63

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DISEÑO Y DESARROLLO DE UN SISTEMA DE MENUS GRAFICOS POR MEDIO DE ICONOS DE PROPOSITO GENERAL

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A N

CLAUDIA M. NURICUMBO ROMERO

RUBEN SANTIAGO MAXIMINO

DIRECTOR DE LA TESIS:

ING. RAUL MARTINEZ MERCADO

MEXICO, D. F. 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

AGRADECIMIENTOS

A mi madre quien me ha brindado su apoyo y confianza, a quien le debo lo que soy y sabiendo que no existe ninguna forma de agradecer toda una vida de esfuerzos, quiero que vea que un objetivo que ha sido de las dos ha llegado a su término.

A mi abuela y tíos por el cariño que siempre me han brindado.

A mi hermana porque siempre encontre en ella apoyo y aliento y por haber estado a mi lado en la realización de uno de mis objetivos.

CLAUDIA MONICA NURICUMBO ROMERO

AGRADECIMIENTOS

A mis padres, con mucho cariño y respeto, por todo lo que les debo y en especial a mi madre que siempre me apoyo en todo momento y que al final de todo, hemos realizado un sueño.

A mis hermanos, que siempre me alentaron a seguir en una empresa difícil y que al final logramos terminar, en especial a Arnulfo y Toño que estuvieron conmigo en los momentos de flaqueza y duda.

A mis amigos, Carlos, Armando, Joel, Toño, Oscar, Fernando, Miguel, Andrés y Angelina que siempre me dieron su apoyo y palabras de aliento.

A mis compañeros y amigos de trabajo, Ana Lilia, Martha, Bertha, Luis y Lucía que estuvierón alentandome a terminar esta obra.

RUBEN SANTIAGO MAXIMINO

AGRADECIMIENTOS

Con nuestro más profundo respeto, admiración y gratitud al Ing. Raúl Martínez Mercado quien nos ha dado su confianza, apoyo y muy valioso tiempo durante la realización de este trabajo, sin dejar de resaltar la excelente dirección y al mismo tiempo el haber compartido con nosotros su sapiencia.

CLAUDIA MONICA Y RUBEN

INDICE

INTRODUCCION.

En la actualidad existe una gran variedad de equipos de cómputo que, entre otras cosas, almacenan grandes cantidades de información y la procesan a velocidades muy rápidas, además de ofrecer muchas opciones para lograr una excelente presentación. Por otro lado, la ejecución de los diversos sistemas que se desarrollan en la actualidad, tienen mucho mejor calidad y presentación, ya que éstos combinan texto, números y gráficos.

En el mercado existen páquetes como Harvard Graphics, Word, Lotus, Fox Pro, etc. Los cuales, han mejorado mucho su ejecución gracias a las modificaciones que han tenido para trabajar en ambiente Windows. Agregando a todo lo anterior, la incorporación de un dispositivo de entrada como es el "ratón", se logra un movimiento mucho más rápido sobre las opciones de los menús de los diferentes paquetes y la operación de los mismos se hace de forma más eficiente y rápida.

También podemos observar que el ambiente windows, se auxilia principalmente de el concepto de íconos, siendo éste una representación gráfica de alguna acción a ejecutar.

Por otro lado, sabemos que en el desarrollo de software existe un gran problema que es el de la comunicación entre el lenguaje en el que se presentan los comandos del paquete y el lenguaje que hablan los diversos usuarios. Una solución al problema anterior es la utilización de gráficos y/o íconos con los cuales se pueden representar las diferentes acciones y

opciones con que cuenta un sistema haciendo más fácil el entendimiento y la ejecución de las mismas.

Motivados por todo lo anterior y ante la curiosidad de saber como operan a nivel programación tanto de hardware como software, nos hemos fijado como objetivo el desarrollo de un sistema, el cual tendrá dos funciones:

- Generar un editor de íconos, donde podamos realizar imágenes gráficas que representen una acción a ejecutar. Cabe aclarar que éstas serán mostradas en dos dimensiones, es decir, que al desplegarlas en el monitor de la computadora se verán planas.

- Generar un menú comprendido por los íconos ya elaborados en el editor. Y cuando se seleccione una representación gráfica por medio del teclado o del ratón se ejecute un comando de MS-DOS.

Para poder realizar el sistema anterior fué necesario investigar diversos temas entre los cuales se encuentran los siguientes:

La teoría de monitores y adaptadores gráficos, es decir, tener el conocimiento de los diferentes displays que existen, la operación de los monitores monocromáticos y a color.

Conocer y entender los diferentes adaptadores gráficos, las tarjetas que existen en la actualidad, como las MDA, CGA,

EGA, HGA, entre otras, y los diferentes modos gráficos en que pueden operar.

Estudiar conceptos generales sobre gráficos, entre los cuales están:

La interpretación que el cerebro humano tiene al visualizar texto y gráficos.

Los tipos de velocidades para la ejecución de gráficos sobre la pantalla.

Las funciones gráficas avanzadas.

Las herramientas con que se cuenta para hacer más fácil el trabajo con gráficos, la combinación de colores, distinguir la diferencia entre sombras, tonos y tintes.

Por último, la investigación sobre como realizar las interrupciones al Sistema Operativo siendo éste un tema básico para el desarrollo de nuestro sistema, especifícamente sobre la interrupción "0x16" que se refiere a las interrupciones con el BIOS y sobre la interrupción "0x33" que se refiere a las interrupciones con el ratón

Con la investigación realizada, se tienen las bases para el desarrollo de el sistema. El cual será presentado en cuatro capítulos que se mencionan a continuación:

-Teoría de Monitores y Adaptadores Gráficos.

-Conceptos Generales Sobre Gráficos.

-Análisis del Sistema del Menú de Iconos.

-Desarrollo del Editor de Iconos.

CAPITULO UNO

TEORIA DE MONITORES Y ADAPTADORES GRAFICOS

1.-TEORIA DE MONITORES Y ADAPTADORES GRAFICOS.

El principio en que se basa el funcionamiento de un monitor son los displays electrónicos , los cuales se pueden clasificar en dos tipos:

1.1.-CLASIFICACION DE LOS DISPLAYS.

Displays Emisores:

Aquellos que son capaces de generar su propia energía lumínica. El que ha dominado es el llamado Tubo de Rayos Catódicos (CTR) por su facilidad de uso y bajo costo. Recientemente se ha dado a conocer el gas discharged panel o plasma panel , el cual cuando se combina con paneles muy sensibles al tacto sirve para interactúar con las imágenes desplegadas. También existen los displays cathodoluminiscentes los cuales usan una pequeña matríz especial de pixeles que requiere un display character-only, son conocidos como Vacuum Fluorescentes Displays. Por último están los Diodos Emisores de Luz los cuales fuerón muy populares por su portabilidad , actúalmente pueden ser arreglados en forma de matríz para formar un display de dos dimensiones.

Displays no Emisores:

Aquellos cuya reflexión o transmisión depende de una fuente de luz separada, la que puede ser un bulbo o dayligth.

1.2.-FUNCIONAMIENTO DEL CTR.

El CTR o Tubo de Rayos Catódicos es un tubo de vacío dentro de el cual se generan las imágenes que son desplegadas en la pantalla.

En la figura 1-1 se muestra el esquema del CTR básico. El cual funciona de la siguiente manera:

En primer lugar se tiene un disparador de electrones, el cual emite un haz de electrones (rayos catódicos) que atraviesan los sistemas de enfoque y deflexión que dirigen el haz hacia puntos específicos sobre la pantalla previamente cubierta de fósforo.

Es de esta forma como el fósforo emite una mancha de luz en cada punto contactado por el haz de electrones.

La deflexión del haz de electrones se puede realizar con campos eléctricos o magnéticos, el método electrostático es el que se muestra en la fig. 1-1.

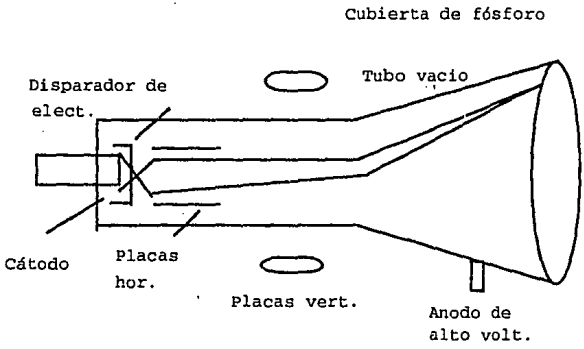


Fig.-1-1.-Funcionamiento de un CTR.

El haz pasa a través de dos pares de placas de metal uno vertical y otro horizontal, se aplica cierto voltaje dependiendo de la cantidad del haz que se deflexionará primero en cada dirección. Conforme el haz pasa entre cada par de placas, este se dobla hacia el par que tenga un voltaje más positivo. De esta forma, el haz de electrones disparado primero se mueve hacia un lado de la pantalla y al pasar por el siguiente par de placas se mueve hacia arriba o abajo de la misma.

Por lo tanto variando las deflexiones en forma continúa y simultánea se pueden trazar líneas y curvas que para hacerlas más complicadas requieren que se hagan cambios no lineales de corriente en las placas.

Para conservar el fósforo brillando se traza la imagen varias veces dirigiendo rápidamente el haz de electrones hacia atrás sobre los mismos puntos, a este tipo de despliegue se le denomina CTR de renovación.

Una característica importante del fósforo es su persistencia, es decir, el tiempo que continúará emitiendo luz después de que se suprime el haz de electrones. Esta se define como el tiempo que tarda la luz emitida en perder una décima parte de su intensidad original. Los fósforos con persistencia pequeña requieren de renovación más alta para poder tener una imagen constante en la pantalla, estos son útiles en la animación, es decir, cuando se requiere tener formas con movimiento; al contrario de los fósforos de alta persistencia que se utilizan para desplegar imágenes estáticas muy complejas.

Cubriendo la parte interna de la superficie del display del CTR (Tubo de Rayos Catódicos) con un tipo de fósforo en particular el color de la luz producida y el tiempo de permanencia de esta pueden ser controlados. De esta forma si se aplican diferentes tipos de fósforo en dicha superficie, se pueden generar varios colores.

En el caso de la deflexión con campos magnéticos, para obtener la dirección exacta del haz de electrones es necesario ajustar la corriente que pasa entre las bobinas que se encuentran colocadas alrededor de la parte exterior de la cubierta del CTR.

Los componentes más importantes en un disparador de electrones en un CTR son el cátodo de metal caliente y la retícula de control. Si se aplica corriente en la bobina de alambre, llamada filamento, se trasmite calor al cátodo cilíndrico. Por esta razón, se pierden electrones de la superficie del cátodo caliente. En el interior, al vacío de la cubierta del CTR, los electrones libres con voltaje negativo se aceleran hacia la cubierta fosforescente, por medio de una tensión positiva muy alta. Está puede generarse por medio de una cubierta de metal con carga positiva que sea colocada en el interior de la cubierta del CTR cercana a la pantalla fosforescente.

Otra forma de generar esta tensión es emplear un ánodo acelerador. Lo anterior se puede observar en la figura 1-2 que muestra la retícula de control por medio de la cual si se fijan distintos niveles de voltaje se puede controlar la intensidad del haz de electrones.

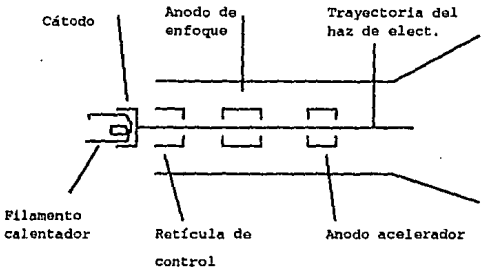


Fig.-1-2.-Operación de un disparador de elect. con ánodo acelerador.

De esta forma, si se aplica un voltaje negativo alto se interrumpe el haz repeliendo y deteniendo los electrones, para que no pasen entre el agujero situado en el extremo de la retícula. Para un voltaje negativo menor, la retícula disminuye el número de electrones que pueden atravesar.

La cantidad de luz emitida por la cubierta fosforescente, es decir la brillantez, depende del número de electrones que chocan contra la pantalla. Por lo tanto para controlar la brillantez de un despliegue se tiene que variar el voltaje en la retícula de control. En los monitores comunes se cuenta con un botón que controla la brillantez de toda la pantalla pero se pueden fijar niveles de intensidad para distintos lugares de la pantalla aplicando comandos de programa.

1.2.1.-MONITORES DE RASTREO AL AZAR Y DE RASTREO CON RASTREADOR.

1.2.1.1.-MONITORES DE RASTREO AL AZAR.

Cuando un CTR trabaja en modo de rastreo al azar tiene el haz de electrones dirigido solamente a ciertas partes de la pantalla donde se trazará la nueva figura. Estos monitores con rastreo al azar dibujan una figura línea por línea por lo que también son llamados monitores de despliegues vectoriales o monitores con despliegue de escritura con golpe. Las líneas que componen cierta figura son trazadas y renovadas por un sistema de rastreo al azar en cualquier órden que se especifique. Este procedimiento de escritura, es muy similar al principio de operación de una graficadora con pluma y es un ejemplo de un dispositivo de rastreo al azar de copia dura.

En lo que se refiere a la renovación de imagen estos sistemas se diseñan, para trazar las líneas componentes de una figura 30 a 60 veces cada segundo, aunque algunos están diseñados, para dar inicio al ciclo de renovación inmediatamente después de que se trazan todas las líneas. Como estos sistemas estan diseñados para el trazo de líneas y no almacenan valores de intensidad para todos los puntos de la pantalla, por lo general tienen resoluciones mayores que los sistemas con rastreador.

1.2.1.2.-MONITORES DE RASTREO CON RASTREADOR.

Los monitores de rastreo con rastreador emiten su haz de electrones de tal forma que se pueda cubrir toda la pantalla, subiendo y bajando la intensidad del haz para que se llegue a formar la imagen deseada. En este caso, la imagen se crea como un conjunto de puntos comenzando a dibujarse al principio de la pantalla. La definición de una imagen se almacena como un conjunto de valores de intensidad de todos los puntos de la pantalla, posteriormente los valores almacenados se despliegan en la pantalla en forma de hileras (línea de rastreo) una por una. Por lo anterior estos sistemas son adecuados para desplegar areas con sombras o con colores, cosa que no pueden hacer los sistemas de vectores restringidos, a las aplicaciones de trazos de líneas.

El ciclo de renovación para este tipo de monitores, se lleva a cabo llevando el haz de electrones a través de cada tercer línea en una vuelta de arriba hacia abajo, después regresando, a lo que se llama repaso vertical, para recorrer las líneas restantes de la pantalla, se hace en forma descendente sobre la misma.

La renovación para este tipo de sistemas se lleva a cabo a intensidades de 30 a 60 cuadros por segundo. A razón de 30 cuadros por segundo, el haz de electrones recorre todas las líneas de la pantalla de arriba hacia abajo 30 veces cada segundo. Con una tasa de renovación menor a 25 cuadros por segundo, la figura se desvanece en la pantalla. En el caso en el que se tengan sistemas con más de 1000 líneas de rastreo

se necesita una tasa de renovación mucho mayor, ésto se logra, por el método de entrelazado.

1.3.-FUNCIONAMIENTO DE UN CTR DE COLOR.

Para que un CTR pueda producir cierto rango de colores es necesario combinar diferentes tipos de fósforos. Como se ilustra en la tabla uno. Se pueden observar diferentes tipos de fósforo, el color que corresponde a cada uno y su aplicación.

FOSFORO	COLOR	APLICACION
P1	Amarillo/Verde	Osciloscopios y radares
P4	Blanco	TV monocromáticos
P22-B	Azul	TV a color
P22-G	Amarillo/Verde	TV a color
P22-R	Rojo	TV a color
P22-Gp	Amarillo/Verde	Gráficas a color
P39	Verde	Displays de baja resolución
P45	Blanco	Displays de unidades enteras

Tabla 1.-Fósforos y sus aplicaciones.

Básicamente existen dos técnicas para producir despliegues a color estas son:

El método de penetración del haz.

El método de la máscara con sombra.

1.3.1.-METODO DE PENETRACION DEL HAZ.

Este método se utiliza con monitores de rastreo al azar y se emplean fósforos que producen colores rojos y verdes los cuales se colocan en forma de una capa muy delgada sobre la superficie de la pantalla , el color que se percibe depende de cuanto penetre el haz de electrones en las capas anteriores. De esta forma, un haz de electrones con poca velocidad excita solamente la capa roja, un haz de electrones con gran velocidad excita la capa verde, un haz de electrones con una velocidad media, emite combinaciones de luz roja y verde para hacer que se perciban los colores amarillo y naranja. Por este método, se pueden producir solo cuatro colores y la calidad de las imágenes no es muy buena, sin embargo, es una forma muy económica de producir colores en un monitor de rastreo al azar.

1.3.2.-METODO DE MASCARA CON SOMBRA.

Para este caso se cubre la pantalla con pequeños modelos triangulares que están conformados por tres puntos fosforecentes diferentes con un mínimo de espacio entre sí, los colores que pueden producir estos modelos triangulares son el rojo, verde y azul.

Este tipo de CTR tiene tres disparadores de electrones, que corresponden a cada punto de color y una máscara con sombra que se coloca détras de la pantalla antes cubierta con las substancias fosforescentes.

Los tres haces de electrones que dependen del disparador de electrones son desviados y enfocados como un solo grupo sobre la máscara con sombra, la cual esta compuesta por un grupo de orificios que están alineados con los modelos triángulares fosforescentes. Cuando los tres haces penetran en un orificio se activa un triágulo que se percibe como una pequeña mancha de color en la pantalla. Para controlar los colores, los puntos que forman los triángulos se disponen de manera que cada haz de electrones puede activar solamente un punto de color cuando atraviese la máscara de sombra. Las variaciones de color se obtienen combinando varios niveles de intensidad en los tres haces de electrones, de esta forma, al no activar los disparadores rojo y verde, se percibe solo el color que proviene de la sustancia azul de igual forma se perciben los colores rojo y verde. Para obtener otros colores, es necesario hacer combinaciones de intensidades de los haces los cuales producen una pequeña mancha de diferente color por cada triángulo; De esta forma, para obtener un color gris o blanco se activan los tres disparadores con igual intensidad. El color amarillo se produce si se excitan solamente el color verde y rojo, el color magneta se produce, si se excitan los puntos rojo y azul.

Si se fabrican sistemas de bajo costo con este sistema, se utiliza un haz de electrones que solo puede apagarse o encenderse limitando el despliegue a ocho colores. Para sistemas más costosos se pueden fijar niveles de intensidad

intermedios de los haces de electrones produciéndose de esta forma, varios millones de colores.

1.4.-OTRAS TECNICAS PARA EL DESPLIEGUE DE IMÁGENES.

Las técnicas para el despliege de imágenes que se tratarán a continúación son tres, la de los tubos de almacenamiento con vista directa (DVST), la de los monitores de cristal líquido y la de despliegues en panel de plasma

1.4.1.-TUBOS DE ALMACENAMIENTO CON VISTA DIRECTA (DVST).

Este tipo de dispositivos almacenan la información correspondiente a la figura como una distribución de carga détras de la pantalla que antes fué cubierta con algún fósforo.

Los DVST's utilizan dos disparadores de electrones. El primero, llamado disparador primario, es utilizado para almacenar el modelo de la imagen; el segundo, llamado disparador de flujo, conserva el despliegue de la imagen.

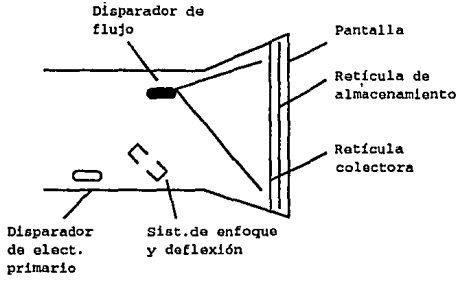


Fig.-1-4.-Operación de un DVST.

En el caso de la figura 1-4, se muestra una sección transversal del DVST, donde el disparador primario se usa para trazar la definición de la figura en la retícula de almacenamiento, que esta formada por un material no conductor. De esta forma, los electrones con alta velocidad, que salen del disparador primario chocan contra la retícula, eliminando electrones, que son atraídos hacia la retícula de control.

Ya que la retícula de almacenamiento no es conductora, las áreas de donde se han suprimido electrones conservarán una carga positiva neta. De esta forma, el modelo de carga positiva, almacenada en la retícula de almacenamiento es la definición de la figura.

