 005    0    0
00           15 005    0    0
00           16 005    0    0
00           17 005    0    0
00           18 005    0    0
00           19 005    0    0
02 FIN       20 005    0    0
00 --------  00 000    0    0
01 MON-COMP  01 000    0    0    Estado inicial.
03 PANT-COM  00 900    0    0
02 FIN       01 000    0    0
```

PACO.ANM

```
01 EXPLICA   22 010    0    0    Secuencia de animación en la que
```

```
00            23 010     0    0     Paco explica algo.
00            24 010     0    0
00            25 010     0    0
00            24 010     0    0
00            25 010     0    0
00            26 010     0    0
00            25 010     0    0
00            27 010     0    0
00            28 010     0    0
00            26 020     0    0
00            25 010     0    0
00            24 010     0    0
00            23 010     0    0
00            15 010     0    0
00            16 020     0    0
00            17 010     0    0
00            18 010     0    0
00            19 010     0    0
00            20 010     0    0
00            21 010     0    0
00            16 020     0    0
00            20 010     0    0
00            17 010     0    0
02 FIN        18 020     0    0
00 --------   00 000     0    0
01 REPOSO     08 015     0    0     Secuencia de animación en la
00            07 015     0    0      que Paco deja de trabajar.
00            06 015     0    0
00            05 015     0    0
00            04 015     0    0
00            05 015     0    0
00            06 015     0    0
00            07 015     0    0
00            08 015     0    0
00            09 015     0    0
00            10 015     0    0
00            11 015     0    0
00            12 015     0    0
00            11 015     0    0
00            10 015     0    0
00            11 015     0    0
00            12 015     0    0
00            11 015     0    0
00            10 015     0    0
00            11 015     0    0
00            12 015     0    0
00            13 015     0    0
00            14 015     0    0
00            29 015     0    0
00            31 015     0    0
00            30 015     0    0
00            29 015     0    0
00            30 015     0    0
00            31 015     0    0
02 FIN        18 015     0    0
00 --------   00 000     0    0
```

```
01 TRABAJA   18 000     0     0    Secuencia de animación donde Paco
03 EXPLICA   00 010     0     0     trabaja.
02 FIN       18 000     0     0
00 --------  00 000     0     0
01 DUERME    32 050     0     0    Secuencia de animación donde Paco se
00           33 100     0     0     queda dormido.
00           34 200     0     0
00           01 200     0     0
00           02 020     0     0
00           03 010     0     0
00           04 010     0     0
02 FIN       18 010     0     0
00 --------  00 000     0     0
01 PIENSA1   18 000     0     0    Secuencia de animación donde Paco
03 REPOSO    00 002     0     0     deja de trabajar.
03 DUERME    00 001     0     0
02 FIN       18 000     0     0
00 --------  00 000     0     0
01 PIENSA    18 000     0     0    Repetición infinita de la secuencia
03 PIENSA1   00 900     0     0     PIENSA1.
02 FIN       18 000     0     0
00 --------  00 000     0     0
01 PLATICA1  32 001     0     0    Itsmael platica con Paco por
11 CURSOR    00 000     0     0     primera vez.
07 DES_LET   00 000     0     0
01           32 001     0     0
08           03 000     0     0
06           03 060    10     1
08           04 000     0     0
06           04 070     4     1
08           05 000     0     0
06           05 200    10     1
08           06 000     0     0
06           06 190     4     1
08           07 000     0     0
06           07 120    10     1
08           08 000     0     0
06           08 350     4     1
08           09 000     0     0
06           09 060    10     1
11 CURSOR    01 000     0     0
13 BANDERAS  02 001     0     0
13 BANDERAS  01 001     0     0
03 PIENSA    18 000     0     0
02 FINAL     00 000     0     0
00 --------  00 000     0     0
01 PLATICA2  32 001     0     0    Paco le dice a Itsmael que está
11 CURSOR    00 000     0     0     muy ocupado.
07 DES_LET   00 000     0     0
00           18 001     0     0
08           11 000     0     0
06           11 200     4     1
11 CURSOR    01 000     0     0
03 PIENSA    00 001     0     0
02 FINAL     00 000     0     0
00 --------  00 000     0     0
```

```
01 PLATICA3  32 001   0   0   Periódico/Paco.
11 CURSOR    00 000   0   0   Itsmael le enseña el periódico a
07 DES_LET   00 000   0   0    Paco en donde se encuentra la
00           32 001   0   0    noticia del descubrimiento Maya,
08           12 000   0   0    y este le pide un scanner.
06           12 100   4   1
08           13 000   0   0
06           13 150  10   1
08           14 000   0   0
06           14 050   4   1
08           15 000   0   0
06           15 120  10   1
08           16 000   0   0
06           16 070   4   1
08           17 000   0   0
06           17 150  10   1
08           18 000   0   0
06           18 150   4   1
08           19 000   0   0
06           19 150  10   1
11 CURSOR    01 000   0   0
13 BANDERAS  03 002   0   0
13 BANDERAS  02 002   0   0
16 ESCENA    02 000   0   0
17 ELIMINAO  05 000   0   0
03 PIENSA    00 001   0   0
02 FINAL     33 050   0   0
00 --------  00 000   0   0
01 PLATICA4  32 001   0   0   Paco le recuerda a Itsmael el
11 CURSOR    00 000   0   0   encargo.
07 DES_LET   00 000   0   0
00           18 001   0   0
08           20 000   0   0
06           20 110   4   1
11 CURSOR    01 000   0   0
03 PIENSA    00 001   0   0
02 FINAL     00 000   0   0
00 --------  00 000   0   0
01 PLATICA5  32 001   0   0   Scanner/Paco.
07 DES_LET   00 000   0   0    Itsmael le da el scanner a Paco,
11 CURSOR    00 000   0   0    y este descrifa el contenido del
00           32 001   0   0    jeroglífico.
08           46 000   0   0
06           46 100  10   1   Esta es la secuencia final del juego.
08           47 000   0   0
06           47 060   4   1
08           48 000   0   0
06           48 160   4   1
08           49 000   0   0
06           49 170   4   1
13 BANDERAS  03 003   0   0
11 CURSOR    01 000   0   0
09 CONTINUA  00 000   0   0
10 ESPERA    50 999   2   0
99 FINJUEGO  00 000   0   0   Termina el juego y restaura las
03 PIENSA    00 001   0   0    condiciones iniciales.
```

```
02 FINAL     32 001     0     0
```

TV.ANM

```
01 ANUNCIO   01 020     0     0    Secuencia donde se anuncia la
00           02 020     0     0     noticia del descubrimiento Maya.
00           03 020     0     0
00           04 020     0     0
00           05 020     0     0
00           06 020     0     0
00           07 020     0     0
00           08 020     0     0
02 FIN       09 020     0     0
00 --------  00 000     0     0
01 NIEVE     10 002     0     0    Secuencia que pone ruido blanco o
00           11 002     0     0     "nieve" en la pantalla.
00           12 002     0     0
00           13 002     0     0
02 FIN       14 002     0     0
00 --------  00 000     0     0
01 FUT1      15 003     0     0    Secuencia que pone un fragmento de
00           16 003     0     0     partido de futbol.(Tribunas)
00           17 003     0     0
00           18 003     0     0
00           19 003     0     0
00           20 003     0     0
00           21 003     0     0
00           23 003     0     0
00           24 003     0     0
00           25 003     0     0
00           26 003     0     0
00           27 003     0     0
00           28 003     0     0
02 FIN       29 003     0     0
00 --------  00 000     0     0
01 FUT2      30 005     0     0    Secuencia que pone un fragmento de
00           31 005     0     0     un juego de futbol. (Jugadores y
00           32 005     0     0     cancha).
00           33 005     0     0
00           34 005     0     0
00           35 005     0     0
00           36 005     0     0
00           37 005     0     0
00           38 005     0     0
02 FIN       39 005     0     0
00 --------  00 000     0     0
01 FUTBOL    15 000     0     0    Secuencia que combina las secuencias
03 FUT1      00 003     0     0     FUT1 y FUT2.
03 FUT2      00 003     0     0
02 FIN       39 000     0     0
00 --------  00 000     0     0
01 TV1       01 000     0     0    Secuencia que muestra toda la
03 ANUNCIO   00 002     0     0     "programación" de la T.V.
03 NIEVE     00 005     0     0
```

```
03 FUTBOL    00 005    0    0
03 NIEVE     00 005    0    0
02 FIN       01 000    0    0
00 --------  00 000    0    0
01 TV        01 000    0    0    Repite indefinidamente la
03 TV1       00 900    0    0     secuencia TV1.
02 FIN       01 000    0    0
00 --------  00 000    0    0
01 OFF       40 900    0    0    Control/TV.
01 ON        01 001    0    0    Apaga/enciende la TV.
12 DEFANIMA  05 002    0    0
03 TV        00 900    0    0
```

VENTILA.ANM

```
01 AIRE      01 001    0    0    Secuencia de movimiento del
00           02 001    0    0     ventilador.
00           03 001    0    0
00           04 001    0    0
00           05 001    0    0
00           06 001    0    0
02 FIN       07 001    0    0
00 --------  00 000    0    0
01 VENTILAR  01 000    0    0    Repite indefinidamente la secuencia
03 AIRE      00 900    0    0     AIRE.
02 FIN       00 000    0    0
00 --------  00 000    0    0
01 ON        01 001    0    0    Apaga/Enciende el ventilador.
03 AIRE      00 900    0    0
01 OFF       01 900    0    0
00 --------  00 000    0    0
```

10.5. SONIDOS O EFECTOS ESPECIALES

El sonido de pasos, el click del televisor, el rechinido de una puerta, el ulular del viento, pueden darle una nueva dimensión de realismo a cualquier juego. Todos los sonidos del juego son cargados inicialmente en la memoria extendida por el programa ejecutor y son llamados cuando es necesario. El archivo donde se encuentran todos los sonidos se llama FINAL.VOC y se encuentra en el subdirectorio de trabajo ..\SB.

Para este juego se grabaron las voces de los cuatro protagonistas (Itsmael, Paco, Estefana y Victor{jefe}), para todos los diálogos. Las voces se grabaron a 8 Mhz y monoaurales. El archivo FINAL.VOC ocupó 1.3 Mbytes de espacio en disco.

10.6. LIBRETOS

Con los libretos definimos cuales elementos intervienen en cada escena, y como interactuan los objetos con las animaciones. En este juegos se utilizaron tres libretos distintos, que representaban las

tres escenas de las que consta el juego. El listado de cada libreto y su explicación se muestra a continuación.

MAYA1.LIB (Cuarto de científico loco)

```
 - Nombre -      [ INSERTA PCX ]
MAYA1


No. -Nombre-                              { Inserta duende     }
  1,ITSMAEL
No. -Nombre- Etiqueta  Dnd  X1   Y1 { Inserta animación }
  1,ITSMAEL ,QUIETO   ,  1, 231,  84
  2,ENTORNO ,F2       ,  2, 307, 169
  3,ENTORNO ,BOTON2   ,  2, 108, 169
No. -Nombre-                              { Inserta duende     }
  2,ENTORNO
No. -Nombre- Etiqueta  Dnd  X1   Y1 { Inserta animación }
  4,ENTORNO ,F1       ,  2, 307, 181
No.  X1   Y1   X2    Y2   Anm          { Inserta restricción }
  1,   9,   7, 310,  91,  0
  2,   0,   0,  11, 147,  0
  3, 306,   0, 319, 147,  0
No. -Nombre-                              { Inserta duende     }
  3,EJECUTOR
No. -Nombre- Etiqueta  Dnd  X1   Y1 { Inserta animación }
  5,MAYAOBJ ,REVISTA ,  3, 122,  85
  6,MAYAOBJ ,BALON   ,  3, 183,  88
  7,MAYAOBJ ,CONTROL ,  3, 152,  79
  8,MAYAOBJ ,SCANNER ,  3, -30, -30
  9,MAYAOBJ ,LIBRO-1 ,  3, -10, -10
 10,MAYAOBJ ,LIBROMAY,  3, -10, -10
 11,MAYAOBJ ,CREDEBIB,  3, -10, -10
 12,MAYAOBJ ,9       ,  3, -10, -10
 13,MAYAOBJ ,10      ,  3, -10, -10
 14,MAYAOBJ ,11      ,  3, -10, -10
 15,MAYAOBJ ,12      ,  3, -10, -10
 16,MAYAOBJ ,13      ,  3, -10, -10
 17,MAYAOBJ ,14      ,  3, -10, -10
 18,MAYAOBJ ,15      ,  3, -10, -10
 19,MAYAOBJ ,16      ,  3, -10, -10
 20,MAYAOBJ ,17      ,  3, -10, -10
No. Anm                                   { Inserta objeto     }
  6,  0
  7,  0
No. -Nombre-                              { Inserta duende     }
  4,PACO
  5,VENTILA
  6,MONCOMP
  7,TV
No. -Nombre- Etiqueta  Dnd  X1   Y1 { Inserta animación }
 21,PACO    ,PIENSA  ,  4,  18,  57
 22,VENTILA ,OFF     ,  5, 267,  29
```

```
 23,MONCOMP ,MON-COMP,  6, 206,  58
 24,TV      ,OFF     ,  7,  80,  61
No. -Nombre- Rep                    { Inserta música FM  }
  1,BH_MU8   ,99
No.  X1   Y1   X2   Y2   Anm        { Inserta restricción }
  4, 126, 136, 200, 147,  0
  5,  68,  87, 109, 102,  0
  6, 104,  86, 186, 121,  0
  7, 236,  86, 258, 103,  0
  8, 259,  85, 296,  99,  0
  9,  12,  79,  57, 105,  0
 10, 175,  80, 241,  99,  0
  0, 181,  19, 206,  39,  1
  0, 261,  50, 293,  95,  2
Obj Anm Etiqueta EX1 E2 BN1 B2 BN3  { Inserta Acción Obj.}
  4, 24,APAGADOR,  1, 1,  0, 0,  0
  2,  1,VERNOTI ,  4, 0,  0, 0,  0
  1, 22,APAGADOR,  0, 0,  0, 0,  0
  1, 21,PLATICA1,  2, 0,  0, 0,  0
  1, 21,PLATICA2,  2, 1,  0, 0,  0
  2, 21,PLATICA3,  3, 1,  0, 0,  0
  1, 21,PLATICA4,  3, 2,  0, 0,  0
  5, 21,PLATICA5,  3, 2,  0, 0,  0
```

MAYA2.LIB (Laboratorio de computación)

```
 - Nombre -     { INSERTA PCX }
MAYA2
No. -Nombre-                        { Inserta duende     }
  1,ITSMAEL
No. -Nombre- Etiqueta  Dnd  X1   Y1 { Inserta animación }
  1,ITSMAEL ,QUIETO  ,  1, 231,  84
  2,ENTORNO ,F2      ,  2, 307, 169
  3,ENTORNO ,BOTON2  ,  2, 108, 169
No. -Nombre-                        { Inserta duende     }
  2,ENTORNO
No. -Nombre- Etiqueta  Dnd  X1   Y1 { Inserta animación }
  4,ENTORNO ,F1      ,  2, 307, 181
No.  X1   Y1   X2   Y2   Anm        { Inserta restricción }
  1,   9,   7, 310,  91,  0
  2,   0,   0,  11, 147,  0
  3, 306,   0, 319, 147,  0
No. -Nombre-                        { Inserta duende     }
  3,EJECUTOR
No. -Nombre- Etiqueta  Dnd  X1   Y1 { Inserta animación }
  5,MAYAOBJ ,REVISTA ,  3, -30, -30
  6,MAYAOBJ ,BALON   ,  3, -30, -30
  7,MAYAOBJ ,CONTROL ,  3, -30, -30
  8,MAYAOBJ ,SCANNER ,  3, 261,  65
  9,MAYAOBJ ,LIBRO-1 ,  3, -10, -10
 10,MAYAOBJ ,LIBROMAY,  3, -10, -10
 11,MAYAOBJ ,CREDEBIB,  3, 150,  70
```

```
 12,MAYAOBJ ,9          ,  3, -10, -10
 13,MAYAOBJ ,10         ,  3, -10, -10
 14,MAYAOBJ ,11         ,  3, -10, -10
 15,MAYAOBJ ,12         ,  3, -10, -10
 16,MAYAOBJ ,13         ,  3, -10, -10
 17,MAYAOBJ ,14         ,  3, -10, -10
 18,MAYAOBJ ,15         ,  3, -10, -10
 19,MAYAOBJ ,16         ,  3, -10, -10
 20,MAYAOBJ ,17         ,  3, -10, -10
No. -Nombre-                          { Inserta duende     }
  4,JEFE
No. -Nombre- Etiqueta  Dnd  X1   Y1 { Inserta animación }
 21,JEFE     ,ENTRADA ,  4, -40,  87
No.  X1   Y1   X2   Y2   Anm          { Inserta restricción }
  4,  71,  87, 150, 132,  0
  5,   8,  91,  34, 131,  0
  6, 141,  88, 298, 101,  0
  7, 167,  94, 199, 109,  0
  0, 204,  38, 214,  58, 45
No. -Nombre- Rep                      { Inserta música FM   }
  1,BH_MU3   ,99
Obj Anm Etiqueta EX1 E2 BN1 B2 BN3  { Inserta Acción Obj.}
  1, 21,PLATICA1,  7, 0,  0, 0,  0
  1, 21,PLATICA2,  7, 1,  0, 0,  0
  1, 21,PLATICA3,  7, 2,  0, 0,  0
  7, 21,PLATICA4,  7, 3,  0, 0,  0
  6, 21,PLATICA5,  7, 3,  0, 0,  0
```

MAYA3.LIB (Biblioteca)

```
 - Nombre -     { INSERTA PCX }
MAYA3
No. -Nombre- Rep                      { Inserta música FM   }
  1,REGGAE   ,99
No. -Nombre-                          { Inserta duende     }
  1,ITSMAEL
No. -Nombre- Etiqueta  Dnd  X1   Y1 { Inserta animación }
  1,ITSMAEL ,QUIETO   ,  1, 284,  70
  2,ENTORNO ,F2       ,  2, 307, 169
  3,ENTORNO ,BOTON2   ,  2, 108, 169
No. -Nombre-                          { Inserta duende     }
  2,ENTORNO
No. -Nombre- Etiqueta  Dnd  X1   Y1 { Inserta animación }
  4,ENTORNO ,F1       ,  2, 307, 181
No.  X1   Y1   X2   Y2   Anm          { Inserta restricción }
  1,   9,   7, 310,  91,  0
  2,   0,   0,  11, 147,  0
  3, 306,   0, 319, 147,  0
No. -Nombre-                          { Inserta duende     }
  3,EJECUTOR
No. -Nombre- Etiqueta  Dnd  X1   Y1 { Inserta animación }
  5,MAYAOBJ ,REVISTA  ,  3, -30, -30
```

```
  6,MAYAOBJ ,BALON    ,  3, -30, -30
  7,MAYAOBJ ,CONTROL  ,  3, -30, -30
  8,MAYAOBJ ,SCANNER  ,  3, -30, -30
  9,MAYAOBJ ,LIBRO-1  ,  3, 137,  40
 10,MAYAOBJ ,LIBROMAY ,  3, 136,  68
 11,MAYAOBJ ,CREDEBIB ,  3, -10, -10
 12,MAYAOBJ ,9        ,  3, -10, -10
 13,MAYAOBJ ,10       ,  3, -10, -10
 14,MAYAOBJ ,11       ,  3, -10, -10
 15,MAYAOBJ ,12       ,  3, -10, -10
 16,MAYAOBJ ,13       ,  3, -10, -10
 17,MAYAOBJ ,14       ,  3, -10, -10
 18,MAYAOBJ ,15       ,  3, -10, -10
 19,MAYAOBJ ,16       ,  3, -10, -10
 20,MAYAOBJ ,17       ,  3, -10, -10
No. -Nombre-                          { Inserta duende     }
  4,BIBLIO
No. -Nombre- Etiqueta  Dnd  X1   Y1 { Inserta animación }
 21,BIBLIO  ,MESA     ,  4, 219, 118
No. -Nombre-                          { Inserta duende     }
  5,STEFY
No. -Nombre- Etiqueta  Dnd  X1   Y1 { Inserta animación }
 22,STEFY   ,QUIETO   ,  5, 239,  84
No.  X1   Y1   X2   Y2   Anm          { Inserta restricción }
  4,  10,  10,   0,   0, 22
  5, 219,  89, 281, 125,  0
  6, 215, 140, 288, 148,  0
Obj Anm Etiqueta EX1 E2 BN1 B2 BN3  { Inserta Acción Obj.}
  1, 22,CREDE    ,  5, 0,  0, 0,  0
  8, 22,PASO     ,  6, 1,  0, 0,  0
  1, 22,CREDE2   ,  6, 2,  0, 0,  0
  8, 22,PASO2    ,  6, 2,  0, 0,  0
  1, 22,CREDE2   ,  5, 1,  0, 0,  0
No.  X1   Y1   X2   Y2   Anm          { Inserta restricción }
  0, 190,  25, 243,  90, 32
  0,  59,  24,  91,  90, 32
  0,  92,  24, 122,  90, 33
  0, 158,  26, 189,  91, 34
  0, 124,  25, 157,  90, 35
```

CAPITULO 11

VIDEO JUEGOS EN MEXICO

CAPITULO 11
VIDEO JUEGOS EN MEXICO

¡Perfecto!, ya terminamos el juego ¿y ahora que?, no es divertido jugar algo que ya nos sabemos de memoria. Tal vez podriamos distribuirlos entre nuestros amigos, o incluso venderlo. Bueno, si nos dedicáramos seriamente al desarrollo de juegos para computadora, necesitariamos darle otro tipo de presentación más comercial, proponer un precio y buscar medios de distribución para poder venderlo.

En Estados Unidos la venta de juegos para computadora representa un mercado enorme (con una gran competencia). En México, sin embargo, el desarrollo de este tipo de programas parece no llamarle la atención a los empresarios y/o programadores profesionales ¿por qué?.

La información siguiente se pretende sea de utilidad para aquellos que pretenden comercializar su juego (aunque puede servir también para otro tipo de programas).

11.1. REGISTRO DEL JUEGO

11.1.1. Derechos de Autor:

El derecho de autor protege cualquier obra artística o intelectual. Asi, los trabajos de literatura, tales como libros de cualquier tipo, revistas, cuentos, artículos y cartas; las obras musicales, grabadas o impresas de cualquier modo; las obras dramáticas, incluidas las partituras y permisos de representación en cualquier forma; y los trabajos de arte, tales como las pinturas, dibujos, carteles, fotografías, esculturas y artesanias, están amparados por la Ley Federal del Derecho de Autor. Esta ley cubre también a las obras que por analogia se consideren dentro de los tipos aquí mencionados, y les brinda la protección legal que les corresponde.

11.1.2. Publicación de la tesis:

Si la tesis para obtener el título profesional es sumamente original en su temática es posible publicarla. Si se va a elaborar la tesis con el único propósito de cumplir con uno de los requisitos que se exigen para presentar el examen profesional, no es estrictamente necesario que se obtenga el derecho de autor de la misma. Sin embargo, como una tesis resume el esfuerzo de varios años de estudios superiores y representa el producto de una labor intelectual y de investigación, quizá a los autores les convenga publicarla tan pronto como la terminen, para evitar que alguien la use o modifique sin su permiso; en tal caso, si deben registrarla y obtener el derecho de autor. En cuanto al programa, se puede lograr la exclusividad sobre el juego si se elabora un prototipo de éste y se tramitan los correspondientes derechos de autor en la Dirección General del Derecho de Autor, dependencia oficial a cargo de la Secretaria de Educación Pública. El juego por computadora seguramente quedará amparado por el

derecho de autor, tanto como obra artística como literaria, pues el diseño de las pantallas y dibujo de los personajes pueden ser considerados como trabajo artístico; aparte, las instrucciones u otro material escrito corresponden a la labor literaria. En cuanto al manual, debe entregar junto con la solicitud de registro (que debe incluir una breve descripción de la obra), se tendrán que entregar tres copias del original y pagar una módica cantidad por concepto del registro. Ahora que si se prefiere, se pueden obtener resultados positivos más inmediatos, y quizá con menos dificultades, se encarga el trámite a un abogado especializado en la materia.

11.1.3. Dominio público:

Una vez que un artista o escritor registra y obtiene el derecho de autor de su obra, éste le pertenece hasta su muerte y, si él así lo dispone en su testamento, puede continuar 30 años más en manos de sus herederos. Al cumplirse este plazo (o antes, si el creador fallece sin dejar beneficiarios y las autoridades consideran que el trabajo artístico tiene interés público), la obra pasa a ser del dominio público, lo que significa que cualquier persona puede usarla en provecho propio. Quien decida utilizarla con fines de lucro, tendrá que cuidarse de respetar los derechos adquiridos con anterioridad sobre arreglos, adaptaciones y medios de explotación.

11.2. TIPOS DE DISTRIBUCIÓN Y/O VENTA DEL JUEGO

Existen varias formas para poder vender y/o el juego, cada una de las cuales tiene sus pros y sus contras. Con excepción de las dos últimas formas de distribución, las demás utilizan los boletines de noticias electrónicos (más conocidos como BBS's) los cuales pueden accederse vía módem, o por medio de la red internacional de computadoras (internet), almacenando (o subiendo) sus programas en un servidor público de archivos para que queden a disposición del público en general.

11.2.1. Freeware

(*free* = libre, gratis). El programa puede ser usado y distribuido libremente sin ningún compromiso, a veces con la única condición de que no se modifique el código y que se respete o reconozca el nombre del autor.

En esta categoría generalmente se encuentran generalmente los programas demostrativos (demos), códigos fuentes para algún lenguaje de programación específico, algunas pequeñas utilerías y unos cuantos programas de otro tipo. El autor los distribuye, como una forma de aportación, para promover un producto o simplemente para obtener reconocimiento dando una muestra de su trabajo.

11.2.2. Cardware

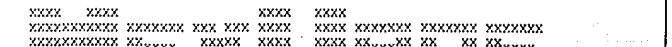
(*Card* = Carta): Este tipo de distribución tiene las mismas características del freeware, excepto que el autor espera una carta o una postal con la opinión del usuario del producto. Este curioso tipo de distribución es rarísimo, de hecho sólo lo hemos visto una vez.

11.2.3. Shareware

(*share* = compartir): Shareware es software que se puede compartir con otras personas. También significa "Probar antes de comprar". A diferencia del freeware, el usuario debe pagar por el programa, aunque puede usarlo gratis por un cierto periodo de tiempo. Generalmente los programas que se distribuyen como freeware tienen deshabilitadas algunas características (como por ejemplo salvar el trabajo en un archivo) y si al usuario le pareció funcional y adecuado el programa puede mandar una tarjeta de registro con una forma de pago, por lo cual recibirá el programa completo, manuales y tal vez algún otro programa de regalo. Otros programas funcionan con todas sus características, pero sólo trabajan un número limitado de veces o hasta cierta fecha. El shareware permite a los programadores de escasos recursos distribuir sus productos con muy poco o ningún costo y a los usuarios adquirir programas con un precio relativamente bajo. Contrario a los que se pudiera pensar, la calidad de los programas desarrollados para ser vendidos como shareware en muchos casos es tan alta o mejor que la que tiene los programas vendidos de forma comercial.

Los programadores que prefieren este tipo de distribución/venta, confían en que después de un periodo de prueba, los usuarios queden lo suficientemente impresionados por el programa para pedir (y pagar) la versión completa del software.

Si queremos ahorrarnos la molestia de subscribirnos a algún BBS y preferimos concentrarnos tan sólo en el desarrollo del juego, en México existe una asociación llamada Mexware que se dedica a distribuir por distintos medios el programa (debe ser del tipo freeware o shareware). A continuación se reproduce íntegro el archivo Mexware.nfo que fue obtenido de un BBS en México llamado Coyoacán. (Nosotros no hemos tenido ninguna clase de contacto con la siguiente sociedad, pero nos pareció buena idea ponerla como alternativa).



¿Qué es MexWare?
Es una sociedad que se encarga de apoyar el Shareware y el Freeware mexicano por diferentes medios.
Uno de ellos es la distribución. Tenemos una extensa lista de personas que se interesan en obtener estos programas, pero debido a diferentes razones les es imposible. Cuando tu mandas tu catalogo se va a distribuir a todas las personas interesadas posibles, con tu información; para que ellos se puedan comunicar contigo y registrar el programa.
Otra de ellas es dar asesoría en como escribir Shareware/Freeware.
Les vamos a ayudar en muchas cosas, en cómo escribir manuales, cómo es más atractiva la presentación y lo demás acerca de ello.

Yo soy programador, ¿cómo me puedo unir a MexWare?
Es muy fácil y barato, lo que tienes que hacer es lo siguiente:

1. Llenar la solicitud que se incluye en este programa
2. Mandarnos tus discos/catálogo que deseas distribuir
3. Pagar la cuota requerida

Hey!, ¿cuál es la cuota requerida?
Ok, se divide en tres formas diferentes. Debido al precio de los gastos de envío y sellos postales, el precio de la distribución es el siguiente:

N$ 10.00 -- Para mandar solamente el catalogo de programas en hojas
N$ 20.00 -- Para mandar los discos que contengan los programas
N$ 5.00 -- Para mandar catalogo por medio de BBS (solo catálogos)

La cuota requerida no es con fines de lucro, es solamente para el pago de los gastos de envío.

Estoy interesado(a)!!, ¿a dónde debo mandar lo requerido?
Debes mandar lo necesario a la siguiente dirección:

MexWare
Guerrero #343, Ent D-108
CP 06900, Mex DF
MEXICO

Muchas gracias por tu atención.
Si estás interesado ó deseas más información puedes mandarnos una carta con tus preguntas y dudas y ten por seguro se van a contestar lo mas pronto posible.

11.2.4. Venta directa

Otra opción es venderle directamente el programa a los usuarios. Esta no es la forma ideal de vender juegos (a menos que se tenga MUCHOS conocidos). Generalmente, conviene más vender programas especializados de administración contabilidad, nominas, etc. a empresas de todo tipo, las cuales tienen más posibilidad de pagar cantidades fuertes de dinero si el programa lo justifica.

11.2.5. Venta con apoyo de distribuidores:

Cuando se tienen los recursos para fabricar paquetes de software en formato comercial y al mayoreo, se pueden buscar distribuidores que lo ofrezcan al público en general.

11.2.6. Unirse a una empresa desarrolladora de software:

Como dice el viejo y conocido refrán, "si no puedes con ellos, únetelos". Se pueden ofrecer los servicios a una empresa ya establecida que se dedique al desarrollo y venta de programas. Tal vez, con el tiempo, se adquiera la experiencia necesaria para volverse un diseñador independiente.

Epic MegaGames es una corporación dedicada a desarrollar, distribuir y vender juegos por medio del shareware. A continuación reproducimos una solicitud de programadores y artistas que venia en la orden de compra de un juego llamado Pinball 2 (la solicitud venia en inglés, asi que la traducción fue lo más fiel que se pudo)

Atención programadores y artistas:

¡Unete al equipo de Epic MegaGames!. Estamos buscando buenos programadores y artistas que tengan el talento y la inspiración para crear la siguiente generación de juegos para computadora.

Los programadores deben tener experiencia en codificar grandes juegos y gráficas en lenguaje 'C' y ensamblador. También, es importante la habilidad en el diseño de juegos.

Los artistas deben tener familiaridad con arte del mundo real asi como arte por computadora y animación. Crear arte para un juego requiere un amplio rango de talentos artísticos en dibujo, sombreado y animación.

Como unirse al equipo EPIC.

Algunos desarrolladores trabajan en nuestra oficina en Maryland, pero muchos otros equipos de desarrolladores trabajan independientemente desde E.U.A., Canadá, Holanda, Dinamarca, Finlandia y Alemania.

Si estás motivado y piensas que eres lo suficientemente bueno para sorprendernos, por favor envia una muestra de tu mejor trabajo a:

EPIC MegaGames
ATTN: RECLUTING DEPARTMENT
10406 HOLBROOK DRIVE
POTOMAC, MD 20854 U.S.A.

CONCLUSIONES

CONCLUSIONES

En esta tesis hemos presentado tres herramientas para realizar juegos de video. La elaboración de estas nos llevo alrededor de 4 meses de trabajo. La elaboración propia del juego, utilizando las herramientas ya hechas, nos llevó alrededor de 1 mes. Debe considerarse, sin embargo, el periodo de tiempo que se dedicaba al diseño, la corrección de pequeños errores y la adición de código en los programas, proceso que se debia hacer sobre la marcha, para el mejor desempeño del juego. En el capitulo 10 se muestran y explican los pasos que se realizaron para la elaboración del juego, pero no se mencionan los problemas y contratiempos que se iban presentando. Por ejemplo, algunas filmaciones debían repetirse; varias imágenes capturadas debian retorcarse, ya que el video capturado no siempre conservaba las mismas dimensiones y colores; la captura de voces debia realizarse en dias diferentes, según la disponibilidad de las personas involucradas; con la práctica encontrábamos que habia formas más fáciles y rápidas de hacer los mismos procesos, por lo que se podía haber acelerado el trabajo; etc. Por supuesto que en este momento se puede realizar el mismo trabajo en la mitad del tiempo o menos, y cualquier persona que maneje las herramientas en dos o tres ocasiones puede adquirir la misma habilidad y rapidez.

El utilizar software de apoyo fue una forma de acelerar los procesos de diseño, pero representó una dependencia de estos programas extranjeros. Diseñar un software de este tipo con recursos exclusivamente propios no es tan fácil, pero si es posible. Para hacerlo se debe utilizar un poco más de tiempo para elaborar herramientas específicas para cada proceso. Para complementar la lista de herramientas de esta tesis se puede:

- Agregar una herramienta de dibujo para sustituir a los editores de gráficas.
- Diseñar un editor de archivos CMF o de formato propio para la música de fondo y no tener que utilizar software de composición externo.
- Elaborar una herramienta de grabación de sonidos.
- Y tal vez los más difícil, elaborar un programa de captura de imágenes para ser utilizado con una tarjeta digitalizadora de video.

Con nuestros conocimientos actuales, podriamos hacer todas las herramientas mencionadas excepto la última (por el momento).

La inversión monetaria que hemos hecho, la hemos calculado alrededor de N$12000.00 para un solo equipo de computo. A esto habrá que añadir el costo que tendria el equipo complementario tal como, cámara de filmar y un equipo de sonido. Este gasto se realiza solo una vez y se pueden generar una enorme cantidad de juegos que se venden por lo menos en N$200.00 la copia.

Es importante señalar, que parte del equipo que utilizamos no es indispensable (tarjeta digitalizadora, scanner, cámara de video, etc.), sino que ayudan a agilizar el desarrollo del juego. De esta manera, la relación costo - beneficio puede adaptarse a las necesidades del programador por medio de la relación tiempo - equipo especial, es decir, un mismo juego puede hacerse sin equipo adicional lo que ahorraría la inversión inicial, pero se necesitaría más tiempo para su diseño.

Retomando nuestros objetivos iniciales, podemos proporcionar una idea de la enorme cantidad de código necesario para generar un solo juego, pero que tiene el nivel suficiente para competir con un mercado internacional tan difícil.

Las herramientas presentadas, no se encuentran disponibles dentro del software comercial actual, por lo que son innovadoras y fáciles de usar. Herramientas semejantes son las que utilizan las casas productoras de juegos de video, y solo son para su uso exclusivo.

Como los juegos también pueden ser educativos, bien aplicados pueden representar una aportación al desarrollo tecnológico y cultural de nuestro país, y no solo una forma de lucro personal.

Este trabajo sin duda alguna, resolverá una enorme cantidad de dudas a aquellos interesados en la programación de juegos. Cualquier persona interesada en el tema - sin importar su nivel -, encontrará muchas cosas interesantes en esta obra.

Esperamos con esto que esta tesis sirva para tapar ese enorme hueco que existe sobre el tema en nuestro país.

GLOSARIO,

APENDICES Y

BIBLIOGRAFIA.

GLOSARIO

⇒ **Animación:** Ilusión de movimiento continuo, provocado por una sobreimposición de imágenes ligeramente distintas entre si. A diferencia de una película en donde todos los elementos ya se encuentran en cada cuadro, la animación implica "armar" las pantallas, o sea, tomar una gráfica base e irle agregando duendes que cambian de forma y ubicación.

⇒ **Asistente:** Es un procedimiento que permite especificar datos y/o valores sin necesidad de capturarlos, por medio de una interface gráfica.

⇒ **AVI:** (Audio Video Interleaved). Formato para archivos de video digitalizado.

⇒ **CMF:** (Creative Music File). Formato diseñado por Creative Labs para almacenar música F.M. Es similar al formato MIDI, solo que de menos calidad.

⇒ **Duende:** Bloque gráfico que puede moverse independientemente a la pantalla de fondo.

⇒ **GIF:** (Graphic Interchange Format). Formato gráfico desarrollado por Compuserve.

⇒ **Icono:** Representación gráfica de algún elemento. Generalmente es un pequeño bloque gráfico que contiene un diagrama caracterizando un proceso, un programa, etc.

⇒ **Interrupción:** Es una llamada de atención al CPU por parte de algún dispositivo para indicarle que ha sucedido algún evento que requiere sea atendido.

⇒ **Libreto:** Herramienta que nos permite definir e integrar todos los elementos que conforman una escena especifica en una sola estructura de datos.

⇒ **MCGA:** (Multi-Color Graphics Array). Es un tipo de tarjeta para gráficos, pero también se denomina así al modo de video 13h (320x200 256 colores simultáneos). Debido a su notable capacidad cromática, es el modo de video más usado por los programadores para realizar juegos para computadora.

⇒ **Memoria extendida:** Es la memoria fisica direccionada más allá de 1 MB.

⇒ **MIDI:** (Musical Instrument Digital Interface). Dispositivo de sonido que permite reproducir música producida por sintetizadores.

⇒ **MPC:** (Multimedia PC). Especificación estandar de requerimientos para un equipo multimedia. Actualmente existen dos niveles.

⇒ **Paleta:** Es una matriz que contiene todas las caracteristicas necesarias para representar cada color en un modo gráfico determinado. Cada modo gráfico tiene un determinado número de colores simultáneos (2, 4, 16, 256, etc.) el cual no puede modificarse. La tonalidad de cada color es resultado de la combinación de los tres matices primarios (Rojo, Verde y Azul), la cantidad de cada matiz para cada color se almacena en la paleta.

⇒ **PCX:** Formato gráfico desarrollado para el programa Paintbrush por ZSoft.

⇒ **Pixel:** (De Picture X Element). Unidad básica con la cual se componen las gráficas.

⇒ **Programa ejecutor:** Programa que interactua con el usuario, de modo que pueda utilizar todos los elementos definidos previamente por el libreto y el editor de duendes/animaciones.

⇒ **Programa residente:** Es un programa que se carga en memoria pero no se ejecuta inmediatamente, en su lugar devuelve el control al sistema operativo y permanece inactivo hasta que se presenten las condiciones adecuadas para que empiece a funcionar (generalmente cuando se activa una interrupción por medio de una combinación de teclas).

⇒ **Video:** Televisión. Nombre con que se designa la parte de una señal correspondiente a la imagen, por contraposición a la voz *audio*.

⇒ **XMS:** (eXtended Memory Specification). Especificación de la memoria extendida la cual es controlada por el manejador HIMEM.SYS, y que permite administrarla más fácilmente.

APENDICE 1. FUENTE DEL PROGRAMA EDITOR.PAS

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
ENEP - ARAGON
INGENIERIA EN COMPUTACION

TESIS

FUNDAMENTOS DE PROGRAMACION PARA LA ELABORACION DE JUEGOS DE VIDEO

Editor.pas : Programa de edición de duendes, animación y paletas.

Programado por :
Itsmael Manzo Salazar
Francisco Xavier Espinosa Madrigal

```
{$M 30000,0,655300}

Uses LibSB, libdnd, librat, libgra, libuti, crt, libpan, libxms, libdir, libani,
libPal;

Type
  Principal = object(animar)
     Constructor inicio;
     Destructor  final;
     Procedure Pantalla_Menu1;
     Procedure Restaura_duende;
     Procedure Menu1;
     Function  Menu2 : byte;
     Function  Menu3 : byte;
     Procedure Menu4(Var Mono : Duende);
     Procedure Menu5;
   end;

Var
  P11 : Paletas;
  P   : principal;          { Variable para procedimiento principal          }
  F   : formato;            { Variable para despliegue de formatos           }
  D   : Dir_mcga;           { variable para despliegue de directorios        }
  Mem_datos : boolean;      { Indica si hay un duende en memoria             }
  Cont_duende : integer;    { Contador de imagen actual de duende            }
  nomduende : string;       { Nombre en disco del duende actual              }
  nompantalla : string;     { Nombre en disco de la pantalla actual          }


SE EJECUTA AL INICIO DEL PROGRAMA
Constructor principal.inicio;
Var
   Pal1, Pal2 : Paleta;
   DA : datosanima;
   carro : duende;
   x,y,boton : Word;
Begin
```

```
  Carro.Total := 0;
  if checa_raton then;
  mem_datos:=false;
  venx1:=14; veny1:=122; venx2:=319; veny2:=199;
  ModoGrafico($13);
  getmem(pantemporal,64000);
  leerduende(bot,'editor.dnd');
  if ReservaXms(189,xmsman[30]) = 0 then;
  f.verbin('editor.bin',pantemporal^);
  escribexms(xmsman[30],pantemporal^,64000,0);
  Pll.DiscoL('Defecto.Pal',1);
  F.verpcx('..\pcx\animgraf.pcx',pantemporal^);
  Pll.Iguala_Pal(Pantemporal^);
  If Busca_FMdriver Then
  begin
     Inicializa_FM;
     Lee_FM('..\sb\starfm.cmf');
     toca_fm;
  end;
  escribexms(xmsman[30],pantemporal^,64000,128000);
  nomduende:='..\DND\*.dnd';
  nomanima:='..\ANM\*.anm';
  D.Define_Unidades;
  NomPantalla := '';
  Leerduende(carro,'..\dnd\carro.dnd');
  da.x:=-20; da.y:=130;
  IniciaAni(dA);
  lee_archivo('..\anm\carro.anm',DA);
  Primer_anima(DA,'CARRO');
  Boton := 0;
  while (not keypressed) And (Boton=0) do
    begin
       PosRaton(Boton, X, Y);
       {Pantalla de fondo actual importada}
       leexms(xmsman[30],pantemporal^,64000,128000);
       Corre_Anima(DA.AniAct.etiqueta,DA,carro);
       despliega(pantemporal);
    end;
  While keypressed do readkey;
  FinalizaAni(DA);
  liberaduende(carro);
  venx1:=0; veny1:=0; venx2:=319; veny2:=199;
  deten_FM;
End;
```

SE EJECUTA AL FINALIZAR EL PROGRAMA

```
Destructor Principal.final;
Begin
  delay(100);
  liberaduende(bot);
  freemem(pantemporal,64000);
  if mem_datos then leerduende(monos[1],'editor.dnd');
  if LiberaXms(xmsman[30]) = 0 then;
  ModoGrafico($03);
  Termina_SBP;
  Finaliza_FM;
End;
```

RUTINAS PARA DESPLEGAR PANTALLAS

```
Procedure Principal.Pantalla_menu1;
Begin
  leexms(xmsman[30],mem[$a000:0],64000,0);  {Pantalla Menu Principal          }
```

```
  putimagen(77,177,bot.dnd[33]);     { <                                  }
  putimagen(105,177,bot.dnd[35]);    { >                                  }
  putimagen(41,177,bot.dnd[37]);     { << .                               }
  putimagen(133,177,bot.dnd[39]);    { >>                                 }
  putimagen(5,177,bot.dnd[41]);      { I<<                                }
  putimagen(169,177,bot.dnd[43]);    { >>I                                }
  putimagen(281,177,bot.dnd[45]);    { Puerta                             }
  Putimagen(265,9,bot.dnd[3]);       { Sacar de disco                     }
  Putimagen(292,9,bot.dnd[1]);       { Meter a disco                      }
  Putimagen(265,33,bot.dnd[9]);      { Rectangulo                         }
  Putimagen(292,33,bot.dnd[5]);      { Bote de basura                     }
  Putimagen(265,57,bot.dnd[11]);     { Leer pantalla                      }
  Putimagen(292,57,bot.dnd[21]);     { Animacion                          }
  Putimagen(265,81,bot.dnd[19]);     { Posición del duende                }
  Putimagen(292,81,bot.dnd[29]);     { Copiar duendes                     }
  If FijarPaleta Then
    Putimagen(265,105,bot.dnd[49])   { Paleta Fija                        }
  Else
  Putimagen(265,105,bot.dnd[7]);     { Manipular paletas                  }
  cuadrado(228,180,256,193,0);
  cuadro(228,180,256,193,1);
  Letrero(33,20,Espacios(Nac(MaxAvail),6,2)+'B',5);
  Cuadrado2(33*8,20*8,319,21*8-1,2);
End;
```

UTILIZADO PARA DESPLEGAR LO RELACIONADO A UN DUENDE

```
Procedure Principal.restaura_duende;
Begin
  venx1:=8; veny1:=14; venx2:=253; veny2:=168;
  cursorraton(false);
  cuadrado(7,13,255,170,0);
  If Cont_duende>0 Then putduende(10,15,monos[1],mem[$a000:0],cont_duende);
  cuadrado(229,181,255,192,0);
  letrero(29,23,nac(cont_duende),2);
  Letrero(33,20,Espacios(Nac(MaxAvail),6,2)+'B',5);
  Cuadrado2(33*8,20*8,319,21*8-1,2);
  cursorraton(true);
End;
```

MANEJADORES DE PANTALLLAS (SUBMENUS)

```
Function principal.Menu2 : byte;
Var
  salir : boolean;          { Para salir del programa                   }
  boton : word;             { Boton del raton                           }
Begin
  cursorraton(false);
  escribexms(xmsman[30],mem[$a000:0],64000,64000);
  Putimagen(10,150,bot.dnd[23]);
  Putimagen(40,150,bot.dnd[25]);
  Putimagen(70,150,bot.dnd[27]);
  Putimagen(232,150,bot.dnd[31]);
  Cursorraton(true);
  salir:=false;
  LimiteRaton(7*2,13,253*2,168);
  while salir<>true do
   Begin
     Icono
        FLIP HORIZONTAL
     if Checa_boton(10,150,bot,23, 'h') then Begin menu2:=1; salir:=true; end;
```

```
      Icono
          FLIP VERTICAL
      if Checa_boton(40,150,bot,25, 'v') then Begin menu2:=2; salir:=true; end;

      Icono
          ROTACION
      if Checa_boton(70,150,bot,27, 'r') then Begin menu2:=3; salir:=true; end;

      Icono
          PUERTA DE SALIDA
      if Checa_boton(232,150,bot,31,#27) then Begin menu2:=4; salir:=true; end;
    end;
  cursorraton(false);
  leexms(xmsman[30],mem[$a000:0],64000,64000);  {Pantalla Menu Principal      }
  Cursorraton(true);
  LimiteRaton(0,0,319*2,199);
End;
```

MENU DE IMPORTACION DE IMAGENES CON OTROS FORMATOS GRAFICOS.