Por otro lado el disparador de flujo produce una línea continúa de electrones de baja velocidad que atraviesan después la retícula de control y son atraídos hacia las áreas

positivas de la retícula de almacenamiento cubierta de fósforo, sin afectar notablemente el modelo de carga en la superficie de almacenamiento.

La principal desventaja de los sistemas DVST, es que generalmente no proyectan sus imágenes con color por la forma en que están construídos. La segunda es que las partes seleccionadas de una figura no pueden borrarse, ya que para eliminar un pedazo de imagen es necesario borrar toda la pantalla, almacenando una carga positiva en todas las partes de la retícula de almacenamiento y después los electrones del disparador de flujo chocan contra la cubierta fosforescente en todos los puntos de la pantalla, borrando así la figura en un destello de luz. Posteriormente se vuelve a trazar la imagen completa, exceptuando las partes que se quieran omitir.

1.4.2.-DESPLIEGUES EN PANEL DE PLASMA.

Estos dispositivos, se construyen principalmente de placas de vidrio y gas neón. La región entre dos placas de vidrio, se rellena con gas neón. Un conjunto de electrodos horizontales y verticales, colocados en los paneles frontal y posterior, se utilizan para iluminar puntos individuales en el gas neón. Una forma de ofrecer los puntos individuales consiste en separar el gas en lo que se llaman pequeños 'bulbos', con una placa de vidrio dental que contenga varios orificios con muy poco espacio entre sí.

Un punto de neón individual, en un panel de plasma se enciende aplicando una "tensión de encendido" de aproximadamente 120 volts al par de electrones horizontales y verticales adecuados. Una vez que el punto se enciende, la tensión en estos electrodos se reduce hasta un nivel de "tensión de sustento", aproximadamente 90 v; con esta, se mantiene centellando la celda de neón.

Puede aplicarse una tensión de 90 v en todos los pares de electrones y aquellos puntos que serán iluminados tienen sus potenciales de electrodos momentáneamente elevados con la tensión de encendido. Esto asegura que los puntos que se desplieguen continúen centellando, mientras que todos los otros permanecen apagados. Por lo tanto, el panel de plasma es un dispositivo de memoria inherente, que no requiere renovación de imagen. Para borrar la pantalla, solo es necesario disminuir el voltaje de cada electrodo abajo del voltaje de "tensión de sustento".

1.4.3.-MONITORES DE CRISTAL LIQUIDO.

Estos monitores, combinan organismos que a cierta temperatura alcanzan un estado semejante al líquido. Estando en este estado los cristales, es más fácil controlar su orientación por medio de campos eléctricos. Si los cristales tienen una orientación paralela el líquido, es relativamente transparente, pero si los cristales toman cualquier otra

posición, no paralela a la estructura, entonces el líquido o la luz que emite, será opaca.

Un típico LCD, está compuesto por un delgado rayo de cristal líquido, colocado entre dos capas aislantes transparentes. El rayo delgado, es tratado químicamente para determinar su orientación inicial; de esta forma, el cristal adopta una posición paralela o perpendicular a la superficie sin necesidad de que se le aplique un campo eléctrico. En un display típico las superficies tratadas son colocadas de forma que los cristales de una superficie, sean colocados en el ángulo derecho de los cristales de la segunda superficie. Los cristales que quedan dentro, adoptan una forma espiral o twisted lattice porque su orientación cambia 90 grados por la capa aislante. Existen unas capas polarizadas las cuales orientan la luz que las atravieza en la misma dirección que la de sus vecinas, las capas aislantes.

Si la forma espiral de la capa (el twisted lattice de la capa)de cristal, sufre algún disturbio por un campo eléctrico, la luz producida es corta en la correcta orientación y pasa a través del material polarizado y la estructura comienza a opacarse produciendo una imagen obscura, sobre una luz generalmente verde.

La construcción de un LCD (Liquid Cristal Display) se muestra en la fig. 1-5 . En esta figura la aplicación de un potencial eléctrico sobre los electrodos en cada dirección, genera un campo eléctrico sobre sus intersecciones y el giro del cristal es roto entre los dos. La luz que llega a la capa de cristal es alineada incorrectamente para pasar a través de

la capa polarizada superior. En cambio la fuente de luz, comienza en la parte baja de la estructura, está es comunmente usada como capa reflejante en la parte baja y la fuente de luz hasta arriba de la estructura. Con este arreglo, la luz viaja a través de la estructura con lo cual incrementa la absorción total y el máximo contraste de despliegue. El arreglo de electrodos puede ser reemplazado por "character segment arrays", como en el caso de los displays de las calculadoras de mano.

Relativamente se necesitan voltajes bajos de aproximadamente 6 volts para su operación y su tiempo de respuesta esta en un rango de 100 ms.

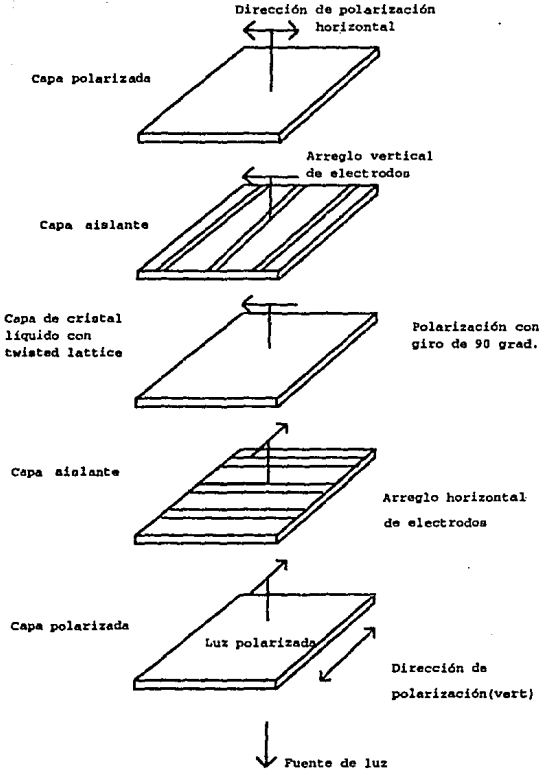


Fig.-1-5.-Construcción de un LCD

1.5.-ADAPTADORES GRAFICOS.

Actúalmente existen una gran variedad de adaptadores gráficos, a continúación se explicará el funcionamiento de la tarjeta MDA, la tarjeta CGA, la tarjeta EGA y la tarjeta HGA.

1.5.1.-TARJETA MDA.

MONOCHROME DISPLAY ADAPTER.-Este adaptador puede desplegar solamente carácteres en colores blanco y negro no es compatible con gráficos. Los carácteres son dibujados en una área de 9 por 14 y son excepcionalmente claros.

Los 8 bits de un byte son completamente usados, por lo que en toda la pantalla se pueden desplegar 256 carácteres, también es posible dibujar carácteres en código ASCII, carácteres que representan líneas de un dibujo, carácteres matemáticos y algunos carácteres " misteriosos " que son los llamados **dingbats** (caras sonrientes), y notas musicales. Cada carácter es asociado con un atributo de un byte. Los atributos de los carácteres pueden ser:

-Que sean centellantes
-Que sean intensos
-Que puedan colocarse en forma inversa

-Que puedan ser subrayados, normales, separados
-O la combinación de cualquiera de los anteriores.

Los otros bits del byte son desperdiciados, esto es porque el mismo arreglo de memoria es usado para el color. El controlador del despliegue y el CPU reparten la memoria del buffer. Cada uno puede accesar este en cualquier tiempo sin que haya una interferencia.El display opera a una velocidad aproximada de 112,000 bauds, esto es, cerca de 100 veces más rápido que una típica terminal de 1,200-bauds. El controlador se puede operar con un buen display, llamado comunmente display TTL.

1.5.2.-TARJETA CGA

COLOR GRAPHICS ADAPTER.-Este adaptador cuando se utilizá en modo carácter, actúa de una forma igual al adaptador monocromático, ya que utiliza el mismo arreglo de memoria, tiene la capacidad de desplegar los mismos carácteres y soporta los mismos atributos exceptuando el subrayado; el bit, que se utilizá para este atributo, así como otros, son usados para desplegar colores. En este caso, se tienen diéciseis colores para el frente o de primer plano y ocho colores de fondo o de segundo plano.

Cuando se trabaja en modo gráfico se presenta la resolución de 200 pixeles de largo por 320 o 640 de ancho.

Cuando se tiene la más baja resolución un pixel puede tener uno de cuatro colores, pero en la más alta resolución, el pixel desplegado, puede ser blanco o negro.

En el modo carácter o de alta resolución, el CPU y el controlador de despliegue no pueden accesar al buffer al mismo tiempo, por lo que el CPU no puede volver excepto en intervalos retrasados.

Se puede tener una gran variedad de monitores para el controlador, desde un monitor para TV hasta uno RBG (llamado así porque generan colores rojos, verdes y azules respondiendo directamente a las señales emitidas por el controlador) el monitor no debe de ser de un solo color porque mostrará solamente sombras de gris.

1.5.3.-TARJETA EGA.

ENHANCED GRAPHICS ADAPTER.-Este adaptador puede emular a los dos anteriores, con la diferencia de que esté puede accesar el frame del buffer sin ninguna interferencia. Con la característica anterior se pueden tener algunos modos gráficos adicionales.

Uno de estos modos gráficos es el que permite que se despliegen gráficas en el mismo display que utilizan los

adaptadores monocromáticos con una resolución de 350 por 640 pixeles.

Los otros adaptadores son solo para color pudiendose dar las resoluciones de : 200 por 320 o 200 por 640 pixeles con diéciseis colores, y 350 por 640 con sesenta y cuatro colores. Para obtener diferentes colores con un adaptador EGA es necesario usar un monitor especial, ya que si se utiliza el estandard, que es un RGB el adaptador no funciona.

1.5.4.-TARJETA HGA.

HERCULES GRAPHICS ADAPTER.-Este adaptador gráfico trabaja igual que el EGA en el aspecto de que puede trabajar gráficos, sobre un monitor monocromático, pero con una resolución más alta que es de 348 por 720 pixeles.

En lo que se refiere al despliegue de texto el HGA trabaja en forma idéntica al adaptador MDA, ya que en su despliegue no puede mostrar todos los colores y no puede direccionar o trabajar en otro tipo de monitores.

En conclusión, para trabajar en modo gráfico, se tienen varias resoluciones dependiendo de la combinación que se haga del monitor y del adaptador. Recientemente IBM lanzó al mercado un adaptador que alcanza una resolución de 480 por 640

pixels. A diferencia de cuando se trabaja en modo texto ya que solo se tienen 25 renglones por 80 columnas.

1.6.-INTERRUPCIONES LOGICAS.

Para el desarrollo del menú de íconos se utilizará una máquina que como mínimo, cuente con un microprocesador 286, esto se debe a las carácterístacas de dicho microprocesador. A continúación se presenta una breve reseña de como han hido evolucionando los microprocesadores y sus carácterísticas.

La velocidad de desarrollo de los microprocesadores ha sido muy rápida. El primer circuito integrado fué construído alrededor de los años sesentas, este circuito ofrecía bastantes ventajas ya que reducia muchísimo el tamaño del conjunto de capacitores, diodos y transistores colocándolos en una pequeña oblea de silicio puro. Posteriormente, a principios de la década de los setentas Intel desarrollo su primer microprocesador de 8 bits el 8088. Para 1974 se forma lo que sería la segunda generación de microprocesadores, el 8080; Y al mismo tiempo se lanzarón al mercado productos como el Zilog Z-80. Cuatro años más tarde, llega la tercera generación de microprocesadores el 8086, y poco tiempo después el 8088, que permitía diseños más simples y era compatible con dispositivos de E/S más actúales, como en esa época era de los más avanzados IBM lanzó su primera generación de computadoras personales basadas en este.

Para el año de 1984, Intel logró un gran éxito al desarrollar el 80286, el cual fué diseñado principalmente para aplicaciones que requerían alta precisión, compatible con el 8086/8088, debido a su conjunto común de modos de direccionamiento e instrucciones básicas, aprovechó las

carácterísticas del estado de arte en gestión de memoria, mecanismos de protección, gestión de tareas y mecanismos de memoria virtual. Lo que proporciona al usuario de microcomputadoras carácterísticas de arquitectura y cálculo de las minicomputadoras.

Su arquitectura base soporta lenguajes de alto nivel como Pascal, PL/m, y C ya que el diseño del conjunto de registros está bien adaptado al código generado por el compilador. Soporta diferentes tipos de datos muy potentes como cadenas BCD y formatos en punto flotante; También, soporta un direccionamiento eficiente de estructuras complejas de datos, como arrays estáticos/dinámicos, registros y arrays de registros.

Por otra parte su arquitectura de memoria, soporta técnicas de programación modular, lo que permite dividir la memoria en segmentos. La segmentación de memoria proporciona un código más corto, ya que las referencias a un segmento pueden ser menores. El esquema de segmentación, se presta a implementaciones eficientes de gestiones sofisticadas de memoria, por ejemplo memoria virtual y protección de memoria.

El 80286, proporciona un gran espacio de direcciones para soportar los requerimentos de las aplicaciones actúales. La memoria real consta de 16 megabits de RAM o ROM. Este espacio permite al procesador, guardar en memoria programas muy grandes y sus correspondientes estructuras de datos, permitiendo accesos a alta velocidad.

Para aplicaciones con requerimientos dinámicos de memoria, por ejemplo sistemas multiusuarios, suministra a cada usuario aproximadamente, un gigabyte de espacio virtual de direcciones. El gran espacio de direcciones casi elimina las restricciones, sobre el número o tamaño de los programas que pueden ser parte del sistema.

Para las redes de computadoras que dan servicio a cuatro o cinco usuarios simultáneamente pueden expanderse, para soportar más de tres veces ese número y los sistemas en tiempo real pueden responder en hasta un sexto del tiempo, que se necesitaba antes.

Por último, se tiene el 80386 para el cual el bus de datos externos es de 32 bits, el doble que el del 80286, y puede direccionar cuatro gigabytes de memoria.

1.6.1.- INTERRUPCIONES DEL BIOS Y DOS.

Se podría decir que las rutinas del BIOS y DOS son el cerebro de una computadora. Todas estan programadas con cierta cantidad de información la cual esta almacenada en el hardware (ROM) de la máquina y la mayoría de las rutinas del BIOS, estan guardadas aquí y son permanentes. El DOS, es consideardo como una memoria añadida posteriormente. Realmente es memoria que se añade, cada vez que la computadora se inicializa con el DOS.

Típicamente, las operaciones del DOS, son almacenadas en la RAM o residen en el disco hasta que se necesitan, juntas la ROM y la RAM controlan la operación de la computadora.

Cada tipo de máquina (8088,80286,80386), tiene su propio conjunto de órdenes del BIOS y DOS, pero la mayoría de las rutinas principales son compartidas.

La gran mayoría de las veces, cuando se realiza un programa, la entrada o salida de la computadora necesita ser mostrada en la pantalla del monitor. Con frecuencia, ésto requerirá una simple operación de borrado de pantalla. El control de pantalla, puede conseguirse en forma rápida utilizando las interrupciones del BIOS tipo 10H.

CAPITULO DOS

CONCEPTOS GENERALES SOBRE GRAFICOS

2.-CONCEPTOS GENERALES SOBRE GRAFICOS.

2.1.-DIFERENTES MODOS GRAFICOS.

Actualmente los gráficos son un componente vital en la programación de computadoras. Ya que la mayoría de los programas, los incluyen para una mayor comprensión

La gran mayoría de las computadoras personales IBM y compatibles poseen un adaptador gráfico ya sea un VGA, CGA, EGA, etc. Por otro lado, un gran número de usuarios que ya estan familiarizados con los gráficos esperan los programas donde se utilicen interfases gráficas.

Estas personas encontrarón que al usar cualquier sistema o paquete que se maneje a través de gráficos se tiene un gran ahorro de tiempo para aprender a usarlo y para ejecutar diversas tareas. Sin embargo, se toma consciencia de que los gráficos no son un sustituto de los textos, el uso de los gráficos sirve para, junto con un texto mejorar la ejecución y presentación de cualquier sistema.

2.2.-INTERPRETACION DE LOS GRAFICOS POR EL CEREBRO HUMANO.

Usando la combinación de gráficos y textos en la mayoría de los sistemas, se puede mejorar en gran medida la calidad de comunicación entre el sistema y el usuario final. Esta mejora se debe a la forma en que trabaja la psicología del cerebro humano.

El hemisferio izquierdo del cerebro tiene la función de interpretar material en forma de texto mientras que el hemisferio derecho tiene la habilidad de interpretar objetos, es decir, posee una interpretación visual. Investigaciones médicas concluyerón que el hemisferio izq. trabaja con partes de cualquier cantidad y secuencias de partes, por otro lado, la parte derecha trabaja con cantidades completas, es decir, con objetos enteros.

Una porción significativa de la mente humana no es capaz de retener solo texto con gran facilidad, por lo que si se le ayuda introduciendo gráficos se incrementa el canal de comunicación del sistema con el entendimiento del cerebro humano.

Como un ejemplo la figura 2-1 se representa en dos formas distintas. En la parte superior existen solo frases para presentar una idea, mientras que la parte de abajo contiene textos y gráficos para representar la misma idea. Por lo que, se puede observar que es más fácil comprender la segunda opción.

Sin embargo, esta forma de comprender las cosas se puede mejorar. Si la segunda opción del dibujo anterior tuviera

animación sería mucho más fácil la comprensión de la idea además de agradable. Recientes estudios científicos demuestran que la comprensión de las ideas se incrementa cuando la animación es adherida a alguna secuencia de imágenes. La conclusión es que si se requiere de buena comunicación del usuario con el sistema es necesario usar gráficos pero, si se requiere la máxima comunicación se deben usar gráficos animados.

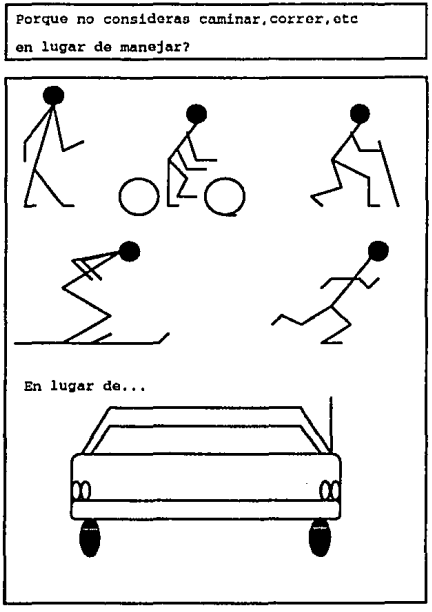


Fig.-2-1.-Combinación de gráficos y texto para una mayor comprensión.

Los siguientes paquetes basan su existencia en la creación de gráficos por computadora:

CAD.-Diseño Asistido por Computadora
CAE.-Ingenieria Asistida por Computadora
CAS.-Estilo Asistido por Computadora

Inclusive programas análiticos (negocios,ciencia) requieren gráficos animados, aunque frecuentemente solo expresan fórmulas y textos. Estos pueden experimentar una mejora en la ejecución si se representan en dos y tres dimensiones. Ultimamente hasta los procesadores de texto y los manejadores de Bases de Datos están empezando a incluir gráficos.

2.3.-TIPOS DE VELOCIDADES PARA LA EJECUCION DE GRAFICOS.

Para propósitos prácticos se consideran tres tipos de velocidad para la ejecución de gráficos animados en una computadora personal equipada con un adaptador gráfico de IBM.

1)Frame Animation.
2)Bitblt Animation
3)Real-time Animation.

2.3.1.-FRAME ANIMATION

En términos del programador, una página es una pantalla de gráficos. La página que comienza a desplegarse sobre la pantalla frecuentemente es llamada página de despliegue; Por otro lado la computadora es capaz de guardar varias páginas en la memoria antes de ser desplegadas y estas páginas no mostradas son llamadas "frames" fig. 2-2.

Si cada una de estas frames anteriores contienen versiones ligeramente diferentes de la misma imágen, entonces la animación puede ser creada cuando estas páginas son pasadas sobre la pantalla en una rápida secuencia. Esta técnica es llamada "frame animation" porque todas las gráficas tienen que ser creadas en avance la primera respecto a la segunda y ésta respecto a la siguiente. Esta animación es muy rápida 18 fps (frames per second).

2.3.2.-BITBLT ANIMATION.

Existen casos en los que solo una pequeña porción de la pantalla necesita ser animada, un ejemplo se muestra en la figura 2-3.

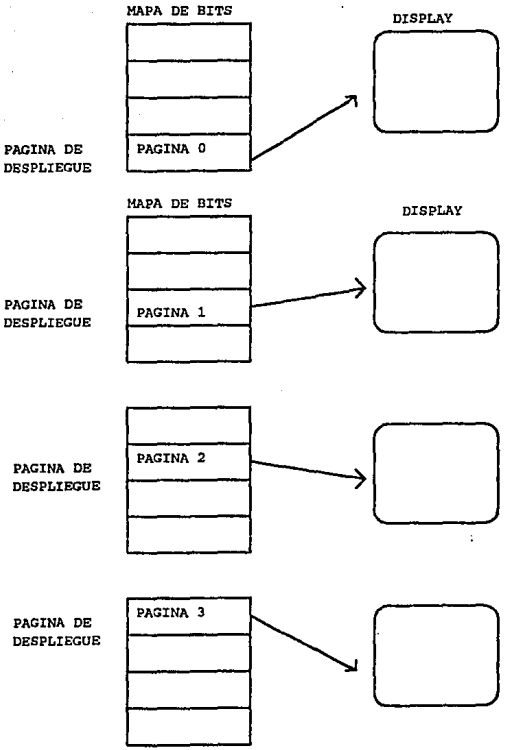


Fig.-2-2.-Frame animation.

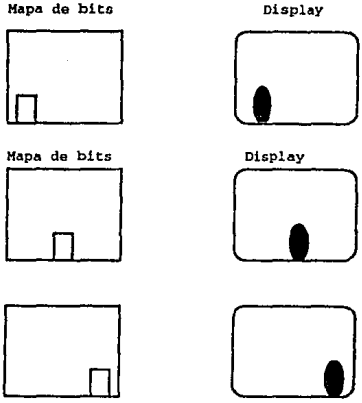


Fig.-2-3.-Bitblt animacion.

Como se ilustra en la fig 2-3, existen casos en los que es necesario mover un pequeño bloque gráfico a través de la pantalla al movimiento resultante se le llama " bitblt animation".

La palabra bitblt se compone de las palabras bit, block y transferencia. También se le llama block graphics o graphics array animation, porque solo una pequeña parte de la pantalla esta siendo manipulada. Este método es muy rápido y usando TC se llega a una velocidad de hasta 30 fps.

2.3.3.-REAL-TIME ANIMATION

Alguna veces un programa tiene que ser diseñado para recibir entradas mientras está corriendo. Para este tipo de programas se utiliza la técnica de Real-Time Animation que se ilustra en la figura 2-4.

Para lograr este propósito se podría emplear un programa que frecuentemente utiliza tiempo real o hidden page animation. Esta técnica de animación también es llamada animación **ping pong**, porque consiste en que una página salta hacia atrás y regresa hacia adelante entre dos páginas.

Tan pronto como una imágen tiene que ser creada la computadora la lanza sobre la pantalla. Mientras el usuario esta observando la nueva imágen sobre la pantalla la computadora esta ocupada dibujando la siguiente imágen en alguna página escondida. Cuando la segunda ilustración es terminada esta se lanza sobre la pantalla creando asi una secuencia de animación.

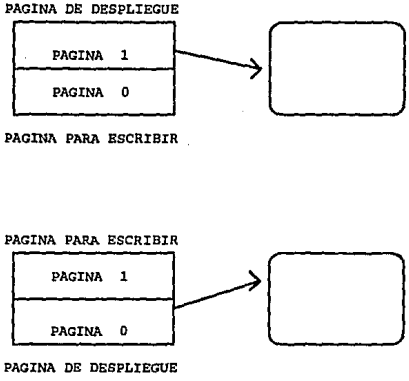


Fig.-2-4.-Real-time animation.

La computadora esta creando los gráficos durante el ciclo de animación, sin embargo el programa puede alterar los tipos de imágenes creando y aceptando las entradas del usuario. Por lo que la computadora está haciendo dos trabajos (procesar imágenes y animarlas) es por ésta razón que la velocidad de la animación en tiempo real es limitada por la cantidad de tiempo que se requiere para crear cada nueva imagen.

2.4.-HERRAMIENTAS GRAFICAS

En general, las herramientas que se requieren para generar rutinas gráficas escencialmente son cinco y se listan a continuación:

a)La habilidad para establecer un modo gráfico en la pantalla.

b)La habilidad para limpiar la pantalla.

c)La habilidad para dibujar líneas.

d)La habilidad para rellenar gráficas con color.

e)La habilidad para pasar material alfanumérico a gráficos.

f)La habilidad para regresar al modo texto por default.

2.4.1.-ESTABILIZANDO EL MODO GRAFICO.

Los adaptadores gráficos VGA,EGA Y CGA pueden soportar una gran variedad de modos gráficos como se muestra en la figura 2-5.

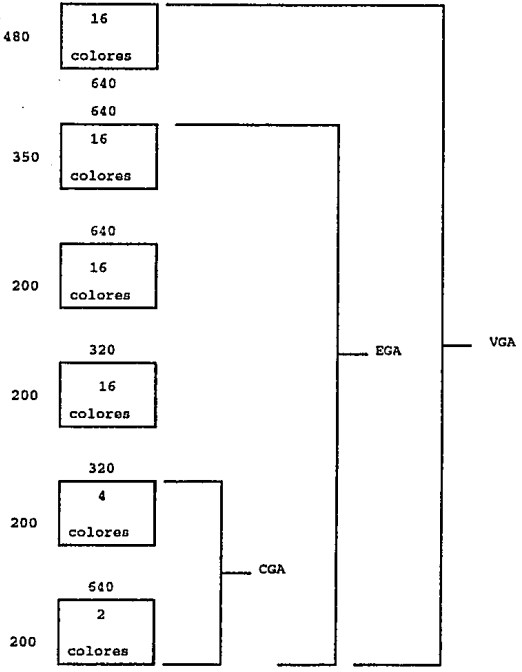


Fig.-2-5.-Modos gráficos para los adaptadores VGA, CGA, EGA.

A continuación se mencionan algunas funciones para cambiar el monitor a modo gráfico tanto para QUICK C como para TURBO C.

QUICK C usa la instrucción _setvideomode, para invocar un modo gráfico. Turbo C comienza el modo gráfico, usando la instrucción initgraph.

Con quick c:

Se puede usar:	Para invocar el modo:
_setvideomode(_vres16color)	640x480 16-color mode P/VGA
_setvideomode(_erescolor)	640x350 16-color mode P/VGA, EGA.
_setvideomode(_hres16color)	640x200 16color P/VGA,EGA
_setvideomode(_mres4color)	320x200 4-color P/VGA,EGA,CGA.
_setvideomode(_hresbw)	640x200 2-color P/VGA,EGA,CGA.

Con Turbo c:

Se puede usar:	Para invocar el modo:
_initgraph(vgahj)	640x480 16_color P/VGA
-initgraph(egahi)	640x350 16-color P/VGA,EGA
_initgraph(egalo)	640x200 16-color P/VGA,EGA
_initgraph(cgac3)	320X200 4-color P/VGA,EGA, CGA.
_initgraph(cgahi)	640x200 2-color P/VGA, CGA, EGA.

En caso de que no se presenten el adaptador y monitor necesarios, las funciones anteriores regresarán un error.

Al establecer un modo gráfico para poder crear gráficas sobre la pantalla, generalmente se tienen que utilizar coordenadas 'x','y'. Si se usa el modo 640x200 y 'x' tiene un rango de 0-639 'y' de 0-199 el centro de la pantalla tendrá las coordenadas (319,99), es necesario primero definir la coordenada horizontal y después la vertical.

2.4.2.-LIMPIANDO LA PANTALLA

La acción de limpiar la pantalla se ejecuta en una forma muy sencilla, simplemente haciendo la llamada a las funciones correspondientes. En caso de trabajar con Quick C se puede utilizar la instrucción _clearscreen(gclearscreen) para blanquear la pantalla gráfica, en caso de que se trabaje con TC, la instrucción cleardevice().

2.4.3.-DIBUJANDO LÍNEAS

La habilidad para dibujar líneas simples, es muy importante ya que de aquí se pueden hacer polilíneas, curvas, círculos, pequeños segmentos, etc.

El procedimiento que se sigue es el mismo para Quick C y Turbo C. Las líneas se deben dibujar con la instrucción _lineto (x,y), pero antes se debe usar la instrucción _setcolor() para definir el color de la línea a dibujar y _moveto() para establecer el punto de comienzo de la línea.

En caso de que se hubiese dibujado una línea en el centro de la pantalla de 640x200 y se requiera mover a la esquina inferior derecha de la pantalla con Quick C seria así:

```
_moveto(319,99);

_lineto(639,199):
```

Con TC :

```
moveto(319,99);

lineto(639,199);
```

2.4.4.-ILUMINACION O RELLENO DEL AREA.

La función gráfica que rellena las áreas de un color en partícular es llamada areafill, floodfill o painting. Opera definiendo un punto como semilla o comienzo para empezar a rellenar el área con cierto color hasta que otro color partícular es encontrado y éste es el límite de la figura.

Para empezar a iluminar el área se utiliza la función floodfill(x,y,lim) y así el área comienza a iluminarse en el punto descrito por 'x','y' y continua en todas las direcciones hasta que un pixel que contiene el valor de la variable, 'lim' es encontrado.

2.4.5.-ESCRIBIENDO TEXTO SOBRE LA PANTALLA DE GRAFICOS.

Al usar quick y TC, se pueden cambiar carácteres alfanuméricos a gráficos pero, se usan diferentes aproximaciones para esta función.

Al usar Quick C la tonalidad del texto es descrita en renglones y columnas, en el modo 640x200 y 640x350 se tienen 80 carácteres sobre 25 renglones, en el modo 640x480 se tienen 80 carácteres por cada 30 renglones. Por ejemplo, para desplegar la palabra 'hola' en el modo 640x200, se tienen que seguir los siguientes pasos:

1.-Llamar a la función _settexcolor(color) para elegir el color a usar, después usar la instrucción _settexposition (renglon , columna) para identificar la localización del primer carácter,el parámetro de la columna abarca del 1 al 80 y para renglones del 1 al 25.

2.-Finalmente utilizar la instrucción _outtext ('hola') para escribir carácteres alfanuméricos sobre la pantalla gráfica.

En TC la localización del texto a desplegar esta dada por las coordenadas gráficas 'x','y'. En el modo 640x200/350/480 un máximo de 80 carácteres son desplegados a través de la pantalla. Un máximo de 25 carácteres pueden ser impresos desde

la parte alta en el modo 640x200 y 640x350. En el modo 640x480 80 carácteres pueden ser impresos en un máximo de 80 renglones. La posición de comienzo del texto es identificada por las coordenadas 'x','y'.

En caso de que se requiera usar TC se tienen que seguir los siguientes pasos:

1.-Llamar a la función setcolor(color) para seleccionar el color del texto (la cual también define el color de la gráfica).

2.-Posteriormente usar la función moveto (x,y) para identificar la posición del primer carácter en el texto. El parámetro 'x' tiene un rango de 0-639, el parámetro 'y' va de 0-199 o 0-349 o 0-479 dependiendo del modo gráfico de la pantalla.

3.-Finalmente podras usar la función outtext('hola') para escribir alfanúmericos a pantalla gráfica.

Nótece que quick C, sobreescribe la existente pantalla gráfica cuando el texto es escrito sobre un modo gráfico, desplegando cada carácter como un color alfanúmerico sobre un campo negro. TC por otro lado sobrepone el texto sobre la existente pantalla gráfica, permitiendo que los antecedentes gráficos se muestren continuamente.

2.4.6.-REGRESANDO AL MODO TEXTO.

Un programa creado para desplegar gráficas debe tener la habilidad de regresar a modo texto una vez que se halla completado la tarea ver la figura 2-6.

Si se está trabajando con Quick C la instrucción a ejecutar será _setvideomode (_DEFAULTMODE). Y en el caso de Turbo C restorecrtmode().

Fig.-2-6.-Procedimiento para trabajar en modo texto.

2.5.-FUNCIONES GRAFICAS AVANZADAS.

Las funciones gráficas que se explicarán en esté punto son cinco:

Halftoning, Line dithering, Page copying, Block graphics y la habilidad para dibujar gráficas en páginas escondidas.

A) Medio Tono (Halftoning)

B) Línea vibrando (line dithering)

C) Copiando páginas (page copying)

D) Habilidad para dibujar gráficas en páginas escondidas.

E) Gráficas de bloque.(block graphics)

2.5.1.-HALFTONING.

Lo escencial de esta técnica es que se pueden crear patrones. También, es llamada bit tiling.

La fig. 2-7, ilustra diferentes niveles de sombra que pueden ser creados en este proceso. Si sólo se rellenan la mitad de los pixeles, con una función de relleno de áreas se creará una sombra del 50%. De la misma manera si se rellena una cuarta parte se creará una sombra del 25%. De esta forma se puede buscar la manera de crear nuevos colores en base a los colores estándar que se tienen dependiendo del adaptador gráfico.

Con Turbo C se pueden generar un conjunto de patrones con la instrucción setfillpattern. Y utilizando la instrucción floodfill se rellena un polígono o con bar (x1,y1,x2,y2), se crea y rellena un rectángulo.

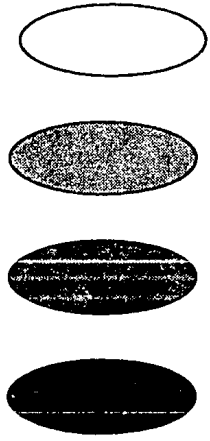


Fig.-2-7.-Diferentes tonos de sombras.

2.5.2.-LÍNEA VIBRANDO.

Esta técnica, también es llamada line style y se refiere al estilo de los puntos usados, para dibujar una línea. Turbo C dibuja las líneas como una solida identidad, pero puede crear líneas con efectos vibradores los cuales son usados en gráficas de tercera dimensión. En Turbo C se inicializa la línea patrón, con la instrucción:

```
setlinestyle (userbitline,OxAAAA,norm_width).
```

2.5.3.-COPIANDO PAGINAS

En esta técnica, se tiene cierta imágen dentro de una página y ésta pasa de una forma muy rápida a otra página limpia o en blanco, generalmente es así como funcionan la mayoría de los programas de animación. Al movimiento de la imagen de una página a otra se le llama page coping.

Este método se puede utilizar cuando se tienen adaptadores gráficos EGA y VGA. En caso de que se cuente con un CGA se podría hacer una simulación de páginas gráficas con la ayuda de la memoria RAM.

2.5.4.-HABILIDAD PARA DIBUJAR GRAFICAS EN PAGINAS ESCONDIDAS.

Como los adaptadores VGA y EGA ofrecen numerosas páginas gráficas no es necesario hacer una gráfica en la misma página que esta siendo desplegada sobre el monitor. La habilidad para mostrar una página escondida depende de la capacidad para crear animación en tiempo real.

Cuando se está utilizando Turbo C la instrucción setactivepage() contiene o permite la escritura a una página y la instrucción setvisualpage() permite o contiene la página desplegada.

Como los adaptadores CGA solo contienen la suficiente memoria para desplegar una página puede utilizarce la técnica de dibujar en una página escondida y así ocupar cierta area en la RAM para simular páginas de gráficos.

2.5.5.-GRAFICAS DE BLOQUE.

El tipo de velocidad Bitblt animation utiliza la técnica de gráficas de bloque para poder mover entidades gráficas a través de la pantalla. En algunos otros lenguajes de programación esto es conocido como **graphics array animation.**

En escencia, los bytes con los cuales se forma el block sobre la pantalla son salvados en la memoria como un arreglo de una sola dimensión.

Una pequeña cabeza en el comienzo del arreglo cuenta los bits a lo ancho y profundo del block. Usando esta información, el programa puede colocar el arreglo anterior sobre la pantalla. Tecnicamente hablando, no se coloca el arreglo sobre la pantalla, sino que se escribe el arreglo dentro de la memoria del adaptador gráfico. La técnica de Bitblt puede producir efectos de gran velocidad, debido a los diferentes operadores lógicos, los que son usados para controlar la forma en la cual cada byte es escrito sobre la pantalla.

Con Turbo C se puede almacenar el arreglo del bloque gráfico en la RAM, usando la instrucción getimage y por último, el arreglo es colocado sobre la pantalla usando la instrucción putimage.

2.6.-DIBUJANDO SOBRE LA PANTALLA.

Básicamente existen dos métodos para dibujar sobre la pantalla, el método de las coordenadas dinámicas y el método de las coordenadas de la base de datos. Ambas aproximaciones utilizan las coordenadas xy derivadas de una plantilla patrón o módelo .Como se muestra en la figura 2-8.

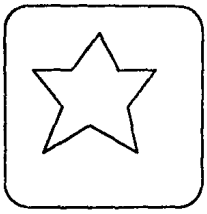


Fig.-2-8.-Patrón o modelo.

2.6.1.-COORDENADAS DINAMICAS Y DE BASE DE DATOS.

Las coordenadas xy, sobre la pantalla son construidas con las instrucciones, para dibujar de TC, pero ésto dificulta la creación de ciclos que pueden ser necesitados en procesos de dibujo repetitivamente. Con las coordenadas dinámicas se ofrece al programador en TC la opción para desarrollar prototipos muy rapidamente.

Cuando se utilizan coordenadas de base de datos, se almacenan todas las coordenadas xy en una base de datos (usualmente en un arreglo en RAM). Mientras los procesos de dibujo están en forma escondida, el programa trae las coordenadas que se requieren desde la base de datos. Por lo que las coordenadas xy no son actualmente codificadas a lo largo de las instrucciones de cada dibujo, ésto hace posible que se puedan escribir loops para repetir procedimientos de dibujo. Por cada pasada a través del loop el programa recupera otro conjunto de coordenadas xy desde la base de datos.

Si se quisiera dibujar con coordenadas dinámicas el patrón de la fig.2-8, se seguirian los pasos de la figura 2-9.

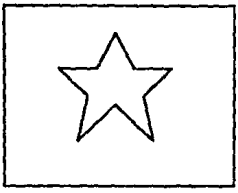
Por inspección de la figura anterior se puede comenzar a crear un algoritmo para producir la imagen que se ve en la pantalla. Por lo que se tienen que seguir los siguientes pasos:

Primero el programa puede llenar una área para crear un campo antecedente al sólido. Entonces puede ser creado un rectángulo negro sólido representando la sombra que es arrojada por el gráfico (llamada dropshadow por los artistas y diseñadores gráficos).

Posteriormente puede ser creado un cuadro sólido coloreado.

Finalmente se requiere una polilínea para dibujar la silueta de la estrella la cual, puede ser iluminada con el color apropiado posteriormente.

En el diagrama de la fig 2-10 se muestra el flujo lógico, para la creación del programa de la fig. 2-9.



Plantilla de 640x480

Encontrar las coordenadas x,y

Pasar a la pantalla las coordenadas

Fig.-2-9.-Coordenadas dinámicas.

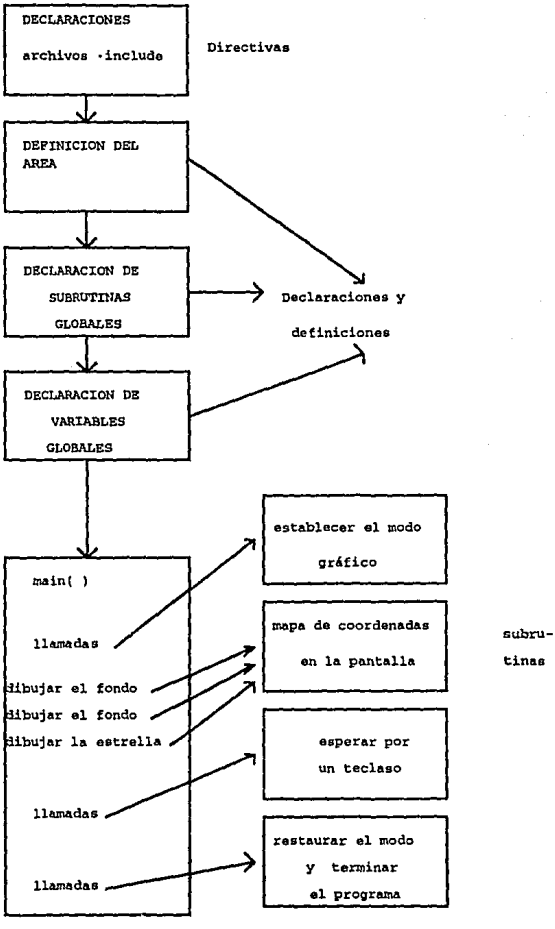


Fig.-2-10.-Flujo lógico para la creación de la fig. 2-9.