```
Function principal.Menu3 : byte;
Var
  salir : boolean;        { Para salir del programa                          }
  boton : word;           { Boton del raton                                  }
  mientras : byte;
Begin
  cursorraton(false);
  escribexms(xmsman[30],mem[$a000:0],64000,64000);
  cuadrado(260,10,316,100,2);
  letrero(33,2,'IMAGEN',4);
  cuadro(260,10,316,100,1);
  Putimagen(265,33,bot.dnd[3]);
  Putimagen(292,33,bot.dnd[1]);
  Putimagen(292,80,bot.dnd[31]);
  cursorraton(true);
  salir:=false;
  LimiteRaton(260*2,10,316*2,100);
  while salir<>true do
   Begin
      Icono
         LEER DE DISCO
      if Checa_boton(265,33,bot,3, '1') then
        Begin mientras:=1; salir:=true; end;

      Icono
         ESCRIBIR EN DISCO
      if Checa_boton(292,33,bot,1, 's') then
        Begin mientras:=2; salir:=true; end;

      Icono
         SALIR DE LA RUTINA
      if Checa_boton(292,80,bot,31,#27) then
        Begin mientras:=3; salir:=true; end;
   end;

  if mientras<>3 then
    begin
      cursorraton(false);
      cuadrado(260,10,316,100,2);
      if mientras =1 then letrero(33,2,' LEER ',4);
      if mientras =2 then letrero(33,2,'SALVAR',4);
```

```
    cuadro(260,10,316,100,1);
    Putimagen(265,33,bot.dnd[13]);
    Putimagen(292,33,bot.dnd[15]);
    Putimagen(265,57,bot.dnd[17]);
    Putimagen(292,57,bot.dnd[47]);
    Putimagen(292,80,bot.dnd[31]);
    Cursorraton(true);
    salir:=false;
    while salir<>true do
     Begin
      Icono
          FORMATO PCX
      if Checa_boton(265,33,bot,13, 'p') then
        Begin
          if mientras=1 then mientras:=4;
          if mientras=2 then mientras:=5;
          salir:=true;
        end;
      Icono
          FORMATO GIF
      if Checa_boton(292,33,bot,15, 'g') then
        Begin
          if mientras=1 then mientras:=6;
          if mientras=2 then mientras:=7;
          salir:=true;
        end;
      Icono
          FORMATO BIN
      if Checa_boton(265,57,bot,17, 'b') then
        Begin
          if mientras=1 then mientras:=8;
          if mientras=2 then mientras:=9;
          salir:=true;
        end;
      Icono
          FORMATO AVI
      if Checa_boton(292,57,bot,47, 'a') then
        Begin
          if mientras=1 then mientras:=10;
          if mientras=2 then mientras:=11;
          salir:=true;
        end;
      Icono
          SALIR DE LA RUTINA
      if Checa_boton(292,80,bot,31,#27) then
        Begin mientras:=3; salir:=true; end;
    end;
  end;
 cursorraton(false);
 leexms(xmsman[30],mem[$a000:0],64000,64000);  {Pantalla Menu Principal      }
 Cursorraton(true);
 LimiteRaton(0,0,319*2,199);
 menu3:=mientras;
End;

MENU DE COPIADO DE DUENDES
Procedure principal.Menu4(Var mono : Duende);
Var
  IniciaNum,
  salir    : boolean;        { Para salir del programa                      }
  Desde, Hasta, En,
```

```
  Xr, Yr,
  boton    : word;          { Boton del raton                    }
  Tecla1,
  Tecla2   : Char;
  ValorAct : Byte;
  Cad, Cad2: String;
Begin
  IniciaNum := True;
  cursorraton(false);
  escribexms(xmsman[30],mem[$a000:0],64000,64000);
  letrero(29,23,nac(cont_duende),2);
  Putimagen(232,150,bot.dnd[31]);
  Putimagen(208,150,bot.dnd[29]);
  Desde := Cont_Duende;
  Hasta := Cont_Duende;
  En    := 1;
  ValorAct := 1;
  Cuadrado(4*8,19*8-1,24*8,21*8,2);
  Letrero(5, 19,'DESDE  HASTA  EN: ', 4);
  Cuadrado(3*8-1,19*8-1,4*8-1,21*8,1);
  Letrero(3,19,'+',6);   Letrero(3,20,'-',6);
  Str(Desde:3,Cad);  Letrero(6,20,Cad+' ',14);
  Str(Hasta:3,Cad);  Letrero(13,20,Cad+' ',14);
  Str(En:3,Cad);     Letrero(19,20,Cad+' ',14);
  Cuadrado(10*8,19*8,12*8-1,21*8-1,1);
  Cuadrado(17*8,19*8,19*8-1,21*8-1,1);
  salir:=false;
  venx1:=8; veny1:=14; venx2:=253; veny2:=140;
  LimiteRaton(8,14,253*2,168);
  PosRaton(Xr, Yr, Boton);
  while salir<>true do
   Begin
     Icono
        PUERTA DE SALIDA
     Case ValorAct Of
         1 : Begin
               LeeEntero(Desde,6,20,3,Tecla1, Tecla2
                         ,Boton, Xr, Yr, 14,Inicianum);
               If RangoRaton(3*8,19*8,4*8-1,20*8-1) Then Inc(Desde);
               If RangoRaton(3*8,20*8,4*8-1,21*8-1) Then
                  If Desde>0 Then Dec(Desde);
               If Desde=0 Then Desde := 1;
               If Desde>Hasta Then Desde := Hasta;
             End;
         2 : Begin
               LeeEntero(Hasta,13,20,3,Tecla1, Tecla2
                         ,Boton, Xr, Yr, 14,Inicianum);
               If RangoRaton(3*8,19*8,4*8-1,20*8-1) Then Inc(Hasta);
               If RangoRaton(3*8,20*8,4*8-1,21*8-1) Then
                  If Hasta>0 Then Dec(Hasta);
               If Hasta<Desde Then Hasta := Desde;
               If Hasta>Mono.Total Then Hasta := Mono.Total;
             end;
         3 : Begin
               LeeEntero(En,19,20,3,Tecla1, Tecla2
                         ,Boton, Xr, Yr, 14,Inicianum);
               If RangoRaton(3*8,19*8,4*8-1,20*8-1) Then Inc(En);
               If RangoRaton(3*8,20*8,4*8-1,21*8-1) Then
                  If En>0 Then Dec(En);
               If En=0 Then En := 1;
               If En>Mono.Total+1 Then En := Mono.Total+1;
             End;
```

```
    End;
    If Odd(Boton) Then
    begin
       If RangoRaton(5*8,20*8,9*8,21*8) Then ValorAct := 1;
       If RangoRaton(13*8,20*8,17*8,21*8) Then ValorAct := 2;
       If RangoRaton(19*8,20*8,23*8,21*8) Then ValorAct := 3;
    end;
    If (Tecla1=#9) Or (Tecla2 In [#72]) Then
      If ValorAct<3 Then Inc(ValorAct) Else ValorAct := 1;
    If (Tecla2 In [#80, #15]) Then If ValorAct>1
      Then Dec(ValorAct) Else ValorAct := 3;
    CursorRaton(False);
    Cuadrado(VenX1, Veny1, Venx2, Veny2, 0);
    Case ValorAct Of
      1 : PutDuende(8,14,Mono,Mem[$A000:0],Desde);
      2 : PutDuende(8,14,Mono,Mem[$A000:0],Hasta);
      3 : If En<Mono.Total Then PutDuende(8,14,Mono,Mem[$A000:0],En);
    End;
    Str(Desde:3,Cad);  Letrero(6,20,Cad+' ',14);
    Str(Hasta:3,Cad);  Letrero(13,20,Cad+' ',14);
    Str(En:3,Cad);     Letrero(19,20,Cad+' ',14);
    CursorRaton(True);
    If (Checa_boton(208,150,bot,29,'c')) Or (UpCase(Tecla1)='C') then
    Begin
      CopiaDuende(Mono,Desde, Hasta, En);  Salir := True;
    End;
    If (Checa_boton(232,150,bot,31,#27)) Or (Tecla1=#27) then salir:=true;
    If Odd(Boton) Then Delay(100);
  end;
 LimiteRaton(0,0,639,199);
 leexms(xmsman[30],mem[$a000:0],64000,64000);  {Pantalla Menu Principal      }
 Cursorraton(true);
End;

MENU DE MANEJO DE PALETAS
Procedure Principal.Menu5;
Var
  IniciaNum,
  salir      : boolean;     { Para salir del programa                        }
  Desde,
  Actual,
  Xr, Yr,
  boton      : word;        { Boton del raton                                }
  Tecla1,
  Tecla2     : Char;
  Cad, Cad2  : String;
  Pal2       : Paleta;
Begin
  IniciaNum := True;
  cursorraton(false);
  escribexms(xmsman[30],mem[$a000:0],64000,64000);
  Putimagen(232,150,bot.dnd[31]);
  If FijarPaleta Then
     Putimagen(208,150,bot.dnd[49])
  Else
     Putimagen(208,150,bot.dnd[7]);
  Putimagen(184,150,bot.dnd[3]);
  Putimagen(160,150,bot.dnd[1]);
  Putimagen(14,130,bot.dnd[5]);
  Putimagen(14,106,bot.dnd[49]);
  Desde := 1;  Actual := 0;
  Cuadrado(4*8,19*8-1,18*8,21*8,2);
```

```
  Letrero(5, 19,'PALETA  TOTAL', 4);
  Cuadrado(3*8-1,19*8-1,4*8-1,21*8,1);
  Letrero(3,19,'+',6);   Letrero(3,20,'-',6);
  Str(Desde:3,Cad);  Letrero(6,20,Cad+' ',14);
  Str(TotalPal:3,Cad);  Letrero(14,20,Cad+' ',10);
  Cuadrado(11*8,19*8,13*8-1,21*8-1,1);
  salir:=false;
  LimiteRaton(8,14,253*2,168);
  PosRaton(Xr, Yr, Boton);
  Pll.LeePaleta;
  Pal2 := Pll.Pal;
  while salir<>true do
   Begin
     Icono
        PUERTA DE SALIDA
     If Actual<>Desde Then
     begin
        If Actual>0 Then Pll.DiscoL('Defecto.Pal',Desde);
        Pll.PonPaleta;
        Pll.Escala2(PanTemporal^);  Actual := Desde;
     end;
     LeeEntero(Desde,6,20,3,Tecla1, Tecla2, Boton, Xr, Yr, 14,Inicianum);
     If RangoRaton(3*8,19*8,4*8-1,20*8-1) Then Inc(Desde);
     If RangoRaton(3*8,20*8,4*8-1,21*8-1) Then
                    If Desde>1 Then Dec(Desde);
     If Desde>TotalPal+1 Then Desde := TotalPal+1;
     If FijarPaleta Then
     begin
       If (Checa_boton(208,150,bot,49,'p')) Or (UpCase(Tecla1)='P') then
       Begin
          Pll.Pal := Pal2;   Pll.PonPaleta;
       End;
     End
     Else begin
       If (Checa_boton(208,150,bot,7,'p')) Or (UpCase(Tecla1)='P') then
       Begin
          Pll.Pal := Pal2;   Pll.PonPaleta;
       End;
     End;
     If ((Checa_boton(184,150,bot,3,'l')) Or (UpCase(Tecla1)='L')) And
        (Desde<=TotalPal) then Pll.DiscoL('Defecto.Pal',Desde);
     If (Checa_boton(160,150,bot,1,'s')) Or (UpCase(Tecla1)='S') then
       Pll.DiscoE('Defecto.Pal',Desde);
     If (Checa_boton(232,150,bot,31,#27)) Or (Tecla1=#27) then salir:=true;
     If ((Checa_boton(14,130, bot,5,'b')) Or (UpCase(Tecla1)='B')) And
        (Desde<=TotalPal) then Pll.DiscoB('Defecto.Pal',Desde);
     If ((Checa_boton(14,106,bot,49,'f')) Or (UpCase(Tecla1)='F')) Then
     begin
        FijarPaleta := Not FijarPaleta;
        If FijarPaleta Then
          Putimagen(208,150,bot.dnd[49])
        Else
          Putimagen(208,150,bot.dnd[7]);
     end;
     If Odd(Boton) Then Delay(100);
     CursorRaton(False);
     Str(TotalPal:3,Cad);  Letrero(14,20,Cad+' ',10);
     CursorRaton(True);
   end;
  LimiteRaton(0,0,639,199);
  Cursorraton(true);
End;
```

MENU PRINCIPAL

```
Procedure principal.Menu1;
Var
  salir : boolean;        { Para salir del programa                    }
  x,y   : integer;
  boton : word;           { Boton del raton                            }
  xs,ys,xi,yi : integer;  { coordenadas del rectangulo                 }
  ciclo : word;           { variable auxiliar es reusable              }
  largo,ancho : word;     { Largo y ancho de la imagen actual          }
  Valor_menu2 : byte;     { Indica la opcion en submenu de efectos     }
  Valor_menu3 : byte;     { Indica la opcion en submenu de imagenes    }
  Pal1, Pal2  : Paleta;
Begin
  xi:=10; yi:=10;
  Pantalla_Menu1;
  cursorraton(true);
  salir:=false;
  while salir<>true do
   Begin
     Icono
        REGRESO UNITARIO
     if Checa_boton(77, 177,bot,33,#0#75) and (monos[1].total>0)
        and (cont_duende<>1) then
       begin
         dec(cont_duende);
         if cont_duende <1 then cont_duende:=1;
         Restaura_duende;
       end;
     Icono
        AVANCE UNITARIO
     if Checa_boton(105,177,bot,35,#0#77) and (monos[1].total>0)
        and (cont_duende<>monos[1].total) then
       begin
         inc(cont_duende);
         if cont_duende > monos[1].total then cont_duende:=monos[1].total;
         Restaura_duende;
       end;
     Icono
        REGRESO DE PAGINA
     if Checa_boton(41, 177,bot,37,#0#73) and (monos[1].total>0)
        and (cont_duende<>1) then
       begin
         dec(cont_duende,5);
         if cont_duende <1 then cont_duende:=1;
         Restaura_duende;
       end;
     Icono
        AVANCE DE PAGINA
     if Checa_boton(133,177,bot,39,#0#81) and (monos[1].total>0)
        and (cont_duende<>monos[1].total) then
       begin
         inc(cont_duende,5);
         if cont_duende > monos[1].total then cont_duende:=monos[1].total;
         Restaura_duende;
       end;
     Icono
        POSICION INICIAL
     if Checa_boton(5,  177,bot,41,#0#71) and (monos[1].total>0)
        and (cont_duende<>1) then
       begin
```

```
      cont_duende:=1;
      Restaura_duende;
    end;

  Icono
      POSICION_FINAL
  if Checa_boton(169,177,bot,43,#0#79) and (monos[1].total>0)
     and (cont_duende<>monos[1].total) then
    begin
      cont_duende:=monos[1].total;
      Restaura_duende;
    end;
  Icono
      LEER DE DISCO
  if Checa_boton(265,9, bot,3, '1') then
    Begin
      cursorraton(false);
      if D.lee_dirarch(NomDuende, false) then
        begin
          if mem_datos then
            begin
              liberaduende(monos[1]);
              mem_datos:=false;
            end;
          leerduende(monos[1],nomduende);
          mem_datos:=true;
          cont_duende:=1;
          Restaura_duende;
        end;
      cursorraton(true);
    end;
  Icono
      SALVAR EN DISCO
  if Checa_boton(292,9, bot,1, 's') and (monos[1].total>0) then
    begin
      cursorraton(false);
      if D.lee_dirarch(NomDuende, true) then
        begin
          salvardnd(monos[1],nomduende);
        end;
      cursorraton(true);
    end;
  Icono
      RECTANGULO
  if Checa_boton(265,33, bot,9, 'r') then
    Begin
      boton:=1;
      cursorraton(false);
      leexms(xmsman[30],pantemporal^,64000,128000); {Fondo actual          }
      despliega(pantemporal);
      while boton<>2 do
        begin
          Rectangulo(boton,xs,ys,xi,yi);
          if boton=4 then  {Salva una imagen mas en memoria}
            begin
              inc(monos[1].total);
              monos[1].tam[monos[1].total]:=SizeImagen(Xs,YS,Xs+Xi,Ys+Yi);
              Getmem(monos[1].dnd[monos[1].total]
                     ,monos[1].tam[monos[1].total]);
              Getimagen(xs,ys,xs+xi,ys+yi,monos[1].dnd[monos[1].total]);
              delay(300);
```

```
            if cont_duende<1 then cont_duende:=1;
            mem_datos:=true;
          end;
      end;
    pantalla_menu1;
    if mem_datos then restaura_duende;
    cursorraton(true);
  end;
Icono
    BOTE DE BASURA
if Checa_boton(292,33, bot,5, 'b') and mem_datos then
  begin
    freemem(monos[1].dnd[cont_duende],monos[1].tam[cont_duende]);
    monos[1].tam[cont_duende]:=0;
    if (cont_duende=1) and (monos[1].total=1) then mem_datos:=false;
    if mem_datos then
      begin
        for ciclo:=cont_duende+1 to monos[1].total do
          begin
            monos[1].tam[ciclo-1]:=monos[1].tam[ciclo];
            getmem(monos[1].dnd[ciclo-1],monos[1].tam[ciclo-1]);
            move(monos[1].dnd[ciclo]^,monos[1].dnd[ciclo-1]^
                 ,monos[1].tam[ciclo]);
            freemem(monos[1].dnd[ciclo],monos[1].tam[ciclo]);
            monos[1].tam[ciclo]:=0;
          end;
      end;
    if (monos[1].total=cont_duende) and (cont_duende>1)
      then dec(cont_duende);
    dec(monos[1].total);
    if mem_datos then restaura_duende
    else
      begin
        cuadrado(7,13,255,170,0);
        cuadrado(229,181,255,192,0);
      end;
  end;
Icono
    MANEJO DE IMAGEN
if Checa_boton(265,57,bot,11, 'i') then
  Begin
    valor_menu3:=menu3;
    case (valor_menu3) of
      4 : begin
            cursorraton(false);
            If Pos('.PCX',nompantalla)=0
              Then nompantalla:='..\pcx\*.PCX';
            if D.lee_dirarch(Nompantalla, false) then
              begin
                f.VerPCX(nompantalla,pantemporal^);
                If FijarPaleta Then Pll.Iguala_Pal(PanTemporal^);
                escribexms(xmsman[30],pantemporal^,64000,128000);
                If FijarPaleta Then Restaura_Duende;
              end;
            cursorraton(true);
          end;
      6 : begin
            cursorraton(false);
            If Pos('.GIF',nompantalla)=0
              Then nompantalla:='..\GIF\*.GIF';
            if D.lee_dirarch(Nompantalla, false) then
              begin
```

```
                f.VerGIF(nompantalla,pantemporal^);
                If FijarPaleta Then Pll.Iguala_Pal(PanTemporal^);
                escribexms(xmsman[30],pantemporal^,64000,128000);
                If FijarPaleta Then Restaura_Duende;
              end;
            cursorraton(true);
          end;
      8 : begin
            cursorraton(false);
            If Pos('.BIN',nompantalla)=0
              Then nompantalla:='..\BIN\*.BIN';
            if D.lee_dirarch(Nompantalla, false) then
              begin
                f.VerBIN(nompantalla,pantemporal^);
                If FijarPaleta Then Pll.Iguala_Pal(Pantemporal^);
                escribexms(xmsman[30],pantemporal^,64000,128000);
                If FijarPaleta Then Restaura_Duende;
              end;
            cursorraton(true);
          end;
      9 : begin
            cursorraton(false);
            If Pos('.BIN',nompantalla)=0
              Then nompantalla:='..\BIN\*.BIN';
            if D.lee_dirarch(Nompantalla, true) then
              begin
                f.EscribeBIN(nompantalla,pantemporal^);
                escribexms(xmsman[30],pantemporal^,64000,128000);
              end;
            cursorraton(true);
          end;
     10 : begin
            cursorraton(false);
            escribexms(xmsman[30],mem[$a000:0],64000,64000);
            If Pos('.AVI',nompantalla)=0
              Then nompantalla:='..\AVI\*.AVI';
            if D.lee_dirarch(Nompantalla, false) then
              begin
                f.VerAVI(nompantalla,pantemporal^);
                If FijarPaleta Then Pll.Iguala_Pal(Pantemporal^);
                escribexms(xmsman[30],pantemporal^,64000,128000);
              end;
            leexms(xmsman[30],mem[$a000:0],64000,64000);
            cursorraton(true);
          end;
    end;
  end;
Icono
    MANEJO DE ANIMACION
if Checa_boton(292,57,bot,21, 'a') and (monos[1].total>0) then
  begin
    VenX1 := 0;   VenY1 := 0;   VenX2 := 319;  VenY2 := 200;
    Cursorraton(false);
    escribexms(xmsman[30],mem[$a000:0],64000,64000);
    IniciaAni(anis[1]);
    Zona_Anima(monos[1],anis[1]);
    FinalizaAni(anis[1]);
    cursorraton(false);
    leexms(xmsman[30],mem[$a000:0],64000,64000);  {Menu Principal      }
    CursorRaton(true);
  end;
```

Icono
POSICION DEL DUENDE

```
if (Checa_boton(265,81,bot,19,'p')) And (Monos[1].Total>0)  then
  Begin
    boton:=1;
    cursorraton(false);
    leexms(xmsman[30],pantemporal^,64000,128000); {Fondo actual          }
    despliega(pantemporal);
    venx1 := 0;    veny1 := 0;    venx2 := 320;    veny2 := 199;
    x := 0;    y := 0;
    while boton<>2 do
       Ubica_Duende(Boton,x,y,monos[1],Cont_Duende);
    pantalla_menu1;
    restaura_duende;
    cursorraton(true);
  End;
```

Icono
MULTIPLICAR DUENDES

```
if (Checa_boton(292,81,bot,29,'m')) And (Monos[1].Total>0)  then
  Begin
    boton:=1;
    menu4(monos[1]);
    CursorRaton(False);
    pantalla_menu1;
    restaura_duende;
    cursorraton(true);
  End;
```

Icono
UTILIZAR PALETAS

```
  if (FijarPaleta And Checa_boton(265,105,bot,49,'c'))
     Or ((Not FijarPaleta) And Checa_boton(265,105,bot,7,'c')) then
    Begin
      boton:=1;
      menu5;
      CursorRaton(False);
      pantalla_menu1;
      If Cont_duende>0 Then restaura_duende;
      cursorraton(true);
    End;
```

Icono
PUERTA DE SALIDA

```
if Checa_boton(281,177,bot,45,#27) then
Begin
   cursorraton(False);
   salir := Mensaje('¿Salir de AnimaGraf?',2)=1;
   cursorraton(True);
End;
```

Rango de la imagen
PUERTA DE SALIDA

```
if numbotonraton=1 then
  begin
    move(mem[seg(monos[1].dnd[cont_duende]^):
         ofs(monos[1].dnd[cont_duende]^)],largo,2);
    move(mem[seg(monos[1].dnd[cont_duende]^):
         ofs(monos[1].dnd[cont_duende]^)+2],ancho,2);
    if RangoRaton(9,15,largo+9,ancho+15)
       and rangoRaton(9,15,253,168) then
```

```
          begin
            valor_menu2:=menu2;
            case (valor_menu2) of
              1 : espejo_hor(monos[1].dnd[cont_duende]);
              2 : espejo_ver(monos[1].dnd[cont_duende]);
              3 : Rotacion(monos[1].dnd[cont_duende]);
            end;
            restaura_duende;
          end;
      end;
  end;
End;
```

RUTINA PRINCIPAL

```
BEGIN
  P.inicio;
  P.menu1;
  P.Final;
END.
```

APENDICE 2. FUENTE DEL PROGRAMA LIBRETO.PAS

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
ENEP - ARAGON
INGENIERIA EN COMPUTACION

TESIS

FUNDAMENTOS DE PROGRAMACION PARA LA ELABORACION DE JUEGOS DE VIDEO

Libreto.pas : Programa para capturar los elementos que contituyen la historia.

Programado por :
Francisco Xavier Espinosa Madrigal
Itsmael Manzo Salazar

```
Uses LibSB, Crt, Dos, Libuti, LibPan, LibPal, LibXMS, LibGra, LibRat,
                LibDnd, LibDir, LibAni;

Const
  MaxLinea = 100;

Type
  Principal = object
    Procedure Inicio(nombre : string);
    Procedure Final;
    Procedure Crea_libreto;
  end;

Var
  arlib : File;
  Anml  : Animar;
  P     : principal;
  F     : Formato;
  Dnd1  : Duendes;
  Rat   : Raton;
  Pll   : Paletas;
  LibNom    : String;
  DatLib    : Array[1..MaxLinea] Of Pointer;
  TotalLin  : Word;
  Dir1      : Dir_MCGA;
  MaxMusica,
  MaxAnima,
  MaxDuende : Byte;

 Procedimiento inicial.
Procedure Principal.Inicio(nombre : string);
Var
   i : Word;
Begin
  Busca_FMDriver;
  Inicializa_FM;
```

```
  Leer_FontUsuario('..\FONT\6X8.FNT',6,8);
  MaxDuende := 1;     MaxAnima  := 1;  MaxMusica := 1;
  Rat.Checa_Raton;
  Dir1.Define_Unidades;
  TotalLin := 0;
  For i := 0 To MaxLinea do DatLib[i] := Nil;
  GetMem(PanTemporal, 64000);
  ObtenXMS;
  ReservaXMS(189,XMSMan[30]);
  libNom := nombre;
  Assign(ArLib,LibNom);
  ModoGrafico(19);
  EscribeXMS(XMSMan[30], Mem[$A000:0], 64000, 128000);
  Pll.DiscoL('Defecto.Pal',1);
  F.VerPcx('Libreto.pcx', PanTemporal^);
{ Pll.Iguala_Pal(Pantemporal^); }
  Despliega(Pantemporal);
  EscribeXMS(XMSMan[30], PanTemporal^, 64000, 0);
End;
```

Procedimiento final.

```
Procedure Principal.Final;
Begin
  Termina_SBP;
  Finaliza_FM;
  FreeMem(PanTemporal, 64000);
  LiberaXMS(XMSMan[30]);
  ModoGrafico(3);
End;
```

Despliega las cabeceras de cada acción.

```
PROCEDURE EscribeParam(Col, Numero, Linea : Word; Enfatiza : Boolean);
Var
  Cad     : String;
  Accion  : Byte;
  S1, O1,
  i       : Word;
  code    : Integer;
Begin
  S1 := Seg(DatLib[Numero]^);
  O1 := Ofs(DatLib[Numero]^);
  Accion := Mem[S1:O1];
  Letrero(1,Linea,Espacios(NaC(Numero),3,1),4);
  Case Accion Of
    1 : Begin
          If Enfatiza then
          begin
             Letrero(8,8,'-Nombre-                    ',4);
             Letrero(8,23,'Inserta gráfico PCX',4);
          end;
          Letrero(5, Linea, Espacios(NaC(Mem[S1:O1]),2,1),4);
          Move(Mem[S1:O1+1], Cad, 9);
          Letrero(8, Linea, Espacios(Cad,8,2), 4);
        End;
    4 : Begin
          If Enfatiza then
          begin
             Letrero(8,8,'No. -Nombre-                    ',4);
             Letrero(8,23,'Inserta duende      ',4);
          end;
          Letrero(5, Linea, Espacios(NaC(Mem[S1:O1]),2,1),4);
```

```
        Letrero(8, Linea, Espacios(NaC(Mem[S1:O1+1]),3,1),4);
        Move(Mem[S1:O1+2], Cad, 9);
        Letrero(11, Linea, Espacios(Cad,8,2), 4);
    End;
5 : Begin
        If Enfatiza then
        begin
            Letrero(8,8,'No. -Nombre- Etiqueta  Dnd  X1   Y1',4);
            Letrero(8,23,'Inserta animación  ',4);
        end;
        Letrero(5, Linea, Espacios(NaC(Mem[S1:O1]),2,1),4);
        Letrero(8, Linea, Espacios(NaC(Mem[S1:O1+1]),3,1),4);
        Move(Mem[S1:O1+2], Cad, 9);
        Letrero(11, Linea, Espacios(Cad,8,2), 4);
        Move(Mem[S1:O1+11], Cad, 9);
        Letrero(18, Linea, Espacios(Cad,8,2), 4);
        Letrero(25, Linea, Espacios(NaC(Mem[S1:O1+20]),3,1),4);
        Move(Mem[S1:O1+21],code,2);
        Letrero(28, Linea, Espacios(NaC(code),4,1),4);
        Move(Mem[S1:O1+23],code,2);
        Letrero(32, Linea, Espacios(NaC(code),4,1),4);
    End;
6 : Begin
        If Enfatiza then
        begin
            Letrero(8,8,'No. Anm                              ',4);
            Letrero(8,23,'Inserta objeto      ',4);
        end;
        Letrero(5, Linea, Espacios(NaC(Mem[S1:O1]),2,1),4);
        Letrero(8, Linea, Espacios(NaC(Mem[S1:O1+1]),3,1),4);
        Letrero(11, Linea, Espacios(NaC(Mem[S1:O1+2]),3,1),4);
    End;
7 : Begin
        If Enfatiza then
        begin
            Letrero(8,8,'No. -Nombre- Rep                     ',4);
            Letrero(8,23,'Inserta música FM  ',4);
        end;
        Letrero(5, Linea, Espacios(NaC(Mem[S1:O1]),2,1),4);
        Letrero(8, Linea, Espacios(NaC(Mem[S1:O1+1]),3,1),4);
        Move(Mem[S1:O1+2], Cad, 9);
        Letrero(11, Linea, Espacios(Cad,8,2), 4);
        Letrero(18, Linea, Espacios(NaC(Mem[S1:O1+11]),2,1),4);
    end;
8 : Begin
        If Enfatiza then
        begin
            Letrero(8,8,'No.  X1   Y1   X2   Y2   Anm       ',4);
            Letrero(8,23,'Inserta restricción',4);
        end;
        Letrero(5, Linea, Espacios(NaC(Mem[S1:O1]),2,1),4);
        Letrero(8, Linea, Espacios(NaC(Mem[S1:O1+1]),3,1),4);
        Move(Mem[S1:O1+2],code,2);
        Letrero(11, Linea, Espacios(NaC(code),4,1),4);
        Move(Mem[S1:O1+4],code,2);
        Letrero(15, Linea, Espacios(NaC(code),4,1),4);
        Move(Mem[S1:O1+6],code,2);
        Letrero(19, Linea, Espacios(NaC(code),4,1),4);
        Move(Mem[S1:O1+8],code,2);
        Letrero(23, Linea, Espacios(NaC(code),4,1),4);
        Letrero(27, Linea, Espacios(NaC(Mem[S1:O1+10]),3,1),4);
    End;
```

```
    9 : Begin
          If Enfatiza then
          begin
             Letrero(8,8,'Obj Anm Etiqueta EX1 E2 BN1 B2 BN3 ',4);
             Letrero(8,23,'Inserta Acción Obj.',4);
          end;
          Letrero(5, Linea, Espacios(NaC(Mem[S1:O1]),2,1),4);
          Letrero(8, Linea, Espacios(NaC(Mem[S1:O1+1]),3,1),4);
          Letrero(11, Linea, Espacios(NaC(Mem[S1:O1+2]),3,1),4);
          Move(Mem[S1:O1+3], Cad, 9);
          Letrero(14, Linea, Espacios(Cad,8,2), 4);
          Letrero(21, Linea, Espacios(NaC(Mem[S1:O1+12]),3,1),4);
          Letrero(24, Linea, Espacios(NaC(Mem[S1:O1+13]),2,1),4);
          Letrero(26, Linea, Espacios(NaC(Mem[S1:O1+14]),3,1),4);
          Letrero(29, Linea, Espacios(NaC(Mem[S1:O1+15]),2,1),4);
          Letrero(31, Linea, Espacios(NaC(Mem[S1:O1+16]),3,1),4);
        end;
    Else begin
       If Enfatiza Then Letrero(8,23,'No hay acción definida',4);
    end;
  End;
  If Enfatiza Then
  begin
     Cuadrado2(1*8, Linea*TallaLetraY, 4*8+TallaLetraX-1
               ,Linea*TallaLetraY+7, 3);
     Cuadrado2(5*8, Linea*TallaLetraY, 7*8+TallaLetraX-1
               ,Linea*TallaLetraY+7, 3);
     Cuadrado2(8*8, Linea*TallaLetraY, 35*8+TallaLetraX-1
               ,Linea*TallaLetraY+7, 3);
  end;
End;
```

Inserta una nueva acción.

```
PROCEDURE ReservaAccion(Numero : Byte; Var Accion : Byte; Var Linea : Word);
Var
  i : Word;
  p : Pointer;
Begin
   Inc(TotalLin);
   Accion := Numero;
   Case Accion Of
     1 : Begin
           GetMem(DatLib[TotalLin], 10);
           Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)] := Accion;
           Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+1] := 0;
         End;
     4 : Begin
           GetMem(DatLib[TotalLin], 11);
           Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)] := Accion;
           Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+1] := MaxDuende;
           Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+2] := 0;
           Inc(MaxDuende);
         End;
     5 : Begin
           GetMem(DatLib[TotalLin], 25);
           Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)] := Accion;
           Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+1] := MaxAnima;
           Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+2] := 0;
           Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+11] := 0;
           Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+20] := 0;
           MemW[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+21] := 0;
           MemW[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+23] := 0;
```

```
          Inc(MaxAnima);
        End;
    6 : Begin
          GetMem(DatLib[TotalLin], 3);
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)] := Accion;
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+1] := 0;
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+2] := 0;
        end;
    7 : Begin
          GetMem(DatLib[TotalLin], 12);
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)] := Accion;
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+1] := MaxMusica;
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+2] := 0;
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+11] := 0;
          Inc(MaxMusica);
        End;
    8 : Begin
          GetMem(DatLib[TotalLin], 11);
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)] := Accion;
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+1] := 0;
          MemW[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+2] := 0;
          MemW[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+4] := 0;
          MemW[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+6] := 10;
          MemW[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+8] := 10;
          MemW[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+10] := 0;
        End;
    9 : Begin
          GetMem(DatLib[TotalLin], 17);
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)] := Accion;
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+1] := 0;
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+2] := 0;
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+3] := 0;
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+12] := 0;
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+13] := 0;
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+14] := 0;
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+15] := 0;
          Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+16] := 0;
        End;
  End;
  If Linea+1<TotalLin Then
  begin
    p := DatLib[TotalLin];
    For i := TotalLin DownTo Linea+1 Do DatLib[i] := DatLib[i-1];
    DatLib[Linea+1] := p;
  end;
  Inc(Linea);
End;
```

Borra una acción.

```
PROCEDURE BorraAccion(Numero : Word);
Var
  i, j : Word;
Begin
  If (TotalLin>0) And (Numero In [1..TotalLin]) Then
  begin
    Case Mem[Seg(DatLib[Numero]^):Ofs(DatLib[Numero]^)] Of
      1 : Begin
            FreeMem(DatLib[Numero], 10);
            DatLib[Numero] := Nil;
          End;
      4 : Begin
            j := Mem[Seg(DatLib[Numero]^):Ofs(DatLib[Numero]^)+1];
```

```
            For i := 1 To TotalLin Do
            Begin
             If (Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)]=5) And
                (Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)+20]=j)
                Then Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)+20] := 0;
             If (Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)]=4) And
                (Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)+1]>j)
                Then Dec(Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)+1]);
            End;
            FreeMem(DatLib[Numero], 11);
            DatLib[Numero] := Nil;
            Dec(MaxDuende);
           End;
      5 : Begin
            j := Mem[Seg(DatLib[Numero]^):Ofs(DatLib[Numero]^)+1];
            For i := 1 To TotalLin Do
             If (Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)]=5) And
                (Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)+1]>j)
                Then Dec(Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)+1]);
            FreeMem(DatLib[Numero], 25);
            DatLib[Numero] := Nil;
            Dec(MaxAnima);
           End;
      6 : Begin
            FreeMem(DatLib[Numero], 3);
            DatLib[Numero] := Nil;
           End;
      7 : Begin
            j := Mem[Seg(DatLib[Numero]^):Ofs(DatLib[Numero]^)+1];
            For i := 1 To TotalLin Do
            Begin
             If (Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)]=7) And
                (Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)+1]>j)
                Then Dec(Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)+1]);
            End;
            FreeMem(DatLib[Numero], 12);
            DatLib[Numero] := Nil;
            Dec(MaxMusica);
           End;
      8 : Begin
            FreeMem(DatLib[Numero], 11);
            DatLib[Numero] := Nil;
           End;
      9 : Begin
            FreeMem(DatLib[Numero], 17);
            DatLib[Numero] := Nil;
           End;
    End;
    For i := Numero To TotalLin-1 Do DatLib[i] := DatLib[i+1];
    Dec(TotalLin);
  end;
End;

{$I-}
```

Escribe el libreto en disco (*.LIB).

```
PROCEDURE Salva_Libreto;
Var
   Accion : Byte;
   i, Tot : Word;
   Salir  : Boolean;
   Arch   : File;
```

```
Begin
  Rat.CursorRaton(False);
  If Dir1.Lee_DirArch(LibNom, True) Then
  Begin
     Assign(Arch, LibNom);
     Repeat
       Rewrite(Arch,1);
       If IOResult<>0 Then
       Begin
         Salir := False;
         If Mensaje('Error al leer: '+LibNom+' ¿Reintentar?',2)=2 Then exit;
       End Else Salir := True;
     Until Salir;
     For i := 1 To TotalLin Do
     begin
       Repeat
         Accion := Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)];
         Case Accion Of
           1 : Tot := 10;
           4 : Tot := 11;
           5 : Tot := 25;
           6 : Tot := 3;
           7 : Tot := 12;
           8 : Tot := 11;
           9 : Tot := 17;
           Else Tot := 0;
         End;
         BlockWrite(Arch, Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)], Tot);
         If IOResult<>0 Then
         Begin
           Salir := False;
           If Mensaje('Error al escribir: '+LibNom+' ¿Reintentar?',2)=2
            Then begin
               Close(Arch);  If IOResult=0 Then;
               exit;
            end;
         End Else Salir := True;
       Until Salir;
     end;
  End;
  Close(arch);  If IOResult=0 Then;
  Rat.CursorRaton(True);
End;
```

Lee el libreto de disco (*.LIB).

```
PROCEDURE Lee_Libreto;
Var
   Cad     : String;
   Accion  : Byte;
   Leidos,
   j,
   i, Tot  : Word;
   Pantalla,
   Salir   : Boolean;
   Arch    : File;
Begin
  Rat.CursorRaton(False);
  Pantalla  := False;
  If Dir1.Lee_DirArch(LibNom, False) Then
  Begin
     For i := 1 To TotalLin Do BorraAccion(i);
     TotalLin  := 0;
```

```
MaxDuende := 1;      MaxAnima  := 1;   MaxMusica := 1;
Assign(Arch, LibNom);
Repeat
  Reset(Arch,1);
  If IOResult<>0 Then
  Begin
    Salir := False;
    If Mensaje('Error al escribir: '+LibNom+' ¿Reintentar?',2)=2
    Then exit;
  End Else Salir := True;
Until Salir;
BlockRead(Arch, Accion, 1, Leidos);
j := 0;
While Leidos>0 Do
begin
  Repeat
    ReservaAccion(Accion, Accion, j);
    Case Accion Of
      1 : Tot := 10-1;
      4 : Tot := 11-1;
      5 : Tot := 25-1;
      6 : Tot := 3-1;
      7 : Tot := 12-1;
      8 : Tot := 11-1;
      9 : Tot := 17-1;
      Else Tot := 0;
    End;
    Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)] := Accion;
    BlockRead(Arch, Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+1]
              ,Tot, Leidos);
    Case Accion Of
      1 : If Not Pantalla Then
          begin
            Pantalla := True;
            Move(Mem[Seg(DatLib[TotalLin]^):Ofs(DatLib[TotalLin]^)+1],
                 Cad, 9);
            If (Cad<>'        ') And (Cad<>'') Then
            Begin
              Cad := '..\PCX\'+Cad+'.PCX';
              F.VerPCX(Cad, Pantemporal^);
              Pll.Iguala_Pal(Pantemporal^); }
              EscribeXMS(XMSMan[30], Pantemporal^, 64000, 128000);
            End;
          end;
      4 : If Mem[Seg(DatLib[TotalLin]^):
                 Ofs(DatLib[TotalLin]^)+1]>MaxDuende Then
            MaxDuende := Mem[Seg(DatLib[TotalLin]^):
                             Ofs(DatLib[TotalLin]^)+1];
      5 : If Mem[Seg(DatLib[TotalLin]^):
                 Ofs(DatLib[TotalLin]^)+1]>MaxAnima Then
            MaxAnima := Mem[Seg(DatLib[TotalLin]^):
                            Ofs(DatLib[TotalLin]^)+1]; }
    End;
    If IOResult<>0 Then
    Begin
      Salir := False;
      If Mensaje('Error al leer: '+LibNom+' ¿Reintentar?',2)=2
       Then begin
          Close(Arch);  If IOResult=0 Then;
          exit;
       end;
    End Else Salir := True;
```

```
      Until Salir;
      BlockRead(Arch, Accion, 1, Leidos);
    End;
  End;
  Close(arch);  If IOResult=0 Then;
  Rat.CursorRaton(True);
End;
{$I+}
```

Regresa únicamente el nombre del archivo de una cadena, eliminando la via de acceso.

```
FUNCTION NomArch(Nombre : String) : String;
Var
  i   : Byte;
  Cad : String;
Begin
  i := Length(Nombre);
  Cad := '';
  While (i>0) And (Not (Nombre[i] In ['\',':'])) Do
  begin
    Cad := Nombre[i]+Cad; Dec(i);
  end;
  NomArch := Cad;
End;
```

Procedimiento principal para captura/edición de acciones.

```
PROCEDURE Principal.Crea_libreto;
Var
   Col, Accion   : Byte;
   Tecla, Tecla2 : Char;
   i, Ini, Act,
   Boton, Xr, Yr : Word;
   xi, yi, int1,
   xi1, yi1,
   xi2, yi2      : Integer;
   Nombre, Cad   : String;
   Iconos, Mono  : Duende;

   Procedure Redibuja;
   Begin
     LeeXMS(XMSMan[30], PanTemporal^, 64000, 0);
{     Despliega(PanTemporal); }
     Dnd1.PutDuende(226+24*2,22,Iconos,pantemporal^,5);  { Insertar          }
{     Cuadrado(297, 178,  319, 199, 1); }
     Dnd1.PutDuende(298,179,Iconos,pantemporal^,1);      { Puerta de Salida }
     Dnd1.PutDuende(226,22,Iconos,pantemporal^,11);      { Leer libreto     }
   End;
Begin
  Iconos.Total := 0;   Mono.Total := 0;
{ Columna Numeros    = 1       Renglones = 10..21
  Columna Acción     = 5       Renglones = 10..21
  Columna Parametros = 8       Renglones = 10..21
  Columna Nom. Arch. = 27      Renglon   = 1
  Columna Expl. Par. = 8       Renglon   = 8
  Columna Estado     = 23      Renglon   = 8  }
  Dnd1.LeerDuende(Iconos, 'LIBRETO.DND');
{  Redibuja;}
  Nombre := '*.*';
  Ini := 1;  Act := 0;  Accion := 0;  Col := 1;
  Rat.CursorRaton(True);
  Repeat
```

```
    DestinoLet := Pantemporal;
    Redibuja;
    i := 0;
    Rat.CursorRaton(False);
    Cuadrado(8*TallaLetraX, 10*TallaLetraX
             ,35*TallaLetraX-1, 21*TallaLetraX+7,2);
    Cuadrado(1*TallaLetraX, 10*TallaLetraX
             ,4*TallaLetraX-1, 21*TallaLetraX+7,2);
    Cuadrado(5*TallaLetraX, 10*TallaLetraX
             ,7*TallaLetraX-1, 21*TallaLetraX+7,2);
    Cuadrado(297, 22,  319, 177, 1);
    Cuadrado(250, 22,  273,  42, 1); }
    While (i<12) And (Ini+i<=TotalLin) Do
    begin
      EscribeParam(Col, Ini+i, i+10, (Ini+i=Act)];  Inc(i);
    end;
    Letrero(27,1,Espacios(NomArch(LibNom),12,2),4);
    If (TotalLin>0) And (Accion<>255) Then
      Accion := Mem[Seg(DatLib[Act]^):Ofs(DatLib[Act]^)]
    Else Accion := 0;
    Case Accion Of
      0 : Begin
            For xr := 0 to 3 do
              For yr := 0 to 7 do
                Dndl.PutDuende(265+xr*20,48+yr*16,Iconos,pantemporal^,25);
            Dndl.PutDuende(298,50,Iconos,pantemporal^,13);
            Dndl.PutDuende(298,70,Iconos,pantemporal^,15);
            Dndl.PutDuende(298,90,Iconos,pantemporal^,19);
            Dndl.PutDuende(298,110,Iconos,pantemporal^,21);
            Dndl.PutDuende(298,130,Iconos,pantemporal^,23);
            Dndl.PutDuende(275,50,Iconos,pantemporal^,26);
            Dndl.PutDuende(275,70,Iconos,pantemporal^,30);
            If TotalLin>0 Then Dndl.PutDuende(298,155,Iconos,pantemporal^,17);
          End;
      1 : Dndl.PutDuende(298,50,Iconos,pantemporal^,3);
      4 : Dndl.PutDuende(298,50,Iconos,pantemporal^,15);
      5 : Dndl.PutDuende(298,50,Iconos,pantemporal^,19);
      7 : Dndl.PutDuende(298,50,Iconos,pantemporal^,23);
      8 : Dndl.PutDuende(298,50,Iconos,pantemporal^,21);
    End;
    If Accion>0 Then
    begin
      Dndl.PutDuende(226+24*3,22,Iconos,pantemporal^,7);   { Borrar      }
      Dndl.PutDuende(226+24*1,22,Iconos,pantemporal^,9);   { Salvar Libreto }
    end;
    Letrero(1,23,Espacios(Nac(MaxAvail),6,2)+' B',3);
    Despliega(Pantemporal);
    If TotalLin>0 Then
    begin
       Cuadrado2(1*8, (Act-Ini+10)*TallaLetraY, 4*8+TallaLetraX-1
                ,(Act-Ini+10)*TallaLetraY+7, 3);
       Cuadrado2(5*8, (Act-Ini+10)*TallaLetraY, 7*8+TallaLetraX-1
                ,(Act-Ini+10)*TallaLetraY+7, 3);
       Cuadrado2(8*8, (Act-Ini+10)*TallaLetraY, 35*8+TallaLetraX-1
                ,(Act-Ini+10)*TallaLetraY+7, 3);
    end;
    DestinoLet := Ptr($A000,0);
    Rat.CursorRaton(True);
    Repeat
       Tecla := #1;  Tecla2 := #0; Boton := 0;
       if keypressed then
       begin
```

```
    tecla := readkey;
    if tecla=#0 then tecla2 := readkey;
  end; }

  Case Accion Of
    0 : Begin
          If Dnd1.Checa_boton(298,50,iconos,13,'p') then
          begin
            ReservaAccion(1, Accion, Act);
            Tecla2 := #3;
          end;
          If Dnd1.Checa_boton(298,70,iconos,15,'d') then
          begin
            ReservaAccion(4, Accion, Act);
            Tecla2 := #3;
          end;
          If Dnd1.Checa_boton(298,90,iconos,19,'a') then
          begin
            ReservaAccion(5, Accion, Act);
            Tecla2 := #3;
          end;
          If Dnd1.Checa_boton(275,50,iconos,26,'o') then
          begin
            ReservaAccion(6, Accion, Act);
            Tecla2 := #3;
          end;
          If Dnd1.Checa_boton(298,130,iconos,23,'m') then
          begin
            ReservaAccion(7, Accion, Act);
            Tecla2 := #3;
          end;
          If Dnd1.Checa_boton(298,110,iconos,21,'r') then
          begin
            ReservaAccion(8, Accion, Act);
            Tecla2 := #3;
          end;
          If Dnd1.Checa_boton(275,70,iconos,30,'u') then
          begin
            ReservaAccion(9, Accion, Act);
            Tecla2 := #3;
          end;
        End;
    1 : Begin
          Col := 1;
          Move(Mem[Seg(DatLib[Act]^):Ofs(DatLib[Act]^)+1], Cad, 9);
          LeeDato(Cad,0,8,Act-Ini+10,8,Tecla,Tecla2,Boton,Xr,Yr,4);
          If Tecla<>#27 Then
            Move(Cad, Mem[Seg(DatLib[Act]^): Ofs(DatLib[Act]^)+1], 9);
          If (Dnd1.Checa_boton(298,50,iconos,3,#2)) Or (Tecla=#4) Then
          begin
            While (Cad<>'') And (Cad[Length(Cad)]=' ') Do
              Cad := Copy(Cad,1,Length(Cad)-1);
            If Cad<>'' Then Nombre := Cad Else Nombre := '*';
            Nombre := '..\PCX\'+Nombre+'.PCX';
            Rat.CursorRaton(False);
            If Dir1.Lee_DirArch(Nombre, False) Then
            begin
              Cad := NomArch(Nombre);
              Cad := Copy(Cad,1,Length(Cad)-4);
              Move(Cad, Mem[Seg(DatLib[Act]^): Ofs(DatLib[Act]^)+1],9);
              F.VerPCX(Nombre, PanTemporal^);
               Pl1.Iguala_Pal(Pantemporal^); }
```

```
            EscribeXMS(XMSMan[30], Pantemporal^, 64000, 128000);
            Despliega(Pantemporal);
            Boton := 0;
            Repeat
              Rat.PosRaton(Boton,xr,yr);
              If keypressed Then Boton := 1;
            Until Boton<>0;
            Repeat
              Rat.PosRaton(Boton,xr,yr);
              While keypressed Do Readkey;
            Until Boton=0;
            Tecla2 := #2;
          end;
          Rat.CursorRaton(True);
        end;
      End;
  4 : Begin
        If Col>1 Then Col := 1;
        Case Col Of
          1 : Begin
                Move(Mem[Seg(DatLib[Act]^):
                         Ofs(DatLib[Act]^)+2], Cad, 9);
              LeeDato(Cad,0,11,Act-Ini+10,8,Tecla,Tecla2
                      ,Boton,Xr,Yr,4);
              If Tecla<>#27 Then
                Move(Cad,Mem[Seg(DatLib[Act]^):
                     Ofs(DatLib[Act]^)+2],9);
            end;
        End;
        If (Dndl.Checa_boton(298,50,iconos,15,#2)) Or (Tecla=#4) then
        begin
          Move(Mem[Seg(DatLib[Act]^):Ofs(DatLib[Act]^)+2], Cad, 9);
          While (Cad<>'') And (Cad[Length(Cad)]=' ') Do
            Cad := Copy(Cad,1,Length(Cad)-1);
          If Cad='' Then Nombre := '*' Else Nombre := Cad;
          Nombre := '..\DND\'+Nombre+'.DND';
          Rat.CursorRaton(False);
          If Dir1.Lee_DirArch(Nombre, False) Then
          begin
            Cad := NomArch(Nombre);
            Cad := Copy(Cad,1,Length(Cad)-4);
            Move(Cad, Mem[Seg(DatLib[Act]^): Ofs(DatLib[Act]^)+2], 9);
            Dndl.LeerDuende(Mono, Nombre);
            If Mono.Total>0 Then
               Dndl.RellenoDnd(Mono, 1, Mem[$A000:0]);
            Boton := 0;
            Repeat
              Rat.PosRaton(Boton,xr,yr);
              If keypressed Then Boton := 1;
            Until Boton<>0;
            Repeat
              Rat.PosRaton(Boton,xr,yr);
              While keypressed Do Readkey;
            Until Boton=0;
            Tecla2 := #2;
          End;
          Rat.CursorRaton(True);
        end;
        If Boton=1 Then
        Begin
          If Rat.RangoRaton(12*8, (Act-Ini+10)*TallaLetraY
             ,20*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
```

```
            Then Col := 1;
          End;
        End;
    5 : Begin
          If Col>6 Then Col := 1;
          If Col<2 Then Col := 2;
          Case Col Of
            1 : Begin
                  Intl := 0;
                  Move(Mem[Seg(DatLib[Act]^):
                          Ofs(DatLib[Act]^)+1],Intl,1);
                  LeeDato(Intl,2,8,Act-Ini+10,3,Tecla,Tecla2
                          ,Boton,Xr,Yr,4);
                  If Tecla<>#27 Then
                    Move(Intl,Mem[Seg(DatLib[Act]^):
                                  Ofs(DatLib[Act]^)+1], 1);
                End;        }
            2 : Begin
                  Move(Mem[Seg(DatLib[Act]^):
                       Ofs(DatLib[Act]^)+2],Cad,9);
                  LeeDato(Cad,0,11,Act-Ini+10,8,Tecla,Tecla2
                          ,Boton,Xr,Yr,4);
                  If Tecla<>#27 Then
                    Move(Cad, Mem[Seg(DatLib[Act]^):
                         Ofs(DatLib[Act]^)+2], 9);
                end;
            3 : Begin
                  Move(Mem[Seg(DatLib[Act]^):
                       Ofs(DatLib[Act]^)+11], Cad, 9);
                  LeeDato(Cad,0,18,Act-Ini+10,8,Tecla,Tecla2
                       ,Boton,Xr,Yr,4);
                  If Tecla<>#27 Then
                    Move(Cad, Mem[Seg(DatLib[Act]^):
                         Ofs(DatLib[Act]^)+11], 9);
                end;
            4 : Begin
                  Intl := 0;
                  Move(Mem[Seg(DatLib[Act]^):
                       Ofs(DatLib[Act]^)+20], Intl, 1);
                  LeeDato(Intl,2,25,Act-Ini+10,3,Tecla,Tecla2
                       ,Boton,Xr,Yr,4);
                  If Tecla<>#27 Then
                    Move(Intl,Mem[Seg(DatLib[Act]^):
                         Ofs(DatLib[Act]^)+20], 1);
                End;
            5 : Begin
                  Intl := 0;
                  Move(Mem[Seg(DatLib[Act]^):
                       Ofs(DatLib[Act]^)+21], Intl, 2);
                  LeeDato(Intl,2,28,Act-Ini+10,4,Tecla,Tecla2
                       ,Boton,Xr,Yr,4);
                  If Tecla<>#27 Then
                    Move(Intl,Mem[Seg(DatLib[Act]^):
                         Ofs(DatLib[Act]^)+21], 2);
                End;
            6 : Begin
                  Intl := 0;
                  Move(Mem[Seg(DatLib[Act]^):
                       Ofs(DatLib[Act]^)+23], Intl, 2);
                  LeeDato(Intl,2,32,Act-Ini+10,4,Tecla,Tecla2
                       ,Boton,Xr,Yr,4);
                  If Tecla<>#27 Then
```

```
                    Move(Int1,Mem[Seg(DatLib[Act]^):
                      Ofs(DatLib[Act]^)+23], 2);
              End;
        End;
        If (Dnd1.Checa_boton(298,50,iconos,19,#4)) then
        Begin
          Move(Mem[Seg(DatLib[Act]^):Ofs(DatLib[Act]^)+2], Cad, 9);
          While (Cad<>'') And (Cad[Length(Cad)]=' ') Do
            Cad := Copy(Cad,1,Length(Cad)-1);
          If Cad='' Then Nombre := '*' Else Nombre := Cad;
          Nombre := '..\ANM\'+Nombre+'.ANM';
          Rat.CursorRaton(False);
          If Dir1.Lee_DirArch(Nombre, False) Then
          begin
            Cad := NomArch(Nombre);
            Cad := Copy(Cad,1,Length(Cad)-4);
            Move(Cad, Mem[Seg(DatLib[Act]^): Ofs(DatLib[Act]^)+2], 9);
            Int1 := 0;
            If MaxDuende>1 Then
              For i := 1 To TotalLin Do
               If (Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)]=4) And
                  (Mem[Seg(DatLib[Act]^):Ofs(DatLib[Act]^)+20]=
                   Mem[Seg(DatLib[i]^): Ofs(DatLib[i]^)+1])
                   Then Int1 := i;
            If Int1<>0 Then
            Begin
              Move(Mem[Seg(DatLib[Int1]^):
                       Ofs(DatLib[Int1]^)+2] ,Cad ,9);
              While (Cad<>'') And (Cad[Length(Cad)]=' ') Do
                Cad := Copy(Cad,1,Length(Cad)-1);
              If Cad='' Then Nombre := '*' Else Nombre := Cad;
              Nombre := '..\DND\'+Nombre+'.DND';
              If Dir1.Lee_DirArch(Nombre, False) Then
              begin
                Cad := NomArch(Nombre);
                Cad := Copy(Cad,1,Length(Cad)-4);
                Move(Cad, Mem[Seg(DatLib[Int1]^):
                             Ofs(DatLib[Int1]^)+2], 9);
                Dnd1.LeerDuende(Mono, Nombre);
                Move(Mem[Seg(DatLib[Act]^):
                         Ofs(DatLib[Act]^)+21], xi, 2);
                Move(Mem[Seg(DatLib[Act]^):
                         Ofs(DatLib[Act]^)+23], yi, 2);
                If (xi<0) Or (xi>319) Then xi := 0;
                If (yi<0) Or (yi>1999) Then yi := 0;
                LeeXMS(XMSMan[30], Pantemporal^, 64000, 128000);
                Boton := 0;       Rat.IniciaRaton(xi, yi);
                Despliega(Pantemporal);
                 Move(Mem[$A000:0], Pantemporal^, 64000); }
                Dnd1.PutDuende(xi,yi,Mono,Mem[$A000:0],1);
                Repeat
                  For i := 1 To TotalLin Do
                    If (Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)]=8)
                    Then
                    begin
                      Move(Mem[Seg(DatLib[i]^):
                               Ofs(DatLib[i]^)+2], xi1, 2);
                      Move(Mem[Seg(DatLib[i]^):
                               Ofs(DatLib[i]^)+4], yi1, 2);
                      Move(Mem[Seg(DatLib[i]^):
                               Ofs(DatLib[i]^)+6], xi2, 2);
                      Move(Mem[Seg(DatLib[i]^):
```

```
                                Ofs(DatLib[i]^)+8], yi2, 2);
                        Cuadro(xi1,yi1,xi2,yi2,5);
                      end;
                    Anml.Ubica_Duende(Boton, xi, yi, mono, 1);
                  Until Boton In [1,2];
                  If Boton=1 Then
                  begin
                    Move(xi,Mem[Seg(DatLib[Act]^):
                                Ofs(DatLib[Act]^)+21], 2);
                    Move(yi,Mem[Seg(DatLib[Act]^):
                                Ofs(DatLib[Act]^)+23], 2);
                  end;
                End;
              End;
              Tecla2 := #2;  Boton := 0;
            End;
            Rat.CursorRaton(True);
          End;
          If Odd(Boton) Then
          Begin
            If Rat.RangoRaton(11*8, (Act-Ini+10)*TallaLetraY
              ,20*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
              Then Col := 2;
            If Rat.RangoRaton(18*8, (Act-Ini+10)*TallaLetraY
              ,29*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
              Then Col := 3;
            If Rat.RangoRaton(25*8, (Act-Ini+10)*TallaLetraY
              ,32*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
              Then Col := 4;
            If Rat.RangoRaton(28*8, (Act-Ini+10)*TallaLetraY
              ,31*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
              Then Col := 5;
            If Rat.RangoRaton(32*8, (Act-Ini+10)*TallaLetraY
              ,35*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
              Then Col := 6;
          End;
        End;
6 :     Begin
          If Col>2 Then Col := 1;
          Case Col Of
            1 : Begin
                  Int1 := 0;
                  Move(Mem[Seg(DatLib[Act]^):
                           Ofs(DatLib[Act]^)+1], Int1, 1);
                  LeeDato(Int1,1,8,Act-Ini+10,3,Tecla,Tecla2
                          ,Boton,Xr,Yr,4);
                  If Tecla<>#27 Then
                    Move(Int1,Mem[Seg(DatLib[Act]^):
                                  Ofs(DatLib[Act]^)+1], 1);
                End;
            2 : Begin
                  Int1 := 0;
                  Move(Mem[Seg(DatLib[Act]^):
                           Ofs(DatLib[Act]^)+2], Int1, 1);
                  LeeDato(Int1,1,11,Act-Ini+10,3,Tecla,Tecla2
                          ,Boton,Xr,Yr,4);
                  If Tecla<>#27 Then
                    Move(Int1,Mem[Seg(DatLib[Act]^):
                                  Ofs(DatLib[Act]^)+2], 2);
                End;
          End;
          If Odd(Boton) Then
```

```
        Begin
          If Rat.RangoRaton(8*8,(Act-Ini+10)*TallaLetraY
              ,11*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
            Then Col := 1;
          If Rat.RangoRaton(11*8,(Act-Ini+10)*TallaLetraY
              ,15*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
            Then Col := 2;
        End;
      End;
    7 : Begin
        If Col>2 Then Col := 1;
        Case Col Of
          1 : Begin
                Move(Mem[Seg(DatLib[Act]^):
                         Ofs(DatLib[Act]^)+2], Cad, 9);
                LeeDato(Cad,0,11,Act-Ini+10,8,Tecla,Tecla2
                        ,Boton,Xr,Yr,4);
                If Tecla<>#27 Then
                  Move(Cad, Mem[Seg(DatLib[Act]^):
                                Ofs(DatLib[Act]^)+2], 9);
              end;
          2 : Begin
                Int1 := 0;
                Move(Mem[Seg(DatLib[Act]^):
                         Ofs(DatLib[Act]^)+11], Int1, 1);
                LeeDato(Int1,1,18,Act-Ini+10,2,Tecla,Tecla2,
                        Boton,Xr,Yr,4);
                If Tecla<>#27 Then
                  Move(Int1,Mem[Seg(DatLib[Act]^):
                                Ofs(DatLib[Act]^)+11], 1);
              End;
        End;
        If (Dnd1.Checa_boton(298,50,iconos,23,#4)) then
        begin
          Move(Mem[Seg(DatLib[Act]^):Ofs(DatLib[Act]^)+2], Cad, 9);
          While (Cad<>'') And (Cad[Length(Cad)]=' ') Do
            Cad := Copy(Cad,1,Length(Cad)-1);
          If Cad='' Then Nombre := '*' Else Nombre := Cad;
          Nombre := '..\sb\'+Nombre+'.CMF';
          Rat.CursorRaton(False);
          If Dir1.Lee_DirArch(Nombre, False) Then
          begin
            Cad := NomArch(Nombre);
            Cad := Copy(Cad,1,Length(Cad)-4);
            Move(Cad, Mem[Seg(DatLib[Act]^):
                          Ofs(DatLib[Act]^)+2], 9);
            If Lee_FM(Nombre) Then
            Begin
              Toca_FM;
              Mensaje('Tocando: '+Nombre+' ¿Detener música?',1);
              Deten_FM;
            End;
            Tecla2 := #2;
          End;
          Rat.CursorRaton(True);
        end;
        If Boton=1 Then
        Begin
          If Rat.RangoRaton(11*8,(Act-Ini+10)*TallaLetraY
              ,17*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
            Then Col := 1;
          If Rat.RangoRaton(18*8, (Act-Ini+10)*TallaLetraY
```

```
                ,19*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
            Then Col := 2;
        End;
      End;
8 : Begin
      If Col>6 Then Col := 1;
      Case Col Of
        1 : Begin
              Int1 := 0;
              Move(Mem[Seg(DatLib[Act]^):
                       Ofs(DatLib[Act]^)+1], Int1, 1);
              LeeDato(Int1,1,8,Act-Ini+10,3,Tecla,Tecla2
                      ,Boton,Xr,Yr,4);
              If Tecla<>#27 Then
                Move(Int1,Mem[Seg(DatLib[Act]^):
                              Ofs(DatLib[Act]^)+1], 1);
            End;
        2 : Begin
              Int1 := 0;
              Move(Mem[Seg(DatLib[Act]^):
                       Ofs(DatLib[Act]^)+2], Int1, 2);
              LeeDato(Int1,2,11,Act-Ini+10,4,Tecla,Tecla2
                      ,Boton,Xr,Yr,4);
              If Tecla<>#27 Then
                Move(Int1,Mem[Seg(DatLib[Act]^):
                              Ofs(DatLib[Act]^)+2], 2);
            End;
        3 : Begin
              Int1 := 0;
              Move(Mem[Seg(DatLib[Act]^):
                       Ofs(DatLib[Act]^)+4], Int1, 2);
              LeeDato(Int1,2,15,Act-Ini+10,4,Tecla,Tecla2
                      ,Boton,Xr,Yr,4);
              If Tecla<>#27 Then
                Move(Int1,Mem[Seg(DatLib[Act]^):
                              Ofs(DatLib[Act]^)+4], 2);
            End;
        4 : Begin
              Int1 := 0;
              Move(Mem[Seg(DatLib[Act]^):
                       Ofs(DatLib[Act]^)+6], Int1, 2);
              LeeDato(Int1,2,19,Act-Ini+10,4,Tecla,Tecla2
                      ,Boton,Xr,Yr,4);
              If Tecla<>#27 Then
                Move(Int1,Mem[Seg(DatLib[Act]^):
                              Ofs(DatLib[Act]^)+6], 2);
            End;
        5 : Begin
              Int1 := 0;
              Move(Mem[Seg(DatLib[Act]^):
                       Ofs(DatLib[Act]^)+8], Int1, 2);
              LeeDato(Int1,2,23,Act-Ini+10,4,Tecla,Tecla2
                      ,Boton,Xr,Yr,4);
              If Tecla<>#27 Then
                Move(Int1,Mem[Seg(DatLib[Act]^):
                              Ofs(DatLib[Act]^)+8], 2);
            End;
        6 : Begin
              Int1 := 0;
              Move(Mem[Seg(DatLib[Act]^):
                       Ofs(DatLib[Act]^)+10], Int1, 1);
              LeeDato(Int1,1,27,Act-Ini+10,3,Tecla,Tecla2
```

```
                    ,Boton,Xr,Yr,4);
          If Tecla<>#27 Then
            Move(Int1,Mem[Seg(DatLib[Act]^):
                          Ofs(DatLib[Act]^)+10], 1);
        End;
    End;
    If (Dnd1.Checa_boton(298,50,iconos,21,#4)) then
    Begin
      Rat.CursorRaton(False);
      LeeXMS(XMSMan[30], Pantemporal^, 64000, 128000);
      Despliega(Pantemporal);
      For i := 1 To TotalLin Do
        If (Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)]=8) And (i<>Act)
        Then
        begin
          Move(Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)+2], xi, 2);
          Move(Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)+4], yi, 2);
          Move(Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)+6], xi2, 2);
          Move(Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)+8], yi2, 2);
          If Mem[Seg(DatLib[i]^):Ofs(DatLib[i]^)+1]=0 Then
            Cuadro(xi,yi,xi2,yi2,255)
          Else
            Cuadro(xi,yi,xi2,yi2,5);
        end;
      Move(Mem[$A000:0], Pantemporal^, 64000);
      Move(Mem[Seg(DatLib[Act]^):Ofs(DatLib[Act]^)+2], xi, 2);
      Move(Mem[Seg(DatLib[Act]^):Ofs(DatLib[Act]^)+4], yi, 2);
      Move(Mem[Seg(DatLib[Act]^):Ofs(DatLib[Act]^)+6], xi2, 2);
      Move(Mem[Seg(DatLib[Act]^):Ofs(DatLib[Act]^)+8], yi2, 2);
      Rat.IniciaRaton(xi, yi);
      If xi<0  Then xi  := 0;
      If xi2<0 Then xi2 := 0;
      If yi<0  Then yi  := 0;
      If yi2<0 Then yi2 := 0;
      xi2 := xi2-xi;     yi2 := yi2-yi;
      Rectangulo(Boton, xi, yi, xi2, yi2);
      If Boton=4 Then
      begin
        xi2 := xi2+xi;     yi2 := yi2+yi;
        If yi>yi2 Then
        begin
          int1 := yi; yi := yi2;  yi2 := Int1;
        end;
        If xi>xi2 Then
        begin
          int1 := xi; xi := xi2;  xi2 := Int1;
        end;
        Move(xi,  Mem[Seg(DatLib[Act]^):Ofs(DatLib[Act]^)+2], 2);
        Move(yi,  Mem[Seg(DatLib[Act]^):Ofs(DatLib[Act]^)+4], 2);
        Move(xi2, Mem[Seg(DatLib[Act]^):Ofs(DatLib[Act]^)+6], 2);
        Move(yi2, Mem[Seg(DatLib[Act]^):Ofs(DatLib[Act]^)+8], 2);
      end;
      Rat.CursorRaton(True);
      Tecla2 := #2;
    End;
    If Odd(Boton) Then
    Begin
      If Rat.RangoRaton(8*8,  (Act-Ini+10)*TallaLetraY
        ,11*8+TallaLetraX-1,  (Act-Ini+10)*TallaLetraY+7)
        Then Col := 1;
      If Rat.RangoRaton(11*8,  (Act-Ini+10)*TallaLetraY
        ,15*8+TallaLetraX-1,  (Act-Ini+10)*TallaLetraY+7)
```

```
          Then Col := 2;
        If Rat.RangoRaton(15*8, (Act-Ini+10)*TallaLetraY
          ,19*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
          Then Col := 3;
        If Rat.RangoRaton(19*8, (Act-Ini+10)*TallaLetraY
          ,23*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
          Then Col := 4;
        If Rat.RangoRaton(23*8, (Act-Ini+10)*TallaLetraY
          ,27*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
          Then Col := 5;
        If Rat.RangoRaton(27*8, (Act-Ini+10)*TallaLetraY
          ,31*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
          Then Col := 6;
      End;
    End;
9 : Begin
      If Col>8 Then Col := 1;
      Case Col Of
        1 : Begin
              Int1 := 0;
              Move(Mem[Seg(DatLib[Act]^):
                      Ofs(DatLib[Act]^)+1], Int1, 1);
              LeeDato(Int1,1,8,Act-Ini+10,3,Tecla,Tecla2
                      ,Boton,Xr,Yr,4);
              If Tecla<>#27 Then
                Move(Int1,Mem[Seg(DatLib[Act]^):
                              Ofs(DatLib[Act]^)+1], 1);
            End;
        2 : Begin
              Int1 := 0;
              Move(Mem[Seg(DatLib[Act]^):
                      Ofs(DatLib[Act]^)+2], Int1, 1);
              LeeDato(Int1,1,11,Act-Ini+10,3,Tecla,Tecla2
                      ,Boton,Xr,Yr,4);
              If Tecla<>#27 Then
                Move(Int1,Mem[Seg(DatLib[Act]^):
                              Ofs(DatLib[Act]^)+2], 1);
            End;
        3 : Begin
              Move(Mem[Seg(DatLib[Act]^):
                      Ofs(DatLib[Act]^)+3], Cad, 9);
              LeeDato(Cad,0,14,Act-Ini+10,8,Tecla,Tecla2
                      ,Boton,Xr,Yr,4);
              If Tecla<>#27 Then
                Move(Cad, Mem[Seg(DatLib[Act]^):
                              Ofs(DatLib[Act]^)+3], 9);
            end;
        4 : Begin
              Int1 := 0;
              Move(Mem[Seg(DatLib[Act]^):
                              Ofs(DatLib[Act]^)+12], Int1, 1);
              LeeDato(Int1,1,21,Act-Ini+10,3,Tecla,Tecla2
                      ,Boton,Xr,Yr,4);
              If Tecla<>#27 Then
                Move(Int1,Mem[Seg(DatLib[Act]^):
                              Ofs(DatLib[Act]^)+12], 1);
            End;
        5 : Begin
              Int1 := 0;
              Move(Mem[Seg(DatLib[Act]^):
                      Ofs(DatLib[Act]^)+13], Int1, 1);
              LeeDato(Int1,1,24,Act-Ini+10,2,Tecla,Tecla2
```

```
                    ,Boton,Xr,Yr,4);
            If Tecla<>#27 Then
              Move(Int1,Mem[Seg(DatLib[Act]^):
                           Ofs(DatLib[Act]^)+13], 1);
          End;
      6 : Begin
            Int1 := 0;
            Move(Mem[Seg(DatLib[Act]^):
                     Ofs(DatLib[Act]^)+14], Int1, 1);
            LeeDato(Int1,1,26,Act-Ini+10,3,Tecla,Tecla2
                    ,Boton,Xr,Yr,4);
            If Tecla<>#27 Then
              Move(Int1,Mem[Seg(DatLib[Act]^):
                           Ofs(DatLib[Act]^)+14], 1);
          End;
      7 : Begin
            Int1 := 0;
            Move(Mem[Seg(DatLib[Act]^):
                     Ofs(DatLib[Act]^)+15], Int1, 1);
            LeeDato(Int1,1,29,Act-Ini+10,2,Tecla,Tecla2
                    ,Boton,Xr,Yr,4);
            If Tecla<>#27 Then
              Move(Int1,Mem[Seg(DatLib[Act]^):
                           Ofs(DatLib[Act]^)+15], 1);
          End;
      8 : Begin
            Int1 := 0;
            Move(Mem[Seg(DatLib[Act]^):
                     Ofs(DatLib[Act]^)+16], Int1, 1);
            LeeDato(Int1,1,31,Act-Ini+10,3,Tecla,Tecla2
                    ,Boton,Xr,Yr,4);
            If Tecla<>#27 Then
              Move(Int1,Mem[Seg(DatLib[Act]^):
                           Ofs(DatLib[Act]^)+16], 1);
          End;
    End;
    If Boton=1 Then
    Begin
      If Rat.RangoRaton(8*8, (Act-Ini+10)*TallaLetraY
        ,11*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
        Then Col := 1;
      If Rat.RangoRaton(11*8, (Act-Ini+10)*TallaLetraY
        ,14*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
        Then Col := 2;
      If Rat.RangoRaton(14*8, (Act-Ini+10)*TallaLetraY
        ,21*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
        Then Col := 3;
      If Rat.RangoRaton(21*8, (Act-Ini+10)*TallaLetraY
        ,24*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
        Then Col := 4;
      If Rat.RangoRaton(24*8, (Act-Ini+10)*TallaLetraY
        ,26*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
        Then Col := 5;
      If Rat.RangoRaton(26*8, (Act-Ini+10)*TallaLetraY
        ,29*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
        Then Col := 6;
      If Rat.RangoRaton(29*8, (Act-Ini+10)*TallaLetraY
        ,31*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
        Then Col := 7;
      If Rat.RangoRaton(31*8, (Act-Ini+10)*TallaLetraY
        ,34*8+TallaLetraX-1, (Act-Ini+10)*TallaLetraY+7)
        Then Col := 8;
```

```
        End;
      End;
End;

If Dnd1.Checa_boton(298,179,iconos,1,#27) then Tecla := #27;
If (Dnd1.Checa_boton(226,22,iconos,11,'l')) Or (Tecla=#12) then
Begin
   Lee_Libreto;  Tecla2 := #3; { redibuja; }
   Act := TotalLin;
End;
If Dnd1.Checa_boton(274,22,iconos,5,#2) then
begin
  Tecla := #0;  Tecla2 := #82;
end;
If Accion>0 Then
Begin
  If (Dnd1.Checa_boton(250,22,iconos,9,#2)) Or
     (Tecla=#19) Then Salva_Libreto;
  If Dnd1.Checa_boton(298,22,iconos,7,#2) then
  begin
    Tecla := #0;  Tecla2 := #83;
  end;
End
Else
  If TotalLin>0 Then
    If Dnd1.Checa_boton(298,155,iconos,17,#8) then Tecla2 := #2;
If Boton=1 Then
Begin
  If (Xr>=8*8) And (Xr<32*8) And (Yr>=10*8) And (Yr<22*8) Then
  begin
    i := (Yr Div 8)-10;
    If Ini+i<=TotalLin Then
    begin
      Act := Ini+i;  Tecla2 := #2;
    End;
  end;
  If Rat.RangoRaton(1*8,73,32*8+TallaLetraX-1,76) Then
  begin
    tecla := #0;  Tecla2 := #73;
  end;
  If Rat.RangoRaton(1*8,179,32*8+TallaLetraX-1,181) Then
  begin
    tecla := #0;  Tecla2 := #81;
  end;
End;
Case Tecla Of
  #0  : Begin
          Case Tecla2 of
            #71 : If Act>1 Then Act := 1 Else Tecla2 := #0;
            #72 : If Act>1 Then Dec(Act) Else Tecla2 := #0;
            #73 : If Act>1 Then
                  begin
                    If Act>12 Then Dec(Act,12) Else Act := 1;
                  end Else Tecla2 := #0;
            #79 : If Act<TotalLin Then Act := TotalLin;
            #80 : If Act<TotalLin Then Inc(Act) Else Tecla2 := #0;
            #81 : If Act<TotalLin Then
                  begin
                    If Act+12<=TotalLin Then Inc(Act,12)
                    Else Act := TotalLin;
                  end Else Tecla2 := #0;
            #82 : Accion := 255;
```

```
                #83 : BorraAccion(Act);
                #15 : If Col>1 Then Dec(Col);
              end;
            End;
        #9 : Inc(Col);
        #27 : Begin
                Rat.CursorRaton(False);
                If Mensaje('¿Salir del Libreto?',2)=2 Then Tecla := #1;
                Rat.CursorRaton(True);
              end;
      End;
  Until (Tecla In [#27]) Or (Tecla2 In [#2,#3,#71..#73,#79..#81,#82,#83]);
  If Act>TotalLin Then Act := TotalLin;
{   If Tecla2=#3 Then Act := TotalLin; }
  If (Act>0) And (Ini>Act) Then Ini := Act;
  If (Act>11) And ((Act-11)>Ini) Then Ini := Act-11;
Until Tecla=#27;
Rat.CursorRaton(False);
End;
```

RUTINA PRINCIPAL.

```
Begin
  P.inicio('..\ejecutor\SinNombr.LIB');
  P.Crea_libreto;
  P.final;
End.
```

Acción	Procedimiento	Parámetros	Bytes totales
1	Inserta_pcx	nombre de archivo	10
4	Inserta_Duende	número, archivo	11
5	Inserta_Anima	número, arch, Etiq, NoDnd, IniX, IniY	25
9	Inserta_Objeto	número, animación	3
7	Inserta_Musica	número, nombre de archivo, Repetir	12
8	Inserta_Restricc.	número, X1, Y1, X2, Y2, Anm	11
9	Inserta_Uso	# Objeto, # animación, Etiq, e1..e4	17

APENDICE 3. FUENTE DEL PROGRAMA EJECUTOR.PAS

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
ENEP - ARAGON
INGENIERIA EN COMPUTACION

TESIS

FUNDAMENTOS DE PROGRAMACION PARA LA ELABORACION DE JUEGOS DE VIDEO

Ejecutor.pas : Programa que interactua con el usuario utilizando todos los elementos del juego.

Programado por :
Itsmael Manzo Salazar
Francisco Xavier Espinosa Madrigal

```
Uses LibSB,Crt,Dos,LibAni,libXms,LibGra,LibPan,LibDnd,libRat,libUti,
     LibDir, LibTxt,libpal;

Const
  DefLado = 30;      {Define rango de validacion de lado                }
  Deffrente = 6;     {Define rango de validacion de frente              }
  Maxmenu  = 30;     {Maximo de objetos en el menu de objetos           }
  Maxaccion = 30;    {Maximo de acciones por escena                     }
  Objetoini = 5;     {Objeto inicial declarado en el libreto            }
  ObjetoFin = 20;    {Objeto final declarado en el libreto              }

Type
  MenuObjetos=Record
    X, Y   : Integer;
    Anima  : byte;
  End;

  ListaAcciones=Record  {Se carga con datos del numero 9}
    Cursor : byte;
    Anima  : byte;
    Etiqueta : string[8];
    mapa1 : byte;
    valor1 : byte;
    mapa2 : byte;
    valor2 : byte;
    extra : byte;
  End;

  EstructuraPrin = record{Estructura principal para el intercambio de escenas}
      menu : array[1..maxmenu] of menuobjetos;
      activos : array[objetoini..objetofin] of boolean;
      etiquetas : array[1..numanima] of string[8];
      flechas : array[2..4] of boolean;
    end;

  Principal = object(animar)
    Procedure Inicio;
```

```
    Procedure Final;
    Procedure Mueve_Protagonista(Var ani : datosanima; var ultima:byte);
    Procedure Restringe_Protagonista(Var ani : datosanima;
              Var alcanzado : boolean; var ultima:byte);
    Procedure LeeLibreria(nombre : string);
    Function  Mapa : byte;
    Procedure Escena(libreto : string);
    Procedure MapaPrincipal;
    Procedure SalvaBanderas;
    Procedure LeeBanderas;
  end;

Var
  P          : principal;{Variable de tipo PRINCIPAL                          }
  F          : Formato;  {Usado para leer otros formatos graficos             }
  Maxanima   : byte;     {Numero maximo de animaciones                        }
  iconos     : duende;   {Conjunto de objetos utilizados como cursor         }
  escenas    : duende;   {Conjunto de escenas reducidas para el menu         }
  Pos_animas : lista;    {Posicion de los duendes con respecto a los otros    }
  NombreLib  : String;   {Nombre del libreto a ejecutar                       }
  NombreMus  : Array[1..20] Of String[8];
  Menu       : Array[1..maxmenu] Of byte;{Conjunto de posiciones de objetos   }
  ContObj    : byte;     {Contador de objetos                                 }
  Actua      : Array[0..maxaccion] of ListaAcciones; {Acciones realizables    }
  Contmenu: byte;        {Contador de menu de objetos                         }
  SalvaD     : EstructuraPrin; {Conjunto de datos a salvar de banderas        }
  ActivaSalvado : boolean; {Activa el salvado y lectura de banderas           }
  Memoria : boolean;     {Si esta activo despliega la memoria disponible      }
```

Procedimiento inicial.

```
Procedure Principal.inicio;
Var
  cont   : byte;
  arlib  : file;
  Dirl   : Dir_MCGA;
  pa     : paletas;
Begin
{$I+}
  venx1:=0; veny1:=0; venx2:=319; veny2:=199;
  contobj:=0; contmenu:=1;
  activasalvado:=false;
  ModoGrafico($13);
  escenamax:=1;
  Lee_Frases('FRASES.FRA', Manip_Frases);
  Lee_Frases('OBJETOS.FRA', Manip_Objetos);
  getmem(pantemporal,64000);
  if ReservaXms(252,xmsman[30]) = 0 then;
  Leer_FontUsuario('..\FONT\6x8.fnt',6,8);
  F.VerPCX('..\PCX\piedra.pcx', Pantemporal^);
  EscribeXMS(XMSMan[30], Pantemporal^, 64000, 0);
  F.VerPCX('..\PCX\mapa.pcx', Pantemporal^);
  EscribeXMS(XMSMan[30], Pantemporal^, 64000, 192000);
  randomize;
  Lee_ExtVOC('..\sb\final.VOC');
  leerduende(iconos,'..\dnd\ejecutor.dnd');
  leerduende(escenas,'..\dnd\escenas.dnd');

  for cont := 1 to 20 Do NombreMus[cont][0] := #255;
  For cont:=0 to 255 do banderas[cont]:=0; {Inicializa las banderas}
  Bandera_Mus := 0;
  if checa_raton then;
  Busca_FMDriver;
```

```
  Inicializa_FM;
  memoria:=false;
  if si_parametro('/MEM') or si_parametro('/mem') then memoria:=true;
  if Si_parametro('/n') or si_parametro('/N') then
    begin
        modografico(19);
        pa.discol('defecto.pal',1);
        pa.ponpaleta;
        NombreLib := '*.lib';
        Dirl.Define_Unidades;
        If Not Dirl.Lee_DirArch(NombreLib,False) Then
        begin
          ModoGrafico(3);
          Termina_SBP;
          Finaliza_FM;
          Halt;
        end;
    end;
  assign(arlib,NombreLib);
{$I-} reset(arlib,1);
  if IOresult<>0 then
    begin
      Finaliza_FM;
      Mensaje('Error al abrir el archivo: '+NombreLib,1);
      ModoGrafico(3);
      Halt;
    end;
  close(arlib);
  If IOResult=0 Then;
end;
```

Procedimiento Final.

```
Procedure Principal.final;
Var
  Cani : byte;
Begin
  Termina_SBP;
  Finaliza_FM;
  liberaduende(iconos);
  liberaduende(escenas);
  freemem(pantemporal,64000);
  if LiberaXms(xmsman[30]) = 0 then;
  ModoGrafico($03);
End;
```

Salva los objetos tomados de una escena

```
Procedure Principal.SalvaBanderas;
Var
  cont : byte;
Begin
  for cont:= 1 to contobj do
    begin
      salvad.menu[cont].anima:=menu[cont];
      salvad.menu[cont].x:=anis[menu[cont]].x;
      salvad.menu[cont].y:=anis[menu[cont]].y;
      salvad.activos[cont]:=anis[menu[cont]].activo;
    end;
  for cont:=2 to 4 do salvad.flechas[cont]:=anis[cont].activo;
End;
```

Lee los objetos tomados de otra escena

```
Procedure Principal.Leebanderas;
Var
  cont : byte;
Begin
  for cont:= 1 to contobj do
    begin
      menu[cont]:=salvad.menu[cont].anima;
      anis[menu[cont]].x:=salvad.menu[cont].x;
      anis[menu[cont]].y:=salvad.menu[cont].y;
      anis[menu[cont]].activo:=salvad.activos[cont];
      anis[menu[cont]].prota:=2;
      Primer_Anima(Anis[menu[cont]],salvad.etiquetas[menu[cont]]);
    end;
  for cont:=2 to 4 do anis[cont].activo:=salvad.flechas[cont];
End;
```

Localiza el punto central horizontal de un cursor

```
Function Cx(Xr : word; curact : byte) : word;
Var
   Cad    : String[3];
   Ancho : Word;
Begin
   Move(iconos.dnd[curact]^, Cad, 4);
   Move(Cad,Ancho,2);
   if curact<>1 then CX:=xr + (ancho div 2)
   else cx:=xr;
End;
```

Localiza el punto central vertical de un cursor

```
Function Cy(Yr : word; curact : byte) : word;
Var
   Cad    : String[3];
   largo : Word;
Begin
   Move(iconos.dnd[curact]^, Cad, 4);
   Move(Cad[2],largo,2);
   if curact<>1 then CY:=yr + (largo div 2)
   else cy:=yr;
End;
```

Checa que las coordenadas xr, yr se encuentren sobre un objeto.

```
FUNCTION Checa_Objeto(Xr, Yr : Word; Numero : Byte) : Boolean;
Var
   Cad          : String[3];
   Ancho, Largo : Word;
   XIni, YIni   : Integer;
   ok : boolean;
Begin
   Move(monos[anis[numero].duendo].dnd[anis[numero].aniact.imagen]^, Cad, 4);
   Move(Cad,Ancho,2);  Move(Cad[2],Largo,2);
   XIni := anis[Numero].x;
   YIni := anis[Numero].y;
   If (xr>=xini) and (xr<=xini+ancho-1) and (yr>=yini) and (yr<=yini+largo)
      then ok:= True Else ok := False;
   checa_objeto:=ok;
End;
```

Inserta un objeto en la mochila.

```
Procedure Inserta_objeto(anima:byte);
Var
  cont : byte;
```

```
Begin
  inc(contobj);
  for cont:=contobj downto 1 do menu[cont]:=menu[cont-1];
  menu[1]:= anima;
end;
```

Quita un objeto de la mochila.

```
Procedure Quita_objeto(numero : byte);
Var
  cont : byte;
begin
  for cont:=numero to contobj-1 do menu[cont]:=menu[cont+1];
  dec(contobj);
end;
```

Pone los objetos que hay en la mochila.

```
Procedure Lista_objetos(inicio:byte);
Var
  cont : byte;
  final : byte;
  x : byte;
  Cad          : String[3];
  Ancho, Largo : Word;
  rx,ry : byte;
Begin
  if contobj=3 then {Desactiva las flecha}
     begin
       anis[4].activo:=false;
       anis[2].activo:=false;
     end;
  x:=0;
  for cont:=1 to contobj do anis[menu[cont]].activo:=false;
  if inicio+2>contobj then final:=contobj else final:=inicio+2;
  for cont:=inicio to final do
    begin
      Move(monos[anis[menu[cont]].duende].dnd[anis[menu[cont]].aniact.imagen]^
           ,Cad, 4);
      Move(Cad,Ancho,2);  Move(Cad[2],Largo,2);
      rx:=ancho div 2;
      ry:=largo div 2;
      anis[menu[cont]].activo:=true;
      anis[menu[cont]].x:=214+(x*36)-rx;
      anis[menu[cont]].y:=180-ry;
      inc(X);
    end;
End;
```

Mueve al protagonista.

```
Procedure Principal.Mueve_Protagonista(Var ani : datosanima; var ultima:byte);
Var
 Prinx,Priny : integer; {Coordenadas centrales del personaje principal      }
 boton,rx,ry : word;    {Coordenadas y boton del raton                      }
Begin
  if ani.aniact.imagen<>0 then
    begin
      posraton(boton,rx,ry);
      if ani.pila[ani.nivel].tociclo=900 then ultima:=0;
      Puntoduende(monos[ani.duende],ani,prinx,priny,ani.aniact.imagen);
      ani.nuevadur:=abs(rx-prinx) div deflado;
      if ani.nuevadur>0 then ani.nuevadur:=100;
      ani.xfinal:=rx-(prinx-ani.x); ani.yfinal:=ry-(priny-ani.y);
```

```
    if (ani.nuevadur>0) then
      begin
        if rx>prinx then
          begin
            if ultima=3 then
              begin
                ani.pila[ani.nivel].ciclo:=1;
                ani.pila[ani.nivel].tociclo:=ani.nuevadur;
                Ani.inc2y:=signo(ry,priny)*2;
              end
            else
              begin
                Primer_Anima(ani,'DER');
                Ani.inc2y:=signo(ry,priny)*2;
              end;
            ultima:=3;
          end;
        if rx<prinx then
          begin
            if ultima=4 then
              begin
                ani.pila[ani.nivel].ciclo:=1;
                ani.pila[ani.nivel].tociclo:=ani.nuevadur;
                Ani.inc2y:=signo(ry,priny)*2;
              end
            else
              begin
                Primer_Anima(ani,'IZQ');
                Ani.inc2y:=signo(ry,priny)*2;
              end;
            ultima:=4;
          end;
      end
    else
      begin
        ani.nuevadur:=100;
        ani.xfinal:=rx-(prinx-ani.x); ani.yfinal:=ry-(priny-ani.y);
        if ani.nuevadur>0 then
          begin
            if ry>priny then
              begin
                if ultima=2 then
                  begin
                    ani.pila[ani.nivel].ciclo:=0;
                    ani.pila[ani.nivel].tociclo:=ani.nuevadur;
                    Ani.inc2x:=signo(rx,prinx)*2;
                  end
                else
                  begin
                    Primer_Anima(ani,'ABAJO');
                    Ani.inc2x:=signo(rx,prinx)*2;
                  end;
                ultima:=2;
              end;
            if ry<priny then
              begin
                if ultima=1 then
                  begin
                    ani.pila[ani.nivel].ciclo:=0;
                    ani.pila[ani.nivel].tociclo:=ani.nuevadur;
                    Ani.inc2x:=signo(rx,prinx)*2;
                  end
```

```
                else
                  begin
                    Primer_Anima(ani,'ARRIBA');
                    Ani.inc2x:=signo(rx,prinx)*2;
                  end;
                ultima:=1;
              end;
          end;
      end;
  end;
End;
```

Restringe los movimientos del protagonistá.

```
Procedure Principal.Restringe_Protagonista(Var ani : datosanima;
          Var alcanzado : boolean; var ultima:byte);
Begin
  if (abs(ani.x-ani.xfinal)<5) and (abs(ani.y-ani.yfinal)<>0)
    and ((ultima=3) or (ultima=4)) then
    Begin
      if ani.Yfinal>ani.Y then
        begin
          Primer_Anima(ani,'ABAJO');
          Ani.inc2Y:=signo(ani.yfinal,ani.y)*2;
          ultima:=2;
        end;
      if ani.Yfinal<ani.Y then
        begin
          Primer_Anima(ani,'ARRIBA');
          Ani.inc2y:=signo(ani.yfinal,ani.y)*2;
          ultima:=1;
        end;
    end;

  if abs(ani.xfinal-ani.x)<5 then
    begin
      ani.x:=ani.xfinal;
      if (abs(ani.yfinal-ani.y)=0) and alcanzado then
        begin
          ani.nuevadur:=0;
          ani.pila[ani.nivel].ciclo:=ani.pila[ani.nivel].tociclo;
          alcanzado:=false;
          ultima:=0;
        end;
      ani.aniact.increx:=0; ani.inc2x:=0;
    end;
  if abs(ani.yfinal-ani.y)<3 then
    begin
      ani.y:=ani.yfinal;
      if (abs(ani.xfinal-ani.x)=0) and alcanzado then
        begin
          ani.nuevadur:=0;
          ani.pila[ani.nivel].ciclo:=ani.pila[ani.nivel].tociclo;
          alcanzado:=false;
          ultima:=0;
        end;
      ani.aniact.increy:=0; ani.inc2y:=0;
    end;
End;
```

Lee una libreria *.LIB

```
Procedure Principal.leelibreria(nombre : string);
```

```
Var
  Arlib  : file;
  Accion : byte;
  archivo: string;
  contmus,
  numbyte,numbyte2: byte;
  x1,y1,x2,y2,
  numinteger : integer;
  cadena : string;

Begin
  contmus := 0;
  assign(arlib,nombre);
  {$I-}
  reset(arlib,1);
  {$I+}
  if IOresult<>0 then
    begin
      writeln('Error al abrir el archivo : ',nombre);
      halt;
    end;
  while not eof (arlib) do
    begin
      blockread(arlib,accion,1);
      case (accion) of
        1 : begin {Inserta pantalla de fondo de un archivo PCX                }
              blockread(arlib,archivo,9);
              archivo:=archivo+' ';
              archivo:='..\pcx\'+copy(archivo,1,(pos(' ',archivo)-1))+'.pcx';
              F.verpcx(archivo,pantemporal^);
              escribexms(xms^man[30],pantemporal^,64000,128000);
            end;
        4 : begin {Inserta un duende                                          }
              Blockread(arlib,numbyte,1);
              blockread(arlib,archivo,9);
              archivo:=archivo+' ';
              archivo:='..\dnd\'+copy(archivo,1,(pos(' ',archivo)-1))+'.dnd';
              Leerduende(monos[numbyte],archivo);
            end;
        5 : begin {Inserta una animacion                                      }
              Blockread(arlib,numbyte,1);
              blockread(arlib,archivo,9);
              archivo:=archivo+' ';
              archivo:='..\anm\'+copy(archivo,1,(pos(' ',archivo)-1))+'.anm';
              IniciaAni(Anis[numbyte]);
              lee_archivo(archivo,Anis[numbyte]);
              Anis[numbyte].activo:=true;
              blockread(arlib,cadena,9);
              blockread(arlib,anis[numbyte].duende,1);
              blockread(arlib,anis[numbyte].x,2);
              blockread(arlib,anis[numbyte].y,2);
              Primer_Anima(Anis[numbyte],cadena);
              salvad.etiquetas[numbyte]:=cadena;
              if numbyte>maxanima then maxanima:=numbyte;
            end;
        6 : begin {Inserta un objeto                                          }
              Blockread(arlib,numbyte,1);
              Blockread(arlib,numbyte2,1);
              anis[numbyte].prota:=2;
            end;
        7 : begin {Inserta fondo musical de un archivo CMF                    }
              Blockread(arlib,numbyte,1);
```

```
            blockread(arlib,archivo,9);
            archivo:=archivo+' ';
            archivo:=copy(archivo,1,(pos(' ',archivo)-1));
            Inc(ContMus);
            NombreMus[ContMus] := Espacios(Archivo,8,2);
            Blockread(arlib,numbyte,1);
            NombreMus[ContMus][0] := Chr(numbyte);
          end;
      8 : Begin {Inserta un rectangulo de restriccion                    }
            Blockread(arlib,numbyte,1);
            Blockread(arlib,x1,2);
            Blockread(arlib,y1,2);
            Blockread(arlib,x2,2);
            Blockread(arlib,y2,2);
            if ((numbyte=0) or (numbyte=255)) then
              begin {Utilizado para mensajes}
                Blockread(arlib,numbyte2,1);
                if numbyte=255 then
                     inserta_men(x1,y1,x2,y2,bloque,numbyte2,true)
                else inserta_men(x1,y1,x2,y2,bloque,numbyte2,false);
              end
            else
              begin {Utilizado para restricciones}
                if ((numbyte<>0) and (numbyte<>255)) then
                  begin
                    inserta_res(x1,y1,x2,y2,nopaso,numbyte);
                    Blockread(arlib,numbyte2,1);
                    if numbyte<>0 then
                     begin
                       anis[numbyte2].restric:=numbyte;
                       anis[numbyte2].prota:=1;
                       anis[numbyte2].xres:=x1;
                       anis[numbyte2].yres:=y1;
                     end;
                  end;
              end;
          End;
      9 : begin {Inserta una accion }
            inc(actua[0].cursor);
            Blockread(arlib,numbyte,1);
            actua[actua[0].cursor].cursor:=numbyte;
            Blockread(arlib,numbyte,1);
            actua[actua[0].cursor].anima:=numbyte;
            blockread(arlib,cadena,9);
            actua[actua[0].cursor].etiqueta:=cadena;
            Blockread(arlib,numbyte,1);
            actua[actua[0].cursor].mapa1:=numbyte;
            Blockread(arlib,numbyte,1);
            actua[actua[0].cursor].valor1:=numbyte;
            Blockread(arlib,numbyte,1);
            actua[actua[0].cursor].mapa2:=numbyte;
            Blockread(arlib,numbyte,1);
            actua[actua[0].cursor].valor2:=numbyte;
            Blockread(arlib,numbyte,1);
            actua[actua[0].cursor].extra:=numbyte;
          end;
    end;
  end;
  close(arlib);
  If ContMus>0 Then Bandera_Mus := 1;
End;
```

Despliega el mapa de control de escenas.