2.7.-COMO SELECCIONAR EL COLOR.

Los adaptadores gráficos VGA, EGA, CGA, etc. encontrados en las computadoras personales IBM y compatibles son capacez de generar una amplia gama de colores, y ésta carácterística se puede incrementar al conocer y aplicar la terminologia de los colores que se presenta más adelante.

2.7.1.-TERMINOLOGIA DE COLORES

Los colores cromáticos son aquellos que tienen su propio matíz (o tinte), por ejemplo los que se muestran en la figura 2-11. Por otro lado los colores acromáticos son aquellos neutros, por ejemplo blanco, gris y negro. Estos no tienen su propio matíz o tinte.

Un punto importante de entender es conocer la diferencia entre matíz, valor y croma a continuación se mencionarán las definiciones de cada una.

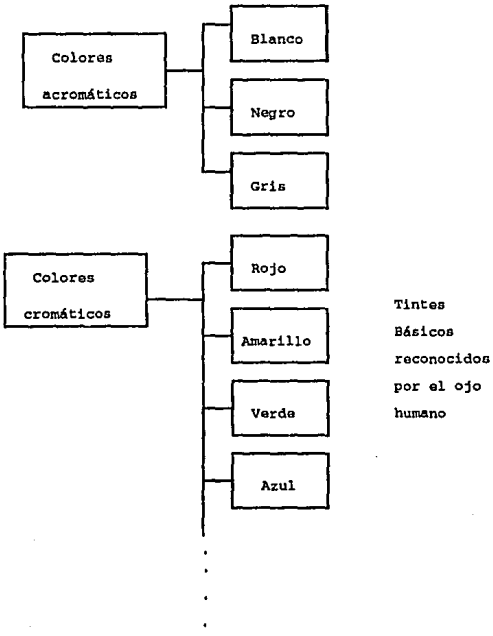


Fig.-2-11.-Los colores acromáticos no tienen tinte propio a diferencia de los colores cromáticos

Por medio de un matíz podemos distinguir un color cromático de uno que no lo es. Este tiene su más útil aplicación en las combinaciones circulares de los colores como se muestra en la fig. 2-12. Las cuales pueden ser usadas para

producir diferentes efectos emocionales, efectos psicológicos e impactos visuales.

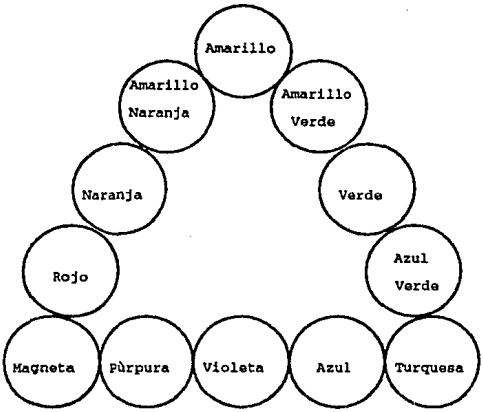


Fig.-2-12.-Combinaciones circulares de los colores.

Los matices se encuentran divididos en dos categorias:

a) Matíz cálido.- Que son el rojo, amarillo y anaranjado.

b) Matíz fresco.- Que son el verde y el azul.

Por otro lado, el valor se refiere a el brillo o claridad que presentan los colores.

El término croma, se refiere a la relativa pureza del color. Por ejemplo, el anaranjado tiene un croma muy fuerte. El croma también es llamado intensidad o saturación.

2.7.2.- DIFERENCIAS ENTRE: SOMBRAS, TONOS Y TINTES.

La sombra, es formada por la combinación de cualquier color puro con el negro. Por ejemplo el café, marrón y olivo se consideran sombras.

En lo que se refiere a los tonos,éstos se definen como la mezcla de cualquier color puro con el gris. El beige y el bronceado son tonos.

Por último, los tintes son formados de la combinación de cualquier color puro con el blanco. Son ejemplos de tintes el rosa y durazno.

Las sombras, tonos y tintes creados de un mismo matíz cálido resultan ser diferentes entre sí. Por ejemplo, el rosa (un tinte) es muy diferente al rojo (el matíz original);

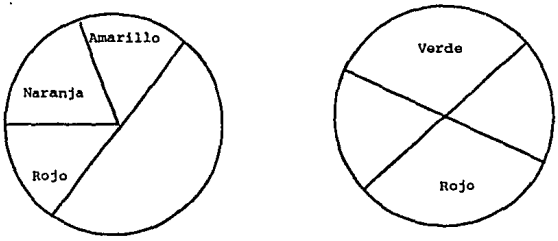
el café (una sombra) es diferente al anaranjado (el matíz original). Por otro lado sombras, tonos y tintes de un matíz fresco escencialmente tienen el mismo tipo de colores. Por ejemplo un tinte verde y una sombra verde se asemejan mucho al matíz original verde.

2.7.3.- CIRCULOS DE COLORES.

Un círculo de colores, es creado por cuatro matices básicos reconocidos por el ojo humano: Rojo, Amarillo, Verde y Azul. Usando la física de la luz, la cual es utilizada por el monitor. Los matices básicos son combinados con otros colores, para crear un amplio rango de los mismos.

2.7.4.- ARMONIA DE COLORES.

Ciertas mezclas de colores producen efectos psicológicos y emocionales. Estas combinaciones son llamadas armonias y se muestran en la fig. 2-13.



tipica armonia anàloga

tipica armonia directa

Fig.-2-13.-Diversas combinaciones de colores.

2.7.5.- ESCALAS DE COLORES.

Un simple matíz al ser combinado con diferentes cantidades de blanco y negro produce un ilimitado rango de sombras, tonos y tintes. La mezcla entre el matíz puro y el blanco forman la escala del tinte. La mezcla entre el matíz y el negro forman la escala de la sombra. La mezcla entre el blanco puro y el negro produce el gris. Las combinaciones formadas por el matíz puro y diferentes tonalidades de gris son llamadas tonos.

El ojo humano es capaz de distinguir nueve sombras distintas entre blanco y negro puros. Con Turbo C se puede lograr una imitación de estos tonos sobre la pantalla. También se puede ampliar la escala de colores de 16 a 149; Si

diferentes matices son intermezclados (diferentes porcentajes de verde y azul) aproximadamente 2400 colores pueden ser creados y desplegados en un monitor de 16 colores o modo 16.

CAPITULO TRES

ANALISIS DEL SISTEMA DE MENU DE ICONOS

3.-ANÁLISIS DEL SISTEMA DE MENÚ DE ÍCONOS.

Con el desarrollo de un sistema para presentar un menú de íconos se desea ejecutar una serie de comandos, funciones y aplicaciones de MS-DOS. Las cuales pueden ser ejecutadas al seleccionar un ícono del menú, el cual deberá presentar la acción deseada, es decir, al presentar el menú, éste cargará algunos íconos, que nos representan una función de MS-DOS. Además de poder realizar el movimiento de los mismos por medio de las flechas o ratón.

El sistema trabajará en modo real, es decir que sólo trabajará con los primeros 640 Kbytes de RAM, siendo la vista de los íconos en dos dimensiones.

A lo largo del capítulo se explicará a detalle el desarrollo del sistema y su forma de operación.

3.1.-CREACION DEL MENÚ DE ÍCONOS.

Lo primero que se debe desarrollar es la presentación de un menú que permita ejecutar alguna acción cuando se elija una opción de éste.

Para la presentación del menú se tendrán que seguir los siguientes pasos:

1.- Desplegar los íconos en pantalla.

2.- Generar el ambiente de presentación del menú, para que sea amigable con el usuario.

3.- Ejecutar la función que represente el ícono seleccionado.

4.- Regresar al paso uno, hasta seleccionar la opción que termine sesión y regresar el control al Sistema Operativo.

3.1.1.-PRESENTACION DE ÍCONOS EN PANTALLA.

Se deberá generar un arreglo de íconos, que será presentado en pantalla. El arreglo podrá ser desplegado cada vez que sea llamada la rutina que lo génere.

Para poder desplegar los íconos en pantalla, éstos tendrán que ser creados antes de presentar el menú. La función que genera los íconos será descrita más adelante.

El menú estará compuesto por íconos que nos representen la acción de crear a los mismos, además de realizar algunos

comandos de MS-DOS y al finalizar la tarea seleccionada regresar el control al menú de íconos.

3.1.2.-AMBIENTE DE OPERACION DEL MENÚ.

En este punto, se deberán desplegar las diferentes opciones que se pueden ejecutar y la forma de poder operar el sistema.

Las indicaciones deberán ser claras, para que el usuario pueda operar adecuadamente el sistema.

3.1.3.-EJECUCION DEL ÍCONO SELECCIONADO.

Los comandos del sistema operativo que se pueden ejecutar son:

Realizar una copia de archivo, borrar un archivo, ver el estado de volúmen, ejecutar una aplicación, por ejemplo trabajar con un sistema de captura de bases de datos y por último el generador de íconos. Para lograr esto basta con

posicionarce en el ícono deseado y dar un enter o click con el botón izquierdo del ratón.

3.1.4.-RESTABLECER EL MENÚ DE ÍCONOS.

En este punto, debemos llamar a la función que genera la presentación del menú y ambiente de operación , que se menciona en el punto 3.1.1, hasta seleccionar la opción que finalice la sesión, para regresar el control al Sistema Operativo.

3.2.-MOVIMIENTO DE ÍCONOS.

El movimiento de los íconos se puede realizar, siempre que se tenga un control de sus posiciones en pantalla. Existen dos formas de realizar este desplazamiento:

1. Con el movimiento de flechas.

2. Mediante el ratón.

3.2.1.-MOVIMIENTO DE ÍCONOS POR MEDIO DE FLECHAS.

Para realizar el movimiento de íconos por medio de las flechas, debemos seleccionar la opción correspondiente que se presenta en el menú, posicionarse en el ícono deseado y con las flechas moverlo a su nueva posición. Al realizar el desplazamiento del ícono sobre la pantalla, se verá el arrastre de una caja de las mismas dimensiones del ícono para ir observando el movimiento de éste y cuando se encuentre en la posición deseada se deberá confirmar el movimiento y entonces el ícono será colocado en su nueva posición dentro de la pantalla. Este procedimiento debe realizar el movimiento de cualquier ícono o reposicionar uno que haya sido desplazado de su lugar inicial.

3.2.2.-MOVIMIENTO DE ÍCONOS POR MEDIO DEL MOUSE.

El movimiento de los íconos por medio del ratón deberá seguir el mismo procedimiento de seleccionar la opción del menú y posicionarse en el ícono deseado por medio del ratón. Para hacer esto, se deberá arrastrar el ratón hasta encontrarse en el ícono deseado, presionar el botón izquierdo del ratón, y arrastrarlo sobre la pantalla, hasta la nueva posición y por último soltarlo para reposicionar el ícono.

3.3.-ANÁLISIS DEL EDITOR DE ÍCONOS.

El análisis para el desarrollo de un editor de íconos se resume en cinco pasos o etapas que son las siguientes:

Configurar el monitor a modo gráfico.

Tener la dirección de memoria de la pantalla en modo gráfico.

Implementar la función del editor de íconos.

Inicializar la matríz de pixeles.

Cargar argumentos .

Desplegar el menú del editor de íconos.

Regresar a modo texto el monitor.

3.3.1.-CONFIGURAR EL MONITOR EN MODO GRAFICO

En general, para configurar el monitor en modo gráfico se requiere hacer una llamada a las interrupciones del BIOS. Seleccionar el modo gráfico que acepte la tarjeta del

microcomputador y que cubra las necesidades de la aplicación para la que está siendo configurada.

La configuración en modo gráfico se realiza con la interrupción 16 función cero. Para poder realizar lo anterior se utiliza la función int86() que es una implementación del lenguaje "C" para habilitar las interrupciones del BIOS.

Dentro de las interrupciones se tienen varias funciones que tienen diferente propósito; en nuestro caso se utilizan las funciones 13 y 12 de la interrupción 16, para lectura y escritura de un pixel, respectivamente.

La función int86() es llamada desde la función modo() (que será implementada en el capítulo IV) para configurar el monitor a modo gráfico. La llamada a esta función se muestra a continúación:

```
int86(int num,union REGS*inregs,union REGS*outregs)
```

El primer parámetro de esta función corresponde al número de interrupción que se requiere, para los dos siguientes parámetros se tiene la unión de dos estructuras, una que guarda los valores de los registros en bytes y la otra que almacena los valores en registros de 16 bits (palabras). En inregs se pueden mandar los valores que se necesiten para cada interrupción y el valor regresado por la función se encuentra en la parte baja del registro en AX.

En el desarrollo de este sistema se utiliza el modo 16 que es un modo gráfico de 16 colores con una resolución de 640x350 pixeles para adaptadores gráficos EGA/VGA.

Por lo tanto para colocar el adaptador de video en modo 16 solo es necesario utilizar la siguiente instrucción:

```
modo(16);
```

con la acción anterior, queda configurado el adaptador de video a el modo gráfico deseado.

3.3.2.-DIRECCION DE MEMORIA EN PANTALLA.

Para obtener la dirección de memoria donde se guardará el valor del color del pixel se presenta la siguiente fórmula:

dirección del pixel=A000:0000 + (Y*80) +X/8

La cual se deriva del siguiente razonamiento:

La dirección de la RAM desde donde se comenzarán a guardar los valores de cada pixel es la **A000:0000.**

En modo 16 cada renglón sobre la pantalla contiene 640 pixeles en cada plano de bit y cada pixel corresponde a un

bit. De lo anterior se tiene que un byte de memoria contiene 8 pixeles.

Se deduce de lo anterior que un renglón requiere 80 bytes de memoria 640/8=80. Dando origen al segundo miembro de la relación citada arriba, para localizar la posición de un pixel.

Para posicionarse en la columna exacta del pixel, debemos de recorrer los "X" pixeles faltantes, sabiendó que un byte es igual a ocho pixeles, debemos de dividir la "coordena" "X" entre 8, obteniendo así la posición del byte donde se encuentra el pixel. Con el anterior razonamiento, obtenemos el tercer argumento de la relación.

Por ejemplo, si las coordenadas de un pixel son X=320 , Y=175 y de la relación **dirección del pixel = A000:0000 + (Y*80) +X/8** encontrar la dirección del pixel.

Con la operación de Y*80 obtenemos la memoria necesaria para desplazarse hasta el renglón donde se encuentra el pixel, para obtener la columna, se divide la coordenada "X" entre ocho para obtener la cantidad de bytes que se tiene que desplazar para llegar al byte en donde se encuentra el pixel. De ésta operación se obtienen la cantidad de bytes totales que hay que desplazarce en el monitor para localizar la dirección del byte donde se encuentra el pixel.

Debemos hacer notar que con la relación sólo tenemos la dirección del byte donde se encuentra el pixel, pero podría no ser el pixel que deseamos. Esto es, que al realizar la división de la coordenada "X" entre ocho el resultado puede no ser exacto, y así no estar en el pixel deseado.

Por la tanto para solucionar este problema, se tiene que generar un corrimiento de bits a la izquierda, de la siguiente forma:

1.- Se define a una variable máscara con un valor hexadecimal de 0x80:

	Hexadecimal	Decimal	Binario
mascara=	**0x80**	**128**	**10000000**

2.- Se calcula, con la fórmula antes descrita, la cantidad de bytes que se tiene que desplazar el cursor, para posicionarse en el byte donde se encuentra el pixel.

dirección del pixel = A000:0000 + Y * 80 + X/ 8

3.- Realizar un corrimiento de bits, igual al residuo de X/8 para posicionarse en el pixel correcto.

```
mascara >>= x%8
```

La instrucción anterior es un operador del lenguaje "C" que permite hacer un corrimiento de bits a la derecha, tantas veces como sea el residuo de la división.

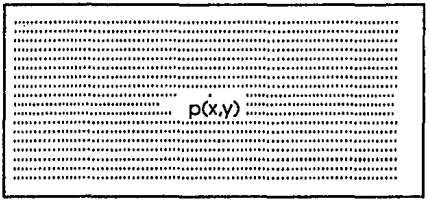


Figura.-3-1.- Matríz de pixeles.

Con el razonamiento anterior, podemos controlar la posición exacta del pixel que será activado para realizar el dibujo del ícono, como se muestra en la figura 3-1.

3.3.3.-ANALISIS DEL EDITOR DE ÍCONOS.

El editor de íconos es la parte escencial del desarrollo del Sistema de Menú de Iconos, ya que es en esta función donde se van a generar los íconos que representarán en forma gráfica un comando de MS-DOS.

Para generar un menú de íconos se requiren tres pasos fundamentales que son los siguientes:

Activar el Modo Video

Leer un Pixel de la Pantalla

Escribir un Pixel en Pantalla

3.3.3.1.-ACTIVAR MODO VIDEO

El monitor puede trabajar en dos formas, modo texto y modo gráfico. En el modo texto el monitor trabaja con carácteres y por lo general está definido con 80 columnas X 24 renglones. En modo gráfico el monitor trabaja por medio de pixeles y dependiendo de su configuración se puede tener mayor o menor nítidez de la imagen que se despliege en la pantalla. Para nuestro desarrollo se configurará el monitor en modo gráfico con una resolución de 640 X 320 pixeles. El objetivo se logra por medio de interrupciones. Y como se menciono anteriormente con la función modo(16) se configura el monitor en modo gráfico con una resolución de 640X340 pixeles.

3.3.3.2.-LEER UN PIXEL DE LA PANTALLA

Para leer un pixel de la pantalla, se requiere de hacer una llamada a las interrupciones del BIOS. Para lograr lo anterior, se puede utilizar la función de Turbo "C" con la cual se configura a modo gráfico el monitor, la función int86().

Se utilizará la interrupción 16 hexadecimal, función 13. La cual permite leer un pixel desde la pantalla. Por lo que al llamar a la función int86(), se deberán pasar los parámetros correspondientes.

3.3.3.3.-ESCRIBIR UN PIXEL EN PANTALLA

Los puertos de Entrada,Salida (E/S) son una zona física utilizada como medio de comunicación entre el CPU y un dispósitivo periférico. Los puertos se identifican por direcciones especiales de memoria de E/S de 16 bits. Para transmitir un dato, primero se envía éste al "BUS" de datos. Normalmente se utiliza cuando los servicios de interrupciones del BIOS y del DOS no proveen soporte para un dispositivo periférico; por ejemplo, si queremos enviar datos al monitor, tenemos que hacerlo a través de su puerto de E/S. El BIOS provee el acceso a los dispositivos periféricos estándar como el teclado, pantalla, unidad de disco flexibe, unidad de disco duro, impresora, monitor, etc; mientras que el DOS provee servicios adicionales que suplementan al BIOS.

3.3.3.3.1.-ESCRIBIR UN PIXEL UTILIZANDO FUNCIONES DE TURBO "C"

Para escribir un pixel en pantalla se tienen funciones en TC que son muy lentas, ya que por cada pixel que se requiera escribir sobre la pantalla se tiene que hacer una llamada a la interrupción correspondiente, habilitar el puerto, escribir el pixel y regresar para saber la posición donde se dibujará el siguiente pixel.

Las funciones de Turbo "C" utilizan la función int86() para escribir un pixel en pantalla, pasando como parámetros el número de interrupción,en nuestro caso 16, función 12, que permite escribir un pixel en pantalla.

Por lo anterior, se requiere desarrollar una función con la cual la escritura de cada pixel en la pantalla sea mucho más rápida, ya que una figura por más sencilla que sea se compone da varios pixeles.

Para desarrollar la función se tiene que dividir el valor de cada pixel en cuatro planos de bit, cada uno de los cuales tiene una porción del color del pixel que será escrito. Por lo que al juntar los cuatro planos se tiene el pixel completo.

3.3.3.3.2.-ESCRIBIR UN PIXEL DIRECTAMENTE EN PANTALLA

En el bloque de definiciones se define la función outindex como el conjunto de dos funciones outp(). En esta función se tienen que hacer dos escrituras una que inicializa el puerto y la segunda que escribe el valor en el lugar corrrespondiente. Por lo tanto en el primer parámetro se recibe el valor del registro índice y en el segundo el valor del pixel que será escrito.