```
Function Principal.mapa : byte;
Const
  tabla : array[1..12,1..4] of word = (Numero de escenas en el juego)
       ((167,166,233,199),(83 ,166,149,199),(1   ,166,66 ,199),
        (0  ,124,66 ,157),(0  ,83 ,66 ,116),(0  ,42 ,66 ,75 ),
        (0  ,0  ,66 ,33 ),(83 ,0  ,149,33 ),(167,0  ,233,33 ),
        (253,0  ,319,33 ),(253,42 ,319,75 ),(253,83 ,319,116));

Var
  Boton : byte;
  salida : byte;
  cont  : byte;
Begin
  salida:=0;
  LeeXms(xmsman[30],mem[$A000:0],64000,192000); {Pantalla de Menu piramide}
  for cont := 1 to escenamax
  do putduende(tabla[cont,1],tabla[cont,2],escenas,mem[$a000:0],cont);
  iniciaraton(160,100);
  while salida=0 do
    begin
      boton:=0;
      cursorraton(true);
      while boton <> 1 do boton:=NumbotonRaton;
      cursorraton(false);
      {Salir del programa}
      if rangoraton(260,142,300,196) or rangoraton(239,183,315,195)
         or rangoraton(251,155,310,165) then
         if mensaje('Esta seguro de salir',2)=1 then salida:=255;
      for cont:=1 to escenamax do
        begin
          if rangoraton(tabla[cont,1],tabla[cont,2],tabla[cont,3]
             ,tabla[cont,4])
             then salida:=cont;
        end;
    end;
  mapa:=salida;
End;
```

Ejecuta una escena.

```
Procedure Principal.Escena(libreto : string);
Var
 Cani : byte;            {Cuenta animaciones                          }
 Tecla : char;           {Tecla de salida                             }
 boton,rx,ry : word;     {Coordenadas y boton del raton               }
 Ultima : byte;          {Ultima accion realizada                     }
 posx, posy : integer;
 alcanzado : boolean;
 aprieta : boolean;      {Indica si se ha soltado el boton del raton  }
 aux     : byte;
 Cad     : String;
 alto    : boolean;      {Se detiene toda la accion, menu de opciones }
 Muevete : boolean;      {Indica que si se puede mover el protagonista}
 CurAct  : byte;         {Cursor de despliegue actual                 }
 CurAni  : byte;
 Noobj   : boolean;      {No valida el cuadro de objetos              }
 Nummen  : integer;
 Accion  : byte;         {Apuntador a la lista de acciones a ejecutar }
 cadena  : string[8];    {Cadena auxiliar}
 Salir   : byte;         {Condicion de salida}
```

```
                        {0   Quedarse      }
                        {255 Salir         }
                        {1-254 otra escena}
 ContVueltas : longint; {Contador de pantallas armadas por escena            }
Begin
  for cani:=1 to numanima do {Inicializa las animaciones}
    begin
      iniciaAni(anis[cani]);
      Anis[cani].activo:=true;
      Anis[cani].prota:=0;
      Anis[cani].restric:=0;
      Anis[cani].xres:=0;
      Anis[cani].yres:=0;
      Anis[cani].x:=-100;
      anis[cani].y:=-100;
    end;
  for cani:=0 to maxaccion do {Inicializa las actuaciones}
    begin
      actua[cani].cursor:=0;
      actua[cani].anima:=0;
      actua[cani].etiqueta:='';
      actua[cani].mapal:=0;
      actua[cani].valor1:=0;
      actua[cani].mapal:=0;
      actua[cani].valor1:=0;
      actua[cani].extra:=0;
    end;
  INTER:=TRUE;    {Indica que si habra un protagonista}
  contvueltas:=0;
  if inter then anis[1].prota:=1;
  maxanima:=0; ultima:=0; alto:=false; curact:=1;  curani:=1;
  inicia_Res(nopaso); {Usado para restricciones}
  inicia_Res(bloque); {Usado para mensajes      }
  P.Leelibreria(libreto);
  anis[2].activo:=false;
  anis[4].activo:=false;
  if activasalvado then begin leebanderas; activasalvado:=false; end;
  for Cani:=1 to maxanima do
    begin
      Puntoduende(monos[anis[cani].duende],anis[cani],posx,posy
      ,anis[cani].aniact.imagen);
      pos_animas[cani].largo:=posy-anis[cani].y;
      pos_animas[cani].numero:=cani;
    end;
  tecla:=#255;
  while tecla<>#27 do {Inicia el ciclo de animacion e interfase de usuario    }
    begin
      LeeXms(xmsman[30],pantemporal^,64000,128000); {Pantalla de fondo actual}
{ Pone el nombre del objeto }
      if (memoria) and (curact=1) then
         {Memoria Disponible, parametro /mem requerido}
         begin
           destinolet:=pantemporal;
           Letrero(2,19,Espacios(Nac(MaxAvail),6,2)+' B',255);
           destinolet:=ptr($A000,0);
         end;

      If CurAct>1 Then
      begin
        DestinoLet := Pantemporal;
        Letrero(2,19,Espacios(Cad_Objetos(CurAct),25,2),255);
        DestinoLet := Ptr($A000,0);
```

```
      end;
      for Cani:=1 to maxanima do {Da prioridades a las animaciones          }
        begin
          pos_animas[cani].pos:=
          anis[pos_animas[cani].numero].y+pos_animas[cani].largo;
        end;
      ordena(pos_animas,1,maxanima);
      for Cani:=1 to maxanima do {Arma la pantalla temporal para desplegarla }
        begin
          aux:=pos_animas[cani].numero;
          if (anis[aux].activo) then
            begin
              Corre_Anima(Anis[aux].AniAct.etiqueta,Anis[aux]
                          ,monos[anis[aux].duende]);
              if anis[aux].restric<>0 then
                begin
                  inserta_res(anis[aux].x-anis[aux].xres,anis[aux].y
                  +(anis[aux].puntoy-5)-anis[aux].yres
                  ,anis[aux].x+(anis[aux].puntox*2)+anis[aux].xres
                  ,anis[aux].y+(anis[aux].puntoy+5)+anis[aux].yres
                  ,nopaso,anis[aux].restric);
                end;
            end;
        end;
      posraton(boton,rx,ry);
      if boton=0 then aprieta:=true; {aprieta:=true;}
      if (inter) and (alto=false) then
        putduende(rx,ry,iconos,pantemporal^,curact);

{ Pone la frase si es requerido }
      If Retardo_Frases>0 Then
      begin
        Pon_Frase(Numero_Frase, Color_Frase, true);
        Dec(Retardo_Frases);
        If Boton=2 Then Retardo_Frases := 0;
      end;

      Despliega(pantemporal); {Despliega en video la pantalla armada        }

{       inc(contvueltas); letrero(1,2,nac(contvueltas),5);}
{Contador de pantallas armadas}

      if inter then
        begin
          if alto then
            begin
              salir:=mapa;
              if salir<>escenaact then
                begin escenaact:=salir; tecla:=#27; end
              else delay(150);
              alto:=false;
              boton:=0;
            end;
          restringe_protagonista(anis[1],alcanzado,ultima);
          if keypressed then
            begin
              tecla:=readkey;
              if tecla=#27 then tecla:=#255;
              if tecla=#13 then boton:=1;
              if tecla=#0 then
                begin
                  tecla:=readkey;
```

```
        case tecla of
          #77 : begin
                  if ((mem[$40:$17] and 1)=1) then inc(rx,1)
                  else inc(rx,10);
                  iniciaraton(rx,ry);
                end;
          #75 : begin
                  if ((mem[$40:$17] and 1)=1) then dec(rx,1)
                  else dec(rx,10);
                  iniciaraton(rx,ry);
                end;
          #80 : begin
                  if ((mem[$40:$17] and 1)=1) then inc(ry,1)
                  else inc(ry,8);
                  iniciaraton(rx,ry);
                end;
          #72 : begin
                  if ((mem[$40:$17] and 1)=1) then dec(ry,1)
                  else dec(ry,8);
                  iniciaraton(rx,ry);
                end;
        end;
      end;
  end;
if (boton=1) and (aprieta) then {Rutinas de apretar boton          }
  begin
    muevete:=true; noobj:=true;
    boton:=0;
    {Menu de objetos hacia abajo}
    if rangoraton(307,181,317,191) and (contobj>3)
       and (contmenu<contobj) and (contmenu<>contobj-2) then
      begin
        if (contmenu+3) <= contobj then inc(contmenu,3);
        if contmenu+3>contobj then anis[4].activo:=false;
        if contmenu>3 then anis[2].activo:=true;
        lista_objetos(contmenu);
        delay(150);
      end;
    {Menu de objetos hacia arriba}
    if rangoraton(307,169,317,179) and (contobj>1)
       and (contmenu<>1) then
      begin
        if (contmenu-3) >= 1 then dec(contmenu,3)
        else contmenu:=1;
        if contmenu=1 then anis[2].activo:=false;
        if contmenu+3<=contobj then anis[4].activo:=true;
        if contobj=3 then anis[4].activo:=false;
        lista_objetos(contmenu);
        delay(150);
      end;

    {Verifica que se tome un objeto      }
    for cani:=1 to maxanima do
      begin
        if (anis[cani].activo) And
        (Checa_Objeto(cx(rx,curact), Cy(ry,curact), cani)) then
          begin
            {Zona de cambio de objeto}
            if (RangoRaton(9,7,310,147)) and (curact<>1) then else
            if (anis[cani].prota=2) then
            begin
              if rangoraton(199,168,228,192) and (contobj>=1) then
```

```
                        begin
                          if curact=1 then
                            begin
                              quita_objeto(contmenu);
                              if (contmenu>3) and (contobj+1=contmenu) then
                              dec(contmenu,3);
                              lista_objetos(contmenu);
                            end
                          else
                            begin
                              menu[contmenu]:=curani;
                            end;
                        end;
                      if rangoraton(235,168,264,192)
                         and (contobj>=contmenu+1) then
                        begin
                          if curact=1 then
                            begin
                              quita_objeto(contmenu+1);
                              lista_objetos(contmenu);
                            end
                          else
                            begin
                              menu[contmenu+1]:=curani;
                            end;
                        end;
                      if rangoraton(271,168,300,192)
                         and (contobj>=contmenu+2) then
                        begin
                          if curact=1 then
                            begin
                              quita_objeto(contmenu+2);
                              lista_objetos(contmenu);
                            end
                          else
                            begin
                              menu[contmenu+2]:=curani;
                            end;
                        end;

                       if (curact<>1)  then
                         begin
                           anis[curani].activo:=true;
                           anis[curani].x:=rx;
                           anis[curani].y:=ry;
                           rx:=cx(rx,curact);
                           ry:=cy(ry,curact);
                           lista_objetos(contmenu);
                         end;
{ Cambia el tipo de cursor }
                      curact:=anis[cani].aniact.imagen;
                      iniciaraton(rx-(cx(rx,curact)-rx)
                                  ,ry-(cy(ry,curact)-ry));
                      Curani:=cani;
                      anis[Curani].activo:=false;
                      muevete:=false;
                      delay(150);
                      noobj:=false;
                    end;
                {Zona que ejecuta una accion}
                  If actua[0].cursor=0 then accion:=0 else accion:=1;
                  while (accion<=actua[0].cursor) do
```

```
                begin
                  if ((curact=actua[accion].cursor)
                  and (cani=actua[accion].anima))
                  and (banderas[actua[accion].mapa1]=
                       actua[accion].valor1) then
                    begin
                      cadena:=actua[accion].etiqueta;
                      If cadena='APAGADOR' then {rutina de apagador}
                        begin {Solo cambia la etiqueta a ejecutar}
                          case (actua[accion].extra) of
                            0 : begin
                                  actua[accion].extra:=1;
                                  cadena:='ON';
                                end;
                            1 : begin
                                  actua[accion].extra:=0;
                                  cadena:='OFF';
                                end;
                          end;
                        delay(100);
                        end;
                      Primer_Anima(Anis[actua[accion].anima],cadena);
                      accion:=actua[0].cursor;
                    end;
                  inc(accion);
                end;
            end;
        end;
    {Zona de accion, mueve al protagonista}
      if (RangoRaton(9,7,310,147)) and (muevete) and (curact=1) then
        Begin
          nummen:=checa_men(rx,ry,bloque);
          if nummen<>0 then {Utilizado para mensajes en recuadros}
            begin
              Numero_Frase:=nummen;
              Retardo_Frases:=50;
              Color_Frase := 255;
              delay(150);
            end;
          if nummen=0 then  {Utilizado para mover al protagonista}
            begin
              Mueve_Protagonista(anis[1],ultima);
              alcanzado:=true;
              aprieta:=false;
            end;
        end;
    {Zona de la piramide o menu de opciones}
      If RangoRaton(107,168,141,192) and (curact=1) then
        begin
          alto:=true;
        end;

      {Zona de insercion de objetos en la mochila}
      If rangoraton2(cx(rx,curact),Cy(ry,curact),163,168,192,192)
         and (noobj) then
        begin
          if curact<>1 then
            begin
              anis[curani].x:=214-(cx(rx,curact)-rx);
              anis[curani].y:=180-(cy(ry,curact)-ry);
              anis[curani].activo:=true;
              iniciaraton(cx(rx,curact),Cy(ry,curact));
```

```
                    curact:=1;
                    inserta_objeto(curani);
                    contmenu:=1;
                    lista_objetos(contmenu);
                    if contobj>3 then anis[4].activo:=true;
                    anis[2].activo:=false;
                  end;
                end;
              end;
          end;
{ Activa la música si es requerido }
      If Bandera_Mus<>0 Then
      Begin
        If (Bandera_Mus<21) And (NombreMus[Bandera_Mus][0]<#255) Then
        Begin
           Move(NombreMus[Bandera_Mus], Cad, 9);
           Cad[0] := #8;
           Cad := Cad+' ';
           Cad :='..\sb\'+copy(Cad,1,(pos(' ',Cad)-1))+'.CMF';
           Lee_FM(Cad);
           Replte_FM := Ord(NombreMus[Bandera_Mus][0]);
           Toca_FM;
           Bandera_Mus := 0;
        End Else Bandera_Mus := 0;
      End;
    end; {Final del ciclo que siempre funciona }

  {Se libera de memoria a duendes y animaciones}
  For cani:=1 to numduende do
      if monos[cani].total>0 then LiberaDuende(monos[cani]);
  For cani:=1 to numanima do
      if anis[cani].tamani>0 then FinalizaAni(Anis[cani]);
  salvabanderas;
  activasalvado:=true;
  Retardo_Frases := 0;
End;
```

Manejador de los mapas.

```
Procedure Principal.MapaPrincipal;
Begin
  EscenaAct:=1;
  while escenaact<>255 do
    begin
      case (escenaact) of
        1 : Escena('Maya1.lib');
        2 : Escena('Maya2.lib');
        3 : Escena('Maya3.lib');
        4 : Escena('Maya4.lib');
        5 : Escena('Maya5.lib');
        6 : Escena('Maya6.lib');
        7 : Escena('Maya7.lib');
        8 : Escena('Maya8.lib');
        9 : Escena('Maya9.lib');
       10 : Escena('Maya10.lib');
       11 : Escena('Maya11.lib');
       12 : Escena('Maya12.lib');
      end;
    end;
End;
```

```
 RUTINA PRINCIPAL.
Begin
  P.inicio;
  if Si_parametro('/n') or si_parametro('/N') then P.Escena(nombrelib)
  else P.MapaPrincipal;
  P.final;
End.
```

```
Ultima
   1  Arriba
   2  Abajo
   3  Derecha
   4  Izquierda
```

APENDICE 4. FUENTES DE LAS LIBRERIAS

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
ENEP - ARAGON
INGENIERIA EN COMPUTACION

TESIS

FUNDAMENTOS DE PROGRAMACION PARA LA ELABORACION DE JUEGOS DE VIDEO

Programado por :
Itsmael Manzo Salazar
Francisco Xavier Espinosa Madrigal

LibAni.pas : Libreria para el manejo de las animaciones.

```
UNIT LIBANI;

INTERFACE
Uses libdnd, libuti, crt, libgra, libdir,libxms,librat, LibTxt, LibSB, libPan;

Const

   RegistroAnima = 17;
   NumDuende = 15;
   NumAnima    = 35;

Type
   arreglo1 = array[1..6] of integer;
   Restric = array[0..30] of arreglo1;

   Anima = record
     Estado    : byte;          {1 byte }   {Accion a realizar                }
     Etiqueta  : string[8];     {9 bytes}   {Etiqueta de la accion            }
     Imagen    : byte;          {1 bytes}   {Numero de imagen actual de duende}
     Duracion  : word;          {2 bytes}   {Duracion de la accion            }
     IncreX    : integer;       {2 bytes}   {Incremento en X                  }
     IncreY    : integer;       {2 bytes}   {Incremento en Y                  }
   end;                         {17total}

   PilaAnima  = record
     Posicion  : word;          {2 bytes}   {Posicion actual dentro del apun. }
     vuelta    : boolean;       {1 bytes}   {Vuelta completada si o no        }
     Origen    : word;          {2 bytes}   {Posicion en la que se inicia todo}
     ciclo     : word;          {2 bytes}   {Contador de numero de ciclos     }
     ToCiclo   : word;          {2 bytes}   {Numero de ciclos                 }
   end;                         {9 total}

   DatosAnima = record
     Restric   : byte;          {1 byte }   {Numero de restriccion movible    }
     xres,yres : integer;       {4 bytes}   {Suma extra a las restricciones   }
     puntox,puntoy : integer;   {4 bytes}   {Puntos centrales de el duende    }
     Duende    : byte;          {1 byte }   {Numero de duende activo          }
     prota     : byte;          {1 byte }   {1 Es protagonista o no (restric.)}
```

```
                                                    {0 Es una animacion normal          }
                                                    {2 Es un objeto                     }
    inc2x,inc2y : integer;     {4 bytes}            {Incrementos extras a la animacion  }
    Xfinal,yfinal : integer;   {4 bytes}            {Lugar al que se debe llegar prota  }
    x,y         : integer;     {4 bytes}            {Coordenadas actuales               }
    Animacion   : Pointer;     {4 bytes}            {Rutinas de animacion (tipo anima)  }
    AniAct      : Anima;       {17 b    }           {Animacion actual                   }
    TamAni      : Word;        {2 bytes}            {tamaño del bloque de animacion     }
    duracion    : word;        {2 bytes}            {Contador duracion actual           }
    Nivel       : byte;        {1 byte }            {Nivel actual en la pila            }
    Pila        : array[1..10] of PilaAnima;        {Pila de animaciones (para saltos)  }
    Activo      : boolean;     {1 byte }            {Animacion activa o no activa       }
    NuevaDur    : word;        {2 bytes}            {Duracion extra a la animacion      }
  end;

  Animar = object(duendes)
    Procedure Iniciaani(Var D : DatosAnima);
    Procedure FinalizaAni(Var D : DatosAnima);
    Procedure Puntoduende(mono:duende; Var D : datosanima;
                var xvir, yvir : integer; numero:byte);
    Procedure Lee_Archivo(nombre:string; Var D : DatosAnima);
    Procedure Escribe_Archivo(nombre:string; D : DatosAnima);
    Procedure Busca_anima(numero : word; Var D : DatosAnima);
    Procedure Pon_anima(numero : word; Var D : DatosAnima);
    Procedure Inserta_Anima(numero : word; Var D : DatosAnima);
    Procedure Borra_Anima(Lugar : word; Var D : DatosAnima);
    Function  Busca_Etiqueta(eti:string; Var D : DatosAnima) : word;
    Procedure Ubica_duende(Var boton : word; Var xi,yi: integer;
                mono : duende; num : byte);
    Procedure Renglon_anima(cory : byte; D : DatosAnima; mono:duende;
                estado : boolean);
    Procedure Lista_anima(numero : word; cory : byte; D : DatosAnima;
                Mono : Duende);
    Procedure Otra_Anima(Var D:DatosAnima);
    Procedure Corre_Anima(nombre:string; Var D:DatosAnima; Var Mono:Duende);
    Procedure Primer_Anima(var D:DatosAnima; nombre:string);
    Procedure Zona_anima(Var mono : duende; Var DA : DatosAnima);
  end;

Procedure Inicia_res(var Nopaso : restric);
Procedure Inserta_res(xs,ys,xi,yi : integer; var Nopaso : restric;
            numero : byte);
Function  Checa_res(xcor, ycor : integer; Nopaso : restric;
            menos:byte ): boolean;
Procedure Inserta_men(xs,ys,xi,yi : integer; var Nopaso : restric;
            men:integer; valor : boolean);
Function  Checa_men(xcor,ycor:integer; var Nopaso : restric): integer;


Var
  ñomanima : string;          { Nombre de la animacion en disco                 }
  Bot : duende;               { Variable tipo duende para botones               }
  monos : array[1..Numduende] of duende; {Matriz de duendes                     }
  anin  : array[1..Numanima] of DatosAnima;{Matriz de animaciones               }
  nopaso : restric;           { Variable de marcos de restriccion               }
  Bloque : restric;
  Bandera_Mus : Byte;
  Inter : boolean;            {Indica que esta activo el cursor del protagonista }
  BANDERAS: Array[0..255] of byte;
  pausa  : word;
  escenaact,escenamax: byte;{Escena actual y escena maxima                      }


IMPLEMENTATION

INICIA RESTRICCIONES
Procedure Inicia_res(var Nopaso : restric);
```

```
var
 cont : byte;
Begin
  for cont:=0 to 30 do
    begin
     nopaso[cont,1]:=0;
     nopaso[cont,2]:=0;
     nopaso[cont,3]:=0;
     nopaso[cont,4]:=0;
     nopaso[cont,5]:=0;
     nopaso[cont,6]:=0;
    end;
    Nopaso[0,1]:=0;
End;
```

INSERTA RESTRICCIONES

```
Procedure Inserta_res(xs,ys,xi,yi : integer; var Nopaso : restric;
                      numero:byte);
Begin
  if numero > Nopaso[0,1] then Nopaso[0,1]:=numero;
  nopaso[numero,1]:=xs;
  nopaso[numero,2]:=ys;
  nopaso[numero,3]:=xi;
  nopaso[numero,4]:=yi;
End;
```

INSERTA MENSAJES POR RECUADROS

```
Procedure Inserta_men(xs,ys,xi,yi : integer; var Nopaso : restric;
                      men:integer; valor:boolean);
Var
  numero : byte;
Begin
  inc(Nopaso[0,1]); numero:=Nopaso[0,1];
  nopaso[numero,1]:=xs;
  nopaso[numero,2]:=ys;
  nopaso[numero,3]:=xi;
  nopaso[numero,4]:=yi;
  nopaso[numero,5]:=men;
  if valor then nopaso[numero,6]:=1
  else nopaso[numero,6]:=0;
End;
```

CHECA RESTRICCIONES

```
Function Checa_res(xcor, ycor : integer; Nopaso : restric;
                   menos:byte): boolean;
Var
  cont : byte;
Begin
  checa_res:=false;
  for cont:= 1 to Nopaso[0,1] do
    begin

      if (xcor >= nopaso[cont,1]) and (ycor >= nopaso[cont,2])
      and (xcor <= nopaso[cont,3]) and (ycor <= nopaso[cont,4])
      and (menos<>cont) then checa_res:=true;
{       cuadro(nopaso[cont,1],nopaso[cont,2],nopaso[cont,3],nopaso[cont,4],3) }
    end;
End;
```

CHECA QUE SE SOLICITE EL MENSAJE

```
Function Checa_men(xcor,ycor:integer; Var Nopaso : restric): integer;
Var
  cont : byte;
  posi : byte;
Begin
  checa_men:=0;
```

```
  posi:=0;
  for cont:= 1 to Nopaso[0,1] do
    begin
      if (xcor >= nopaso[cont,1]) and (ycor >= nopaso[cont,2])
      and (xcor <= nopaso[cont,3]) and (ycor <= nopaso[cont,4]) then
        begin
        if nopaso[cont,6]=1 then
          begin
            posi:=cont;
            nopaso[cont,6]:=2;
          end
          else if nopaso[cont,6]<>2 then posi:=cont;
        end;
{     cuadro(nopaso[cont,1],nopaso[cont,2],nopaso[cont,3],nopaso[cont,4],3)}
    end;
  if posi<>0 then checa_men:=nopaso[posi,5];
End;
```

INICIA UNA ANIMACION

```
Procedure Animar.Iniciaani(Var D : DatosAnima);
Begin
  D.tamani:=0;
  getmem(D.animacion,D.tamani);
End;
```

FINALIZA UNA ANIMACION

```
Procedure Animar.FinalizaAni(Var D : DatosAnima);
Begin
  freemem(D.animacion,D.tamani);
End;
```

LOCALIZA EL PUNTO PRINCIPAL DE UN DUENDE (LARGO / 2) = (ANCHO)

```
Procedure Animar.Puntoduende(mono:duende; Var D : datosanima;
          var xvir, yvir : integer; numero:byte);
Var
  x2   : word;
Begin
  xvir:=mem[seg(mono.dnd[numero]^):ofs(mono.dnd[numero]^)+0]+
        mem[seg(mono.dnd[numero]^):ofs(mono.dnd[numero]^)+1]*256;
  yvir:=mem[seg(mono.dnd[numero]^):ofs(mono.dnd[numero]^)+2]+
        mem[seg(mono.dnd[numero]^):ofs(mono.dnd[numero]^)+3]*256;
  x2:=xvir div 2;
  D.puntox:=x2;
  D.puntoy:=yvir;
  xvir:=d.x+x2;
  yvir:=d.y+yvir;
End;
```

LECTURA DE ARCHIVOS DE DISCO

```
Procedure Animar.Lee_Archivo(nombre:string; Var D : DatosAnima);
Var
  archi : file;
Begin
  assign(archi,nombre);
  {$I-}
  reset(archi,1);
  {$I+}
  if IOresult=0 then
    begin
      D.Tamani:=filesize(archi);
      Getmem(D.animacion,D.Tamani);
      Blockread(archi,D.animacion^,D.tamani);
      close(archi);
    end
  else
```

```
    Begin
      rewrite(archi,1);
      close(archi);
      D.tamani:=0;
      getmem(D.animacion,D.tamani);
    end;
End;
```

ESCRITURA DE ARCHIVOS EN DISCO

```
Procedure Animar.Escribe_Archivo(nombre:string; D : DatosAnima);
Var
  archi : file;
Begin
  assign(archi,nombre);
  rewrite(archi,1);
  Blockwrite(archi,D.animacion^,D.tamani);
  close(archi);
End;
```

BUSQUEDA DE ANIMACIONES EN MEMORIA POR NUMERO DE REGISTRO

```
Procedure animar.Busca_anima(numero : word; Var D : DatosAnima);
Var
  posicion : word;
Begin
  posicion:=(numero-1)*registroanima;
  move(mem[seg(D.animacion^):ofs(D.animacion^)+
       posicion],D.aniact,registroanima);
end;
```

ESCRITURA DE ANIMACION EN MEMORIA POR NUMERO DE REGISTRO

```
Procedure animar.Pon_anima(numero : word; Var D : DatosAnima);
Var
  posicion : word;
Begin
  D.aniact.etiqueta:=espacios(D.aniact.etiqueta,8,2);
  posicion:=(numero-1)*registroanima;
  move(D.aniact,mem[seg(D.animacion^):
       ofs(D.animacion^)+posicion],registroanima);
end;
```

INSERTA ANIMACIONES EN MEMORIA EN LA ULTIMA POSICION

```
Procedure animar.Inserta_Anima(numero : word; Var D : DatosAnima);
Var
  posicion : word;
  tamresp  : word;
  mayor    : word;
  extra    : anima;
  resp     : pointer;
Begin
  D.aniact.estado:=0;
  D.aniact.etiqueta:=espacios('',8,2);
  D.aniact.imagen:=0; D.aniact.duracion:=0;
  D.aniact.increx:=0; D.aniact.increy:=0;
  tamresp:=D.tamani; getmem(resp,tamresp);
  move(D.animacion^,resp^,tamresp);
  freemem(D.animacion,D.tamani);
  inc(D.tamani,RegistroAnima);
  getmem(D.animacion,D.Tamani);
  move(resp^,D.animacion^,tamresp);
  posicion:=(numero-1)*RegistroAnima;
  move(D.aniact,mem[seg(D.animacion^):
       ofs(D.animacion^)+posicion],RegistroAnima);
  move(mem[seg(resp^):ofs(resp^)+posicion],
```

```
  mem[seg(D.animacion^):ofs(D.animacion^)+
      posicion+registroanima],tamresp-((numero-1)*RegistroAnima));
  freemem(resp,tamresp);
end;
```

BORRA UNA ANIMACION DE MEMORIA SEGUN EL NUMERO DE REGISTRO

```
Procedure animar.Borra_Anima(Lugar : word; Var D : DatosAnima);
Var
  posicion : word;
  tamresp  : word;
  resp     : pointer;
Begin
  tamresp:=D.Tamani;
  dec(D.Tamani,RegistroAnima);
  getmem(resp,D.Tamani);
  lugar:=lugar-1;
  move(D.animacion^,resp^,(tamresp)-((tamresp-(lugar*RegistroAnima))));
  if (lugar*RegistroAnima+RegistroAnima)<(tamresp) then
    begin
      posicion:=(lugar+1)*RegistroAnima;
      move(mem[seg(D.animacion^):ofs(D.animacion^)+posicion],
      mem[seg(resp^):ofs(resp^)+posicion-RegistroAnima],
      tamresp-((lugar+1)*RegistroAnima));
    end;
  freemem(D.animacion,tamresp);
  getmem(D.animacion,D.Tamani);
  move(resp^,D.animacion^,D.Tamani);
  freemem(resp,D.Tamani);
end;
```

DEVUELVE EL NUMERO DE REGISTRO A PARTIR DE LA ETIQUETA DE LA ANIMACION

```
Function Animar.Busca_Etiqueta(eti:string; Var D : DatosAnima) : word;
Var
  cont : word;
  ok   : boolean;
  actual : anima;
Begin
  eti:=espacios(eti,8,2);
  ok:=false;
  for cont :=1 to (D.tamani div Registroanima) do
    begin
      busca_anima(cont,D);
      if D.aniact.estado=1 then
        begin
          if D.aniact.etiqueta=eti then
            begin
              ok:=true;
              Busca_Etiqueta:=cont;
              Cont := (D.tamani div RegistroAnima);
            end;
        end;
    end;
  if ok=false then Busca_Etiqueta:=9999;
End;
```

ASISTENTE CON EL QUE SE OBTIENEN LAS COORDENADAS DE UN DUENDE UBICANDOLO EN LA PANTALLA DE FONDO

```
PROCEDURE Animar.Ubica_duende(Var boton : word; Var xi,yi: integer;
          mono : duende; num : byte);
Var
  Cad, Cad2 : String;
  Rat       : Raton;
  x,y       : Word;
  Tecla, tecla2 : Char;
begin
  Tecla := #1;  Tecla2 := #0;
```

```
  Rat.Posraton(boton,x,y);
  If keypressed Then
  begin
    x := xi;   y := yi;
    Tecla := Readkey;
    Case tecla of
       #0 : begin
              Tecla2 := Readkey;
              Case Tecla2 of
                #72 : If y>0 Then Dec(y);
                #80 : If y<199 Then Inc(y);
                #75 : If x>0 Then Dec(x);
                #77 : If x<319 Then Inc(x);
              end;
            end;
      '8' : If y>10  Then Dec(y,10) Else y := 0;
      '2' : If y<189 Then Inc(y,10) Else y := 199;
      '4' : If x>10  Then Dec(x,10) Else x := 0;
      '6' : If x<319 Then Inc(x,10) Else x := 319;
      #27 : Boton := 2;
    end;
    IniciaRaton(x,y);
    Tecla2 := #1;
  end;
  if (xi<>x) or (yi<>y) Or (Tecla2<>#0) Then
  begin
    leexms(xmsman[30],pantemporal^,64000,128000); {Pantalla de fondo actual}
    putduende(x,y,mono,pantemporal^,num);
    despliega(pantemporal);
{    Cuadro(x,y,x+MemW[Seg(Mono.Dnd[Num]^):Ofs(Mono.Dnd[Num]^)],
              y+MemW[Seg(Mono.Dnd[Num]^):Ofs(Mono.Dnd[Num]^)+2],5); }
    Str(X:3,Cad2);  Cad := 'X='+Cad2;
    Str(Y:3,Cad2);  Cad := Cad+'  Y='+Cad2;


    If y>16 Then
    begin
       Letrero(1,1,Cad,5);
       Cuadrado2(8,8,103,15,2);
       Cuadro(6,6,105,17,255);
       Cuadro(7,7,104,16,0);
    end
    else begin
       Letrero(26,23,Cad,5);
       Cuadrado2(208,184,303,190,2);
       Cuadro(206,182,305,192,255);
       Cuadro(207,183,304,191,0);
    end;
    xi := x;  yi := y;
    If Boton<>2 Then Boton := 9;
  end;
end;
```

GENERA UN RENGLON DEL EDITOR DE ANIMACION

```
Procedure Animar.Renglon_anima(cory : byte; D : DatosAnima; mono:duende;
          estado : boolean);
Var
 color : byte;
 valor : word;
Begin
  if estado then color:=8 else color:=10;
  letrero(2,cory+16,
  ceros(nac(D.aniact.Estado),2,1)+' '+espacios(D.aniact.Etiqueta,8,2)+
  espacios(nac(D.aniact.Imagen),3,1)+
  espacios(nac(D.aniact.Duracion),4,1)+espacios(nac(D.aniact.increx),5,1)+
  espacios(nac(D.aniact.increy),5,1)+' ',color);
  cuadro(228,180,256,193,1);
```

```
  Letrero(33,20,Espacios(Nac(MaxAvail),6,2)+'B',5);
  Cuadrado2(33*8,20*8,319,21*8-1,2);
  if (D.aniact.imagen<>0) and (estado) then
    begin
      venx1:=8; veny1:=14; venx2:=253; veny2:=117;
      cursorraton(false);
      cuadrado(7,13,255,117,0);
      if (d.aniact.estado in [0,1,2]) and (d.aniact.imagen<>0) then
        putduende(10,15,mono,mem[$a000:0],D.aniact.imagen);
      cuadrado(229,181,255,192,0);
      letrero(29,23,nac(D.aniact.imagen),2);
      cursorraton(true);
    end;
  if (D.aniact.estado=3) and (estado) then
    begin
      valor:=busca_etiqueta(d.aniact.etiqueta,D);
      if valor=9999 then mensaje('La etiqueta no ha sido definida',1);
    end;
End;
```

GENERA TODOS LOS RENGLONES DEL EDITOR DE ANIMACION

```
Procedure Animar.Lista_anima(numero : word; cory : byte;
          D:DatosAnima; Mono : Duende);
Var
  Cont : word;
Begin
  if (D.tamani div registroanima) > 4 then
    begin
      for cont:=numero to numero+3 do

        begin
          Busca_anima(cont,D);
          Renglon_anima(cory,D,mono,false);
          inc(cory);
        end;
    end
  else
    begin
      if (D.tamani div registroanima) > 0 then
      for cont:=1 to  (D.tamani div registroanima) do
        begin
          Busca_anima(cont,D);
          Renglon_anima(cory,D,mono,false);
          inc(cory);
        end;
    end;
End;
```

UTILIZADO PARA GENERAR LAS DISTINTAS ACCIONES DEL EDITOR DE ANIMACION

```
Procedure Animar.Otra_Anima(Var D:DatosAnima);
Var
  salir : boolean;
  f     : formato;
  panm  : pointer;
Begin
  salir:=false;
  While salir<>true do
    begin
      case (D.aniact.estado) of
        4 : begin {Redefine movimiento de otra animacion}
              if anis[d.aniact.imagen].tamani>0 then
              Primer_Anima(anis[d.aniact.imagen],d.aniact.etiqueta);
            end;
        5 : Begin
              Bandera_Mus := d.aniact.imagen;
            end;
```

```
        6 : Begin {Activa frases}
              Retardo_Frases := d.aniact.duracion;
              Numero_Frase   := d.aniact.imagen;
              If d.aniact.IncreX>0 Then Color_Frase := d.aniact.IncreX;
              if d.aniact.increy=1 then
                begin
                  delay(100);
                  move(mem[$a000:0],pantemporal^,64000);
                  Pon_Frase(Numero_Frase, Color_Frase,false);
                  espera2(retardo_frases*500);
                  retardo_frases:=0;
                  despliega(pantemporal);
                end;
            End;
        7 : Begin
              Retardo_Frases := 0;
              Numero_Frase   := 0;
              Color_Frase    := 255;
            End;
        8 : Begin
{             Lee_ExtVOC('..\sb\'+d.aniact.etiqueta+'.VOC'); }
              Repite_VOC := d.aniact.duracion;
              Marca_VOC  := d.aniact.imagen;
              Toca_ExtVOC;
            End;

        9 : begin {Carga un pcx directo a pantalla}
              f.verpcx('..\PCX\'+d.aniact.etiqueta+'.pcx',pantemporal^);
              despliega(pantemporal);
            end;
       10 : begin {Tiempo de espera}
              espera2(d.aniact.duracion*d.aniact.imagen*d.aniact.increx);
            end;
       11 : begin {Activa y desactiva el cursor}
              case (d.aniact.imagen) of
                1 : inter:=true;
                0 : inter:=false;
              end;
            end;
       12 : begin {Redefinir una animacion}
              anis[d.aniact.imagen].prota:=d.aniact.duracion;
            end;
       13 : begin {Definir banderas}
              banderas[d.aniact.imagen]:=d.aniact.duracion;
            end;
       14 : begin {Redefine ventana de accion}
              venx1:=d.aniact.imagen;
              veny1:=d.aniact.duracion;
              venx2:=d.aniact.increx;
              veny2:=d.aniact.increy;
            end;
       15 : begin {Salta una animacion a un lugar absoluto}
              anis[d.aniact.imagen].x:=d.aniact.increx;
              anis[d.aniact.imagen].y:=d.aniact.increy;
            end;
       16 : begin {Activa una nueva escena}
              escenamax:=d.aniact.imagen;
            end;
      end;
      inc(D.pila[D.nivel].posicion);
      D.duracion:=0;
      Busca_anima(D.pila[D.nivel].posicion,D);
      if d.aniact.estado in [0,1,2,3] then salir:=true;
    end
End;
```

EJECUTA UNA ANIMACION (RUTINA MAS IMPORTANTE)

```
Procedure Animar.Corre_Anima(nombre:string; Var D:DatosAnima;
                             Var Mono:Duende);
{Prota    : es utilizado para validar restricciones en duendes protagonistas }
{Nombre   : indica la etiqueta inicial en la que empezara la animacion       }
Var
  vuelta : word;
  salir  : boolean;
  Posx,Posy : integer;
Begin
  if D.aniact.estado = 3 then   {Salto a subrutina                           }
    Begin
      D.duracion:=0;
      inc(D.nivel);
      D.pila[D.nivel].ciclo:=1;

      case (D.aniact.duracion) of
        {Se sustituye por el nuevo valor}
        999 : D.pila[d.nivel].tociclo:=d.nuevadur;



        {Se sustituye por un valor aleatorio}
        998 : Begin
                D.pila[d.nivel].tociclo:=random(D.aniact.imagen)+1;
              end;
        else D.pila[D.nivel].tociclo:=D.aniact.duracion;
      end;
      if D.aniact.duracion=900 then begin D.inc2x:=0; D.inc2y:=0; end;
      D.pila[D.nivel].vuelta:=false;
      D.pila[D.nivel].posicion:=0;
      D.pila[D.nivel].origen:=busca_etiqueta(nombre,D);
    end;
  {Control de ciclos }
  if (D.pila[D.nivel].vuelta=true)
     and (D.pila[D.nivel].ciclo<D.pila[d.nivel].tociclo) then
    begin
      if (D.pila[d.nivel].tociclo<>900) then inc(d.pila[D.nivel].ciclo);
      D.pila[D.nivel].posicion:=d.pila[D.nivel].origen;
      D.duracion:=0;
      Busca_anima(D.pila[D.nivel].posicion,D);
      nombre:=D.aniact.etiqueta;
      d.pila[D.nivel].vuelta:=false;
    end;
  if D.duracion=0 then {Control de cambio de imagen}
    Begin
      if D.aniact.estado = 1 then
        begin
          D.pila[D.nivel].posicion:=busca_etiqueta(nombre,D);
          Busca_anima(D.pila[D.nivel].posicion,D);
        end;
      if D.prota=1 then {Incrementa X y Y para un protagonista}
        begin
          PuntoDuende(monos[d.duende],D,posx,posy,D.aniact.imagen);
          with D do
          begin
           if checa_res(posx+inc2x+aniact.increx,posy+inc2y+
              aniact.increy,nopaso,restric) Then
             begin
               if checa_res(posx+inc2x+aniact.increx,posy,nopaso,restric) Then
                 begin
                   if checa_res(posx,posy+inc2y+aniact.increy,
                     nopaso,restric) Then
                     begin
                       if d.nivel>1 then dec(d.nivel);
                     end
                   else
                     begin
```

```
                        inc(D.y,D.aniact.increy+D.inc2y);
                        if abs(anis[1].yfinal-anis[1].y)<3 then
                        begin
                          D.xfinal:=D.x;
                          D.nuevadur:=0;
                          D.pila[D.nivel].ciclo:=D.pila[D.nivel].tociclo;
                        end;
                      end
                  end
                else
                  begin
                    inc(D.x,D.aniact.increx+D.inc2x);
                    if abs(anis[1].xfinal-anis[1].x)<5 then
                    begin
                      D.yfinal:=D.y;
                      D.nuevadur:=0;
                      D.pila[D.nivel].ciclo:=D.pila[D.nivel].tociclo;
                    end;
                  end;
              end
            else
              begin
                inc(D.x,D.aniact.increx+D.inc2x);
                inc(D.y,D.aniact.increy+D.inc2y);
              end;
          end;
      end
    else
      begin
        inc(D.x,D.aniact.increx+D.inc2x);
        inc(D.y,D.aniact.increy+D.inc2y);
      end; {Fin de rutinas especiales para protagonista}
  end;

{Control de tiempo}
if (D.duracion=D.aniact.duracion) and (D.aniact.estado=2) then
  Begin
    Busca_anima(D.pila[D.nivel].posicion,D);
    if d.aniact.imagen<>0 then
    putduende(D.x,D.y,mono,pantemporal^,D.aniact.imagen);
    D.pila[D.nivel].vuelta:=true;
    if ((d.nivel>1)) and
       (D.pila[d.nivel].ciclo=D.pila[d.nivel].tociclo) then
      begin
        dec(d.nivel);
        inc(d.pila[d.nivel].posicion);
        D.duracion:=0;
        Busca_anima(D.pila[D.nivel].posicion,D);
        if D.aniact.estado>3 then Otra_anima(D);
      end;
    exit;
  End;
case (D.aniact.estado) of {Despliegue en pantemporal de la imagen actual}
 2,1,0 : begin
           if D.aniact.duracion <> 900 then inc(D.duracion);
           if d.aniact.imagen<>0 then
           putduende(D.x,D.y,mono,pantemporal^,D.aniact.imagen);
           if (D.duracion>D.aniact.duracion) then
             Begin
               inc(D.pila[D.nivel].posicion);
               D.duracion:=0;
               Busca_anima(D.pila[D.nivel].posicion,D);
               if D.aniact.estado>3 then Otra_anima(D);
             end;
         end;
  End;
End;
```

DA UNA ETIQUETA DE ENTRADA A LAS ANIMACIONES

```
Procedure animar.Primer_Anima(var D:DatosAnima; nombre:string);
Begin
  D.duracion:=0;
  D.nivel:=1;
  D.inc2x:=0;
  D.inc2y:=0;
  D.pila[D.nivel].tociclo:=1;
  D.pila[D.nivel].ciclo:=1;
  D.pila[D.nivel].vuelta:=false;
  D.pila[D.nivel].posicion:=0;
  D.pila[D.nivel].origen:=busca_etiqueta(nombre,D);
  if D.pila[d.nivel].origen=9999 then
    mensaje('La etiqueta '+nombre+' no ha sido definida',1);
End;
```

MODULO DE CAPTURA DEL EDITOR DE ANIMACION

```
Procedure animar.Zona_anima(Var mono : duende; Var DA : DatosAnima);
Var
  ContAni : integer;
  tecla   : char;
  tecla1  : char;
  tecla2  : char;
  cory    : integer;
  CadAux  : string;
  opcion  : byte;
  codigo  : integer;
  D       : dir_mcga;
  extra   : word;        {Variable auxiliar                    }
  boton   : word;
  DB      : DatosAnima;
  mono2   : duende;
Begin
  D.Define_Unidades;
  cuadrado(260,10,316,100,2);
  letrero(33,2,'ANIMAR',4);
  cuadro(260,10,316,100,1);
  Putimagen(265,33,bot.dnd[3]);
  Putimagen(292,33,bot.dnd[1]);
  Putimagen(292,80,bot.dnd[31]);
  Putimagen(265,57,bot.dnd[21]);
  cursorraton(true);
  LimiteRaton(260*2,10,316*2,100);

  cuadrado(10,132,252,169,1); Cuadro(10,132,252,169,2);
  cuadrado(10,118,252,129,1); Cuadro(10,118,252,129,2);
  cuadrado(16,135,247,167,0);
  Letrero(2,15,' E ETIQUETA ND DUR INCX INCY ',7);
  tecla:=#255; tecla2:=#255;
  ContAni:=0; cory:=1;
  opcion:=1;
  while tecla<>#27 do
    Begin
      Icono
        LEER DE DISCO
      if Checa_boton(265,33,bot,3, '1') then
      Begin
        cursorraton(false);
        If Pos('.anm',nomanima)=0 Then nomanima:='..\anm\*.anm';
        LimiteRaton(12*2,13,304*2,153);
        if D.lee_dirarch(Nomanima, false) then
          begin
            freemem(DA.Animacion,DA.TamAni);
            lee_archivo(nomanima,DA);
            if DA.TamAni>0 then
```

```
          begin
            ContAni:=1; cory:=1;
            lista_anima(contani,1,DA,mono);
            Busca_anima(ContAni,DA);
            Renglon_Anima(cory,DA,mono,true);
          end;
      end;
    LimiteRaton(260*2,10,316*2,100);
    cursorraton(true);
  end;
    Icono
       ESCRIBIR EN DISCO
if Checa_boton(292,33,bot,1, 'a')
  and ((DA.tamani div registroanima)>0) then
  Begin
    cursorraton(false);
    If Pos('.anm',nomanima)=0 Then nomanima:='..\anm\*.anm';
    LimiteRaton(12*2,13,304*2,153);
    if D.lee_dirarch(Nomanima, true) then
      begin
        escribe_archivo(nomanima,DA);
      end;
    LimiteRaton(260*2,10,316*2,100);
    cursorraton(true);
  end;
Icono
  RECREAR ANIMACION
if Checa_boton(265,57,bot,21, 'a') and (DA.AniAct.estado=1)
   and (mono.total>0) and (contAni>0) then
  begin
    cursorraton(false);
    LimiteRaton(0,0,319*2,199);
    extra:=1; while xmsman[extra]<>0 do inc(extra);
    if ReservaXms(63,xmsman[extra]) = 0 then;
    VideoPantalla(pantemporal);
    escribexms(xmsman[extra],pantemporal^,64000,0);
    {Pantalla de fondo actual}
    leexms(xmsman[30],pantemporal^,64000,128000);
    despliega(pantemporal);
    venx1:=0; veny1:=0; venx2:=320; veny2:=199;
    DA.x:=0; DA.y:=0;
    while numbotonraton=0 do
      Ubica_Duende(Boton,DA.x,DA.y,mono,DA.AniAct.imagen);
    if boton<>2 then
      begin
        libera_Boton;
        Primer_Anima(DA,DA.aniact.etiqueta);
        while (not keypressed) And (NumBotonRaton=0) do
          begin
            {Pantalla de fondo actual importada}
            leexms(xmsman[30],pantemporal^,64000,128000);
            Corre_Anima(DA.AniAct.etiqueta,DA,mono);
            despliega(pantemporal);
          end;
      end;
    While Keypressed Do keypressed;
    Libera_Boton;
    {Pantalla de captura de animacion actual}
    leexms(xmsman[extra],mem[$A000:0],64000,0);
    if LiberaXms(xmsman[extra]) = 0 then; xmsman[extra]:=0;
    LimiteRaton(260*2,10,316*2,100);
    Busca_anima(ContAni,DA);
    Renglon_Anima(cory,DA,mono,true);
    cursorraton(true);
  end;
Icono
```

```
   SALIR DE LA RUTINA
if Checa_boton(252,80,bot,31,#27) then Begin tecla:=#27 end;

  Teclado
    MODULO DE CAPTURA
if keypressed then
  begin
    tecla:=readkey;
    if (tecla=' ') and (contani>0) then
      begin
        Busca_anima(ContAni,DA);
        while (tecla<>#13) and (tecla2<>#72)
              and (tecla2<>#80) and (tecla<>#27) do
          begin
            case (opcion) of
              1 : Begin
                   letrero(2,cory+16,'  ',3);
                   Leebyte(2,cory+16,2,tecla,tecla2,DA.AniAct.estado,3);
                  end;
              2 : Begin
                   cadaux:=DA.AniAct.etiqueta;
                   LeeCadena(5,cory+16,8,1,tecla,tecla2,cadaux,3);
                   DA.AniAct.etiqueta:=cadaux;
                  end;
              3 : Begin
                   letrero(14,cory+16,'  ',3);
                   Leebyte(14,cory+16,2,tecla,tecla2,
                           DA.AniAct.imagen,3);
                    if (DA.AniAct.imagen>mono.total)
                       and (da.aniact.estado in [0,1,2]) then
                      begin
                        DA.AniAct.imagen:=mono.total;
                        letrero(14,cory+16,
                        espacios(nac(DA.AniAct.imagen),2,2),3);
                      end;
                  end;
              4 : Begin
                   letrero(17,cory+16,'    ',3);
                   Leeword(17,cory+16,3,tecla,tecla2,
                           DA.AniAct.duracion,3);
                  end;
              5 : Begin
                   letrero(21,cory+16,'    ',3);
                   Leeinteger(21,cory+16,4,tecla,tecla2,
                           DA.AniAct.increx,3);
                  end;
              6 : Begin
                   letrero(26,cory+16,'    ',3);
                   Leeinteger(26,cory+16,4,tecla,tecla2,
                           DA.AniAct.increy,3);
                  end;
            end;
            if (tecla=#0) and (tecla2=#77) then inc(opcion);
            if (tecla=#0) and (tecla2=#75) then dec(opcion);
            case (da.aniact.estado) of
              2 : da.aniact.etiqueta:='FINAL';
              7 : da.aniact.etiqueta:='DES_LET';
             10 : da.aniact.etiqueta:='ESPERA';
             11 : da.aniact.etiqueta:='CURSOR';
             12 : da.aniact.etiqueta:='DEFANIMA';
             13 : da.aniact.etiqueta:='BANDERAS';
             14 : da.aniact.etiqueta:='VENTANA';
             15 : da.aniact.etiqueta:='X_Y_ABSO';

              16 : da.aniact.etiqueta:='ESCENA';
            end;
```

```
        Renglon_Anima(cory,DA,mono,true);
        if opcion<1 then opcion:=6;
        if opcion>6 then opcion:=1;
      end;
    if tecla<> #27 then pon_anima(contAni,DA)
    else
      begin
        Busca_anima(ContAni,DA);
        Renglon_Anima(cory,DA,mono,true);
        tecla:=#255;
      end;
  end;
if tecla=#0 then
  begin
    if tecla2=#255 then tecla:=readkey;

    {Cursor Arriba}
    if ((tecla=#72) or (tecla2=#72))
      and ((DA.TamAni div RegistroAnima)>0) and (contani<>1) Then
      Begin
        Renglon_Anima(cory,DA,mono,false);
        dec(ContAni); if ContAni<1 then inc(ContAni)
        else dec(cory);
        if cory < 1 then
          begin
           cory:=1;
           Lista_Anima(contani,cory,DA,mono);
          end;
        Busca_anima(ContAni,DA);
        Renglon_Anima(cory,DA,mono,true);
        tecla2:=#255; tecla:=#255;
      End;
    {Cursor Abajo}
    if ((tecla=#80) or (tecla2=#80))
       and ((DA.TamAni div RegistroAnima)>0)
    and (ContAni< (DA.TamAni div RegistroAnima)) then
      begin
        Renglon_Anima(cory,DA,mono,false);
        inc(ContAni); if ContAni>(DA.TamAni div RegistroAnima)
        then dec(ContAni)
        else inc(cory);
        if cory > 4 then
          begin cory:=4; Lista_Anima(ContAni-3,cory-3,DA,mono); end;
        Busca_anima(ContAni,DA);
        Renglon_Anima(cory,DA,mono,true);
        tecla2:=#255; tecla:=#255;
      end;
    {Pagina Atras}
    if ((tecla=#73)  or (tecla2=#73))
      and ((DA.TamAni div RegistroAnima)>0) and (ContAni<>1) Then
      begin
        Renglon_Anima(cory,DA,mono,false);
        dec(ContAni,3); if ContAni<1 then
          begin
            ContAni:=1;
            cory:=1;
            Lista_Anima(ContAni,cory,DA,mono);
          end
        else dec(cory,3);

        if cory < 1 then
          begin cory:=1; Lista_Anima(ContAni,cory,DA,mono);end;
        Busca_anima(ContAni,DA);
        Renglon_Anima(cory,DA,mono,true);
        tecla2:=#255; tecla:=#255;
      end;
    {Pagina Adelante}
```

```
if ((tecla=#81) or (tecla2=#81))
  and ((DA.TamAni div RegistroAnima)>0)
and (ContAni< (DA.TamAni div RegistroAnima)) Then
  begin
    Renglon_Anima(cory,DA,mono,false);
    inc(ContAni,3);
    if ContAni>(DA.TamAni div RegistroAnima) then
    begin
      ContAni:=(DA.TamAni div RegistroAnima);
      if (DA.TamAni div RegistroAnima)>4 then
        begin
          cory:=4; Lista_Anima(ContAni-3,cory-3,DA,mono);
        end
      else begin cory:=(DA.TamAni div RegistroAnima);
      Lista_Anima(ContAni-(DA.TamAni div RegistroAnima)+1
      ,cory-(DA.TamAni div RegistroAnima)+1,DA,mono); end;
    end
    else inc(cory,3);
    if cory >4 then
      begin
        cory:=4;
        Lista_Anima(ContAni-3,cory-3,DA,mono);
      end;
    Busca_anima(ContAni,DA);
    Renglon_Anima(cory,DA,mono,true);
    tecla2:=#255; tecla:=#255;
  end;
{Insertar nueva animacion}
if ((tecla=#82) or (tecla2=#82)) Then
  begin
    inc(contani);
    Inserta_anima(contani,DA);
    if ((DA.TamAni div RegistroAnima)>0) then
      begin
        if (contani-cory+4>(DA.TamAni div RegistroAnima))
           and ((DA.TamAni div RegistroAnima)>4)
         then dec(contani);
        lista_anima(contani-cory+1,1,DA,mono);
        if ((DA.TamAni div RegistroAnima)<=4) then
          begin
            cory:=contani;
          end;
      end;
    Busca_anima(ContAni,DA);
    Renglon_Anima(cory,DA,mono,true);
    tecla2:=#255; tecla:=#255;
  end;
{Borrar una animacion}
if ((tecla=#83) or (tecla2=#83))
  and ((DA.TamAni div RegistroAnima)>0) Then
  begin
    borra_anima(contani,DA);
    if contani>(DA.TamAni div registroanima)
      then contani:=(DA.TamAni div RegistroAnima);

    if cory>(DA.TamAni div RegistroAnima)
      then cory:=(DA.TamAni div RegistroAnima);
    if ((DA.TamAni div RegistroAnima)=0) then
      begin
        letrero(2,(DA.TamAni div RegistroAnima)+17
        ,'                                    ',3);
      end;
    if ((DA.TamAni div RegistroAnima)>0) then
      begin
        if (contani-cory+4>(DA.TamAni div RegistroAnima))
           and ((DA.TamAni div RegistroAnima)>3)
         then dec(contani);
```

```
                    lista_anima(contani-cory+1,1,DA,mono);
                    if ((DA.TamAni div RegistroAnima)<=3) then
                      letrero(2,(DA.TamAni div RegistroAnima)+17
                      ,'                                  ',3);
                    if contani>0 then
                     begin
                      Busca_anima(contani,DA);
                      Renglon_anima(cory,DA,mono,true);
                     end;
                  end;
                tecla2:=#255; tecla:=#255;
              end;
            tecla:=#255; tecla2:=#255;
          end;

      end;
  end;
  LimiteRaton(0,0,319*2,199);
End;

Begin
  For Bot.Total := 1 To 10 Do Monos[Bot.Total].Total := 0;
  Bot.Total:= 0;
  Bandera_Mus := 0;
END.
```

LibDir.pas : Libreria para el manejo archivos y directorios.

```
UNIT LibDir;

INTERFACE
uses Crt, Dos, LibGra, LibUti, LibRat, LibXMS;


TYPE
   DIR_MCGA = OBJECT(Raton)
      Unidades : Array[1..26] Of Boolean;
      FUNCTION  Num_Floppys : Byte;
      PROCEDURE LeeDirectorio(Var Dir : Pointer; Var Talla : Word;
                  Via : String; Tipo : Byte);
      PROCEDURE Obten_Arch(Dir : Pointer; Indice : Word;
                  Var DirInfo : SearchRec);
      PROCEDURE Define_Unidades;
      FUNCTION  Checa_Via(Via_Inicial : String;
                  Var DirAct, Mascara : String) : Boolean;

      FUNCTION  Lee_DirArch(Var Via_Inicial : String;
                  Salvar : Boolean) : Boolean;
      FUNCTION  Click_Raton(Var Boton, Yr, Min, Act, Max : Word;
                                  Rango : Boolean) : Boolean;
   End;


IMPLEMENTATION
```

FUNCION QUE REGRESA EL NUMERO DE UNIDADES DE DISCOS FLEXIBLES QUE ESTAN INSTALADAS EN LA COMPUTADORA

```
FUNCTION Dir_MCGA.Num_Floppys : Byte;
Var
   Config : Word;
Begin
   Config := MemW[$40:$10];
   If Odd(Config) Then Num_Floppys := ((Config And 192) Div 64)+1
    Else Num_Floppys := 0;
End;
```

OBTIENE EL VALOR DE UN REGISTRO EN UNA POSICION DADA

```
Procedure DatoDir(D : Pointer; Var DirReg : SearchRec; Lugar : Word);
Begin
   Move(Mem[Seg(D^):Ofs(D^)+(Lugar-1)*SizeOf(SearchRec)],
        DirReg, SizeOf(SearchRec));
End;
```

GUARDA EL VALOR DE UN REGISTRO EN UNA POSICION DADA

```
Procedure PonDatoDir(Var D : Pointer; Var DirReg : SearchRec; Lugar : Word);
Begin
   Move(DirReg, Mem[Seg(D^):Ofs(D^)+(Lugar-1)*SizeOf(SearchRec)],
        SizeOf(SearchRec));
End;
```

ACOMODA LOS NOMBRES DE LOS ARCHIVOS EN FORMA ALFABETICA POR MEDIO DEL ALGORITMO DE ORDENAMIENTO RAPIDO.

```
procedure Ordena(Var A: Pointer; Lo, Hi: Integer);
```

```
procedure RapOrd(Izq, Der: Integer);
var
  i, j     : Integer;
  DReg,
  DReg2,
  x, y     : SearchRec;
  Salir    : Boolean;
begin
  i := Izq;
  j := Der;
  DatoDir(a, x, (Izq+Der) DIV 2);
  repeat
    Salir := False;
    Repeat
      DatoDir(a, DReg, i);
      If DReg.Name < x.Name Then Inc(i) Else Salir := True;
    Until Salir;
    Salir := False;
    Repeat
      DatoDir(a, DReg, j);
      If x.Name < DReg.Name Then Dec(j) Else Salir := True;
    Until Salir;

    if i <= j then
    begin
      DatoDir(a, DReg, i);
      DatoDir(a, DReg2, j);
      y := Dreg;
      PonDatoDir(a, DReg2, i);  { a[i] := a[j]; }
      PonDatoDir(a, y, j);      { a[j] := y; }

      i := i + 1; j := j - 1;
    end;
  until i > j;
  if Izq < j then RapOrd(Izq, j);
  if i < Der then RapOrd(i, Der);
end;

begin
  RapOrd(Lo,Hi);
end;

{ Fin de procedimientos de ordenación }
```

```
LEE TODOS LOS NOMBRES DE ARCHIVOS Y DIRECTORIOS QUE SE ENCUENTREN EN <<VIA>>. SE
PUEDE ESPECIFICAR TAMBIEN UNA MASCARA
PROCEDURE Dir_MCGA.LeeDirectorio(Var Dir : Pointer; Var Talla : Word;
                                 Via : String; Tipo : Byte);
Var
  DI2,
  DirInfo : SearchRec;
  p       : Pointer;
Begin
  If (Talla>0) And (Dir<>Nil) Then FreeMem(Dir, Talla*SizeOf(DirInfo));
  Talla := 0;
  Dir := Nil;
  DosError:=0;
  FindFirst(Via, Tipo, DirInfo);
  If (Tipo=$10) Then
    If DirInfo.Name='.' Then DirInfo.Name := '\'
    Else
      If Via[0]>#6 Then
      Begin
        GetMem(Dir, 2*SizeOf(DirInfo));
        DI2.Name := '\';
        Move(DI2, Mem[Seg(Dir^):Ofs(Dir^)], SizeOf(DirInfo));
```

```
        DI2.Name := '..';
        Move(DI2, Mem[Seg(Dir^):Ofs(Dir^)+SizeOf(DirInfo)],
          SizeOf(DirInfo));
        Talla := 2;
      End;
  While DosError = 0 do
  begin
    If ((DirInfo.Attr And Tipo)>0) Or
       (((Tipo And $10)=0) And (DirInfo.Attr=0)) Then
    begin
      GetMem(p, (Talla+1)*SizeOf(DirInfo));
      If Talla>0 Then
      begin
        Move(Dir^, p^, Talla*SizeOf(DirInfo));
        Freemem(Dir, Talla*SizeOf(DirInfo));
      end;
      Dir := p;
      Move(DirInfo, Mem[Seg(Dir^):Ofs(Dir^)+Talla*SizeOf(DirInfo)],
           SizeOf(DirInfo));
      Inc(Talla);
    End;
    FindNext(DirInfo);
  end;
  If Talla>0 Then
  begin
    If (Tipo=$10) And (Via[0]>#6) Then Ordena(Dir, 2, Talla)
     Else Ordena(Dir, 1, Talla);
  end;
End;
```

OBTIENE EL NOMBRE DE UN ARCHIVO QUE SE ENCUENTRA EN LA POSICION DADA

```
PROCEDURE Dir_MCGA.Obten_Arch(Dir : Pointer; Indice : Word;
                              Var DirInfo : SearchRec);
Begin
  If Dir<>Nil Then
  Begin
    Move(Mem[Seg(Dir^):Ofs(Dir^)+(Indice-1)*SizeOf(DirInfo)], DirInfo,
         SizeOf(DirInfo));
  End;
End;
```

DETERMINA TODAS LAS UNIDADES FISICAS Y LOGICAS EXISTENTES

```
PROCEDURE Dir_MCGA.Define_Unidades;
Var
  i      : Byte;
  DirAct : String;
Begin
  GetDir(0, DirAct);
  For i := 1 To 26 Do Unidades[i] := False;
  Case Num_Floppys Of
    1 : Unidades[1] := True;
    2..4 : Begin
             Unidades[1] := True;
             Unidades[2] := True;
           End;
  End;
{$I-}
  For i := 3 To 26 Do
  Begin
    ChDir(Chr(i+64)+':.');
    If IOResult=0 Then Unidades[i] := True;
  End;
  If IOResult=0 Then;
{$I-}
  ChDir(DirAct);
End;
```

```
VERIFICA LA VALIDEZ DE LA CADENA PROPORCIONADA COMO VIA DE ACCESO, MASCARA O
ARCHIVO
FUNCTION Dir_MCGA.Checa_Via(Via_Inicial : String;
                            Var DirAct, Mascara : String) : Boolean;
Var
   Arch     : Text;
   Ok       : Boolean;
   Cad, DA : String;
   i        : Word;
   DirInfo : SearchRec;
Begin
{$I-}
   Ok := True;
   GetDir(0, DA);
   ChDir(DirAct);  If IOResult=0 Then;
   Chdir(Via_Inicial);
   If IOResult=0 Then
   begin
      GetDir(0, DirAct);
      If Length(DirAct)>3 Then DirAct := DirAct+'\';
   end
   Else begin
     If (Via_Inicial<>'') And (Via_Inicial[2]<>':') Then
     Begin
        If (Via_Inicial[1]<>'\') Then Via_Inicial := DirAct+Via_Inicial
          Else Via_Inicial := Copy(DirAct,1,2)+Via_Inicial;
     End;
     Assign(Arch, Via_Inicial);
     Reset(Arch);
     If IOResult=0 Then
     Begin
        Close(Arch);    { Nombre de un solo archivo }
        Mascara := '';
        i := 0;
        While (Not (Via_Inicial[Length(Via_Inicial)-i] In ['\',':'])) And
              ((Length(Via_Inicial)-i)>0) Do
        begin
          Mascara := Via_Inicial[Length(Via_Inicial)-i]+Mascara;
          Inc(i);
        end;
        If Length(Via_Inicial)=i Then
        begin
          If Length(DA)=3 Then DirAct := DA Else DirAct := DA+'\';
        end
        Else begin
           DirAct := Copy(Via_Inicial,1,Length(Via_Inicial)-i);
           ChDir(DirAct+'.');
           If IOResult=0 Then GetDir(0,DirAct);
           If Length(DirAct)>3 Then DirAct := DirAct+'\';
        End;
     End
      Else begin
        i := Ord(UpCase(Via_Inicial[1]))-64;
        GetDir(i, Cad);
        ChDir(Via_Inicial);
        If IOResult=0 Then
        begin
           GetDir(0, DirAct);
           ChDir(Cad);   If IOResult=0 Then;    { Un nombre de subdirectorio }
           If DirAct[Length(DirAct)]<>'\' Then DirAct := DirAct+'\';
        end
        Else begin
          FindFirst(Via_Inicial, $3F, DirInfo);
          If (DosError>0) And (DosError<>18) Then { Nombre invalido }
          Begin
            If DA[0]=#3 Then DirAct := DA Else DirAct := DA+'\';
```

```
            Ok := False;
            Mensaje('Nombre de archivo no valido o error de    lectura',1);
          End
          Else
           begin          { Mascara }
             Mascara := '';
             i := 0;
             While (Not (Via_Inicial[Length(Via_Inicial)-i] In ['\',':'])))
               And ((Length(Via_Inicial)-i)>0) Do
             begin
               Mascara := Via_Inicial[Length(Via_Inicial)-i]+Mascara;
               Inc(i);
             end;
             If Length(Via_Inicial)=i Then
             begin
                If Length(DA)=3 Then DirAct := DA Else DirAct := DA+'\';
             end
             Else begin
                DirAct := Copy(Via_Inicial,1,Length(Via_Inicial)-i);
                ChDir(DirAct+'.');
                If IOResult=0 Then GetDir(0,DirAct);
                If Length(DirAct)>3 Then DirAct := DirAct+'\';
             End;
           end;
       end;
     ChDir(DA);  If IOResult=0 Then;
     If Pos(':',DirAct)=0 Then
     begin
        Cad := DirAct;
        If Cad[1]='\' Then Cad := Copy(DA,1,2)+Cad
          Else Cad := DA+'\'+Cad;
        DirAct := Cad;
     end;
     Cad := Copy(DirAct,2,2);
     If Cad<>':\' Then
     Begin
       i := Ord(UpCase(DirAct[1]))-64;
       GetDir(i, Cad);
       If Cad[0]>#3 Then Cad := Cad+'\';
       DirAct := Cad+Copy(DirAct,3,Length(DirAct));
     End;
   End;
   Checa_Via := Ok;
{$I+}
End;
```

DETERMINA EN QUE POSICION SE PULSO EL BOTON DEL RATON

```
FUNCTION Dir_MCGA.Click_Raton(Var Boton, Yr, Min, Act, Max : Word;
                                  Rango : Boolean) : Boolean;
Begin
   If Odd(Boton) And (Rango) Then
   Begin
      Yr := Yr Div 8;
      Dec(Yr,7);
      If Yr+Min<=Max Then
      begin
         Act := Min+Yr;
         Click_Raton := True;
      end
      Else Click_Raton := False;
   End
     Else Click_Raton := False;
End;
```

FUNCION PRINCIPAL DE LECTURA DE ARCHIVOS Y DIRECTORIOS

```
FUNCTION Dir_MCGA.Lee_DirArch(Var Via_Inicial : String;
                                  Salvar : Boolean) : Boolean;
Var
  DirInfo              : SearchRec;
  Unid_Act             : Array[1..26] Of Char;
  Columna,i , j        : Byte;
  Mascara,
  Cad2,
  Via_Usuario,
  DA, Cad, DirAct      : String;
  Dir2, Dir1           : Pointer;
  Ini                  : Array[1..3,1..3] Of Word;
                           { 1,x = Unidades      } { x,1 = Posición inicial   }
                           { 2,x = Directorios   } { x,2 = Posición actual    }
                           { 3,x = Archivos      } { x,3 = Total de elementos }
  Tecla, Tecla2        : Char;
  Redibuja             : Array[1..3] Of Boolean;
                           { 1 = Unidades } { 2 = Directorios } { 3 = Archivos }
  Boton, Xr, Yr        : Word;    { Variables para el uso del ratón }
  Arch                 : Text;
  P                    : Byte;
Begin
{$I-}
  P := LibreXMS(63);
  EscribeXMS(XMSMan[P], Mem[$A000:1600], 64000, 0);
  GetDir(0,DA);
  Mascara := '*.*';
  Via_Usuario := '';
  DirAct := '';
  If Checa_Via(Via_Inicial, DirAct, Mascara) Then Columna := 3
     Else Columna := 1;
  For i := 1 To 3 Do
    For j := 1 To 3 Do Ini[i, j] := 0;
  Dir1   := Nil;  Dir2 := Nil;
  Cuadrado(17*TallaLetraX-1,1*TallaLetraY-1,
    23*TallaLetraX,2*TallaLetraY-1,255);
  Cuadrado(2*TallaLetraX-1, 2*TallaLetraY,
    38*TallaLetraX, 19*TallaLetraY, 2);
  Cuadro(2*TallaLetraX-2, 2*TallaLetraY-1,
    38*TallaLetraX+1, 19*TallaLetraY+1, 0);
  Cuadro(2*TallaLetraX-3, 2*TallaLetraY-2,
    38*TallaLetraX+2, 19*TallaLetraY+2, 255);
  If Salvar Then letrero2(17,1,'SALVAR',5,0)
   Else letrero2(17,1,' LEER ',5,0);
  Cuadrado2(17*TallaLetraX-1,1*TallaLetraY-1,
    23*TallaLetraX-1,2*TallaLetraY-1,2);
  letrero2(3,5,'UNIDAD   DIRECTORIO   ARCHIVO      ', 4, 0);
  Cuadrado(9*TallaLetraX, 5*TallaLetraY,
    10*TallaLetraX-1, 17*TallaLetraY-1, 1);
  Cuadrado(23*TallaLetraX, 5*TallaLetraY,
    24*TallaLetraX-1, 17*TallaLetraY-1, 1);
  Cuadrado(9*TallaLetraX, 7*TallaLetraY,
    10*TallaLetraX-1, 8*TallaLetraY-1, 8);
  Cuadrado(23*TallaLetraX, 7*TallaLetraY,
    24*TallaLetraX-1, 8*TallaLetraY-1, 8);
  Cuadrado(9*TallaLetraX, 16*TallaLetraY,
    10*TallaLetraX-1, 17*TallaLetraY-1, 8);
  Cuadrado(23*TallaLetraX, 16*TallaLetraY,
    24*TallaLetraX-1, 17*TallaLetraY-1, 8);
  Cuadro(9*TallaLetraX, 7*TallaLetraY,
    10*TallaLetraX-1, 8*TallaLetraY-1, 7);
  Cuadro(23*TallaLetraX, 7*TallaLetraY,
    24*TallaLetraX-1, 8*TallaLetraY-1, 7);
  Cuadro(9*TallaLetraX, 16*TallaLetraY,
    10*TallaLetraX-1, 17*TallaLetraY-1, 7);
  Cuadro(23*TallaLetraX, 16*TallaLetraY,
    24*TallaLetraX-1, 17*TallaLetraY-1, 7);
```

```
Cuadrado(5*TallaLetraX, 7*TallaLetraY,
  7*TallaLetraX-1, 17*TallaLetraY-1,0);
Cuadrado(10*TallaLetraX, 7*TallaLetraY,
  23*TallaLetraX-1, 17*TallaLetraY-1,0);
Cuadrado(24*TallaLetraX, 7*TallaLetraY,
  37*TallaLetraX-1, 17*TallaLetraY-1,0);
For i := 1 To 3 Do Redibuja[i] := True;
j := 1;
For i := 1 to 26 Do
  If Unidades[i] Then
  begin
     Inc(Ini[1,3]);
     If Chr(i+64)=DirAct[1] Then Ini[1,2] := Ini[1,3];
     Unid_Act[j] := Chr(i+64);
     Inc(j);
  end;
If Columna=3 Then
begin
   Cad := Mascara;
   For i := 1 To Length(Cad) Do Cad[i] := UpCase(Cad[i]);
   i := Pos('.',Mascara)+1;
   If i>1 Then Mascara := '*.'+Copy(Mascara,i,3);
end;
LeeDirectorio(Dir1, Ini[2,3], DirAct+'*.*', $10);
LeeDirectorio(Dir2, Ini[3,3], DirAct+Mascara, $21);
For i := 1 To 3 Do
  If Ini[i,3]>0 Then
  begin
     Ini[i,1] := 1;    Ini[i,2] := 1;
  end;
If Columna=3 Then
begin
   i := 1;
   While i<=Ini[3,3] Do
   begin
      Obten_Arch(Dir2, i, DirInfo);
      If DirInfo.Name=Cad Then
      begin
         Ini[3,2] := i;
         If i<10 Then Ini[3,1] := 1 Else Ini[3,1] := i-9;
         i := Ini[3,3];
      End;
      Inc(i);
   end;
End;
Tecla := #1;
Repeat
   If Tecla In [#1,#8,#33..#165] Then
     If Length(Via_Usuario)>34 Then
        letrero2(2,18,'<'+Copy(Via_usuario,
          Length(Via_usuario)-33,34)+'_',6,0)
     Else
        letrero2(2,18,' '+Espacios(Via_usuario+'_',35,2),6,0);
   If (Redibuja[1]) Or (Redibuja[3]) Then
   Begin
      Cad := DirAct+Mascara;
      letrero2(2,3,Espacios(Cad,36,2),10,0);
   End;
   If Redibuja[1] Then
    For i := 0 to 9 Do
      If (Ini[1,1]+i)<=Ini[1,3] Then
      Begin
        If (i=Ini[1,2]-Ini[1,1]) Then
        begin
           letrero2(6, 7+i, Unid_Act[Ini[1,1]+i], 2,0);
           If Columna=1 Then letrero2(5,7+i,'>', 2,0)
             Else letrero2(5,7+i,' ', 2, 0);
```

```
          end
          Else begin
             letrero2(6, 7+i, Unid_Act[Ini[1,1]+i], 7, 0);
             letrero2(5,7+i,' ', 2, 0);
          end;
        End
          Else letrero2(5, 7+i, ' ', 0, 0);
      If Redibuja[2] Then
        For i:=0 To 9 Do
        Begin
          If (Ini[2,3]>0) And (Ini[2,1]+i<=Ini[2,3]) Then
          begin
             Obten_Arch(Dir1, Ini[2,1]+i, DirInfo);
             Cad := DirInfo.Name;
          end
          Else Cad := '';
          If (i=Ini[2,2]-Ini[2,1]) Then
          Begin
            If Columna=2 Then letrero2(10,7+i,'>', 2,0)
              Else letrero2(10,7+i,' ', 2, 0);
            letrero2(11, 7+i, Espacios(Cad,12,2), 2,0);
          End
          Else begin
            letrero2(10,7+i,' ', 2, 0);
            letrero2(11, 7+i, Espacios(Cad,12,2), 7,0);
          End;
        End;
      If Redibuja[3] Then
        For i := 0 To 9 Do
        Begin
          If (Ini[3,3]>0) And ((Ini[3,1]+i)<=Ini[3,3]) Then
          begin
             Obten_Arch(Dir2, Ini[3,1]+i, DirInfo);
             Cad := DirInfo.Name;
          end
          Else Cad := '';
          If (i=Ini[3,2]-Ini[3,1]) Then
          Begin
            If Columna=3 Then letrero2(24,7+i,'>', 2, 0)
              Else letrero2(24,7+i,' ', 2, 0);
            letrero2(25, 7+i, Espacios(Cad,12,2), 2,0);
          End
          Else begin
            letrero2(24,7+i,' ', 2,0);
            letrero2(25, 7+i, Espacios(Cad,12,2), 7,0);
          End;
        End;
      For i := 1 To 3 Do Redibuja[i] := False;
      Repeat
        Tecla := #1;   Tecla2 := #0;
        CursorRaton(True);  Boton := 0;
        Repeat
          PosRaton(Boton, Xr, Yr);
{ Unids }  If Click_Raton(Boton, Yr, Ini[1,1], Ini[1,2], Ini[1,3],
             RangoRaton(5*TallaLetraX, 7*TallaLetraY,
             7*TallaLetraX, 17*TallaLetraY)) Then


           Begin
             Tecla := #32;  Columna := 1;  Boton := 0;
           End;
{ dirs  }  If Click_Raton(Boton, Yr, Ini[2,1], Ini[2,2], Ini[2,3],
             RangoRaton(11*TallaLetraX, 7*TallaLetraY,
             23*TallaLetraX, 17*TallaLetraY)) Then
           Begin
             Tecla := #32;  Columna := 2;  Boton := 0;
           End;
```

```
{ Archs }    If Click_Raton(Boton, Yr, Ini[3,1], Ini[3,2], Ini[3,3],
               RangoRaton(25*TallaLetraX, 7*TallaLetraY,
               37*TallaLetraX, 17*TallaLetraY)) Then
             Begin
               Tecla := #32;  Columna := 3;  Boton := 0;
             End;
{ Salir }    If (Boton And 2)>0 Then Tecla := #27;
{ Elige }    If (Boton And 4)>0 Then Tecla := #32;
{ Dir.Ar }   If Odd(Boton) And (RangoRaton(9*TallaLetraX,7*TallaLetraY,
               10*TallaLetraX,8*TallaLetraY)) Then
             Begin
               Tecla := #0;  Tecla2 := #73;  Columna := 2;  Boton := 0;
             End;
{ Dir.Ab }   If Odd(Boton) And (RangoRaton(9*TallaLetraX,16*TallaLetraY,
               10*TallaLetraX,17*TallaLetraY)) Then
             Begin
               Tecla := #0;  Tecla2 := #81;  Columna := 2;  Boton := 0;
             End;
{ Arch.Ar }  If Odd(Boton) And (RangoRaton(23*TallaLetraX,7*TallaLetraY,
               24*TallaLetraX,8*TallaLetraY)) Then
             Begin
               Tecla := #0;  Tecla2 := #73;  Columna := 3;  Boton := 0;
             End;
{ Arch.Ab }  If Odd(Boton) And (RangoRaton(23*TallaLetraX,16*TallaLetraY,
               24*TallaLetraX,17*TallaLetraY)) Then
             Begin
               Tecla := #0;  Tecla2 := #81;  Columna := 3;  Boton := 0;
             End;
           Until (Keypressed) Or (Tecla<>#1);
           CursorRaton(False);
           If Tecla=#32 Then
            Case Columna Of
              1 : Cuadrado2(6*TallaLetraX, (7+Ini[1,2]-Ini[1,1])*TallaLetraY,
                    7*TallaLetraX-1, (8+Ini[1,2]-Ini[1,1])*TallaLetraY-1, 13);
              2 : Cuadrado2(11*TallaLetraX, (7+Ini[2,2]-Ini[2,1])*TallaLetraY,
                    23*TallaLetraX-1, (8+Ini[2,2]-Ini[2,1])*TallaLetraY-1, 13);
              3 : Cuadrado2(25*TallaLetraX, (7+Ini[3,2]-Ini[3,1])*TallaLetraY,
                    37*TallaLetraX-1, (8+Ini[3,2]-Ini[3,1])*TallaLetraY-1, 13);
            End;
           Libera_Boton;
           If KeyPressed Then Tecla  := Readkey;
           If Keypressed Then Tecla2 := Readkey;
           Case Tecla Of
              #0 : Case Tecla2 Of
                     #72 : Begin
                             If Ini[Columna,2]>1 Then
                             begin
                               Dec(Ini[Columna,2]);
                               If Ini[Columna,1]>Ini[Columna,2]
                                 Then Ini[Columna,1] := Ini[Columna,2];
                             End Else Tecla2 := #0;
                           End;

                     #80 : Begin
                             If Ini[Columna,2]<Ini[Columna,3] Then
                             Begin
                               Inc(Ini[Columna,2]);
                               If Ini[Columna,1]+9<Ini[Columna,2] Then
                                 Ini[Columna,1] := Ini[Columna,2]-9;
                             End
                               Else Tecla2 := #0;
                           End;
                     #75 : If Columna>1 Then Dec(Columna) Else Columna := 3;
                     #77 : If Columna<3 Then Inc(Columna) Else Columna := 1;
                     #71 : Begin
                             Ini[Columna,1] := 1;  Ini[Columna,2] := 1;
                           End;
```

```
        #79 : Begin
                Ini[Columna,2] := Ini[Columna,3];
                Ini[Columna,1] := Ini[Columna,2];
                i := 0;
                While (Ini[Columna,1]-i>1) And (i<9) Do Inc(i);
                Ini[Columna,1] := Ini[Columna,1]-i;
              End;
        #81 : Begin
                If Ini[Columna,2]+9<Ini[Columna,3] Then
                   Inc(Ini[Columna,2],10)
                Else Ini[Columna,2] := Ini[Columna,3];
                Ini[Columna,1] := Ini[Columna,2];
                i := 0;
                While (Ini[Columna,1]-i>1) And (i<9) Do Inc(i);
                Ini[Columna,1] := Ini[Columna,1]-i;
              End;
        #73 : Begin
                If Ini[Columna,2]>9 Then Dec(Ini[Columna,2],9)
                Else Ini[Columna,2] := 1;
                If Ini[Columna,1]>Ini[Columna,2] Then
                  Ini[Columna,1] := Ini[Columna,2];
              End;
      End;
#32 : Case Columna Of
        1 : Begin
              ChDir(Unid_Act[Ini[1,2]]+':.');
              If IOResult<>0 Then
              Begin
                 Mensaje('Unidad no lista: '+
                         Unid_Act[Ini[1,2]]+':',1);
                 Tecla := #0;
                 Redibuja[1] := True;
              end Else
              begin
                GetDir(Ord(Unid_Act[Ini[1,2]])-64, DirAct);
                ChDir(DA);
                If DirAct[0]>#3 Then DirAct := DirAct+'\';
                LeeDirectorio(Dir1,Ini[2,3],DirAct+'*.*',$10);
                LeeDirectorio(Dir2,Ini[3,3],DirAct+Mascara,$21);
                For i := 2 To 3 Do
                If Ini[i,3]>0 Then
                begin
                  Ini[i,1] := 1;   Ini[i,2] := 1;
                end;
              end;
            End;


        2 : If (Ini[2,3]>0) And (Ini[2,2]>0) Then
            Begin
              Obten_Arch(Dir1, Ini[2,2], DirInfo);
              GetDir(Ord(DirAct[1])-64, Cad);
              If DirInfo.Name='\' Then ChDir(Copy(DirAct,1,3))
              Else
                 ChDir(DirAct+DirInfo.Name);
              If IOResult<>0 Then
              Begin
                 Mensaje('Error leyendo: '+Cad,1);
                 Tecla := #0;
                 Redibuja[2] := True;
              end Else
              begin
                GetDir(Ord(DirAct[1])-64, DirAct);
                ChDir(Cad);
                ChDir(DA);
                If IOResult<>0 Then;
                If DirAct[0]>#3 Then DirAct := DirAct+'\';
```

```
                        LeeDirectorio(Dir1,Ini[2,3],DirAct+'*.*',$10);
                        LeeDirectorio(Dir2,Ini[3,3],DirAct+Mascara,$21);
                        For i := 1 To 3 Do
                        If Ini[i,3]>0 Then
                        begin
                          Ini[i,1] := 1;    Ini[i,2] := 1;
                        end;
                      end;
                    End;
              3 : If (Ini[3,3]>0) And (Ini[3,2]>0) Then
                    Begin
                      Obten_Arch(Dir2, Ini[3,2], DirInfo);
                      Cad := DirAct+DirInfo.Name;
{$I-}
                      Assign(Arch, Cad);
                      Reset(Arch);
                      If IOResult=0 Then
                      begin
                        Close(Arch); If IOResult<>0 Then;
                        If (Not Salvar)  Or
                        (Mensaje('Ok, sobreescribir Archivo : '+Cad,2)=1)
                        Then
                        Begin
                          Tecla := #2;
                          Cad := Mascara;
                          Mascara := DirInfo.Name;
                          Redibuja[3] := True;
                        End Else
                        Begin
                          Tecla := #0;   Redibuja[3] := True;
                          If Not Salvar Then
                            Mensaje('Error leyendo: '+Cad,1);
                        End;
                      End Else Mensaje('Error leyendo: '+Cad,1);
                    End;
          End;
    #33..#165 : If Length(Via_Usuario)<250 Then
                  Via_Usuario := Via_Usuario+UpCase(Tecla);
    #8        : If Via_Usuario<>'' Then
          Via_Usuario := Copy(Via_Usuario,1,length(Via_usuario)-1);
    #13       : If Via_Usuario<>'' Then
                Begin
                  If Salvar Then
                  Begin
                    GetDir(0, Cad2);
                    ChDir(DirAct+'.');
                    If IOResult=0 Then;
                    Assign(Arch, Via_Usuario);
                    Reset(Arch);
                    If IOResult=0 Then
                    Begin
                      Close(Arch);  If IOResult=0 Then;
                      If Mensaje('Ok, sobreescribir Archivo : '+
                        Cad,2)=1 Then
                      begin
                        Tecla := #2;
                        Cad := Mascara;
                        Mascara := '';
                        i := Length(Via_Usuario);
                        While (i>0) And
                              (Not (Via_Usuario[i] In ['\',':'])) Do
                        Begin
                          Mascara := Via_Usuario[i]+Mascara;
                          Dec(i);
                        End;
                        DirAct  := Copy(Via_Usuario,1,i);
                        If (DirAct<>'')
```

```
            And (DirAct[Length(DirAct)]<>'\')
            Then DirAct := DirAct+'\';
          ChDir(DirAct+'.');
          If IOResult<>0 Then;
          GetDir(0,DirAct);
          If (DirAct<>'')
            And (DirAct[Length(DirAct)]<>'\')
            Then DirAct := DirAct+'\';
        end Else Tecla := #1;
      End;
      If Tecla=#13 Then
      Begin
        ChDir(DirAct+'.');
        If IOResult<>0 Then;
        ReWrite(Arch);
        If IOResult=0 Then
        Begin
          Close(arch);
          If Mensaje('Ok Archivo :'+Via_Usuario,2)=1
          Then
          Begin
            i := Length(Via_Usuario);
            Cad     := Mascara;
            Mascara  := '';
            While (i>0) And
              (Not (Via_Usuario[i] In ['\',':'])) Do
            Begin
              Mascara := Via_Usuario[i]+Mascara;
              Dec(i);
            End;
            DirAct  := Copy(Via_Usuario,1,i);
            If (DirAct<>'')
              And (DirAct[Length(DirAct)]<>'\')
               Then DirAct := DirAct+'\';
            ChDir(DirAct+'.');
            If IOResult<>0 Then;
            GetDir(0,DirAct);
            If (DirAct<>'')
              And (DirAct[Length(DirAct)]<>'\')
               Then DirAct := DirAct+'\';
            Tecla := #2;
          End Else begin
               Erase(Arch);
               Tecla := #1;
             end;
        End;
      End;
      ChDir(Cad2);  If IOResult<>0 Then;
    End;
  If Tecla=#13 Then
  Begin
   Cad := DirAct;   Cad2 := Mascara;
   If Checa_Via(Via_Usuario, Cad, Cad2) Then
   Begin
     DirAct  := Cad;
     Cad     := Mascara;
     Mascara := Cad2;
     LeeDirectorio(Dir1, Ini[2,3], DirAct+'*.*', $10);
     LeeDirectorio(Dir2, Ini[3,3], DirAct+Mascara, $21);
     For i := 1 To 3 Do
     If Ini[i,3]>0 Then
     begin
       Ini[i,1] := 1;   Ini[i,2] := 1;
     end;
     Via_Usuario := '';
     Tecla := #1;
     If (Ini[3,3]>0) And (Pos('*',Mascara)=0)
```

```
                                    And (Pos('*',Mascara)=0)
                                  Then begin
                                     Assign(Arch, DirAct+Mascara);
                                     Reset(Arch);
                                     If IOResult=0 Then
                                     Begin
                                        Close(Arch);  If IOResult<>0 Then;
                                        Tecla := #2;
                                        Redibuja[3] := True;
                                     End Else
                                     Begin
                                        Tecla := #1;   Redibuja[3] := True;
                                        Write(#7);
                                     End;
{$I+}                              End;
                                End;
                               End;
                              End;
        End;
        If Tecla2 In [#71..#73,#79..#81] Then Redibuja[Columna] := True;
        If (Tecla2 In [#75,#77]) Or (Tecla in [#32,#1]) Then
           For i := 1 To 3 Do Redibuja[i] := True;
     Until (Redibuja[1]) Or (Redibuja[2]) Or (Redibuja[3])
       Or (Tecla In [#1,#2,#8,#27,#33..#165]);
     If Tecla=#2 Then
     Begin
        If (Not Salvar)
          And (Mensaje('Ok, Archivo : '+DirAct+Mascara, 2)<>1) Then
        Begin
           Tecla := #1;  Mascara := Cad;
        End;
     End;
  Until Tecla In [#27, #2];
  ChDir(DA);
  If IOResult<>0 Then;
{$I+}
  FreeMem(Dir1, Ini[2,3]*SizeOf(DirInfo));
  FreeMem(Dir2, Ini[3,3]*SizeOf(DirInfo));
  LeeXMS(XMSMan[P], Mem[$A000:1600], 64000, 0);
  LiberaXMS(XMSMan[P]);
  XMSMan[P] := 0;
  If Tecla<>#27 Then
  Begin
     Via_Inicial := DirAct+Mascara;
     Lee_DirArch := True;
  end
  Else Lee_DirArch := False;
End;

{ Incluir en la rutina principal  Dir_MCGA.Define_Unidades; }
end.
```

LibDnd.pas : Libreria para el manejo duendes.