Para escribir un pixel directamente en la pantalla es necesario utilizar las siguientes funciones:

La función putpoint() llama a la función outindex que a la vez esta compuesta por dos funciones outp().

En el caso de la función putpoint() en los dos primeros parámetros se tienen las coordenadas donde se escribirá el pixel, en el tercero el color y el cuarto corresponde a la acción que se requiera desarrollar sobre el pixel, los valores que puede tomar son los siguientes:

VALOR	ACCION
SOBREESCRIBIR	0
XOR	0x18
AND	8

OR 0X10

Con la operación que ya fue definida en el punto 3.3.2 :

mascara >> = x % 8

Se hace el ajuste para tener la dirección exacta donde será escrito el pixel de la siguiente forma:

La dirección donde se inicializa la variable máscara en binario es:10000000, en el caso de que el resultado de x % 8 sea diferente de cero el uno se recorre a la derecha tantas veces como el residuo de la operación lo indique y se obtienen la dirección exacta.Por ejemplo si el módulo es tres, la nueva dirección será 00010000.

A una variable se le asigna un valor, el cual está apuntando a la dirección donde inicia la memoria de video, para que no se pierda esta. Posteriormente se manda llamar a la función outindex() cuatro veces donde el primer parámetro corresponde al plano de bit y el segundo al color.

Con la instrucción *base = 1 se limpia la dirección del puerto.

Nuevamente se llama a la función outindex() ahora para inicializar los valores y la dirección de la variable máscara.

3.3.4.-INICIALIZAR MATRÍZ CON PIXELES.

El siguiente paso es el de llenar la matríz, donde se escribirá el ícono, esto se hará por medio de una función la cual por medio de dos FOR's anidados llenará la matríz de puntos en base a la variable, misma que se inicializa con el valor de 9 que corresponde al color de los pixeles.

3.3.5.-CARGAR ARGUMENTOS.

Consiste en cargar un archivo que haya sido elaborado antes de iniciar una nueva sesión con el editor de íconos y llenar la matríz con la figura correspondiente, dando opción a realizar modificaciones.

Para poder traer un archivo y cargarlo en la matríz de pixeles se debe generar una función la cual recibe como parámetro un apuntador a carácter.

Si el valor del parámetro que recibe no existe pregunta desde donde se debe cargar el ícono y lee la respuesta de el teclado por medio de la función get() de lo contrario, por medio de la función strcpy() carga el ícono deseado.

3.3.6.-INICIALIZAR MENÚ DEL EDITOR DE ÍCONOS.

Se presentará un menú, como el mostrado en la figura 3-2. Indicandonos las operaciones que se pueden realizar con el mismo, además de presentar la matríz donde se dibujará el ícono y el tamaño normal de éste. Las opciones que se podrán realizar serán las de cambiar el color del pixel para ir dibujando una figura, moverse por la matríz sin alterar el color de fondo de los pixeles, salvar un ícono en disco, cargarlo y modificarlo si se desea y una opción que finalice sesión con el Editor de Iconos.

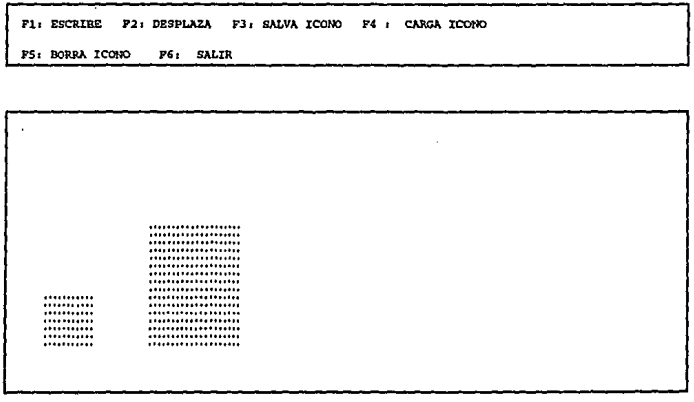


Figura.-3-2.-Presentación del editor de íconos.

3.3.7.-CONFIGURAR A MODO TEXTO Y REGRESAR EL CONTROL AL SISTEMA OPERATIVO.

Se debe regresar a modo texto la configuración del monitor para que no existan distorciones en las aplicaciones que requieran este tipo de configuración. Esto se hace sencillamente ejecutando la función modo() antes mencionada, sólo que ahora se manda como parámetro el número dos, el cual configura el monitor en modo texto.

```
MODO(2);
```

Con está acción, la máquina queda configurada en modo texto, no teniendo problemas para trabajar con aplicaciones que requieren de ésta carácterística. Una vez que se ha ejecutado la instrucción se da por terminada la sesión con el sistema.

En este capítulo se desarrollo el análisis del Editor de Iconos y del Menú de Iconos, para el siguiente capítulo se procedera a desarrollar los pseudocódigos y diagramas de flujo de las funciones que se requieren para la implementación del sistema.

CAPITULO CUATRO

DESARROLLO DEL SISTEMA DE MENU DE ICONOS

4.-DESARROLLO DEL SISTEMA DE MENÚ DE ÍCONOS.

Del análisis hecho en el capítulo 3, desarrollaremos dos módulos para la presentación de los íconos. Un módulo que nos permita generar los íconos y el otro es para presentarlos en un menú y al seleccionar uno, se ejecute un comando del sistema operativo.

Se requiere generar varias funciones que simplificarán el desarrollo del sistema, ya que éstas se utilizarán en los diferentes módulos del sistema. El siguiente tema explicará y desarrollará las funciones que se necesitarán.

4.1.-ACTIVAR EL MODO VIDEO

Para activar el modo video, necesitamos de las interrupciones del BIOS. En específico de la interrupción 16 función cero. Para realizar esto, existe en Turbo "C" una función que me permite activar una interrupción llamada int int86(). Esta genera interrupciones, dependiendo de los parámetros que se le pasen. A continuación, se describe el funcionamiento de la misma :

La sintáxis de la función int86() es de la siguiente forma:

```
int  int86( int num, /*  num es el número de interrupción */
unión REGS  *inregs, /* valor de los registros de entrada
*/unión REGS *outregs /* valor de los registros de salida */)
```

Esta función básicamente requiere de tres párametros, el número de interrupción, el número de función asociada con la interrupción y el párametro que regresa la función[1]. Además se observa que la defición de REGS es la unión de dos estructuras, una que permite accesar registros de CPU en cantidades de 16-Bits por medio de WORD-REGS y con BYTE-REGS accesar registros de 8-Bits. Por ejemplo, si se desea la interrupción 16 función 5, debemos mandar la siguiente secuencia de código:

```
unión REGS in, out;
in.h.ah = 5;  /* Se define la función */
int int86(16,&in,&out);/* El Primer párametro define la
interrupción, el 2o.el valor de la función y en
el tercero se regresa el valor de la misma */
```

Para activar el video en modo gráfico, se requiere utilizar la función int86() con los siguientes párametros:

```
int num = 16
&inregs = 0
```

El valor de num es pasado en el registro AH y la interrupción en AL regresando un valor en AX.

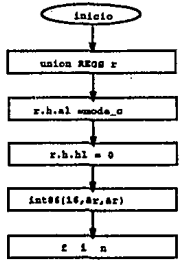
[1] No siempre se regresa un valor, ésto depende de que función se esté utilizando.

Se desarrollará una función llamada modo() la cuál requiere de un párametro que define la interrupción 16, para pasar a modo gráfico el estado del monitor, el desarrollo del pseudocódigo y diagrama de flujo se muestra a continuación:

PSEUDOCÓDIGO :

-Definir "r" como estructura unión REGS
-Asignar en "AL" el valor de la interrupción
-Asignar en "AH" el valor de la función
-Llamar a int86(16,&r,&r)
-Fin

AGRAMA DE FLUJO



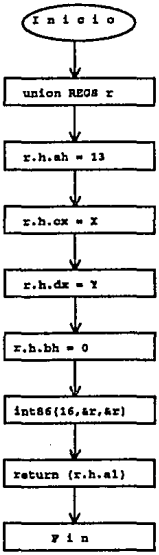
4.1.1.-LEER UN PIXEL DE LA PANTALLA.

Para leer un pixel de la pantalla se requiere del uso de la interrupción 16 función 13, que permite leer un pixel desde el monitor, para esto se tiene que pasar como párametro la posición de donde se quiere leer el pixel. A continuación se muestra el pseudocódigo y diagrama de flujo que permita realizar lo anterior:

PSEUDOCÓDIGO

```
-Definir "r" como una estructura REGS
-Especificar el valor de la función en AH = 13
-Especificar la coordenada  "X " en el registro CX
-Especificar la coordenada  "Y"  en el registro DX
-Especificar la página de video en el registro BH
-Especificar el valor del pixel en  AL
-Llamar a la función int86(16,&r,&r)
-Regresar el valor del pixel  en  AL
```

DIAGRAMA DE FLUJO



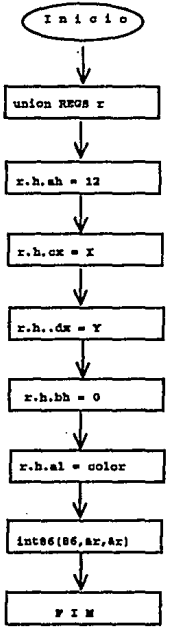
4.1.2.-ESCRIBIR UN PIXEL EN PANTALLA.

La escritura de un pixel en pantalla, como se explicó en el capítulo 3, se puede realizar por medio de la interrupción 16 función 12, Especificando el valor del pixel en el registro AL. Las coordenadas se especifican en CX y DX, la página de video en BH y el color del pixel en AL. Está función, puede trabajar en todos los adaptadores de video pero tiene el inconveniente de que es demasiado lenta. A continuación se presenta el pseudocódigo y diagrama de flujo.

PSEUDOCÓDIGO

```
-Definir "r" como una estructura REGS
-Especificar el valor de la función en AH = 12
-Especificar la coordenada  "X " en el registro  CX
-Especificar la coordenada  "Y"  en el registro  DX
-Especificar la página de video en el registro  BH = 0
-Especificar el color del pixel en AL = color
-Llamar a la función int86(16,&r,&r)
```

DIAGRAMA DE FLUJO



Como se hizo mención anteriormente, esta función es muy lenta, por la forma en que trabaja, ya que cada vez que requiera de escribir un pixel debe de realizar una interrupción y esto hace que la escritura de pixeles sea lenta. Por lo que se desarrollará una función mucho más rápida, haciendo la consideración de que sólo funcionará en

adaptadores gráficos CGA, VGA o super VGA y será utilizada en el desarrollo del Sistema de MENÚ DE ÍCONOS.

Se sabe que las grandes resoluciones de los modos de video para un EGA, VGA o SVGA requieren de mucho mayor memoria RAM que los adaptadores de video monocromáticos y CGA original. Ya que éstos pueden ser leídos y escritos directamente en la localidad de memoria que ocupa en la RAM de video. Para los modos 13 al 16 la RAM de video inicia en la localidad A000:0000.

La RAM de video de un EGA,VGA, SVGA no es accesada directamente por el programa. En su lugar, para modos gráficos, ésta se obtiene escribiendo hacia dos puertos. El primer puerto, "0xC3E", el cual se refiere al puerto índice y el segundo "0xC3F" es llamado el valor del puerto. El propósito del puerto índice es determinar la forma en como se pasará la información del puerto valor. En sí, el valor que es la salida por el puerto índice selecciona un modo específico en el adaptador. Este modo determina la forma de cómo será escrito el valor del puerto.

A continuación se presenta el pseudocódigo y diagrama de flujo de la función que escribe un píxel directamente en pantalla.

PSEUDOCÓDIGO

-Inicio
-Definir máscara para obtener posición del píxel
máscara = 0x80 = 10000000 (en binario)
-Definir variable que apuntará a la dirección de video
FAR *BASE
-Si variable x < XMIN regresar a donde fue llamada, de lo contrario:
+Si es x > XMAX regresar a donde fue llamada, de lo contrario:
++ Si y < YMIN regresar a donde fue llamado de lo contrario:
+++Si y > YMAX regresar a donde fue llamada de lo contrario obtener la posición del byte donde se quiere activar el pixel :

base = egabase + y*80L + x/8

-Realizar corrimiento de bits para posicionarse en el píxel deseado:

máscara >> x%8

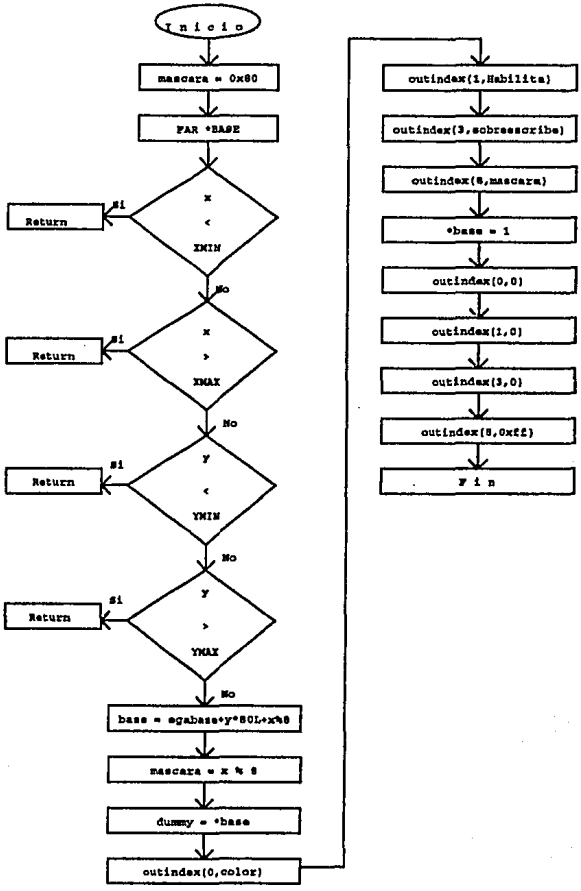
-Asignar el contenido de la dirección de base a una variable:

dummy = *base

-Cargar el color deseado
outindex(0,color)
-Habilitar escritura
outindex(1, habilita)
-Definir cual es el tipo de escritura a usar (en nuestro caso sobreescritura)
outindex(3,sobreescribe)
-Activar el pixel deseado
outindex(8,máscara)

```
-Inicializar valores por default
   *base = 1
   outindex(0,0)
   outindex(1,0)
   outindex(3,0)
   outindex(8,0xff) -> Habilita todos los pixeles
-Fin de rutina
```

DIAGRAMA DE FLUJO



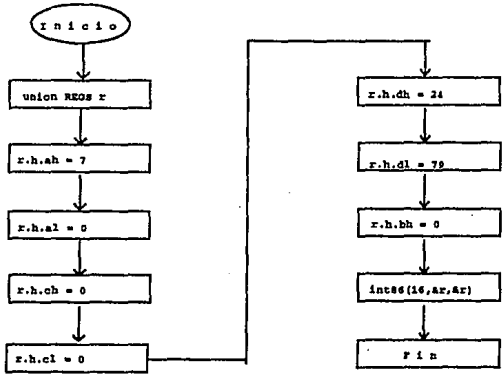
4.1.3.-LIMPIAR LA PANTALLA

Función que me permite limpiar la pantalla, ésta se realizará llamando a la interrupción 16 del BIOS, función 7. El valor del renglón superior izquierdo se especifica en el registro CH, la columna superior izquierda en CL, el valor del renglón inferior derecho en DH y el valor de la columna inferior derecha en DL. Los valores que serán pasados, estarán dados en carácteres y no en pixeles. La página de video se especifica en BH, se pasará un cero en el registro AL, indicando con esto que se quiere limpiar toda la pantalla, además de llamar a la función int86() para generar la interrupción.

PSEUDOCÓDIGO

```
-Definir "r" como una estructura REGS
-Especificar el valor de la función en AH = 7
-Indicar que se quiere limpiar toda la pantalla  AL = 0
-Especificar el renglón superior izquierdo en CH= 0
-Especificar la columna superior izquierda en CL= 0
-Especificar el renglón inferior derecho en DH=24
-Especificar la columna  inferior derecha en DH=79
-Asignar página de video  en BH=0
-Llamar a la función int86(0X10,&r,&r)
```

DIAGRAMA DE FLUJO



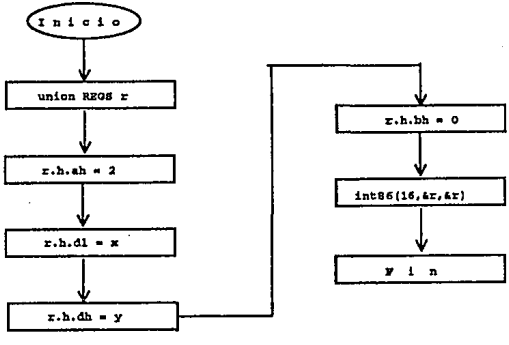
4.1.4.-POSICIONAR EL CURSOR EN UN PIXEL ESPECÍFICO

Para generar esta función, se utilizará la interrupción 16 función 2, con la que se puede posicionar el cursor en un punto específico. Las coordenadas X,Y serán pasadas en los registros DL y DH respectivamente y éstas estarán dadas en caractéres y no en pixeles. La página de video se pasará en el registro BH.

PSEUDOCÓDIGO

-Definir "r" como una estructura REGS
-Especificar el valor de la función en AH = 2
-Especificar la columna en DL = X
-Especificar el renglón en DH = Y
-Especificar la página de video en BH = 0
-Llamar a la función int86(0X10,&r,&r)

DIAGRAMA DE FLUJO



4.1.5.-LIMPIAR UN RENGLÓN

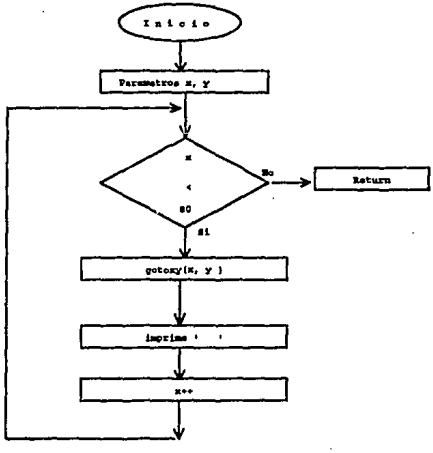
Con la función a desarrollar, se limpiará un renglón cuando se requiera de hacerlo, esto se realizará de una forma

fácil, utilizando la función que permite posicionar el cursor en un punto en específico, se borrará de está posición hasta el final de la línea.

PSEUDOCÓDIGO

```
-Inicio
-Si x < 80
      gotoxy(x, y)
      imprimir blanco
      x = x+1
-Fin
```

DIAGRAMA DE FLUJO



4.2.-DESARROLLO DEL EDITOR DE ÍCONOS

El Editor de ÍCONOs permite generar las representaciones gráficas que se muestrarán en el Menú de ÍCONOs. El desarrollo de este módulo tiene la siguiente secuencia:

Inicializar el modo video con la función modo(), presentar las opciones del editor como las de cargar ícono, modificar características del píxel, salvar a un archivo el ícono editado, borrar un ícono de disco duro, finalizar sesión del editor, activando modo texto antes de finalizar la sesión.

Posteriormente, debemos de cargar una matríz expandida donde se dibujará el ícono, Además de ir muestrando el tamaño real del ícono, para ir observando cómo quedará representado.

Ahora desarrollaremos el pseudocódigo y diagrama de flujo del Editor de ÍCONOs.