```
Unit libdnd;
INTERFACE
Uses dos, crt, librat, libuti;
Const
  tam   = 64000;          { Tamaño de la pantalla de video  - MCGA 320x200      }

TYPE

Duende = record
  Dnd   : array[1..80] of pointer;
  Tam   : array[1..80] of word;
  Total : byte;
end;

Duendes = object(raton)
  Function  Sizeimagen(xs,ys,xi,yi: word) : word;
  Procedure Getimagen(xs,ys,xi,yi: integer; var imagen : pointer);
  Procedure Putimagen(xs,ys : integer; imagen : pointer);
  Procedure PutDuende(X1,Y1 : integer; imagen : duende; var pant; num : byte);
  Procedure LeerDuende(Var mono : duende; nombre : string);
  Procedure IniciaDuende(Var mono : duende; corx,cory : word; numero : byte);
  Procedure LiberaDuende(Var mono : duende);
  Procedure SalvarDnd(mono : duende; nombre : string);
  Function  Checa_boton(x,y:word; boton:duende; numero:byte;
            car:string) : boolean;
  Procedure Espejo_Hor(Var imagen : Pointer);
  Procedure Espejo_Ver(Var imagen : Pointer);
  Procedure Rotacion(Var imagen : Pointer);
  Procedure RellenoDnd(D1 : Duende; Numero : Byte; Var Destino);
  Procedure CopiaDuende(Var D1 : Duende; Ini, Fin, Dest : Byte);
end;
Var
   venx1, venx2, veny1, veny2 : integer;



IMPLEMENTATION

LEE UN BLOQUE DE VIDEO Y LO METE EN UN APUNTADOR
Procedure Duendes.Getimagen(xs,ys,xi,yi: integer; var imagen : pointer);
Var
  x,y : integer;
  cont : word;
  mxs,mys : integer;
begin
  mxs:=xs; mys:=ys;
  if xs>xi then begin xs:=xi; xi:=mxs; end;
  if ys>yi then begin ys:=yi; yi:=mys; end;
  x:=xi-xs+1; y:=yi-ys;
  mem[seg(imagen^):ofs(imagen^)+0]:=lo(x);
  mem[seg(imagen^):ofs(imagen^)+1]:=hi(x);
  mem[seg(imagen^):ofs(imagen^)+2]:=lo(y);
  mem[seg(imagen^):ofs(imagen^)+3]:=hi(y);
  cont:=0;
  for y := ys to yi do
    begin
    move(mem[$A000:xs+y*320],mem[seg(imagen^):ofs(imagen^)+cont+4],x);
     inc(cont,x);
    end;
```

```
end;
```

LEE UN BLOQUE DE UN APUNTADOR Y LO METE EN VIDEO

```
Procedure Duendes.Putimagen(xs,ys : integer; imagen : pointer);
Var
  x,y : integer;
  xi, yi : integer;
  cont : integer;
  rangox : integer;
  rangoxs : integer;

begin
  x:=mem[seg(imagen^):ofs(imagen^)+0]+mem[seg(imagen^):ofs(imagen^)+1]*256;
  y:=mem[seg(imagen^):ofs(imagen^)+2]+mem[seg(imagen^):ofs(imagen^)+3]*256;
  xi:=xs+x;
  yi:=ys+y;
  cont :=0;
  rangox:=x;
  rangoxs:=0;
  for y:= ys to yi do
    begin
      if (x+xs)>320 then dec(rangox,rangox+xs-320);
      if  xs< 0 then
        begin
          inc(rangoxs,abs(xs));
          dec(rangox,abs(xs));
        end;
      if (rangox+xs) < 321 then
        begin
          move(mem[seg(imagen^):ofs(imagen^)+cont+rangoxs+4],
                mem[$A000:xs+y*320],rangox);
        end;
      inc(cont,x);
    end;
end;
```

PONE UN BLOQUE QUITANDO EL BYTE DE CONDICION (254)

```
Procedure Duendes.PutDuende(X1,Y1:integer; imagen:duende; var pant; num:byte);
Var
   i, largo, ancho,
   L1 , A1,
   A2, L2,
   Ofs1A, Ofs2a,
   Seg1, Ofs1,
   Seg2, Ofs2,
   IniX, IniY,
   FinX, FinY,
   Talla : Word;
   p, p1 : Pointer;
   apunta : pointer;
   label c1, c2, c3;
Begin
   Seg2 := Seg(imagen.Dnd[num]^);
   Ofs2 := Ofs(imagen.Dnd[num]^);
   apunta := Addr(mem[Seg(pant):ofs(pant)]);
   seg1 :=seg(apunta^);
   ofs1 :=ofs(apunta^);
   Move(Mem[Seg2:Ofs2], Ancho, 2);
   Move(Mem[Seg2:Ofs2+2], Largo, 2);
   Inc(Ofs2,4);
   A2 := 0;
   L2 := 0;
   IniX := X1;
```

```
  FinX := X1+Ancho;
  IniY := Y1;
  FinY := Y1+Largo;
  If X1<VenX1 Then
     If X1+Ancho<VenX1 Then
        exit
     Else begin
        IniX := VenX1;
        A2   := VenX1 - X1;
     end;
  If X1>VenX2 Then
     Exit
  Else
     If X1+Ancho>VenX2 Then
        FinX := VenX2;
  A1 := FinX-IniX;

  If Y1<VenY1 Then
     If Y1+Largo<VenY1 Then
        exit
     Else begin
        IniY := VenY1;
        L2   := VenY1 - Y1;
     end;
  If Y1>VenY2 Then
     Exit
  Else
     If Y1+Largo>VenY2 Then
        FinY := VenY2;
  L1 := FinY-IniY;

  If ((L2+L1)>0) And (A1>0) Then
  For i:= L2 To L2+L1 Do
  begin
    Ofs2A := Ofs2+i*Ancho+A2;
    Ofs1A := Ofs1+IniX+(IniY+i-L2)*320;
    if (ofs2a>64000) or (ofs1A>64000) then
       ancho := ancho;
 asm
    push ds
    mov ax, Seg2
    mov ds, ax
    mov ax, Ofs2A
    mov si, ax
    mov ax, ofs1A
    mov di, ax
    mov ax, seg1
    mov es, ax
    mov cx, A1
c1: mov ah, ds:[si]
    cmp ah, 254
    je c2
    mov es:[di], ah
c2: inc di
    inc si
c3: loop c1
    pop ds
  end;
   end;
end;
```

DEVUELVE EL TAMAÑO DE UN BLOQUE LIMITADO POR UN RECTANGULO

```
Function Duendes.Sizeimagen(xs,ys,xi,yi: word) : word;
Var
  cont : word;
  x,y : integer;
Begin
```

```
  cont:=0;
  cont:=(xi-xs+1)*(yi-ys+1)+4;
  sizeimagen:=cont;
End;
```

PROCEDIMIENTO QUE LEE BOTONES EN FORMATO DE DUENDE

```
Procedure Duendes.leerDuende(Var mono : duende; nombre : string);
Var
  apuntador : word;
  bytel     : byte;
  tamano    : word;
  Bloque    : pointer;
  alto,bajo : byte;
  video     : byte;
  archivo   : file;
  buffer    : array[1..2] of byte;

Begin
  If Mono.Total>0 Then Liberaduende(mono);
  Assign(archivo, nombre);
  reset(archivo,1);
  mono.total:=0;
  While not eof (archivo) do
    Begin
      Inc(mono.total);
      Blockread(archivo,buffer,2);
      bajo:=buffer[1]; alto:=buffer[2];
      tamano:=alto*256+bajo;
      getmem(mono.dnd[mono.total],tamano);
      mono.tam[mono.total]:=tamano;
      Blockread(archivo,mono.dnd[mono.total]^,tamano);
    End;
  close(archivo);
End;
```

PROCEDIMIENTO QUE INICIALIZA LA POSICION DE LOS BOTONES

```
Procedure Duendes.IniciaDuende(Var mono:duende; corx,cory:word; numero:byte);
Begin
  Putimagen(corx,cory,mono.dnd[numero]);
End;
```

PROCEDIMIENTO QUE LIBERA LOS BOTONES DE LA MEMORIA

```
Procedure Duendes.LiberaDuende(Var mono : duende);
Var
 Cont : word;
Begin
  if mono.total > 0 then
  Begin
    For cont:=1 to mono.total do
    begin
     if mono.tam[cont]>0 then freemem(mono.dnd[cont],mono.tam[cont]);
    end;
    Mono.Total := 0;
  End;
End;
```

PROCEDIMIENTO QUE SALVA UN DUENDE EN DISCO

```
Procedure Duendes.SalvarDnd(mono : duende; nombre : string);
Var
  ArchiDnd  : file;
  alto,bajo : byte;
  vide      : byte;
  cont      : byte;
Begin
  assign(archidnd,nombre);
  rewrite(archidnd,1);
```

```
  for cont := 1 to mono.total do
    begin
      alto:=hi(mono.tam[cont]);
      bajo:=lo(mono.tam[cont]);
      vide:=$13;
{       blockwrite(archidnd,vide,1);}
      blockwrite(archidnd,bajo,1);
      blockwrite(archidnd,alto,1);
      blockwrite(archidnd,mono.dnd[cont]^,mono.tam[cont]);
    end;
  close(archidnd);
End;
```

FUNCION QUE INDICA SI SE APRETO EL BOTON

```
Function Duendes.Checa_boton(x,y:word; boton:duende; numero:byte;
         car:string) : boolean;
Var
  RaX, RaY, RaB : word;
  largo, ancho : word;
  ok : boolean;
  teclado : Char;
  tecla : char;
Begin
   checa_boton:=false; ok:=false;
   move(mem[seg(boton.dnd[numero]^):ofs(boton.dnd[numero]^)],largo,2);
   move(mem[seg(boton.dnd[numero]^):ofs(boton.dnd[numero]^)+2],ancho,2);
   PosRaton(RaB,RaX,RaY);
   if rab=1 then if RangoRaton(x-1,y,largo+x-1,ancho+y) then ok:=true;
   teclado:=UpCase(Chr(mem[$40:mem[$40:28]-2]));
   if (keypressed) and (teclado=#0) and (car[1]=#0) then
     begin
       teclado:=Upcase(Chr(mem[$40:mem[$40:28]-1]));
       car[1]:=car[2];
     end;
   if (keypressed) and (teclado=UpCase(car[1])) then
     begin ok:=true; bufferteclado; end;
   if ok then
     begin
       cursorraton(false);
       putimagen(x,y,boton.dnd[numero+1]);
       cursorraton(true);
       delay(100);
       while numbotonraton=1 do;
       cursorraton(false);
       putimagen(x,y,boton.dnd[numero]);
       cursorraton(true);
       checa_boton:=true;
     end;
End;
```

PROCEDIMIENTO QUE INVIERTE HORIZONTALMENTE UN DUENDE

```
PROCEDURE Duendes.Espejo_Hor(Var imagen : Pointer);
Var
   Seg1, Ofs1,
   Ancho, Largo,
   x, y            : Word;
   Dato            : Byte;
Begin
   Seg1 := Seg(Imagen^);
   Ofs1 := Ofs(Imagen^);
   Move(Mem[Seg1:Ofs1], Ancho, 2);
   Move(Mem[Seg1:Ofs1+2], Largo, 2);
   Inc(Ofs1, 4);
   For x := 0 To (Ancho-1) Div 2 Do
      For y := 0 To Largo Do
      Begin
```

```
            Dato := Mem[Seg1:Ofs1+y*Ancho+Ancho-x-1];
            Mem[Seg1:Ofs1+y*Ancho+Ancho-x-1] := Mem[Seg1:Ofs1+y*Ancho+x];
            Mem[Seg1:Ofs1+y*Ancho+x] := Dato;
        End;
End;
```

PROCEDIMIENTO QUE INVIERTE VERTICALMENTE UN DUENDE

```
PROCEDURE Duendes.Espejo_Ver(Var imagen : Pointer);
Var
   Seg1, Ofs1,
   Ancho, Largo,
   x, y          : Word;
   Dato          : Byte;
Begin
   Seg1 := Seg(Imagen^);
   Ofs1 := Ofs(Imagen^);
   Move(Mem[Seg1:Ofs1], Ancho, 2);
   Move(Mem[Seg1:Ofs1+2], Largo, 2);
   Inc(Ofs1, 4);
   For y := 0 To Largo Div 2 Do
      For x := 0 To Ancho-1 Do
      Begin
         Dato := Mem[Seg1:Ofs1+(Largo-y)*Ancho+x];
         Mem[Seg1:Ofs1+(Largo-y)*Ancho+x] := Mem[Seg1:Ofs1+y*Ancho+x];
         Mem[Seg1:Ofs1+y*Ancho+x] := Dato;
      End;
End;
```

ROTA UNA IMAGEN

```
PROCEDURE Duendes.Rotacion(Var imagen : Pointer);
Var
   Seg1, Ofs1,
   Seg2, Ofs2,
   Ancho, Largo,
   x, y          : Word;
   P             : Pointer;
Begin
   Seg1 := Seg(Imagen^);
   Ofs1 := Ofs(Imagen^);
   Move(Mem[Seg1:Ofs1], Ancho, 2);
   Move(Mem[Seg1:Ofs1+2], Largo, 2);
   Inc(Ofs1, 4);
   inc(Largo);
   GetMem(P, Ancho*Largo+4);
   Seg2 := Seg(P^);
   Ofs2 := Ofs(P^);
   Inc(Ofs2, 4);
   For x := 0 To Ancho-1 Do
      For y := 0 To Largo-1 Do
         Mem[Seg2:Ofs2+x*Largo+(Largo-y-1)] := Mem[Seg1:Ofs1+y*Ancho+x];
   Dec(Ofs2, 4);
   dec(ancho);
   Move(Largo, Mem[Seg2:Ofs2], 2);
   Move(Ancho, Mem[Seg2:Ofs2+2], 2);
   FreeMem(Imagen, (Ancho+1)*Largo+4);
   Imagen := P;
End;
```

RELLENA LA PANTALLA CON LA IMAGEN DE UN DUENDE

```
PROCEDURE Duendes.RellenoDnd(D1 : Duende; Numero : Byte; Var Destino);
Var
   ancho, largo,
   i, j : Word;
Begin
   i := 0;  j := 0;
   ancho := MemW[seg(d1.dnd[1]^):ofs(d1.dnd[1]^)];
```

```
    largo := MemW[seg(d1.dnd[1]^):ofs(d1.dnd[1]^)+2];
    Repeat
       i := 0;
       repeat
          putduende(i,j,d1,Destino,Numero);
          inc(i,ancho);
       until (i>venx2);
       Inc(j,largo);
    until j>veny2;
end;
```

COPIA UN RANGO DE IMAGENES DENTRO DE UN DUENDE

```
PROCEDURE Duendes.CopiaDuende(Var D1 : Duende; Ini, Fin, Dest : Byte);
Var
   i, incr : byte;
Begin
   Incr := Fin-Ini+1;
   If Dest>D1.Total Then Dest := D1.Total+1;
   For i := D1.Total DownTo Dest Do
   begin
     D1.dnd[i+incr] := D1.dnd[i];
     D1.tam[i+incr] := D1.tam[i];
   end;
   For i := 0 To incr-1 Do
     If Ini+i<Dest then
     Begin
        GetMem(D1.Dnd[Dest+i], D1.Tam[Ini+i]);
        Move(D1.dnd[Ini+i]^, D1.dnd[Dest+i]^, D1.Tam[Ini+i]);
        D1.Tam[Dest+i] := D1.Tam[Ini+i];
     End
     Else
     Begin
        GetMem(D1.Dnd[Dest+i], D1.Tam[Ini+i+incr]);
        Move(D1.dnd[Ini+i+incr]^, D1.dnd[Dest+i]^, D1.Tam[Ini+i+incr]);
        D1.Tam[Dest+i] := D1.Tam[Ini+i+incr];
     End;
  Inc(D1.total, incr);
End;

BEGIN
   VenX1 := 0;    VenY1 :=0;    VenX2 := 319;    VenY2 := 199;
END.
```

LibGra.pas : Libreria para el manejo del entorno gráfico.

```
Unit LibGra;
INTERFACE
Uses dos, crt, LibRat,LibUti,libxms;
Const
  tam   = 64000;          { Tamaño de la pantalla de video  - MCGA 320x200  }

VAR
   DestinoLet,
   Pantemporal : pointer;
   TallaLetraX,
   TallaLetraY  : Byte;

Procedure Despliega(pantalla : pointer);
Procedure Despliega2(pantalla : pointer);
Procedure ModoGrafico(modo : byte);
Procedure Letrero(corx,cory : byte; cadena : string; color : byte);
Procedure Letrero2(corx,cory : byte; cadena : string; color, Tipo : byte);
Procedure VideoPantalla(Var pantalla : pointer);
Function  NumeroACadena(numero : longint) : string;
Procedure PonPixel(corx, cory : word; color : byte);
Procedure Cuadrado(xs,ys,xi,yi : integer; color : byte);
Procedure Cuadrado2(xs,ys,xi,yi : integer; color : byte);
Procedure Cuadro(xs,ys,xi,yi : integer; color : byte );
Procedure Rectangulo(Var boton:word; Var xs,ys,xi,yi:integer);
Procedure LeeCadena(x,y : byte; tam : byte ; tipo : byte;
            Var tecla1,tecla2 : char; Var ante : string; color : byte);
Procedure Leeinteger(x,y : byte; tam : byte; Var tecla1, tecla2 : char;
            Var ante : integer; color : byte);
Procedure Leeword(x,y : byte; tam : byte; Var tecla1, tecla2 : char;
            Var ante : word; color : byte);
Procedure Leebyte(x,y : byte; tam : byte; Var tecla1, tecla2 : char;
            Var ante : byte; color:byte);
Procedure PonPaleta(Var p);
Function  Mensaje(D : String; Tipo : Byte) : Byte;
Procedure LeeEntero(Var Numero : Word; X, Y : Byte; Enteros : Byte;
            Var Tecla1, Tecla2 : Char; Var Boton, Xr, Yr :Word;
            Color : Byte; Var Primero : Boolean);
PROCEDURE LeeDato(Var Dato; Tipo, X, Y, Talla : Byte;
            Var Tecla1, Tecla2 : Char; Var Boton, Xr, Yr:Word; Color : Byte);
Function  Escala(X : Word) : Byte;
Procedure Leer_FontUsuario(Nombre : String; Ancho, Alto : Byte);

IMPLEMENTATION

Var
   LetraUsuario : Boolean;
   TamLetra     : Word;
   AlmacenLetra : Pointer;

Function Escala(X : Word) : Byte;
Begin
end;
```

PROCEDIMIENTO QUE DESPLIEGA UNA IMAGEN EN VIDEO

```
Procedure Despliega2(pantalla : pointer);
Begin
```

```
  move(pantalla^, mem[$A000:$0000], tam);
End;

CAPTURA LA IMAGEN ACTUAL DEL MONITOR EN LA VARIABLE PANTALLA
Procedure VideoPantalla(Var pantalla : pointer);
Begin
  move(mem[$A000:0],pantalla^,tam);
End;

ACTIVA Y DESACTIVA EL MODO GRAFICO
Procedure ModoGrafico(modo : byte);
Var
  reg : registers;
Begin
  with reg do
    begin
      ah:=$00;
      al:=modo;
      intr($10,reg);
    end;
End;

PROCEDIMIENTO QUE PONE UN LETRERO EN LA POSICION Y CON EL COLOR INDICADOS.
(PERMITE ESPECIFICAR MODO TRANSPARENTE U OPACO)
Procedure Letrero2(corx,cory:byte; cadena:string; color:byte; Tipo:Byte);
Const
  Binario : Array[0..7] Of Byte = (128,64,32,16,8,4,2,1);
Var
  cont : Integer;
  car  : byte;
  reg  : registers;
  ColSec : byte;
  X1, Y1,
  S1, O1,
  S2, O2,
  x,y : word;
Begin
  If LetraUsuario Then
  Begin
    If (Corx In [0..39]) And (Cory In [0..24]) Then
    begin
      S1 := Seg(AlmacenLetra^);  O1 :=Ofs(AlmacenLetra^);
      S2 := Seg(DestinoLet^);  O2 :=Ofs(DestinoLet^);
      For Cont := 0 To Length(Cadena)-1 Do
        For y := 0 To TallaLetraY-1 Do
        Begin
          Car := Ord(Cadena[Cont+1]);
          For x := 0 To TallaLetraX-1 Do
          Case Tipo Of
            1 : If (Mem[S1:O1+Car*TallaLetraY+y] And Binario[x])>0 Then
                begin
                  Mem[S2:O2+(CorY*TallaLetraY+y)*320+
                        CorX*TallaLetraX+x+Cont*TallaLetraX] := Color;
                  Mem[S2:O2+(CorY*TallaLetraY+y+1)*320+
                        CorX*TallaLetraX+x+Cont*TallaLetraX+1] := 0;
                end;
            0 : If (Mem[S1:O1+Car*TallaLetraY+y] And Binario[x])>0 Then
                  Mem[S2:O2+(CorY*TallaLetraY+y)*320+
                        CorX*TallaLetraX+x+Cont*TallaLetraX] := Color
                Else
                  Mem[S2:O2+(CorY*TallaLetraY+y)*320+
                        CorX*TallaLetraX+x+Cont*TallaLetraX] := 0;
          End;
        End;
    end;
  End
```

```
  Else begin
     colsec:=10;
     For cont := 1 to length(cadena) do
       begin
         with reg do
           begin
             car:=ord(cadena[cont]);
             ah:=$02;
             bh:=0;
             dh:=cory;
             dl:=corx;
             intr($10,reg);
             ah:=$0A;
             al:=car;
             bh:=0;
             bl:=color;
             cx:=1;
             intr($10,reg);
           end;
         inc(corx,1);
       end;
   End;
End;
```

PROCEDIMIENTO QUE PONE UN LETRERO EN LA POSICION Y CON EL COLOR INDICADOS. (SOLO MODO OPACO)

```
Procedure Letrero(corx,cory : Byte; cadena : string; color : byte);
Const
  Binario : Array[0..7] Of Byte = (128,64,32,16,8,4,2,1);
Var
  cont : Integer;
  car  : byte;
  reg  : registers;
  ColSec : byte;
  X1, Y1,
  S1, O1,
  S2, O2,
  x,y : word;
Begin
  If LetraUsuario Then
  Begin
     If (Corx In [0..39]) And (Cory In [0..24]) Then
     begin
        S1 := Seg(AlmacenLetra^);   O1 :=Ofs(AlmacenLetra^);
        S2 := Seg(DestinoLet^);   O2 :=Ofs(DestinoLet^);
        For Cont := 0 To Length(Cadena)-1 Do
          For y := 0 To TallaLetraY-1 Do
          Begin
            Car := Ord(Cadena[Cont+1]);
            For x := 0 To TallaLetraX-1 Do
              If (Mem[S1:O1+Car*TallaLetraY+y] And Binario[x])>0 Then
                 Mem[S2:O2+(CorY*8+y)*320+
                           CorX*8+x+Cont*TallaLetraX] := Color
              Else
                 Mem[S2:O2+(CorY*8+y)*320+
                           CorX*8+x+Cont*TallaLetraX] := 0;
          End;
     end;
  End
  Else begin
     colsec:=10;
     For cont := 1 to length(cadena) do
       begin
         with reg do
           begin
             car:=ord(cadena[cont]);
             ah:=$02;
```

```
            bh:=0;
            dh:=cory;
            dl:=corx;
            intr($10,reg);
            ah:=$0A;
            al:=car;
            bh:=0;
            bl:=color;
            cx:=1;
            intr($10,reg);
          end;
{       for x:= corx*8-1 to (corx+1)*8-2 do
          for y := cory*8 to (cory+1)*8-2 do
          begin
            if mem[$A000:x+y*320] <> color then mem[$A000:x+y*320]:=Colsec;
          end;}
        inc(corx,1);
      end;
  End;
End;
```

CONVIERTE UN NUMERO EN CADENA

```
Function NumeroACadena(numero : longint) : string;
var
  texto : string;

Begin
  str(numero,texto);
  numeroacadena:=texto;
End;
```

PONE UN PIXEL DEL COLOR Y EN LAS CORDENADAS INDICADAS

```
Procedure PonPixel(corx, cory : word; color : byte);
Begin
  mem[$A000:corx+cory*320]:=color;
End;
```

RELLENA UN CUADRADO DEL COLOR INDICADO

```
Procedure Cuadrado(xs,ys,xi,yi : integer; color : byte);
var
  x,y : word;
  largo : word;
begin
  largo:=xi-xs+1;
{  for y:=ys to yi do for x:= xs to xi do mem[$A000:x+y*320]:=color;}
  for y:=ys to yi do fillchar(mem[$A000:xs+y*320],largo,color);
end;
```

RELLENA UN CUADRADO DEL COLOR INDICADO SOLO EN AREAS EN NEGRO

```
Procedure Cuadrado2(xs,ys,xi,yi : integer; color : byte);
var
  x,y : word;
begin
  for y:=ys to yi do for x:= xs to xi do
    If mem[$A000:x+y*320]=0 Then mem[$A000:x+y*320]:=color;
end;
```

DIBUJA UN CUADRADO DEL COLOR Y EN LAS COORDENADAS INDICADAS

```
Procedure Cuadro(xs,ys,xi,yi : integer; color : byte );
var
  x, y : Integer;
begin
  if xs>xi then
  Begin
```

```
    x := xi; xi := xs;  xs := x;
  End;
  if ys>yi then
  Begin
    y := yi; yi := ys;  ys := y;
  End;

  If (ys>=0) And (ys<200) Then
    for x:= xs to xi do
      If (x>=0) And (x<320) Then mem[$A000:x+ys*320]:=color;
  If (yi>=0) And (yi<200) Then
    for x:= xs to xi do
      If (x>=0) And (x<320) Then mem[$A000:x+yi*320]:=color;
  If (xs>=0) And (xs<320) Then
    for y:= ys to yi do
      If (y>=0) And (y<200) Then mem[$A000:xs+y*320]:=color;
  If (xi>=0) And (xi<320) Then
    for y:= ys to yi do
      If (y>=0) And (y<200) Then mem[$A000:xi+y*320]:=color;
end;


DEVUELVE LA COORDENADAS DE UN RECTANGULO. SE AUXILIA DE PANTEMPORAL (GLOBAL)
Procedure Rectangulo(Var boton:word; Var xs,ys,xi,yi:integer);
Var
  Bandera,primero : boolean;
  mx,my   : word;
  mxi,myi : integer;
  r       : raton;
  corx, cory : word;
  color   : byte;
  tecla   : char;
  incre   : byte;
Begin
  tecla:=#255;
  bufferteclado;
  delay(50);
  incre:=5;
  boton:=255;
  primero:=true;
  while (boton<>2) and (boton<>4) do
    begin
      r.posraton(boton, corx, cory);
      if keypressed then
        begin
          tecla:=readkey;
          case (tecla) of
           #13 : boton:=4;
           #27 : boton:=2;
           '+' : inc(incre);
           '-' : dec(incre);
          end;
          if incre >15 then incre:=15;
          if incre < 1 then incre:=1;
        end;
      bandera:=false;
      mx:=xs;
      my:=ys;
      mxi:=xi;
      myi:=yi;

      if tecla=#0 then
        begin
          tecla:=readkey;
          case (tecla) of
            #77 : inc(corx,incre);
```

```
      #75 : if corx >= incre then dec(corx,incre) else corx:=0;
      #72 : if cory >= incre then dec(cory,incre) else cory:=0;
      #80 : inc(cory,incre);
      #71 : begin inc(xs,incre); boton:=1; end;
      #73 : begin dec(xs,incre); boton:=1; end;
      #79 : begin inc(ys,incre); boton:=1; end;
      #81 : begin dec(ys,incre); boton:=1; end;
    end;
    r.IniciaRaton(corx,cory);
  end;
if (corx<>xs) and (boton<>1) then
  begin
    xs:=corx;
    bandera:=true;
  end;
if (cory<>ys) and (boton<>1) then
  begin
    ys:=cory;
    bandera:=true;
  end;
if (corx<>xs) and (boton=1) then
  begin
    if corx>xs then inc(xi,corx-xs);
    if corx<xs then dec(xi,xs-corx);
    xs:=mx;
    bandera:=true;
    r.IniciaRaton(xs,ys);
  end;

if (cory<>xs) and (boton=1) then
  begin
    if cory>ys then inc(yi,cory-ys);
    if cory<ys then dec(yi,ys-cory);
    ys:=my;
    bandera:=true;
    r.IniciaRaton(xs,ys);
  end;

if (xi+xs>xs) and (xi+xs>319) then
  begin
    xi:=mxi;
    xs:=319-xi;
    bandera:=true;
    r.IniciaRaton(xs,ys);
  end;

if (yi+ys>ys) and (yi+ys>199) then
  begin
    yi:=myi;
    ys:=199-yi;
    bandera:=true;
    r.IniciaRaton(xs,ys);
  end;

if (xi+xs<xs) and (xi+xs<0) then
  begin
    xi:=mxi;
    xs:=-xi;
    bandera:=true;
    r.IniciaRaton(xs,ys);
  end;

if (yi+ys<ys) and (yi+ys<0) then
  begin
    yi:=myi;
    ys:=-yi;
    bandera:=true;
```

```
          r.IniciaRaton(xs,ys);
        end;

      if (bandera) or (primero) then
        begin
          inc(color);
          if color>15 then color:=1;
          Despliega(pantemporal);
          cuadro(xs,ys,xs+xi,ys+yi,color);
          primero:=false;
        end;
    end;

  Despliega(pantemporal);
End;
```

LEE UNA CADENA DEL TECLADO

```
Procedure LeeCadena(x,y : byte; tam : byte ; tipo : byte;
          Var tecla1,tecla2 : char; Var ante : string; color : byte);
{Tipo
 1 : cadena
 2 : entero positivo y negativo
 3 : flotante
 4 : entero positivo
}
Var
  Cadena : string;
  cont   : byte;
  salir : boolean;
  tecla : char;
  flechas, fin : boolean;
Begin
  flechas:=true;
  fin:=false;
  cadena:=ante; cadena:=cadena+' ';
  if cadena<>'' then cadena:=copy(cadena,1,(pos(' ',cadena)-1));
  cont:=length(cadena);
  salir := false;
  if (tam<1) then
    begin
      letrero(1,1,'Valor no valido',color);
      exit;
    end;
  while salir <> true do
    begin
      letrero(x,y,Espacios(cadena,tam,2),color);
      Cuadro(x*TallaLetraY,(Y+1)*TallaLetraY-1,(x+tam+1)*TallaLetraY-4,
        (y+1)*TallaLetraY-1,0);
      Cuadro((x+Cont)*TallaLetraY,(y+1)*TallaLetraY-1,
        (x+Cont+1)*TallaLetraY-4,(y+1)*TallaLetraY-1,color);
      tecla:=readkey;
      case (tecla) of
        #27 : begin
                ante:=cadena;
                letrero(x,y,cadena+' ',color);
                salir:=true;
                tecla1:=#27;
                tecla2:=#0;
              end;
        #13 : begin
                ante:=cadena;
                salir:=true;
                tecla1:=#13;
                tecla2:=#0;
              end;
        #8  : Begin
                if cont>0 then
```

```
            begin
              dec(cont);
              cadena:=copy(cadena,1,cont);
            end;
        end;
' '..'}' : begin
            if (cont>tam-1) and (fin) then

              begin
                ante:=cadena;
                salir:=true;
              end;
            if cont<=tam-1 then
              begin
                case (tipo) of
                  1 : begin
                        If tecla=#39 then
                          Begin
                            tecla:=readkey;
                            if tecla
                              in ['a','e','i','o','u','n','N'] then
                              begin
                                case tecla of
                                  'a': tecla:=#160;
                                  'e': tecla:=#130;
                                  'i': tecla:=#161;
                                  'o': tecla:=#162;
                                  'u': tecla:=#163;
                                  'n': tecla:=#164;
                                  'N': tecla:=#165;
                                end;
                                cadena:=cadena+tecla;
                                inc(cont);
                              end;
                          end
                        Else
                          Begin
                            cadena:=cadena+tecla;
                            inc(cont);
                          end;
                      end;
                  2 : begin
                        if tecla in['0'..'9','-'] then
                          begin
                            cadena:=cadena+tecla;
                            inc(cont);
                          end;
                      end;
                  3 : begin
                        if tecla in['0'..'9','.'] then
                          begin
                            if (tecla='.') and
                              (pos('.',cadena)<>0) then
                            else
                              begin
                                cadena:=cadena+tecla;
                                inc(cont);
                              end;
                          end;
                      end;
                  4 : begin
                        if tecla in['0'..'9'] then
                          begin
                            cadena:=cadena+tecla;
                            inc(cont);
                          end;
                      end;
```

```
                    end;
                  end;
              end;
        #0 : begin
               tecla:=readkey;
               if flechas then
                 begin
                   ante:=cadena;
                   salir:=true;
                   tecla1:=#0;
                   tecla2:=tecla;
                 end;
             end;
    end;
  end;
end;
```

LEE UN NUMERO DE TIPO INTEGER DEL TECLADO. USA EL PROCEDIMIENTO ANTERIOR

```
Procedure Leeinteger(x,y : byte; tam : byte; Var tecla1, tecla2 : char;
                     Var ante : integer; color : byte);
Var
cadena : string;
codigo : integer;
numero : integer;
cadena2 : string;
Begin
  str(ante,cadena);
  if ante=0 then cadena:='';
  leecadena(x,y,tam,2,tecla1,tecla2,cadena,color);
  val(cadena,numero,codigo);
  ante:=numero;
End;
```

LEE UN NUMERO DE TYPO BYTE DEL TECLADO. USA EL PROCEDIMIENTO ANTERIOR

```
Procedure Leebyte(x,y : byte; tam : byte; Var tecla1, tecla2 : char;
          Var ante : byte; color:byte);
Var
cadena : string;
codigo : integer;
numero : integer;
Begin
  str(ante,cadena);
  cadena:=ceros(cadena,tam,1);
  if ante=0 then cadena:='';
  leecadena(x,y,tam,4,tecla1,tecla2,cadena,color);
  val(cadena,numero,codigo);
  ante:=numero;
End;
```

LEE UN NUMERO DE TIPO PALABRA DEL TECLADO. USA EL PROCEDIMIENTO ANTERIOR

```
Procedure Leeword(x,y : byte; tam : byte; Var tecla1, tecla2 : char;
                  Var ante : word; color : byte);
Var
cadena : string;
codigo : integer;
numero : word;
cadena2 : string;
Begin
  str(ante,cadena);
  cadena:=ceron(cadena,tam,1);
  if ante=0 then cadena:='';
  leecadena(x,y,tam,4,tecla1,tecla2,cadena,color);
  val(cadena,numero,codigo);
  ante:=numero;
End;
```

CARGA LA PALETA DE COLORES

```
PROCEDURE PonPaleta(Var p);
Var
   r      : registers;
   i, Aj1 : integer;
Begin
   Aj1 := 0;
   Port[$3C6] := $FF;
   For i := 0 to 255 Do
   begin
      Port[$3C8] := i;
      Port[$3C9] := Mem[Seg(p):Ofs(p)+Aj1] Shr 2; Inc(Aj1);
      Port[$3C9] := Mem[Seg(p):Ofs(p)+Aj1] Shr 2; Inc(Aj1);
      Port[$3C9] := Mem[Seg(p):Ofs(p)+Aj1] Shr 2; Inc(Aj1);
   End;
   r.ax := $1001;
   r.bh := 0;
   intr($10,r);
End;
```

PONE UN MENSAJE CON OPCION DE RESPUESTA

```
FUNCTION Mensaje(D : String; Tipo : Byte) : Byte;
Var
   Cad       : String;
   i,j,c,
   MaxAncho  : Byte;
   Marcas    : Array[0..10] Of Byte;
   Boton, Xr, Yr : Word;
   Tecla     : Char;
   Salir     : Boolean;
   P, Norma  : Byte;
   r         : raton;
Begin
   Norma := 200 Div TallaLetraX;
   While (D<>'') And (D[1]=' ') Do D := Copy(D,2,Length(D)-1);
   While (D<>'') And (D[Length(D)]=' ') Do D := Copy(D,1,Length(D)-1);
   Cad := '';  j := 0;  C := 0;
   MaxAncho := 10;
   Marcas[0] := 1;
   If Tipo In [1,2] Then
   begin
      P := LibreXMS(30);
      EscribeXMS(XMSMan[P], Mem[$A000:1600], 30400, 0);
   end;
   If D<>'' Then
   Begin
     For i := 1 To Length(D) Do
     Begin
       If (j>Norma) Or
          ((j>Norma-5) And (Not (Upcase(D[i]) In ['A'..'Z','0'..'9']))) Or
          ((j>Norma-2) And (Not (UpCase(D[i]) In ['A','E','I','O','U'])))
       Then
       begin
          Inc(c); Marcas[c] := i;
          If (Marcas[c]-Marcas[c-1])>MaxAncho Then
             MaxAncho := (Marcas[c]-Marcas[c-1]);
          j := 0;
       end Else Inc(j);
     End;
     Inc(c);   Marcas[c] := Length(D)+1;
     If (Marcas[c]-Marcas[c-1])>MaxAncho Then
        MaxAncho := (Marcas[c]-Marcas[c-1]);
     Cuadrado((Norma-6-MaxAncho Div 2)*TallaLetraX,5*TallaLetraY,
       (Norma-4+MaxAncho Div 2)*TallaLetraX,(C+7)*TallaLetraY,2);
     Cuadro((Norma-6-MaxAncho Div 2)*TallaLetraX-1, 5*TallaLetraY-1,
```

```
          (Norma-4+MaxAncho Div 2)*TallaLetraX+1, (C+7)*TallaLetraY+1, 1);
   Cuadro((Norma-6-MaxAncho Div 2)*TallaLetraX-2, 5*TallaLetraY-2,
          (Norma-4+MaxAncho Div 2)*TallaLetraX+2, (C+7)*TallaLetraY+2, 255);
   Case Tipo Of
     1 : Letrero2(Norma-7, C+6, ' SI ', 4, 0);
     2 : Begin
           Letrero2(Norma-10, C+6, ' SI ', 4, 0);
           Letrero2(Norma-4, C+6, ' NO ', 4, 0);
         End;
   End;
   For i := 1 To c Do
   Begin
      If D[Marcas[i-1]]<>' ' Then
        Cad := Copy(D, Marcas[i-1], Marcas[i]-Marcas[i-1])
      Else
        Cad := Copy(D, Marcas[i-1]+1, Marcas[i]-Marcas[i-1]-1);
      If (i<c) And (D[Marcas[i]]<>' ') Then Cad := Cad+'_';
      Letrero2(Norma-5-(Length(Cad) Div 2), i+4, Cad, 5, 0);
   End;
   Cuadrado2((Norma-6-MaxAncho Div 2)*TallaLetraX, 5*TallaLetraY,
     (Norma-4+MaxAncho Div 2)*TallaLetraX, (C+5)*TallaLetraY, 2);
   If Length(Cad)>MaxAncho Then MaxAncho := Length(Cad);
 End;
 Salir := False;
 Repeat
    R.CursorRaton(True);
    Tecla := #1;
    r.PosRaton(Boton, Xr, Yr);
    If Keypressed Then Tecla := UpCase(Readkey);
    Case Tipo Of
      1 : Begin
            If (Odd(Boton)) And
            (r.RangoRaton((Norma-7)*TallaLetraX, (C+6)*TallaLetraY,
                          (Norma-3)*TallaLetraX, (C+7)*TallaLetraY))
              Then Tecla := 'S';
            If (Tecla In ['S',#27,#13]) Then
            Begin
               R.CursorRaton(False);
               Cuadrado2((Norma-7)*TallaLetraX, (C+6)*TallaLetraY,
                         (Norma-3)*TallaLetraX, (c+7)*TallaLetraY, 11);
               Mensaje := 1;
               Salir := True;
            End;
          End;
      2 : Begin
            If (Odd(Boton)) And
            (r.RangoRaton((Norma-10)*TallaLetraX, (C+6)*TallaLetraY,
                          (Norma-6)*TallaLetraX, (C+7)*TallaLetraY))
              Then Tecla := 'S';
            If (Odd(Boton)) And
            (r.RangoRaton((Norma-4)*TallaLetraX, (C+6)*TallaLetraY,
                          (Norma)*TallaLetraX, (C+7)*TallaLetraY))
              Then Tecla := 'N';
            If (Tecla In ['S','N',#27,#13]) Then
            Begin
               R.CursorRaton(False);
               If Tecla In ['S',#13] Then
               begin
                 Mensaje := 1;
                 Cuadrado2((Norma-10)*TallaLetraX, (c+6)*TallaLetraY,
                           (Norma-6)*TallaLetraX, (c+7)*TallaLetraY, 11);
               end;
               If Tecla In [#27,'N'] Then
               begin
                 Mensaje := 2;
                 Cuadrado2((Norma-4)*TallaLetraX, (c+6)*TallaLetraY,
                           (Norma)*TallaLetraX, (c+7)*TallaLetraY, 11);
```

```
                end;
                Salir := True;
              End;
            End;
          Else begin
            Salir := True;  R.CursorRaton(False);
          end;
      End;
  Until Salir;

  r.Libera_Boton;
  If Tipo In [1,2] Then
  begin
    LeeXMS(XMSMan[P], Mem[$A000:1600], 30400, 0);
    LiberaXMS(XMSMan[P]);
    XMSMan[P] := 0;
  end;
End;
```

LEE UN ENTERO

```
PROCEDURE LeeEntero(Var Numero : Word; X, Y : Byte; Enteros : Byte;
                    Var Tecla1, Tecla2 : Char; Var Boton, Xr, Yr:Word;
                    Color : Byte; Var Primero : Boolean);
Const
  Confirma = True;
Var
  Cad,
  CEnt   : String;
  X1     : Byte;
  code   : Integer;
  Rat    : Raton;
Begin
  Str(Numero,CEnt);
  While Length(CEnt)<Enteros Do CEnt := ' '+CEnt;
  X1 := 0;
  Letrero(X,Y,CEnt, 255);
  Cuadrado2(X*8,Y*8,X*8+Enteros*TallaLetraX-1,Y*8+TallaLetraY-1,5);
  Repeat
    Cuadro(X*8,Y*8+TallaLetraY-1,X*8+(Length(CEnt)+1)*TallaLetraX-1,
      Y*8+TallaLetraY-1,0);
    Cuadro(X*8+X1*TallaLetraX,Y*8+TallaLetraY-1,X*8+(X1+1)*TallaLetraX-1,
      Y*8+TallaLetraY-1,4);
    Tecla1 := #1;  Tecla2 := #0;
    Rat.CursorRaton(true);
    Repeat
      If Keypressed Then Tecla1 := Readkey;
      Rat.PosRaton(Boton, Xr, Yr);
      If Odd(Boton) Then
      begin
        If Rat.RangoRaton(X*8,Y*8,X*8+(Length(CEnt)+1)*TallaLetraX,
          Y*8+TallaLetraY)
        Then begin
          X1 := (Xr-X*8) Div TallaLetraX;  Tecla1 := #2;
        end
        Else Tecla1 := #13;
      End;
    Until Tecla1<>#1;
    Rat.CursorRaton(False);
    Case Tecla1 Of
      #0  : Begin
              If Keypressed Then Tecla2 := Readkey;
              Case Tecla2 Of
                #75 : If X1>0 Then
                      Begin
                        Dec(X1);
                        If X1=Enteros Then Dec(X1);
                      End;
```

```
                #77 : If X1<Enteros Then Inc(X1);
                #83 : If X1<Enteros Then CEnt[X1+1] := ' ';
              Else
                Tecla1 := #0;
              End;
          End;
    '0'..'9' : Begin
                 If Primero Then
                 begin
                   CEnt := '';
                   For code := 1 to Enteros Do CEnt := CEnt+' ';
                 end;
                 If X1<Enteros Then
                 begin
                   CEnt[X1+1] := Tecla1;
                   If X1<Enteros Then Inc(X1);
                 end;
               End;
    #8   : If X1>0 Then
           Begin
             If X1<=Enteros Then
             begin
               For code := X1 To Enteros-1 Do
                 CEnt[code] := CEnt[code+1];
               CEnt[Enteros] := ' ';
             end;
             Dec(X1);
           End;
  End;
  Primero := False;
  Letrero2(X,Y,CEnt,Color, 0);
Until (Not (Tecla1 In ['0'..'9',#8,#1,#2]))
  And (Not (Tecla2 In [#77,#75]));
If Tecla1<>#27 Then
begin
  Cad := '';
  For code := 1 To Enteros Do
    If CEnt[code]<>' ' Then Cad := Cad+CEnt[Code];
  Val(Cad, Numero, code);
  Letrero(X,Y,Espacios(Cad,Enteros,2),Color);
end;
end;
```

LEE VARIOS TIPOS DE DATOS

```
PROCEDURE LeeDato(Var Dato; Tipo, X, Y, Talla : Byte;
                  Var Tecla1, Tecla2 : Char; Var Boton, Xr, Yr:Word;
                  Color : Byte);
Const
  Confirma  = True;
Var
  Cad,
  CEnt    : String;
  X1      : Byte;
  Ent,
  code    : Integer;
  Rat     : Raton;
  i,
  S1, O1 : Word;
Begin
  Rat.CursorRaton(False);
  S1 := Seg(Dato);   O1 := Ofs(Dato);
  Case Tipo Of
{ Cadena }
    0 : Move(Dato,Cad,Talla+1);
```

```
{ Palabra }
    1 : If MemW[S1:O1]=0 Then Cad := '' Else Str(MemW[S1:O1],Cad);
{ Entero }
    2 : Begin
          Move(Mem[S1:O1], code, 2);
          If code=0 Then Cad := '' Else Str(code,Cad);
        End;
  End;
  If Cad[0]>Chr(Talla) Then Cad[0] := Chr(Talla);
  X1  := Length(Cad);
  CEnt := Cad;
  Letrero(X,Y,Espacios(Cad,Talla,2), 255);
  Cuadrado2(X*8,Y*8,X*8+Talla*TallaLetraX-1,Y*8+TallaLetraY-1,5);
  Repeat
     Cuadro(X*8,Y*8+TallaLetraY-1,X*8+Talla*TallaLetraX-1,
       y*8+TallaLetraY-1,0);
     Cuadro(X*8+Talla*TallaLetraX,Y*8+TallaLetraY-1,
       X*8+(Talla+1)*TallaLetraX-1,y*8+TallaLetraY-1,2);
     Cuadro(X*8+X1*TallaLetraX,Y*8+TallaLetraY-1,
       X*8+(X1+1)*TallaLetraX-1,y*8+TallaLetraY-1,4);
     Tecla1 := #1;  Tecla2 := #0;
     Rat.CursorRaton(true);
     Repeat
        If Keypressed Then Tecla1 := Readkey;
        Rat.PosRaton(Boton, Xr, Yr);
        If Odd(Boton) Then
        begin
          If Rat.RangoRaton(X*8,Y*8,X*8+(Talla+1)*TallaLetraX,
            Y*8+TallaLetraY)
          Then begin
             X1 := (Xr-X*8) Div TallaLetraX;
             If X1>Length(CEnt) Then X1 := Length(CEnt);
             Tecla1 := #2;
          end
          Else Tecla1 := #13;
        End;
     Until Tecla1<>#1;
     Rat.CursorRaton(False);
     If Tecla1=#0 Then
     Begin
       If Keypressed Then Tecla2 := Readkey;
       Case Tecla2 Of
         #75 : If X1>0 Then
           Begin
             Dec(X1);
             If X1=Talla Then Dec(X1);
           End;
         #77 : If X1<Length(CEnt) Then Inc(X1);
       End;
     End;
     If (Tipo=0) And (Tecla1 In [#33..#254]) And (CEnt[0]<Chr(Talla)) And
        (X1<=Length(CEnt)) Then
        begin
          CEnt := Copy(CEnt,1,X1)+Tecla1+Copy(CEnt,X1+1,Talla-X1+1);
          If X1<Talla Then Inc(X1);
        end;
      If (Tipo In [1,2]) And (Tecla1 In ['0'..'9'])
        And (CEnt[0]<Chr(Talla)) And (X1<Talla) Then
         begin
           If (Tipo=2) And (CEnt<>'') And (CEnt[1]='-') And (X1=0) Then
             CEnt := Copy(CEnt,1,X1+1)+Tecla1+Copy(CEnt,X1+2,Talla-X1+1)
           Else
             CEnt := Copy(CEnt,1,X1)+Tecla1+Copy(CEnt,X1+1,Talla-X1+1);
           Inc(X1);
         end;
      If (Tecla1=#8) And (X1>0) Then
      Begin
```

```
          CEnt := Copy(CEnt,1,X1-1)+Copy(CEnt,X1+1,Talla-X1+1);
          Dec(X1);
        End;
        If (Tipo=2) And (Tecla1='-') Then
        Begin
          If CEnt<>'' Then
          begin
            If CEnt[1]='-' Then CEnt := Copy(CEnt,2,Length(CEnt))
            Else
              If CEnt[0]=Chr(Talla) Then
              CEnt := '-'+Copy(CEnt,1,Length(CEnt)-1)
              Else CEnt := '-'+CEnt;
          End Else CEnt := '-';
          If X1<Talla Then Inc(X1);
        End;
      Letrero(X,Y,Espacios(CEnt,Talla,2),Color);
    Until (Not ((Tecla1 In [#8,#1,#2]) Or
                ((Tecla1 In [#33..#254])    And (Tipo=0)) Or
                ((Tecla1 In ['0'..'9'])     And (Tipo=1)) Or
                ((Tecla1 In ['-','0'..'9']) And (Tipo=2))) And
           (Not (Tecla2 In [#77,#75])));
    If Tecla1<>#27 Then
    begin
      Case Tipo Of
    { Cadena }
        0 : Move(CEnt,Dato,Talla+1);
    { Palabra }
        1 : Begin
              Val(CEnt, i, code);
              MemW[S1:O1]:=i;
            End;
    { Entero }
        2 : Begin
              Val(CEnt, Ent, code);
              Move(Ent, Mem[S1:O1], 2);
            End;
      End;
      Cuadro(X*8+Talla*TallaLetraX,Y*8+TallaLetraY-1,
        X*8+(Talla+1)*TallaLetraX-1,y*8+TallaLetraY-1,2);
      Letrero(X,Y,Espacios(CEnt,Talla,2),Color);
    end;
    Rat.CursorRaton(True);
end;
```

```
LEE EL ARCHIVO DE TIPOS DE LETRA (*.FNT)
PROCEDURE Leer_FontUsuario(Nombre : String; Ancho, Alto : Byte);
Var
   Arch : File;
Begin
{$I-}
  Assign(Arch,Nombre);
  Reset(Arch,1);
  If MaxAvail>FileSize(Arch) Then
  begin
     TamLetra := FileSize(Arch);
     GetMem(AlmacenLetra, TamLetra);
  end;
  BlockRead(Arch, AlmacenLetra^, TamLetra);
  Close(Arch);
  If (IOResult=0) And (AlmacenLetra<>Nil) Then
  begin
    TallaLetraX := Ancho;  TallaLetraY := Alto;
    LetraUsuario := True;
  end Else LetraUsuario := False;
{$I+}
End;
```

DESPLIEGA EN VIDEO EL CONTENIDO DE PANTALLA

```
procedure Despliega(Pantalla : Pointer);
var
  ofs1 : word;
begin
    ofs1 := ofs(Pantalla^);
    asm
      push ds
      push es
      xor  si,si
      xor  di,di
      cld
      mov  si,ofs1
      mov  ax,word(Pantalla+2)
      mov  ds,ax
      mov  ax,0A000h
      mov  es,ax
      mov  cx,7D00h
      rep  movsw
      pop  es
      pop  ds
    end;
end;

BEGIN
   LetraUsuario  := False;   AlmacenLetra := Nil;
   TamLetra      := 0;
   TallaLetraX   := 8;  TallaLetraY := 8;
   Leer_FontUsuario('..\FONT\8x8.fnt',8,8);
   DestinoLet := Ptr($A000,0);
END.
```

LibPal.pas : Librería para el manejo de paletas.