PSEUDOCÓDIGO

```
-Inicio
-Inicializar 'x' con el valor  definido en el programa
 principal; x = IX
-Inicializar 'y' con el valor definido en el programa
      principal; y = IY
```

```
-Inicializar pen con el valor definido en el programa
 principal :
      BCK_GND; pen = BCK_GND
-Llamar a la función display_icon( 0, IY)
-Llamar a la función display_grid() que muestra la
 matríz expandida :
      display_grid(  )
-Mientras la tecla presionada sea diferente de "F6"
      +Guardar el valor de pen(color del pixel) en un
       arreglo
icon.image[x-IX][y-IY] = pen
      +Escribir el valor guardado en icon.image en la
       matríz
          putpoint(x-IX, y, icon.image[x-IX][y-IY],0)
      +Almacenar en temp el valor leído de la matríz
         temp=getpoint(x+((x-IX)*XPAND,y+((y-IY)*XPAND))
      +Escribir el valor que identifique la posición del
       cursor en la matríz
      putpoint(x+((x-IX)*XPAND),y+((y-
               IY)*XPAND),HIGHLIGTH, 0)
      +Leer una tecla
          tecla = getkey()
      +Escribir el valor correspondiente que fue
       almacenado en temp:
            putpoint(x+((x-IX)*XPAND), y+((y-IY)*XPAND),
                     temp, 0)
      +Realizar una acción, dependiendo de la tecla
       leída
      switch(tecla)
      case 75
            x--
            break
      case 77
            x++
```

```
      break
case 72
      y--
      break
case 80
      y--
      break
case 71
      x--;
      y--;
      break
case 73
      x++;y--
      break
case 79
      x--;
      y++;
      break
case 81
      x++;
      y++;
      break
case 59
      pen=fore_gnd /* Activar el color de fondo */
      break
case 60
      pen=BCK_GND  /* Activar el color amarillo */
      break
case 61
      salva_icon() /* Salvar el ícono modificado*/
      break
case 62
      carga_ÍCONO()
      display_ÍCONO()
```

```
                display_grid()
                break
          case 63
                init_icon()
                display_icon(0, IY )
                display_grid( )
                break
```

-Si el valor de x es menor de IX incrementarlo en 1

```
          if (x<IX); x++
```

-Si el valor de x es mayor de las especificaciones de la matríz decrementar en uno su valor:

```
          Si (x > IX+XDIM-1);
          x--
```

-Si el valor de y es menor de IY incrementarlo en 1

```
          Si (y < IY) ;
          y++
```

-Si el valor de y es mayor de las especificaciones de la matríz decrementar en uno su valor

```
          Si (y > IY+YDIM-1);
          y--
```

-Fin

DIAGRAMA DE FLUJO

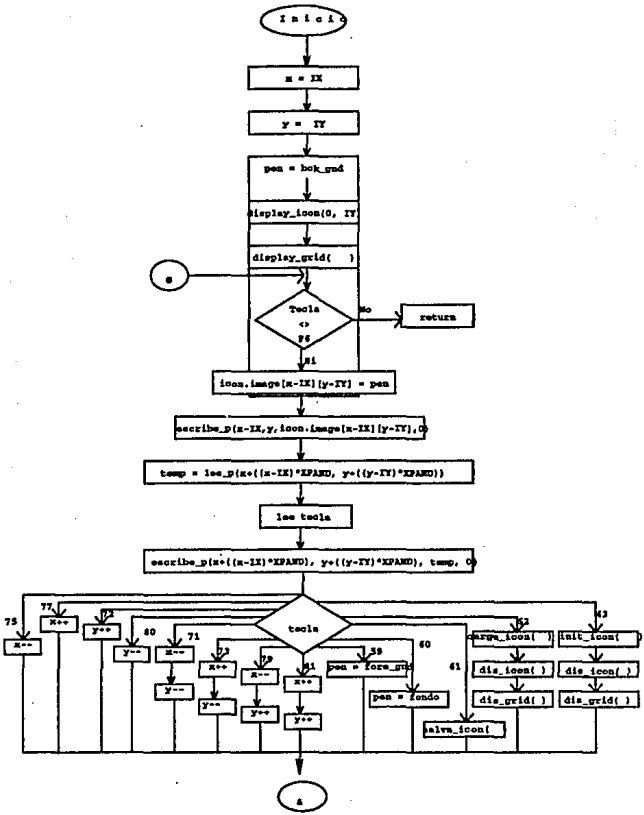
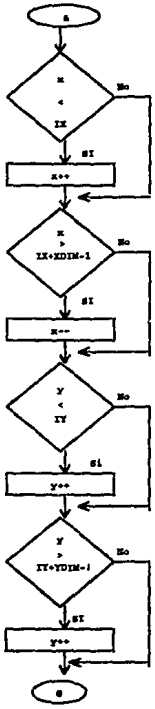


DIAGRAMA DE FLUJO



El pseudocódigo presentado, es la parte principal del desarrollo del editor de íconos. Pero como se puede observar, se auxilia de otras funciones para su operación(llama a

display_icon, display_matríz), las cuales serán desarrolladas a continuación.

4.2.1.-DESPLEGAR MATRÍZ DE PUNTOS.

La edición de los íconos se hará en una matríz cuyos valores se definen en el programa principal, ésta será una ampliación del tamaño original de los íconos. Esto con el fin de facilitar la visualización al momento de dibujar el ícono.

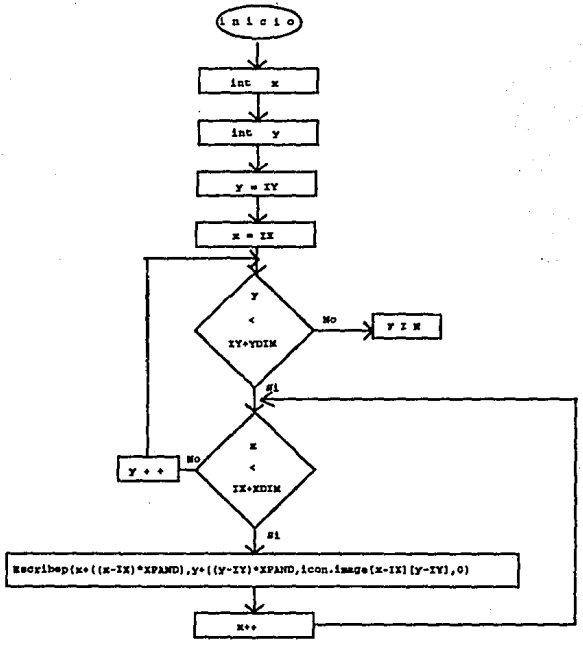
Para desplegar la matríz de puntos, se requiere de llamar a la función que escribe un pixel en la pantalla. Esto se realizará, mediante el control de dos for's anidados para controlar la posición de escritura de los pixeles.

PSEUDOCÓDIGO :

```
-Inicio
-Definir las variables 'x','y' de tipo entero
-Controlar la escritura del pixel mediante dos FOR's
 anidados
      For y = IY; y < (IY+YDIM);
         y++
         For x = IX; x < ( IX+XDIM);
            x++
```

```
            escribe_pixel(x+((x-IX)*XPAND, y+((y-IY)*XPAND,
               icon.image[x-IX][y-IY], 0)
      -Fin
```

DIAGRAMA DE FLUJO



4.2.2.-DESPLEGAR TAMAÑO ORIGINAL DEL ÍCONO

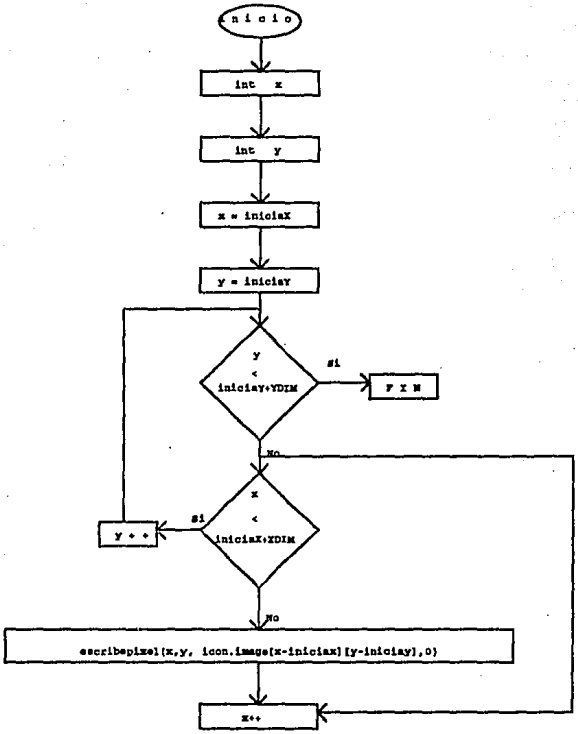
Al trabajar con el Editor de ÍCONOs, la edición se hará en matríz expandida, y simultáneamente se muestrará el tamaño original de los íconos. Con esto, se podrá observar la forma en cómo se mostrarán los íconos en el menú.

El desarrollo de éste, es similar al de la matríz expandida, recibe como parámetros los valores de 'x','y' dónde se comenzará a dibujar el ícono. Entrará en un bucle de dos For's anidados, llamará a la función escribe_pixel.

PSEUDOCÓDIGO :

```
-Inicio
-Definir las variables x,y de tipo register enteras
-Controlar la escritura del pixel mediante dos FOR's
 anidados
   For y = starty; y < starty+YDIM;
      y++
      For x = startx; x < startx+XDIM;
        x++
  escribe_pixel(x, y,icon.image[x-startx][y-starty],0)
-Fin
```

DIAGRAMA DE FLUJO



4.2.3. INICILIZAR EL ÍCONO CON EL COLOR DE FONDO DEFINIDO

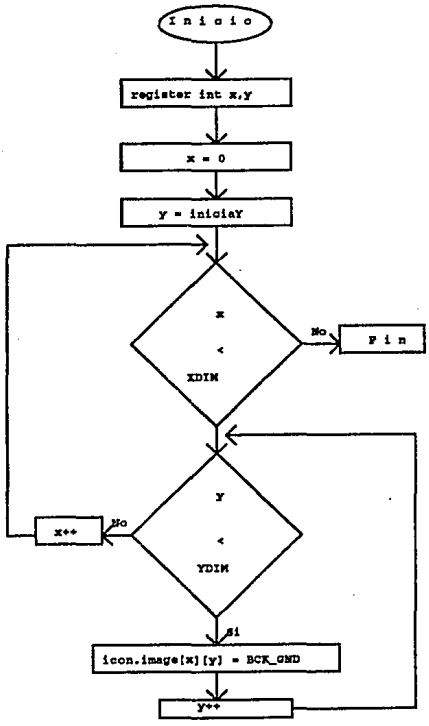
Lo que se pretende realizar con esta función, es presentar el color de los pixeles como se hayan definido previamente en la inicialización de variables. La forma de operar de esta función es similar a la de las dos anteriores, sólo que ahora se muestra el color de fondo del ícono.

A continuación se presenta el pseudocódigo y diagrama de flujo de la función.

PSEUDOCÓDIGO :

```
-Inicio
-Definir las variables x,y de tipo register enteras
-Controlar la escritura del pixel mediante dos FOR's
 anidados
 For x = 0; x < XDIM;
   x++
   For y = starty; y < YDIM;
      y++
      icon.image[x][y] = BCK_GND
-Fin
```

DIAGRAMA DE FLUJO



4.2.4.-SALVAR UN ÍCONO EN DISCO COMO UN ARCHIVO

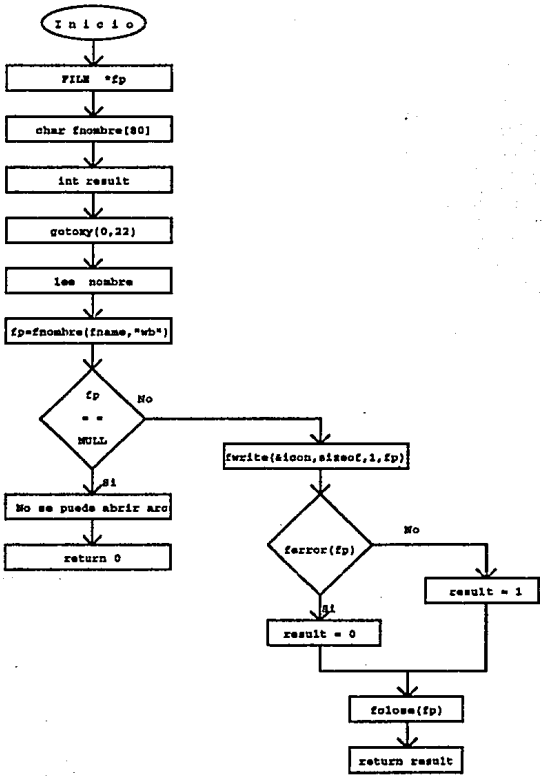
Con esta función, se salvará en disco duro y con extensión .ico el ícono que se esté trabajando en el editor.

Para salvar el ícono, se deberá preguntar el nombre con que se quiere salvar éste, en caso de que el nombre dado exista, debe de verificar si se puede sobreescribir, en caso contrario se debe mandar un mensaje de que no se puede salvar el ícono con ese nombre.

PSEUDOCÓDIGO

-Inicio
-Declarar un apuntador a archivo
-Declarar cadena para almacenar nombre de archivo
-Declarar variable entera
-Preguntar el nombre con el que se salvará el ícono
-Si el archivo no es de escritura, indicarlo con un mensaje, regresando a donde fue llamada la función
-En caso contrario salvar el ícono
-Cerrar el archivo
-Regresar a donde fue llamada la función

DIAGRAMA DE FLUJO



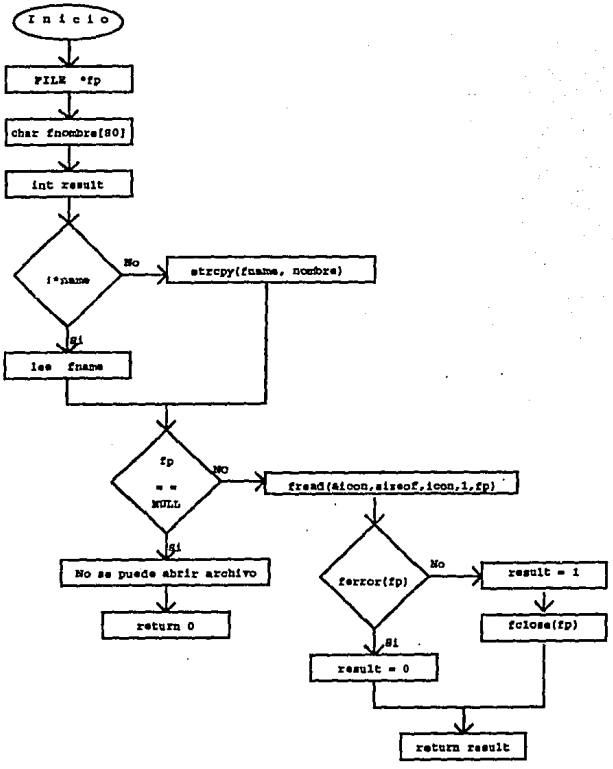
4.2.5.-CARGAR UN ÍCONO DE DISCO.

Al encontrarnos en el Editor de ÍCONOs, con esta función se cargará un ícono que ya exista y se podrán las modificaciones correspondientes.

PSEUDOCÓDIGO :

-Inicio
-Declarar un apuntador a archivo
-Declarar cadena para almacenar nombre de archivo
-Declarar variable entera
-Preguntar el nombre del ícono que se cargará
-Si el archivo no existe, indicar que no se puede cargar
-En caso contrario cargar el ícono
-Regresar al Editor de ÍCONOs, para realizar las modificaciones que se requieran
-Fin

DIAGRAMA DE FLUJO



4.2.6. DESARRROLLO DE LA FUNCION PRINCIPAL

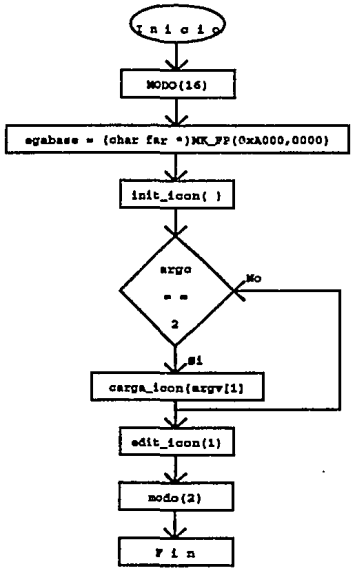
Con lo desarrollado hasta aquí, se podrá elaborar la función principal que engloba todo lo hecho. Lo que nos dará la primer parte de nuestro tema. A continuación se presenta el pseudocódigo de la función principal.

PSEUDOCÓDIGO :

```
-Inicia procedimiento
-Activar modo gráfico
      modo(16)
-Inicializar un apuntador a la memoria de video
      egabase = (char far *) 0xA0000000
-Presentar las opciones que se puedan ejecutar
-Inicializar el ícono llamando a la función
 correspondiente
      init_ÍCONO()
-Si al  ejecutar  el programa se pasa como parámetro el
 nombre de  un archivo(un ícono ya existente), cargarlo
-Llamar a la función del editar ícono
      edit_ícono()
-Configurar el video a modo texto
-Fin de Función
```

DIAGRAMA DE FLUGO

FUNCION PRINCIPAL DEL EDITOR



Con lo desarrollado, queda implementada la primera parte de nuestro tema, ya que así tendremos los íconos que nos representarán algunos comandos de MS-DOS, y sólo queda por implementar el Menú de ÍCONOs para terminar nuestro desarrollo.

4.3.-DESARROLLO DEL MENÚ DE ÍCONOS.

Al tener elaborados los íconos, con el siguiente módulo se presentarán en un arreglo. Y al seleccionar uno de ellos, se ejecutará un comando de MS-DOS.

Para desarrollar el Menú de ÍCONOs, además de utilizar las funciones desarrolladas para el Editor de ÍCONOs(escribe_pixel, leer_pixel, borrar línea, posicionar cursor, etc.), se requiere de la generación de otras funciones de las que nos auxiliaremos para el desarrollo, las cuales se explicarán a continuación.

4.3.1 GENERACION DEL MENÚ DE ÍCONOS.

El Menú de ÍCONOs, tiene tres funciones básicas: Primero, muestrará el menú de íconos, la segunda consiste en permitir al usuario la elección del ícono y por último, hacer una llamada a un comando del DOS-SHELL. Al llamar a la función, se debén de pasar como parámetros un apuntador al arreglo de íconos y el tamaño del arreglo.

PSEUDOCÓDIGO

-Inicia procedimiento
-Definir variable selección
-Definir variable ch
-Llamar a la función display_icons
-Llamar a la función que me posiciona el cursor
-Desplegar el menú de opciones
-Llamar a la función que invierte el color del ícono y su fondo

```
invert_icon( )
```

-Leer una flecha o movimiento del mouse
-LLamar a la función display_icon()
-Se entra en un case dependiendo del valor de la tecla leída:

```
+Si valor de tecla es 75 hacer (selección--)
+Si valor de tecla es 77 hacer (selección++)
+Si valor de tecla es 72 hacer :
      (selección - = NUM_ACROSS)
+Si valor de tecla es 80 hacer(selección + =
      NUM_ACROSS)
```

+Si valor de tecla es 60 llamar a la función que desplazará un ícono de su posición original
-Hasta que el valor de tecla sea igual a 59 regresar el valor de la tecla

DIAGRAMA DE FLUJO

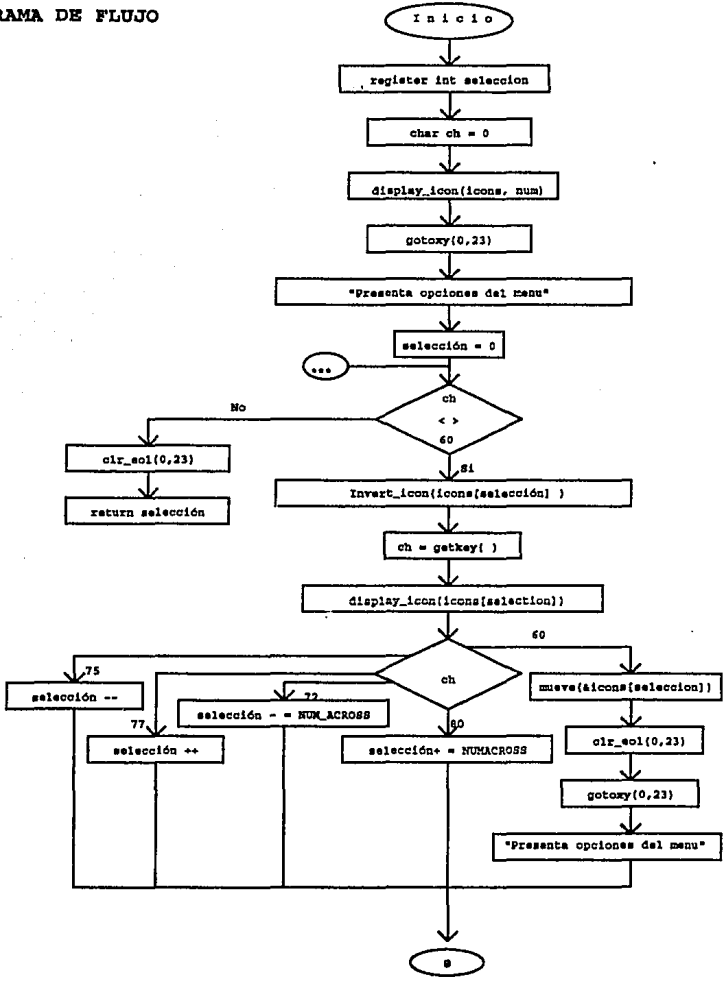
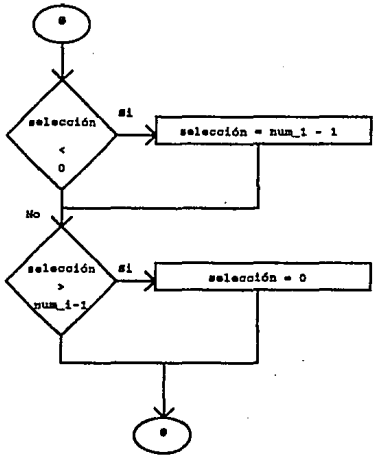


DIAGRAMA DE FLUGO



Como se puede observar en el diagrama de flujo, se declarán variables, posteriormente se llama a la función display_icons, que muestra el Menú de ÍCONOs, a continuación se muestran las opciones que se pueden realizar con el menú, se pregunta si el valor de la variable "ch" es igual a F6 si es verdadero se regresa a donde fue llamada la función, si no se llama a la función que invierte los colores del ícono, se lee una opción del teclado o ratón, se posiciona en el ícono deseado llamando a la función display_icon para entrar en un 'case' y dependiendo del valor leído, se puede posicionar en

cualquiera de los íconos del menú, todo esto hasta que se presione la tecla F6 para regresar a donde fue llamada.