```
UNIT LIBPAL;

INTERFACE
USES
  crt,dos,LibGra,libdnd;

TYPE
  PALETA = ARRAY[0..767] OF BYTE;

  PALETAS = OBJECT
       tamano : array[1..10] of byte;
       vuelta : array[1..10] of byte;
       pal    : paleta;
       actual   : array[0..2] of byte;
       retardo  : array[1..10] of word;
       Procedure iniciar(nombre:string);
       Procedure Rellena(inicio,final,atributo,color:byte);
       Procedure DiscoL(nombre:string; Num : Byte);
       Procedure DiscoE(nombre:string; Num : Byte);
       Procedure DiscoB(nombre:string; Num : Byte);
       Procedure Leepaleta;
       Procedure Ponpaleta;
       Procedure Escala;
       Procedure Escala2(Var Pant);
       Procedure Barre(inicio,final,ciclo,direccion:byte; tiempo:word;
                 numero:byte);
       Procedure Desvanece(inicio,final:byte; tiempo:word; atributo:byte);
       Procedure Aproxima(inicio,final:byte; tiempo:word; atributo:byte;
                 nombre:string);
       Procedure Sombra(x1,y1,x2,y2 : integer; Var Pantalla);
       Procedure MediaPaleta(decre : byte; sentido : byte);
       Procedure IgualaGris;
       Procedure ColorAGris;
       Procedure Paleta_Promedio(Pal2 : Paleta; Var Pant);
       Procedure Iguala_Pal(Var Pant);
     end;
Var
   TotalPal    : Byte;
   FijarPaleta : Boolean;

IMPLEMENTATION
INICIALIZA LAS VARIABLES DE PALETA
Procedure paletas.iniciar(nombre : string);
Var
  contador : byte;
Begin
  for contador:=1 to 10 do tamano[contador]:=0;
  for contador:=1 to 10 do vuelta[contador]:=1;
  for contador:=1 to 10 do retardo[contador]:=0;
  discol(nombre, 1);
end;



PROCEDIMIENTO QUE ALCANZA LA PALETA
Procedure Paletas.Rellena(inicio,final,atributo,color:byte);
{Atributos              }
```

```
{1 - Componente Verde    }
{2 - Componente Rojo     }
{4 - Componente Azul     }
Var
  Contador : word;
Begin
  for contador := inicio to final do
    begin
      if (atributo and 1) = 1 then pal[contador*3]:=color;
      if (atributo and 2) = 2 then pal[contador*3+1]:=color;
      if (atributo and 4) = 4 then pal[contador*3+2]:=color;
    end;
end;
```

PROCEDIMIENTO QUE LEE UNA PALETA DE DISCO

```
Procedure Paletas.DiscoL(nombre : string; Num : Byte);
Var
  archivo : file;
Begin
{$I-}
  assign(archivo,nombre);
  reset(archivo,1);
  If IOResult<>0 Then
  begin
    LeePaleta; exit;
  end;
  If (FileSize(Archivo) Div 768)<Num Then
  begin
    Close(Archivo);  If IOResult=0 Then;  LeePaleta; exit;
  end;
  Seek(archivo,(Num-1)*768);
  blockread(archivo,pal,768);
  TotalPal := FileSize(Archivo) Div 768;
  close(archivo);
  If IOResult<>0 Then;
{$I+}
End;
```

PROCEDIMIENTO QUE ESCRIBE UNA PALETA EN DISCO

```
Procedure Paletas.DiscoE(nombre : string; Num : Byte);
Var
  archivo : file;
Begin
  assign(archivo,nombre);
{$I-}
  reset(archivo,1);
  If IOResult=2 Then ReWrite(Archivo, 1)
  Else
    If IOResult<>0 Then exit;
  Reset(Archivo,1);
  If (FileSize(Archivo) Div 768)<Num Then
    Num := (FileSize(Archivo) Div 768)+1;
  If Num=0 Then Num := 1;
  Seek(Archivo,(num-1)*768);
  blockwrite(archivo,pal,768);
  close(archivo);
  Reset(Archivo,1);


  If IOResult=0 then
  begin
    TotalPal := FileSize(Archivo) Div 768;
    Close(Archivo);
  end;
  If IOResult<>0 Then;
{$I+}
```

```
End;
```

PROCEDIMIENTO QUE BORRA UNA PALETA EN DISCO

```
Procedure Paletas.DiscoB(nombre : string; Num : Byte);
Var
  Arch2,
  archivo : file;
  PalTemp   : Paleta;
  i, Leidos : Word;
Begin
   assign(archivo,nombre);
   Assign(Arch2,'TEMP-PAL.$$$');
{$I-}
   ReWrite(Arch2,1);
   Reset(archivo,1);
   If IOResult<>0 Then exit;
   i := 0;
   Repeat
      BlockRead(Archivo, PalTemp, 768, Leidos);
      If Leidos>0 Then
      begin
         Inc(i);
         If i<>Num Then BlockWrite(Arch2, PalTemp, 768);
         If IOResult<>0 Then;
      End;
   Until Leidos=0;
   TotalPal := i-1;
   Close(archivo);   Close(Arch2);
   Erase(Archivo);
   Rename(Arch2, Nombre);
   If IOResult<>0 Then;
{$I+}
End;
```

LEE LA PALETA DEL VIDEO ACTUAL

```
Procedure paletas.LeePaleta;
Var
 seg1,ofs1 : word;
Begin
  seg1:=seg(pal[0]);
  ofs1:=ofs(pal[0]);
  asm
    mov ax,seg1
    mov es,ax
    mov ax,ofs1
    mov dx,ax
    mov ax,$1017
    mov bx,0
    mov cx,256
    int $10
  end;
End;
```

ESCRIBE LA PALETA EN EL VIDEO ACTUAL

```
Procedure paletas.PonPaleta;
Var
 seg1, ofs1 : word;
Begin
  seg1:=seg(pal[0]);
  ofs1:=ofs(pal[0]);
  asm
    mov ax, seg1
    mov es, ax
    mov ax, ofs1
```

```
    mov dx, ax
    mov ax, $1012
    mov bx, 0
    mov cx, 256
    int $10
  end;
End;
```

DESPLIEGA LA PALETA ACTUAL DE COLORES

```
procedure paletas.escala;
Var
  corx : word;
  cory : word;
  color : byte;
begin
  cuadrado(67,17,231,133,0);
  corx:=70;
  cory:=20;
  color:=0;
  cuadro(68,18,230,132,255);
  cuadro(67,17,231,133,6);
  repeat
    repeat
      cuadrado(corx,cory,corx+8,cory+5,color);
      inc(corx,10);
      inc(color);
    until corx=230;
    corx:=70;
    inc(cory,7);
  until cory=132;
end;
```

DESPLIEGA LA PALETA ACTUAL DE COLORES EN USO

```
procedure paletas.escala2(Var Pant);
Var
  corx : word;
  cory : word;
  color : byte;
  Ofs1,i,
  Seg1  : Word;
  Usados : Array[0..255] Of Boolean;
begin
  Seg1 := Seg(Pant);     Ofs1 := Ofs(Pant);
  For i := 0 To 255 Do Usados[i] := False;
  For i := 0 To 63999 Do Usados[Mem[Seg1:Ofs1+i]] := True;
  cuadrado(67,17,231,133,0);
  corx:=70;
  cory:=20;
  color:=0;
  cuadro(68,18,230,132,255);
  cuadro(67,17,231,133,6);
  repeat
    repeat
      If Usados[Color] Then Cuadro(corx-1,cory-1,corx+9,cory+6,255)
        Else Cuadro(corx-1,cory-1,corx+9,cory+6,0);
      cuadrado(corx,cory,corx+8,cory+5,color);
      inc(corx,10);
      inc(color);
    until corx=230;
    corx:=70;
    inc(cory,7);
  until cory=132;
end;
```

PROCEDIMIENTO DE BARRIDO DE PALETA

```
Procedure paletas.barre(inicio,final,ciclo,direccion : byte;
```

```
            tiempo : word; numero:byte);
Var
  valor : byte;
  byte1 : byte;
  cont  : byte;
Begin
  if (retardo[numero]<>0) and (retardo[numero]<tiempo) then
     begin
       inc(retardo[numero]);
       exit;
     end;
  inc(retardo[numero]);
  valor:=final-inicio+1;
  if tamano[numero]<valor then
    begin
      if direccion=1 then
        begin
          actual[0]:=pal[inicio*3];
          actual[1]:=pal[inicio*3+1];
          actual[2]:=pal[inicio*3+2];
          for byte1 := inicio to final do
            begin
              pal[byte1*3]:=pal[(byte1+1)*3];
              pal[byte1*3+1]:=pal[(byte1+1)*3+1];
              pal[byte1*3+2]:=pal[(byte1+1)*3+2];
            end;
          pal[final*3]:=actual[0];
          pal[final*3+1]:=actual[1];
          pal[final*3+2]:=actual[2];
          ponpaleta;
          inc(tamano[numero]);
          retardo[numero]:=1;
        end;
      if direccion=2 then
        begin
          actual[0]:=pal[final*3];
          actual[1]:=pal[final*3+1];
          actual[2]:=pal[final*3+2];
          for byte1 := final downto inicio do
            begin
              pal[byte1*3]:=pal[(byte1-1)*3];
              pal[byte1*3+1]:=pal[(byte1-1)*3+1];
              pal[byte1*3+2]:=pal[(byte1-1)*3+2];
            end;
          pal[inicio*3]:=actual[0];
          pal[inicio*3+1]:=actual[1];
          pal[inicio*3+2]:=actual[2];
          ponpaleta;
          inc(tamano[numero]);
          retardo[numero]:=1;
        end;
    end;
  if ((vuelta[numero]<ciclo) and (tamano[numero]>=valor)) or (ciclo=255) then
    begin
      tamano[numero]:=1;
      inc(vuelta[numero]);
    end;
End;
```

PROCEDIMIENTO DE BORRADO DE PALETA

```
Procedure Paletas.Desvanece(inicio,final:byte; tiempo:word; atributo:byte);
{Atributos                }
{1 - Componente Verde     }
{2 - Componente Rojo      }
{4 - Componente Azul      }
Var
  Contador : word;
```

```
  arre : array[0..767] of boolean;
  salir : boolean;
Begin
  salir:=false;
  for contador := inicio to final do
    begin
      if (atributo and 1) = 1 then arre[contador*3]:=false;
      if (atributo and 2) = 2 then arre[contador*3+1]:=false;
      if (atributo and 4) = 4 then arre[contador*3+2]:=false;
    end;
  while salir<>true do
  begin
    for contador := inicio to final do
      begin
        if (atributo and 1) = 1 then
           if pal[contador*3] > 0 then dec(pal[contador*3]);
        if (atributo and 2) = 2 then
           if pal[contador*3+1] > 0 then dec(pal[contador*3+1]);
        if (atributo and 4) = 4 then
           if pal[contador*3+2] > 0 then dec(pal[contador*3+2]);
        if (atributo and 1) = 1 then
           if pal[contador*3] = 0 then arre[contador*3]:=true;
        if (atributo and 2) = 2 then
           if pal[contador*3+1] = 0 then arre[contador*3+1]:=true;
        if (atributo and 4) = 4 then
           if pal[contador*3+2] = 0 then arre[contador*3+2]:=true;
      end;
    delay(tiempo);
    ponpaleta;
    salir:=true;
    for contador := inicio to final do
      begin
        if (atributo and 1) = 1 then
           if arre[contador*3]=false then salir:=false;
        if (atributo and 2) = 2 then
           if arre[contador*3+1]=false then salir:=false;
        if (atributo and 4) = 4 then
           if arre[contador*3+2]=false then salir:=false;
      end;
  end;
End;
```

PROCEDIMIENTO QUE ALCANZA LA PALETA

```
Procedure Paletas.Aproxima(inicio,final:byte; tiempo:word;
          atributo:byte; nombre:string);
```

```
Atributos
  1 - Componente Verde
  2 - Componente Rojo
  4 - Componente Azul
```

```
Var
  Palsec : paleta;
  Contador : word;
  arre : array[0..767] of boolean;
  salir : boolean;
Begin
  move(pal[0],palsec[0],768);
  discol(nombre,1);
  for contador := inicio to final do
    begin
      if (atributo and 1) = 1 then palsec[contador*3]:=pal[contador*3];
      if (atributo and 2) = 2 then palsec[contador*3+1]:=pal[contador*3+1];
      if (atributo and 4) = 4 then palsec[contador*3+2]:=pal[contador*3+2];
    end;
  move(palsec[0],pal[0],768);
  for contador := inicio to final do
    begin
```

```
      if (atributo and 1) = 1 then pal[contador*3]:=0;
      if (atributo and 2) = 2 then pal[contador*3+1]:=0;
      if (atributo and 4) = 4 then pal[contador*3+2]:=0;
    end;
  salir:=false;
  for contador := inicio to final do
    begin
      if (atributo and 1) = 1 then arre[contador*3]:=false;
      if (atributo and 2) = 2 then arre[contador*3+1]:=false;
      if (atributo and 4) = 4 then arre[contador*3+2]:=false;
    end;
  while salir<>true do
  begin
    for contador := inicio to final do
      begin
        if (atributo and 1) = 1 then
           if pal[contador*3] < palsec[contador*3] then
              inc(pal[contador*3]);
        if (atributo and 2) = 2 then
           if pal[contador*3+1] < palsec[contador*3+1] then
              inc(pal[contador*3+1]);
        if (atributo and 4) = 4 then
           if pal[contador*3+2] < palsec[contador*3+2] then
              inc(pal[contador*3+2]);
        if (atributo and 1) = 1 then
           if pal[contador*3] = palsec[contador*3] then
              arre[contador*3]:=true;
        if (atributo and 2) = 2 then
           if pal[contador*3+1] = palsec[contador*3+1] then
              arre[contador*3+1]:=true;
        if (atributo and 4) = 4 then
           if pal[contador*3+2] = palsec[contador*3+2] then
              arre[contador*3+2]:=true;
      end;
    delay(tiempo);
    ponpaleta;

    salir:=true;
    for contador := inicio to final do
      begin
        if (atributo and 1) = 1 then
           if arre[contador*3]=false then salir:=false;
        if (atributo and 2) = 2 then
           if arre[contador*3+1]=false then salir:=false;
        if (atributo and 4) = 4 then
           if arre[contador*3+2]=false then salir:=false;
      end;
  end;
End;
```

```
CAMBIA UN RANGO DE VIDEO EN SUMA DE 128 BYTES A SU PALETA
Procedure Paletas.Sombra(x1,y1,x2,y2 : integer; Var Pantalla);
Var
   i, largo, ancho,
   L1 , A1,
   A2, L2,
   Ofs1A, Ofs2a,
   Seg1, Ofs1,
   Seg2, Ofs2,
   IniX, IniY,
   FinX, FinY,
   Talla : Word;
   p, p1 : Pointer;
   apunta : pointer;
   label c1, c2, c3;
Begin
   ancho:=x2-x1+1;
```

```
    largo:=y2-y1;
    apunta := Addr(mem[seg(pantalla):ofs(pantalla)]);
    seg1 :=seg(apunta^);
    ofs1 :=ofs(apunta^);
    seg2:=seg1;
    ofs2:=ofs1;
    A2 := 0;
    L2 := 0;
    IniX := X1;
    FinX := X1+Ancho;
    IniY := Y1;
    FinY := Y1+Largo;
    If X1<VenX1 Then
       If X1+Ancho<VenX1 Then
          exit
       Else begin
          IniX := VenX1;
          A2   := VenX1 - X1;
       end;
    If X1>VenX2 Then
      Exit
    Else
       If X1+Ancho>VenX2 Then
          FinX := VenX2;
    A1 := FinX-IniX;

    If Y1<VenY1 Then
       If Y1+Largo<VenY1 Then
          exit
       Else begin
          IniY := VenY1;
          L2   := VenY1 - Y1;
       end;
    If Y1>VenY2 Then
       Exit
    Else
       If Y1+Largo>VenY2 Then
          FinY := VenY2;
    L1 := FinY-IniY;

    If ((L2+L1)>0) And (A1>0) Then
    For i:= L2 To L2+L1 Do
    begin
       Ofs2A := Ofs2+i*Ancho+A2;
       Ofs1A := Ofs1+IniX+(IniY+i-L2)*320;
       if (ofs2a>64000) or (ofs1A>64000) then
          ancho := ancho;
  asm
     push ds
     mov ax, Seg2
     mov ds, ax
     mov ax, Ofs1A
     mov si, ax
     mov ax, ofs1A
     mov di, ax
     mov ax, seg1
     mov es, ax
     mov cx, A1
c1:  mov ah, ds:[si]
     add ah,128
     mov es:[di], ah
c2:  inc di
     inc si
c3:  loop c1
     pop ds
  end;
   end;
```

```
{  cuadro(x1,y1,x2,y2,1);
  cuadro(x1,y1,x2-1,y2-1,10);}
end;
```

AUMENTA O DECREMENTA MEDIA PALETA

```
Procedure Paletas.MediaPaleta(decre : byte; sentido : byte);
Var
  archi : paleta;
  cont1 : byte;
Begin
  if sentido = 0 then
    begin
      for cont1:=0 to 127 do
        begin
          if pal[cont1*3] >= decre then
             pal[(cont1+128)*3]:=pal[cont1*3]-decre
          else pal[(cont1+128)*3]:=0;
          if pal[cont1*3+1] >= decre then
             pal[(cont1+128)*3+1]:=pal[cont1*3+1]-decre
          else pal[(cont1+128)*3+1]:=0;
          if pal[cont1*3+2] >= decre then
             pal[(cont1+128)*3+2]:=pal[cont1*3+2]-decre
          else pal[(cont1+128)*3+2]:=0;
        end;
    end;
  if sentido = 1 then
    begin
      inc(decre,67);
      for cont1:=0 to 127 do
        begin
          if (127-pal[cont1*3]) >= decre then
             pal[(cont1+128)*3]:=pal[cont1*3]+decre
          else pal[(cont1+128)*3]:=127;
          if (127-pal[cont1*3+1]) >= decre then
             pal[(cont1+128)*3+1]:=pal[cont1*3+1]+decre
          else pal[(cont1+128)*3+1]:=127;
          if (127-pal[cont1*3+2]) >= decre then
             pal[(cont1+128)*3+2]:=pal[cont1*3+2]+decre
          else pal[(cont1+128)*3+2]:=127;
        end;
    end;
End;
```

DISMINUYE LOS COLORES DE UNA IMAGEN EN TONOS DE GRIS

```
procedure Paletas.IgualaGris;
Var
  cont1 : byte;
  cont  : word;
Begin
  for cont1:=0 to 79 do
    begin
      pal[cont1*3+16*3]:=pal[(cont1*3)*3+16*3];
      pal[cont1*3+1+16*3]:=pal[(cont1*3)*3+1+16*3];
      pal[cont1*3+2+16*3]:=pal[(cont1*3)*3+2+16*3];
    end;
  for cont := 0 to tam-1 do
   Begin
     if mem[$A000:cont]>230 then mem[$A000:cont]:=95 else
     if mem[$A000:cont]=15 then mem[$a000:cont]:=95 else
     mem[$A000:cont]:=(mem[$A000:cont] div 3) + (mem[$A000:cont] mod 3) + 17
   End;
End;
```

CAMBIA LA PALETA DE COLOR A TONOS DE GRIS

```
Procedure Paletas.ColorAgris;
var
```

```
  cont, promedio : byte;
Begin
  for cont:=0 to 255 do
    begin
      promedio:=(pal[cont*3] + pal[cont*3+1] + pal[cont*3+2] ) div 3;
      pal[cont*3]:=promedio;
      pal[cont*3+1]:=promedio;
      pal[cont*3+2]:=promedio;
    end;
  ponpaleta;
End;
```

ENCUENTRA EL VALOR PROMEDIO DE UNA PALETA DADA Y LA PALETA ACTUAL

```
PROCEDURE Paletas.Paleta_Promedio(Pal2 : Paleta; Var Pant);
Var
   Orden    : Array[1..3] Of Byte;
   Prom1,
   Prom2,
   Dif      : LongInt;
   Optimo,
   PosDif,
   i, j, k     : Word;
   Sustituye  : Array[0..255] Of Byte;

Begin
   Mensaje('Igualando Paletas...',0);
   For j := 0 To 255 Do Sustituye[j] := j;
   For i := 0 To 255 Do
   If i<>254 Then
   begin
     For j := 1 To 3 Do Orden[j] := j-1;
     For j := 0 To 2 Do
     begin
        Optimo := j+1;
        For PosDif := j+1 To 2 Do
          If Pal2[i*3+Orden[PosDif]]>Pal2[i*3+Orden[Optimo]]
           Then Optimo := PosDif+1;
        k := Orden[j+1];
        Orden[j+1]     := Orden[Optimo];
        Orden[Optimo]  := k;
     end;

     Optimo := 0;  Dif := 1677215;
     For j := 0 To 253 Do
     begin
       Prom1 := Abs(Pal[j*3+Orden[1]]-Pal2[i*3+Orden[1]])+
                Abs(Pal[j*3+Orden[2]]-Pal2[i*3+Orden[2]])+
                Abs(Pal[j*3+Orden[3]]-Pal2[i*3+Orden[3]]);
       If Prom1<Dif Then
          begin
             Dif := Prom1;
             Optimo := j;
             If Dif=0 Then j := 253;
          end;
     end;
     Sustituye[i] := Optimo;
   End;
   For j := 0 To 63999 Do
     Mem[Seg(Pant):Ofs(Pant)+j] := Sustituye[Mem[Seg(Pant):Ofs(Pant)+j]];
   Mensaje('Igualacion concluida...',0);
End;
```

IGUALA DOS PALETAS

```
PROCEDURE Paletas.Iguala_Pal(Var Pant);
Var
   Pal1, Pal2 : Paleta;
```

```
Begin
   Pal1 := Pal;
   LeePaleta;
   Pal2 := Pal;
   Pal  := Pal1;
   Paleta_Promedio(Pal2, Pant);
   PonPaleta;
End;
```

RUTINA DE INICIALIZACION

```
BEGIN
   TotalPal := 0;
   FijarPaleta := True;
END.
```

LibPan.pas : Libreria para el manejo de otros formatos graficos.

```
Unit LibPan;
INTERFACE
Uses dos, crt, LibGra;
Type

  Formato = Object
    Procedure VerPCX(nombreimagen : string; Var Panta_var);
    Procedure VerBin(nombreimagen : string; Var Panta_var);
    Procedure VerPuc(nombreimagen : string; Var Panta_var);
    Procedure EscribeBin(nombreimagen : string; Var Panta_var);
    Procedure VerGIF(nombreimagen : string; Var Panta_var);
    Procedure VerAVI(nombreimagen : string; Var Panta_var);
  End;

IMPLEMENTATION
PROCEDIMIENTO QUE LEE UNA IMAGEN DE DISCO EN FORMATO PCX
Uses LibXMS, LibDir;
Procedure Formato.VerPCX(nombreimagen : string; Var Panta_var);
Type

  CabPCX = Record                          {Cabecera del formato PCX}
    Manufactura, Version,
    Codif, bits_por_pixel  : Byte;
    xmin, ymin, xmax, ymax,
    resh, resv : Word;
    palette : Array[0..47] of byte;
    Reservado, PlanosColor : Byte;
    byte_por_Linea, TipoPaleta : Word;
    relleno : Array[0..57] Of Byte;
  end;

Const
   MaxAlmacen = 2000;
var
   Cabecera   : CabPCX;
   Bits,Bytes,Ancho,Largo,c,n,ofs1,seg1,leidos,tam,indice : Word;
   i, j, c2,b    : Byte;
   Paleta     : Array[0..767] Of byte;
   Arch       : File;
   Nombre2, Nombre, parametro1,parametro2 : String;
   almacen    : pointer;
   Talla      : LongInt;
   Ordenar    : Boolean;
```

```
   tecla       : char;

Begin
  parametro1:=''; parametro2:='';
  Ordenar := False;
  Nombre := nombreimagen;
  If Pos('.', Nombre)=0 Then Nombre := Nombre+'.PCX';
  Nombre2 := '';
  nombre2:=parametro1;
  Assign(Arch, Nombre);
{$I-}
  Reset(Arch, 1);
  if ioresult<>0 then
    begin
      fillchar(mem[$A000:0],tam,0);
      letrero(1,1,'ERROR EN TIEMPO DE EJECUCION',1);
      letrero(1,3,'REPORTADO POR VERPCX',1);
      letrero(1,5,'Archivo '+nombreimagen+' no encontrado',3);
      exit;
    end;
  getmem(almacen,Maxalmacen);
  If Nombre2='' Then Nombre2 := Copy(Nombre,1,Pos('.',Nombre))+'BIN';
  Talla := FileSize(Arch);
  BlockRead(Arch, Cabecera, SizeOf(Cabecera));
  If Cabecera.bits_por_pixel=1 Then Bits:=Cabecera.PlanosColor
    Else bits := Cabecera.Bits_por_pixel;
  If Talla<769 Then begin end
  Else begin
      Seek(Arch, Talla-769);
      BlockRead(Arch, i, 1);
      BlockRead(Arch, paleta, 768);
  end;
  Seek(Arch,128);
  Ancho := (Cabecera.xmax-Cabecera.xmin)+1;
  Largo := (Cabecera.ymax-Cabecera.ymin)+1;
  Tam   := Ancho * 200;
  If Largo>200 Then Largo := 200;
  Largo := (Cabecera.ymax-Cabecera.ymin)+1;
  Largo := (Cabecera.ymax-Cabecera.ymin);
  bytes := cabecera.byte_por_linea;
  For Seg1 := 0 To 767 Do Paleta[Seg1] := Paleta[Seg1] Shr 2;
  Seg1 := Seg(Paleta);
  Ofs1 := Ofs(Paleta);
  ASM
     mov ax, $1012
     mov bx, 0
     mov cx, 256
     mov dx, ofs1
     mov es, seg1
     int $10
  End;
  c := 0;
  Indice := 0;
  Tecla := #0;
  While (c<=Largo) And (c<200) Do
  begin
     n := 0;
     Repeat
        {$I-}
           Leidos := 1;
           If (Indice>=MaxAlmacen) Or (Indice=0) Then
           begin
              BlockRead(Arch, Almacen^, MaxAlmacen, Leidos);
              Indice := 0;
           end;
           Inc(Indice);
           c2 := mem[seg(Almacen^):ofs(almacen^)+Indice-1];
```

```
        {$I+}
           C2 := C2 and $FF;
           If (C2 And $C0)=$C0 Then
           begin
              i := C2 And $3F;
              {$I-}
                 Leidos := 1;
                 If (Indice>=MaxAlmacen) Or (Indice=0) Then
                 begin
                    BlockRead(Arch, Almacen^, MaxAlmacen, Leidos);
                    Indice := 0;
                 end;
                 Inc(Indice);
                 c2 := mem[seg(almacen^):ofs(almacen^)+indice-1];
              {$I+}
              For j := 1 To i Do
              begin
                 If n<320 Then
                 Mem[seg(Panta_var):ofs(Panta_var)+c*320+n] := c2;
                 inc(n);
              end;
           end
           Else begin
              If n<320 Then
              Mem[seg(Panta_var):ofs(Panta_var)+c*320+n] := c2;
              inc(n);
           end;
     Until n>=Bytes;
     Inc(c);
     If Keypressed Then Tecla := Readkey;
  end;
  If Ordenar Then
    For c:=0 To 63999 Do
    begin
       b := Mem[seg(Panta_var):ofs(Panta_var)+C];
       Mem[seg(Panta_var):ofs(Panta_var)+c] := b;
    end;
  While KeyPressed Do Tecla := Readkey;
  Close(Arch);
{$I+}
  freemem(almacen,maxalmacen);
End;
```

PROCEDIMIENTO QUE LEE UNA IMAGEN DE DISCO EN FORMATO BINARIO

```
Procedure Formato.VerBin(nombreimagen : String; Var Panta_var);
Var
  archivo : file;
  Seg1, Ofs1 : word;
  pal1 : array[0..767] of byte;
Begin
  assign(archivo,nombreimagen);
  {$I-}
  reset(archivo,1);
  if ioresult<>0 then
    begin
     fillchar(mem[$A000:0],tam,0);
     letrero(1,1,'ERROR EN TIEMPO DE EJECUCION',1);
     letrero(1,3,'REPORTADO POR VERBIN',1);
     letrero(1,5,'Archivo '+nombreimagen+' no encontrado',3);
     exit;
    end;
  blockread(archivo,pal1,768);
  Seg1 := Seg(Pal1);
  Ofs1 := Ofs(Pal1);
  ASM
     mov ax, $1012
     mov bx, 0
```

```
    mov cx, 256
    mov dx, ofs1
    mov es, seg1
    int $10
  End;
  blockread(archivo,mem[seg(Panta_var):ofs(Panta_var)],tam);
  close(archivo);
  {$I+}
End;
```

PROCEDIMIENTO QUE LEE UNA IMAGEN DE DISCO EN FORMATO BINARIO

```
procedure Formato.VerPuc(nombreimagen : string; Var Panta_var);
var
   fuente    : file;
   comienzo,leido,byte1 : byte;
   cont,cont2 : byte;
   tamano, cont3 : integer;
   apuntador,apuntador2 : longint;
   p : pointer;
   NumR,NumW: Word;
   buffer: array[1..2048] of byte;
   bloques, residuo,vueltas : integer;
   seg1,ofs1 : word;

begin
{$I-}
  if nombreimagen='' then nombreimagen:='tttt';
  assign(fuente,nombreimagen);
  reset(fuente,1);
  if ioresult<>0 then
    begin
     fillchar(mem[$A000:0],tam,0);
     letrero(1,1,'ERROR EN TIEMPO DE EJECUCION',1);
     letrero(1,3,'REPORTADO POR VERPUC',1);
     letrero(1,5,'Archivo '+nombreimagen+' no encontrado',3);
     exit;
    end;
  tamano:=filesize(fuente)-1;
  getmem(p,filesize(fuente));
  apuntador2:=0;
  repeat
    BlockRead(fuente,buffer,SizeOf(buffer),NumR);
    for apuntador:=1 to numr do
      begin
        byte1:=buffer[apuntador];
        mem[seg(p^):ofs(p^)+apuntador2]:=byte1;
        inc(apuntador2);
      end;
  until (NumR = 0);
  close(fuente);
  bloques:= tamano div 2048;
  residuo:= tamano mod 2048;
  vueltas:=1;
  comienzo:=mem[seg(p^):ofs(p^)+0];
  cont3:=0;
  apuntador:=0;
  while tamano > cont3 do
    begin
      inc(cont3); byte1:=mem[seg(p^):ofs(p^)+cont3];
      if byte1=comienzo then
        begin
          inc(cont3); leido:=mem[seg(p^):ofs(p^)+cont3];
          inc(cont3); cont2:=mem[seg(p^):ofs(p^)+cont3];
          for cont:=1 to cont2 do
            begin
              inc(apuntador);mem[seg(Panta_var):ofs(Panta_var)+
                apuntador-769]:=leido;
```

```
          end;
        end
      else
        begin
          inc(apuntador);mem[seg(Panta_var):ofs(Panta_var)+
            apuntador-769]:=bytei;
        end;
    end;
  Segl := Seg(Panta_var);
  Ofsl := Ofs(Panta_var)-768;
  ASM
     mov ax, $1012
     mov bx, 0
     mov cx, 256
     mov dx, ofsl
     mov es, segl
     int $10
  End;
  freemem(p,tamano+1);
{I+}
end;
```

PROCEDIMIENTO QUE ESCRIBE UNA IMAGEN DE DISCO EN FORMATO BINARIO

```
Procedure Formato.EscribeBin(nombreimagen : string; Var Panta_var);
Var
  archivo : file;
  Segl, Ofsl : word;
  pal1 : array[0..767] of byte;
Begin
  assign(archivo,nombreimagen);
  {$I-}
  rewrite(archivo,1);
  Segl := Seg(Pal1);
  Ofsl := Ofs(Pal1);
  ASM
     mov ax, $1017
     mov bx, 0
     mov cx, 256
     mov dx, ofsl
     mov es, segl
     int $10
  End;
  blockwrite(archivo,pal1,768);
  blockwrite(archivo,mem[seg(Panta_var):ofs(Panta_var)],tam);
  close(archivo);
  {$I+}
End;
```

RUTINAS DE APOYO PARA EL MANEJO DEL FORMATO GIF

```
{$I-}
CONST
   NO_CODE       = -1;

   SCREENWIDE    = 320;
   SCREENDEEP    = 200;
   STEP          = 32;

   HOME          = $4700;
   CURSOR_UP     = $4800;
   CURSOR_LEFT   = $4b00;
   CURSOR_RIGHT  = $4d00;
   FIN           = $4f00;
   CURSOR_DOWN   = $5000;
TYPE
  GIFHEADER = Record
     sig                    : Array[0..5] Of Char;
```

```
    screenwidth, screendepth  : Word;
    flags, background, aspect : Byte;
  End;
  IMAGEBLOCK = Record
    left, top, width, depth : Word;
    flags                   : Byte;
  End;

Var
   Destino : Pointer;

{ Esta función es llamada cuando el decodificador de GIF encuentra
  una extensión }
PROCEDURE doextension(Var Arch : File);
Var
   D    : Byte;
   N, I : Integer;
Begin
{ Saltando la extensión }
   Repeat
    BlockRead(Arch, D, 1);
    If Not (EOF(Arch)) Then
      For i := 1 To D Do BlockRead(Arch, n, 1);
   Until (n=0) Or (EOF(Arch));
End;

{ Descompacta una imagen LZW }
PROCEDURE unpackimage(Var fp : File; buffer : Pointer;
                      bits,width,depth,flags : Integer);
Const
   wordmasktable : Array[0..15] Of Word =
        ($0000,$0001,$0003,$0007,
         $000f,$001f,$003f,$007f,
         $00ff,$01ff,$03ff,$07ff,
         $0fff,$1fff,$3fff,$7fff );
   inctable  : Array[0..4] Of Byte = (8,8,4,2,0);
   startable : Array[0..4] Of Byte = (0,4,2,1,0);
Var
   bits2, codesize, codesize2,
   nextcode, thiscode, oldtoken,
   currentcode, oldcode, bitsleft,
   blocksize, line,
   byte1, pass                    : Integer;
   p, q, u          : Byte;
   linebuffer       : Pointer;
   b                : Array[0..255] Of Byte;
   firstcodestack   : Array[0..4095] Of Byte;
   lastcodestack    : Array[0..4095] Of Byte;
   codestack        : Array[0..4095] Of Word;
   Dato             : Byte;
   Salir, Salir2    : boolean;
   label sigue;
Begin
   Line := 0;   byte1 := 0;   Pass := 0;
   p := 0;  q := 0;
   bitsleft  := 8;
   if (bits<2) Or (bits>8) Then exit;
   bits2     := 1 Shl bits;
   nextcode  := bits2+2;
   CodeSize  := bits+1;
   codesize2 := 1 Shl codesize;
   oldtoken  := NO_CODE;
   oldcode   := oldtoken;
   If MaxAvail<Width Then exit;
   GetMem(LineBuffer, Width);

  { repite hasta que algo rompa el ciclo }
```

```
   Salir := False;
   Repeat
sigue:
      if bitsleft=8 Then
       Begin
          Inc(p);
          If p>=q Then
          Begin
            BlockRead(fp, Dato, 1);  BlockSize := Dato;
            BlockRead(fp, b, BlockSize);
            p := 0;   q := BlockSize;
            If (BlockSize<1) Or (IOResult<>0) Then
            Begin
              freeMem(linebuffer,Width);  exit;
            End;
          End;
          bitsleft := 0;
       End;
       thiscode := b[p];
       currentcode:=codesize+bitsleft;
       if (currentcode<=8) Then
       Begin
         b[p]      := b[p] Shr CodeSize;
      bitsleft := currentcode;
       End
      else Begin
           Inc(p);
           If p>=q Then
           Begin
             BlockRead(fp, Dato, 1);  BlockSize := Dato;
             BlockRead(fp, b, BlockSize);
             p := 0;   q := BlockSize;
             If (BlockSize<1) Or (IOResult<>0) Then
             Begin
               freeMem(linebuffer,Width);
               exit;
             End;
           End;
       thiscode := Thiscode Or (b[p] Shl (8-bitsleft));
       if currentcode<=16 Then
          begin
            bitsleft := Currentcode-8;
            b[p] := b[p] Shr bitsleft;
          end
          else Begin
            Inc(p);
         if p>=q Then
            Begin
               BlockRead(fp, Dato, 1);  BlockSize := Dato;
               BlockRead(fp, b, BlockSize);
               p := 0;   q := BlockSize;


               If (BlockSize<1) Or (IOResult<>0) Then
               begin
           freeMem(linebuffer, Width);  exit;
               End
          End;
          thiscode := thiscode Or (b[p] Shl (16-bitsleft));
            bitsleft := currentcode-16;
            b[p] := b[p] Shr bitsleft;
       End;
      End;
          thiscode := thiscode And wordmasktable[codesize];
          currentcode := thiscode;
          if thiscode=(bits2+1) Then Salir := True;
          If Not Salir Then
```

```
begin
  If thiscode>nextcode Then
  begin
     FreeMem(linebuffer, Width);  exit;
  end;
  If Thiscode=bits2 Then
  begin
     nextcode  := bits2+2;
     codesize  := bits+1;
     codesize2 := 1 Shl codesize;
     oldcode   := NO_CODE;
     oldtoken  := oldcode;
     goto sigue;
  End;
  u := 0;

  if thiscode=nextcode Then
  begin
     if oldcode=NO_CODE Then
     begin
       FreeMem(linebuffer, Width);  exit;
     end;
     firstcodestack[u] := oldtoken;
     inc(u);
     thiscode := oldcode;
  end;
  While thiscode>=bits2 Do
  begin
     firstcodestack[u] := lastcodestack[thiscode];
     Inc(u);
     thiscode := codestack[thiscode];
  end;
  oldtoken := thiscode;
  Salir2 := False;
  Repeat
     Mem[Seg(Linebuffer^):ofs(Linebuffer^)+byte1] := thiscode;
     Inc(byte1);
     If byte1>=width Then
     begin
        Move(Linebuffer^, Mem[Seg(Buffer^):
            Ofs(buffer^)+Width*Line],width);
        byte1 := 0;
{ verifica si la imagen es entrelazada }
        if (flags And $40)>0 Then
        begin
          line := line+inctable[pass];


          if line>=depth Then
          begin
             Inc(pass); line := startable[pass];
          end;
        end
        else Inc(line);
  End;
     If u<=0 Then Salir2 := True;
     If Not Salir2 Then
     begin
        Dec(u);  Thiscode := Firstcodestack[u];
     end;
Until Salir2;
  if (nextcode<4096) And (oldcode<>NO_CODE) Then
  begin
     codestack[nextcode] := oldcode;
     lastcodestack[nextcode] := oldtoken;
     inc(nextcode);
     If (nextcode>=codesize2) And (codesize<12) Then
```

```
                begin
                   Inc(codesize);
                   codesize2 := 1 shl codesize;
                end;
             end;
             oldcode := currentcode;
          End;
       Until Salir;
       FreeMem(linebuffer, width);
End;

{ Ajusta la paleta del VGA y el fondo }
PROCEDURE setvgapalette(Var p; n,b : Integer);
Var
   r      : registers;
   i, Aj1 : integer;
Begin
   Aj1 := 0;
   Port[$3C6] := $FF;
   For i := 0 to n-1 Do
   begin
      Port[$3C8] := i;
      Port[$3C9] := Mem[Seg(p):Ofs(p)+Aj1] Shr 2; Inc(Aj1);
      Port[$3C9] := Mem[Seg(p):Ofs(p)+Aj1] Shr 2; Inc(Aj1);
      Port[$3C9] := Mem[Seg(p):Ofs(p)+Aj1] Shr 2; Inc(Aj1);
   End;
   r.ax := $1001;
   r.bh := b;
   intr($10,r);
End;

{ Muestra la imágen en pantalla }
PROCEDURE showpicture(p : Pointer; width,depth,bits,background: Integer;
                      Var palette);
Var
  i,n    : Integer;
Begin
   setvgapalette(palette,1 shl bits,background);
   if (width>SCREENWIDE) Then n:=SCREENWIDE else n:=width;



   For i := 0 To SCREENDEEP-1 Do
   Begin
     Move(Mem[Seg(p^):Ofs(p^)+width*i],
          Mem[Seg(Destino^):Ofs(Destino^)+SCREENWIDE*i], n);
   End;
End;

{ Descompacta un archivo GIF }
PROCEDURE unpackgif(Var fp : File);
Var
  gh       : GIFHEADER;
  iblk     : IMAGEBLOCK;
  p        : Pointer;
  Dato     : Char;
  palette  : Array[1..768] Of Byte;
  b,bits,background,width,depth,flags : Integer;
  c        : byte;
  i        : Word;
  Cad      : String;
Begin
  { Asegurarse que es un archivo GIF }
    BlockRead(fp, gh, sizeof(GIFHEADER));
    If IOResult<>0 Then exit;
    Cad[0] := #6;  Move(gh.sig, Cad[1], 6);
```

```
  If Copy(Cad,1,3)<>'GIF' Then exit;

{ Tomar las caracteristicas de la pantalla }
  bits       := (gh.flags And 7)+1;
  background := gh.background;

{ Tomar el mapa de colores si existe }
  if (gh.flags And $80)>0 Then
  Begin
      i := 3*(1 Shl ((gh.flags And 7)+1));
      BlockRead(fp,palette,i);
      If IOResult<>0 Then exit;
  End;
{ Recorriendo los bloques }
  Repeat
     Dato := #1;
     BlockRead(fp, Dato, 1);
     If (IOResult=0) And (Dato<>#0) Then
     Begin

  { si es un bloque de imagen }
       If Dato=',' Then
       Begin

  { toma el inicio del bloque de la imagen }
          BlockRead(fp, iblk, sizeof(IMAGEBLOCK));
          If IOResult<>0 Then exit;

  { toma las dimensiones de la imagen }
          width := iblk.width;
          depth := iblk.depth;

  { toma el mapa de colores local si existe }
          If (iblk.flags And $80)>0 Then
          Begin
             b     := 3*(1 Shl ((iblk.flags And 7) + 1));
             BlockRead(fp,Palette,b);
             If IOResult<>0 Then exit;
             bits :=(iblk.flags And 7) +1;
          End;

  { toma el código de talla inicial }
          BlockRead(fp, c, 1);
          If IOResult<>0 Then exit;
          flags := iblk.flags;
          If (Width*Depth>65535) Or (MaxAvail<Width*Depth)
             Then exit;
          GetMem(p, Width*Depth);

  { descompacta la imagen }

          unpackimage(fp,p,c,width,depth,flags);

  { muestra la figura }

          showpicture(p,width,depth,bits,background,palette);
          FreeMem(p, Width*Depth);

  { eso es todo }
       End

  { de otro modo es una extensión }
       Else If Dato='!' Then doextension(fp);
       If Dato=#0 Then exit;
     End;
  Until (EOF(fp)) Or (Dato=#1);
End;
```

PROCEDIMIENTO QUE LEE UNA IMAGEN DE DISCO EN FORMATO GIF

```
PROCEDURE Formato.VerGIF(nombreimagen : string; Var Panta_var);
Var Arch : File;
Begin
   Assign(Arch, NombreImagen);
   Reset(Arch,1);
   Destino := Ptr(Seg(Panta_Var),Ofs(Panta_Var));
   If IOResult=0 Then UnpackGIF(Arch);
   Close(Arch);
   If IOResult=0 Then;
End;
```

PONE LA PALETA ENCONTRADA EN EL ARCHIVO .AVI

```
PROCEDURE PonPaletaAVI(Var p);
Var
   r       : registers;
   i, Aj1 : integer;
Begin
   Aj1 := 45;
   Port[$3C6] := $FF;
   For i := 0 To 255 Do
   begin
      Aj1 := i*3;
      Port[$3C8] := i;
{3}      Port[$3C9] := Mem[Seg(p):Ofs(p)+Aj1+2] Shr 2;
{2}      Port[$3C9] := Mem[Seg(p):Ofs(p)+Aj1+1] Shr 2;
{1}      Port[$3C9] := Mem[Seg(p):Ofs(p)+Aj1] Shr 2;
   End;
   r.ax := $1001;
   r.bh := 0;
   intr($10,r);
End;
```

PROCEDIMIENTO QUE LEE UNA IMAGEN DE DISCO EN FORMATO AVI

```
PROCEDURE Formato.VerAVI(nombreimagen : string; Var Panta_var);
Var
   Cad                  : String;
   Arch                 : File;
   AnchoPant, Ancho,
   Largo, LargoPant     : Word;
   inicio,
   i, j, AjImagen       : Longint;
   Tecla2, Tecla        : Char;
   Paleta               : Array[1..769] Of Byte;
   Total, segi, ofsi    : word;
{    Dir1                 : Dir_MCGA;}
   Pant                 : Byte;
Begin
{$I-}
  Pant := LibreXMS(67);
  Assign(Arch,NombreImagen);
  Reset(Arch,1);
  If IOResult<>0 Then exit;
  Seek(Arch,48);
  BlockRead(Arch, Total, 2);
  Seek(Arch,64);
  BlockRead(Arch, Ancho, 2);
  BlockRead(Arch, largo, 2);
  BlockRead(Arch, Largo, 2);
  Seek(Arch, 212);
  If IOResult<>0 Then exit;
  For i := 0 To 255 Do BlockRead(Arch, Paleta[i*3+1], 4);
  Cad := '';
  For i := 1 to 255 Do Cad := Cad+' ';
  Inicio := 2000;
```

```
While (Pos('rec 00db',Cad)=0) And (Inicio<20000) Do
begin
   Seek(Arch,Inicio);
   Blockread(Arch, Cad[1], 255);
   If Pos('r',Cad)>0 Then
   begin
      If Pos('rec 00db',Cad)=0 Then Inc(Inicio,Pos('r',Cad))
        Else Inc(Inicio,Pos('rec 00db',Cad)+11);
   end Else Inc(Inicio,255);
end;
If Inicio>=20000 Then
   mensaje(';;;; Error buscando el inicio de la figura !!!',1)
Else begin
   PonPaletaAVI(paleta);
   If Largo>200 Then LargoPant := 200 Else LargoPant := Largo;
   If Largo>320 Then AnchoPant := 320 Else AnchoPant := Ancho;
   j := 0;  Tecla2:= #75;
   Repeat
      If Tecla2<>#0 Then
      begin
        AjImagen := (largo*ancho+20)*j+Inicio;
        Seek(Arch, AjImagen);
        LeeArchXMS(Arch, XMSMan[Pant], Largo*Ancho);
        AjImagen := 0;
        For i := LargoPant-1 Downto 0 Do
        begin
          LeeXMS(XMSMan[Pant], Mem[$A000:i*320], AnchoPant, AjImagen);
          Inc(AjImagen, Ancho);
        end;
        Str((j+1):5,Cad);
        Letrero(1,23,Cad,128);
      End;
      Tecla := Readkey;  Tecla2 := #0;
      Case Tecla Of
         #0 : Begin
                Tecla2 := Readkey;
                Case Tecla2 Of
                  #77 : If j+1<Total Then Inc(j,1) Else Tecla2 := #0;
                  #75 : If j>0 Then Dec(j,1) Else Tecla2 := #0;
                  #79 : If j<>Total-1 Then j := Total-1 Else Tecla2 := #0;
                  #71 : If j>0 Then j := 0 Else Tecla2 := #0;
                End;
              End;
         #13 : Begin
                AjImagen := (largo*ancho+20)*j+Inicio;
                Seek(Arch, AjImagen);
                LeeArchXMS(Arch, XMSMan[Pant], Largo*Ancho);
                AjImagen := 0;
                For i := LargoPant-1 Downto 0 Do
                begin
                  LeeXMS(XMSMan[Pant], Mem[$A000:i*320]
                    ,AnchoPant, AjImagen);
                  Inc(AjImagen, Ancho);
                end;
                Move(Mem[$A000:0], Panta_Var, 64000);
               End;
      end;
   Until Tecla In [#27,#13];
End;
Close(Arch);
If IOResult<>0 Then;
LiberaXMS(XMSMan[Pant]);
{$I+}
End;

END.
```

LibRat.pas : Librería para el manejo del dispositivo ratón.

```
Unit LibRat;

Interface

Type
  Raton = object
    Procedure IniciaRaton(col, ren : word);
    Procedure Espera2(retardo : longint);
    Function  RangoRaton(xs,ys,xi,yi : word) : boolean;
    Function  RangoRaton2(corx,cory, xs,ys,xi,yi : word) : boolean;
    Function  NumBotonRaton : byte;
    Procedure CursorRaton(modo : boolean);
    Procedure VerRaton;
    Procedure PosRaton(Var boton, col, ren : word);
    Function  Checa_Raton : Boolean;
    Procedure Libera_Boton;
    Procedure LimiteRaton(HorMin, VerMin, HorMax, VerMax : Word);
  end;

Var
    Si_Hay_Raton : Boolean;

Implementation
uses dos,crt;
INDICA SI EL CURSOR SE ENCUENTRA EN UN RANGO DADO
Procedure Raton.IniciaRaton(col, ren : word);
Var
  reg : registers;
Begin
  If Si_Hay_Raton Then
  with reg do
    begin
      reg.ax:=$0004;
      reg.cx:=col*2;
      reg.dx:=ren;
      intr($33,reg);
    end;
End;

ESPERA A QUE SE OPRIMA UNA TECLA, UN BOTON DEL RATON O TRANSCURRA RETARDO
Procedure Raton.Espera2(retardo : longint);
var
  tecla : char;
  salir : boolean;
  boton : word;
  cont  : longint;
Begin
  salir:=false;
  cont:=0;
  while salir=false do
    begin
      inc(cont); if cont>retardo then salir:=true;
      if keypressed then
        begin
          tecla:=readkey;
          salir:=true;
        end;
      if si_hay_raton then
        begin
```

```
          asm
            mov ax,$0003
            int $33
            mov boton,bx
          end;
          if boton<>0 then salir:=true;
        end;
    end;
end;
```

INDICA SI EL RATON SE ENCUENTRA DENTRO DE ESE RANGO

```
Function Raton.RangoRaton(xs,ys,xi,yi : word) : boolean;
var
  reg : registers;
  modo : byte;
  columnas : byte;
begin
  if Si_Hay_raton then
    begin
      reg.ah:=$0F;
      intr($10,reg);
      modo:=reg.al;
      columnas:=reg.ah;
      reg.ax:=$0003;
      intr($33,reg);

      {Modo texto baja Resolucion}
      if ((modo=1) or (modo=0)) and (columnas=40) then
          begin xs:=xs*16-16; xi:=xi*16-16; ys:=ys*8-8; yi:=yi*8-8; end;

      {Modo texto Alta resolucion}
      if (modo In [3,2]) and (columnas=80) then
          begin xs:=xs*8-8; xi:=xi*8-8; ys:=ys*8-8; yi:=yi*8-8; end;

      {Modo CGA Alta resolucion}
      if (modo=6) and (columnas=80) then
          begin xs:=xs-1; xi:=xi-1; ys:=ys-1; yi:=yi-1; end;

      {Modo CGA Baja resolucion}
      if (modo In [4,5,19]) and (columnas=40) then
          begin xs:=xs*2-1; xi:=xi*2-1; ys:=ys-1; yi:=yi-1; end;

      if (reg.cx>=xs) and (reg.cx<=xi) and (reg.dx>=ys) and (reg.dx<=yi)
      then RangoRaton:=true
      else RangoRaton:=false;
    end
  else rangoRaton:=false;
end;
```

INDICA SI CORX Y CORY SE ENCUENTRAN DENTRO DE ESE RANGO

```
Function Raton.RangoRaton2(corx,cory, xs,ys,xi,yi : word) : boolean;
var
  reg : registers;
  modo : byte;
  columnas : byte;
begin
  if Si_Hay_raton then
    begin
      reg.ah:=$0F;
      intr($10,reg);
      modo:=reg.al;
      columnas:=reg.ah;
      reg.ax:=$0003;
      intr($13,reg);
      reg.cx:=corx*2;
      reg.dx:=cory;
```

```
      {Modo texto baja Resolucion}
      if ((modo=1) or (modo=0)) and (columnas=40) then
          begin xs:=xs*16-16; xi:=xi*16-16; ys:=ys*8-8; yi:=yi*8-8; end;

      {Modo texto Alta resolucion}
      if (modo In [3,2]) and (columnas=80) then
          begin xs:=xs*8-8; xi:=xi*8-8; ys:=ys*8-8; yi:=yi*8-8; end;

      {Modo CGA Alta resolucion}
      if (modo=6) and (columnas=80) then
          begin xs:=xs-1; xi:=xi-1; ys:=ys-1; yi:=yi-1; end;

      {Modo CGA Baja resolucion}
      if (modo In [4,5,19]) and (columnas=40) then
          begin xs:=xs*2-1; xi:=xi*2-1; ys:=ys-1; yi:=yi-1; end;

      if (reg.cx>=xs) and (reg.cx<=xi) and (reg.dx>=ys) and (reg.dx<=yi)
      then RangoRaton2:=true
      else RangoRaton2:=false;
    end
  else rangoRaton2:=false;
end;
```

DEVUELVE EL NUMERO DE BOTON OPRIMIDO EN EL RATON

```
function Raton.NumBotonRaton : byte;
var
  reg : registers;
  ok  : word;
begin
  If Si_Hay_Raton Then
  Begin
      reg.ax:=$0003;
      intr($33,reg);
      NumBotonRaton:=reg.bx;
  End
  else NumBotonRaton:=0;
end;
```

ACTIVA Y DESACTIVA EL RATON

```
procedure Raton.CursorRaton(modo : boolean);
Var
  reg : registers;
Begin
  If Si_Hay_Raton Then
  with reg do
    begin
      if modo=true then
        begin
          ax:=$0001;
        end
      else
        begin
          ax:=0002;
        end;
      intr($33,reg);
      ax:=$000f;
      cx:=8;
      dx:=8;
      intr($33,reg);
    end;
end;
```

DESPLIEGA LA POSICION ACTUAL DEL RATON EN VIDEO

```
Procedure Raton.VerRaton;
```

```
var
  reg : registers;
begin
  reg.ax:=$0003;
  intr($33,reg);
  writeln;
  writeln(reg.cx div 8 + 1,'  ',Reg.dx div 8 +1);
end;
```

DEVUELVE EL BOTON PRESIONADO Y LAS COORDENADAS DEL RATON

```
Procedure Raton.PosRaton(Var boton, col, ren : word);
Var
  reg : registers;
Begin
  If Si_Hay_Raton Then
  with reg do
    begin
      ax:=$0003;
      intr($33,reg);
      boton:=bx;
      col:=cx div 2;
      ren:=dx;
    end
  else
    begin
      boton:=0;
      col:=0;
      ren:=0;
    end;
End;
```

VERIFICA LA INSTALACION DEL RATON

```
FUNCTION Raton.Checa_Raton : Boolean;
Var
   r  : Registers;
Begin                              { Si hay ratón y está instalado    }
   With r Do                       { el controlador del ratón se devuelve }
   Begin                           { un valor verdadero para la funcion. }
     AX := 0;
     Intr($33, r);
     If AX=0 Then
        Si_Hay_Raton := False
     Else
        Si_Hay_Raton := True;
     Checa_Raton := Si_Hay_Raton;
   end;
End;
```

ESPERA A QUE NO HAYA NINGUN BOTON PULSADO

```
PROCEDURE Raton.Libera_Boton;
Var
   Boton, Xr, Yr : Word;
Begin
   If Si_Hay_Raton Then
     Repeat
        PosRaton(Boton, Xr, Yr);
     Until Boton=0;
End;
```

ESTABLECE LOS LIMITES DE MOVIMIENTO DEL RATON

```
PROCEDURE Raton.LimiteRaton(HorMin, VerMin, HorMax, VerMax : Word);
Var
   r : Registers;
Begin
   If Si_Hay_Raton Then
   With r do
```

LibSb pas : Libreria para el manejo de la tarjeta Sound Blaster.