Esta función, se auxilia para su operación de las funciones display_icons e invert_icon. Su desarrollo se explica en el siguiente tema.

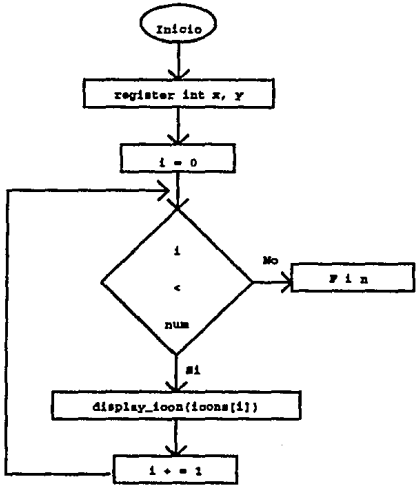
4.3.2.-FUNCION DISPLAY_ICONS.

El objetivo de esta función es desplegar todos los íconos que fueron previamente elaborados y definidos en el arreglo de íconos, auxiliandose de la función display_icon, que escribe un ícono en el monitor. A continuación se presentan el pseudocódigo y diagrama de flujo.

PSEUDOCÓDIGO

```
-Declarar variables 'x', 'y' de tipo entero
-Igualar i a cero
-Mientras i sea menor al número de íconos
   +Llamar  a la función que despliega un ícono e
    incrementar i
      'i + = 1'
-Fin de Función.
```

DIAGRAMA DE FLUJO



Se entra en un for hasta que el contador sea igual al valor definido por 'num', número de íconos que se desplegarán en el menú, llamando a la función display_icon(), que ya fué explicada en 'Editor de ÍCONOs'.

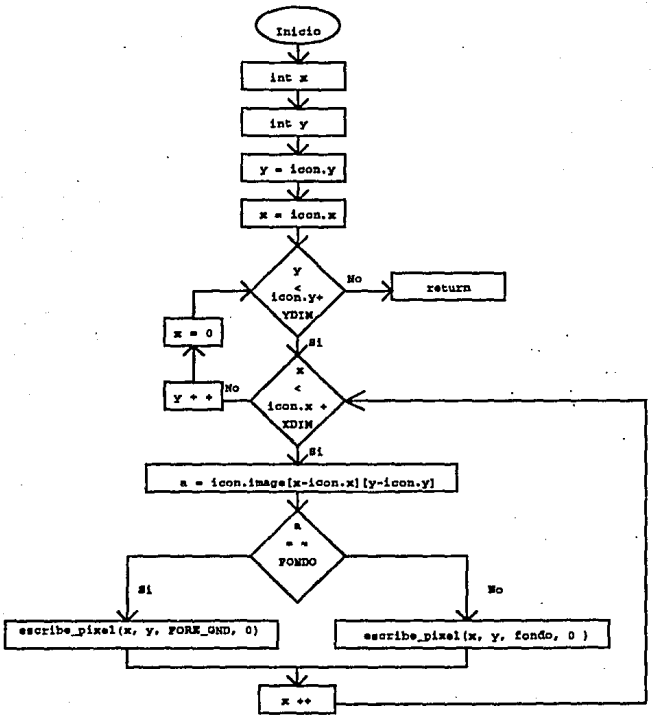
4.3.3.-FUNCION INVIERTE ÍCONO.

Para identificar en que ícono está posicionado el cursor, se invierten los colores de fondo y del ícono, como sí estuviera en video inverso, resaltando sobre los demás. La función debe de recibir como parámetro una estructura del tipo ícono.

PSEUDOCÓDIGO

- Declarar variables 'x', 'y'.
- Entrar en dos for´s anidados, siendo el más externo el que condiciona a 'y' y el interno sea 'x', para cambiar el color del pixel:
  Si el color es BCK_GND cambiarlo a FONDO
  Si el color es FONDO cambiarlo a BCK_GND
- Fin de función

DIAGRAMA DE FLUJO



4.3.4.-MOVIMIENTO DINÁMICO DE ÍCONOS

El movimiento de los íconos consiste en poder mover de posición uno de estos en cualquier área del monitor. Esto se realiza mediante el control de las coordenadas de posición. Se debe de seleccionar la opción de movimiento de íconos y por medio de las flechas o ratón se procederá a realizar el movimiento.

PSEUDOCÓDIGO

- Inicia procedimiento
- Declarar variables enteras int x, y, oldx, oldy
- Declarar variable carácter char ch
- Muestrar menú de opciones
- Asignar en x el valor del apuntador del ícono de la coordenada x:
  x= oldx = icon->x;
- Asignar en y el valor del apuntador del ícono de la coordenada y:
  y= oldy = icon->y;
- Ejecutar
  + llamada a box(x, y, x+XDIM-1, y+YDIM, 0x18)
  + leer variable ch
  + llamada a box(x, y, x+XDIM-1, y+YDIM, 0x18)
  + Entrar a un switch
    Si ch=75
      x-=5
      break
    Si ch=77

```
                x+=5
                break
          Si ch=72
                y-=5
                break
          Si ch=80
                y+=5
                break
          Si ch=71
                y-=5; x-=5
                break
          Si ch=73
                y-=5; x+=5
                break
          Si ch=79
                y+=5; x-=5
                break
          Si ch=81
                y+=5; x+=5
                break
          Si ch=59
                return ( Cuando se teclee F1)
     Hasta que ch=60
- Actualizar el valor de la dirección de ícono en x
     icon->x = x
- Actualizar el valor de la dirección de ícono en y
     icon->y = y
- Borrar la última posición del ícono por medio de dos
  FOR's  anidados
     Desde que x=oldx; x<oldx+XDIM; x++
          desde que y=oldy; y<oldy+YDIM; y++
               putpoint(x,y, 0, 0)
- Fin de procedimiento.
```

DIAGRAMA DE FLUJO

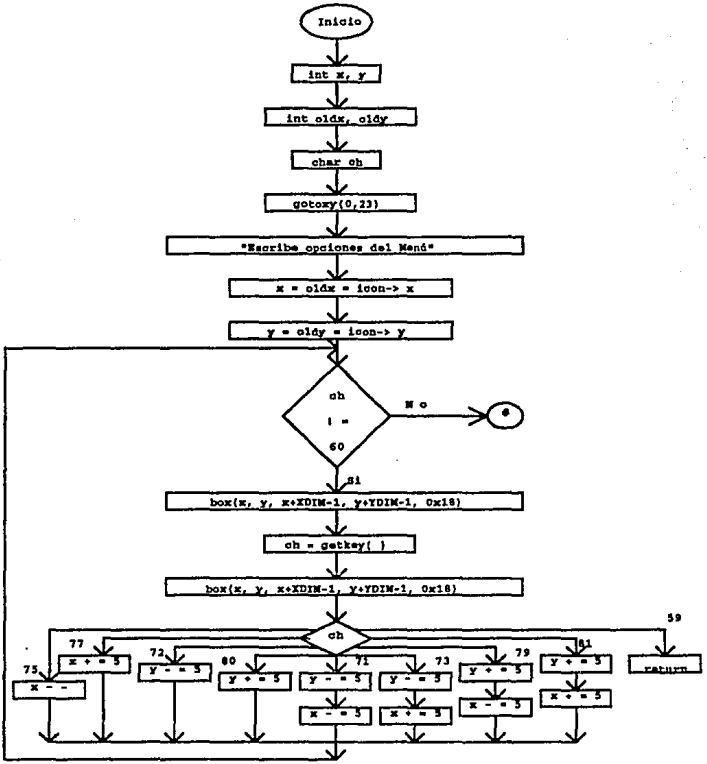
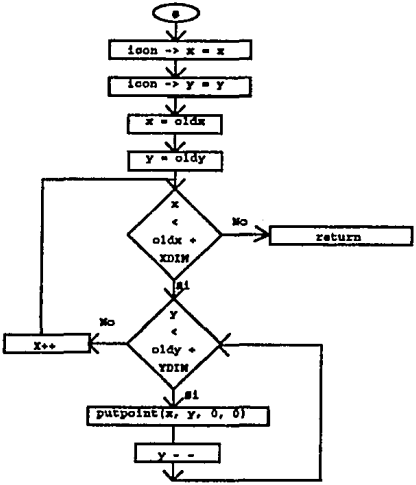


DIAGRAMA DE FLUJO



4.3.5.-DIBUJANDO UNA CAJA

Cuando se esté realizando el movimiento del ícono, lo único que se estará observando antes de confirmar su nueva posición es una caja de las mismas dimensiones del ícono, esto es porque dibujar todo el ícono en toda la trayectoria sería muy lento, ya que se tendría que estar dibujando y borrando todo el ícono.

Se deberá de pasar como parámetros iniciax, iniciay, finx, finy y how. Mediante ésta función se observará el movimiento del contorno del ícono sobre el monitor, hasta que se confirme la nueva posición, ésta será actualizada.

PSEUDOCÓDIGO

- Inicia procedimiento.
- Se recibe como párametros iniciax, iniciay, finx y finy
- Declaración de variables enteras x,y
- Se asigna x=iniciax y y=iniciay
- Mientras x<finx
  putpoint(x, finy, FORE_GND, how)
  x++
- Mientras y<finy
  putpoint(finx, y, FORE_GND, how)
  y++
- Mientras x<finx
  putpoint(x, iniciay, FORE_GND, how)
  x++

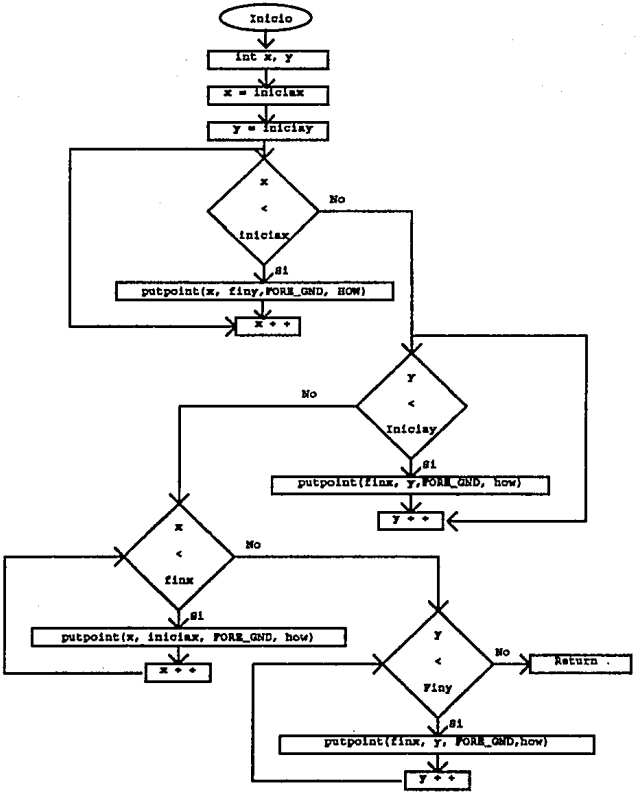
```
- Mientras y<finy
      putpoint(iniciax, y, FORE_GND, how)
      y++
- Fin de procedimiento
```

Con el anterior pseudocódigo y el diagrama de flujo que se presenta en la siguiente página, se tienen los elementos para ejecutar el editor de íconos y su menú.

Se presentarán las rutinas que inicializan el ratón, que pueden ser o no incluídas en el sistema, para que este opere con un segundo dispositivo de entrada.

Posteriormente, se desarrollarán las funciones, que realizán una llamada al sistema operativo y ejecutan un comando de MS-DOS y finalmente se presentará el desarrollo del menú de íconos.

DIAGRAMA DE FLUJO



4.3.6.-INTERFAZ PARA EL RATÓN.

Después del teclado, el dispositivo de entrada para el microcomputador es el ratón. El ratón se hizo popular con la llegada de Apple Lisa, el cual traía un ratón y una interfaz de íconos para su Sistema Operativo. Con la introducción del sistema IBM PS/2, ésta se incorporó con un puerto para el ratón como un periférico de entrada para la microcomputadora.

Antes de que el ratón pueda ser usado es necesario instalar su controlador de dispositivo. Para el ratón de Microsoft, es necesario incluír la siguiente línea en el archivo CONFIG.SYS :

```
device = mouse.sys
```

Para instalar el controlador del ratón de IBM, debe ejecutarse el progama MOUSE.COM. Para este ratón, se debe de situar la línea muestrada a continuación en el archivo AUTOEXEC.BAT :

```
mouse
```

Una vez instalado el controlador del ratón, cuando se mueva el este o pulse un botón es generada una interrupción la 33H en específico, El controlador del ratón procesa la interrupción, seleccionando las variables internas apropiadas

y devuelve el control, debido a que la interrupción es generada sólo cuando el ratón cambia de estado, sí esté permanece inactivo, no tiene ninguna interacción en el comportamiento de la computadora.

De igual forma que un cursor es asociado con el teclado, otro, está asociado con el ratón. Las rutinas en la Biblioteca del Ratón de Microsoft definen una forma de cursor por defecto: una flecha en modo gráfico y un rectángulo en modo texto. El cursor indica la posición actual en pantalla. Igual que con el cursor del teclado, el cursor del ratón puede ser activado o desactivado. Normalmente está activado sólo cuando el ratón está funcionando. En otro caso, está desactivado, es decir, no está interaccionando con la aplicación.

Aunque físicamente separado, puede pensarse en el ratón como enlazado a la pantalla, porque el controlador del ratón mantiene automáticamente contadores que indican dónde está posicionado el cursor del ratón. Cuando se mueve el ratón, el cursor se mueve automáticamente a través de la pantalla en la misma dirección del ratón.

La distancia recorrida por el ratón es medida en "mickeys". Un mickey equivale a 1/200 de pulgada.

4.3.7.-PANTALLA VIRTUAL Y/O REAL.

Las rutinas del Ratón de Microsoft operán con una *pantalla virtual*, con las dimensiones en pixel, que puede ser distinta de la pantalla física real. Cuando el ratón se mueve, los contadores de posición del ratón son actualizados. Cuando el cursor aparece en pantalla, las coordenadas del cursor virtual están proyectadas sobre las coordenadas de la pantalla real.

4.4.-IMPLEMENTACION DE RUTINAS DEL MOUSE

Las rutinas que realizarán la interface con el ratón, se pueden implementar en dos formas:

-Usando las Funciones de Biblioteca del Ratón.

-Utilizando las Interrupciones del BIOS(Interrupción 33H)

4.4.1.-FUNCIONES DE BIBLIOTECA DEL RATÓN.

Para realizar la interface con el ratón con este método, se debe de tener instalada la librería Mouse.lib, que está incluída en "*La Guía de Referencia del Programdor del Ratón de Microsoft*" con el disco que contiene la librería.

Las rutinas, son llamadas mediante una simple función usando el número de la función específica de ratón que deseamos llamar. El nombre real de las funciones del ratón depende del modelo de memoria con el que se esté compilando. Usar **cmouses()** para el modelo pequeño, **cmousec()** para el compacto, **cmousem()** para el modelo medio y **cmousel()** para el modelo grande o muy grande (éste no es soportado por el módelo muy pequeño).

Aquí se utilizará el modelo pequeño para la realización de las rutinas del mouse.

El formato para las funciones **cmouses()** es :

```
void cmouses(fnum, arg2, arg3, arg4);

int *fnum, *arg2, *arg3, *arg4;
```

Donde fnum es el número de la función del ratón que se quiere llamar. Los otros parámetros contienen información requerida por una función determinada. La función cmouses() devuelve los resultados en los parámetros.

Microsotf define treinta funciones de ratón, pero sólo se presentarán las necesarias para nuestro desarrollo. explicando a continuación el desarrollo de las mismas.

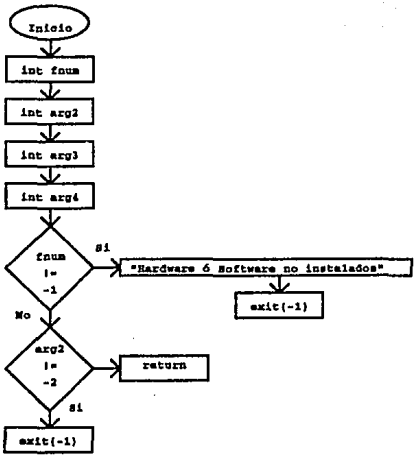
4.4.2.-INICIALIZAR EL ESTADO DEL MOUSE.

La función "0" inicializa el ratón. Sitúa el ratón en el centro de la pantalla con el cursor desactivado. Devuelve el número de botones que tiene el ratón en arg2. En el regreso, fnum puede ser "0" si el ratón y el software no están instalados, y "-1" si lo están. Su pseudocódigo y diagrama de flujo son muestrados a continuación:

PSEUDOCÓDIGO

```
-Inicia procedimiento
-Inicializa variables enteras fnum, arg2, arg3, ar4.
-Inicializa fnum con cero
-LLama a la función  cmouses con los parámetros
 &fnum,&arg2, &arg3 y &arg4.
-Si el  valor  de  fnum,  regresado  por  la  función
 cmouses es diferente de "-1", el hardware o software no
 está instalado
      exit(-1)
- Si el valor de  arg2 es difernte de 2, "Se requiere de
  un ratón  con dos botones"
      exit(-1)
-Fin de Función
```

DIAGRAMA DE FLUJO



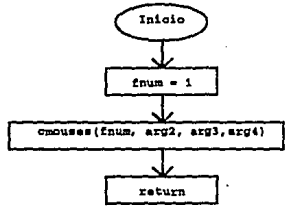
4.4.3.-VISUALIZAR EL CURSOR

La función "1" de las definidas por Microsoft para trabajar con el ratón, visualiza el cursor de esté en la pantalla. No devuelve valores, como se observa en el pseudocódigo y diagrama de flujo muestrados a continuación.

PSEUDOCÓDIGO

-Inicia procedimiento
-Define fnum como entero
-Inicilaiza a fnum con "1"
-LLama a la función cmouses(&fnum, &atg2, &arg3, arg4)
-Fin de Procedimiento

DIAGRAMA DE FLUJO



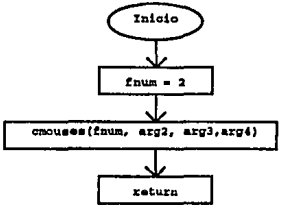
4.4.4.-BORRAR CURSOR

La función 2 hace desaparecer el cursor de la pantalla. No devuelve valores. Esta función al igual que la anterior, no regresa ningún valor.