```
{$M 30000,0,655000}
UNIT LIBSB;

INTERFACE
{ Procedimientos de Música F.M. }
   Function  Busca_FMDriver : Boolean;
   Function  FM_Version : Word;
   Function  Inicializa_FM : Word;
   Function  Lee_FM(Nombre : String) : Boolean;
   Function  Toca_FM : Word;
   Procedure Deten_FM;
   Procedure Finaliza_FM;
{ Procedimientos de sonido digital }
   Procedure Lee_DrvVoz;
   Function  Inicia_DrvVoz(DirBase, DMA : Word; Var Version : Word) : Byte;
   Procedure Lee_ExtVoc(Nombre : String);
   Procedure Toca_ExtVoc;
   Procedure Lee_Voc(Nombre:String; Var DirMem:Pointer; Var TallaPtr:Word);
   Procedure Toca_Voc(Var DirMem);
   Procedure Termina_Ciclo;
   Procedure Sigue;
   Procedure SalidaSB(Encendida : Boolean);
   Procedure Termina_SBP;
   Procedure Deten_Voc;

Var
   Repite_VOC,
   Repite_FM,
   EstadoFM : Byte;
   Marca_VOC,
   Manip_VOC,
   ST_SBPro    : Word;
   EstadoVoc   : Boolean;

IMPLEMENTATION

{$F+}

Uses Crt, Dos, LibXMS;
Var
   IntFMDrv    : Byte;
   Regs        : Registers;
   TallaMus    : Word;
   DrvVoz,
   SalvaInt1C,
   DirMus      : Pointer;
   Activo1C,
   EstadoMus   : Boolean;
   TallaDrvVoz : LongInt;
```

```
VERIFICA SI EL CONTROLADOR SBFMDRV.COM A SIDO PREVIAMENTE CARGADO EN MEMORIA Y
DEFINE SU POSICION
FUNCTION Busca_FMDriver : Boolean;
Var
   Firma    : String[7];
   Segl,
   Ofsl, i : Word;
Begin
   Firma[0] := #7;
   i := $80-1;
```

```
   Begin
      AX := $0007;     { Limites Horizontales del ratón }
      CX := HorMin;
      DX := HorMax;
      Intr($33, r);
      AX := $0008;     { Limites Verticales del ratón }
      CX := VerMin;
      DX := VerMax;
      Intr($33, r);
   End;
End;

end.
```

```
   Repeat
      Inc(i);
      Segl := MemW[0:i*4+2];
      Ofsl := MemW[0:i*4];
      Move(Mem[Segl:Ofsl+$2], Firma[1], 7);
   Until (Firma='PSQRWVU') Or (i>=$BF);
   If Firma<>'PSQRWVU' Then Busca_FMDriver := False
   Else begin
      IntFMDrv := i;
      Busca_FMDriver := True;
   End;
End;
```

DEVUELVE LA VERSION DEL CONTROLADOR DE FM

```
FUNCTION FM_Version : Word;
Begin
   If IntFMDrv>0 Then
   With Regs Do
   Begin
      BX := 0;
      Intr(IntFMdrv,Regs);
      FM_Version := AX;
   End;
End;
```

INICIALIZA EL CONTROLADOR DE FM

```
FUNCTION Inicializa_FM : Word;
Var
   Error       : Word;
Begin
{ Define la ubicación del byte de estado }
   Error := 1;
   If IntFMDrv>0 Then
   With Regs Do
   Begin
      DX := Seg(EstadoFM);
      AX := Ofs(EstadoFM);
      BX := 1;
      Intr(IntFMdrv,Regs);

{ Reinicia el controlador de FM }
      If EstadoMus Then Deten_FM;

      BX := 0;
      Intr(IntFMdrv,Regs);       Error := AX;
   End;
   Inicializa_FM := Error;
End;
```

LEE UN ARCHIVO CMF EN MEMORIA CONVENCIONAL

```
FUNCTION Lee_FM(Nombre : String) : Boolean;
Var
   Arch  : File;
   i     : Word;
   Talla : LongInt;
Begin
{$I-}
   If IntFMDrv>0 Then
   Begin
      If (DirMus<>Nil) And (TallaMus>0) Then FreeMem(DirMus, TallaMus);
      Assign(Arch, Nombre);
      Reset(Arch,1);
      Talla := FileSize(Arch);
      If Talla>65535 Then i := 65535 Else i := Talla;
      GetMem(DirMus, i);
      BlockRead(Arch, DirMus^, i);
```

```
      Close(Arch);
      TallaMus := i;
    End;
    If IOResult<>0 Then Lee_FM := False Else Lee_FM := True;
{$I+}
End;
```

EMPIEZA A TOCAR EL ARCHIVO DE MUSICA CMF

```
FUNCTION Toca_FM : Word;
Var
  Error,
  Seg1, Ofs1 : Word;
Begin
    Error := 1;
    If IntFMDrv>0 Then
    Begin
      If EstadoMus Then Deten_FM;
      Seg1 := Seg(DirMus^);
      Ofs1 := Ofs(DirMus^);
      With Regs Do
      Begin
  { Define los parámetros de la tabla de elementos }
        CX := MemW[Seg1:Ofs1+$24];
        BX := 2;
        AX := Ofs1+MemW[Seg1:Ofs1+6];
        DX := Seg1;
        Intr(IntFMdrv,Regs);

  { Define la velocidad de dispositivo }
        BX := 4;
        AX := $1234DC Div MemW[Seg1:Ofs1+$0C];
        Intr(IntFMdrv,Regs);

  { Toca el Archivo de música y devuelve un código error }
        DX := Seg1;
        AX := Ofs1+MemW[Seg1:Ofs1+8];
        BX := 6;
        Intr(IntFMdrv,Regs);
        Error := AX;
      END;
      EstadoMus := True;
    End;
    Toca_FM := Error;
End;
```

DETIENE LA MUSICA DE FONDO

```
PROCEDURE Deten_FM;
Begin
    If IntFMDrv>0 Then
    With Regs Do
    Begin
       BX := 7;
       Intr(IntFMdrv,Regs);
       Repite_FM := 0;
    END;
    EstadoMus := False;
End;
```

RESTAURA LAS CONDICIONES EN QUE SE ENCONTRADA LA COMPUTADORA ANTES DE INICIALIZAR LAS VARIABLES PARA EL CONTROLADOR DE FM

```
PROCEDURE Finaliza_FM;
Begin
    If EstadoMus Then Deten_FM;
    FreeMem(DirMus, TallaMus);
    SetIntVec($01C,SalvaInt1C);
End;
```

LEE EL CONTROLADOR DE SONIDO DIGITALIZADO EN MEMORIA CONVENCIONAL

```
PROCEDURE Lee_DrvVoz;
Var
   Arch : File;
Begin
{$I-}
   Assign(Arch, '..\SB\DRV\CT-VOICE.DRV');
   Reset(Arch,1);
   TallaDrvVoz := FileSize(Arch);
   GetMem(DrvVoz, TallaDrvVoz);
   BlockRead(Arch, DrvVoz^, TallaDrvVoz);
   Close(Arch);
   If IOResult<>0 Then;
{$I+}
End;
```

DEVUELVE EL ESTADO ACTUAL DEL SONIDO DIGITALIZADO

```
PROCEDURE Estado;
Var
  Seg1, Ofs1 : Word;
Begin
  Seg1 := Seg(ST_SBPro);
  Ofs1 := Ofs(ST_SBPro);
  ASM
     PUSH AX
     PUSH DX
     MOV ES, Seg1
     MOV DI, Ofs1
     MOV BX, $05
     CALL DrvVoz
     POP DX
     POP AX
  END;
End;
```

INICIALIZA LA TARJETA PARA USAR SONIDOS DIGITALIZADOS

```
FUNCTION Inicia_DrvVoz(DirBase, DMA : Word; Var Version : Word) : Byte;
Var
   Ver, Error : Word;
Begin
 If DrvVoz=Nil Then
 Begin
  Lee_DrvVoz;
  ASM
    PUSH AX
    PUSH DX
    MOV BX, $00    { Obtiene la versión del driver }
    CALL DrvVoz
    MOV Ver, AX

    MOV BX, $01    { Dirección base de E/S }
    MOV AX, DirBase
    CALL DrvVoz

    MOV BX, $02    { Interrupción DMA }
    MOV AX, DMA
    CALL DrvVoz

    MOV BX, $13    { Canal DMA }
    MOV AX, 1
    CALL DrvVoz
```

```
    MOV BX, $03     { Inicializa Driver }
    CALL DrvVoz
    MOV Error, AX

    MOV BX, $14     { Obtiene tipo de tarjeta }
    CALL DrvVoz

    POP DX
    POP AX
  END;
  Estado;
  Inicia_DrvVoz := Error;  { 0 = No error }
                           { 1 = Versión de Driver incorrecta }
                           { 2 = Falla de E/S }
                           { 3 = Falla en interrupción DMA }
  Version := Ver;
 End;
End;
```

LEE UN ARCHIVO DE VOZ (*.VOC) EN MEMORIA EXTENDIDA

```
PROCEDURE Lee_ExtVoc(Nombre : String);
Var
   Arch  : File;
   P     : Pointer;
   i     : Word;
   T1, T2,
   Talla : LongInt;
Begin
{$I-}
   If Manip_VOC>0 Then LiberaXMS(Manip_VOC);
   GetMem(p, 40000);
   Assign(Arch, Nombre);
   Reset(Arch,1);
   Talla := FileSize(Arch);
   i := (Talla Div 1024)+1;
   If ReservaXMS(i, Manip_VOC)=0 Then;
   i := 0;  T2 := 0;
   Repeat
     T1 := i;
     EscribeXMS(Manip_VOC, P^, T1, T2);
     T2 := T2+T1;
     BlockRead(Arch, P^, 40000, i);
     If (i<40000) And (Odd (i)) Then Inc(i);
   Until i=0;
   Close(Arch);
   If IOResult<>0 Then;
   FreeMem(p, 40000);
{$I+}
End;
```

REPRODUCE UN SONIDO DIGITALIZADO QUE SE ENCUENTRE EN MEMORIA EXTENDIDA

```
PROCEDURE Toca_ExtVoc;
Var
   Act     : LongInt;
   IncVoc,
   Marca2,
   Firma   : Word;
   Cad     : String[3];
Begin
  If EstadoVOC Then Deten_VOC;
  Act := 26;
  If Marca_VOC>0 Then
  begin
    repeat
      LeeXMS(Manip_VOC, Firma, 2, Act);
```

```
      LeeXMS(Manip_VOC, Marca2, 2, Act+4);
      LeeXMS(Manip_VOC, IncVoc, 2, Act+7);
      If Marca_VOC<>Marca2 Then Inc(Act, IncVoc+10);
    until (Marca_VOC=Marca2) Or (Firma<>$0204);
    If Marca_VOC<>Marca2 Then
    begin
      Marca_VOC := 0; Act := 26;
    end;
  end;
  If Marca_VOC>0 Then
  begin
    Move(Act, Cad, 4);
    Move(Cad, IncVoc, 2);
    Move(Cad[2], Marca2, 2);
    ASM
      PUSH AX
      PUSH DX
      MOV BX, $0E
      MOV DX, Manip_VOC
      MOV DI, Marca2
      MOV SI, IncVoc
      CALL DrvVoz
      POP DX
      POP AX
    END;
  End;
  EstadoVOC := True;
End;
```

LEE UN ARCHIVO DE VOZ (*.VOC) EN MEMORIA CONVENCIONAL

```
PROCEDURE Lee_Voc(Nombre:String; Var DirMem:Pointer; Var TallaPtr:Word);
Var
   Arch  : File;
   i     : Word;
   Talla : LongInt;
Begin
{$I-}
   Assign(Arch, Nombre);
   Reset(Arch,1);
   Talla := FileSize(Arch);
   If Talla>65535 Then i := 65535 Else i := Talla;
   GetMem(DirMem, i);
   BlockRead(Arch, DirMem^, i);
   Close(Arch);
   If IOResult<>0 Then;
   TallaPtr := i;
{$I+}
End;
```

REPRODUCE UN SONIDO DIGITALIZADO QUE SE ENCUENTRE EN LA MEMORIA CONVENCIONAL

```
PROCEDURE Toca_Voc(Var DirMem);
Var
   Seg1, Ofs1 : Word;
Begin
   Seg1 := Seg(DirMem);
   Ofs1 := Ofs(DirMem)+26;
   ASM
      PUSH AX
      PUSH DX
      MOV ES, Seg1
      MOV DI, Ofs1
      MOV BX, $06
      CALL DrvVoz
```

```
      POP DX
      POP AX
   END;
   EstadoVOC := True;
End;

TERMINA UN CICLO DEFINIDO POR EL ARCHIVO DE VOZ
PROCEDURE Termina_Ciclo;
Begin
  ASM
     PUSH AX
     PUSH DX
     MOV BX, $0C
     MOV AX, $0001
     CALL DrvVoz
     POP DX
     POP AX
  END;
End;

CONTINUA TOCANDO UN SONIDO DIGITALIZADO QUE FUE PUESTO EN PAUSA ANTERIORMENTE
PROCEDURE Sigue;
Begin
  ASM
     PUSH AX
     PUSH DX
     MOV BX, $0B
     CALL DrvVoz
     POP DX
     POP AX
  END;
End;

CONECTA/DESCONECTA LA SALIDA DE AUDIO DE LA TARJETA SOUNDBLASTER
PROCEDURE SalidaSB(Encendida : Boolean);
Var
   Enc : Byte;
Begin
  If Encendida Then Enc := 1 Else Enc := 0;
  ASM
     MOV AL, Enc
     MOV BX, $04
     CALL DrvVoz
  END;
End;

MANDA UNA SEÑAL A LA TARJETA DE SONIDO INDICANDOLE QUE YA NO SE VAN A REALIZAR
OPERACIONES DE ENTRADA/SALIDA
PROCEDURE Termina_SBP;
Begin
  ASM
     PUSH AX
     PUSH DX
     MOV BX, $09
     CALL DrvVoz
     POP DX
     POP AX
  END;
  SalidaSB(False);
  If EstadoVOC Then Deten_VOC;
  If Manip_VOC>0 Then LiberaXMS(Manip_VOC);
End;

DETIENE LA REPRODUCCIÓN DE UN SONIDO DIGITALIZADO
PROCEDURE Deten_Voc;
```

```
Var
   Enc : Byte;
Begin
  ASM
     MOV BX, $08
     CALL DrvVoz
  END;
  Repite_VOC := 0;
  EstadoVOC := False;
End;
```

PROCEDIMIENTO RESIDENTE QUE CONTROLA LA REPRODUCCION DE LOS SONIDOS DIGITALIZADOS Y LA MUSICA DE FONDO

```
{$F+,S-,W-}
PROCEDURE IntTiempo; Interrupt;
Var
  Temp : Byte;
BEGIN
   ASM
     PUSHF
     CALL SalvaInt1C
   END;
   If (Not Activo1C) And (EstadoMus Or EstadoVOC) Then
   Begin
      Activo1C := True;
      If EstadoFM=0 Then
         If Repite_FM=0 Then Deten_FM
          Else begin
            Temp := Repite_FM;
            Toca_FM;
            Repite_FM := Temp;
            If Repite_FM<99 Then Dec(Repite_FM);
          end;

      If Marca_VOC = 0 Then
      begin
         EstadoVOC  := False;
         Repite_VOC := 0;
      end
      Else
       If ST_SBPro<>Marca_VOC Then
       begin
         If (Repite_VOC=0) Or (Marca_VOC=0) Then
         begin
           Deten_Voc;
           If Marca_VOC>0 Then
         end
         Else begin
           Temp := Repite_Voc;
           Toca_ExtVoc;
           Repite_Voc := Temp;
           If Repite_Voc<99 Then Dec(Repite_Voc);
         end;
       end;
      Activo1C := False;
   End;
END;
```

RUTINA DE INICIALIZACION

```
Var Version : Word;
Begin
   DrvVoz := Nil;
   If Inicia_DrvVoz($220, 7, Version)<>0 Then;
   Marca_VOC := 0;
   Manip_VOC := 0;
```

```
    ST_SBPro  := 0;
    EstadoFM  := $FF;
    IntFMDrv  := 0;
    TallaMus  := 0;
    DirMus    := Nil;
    EstadoMus := False;
    EstadoVoc := False;
    Repite_FM := 0;
    GetIntVec($1C,SalvaInt1C);
    SetIntVec($1C,Addr(IntTiempo));
End.
```

LibTxt.pas Libreria para el manejo de mensajes de texto

```
Unit LibTxt;

INTERFACE

Var
   Manip_Frases,
   Manip_Objetos  : Byte;
   Retardo_Frases : Word;
   Numero_Frase   : Word;
   Color_Frase    : Byte;

PROCEDURE Lee_Frases(Nombre : String; Var Manipl : Byte);
PROCEDURE Convierte_Arch(Nombre : String);
PROCEDURE Pon_Frase(Numero, Color : Byte; donde : boolean);
FUNCTION  Cad_Objetos(Numero : Byte) : String;

IMPLEMENTATION

Uses libgra, Crt, Dos, LibXMS, LibPan, LibPal;
```

DESPLIEGA EL NOMBRE UN OBJETO DADO

```
FUNCTION  Cad_Objetos(Numero : Byte) : String;
Var
  Indice : Word;
  Cad    : String;
  Ancho  : Word;
  x      : array[1..50] of char;
Begin
  Cad := '';
  If (Manip_Objetos>0) And (Numero>0) Then
  begin
    Indice := 0;
    LeeXMS(XMSMan[Manip_Objetos], Indice, 2, (Numero-1)*2);
    If Indice<>65535 Then
    begin
      LeeXMS(XMSMan[Manip_Objetos], x, 50, Indice);
      Inc(Indice,2);
      LeeXMS(XMSMan[Manip_Objetos], Ancho, 2, Indice);
      Ancho := Lo(Ancho)+2;
      LeeXMS(XMSMan[Manip_Objetos], Cad, Ancho, Indice);
    End;
  end;
  Cad_Objetos := Cad;
End;
```

LEE EN MEMORIA EXTENDIDA EL ARCHIVO QUE CONTIENE TODAS LAS FRASES O NOMBRES DE OBJETOS UTILIZADOS POR EL PROGRAMA

```
PROCEDURE Lee_Frases(Nombre : String; Var Manipl : Byte);
Var
   Arch   : File;
   p      : Pointer;
   SubTot,
   Leer,
   Leidos,
   Talla,
   Act,
   Indice : Word;
   Marca  : Word;
Begin
```

```
{$I-}
  GetMem(p,10000);
  Assign(Arch, Nombre);
  Reset(Arch,1);
  manip1 := LibreXMS(FileSize(Arch) Div 1024+2);
  Indice := 65535;
  For Act := 0 to 499 Do
     EscribeXMS(XMSMan[manip1], Indice, 2, Act*2);
  Indice := 1000;
  Act    := 0;
  BlockRead(Arch, Marca, 2);
  While Not EOF(Arch) Do
  Begin
    BlockRead(Arch, Talla, 2);
    Dec(Talla,2);
    EscribeXMS(XMSMan[manip1], Talla, 2, Indice);
    Dec(Talla,2);
    SubTot := 0;
    Repeat
      If SubTot+10000>Talla Then Leer := Talla-SubTot Else Leer := 10000;
      BlockRead(Arch, p^, Leer, leidos);
      EscribeXMS(XMSMan[manip1], p^, leidos, Indice+SubTot+2);
      Inc(SubTot, leidos);
    Until Leer<=10000;
    EscribeXMS(XMSMan[manip1], Indice, 2, Act);
    Inc(Act,2);
    Inc(Indice, Talla+2);
    BlockRead(Arch, Marca, 2);
  end;
  Close(Arch);
  If IOResult<>0 Then;
  FreeMem(p,10000);
{$I+}
End;
```

CONVIERTE UN ARCHIVO DE TEXTO AL FORMATO BINARIO UTILIZADO POR EL PROGRAMA

```
PROCEDURE Convierte_Arch(Nombre : String);
Var
   Arch    : Text;
   Arch2   : File;
   Cad     : string;
   NFrase,
   Ini,
   SubTotal,
   Act, Total : Word;
   code    : Integer;
   p       : Pointer;
   AlmacenFrases : Pointer;
   TallaFrases   : Word;

  Procedure Inserta_Frase(Var Dato; Var Act : Word; NumTot : Byte);
  Var
    S1, O1 : Word;
  Begin
    S1 := Seg(AlmacenFrases^);
    O1 := Ofs(AlmacenFrases^);
    Move(Dato, Mem[S1:O1+Act], NumTot);
    Inc(Act, NumTot);
  end;

Begin
  TallaFrases   := 0;
  AlmacenFrases := Nil;
  clrscr;
  GetMem(AlmacenFrases, 64000);
  Act := 0;     Total := 0;
  Ini := 2;
```

```
  Assign(Arch,Nombre+'.Txt');
{$I-}
  Reset(Arch);
  ReadLn(Arch,Cad);
  While Not EOF(Arch) Do
  Begin
     If Cad='{' Then
     begin
        ReadLn(Arch,Cad);
        If IOResult=0 Then
        begin
          WriteLn('frase: ',Cad);
          Val(Cad, NFrase, code);
          Inserta_Frase(NFrase, Act, 2);
          Inserta_Frase(NFrase, Act, 2);
          While (Cad<>'}') And (Not EOF(Arch)) Do
          begin
            Readln(Arch, Cad);
            While (Cad<>'') And (Cad[Length(Cad)]=' ')
              Do Cad := Copy(Cad,1,Length(Cad)-1);
            If (IOResult=0) And (Cad<>'}') Then
            begin
               WriteLn('>>>  '+Cad);
               Inserta_Frase(Cad, Act, Ord(Cad[0])+1);
            end;
          end;
          SubTotal := Act-Total;
          If Odd(Subtotal) Then
          begin
            Inc(Subtotal);
            Cad[0] := #0;
            Inserta_Frase(Cad, Act, 1);
          end;
          Total := Act;
          Inserta_Frase(Subtotal, Ini, 2);
          Ini := Act+2;
        End;
     end;
     ReadLn(Arch,Cad);
  End;
  Close(Arch);
  GetMem(p, Act);
  Move(AlmacenFrases^, p^, Act);
  FreeMem(AlmacenFrases, 64000);
  AlmacenFrases := p;
  TallaFrases := Act;
  Assign(Arch2, Nombre+'.FRA');
  ReWrite(Arch2, 1);
  BlockWrite(Arch2, AlmacenFrases^, Act);
  FreeMem(AlmacenFrases, TallaFrases);
  Close(Arch2);
{$I+}
End;
```

PONE UNA FRASE EN LA PANTALLA

```
PROCEDURE Pon_Frase(Numero, Color : Byte; donde : boolean);
Var
   Indice,
   Largo,
   S1, O1,
   i, Dir1 : Word;
   Cad     : String;
   Ancho,
   Linea   : Byte;
   p       : pointer;
Begin
   If (Manip_Frases>0) And (Numero>0) Then
```

```
  begin
    Indice := 0;
    Largo := 0;
    LeeXMS(XMSMan[Manip_Frases], Indice, 2, (Numero-1)*2);
    If Indice<>65535 Then
    begin
      LeeXMS(XMSMan[Manip_Frases], Largo, 2, Indice);
      GetMem(p, Largo);
      S1 := Seg(p^);
      O1 := Ofs(p^);
      MemW[S1:O1] := 0;
      MemW[S1:O1+2] := 1;
      LeeXMS(XMSMan[Manip_Frases], p^, largo, Indice);
      if donde then DestinoLet := Pantemporal
      else DestinoLet := Ptr($A000,0);

      Largo := 0;
      Ancho := 0;
      Linea := 0;
      i := 0;
      While i+3<MemW[S1:O1] Do
      begin
        Move(Mem[S1:O1+i+2], Cad, Mem[S1:O1+i+2]+1);
        If Cad<>'' Then
        begin
          Ancho := Length(Cad) Div 2;
          If (Ancho+1)*6>Largo Then Largo := (Ancho+1)*6;
          Inc(Linea);
        end;
        Inc(i,Mem[S1:O1+i+2]+1);
      End;

      for i := 20 to (linea+3)*8+3 do
        begin
        if donde then
        LeeXMS(XMSMan[30], Mem[Seg(Pantemporal^):Ofs(Pantemporal^)+
                               i*320+165-Largo],
               Largo*2, i*320+165-Largo)
        else
        LeeXMS(XMSMan[30], Mem[$A000:i*320+165-Largo],
               Largo*2, i*320+165-Largo);
        end;

      Largo := 0;
      Ancho := 0;
      Linea := 0;
      i := 0;


      While i+3<MemW[S1:O1] Do
      begin
        Move(Mem[S1:O1+i+2], Cad, Mem[S1:O1+i+2]+1);
        Ancho := Length(Cad) Div 2;
        Letrero2(27-Ancho, Linea+3, Cad, Color,1);
        Inc(Linea);
        Inc(i,Mem[S1:O1+i+2]+1);
      End;
      FreeMem(p, MemW[S1:O1]);
      DestinoLet := Ptr($A000,0);
    end;
  end;
End;
```

RUTINA DE INICIALIZACION

```
Begin
   Manip_Frases   := 0;
   Manip_Objetos  := 0;
```

```
    Retardo_Frases := 0;
    Numero_Frase   := 0;
    Color_Frase    := 255;
end.
```

LibUti.pas : Librería de rutinas generales.

```
UNIT LIBUTI;

INTERFACE
USES Crt;

const
  Max = 30;

type
  Registro = Record
     pos    : Word;
     numero : byte;
     largo  : word;
  End;
  Lista = array[1..Max] of Registro;

Procedure Cualtecla;
Procedure Bocina(valor : word);
Procedure Espera;
Procedure BufferTeclado;
Procedure Cursor(Valor : boolean);
Function  ceros(cadena : string; tamano : byte; tipo : byte) : string;
Function  NAC(numero : longint) : string;
Function  Espacios(cadena : string; tamano : byte; tipo : byte) : string;
Function  Si_Parametro(cadena : string) : boolean;
Function  signo(a,b : integer): integer;
Procedure Ordena(var A: Lista; Lo, Hi: Integer);

IMPLEMENTATION
```

INDICA EL VALOR DE LA TECLA OPRIMIDA

```
Procedure Cualtecla;
var
  tecla : char;
  cont : byte;
begin
  tecla:=#0;
  clrscr;
  while tecla<>#27 do
    begin
      tecla:=readkey;
      write(tecla);
      for cont:=0 to 255 do if tecla=chr(cont) then writeln(cont);
    end;
end;
```

GENERA UN SONIDO EN LA BOCINA DE LA COMPUTADORA

```
Procedure Bocina(valor : word);
var
  f: word;
 begin
   port[67]:=182;
   F := 1193180 Div valor;
   port[66]:=lo(f);
   port[66]:=hi(f);
   port[97]:=port[97] or 3;
   delay(5);
   port[97]:=port[97] and 253;
 end;
```

ESPERA A QUE SE OPRIMA UNA TECLA

```
Procedure Espera;
var
  tecla : char;
  salir : boolean;
  boton : word;
  ok    : word;
Begin
  salir:=false;
  ok:=1;
  asm
    mov ax,$0000
    int $33
    mov ok, ax
  end;
  while salir=false do
    begin
      if keypressed then
        begin
          tecla:=readkey;
          salir:=true;
        end;
      if ok<>0 then
        begin
          asm
            mov ax,$0003
            int $33
            mov boton,bx
          end;

          if boton<>0 then salir:=true;
        end;
    end;
end;
```

LIMPIA EL BUFFER DEL TECLADO

```
Procedure BufferTeclado;
begin
  memw[$0000:$041C]:=memw[$0000:$041A];
end;
```

ACTIVA Y DESACTIVA EL CURSOR

```
Procedure Cursor(Valor : boolean);
begin
  if valor=true then
    Begin
      asm
        mov ah,$01
        mov ch,$06
        mov cl,$07
        int $10
      end;
    end
  else
    Begin
      asm
        mov ah,$01
        mov ch,$20
        int $10
      end;
    end;
end;
```

AÑADE CEROS A LA IZQUIERDA O DERECHA DE LA CADENA

```
Function ceros(cadena : string; tamano : byte; tipo : byte) : string;
```

```
tipo
  1 - izquierda
  2 - derecha
Var
  cont1 : byte;
  largo : byte;
  aux   : string;
Begin
  aux:='';
  largo:=length(cadena);
  if largo<=tamano then
    begin
      for cont1 := 1 to tamano-largo do aux:=aux+'0';
      if tipo = 1 then ceros:=aux+cadena;
      if tipo = 2 then ceros:=cadena+aux;
    end
  else
    begin
      aux:='';
      for cont1 := 1 to tamano do aux:=aux+cadena[cont1];
      ceros:=aux;
    end;
End;
```

CONVIERTE UN NUMERO EN CADENA

```
Function NAC(numero : longint) : string;
var
  texto : string;
Begin
  str(numero,texto);
  nac:=texto;
End;
```

INSERTA ESPACIOS EN UNA CADENA

```
Function espacios(cadena : string; tamano : byte; tipo : byte) : string;
{tipo
1 - izquierda
2 - derecha}
Var
  cont1 : byte;
  largo : byte;
  aux   : string;
Begin
  aux:='';
  largo:=length(cadena);
  if largo<=tamano then
    begin
      for cont1 := 1 to tamano-largo do aux:=aux+' ';
      if tipo = 1 then espacios:=aux+cadena;
      if tipo = 2 then espacios:=cadena+aux;
    end
  else
    begin
      aux:='';
      for cont1 := 1 to tamano do aux:=aux+cadena[cont1];
      espacios:=aux;
    end;
End;
```

INDICA QUE SI EXISTE EL PARAMETRO

```
Function Si_parametro(cadena : string) : boolean;
Var
  cont : byte;
Begin
```

```
  Si_parametro:=false;
  for cont := 1 to 20 do
    begin
      if paramstr(cont)=cadena then si_parametro:=true;
    end;
End;
```

INDICA CUAL NUMERO ES MAYOR ENTRE A Y B

```
Function  signo(a,b : integer): integer;
Begin
  if (a=b)  then signo:=0;
  if (a>b)  then signo:=1;
  if (a<b)  then signo:=-1;
end;
```

ORDENA UN ARREGLO DE TIPO LISTA

```
procedure Ordena(var A: Lista; Lo, Hi: Integer);
procedure RapOrd(Izq, Der: Integer);
var
  i, j     : Integer;
  x, y     : Registro;
  Salir    : Boolean;
begin
  i := Izq;
  j := Der;
  x := a[(Izq+Der) DIV 2];
  repeat
    Salir := False;
    Repeat
       If a[i].pos < x.pos Then Inc(i) Else Salir := True;
    Until Salir;
    Salir := False;
    Repeat
       If x.pos < a[j].pos Then Dec(j) Else Salir := True;
    Until Salir;
    if i <= j then
    begin
      y := a[i];  a[i] := a[j];  a[j] := y;
      i := i + 1; j := j - 1;
    end;
  until i > j;
  if Izq < j then RapOrd(Izq, j);
  if i < Der then RapOrd(i, Der);
end;

begin RapOrd(Lo,Hi); end;
END.
```

LibXms.pas : Libreria para el manejo memoria extendida.

```
{$A+}
UNIT LIBXMS;

INTERFACE
   FUNCTION  ObtenXMS : Boolean;
   FUNCTION  ReservaXMS(Talla : Word; Var Handle : Word) : Byte;
                        { Talla en KBytes }
   FUNCTION  LiberaXMS(Handle : Word) : Byte;
   PROCEDURE EscribeXMS(Handle : Word; Var Memoria; Tam, Inicio: LongInt);
   PROCEDURE LeeXMS(Handle : Word; Var Memoria; Tam, { El tamaño debe ser par}
                            Inicio: LongInt);
   PROCEDURE InfoXMS;
```

```
   PROCEDURE InfoBloqueXMS(Handle : Word);
   FUNCTION  LibreXMS(Talla: Word) : Byte;
   PROCEDURE LeeArchXMS(Var Arch : File; Man1 : Word; Total : LongInt);
   PROCEDURE LiberaXMSMan;

Var
   XMSMan : array[1..30] of word;

IMPLEMENTATION

TYPE
  MueveXMS = Record
    Talla       : LongInt;
    ManipOrig   : Word;
    Origen      : LongInt;
    ManipDest   : Word;
    Destino     : LongInt;
  End;

Var
   DirXMS : Pointer;
```

MANDA UN MENSAJE ESPECIFICANDO EL TIPO DE ERROR REPORTADO POR EL CONTROLADOR XMS, CAMBIA A MODO TEXTO Y DETIENE EL PROGRAMA

```
PROCEDURE ReportaError(Origen : String; Error : Byte; M : MueveXMS);
Var
   i : Byte;
Begin
   ASM
      MOV AX, $0003
      INT $10
   END;
   WriteLn('Error de memoria extendida #', Error);
   Write(' Descripción: ');
   Case Error of
     $80 : Write('Función no implementada');
     $81 : Write('Dispositivo VDISK presente');
     $82 : Write('Ocurrió un error A20');
     $A3 : Write('Manipulador origen no válido');
     $A4 : Write('Ajuste origen no válido');
     $A5 : Write('Manipulador destino no válido');
     $A6 : Write('Ajuste destino no válido');
     $A7 : Write('Longitud inválida');
     $A8 : Write('Movimiento invalido. Sobreescritura');
     $A9 : Write('Error de paridad');
     Else Write('Error desconocido');
   end;
   WriteLn('  Reportado por: ',Origen);
   With M Do
     WriteLn('Talla: ',Talla,'  Manip.Origen: ',ManipOrig,'  Origen: ',Origen,
             '  Manip.Destino: ',ManipDest,'  Destino: ',Destino);
   For i := 1 to 30 do
     If XMSMan[i]>0 Then Liberaxms(xmsman[i]);
   halt;
End;
```

DETERMINA SI ES POSIBLE UTILIZAR LA MEMORIA XMS, SI ES ASI, INICIALIZA LAS VARIABLES NECESARIAS

```
FUNCTION ObtenXMS : Boolean;
Var
   Seg1,
   Ofs1 : Word;
   i, j : Byte;
   p    : Pointer;
   cont : byte;
Begin
```

```
   ASM
      MOV AX, $4300
      INT $2F
      MOV I, AL
   END;
   DirXMS := Nil;
   IF I=$80 Then
   Begin
      ASM
        MOV AX, $4310
        INT $2F
        MOV Seg1, ES
        MOV Ofs1, BX
      END;
      DirXMS := Ptr(Seg1,Ofs1);
   End;
   If DirXMS<>Nil Then ObtenXMS := True Else ObtenXMS := False;
   for cont:=1 to 30 do xmsman[cont]:=0;
End;
```

RESERVA UN BLOQUE DE MEMORIA XMS

```
FUNCTION ReservaXMS(Talla : Word; Var Handle : Word) : Byte;
                    { Talla en KBytes }
Var
   Error : Byte;
   Manip,
   Exito : Word;
Begin
  If DirXMS<>Nil Then
  begin
   ASM
      MOV AX, $0900;
      MOV DX, Talla;
      CALL DirXMS
      MOV Manip, DX;
      MOV Error, BL
      MOV Exito, AX
   END;
   Handle := Manip;
   If Exito<>0 Then Error := 0;
   ReservaXMS := Error;   { BL = Código de Error
                            080h, función no implementada
                            081h, VDISK detectado
                            0A0h, toda la memoria extendida está ocupada
                            0A1h, todos los manipuladores están en uso }
  End Else ReservaXMS := $80;
End;
```

LIBERA UN BLOQUE DE MEMORIA XMS

```
FUNCTION LiberaXMS(Handle : Word) : Byte;
Var
   Error : Byte;
   Exito : Word;
Begin
  If DirXMS<>Nil Then
  begin
   ASM
      MOV AX, $0A00;
      MOV DX, Handle;
      CALL DirXMS
      MOV Error, BL
      MOV Exito, AX
   END;
   If Exito<>0 Then Error := 0;
   LiberaXMS := Error;   { BL = Código de Error
                           080h, función no implementada
```

```
                                  081h, VDISK detectado
                                  0A0h, toda la memoria extendida está ocupada
                                  0A1h, todos los manipuladores están en uso }
  End Else LiberaXMS := $80;
End;
```

TRANSFIERE UN BLOQUE DE MEMORIA CONVENCIONAL A MEMORIA XMS

```
PROCEDURE EscribeXMS(Handle : Word; Var Memoria; Tam, Inicio: LongInt);
Var
  MXMS       : MueveXMS;
  p,
  DirXMS2    : Pointer;
  Ok,
  Seg1, Ofs1 : Word;
  Error      : Byte;
Begin
  If DirXMS<>Nil Then
  Begin
    p := Addr(Memoria);
    If Odd(Tam) Then Dec(Tam);
    With MXMS Do
    Begin
       Talla     := Tam;            { Número de bytes a transferir  }
       ManipOrig := 0;              { Desde la memoria convencional }
       Move(p,Origen,4);            { Dirección de la memoria convencional }
       ManipDest := Handle;         { Hacia la memoria XMS }
       Destino   := Inicio;         { Dirección de la memoria XMS }
    End;
    Seg1   := Seg(MXMS);     Ofs1 := Ofs(MXMS);
    DirXMS2 := DirXMS;
    ASM
      PUSH BP
      PUSH DS
      PUSH SI
      MOV AX, $0B00      { Mueve datos de Memoria convencional a XMS }
      MOV SI, Ofs1
      MOV DS, Seg1
      CALL DirXMS2
      MOV Ok, AX
      MOV Error, BL
      POP SI
      POP DS
      POP BP
    END;
    If Ok=0 Then ReportaError('EscribeXMS',Error,MXMS);
  End;
End;
```

TRANFIERE UN BLOQUE DE MEMORIA XMS A MEMORIA CONVENCIONAL

```
PROCEDURE LeeXMS(Handle : Word; Var Memoria; Tam, Inicio: LongInt);
Var
  MXMS       : MueveXMS;
  p,
  DirXMS2    : Pointer;
  Ok,
  S1, O1,
  Seg1, Ofs1 : Word;
  Error      : Byte;
Begin
  If DirXMS<>Nil Then
  Begin
    p := @Memoria;
    S1 := Seg(Memoria);
    O1 := Ofs(Memoria);
    If Odd(Tam) Then Dec(Tam);
    With MXMS Do
```

```
    Begin
      Talla     := Tam;              { Número de bytes a transferir  }
      ManipOrig := Handle;           { Desde la memoria XMS }
      Origen    := Inicio;           { Dirección de la memoria XMS }
      ManipDest := 0;                { Hacia la memoria convencional }
      Move(p,Destino,4);             { Dirección de la memoria convencional }
    End;
    Seg1  := Seg(MXMS);     Ofs1 := Ofs(MXMS);
    DirXMS2 := DirXMS;
    ASM
      PUSH BP
      PUSH DS
      PUSH SI
      MOV AX, $0B00      { Mueve datos de Memoria XMS a convencional}
      MOV SI, Ofs1
      MOV DS, Seg1
      CALL DirXMS2
      MOV Ok, AX
      MOV Error, BL
      POP SI
      POP DS
      POP BP
    END;
    If Ok=0 Then ReportaError('LeeXMS',Error,MXMS);
  End;
End;

{PROPORCIONA UN REPORTE DEL ESTADO ACTUAL DE LA MEMORIA XMS}
PROCEDURE InfoXMS;
Var
   HMA,
   Version, Revision : Word;
   Cad : String;
Begin
  If DirXMS<>Nil Then
  Begin
    ASM
      MOV AX,0
      CALL DirXMS
      MOV Version, AX
      MOV Revision, BX
      MOV HMA, DX
    END;
    Str(Lo(Version),Cad);
    If Cad[0]=#1 Then Cad := '0'+Cad;
    WriteLn('XMS Versión : ', Hi(Version),'.',Cad);
    Str(Lo(Revision),Cad);
    If Cad[0]=#1 Then Cad := '0'+Cad;
    WriteLn('XMS Revisión: ', Hi(Revision),'.',Cad);
    Write('HMA ');
    If HMA=0 Then WriteLn('NO EXISTE') Else WriteLn('EXISTE');
    ASM
      MOV AX, $0800
      CALL DirXMS
      MOV Version, AX
      MOV Revision, DX
      MOV HMA, BX
    END;
    Case Lo(HMA) Of
      0 : Begin
            WriteLn('Talla del bloque mayor disponible de memoria XMS: ',
                    Version, ' Kbytes');
            WriteLn('Total de memoria XMS disponible: ', Revision,' Kbytes');
          End;
      $80 : WriteLn('Función no implementada.');
      $81 : WriteLn('Dispositivo VDISK detectado.');
      $A2 : WriteLn('Manipulador Inválido.');
```

```
      Else
        WriteLn('Error desconocido.');
      End;
  End;
End;
```

PROPORCIONA INFORMACION REFERENTE A UN MANEJADOR DADO

```
PROCEDURE InfoBloqueXMS(Handle : Word);
Var
   Exito,
   Talla,
   Bloque : Word;
Begin
  ASM
    MOV AX, $0E00
    MOV DX, Handle
    CALL DirXMS
    MOV Exito, AX
    MOV Talla, DX
    MOV Bloque, BX
  END;
  If Exito=0 Then
  Begin
     Write('Error: ');
     Case Bloque Of
       $80 : WriteLn('Función no implementada.');
       $81 : WriteLn('Dispositivo VDISK detectado.');
       $A2 : WriteLn('Manipulador Inválido.');
     Else
       WriteLn('Error desconocido.');
     End;
  End
  Else begin
     WriteLn('Bloques con llave: ', Hi(Bloque));
     WriteLn('Número de manipuladores EMB disponibles: ', Lo(Bloque));
     WriteLn('Talla del bloque ',Handle,' = ',Talla,' KBytes.');
  end;
End;
```

BUSCA UN LUGAR LIBRE DONDE SE PUEDA RESERVAR UN BLOQUE DE MEMORIA XMS

```
FUNCTION LibreXMS(Talla: Word) : Byte;
Var
  i : Byte;
Begin
   i := 1;
   While (i<31) And (XMSMan[i]<>0) Do Inc(i);
   If i<31 Then
      ReservaXMS(Talla, XMSMan[i]);
   LibreXMS := i;
End;
```

LEE EL CONTENIDO DE UN ARCHIVO EN MEMORIA XMS

```
PROCEDURE LeeArchXMS(Var Arch : File; Man1 : Word; Total : LongInt);
Var
   Almacen : Pointer;
   Leidos  : Word;
   Leer,
   Inicio  : LongInt;

Begin
{$I-}
   GetMem(Almacen, 25000);
   Inicio := 0; Leer := 0;
   Repeat
      If Inicio+25000<=Total Then Leer := 25000 Else Leer := Total-Inicio;
      BlockRead(Arch, Almacen^, Leer, Leidos);
```

```
      EscribeXMS(Man1, Almacen^, Leidos, Inicio);
      Inc(Inicio, Leidos);
      If IOResult=0 Then;
   Until (EOF(Arch)) Or (Inicio>=Total);
   FreeMem(Almacen, 25000);
{$I+}
End;
```

LIBERA TODA LA MEMORIA XMS UTILIZADA POR EL PROGRAMA

```
PROCEDURE LiberaXMSMan;
Var
   i : Byte;
Begin
  For i := 1 To 30 Do
    If XMSMan[i]>0 Then LiberaXMS(XMSMan[i]);
End;
```

RUTINA DE INICIALIZACION

```
Begin
   DirXMS := Nil;  ObtenXMS;
End.
```

BIBLIOGRAFIA

- Curran, S.; Curnow, R.
  JUEGOS, IMAGENES Y SONIDOS
  Original (Londres, Frances Lincon Limited, 1983)
  Traducción (Barcelona, España; Gustavo Gilli; 1984). 168 págs.

- Dettmann, Sr; Wyatt, Sr.
  DOS. PROGRAMMER'S REFERENCE
  (Indiana, EUA; QUE, 1993) 4a edición. 1056 págs.

- Joyanes Aguilar, Luis
  PROGRAMACION EN TURBO PASCAL. VERSIONES 4.0,5.0 Y 5.5
  (Madrid, España; McGraw Hill, 1990). 822 págs.

- Sierra On-Line Corp
  LEISURE SUIT LARRY 5. HINTBOOK
  (California, USA; Sierra On-Line, 1991). 64 págs.

- Schildt, Herbert
  C. GUIA PARA USUARIOS EXPERTOS
  (Madrid, España; Osborne/McGraw Hill, 1989). 358 págs.

- Ray Duncan, Ray
  ADVANCED MS-DOS PROGRAMMING
  (E.U.A.; Microsoft Press, 1986). 670 págs.

- Reader's Digest
  MI ABOGADO PERSONAL
  (México, Reader's Digest México, 1991). 640 págs.

- Wyat, Allen L.
  BLASTER MASTERY
  (Indiana, U.S.A.; Sams publishing, 1993). 472 págs.