PSEUDOCÓDIGO

-Inicia procedimiento
-Define fnum como entero
-Inicilaiza a fnum con "1"
-LLama a la función
   cmouses(&fnum, &atg2,&arg3, arg4)
-Fin de Procedimiento

DIAGRAMA DE FLUJO



4.4.5.-LEER ESTADO DE BOTÓN Y POSICIÓN DEL CURSOR

La función 3 devuelve el estado de los botones en arg2. La posición virtual horizontal del cursor está en arg3 y la posición virtual vertical del cursor está en arg4.

El estado de los botones está codificado en arg2 con los bits 0 y 1. Cuando es colocado el bit 0, el botón izquierdo ha sido pulsado; cuando es colocado el bit 1, el botón derecho ha sido pulsado. Cuando un bit no está, es asociado con que ningún boton ha sido pulsado.

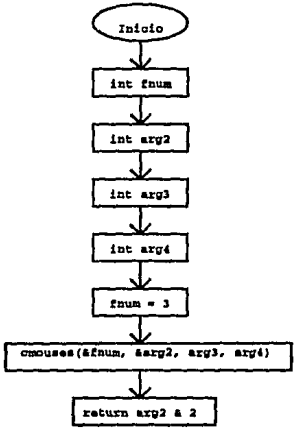
4.4.6.-DETEMINANDO SI EL BOTÓN DERECHO ES PRESIONADO.

Regresa un valor lógico verdadero si es presionado el boton derecho.

PSEUDOCÓDIGO

```
-Inicia procedimiento.
-Define variables enteras
 fnum, arg2, arg3, arg4.
-Inicializa fnum con el valor de la función fnum = 3.
-LLama a la función cmouses( )
-Regresa un valor lógico .AND. de "arg2 & 2"
-Fin
```

DIAGRAMA DE FLUJO



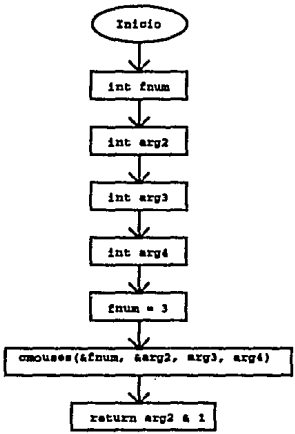
4.4.7.-DETERMINANDO SI EL BOTÓN IZQUIERDO ES PRESIONADO

Regresa un valor lógico verdadero si es presionado el boton izquierdo.

PSEUDOCÓDIGO

-Inicia procedimiento.
-Define variables enteras fnum, arg2, arg3, arg4.
-Inicializa fnum con el valor de la función fnum = 3.
-LLama a la función cmouses( )
-Regresa un valor lógico .AND. de "arg2 & 1"
-Fin de procedimiento.

DIAGRAMA DE FLUJO



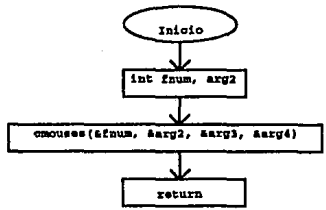
4.4.8.-ACTUALIZAR LOS VALORES DE POSICIÓN DEL RATÓN.

Se actualizan los valores de posición del ratón, si el mismo ha tenido un movimiento, no se regresa ningún valor.

PSEUDOCÓDIGO.

-Inicia procedimiento.
-Define variable entera fnum y arg2
-Inicializa variable fnum = 4.
-LLamar a función
   cmouses(&fnum, &arg2, &arg3, &arg4).
-Fin de proceso.

DIAGRAMA DE FLUJO



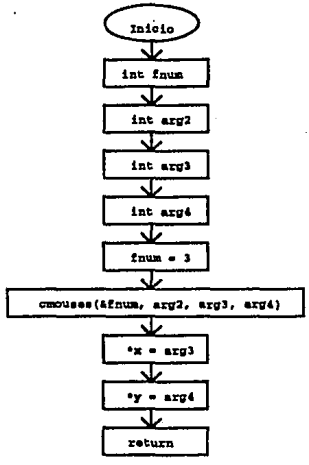
4.4.9.-ESTABLECER POSICIÓN DEL CURSOR.

La función "3" establece la posición del cursor del ratón. El valor del arg3 determina la posición horizontal, y el valor de arg4 establece la posición vertical. Se debe asegurar que sólo son válidos los valores dentro del rango de la pantalla virtual utilizada.

PSEUDOCÓDIGO

-Inicia procedimiento.
-Recibir como parámetros apuntadores a variables x,y.
-Definir variables enteras fnum, arg2, arg3, arg4.
-Inicializar la variable fnum = 3.
-LLamar a la función
    cmouses(&fnum, arg2, arg3,arg4).
-Asignar el valor de las posiciones vertical horizontal en las variables:
    *x = arg3; *y = arg4.
-Return *x, *y.

DIAGRAMA DE FLUJO



4.4.10.-INDICADOR DE MOVIMIENTO

La función "11" devuelve los contadores verticales y horizontales desde la última llamada a ésta, esto es, las distancias verticales y horizontales desplazadas. También inicializa sus registros contadores internos a "0". El contador vertical es devuelto en arg3 y el contador horizontal en arg4. Esto significa que si el ratón no ha sido movido desde la última llamada, ambos contadores, el vertical y el horizontal, estarán a 0. Si algún contador, o ambos, tienen un valor distinto de "0", el ratón ha tenido movimiento.

Un contador vertical positivo significa que el ratón tenido movimiento hacia abajo. Un contador negativo indica que el ratón se a desplazado hacia arriba. Un valor positivo horizontal significa que el cursor se ha desplazado a la derecha. Un valor negativo del contador indica que el ratón se ha movido a la izquierda.

PSEUDOCÓDIGO

```
-Inicia procedimiento
-Definir variables enteras fnum, arg2, arg3, arg4.
-Inicializar fnum = 11
-LLamar a función cmouses(&fnum, &arg2, &atg3, &arg4)
-Fin
```

4.4.11.-ACTUALIZANDO EL RADIO DE MICKEY.

Por default, el ratón se debe mover 8 mickey sobre la pantalla para mover 8 pixeles(un byte) horizontalmente y 16 mickey sobre la pantalla para mover 8 pixeles(un byte) verticalmente. El radio entre el número de mickeys que mueve el ratón en el área de trabajo correspondiente para mover un número de pixeles en la pantalla es llamado ***"radio mickey"***. Esto es controlado por medio de la función 15

4.5.-APLICACION DE LOS ÍCONOS BASADOS EN DOS-SHELL

Con las funciones que se desarrollarán en esta sección, se realizarán algunos de los comandos del sistema operativo, como se explicó en el capítulo 3. Se ejecutarán al seleccionar uno de los íconos del menú, para posteriormente regresar al menú, hasta que se elija uno que finalice la sesión con el sistema.

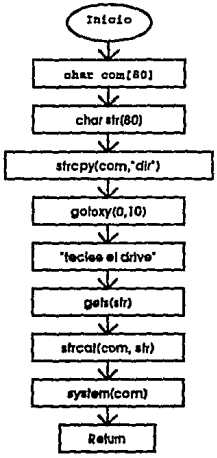
4.5.1.-FUNCIÓN "DIR" DEL DOS-SHELL

Esta función, nos permitirá realizar el "DIR" del sistema operativo desde el Menú de ÍCONOs, para observar el directorio del drive que se indique y posteriormente regresar al menú.

PSEUDOCÓDIGO

-Declarar variables com[80] y str[80]
-Copiar el contenido de str[80] en com[80]
-Pedir el Drive a desplegar su contenido .
-Concatenar el Drive leído con com[80]
-Hacer una llamada al sistema operativo por medio de system() para realizar el comando "DIR".
-Regresar el control al Menú de ÍCONOs.

DIAGRAMA DE FLUJO



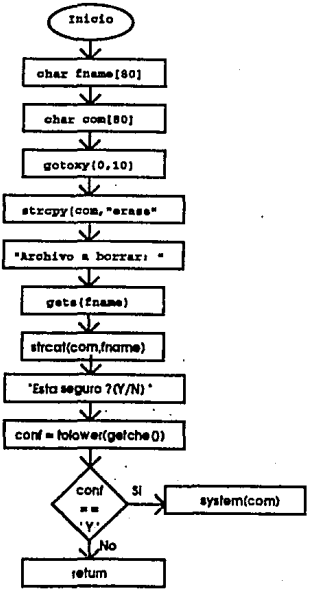
4.5.2.-FUNCIÓN PARA BORRAR UN ARCHIVO

Con ésta función, se podrá borrar un archivo del drive que se indique y regresar el control al Menú de ÍCONOs.

PSEUDOCÓDIGO

-Declarar variables fname[80] y com[80]
-Concatenar la "erase" a com[80]
-Pedir el nombre del archivo a borrar
-Concatenar el nombre del archivo a com[80]
-Confirmar la acción
->Si es aceptada relizar la operación por medio de system pasando como parámetro a "com[80]"
-> Si la respuesta es "No", regresar al Menú de íconos

DIAGRAMA DE FLUJO



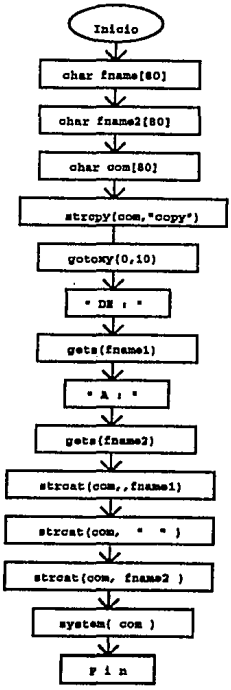
4.5.3.-COPIAR UN ARCHIVO

Con ésta función, se hará una copia de un archivo, desde el Menú de ÍCONOs, auxiliándose de la función system(), para hacer una llamada al Sistema Operativo y posteriormente regresar al menú.

PSEUDOCÓDIGO

```
-Declarar variables fnom1[80], fnom2[80], com[80]
-Copiar la cadena "copy" en "com[80]"
-Pedir el nombre del archivo a copiar en "fnom1[80]"
-Pedir el nombre del archivo destino en "fnom2[80]"
-Concatenar fnom1[80] en "com[80]"
-Concatenar " " en "com[80]
-Concatenar "fnom2[80]" en "com[80]"
-Realizar la copia llamando a system() pasando como
 parámetro a "com[80]"
-Regresar el control al Menú de ÍCONOs
```

DIAGRAMA DE FLUJO



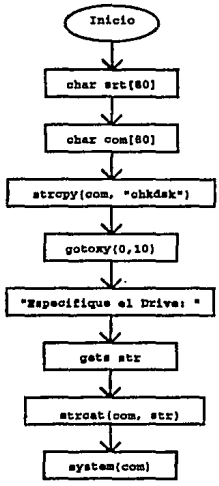
4.5.4.-FUNCIÓN "CHKDSK" DE DOS-SHELL

Se realizará un chequeo de un drive, con el comando del Sistema Operativo "CHKDSK", desde el Menú de ÍCONOs, por medio de la función "System()".

PSEUDOCÓDIGO

```
-Declarar variables str[80], com[80]
-Realizar una copia de la cadena "chkdsk" en "com[80]"
-Pedir el drive a checar en "str[80]"
-Concatenar la cadena "str[80]" con "com[80]"
-Ejecutar el comando por medio de la función system()
 pasando
 como parámetro a "com[80]"
-Regresar el control al Menú de ÍCONOs
```

DIAGRAMA DE FLUJO



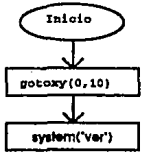
4.5.5.-FUNCIÓN "VER" DEL DOS-SHELL

Se observará la versión del sistema Operativo en uso, mediante está función, desde el Menú de ÍCONOs.

PSEUDOCÓDIGO

-Posicionarse en un punto específico de pantalla
-Ejecutar el comando "ver" medio de la función system(), pasando como parámetro la cadena "ver"
-Regresar al Menú de ÍCONOs

DIAGRAMA DE FLUJO



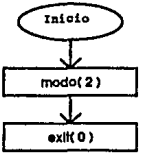
4.5.6.-FIN DE SESION

Cuando se seleccione el ícono que ejecute esta función, se dará por terminada la sesión con el Menú de íconos y se regresará el control al Sistema Operativo.

PSEUDOCÓDIGO

- Llamar a la función modo(), para activar el monitor a modo texto.
- Ejecutar la función exit() con el valor "0", para regresar el control al Sistema Operativo.

DIAGRAMA DE FLUJO



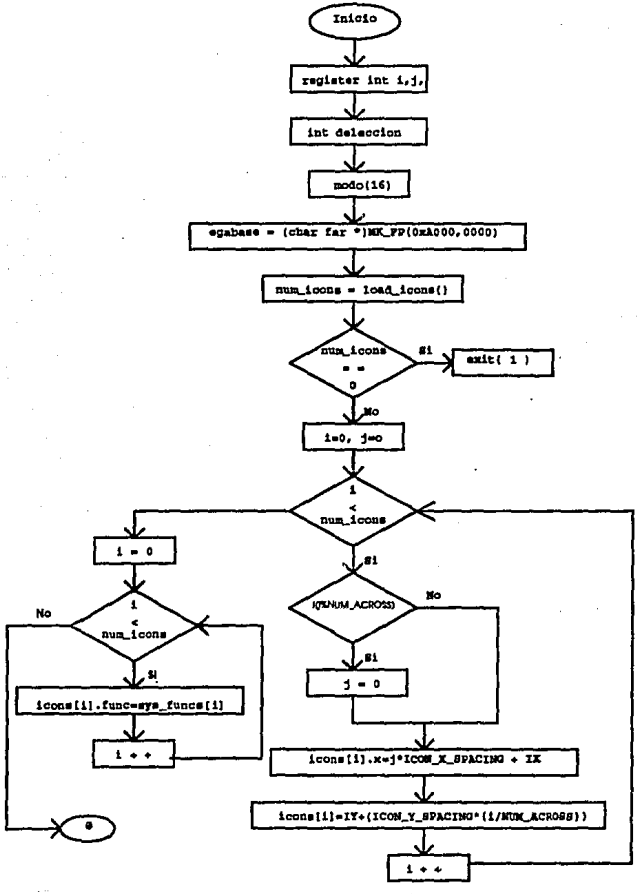
4.5.7.-FUNCIÓN PRINCIPAL DEL "MENÚ DE ÍCONOS"

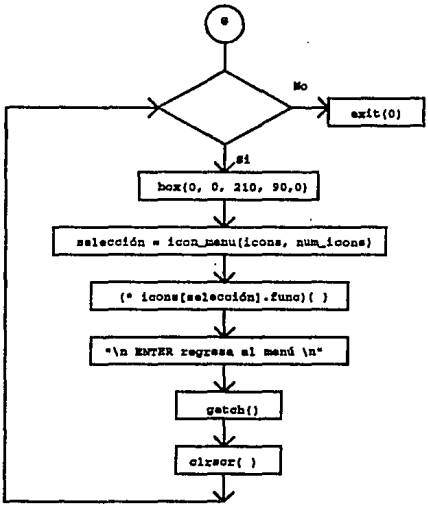
Las funciones previamente desarrolladas, serán llamadas en el cuerpo de la función principal del Menú de ÍCONOs.. Quedando completa la segunda parte de nuestro tema, muestrando su pseudocódigo y diagrama de flujo a continuación.

PSEUDOCÓDIGO

```
-Definir variables
-Inicializar a modo gráfico el video
-Almacenar el valor de la página de video en una
 variable
-Asignar en una variable el número de íconos del menú.
-Si la variable que contiene el número de íconos es cero
  regresar el control al Sistema Operativo y si no:
      +Inicializar i,j con cero
      +Mientras i < num_icons
      **Si el residuo de j%NUM_ACROSS es diferente de
           cero,entonces j= 0 en caso contrario j=
          j%NUM_ACROSS
```

DIAGRAMA DE FLUJO





Como se observa en el diagrama de flujo, la función principal llama a las funciones que fueron diseñadas previamente, observándose que en este tema, también se busca aplicar la programación estructurada.

CONCLUSIONES

El objetivo del tema desarrollado, como se mencionó al inicio de el mismo, fué el de mostrar la forma en como se trabaja el video en modo gráfico, además de integrar en nuestro desarrollo al mouse como un dispositivo de entrada. Todo ésto motivado por la curiosidad de investigar como operan los paquetes que trabajan bajo el ambiente windows(Excel, Winword, Harvard Graphics, entre otros) y el software que requieren para operar en modo gráfico.

Considerando que se buscaba desarrollar un tema, donde se pudieran aplicar de una u otra forma todos los conocimientos adquiridos en esta "Honorable Facultad de Ingeniería", tanto a nivel software como hardware; nos sentimos satisfechos de lo desarrollado, ya que al investigar y buscar información hemos retroalimentado los conocimientos adquiridos y en otros casos despejado las dudas que se tenían antes de iniciar esta empresa.

En su mayoría, el desarrollo del tema se realizó a nivel software, pero se tuvo que investigar sobre la operación del hardware para poder realizar la progamación que lo controlará.

Teniendo presente que lo desarrollado en esta tesis trabaja en modo gráfico de "640 x 350" pixeles, y en dos dimensiones. Se puede concluir que el objetivo ha sido alcanzado con lo presentado aquí. Considerando los siguientes puntos de nuestra meta fijada:

- Los diferentes tipos de video con que se puede trabajar y las resoluciones de éstos.

- Entender el funcionamiento de las interrupciones, para facilitar el uso y control de los dispositivos periféricos con que cuenta la microcomputadora.

- Utilizar las rutinas del ratón para aplicaciones de DOS y aplicaciones que trabajen en modo gráfico, donde es más común su aplicación.

- El manejo y aplicación de los apuntadores con archivos, ya que ésta es una herramienta de programación muy poderosa, cuando se le da una aplicación correcta.

- El manejo de mapas de memoria para las diferentes aplicaciones que se realizan en una PC.

- La investigación sobre las diferentes técnicas para dibujar cualquier tipo de gráficos con el modo de video adecuado.

- Realizar una programación en Lenguaje "C", aplicando todas las herramientas y potencialidad de programación que éste nos da.

Como en todo ciclo de vida de un sistema, en éste queda abierta la posibilidad de generar una nueva versión del mismo que permita, por ejemplo, hacer un menú de íconos dinámico, es decir, que exista la posibilidad de agregar nuevos íconos en el menú desde los dispositivos de entrada como el teclado y ratón; además de realizar las imágenes de los íconos en tercera dimensión, como una mejora a lo ya desarrollado.

La Universidad Nacional Autónoma de México através de la Facultad de Ingeniería, nos proporciona las herramientas técnicas y humanísticas para interrelacionarnos con las diferentes áreas de trabajo (Medicina, Arqitectura, Química, Areas Administrativas, Pedagogía, etc.) para analizar los problemas a que se enfrenten y así podamos interactuar con éstas para ayudarles a solucionarlos de la forma más óptima.

Sólo nos queda agradecer a los que participamos en este trabajo a nuestra apreciada Facultad de Ingeniería y al grupo de profesores que la conforman, todos los conocimientos que nos dieron y la formación profesional que hicieron de nosotros, para poder enfrentar los retos que se presenten en nuestra vida profesional y resolverlos en forma adecuada y eficiente, siempre buscando las mejores alternativas de solución en beneficio de la comunidad.

BIBLIOGRAFIA:

"C" Guía para usuarios expertos.
Herbert Schildt.
McGraw- Hill, 1990.

Enciclopedia del lenguaje "C".
Francisco Javier Ceballos.
Addison-Wesley, Iberoamericana.

Programación en Turbo "C", segunda edición.
Herbert Schildt.
Borland-Osborne, McGraw- Hill.

Programación en lenguaje "C".
Herbert Schildt.
McGraw- Hill.

Born to code in "C".
Hebert Schildt
Osborne McGraw-Hill.

Lenguaje "C", programación avanzada.
Herbert Schildt.
McGraw-Hill.

Advanced "C" programming for displays.
Marc J. Rochkind.
Prentice Hall.

High-Performance Graphics in "C"
Animation and Simulation.
Lee Adams.
Windcrest

Computer Graphics Systems and Concepts.
Salomon, Mel Slater.

La Guía de Referencia del Programador
del ratón de Microsoft, 1990.
Mouse.lib

80386/80286 Programación en Lenguaje Ensamblador.
Willian H. Murray, Chris H. Pappas
McGraw-Hill, 1990.

Avanced Assembly Languaje.
The Peter Norton Library.
Holzner
Brady.